

**APPLIED MICROSOFT
POWER BI**
Bring Your Data to Life

Teo Lachev

Foreword by Jen Underwood

Applied Microsoft Power BI

Bring your data to life!

Teo Lachev



Prologika Press

Applied Microsoft Power BI

Bring your data to life!

Published by:

Prologika Press

info@prologika.com

http://www.prologika.com

Copyright © 2016 Teo Lachev

All rights reserved. No part of this book may be reproduced, stored, or transmitted in any form or by any means, without the prior written permission of the publisher. Requests for permission should be sent to *info@prologika.com*.

Trademark names may appear in this publication. Rather than use a trademark symbol with every occurrence of a trademarked name, the names are used strictly in an editorial manner, with no intention of trademark infringement. The author has made all endeavors to adhere to trademark conventions for all companies and products that appear in this book, however, he does not guarantee the accuracy of this information.

The author has made every effort during the writing of this book to ensure accuracy of the material. However, this book only expresses the author's views and opinions. The information contained in this book is provided without warranty, either express or implied. The author, resellers or distributors, shall not be held liable for any damages caused or alleged to be caused either directly or indirectly by this book.

Printed in the United States of America

ISBN 13 978-0-9766353-6-9

ISBN 10 0-9766353-6-4

Author: Teo Lachev

Editor: Edward Price

Reviewer: Jen Underwood

Cover Designer: Zamir Creations

contents

CHAPTER 1 INTRODUCING POWER BI

PART 1 POWER BI FOR BUSINESS USERS

CHAPTER 2 THE POWER BI SERVICE

CHAPTER 3 VISUALIZING DATA

CHAPTER 4 POWER BI MOBILE

PART 2 POWER BI FOR DATA ANALYSTS

CHAPTER 5 DATA MODELING FUNDAMENTALS

CHAPTER 6 TRANSFORMING DATA

CHAPTER 7 REFINING THE MODEL

CHAPTER 8 IMPLEMENTING CALCULATIONS

PART 3 POWER BI FOR PROS

CHAPTER 9 ENABLING TEAM BI

CHAPTER 10 ORGANIZATIONAL BI

PART 4 POWER BI FOR DEVELOPERS

CHAPTER 11 PROGRAMMING FUNDAMENTALS

CHAPTER 12 EMBEDDING REPORTS

CHAPTER 13 CREATING CUSTOM VISUALS

GLOSSARY OF TERMS

foreword

For eight consecutive years, Microsoft has been positioned as a Leader in the Gartner Magic Quadrant for Business Intelligence and Analytics Platforms. While each year has seen advances across our product line, this past year's progress is truly unprecedented. We firmly believe Microsoft now offers the industry's most complete and modern business intelligence product family (with unmatched breadth and depth) on premises or in the cloud. Although our teams knew where we were headed, our customers did not. We addressed those concerns by revealing the first ever, public Business Intelligence roadmap. Now everyone knows that our future is bright with Power BI at our core.

Microsoft Power BI 2.0 has just begun to fundamentally disrupt the business intelligence market. For Microsoft business intelligence fans, the journey up to this point has been unquestionably challenging at times. Today our challenge is keeping up with all of the exciting monthly, weekly and even daily releases. Power BI development teams are moving at an astonishing pace. The data visualization framework was openly shared via GitHub empowering anyone to extend the offerings. Unlike Power BI 1.0 that went unnoticed, Power BI 2.0 is making waves in the market. Since the December 2014 Power BI preview, there has already been widespread adoption in over one million organizations across 185 countries worldwide.

Looking back there were many invaluable lessons learned from the prior unsuccessful launch. Most importantly the engineering teams learned to listen more closely to customers and partners on a daily basis. Since the cloud environment enables agile development with continuous release cycles, the teams are able to respond to market changes faster than they ever could in the past. From technical architecture to authoring tools and user experiences, the entire solution was completely reimaged and redeployed as a preview in just a few months. The cloud is a game changer.

One of the most significant and sensational changes was the introduction of free Power BI Desktop. Power BI Desktop unifies the former Excel Power Tools (Power Pivot, Power Query and Power View) into one vastly improved, stand-alone, data discovery desktop application built on a modernized HTML5 visualization framework. Unlike the Excel Power Tools predecessors, Power BI Desktop has absolutely no dependencies on Excel or Office. Power BI Desktop removes adoption friction, improves the analytics authoring user experience, allows for third-party extension and ultimately is the result of Microsoft listening to the analytics market. The move of Power BI outside of Excel was no easy feat politically and technically.

The July 2015 GA release of Power BI 2.0 unveiled a sleek new design along with the new visualization framework and scalable hybrid cloud architectural foundation. It also brought native mobile apps for iOS, Android and Windows tablets and smart phones. The new architecture now includes a plethora of in-memory or direct connect data sources for

large scale deployments. To further supplement the third-party open source custom data visualization extensions, R script integration was added to open up a world of unlimited visualization options. Pre-packaged Software-as-a-Service (SaaS) and organizational level data models, reports and dashboards can be shared via content packs making Power BI easier than ever for non-technical users to enjoy.

Natural language query has improved and now includes voice commands via Cortana. Intelligent automated analytics is just starting to surface with the latest Get Insights release that highlights key findings in a data set from identifying outliers to noting significant changes. Power BI Enterprise Gateway, Developer APIs and the sea of incremental feature releases continuously surprises and delights a growing worldwide community. It has never been a better time to be a Microsoft BI fan. Even our skeptics have voiced appreciation for the changes in our development approach and our latest offerings.

As Power BI swiftly blossoms into the market leading business intelligence solution, it will be a weapon that all data analysts will want to include in their analytics arsenal. The true power in Power BI cannot be appreciated without understanding what the offering can do and how to best use it. That is why resources like this fantastic book will become instrumental for you. This book starts by providing an overview of the foundational components of Power BI. It introduces Power BI Desktop, data modeling concepts, building reports, publishing and designing dashboards. Readers will be up and running in no time.

It then moves on to bring you up to speed on deeper dive topics such as data gateways, data refresh, streaming analytics, embedding and the Power BI data visualization API. Not only is Teo one of the first people in the world to learn and write about Power BI 2.0, he also brings a wealth of knowledge from deploying the first real-world implementations. Much like Teo's previous books on Analysis Services and Reporting Services, this Power BI book will be a must read for serious Microsoft professionals. It will also empower data analysts and enthusiasts everywhere.

On a closing note, please share your feedback with Teo and the Microsoft Power BI teams as you embark on your own Power BI journey. Microsoft Power BI teams do actively monitor social media channels, the Power BI community site, external blogs, YouTube videos, and User Voice more closely than ever before. You can be heard. You can make a difference. This is not only a new, different and better Power BI 2.0 – it is a new, different and better Microsoft.

Jen Underwood
Principal Program Manager
Microsoft Business Intelligence

preface

To me, Power BI is the most exciting milestone in the Microsoft BI journey since circa 2005, when Microsoft got serious about BI. Power BI changes the way you gain insights from data; it brings you a cloud-hosted, business intelligence and analytics platform that democratizes and opens BI to everyone. It does so under a simple promise: “five seconds to sign up, five minutes to wow!”

Power BI has plenty to offer to all types of users who’re interested in data analytics. If you are an information worker, who doesn’t have the time and patience to learn data modeling, Power BI lets you connect to many popular cloud services (Microsoft releases new ones every week!) and get insights from prepackaged dashboards and reports. If you consider yourself a data analyst, you can implement sophisticated self-service models whose features are on a par with organizational models built by BI pros.

Speaking of BI pros, Power BI doesn’t leave us out. We can architect hybrid organizational solutions that don’t require moving data to the cloud. And besides classic solutions for descriptive analytics, we can implement innovative Power BI-centric solutions for real-time and predictive analytics. If you’re a developer, you’ll love the Power BI open architecture because you can integrate custom applications with Power BI and visualize data your way by extending its visualization framework.

From a management standpoint, Power BI is a huge shift in the right direction for Microsoft and for Microsoft BI practitioners. Not so long ago, Microsoft BI revolved exclusively around Excel on the desktop and SharePoint Server for team BI. This strategy proved to be problematic because of its cost, maintenance, and adoption challenges. Power BI overcomes these challenges. Because it has no dependencies to other products, it removes adoption barriers. Power BI gets better every week and this should allow us to stay at the forefront of the BI market. As a Power BI user you’re always on the latest and greatest version. And Power BI has the best business model: most of it it’s free!

I worked closely with Microsoft’s product groups to provide an authoritative (yet independent) view of this technology and to help you understand where and how to use it. Over more than a decade in BI, I’ve gathered plenty of real-life experience in solving data challenges and helping clients make sense of data. I decided to write this book to share with you this knowledge, and to help you use the technology appropriately and efficiently. As its name suggests, the main objective of *Applied Microsoft Power BI* is to teach you the practical skills to take the most of Power BI from whatever angle you’d like to approach it.

Some people discouraged me to write this book. After all, trying to cover a product that changes every week is like trying to hit a moving target! However, I believe that the product’s fundamentals won’t change and once you grasp them, you can easily add on knowledge as Power BI evolves over time. Because I had to draw a line somewhere,

Applied Microsoft Power BI covers all features that were announced at the PASS Summit 2015 and that were released by December 2015.

Although this book is designed as a comprehensive guide to Power BI, it's likely that you might have questions or comments. As with my previous books, I'm committed to help my readers with book-related questions and welcome all feedback on the book discussion forums on my company's web site (<http://www.prologika.com/cs/forums>). Consider following my blog at <http://prologika.com/cs/blogs> and subscribing to my newsletter at www.prologika.com to stay on the Power BI latest. Happy data analytics with Power BI!

Teo Lachev
Atlanta, GA

acknowledgements

Writing books is hard! And writing a book about a cloud platform, which adds features weekly, is even harder. On the upside, I can claim that this book has no bugs. After all, if something doesn't work now, it used to work before, right? On the downside, I had to change the manuscript every time a new feature popped up. Fortunately, I had people who supported me.

The book (my seventh) would not have been a reality without the help of many people to whom I'm thankful. As always, I'd like to first thank my family for their ongoing support.



The main personas mentioned throughout the book, as imagined by my 12-year old son, Martin, and 15-year old daughter, Maya.

As a Microsoft Most Valuable Professional (MVP), I've been privileged to enjoy close relationships with the Microsoft product groups. It's great to see them working together! I must mention a few names. Jen Underwood (Principal Program Manager at the Power BI team) contributed the most to this book! She helped me connect the Power BI dots (bars), reviewed the book manuscript, and provided valuable feedback. Thanks to Lukasz Pawlowski (Senior Program Manager at the Power BI team) for shedding light on report embedding.

As always, the Analysis Services team has been very responsive. Special thanks to

Akshai Mirchandani, Ashvini Sharma, Jeffrey Wang, Kasper de Jonge, and Marius Dumitru. Thanks to my editor, Ed Price, for helping me polish my writings.

Finally, thank *you* for purchasing this book!

about the book

The book doesn't assume any prior experience with data analytics. It's designed as an easy-to-follow guide for navigating the personal-team-organizational BI continuum with Power BI and shows you how the technology can benefit the four types of users: information workers, data analysts, pros, and developers. It starts by introducing you to the Microsoft Data Platform and to Power BI. You need to know that each chapter builds upon the previous ones, to introduce new concepts and to practice them with step-by-step exercises. Therefore, I'd recommend you read the chapters and do the exercises in the order they appear in the book.

Part 1, *Power BI for Information Works*, teaches regular users interested in basic data analytics how to analyze simple datasets without modeling and how to analyze data from popular cloud services with predefined dashboards and reports. Chapter 2, *The Power BI Service*, lays out the foundation of personal BI, and teaches you how to connect to your data. In Chapter 3, *Visualizing Data*, information workers will learn how to create their own reports and dashboards. Chapter 4, *Power BI Mobile*, discusses the Power BI native mobile applications that allow you to view and annotate BI content on the go.

Part 2, *Power BI for Data Analysts*, educates power users how to create self-service data models with Power BI Desktop. Chapter 5, *Data Modeling Fundamentals*, lays out the ground work to understand self-service data modeling and shows you how to import data from virtually everywhere. Because source data is almost never clean, Chapter 6, *Transforming Data*, shows you how you can leverage the unique query capabilities of Power BI Desktop to transform and shape the data. Chapter 7, *Refining the Model*, shows you how to make your self-service model more intuitive and how to join data from different data sources. And, in Chapter 8, *Implementing Calculations*, you'll further extend the model with useful business calculations.

Part 3, *Power BI for Pros*, teaches IT pros how to set up a secured environment for sharing and collaboration, and it teaches BI pros how to implement Power BI-centric solutions. Chapter 9, *Enabling Team BI*, shows you how to use Power BI workspaces and organizational content packs to promote sharing and collaboration, where multiple coworkers work on the same BI artifacts, and how to centralize access to on-premises data. Written for BI pros, Chapter 10, *Organizational BI*, walks you through the steps to implement descriptive, predictive, and real-time solutions that integrate with Power BI.

Part 4, *Power BI for Developers*, shows developers how to integrate and extend Power BI. Chapter 11, *Programming Fundamentals*, introduces you to the Power BI REST APIs and teaches you how to use OAuth to authenticate custom applications with Power BI. In Chapter 12, *Embedding Reports*, you'll learn how to report-enable custom applications with embedded dashboards and reports. In Chapter 13, *Creating Custom Visuals*, you'll learn how to extend the Power BI visualization capabilities by creating custom visuals to

present effectively any data.

source code

Applied Microsoft Power BI covers the entire spectrum of Power BI features for meeting the data analytics needs of information workers, data analysts, pros, and developers. This requires installing and configuring various software products and technologies. **Table 1** lists the software that you need for all the exercises in the book. Depending on your computer setup, you might need to download and install other components, as I explain throughout the book.

Table 1 The complete software requirements for practices and code samples in the book

Software	Setup	Purpose	Chapters
Power BI Desktop	Required	Implementing self-service data models	5, 6, 7, 8
Visual Studio 2012 (or higher) Professional	Required	Power BI programming	11, 12, 13
Power BI Mobile native apps (iOS, Android, or Windows depending on your mobile device)	Recommended	Practicing Power BI mobile capabilities	4
SQL Server Database Engine Developer, Standard, or Enterprise 2012 or later with the AdventureWorksDW database	Recommended	Importing and processing data	5
Analysis Services Tabular Developer, Business Intelligence, or Enterprise 2012 or later edition	Recommended	Live connectivity to Tabular	2, 10
Analysis Services Multidimensional Developer, Standard, Business Intelligence, or Enterprise 2012 or later edition	Optional	Live connectivity to Multidimensional	5
Reporting Services Developer, Standard, Business Intelligence, or Enterprise 2012 or later	Optional	Importing from SSRS	5

Although the list is long, don't despair! As you can see, most of the software is not required. In addition, the book provides the source data as text files and it has alternative steps to complete the exercises if you don't install some of the software, such as SQL Server or Analysis Services.

You can download the book source code from the book page at <http://bit.ly/powerbobook>.

After downloading the zip file, extract it to any folder of your hard drive. Once this is done, you'll see a folder for each chapter that contains the source code for that chapter. The source code in each folder includes the changes you need to make in the exercises in the corresponding chapter, plus any supporting files required for the exercises. For example, the Adventure Works.pbix file in the Ch05 folder includes the changes that you'll make during the Chapter 5 practices and includes additional files for importing data. Save your files under different names or in different folders in order to avoid overwriting the files that are included in the source code.

NOTE The data source settings of the sample Power BI Desktop models in this book have connection strings to databases and text files. If you decide to test the provided samples and refresh the data, you have to update some data sources to reflect your specific setup. To do so, open the Power BI Desktop model, and then click the Edit Queries

button in the ribbon's Home tab. Select the query that fails to refresh in the Queries pane, and then double-click the Source step in the Applied Steps list (Query Settings pane). Change the server name or file location as needed.

Installing the Adventure Works databases

Some of the code samples import data from the AdventureWorksDW database. This is a Microsoft-provided database that simulates a data warehouse.

NOTE Microsoft ships Adventure Works databases with each version of SQL Server. More recent versions of the databases have incremental changes and they might have different data. Although the book exercises were tested with the AdventureWorksDW2012 database, you can use a later version if you want. Depending on the database version you install, you might find that reports might show somewhat different data.

Follow these steps to download the AdventureWorksDW2012 database:

1. Open the Microsoft SQL Server Product Samples Database webpage on Codeplex (<http://msftdbprodsamples.codeplex.com>).
2. Click the SQL Server 2012 DW tile. The link URL as of the time of this writing is <http://msftdbprodsamples.codeplex.com/releases/view/55330>. Click the AdventureWorksDW2012 Data File link.
3. When Internet Explorer prompts you, click Run to download the file.
4. Open SQL Server Management Studio (SSMS) and connect to your SQL Server database instance. Attach the AdventureWorksDW2012_Data.mdf file. If you're not sure how to attach a database file, read the instructions at <https://msdn.microsoft.com/en-us/library/ms190209.aspx>.

Installing the Adventure Works Analysis Services models

In chapter 2 and 10, you connect to the Adventure Works Tabular model, and chapter 5 has an exercise for importing data from the Adventure Works Multidimensional cube. If you decide to do these exercises, install the Analysis Services models as follows:

1. Open the Microsoft SQL Server Product Samples Database webpage on Codeplex (<http://msftdbprodsamples.codeplex.com>).
2. Click the SQL Server 2012 DW tile. The link URL as of the time of this writing is <http://msftdbprodsamples.codeplex.com/releases/view/55330>.
3. Click the “AdventureWorks Multidimensional Models SQL Server 2012” link to download the zip file.
4. Follow the steps in the “Readme for Analysis Services Tutorial on Multidimensional Modeling” section of the of the “SQL Server Samples Readme” document at <http://bit.ly/1PwLLP2> to deploy the Adventure Works cube.
5. Back to the SQL Server 2012 DW Codeplex page, download and unzip the “AdventureWorks Tabular Model SQL Server 2012” file.
6. Follow the steps in the “Readme for Adventure Works DW Tabular SQL 2012” section of the of the “SQL Server Samples Readme” document at <http://bit.ly/1PwLLP2> to deploy the Adventure Works Tabular model.
7. In SQL Server Management Studio, connect to your Analysis Services instance. (Multidimensional and Tabular must be installed on separate instances.)
8. Expand the Databases folder. You should see the SSAS database.

Reporting errors

Please submit bug reports to the book discussion list on <http://prologika.com/cs/forums>. Confirmed bugs and inaccuracies will be published to the book errata document. A link to the errata document is provided in the book web page. The book includes links to web resources for further study. Due to the transient nature of the Internet, some links might be no longer valid or might be broken. Searching for the document title is usually sufficient to recover the new link.

Your purchase of APPLIED MICROSOFT POWER BI includes free access to a web forum sponsored by the author, where you can make comments about the book, ask technical questions, and receive help from the author and the community. The author is not committed to a specific amount of participation or successful resolution of the question and his participation remains voluntary. You can subscribe to the forum from the author's personal website <http://prologika.com/cs/forums>.

about the author

Teo Lachev is a consultant, author, and mentor, with a focus on Microsoft Business Intelligence. Through his Atlanta-based company “Prologika” (a Microsoft Gold Partner in Data Analytics) he designs and implements innovative solutions that bring tremendous value to his customers. Teo has authored and co-authored several SQL Server BI books, and he has been leading the Atlanta Microsoft Business Intelligence group since he founded it in 2010. Microsoft has recognized Teo’s expertise and contributions to the technical community by awarding him the prestigious Microsoft Most Valuable Professional (MVP) status for twelve consecutive years.

Chapter 1

Introducing Power BI

Without supporting data, you are just another person with an opinion. But data is useless if you can't derive knowledge from it. And, this is where Microsoft data analytics and Power BI can help! Power BI changes the way you gain insights from data; it brings you a cloud-hosted, business intelligence and analytics platform that democratizes and opens BI to everyone. Power BI makes data analytics pervasive and accessible to all users under a simple promise "five seconds to sign up, five minutes to wow!"

This guide discusses the capabilities of Power BI, and this chapter introduces its innovative features. I'll start by explaining how Power BI fits into the Microsoft Data Platform and when to use it. You'll learn what Power BI can do for different types of users, including business users, data analysts, professionals, and developers. I'll also take you on a tour of the Power BI features and its toolset.

1.1 What is Microsoft Power BI?

Before I show you what Power BI is, I'll explain business intelligence (BI). You'll probably be surprised to learn that even BI professionals disagree about its definition. In fact, Forester Research offers two definitions (see https://en.wikipedia.org/wiki/Business_intelligence).

DEFINITION Broadly defined, BI is a set of methodologies, processes, architectures, and technologies that transform raw data into meaningful and useful information that's used to enable more effective strategic, tactical, and operational insights and decision-making. A narrower definition of BI might refer to just the top layers of the BI architectural stack, such as reporting, analytics, and dashboards.

Regardless of which definition you follow, Power BI can help you with your data analytics needs.

1.1.1 Understanding Business Intelligence

The definition above is a good a starting point but to understand BI better, you need to understand its flavors. First, I'll categorize who's producing the BI artifacts, and then I'll show you the different types of analytical tasks that these producers perform.

Understanding BI usage scenarios

I'll classify BI by its main users and produced artifacts and divide it into self-service, team, and organizational BI.

- Self-service BI (or personal BI) – Self-service BI enables data analysts to offload effort from IT pros. From example, Maya is a business user and she wants to analyze CRM data from Salesforce. Maya can connect to Salesforce and get prepackaged dashboards and reports without building a data model. In the more advanced scenario, Power BI empowers analysts to build data models for self-service data exploration and reporting. Suppose that Martin from the sales department wants to analyze some sales data that's stored in a Microsoft Access database or in an Excel workbook. With a few clicks, Martin can import the data from various data sources into a data model (similar to the one shown in **Figure 1.1**), build reports, and gain valuable insights. In other words, Power BI opens makes data analytics more pervasive because it enables more employees to perform BI tasks.

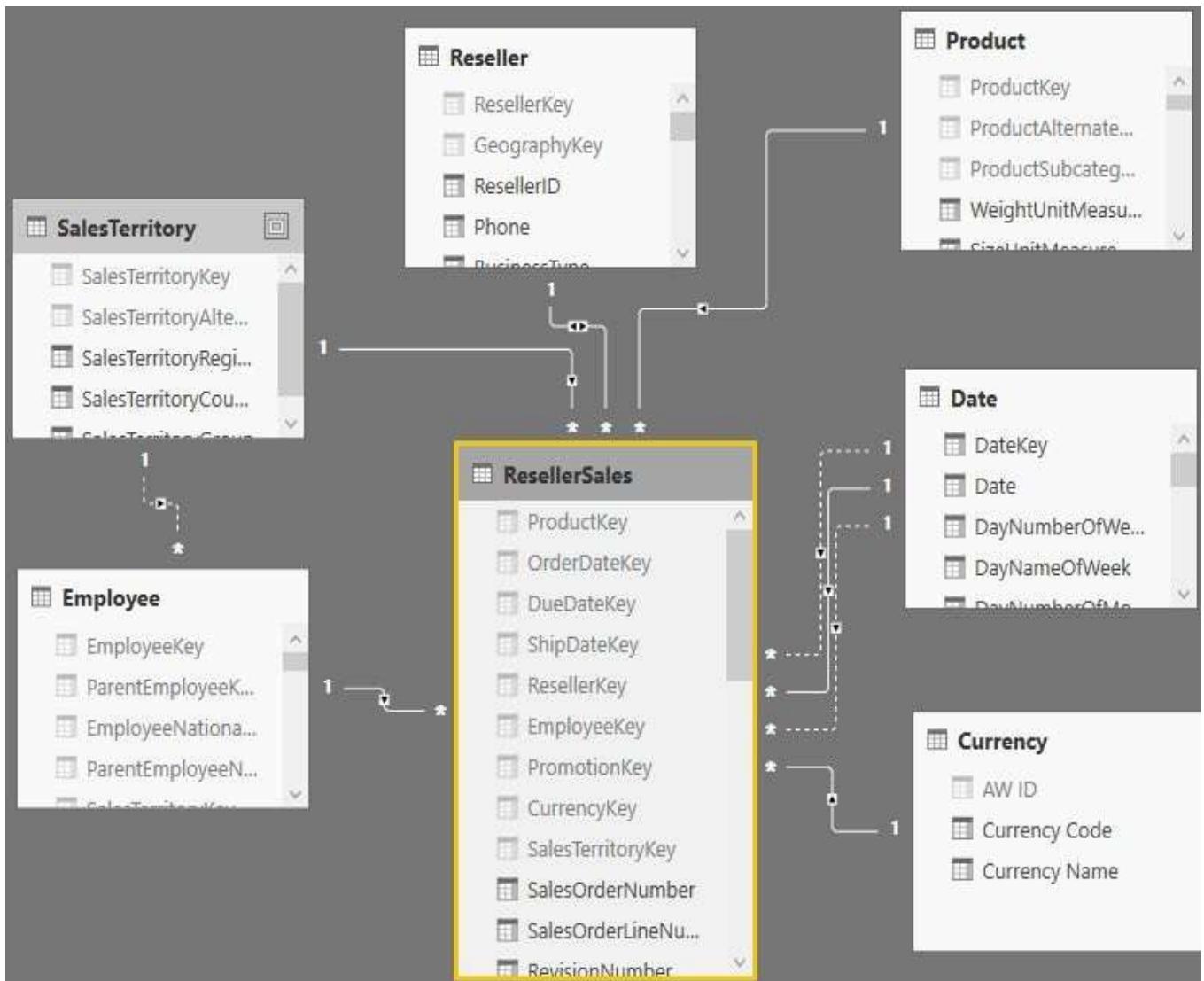


Figure 1.1 Power BI allows business users to build data models whose features are on par with professional models implemented by BI pros.

- **Team BI** – Business users can share the reports and dashboards they've implemented with other team members without requiring them to install modeling or reporting tools. Suppose that Martin would like to share his sales model with his coworker, Maya. Once Martin has uploaded the model to Power BI, Maya can go online and view the reports and dashboards Martin has shared with her. She can even create her own reports and dashboards that connect to Martin's model.
- **Organizational BI (or corporate BI)** – BI professionals who implement Microsoft SQL Server Analysis Services Multidimensional and Tabular models will find that Power BI allows them to implement hybrid solutions that eliminate the need to move data to Power BI. For example, as a BI pro, Elena has developed a Multidimensional or Tabular model layered on top of the company's data warehouse. Elena can install connectivity software on an on-premises computer so that Power BI can connect to her model. This allows business users to create instant reports and dashboards in Power BI by leveraging the existing investment in Analysis Services without moving data to the cloud!

NOTE To learn more about Analysis Services, I covered implementing Analysis Services Multidimensional models in my books “Applied Microsoft Analysis Services 2005” and Tabular models in “Applied Microsoft SQL Server 2012 Analysis Services: Tabular Modeling”.

Understanding types of data analytics

The main goal of BI is to get actionable insights that lead to smarter decisions and better business outcomes. There are three types of data analytics (descriptive, predictive, and prescriptive) that can help users achieve this goal.

Descriptive analytics is retrospective. It focuses on what has happened in the past to understand the company's performance. This type of analytics is the most common and well understood. Coupled with a good data exploration tool, such as Power BI or Microsoft Excel, descriptive analytics helps you discover import trends and understand the factors that led to these trends. You do descriptive analytics when you slice and dice data. For example, a business analyst can create a Power BI report to discover sale trends by year. Descriptive analytics can answer questions, such as "Who are my top 10 customers?", "What is the company's sales by year, quarter, month, and so on?", "How does the company's profit compare against the predefined goal by business unit, product, time, and other subject areas?"

Predictive analytics is concerned with what will happen in the future. It uses data mining and machine learning algorithms determine probable future outcomes and discover patterns that might not be easily discernible based on historical data. These hidden patterns can't be discovered with traditional data exploration since data relationships might be too complex or because there's too much data for a human to analyze. Typical data mining tasks include forecasting, customer profiling, and basket analysis. Data mining can answer questions, such as, "What are the forecasted sales numbers for the next few months?", "What other products is a customer likely to buy along with the product he or she already chose?", and, "What type of customer (described in terms of gender, age group, income, and so on) is likely to buy a given product?" As it stands Power BI doesn't have native predictive capabilities but it can be integrated with other services and products, such as Azure Machine Learning and R. For example, an analyst can build a predictive model with the Azure Machine Learning service and then visualize the results in Power BI. Or, he can import forecasted data from an R script and then present it on a Power BI report.

Finally, *prescriptive analytics* goes beyond predictive analytics to not only attempt to predict the future but also recommend the best course of action and the implications of each decision option. Typical prescriptive tasks are optimization, simulation, and goal seek. While tools for descriptive and predictive needs have matured, prescriptive analytics is a newcomer and currently is in the realm of startup companies. The good news is that you can get prepackaged advanced analytics and prescriptive solutions with Cortana Analytics Suite, such as solutions for product recommendations and customer churn. In July 2015, Microsoft unveiled Cortana Analytics Suite as "a fully managed big data and advanced analytics suite that enables you to transform your data into intelligent action". The suite includes various cloud-based services, such as Azure Machine Learning for predictive analytics, Stream Analytics for real-time BI, and Power BI for dashboards and reporting. I'll show you some of these capabilities, including the Cortana digital assistant in Chapter 3, and Azure Machine Learning and Stream Analytics in Chapter 10.

1.1.2 Introducing the Power BI Products

Now that you understand BI better, let's discuss what Power BI is. Power BI is a set of products and services that enables you to connect to your data, visualize it, and share insights with other users. At a high level, Power BI consists of three products:

- Power BI Service – A *cloud-based* business analytics service (powerbi.com) that allows you to host your data, reports and dashboards online and share them with your coworkers. Because Power BI is hosted in the cloud and maintained by Microsoft, your organization doesn't have to purchase, install, and maintain an on-premises infrastructure. Microsoft delivers weekly updates to the Power BI Service so the pace of innovation and improvement will continue unabated. To stay up to date with the latest features, follow the Power BI blog (<http://blogs.msdn.com/b/powerbi>).
- Power BI Mobile – A set of native applications for iOS, Android, and Windows that allow users to use mobile devices, such as tablets and smartphones, to get data insights on the go. For example, a mobile user can view and interact with reports and dashboards deployed to Power BI.
- Power BI Desktop – A freely available Windows desktop application that allows analysts to design data models and reports. For readers familiar with Power Pivot for Excel, Power BI Desktop offers similar self-service BI features in a standalone application outside Excel.

DEFINITION Microsoft Power BI is a data analytics platform for self-service, team, and organizational BI that consists of Power BI Service, Power BI Mobile and Power BI Desktop products. Sometimes referred to as Power BI 2.0, it replaces Power BI for Office 365, and it doesn't require an Office 365 subscription.

As you could imagine, Power BI is a versatile platform that enables different groups of users to implement a wide range of BI solutions depending on the task at hand.

1.1.3 How Did We Get Here?

Before I delve into the Power BI capabilities, let's step back for a moment and review what events led to its existence. **Figure 1.2** shows the major milestones in the Power BI journey.

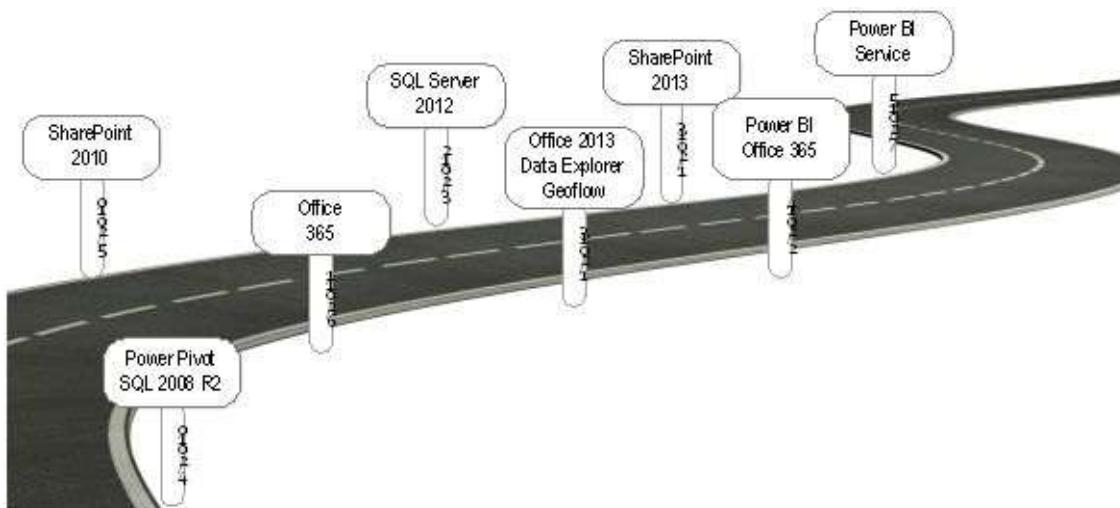


Figure 1.2 Important milestones related to Power BI.

Power Pivot

Realizing the growing importance of self-service BI, in 2010 Microsoft introduced a new technology for personal and team BI called PowerPivot (renamed to Power Pivot in 2013 as a result of Power BI rebranding). Power Pivot was initially implemented as a freely available add-in to Excel 2010 that had to be manually downloaded and installed. Office 2013 delivered deeper integration with Power Pivot, including distributing it with Excel 2013 and allowing users to import data directly into the Power Pivot data model.

NOTE I covered Excel and Power Pivot data modelling in my book “Applied Microsoft SQL Server 2012 Analysis Services: Tabular Modeling”. Although the book targets Excel 2010, it should give you the necessary foundation to understand Power Pivot and learn how to use it to implement self-service data models and how to integrate them with SharePoint Server.

The Power Pivot innovative engine, called xVelocity, transcended the limitations of the Excel native pivot reports. It allows users to load multiple datasets and import more than one million rows (the maximum number of rows that can fit in an Excel spreadsheet). xVelocity compresses the data efficiently and stores it in the computer’s main memory. For example, using Power Pivot, a business user can import data from a variety of data sources, relate the data, and create a data model. Then the user can create pivot reports or Power View reports to gain insights from the data model.

DEFINITION xVelocity is a data engine that compresses and stored data in memory. Originally introduced in Power Pivot, the xVelocity data engine has a very important role in Microsoft BI. xVelocity is now included in other Microsoft offerings, including SQL Server columnstore indexes, Tabular models in Analysis Services, Power BI Desktop, and Power BI.

SQL Server

Originally developed as a relational database management system (RDBMS), Microsoft SQL Server is now a multi-product offering. In the context of organizational BI, SQL Server includes Analysis Services which has traditionally allowed BI professionals to implement multidimensional cubes. SQL Server 2012 introduced another path for implementing organizational models called Tabular. Think of Analysis Services Tabular as Power Pivot on steroids. Just like Power Pivot, Tabular allows you to create in-memory data models but it also adds security and performance features to allow BI pros to scale these models and implement data security that is more granular.

SQL Server includes also Reporting Services which has been traditionally used to implement paper-oriented standard reports. However, SQL Server 2012 introduced a SharePoint 2010-integrated reporting tool, named Power View, for authoring ad hoc interactive reports. Power View targets business users without requiring query knowledge and report authoring experience. Suppose that Martin has uploaded his Power Pivot model to SharePoint Server. Now Maya (or anyone else who has access to the model) can quickly build a great-looking tabular or chart report in a few minutes to visualize the data from the Power Pivot model. Or, Maya can use Power View to explore data in Multidimensional or Tabular organizational model.

In Office 2013, Microsoft integrated Power View with Excel 2013 to allow business users to create interactive reports from Power Pivot models and organizational Tabular models. And Excel 2016 extended Power View to connect to multidimensional cubes.

This enhanced version of Power View enables reports and dashboards in Power BI.

SharePoint Server

Up to the release of Power BI, Microsoft BI has been intertwined with SharePoint. SharePoint Server is a Microsoft on-premises product for document storage, collaboration, and business intelligence. In SharePoint Server 2010, Microsoft added new services, collectively referred to as Power Pivot for SharePoint, which allowed users to deploy Power Pivot data models to SharePoint and then share reports that connect to these data models. For example, a business user can upload the Excel file containing a data model and reports to SharePoint. Authorized users can view the embedded reports and create their own reports.

SharePoint Server 2013 brought better integration with Power Pivot and support for data models and reports created in Excel 2013. When integrated with SQL Server 2012, SharePoint Server 2013 offers other compelling BI features, including deploying and managing SQL Server Reporting Services (SSRS) reports, team BI powered by Power Pivot for SharePoint, and PerformancePoint Services dashboards.

Microsoft Excel

While SharePoint Server has been the Microsoft premium server-based platform for BI, Microsoft Excel has been their premium BI tool on the desktop. Besides Power Pivot and Power View, which I already introduced, Microsoft added other BI-related add-ins to extend the Excel data analytics features. To help end users perform predictive tasks in Excel, Microsoft released a Data Mining add-in for Microsoft Excel 2007, which is also available with newer Excel versions. For example, using this add-in an analyst can perform a market basket analysis, such as to find which products customers tend to buy together.

NOTE In 2014, Microsoft introduced a cloud-based Azure Machine Learning Service (<http://azure.microsoft.com/en-us/services/machine-learning>) to allow users to create predictive models in the cloud, such as a model that predicts the customer churn probability. Azure Machine Learning supersedes the Data Mining add-in for self-service predictive analytics.

In January 2013, Microsoft introduced a freely available Data Explorer add-in, which was later renamed to Power Query. Unique in the self-service BI tools market, Power Query allows business users to transform and cleanse data before it's imported. For example, Martin can use Power Query to replace wrong values in the source data or to un-pivot a crosstab report. In Excel, Power Query is an optional path for importing data. If data doesn't require transformation, a business user can directly import the data using the Excel or Power Pivot data import capabilities. However, Power BI always uses Power Query when you import data so that its data transformation capabilities are there if you need them.

Another data analytics add-in that deserves attention is Power Map. Originally named Geoflow, Power Map is another freely available Excel add-in that's specifically designed for geospatial reporting. Using Power Map, a business user can create interactive 3D maps, such as the one shown in **Figure 1.3**. In this case, Power Map is used to analyze the correlation of power consumption and the age of the buildings in a particular geographic

region. You can get some of the Power Map capabilities in Power BI when you import the GlobeMap custom visual from the Power BI visual gallery (<http://visual.powerbi.com>).

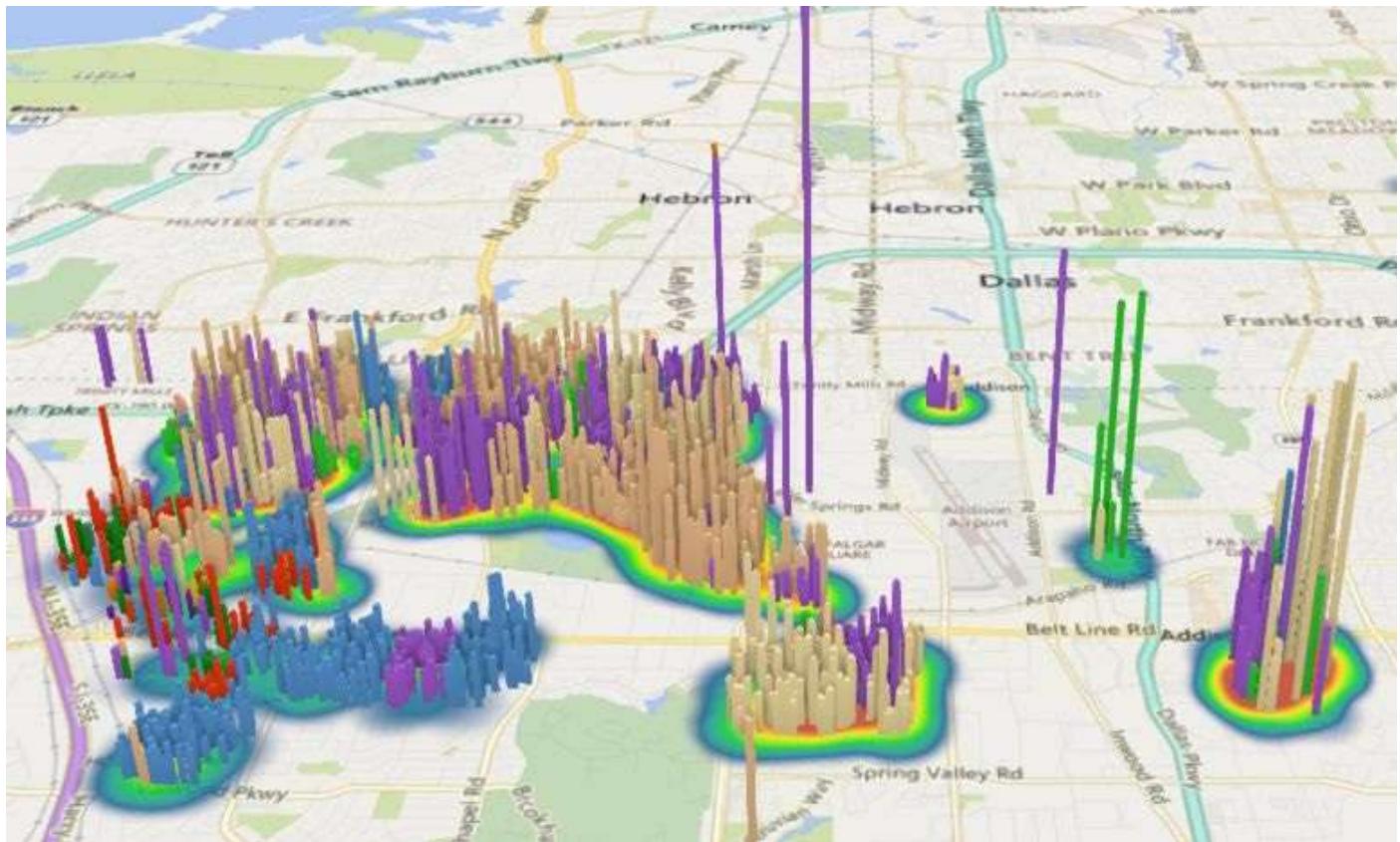


Figure 1.3 A free Excel add-in, Power Map enables you to analyze geospatial data by creating 3D visualizations with Bing maps.

Power BI for Office 365

Unless you live under a rock, you know that one of the most prominent IT trends nowadays is toward cloud computing. Chances are that your organization is already using the Microsoft Azure Services Platform - a Microsoft cloud offering for hosting and scaling applications and databases through Microsoft datacenters. Microsoft Azure gives you the ability to focus on your business and to outsource infrastructure maintenance to Microsoft.

In 2011, Microsoft unveiled its Office 365 cloud service to allow organizations to subscribe to and use a variety of Microsoft products online, including Microsoft Exchange and SharePoint. For example, at Prologika we use Office 365 for email, a subscription-based (click-to-run) version of Microsoft Office, OneDrive for Business, Skype for Business, and other products. From a BI standpoint, Office 365 allows business users to deploy Excel workbooks and Power Pivot data models to the cloud. Then they can view the embedded reports online, create new reports, and share BI artifacts with other users.

In early 2014, Microsoft further extended SharePoint for Office 365 with additional BI features, including natural queries (Q&A), searching and discovering organizational datasets, and mobile support for Power View reports. Together with the “power” desktop add-ins (Power Pivot, Power View, Power Query, and Power Map), the service was marketed and sold under the name “Power BI for Office 365”. While the desktop add-ins were freely available, Power BI for Office 365 required a subscription. Microsoft sold Power BI for Office 365 independently or as an add-on to Office 365 business plans.

Power BI for Office 365 eliminated the need to host SharePoint Server on premises. For example, if Martin’s organization didn’t want to install and maintain SharePoint on premises, they could purchase a Power BI for Office 365 subscription plan. This allows Martin to deploy and share Power Pivot data models, just like he can do by deploying them to SharePoint Server on premises. Surpassing the SharePoint Server BI capabilities, Power BI for Office 365 also allows business users to type in natural queries to gain insights from Martin’s data model, such as “show me sales by year”. Behind the scenes, Power BI for Office 365 would interpret the question and use a suitable Power View visualization to display the results.

Data discovery is a big issue with larger organizations. Another feature of Power BI for Office 365 is sharing and discovering Power Query-based datasets. It allows a data steward to publish curated queries, such as a query that returns a list of the company’s products (only the query is published, not the data). Then, other users can search and discover this query, and then use it to import the list of products in their self-service data model.

NOTE In July 2015, Microsoft introduced a new cloud service outside Office 365, called Azure Data Catalog (<http://azure.microsoft.com/en-us/services/data-catalog>). This service extends Power Query dataset sharing and discovery.

Power BI Service (Power BI 2.0)

Finally, the winding road brings us to Power BI which is the subject of this book. In July 2015, after several months of public preview, Microsoft officially launched a standalone version of Power BI (initially referred to as Power BI 2.0) that had no dependencies on Office 365, SharePoint and Microsoft Office. What caused this change? The short answer is removing adoption barriers for both Microsoft and consumers. For Microsoft it became clear that to be competitive in today’s fast-paced marketplace, its BI offerings can’t depend on other product groups and release cycles. Waiting for new product releases on two and three-year cadences couldn’t introduce the new features Microsoft needed to compete effectively with “pure” BI vendors (competitors who focus only on BI tools) who have entered the BI market in the past few years.

After more than a decade working with different BI technologies and many customers, I do believe that Microsoft BI is the best and most comprehensive BI platform on the market! But it’s not perfect. One ongoing challenge is coordinating BI features across product groups. Take for example SharePoint, which Microsoft promoted as a platform for sharing BI artifacts. Major effort underwent to extend SharePoint with SSRS in SharePoint integration mode, PerformancePoint, Power Pivot, and so on. But these products are owned by different product groups and apparently coordination has been problematic. For example, after years of promises for mobile rendering, Power View in SharePoint Server still requires Microsoft Silverlight for rendering, thus barring access from non-Windows devices.

Seeking a stronger motivation for customers to upgrade, Excel added the “power” add-ins and was promoted as the Microsoft premium BI tool on the desktop. However, the Excel dependency turned out to be a double-edge sword. While there could be a billion Excel users worldwide, adding a new feature has to be thoroughly tested to ensure that

there are no backward compatibility issues or breaking changes, and that takes a lot of time. Case in point: we had to wait almost three years until Excel 2016 to connect Power View reports to multidimensional cubes (only Tabular was supported before), although Analysis Services Multidimensional has much broader adoption than Tabular.

For consumers, rolling out a Microsoft BI solution has been problematic. Microsoft BI has been traditionally criticized for its deployment complexity and steep price tag. Although SharePoint Server offers much more than just data analytics, having a SharePoint server integrated with SQL Server has been a cost-prohibitive proposition for smaller organizations. As many of you would probably agree, SharePoint Server adds complexity and troubleshooting it isn't for the faint of heart. Power BI for Office 365 alleviated some of these concerns by shifting maintenance to become Microsoft's responsibility but many customers still find its "everything but the kitchen sink" approach too overwhelming and cost-prohibitive if all they want is the ability to deploy and share BI artifacts.

On the desktop, Excel wasn't originally designed as a BI tool, leaving the end user with the impression that BI was something Microsoft bolted on top of Excel. For example, navigating add-ins and learning how to navigate the cornucopia of features has been too much to ask from novice business users.

How does the new Power BI address these challenges?

Power BI embraces the following design tenets to address the previous pain points:

- Simplicity – Power BI was designed for BI from the ground up. As you'll see, Microsoft streamlined and simplified the user interface to ensure that your experience is intuitive and you aren't distracted by other non-BI features and menus.
- No dependencies to SharePoint and Office – Because it doesn't depend on SharePoint and Excel, Power BI can evolve independently. This doesn't mean that business users are now asked to forgo Excel. To the contrary, if you like Excel and prefer to create data models in Excel, you'll find that you can still deploy them to Power BI.
- Frequent updates – Microsoft promises weekly updates for Power BI Service and monthly updates for Power BI Desktop. This should allow Microsoft to stay at the forefront of the BI market.
- Always up to date – Because of its service-based nature, as a Power BI subscriber you're always on the latest and greatest version.
- Free – As you'll see in "1.2.4 Power BI Editions and Pricing" (later in this chapter), Power BI has the best business model: most of it it's free! Power BI Desktop and Power BI Mobile are free. Power BI Service is free and has a Power BI Pro subscription option that you could pay for, following a freemium model. Cost was the biggest hindrance of Power BI, and it's now been turned around completely. You can't beat free!

1.1.4 Power BI and the Microsoft Data Platform

Power BI isn't the only BI product that Microsoft provides. It's an integral part of the

Microsoft Data Platform that started in early 2004 with the powerful promise to bring “BI to the masses.” Microsoft subsequently extended the message to “BI to the masses, by the masses” to emphasize its commitment to democratize. Indeed, a few years after Microsoft got into the BI space, the BI landscape changed dramatically. Once a domain of cost-prohibitive and highly specialized tools, BI is now within the reach of every user and organization!

Understanding the Microsoft Data Platform

Figure 1.4 illustrates the most prominent services of the Microsoft Data Platform (and there are new cloud services added almost every month!)



Figure 1.4 The Microsoft Data Platform provides services and tools that address various data analytics and management needs on premises and in the cloud.

DEFINITION The Microsoft Data Platform is a multi-service offering that addresses the data capturing, transformation, and analytics needs to create modern BI solutions. It's powered by Microsoft SQL Server, SharePoint Server and Microsoft Office on premises and Microsoft Azure in the cloud.

Table 1.1 summarizes the various services of the Microsoft Data Platform and their purposes.

Table 1.1 The Microsoft Data Platform consists of many products and services, with the most prominent described below.

Category	Service	Audience	Purpose
Capture and manage	Relational	IT	Capture relational data in SQL Server, Analytics Platform System, Microsoft Access, and others.
	Non-relational	IT	Capture Big Data in Azure HDInsight Service and Microsoft HDInsight Server.
	NoSQL	IT	Capture NoSQL data in cloud structures, such as Azure Table Storage, DocumentDB, and others.
	Streaming	IT	Allow capturing of data streams from Internet of Things (IoT).
	Internal and External	IT/Business	Referring to cloud on your terms, allow connecting to both internal and external data, such as connecting Power BI to online services (Google Analytics, Salesforce, Dynamics CRM, and many others).
Transform and analyze	Orchestration	IT/Business	Create data orchestration workflows with SQL Server Integration Services (SSIS), Azure Data Factory, Power Query, Power BI Desktop, and Data Quality Services (DQS).
	Information management	IT/Business	Allow IT to establish rules for information management and data governance using SharePoint, Azure Data Catalog, and Office 365, as well as manage master data using SQL Server Master Data Services.
	Complex event processing	IT	Process data streams using SQL Server StreamInsight on premise and Azure Stream Analytics Service in the cloud.
	Modelling	IT/Business	Transform data in semantic structures with Analysis Services Multidimensional, Tabular, Power Pivot, and Power BI.
	Machine learning	IT/Business	Create data mining models in SQL Server Analysis Services, Excel data mining add-in, and Azure Machine Learning Service.
Visualize and decide	Applications	IT/Business	Analyze data with desktop applications, including Excel, Power BI Desktop, SSRS Designer, Report Builder, Datazen, Power View, Power Map.
	Reports	IT/Business	Create operational and ad hoc reports with SSRS, Excel, Datazen, and Power BI.
	Dashboards	IT/Business	Implement and share dashboards with Power BI, Excel, SharePoint, SSRS, and Datazen.
	Mobile	IT/Business	Support mobile devices

For more information about the Microsoft Data Platform, please visit <https://www.microsoft.com/en-us/server-cloud/solutions/business-intelligence>.

The role of Power BI in the Microsoft Data Platform

In Table 1.1, you can see that Power BI plays an important role in the Microsoft Data Platform by providing services for getting, transforming and visualizing your data. As far as data acquisition goes, it can connect to cloud and on-premises data sources so that you can import and relate data irrespective of its origin.

Capturing data is one thing but making dirty data suitable for analysis is quite another. However, you can use the data transformation capabilities of Power BI Desktop or Excel Power Query to cleanse and enrich your data. For example, someone might give you an Excel crosstab report. If you import the data as it is, you'll quickly find that you won't be able to relate it to the other tables in your data model. However, with a few clicks, you can un-pivot your data and remove unwanted rows. Moreover, the transformation steps are recorded so that you can repeat the same transformations later if you're given an updated

file.

The main purpose and strength of Power BI is visualizing data in reports and dashboards without requiring any special skills. You can explore and understand your data by having fun with it. To summarize insights from these reports, you can then compile a dashboard. Or, you can build the dashboard by asking natural questions. **Figure 1.5** shows a sample dashboard assembled from existing reports.



Figure 1.5 Power BI lets you assemble dashboards from existing reports or by asking natural questions.

1.2 Understanding the Power BI Products

Now that I've introduced you to Power BI and the Microsoft Data Platform, let's take a closer look at the Power BI building blocks. Don't worry if you don't immediately understand some of these technologies or if you find this section too technical. I'll clarify them throughout the rest of this chapter and the book. As I mentioned in section 1.1, Power BI is an umbrella name that unifies three products: Power BI Service, Power BI Mobile, and Power BI Desktop.

1.2.1 Understanding Power BI Service

At the heart of Power BI is the cloud-based business analytics service referred to *Power BI Service* or just *Power BI*. You use the service every time you utilize any of the powerbi.com features, such as connecting to online services, deploying and refreshing data models, viewing reports and dashboards, sharing content, or using Q&A (the natural language search feature). Next, I'll introduce you to some of Power BI Service's most prominent features.

Connect to any data source

The BI journey starts with connecting to data that could be a single file or multiple data sources. Power BI allows you to connect to virtually any accessible data source. Your self-service project can start small. If all you need is to analyze a single file, such as an Excel workbook, you might not need a data model. Instead, you can connect Power BI to your file, import its data, and start analyzing data immediately. However, if your data acquisition needs are more involved, such as when you have to relate data from multiple sources, you can use Power BI Desktop to implement a data model whose capabilities can be on par with professional data models and cubes!

Some data sources, such as Analysis Services models, support live connections. Because data isn't imported, live connections allow reports and dashboards to be always up to date. In the case when you have to import data, you can specify how often the data will be refreshed to keep it synchronized with changes in the original data source. For example, Martin might have decided to import data from the corporate data warehouse and deploy the model to Power BI. To keep the published model up to date, Martin can schedule the data model to refresh daily.

Content packs for online services

Continuing on data connectivity, chances are that your organization uses popular cloud services, such as Salesforce, Marketo, Dynamics CRM, Google Analytics, Zendesk, and others. Power BI content packs for online services allow business users to connect to such services and analyze their data without technical setup and data modeling. Content packs include a curated collection of dashboards and reports that continuously update with the latest data from these services. With a few clicks, you can connect to one of the supported online services and start analyzing data using prepackaged reports and dashboards. **Figure 1.6** shows a prepackaged dashboard I quickly implemented for analyzing website traffic

with Google Analytics.

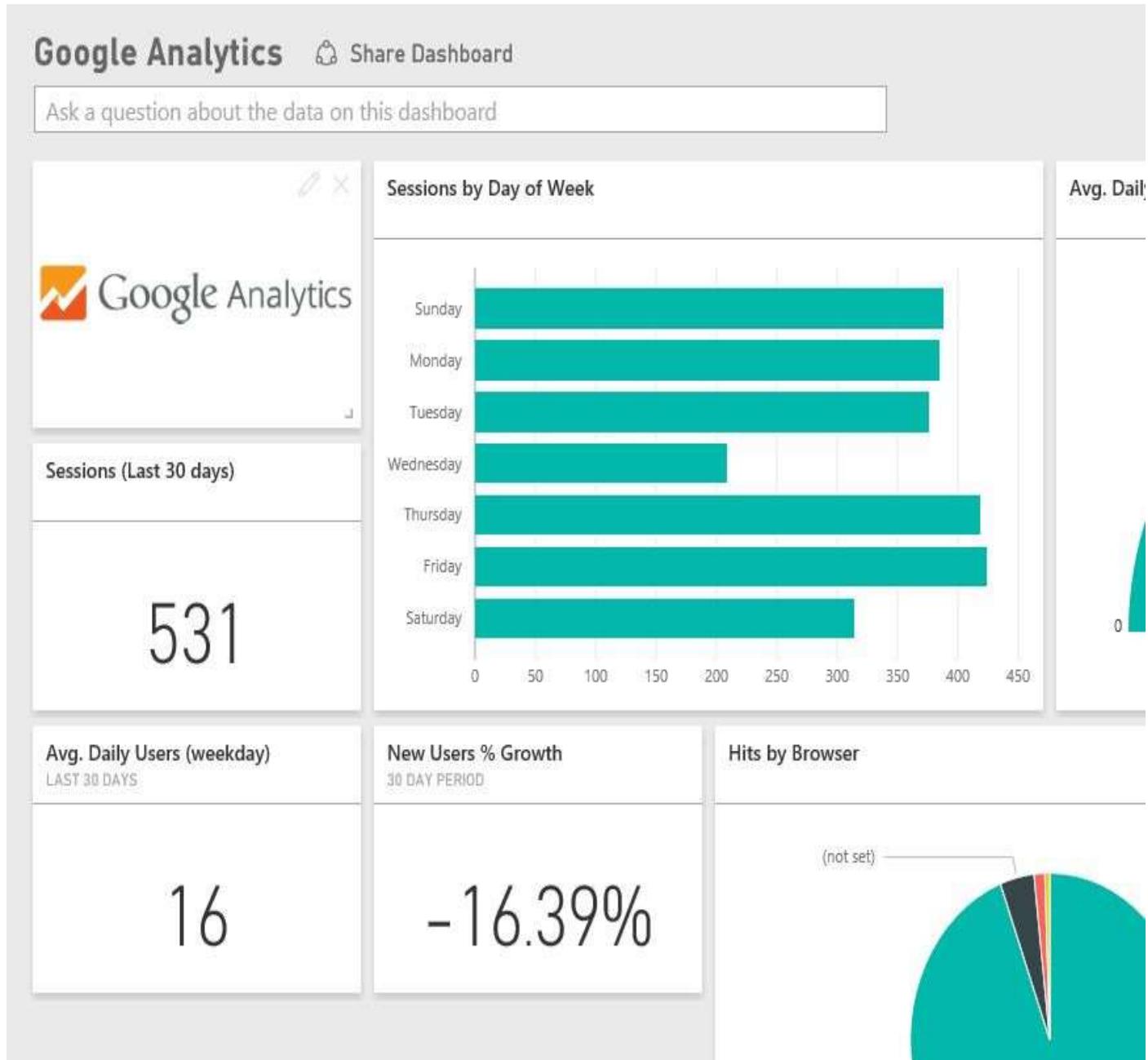


Figure 1.6 Content packs allow you to connect to online services and analyze data using prepackaged reports and dashboards.

Dashboards and reports

Collected data is meaningless without useful reports. Insightful dashboards and reports is what Power BI Service is all about. With a few clicks, a business user can create interactive reports. If you're familiar with Power View, you'll find that Power BI reports offer the same engaging experience for analyzing data by having fun with it!

For example, the report in **Figure 1.7** demonstrates one of these interactive features. In this case, the user selected Linda in the Bar Chart on the right. This action filtered the Column Chart on the left so that the user can see Linda's contribution to the overall sales.

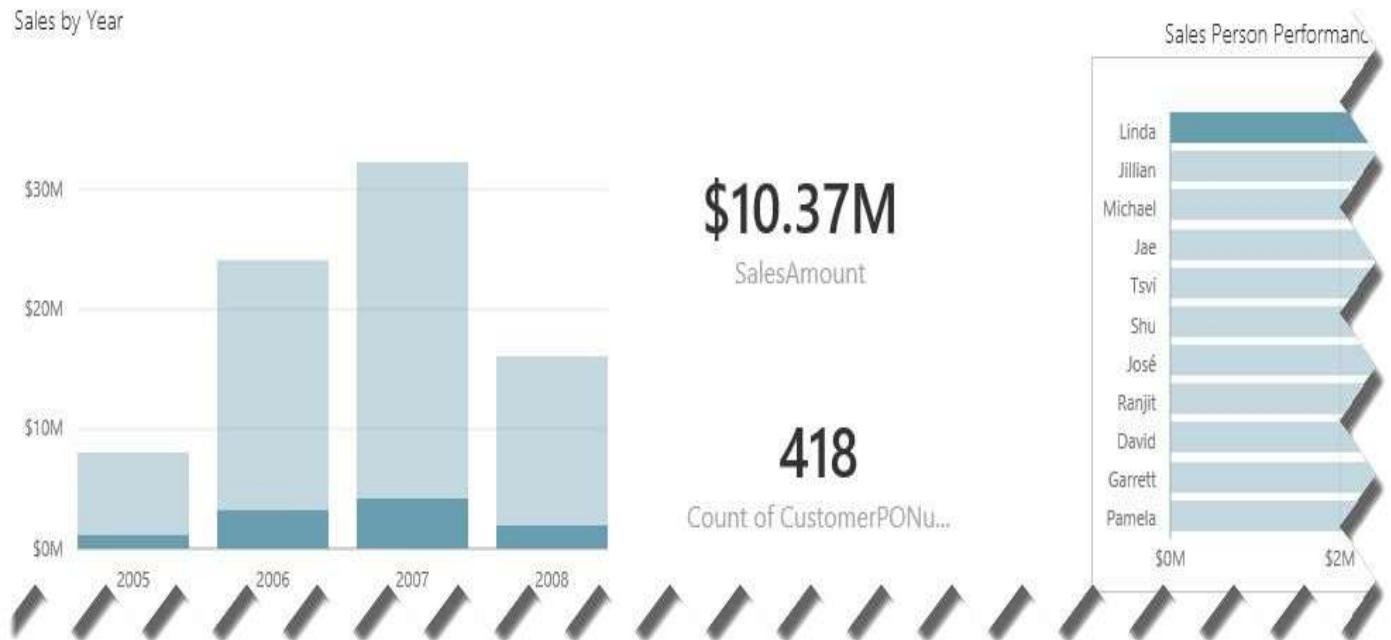


Figure 1.7 Interactive reports allow users to explore data in different ways.

