

OFFICIAL MICROSOFT LEARNING PRODUCT

10990C

Analyzing Data with SQL Server Reporting Services

MCT USE ONLY. STUDENT USE PROHIBITED

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links may be provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

© 2018 Microsoft Corporation. All rights reserved.

Microsoft and the trademarks listed at

<https://www.microsoft.com/en-us/legal/intellectualproperty/trademarks/en-us.aspx> are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners

Product Number: 10990C

Part Number (if applicable): X21-64452

Released: 02/2018

MICROSOFT LICENSE TERMS MICROSOFT INSTRUCTOR-LED COURSEWARE

These license terms are an agreement between Microsoft Corporation (or based on where you live, one of its affiliates) and you. Please read them. They apply to your use of the content accompanying this agreement which includes the media on which you received it, if any. These license terms also apply to Trainer Content and any updates and supplements for the Licensed Content unless other terms accompany those items. If so, those terms apply.

**BY ACCESSING, DOWNLOADING OR USING THE LICENSED CONTENT, YOU ACCEPT THESE TERMS.
IF YOU DO NOT ACCEPT THEM, DO NOT ACCESS, DOWNLOAD OR USE THE LICENSED CONTENT.**

If you comply with these license terms, you have the rights below for each license you acquire.

1. DEFINITIONS.

- a. "Authorized Learning Center" means a Microsoft IT Academy Program Member, Microsoft Learning Competency Member, or such other entity as Microsoft may designate from time to time.
- b. "Authorized Training Session" means the instructor-led training class using Microsoft Instructor-Led Courseware conducted by a Trainer at or through an Authorized Learning Center.
- c. "Classroom Device" means one (1) dedicated, secure computer that an Authorized Learning Center owns or controls that is located at an Authorized Learning Center's training facilities that meets or exceeds the hardware level specified for the particular Microsoft Instructor-Led Courseware.
- d. "End User" means an individual who is (i) duly enrolled in and attending an Authorized Training Session or Private Training Session, (ii) an employee of a MPN Member, or (iii) a Microsoft full-time employee.
- e. "Licensed Content" means the content accompanying this agreement which may include the Microsoft Instructor-Led Courseware or Trainer Content.
- f. "Microsoft Certified Trainer" or "MCT" means an individual who is (i) engaged to teach a training session to End Users on behalf of an Authorized Learning Center or MPN Member, and (ii) currently certified as a Microsoft Certified Trainer under the Microsoft Certification Program.
- g. "Microsoft Instructor-Led Courseware" means the Microsoft-branded instructor-led training course that educates IT professionals and developers on Microsoft technologies. A Microsoft Instructor-Led Courseware title may be branded as MOC, Microsoft Dynamics or Microsoft Business Group courseware.
- h. "Microsoft IT Academy Program Member" means an active member of the Microsoft IT Academy Program.
- i. "Microsoft Learning Competency Member" means an active member of the Microsoft Partner Network program in good standing that currently holds the Learning Competency status.
- j. "MOC" means the "Official Microsoft Learning Product" instructor-led courseware known as Microsoft Official Course that educates IT professionals and developers on Microsoft technologies.
- k. "MPN Member" means an active Microsoft Partner Network program member in good standing.

- l. "Personal Device" means one (1) personal computer, device, workstation or other digital electronic device that you personally own or control that meets or exceeds the hardware level specified for the particular Microsoft Instructor-Led Courseware.
- m. "Private Training Session" means the instructor-led training classes provided by MPN Members for corporate customers to teach a predefined learning objective using Microsoft Instructor-Led Courseware. These classes are not advertised or promoted to the general public and class attendance is restricted to individuals employed by or contracted by the corporate customer.
- n. "Trainer" means (i) an academically accredited educator engaged by a Microsoft IT Academy Program Member to teach an Authorized Training Session, and/or (ii) a MCT.
- o. "Trainer Content" means the trainer version of the Microsoft Instructor-Led Courseware and additional supplemental content designated solely for Trainers' use to teach a training session using the Microsoft Instructor-Led Courseware. Trainer Content may include Microsoft PowerPoint presentations, trainer preparation guide, train the trainer materials, Microsoft One Note packs, classroom setup guide and Pre-release course feedback form. To clarify, Trainer Content does not include any software, virtual hard disks or virtual machines.

2. USE RIGHTS. The Licensed Content is licensed not sold. The Licensed Content is licensed on a ***one copy per user basis***, such that you must acquire a license for each individual that accesses or uses the Licensed Content.

2.1 Below are five separate sets of use rights. Only one set of rights apply to you.

a. If you are a Microsoft IT Academy Program Member:

- i. Each license acquired on behalf of yourself may only be used to review one (1) copy of the Microsoft Instructor-Led Courseware in the form provided to you. If the Microsoft Instructor-Led Courseware is in digital format, you may install one (1) copy on up to three (3) Personal Devices. You may not install the Microsoft Instructor-Led Courseware on a device you do not own or control.
- ii. For each license you acquire on behalf of an End User or Trainer, you may either:
 - 1. distribute one (1) hard copy version of the Microsoft Instructor-Led Courseware to one (1) End User who is enrolled in the Authorized Training Session, and only immediately prior to the commencement of the Authorized Training Session that is the subject matter of the Microsoft Instructor-Led Courseware being provided, **or**
 - 2. provide one (1) End User with the unique redemption code and instructions on how they can access one (1) digital version of the Microsoft Instructor-Led Courseware, **or**
 - 3. provide one (1) Trainer with the unique redemption code and instructions on how they can access one (1) Trainer Content,**provided you comply with the following:**
- iii. you will only provide access to the Licensed Content to those individuals who have acquired a valid license to the Licensed Content,
- iv. you will ensure each End User attending an Authorized Training Session has their own valid licensed copy of the Microsoft Instructor-Led Courseware that is the subject of the Authorized Training Session,
- v. you will ensure that each End User provided with the hard-copy version of the Microsoft Instructor-Led Courseware will be presented with a copy of this agreement and each End User will agree that their use of the Microsoft Instructor-Led Courseware will be subject to the terms in this agreement prior to providing them with the Microsoft Instructor-Led Courseware. Each individual will be required to denote their acceptance of this agreement in a manner that is enforceable under local law prior to their accessing the Microsoft Instructor-Led Courseware,
- vi. you will ensure that each Trainer teaching an Authorized Training Session has their own valid licensed copy of the Trainer Content that is the subject of the Authorized Training Session,

- vii. you will only use qualified Trainers who have in-depth knowledge of and experience with the Microsoft technology that is the subject of the Microsoft Instructor-Led Courseware being taught for all your Authorized Training Sessions,
- viii. you will only deliver a maximum of 15 hours of training per week for each Authorized Training Session that uses a MOC title, and
- ix. you acknowledge that Trainers that are not MCTs will not have access to all of the trainer resources for the Microsoft Instructor-Led Courseware.

b. If you are a Microsoft Learning Competency Member:

- i. Each license acquired on behalf of yourself may only be used to review one (1) copy of the Microsoft Instructor-Led Courseware in the form provided to you. If the Microsoft Instructor-Led Courseware is in digital format, you may install one (1) copy on up to three (3) Personal Devices. You may not install the Microsoft Instructor-Led Courseware on a device you do not own or control.
- ii. For each license you acquire on behalf of an End User or Trainer, you may either:
 - 1. distribute one (1) hard copy version of the Microsoft Instructor-Led Courseware to one (1) End User attending the Authorized Training Session and only immediately prior to the commencement of the Authorized Training Session that is the subject matter of the Microsoft Instructor-Led Courseware provided, **or**
 - 2. provide one (1) End User attending the Authorized Training Session with the unique redemption code and instructions on how they can access one (1) digital version of the Microsoft Instructor-Led Courseware, **or**
 - 3. you will provide one (1) Trainer with the unique redemption code and instructions on how they can access one (1) Trainer Content,**provided you comply with the following:**
- iii. you will only provide access to the Licensed Content to those individuals who have acquired a valid license to the Licensed Content,
- iv. you will ensure that each End User attending an Authorized Training Session has their own valid licensed copy of the Microsoft Instructor-Led Courseware that is the subject of the Authorized Training Session,
- v. you will ensure that each End User provided with a hard-copy version of the Microsoft Instructor-Led Courseware will be presented with a copy of this agreement and each End User will agree that their use of the Microsoft Instructor-Led Courseware will be subject to the terms in this agreement prior to providing them with the Microsoft Instructor-Led Courseware. Each individual will be required to denote their acceptance of this agreement in a manner that is enforceable under local law prior to their accessing the Microsoft Instructor-Led Courseware,
- vi. you will ensure that each Trainer teaching an Authorized Training Session has their own valid licensed copy of the Trainer Content that is the subject of the Authorized Training Session,
- vii. you will only use qualified Trainers who hold the applicable Microsoft Certification credential that is the subject of the Microsoft Instructor-Led Courseware being taught for your Authorized Training Sessions,
- viii. you will only use qualified MCTs who also hold the applicable Microsoft Certification credential that is the subject of the MOC title being taught for all your Authorized Training Sessions using MOC,
- ix. you will only provide access to the Microsoft Instructor-Led Courseware to End Users, and
- x. you will only provide access to the Trainer Content to Trainers.

c. If you are a MPN Member:

- i. Each license acquired on behalf of yourself may only be used to review one (1) copy of the Microsoft Instructor-Led Courseware in the form provided to you. If the Microsoft Instructor-Led Courseware is in digital format, you may install one (1) copy on up to three (3) Personal Devices. You may not install the Microsoft Instructor-Led Courseware on a device you do not own or control.
- ii. For each license you acquire on behalf of an End User or Trainer, you may either:
 1. distribute one (1) hard copy version of the Microsoft Instructor-Led Courseware to one (1) End User attending the Private Training Session, and only immediately prior to the commencement of the Private Training Session that is the subject matter of the Microsoft Instructor-Led Courseware being provided, **or**
 2. provide one (1) End User who is attending the Private Training Session with the unique redemption code and instructions on how they can access one (1) digital version of the Microsoft Instructor-Led Courseware, **or**
 3. you will provide one (1) Trainer who is teaching the Private Training Session with the unique redemption code and instructions on how they can access one (1) Trainer Content,**provided you comply with the following:**
- iii. you will only provide access to the Licensed Content to those individuals who have acquired a valid license to the Licensed Content,
- iv. you will ensure that each End User attending an Private Training Session has their own valid licensed copy of the Microsoft Instructor-Led Courseware that is the subject of the Private Training Session,
- v. you will ensure that each End User provided with a hard copy version of the Microsoft Instructor-Led Courseware will be presented with a copy of this agreement and each End User will agree that their use of the Microsoft Instructor-Led Courseware will be subject to the terms in this agreement prior to providing them with the Microsoft Instructor-Led Courseware. Each individual will be required to denote their acceptance of this agreement in a manner that is enforceable under local law prior to their accessing the Microsoft Instructor-Led Courseware,
- vi. you will ensure that each Trainer teaching an Private Training Session has their own valid licensed copy of the Trainer Content that is the subject of the Private Training Session,
- vii. you will only use qualified Trainers who hold the applicable Microsoft Certification credential that is the subject of the Microsoft Instructor-Led Courseware being taught for all your Private Training Sessions,
- viii. you will only use qualified MCTs who hold the applicable Microsoft Certification credential that is the subject of the MOC title being taught for all your Private Training Sessions using MOC,
- ix. you will only provide access to the Microsoft Instructor-Led Courseware to End Users, and
- x. you will only provide access to the Trainer Content to Trainers.

d. If you are an End User:

For each license you acquire, you may use the Microsoft Instructor-Led Courseware solely for your personal training use. If the Microsoft Instructor-Led Courseware is in digital format, you may access the Microsoft Instructor-Led Courseware online using the unique redemption code provided to you by the training provider and install and use one (1) copy of the Microsoft Instructor-Led Courseware on up to three (3) Personal Devices. You may also print one (1) copy of the Microsoft Instructor-Led Courseware. You may not install the Microsoft Instructor-Led Courseware on a device you do not own or control.

e. If you are a Trainer.

- i. For each license you acquire, you may install and use one (1) copy of the Trainer Content in the form provided to you on one (1) Personal Device solely to prepare and deliver an Authorized Training Session or Private Training Session, and install one (1) additional copy on another Personal Device as a backup copy, which may be used only to reinstall the Trainer Content. You may not install or use a copy of the Trainer Content on a device you do not own or control. You may also print one (1) copy of the Trainer Content solely to prepare for and deliver an Authorized Training Session or Private Training Session.

- ii. You may customize the written portions of the Trainer Content that are logically associated with instruction of a training session in accordance with the most recent version of the MCT agreement. If you elect to exercise the foregoing rights, you agree to comply with the following: (i) customizations may only be used for teaching Authorized Training Sessions and Private Training Sessions, and (ii) all customizations will comply with this agreement. For clarity, any use of “customize” refers only to changing the order of slides and content, and/or not using all the slides or content, it does not mean changing or modifying any slide or content.

2.2 Separation of Components. The Licensed Content is licensed as a single unit and you may not separate their components and install them on different devices.

2.3 Redistribution of Licensed Content. Except as expressly provided in the use rights above, you may not distribute any Licensed Content or any portion thereof (including any permitted modifications) to any third parties without the express written permission of Microsoft.

2.4 Third Party Notices. The Licensed Content may include third party code that Microsoft, not the third party, licenses to you under this agreement. Notices, if any, for the third party code are included for your information only.

2.5 Additional Terms. Some Licensed Content may contain components with additional terms, conditions, and licenses regarding its use. Any non-conflicting terms in those conditions and licenses also apply to your use of that respective component and supplements the terms described in this agreement.

3. LICENSED CONTENT BASED ON PRE-RELEASE TECHNOLOGY. If the Licensed Content’s subject matter is based on a pre-release version of Microsoft technology (“**Pre-release**”), then in addition to the other provisions in this agreement, these terms also apply:

- a. **Pre-Release Licensed Content.** This Licensed Content subject matter is on the Pre-release version of the Microsoft technology. The technology may not work the way a final version of the technology will and we may change the technology for the final version. We also may not release a final version. Licensed Content based on the final version of the technology may not contain the same information as the Licensed Content based on the Pre-release version. Microsoft is under no obligation to provide you with any further content, including any Licensed Content based on the final version of the technology.
- b. **Feedback.** If you agree to give feedback about the Licensed Content to Microsoft, either directly or through its third party designee, you give to Microsoft without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft technology, Microsoft product, or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its technology, technologies, or products to third parties because we include your feedback in them. These rights survive this agreement.
- c. **Pre-release Term.** If you are an Microsoft IT Academy Program Member, Microsoft Learning Competency Member, MPN Member or Trainer, you will cease using all copies of the Licensed Content on the Pre-release technology upon (i) the date which Microsoft informs you is the end date for using the Licensed Content on the Pre-release technology, or (ii) sixty (60) days after the commercial release of the technology that is the subject of the Licensed Content, whichever is earliest (“**Pre-release term**”). Upon expiration or termination of the Pre-release term, you will irretrievably delete and destroy all copies of the Licensed Content in your possession or under your control.

- 4. SCOPE OF LICENSE.** The Licensed Content is licensed, not sold. This agreement only gives you some rights to use the Licensed Content. Microsoft reserves all other rights. Unless applicable law gives you more rights despite this limitation, you may use the Licensed Content only as expressly permitted in this agreement. In doing so, you must comply with any technical limitations in the Licensed Content that only allows you to use it in certain ways. Except as expressly permitted in this agreement, you may not:
- access or allow any individual to access the Licensed Content if they have not acquired a valid license for the Licensed Content,
 - alter, remove or obscure any copyright or other protective notices (including watermarks), branding or identifications contained in the Licensed Content,
 - modify or create a derivative work of any Licensed Content,
 - publicly display, or make the Licensed Content available for others to access or use,
 - copy, print, install, sell, publish, transmit, lend, adapt, reuse, link to or post, make available or distribute the Licensed Content to any third party,
 - work around any technical limitations in the Licensed Content, or
 - reverse engineer, decompile, remove or otherwise thwart any protections or disassemble the Licensed Content except and only to the extent that applicable law expressly permits, despite this limitation.
- 5. RESERVATION OF RIGHTS AND OWNERSHIP.** Microsoft reserves all rights not expressly granted to you in this agreement. The Licensed Content is protected by copyright and other intellectual property laws and treaties. Microsoft or its suppliers own the title, copyright, and other intellectual property rights in the Licensed Content.
- 6. EXPORT RESTRICTIONS.** The Licensed Content is subject to United States export laws and regulations. You must comply with all domestic and international export laws and regulations that apply to the Licensed Content. These laws include restrictions on destinations, end users and end use. For additional information, see www.microsoft.com/exporting.
- 7. SUPPORT SERVICES.** Because the Licensed Content is "as is", we may not provide support services for it.
- 8. TERMINATION.** Without prejudice to any other rights, Microsoft may terminate this agreement if you fail to comply with the terms and conditions of this agreement. Upon termination of this agreement for any reason, you will immediately stop all use of and delete and destroy all copies of the Licensed Content in your possession or under your control.
- 9. LINKS TO THIRD PARTY SITES.** You may link to third party sites through the use of the Licensed Content. The third party sites are not under the control of Microsoft, and Microsoft is not responsible for the contents of any third party sites, any links contained in third party sites, or any changes or updates to third party sites. Microsoft is not responsible for webcasting or any other form of transmission received from any third party sites. Microsoft is providing these links to third party sites to you only as a convenience, and the inclusion of any link does not imply an endorsement by Microsoft of the third party site.
- 10. ENTIRE AGREEMENT.** This agreement, and any additional terms for the Trainer Content, updates and supplements are the entire agreement for the Licensed Content, updates and supplements.
- 11. APPLICABLE LAW.**
- a. United States. If you acquired the Licensed Content in the United States, Washington state law governs the interpretation of this agreement and applies to claims for breach of it, regardless of conflict of laws principles. The laws of the state where you live govern all other claims, including claims under state consumer protection laws, unfair competition laws, and in tort.

- b. Outside the United States. If you acquired the Licensed Content in any other country, the laws of that country apply.

- 12. LEGAL EFFECT.** This agreement describes certain legal rights. You may have other rights under the laws of your country. You may also have rights with respect to the party from whom you acquired the Licensed Content. This agreement does not change your rights under the laws of your country if the laws of your country do not permit it to do so.
- 13. DISCLAIMER OF WARRANTY. THE LICENSED CONTENT IS LICENSED "AS-IS" AND "AS AVAILABLE." YOU BEAR THE RISK OF USING IT. MICROSOFT AND ITS RESPECTIVE AFFILIATES GIVES NO EXPRESS WARRANTIES, GUARANTEES, OR CONDITIONS. YOU MAY HAVE ADDITIONAL CONSUMER RIGHTS UNDER YOUR LOCAL LAWS WHICH THIS AGREEMENT CANNOT CHANGE. TO THE EXTENT PERMITTED UNDER YOUR LOCAL LAWS, MICROSOFT AND ITS RESPECTIVE AFFILIATES EXCLUDES ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.**
- 14. LIMITATION ON AND EXCLUSION OF REMEDIES AND DAMAGES. YOU CAN RECOVER FROM MICROSOFT, ITS RESPECTIVE AFFILIATES AND ITS SUPPLIERS ONLY DIRECT DAMAGES UP TO US\$5.00. YOU CANNOT RECOVER ANY OTHER DAMAGES, INCLUDING CONSEQUENTIAL, LOST PROFITS, SPECIAL, INDIRECT OR INCIDENTAL DAMAGES.**

This limitation applies to

- anything related to the Licensed Content, services, content (including code) on third party Internet sites or third-party programs; and
- claims for breach of contract, breach of warranty, guarantee or condition, strict liability, negligence, or other tort to the extent permitted by applicable law.

It also applies even if Microsoft knew or should have known about the possibility of the damages. The above limitation or exclusion may not apply to you because your country may not allow the exclusion or limitation of incidental, consequential or other damages.

Please note: As this Licensed Content is distributed in Quebec, Canada, some of the clauses in this agreement are provided below in French.

Remarque : Ce le contenu sous licence étant distribué au Québec, Canada, certaines des clauses dans ce contrat sont fournies ci-dessous en français.

EXONÉRATION DE GARANTIE. Le contenu sous licence visé par une licence est offert « tel quel ». Toute utilisation de ce contenu sous licence est à votre seule risque et péril. Microsoft n'accorde aucune autre garantie expresse. Vous pouvez bénéficier de droits additionnels en vertu du droit local sur la protection des consommateurs, que ce contrat ne peut modifier. La ou elles sont permises par le droit locale, les garanties implicites de qualité marchande, d'adéquation à un usage particulier et d'absence de contrefaçon sont exclues.

LIMITATION DES DOMMAGES-INTÉRÊTS ET EXCLUSION DE RESPONSABILITÉ POUR LES DOMMAGES. Vous pouvez obtenir de Microsoft et de ses fournisseurs une indemnisation en cas de dommages directs uniquement à hauteur de 5,00 \$ US. Vous ne pouvez prétendre à aucune indemnisation pour les autres dommages, y compris les dommages spéciaux, indirects ou accessoires et pertes de bénéfices.

Cette limitation concerne:

- tout ce qui est relié au le contenu sous licence, aux services ou au contenu (y compris le code) figurant sur des sites Internet tiers ou dans des programmes tiers; et
- les réclamations au titre de violation de contrat ou de garantie, ou au titre de responsabilité stricte, de négligence ou d'une autre faute dans la limite autorisée par la loi en vigueur.

Elle s'applique également, même si Microsoft connaissait ou devrait connaître l'éventualité d'un tel dommage. Si votre pays n'autorise pas l'exclusion ou la limitation de responsabilité pour les dommages indirects, accessoires ou de quelque nature que ce soit, il se peut que la limitation ou l'exclusion ci-dessus ne s'appliquera pas à votre égard.

EFFET JURIDIQUE. Le présent contrat décrit certains droits juridiques. Vous pourriez avoir d'autres droits prévus par les lois de votre pays. Le présent contrat ne modifie pas les droits que vous confèrent les lois de votre pays si celles-ci ne le permettent pas.

Revised July 2013

Welcome!

Thank you for taking our training! We've worked together with our Microsoft Certified Partners for Learning Solutions and our Microsoft IT Academies to bring you a world-class learning experience—whether you're a professional looking to advance your skills or a student preparing for a career in IT.

- **Microsoft Certified Trainers and Instructors**—Your instructor is a technical and instructional expert who meets ongoing certification requirements. And, if instructors are delivering training at one of our Certified Partners for Learning Solutions, they are also evaluated throughout the year by students and by Microsoft.
- **Certification Exam Benefits**—After training, consider taking a Microsoft Certification exam. Microsoft Certifications validate your skills on Microsoft technologies and can help differentiate you when finding a job or boosting your career. In fact, independent research by IDC concluded that 75% of managers believe certifications are important to team performance¹. Ask your instructor about Microsoft Certification exam promotions and discounts that may be available to you.
- **Customer Satisfaction Guarantee**—Our Certified Partners for Learning Solutions offer a satisfaction guarantee and we hold them accountable for it. At the end of class, please complete an evaluation of today's experience. We value your feedback!

We wish you a great learning experience and ongoing success in your career!

Sincerely,

Microsoft Learning
www.microsoft.com/learning

Microsoft | Learning

¹ IDC, Value of Certification: Team Certification and Organizational Performance, November 2006

Acknowledgements

Microsoft Learning would like to acknowledge and thank the following for their contribution towards developing this title. Their effort at various stages in the development has ensured that you have a good classroom experience.

Ed Harper – Content Developer

Ed Harper is a database developer specializing in Microsoft SQL Server. Ed has worked with SQL Server since 1999, and has developed and designed transaction-processing and reporting systems for cloud security, telecommunications, and financial services companies.

John Daisley – Content Developer

John Daisley is a mixed vendor Business Intelligence and Data Warehousing contractor with a wealth of data warehousing and Microsoft SQL Server database administration experience. Having worked in the Business Intelligence arena for over a decade, John has extensive experience of implementing business intelligence and database solutions using a wide range of technologies and across a wide range of industries including airlines, engineering, financial services, and manufacturing.

Nick Anderson – Content Developer

Nick Anderson MBCS MISTC has been a freelance Technical Writer since 1987 and Trainer since 1999. Nick has written internal and external facing content in many business and technical areas including development, infrastructure and finance projects involving SQL Server, Visual Studio and similar tools. Nick provides services for both new and existing document processes from knowledge capture to publishing.

Phil Stollery – Content Developer

Phil has been providing IT consultancy to South West England since graduating in Computer Science. He has worked with small and large organizations to improve their use of SQL Server, predominantly focusing on business information and surrounding technologies such as SharePoint. Most recently, Phil worked with the National Health Service in Gloucestershire on a custom intranet built on SharePoint. A trusted partner, he can communicate at all levels, from technical staff to senior management. Phil brings a wealth of experience that enhances any project.

Rachel Horder – Content Developer

Rachel Horder graduated with a degree in Journalism and began her career in London writing for The Times technology supplement. After discovering a love for programming, Rachel became a full-time developer, and now provides SQL Server consultancy services to businesses across a wide variety of industries. Rachel is MCSA certified, and continues to write technical articles and books, including *What's New in SQL Server 2012*. As an active member of the SQL Server community, Rachel organizes the Bristol SQL Server Club user group, runs the Bristol leg of SQL Relay, and is a volunteer at SQLBits.

Geoff Allix – Technical Reviewer

Geoff Allix is a Microsoft SQL Server subject matter expert and professional content developer at Content Master—a division of CM Group Ltd. As a Microsoft Certified Trainer, Geoff has delivered training courses on SQL Server since version 6.5. Geoff is a Microsoft Certified IT Professional for SQL Server and has extensive experience in designing and implementing database and BI solutions on SQL Server technologies, and has provided consultancy services to organizations seeking to implement and optimize database solutions.

Lin Joyner – Technical Reviewer

Lin is an experienced Microsoft SQL Server developer and administrator. She has worked with SQL Server since version 6.0 and previously as a Microsoft Certified Trainer, delivered training courses across the UK. Lin has a wide breadth of knowledge across SQL Server technologies, including BI and Reporting Services. Lin also designs and authors SQL Server and .NET development training materials. She has been writing instructional content for Microsoft for over 15 years.

Contents

Module 1: Introduction to Reporting Services

Module Overview	1-1
Lesson 1: Introduction to Reporting Services	1-2
Lesson 2: Reporting Services components	1-6
Lesson 3: Reporting Services tools	1-12
Lab: Exploring Reporting Services	1-19
Module Review and Takeaways	1-22

Module 2: Reporting Services Data Sources

Module Overview	2-1
Lesson 1: Data sources	2-2
Lesson 2: Connection strings	2-8
Lesson 3: Datasets	2-17
Lab A: Configuring data access with Report Builder	2-21
Lab B: Configuring data access with Report Designer	2-23
Module Review and Takeaways	2-25

Module 3: Creating Paginated Reports

Module Overview	3-1
Lesson 1: Creating a report with the Report Wizard	3-2
Lesson 2: Creating a report	3-7
Lesson 3: Publishing a report	3-15
Lab: Creating reports	3-20
Module Review and Takeaways	3-25

Module 4: Working with Reporting Services Data

Module Overview	4-1
Lesson 1: Data filters	4-2
Lesson 2: Report parameters	4-9
Lesson 3: Implementing report filters and parameters	4-14
Lab: Create a parameterized report	4-23
Module Review and Takeaways	4-29

Module 5: Visualizing Data with Reporting Services

Module Overview	5-1
Lesson 1: Formatting data	5-2
Lesson 2: Images and charts	5-9
Lesson 3: Databars, sparklines, indicators, gauges, and maps	5-15
Lesson 4: KPIs	5-24
Lab: Manage formatting	5-26
Module Review and Takeaways	5-32

Module 6: Summarizing Report Data

Module Overview	6-1
Lesson 1: Sorting and grouping	6-2
Lesson 2: Report subreports	6-13
Lesson 3: Drilldown and drillthrough	6-20
Lab: Summarizing report data	6-27
Module Review and Takeaways	6-32

Module 7: Sharing Reporting Services Reports

Module Overview	7-1
Lesson 1: Schedules	7-2
Lesson 2: Report caching, snapshots, and comments	7-7
Lesson 3: Report subscription and delivery	7-13
Lab: Sharing Reporting Services reports	7-20
Module Review and Takeaways	7-23

Module 8: Administering Reporting Services

Module Overview	8-1
Lesson 1: Administering Reporting Services	8-2
Lesson 2: Reporting Services configuration	8-6
Lesson 3: Reporting Services performance	8-12
Lab: Administering Reporting Services	8-18
Module Review and Takeaways	8-22

Module 9: Extending and Integrating Reporting Services

Module Overview	9-1
Lesson 1: Expressions and embedded code	9-2
Lesson 2: Extending Reporting Services	9-10
Lesson 3: Integrating Reporting Services	9-18
Lab: Extending and integrating Reporting Services	9-26
Module Review and Takeaways	9-31

Module 10: Introduction to Mobile Reports

Module Overview	10-1
Lesson 1: Overview of mobile reports	10-2
Lesson 2: Preparing data for mobile reports	10-7
Lesson 3: Mobile Report Publisher	10-14
Lab: Introduction to mobile reports	10-18
Module Review and Takeaways	10-21

Module 11: Developing Mobile Reports

Module Overview	11-1
Lesson 1: Designing and publishing mobile reports	11-2
Lesson 2: Drillthrough in mobile reports	11-12
Lab: Developing mobile reports	11-18
Module Review and Takeaways	11-23

Lab Answer Keys

Module 1 Lab: Exploring Reporting Services	L01-1
Module 2 Lab A: Configuring data access with Report Builder	L02-1
Module 2 Lab B: Configuring data access with Report Designer	L02-3
Module 3 Lab: Creating reports	L03-1
Module 4 Lab: Create a parameterized report	L04-1
Module 5 Lab: Manage formatting	L05-1
Module 6 Lab: Summarizing report data	L06-1
Module 7 Lab: Sharing Reporting Services reports	L07-1
Module 8 Lab: Administering Reporting Services	L08-1
Module 9 Lab: Extending and integrating Reporting Services	L09-1
Module 10 Lab: Introduction to mobile reports	L10-1
Module 11 Lab: Developing mobile reports	L11-1

About This Course

This section provides a brief description of the course, audience, suggested prerequisites, and course objectives.

Course Description



Note: This C version of this course has been completely re-written and extended to 5 days to better cover the functionality of SQL Server Reporting Services. Labs have been designed for report builder and report designer, whichever the student will be using.

This five-day instructor-led course teaches students how to implement a SQL Server Reporting Services solution for data analysis in an organization. The course discusses how to use the Reporting Services development tools to create and manage reports and implement self-service BI solutions.

Audience

The primary audience for this course is database professionals who need to fulfil a BI developer role to create reports. Primary responsibilities will include implementing reports and mobile reports.

The secondary audiences for this course are power information workers.

Student Prerequisites

In addition to their professional experience, students who attend this training should have technical knowledge equivalent to the following course:

- 20761C: Querying Data with Transact-SQL

Course Objectives

After completing this course, students will be able to:

- Describe reporting services and its components
- Describe reporting services data sources
- Implement paginated reports
- Work with reporting services data
- Visualize data with reporting services
- Aggregate report data
- Share reporting services reports
- Administer reporting services
- Expand and integrate reporting services
- Describe mobile reports
- Develop mobile reports

Course Outline

The course outline is as follows:

- Module 1: 'Introduction to reporting services' introduces Microsoft® SQL Server® Reporting Services, its components, and the tools used to work with it.
- Module 2: 'Reporting services data sources' describes various Report Services data sources and how these are configured.
- Module 3: 'Creating paginated reports' describes how to create reports with report designer or report builder.
- Module 4: 'Working with reporting services data' describes how to configure and re-configure reports with report builder or report designer, as business needs dictate.
- Module 5: 'Visualizing data with report services' describes how to use data visualization to make your data easier to understand.
- Module 6: 'Summarizing report data' .This module describes how to create group structures, summarize data, and provide interactivity in your reports, so that users see the level of detail or summary that they need.
- Module 7: 'Sharing reporting services reports' covers report scheduling, caching and the report lifecycle, automatic subscription and delivery of reports.
- Module 8: 'Administering reporting services' covers administrative tasks such as configuration of the web portal and web service, branding the web portal, and ensuring access to sensitive reports is carefully controlled.
- Module 9: 'Extending and integrating reporting services' covers methods for extending the functionality of reporting services with expressions and custom code. It also covers methods for working with reporting services programmatically, and integrating reporting services reports into other applications.
- Module 10: 'Introduction to mobile reports' introduces the design and publication of reports that are intended for consumption on mobile devices such as smartphones and tablets.
- Module 11: 'Developing mobile reports' covers the different element types you can add to your reporting services mobile reports.

Course Materials

The following materials are included with your kit:

- **Course Handbook:** a succinct classroom learning guide that provides the critical technical information in a crisp, tightly-focused format, which is essential for an effective in-class learning experience.
 - **Lessons:** guide you through the learning objectives and provide the key points that are critical to the success of the in-class learning experience.
 - **Labs:** provide a real-world, hands-on platform for you to apply the knowledge and skills learned in the module.
 - **Module Reviews and Takeaways:** provide on-the-job reference material to boost knowledge and skills retention.
 - **Lab Answer Keys:** provide step-by-step lab solution guidance.



Additional Reading: Course Companion Content on the <https://www.microsoft.com/en-us/learning/companion-moc.aspx> Site: searchable, easy-to-browse digital content with integrated premium online resources that supplement the Course Handbook.

- **Modules:** include companion content, such as questions and answers, detailed demo steps and additional reading links, for each lesson. Additionally, they include Lab Review questions and answers and Module Reviews and Takeaways sections, which contain the review questions and answers, best practices, common issues and troubleshooting tips with answers, and real-world issues and scenarios with answers.
- **Resources:** include well-categorized additional resources that give you immediate access to the most current premium content on TechNet, MSDN®, or Microsoft® Press®.



Additional Reading: Student Course files on the <https://www.microsoft.com/en-us/learning/companion-moc.aspx> Site: includes the Allfiles.exe, a self-extracting executable file that contains all required files for the labs and demonstrations.

- **Course evaluation:** at the end of the course, you will have the opportunity to complete an online evaluation to provide feedback on the course, training facility, and instructor.
 - To provide additional comments or feedback on the course, send an email to mcspprt@microsoft.com. To inquire about the Microsoft Certification Program, send an email to mcphep@microsoft.com.

Virtual Machine Environment

This section provides the information for setting up the classroom environment to support the business scenario of the course.

Virtual Machine Configuration

In this course, you will use Microsoft® Hyper-V™ to perform the labs.



Note: At the end of each lab, you must revert the virtual machines to a snapshot. You can find the instructions for this procedure at the end of each lab

The following table shows the role of each virtual machine that is used in this course:

Virtual machine	Role
10990C-MIA-DC	MIA-DC1 is a domain controller.
10990C-MIA-SQL	MIA-SQL has SQL Server 2017 installed
MSL-TMG1	TMG1 is used to access the internet

Software Configuration

The following software is installed on the virtual machines:

- Windows Server 2016
- SQL2017
- Microsoft Office 2016
- SharePoint 2016

Course Files

The files associated with the labs in this course are located in the D:\Labfiles folder on the 10990C-MIA-SQL virtual machine.

Classroom Setup

Each classroom computer will have the same virtual machine configured in the same way.

Course Hardware Level

To ensure a satisfactory student experience, Microsoft Learning requires a minimum equipment configuration for trainer and student computers in all Microsoft Certified Partner for Learning Solutions (CPLS) classrooms in which Official Microsoft Learning Product courseware is taught.

- Processor*: 2.8 GHz 64-bit processor (multi-core) or better
 - **AMD:
 - AMD Virtualization (AMD-V)
 - Second Level Address Translation (SLAT) - nested page tables (NPT)
 - Hardware-enforced Data Execution Prevention (DEP) must be available and enabled (NX Bit)
 - Supports TPM 2.0 or greater

- ****Intel:**
 - Intel Virtualization Technology (Intel VT)
 - Supports Second Level Address Translation (SLAT) – Extended Page Table (EPT)
 - Hardware-enforced Data Execution Prevention (DEP) must be available and enabled (XD bit)
 - Supports TPM 2.0 or greater
- Hard Disk: 500GB SSD System Drive
- RAM: 32 GB minimum
- Network adapter
- Monitor: Dual monitors supporting 1440 X 900 minimum resolution
- Mouse or compatible pointing device
- Sound card with headsets

In addition, the instructor computer must:

- Be connected to a projection display device that supports SVGA 1024 x 768 pixels, 16 bit colors.
- Have a sound card with amplified speakers

MCT USE ONLY. STUDENT USE PROHIBITED

Module 1

Introduction to Reporting Services

Contents:

Module Overview	1-1
Lesson 1: Introduction to Reporting Services	1-2
Lesson 2: Reporting Services components	1-6
Lesson 3: Reporting Services tools	1-12
Lab: Exploring Reporting Services	1-19
Module Review and Takeaways	1-22

Module Overview

Business intelligence (BI) is a term that has become increasingly common in recent years. Along with big data, data mining, predictive analytics, data science, and data stewards, BI is now very much part of the business vocabulary. Much of the impetus behind this is the need for organizations to cope with ever increasing datasets. It's now normal to have databases that contain millions of rows, requiring gigabytes, terabytes, or even petabytes, of storage space. Data is no longer confined to an on-premises server room—it's hosted in the cloud, feeds are taken from third-party providers, public datasets are freely available, and social media interactions generate ever-expanding datasets.

Reporting and analysis is certainly not a new concept to business, but the difference between how data analysis is done today, compared with five or 10 years ago, is immense. Today, organizations need BI to see not only what has been done in the past, but also more of what's to come in the future. There is now an overwhelming amount of data to gather and compose into reports. Add to this an increasing need for data to offer up-to-date numbers, so business reacts faster to changing trends in markets and industries.

Businesses that react fast and predict near-term trends to provide products and services where there is consumer demand, have the best chance of survival in a modern and highly competitive world. With the rise of big data, there's an increasing need for data analysts who take this data, and find the critical points within a plethora of information.

This module introduces Microsoft® SQL Server® Reporting Services, its components, and the tools used to work with it.

Objectives

At the end of this module, you should be able to:

- Give a context for Reporting Services.
- Describe the components of Reporting Services.
- Explain Reporting Services tools.

Lesson 1

Introduction to Reporting Services

This lesson introduces Reporting Services by establishing a broader context for BI tools in general, then highlighting the role that Reporting Services is designed to fulfil.

Lesson Objectives

At the end of this lesson, you should be able to:

- Describe some common scenarios for BI.
- Describe trends in BI collaboration.
- Describe Reporting Services, and its place in the Microsoft BI stack.
- Describe scenarios where Reporting Services might be used.

Business intelligence scenarios

Since the rise of the internet, social media, and the rapid growth of e-commerce, more and more data is being generated, gathered, and analyzed. Supermarkets and retail outlets offer store cards, loyalty cards, or reward cards—depending on which label they use—because they want to track spending habits and use this data to sell you more. They gather data, analyze what you like to buy, and then offer incentives to entice you to buy more of the same, or similar. Meanwhile, your online habits are monitored by cookies, and advertisements show up on sites, tempting you to buy something you might have searched for earlier.

- Big data is the result of data generated by the internet, social media, and e-commerce:
 - Data is constantly being gathered for commercial use
 - Data is constantly growing in size
- Reporting:
 - Extracting data and presenting it to enable decision-making
 - Show metrics for organizational performance
- Analysis:
 - Evaluating data to discover insights
 - Data should answer questions, but it quickly becomes outdated
- BI is widely adopted, even by small organizations

Reporting

Extracting data from your company's database and presenting it in reports is certainly not a new phenomenon. Most organizations, whatever their size, use some form of reporting, as a reflection of performance within their sector. Until recently, most organizations were happy with end-of-month and annual reports, as a backward reflection of their performance. Modern reporting still needs this, but it should also look to the future and predict where and how to sell more, thereby increasing turnover and reducing the bottom line.

Traditionally, reports have been compiled by department heads, and then given to directors to guide their decision-making. Organizational data, or BI, was the privilege of a few. For example, reports show metrics—how much did we sell last month? How many new customers have we acquired this year? How many mentions did our latest promotion receive on social media? A report can provide the answers to questions that the organization needs to make decisions. Reports might be contained in spreadsheets, or created using a visual tool, and distributed on a daily, weekly, or other regular schedule. Reflecting on past performance is a worthwhile task, but modern reports must also predict the future.

Analysis

Analysis is the process of evaluating data to find insights. Data analysis should answer questions, and offer guidance in decision-making. Data is extracted from source, and then cleaned, modeled, and transformed until it can be appropriately presented in a report. The report might be a simple table in a spreadsheet, or a visual and dynamic, colorful solution—how the data is presented affects the analysis and the conclusions drawn. For example, you might present data in a column chart, but not notice patterns in that data until you use a different type of chart—such as a map or scatter chart—and discover clusters of behavior because of geographic location, or outliers that are skewing results.

With so much data to analyze, and constant changes in consumer and market trends, modern data has a limited lifespan. Data quickly becomes outdated, so the process of analysis is ongoing. However, with bigger data to analyze, more questions are asked of it. With an increase in publicly available datasets, including population changes, socioeconomic data, weather patterns, and climate change, corporate data can be analyzed against a backdrop of relevant statistics.

Increasing adoption of BI by a wider range of organizations

BI is no longer the reserve of big organizations that have large budgets to throw at data warehousing projects. Any business that operates on the web gathers information about their customers' spending habits, the products they viewed, and their buying decisions. It now seems that our online presence, enhanced through the proliferation of mobile devices, is continuously monitored, with all our moves and preferences stored for analysis. To be more efficient, and therefore more competitive, organizations of all sizes must gather data to some extent. However, gathering this data is of no use unless it's converted to actionable information. Along with increasing volumes of data, the availability of cheaper, easier to use solutions has helped drive the market, meaning organizations that have even the smallest of budgets can devote some level of resource to BI.

Collaboration trends in business intelligence

Data is ubiquitously generated and consumed. Data is no longer retained and controlled by a handful of decision makers in an organization; instead, it's used at all levels, meaning that colleagues can react to data, and change the course of their work if necessary. Information is critical to companies of all sizes and across industries, with information workers needing to collaborate, and share data and results. Microsoft Excel® has long been the dominant tool of the business user—spreadsheets are created, shared, published, altered, emailed, printed, saved, and distributed without version control, or adherence to security policies. As spreadsheets are shared and changed, and shared again, analysts work from different datasets, see different results, and reach different conclusions. To collaborate and work cohesively, analysts must be able to synchronize their teamwork.

- Increasing data volumes
- Real-time or near real-time reporting
- Self-service reporting

Self-service reporting and analysis

It could be argued that self-service BI has been around since spreadsheets first entered the software market, enabling users to crunch numbers at their desks. The almost universal adoption of Excel has enabled this trend to continue. Following the integration of the four power tools—PowerPivot, Power Query, Power View, and Power Map—Excel users acquire data from a myriad of sources, and then model, transform, and present that data in sophisticated visualizations. The attraction of Excel and its power tools

is the independence it offers to business users. If users access the data they need, they can immediately begin shaping and formatting that data, and designing reports to their own specification.

Using a more sophisticated reporting solution generally requires a dedicated report developer, a lengthy process to submit a feature requirement to IT, and a wait for the report to be developed and published—only to find it doesn't deliver the correct data. And so begins another lengthy process of submitting a change request, and waiting for the report developer to make the changes. Giving users access to the data means they see what's available for analysis, and lets them decide what's useful. The delay in waiting for a report not only frustrates users and holds back their work, but also delays decision-making and the ability for organizations to react to changing circumstances.

Reporting Services and the Microsoft BI stack

Reporting Services is a platform that you use to create, manage, share, and distribute traditional paginated reports, reports designed for consumption on mobile devices, and Power BI™ Desktop reports—either on-premises or in the cloud.

Reporting Services is part of the Microsoft BI stack. Data is central to BI and SQL Server provides a data platform with the performance, reliability, and security required for a mission critical system. Azure SQL Database and Azure SQL Data Warehouse extend this capability to the cloud and add secure global access, and scalability on the fly, in addition to built-in data protection and high availability.

Microsoft SQL Server Analysis Services (SSAS) provides multidimensional and tabular modeling, and data mining capabilities to gain insights into your data.

Microsoft SQL Server Integration Services (SSIS) provides scalable, enterprise-grade tools for extracting, transforming and loading data between source and destination systems.

Power BI includes both Power BI on the internet and Power BI Desktop for Windows®. You use these systems to develop rich visualizations and dashboards for consumption through web browsers or mobile apps.

- Data platform
 - SQL Server Database Engine
 - Azure SQL Database
 - Azure SQL Data Warehouse
- Data modeling and mining
 - SQL Server Analysis Services
- Data transformation and load
 - SQL Server Integration Services
- Rich reporting
 - SQL Server Reporting Services
 - SQL Azure Reporting Services
 - Power BI and Power BI Desktop

Reporting Services business scenarios

You use Reporting Services to:

- Publish reports for consumption through a web browser, on a mobile device, or by email message.
- Share Excel workbooks.
- Organize reports in a web portal.
- Control access to reports through granular security.
- Publish shared data sources for self-serve reporting.
- Publish dashboards and performance indicators.
- Generate interactive reports that have rich visual representations of data.

Question: How does your organization approach BI? Is this a major part of the corporate strategy? What BI solutions does your organization use? What do you think are the major issues with your organization's approach to BI?

You use Reporting Services to:

- Publish reports for consumption through a web browser, on a mobile device, or by email message
- Organize reports in a web portal
- Control access to reports through granular security
- Publish shared data sources for self-serve reporting
- Publish dashboards and performance indicators
- Generate interactive reports that have rich visual representations of data

Lesson 2

Reporting Services components

This lesson introduces the major components and features of Reporting Services, and discusses different ways that you choose to deploy Reporting Services, both on-premises and in the cloud.

Lesson Objectives

At the end of this lesson, you should be able to:

- Describe the Reporting Services web portal.
- Describe the Report Server.
- Describe the ReportViewer control.
- Explain the features of different editions of Reporting Services.
- Explain the Reporting Services on-premises and cloud deployment options.
- Describe the different deployment models when you install Reporting Services on-premises.

Web portal

You use the Reporting Services web portal to organize and view reports, and administer your Reporting Services instance, through a web browser. You access the web portal by going to the relevant URL—by default, this is **http://<server_name>/reports**. Depending on how the portal is configured, you might need to use an encrypted connection with **https://**, or reference a specific TCP port number—**<server name>:<port>**.

You use the web portal to perform many tasks, including:

- Search for, view, print, and subscribe to reports.
- Upload reports, data sources, and datasets.
- Configure report properties.
- Organize reports—and other items on the server—in a folder hierarchy.
- Configure role-based security to control access to reports and folders.
- Create KPIs.
- Create data-driven subscriptions to deliver reports to a recipient list.
- Download Reporting Services tools.

- View, organize, and administer reports
- Default URL `http://<server_name>/reports`
- Perform many tasks, including:
 - Search for, view, print, and subscribe to reports
 - Upload reports, data sources, and datasets
 - Configure report properties
 - Organize server items in a folder hierarchy
 - Configure role-based security
 - Create KPIs
 - Create data-driven subscriptions to deliver reports to a recipient list
 - Download Reporting Services tools



Note: The web portal is only available when Reporting Services is configured in native mode. You will learn more about this later in the lesson.

In SQL Server 2008 and later, the web portal does not require Internet Information Services (IIS) to operate.

For more information about the Reporting Services web portal, see the topic *Web portal (SSRS Native Mode)* in Microsoft Docs:

Web portal (SSRS Native Mode)

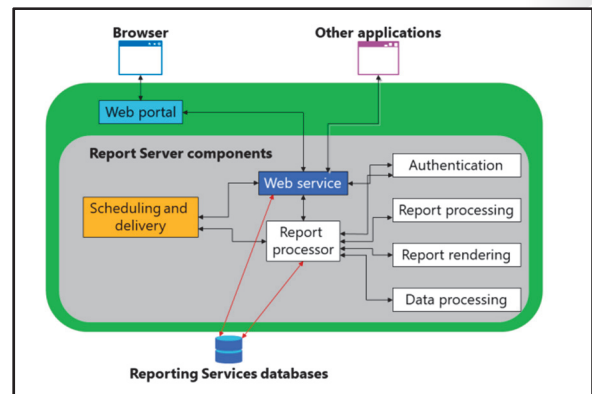
<https://aka.ms/Cgocyk>


Report Server

Report Server is the backend application server for Reporting Services. Report Server is responsible for storing and rendering reports, in addition to running scheduled reports, and managing report security.

Report Server is made up of two main elements:

- Report Server service:** A Windows service that powers the web portal, and the Report Server web service (an XML web service with a SOAP API that you use to interact with Reporting Services from external applications). The Report Server also runs background tasks that are responsible for running and delivering scheduled reports and report subscriptions.
- Report Server databases:** Report Server uses a database to store reports and report metadata, and another database to store temporary data that's used for report processing. The databases are held in an instance of the SQL Server database engine. By default, the databases are named **reportserver** and **reportservertempdb**. The Report Server databases do not need to be hosted on the same computer as the Report Server instance that they relate to; you might achieve better performance by hosting Reporting Services and SQL Server on different computers.



 **Note:** In a Reporting Services instance that runs in SharePoint mode, the Report Server is not responsible for organizing or securing reports; these are the responsibility of the SharePoint cluster.

For more information about the Report Server, see the topic *Reporting Services Report Server (Native Mode)* in Microsoft Docs:

Reporting Services Report Server (Native Mode)

<https://aka.ms/Fjgain>

ReportViewer control

You use the ReportViewer control to embed Reporting Services reports in web pages and applications. It comes in two versions:

- **ReportViewer web control:** suitable for embedding in web pages. The ReportViewer web control is used by the Reporting Services web portal to render reports when you view them from a web browser.
- **ReportViewer window control:** suitable for embedding in desktop applications.

- Embed Reporting Services reports in web pages and applications:
- ReportViewer web control for web pages
- ReportViewer window control for desktop applications

Typically, instances of the ReportViewer control connect to a Reporting Services instance to display reports; the Report Server generates and renders the report, returning the result to the application. You also use ReportViewer controls to generate reports locally, without a connection to a Report Server—this mode has reduced functionality.

To learn more about ReportViewer controls, see the topic *Integrating Reporting Services Using ReportViewer Controls* in Microsoft Docs:



Integrating Reporting Services Using ReportViewer Controls

<https://aka.ms/Va6aq7>

Reporting Services features by edition

Reporting Services is available in the following editions of SQL Server:

- Express with Advanced Services edition
- Web edition
- Standard edition
- Enterprise edition
- Developer edition

- Reporting Services available with:
 - Express with Advanced Services edition
 - Web edition
 - Standard edition
 - Enterprise edition
 - Developer edition
- Not all features are available to all editions

Not all the features of Reporting Services are available to every edition of SQL Server. Enterprise edition (and Developer edition—that includes all the features of Enterprise edition but licensed for nonproduction use) includes all the features of Reporting Services.

Some features that are not available to all editions of Reporting Services include:

- SQL Server data source: Web edition and Express edition are restricted to querying SQL Server Web edition and SQL Server Express edition respectively as a data source.
- Custom security roles are not available to Web edition and Express edition.
- Mobile report and Power BI support are available only in Enterprise edition.

For more information about the features available to each edition of Reporting Services, see the topic *Reporting Services Features Supported by the Editions of SQL Server 2016* in Microsoft Docs:

 **Reporting Services Features Supported by the Editions of SQL Server 2016**

<https://aka.ms/H00ovk>

Cloud and on-premises deployments

You choose to deploy Reporting Services either on-premises, or on a virtual machine in a cloud service provider such as Microsoft Azure. There are several factors to consider when you select how to deploy Reporting Services:

Low initial capital outlay

By using a cloud-based solution, there is no capital outlay for hardware, and most software costs become part of a monthly subscription model.

Scalable

Cloud computing systems are massively scalable.

Not only is it straightforward to scale up the virtual machines and storage to cope with additional demand, but it's also easy to scale down the system to save money when demand is lower.

Global access to systems

To access a cloud-based system, you just need an internet connection. Developers of your systems and consumers of your data can, therefore, be located almost anywhere in the world.

High availability

Cloud-based systems run on dedicated server farms that have built-in redundancy. This enables extremely high availability when compared with traditional on-premises solutions, and avoids the complication of managing complex clustered solutions.

Disaster recovery

Azure storage supports geo-replication to multiple data centers around the world; data is stored three times redundantly in each data center. This would not stop incorrect data in your system; in fact, you would now have many copies of the incorrect data—but there's also the ability to manage backups in Azure.

Security

By storing data in the cloud, there is no risk that lost hardware will make sensitive data available. Data is securely stored and is accessed by mobile devices that are more portable, more cost effective, and by those devices that can be remotely wiped if lost.

Of course, cloud computing is not the best solution in every scenario. You might find it beneficial to use existing hardware and software with no additional cost to your organization; you might need to store data on-premises locally to meet data governance requirements; you might be using cloud storage for backups to provide the benefits of cloud-based disaster recovery for your on-premises solution. The key thing is that every new solution is evaluated, based on that specific solution.

Benefits of cloud computing include:

- Low initial capital outlay
- Scalable
- Global access to systems
- High availability
- Disaster recovery
- Security



Note: The Power BI Report Server service offered for on-premises and in Azure can run both Power BI and Reporting Services reports, and offers another method for delivering reports.

For more information about Power BI Report Server, see the topic *On-premises reporting with Power BI Report Server* on the Microsoft Power BI site:



On-premises reporting with Power BI Report Server

<https://aka.ms/Dfz5l1>

On-premises deployment models

In versions before SQL Server 2017, you choose to install Reporting Services in one of two modes:

- Native mode
- SharePoint integrated mode



Note: In SQL Server 2017, only native mode is available. SharePoint integrated mode is no longer supported.

- Native mode
 - All features managed by Reporting Services
- SharePoint integrated mode
 - Security and report organization managed by SharePoint
 - Web portal not available
 - Not supported by SQL Server 2017

In SharePoint integrated mode, Reporting Services is responsible only for rendering reports. Other functions, such as security and report organization, are carried out by SharePoint. You do not use the Reporting Services web portal in SharePoint integrated mode. Some features introduced in SQL Server 2016, such as mobile reports and KPIs, are not available in SharePoint integrated mode for SQL Server 2016.

In native mode, all functions are performed by Reporting Services.

If your organization uses Reporting Services for SQL Server 2016 or earlier, there's more information about the differences between native mode and SharePoint integrated mode in the topic *Reporting Services report server* on Microsoft Docs:



Reporting Services report server

<https://aka.ms/Rwve6u>

Demonstration: Running a report with the web portal

In this demonstration, you will see how to:

- Run a query with Transact-SQL.
- View the query data in a Reporting Services report.

Demonstration Steps

View data in a Transact-SQL query

1. On the taskbar, click **Microsoft SQL Server Management Studio 17**.
2. In the **Connect to Server** dialog box, in the **Server name** box, type **MIA-SQL**, in the **Authentication** list, click **Windows Authentication**, and then click **Connect**.
3. On the **File** menu, point to **Open**, and then click **File**.
4. In the **Open File** dialog box, go to **D:\Demofiles\Mod01**, click **Sales_Order_Detail_Query.sql**, and then click **Open**.
5. Review the query, and click **Execute**.

View data in a Reporting Services report

1. On the taskbar, click **Internet Explorer**.
2. In the address bar, type **mia-sql/Reports_SQL2**, and then press Enter.
3. Click the **AdventureWorks Sample Reports** folder, and then click **Sales_Order_Detail**.
4. In the **First Order ID** box, type **57033**.
5. In the **Last Order ID** box, type **57033**, and then click **View Report**.
6. Compare the output from the query and the report. When you have finished, close SQL Server Management Studio without saving any changes.

Check Your Knowledge

Question	
If you are installing SQL Server 2017 Reporting Services on an on-premises server, what modes can you choose to run Reporting Services in?	
Select the correct answer.	
<input type="checkbox"/>	SharePoint integrated mode only
<input type="checkbox"/>	Native mode only
<input type="checkbox"/>	SharePoint integrated mode or native mode
<input type="checkbox"/>	None of the above

Lesson 3

Reporting Services tools

This lesson introduces tools for designing Reporting Services reports, including Report Builder, SQL Server Data Tools Report Designer, and Mobile Report Publisher. You will also learn about tools for administering and configuring Reporting Services in native mode.

Lesson Objectives

At the end of this lesson, you should be able to:

- Describe the tools you use to create Reporting Services reports.
- Explain which tools might be used for various business roles.
- Describe Reporting Services administration tools.

Report creation tools

Microsoft provides three tools that you use to create Reporting Services reports:

- Report Builder
- Report Designer
- SQL Server Mobile Report Publisher

- Report Builder
 - Standalone Windows application for paginated reports
- Report Designer
 - Part of the SSDT add-in for Visual Studio for paginated reports
- SQL Server Mobile Report Publisher
 - Standalone Windows application for mobile reports

Report Builder

Report Builder is a standalone Windows desktop application that you use to design, edit, and run Reporting Services paginated reports. You use Report Builder to open and edit reports from local report definition files, or directly from Reporting Services. You download Report Builder from the Reporting Services web portal, or from Microsoft.

Report Designer

Report Designer is part of the SQL Server Data Tools (SSDT) add-in for Visual Studio® on Windows. You use Report Designer to create and edit Reporting Services paginated reports, shared datasets, and shared data sources, in addition to organizing your reports into projects and solutions. You deploy individual items, projects, or solutions to a target Reporting Services instance. Report Designer also includes an IntelliSense® editor for custom expressions written in Visual Basic®.

SSDT is compatible with all editions of Visual Studio, including the community edition. You can download SSDT from Microsoft.

SQL Server Mobile Report Publisher

SQL Server Mobile Report Publisher is a standalone Windows desktop application that you use to design, edit, and run Reporting Services mobile reports. You use Mobile Report Publisher to open and edit reports from local report definition files, or directly from Reporting Services. You download Mobile Report Publisher from the Reporting Services web portal, or from Microsoft.

For more information about report creation tools, see the *Tools for Report Authoring* section of the *Reporting Services Tools* topic in Microsoft Docs:

 **Reporting Services Tools—Tools for Report Authoring**

<https://aka.ms/lqwdr>

Roles that create reports—and the tools they use

Report authors come from many different job roles. The goal for a successful BI project is to enable a business to make quicker and better decisions. Therefore, reports will be created, managed and improved by many different roles within an organization. Some common scenarios are explored in further detail here, considering an individual's job role and how this might affect their choice of report authoring tool.

End user <ul style="list-style-type: none"> • Business expert • Information first approach • Likely to use Excel • Will consume reports used by other roles 	BI developer <ul style="list-style-type: none"> • Database background • Data first approach • Likely to use Report Designer • Will also use Mobile Report Publisher if mobile reports are required
Data stewards <ul style="list-style-type: none"> • Data expert • Quality first approach • Likely to use Excel and, in time, Report Builder 	Business analyst <ul style="list-style-type: none"> • Business background • Design first approach • Likely to use Report Builder and Mobile Report Publisher

End user

When the data becomes easily accessible to end users they will likely need to make amendments, improvements, and additions to reports. They might also realize that there are further reports that require development. The end users will probably create and edit their reports in Excel. These reports might include pivot tables that use data that's stored in preexisting analysis server-based data cubes, made available by the BI project.

Data steward

Data stewards are business area data experts. Within their role, data stewards are tasked with ensuring data quality and will, therefore, have specific reporting requirements to support that role. Data stewards will be the reporting experts in their respective business areas, and will support end users in locating data, defining its meaning, and shaping it into useful information. Data stewards will be very familiar with Excel. Over time, if they are producing more complex reports that need to be shared within the organization, they might begin using Report Builder.

Business analyst

Traditionally, business analysts work closely with end users and stakeholders to define any new reports. Business analysts are interested in the output of a report—they are likely to work back from the output that they require, and then find the data that will produce this output. Typically, they will be very familiar with using Excel to create reports, and should find it relatively straightforward to make the transition to Report Builder and Mobile Report Publisher.

BI developer

Traditionally, report developers have started as database professionals, and so have created Reporting Services reports with the SQL Server tools. They are likely to have a deep understanding of the data and create reports in a data forward way—they will create the most suitable reports based on the data. A BI developer is a professional who should be familiar with SSDT and is therefore more productive when using Report Designer.

Generalizations might be made about which tools different job roles would favor. It's possible that a business analyst would start to use Report Designer if they needed to add Visual Basic functionality.

Equally, a database professional might start to use Report Builder for its intuitive nature, or Mobile Report Publisher because of the increasing demands for report consumption on mobile devices.

Reporting Services administration tools

There are several tools that you use to manage different aspects of the configuration of a Reporting Services instance. These tools fall into two groups:

- Tools for managing the behavior of Reporting Services.
- Tools for managing Reporting Services content.

- Tools for Managing Reporting Services behavior (native mode)
 - Reporting Services Configuration Manager
 - SQL Server Management Studio
 - SQL Server Configuration Manager
 - Rsconfig.exe
 - Rskeymgmt.exe
- Tools for managing Reporting Services content
 - Web portal
 - RS.exe

Tools for managing Reporting Services behavior

You use the tools covered in this section to manage Reporting Services running in native mode.

Reporting Services Configuration Manager

You use Reporting Services Configuration Manager to configure the Reporting Services instance, setting such properties as:

- The Web Portal and Report Server URLs.
- The name and location of the Reporting Services databases.
- Managing the encryption keys and SSL certificates that are used by Reporting Services.

Reporting Services Configuration Manager is installed when you install Reporting Services.

SQL Server Management Studio

You use SQL Server Management Studio (SSMS) to manage the running instances of Reporting Services, carrying out such actions as:

- Turning Report Server features on and off.
- Managing jobs.
- Managing shared schedules.
- Managing role definitions.

You install SSMS with a download from Microsoft.

SQL Server Configuration Manager

You use SQL Server Configuration Manager to stop and start the Reporting Services service, and to configure error reporting and memory dumps.

SQL Server Configuration Manager is installed when you install SQL Server.

Rsconfig.exe

You use the Rsconfig command-line utility to configure how Reporting Services connects to the Report Server database, and to specify an account for use in unattended processing.

Rsconfig is installed when you install Reporting Services.

Rskeymgmt.exe

You use the Rskeymgmt command-line utility to work with the Reporting Services encryption key.

Rskeymgmt is installed when you install Reporting Services.

Tools for managing Reporting Services content

You use the tools covered in this section to manage Reporting Services running in native mode.

Web portal

In addition to viewing reports, you use the web portal to manage all aspects of Reporting Services content through a web browser, including:

- Organizing the folder hierarchy.
- Configuring security.
- Configuring report execution properties.
- Configuring data sources and datasets.
- Working with schedules and subscriptions.

RS.exe

You use the RS command-line utility to manage and configure a Reporting Services instance through Visual Basic scripts. The RS utility interacts with Reporting Services through the Report Server web service.

RS is installed when you install Reporting Services.

For more information about administration tools, see the *Tools for Report Server Administration* section of the *Reporting Services Tools* topic in Microsoft Docs:

**Reporting Services Tools—Tools for Report Server Administration**

<https://aka.ms/lqwdr>

Demonstration: Working with report creation tools and data in Azure

In this demonstration, you will see how to:

- Add an Azure data source to a Report Builder report.
- Add an Azure dataset to a Report Builder report.
- Display data from both an on-premises and Azure-based database.

Demonstration Steps

Open the Sales_Order_Detail report in Report Builder

1. On the taskbar, click **Internet Explorer**.
2. In the address bar, type **mia-sql/Reports_SQL2**, and then press Enter.
3. Click the **AdventureWorks Sample Reports** folder.
4. In the upper right of the **Sales_Order_Detail** report, click the ellipsis, and then click **MANAGE**.
5. In the toolbar, click **Edit in Report Builder**.
6. In the **Internet Explorer** dialog box, clear the **Always ask before opening this type of address** check box, and then click **Allow**.
7. In the **Connect to Report Server** dialog, click **Yes**.

Add an Azure data source

1. In the **Sales_Order_Detail - Microsoft SQL Server Report Builder** window, in the **Report Data** pane, right-click **Data Sources**, click **Add Data Source**.
2. In the **Data Source Properties** dialog box, in the **Name** box, type **AdventureWorks_on_Azure**.
3. Click **Use a connection embedded in my report**.
4. In the **Select connection type** list, click **Microsoft Azure SQL Database**.
5. On the taskbar, click **Internet Explorer**.
6. In the address bar, type **portal.azure.com**, and then press Enter.
7. Use your Azure Pass credentials to sign in to the portal.
8. On the left, click **SQL databases**.
9. Click the **AdventureWorksLT** database that you created in the Preparation step.
10. Copy the **Server name** into the clipboard.
11. On the taskbar, click **Sales_Order_Detail - Microsoft SQL Server Report Builder**, and then click **Build**.
12. In the **Connection Properties** dialog box, in the **Server name** box, paste the copied server name.
13. Click **Use SQL Server Authentication**.
14. In the **User name** box, type **Student**.
15. In the **Password** box, type **Pa55w.rd**, and then select the **Save my password** check box.
16. In the **Select or enter a database name** list, click **AdventureWorksLT**, and then click **OK**.
17. In the **Data Source Properties** dialog box, click **Test Connection**.
18. In the **Test Connection Result** dialog box, click **OK**.

19. In the **Data Source Properties** dialog box, click **OK**.

Add an Azure dataset and chart

1. In the **Report Data** pane, right-click **Datasets**, and then click **Add Dataset**.
2. In the **Dataset Properties** dialog box, in the **Name** box, type **Online_Sales**, and then click **Use a dataset embedded in my report**.
3. In the **Data source** list, click **AdventureWorks_on_Azure**.
4. In the **Query** box, type the following code, and then click **OK**:

```
SELECT
    SalesOrderHeader.SalesOrderNumber
    ,Product.Name
    ,SalesOrderDetail.UnitPrice
    ,SalesOrderDetail.UnitPriceDiscount
    ,SalesOrderHeader.OrderDate
FROM
    SalesLT.SalesOrderHeader AS SalesOrderHeader
    INNER JOIN SalesLT.SalesOrderDetail AS SalesOrderDetail
        ON SalesOrderDetail.SalesOrderID = SalesOrderHeader.SalesOrderID
    INNER JOIN SalesLT.Product AS Product
        ON Product.ProductID = SalesOrderDetail.ProductID
WHERE SalesOrderHeader.SalesOrderNumber BETWEEN 'SO' + @FirstOrderID AND 'SO'+
@LastOrderID
```

5. On the **Insert** tab, in the **Data Visualizations** group, click **Chart**, and then click **Chart Wizard**.
 6. In the **New Chart** dialog box, on the **Choose a dataset** page, click **Online_Sales**, and then click **Next**.
 7. On the **Choose a chart type** page, click **Pie**, and then click **Next**.
 8. On the **Arrange chart fields** page, drag the **Name** field into the **Categories** box.
 9. Drag the **Name** field into the **Values** box, then click the drop-down list for the field, click **Count**, and then click **Next**.
 10. On the **Preview** page, click **Finish**.
 11. Resize and move the inserted pie chart above the table.
 12. Delete the Legend, and then click **Run**.
 13. In the **First Order ID** boxes, type **57033**.
 14. In the **Last Order ID** boxes, type **57033**, and then click **View Report**.
- Note that there is no data for the pie chart.
15. In the **First Order ID** boxes, type **71774**.
 16. In the **Last Order ID** boxes, type **71774**, and then click **View Report**.
 17. In the **First Order ID** boxes, type **71780**.
 18. In the **Last Order ID** boxes, type **71780**, and then click **View Report**.
 19. Note that the chart is querying data on the Azure database, and the table shows data from the on-premises database.
 20. Close all open windows, without saving any changes.

Check Your Knowledge

Question	
Which of the following tools is <i>not</i> used to design Reporting Services reports?	
Select the correct answer.	
<input type="checkbox"/>	Reporting Services web portal
<input type="checkbox"/>	SQL Server Mobile Report Designer
<input type="checkbox"/>	Report Designer
<input type="checkbox"/>	Report Builder

Lab: Exploring Reporting Services

Scenario

You have data stored in a data warehouse that is then aggregated in an Analysis Services cube, and is presented in a Reporting Services report. You have decided to compare the different ways that these systems present the data.

You will also inspect the configuration of a Reporting Services instance using different tools.

Objectives

After completing this lab, you will have:

- Explored the same data as it's presented by a data warehouse, an Analysis Services cube, and a Reporting Services report.
- Examined the configuration of a Reporting Services instance.

Estimated Time: 45 minutes

Virtual machine: **10990C-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa55w.rd**

Exercise 1: Exploring reports

Scenario

You want to compare how to query data from a data warehouse database, an Analysis Services cube, and with a Reporting Services report. You should consider how complicated each method would be for users, and how the results are presented.

The main tasks for this exercise are as follows:

1. Prepare the lab environment
2. View the design of the AdventureWorksDW data warehouse
3. View sales data from the AdventureWorksDW data warehouse
4. Display the total reseller sales from a data model
5. Display the reseller sales for each employee
6. Filter the results
7. View the data in a report

► Task 1: Prepare the lab environment

1. Ensure the **MT17B-WS2016-NAT**, **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. Run **Setup.cmd** in the **D:\Labfiles\Lab01\Starter** folder.

► Task 2: View the design of the AdventureWorksDW data warehouse

- Connect to the MIA-SQL database engine and display the **dbo.Reseller Sales** database diagram in the **AdventureWorksDW** database.

► **Task 3: View sales data from the AdventureWorksDW data warehouse**

- Open and execute the **D:\Labfiles\Lab01\Starter\SalesQuery.sql** query.

Note that the total sales for the employee **David Campbell** is **\$106,380.438**.

► **Task 4: Display the total reseller sales from a data model**

1. Connect to the **MIA-SQL** Analysis Services database.
2. Browse the **AdventureWorks** cube in the **AdventureWorksDWMultidimensional-EE** database.
3. Display the total Reseller Sales Amount for the cube.

► **Task 5: Display the reseller sales for each employee**

- Display Reseller Sales amount by employee, showing their full name. Note the sales amount for **David R. Campbell**.

► **Task 6: Filter the results**

- Filter the Reseller Sales amount to only show data for **April 2013**. Note that the total sales for the employee **David R. Campbell** is **\$106,380.438**.

► **Task 7: View the data in a report**

1. Connect to the report server at **mia-sql/Reports_SQL2/Pages/Folder.aspx**.
2. In Report Builder, run the **Employee_Sales_Summary** report from the **AdventureWorks Sample Reports** folder.
3. Changing the filters of the report, to show employee **David R. Campbell** sales data for **April 2013**.
4. Note that the total sales for the employee David R. Campbell amount to \$106,380.
5. Experiment with changing the filters at the top of the report, and click **View Report** to refresh the data.
6. Close all open windows, without saving any changes.

Results: After completing this exercise, you will have:

Explored a data warehouse.

Explored a data model.

Explored a report.

Exercise 2: Reporting Services configuration

Scenario

You want to inspect different aspects of the configuration of a Reporting Services instance using Reporting Services Configuration Manager and the web portal.

The main tasks for this exercise are as follows:

1. Use Reporting Services Configuration Manager
2. Use the web portal to view configuration

► Task 1: Use Reporting Services Configuration Manager

- Open Reporting Services Configuration Manager then inspect the configuration of the SSRS instance on MIA-SQL.

Answer the following questions:

- What is the Reporting Services edition?
- What is the Report Server mode?
- What is the Report Server URL?
- Where is the Report Server database located?
- What is the Web Portal URL?

► Task 2: Use the web portal to view configuration

- Open the web portal for the SSRS instance on MIA-SQL then go to the site settings.

Answer the following questions:

- What is the default report timeout?
- How many schedules are defined?
- How many security groups are defined?

Results: At the end of this exercise, you will have viewed Reporting Services configuration with:

Reporting Services Configuration Manager.

Reporting Services web portal.

Question: In the first lab, which method of data access seemed easiest for users to understand? Which results were presented most clearly?

Module Review and Takeaways

In this module, you've seen the tools that are provided by Microsoft to analyze data and produce business information reports. You've learned about the components of Reporting Services, and about the tools you use to create and manage reports, and configure Reporting Services. Finally, you've seen how easy it is to query data on Azure and combine different data sources inside a single Reporting Services report. The following modules in this course will go into further detail about the tools discussed in this module.

Review Question(s)

Question: Is Reporting Services already in use at your organization? If so, are any of the tools discussed in this module already in use?

Module 2

Reporting Services Data Sources

Contents:

Module Overview	2-1
Lesson 1: Data sources	2-2
Lesson 2: Connection strings	2-8
Lesson 3: Datasets	2-17
Lab A: Configuring data access with Report Builder	2-21
Lab B: Configuring data access with Report Designer	2-23
Module Review and Takeaways	2-25

Module Overview

Almost every report that you publish by using SQL Server Reporting Services (SSRS) will be built using data that's obtained from one or more source systems. This lesson explains how to configure SSRS to interact with source data systems by working with data sources.

Objectives

After completing this lesson, you will be able to:

- Describe data sources.
- Work with connection strings.
- Describe datasets.

Lesson 1

Data sources

To generate SSRS reports, you must create and manage connections to the source data that will form the basis of your reports. Even when you don't need to define data sources—because your reports are based on shared data sources that a coworker has defined for you—you still need to be able to work with data sources to use them in reports that you design.

Lesson Objectives

At the end of this lesson, you will be able to:

- Define data sources.
- Explain the differences between shared data sources and embedded data sources.
- Describe data processing extensions.

What are data sources?

A data source—sometimes referred to as a data connection—is an object that contains the information that's needed to connect to a source system for a report.

Data sources are made up of three main components:

- **The source system type.** This indicates the technology of the source system. For example, a source system might be an instance of SQL Server, a SQL Server Analysis Services (SSAS) model, or another database technology to which SSRS can connect.
- **The location of the source system.** This depends on the source system type, but it's typically a computer name or IP address.
- **Authentication credentials for the source system.** This might be a user name and password for the source system.

- An object that defines how to connect to a system that is a source for report data
- Three main properties:
 - Source system type
 - Source system location
 - Authentication credentials
- Data sources enable connection information to be managed in one place
- Data sources do not contain queries that return report data

The advantage of encapsulating the connection information in a data source object is that the connection information for a source system needs to be defined only once in a report definition, even if the report is based on many different queries from the source system. If the connection information for the source system changes—for example, if the user name and password that are used to authenticate with the source system must be updated—the change needs to be made in only one place for the report to continue to operate correctly.



Note: Data sources contain no information about the data to be retrieved from the source system. Source system queries are defined independently of the connection information, in *dataset* objects. A dataset is defined separately and is linked to a data source.

For more information about data sources, see the topic *Report Data (SSRS)*, in Microsoft Docs:



Report Data (SSRS)

<https://aka.ms/J8hrsv>

Shared and embedded data sources

There are two ways to include data sources in your reports:

- Embedded data sources
- Shared data sources

Embedded data sources

When you use an embedded data source, the data source definition is included as part of the definition of a single report. An embedded data source is not a separate object on the SSRS server, and it cannot be shared between different report definitions.

You might use an embedded data source if you want to create a report that has no external dependencies on other objects in your SSRS instance—or if a data source will only be used as the basis for a single report.

Shared data sources

A shared data source is defined independently of any report definitions that reference it, and is created as a separate object on the SSRS instance. Many different report definitions reference a single shared data source.

It's recommended that you use shared data sources wherever possible. Typically, using shared data sources will reduce the cost of administering an SSRS installation because, when connection information needs to be amended, the change is limited to the shared data source. If you have many reports that reference the same source system by using embedded data sources, you must edit each report definition if the details of the actual data source change.

Another advantage of using shared data sources is that you use the same report definition on different SSRS instances, even if the details of the data source are different for each instance. For example, if report development is carried out in a development environment that is isolated from your production system, both the development and production systems could use the same data sources. Providing a shared data source with the same name exists in both SSRS instances, report definitions that reference the shared data source will work in both environments, even if some properties of the data source are not the same.

Working with data sources

Whether you use Report Designer or Report Builder, you either always create reports that are based on existing shared data sources—or you use embedded data sources.

You cannot create new shared data sources by using Report Builder. If you use Report Builder to design a report—and you need to define a new data source in that report—you must either use an embedded data source or use another method to create a new shared data source that you reference in your report.

- **Embedded data sources:**
 - Included in a single report definition
 - Cannot be shared between reports
- **Shared data sources:**
 - Defined as independent objects on an SSRS instance
 - Can be shared by many report definitions
 - Recommended, simplified administration
- **Working with data sources:**
 - Both Report Designer and Report Builder use either embedded data sources or existing shared data sources
 - Report Builder cannot define new shared data sources
 - Report Designer or the SSRS portal are used to create new shared data sources

When you define a data source in Report Designer, you choose to make the data source either a shared data source or an embedded data source. An administrator also defines shared data sources directly by using the SSRS portal.

For more information about how to work with data sources, see the topic *Data Connections, Data Sources, and Connection Strings (Report Builder and SSRS)*, in Microsoft Docs:



Data Connections, Data Sources, and Connection Strings (Report Builder and SSRS)

<https://aka.ms/Wmtw7f>

Data processing extensions

As you saw at the start of this lesson, a data source definition includes a property that defines the source system type. Data processing extensions define the source system type of a data source and simplify the interaction between SSRS and a diverse range of data sources.

By default, SSRS includes data processing extensions for different source system types, including:

- SQL Server Database Engine
- SQL Server Analysis Services
- Microsoft SharePoint® list
- Microsoft Azure SQL Database
- Oracle
- SAP NetWeaver Business Intelligence
- Hyperion Essbase
- Teradata
- OLE DB data sources
- Open Database Connectivity (ODBC) data sources
- XML

- Data processing extensions define the source system type
- SSRS includes many data processing extensions
- You must select the correct source system type when you define a data source

Additionally, SSRS can use any Microsoft ADO.NET data provider.



Note: You write your own custom data processing extensions if you need to connect to data sources that SSRS does not currently support. Doing so requires detailed programming knowledge; this topic is beyond the scope of this course.

You must select the correct source system type when you define a data source; otherwise, the data source is unlikely to function correctly, if it functions at all.

For more information about data processing extensions, see the topic *Extensions (SSRS)*, in Microsoft Docs.



Extensions (SSRS)

<https://aka.ms/P37xys>

For a complete list of data sources that SSRS supports, see the topic *Data Sources Supported by Reporting Services (SSRS)*, in Microsoft Docs.



Data Sources Supported by Reporting Services (SSRS)

<https://aka.ms/Fa2r4m>

Demonstration: Creating a data source

In this demonstration, you will see how to:

- Use the SSRS portal to create a shared data source.
- Use Report Designer to define a shared data source.
- Use Report Builder to define an embedded data source.

Demonstration Steps

Use the SSRS portal to create a shared data source

1. Ensure that the **MT17B-WS2016-NAT**, **10990C-MIA-DC**, and **10990C-MIA-SQL** virtual machines are running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Demofiles\Mod02** folder, run **Setup.cmd** as administrator.
3. In the **User Account Control** dialog box, click **Yes**.
4. In Windows® Internet Explorer®, browse to **mia-sql/Reports_SQL2**, and then click **DataSources**.
5. On the toolbar, click **+**, and then click **Data Source**.
6. On the **New data source** page, in the **Name** box, type **AdventureWorksDW-portal**.
7. In the **Type** box, verify that **Microsoft SQL Server** is selected.
8. In the **Connection string** box, type the following code:

```
data source="(local)";initial catalog=AdventureWorksDW
```

9. Click **Test connection**. Note that the test succeeds, and then click **Create**.
10. Leave Internet Explorer open for the next demonstration.

Use Report Designer to define a shared data source

1. Start Microsoft Visual Studio® and, on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, in **D:\Demofiles\Mod02\Demo**, click **Demo.sln**, and then click **Open**. This opens a new, empty SSRS project.
3. If a security warning appears, click **OK**.
4. In Solution Explorer, right-click **Shared Data Sources**, and then click **Add New Data Source**.

5. In the **Shared Data Source Properties** dialog box, in the **Name** box, type **AdventureWorksDW-SSDT**.
6. In the **Type** box, verify that **Microsoft SQL Server** is selected, and then click **Edit**.
7. In the **Connection Properties** dialog box, in the **Server name** box, type **MIA-SQL**.
8. In the **Select or enter a database name** box, select **AdventureWorksDW**, and then click **Test Connection**.
9. In the **Test results** dialog box, click **OK**.
10. In the **Connection Properties** dialog box, click **OK**.
11. In the **Shared Data Source Properties** dialog box, on the **Credentials** page, point out that you can amend the credentials that are used for the data source. Verify that **Use Windows Authentication** is selected, and then click **OK**.
12. In Solution Explorer, expand **Shared Data Sources**, and then verify that the definition for the data source has been created. Point out that the data source definition is a file with an **.rds** extension.
13. In Internet Explorer, on the **SQL Server Reporting Services** page, click **Home**, notice that the **AdventureWorksDW-SSDT** data source does not yet exist on the server; it must be deployed.
14. In Visual Studio, in Solution Explorer, right-click **AdventureWorksDW-SSDT.rds**, and then click **Deploy**. Wait for the deployment to succeed.
15. In Internet Explorer, on the **SQL Server Reporting Services** page, click **Refresh**.
16. Click **Designer**, and verify that the **AdventureWorksDW-SSDT** data source has now been created.

Use Report Builder to define an embedded data source

1. In Internet Explorer, click **Home**.
2. On the toolbar, click **+**, and then click **Paginated Report**.
3. In the **Internet Explorer** dialog box, click **Allow**.
4. In the **Connect to Report Server** dialog box, click **Yes**.
5. In the **New Report or Dataset** dialog box, click **Blank Report**.
6. In the **Report Data** pane, right-click **Data Sources**, and then click **Add Data Source**.
7. In the **Data Source Properties** dialog box, on the **General** page, in the **Name** box, type **AdventureWorksDW_embedded**, and then click **Use a connection embedded in my report**.
8. In the **Select connection type** box, verify that **Microsoft SQL Server** is selected, and then click **Build**.
9. In the **Connection Properties** dialog box, in the **Server name** box, type **MIA-SQL**.
10. In the **Select or enter a database name** list, click **AdventureWorksDW**, and then click **Test Connection**.
11. In the **Test results** dialog box, click **OK**.
12. In the **Connection Properties** dialog box, click **OK**.
13. In the **Data Source Properties** dialog box, on the **Credentials** page, point out that you can amend the credentials that you use for the data. Note the warning message: **This information is only stored when you save the report to a report server**. Verify that **Use current Windows user** is selected, and then click **OK**.

14. On the **File** menu, click **Save**.
15. In the **Save As Report** dialog box, double-click **DataSources**.
16. In the **Name** box, type **report_builder.rdl**, and then click **Save**.
17. In Internet Explorer, close the **We're opening Report Builder** window if it's displayed, and then click **DataSources**. Note that, although the report definition has been saved to the folder, there is not an independent data source definition for the data source that's used in the report.
18. In the **report_builder** object, in the upper right corner, click the ellipsis (...) button, and then click **Manage**.
19. On the **Manage report_builder** page, click **Data sources**, and verify that you can amend the properties of an embedded connection after the report is published. When you have finished examining the report properties, click **DataSources**.
20. Leave Internet Explorer and Report Builder open for the next demonstration.

Check Your Knowledge

Question	
You want to create a data source that several reports will use. Which type of data source should you create?	
Select the correct answer.	
<input type="checkbox"/>	A shared data source.
<input type="checkbox"/>	An embedded data source.
<input type="checkbox"/>	Either a shared data source or an embedded data source; it does not matter which.
<input type="checkbox"/>	<Enter option 4 here> <Enter option 4 here>
<input type="checkbox"/>	<Enter option 5 here> <Enter option 5 here>

Lesson 2

Connection strings

SSRS uses connection strings to locate data sources for reports and to correctly configure a connection to a data source to retrieve data. This lesson describes the connection strings for the different types of system that you might use when you configure data sources for your reports.

Lesson Objectives

At the end of this lesson, you will be able to:

- Explain the purpose of connection strings.
- Work with SQL Server database engine connection strings.
- Work with SSAS Services connection strings.
- Recognize connection strings for other data providers.

What are connection strings?

“Connection string” is a widely used computing term that refers to the information that is required to configure a connection from an application to a data source, stored as a delimited string. A connection string typically appears in the format that’s shown in the following code example:

```
property name=value; [property  
name=value; ...]
```

A connection string might be made up of many properties and values. When an application attempts to connect to a data source, the connection string is passed as an argument to the software component that’s responsible for establishing the connection. This software component is often referred to as a *data provider* or *database driver*.

Although the contents of the connection string vary according to the data source type and the environment where the connecting application is running, all connection strings might contain several different types of information, such as:

- The location of the data source.
- Authentication credentials.
- Properties to further configure the data source.

Data source location

The data provider uses the data source location to find the data source; a data source location is common to all connection strings, although its form might vary, based on the data source type and the configuration of your network. Most commonly, the data source location takes the form of a server name or an IP address, but it might take the form of a URL or a file system location. In some more complex environments, the data source location information in a connection string might also include the names of secondary servers to be used if the primary server is inaccessible.

- A string of property and value pairs that configure a data connection
- Property types include:
 - Data source location:
 - A server name, IP address, URL, or file name
 - Authentication credentials:
 - For example, user name and password
 - Additional properties:
 - For example, a database name

Authentication credentials

Authentication credentials typically take the form of a user name and password or an instruction to use the user's Windows credentials. You can also configure an SSRS data source to request Windows credentials from the user who is running a report that uses the data source. Not every type of data source requires an authentication method to be explicitly nominated; many data providers have a default authentication method that's used if one is not included in the connection string. For example, a SQL Server Native Client connection will use Windows authentication if no authentication method is specified in the connection string.



Note: You might encounter connection strings in other data-driven applications that include a user name and password in the connection string. Storing sensitive data—such as user names and passwords in plain text—in a connection string, might be a security risk. Therefore, SSRS reports only use securely stored authentication credentials that are independent of the connection string. When SSRS connects to a data source to generate a report, the credentials are recovered from secure storage then added to the connection string.

Additional properties

Additional connection string properties are used to configure aspects of the connection after it's initiated—for example, connection string properties might define the database name that the connection should attempt to use, or whether the connection requires encryption. Although connection strings for most source data system types require a data source location and an authentication method, additional properties are typically optional. Each source system type will support a different group of additional properties.

SQL Server Database Engine connection strings

Although you might configure SQL Server in several different ways, the connection string that's required to use a database engine instance as a data source is similar for each of them. For all SQL Server Database Engine connections, the data source location is defined by using the data source property. The database that the connection should open is defined by using the initial catalog property.



Note: Remember that the authentication details—a user name and password or Windows authentication—are not specified in the connection string for SSRS data sources. Instead, authentication details are specified separately and held in secure storage on the report server.

- Connection string properties:
 - Location: *data source* property
 - Database: *initial catalog* property

- SQL Server Database Engine

```
data source=DBServer\SQL3;initial catalog=AdventureWorksDW
```

- Azure SQL Database

```
data source=azuresql.database.windows.net;  
initial catalog=AdventureWorksDW;  
Encrypt=True;  
TrustServerCertificate=False
```

SQL Server Database Engine

The following examples show the connection string that's required to connect to database engine instances that are installed in different ways. You use these connection strings with the data source type *Microsoft SQL Server*.

The following example shows how to connect to the default database engine instance that's installed on the same computer as the SSRS instance, starting in the AdventureWorksDW database:

Connecting to the default instance on the local instance of SQL Server

```
data source="(local)";initial catalog=AdventureWorksDW
```

The following example shows how to connect to a named database engine instance—called SQL2—that's installed on the same computer as the SSRS instance, starting in the AdventureWorksDW database:

Connecting to a named instance on the local instance of SQL Server

```
data source=localhost\SQL2;initial catalog=AdventureWorksDW
```

The following example shows how to connect to the default database engine instance that's installed on a different computer from the SSRS instance—called DBServer—starting in the AdventureWorksDW database:

Connecting to the default instance on a remote instance of SQL Server

```
data source=DBServer;initial catalog=AdventureWorksDW
```

The following example shows how to connect to a named database engine instance—called SQL3—that's installed on a different computer from the SSRS instance—called DBServer—starting in the AdventureWorksDW database:

Connecting to a named instance on a remote instance of SQL Server

```
data source=DBServer\SQL3;initial catalog=AdventureWorksDW
```



Note: When you define connections to named database engine instances in Report Designer or Report Builder, the slash in the data source property of the connection string will be doubled, as the following code example shows:

```
DBServer\\SQL3
```

This is to prevent the slash being interpreted as a special character. When the data source is saved to a report server instance, the extra slash is removed.

Azure SQL Database

When you use Azure SQL Database as a data source for an SSRS report, the connection string is almost identical to the connection string for a SQL Server Database Engine instance. You use the data source type *Microsoft SQL Azure* with this connection string.

The following example shows how to connect to an Azure SQL Database called AdventureWorksDW that runs on an Azure SQL Database server called azuresql:

Connecting to an Azure SQL Database

```
data source=azuresql.database.windows.net;initial  
catalog=AdventureWorksDW;Encrypt=True;TrustServerCertificate=False
```



Note: Connection strings for database engine instances and Azure SQL Database might include more than 30 optional additional parameters. For a complete list, in Report Builder or Report Designer, in the Connection Properties window, click **Advanced**.

For more information about how to use SQL Server Database Engine connections in SSRS, see the topic *SQL Server Connection Type (SSRS)*, in Microsoft Docs.

SQL Server Connection Type (SSRS)

<https://aka.ms/Jmpbmk>

For more information about how to use Azure SQL Database connections in SSRS, see the topic *SQL Azure Connection Type (SSRS)*, in Microsoft Docs.

SQL Azure Connection Type (SSRS)

<https://aka.ms/Ff6e9z>

SSAS connection strings

When you install SSAS, you configure it to support either the multidimensional model or the tabular model. Regardless of the model type, the connection strings that are required to use SSAS as a data source for an SSRS report are very similar. As with the SQL Server Database Engine, you define the data source location by using the data source property. You define the database that the connection should open by using the initial catalog property.

- Connection string properties:
 - Location: **data source** property
 - Database: **initial catalog** property

- SSAS multidimensional

```
data source=AS_SERVER;initial catalog=AWCube2
```

- SSAS tabular

```
data source=AS_TAB_SERVER;initial catalog=AW3;Cube='Sales'
```

Multidimensional models

The connection string for a multidimensional SSAS data source is the same whether you plan to write queries by using the Multidimensional Expressions (MDX) or Data Mining Extensions (DMX) language. The following examples show connection strings that are used for multidimensional SSAS instances in typical configurations. You use these connection strings with the **Microsoft SQL Server Analysis Services** data source type.

The following example shows how to connect to a multidimensional SSAS instance that's installed on the same computer as the SSRS instance, starting in the **AWCube** model:

Connecting to an SSAS local multidimensional instance

```
data source=localhost;initial catalog=AWCube
```

The following example shows how to connect to a multidimensional SSAS instance that's installed on a different computer from the SSRS instance—AS_SERVER—starting in the **AWCube2** model:

Connecting to an SSAS remote multidimensional instance

```
data source=AS_SERVER;initial catalog=AWCube2
```

Tabular models

The connection string for a tabular SSAS data source is the same as for a multidimensional SSAS data source, with an additional optional property that helps you to define which tabular perspective to use. You also use the **Microsoft SQL Server Analysis Services** data source type to connect to a tabular SSAS model.

The following example shows how to connect to a tabular SSAS instance that's installed on a different computer from the SSRS instance—AS_TAB_SERVER—starting in the **AW3** model and using the **Sales** perspective:

Connecting to an SSAS local tabular instance

```
Data source=AS_TAB_SERVER;initial catalog=AW3;cube= 'Sales'
```



Note: Connection strings for SSAS data sources might include more than 60 optional additional parameters. For a complete list, in Report Builder or Report Designer, in the Connection Properties window, click **Advanced**.

For more information about how to use SSAS multidimensional MDX queries as SSRS data sources, see the topic *Analysis Services Connection Type for MDX (SSRS)* in Microsoft Docs.



Analysis Services Connection Type for MDX (SSRS)

<https://aka.ms/Ngwuib>

For more information about how to use SSAS multidimensional DMX queries as SSRS data sources, see the topic *Analysis Services Connection Type for DMX (SSRS)* in Microsoft Docs.



Analysis Services Connection Type for DMX (SSRS)

<https://aka.ms/B6d5df>

Connection strings for other providers

Each data source type that SSRS supports has a different connection string, although many data source types use a pattern that's similar to the connection strings for SQL Server that you saw earlier in this lesson. The following examples show sample connection strings for some commonly used data sources.

SQL Server Parallel Data Warehouse

The following example shows a connection string for an instance of SQL Server Parallel Data Warehouse that runs on port 6000 of the 10.0.0.1 IP address, and uses the AdventureWorksDW database:

- SQL Server Parallel Data Warehouse
- Oracle
- Report server model
- SharePoint list
- SAP Netweaver BI
- Hyperion Essbase
- Teradata
- XML
- OLE DB and ODBC

Connecting to an instance of SQL Server Parallel Data Warehouse

```
HOST=10.0.0.1;database=AdventureWorksDW;port=6000
```

Oracle

To use an Oracle data source, the Oracle client software must be installed on the SSRS server.

The following example shows a connection string for an Oracle instance—**AWOracle**—that is defined in the local TNSNAMES.ORA file:

Connecting to Oracle

```
data source=AWOracle
```

Report server model

The connection string for a report server model will depend on whether the report server instance is running in native mode or SharePoint integrated mode.

The following example shows a connection string for a report server instance that's running in native mode—**ModelServer**—accessing a model called **AWModel** in the ReportModels folder:

Connecting to a report server model by using native mode

```
Server=http://ModelServer/reportserver; datasource=/ReportModels/AWModel
```

The following example shows a connection string for a report server instance that's running in SharePoint integrated mode—**ModelServerSP**—accessing a model called **AWModel** in the ReportModels folder:

Connecting to a report server model by using SharePoint integrated mode

```
Server=http://ModelServerSP;  
datasource=http://ModelServerSP/site/documents/ReportModels/AWModel.smdl
```

SharePoint list

The connection string for a SharePoint list includes the server name, site name, and any subsite names. The group of available lists in the site that's specified in the connection string is shown in the query designer.

The following example shows a connection string for SharePoint lists in the AdventureWorks site on the AWSP instance of SharePoint Server:

Connecting to a SharePoint list

```
data source=http://AWSP/AdventureWorks/
```

SAP NetWeaver BI

The following example shows a connection string for a SAP NetWeaver BI instance that runs on port 9000 of the AWSAP server:

Connecting to SAP NetWeaver BI

```
DataSource=http://AWSAP:9000/sap/bw/xml/soap/xmla
```

Hyperion Essbase

The following example shows a connection string for a Hyperion Essbase instance that runs on port 13090 of the AWEssbase server and works in the AWSales database:

Connecting to Hyperion Essbase

```
Data Source=http://AWEssbase:13090/aps/XMLA;Initial Catalog=AWSales
```

Teradata

To use a Teradata data source, the Teradata client software must be installed on the SSRS server.

The following example shows a connection string for a Teradata instance that runs on port 10.0.0.2 and connects to the AWtera database:

Connecting to Teradata

```
data source=10.0.0.2;database=AWtera
```

XML

You use XML that's returned from a web service, is in a file, or is embedded in a report definition as a data source.

The following example shows a connection string for an XML document that's returned by a call to the **sales-report.aspx** page on the AWReports web server:

Connecting to an XML web service

```
http://AWReports/sales-report.aspx
```

The following example shows a connection string for the CurrentPriceList.xml document on the AWPUBLIC web server:

Connecting to an XML file

```
http://AWPublic/CurrentPriceList.xml
```

An XML document that's embedded in a report definition has no connection string when it's used as a data source.

OLE DB and ODBC

The connection string for an OLE DB or ODBC data source will be formatted according to the rules for the data provider. There are many OLE DB and ODBC data providers that have different connection string formats.

For more information about data source connection strings, see the topic *Add Data from External Data Sources (SSRS)* in Microsoft Docs.



Add Data from External Data Sources (SSRS)

<https://aka.ms/E4ajn2>

Demonstration: Working with Azure SQL Database connection strings

In this demonstration, you will see how to create a connection to Azure SQL Database from:

- Report Builder
- Report Designer

Demonstration Steps

Connect to Azure SQL Database from Report Builder

1. In Report Builder, in the **Report Data** pane, right-click **Data Sources**, and then click **Add Data Source**.
2. In the **Data Source Properties** dialog box, on the **General** page, in the **Name** box, type **AdventureWorksLT**, and then click **Use a connection embedded in my report**.
3. In the **Select connection type** list, click **Microsoft Azure SQL Database**. Notice that two optional properties are added to the connection string, and then click **Build**.
4. In the **Connection Properties** dialog box, in the **Server name** box, type the name of your Azure SQL Database server; this needs to be the fully qualified name, ending in **.database.windows.net**.
5. Verify that **Use SQL Server Authentication** is selected, in the **User name** box type **Student**, in the **Password** box type **Pa55w.rd**.
6. In the **Select or enter a database name** box, type **AdventureWorksLT**, and then click **Test Connection**.
7. In the **Test results** dialog box, click **OK**.
8. In the **Connection Properties** dialog box, click **OK**.
9. In the **Data Source Properties** dialog box, click **OK**.
10. On the **File** menu, click **Save**.
11. Leave Report Builder open for the next demonstration.

Connect to Azure SQL Database from Report Designer

1. In Visual Studio, in Solution Explorer, right-click **Shared Data Sources**, and then click **Add New Data Source**.
2. In the **Shared Data Source Properties** dialog box, in the **Name** box, type **AdventureWorksLT**.
3. In the **Type** list, click **Microsoft Azure SQL Database**, and then click **Edit**.
4. In the **Connection Properties** dialog box, in the **Server name** box, type the name of your Azure SQL Database server; this needs to be the fully qualified name, ending in **.database.windows.net**.
5. In the **Authentication** box, verify that **SQL Server Authentication** is selected, in the **User name** box type **Student**, in the **Password** box type **Pa55w.rd**.
6. Select the **Save my password** check box.
7. In the **Select or enter a database name** list, click **AdventureWorksLT**, and then click **Test Connection**.
8. In the **Test results** dialog box, click **OK**.
9. In the **Connection Properties** dialog box, click **OK**.
10. In the **Shared Data Source Properties** dialog box, click **OK**.

11. Leave Visual Studio open for the next demonstration.

Check Your Knowledge

Question	
Which of the following connection strings will configure a SQL Server connection to connect to the AWSales database on the Sales database engine instance on a server called SALES2?	
Select the correct answer.	
<input type="checkbox"/>	data source=SALES2;initial catalog=Sales\AWSales
<input type="checkbox"/>	data source=SALES2\Sales;initial catalog=AWSales
<input type="checkbox"/>	data source=SALES2\Sales;initial catalog=AdminDB
<input type="checkbox"/>	data source=SALES2;initial catalog=AWSales
<input type="checkbox"/>	None of the above

Lesson 3

Datasets

After you have defined a connection to a source system by using a data source, you must define how the data you want to include in your report will be retrieved from the source system; you do this by using the dataset object. This lesson is an introduction to working with datasets.

Lesson Objectives

At the end of this lesson, you will be able to:

- Describe datasets.
- Explain the differences between shared datasets and embedded datasets.

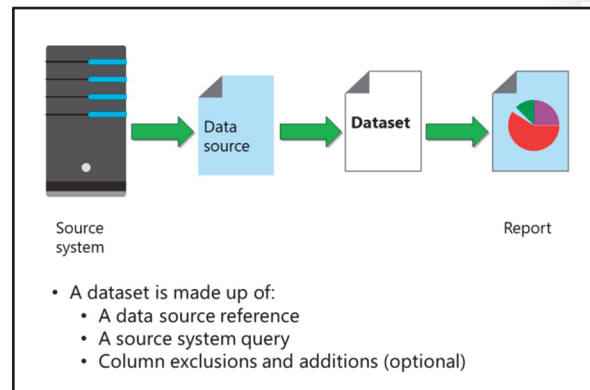
What is a dataset?

A dataset defines a result set to use as the basis for a report. It defines a query—or other data extraction statement—that's executed against a data source. Therefore, the two most important properties of a dataset are:

- Data source
- Query text



Note: The dataset does not contain result set data; instead, it contains the definition of how a result set will be acquired from a data source.



Data source

The data source for a dataset must be a data source that already exists. The data source might be an embedded data source that's included in the report definition, or it can be a shared data source on the report server. A dataset only contains a link to the data source name; no other information about the data source is stored in the dataset definition.

Query text

The query text in a dataset takes the form of a query statement in a language that's appropriate to the data source type. For example, if the data source for a dataset is a SQL Server Database Engine instance, the query text is a Transact-SQL statement.

Column exclusions and additions

You don't have to include all of the columns that the source system query returns. You can also add calculated columns to a dataset.

For more information about datasets, see the topic *Report Datasets (SSRS)* in Microsoft Docs.



Report Datasets (SSRS)

<https://aka.ms/Gs3hay>

Shared and embedded datasets

Similar to the data sources that you encountered earlier in this module, datasets are published in their own right and shared between many reports—or they might be embedded in a report definition.

Embedded datasets

When you use an embedded dataset, the dataset definition is included as part of the definition of a single report. An embedded dataset is not a separate object on the SSRS server, and it cannot be shared between different report definitions.

- **Embedded datasets:**
 - Included in a single report definition
 - Cannot be shared between reports
- **Shared datasets:**
 - Defined as independent objects on an SSRS instance
 - Can be shared by many report definitions
 - Recommended—improved performance because of caching
- **Working with datasets:**
 - Use Report Designer or Report Builder to embed datasets or shared datasets in report definitions
 - Use Report Builder or Report Designer to define shared datasets
 - Use Report Builder, Report Designer, or the SSRS portal to manage shared datasets

You might use an embedded dataset if you want to create a report that has no external dependencies on other objects in your SSRS instance or if a dataset will only be used as the basis for a single report.



Note: You create an embedded dataset by using either an embedded data source or a shared data source.

Shared datasets

A shared dataset is defined independently of any report definitions that reference it and is created as a separate object on the SSRS instance. Many different report definitions can reference a single shared dataset.



Note: A shared dataset must be based on a shared data source. A shared dataset cannot be based on an embedded data source.

Using shared datasets might improve report performance, because the results from a shared dataset are temporarily stored on the report server in a cache. This means that reports that are based on the shared dataset can be generated without the dataset query being executed for each run of the report. If a report is frequently executed, or the dataset query takes a long time to execute, caching the results from a shared dataset might significantly improve the responsiveness of an SSRS instance. For this reason, it's recommended that you use shared datasets whenever possible.

Working with datasets

Reports that you create in Report Designer or Report Builder might include both embedded datasets and shared datasets. You create shared datasets by using Report Designer or Report Builder; you manage shared datasets by using the Reporting Services portal, Report Designer, or Report Builder.

For more information about shared and embedded datasets, see the topic *Embedded and Shared Datasets (Report Builder and SSRS)* in Microsoft Docs.



Embedded and Shared Datasets (Report Builder and SSRS)

<https://aka.ms/R3dnf>

Demonstration: Creating a shared dataset

In this demonstration, you will see how to:

- Use Report Designer to create a shared dataset.
- Use Report Builder to create a shared dataset.

Demonstration Steps

Use Report Designer to create a shared dataset

1. In Visual Studio, in Solution Explorer, right-click **Shared Datasets**, and then click **Add New Dataset**.
2. In the **Shared Dataset Properties** dialog box, in the **Name** box, type **AssociatedOrders-SSDT**.
3. In the **Data source** list, click **AdventureWorksDW-SSDT**, and then click **Query Designer**.
4. In the **Query Designer** dialog box, click **Add Table**.
5. In the **Add Table** dialog box, on the **Views** tab, click **vAssocSeqOrders**, click **Add**, and then click **Close**.
6. In the Diagram pane, in the **vAssocSeqOrders** object, select **OrderNumber**, **CustomerKey**, **Region**, and **IncomeGroup**.
7. Click the **Run** button (!) to preview the results of the query. Briefly review the data in the Result pane then click **OK**.
8. In the **Shared Dataset Properties** dialog box, click **OK**.
9. In Solution Explorer, expand **Shared Datasets**, right-click **AssociatedOrders-SSDT.rsd**, and then click **Deploy**.
10. In Internet Explorer, on the **SQL Server Reporting Services** page, click **Home**, and then click **Designer**. Verify that the **AssociatedOrders-SSDT** shared data source has been created. If the shared data source does not appear, refresh the page. When you have finished, close Visual Studio.

Use Report Builder to create a shared dataset

1. In Report Builder, on the **File** menu, click **New**.
2. In the **New Report or Dataset** dialog box, click **New Dataset**, and then click **Browse other data sources**.
3. In the **Select Data Source** dialog box, double-click **DataSources**, click **AdventureWorksDW-portal**, and then click **Open**.
4. In the **New Report or Dataset** dialog box, click **Create**. Report Builder will switch to the Shared Dataset Design view.
5. In the **Database view** pane, expand **Views**, select **vAssocSeqOrders**, and then click **Edit as Text** to switch to the Query Designer view.
6. Click the **Run** button (!) to preview the output of the query. Briefly examine the query results and then, on the **File** menu, click **Save**.
7. In the **Save As Dataset** dialog box, verify that the value of the **Look in** box is **http://mia-sql/ReportServer_SQL2/DataSources**. In the **Name** box, type **AssociatedOrders-builder**, and then click **OK**.

8. In Internet Explorer, on the **SQL Server Reporting Services** page, click **Home**, and then click **DataSources**. Verify that the **AssociatedOrders-builder** shared dataset appears on the report server portal.
9. When you have finished, close Report Builder, Internet Explorer and Visual Studio.

Verify the correctness of the statement by placing a mark in the column to the right.

Statement	Answer
True or false? A dataset definition includes a connection string.	

Lab A: Configuring data access with Report Builder

Scenario

As a reports developer at Adventure Works Cycles, you need to configure a data source, and a shared dataset, to facilitate future development of SSRS reports.

Objectives

After completing this lab, you will be able to use Report Builder to create shared data sources and shared datasets.

Estimated Time: 45 minutes

Virtual machine: **10990C-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa55w.rd**

Exercise 1: Configuring data access

Scenario

You expect to create many reports that are based on the AdventureWorksDW database, so your first task is to create a shared data source that connects to this database. You cannot create shared data sources by using Report Builder, so you will create the shared data source by using the SSRS portal.

After you configure the shared data source, you will create a shared dataset that returns all of the rows and columns from the **dbo.vAllSales** view.

The main tasks for this exercise are as follows:

1. Prepare the lab environment
2. Create a shared data source
3. Create a shared dataset

► Task 1: Prepare the lab environment

1. Ensure that the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab02\Starter** folder, run **Setup.cmd** as administrator.

► Task 2: Create a shared data source

1. In Internet Explorer, use the SSRS portal at **mia-sql/Reports_SQL2** to create a shared data source called **AdventureWorksDW** in the Builder folder. Use the following information to configure the data source:
 - **Data source type:** Microsoft SQL Server
 - **Server name:** MIA-SQL
 - **Database name:** AdventureWorksDW
 - **User name:** reporting
 - **Password:** Pa55w.rd
2. Use the **Test Connection** functionality to verify that you can connect to the data source.

► Task 3: Create a shared dataset

1. Using Internet Explorer, start Report Builder from **http://mia-sql/Reports_SQL2**.
2. Using the shared data source that you created in the previous task, create a shared dataset. The dataset should return all of the rows and columns from the **dbo.vAllSales** view in the **AdventureWorksDW** database.
3. Save the dataset in the Builder folder on the SSRS server. Give the dataset the name **AllSales**.
4. In Internet Explorer, browse to **http://mia-sql/Reports_SQL2/browse/Builder** to verify that the dataset has been created.
5. Close Report Builder, and then close Internet Explorer.

Results: At the end of this lab, a shared data source and a shared dataset will exist in the Builder folder of the Reporting Services portal at **http://mia-sql/Reports_SQL2**.

Verify the correctness of the statement by placing a mark in the column to the right.

Statement	Answer
True or false? A data source includes the text of the query needed to return a result set.	

Lab B: Configuring data access with Report Designer

Scenario

As a reports developer at Adventure Works Cycles, you need to configure a data source, and a shared dataset, to facilitate future development of SSRS reports.

Objectives

After completing this lab, you will be able to use Report Designer to create shared data sources and shared datasets.

Estimated Time: 45 minutes

Virtual machine: **10990C-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa55w.rd**

Exercise 1: Configuring data access

Scenario

You expect to create many reports that are based on the **AdventureWorksDW** database, so your first task is to create a shared data source that connects to this database.

After you configure the shared data source, you will create a shared dataset that returns all of the rows and columns from the **dbo.vAllSales** view.

The main tasks for this exercise are as follows:

1. Prepare the lab environment
2. Create a shared data source
3. Create a shared dataset

► Task 1: Prepare the lab environment

1. Ensure that the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab02\Starter** folder, run **Setup.cmd** as administrator.

► Task 2: Create a shared data source

1. Using Visual Studio, in **D:\Labfiles\Lab02\Starter\Project**, open the project file **Project.sln**.
2. Create a shared data source that has the name **AdventureWorksDW**. Use the following information to configure the data source:
 - **Data source type:** Microsoft SQL Server
 - **Server name:** MIA-SQL
 - **Authentication:** SQL Server
 - **User name:** reporting
 - **Password:** Pa55w.rd
 - **Saved password:** Yes
 - **Database name:** AdventureWorksDW

3. When the data source definition is complete, deploy the shared data source.

► **Task 3: Create a shared dataset**

1. Using the shared data source that you created in the previous task, create a shared dataset. The dataset should return all of the rows and columns from the **dbo.vAllSales** view in the AdventureWorksDW database. Give the dataset the name **AllSales**.
2. Deploy the dataset to the Designer folder of the SSRS instance.
3. In Internet Explorer, browse to **http://mia-sql/Reports_SQL2_Preview** to verify that the data source and dataset have been created in the Designer folder.
4. Close Visual Studio, and then close Internet Explorer.

Results: At the end of this lab, a shared data source and a shared dataset will exist in the Designer folder of the Reporting Services portal at **http://mia-sql/Reports_SQL2**.

Verify the correctness of the statement by placing a mark in the column to the right.

Statement	Answer
True or false? A data source includes the text of the query needed to return a result set.	

Module Review and Takeaways

In this module, you have seen how to work with data sources and datasets to bring data from source systems into your SSRS reports.



Best Practice: To simplify the management and administration of your SSRS servers, use shared data sources whenever possible.

Review Question(s)

Check Your Knowledge

Question	
Which SSRS data access object would you typically create first when you configure data access to a source system?	
Select the correct answer.	
<input type="checkbox"/>	An embedded dataset
<input type="checkbox"/>	A shared dataset
<input type="checkbox"/>	A connection string
<input type="checkbox"/>	A data source

MCT USE ONLY. STUDENT USE PROHIBITED

Module 3

Creating Paginated Reports

Contents:

Module Overview	3-1
Lesson 1: Creating a report with the Report Wizard	3-2
Lesson 2: Creating a report	3-7
Lesson 3: Publishing a report	3-15
Lab: Creating reports	3-20
Module Review and Takeaways	3-25

Module Overview

Now that you have learned about BI and data modeling, and how to access data from Report Designer and Report Builder, you need to learn how to create reports. This module shows you how to create different types of reports in both applications, in addition to using the Report Wizard.

Objectives

After completing this module, you should be able to:

- Create a report with the Report Wizard.
- Create a report using Report Designer or Report Builder.
- Publish a report to Reporting Services.

Lesson 1

Creating a report with the Report Wizard

The Report Wizard offers a quick and efficient way to develop reports in SQL Server Data Tools and Report Builder. In this module, you will see how to create a report graphically in the Report Wizard.

Lesson Objectives

After completing this lesson, you will be able to use the Report Wizard to:

- Define a data source.
- Design a query.
- Specify a report type.

Define a data source

You start the Report Wizard from within Visual Studio® in a SQL Server Data Tools Report Solution—in Solution Explorer, right-click **Reports**, and then click **New Report**. In Report Builder, on the **File** menu, click **New**, and then click **Table or Matrix Wizard**.

After the Welcome page, you will see the Select the Data Source page where you either select an existing data source or create a new one.

Microsoft SQL Server Reporting Services (SSRS) supports the following data source types:

- Microsoft SQL Server Database
- Microsoft SQL Server Analysis Services
- Microsoft Azure SQL Database
- SQL Server Parallel Data Warehouse
- Oracle
- SAP NetWeaver BI
- Hyperion Essbase
- Microsoft SharePoint® List
- Teradata
- OLE DB
- ODBC
- XML

Supported data source types:

- SQL Server Database
- SQL Server Analysis Services
- Azure SQL Database
- SQL Server Parallel Data Warehouse
- Oracle
- SAP NetWeaver BI
- Hyperion Essbase
- Microsoft SharePoint List
- Teradata
- OLE DB
- ODBC
- XML

- In Report Wizard, use an existing shared data source or create a new data source

You create and register support for custom data source types on the system. You also download many third-party .NET Framework data providers from Microsoft and third-party developers' sites.

You define a data source in the Report Wizard by selecting an existing shared data source on the Select the Data Source page or by defining a new data source connection. When defining a new data source, you specify the connection string information and credentials to use to access the data source. You might need to install connection drivers for some data sources before you access them.

For more information on data sources and connection strings, see the topic *Data Connections, Data Sources, and Connection Strings (Report Builder and SSRS)* in Microsoft Docs:

 **Data Connections, Data Sources, and Connection Strings (Report Builder and SSRS)**

<https://aka.ms/Wmtw7f>

Design a query

After selecting or configuring a new data source connection, you must define the dataset for your report. In the Report Wizard, on the Design the Query page, you type a query directly to define the dataset. The query syntax required depends on the type of data source you are querying. If the data source supports the Query Designer graphical tool, you use it to generate the query.

You use Query Designer to specify aliases, sort types, sort orders, and filters in an easy-to-use graphical environment that generates the required query for you. Query Designer also gives you the option to edit the query as text, and therefore manually edit and amend the produced SQL.

- Write query manually
- Use the Query Builder (for supported data types)
- Queries in SQL Server
 - Tables, views, functions, stored procedures and synonyms
- For better performance
 - Perform grouping, aggregation, and sorting on the database

Queries possible with SQL Server

SSRS supports any valid SELECT DML query. It's therefore possible to create SQL statements that return data from tables, views, functions, stored procedures, synonyms—or a combination of these.

Performance considerations

When designing reports, a primary consideration should be the time it takes for a report to be displayed to the end user. Most of the processing time to display a report is taken in performing and returning the data for a query. A common performance improvement for an SSRS report is to perform as much grouping, aggregation, and sorting of data on the database engine instead of within a report.

For further considerations regarding queries, see the next lesson on creating datasets.

Specify a report type and design the report

After designing the query for the data you need in your report, the next step is to choose a report type—tabular or matrix. You do this on the Select a Report Type page of the Report Wizard.

Tabular reports display data in rows, while matrix reports show data fields grouped in rows and columns, with aggregated values in the intersecting cells, like a PivotTable or crosstab.

When you have chosen your report type, you then need to design the report on the Design the Table or Design the Matrix page. To design the report, select your required field in the Available fields list and add it to the report in the Page, Column, Row, or Detail area. You can select more than one field for each section of the report. Reorder fields by using the up and down arrows on the page.

- Two report types available:
 - Tabular: displays data in rows
 - Matrix: groups data fields with values at intersection points
- Design the report by selecting rows for:
 - Page
 - Column
 - Row
 - Detail

Demonstration: Using the Report Wizard to build a report

In this demonstration, you will see how to use the Report Wizard to build a report.

Demonstration Steps

Prepare the environment

1. Ensure that both **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are running, and then log on to **10990C -MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Demofiles\Mod03** folder, run **Setup.cmd** as Administrator. Click **Yes** when prompted to confirm that you want to run the command file, and wait for the script to finish.

Using the Report Wizard from Report Designer

1. Start Visual Studio 2015, then on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, go to the **D:\Demofiles\Mod03\Demo** folder, click **Reports Demo.sln**, and then click **Open**.
3. In the **Security Warning for Reports Demo** dialog box, clear the **Ask me for every project in this solution** check box, and then click **OK**.
4. In Solution Explorer, right-click **Reports**, and then click **Add New Report**.
5. On the **Welcome to the Report Wizard** page, click **Next**.
6. On the **Select the Data Source** page, click **Next**.
7. On the **Design the Query** page, click **Query Builder**.
8. In the **Query Designer** dialog box, on the toolbar, click **Add Table**.
9. In the **Add Table** dialog box, click **DimGeography**, hold down the CTRL key, click **DimReseller**, click **FactResellerSales**, click **Add**, and then click **Close**.

10. In the **Query Designer** dialog box, in the upper pane, select the following columns:
 - DimGeography.EnglishCountryRegionName
 - DimGeography.StateProvinceName
 - DimGeography.City
 - DimReseller.ResellerName
 - FactResellerSales.SalesOrderNumber
 - FactResellerSales.OrderDate
 - FactResellerSales.SalesAmount
11. In the second pane from the top, add the following values to the **Alias** column:
 - EnglishCountryRegionName: **Country-Region**
 - StateProvinceName: **State**
 - ResellerName: **Reseller**
 then click **OK**.
12. On the **Design the Query** page, click **Next**.
13. On the **Select the Report Type** page, ensure that **Tabular** is selected, and then click **Next**.
14. On the **Design the Table** page, add all fields to the **Details** section then click **Next**.
15. On the **Completing the Wizard** page, in the **Report name** box, type **Reseller Sales**, and then click **Finish**.
16. Click the **Preview** tab to view the report.
17. Leave Visual Studio 2015 open for the next demonstration.

Using the Report Wizard from Report Builder

1. Click **Start** then type **Report Builder** then press Enter.
2. In Report Builder, in the **Getting Started** window, click **Table or Matrix Wizard**.
3. In the **New Table or Matrix** window, on the **Choose a dataset** page select **Create a dataset**, then click **Next**. On the **Choose a connection to a data source** page, click **Browse**.
4. In the **Select Data Source** window, in the **Name** box type **http://mia-sql/ReportServer_SQL2**, then click **Open**. Double-click **AdventureWorks Sample Reports**, then click **AdventureWorksDW**, then click **Open**.
5. On the **Choose a connection to a data source** page, click **Next**.
6. On the **Design a query** page, in the **Database view** pane, expand **Tables**, then expand **DimGeography** then select the following columns:
 - **EnglishCountryRegionName**
 - **StateProvinceName**
 - **City**
7. Expand **DimReseller**, then select the **ResellerName** column.

8. Expand **FactResellerSales** and select the following columns:
- **SalesOrderNumber**
 - **SalesAmount**
 - **OrderDate**
- then click **Next**.
9. On the **Arrange fields** page shift-click all the columns in the **Available fields** box, then drag and drop them into the **Values** box, then click **Next**. Click **Next**, then click **Finish**.
10. To preview the report, click **Run**. Close Report Builder without saving any changes when you have finished.

Sequencing Activity

Put the following steps for creating a report with the Report Wizard in the correct numerical order.

	Steps
	Select or create a data source.
	Design a query to define the dataset.
	Specify the report type.
	Add fields to the report.
	Give the report a name.
	Preview the report.

Lesson 2

Creating a report

You develop reports in both Report Designer and Report Builder. Report Designer is a professional report authoring interface for the Visual Studio development environment that you use to build and publish reports in Visual Studio. Report Builder is a standalone report authoring tool that's aimed at business users who prefer a Microsoft Office-style reporting environment.

This lesson describes how to use Report Designer and Report Builder to develop reports.

Lesson Objectives

After completing this lesson, you will be able to:

- Create a dataset in Report Builder or Report Designer.
- Design a report in Report Builder or Report Designer.
- Use tablix data regions.
- Understand the difference between tables, lists, and matrices.
- Format a report in Report Builder or Report Designer.
- Understand considerations when designing reports.

Creating a dataset

In the previous module in this course, you learned how to define data sources for your reports. After defining the data source, you must define the datasets that contain the data to be displayed. A dataset:

- Defines a query to retrieve data from a data source.
- Determines the fields that are available for use in the report.
- Can be shared across multiple reports or embedded in a single report.

- Defines a query that's used to retrieve data from a data source
- Determines the fields in the report
- Can be shared or embedded:
 - Shared—the same dataset can be used by multiple reports
 - Embedded—the dataset is specific to the report
 - Can convert an embedded dataset to a shared dataset

Depending on your choice of application, you create one or more datasets for each data source in the following ways:

- Specify details for the data source on the Design the Query page of the Report Wizard.
- Add a dataset to the report in the Report Data pane.
- Add a shared dataset to the project in Solution Explorer.
- In the Report Data pane, you can also convert an embedded dataset into a shared dataset.

Dataset guidelines

When creating datasets, consider the following guidelines:

- Include only columns that are required for display, sorting, or grouping in the report. Avoid using a `SELECT *` statement; instead, you should list only the columns that are required to build the output query.
- Include only the rows that are required for the detail level of the report. For example, if you are creating a report to show sales performance for each product, grouped by category, there is no need to include the individual order rows. Instead, use a query with a `GROUP BY` clause that aggregates the orders by product and category. Product-level aggregation in the query is then the detail level of the report.
- Sort data in the query. Although you can sort data in a data region in a report, it's generally more efficient to sort it in the query by using an `ORDER BY` clause.
- Use views instead of base tables.

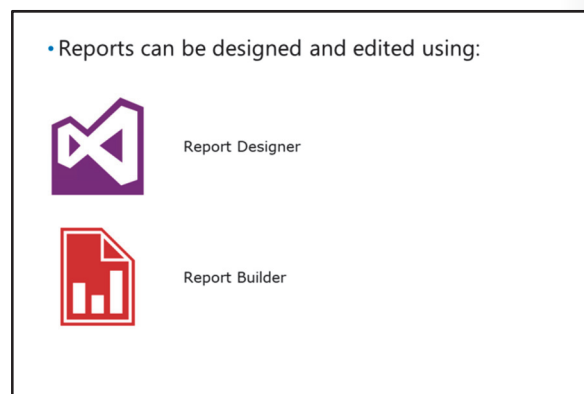
Designing the report

Key elements of the Report Designer interface include:

- **Solution Explorer.** This pane shows the projects and associated items in the Visual Studio solution that's currently open. For a Report Server project, it shows the reports, shared data sources, shared datasets, and other resources.
- **Report Design surface.** This is the graphical report development area on the design tab of the Report Designer. You use this part of the interface to design a report.
- **Report body.** This represents the body of your report, where you drag items to define the layout.
- **Properties pane.** This pane shows the properties of the selected item.
- **Report Data pane.** This pane shows the data sources, datasets, parameters, and images you have defined, in addition to built-in fields you can use in any report.
- **Toolbox.** This shows the report items you place in the report body.
- **Grouping pane.** This pane shows the data groupings defined in your report; you use it to define and manage groupings to aggregate report data.
- **Preview tab.** This tab previews the report, so you can verify how it's rendered before publishing.

Key elements of the Report Builder interface include:

- **Ribbon.** This contains controls and tools for formatting, designing, and running reports.
- **Report design surface.** This is the graphical report development area of Report Builder that's reached by clicking Design in the Views section of the ribbon. You use this part of the interface to design a report.



- **Report Data pane.** This pane shows the data sources, datasets, parameters, and images you have defined, in addition to built-in fields you can use in any report.
- **Row groups/column groups pane.** This pane shows the data groupings defined in your report; you use it to define and manage groupings to aggregate report data.


Using a tablix data region

After defining a data source and a dataset, you use a tablix data region to display data field values in a report. The term “tablix” is derived from the words “table” and “matrix”. A tablix data region provides a common user interface element that supports both different types of data layout.

When you create a report by using the Report Wizard, it automatically generates a table or matrix data region, depending on the type of report you select in the wizard. To add a tablix data region to an existing report, drag the appropriate data region item from the toolbox to the report body.

- Used for displaying data values
- A tablix data region can be:
 - A table
 - A list
 - A matrix

After adding a tablix data region to the report, you specify the data it should display by dragging fields from the Report Data pane into the appropriate space in the data region. A tablix data region can only be bound to a single dataset; the first field you add creates this binding. Attempting to drag fields from a second dataset into a data region results in an error.

 **Note:** You use lookup functions to include data from multiple related datasets in a single data region.

Differences between tablix, table, and matrix

A tablix can be one of three types of data region—a table, a list, or a matrix. Each data region displays data in a different way, as described in the following list:

- **Table.** A table data region shows data as rows in a table with a column for each field. For example, you could use a table to show sales results aggregated by country, with a row for each country.
- **Matrix.** A matrix data region shows data fields grouped in rows and columns, with aggregated values in the intersecting cells, like a PivotTable or crosstab. For example, you could use a matrix to show sales for each country for a range of years, with a row for each country and column for each year.

Table

Country	Sales
Australia	\$2,640,812.09
Canada	\$17,737,517.75
France	\$6,543,878.32
Germany	\$3,227,276.48
United Kingdom	\$6,074,124.22
United States	\$66,779,214.09

List

A Bike Store

Business Type: Value Added Reseller
City: Seattle
Phone: 206-655-0173

A Great Bicycle Company

Business Type: Specialty Bike Shop
City: Jefferson City
Phone: 602-655-0189

A Typical Bike Shop

Business Type: Value Added Reseller
City: Round Rock
Phone: 888-655-0188

Matrix

	2001	2002	2003	2004
Australia			\$847,430.96	\$1,793,381.13
Canada	\$1,513,359.46	\$4,822,999.20	\$5,651,305.43	\$5,749,853.66
France		\$857,123.18	\$2,373,804.04	\$3,312,951.10
Germany			\$1,098,866.68	\$2,128,409.79
United Kingdom		\$841,757.76	\$2,160,145.83	\$3,072,220.63
United States	\$6,552,075.85	\$17,622,549.51	\$20,071,116.48	\$22,533,472.24

- **List.** A list data region shows a repeated arrangement of data items in a free-form layout. For example, you could use a list to show details for each store in the organization as a store name field formatted as a heading—with labels and field values for business type, city, and phone number.

Formatting the report

You can format all elements of a report, in both Report Designer and Report Developer, by selecting the element and using the format options on the ribbon in Report Builder or the toolbar in Report Designer.

Selecting a row or column in a tablix formats all values in the selected row or column.

Page breaks

You can view reports online in a browser but, in many cases, they are printed or exported to another format for offline viewing and further analysis. Reporting Services provides a number of ways to control how reports are divided into pages when they are printed or rendered in a format that supports pagination. Pagination is particularly important when working with groups; for example, you can then start a new page for each new instance of a group.

Key ways in which you control pagination include:

- Defining page breaks in tablix data regions, groups, and rectangles. You define explicit behaviors for page breaks by setting the **BreakLocation**, **ResetPageNumber**, and **PageName** properties in the **PageBreak** category.
 - The **BreakLocation** property specifies where the page break should occur. For data regions and rectangles, this can be before, after—or before and after—the report element on which the page break is defined. For groups, you might also specify that the page break should occur between instances. For example, you might force a new page for each employee in a report that shows sales grouped by employee.
 - The **ResetPageNumber** property causes page numbering to be restarted on the new page that the page break generates.
 - The **PageName** property specifies a name for the new page that the page break generates. Some renderers, such as Microsoft Excel®, use page names to identify pages. You can set an explicit value for this property, or use an expression to generate the value dynamically. This might be used, for example, to specify the name of the employee in the grouped sales report that was described in the previous topic.
 - You can also set the **Disabled** property to disable a page break.
- Setting the **InitialPageName** property of the report. This property defines the page name for the first page in the report (or for all pages, if it contains no explicitly-defined page breaks).

- Formatting ensures that reports are usable when exported
- Create page break location for groups:
 - Start, End, Between, or Start and End
- Use expressions to set the **PageName** property of page breaks dynamically
 - **InitialPageName** report property sets default page name
 - Page names determine worksheet names when rendering to Excel

Pagination and Microsoft Excel

Being able to control pagination is particularly useful when you export reports to Microsoft Excel. By default, the Excel rendering extension creates a workbook with a single worksheet that contains the report data. The worksheet name, which is shown on the tabs at the bottom of the workbook, is based on the name of the report by default. However, if you specify an **InitialPageName** property value, that is used for the worksheet name instead.

In many cases, you might want to render the report data on multiple worksheets in Excel. One reason for doing this is to make it easier for users to navigate the report data; for example, by creating a new worksheet (and corresponding tab) for each employee grouping in the sales report described in the previous example. Another reason is that Excel enforces a maximum number of rows for each worksheet. A very large report might not render successfully if it contains more rows than are available in a worksheet.

Setting a page break in a report causes the Excel rendering extension to generate a new worksheet. If the page break includes a **PageName** property value, the worksheet is named accordingly. This means that you can design multiple worksheet Excel-based reports that are easy to navigate.

Considerations for designing a report

Consideration should be given to the following items before developing reports in Report Builder or Report Designer.

Data regions

A data region displays data from a report dataset and can be saved as a report part. When used effectively, report parts greatly reduce the amount of work required to develop a reporting solution by enabling reuse of work.

These considerations apply equally to Report Builder and Report Designer

- Data regions
 - Enable greater reuse of existing reports
- Graphical vs. text
 - Graphics turn data into rich information
 - Graphics come at a cost in performance
- Output
 - Where will the report be consumed? Online, printed, or in Excel

Graphical vs. text data interpretation

Graphical report elements might improve the presentation and help users to understand the data that a report is presenting to them; however, they take considerably more resources to render than textual data. Before adding graphical elements to a reporting solution, think about whether they add value. Avoid adding graphical elements that have no benefit to end users.

Report output format

When planning a reporting solution, you should consider the ways in which reports will be consumed. Typically, reports are consumed in one of the following ways:

- **Online:** users view the report in a web browser.
- **As a document in an application:** users open the report in an application, such as Microsoft Excel.
- **Printed:** users print the report and view it in hard copy.

When you publish a report to a Reporting Services server, it might be rendered in an interactive HTML-based report viewer that helps users to consume the report online. However, users can also export the report to a number of formats, and print it for offline viewing. You can also deliver reports in specific file formats through subscriptions, where users do not need to view the report online. They open the report directly in an application for viewing or printing.

Reporting Services supports this range of options for viewing reports by using an extensible rendering architecture, in which a report is provided in any format, where a renderer is available. Each renderer has its own strengths, limitations, and features—so understanding which renderers must be supported in your reporting scenario might help optimize reports to suit the pagination, image support, and interactivity available in your target formats. By default, Reporting Services includes renderers for the following formats:

- Microsoft Excel
- Microsoft Word
- Comma separated value (CSV)
- XML
- Web archive
- Image
- Adobe Acrobat (PDF)
- Atom feed



Note: When developing reports that might be rendered in multiple formats, test each render format to ensure that the pagination and formatting provides the desired output.

Demonstration: Creating a report

In this demonstration, you will see how to:

- Create a report in Report Designer.
- Create a report in Report Builder.

Demonstration Steps

Create a report in Report Designer

1. In Visual Studio 2015, in Solution Explorer, right-click **Reports**, point to **Add**, and then click **New Item**.
2. In the **Add New Item – Reports Demo** dialog box, click **Report**, in the **Name** box, type **Reseller Sales 2** and then click **Add**.
3. In Solution Explorer, right-click **Shared Datasets**, and then click **Add New Dataset**.
4. In the **Shared Dataset Properties** dialog box, in the **Name** box, type **MyDataset**, in the **Data source** list, click **AdventureWorksDW**, in the **Query** box, type the following code, and then click **OK**:

```
SELECT      DimGeography.EnglishCountryRegionName, DimGeography.StateProvinceName,
DimGeography.City, DimReseller.ResellerName, FactResellerSales.SalesOrderNumber,
FactResellerSales.OrderDate, FactResellerSales.SalesAmount
FROM
DimGeography INNER JOIN
DimReseller ON DimGeography.GeographyKey = DimReseller.GeographyKey INNER JOIN
FactResellerSales ON DimReseller.ResellerKey = FactResellerSales.ResellerKey
```

5. From the Toolbox, drag a Table item to the report design surface.

6. In the **Dataset Properties** dialog box, in the **Name** box, type **MyDataset**, and then in the list of datasets, click **MyDataset**.
7. On the **Fields** page, change the Field Name values as shown in the following table, and then click **OK**:

Field Name	Field Source
Country_Region	EnglishCountryRegionName
State	StateProvinceName
Reseller	ResellerName

8. In the **Report Data** pane, expand **Datasets**, expand **MyDataset**, and then drag each data item in turn to the table header. If the available cells are full, position the item on the right-side border of the table and a new cell will automatically appear.
9. From the **Toolbox**, drag a text box to the report design surface, aligning it with the top of the table.
10. In the text box, type the text **Reseller Sales**.
11. Click the **Preview** tab to view the report.
12. Leave Visual Studio 2015 open for the next demonstration.

Create a report in Report Builder

1. Start Internet Explorer and browse to **mia-sql/Reports_SQL2**.
2. On the **Home** page, in the toolbar, click **New**, then click **Paginated Report**.
3. In the **Internet Explorer** dialog box, click **Allow**.
4. If the **Connect to Report Server** dialogue appears, click **Yes**.
5. In the **New Report or Dataset** dialog box, click **Blank Report**.
6. In Report Data, right-click **Datasets**, and then click **Add Dataset**.
7. In the **Dataset Properties** dialog box, in the **Name** box, type **MyDataset**, select **Use a dataset embedded in my report**, and then click **New**.
8. In the **Data Source Properties** dialog box, in the **Name** box, type **AdventureWorksDW**, and then click **Use a connection embedded in my report**.
9. In the **Connection string** text box, type the following:

```
Data Source=MIA-sql;Initial Catalog=AdventureWorksDW
```

10. In the Credentials pane, click **Use current Windows user. Kerberos delegation might be required**. Then click **OK**.
11. In the **Dataset Properties** dialog box, in the **Query** box, type the following:

```
SELECT      DimGeography.EnglishCountryRegionName, DimGeography.StateProvinceName,
DimGeography.City, DimReseller.ResellerName, FactResellerSales.SalesOrderNumber,
FactResellerSales.OrderDate, FactResellerSales.SalesAmount,
DimDate.CalendarYear AS SalesYear
FROM
DimGeography INNER JOIN
DimReseller ON DimGeography.GeographyKey = DimReseller.GeographyKey INNER JOIN
FactResellerSales ON DimReseller.ResellerKey = FactResellerSales.ResellerKey INNER
JOIN
DimDate ON [DateKey] = FactResellerSales.OrderDateKey
```

12. Click **Refresh Fields**.
13. On the **Fields** page, change the **Field Name** values to match the following table, and then click **OK**.

Field Name	Field Source
Country_Region	EnglishCountryRegionName
State	StateProvinceName
Reseller	ResellerName

14. On the **Insert** menu, click **Table**, and then click **Insert Table**.
15. Click anywhere on the report design surface to insert a new table.
16. In **Report Data**, expand **Datasets**, expand **MyDataset**, and then drag each data item in turn to the table header. If the available cells are full, position the item on the right-side border of the table and a new cell will automatically appear.
17. Click the **Click to add title** text box.
18. In the text box, type **Reseller Sales**.
19. On the **Home** menu, click **Run**.
20. Leave Report Builder open for the next demonstration.

Verify the correctness of the statement by placing a mark in the column to the right.

Statement	Answer
True or false? For every page break added into a report, the Excel rendering extension will generate a new worksheet.	

Lesson 3

Publishing a report

When you have finished designing your report, you commonly want to share it with others. Publishing a report to Reporting Services is a simple way to share and organize a library of reports.

Lesson Objectives

At the end of this lesson, you should be able to:

- Preview reports.
- Publish a report to a report server.
- View reports.

Previewing reports

It's useful to preview reports before you publish them. Previewing reports is straightforward in both Report Builder and Report Designer. Follow these steps and specify values for any required parameters:

Report Builder

To preview a report, click **Run**. If you preview the report again it will use a cached copy of the data to improve performance. To update the data, click **Refresh**.

Report Designer

To preview a report in Report Designer, click the **Preview** tab.

Report Builder

- Click **Run**

Report Designer

- Click the **Preview** tab

Publishing a report to a report server

After creating your reports, you publish them to a report server for viewing by users.

Publishing reports from Report Designer

Professional report developers use Report Designer in Visual Studio to publish reports at the project level or the individual report level. In Report Designer, the term for publishing a report is "deploying" a report. The terms "publish" and "deploy" are interchangeable in this respect.

- Publish reports from Report Designer either as single reports or complete projects
 - Ensure that project properties are set appropriately before publishing
- Publish reports from Report Builder using the File menu

Before publishing reports, you must be aware of how they are affected by the project properties. Before you publish reports that you create in Visual Studio, you should ensure that the following properties are set appropriately for the report server where you intend to publish them:

Property	Description
OverwriteDataSources	Specifies whether to replace existing data sources when publishing shared data sources. If this property is set to False , an attempt to publish a data source with the same name as an existing one will fail.
TargetDataSetFolder	Folder where shared datasets are published. When publishing to a report server in native mode, this property should be set to the folder name. When publishing to a report server in SharePoint integrated mode, this property should be set to the full URL for the document library in the SharePoint site.
TargetDataSourceFolder	Folder where shared data sources are published. When publishing to a report server in native mode, this property should be set to the folder name. When publishing to a report server in SharePoint integrated mode, this property should be set to the full URL for the document library in the SharePoint site.
TargetReportFolder	Folder where report parts are published. When publishing to a report server in native mode, this property should be set to the folder name. When publishing to a report server in SharePoint integrated mode, this property should be set to the full URL for the document library in the SharePoint site.
TargetReportPartFolder	Folder where report parts are published. When publishing to a report server in native mode, this property should be set to the folder name. When publishing to a report server in SharePoint integrated mode, this property should be set to the full URL for the document library in the SharePoint site.
TargetServerURL	Service endpoint for the report server. When publishing to a report server in native mode, this property should be set to the URL for the report server web service. When publishing to a report server in SharePoint integrated mode, this property should be set to the URL for the SharePoint site where you want to publish the reports.
TargetServerVersion	Compatibility version for reports. The RDL format used by Reporting Services in SQL Server 2008 R2 and later is different from the format used in earlier versions. If you are publishing to an older report server, you must set the TargetServerVersion appropriately. You publish reports in either version to report servers that run SQL Server 2008 R2 or later, but some features are not supported in the older RDL format.

After you have set the project properties, you publish reports, shared data sources, and shared datasets from the project. To publish all items in the project, in Solution Explorer, right-click the project, and then click **Deploy**. To publish an individual item in the project, in Solution Explorer, right-click the item, and then click **Deploy**.

When you deploy a project or an individual item, the deployment status is shown in the Output pane.

Publishing reports from Report Builder

When publishing a report from Report Builder, you have less control over its configuration. To publish a report, you save the report to the report server. If you want to save a report from Report Builder using a different name, use the **Save As** functionality.

For more information about setting deployment properties for Report Designer, see the topic *Set Deployment Properties (Reporting Services)* in Microsoft Docs:

 **Set Deployment Properties (Reporting Services)**

<https://aka.ms/Guorqm>

Viewing reports

After a report has been deployed, users view it by performing the following steps:

1. Open a web browser and go to the report location. If the report has been published to a report server running in native mode, the user must go to the Report Manager site and, if necessary, enter appropriate authentication credentials. If the report has been published to a report server in SharePoint integrated mode, the user must go to the document library in the SharePoint site where the reports have been published.

1. Browse to the report location
 - SharePoint document library
 - Report Manager folder
 2. Click the report
 3. Enter any required parameters
 4. Use the Report Viewer toolbar to go to the report
 5. Export to a file or feed if desired
2. Click the report. If the report has no parameters, or all parameters have default values, it will be rendered in the browser.
3. Enter parameter values to filter the report as required, and click **View Report** to render it with the specified parameters.
4. Use the Report Viewer toolbar to navigate the report by scrolling through pages or searching for a specified text value.
5. Export the report to a file or click the feed icon to download the report as an atom feed.

Demonstration: Publishing a report

In this demonstration, you will see how to:

- Publish a report from Report Designer.
- Publish a report from Report Builder.

Demonstration Steps

Publishing a report from Report Designer

1. In Visual Studio 2015, in Solution Explorer, right-click **Reports Demo**, and then click **Properties**.
2. In the **Reports Demo Property Pages** dialog box, ensure the **TargetServerURL** property is set as **http://mia-sql/ReportServer_SQL2** and the **TargetReportFolder** property is set as **Designer**, and then click **OK**.
3. In Solution Explorer, right-click **Reports Demo**, and click **Deploy**. Wait for the deployment to complete.
4. Start Internet Explorer and go to **mia-sql/Reports_SQL2**.
5. Click **Designer**, and then click **Reseller Sales**. Check that the report has been correctly published.
6. Close Internet Explorer, then close Visual Studio 2015.

Publishing a report from Report Builder

1. In Report Builder, on the **File** menu, click **Save As**.
2. In the **Look in** drop-down list, ensure that **http://mia-sql/ReportServer_SQL2** is selected.
3. Double-click **Builder** and then in the **Name** box, type **Reseller Sales**, and then click **Save**. If the **Save As Report** window appears, click **Yes**.
4. Start Internet Explorer and go to the address **mia-sql/Reports_SQL2**.
5. Click **Builder**, and then click **Reseller Sales**. Check that the report has been correctly published.
6. Close Internet Explorer, then close Report Builder

Check Your Knowledge

Question	
In Report Designer, which project property determines the folder where data sources will be deployed?	
Select the correct answer.	
<input type="checkbox"/>	OverwriteDataSources
<input type="checkbox"/>	TargetDataSourceFolder
<input type="checkbox"/>	TargetReportFolder
<input type="checkbox"/>	TargetServerURL
<input type="checkbox"/>	TargetDataSetFolder

Check Your Knowledge

Question	
In Report Builder, how do you publish a report to a Reporting Services instance?	
Select the correct answer.	
<input type="radio"/>	By deploying it from Visual Studio.
<input type="radio"/>	By copying and pasting the file into the file system of the report server.
<input type="radio"/>	By using the "Save" or "Save As" commands on the File menu.
<input type="radio"/>	By saving the report to the local file system.

Lab: Creating reports

Scenario

The sales manager at Adventure Works Cycles has asked you to create two reports:

- A report of all products.
- A report of all sales made via the internet channel.

You decide to create one report using the Report Wizard, and another report from scratch. When you have created these reports, you will publish the internet channel report to the Reporting Services server.

Objectives

At the end of this lab, you should be able to:

- Use the Report Wizard to generate tabular reports.
- Create and publish reports from scratch.

Estimated Time: 90 minutes

Virtual machine: **10990C-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa55w.rd**

Exercise 1: Use the Report Wizard—Report Designer

Scenario

You've been asked to generate a product listing from the **AdventureWorksDW** database. The report should include the following columns:

- **DimProduct.ProductAlternateKey**
- **DimProduct.EnglishProductName**
- **DimProduct.StandardCost**
- **DimProduct.ListPrice**
- **DimProduct.DealerPrice**
- **DimProductCategory.EnglishProductCategoryName**
- **DimProductSubcategory.EnglishProductSubcategoryName**

You decide to use the Report Designer Report Wizard to create the report.

The main tasks for this exercise are as follows:

1. Prepare the environment
2. Use Report Wizard

► Task 1: Prepare the environment

1. Ensure the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. Run **Setup.cmd** in the **D:\Labfiles\Lab03\Starter** folder as Administrator.

► Task 2: Use Report Wizard

1. Start Visual Studio 2015 and open the project named **Reports.sln** in the **D:\Labfiles\Lab03\Starter** folder.
2. Create a new report called **Product Listing** using the Report Wizard. Refer to the exercise scenario for the list of columns to include in the report.
3. Preview the report in Visual Studio 2015. Leave Visual Studio 2015 open for the next exercise.

Results: At the end of this exercise, you should have created a report using the Report Wizard.

Exercise 2: Use the Report Wizard—Report Builder

Scenario

You have been asked to generate a product listing from the **AdventureWorksDW** database. The report should include the following columns:

- **DimProduct.ProductAlternateKey**
- **DimProduct.EnglishProductName**
- **DimProduct.StandardCost**
- **DimProduct.ListPrice**
- **DimProduct.DealerPrice**
- **DimProductCategory.EnglishProductCategoryName**
- **DimProductSubcategory.EnglishProductSubcategoryName**

You decide to use the Report Builder Report Wizard to create the report.

The main tasks for this exercise are as follows:

1. Use Report Wizard

► Task 1: Use Report Wizard

1. Start Report Builder, then create a new Product Listing report using the Table or Matrix Wizard. Refer to the exercise scenario for the list of columns to include in the report.
2. Preview the report in Report Builder.
3. Close Report Builder when you have finished without saving any changes.

Results: At the end of this exercise, you should have created a report using the Report Wizard.

Exercise 3: Creating and publishing a report—Report Designer

Scenario

You have been provided with a query that returns a listing of all sales made via the internet channel. You will use this query to build a report then publish it to Reporting Services.

The main tasks for this exercise are as follows:

1. Create the report
2. Publish the report

► Task 1: Create the report

1. Using Visual Studio 2015 create a new report called **Internet Sales Detail**.
2. Using the following query against the **AdventureWorksDW** data source, create a shared dataset called **InternetSales** (the text of this query can be found in **D:\Labfiles\Lab03\Starter\Internet Sales.sql**):

```
SELECT fis.SalesOrderNumber, fis.SalesAmount, fis.TaxAmt, fis.OrderQuantity,
dp.EnglishProductName,
dpr.EnglishPromotionName, dc.FirstName, dc.LastName
FROM FactInternetSales AS fis
INNER JOIN DimCustomer AS dc ON fis.CustomerKey = dc.CustomerKey
INNER JOIN DimProduct AS dp ON fis.ProductKey = dp.ProductKey
INNER JOIN DimPromotion AS dpr ON fis.PromotionKey = dpr.PromotionKey
```

3. Add a Table item to the new report, linking it to the shared **InternetSales** dataset. In the Table dataset, rename the following columns:

Field Name	Field Source
Tax	TaxAmt
Product	EnglishProductName
Promotion	EnglishPromotionName

4. Add all the dataset columns to the Table item in the report.
5. Add a text box to act as a report title. Make the report title **Internet Sales Detail**.
6. Preview the report, then return to design view to change the width of any columns that appear too wide or too narrow.