Natural queries (Q&A)

Based on my experience, the feature that excites the users the most is Power BI natural queries or Q&A. End users are often overwhelmed when asked to create ad hoc reports from a data model. They don't know which fields to use and where to find them. The unfortunate "solution" by IT is to create new reports in an attempt to answer new questions. This might result in a ton of reports that quickly get replaced by new reports and are never used again. However, Power BI allows users to ask natural questions, such as "show sales amount by country in descending order by sales amount" (see **Figure 1.8**).

Reseller Sales

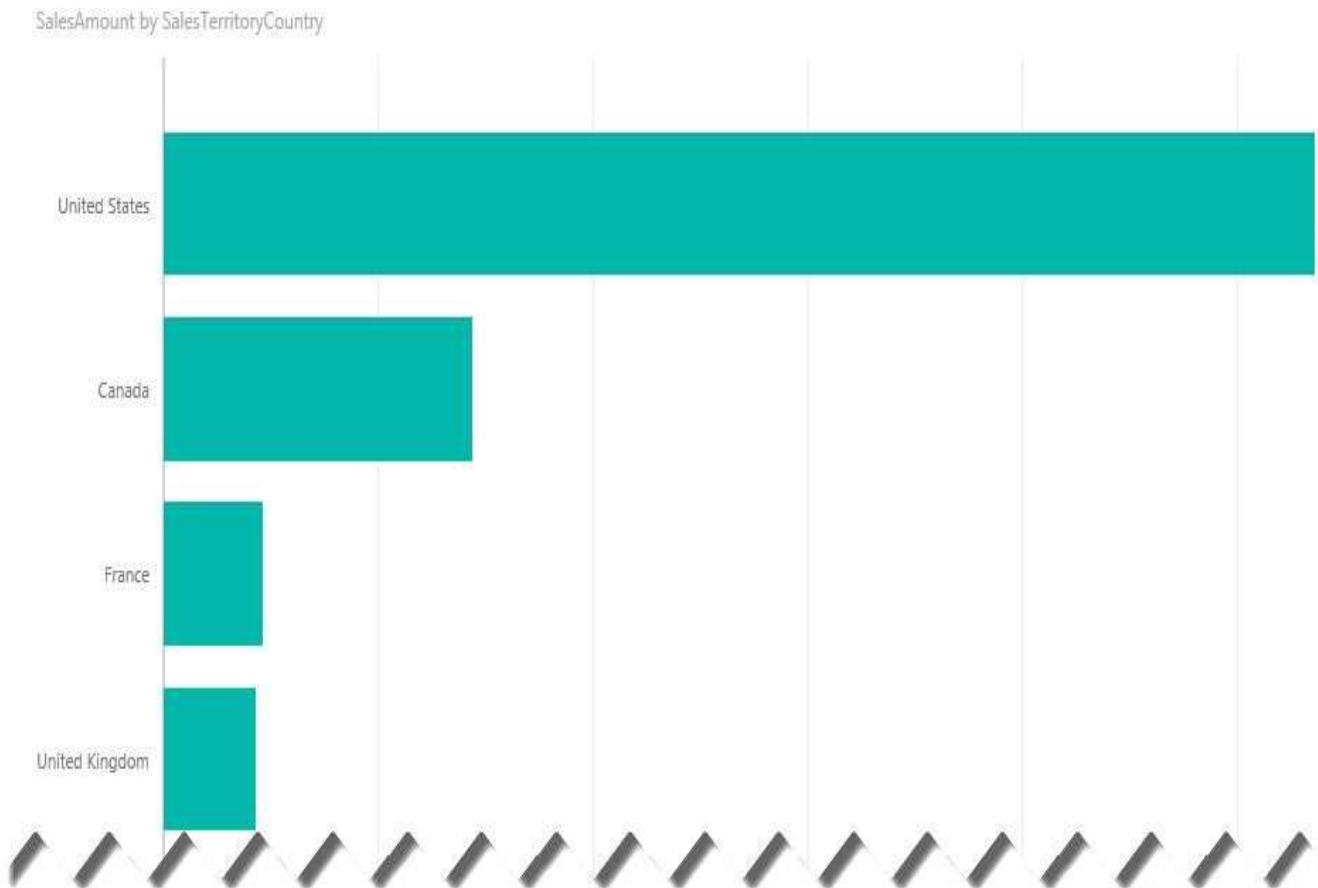


Figure 1.8 Q&A allows users to explore data by asking natural questions.

Not only can Power BI interpret natural questions, but it also chooses the best visualization! While in this case Q&A has decided to use a Bar Chart, it might have chosen a map if the question was phrased in a different way. And, the user can always change the visualization manually if the Power BI selection isn't adequate.

NOTE As of the time of writing this book, Q&A is supported only when data is imported into Power BI, such as when you get data from cloud services, Excel files, and Excel or Power BI Desktop data models. Q&A isn't currently supported with live connectivity, such as when you connect to on-premises Analysis Services data models. Q&A is also currently in English only.

Sharing and collaboration

Once you've created informative reports and dashboards, you might want to share them with your coworkers. Power BI supports several sharing options. Power BI Free allows you to share dashboards as read-only with your coworkers. Or you can use Power BI Pro workspaces to allow groups of people to have access to the same workspace content. For example, if Maya works in sales, she can create a Sales Department workspace and grant her coworkers access to the workspace. Then all content added to the Sales Department workspace will be shared among the group members.

Yet a third way to share content is to create an organizational content pack. Organizational content packs allow you to share content across teams or even with everyone from your organization. Users can discover and open content packs from the Power BI Content Gallery (see **Figure 1.9**). In this case, the user sees that someone has published a Reseller Sales content pack. The user can click the pack to see who published it, when it was published, and what it contains. Then the user can connect to the pack and access its content.

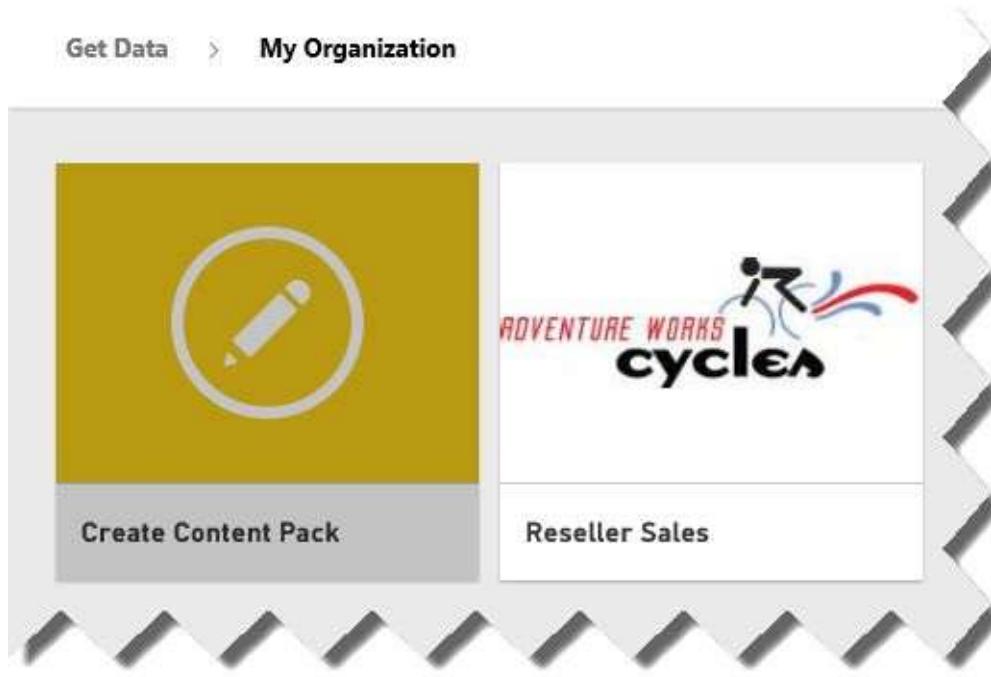


Figure 1.9 Users within your organization can use the Power BI Content Gallery to discover published organizational content packs.

1.2.2 Understanding the Power BI Service Architecture

The Power BI Service is hosted on Microsoft Azure cloud platform and it's currently deployed in 14 data centers. **Figure 1.10** shows a summarized view of the overall technical architecture that consists of two clusters: a Web Front End (WFE) cluster and a Back End cluster.

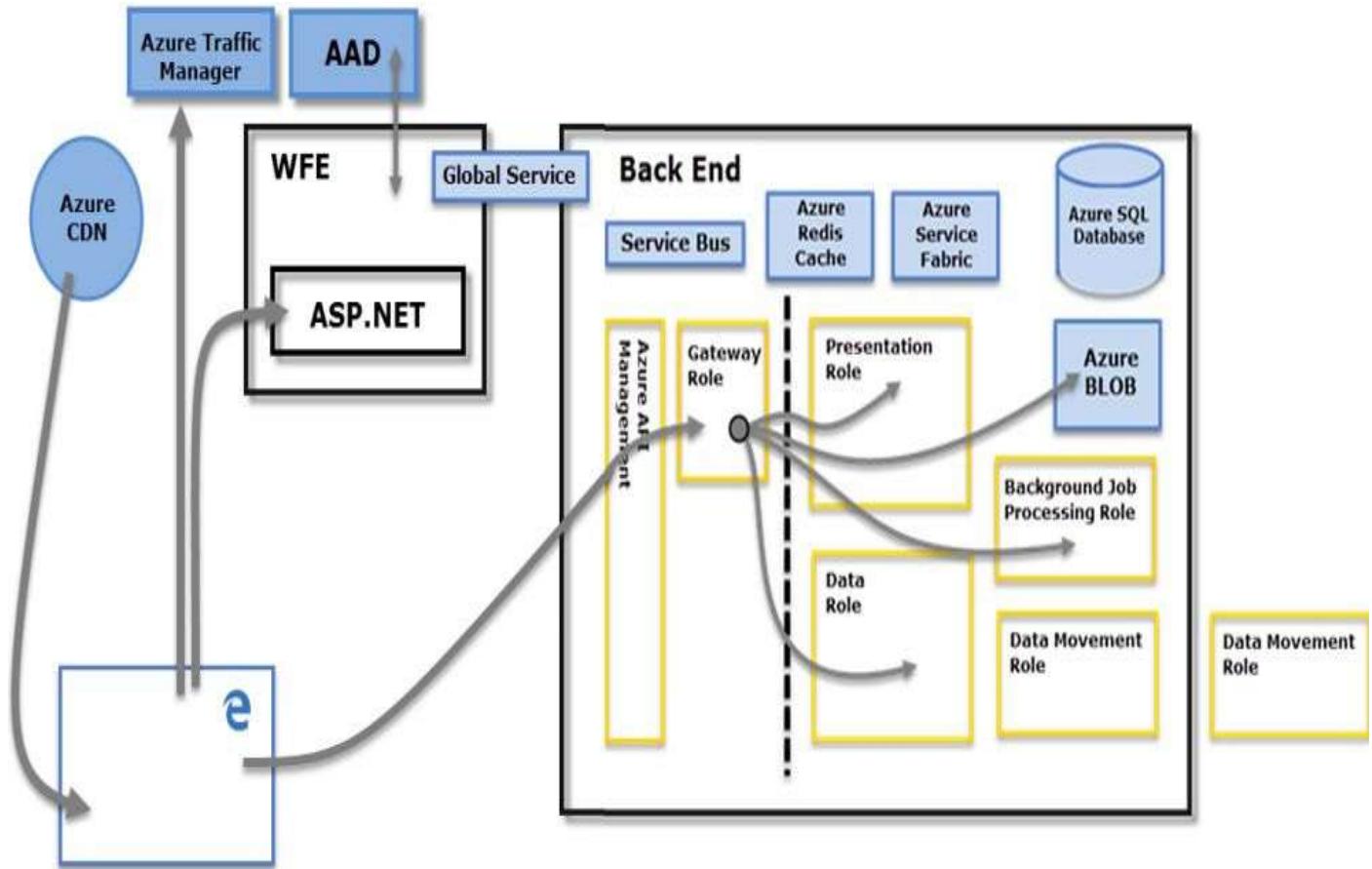


Figure 1.10 Power BI is powered by Microsoft Azure clusters.

Understanding the Web Front End (WFE) cluster

Microsoft has put a significant effort into building a scalable backend infrastructure consisting of various Azure services that handle data storage, security, load balancing, disaster recovery, logging, tracing, and so on. Although it's all implemented and managed by Microsoft (that's why we like the cloud), the following sections give you a high-level overview of these services to help you understand their value and Microsoft's decision to make Power BI a cloud service.

The WFE cluster manages connectivity and authentication. Power BI relies on Azure Active Directory (AAD) to manage account authentication and management. Power BI uses the Azure Traffic Manager (ATM) to direct user traffic to the nearest datacenter. Which data center is used is determined by the DNS record of the client attempting to connect. The DNS Service can communicate with the Azure Traffic Manager to find the nearest datacenter with a Power BI deployment.

TIP To find where your data is stored, log in to Power BI and click the Help (?) menu in the top-right corner, and then click “About Power BI”. Power BI shows a prompt that includes the Power BI version and the data center.

Power BI uses the Azure Content Delivery Network (CDN) to deliver the necessary static content and files to end users based on their geographical locale. The WFE cluster nearest to the user manages the user login and authentication, and provides an access token to the user once authentication is successful. The ASP.NET component within the WFE cluster parses the request to determine which organization the user belongs to, and then consults

the Power BI Global Service.

The Global Service is implemented as a single Azure Table that is shared among all worldwide WFE and Back End clusters. This service maps users and customer organizations to the datacenter that host their Power BI tenant. The WFE specifies to the browser which Back End cluster houses the organization's tenant. Once a user is authenticated, subsequent client interactions occur with the Back End cluster directly and the WFE cluster is not used.

Understanding the Back End cluster

The Back End cluster manages all actions the user does in Power BI Service, including visualizations, dashboards, datasets, reports, data storage, data connections, data refresh, and others. The Gateway Role acts as a gateway between user requests and the Power BI service. As you can see in the diagram, only the Gateway Role and Azure API Management (APIM) services are accessible from the public Internet. When an authenticated user connects to the Power BI Service, the connection and any request by the client is accepted and managed by the Gateway Role, which then interacts on the user's behalf with the rest of the Power BI Service. For example, when a client attempts to view a dashboard, the Gateway Role accepts that request, and then sends a request to the Presentation Role to retrieve the data needed by the browser to render the dashboard.

As far as data storage goes, Power BI uses two primary repositories for storing and managing data. Data that is uploaded from users is typically sent to Azure BLOB storage but all the metadata definitions (dashboards, reports, recent data sources, workspaces, organizational information, tenant information) are stored in Azure SQL Database.

The working horse of the Power BI service is Microsoft Analysis Services in Tabular mode, which has been architected to fulfill the role of a highly scalable data engine where many servers (nodes) participate in a multi-tenant, load-balanced farm. For example, when you import some data into Power BI, the actual data is stored in Azure BLOB storage but an in-memory Tabular database is created to service queries.

For BI pros who are familiar with Tabular, new components have been implemented so that Tabular is up to its new role. These components enable various cloud operations including tracing, logging, service-to-service operations, reporting loads and others. For example, Tabular has been enhanced to support the following features required by Power BI:

- Custom authentication – Because the traditional Windows NTLM authentication isn't appropriate in the cloud world, certificate-based authentication and custom security were added.
- Resource governance per database – Because databases from different customers (tenants) are hosted on the same server, Tabular ensures that any one database doesn't use all the resources.
- Diskless mode – For performance reasons, the data files aren't initially extracted to disk.
- Faster commit operations – This feature is used to isolate databases from each other.

When committing data, the server-level lock is now only taken for a fraction of time, although database-level commit locks are still taken and queries can still block commits and vice versa.

- Additional Dynamic Management Views (DMVs) – For better status discovery and load balancing.
- Data refresh – From the on-premises data using the Analysis Services connector.
- Additional features – Such as the new features added to Analysis Services in SQL Server 2016.

Data on your terms

The increasing number of security exploits in the recent years have made many organizations cautious about protecting their data and skeptical about the cloud. You might be curious to know what is uploaded to the Power BI service and how you can reduce your risk for unauthorized access to your data. In addition, you control where your data is stored. Although Power BI is a cloud service, this doesn't necessarily mean that your data must be uploaded to Power BI.

In a nutshell, you have two options to access your data. If the data source supports live connectivity, you can choose to leave the data where it is and only create reports and dashboards that connect live to your data. Currently, only a small subset of data sources supports live connectivity but that number is growing! Among them are Analysis Services, SQL Server (on premises and on Azure), Azure SQL Data Warehouse, and Hadoop Spark.

For example, if Elena has implemented an Analysis Services model and deployed to a server in her organization's data center, Maya can create reports and dashboards in Power BI Service by directly connecting to the model. In this case, the data remains on premises; only the report and dashboard definitions are hosted in Power BI. When Maya runs a report, the report generates a query and sends the query to the model. Then, the model returns the query results to Power BI. Finally, Power BI generates the report and sends the output to the user's web browser. Power BI always uses the Secure Sockets Layer (SSL) protocol to encrypt the traffic between the Internet browser and the Power BI Service so that sensitive data is protected.

NOTE Although in this case the data remains on premises, data summaries needed on reports and dashboards still travel from your data center to Power BI Service. This could be an issue for software vendors who have service level agreements prohibiting data movement. You can address such concerns by referring the customer to the Power BI Security document (<http://bit.ly/1SkEzTP>) and the accompanying Power BI Security whitepaper.

The second option is to upload and store the data in Power BI. For example, Martin might want to build a data model to analyze data from multiple data sources. Martin can use Power BI Desktop to import the data and analyze it locally. To share reports and allow other users to create reports, Martin decides to deploy the model to Power BI. In this case, the model and the imported data are uploaded to Power BI, where they're securely stored. To synchronize data changes, Martin can schedule a data refresh. Martin doesn't need to worry about security because data transfer between Power BI and on-premises data sources is secured through Azure Service Bus. Azure Service Bus creates a secure channel between Power BI Service and your computer. Because the secure connection happens

over HTTPS, there's no need to open a port in your company's firewall.

TIP If you want to avoid moving data to the cloud, one solution you can consider is implementing an Analysis Services model layered on top your data source. Not only does this approach keep the data local, but it also offers other important benefits, such as the ability to handle larger datasets (millions of rows), a single version of the truth by centralizing business calculations, row-level security, and others.

1.2.3 Understanding Power BI Mobile

Power BI Mobile is a set of native mobile applications for iOS (iPad and iPhone), Windows and Android devices. You can access the download links from <https://powerbi.microsoft.com/mobile>. Why do you need these applications? After all, thanks to Power BI HTML5 rendering, you can view Power BI reports and dashboards in your favorite Internet browser. However, the native applications offer features that go beyond just rendering. Although there are some implementation differences, this section covers some of the most compelling features (chapter 4 has more details).

Favorites

Suppose that, while viewing dashboard tiles on your iPad, you want to put your favorite tiles in one place. You can just tap a tile to mark it as a favorite. These tiles appear in a separate “Favorites” folder. The dashboard tiles displayed on your device are live snapshots of your data. To interact with a tile, just tap it!

Alerts

Do you want to be notified when your data changes beyond certain levels? Of course you do! You can set rules to be alerted when single number tiles in your dashboard exceed limits that you set. With data-driven alerts, you can gain insights and take action wherever you're located.

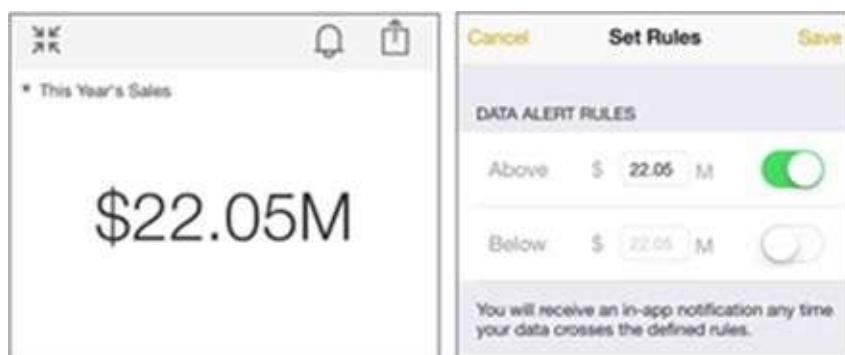


Figure 1.11 Power BI Mobile data alerts allow you to be notified when important data changes happen.

For example, **Figure 1.11** shows that I've enabled an iPhone data alert with a data rule above \$22.05 million. When sales exceed this threshold, I'll get an in-app notification.

Annotations

Annotations allow you to add comments (lines, text, and stamps) to dashboard tiles (see **Figure 1.12**). Then you can mail a screen snapshot to recipients, such as to your manager.



Figure 1.12 Annotations allow you to add comments to tiles and then send screenshots to your coworkers.

Sharing

Similar to Power BI simple sharing, you can use mobile device to share a dashboard by inviting coworkers to access the dashboard. Dashboards shared by mail are read-only, meaning that the people you shared with can only view the dashboard without making changes.

1.2.4 Understanding Power BI Desktop

Oftentimes, data analytics go beyond a single dataset. To meet more advanced needs, business analysts create data models, such as to relate data from multiple data sources and then implement business calculations. The Power BI premium design tool for implementing such models is Power BI Desktop.

Installing Power BI Desktop

Power BI Desktop is a freely available Windows application for implementing self-service data models and reports. You can download it for free from <https://powerbi.microsoft.com/en-us/desktop> or from the Downloads menu in Power BI Service. Power BI Desktop is available as 32-bit and 64-bit Windows installations. The download page determines what version of Windows you have (32-bit or 64-bit) and downloads the appropriate executable.

Nowadays, you can't buy a 32-bit computer (not easily, anyway). However, even if you have a 64-bit computer and 64-bit Windows OS, you can still install 32-bit applications. The problem is that 32-bit applications are limited to 2 GB of memory. By contrast, 64-bit computing enables applications to use more than 2 GB of memory. This is especially useful for in-memory databases that import data, such as xVelocity (remember that

xVelocity is the storage engine of Power BI Service and Power BI Desktop). In general, if you have a 64-bit version of Windows, you should install the 64-bit version of any software if a 64-bit version is available. Therefore, the 64-bit version of Power BI Desktop is a better choice. However, although your model on the desktop can grow and grow until it exhausts all the memory, remember that Power BI Service won't let you upload a file that is larger than 250 MB (this limit will probably increase) so this keep in mind as well if you plan to publish the model.

NOTE Readers familiar with Excel data modeling might remember that the Office setup installs the 32-bit version of Office by default and getting IT to install the 64-bit version has been a struggle. The Office setup favors the 32-bit version in case you use 32-bit add-ins. Because Power BI Desktop doesn't depend on Office, you can go ahead and install the 64-bit version even if you have the 32-bit version of Office installed.

Understanding Power BI Desktop features

Before Power BI, data analysts could implement data models in Excel. This option is still available, and you can upload your Excel data models to Power BI. However, to overcome the challenges associated with Excel data modeling (see section 1.1.3), Microsoft introduced Power BI Desktop.

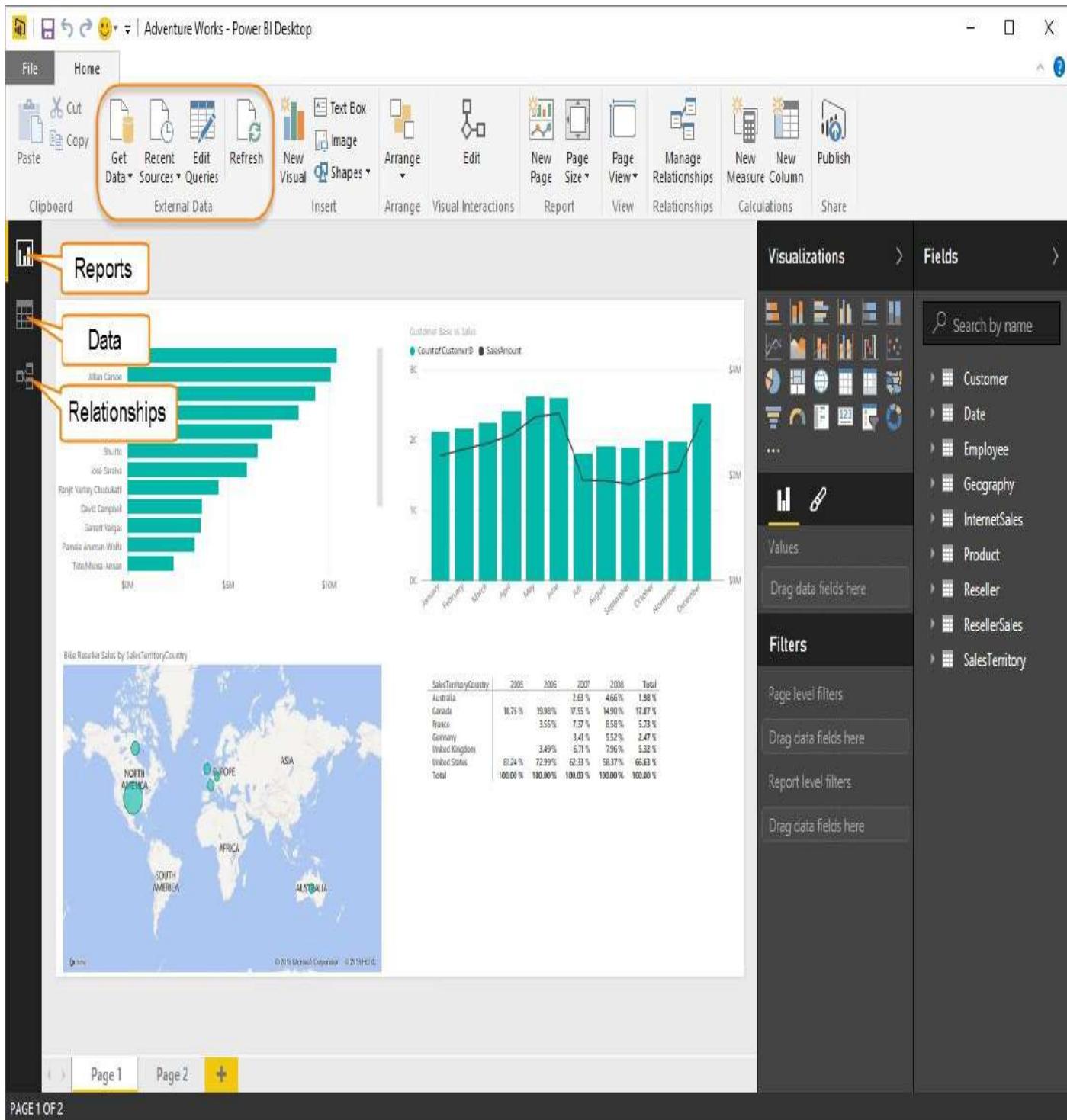


Figure 1.13 Power BI Desktop unifies the capabilities of Power Pivot, Power Query, and Power View.

Think of Power BI Desktop as the unification of Power Pivot, Power Query, and Power View. Previously available as Excel add-ins, these tools now blend into a single flow. No more guessing which add-in to use and where to find it! At a high level, the data modelling experience in Power BI Desktop now encompasses the following steps (see **Figure 1.13**):

1. Former Power Query – Use the Get Data button in the ribbon to connect to and transform the data. This process is similar to using Excel Power Query. When you import a dataset, Power BI Desktop creates a table and loads the data. The data is stored in a highly compressed format and loaded in memory to allow you to slice and dice the data without

sacrificing performance. However, unlike Excel, Power BI Desktop allows you to connect directly to a limited number of fast databases, such as Analysis Services and Azure SQL Data Warehouse, where it doesn't make sense to import the data.

2. Former Power Pivot – View and make changes to the data model using the Data and Relationships tabs in the left navigation bar. This is the former Power Pivot part.
3. Former Power View – Create interactive reports using the Reports tab on the left, as you can do using Power View in Excel (version 2013 or higher).

NOTE Some data sources, such as Analysis Services, support live connectivity. Once you connect to a live data source, you can jump directly to step 3 above and start creating reports because there are no queries to edit and models to design. In this case, Power BI Desktop acts as a presentation layer that's directly connected to the data source.

Comparing design environments

Because there are many Power Pivot models out there, Power BI allows data analysts to deploy Excel files with embedded data models to Power BI Service and view the included pivot reports and Power View reports online. Analysts can now choose which modeling tool to use:

- Microsoft Excel – Use this option if you prefer to work with Excel and you're familiar with the data modelling features delivered by Power Pivot, Power Query, Power View and Power Map.
- Power BI Desktop – Use this free option if you prefer a simplified tool that's specifically designed for data analytics and that's updated more frequently than Excel.

Table 1.2 compares these two design options side by side to help you choose a design environment.

Table 1.2 This table compares the data modelling capabilities of Microsoft Excel and Power BI Desktop.

Feature	Excel	Power BI Desktop
Data import	Excel native import, Power Pivot, Power Query	Query Editor
Data transformation	Power Query	Query Editor
Modeling	Power Pivot	Data and Relationships tabs
Reporting	Excel pivot reports, Power View, Power Map	Power View-based reports
Update frequency	Office releases or more often with Office 365 click-to-run	Monthly
Server deployment	SharePoint Server and Power BI	Power BI
Power BI deployment	Deployed as Excel (*.xlsx) file	Deployed as Power BI Desktop (pbix) file
Convert models	Can't import Power BI Desktop models	Can import Excel data model
Upgrade to Tabular	Yes	No
Cost	Excel license	Free

Let's go quickly through the list. While Excel supports at least three ways to import data, many users might struggle in understanding how they compare. By contrast, Power BI Desktop has only one data import option which is the equivalent of Power Query in Excel. Similarly, Excel has various menus in different places that relate to data modelling. By contrast, if you use Power BI Desktop to import data, your data modelling experience is much more simplified.

Excel allows you to create pivot, Power View, and Power Map reports from data models. At this point, Power BI Desktop supports only the equivalent of Excel Power View reports and some of the Power Map features, although it regularly adds more visualizations and features.

The Excel update frequency depends on how it's installed. If you install it from a setup disk (MSI installation), you need to wait for the next version to get new features. Office 365 supports subscription-based Microsoft Office (click-to-run installation) which delivers new features as they get available. If you take the Power BI Desktop path, you'll need to download and install updates as they become available. Microsoft announced that Power BI Desktop will get monthly updates so you'll get new features faster.

As far as deployment goes, you can deploy Excel files with data models to Power BI, just like you can deploy them to SharePoint Server or Office 365. Power BI Desktop models (files with extension *.pbix) can be deployed to Power BI only. Behind the scenes, both Excel and Power BI Desktop use the in-memory xVelocity engine to compress and store imported data.

NOTE At the PASS Summit in October 2015, Microsoft announced that a future SQL Server Reporting Services edition (past SQL Server 2016) will support Power BI Desktop files. Currently, only Power BI Service supports online rendering of Power BI Desktop models.

As of the time of writing, Power BI Desktop supports importing Power Pivot models from Excel to allow you to migrate models from Excel to Power BI Desktop. Excel doesn't support importing Power BI Desktop models yet so you can't convert your Power BI Desktop files to Excel data models. A BI pro can migrate Excel data models to Tabular models when organizational features, such as scalability and security, are desirable. Currently, Tabular can't import Power BI Desktop models because Power BI adds features constantly, which could put it ahead of the Tabular box features.

1.2.5 Power BI Editions and Pricing

Power BI editions and pricing is simple and well documented at <https://powerbi.microsoft.com/pricing>. This page also includes a table showing which features are available by which edition. As it stands, Power BI is available in two editions: Power BI (Free) and Power BI Pro.

NOTE These editions apply to Power BI Service (powerbi.com) only. Power BI Desktop and Power BI Mobile are freely available and they don't require a Power BI Service subscription.

Understanding the Power BI Free edition

The Power BI edition is a free offering but it has the following limitations:

- Data capacity – If you import data, you are limited to one GB of data storage.
- Data refresh – If you connect to online services, such as Google Analytics or Salesforce, you can't refresh data more frequently than daily.
- Data streaming – If Power BI connects to Azure Stream Analytics (<http://azure.microsoft.com/en-us/services/stream-analytics>) for real-time dashboards, streaming is capped at 10,000 rows per hour.

- Data connectivity – You can't refresh data from on-premises data sources. Let's say you've imported data from a corporate data warehouse into Power BI Desktop and deployed the model to Power BI. While you can analyze the imported data, this limitation prevents you from scheduling an automated refresh schedule to update the model. The free edition also doesn't include connecting to data sources that support live connections, such as Analysis Services, Spark on Azure HDInsight, Azure SQL Data Warehouse, and SQL Server.
- Content sharing and collaboration – Only simple sharing by mail is supported.

Despite these limitations, the free edition is packed with features. For example, Power BI Free will be appealing to a business user who's interested in Power BI content packs to analyze online data. This edition would allow a data analyst to create and publish sophisticated data models, refresh imported data, and share dashboards with a limited number of people.

Understanding the Power BI Pro edition

This paid edition currently charges currently \$9.99 per user per month and offers the following extra features:

- Data capacity – It increases the data storage quota to 10 GB per user.
- Data refresh – It supports hourly data refreshes.
- Data streaming – It supports streaming to one million rows per second.
- Data connectivity – No data connectivity limitations.
- Content sharing and collaboration – Besides simple sharing, Power BI Pro also supports workspaces and organizational content packs.

Currently, Microsoft treats Power BI for Office 365 user licenses the same as Power BI Pro. So any user who has a Power BI for Office 365 business plan and signs up for Power BI Service, will automatically get a Power BI Pro license. If this isn't the desired outcome, the Office 365 administrator can use the Office 365 portal to remove the Power BI license from that user. Then the user will be downgraded to the free Power BI edition. This is explained in more detail in the "Power BI Transition" document at <http://www.microsoft.com/en-us/powerBI/licensing.aspx>.

NOTE Not sure if the Power BI Pro edition is right for you? You can evaluate it for free for 60 days. To start the trial period, log in the Power BI portal, click the Settings menu in the top right corner, and then click "Manage Personal Storage". Then click the "Try Pro for free" link.

1.3 Power BI and You

Now that I've introduced you to Power BI and its building blocks, let's see what Power BI means for you. As you'll see, Power BI has plenty to offer to anyone interested in data analytics, irrespective of whether you're a content producer or consumer, as shown in **Figure 1.14**.

By the way, the book content follows the same organization so that you can quickly find the relevant information depending on what type of user you are. For example, if you're a business user, the first part of the book is for you and it has three chapters for the first three features shown in the "For business users" section in the diagram.

1.3.1 Power BI for Business Users

To clarify the term, a business user is someone in your organization who is mostly interested in consuming BI artifacts, such as reports and dashboards. This group of users typically includes executives, managers, business strategists, and regular information workers. To get better and faster insights, some business users often become basic content producers, such as when they create reports to analyze simple datasets or data from online services.



Figure 1.14 Power BI supports the BI needs of business users, analysts, pros, and developers.

For example, Maya is a manager in the Adventure Works Sales & Marketing department. She doesn't have skills to create sophisticated data models and business calculations. However, she's interested in monitoring the Adventure Works sales by using reports and

dashboards produced by other users. She's also a BI content producer because she has to create reports for analyzing data in Excel spreadsheets, website traffic, and customer relationship management (CRM) data.

Connect to your data without creating models

Thanks to the Power BI content packs, Maya can connect to popular cloud services, such as Google Analytics and Dynamics CRM, and get instant dashboards. She can also benefit from prepackaged datasets, reports, and dashboards, created jointly by Software as a Service (SaaS) partners and Microsoft.

For example, the Dynamics CRM connector provides an easy access to analyze data from the cloud-hosted version of Dynamics CRM. This connector uses the Dynamics CRM OData feed to auto-create a descriptive model that contains the most important entities, such as Accounts, Activities, Opportunities, Products, Leads, and others. Pre-built dashboards and reports, such as the one shown in **Figure 1.15**, provide immediate insights and can be further customized.

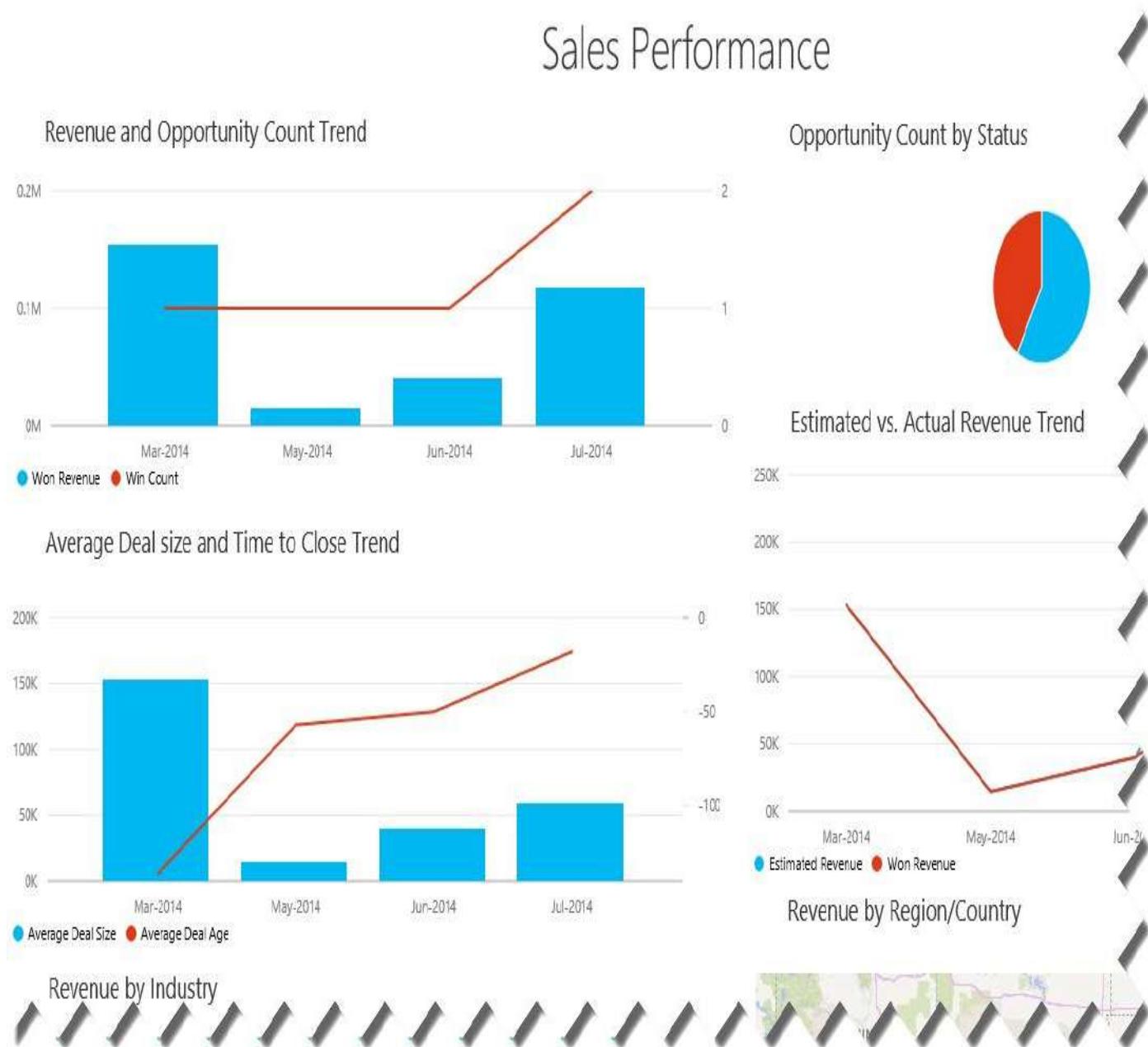


Figure 1.15 The Dynamics CRM content pack includes prepackaged reports and dashboards.

Similarly, if Adventure Works uses Salesforce as a CRM tool, Power BI has a connector to allow Maya to connect to Salesforce in a minute. Power BI content packs support data refresh, such as to allow Maya to refresh the CRM data daily.

Explore data

Power BI can also help Maya analyze simple datasets without data modeling. For example, if Maya receives an Excel file with some sales data, she can import the data into Power BI and create ad hoc reports and dashboards with a few mouse clicks. She can easily share dashboards with coworkers. For example, Maya can navigate to the Power BI portal, select a dashboard, and then click the Share button next to the dashboard name (see **Figure 1.16**).

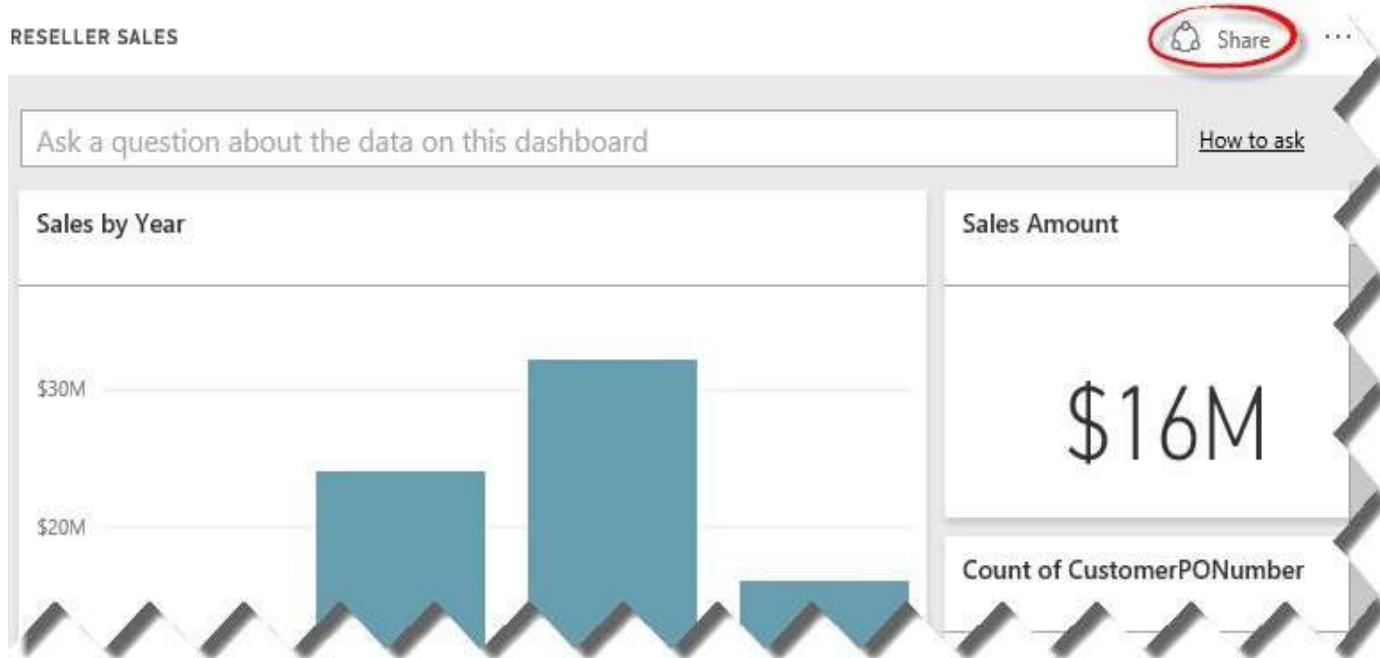


Figure 1.16 Business users can easily share dashboards with coworkers using the Power BI portal or Power BI Mobile.

Mobile applications

Some business users, especially managers, executives, and sales people, would need access to BI reports on the go. This type of users would benefit from the Power BI Mobile native applications for iPad, iPhone, Android, and Windows. As I explained in section 1.2.2, Power BI Mobile allows users to not only to view Power BI reports and dashboards, but to also receive alerts about important data changes, as well to share and annotate dashboards.

For example, while Maya travels on business trips, she needs access to her reports and dashboards. Thanks to the cloud-based nature of Power BI, she can access them anywhere she has an Internet connection. Depending on what type of mobile device she uses, she can also install a Power BI native application so she can benefit from additional useful features, such as favorites, annotations, and content sharing.

1.3.2 Power BI for Data Analysts

A data analyst or BI analyst is a power user who has the skills and desire to create self-service data models. A data analyst typically prefers to work directly with the raw data, such as to relate corporate sales data coming from the corporate data warehouse with external data, such as economic data, demographics data, weather data, or any other data purchased from a third party provider.

For example, Martin is a BI analyst with Adventure Works. Martin has experience in analyzing data with Excel and Microsoft Access. To offload effort from IT, Martin wants to create his own data model by combining data from multiple data sources.

Import and mash up data from virtually everywhere

As I mentioned previously, to create data models, Martin can use Microsoft Excel and/or Power BI Desktop, which combines the best of Power Query, Power Pivot and Power View in a single and simplified design environment. If he has prior Power Pivot experience, Martin will find Power BI Desktop easier to use and he might decide to switch to it in order stay on top of the latest Power BI features. Irrespective of the design environment chosen, Martin can use either Excel or Power BI Desktop to connect to any accessible data source, such as a relational database, file, cloud-based services, SharePoint lists, Exchange servers, and many more.

Figure 1.17 shows the supported data sources in Power BI Desktop and Excel. Microsoft regularly adds new data sources and content packs. Once Martin deploys the model to Power BI, he can schedule a data refresh to keep the imported data up to date.

 Excel	 Microsoft Azure Marketplace	 R Script (Beta)
 CSV	 Microsoft Azure HDInsight	 SAP HANA Database (Beta)
 XML	 Microsoft Azure Blob Storage	 appFigures (Beta)
 Text	 Microsoft Azure Table Storage	 GitHub (Beta)
 Folder	 Web	 MailChimp (Beta)
 SQL Server Database	 SharePoint List	 QuickBooks Online (Beta)
 Access Database	 OData Feed	 SweetIQ (Beta)
 SQL Server Analysis Services Database	 Hadoop File (HDFS)	 Twilio (Beta)
 Oracle Database	 Active Directory	 Zendesk (Beta)
 IBM DB2 Database	 Microsoft Exchange	 Marketo (Beta)
 MySQL Database	 Dynamics CRM Online	 Azure HDInsight Spark (Beta)
 PostgreSQL Database	 Facebook	 Spark (Beta)
 Sybase Database	 Google Analytics	 Microsoft Azure DocumentDB (Beta)
 Teradata Database	 Salesforce Objects	 Microsoft Azure Data Lake Store (Beta)
 Microsoft Azure SQL Database	 Salesforce Reports	 Blank Query
 Microsoft Azure SQL Data Warehouse	 ODBC	

Figure 1.17 Power BI self-service data models can connect to a plethora of data sources.

Cleanse, transform, and shape data

Data is rarely cleaned. A unique feature of Power BI Desktop is cleansing and transforming data. Inheriting these features from Power Query, Power BI Desktop allows a data analyst to apply popular transformation tasks that saves tremendous data cleansing effort, such as replacing values, un-pivoting data, combining datasets and columns, and many more tasks.

For example, Martin may need to import an Excel financial report that was given to him in a crosstab format where data is pivoted by months on columns. Martin realizes that if he imports the data as it is, he won't be able to relate it to a date table that he has in the model. However, with a couple of mouse clicks, Martin can use a Power BI Desktop query to un-pivot months from columns to rows. And once Martin gets a new file, the query will apply the same transformations so that Martin doesn't have to go through the steps again.

Implement self-service data models

Once the data is imported, Martin can relate the datasets to analyze the data from different angles by relating multiple datasets (see **Figure 1.1** again). No matter which source the data came from, Martin can use Power BI Desktop or Excel to relate tables and create data models whose features are on par with professional models. Power BI supports relationships natively with one-to-many and many-to-many cardinality so Martin can model complex requirements, such as analyzing financial balances of joint bank accounts.

Create business calculations

Martin can also implement sophisticated business calculations, such as time calculations, weighted averages, variances, period growth, and so on. To do so, Martin will use the Data Analysis Expression (DAX) language and Excel-like formulas, such as the formula shown in **Figure 1.18**. This formula calculates the year-to-date (YTD) sales amount. As you can see, Power BI Desktop supports IntelliSense and color coding to help you with the formula syntax. IntelliSense offers suggestions as you type.

The screenshot shows the Power BI Desktop interface with the Modeling ribbon tab selected. The formula bar at the top contains the DAX formula: `SalesAmountYTD = TOTALYTD(Sum([SalesAmount]), 'Date'[Date])`. A tooltip for the `TOTALYTD` function is displayed, stating: "Evaluates the specified expression over the interval which begins on the first day of the year and ends with the last date in the specified date column after applying specified filters." Below the formula bar, a data grid is visible with columns for ProductKey, OrderDateKey, DueDate, SalesAmount, and CurrencyKey.

Figure 1.18 Business calculations are implemented in DAX.

Once the model is created, the analyst can visualize and explore the data with interactive reports. If you come from using Excel Power Pivot and would like to give Power BI Desktop a try, you'll find that not only does it simplify the design experience, but it also supports new visualizations, including Funnel and Combo Charts, Treemap, Filled Map, and Gauge visualizations, as shown in **Figure 1.19**.

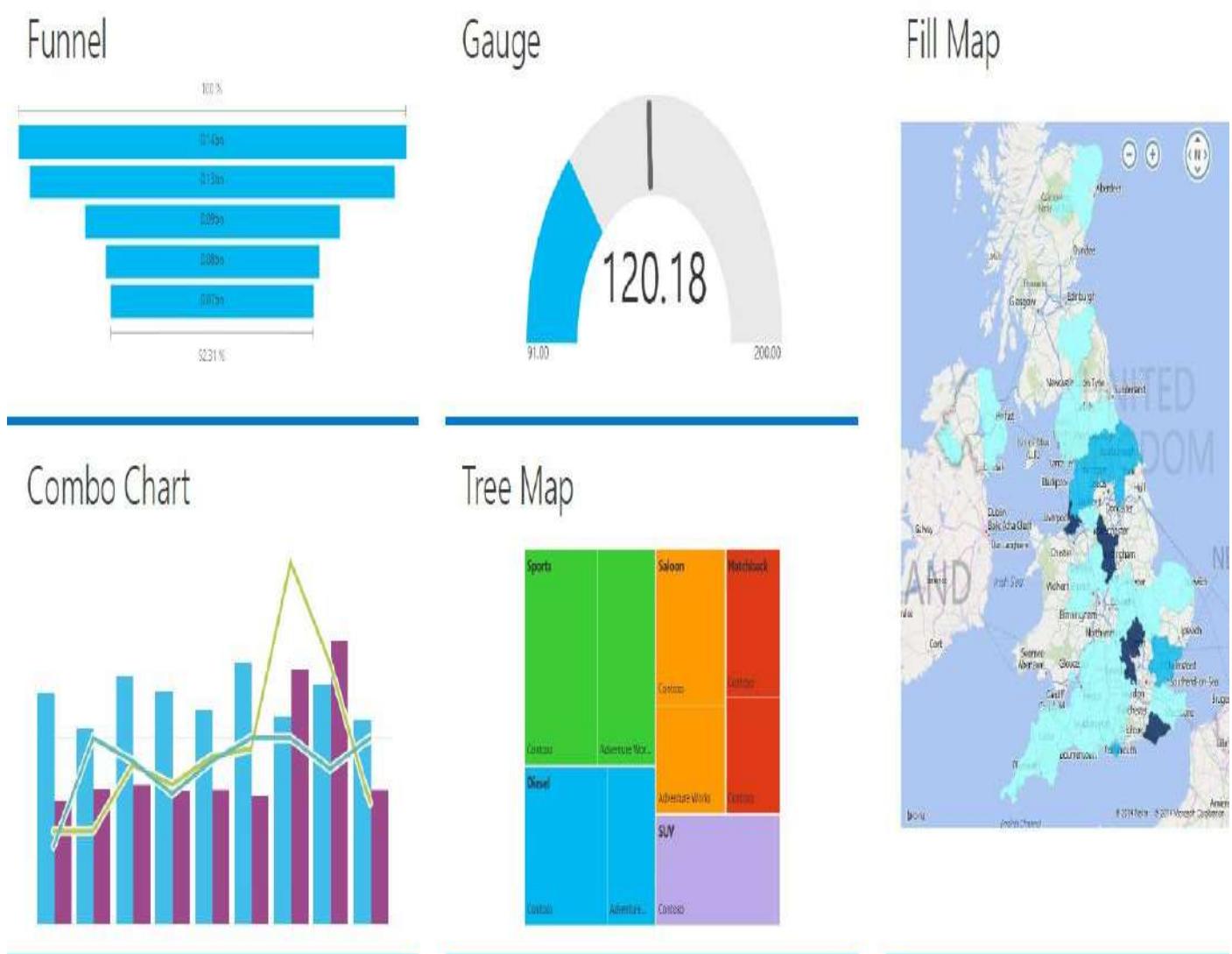


Figure 1.19 Power BI Desktop adds new visualizations.

And when the Microsoft-provided visualizations aren't enough, Martin can download a custom visual contributed by Microsoft and the Power BI community. To do this, Martin will go to the Power BI visuals gallery (<http://visuals.powerbi.com>) and download a visual file with the *.pbviz file extension. Then, Martin can import the visual into Power BI Service or Power BI Desktop and start using it immediately!

Once Martin is done with the report in Power BI Desktop, he can publish the model and reports to Power BI, so that he can share insights with other users. If they have permissions, his coworkers can view reports, gain more insights with natural query (Q&A) questions, and create dashboards.

1.3.3 Power BI for Pros

BI pros and IT pros have much to gain from Power BI. BI pros are typically tasked to create the backend infrastructure required to support organizational BI initiatives, such as data marts, data warehouses, cubes, ETL packages, operational reports, and dashboards. IT pros are also concerned with setting up and maintaining the necessary environment that facilitates self-service and organizational BI, such as providing access to data, managing security, data governance, and other services.

In a department or smaller organization, a single person typically fulfills both BI and IT pro tasks. For example, Elena has developed an Analysis Services model on top of the corporate data warehouse. She needs to ensure that business users can gain insights from the model without compromising security.

Enable team BI

Once she provided connectivity to the on-premises model, Elena must establish a trustworthy environment needed to facilitate content sharing and collaboration. To do so, she can use Power BI workspaces (a Power BI Pro feature). As a first step, Elena would set up groups and add members to these groups. Then Elena can create workspaces for the organizational units interested in analyzing the SSAS model. For example, if the Sales department needs access to the organizational model, Elena can set up a Sales Department group. Next, she can create a Sales Department workspace and grant the group access to it. Finally, she can deploy to the workspace her sales-related dashboards and reports that connect to the model.

If Elena needs to distribute BI artifacts to a wider audience, such as the entire organization, she can create an organizational content pack and publish it to the Power BI Content Gallery. Then her coworkers can search, discover, and use the content pack. Furthermore, they can create their own copies of the original dashboard and reports so that they can make changes without affecting the original copy.

Implementing BI solutions

Based on my experience, most organizations could benefit from what I refer to as a classic BI architecture that includes a data warehouse and semantic model (Analysis Service Multidimensional or Tabular mode) layered on top of the data warehouse. I'll discuss the benefits of this architecture in Part 3 of this book. If you already have or are planning such a solution, you can use Power BI as a presentation layer. This works because Power BI can connect to the on-premises Analysis Services, as shown in **Figure 1.20**.

Live dashboards and exploration

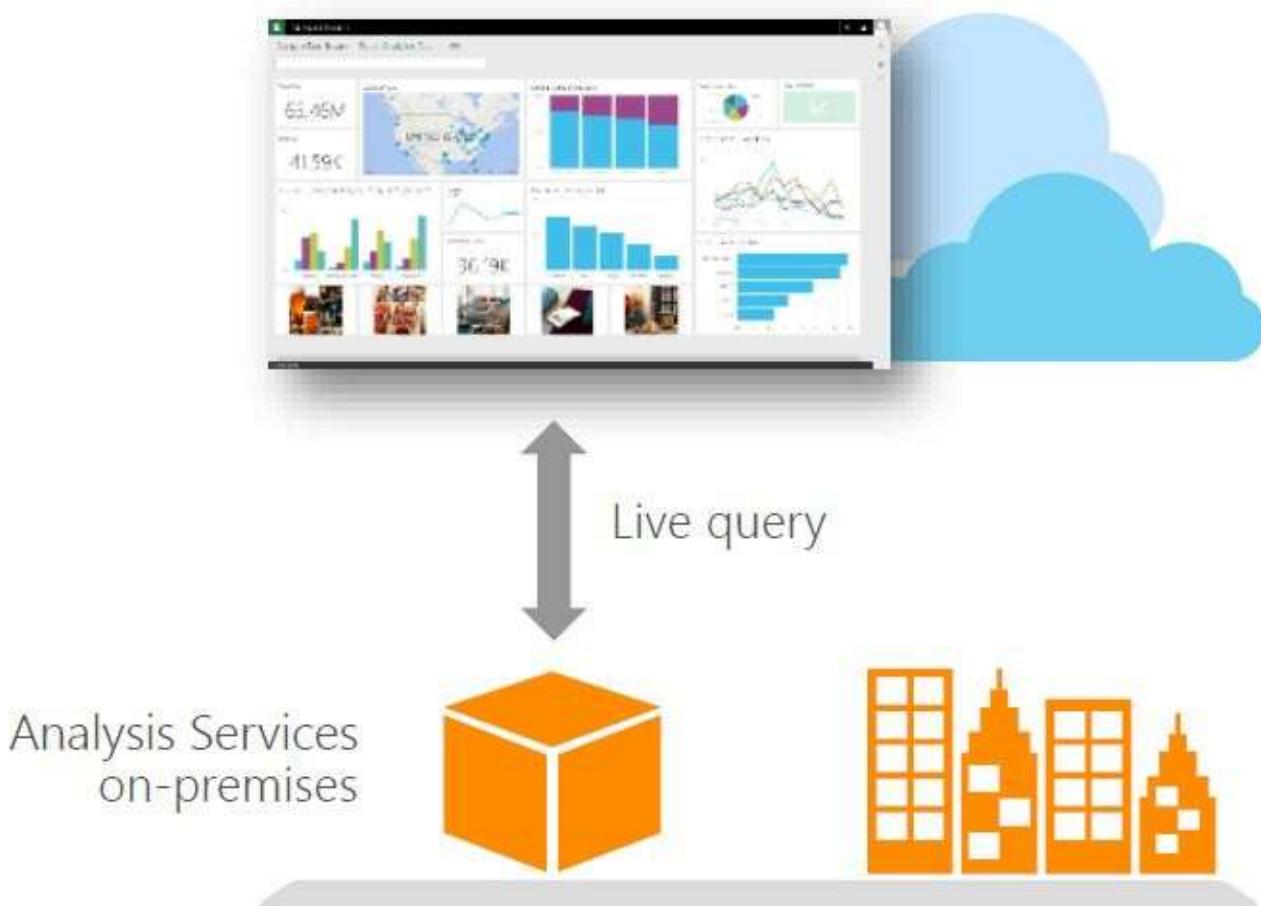


Figure 1.20 Power BI can directly connect to on-premises Analysis Services models.

So that Power BI can connect to on-premises SSAS models, Elena needs to download and install a component called Analysis Services Connector to an on-premises computer that can connect to the model. Elena can also set up the Power BI Enterprise Gateway to centralize management and access to on-premises data sources. Then Elena can implement reports and dashboards that connect live to Analysis Services and deploy them to Power BI. When users open a report, the report will generate a query and send it to the on-premises model via the Analysis Services connector. Now you have a hybrid solution where data stays on premises but reports are hosted in Power BI.

If you're concerned about the performance of this architecture, you should know that Power BI only sends queries to Analysis Services so there isn't much overhead on the trip from Power BI to SSAS. Typically, BI reports and dashboards summarize data. Therefore, the size of the datasets that travel back from SSAS to Power BI probably won't be very large either. Of course, the speed of the connection between Power BI and the data center where the model resides will affect the duration of the round trip.