► Task 2: Publish the report

1. Update the properties of the **Reports** project settings:
 - Make sure that datasets, data sources, and reports are deployed to the **Designer** folder on the Reporting Server.
 - Use the following server URL: **http://mia-sql/ReportServer_SQL2**
2. Deploy the **Reports** project, then use Internet Explorer to connect to the Reporting Services web portal to confirm that the deployment was successful.
3. Close Internet Explorer and Visual Studio 2015 when you have finished.

Results: At the end of this exercise, you should be able to:

Create a report from scratch by using Report Designer.

Customize elements of the report design.

Publish the report to Reporting Services.

Exercise 4: Creating and publishing a report—Report Builder

Scenario

You have been given a query that returns a listing of all sales made via the internet channel. You will use this query to build a report then publish it to Reporting Services.

The main tasks for this exercise are as follows:

1. Create the report
2. Publish the report

► Task 1: Create the report

1. Using Report Builder create a new report titled **Internet Sales Detail**.
2. Create an embedded dataset called **InternetSales** that connects to an embedded data source for the **AdventureWorksDW** with the following connection string:

```
Data Source=MIA-SQL;Initial Catalog=AdventureWorksDW
```

the dataset should use the following query (the text of this query can be found in **D:\Labfiles\Lab03\Starter\Internet Sales.sql**):

```
SELECT fis.SalesOrderNumber, fis.SalesAmount, fis.TaxAmt, fis.OrderQuantity,
dp.EnglishProductName,
dpr.EnglishPromotionName, dc.FirstName, dc.LastName
FROM FactInternetSales AS fis
INNER JOIN DimCustomer AS dc ON fis.CustomerKey = dc.CustomerKey
INNER JOIN DimProduct AS dp ON fis.ProductKey = dp.ProductKey
INNER JOIN DimPromotion AS dpr ON fis.PromotionKey = dpr.PromotionKey
```

3. Add a Table item to the new report, linking it to the embedded **InternetSales** dataset. In the Table dataset, rename the following columns:

Field Name	Field Source
Tax	TaxAmt
Product	EnglishProductName
Promotion	EnglishPromotionName

4. Add all the dataset columns to the Table item in the report.
5. Preview the report, then return to design view to change the width of any columns that appear too wide or too narrow.

► Task 2: Publish the report

1. Save the report as **Internet Sales Detail** in the **Builder** folder on the Reporting Server.
2. Use Internet Explorer to connect to the Reporting Services web portal to confirm that the deployment was successful.
3. Close Internet Explorer and Report Builder when you have finished.

Results: At the end of this exercise, you should be able to:

Create a report from scratch by using Report Builder.

Customize elements of the report design.

Publish the report to Reporting Services.

Question: In the exercise where you created and published a report, you used the dataset properties to change the labels of some of the columns in the report. How else might you have achieved this? What factors should you consider when selecting a method for renaming columns?

Module Review and Takeaways

In this module, you have learned about different techniques for creating simple tabular reports—using the Report Wizard, or starting from a blank report. You have also learned about how to publish reports to a Reporting Services server.

Review Question(s)

Question: For the types of reporting used in your organization, would you use the Report Wizard or a blank report when starting to develop a new report?

MCT USE ONLY. STUDENT USE PROHIBITED

Module 4

Working with Reporting Services Data

Contents:

Module Overview	4-1
Lesson 1: Data filters	4-2
Lesson 2: Report parameters	4-9
Lesson 3: Implementing report filters and parameters	4-14
Lab: Create a parameterized report	4-23
Module Review and Takeaways	4-29

Module Overview

It's common for business requirements to change regarding the information they need and how they want data to be presented. On viewing a detailed report, senior management might ask for higher level, summarized, or filtered versions of the same report.

Report Builder and Report Designer support these scenarios via filtering, sorting, drilldowns, grouping and the parameterizing of reports.

In this module, you will see how to use filters and parameters to make reports more dynamic and useful to business users.

Objectives

After completing this module, you will be able to:

- Filter data in reports.
- Use parameters in reports.

Lesson 1

Data filters

One way of making data more meaningful is to focus on the specific aspects. SQL Server Reporting Services (SSRS) helps you to do this by filtering the data within the report.

Lesson Objectives

After completing this lesson, you will be able to:

- Explain what data filters are.

What are data filters?

Filters exclude certain rows from a dataset. You might want to show data that relates to a specific region, or country, whereas the dataset contains data for the whole country or region.

There are two ways you filter data for a report:

1. Filter data at source.
2. Filter data within the report.

The preferred method of filtering data is at source—before it gets to the report. This is an efficient process because only the required data is sent over the network. In addition, database systems such as SQL Server are designed to work with large volumes of data, and select the required information quickly.

There are occasions, however, when filtering must be done in the report. This is less efficient because unnecessary data must be processed by the report, but sometimes it's unavoidable. For example:

- When many people are using the same dataset, and you can't change it.
- You want to use one dataset for different reports, and each requires a subset of the data.
- You have no control over how the data is presented from the source system.

Filtering data at source

If the data is coming from SQL Server, you use a WHERE clause in the query to define the data that's required. When data is filtered in this way, the report uses the dataset without further processing.

Report filters

A filter expression can be applied to a report in different places. You should consider where and how filters will be used. For example, a filter might be added:

- **On a shared dataset.** Set a filter on a shared dataset when you want multiple data regions, or charts bound to the same dataset, to be filtered in the same way.
- **On a data region.** Set a filter on the data region when you want one or more data regions that are bound to the same dataset to be filtered differently, and therefore provide a different view of the same dataset.

- Dataset filters
 - WHERE clause on the SQL query
- Report filters
 - Shared datasets
 - Data regions
 - Row or column groups
 - Detail groups
 - Series or categories for charts
- Use the Report Data pane to view data sources, datasets, and parameters

- **On a row or column group in a tablix.** Set a filter on a group when you want to include or exclude certain values for a group expression to control which group values appear in the table, matrix, or list.
- **On a detail group in a tablix.** Set a filter on the detail group when you have multiple detail groups for a data region, and want each detail group to display a different set of data from the same dataset.
- **On the series or category groups in a chart.** Set a filter on a series or category group when you want to include or exclude certain values for a group expression to control which values appear in the chart.

All of the above might use constants to filter the datasets. However, you might want to use a value that's entered by a user. Combining user selected values—called parameters—with filters enables a report to provide different views of the same data. Parameters are discussed in detail later in this module.

Report Data pane in Report Designer

The Report Data pane displays the parameters, data sources, datasets, images, and built-in fields that are either being used in your report, or are available to your report. Ensure the Report Data pane is visible when you work with filters or parameters.



Note: If the Report Data pane is not displayed, click **View**, and then click **Report Data**—or click **Ctrl-Alt-D**. If you can't see the Report Data pane on the View menu, or Ctrl-Alt-D doesn't work, click the **Design** tab and try again.

For more information about working with the Report Data pane, see Microsoft Docs:



Report Data Pane

<https://aka.ms/Yxs3mm>

Report Data view in Report Builder

The Report Data pane in Report Builder displays the parameters, data sources, datasets, images, and built-in fields that are either being used in the report, or are available. To display the Report Data pane in Report Builder, click the **View** menu and, in the **Show/Hide** group, select **Report Data**.

Where to set filters

You add filters to a dataset, a data region, or a data region group. In each case, the filter will restrict the data to the rows that match certain criteria.

Filters are applied in order:

- The dataset.
- The data region.
- Groups from the top downwards.

Deciding where to add a filter will depend on the design of your report. If you filter at the dataset level, then all elements of the report will work from the same subset of data. Filtering at, for example, the data region level will allow other elements in the report to show other information.

- You set data filters at:
 - Dataset level—providing consistency to all elements within a report
 - Data region level—allowing different sets of the data to be displayed in different data regions
 - Chart level—enabling a subset of data to be shown visually

Filters at the dataset level

Filtering at the dataset level ensures that all the data in a report is consistent. By filtering the dataset, all tables, lists, charts, and so on, work from the same data. This provides consistency, and makes reports easier to understand.

Filters at the data region level

Filters at the data region level help you to show portions of the data in different data regions. For example, you might want to divide the report into the top 20 performing sales executives, and all other sales executives. To achieve this, you create filters at the data region level, by using the same dataset for each. You also filter at the group and detail level.

Filter at the chart level

You use filters at the chart level to produce a chart that shows a subset of the data. This might be the top performing sales executives, or data relevant to a particular region. When you filter at the chart level, ensure that the chart title reflects how the data has been filtered.

For more information about where to set filters, see Microsoft Docs:



Add Dataset Filters, Data Region Filters, and Group Filters

<https://aka.ms/lptwiw>

Add or delete filters

You add filters to a dataset, a data region, or a data region group.

Add or delete a dataset filter

Filters are set in the same way in both Report Designer, and Report Builder. To set a filter at the dataset level:

1. Ensure the Report Data pane is displayed.
2. Expand the **Datasets** group.
3. Right-click the appropriate dataset, and then click **Dataset Properties**. The **Dataset Properties** dialog box is displayed.
4. In the left pane, click **Filters**.
5. Under Change filters, click **Add**.
6. In Expression, select a **field** to match against. Ensure that the data type in the drop-down list on the right is displayed correctly.
7. In Operator, select the appropriate **operator**.
8. In Value, enter a **value** to match against.
9. Alternatively, enter an expression by clicking **fx**.
10. To delete a filter, select the filter, and then click **Delete**.
11. Click **OK**.

- Adding filters is the same in Report Builder and Report Designer
- Whether you add a filter to a dataset, a tablix region, a chart or gauge, you should:
 - Display the properties
 - Select **Filter**
 - Use the drop-down lists to add a filter, or click **fx**
 - Click **OK**

Add or delete a filter at the data region

Filters at the data region level are set in the same way in both Report Designer, and Report Builder. You create the filter in the same way as for a dataset filter—the only difference is how the **Properties** dialog box is opened.

To set a filter at the data region level:

1. Click somewhere within the data region. Gray bars appear at the top and the side of the Tablix region.
2. Right-click the gray bars, and then click **Tablix Properties**. The **Tablix Properties** dialog box is displayed.
3. From the left pane, select **Filters**.
4. Under Change filters, click **Add**.
5. Add one or more filters using the drop-down lists, or click **fx**.
6. To delete a filter, select the filter, and then click **Delete**.
7. Click **OK**.

Add or delete a filter to a chart

You add or delete a filter to a chart in the same way in Report Designer and Report Builder:

1. In design view, right-click the chart, and then click **Chart Properties**. The **Chart Properties** dialog box is displayed.
2. In the left pane, click **Filters**.
3. Under Change filters, click **Add**.
4. Add one or more filters by using the drop-down lists, or click **fx**.
5. To delete a filter, select the filter, and then click **Delete**.
6. Click **OK**.

You add filters to gauges in the same way.

For more information about adding filters, see Microsoft Docs:



Add Dataset Filters, Data Region Filters, and Group Filters

<https://aka.ms/lptwiw>

Demonstration: Filters in Report Designer

In this demonstration, you will see how to:

- Filter data at the dataset level.
- Filter data at the data region level.

This demonstration uses Report Designer.

Demonstration Steps

Filter the dataset

1. On the taskbar, click **Visual Studio 2015** to open it.
2. On the **File** menu, point to **Open**, click **Project/Solution**, and then navigate to **D:\Demofiles\Mod04\Filters**.
3. Click **Filters.sln** and then **Open**.
4. In Solution Explorer, in the **Reports** folder, double-click **Filters.rdl**, the report is displayed in Design mode.
5. Click the **Preview** tab, and then click the **right** arrow to go to the last page. Note the number of pages in the report.
6. Click the **Design** tab, and in the **Report Data** pane, expand **Datasets**.
7. Right-click **Dataset1**, and then click **Dataset Properties**.
8. In the **Dataset Properties** dialog box, in the left pane, click **Filters**.
9. In the **Change filters** screen, under **Change filters**, click **Add**.
10. In **Expression**, select **[Year]**. Note that the data type changes to Integer.
11. In **Operator**, select **Between**.
12. In **Value**, type **2012** in the first box, and **2013** in the second box.
13. Click **OK**.
14. Click **Preview**. Click the right arrow to go to the last page. Note the number of pages in the report.
15. Click **Design**.

Filter the data region

1. Click somewhere within the table, and then right-click the gray bar above the table.
2. From the menu, click **Tablix Properties**.
3. In the left pane, click **Filters**.
4. Under **Change filters**, click **Add**.
5. In **Expression**, select **[SalesAmount]**. Note that the data type has changed to Float.
6. In **Operator**, select **Top %**.
7. In **Value**, type **10**.
8. Click **OK**.
9. Click **Preview**. Click the right arrow to go to the last page. Note the number of pages in the report.

Demonstration: Filters in Report Builder

In this demonstration, you will see how to:

- Filter data at the dataset level.
- Filter data at the data region level.

This demonstration uses Report Builder.

Demonstration Steps

Filter the dataset

1. Click **Start**, and then type **Report Builder**. Click to open it.
2. Click **Open** and then navigate to **D:\Demofiles\Mod04\Filters**.
3. Click **Filters.rdl** and then click **Open**.
4. Click **Run**, and then click the **Last** icon to go to the last page. Note the number of pages in the report.
5. Click **Design**, and in the **Report Data** pane, expand **Datasets**.
6. Right-click **Dataset1**, and then click **Dataset Properties**.
7. In the **Dataset Properties** dialog box, in the left pane, click **Filters**.
8. In the **Change filters**, under **Change filters**, click **Add**.
9. In **Expression**, select **[Year]**. Note that the data type changes to Integer.
10. In **Operator**, select **Between**.
11. In **Value**, type **2012** in the first box, and **2013** in the second box.
12. Click **OK**.
13. Click **Run**. Click the **Last** arrow to go to the last page. Note the number of pages in the report.
14. Click **Design**.

Filter the data region

1. Click somewhere within the table, and then right-click the gray bar above the table.
2. From the menu, click **Tablix Properties**.
3. In the left pane, click **Filters**.
4. Under **Change filters**, click **Add**.
5. In **Expression**, select **[SalesAmount]**. Note that the data type has changed to Float.
6. In **Operator**, select **Top %**.
7. In **Value**, type **10**.
8. Click **OK**.
9. Click **Run**. Click the last arrow to go to the last page. Note the number of pages in the report.

Check Your Knowledge

Question	
You want to filter report data. Where is the most efficient place to specify the filter?	
Select the correct answer.	
<input type="checkbox"/>	On a row in a tablix.
<input type="checkbox"/>	On a detail group in a tablix.
<input type="checkbox"/>	On a category group in a chart.
<input type="checkbox"/>	On a data region.
<input type="checkbox"/>	On a dataset source, using the WHERE clause.

Lesson 2

Report parameters

This lesson discusses report parameters, including what they are and how to add them to a report.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe what a report parameter is.
- Add a report parameter to a report.

What are report parameters?

A report parameter is used to define the data that appears in the report. You use report parameters to interact with a report, by typing or selecting a value from a list. The parameter is then used to filter information in the report.



Best Practice: Rather than allowing a user to enter any value, it's best practice to provide a list of values to select from. This might be a list you create, or include values from a dataset. It's best practice to provide a default value that allows the report to execute correctly without any input from the user.

- Dataset parameters
 - Filter a source query
- Report parameters
 - Control data in a report
 - Connect related reports together
 - Vary the presentation in a report

Report parameters might be associated with dataset parameters—although they don't have to be. If a report parameter is set to be visible, its main purpose is to help report users to interact with the report. You use parameters in conjunction with filters to provide different views of the same data.

You also use report parameters in the following ways:

- Use two parameters to control the range of data used in a report.
- Use a parameter to relate main reports to drillthrough reports.
- Select a value from one parameter that determines the values that are available in another parameter.
- Run the report with a default parameter so that a user doesn't have to select a value.
- Pass a URL string to set a default value for a report parameter.

Parameters control the data contained in a report—they connect related reports together—and enable a report user to vary the presentation of a report.

Adding a parameter

You add a parameter to a report in the same way, in both Report Builder and Report Designer.

To add a parameter to a report:

1. In the Report Data pane, right-click **Parameters**. The **Report Parameter Properties** dialog box is displayed.
2. In Name, type the **name** of the parameter.
3. In Prompt, type the **label text** that will appear next to the parameter box.
4. In Data, select the correct **data type** for the parameter value. The data type must match the data type of the parameter column in the dataset.
5. In Allow blank value, select whether or not **blank** is allowed.
6. In Allow null value, select whether or not **null** is allowed.
7. In Allow multiple values, select whether or not **more than one value** is allowed.
8. In Select Parameter Visibility, select whether the parameter should be **visible**, **hidden**, or **internal**. Internal prevents the parameter from being modified after it's published. If you set the parameter to internal, ensure that it has a default value or accepts null.
9. Click **OK**.

- Add, delete, or edit parameters in the Report Data pane
- The parameter must have the same data type as the column in the dataset
- You use parameter properties to enter:
 - Name
 - Label text
 - Allowable values
 - Default
 - If more than one value can be entered
 - If blank or null is allowed
 - Visibility

To delete a report parameter:

1. In the Report Data pane, expand the **Parameters** node.
2. Right-click the parameter you want to delete, and click **Delete**.

To amend the properties of a parameter, you right-click the parameter and select **Report Parameter Properties**.

For more information about adding parameters to a report, see Microsoft Docs:

 **Add, Change, or Delete a Report Parameter (Report Builder and SSRS)**

<https://aka.ms/Bmluby>

Comparing filters and parameters

Filters and parameters work in different ways. Although they are both used to define the data that appears in a report, they can't be interchanged.

Filters

A filter is applied to the data, and is designed to limit the data in some way. As you have seen, filters can be applied in many places within a report: at the dataset level, or data region level, group or detail level, and so on. Filters are applied when the report is run, to define the data that's passed to the report or data region.

- Comparing filters and parameters
- Filters—limit the available data
 - Applied at the dataset, data region, or group level
- Parameters—provide input to something
 - Allow users to enter data to change how a report is viewed
 - Used in different places

Parameters

A parameter is a variable that's used as an input to something. In a report, a parameter is often used to allow a user to change an item that's being displayed. When a parameter is defined, it's used in many different places.

Although filters and parameters have some similarity—they can both be used to restrict the data that is displayed—in practice, they are used in very different ways.

Demonstration: Adding parameters

In this demonstration, you will see how to add a parameter to a report.

Demonstration Steps

Adding a parameter in Report Designer

1. From the taskbar, click **Visual Studio 2015** to open it.
2. On the **File** menu, point to **Open** and then click **Project/Solution**.
3. Navigate to **D:\Demofiles\Mod04**. Double-click the **Parameters** folder, Click **Parameters.sln** and then click **Open**.
4. The **AddParameter.rdl** report is opened in design mode.
5. Click **Preview** to view the report.
6. Click **Design**.
7. In the **Report Data** pane, right-click **Parameters** and then click **Add Parameter**.
8. In **Name**, type **ProductCat**.
9. In **Prompt**, type **Product Category**.
10. In **Data type**, leave as **Text**.
11. Click **Allow multiple values**.
12. Ensure **Visible** is selected.
13. In the left pane, select **Available Values**.

14. Click **Specify Values**.
15. Click **Add**.
16. In **Label** type **Bikes** and in **Value** type **Bikes**.
17. Click **Add**.
18. In **Label** type **Clothing** and in **Value** type **Clothing**.
19. Click **OK**.
20. Note that a parameter has appeared in the bar at the top of the report.
21. In the **Report Data** pane, expand **Datasets**, then right-click **Dataset1** and click **Dataset Properties**.
22. In Query, add the following line before the Order By statement:

```
WHERE C.EnglishProductCategoryName = @ProductCat
```

23. Click **OK**.
24. Click **Preview** and demonstrate the list of parameters.
25. Click **Design**.
26. **Close** the report without saving.

Adding a parameter in Report Builder

1. Click **Start** then type **Report Builder** then press Enter.
2. Click **Open** and navigate to **D:\Demofiles\Mod04**. Double click the **Parameters** folder, and double-click **Parameters.rdl**. The report is opened in design mode.
3. In the **Report Data** pane, right-click **Parameters** and then click **Add Parameter**.
4. In **Name**, type **ProductCat**.
5. In **Prompt**, type **Product Category**.
6. In **Data type**, leave as **Text**.
7. Click **Allow multiple values** to select it.
8. Ensure **Visible** is selected.
9. In the left pane, select **Available Values**.
10. Click **Specify Values**.
11. Click **Add**.
12. In **Label** type **Bikes** and in **Value** type **Bikes**.
13. Click **Add**.
14. In **Label** type **Clothing** and in **Value** type **Clothing**.
15. Click **OK**.
16. Note that a parameter has appeared in the bar at the top of the report.
17. In the Report Data pane, right-click **Dataset1** and click **Dataset Properties**.
18. In Query, add the following line before the Order By statement:

```
WHERE C.EnglishProductCategoryName = @ProductCat
```

19. Click **OK**.
20. Click **Run** and demonstrate the list of parameters.
21. Click **Design**.
22. Close the report without saving.

Lesson 3

Implementing report filters and parameters

Now you've considered report parameters, this lesson looks at how you use them within reports.

Lesson Objectives

After completing this lesson, you will be able to:

- Add and configure a report parameter.

Uses for report parameters

Parameters are used in many different ways within reports. The most common way of using a report parameter is to help a user to change the data that's presented in the report.

Connect reports together

Drillthrough reports use parameters to connect reports together. The parameter allows data to be passed from one report to another so that the correct information is displayed.

- Use report parameters to:
 - Allow users to define what information they see
 - Pass data to a drillthrough report
 - Allow users to control how much detail they see

Vary report presentation

You use a Boolean parameter to expand or collapse nested row groups, allowing the user to control how much information they view. You use the parameter in combination with the visibility property, meaning the tablix is shown when the parameter is true, and hidden when the parameter is false.

In the following example, a report parameter called **ShowSalesDetails** has been created:

Visibility can be set at different levels

Set the visibility expression

```
=Not (Parameters!ShowSalesDetails.Value)
```

For more information about using parameters to show or hide detail, see Technet:



Adding a Boolean Parameter to Control the Initial Drilldown State

<https://aka.ms/Qujcoz>

Creating parameters

Automatically

You create parameters automatically by adding a query parameter to a dataset. This dataset is either shared or embedded.

Here's an example query that contains a query parameter:

Dataset with a query parameter

```
SELECT      d.CalendarYear [Year],
            d.MonthNumberOfYear [MonthNo],
            d.EnglishMonthName [Month],
            i.SalesOrderNumber,
            i.OrderDate,
FROM dbo.FactInternetSales i ON i.ProductKey = p.ProductKey
INNER JOIN dbo.DimDate d ON i.OrderDateKey = d.DateKey
WHERE d.CalendarYear = @Year
```

- Automatically
 - By adding a query parameter to a dataset
- Manually
 - In either Report Builder or Report Designer

Using the previous query for a dataset would automatically create a **@Year** parameter in Report Designer or Report Builder.

Manually

As you have seen, you can also create parameters manually in Report Designer and Report Builder.

To add a parameter:

1. Right-click the **Parameters** folder in the Report Data pane.
2. Click **Add Parameter**. The **Report Parameter Properties** dialog box appears.
3. Complete the properties for the new parameter.
4. Add a WHERE clause to the query to include the parameter.

For more information about creating report parameters, see Microsoft Docs:




Add, Change, or Delete a Report Parameter (Report Builder and SSRS)

<https://aka.ms/Bmluby>

Report Parameter Properties

Report parameters have a number of properties that determine how they behave. Parameter properties are set in the Report Parameter Properties dialog box. The following table gives details of the most commonly used properties.

For a complete list of report parameter properties, see Microsoft Docs:

 **Report Parameters (Report Builder and Report Designer)**

<http://aka.ms/Trb407>

Properties

Name

Prompt

Data type

Multiple values

Available values

Default values

When published, parameter properties can be edited independently from the reports they are published with.

Property	Description
Name	The parameter name. It cannot contain spaces and must start with a letter. It can contain letters, numbers, and underscore. If this parameter has been automatically created, it will match the defined query parameter.
Prompt	The text shown in the report viewer toolbar of the report.
Data type	<ul style="list-style-type: none">• A parameter can be one of the following values:• Boolean. Displays a radio button—the user selects True or False.• DateTime. Displays a calendar control.• Integer. Displays a text box.• Float. Displays a text box.• Text. Displays a text box.
Allow multiple values	Displays check boxes in the list of values, allowing multiple values to be selected.
Available values	The list of values that can be selected. These values appear in a drop-down list.
Default values	Default values can be hard coded, or selected via a query. If a default value has been supplied, the report will run automatically when viewed.
Hidden	The report parameter is not visible in the published report. However, parameter values can be set on a report URL, subscription definition, or on the report server.
Internal	Hides the report parameter. The report parameter is only viewed in the report definition.

Published parameter properties

After a report has been published to a report server, you modify and update parameters independently from the reports with which they are associated. The properties that can be changed are:

- Whether a default value is used—in some circumstances, the value is set. If the default value is based on a query, then it's not possible to set the value.
- Whether the parameter is hidden, internal or visible.
- The prompt text, if it's visible.

Being able to modify parameters on published reports means the same report can be published to different locations with different default values.

In the case of a report showing all the employees of a department, a hidden department parameter enables a different report to be displayed for each department. The report would show only the employees for that department—without the need for users to enter a parameter.

Default values for report parameters

Report parameters might take a default value that's used if no other value is provided. Default values allow the report to execute with input from the user. After a report has run for the first time, users input a new value instead of the default value.

To specify a default value in Report Builder or Report Designer:

1. Right-click a report parameter, and select **Parameter Properties**.
2. From the left pane, select **Default Values**.
3. To specify a default value, select **Specify values**, and click **Add**. Enter a value or an expression. The value must be the same type as the parameter.
4. To get values from a query, select **Get values from a query**, and then select a dataset and a value field. Click **OK**. Note the warning about performance impact.

- Default report parameters provide a value for a parameter so the report executes with no input
- Add or delete parameters using Parameter Properties
- Specify values or get values from a query

You can also delete a default value, and change the order of default values using the up and down arrows.

For more information about default values for report parameters, see Microsoft Docs:



Add, Change, or Delete Default Values for a Report Parameter

<https://aka.ms/G6rt1d>

Report parameter pane

The report parameter pane displays the parameters that are associated with a report. If a default has been specified, this is displayed—the first time the report is run, the default parameter is used. Users either click to display a list of allowed values for the parameter or they type a value. If possible, it's better to provide a list of allowed values. The parameter pane is displayed at the top of a report.

- The parameters pane is displayed at the top of the report
- In design mode, the parameters pane displays a grid
- Click and drag parameters to reposition them
- Add rows or columns
- Position parameters logically
 - More important first

Changing the position of parameters

Parameters are added to the parameter pane in the order they are created. However, this is not always the most logical order for users. You change the order that the parameters appear in the parameter pane by clicking and dragging parameters to a new position.

Parameters are positioned on a design grid—you select whether parameters go side by side on a line, or beneath each other in a list. You insert rows and columns to the grid, giving a great deal of control over where parameters are positioned, including adding space between parameters.

This ability to lay out parameters in specific places means that you logically group parameters, presenting the most important parameters first, and more detailed parameters later. Consider a report that displays sales data. The most important parameter is the financial year that the data relates to, and so should be displayed first. The region is the next most relevant parameter, followed by the sales person.

Demonstration: Parameters in Report Builder

In this demonstration, you will see how to:

- Add a parameter automatically.
- Set the available values and default values for a parameter.

This demonstration uses Report Builder.

Demonstration Steps

Create a parameter automatically

1. On the toolbar, click **Internet Explorer**.
2. In the address bar, type **mia-sql/Reports_SQL2**, and then press Enter.
3. In the **SQL Server Reporting Services** portal, click **New**, and then **Paginated Report**. Click **Allow**.
4. In Report Builder, click **Blank Report**.
5. If the **Connect to Report Server** dialog box appears, click **Yes**.
6. In **Report Data**, right-click **Data Sources**, then click **Add Data Source**.
7. In the **Data Source Properties** dialog box, in **Name**, type **AdventureWorksDW**.
8. Select **Use a shared connection or report model**, then click **Browse**.

9. In the **Select Data Source** dialog box, double-click **AdventureWorks Sample Reports**, then click **AdventureWorksDW**.
10. Click **Open**, then click **Test Connection**.
11. When Connection created successfully is displayed, click **OK**.
12. Click **OK**.
13. In **Report Data**, right-click **Datasets**.
14. Click **Add Dataset**.
15. Click **Use a dataset embedded in my report**.
16. In **Data Source**, select **AdventureWorksDW**.
17. In the **Query** box, type:

```
SELECT      DimGeography.EnglishCountryRegionName, DimGeography.StateProvinceName,
DimGeography.City, DimReseller.ResellerName, FactResellerSales.SalesOrderNumber,
FactResellerSales.OrderDate, FactResellerSales.SalesAmount,
DimDate.CalendarYear AS SalesYear
FROM
DimGeography INNER JOIN
DimReseller ON DimGeography.GeographyKey = DimReseller.GeographyKey INNER JOIN
FactResellerSales ON DimReseller.ResellerKey = FactResellerSales.ResellerKey INNER
JOIN
DimDate ON [DateKey] = FactResellerSales.OrderDateKey
WHERE DimDate.CalendarYear = @Year
```

18. Click **OK**.
19. Click the **Insert** tab, menu, click **Table**, and then click **Table Wizard**.
20. Select **Dataset1**, and click **Next**.
21. Drag **SalesAmount** to **Values**.
22. Drag **StateProvinceName** to **Row groups**.
23. Drag **City** to **Column groups**.
24. Click **Next**.
25. On the **Choose the layout** page, click **Next**.
26. On the **Preview** page, click **Finish**. The report is displayed in design mode.
27. In the **Report Data** pane, expand **Parameters**. Note that there is now a Year parameter.
28. Right-click **Year**, and click **Parameter Properties**.
29. In the **Report Parameter Properties** dialog box, in the left pane, click **Default Values**.
30. Click **Specify values**, and then click **Add**.
31. In **Value**, enter **2010** and click **OK**.
32. Click **Run**.
33. In **Year**, enter **2011** and then click **View Report**.
34. Click **Design**.

Add available values to a parameter

1. In the **Report Data** pane, right-click **Datasets**, then click **Add Dataset**.
2. In the **Name** box, type **AvailableYears**, then click **Use a dataset embedded in my report**.
3. In the **Data source** drop-down select **AdventureWorksDW**.
4. In the **Query** box, type:

```
SELECT Year(FactResellerSales.OrderDate) AS Year
FROM FactResellerSales
GROUP BY Year(FactResellerSales.OrderDate)
ORDER BY Year(FactResellerSales.OrderDate) DESC;
```

5. Click **OK**.
6. In the Report Data pane, under Parameters, right-click **Year**, then click **Parameter Properties**.
7. In the Report Parameter Properties dialog box, click **Available Values**.
8. Click **Get values from a query**.
9. In the Dataset: (Warning: Possible performance impact) drop-down, select **AvailableYears**.
10. In the **Value field** drop-down, select **Year**.
11. In the **Label field** drop-down, select Year, and then click **OK**.
12. Click **Run**.
13. In the **Year** drop-down list, select **2013** and then click **View Report**.
14. Click the **Design** tab.
15. Close Report Builder without saving changes.

Demonstration: Parameters in Report Designer

In this demonstration, you will see how to:

- Add a parameter automatically.
- Set the allowed values and default values.

This demonstration uses Report Designer.

Demonstration Steps

Create a parameter automatically in Report Designer

1. On the toolbar, click **Visual Studio 2015** if it is not already open.
2. On the **File** menu, point to **New**, click **Project**, and then click **Report Server Project**.
3. In **Name**, type **AutoParam**.
4. In **Location**, navigate to **D:\Demofiles\Mod04**.
5. In **Solution name**, type **AutoParam**.
6. Click **OK**.
7. In Solution Explorer, right-click **Reports** and then click **Add New Report**.
8. In the **Report Wizard**, click **Next**.

9. In **New data source Name**, type **AdventureWorksDW**.
10. In **type**, ensure **Microsoft SQL Server** is selected.
11. Click **Edit**, and in **Server name** type **MIA-SQL**.
12. In **Select or enter a database name**, select **AdventureWorksDW** from the drop-down list.
13. Click **Test Connection**. Test connection succeeded is displayed. Click **OK**.
14. Click **OK**.
15. Click **Next**.
16. In **Query String**, type the following, pointing out the WHERE clause to students:

```
SELECT      DimGeography.EnglishCountryRegionName, DimGeography.StateProvinceName,
DimGeography.City, DimReseller.ResellerName, FactResellerSales.SalesOrderNumber,
FactResellerSales.OrderDate, FactResellerSales.SalesAmount,
DimDate.CalendarYear AS SalesYear
FROM
DimGeography INNER JOIN
DimReseller ON DimGeography.GeographyKey = DimReseller.GeographyKey INNER JOIN
FactResellerSales ON DimReseller.ResellerKey = FactResellerSales.ResellerKey INNER
JOIN
DimDate ON [DateKey] = FactResellerSales.OrderDateKey
WHERE DimDate.CalendarYear = @Year
```

17. Click **Next**.
18. On **Select the Report Type**, leave the selection on **Tabular**. Click **Next**.
19. On **Design the Table**, drag **OrderDate** and **SalesAmount** to **Details >**.
20. Drag **StateProvinceName** and **City** to **Group >** and click **Next**.
21. On **Choose the Table Layout**, click **Block** and select **Include subtotals**, then click **Next**.
22. In **Report name**, type **AutoParam** and then click **Finish**. The report is displayed in design mode. Note that the parameters pane is displayed with the Year parameter.
23. In the **Report Data** pane, expand **Parameters**. Note that there is now a Year parameter.
24. Right-click **Year**, and click **Parameter Properties**.
25. In the **Report Parameter Properties** dialog box, in the left pane, click **Default Values**.
26. Click **Specify values**, and then click **Add**.
27. In **Value**, enter **2010** and click **OK**.
28. Click **Preview**.
29. In **Year**, enter **2011** and then click **View Report**.
30. Click **Design**.

Add available values to a parameter

1. In the **Report Data** pane, right-click **Datasets**, then click **Add Dataset**.
2. In the **Name** box, type **AvailableYears**, then click **Use a dataset embedded in my report**.
3. In the **Data source** drop-down select **AdventureWorksDW**.

4. In the **Query** box, type:

```
SELECT Year(FactResellerSales.OrderDate) AS Year
FROM FactResellerSales
GROUP BY Year(FactResellerSales.OrderDate)
ORDER BY Year(FactResellerSales.OrderDate) DESC;
```

5. Click **OK**.
6. In the **Report Data** pane, expand **Parameters**, right-click **Year**, then click **Parameter Properties**.
7. In the **Report Parameter Properties** dialog box, click **Available Values**.
8. Click **Get values from a query**.
9. In the **Dataset: (Warning: Possible performance impact)** drop-down, select **AvailableYears**.
10. In the **Value field** drop-down, select **Year**.
11. In the **Label field** drop-down, select **Year**, and then click **OK**.
12. Click **Preview**.
13. In the **Year** drop-down list, select **2013** and then click **View Report**.
14. Click the **Design** tab.
15. Close all open windows, without saving any changes.

Lab: Create a parameterized report

Scenario

Staff at Adventure Works Cycles want to make reports more interactive for users. To meet these new business requirements, you must create a parameterized report.

Objectives

After completing this lab, you will have:

- Added parameters to an existing report.
- Set a default parameter for a parameter.
- Set the allowed values for the parameters.
- Filtered the dataset for a parameter based on a user's selection.

Estimated Time: 60 minutes

Virtual machine: **10990C-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa55w.rd**

Exercise 1: Using parameters in Report Designer

Scenario

As a developer for Adventure Works Cycles, you have been asked to consolidate several sales reports into one parameterized report. You have created the sales report, and now need to add the parameters.

The main tasks for this exercise are as follows:

1. Prepare the lab environment
2. Add parameters to a report

► Task 1: Prepare the lab environment

1. Ensure the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab04** folder, right-click **Setup.cmd** and click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**.

► Task 2: Add parameters to a report

Open the sales report

- Open **Visual Studio 2015** and then navigate to **D:\Labfiles\Lab04\Starter\SalesReport** and open **SalesReport.sln**. **SalesReport.rdl** is opened in design mode.

Add two datasets to use as lookups

1. Add an embedded dataset using **AdventureWorksDW** as the data source, and the name **CountryLookup**. Use the query **CountryRegionLookup.sql**.
2. Add a second embedded dataset using **AdventureWorksDW** as the data source, and the name **StateProvinceLookup**. Use the query **StateProvinceLookup.sql**.

Create the Country parameter

1. Add a parameter, with the name **CountryRegion** and prompt **Country**. The data type should be **Text**.
2. Set the parameter to get values from the **CountryLookup** dataset.
3. For the **Value field**, select **CountryRegionCode** and for the **Label field**, select **EnglishCountryRegionName** and click **OK**.
4. Click **Preview** to view the report. Test the drop-down list, noting that the contents of the report do not change.
5. Click **Design**.

Create the StateProvince parameter

1. Add a parameter, with the name **StateProvince** and prompt **State or Province**. The data type should be **Text**, it should allow **multiple values**, and be set to **Visible**.
2. Set the parameter to Get values from the **StateProvinceLookup** dataset.
3. For the **Value field**, select **StateProvinceCode** and for the **Label field**, select **StateProvinceName**.
4. Click and drag the **State or Province** parameter so that it is positioned below the **Country** parameter.
5. Click **Preview** to review the report.
6. Click **Design**.

Amend the report dataset

1. Amend the query for **Dataset1** by adding the following **WHERE** statement above the **GROUP BY** statement:

```
WHERE (G.CountryRegionCode = @CountryRegion) AND
      (G.StateProvinceCode IN (@StateProvince))
```

2. In Parameters, select the matching parameter value from the drop-down list and then click **OK**.

Create date range parameters

1. Create a parameter with the name **DateFrom** and the prompt **Date from**.
2. The data type is **Date/Time**.
3. Set the parameter to **Specify values** with a default of **2010-01-01**.
4. Add another parameter with the name **DateTo** and the prompt **Date to**.
5. The data type is **Date/Time**.
6. Set the parameter to Specify values with a default of **2020-12-31**.
7. Click and drag the parameters so that Date From appears above Date To. The date parameters should be in the column adjacent to the Country and State or Province parameters.
8. Click **Preview** to view the report, change the values and note that the dates do not filter the report.
9. Click **Design**.

Amend the report dataset

1. Right-click **Dataset1** and click **Dataset Properties**.
2. Amend the **WHERE** statement as follows:

```
WHERE (G.CountryRegionCode = @CountryRegion) AND  
      (G.StateProvinceCode IN (@StateProvince)) AND  
      (S.OrderDate BETWEEN (@DateFrom) AND (@DateTo))
```

3. In the left pane, click **Parameters**.
4. Select the correct **Parameter Value** for each Parameter Name
5. Click **OK**.

Link the Country parameter to the StateProvince dataset

1. In the Report Data pane, amend the **StateProvinceLookup** dataset.
2. In Query, add the following WHERE clause:

```
WHERE CountryRegionCode = @CountryRegion
```

3. **Preview** the report and test the **StateProvince** parameter.

Test the parameters

1. Test the parameters by changing the country, and noting that the state or province list is changed.
2. Test different date ranges.
3. Compare your report with the report in **D:\Labfiles\Lab04\Solution\SalesSolution**.
4. Close all open windows, without saving any changes.

Results: After completing this lab exercise, you will be able to:

Add different types of parameters to a report.

Use a dataset to provide values for a parameter.

Use a parameter to change the available values for a parameter.

Exercise 2: Using parameters in Report Builder

Scenario

As a developer for the Adventure Works Bicycle Company, you have been asked to consolidate several sales reports into one parameterized report. You have created the sales report, and now need to add the parameters.

The main tasks for this exercise are as follows:

1. Prepare the lab environment
2. Add parameters to a report

► Task 1: Prepare the lab environment

Prepare the lab environment

1. Ensure the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab04** folder, right-click **Setup.cmd** and click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**.

► Task 2: Add parameters to a report

Open the sales report

1. Start **Report Builder** and then open **SalesReport.rdl** in
2. **D:\Labfiles\Lab04\Starter\SalesReport**.

Add two datasets to use as lookups

1. Add an embedded **dataset** using **AdventureWorksDW** as the data source, and the name **CountryLookup**. Use the query **CountryRegionLookup.sql**.
2. Add a second embedded **dataset** using **AdventureWorksDW** as the data source, and the name **StateProvinceLookup**. Use the query **StateProvinceLookup.sql**.

Create the Country parameter

1. Add a parameter, with the name **CountryRegion** and prompt **Country**. The data type should be **Text**.
2. Set the parameter to get values from the **CountryLookup** dataset.
3. For the **Value field**, select **CountryRegionCode** and for the **Label field**, select **EnglishCountryRegionName** and click **OK**.
4. Click **Run** to view the report. Test the drop-down list, noting that the contents of the report do not change.
5. Click **Design**.

Create the StateProvince parameter

1. Add a parameter, with the name **StateProvince** and prompt **State or Province**. The data type should be **Text**, it should allow **multiple values**, and be set to **Visible**.
2. Set the parameter to Get values from the **StateProvinceLookup** dataset.
3. For the **Value field**, select **StateProvinceCode** and for the **Label field**, select **StateProvinceName**.

4. Click and drag the **State or Province** parameter so that it is positioned below the **Country** parameter.
5. Click **Run** to review the report.
6. Click **Design**.

Amend the report dataset

1. Right-click **Dataset1** and click **Dataset Properties**.
2. Add the following **WHERE** statement to the query:

```
WHERE (G.CountryRegionCode = @CountryRegion) AND
(G.StateProvinceCode IN (@StateProvince))
```

3. In the left pane, click **Parameters**.
4. Next to Parameter Name **@CountryRegion** select the Parameter Value **[@CountryRegion]** from the drop-down box.
5. Next to **@StateProvince**, select **[@StateProvince]**.
6. Click **OK**.

Create date range parameters

1. Create a parameter with the name **DateFrom** and the prompt **Date from**.
2. The data type is **Date/Time**.
3. Set the parameter to **Specify values** with a default of **2010-01-01**.
4. Add another parameter with the name **DateTo** and the prompt **Date to**.
5. The data type is **Date/Time**.
6. Set the parameter to **Specify values** with a default of **2020-12-31**.
7. Click and drag the parameters so that **Date From** appears above **Date To**. The date parameters should be in the column adjacent to the Country and State or Province parameters.
8. Click **Run** to view the report, and then click **Design**.

Amend the report dataset

1. Right-click **Dataset1** and click **Dataset Properties**.
2. Amend the **WHERE** statement as follows:

```
WHERE (G.CountryRegionCode = @CountryRegion) AND
(G.StateProvinceCode IN (@StateProvince)) AND
(S.OrderDate BETWEEN (@DateFrom) AND (@DateTo))
```

3. In the left pane, click **Parameters**.
4. Select the correct Parameter Value for each Parameter Name
5. Click **OK**.

Link the Country parameter to the StateProvince dataset

1. In the Report Data pane, right-click the **StateProvinceLookup** dataset and then click **Dataset Properties**.

2. In Query, add the following **WHERE** clause:

```
WHERE CountryRegionCode = @CountryRegion
```

3. Click **OK**.
4. Click **Run** and test the **StateProvince** parameter.

Test the parameters

1. Test the parameters by changing the country, and noting that the state or province list is changed.
2. Test different **date ranges**.
3. Compare your report with the report in **D:\Labfiles\Lab04\Solution\SalesSolution**.
4. Close all open windows, without saving any changes.

Results: After completing this lab exercise, you will be able to:

Add different types of parameters to a report.

Use a dataset to provide values for a parameter.

Use a parameter to change the available values for a parameter.

Question: How might parameters cause problems to users?

Question: How do parameters make reports valuable to more users?

Module Review and Takeaways

It's common for business requirements to change regarding the information they need and how they want data to be presented. On viewing a detailed report, senior management will likely request higher level, summarized, and filtered versions of the same report.

Report Builder and Report Designer support these scenarios via filtering and the parameterizing of reports.

In this module, you have seen how it's possible to create reports that are dynamic and useful for business users.

MCT USE ONLY. STUDENT USE PROHIBITED

Module 5

Visualizing Data with Reporting Services

Contents:

Module Overview	5-1
Lesson 1: Formatting data	5-2
Lesson 2: Images and charts	5-9
Lesson 3: Databars, sparklines, indicators, gauges, and maps	5-15
Lesson 4: KPIs	5-24
Lab: Manage formatting	5-26
Module Review and Takeaways	5-32

Module Overview

As the amount of data being generated continues to grow, the need to make sense of its meaning increases. You use data visualization to make data easier to understand and faster to interpret.

Data visualizations highlight comparisons, show trends, and convey scale much faster than a table of numbers could. The detail is important but visualizations are a highly effective way of conveying meaning and insights quickly and accurately.

Objectives

After completing this module, you will be able to:

- Format data in reports.
- Use images and charts in your reports.
- Use databars, sparklines, and indicators in reports.
- Explain what a KPI is, and how it's used.

Lesson 1

Formatting data

Source data is normally held without any formatting. For example, \$350 would be held as 350.00 in a database, with the currency common to all values in the column, or the currency type held in a separate column.

When values are displayed in a report, however, they need to be formatted correctly—to show that they represent money, and the type of currency. Without formatting, the value can be misinterpreted.

This lesson discusses how to format data, including numeric, date and time, text, and currency and custom formats.

Lesson Objectives

After completing this lesson, you will be able to:

- Format data in a report.
- Use custom formatting.

Formatting text

Formatting helps to make a report easier to read, by making the meaning of text clearer.



Note: With text formatting, less often means more. You should ensure that formatting aids understanding and draws the eye to what's important.

There are a number of reasons why you might want to format text in your reports:

- To indicate a hyperlink.
- To make headings more (or less) prominent.
- To make totals more (or less) prominent.
- To follow your organization's style.
- To add a tooltip.

- **Format text to:**
 - Indicate a hyperlink
 - Make headings more (or less) prominent
 - Make totals more (or less) prominent
 - Follow an organization's style
- **Format text using home, font group**
 - Bold, italic, underline, and so on
 - Font and size
- **For best results, keep formatting to a minimum**
- **Rotate long headings**

Format text

To format text in Report Builder or Report Designer:

1. Right-click in the field or heading.
2. Click **Text Box Properties** from the drop-down menu.
3. Click **Font** from the left pane, and then format the text appropriately.

Rotate text

When you need to include long text in a report, perhaps in a heading, you rotate text so that it displays at an angle—rather than abbreviating the text.



Note: Rotating text is a property of the text box, not the text itself. Be sure that the text box, and not the text, is selected.

To rotate text:

1. Click the text box you want to rotate.
2. If properties are not displayed, click **View, Properties**.
3. In **WritingMode**, click the down arrow to select **Rotate270**. If properties are categorized, **WritingMode** is in the Localization group.

For more information about formatting text, see Microsoft Docs:



Formatting Text and Placeholders (Report Builder and SSRS)

<https://aka.ms/Xwnufv>

Formatting numbers

A number might represent many different things, such as money, a measurement, a value, an age, and so on. To ensure that numbers are understood, you must format them to include currency symbols and the correct number of decimal places.

Numbers are typically displayed in a table or matrix data region. You should format individual text boxes, and apply the format to the column.

To format a text box in either Report Builder or Report Designer:

- Numeric values must be formatted in the report design
- Select regional formatting, or specific formatting
- Options include:
 - Number of decimal places
 - A "thousand" separator
 - How negative numbers are shown

- Select the text box, right-click, and then select **Text Box Properties**.
- Click **Number**, and select the format you require. In the following table, the sample box shows how values will be displayed, by using the selected formatting options.

Category	Options
Number	Select Use regional formatting to use default regional formatting—alternatively, apply each option separately. The current regional setting is displayed at the foot of the dialog box. Options include decimal places, a "thousand" separator, and showing how negative numbers are displayed. If you select regional formatting, other options are greyed out.
Currency	Format options include the currency symbol and number formatting options. You can also select to show the currency symbol before or after the value, and with or without a space in between.
Date	Select the date format from the list. If the source data includes both the date and time, you select a date/time format.
Time	Select the time format from the list. If the source data includes both date and time, you select a date/time format.

Category	Options
Percentage	Select the number of decimal places, and whether to leave a space between the value and percentage symbol.
Scientific	Select the number of decimal places.
Custom	Enter an expression. Click the expression (fx) button. When entered, a custom expression is used instead of regional and default settings. For more information, see the following topic.

For more information about formatting numbers and dates, see Microsoft Docs:



Formatting Numbers and Dates (Report Builder and SSRS)

<https://aka.ms/Sk0k58>

For more information about entering expressions, see Microsoft Docs:



Expression Uses in Reports (Report Builder and SSRS)

<https://aka.ms/Gnsljm>

Formatting date and time

If you're using SQL Server as the data source, date and time data are held using a number of different data types:

- To format date and/or time:
 - Text box properties
 - Number
 - Date or time
- Choose the format
- A leading asterisk uses the regional settings

Data type	Holds	Comments
time	Time only	
date	Date only	
datetime2	Date and time	
datetimeoffset	Date and time—time zone aware	
datetime	Date and time	Legacy data type
smalldatetime	Date and time	Legacy data type

To format a text box as date and/or time in either Report Builder or Report Designer:

- Select the text box, right-click, and then select **Text Box Properties**.
- Click **Number**.
- Under category, select **Date** or **Time**, depending on the data value.
- Select the date and/or time format from the list. Formats that have a leading asterisk (*) use the regional settings of the report. The regional setting is displayed at the foot of the dialog box.

Custom formats

A custom format uses an expression to format the data. You use the custom format option to perform calculations, and use Visual Basic®.

If you specify a custom format, that will be used instead of regional-specific formatting. If possible, you should use regional formatting rather than custom formats—regional formatting makes reports understandable in different countries. However, there will be occasions when custom formats are required.

Two examples of custom formats are date formatting, and time formatting.

- Custom formats:
 - Enter using an expression
 - Will be used instead of regional formatting
- Use formatting patterns—for example, dd-MM-yyyy

Date formatting

Although there are a number of different date formats available, you choose how to format dates. Use combinations to display dates as required—for example, dd-MM-yyyy, or dd-MMM-yy.

The following table shows the allowed date formatting patterns:

Format pattern	Description
dd	Day number, with a leading zero for single digit days
ddd	Abbreviated name of day
dddd	The full name of day
MM	Month number
MMM	Abbreviated name of the month
MMMM	The full name of the month
yy	Two-digit year
yyyy	Four-digit year



Note: Months are represented by an uppercase “M”. A lowercase “m” is used for minutes.

Time formatting

As with dates, you use custom formatting for the time. The following table shows the allowed time formatting patterns:

Format pattern	Description
hh	The hour using a 12-hour clock
HH	The hour using a 24-hour clock
mm	Minute
ss	Second
ff	Fraction of a second

Demonstration: Defining custom formats

In this demonstration, you will see how to format a text field using a custom format.

Demonstration Steps

Formatting text in Report Designer

1. From the taskbar, click **Visual Studio 2015** to open it.
2. Click **File, Open** and then **Project/Solution**.
3. Navigate to **D:\Demofiles\Mod05\Formatting** folder, click **Formatting.sln** and then click **Open**.
4. In Solution Explorer, double-click **Format.rdl**. The report is opened in design mode.
5. Click **Preview** to view the report.
6. Ask students what formatting problems they see.
 - a. **Sales Amount** and **Unit Price** not formatted as currency.
 - b. **Order Date** displayed as date and time.
 - c. Headings not formatted neatly.
7. Click the **Design** tab, then right-click the **[Order Date]** cell and then click **Text Box Properties**.
8. Click **Number**, and then select **Date**.
9. Select the format **31-Jan-00**.
10. Click **Custom** and point out the format pattern and click **OK**.
11. Right-click the **Order Date** heading cell, and click **Text Box Properties**.
12. Click **Alignment**, and in **Horizontal** select **Right** from the drop-down list. Click **OK**.
13. Right-click the **[SalesAmount]** cell. Click **Text Box Properties**.
14. Click **Number**, and then select **Currency**.
15. In **Symbol**, select **\$ English (United States)** or a symbol appropriate for your country. Click **OK**.
16. Right-click the **[Unit Price]** cell, and then click **Text Box Properties**.

17. Click **Number**, and then select **Currency**.
18. In **Symbol**, select **\$ English (United States)** or a symbol appropriate for your country. Click **OK**.
19. Right-click **[Sum(SalesAmount)]**, and click **Text Box Properties**. Note how the value is calculated.
20. Click **Number**, **Custom** and then click **fx**.
21. Expand **Common Functions**, and select **Aggregate**. Double-click **Avg**.
22. Click **Fields (DataSet1)** and in **Values**, double-click **SalesAmount**.
23. Type **)** (close bracket) to finish the expression, and then click **OK**.
24. Click **OK**.
25. Highlight **Total** and type **Average**. You can optionally format other totals as average.
26. Click within the report, and position the cursor over the first column divider. Repeat for all columns.
27. Click in the **English Product Name** heading cell and delete the word **English**.
28. Click **Preview**.
29. Close the report without saving and leave Visual Studio 2015 open for the next demonstration.

Formatting text in Report Builder

1. Click **Start** then type **Report Builder** then press Enter.
2. Click **Open** and navigate to **D:\Demofiles\Mod05\Formatting**, and double-click **Format.rdl**. The report is opened in design mode.
3. Click **Run** to view the report.
4. Ask students what formatting problems they see.
 - a. Sales amount and Unit price not formatted as currency.
 - b. Order date displayed as date and time.
 - c. Headings not formatted neatly.
5. Click the **Design** tab to return to design mode.
6. Right-click the **[Order Date]** cell and then click **Text Box Properties**.
7. Click **Number**, and then select **Date**.
8. Scroll through the different date options so students see the different formats.
9. In **Category**, click **Custom** and then click **fx**.
10. In **Set expression for: Format**, type **dd-MMM-yyyy** and click **OK**.
11. Click **OK**.
12. Right-click the **Order Date** heading cell, and click **Text Box Properties**.
13. Click **Alignment**, and in **Horizontal** select **Right** from the drop-down list. Click **OK**.
14. Right-click the **[SalesAmount]** cell and click **Text Box Properties**.
15. Click **Number**, and then select **Currency**.
16. In **Symbol**, select **\$ English (United States)**. Or a symbol appropriate for your country. Click **OK**.
17. Right-click the **[Unit Price]** cell, and then click **Text Box Properties**.

18. Click **Number**, and then select **Currency**.
19. In Symbol, select **\$ English (United States)**. Or a symbol appropriate for your country. Click **OK**.
20. Click within the report, and position the cursor over the first column divider. Widen the column to fit the heading. Repeat for all columns.
21. Click in the **English Product Name** heading cell, and delete the word **English**.
22. Click **Run**.
23. Leave the report open for the next demonstration.

Lesson 2

Images and charts

Adding images and charts to reports makes them more attractive and accessible. You use Reporting Services to add images to reports, including logos or images that enhance the report's message. Charts graphically show the data, making it quicker to understand and faster to spot issues.

Lesson Objectives

After completing this lesson, you will be able to:

- Add an image to a report.
- Add a chart to a report.

Images

You enhance the appearance of a report by adding images. Common reasons for including images are:

- **Adding a logo.** You might include a company logo in reports to provide a consistent look and feel that reflects corporate branding.
- **Binding an image to a data field.** For example, a report showing a sales breakdown by product might include a photograph of each product.
- **Background image.** Consider an annual report for a bicycle manufacturing company that includes a subtle image of the corporate headquarters as a background watermark.

- Adding an image to a report when you want to:
 - Add corporate branding to a report
 - Bind an image to a data field
 - Use an image as a background
- Use images from difference sources:
 - External—such as a website
 - Embedded—an image file embedded in the report
 - Database—an image stored in a dataset

Adding an image in Report Designer

To add an image to a report, you drag the **image** icon from the **Toolbox** to the report body then use the **Image Properties** dialog box to configure the image. The key property is **ImageSource**, which can be one of the following three types:

- **External.** An image that's published in an external location such as a website. If you select this option, you must specify the location of the image as a URL in the **Use this image** list box. Alternatively, if the dataset used by the report includes a field that contains a URL, you select this field.
- **Embedded.** An image that's embedded into the report. If you select this option, you import an image file to embed in the report.
- **Database.** An image field that's in the dataset. If you choose this option, you must specify the field to bind to the image control, and the Multipurpose Internet Mail Extension (MIME) type you want to use when you render the image.

To display a background image for a report body or data region, set the following properties in the **BackgroundImage** category of the Properties pane:

- **Source.** Specify external, embedded, or database as described previously.
- **Value.** Specify the source of the image as described previously.

- **MIMEType.** Specify the MIME type to be used for image data fields as described previously.
- **BackgroundRepeat.** Specify whether or not the image should be repeated to fill the background.

Adding an image in Report Builder

To add an image in Report Builder, on the **Insert** tab, click **Image**, and then click the design surface area where you would like the image to be placed. The **Image Properties** dialog box will then appear, enabling you to set properties for the image.

Demonstration: Using images

In this demonstration, you will see how to add images to a report.

Demonstration Steps

Add an image to a report in Report Designer

1. From the taskbar, open **Visual Studio 2015**.
2. Click **File, Open** and then **Project/Solution**.
3. Navigate to **D:\Demofiles\Mod05\Formatting** folder, click **Formatting.sln** and then click **Open**.
4. In Solution Explorer, double-click **Forma.rdl**.
5. Click the **Design** tab.
6. From the **Toolbox**, drag the **Image** icon to the top right of the report. Click and drag to create a small rectangle. The Image Properties dialog box is displayed.
7. In **Name**, type **AWLogo**.
8. In **ToolTip**, type **AdventureWorks Logo**.
9. In **Select the image source**, select **Embedded**.
10. Next to **Use this image**, click **Import**.
11. Navigate to the folder **D:\Demofiles\Mod05**, select the **adventureworks.jpg** image, click **Open**, and then click **OK**.
12. Using the resize handles, make the image larger and position it to the top right corner of the report.
13. Click **Preview** to view the report with the image.
14. Close the report without saving and leave Visual Studio 2015 open for the next demonstration.

Add an Image to a Report in Report Builder

1. Click **Start** then type **Report Builder** then press Enter.
2. Click **Open** and navigate to **D:\Demofiles\Mod05\Formatting**, and double-click **Format.rdl**.
3. Click the **Insert** tab above the ribbon.
4. From the Report Items group, click the **Image icon** to the top right of the report. Click and drag to create a small rectangle. The Image Properties dialog box is displayed.
5. In Name, type **AWLogo**.
6. In ToolTip, type **AdventureWorks Logo**.
7. In Select the image source, select **Embedded**.

8. Next to Use this image, click **Import**.
9. Navigate to the folder **D:\Demofiles\Mod05**, select the **adventureworks.jpg** image, click **Open**, and then click **OK**.
10. Using the resize handles, make the image larger and position it to the top right corner of the report.
11. Click **Run** to view the report with the image.
12. Close the report without saving and leave Report Builder open for the next demonstration.

Charts

You use charts to visually express numeric data values. You might include a chart as a visual summary of the detailed data in a table or matrix, or use just a chart to represent the data. Charts are highly effective for showing comparisons and trends, without having to read the detail.

Adding a chart in Report Designer

To add a chart to a report in Report Designer:

1. Drag a **Chart** item from the **Toolbox** to the body of the report.
2. In the **Select Chart Type** dialog box, select an appropriate type of chart for your data. Available chart types include:
 - Column charts
 - Line charts
 - Shape charts
 - Bar charts
 - Area charts
 - Range charts
 - Scatter charts
 - Polar charts
3. Select the chart to display the **Chart Data** pane, and then specify the fields that the chart will display:
 - a. Add one or more data fields to the **Values** area to specify the fields you need to plot on the value axis of the chart. For example, add a **SalesAmount** field from the dataset showing customer sales orders.
 - b. Add fields to the **Category Groups** area to create the data points for the chart. For example, add a **Year** field from the sales orders dataset to show a summarized value for each year.
 - c. Add fields to the **Series Groups** area to plot values for multiple series. For example, add a **Country** field from the sales orders dataset to show summarized sales values for each country.
4. In properties, edit the chart elements, such as **axis titles**, the **chart legend**, and the **chart title**.

- Charts are a visual representation of numeric data
- Select an appropriate chart type for your data
- Specify the values to be plotted
- Add category groups to define the data points for the chart series
- Add series groups to show multiple series

If you later decide that the data would be better represented using a different visualization, you can change the chart type. For example, you might change a column chart showing sales grouped by year to a line chart that displays a trend over time more clearly.

Adding a chart in Report Builder

To add a chart to a report in Report Builder:

1. On the **Insert** tab, click **Chart**, and then click Insert **Chart**.
2. Drag to an area on the design surface where you want the chart to appear.
3. In the **Select Chart Type** dialog, choose an appropriate chart type.
4. Click the newly inserted chart so that the **chart data** drop zones appear.
5. Place the required data items from a dataset onto the chart drop zones.

Demonstration: Using charts

In this demonstration, you will see how to add a chart to a report.

Demonstration Steps

Add a chart to a report in Report Designer

1. From the taskbar, open **Visual Studio 2015**.
2. Click **File, Open** and then **Project/Solution**.
3. Navigate to **D:\Demofiles\Mod05**. Double click the **InternetSales** folder.
4. Click **InternetSales.sln** and then click **Open**.
5. In Solution Explorer, double-click **InternetSales.rdl**. The report is opened in design mode.
6. Click the tablix data region so that the gray row and column headers appear, and click the gray box where the row and column headers intersect at the top left to select the data region. Then drag the multidirectional arrow handle to move the data region down to make room for a good sized chart.
7. From the Toolbox, click the **chart** icon and click and drag a large rectangular to fit the area you just created. The Select Chart Type dialog box is displayed.
8. In the **Select Chart Type** dialog box, select first column chart style and click **OK**. Optionally browse the chart types to show what is available.
9. Click the **chart** to display Chart Data.
10. In Values, click + and then select **SalesAmount**.
11. In Category Groups, click + and then select **Month**.
12. In the Series Groups, click + and then select **Year**.
13. Click **Preview** to see the chart. Ask the students what issues they see with the chart.
 - a. Months are displayed in the wrong order.
 - b. Chart title placeholder.
 - c. Vertical axis is not formatted.
14. Click the **Design** tab.

15. Click the **chart** to display Chart Data.
16. Click the **down arrow** next to Month and click **Category Group Properties**. The Category Group dialog box is displayed.
17. Click **Sorting** in the left pane.
18. In Sort by, select **Month No**. Click **OK**.
19. Click the **Chart title** to select it, and click **Delete**.
20. Right-click the Vertical axis values and click **Vertical Axis Properties**.
21. Click **Number**, and then click **Currency**.
22. In Decimal places, click the down arrow twice to reduce the decimal places to **0**.
23. Click **Show values in Thousands**. Click **OK**.
24. Click **Preview**.
25. Close the report without saving and leave Visual Studio 2015 open for the next demonstration.

Add a chart to a report in Report Builder

1. Click **Start** then type **Report Builder** then press Enter.
2. Click **Open** and navigate to **D:\Demofiles\Mod05\InternetSales**, and double-click **InternetSales.rdl**. The report is opened in design mode.
3. Click the tablix data region so that the gray row and column headers appear, and click the gray box where the row and column headers intersect at the top left to select the data region. Then drag the multidirectional arrow handle to move the data region down to make room for a good sized chart.
4. From the Menu, click **Insert, Chart** icon and click and drag a large rectangular to fit the area you just created. The Select Chart Type dialog box is displayed.
5. In the Select Chart Type dialog box, select first **column chart** style and click **OK**. Optionally browse the chart types to show what is available.
6. Click the **chart** to display Chart Data.
7. In Values, click **+** and then select **SalesAmount**.
8. In Category Groups, click **+** and then select **Month**.
9. In the Series Groups, click **+** and then select **Year**.
10. Click **Run** to see the chart. Ask the students what issues they see with the chart.
 - a. Months are displayed in the wrong order.
 - b. Chart title placeholder.
 - c. Vertical axis is not formatted.
11. Click **Design**.
12. Click the **chart** to display Chart Data.
13. Click the down arrow next to **Month** and click **Category Group Properties**. The Category Group Properties dialog box is displayed.
14. Click **Sorting** in the left pane.
15. In Sort by, select **Month No**. Click **OK**.

16. Click the **Chart title** to select it, and click **Delete**.
17. Right-click the Vertical axis values and click **Vertical Axis Properties**.
18. Click **Number**, and then click **Currency**.
19. In Decimal places, click the down arrow twice to reduce the decimal places to **0**.
20. Click the checkbox next to **Show values in Thousands**. Click **OK**.
21. Click **Run**.
22. Close Report Builder without saving your changes.

Question: What are the benefits of adding a chart to a report?

Lesson 3

Databars, sparklines, indicators, gauges, and maps

This lesson discusses databars, sparklines, indicators, gauges, and maps.

Lesson Objectives

After completing this lesson, you will be able to explain:

- What a databar is, and when to use one.
- What a sparkline is, and when to use one.
- What an indicator is, and when to use one.
- What a gauge is, and when to use one.
- How to use maps.

Databars

Data bars are mini charts that are displayed as either a horizontal bar or a vertical column. Because databars are small, they can be displayed in line with the data, providing a visual short cut to understanding.

Data bars provide the same visual representation of the data as a full size chart—but without the legends, axis lines, labels, and so on. Data bars are ideal for conveying comparisons and trends in a very small space. Data bars help potential issues and outliers to be identified more quickly than would be possible without a graphic.

- Data bars are mini charts
- They convey a lot of information in a small space
- They're used inline—that is, next to the value they represent
- They aid fast understanding and identifying of issues

Data bars either represent a single value, or multiple values. Single values display the relative size of the bar or column, and multiple values display stacked bars or columns.

Adding data bars using Report Designer

1. Create a table or matrix in your report.
2. Insert a new column in the table or matrix.
3. From the **Toolbox**, drag the **data bar** icon to a text box in a table or matrix. The **Select Data Bar Type** dialog box is displayed.
4. Select either **Data Bar** or **Data Column**. Click **OK**.
5. Right-click the data bar, and select **Chart Properties**.

For more information about adding data bars to your reports, see Microsoft Docs:

 **Add Sparklines and Data Bars (Report Builder and SSRS)**

<https://aka.ms/Vevbxa>

Sparklines

Sparklines are small graphics that are used to represent data in a report. They are similar to data bars, but display a trend rather than a single value. Sparklines display summaries of groups, and cannot be used in a detail group. As with data bars, sparklines provide an effective way to communicate.

There's a variety of different sparkline types, with different styles in each:

- Column
- Line
- Area
- Shape
- Range

- Sparklines illustrate trends
- Used at the group level, not the detail level
- Can be columns, lines, area, shape, or range

To add a sparkline to a table or matrix:

1. Create a **table** or **matrix** in your report.
2. Insert a **column** in the table or matrix.
3. Create a **group** level.
4. From the **Toolbox**, drag the **Sparkline** icon to a text box at the group level. The **Select Sparkline Type** dialog box is displayed.
5. Select the type of sparkline you want. Choose from column, line, area, shape, or range.
6. Click the **sparkline**. The **Chart Data** pane is displayed.
7. In Values, click **+**. A list of fields from your dataset is displayed.
8. Click the **field** you want to use to calculate the sparkline. The field is displayed under Values.
9. Click the **down arrow** next to Sum, and then click **Aggregate**. Select the aggregation method for the field.
10. In Category Groups, click **+**. A list of fields from your dataset is displayed.
11. Click the **field** that you want to group by.
12. Click **Preview** to view the sparkline in your report.



Note: Sparklines must be positioned at group level, not in a detail group—they are designed to summarize data.

For more information about adding sparklines to your reports, see Microsoft Docs:



Add Sparklines and Data Bars (Report Builder and SSRS)

<https://aka.ms/Vevbxa>

Indicators

Indicators are small graphics that show trends, ratings, status, or the condition of something. They are used to quickly convey meaning in a report. For example:

- Indicators are small graphics that show:
 - Trend
 - Status
 - Ratings

- **Trends.** Add an arrow pointing up, down, or horizontal to indicate whether values are increasing, decreasing, or staying the same. This works well for sales results, readership figures, or viewing figures.
- **Status.** Add traffic light colored symbols or shapes to indicate whether something is on track, behind, or neutral. This works well for the status of a project, or whether a sales person is on target to meet their sales quota.
- **Ratings.** Use rating stars, bars, pies or blocks to indicate whether something is rated highly, or badly. This works well for customer satisfaction, rating products or services, and the likelihood to recommend something.

Indicators are used with either numeric or percentage values:

- **Numeric.** Define the range of values that will determine how the indicator is displayed. Specify a start and end value for each icon in the indicator set. These start and end values are either absolute numbers or are derived at runtime from an expression. Numeric indicators are useful when you want to show how a particular data value compares to a target value.
- **Percentage.** Specify the start and end values for each icon. This range is usually between 0 and 100. The indicator will show the appropriate icon based on where, in that percentile range, a particular instance of a data value falls across the grouping. The percentage scale is determined by the minimum and maximum values specified for the indicator. In most cases, this is determined automatically by taking the lowest value for the specified data field in the grouping as the minimum and the highest value as the maximum. If you override the minimum and maximum values, any rows with a value outside the range you specify will not have an indicator icon displayed. Percentage indicators are useful when you want to show how rows in a data grouping compare with one another; for example, to grade salespeople by the volume of sales they have achieved.

Add an indicator to a report in Report Designer

1. Create a table or matrix in your report.
2. Insert a **column** in the table or matrix.
3. From the **Toolbox**, drag the **Indicator** icon to a text box in the new column. The **Select Indicator Type** dialog box is displayed.
4. Select the type of indicator you want. Choose from directional, symbols, shapes, or ratings. Click **OK**.
5. Right-click the indicator icon and click **Indicator Properties**.
6. In Values and States, select the **Value** and **Measurement Unit**, in addition to **Indicator States**.
7. Click **OK**.
8. Click **Preview** to view the indicator in your report.

Add an indicator to a report in Report Builder

1. Create a table or matrix in your report.
2. Insert a **column** in the table or matrix.
3. From the **Insert** menu, drag the **Indicator** icon to a text box in the new column. The **Select Indicator Type** dialog box is displayed.
4. Select the type of indicator you want. Choose from directional, symbols, shapes, or ratings. Click **OK**.
5. Right-click the **indicator** icon and click **Indicator Properties**.
6. In Values and States, select the **Value and Measurement Unit**, in addition to **Indicator States**.
7. Click **OK**.
8. Click **Run** to view the indicator in your report.

For more information about adding indicators to your reports, see Microsoft Docs:



Add or Delete an Indicator (Report Builder and SSRS)

<https://aka.ms/Eh7d3w>

Maps

You use maps to display geographical data in a report. When you add one or more layers to a map, each layer can define:

- Spatial data to define the geographical features to be displayed.
- Analytical data to show business values that are related to the spatial data.
- Legends and scales to help viewers interpret the map.

- Add a map to a report to show geographic data
- A map consists of one or more layers
- Each map layer might define:
 - Spatial features
 - Analytical values
 - Legends and scales to help interpret the map
- Add a Bing Maps layer to show geographic details

Add a map to a report in Report Designer

To add a map to a report:

1. Drag a **Map** item from the **Toolbox** to the report canvas. The **New Map Layer** wizard is displayed.
2. On the **Choose a source of spatial data** page, select the method of defining the maps. This will define points, lines, or polygons that represent spatial locations and features. You obtain the spatial data for a map layer from:
 - **Map gallery.** This is provided with SQL Server Reporting Services and contains a number of maps of the United States.
 - **ESRI shapefile.** An Environmental Systems Research Institute (Esri)-compatible shapefile. If you choose this option, you are prompted to provide the dataset.
 - **SQL Server spatial query.** A query that returns spatial data from a SQL Server database. If you choose this option, you are prompted to provide the dataset.
3. On the **Choose spatial data and map view options** page, select an appropriate map resolution and size, and optionally add a Bing Maps layer to overlay the map with geographical imagery.

4. On the **Choose map visualization** page, select the type of map you want to create. The options available are:
 - **Basic map.** This map shows spatial elements with no analytical data. Typically, a basic map shows polygons that represent geographical areas, such as states or sales territories.
 - **Color analytical map.** You add analytical data to a basic map and use it to format areas in different colors to represent the analytical values. For example, you compare figures in multiple sales territories by coloring or shading each territory, based on the sales value.
 - **Bubble map.** You use different sized bubbles to indicate analytical values for each area shown by the map. For example, each sales territory might have a bubble at its center—the size of the bubble indicates the relative sales volume for that territory.
5. The **Choose the analytical dataset** page is displayed if you choose a map visualization that requires analytical data. Select an existing dataset or add a new dataset. The analytical dataset must include a field in common with the spatial dataset, so you relate each instance of the analytical values to a spatial feature or location. For example, you might define a map layer that contains spatial data from a Transact-SQL query that returns the ID and geographical location of each store, and analytical data from a query that returns the store ID and total sales. You then use the common field (the store ID) to match the analytical data (the total sales) to the spatial data (the geographical location). Alternatively, you might create a map layer using a US state map from the map gallery and add an analytical dataset that includes sales volumes grouped by state code. You then match the state code field in the analytical dataset to the United States Postal Service (USPS) state code attribute in the US state map dataset.
6. You use the **Specify the match fields for spatial and analytical data** page to specify the fields to match on. Select a spatial dataset field, and select a field from your dataset.
7. On the **Choose color theme and data visualization** page, select a visual theme and specify which field to visualize on the map. For example, you could visualize sales data in an analytical dataset by using a color rule that colors states on the map as a shade of green, yellow, or red, depending on the sales volume for each state.

After you've added a map, you add additional map layers, including Bing Maps imagery, by using the Map Layer Wizard.

Add a map to a report in Report Builder

To add a map to a report:

1. On the **Insert** menu, point to **Map**, and then click **Map Wizard**.
2. On the **Choose a source of spatial data** page, select a source for the spatial data that will be used to define the map. This will define points, lines, or polygons that represent spatial locations and features. You obtain the spatial data for a map layer from:
 - **Map gallery.** This is provided with SQL Server Reporting Services and contains a number of maps of the United States.
 - **ESRI shapefile.** An Environmental Systems Research Institute (Esri)-compatible shapefile. If you choose this option, you will be prompted to provide the data.
 - **SQL Server spatial query.** A query that returns spatial data from a SQL Server database. If you choose this option, you will be prompted to provide the dataset.
3. On the **Choose spatial data and map view options** page, select an appropriate map resolution and size, and optionally add a Bing Maps layer to overlay the map with geographical imagery.

4. On the **Choose map visualization** page, select the type of map you want to create. The options available are:
 - **Basic map.** This map shows spatial elements that have no analytical data. Typically, a basic map shows polygons that represent geographical areas, such as states or sales territories.
 - **Color analytical map.** You add analytical data to a basic map and use it to format areas in different colors to represent the analytical values. For example, you might compare figures in multiple sales territories by coloring or shading each territory based on the sales value.
 - **Bubble map.** You use different sized bubbles to indicate analytical values for each area shown by the map. For example, each sales territory might have a bubble at its center—the size of the bubble indicates the relative sales volume for that territory.
5. The **Choose the analytical dataset** page is displayed if you choose a map visualization that requires analytical data. Select an existing dataset or add a new dataset. The analytical dataset must include a field in common with the spatial dataset, so you relate each instance of the analytical values to a spatial feature or location. For example, you might define a map layer that contains spatial data from a Transact-SQL query that returns the ID and geographical location of each store, and analytical data from a query that returns the store ID and total sales. You then use the common field (the store ID) to match the analytical data (the total sales) to the spatial data (the geographical location). Alternatively, you might create a map layer using a US state map from the map gallery, and add an analytical dataset that includes sales volumes grouped by state code. You then match the state code field in the analytical dataset to the United States Postal Service (USPS) state code attribute in the US state map dataset.
6. You use the **Specify the match fields for spatial and analytical data** page to specify the fields to match on. Select a spatial dataset field, and select a field from your dataset.
7. On the **Choose color theme and data visualization** page, select a visual theme and specify which field to visualize on the map. For example, you could visualize sales data in an analytical dataset by using a color rule that colors states on the map as a shade of green, yellow, or red, depending on the sales volume for each state.

After you have added a map, you add additional map layers, including Bing Maps imagery, by using the Map Layer Wizard.



Note: You can only use Bing Maps when connected to the internet.

Gauges

Gauges are another graphical device that you use to make reports more compelling. There are two types of gauges that you use in reports:

- **Radial.** A circular graphic that looks like a car's speedometer. A radial gauge might display a full circle, or part of a circle.
- **Linear.** A rectangular graphic that looks like a ruler or thermometer.

Gauges are used to represent KPIs or other data. Gauges are typically used on dashboards.

A gauge typically displays a single value.

- Gauges can be radial or linear
- Radial: circular like a car's speedometer
- Linear: like a ruler or thermometer
- Commonly used on dashboards

To add a gauge to a report:

1. From the **Toolbox** or **Insert** menu, click **Gauge**.
2. Click and drag a rectangle on the report.
3. Drag a field from the dataset to the gauge.
4. Click the **Gauge** to display **Gauge Data**.
5. Amend the **Gauge Properties**, **Scale Properties**, **Pointer Properties**, and so on.



Note: You cannot change the gauge type after it has been created. To change the gauge type, delete the gauge and recreate it.

For more information about using gauges in your reports, see Microsoft Docs:



Gauges (Report Builder and SSRS)

<https://aka.ms/Otvvp1>

Demonstration: Using maps

In this demonstration, you will see how to add maps to a report.

Demonstration Steps

Add a map to a report in Report Designer

1. Open **Visual Studio 2015**, click **File, Open, Project/Solution**, and navigate to **D:\Demofiles\Mod05\Mapping**.
2. Select **Mapping.sln** and click **Open**.
3. In **Solution Explorer**, under Reports, double-click **Mapping.rdl**.
4. In Design mode, click the tablix data region so that the gray row and column headers appear, and click the gray box where the row and column headers intersect at the top left to select the data region. Then drag the multidirectional arrow handle to move the data region down to make room for a good sized map.
5. From the Toolbox, click the **Map** icon and then click and drag a large rectangle in the area above the tablix. The **New Map Layer** wizard is displayed.
6. On the **Choose a source of spatial data** page, select **Map Gallery** and **USA by State Exploded**, and then click **Next**.
7. On the **Choose spatial data and map view options** page, click **Next**.
8. On the **Choose map visualization** page, click **Bubble Map**, and then click **Next**.
9. On the **Choose the analytical dataset** page, click **Dataset1**, and then click **Next**.
10. On the **Specify the match fields for spatial and analytical data** page, click the check box in the Match Fields column next to **STATENAME**, in the Select a field drop-down list, select **StateProvinceName**, and click **Next**.

11. On the **Choose color theme and data visualization** page, clear **Use bubble sizes to visualize data**, check the **Use polygon colors to visualize data**. In the Data field, select **[Sum(SalesAmount)]**, in the Color rule, select **White-Blue**. Click **Display labels** and in Data field select **#STUSPS**. Click **Finish**.
12. Double-click **Map Title** and then over-type **Sales by State**.
13. Right-click the **Map Legend** with the Title text, and then select **Text box Properties**. Click **Visibility** and then **Hide**. Click **OK**.
14. Click **Preview** to view the completed report.
15. Close Visual Studio 2015, without saving changes.

Add a map to a report in Report Builder

1. Click **Start**, type **Report Builder** then press Enter.
2. Click **Open** and navigate to **D:\Demofiles\Mod05\Mapping**, and double-click **Mapping.rdl**. The report is opened in design mode.
3. In Design mode, click the tablix data region so that the gray row and column headers appear, and click the gray box where the row and column headers intersect at the top left to select the data region. Then drag the multidirectional arrow handle to move the data region down to make room for a good sized map.
4. From the menu click **Insert, Map**, and then **Map Wizard**. The New Map wizard is displayed.
5. On the **Choose a source of spatial data** page, select **Map gallery** and **USA by State Exploded**, and then click **Next**.
6. On the **Choose spatial data and map view options** page, click **Next**.
7. On the **Choose map visualization** page, click **Bubble Map**, and then click **Next**.
8. On the **Choose the analytical dataset** page, click **Dataset1**, and then click **Next**.
9. On the **Specify the match fields for spatial and analytical data** page, select the check box in the **Match Fields** column next to **STATENAME**, in the **Select a field** drop-down list, select **StateProvinceName**, and click **Next**.
10. On the **Choose color theme and data visualization** page, clear **Use bubble sizes to visualize data**, check the **Use polygon colors to visualize data**. In the **Data field**, select **[Sum(SalesAmount)]**, in the **Color rule**, select **White-Blue**. Click the check box next to **Display labels**. In Data field select **#STUSPS** and then click **Finish**.
11. Double-click **Map Title** and then over-type **Sales by State**.
12. Right-click the Map Legend with the **Title** text, and then select **Legend Properties**. Click **Visibility** and then **Hide**. Click **OK**.
13. Click **Run** to view the completed report.
14. Close **Report Builder**, saving changes if you are prompted.

Check Your Knowledge

Question	
Where should you position sparklines on a report?	
Select the correct answer.	
<input type="checkbox"/>	At the foot of the report
<input type="checkbox"/>	In the header
<input type="checkbox"/>	At the group level
<input type="checkbox"/>	At the detail level

Lesson 4

KPIs

This lesson discusses key performance indicators, commonly referred to as KPIs.

Lesson Objectives

After completing this lesson, you will be able to use KPIs in the web portal.

What is a KPI?

A key performance indicator, or KPI, is a measure of performance. Each KPI comprises a goal, and a result.

KPIs provide a popular management tool you use to measure important aspects of a business, a team, or a person's performance. KPIs are often represented visually to make them easy and fast to understand. This means that anyone can see, at a glance, whether you are on target to meet the goal, or whether there are problems.

- A KPI is a management tool to assess whether performance is on track or behind
- Each KPI comprises a goal, and a result
- Often represented visually
- Used to:
 - Assess whether projects (or people) are on target or behind
 - Manage a number of different goals at the same time
 - Prioritize actions

How are KPIs used?

KPIs make it possible to:

- Assess whether projects are on target or behind schedule.
- Manage a number of different goals at the same time, as progress is seen very quickly.
- Prioritize actions, either according to the importance of the project, or by assessing how far away it is from the goal.

Implementing KPIs

KPIs are driven either from a dataset, or by setting the values manually. You create KPIs in the Reporting Services web portal, in their own folder. A Reporting Services KPI provides all the information that relates to the KPI, like a mini report.

To create a new KPI, you need either a dataset that contains the relevant information, or the data that you input manually. You then create a new KPI in the Reporting Services portal.

To create a new KPI:

- Open the **SQL Server Reporting Services** web portal.
- Navigate to the correct folder.
- Click **New**, then click **KPI**. The new KPI screen is displayed.

- Create a KPI in the Reporting Services web portal
- KPIs can be driven from:
 - A dataset
 - Entering values manually
- A Reporting Services KPI is like a mini report
- Create KPIs in the web portal
 - New
 - KPI

- The fields are described in the following table:

Field	Description
Name	KPI name. This might include spaces, but not the characters / @ \$ & * + = > < \ : , ' ?
Description	Description of the KPI.
Hide in tile view	
Value format	<ul style="list-style-type: none"> Drop-down list. Select from: Abbreviated Currency ... Currency with decimals ... Abbreviated currency ... Percent Percent with decimals
Value	This is the KPI value that is compared to the goal. This is set manually or taken from the first row of a dataset. The value is shown as a percentage of the goal.
Goal	The goal value is used to compare the KPI value. The goal is either set manually or taken from the first row of a dataset.
Status	The status is either set manually or taken from the first row of a dataset. If set manually, this is set to 1 (good), 0 (neutral), or -1 (bad).
Trend set	

For more information about implementing KPIs in Reporting Services, see Microsoft Docs:

 **Working with KPIs in Reporting Services**

<https://aka.ms/Ngqhxz>

Lab: Manage formatting

Scenario

The sales manager at Adventure Works Cycles spends a lot of time creating reports that show sales by product category, subcategory, and individual product.

The sales manager has requested a solution that generates the required report on demand. To accomplish this, you will create a formatted report that includes a chart, and add a map to an existing report.

Objectives

After completing this lab exercise, you will be able to:

- Use formatting.
- Add a chart to a report.
- Add a map to a report.

Estimated Time: 45 minutes

Virtual machine: **10990C-MIA-SQL**

User name: ADVENTUREWORKS\Student

Password: **Pa55w.rd**

Exercise 1: Report Designer

Scenario

You are a developer at Adventure Works Cycles. You've been asked to improve some reports that were created by a previous developer but were never finished. You will format a report, and add a chart and map.

The main tasks for this exercise are as follows:

1. Prepare the environment
2. Formatting a report
3. Add a chart to a report in Report Designer
4. Add a map to a report using Report Designer
5. Adding databars and sparklines to a report using Report Designer

► Task 1: Prepare the environment

1. Ensure the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab05** folder, right-click **Setup.cmd** and click **Run as administrator**.
3. Click **Yes** when prompted.

► Task 2: Formatting a report

1. Open **Visual Studio 2015** and open the project called **Visuals** located in **D:\Labfiles\Lab05\Starter**.
2. Open the report **Visuals.rdl**.
3. Format the **Order Date** detail cell with a custom format **dd-MMM-yy**.
4. Right-align **Order Date** detail cell, and the **heading**.

5. **Widen** the columns appropriately.
6. Format the **SalesAmount** detail cell with:
 - a. A **currency symbol** appropriate to your country.
 - b. Zero decimal places.
 - c. A thousands separator.
7. Format the group **Sum(SalesAmount)** cell with:
 - a. A currency symbol appropriate to your country.
 - b. Zero decimal places.
 - c. A thousands separator.
8. Format the grand total **Sum(SalesAmount)** cell with:
 - a. A currency symbol appropriate to your country.
 - b. Zero decimal places.
 - c. A thousands separator.
9. **Preview** the report.
10. **Save** the report.

► **Task 3: Add a chart to a report in Report Designer**

1. Open **Visual Studio 2015**, if not already open.
2. Navigate to **D:\Labfiles\Lab05\Starter\Visuals** and open **Chart.rdl**.
3. Move the Tablix area down about 10cm to create space for the chart.
4. From the **Toolbox**, click the **chart** icon and drag a large rectangle in the space you have just created above the Tablix area. The Select Chart Type dialog box is displayed.
5. Select the first **bar chart** style and click **OK**.
6. Click the chart to display **Chart Data**, and:
 - a. In Values add **SalesAmount**.
 - b. In Category Groups add **Product**.
 - c. In Series Groups add **StateProvinceName**.
7. Delete the **chart title**.
8. **Preview** the chart.
9. Return to the **Design** mode.

► **Task 4: Add a map to a report using Report Designer**

1. Open **Visual Studio 2015**, if not already open.
2. Navigate to **D:\Labfiles\Lab05\Starter** and open the report **map.rdl**.
3. Move the tablix data region down about 10cm.
4. From the Toolbox, select the **Map** icon, and then drag a large rectangle in the space above the tablix area. The New Map Layer wizard is displayed.
5. From the **Map Galley**, select **USA by State Exploded**.

6. Do not change the options on Chooses spatial data and map view options.
7. Select **Bubble Map** as they type of visualization.
8. Select **Dataset1** as they dataset.
9. Uses **STATENAME** as the match field, matching to **StateProvinceName** in the dataset.
10. To visualize the data, uses **polygon colors** instead of bubble sizes. In data field, select **#STUSPS**. For the color rule, select **White-Green**.
11. Add **#STUSPS** labels to the map.
12. **Hide** the **map legend** so it does not display on the map.
13. Overtyping the map title to read **Sales by State**.
14. **Preview** the map.
15. Return to **Design** mode.

► **Task 5: Adding databars and sparklines to a report using Report Designer**

1. Open **Visual Studio 2015** if it is not already open.
2. Navigate to **D:\Labfiles\Lab05\Starter**. Open the **Sparklines** solution,
3. In Solution Explorer, double-click **Sparklines.rdl** to open the report in design mode.
4. Insert a column to the right of the Sales Value column.
5. In the new column, add the heading **Comparison by Category**.
6. In the new cell below the heading, add a **databar** with the default type.
7. Select **Chart Data**, and add **Sales Value** in Values.
8. In Horizontal Axis Properties, set the Align Axes to **table1_EnglishProductCategoryName**.
9. Set the color of the data bars to **Earth tones**.
10. **Insert** a column to the right of the Comparison by Category column.
11. Name the new column **Sales by Year**.
12. In the new cell beneath Sales by Year, add a **sparkline** of type **column**.
13. In Chart Data, add **Sales Value** as the value, and **Year** in Category Groups.
14. In Horizontal Axis Properties, for Align axes in, select **table1**.
15. In **Interval**, select **fx** and overtype **1** in Set expression for.
16. **Preview** the report.
17. Close Visual Studio 2015.

Results: After completing this lab exercise, you will be able to:

Format a report.

Add a chart to a report.

Add a map to a report.

Add databars and sparklines to a report.

Exercise 2: Report Builder

Scenario

You are a developer at Adventure Works Cycles. You've been asked to improve some reports that were created by a previous developer but were never finished. You will format a report, and add a chart and map.

The main tasks for this exercise are as follows:

1. Prepare the environment
2. Formatting a report using Report Builder
3. Add a chart to a report using Report Builder
4. Adding a map to a report using Report Builder
5. Adding data bars and sparklines to a report using Report Builder

► Task 1: Prepare the environment

1. Ensure the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab05** folder, right-click **Setup.cmd** and click **Run as administrator**.
3. Click **Yes** when prompted.

► Task 2: Formatting a report using Report Builder

1. Open **Report Builder**.
2. Open the report called **VisualsBuilder.rdl** located in **D:\Labfiles\Lab05\Starter\Visuals**.
3. Format the **Order Date** detail cell with a custom format **dd-MMM-yy**.
4. Right-align **Order Date** detail cell, and the heading.
5. **Widen** the columns appropriately.
6. Format the **SalesAmount** detail cell with:
 - a. A currency symbol appropriate to your country.
 - b. Zero decimal places.
 - c. A thousands separator.
7. Format the group **Sum(SalesAmount)** cell with:
 - a. A currency symbol appropriate to your country.
 - b. Zero decimal places.
 - c. A thousands separator.
8. Format the grand total **Sum(SalesAmount)** cell with:
 - a. A currency symbol appropriate to your country.
 - b. Zero decimal places.
 - c. A thousands separator.
9. Click **Run** to view your changes.
10. Leave Report Builder open for the next lab exercise.

► Task 3: Add a chart to a report using Report Builder

1. Open **Report Builder**, if not already open.
2. From **D:\Labfiles\Lab05\Starter\Visuals** open **ChartBuilder.rdl**.
3. Move the Tablix data region down about 10cm to make room for the chart.
4. From the **Insert** menu, select **Chart**, and then **Chart Wizard**.
5. Complete the wizard as follows:
 - a. In Chooses a dataset, select **DataSet1**.
 - b. In Choose a chart type, select **Bar**.
 - c. From Available Fields, drag:
 - i. **SalesAmount** to **Values**.
 - ii. **Product** to **Categories**.
 - iii. **StateProvinceName** to **Series**.
6. **Preview** the chart and click **Finish**.
7. **Position** and **size** the chart to fit the area.
8. **Delete** the chart title.
9. Click **Run** to view the report.
10. Click **Design**.

► Task 4: Adding a map to a report using Report Builder

1. If Report Builder is not already open it, click **Start** and type **Report Builder** to open it.
2. Click **Open** and navigate to **D:\Labfiles\Lab05\Starter\Visuals**.
3. Open **MapBuilder.rdl**.
4. Click and drag the Tablix data region to create a space to add a map.
5. From the **Insert** menu, click **Map**, and then **Map Wizard**.
6. Select **USA by State Exploded** from the Map Gallery, and click **Next**.
7. On the Choose spatial data and map view options page, click **Next** without making changes.
8. On the Choose map visualization page, select **Bubble Map**.
9. On the Choose the analytical dataset page, select **Dataset1**.
10. On the Specify the match fields for spatial and analytical data page, select **STATENAME** and **StateProvinceName**.
11. On the Choose color theme and data visualization page, clear the current selection and select **Use polygon colors to visualize data**. Select **[Sum(SalesAmount)]** as the data field, and **White-Green** for the color rule. Select to **Display labels**, and select **#STUSPS**.
12. Click **Finish**.
13. Delete the **Map Title**.
14. Make the **Map Legend** hidden.
15. View the completed report.

► **Task 5: Adding data bars and sparklines to a report using Report Builder**

1. Open **Report Builder** if it is not already open
2. Click **Open** and navigate to **D:\Labfiles\Lab05\Starter\Sparklines** and open **SparklinesBuilder.rdl**.
3. Insert a column to the right of the **Sales Value** column.
4. In the new column, add the heading **Comparison by Category**.
5. In the new cell below the heading, add a **databar** with the default type.
6. Select **Chart Data**, and add **Sales Value** in Values.
7. In Horizontal Axis Properties, set the Align Axes to **table1_EnglishProductCategoryName**.
8. Set the color of the data bars to **Earth tones**.
9. **Insert a column** to the right of the Comparison by Category column.
10. Name the new column **Sales by Year**.
11. In the new cell beneath Sales by Year, add a **sparkline** of type **column**.
12. In Chart Data, add **Sales Value** as the value, and **Year** in Category Groups.
13. In Horizontal Axis Properties, for Align axes in, select **table1**.
14. In Interval, select **fx** and overwrite **1** in Set expression for.
15. **Run** the report.
16. Close Report Builder.

Results: After completing this lab exercise, you will be able to:

Format a report.

Add a chart to a report.

Add a map to a report.

Add databars and sparklines to a report.

Question: What graphical elements do you think work best in reports—and why?

Question: Where do sparklines appear in a report?

Module Review and Takeaways

In this module, you've learned how to format reports, and add different visual elements including maps, charts, indicators, gauges, databars, and sparklines. You've also learned how to create KPIs in the Reporting Services web portal.

You use visuals to make data more meaningful and faster to understand.



Best Practice: Visuals should tell a story, so choose appropriate visuals for the data, and the available space on a report.

Review Question(s)

Question: In your organization, what reports might include visuals? How could they be improved?

Module 6

Summarizing Report Data

Contents:

Module Overview	6-1
Lesson 1: Sorting and grouping	6-2
Lesson 2: Report subreports	6-13
Lesson 3: Drilldown and drillthrough	6-20
Lab: Summarizing report data	6-27
Module Review and Takeaways	6-32

Module Overview

As the amount of data we need to deal with increases, so does the requirement to manage data by grouping and summarizing. In this module, you will learn how to create group structures, summarize data, and provide interactivity in your reports, so that users see the level of detail or summary that they need.

Objectives

After completing this module, you will be able to:

- Group and sort data in reports.
- Create report subsections.
- Use drilldown and drillthrough.

Lesson 1

Sorting and grouping

All but the simplest datasets benefit from grouping and sorting. Data is easier to understand when it's presented with similar information together, and sorted appropriately. Most reports benefit from grouping at different levels, and aggregating data for each grouping level.

Grouping and sorting also ensures that specific information is found quickly.

Lesson Objectives

After completing this lesson, you will be able to:

- Sort data in a report.
- Implement group expressions.
- Describe the details groups.
- Explain recursive hierarchy groups.
- Add a total for a tablix data region.
- Add a total for a group in a tablix data region.
- Add aggregated fields by using expressions.
- Configure page breaks for groupings.
- Implement drilldown and drillthrough actions.

Sorting data

Sorting data means you present it in a specific sequence, depending on the data type of the column. Character values are sorted from A to Z or Z to A. You sort numeric from highest to lowest or lowest to highest. You sort date and time values by oldest to most recent, or most recent to oldest.

You sort data either at source or in the report. For large datasets, it's more efficient to sort data at source, because you take advantage of indexes. However, you might not always have control over how data is presented so you would need to sort data within the report. For small datasets, it's less important to decide whether data is sorted at source or in the report.

- Sort the source data
 - Most efficient for sorting large volumes of data
 - In SQL Server, use the ORDER BY clause
- Sort in the report
 - A sort expression can be applied to a tablix data region, group, or chart
- Interactive sorting
 - Most users want to interact with the data
 - Set columns to have interactive sorting

Data sorted at source

If the data source is SQL Server, you use an ORDER BY clause to sort the data—either in ascending or descending order. The default sort order is ascending. The following SELECT statement is an example of how to return sorted data:

Sorting data from SQL Server.

Sort data using the ORDER BY clause

```
SELECT DimGeography.EnglishCountryRegionName AS [Country-Region],
       DimGeography.StateProvinceName AS State,
       DimGeography.City,
       DimReseller.ResellerName AS Reseller,
       FactResellerSales.SalesOrderNumber,
       FactResellerSales.OrderDate, FactResellerSales.SalesAmount
FROM   DimGeography
       INNER JOIN DimReseller ON DimGeography.GeographyKey = DimReseller.GeographyKey
       INNER JOIN FactResellerSales ON DimReseller.ResellerKey =
FactResellerSales.ResellerKey
ORDER BY [Country-Region], SalesAmount DESC
```

For more information about the ORDER BY clause, see Microsoft Docs:



SELECT - ORDER BY Clause (Transact-SQL)

<https://aka.ms/Ss4wfb>

Sorting on a data region or group

The ability to sort is available in a number of places within a SQL Server Reporting Services (SSRS) report. Both Report Designer and Report Builder enable sorting in the following places:

- Tablix data region.
- Tablix data region group.
- Chart data region.

A typical sort expression consists of a column name, and the sort order.

You can also sort on a value that isn't contained in the source data. The sorting options on a tablix data region offer the possibility of using a complex expression as the sort order. For example:

=Lookup(Fields!ProductID.Value, Fields!ID.Value, Fields!Name.Value, "Product").

Example of a sort expression.

A sort expression

```
=Lookup(Fields!ProductID.Value, Fields!ID.Value, Fields!Name.Value, "Product")
```

For more information about using simple and complex expressions in reports, see Microsoft Docs:



Expression Uses in Reports (Report Builder and SSRS)

<http://aka.ms/Gsrlnt>

Interactive sorting

As users view reports, they are likely to want to change the sort order. Any values contained in a table can be set to be sorted by users—a user clicks and sorts the data in that column.

Group expressions

Groups are used to organize data. Groups help to make reports more understandable by breaking the data into manageable chunks, and enabling calculations on subsets of the data.

SSRS uses group expressions to create groups—these expressions are created automatically as you build a report layout. Examples of when they are created automatically include:

- Arranging datasets in tables or charts.
- Adding fields to a row of column groups in the Grouping pane.
- Adding fields to category or series groups on a chart.

- Group expressions can be created automatically
 - By dragging fields into the row or columns groups pane
 - By adding fields to a category or series group on a chart
- Group expressions can be created manually
 - In Report Designer or Report Builder
- Example of an expression
 - =Fields!Postcode.Value

You can also create group expressions manually by specifying a grouping criteria. It's possible to change a group expression after it has been created, whether it was originally created automatically or manually.

Specifying groups

You group data by a single field, multiple fields or parameters, conditional statements, or even custom code. When it's been created, a group will be given a name—this is then used to specify the scope of aggregate calculations, or to define the parent and child nodes in a drilldown report. When a report has groups defined within it, these groups are used to add page breaks.

Examples of group expressions

Group expression	Description
=Fields!Product.Value	A group can be defined on a single field. In this expression, it's defined on the product field.
=Fields!Postcode.Value.Substring(0,4)	The expression might be more complex than simply using the value in a field. Here, the group is based on the first four characters of a postal code.
=IIF(First(Fields!Age.Value)<18,"Under 18",(IIF(First(Fields!Age.Value)>=18 AND First(Fields!Age.Value)<=40,"Between 18 and 40","Over 40")))	Complex logical expressions are also used. In this example, both aggregate functions (First) and VB condition (IIF) are used to create grouping that will be "Under 18", "Between 18 and 40", and "Over 40".

Recursive hierarchy groups

A recursive hierarchy group uses data from a dataset that has multiple hierarchical levels. Examples of this kind of data include a report that shows an organization structure with employee and manager relationships, or a product report that contains product categories and subcategories.

Before you create a table as a recursive hierarchy group, you must have a dataset that contains recursive hierarchical relationships. In the organization chart example, the dataset would have to contain at least the employee's name, an employee ID, and a manager ID.

Level	Name	EmployeeID	ManagerID
0	David Bradley	23	
1	Kevin Brown	76	23
1	Mary Gibson	54	23
2	Jill Williams	105	54
3	John Wood	276	105
1	Rob Walters	88	23

Using recursive hierarchy groups

Using the organization report example, the report might contain a table that's associated with the previously described dataset. As a minimum, there would be a details row group, grouped by employee ID. The recursive parent would then be set to the manager ID.

A common way to display the hierarchical relationship in a table is to indent a field by the level where it's in the hierarchy. A text box uses an expression to define the padding within a report.

Here is an example of how a padding expression might be used:

In this padding expression, the left padding for each level is indented by 22 points.

Text box padding expression

```
=CStr(2 + (Level() * 20)) + "pt"
```


Adding a document map

SSRS automatically generates a document map for hierarchical data. When specifying the recursive parent, a report writer might also specify a field, or expression, that will be used for the text in the document map. In the previous example, the employee's name would be a good candidate for this text.

Details group

The details group displays the detailed data in a report. The details group is the only group that has no group expression.

The details group contains a row for every row in the dataset, and all the fields in the dataset. Use a details group in a paginated report to show all the data from the dataset.



- The details group displays the detailed data, with no aggregation
- It's the innermost group


When you insert a table or list into a paginated report, Report Builder and Report Designer automatically create a details group, and add a row to display the detail data. When the report is viewed, the details row repeats once for every value in the result set.

You also add a details group to an existing report by selecting the option called **show detail data**. This creates a details group.

Use Report Builder to add a details group to a tablix data region:

1. **Click** the tablix data region to select it. The grouping pane is displayed.
2. **Right-click** the inner child group.
3. Click **Add Group**, and then click **Child Group**. The **Tablix Group** dialog box is displayed.
4. Leave the Group expression blank.
5. Select **Show detail data**.
6. Click **OK**.

For more information about details groups, see Microsoft Docs:

 **Add a Details Group (Report Builder and SSRS)**

<https://aka.ms/Fkjl3a>

Totals on a tablix data region

One of the benefits of grouping data is that you can show aggregated values for each group. For example, a report that shows sales grouped by product category, subcategory, and individual products might show sales totals for each of those groups.

To add totals to a group:

1. Right-click a group. The Groupings pane is displayed.
2. Click **Add Total**, and then click **Before** or **After**, to indicate where the total should appear.
3. A row (or column) that contains a sum function is added for each numeric field.

Aggregations can be totals, averages, or other summaries.

Adding a sum expression.

A sum expression

```
=Sum(Fields!SalesAmount.Value)
```

You can specify the scope of an aggregation.

- Groups enable aggregations for each group
- A row or column is added for the aggregation
- Specify scope to aggregate a containing group

You show comparisons by using a total from another group level.

Specify the scope of the aggregation

```
=Sum(Fields!SalesAmount.Value) & " of " &  
Sum(Fields!SalesAmount.Value, "Product_Category")
```

Adding aggregated fields by using expressions

Summarizing numeric values include sum, average, minimum, maximum, count, or a more complex function.

You use aggregate functions for the same group and level, or to specify the scope. To add an aggregate function, add a row or column to the group and enter an expression that includes the aggregation function.

By default, the function aggregates data at the same level as the group within which it's used. For example, you might use the following expression in a Product group to show the total sales for each product:

Add an expression that includes the aggregation function.

Expression to total the Sales Amount

```
=Sum(Fields!SalesAmount.Value)
```

You can, however, use an aggregation function that has a named scope to display a summarized value at another group level. For example, the following expression in a Subcategory group shows the total sales for each subcategory—and how that compares with the total sales for the Product_Category parent group:

You specify an aggregate value from another level.

Comparison of Product.Category total to Subcategory total

```
=Sum(Fields!SalesAmount.Value) & " of " & Sum(Fields!SalesAmount.Value,  
"Product_Category")
```

- You specify an aggregate function:
 - For the same group and level
 - For a named scope
 - Enables comparisons

Configuring page breaks for groupings

You view reports online in a browser, in print form, or when exported to another format. Reporting Services provides ways to control how reports are divided into pages when they are printed or rendered in a format that supports pagination. Pagination is important when working with groups; for example, when you start a new page for each instance of a group.

Ways in which you control pagination include:

- Define page breaks in tablix data regions, groups, and rectangles. You define explicit behaviors for page breaks by setting the **BreakLocation**, **ResetPageNumber**, and **PageName** properties in the **PageBreak** category.
- The **BreakLocation** property specifies where the page break should occur. For data regions and rectangles, this might be before, after—or before and after—the report element on which the page break is defined. For groups, you might also specify that the page break should occur between instances. For example, you can force a new page for each employee in a report that shows sales grouped by employee.
- The **ResetPageNumber** property causes page numbering to be restarted on the new page that the page break generates.
- The **PageName** property specifies a name for the new page that the page break generates. Some renderers, such as Microsoft Excel®, use page names to identify pages. You might set an explicit value for this property, or use an expression to generate the value dynamically. This might be used, for example, to specify the name of the employee in the grouped sales report that's described in the previous topic.
- You can also set the **Disabled** property to disable a page break.
- Set the **InitialPageName** property of the report. This property defines the page name for the first page in the report (or for all pages, if it doesn't contain any explicitly defined page breaks).

- Create page break location for groups:
 - Start, End, Between, or Start and End
- Use expressions to set the PageName property of page break dynamically
 - InitialPageName report property sets default page name
 - Page names determine worksheet names when rendering to Excel

Year	Country
2005	Canada
2006	Canada
2007	Canada
2008	Canada
2005	France
2006	France
2007	France
2008	France
2005	Germany
2006	Germany
2007	Germany
2008	Germany
2005	United Kingdom
2006	United Kingdom
2007	United Kingdom
2008	United Kingdom
2005	United States
2006	United States
2007	United States
2008	United States

Pagination and Microsoft Excel

The ability to control pagination is particularly useful when you export reports to Excel. By default, the Excel rendering extension creates a workbook with a single worksheet that contains the report data. By default, the worksheet name, which is shown in the tabs at the bottom of the workbook, is based on the name of the report. However, if you specify an **InitialPageName** property value, that's used for the worksheet name.

In many cases, you might want to render the report data on multiple worksheets in Excel. One reason for doing this is to make it easier for users to navigate the report data; for example, by creating a new worksheet (and corresponding tab) for each employee grouping in the sales report that was described in the previous topic. Another reason is that Excel enforces a maximum number of rows for each worksheet. A very large report might not render successfully if it contains more rows than are available in a worksheet.

Setting a page break in a report causes the Excel rendering extension to generate a new worksheet. If the page break includes a PageName property value, the worksheet is named accordingly. This enables you to design multiple-worksheet, Excel-based reports that are easy to navigate.

Demonstration: Sorting and grouping in Report Builder

In this demonstration, you will see how to use Report Designer to:

- Add groups to a table.
- Change the sort order.

Demonstration Steps

Grouping on subcategory

1. Click the **Start** menu, and type **Report Builder** and then click to open it.
2. Click **Open** and navigate to **D:\Demofiles\Mod06\SortGroup\SortGroup** and double click **SortGroup.rdl** to open it.
3. Click **Insert, Table**, and then **Insert Table**. Click and drag a table onto the report designer and position it top left.
4. In the second row of the left column, click the **button** and select **Year**.
5. In the second row of the middle column, click the **button** and select **SalesValue**. Note that the orange scope indicator appears when the text box for **[SalesValue]** is selected. Also point out that the three-bar detail indicator appears on the row selector.
6. Right-click **[SalesValue]** and select **Text Box Properties**.
7. Click **Number** and the **Number** and click twice to reduce to **0 decimal places** and click **Use 1000 Separator**.
8. Click **OK**.
9. Highlight the rightmost column, right-click, and then click **Delete Columns** (leaving two columns)
10. Click **Run** view the report.
11. Click **Design** and then select the **[SalesValue]** text box.
12. Right-click the outer grey row selector button (far left column containing the three-bar indicator), and then click **Add Group, Row Group** and then **Parent Group**.
13. In the **Tablix group** dialog box, in **Group by**, select **[EnglishProductSubcategoryName]** from the drop-down list.
14. Select both **Add group header** and **Add group footer** and then click **OK**.
15. Right-click the **[SalesValue]** text box, and then click **Add Total**.
16. Point out to students:
 - a. The additional rows and column added to the table.
 - b. The orange scope indicator includes the three rows. The Sum of SalesValue will include the details rows within each Subcategory group.
 - c. The graphic on the row selectors shows an outer group (the left bracket) and a details row (the three bar icon).
 - d. There is a vertical double dotted line to the right of the Subcategory column. This indicates where the table body starts.
17. Widen the **English Product Subcategory Name** column and amend the heading to **Subcategory**.

18. Right-click the text box **[Sum(SalesValue)]** and click **Expression**. Note the expression, and then click **OK**.
19. Right-click the text box **[Sum(SalesValue)]** again, and select **Text Box Properties**.
20. In **Number** category, ensure **decimal places** is set to **0**, and **Use 1000 Separator** is enabled.
21. Click **Font** and select **Bold** and then click **OK**. Point out that the Row Groups pane now has a group row and a details row.
22. Click **Run** to view the report and then click **Design**.

Add a category group

1. In Report Data, expand **Datasets**, and **DataSet1**. Drag **EnglishProductCategoryName** into the Row Groups pane so that it is inserted above **EnglishProductSubcategoryName**. Point out that a new column appears in the table.
2. Widen the **Category** column.
3. In the Row Groups pane, right-click **[EnglishProductSubcategoryName]**, point to **Add Total**, and then click **After**.
4. In the bottom right cell containing **[Sum(SalesValue)]**, right-click and select **Text Box Properties**.
5. In **Number** category, ensure **decimal places** is set to **0**, and **Use 1000 Separator** is enabled..
6. Right-click the bottom cell in the Subcategory column containing Total, and click **Expression**.
7. In Set expression for: Value, type:


```
="Total for " + Fields!EnglishProductCategoryName.Value
```
8. Click **OK**.
9. Delete the **[Sum(Year)]** text.
10. Click **Run** to view the report.
11. Scroll down to view the Bikes category, note the **Total for Accessories** is **700,760**.
12. Close the Report Builder without saving.

Demonstration: Sorting and grouping in Report Designer

In this demonstration, you will see how to use Report Designer to:

- Add groups to a table.
- Change the sort order.

Demonstration Steps

Grouping on subcategory

1. From the taskbar, click **Visual Studio 2015** to open it.
2. Click the **File** menu, click **Open**, click **Project/Solution**, then navigate to **D:\Demofiles\Mod06\SortGroup**.
3. Click **SortGroup.sln**, then click to **Open**. If any security warning dialog box appears, click **OK**.

4. In the Solution Explorer, double click the report **SortGroup.rdl** to open it in Design mode. Note that the report is empty.
5. If the Report Data pane is not displayed, click **View, Report Data** (or press the **CTRL-ALT-D** keys).
6. From the **Toolbox**, drag a **Table** onto the report designer and position it top left.
7. In the second row of the left column, click the **button** and select **Year**.
8. In the second row of the middle column, click the **button** and select **SalesValue**. Note that the orange scope indicator appears when the text box for **[SalesValue]** is selected. Also point out that the three-bar detail indicator appears on the row selector.
9. Right-click **[SalesValue]** and select **Text Box Properties**.
10. Click **Number** and then **Number**, and reduce to **0 decimal places** and click **Use 1000 Separator**.
11. Click **OK**.
12. Highlight the rightmost column, right-click, and then select **Delete Columns**.
13. Click **Preview** to view the report.
14. Click **Design** and then select the **[SalesValue]** text box.
15. Right-click the outer grey row selector button (far left column containing the three-bar indicator), and then click **Add Group, Row Group** and then **Parent Group**. The Tablix group dialog box is displayed.
16. In Group by, select **[EnglishProductSubcategoryName]** from the drop-down list.
17. Select both **Add group header** and **Add group footer** and then click **OK**.
18. Right-click **[SalesValue]**, then click **Add Total**.
19. Point out to students:
 - a. The additional rows and column added to the table.
 - b. The orange scope indicator includes the three rows. The Sum of SalesValue will include the details rows within each Subcategory group.
 - c. The graphic on the row selectors shows an outer group (the left bracket) and a details row (the three bar icon).
 - d. There is a vertical double dotted line to the right of the Subcategory column. This indicates where the table body starts.
20. Widen the **English Product Subcategory Name** column and amend the heading to **Subcategory**.
21. Right-click the text box **[Sum(SalesValue)]** and click **Expression**. Note the expression, and then Click **OK**.
22. Right-click the text box **[Sum(SalesValue)]** again, and select **Text Box Properties**.
23. In **Number** category, ensure **decimal places** is set to **0**, and **Use 1000 Separator** is enabled.
24. Click **Font** and select **Bold** and then click **OK**. Point out that the Row Groups pane now has a group row and a details row.
25. Click **Preview** to view the report and then click **Design**.

Add a category group

1. In Report Data, expand **Datasets**, and **DataSet1**. Drag **EnglishProductCategoryName** into the Row Groups pane so that it is inserted above **EnglishProductSubcategoryName**. Point out that a new column appears in the table.
2. Widen the **English Product Category Name** column.
3. In the Row Groups pane, right-click [**EnglishProductSubcategoryName**], point to **Add Total**, and then click **After**.
4. In the bottom right cell, right-click and select **Text Box Properties**.
5. In **Number** category, ensure **decimal places** is set to **0**, and **Use 1000 Separator** is enabled.
6. Right-click the bottom cell in the Subcategory column containing the word Total, and click **Expression**.
7. In Set expression for: Value, type:

```
= "Total for " + Fields!EnglishProductCategoryName.Value
```
8. Click **OK**.
9. Delete the **[Sum(Year)]** text.
10. Click **Preview** to view the report.
11. Scroll down to view the Bikes category, note the **Total for Accessories** is **700,760**.
12. Close Visual Studio 2015 without saving your changes.

Review sorting and grouping

Sorting and grouping allows users to see data in a certain way. Your choice of groups and sort order is a major influence on how the data is perceived. For example, grouping by product category and sorting by descending sales value will highlight the best-selling products for the period. However, the same data grouped by month, and sorted by product name, would highlight the variance in monthly sales. The data doesn't change, but the user's understanding of the data is influenced by how the data is presented.

Lesson 2

Report subreports

This lesson discusses subreports, including what they are and when to use them.

Lesson Objectives

After completing this lesson, you will be able to:

- Explain what a subreport is.
- Know when to use a subreport.

Understanding subreports

A subreport is a report within a report—it's a separate report, normally stored in the same folder as the parent report, and always on the same reporting server. The subreport is embedded within the parent report. You can pass parameters, allowing subreports to contain relevant data. You can also embed more than one subreport within a report.

You can think of the parent report as a container for other reports. Although the user is not aware that they are viewing a subreport, a second report is being displayed. Subreports are invoked for every row in the report, and so have a performance overhead. To improve performance, consider using a drillthrough report—this is discussed in the following lesson.

- A subreport is a report within a report
- It's a separate report
- Normally in the same folder as the parent report
- You pass parameters to the subreport

For more information about subreports, see Microsoft Docs:

 **Subreports (Report Builder and SSRS)**

<https://aka.ms/Ewps68>

For more information about passing parameters to a subreport, see Microsoft Docs:

 **Add a Subreport and Parameters (Report Builder and SSRS)**

<https://aka.ms/Cct244>

Subreports vs. data regions

When you decide whether to use a subreport or a data region within a report, there are a number of considerations:

- A data region, such as a table, matrix or chart, might perform better than a subreport.
- When you want to combine data from different data sources, subreports might be a better option.
- If you want to reuse the same report layout in a number of different parent reports, subreports might be the better option.

- Use a subreport:
 - When you want to combine data from more than one source
 - When you want to use the same report layout in more than one parent report
- Use a data region:
 - When performance is important—data regions will perform better
 - When data is from the same data source

The main consideration to be aware of is that a subreport is a separate report—with all the overheads that entails. If you want performance, consider if you need a subreport is necessary. You can achieve the same result with data regions.

For more information about using data regions in place of subreports, see Microsoft Docs:



Nested Data Regions (Report Builder and SSRS)

<https://aka.ms/Pf6jae>

Demonstration: Creating a subreport in Report Designer

In this demonstration, you will see how to create a subreport using Report Designer.

Demonstration Steps

View the main report

1. From the taskbar, click **Visual Studio 2015** to open it.
2. Click the **File** menu, click **Open**, click **Project/Solution**, then navigate to **D:\Demofiles\Mod06\SubReport**.
3. Click **Subreport.sln**, then click to **Open**.
4. In the Solution Explorer, double-click **SalesReasons.rdl** to open it in **design** mode.
5. Click **Preview** to view the report. Note the missing data in the **Sales Reason** column.
6. Click **Design**.

Create the subreport

1. In Solution Explorer, right-click **Reports**, click **Add** and then click **New Item**.
2. In the Add New Item – Subreport dialog box, select **Report**, and change the name to **ReasonsSubreport.rdl**. Click **Add**.
3. Under **Report Data**, right-click **Data Sources**, and click **Add Data Source...**
4. In Name, type **AdventureWorksDW**.
5. With Embedded connection selected, click **Edit**.

6. In Server name, type **MIA-SQL**.
 7. In Select or enter a database name: select **AdventureWorksDW** from the drop-down list.
 8. Click **Test Connection** and then click **OK**.
 9. Click **OK** and then click **OK** to create the data source.
 10. Right-click **Datasets**, and click **Add Dataset...**
 11. Click **Use a dataset embedded in my report**.
 12. In Data Source, select **AdventureWorksDW** from the drop-down list.
 13. Under Query, click **Import...**
 14. Navigate to **D:\Demofiles\Mod06\Subreport** and click **SubReport.sql** and click **Open**.
 15. Click **OK**.
 16. From the **Toolbox**, drag a **Table** to the top left corner of the report design area.
 17. In the data row of the first column, click the **button** and select **[SalesReasonName]**.
 18. Widen the first column.
 19. In the data row of the second column, click the **button** and select **[Occurrences]**.
 20. Right-click the **[Occurrences]** cell and click **Text Box Properties**.
 21. In the left-pane, click **Number** and under Category, click **Number**. Click twice to reduce to **0 decimal places**, and click **Use 1000 Separator**. Click **OK**.
 22. Right-click the rightmost column heading and click **Delete Columns**.
 23. Select the row containing the column headings and from the **Properties** menu, expand and click **Font** and then click drop down at **FontWeight**, click **Bold**.
 24. **Resize** the report area so that it exactly fits around the table.
 25. Click **Preview** to view the report.
 26. Click **Design**.
- Add a parameter to the subreport
1. In the Report Data pane, right-click **Parameters**.
 2. Click **Add Parameter**. The Report Parameter Properties dialog box is displayed.
 3. In **Name**, type **CategoryKey**.
 4. In **Prompt**, type **Category**.
 5. In **Data Type**, select **Integer**.
 6. In **Select parameter visibility**, click **Hidden**.
 7. In the left pane, click **Default Values**.
 8. Click **Specify Values**.
 9. Click **Add**, and in Value highlight **(Null)** and overwrite **1**.
 10. Click **OK**.

11. In the Report Data pane, right-click **Dataset1** and click **Dataset Properties**. The Dataset Properties dialog box is displayed. Under **Query**, amend the query by adding the following **WHERE** clause above the **GROUP BY** clause, and then press **Enter**:

```
WHERE SC.ProductCategoryKey = @CategoryKey
```

12. In the left pane, click **Parameters**. In Parameter Value, select **[@CategoryKey]** and click **OK**.
13. Click **File, Save All**, and then click **X** to close the report.

Embed the subreport

1. With **SalesReasons.rdl** open in design mode, from the **Toolbox** drag **Subreport** and drop it into the bottom right cell. It appears in the cell as <Subreport>.
2. Right-click <Subreport> and click **Subreport Properties**. The Subreport Properties dialog box is displayed.
3. In Name, type **Reasons**.
4. In Use this report as a subreport, select **ReasonsSubreport**.
5. In the left pane, click **Parameters**.
6. Click **Add**.
7. In Name, select **CategoryKey**.
8. In Value, select **[ProductCategoryKey]**.
9. Click **OK**.
10. Widen the column and in the column heading type **Reasons**. From the **Properties** menu, click **Center**.
11. Click **Preview** to view the report with the embedded subreport.
12. Close the solution without saving.

Demonstration: Creating a subreport in Report Builder

In this demonstration, you will see how to create a subreport using Report Builder.

Demonstration Steps

View the main report

1. Click the **Start** menu and type **Report Builder**. Click to open it.
2. Click **Open** and navigate to **D:\Demofiles\Mod06\SubReport\ Subreport** then double-click **SalesReasonsBuilder.rdl** to open it in design mode.
3. Click **Run** to view the report. Note the missing data in the Sales Reason column.
4. Click **Design**.

Create the subreport

1. Click **File, New, New Report, Blank Report**. A blank report is displayed.
2. Click **File, Save As**.
3. In the Save As Report dialog box, click **Recent Sites and Servers**.

4. In Name type **http://mia-sql/ReportServer_SQL2**, and then in press **enter**.
5. In Name type **ReasonsSubreport.rdl**, then click **Save**.
6. Right-click **Data Sources**, and click **Add Data Source...**
7. In Name, type **AdventureWorksDW**.
8. Select **Use a connection embedded in my report**, and then click **Build**.
9. In the Connection Properties window, in Server name, type **MIA-SQL**.
10. In Select or enter a database name: select **AdventureWorksDW** from the drop-down list.
11. Click **Test Connection** and then click **OK**.
12. Click **OK** and then click **OK** to create the data source.
13. Right-click **Datasets**, and click **Add Dataset**.
14. Click **Use a dataset embedded in my report**.
15. In Data Source, select **AdventureWorksDW** from the drop-down list.
16. Under Query, click **Import**.
17. Navigate to **D:\Demofiles\Mod06\Subreport** and click **SubReport.sql** and click **Open**.
18. Click **OK**.
19. From the menu, click **Insert, Table, Insert Table**. Click and drag a **Table** to the top left corner of the report design area.
20. In the data row of the first column, click the **button** and select **[SalesReasonName]**.
21. **Widen** the first column.
22. In the data row of the second column, click the **button** and select **[Occurrences]**.
23. Right-click the **[Occurrences]** cell and click **Text Box Properties**.
24. In the left-pane, click **Number** and under Category, click **Number**. Click twice to reduce to **0 decimal places**, and click **Use 1000 Separator**. Click **OK**.
25. Right-click the rightmost column and click **Delete Columns**.
26. Select the row containing the column headings and from the ribbon menu, click **Bold**.
27. Resize the report area so that it exactly fits around the table.
28. Click **Run** to view the report.
29. Click **Design**.

Add a parameter to the subreport

1. In the Report Data pane, right-click **Parameters**.
2. Click **Add Parameter ...** The Report Parameter Properties dialog box is displayed.
3. In Name, type **CategoryKey**.
4. In Prompt, type **Category**.
5. In Data Type, select **Integer**.
6. In Select parameter visibility, click **Hidden**.
7. In the left pane, click **Default Values**.

8. Click **Specify Values**.
9. Click **Add**, and in Value highlight **(Null)** and overwrite **1**.
10. Click **OK**.
11. In the Data pane, right-click **Dataset1**, and then click **Dataset Properties**. The Dataset Properties dialog box is displayed. Under Query, amend the query by adding the following **WHERE** clause above the **GROUP BY** clause, and then press **Enter**:

```
WHERE SC.ProductCategoryKey = @CategoryKey
```

12. In the left pane, click **Parameters**. In Parameter Value, select **[@CategoryKey]** and click **OK**.
13. Click **File**, and then **Save**.

Embed the subreport

1. Click **File**, then **Open**.
2. In the Open Report, in Name, type **D:\Demofiles\Mod06\SubReport\Subreport\SalesReasonsBuilder.rdl**, then click **Open**
3. From the ribbon menu click **Insert**, and then double-click **Subreport**. A subreport placeholder is displayed. Resize and move it to the bottom right cell. It appears in the cell as **<Subreport>**.
4. Right-click **<Subreport>** and click **Subreport Properties**. The Subreport Properties dialog box is displayed.
5. In Name, type **Reasons**.
6. In Use this report as a subreport, click **Browse**.
7. Select **ReasonsSubreport**, and then click **Open**.
8. In the left pane, click **Parameters**.
9. Click **Add**.
10. In Name, select **CategoryKey**.
11. In Value, select **[ProductCategoryKey]**.
12. Click **OK**.
13. Widen the column and in the column heading type **Reasons**. From the ribbon menu, click **Center**.
14. Click **Run** to view the report with the embedded subreport.
15. Close Report Builder without saving.

Check Your Knowledge

Question	
<p>One of your reports has a large amount of detail. To allow the report to render quickly, you decide to hide the detail until users want to see it. How should you implement this?</p>	
Select the correct answer.	
<input type="checkbox"/>	Use a subreport
<input type="checkbox"/>	Use a drillthrough action
<input type="checkbox"/>	Create separate reports
<input type="checkbox"/>	Create a report with less detailed data

Lesson 3

Drilldown and drillthrough

Well-designed reports effectively convey summary information, and then allow users to get more detail about the areas of specific interest. Drilldown and drillthrough are two ways of hiding the detail, so that the user drills into the detail as needed.

Lesson Objectives

After completing this lesson, you will be able to:

- Implement drilldown and drillthrough.

What is drilldown?

When a user clicks to reveal more detailed information in a report, this is referred to as drilldown.

Users often want to view summary information, and only explore details for specific groupings when they see an issue or an anomaly. Reporting Services supports this drilldown functionality—you hide or show groups in a report by clicking a toggle button that's associated with a field in the parent group.

To add drilldown functionality in a report:

- In the Report Wizard, select **Enable Drilldown** in the **Choose the Table Layout** page for a tabular report that contains groups.

In an existing report that contains groups, right-click a child group in the Groupings pane and click **Group Properties**. In the **Group Properties** dialog box, on the **Visibility** tab, select **Hide**, select **Display can be toggled by this report item**, and select the field in the parent group that you want viewers to use, to show or hide the group.

Using drilldown

The drilldown functionality is only available when you view a published report in a browser or export it to a format such as Excel, which supports this kind of interactivity.

When you export to a format that does not support interactivity, the data is rendered statically. To overcome this problem, you create a version of the same report for each target format; for example, an interactive report that includes a hyperlink to a static version for printing. This might entail a lot of additional work to develop and manage multiple versions of each report.

An alternative is to design adaptive reports that modify their behavior, depending on the rendering extension being used. To help you accomplish this, Reporting Services supports the following global variables:

- **RenderFormat.IsInteractive**. You use this variable to determine if the render format supports interactivity, such as drilldown functionality to show hidden groups.
- **RenderFormat.Name**. You use this variable to determine the specific rendering extension being used and apply format-specific settings.

- Use drilldown to hide a child group
- Display detail by clicking a report item

Product Category	Product Subcategory	Product	Sales
Accessories			\$799,887.42
Bikes			\$85,148,124.39
	Mountain Bikes		\$32,015,700.48
	Road Bikes		\$35,601,815.12
	Touring Bikes		\$17,530,608.79
Clothing			\$2,328,221.96
Components			\$14,726,589.17

For example, the following expression might be used to set the **Hidden** property of a group in a tablix data region. The group would then be hidden when rendered to an interactive format, but would be visible in formats that do not support interactivity:

Determining whether the render format is interactive.

RenderFormat.IsInteractive

```
=iif(Globals!RenderFormat.IsInteractive, True, False)
```

For more information about RenderFormat, see Microsoft Docs:



Built-in Collections - Built-in Globals and Users References (Report Builder)

<https://aka.ms/Lwfywq>

For more information about drilldown, see Microsoft Docs:



Drillthrough, Drilldown, Subreports, and Nested Data Regions

<https://aka.ms/Tkx02o>

What is drillthrough?

Drillthrough refers to the action when a user clicks a link in a summary report to find more detailed information. When a user clicks, the report that contains the detail is displayed. The detailed report might use the same dataset as the summary report—or a different dataset.

Drillthrough actions

There are three different actions that you can add:

- Go to a report.
- Go to a bookmark.
- Go to a URL.

- Drillthrough adds an action to:
 - A URL
 - A bookmark
 - Another report containing detailed data
- Add actions to images, text boxes, and chart data points
- Specify the action in the properties of the item
- Add parameters to the action

If the drillthrough action is to another report, that report must already exist and be on the same report server. However, it might be located in a different folder to the summary report. Actions can be added to many different items including images, text boxes, and data points on a chart. The item must, however, have an Action property.

To add a drillthrough action

Using Report Builder, you add a drillthrough action as follows:

1. Select the relevant item—for example, a text box.
2. In **Properties**, under **Action**, click the drop-down box next to **Action**.
3. Select **Go to report**.
4. In **Specify a report**, either type the name of the report, or browse to the report location.
5. To remove an action, select **None**.

Drillthrough report parameters

When you drill through from the summary report to the detailed report, you might want to show specific detailed information, rather than generic detailed data. You provide specific information by passing one or more parameters from the summary report to the detailed report.

For example, a summary report that shows exam results for schools in a certain area might drill through to a summary report that shows exam results by subject for a specific school. In this case, a parameter that identifies the school would be passed to the detailed report.

To display a parameterized subreport by using the drillthrough action:

1. Create a report parameter in the subreport.
2. Amend the dataset query to return only rows that match the parameter.

For more information about report parameters, see Microsoft Docs:



Report Parameters (Report Builder and Report Designer)

<https://aka.ms/Xgv5xo>

For more information about drillthrough, see Microsoft Docs:



Drillthrough Reports (Report Builder and SSRS)

<https://aka.ms/Nvrsrt>

Demonstration: Drillthrough in Report Designer

In this demonstration, you will see how to add drillthrough actions to an existing summary report. This demonstration uses Report Designer.

Demonstration Steps

View the summary report

1. From the taskbar, open **Visual Studio 2015**.
2. Click the File menu, click **Open**, click **Project/Solution**, then navigate to **D:\Demofiles\Mod06\Drillthrough**.
3. Click **Drillthrough.sln**, and then click **Open**.
4. In the Solution Explorer pane, double-click **Summary.rdl** to open it in design mode.
5. Click **Preview** to view the report. Note that users might want to see more detail about the products sold in each category.

Create the detail report

1. In Solution Explorer, right-click **Reports**, click **Add** and click **New Item**.
2. Select **Report**, and in Name type **ProductDetail.rdl**, then click **Add**.
3. From the **Toolbox**, drag a **text box** to the top left of the report. Type **Products** and from the **Properties** menu select **Font > FontSize** and type **16pt** and then press **Enter**.
4. From the **Toolbox**, drag a **table** directly below the text box. The Dataset Properties dialog box is displayed.

5. In Query, select **Use a dataset embedded in my report**.
6. In Data source, click **New**.
7. In the Data Source Properties dialog box, in Name type **AdventureWorksDW**.
8. Under Connection string, click **Edit**.
9. In the Connection Properties dialog box, in Server name, type **MIA-SQL**.
10. In Connect to a database, click **Select or enter a database name**, then select **AdventureWorksDW**.
11. Click **OK**, then click **OK**.
12. Click **Import**, and from **D:\Demofiles\Mod06\Drillthrough** click **ProductDetail.sql**.
13. Click **Open**, then click **OK**.
14. Right-click the rightmost column, and click **Insert Column** and then click **Right**. Repeat until you have a total of five columns.
15. In the details row of the first column, click the **button** and click **EnglishProductName**.
16. In the details row of the second column, click the **button** and click **Size**.
17. In the details row of the third column, click the **button** and click **Color**.
18. In the details row of the fourth column, click the **button** and click **InternetSales**.
19. In the details row of the fifth column, click the **button** and click **ResellerSales**.
20. In Row Groups, click the **= (Details)** line and point to **Add Total**. Click **After**.
21. On the report, highlight the **totals row** and on the **Properties** menu, click **Bold**.
22. In the Report Data pane, expand the **Parameters** folder.
23. Right-click **[@] Subcategory**, then click **Parameter Properties**.
24. In Data type, select **Integer** from the drop-down list.
25. In Select parameter visibility, click **Hidden**.
26. In the left pane, click **Default Values**.
27. Click **Specify Values**.
28. Click **Add** and overtype **(Null)** with **1**. Click **OK**.
29. Click **File, Sale All**.

Add an action to the main report

1. Click on the **Summary.rdl** tab at the top of the window.
2. If you are in Preview mode, click **Design**.
3. Right-click in **[EnglishProductSubcategory]**, and then click **Text Box Properties**.
4. In the left pane, click **Action**.
5. Click **Go to report**.
6. In the Specify a report field, in the drop-down, select **ProductDetail**.
7. Click **Add**.
8. In the Name drop-down, select **Subcategory**.

9. In the Value drop-down, select **[ProductSubCategoryKey]**.
10. Click **OK**.
11. Click **Preview** to view the report.
12. In the report, click **Bike Racks**, and note that the report drills down to a summary report for accessories.
13. Close the solution without saving any changes.

Demonstration: Drillthrough in Report Builder

In this demonstration, you will see how to add drillthrough actions to an existing report using Report Builder.

Demonstration Steps

View the summary report

1. Click the **Start** menu and type **Report Builder**. Click to open it.
2. In the Getting Started dialog box, click **Open**, then navigate to **D:\Demofiles\Mod06\Drillthrough\Drillthrough** and open **Drillthrough.rdl**.
3. Click **Run** to view the report. Note that users might want to see more detail about the products sold in each category.

Create the detail report

1. Click **File, New, New Report**, and then click **Blank Report**.
2. Click **File, Save As**.
3. In the **Save As Report** dialog box, click **Recent Sites and Servers**.
4. In Name type **http://mia-sql/ReportServer_SQL2**, and then press enter.
5. In Name type **ProductDetail.rdl**, then click **Save**.
6. In the heading, type **Products**.
7. Click **Insert, Table, Insert Table**, and then drag a table beneath the heading against the left border. The Dataset Properties dialog box is displayed.
8. In Query, click **Use a dataset embedded in my report**.
9. In Data source, click **New**.
10. In the Data Source Properties, in Name type **AdventureWorksDW**.
11. Click **Use a connection embedded in my report**.
12. Click **Build**.
13. In the **Connection Properties** dialog box, in Server name type **MIA-SQL**.
14. Under Select or enter a database name, select **AdventureWorksDW**.
15. Click **OK**, and then click **OK** again.
16. Click **Import**, and from **D:\Demofiles\Mod06\Drillthrough** click **ProductDetail.sql**.
17. Click **Open**, and then click **OK**.

18. Right-click the rightmost column and click **Insert Column** and then click **Right**. Repeat until you have a total of five columns. Drag the right side of the report to increase the width to contain the table.
19. In the details row of the first column, click the **button** and click **EnglishProductName**.
20. In the details row of the second column, click the **button** and click **Size**.
21. In the details row of the third column, click the **button** and click **Color**.
22. In the details row of the fourth column, click the **button** and click **InternetSales**.
23. In the details row of the fifth column, click the **button** and click **ResellerSales**.
24. In Row Groups pane, right-click the **= (Details)** line and point to **Add Total**. Click **After**.
25. On the report, highlight the **totals row**, and on the ribbon menu, click **Bold**.
26. In the Report Data pane, expand the **Parameters** folder.
27. Right-click **[@] Subcategory**, then click **Parameter Properties**.
28. In Data type, select **Integer** from the drop-down list.
29. In Select parameter visibility, click **Hidden**.
30. In the left pane, click **Default Values**.
31. Click **Specify Values**.
32. Click **Add** and overtype **(Null)** with **1**. Click **OK**.
33. Click **File, Save**.

Add an action to the main report

1. Click **File**, then **Open**.
2. In the Open Report dialog box, navigate to D:\Demofiles\Mod06\Drillthrough\Drillthrough.
3. Click **Drillthrough.rdl**, then click **Open**.
4. Right-click in **[EnglishProductSubcategory]**, and then click **Text Box Properties**.
5. In the left pane, click **Action**.
6. Click **Go to report**.
7. In the Specify a report field, click **Browse**.
8. In the Select Report dialog box, click **ProductDetail**, then click **Open**.
9. Under Use these parameters to run the report, click **Add**.
10. In the Name drop-down, select **Subcategory**.
11. In the Value drop-down, select **[ProductSubCategoryKey]**.
12. Click **OK**.
13. Click **Run** to view the report.
14. In the report, click **Bike Racks**, and note that the report drills down to a summary report for accessories.
15. Close Report Builder, click **No** when prompted to Save.

Check Your Knowledge

Question	
<p>You have created a summary report that contains economic data by country. You want to give users the option to get further information for regions within a country. What is the most straightforward way to implement this?</p>	
<p>Select the correct answer.</p>	
<input type="checkbox"/>	Create a drillthrough action for each country that has detailed region information.
<input type="checkbox"/>	Create groupings by country and add a drilldown for the country column.
<input type="checkbox"/>	Sort by country then by region.
<input type="checkbox"/>	Configure a page break for each country.

Lab: Summarizing report data

Scenario

The CIO of Adventure Works Cycles wants you to develop reports that show Product Category information. You are considering Report Builder and Report Designer as report creation tools and will use this task to test the capabilities and ease-of-use of each tool.

Objectives

After completing this lab exercise, you will be able to:

- Add groups to your reports.
- Sort data within your reports.
- Implement drilldown in your reports.

Estimated Time: 60 minutes

Virtual machine: **10990C-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa55w.rd**

Exercise 1: Sorting and grouping in Report Builder

Scenario

The CIO of Adventure Works Cycles has asked you to develop a report that shows sales value by Product Category and Product Subcategory. You must develop a report that groups and sorts data appropriately.

The main tasks for this exercise are as follows:

1. Prepare the lab environment
2. Create a matrix with groups
3. Add drilldown to a report

► Task 1: Prepare the lab environment

1. Ensure the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab06** folder, right-click **Setup.cmd** and click **Run as administrator**.
3. Click **Yes** when prompted.

► Task 2: Create a matrix with groups

Add groups to a matrix

1. Open **Report Builder** and open **Groups.rdl**. The report is in the folder **D:\Labfiles\Lab06\Starter\Groups\Groups**. The report contains an empty matrix.
2. Add the field **EnglishProductSubcategoryName** to the second row of the left column.
3. **Widen** the column, amend the column heading to read **Subcategory**, and make it **Bold**.
4. In the second row of the second column, add **SalesValue**.
5. Format **SalesValue** to be a **Number** with **0 decimal places**, and **Use 1000 Separator**.
6. Make the column heading Sales Value **bold**.

7. Click the **[Sum(SalesValue)]** text box and note the orange scope indicators on the row and column.
8. Click **Run** to view the report.
9. Click **Design**.

Add a column group

1. In the Column Groups pane, double-click **ColumnGroup** and add a group for **Year**.
2. Overtyping the **ColumnGroup** name and type **Year**.
3. Group on **[Year]**.
4. Sort on **[Year]** in **A to Z** sequence.
5. Change the column heading from Sales Value to **[Year]** (ensure you type each character in turn including the brackets to denote a field).
6. Click **Run** to view the report, and then click **Design**.

Add a row group

1. Add a **New Row Group**, selecting **Parent Group** and using **[EnglishProductSubCategoryName]** with a **group footer**.
2. In the group footer, type **Total** in the middle column. Make this **right-aligned**.
3. In the bottom-right cell add **SalesValue** and format it to **0 decimal places**, and **Use 1000 Separator**.
4. Click **Run** to view the report, and then click **Design**.

Add a totals column

1. Add a new column to the right of the **[Year]** column.
2. Add **SalesValue** to the details row.
3. Amend the heading to **Total**, **right align** it and make it **bold**.
4. Format both the bottom right cell, and the one above it to **0 decimal places**, and **Use 1000 Separator**.
5. **Run** the report, and then click **Design**.

Add a totals row

1. Add a **new row** below the bottom row, and select **Outside Group Below**.
2. In the bottom cell of the Category column, type **Grand Total** and make it **bold** and **right aligned**.
3. In the bottom cell of the **[Year]** column, add **SalesValue**.
4. In the bottom cell of the **Total** column, add **SalesValue**.
5. Format both new total cells to be numbers with **0 decimal places**, and with a **thousands separator**.
6. Format both cells to be **bold** and **right-aligned**.
7. Click **Run** to preview the report, then click **Design**.
8. Leave **Report Builder** open for the next lab exercise.

► Task 3: Add drilldown to a report

1. Navigate to **D:\Labfiles\Lab06\Starter\Drilldown\Drilldown**.
2. Open **Drilldown.rdl** in design mode.

3. Click **Run** to view the report.

Add drilldown actions

1. View the report in **Design** mode.
2. Click the **down arrow** next to Row Groups and Column Groups and select **Advanced Mode**.
3. In Row Groups, select **EnglishProductSubcategoryName**.
4. In Properties, under Visibility, change the Hidden property to **True**.
5. In **ToggleItem**, select **EnglishProductCategoryName**.
6. Preview your changes.
7. Repeat these steps to add drilldown to the **detail row**.
8. Preview your changes.
9. **Exit** without saving changes.

Results: After completing this lab exercise, you will be able to:

Create groups within a report.

Set the sort order.

Create drilldown actions within a report.

Exercise 2: Sorting and grouping in Report Designer

Scenario

The CIO of Adventure Works Cycles has asked you to develop a report that shows sales value by Product Category and Product Subcategory. You must develop a report that groups and sorts data appropriately.

The main tasks for this exercise are as follows:

1. Prepare the lab environment
2. Create a matrix with groups
3. Add drilldown to a report

► Task 1: Prepare the lab environment

1. Ensure the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab06** folder, right-click **Setup.cmd** and click **Run as administrator**.
3. Click **Yes** when prompted.

► Task 2: Create a matrix with groups

Add groups to a matrix

1. Click **Visual Studio 2015** on the toolbar to open it. Navigate to **D:\Labfiles\Lab06\Starter\Groups** and open **Groups.sln**.
2. Open **Groups.rdl** in design mode. The report contains an empty matrix.
3. Add the field **EnglishProductSubcategoryName** to the second row of the left column.

4. Amend the column heading to read **Subcategory**, and make it **Bold**.
5. In the second row of the second column, add **SalesValue**.
6. **Format** SalesValue to be a **Number** with **0 decimal places**, and **Use 1000 Separator**.
7. Make the column heading Sales Value **bold**.
8. Click in **Sum(Sales Value)** and note the orange scope indicators on the row and column.
9. Click **Preview** to view the report and then click **Design**.

Add a column group

1. In the Column Groups pane, double-click **ColumnGroup** and add a group for **[Year]**.
2. Sort on **Year** in **A to Z** sequence.
3. Change the column heading from Sales Value to **[Year]** (ensure you type each character in turn including the brackets to denote a field).
4. Click **Preview** to view the report, and then click **Design**.

Add a row group

1. Add a **New Row Group**, selecting **Parent Group** and using **[EnglishProductCategoryName]** with a **group footer**.
2. In the group footer, type **Total** in the middle column. Make this **right-aligned**.
3. In the bottom-right cell add **SalesValue** and format it to **0 decimal places**, and **Use 1000 Separator**.
4. Click **Preview** to view the report, and then click **Design**.

Add a totals column

1. Add a **new column** specifying **Outside Group Right** to the right of the **[Year]** column.
2. Add **SalesValue** to the details row.
3. Amend the heading to **Total**, **right align** it and make it **bold**.
4. Format both the bottom right cell, and the one above it to **0 decimal places**, and **Use 1000 Separator**.
5. **Preview** the report, and then click **Design**.

Add a totals row

1. Add a **new row** below the bottom row, and select **Outside Group Below**.
2. In the bottom cell of the Category column, type **Grand Total** and make it **bold** and **right aligned**.
3. In the bottom cell of the **[Year]** column, add **SalesValue**.
4. In the bottom cell of the **Total** column, add **SalesValue**.
5. Format both new total cells to be numbers with **0 decimal places**, and with a **thousands separator**.
6. Format both cells to be **bold** and **right-aligned**.
7. Click **Preview** to preview the report, then click **Design**.
8. Save and close the **Solution**.
9. Leave **Visual Studio 2015** open for the next lab exercise.

► **Task 3: Add drilldown to a report**

1. Open **Drilldown.sln** located in **D:\Labfiles\Lab06\Starter\Drilldown**.
2. Open **Drilldown.rdl** in design mode.
3. **Preview** the report.

Add drilldown actions

1. View the report in **Design** mode.
2. Click the **down arrow** next to Row Groups and Column Groups and select **Advanced Mode**.
3. In Row Groups, select **[EnglishProductSubcategoryName]**.
4. In Properties, under Visibility, change the **Hidden** property to **True**.
5. In **ToggleItem**, select **EnglishProductCategoryName**.
6. **Preview** your changes.
7. Repeat these steps to **add drilldown** to the detail row.
8. **Preview** your changes.
9. Exit without saving changes.

Results: After completing this lab exercise, you will be able to:

Create groups within a report.

Set the sort order.

Create drilldown actions within a report.

Question: Why would you add drilldown to a report?

Module Review and Takeaways

In this module, you have explored how to sort and group, create subreports, and use drilldown and drillthrough to make reports interactive. Careful design of these elements within a report will influence how well users work with a report.



Best Practice: Grouping and sorting is as much part of the report design as the data that will be included. Before you develop your report, consider the different ways that the data might be grouped.

Review Question(s)

Question: Of the reports that are circulated within your company, which ones have poorly designed grouping or sorting? How could these be improved?

Module 7

Sharing Reporting Services Reports

Contents:

Module Overview	7-1
Lesson 1: Schedules	7-2
Lesson 2: Report caching, snapshots, and comments	7-7
Lesson 3: Report subscription and delivery	7-13
Lab: Sharing Reporting Services reports	7-20
Module Review and Takeaways	7-23

Module Overview

When you have published a Reporting Services report, users can view the report interactively. In some situations, it's advantageous to run reports automatically, either to improve performance through caching and snapshots, or to deliver reports to users—by using email or other mechanisms. To run reports automatically, you need to understand how Reporting Services manages scheduling.

This module covers report scheduling, caching and the report life cycle, and automatic subscription and delivery of reports.

Objectives

At the end of this module, you should be able to:

- Explain schedules.
- Describe report caching, report snapshots, and report comments.
- Configure report subscriptions.