Another increasingly popular scenario that Power BI enables is real-time BI. You've probably heard about Internet of Things (IoT) which refers to an environment of many connected devices, such as barcode readers, sensors, or cell phones, that transfer data over a network without requiring human-to-human or human-to-computer interaction. If your organization is looking for a real-time platform, you should evaluate Microsoft Azure

Stream Analytics. Microsoft Azure Stream Analytics allows you to monitor event streams in real time and push results to a Power BI dashboard.

Finally, BI pros can implement predictive data analytics solutions that integrate with Power BI. For example, Elena can use the Azure Machine Learning Service to implement a data mining model that predicts the customer probability to purchase a product. Then she can easily set up a REST API web service, which Power BI can integrate with to display results.

If all these BI pro features sound interesting, I'll walk you through these scenarios in detail in Part 3 of this book.

1.3.4 Power BI for Developers

Power BI has plenty to offer to developers as well because it's built on an open and extensible architecture. In the context of data analytics, developers are primarily interested in incorporating BI features in their applications or in providing access to data to support integration scenarios. For example, Teo is a developer with Adventure Works. Teo might be interested in embedding Power BI dashboards and reports in a web application that will be used by external customers. Power BI supports several extensibility options, including content packs, real-time reporting, custom visuals, and embedded reporting.

Real-time content and management

Power BI has a set of REST APIs to allow developers to programmatically manage certain Power BI resources, such as enumerating datasets, creating new datasets, and adding and removing rows to a dataset table. This allows developers to push data to Power BI, such as to create real-time dashboards. In fact, this is how Azure Stream Analytics integrates with Power BI. When new data is streamed, Azure Data Analytics pushes the data to Power BI to update real-time dashboards.

The process for creating such applications is straightforward. First you need to register your application with Microsoft Azure. Then you write OAuth2 security code to authenticate your application with Power BI. Then you'd write code to manipulate the Power BI objects using REST APIs. For example, here's a sample method invocation for adding one row to a table:

```
POST https://api.powerbi.com/beta/myorg/datasets/2C0CCF12-A369-4985-A643-0995C249D5B9/Tables/Product/Rows  
HTTP/1.1
```

```
Authorization: Bearer {AAD Token}
```

```
Content-Type: application/json
```

```
{
```

```
    "rows":
```

```
[
```

```
{
```

```
    "ProductID":1,
```

```
    "Name":"Adjustable Race",
```

```
    "Category":"Components",
```

```
        "IsCompete":true,  
        "ManufacturedOn":"07/30/2014"  
    }  
]
```

}

Microsoft supports a Power BI Developer Center website

(<https://powerbi.microsoft.com/developers>) where you can read the REST API documentation and try the REST APIs.

Embed reports in custom applications

Many of you would like to embed beautiful Power BI dashboards and reports in custom web applications. For example, your company might have a web portal to allow external customers to log in and access reports and dashboards. Up until Power BI, Microsoft hasn't had a good solution to support this scenario. True, Microsoft provides an ASP.NET ReportViewer control in Visual Studio to let developers embed SSRS reports on custom ASP.NET pages but SSRS reports lack interactive features.

The good news is that Power BI has REST APIs to let developers embed dashboard tiles and reports. For example, Teo codes a customer-facing web application and he needs to allow end users to view Power BI content. Teo can integrate the web app with the Power BI REST APIs and embed Power BI reports. When the user logs in and navigates to a report, the user can see the report inside the application without having to navigate to Power BI Service. And, because embedded reports preserve interactive features, users can enjoy the same engaging experience, including report filtering, interactive sorting and highlighting.

Implement custom visuals

Microsoft has published the required interfaces to allow developers to implement and publish custom visuals using any of the JavaScript-based visualization frameworks, such as D3.js, WebGL, Canvas, or SVG.

Do you need visualizations that Power BI doesn't support to display data more effectively? With some coding wizardry, you can implement your own!

Moreover, to speed up development Microsoft has published the code of all of the existing Power BI visualizations to GitHub (<https://github.com/Microsoft/PowerBI-Visuals>). Once you follow the instructions to install the Power BI Visuals Sample, you can see the source code (visuals are coded in TypeScript) and then create and test new visuals. When the custom visual is ready, you can publish it to the Power BI visuals gallery at <https://visuals.powerbi.com> where Power BI users can search for it and download it.

Implement content packs

I've already discussed how Power BI content packs can help you connect to popular online data sources, such as Dynamics CRM or Google Analytics. If you work for an independent software vendor (ISV) or System Integrator (SI), you can implement new content packs to facilitate access to data and to provide prepackaged content. You can contact Microsoft at <http://solutions.powerbi.com/appsuggestion.html> and sign up for the

Microsoft partner program which coordinates this initiative.

As part of implementing a content pack, you'd need to create a connector using REST, OData, ODBC or other APIs to allow Power BI to connect to your data source and retrieve data. Your content pack can also include custom data models, reports, and dashboards.

1.4 Summary

This chapter has been a whirlwind tour of the innovative Power BI cloud data analytics service and its features. By now, you should view Power BI as a flexible platform that meets a variety of BI requirements. An important part of the Microsoft Data Platform, Power BI is a collective name of three products: Power BI Service, Power BI Mobile, and Power BI Desktop. You've learned about the major reasons that led to the release of Power BI. You've also taken a close look at the Power BI architecture and its components, as well as its editions and pricing model.

Next, this chapter discussed how Power BI can help different types of users with their data analytics needs. It allows business users to connect to their data and gain quick insights. It empowers BI analysts to create sophisticated data models. It enables IT and BI pros to implement hybrid solutions that span on-premises data models and reports deployed to the cloud. Finally, its extensible and open architecture let developers enhance the Power BI data capabilities and integrate Power BI with custom applications.

Having laid the foundation of Power BI, you're ready to continue the journey. Next, you'll witness the value that Power BI can deliver to business users.



Power BI for Business Users

If you're new to Power BI, welcome! This part of the book provides the essential fundamentals to help you get started with Power BI. It specifically targets business users: people who use Excel as part of their job, such as information workers, executives, financial managers, business managers, people managers, HR managers, and marketing managers. But it'll also benefit anyone new to Power BI. Remember from Chapter 1 that Power BI consists of Power BI Service, Power BI Mobile, and Power BI Desktop. This part of the book teaches business users how to use the first two products, Power BI Service and Power BI Mobile.

First, you'll learn how to sign up and navigate the Power BI portal. You'll also learn how to use content packs so that you connect to popular online services. Because business users often have to analyze simple datasets, this chapter will teach you how to import data from files without explicit data modelling.

Next, you'll learn how to use Power BI Service to create reports and dashboards and uncover valuable insights from your data. As you'll soon see, Power BI doesn't assume you have any query knowledge or reporting skills. With a few clicks, you'll be able to create ad hoc interactive reports! Then you'll create dashboards from existing visualizations or by asking natural questions.

If you frequently find yourself on the go, I'll show you how you can use Power BI Mobile to access your reports and dashboards on the go as long as you have Internet connectivity. Besides mobile rendering, Power BI Mobile offers interesting features to help you stay on top of your business, including data alerts, favorites, and annotations.

As with the rest of the book, step-by-step instructions will guide you through the tour. Most of the features that I'll show you in this part of the book are available in the free edition of Power BI, so you can start practicing immediately. The features that require Power BI Pro will be explicitly stated.

Chapter 2

The Power BI Service

In the previous chapter, I explained that Power BI aims to democratize data analytics and make it available to any user...BI for the masses! As a business user, you can use Power BI to get instant insights from your data irrespective if it's located on premises or in the cloud. Although no clear boundaries exist, I define a business user as someone who would be mostly interested in consuming BI artifacts, such as reports and dashboards. However, when requirements call for it, business users could also produce content, such as to visualize data stored in Excel files. Moreover, basic data analytics requirements can be met without explicit modeling.

This chapter lays out the foundation of self-service data analytics with Power BI. First, I'll help you understand when self-service BI is a good choice. Then I'll get you started with Power BI by showing you how to sign up and navigate the Power BI portal. Next, I'll show you how to use content packs to connect to a cloud service and quickly gain insights from prepackaged reports and dashboards. If you find yourself frequently analyzing data in Excel files, I'll teach you how do so using the Power BI Free edition.

2.1 Introducing Self-service Business Intelligence

Remember that self-service BI enables business users (information workers, like business managers or marketing managers, and power users) to offload effort from IT pros so they have to stay in line waiting for someone to enable BI for them. And, team BI allows the same users to share their reports with other team members without requiring them to install modeling or reporting tools. Before we go deeper in personal and team BI, let's take a moment to compare it with organizational BI. This will help you view self-service BI not as a competing technology but as a completing technology to organizational BI. In other words, self-service BI and organizational BI are both necessary for most businesses, and they complement each other.

2.1.1 Understanding Organizational BI

Organizational BI defines a set of technologies and processes for implementing an end-to-end BI solution where the implementation effort is shifted to IT professionals (as opposed to information workers and people who use Power BI Desktop or Excel as part of their job).

Classic organizational BI architecture

The main objective of organizational BI is to provide accurate and trusted analysis and reporting.

Figure 2.1 shows a classic organizational BI solution.

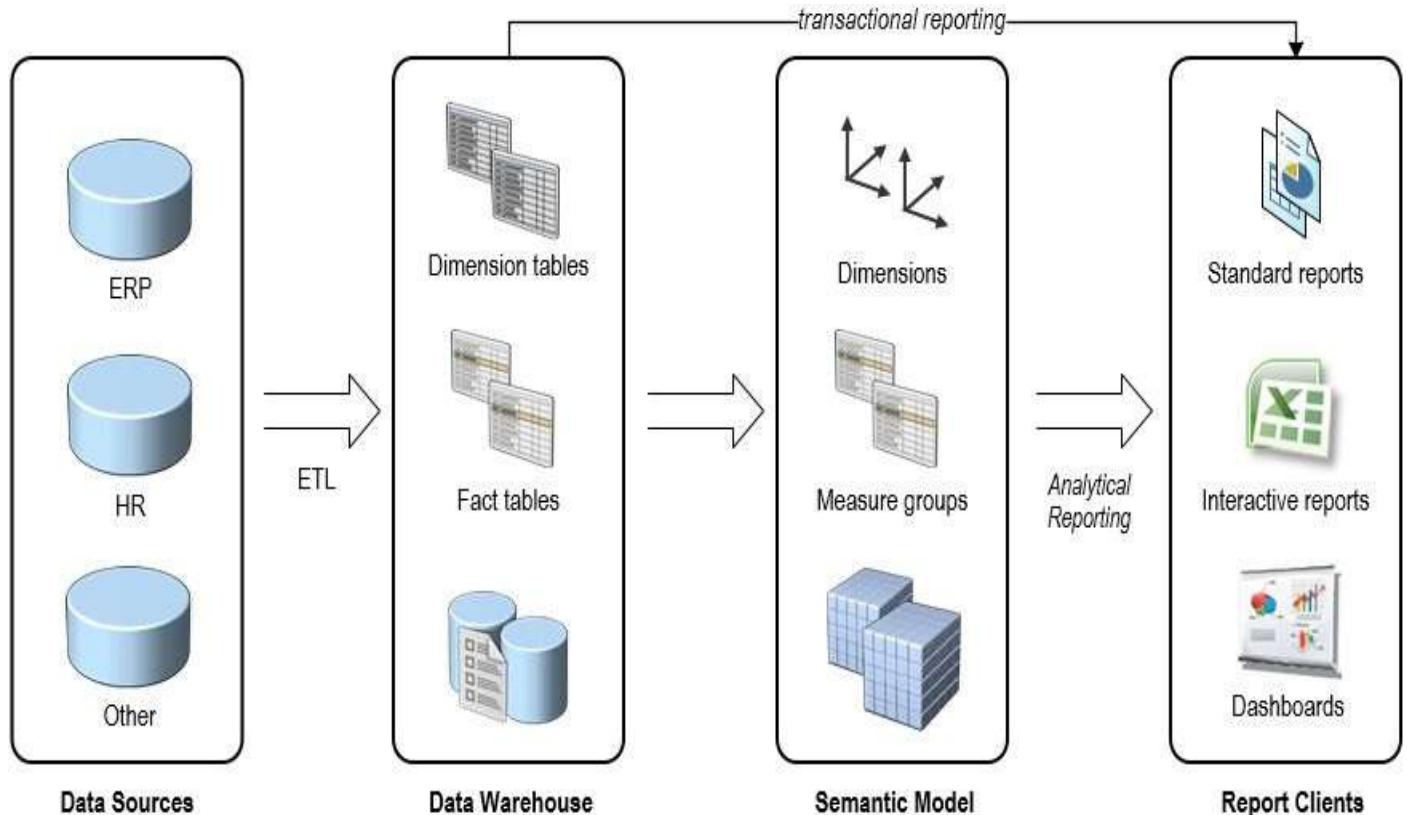


Figure 2.1 Organizational BI typically includes ETL processes, data warehousing, and a semantic layer.

In a typical corporate environment, data is scattered in a variety of data sources, and

consolidating it presents a major challenge. Extraction, transformation, and loading (ETL) processes extract data from the original data sources, clean it, and then load the trusted data in a data warehouse or data mart. The data warehouse organizes data in a set of dimensions and fact tables. When designing the data warehouse, BI pros strive to reduce the number of tables in order to make the schema more intuitive and facilitate reporting processes. For example, an operational database might be highly normalized and have Product, Subcategory, and Category tables. However, when designing a data warehouse, the modeler might decide to have a single Product table that includes columns from the Subcategory and Category tables. So instead of three tables, the data warehouse now has only one table, and end users don't need to join multiple tables.

While end users could run transactional reports directly from the data warehouse, many organizations also implement a semantic model in the form of one or more Analysis Services Multidimensional cubes or Tabular models for analytical reporting. As an information worker, you can use Power BI Desktop, Excel, or another tool to connect to the semantic model and then start slicing and dicing data, such as to see how the product sales are doing over time. And IT pros can create operational reports and dashboards from the cube.

Understanding organizational BI challenges

Although it's well-defined and established, organizational BI might face a few challenges, including the following:

- Significant implementation effort – Implementing an organizational BI solution isn't a simple undertaking. Business users and IT pros must work together to derive requirements. Most of the implementation effort goes into data logistics processes to clean, verify, and load data. For example, Elena from the IT department is tasked to implement an organizational BI solution. First, she needs to meet with business users to obtain the necessary business knowledge and gather requirements (business requirements might be hard to come by). Then she has to identify where the data resides and how to extract, clean, and transform the data. Next, Elena must implement ETL processes, models, and reports. Quality Assurance must test the solution. And IT pros must configure the hardware and software, as well as deploy and maintain the solution. Security and large data volumes bring additional challenges.
- Highly specialized skillset – Organizational BI requires specialized talent, such as ETL developers, Analysis Services developers, and report developers. System engineers and developers must work together to plan the security, which sometimes might be more complicated than the actual BI solution.
- Less flexibility – Organization BI might not be flexible enough to react quickly to new or changing business requirements. For example, Maya from the Marketing department might be tasked to analyze CRM data that isn't in the data warehouse. Maya might need to wait for a few months before the data is imported and validated.

The good news is that self-service BI can complement organizational BI quite well to address these challenges. Given the above example, while waiting for the pros to enhance

the organization BI solution, Maya can use Power BI to analyze CRM data or Excel files. She already has the domain knowledge. Moreover, she doesn't need to know modeling concepts. At the beginning, she might need some guidance from IT, such as how to get access to the data and understand how the data is stored. She also needs to take *responsibility* that her analysis is correct and can be trusted. But isn't self-service BI better than waiting?

2.1.2 Understanding Self-service BI

Self-service BI empowers business users to take analytics in their own hands with guidance from their IT department. For companies that don't have organizational BI or can't afford it, self-service BI presents an opportunity for building customized ad hoc solutions to gain data insights outside the capabilities of organizational BI solutions and line-of-business applications. On the other hand, organizations that have invested in organizational BI might find that self-service BI opens additional options for valuable data exploration and analysis.

REAL WORLD I led a Power Pivot training class for a large company that has invested heavily in organizational BI. They had a data warehouse and OLAP cubes. Only a subset of data in the data warehouse was loaded in the cubes. Their business analysts were looking for a tool that would let them join and analyze data from the cubes and data warehouse. In another case, an educational institution had to analyze expense report data that wasn't stored in a data warehouse. Such scenarios can benefit greatly from self-service BI.

Self-service BI benefits

When done right, self-service BI offers important benefits. First, it makes BI pervasive and accessible to practically everyone! Anyone can gain insights if they have access to and understand the data. Users can import data from virtually any data source, ranging from flat files to cloud applications. Then they can mash it up and gain insights. Once data is imported, the users can build their own reports. For example, Maya understands Excel but she doesn't know SQL or relational databases. Fortunately, Power BI doesn't require any technical skills. Maya could import her Excel file and build instant reports.

Besides democratizing BI, the agility of self-service BI can complement organizational BI well, such as to promote ideation and divergent thinking. For example, as a BI analyst, Martin might want to test a hypothesis that customer feedback on social media, such as Facebook and Twitter, affects the company's bottom line. Even though such data isn't collected and stored in the data warehouse, Martin can import data from social media sites, relate it to the sales data in the data warehouse and validate his idea.

Finally, analysts can use self-service BI tools, such as Power Pivot and Power BI Desktop, to create prototypes of the data models they envision. This can help BI pros to understand their requirements.

Self-service BI cautions

Self-service BI isn't new. After all, business users have been using tools like Microsoft Excel and Microsoft Access for isolated data analysis for quite a while (Excel has been around since 1985 and Access since 1992). Here are some considerations you should keep in mind about self-service BI:

- What kind of user are you? – Are you a data analyst (power user) who has the time, desire, and patience to learn a new technology? If you consider yourself a data analyst, then you should be able to accomplish a lot by creating data models with Power BI Desktop and Excel Power Pivot. If you’re new to BI or you lack data analyst skills, then you can still gain a lot from Power BI and this part of the book shows you how.
- Data access – How will you access data? What subset of data do you need? Data quality issues can quickly turn away any user, so you must work with your IT to get started. A role of IT is to ensure access to clean and trusted data. Analysts can use Power BI Desktop or Excel Power Query for simple data transformations and corrections, but these aren’t meant as ETL tools.
- IT involvement – Self-service BI might be good, but managed self-service BI (self-service BI under the supervision of IT pros) is even better and sometimes a must. Therefore, the IT group must budget time and resources to help end users when needed, such as to give users access to data, to help with data integrity and more complex business calculations, and to troubleshoot issues when things go wrong. They also must monitor the utilization of the self-service rollout.
- With great power comes great responsibility – If you make wrong conclusions, damage can be easily contained. But if your entire department or even organization uses wrong reports, you have a serious problem! You must take the responsibility and time to verify that your model and calculations can be trusted.
- “Spreadmarts” – I left the most important consideration for the CIO for last. If your IT department has spent a lot of effort to avoid decentralized and isolated analysis, should you allow the corporate data to be constantly copied and duplicated?

TIP Although every organization is different, I recommend an 80/20 split between organizational BI and self-service BI. This means that 80% of the effort and budget should be spent in designing and implementing organizational BI artifacts and backend services, including data warehouses, data quality, centralized semantic models, trusted reports, dashboards, Big Data initiatives, master data management, and so on. The remaining 20% would be focused on agile and managed self-service BI.

Now that you understand how organizational BI and self-service BI compares, let’s dive into the Power BI self-service BI capabilities which benefit business users like you.

2.2 Getting Started with Power BI Service

In Chapter 1, I introduced you to Power BI and its products. Recall that the main component of Power BI is its cloud-hosted Power BI Service (powerbi.com). Assuming that you're a novice user, this section lays out the necessary startup steps, including signing up for Power BI and understanding its web interface. As you'll soon find out, because Power BI was designed with business users and data analytics in mind, it won't take long to learn it!

2.2.1 Signing Up for Power BI

The Power BI motto is, “5 seconds to sign up, 5 minutes to wow!” Because Power BI is a cloud-based offering, there’s nothing for you to install and set up. But if you haven’t signed up for Power BI yet, let’s put this promise to the test. But first, read the following steps.

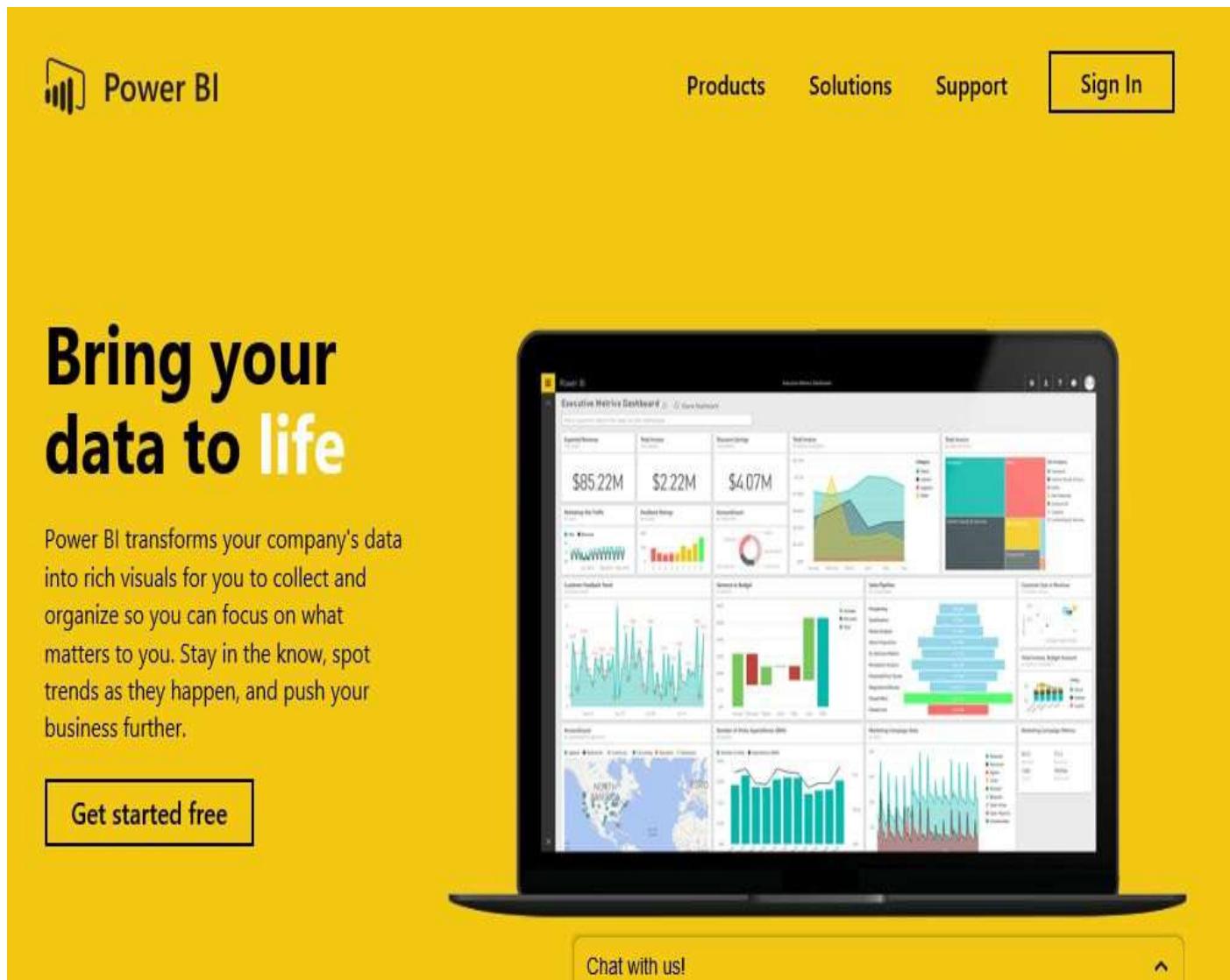


Figure 2.2 This is the landing Power BI Service web page.

Five seconds to sign up

Follow these steps to sign up for the Power BI Service:

1. Open your Internet browser and navigate to <http://www.powerbi.com> and click “Get

started free” (see **Figure 2.2**).

- 2.The next page asks you how you want to start with Power BI: download Power BI Desktop so that you can create a self-service data model on your desktop or sign up for Power BI so that you can use the Power BI Service. I’ll discuss Power BI Desktop in Part 2 of this book so let’s ignore this option for now. Click the Sign Up button because you want to use Power BI Service.
- 3.In the Get Started page, enter your work email address. Notice that the email address must be your work email. At this time, you can’t use a common email, such as @hotmail.com, @outlook.com, or @gmail.com. This might be an issue if you plan to use Power BI for your personal use. Until this limitation is lifted, consider registering a domain, such a domain for your family (some providers give away free domains).
- 4.If your organization already uses Office 365, Power BI will detect this and ask you to sign up using your Office 365 account. If you don’t have an Office 365 subscription, Power BI will ask you to confirm the email you entered and then to check your inbox for a confirmation email.
- 5.Once you receive your email conformation with the subject “Time to complete Microsoft Power BI signup”, click the “Complete Microsoft Power BI Signup” link in the email. Clicking on the link will take you to a page to create your account (see **Figure 2.3**).



Figure 2.3 Use this page to create a Power BI account.

- 6.You need to provide a name and a password, and then click Start.

This completes the process which Microsoft refers to as the “Information Worker (IW) Sign Up” flow. As I said, this signup flow is geared for an organization that don’t have an Office 365 tenant. By the way, this is similar to the signup process for the Office 365 Educational plan (EDU) for O365.

NOTE Your organization might have concerns about indiscriminate enrolling to Power BI, uploading corporate data, removing users who have signed up, and so on. Answers can be found in the “Power BI in your organization” document by Microsoft at <http://bit.ly/1IY87Qt>.

The main page

After you complete the signup process, the next time you go to powerbi.com, click the Sign In button in the top-right corner of the main page (see **Figure 2.2** again). The main page includes the following menus:

- Products – Explains the Power BI product family and pricing. The Power BI Mobile page includes download links for the native mobile applications.
- Solutions – Explains how Power BI addresses various data analytics needs.
- Support – Includes links to the product documentation, the community forums where you can ask questions, a page to provide feedback to Microsoft, the Power BI blog, and a page to find a Microsoft partner to help you if you need training or implementation assistance.

What happens during signup?

You might be curious why you’re asked to provide a password given that you sign up with your work email. Behind the scenes, Power BI stores the user credentials in Azure Active Directory (Azure AD). If your organization doesn’t have an Office 365 subscription, the Information Worker flow creates a shadow tenant for the domain you used to sign up. For example, if I sign up as teo@prologika.com and my company doesn’t have an Office 365 subscription, a shadow tenant for prologika.com will be created in Azure AD and that tenant won’t be managed by anyone at my company.

NOTE What is a Power BI tenant? A tenant is a dedicated instance of the Azure Active Directory that an organization receives and owns when it signs up for a Microsoft cloud service such as Azure, Microsoft Intune, Power BI, or Office 365. A tenant houses the users in a company and the information about them - their passwords, user profile data, permissions, and so on. It also contains groups, applications, and other information pertaining to an organization and its security. For more information about tenants, see “What is an Azure AD tenant” at <http://bit.ly/1FTFObb>.

If your organization decides one day to use more of the Azure services, it can synchronize or federate your corporate Active Directory with Azure, but this isn’t required. To unify the corporate and cloud directories, the company IT administrator can then take over the shadow tenant. To make this easier, he can use the DirSync tool, which will do this automatically, or he can extend your corporate AD to Azure. For more information about the process, refer to the blog “How to perform an IT Admin Takeover with O365” by Adam Saxton at <http://blogs.technet.com/b/powerbisupport/archive/2015/02/06/how-to-perform-an-it-admin-takeover-with-o365.aspx>.

2.2.2 Understanding the Power BI Portal

I hope it took you five seconds or less to sign up with Power BI. (Or at least hopefully it feels quick.) After completing this signup steps, you’ll have access to the free edition of Power BI. Let’s take a moment to get familiar with the Power BI portal where you’ll spend most of your time when analyzing data.

Welcome to Power BI page

Upon signup, Power BI discovers that you don’t have any BI artifacts yet. This is why it navigates you to the “Welcome to Power BI” page which is shown in **Figure 2.4**.

NOTE Currently, the UI of the Power BI portal isn’t customizable. For example, you can’t rearrange or remove menus, and you can’t brand the portal. If these features are important to your organization, your IT department can

consider installing an on-premises SharePoint Server or SharePoint Online and then embed Power BI dashboards and reports inside SharePoint. SharePoint supports a comprehensive set of branding and UI customization features and you can embed Power BI content on SharePoint pages. I'll discuss Power BI content embedding in Chapter 11.

The screenshot shows the 'Welcome to Power BI' page. At the top, there's a navigation bar with icons for settings, download, help, and user profile, along with a search bar. The main title 'Welcome to Power BI' is displayed prominently. Below it, a message says 'You're on your way to exploring your data and monitoring what matters.' followed by 'Let's start by getting some data.' A link 'Need more guidance? Try this tutorial' is provided. The page is divided into sections: 'Content Pack Library' and 'Import or Connect to Data'. Under 'Content Pack Library', there are four tiles: 'My Organization', 'Services', 'Files', and 'Databases', each with a 'Get' button. Under 'Import or Connect to Data', there is one tile with a 'Samples' icon. The overall layout is clean and modern, designed to guide users through their first steps in Power BI.

Figure 2.4 The “Welcome to Power BI” page allows you to connect to data and install samples.

Before analyzing data, you need to first connect to wherever it resides. This is why the “Welcome to Power BI” page prompts you to start your data journey by connecting to your data. The My Organization tile under the “Content Pack Library” section allows you to browse and use organizational packs, assuming that someone within your organization has already published BI content as organizational packs. The Services tile allows you to use Microsoft-provided content packs to connect to popular online services, such as Google Analytics, Salesforce, Microsoft Dynamics CRM, and many more.

The Files tile under the “Import or Connect to Data” section lets you import data from

Excel, Power BI Desktop, and CSV files. The Databases tile allows you to connect to data sources that support live connections, such as Azure SQL Database and SQL Server Analysis Services.

NOTE A popular option that's missing on this page is connecting to an on-premises database, such as a SQL Server or Oracle. Currently, this scenario requires you create a data model using Power BI Desktop or Excel before you can import data from on-premises databases. Power BI Desktop also supports connecting directly to some data sources, such as SQL Server. Then you can upload the model to Power BI. Because it's a more advanced scenario, I'll postpone discussing Power BI Desktop until Chapter 5.

- 1.To get some content you can explore in Power BI, click the Samples link.
- 2.In the Samples page, click the Retail Analysis Sample tile. As the popup informs you, the Retail Analysis Sample is a sample dashboard provided by Microsoft to demonstrate some of the Power BI capabilities. Click the Connect button. Are you concerned that samples might clutter the portal? Don't worry; it's easy to delete the sample later. To do this, you can just delete the Retail Analysis Sample dataset which will delete the dependent reports and dashboards.

The Power BI portal

After installing the Retail Analysis Sample, Power BI navigates you to the home page, which has the following main sections (see the numbered areas in **Figure 2.5**):

- 1.Navigation Pane – The navigation pane organizes the content deployed to Power BI. By default, all content is deployed to your personal workspace (My Workspace). If you have a Power BI Pro subscription and you create other workspaces to collaborate with coworkers, the My Workspace drop-down allows you to select another workspace. The Dashboards section includes the dashboards that you've developed or that someone shared with you. Similarly, the Reports section contains all reports that are available to you. Similarly, the Datasets section shows all the datasets that you've created or that someone shared with you. Finally, the Get Data button at the bottom brings you to the Get Data page. Similar to the "Welcome to Power BI" page, the Get Data page allows you to create additional datasets. Every dashboard, report, and dataset has a context menu which you can open by clicking the ellipsis button (...) to the right of the item. This context menu allows you to take additional actions, such as to rename the item.

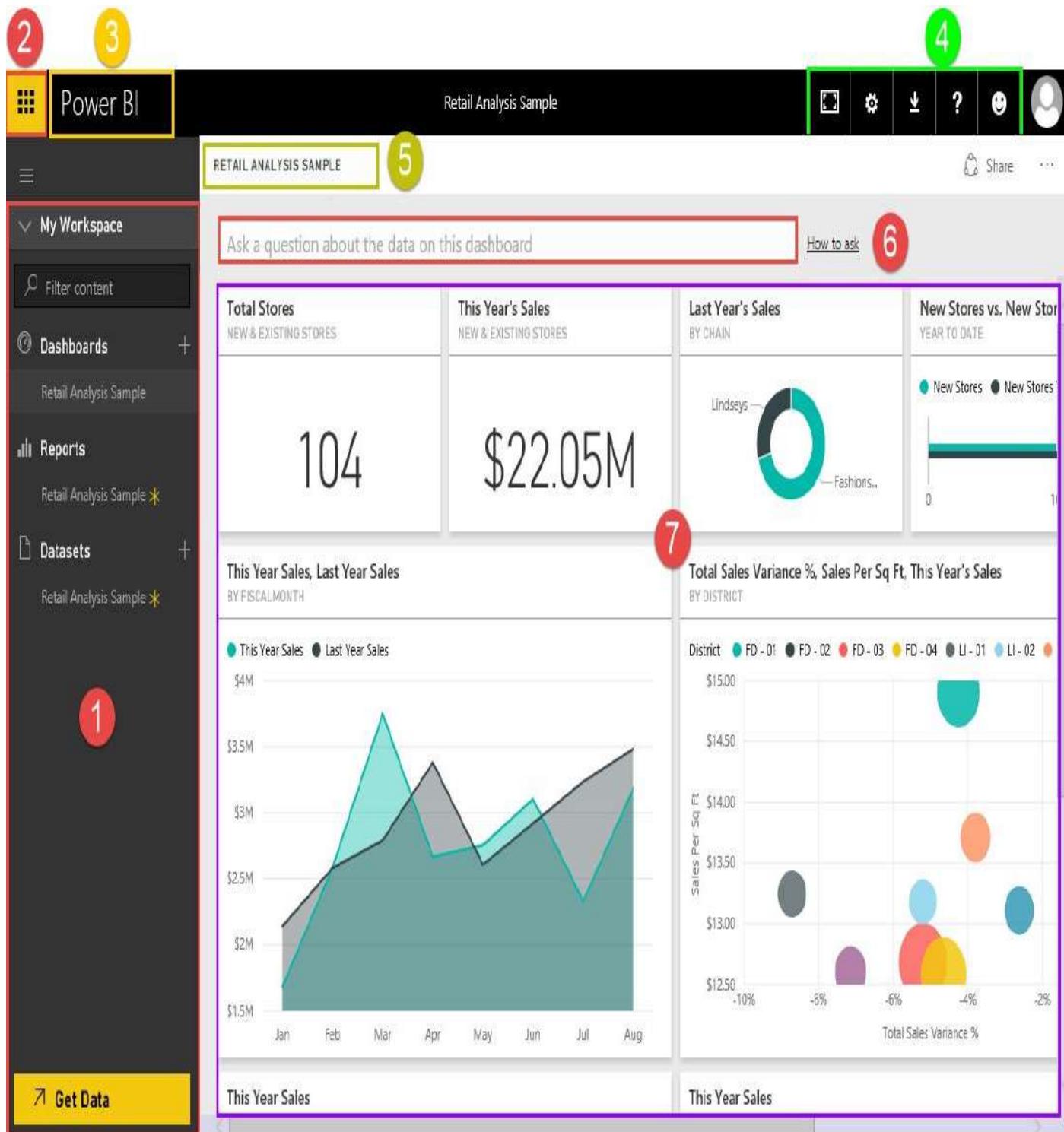


Figure 2.5 The Power BI home page.

2. Office 365 Application Launcher – If you have an Office 365 subscription, this menu allows you to access the Office 365 application you are licensed to use. Doesn't Microsoft encourage you to use Office 365?
3. Power BI Home – No matter which page you've navigated to, this menu takes you to the main portal page (shown in **Figure 2.5**).
4. Application Toolbar – Let's explain the available menus starting from the left:
 - Enter Full Screen Mode menu – Shows the active content in full screen and removes the Power BI UI (also called “chrome”). Once you're in Full Screen mode, you have options to resize the content to fit to screen and to exit this mode (or press Esc). Another way to

open a dashboard in a full screen mode is to append the chromeless=1 parameter to the dashboard URL, such as:

<https://app.powerbi.com/groups/me/dashboards/3065afc5-63a5-4cab-bcd3-0160b3c5f741?chromeless=1>

- Settings menu – This menu expands to several submenus. Click “Manage Personal Storage” to check how much storage space you’ve used (recall that the Power BI Free and Power BI Pro editions have different storage limit). If you have Power BI Pro, the “Create content pack” submenu allows you to create an organizational content pack (discussed in Chapter 9) while “View content pack” allows you to access published content packs. I mentioned how Power BI Service supersedes Power BI for Office 365 but in case you haven’t transitioned yet, the “The “Power BI for Office 365” submenu navigates you to Office 365 Power BI. Use the Settings submenu to view and change some Power BI Service settings, such as if the Q&A box is available for a given dashboard. The Dev Tools menu allows developers to create and test custom visuals.

TIP Not sure what Power BI edition you have? Click the Settings menu, and then click “Manage Personal Storage”. At the top of the next page, notice the message next to your name. If it says “Free User” you have the Power BI free edition. If it says “Pro User” then you have Power BI Pro subscription.

- Download menu – This menu allows you to download useful Power BI components, including Power BI Mobile (a set of native Power BI apps for your mobile devices), Power BI Desktop (for analysts wanting to create self-service data models), Analysis Services Connector (to connect to on-premises Analysis Services models), and Power BI gateways (to connect to on-premises data sources).
- Help and Support menu – Includes several links to useful resources, such as product documentation, the community site, and developer resources.
- Feedback menu – Rate your experience with Power BI on a scale from 1 to 10.

5.Dashboard name – Displays the dashboard name. The Share link to the right of the dashboard allows you to share a dashboard with your coworkers. And the ellipsis menu (...) to the right of the Share link lets you duplicate the dashboard.

6.Natural question box (Q&A) – When you select a dashboard and the dashboard uses a dataset that supports natural queries, you can use this box to enter the natural query. For example, you can ask it how many units were shipped in February last year.

7.Content area – This is where the content of the selected item in the navigation pane is shown. For example, if a dashboard is selected, the content pane shows the dashboard tiles.

2.3 Understanding Power BI Content Items

The key to understanding how Power BI works is to understand its three main content items: datasets, reports, and dashboards. These elements are interdependent, and you must understand how they relate to each other. For example, you can't have a report or dashboard without creating one or more datasets.

Figure 2.6 should help you understand these dependencies.

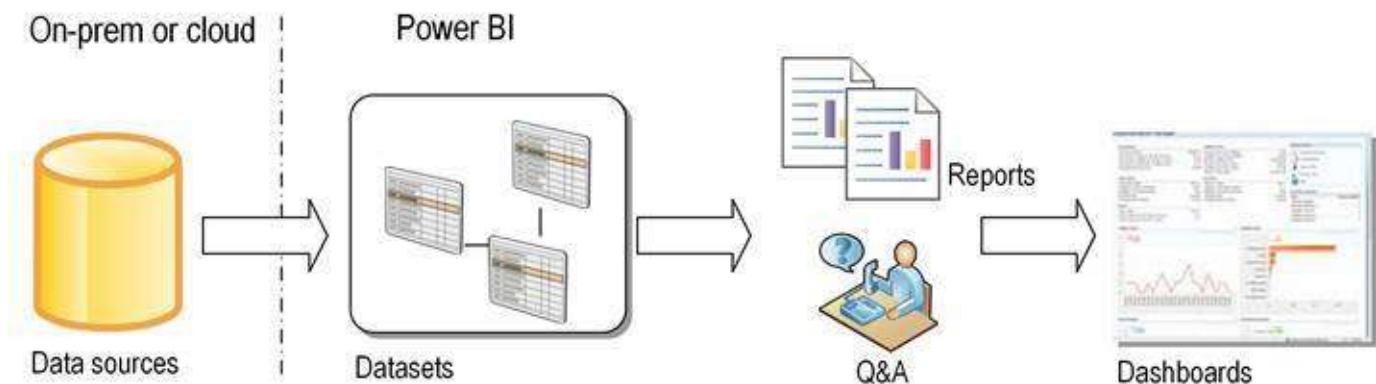


Figure 2.6 The Power BI main content items are datasets, reports, and dashboards.

2.3.1 Understanding Datasets

Think of a dataset as a blueprint or a definition of the data that you analyze. For example, if you want to analyze some data stored in an Excel spreadsheet, the corresponding dataset represents the data in the Excel spreadsheet. Or, if you import data from a database table, the dataset will represent that table. Notice in **Figure 2.6**, however, that a dataset can have more than one table. For example, if Martin uses Power BI Desktop or Excel to create a data model, the model might have multiple tables (potentially from different data sources). When Martin uploads the model to Power BI, his entire model will be shown as a single dataset in the navigation pane but when you explore it (click the ellipsis next to the dataset and then click Explore), you'll see that the Fields pane shows multiple tables. You'll encounter another case of a dataset with multiple tables when you connect to an Analysis Services model.

Getting data

Data sources with useful data for analysis are everywhere (see **Figure 2.7**). As far as the data source location goes, we can identify two main types of data sources:

- Cloud (SaaS) services – These data sources are hosted in the cloud and available as online services. Examples of Microsoft cloud data sources that Power BI supports include OneDrive, Dynamics CRM, Azure SQL Database, Azure SQL Data Warehouse, and Spark on Azure HDInsight. Power BI can also access many popular cloud data sources from other vendors, such as Salesforce, Google Analytics, Marketo, and many others (the list is growing every month!).
- On-premises data sources – This category encompasses all other data sources that are internal to your organization, such as operational databases, cubes, Excel and other files.

Depending on the capabilities and location of the data source, data can be either a) imported in a Power BI dataset or b) left in the original data source without importing it, but it can be accessed directly via a live connection. If the data source supports it, direct connectivity (requires Power BI Pro edition) is appropriate when you have fast data sources. In this case, when you generate a report, Power BI creates a query using the syntax of the data source and sends the query directly to the data source. So, the Power BI dataset has only the definition of the data but not the actual data. As of the time of writing this book, only a few data sources support direct connectivity, including SQL Server Analysis Services, Azure SQL Database, and Azure SQL Data Warehouse, Spark on Azure HDInsight. (Power BI Desktop also supports direct connections to SQL Server and SAP Hana.) The list of directly accessible data sources is likely to grow in time.

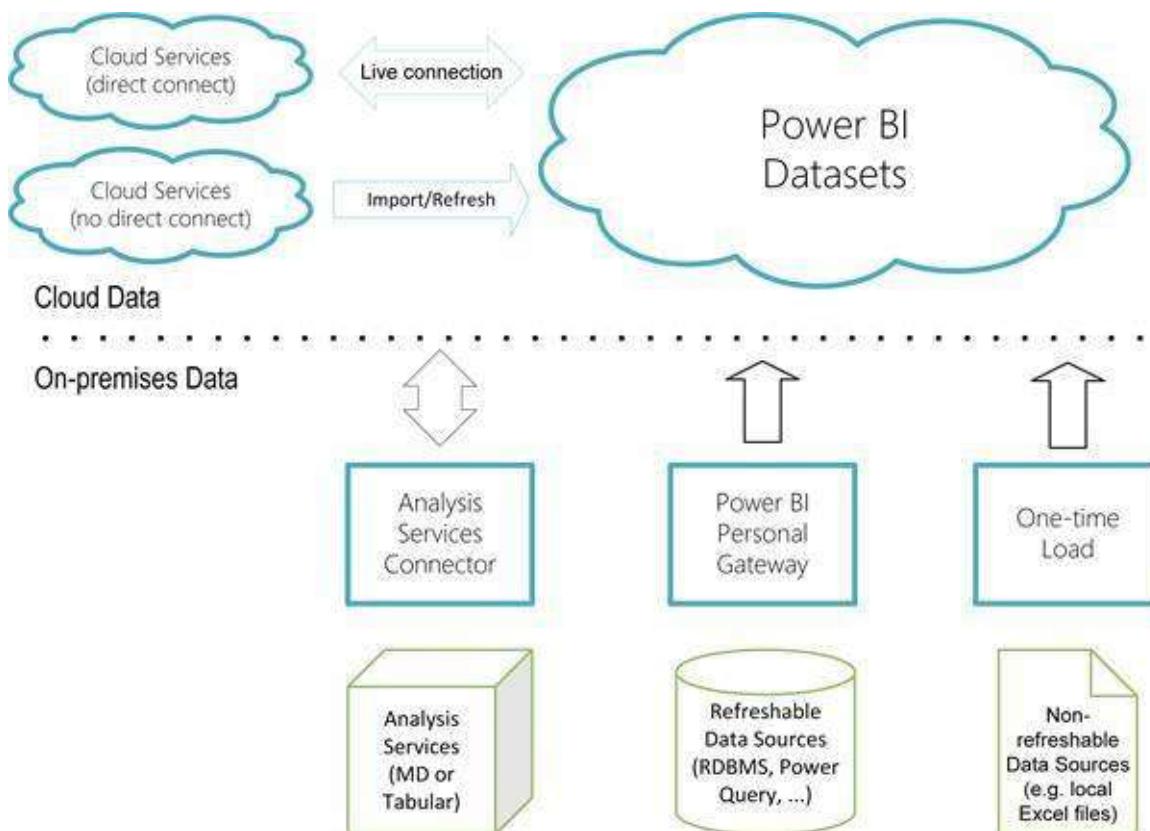


Figure 2.7 Power BI can import data or create live connections to some data sources.

Because only a limited set of data sources supports live connectivity, in most cases you'll be *importing* data irrespective of whether you access cloud and on-premises data sources. When you import data, the Power BI dataset has the definition of the data *and* the actual data. In Chapter 1, I showed you how when you import data, Microsoft deploys the dataset to scalable and highly-performant Azure backend services. Therefore, when you create reports from imported datasets, performance is good and predictable. But the moment the data is imported, it becomes outdated because changes in the original data source aren't synchronized with the Power BI datasets. Which brings me to the subject of refreshing data.

Refreshing data

Deriving insights from outdated data is rarely useful. Fortunately, Power BI supports

automatic data refresh from many data sources. Refreshing data from cloud services is easy because most vendors already have connectivity APIs that allow Power BI to get to the data. In fact, chances are that if you use a content pack to access a cloud data source, it'll enable automatic data refresh by default. For example, the Google Analytics dataset refreshes by default every hour without any manual configuration.

On-premises data sources are more problematic because Power BI needs to connect to your corporate network, which isn't accessible from outside. Therefore, if you import corporate data, you or IT will need to install special software, called a "gateway", to let Power BI connect to the original data source. For personal use, you can use the Downloads menu in the Power BI portal to download and install the Personal Power BI Gateway without waiting for IT help but remember that it requires Power BI Pro. For enterprise deployments, IT can centralize data access by setting up Enterprise Power BI Gateway (discussed in chapter 9). Not all on-premises data sources are refreshable. For example, local Excel files can't be refreshed unless they're uploaded to OneDrive or OneDrive for Business or imported with Power BI Desktop. Table 2.1 summarizes the refresh options for popular data sources.

Table 2.1 This table summarizes data refresh options when data is imported from cloud and on-premises data sources.

Location	Data Source	Refresh Type	Frequency
Cloud (Gateway not required)	Most cloud data sources, including Dynamics CRM, Salesforce, Marketo, Zendesk, SendGrid, GitHub	Automatic	Once a day
	Excel, CSV, and Power BI Desktop files uploaded to OneDrive or OneDrive for Business	Automatic	Once every hour
On premises (via gateway)	Supported data sources (see https://support.powerbi.com/knowledgebase/articles/474669-refresh-data-in-power-bi)	Scheduled or manual	Daily with Power BI free or hourly with Power BI Pro
	Excel 2013 (or later) data models with Power Query data connections or Power BI Desktop data models	Scheduled or manual	Supported except Exchange, Active Directory and ODBC
	Excel files via Get Data	Not supported	

Understanding dataset tasks

Once the dataset is created, it appears under the Datasets section in the navigation pane. For example, when you installed the Retail Analysis Sample, Power BI added a dataset with the same name. You can perform several dataset tasks from the dataset properties window (**Figure 2.8**), which you can access by clicking the ellipsis menu next to the dataset.

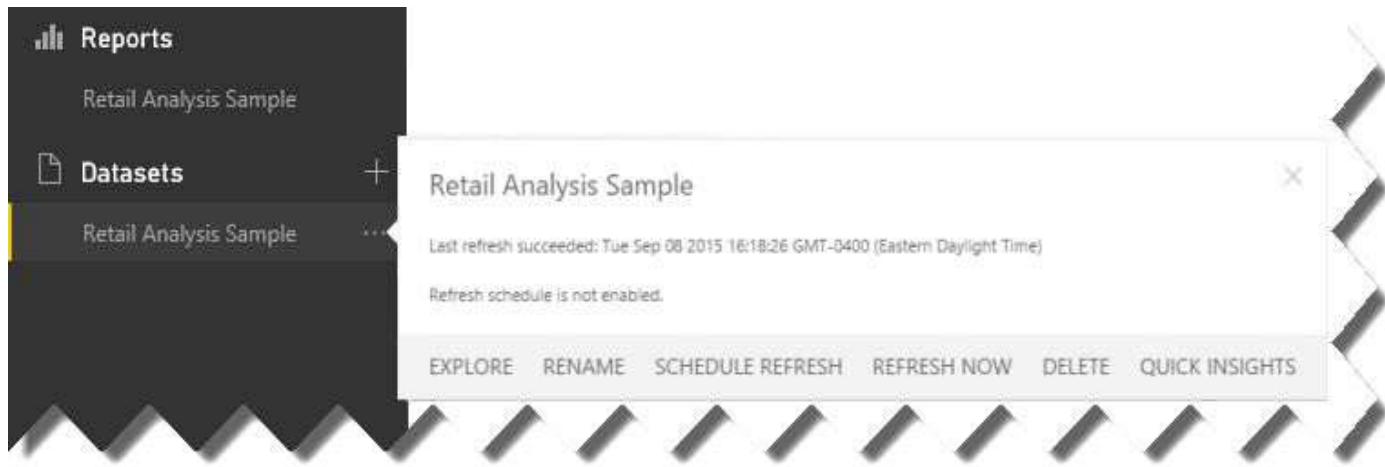


Figure 2.8 The dataset context menu allows you to perform several dataset tasks.

Going through the datasets tasks, the Explore link lets you visualize the data by creating a new report (the subject of the next section). Use the Rename link to rename a dataset. Don't worry if you have existing reports connected to the dataset because changing the dataset name won't break dependent reports and dashboards. The Schedule Refresh link allows you to set up automatic data refresh for datasets that contain imported data. If you want to refresh the data now, Refresh Now connects to the data source and immediately refreshes the data (subject to the data refresh requirements and limitations I've just covered).

If you delete a dataset, Power BI will automatically remove dependent reports and dashboard tiles that connect to that dataset. And Quick Insights (covered in the next chapter) auto-generates useful reports that might help you to understand the root cause of data fluctuations.

2.3.2 About Reports

Reports are the main way to analyze data in Power BI. Reports are found under the Reports section in the left navigation pane. For example, if you click the Retail Analysis Sample report, Power BI will open it in a reading mode (called Reading View) that supports interactive features, such as filtering, but it doesn't allow you to change the report layout. You can change the report layout by clicking the Edit Report menu on the top of the report (see **Figure 2.9**).

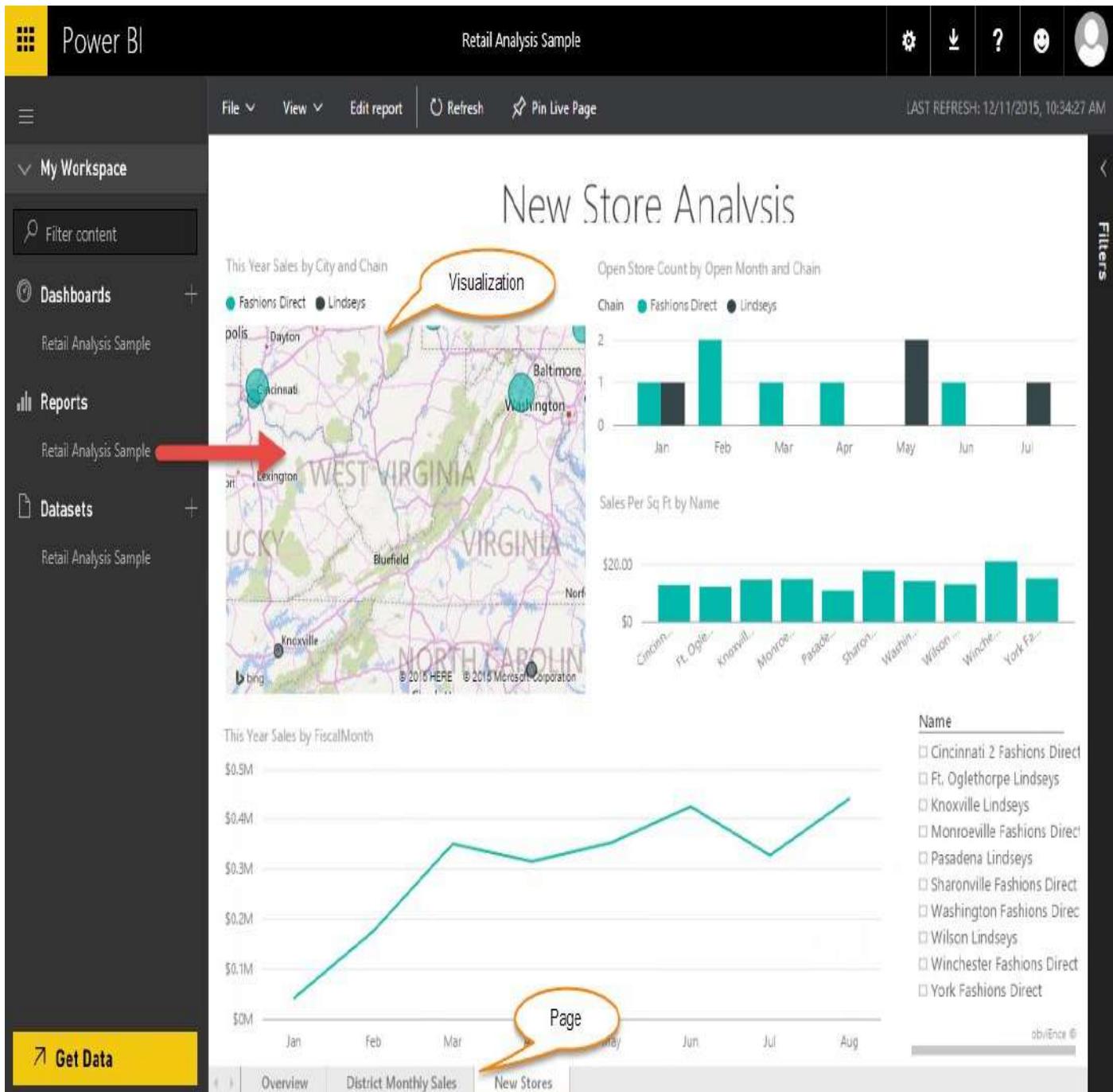


Figure 2.9 A report helps you visualize data from a single dataset.

Creating reports

Reports can be created in several ways:

- Creating reports from scratch – Once you have a dataset, you can create a new report by exploring the dataset (the Explore link in the dataset context menu) or by just clicking the dataset. Then you can save the report.
- Importing reports – If you import a data model created with Excel or Power BI Desktop and the data model include reports, Power BI will import all the reports and add them to the Reports section in the navigation pane. Currently, if you import Excel data models, only Power View reports are imported (Excel pivot and chart reports aren't imported).
- Distributing reports – If you use Power BI or organizational content packs, the reports included in the content pack are automatically distributed when you use the content

pack. This is how you got the Retail Analysis Sample report when you installed the sample.

NOTE Power BI Service can also connect to Excel files and show pivot table reports and chart reports contained in Excel files but I'll postpone discussion them to the next chapter. For now, when I talk about reports I'll mean the type of reports you can create in the Power BI portal.

How reports relate to datasets

Currently, a Power BI report can only connect to and source data from a single dataset only. Suppose you have two datasets: Internet Sales and Reseller Sales. You can't have a report that combines data from these two datasets. Although this might sound like a limitation, you have options:

- 1.Create a dashboard – If all you want is to show data from multiple datasets as separate visualizations on a single page, you can just create a dashboard.
- 2.Implement a self-service model – Remember that a dataset can include multiple tables. So, if you need a consolidated report that combines multiple datasets, a data analyst could build a self-service data model using Power BI Desktop or Excel. This works because when published to Power BI, the model will be exposed as a single dataset with multiple tables.
- 3.Connect to an organizational model – To promote a single version of the truth, a BI pro can implement an organizational data model using Analysis Services. Then you can just connect to the model; there's nothing to build or import. Finally, if all you want is to show data from multiple datasets as separate visualizations on a single page, you can just create a dashboard.

For the purposes of this chapter, this is all you need to know about reports. You'll revisit them in more detail in the next chapter.

2.3.3 About Dashboards

For a lack of a better definition, a Power BI dashboard is a summarized view with important metrics related to the data you're analyzing. Dashboards are typically utilized to convey important metrics so that management can get a high-level view of the business. To support root cause analysis, dashboards typically allow users to drill from summary sections (also called tiles in Power BI) down to more detailed reports.

Creating dashboards

Similar to reports, Power BI dashboards can be created in several ways:

- From existing reports – If you have an existing report, you can pin one or more of its visualizations to a dashboard or even an entire report page! For example, the Retail Analysis Sample dashboard was created by pinning visualizations from the report with the same name. It's important to understand that you can pin visualizations from multiple reports into the same dashboard. This allows the dashboard to display a consolidated view that spans multiple reports and datasets.
- By using Q&A – Another way to create a dashboard is to type in a question in the natural question box (see **Figure 2.5** again). This allows you to pin the resulting

visualization without creating a report. For example, you can type a question like “sales by country” if you have a dataset with sales and geography entities. If Power BI understands your question, it will show you the most appropriate visualization.

- Distributing dashboards – Dashboard can be shared via mail or distributed with content packs.

Drilling through content

To allow users to see more details below the dashboards, users can drill into dashboard tiles. What happens when you drill through depends on how the tile was created. If it was created by pinning a report visualization, you’ll be navigated to the corresponding report page. If it was created through Q&A, you’ll be navigated to the page that has the visualization and the natural question that was asked.

- 1.In the Power BI portal, click the Retail Analysis Sample dashboard in the Dashboard section.
- 2.Click the “This Year Sales, Last Year Sales” surface Area Chart. Notice that Power BI navigates to the “District Monthly Sales” tab of the Retail Analysis Sample report.

That’s all about dashboards for now. You’ll learn much more in the next chapter. Now let’s get back to the topic of data and practice the different connectivity options.

2.4 Connecting to Data

As a first step in the data exploration journey, you need to connect to your data. Let's practice what we've learned about datasets. Because this part of the book targets business users, we'll practice three data connectivity scenarios that doesn't require modeling. It might be useful to refer back to **Figure 2.4** or click the Get Data button to see these options. First, you'll see how you can use a Power BI content pack to analyze Google Analytics data. Next, I'll show you how you can import an Excel file. Finally, I'll show you how to connect live to an organizational Analysis Services model.

2.4.1 Using Content Packs

Power BI comes with pre-defined content packs that allow business users to connect to popular online services. Suppose that Maya wants to analyze the Adventure Works website traffic. Fortunately, Power BI includes a Google Analytics content pack to get her started with minimum effort. On the downside, Maya will be limited to whatever data the content pack author has decided to import which could be just a small subset of the available data.

TIP If you need more data than what's included in the content pack, consider creating a data model using Excel or Power BI Desktop that connects to the online service to access all the data. Besides data modeling knowledge, this approach requires that you understand the vendor entities and how they relate to each other. So, I suggest you first determine if the content pack has the data you need.

To perform this exercise, you'll need a Google Analytics account and you must add a tracking code to the website you want to analyze. Google supports free Google Analytics accounts. For more information about the setup, refer to <http://www.google.com/analytics>. If setting up Google Analytics is too much trouble, you can use similar steps to connect to any other online service that you use in your organization, provided that it has a Power BI connector. To see the list of the available connectors, click the Get Data button, and then click the Get button in the Services tile.

Connecting to Google Analytics

Assuming that Maya has already done the required Google Analytics setup, connecting to her Google Analytics account takes a few simple steps:

- 1.To avoid cookie issues with cached accounts, I suggest you use private browsing. If you use Internet Explorer (IE), open it and then press Ctrl-Shift-P to start a private session that ignores cookies. (Or right-click IE on the start bar and click "Start InPrivate Browsing".) If you use Google Chrome, open it and press Ctrl-Shift-N to start an incognito session. (Or right-click it on the start bar and click "New incognito window".)
- 2.Go to powerbi.com and sign in with your Power BI account. In the Power BI portal, click the Get Data button.
- 3.In the Get Data page, click the Get button in the Services tile whose message reads "Choose content packs from online services that you use".
- 4.In the Services page, click Google Analytics. In the popup that follows, click Connect.