Lesson 1


Schedules

Reporting Services schedules allow you to automate the processing and distribution of reports, by permitting you to carry out Reporting Services tasks—such as delivering a report, or refreshing a data snapshot—on a recurring schedule.

Reporting Services schedules come in two forms:

- Shared schedules
- Report schedules

Although they are created in different ways, both forms use the same underlying mechanisms. Reporting Services schedules depend on the SQL Server Agent of the SQL Server instance that hosts the **ReportServer** database to carry out scheduling.

 **Note:** Because schedules are dependent on SQL Server Agent, they are not available in editions of SQL Server that do not include the SQL Server Agent component—such as Express edition.

Lesson Objectives

At the end of this lesson, you should be able to:

- Describe shared schedules.
- Describe report schedules.
- Explain the configuration steps that are needed for schedules.

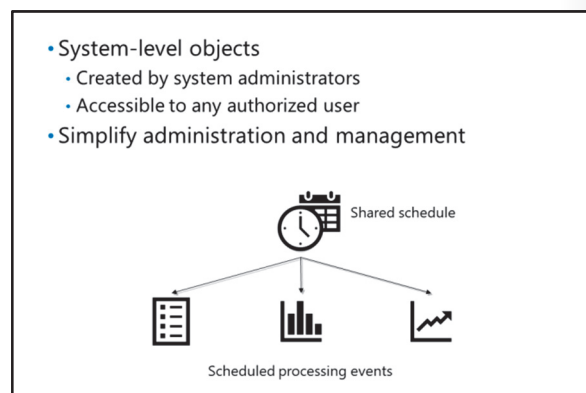
Shared schedules

Shared schedules are system-level schedules that you manage as objects on a Reporting Server. As the name suggests, a single shared schedule can be used to start many scheduled operations.


Because shared schedules are system-level objects, they are only created by system administrators. You view and manage shared schedules from the Reporting Services web portal, or with Microsoft SQL Server Management Studio 2017. Any authorized user can reference a shared schedule.

The principal advantage of using shared schedules is that they make it easy to manage many scheduled events. Imagine that you are managing a server with 100 scheduled events; if you need to adjust the start time of all the schedules by an hour, it will be quicker and less error-prone if you edit a few shared schedules, rather than editing an independent schedule attached to each event.

Another advantage of shared schedules is that they are easier to visualize and compare to other scheduled tasks in your environment.



For detailed instructions on managing shared schedules, see the topic *Create, Modify, and Delete Schedules* in Microsoft Docs:

 **Create, Modify, and Delete Schedules**

<https://aka.ms/l1o6y9>

Report schedules


Report schedules are linked to an individual report, and are stored as part of the report definition in the Reporting Services web portal. Report schedules are used to create report execution properties, or create a subscription to a report.

Because they are stored at report level, report schedules are created by any user who is authorized to view the associated report.

You typically use report schedules when no shared schedule exists that meets your exact scheduling requirements.

- One-to-one relationship between report and schedule
 - Can be created by any user who is authorized to view the report
- Used when a suitable shared schedule is not available



 **Best Practice:** You should plan to use shared schedules whenever possible; although report schedules offer convenience for users, they must be managed individually.

Configuring schedules

Dependencies for schedules

Schedules are dependent on the SQL Server Agent and the Report Server service. When a schedule is active, an associated SQL Server Agent job runs at the scheduled interval; when the job runs successfully, it adds rows to an event queue table in the **ReportServer** database. The Report Server service reads rows from the event queue table and carries out the associated action.

If the SQL Server Agent is not available, no events are added to the event queue; when the SQL Server Agent becomes available, it will start adding new events to the event queue table. No events are added retrospectively for the period that SQL Server Agent was not available.

If the Report Server service is not available, SQL Server Agent continues to add events to the event queue table. When the Report Server service becomes available, the queued events are processed.

- Dependencies for schedules
 - Report Server
 - SQL Server Agent
- Enabling schedules
 - Enabled by default
 - rsreportserver.config flags
- Unattended processing account



Note: When backing up a **ReportServer** database, you should consider that the definitions for SQL Server Agent jobs associated with Reporting Services schedules are stored in the **msdb** database. You will need a backup of **msdb** (or scripts to recreate the jobs) if you want to transfer the schedules to another server.

Enabling schedules

When you install Reporting Services, schedules are enabled by default. You disable and enable schedules by editing the `rsreportserver.config` file (located by default at `C:\Program Files\Microsoft SQL Server Reporting Services\<instance name>\ReportServer`).

To disable schedules, change the values of the following three properties to **False**:

```
<IsSchedulingService>
<IsNotificationService>
<IsEventService>
```

You enable and disable individual shared and report schedules through the Reporting Services web portal, or through Microsoft SQL Server Management Studio 2017 (shared schedules only).

Unattended processing account

For scheduled events to connect to remote data sources, you must either configure your data sources with credentials that are not dependent on the current user's identity, or configure an unattended processing account.

You configure an unattended processing account through the Reporting Services Configuration Utility. When an unattended processing account is configured, it's used to connect to remote data sources during processing.

For more information about configuring schedules, see the topic *Schedules* in Microsoft Docs:



Schedules

<https://aka.ms/Ufk4he>

Demonstration: Shared schedules

In this demonstration, you will see how to configure a shared schedule.

Demonstration Steps

1. Ensure that both **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Demofiles\Mod07** folder, run **Setup.cmd** as Administrator. Click **Yes** when prompted to confirm that you want to run the command file, and wait for the script to finish.
3. Using Internet Explorer, navigate to **http://mia-sql/reports_sql2**, then click the settings icon (the gear icon in the upper right of the page), then click **Site settings**.
4. On the **Site settings** page, click **Schedules** then click **+New schedule**.
5. In the **Schedule name** box, type **Every minute**, then select **Hour**.
6. In the **hours** box type **0**, then in the **minutes** box type **01**, then click **Apply**.
7. On the **Schedules** page, select the row with the **Every minute** schedule and observe that you can enable and disable the schedule from this page.

8. Start Microsoft SQL Server Management Studio 2017.
9. In the **Connect to Server** dialog box, in the **Server type** box select **Reporting Services**.
10. Confirm that the **Server name** box has the value **MIA-SQL\SSRS**, then click **Connect**.
11. In Object Explorer, expand the **Shared Schedules** node (if Object Explorer is not visible, click **View** then click **Object Explorer**).
12. Right-click **Every minute** then click **Properties**.
13. In the **Schedule Properties** window, on the **General** page, observe that the schedule definition matches the schedule you defined in the web portal.
14. Click **Reports** and note that the **Reports that use this schedule** box is empty; this is because no reports are currently attached to the schedule, then click **Cancel**.
15. In Object Explorer, click **Connect**, then click **Database Engine**.
16. In the **Connect to Server** dialog box, in the **Server name** box type **MIA-SQL\SQL2**, then click **Connect**.
17. In Object Explorer, expand the **SQL Server Agent** node, then expand **Jobs**. One job will have a GUID name (in the pattern xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx)—the exact value will vary. This is the SQL Server Agent job associated with the schedule.
18. Right-click the job, then click **View History**.
19. In the **Log File Viewer - MIA-SQL\SQL2** window, observe that the job has run every minute since it was created, then close the window.
20. In Object Explorer, right-click the schedule job, then click **Properties**.
21. In the **Job Properties - <job>** window, on the **General** page, note the warning about modifying the job.
22. On the **Schedules** page, click **Edit**.
23. In the **Job Schedule Properties - Schedule_1** window, observe that the job configuration matches the schedule you defined in the web portal, then click **Cancel**, then click **Cancel**.
24. Click **New Query**, then in the query pane, type the following, and then click **Execute**:

```
USE ReportServer;
GO
SELECT * FROM Schedule;
```
25. In the results, observe that one row is returned; the value of the **Name** column is **Every minute**, and the value of the **ScheduleID** column matches the name of the SQL Server Agent job that you reviewed in the previous step.
26. Leave Microsoft SQL Server Management Studio 2017 and Internet Explorer open for the next demonstration.

Check Your Knowledge

Question	
If the SQL Server Agent service is running, but the Report Server service is stopped, what will happen when a schedule's start time is reached?	
Select the correct answer.	
<input type="radio"/>	No events are added to the event queue in the ReportServer database. No actions associated with the schedule will run.
<input type="radio"/>	Events are added to the event queue in the ReportServer database, but are not processed. No actions associated with the schedule will run until Report Server is started.
<input type="radio"/>	Any actions associated with the schedule will run as normal.
<input type="radio"/>	Any actions associated with the schedule will run, but the actions will not be reflected in the Reporting Services web portal.

Lesson 2

Report caching, snapshots, and comments

This lesson covers the processing life cycle of a report, and how using those concepts introduces report caching and report snapshots—which are techniques that you use to improve performance on a Reporting Services server. The lesson also introduces comments that give users a way to collaborate on interpreting reports within the Reporting Services web portal.

Lesson Objectives

At the end of this lesson, you should be able to:

- Explain the life cycle of a Reporting Services report.
- Describe the effects of caching report data.
- Describe report history snapshots.
- Use report comments.

The report life cycle

When you access a Reporting Services report, the report is processed by the Report Server in three phases:

- Report processing
- Data processing
- Rendering

Report processing

In the report processing phase, the Report Server reads the report definition to extract information that's relevant to the layout of the report, in addition to the dataset(s) associated with each report element.

Data processing

In the data processing phase, the Report Server executes any queries that are associated with the report datasets. When these queries return, the results are combined with the layout information that's extracted from the report definition in the report processing phase. Results are compiled into an intermediate format, in the form of a common language runtime assembly.

The intermediate format can be stored and used as the basis for a cached report, report snapshot, or an entry in the report history.

Rendering

In the rendering phase, the intermediate format generated in the data processing phase is rendered into an output format, such as HTML or PDF.



For more information about the report life cycle, see the *Stages of Reporting Services paginated reports* section of the Reporting Services Concepts (SSRS) topic in Microsoft Docs:



Reporting Services Concepts (SSRS) - Stages of Reporting Services paginated reports

<https://aka.ms/Enuvue>

Benefits and costs of caching

Reports that contain a large amount of data or complex functions might be slow to render. To improve performance, you use the caching functionality in Reporting Services to cache datasets and reports when they are first accessed—so they are rendered more quickly for subsequent requests. You can also use a refresh plan to recache reports, so they are already in cache for the first time they are requested.

Cached reports and datasets are stored in memory, so data is not retrieved from the data source when the report is rendered. Additionally, if a report or dataset includes parameters, a cached instance is created for each distinct parameter combination requested. Caching is not suitable in the following scenarios:

- When real-time reporting is required, and data should update every time the report is viewed.
- If a report has many parameters or a wide range of commonly requested parameter value combinations.
- If the data source for a report or dataset must use Windows® integrated authentication or credentials provided by the user who makes the request.

You configure caching in the properties of a report in the Reporting Services web portal. When you configure caching for a report or dataset, you use the following options to specify when the cached instance should expire:

- **After a specified period.** For example, you could cache a report for one hour after it's rendered. The cached instance then expires, and a subsequent request will result in a fresh cached instance.
- **On a report or dataset-specific schedule.** For example, you could configure a report to be cached until midnight each day. A request after midnight will render a fresh copy.
- **On a schedule.** For example, if you need to cache multiple reports and datasets, you use a shared schedule to ensure that they all expire at the same time.

Additionally, a report or dataset might be removed from the cache at any time if it's replaced with a new version.

You configure caching in the properties of a report in the Reporting Services web portal.

For more information about configuring report caching, see the section *Creating a cache snapshot* of the *Working with snapshots (web portal)* topic in Microsoft Docs:



Working with snapshots (web portal) - Creating a cache snapshot

<https://aka.ms/K409m4>

For more information about the behavior of cached reports, see the topic *Caching Reports (SSRS)* in Microsoft Docs:

 **Caching Reports (SSRS)**

<https://aka.ms/C2o6tg>

Report history snapshots


You use the caching technique to optimize report performance. However, there might be situations where you need greater predictability about the specific point in time that data in a report represents. In such scenarios, you use snapshots to create a report history and enable users to view specific versions of a report, relating to the time when the snapshot was created.

Snapshots are created for specific combinations of parameters and require that the credentials used by the report data sources are stored securely on the server. Additionally, when using snapshots, you should not map report parameters to dataset parameters. Instead, use a dataset query that returns all the data, and apply filters in the report, based on report parameters.

You configure snapshots for a report in the Reporting Service web portal, and have them generated on a scheduled basis. You also create a snapshot on demand when viewing the history of a report in the web portal.

For more information about working with snapshots, see the topic *Working with snapshots (web portal)* in Microsoft Docs:

Working with snapshots (web portal)

 <https://aka.ms/Dzgkxq>

- Create a report history of snapshots—rendered reports for specific points in time
- Create snapshots manually or on a scheduled basis

Comments

Comments are a new feature in Reporting Services 2017. They enable users to add comments and reply to comments on supported report types: paginated, mobile, and Power BI. Users can also include attachments in their comments, enabling a full discussion to be hosted about a report.

The comments are stored in a table named **dbo.Comments** in the **ReportServer** database. The table holds:

- A unique identifier for the comment.
- GUIDs to relate it to the report.

- Users can add comments to reports, and
 - Reply to comments
 - Attach documents to comments
- Enables a discussion to be kept with the report
- Permissions for comments can be set at different levels

- Details for the user who makes the comment.
- An identifier to link replies to their original comment.

Permissions

In some scenarios, you might not want users to be able to comment on a report. There are three levels of security for the comments:

1. Comment on reports. Allow a user to see comments, and post, edit, and delete their own comments.
2. Manage comments. Allow a user to see comments, and post, edit, and delete their own comments—and be able to delete other people's comments.
3. Neither. Users cannot see the comments feature.

By default, members of the **Publisher**, **My Reports**, and **Content Manager** roles can manage comments, and the same group of roles. Members of the **Browser** role can also comment on reports.

How to add comments

1. Open the web portal.
2. In the upper right corner, select **Comments**. The Add Comment blade appears.
3. In the box, type your comment then click **Post Comment**.
4. To attach a file, click **Attach file**.
5. Comments can be viewed with newest first, or oldest first.
6. Click **Reply here** to reply to a comment.

Supported formats for attached files

The supported file formats for attached files are:

- .tif
- .tiff
- .png
- .jpg
- .jpeg
- .gif
- .gmp
- .pdf



Note: Comments are not available in versions of Reporting Services prior to SQL Server 2017.

Demonstration: Controlling caching with shared schedules

In this demonstration, you will see how to manage report caching with a shared schedule.

Demonstration Steps

1. Using Internet Explorer, navigate to **http://mia-sql/reports_sql2**, then click the settings icon (the gear icon in the upper right of the page), then click **Site settings**.
2. On the **Site settings** page, click **Schedules** then click **+New schedule**.
3. In the **Schedule name** box, type **Daily**, then select **Day**, then click **Apply**.
4. Click **SQL Server Reporting Services** to return to the home page.
5. On the portal home page click **AdventureWorks Sample Reports**. Click the **AdventureWorks** data source.
6. On the **Properties** page, select **Using the following credentials**.
7. In the **Type of credentials** box, select **Database user name and password**.
8. In the **User name** box, type **SSRS**, then in the **Password** box type **Pa55w.rd**, then click **Apply**.
9. In the breadcrumb trail at the top of the page, click **AdventureWorks Sample Reports**, then right-click the **Sales_Order_Detail** report, then click **Manage**.
10. On the **Manage Sales_Order_Detail** page, click **Caching**. Select **Always run this report against pregenerated snapshots**, then select **Create cache snapshots on a schedule**.
11. Select **Shared schedule**, then in the shared schedule box select **Daily**. Select **Create a cache snapshot when I click Apply on this page**, then click **Apply**.
12. In the breadcrumb trail at the top of the page, click **Sales_Order_Detail**. Notice that the line total for line 1 of sales order # **SO57030** is **\$899.24**. If the figure doesn't match, repeat steps 9 to 12.
13. In Microsoft SQL Server Management Studio 2017, click **New Query**.
14. On the **Query** menu, point to **Connection**, then click **Change Connection**.
15. In the **Connect to Database Engine** dialog box, in the **Server name** box, type **MIA-SQL**, then click **Connect**.
16. In the query pane, type the following query, and then click **Execute**:

```
UPDATE AdventureWorks.Sales.SalesOrderDetail
SET UnitPrice = 100.00
WHERE SalesOrderDetailId = 60669
```

17. In Internet Explorer, click the **Refresh** button. Notice that the line total for line 1 of sales order # **SO57030** is still **\$899.24**, even though the value has been updated in the database. The report is being generated from the cache.
18. In the breadcrumb trail at the top of the page, click **AdventureWorks Sample Reports**, then right-click the **Sales_Order_Detail** report, then click **Manage**.
19. On the **Manage Sales_Order_Detail** page, click **Caching**.
20. Select **Create a cache snapshot when I click Apply on this page**, then click **Apply**.
21. In the breadcrumb trail at the top of the page, click **Sales_Order_Detail**. Notice that the line total for line 1 of sales order # **SO57030** is now reflects the updated value of **\$600.00**. If the figure doesn't match, repeat steps 18 to 21.

22. Leave Internet Explorer and Microsoft SQL Server Management Studio 2017 open for the next demonstration.

Question: In the previous demonstration, why was it necessary to change the username and password for the **AdventureWorks** data source? Are there any alternative methods that you could use to resolve this issue?

Demonstration: Using comments

In this demonstration, you will see how to add comments to a report.

Demonstration Steps

1. In Internet Explorer, click **Comments** in the upper right of the web portal.
2. Click the comment box (that contains the text **Start typing here**) then type **This report is cached, and updates once a day**, then click **Post Comment**.
3. In the breadcrumb trail at the top of the page, click **AdventureWorks Sample Reports**, then click **Sales_Order_Detail**.
4. Click **Comments** in the upper right of the web portal. Observe that the comment you just added is visible, and you have the option to add a new comment or reply to an existing comment. You can also edit or delete the existing comment using the pencil or trashcan icons.
5. Leave Internet Explorer open for the next demonstration.

Lesson 3

Report subscription and delivery

In addition to optimizing report execution for interactive viewing, you can configure reports for automatic delivery through subscriptions. Users receive reports automatically by email or in a shared folder.

Lesson Objectives

At the end of this lesson, you should be able to:

- Describe email delivery of report subscriptions.
- Describe file share delivery of report subscriptions.
- Create and manage subscriptions.
- Explain data-driven subscriptions.

Email delivery

You use the email extension in Reporting Services to deliver a report to one or more email addresses. Email delivery takes place over the simple mail transfer protocol (SMTP). To deliver a report by email, you must configure the SMTP server that Reporting Services will use to send email, then configure report delivery using a report subscription. The SMTP server must be on the same network as your Reporting Services instance; mail sent by Reporting Services is not encrypted using transport-layer security/secure sockets layer (TLS/SSL). Each subscription is attached to a single report; you will need to create a subscription for each report that you want to deliver by email.

- Report delivery by SMTP email
- Delivery methods
 - Notification only
 - Notification and URL
 - Embedded HTML report
 - Attached report
- Delivery addresses are not validated
- Control access by restricting delivery domains and delivery methods in `rsreportserver.config`

Configuring email delivery

You configure SMTP server settings for a Reporting Services instance through the Reporting Services Configuration Manager.

When you configure email delivery, you choose how the report is delivered to the recipients. The following choices are available—the email might contain:

- A notification message only.
- A notification message and a hyperlink to the report on the Reporting Services web portal.
- An embedded report—if you select the **Web archive** rendering format, the report will be embedded as HTML in the body of the email message.
- An attached report—if you select a rendering format other than **Web archive**, such as a PDF or CSV, the report will be delivered as an attachment to the email message.

You should select the *notification message only* option or, for large reports that might exceed your organization's maximum message limit or attachment size limit, choose the *notification and hyperlink* option.

In addition to configuring the delivery method, you can customize the subject line of the email message that delivers the report. Use the following placeholders in the subject text—they will be replaced dynamically when the report is delivered:

- **@ReportName**—will be replaced with the report name.
- **@ExecutionTime**—will be replaced with the report execution time. The time displayed will use the local time of the Reporting Services server.

Working with delivery addresses

When you set up subscriptions to use email delivery, you must specify the email address or addresses in the To:, Cc:, and Bcc: fields correctly. Reporting Services does not download a list of email addresses from the SMTP server, or validate the email addresses that you supply. To specify multiple email addresses in the To:, Cc:, and Bcc: fields, delimit them with a semicolon (",").

Controlling email delivery

By default, users configure subscriptions to send email to any domain; you restrict the domains that users deliver to by email by editing the `rsreportserver.config` file. You can also configure Reporting Services not to show the To:, Cc: or Bcc: fields to users when they create a subscription—when you do this, users can only create subscriptions that deliver reports by email for their own email address.

If you are concerned about preventing your reports leaving your organization, you configure Reporting Services to prevent users from creating embedded or attached reports with email delivery. You use *notification only* or *notification and URL* delivery to control access to Reporting Services reports that have built-in security.

For more information about email delivery in Reporting Services, see the topic *E-Mail Settings - Reporting Services Native mode (Configuration Manager)* in Microsoft Docs:

 **E-Mail Settings - Reporting Services Native mode (Configuration Manager)**

<https://aka.ms/lmg18n>


File share delivery

You use the file share delivery extension in Reporting Services to deliver reports to a folder in a file system that has a report subscription. The file share delivery extension requires no configuration, although you must supply the credentials that Reporting Services will use to access the target folder; you do this as part of the subscription definition, or by configuring a file share account using Reporting Services Configuration Manager.

When a report is delivered to a file share, it's rendered as a static file that's written to the target folder. Any interactive features in the source report do not operate when the report is viewed from a static file. These features include:

- Dynamic matrix reports—only the default view of the top-level items in the matrix is accessible.
- Charts—only the default presentation is visible.

- Delivery to a UNC share
 - Specify permissions as part of a subscription or with a file share account
 - Target folder must exist
- Reports are rendered to a static file
 - Interactive elements are not supported
- Select an output format where the report is rendered to a single file
 - Avoid HTML 4.0 targets

 **Note:** If you want to deliver a report and retain interactive features, consider using email delivery to send a URL to the Reporting Services web portal as part of the delivery message. All interactive features are available in reports that are viewed from the web portal.

Target folders

You specify the target folder for file share delivery with a UNC path in the form:

```
\\<server name>\<folder path>
```

The target folder must exist; Reporting Services will not create folders in the target file system.

Delivery formats

Although reports that are delivered to a file share can be rendered into any supported format—including Excel, PDF, CSV and TIFF—not all formats are ideally suited to file share delivery. File share delivery works most successfully when the delivery format contains the report in a single file; formats such as HTML 4.0 should be avoided, because they render the report into multiple files. Consider using the following formats for file share delivery:

- Excel
- PDF
- TIFF
- Web archive

For more information about using file share delivery, see the topic *File Share Delivery in Reporting Services* in Microsoft Docs:

 **File Share Delivery in Reporting Services**


<https://aka.ms/Gnmtn7>

For more information about configuring a file share account, see the topic *Subscription Settings and a File Share Account (Configuration Manager)* in Microsoft Docs:

 **Subscription Settings and a File Share Account (Configuration Manager)**

<https://aka.ms/Kh bzlk>

Creating and managing standard subscriptions

 **Note:** Standard subscriptions are created by Reporting Services users to deliver a report by email or to a file share. Data-driven subscriptions are covered later in this lesson.

Creating subscriptions

You create subscriptions through the Reporting Services web portal using the **Subscribe** menu of the report to which you want to subscribe. Using

- Create subscriptions
 - Use the **Subscribe** menu in the Reporting Services portal
 - Subscriptions are built from:
 - A schedule for delivery
 - Delivery method configuration
 - Parameter values (optional)
- Modify and delete subscriptions:
 - Find your subscriptions in the **My Subscriptions** page
 - Only the subscription owner (or an administrator) can change or delete a subscription

the **Subscribe** menu, you create an email or a file share subscription to the report.

A subscription is built from a schedule—which can be a shared schedule, or a schedule embedded in the subscription—to determine the frequency of the report that's delivered, and configuration for the delivery type. If the report accepts parameters, the subscription will also contain values to use for the parameters when the subscription is delivered.

Modifying and deleting subscriptions

The user who creates a subscription is the subscription owner. A user modifies or deletes the subscriptions that they own. You change the ownership of a subscription using a script, or through the properties of the subscription. An administrator can modify subscriptions that belong to any user.

You view all the subscriptions that you own on the **My Subscriptions** page in the Reporting Services web portal.



Note: Subscriptions are only available in Enterprise and Standard editions of SQL Server Reporting Services.

For more information about working with subscriptions, see the topic *Create and Manage Subscriptions for Native Mode Report Servers* in Microsoft Docs:



Create and Manage Subscriptions for Native Mode Report Servers

<https://aka.ms/N4manh>

Data-driven subscriptions

You use a data-driven subscription to deliver a subscription dynamically, based on values that are retrieved from a data source when the subscription runs. You also use a data-driven subscription to deliver a report to a dynamic list of recipients, and to vary such features as the parameters passed to the report, and the delivery format for each recipient.



Note: Data-driven subscriptions are only available in SQL Server Enterprise edition.

- Deliver a subscription from a dynamic configuration retrieved at runtime
- Some static elements:
 - Report
 - Delivery extension
 - Report data source
 - Subscription data query
- Other elements are dynamically mapped from the subscription data query, including:
 - Recipients
 - Recipient report format
 - Recipient parameters

The following elements of a data-driven subscription are fixed:

- The report associated with the subscription.
- The report delivery extension. This must be one of the following:
 - Email delivery.
 - File share delivery.
 - Null delivery—the null delivery extension is used to add a report to the Reporting Services cache, to improve performance in subsequent queries.
 - A custom delivery extension.

- The report data source.
- The subscriber data query.

Other elements of the subscription might be variable, and queried from the subscriber data query each time the subscription is processed. Variable elements include:

- Subscriber name.
- Subscriber email address.
- Subscriber report output format.
- Report parameter values for the subscriber.

You create and manage data-driven subscriptions through the Reporting Services web portal. When you define the data-driven subscription, you map columns from the subscriber data query to values in the subscription configuration. Data-driven subscriptions that you own will appear on your **My Subscriptions** page.

For more information on data-driven subscriptions, see the topic *Data-Driven Subscriptions* in Microsoft Docs:



Data-Driven Subscriptions

<https://aka.ms/C8jzo8>

For more information on managing data-driven subscriptions, see the topic *Create, Modify, and Delete Data-Driven Subscriptions* in Microsoft Docs:



Create, Modify, and Delete Data-Driven Subscriptions

<https://aka.ms/Rslger>

Demonstration: Using data-driven subscriptions

In this demonstration, you will see how to configure a data-driven subscription.

Demonstration Steps

1. Ensure the **10990C-MIA-CLI** virtual machine is running.
2. On **10990C-MIA-SQL**, in the **D:\Demofiles\Mod07\Demo04** folder, double-click **Report_Subscription.sql**, and then double-click **SSMS** to open it in Microsoft SQL Server Management Studio 2017.
3. On the **Query** menu, point to **Connection**, and then click **Change Connection**.
4. In the **Connect to Database Engine** dialog box, in the **Server name** box type **MIA-SQL** then click **Connect**.
5. Review the Transact-SQL code and note that it creates and populates a table named **Report_Subscription**, that contains the following columns:
 - **SubscriptionId** – a unique primary key.
 - **RecipientEmail** – the email address of a subscription recipient.
 - **ReportFormat** – the format in which the report should be rendered.

- **IncludeLink** – a Boolean value that indicates whether the subscription email message should include a link to the report on the report server.
- 6. Click **Execute** to run the query; when it has completed, close Microsoft SQL Server Management Studio 2017.
- 7. In the **Microsoft SQL Server Management Studio 2017** dialog box, click **No**.
- 8. In Internet Explorer, navigate to http://mia-sql/reports_sql2, click **AdventureWorks Sample Reports**, right-click **Sales_by_Region**, and then click **Subscribe**.
- 9. On the **New Subscription** page, in the **Description** box, type **Data-driven demo**, and then select **Data-driven subscription**.
- 10. Confirm that the **Deliver the report to** box has the value **E-Mail**, then click **Edit dataset**.
- 11. Click the ellipsis button next to **Select a shared data source**. Click **AdventureWorks Sample Reports** then click **AdventureWorks**.
- 12. In the **Query** box, type **SELECT * FROM dbo.Report_Subscription** then click **Apply**.
- 13. In the **Delivery options (E-Mail)** section, change the settings for the following parameters:

Parameter	Source of Value	Value/field
To	Get value from dataset	RecipientEmail
Render Format	Get value from dataset	ReportFormat
Include Link	Get value from dataset	IncludeLink

- 14. In the **Schedule** section (near the top of the page) click **Edit schedule**.
- 15. On the **Edit schedule** page, select **Once**. Amend the values in the **Start time** hour and minute boxes to be two minutes ahead of the current server time (the server time is displayed in the bottom right of the screen, on the taskbar), then click **Apply**.
- 16. On the **New Subscription** page, click **Create subscription**.
- 17. Wait for two minutes, then click the **Settings** icon (the gear icon in the upper right of the page) then click **My subscriptions**. The **Last run** column should contain a recent date and time, and the **Result** column should contain the value **Processing: 3 processed of 3 total; 0 errors**. If you get any errors reported, delete the subscription, and repeat the steps from 8 onwards.
- 18. Log onto **10990C-MIA-CLI** with the username **Student** and the password **Pa55w.rd**.
- 19. In File Explorer, in the address bar type **\\mia-sql\mailroot** then press ENTER.
- 20. In the **Windows Security** dialog box, enter the username **student@adventureworks.msft** and the password **Pa55w.rd**, and then click **OK**.
- 21. When the connection is successful, double-click **Drop**. After a short period of time—up to five minutes—three .eml files should appear in the folder. Each of these corresponds to one of the rows in the **Report_Subscriptions** table that you created at the beginning of this demonstration.
- 22. Double-click each .eml files, and then double-click **Outlook 2016** to open it in Outlook and view the contents of the message. Observe that the message sent to **pdf@adventureworks.msft** contains a link back to the Reporting Server.
- 23. When you have finished reviewing the email files, close any open files and disconnect from **10990C-MIA-CLI**.

24. On **10990C-MIA-SQL**, close Internet Explorer and Microsoft SQL Server Management Studio 2017 without saving any changes.

Check Your Knowledge

Question	
Which of the following is an element of a data-driven subscription that you can configure dynamically?	
Select the correct answer.	
<input type="checkbox"/>	Report name
<input type="checkbox"/>	Report data source
<input type="checkbox"/>	Delivery extension
<input type="checkbox"/>	Rendering format
<input type="checkbox"/>	Subscriber data query

Lab: Sharing Reporting Services reports

Scenario

You are planning to deliver a report to users who utilize a subscription that writes to a file share. To facilitate this process, you decide to create a shared schedule and enable caching for the report.

Objectives

At the end of this lab, you should be able to:

- Create a shared schedule.
- Configure caching.
- Create a standard subscription.

Estimated Time: 60 minutes

Virtual machine: **10990C-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa55w.rd**

Exercise 1: Create a shared schedule

Scenario

In this exercise, you will create a shared schedule that runs every two minutes.

The main tasks for this exercise are as follows:

1. Prepare the lab environment
2. Create a shared schedule

► Task 1: Prepare the lab environment

1. Ensure the **10990C-MIA-CLI**, **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. Run **Setup.cmd** in the **D:\Labfiles\Lab07\Starter** folder as Administrator.

► Task 2: Create a shared schedule

- Using the Reporting Services web portal at **http://mia-sql/reports_sql2**, create a new shared schedule with name **Every 2 minutes** that runs every two minutes.

Results: At the end of this exercise, you should have created a shared schedule.

Exercise 2: Configure caching

Scenario

The **Sales_Order_Detail** report is executed many times per minute by different users, and you have identified that this causes a performance problem on the Reporting Services server. You have decided to implement caching on this report by using the shared schedule that you just created.

To enable caching, you must change the data source to use fixed credentials.

The main tasks for this exercise are as follows:

1. Configure execution account
2. Configure caching

► Task 1: Configure execution account

- Using the Reporting Services web portal, change the credentials used by the **AdventureWorks** data source in the **AdventureWorks Sample Reports** folder to use fixed credentials. Use the following credentials:
 - **User name:** SSRS
 - **Password:** Pa55w.rd

► Task 2: Configure caching

- Using the Reporting Services web portal, configure caching for the **Sales_Order_Detail** report in the **AdventureWorks Sample Reports** folder. Configure caching to be triggered by the **Every 2 minutes** schedule that you set up in the first exercise.

Results: At the end of this exercise, you should have configured report caching.

Exercise 3: Subscribe to a report

Scenario

You want to make a history of the **Sales_Order_Detail** report accessible to users who cannot access the Reporting Services web portal. To achieve this, you will set up a subscription to deliver a copy of the report, rendered as an Excel document, to a file share each time a new cache snapshot is created.

The main tasks for this exercise are as follows:

1. Configure the file share
2. Create the subscription
3. Verify the configuration

► Task 1: Configure the file share

- Using File Explorer, share the **D:\Labfiles\Lab07\Starter\Share** folder as **\\mia-sql\share**. Give the **Everyone** group read/write permissions.

► **Task 2: Create the subscription**

- Using the Reporting Services web portal, create a standard subscription to the **Sales_Order_Detail** report that writes an Excel version of the report to \\mia-sql\share each time the cache snapshot is updated, incrementing the file name on each run. Use the following properties:
 - **Subscription name:** Write to Excel
 - **Credentials to access the file share:**
 - **User Name:** adventureworks\ExecutionAccount
 - **Password:** Pa55w.rd

► **Task 3: Verify the configuration**

1. Log on to **10990C-MIA-CLI** as **Student** with the password **Pa55w.rd**.
2. Using File Explorer, connect to \\mia-sql\share using **adventureworks\student** and the password **Pa55w.rd**, and verify that copies of the report are being placed in the shared folder. Close File Explorer when you have finished.
3. On **10990C-MIA-SQL**, close all open windows without saving any changes.

Results: At the end of this lab, you should have created a subscription that writes copies of a report to a file share.

Question: What changes would you make to amend this subscription to generate the output file in both Excel and PDF format?

Module Review and Takeaways

In this module, you have learned about the different types of schedule used in Reporting Services, and how to use schedules to control caching, snapshots, and subscriptions. You have also learned about adding comments to reports through the web portal.

Review Question(s)

Question: How might you use caching and subscriptions in your organization?

MCT USE ONLY. STUDENT USE PROHIBITED

Module 8

Administering Reporting Services

Contents:

Module Overview	8-1
Lesson 1: Administering Reporting Services	8-2
Lesson 2: Reporting Services configuration	8-6
Lesson 3: Reporting Services performance	8-12
Lab: Administering Reporting Services	8-18
Module Review and Takeaways	8-22

Module Overview

System administrators take responsibility for the configuration and day-to-day operations of IT systems. In SQL Server® Reporting Services (SSRS), administrative tasks include the configuration of the web portal and web service, branding the web portal, and ensuring that access to sensitive reports is carefully controlled. Administrators also monitor and optimize performance.

Objectives

At the end of this module, you will be able to:

- Secure access to reports by using authentication, permissions, and SSL encryption.
- Use Report Services Configuration Manager to reconfigure SSRS servers.
- Monitor and optimize the performance of SSRS on a given set of server hardware.

Lesson 1

Administering Reporting Services

Reporting Services frequently presents data that should be restricted to a subset of your user base. You might consider data sensitive if, for example, it relates to personal information from your customers or employees; it relates to company performance figures not yet published; or it relates to information that might give your competitors an advantage. It's essential, therefore, to understand how to positively identify users, and then control those users' access to reports. In this lesson, you will see how to secure access to Reporting Services and how to encrypt communications between clients and the server.

Lesson Objectives

At the end of this lesson, you will be able to:

- Understand the methods that Reporting Services administrators use to secure access to sensitive report data.
- Describe and configure the authentication methods that Reporting Services uses to identify users.
- Describe and configure the authorization methods that Reporting Services uses to grant access to reports.
- Configure Secure Sockets Layer (SSL) encryption for Reporting Services.

Introduction to Reporting Services security

SQL Server Reporting Services (SSRS) provides a flexible security model that helps you to apply security settings to protect business data. While the specific details of security models vary between organizations, every reporting solution must address the following fundamental security considerations:

Authentication

Your reporting solution should authenticate users to verify their identity. Authentication usually involves validating user credentials, such as a user name and password, before allowing the user to access the reporting solution. SSRS supports a number of authentication mechanisms that you configure to match the requirements of your organization's application infrastructure.

- Authentication
 - Verify user credentials
- Authorization
 - Enforce user permissions
- Secure communication
 - Encrypt data on the network

Authorization

After verifying the identity of each user, your reporting solution must authorize access to specific reports, data sources, or other objects based on assigned permissions. SSRS supports a role-based authorization model in which you apply permissions to a specific set of roles, and assign users to those roles to authorize access to report items.

Secure communication

When users access a report server, data is transmitted across network connections. In some cases, you should consider using encrypted connections to ensure that data cannot be intercepted and read by a third party.

Authentication

SSRS supports a number of authentication models.

By default, users who connect to a report server are authenticated based on their Microsoft Windows® credentials, but you can edit the RSReportServer.config file to enable any of the following authentication models:

- **RSWindowsNegotiate.** This is the default authentication model for report servers in native mode and authenticates users based on their Windows credentials, using Kerberos where supported. In environments where Kerberos is not supported, this authentication model falls back to Windows NT LAN Manager (NTLM) authentication.
- **RSWindowsNTLM.** This authentication model uses NTLM access tokens to authenticate users, and does not support delegation of credentials (in which the report server impersonates the user to access resources on another server).
- **RSWindowsKerberos.** This authentication model uses a Kerberos ticket to identify a user. The user's credentials can be delegated across multiple servers in the same domain. Kerberos authentication requires specific infrastructure configuration, including registering Service Principle Names (SPNs) for service accounts.
- **RSWindowsBasic.** This authentication model uses HTTP basic authentication to authenticate user access to the report server. Credentials are passed in base64 clear text, so you should ensure that your report server is configured to use an encrypted channel in this model.
- **Custom.** You use a custom model that employs ASP.NET Forms authentication to validate users. At the HTTP layer, requests are treated as anonymous and redirected to your custom module.

- Windows-based authentication is used by default
- Specify authentication types in RSReportServer.config

Authentication Type	Description
RSWindowsNegotiate	Attempts to use Kerberos, and falls back to NTLM if a Kerberos ticket cannot be granted
RSWindowsNTLM	Uses Windows integrated authentication based on an NTLM access token with no support for delegation of credentials
RSWindowsKerberos	Uses Windows integrated authentication with a Kerberos ticket—delegation of credentials is possible
RSWindowsBasic	Windows credentials are passed from the client to the report server in base64 encoding
Custom	All requests are authenticated as anonymous at the HTTP layer, and then directed to a custom ASP.NET Forms authentication module

You enable any of these authentication modes by adding tags to the RSReportServer.config file, which is in XML format.

In the following XML code, The **RSWindowsNegotiate** mode is enabled in the configuration file:

Enabling Windows Integrated Authentication

```
<Authentication>
  <AuthenticationTypes>
    <RSWindowsNegotiate />
  </AuthenticationTypes>
  <EnableAuthPersistence>true</EnableAuthPersistence>
</Authentication>
```

For more information about configuring authentication, see:

 **Authentication with the Report Server**

<https://aka.ms/Esj43r>

Authorization

Access to reporting items such as reports, data sources, datasets, and report parts is based on permissions that are assigned to roles. You use system-level roles to control access to the report server and server-level functionality, such as shared schedules.

Item-level roles are used to control access to individual report items. Reporting Services includes the following predefined roles:

- System-level roles:
 - **System User.** Members access the report server.
 - **System Administrator.** Members administer the report server.
- Item-level roles:
 - **Content Manager.** Members have full control of items, including the ability to manage permissions.
 - **Publisher.** Members add, update, view, and delete items.
 - **Report Builder.** Members use Report Builder to create and edit items.
 - **Browser.** Members view items and create subscriptions.
 - **My Reports.** Members manage reports in a personal folder named My Reports.

- Add users to roles
- Assign individual permissions only when necessary

Permission	Reporting Services Role
Access the report server	System User
Manage the report server	System Administrator
Full control of items	Content Manager
Add, update, view, and delete items	Publisher
Use Report Builder	Report Builder
View reports	Browser
Use the My Reports folder	My Reports

If necessary, you can customize the individual permissions that are assigned to the predefined roles. Additionally, you can define your own custom roles and assign specific permissions to them.

Item-level role membership is assigned for individual folders and items. By default, role membership assignments are inherited by subfolders—however, you override inherited role memberships by creating custom role assignments at any level in a folder hierarchy.



Note: In earlier versions of SQL Server, when SSRS was installed in SharePoint® integrated mode, access to report items was controlled by using permissions assigned to SharePoint groups. In SQL Server 2017, integration with SharePoint has been simplified and there is no dedicated SharePoint mode. Therefore, SharePoint groups cannot be used to authorize access to reports.

Encrypting communications

Users generally access a report server with a web browser such as Windows Internet Explorer®. When this is performed across the internet or any network where data must be secure while in transit, you can configure the report server to use Secure Sockets Layer (SSL) security to encrypt network communication between the user and the report server.

- Secure Sockets Layer (SSL) is used to encrypt communications between a web client and a web server
- Installation:
 1. Install a server certificate
 2. Bind the certificate to a Reporting Services URL reservation
 - Use Reporting Services Configuration Manager
 - Report Server and Report Manager URLs are bound individually

Configuring SSL security

You use the following procedure to enable SSL for secure network communication:

1. Install a server certificate for SSL communication.

You either obtain a certificate from a trusted certificate-issuing authority, or you issue your own certificates and configure clients to trust them.

2. Bind the certificate to each Reporting Services endpoint for which you want to enable secure communication.

Use Reporting Services Configuration Manager to bind the certificate and specify the port to be used for secure communication. You must bind the Reporting Services Web Service and Report Manager URLs individually.

Reporting Services does not use Internet Information Services (IIS) to handle HTTP requests. If IIS is installed on a server where Reporting Services is configured to use SSL, the W3SVC service must be running.

For more information about using SSL with Reporting Services, see:



Configure SSL Connections on a Native Mode Report Server

<https://aka.ms/Bm57ar>

Check Your Knowledge

Question	
Which of the following authentication methods sends the username and password across the network in plain text?	
Select the correct answer.	
<input type="checkbox"/>	RSWindowsNegotiate
<input type="checkbox"/>	RSWindowsNTLM
<input type="checkbox"/>	RSWindowsKerberos
<input type="checkbox"/>	RSWindowsBasic

Lesson 2

Reporting Services configuration

The configuration of your SSRS server is vital for maintaining a functional and efficient reporting environment. For example, when systems administrators alter email servers, you must update the names of the servers in SSRS so that reports can continue to be emailed to stakeholders. In this lesson, you will see how to use Reporting Services Configuration Manager and SQL Server Management Studio 2017 to configure SSRS. You will also see how to apply a set of colors and a logo to the Reporting Services web portal.

Lesson Objectives

At the end of this lesson, you will be able to:

- Use the Reporting Services Configuration Manager to configure an instance of SSRS.
- Control how users print reports and store custom reports.
- Match the Reporting Services web portal to your corporate branding.

Reporting Services Configuration Manager

Reporting Services Configuration Manager is a graphical user interface (GUI) tool that you use to perform a wide range of administrative tasks on any SSRS server.

If you perform a file-only installation of SSRS on a server, you must use Reporting Services Configuration Manager to configure that installation before you use it. If you used the default installation option, you can use Reporting Services Configuration Manager to alter the settings you chose during installation.

Reporting Services Configuration Manager is installed automatically whenever you install SSRS. You use Reporting Services Configuration Manager to connect to and configure the local SSRS server, or any other server that you can reach over the network.

- Common tasks
 - Configure the service account
 - Configure the web service URL
 - Configure the web portal URL
 - Configure the Reporting Services database
 - Configure email settings
 - Configure the unattended execution account
 - Manage the symmetric encryption key
 - Configure a scale-out topology
- Requirements
 - Local connections
 - Remote connections

Common tasks

You use Reporting Services Configuration Manager to perform the following tasks. These are shown in the navigation on the left of the tool:

- **Configure the service account.** This is the security context that SSRS runs under on the server.
- **Configure the web service URL.** This is the location of the Report Server web service that publishes reports in the form of XML with a SOAP API. Reporting services clients use this web service to access report data and present it in their own formats to users.
- **Configure the web portal URL.** This is the location of the Report Server portal website that users connect to using their browser, and use to view reports. Administrators might also use the portal for certain configuration tasks.
- **Configure the Reporting Services database.** SSRS uses a SQL Server database to store all its configuration information, including the definitions of all reports. You use Reporting Services Configuration Manager to set the location of this database.

- **Configure email settings.** SSRS delivers reports to stakeholders automatically in email messages. To do this, SSRS requires a Simple Mail Transfer Protocol (SMTP) server, such as an Exchange server. You use Reporting Services Configuration Manager to set the location of this server.
- **Configure the unattended execution account.** This is the account SSRS uses to make remote connections during scheduled report processing. This might be for sending queries to remote databases or to access external image files.
- **Manage the symmetric encryption key.** This key is used to encrypt stored connection strings and credentials. If you lose this key, you must create those strings and credentials again—so it's important to back up and protect the key.
- **Configure a scale-out topology.** A scale-out topology is one in which multiple instances of SSRS share a Reporting Services database. Such topologies permit greater capacity and increase the resilience of the system to server failures.

Reporting Services Configuration Manager cannot be used to define reports or to grant access to the server.

Requirements


Reporting Services Configuration Manager is version-specific, so you must use the 2017 version of this tool to administer SQL Server 2017 SSRS servers. If your SSRS servers have different version numbers, you must install multiple versions of the configuration manager to administer them.

To use Reporting Services Configuration Manager to manage SSRS on the local server, you must be using an account that has administrator privileges on that server. In addition, you must have permission to create databases on the instance of SQL Server where the Reporting Services database is stored.

Reporting Services Configuration Manager uses Windows Management Instrumentation (WMI) to connect to a remote SSRS server. For this reason, to administer a remote server, you must enable WMI calls to pass through any firewall that separates your local computer from the SSRS server. In addition, if you want to enable a user without administrator permissions on the remote machine to configure SSRS, you must set DCOM and WMI permissions. For more information, see:

Configure a Report Server for remote administration

<https://aka.ms/Cjg0uv>

 **Note:** For SSRS 2012 and later releases, Reporting Services Configuration Manager is not designed to administer a server in SharePoint integrated mode. Instead, you use properties in the SharePoint Central Administration website and PowerShell cmdlets for administration. In SSRS 2017, there is no SharePoint integrated mode because integration with SharePoint has been simplified.

Controlling rendering and print options

Administrators might wish to control how users view and disseminate reports. You choose whether to enable users to print reports by using the report viewer—or whether to enable My Reports.

Controlling client-side printing

The report viewer toolbar includes a **Print** button that creates a Portable Document Format (PDF) file that users send to a printer from their browser. This PDF file is created by the PDF rendering extension that is included with SSRS. Any client-side desktop application—for example, Microsoft Edge—can display and print this PDF. No extra software component is required on the client computer.

- Controlling client-side reports
 - **EnableClientPrinting** property
- Controlling My Reports
 - **Enable a My Reports folder for each user** property

You might wish to prevent users from printing reports—for example, to save paper or to prevent old data from being circulated as hard copies. To prevent printing:

1. On the SSRS server, start SQL Server Management Studio 2017 (SSMS) with administrative privileges.
2. Connect to the local SSRS server.
3. In Object Explorer, right-click the report server, and then click **Properties**.
4. Under **Select a Page** click **Advanced**.
5. Set the **EnableClientPrinting** property to **False**, and then click **OK**.

For more information, see:



Enable and Disable Client-Side Printing for Reporting Services

<https://aka.ms/Qiqh8q>

Controlling My Reports

The SSRS My Reports feature provides a dedicated private folder in the Reporting Services database for each user who creates and stores their own reports. Many organizations find this feature useful because it enables users to create and manage reports for their own role, which might not be useful to anyone else.

My Reports is disabled by default. To enable it:

1. On the SSRS server, start SSMS with administrative privileges.
2. Connect to the level SSRS server.
3. In Object Explorer, right-click the report server, and then click **Properties**.
4. On the **General** tab, select **Enable a My Reports folder for each user**, and then click **OK**.

For more information, see:



Enable and Disable My Reports

<https://aka.ms/Osola6>

Branding the web portal

In SSRS 2017, it's easy to modify the Reporting Services web portal to reflect your own corporate branding, logos and colors. This is done by creating a set of files called a brand package, and then uploading them to the portal.

A brand package consists of the following files:

- **colors.json**. This file fixes the colors that will be presented to web portal visitors in JavaScript Object Notation (JSON) format.
- **logo.png**. This image file will be used as the main site logo.
- **metadata.xml**. This file fixes the name of the branding package and links to the previous two files.

- metadata.xml
 - Name the brand package
 - Link to colors.json and Logo.png
- colors.json
 - Set colors by using JavaScript Object Notation (JSON)
- logo.png
 - Logo to be displayed in the upper left of the portal
 - 290 x 60 pixels

You must place these files in a zip file and upload that to the portal.

The following example creates a brand package named Wide World Importers:

Metadata.xml example

```
<?xml version="1.0" encoding="utf-8"?>
<SystemResourcePackage

xmlns="http://schemas.microsoft.com/sqlserver/reporting/2016/01/systemresourcepackagemeta
data"
  type="UniversalBrand"
  version="2.0.2"
  name="Wide World Importers" >
  <Contents>
    <Item key="colors" path="colors.json" />
    <Item key="logo" path="logo.png" />
  </Contents>
</SystemResourcePackage>
```

In the colors.json file, you use JSON text to specify colors for various controls in the portal. Properties you can use include colors for KPIs, backgrounds, fonts, and buttons. For example, you might set the background color of primary buttons in the portal by using the **primary** property.

The following code from the colors.json file sets the background color and foreground color of buttons in the portal:

Example properties from colors.json

```
"primary": "#999999",
"primaryContrast": "#ffffff"
```

You can obtain six-digit hexadecimal color numbers from stylesheets that are already in use in your business, such as the style sheets for your intranet or internet websites. For a complete colors.json file example, see:

 **Branding the web portal**

<https://aka.ms/A24w6f>

Logo.png is an image file you might use for your company logo. It's displayed at the upper left of a branded Reporting Services web portal. The image is automatically scaled to a size of 290 x 60 pixels, so you should choose an image with those proportions.

When you have created these files and added them to a zip file, use the following steps to upload and apply the brand package:

1. Log on to the web portal as an administrator.
2. Click the gear icon in the upper right of the page, and then click **Site Settings**.
3. In the navigation on the left, click **Branding**.
4. Click **Upload Brand Package**, and then browse to the zip file. The package will be applied to the portal as soon as the upload is complete.

Demonstration: Using the Report Server Configuration Manager to apply branding

In this demonstration, you will see how to:

- Use the Report Server Configuration Manager to set properties for the web portal.
- Apply a brand package to the web portal.

Demonstration Steps

Prepare the environment

1. Ensure that both **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Demofiles\Mod08** folder, run **Setup.cmd** as Administrator. Click **Yes** when prompted to confirm that you want to run the command file, and wait for the script to finish.

Configuring the web portal

1. Start Report Server Configuration Manager.
2. In the **User Account Control** dialog box, click **Yes**.
3. In **The Report Server Configuration Connection** dialog box, click **Connect**.
4. In the hierarchy on the left, click **Web Portal URL**. Note that the virtual directory is **reports_sql2** because the reporting database is on the SQL2 instance of SQL Server.
5. In the **Virtual Directory** text box, type **reports** and then click **Apply**. Wait for the process to complete. This may take some time.
6. When the **Results** pane shows that the task has completed, click **http://MIA-SQL:80/reports**. Internet Explorer displays the Report Server portal at the new URL.



Note: If the portal fails to load correctly refresh the page in the browser.

Apply a brand package to the web portal

1. In Internet Explorer, in the SQL Server Reporting Services web portal, click the cog icon, and then click **Site Settings**.
2. In the navigation on the left, click **Branding** and then click **Upload brand package**.
3. Browse to the **D:\Demofiles\Mod08** folder, click the **AdWorksBrand.zip** file, and then click **Open**. When the file is uploaded, Report Services applies the new color scheme.
4. In the top-left, click **Browse** to return to the home page.

Revert the web portal changes

1. In Report Server Configuration Manager, on the left click **Web Portal URL**.
2. In the **Virtual Directory** text box, type **reports_sql2**, and then click **Apply**.
3. Close the Report Server Configuration Manager.
4. Close Internet Explorer.

Check Your Knowledge

Question	
Which of the following files should not be included in a brand package for the SSRS web portal?	
Select the correct answer.	
<input type="checkbox"/>	RSReportServer.config
<input type="checkbox"/>	logo.png
<input type="checkbox"/>	metadata.xml
<input type="checkbox"/>	colors.json

Lesson 3

Reporting Services performance

Preparing and rendering reports is a complex task that requires CPU cycles and memory use on SSRS servers. If you have many users or create large and complex reports, this might place large loads on your server hardware. If this causes reports to be delivered slowly, this might negatively affect users' productivity. It is the administrator's responsibility to ensure that reports are delivered in a timely fashion by configuring servers for maximum performance. In this lesson, you will learn how to optimize performance and diagnose bottlenecks that might slow report delivery.

Lesson Objectives

At the end of this lesson, you will be able to:

- Describe the general principles that administrators might follow to optimize SSRS performance.
- Use the Performance Console to diagnose a performance bottleneck.
- Use the Reporting Services execution log to investigate report performance.
- Use the Reporting Services HTTP log to investigate report performance.
- Locate information about Reporting Services in the Windows application log.

Optimizing performance

Users don't like waiting for a report to render in their browser or client application. However, many factors combine to slow the execution of reports—these include the current demand for reports, the complexity of those reports, the speed of network connections to database servers, and so on. Reporting Services administrators are not responsible for all these factors but there are many things you might do to accelerate report delivery and optimize the use of your hardware resources.

- Performance guidelines
 - Memory
 - Separate server roles
 - Scale-out servers
 - Tune reports
- Comparing performance against baselines
 - Spotting trends in demand
 - Measuring the impact of optimizations

Performance guidelines

Follow these general principals to optimize performance:

- **Memory.** The processes of report processing and rendering are memory intensive, and the price of server memory continues to fall. Therefore, it's cost-effective to ensure that an SSRS server has significant memory installed.
- **Separate server roles.** Install SSRS on a different server than the reporting database then optimize your hardware investment by designing servers specifically for their roles. This approach tends to provide higher performance at a lower cost.
- **Scale-out servers.** If all reports are running slowly, you can install multiple SSRS servers that use the same reporting database. By using a load balancing system, such as Windows Load Balancing, to share load between front-end servers, you significantly increase the capacity of your system. This approach also increases the system's resilience to server failures.
- **Tune reports.** If a single report executes slowly, examine and optimize the queries that report makes against data sources. Use snapshots and caching to optimize the delivery of large reports.

Comparing performance against baselines

In the following topics, you will learn about tools that you use to measure the performance of your SSRS server. This performance might alter over time because of changes in user demand or because of any optimizations you make. To understand performance changes, you need to measure a baseline performance profile.

To create a baseline performance profile, measure a range of general-purpose counters over a typical time period, such as a working day. Subsequently, make regular measurements of the same counters and compare them to the baseline, to understand if demand is increasing or delivery times are slowing. By regularly monitoring against a baseline in this way, you spot developing performance bottlenecks and remove them before they have a significant impact on users.

Similarly, make a baseline performance profile before you make a configuration change that you think will increase performance. Measure a second profile after you make the change and compare the two profiles to determine an empirical measurement of the impact of your change. Use this data to show budget holders that the change is effective.

Performance Monitor counters

Use the Performance Console on any Windows server to collect and analyze performance data. Use the Performance Monitor tool within the console to display the current value of counters and how they are changing. You also use data collector sets to record the value of counters over an extended time period for later examination and to compare against a baseline.

You add counters from the local computer or any remote computer that you can connect to across the network. Counters in Performance Console are organized into performance objects. For example, the **% Processor Time** counter is found in the **Processor** object. Sometimes you must also select an instance of the object. For example, for the **% Processor Time** counter, you choose a specific processor core or choose **<All Instances>** to see how the counter averages across all cores.

- Windows performance objects
 - Processor
 - Memory
 - Process
 - .NET CLR data
- Reporting Services performance objects
 - MSRS 2011 Web Service
 - MSRS 2011 Windows Service
 - ReportServer:Service

Some common general-purpose counters, present on all Windows computers, include:

- **Processor/%Processor Time.** If this counter persistently shows more than 80% for long periods, consider upgrading your processors.
- **Memory/Available MBytes.** This counter shows the volume of physical memory that's available for use by processes.
- **Memory/Page Faults/sec.** A page fault occurs when a process requests some data that has been moved to the paging file on the hard disk. If this counter is often more than 5, consider adding physical memory.
- **Process/Elapsed Time.** Shows how long a process, such as **ReportingServicesService**, has been running.
- **.NET CLR Data.** Because SSRS uses the .NET runtime, counters in this object can help to diagnose SSRS performance bottlenecks.

When you install SSRS, some specialized objects and counters are added to the Performance Console. The objects include:

- **MSRS 2011 Web Service.** You use the counters in this object to measure the performance of reports that were initiated interactively in the web portal or a client application.
- **MSRS 2011 Windows Service.** You use the counters in this object to measure the performance of reports that were initiated through scheduled operations, such as snapshots and report subscriptions.
- **ReportServer:Service.** You use the counters in this object to measure requests, connections, login attempts and other HTTP events. There are also counters in this object that relate to memory management for the SSRS instance.



Note: In earlier versions of SSRS, there were also performance objects for SharePoint integrated mode. For example, instead of **MRSS 2011 Web Service**, in a SharePoint integrated instance of SSRS, you would use the **MRSS 2011 SharePoint Mode Web Service** object and its counters. In SQL Server 2017, there is no SharePoint integrated mode.

Report Server execution log

In SSRS, the Report Server execution log records information about all the reports that execute on a server or on multiple servers in a scale-out deployment. You use the execution log to find out:

- How often a report is requested.
- What report formats are requested.
- How long it takes to execute the queries necessary to build a report's dataset.
- How long it takes to process the dataset.

- Use the execution log to determine:
 - How often a report is requested
 - What report formats are requested
 - How long it takes to execute queries
 - How long it takes to process data
- Enable the execution log by using SQL Server Management Studio
- Query the ExecutionLog3 view:

```
USE ReportServer
SELECT * FROM ExecutionLog3
ORDER BY TimeStart DESC;
GO
```

You use this information to identify those reports that take a long time to complete and suggest to report authors how they might reduce delays in report delivery.

The report execution log information is stored as an internal table in the reporting services database. You investigate the data by executing Transact-SQL queries against the ExecutionLog3 view.



Best Practice: There are three views for the execution log: ExecutionLog, ExecutionLog2, and ExecutionLog3. ExecutionLog3 was added most recently and includes more fields and friendlier names. Use ExecutionLog3 unless you have older queries that depend on the previous views.

To enable execution logging, complete the following steps:

1. Start SQL Server Management Studio 2017 as an administrator.
2. Connect to the instance of SSRS that you want to configure.
3. In Object Explorer, right-click the server, and then click **Properties**.
4. Click the **Logging** page.
5. Select **Enable report execution logging**, and then click **OK**.

The following Transact-SQL query displays all information in the execution log:

Examining the execution log

```
USE ReportServer
SELECT * FROM ExecutionLog3 ORDER BY TimeStart DESC;
GO
```

For more information about the execution log, including a list of view columns, see:



Report Server ExecutionLog and the ExecutionLog3 View

<https://aka.ms/Djwifk>

Report Server HTTP log

Users utilize a web browser to connect to the Reporting Services web portal and view reports. Alternatively, they might use a client application that communicates with the Reporting Services web service and communicates by using SOAP messages. Both the web portal and the web service communicate by using HTTP. SSRS records this activity in the HTTP log file on the hard disk.

This log includes some events at the HTTP level that do not appear in the execution log—for example, timeout errors and request overflow errors.

- The HTTP log includes some events that do not reach the execution log
- Enabling and configuring the HTTP log
 - ReportingServicesService.exe.config

```
<RStrace>
  <add name="FileSizeLimitMb" value="32" />
  <add name="KeepFilesForDays" value="14" />
  <add name="HttpTraceFileName"
        value="ReportServerService_HTTP_" />
  <add name="Components" value="all:3,http:4" />
</RStrace>
```

- File format and retention

Enabling and configuring the HTTP log

HTTP logging is not enabled by default—to enable and configure it, you must edit the following XML file:

\Program Files\Microsoft SQL Server\MSSQL.n\Reporting Services\ReportServer\Bin\ReportingServicesService.exe.config

To enable logging, you must add “http:4” to the <RStrace> section of the file.

The following XML from the ReportingServicesService.exe.config file enables the HTTP log and sets the file name and file retention options:

Configuring the HTTP log

```
<RStrace>
  <add name="FileSizeLimitMb" value="32" />
  <add name="KeepFilesForDays" value="14" />
  <add name="HttpTraceFileName" value="ReportServerService_HTTP_" />
  <add name="Components" value="all:3,http:4" />
</RStrace>
```

When you have enabled HTTP logging, you must restart the Reporting Services service. When the service receives the first request, the log file is created in the following location:

\Program Files\Microsoft SQL Server\<SQL Server Instance>\Reporting Services\LogFiles

The log is a text file that has ASCII characters like the W3C log files that Internet Information Services (IIS) uses. By default, log files are limited to 32 MB and are deleted after 14 days but you can configure retention values in the ReportingServicesService.exe.config file, as in the preceding code example.

For more information about the HTTP log file, see:



Report Server HTTP Log

<https://aka.ms/l8u2jk>

Windows Event Log

Reporting Services also logs information to the Windows Event Log. Messages are logged to the application log and viewed by using Event Viewer, in the same way as you view events from other applications and services installed on Windows.

Reporting Services event logging is always turned on—you cannot modify the schema of events or the events that are logged. The application log typically contains many thousands of events from lots of sources. To use the log effectively to diagnose problems, it's essential to use the tools provided to find or filter events to focus on the ones that interest you.

- Reporting Services logs information to the Windows Event Log
- You cannot configure the events or information recorded
- Reporting Services event sources:
 - Report Service (Report Server Windows Service)
 - Report Manager
 - Scheduling and Delivery Processor

If an event is logged by Reporting Services, it will be from one of three event sources:

- Report Service (Report Server Windows Service)
- Report Manager
- Scheduling and Delivery Processor

For a complete list of the events that Reporting Services generates in the application log, see:



Errors and Events Reference (Reporting Services)

<https://aka.ms/C1m9po>

Demonstration: Configure and query the execution log

In this demonstration, you will see how to configure the SSRS execution log and how to execute a query against the execution log.

Demonstration Steps

Configure the execution log

1. Start SQL Server Management Studio 2017.
2. In the **Connect to Server** dialog box, in the **Server type** drop-down list, select **Reporting Services**.
3. In the **Server name** drop-down list, select **MIA-SQL\SSRS**, and then click **Connect**.
4. In the Object Explorer, right-click **MIA-SQL\SSRS** and then click **Properties**.
5. In the **Select a page** list, click **Logging**. Note that execution logging is enabled by default.
6. In the **Remove log entries older than this number of days** text box, type **10**, and then click **OK**.
7. In the Object Explorer, right-click **MIA-SQL\SSRS** and then click **Disconnect**.

Query the execution log

1. In SQL Server Management Studio 2017, on the **File** menu, click **Connect Object Explorer**.
2. In the **Connect to Server** dialog box, in the **Server type** drop-down list, select **Database Engine**.
3. In the **Server name** drop-down list, select **MIA-SQL\SQL2** and then click **Connect**.
4. In the Object Explorer, expand **Databases**.
5. Right-click the **ReportServer** database, and then click **New Query**.
6. Type the following query and then press F5:

```
SELECT * FROM ExecutionLog3 ORDER BY TimeStart DESC;
```

Examine the results.

7. Open Windows Explorer and browse to the folder **D:\Demofiles\Mod08\ReportDesigner**.
8. Double-click the **Reports.sln** file.
9. If you get **How do you want to open this file** window select **Microsoft Visual Studio Version Selector** and then click **OK**.
10. In Visual Studio 2015, on the **Build** menu, click **Build Solution**.
11. On the **Build** menu, click **Deploy Solution**.
12. Open Internet Explorer and then browse to **http://mia-sql/reports_sql2**.
13. Under **Folders** click **Designer**. If the **Designer** folder isn't listed, refresh the web page.
14. Under **Paginated Reports** click **Product Listing**.
15. When the report has loaded, switch to SQL Server Management Studio 2017.
16. On the **Query** menu, click **Execute**. Examine the results.

Check Your Knowledge

Question	
You have enabled Reporting Services execution logging. You want to find out how long, on average, a particular report takes to render. Where should you search for this information?	
Select the correct answer.	
<input type="checkbox"/>	The Windows system log
<input type="checkbox"/>	The Windows application log
<input type="checkbox"/>	Log files on the SSRS server hard disk
<input type="checkbox"/>	The Reporting Services database

Lab: Administering Reporting Services

Scenario

In preparation for using SSRS in production, you want to create user accounts and security groups that you will use to secure access to SSRS reports. You also want to customize the web portal to include a company logo and company colors.

Objectives

At the end of this lab, you will be able to:

- Secure access to folders within the SSRS web portal, by using Reporting Services permissions.
- Apply a branding package to the SSRS web portal.

Estimated Time: 60 minutes

Virtual machine: **10990C-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa55w.rd**

Exercise 1: Authorize access to reports

Scenario

You have been asked to begin implementing groups and permissions for the Adventure Works reporting system. You want to implement permissions for two groups of users:

- **Sales.** Members of this group should have their own Sales Reports folder, in which they create reports and browse other reports.
- **Directors.** Members of this group should have read access to the Sales Reports folder. They should also have their own Directors Reports folder where they create and browse reports.

Because the Directors Reports folder might contain highly sensitive information, sales staff should have no access to it.

The main tasks for this exercise are as follows:

1. Prepare the lab environment
2. Create reporting folders
3. Create Active Directory groups and users
4. Assign permissions to groups
5. Test permissions

► Task 1: Prepare the lab environment

1. Ensure that the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab08\Starter** folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes** to run the command file, and then wait for the script to finish.

► **Task 2: Create reporting folders**

1. Use Internet Explorer to browse the Reporting Services web portal at **http://mia-sql/reports_sql2**.
2. In the Reporting Services **Home** page, create two folders called **Sales Reports** and **Directors Reports**.

► **Task 3: Create Active Directory groups and users**

1. Switch to the **10990C-MIA-DC** virtual machine and log on as **ADVENTUREWORKS\Administrator** with the password **Pa55w.rd**.
2. Use Active Directory Users and Computers to create the following user in the **adventureworks.msft** domain:
 - **First name:** Sophia
 - **Last name:** Garner
 - **User login name:** Sophia@adventureworks.msft
 - **Password:** Pa55w.rd
 - Ensure that the user never needs to change the password.
3. Use Active Directory Users and Computers to create the following user in the **adventureworks.msft** domain:
 - **First name:** Rodrigo
 - **Last name:** Romani
 - **User login name:** Rodrigo@adventureworks.msft
 - **Password:** Pa55w.rd
 - Ensure that the user never needs to change the password.
4. Create a new global security group named **Directors** and add **Sophia Garner** to the new group.
5. Create a new global security group named **Sales** and add **Rodrigo Romani** to the new group.

► **Task 4: Assign permissions to groups**

1. Switch to the **10990C-MIA-SQL** virtual machine and browse to **http://mia-sql/reports_sql2**.
2. Customize the security of the **Directors Reports** folder and assign to the **Directors** Active Directory group **Browser** and **Publisher** permissions.
3. Customize the security of the **Sales Reports** folder and assign to the **Sales** Active Directory group **Browser** and **Publisher** permissions.
4. Also assign **Browser** permissions to the **Directors** group for the **Sales Reports** folder.
5. Assign **Browser** permissions to the **Home** folder for both **Directors** and **Sales** groups.
6. Sign out of **10990C-MIA-SQL**.

► **Task 5: Test permissions**

1. Log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Sophia** with the password **Pa55w.rd**. Use the Reporting Services web portal to explore what the user is permitted to do in the **Directors Reports** and **Sales Reports** folders.
2. Add a new folder, **Test Folder**, to **Directors Reports**.

3. Log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Rodrigo** with the password **Pa55w.rd**. Use the Reporting Services web portal to explore what the user is permitted to do in the Directors Reports and Sales Reports folders.
4. Add a new folder, **Test Folder**, to **Sales Reports**.
5. Close Internet Explorer and log off **10990C-MIA-SQL**.

Results: At the end of this exercise, you will have created and authorized access to a set of folders within the Reporting Services web portal.

Exercise 2: Web portal branding

Scenario

You have been asked to apply company colors and a logo to the Reporting Services web portal. You have been given a brand package, which you must complete, compress, and upload to the portal.

The main tasks for this exercise are as follows:

1. Complete the brand package
2. Upload and use the brand package

► Task 1: Complete the brand package

1. Log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab08\Starter\BrandPackage\metadata.xml** file, add a **name** attribute to the **<SystemResourcePackage>** tag with the value "Adventure Works".
3. Add the following tag to the **<Contents>** section:
 - Tag: **<Item>**
 - key attribute: "colors"
 - path attribute: "colors.json"
4. Add the following tag to the **<Contents>** section:
 - Tag: **<Item>**
 - key attribute: "logo"
 - path attribute: "AW-Logo.png"
5. Zip the following three files into a compressed file:
 - metadata.xml
 - logo.png
 - colors.json

► Task 2: Upload and use the brand package

1. In Internet Explorer, browse to the Report Services web portal and then, in Site Settings, upload the following brand package: **D:\Labfiles\Lab08\Starter\BrandPackage\AdworksBrand.zip**.
2. Examine the new colors and logo.
3. Close all open windows.

Results: At the end of this exercise, you will have modified the colors and logo for the Reporting Services web portal.

Question: In the lab, you secured access to the Directors Reports folder by using permissions. How else might malicious users gain access to these reports and how do you protect against such attacks?

Module Review and Takeaways

In this module, you have seen how administrators secure and configure SSRS to provide the most appropriate behavior for users. You have also seen how administrators identify performance bottlenecks and remove them to ensure an optimal user experience.

Review Question(s)

Question: Your users report that they often receive timeout errors when they execute reports. Where should you look for more information?

Module 9

Extending and Integrating Reporting Services

Contents:

Module Overview	9-1
Lesson 1: Expressions and embedded code	9-2
Lesson 2: Extending Reporting Services	9-10
Lesson 3: Integrating Reporting Services	9-18
Lab: Extending and integrating Reporting Services	9-26
Module Review and Takeaways	9-31

Module Overview

Although Reporting Services is a powerful tool, its built-in capabilities might not always meet your needs. This module covers the methods for extending the functionality of Reporting Services with expressions and custom code. You will also learn about the methods for working with Reporting Services programmatically, and integrating Reporting Services reports into other applications.

Objectives

At the end of this module, you should be able to:

- Work with expressions and embedded code.
- Extend Reporting Services functionality with external assemblies.
- Integrate Reporting Services with other software.

Lesson 1

Expressions and embedded code

This lesson introduces expressions and embedded code in Reporting Services reports. You use expressions and embedded code to customize the behavior of your report without importing external assemblies.

Lesson Objectives

At the end of this lesson, you should be able to:

- Describe Reporting Services expressions.
- Explain how to use embedded code in report definitions.

Expressions

You use expressions in Reporting Services paginated reports to control many aspects of a report's appearance. Most properties of most report items can be set using an expression, including properties such as display value, formatting, and visibility.

You write expressions using Visual Basic®; your expressions are stored in the report definition, and are evaluated when you run the report. You write expressions using Report Builder or SQL Server Data Tools (SSDT). Expressions always begin with an equal sign (=).

- Many aspects of a report can be controlled with expressions, including:
 - Data
 - Formatting
- Simple expressions:
 - Reference to a single item from a built-in collection
- Complex expressions
 - Reference to multiple items
 - Constants
 - Operators
 - Built-in collections
 - Built-in functions
 - Custom code and external assemblies

If you are used to working with formulas in Microsoft Excel®, you might be used to working with data directly; in Reporting Services reports, expressions act as placeholders for data. You can only view the result of expressions by running the report, either in preview mode or by publishing it to a Reporting Server.

Although you might not realize it, you have already used expressions extensively in this course. Every time you add a column from a dataset to a paginated report, the column name appears in the report design, enclosed in square brackets. This notation is shorthand for an expression—for example, a column called **ProductId** appears in the report design as **[ProductId]**; this represents the expression **=Fields!ProductId.Value**.

Simple and complex expressions

An expression falls into one of the following two categories:

- Simple
- Complex

Simple expressions take the form of a reference to a single item from a collection—such as a parameter or a dataset field. In design view, simple expressions appear as the item name in square brackets, which represents an expression with the form:

```
=<collection name>!<item name>.value
```

Some examples of simple expressions include:

Collection type	Display text example	Equivalent expression
Dataset field	[ProductId]	=Fields!ProductId.Value
Report parameter	[@OrderId]	=Parameters!OrderId.Value
Built-in field	[&ReportName]	=Globals!ReportName.Value

Simple expressions can also include built-in aggregations—for example:

Collection type	Display text example	Equivalent expression
Dataset field	[SUM(SalesAmount)]	=SUM(Fields!SalesAmount.Value)

Complex expressions contain references to two or more collection items; the items might be from different collections. Like simple expressions, complex expressions are represented in design view by a placeholder name in square brackets.