Connect to Google Analytics



To start using your Google Analytics data in Power BI, follow the prompts below.

Need help connecting? [Learn More](#)

Account

Account Name (case sensitive) configured for your Google Analytics account

Great Outdoors Sandbox Account

Property

Property Name (case sensitive)

Great Outdoors Retail

View

View Name (case sensitive)

All Web Site Data

Next

Cancel

Figure 2.10 As a part of using the Google Analytics content pack, you need to specify the account details.

- 5.In the “Connect to Google Analytics” window, specify the Google Analytics details, including account, property, and view. You can get these details by logging into Google Analytics and navigating to the Administration section (Property Settings page). In my case, I’m using a fictitious company provided by Google for testing purposes (see **Figure 2.10**). Click Next.
- 6.When asked to choose an authentication method, the only option you should see is OAuth. Click Next.
- 7.In the Google confirmation page that follows, click the Allow button to let Power BI connect to Google Analytics. When asked to authenticate, enter your Google account credentials (typically your Gmail account credentials).
- 8.Return to the Power BI portal. If all is well, Power BI should display a popup informing you that it’s importing your content.

Understanding changes

Similar to the Retail Analysis Sample, the Google Analytics content pack installs the following content in Power BI:

- A Google Analytics dataset – A dataset that connects to the Google Analytics data.
- A Google Analytics report – This report has multiple pages to let you analyze site traffic,

system usage, total users, page performance, and top requested pages.

- A Google Analytics dashboard – A dashboard with pinned visualizations from the Google Analytics report.

That's it! After a few clicks and no explicit modeling you now have prepackaged reports and dashboards to analyze your website data! If the included visualizations aren't enough, you can explore the Google Analytics dataset and create your own reports. As I previously mentioned, content packs usually schedule an automatic data refresh to keep your data up to date. To verify:

- 1.In the navigation pane, hover on the Google Analytics dataset and click the “...” button to open the context menu. Notice that the menu informs you when the last refresh happened and when the next one will take place.

Settings

The screenshot shows the 'Settings' interface for a 'Google Analytics' dataset. The top navigation bar includes 'General', 'Dashboards', 'Datasets' (which is selected), and 'Workbooks'. The left sidebar lists 'Google Analytics' and 'Retail Analysis Sample'. The main area is titled 'Settings for Google Analytics' and displays the following configuration:

- Next refresh:** Wed Sep 09 2015 21:17:18 GMT-0400 (Eastern Daylight Time)
- Data Source Credentials:** (indicated by a right-pointing arrow)
- Schedule Refresh:** (indicated by a left-pointing arrow)
- Keep your data up-to-date:** Yes (radio button selected)
- Refresh frequency:** Daily (dropdown menu)
- Time Zone:** (UTC) Coordinated Universal Time (dropdown menu)
- Refresh between:** 12 AM - 6 AM (dropdown menu)
- Note:** To schedule hourly data refresh, try Power BI Pro for 60 days, or contact your Office 365 tenant.
Try Pro for free
- Send refresh failure notification email to me:** (checkbox)
- Refresh History:** (link)
- Buttons:** 'Apply' and 'Discard'

Figure 2.11 The content pack configures automatic daily refresh to synchronize the

imported data with the latest changes in the data source.

- 2.Click the Schedule Refresh link. Notice the content pack has scheduled a daily refresh between 12 AM and 6 AM (see **Figure 2.11**).

NOTE As you can imagine, thousands unattended data refreshes scheduled by many users can be expensive in a multi-tenant environment, such as Power BI. This is why Power BI limits the Power BI Free edition to daily refreshes and you can't specify the exact refresh time. Power BI queues and distributes the refresh jobs using internal rules. However, the Power BI Pro edition allows you to configure more frequent refreshes at specific times during the day.

- 3.(Optional) Expand the Data Source Credentials section and notice that you can change the connectivity details if needed.

2.4.2 Importing Local Files

Another option to get data is to upload a file. Suppose that Maya wants to analyze some sales data given to her as an Excel file. Thanks to the Power BI Get Data feature, Maya can import the Excel file in Power BI and analyze it without creating a model.

NOTE As it stands, Power BI Service limits the size of the imported file to 250 MB. This applies to all files, including Excel files, Excel data models, and Power BI Desktop files, and it's explained in more detail in the blog "The Conceptual Data Model and Limits" by Adam Saxton at <http://blogs.technet.com/b/powerbisupport/archive/2015/08/11/the-conceptual-data-model-and-limits.aspx>. This is still a lot of data but if you find this limiting there are other options. For example, IT can implement an on-premises Tabular model as I discuss in Chapter 10.

Importing Excel data

In this exercise, you will explore the sample dataset that you will analyze later in Power BI. Start by familiarizing yourself with the raw data in the Excel workbook.

- 1.Open the Internet Sales.xlsx workbook in Excel. You can find this file in the \Source\ch02 folder of the source code.
- 2.If Sheet1 isn't selected, click Sheet1 to make it active. Notice that it contains some sales data. Specifically, each row represents the product sales for a given date, as shown in **Figure 2.12**. Also, notice that the Excel data is formatted as a table so that Power BI knows where the data is located.

Date	Product	SalesAmount	OrderQuantity
7/1/2005	Mountain-100 Silver, 44	30599.91	9
7/1/2005	Mountain-100 Black, 38	10124.97	3
7/1/2005	Road-650 Black, 44	1398.1964	2
7/1/2005	Road-650 Red, 52	1398.1964	2
7/1/2005	Road-650 Black, 52	1398.1964	2
7/1/2005	Road-650 Black, 58	2097.2946	3
7/1/2005	Road-150 Red, 56	53674.05	15
7/1/2005	Mountain-100 Black, 48	10124.97	3
7/1/2005	Road-150 Red, 62	78721.94	22
7/1/2005	Mountain-100 Silver, 48	10199.97	3
7/1/2005	Mountain-100 Silver, 42	6799.98	2
7/1/2005	Road-150 Red, 44	82300.21	23
7/1/2005	Road-650 Red, 44	1398.1964	2
7/1/2005	Road-650 Red, 62	699.0932	1

Figure 2.12 The first sheet contains Internet sales data where each row represents the product sales amount and order quantity for a specific date and product.

TIP The Excel file can have multiple sheets with data, and you can import them as separate tables. Currently, the Power BI Get Data feature doesn't allow you to relate multiple tables (you need a data model to do so). In addition, Power BI requires that the Excel data is formatted as a table. You can format tabular Excel data as a table by clicking any cell with data and pressing Ctrl-T. Excel will automatically detect the tabular section. After you confirm, Excel will format the data as a table!

3.Close Excel.

4.Next, you'll import the data from the Internet Sales.xlsx file in Power BI. In Power BI, click the Get Data button.

5.In the Files tile, click the Get button.

6.In the Files page, click "Local File" because you'll be importing from a local Excel file. Navigate to the source code Module2 folder, and then double-click the Internet Sales file.

7.Power BI imports the data from the Excel file into the Power BI Service.

Understanding changes

After you imported the file, the Power BI user interface updates as follows:

- 1.A new dataset Internet Sales is added to the Dataset section. Power BI adds an asterisk icon to the right of the dataset name to show you that this is new content.
- 2.Power BI also places a link to the Internet Sales dataset in the Internet Sales dashboard, as shown in **Figure 2.13**. You can click this link as a shortcut to explore the Internet Sales dataset. Once you add content to the dashboard, you won't need the Internet Sales.xlsx link anymore, so you can delete it if it gets in the way.

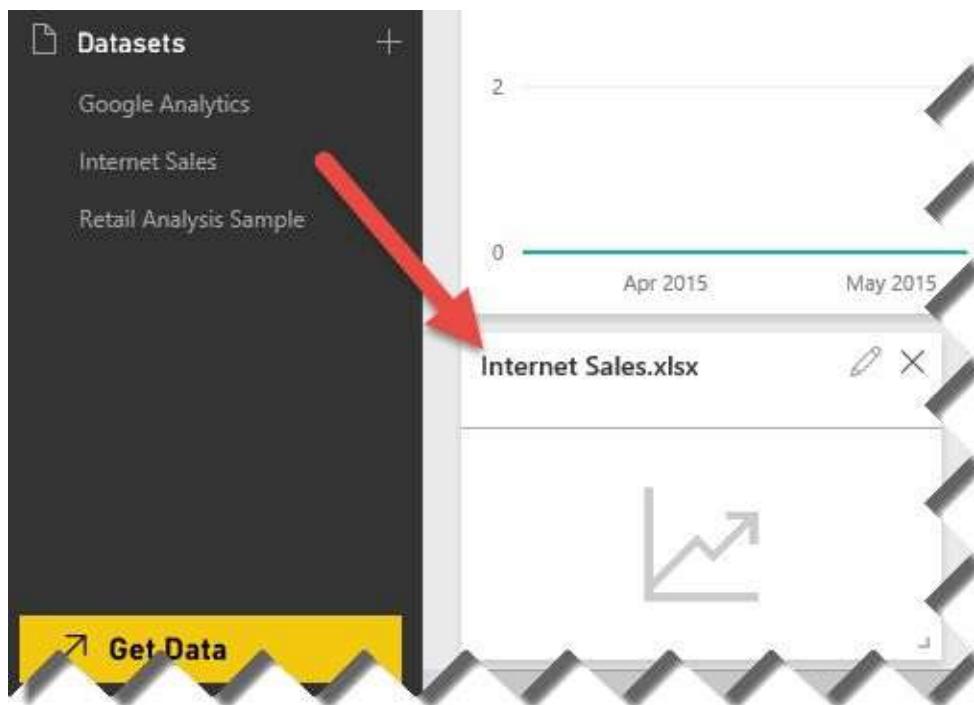


Figure 2.13 The first sheet contains Internet sales data where each row represents the product sales amount and order quantity for given date and product.

3.(Optional) In the Dataset section of the navigation bar, click the Internet Sales dataset to explore it. Alternatively, you can open the dataset context menu and click Explore. Notice

that this action creates an empty report. The Fields pane shows a single table (Internet Sales) whose fields correspond to the columns in the original Excel table. From here, you can just select which fields you want to see on the report.

TIP As I mentioned previously, Power BI can't refresh local Excel files. Suppose that Maya receives an updated Excel file on a regular basis. Without the ability to schedule an automatic refresh, she needs to delete the old dataset (which will delete the dependent reports and dashboard tiles), reimport the data, and recreate the reports. As you can imagine, this can get tedious. A better option would be to save the Excel file to OneDrive or OneDrive for Business. Power BI refreshes files saved to OneDrive every hour and whenever it detects that the file is updated.

2.4.3 Using Live Connections

Suppose that Adventure Works has implemented an organizational Analysis Services semantic model on top of the corporate data warehouse. In the next exercise, you'll see how easy is for Maya to connect to the model and analyze its data.

Understanding prerequisites

Power BI requires special connectivity software, called Power BI Analysis Services Connector, to be installed on an on-premises computer so that it can connect to Analysis Services. This step needs to be performed by IT because it requires admin rights to Analysis Services. I provide step-by-step setup instructions to install and configure the connector in Chapter 9 of this book.

Besides setting up the connector, to perform this exercise, you'll need help from IT to install the sample Adventure Works database and Tabular model (as per the instructions in the book front matter) and to give you permissions to access the Adventure Works Tabular model. In addition, you'll need a Power BI Pro subscription because Power BI Free doesn't allow you to connect to live data sources. If you don't have Power BI Pro, you can get a free 60-day trial.

Connecting to on-premises Analysis Services

Once the Analysis Services connector is set up, connecting to the Adventure Works Tabular model is easy.

- 1.In the Power BI portal, click Get Data.
- 2.In the Get Data page, click the Get button in the Databases pane that reads “Connect to live data in Azure SQL Database and more.”
- 3.In the Databases & More page (see **Figure 2.14**), click the SQL Server Analysis Services tile. In the popup that follows, click Connect. If you don't have a Power BI Pro subscription, this is when you'll be prompted to start a free trial.



Figure 2.14 Use the SQL Server Analysis Services tile to create a live connection to an on-premises SSAS model.

- 4.In the SQL Server Analysis Services page that follows, you should see the name of your Analysis Services connector. Please check with your IT department which one you should use. Once you know the name, click it to select it.
- 5.Power BI verifies connectivity. If something goes wrong, you'll see an error message. Otherwise, you should see a list of the models and perspectives that you have access to. Select the “Adventure Works Tabular Model SQL 2012 – Model” item and click Connect.

This action adds a new dataset to the Datasets section of the navigation bar. You can click the new dataset to explore it. The Fields lists will show all the entities defined in the SSAS models. From here, you can create an interactive report by selecting specific fields from the Fields pane. This isn't much different from creating Excel reports that are connected to an organizational data model.

2.5 Summary

Self-service BI broadens the reach of BI and enables business users to create their own solutions for data analysis and reporting. By now you should view self-service BI not as a competing technology but as a complementing technology to organizational BI.

Power BI is a cloud service for data analytics and you interact with it using the Power BI portal. The portal allows you to create datasets that connect to your data. You can either import data or you can connect live to data sources that support live connections. Once you create a dataset, you can explore it to create new reports. And once you have reports, you can pin their visualizations to dashboards.

As a business user, you don't have to create data models to meet simple data analytics needs. This chapter walked you through a practice that demonstrated how you can perform basic data connectivity tasks, including using a content pack to connect to an online service (Google Analytics), importing an Excel file, and connecting live to an on-premises Analysis Services model. The next chapter will show you how you can analyze your data by creating insightful reports and dashboards!

Chapter 3

Visualizing Data

In the previous chapter, I showed you how Power BI allows business users to connect to data without explicit modeling. The next logical step is to visualize the data so that you can derive knowledge from it. Fortunately, Power BI lets you create meaningful reports with just a few mouse clicks. And Power BI dashboards summarize important metrics so that you can get a quick high-level view of how your business is doing at a glance.

I'll start this chapter by explaining the building blocks of Power BI reports. Then I'll walk you through the steps to explore Power BI datasets and to create reports with interactive visualizations. Next, I'll discuss the anatomy of a Power BI dashboard. I'll also walk you through different ways to create a dashboard, including pinning visualizations and using natural queries by typing them in the Q&A box or by using the Cortana digital assistant. Because this chapter builds on the previous one, make sure you've completed the exercises in the previous chapter to install the Retail Analysis Sample and to import the Internet Sales dataset from the Excel file.

3.1 Understanding Reports

In the previous chapter, I introduced you to Power BI reports. I defined a Power BI report as a visual representation of a dataset. As it stands, Power BI supports two report types:

- Power BI native reports – This report type is a highly visual and interactive report that has its roots in Power View. This is the report type I'll mean when I mention about Power BI reports. For example, the Retail Analysis Sample report is an example of a Power BI report. You can use Power BI Service, Power BI Desktop, and Excel to create these reports.
- Excel reports – Power BI Service allows you to connect to Excel 2013 (or later) files and view the embedded pivot and Power View reports. For example, you might have invested significant effort into creating Power Pivot models and reports. You don't want to migrate them to Power BI Desktop or Tabular, but you'd like to use the Power BI Mobile capabilities so that managers can view and interact with these reports on mobile devices. To get this to work, you can just connect Power BI to your Excel files. However, you must save the Excel files to OneDrive for Business. In addition, you still must use Excel Desktop to create or modify the reports and data model (if the Excel file has a Power Pivot model).

NOTE At the PASS Summit in October 2015, Microsoft announced plans to integrate Power BI with SQL Server 2016 Reporting Services reports. When this feature is released, you'll be able also to add SSRS organizational reports to Power BI.

Most of this chapter will be focused on Power BI native reports but I'll also show you how Power BI integrates with Excel reports.

3.1.1 Understanding Reading View

Power BI supports two report viewing modes. Reading View allows you to explore the report and interact with it, without worrying that you'll break something. Editing View lets you make changes to the report layout, such as to add or remove a field.

Opening a report in Reading View

Power BI defaults to Reading View when you open a report. This happens when you click the report name in the navigation pane or when you click a dashboard tile to open the underlying report.

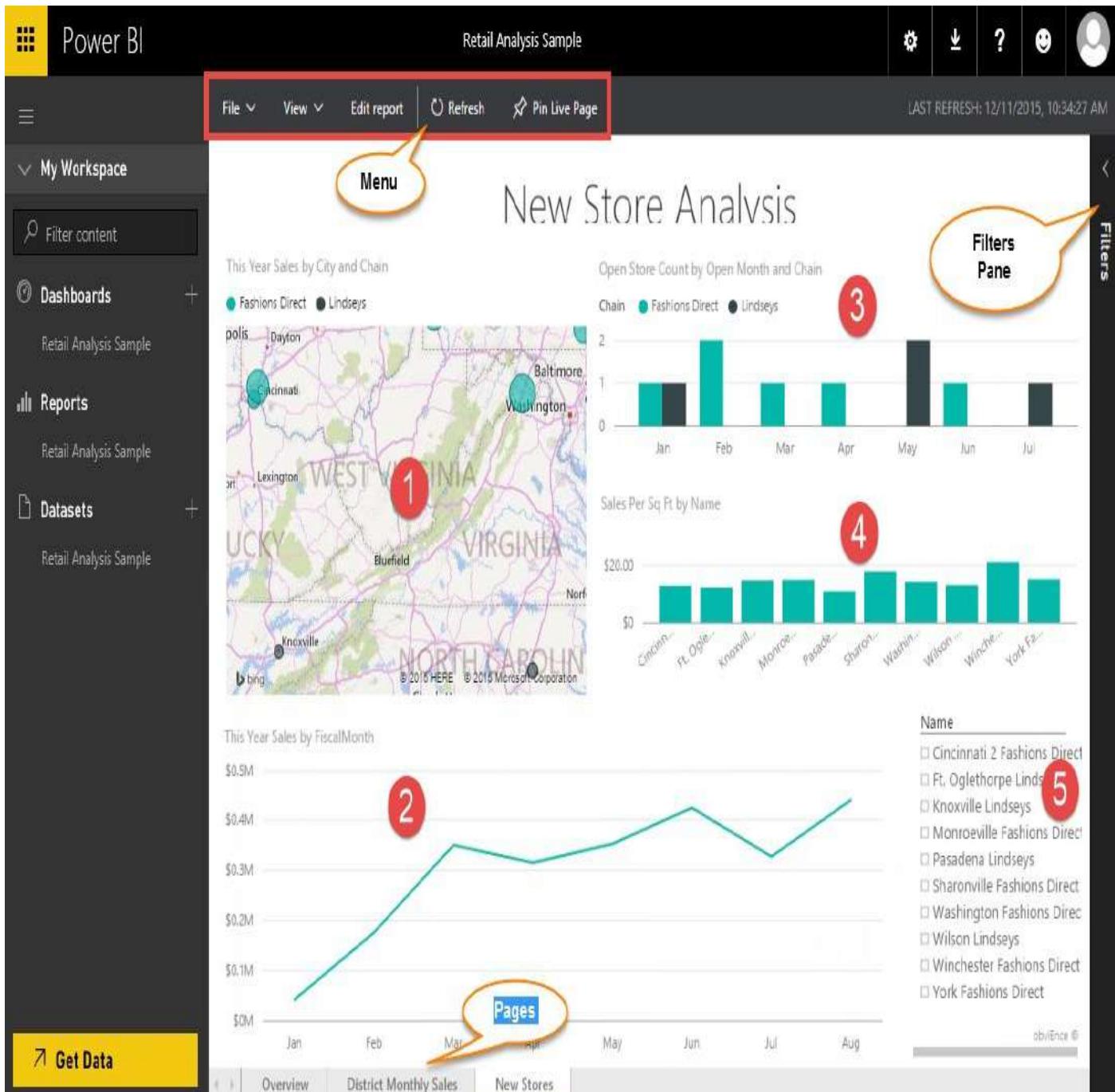


Figure 3.1 Reading View allows you to analyze and interact with the report, without changing it.

- 1.In the Power BI portal, click the Retail Analysis Sample report that you can find under the Reports section in the left navigation bar.
- 2.On the bottom-left area of the report, notice that this report has three pages. A report page is conceptually similar to a slide in a PowerPoint presentation – it gives you a different view of the data story. Click the “New Stores” page to activate it. Notice that the page has five visualizations (see **Figure 3.1**).

Power BI opens the report in Reading View. The top menu allows you to save a copy of the report (File menu), adjust the report presentation (View menu), such as to fit the report to your browser page width, switch to Editing View to edit the report layout (Edit Report menu), refresh the data on the report (Refresh menu), and pin the entire page to a dashboard (Pin Live Page menu). The “Last Refresh” status on the right shows the last

time the underlying dataset was refreshed. In other words, it shows you how current is the data on the report.

NOTE The report always queries the underlying dataset when you view it. The report Refresh menu could be useful if the underlying dataset was refreshed or has a live connection and you want to get the latest data without closing and reopening the report.

3.Expand the Filters pane on the right of the report. Notice that the report author has added a Store Type filter that filters only new stores. Note also that you can change the filter, such as to show all store types.

Interacting with the report

Although the name might mislead you, Reading View allows you to interact with the report.

1.In the fourth visualization (“Sales Per Sq Ft by Name”), click the first column “Cincinnati 2 Fashions Direct” (you can hover on the column bar and a tooltip pops up to show the full name). Notice that the other visualizations change to show data only for the selected store. This feature is called *interactive highlighting*, and it’s another way to filter data on the report. Interactive highlighting is automatic, and you don’t need to do anything special to enable it. Click the column again or an empty area in the same chart to show all the data.

2.Hover on the same visualization and notice that Power BI shows an ellipsis menu in the top-right corner (see **Figure 3.2**).



Figure 3.2 You can see how the visualization is sorted and change or remove the sort if needed.

3.Click the “Sort By” menu; notice that the report author has sorted the visualization by the “Sales Per Sq Ft” measure. You can change or remove the sort (the sort is active if there is an orange bar to the left of it). Notice the pin button to the left of the ellipsis menu. It allows you to pin the visualization to a dashboard.

4.On the right of the pin button, there is another button that lets you pop out the visualization in an “in-focus” mode in case you want to examine its data in more detail. Try it out.

If the visualization is designed for drilling down (the visualizations on this report are not) you’ll also see drilling up and down indicators (shown by the two red circles at the top of **Figure 3.3**).

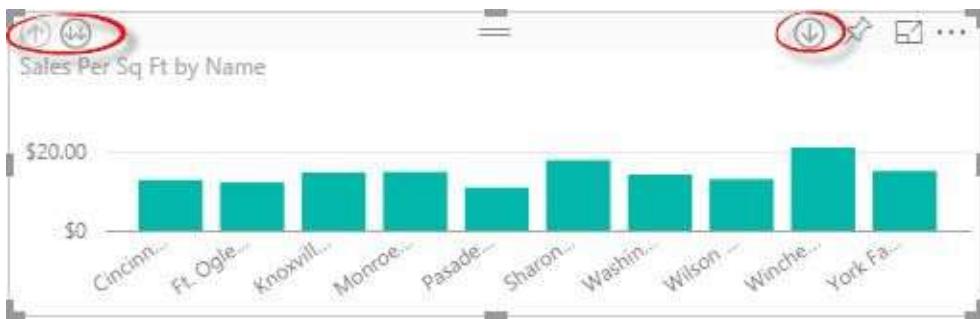


Figure 3.3 You can drill down the next level if the visualization is designed for this feature.

Because, by default, Power BI initiates interactive highlighting when you click a chart element, the indicators allow you to switch to a drill mode. For example, you can click the down arrow indicator (in the top-right corner) to switch to a drill mode, and then click a column to drill down. To drill up, just click the up arrow indicator in the top-left corner. And to drill down all data points click the double-arrow indicator.

TIP Some visualizations, such as Column Chart allow you to add multiple fields to specific areas when you configure the chart, such as the Axis the area. So to configure a chart for drilling, you need to open the report in Editing View and just add more fields to the Axis area of the chart. These fields define the levels that you drill down to when you click the down-arrow indicator.

5.Hover on top of any column in the chart. Notice that a tooltip pops up to let you know the data series name and the exact measure amount.

3.1.2 Understanding Editing View

If you have report editing rights, you can make changes to the report layout. You have editing rights when you're the original report author, when the report is available in a workspace you're a member of and you have rights to edit the content, or when you take ownership of the report, such as when you create a copy of a report that was distributed via an organizational content pack. You can switch to Editing View by clicking the Edit Report menu.

Understanding the Editing View menu

One of the first thing you'll notice when you switch to Editing View is that the report menu changes (see **Figure 3.4**).



Figure 3.4 The Editing View menu adds links to create text elements and save the report.

The File menu adds a Save submenu to let you save changes to the report. The View menu lets you resize the report in different modes: dynamic, fit to page, fit to width, and actual size. The Reading View menu brings you back to opening the report as read-only.

Editing View also adds menus on the right side of the report. Use the Text Box menu to add text boxes to the report which could be useful for report or section titles, or for any text you want on the report. The Text Box menu opens a comprehensive text editor that allows you to format the text and implement hyperlinks. The Shapes menu allows you to add rectangle, oval, line, triangle, and arrow shapes to the report for decorative or illustrative purposes.

The Visual Interactions (also known as “Brushing and Linking”) allows you to customize the behavior of the page interactive features. You can select a visual that would act as the source of brushing and then set the interactivity level for the other visualizations on the same page. For example, you can use this feature to disable interactive highlighting to selected visualizations. To see this feature in action, watch the “Power BI Desktop November 2015 Update” video at <https://youtu.be/ErHvpkyQjSg>.

The Refresh menu behaves the same as its Reading View counterpart. The Duplicate Page menu creates a copy of the current report page. This could be useful if you want to add a new page to the report that has similar visualizations as an existing page but you want to show different data. The Save menu is a shortcut that does the same thing as the File ð Save menu. And, as in Reading View, the “Pin Live Page” menu pins the current report page to a dashboard.

Understanding the Visualizations pane

The next thing you’ll notice is that Editing View adds two panes on the right of the report: Visualizations and Fields. Use the Visualizations pane to configure the active visualization, such as to switch from one chart type to another.

NOTE When you make changes to the Visualizations pane, they are applied to the currently selected (active)

visualization. An active visualization has a border around it with resize handles. You need to click a visualization to activate it.

- 1.If it's not already active, click the "New Stores Analysis" page to select it.
- 2.Click the "Sales Per Sq Ft and Last Year Sales by Name" visualization to activate it.
Figure 3.5 shows the Visualizations pane. The Filters section actually occupies the bottom part of the Visualizations pane but is shown adjacent to it on the screenshot to accommodate space constraints.

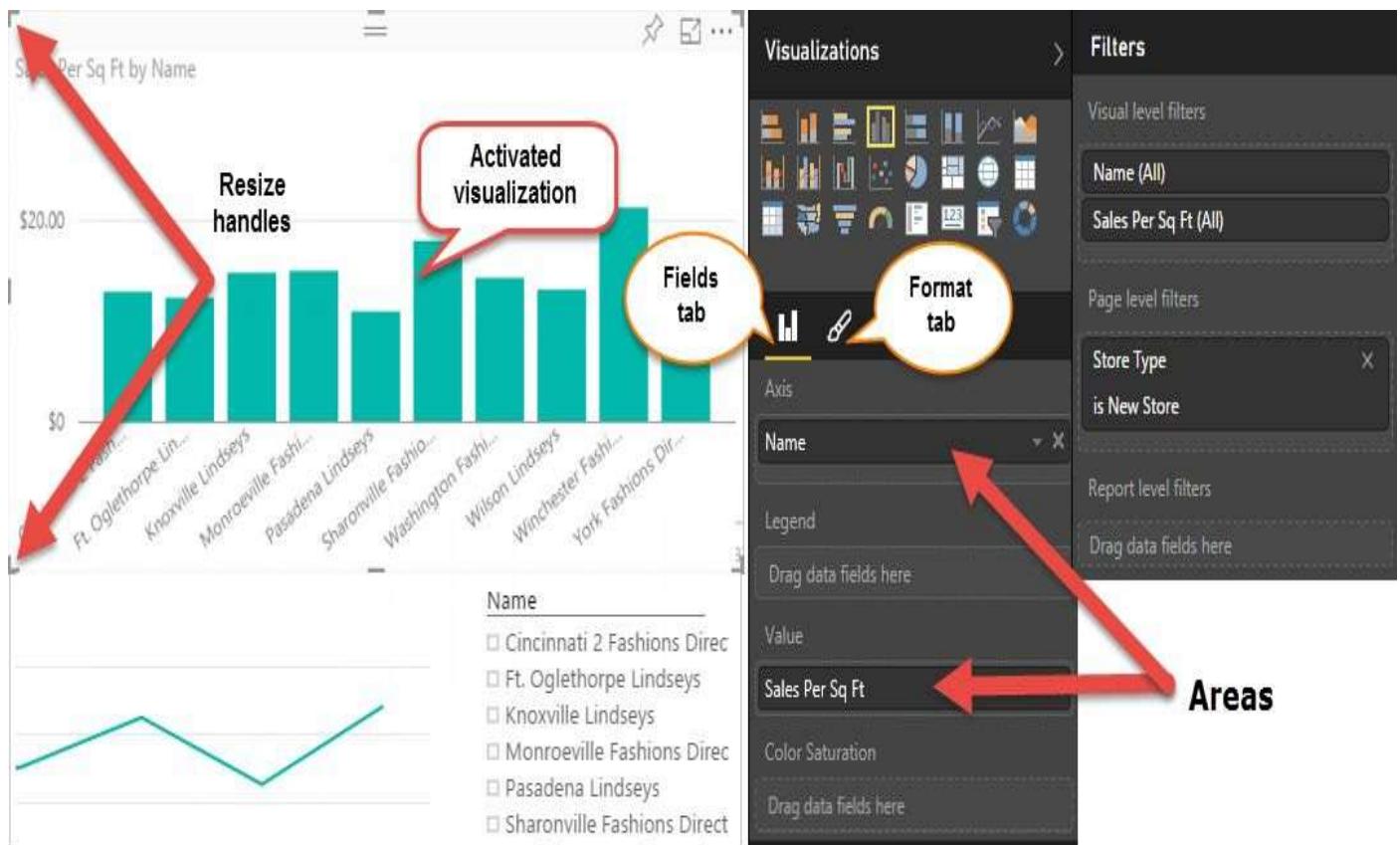


Figure 3.5 The Visualizations pane allows you to switch visualizations and to make changes to the active visualization.

The Visualizations pane consists of several sections. The first section shows the Power BI visualization types, which I'll discuss in more details in the next section "Understanding Power BI Visualizations". The ellipsis button below the visualizations allows you to import custom visuals from the Power BI visuals gallery (<http://visuals.powerbi.com>) that are contributed by Microsoft and the community. So, when the Power BI-provided visualizations are not enough for your data presentation needs, check the gallery. Chances are that you'll find a custom visual that can fill in the gap!

The Fields tab consists of areas that you can use to configure the active visualization, similarly to how you would use the zones of the Excel Fields List when you configure a pivot report. For example, this visualization has the Name field of the Store table added to the Axis area and the "Sales Per Sq Ft" field from the Sales table added to the Value area.

TIP You can find which table a field comes from by hovering on the field name. You'll see a tooltip pop up that shows the table and field names, such as 'Store'[Name]. This is the same naming convention that a data analyst would use to create custom calculations in a data model using Data Analysis Expressions (DAX).

The Filters section defines visual-level and page filters. Use the "Visual level filters"

section to filter the data in the active visualization. By default, you can filter any field that's used in the visualization, but you can also add other fields. For example, the “Visual level filters” has the Name and “Sales per Sq Ft” fields because they are used on the chart. The (All) suffix next to the field tells you that these two fields are not filtered (the chart shows all stores irrespective of their sales). Use the “Page Level Filters” section to apply filters to all visualizations on the active page. For example, all four visualizations on this page are filtered to show data for new stores (Store Type is “New Store”). Finally, filters in the “Report Level Filters” section are applied globally to all visualizations on the report even if they are on different pages.

The Format tab of the Visualizations pane is for applying format settings to the active visualization. Different visualizations support different format settings. For example, column charts support custom colors, data labels, title, axis labels, and other settings. As Power BI evolves, you can expect more options to give you more control over customizing the visual appearance.

NOTE Do you think that Power BI visualizations are rather basic and they don't support enough customization compared to other tools, such as Excel? You might not have to wait for long. Remember from Chapter 1 that Microsoft committed to a weekly release cadence so features should be coming fast. But to prioritize your feature wish, I encourage you to submit your idea or vote for an existing feature at <https://support.powerbi.com>. If you don't want to wait, a web developer in your organization with some coding wizardry can customize the Power BI visualizations because their code is open source. Chapter 13 shows how this can be done.

Understanding the Fields pane

The Fields pane shows the tables in your dataset. When implementing the Retail Analysis Sample, the author implemented a self-service data model by importing several tables. By examining the Fields pane, you can see these tables and their fields (see **Figure 3.6**). For example, the Fields pane shows District, Item, Sales, and Store tables. The Store table is expanded and you see some of its fields, such as Average Selling Area Size, Chain, City, and so on.

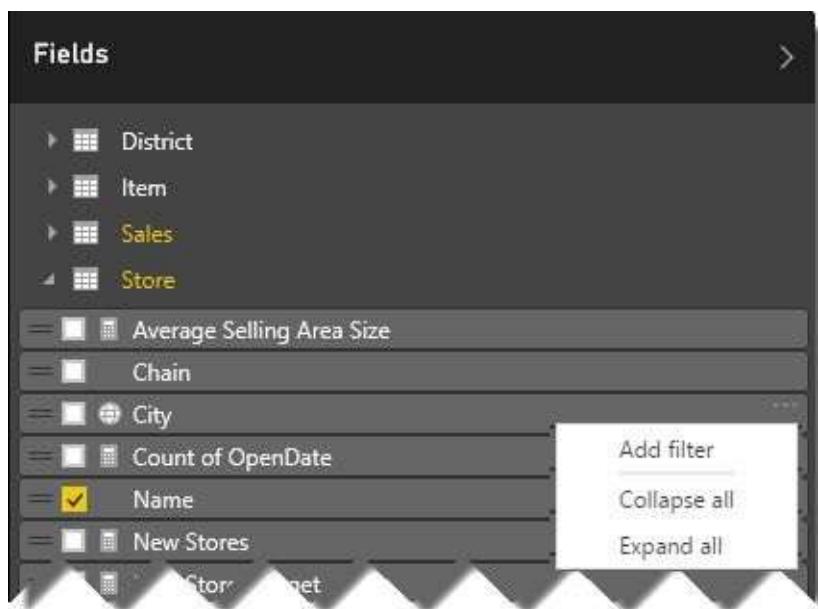


Figure 3.6 The Fields pane shows the dataset tables and fields.

Although you can't preview the data in the Fields pane, Power BI gives you clues about

the field content. For example, if the field is prefixed with a calculator icon , such as the “Average Selling Area Size” field, it’s a calculated field that uses a formula. Fields prefixed with a globe icon  are geography-related fields, such as City, that can be visualized on a map. If the field is checked, it’s used in the selected visualization. If the name of the table has an orange color, one or more of its fields are used in the selected visualization. For example, the Sales and Store tables are orange because they each have at least one field used in the selected visualization.

Each field has an ellipsis menu to the right of the field that allows you to add the field as a filter. If you have selected a visualization, the field will be added as a visual-level filter. For example, if you select a chart on the report add the City field as a filter, you can filter the chart data by city, such as to show data for Atlanta only. If no visualization is selected, the field will be added as a page-level filter. For example, you add the City field as a filter but you haven’t selected a specific visualization, you can filter all the visualizations on the page by this field. The “Collapse All” option collapses all the fields so you can see only the table names in the Fields list. And “Expand All” expands all tables so that you can see their fields.

Working with fields

Fields are the building blocks of reports because they define what data is shown. In the process of creating a report, you add fields from the Fields pane to the report. There are several ways to do this:

- Drag a field on the report – If you drag the field to an empty area on the report canvas, you’ll create a new visualization that uses that field. If you drag it to an existing visualization, Power BI will add it to one of the areas of the Visualizations pane.

NOTE Power BI always attempts to determine the right default. For example, if you drag the City field to an empty area, I’ll create a map visualization because City is a geospatial field. If you drag a field to an existing visualization, Power BI will attempt to guess how to use it best. For example, assuming you want to aggregate a numeric field, it’ll add it to the Value area.

- Check the field’s checkbox – It accomplishes the same result as dragging a field.
- Drag a field to an area – Instead of relying on Power BI to infer what you want to do with the field, you can drag and drop a field into a specific area of the Fields tab in the Visualizations pane. For example, if you want a chart to create a data series using the “Sales per Sq Ft” field, you can drag this field to the Value area of the Fields tab in the Visualizations pane (see again **Figure 3.5**).

Similarly, to remove a field, you can uncheck its checkbox in the Fields pane. Or, you can drag the field away from the Visualizations pane to the Fields pane. If the field ends up in the wrong area of the Visualizations pane, you can drag it away from it and drop it on the correct area.

TIP Besides dragging a field to an empty area, you can create a new visualization by just clicking the desired visualization type in the Visualizations pane. This adds an empty visualization to the report area. Then, you can drag and drop the required fields to bind it to data.

3.1.3 Understanding Power BI Visualizations

You use visualizations to help you analyze your data in the most intuitive way. Power BI supports various common visualizations and their number is likely to grow in time. And because Power BI supports custom visuals, you'll be hard pressed not to find a suitable way to present your data. Let's take a look at the Power BI-provided visualizations.

Column and Bar charts

Power BI includes the most common charts, including Column Chart, Bar Chart, and other variants, such as Clustered Column Chart, Clustered Bar Chart, 100% Stacked Bar Chart, and 100% Stacked Column Chart (see **Figure 3.7**). The difference between column and bar charts is that the Bar Chart displays a series as a set of horizontal bars. In fact, the Bar Chart is the only chart type that displays data horizontally by inverting the axes, so the x-axis shows the chart values and the y-axis shows the category values.

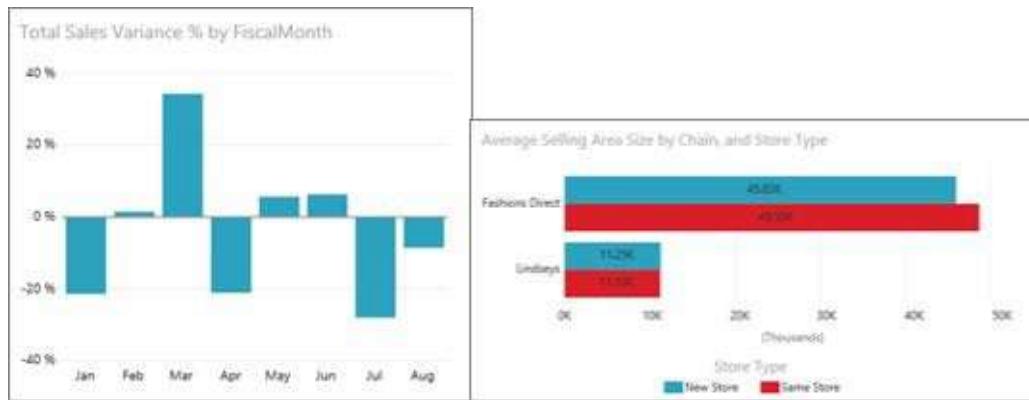


Figure 3.7 Column and bar charts display data points as bars.

Line charts

Line charts are best suited to display linear data. Power BI supports basic line charts and area charts, as shown in **Figure 3.8**. Similar to a Line Chart, an Area Chart displays a series as a set of points connected by a line with the exception that all of the area below the line is filled in. The Line Chart and Area Chart are the only chart types that display data contiguously. They're commonly used to represent data that occurs over a continuous period of time.

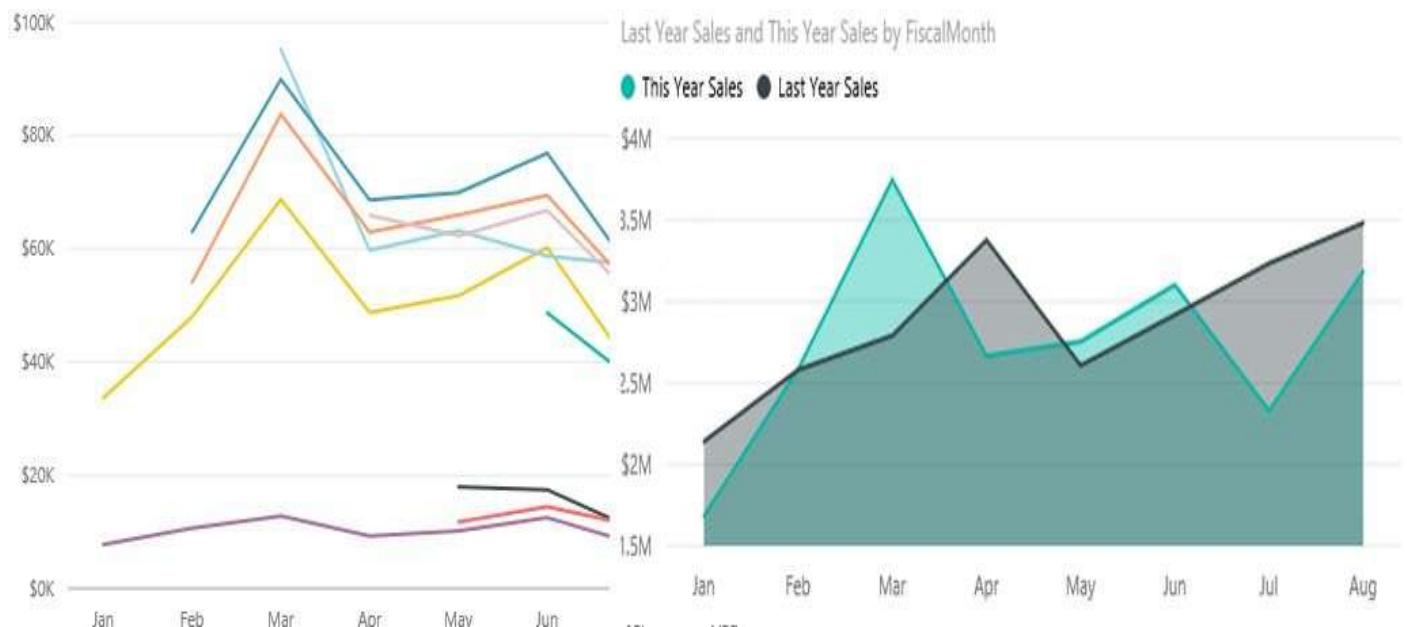


Figure 3.8 Power BI supports line charts and area charts.

Combination Chart

The Combination (combo) Chart combines a Column Chart and a Line Chart. This chart type is useful when you want to display measures on different axes, such as sales on the left Y-axis and order quantity on the right Y-axis. In such cases, displaying measures on the same axis would probably be meaningless if their units are different. Instead, you should use a Combination Chart and plot one of the measures as a Column Chart and the other as a Line Chart, as shown in **Figure 3.9**.



Figure 3.9 A Combo Chart allows you to plot measures on different axes. In this example, the This Year Sales and Last Year Sales measures are plotted on the left Y-axis while Store Count is plotted on the right Y-axis.

Scatter Chart

The Scatter Chart (**Figure 3.10**) is useful when you want to analyze correlation between two variables. Suppose that you want to find a correlation between “Sales Per Square Ft” and “Total Sales Variance”. You can use a scatter chart to show “Sales Per Square Foot” along the y-axis and “Total Sales Variance” along the x-axis. The resulting chart helps you understand if the two variables are related and, if so how. For example, you can determine if these two measures have a linear relationship; when the “Sales Per Sq Ft” increases, ”Total Sales Variance” increases as well.

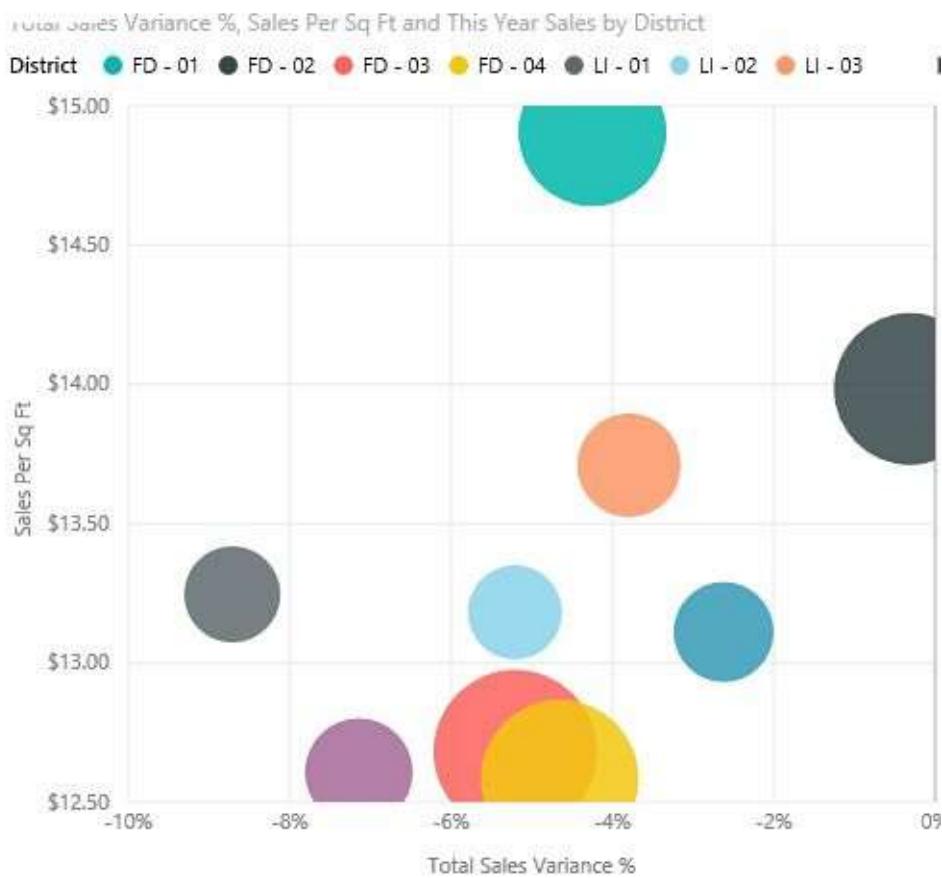


Figure 3.10 Use a Scatter Chart to analyze correlation between two variables.

Shape charts

Shape charts are commonly used to display values as percentages of a whole. Categories are represented by individual segments of the shape. The size of the segment is determined by its contribution. This makes a shape chart useful for proportional comparison between category values. Shape chart variations include Pie, Doughnut, and Funnel charts, as shown in **Figure 3.11**. All shape charts display each group as a slice on the chart. The Funnel Chart order categories from largest to smallest.

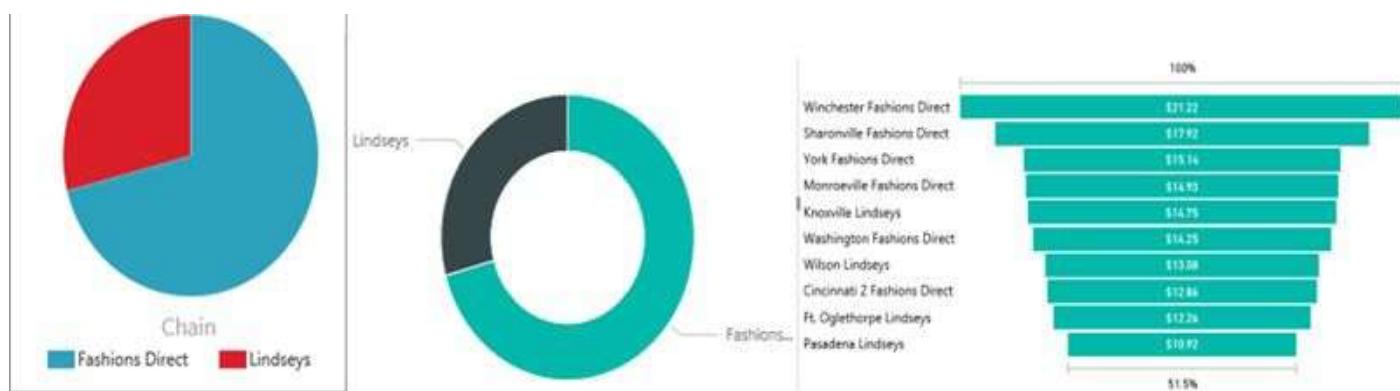


Figure 3.11 Pie, Doughnut, and Funnel charts can be used to display values as percentages of a whole.

Treemap Chart

A treemap is a hierarchical view of data. It breaks an area into rectangles representing branches of a tree. Consider the Treemap Chart when you have to display large amounts of hierarchical data that doesn't fit in column or bar charts, such as the popularity of product

features. Power BI allows you to specify custom colors for the minimum and maximum values. For example, the chart shown in **Figure 3.12** uses a red color for show stores with less sales and a green color to show stores with the most sales.



Figure 3.12 Consider Treemap to display large amounts of hierarchical data that doesn't fit in Column or Bar charts.

Table and Matrix visualizations

Use the Table and Matrix visualizations to display text data as tabular or crosstab reports. The Table visualization (left screenshot in **Figure 3.13**) display text data in a tabular format, such as the store name and sales as separate columns.

Name	This Year Sales	Name	Jan	Feb	Mar	Apr	May
Akron Fashions Direct	\$453,623	Akron Fashions Direct	\$39,691	\$52,016	\$80,023	\$54,942	\$62,699
Alexandria Lindseys	\$141,971	Alexandria Lindseys	\$12,210	\$14,297	\$23,529	\$16,089	\$17,231
Altoona Fashions Direct	\$410,079	Altoona Fashions Direct	\$37,152	\$48,926	\$78,162	\$45,465	\$49,206
Anderson Lindseys	\$111,359	Anderson Lindseys	\$8,623	\$17,745	\$20,430	\$13,908	\$11,499
Asheville Lindseys	\$112,270	Asheville Lindseys	\$9,498	\$12,911	\$19,364	\$14,536	\$14,349
Athens Lindseys	\$90,303	Athens Lindseys	\$7,364	\$13,219	\$15,411	\$11,301	\$10,420
Augusta Lindseys	\$88,351	Augusta Lindseys	\$6,464	\$11,200	\$17,093	\$11,178	\$9,021
Total	\$22,051,952	Beavercreek Fashions Direct	\$29,639	\$46,539	\$70,320	\$44,342	\$46,614

Figure 3.13 Use Table and Matrix visualizations for tabular and crosstab text reports.

The Matrix visualization (right screenshot in **Figure 3.13**) allows you to pivot data by one or more columns added to the Columns area of the Visualization pane, in order to create crosstab reports. Both visualizations support interactive sorting by clicking a column header, such as to sort stores in an ascending order by name.

Map visualizations

Use map visualizations to illustrate geospatial data. Power BI includes two map visualizations (see **Figure 3.14**)

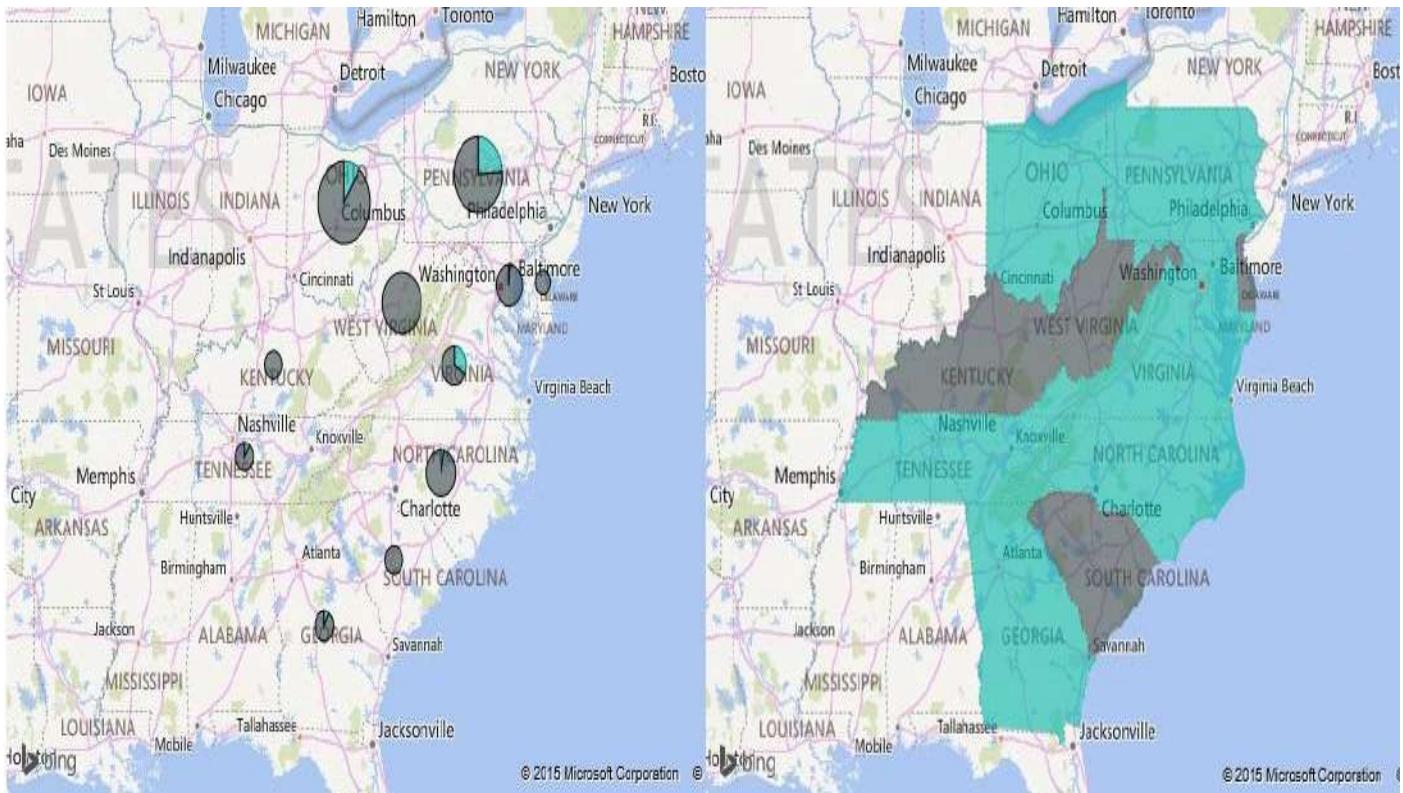


Figure 3.14 Power BI supports basic maps and filled maps.

You can use a Basic Map (left screenshot in **Figure 3.14**) to display categorical and quantitative information with spatial locations. Adding locations and fields places dots on the map. The larger the value, the bigger the dot. When you add a field to the Legend area of the Visualization pane, the Basic Map shows pie charts on the map, where the segments of the chart correspond to the field's values. For example, each Pie Chart in the Basic Map on left of **Figure 3.14** breaks the sales by the store type.

As the name suggests, the Filled (choropleth) Map (right screenshot in **Figure 3.14**) fills geospatial areas, such as USA states. This visualization can use shading or patterns to display how a value differs in proportion across a geography or region.

Both map types are powered by Bing maps. Because of this, you need a connection to the Internet for the maps to work. You can zoom in and out interactively, by pressing the Ctrl key and using the mouse wheel. Besides being able to plot precise locations (latitude and longitude), they can infer locations using a process called geo-coding, such as to plot addresses.

Gauge visualization

Gauges are typically used on dashboards to display key performance indicators (KPIs), such as to measure actual sales against budget sales. Power BI supports a radial Gauge visual for this purpose (**Figure 3.15**).



Figure 3.15 The Gauge visualization allows you to display progress toward a goal.

This visualization has a circular arc and displays a single value that measures progress toward a goal. The goal, or target value, is represented by the line (pointer). Progress toward that goal is represented by the shaded scale. And the value that represents that progress is shown in bold inside the arc.

Card visualizations

Power View supports Single Card and Multi Row card visualizations, as shown in **Figure 3.16**.

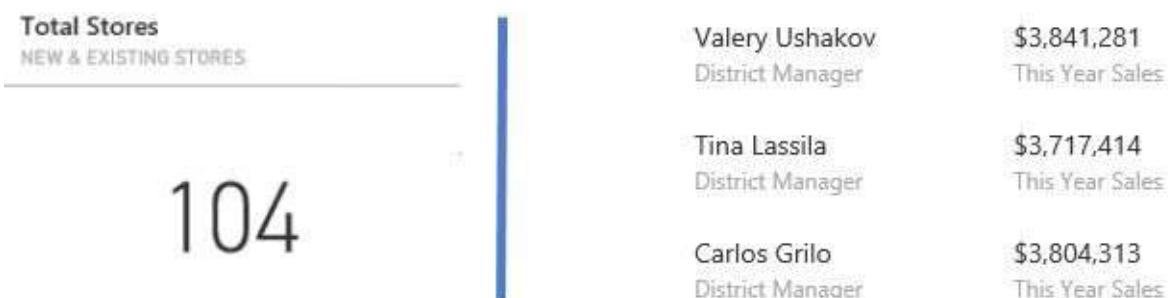


Figure 3.16 The Single Card on the left displays a single value (total stores) while the Multi Row Card displays managers and their sales.

The Single Card visualization (left screenshot in **Figure 3.16**) displays a single value to draw attention to the value. If you use the iPhone Power BI mobile application, you can set up data alerts on single cards, such as to receive a notification when the number of stores exceed a given value.

If you’re looking for another way to visualize tabular data than plain tables, consider the Multi Row Card visualization (right screenshot in **Figure 3.16**). It converts a table to a series of cards that display the data from each row in a card format, like an index card.

Slicer

The Slicer visualization isn’t really meant to visualize data but to filter data (see **Figure 3.17**). Unlike page-level filters, which are found in the Filter pane when the report is displayed in Reading View, the Slicer visualization is added on the report so users can see what’s filtered and interact with the slicer without expanding the Filter pane. As shown in the screenshot, it supports multi-value selection.

District Manager

- Allan Guinot
- Andrew Ma
- Annelie Zubar
- Brad Sutton
- Carlos Grilo
- Chris Gray
- Chris McGurk
- Tina Lassila
- Valery Ushakov

Figure 3.17 Use the Slicer visualization to create a filter that filters all visualizations on the report page.

3.1.4 Understanding Custom Visualizations

No matter how much Microsoft improves the Power BI visualizations, it might never be enough. When it comes to data presentation, beauty is in the eye of the beholder. However, the Power BI presentation framework is open and UX developers can contribute custom visuals that you can use with your reports for free!

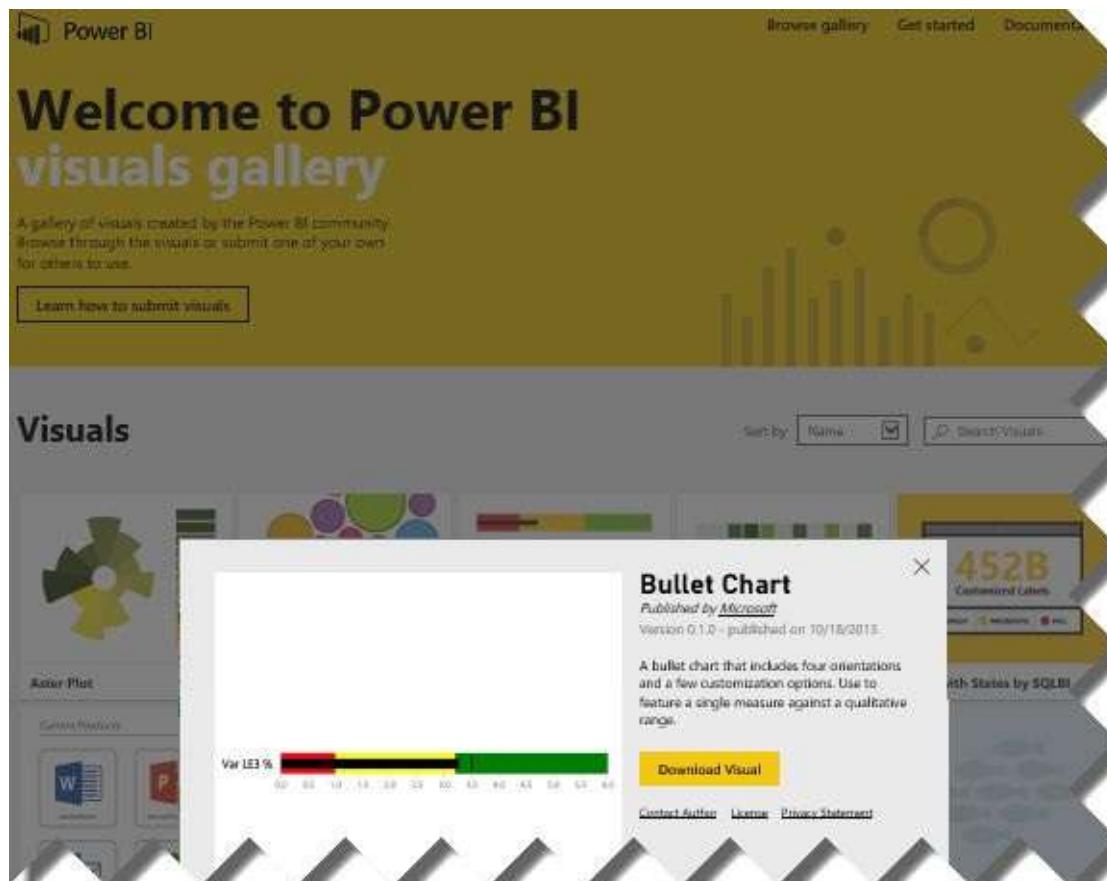


Figure 3.18 Use the Power BI visuals gallery to find and download custom visuals contributed by Microsoft and the community.