Complex expressions can include items of the following types:

- **Constants:** hard-coded values of string, integer, or other Visual Basic data types, such as **=1000**.
- **Operators:** arithmetic, comparison, logical, or string operators, such as * (multiply), / (divide), < (less than), & (string concatenation).
- **Built-in collections:** as used in simple expressions (see above).
- **Built-in functions:** built-in report functions, such as **SUM** or **Previous**.
- **References to custom code and external assemblies:** references of this type are covered later in this module.

For more information about working with expressions, see the topic *Expressions (Report Builder and SSRS)* in Microsoft Docs:

Expressions (Report Builder and SSRS)

<https://aka.ms/Fuvj0u>

For more information about constants in complex expressions, see the topic *Constants in Expressions (Report Builder and SSRS)* in Microsoft Docs:

Constants in Expressions (Report Builder and SSRS)

<https://aka.ms/Poy7j8>

For more information about operators in complex expressions, see the topic *Operators in Expressions (Report Builder and SSRS)* in Microsoft Docs:

Operators in Expressions (Report Builder and SSRS)

<https://aka.ms/Otuwhq>

For more information about built-in collections, see the topic *Built-in Collections in Expressions (Report Builder)* in Microsoft Docs:



Built-in Collections in Expressions (Report Builder)

<https://aka.ms/X7wu4o>

For more information about built-in functions, see the topic *Report Builder Functions - Aggregate Functions Reference* in Microsoft Docs:



Report Builder Functions - Aggregate Functions Reference

<https://aka.ms/Jbrm0i>

Embedded code

You can embed custom Visual Basic code in a report definition, and refer to methods in the embedded code in the report design. You might use embedded code to work with a method that is too complex for the expression editor, or when you want to be able to reuse a method several times in a report.

You edit embedded code through the Code tab of the **Report Properties** page. The code block that you embed in your report might include multiple methods.

- A Visual Basic code block—including one or more methods—can be embedded in a report definition
- Reference embedded code using the **Code** global member
- Reference report collections from embedded code using the **Report** global member
- Constants and variables declared in embedded code can be referenced from expressions, using the **Code** global member

References in, and to, embedded code

You reference methods in embedded code through the global **Code** member.

The following example shows how you would reference an embedded code method called **ToBitcoin**—that converts a value in US dollars to a current value in Bitcoin—in a report definition:

Referencing embedded code

```
=Code.ToBitcoin(Fields!OrderTotal.Value)
```

You reference built-in report collections in embedded code through the **Report** object.

The following example shows how you would reference the report title from embedded code:

Referencing built-in collections from embedded code

```
Dim title As String = Report.Globals!ReportName.Value
```

Assembly references

References to the `System.Convert` and `System.Math` classes are automatically included when you use an embedded code block. You use other classes in the `System` namespace by referring to them by full name—for example, **System.Text.StringBuilder()**.

Constants and variables

You declare constants and variables in the embedded code block and reference them from expressions in the report definition by using the global **Code** member. The data type of custom constants is always treated as **Variant** when you reference them in an expression.

The following shows an example of a constant and a variable declared in embedded code, and how you would reference them in the report design:

Constants and variables

```
'In embedded code:
'Declare a constant
Public Const CopyrightHolder = "Adventure Works Cycles, Ltd."

'Declare a variable
Public Dim BaseVersion AS String = "1.0."

'In the report definition:
'Reference a constant
=Code.CopyrightHolder

'Reference a variable
=Code.BaseVersion
```

For more information about working with embedded code, see the section *Including Embedded Code* in the topic *Custom Code and Assembly References in Expressions in Report Designer (SSRS)*, in Microsoft Docs:

 **Custom Code and Assembly References in Expressions in Report Designer (SSRS) – Including Embedded Code**

<https://aka.ms/Vows7h>

Demonstration: Using expressions

In this demonstration, you will see how to work with expressions in Report Designer and Report Builder.

Demonstration Steps

Prepare the environment

1. Ensure that both **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Demofiles\Mod09** folder, run **Setup.cmd** as **Administrator**. Click **Yes** when prompted to confirm that you want to run the command file, and wait for the script to finish.

Expressions in Report Designer

1. Start Visual Studio® 2015, then on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, go to the **D:\Demofiles\Mod09\Demo** folder, click **Demo.sln**, and then click **Open**.
3. In the **Security Warning for Demo** dialog box, clear the **Ask me for every project in this solution** check box, and then click **OK**.
4. In Solution Explorer, expand **Reports** then double-click **Expressions.rdl**.
5. Click **Preview** to preview the report, then click **Design** to return to design mode.

6. Click **[SalesAmount]** in the second row of the report table, then position the cursor over **[SalesAmount]**. Observe that the field expression is displayed.
7. Double-click **[SalesAmount]**.
8. In the **Placeholder Properties** dialog box, on the **General** page, observe that the **Value** box contains the value **[SalesAmount]**.
9. Click the function editor button to the right of the **Value** box.
10. In the **Expression** dialog box, in the **Set expression for: Value** box, observe that the full expression for the **SalesAmount** value is displayed, and then click **Cancel**.
11. In the **Placeholder Properties** dialog box, click **Cancel**.
12. In the report page header, double-click **[&PageNumber]**.
13. In the **Placeholder Properties** dialog box, on the **General** page, click the function editor button to the right of the **Value** box.
14. In the **Expression** dialog box, observe that the full expression for the page number global value is displayed. Edit the **Set expression for: Value** box so that it reads

```
= "Page " & Globals!PageNumber & " of " &
```

15. Leave the cursor at the end of the text you have just typed, then in the **Category** box, click **Built-in Fields**.
16. In the **Item** box, double-click **TotalPages** to add it to the expression. The completed expression should read:

```
= "Page " & Globals!PageNumber & " of " & Globals!TotalPages
```

17. Click **OK**.
18. In the report design view, observe that the placeholder for **[&PageNumber]** now reads **<<Expr>>**.
19. Double-click **<<Expr>>**.
20. In the **Placeholder Properties** dialog box, on the **General** page, in the **Label** box type **Page** then click **OK**.
21. In the report design view, observe that the placeholder now reads **[Page]**.
22. Click **Preview** to preview the report, then click **Design** to return to design mode.
23. Right-click **[SalesAmount]**, then click **Text Box Properties**.
24. In the **Text Box Properties** dialog box, observe that the **General** page repeats the information on the **Placeholder Properties** dialog box.
25. On the **Font** page, click the function button next to the **Color** box.
26. In the **Expression** dialog box, click on several colors in the **Values** box; observe that the expression text changes to the name of each color that you click.
27. Edit the expression box so that it reads, and then click **OK**:

```
= iif(Fields!SalesAmount.Value < 10000, "Red", "Black")
```

28. In the **Text Box Properties** dialog box, click **OK**.

29. Click **Preview** to preview the report and observe that values in the **Sales Amount** column with a value less than 10000 are displayed in red, then click **Design** to return to design mode.

Using the Report Wizard from Report Builder

1. Click **Start**, type **Report Builder** then press Enter.
2. In Report Builder, in the **Getting Started** window, click **Open**.
3. In the **Open Report** dialog box, in the **Name** box, type **D:\Demofiles\Mod09\Builder\Expressions.rdl**, then click **Open**.
4. Click **Run** to preview the report, then click **Design** to return to design mode.
5. Click **[SalesAmount]** in the second row of the report table, then position the cursor over **[SalesAmount]**. Observe that the field expression is displayed.
6. Double-click **[SalesAmount]**.
7. In the **Placeholder Properties** dialog box, on the **General** page, observe that the **Value** box contains the value **[SalesAmount]**.
8. Click the function editor button to the right of the **Value** box.
9. In the **Expression** dialog box, observe that the full expression for the **SalesAmount** value is displayed, and then click **Cancel**.
10. In the **Placeholder Properties** dialog box, click **Cancel**.
11. In the report page header, double-click **[&PageNumber]**.
12. In the **Placeholder Properties** dialog box, on the **General** page, click the function editor button to the right of the **Value** box.
13. In the **Expression** dialog box, observe that the full expression for the page number global value is displayed.
14. Edit the **Set expression for: Value** box so that it reads:

```
= "Page " & Globals!PageNumber & " of " &
```

15. Leave the cursor at the end of the text you have just typed, then in the **Category** box, click **Built-in Fields**.
16. In the **Item** box, double-click **TotalPages** to add it to the expression. The completed expression should read:

```
= "Page " & Globals!PageNumber & " of " & Globals!TotalPages
```

17. Click **OK**.
18. In the **Placeholder Properties** dialog box, click **OK**.
19. In the report design view, observe that the placeholder for the page number now reads **<<Expr>>**.
20. Double-click **<<Expr>>**.
21. In the **Placeholder Properties** dialog box, on the **General** page, in the **Label** box, type **Page** then click **OK**.

In design view, observe that the page number placeholder is now named **[Page]**.

22. Click **Run** to preview the report, then click **Design** to return to design mode.

23. Right-click **[SalesAmount]**, then click **Text Box Properties**.
24. In the **Text Box Properties** dialog box, observe that the **General** page repeats the information on the **Placeholder Properties** dialog box.
25. On the **Font** page, click the function button next to the **Color** box.
26. In the **Expression** dialog box, click on several colors in the **Values** box; observe that the expression text changes to the name of each color that you click.
27. Edit the expression box so that it reads, and then click **OK**:

```
=iif(Fields!SalesAmount.Value < 10000, "Red","Black")
```

28. In the **Text Box Properties** dialog box, click **OK**.
29. Click **Run** to preview the report and observe that values in the **Sales Amount** column with a value less than 10000 are displayed in red, then click **Design** to return to design mode.
30. Leave Report Builder open for the next demonstration.

Categorize Activity

Place each expression into the appropriate category. Indicate your answer by writing the category number to the right of each item.

Items	
1	[CustomerID]
2	=Fields!SalesAmount.Value * 0.1
3	=Fields!CustomerID.Value
4	="Report Date:"
5	[&PageNumber]
6	=DatePart(DateInterval.WeekOfYear, today())
7	=Parameters!Month.Value
8	=FORMAT(Fields!SellStartDate.Value, "MMM-yy")
9	SUM([TaxAmount])
10	= Join(Parameters!MySelection.Value)

Category 1	Category 2
Simple Expressions	Complex Expressions

Lesson 2

Extending Reporting Services

While expressions and embedded code allow you to make many customizations to a report, there might be occasions when you need to extend a report by adding references to external assemblies. You might do this either to incorporate functionality provided by third parties, or custom assemblies that you write yourself. This lesson covers the techniques that you use to work with external assemblies in Reporting Services.

Lesson Objectives


At the end of this lesson, you should be able to:

- Use assembly references in reports.
- Work with custom assemblies.
- Describe methods to extend Reporting Services functionality.

Assembly references

You add references to assemblies to your report definition in the following two ways:

- With the **References** tab of the **Report Properties** in Report Designer.
- By manually editing the report definition file.

 **Note:** Although you edit reports that use references to external assemblies with Report Builder, you cannot locally preview reports with external assembly references.

- Add assembly references:
 - Through report properties
 - By editing the **CodeModules** element of the report definition
- Add class instances
 - Through report properties
 - By editing the **Classes** element of the report definition

You manually add assembly references to the report definition file by editing the **CodeModules** element.

The following example shows a fragment of a report definition file that adds a reference to the **CurrencyConverter** assembly to the report:

CodeModules

```
<CodeModules>
  <CodeModule>CurrencyConverter, Version=1.0.62.311, Culture=neutral,
  PublicKeyToken=null</CodeModule>
</CodeModules>
```

If your class contains instance methods (as opposed to static methods), you can instantiate an instance of the class for use in the report in the following two ways:

- With the **References** tab of the **Report Properties** in Report Designer.
- By manually editing the report definition file.

You manually add class instances to the report definition file by editing the **Classes** element:

The following example shows a fragment of a report definition file that adds an instance of the **BitCoinConverter** class called **myBitcoinConverter** from the **CurrencyConverter** assembly to the report:

Class instances

```
<Classes>
  <Class>
    <ClassName>CurrencyConverter.BitCoinConverter</ClassName>
    <InstanceName>myBitcoinConverter</InstanceName>
  </Class>
</Classes>
```



Note: When you create an instance of a class, you must use the fully qualified class name, including the namespace.

For more information about adding assembly references to a report, see the topic *Referencing Assemblies in an RDL File* in Microsoft Docs:



Referencing Assemblies in an RDL File

<https://aka.ms/Drve7h>

Custom assemblies

Using custom assembly methods

The syntax you use to call methods from custom assemblies in report expressions varies according to the method type.

You call static methods by calling them with a fully qualified name.

In the following example, the expression calls the static **CelsiusToFahrenheit** method of the **TemperatureConverter** class in the **Converters** namespace:

Calling static methods

```
=Converters.TemperatureConverter.CelsiusToFahrenheit(Fields!TempCelsius.Value)
```

You call instance methods by referencing the class instance you defined when you added the assembly reference to the project. Class instances are members of the **Code** global member.

In the following example, the expression calls the **FromBitcoin** method of the **myBitcoinConverter** class instance:

Calling instance methods

```
=Code.myBitcoinConverter.FromBitcoin(Fields!SalesAmount.Value,"USD")
```

- Accessing assembly methods:
 - Refer to static classes with a fully-qualified name
 - Refer to class instances through the **Code** global member
- Previewing reports that reference assemblies:
 - Deploy a copy of the assembly to the Visual Studio **PrivateAssemblies** folder
- Deploying reports that reference assemblies:
 - Deploy a copy of the assembly to the **ReportServer\bin** folder before deploying the report

For more information about working with custom assembly methods, see the topic *Accessing Custom Assemblies Through Expressions* in Microsoft Docs:



Accessing Custom Assemblies Through Expressions

<https://aka.ms/Ovnr1>

Previewing reports with custom assemblies

To enable the previewing of reports that reference a custom assembly in Report Designer, you must add a copy of the custom assembly to the **PrivateAssemblies** folder of your Visual Studio installation. The location of this folder varies with the version of Visual Studio that you are using, and the path where you have Visual Studio installed.

For Visual Studio 2015, the default location of the **PrivateAssemblies** folder is:

```
C:\Program Files (x86)\Microsoft Visual Studio 14.0\Common7\IDE\PrivateAssemblies
```



Best Practice: Each time you change the assembly, you must update a copy in the **PrivateAssemblies** folder for Report Designer preview to use the latest version of the assembly. If you are developing a custom assembly in parallel with a dependent report, you could consider adding a post-build step to the assembly project to automatically copy the assembly to the **PrivateAssemblies** folder.



Note: You cannot preview reports that use custom assemblies locally in Report Builder. You preview reports that use custom assemblies when connected to a Report Server that has the assembly deployed to it.

Deploying reports with custom assemblies

When you create a report with a reference to a custom assembly, you must deploy the custom assembly to the report server before you can deploy the report. To deploy the assembly, you must copy it to the **ReportServer\bin** folder on the server where the Reporting Services instance is hosted. The location of this folder varies with the version of Reporting Services you have installed, the instance name you selected, and the path where you have installed Reporting Services.

For Reporting Services 2017, the location of the **ReportServer\bin** folder is:

```
C:\Program Files\Microsoft SQL Server Reporting Services\<instance name>\ReportServer\bin
```

For more information on deploying custom assemblies, see the topic *Deploying a Custom Assembly* in Microsoft Docs:



Deploying a Custom Assembly

<https://aka.ms/Vatjui>

Extending functionality

In addition to referencing assemblies to add methods that you use in expressions in Reporting Services reports, you use custom assemblies to add new features to Reporting Services. You might use the following two ways to extend Reporting Services functionality:

- Custom report items
- Extensions

- Custom report items to extend reports:
 - Design-time
 - Runtime
- Extensions, to extend processing:
 - Data processing
 - Delivery
 - Rendering
 - Security

Custom report items

You create custom report items for use at design time, or at runtime. Typically, you will create a design-time component and link it to a runtime component.

A design-time component is a custom item that extends the Report Designer toolbox to add a component that you interact with in the Report Designer design view.

A runtime component is a custom item that appears in the rendered report; a runtime component is passed configuration from a corresponding design-time component.

For more information on working with custom report items, see the topic *Custom Report Items* in Microsoft Docs:



Custom Report Items

<https://aka.ms/LI3I03>

Extensions

Extensions add capabilities to Reporting Services in areas other than report design. You use extensions to add capabilities to Reporting Services in the following areas:

- **Data processing:** you use a data processing extension to retrieve data from a data source. Reporting Services includes data processing capabilities for many common data sources, but you can use an extension to add your own.
- **Delivery:** a delivery extension provides a mechanism to notify users who are subscribed to a report that the report is ready. Reporting Services includes delivery extensions for email and Windows file shares; you use an extension to add other delivery methods.
- **Rendering:** a rendering extension controls the output format of a report. Reporting Services includes rendering extensions for HTML, PDF, Image, XML, Text (including CSV), Excel, and Word. You use an extension to add more rendering formats.
- **Security:** a security extension is used to authenticate and authorize users; by default, Reporting Services authenticates users through Active Directory. You use an extension to authenticate and authorize users by other mechanisms.

For more information on extensions, see the topic *Reporting Services Extension Library* in Microsoft Docs:



Reporting Services Extension Library

<https://aka.ms/Ojckx4>

Demonstration: Using custom assemblies

In this demonstration, you will see how to:

- Add references to a custom assembly to a report.
- Refer to custom assembly methods in report expressions.

Demonstration Steps

Custom assemblies with Report Designer

1. In Visual Studio 2015, in Solution Explorer, double-click **CurrencyConverters.cs** in the **ExampleAssembly** project. Observe that this is an instanced class.
2. In Solution Explorer, double-click **TemperatureConverters.cs** in the **ExampleAssembly** project. Observe that this is a static class.
3. In Solution Explorer, right-click **ExampleAssembly** then click **Build**.
4. In Solution Explorer, double-click **Assemblies.rdl** in the **Demo** project.
5. In the design pane, right-click anywhere in the central pane that is not within the report body, then click **Report Properties**.
6. In the **Report Properties** dialog box, on the **References** page, under **Add or remove assemblies** click **Add**.
7. In the **Add or remove assemblies** box, click the ellipsis button next to the new row.
8. In the **Add Reference** dialog box, on the **Browse** tab, in the **File name** box, type **D:\Demofiles\Mod09\Demo\ExampleAssembly\bin\Debug\ExampleAssembly.dll** then click **OK**.
9. In the **Report Properties** dialog box, click **OK**.
10. Right-click the first text box in the design pane, then click **Expression**.
11. In the **Expression** dialog box, in the **Set expression for: Value** box, type the following, and then click **OK**:


```
=ExampleAssembly.TemperatureConverters.CelsiusToFahrenheit(36.0)
```
12. Click **Preview** to attempt to preview the report. Observe that an error is raised: the preview cannot be rendered because the assembly cannot be located. Click **Design**.
13. In File Explorer, navigate to **D:\Demofiles\Mod09\Demo\ExampleAssembly\bin\Debug**, right-click **ExampleAssembly.dll**, and then click **Copy**.
14. Navigate to **C:\Program Files (x86)\Microsoft Visual Studio 14.0\Common7\IDE\PrivateAssemblies**, then on the toolbar click **Paste**.
15. In the **Destination Folder Access Denied** dialog box, click **Continue**.
16. In Visual Studio 2015, click **Preview**. The report should render successfully, showing a single value: **96.8**. Click **Design**.
17. In the design pane, right-click anywhere in the central pane that is not within the report body, then click **Report Properties**.
18. In the **Report Properties** dialog box, on the **References** page, under **Add or remove classes** click **Add**.

19. In the **Class Name** box, type **ExampleAssembly.CurrencyConverters**, then in the **Instance Name** box, type **myCC** then click **OK**.
20. Right-click the second text box in the design pane, then click **Expression**.
21. In the **Expression** dialog box, in the **Set expression for: Value** box, type the following, and then click **OK**:

```
=Code.myCC.USDToBitcoin(100,101)
```

22. In Visual Studio 2015, click **Preview**. The report should render successfully, showing two values: **96.8** and **10100**. Click **Design**.
23. In Solution Explorer, right-click **Assemblies.rdl** then click **Deploy**. Observe that deployment fails because the assembly cannot be found on the report server.
24. In File Explorer, navigate to **D:\Demofiles\Mod09\Demo\ExampleAssembly\bin\Debug**, right-click **ExampleAssembly.dll**, and then click **Copy**.
25. Navigate to **C:\Program Files\Microsoft SQL Server Reporting Services\SSRS\ReportServer\bin**, then on the toolbar click **Paste**.
26. In the **Destination Folder Access Denied** dialog box, click **Continue**.
27. Right-click the **Start** button, then click **Windows PowerShell (Admin)**.
28. In the **User Account Control** dialog box, click **Yes**.
29. At the PowerShell prompt, type the following and then press Enter:

```
Restart-Service -name SQLServerReportingServices
```

30. Close Windows PowerShell.
31. In Visual Studio 2015, in Solution Explorer, right-click **Assemblies.rdl** then click **Deploy**.
32. When deployment succeeds, in Internet Explorer, navigate to **http://mia-sql/reports_sql2**, click **Module09** then click **Assemblies**. Observe that the report renders as expected.

Custom assemblies in Report Builder

1. In Report Builder, on the **File** menu, click **Open**.
2. If the **Microsoft SQL Server Report Builder** dialog box appears, click **No**.
3. In the **Open Report** dialog box, in the **Name** box, type **D:\Demofiles\Mod09\Builder\Assemblies_builder.rdl**, then click **Open**.
4. Right-click anywhere in the dark grey background surrounding the report body, then click **Report Properties**.
5. In the **Report Properties** window, on the **References** page, under **Add or remove assemblies** click **Add**.
6. In the **Add or remove assemblies** box, click the ellipsis button next to the new row.
7. In the **Open** dialog box, in the **File name** box, type **D:\Demofiles\Mod09\Demo\ExampleAssembly\bin\Debug\ExampleAssembly.dll** then click **Open**.
8. Under **Add or remove classes** click **Add**.

9. In the **Class Name** box, type **ExampleAssembly.CurrencyConverters**, then in the **Instance Name** box, type **myCC** then click **OK**.
10. Click **Run**, and observe that an error is raised because the assembly cannot be located.
11. In the **Microsoft SQL Server Report Builder** box, click **OK**.
12. Click **Design**.
13. Right-click the first text box in the design pane, then click **Expression**.
14. In the **Expression** dialog box, in the **Set expression for: Value** box, type the following, and then click **OK**:

```
=ExampleAssembly.TemperatureConverters.CelsiusToFahrenheit(36.0)
```

15. Right-click the second text box in the design pane, then click **Expression**.
16. In the **Expression** dialog box, in the **Set expression for: Value** box, type the following, and then click **OK**:

```
=Code.myCC.USDTtoBitcoin(100,101)
```

You do not need to perform steps 17 to 23 if you completed the Report Designer section of this demo. If you completed the Report Designer demo, continue at step 24.

17. In File Explorer, navigate to **D:\Demofiles\Mod09\Demo\ExampleAssembly\bin\Debug**, right-click **ExampleAssembly.dll**, and then click **Copy**.
18. Navigate to **C:\Program Files\Microsoft SQL Server Reporting Services\SSRS\ReportServer\bin**, then on the toolbar click **Paste**.
19. In the **Destination Folder Access Denied** dialog box, click **Continue**.
20. Right-click the **Start** button, then click **Windows PowerShell (Admin)**.
21. In the **User Account Control** dialog box, click **Yes**.
22. At the PowerShell prompt, type the following, and then press Enter:

```
Restart-Service -name SQLServerReportingServices
```

23. Close Windows PowerShell.
24. In Report Builder, in the bottom left of the screen click **Connect**.
25. In the **Connect to Report Server** box, type **http://mia-sql/reportserver_sql2** then click **Connect**.
26. Click **Run**. The report renders successfully because the preview is rendered on the Reporting Server, where the custom assembly has been deployed.

Check Your Knowledge

Question	
Which of the following statements about custom assemblies is not true?	
Select the correct answer.	
<input type="checkbox"/>	You can add references to custom assemblies to a report definition using Report Builder.
<input type="checkbox"/>	You can deploy a report definition that references a custom assembly to a report server before you deploy the custom assembly to the report server.
<input type="checkbox"/>	You refer to static members in custom assemblies with a fully referenced name.
<input type="checkbox"/>	You create class instances to work with classes that are not static.
<input type="checkbox"/>	You must copy a private assembly to the PrivateAssemblies folder of your Visual Studio installation to be able to preview reports in Report Designer.

Lesson 3

Integrating Reporting Services

When you have designed and deployed a Reporting Services report, you are not restricted to viewing the report through the Reporting Services web portal. This lesson introduces different methods that you use to integrate Reporting Services reports with other applications.

Lesson Objectives

At the end of this lesson, you should be able to:

- Describe the ReportViewer controls for Visual Studio.
- Explain how to use URL access to interact with Reporting Services.
- Describe the Report Server Web Service APIs.

ReportViewer controls for Visual Studio

You use the ReportViewer control to embed Reporting Services reports in .NET web applications and desktop applications. It comes in two versions:

- **ReportViewer web control:** suitable for embedding in webpages. The ReportViewer web control is used by the Reporting Services web portal to render reports when you view them from a web browser.
- **ReportViewer window control:** suitable for embedding in desktop applications.

- Embed Reporting Services reports in .NET webpages and applications:
 - ReportViewer web control for webpages
 - ReportViewer window control for desktop applications
- Can run in two modes:
 - Remote processing mode
 - Local processing mode

Typically, instances of the ReportViewer control connect to a Reporting Services instance to display reports—this is called remote processing mode. In remote processing mode, the Report Server generates and renders the report, returning the result to the application—where the rendered report is displayed in the ReportViewer control. In remote processing mode, the full power of Reporting Services is available to not only render the report, but also to control access through security settings, and optimize performance with caching and snapshots.

You also use ReportViewer controls to generate reports locally—local processing mode—without a connection to a Report Server; this mode has reduced functionality when compared to remote processing mode. In local processing mode, data processing must be handled by the application—which is responsible for creating the dataset(s) that the report is based on. The ReportViewer control handles report processing; you only render reports to PDF, Word, Excel, or Image when in local processing mode.

To use the ReportViewer control, you add the appropriate version of the control (the web control for web applications, the window control for desktop applications) to your .NET application from the Visual Studio toolbox, then configure the control properties.

To learn more about the ReportViewer controls, see the topic *Integrating Reporting Services Using ReportViewer Controls* in Microsoft Docs:



Integrating Reporting Services Using ReportViewer Controls

<https://aka.ms/U3ap9j>

For an example of how to add the ReportViewer control to a web application, see the topic *Using the WebForms ReportViewer Control* in Microsoft Docs:

 **Using the WebForms ReportViewer Control**

<https://aka.ms/A8u0j1>

For an example of how to add the ReportViewer control to a desktop application, see the topic *Using the WinForms ReportViewer Control* in Microsoft Docs:

 **Using the WinForms ReportViewer Control**

<https://aka.ms/A0fex6>

URL access

You use URL access to control a Reporting Services report server instance by composing a command in a URL request. By using URL access, you control the values passed to report parameters, in addition to configuring other features, such as the rendered format of the report. You choose whether the report is displayed in the Reporting Services web portal, or whether the rendered version of the report is returned as part of the response to the URL request.

- Render reports and view metadata with URL requests
- URLs are constructed from:
 - Report server URL
 - Delimiter (?)
 - Report path
 - Parameters:
 - Name/value pairs
 - Delimited by ampersand (&)
 - Prefix for HTML viewer (rc:) or report server (rs:) parameters

You also use URL requests to read report metadata—for example, to return a listing of the contents of a folder on the report server, or to view the definition of an item on the report server, in addition to controlling features of the Report Viewer control in the web portal.

URL access offers a straightforward way to integrate Reporting Services into your applications and scripts, because most programming languages support URL requests.

Building URL requests

You start a URL for Reporting Services URL access with the URL of the report server—note that this URL is different to the Web Portal URL. You view the report server URL in Reporting Services Configuration Manager, on the **Web Server URL** page.

The report server URL is followed by a question mark (?) that indicates the start of the query string.

The query string is made up of:

- The resource path: this is a path to the report (or other Reporting Services resource) that you want to access with the URL. The path is relative to the root of the report folder tree. The path must begin with a slash (/):

The following example shows the URL that's required to access the **Product_List** report in the **Products** folder on the **MySSRS** Reporting Services instance on the **RPT-SRV-01** server:

Simple URL access

```
http://RPT-SRV-01/MySSRS?/Products/Product_List
```

- Zero or more name and value pairs for URL parameters: name and value pairs take the form **<name>=<value>**. Name and value pairs are delimited by the ampersand symbol (&). When a name and value pair is used to configure the HTML viewer or the report server, it has a prefix to indicate the target of the parameter:
 - HTML viewer parameters have the prefix **rc:**.
 - Report server parameters have the prefix **rs:**.

A name and value pair without a prefix is treated as a report parameter.

The following example shows the URL that's required to access the **Sales Detail** report in the root folder on the **MySSRS** Reporting Services instance on the **RPT-SRV-01** server. The report takes a single parameter **OrderId**; in this example, the parameter has the value 54867:

URL access report parameter

```
http://RPT-SRV-01/MySSRS?/Sales_Detail&OrderId=54867
```

The following example shows the same report and parameters as the previous example. In this case, the URL includes an HTML viewer parameter **Toolbar** that's set to **false** to hide the HTML viewer toolbar:

URL access HTML viewer

```
http://RPT-SRV-01/MySSRS?/Sales_Detail&OrderId=54867&rc:Toolbar=false
```

The following example shows the same report and parameters as the penultimate example. In this case, the URL includes a report server parameter **Format** that's set to **PDF** to return the report as a PDF:

URL access report server

```
http://RPT-SRV-01/MySSRS?/Sales_Detail&OrderId=54867&rs:Format=PDF
```

For more information about working with URL access, see the topic *URL Access (SSRS)* in Microsoft Docs:



URL Access (SSRS)

<https://aka.ms/Q47nyv>

For a complete list of HTML viewer and report server parameters for URL access, see the topic *URL Access Parameter Reference* in Microsoft Docs:



URL Access Parameter Reference

<https://aka.ms/R2lnmp>

Report Server Web Service

Reporting Services provides programmatic access to the report server with the Report Server Web Service. Depending on the edition of Reporting Services that you're using, you might have more than one protocol to choose from to interact with the web service:

- Simple Object Access Protocol (SOAP).
- Representational State Transfer (REST).


SOAP access

The Reporting Services Web Service XML service is accessible using SOAP over HTTP. SOAP access is available in all currently supported versions of Reporting Services, including SQL Server 2008, 2008R2, 2012, 2014, 2016, and 2017.

Two endpoints are available for SOAP access:

- **ReportingService2010**: used for managing the Reporting Services instance, including such operations as creating and deleting folders, data sources, datasets, and paginated reports.
- **ReportExecutionService**: used for rendering reports.

You can develop applications that interact with the SOAP API using the .NET framework, or any programming language that includes a SOAP client. You can also develop Visual Basic scripts that interact with the SOAP API that you execute with the **rs.exe** application that's distributed with Reporting Services. You use the **rs** application to manage Reporting Services from the command line without having to implement your own client for the SOAP service.

 **Note:** When you interact with the SOAP API, you should use the Web Service URL for the Reporting Services instance. You view the report server URL in Reporting Services Configuration Manager, on the **Web Server URL** page.

For more information about the Reporting Services SOAP service, see the topic *Report Server Web Service* in Microsoft Docs:

 **Report Server Web Service**

<https://aka.ms/Lqbkiv>

For information about the methods supported by the SOAP service endpoints, see the topic *Report Server Web Service Methods* in Microsoft Docs:

 **Report Server Web Service Methods**

<https://aka.ms/Qzcbia>

REST API

In SQL Server 2017, the SOAP API is deprecated and replaced by a REST API. The Reporting Services REST API is not available in versions of Reporting Services before SQL Server 2017.

- SOAP API
 - XML service presented using SOAP over HTTP
 - In all currently supported versions of Reporting Services
 - Interact using .NET SDK, rs.exe, or any SOAP client
 - Use the Web Service URL
- REST API
 - SQL Server 2017 only—supersedes SOAP API
 - Includes support for mobile reports, KPIs, and other new features
 - Interact using any REST client
 - Use the Web Portal URL

REST is a more modern, lighter-weight protocol than SOAP. REST clients pass requests to server endpoints using HTTP methods—sometimes also referred to as HTTP verbs—GET, POST, PUT, DELETE, and PATCH.

The Reporting Services REST API gives you the same capabilities as the SOAP API—including managing Reporting Services objects and rendering reports—and adds capabilities to manage features added in recent versions of Reporting Services, including mobile reports and KPIs.

You can develop applications that interact with the REST API by using any programming language that includes a REST client.



Note: When you interact with the REST service, you should use the Web Portal URL for the Reporting Services instance. You view the report portal URL in Reporting Services Configuration Manager, on the **Web Portal URL** page.

The REST methods are in the /api/v2.0/ subfolder of the Web Portal URL.

For more information about the REST API, see the topic *Develop with the REST APIs for Reporting Services* in Microsoft Docs:



Develop with the REST APIs for Reporting Services

<https://aka.ms/Qqfwf2>

Complete documentation for the REST API is not held in Microsoft Docs; instead, it's hosted on the OpenAPI (Swagger hub) site:



Reference Links: <https://app.swaggerhub.com/apis/microsoft-rs/SSRS/2.0>

Demonstration: Using the Report Server Web Service

In this demonstration, you will see how to interact with the Report Server Web Service using:

- The SOAP API and rs.exe.
- The REST API.

You will see how to:

- List the contents of a folder.
- Create a new folder.

Run a report.

Demonstration Steps

Identify the Web Service URL

1. Click **Start**, then type **Reporting Services Configuration Manager** then press Enter.
2. In the **User Account Control** dialog box, click **Yes**.
3. In the **Reporting Services Configuration Connection** dialog box, confirm that the value in the **Server name** box is **MIA-SQL**, then click **Connect**.
4. Click **Web Service URL**

5. On the **Web Service URL** page, note the value in the **Report Server Web Service URLs** box.
6. Close Reporting Services Configuration Manager.

SOAP API and rs.exe

1. In File Explorer, navigate to **D:\Demofiles\Mod09\API**, then double-click **SOAP_listing.vb**.
2. In the **Open File - Security Warning** dialog box, click **Run**.
3. When the file opens in Visual Studio 2015, review the contents of the file.
4. Right-click the **Start** button then click **Windows PowerShell**.
5. At the PowerShell prompt, type the following, and then press Enter:

```
cd D:\Demofiles\Mod09\API
```

6. At the PowerShell prompt, type the following, and then press Enter:

```
rs -i SOAP_listing.vb -s http://mia-sql/ReportServer_SQL2 -e Mgmt2010
```

Note that the command uses **rs.exe** to execute **SOAP_listing.vb** against the **Mgmt2010** endpoint of the MIA-SQL reporting services instance, then press Enter. A list of items in the **Adventureworks Sample Reports** folder is returned.

7. In File Explorer, navigate to **D:\Demofiles\Mod09\API**, then double-click **SOAP_create_folder.vb**.
8. In the **Open File - Security Warning** dialog box, click **Run**.
9. When the file opens in Visual Studio 2015, review the contents of the file.
10. In Windows PowerShell, at the PowerShell prompt, type the following, and then press Enter:

```
rs -i SOAP_create_folder.vb -s http://mia-sql/ReportServer_SQL2 -e Mgmt2010
```

11. In Internet Explorer, navigate to **http://mia-sql/reports_sql2** and observe that the **NewFolderSOAP** folder has been created.
12. In File Explorer, navigate to **D:\Demofiles\Mod09\API**, then double-click **SOAP_run_report.vb**.
13. In the **Open File - Security Warning** dialog box, click **Run**.
14. When the file opens in Visual Studio 2015, review the contents of the file.
15. In Windows PowerShell, at the PowerShell prompt, type the following, and then press Enter:

```
rs -i .\SOAP_run_report.vb -s http://mia-sql/ReportServer_SQL2 -e Exec2005
```

Note that the command targets the **Exec2005** endpoint.

16. In File Explorer, navigate to **D:\Demofiles\Mod09\API**, then double-click **report.mht** to review the content of the report.
17. Close the Internet Explorer tab.

REST API

The scripts for this demonstration can be found in **D:\Demofiles\Mod09\API\rest.txt** on 10990C-MIA-SQL.

1. In Windows PowerShell, at the PowerShell prompt, type the following, and then press Enter:

```
Invoke-RestMethod -Uri "http://mia-sql/reports_sql2/api/v2.0/Folders(Path='/AdventureWorks Sample Reports')/CatalogItems" -Method Get -UseDefaultCredentials | % {$_.value | Format-Table Type,Name}
```

Observe that the URI is a subfolder of the Web Portal URL.

2. At the PowerShell prompt, to create a folder, type the following, and then press Enter:

```
$payload = ConvertTo-Json(@{"Path" = "/" ; "Name" = "NewFolderREST";})
```

3. At the PowerShell prompt, type the following, and then press Enter:

```
Invoke-RestMethod -Uri "http://mia-sql/reports_sql2/api/v2.0/Folders" -Method Post -UseDefaultCredentials -Body $payload -ContentType "application/json"
```

4. In Internet Explorer, navigate to **http://mia-sql/reports_sql2** and observe that the **NewFolderREST** folder has been created.
5. In Windows PowerShell, at the PowerShell prompt, to run a report, type the following, and then press Enter:

```
$response = Invoke-WebRequest -Uri "http://mia-sql/reportserver_sql2?/AdventureWorks+Sample+Reports/Employee_Sales_Summary&EmployeeID=283&ReportMonth=12&ReportYear=2013&rs:Format=MHTML" -Method Get -UseDefaultCredentials
```

Observe that this command uses URL access—the REST API has no method for executing reports

6. At the PowerShell prompt, type the following, and then press Enter:

```
[System.IO.File]::WriteAllBytes("D:\demofiles\mod09\api\rest_report.mht", $response.Content)
```

7. In File Explorer, navigate to **D:\Demofiles\Mod09\API**, then double-click **rest_report.mht** to review the content of the report.
8. Close all open applications without saving any changes.

Check Your Knowledge

Question	
You want to use the REST API to access a Reporting Services instance called reports on the SSRS01 server. Which of the following is the correct service URL?	
Select the correct answer.	
<input type="checkbox"/>	http://SSRS01/reports/api/v2.0
<input type="checkbox"/>	http://SSRS01/reportserver/api/v2.0
<input type="checkbox"/>	http://reports/SSRS01/api/v2.0
<input type="checkbox"/>	http://SSRS01/reports/
<input type="checkbox"/>	http://SSRS01/reportserver

Lab: Extending and integrating Reporting Services

Scenario

You need to customize a report design with features that you will add through embedded code and expressions. You will also prepare and test URLs for URL access to reports from an application.

Objectives

At the end of this lab, you should be able to:

- Customize Reporting Services reports with embedded code.
- Compose URLs for URL access to reports.

Estimated Time: 90 minutes

Virtual machine: **10990C-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa55w.rd**

Exercise 1: Custom code—Report Designer

Scenario

You have been asked to create a report based on a dataset provided by an external source. The dataset contains three columns:

- Customer name, in the format <last name>, <first name>.
- Last order date, in the format YYYYMMDD.
- Last order amount.

The data type of all three columns is string (varchar/nvarchar).

Your colleague started to create the report, but did not know how to convert the source data into the required format:

- The report should contain four columns:
 - First name.
 - Last name.
 - Last order date, formatted as <day> <short month name> <year>.
 - Last order amount, formatted as an integer (discarding the decimal portion).

Alternate columns should be formatted with a light gray background (this is a “nice-to-have” feature but is not essential).

Using the report provided as a starting point by your colleague, you should create a report to satisfy these requirements.

The main tasks for this exercise are as follows:

1. Prepare the lab environment
2. Review the requirements
3. Format last order amount
4. Format last order date

5. Format customer name

6. Alternate row colors

► **Task 1: Prepare the lab environment**

1. Ensure the **10990C-MIA-CLI**, **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. Run **Setup.cmd** in the **D:\Labfiles\Lab09\Starter** folder as Administrator.

► **Task 2: Review the requirements**

- Review the requirements in the exercise scenario.

► **Task 3: Format last order amount**

1. Using Visual Studio 2015, open the **D:\Labfiles\Lab09\Starter\Lab\Lab.sln** solution, then open **OrdersReport.rdl**.
2. Use an expression to format **Last Order Amount** as an integer.

► **Task 4: Format last order date**

1. Use embedded code to create a function **yyymmdd_to_short_date** that converts a string in the format **YYYYMMDD** to a date with the format **<day> <month short name> <year>** (format string "dd MMM yyyy").
2. Update the definition of the **Last Order Date** column to apply this function to the **LastOrderDate** column in the report dataset.

You can find a sample implementation of the function in **D:\Labfiles\Lab09\Starter\Code\date_function.txt**, or you can write your own implementation.

► **Task 5: Format customer name**

1. Use embedded code to write a function called **SplitName** that splits a string in the format **<last name>, <first name>** into two strings. Return these strings as an array.
2. Update the definitions of the **First Name** and **Last Name** column to apply this function to the **CustomerName** column in the report dataset so that the **First Name** column contains the customer's first name, and the **Last Name** column contains the customer's last name.

You can find a sample implementation of the function in **D:\Labfiles\Lab09\Starter\Code\name_function.txt**, or you can write your own implementation.

► **Task 6: Alternate row colors**

- Use an expression to set the background color of alternate rows in the report to light gray. When you have finished, close Visual Studio 2015 without saving any changes.

*Hint: You can use the Visual Basic **RowNumber()** function to track the current row number.*

Results: At the end of this lab, you should be able to use expressions and embedded code to control the appearance of a Reporting Services report.

Exercise 2: Custom code—Report Builder

Scenario

You have been asked to create a report based on a dataset provided by an external source. The dataset contains the following three columns:

- Customer name, in the format <last name>, <first name>.
- Last order date, in the format YYYYMMDD.
- Last order amount.

The data type of all three columns is string (varchar/nvarchar).

Your colleague started to create the report, but did not know how to convert the source data into the required format:

- The report should contain four columns:
 - First name.
 - Last name.
 - Last order date, formatted as <day> <short month name> <year>.
 - Last order amount, formatted as an integer (discarding the decimal portion).

Alternate columns should be formatted with a light gray background (this is a “nice-to-have” feature but is not essential).

Using the report provided as a starting point by your colleague, you should create a report to satisfy these requirements.

The main tasks for this exercise are as follows:

1. Prepare the lab environment
2. Review the requirements
3. Format last order amount
4. Format last order date
5. Format customer name
6. Alternate row colors

► Task 1: Prepare the lab environment

- Run **Setup.cmd** in the **D:\Labfiles\Lab09\Starter** folder as Administrator.

► Task 2: Review the requirements

- Review the requirements in the exercise scenario.

► Task 3: Format last order amount

1. Using Report Builder, open **D:\Labfiles\Lab09\Starter\Builder\OrdersReport.rdl**.
2. Use an expression to format **Last Order Amount** as an integer.

► Task 4: Format last order date

1. Use embedded code to create a function **yyymmdd_to_short_date** that converts a string in the format **YYYYMMDD** to a date with the format **<day> <month short name> <year>** (format string “dd MMM yyyy”).

2. Update the definition of the **Last Order Date** column to apply this function to the **LastOrderDate** column in the report dataset.

You can find a sample implementation of the function in `D:\Labfiles\Lab09\Starter\Code\date_function.txt`, or you can write your own implementation.

► Task 5: Format customer name

1. Use embedded code to write a function called **SplitName** that splits a string in the format **<last name>,<first name>** into two strings. Return these strings as an array.
2. Update the definitions of the **First Name** and **Last Name** column to apply this function to the **CustomerName** column in the report dataset so that the **First Name** column contains the customer's first name, and the **Last Name** column contains the customer's last name.

You can find a sample implementation of the function in `D:\Labfiles\Lab09\Starter\Code\name_function.txt`, or you can write your own implementation.

► Task 6: Alternate row colors

- Use an expression to set the background color of alternate rows in the report to light gray. When you have finished, close Report Builder without saving any changes.

*Hint: You can use the Visual Basic **RowNumber()** function to track the current row number.*

Exercise 3: URL access

Scenario

An application developer in the AdventureWorks IT team wants to integrate some Reporting Services reports into an application for use by members of the Sales department. You have agreed with the developer that URL access is a suitable solution. You need to provide example URLs to allow the application developer to include the following actions in the application (the reports are all found in the **Adventureworks Sample Reports** folder on the reporting server).

- Provide a link to the **Customers_Near_Stores** report for viewing on the web portal. The report toolbar should be hidden.
- Generate the **Employee_Sales_Summary** report as an image for display in the application.

The main tasks for this exercise are as follows:

1. Identify the Web Service URL
2. Link for Customers_Near_Stores
3. Employee_Sales_Summary as an image

► Task 1: Identify the Web Service URL

- Use Reporting Services Configuration Manager to identify the Web Service URL for the **SSRS** instance on **MIA-SQL**. Note this value in `D:\Labfiles\Lab09\Starter\URLAccess.rtf`.

► Task 2: Link for Customers_Near_Stores

1. In the Reporting Services web portal, review the **Customers_Near_Stores** report in the **Adventureworks Sample Reports** folder. Note the names of the report parameters.
2. Use the information you have gathered so far to build a URL to access the **Customers_Near_Stores** report in the portal, for customers within 50 miles of the Area Bike Accessories store. Test the URL in Internet Explorer to confirm that it works. Note that you must supply the store location as a spatial

POINT data type. The location of Area Bike Accessories is **POINT (-120.928658084295 37.6746342922879)**.

3. Add a parameter to the URL to hide the report viewer toolbar. Test the URL in Internet Explorer to confirm that it works, then note the URL in **URLAccess.rtf**.

► **Task 3: Employee_Sales_Summary as an image**

1. In the Reporting Services web portal, review the **Employee_Sales_Summary** report in the **Adventureworks Sample Reports** folder. Note the names of the report parameters.
2. Use the information you have gathered so far to build a URL to access the **Employee_Sales_Summary** report as an image. Use employee id 283 and November of 2013 as the sample parameter values. Test the URL in Internet Explorer to confirm that it works, then note the URL in **URLAccess.rtf**. Close all open applications when you have finished.

Results: At the end of this lab, you should be able to build URLs for URL access to Reporting Services reports.

Question: Did the functions you wrote in exercise 1 or 2 match the examples? If not, how were they different? If they did match, how else could you achieve the same ends?

Module Review and Takeaways

In this module, you have learned different methods for extending the functionality of Reporting Services using functions, embedded code, and external assemblies. You have also learned about methods for integrating Reporting Services into other applications and programming languages, including .NET controls, URL access, and the REST and SOAP APIs for the Report Server Web Service.

Review Question(s)

Question: Are your organization's reporting needs met by the Reporting Services standard functionality, or might you use the techniques that you have learned in this module to extend its functionality?

MCT USE ONLY. STUDENT USE PROHIBITED

Module 10

Introduction to Mobile Reports

Contents:

Module Overview	10-1
Lesson 1: Overview of mobile reports	10-2
Lesson 2: Preparing data for mobile reports	10-7
Lesson 3: Mobile Report Publisher	10-14
Lab: Introduction to mobile reports	10-18
Module Review and Takeaways	10-21

Module Overview

This module introduces the design and publication of reports that are intended for consumption on mobile devices, such as smartphones and tablets. Microsoft® SQL Server® Reporting Services (SSRS) includes support for mobile reports, although the tools that are used to design and publish mobile reports are different to the tools used for the paginated reports discussed in the earlier modules of this course.



Note: Mobile Reports are supported in SQL Server 2016 and SQL Server 2017 on Windows. Earlier versions of SQL Server do not support mobile reports. Mobile Reports are supported in Enterprise edition only (and in Developer edition for nonproduction use).

Objectives

After completing this module, you should be able to:

- Describe considerations for mobile reports.
- Prepare data for publication in mobile reports.
- Use SQL Server Mobile Report Publisher to create mobile reports.

Lesson 1

Overview of mobile reports

Designing mobile reports requires careful planning, due to the limitations of the mobile devices that they will be viewed on. Traditional SSRS reports are likely to be difficult to interpret on a mobile device. You must also consider whether to use a data-first or design-first approach for your reports.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe the motivation for mobile reporting solutions.
- Explain the design-first approach to mobile reporting.
- Explain the data-first approach to mobile reporting.
- Use mobile reporting key performance indicators (KPIs).

Why do we need mobile reports?

There are many reasons for the introduction of mobile reports, some of which are covered in this lesson.

Better mobile experience

High-powered mobile computing devices—such as smartphones and tablets—are ubiquitous in many modern businesses. However, the screen size and touch interface of these devices might make it awkward to consume the paginated reports generated by traditional reporting tools such as SSRS. SQL Server mobile reports use an intuitive grid-based layout with which you can

quickly design reports for different mobile device form factors. Mobile reports are viewed in a browser on any device, or accessed through the Power BI app for iOS, Android or Windows Phone.

- Better experience for mobile users
 - Browser or Power BI for iOS, Android or Windows Phone
- KPI and dashboard components
- Draw together data from many sources
- Consistent look and feel across desktop, mobile, and tablet

KPI and dashboard components

Business decision makers typically prefer to quickly view and interpret summarized information in the form of KPIs and dashboards. SQL Server mobile reports might include many visual components—such as gauges, graphs, and maps—that you easily link to your data and add to your reports.

For more information about the visual components that are supported by mobile reporting, see the topic *Add visualizations to Reporting Services mobile reports* in Microsoft Docs:



Add visualizations to Reporting Services mobile reports

<https://aka.ms/D9hvog>

Data from many sources

Business dashboards and KPI scorecards typically draw information from diverse sources into a single report. SQL Server mobile reports can be based on SSRS data sources or created directly from data in a Microsoft Excel® spreadsheet. You don't need to define formal relationships between source data; SSRS automatically correlates data from different sources based on dates and times or on key values—so that report elements from different sources always display related data.

For more information about the mobile reporting data model, see the topic *Data for Reporting Services mobile reports* in Microsoft Docs:



Data for Reporting Services mobile reports

<https://aka.ms/Kf60qs>

Consistent look and feel

Users expect a consistent look and feel when they view reports on different devices. Although they are called “mobile” reports, the reports generated by the mobile report publishing process can be viewed on any computer. The SSRS portal accommodates and displays mobile reports and KPIs alongside paginated reports.

Data-first approach to mobile reports

There are two approaches you might take when designing mobile reports. Data-first is like the approach you typically use when designing reports using Report Builder or Report Designer, which has the following steps:

1. Identify data sources.
2. Compose data source queries.
3. Link reporting elements to query result sets.

You might use the data-first approach when you are confident that the source data is formatted correctly for use with mobile reports, or when you convert an existing paginated report to a mobile report.

- Start from the source data, then link the source data to reporting elements
- Use when:
 - Your source data is correctly formatted for mobile reports
 - The format of the source data is fixed (for example, when converting an existing report to a mobile report)
- Shared datasets
 - Use SSRS shared datasets as a source for report data

Shared datasets

Mobile reports use data drawn from Excel files, or from SSRS shared datasets. After a shared dataset is created on your Reporting Services server, you use it as the basis for a mobile report. In its current version, the Mobile Report Publisher does not have the capability to create new shared datasets; you must use Report Manager or SQL Server Data Tools (SSDT) to define new shared datasets. After data is imported into the Mobile Report Publisher application, it's treated in the same way, whatever its source.

When a report based on a shared dataset has been published to SSRS, by default the report will query the latest data from the data source each time the report is run. You can optionally configure caching for the shared dataset; this causes SSRS to cache a copy of the query results and use the cached values to serve the report. The report data in the cache is refreshed on a schedule that you define.



Note: Using an Excel file as a data source for a mobile report is covered in the next lesson of this module.

For more information about working with shared data sources in mobile reports, see the topic *Get data from shared datasets in Reporting Services mobile reports* in Microsoft Docs:



Get data from shared datasets in Reporting Services mobile reports

<https://aka.ms/Hvymqh>

Design-first approach to mobile reports

The second approach to designing mobile reports is design-first. With this approach, you define the visual composition of the report before you link it to one or more sources of data. When you use the design-first approach, the Mobile Report Publisher generates a small set of random simulated data on which your report controls might be based while you are configuring them.

The design-first approach is most useful when you are exploring the best way to present data, or the data sources you plan to use are not available in a format suitable for mobile reports. Because the simulated dataset is part of the report definition, you can publish a mobile report based on simulated data alone; this is useful when you want to get early feedback from other users about the report design.

You also use the design-first approach to guide you when creating your data sources. The schema and format of the simulated test data is a useful guide to the format that the final data source for the report element should use. You export the simulated data generated by Mobile Report Publisher to an Excel file, which can be used to guide the creation of actual datasets.

For more information about design-first and data-first approaches to mobile report design, see *Design first or data first when creating in Reporting Services mobile reports* in Microsoft Docs:



Design first or data first when creating in Reporting Services mobile reports

<https://aka.ms/Nhue5b>

For more information about working with simulated data in the design-first approach, see the topic *Work with simulated data in Reporting Services mobile reports* in Microsoft Docs:



Work with simulated data in Reporting Services mobile reports

<https://aka.ms/Bobuvp>

- Report layout is designed before data is available
- Use:
 - When data is not available in suitable format
 - To prototype report design
 - To guide data design

Mobile report KPIs

SQL Server mobile reports support KPIs that you create directly in Reporting Services folders through the new portal. You configure a mobile report KPI entirely by using manual data, using fields from a shared dataset, or from a mixture of manual and shared dataset data.

Each KPI might have the following attributes:

- **KPI name.** The title that appears on the KPI.
- **Description.** A free-text description of the KPI—optional.
- **Value format.** The formatting applied to the KPI value.
- **Value.** The value for the KPI. Can be set manually or be drawn from a dataset field.
- **Goal.** The KPI goal—optional. Can be not set, set manually, or drawn from a dataset field.
- **Status.** The KPI status: Red, Amber, or Green (RAG)—optional. Can be not set, set manually, or drawn from a dataset field.
- **Trend set.** Comma separated values used to generate a trend line—optional. Can be not set, set manually, or drawn from a dataset field. If a dataset is used, it must be a column of numeric values.
- **Visualization.** A trend line visualizing the trend set data. Can be none, a bar graph, a line graph, a stepped line graph, or an area graph.

- Very high-level performance indicators
- Can be manually configured, or configured from data, or from a mixture of manual configuration and data
- Security controlled at object level

In common with other Reporting Services objects, you configure object-level security to control which users have permission to view the KPI.

Demonstration: Mobile reports in action

In this demonstration, you will see several examples of mobile reports.

Demonstration Steps

1. Ensure that the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Demofiles\Mod10** folder, run **Setup.cmd** as Administrator.
3. In the **User Account Control** dialog box, click **Yes**.
4. When setup steps are complete, open Internet Explorer, navigate to **mia-sql/Reports_SQL2**, click **MobileReports**.
5. Click **Gauge Example**.

The report shows an example of the same data presented on various gauge styles supported by mobile reporting. The subtitle of each gauge shows the gauge type.

6. Click the title bar of the **Total Sales KPI Linear gauge** report element. Notice that the element expands to fill the whole view. Click the title bar of the **Total Sales KPI Linear gauge** report element again to return to the default view. This facility makes it easier to view detailed charts on small screens.

7. Click the browser **Back** button.
8. Click **Simulated Example**.

The report shows various report elements linked by a scorecard grid. The report uses simulated data.

9. In the **Scorecard grid 1** region of the report, click **Simulated Item A2**. Notice that the values of all the other reporting elements on the page change to reflect the subtotals for the simulated data item.
10. In the **Scorecard grid 1** region of the report, click **Simulated Item B1**. Notice that the other reporting elements again change to reflect your selection.
11. In the **Scorecard grid 1** region of the report, click **All** to return to the overall summary view.
12. Click the browser **Back** button.
13. Click **Time Navigator Example**.

The report shows a bar graph and a map linked to a time navigator element. The report will take a few seconds to open because it is based on data retrieved from the **AdventureWorks** and **AdventureWorksDW** databases.

14. Observe that the **All** section of the time navigator (the unlabeled element in the top left of the report) is selected. Observe that the other bar graph shows data by year.
15. In the time navigator click **2013**. Observe that the bar graph changes to show values by month. Also notice that the values on the **Sales by State** map update; the map now shows aggregated data for the previous year.
16. Click the browser **Back** button.
17. Click **Tree Map Example**.

The report shows a tree map filtered by a selection list. The report is based on a dataset embedded in the report. The size of each box in the tree map is determined by the value of the **Amount** field, relative to the other **Amount** values. The color of each box is determined by a comparison of **Amount** to **Delta** values.

18. On the left, click **Media** and demonstrate that the tree map adjusts to show data related to **Media**.
19. Close Internet Explorer.

Verify the correctness of the statement by placing a mark in the column to the right.

Statement	Answer
True or false? When you follow the design-first approach to mobile report development, you configure your data sources before designing your report.	

Lesson 2

Preparing data for mobile reports

SQL Server mobile reports determine links between datasets and filter data accordingly, but this ability is reliant on report data being properly formatted. Report performance might be significantly affected by the amount of data aggregation that must be carried out when the report is generated. This lesson introduces some considerations for preparing data for use in mobile reports, including formatting and the preaggregation of data.

You will also learn about the steps needed to prepare an SSRS instance for mobile reporting.

Lesson Objectives

After completing this lesson, you'll be able to:

- Optimize reports by preaggregating data.
- Properly format dates and times.
- Prepare data for use in filters.
- Use Excel files in mobile reports.
- Configure an SSRS instance for mobile access.

Optimizing reports with preaggregation

In general, the performance of mobile reports will improve when the amount of aggregation that must be carried out by Reporting Services is kept to a minimum. For example, for a report that shows order values by month, Reporting Services does not have to aggregate data if the source dataset for the report already contains values aggregated by month. However, if the source dataset is composed of many individual sales order detail rows, Reporting Services must aggregate the data by month before the report can be rendered.

- Mobile reports perform fastest when they do minimal processing on small datasets
 - Preaggregate data for best performance
- Preaggregation when using shared data sources allows performance tuning of source queries
- Shared dataset caching is most effective when minimal aggregation is done on cached result sets
- Trade-off between performance and reuse of datasets between reports



Note: At the time of writing, the unofficial guidance from the mobile report development team is that mobile report datasets should ideally have no more than 100,000 rows.

Aggregated datasets might be substantially smaller in size than the source data from which they are drawn. The smaller the dataset, the faster the report is generated.

When you use SSRS datasets as sources for your mobile reports, it will typically be more efficient to carry out aggregations as part of the source query, rather than relying on Reporting Services to perform the aggregation. Aggregation in the source query gives you the opportunity to address query performance issues by indexing, or other performance tuning techniques. Minimizing the aggregation performed by Reporting Services also helps you to make the most of dataset caching; you configure Reporting Services shared datasets to cache their results for a period before requerying for the original data source. Report performance is at its best if the cached data is used in reports that require no further aggregation.

There is, however, a trade-off between optimizing your reports by aggregating data at source, and creating datasets that are reusable for multiple reports—the more aggregated the data in a dataset, the less likely it is that you can use it for multiple reports. You should be prepared to produce aggregated datasets for your most commonly used queries.

Formatting time data

Date and time data must be presented in a format that Mobile Report Publisher correctly recognizes. Many date and time formats are supported. Examples of supported data and time formats include:

04/02/2016
2016-04-02
04/02/2009 16:57:32.8
2016-04-02 16:57:32.8
2016-04-02T16:57:32.8375298-04:00
4/02/2016 16:57:32.80 -04:00
2 April 2016 4:57:32.8 PM
Sat, 2 April 2016 20:57:32 GMT

- Dates and times must be presented in a suitable format
 - Many formats are supported
- Data sources with date and time types will typically present data in a suitable format
- Be aware of date formatting in untyped data, and data from Excel
- Most important for time navigator reports

When you use a shared dataset from a data source that uses date and time data types—such as a SQL Server database or an Analysis Services cube—dates and times will normally be presented in a suitable format by default, providing that the date and time data types are used in the source data. You will typically have to consider date formatting when working with data that isn't typed, or data from Excel files.

Correct date and time formatting is most important for reports that are filtered by a time navigator.

Prepare filter data

You filter data in mobile reports by date and time, or on a key field. You designate a key field for each report element when you link it to a source dataset. Compound keys—made up of more than one column in the dataset—are not supported; to be an effective filter key, a key field should be a single column. However, you can define more than one filter key for use with different filter navigators on the same report.

For example, the following data could use country or region as a filter key, but not a combination of country and region:

- Each filter can only have one filter key column
- Different filters on the same report can use different filter key columns
- Hierarchical filter data for use with a **Selection List** navigator might be represented as an adjacency list

Country	Region
Benin	Africa
Botswana	Africa
United Arab Emirates	Arab States
Bahrain	Arab States
Australia	Asia and Pacific
Bangladesh	Asia and Pacific
Armenia	CIS
Belarus	CIS
Andorra	Europe
Bosnia and Herzegovina	Europe
Bermuda	North America
Antigua and Barbuda	South/Latin America

You also define hierarchical key filters for use with a **Selection List** navigator. To use this facility, prepare a lookup table that describes the hierarchy as an adjacency list, with a column for the filter key and a second column for the parent filter key.

For example, the following table shows the country and region data from the previous example formatted as a lookup table that describes a hierarchy:

Key	ParentKey
Africa	
Arab States	
Asia and Pacific	
CIS	
Europe	
North America	
South/Latin America	
Benin	Africa
Botswana	Africa
United Arab Emirates	Arab States
Bahrain	Arab States
Australia	Asia and Pacific
Bangladesh	Asia and Pacific
Armenia	CIS
Belarus	CIS
Andorra	Europe
Bosnia and Herzegovina	Europe
Bermuda	North America
Antigua and Barbuda	South/Latin America

For more information about formatting data, see the topic *Prepare data for Reporting Services mobile reports* in Microsoft Docs:



Prepare data for Reporting Services mobile reports

<https://aka.ms/Skr1j7>

Using Excel files in mobile reports

When you base a mobile report on an Excel workbook, the data from the workbook becomes part of the mobile report definition. At that point, the link between the report data and the workbook data is broken. If you update the workbook, you must reimport the data to the mobile report definition.

Rules for using Excel data

There are several rules to follow when you prepare data for import from an Excel data source:

- After Excel data is imported to a report definition, it must be reimported to pick up changes
- Rules for using Excel data include:
 - One dataset per worksheet
 - Put column names in the first row
 - Use consistent data types
 - Format dates as dates
 - Use consistent formulas
 - Images, charts, and pivot tables will be ignored
 - Excel 2007 or later (.xlsx file format)

- Mobile Report Publisher might create a dataset for each worksheet in your workbook, so make sure you separate your data into one dataset per worksheet.
- You can have more than one worksheet with the same name, but Mobile Report Publisher will rename them when they are imported. For example, if you have two worksheets named **SalesData**, the imported datasets will be named **SalesData0**, and **SalesData1**.
- Use the first row of each worksheet for column headers, and name the columns appropriately. Although having a column header is not a necessity, any unnamed columns will be named automatically on import using the Excel naming convention of A, B, C ... AA, BB, and so on.
- When importing data from Excel, Mobile Report Publisher compares the data types of the first two rows in each column to detect the data type. If you have a numeric column name, prefix this with a string value before importing the data, so that Mobile Report Publisher can determine this to be the header.
- Keep the data type the same within each column. Mobile Report Publisher scans each column to establish the data type; this process might fail or cause issues if you have mixed types.
- Consider where you store the Excel files that you use to import data. The location doesn't matter but, if you move or delete a report, you will not be able to refresh the data if it changes.
- Ensure cells that contain date values are formatted as dates.
- If you have formulas in your worksheets, make sure all cells in the column are calculated using the same formula.
- Custom objects such as PivotTables, visualizations, and images, will not be imported into Mobile Report Publisher.
- Always use Microsoft Excel 2007 or later, so that files are saved with the **.xlsx** extension.

For more information about preparing Excel files for use in mobile reports, see the topic *Prepare Excel data for Reporting Services mobile reports* in Microsoft Docs:

 **Prepare Excel data for Reporting Services mobile reports**

<https://aka.ms/K1tfjl>

Enable a report server for mobile access

Some configuration might be required to enable access to mobile reports from mobile devices.

Firewall settings

You should confirm that firewall settings on your report server and on your network infrastructure will accept the connections you wish to support. To provide access to mobile reports from mobile devices outside your network, you should make one or more Reporting Services HTTP endpoints accessible from the public internet. Access to mobile reports requires no further firewall settings in addition to those required to permit connections to Reporting Services.

- Configure firewalls to allow connection to mobile reports
 - No special requirements in addition to the firewall rules required for access to SSRS
- For Power BI app access:
 - SSRS must run in native mode
 - Basic authentication must be enabled
 - Enable SSL to prevent information leakage of user names and passwords when using basic authentication

For more information about configuring firewalls to grant access to Reporting Services, see the topic *Configure a Firewall for Report Server Access* in Microsoft Docs:



Configure a Firewall for Report Server Access

<https://aka.ms/T01b4m>

Using Power BI for iOS

If you plan to provide access to mobile reports through device web browsers, no additional configuration is required. However, if you want to give access to mobile reports through the Power BI for iOS app for iPad and iPhone, additional configuration is required.

Native mode

Access to mobile reports through the Power BI app is only supported by a Reporting Services instance running in native mode. Power BI access is not supported by a Reporting Services instance running in SharePoint® integrated mode.

Basic authentication

For access through the Power BI app, Reporting Services must be configured to accept basic authentication. Basic authentication is disabled by default, and is enabled by adding a section to the <Authentication> section of the RSReportServer.config file.

When you use basic authentication, user names and passwords are sent to the server in plain text; for this reason, you should configure SSL for Reporting Services instances. After SSL is configured, connections to Reporting Services can be made over HTTPS, meaning that all reporting traffic is encrypted.



Note: It's not necessary to register a Reporting Services instance for Power BI to provide access to mobile reports through the Power BI app.

For more information about configuring Reporting Services for access to mobile reports, see the topic *Enable a report server for Power BI Mobile access* in Microsoft Docs:



Enable a report server for Power BI Mobile access

<https://aka.ms/G8mzch>

For more information about enabling basic authentication, see the topic *Configure Basic Authentication on the Report Server* in Microsoft Docs:



Configure Basic Authentication on the Report Server

<https://aka.ms/RI5s05>

For more information about configuring SSL, see the topic *Configure SSL Connections on a Native Mode Report Server* in Microsoft Docs:



Configure SSL Connections on a Native Mode Report Server

<https://aka.ms/Yo7imf>

Check Your Knowledge

Question	
How many datasets should you have per Excel worksheet, for use as a data source for a mobile report?	
Select the correct answer.	
<input type="radio"/>	1
<input type="radio"/>	2
<input type="radio"/>	3
<input type="radio"/>	Any number

Lesson 3

Mobile Report Publisher

This lesson introduces the SQL Server Mobile Report Publisher application, and how you use it to work with mobile reports.

Lesson Objectives

After completing this lesson, you will be able to:

- Describe SQL Server Mobile Report Publisher.
- Add a dataset to a mobile report.
- Create a mobile report.

What is SQL Server Mobile Report Publisher?

SQL Server Mobile Report Publisher (SSMRP) is the tool that you use to design and publish mobile reports; at the time of writing, it's the only tool that's available to carry out these tasks.

Accessing SSMRP

SSMRP is not installed as part of SQL Server setup or Reporting Services setup. You access SSMRP by downloading it from the Microsoft website, either directly, or from the link in the Reporting Services portal.

To download SSMRP, use the following link to the Microsoft website:

 **Download SQL Server Mobile Report Publisher**

<https://aka.ms/G9tzlx>

- Accessing SQL Server Mobile Report Publisher (SSMRP):
 - Direct download
 - Download via SSRS portal
- Working with SSMRP:
 - Layout tab
 - Data tab
 - Settings tab
 - Preview tab

Working with SSMRP

You interact with SSMRP through four tabs in the application:

- **Layout.** Configure the layout of the visual components of a report on the **Layout** tab.
- **Data.** Manage datasets and configure how datasets link to visual components and to each other on the **Data** tab. Unlike other SSRS tools, such as Report Builder and Report Designer, SSMRP does not include tools for writing queries against data sources. You must import source data from Excel files or from Reporting Services shared datasets.
- **Settings.** Configure settings for the report, such as the report name, currency code, and first day of the week on the **Settings** tab.
- **Preview.** Use the **Preview** tab to interact with a preview of the published report.

Adding datasets

SSMRP uses a simple data model whereby data is imported into the report as a collection of data views. You don't need to establish formal relationships between data views and, providing the key values match, lookups between one data view and another will work. The mobile report runtime can match date and time aggregations between different data views, even when the granularity is different between the views.

You import data into Mobile Report Publisher from two sources:

- You build a report from multiple datasets
- Formal relationships don't need to be defined between datasets
- Two sources of datasets:
 - Excel workbook
 - SSRS shared datasets

- **Excel local workbook.** After importing the data from Excel, it's stored in the report. You update the data by using the Refresh All Data button.
- **SSRS dataset.** Create a server connection to one or more instances of Reporting Services. Browse the list of datasets published on the server, and choose one to add—you add further datasets separately. The data from Reporting Services will always be up to date, so you don't need to refresh it. The supported data sources comprise those supported by Report Builder and SQL Server Data Tools including, but not limited to, Microsoft Azure SQL Database, Microsoft SQL Server Analysis Services for MDX, Tabular, Microsoft Power Pivot and DMX models, Oracle, Teradata, Microsoft SharePoint List, and XML.

For a complete list of supported data sources and more detailed information, see the topic *Data Sources Supported by Reporting Services (SSRS)* in Microsoft Docs:

 **Data Sources Supported by Reporting Services (SSRS)**

<https://aka.ms/Cbikf9>

You import datasets by using the **Add Data** button on the **Data** tab of SSMRP.

Adding report elements

You design a mobile report using the **Layout** tab in SSMRP, by dragging report elements from the gallery on the left of the tab to the design grid on the right. A single report might contain many elements.

A report element might occupy one or more cells in the design grid. A report element is associated with the data that drives it in the **Data** tab in SSMRP. The order that you add report elements and data to a report will depend on whether you are following the data-first methodology or the design-first methodology.

- Arrange elements on the **Layout** tab:
 - Placement on the screen (represented as a grid)
 - Visual properties
- Work with report element data on the **Data** tab

You configure the visual properties of a report element in the **Visual properties** section of the **Layout** tab. The visual properties that you configure will vary, depending on the type of the element.



Note: Report elements divide into two categories—navigators and visualizations. Navigators and visualizations are covered in more detail in the next module of this course.

For a worked example of creating a mobile report, see the topic *Create a Reporting Services mobile report* in Microsoft Docs:



Create a Reporting Services mobile report

<https://aka.ms/pucrxr>

Demonstration: Creating a mobile report

In this demonstration, you will see how to:

- Format Excel data for use in a mobile report.
- Add a dataset from an Excel file.
- Update a dataset from an Excel file.
- Add a dataset from an SSRS shared dataset.
- Create a simple mobile report.

Demonstration Steps

Format Excel data for use in a mobile report

1. Make sure the **10990C-MIA-CLI** virtual machine is running, then log on to **10990C-MIA-CLI** as **Student** with the password **Pa55w.rd**.
2. Using File Explorer, navigate to **\\mia-sql\Mod10**. In the **Windows Security** dialog, in the **User name** box type **Adventureworks\student** then in the **Password** box type **Pa55w.rd**, then click **OK**.
3. In Windows Explorer, double-click **report_data.xlsx** to open the workbook in Excel, and then click **Enable Editing**. To make the data in the **Sheet1** worksheet suitable for use in mobile reports, each dataset must be moved to a separate worksheet.
4. On the **Home** tab, in the **Cells** group, click the **Insert** drop-down arrow, and then click **Insert Sheet**. A new sheet called **Sheet2** is added to the workbook.
5. In **Sheet1**, select the cell range **F4:G11**, right-click the highlighted area, and then click **Copy**.
6. In **Sheet2**, click cell **A1**, right-click the highlighted area, and then click **Paste**.
7. Repeat steps 4 and 6 to copy the data in **Sheet1!A1:D36** to a new worksheet called **Sheet3**.
8. Save the workbook and close Excel.

Add a dataset from Excel

1. On **10990C-MIA-SQL**, on the Start page, type **Microsoft SQL Server Mobile Report Publisher**, and then press Enter. The application opens with an empty report.
2. On the **Data** tab, click **Add data**, and then click **Excel**.
3. In the **Open** dialog box, navigate to **D:\Demofiles\Mod10**, and then double-click **report_data.xlsx**.
4. In the **Add data** window, select the **Sheet1**, **Sheet2**, and **Sheet3** check boxes, and then click **Import**. The data from the three sheets is imported to SSMRP.