Understanding Power BI visuals gallery

To make it easier for users to find a custom visual, Microsoft hosts a Power BI visuals gallery website at <http://visuals.powerbi.com> (see **Figure 3.18**).

The gallery allows you to search and view custom visuals. When you find an interesting visual, click it to see more information about the visual and its author. Custom visuals are contributed by Microsoft and the Power BI community. If you decide to use the visual, click Download Visual to download the visual, and import it using the ellipsis menu (...) in the Visualizations pane. Visuals are distributed as files with the *.pbviz extension.

Using custom visuals

Business users can use custom visuals in Power BI Service and data analysts can do the same in Power BI Desktop. As a first step, you need to import the visual:

- 1.In the Visualizations pane, click the ellipsis (...) button found below the visualization icons.
- 2.Read the prompt that warns you that custom visuals could contain security and privacy risks. Although the Microsoft terms of agreement requires the developer to agree that the code doesn't contain any threats, Microsoft isn't responsible for any damages the code might cause. So use custom visuals at your own risk.

NOTE Custom visuals are written in JavaScript which browsers run in a protected sandbox environment that restricts what the script can do. However, the script is executed on every user who renders a report with a custom visual. When it comes to security you should do your homework to verify the visual origin and safety. If you're unsure, consider involving IT to test the visual with anti-virus software and make sure that it doesn't pose any threats. For more information about how you or IT can test the visual, read the "Review custom visuals for security and privacy" document at <https://support.powerbi.com/knowledgebase/articles/750219>.

- 3.Navigate to the location where you downloaded the custom visual file and import the visual. Power BI adds the visual to the Visualizations pane.



Figure 3.19 A custom visual is added to the Visualizations pane and can be used on reports.

Once you import the visual, you can use it on reports. **Figure 3.19** shows that I imported the Bullet Chart visual and its icon appears at the bottom of the Visualizations pane. Then I added the visual to a report and configured it to show this year sales.

- 4.Save the report. Navigate to another item in the navigation pane and then click the report with a custom visual again. Notice that Power BI displays a warning that this report contains a custom visual. You must click the "Enable custom visuals" button before Power BI renders the report. This is another security measure to warn users about potential threats associated with the custom code. If you don't confirm the prompt, Power BI won't render the visual.

3.2 Working with Reports

Now that you know about visualizations, let's use them on reports. In the first exercise that follows, you'll create a report from scratch. The report will source data from the Internet Sales dataset that you created in Chapter 2. In the second exercise, you'll modify an existing report. Finally, you'll practice working with Excel reports.

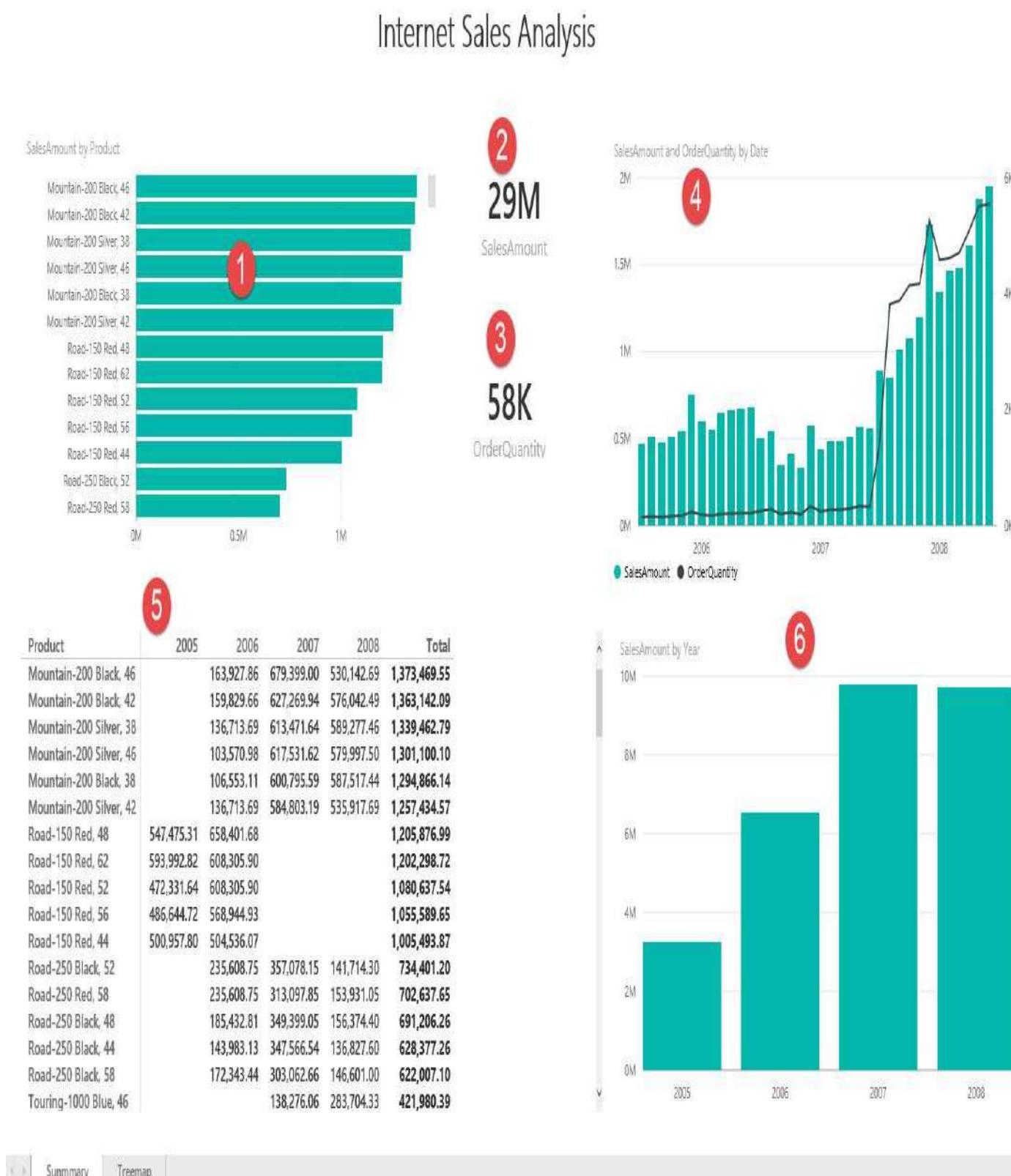


Figure 3.20 The Summary page of the Internet Sales Analysis report includes six

visualizations.

3.2.1 Creating Your First Report

In Chapter 2, you imported the Internet Sales Excel file in Power BI. As a result, Power BI created a dataset with the same name. Let's analyze the sales data by creating the report shown in **Figure 3.20**. This report consists of two pages. The Summary page has six visualizations and the Treemap page (not shown in **Figure 3.20**) uses a Treemap visualization to help you analyze sales by product at a glance. (For an example of a Treemap visualization skip ahead to **Figure 3.22**.)

Getting started with report authoring

One way to create a new report in Power BI is to explore a dataset.

- 1.In the Power BI portal, click the Internet Sales dataset to create a new report that is connected to this dataset. Alternatively, you can click the ellipsis (...) next to the dataset in the left navigation pane, and then click Explore.
- 2.Power BI opens a blank report in Editing View. Click the Text Box menu to create a text box for the report title. Type “*Internet Sales Analysis*” and format as needed. Position the text box on top of the report.
- 3.Note the Fields pane shows only table “Internet Sales” because the Internet Sales dataset, which you imported from an Excel file, has only one table.
- 4.Click the Save menu and save the report as *Internet Sales Analysis*. Remind yourself to save the report (you can press Ctrl-S) every now and then so that you don’t lose changes.

NOTE Power BI times out your session after a certain period of inactivity to conserve resources in a shared environment. When this happens and you return to the browser, it'll ask you to refresh the page. If you have unsaved changes, you might lose them when you refresh the page so get in the habit to press Ctrl-S often.

Creating a Bar Chart

Follow these steps to create a bar chart that shows the top selling products.

1. In the Fields pane, check the SalesAmount field. Power BI defaults to a Column Chart visualization that displays the grand total of the SalesAmount field.
- 1.Check the Product field. Power BI adds it to the Axis area of the chart.
- 2.In the Visualizations pane, click the Bar Chart icon to flip the Column Chart to a Bar Chart. Power BI sorts the bar chart by the product name in an ascending order.
- 3.Point your mouse cursor to the top-right corner of the chart. Click the ellipsis “...” menu and change the sorting order to sort by SalesAmount in a descending order. Compare your results with the “SalesAmount by Product” visualization in upper left of **Figure 3.20**.

TIP Clicked the wrong button or menu? Don't worry, you can undo your last step by pressing Ctrl-Z. To undo multiple steps in a reverse order, press Ctrl-Z repeatedly.

Adding Card visualizations

Let's show the total sales amount and order quantity as separate card visualizations (items 2 and 3 in **Figure 3.20**) to draw attention to them:

1. Click an empty space on the report canvas outside the Bar Chart to deactivate it.

TIP As I explained, another way to create a new visualization is to drag a field to an empty space on the canvas. If the field is numeric, Power BI will create a Column Chart. For text fields, it'll default to a Table.

- 1.In the Field list, check the SalesAmount field. Change the visualization to Card. Position it as needed.
- 2.Repeat the last two steps to create a new card visualization using the OrderQuantity field.

Creating a Combo Chart visualization

The fourth chart in **Figure 3.20** shows how the sales amount and order quantity change over time:

1. Drag the SalesAmount field and drop it onto an empty area next to the card visualizations to create a Column Chart.
- 1.Drag the Date field and drop it onto the new chart.
- 2.Switch the visualization to “Line and Stacked Column Chart”. This adds a new Line Values area to the Visualizations pane.
- 3.Drag the OrderQuantity field and drop it on the Line Values area. Power BI adds a line chart to the visualization and plots its values to a secondary Y-axis. Compare your results with the “SalesAmount and OrderQuantity by Date” visualization (item 4 in **Figure 3.20**).

Creating a Matrix visualization

The fifth visualization (from **Figure 3.20**) represent a crosstab report showing sales by product on rows and years on columns. Let's build this with the Matrix visualization:

1. Drag the SalesAmount field and drop it onto an empty space on the report canvas to create a new visualization. Change the visualization to Matrix.
- 1.Check the Product field to add it to the visualization on rows.
- 2.Drag the Year field and drop it on the Columns zone to pivot on Year on columns.
- 3.Resize the visualization as needed. As an optional step, click any of the column headers to sort the visualization interactively in an ascending or descending order.

Creating a Column Chart visualization

The sixth visualization listed shows sales by year:

1. Create a new visualization that uses the SalesAmount field. Power BI should default to Column Chart.
- 1.In the Fields pane, check the Year field to place it in the Axis area of the Column Chart.

Filtering the report

Next, you'll implement page-level and visual-level filters. Let's start by creating a page-level Date filter that will allow you to filter all visualizations on the page by date.

1. Click an empty area on the report canvas to make sure that no visualization is activated.
- 1.Drag the Date field onto the Page Level Filters area. This creates a page-level filter that filters all visualizations on the activated page.
- 2.Practice different ways to filter. For example, switch to Advanced Filtering mode and

filter out dates after June 30th, 2008, as shown on the left screenshot in **Figure 3.21**.

3.To work with visual-level filters, click the fifth (Matrix) visualization. To practice another way to create a filter besides drag and drop, hover on the Product field in the Fields pane. Then expand the ellipsis menu and click Add Filter.

Because there's an activated visualization, this action configures a visual-level filter. Notice that the Visual Level Filters (see the right screenshot in **Figure 3.21**) already includes the three fields used in the visualization so that you can filter on these fields without explicitly adding them as filters.

The image shows two side-by-side screenshots of the Power BI interface. The left screenshot displays the 'Advanced Filtering' mode for a 'Date (All)' filter. It shows a dropdown set to 'is after' with the value '6/30/2008' and a time selector showing '12 : 00 AM'. Below this are radio buttons for 'And' and 'Or', and a yellow 'Apply Filter' button. The right screenshot shows the 'Visual Level Filters' pane, which lists 'Product (All)', 'SalesAmount (All)', and 'Year (All)'. Under 'Product (All)', there is a list of products with their counts: (All) 13, All-Purpose Bike Stand 13, AWC Logo Cap 13, Bike Wash - Dissolver 13, Classic Vest, L 13, Classic Vest, M 13, Classic Vest, S 13, and Fender Set - Mountain 13. At the bottom of the pane is a 'Basic Filtering' button.

Figure 3.21 The Advanced Filter mode (left screenshot) allows you to specify more complex criteria and multiple conditions for filtering, such as filter dates where the Date field is after June 30th, 2008. The Visual Level Filters area (right screenshot) includes by default all the fields that are used in the visualization.

Creating a Treemap

Let's add a second page to the report that will help you analyze product sales using a Treemap visualization (see **Figure 3.22**).

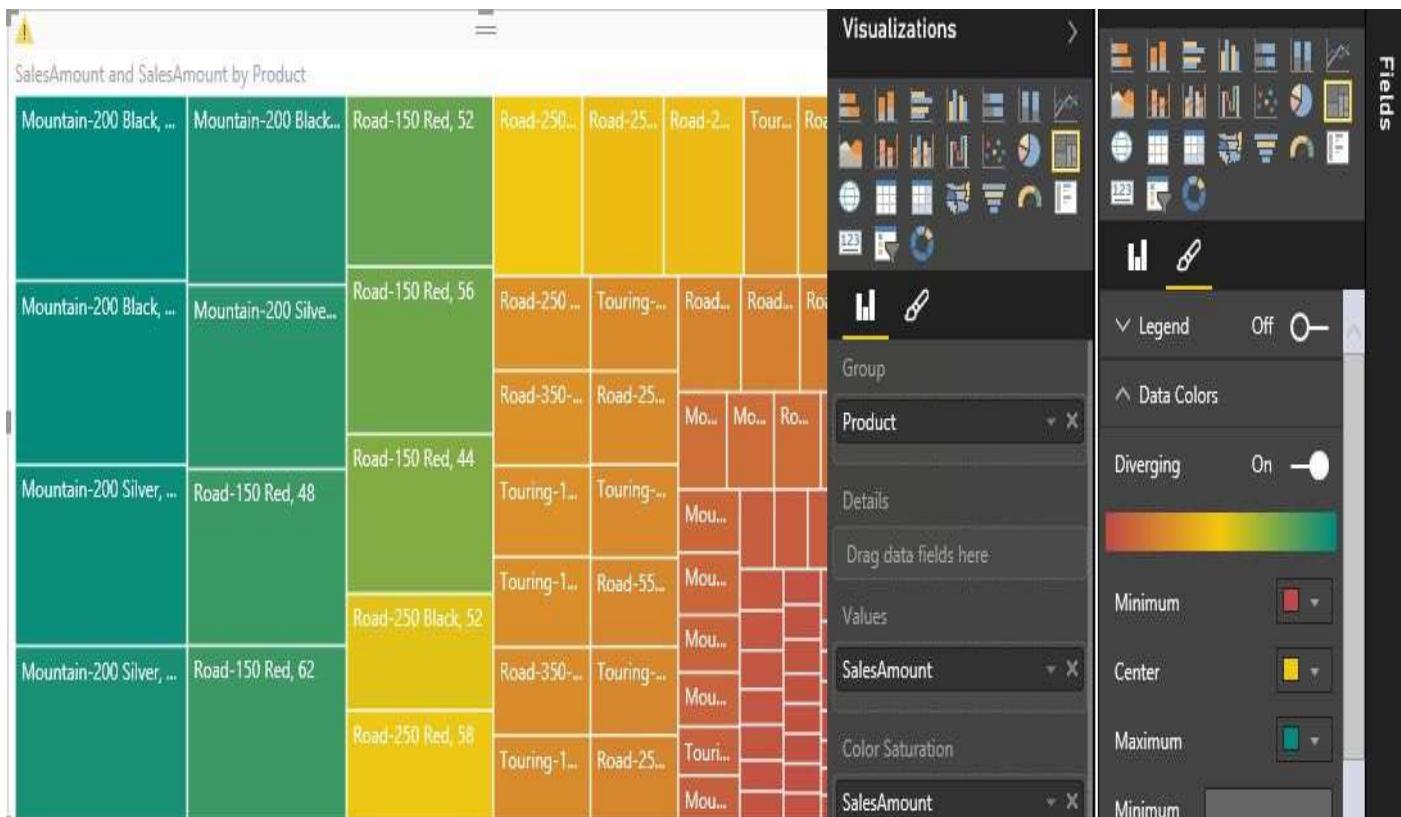


Figure 3.22 The Treemap visualization helps you analyze product sales.

1. At the bottom of the report, click the plus sign to add a new page. Rename the page in place to *Treemap*.
2. In the Fields list, check the SalesAmount and Product fields.
3. By default, Power BI uses arbitrary colors for the tree map tiles. Assuming you want to color the bestselling products in green and worst-selling products in red, drag the SalesAmount field to the Color Saturation area of the Visualizations pane.
4. In the Format tab of the Visualizatons pane, change the Data Colors settings, as shown in **Figure 3.22**. Turning of the Diverging option allows you to specify a color for the values that fall in the middle. You can use the Minimum, Center, and Maximum fields to fine tune the ranges.

3.2.2 Getting Quick Insights

Let's face it, slicing and dicing data to perform root cause analysis (RCA) could be time consuming and tedious. For example, a report might show you that sales are increasing or decreasing but it won't tell you why. Retrospectively, such tasks required you to produce more detail reports, in order to explain sudden data fluctuations. And this gets even more difficult if you're analyzing a model created by someone else because you don't know which fields to use and how to use them to get answers. Enters Quick Insights!

Understanding Quick Insights

Power BI Quick Insights gives you new ways to find insights hidden in your data. With a click of button, Quick Insights run various sophisticated algorithms on your data to search for interesting fluctuations. Originating from Microsoft Research, these algorithms can

discover correlations, outliers, trends, seasonality changes, change points in trends, automatically and within seconds. Table 3.1 lists some of the insights that these algorithms can uncover.

Table 3.1 This table summarizes the available insights.

Insight	Explanation
Major factors(s)	Finds cases where a majority of a total value can be attributed to a single factor when broken down by another dimension.
Category outliers (top/bottom)	Highlights cases where, for a measure in the model, one or two members of a dimension have much larger values than other members of the dimension.
Time series outliers	For data across a time series, detects when there are specific dates or times with values significantly different than the other date/time values.
Overall trends in time series	Detects upward or downward trends in time series data.
Seasonality in time series	Finds periodic patterns in time series data, such as weekly, monthly, or yearly seasonality.
Steady Share	Highlights cases where there is a parent-child correlation between the share of a child value in relation to the overall value of the parent across a continuous variable.
Correlation	Detects cases where multiple measures show a correlation between each other when plotted against a dimension in the dataset

By default, Quick Insights queries as much of the dataset as possible in a fixed time window (about 20 seconds). As it stands, similar to Q&A, Quick Insights requires data to be imported in Power BI. Quick Insights isn't available for datasets that connect directly to data.

Working with Quick Insights

Let's find what insights we can uncover by applying Quick Insights to the Retail Analysis Sample dataset:

- 1.In Power BI, click the ellipsis (...) button to the right of the “Retail Analysis Sample” dataset.
- 2.Click Quick Insights. While Power BI runs the algorithms, it displays a “Searching for insights” message. Once it's done, it shows “Insights are ready” message.
- 3.Click the ellipsis next to the “Retail Analysis Sample” dataset again. Note that the Quick Insights link is renamed to View Insights. Click View Insights.

Power BI opens a “Quick Insights for Retail Analysis Sample” page that shows 40 auto-generated insights! **Figure 3.23** shows the first four. The “Count of Stores (By Month)” report has found that the number of stores increased dramatically in February 2014. This is an example of a Time Series Outlier insight. The “Gross Margin This Year (By Buyer)” insight shows that Kermit Ward has the smallest gross profit margin. This is an example of a Category Outlier insight. Moving down the list, Quick Insights has found another category outlier in the “Gross Margin This Year (by Segment)” insight, where segments 5071 and 5051 have the highest gross margin. Finally, the “Count of Store and Gross Margin Last Year % (By Period) insight found a correlation between two variables. This is an example of a Correlation insight.

As you can see, Quick Insights can really help understand data changes. The refresh button (next to the page title) allows you to rerun the Quick Insight algorithms. Currently, Power BI deactivates them when you close your browser. However, if you find a particular

insight useful, you can click the pin button in the top-right corner to pin to a dashboard. (I discuss creating dashboards in more detail in section 3.4.)

Quick Insights for Retail Analysis Sample

A subset of your data was analyzed and the following insights were found. [Learn more](#)

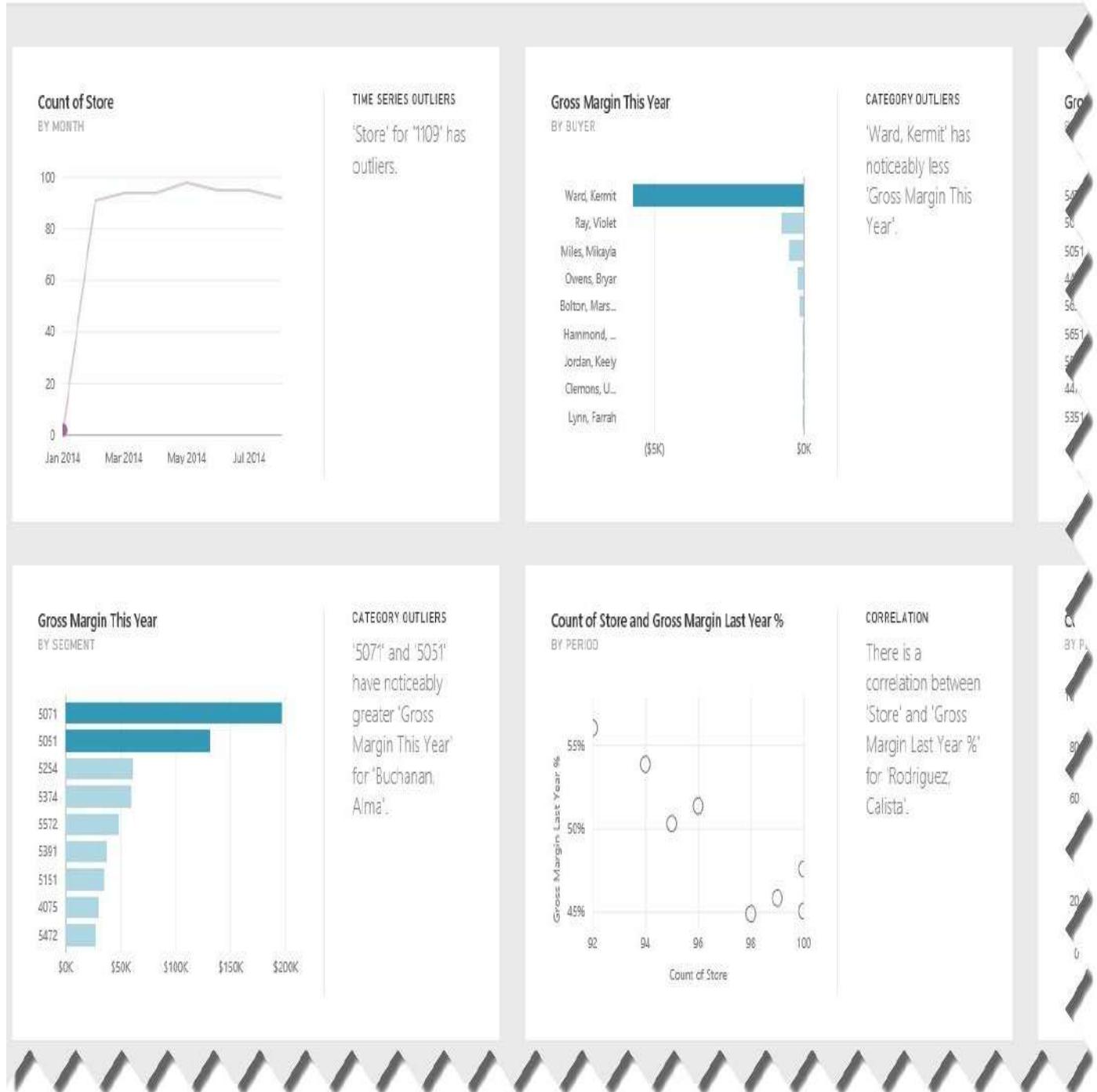


Figure 3.23 This screenshot shows the first four insights for the Retail Analysis Sample dataset.

3.2.3 Working with Excel Reports

In Chapter 1, I mentioned that Power BI Service succeeds Power BI for Office 365. However, thanks to its deep SharePoint integration, Power BI for Office 365 is capable of rendering Excel pivot reports that are embedded in Excel workbooks. To offer the same

feature, Power BI supports rendering Excel reports but the file needs to be located in OneDrive for Business. Power BI requires OneDrive for Business because rendering Excel reports is accomplished via Excel Online. Excel Online is a component of SharePoint Online which is available with Office 365 business plans that include OneDrive for Business.

NOTE OneDrive for Business is a place where business users can store, sync, and share work files. While the personal edition of OneDrive is free, OneDrive for Business requires an Office 365 plan.

Understanding Excel reports

When you use Get Data and choose an Excel file stored in OneDrive for Business, Power BI will ask you if you want to import the data or if you want to connect to the Excel file and render its reports online. You practiced importing from Excel in Chapter 2. If you choose the second option, Power BI will just connect to the file and will show an Excel icon to the left of the report name in the navigation pane (see **Figure 3.24**).

When you click on the report, Power BI renders online the pivot reports and Power View reports that are included in the Excel file. Moreover, Power BI preserves the report interactive features. For example, you can change report filters and slicers. Online Excel reports have their own limitations which are detailed in the “Bring Excel files in Power BI” article by Microsoft at <https://support.powerbi.com/knowledgebase/articles/640168-bring-whole-excel-files-into-power-bi>. For example, currently Power BI doesn’t support reports connected to Analysis Services although reports connected to Power Pivot data models work just fine.

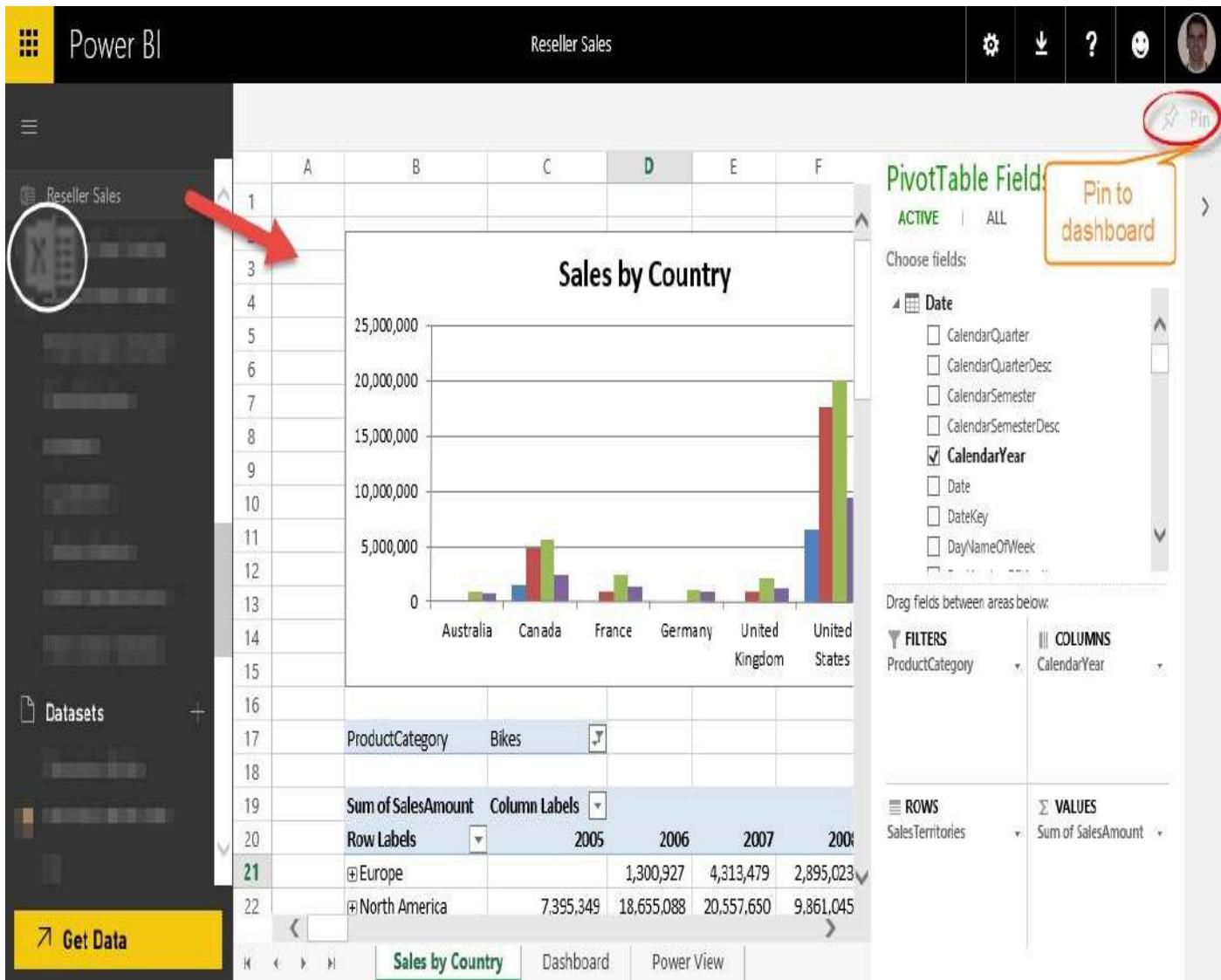


Figure 3.24 Power BI supports rendering Excel reports online if the Excel file is stored in OneDrive for Business.

At this point, you might be confused about which option to use when working with Excel files. Table 3.2 should help you make the right choice.

Table 3.2 This table compares the two Power BI options to work with Excel files.

Criteria	Import Excel files	Connect to Excel files
Data in Excel sheets	Power BI parses the Excel file and imports data as a dataset.	Power BI doesn't parse and import the data. Instead, Power BI connects to the Excel file hosted on OneDrive.
Data model	Power BI imports the data model and creates a dataset.	Power BI doesn't import the data model.
Pivot reports	Power BI doesn't import pivot reports.	Power BI renders pivot reports via Excel Online.
Power View reports	Power BI imports Power View reports and adds them to the Reports section in the left navigation bar.	Power BI renders Power View reports via Excel Online.
Change reports	You can change the imported Power View reports but the original reports in the Excel file remain intact.	You can't change reports. You must open the file in Excel, make report changes, and upload the file to OneDrive.
Data refresh	Scheduled dataset refresh (automatic refresh if saved to OneDrive or OneDrive for Business).	Dashboard tiles from Excel reports are refreshed automatically every few minutes.

Viewing Excel reports

In this exercise, you'll connect an Excel file saved to OneDrive for Business and you'll view its containing reports online. As a prerequisite, your organization must have an

Office 365 subscription and you must have access to OneDrive for Business. The Reseller Sales.xlsx file in the \Source\ch03 folder includes a Power Pivot data model with several tables. The first two sheets have Excel pivot table and chart reports, while the third sheet has a Power View report. While all reports connect to an embedded Power Pivot data model, they don't have to. For example, your pivot reports can connect to Excel tables.

- 1.Copy and save the Reseller Sales.xlsx to your organization's OneDrive for Business.
- 2.In Power BI, click Get Data. Then click the Get button in the Files tile.
- 3.In the next page, click the One Drive for Business tile. In the One Drive for Business page, navigate to the OneDrive for Business folder where you saved the Reseller Sales.xlsx file, and then click Connect.

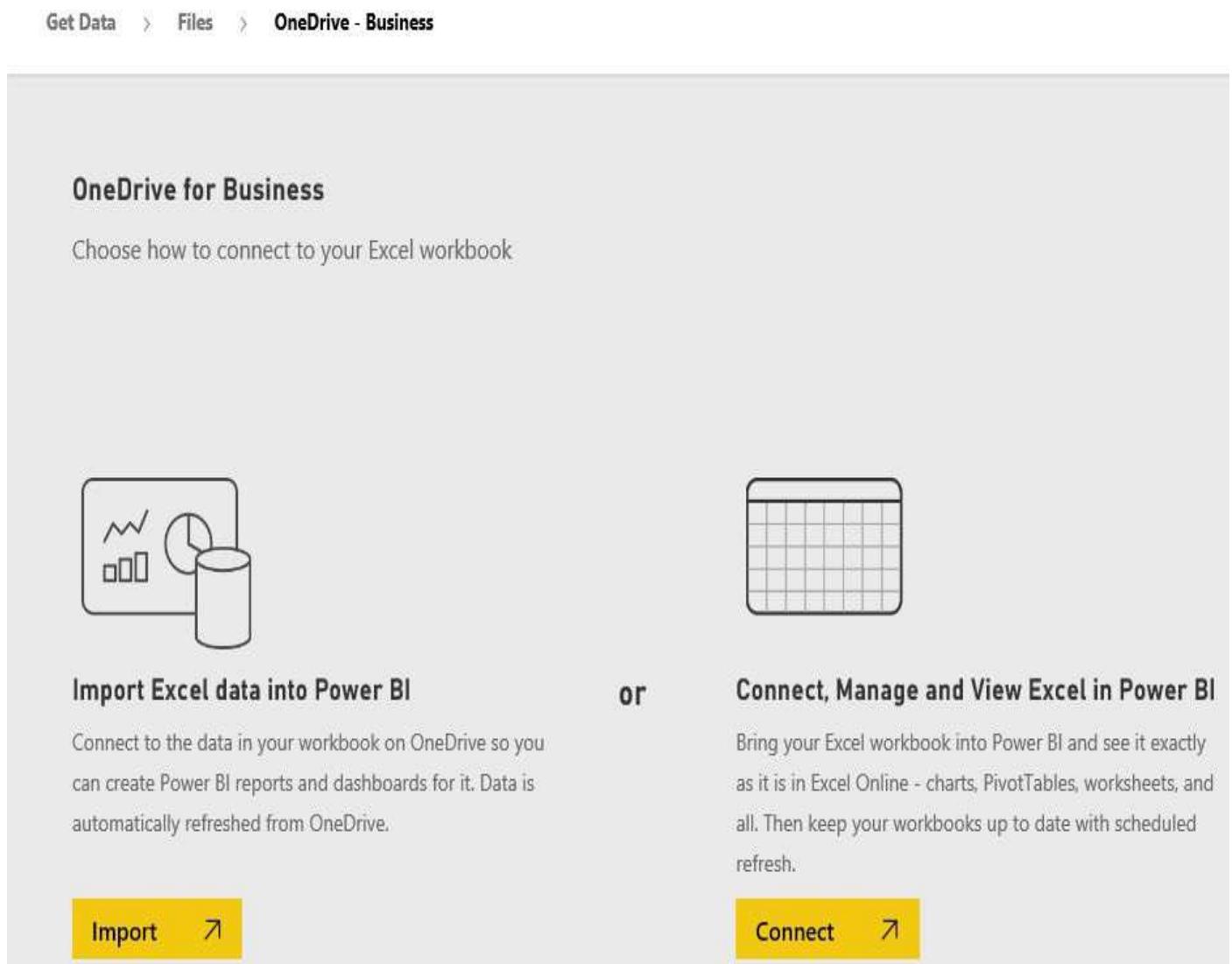


Figure 3.25 When you connect to an Excel file stored on OneDrive for Business, Power BI asks you how you want to work with the file.

- 4.Power BI prompts you how to work with the file (see **Figure 3.25**). Click the Connect button. Power BI adds a report named Reseller Sales to the Reports section in the navigation bar. This report represents the entire Excel file.
- 5.Click the Reseller Sales report and then click View in the context menu that pops up. Notice that Power BI renders the pivot reports and the Power View report online via

Excel Online.

6.(Optional) Try some interactive features, such as changing the report filters and slicers, and notice that they work as they work with SharePoint Server or SharePoint Online.

TIP You can pin a range from an Excel report to a Power BI dashboard. To do so, select the range on the report and then click the Pin button in the upper-right corner of the report (see again **Figure 3.24**). The Pin to Dashboard window allows you to preview the selected section and prompts you if you want to pin it to a new or an existing dashboard. For more information about this feature, read the “Pin a range from Excel to your dashboard!” blog at <http://blogs.msdn.com/b/powerbi/archive/2015/11/24/pin-a-range-from-excel-to-your-dashboard.aspx>.

3.3 Understanding Dashboards

Similar to an automobile's dashboard, a digital dashboard enables users to get a "bird's eye view" of the company's health and performance. A dashboard page typically hosts several sections that display data visually in charts, graphs, or gauges, so that data is easier to understand and analyze. You can use Power BI to quickly assemble dashboards from existing or new visualizations.

NOTE Power BI isn't the only Microsoft-provided tool for creating dashboards. For example, if you need an entirely on-premises dashboard solution, dashboards can be implemented with Excel (requires SharePoint Server) and Reporting Services (requires SQL Server). While Power BI dashboards might not be as customizable SSRS dashboards, they are by far the easiest to implement. They also gain in interactive features, ability to use natural queries, and even real-time updates (when data is streamed to Power BI)!

3.3.1 Understanding Dashboard Tiles

In Chapter 2, I've introduced you to Power BI dashboards and you've learned that dashboards are one of the three main Power BI content items (the other two were datasets and reports). I defined a Power BI dashboard as a summarized view of important metrics that typically fit on a single page. A Power BI dashboard has one or more tiles. Each tile shows data from one report. For example, the Total Stores tile in the Retail Analysis Sample dashboard (see **Figure 3.26**) shows the total number of stores. Although you can add as many tiles as you want, as a rule of thumb try to limit the number of tiles so that they can fit into a single page and so the user doesn't have scroll horizontally or vertically.

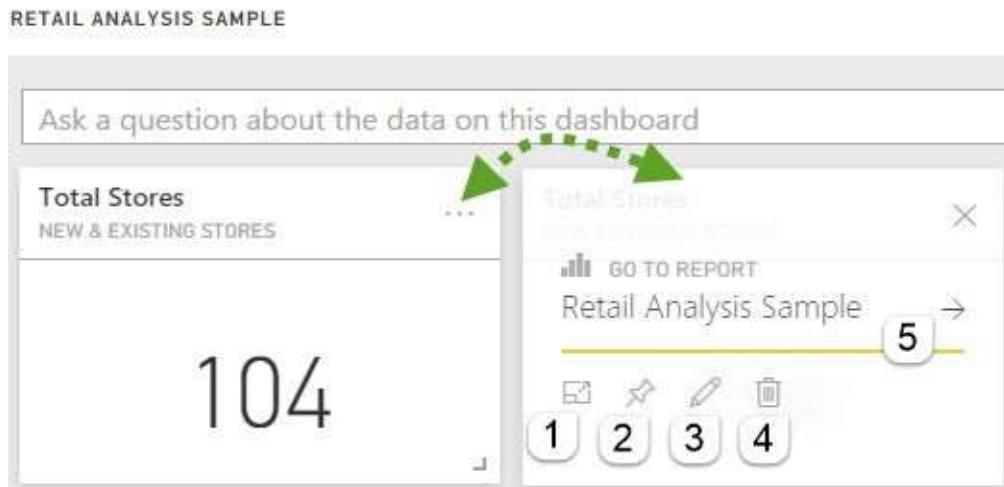


Figure 3.26 When you hover on a tile, the ellipsis menu (...) allows you to access the tile settings.

A tile has a resize handle that allows you to change the tile size. Because tiles can't overlap, when you enlarge a tile it pushes the rest of the content out of the way. When you make the tile smaller, adjacent tiles "snap in" to occupy the empty space.

Understanding tile settings

Power BI supports a limited customization of dashboard tiles. When you hover on a tile, an ellipsis menu (...) shows up in the top-right corner of the tile. When you click the ellipsis menu, the tile changes (see the right snapshot in **Figure 3.26**) and it has the following buttons (numbers corresponds to the numbers in the figure):

- 1.In-focus mode – Similar to popping out visualizations on a report, this button pops out the tile so that you can examine it in more details.
- 2.Pin visual – Pins a tile to another dashboard. Why would you pin a tile from a dashboard instead of from the report? Pinning it from a dashboard allows you to apply the same customizations, such as the title, subtitle, and custom link, to the other dashboard, even though they're not shared (once you pin the tile to another dashboard, both titles have independent customizations).
- 3.Tile details – Allows you to change the tile settings, such as the tile title and subtitle.
- 4.Delete tile – Removes the tile from the dashboard.
- 5.Navigate to report – By default, when you click a tile, Power BI “drills through” and navigates you to the underlying report. Another way to navigate to the report is to click the arrow button next to the report name.

Understanding the in-focus mode

When you click the “In-focus mode” button, Power BI opens another page and zooms out the visualization (see **Figure 3.27**). Tooltips allow you to get precise values. If you pop out a line chart, you can also drag a vertical line to see the precise value of a measure at the intersection of the vertical bar and the line.

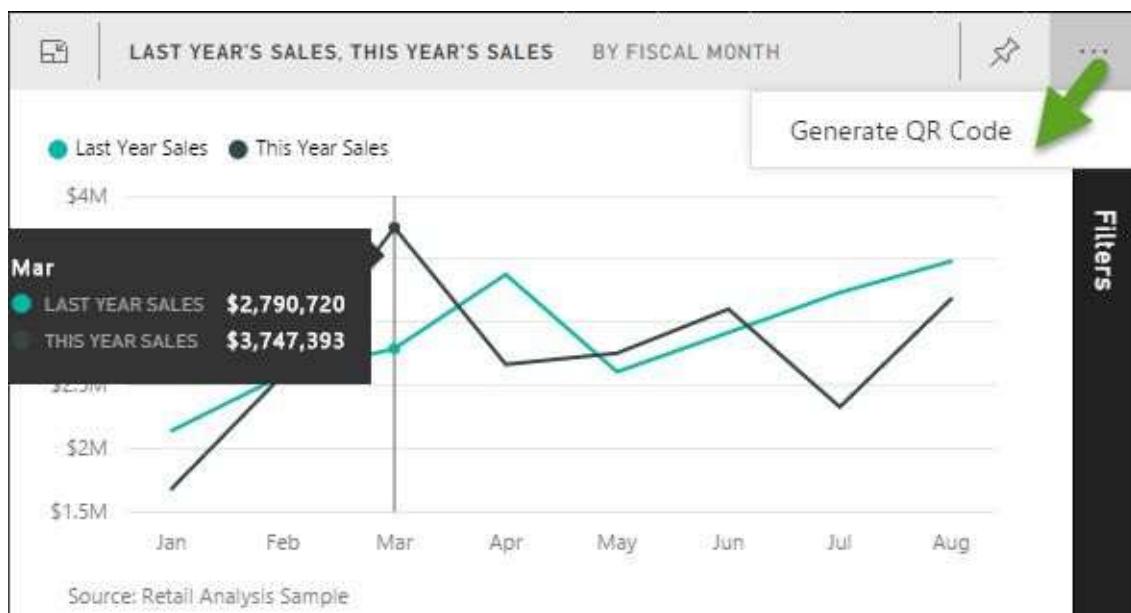


Figure 3.27 The in-focus mode page allows you to examine the tile in more details and to generate a QR code.

The in-focus page has an ellipsis menu (...) in the top-right corner. When you click it, a “Generate QR Code” menu appears. A QR Code (abbreviated from Quick Response Code) is a barcode that contains information about the item to which it is attached. In the case of a Power BI tile, it contains the URL of the tile. How's this useful, you might wonder? You can download the code, print it, and display it somewhere or post the image online. When other people scan the code (there are many QR Code reader mobile apps, including the one included in the Power BI iPhone app), they'll get the tile URL. Now they can quickly navigate to the dashboard tile. So QR codes give users convenient and instant access to dashboard tiles.

For example, suppose you’re visiting a potential customer and they give you a pamphlet. It starts gushing about all these stats about how great their performance has been. You have a hard time believing what you hear or even understanding the numbers. You see the QR Code. You scan it with your phone. It pops up Power BI Mobile on your phone, and rather than just reading the pamphlet, now you’re sliding the controls around in Power BI and exploring the data. You go back and forth between reading the pamphlet and then exploring the associated data on your phone.

Or, suppose you’re in a meeting. The presenter is showing some data, but wants you to explore it independently. He includes a QR Code on their deck. He also might pass around a paper with the QR Code on it. You scan the code and navigate to Power BI to examine the data in more details. As you can imagine, QR codes open new opportunities for getting access to relevant information that’s available in Power BI. For more information about the QR code feature, read the blog “Bridge the gap between your physical world and your BI using QR codes” at <http://bit.ly/1lsVGJ5>.

Understanding tile details

Going back to **Figure 3.26**, let’s see what happens when you click the Tile Details button (button 3). It brings you to the Tile Details window (see **Figure 3.28**). Since report visualizations might have Power BI-generated titles that might not be very descriptive, the Tile Details window allows you to specify a custom title and subtitle for the tile. However, if you want the user to be navigated to another page, you can check the “Set custom link” checkbox and enter the page URL

TIP A custom link could navigate the user to any URL-based resource, such as to an on-premises SSRS report. This could be useful if you want to link a Power BI dashboard to a more detailed report.

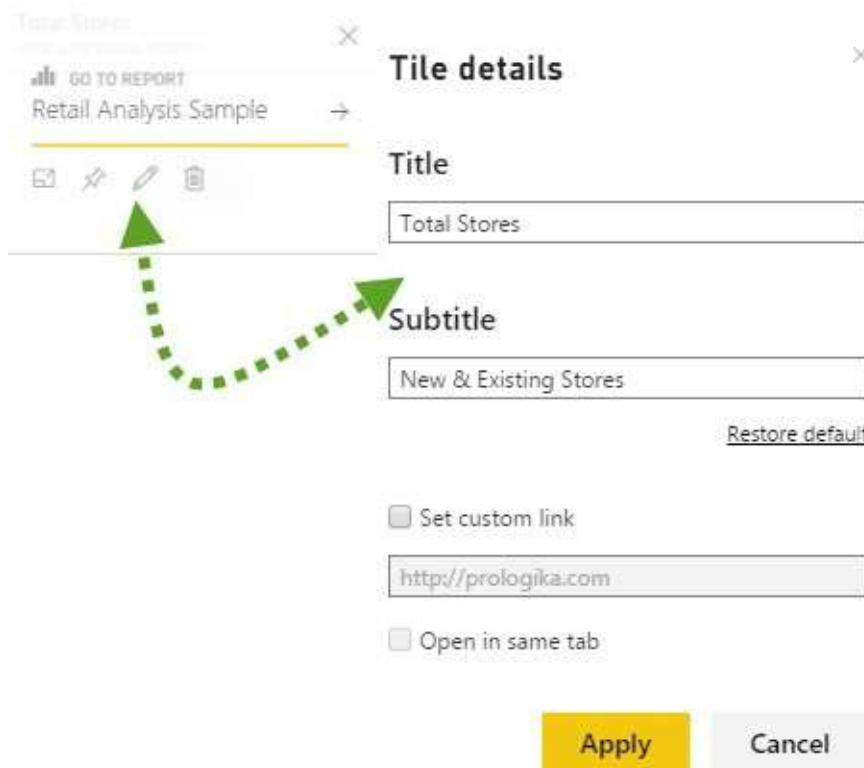


Figure 3.28 The Tile Details window lets you change the tile’s title, subtitle, and to

specify a custom link.

Understanding dashboard settings

Once you've created your dashboard, it will appear under the Dashboards section in the navigation bar. Similar to reports and datasets, you can hover on the dashboard name and click the ellipsis “...” to open the dashboard context menu, which allows you to perform various tasks. For example, sharing, renaming, and deleting dashboards. Deleting a dashboard only removes the dashboard but it leaves the reports and datasets behind.

You can use the Settings link (another way to access this page is to click the Settings menu in the Power BI Application Toolbar on the top-right side of the screen and then click Settings) to disable Q&A (see **Figure 3.29**) if you don't want the Q&A box to appear on the dashboard.

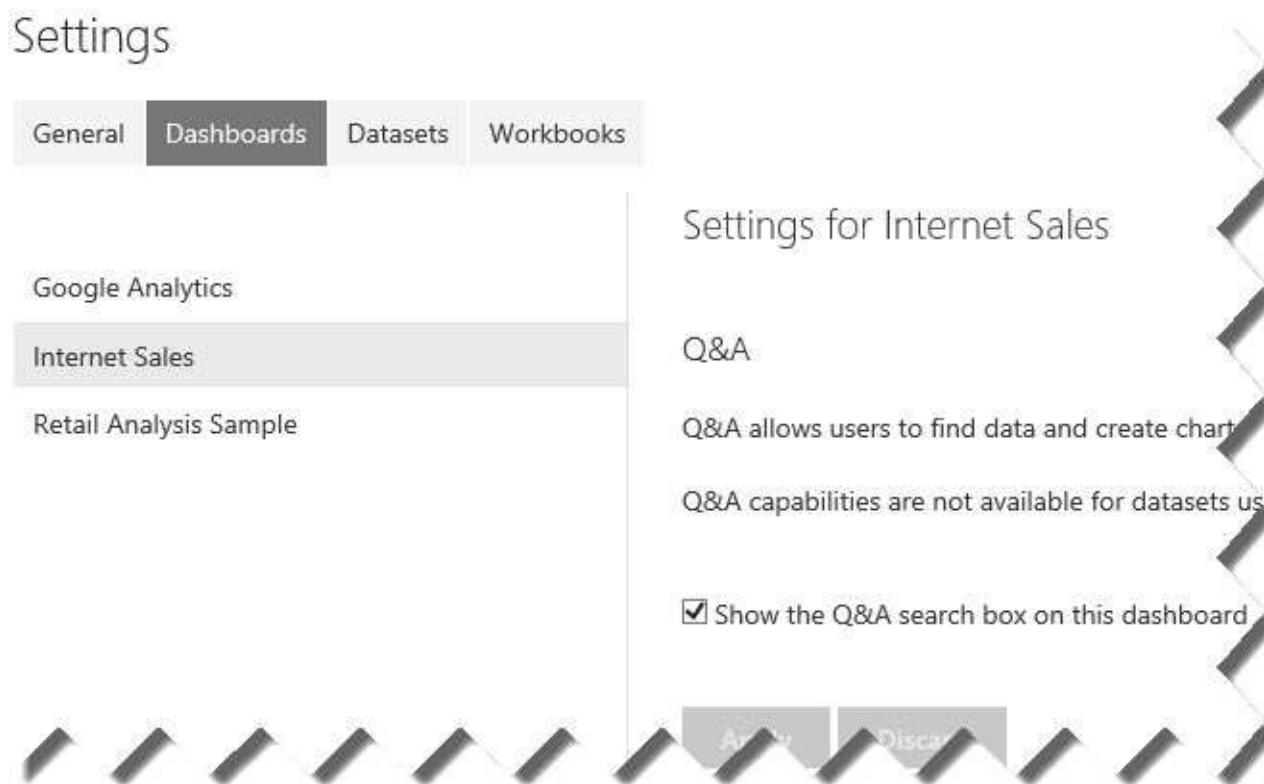


Figure 3.29 Use the dashboard Settings page to disable Q&A for this dashboard.

3.3.2 Adding Dashboard Content

You can create as many dashboards as you want. One way to create an empty dashboard is to click the plus sign (+) next to the Dashboards section in the navigation bar and give the new dashboard a name. Then you can add content to the dashboard by either pinning visualizations from existing reports and dashboards, or by using natural queries.

Adding content by pinning visualizations

To pin a visualization to a dashboard from an existing report or dashboard, you hover on the visualization and click the pushpin button (📌). This opens the Pin to Dashboard window, as shown in **Figure 3.30**. This window shows a preview of the selected visualization and asks if you want to add the visualization to an existing dashboard or to create a new dashboard. If you choose the “Existing dashboard”, you can select the target

dashboard from a drop-down list. Power BI defaults to the last dashboard that you open. If you choose a new dashboard, you need to type in the dashboard name.

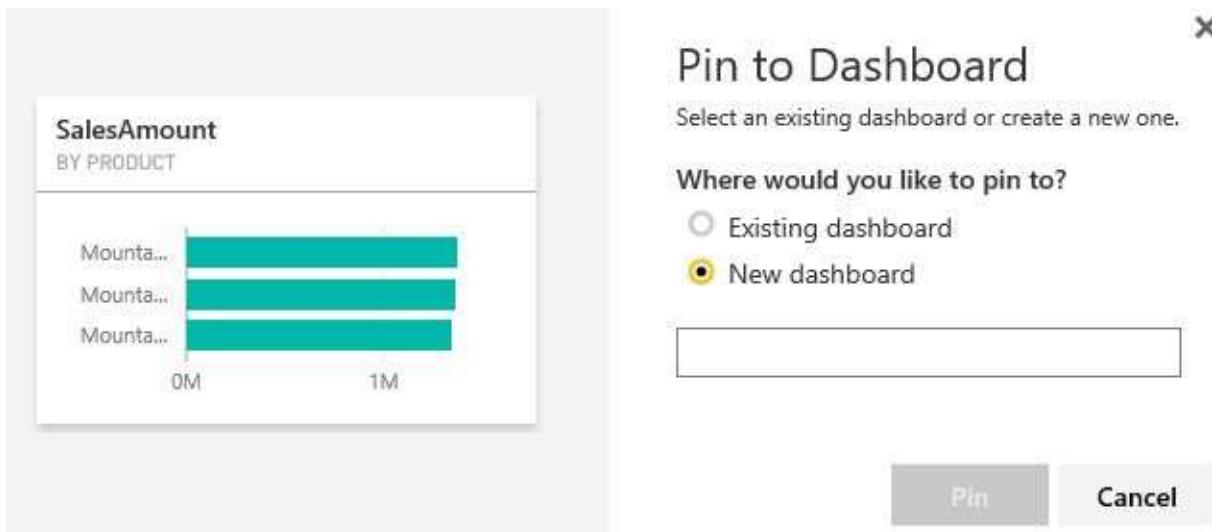


Figure 3.30 Use the Pin to Dashboard window to select which dashboard you want the visualization to be added to.

Think of pinning a visualization like adding a shortcut to the visualization on the dashboard. You can't make layout changes directly to the visualization on the dashboard once it's pinned to a dashboard tile. (You must make such changes to the underlying report where the visualization is pinned from.) Interactive features, such as automatic highlighting and filtering, also aren't available. You'll need to click the visualization to drill through the underlying report in order to make changes or use interactive features.

Pinning report pages

As you've seen, pinning specific visualizations allows you to quickly assemble a dashboard from various reports in a single summary view. However, the pinned visualizations "lose" their interactive features, including interactive highlighting, sorting, and tooltips. The only way to restore these features is to drill the dashboard tile through the underlying report. However, besides pinning specific report visualizations, you can pin entire report pages. This has the following advantages:

- Preserve report interactive features – When you pin a report page, the tile preserves the report layout and interactivity. You can fully interact with all the visualizations in the report tile, just as you would with the actual report.
- Reuse existing reports for dashboard content – You might have already designed your report as a dashboard. Instead of pinning individual report visualizations one by one, you can simply pin the whole report page.
- Synchronize changes – A report tile is always synchronized with the report layout. So if you need to change a visualization on the report, such as from a Table to a Chart, the dashboard tile is updated automatically. No need to delete the old tile and re-pin it.

Follow these steps to pin a report page to a dashboard:

1. Open the report in Reading View or Editing View.
2. Click the “Pin Live Page” menu (see **Figure 3.1** again).
3. In the “Pin to Dashboard” window, select a new or existing dashboard to pin the report page to, as you do when pinning single visualizations. Now you have the entire report page pinned and interactivity works! For example, **Figure 3.31** shows the “New Stores Analysis” page from the “Retail Analysis Sample” report that is now pinned to a dashboard.

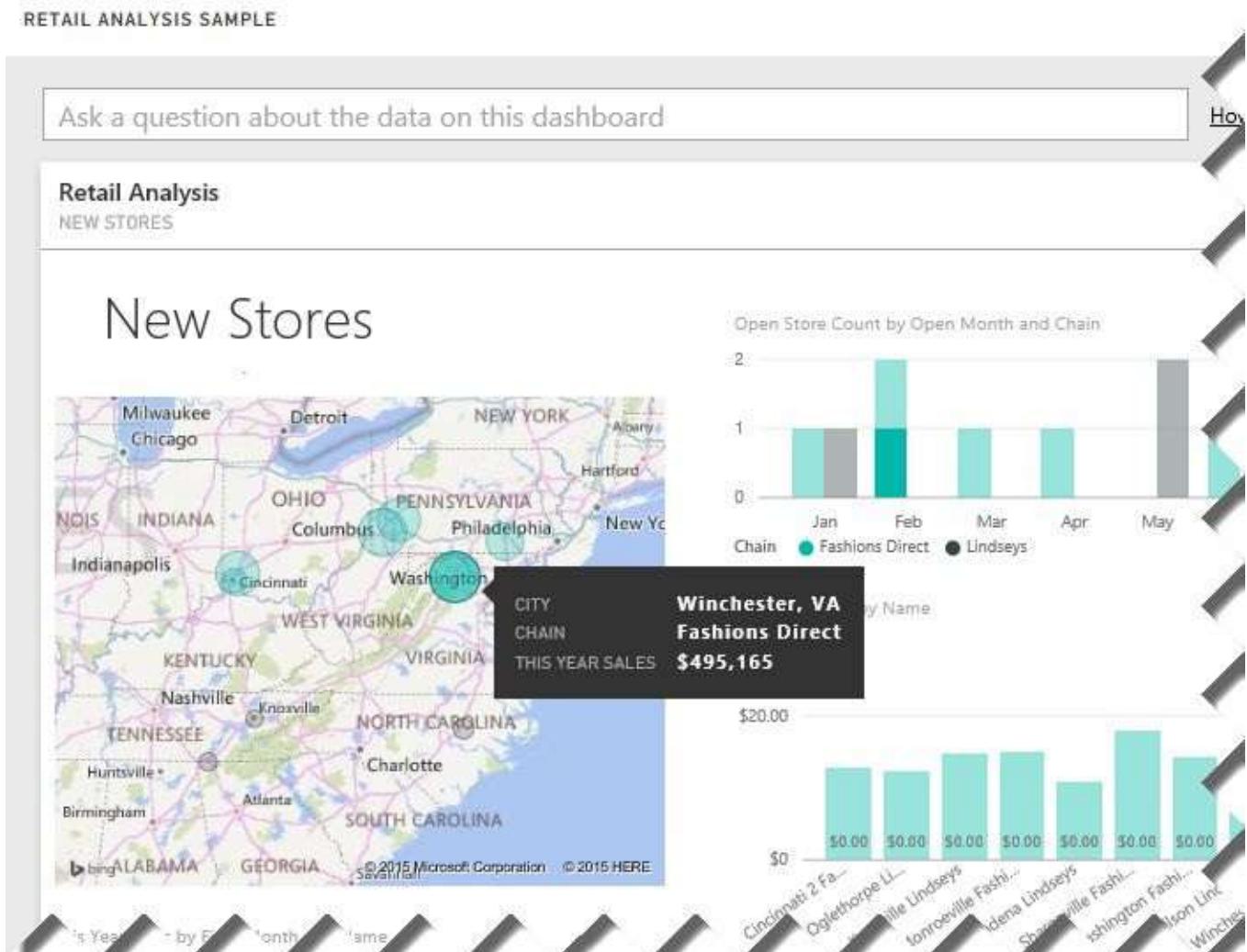


Figure 3.31 You can pin report pages to your dashboards to preserve interactive features, reuse reports as dashboards, and synchronize layout changes.

Adding content by asking natural queries

Another way to add dashboard content is to use natural questions (Q&A). Natural queries let data speak for itself by responding to questions entered in natural language, similar to how you search the Internet. The Q&A box appears on top of every dashboard that connects to datasets with imported data.

NOTE As of the time of writing, natural queries are available only with datasets created by importing data. Microsoft indicated plans to extend this feature to support live connections, such as to allow you to use Q&A with on-premises SSAS models. Also, it currently supports English only (as each language is so different).

When you click the Q&A box, it suggests questions you could ask about the dashboard data (see

Figure 3.32).