- Click **Sheet1** and scroll right to demonstrate that all the data from the different regions on the worksheet has been imported, making it unsuitable for use in a mobile report.
- On the **Sheet1** tab, click the **cog** icon, and then click **Remove**.

Update a dataset from Excel

- On **10990C-MIA-CLI**, in File Explorer, navigate to `\\mia-sql\Mod10`, and then double-click **report_data.xlsx**.
- On **Sheet2**, click cell **B7**, type **Rebecca Green**, and then press Enter.
- Save the workbook and close Excel.
- On **10990C-MIA-SQL** in SSMRP, click **Sheet2**, then click **Refresh all data**. Notice that the dataset updates to reflect the changes you made in the Excel file.

Add a dataset from an SSRS shared dataset

- Click **Add data**, and then click **Report server**.
- If the **Connect to a server** dialog box appears, in the **Server address** box, type **mia-sql/Reports_SQL2**.
- Clear the **Use secure connection** check box, and then click **Connect**.
- In the **Add data from server** dialog box, click **mia-sql/Reports_SQL2**, click **MobileReports**, and then click **SalesByRegionAndYear**. The result set will be imported.

Create a mobile report

- In SSMRP, click **Layout**, then scroll down to the bottom of the element list (on the left of the screen), then click and drag **Simple data grid** to the top-left cell in the report layout grid.
- Using the resize handle in the bottom right of the data grid element, resize the element until it fills the entire grid.
- Click **Data**, then in the **Data properties** section, in the **Data for the grid view** box, select **Sheet3**, then click **Preview**. Observe that the data from the spreadsheet is displayed in the report. When you have finished, close SSMRP without saving any changes.

Check Your Knowledge

Question	
Which of the following cannot be used as a data source for a mobile report?	
Select the correct answer.	
<input type="checkbox"/>	A Reporting Services shared dataset
<input type="checkbox"/>	An Excel workbook
<input type="checkbox"/>	A query-driven dataset that's embedded in the report

Lab: Introduction to mobile reports

Scenario

Your workforce is increasingly mobile and needs a solution to view reports on a range of mobile devices. In this lab, you will design datasets for mobile reports, then create a simple report. You've been asked to create a mobile report that lists the names and job titles of employees in the Sales department.

Objectives

After completing this lab, you will be able to:

- Define datasets for mobile reports.
- Create mobile reports.
- Create KPIs.

Lab Setup

Estimated Time: 60 minutes

Virtual machine: **10990C-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa55w.rd**

Office 2016 is not compatible with SharePoint 2016 so, to use Excel in this demo, you must connect to 10990C-MIA-SQL from 10990C-MIA-CLI. The necessary network share is created by the setup script.

Exercise 1: Format data for a mobile report

Scenario

The sales manager has given you a spreadsheet that contains the contact information for the sales team, along with some other information that should not appear in the report.

The main tasks for this exercise are as follows:

1. Prepare the lab environment
2. Format the report data

► Task 1: Prepare the lab environment

1. Ensure the **10990C-MIA-CLI**, **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. Run **Setup.cmd** in the **D:\Labfiles\Lab10\Starter** folder as Administrator.

► Task 2: Format the report data

1. Log on to **10990C-MIA-CLI** as **Student** with the password **Pa55w.rd**.
2. With Excel, edit **\\mia-sql\Lab10\Starter\sales.xlsx** so that the sales staff details in **Sheet1!K9:L26** can be used in a mobile report. Use the user name **Adventureworks\student** and the password **Pa55w.rd** to connect to the network share. When you have finished, save your changes and close Excel.

Results: At the end of this exercise, you will have defined datasets for your new report.

Exercise 2: Create a mobile report

Scenario

Now that you have suitably formatted data, you will use the data to create a simple mobile report from an Excel source. When the report is working, you will change the data source to a Reporting Services shared dataset.

The main tasks for this exercise are as follows:

1. Import Excel data to a report
2. Add a report element
3. Use a shared dataset

► Task 1: Import Excel data to a report

- On **10990C-MIA-SQL**, use SSMRP to import the data from **D:\Labfiles\Lab10\Starter\sales.xlsx**. Do not import any datasets that you do not need.

► Task 2: Add a report element

- Connect the sales staff information that you imported to a **Simple Data Grid**. Preview the report when you have finished, then return to design view.

► Task 3: Use a shared dataset

- Import the **SalesEmployees** shared dataset from the **mia-sql/Reports_SQL2** SSRS server. Change the data source for the **Simple Data Grid** control to use the shared dataset. Do not include the **BusinessEntityID** column in the output. Preview the report, then close SSMRP without saving changes.

Results: At the end of this exercise, you should be able to create mobile reports from Excel or Reporting Services datasets.

Exercise 3: Create KPIs

Scenario

You have been asked to create a mobile KPI that will track the number of production line stoppages in the Adventure Works factory. Because this data is not available in an Excel file or from the database, you will create a manually maintained KPI.

You will also create a KPI from a shared dataset.

The main tasks for this exercise are as follows:

1. Create a manual KPI
2. Create a KPI from a shared dataset

► Task 1: Create a manual KPI

1. Use Internet Explorer to create a new KPI in the root folder of **mia-sql/Reports_SQL2**. The KPI should have the following properties:
 - **KPI name:** Production Line Stoppages
 - **Value format:** General

- **Value:** Set manually, 5
 - **Goal:** Not set
 - **Status:** Set manually, -1 (bad)
 - **Trend set:** Set manually, 5;0;1;0;0;6;5;1
2. Leave Internet Explorer open for the next task.
- **Task 2: Create a KPI from a shared dataset**
1. Create another mobile KPI. The KPI should have the following properties:
 - **KPI name:** Total Sales
 - **Value format:** Abbreviated currency...
 - **Value:** Dataset field, SalesByStateByMonth.ActualSales
 - **Goal:** Dataset field, SalesByStateByMonth.TargetSales
 - **Status:** Set manually, 1 (good)
 - **Trend set:** Not set
 2. Observe the new mobile KPI in the **KPIs Section**, then close Internet Explorer.

Results: At the end of this exercise, you will have created KPIs on the server.

Question: If you wanted to add the data from Sheet1 of sales.xlsx to the report that you created in this lab, how would you do it?

Module Review and Takeaways

In this module, you have learned about the fundamentals of Reporting Services mobile reports. You have learned about how to prepare data for use in mobile reports, and how to import data sources and link them to elements in mobile reports.

Review Question(s)

Question: Do you prefer working in Report Designer/Report Builder or in SQL Server Mobile Report Publisher? What are the reasons for your choice?

MCT USE ONLY. STUDENT USE PROHIBITED

Module 11

Developing Mobile Reports

Contents:

Module Overview	11-1
Lesson 1: Designing and publishing mobile reports	11-2
Lesson 2: Drillthrough in mobile reports	11-12
Lab: Developing mobile reports	11-18
Module Review and Takeaways	11-23

Module Overview

In this module, you will learn about the element types that you can add to your Microsoft® SQL Server® Reporting Services mobile reports. You will also learn about working with dataset parameters, and how to add drillthrough actions to your reports.

Objectives

At the end of this module, you should be able to:

- Describe how to design and publish mobile reports.
- Explain how to drill through from mobile reports.

Lesson 1

Designing and publishing mobile reports

When you design mobile reports, there's a large range of report elements that you can select to display or visualize your data. By defining phone and tablet views, you also customize the view of your report that users on different devices will see—there's also a report master view. When you have designed your report, you publish it to a Reporting Services instance.

Lesson Objectives

At the end of this lesson, you should be able to:

- Describe mobile report navigators.
- Describe mobile report visualizations.
- Explain views for different devices.
- Publish mobile reports.
- Explain how to work with parameters.

Navigators

Navigators are controls that filter the values displayed on a report to a subset of the dataset, either by selecting a range of values or an individual value. A report that includes more than one navigator might be used to create multiple filters. Three types of navigator are available in SQL Server Mobile Report Publisher (SSMRP):

- **Time navigator:**
 - For time series data
- **Selection list:**
 - For lists or hierarchies
- **Scorecard grid:**
 - For lists including a score

- **Time navigator.** Use to filter report data by a time range. By default, any dataset in the report with a date and time column is filtered by the time navigator. If a dataset contains more than one date and time column, the first column is used for filtering. You use time navigators to filter time series data.
- **Selection list.** Use to filter report data by a string or a numeric value. You configure a key column and a label column from your source data for the selection list—these can be the same column. After the key column is defined, you link other datasets in the report to be filtered on the key. Values that match the selected key will be displayed on the report. The key data for a selection list might be hierarchical. You use selection lists for presenting list or hierarchical data.
- **Scorecard grid.** Like a selection list, you use a scorecard grid to filter report data by a string or a numeric value; the key column, label column, and filtered datasets are defined in the same way as for a selection list. A scorecard grid also displays value columns and score indicators for each row; by default, two scores are displayed (Status and Trend), but you add or remove scores and values as required by your report. You use scorecard grids for representing score values, like key performance indicators (KPIs) in a compact view.

For more information about configuring mobile report navigators, see the topic *Add navigators to Reporting Services mobile reports* in Microsoft Docs:

 **Add navigators to Reporting Services mobile reports**

<https://aka.ms/Gghl2e>

Visualizations

Visualizations are report elements that display data from one or more report datasets. They are not interactive, although the data they display can be filtered by one or more navigators on the report. The following types of visualization are available:

- Gauges
- Charts:
 - Time chart
 - Category chart
 - Totals chart
 - Comparison chart
- Pie and funnel charts
- TreeMap
- Maps
- DataGrids

- **Gauges.** Gauges display a single numeric value, or a numeric value in comparison to a second numeric value. SSMRP provides 11 types of gauge. You configure gauges to carry out aggregation calculations if the source dataset contains more than one row; the aggregation operators that are available include sum, count, average, minimum, and maximum. Some gauge types can also be configured to show RAG status, or a trend indicator.
- **Charts.** Mobile report charts are configured in a similar way to charts in Microsoft Excel®. One or more data series define the x-axis and the y-axis. The following chart types are available:
 - **Time chart.** The x-axis is the first date and time column in the source dataset.
 - **Category chart.** The x-axis is defined by a grouping of data values.
 - **Totals chart.** The sum of each group in the main data series is presented as a bar on the chart. You configure groups based on rows or on columns in the source dataset.
 - **Comparison chart.** Time charts, category charts, and totals charts can optionally be configured with a secondary data series against which the values from the main series are compared.
- **Pie and funnel charts.** A segmented view of the dataset that can be configured by rows or by columns.
- **TreeMap.** Values from the dataset are compared by relative area. Use color to indicate a comparison against a second series of data values. TreeMaps are configured to display a hierarchy, where the members of each group appear together on the grid.
- **Maps.** SSMRP includes 10 geographical maps; you upload custom maps using ESRI shapefiles. When you use a map visualization, rows in your dataset must match the values of the region names in the map definition. Maps are used as the basis of three different types of visualization:
 - **Gradient heat map.** Data series values are mapped to a color gradient that's used to shade map regions.
 - **Bubble map.** Data series values control the radius of a circle shown over the center of each map region.

- **Range stop heat map.** The main data series is compared against a secondary data series. You define a gradient that determines whether a region will be red, amber, or green, based on the main series data value as a percentage of the secondary series data value.
- **DataGrids.** DataGrids show data in a tabular form. Three types of DataGrid are available:
 - **Simple DataGrid.** A simple tabular representation of the dataset.
 - **Indicator DataGrid.** A tabular representation of the dataset, with one or more indicator columns for each row, based on the delta between two values in the source dataset. You configure the indicator column to use one of several visualizations.
 - **Chart DataGrid.** A tabular representation of the dataset. One or more indicators can be included—as in an indicator DataGrid. The DataGrid might also include one or more chart columns, which indicate trend data; the data for the chart is typically derived from a second dataset.

For more information about configuring visualizations in mobile reports, see the topic *Add visualizations to Reporting Services mobile reports* in Microsoft Docs:



Add visualizations to Reporting Services mobile reports

<https://aka.ms/Czv3cr>

For more information about using maps in mobile reports, see the topic *Maps in Reporting Services mobile reports* in Microsoft Docs:



Maps in Reporting Services mobile reports

<https://aka.ms/Eec6qm>

Phone, tablet, and master views

By using SSMRP, you define the various views of a report suitable for viewing on different screen aspect ratios. When a report is opened on a device, it will display the report view that best matches its screen resolution and aspect ratio. Each view is defined by the size of the grid in the **Design** tab. Each view might have a different combination of report elements selected from the control instance gallery.

Master view

The master view is the default design grid that you see when you create a new mobile report. By default, the master view grid is 10 cells wide by five cells high. The master view is typically used by client browsers that use 16:9 or 16:10 aspect ratios. The master view is the only view where you add new elements to the report.

Tablet view

Tablet view is presented to devices that run in an aspect ratio close to 3:4—these are typically tablet devices. By default, the design grid for tablet view is six cells wide by eight cells high.

- Master view
 - 5x10 grid
 - New elements are added in the master view
- Tablet view
 - 8x6 grid
- Phone view
 - 4x6 grid
- Customizing views
 - Use the sliders in the **Design** tab

Phone view

Phone view is presented to devices that run in an aspect ratio close to 3:2—these are typically mobile devices that run in portrait mode. By default, the design grid for phone view is four cells wide by six cells high.

Customizing views

You change the grid sizes for all three report views using the column and row sliders in the upper right of the **Design** tab.

For more information on working with device views, see the topic *Lay out a Reporting Services mobile report for phone or tablet* in Microsoft Docs:

 **Lay out a Reporting Services mobile report for phone or tablet**

<https://aka.ms/Dori15>

Publishing mobile reports

You change the appearance of your mobile report by selecting a color palette. When you have finished the design of your mobile report, you save the report definition to a file, or publish it to an SSRS instance.

Color palette

You change the overall color palette of your report by using the color palette selector, which you find in the upper right of the **Layout** tab. You have 30 predefined palettes to choose from.

- Color palette
- Publishing:
 - Publish to a Reporting Services instance
 - File system—.rsmobile extension
 - SSMRP opens report definitions from files or direct from SSRS

Publishing

Mobile report definitions can be saved to a file in the Windows file system, by using an .rsmobile extension.

You publish your mobile reports to a Reporting Services instance that you have permission to write to. After publishing, mobile reports are accessible to users through the new Reporting Services portal.

You open a mobile report definition from the file system, or from SSRS by using SSMRP.



Note: The .rsmobile file that's written to the file system is a ZIP archive of several different files that make up the mobile report definition. By using suitable software, you open an .rsmobile file to better understand how mobile reports function.

Demonstration: Designing and publishing a report

In this demonstration, you will see how to:

- Configure mobile report elements.
- Design phone, tablet, and master views.
- Publish a report.

Demonstration Steps

Open the report

1. Ensure that the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Demofiles\Mod11** folder, run **Setup.cmd** as Administrator. In the **User Account Control** dialog box, click **Yes**.
3. On the Start page, type **Microsoft SQL Server Mobile Report Publisher**, and then press Enter. The application opens with an empty report.
4. Click the **Server connections** button on the application toolbar (at the upper left of the SSMRP window).
5. In the **Connect to a server** dialog box, in the **Server address** box, type **mia-sql/Reports_SQL2** clear the **Use secure connection** check box, and then click **Connect**.
6. Click the **Open mobile report** button on the application toolbar. Click **Open from file system**, then in the **Open** dialog, browse to **D:\Demofiles\Mod11**, click **demo1.rsmobile** then click **Open**.
7. Click **Data**, and observe that the report contains three datasets.

Design the report

1. In SSMRP, click **Settings**, in the **Report title** box, type **Regional Sales**.
2. On the **Layout** tab, click and drag a **Selection List** navigator from the gallery onto the upper left cell of the design grid.
3. Grab the sizer and expand the selection list to fill two grids across and three down.
4. In the **Visual properties** region, in the **Title** box, type **Region Selector**.
5. On the **Data** tab, under **Report elements**, click **Region Selector**.
6. In the **Data properties** region, in the **Keys** list, click **Sheet2**.
7. In the **Filter these details when a selection is made** region, select the **Sheet2** and **Sheet3** check boxes.
8. In the **Sheet3** list, click **Region**.
9. On the **Layout** tab, notice that the contents of the selection list now reflect the values from the dataset.
10. Click and drag a **Simple data grid** from the gallery onto the first empty cell at the upper left of the design grid to the right of the selection list.
11. Grab the sizer and expand the Data Grid to fill three grids across and three down.
12. In the **Visual properties** region, in the **Title** box, type **Sales Person**.
13. In the **Row numbers** list, click **Hide**.

14. On the **Data** tab, click **Sales Person**.
15. In the **Data properties** region, in the **Data for the grid view** list, click **Sheet2**.
16. On the **Preview** tab, in the **Region Selector** list, click on each region name and notice that the content of the **Sales Person** grid does not change, and then click the back button.
17. On the **Data** tab, click **Sales Person**.
18. In the **Data properties** region, click **Options**, select the **Region Selector** check box, and then click **Done**.
19. On the **Preview** tab, in the **Region Selector** list, click on each region name and notice that the content of the **Sales Person** changes according to the selected region, and then click the back button.
20. On the **Layout** tab, click and drag a **Radial gauge** from the gallery onto the first empty cell under the selection list in the design grid.
21. Grab the sizer and expand the gauge to fill two grids across and two down.
22. In the **Visual properties** region, in the **Title** box, type **Total Value**.
23. In the **Visual properties** region, click **Set ranges**.
24. In the **Neutral end** box, type **110.00 %**.
25. In the **Neutral start** box, type **90.00 %**, and then click **Done**.
26. On the **Data** tab, click **Total Value**.
27. In the **Data properties** region, in the **Main value** list, click **Sheet3**, and in the adjacent list, click **Actual**.
28. In the **Comparison value** list, click **Sheet3**, and in the adjacent list, click **Estimated**. Notice that the comparison value does not have to come from the same dataset as the main value.
29. On the **Layout** tab, on the design grid, click the **Total Value** gauge, click the cog icon at the upper right of the gauge, and then click **Copy**.
30. Click the upper left empty cell beneath the **Sales Person** data grid, click the cog icon at the upper right of the cell, and then click **Paste**.
31. In the **Visual properties** region, in the **Title** box, type **Regional Value**.
32. On the **Data** tab, click **Regional Value**.
33. In the **Data properties** region, next to the **Main value** lists, click **Options**, ensure that **Region Selector** is selected, and then click **Done**.
34. Next to the **Comparison value** lists, click **Options**, ensure that **Region Selector** is selected, and then click **Done**.
35. On the **Preview** tab, click on each region name in the **Region Selector** list, and notice that the **Total Value** gauge always displays the same value (because it is not filtered by the selection list). By contrast, the **Regional Value** gauge changes to reflect the currently selected region, and then click the back button.

Define tablet view

1. Demonstrate that the view in which you have been working up to this point is the master view.
2. On the **Layout** tab, in the upper right of the window, in the view drop-down list, click **Tablet**. Notice that the design grid is empty, and the element gallery has been replaced with the Control Instances gallery.
3. Click and drag the **Total Value** gauge from the gallery onto the upper left cell of the design grid. Grab the sizer and expand the gauge to fill three grids across and three down.
4. Click and drag the **Regional Value** gauge from the gallery onto the first empty cell to the upper right of the **Total Value** gauge. Grab the sizer and expand the gauge to fill three grids across and three down.
5. Click and drag the **Region Selector** list from the gallery onto the upper left empty cell below the **Total Value** gauge. Grab the sizer and expand the selection list to fill six grids across and five down.
6. On the **Preview** tab, click the **Regional Value** text to expand the **Regional Value** gauge to fill the screen.
7. Click the **Regional Value** text to return to tablet view, and then click the back button to return to the **Layout** tab.

Define mobile view

1. On the **Layout** tab, in the upper right of the window, in the view drop-down list, click **Phone**.
2. Customize the phone view by changing the value of the **Grid rows** slider to **5**.
3. Click and drag the **Region Selector** list from the gallery onto the upper left cell of the design grid. Grab the sizer and expand the selection list to fill four grids across and one down.
4. Click and drag the **Total Value** gauge from the gallery onto the upper left empty cell below the **Region Selector** list. Grab the sizer and expand the gauge to fill two grids across and two down.
5. Click and drag the **Regional Value** gauge from the gallery onto the first empty cell to the upper right of the **Total Value** gauge. Grab the sizer and expand the gauge to fill two grids across and two down.
6. Click and drag the **Sales Person** DataGrid into the upper left empty cell. Grab the sizer and expand the grid to fill four grids across and two down.
7. On the **Preview** tab, notice that the selection list is rendered as a drop-down box, and then click the back button to return to the **Layout** tab.

Publish the report

1. On the **Layout** tab, click the color palette selector, click the **Green** palette, and then click **Done**.
2. On the menu bar, click **Save mobile report as**.
3. In the **Save mobile report as** dialog box, click **Save to file system**.
4. In the **Save As** dialog box, in the **File name** box, type **D:\Demofiles\Mod11\Regional Sales.rsmobile**, and then click **Save**.
5. On the menu bar, click **Save mobile report as**.
6. In the **Save mobile report as** dialog box, click **Save to Server**, in the **Location** box, type **/MobileReports**, and then click **Save**.
7. In Internet Explorer, navigate to **http://mia-sql/Reports_SQL2**, click **MobileReports**, and then click **Regional Sales**. This is the master view of the report.

8. In File Explorer, navigate to **D:\Demofiles\Mod11**, right-click **browser_size.html**, point to **Open with** then click **Internet Explorer**.
9. In the message bar, click **Allow blocked content**.
10. In Internet Explorer, click **520x730 - tablet**. This starts a new browser window in a tablet aspect ratio.
11. In the new browser window, click **Regional Sales**. This is the tablet view of the report. Close the tablet view window.
12. Click **400x640 - phone**. This starts a new browser window in a phone aspect ratio. In the new browser window, click **Regional Sales**. This is the phone view of the report. Close all Internet Explorer windows. Leave SSMRP open for the next demonstration.

Parameters with shared datasets

There is no equivalent of the parameter selector in the report viewer for paginated reports in mobile reports; you use navigators to select and filter data in a mobile report. However, when designing mobile reports, you might need to work with datasets that require you to supply parameters.

This lesson covers the method for working with parameterized data sources in mobile reports.

- Use with shared datasets only
- Shared dataset parameters must have default values
- Link parameter values to report navigators
- Navigators must exist before they are mapped to parameters



Note: You pass values to dataset parameters and navigators when you access a mobile report with a URL. This is covered in more detail in the next lesson.

Mobile reports only use parameters when they interact with shared datasets that require parameters; parameters cannot be added to Excel data sources.

You might use a shared dataset with parameters when you need to use a common dataset between several reports, or to restrict the number of rows used in the mobile report when the base dataset is too large.

To use a dataset that accepts parameters in a mobile report, the dataset must be configured with a default value for each parameter. Datasets that do not have default parameter values cannot be added to a report in SSMRP.

When you have added a shared dataset with parameters to SSMRP, you configure the report element(s) that will be used to pass parameter values to the shared dataset. You only select parameter values from navigators that already exist in the report.



Note: If you are preparing a report as the target of a drillthrough from another mobile report, you might decide not to add navigators for the report parameters. In this scenario, you should consider hiding the report in tile view in the report properties on the Reporting Services web portal—without a means to supply parameters, the report will be empty if opened from the portal.

For more information about working with parameters in mobile reports, see the topic *Add parameters to a mobile report* | *Reporting Services* in Microsoft Docs:



Add parameters to a mobile report | Reporting Services

<https://aka.ms/Qyz6e5>

Demonstration: Working with parameters

Demonstration Steps

1. On **10990C-MIA-SQL**, start Internet Explorer then navigate to **http://mia-sql/reports_sql2**. Click **Adventureworks Sample Reports**, then right-click **EmployeeSalesDetail** then click **Edit in Report Builder**.
2. In the **Internet Explorer** dialog box, click **Allow**. If the **Connect to Report Server** dialog appears, click **Yes**.
3. When Report Builder starts, click **Set Options**. In the **Shared Dataset Properties** window, on the **Parameters** tab, observe that the report has no default values set for parameter values. Click **Cancel**, then close Report Builder without saving any changes.
4. In Internet Explorer, close the **We're opening your dataset in Report Builder** dialog, then click **SQL Server Reporting Services**. Click **MobileReports** then right-click **EmployeeSalesDetailDefaults** then click **Edit in Report Builder**.
5. In the **Internet Explorer** dialog box, click **Allow**.
6. When Report Builder starts, click **Set Options**. In the **Shared Dataset Properties** window, on the **Parameters** tab, observe that the report has default values set for all parameter values. Click **Cancel**, then close Report Builder without saving any changes.
7. In Internet Explorer, close the **We're opening your dataset in Report Builder** dialog, then click **SQL Server Reporting Services**.
8. In SSMRP click the **Open mobile report** button on the application toolbar. Click **Open from file system**, then in the **Open** dialog, in the **File name** box, type **D:\Demofiles\Mod11\demo2.rsmobile** then click **Open**.
9. To attempt to add the **EmployeeSalesDetail** dataset (that has no defaults), click **Data**, then click **Add Data**. Click **Report Server** then click **mia-sql/reports_sql2**, then click **AdventureWorks Sample Reports**, then click **EmployeeSalesDetail**. Read the error message then click **OK**.
10. To add the version of the dataset with parameter defaults, click the **Up** arrow, then click **MobileReports**, then click **EmployeeSalesDetailDefaults**. After a few moments, the dataset is added to the report. Observe that the tab for the **EmployeeSalesDetailDefaults** dataset has an icon containing green braces; this indicates that the dataset has parameters. Click the **Cog** icon next to **EmployeeSalesDetailDefaults** then click **Param**.
11. On the **Set dataset parameters** page, click the down arrow on the **@ReportYear** row. Under **Year** click **SelectedItem**.
12. Click the down arrow on the **@ReportMonth** row. Under **Month** click **SelectedItem**.
13. Click the down arrow on the **@EmployeeID** row. Under **Sales Person** click **SelectedItem**, then click **Apply**.
14. In the **Report elements** section, click **Sales Data**. In the **Data properties** section, in the **Data for the grid view** box, select **EmployeeSalesDetailDefaults**.

15. Click **Preview**, then in the **Year** box select **FY2013**, then in the **Sales Person** box select **Michael Blythe**. Observe that the **Sales Data** element updates to reflect these choices. Change the values of the navigators again, then click the **Back** button.
16. Click **Settings**, then in the **Report title** box type **Sales_Detail**. On the menu bar, click **Save mobile report as**.
17. In the **Save mobile report as** dialog box, click **Save to Server**, in the **Location** box, type **/MobileReports**, and then click **Save**. Leave SSMRP and Internet Explorer open for the next demonstration.

Check Your Knowledge

Question	
Which of the views in SSMRP do you use to add new elements to a report?	
Select the correct answer.	
<input type="checkbox"/>	Tablet view
<input type="checkbox"/>	Phone view
<input type="checkbox"/>	Master view
<input type="checkbox"/>	All the above

Lesson 2

Drillthrough in mobile reports

You add drillthrough to mobile reports to use a report visualization as a link to another URL. You might use drillthrough to allow users to view detailed data that's associated with an aggregated measure, or to link to supporting information.

You also access mobile reports from a URL, to link to mobile reports from external applications—including paginated reports.

This lesson looks at how to add drillthrough to mobile reports, and also covers URL access to mobile reports.

Lesson Objectives

At the end of this lesson, you should be able to:

- Explain how to drill through from mobile reports.
- Access mobile reports with URLs.

Drillthrough

You can add a drillthrough to any data visualization element in a mobile report. The target of the drillthrough is as follows:

- Another mobile report on the same Reporting Services instance.
- A URL that you specify.



Note: If the target of a drillthrough is a mobile report on a different Reporting Services instance, you must use a custom URL.

- Add drillthrough to any visualization element to:
 - A mobile report (on the same Reporting Server)
 - A custom URL
- Use the **Drillthrough target** button
- Map navigator values in the source report to:
 - Parameters of target mobile report
 - Query string of a custom URL

You add a drillthrough by using the **Drillthrough target** button in the **Visual properties** of a report element on the **Layout** tab in SSMRP.



Note: You test custom URL drillthrough from the **Preview** tab in SSMRP. You can only test drillthrough to another mobile report on the same Reporting Services instance when you have published the report to the reporting server.

Mapping navigator values

When you select a mobile report as the target of a drillthrough, SSMRP examines the report definition. If the target report has navigators, you have the option of mapping navigator values from the source report to the parameters of the drillthrough target report.

When you select a custom URL as the target of a drillthrough, you have the option of mapping navigator values from the source report to the query string of the drillthrough target URL.

The scorecard grid and selection list navigators make the following properties available:

- **SelectedItem:** the key of the selected item.
- **SelectedItems:** the keys of selected items if multiselect is allowed on the control.

The time navigator makes the following properties available:

- **SelectedStartTime:** the start time of the selected date range.
- **SelectedEndTime:** the end time of the selected date range.
- **ViewportStartTime:** the start of the date range that's visible in the control.
- **ViewportEndTime:** the end of the date range that's visible in the control.
- **TimeUnit:** the name of the unit of the selected date range—such as year, month, or day.



Note: For example, if a time navigator showed the years between 2012 and 2017 (inclusive), and 2016 was selected, the properties would be:

- **SelectedStartTime:** 1 January 2016
- **SelectedEndTime:** 1 January 2017
- **ViewportStartTime:** 1 January 2012
- **ViewportEndTime:** 1 January 2017
- **TimeUnit:** year

For more information about using drillthrough in mobile reports, see the topic *Add drillthrough from a mobile report to other mobile reports or URLs* in Microsoft Docs:



Add drillthrough from a mobile report to other mobile reports or URLs

<https://aka.ms/Ve6bm3>

URL access to mobile reports

You use URL access to mobile reports to add links to mobile reports to applications, or drillthroughs from mobile reports on other Reporting Services instances.

Mobile report URLs are constructed from the following mandatory elements:

- The Reporting Services web portal base URL.
- The string **mobilereport**.
- The Reporting Services folder path to the report.
- The report name.

- Mobile report URLs built from:
 - The Reporting Services web portal base URL
 - The Reporting Services folder path to the report
 - The report name
- The URL might also have a query string, built from
 - A question mark (?)
 - One or more parameter and value pairs, (<parameter>=<value>), delimited by an ampersand (&)
 - Set navigators with <navigator name>.<property name>
 - Set dataset parameters with <dataset name>.<parameter name>
 - Parameter name must match the parameter format of the dataset source system

The URL can also have a query string that is built from:

- A question mark (?).
- One or more parameter and value pairs, (<parameter>=<value>), delimited by an ampersand (&).

Setting navigators

The query string might contain values for navigator selections in the form <navigator name>.<property name>. The property name depends on the navigator type.

Scorecard grid and selection list navigators make the following properties available:

- **SelectedItem**: the key of the selected item.
- **SelectedItems**: the keys of selected items if multiselect is allowed on the control, delimited by commas.

The time navigator makes the following properties available:

- **SelectedStartTime**: the start time of the selected date range.
- **SelectedEndTime**: the end time of the selected date range.
- **ViewportStartTime**: the start of the date range that's visible in the control.
- **ViewportEndTime**: the end of the date range that's visible in the control.
- **TimeUnit**: the name of the unit of the selected date range—such as year, month, or day.

The following example shows a mobile report URL accessing the **Products** mobile report in the **Mobile** folder on the **SSRS01** server. The value **2845** is passed for the selection list in the report:

Mobile URL—navigator

```
http://SSRS01/reports/mobilereport/Mobile/Products?SelectionList.SelectedItem=2845
```


Setting dataset parameters

The query string might also contain parameters for shared datasets in the form <dataset name>.<parameter name>. The format of the parameter name must match the query syntax of the source system of the shared dataset—for example, parameters for SQL Server database engine queries must begin with an "at" symbol (@).

The following example shows a mobile report URL accessing the **Logs** mobile report in the **Mobile** folder on the **SSRS02** server. The value **APPSRV01** is passed for the **ServerName** parameter of the **LogData** dataset that runs a SQL Server query:

Mobile URL—dataset parameter

```
http://SSRS02/reports/mobilereport/Mobile/Logs?LogData.@ServerName=APPSRV01
```

 **Note:** Typically, you should use navigator parameters instead of dataset parameters when the target report has both—otherwise a conflict might occur between a dataset parameter and navigator selections. Reporting Services resolves the conflict by applying the dataset parameter regardless of any navigator selections, even if the navigator is updated.

Demonstration: Using drillthrough

In this demonstration, you will see how to:

- Configure drillthrough to a mobile report.
- Configure drillthrough to a custom URL.
- Compose a URL to access a mobile report.

Demonstration Steps

Drillthrough to a mobile report

1. Click the **Open mobile report** button on the application toolbar. Click **Open from file system**, then in the **Open** dialog, browse to **D:\Demofiles\Mod11**, click **demo3.rsmobile** then click **Open**.
2. On the **Layout** tab, click the number gauge on the report named **Mobile Report Drill**, then in the **Visual properties** region, click **Drillthrough target**, then click **Mobile report**. Click **mia-sql/reports_sql2**, then click **MobileReports**, then click **Time Navigator Example**. After a few seconds the parameter list is displayed.
3. In the **Configure target report** dialog, click the down arrow on the **SelectedStartTime** row then in the **Default value** box type **2013-08-01T00:00:00** then press Enter.
4. Click the down arrow on the **SelectedEndTime** row then in the **Default value** box type **2013-09-01T00:00:00** then press Enter.
5. Click the down arrow on the **ViewportStartTime** row then in the **Default value** box type **2013-01-01T00:00:00** then press Enter.
6. Click the down arrow on the **ViewportEndTime** row then in the **Default value** box type **2014-01-01T00:00:00** then press Enter.
7. Click the down arrow on the **TimeUnit** row then in the **Default value** box type **month** then press Enter, then click **Apply**.
8. On the menu bar, click **Save mobile report as**. In the **Save mobile report as** dialog box, click **Save to Server**.
9. In the **New report name**, type **Drillthrough**.
10. In the **Location** box, type **/**, and then click **Save**.
11. In Internet Explorer, navigate to **http://mia-sql/Reports_SQL2**, then click **Drillthrough**. If **Drillthrough** is not visible, click **Refresh** then try again.
12. In the Drillthrough report, click **31,490,208.00** in the **Mobile Report Drill** element. If a warning is displayed that Internet Explorer blocked a pop-up window, click **Allow Once**, then click **31,490,208.00** in the **Mobile Report Drill** element. Observe that the drillthrough uses the current browser tab to view the **Time Navigator Example** report, and that the time navigator is displaying data for August 2013, with the viewport showing the all the months in 2013.

Drillthrough to a custom URL

1. In SSMP, on the **Layout** tab, click the number gauge on the report named **Custom URL Drill**, then in the **Visual properties** region, click **Drillthrough target**, then click **Custom URL**.
2. In the **Enter a URL to go to when this visualization is clicked** box, type:

```
http://mia-sql/test?value=
```

3. In the **Available parameters** list, click **{{ SelectionList.SelectedItem }}**. The completed URL should read:

```
http://mia-sql/test?value={{ SelectionList.SelectedItem }}
```

Click **Apply**, then click **Preview**.

4. In the report preview, in the **Employee** box, select **Linda Mitchell**, then click **31,490,208.00** in the **Custom URL Drill** element. The custom URL will open in Internet Explorer, and an HTTP 404 error message will be displayed (because the webpage cannot be found); however, observe that the URL query string ends with **276** (the EmployeeID value for Linda Mitchell in the report dataset).

Mobile report URL access

1. In SSMRP, click the back button, on the **Layout** tab, click the number gauge on the report named **Custom URL Drill**, then in the **Visual properties** region, click **Drillthrough target**, then click **Edit Drillthrough**.
2. In the **Enter a URL to go to when this visualization is clicked** box, type:

```
http://mia-sql/reports_SQL2/mobilereport/MobileReports/Tree Map  
Example?SelectionList.SelectedItems=Banks,Insurance
```

Click **Apply**, then click **Preview**.

3. In the report preview, click **31,490,208.00** in the **Custom URL Drill** element. The custom URL will open in Internet Explorer, on the **Tree Map Example** report. Observe that the **Banks** and **Insurance** categories are selected in the **Industry & Sector** selection list. When you have finished, close SSMRP and Internet Explorer without saving any changes.

Check Your Knowledge

Question	
<p>You want to use URL access to run a mobile report called New Customers in the Customer folder on the SSRS03/reports Reporting Services instance. The report contains a selection list titled Region; you want to construct your URL to open the report with the selection set to Asia & Pacific. Which of the following options meets this requirement?</p>	
Select the correct answer.	
<input type="checkbox"/>	http://SSRS03/reports/Customer/New Customers?SelectionList.SelectedItem=Asia & Pacific
<input type="checkbox"/>	http://SSRS03/reports/mobilereport/Customer/New Customers?SelectionList.SelectedItem=Asia & Pacific
<input type="checkbox"/>	http://SSRS03/reports/mobilereport/Customer/New Customers?Region.SelectedItem=Asia & Pacific
<input type="checkbox"/>	http://SSRS03/reports/mobilereport/Customer/New Customers?SelectionList.SelectedItems=Asia & Pacific, EMEA

Lab: Developing mobile reports

Scenario

Your workforce is increasingly mobile and needs a solution to view reports on a range of mobile devices. In this lab, you will design and publish mobile reports, using Reporting Services. You have been asked to create a mobile report that senior managers will use to monitor metrics that are related to activity in the Adventure Works Cycles call center.

Objectives

After completing this lab, you will be able to:

- Work with datasets that use parameters.
- Design mobile reports.
- Publish mobile reports.
- Add drillthrough to mobile reports.

Estimated Time: 120 minutes

Virtual machine: **10990C-MIA-SQL**

User name: **ADVENTUREWORKS\Student**

Password: **Pa55w.rd**

Exercise 1: Add a dataset with parameters

Scenario

You decide to use the data-first approach to build this mobile report, so you will import datasets to SSMRP before adding visual elements to the report. You have already added some datasets to the report.

You need to add the **PhoneSalesCommission** dataset from the **AdventureWorks Sample Reports** folder on the **mia-sql/reports_SQL2** Reporting Services instance to the report. The dataset accepts three parameters, @StartDate, @EndDate, and @BusinessEntityId.

- @StartDate and @EndDate will be hard-coded to restrict the data to 2017 only.
- @BusinessEntityId will be linked to a navigator later in the lab.

The main tasks for this exercise are as follows:

1. Prepare the lab environment
2. Add the dataset

► Task 1: Prepare the lab environment

1. Ensure the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. Run **Setup.cmd** in the **D:\Labfiles\Lab11\Starter** folder as Administrator.

► Task 2: Add the dataset

1. Open the **lab01.rsmobile** report from **D:\Labfiles\Lab11\Starter** using SSMRP.
2. Add the **PhoneSalesCommission** dataset from the **AdventureWorks Sample Reports** folder on the **mia-sql/reports_SQL2** Reporting Services instance to the report.

3. Configure the date parameters for the dataset with hard-coded values restricting the data returned by the report to 2017. Leave SSMRP open for the next exercise.

Results: At the end of this exercise, you will have imported a parameterized dataset into your report.

Exercise 2: Design a mobile report

Scenario

In this exercise, you will connect the datasets in the report to navigators and visualizations.

The main tasks for this exercise are as follows:

1. Add a time navigator
2. Add a pie chart
3. Add a selection list
4. Add numbers
5. Add a progress bar

► Task 1: Add a time navigator

1. On the report master view, add a time navigator that is five cells high and three cells wide. The time navigator should have the following visual properties:
 - Time levels: **Years, Months, Weeks**
 - Time range presets: **All**
 - Number format: **General**
 - Visualization type: **Step Area**
 - Show comparison delta: **On**
 - Value direction: **Lower values are better**
2. On the **Data** tab, connect **Time Navigator 1** to the **AbandonedCalls** dataset. Use **Calls** and **AbandonedCalls** as the measures for the series for the background chart. Use **AbandonedCalls** as the measure for the comparison series. Preview the report when you have finished configuring the time navigator.

► Task 2: Add a pie chart

1. On the report master view, add a pie chart, adjacent to the time navigator, that is five cells high and three cells wide. The pie chart should have the following visual properties:
 - Title: **Abandoned Calls**
 - Number Format: **General**
 - Series Visualization: **Donut**
 - Value Display Mode: **Percentage on Chart**
 - Show Legend: **On**
2. On the **Data** tab, connect **Abandoned Calls** to the **AbandonedCalls** dataset. Use **Calls** and **AbandonedCalls** as the measures for the main series for the background chart. Preview the report when you have completed configuring the pie chart.

► Task 3: Add a selection list

1. On the report master view, add a selection list, adjacent to the pie chart, that is five cells high and two cells wide. The selection list should have the following visual property:
 - Title: **Call Center Staff**
2. On the **Data** tab, connect **Call Center Staff** to the **SalesEmployees** dataset. Use **BusinessEntityID** as the list of keys and **Employee** as the list of labels. Configure **Call Center Staff** to filter the **PhoneSalesCommission** and **SalesTarget** tables, on the **BusinessEntityId** column.
3. Amend the **PhoneSalesCommission** dataset to use the values from the Call Centre Staff navigator as the parameter value for the **@BusinessEntityId** parameter.

► Task 4: Add numbers

1. On the report master view, add a number, adjacent to the selection list, that is one cell high and two cells wide. The number should have the following visual properties:
 - Title: **Total Sales**
 - Number Format: **Abbreviated Currency**
2. Add another number, below the first number, that is one cell high and two cells wide. This number should have the following visual properties:
 - Title: **Sales Target**
 - Number Format: **Abbreviated Default Currency**
3. On the **Data** tab, connect **Total Sales** to the **PhoneSalesCommission** dataset. Use **SalesAmount** as the number value.
4. Connect **Sales Target** to the **SalesTarget** dataset. Use **SalesTarget** as the number value.

► Task 5: Add a progress bar

1. On the report master view add a progress bar, in the empty space adjacent to the pie chart, that is two cells high and two cells wide. The selection list should have the following visual properties:
 - Title: **Sales Target Progress**
 - Delta Label: **Percentage from target**
 - Range Stops:
 - Maximum: **150%**
 - Neutral End: **100%**
 - Neutral Start: **90%**
 - Minimum: **0%**
2. On the **Data** tab, connect the main value of **Sales Target Progress** to the **PhoneSalesCommission** dataset, using **SalesAmount** as the main value. Connect the comparison value of **Sales Target Progress** to **SalesTarget**, using **SalesTarget** as the comparison value. Ensure that both values are filtered by **Call Center Staff**. Preview the report when you have finished configuring the progress bar. Leave SSMRP open for the next exercise.

Results: At the end of this exercise, you will have created the master view of a mobile report definition, and linked the view to datasets.

Exercise 3: Publish a mobile report

Scenario

The mobile report you have created is almost ready for publication. In this exercise, you will configure phone and tablet views of the report, then select a color palette, before publishing the report to the SSRS server.

The main tasks for this exercise are as follows:

1. Define the tablet view
2. Define the phone view
3. Select a palette and publish the report
4. View the published report

► Task 1: Define the tablet view

1. In SSMRP, create a tablet view for the report. Add the following elements to the report:
 - **Time navigator 1:** height 5 cells, width 6 cells
 - **Total Sales:** height 3 cells, width 3 cells
 - **Abandoned Calls:** height 3 cells, width 3 cells
2. Preview tablet view when you have completed configuring it.

► Task 2: Define the phone view

1. Add the following elements to the phone view of the report:
 - **Call Center Staff:** height 4 cells, width 4 cells
 - **Sales Target Progress:** height 2 cells, width 4 cells
2. Preview phone view when you have completed configuring it.

► Task 3: Select a palette and publish the report

1. Change the report to use the **Steel** palette. Change the report title to **Call Center Tracking**.
2. Save the report to the root folder of the **mia-sql/Reports_SQL2** reporting server. Leave SSMRP open for the next exercise.

► Task 4: View the published report

1. In Internet Explorer, go to **mia-sql/Reports_SQL2**, to view the **Call Center Tracking**.
2. Use the browser to open **browser_size.html** in **D:\Labfiles\Lab11\Starter**. Use the buttons in the page to view the report in mobile and tablet views. Close all Internet Explorer windows when you have finished using the report.

Results: At the end of this exercise, you will have published a mobile report to the root folder on the **http://mia-sql/Reports_SQL2** reporting server.

Exercise 4: Add a drillthrough to a custom URL

Scenario

Now that you have a functioning version of the report, you want to extend it by adding a drillthrough from the **Total Sales** element to the **Sales_By_Region** report in the **AdventureWorks Sample Reports** folder.

The main tasks for this exercise are as follows:

1. Add the drillthrough

► Task 1: Add the drillthrough

1. In SSMP, edit the definition of the **Total Sales** to add a drillthrough to the **Sales_By_Region** report in the **AdventureWorks Sample Reports** folder.
2. Test the drillthrough from the report preview. Close Internet Explorer and SSMP when you have finished, discarding any changes.

Hint: You can use the URL displayed in Internet Explorer when you view the report from the Reporting Services web portal as the custom URL.

Results: At the end of this lab, you should be able to add drillthroughs to mobile reports.

Check Your Knowledge

Question	
Which type of visual component of mobile reports do you use to filter data?	
Select the correct answer.	
<input type="checkbox"/>	Gauges
<input type="checkbox"/>	Graphs
<input type="checkbox"/>	Maps
<input type="checkbox"/>	Navigators
<input type="checkbox"/>	DataGrids

Module Review and Takeaways

In this module, you have worked with all the key elements of Reporting Services mobile reports. You have learned how to configure and customize navigators and visualizations, and how to create and publish views for different types of mobile devices.

You have learned how to work with datasets that accept parameters, and how to configure drillthrough in mobile reports.

Review Question(s)

Question: Is mobile reporting important to your organization? How will SQL Server mobile reports change your approach to publishing reports?

Course Evaluation

Course Evaluation

- Your evaluation of this course will help Microsoft understand the quality of your learning experience.
- Please work with your training provider to access the course evaluation form.
- Microsoft will keep your answers to this survey private and confidential and will use your responses to improve your future learning experience. Your open and honest feedback is valuable and appreciated.

Your evaluation of this course will help Microsoft understand the quality of your learning experience.

Please work with your training provider to access the course evaluation form.

Microsoft will keep your answers to this survey private and confidential and will use your responses to improve your future learning experience. Your open and honest feedback is valuable and appreciated.

Module 1: Introduction to Reporting Services

Lab: Exploring Reporting Services

Exercise 1: Exploring reports

► Task 1: Prepare the lab environment

1. Ensure the **MT17B-WS2016-NAT**, **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab01\Starter** folder, right-click **Setup.cmd** and click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**, and then wait for the script to finish.
4. At the command prompt, type **R**, press Enter then wait for the script to finish and close.

► Task 2: View the design of the AdventureWorksDW data warehouse

1. On the taskbar, click **Microsoft SQL Server Management Studio 17**.
2. In the **Connect to Server** dialog box, in the **Server name** text box, type **MIA-SQL**, in the **Authentication** list, click **Windows Authentication**, and then click **Connect**.
3. In Object Explorer, expand **Databases**, expand **AdventureWorksDW**, and then expand **Database Diagrams**.
4. In the **Microsoft SQL Server Management Studio** dialog box, click **Yes**.
5. Double-click **dbo.Reseller Sales**.

► Task 3: View sales data from the AdventureWorksDW data warehouse

1. On the **File** menu, point to **Open**, and click **File**.
2. In the **Open File** dialog box, go to **D:\Labfiles\Lab01\Starter**, and then double-click **SalesQuery.sql**.
3. Review the query, and click **Execute**.

Note that the total sales for the employee **David Campbell** is **\$106,380.438**.

► Task 4: Display the total reseller sales from a data model

1. In SQL Server Management Studio Object Explorer, click **Connect**, and then click **Analysis Services**.
2. In Server name, type **MIA-SQL**, and then click **Connect**.
3. In Object Explorer, expand the **MIA-SQL** Analysis Server, expand **Databases**, expand **AdventureWorksDW**, and then expand **Cubes**.
4. Right-click **AdventureWorks**, and then click **Browse**.
5. On the **Metadata** tab, expand **Measures**, and expand **Fact Reseller Sales**.
6. Drag **Sales Amount** to the query pane, and then click **Execute Query**.

► Task 5: Display the reseller sales for each employee

1. On the **Metadata** tab, expand **Dim Employee**.
2. Drag **Full Name** to the query pane to the left of the Sales Amount column, and then click **Execute Query**. A blue bar should highlight the location.

3. Resize the **Full Name** column so that you can see the names.
4. You can now see the total sales for each employee. Note the sales amount for **David R. Campbell**.

► **Task 6: Filter the results**

1. On the **Metadata** tab, expand **Order Date** dimension, expand **Order Date.Calendar**.
2. Drag **Day Number Of Month** to the dimension filter pane.
3. Click the **Filter Expression** column, and click the drop-down button.
4. Expand **All**, expand **2013**, select **April**, and then click **OK**.
5. Click **Execute Query** to view the results.
6. Note that the total sales for the employee **David R. Campbell** is **\$106,380.438**.

► **Task 7: View the data in a report**

1. On the taskbar, click **Internet Explorer**.
2. In the address bar, type **mia-sql/Reports_SQL2/browse**, and then press Enter.
3. Click the **AdventureWorks Sample Reports** folder.
4. In the **Employee_Sales_Summary** report, click the ellipses, and then click **Manage**.
5. In the toolbar, click **Edit in Report Builder**.
6. If the **Internet Explorer** dialog box appears, click **Allow**.
7. If the **Connect to Report Server** dialog box appears, click **Yes**.
8. On the **Home** ribbon, in the **Views** group, click **Run**.
9. Note that you can filter by **Employee**, **Date**, and **Category**. The default view shows all categories for **David Campbell** in **November 2007**.
10. In the **Report Month** list, click **April**.
11. In the **Report Year** box, type **2013**, and then click **View Report**.
12. Scroll down and view the sales data for David Campbell.
13. Note that the 2013 row of the second table on the left lists total sales of \$106,380.
14. Experiment with changing the filters at the top of the report and click **View Report** to refresh the data.
15. Close all open windows, without saving any changes.

Results: After completing this exercise, you will have:

Explored a data warehouse.

Explored a data model.

Explored a report.

Exercise 2: Reporting Services configuration

► Task 1: Use Reporting Services Configuration Manager

1. Click **Start**, type **Reporting**, and then click **Reporting Services Configuration Manager**.
2. In the **User Account Control** dialog box, click **Yes**.
3. In the **Reporting Services Configuration Connection** window, in the **Server name** box, type **MIA-SQL**, confirm that the **Report Server Instance** box is set to **SSRS**, and then click **Connect**.

On the **MIA-SQL\SSRS** page, note the value of:

The **Edition** property.

- The **Report Server Mode** property.
4. On the **Web Service URL** page, note the value of the **URLs** property.
 5. On the **Database** page, note the value of the **SQL Server Name** property.
 6. On the **Web Portal URL** page, note the value of the **URLs** property.

► Task 2: Use the web portal to view configuration

1. In Reporting Services Configuration Manager, on the **Web Portal URL** page, click the hyperlink for the web portal URL (the **URLs** property).

In the web portal in Internet Explorer, click the **Settings** icon in the upper right (the cog icon), then click **Site settings**.

2. On the **Site settings** page, on the **General** subpage, note the value of the **Report timeout (default)** property.
3. On the **Schedules** subpage, note the number of defined schedules.
4. On the **Security** subpage, note the number of security groups. When you have finished, close all open windows without saving any changes.

Results: At the end of this exercise, you will have viewed Reporting Services configuration with:

Reporting Services Configuration Manager.

Reporting Services web portal.

MCT USE ONLY. STUDENT USE PROHIBITED

Module 2: Reporting Services Data Sources

Lab A: Configuring data access with Report Builder

Exercise 1: Configuring data access

► Task 1: Prepare the lab environment

1. Ensure that the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab02\Starter** folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes** to run the command file, and then wait for the script to finish.

► Task 2: Create a shared data source

1. Start Internet Explorer, and then browse to **mia-sql/Reports_SQL2**.
2. On the **SQL Server Reporting Services** page, click the + icon, and then click **Data Source**.
3. On the **New data source** page, in the **Name** box, type **AdventureWorksDW**.
4. In the **Type** box, verify that **Microsoft SQL Server** is selected.
5. In the **Connection string** box, type the following code:

```
data source=MIA-SQL;initial catalog=AdventureWorksDW
```
6. Under **Credentials**, click **Using the following credentials**.
7. In the **Type of credentials** list, click **Database user name and password**.
8. In the **User name** box, type **reporting**.
9. In the **Password** box, type **Pa55w.rd**, and then click **Test connection**. Note that the test succeeds.
10. Click **Create**.

► Task 3: Create a shared dataset

1. In Internet Explorer, click **Home**.
2. On the toolbar, click +, then click **Dataset**.
3. In the **Internet Explorer** dialog box, click **Allow**.
4. In the **New Report or Dataset** dialog box, click **Browse other data sources**.
5. In the **Select Data Source** dialog box, click **AdventureWorksDW**, and then click **Open**.
6. In the **New Report or Dataset** dialog box, click **Create**.
7. In the **Enter Data Source Credentials** dialog box, in the **Password** box, type **Pa55w.rd**.
8. Select the **Save password with connection** check box, and then click **OK**.
9. In the **Database view** pane, expand **Views**, select **vAllSales**, and then click **Run Query**.
10. On the **File** menu, click **Save**.

11. In the **Save As Dataset** dialog box, double-click **Builder**.
12. In the **Name** box, type **AllSales**, and then click **OK**.
13. In Internet Explorer, browse to **mia-sql/Reports_SQL2/browse/Builder** to verify that the dataset has been created.
14. Close Report Builder, and then close Internet Explorer.

Results: At the end of this lab, a shared data source and a shared dataset will exist in the Builder folder of the Reporting Services portal at **http://mia-sql/Reports_SQL2**.

Lab B: Configuring data access with Report Designer

Exercise 1: Configuring data access

► Task 1: Prepare the lab environment

1. Ensure that the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab02\Starter** folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes** to run the command file, and then wait for the script to finish.

► Task 2: Create a shared data source

1. On the taskbar, click **Visual Studio 2015**.
2. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, browse to **D:\Labfiles\Lab02\Starter\Project**, click **Project.sln**, and then click **Open**.
4. If a security warning appears, click **OK**.
5. In Solution Explorer, right-click **Shared Data Sources**, and then click **Add New Data Source**.
6. In the **Shared Data Source Properties** dialog box, in the **Name** box, type **AdventureWorksDW**.
7. In the **Type** box, verify that **Microsoft SQL Server** is selected, and then click **Edit**.
8. In the **Connection Properties** dialog box, in the **Server name** box, type **MIA-SQL**.
9. In the **Authentication** list, click **SQL Server Authentication**.
10. In the **User name** box, type **reporting**, in the **Password** box type **Pa55w.rd**, and then select the **Save my password** check box.
11. In the **Select or enter a database name** list, click **AdventureWorksDW**, and then click **Test Connection**.
12. In the **Test results** dialog box, click **OK**.
13. In the **Connection Properties** dialog box, click **OK**.
14. In the **Shared Data Source Properties** dialog box, on the **Credentials** page, confirm that the credentials that you entered in step 9 are being used, and then click **OK**.
15. In Solution Explorer, expand **Shared Data Sources**, and then verify that the definition for the data source has been created.
16. Right-click **AdventureWorksDW.rds**, and then click **Deploy**.

► Task 3: Create a shared dataset

1. In Solution Explorer, right-click **Shared Datasets**, and then click **Add New Dataset**.
2. In the **Shared Dataset Properties** dialog box, on the **Query** page, in the **Name** box, type **AllSales**.
3. In the **Data source** list, ensure **AdventureWorksDW** is selected.

4. In the **Query** box, type the following code, and then click **OK**:

```
SELECT * FROM dbo.vAllSales;
```

5. In Solution Explorer, expand **Shared Datasets**, right-click **AllSales.rsd**, and then click **Deploy**.
6. On the taskbar, click Internet Explorer.
7. In Internet Explorer, browse to **mia-sql/Reports_SQL2**, click **Designer**, and verify that the data source and dataset have been created on the SSRS server.
8. Close Visual Studio, and then close Internet Explorer.

Results: At the end of this lab, a shared data source and a shared dataset will exist in the Designer folder of the Reporting Services portal at **http://mia-sql/Reports_SQL2**.

Module 3: Creating Paginated Reports

Lab: Creating reports

Exercise 1: Use the Report Wizard—Report Designer

► Task 1: Prepare the environment

1. Ensure the **10990C -MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C -MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab03\Starter** folder, right-click **Setup.cmd** and click **Run as administrator**.
3. Click **Yes** when prompted to confirm you want to run the command file, and wait for the script to finish.

► Task 2: Use Report Wizard

1. Start Visual Studio 2015, then on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, go to the **D:\Labfiles\Lab03\Starter** folder, click **Reports.sln**, and then click **Open**.
3. If the **Security Warning for Reports** dialog box appears, clear the **Ask me for every project in this solution** check box, and then click **OK**.
4. In Solution Explorer, right-click **Reports**, and then click **Add New Report**.
5. On the **Welcome to the Report Wizard** page, click **Next**.
6. On the **Select the Data Source** page, click **Next**.
7. On the **Design the Query** page, click **Query Builder**.
8. In the **Query Designer** dialog box, on the toolbar, click **Add Table**.
9. In the **Add Table** dialog box, click **DimProduct**, hold down the CTRL key, click **DimProductCategory**, click **DimProductSubcategory**, click **Add**, and then click **Close**.
10. In the **Query Designer** dialog box, in the upper pane, select the following columns, and then click **OK**:
 - DimProduct.ProductAlternateKey
 - DimProduct.EnglishProductName
 - DimProduct.StandardCost
 - DimProduct.ListPrice
 - DimProduct.DealerPrice
 - DimProductCategory.EnglishProductCategoryName
 - DimProductSubcategory.EnglishProductSubcategoryName
11. On the **Design the Query** page, click **Next**.
12. On the **Select the Report Type** page, ensure that **Tabular** is selected, and then click **Next**.
13. On the **Design the Table** page, select all the fields in the **Available fields** box, click **Details**, and then click **Next**.
14. On the **Completing the Wizard** page, in the **Report name** box, type **Product Listing**, and then click **Finish**.

15. Click the **Preview** tab to view the report. Leave Visual Studio 2015 open for the next exercise.

Results: At the end of this exercise, you should have created a report using the Report Wizard.

Exercise 2: Use the Report Wizard—Report Builder

► Task 1: Use Report Wizard

1. Click **Start** then type **Report Builder** then press Enter.
2. In Report Builder, in the **Getting Started** window, click **Table or Matrix Wizard**.
3. In the **New Table or Matrix** window, on the **Choose a dataset** page, select **Create a dataset**, then click **Next**.
4. On the **Choose a connection to a data source** page, click **Browse**.
5. In the **Select Data Source** window, in the **Name** box, type **http://mia-sql/ReportServer_SQL2**, then click **Open**.
6. Double-click **AdventureWorks Sample Reports**, then click **AdventureWorksDW**, then click **Open**.
7. On the **Choose a connection to a data source** page, click **Next**.
8. On the **Design a query** page, in the **Database view** pane, expand **Tables**, then expand **DimProduct** then select the following columns:
 - ProductAlternateKey
 - EnglishProductName
 - StandardCost
 - ListPrice
 - DealerPrice
9. Expand **DimProductCategory**, then select the **EnglishProductCategoryName** column.
10. Expand **DimProductSubcategory** and select the **EnglishProductSubcategoryName** column then click **Next**.
11. On the **Arrange fields** page, select all the columns in the **Available fields** box, then drag and drop them into the **Values** box, then click **Next**.
12. On the **Choose the layout** page, click **Next**.
13. On the **Preview** page, click **Finish**.
14. To preview the report, click **Run**.
15. Close Report Builder without saving any changes when you have finished.

Results: At the end of this exercise, you should have created a report using the Report Wizard.

Exercise 3: Creating and publishing a report—Report Designer

► Task 1: Create the report

1. In Visual Studio 2015, in Solution Explorer, right-click **Reports**, point to **Add**, and then click **New Item**.
2. In the **Add New Item - Reports** dialog box, click **Report**, in the **Name** box, type **Internet Sales Detail** and then click **Add**.
3. In Solution Explorer, right-click **Shared Datasets**, and then click **Add New Dataset**.
4. In the **Shared Dataset Properties** dialog box, in the **Name** box, type **InternetSales**, in the **Data source** list, click **AdventureWorksDW**, in the **Query** box, type the following code, and then click **OK** (the text of this query can be found in **D:\Labfiles\Lab03\Starter\Internet Sales.sql**):

```
SELECT fis.SalesOrderNumber, fis.SalesAmount, fis.TaxAmt, fis.OrderQuantity,
dp.EnglishProductName,
dpr.EnglishPromotionName, dc.FirstName, dc.LastName
FROM FactInternetSales AS fis
INNER JOIN DimCustomer AS dc ON fis.CustomerKey = dc.CustomerKey
INNER JOIN DimProduct AS dp ON fis.ProductKey = dp.ProductKey
INNER JOIN DimPromotion AS dpr ON fis.PromotionKey = dpr.PromotionKey
```

5. From the **Toolbox**, drag a **Table** item to the report design surface (if the Toolbox pane is not visible, click **View**, then click **Toolbox**).
6. In the **Dataset Properties** dialog box, in the **Name** box, type **InternetSales**, and then in the list of datasets, click **InternetSales**.
7. On the **Fields** page, change the **Field Name** values as shown in the following table, and then click **OK**:

Field Name	Field Source
Tax	TaxAmt
Product	EnglishProductName
Promotion	EnglishPromotionName

8. In the **Design** pane, click and drag the Table item that you just added to the left-hand side of the report layout, leaving some space at the top of the report.
9. In the **Report Data** pane, expand **Datasets**, expand **InternetSales**, and then drag each data item in turn to the table header. If the available cells are full, position the item on the right-side border of the table and a new cell will automatically appear (If the **Report Data** pane is not visible, click **View**, then click **Report Data**).
10. From the **Toolbox**, drag a text box to the report design surface, aligning it with the top of the table.
11. In the text box, type the text **Internet Sales Detail**. If necessary, change the size of the text box so that the full report title is displayed.
12. Click the **Preview** tab to view the report. Make a note of any columns that appear too wide, or too narrow. For example, you might choose to reduce the width of the **Sales Amount** and **Tax** columns and increase the width of the **Product** and **Promotion** columns.
13. Click the **Design** tab to return to design view and resize any columns that you noted in the last step.

14. Click the **Preview** tab to view the report.

► **Task 2: Publish the report**

1. In Solution Explorer, right-click the **Reports** project, then click **Properties**.
2. In the **Reports Property Pages** window, on the **General** page, update the values of the following properties, and then click **OK**:
 - **TargetDatasetFolder**: Designer
 - **TargetDataSourceFolder**: Designer
 - **TargetReportFolder**: Designer
 - **TargetServerURL**: http://mia-sql/ReportServer_SQL2
3. In Solution Explorer, right-click the **Reports** project, then click **Deploy**.
4. When deployment has finished, open Internet Explorer and go to http://mia-sql/reports_sql2.
5. On the **SQL Server Reporting Services** home page, click **Designer**.
6. In the **Designer** folder, click **Internet Sales Detail** to view your report.
7. Close Internet Explorer and Visual Studio 2015 when you have finished.

Results: At the end of this exercise, you should be able to:

Create a report from scratch by using Report Designer.

Customize elements of the report design.

Publish the report to Reporting Services.

Exercise 4: Creating and publishing a report—Report Builder

► **Task 1: Create the report**

1. Click **Start** then type **Report Builder** then press Enter.
2. In Report Builder, in the **Getting Started** window, click **Blank Report**.
3. In Report Data, right-click **Datasets**, and then click **Add Dataset**.
4. In the **Dataset Properties** dialog box, in the **Name** box, type **InternetSales**, select **Use a dataset embedded in my report**, and then click **New**.
5. In the **Data Source Properties** dialog box, in the **Name** box, type **AdventureWorksDW**, and then click **Use a connection embedded in my report**.
6. In the **Connection string** text box, type the following:

```
Data Source=MIA-SQL;Initial Catalog=AdventureWorksDW
```

7. In the **Credentials** pane, click **Use current Windows user. Kerberos delegation might be required**. Then click **OK**.

8. In the **Dataset Properties** dialog box, in the **Query** box, type the following (the text of this query can be found in **D:\Labfiles\Lab03\Starter\Internet Sales.sql**), and then click **Refresh Fields**:

```
SELECT fis.SalesOrderNumber, fis.SalesAmount, fis.TaxAmt, fis.OrderQuantity,
dp.EnglishProductName,
dpr.EnglishPromotionName, dc.FirstName, dc.LastName
FROM FactInternetSales AS fis
INNER JOIN DimCustomer AS dc ON fis.CustomerKey = dc.CustomerKey
INNER JOIN DimProduct AS dp ON fis.ProductKey = dp.ProductKey
INNER JOIN DimPromotion AS dpr ON fis.PromotionKey = dpr.PromotionKey
```

9. On the **Fields** page, change the **Field Name** values to match the following table, and then click **OK**.

Field Name	Field Source
Tax	TaxAmt
Product	EnglishProductName
Promotion	EnglishPromotionName

10. On the **Insert** menu, click **Table**, and then click **Insert Table**, then click anywhere on the report design surface to insert a new table.
11. In the **Report Data** pane, expand **Datasets**, expand **InternetSales**, and then drag each data item in turn to the table header. If the available cells are full, position the item on the right-side border of the table and a new cell will automatically appear.
12. Click the **Click to add title** text box.
13. In the text box, type **Internet Sales Detail**.
14. On the **Home** menu, click **Run**. Make a note of any columns that appear too wide, or too narrow. For example, you might choose to reduce the width of the **Sales Amount** and **Tax** columns and increase the width of the **Product** and **Promotion** columns.
15. Click **Design** to return to design view and resize any columns that you noted in the last step.
16. Click **Run** to view the report.

► Task 2: Publish the report

- In Report Builder, on the **File** menu, click **Save As**.
- In the **Name** box, type **http://mia-sql/ReportServer_SQL2** then click **Save**.
- Double-click **Builder** and then in the **Name** box, type **Internet Sales Detail**, and then click **Save**.
- If the **Save As Report** window appears, click **Yes**.
- Start Internet Explorer and go to the address **mia-sql/reports_sql2**.
- Click **Builder**, and then **Internet Sales Detail**. Check that the report has been correctly published.
- Close Internet Explorer, then close Report Builder.

Results: At the end of this exercise, you should be able to:

Create a report from scratch by using Report Builder.

Customize elements of the report design.

Publish the report to Reporting Services.

Module 4: Working with Reporting Services Data

Lab: Create a parameterized report

Exercise 1: Using parameters in Report Designer

► Task 1: Prepare the lab environment

1. Ensure the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab04** folder, right-click **Setup.cmd** and click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**, and then wait for the script to finish.

► Task 2: Add parameters to a report

Open the sales report

1. On the taskbar, click **Visual Studio 2015** to open it.
2. On the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In **Location**, navigate to **D:\Labfiles\Lab04\Starter\SalesReport**, and select **SalesReport.sln**.
4. Click **Open** and then double-click **SalesReport.rdl** is opened in design model.

Add two datasets to use as lookups

1. In the **Report Data** pane, right-click **Datasets** and click **Add Dataset**.
2. In **Name**, type **CountryLookup**.
3. Click **Use a dataset embedded in my report**.
4. Under **Query**, click **Import**.
5. Navigate to **D:\Labfiles\Lab04\Starter**, click **CountryRegionLookup.sql** and then click **Open**.
6. In **Data source**, select **AdventureWorksDW** from the drop-down list.
7. Click **OK**.
8. In the **Report Data** pane, right-click **Datasets** and click **Add Datasets**.
9. In **Name**, type **StateProvinceLookup**.
10. Click **Use a dataset embedded in my report**.
11. Under **Query**, click **Import**.
12. Navigate to **D:\Labfiles\Lab04\Starter**, click **StateProvinceLookup.sql** and then click **Open**.
13. In **Data source**, select **AdventureWorksDW** from the drop-down list.
14. Click **OK**.

Create the Country parameter

1. In the **Report Data** pane, right-click **Parameters**, and click **Add Parameter**.
2. In the **Report Parameter Properties** dialog box, in **Name**, type **CountryRegion**.
3. In **Prompt**, type **Country**.
4. In **Data type**, select **Text**.

5. In **Select parameter visibility**, select **Visible**.
6. In the left pane, click **Available Values**.
7. In **Select from one of the following options**, click **Get values from a query**.
8. In **Dataset**, select **CountryLookup** from the drop-down list.
9. In **Value field**, select **CountryRegionCode**.
10. In **Label field**, select **EnglishCountryRegionName**.
11. Click **OK**.
12. Click **Preview** to view the report.
13. Click the **Country** dropdown, then click **Australia**.
14. Click **View Report**.
15. Change the **Country** to **Canada**, then click **View Report**. Note the parameter will not change the contents of the report.
16. Click **Design**.

Create the StateProvince parameter

1. In the **Report Data** pane, right-click **Parameters**, and select **Add Parameter**.
2. In the **Report Parameter Properties** dialog box, in **Name**, type **StateProvince**.
3. In **Prompt**, type **State or Province**.
4. In **Data type**, select **Text**.
5. Click to select **Allow multiple values**.
6. In the left pane, click **Available Values**.
7. In **Select from one of the following options**, click **Get values from a query**.
8. In **Dataset**, select **StateProvinceLookup** from the drop-down list.
9. In **Value field**, select **StateProvinceCode**.
10. In **Label field**, select **StateProvinceName**.
11. Click **OK**.
12. In the **Parameters** pane, click and drag the **State or Province** parameter so that it is positioned below the **Country** parameter.
13. Click **Preview** to view the report.
14. Click **Design**.

Amend the report dataset

1. Right-click **Dataset1** and click **Dataset Properties**.
2. Amend the query by adding the following **WHERE** statement above the **GROUP BY** statement:

```
WHERE (G.CountryRegionCode = @CountryRegion) AND
      (G.StateProvinceCode IN (@StateProvince))
```

3. In the left pane, click **Parameters**.

4. Next to **Parameter Name @CountryRegion** select the Parameter Value **[@CountryRegion]** from the drop-down box.
5. Next to **@StateProvince**, select **[@StateProvince]**.
6. Click **OK**.

Create date range parameters

1. In the **Report Data** pane, right-click **Parameters**, and select **Add Parameter**.
2. In the **Report Parameter Properties** dialog box, in **Name**, type **DateFrom**.
3. In **Prompt**, type **Date from**.
4. In **Data type**, select **Date/time**.
5. In **Select parameter visibility**, select **Visible**.
6. In the left pane, click **Default Values**.
7. Click **Specify values** and then click **Add**.
8. In **Value**, type **2010-01-01**.
9. Click **OK** to create the parameter.
10. Right-click **Parameters**, and select **Add Parameter**.
11. In the **Report Parameter Properties** dialog box, in **Name**, type **DateTo**.
12. In **Prompt**, type **Date to**.
13. In **Data type**, select **Date/time**.
14. In **Select parameter visibility**, select **Visible**.
15. In the left pane, click **Default Values**.
16. Click **Specify values** and then click **Add**.
17. In **Value**, type **2020-12-31**.
18. Click **OK** to create the parameter.
19. Click and drag the **Date from** parameter up one row.
20. Click and drag the **Date to** parameter left one column.
21. Click **Preview** to view the report, and then change some of the parameters and click View Report. Note that the changes you make are not reflected in the report.
22. Click **Design**.

Amend the report dataset

1. Right-click **Dataset1** and click **Dataset Properties**.
2. Amend the **WHERE** statement so that it reads:

```
WHERE (G.CountryRegionCode = @CountryRegion) AND
      (G.StateProvinceCode IN (@StateProvince)) AND
      (S.OrderDate BETWEEN (@DateFrom) AND (@DateTo))
```

3. In the left pane, click **Parameters**.
4. In **Parameter Value**, select the correct **Parameter Value** for each Parameter Name.

5. Click **OK**.

Link the Country parameter to the StateProvince dataset

1. In the **Report Data** pane, right-click the **StateProvinceLookup** dataset and then click **Dataset Properties**.
2. In **Query**, add the following **WHERE** clause before the **GROUP BY** clause.

```
WHERE CountryRegionCode = @CountryRegion
```

3. On the left, click **Parameters**, change the **Parameter Value** to be **[@CountyRegion]**.
4. Click **OK**.
5. Click **Preview**. Test the **StateProvince** parameter by selecting different countries.

Test the parameters

1. Test the parameters by selecting different countries. The State or Province selection will change depending on the country selection.
2. Test different time periods, Select a **Date To** date before 2013 to see the total value decrease.
3. Compare your report with the report in **D:\Labfiles\Lab04\Solution\SalesSolution**.
4. Close all open windows, without saving any changes.

Results: After completing this lab exercise, you will be able to:

Add different types of parameters to a report.

Use a dataset to provide values for a parameter.

Use a parameter to change the available values for a parameter.

Exercise 2: Using parameters in Report Builder

► Task 1: Prepare the lab environment

Prepare the lab environment

1. Ensure the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab04** folder, right-click **Setup.cmd** and click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**, and then wait for the script to finish.

► Task 2: Add parameters to a report

Open the sales report

1. Click the **Start** menu, type **Report Builder** and then click to open it.
2. On the **File** menu, point to **Open**, and then navigate to **D:\Labfiles\Lab04\Starter\SalesReport**.
3. Click **SalesReport.rdl** and then click **Open**. The report is opened in design mode.

Add two datasets to use as lookups

1. In the **Report Data** pane, right-click **Datasets** and click **Add Datasets**.
2. In **Name**, type **CountryLookup**.
3. Click **Use a dataset embedded in my report**.
4. Under **Query**, click **Import**.
5. Navigate to **D:\Labfiles\Lab04\Starter**, click **CountryRegionLookup.sql** and then click **Open**.
6. In **Data source**, select **AdventureWorksDW** from the drop-down list.
7. Click **OK**.
8. In the **Report Data** pane, right-click **Datasets** and click **Add Datasets**.
9. In **Name**, type **StateProvinceLookup**.
10. Click **Use a dataset embedded in my report**.
11. Under **Query**, click **Import**.
12. Navigate to **D:\Labfiles\Lab04\Starter**, click **StateProvinceLookup.sql** and then click **Open**.
13. In **Data source**, select **AdventureWorksDW** from the drop-down list.
14. Click **OK**.

Create the Country parameter

1. In the **Report Data** pane, right-click **Parameters**, and click **Add Parameter**.
2. The **Report Parameter Properties** dialog box, in **Name**, type **CountryRegion**.
3. In **Prompt**, type **Country**.
4. In **Data type**, select **Text**.
5. In **Select parameter visibility**, select **Visible**.
6. In the left pane, click **Available Values**.
7. In **Select from one of the following options**, click **Get values from a query**.
8. In **Dataset**, select **CountryLookup** from the drop-down list.
9. In **Value field**, select **CountryRegionCode**.
10. In **Label field**, select **EnglishCountryRegionName**.
11. Click **OK**.
12. Click **Run** to view the report.
13. In the **Country** list, click any country, and then click **View Report**. The parameter will not change the contents of the report.
14. Click **Design**.

Create the StateProvince parameter

1. In the **Report Data** pane, right-click **Parameters**, and select **Add Parameter**.
2. In the **Report Parameter Properties** dialog box, in **Name**, type **StateProvince**.
3. In **Prompt**, type **State or Province**.
4. In **Data type**, select **Text**.

5. Click to select **Allow multiple values**.
6. In the left pane, click **Available Values**.
7. In **Select from one of the following options**, click **Get values from a query**.
8. In **Dataset**, select **StateProvinceLookup** from the drop-down list.
9. In **Value field**, select **StateProvinceCode**.
10. In **Label field**, select **StateProvinceName**.
11. Click **OK**.
12. If the **Parameters** pane isn't visible, on the toolbar click **View**, then check **Parameters**.
13. In the **Parameters** pane, click and drag the **State or Province** parameter so that it is positioned below the **Country** parameter.
14. Click **Run** to view the report.
15. Click **Design**.

Amend the report dataset

1. Right-click **Dataset1** and click **Dataset Properties**.
2. Amend the query by adding the following **WHERE** statement above the **GROUP BY** statement:

```
WHERE (G.CountryRegionCode = @CountryRegion) AND
      (G.StateProvinceCode IN (@StateProvince))
```

3. In the left pane, click **Parameters**.
4. Next to **Parameter Name @CountryRegion** select the Parameter Value **[@CountryRegion]** from the drop-down box.
5. Next to **@StateProvince**, select **[@StateProvince]**.
6. Click **OK**.

Create date range parameters

1. In the **Report Data** pane, right-click **Parameters**, and select **Add Parameter**.
2. In the **Report Parameter Properties** dialog box, in **Name**, type **DateFrom**.
3. In **Prompt**, type **Date from**.
4. In **Data type**, select **Date/time**.
5. In **Select parameter visibility**, select **Visible**.
6. In the left pane, click **Default Values**.
7. Click **Specify values** and then click **Add**.
8. In **Value**, type **2010-01-01**.
9. Click **OK** to create the parameter.
10. Right-click **Parameters**, and select **Add Parameter**.
11. In the **Report Parameter Properties** dialog box, in **Name**, type **DateTo**.
12. In **Prompt**, type **Date to**.
13. In **Data type**, select **Date/time**.

14. In **Select parameter visibility**, select **Visible**.
15. In the left pane, click **Default Values**.
16. Click **Specify values** and then click **Add**.
17. In **Value**, type **2020-12-31**.
18. Click **OK** to create the parameter.
19. Click and drag the **Date from** parameter up one row.
20. Click and drag the **Date to** parameter left one column.
21. Click **Run** to view the report, and then click **Design**.

Amend the report dataset

1. Right-click **Dataset1** and click **Dataset Properties**.
2. Amend the **WHERE** statement so that it reads:

```
WHERE (G.CountryRegionCode = @CountryRegion) AND
(G.StateProvinceCode IN (@StateProvince)) AND
(S.OrderDate BETWEEN (@DateFrom) AND (@DateTo))
```

3. In the left pane, click **Parameters**.
4. In **Parameter Value**, select the correct Parameter Value for each Parameter Name.
5. Click **OK**.

Link the Country parameter to the StateProvince dataset

1. In the **Report Data** pane, right-click the **StateProvinceLookup** dataset and then click **Dataset Properties**.
2. In **Query**, add the following **WHERE** clause before the **GROUP BY** clause.

```
WHERE CountryRegionCode = @CountryRegion
```

3. Click **OK**.
4. Click **Run**. Test the **StateProvince** parameter by selecting different countries.

Test the parameters

1. Test the parameters by selecting different countries. The State or Province selection will change depending on the country selection.
2. Test different time periods, Select a **Date To** date before 2013 to see the total value decrease.
3. Compare your report with the report in **D:\Labfiles\Lab04\Solution\SalesSolution**.
4. Close all open windows, without saving any changes.

Results: After completing this lab exercise, you will be able to:

Add different types of parameters to a report.

Use a dataset to provide values for a parameter.

Use a parameter to change the available values for a parameter.

MCT USE ONLY. STUDENT USE PROHIBITED

Module 5: Visualizing Data with Reporting Services

Lab: Manage formatting

Exercise 1: Report Designer

► Task 1: Prepare the environment

1. Ensure the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab05** folder, right-click **Setup.cmd** and click **Run as administrator**.
3. Click **Yes** when prompted to confirm you want to run the command file, and wait for the script to finish.

► Task 2: Formatting a report

1. From the taskbar, click **Visual Studio 2015** to open it.
2. Click **File, Open** and then **Project/Solution**.
3. Navigate to **D:\Labfiles\Lab05\Starter**. Double click the **Visuals** folder, Click **Visuals.sln** and then click **Open**.
4. In Solution Explorer, double-click **Visuals.rdl**. The report is opened in design mode.
5. Right-click the **[Order Date]** cell and then click **Text Box Properties**.
6. Click **Number, Custom**, and then click **fx**.
7. In **Set expression for: Format**, overtype **dd-MMM-yy**. Click **OK**.
8. Click **Alignment** and in **Horizontal** select **Right**. Click **OK**.
9. Right-click the **Order Date** heading cell. On the **Properties** menu, under **Alignment** click drop-down at text align and the select the **Align Right** icon.
10. Widen the Order Date column by positioning the cursor in the grey header bar between order date and product. Click and **drag** to widen the column.
11. Right-click the **[SalesAmount]** cell and click **Text Box Properties**.
12. Click **Number**, and then select **Currency**.
13. In **Symbol**, select a symbol appropriate for your country. Click **OK**.
14. Right-click the group **[Sum(SalesAmount)]** cell, and click **Text Box Properties**.
15. Click **Number, Currency** and in **Symbol** select the correct **symbol** for your country.
16. In **decimal places**, click the down arrow twice to display no decimal places. Select **Use 1000 separator (,)**. Click **OK**.
17. Right-click the last **[Sum(SalesAmount)]** that is in bold, and click **Text Box Properties**.
18. Click **Number, Currency** and in **Symbol** select the correct **symbol** for your country.
19. In **decimal places**, click the down arrow twice to display no decimal places. Select **Use 1000 separator (,)**. Click **OK**.
20. Click within the report, and position the cursor over the first column divider. Widen the column to fit the heading. Repeat for all columns.
21. Click in the **Product Category** heading cell, and delete the word **Product**.

22. Click in the **Product Subcategory** heading call, and delete the word **Product**.
23. Click **Preview**.
24. Click the **Design** tab.
25. Click **File, Save All**.

► **Task 3: Add a chart to a report in Report Designer**

1. If not yet open, from the taskbar, click **Visual Studio 2015** to open it.
2. Click **File, Open** and then **Project/Solution**.
3. Navigate to **D:\Labfiles\Lab05\Starter**. Double click the **Visuals** folder, Click **Visuals.sln** and then click **Open**.
4. In Solution Explorer, double-click **Chart.rdl**. The report is opened in design mode.
5. Click the tablix data region so that the gray row and column headers appear, and click the gray box where the row and column headers intersect at the top left to select the data region. Then drag the multidirectional arrow handle to move the data region down to make room for a good sized chart.
6. From the **Toolbox**, click the **chart** icon and click and drag a large rectangular to fit the area you just created. The Select Chart Type dialog box is displayed.
7. In the **Select Chart Type** dialog box, select first **bar chart** style and click **OK**.
8. Click the chart to display **Chart Data**.
9. In Values, click + and then select **SalesAmount**.
10. In Category Groups, click + and then select **Product**.
11. In the Series Groups, click + and then select **StateProvinceName**.
12. Click the **Chart title** to select it, and click **Delete**.
13. Click **Preview** to view the chart.
14. Click **Design**.

► **Task 4: Add a map to a report using Report Designer**

1. If not yet open, from the taskbar, click **Visual Studio 2015** to open it.
2. Click **File, Open** and then **Project/Solution**.
3. Navigate to **D:\Labfiles\Lab05\Starter**. Double click the **Visuals** folder, Click **Visuals.sln** and then click **Open**.
4. In Solution Explorer, double-click **Map.rdl**. The report is opened in design mode.
5. Click the tablix data region so that the gray row and column headers appear, and click the gray box where the row and column headers intersect at the top left to select the data region. Then drag the multidirectional arrow handle to move the data region down to make rooms for a good sized map.
6. From the Toolbox, click the **Map** icon and then click and drag a large rectangle in the area above the tablix. The New Map Layer wizard is displayed.
7. On the Choose a source of spatial data page, select **Map Gallery** and **USA by State Exploded**, and then click **Next**.
8. On the Choose spatial data and map view options page, click **Next** without making any changes.
9. On the Choose map visualization page, click **Bubble Map**, and then click **Next**.

10. On the Choose the analytical dataset page, click **Dataset1**, and then click **Next**.
11. On the Specify the match fields for spatial and analytical data page, click the check box in the Match Fields column next to **STATENAME**, in the Select a field drop-down list, select **StateProvinceName**, and click **Next**.
12. On the Choose color theme and data visualization page, clear **Use bubble sizes to visualize data**, check the **Use polygon colors to visualize data**. In the Data field, select **[Sum(SalesAmount)]**, in the Color rule, select **White-Green**. Click **Display labels** and in Data field select **#STUSPS**.
13. Click **Finish**.
14. Double-click **Map Title** and then over-type **Sales by State**.
15. Right click **Title** text and then point to **Map** and select **Map Properties**. Click **visibility** and then **Hide**. Click **OK**.
16. Click **Preview** to view the completed report.
17. Click **Design**.

► **Task 5: Adding databars and sparklines to a report using Report Designer**

1. If not yet open, from the taskbar, click **Visual Studio 2015** to open it.
2. Click **File, Open** and then **Project/Solution**.
3. Navigate to **D:\Labfiles\Lab05\Starter**. Double click the **Sparklines** folder.
4. Click **Sparklines.sln** and then click **Open**.
5. In Solution Explorer, double-click **Sparklines.rdl**. The report is opened in design mode.
6. Right-click the **Sales Value** header cell, and click **Insert Column** and then **Right**.
7. In the new column heading type **Comparison by Category**.
8. Right-click in the new cell below the heading, click **Insert** and then **Data Bar**.
9. Keep the default type and click **OK**.
10. Double-click the data bar to display **Chart Data**.
11. Under Values click **+** and then click **Sales Value**.
12. Right-click the data bar and then click **Horizontal Axis Properties**.
13. In Axis range and interval, from the drop-down list select **table1_EnglishProductCategoryName** and click **OK**.
14. Right-click the data bar and click **Chart Properties**.
15. In General, under Color Palette, select **Earth tones** from the drop-down list. Click **OK**.
16. Right-click the **Comparison by Category** header cell and click **Insert Column**, then **Right**.
17. In the heading cell of the new column type **Sales by Year**.
18. Right-click in the new cell below the heading and click **Insert** and then **Sparkline**.
19. In Select Sparkline Type, click **Column** and then click **OK**.
20. Double-click the sparkline to display **Chart Data**.
21. In the Values section, click **+** and then **Sales Value**.
22. In Category Groups, click **+** and then **Year**.
23. Right-click the sparkline and select **Horizontal Axis Properties**.

24. Select **Align axes in** and then select **table1** from the drop-down box.
25. In **Interval**, click **fx** and in Set expression for: Interval, overtype **1**. Click **OK**.
26. In Interval type, select **Number** from the drop-down box and then click **OK**.
27. Click **Preview** to view the report.
28. Click **Design**.
29. Click **File, Exit** to close Visual Studio 2015.

Results: After completing this lab exercise, you will be able to:

Format a report.

Add a chart to a report.

Add a map to a report.

Add databars and sparklines to a report.

Exercise 2: Report Builder

► Task 1: Prepare the environment

1. Ensure the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab05** folder, right-click **Setup.cmd** and click **Run as administrator**.
3. Click **Yes** when prompted to confirm you want to run the command file, and wait for the script to finish.

► Task 2: Formatting a report using Report Builder

1. Click **Start**, type **Report Builder** and then click to open it.
2. Click **Open** and navigate to **D:\Labfiles\Lab05\Starter\Visuals**, and then click **VisualsBuilder.rdl** and **Open**. The report is opened in design mode.
3. Right-click the **Order Date** cell and then click **Text Box Properties**.
4. Click **Number, Custom**, and then click **fx**.
5. In **Set expression for: Format**, overtype **dd-MMM-yy**. Click **OK**.
6. Click **Alignment** and in **Horizontal** select **Right**. Click **OK**.
7. Right-click the **Order Date** heading cell, and in the menu click the **Align Right** icon.
8. Widen the **Order Date** column by positioning the cursor in the grey header bar between order date and product. Click and **drag** to widen the column.
9. Right-click the **SalesAmount** cell and click **Text Box Properties**.
10. Click **Number**, and then select **Currency**.
11. In Symbol, select a **symbol** appropriate for your country. Click **OK**.
12. Right-click the group **Sum(SalesAmount)**, and click **Text Box Properties**.
13. Click **Number, Currency** and in Symbol select the correct **symbol** for your country.

14. In decimal places, click the down arrow twice to display **no decimal places**. Select **Use 1000 separator (,)**. Click **OK**.
15. Right-click the last **Sum(SalesAmount)**, and click **Text Box Properties**.
16. Click **Number, Currency** and in Symbol select the correct **symbol** for your country.
17. In decimal places, click the down arrow twice to display **no decimal places**. Select **Use 1000 separator (,)**. Click **OK**.
18. Click within the report, and position the cursor over the first column divider. Widen the column to fit the heading. Repeat for all columns.
19. Click in the Product Category heading cell, and delete the word **Product**.
20. Click in the Product Subcategory heading call, and delete the word **Product**.
21. Click **Run** to view the changes.
22. Click **Design**.
23. Leave the report open for the next lab exercise.

► Task 3: Add a chart to a report using Report Builder

1. If Report Builder is not already open, click **Start**, type **Report Builder** and then **click** to open it.
2. Click **Open** and navigate to **D:\Labfiles\Lab05\Starter\Visuals**, click **ChartBuilder.rdl** and then click **Open**. The report is opened in design mode.
3. Click the tablix data region so that the gray row and column headers appear, and click the gray box where the row and column headers intersect at the top left to select the data region. Then drag the multidirectional arrow handle to move the data region down to make room for a good sized chart.
4. From the **Insert** menu, click **Chart** and then click **Chart Wizard**. The New Chart Wizard is displayed.
5. In Choose a dataset, select **DataSet1** and click **Next**.
6. In Choose a chart type, click **Bar** and click **Next**.
7. From Available Fields, drag:
 - a. **SalesAmount** to **Values**.
 - b. **Product** to **Categories**.
 - c. **StateProvinceName** to **Series**.
8. Click **Next**.
9. Check the preview, and then click **Finish**.
10. **Position** and **size** the chart to fit the area you created earlier.
11. Click the **Chart title** to select it, and click **Delete**.
12. Click **Run** to view the chart.
13. Click **Design**.

► Task 4: Adding a map to a report using Report Builder

1. If Report Builder is not already open, click **Start**, type **Report Builder** and then **click** to open it.
2. Click **Open** and navigate to **D:\Labfiles\Lab05\Starter\Visuals**, click **MapBuilder.rdl** and then click **Open**. The report is opened in design mode.

3. Click the tablix data region so that the gray row and column headers appear, and click the gray box where the row and column headers intersect at the top left to select the data region. Then drag the multidirectional arrow handle to move the data region down to make room for a good sized map.
4. From the **Insert** menu, click **Map** and then click **Map Wizard**. The New Map Wizard is displayed.
5. On the **Choose a source of spatial data** page, select **Map Gallery** and **USA by State Exploded**, and then click **Next**.
6. On the **Choose spatial data and map view options** page, click **Next** without making any changes.
7. On the **Choose map visualization** page, click **Bubble Map**, and then click **Next**.
8. On the **Choose the analytical dataset** page, click **Dataset1**, and then click **Next**.
9. On the **Specify the match fields for spatial and analytical data** page, click the check box in the **Match Fields** column next to **STATENAME**, in the **Select a field** drop-down list, select **StateProvinceName**, and click **Next**.
10. On the **Choose color theme and data visualization** page, clear **Use bubble sizes to visualize data**, check the **Use polygon colors to visualize data**. In the **Data** field, select **[Sum(SalesAmount)]**, in the **Color** rule, select **White-Green**. Click **Display labels** and in Data field select **#STUSPS**.
11. Click **Finish**.
12. Double-click **Map Title** and then over-type **Sales by State**.
13. Right-click the **Map Legend** and then click **Map Properties**. Click **Visibility** and then **Hide**. Click **OK**.
14. Click **Run** to view the completed report.
15. Click **Design**.

► Task 5: Adding data bars and sparklines to a report using Report Builder

1. If Report Builder is not already open, click **Start**, type **Report Builder** and then click to open it.
2. Click **Open** and navigate to **D:\Labfiles\Lab05\Starter\Sparklines**, click **SparklinesBuilder.rdl** and then click **Open**. The report is opened in design mode.
3. Right-click the **Sales Value** header cell, and click **Insert Column** and then **Right**.
4. In the new column heading type **Comparison by Category**.
5. Right-click in the new cell below the heading, click **Insert** and then **Data Bar**.
6. Keep the default type and click **OK**.
7. Double-click the data bar to display **Chart Data**.
8. Under Values click **+** and then click **Sales Value**.
9. Right-click the data bar and then click **Horizontal Axis Properties**.
10. In Axis range and interval, from the drop-down list select **table1_EnglishProductCategoryName** and click **OK**.
11. Right-click the data bar and point to chart and then click **Chart Properties**.
12. In General, under Color Palette, select **Earth tones** from the drop-down list. Click **OK**.
13. Right-click the **Comparison by Category** header cell and click **Insert Column**, then **Right**.
14. In the heading cell of the new column type **Sales by Year**.
15. Right-click in the new cell below the heading and click **Insert** and then **Sparkline**.
16. In Select Sparkline Type, click **Column** and then click **OK**.

17. Double-click the sparkline to display **Chart Data**.
18. In the Values section, click + and then **Sales Value**.
19. In Category Groups, click + and then **Year**.
20. Right-click the sparkline and select **Horizontal Axis Properties**.
21. Select **Align axes in** and then select **table1** from the drop-down box.
22. In Interval, click **fx** and in Set expression for: Interval, overtype **1**. Click **OK**.
23. In Interval type, select **Number** from the drop-down box and then click **OK**.
24. Click **Run** to view the report.
25. Click **File, Exit Report Builder** to close.
26. If **Microsoft SQL Server Report Builder** window displays to save changes click **No**.

Results: After completing this lab exercise, you will be able to:

Format a report.

Add a chart to a report.

Add a map to a report.

Add databars and sparklines to a report.

MCT USE ONLY. STUDENT USE PROHIBITED

Module 6: Summarizing Report Data

Lab: Summarizing report data

Exercise 1: Sorting and grouping in Report Builder

► Task 1: Prepare the lab environment

1. Ensure the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab06** folder, right-click **Setup.cmd** and click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**.
4. If prompted, press any key to continue, and then wait for the script to finish.

► Task 2: Create a matrix with groups

Add groups to a matrix

1. Click the **Start** menu, and type **Report Builder** and then click to open it.
2. Click **Open** and navigate to **D:\Labfiles\Lab06\Starter\Groups\Groups** and double click **Groups.rdl** to open. The report contains an empty matrix.
3. In the second row of the left column, click the **button** and select **EnglishProductSubcategoryName**.
4. **Widen** the column and change the column title to read **Subcategory**.
5. Highlight the column title **Subcategory** and in the ribbon menu, click **Bold**.
6. In the second row of the second column, click the **button** and click **SalesValue**.
7. Right-click **[Sum(SalesValue)]** and click **Text Box Properties**.
8. In the left pane, click **Number** and in **Category** click **Number** and then click twice to reduce to **0 decimal places**. Select **Use 1000 Separator** and click **OK**.
9. Highlight the column heading **Sales Value** and in the ribbon menu, click **Bold**.
10. Click the **[Sum(SalesValue)]** text box and note the orange scope indicators on the row and column.
11. Click **Run** to view the report.
12. Click **Design**.

Add a column group

1. In the **Column Groups** pane, double-click **ColumnGroup**.
2. In the **Group Properties** dialog box, in **Name**, highlight **ColumnGroup** and then overwrite **Year**.
3. Under **Group expressions**, in **Group on:** list, select **[Year]**.
4. In the left pane, click **Sorting**.
5. Under **Change sorting options**, click **Add**.
6. In **Sort by**, select **[Year]**. In **Order**, select **A to Z** and then click **OK**.
7. In the column heading, highlight **Sales Value** and type **[Year]** (ensure you type each character in turn including the brackets to denote a field).
8. Click **Run** to view the report and then click **Design**.

Add a row group

1. Click in the matrix, and then right-click the row selector grey tab to the left of **[EnglishProductSubCategoryName]**.
2. Click **Add Group**, and under **Row Group** click **Parent Group**.
3. In the **Tablix group** dialog box, in **Group By**, select **[EnglishProductCategoryName]** from the drop-down box.
4. Click **Add group footer**, and click **OK**.
5. **Widen** the **English Product Category** column and change the heading to **Category**.
6. In the empty row at the bottom of the **Subcategory** column, type **Total**.
7. In the ribbon menu, in the **Paragraph** group, select **right aligned**.
8. In the bottom right cell, click the **button** and then click **SalesValue**.
9. Right-click in **[Sum(SalesValue)]** and then click **Text Box Properties**.
10. In the left pane click **Number** and under **Category**, click **Number**, click twice to reduce to **0 decimal places** and click **Use 1000 Separator**. Click **OK**.
11. Click **Run** to view the report, and then click **Design**.

Add a totals column

1. Click within the matrix, and then right-click the grey column tab above the **[Year]** column.
2. From the context menu click **Insert Column**, then **Outside Group - Right**.
3. In the new column, middle row cell, click the **button** and then click **SalesValue**.
4. Amend the column heading to **Total**, and in the ribbon menu click the **right-align icon** to right align the heading and then click **Bold** to make it bold.
5. Right-click the bottom right cell and click **Text Box Properties**.
6. In the left pane click **Number**, and under **Category** click **Number**. Click twice to reduce to **0 decimal places**, and select **Use 1000 Separator**. Click **OK**.
7. Right-click the cell above, and click **Text Box Properties**. In the left pane click **Number**, and under **Category** click **Number**. Click twice to reduce to **0 decimal places**, and select **Use 1000 Separator**. Click **OK**.
8. Click **Run** to view the report, and then click **Design**.

Add a totals row

1. Select the cell in the **Category** column.
2. Right-click the bottom grey row selector, and click **Insert Row**, and then **Outside Group - Below**.
3. In the bottom cell of the **Category** column, type **Grand Total**. From the ribbon menu, make it **bold** and **right aligned**.
4. In the bottom cell of the **[Year]** column, click the **button** and select **SalesValue**.
5. In the bottom cell of the **Total** column, click the **button** and select **SalesValue**.
6. Right-click the bottom total in the **[Year]** column, and select **Text Box Properties**.
7. In the left pane, click **Number**, and under **Category** click **Number**. Click twice to reduce to **0 decimal places**, and select **Use 1000 Separator**. Click **OK**.

8. From the ribbon menu, select **bold** and **right-aligned**.
9. Right-click the bottom total in the **Total** column, and select **Text Box Properties**.
10. In the left pane, click **Number**, and under Category click **Number**. Click twice to reduce to **0 decimal places** and select **Use 1000 Separator**. Click **OK**.
11. From the ribbon menu, select **bold** and **right-aligned**.
12. Click **Run** to preview the report, and then click **Design**.
13. Leave **Report Builder** open for the next lab exercise.

► **Task 3: Add drilldown to a report**

1. On the **File** menu, click **Open** and then navigate to **D:\Labfiles\Lab06\Starter\Drilldown\Drilldown**. If a dialog box appears to save the file, click **No**.
2. Double-click **Drilldown.rdl** to open it.
3. Click **Run** to view the report.

Add drilldown actions

1. Click **Design**.
2. In the **Column Groups** pane, click the drop-down arrow, then click **Advanced Mode**. If necessary, make the bottom window larger.
3. In **Row Groups**, select the **EnglishProductSubcategoryName**.
4. Above the ribbon, click the **View** tab, then select **Properties**.
5. In the **Properties** blade, in the **Visibility** section, change **Hidden** to **True**.
6. In **ToggleItem**, select **EnglishProduct CategoryName**.
7. Above the ribbon click the **Home** tab, and then click **Run** to view the report.
8. Click **Design**, then in the **Row Groups** select **(Details)**.
9. In the **Properties** blade, under the **Visibility** section, change **Hidden** to **True**.
10. In **ToggleItem**, select **EnglishProductSubcategoryName**.
11. Click **Run** to see your changes.
12. Click **File, Exit Report Builder** without saving changes.

Results: After completing this lab exercise, you will be able to:

Create groups within a report.

Set the sort order.

Create drilldown actions within a report.

Exercise 2: Sorting and grouping in Report Designer

► Task 1: Prepare the lab environment

1. Ensure the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab06** folder, right-click **Setup.cmd** and click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**,
4. If prompted, press any key to continue and then wait for the script to finish.

► Task 2: Create a matrix with groups

Add groups to a matrix

1. Click **Visual Studio 2015** on the toolbar to open it.
2. Click **File** and then click **Open**, then **Project/Solution** and navigate to **D:\Labfiles\Lab06\Starter\Groups**, click **Groups.sln**, and then click **Open**. If the **Security Warning for Groups** dialog box appears, clear the **Ask me for every project in this solution** check box, and then click **OK**.
3. In Solution Explorer, double-click **Groups.rdl** to open it in design mode. The report contains an empty matrix.
4. In the second row of the left column, click the button and select **EnglishProductSubcategoryName**.
5. Widen the column width and change the column title to read **Subcategory**.
6. Highlight the column title **Subcategory** and in the properties menu scroll and expand **Font**, and then at **Fontweight** click dropdown and select **Bold**.
7. In the second row of the second column, click the button and click **SalesValue**.
8. Right-click **[Sum(Sales Value)]** and select **Text Box Properties**.
9. Click **Number** and then **Number** and click twice to reduce to **0 decimal places** and click **Use 1000 Separator**.
10. Click **OK**.
11. Highlight the column heading **Sales Value** and apply **Bold** formatting from the **Properties** menu.
12. Click in **[Sum(Sales Value)]** and note the orange scope indicators on the row and column.
13. Click **Preview** to view the report and then click **Design**.

Add a column group

1. In the **Column Groups** pane, double-click **ColumnGroup**.
2. In the **Group Properties** dialog box, in **Name**, type **Year**.
3. Under **Group expressions**, in **Group on:** list select **[Year]**.
4. In the left pane, click **Sorting**.
5. Under **Change sorting options**, click **Add**.
6. In **Sort by**, select **[Year]** from the drop-down box. In **Order**, select **A to Z** and then click **OK**.
7. In the column heading cell for the **Sales Value** column, type **[Year]** (ensure you type each character in turn including the brackets to denote a field).

8. Click **Preview** to view the report and then click **Design**.

Add a row group

1. Click in the matrix, and then right-click the row selector grey tab to the left of **[EnglishProductSubcategoryName]**.
2. Click **Add Group**, and under **Row Group** click **Parent Group**.
3. In the **Tablix group** dialog box, in **Group by**, select **[EnglishProductCategoryName]** from the drop-down box.
4. Click **Add group footer**, and click **OK**.
5. Widen the **English Product Category** column and change the heading to **Category**.
6. In the bottom row of the **Subcategory** column, type **Total**.
7. In Properties, under **Indent** at **TextAlign** select the drop-down and click **Right**.
8. In the bottom right cell, in the **[Year]** column, click the **button** and then click **SalesValue**.
9. Right-click in **[Sum(SalesValue)]** and then click **Text Box Properties**.
10. In the left pane click **Number** and under Category, click **Number**, click twice to reduce to **0 decimal places** and click **Use 1000 Separator**. Click **OK**.
11. Click **Preview** to view the report, and then click **Design**.

Add a totals column

1. Click within the matrix, and then right-click the grey column tab above the **[Year]** column.
2. From the context menu click **Insert Column**, then **Outside Group - Right**.
3. In the new column, middle row cell, click the **button** and then click **SalesValue**.
4. Amend the column heading to **Total**, and in Properties, under **Indent** at **TextAlign** field click drop-down and then select **right** to right align the heading and then click **Bold** to make it bold if necessary.
5. Right-click the bottom right cell and click **Text Box Properties**.
6. In the left pane click **Number**, and under Category click **Number**. Click twice to reduce to **0 decimal places**, and select **Use 1000 Separator**. Click **OK**.
7. Click **Preview** to view the report, and then click **Design**.

Add a totals row

1. Select the cell in the **Category** column.
2. Right-click the bottom grey row selector, and click **Insert Row**, and then **Outside Group - Below**.
3. In the bottom cell of the **Category** column, type **Grand Total**. From Properties, make it **bold** and **right aligned**.
4. In the bottom cell of the **Year** column, click the **button** and select **SalesValue**.
5. In the bottom cell of the **Total** column, click the **button** and select **SalesValue**.
6. Right-click the bottom cell in the **[Year]** column, and select **Text Box Properties**.
7. In the left pane, click **Number**, and under Category click **Number**. Click twice to reduce to **0 decimal places**, and select **Use 1000 Separator**. Click **OK**.
8. In **Properties**, select **bold** and **right-aligned**.

9. Right-click the bottom cell in the **Total** column, and select **Text Box Properties**.
10. In the left pane, click **Number**, and under Category click **Number**. Click twice to reduce to **0 decimal places** and select **Use 1000 Separator**. Click **OK**.
11. In **Properties**, select **bold** and **right-aligned**.
12. Click **Preview** to preview the report, and then click **Design**.
13. Click **File, Save All**.
14. Click **File**, close **Solution**.
15. Leave **Visual Studio 2015** open for the next lab exercise.

► **Task 3: Add drilldown to a report**

1. Click **File**, click **Open**, click **Project/Solution** and then navigate to **D:\Labfiles\Lab06\Starter\Drilldown**.
2. Highlight **Drilldown.sln** and click **Open**. If a Security Dialog warning box appears, click **OK**.
3. Double-click **Drilldown.rdl** to open it.
4. Click **Preview** to view the report.

Add drilldown actions

1. Click **Design**.
2. Click the **Down arrow** on the far right of the Row Groups and Column Groups pane, then click **Advanced Mode**. If necessary, make the bottom window larger.
3. In Row Groups, select the **[EnglishProductSubcategoryName]**.
4. In the **Properties** blade, find the **Visibility** section and in **Hidden**, select **True**.
5. In **ToggleItem**, select **EnglishProduct CategoryName**.
6. Click **Preview** to view the report.
7. Click **Design**, then in the Row Groups pane select **≡ (Details)**.
8. In **Properties**, under the Visibility section, change **Hidden** to **True**.
9. In **ToggleItem**, select **EnglishProductSubcategoryName**.
10. Click **Preview** to see your changes.
11. Click **File, Exit** without saving changes.

Results: After completing this lab exercise, you will be able to:

Create groups within a report.

Set the sort order.

Create drilldown actions within a report.

Module 7: Sharing Reporting Services Reports

Lab: Sharing Reporting Services reports

Exercise 1: Create a shared schedule

► Task 1: Prepare the lab environment

1. Ensure the **10990C-MIA-CLI**, **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab07\Starter** folder, right-click **Setup.cmd** and click **Run as administrator**. Click **Yes** when prompted to confirm that you want to run the command file.
3. If any Security warnings appear, type **r**, press Enter, and wait for the script to finish.

► Task 2: Create a shared schedule

1. Using Internet Explorer, navigate to **http://mia-sql/reports_sql2**, then click the settings icon (the gear icon in the upper right of the page), then click **Site settings**.
2. On the **Site settings** page, click **Schedules** then click **+New schedule**.
3. In the **Schedule name** box, type **Every 2 minutes**, then select **Hour**.
4. In the **hours** box type **0**, then in the **minutes** box type **02**, then click **Apply**.

Results: At the end of this exercise, you should have created a shared schedule.

Exercise 2: Configure caching

► Task 1: Configure execution account

1. In Internet Explorer, click **SQL Server Reporting Services** to return to the home page.
2. On the portal home page click **AdventureWorks Sample Reports**, under **DATA SOURCES**, click **AdventureWorks**.
3. On the **Properties** page, select **Using the following credentials**.
4. In the **Type of credentials** box, select **Database user name and password**.
5. In the **User name** box, type **SSRS**, then in the **Password** box type **Pa55w.rd**, then click **Apply**.

► Task 2: Configure caching

1. In Internet Explorer, click **SQL Server Reporting Services** to return to the home page.
2. On the portal home page click **AdventureWorks Sample Reports**, then right-click **Sales_Order_Detail** and then click **Manage**.
3. On the **Manage Sales_Order_Detail** page, click **Caching**.
4. Select **Always run this report against pregenerated snapshots**, then select **Create cache snapshots on a schedule**.
5. Select **Shared schedule**, then in the shared schedule box, select **Every 2 minutes**.

6. Select **Create a cache snapshot when I click Apply on this page**, then click **Apply**.

Results: At the end of this exercise, you should have configured report caching.

Exercise 3: Subscribe to a report

► Task 1: Configure the file share

1. In File Explorer, navigate to **D:\Labfiles\Lab07\Starter**. Right-click **Share**, then click **Properties**.
2. In the **Share Properties** window, on the **Sharing** tab, click **Share**.
3. In the lower box, click **Everyone** then click **Read/Write**, click **Share**, click **Done**, then click **Close**.

► Task 2: Create the subscription

1. In Internet Explorer, click **SQL Server Reporting Services** to return to the home page.
2. On the portal home page, click **AdventureWorks Sample Reports**, then right-click **Sales_Order_Detail** and then click **Subscribe**.
3. On the **New Subscription** page, in the **Description** box type **Write to Excel**.
4. Confirm that **Standard subscription** is selected.
5. Under **Schedule**, select **When the report data is updated (by the cache snapshot schedule)**.
6. Under **Destination**, select **Windows File Share**.
7. Under **Delivery Options**, in the **Path** box type **\\mia-sql\share**.
8. In the **Render format** box, select **Excel**.
9. In the **Credentials used to access the file share** section, use the following values:
 - **User Name:** adventureworks\ExecutionAccount
 - **Password:** Pa55w.rd
10. Select **Increment file names as newer versions are added**, then click **Create subscription**.

► Task 3: Verify the configuration

1. Log on to **10990C-MIA-CLI** as **Student** with the password **Pa55w.rd**.
2. Start File Explorer, then in the navigation bar type **\\mia-sql\share**.
3. In the **Windows Security** dialog box, use the following values:
 - **User Name:** adventureworks\student
 - **Password:** Pa55w.rd
4. Verify that a copy of the report is being written to the share every two minutes. When you have finished, close File Explorer and disconnect from **10990C-MIA-CLI**.
5. On **10990C-MIA-SQL**, close all open windows without saving any changes.

Results: At the end of this lab, you should have created a subscription that writes copies of a report to a file share.

Module 8: Administering Reporting Services

Lab: Administering Reporting Services

Exercise 1: Authorize access to reports

► Task 1: Prepare the lab environment

1. Ensure that the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are both running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab08\Starter** folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes** to run the command file, and then wait for the script to finish.

► Task 2: Create reporting folders

1. Start Internet Explorer.
2. In the Address bar, type **http://mia-sql/reports_sql2** and then press Enter.
3. In the top right of the **SQL Server Reporting Services** page, click **+** and then click **Folder**.
4. In the **Create a new folder in Home** dialog box, in the **Name** text box, type **Sales Reports** and then click **Create**.
5. In the top right of the **SQL Server Reporting Services** page, click **+** and then click **Folder**.
6. In the **Create a new folder in Home** dialog box, in the **Name** text box, type **Directors Reports** and then click **Create**.

► Task 3: Create Active Directory groups and users

1. Switch to the **10990C-MIA-DC** virtual machine and log on as **ADVENTUREWORKS\Administrator** with the password **Pa55w.rd**.
2. Click **Start**, and then click **Server Manager**.
3. Click **Tools** from the top right menu, then click **Active Directory Users and Computers**.
4. In the hierarchy on the left, click the **Users** container.
5. On the **Action** menu, click **New** and then click **User**.
6. In the **New Object – User** dialog box, in the **First Name** text box, type **Sophia**.
7. In the **Last Name** text box, type **Garner**.
8. In the **User logon name** text box, type **Sophia** and then click **Next**.
9. In the **Password** text box, type **Pa55w.rd**.
10. In the **Confirm Password** text box, type **Pa55w.rd**.
11. Clear the **User must change password at next login** check box.
12. Select the **Password never expires** check box.
13. Click **Next** and then click **Finish**.
14. On the **Action** menu, click **New** and then click **User**.

15. In the **New Object – User** dialog box, in the **First Name** text box, type **Rodrigo**.
16. In the **Last Name** text box, type **Romani**.
17. In the **User logon name** text box, type **Rodrigo** and then click **Next**.
18. In the **Password** text box, type **Pa55w.rd**.
19. In the **Confirm Password** text box, type **Pa55w.rd**.
20. Clear the **User must change password at next login** check box.
21. Select the **Password never expires** check box.
22. Click **Next** and then click **Finish**.
23. On the **Action** menu, click **New** and then click **Group**.
24. In the **New Object – Group** dialog box, in the **Group name** text box, type **Directors** and then click **OK**.
25. Right-click **Sophia Garner** and then click **Add to a group**.
26. In the **Select Groups** dialog box, type **Directors** and then click **Check Names**.
27. When **Directors** is underlined, click **OK** and then click **OK**.
28. In the hierarchy on the left, click the **Users** container.
29. On the **Action** menu, click **New** and then click **Group**.
30. In the **New Object – Group** dialog box, in the **Group name** text box, type **Sales** and then click **OK**.
31. Right-click **Rodrigo Romani** and then click **Add to a group**.
32. In the **Select Groups** dialog box, type **Sales** and then click **Check Names**.
33. When **Sales** is underlined, click **OK** and then click **OK**.

► **Task 4: Assign permissions to groups**

1. Switch to the **10990C-MIA-SQL** virtual machine.
2. In Internet Explorer, browse to http://mia-sql/reports_sql2.
3. In the **Home** page, click **Directors Reports**.
4. In the top-right of the page, click the **Manage** button.
5. On the **Manage Directors Reports** page, on the left click **Security**.
6. Click **Customize security**.
7. In the **Confirm** dialog box, click **OK**.
8. Click **Add group or user**.
9. In the **Group or user** text box, type **Directors**.
10. Select the **Browser** check box.
11. Select the **Publisher** check box and then click **OK**.
12. In the top-left, click the **Home** link.
13. Click **Sales Reports**.
14. In the top-right of the page, click the **Manage** button.

15. On the **Manage Sales Reports** page, on the left click **Security**.

16. Click **Customize security**.

17. In the **Confirm** dialog box, click **OK**.

18. Click **Add group or user**.

19. In the **Group or user** text box, type **Sales**.

20. Select the **Browser** check box.

21. Select the **Publisher** check box and then click **OK**.

22. Click **Add group or user**.

23. In the **Group or user** text box, type **Directors**.

24. Select the **Browser** check box and then click **OK**.

25. In the top-left, click the **Home** link.

26. In the top-right of the page, click the **Manage** button.

27. Click **Add group or user**.

28. In the **Group or user** text box, type **Directors**.

29. Select the **Browser** check box and then click **OK**.

30. Click **Add group or user**.

31. In the **Group or user** text box, type **Sales**.

32. Select the **Browser** check box and then click **OK**.

33. Close Internet Explorer and sign out of **10990C-MIA-SQL**.

► Task 5: Test permissions

1. Log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Sophia** with the password **Pa55w.rd**.

2. Start Internet Explorer and browse to **http://mia-sql/reports_sql2**.

3. If the **Set up Internet Explorer 11** dialog appears, select **Use recommended security, privacy, and compatibility settings** and then click **OK**.

4. In the Report Services portal, click **Directors Reports**.

5. In the top-right, click **+** and then click **Folder**.

6. In the **Name** text box, type **Test Folder** and then click **Create**.

7. In the top-left, click the **Home** link and then click **Sales Reports**. Notice that there is no **+** button.

8. Close Internet Explorer and log out of **10990C-MIA-SQL**.

9. Log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Rodrigo** with the password **Pa55w.rd**.

10. Start Internet Explorer and browse to **http://mia-sql/reports_sql2**.

11. If the **Set up Internet Explorer 11** dialog appears, select **Use recommended security, privacy, and compatibility settings** and then click **OK**.

12. Notice that the **Directors Reports** folder is not displayed.

13. In the Report Services portal, click **Sales Reports**.

14. In the top-right, click **+** and then click **Folder**.

15. In the **Name** text box, type **Test Folder** and then click **Create**.
16. Close Internet Explorer and log out of **10990C-MIA-SQL**.

Results: At the end of this exercise, you will have created and authorized access to a set of folders within the Reporting Services web portal.

Exercise 2: Web portal branding

► Task 1: Complete the brand package

1. Log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab08\Starter\BrandPackage** folder, right-click **metadata.xml**, click **Open with**, and then click **Microsoft Visual Studio 2015**.
3. Place the cursor at the end of the following line of code, and then press Enter:

```
version="2.0.2"
```

4. Type the following code:

```
name="Adventure Works"
```

5. Place the cursor between the <Contents> tags.
6. Type the following code and then press Enter:

```
<Item key="colors" path="colors.json" />
```

7. Type the following code and then press Enter:

```
<Item key="logo" path="AW-Logo.jpg" />
```

8. On the **File** menu, click **Save All** and then close Visual Studio 2015.
9. Use Windows Explorer to navigate to the **D:\Labfiles\Lab08\Starter\BrandPackage** folder.
10. On the ribbon, click **Select all**.
11. Right-click the selected files, click **Send to** and then click **Compressed (zipped) folder**.
12. Type **AdworksBrand** and then press Enter.

► Task 2: Upload and use the brand package

1. Start Internet Explorer, and browse to **http://mia-sql/reports_sql2**.
2. In the top-right, click the cog icon and then click **Site settings**.
3. In the **Name** text box, type **Adventure Works Reporting** and then click **Apply**.
4. On the left of the page, click **Branding** and then click **Upload brand package**.
5. Browse to the **D:\Labfiles\Lab08\Starter\BrandPackage** folder.
6. Click **AdworksBrand.zip** and then click **Open**. The portal applies the new colors and logo.
7. Click **Browse** to view the home page colors.

8. Close all open windows.

Results: At the end of this exercise, you will have modified the colors and logo for the Reporting Services web portal.

MCT USE ONLY. STUDENT USE PROHIBITED

Module 9: Extending and Integrating Reporting Services

Lab: Extending and integrating Reporting Services

Exercise 1: Custom code—Report Designer

► Task 1: Prepare the lab environment

1. Ensure the **10990C-MIA-CLI**, **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab09\Starter** folder, right-click **Setup.cmd** and click **Run as administrator**.
3. Click **Yes** when prompted to confirm you want to run the command file, and wait for the script to finish.

► Task 2: Review the requirements

- Review the requirements in the exercise scenario.

► Task 3: Format last order amount

1. Start Visual Studio 2015, then on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, go to the **D:\Labfiles\Lab09\Starter\Lab** folder, click **Lab.sln**, and then click **Open**.
3. In the **Security Warning for Lab** dialog box, clear the **Ask me for every project in this solution** check box, and then click **OK**.
4. In Solution Explorer, expand **Reports** then double-click **OrdersReport.rdl**.
5. Click **Preview** to preview the report, then click **Design** to return to design mode.
6. In the second pane of the Design tab, right-click **[LastOrderAmount]** then click **Expression**.
7. In the **Expression** dialog box, in the **Set expression for: Value** box, move the cursor after the equals sign (=).
8. In the **Category** box, expand **Common Functions** then click **Conversion**.
9. In the **Item** box, double click **Int**.
10. In the **Set expression for: Value** box, move the cursor to the end of the line, type **)**, and then click **OK**.
11. Click **Preview** to preview the report, then click **Design** to return to design mode.

► Task 4: Format last order date

1. In Visual Studio 2015, in the second pane of the Design tab, right-click anywhere outside the boundary of the report body and click **Report Properties**.
2. In the **Report Properties** window, on the **Code** page, in the **Custom code** box, type the following, and then click **OK**:

```
Public Function yyyyymmdd_to_short_date (ByVal s As String) As String
    Dim myDate As Date = Date.ParseExact(s, "yyyyMMdd",
    System.Globalization.DateTimeFormatInfo.InvariantInfo)
    return myDate.ToString("dd MMM yyyy")
End Function
```

You can alternatively copy and paste the function text from
D:\Labfiles\Lab09\Starter\Code\date_function.txt.

3. In the second pane of the Design tab, right-click **[LastOrderDate]** then click **Expression**.
4. In the **Expression** dialog box, in the **Set expression for: Value** box, edit the expression so that it reads, and then click **OK**:

```
=Code.yyyyymmdd_to_short_date(Fields!LastOrderDate.Value)
```

5. Click **Preview** to preview the report, then click **Design** to return to design mode.

► Task 5: Format customer name

1. In Visual Studio 2015, in the second pane of the Design tab, right-click anywhere outside the boundary of the report body and click **Report Properties**.
2. In the **Report Properties** window, on the **Code** page, in the **Custom code** box move the cursor to the end of the existing code, and then press Enter.
3. Type the following, and then click **OK**:

```
Public Function SplitName (ByVal name As String) As String()
    Dim stringSeparators() As String = {" ",""}
    return name.Split(stringSeparators,StringSplitOptions.None)
End Function
```

You can alternatively copy and paste the function text from
D:\Labfiles\Lab09\Starter\Code\name_function.txt.

4. In the second pane of the Design tab, under **Last Name**, right-click **[CustomerName]** then click **Expression**.
5. In the **Expression** dialog box, in the **Set expression for: Value** box, edit the expression so that it reads, and then click **OK**:

```
=Code.SplitName(Fields!CustomerName.Value)(0)
```

6. In the second pane of the Design tab, under **First Name**, right-click **[CustomerName]** then click **Expression**.
7. In the **Expression** dialog box, in the **Set expression for: Value** box, edit the expression so that it reads, and then click **OK**:

```
=Code.SplitName(Fields!CustomerName.Value)(1)
```

8. Click **Preview** to preview the report, then click **Design** to return to design mode.

► Task 6: Alternate row colors

1. In Visual Studio 2015, in the second pane of the Design tab, in the report, click any cell in the table, then click the bottom-left gray cell to select the last row of the table.
2. In the **Properties** pane, scroll down to the **Fill** section, click the value of the **BackgroundColor** property, click the drop-down arrow, and then click **Expression**.
3. In the **Expression** window, in the **Set expression for: BackgroundColor** box, type the following, and then click **OK**:

```
=IIf(LineNumber(Nothing) Mod 2, "LightGrey", "White")
```

4. Click **Preview** to preview the report, then when you have finished, close Visual Studio 2015. If you are prompted to save changes, click **No**.

Results: At the end of this lab, you should be able to use expressions and embedded code to control the appearance of a Reporting Services report.

Exercise 2: Custom code—Report Builder

► Task 1: Prepare the lab environment

1. In the **D:\Labfiles\Lab09\Starter** folder, right-click **Setup.cmd** and click **Run as administrator**.
2. Click **Yes** when prompted to confirm you want to run the command file, and wait for the script to finish.

► Task 2: Review the requirements

- Review the requirements in the exercise scenario.

► Task 3: Format last order amount

1. Click **Start** then type **Report Builder** then press Enter.
2. In Report Builder, in the **Getting Started** window, click **Open**.
3. In the **Open Report** dialog box, in the **Name** box, type **D:\Labfiles\Lab09\Starter\Builder\OrdersReport.rdl**, then click **Open**.
4. Click **Run** to preview the report, then click **Design** to return to design mode.
5. Right-click **[LastOrderAmount]** then click **Expression**.
6. In the **Expression** dialog box, in the **Set expression for: Value** box, move the cursor after the equals sign (=).
7. In the **Category** box, expand **Common Functions** then click **Conversion**.
8. In the **Item** box, double click **Int**.
9. In the **Set expression for: Value** box, move the cursor to the end of the line, type **)**, then click **OK**.
10. Click **Run** to preview the report, then click **Design** to return to design mode.

► Task 4: Format last order date

1. In Report Builder, anywhere outside the boundary of the report body (the dark gray background) and click **Report Properties**.
2. In the **Report Properties** window, on the **Code** page, in the **Custom code** box, type the following, and then click **OK**:

```
Public Function yyyyymmdd_to_short_date (ByVal s As String) As String
    Dim myDate As Date = Date.ParseExact(s, "yyyyMMdd",
    System.Globalization.DateTimeFormatInfo.InvariantInfo)
    return myDate.ToString("dd MMM yyyy")
End Function
```

You can alternatively copy and paste the function text from
D:\Labfiles\Lab09\Starter\Code\date_function.txt.

3. In the query pane, right-click **[LastOrderDate]** then click **Expression**.
4. In the **Expression** dialog box, in the **Set expression for: Value** box, edit the expression so that it reads the following, and then click **OK**:

```
=Code.yyyyymmdd_to_short_date(Fields!LastOrderDate.Value)
```

5. Click **Run** to preview the report, then click **Design** to return to design mode.

► Task 5: Format customer name

1. In Report Builder, anywhere outside the boundary of the report body (the dark gray background) and click **Report Properties**.
2. In the **Report Properties** window, on the **Code** page, in the **Custom code** box, move the cursor to the end of the existing code, and press Enter.
3. Type the following, and then click **OK**:

```
Public Function SplitName (ByVal name As String) As String()
    Dim stringSeparators() As String = {" ",""}
    return name.Split(stringSeparators,StringSplitOptions.None)
End Function
```

You can alternatively copy and paste the function text from
D:\Labfiles\Lab09\Starter\Code\name_function.txt.

4. In the query pane, under **Last Name**, right-click **[CustomerName]** then click **Expression**.
5. In the **Expression** dialog box, in the **Set expression for: Value** box, edit the expression so that it reads, and then click **OK**:

```
=Code.SplitName(Fields!CustomerName.Value)(0)
```

6. In the query pane, under **First Name**, right-click **[CustomerName]** then click **Expression**.
7. In the **Expression** dialog box, in the **Set expression for: Value** box, edit the expression so that it reads, and then click **OK**:

```
=Code.SplitName(Fields!CustomerName.Value)(1)
```

8. Click **Run** to preview the report, then click **Design** to return to design mode

► Task 6: Alternate row colors

1. In Report Builder, click any cell in the table in the report, then click the bottom-left gray cell to select the last row of the table.
2. On the **View** menu, select the **Properties** check box.
3. In the **Properties** pane, scroll down to the **Fill** section, then click the value of the **BackgroundColor** property.
4. Click the drop-down arrow, then click **Expression**.
5. In the **Expression** window, in the **Set expression for: BackgroundColor** box, type the following, and then click **OK**:

```
=IIf(LineNumber(Nothing) Mod 2, "LightGrey", "White")
```

6. On the **Home** tab, click **Run** to preview the report, then when you have finished, close Report Builder. If you are prompted to save changes, click **No**.

Exercise 3: URL access

► Task 1: Identify the Web Service URL

1. Click **Start** then type **Reporting Services Configuration Manager** then press Enter.
2. In the **User Account Control** dialog box, click **Yes**.
3. In the **Reporting Services Configuration Connection** dialog box, confirm that the value in the **Server Name** box is **MIA-SQL**, then click **Connect**.
4. Click **Web Service URL**, then on the **Web Service URL** page note the value in the **Report Server Web Service URLs** box.
5. In File Explorer, double-click **D:\Labfiles\Lab09\Starter\URLAccess.rtf**, then note the Web Service URL under the heading that starts **1**.
6. Close Reporting Services Configuration Manager.

► Task 2: Link for Customers_Near_Stores

1. Start Internet Explorer, then navigate to **http://mia-sql/reports_sql2**.
2. Click **Adventureworks Sample Reports**, right-click **Customers_Near_Stores**, click **Manage**, and then click **Parameters**. Observe that the report takes three parameters; you need to provide values for only two—**GeoLocation**, and **Radius**.

In WordPad, under the heading that starts **2**, type the following:

```
http://mia-sql/reportserver_sql2?/AdventureWorks%20Sample%20Reports/Customers_Near_Stores&GeoLocation=POINT%20(-120.928658084295+37.6746342922879)&Radius=50
```

3. Click the text you have just typed.
4. In the **WordPad** dialog box, click **Yes**. Observe that the report opens in Internet Explorer, but that the toolbar is displayed.
5. Close the Internet Explorer tab.

6. In WordPad, at the end of the text you typed in the previous step, type the following:

```
&rc:toolbar=false
```

7. Click the text you have just typed.
8. In the **WordPad** dialog box, click **Yes**. Observe that the report opens in Internet Explorer and no toolbar is displayed.
9. Close the Internet Explorer tab.

► **Task 3: Employee_Sales_Summary as an image**

1. In Internet Explorer, in the breadcrumb trail, click **Adventureworks Sample Reports**.
2. Right-click **Employee_Sales_Summary**, click **Manage**, and then click **Parameters**. Observe that the report has six parameters; you only need to provide values for three—**EmployeeId**, **ReportMonth**, and **ReportYear**.

In WordPad, under the heading that starts **3**, type the following:

```
http://mia-sql/reportserver_sql12?/AdventureWorks%20Sample%20Reports/Employee_Sales_Summary&EmployeeID=283&ReportYear=2013&ReportMonth=11
```

3. Click the text you have just typed.
4. In the **WordPad** dialog box, click **Yes**. Observe that the report opens in Internet Explorer as a webpage.
5. Close the Internet Explorer tab.
6. In WordPad, at the end of the text you typed in the previous step, type the following:

```
&rs:format=image
```

7. Click the text you have just typed.
8. In the **WordPad** dialog click **Yes**.
9. In the Internet Explorer message bar, click **Save**, and then click **Open**. Observe that the report image opens in the Windows Photo Viewer application.
10. Close Windows Photo Viewer and the close all tabs in Internet Explorer.

Results: At the end of this lab, you should be able to build URLs for URL access to Reporting Services reports.

Module 10: Introduction to Mobile Reports

Lab: Introduction to mobile reports

Exercise 1: Format data for a mobile report

► Task 1: Prepare the lab environment

1. Ensure the **10990C-MIA-CLI**, **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab10\Starter** folder, right-click **Setup.cmd** and click **Run as administrator**.
3. Click **Yes** when prompted to confirm you want to run the command file, and wait for the script to finish.

► Task 2: Format the report data

1. Make sure the **10990C-MIA-CLI** virtual machine is running, then log on to **10990C-MIA-CLI** as **Student** with the password **Pa55w.rd**.
2. Using File Explorer, navigate to **\\mia-sql\Lab10**.
3. In the **Windows Security** dialog, in the **User name** box, type **Adventureworks\student** then in the **Password** box type **Pa55w.rd**, then click **OK**.
4. In File Explorer, double-click **Starter** then double-click **sales.xlsx** to open the workbook in Excel, and then click **Enable Editing** if necessary. To make the data in the **Sheet1** worksheet suitable for use in mobile reports, each dataset be moved to a separate worksheet.
5. On the **Home** tab, in the **Cells** group, click the **Insert** drop-down arrow, and then click **Insert Sheet**. A new sheet called **Sheet2** is added to the workbook.
6. In **Sheet1**, select the cell range **K9:L26**, right-click the highlighted area, and then click **Cut**.
7. In **Sheet2**, click cell **A1**, right-click the highlighted area, and then click **Paste**.
8. Save the workbook and close Excel.

Results: At the end of this exercise, you will have defined datasets for your new report.

Exercise 2: Create a mobile report

► Task 1: Import Excel data to a report

1. On **10990C-MIA-SQL**, on the Start page, type **Microsoft SQL Server Mobile Report Publisher**, and then press Enter. The application opens with an empty report.
2. On the **Data** tab, click **Add data**, and then click **Excel**.
3. In the **Open** dialog box, navigate to **D:\Labfiles\Lab10\Starter**, and then double-click **sales.xlsx**.
4. In the Add data window, select the **Sheet2** check box, and then click **Import**. Review the imported data to make sure it appears as you expect.

► Task 2: Add a report element

1. In SSMRP, click **Layout**, then scroll down to the bottom of the element list (on the left of the screen), then click and drag **Simple Data Grid** to the top-left cell in the report layout grid.
2. Using the resize handle in the bottom right of the data grid element, resize the element until it fills the entire grid.
3. Click **Data**, then in the **Data properties** section, in the **Data for the grid view** box, select **Sheet2**, then click **Preview**. Observe that the data from the spreadsheet is displayed in the report. Click the **Back** button (in the top left of the preview) to return to design view.

► Task 3: Use a shared dataset

1. Click **Data**, then click **Add data**, and then click **Report server**.
2. If the **Connect to a server** dialog box appears, in the **Server address** box, type **mia-sql/Reports_SQL2**.
3. Clear the **Use secure connection** check box, and then click **Connect**.
4. In the **Add data from server** dialog box, click **mia-sql/Reports_SQL2**, click **AdventureWorks Sample Reports**, and then click **SalesEmployees**.
5. When the dataset is imported, on the **Data** tab, in the **Report elements** section, click **Simple data grid 1**. In the **Data properties** section, in the **Data for the grid view** box, select **SalesEmployees**.
6. In the **Data grid columns** section, uncheck the **BusinessEntityID** check box, then click **Preview**. Review the results, then when you have finished, close SSMRP without saving any changes.

Results: At the end of this exercise, you should be able to create mobile reports from Excel or Reporting Services datasets.

Exercise 3: Create KPIs

► Task 1: Create a manual KPI

1. Start Internet Explorer, navigate to **mia-sql/Reports_SQL2**, click **New**, and then click **KPI**.
2. On the **New KPI** page, in the **KPI name** box, type **Production Line Stoppages**.
3. In the **Value format** list, ensure that **General** is selected.
4. In the **Value** list, ensure that **Set manually** is selected, and then in the **Enter value** box, type **5**.
5. In the **Goal** list, ensure that **Not set** is selected.
6. In the **Status** list, ensure that **Set manually** is selected, and then in the **Enter Status** list, click **-1 (bad)**.
7. In the **Trend set** list, ensure that **Set manually** is selected, and then in the **Enter trend set** box, type **5;0;1;0;0;6;5;1** and then click **Create**.
8. Observe the new mobile KPI in the **KPIs Section**, then leave Internet Explorer open for the next task.

► Task 2: Create a KPI from a shared dataset

1. In Internet Explorer click **New**, and then click **KPI**.
2. On the **New KPI** page, in the **KPI name** box, type **Total Sales**.

3. In the **Value format** list, click **Abbreviated currency**.
4. In the **Value** list, click **Dataset field**.
5. In the **Pick dataset field** box, click the ellipses (...).
6. In the **Choose a dataset** window, click the **AdventureWorks Sample Reports** folder, and then click **SalesByStateByMonth**.
7. In the **Choose a field from SalesByStateByMonth** window, in the **Aggregation** list, click **Sum**, click **ActualSales**, and then click **OK**.
8. In the **Goal** list, click **Dataset field**.
9. In the **Pick dataset field** box, click the ellipses (...).
10. In the **Choose a dataset** window, click the **AdventureWorks Sample Reports** folder, and then click **SalesByStateByMonth**.
11. In the **Choose a field from SalesByStateByMonth** window, in the **Aggregation** list, click **Sum**, , click **TargetSales**, and then click **OK**.
12. In the **Status** list, ensure that **Set manually** is selected.
13. In the **Enter status** list, click **1 (good)**.
14. In the **Trend set** list, click **Not set**, then click **Create**.
15. Observe the new mobile KPI in the **KPIs Section**, then close Internet Explorer.

Results: At the end of this exercise, you will have created KPIs on the server.

MCT USE ONLY. STUDENT USE PROHIBITED

Module 11: Developing Mobile Reports

Lab: Developing mobile reports

Exercise 1: Add a dataset with parameters

► Task 1: Prepare the lab environment

1. Ensure the **10990C-MIA-DC** and **10990C-MIA-SQL** virtual machines are running, and then log on to **10990C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab11\Starter** folder, right-click **Setup.cmd** and click **Run as administrator**.
3. Click **Yes** when prompted to confirm you want to run the command file, and wait for the script to finish.

► Task 2: Add the dataset

1. On the Start page, type **Microsoft SQL Server Mobile Report Publisher**, and then press Enter.
2. Click **Open mobile report**, then click **Open from file system**.
3. In the **Open** dialog box navigate to **D:\Labfiles\Lab11\Starter**, click **lab01.rsmobile** then click **Open**.
4. If a Server connection is required dialog is shown, complete the flowing steps:
 - a. Click the **Server connections** button on the application toolbar (at the upper left of the SSMRP window).
 - b. In the Connect to a server dialog box, in the Server address box, type **mia-sql/Reports_SQL2** clear the **Use secure connection** check box, and then click **Connect**.
5. Click **Data**, and observe that the report already includes three datasets. Click **Add Data** then click **Report Server**, then click **mia-sql/reports_sql2**, then click **AdventureWorks Sample Reports**, then click **PhoneSalesCommission**.
6. When the dataset is imported, click the **Cog** icon next to **PhoneSalesCommission**, then click **Param**. Click the down arrow on the first row and in the **Default value** box type **2017-01-01** then press Enter, then click the down arrow on the second row and in the **Default value** box type **2018-01-01** then press Enter, then click **Apply**.
7. At the top right of the window click **Refresh all data**.
8. Leave SSMRP open for the next exercise.

Results: At the end of this exercise, you will have imported a parameterized dataset into your report.

Exercise 2: Design a mobile report

► Task 1: Add a time navigator

1. On the **Layout** tab, click and drag **Time navigator** from the element gallery to the upper left cell of the design grid. Grab the sizer and expand the time navigator to fill three grids across and five down.
2. In the **Visual properties** area, in the **Time Levels** list, ensure only the **Years**, **Months**, and **Weeks** check boxes are selected.

3. In the **Time range presets** list, ensure only **All** is selected. Clear **Last year**, **Last 6 months**, and **Last quarter**.
4. In the **Number format** list, click **General**.
5. In the **Visualization type** list, click **Step area**.
6. Click the **Show comparison delta** switch so that its value is **On**.
7. In the **Value direction** list, click **Lower values are better**.



Note: Note that you might need to scroll to the right in the Visual properties area to find all the properties you should configure.

8. On the **Data** tab, click **Time navigator 1**.
9. In the **Data properties** region, in the **Series for background chart** list, click **AbandonedCalls**.
10. In the adjacent list, select the **Calls**, and **AbandonedCalls** check boxes.
11. In the **Comparison series for tiles data** list, click **AbandonedCalls**.
12. In the adjacent list, clear the **Calls** check box, and then select the **AbandonedCalls** check box.
13. On the **Preview** tab, click the different date ranges (month, week, day) in the time navigator to make sure you understand how it works. When you have finished previewing, click the back button to return to the **Data** tab.

► Task 2: Add a pie chart

1. On the **Layout** tab, click and drag **Pie Chart** from the element gallery to the upper left empty cell of the design grid, adjacent to the time navigator you placed in the previous step.
2. Grab the sizer and expand the pie chart to fill three grids across and five down.
3. In the **Visual properties** area, in the **Title** box, type **Abandoned Calls**.
4. In the **Number format** list, click **General**.
5. In the **Series visualization** list, click **Donut**.
6. In the **Value display mode** list, click **Percentage on Chart**.
7. Click the **Show legend** switch so that its value is **On**.



Note: Note that you might need to scroll the Visual properties area to find all the properties you should configure.

8. On the **Data** tab, click **Abandoned Calls**.
9. In the **Data properties** region, in the **Main Series** list, click **AbandonedCalls**.
10. In the adjacent list, select the **Calls** and **AbandonedCalls** check boxes.
11. On the **Preview** tab, click values in the time navigator, notice that the values on the pie chart change to reflect the data you have selected. When you have finished previewing, click the back button to return to the **Data** tab.

► Task 3: Add a selection list

1. On the **Layout** tab, click and drag **Selection list** from the element gallery to the upper left empty cell of the design grid, adjacent to the pie chart you placed in the previous step.
2. Grab the sizer and expand the selection list to fill two grids across and five down.
3. In the **Visual properties** area, in the **Title** box, type **Call Center Staff**.
4. On the **Data** tab, click **Call Center Staff**.
5. In the **Data properties** region, in the **Keys** list, click **SalesEmployees**.
6. In the adjacent list, click **BusinessEntityID**.
7. In the **Labels** list, in the adjacent list, click **Employee**.
8. With the **PhoneSalesCommission** dataset selected, in the top right click **Refresh all data**. You should now see four column headings.
9. In the **Filter these datasets when a selection is made** area, select the **PhoneSalesCommission** check box, and in the adjacent list, click **BusinessEntityId**.
10. Select the **SalesTarget** check box, and in the adjacent list, click **BusinessEntityId**.
11. Click the **PhoneSalesCommission** dataset, then click the **Cog** icon next to **PhoneSalesCommission**, then click **Param**. Click the down arrow on the third row, then click **SelectedItems** under **Call Center Staff**, then click **Apply**.

► Task 4: Add numbers

1. On the **Layout** tab, click and drag **Number** from the element gallery to the upper left empty cell of the design grid, adjacent to the selection list you placed in the previous step.
2. Grab the sizer and expand the number to fill two grids across and one down.
3. In the **Visual properties** area, in the **Title** box, type **Total Sales**.
4. In the **Number format** list, click **Abbreviated currency**.
5. Click and drag **Number** from the element gallery to the upper left empty cell of the design grid, adjacent to the selection list you placed in the previous step.
6. Grab the sizer and expand the number to fill two grids across and one down.
7. In the **Visual properties** area, in the **Title** box, type **Sales Target**.
8. In the **Number format** list, click **Abbreviated currency**.



Note: Note that you might need to scroll the Visual properties area to find all the properties you should configure.

9. On the **Data** tab, click **Total Sales**.
10. In the **Data properties** region, in the **Value** list, click **PhoneSalesCommission**.
11. In the adjacent list, click **SalesAmount**.
12. Click **Sales Target**.
13. In the **Data properties** region, in the **Value** list, click **SalesTarget**.
14. In the adjacent list, click **SalesTarget**.

► Task 5: Add a progress bar

1. On the **Layout** tab, click and drag **Progress Bar** from the element gallery to the upper left empty cell of the design grid, below the number you placed in the previous step.
2. Grab the sizer and expand the progress bar to fill two grids across and two down.
3. In the **Visual properties** area, in the **Title** box, type **Sales Target Progress**.
4. In the **Delta label** list, click **Percentage from target**.
5. Click **Set ranges**.
6. In the **Maximum** box, type **150%**.
7. In the **Neutral End** box, type **100%**.
8. In the **Neutral Start** box, type **90%**, and then click **Done**.



Note: Note that you might need to scroll the Visual Properties area to find all the properties you should configure.

9. On the **Data** tab, click **Sales Target Progress**.
10. In the **Data properties** region, in the **Main value** list, click **PhoneSalesCommission**.
11. In the adjacent list, click **SalesAmount**, and then click **Options**.
12. Ensure the **Call Center Staff** check box is selected, and then click **Done**.
13. In the **Comparison value** list, click **SalesTarget**.
14. In the adjacent list, click **SalesTarget**, and then click **Options**.
15. Ensure the **Call Center Staff** check box is selected, and then click **Done**.
16. On the **Preview** tab, click the different names in the Call Center Staff list to get a view of each staff member's progress towards their sales target. When you have finished previewing, click the back button to return to the **Data** tab. Leave SSMRP open for the next exercise.

Results: At the end of this exercise, you will have created the master view of a mobile report definition, and linked the view to datasets.

Exercise 3: Publish a mobile report

► Task 1: Define the tablet view

1. In SSMRP, on the **Layout** tab, in the upper right of the page, click the view selector, and then click **Tablet**.
2. Click and drag the **Time navigator 1** element from the Report elements gallery to the upper left cell of the design grid.
3. Grab the sizer and expand the time navigator to fill six grids across and five down.
4. Click and drag the **Total Sales** element from the Control Instances gallery to the upper leftmost empty cell of the design grid, below the time navigator.

5. Grab the sizer and expand the total sales element to fill three grids across and three down.
6. Click and drag the **Abandoned Calls** element from the Control Instances gallery to the upper leftmost empty cell of the design grid, below the time navigator.
7. Grab the sizer and expand the abandoned calls element to fill three grids across and three down.
8. On the **Preview** tab, preview the tablet view. When you have finished previewing, click the back button to return to the **Layout** tab.

► **Task 2: Define the phone view**

1. On the **Layout** tab, in the upper right of the page, click the view selector, and then click **Phone**.
2. Click and drag the **Call Center Staff** element from the Report elements gallery to the upper left cell of the design grid.
3. Grab the sizer and expand the time navigator to fill four grids across and four down.
4. Click and drag the **Sales Target Progress** element from the Report elements gallery to the upper leftmost empty cell of the design grid, below the selection list.
5. Grab the sizer and expand the time navigator to fill four grids across and two down.
6. On the **Preview** tab, preview the phone view. When you have finished previewing, click the back button to return to the **Layout** tab.

► **Task 3: Select a palette and publish the report**

1. On the **Layout** tab, in the upper right of the page, click the palette selector, click **Steel**, and then click **Done**.
2. On the **Settings** tab, in the **Report title** box, type **Call Center Tracking**.
3. On the menu bar, click **Save mobile report as**.
4. In the **Save mobile report as** dialog box, click **Save to server**.
5. In the **Save mobile report as** dialog box, ensure that the **Server** box has the value **mia-sql/reports_sql2**, and that the value in the **Location** box is **/** and then click **Save**. Leave SSMRP open for the next exercise.

► **Task 4: View the published report**

1. In Internet Explorer, navigate to **mia-sql/Reports_SQL2**, click **Call Center Tracking**. This is the master view of the report. (If **Call Center Tracking** is not visible, click **Refresh** then retry).
2. In File Explorer, navigate to **D:\Labfiles\Lab11\Starter**, right-click **browser_size.html**, point to **Open with**, and then click **Internet Explorer**.
3. In the message box, click **Allow blocked content**.
4. Click **520x730 - tablet**. This starts a new browser window in a tablet aspect ratio. In the new browser window, click **Call Center Tracking**. This is the tablet view of the report. Close the tablet view window.
5. Click **400x640 - phone**. This starts a new browser window in a phone aspect ratio. In the new browser window, click **Call Center Tracking**. This is the phone view of the report. Close all Internet Explorer windows.

Results: At the end of this exercise, you will have published a mobile report to the root folder on the **http://mia-sql/Reports_SQL2** reporting server.

Exercise 4: Add a drillthrough to a custom URL

► Task 1: Add the drillthrough

1. In SSMRP, click **Layout**. in the upper right of the page, click the view selector, and then click **Master**.
2. Click the **Total Sales** element (in the upper right of the report master view) then in the Visual properties region, click **Drillthrough target**, then click **Custom URL**.
3. On the **Set drillthrough URL** page, in the **Enter a URL to go to when this visualization is clicked** box, type:

```
http://mia-sql/reports_sql2/report/AdventureWorks Sample Reports/Sales_by_Region
```

then click **Apply**.

4. Click **Preview**, then when the report loads, click **\$3.67M** in the **Total Sales** element. The drillthrough target report will open in Internet Explorer. When you are happy that the drillthrough is working, close Internet Explorer. Close SSMRP, discarding any changes.

Results: At the end of this lab, you should be able to add drillthroughs to mobile reports.