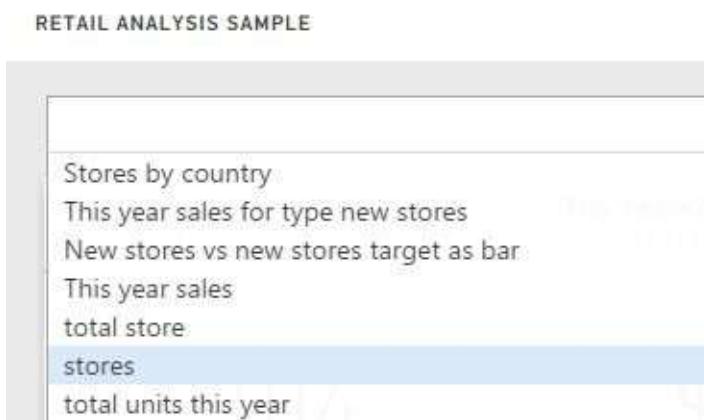


Figure 3.32 The Q&A box has predefined questions.

Of course, these suggestions are just a starting point. Power BI inferred them from the table and column names in the underlying dataset. You can add more predefined questions by following these steps:

- 1.In the navigation pane, click the ellipsis (...) next to the dashboard to open the dashboard Settings page.
- 2.Click the Datasets tab (see **Figure 3.29** again) and then select the desired dataset.
- 3.In the dataset settings, click “Add a question”, and then type a statement that uses dataset fields, such as “sales by country”.

Users aren’t limited to predefined questions. They can ask for something else, such as “what were this year sales”, as shown in **Figure 3.33**.

A screenshot of the Power BI interface showing the Q&A box. A user has typed "what were this year sales" into the input field. Below the input field, the system has interpreted this as "This Year Sales" and displayed a large, bold "\$22M" in a central area, indicating the total sales value. To the right of the Q&A box is the Visualizations pane, which shows various chart and visualization icons. Below the Q&A box is a decorative graphic of a mountain range at the bottom of the dashboard.

Figure 3.33 Q&A box interprets the natural question and defaults to the best visualization.

As you type a question, Power BI shows you how it interpreted the question below the Q&A box. By doing so, Power BI searches not only the datasets used in the dashboard, but

also all your datasets. So that you can understand which dataset answers your question, Power BI displays the source dataset below the visualization (in **Figure 3.33** it displays, “Source: Retail Analysis Sample”).

Power BI attempts to use the best visualization, depending on the question and supporting data. In this case, Power BI has interpreted the question as “Show this year sales” and decided to use a card visualization. If you continue typing so the question becomes “what were this year sales by product”, it would probably switch over to a Bar Chart. However, if you don’t have much luck with Q&A, you can always use the Visualizations and Fields panes to customize the visualization, as you can do with reports.

In other words, think of Q&A as a way to jump start your data exploration by creating a report that you can customize further, such as changing the color of the lines, adding labels to the axes, or even choose another visualization type! Once you’re done with the visualization, you can click the pushpin button to add the visualization to the dashboard. Once the tile is added, you can click it to drill through into the dataset. Power BI brings you the visualization you created and shows the natural question that was used.

So how smart is Q&A? Can it answer any question you might have? Q&A searches metadata, including table, column, and field names. It also has built-in smarts on how to filter, sort, aggregate, group, and display data. For example, the Internet Sales dataset you imported from Excel has columns titled “Product”, “Month”, “SalesAmount”, and “OrderQuantity”. You could ask questions about any of those entities. You could ask it to show SalesAmount by Product or by Month, and others. You should also note that Q&A is smart enough to interpret that SalesAmount is actually “sales amount”, and you can use both interchangeably.

NOTE Data analysts creating Excel Power Pivot data models can fine tune the model metadata for Q&A. For example, Martin can create a synonym to tell Power BI that State and Province mean the same thing.

3.3.3 Sharing Dashboards

Power BI allows you to share dashboards easily with your coworkers. This type of sharing allows other people to see the dashboards you’ve created, and it’s supported by Power BI Free. However, shared dashboards and associated reports are ready-only to people you invite, and recipients can’t personalize them. By contrast, organizational packs (which require Power BI Pro edition) allow you to share content that can be copied and personalized.

Understanding sharing features and limitations

Consider using simple dashboard sharing when you need a quick and easy way to share your content with a limited number of people. It works only with dashboards; you can use simple sharing to directly share reports and datasets. However, your coworkers can only interact with the underlying reports in Reading View (the Edit Report menu will be disabled). They can’t create new reports or save changes to existing reports. When you make and save changes, your coworkers can immediately see the changes. They’ll know that a dashboard is shared with them because they’ll see this icon  in front of the dashboard name in the navigation bar.

You can only share dashboards with people whose email has the same domain as yours or a domain that's registered within the Office 365 tenant. For example, if Maya's email is maya@adventureworks.com, she can share with martin@adventureworks.com but not with martin@prologika.com, unless prologika.com is registered with the adventureworks tenant.

Understanding sharing and resharing

To share a dashboard, click the Share link to the right of the dashboard name (see **Figure 3.26** again). This brings you to the Share Dashboard page, as shown in **Figure 3.34**.

Share dashboard

Not shared with anyone

Invite Shared with

The screenshot shows the 'Share dashboard' interface. At the top, there's a search bar with 'Neal Waterstreet' and an 'X' button. Below it is a text input field containing 'john'. A yellow highlight covers the list of results below. The first result is 'John Lucyjohn john.lucyjohn @prologika.com'. The second result is 'John Pollock john.pollock @prologika.com'. The rest of the page is mostly blank.

Recipients will have access to the same data and reports as you have in this dashboard. [Learn more](#)

- Allow recipients to share your dashboard
- Send email notification to recipients

Share

Figure 3.34 Use the Share Dashboard page to enter a list of recipient emails, separated with a comma or semi-colon.

Enter the email addresses of the recipients separated by comma (,) or semi-colon (;). You can even use both. Power BI will validate the emails and inform you if they are incorrect.

TIP Want to share with many users, such as with everyone in your department, but don't have Power BI Pro? If your organization uses Office 365 for email, you can share with members of a distribution group by entering in the email address associated with the distribution group. If you have Power BI Pro, consider asking IT to create a workspace instead of simple sharing. I discuss workspaces in Part 3 of this book.

Next, enter an optional message. To allow your coworkers to re-share your dashboard with others, check “Allow recipients to share your dashboard”. If you change your mind later on and you want to stop sharing, click the “Shared With” tab. This tab allows you to stop sharing and/or disable re-shares for each coworker you shared the dashboard with.

By default, the “Send email notification to recipients” checkbox is checked. When you click the Share button, Power BI will send an e-mail notification with a link to your dashboard. When the recipient clicks the dashboard link and signs in to Power BI, the shared dashboard will be added to the navigation bar.

You might not always want the person you share a dashboard with to go through the effort of checking their email and clicking a link just for your dashboard to show up in their workspace. If you uncheck the “Send email notification to recipients” checkbox, you can share dashboards directly to a user’s My Workspace without them having to do anything. Now when you click Share, the dashboard will just show up in the other users’ My Workspace with no additional steps required on their end.

3.4 Working with Dashboards

Next, you'll create the Internet Sales dashboard shown in **Figure 3.35**. You'll create the first three tiles by pinning visualizations from an existing report. Then you'll use Q&A to create the fourth tile that will show a Line Chart.



Figure 3.35 The Internet Sales dashboard was created by pinning visualizations and then using a natural query.

3.4.1 Creating and Modifying Tiles

Let's start implementing the dashboard by adding content from a report. Then you'll customize the tiles and practice drilling through the content.

Pinning visualizations

Follow these steps to pin visualizations from the Internet Sales Analysis report that you created in the previous exercise:

1. In the navigation bar, click the Internet Sales Analysis report to open it in Reading View or Editing View.

- 2.Hover on the SalesAmount card and click the pushpin button.
- 3.In the Pin to Dashboard window, select the “New dashboard” option, enter *Internet Sales*, and click Pin.

This adds a new dashboard named Internet Sales to the navigation bar under the Dashboards section. Power BI shows a message that the visualization has been pinned to the Internet Sales dashboard.

- 4.In the Internet Sales Analysis report, pin also the OrderQuantity Card and the “SalesAmount and OrderQuantity by Date” Combo Chart but this time pin them to the Internet Sales existing dashboard.
- 5.In the navigation page, click the Internet Sales dashboard. Hover on the SalesAmount Card, and click the ellipsis menu (...). Click the pencil button. In the Tile Details window, enter *Sales* as a tile title.
- 6.Change the title for the second Card to *Orders*. Configure the Combo Chart tile to have *Sales vs Orders* as a title and *BY DATE* as a subtitle.
- 7.Rearrange the tiles to recreate the layout shown back in **Figure 3.35**.

Drilling through the content

You can drill through the dashboard tiles to the underlying reports to see more details and to use the interactive features.

- 1.Click any of the three tiles, such as the Sales card tile. This action navigates to the Internet Sales Analysis report which opens in Reading View.
- 2.To go back to the dashboard, click its name in the Dashboards section of the navigation bar or click your Internet browser’s Back button.
- 3.(Optional) Pin visualizations from other reports or dashboards, such as from the Retail Analysis Sample report or dashboard.
- 4.(Optional) To remove a dashboard tile, click its ellipsis (...) button, and then click the X button.

3.4.2 Using Natural Queries

Another way to create dashboard content is to use natural queries. Use this option when you don’t have an existing report or dashboard to start from, or when you want to add new visualizations without creating reports first.

Using Q&A

Next, you’ll use Q&A to add a Line Chart to the dashboard.

- 1.In the Q&A box, enter “sales amount by date”. Note that Power BI interprets the question as “Show sales amount sorted by date” and it defaults to a Line Chart, as shown in **Figure 3.36**.

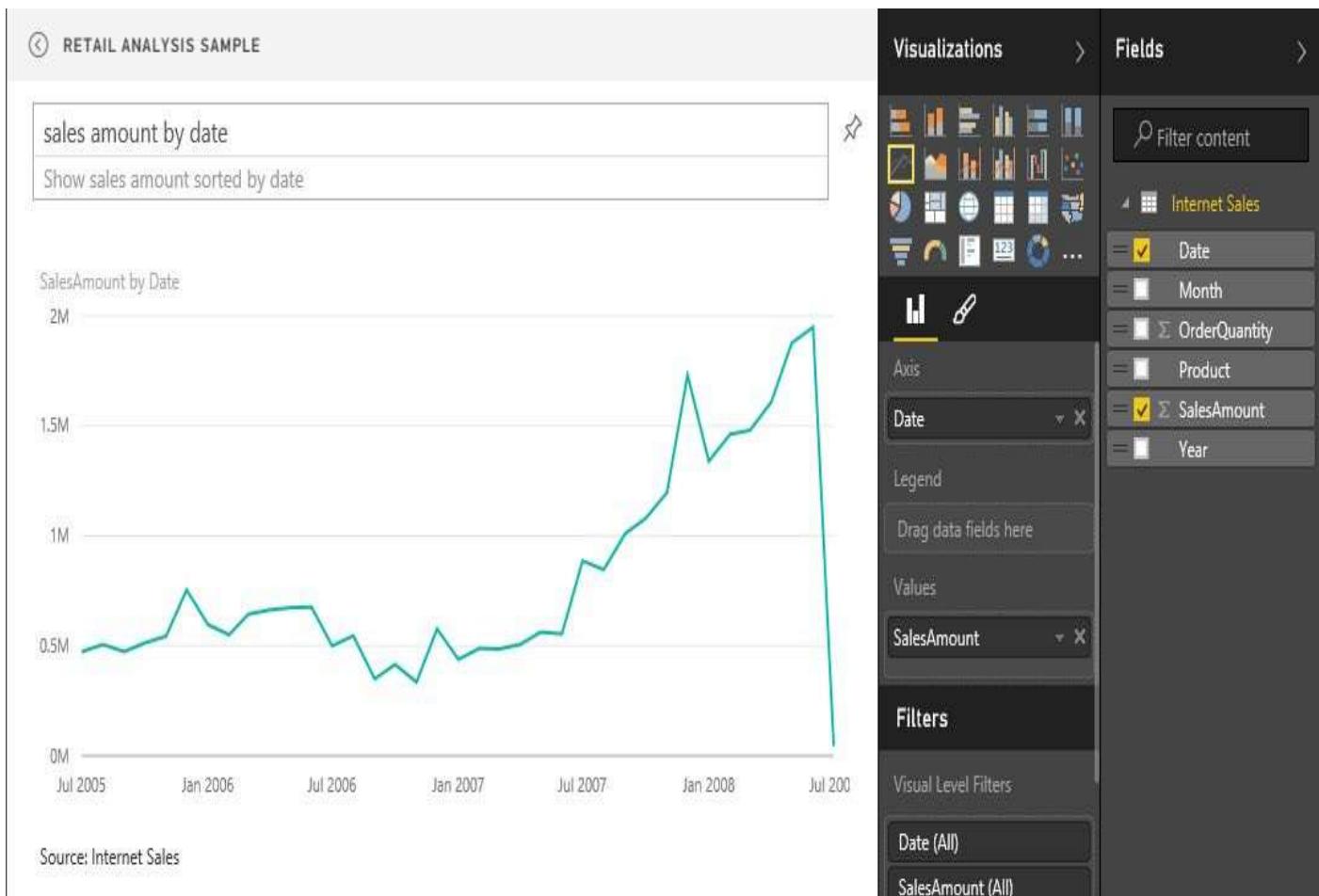


Figure 3.36 Create a Line Chart by typing a natural question.

2. You should also notice that you can use the Visualizations pane to change the visualization. Another way to use a specific visualization is to specify the visualization type in the question. Change the question to “sales mount by date as column chart”. Power BI changes the visualization to a Column Chart.
- 3.(Optional) Practice your reporting skills to customize the visualization using the Visualizations and Fields pane. For example, use the Format tab of the Visualizations pane to turn on data labels.
- 4.Click the pushpin button to pin the visualization to a new dashboard tile.

Drilling through content

Similar to tiles bound to report visualizations, Power BI supports drilling through tiles that are created by Q&A:

- 1.Back in the dashboard, click the new tile that you created with Q&A. Power BI brings you back to the visualization as you left it (see **Figure 3.36**). In addition, Power BI shows the natural question you asked in the Q&A box.
- 2.(Optional) Use a different question or make some other changes, and then click the pushpin button again. This will bring you to the Pin to Dashboard window. If you choose to pin the visualization to the same dashboard, Power BI will add a new tile to the dashboard.

3.4.3 Integrating with Cortana Digital Assistant

You've seen how Q&A can help you gain insights from your data using natural queries. Wouldn't be nice to bring this experience directly to your Windows laptop without having to even open Power BI? If you use Windows 10, you're in luck because Cortana can use the same Q&A capabilities to provide data-driven answers to your questions.

Configuring Cortana for Power BI Q&A

Cortana is an intelligent personal assistant that Microsoft included in Windows phones, Xbox One, and Windows 10. Cortana can help you update your schedule and answer questions using information from Bing, such as current weather and traffic conditions, sports scores, and biographies. For more information about Cortana's features, read "What is Cortana?" at <http://windows.microsoft.com/en-us/windows-10/getstarted-what-is-cortana>. Follow these steps to integrate Cortana with Power BI:

1. Ensure that you have Windows 10 with Windows 10 November Update (version 1511) or higher. To check, in Windows press the hotkey Win+R to open the Run dialog, type `winver`, and then press Enter.
2. To find if Cortana is activated, type *Cortana* in the Windows search box (located in the taskbar on the left).
3. In the Search results window, click "Cortana & Search settings". In the Settings window, make sure that the first setting "Cortana can give you suggestions..." is on. If you want Cortana to respond to your voice when you say "Hey Cortana", turn on the "Hey Cortana" setting as well.
4. So that Cortana can reach out to Power BI and access your datasets, you must first add your work or school account to Windows. Click the Windows button to the left of the search box in the Windows taskbar, and then click Settings. In the Settings page, click Accounts. Scroll to the bottom of the page.
5. In the "Accounts used by other apps", check if the account that used to sign in to Power BI is listed (see **Figure 3.37**). If the account is not listed, click the "Add a Microsoft account" link, and then add the account.

Accounts used by other apps

The screenshot shows the 'Accounts used by other apps' section in the Windows Settings. It lists two accounts: a Microsoft account (@hotmail.com) and a Work or school account (@prologika.com). The Work or school account is highlighted with a red border.

 Microsoft account	@hotmail.com
 Work or school account	@prologika.com

Figure 3.37 Add your work or school account so that Cortana can integrate with Power BI.

6. By default, Power BI datasets are not enabled for Cortana. You must turn on the Cortana integration for each dataset that you want Cortana to access. Open your Web browser and log in to the Power BI portal. In the Application Toolbar at the top-right corner, expand the Settings menu, and then click Settings.

7.In the Settings page, click the Datasets tab (**Figure 3.38**).

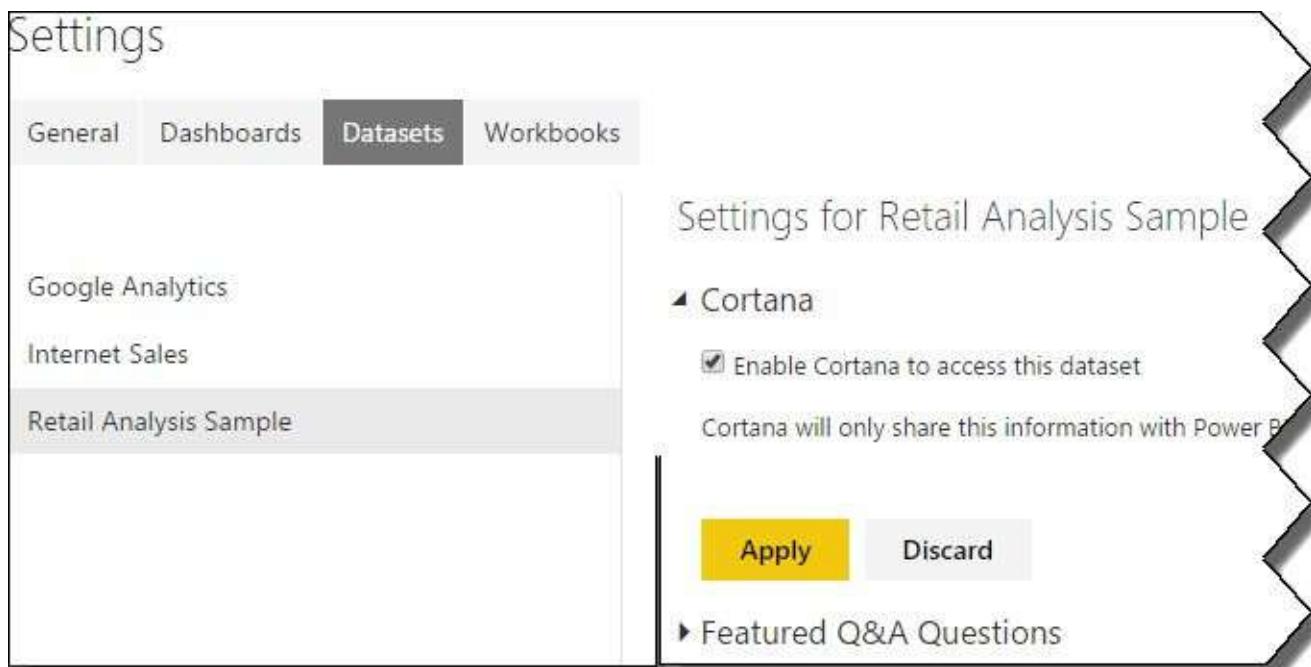


Figure 3.38 Use the Settings page to enable Cortana to access datasets.

8.To enable Cortana for the “Retail Analysis Sample” dataset, click that dataset, and then check the “Enable Cortana to access this dataset” checkbox. Click Apply.

Now Cortana can access this dataset but give it some time to discover it and learn about its data.

Getting data insights with Cortana

Once you've integrated Cortana with Power BI, getting insights is easy. As you've seen, by utilizing Power BI's Q&A data visualization capabilities, answers can range from simple numerical values (“sales for the last quarter”), charts (“gross margin by store”), maps (“this year sales by state”), and even entire reports that are specifically optimized for Cortana! Potential answers are either determined by Cortana on the fly directly from the data, or by using reports that are already created in Power BI that help answer a question.

1.In the Windows search box, enter “gross margin by store”. Cortana should show some matches under the Power BI category. Alternatively, you can say “Hey Cortana” to activate Cortana and dictate your question.

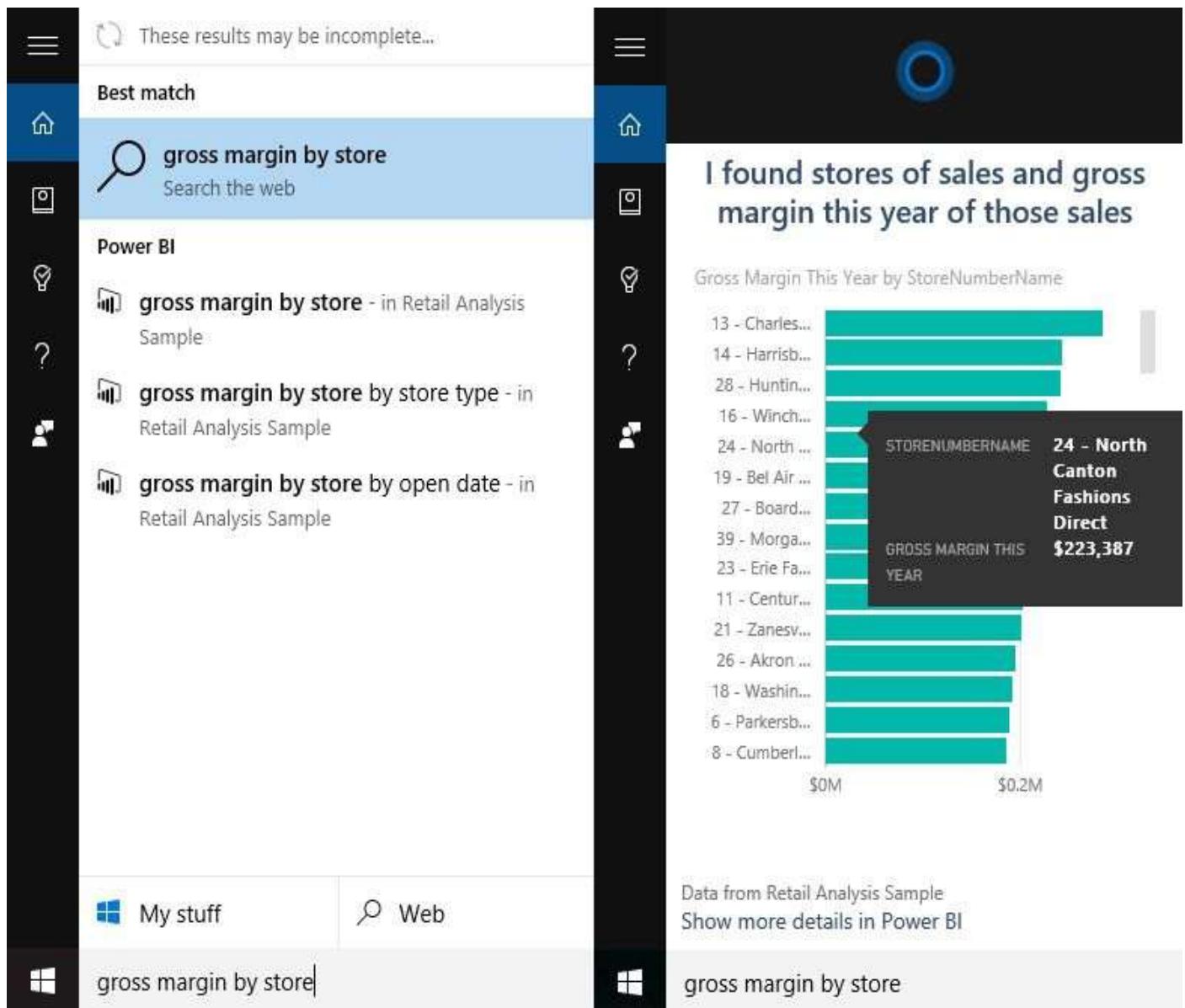


Figure 3.39 Cortana finds Power BI matches to your questions and visualizes your data.

- 2.Click the “gross margin by store – in Retail Analysis Sample” match. Cortana shows a Power BI visualization with the results (see **Figure 3.38**). Note that you can hover on a data value and get a tooltip
- 3.(Optional) To further explore an answer, click the “Show more details in Power BI” link under the visualization. This opens your Web browser and navigates you to the Power BI Q&A page that is prepopulated with the same question.

TIP Data analysts creating self-service models in Power BI Desktop can customize the Cortana answers by adding pages to a report that are optimized for Cortana. For more information about this feature, read the “Use Power BI Desktop to create a custom Answer Page for Cortana” topic at <https://powerbi.microsoft.com/en-us/documentation/powerbi-service-cortana-desktop-entity-cards>.

As you can see, Cortana opens new opportunities to enable your business, and your customers’ businesses, to get things done in more helpful, proactive, and natural ways.

3.5 Summary

As a business user, you don't need any special skills to gain insights from data. With a few clicks, you can create interactive reports for presenting information in a variety of ways that range from basic reports to professional-looking dashboards.

You can create a new report by exploring a dataset. Power BI supports popular visualizations, including charts, maps, gauges, cards, and tables. When those visualizations just won't do the job, you can import custom visuals from the Power BI visuals gallery. To preserve your investment in Excel pivot and Power View reports, save the Excel files in OneDrive for Business and use Excel Online for rendering these reports.

Consider dashboards for displaying important metrics at a glance. You can easily create dashboards by pinning existing visualizations from reports or from other dashboards. Or, you can use natural queries to let the data speak for itself by responding to questions, such as "show me sales for last year". You can drill through to the underlying reports to explore the data in more detail.

Besides using the Power BI portal, you can access reports and dashboards on mobile devices, as you'll learn in the next chapter!

Chapter 4

Power BI Mobile

To reach its full potential, data analytics must not only be insightful but also pervasive. Pervasive analytics is achieved by enabling information workers to access actionable data from anywhere. Mobile computing is everywhere and most organizations have empowered their employees with mobile devices, such as tablets and smartphones. Preserving this investment, Power BI Mobile enriches the user's mobile data analytics experience. Not only does it allow viewing reports and dashboards on mobile devices, but it also enables additional features that your users would appreciate. It does so by providing native mobile applications for iOS, Android, and Windows devices.

This chapter will help you understand the Power BI Mobile capabilities. Although native applications differ somewhat due to differences in device capabilities and roadmap priorities, there's a common set of features shared across all the applications. To keep things simple, I'll demonstrate most of these features with the Windows native application, except for data alerts which are currently available on iPhone only.

4.1 Introducing Native Applications

Power BI is designed to render reports and dashboards in HTML5. As a result, you can view and edit Power BI content from most modern Internet browsers. Currently, Power BI officially supports Microsoft Edge, Microsoft Explorer 10 and 11, the Chrome desktop version, the latest version of Safari for Mac, and the latest Firefox desktop version.

To provide additional features that enrich the user's mobile experience outside the web browser, Power BI currently offers three native applications that target the most popular devices: iOS (iPad and iPhone), Android, and Windows devices. These native applications are collectively known as Power BI Mobile. Next, I'll introduce you briefly to each of these applications.

4.1.1 Introducing the iOS Application

Microsoft released the iOS application on December 18th, 2014, and it was the first native app for Power BI. Initially, the application targeted iPad devices but was later enhanced to support iPhone. Users with these devices can download the Power BI iOS application from the Apple App Store at <https://itunes.apple.com/us/app/microsoft-power-bi/id929738808>.

Understanding capabilities

The iOS application (see **Figure 4.1**) supports an intuitive, touch optimized experience for monitoring business data on iPad or iPhone. You can view your dashboards, interact with charts and tiles, explore additional data by browsing reports, and share dashboard images with your colleagues by email. You can also annotate a dashboard, such as entering a comment, and send the annotation by email.



Figure 4.1 The iOS application targets iPad and iPhone devices.

Landscape mode lets you view and navigate your dashboards in the same way as you do in the Power BI portal. To view your dashboard in landscape, open it and simply rotate your

phone. The dashboard layout changes from a vertical list of tiles to a “Bird’s eye” landscape view. Now you can see all of your dashboard’s tiles as they are in the Power BI portal.

Understanding data alerts

Currently, the iPhone app is the only native application that supports data alerts. You can use data alerts to receive a notification when your data exceeds certain thresholds that you specify. Currently, data alerts are only supported with Single Card visualizations. Follow these steps to set up an alert on iPhone:

- 1.Tap a dashboard tile that was pinned from a Simple Card visualization (see **Figure 4.2**).



Figure 4.2 iPhone supports setting up data alerts on Simple Card visualizations.

- 2.Tap the bell icon and use the Rules window to specify alert rules for above and/or below thresholds (see **Figure 4.3**). For example, you can set an alert if a warehouse supply gets too low, or when sales exceed \$20 million.



Figure 4.3 This alert notifies you when sales exceed \$22.05 million.

When the data alert condition is met, you’ll receive an in-app notification so that you never miss import data changes!

4.1.2 Introducing the Android Application

Microsoft released the Power BI Mobile Android application in July 2015 (see **Figure 4.4**). This application is designed for Android smartphones and it’s available for download from Google Play Store at <https://play.google.com/store/apps/details?id=com.microsoft.powerbim>.



Figure 4.4 The Android application targets Android devices.

Android users can use this app to explore dashboards, invite colleagues to view data, add annotations and share insights over email.

4.1.3 Introducing the Windows Application

In May 2015, Power BI Mobile added a native application for Windows 8.1 and Windows 10 devices, such as Surface tablets (see **Figure 4.5**). Microsoft is currently working on a Power BI app for Windows 10 phones. You can download the Power BI mobile app for Windows devices from Windows Store at <http://go.microsoft.com/fwlink/?LinkId=526478>.

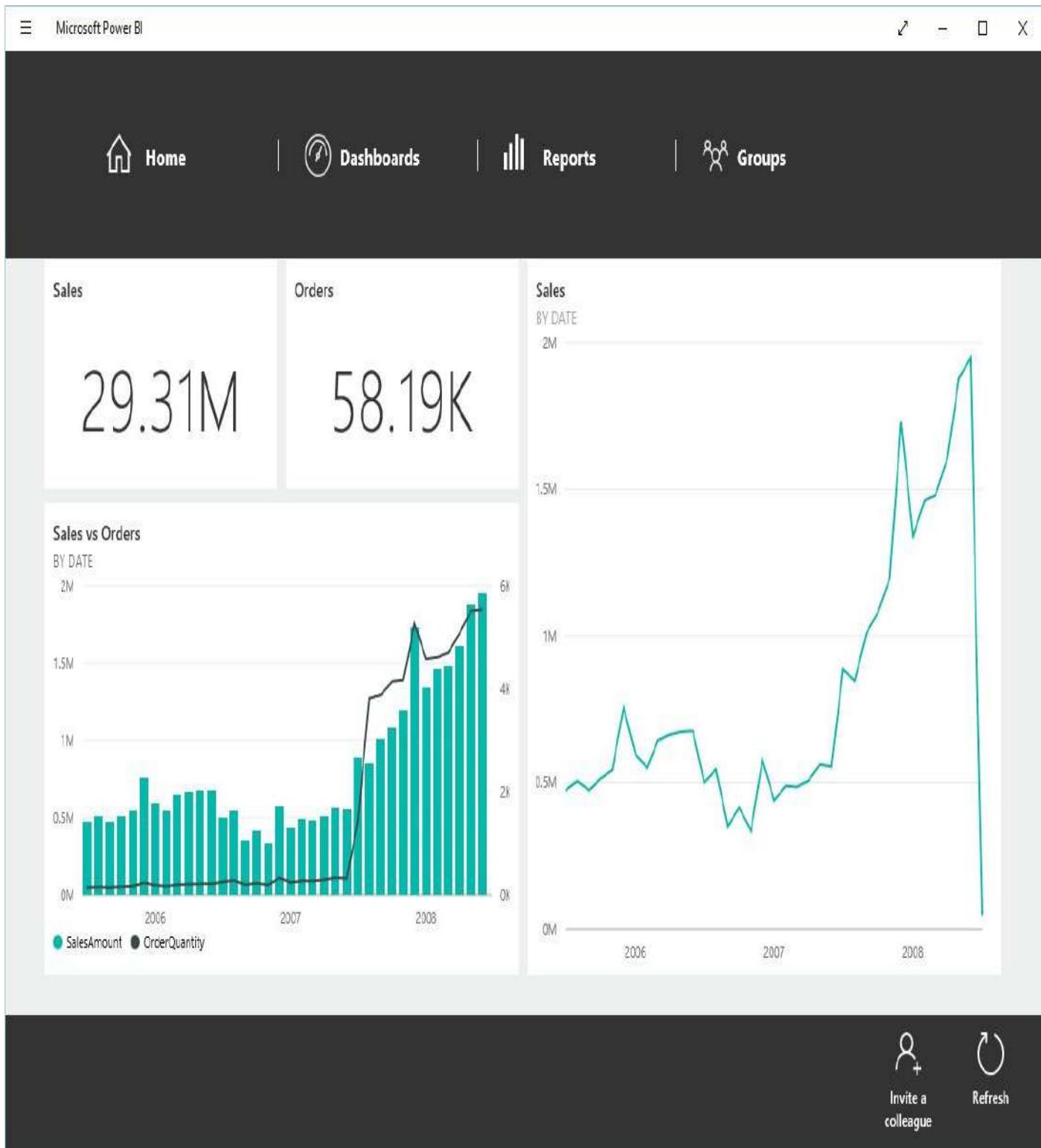


Figure 4.5 The Windows application targets Windows 8.1 and 10 devices.

While viewing reports and dashboards, the user has access to context-aware menus, also called “app commands”. For example, in **Figure 4.5**, I’m viewing the Internet Sales dashboard and the app commands (shown at the bottom of the screen) enable additional tasks, such as inviting a colleague and refreshing the dashboard.

4.2 Viewing Content

Power BI Mobile provides simple and intuitive interface for viewing reports and dashboards. As it stands, Power BI Mobile doesn't allow users to edit the published content. This shouldn't be viewed as a limitation because mobile display capabilities are limited, and mobile users would be primarily interested in viewing content. Next, you'll practice viewing the BI content you created in Chapter 3 using the Windows native app. As a prerequisite, install the Windows native app either on your Windows laptop or on a Windows device, such as a Windows tablet. For the best experience, I recommend you use a touch-enabled device, such as Microsoft Surface.

4.2.1 Getting Started with Power BI Mobile

When you open the Windows native app and sign in to Power BI, you'll be presented with the home page shown in **Figure 4.6**. Let's take a moment to get familiar with the home page.

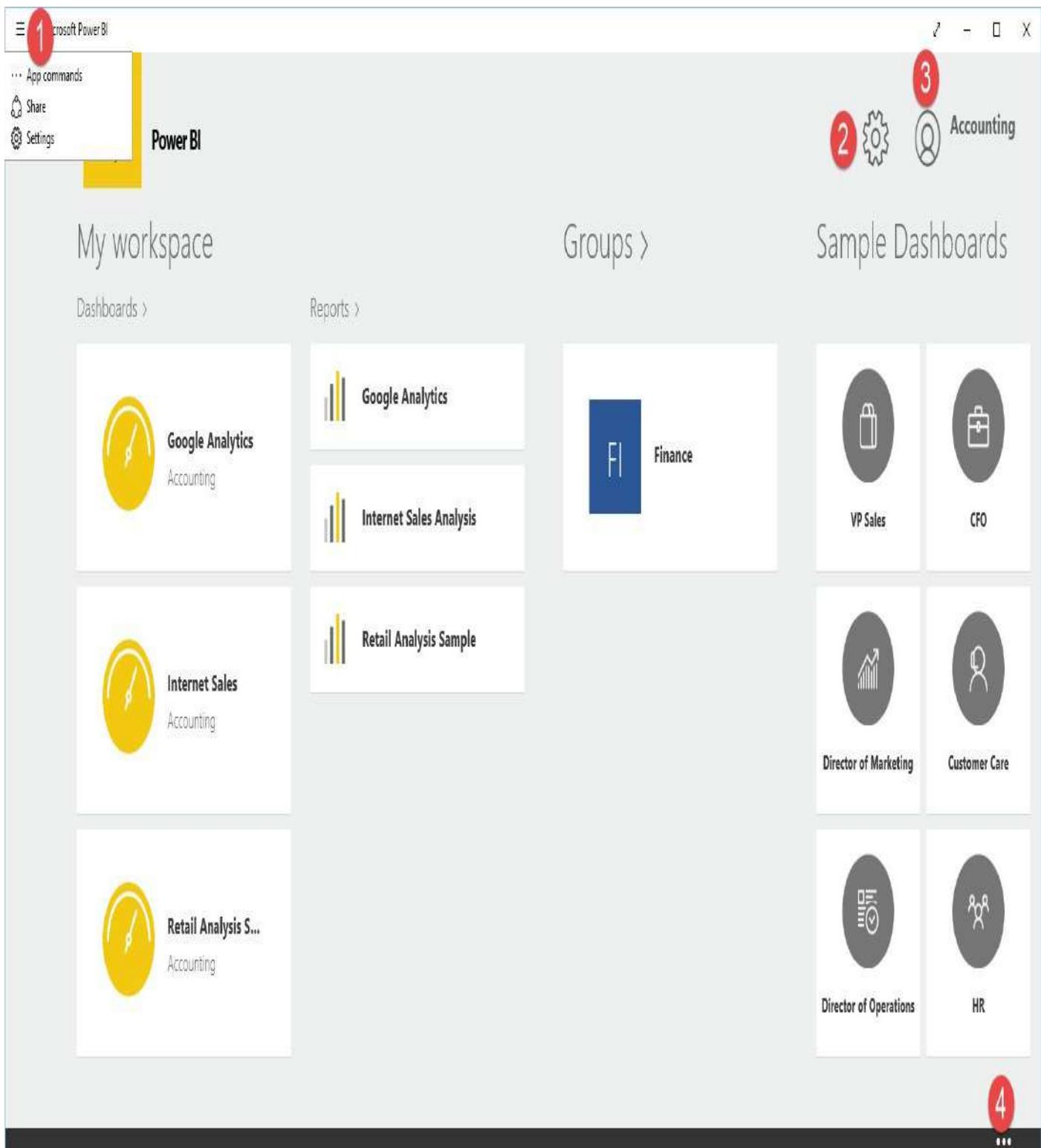


Figure 4.6 The home page of the Windows native app.

Understanding menus

The Home menu (1) is in the top-left corner. The App Commands menu shows the app commands. Another way to show the app commands is to click the “...” menu (item 4 on the screenshot) located at the bottom-right corner of the screen. On a touch-enabled device you can show the app commands by swiping down from the top of the screen or up from the bottom. The Share menu takes a screenshot of the current screen. Then, it navigates you to a flyout window that lets you send the screenshot via email or to OneNote. As a prerequisite for sending email from the Windows app, you need to configure Windows Mail.

The Settings menu is available also as a button in the top-right corner of the screen (see item 2 in the screenshot). It opens a flyout window that gives you access to certain settings, such as to see the version of the Windows app, to send usage data to Microsoft, to view your account settings (available also as a button in the top-right corner), and to send feedback to Microsoft.

My Workspace section

In Chapter 1, I covered how Power BI workspaces allow you to collaborate on the same content with other users. By default, all the content you create is placed in your personal workspace called My Workspace. The My Workspace section of the home page shows your dashboards and reports.

Groups section

If you have a Power BI Pro subscription you can also participate in other workspaces and groups. For example, someone might have created a Finance group and added you to the group. The Groups section allows you to view content that's available in the Finance workspace. I discuss groups and workspaces in more detail in Chapter 9 of this book.

Sample dashboards

The Sample Dashboards section includes a set of sample dashboards that Microsoft prepared to demonstrate the Power BI capabilities with sample data. You can view the dashboards but not the underlying reports.

4.2.2 Viewing Dashboards

Mobile users will primarily use Power BI Mobile to view dashboards that they've created or that are shared with them. Let's see what options are available for viewing dashboards.

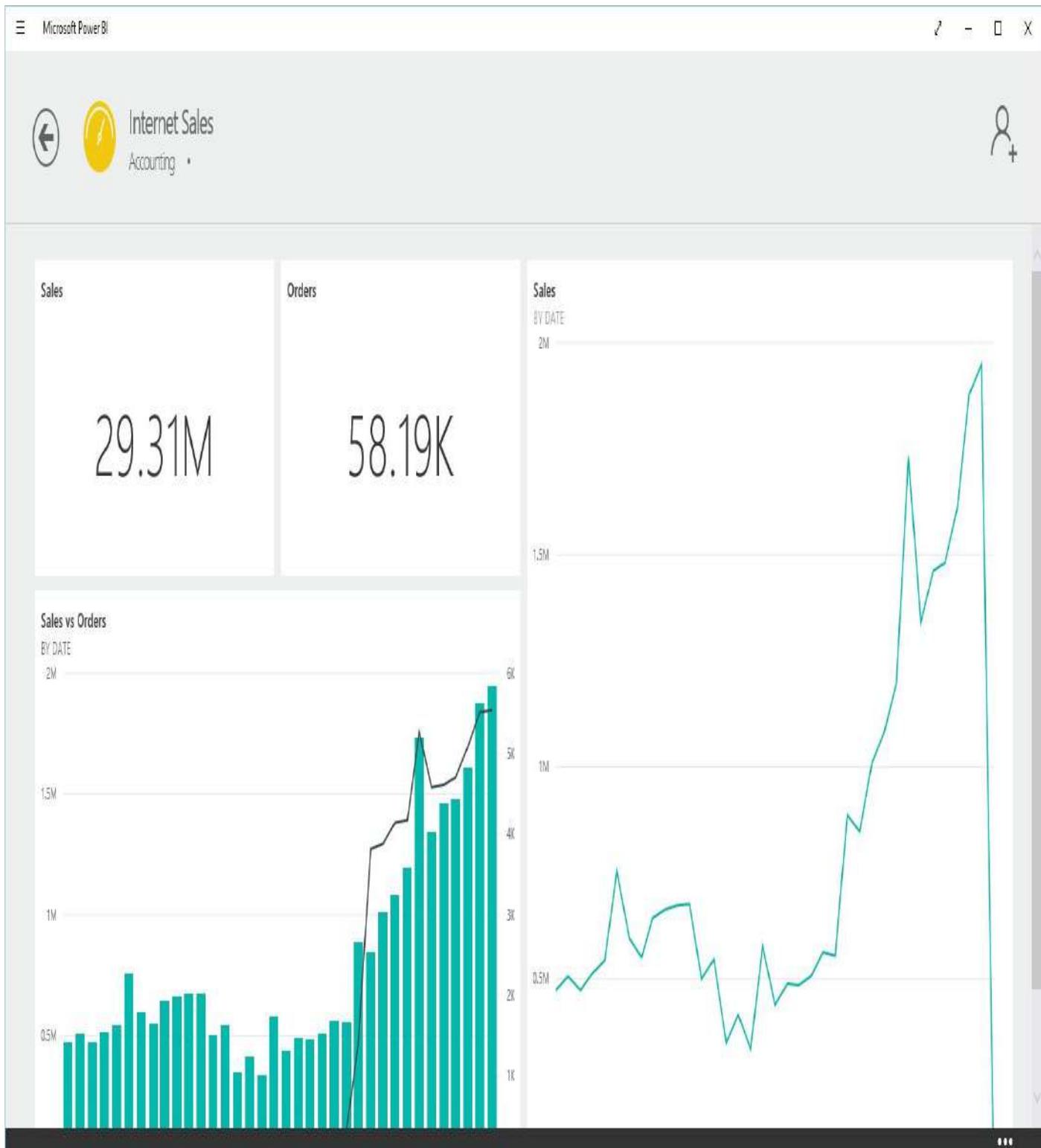


Figure 4.7 The Internet Sales dashboard open in Power BI Mobile.

Viewing dashboard tiles

This exercise uses the Internet Sales dashboard that you created in the previous chapter.

- 1.Click the Internet Sales dashboard to open it (see **Figure 4.7**). Power BI Mobile renders the dashboard content as it appears in the Power BI portal. However, the Q&A box is missing because Power BI Mobile doesn't currently support natural queries.
- 2.Click the Sales tile. As you would recall, clicking a tile in Power BI Portal, drills the tile through the underlying visualization (which could be pinned from a report or created via Q&A) so that you can see more details about your data. However, instead of drilling

through, Power BI Mobile pops the tile out so that you can examine the tile data (see **Figure 4.8**). Microsoft refers to this as “in-focus mode. Because the display of mobile devices is limited, the focus mode makes it easier to view and explore the tile data.

TIP If you want to navigate to the underlying report, click a tile to open it in focus mode and then click the Report menu in the top-right corner. This action opens the report in Reading View (Power BI Mobile doesn’t currently support Editing View). The Reports menu is available only for tiles created by pinning visualizations from existing reports. You won’t see the Report menu for tiles created with Q&A.

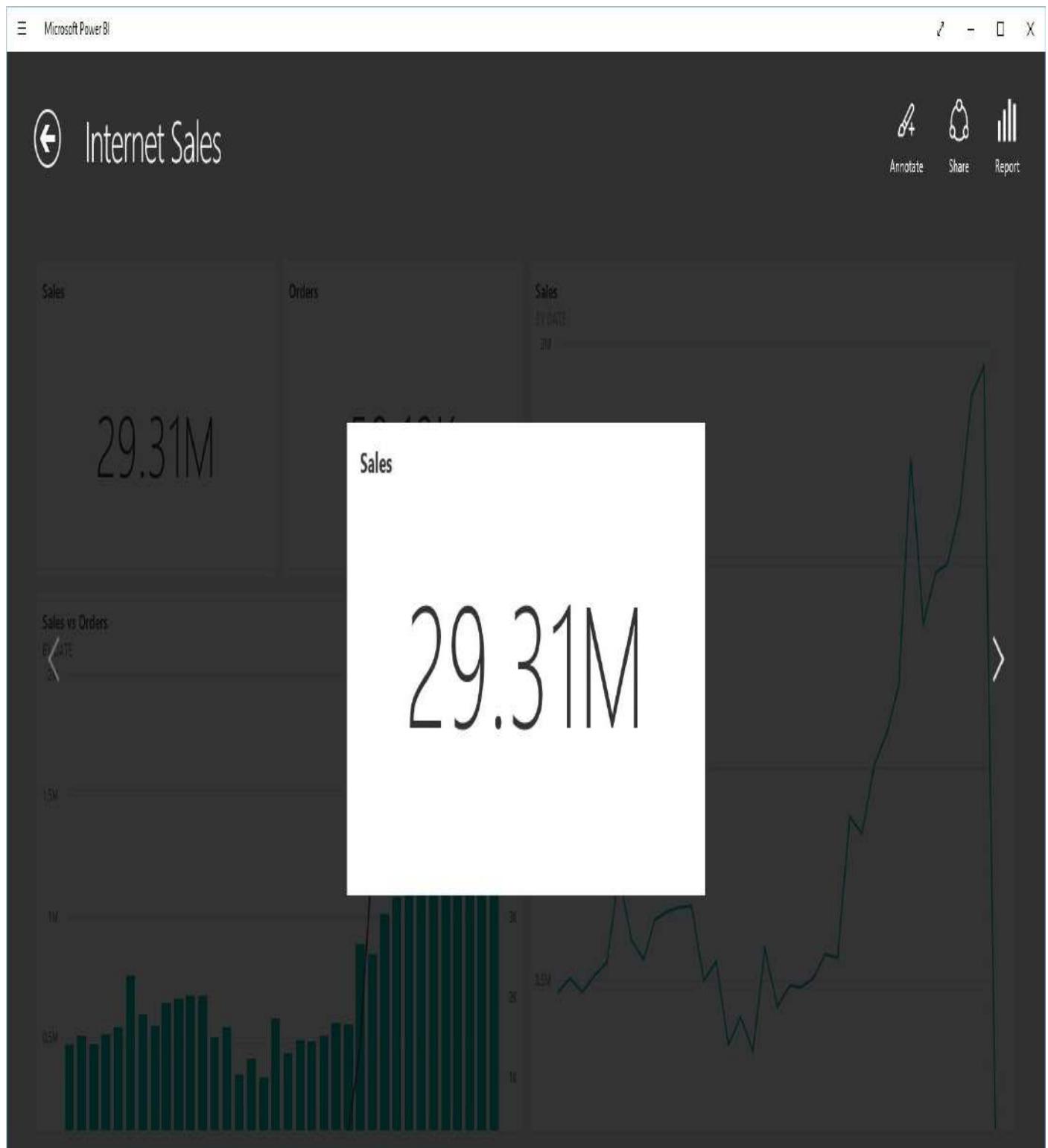


Figure 4.8 Clicking a tile will open the tile in focus mode.

3.Click the right arrow button. This navigates to the next tile, allowing you to examine the

dashboard content tile by tile.



Figure 4.9 You can drag the vertical bar to see the precise chart value.

Examining the data

It might be challenging to understand the precise values of a busy chart on a mobile device. However, Power BI Mobile has a useful feature that you might find helpful.

1. Navigate to the Sales Line Chart.
2. Drag the vertical bar to intersect the chart line for Jan 2006, as shown in **Figure 4.9**.

Notice that Power BI Mobile shows the precise value of the sales amount at the intersection. If you have a Scatter Chart (the Retail Analysis Sample dashboard has a Scatter Chart), you can select a bubble by positioning the intersection of a vertical line and a horizontal line. This allows you to see the values of the fields placed in the X Axis, Y Axis, and Size areas of the Scatter visualization. And for a Pie Chart, you can spin the chart to position the slices so that you can get the exact values.

4.2.3 Viewing Reports

As you've seen, Power BI Mobile makes it easy for business users to view dashboards on the go. You can also view reports. As I mentioned, Power BI Mobile doesn't allow you to edit reports; you can only open and interact with them in Reading View.

Opening a report

Let's open the Internet Sales Analysis report in Power BI Mobile:

1. Navigate to the Home page. You can do this by clicking the Back button in the top-left area of the screen or by clicking the Home menu in the top app command bar.
2. Under the Reports section, click the Internet Sales Analysis report to open it. Power BI Mobile opens the report in Reading View, as shown in **Figure 4.10**.



Figure 4.10 Power BI Mobile opens reports in Reading View but supports interactive features.

Using interactive features

Notice that you can't switch to Editing View to change the report. You shouldn't view this as a limitation because the small display size of mobile devices would probably make reporting editing difficult anyway. Although you can't change the report layout, you can interact with the report:

- 1.Hover over the Bar Chart and note how that the ellipsis “...” menu appears in the top-right corner. Recall that this menu allows you to sort the chart data.
- 2.Click any bar in the Bar Chart or a column in the Column Chart. Notice that automatic highlighting works and you can see the contribution of the selected value to the data shown in the other visualizations.
- 3.Click a column header in the Matrix visualization and note that every click toggles the column sort order.
- 4.The Filters pane let you change or remove filters. However, you can't add new filters and the Fields pane is missing. To add filters, you need to go to Power BI Service, make changes, and save the report.

4.3 Sharing and Collaboration

Power BI Mobile goes beyond just content viewing. It lets mobile workers share and collaborate while on the go. The sharing and collaboration features apply only to dashboards. They include sharing dashboards by e-mail and annotating dashboard tiles.

4.3.1 Sharing Dashboards

Remember from the previous chapter that Power BI Service allows you to share dashboards with your colleagues by sending email invitations. You can do the same with Power BI Mobile. Both options are interchangeable. For example, if you initiate dashboard sharing in Power BI Service, you can see the sharing settings in Power BI Mobile, and vice versa. Let's share a dashboard:

1. Back to the Home page, click the Internet Sales dashboard to open it.



2. Click the Invite a Colleague button in the top-right corner. Alternatively, expand the bottom app commands bar and then click the Invite a Colleague button on the black app command bar.

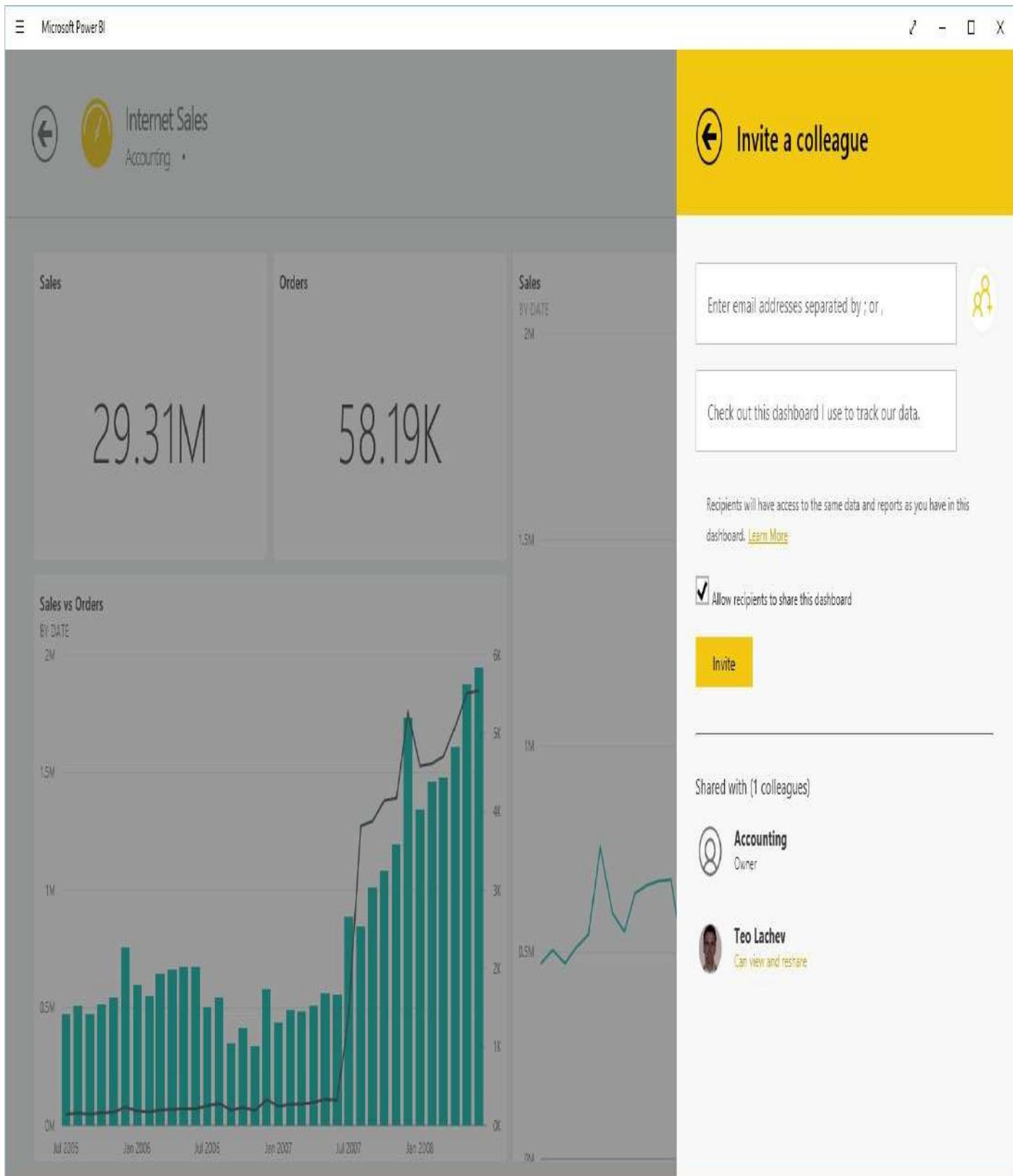


Figure 4.11 Power BI Mobile lets you share dashboards by email.

This action opens the “Invite a colleague” flyout window (see **Figure 4.11**), which asks for the same information as the Share Dashboard page in the Power BI portal. Specifically, you need to enter the recipient email addresses, type in an optional message, and decide if you want them to be able to re-share the dashboards. In addition, you can see who you shared the dashboard with. If you click a person, you can access a menu that allows you to stop sharing with that person.



Figure 4.12 You can annotate a dashboard tile by adding text, simple graphics, and smileys.

4.3.2 Annotating Tiles

While you're viewing the dashboard content, you might want to comment on some data and then share the comment with your coworkers. Annotations allow you to enter text, simple graphics, or a smiley, and then send a screenshot of the annotated tile by email. Annotations aren't saved on the dashboard. Once you exit the annotation mode, all

annotations are lost.

Adding annotations

Let's annotate a tile:

1. With the Internet Sales dashboard, click the “Sales vs Orders” Combo Chart to open it in focus mode.
2. Click the Annotate button in the upper right corner. This switches the tile to annotation mode.
3. Click the Smiley button and then click the first smiley icon to add a smiley to the tile. Position the smiley as shown in **Figure 4.12**.
4. With the smiley menu selected, click the arrow icon and add an arrow that points to the increased sales.
5. Click the aA icon at the bottom of the screen, and then enter the text, “well done!” Position the text next to the annotated bar. If you make a mistake, you can click the Undo button in the bottom-right corner. The trash bin button discards all your annotations.

Sharing screenshots

You can send screenshots to your colleagues by email. Let's send a screenshot of your annotations:

1. In annotation mode, click the Share button in the top right corner of the screen.
2. A flyout window asks you if you want to use Windows Mail or OneNote. Click Mail.
3. Power BI Mobile captures a screenshot of your annotation and inserts in the email body. Enter the recipient address and click Send (see **Figure 4.13**).

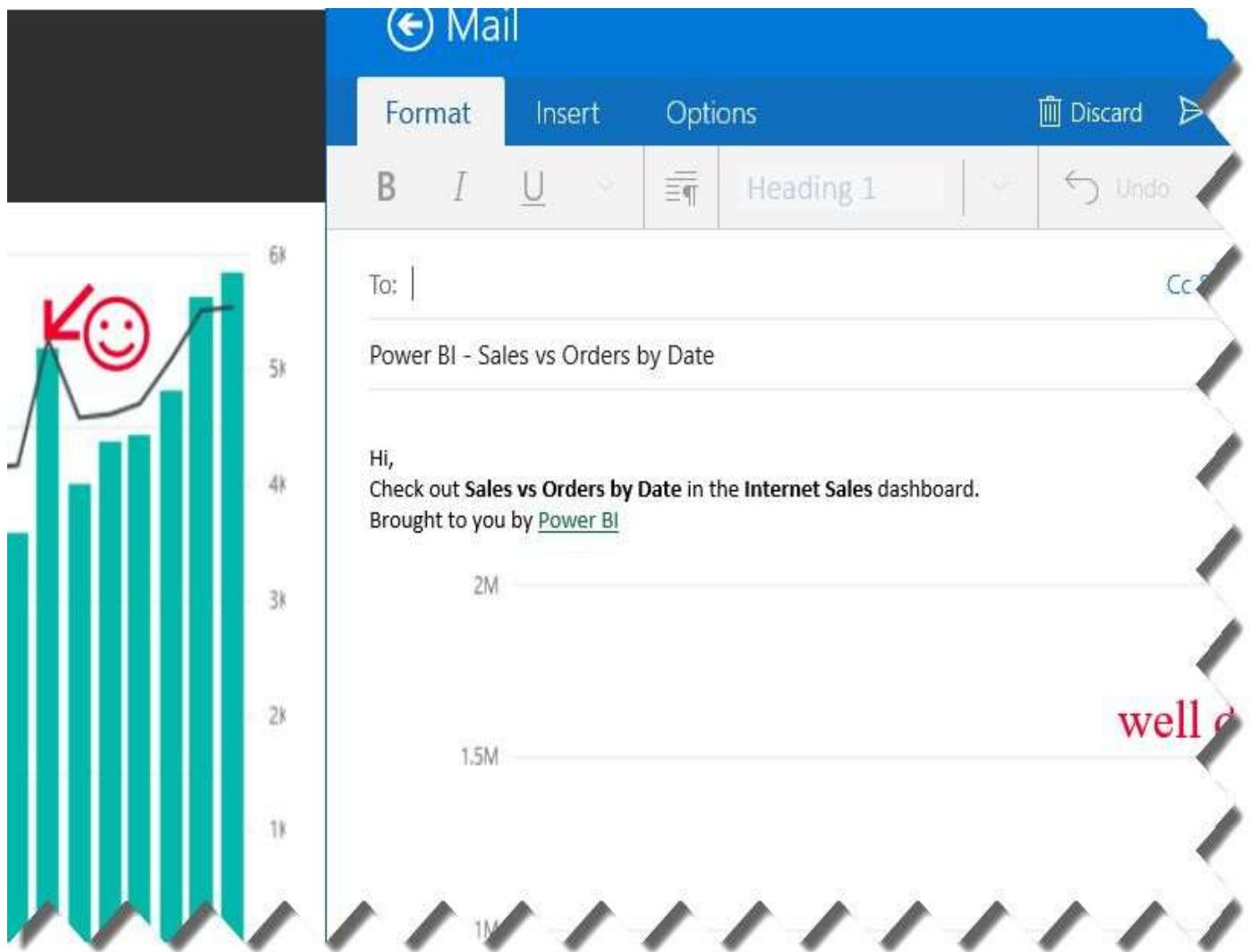


Figure 4.13 You can share screenshots to your coworkers by email.

4.4 Summary

Power BI Mobile is a collective name of three native applications for iOS, Android, and Windows devices. Power BI Mobile enriches the data analysis experience of mobile workers. Besides dashboard and report viewing, it allows users to share dashboards and annotate dashboard tiles. iPhone users can also create data alerts that notify them when data exceeds specific thresholds values that you specify.

This chapter concludes our Power BI tour for business users. Power BI has much more to offer than what you've seen so far, but it requires more knowledge. In the next part of the book, you'll see how data analysts (also referred to as power users) can create sophisticated data models to address more complicated data analytics needs.



Power BI for Data Analysts

If you consider yourself a data analyst or power user, welcome! This part of the book teaches you how to implement self-service data models with Power BI Desktop. As you've seen in the first part of the book, Power BI lets business users perform rudimentary data analysis without requiring data models. However, once real-life needs go beyond content packs and single datasets, you'll need a data model. Although you can still implement models with Excel, Power BI Desktop is the Power BI premium modeling tool for self-service BI. Packed with features, Power BI Desktop is a free tool that you can download and start using immediately to gain insights from your data.

If you have experience with Excel data modeling, you'll find that Power BI Desktop combines the best of Power Pivot, Power Query, and Power View in a simplified and standalone desktop tool. In this part of the book, I'll introduce you to Power BI Desktop and fundamental data modeling concepts. Next, you'll learn how to connect to data from a variety of data sources, ranging from relational databases, text files, Excel files, and cloud services.

Data quality is a big issue with many organizations and chances are that your data is no exception. Fortunately, Power BI Desktop has features that allow you to cleanse and transform dirty data before it enters your model, so you don't have to rely on someone else to clean the data for you. A self-service data model is rarely complete without important business metrics. Thanks to its support for Data Analysis Expressions (DAX), Power BI Desktop lets you implement sophisticated calculations using Excel-like formulas. Then you can explore your data by creating interactive reports as you can do with Power BI Service.

If you're already a Power Pivot user, you'll undoubtedly breeze through the content of this part of the book (this will be a review with some important new changes). As you'll find out, you can almost seamlessly transfer your Excel data modeling knowledge to Power BI Desktop. Again, that's because the underlying technology is the same. However, with Power BI Desktop, you'll always have the latest Power BI features because Microsoft updates it every month.

To practice what you'll learn, the book includes plenty of exercises that will walk you through the steps for implementing a self-service model for analyzing sales data.

Chapter 5

Data Modeling Fundamentals

As a first step to building a data model, you need to acquire the data that you'll analyze and report on. Power BI Desktop makes it easy to access data from a variety of data sources, ranging from relational databases, such as a SQL Server database, to text files, such as a comma-delimited file extracted from a mainframe system. The most common way of bringing data into Power BI Desktop is by importing it from relational databases. When the data isn't in a database, Power BI Desktop supports other data acquisition methods, including text files, cubes, data feeds, and much more. And for some fast databases, Power BI Desktop allows you to connect directly to the data source without importing the data.

In this chapter, you'll learn the fundamentals of self-service data modeling with Power BI Desktop. To put your knowledge in practice, you'll implement a raw data model for analyzing the Adventure Works reseller sales. You'll exercise a variety of basic data import options to load data from the Adventure Works data warehouse, a cube, an Excel workbook, a comma-delimited file, and even from a Reporting Services report. You'll find the resulting Adventure Works model in the \Source\ch05 folder.

5.1 Understanding Data Models

When you work with Power BI Desktop, you create a self-service data model with the data you need to analyze. As a first step, you need to obtain the data. The primary data acquisition option with Power BI Desktop is importing it. This option allows you to load and mash up data from different data sources into a consolidated data model. For example, Martin could import some CRM data from Salesforce, finance data from Excel reports, and sales data from the corporate data warehouse. Once Martin relates all of this data into a single model, he can then create a report that combines all of these three business areas. As you might realize, Power BI Desktop gives you tremendous power and flexibility for creating self-service data models!

Power BI Desktop allows you to import data from a variety of data sources with a few mouse clicks. While getting data is easy, relating data in your model requires some planning on your side in order to avoid inconsistent or even incorrect results. For example, you might have imported a Customer table and a Sales table, but if there isn't a way to relate the data in these tables, you'll get the same sales amount repeated for each customer. Therefore, before you click the Get Data button, you should have some basic understanding about the Power BI data modeling requirements and limitations. So let's start by learning some important fundamentals of data modeling. Among other things, they will help you understand why having a single monolithic dataset is not always the best approach and why you should always have a date table.

5.1.1 Understanding Schemas

I previously wrote that Power BI Desktop imports data in tables, similar to how Excel allows you to organize data into Excel tables. Each table consists of columns, also referred to as *fields* or *attributes*. If all the data is provided to you as just one table, then you could congratulate yourself and skip this section altogether. In fact, as you've seen in Part 1 of this book, you can skip Power BI Desktop and modeling because you can analyze a single dataset directly with Power BI Service. Chances are, however, that you might need to import multiple tables from the same or different data sources. This requires learning some basic database and schema concepts. The term "schema" here is used to describe the table definitions and how tables relate to each other. I'll keep the discussion light on purpose to get you started with data modeling as fast as possible. I'll revisit table relationships in the next chapter.

NOTE Importing all data as a single table might not require modeling but it isn't a best practice. Suppose you initially wanted to analyze reseller sales and you've imported a single dataset with columns such as Reseller, Sales Territory, and so on. Then you decide to extend the model with direct sales to consolidate reporting that spans two business areas. Now you have a problem. Because you merged business dimensions into the reseller sales dataset, you won't be able to slice and dice the two datasets by the same lookup tables (Reseller, Sales Territory, Date, and others). In addition, a large table might strain your computer resources as it'll require more time to import and more memory to store the data. At the same time, a fully normalized schema, such as modeling a product entity with Product, Subcategory, and Category tables, is also not desirable because you'll end up with many tables and the model might become difficult to understand and navigate. When modeling your data, it's important to find a good balance between business requirements and normalization, and that balance is the star schema.

Understanding star schemas

For a lack of better terms, I'll use the dimensional modeling terminology to illustrate the star schema (for more information about star schemas, see http://en.wikipedia.org/wiki/Star_schema). **Figure 5.1** shows two schema types. The left diagram illustrates a star schema, where the ResellerSales table is in the center. This table stores the history of the Adventure Works reseller sales, and each row represents the most atomic information about the sale transaction. This could be a line item in the sales order that includes the order quantity, sales amount, tax amount, discount, and other numeric fields.

Dimensional modeling refers to these tables as *fact tables*. As you can imagine, the ResellerSales table can be very long if it keeps several years of sales data. Don't be alarmed about the dataset size though. Thanks to the state-of-the art underlying storage technology, your data model can still import and store millions of rows!

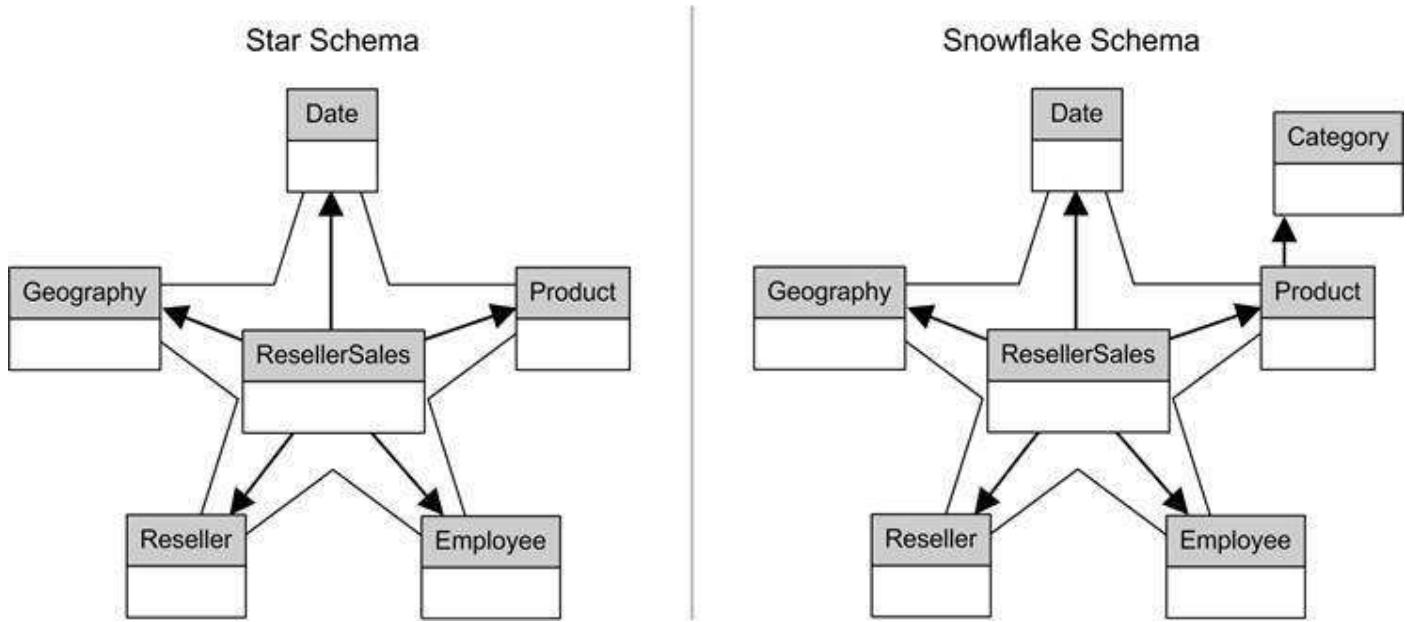


Figure 5.1 Power BI models support both star and snowflake schema types.

The ResellerSales table is related to other tables, called *lookup* or *dimension* tables. These tables provide contextual information to each row stored in the ResellerSales table. For example, the Date table might include date-related fields, such as Date, Quarter, and Year columns, to allow you to aggregate data at day, quarter, and year levels, respectively. The Product table might include ProductName, Color, and Size fields, and so on.

The reason why your data model should have these fields in separate lookup tables, is that, for the most part, their content doesn't need a historical record. For example, if the product name changes, this probably would be an in-place change. By contrast, if you were to continue adding columns to the ResellerSales table, you might end up with performance and maintenance issues. If you need to make a change, you might have to update millions of rows of data as opposed to updating a single row. Similarly, if you were to add a new column to the Date table, such as FiscalYear, you'll have to update all the rows in the ResellerSales table.

Are you limited to only one fact table with Power BI? Absolutely not! For example, you can add an InternetSales fact table that stores direct sales to individuals. In the case of

multiple fact tables, you should model the fact tables to share some common lookup tables so that you could match and consolidate data for cross-reporting purposes, such as to show reseller and Internet sales side by side and grouped by year and product. This is another reason to avoid a single monolithic dataset and to have logically related fields in separate tables (if you have this option). Don't worry if this isn't immediately clear. Designing a model that accurately represents business requirements is difficult even for BI pros but it gets easier with practice.

Understanding snowflake schemas

A *snowflake* schema is where some lookup tables relate to other lookup tables and not directly to the fact table. Going back to **Figure 5.1**, you can see that for whatever reason, the product categories are kept in a Category table that relates to the Product table and not directly to the ResellerSales table. One good motivation for snowflaking is that you might have another fact table, such as SalesQuota, that stores data not at a product level but at a category level. If you keep categories in their own Category table, this design would allow you to join the Category lookup table to the SalesQuota table, and you'll still be able to have a report that shows aggregated sales and quota values grouped by category.

Power BI supports snowflake schemas just fine. However, if you have a choice, you should minimize snowflaking when it's not needed. That's because snowflaking increases the number of tables in the model, making it more difficult for other users to understand your model. If you import data from a database with a normalized schema, you can minimize snowflaking by merging snowflaked tables. For example, you can use a SQL query that joins the Product and Category tables. But, if you import text files, you won't have that option because you can't use SQL. However, when you use Power BI Desktop, you can still handle denormalization tasks in the Query Editor or by adding calculated columns that use DAX expressions to accomplish the same goal, such as by adding a column to the Product table to look up the product category from the Category table. Then you can hide the Category table.

To recap this schema discussion, you can view the star schema as the opposite of its snowflake counterpart. While the snowflake schema embraces normalization as the preferred designed technique to reduce data duplication, the star schema favors denormalization or data entities and reducing the overall number of tables although this process results in data duplication (a category is repeated for each product that has the same category). Demormalization (star schemas) and BI go hand in hand. That's because star schemas reduce the number of tables and required joins. This makes your model faster and more intuitive.

Understanding date tables

Even if the data is given to you as a single dataset, you should strongly consider having a separate date table. A date table stores a range of dates that you need for data analysis. A date table typically includes additional columns for flexible time exploration, such as Quarter, Year, Fiscal Quarter, Fiscal Year, Holiday Flag, and so on. In addition, DAX time calculations, such as TOTALYTD, TOTALQTD, and so on, require a separate date table. A date table must meet several requirements:

- It must have a column of a Date data type.
- It must also have a day granularity, where each row in the table represents a calendar day.
- It must contain a consecutive range of dates you need for analysis, such as starting from the first day with data to a few years in the future. More importantly, a date table can't have gaps (it can't skip days). If it has gaps, DAX time calculations will produce wrong results.

There are many ways to create a date table. You can import it from your corporate data warehouse, if you have one. You can maintain it in an Excel file and import it from there. You can import it from the DateStream feed, as I'll show you in section 5.2.7. You can even generate it in the Query Editor using custom code written in the query language (referred to as "M"), as I'll show you in the next chapter. And you can have more than one date table in your model. This could be useful if you want to aggregate the same fact table by multiple dates, such as order date, ship date, and due date.

5.1.2 Introducing Relationships

Once you import multiple tables, you need a way to relate them. If two tables aren't related, your model won't aggregate data correctly when you use both tables on a report. To understand how to relate tables, you need to learn about Power BI relationships.

Understanding relationships

In order to relate two tables, there must be schema and data commonalities between the two tables. This isn't much different than joins in Microsoft Access or SQL Server. For example, you won't be able to analyze sales by product if there isn't a common column between the ResellerSales and Date tables that ties a date to a sale (see **Figure 5.2**).

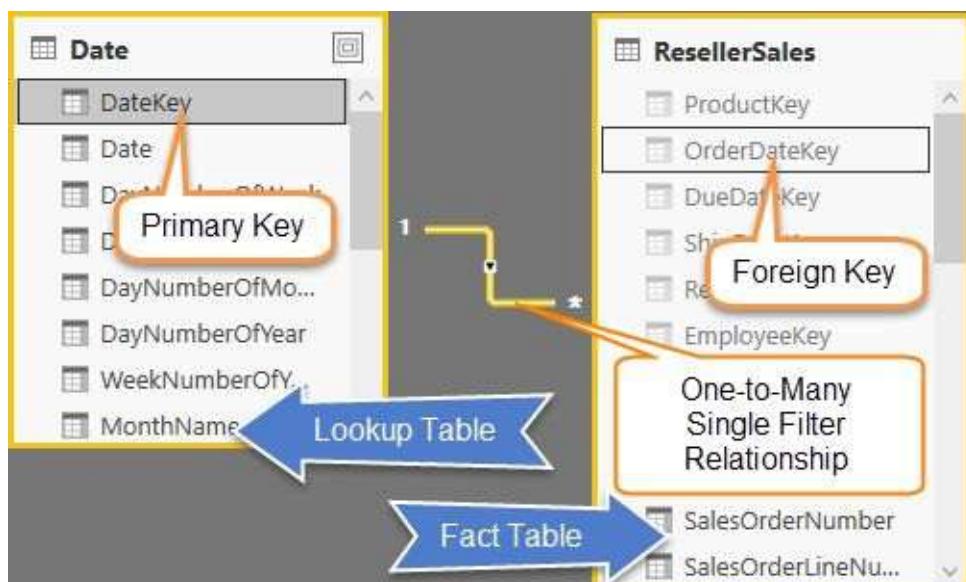


Figure 5.2 The DateKey column (primary key) in the Date table is related to the matching OrderDateKey column (foreign key) in the ResellerSales table.

If the underlying data source has relationships (referential integrity constraints) defined, Power BI Desktop will detect and carry them to the model. If not, Power BI is capable of auto-detecting relationships using internal rules. Of course, you can also create relationships manually. It's important to understand that your data model is layered on top of the original data. No model changes affect the original data source and its design. You only need rights to read from the data source so that you can import the data you need.

Understanding keys

Common columns in each pair of tables are called *keys*. A *primary key* is a column that uniquely identifies each row in a table. A primary key column can't have duplicate values. For example, the DateKey column uniquely identifies each row in the Date table and no other row has the same value. An employee identifier or an e-mail address can be used as a primary key in an Employee table. To join Date to ResellerSales, in the ResellerSales table, you must have a matching column, which is called a *foreign key*. For example, the OrderDateKey column in the ResellerSales table is a foreign key.

A matching column means a column in the fact table that has matching values in the lookup table. The column names of the primary key and foreign key don't have to be the same (values are important). For example, if the ResellerSales table has a sale recorded on 1/1/2015, there should be a row in the Date table with a DateKey of 1/1/2015. If there isn't, the data model won't show an error but all the sales that don't have matching dates in the Data table would appear under an unknown (blank) value in a report that groups ResellerSales data by some column in the Date table.

Typically, a fact table has several foreign keys, so it can be related to different lookup tables. For performance reasons, you should use shorter data types, such as integers or dates. For example, the DateKey column could be a column of a Date data type. Or if you're importing from a data warehouse database, it might have an Integer data type, with values such as 20110101 for January 1st, 2011, and 20110102 for January 2nd, 2011, and so on.

About relationship cardinality

Note that back in in **Figure 5.2**, the number 1 shows on the left side of the relationship towards the Date table and an asterisk (*) is shown next to the Reseller Sales table. This denotes a one-to-many cardinality. To understand this better, consider that one row (one date) in the Date table can have zero, one or many recorded sales in ResellerSales, and one product in the Product table corresponds to one or many sales in ResellerSales, and so on. The important word here is "many".

Although not a common cardinality, Power BI also supports a one-to-one relationship type. For example, you might have Employee and SalesPerson tables in a snowflake schema, where a sales person is a type of an employee and each sales person relates to a single employee. By specifying a one-to-one relationship between Employee and SalesPerson, you're telling Power BI to check the data cardinality. A one-to-one relationship also brings additional simplifications when working with DAX calculations, such as to let you interchangeably use the DAX RELATED and RELATEDTABLE functions.

About relationship cross filter direction

Note also that in **Figure 5.2**, there's an arrow indicator pointing toward the ResellerSales table. This indicates that this relationship has a single cross filtering direction between the Date and Reseller tables. In other words, the ResellerSales table can be analyzed using the Date table, but not the other way around. For example, you can have a report that groups sales by any of the fields of the Date table. However, you can't group dates by a field in the ResellerSales table, which is probably meaningless anyway.

Now suppose you want to know how many times a product is sold by date. You might be able to find the answer by counting the ProductKey field in the ResellerSales table without involving the Product table at all. But what if you need to find how many times a particular product model (the ModelName column in the Product table) was sold on a given date? To avoid adding this column to the ResellerSales table, you'd want to count on the ModelName field in the Product table. This is where a cross filtering direction set to Both can help. This cross filtering type is commonly referred to as a many-to-many relationship (many products can be sold on a single day and a single product can be sold on many days) and it's denoted by a double arrow indicator on the relationship (see **Figure 5.3**).

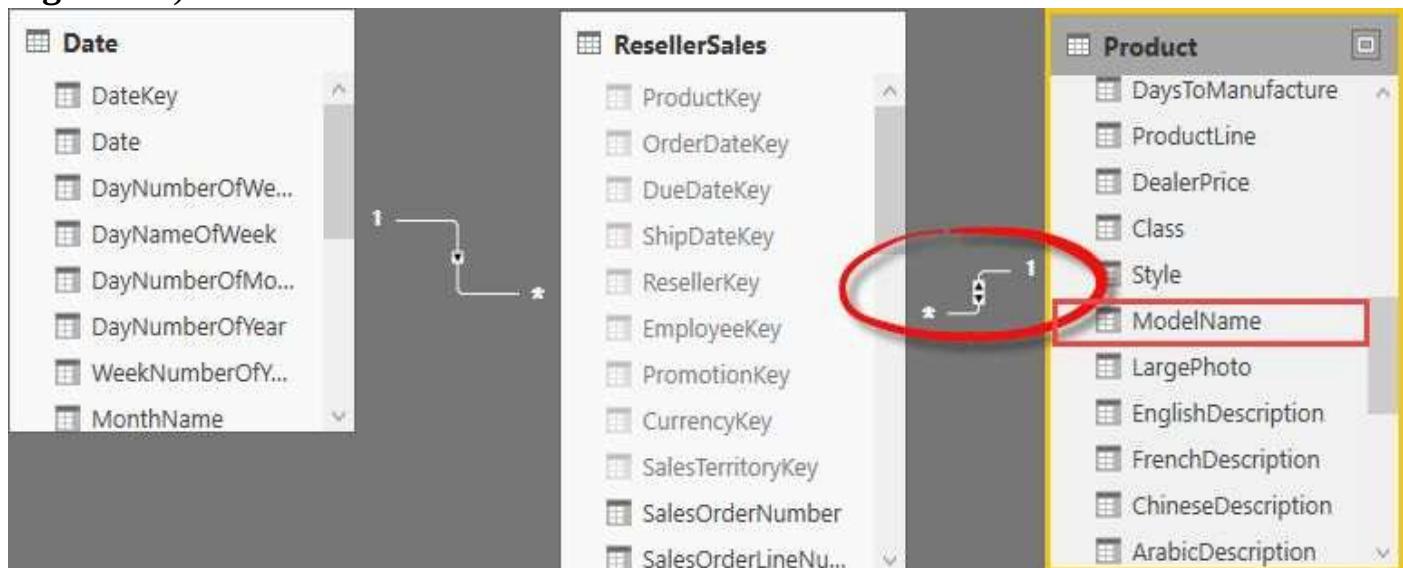


Figure 5.3 The relationship between the ResellerSales and Product table has a cross filtering direction set to Both.

Readers with prior experience in Power Pivot for Excel and Analysis Services Tabular, might recall that these tools didn't support declarative many-to-many relationships. This scenario required custom calculations so that aggregations are computed properly. Interestingly, the many-to-many relationship type (cross filtering set to both) is the default cross filtering type for new relationships in Power BI Desktop.

NOTE When Microsoft reached out to MVPs for a feedback on what the default cross filtering type should be, my immediate concern was the performance overhead associated with many-to-many relationships. As it turned out, there isn't any additional performance overhead, so I welcome the decision to set relationship cross filtering to Both by default. There are scenarios, such as relationships to a Date table and closed-looped relationships, which require Single cross filtering, as you'll see in Chapter 6.

That's all you need to know about data modeling for now. Next, let's see what options you have for connecting to your data.

5.1.3 Understanding Data Connectivity

In Chapter 2, I mentioned that Power BI supports two options for accessing data: data import and live connections, but this was a simplified statement. Technically, Power BI Service and Power BI Desktop support three options to connect to your data. Let me explain them to you now, because it's important to understand how they differ and when to use each one (if you have a choice). **Figure 5.4** should help you understand their differences.

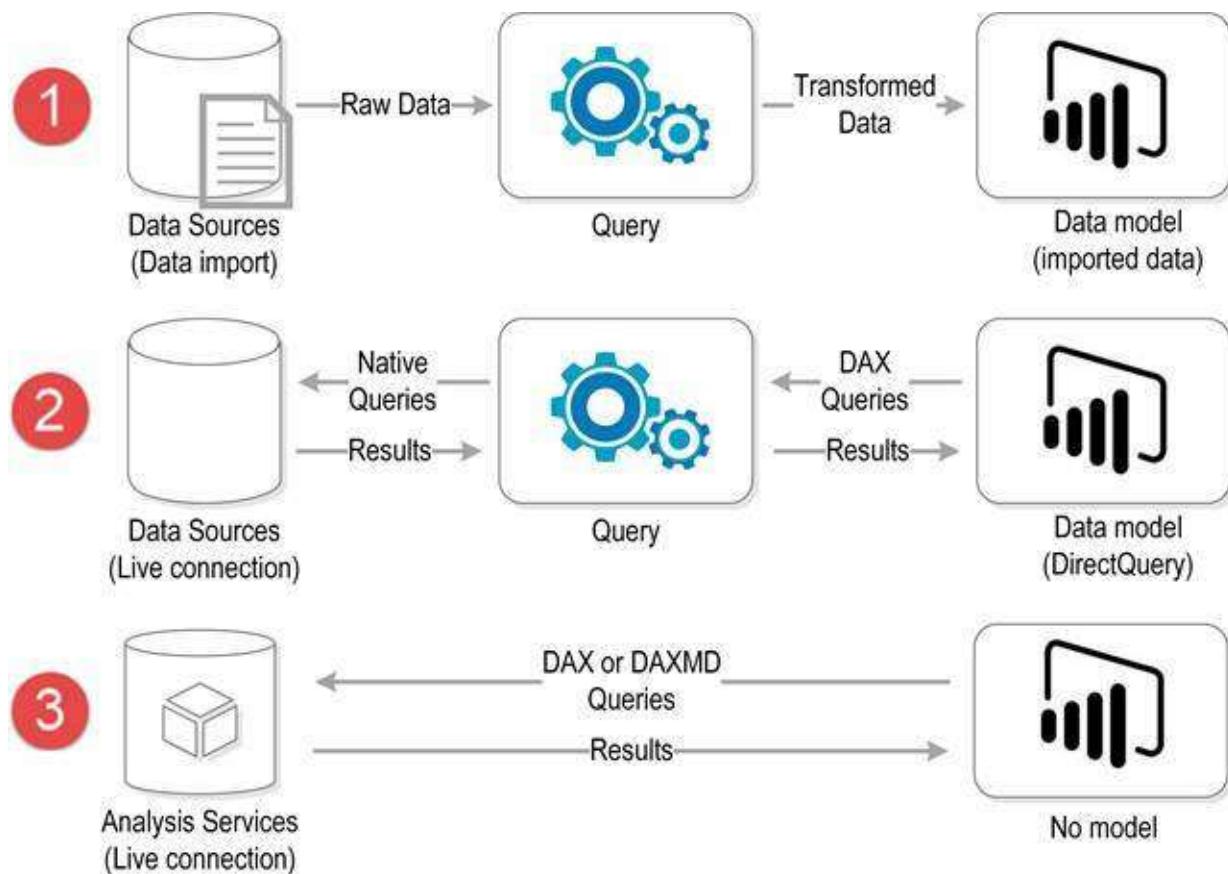


Figure 5.4 Power BI supports three connectivity options.

Importing data

Unless you connect to a single data source that supports direct connectivity, such as Analysis Services, SQL Server, Azure SQL Database, Spark on Azure HDInsight, or SAP Hana (the list is growing as Power BI evolves), you'll need to import data. This is option 1 in the **Figure 5.4** diagram. Unless you connect to the above data sources, you simply don't have another choice but to import the data. What if you need to relate data from a data source that supports direct connections to some other data coming from a data source that doesn't support direct access? You'd still need to import the data and you need to do so from *both* data sources. As it stands, Power BI Desktop only supports a live connection to a single data source, and you can't mix "live" and imported data.

When Power BI imports the data, it stores the data in the xVelocity in-memory data engine. The in-memory engine is hosted in an out-of-process Analysis Services instance that's distributed with Power BI Desktop. So that you can pack millions of rows, data is stored in a highly compressed format and loaded in memory when you analyze the data. When you import data, you can transform and clean it before it's loaded into the model.

Power BI Desktop always connects to raw data that you import via a connectivity component called a “query”. Think of a query as a layer between the raw data and your data model. Yes, the Power BI Desktop query is a successor of Excel Power Query, which you might be familiar with. However, unlike Excel, queries aren’t optional when you import data in Power BI Desktop.

NOTE Data analysts experienced with Excel data modelling might know that Excel has at least three ways to import data into a data model – Excel native import, Power Pivot Table Import Wizard, and Power Query. There hasn’t been a way in Excel to switch easily from one import option to another without recreating your table, such as from Table Import Wizard to Power Query, in order to use the Power Query transformation features. To simplify this, in Power BI Desktop, all the data import roads go through queries, which are the equivalent of Power Query in Excel. On the downside, even if you don’t transform the data, queries add some performance overhead when extracting the data from the data source.

Unless you use advanced query features, such as query functions to automate importing data from multiple files, each imported table has a corresponding query. So if you import three tables from a database and then import two files, you’ll end up with five queries. A query is your insurance against current and future data quality issues. Even if the data is clean, such as when you load it from a data warehouse, it might still require some shaping later on, and the query is the best place to perform these transformations.

Once the data is imported, the data is cached inside the Power BI Desktop file. Data exploration queries are handled internally by Power BI Desktop (or Power BI Service when the model is deployed). In other words, once the data is imported, Power BI Desktop doesn’t open connections and query the original data source unless you refresh the data.

NOTE Similar to Microsoft Office files, a Power BI Desktop file (*.pbix) is an archive zip file. If you rename it to have a zip extension and open it, you’ll see that the imported data is stored in the DataModel folder. xVelocity (the storage engine that powers Excel data models), Power BI, Analysis Services Tabular and SQL Server columnstore indexes) uses internal algorithms to compress the data. Depending on the cardinality of the data you import, expect a compression ratio of 5:1 or higher.

Connecting live via DirectQuery

You can connect Power BI Desktop to fast data sources that support direct (live) connectivity and then create reports that send native queries directly to these data sources (no data is cached in the model). This is option 2 in the **Figure 5.4** diagram. You can connect live to a single data source only. For example, if you connect to a SQL Server database, that’s the only data source you can use in your model. As I explained, you can’t connect to other data sources live or to import some other data. When you connect live to any of the supported data sources (other than SSAS), Power BI Desktop configures xVelocity in a special DirectQuery mode.

With DirectQuery, Power BI Desktop doesn’t import any data, and xVelocity doesn’t store any data. Instead, xVelocity generates DAX queries that the query layer translates to native queries. For example, if you connect Power BI Desktop to SQL Server with DirectQuery, the model will send T-SQL queries to SQL Server. Note that because Power BI uses the query layer in between, you can still perform basic data transformation tasks, such as column renaming and basic calculations.

NOTE Readers familiar with Analysis Services Tabular might know that DirectQuery originated in Tabular, but it had significant limitations, such as no DAX calculated columns, no time calculations, SQL Server as a targeted data source only, and others. Microsoft committed to extending DirectQuery in Power BI to support more data sources and to remove some of these limitations. Currently, Power BI DirectQuery doesn’t support DAX calculations, bi-directional

relationships, and custom queries.

When you connect to a data source that supports DirectQuery and before you load a table, Power BI Desktop will ask you how you want to get data: by importing data or by using DirectQuery (see **Figure 5.5**).

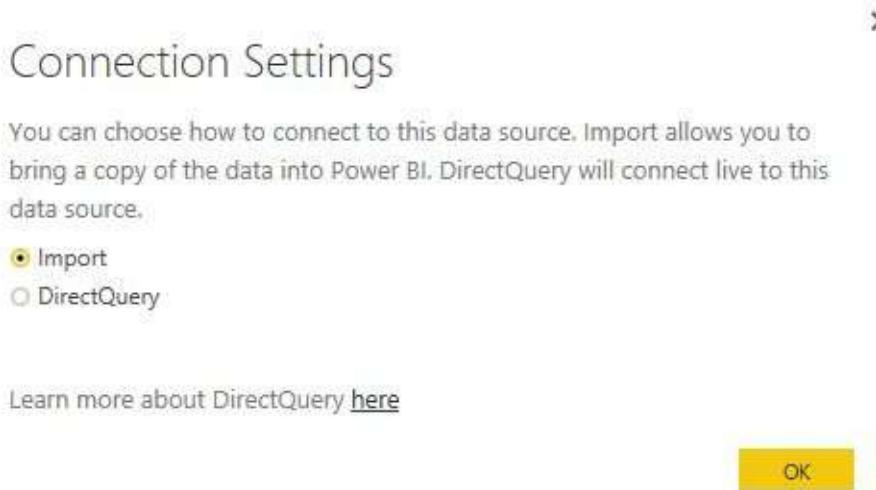


Figure 5.5 When connecting to a data source that supports DirectQuery, you need to decide how to access the data.

In this case, I'm connecting to SQL Server and Power BI Desktop asks me if I want to import the data or use DirectQuery. If I select DirectQuery, Power BI Desktop will auto-generate native queries as a result of my data exploration actions and will show me the results it gets from SQL Server. DirectQuery supports limited modeling capabilities using the Data View and the Fields pane. You can create and manage relationships, rename the metadata (table and column names), and perform basic data transformations.

Currently, DirectQuery is limited to a single data source. If you attempt to import data from another data source (even if it supports DirectQuery), Power BI Desktop will disallow this. You'll be given a choice to cancel the import process or to switch to a data import mode and import all the data.

If you publish a DirectQuery model to Power BI Service, the dataset needs the Enterprise Power BI Gateway to be installed on premises in order to connect to the original data source. If the gateway is not configured, the dataset will be published but it will be disabled (it will show grayed out in the Power BI navigation bar) and you won't be able to use it.

Connecting live to Analysis Services

Finally, a special live connectivity option exists when you connect live to Analysis Services Multidimensional or Tabular (see option 3 in the **Figure 5.4** diagram). In this case, the xVelocity engine isn't used at all. Instead, Power BI connects directly to SSAS. Power BI generates DAX queries when connected to Analysis Services (Multidimensional handles DAX queries through a special DAXMD interop mechanism). There is no additional query layer in between Power BI Desktop and SSAS.

In other words, Power BI becomes a presentation layer that's connected directly to

SSAS, and the Fields pane shows the metadata from the SSAS model. This is conceptually very similar to connecting Excel to Analysis Services. When Power BI discovers that you connect to SSAS, it opens the window shown in **Figure 5.6**. The Explore Live option connects you directly to SSAS.

Similar to DirectQuery, if you publish a model that connects live to SSAS, the dataset needs the Analysis Services connector to be installed on premises in order to connect to the original data source. If the connector is not configured, the dataset will be published but it will be disabled (it will show grayed out in the Power BI navigation bar) and you won't be able to use it. (The Enterprise Gateway will also support SSAS connections.)

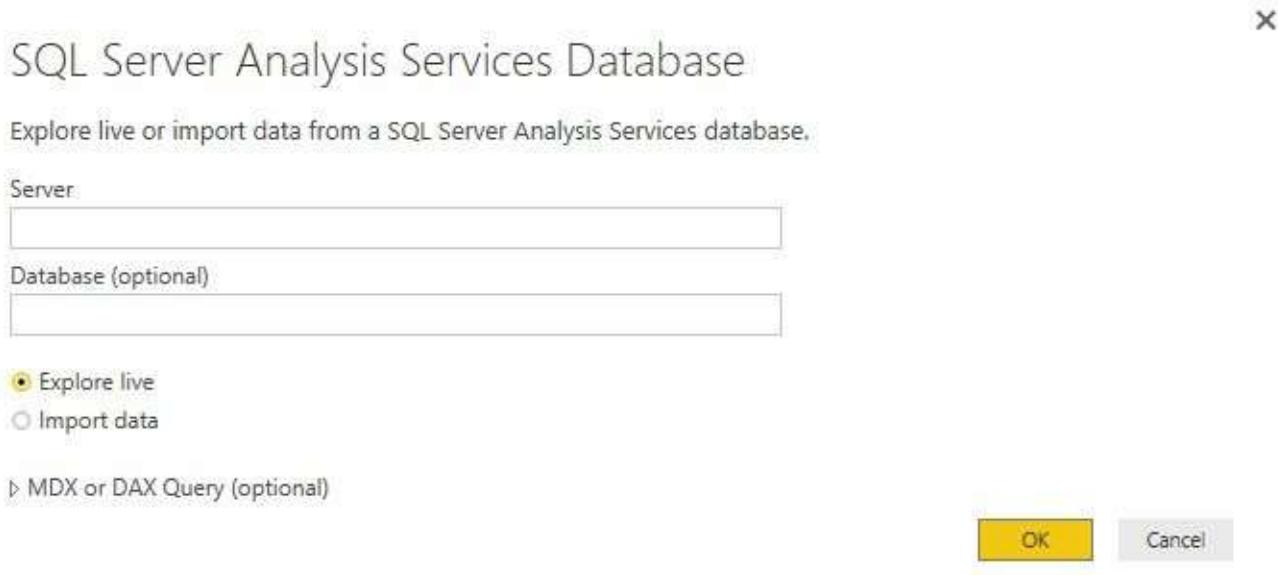


Figure 5.6 When connecting to SSAS, Power BI allows you to connect live or import the data.

Table 5.1 summarizes the key points about the two data connectivity options. Because most real-life needs would require importing data, this chapter focuses on this option exclusively. If you plan to connect to live data sources, your experience would be very similar to how you'll do this in Power BI Service (refer to the section "Using Live Connections" in Chapter 2).

Table 5.1 This table compares the three connectivity options.

Feature	Data import	Connecting Live (DirectQuery)	Connecting Live (SSAS)
Data sources	Any data source	SQL Server (on premises), Azure SQL Database, Spark on Azure HDInsight, Azure SQL Data Warehouse, SAP Hana	Analysis Services (Tabular and Multidimensional)
Usage scenario	When you connect to multiple data sources or a data source that doesn't support DirectQuery	When you connect to a single fast and/or large database that supports DirectQuery and you don't want to import data	When you connect to a single Analysis Services model
Queries for data cleansing and transformation	Available	Available (basic transformations only)	Not available
Connect to multiple data sources	Yes	No	No
Data storage	Data imported in xVelocity	Data is left at the data source	Data is left at the SSAS model
Connect to on-premises	Personal Gateway is required	Enterprise Gateway is required to	Analysis Services connector or

data from published models	to refresh data	connect to data	Enterprise Gateway is required to connect to SSAS models
Data modeling	Available	Available	Not available
Relationships	Available	Available	Not available
Business calculations in a data model	No restrictions	No DAX calculations (not supported currently)	No DAX calculations (usually defined in SSAS)
Data exploration	Handled internally by Power BI Desktop or Power BI Service when the model is deployed without making connections to the original data sources	Power BI Desktop/Power BI Service (published models) autogenerated native queries, sends them to the data source, and shows the results	Power BI Desktop/Power BI Service (published models) autogenerated DAX/DAXMD queries, sends them to SSAS, and shows the results

5.2 Understanding Power BI Desktop

As I mentioned in Chapter 1, data analysts have two options for creating self-service data models. If you prefer Excel, you can continue using the Excel data modeling capabilities and deploy Excel workbooks to Power BI Service (or SharePoint). Be aware though that Excel has its own roadmap so expect BI features to lag behind Power BI. By contrast, if you prefer to stay always on the latest BI features, consider Power BI Desktop, which combines the best of Excel Power Pivot, Power Query, and Power View in a standalone and freely available tool that Microsoft updates every month. However, as it stands, Power BI Desktop can publish only to Power BI Service and Pyramid Analytics (see section 5.2.2 for more information on Pyramid Analytics).

NOTE At the PASS Summit in October 2015, Microsoft revealed a long-term roadmap that will support publishing Power BI Desktop models to SQL Server Reporting Services. For more information, read the blog post “Microsoft Business Intelligence - our reporting roadmap” at <http://bit.ly/1NDEKZ9>.

5.2.1 Understanding Design Environment

Let’s get familiar with the Power BI Desktop design environment. **Figure 5.7** shows its main elements numbered in the typical order of steps you’d take to create and refine self-service data models.

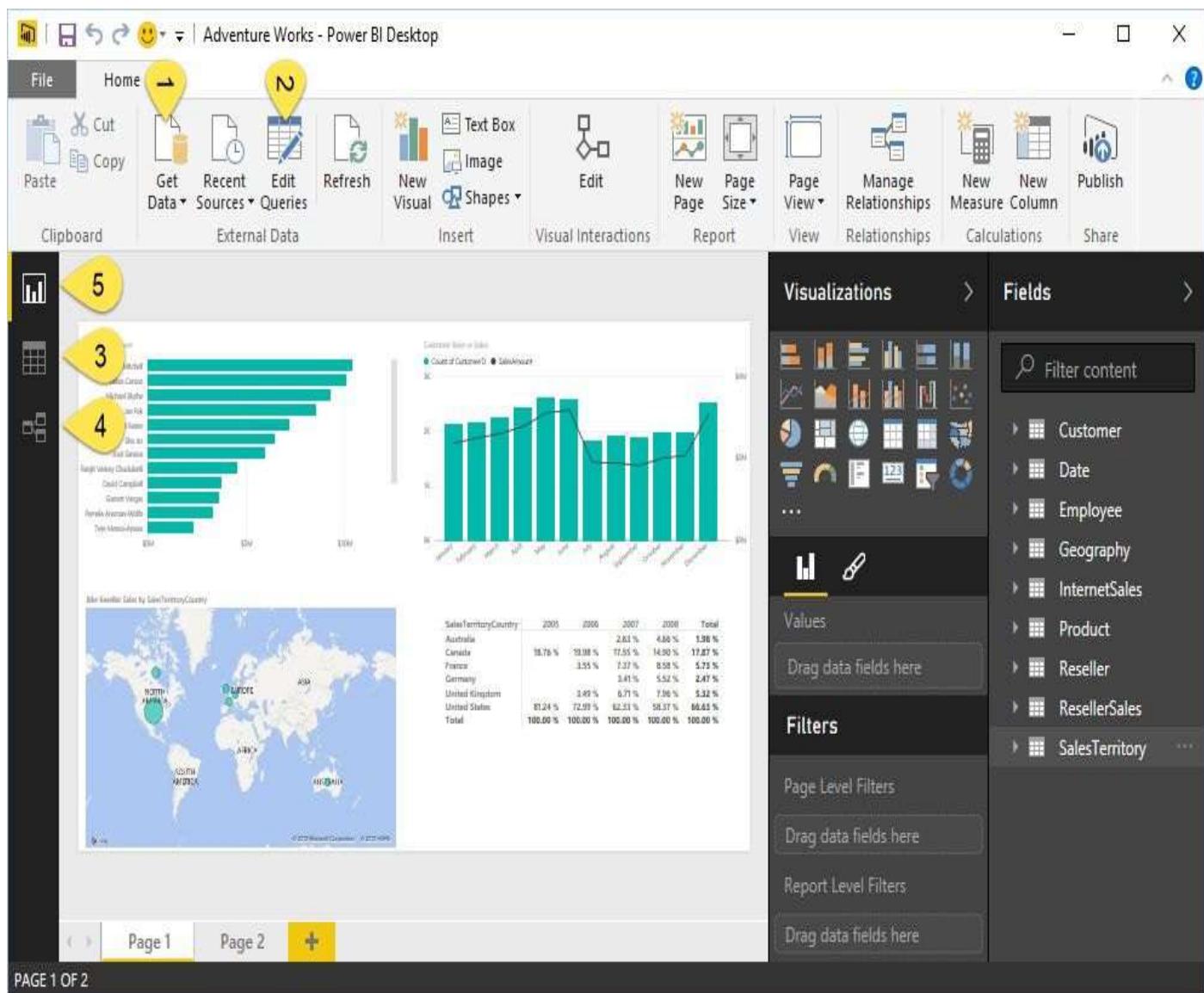


Figure 5.7 Power BI Desktop has the Reports, Data, and Relationships views.

Understanding data modeling steps

A typical self-service data modeling workflow consists of the following five main steps (each step refers back to the numbers in **Figure 5.7**):

1. Connect to data – The process starts with connecting to your data by clicking the Get Data button. Remember that when you import the data or use DirectQuery, Power BI Desktop creates a query for each table you import.
2. Transform data – If the data requires cleansing or transformations, click the Edit Queries button to modify the existing queries and perform data transformation tasks, such as replacing column values.
3. Refining the data model – Switch to the Data View to explore and refine the imported data, such as to change column data types and create business calculations. The Data View isn't available when you connect to DirectQuery data sources and to Analysis Services.
4. Creating relationships – As a part of refining the data model, if you import multiple tables you need to relate them either using the Relationships View or the Manage Relationships button in the ribbon's Home tab. Relationships aren't available when connecting live to Analysis Services.
5. Visualizing data – Finally, you build reports to visualize your data by switching to the Report View, which is available with all the data connectivity options.

Visualizations and Fields panes

The Visualizations and Fields panes should be familiar to you by now, as you've already worked with them in the first part of the book. The Fields pane shows the model metadata consisting of tables and fields. A field can be a table column or a calculation, such as Sales YTD. You can also use the Fields pane to create DAX calculations. The Fields pane is available when you're in the Reports and Data views.

The Visualizations pane (on the right) is only available when the Report View is active. It gives you access to Microsoft-provided and custom visualizations. You can click the ellipsis menu (...) below the visualization icons to import custom visualizations that you downloaded from the Power BI visuals gallery. You can use the Visualizations pane to configure the visual data options and formatting.

5.2.2 Understanding Navigation

Power BI Desktop has a simplified navigation experience using a ribbon interface that should be familiar to you from other Microsoft products, such as Excel. But if you come from Excel data modeling, you'll undoubtedly appreciate the change toward simplicity. There is no more confusion about which ribbon to use and which button to click! By the way, if you need more space, you can minimize the ribbon by clicking the chevron button in the top-right corner or by pressing Ctrl-F1.

Understanding the ribbon's Home tab

The ribbon's Home tab (see **Figure 5.7** again) is your one-stop navigation for common tasks. The Clipboard group allows you to copy and paste text, such as a DAX calculation formula. Currently, it doesn't allow you to create tables by pasting data as you can with Excel Power Pivot.

The External Data group allows you to connect to data (the Get Data button). The Edit Queries button opens a separate Query Editor window so that you can edit queries, if you want to cleanse or transform data. The Recent Sources button let you bypass the first Get Data steps, such as when you want to import additional tables from the same database. The Refresh button refreshes imported data to synchronize it with changes to the data source.

The next four groups (Insert, Visualizations, Report, and Page View) are available when the Report View is active. The Insert group allows you to insert new visuals to a report (alternatively, you can click the visual icon in the Visualizations pane). You can also insert graphical elements, such as text boxes, images, and shapes. Similar to the Power BI Service Visualizations menu (also known as "Brushing and Linking"), the Visualizations group allows you to customize the behavior of the page interactive features, such as interactive highlighting. Use the Report group to add new pages to the report and to change the report size. The Page View button lets you change the preview experience, such as to fit to page.

The Manage Relationships button allows you to relate tables in your model. The Calculations group is for creating DAX calculations. And the Publish button lets you publish your model to Power BI.

Understanding the ribbon's Modeling tab

The ribbon's Modeling tab (**Figure 5.8****Figure 5.14**) allows you to extend and refine your data model. It's available only when the Data View is active.

CustomerKey	GeographyKey	CustomerID	Title	FirstName	MiddleName	LastName	NameStyle
11471	207	AW00011471		Latasha		Suarez	False
11602	135	AW00011602		Larry		Gill	True
11603	244	AW00011603		Geoffrey		Gonzalez	False
11604	275	AW00011604		Edgar		Sanchez	False

Figure 5.8 The ribbon's Modeling tab is for performing data modeling tasks, such as creating calculations.

The Manage Relationships button is available here as well. The Calculations group gives

you access to DAX features to create calculated columns, measures, tables, and to define custom sorting. The Formatting group lets you change the data type and formatting of a table column, such as to format a sales amount with two decimal places. And the Properties group is for changing the purpose of specific columns, such as to tell Power BI Desktop that a Country column has a Country category so that a map knows how to plot it.

Understanding the File menu

The File menu (see **Figure 5.9**) gives you access to additional tasks.

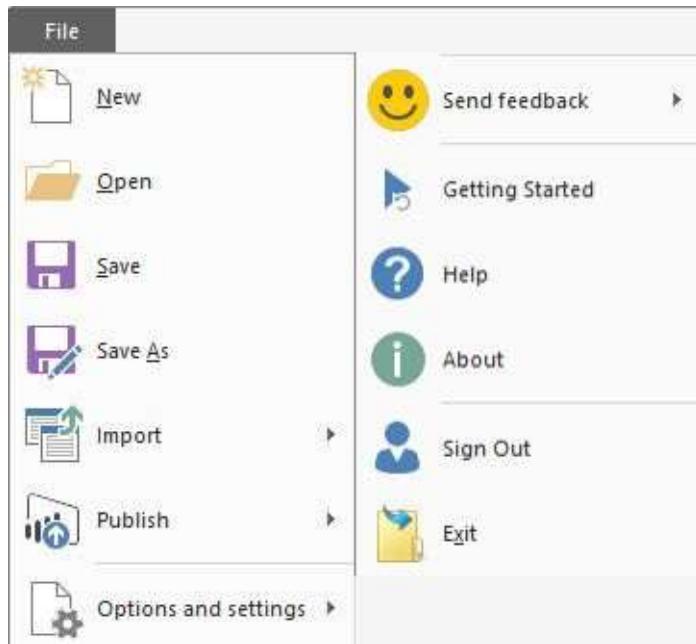


Figure 5.9 The Power BI Desktop File menu.

Use the New menu to create a new Power BI Desktop file and the Open menu to open an existing file. The Save menu saves changes to the current file, and the Save As menu saves the file as another file. If you’re transitioning to Power BI Desktop from Excel, the Import menu allows you to import the Excel data model into Power BI Desktop. You can use the Import menu as another way to import custom visuals.

The Publish menu is for publishing the data model to Power BI Service (same as the Publish button in the ribbon’s Home tab). Remember that although Power BI Desktop can load as much data as fits in your computer memory, Power BI Service caps the file size (currently to 250 MB), so be aware of this limitation when you import a lot of data. The Publish menu allows you to also publish the model to Pyramid Analytics. Pyramid Analytics is a third-party vendor who offers a Web-based on-premises business analytics suite for analytics, dashboards, and reporting. If you’ve purchased Pyramid Analytics, this option allows you to publish your Power BI Desktop models to their server.

The “Options and settings” menu lets you configure certain program and data source features, as I’ll discuss in the next section. The Send Feedback menu lets you share and rate your Power BI Desktop experience with Microsoft (you can send feedback and any issues directly to Microsoft). The Getting Started menu brings you to the Power BI Desktop splash screen which has links to tutorials and forums. The Help menu navigates to the Power BI Desktop Help documentation. Use the About menu to see the version of the installed Power BI Desktop. The Sign Out menu signs you out of Power BI Service in

case you want to sign under another account when you publish the model to Power BI. And the Exit menu closes Power BI Desktop.

Understanding Options and Settings menu

Currently, the “Options and settings” menu has two submenus: Option and Data Source Settings. The Data Source Settings menu lets you change certain security settings of the data sources used in the current model that you initially specified in Get Data. For example, you can use it switch from Windows authentication to the standard authentication that uses a login and password. For data sources that support encrypted connections, such as SQL Server, it allows you to change the connection encryption settings.

The Options menu brings you to the Options window (see **Figure 5.10**) that allows you to change various program features and settings. I’ll cover most of them in appropriate places in this and subsequent chapters but I’d like to explain some now.

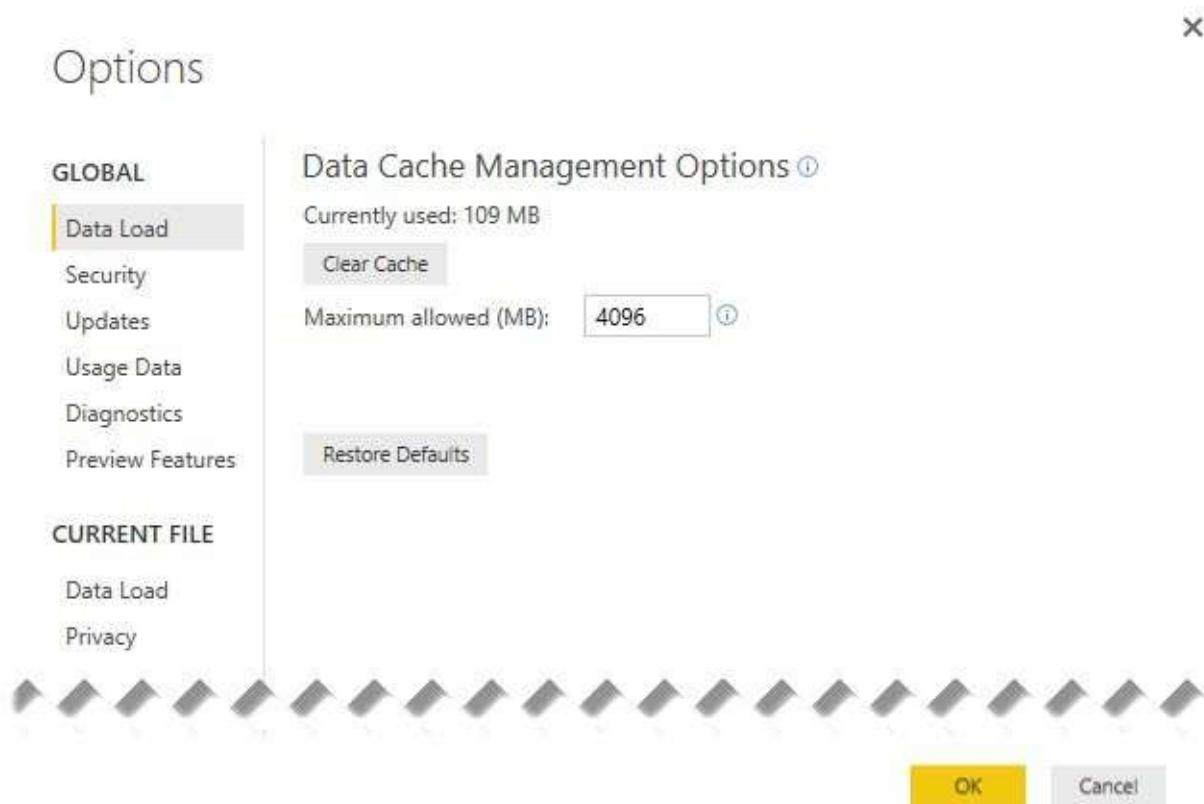


Figure 5.10 Use the Options window to configure various Power BI Desktop features.

The Updates tab allows you to configure Power BI Desktop to receive a notification when there’s a Power BI Desktop update (enabled by default). If you experience issues with Power BI Desktop, you can enable tracing from the Diagnostics tab and send the trace to Microsoft. An interesting setting is Preview Features. It allows you to enable and test features that Microsoft enables for testing with the goal of getting your feedback to help Microsoft improve the feature before enabling it by default.

Now that I’ve introduced you to Power BI Desktop and data modeling, let’s start the process of creating a self-service data model by getting the data!

5.3 Importing Data

Now let's go back to Martin, who's a data analyst with Adventure Works. Martin realizes that to perform comprehensive sales analysis, he needs data from different data sources. He'll implement a self-service data model and he'll import data from multiple data sources, including data residing in the data warehouse, an Analysis Services cube, flat files, and so on.

5.3.1 Understanding Data Import Steps

Power BI Desktop makes it easy to import data. For those of you who are familiar with Excel data modeling, the process is very similar to using Power Query in Excel. Importing data involves the following high-level steps, which you need to repeat them for each data source:

1. Choose a data source
2. Connect to the data
3. (Optional) Transform the raw data if needed
4. Load the data into the data model

Choosing a data source

Power BI Desktop can import data from a plethora of data sources with a few clicks and without requiring any scripting or programming skills. You can start the process by clicking the Get Data button in the Power BI Desktop ribbon. The most common data sources are shown in the drop-down menu (see **Figure 5.11**) but many more are available when you click the More menu. To make it easier to find the appropriate data source, the Get Data window organizes them in File, Database, Azure, and Other categories.

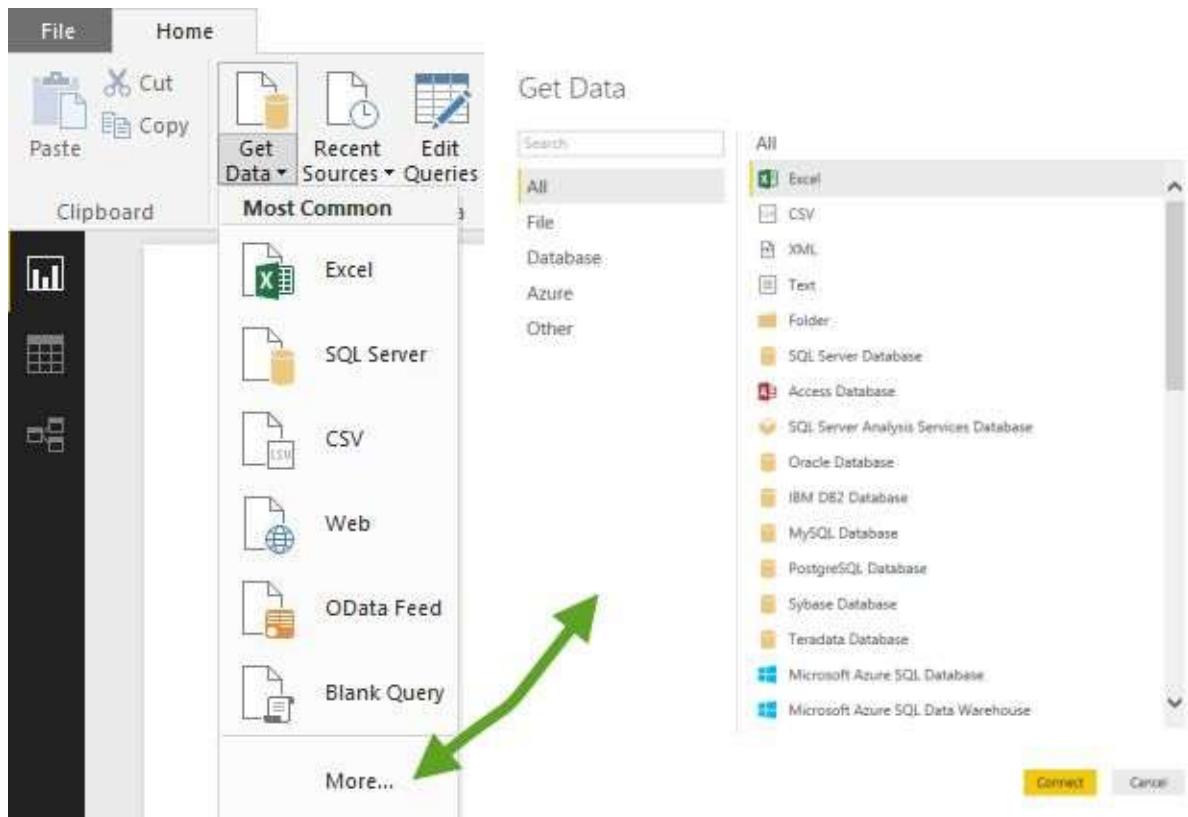


Figure 5.11 Power BI Desktop can connect to a variety of data sources without requiring any scripting or programming.

Table 5.2 summarizes the currently supported data sources but expect the list to grow because Microsoft adds new data sources on a regular basis. It's important to note that every data source requires installing appropriate connectivity software (also called provider or driver). Chances are that if you use Microsoft Office or other Microsoft applications, you already have drivers to connect to SQL Server, Excel files, and text files. If you have to connect to the data sources from other vendors, you need to research what connector (also called driver or provider) is needed and how to install it on your computer. For example, connecting to Oracle requires Oracle client software v8.1.7 or greater on your computer.

TIP Because Power BI Desktop borrowed the Power Query features from Excel, you can find more information about the prerequisites related to connectivity in the “Import data from external data sources (Power Query)” document by Microsoft at <http://bit.ly/1FQrjF3>.

What if you don't find your data source on the list? If the data source comes with an ODBC connector, try connecting to it using the ODBC option. Although the data source might not be officially supported by Microsoft, chances are that you will be able to connect via ODBC. Or, if it comes with an OLE DB driver, try the ODBC for OLE DB Microsoft driver, which is one of the ODBC drivers that come preinstalled with Windows.

Table 5.2 This table summarizes the data sources supported by Power BI Desktop.

Data Source Type	Data Sources
File	Excel, CSV, XML, other delimited and fixed-width text files, a list of files in a folder
Database	SQL Server, Microsoft Access, Analysis Services (Multidimensional, Tabular, PowerPivot workbooks deployed to SharePoint), Oracle, IBM DB2, MySQL, PostgreSQL, Sybase, Teradata, SAP Hana, and other ODBC-compliant databases
Azure	SQL Database, SQL Data Warehouse, Azure Marketplace, HDInsight, Blob Storage, Table Storage, HDInsight Spark, DocumentDB, Data Lake Store
Other	Web, SharePoint list, OData feed, Hadoop File from HDFS, Active Directory, Microsoft Exchange, Dynamics CRM Online, Facebook, Google Analytics, Salesforce, ODBC, RScript, appFigures, GitHub, MailChimp, Quickbooks Online, SweetIQ, Twilio, Zendesk, Marketo, Spark

Connecting to data

Once you select your data source you want to connect to, the next step depends on the type of the data source. For example, if you connect to a database, such as SQL Server, you'll see a window that looks like the one in **Figure 5.12**.



Figure 5.12 Connecting to a database requires a server name.

The only required field is the database server name that will be given to you by your database administrator. You can also specify the database name or a custom SQL statement. For example, if you need to execute a SQL Server stored procedure, you can enter the following statement:

```
exec <StoredProcedureName> parameter1, parameter 2, ...
```

Specifying credentials

If you connect to a database, the next step asks you to enter your credentials as instructed by your database administrator. For example, when connecting to SQL Server, you can use Windows credentials (behind the scenes it uses your Windows login) or standard authentication (user name and password). Power BI Desktop shows the “Access a SQL Server Database” window (see **Figure 5.13**) to let you specify how you want to authenticate. Note the Windows tab allows you specify alternative Windows credentials. This could be useful if the database administrator has given you the credentials of a trusted Windows account. The Database tab lets you use standard authentication.

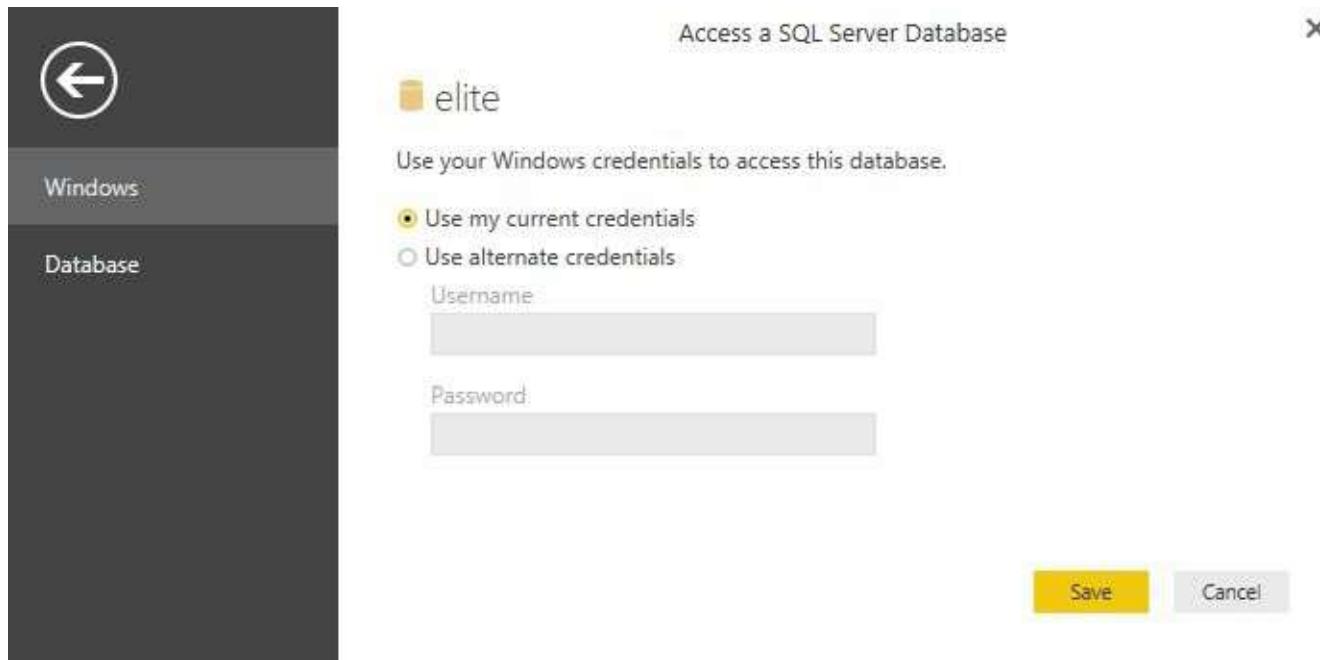


Figure 5.13 SQL Server supports Windows and standard authentication.

If you connect to a data source that it's configured for encrypted connections, such as SQL Server, you'll be asked if you want to encrypt the connection while data is imported. If the data source doesn't support encrypted connections, Power BI Desktop will warn you about it.

TIP Once you've connected to a data source, you don't need to use Get Data to import additional tables. Instead, use Recent Sources button in the Power BI ribbon. If you need to change the data source credentials or encryption options, use the File ð Options and Settings ð Data Source Settings menu.

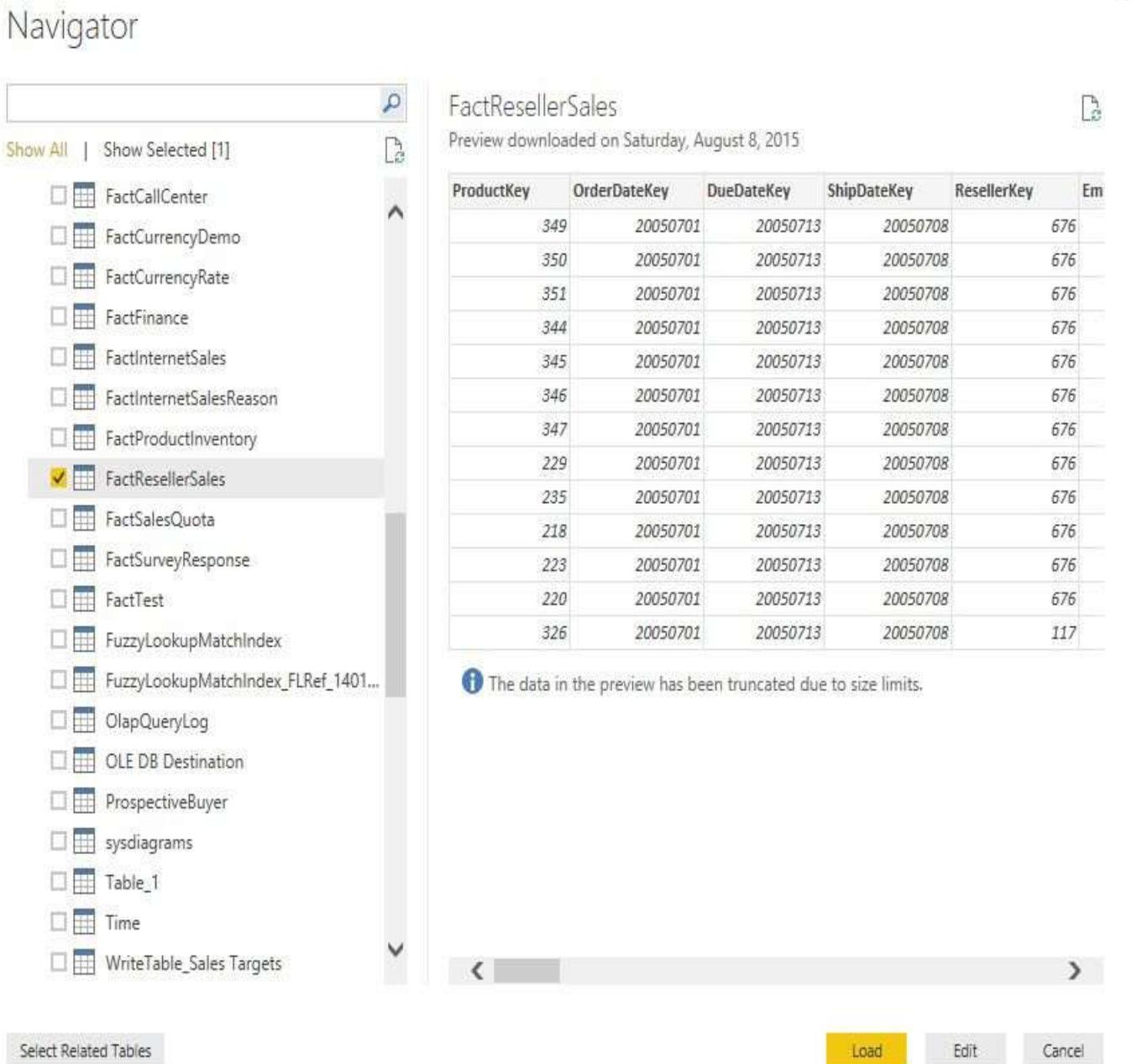


Figure 5.14 The Navigator window allows you to select tables and preview data.

Understanding the Navigator window

If the data source has multiple entities, such as a relational database that has multiple tables or an Excel file with multiple sheets, Power BI Desktop will open the Navigator window (see **Figure 5.14**). Use this window to navigate the data source objects, such as databases and tables, and select one or more tables to import. If the database has many objects, you can search the database metadata to find objects by name. To select a table or a SQL view to import, check its checkbox. The Navigator window shows a preview of the data in the right pane.

If the database has table relationships (the Adventure Works databases have relationships), the “Select Related Tables” button allows you to include related tables so that you can import multiple tables in one step. There are two refresh buttons. The refresh button in the Navigator left pane, refreshes the metadata. This could be useful if someone makes a definition change, such as adding a column, and you don’t want to close the

Navigator window to start over. The refresh button on top of the preview pane refreshes the data preview so you can see the latest data changes or the effect of making column changes to the table whose data you’re previewing, such as removing, adding, deleting, or renaming columns. Again, queries (and everything you do in Power BI Desktop) never make changes to the original data source, so don’t worry about breaking something.

The Edit button launches the Query Editor, and this unlocks a whole new world to shape and transform data, such as if you want to replace values, merge data, unpivot columns, and so on. (I’ll cover the Query Editor in detail in Chapter 6.) The Load button in the Navigator window adds a new table to the data model and loads it with the data from the selected table. If you click the Edit button to open the Query Editor, you can load the data from the Query Editor window once you’re done transforming it.

NOTE Readers familiar with Excel data modelling might know that the Power Pivot Table Import Wizard allows you to filter the data before it’s imported. The Navigator window doesn’t support filtering. However, you can click the Edit button to edit the query. Among many transformation options, you can apply necessary filtering in the Query Editor, such as removing columns and filtering rows for a given date range.

Next, let’s go through a series of exercises to practice importing data from the most common data sources (the ones listed when you drop down the Get Data button), including databases, Excel files, text files, Web, and OData feeds. Because of the growing popularity of R, I’ll also show you how you can import data using R scripts.

5.3.2 Importing from Databases

Nowadays, most corporate data resides in databases, such as SQL Server and Oracle databases. In this exercise, you’ll import the main dataset for analyzing the Adventure Works reseller sales from the FactResellerSales table in the Adventure Works data warehouse database hosted on SQL Server. FactResellerSales represents a fact table and it keeps a historical record of numeric values (facts), such as Sales Amount and Order Quantity. You’ll also import the DimDate table from the same database so that you can aggregate data by date periods, such as month, quarter, year, and so on. As a prerequisite, you need to install the Power BI Desktop (read Chapter 1 for installation considerations), and you need to have the AdventureWorks2012 (or later) database installed locally or on a remote SQL Server machine.

Connecting to the database

Follow these steps to import data from the FactResellerSales table:

1. Open Power BI Desktop. Close the splash screen. Click File → Save (or press Ctrl-S) and then save the empty model as *Adventure Works* in a folder on your local hard drive.
2. Expand the Get Data button in the ribbon and then click SQL Server. This opens the SQL Server Database window.
3. In the Server field, enter the name of the SQL Server instance, such as *ELITE* if SQL Server is installed on a server called ELITE, or *ELITE\2012* if the SQL Server is running on a named instance 2012 on a server called ELITE. Confirm with your database administrator (DBA) what the correct instance name is.

TIP If the AdventureWorksDW database is installed on your local computer, you can enter localhost, (local), or dot (.)

instead of the machine name. However, I recommend that you always enter the machine name. This will avoid connectivity issues that will require you to change the connection string if you decide to deploy the model to Power BI Service and schedule data refresh, or to move the workbook to another computer and then try to refresh the data.

4.If this is the first time you connect to that server, Power BI Desktop opens the “Access a SQL Server Database” window to ask you for your credentials. Assuming that you have access to the server via Windows security, leave the default “Use my current credentials” option selected. Or if the database administrator (DBA) has created a login for you, select the “Use alternate credentials” option, and then enter the login credentials. Click Connect, and then click OK to confirm that you want to use unencrypted connections.

Loading tables

If you connect successfully, Power BI Desktop opens the Navigator window. Let’s load some tables:

- 1.Expand the AdventureWorksDW2012 database. It’s fine if you have a later version of the database, such as AdventureWorksDW2014.
- 2.Scroll down the table list and check the FactResellerSales table. The data preview pane shows the first few rows in the table (see **Figure 5.14** again). Although the Navigator doesn’t show row counts, this table is relatively small (about 60,000 rows), so you can go ahead and click the Load button to import it.
- 3.Because SQL Server is one of the data sources that supports DirectQuery, Power BI Desktop shows the Connection Settings window which asks you if you want to import the data or use DirectQuery. Leave the default Import option selected, and then click OK.

If the source table is large and you don’t want to import all the data, you might want to filter the data before it’s imported (as I’ll show you how in the next step) so that you don’t have to wait for all the data to get loaded and end up with a huge data model.

Filtering data

As a best practice, don’t import data you don’t immediately need for analysis to avoid a large memory footprint and spending an unnecessary long time to load and refresh the data. The Query Editor makes it easy to filter the data before it’s imported. While the Query Editor deserves much more attention (Chapter 6 has the details), let me quickly show you how to filter rows and columns:

- 1.While you are still in the Navigator window, click the Edit button to open Query Editor in a new window.
- 2.In the data preview pane, scroll horizontally to the right until you find the OrderDate column.
- 3.Assuming you want to filter rows by date, expand the dropdown in the OrderDate column header, as shown in **Figure 5.15**. Notice that you can filter by checking or unchecking specific dates. For more advanced filtering options, click the “Date/Time Filters” context menu and notice that you can specify date-related filters, such as After (to filter after a given date) and Between (to filter rows between two dates).
- 4.You can also remove columns that you don’t need. This also helps keep your model more compact, especially with columns that have many unique values because they can’t

compress well. Right-click a column. Note that the context menu includes a Remove option, which you can use to delete a column. Don't worry if you need this column later; you can always bring back removed columns by undoing the Remove Column step.

TIP A more intuitive option for removing and bringing columns back is Choose Columns. (The Choose Columns button is in the Home ribbon of Query Editor.) It allows you to search columns by name which is very useful for wide tables. And you can bring columns back by just checking the column name.

5.If you've decided to open the Query Editor, click the “Close & Apply” button in the Query Editor ribbon to import FactResellerSales.

The screenshot shows the Power BI Query Editor interface. The ribbon has tabs for File, Home, Transform, Add Column, and View. The Home tab is selected. The ribbon bar includes buttons for Close & Apply, New, Recent, Refresh, Properties, Advanced Editor, Choose Columns, Remove Columns, Keep Rows, Remove Rows, Reduce Rows, and Sort. The main area shows a table named "FactResellerSales" with columns: OrderNumber, CustomerPONum..., OrderDate, DueDate, and ShipDate. The "OrderDate" column is currently selected. A filter dialog is open over the table, specifically for the "OrderDate" column. The dialog includes sorting options (Sort Ascending, Sort Descending, Clear Sort), date/time filters (Search, Equals..., Before..., After..., Between..., In the Next..., In the Previous..., Is Earliest, Is Latest, Day, Week, Month), and a note that the list may be incomplete. The table data below the dialog shows several rows of data, such as PO18473189620, 7/1/2005 12:00:00 AM, etc.

Figure 5.15 Use Query Editor to filter rows and columns.

Understanding changes

Irrespective of which path you took (loading the table from the Navigator or from the Query Editor), Power BI Desktop does the following behind the scenes:

1. It creates a query that connects to the database.
2. It adds a FactResellerSales table to the model.
3. It runs the query to extract the data from the FactResellerSales table in the AdventureWorksDW database.

4. It compresses the data and loads it into the FactResellerSales table inside the model.
5. Power BI Desktop switches to the Data View to show you the new table and read-only view of the loaded data (**Figure 5.16**). The Fields pane shows you the table fields.

Don't confuse the Data View, which represents your data model, with the Query Editor, which represents the query used to load a table in the model. While both show a read-only view of the same data, it comes from different places. The Query Editor opens in another window to show you the source data after all transformations you applied but *before* it's loaded to the data model. The Data View shows the data *after* it's loaded into the data model. In other words, the Query Editor shows what will happen to the data after you transform it, while the Data View shows what actually happened after the query was applied and data is loaded.

The screenshot shows the Power BI Desktop interface with the 'Adventure Works - Power BI Desktop' title bar. The ribbon is visible at the top, with 'Data Tools' selected. The 'External Data' group on the ribbon is highlighted with a red box. A callout bubble points to the 'Data tab' in the table preview area. The table preview shows the FactResellerSales table with 60,855 rows. The Fields pane on the right lists the columns of the FactResellerSales table, including ProductKey, OrderDateKey, DueDateKey, ShipDateKey, ResellerKey, EmployeeKey, PromotionKey, CurrencyKey, SalesTerritoryKey, SalesOrderNumber, SalesOrderLineNumber, RevisionNumber, OrderQuantity, UnitPrice, ExtendedAmount, UnitPriceDiscountPct, and DiscountAmount.

ProductKey	OrderDateKey	DueDateKey	ShipDateKey	ResellerKey	EmployeeKey	PromotionKey
317	20050801	20050813	20050808	403	282	
317	20050801	20050813	20050808	403	282	
317	20050801	20050813	20050808	403	282	
314	20050801	20050813	20050808	403	282	
215	20050801	20050813	20050808	403	282	
285	20050801	20050813	20050808	403	282	
272	20051101	20051113	20051108	403	282	
326	20051101	20051113	20051108	403	282	
229	20051101	20051113	20051108	403	282	
336	20051101	20051113	20051108	403	282	
334	20051101	20051113	20051108	403	282	
235	20051101	20051113	20051108	403	282	
313	20051101	20051113	20051108	403	282	
310	20051101	20051113	20051108	403	282	
319	20051101	20051113	20051108	403	282	
328	20051101	20051113	20051108	403	282	
275	20051101	20051113	20051108	403	282	
316	20051101	20051113	20051108	403	282	
262	20051101	20051113	20051108	403	282	
311	20051101	20051113	20051108	403	282	

TABLE: FactResellerSales (60,855 rows) UPDATE AVAILABLE

Figure 5.16 The Data View shows the tables in the model and preview of the data.

The External Data ribbon group is your entry point for data-related tasks. You already

know about the Get Data button. You can use the Recent Sources menu if you need to import another table from the data source that you've already used and if you want to jump directly to the Navigator. The Edit Queries button opens the Query Editor. And the Refresh button reloads all the data so you can get the latest data changes.

NOTE When you work with Power BI Desktop, the only way to synchronize the imported data with the data source changes is to manually refresh the data. You can do so by clicking the Refresh button or by executing the underlying query in the Query Editor. Recall that once you publish the model to Power BI, you have the option to schedule an automatic data refresh.

Importing another table

As I explained, most models would benefit from a date table. Let's import the DimDate table from the same database:

1.In the External Data ribbon group, expand the Recent Sources button. Click the name of the database server that you specified when you connected to SQL Server. This opens the Navigator window.

2.Expand the AdventureWorksDW2012 node and check the DimDate table. Click the Load button. Power BI adds a table with the same name to the model and to the Fields pane.

As I pointed out, you should exclude columns you don't need for analysis. While you can remove columns in the Data View, it's almost always better to remove them in the query so that these columns are excluded from the query that's sent to the data source.

3.Click the Edit Queries button. In the Query Editor, select DimDate in the Queries pane, and then click the Choose Columns button in the Query Editor ribbon's Home tab.

4.Uncheck all columns whose names start with "Spanish" and "French", and then click OK.

5.Click the "Close & Apply" button to apply the query changes and to reload the data. Note that these columns are removed from the DimDate table in the model.

6.Press Ctrl-S to save your data model or click File → Save. Get in a habit to save regularly so you don't lose changes if something unexpected happens and Power BI Desktop shuts down.

5.3.3 Importing Excel Files

Much of corporate data end up in Excel so importing data from Excel files is a common requirement. If you have a choice, ask for an Excel file that has only the data in an Excel list with no formatting. If you have to import an Excel report, things might get more complicated because you'll have to use the Query Editor to strip unnecessary rows and clean the data. In this exercise, I'll show you the simple case for importing an Excel list. In Chapter 6, I'll show you a more complicated case that requires parsing and cleansing an Excel report.

Understanding source data

Suppose that you're given a list of resellers as an Excel file. You want to import this list in the model.

1.Open the Resellers file from \Source\ch05 in Excel (see **Figure 5.17**).

ResellerKey	GeographyKey	ResellerAlternateKey	Phone	BusinessType	ResellerName
1	637	AW00000001	245-555-0173	Value Added Reseller	A Bike Store
2	635	AW00000002	170-555-0127	Specialty Bike Shop	Progressive S...
3	584	AW00000003	279-555-0130	Warehouse	Advanced Bike...
4	572	AW00000004	710-555-0173	Value Added Reseller	Modular Cycle S...
5	322	AW00000005	828-555-0186	Specialty Bike Shop	Metropolitan...
6	303	AW00000006	244-555-0112	Warehouse	Aerobic Exercis...
7	599	AW00000007	192-555-0173	Value Added Reseller	Associated Bike...
8	409	AW00000008	872-555-0171	Specialty Bike Shop	Exemplary C...
9	568	AW00000009	488-555-0130	Warehouse	Tandem Bicycle...
10	44	AW00000010	150-555-0127	Value Added Reseller	Rural Cycle Emp...
11	96	AW00000011	926-555-0159	Specialty Bike Shop	Sharp Bikes
12	96	AW00000012	112-555-0191	Warehouse	Bikes and Motor...

Figure 5.17 The Resellers file represents a list of resellers and it only has the data, without formatting.

2. Notice that the Excel file includes only data on the first sheet. This Excel list includes all the resellers that Adventure Works does business with. Close Excel.

Importing from Excel

Follow these steps to import from an Excel file:

1. With the Adventure Works model open in Power BI Desktop, expand Get Data, and then select Excel.
2. Navigate to the \Source\ch05 folder and double-click the Resellers file.
3. In the Navigator, check Sheet1. The Navigator parses the Excel data and shows a preview of the data in Sheet1 (see **Figure 5.18**).

As you've seen, importing Excel files isn't much different than importing from a database. If the Excel file has multiple sheets with data, you can select and import them in one step. Power BI Desktop will create a query and a corresponding table for each sheet.

TIP Don't like "Sheet1" as a table name in your data model? While you can rename the table in the Data View, you can also rename the query before the data is loaded. Power BI Desktop uses the name of the query as a default table name. While you're still in the Navigator, click the Edit button to open the Query Editor, and then change the query name in the Query Settings pane.

4. Click the Edit button. In the Query Settings pane of the Query Editor, change the query name from Sheet1 to *Resellers*. Click the "Close & Apply" button. Power BI Desktop adds a third table (Resellers) to the data model.

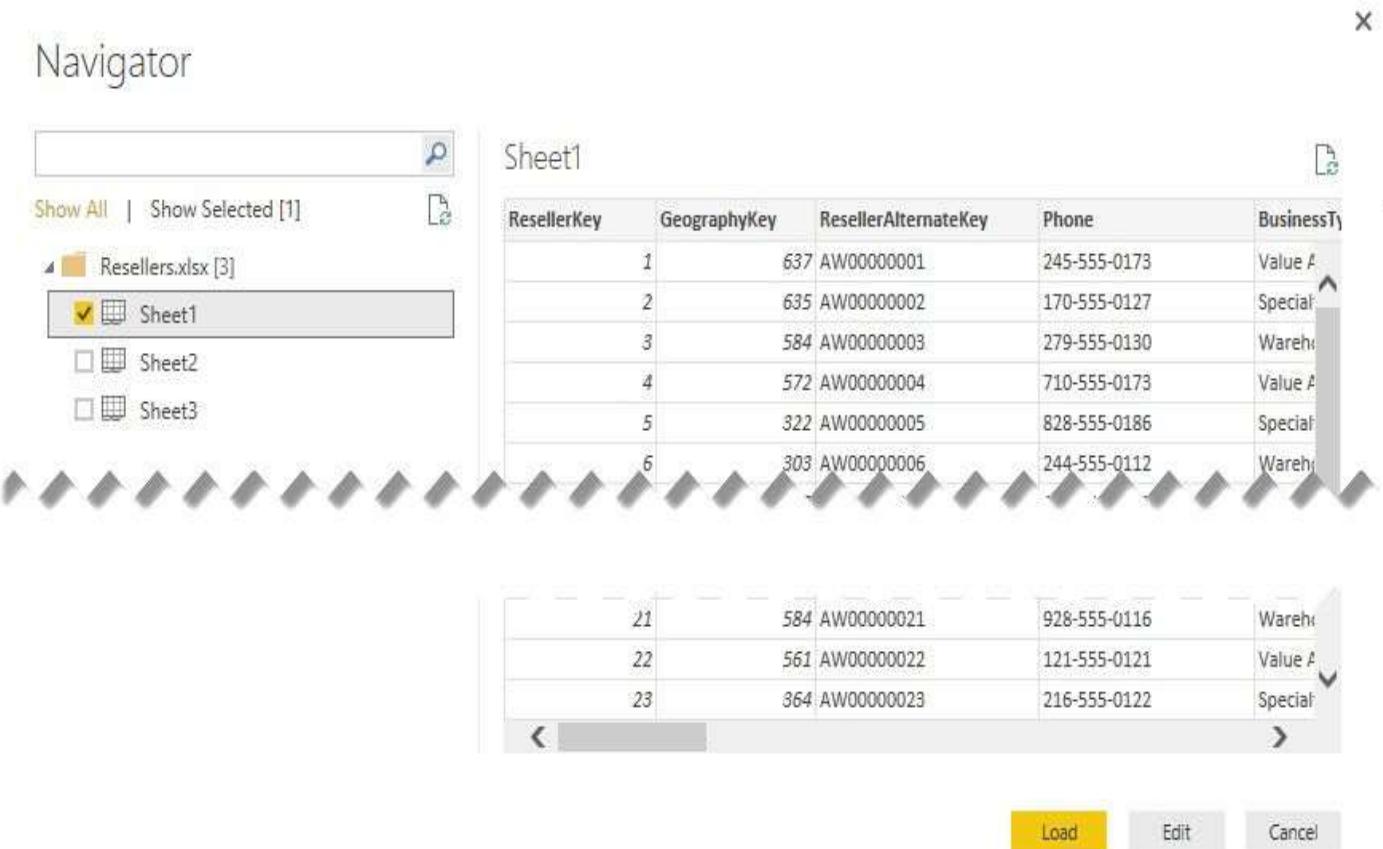


Figure 5.18 The Navigator parses the Excel data and shows a preview.

5.3.4 Importing Text Files

Importing from text files (delimited and fixed-length) is another common requirement. Security and operational requirements might prevent you from connecting directly to a database. In such cases, data could be provided to you as text files. For example, your database administrator might give you a data extract as a file as opposed to granting you direct access to a production database.

Importing from CSV files

Suppose that Adventure Works keeps the employee information in an HR mainframe database. Instead of having direct access to the database, you're given an extract as a comma-separated values (CSV) file. Follow these steps to import this file:

- 1.Expand the Get Data button and click CSV.
- 2.Navigate to the \Source\ch05 folder and double-click the Employees file.
- 3.Because a text file only has a single dataset, Power BI doesn't open the Navigator window. Instead, it just shows you a preview of the data, as shown in **Figure 5.19**. As you can see, it parses the file content and separates it in columns. Click Load to create a new Employees table and to load the data. At this point, the Adventure Works model should have four tables: DimDate, Employees, FactResellerSales, and Resellers.

Employees.txt

EmployeeKey	ParentEmployeeKey	EmployeeNationalIDAlternateKey	ParentEmployeeNationalID
1	18	14417807	NULL
2	7	253022876	NULL
3	14	509647174	NULL
4	3	112457891	NULL
5	3	112457891	NULL
6	267	480168528	NULL
7	112	24756624	NULL
8	112	24756624	NULL
9	23	309738752	NULL
10	189	690627818	NULL

Load **Edit** **Cancel**

Figure 5.19 When importing from files, Power BI Desktop shows a data preview without opening the Navigator pane.

Importing other formats

You might be given a file format other than CSV. For example, the file might use a pipe character (|) as a column separator. Or you might be given a fixed-length file format that I demonstrate with the Employees2.txt file in the \Source\ch05 folder (see **Figure 5.20**).

EmployeeID	FirstName	LastName	Title
14417807	Guy	Gilbert	Production Technician - WC60
253022876	Kevin	Brown	Marketing Assistant
509647174	Roberto	Tamburello	Engineering Manager
112457891	Rob	Walters	Senior Tool Designer

Figure 5.20 The Employees2 file has a fixed-length format where each column starts at a specific position.

You can use the CSV or Text import options to parse other file formats. To make it easier on you, Power BI Desktop will use its smarts to detect the file format. If it's successful, it will detect the delimiters automatically and return a multi-column table. If not, it'll return a single-column table that you can subsequently split into columns using Split Columns and other column tasks in the Query Editor.

NOTE Readers familiar with Power might know that Power Pivot was capable of parsing more complicated file formats using a schema.ini file if it's found in the same folder as the source file. The Power BI Desktop queries doesn't support schema.ini files.

5.3.5 Importing from Analysis Services

If your organization has invested in SQL Server Analysis Services (SSAS), you can import data from SSAS databases or Power Pivot models deployed to SharePoint. This is especially useful when you need results from business calculations or from KPIs defined in a multidimensional cube or a Tabular model. Next, you'll import the Sales Territory

dimension data and a key performance indicator (KPI) from the Adventure Works cube. (The book front matter includes steps for installing the Adventure Works cube.)

IMPORTANT If you don't have an Analysis Services instance with the Adventure Works cube, you can import the DimSalesTerritory CSV file found in the \Source\ch05 folder. Importing DimSalesTerritory won't import the Revenue KPI because it's only defined in the cube. When going through subsequent exercises, ignore steps that reference the Revenue KPI.

Connecting to an SSAS database

Start by connecting to the Adventure Works cube as follows:

1.Expand the Get Data button and click Analysis Services to open the “SQL Server Analysis Services Database” window (see **Figure 5.21**). Remember that Analysis Services supports live connectivity, but it must be the only data source in your Power BI Desktop model. If you combine data from multiple sources, you must import all the data. This is why the “Explore live” option is disabled.

If you don't specify a custom query (or leave the Database field empty) and click OK, the Navigator window will pop up, allowing you to select the dimensions and measures. Once you are done, the Navigator window will auto-generate the MDX query for you. However, unlike the MDX Designer included in Excel, the Navigator doesn't let you specify calculated members (business calculations in cubes). In this case, we need a simple calculation member that returns the key property of the “Sales Territory Region” dimension attribute so we can subsequently relate the SalesTerritory table to Fact-ResellerSales. This is why we need a custom MDX query that includes a SalesTerritoryKey calculated member.

2.In the “SQL Server Analysis Services Database” window, enter the name of your SSAS database, such as *AdventureWorksDW2012Multidimensional-EE*.

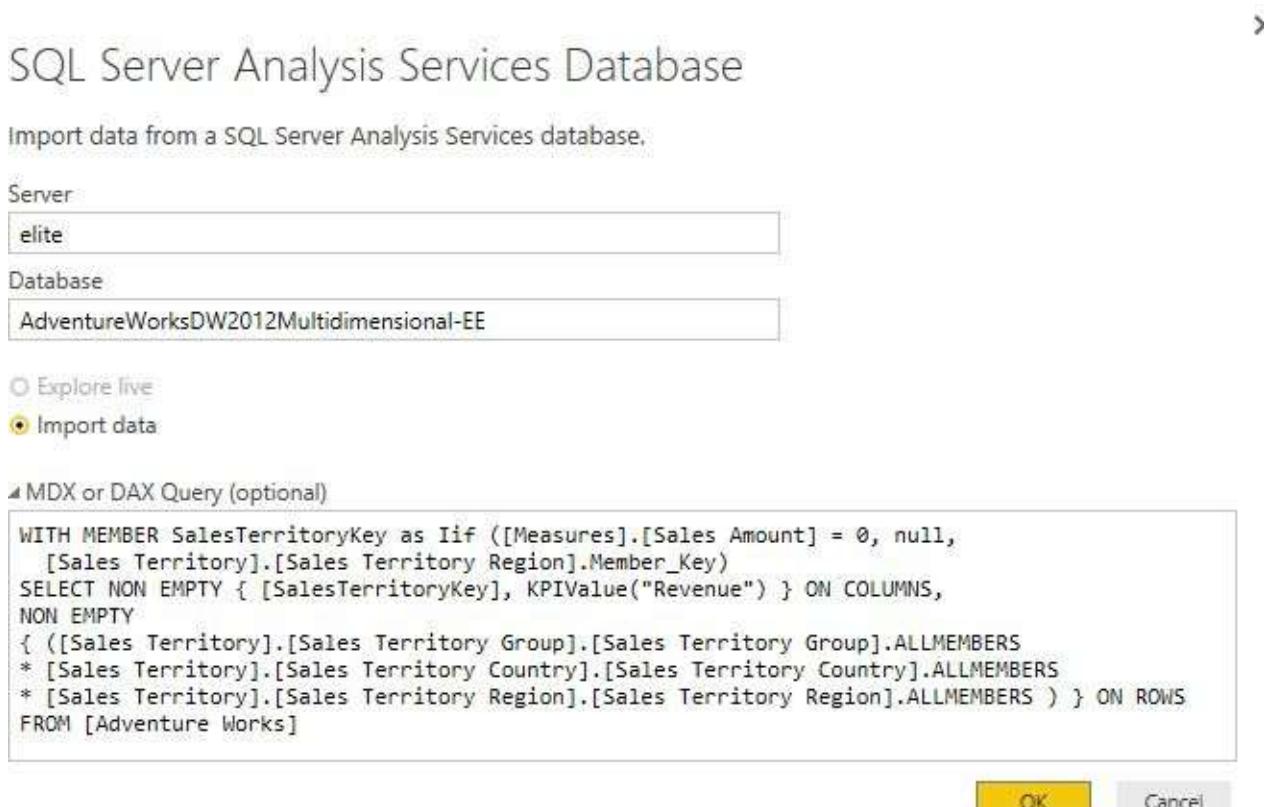


Figure 5.21 Power BI Desktop allows you to import data from SSAS.

3.In the “MDX or DAX Query” field, enter the MDX query, which you can copy from the \Source\ch05\Queries file. Click OK. Power BI Desktop shows a data preview window (see **Figure 5.22**).

The screenshot shows a data preview window in Power BI Desktop. The title bar says "elite: AdventureWorksDW2012Multidimensional-EE". The main area displays a table with two columns: [Sales Territory Region].[Sales Territory] and [Measures].[SalesTerritoryKey]. The [Measures].[SalesAmount] column is visible on the right, though it's not explicitly named in the header row. The data rows show various sales territory keys corresponding to their sales amounts. At the bottom of the window are three buttons: "Load" (yellow), "Edit" (gray), and "Cancel" (gray). A scroll bar is visible on the right side of the table.

[Sales Territory Region].[Sales Territory]	[Measures].[SalesTerritoryKey]	[Measures].[Sales Amount]
	7	7251555.649
	8	4878300.376
	10	7670721.038
	6	16355770.46
	3	7909009.007
	2	6939374.483
	1	16084942.55
	5	7879655.075
	4	24184609.6
	9	10655335.96

Figure 5.22 The SalesTerritoryKey column returns the key value of the Sales Territory Region attribute which you’ll subsequently use as a primary key for creating a relationship.

Renameing metadata

If you click the Load button, you’ll end up with “Query1” as a table name and system-generated column names. You can fix this later, but let’s refine the metadata before the data is loaded.

- 1.Click the Edit button.
- 2.In the Query Settings pane, rename the query from “Query1” to *SalesTerritories*.
- 3.To rename the columns, double-click the column header of each column, and rename the columns to *SalesTerritoryGroup*, *SalesTerritoryCountry*, *SalesTerritoryRegion*, *SalesTerritoryKey*, and *SalesAmount*.
- 4.Click the “Close & Apply” button to create and load a new *SalesTerritories* table.

5.3.6 Importing from the Web

Wealth of information is available on the Web. Power BI Desktop can import tabular data that’s accessible by URL. One popular Web-enabled data source is SQL Server Reporting Services (SSRS). Once a report is deployed to a report server, it’s accessible by URL, which his exactly what the Web import option requires. Next, I’ll show you how to import an SSRS report.

NOTE Readers familiar with the Power Pivot import capabilities might recall that Power Pivot supports importing SSRS reports as data feeds. Unfortunately, as it stands, the Power BI Desktop (and Power Query) OData import option doesn’t support the ATOM data feed format that SSRS generates. However, the report URL can export the report as

CSV, and the output then can be loaded using the Power BI Web import option.

Deploying the report

You can find a sample report named Product Catalog in the \Source\ch05 folder. This report must be deployed to a Reporting Services server that's version 2008 R2 or higher.

NOTE If configuring Reporting Services isn't an option, you can import the required data from the \Source\ch05\DimProduct.txt file or from the AdventureWorksDW database using the custom SQL query I provided in the DimProduct.sql file. The query doesn't return the exact results as the Product Catalog report, and that's okay.

- 1.Upload the Product Catalog.rdl file from the \Source\ch05 folder to your report server, such as to a folder called Power BI in the SSRS catalog. Please note that the report data source uses the AdventureWorks2012 database (see the book front matter for setup instructions), and you probably need to change the connection string in the report data source to reflect your specific setup.
- 2.To test that the report is functional, open the Report Manager by navigating to its URL in your Web browser (assuming SSRS is running in native mode).
- 3.Navigate to the PowerBI folder. Click the Product Catalog report to run the report. The report should run with no errors.

Importing the report

Follow these steps to import the Product Catalog report in the Adventure Works data model:

- 1.In Power BI Desktop, expand the Get Data button and click Web.
- 2.In the “From Web” window that pops up, enter the following URL, but change it to reflect your SSRS server URL (tip: if you test the URL in your Web browser, it should execute fine and it should prompt you to download Product Catalog.csv file):

<http://localhost/ReportServer?/PowerBI/Product Catalog&rs:Command=Render&rs:Format=CSV>

This URL requests the Product Catalog report that's located in the PowerBI folder of my local report server. The Render command is an optimization step that tells SSRS that the requested resource is a report. The Format command instructs the server to export the report in CSV.

- 3.Click OK. Power BI Desktop shows a preview of the report data.
- 4.Click the Edit button. In the Query Editor, rename the query to *Products*.
- 5.Use the Choose Columns feature to remove all the columns whose names start with “Textbox”. Click the “Close & Apply” button to load the Products table.

5.3.7 Importing OData Feeds

Power BI Desktop supports importing data from data feeds that use the OData protocol. Initiated by Microsoft in 2007, Open Data Protocol (OData) is a standard Web protocol for querying and updating data. The protocol allows a client application (consumer), such as Power BI Desktop or Excel, to query a service (provider) over the HTTP protocol and then get the result back in popular data formats, such as Atom Syndication Format (ATOM), JavaScript Object Notation (JSON), or Plain Old XML (POX) formats. Power BI can

integrate with any OData-compliant provider, including Azure Marketplace, SharePoint, and cloud applications. This makes it possible to acquire data from virtually any application and platform where developers have implemented an OData API. For more information about OData, see <http://odata.org>.

To demonstrate importing OData feeds, I'll show you another approach to generate a Date table that uses the DateStream feed available on Microsoft Azure Marketplace. The Microsoft Azure Marketplace is an online market for buying and selling datasets exposed as OData feeds. Some of the datasets, such as DateStream are freely available. Note that this is an optional exercise as it's meant to show you how the OData import works. You won't need the actual data because the data model already has a date table.

Understanding the DateStream feed

Previously, you imported the Date table from the Adventure Works data warehouse. But where do you get a Date table from if you don't have a data warehouse? While a future release of Power BI Desktop might support auto-generating date tables natively, currently this isn't supported. One option is to maintain a date table in Excel and import it from there. Another option is to obtain it from the DateStream feed (<http://datamarket.azure.com/dataset/boyanpenev/datestream>) available at Microsoft Azure Marketplace.

The DateStream feed is documented at <http://datestream.codeplex.com>. Developed by Boyan Penev, it was initially designed to be consumed by Power Pivot. However, because it's implemented as an OData feed, it can be integrated with any OData client. DateStream supports localized date calendars, including US and English calendars. If you need to analyze data by time, it also supports time tables at a minute and second grain.

Importing the DateStream feed

Because Microsoft Azure Marketplace requires authentication, you can't use the OData Feed import option. Instead, you need to import using the Microsoft Azure Marketplace option. As a prerequisite, you need to create a marketplace account so that you can authenticate to Microsoft Azure Marketplace. Once you register with Azure Marketplace, you need to go to <https://datamarket.azure.com/dataset/boyanpenev/datestream> and sign up to use the DateStream service (it's free!) Let's import the DateStream feed:

- 1.In Power BI Desktop, click the Get Data button.
- 2.In the Get Data window, select the Azure tab. Select Microsoft Azure Marketplace and click Connect.
- 3.If this is the first time you connect to Azure Marketplace, you need to authenticate either using a Microsoft account or an account key.

Navigator

The screenshot shows the Power BI Navigator window. On the left, there's a tree view of data feeds. Under the 'DateStream [11]' node, 'BasicCalendarUS' is selected, indicated by a yellow border and checked mark. Other items like 'ExtendedCalendar', 'HourMinute', and 'HourMinuteSecond' are also listed. At the bottom left is an 'Add Data Feeds' button. To the right is a preview pane titled 'BasicCalendarUS' showing a table with columns: DateKey, DateInt, YearKey, QuarterOfYear, MonthOfYear, and Day. The table data starts from 1/1/1900 and goes to 1/23/1900. At the bottom right are 'Load', 'Edit', and 'Cancel' buttons.

DateKey	DateInt	YearKey	QuarterOfYear	MonthOfYear	Day
1/1/1900 12:00:00 AM	19000101	1900	1	1	1
1/2/1900 12:00:00 AM	19000102	1900	1	1	1
1/3/1900 12:00:00 AM	19000103	1900	1	1	1
1/4/1900 12:00:00 AM	19000104	1900	1	1	1
1/5/1900 12:00:00 AM	19000105	1900	1	1	1
1/6/1900 12:00:00 AM	19000106	1900	1	1	1
1/7/1900 12:00:00 AM	19000107	1900	1	1	1
1/8/1900 12:00:00 AM	19000108	1900	1	1	1
1/9/1900 12:00:00 AM	19000109	1900	1	1	1
1/10/1900 12:00:00 AM	19000110	1900	1	1	1
1/11/1900 12:00:00 AM	19000111	1900	1	1	1
1/12/1900 12:00:00 AM	19000112	1900	1	1	1
1/13/1900 12:00:00 AM	19000113	1900	1	1	1
1/14/1900 12:00:00 AM	19000114	1900	1	1	1
1/15/1900 12:00:00 AM	19000115	1900	1	1	1
1/16/1900 12:00:00 AM	19000116	1900	1	1	1
1/17/1900 12:00:00 AM	19000117	1900	1	1	1
1/18/1900 12:00:00 AM	19000118	1900	1	1	1
1/19/1900 12:00:00 AM	19000119	1900	1	1	1
1/20/1900 12:00:00 AM	19000120	1900	1	1	1
1/21/1900 12:00:00 AM	19000121	1900	1	1	1
1/22/1900 12:00:00 AM	19000122	1900	1	1	1
1/23/1900 12:00:00 AM	19000123	1900	1	1	1

Figure 5.23 The DateStream feed includes localized calendars and time tables.

- 4.In the Navigator window, expand the DateStream node, and then check the appropriate calendar, depending on your localization requirements. In **Figure 5.23**, I selected BasicCalendarUS. Note that the data preview shows dates formatted as “MM/dd/yyyy”.

The feed starts with the year 1900 and goes all the way to 2100. If needed, you can use the Query Editor to filter a smaller range. Just click the Edit button and then apply column filtering to filter the date feed, such as on the YearKey column. And if you need additional columns, such as for fiscal years and quarters, add custom columns in the Query Editor, or add DAX calculated columns in the Data View.

- 5.Because you won’t need this data for the purposes of the Adventure Works data model, click Cancel.

This completes the import process of the initial set of tables that you’ll need in order to analyze the Adventure Works sales. At this point, the Fields list should have six tables:

FactResellerSales, DimDate, SalesTerritories, Resellers, Employees, and Products.

5.3.8 Importing from R

With all the buzz surrounding R, data analysts might want to preserve their investment in R scripts and reuse them to import, transform, and analyze data with Power BI.

Fortunately, Power BI supports R as a data source (currently this feature is in preview).

The R integration brings the following benefits:

- Visualize your R data – Once your R script’s data is imported in Power BI, you can use a Power BI visualization to present the data. That’s possible because the R data source is not different than any other data source.
- Share the results – You can leverage the Power BI sharing and collaboration capabilities to disseminate the results computed in R with everyone in your organization.
- Operationalize your R script – Once the data is uploaded to Power BI Service, you can configure the dataset for a scheduled refresh, so that the reports are always up to date.
- Reuse your R visualizations – You might use the R ggplot2 package to plot beautiful statistical graphs. Now you can bring these graphs in Power BI. This opens up a whole world of new data visualizations.

In this exercise, I’ll show you how to use R to forecast time series and visualize it (see **Figure 5.24**). The first segment in the line chart shows the actual sales, while the second segment shows the forecasted sales that are calculated in R. Because we won’t need the forecasted data in the Adventure Works data model, you’ll find the finished example in a separate “R Demo.pbix” file located in the \Source\ch05 folder.

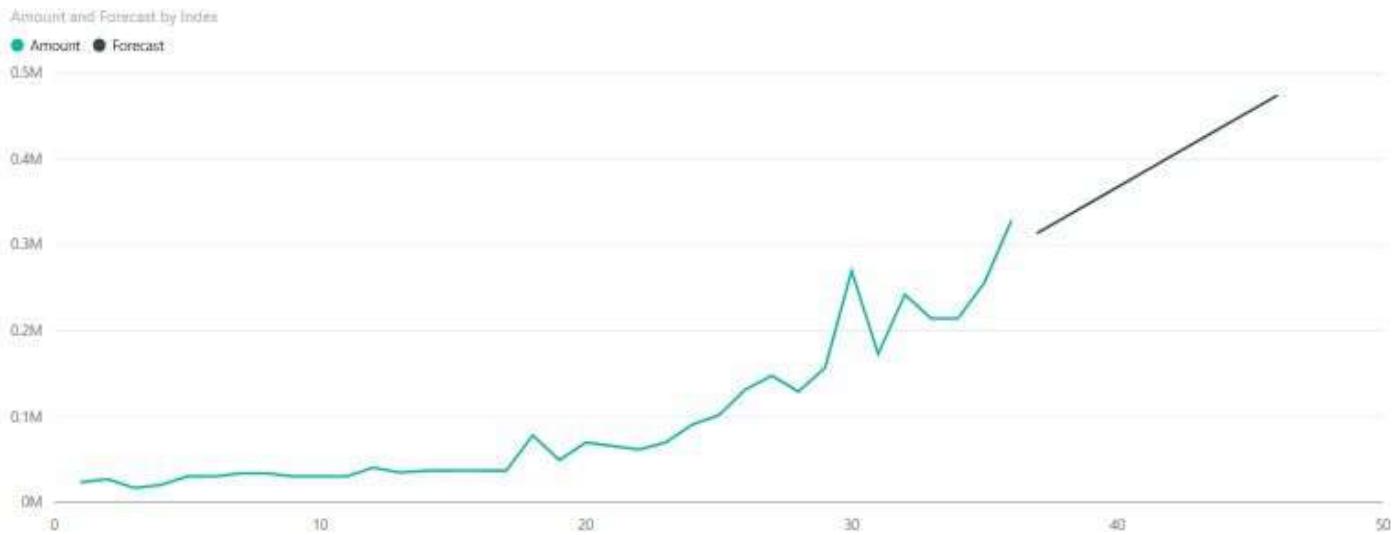


Figure 5.24 This visualization shows actual and forecasted sales.

Getting started with R

R is an open source programming language for statistical computing and data analysis. Over the years the community has contributed and extended the R capabilities through packages that provide various specialized analytical techniques and utilities. Besides supporting R in Power BI, Microsoft invested in R by acquiring Revolution Analytics,

whose flagship product (Revolution R) will be integrated with SQL Server 2016.

Before you can use the Power BI Desktop R integration, you must install R. R can be downloaded from various location, including the Revolution Open download page or CRAN repository.

1. Open your web browser and navigate to one of the R distribution location, such as <https://mran.revolutionanalytics.com/download>, and then download and install R for Windows.

2. I also recommend that you install RStudio (an open source R development environment) from <http://www.rstudio.com>. RStudio will allow you to prepare and test your R script before you import it in Power BI Desktop.

3. Use the Windows ODBC Data Sources (64-bit) tool (or 32-bit if you use the 32-bit version of Power BI Desktop) to set up a new ODBC system data source AdventureWorksDW that points to the AdventureWorksDW2012 (or a later version) database.

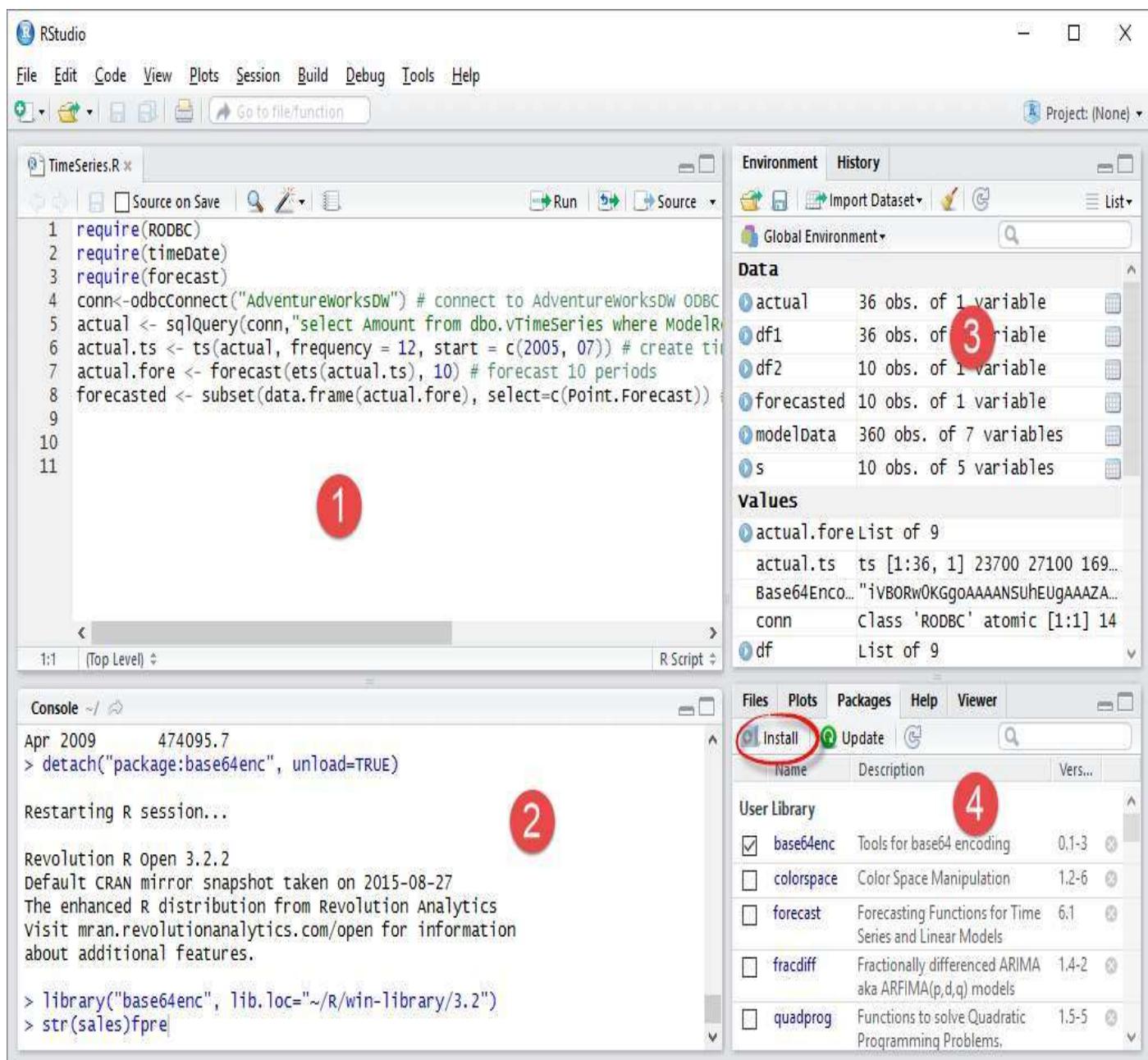


Figure 5.25 Use RStudio to develop and test R scripts.

Using R for time series forecasting

Next, you'll create a basic R script for time series forecasting using RStudio. The RStudio user interface has four areas (see **Figure 5.25**). The first area (shown as 1 in the screenshot) contains the script that you're working on. The second area is the RStudio Console and it allows you to test the script. For example, if you position the mouse cursor on a given script line and press Ctrl-Enter, RStudio will execute the current script line and it will show the output in the console.

The Global Environment area (shown as 3 in **Figure 5.25**) shows some helpful information about your script variables, such as the number of observations in a time series object. Area 4 has a tabbed interface that shows some additional information about the RStudio environment. For example, the Packages tab shows you what packages are loaded, while the Plots tab allows you to see the output when you use the R plotting capabilities. Let's start by importing the packages that our script needs:

- 1.Click File → New File → R Script File (or press Ctrl+Shift+N) to create a new R Script. Or, if you don't want to type the R code, click File → Open File, then open my TimeSeries.R script from the \Source\ch05 folder.
- 2.In the area 4, select the Packages tab, and then click Install.
- 3.In the Install Packages window, enter *RODBC* (IntelliSense helps you enter the correct name), and then click Install. This installs the RODBC package which allows you to connect to ODBC data sources.
- 4.Repeat the last two steps to install the “*timeDate*” and “*forecast*” packages.

Going through the code, lines 1-3 list the required packages. Line 4 connects to the AdventureWorksDW ODBC data source. Line 5 retrieves the Amount field from the vTimeSeries SQL view which is one of the sample views included in the AdventureWorksDW database. The resulting dataset represents the actual sales that are saved in the “actual” data frame. Similar to a Power BI dataset, a R data frame stores data tables. Line 6 creates a time series object with a frequency of 12 because the actual sales are stored by month. Line 7 uses the R forecast package to create forecasted sales for 10 periods. Line 8 stores the Point.Forecast column from the forecasted dataset in a data frame.

NOTE As of the time of writing, the Power BI R Source only imports data frames, so make sure the data you want to load from a R script is stored in a data frame. Going down the list of limitations, columns that are typed as Complex and Vector are not imported, and are replaced with error values in the created table. Values that are N/A are translated to NULL values in Power BI Desktop. Also, any R script that runs longer than 30 minutes will time out. Interactive calls in the R script, such as waiting for user input, halts the script's execution.

Using the R Script source

Once the R script is tested, you can import the results in Power BI Desktop.

- 1.Open Power BI Desktop. Click Get Data → More → R Script, and then click Connect.
- 2.In the “Execute R Script” window (see **Figure 5.26**), paste the R script. Expand the “R Installation Settings” section and make sure that the R installation location matches your

R setup. Click OK.

Execute R Script

Execute an R script on a local R installation and import the resulting data frames.

```
require(RODBC)
require(timeDate)
require(forecast)
conn<-odbcConnect("AdventureWorksDW") # connect to AdventureWorksDW ODBC
actual <- sqlQuery(conn,"select Amount from dbo.vTimeSeries where ModelRegion = 'M200 Europe'")
actual.ts <- ts(actual, frequency = 12, start = c(2005, 07)) # create time series
actual.fore <- forecast(ets(actual.ts), 10) # forecast 10 periods
forecasted <- subset(data.frame(actual.fore), select=c(Point.Forecast)) # flip the forecasted res
```

◀ R Installation Settings

Use R installed in the following location:

C:\Program Files\RRO\R-3.2.2

[How to install R](#)

OK

Cancel

Figure 5.26 Enter the script in “Execute R Script” window and check your R installation settings.

- 3.In the Navigator window, notice that the script imports two tables (actual and forecasted) that correspond to the two data frames you defined in the R script. Click the Edit button to open Query Editor.
- 4.Click the “actuals” table.
- 5.With the “actuals” query selected in the Queries pane, click the Append Queries button in ribbon’s Home tab.
- 6.In the Append window, select the “forecasted” table, and then click OK. This appends the forecasted table to the actual table so that all the data (actual and forecasted) is in a single table.
- 7.Rename the “actual” table to *ActualAndForecast*.
- 8.Rename the Point.Forecast column to *Forecast*.
- 9.(Optional) If you need actual and forecasted values in a single column, in the ribbon’s Add Column tab, click “Add Custom Column”. Name the custom column “Result” and enter the following expression:

```
if [Amount]=null then [Forecast] else [Amount]
```

This expression adds a new Result column that combines Amount and Forecast values into a single column.

- 10.In the ribbon’s Add Column tab, click “Add Index Column” to add an auto-incremented column that starts with 1.

- 11.In the Home ribbon, click Close & Apply to execute the script and import the data.
- 12.To visualize the data, create a Line Chart visualization that has the Index field added to the Axis area, and Amount and Forecast fields added to the Values area.
- 13.(Optional) Deploy the Power BI Desktop file to Power BI Service and schedule the dataset for a scheduled refresh.

5.4 Summary

A Power BI model is a relational-like model and represents data as tables and columns. This chapter started by laying out fundamental data modeling concepts (such as table schemas, relationships, and keys) that you need to understand before you import data. It also explained data connectivity options supported by Power BI Desktop and introduced you to the Power BI premium tool for self-service data modeling.

Next, the chapter explained the data import capabilities of Power BI Desktop. As you've seen, you can acquire data from a wide variety of data sources, including relational and multidimensional databases, Excel files, text files, Web, and data feeds. If you use R to import, transform, and analyze data, you can preserve your investment and import your R scripts. Once the data is imported in the model, every dataset is an equal citizen and it doesn't matter where the data came from!

Source data is seldom clean. Next, you'll learn how to use queries to shape and transform raw data when needed.

Chapter 6

Transforming Data

As you've seen, it doesn't take much effort to import data from wherever it might reside. Importing data is one thing, but transforming arbitrary or dirty data is quite another. Fortunately, Power BI Desktop has a query layer that allows you to clean and transform data before it's loaded in the model. Remember that this layer is available when you import data or when you connect live to data sources using the DirectQuery connectivity mechanism. The query layer isn't available when you connect live to SSAS.

This chapter explores the capabilities of the Query Editor component of Power BI Desktop. It starts by introducing you to the Query Editor design environment. Next, it walks you through an exercise to practice its basic transformation steps. It also teaches you about its more advanced transformation features that require custom code. The query examples for this chapter are located in the `Query Examples.pbix` file in the `\Source\ch06` folder.

6.1 Understanding the Query Editor

The Query Editor is packed with features that let you share and transform data before it enters the data model. The term “transformation” here includes any modification you apply on the raw data. All transformations are repeatable, meaning that if you have to import another data extract that has the same structure, Power BI Desktop will apply the same transformation steps.

NOTE You might have heard of BI pros implementing Extraction, Transformation, and Loading (ETL) processes to clean data in an automated way. The Query Editor (or Excel Power Query) is to self-service BI what ETL is to organizational BI. Although not as powerful as professional ETL tools, the Query Editor should be able to help when issues with source data require attention.

6.1.1 Understanding the Query Editor Environment

Before I dive into the Query Editor’s plethora of features, let’s take a moment to explore its environment. As I mentioned, you launch the Query Editor when you click the Edit Queries button in the Power BI Desktop ribbon’s Home tab. The Query Editor opens in a new window, side by side with the Power BI Desktop main window. **Figure 6.1** shows the main elements of Query Editor when you open it in the Adventure Works model that you implemented in the previous chapter.

Understanding the ribbon’s Home tab

The Home tab in the ribbon (see item 1 in **Figure 6.1**) includes buttons for common tasks and some frequently used columns and table-level transformations. Starting from the left, you’re already familiar with the Close & Apply button. When expanded, this button has three values, giving you options to close the Query Editor without applying the query changes to the data model (Close menu), to apply the changes without closing the editor (Apply), and both (Close & Apply). If you choose to close the editor without applying the changes, Power BI Desktop will display a warning that pending query changes aren’t applied.

NOTE Some structural changes, such as adding a new column, will reload the data in the corresponding table in the data model. Other changes, such as renaming columns, are handled internally without data refresh. Power BI Desktop (more accurately the xVelocity engine) always tries to apply the minimum steps for a consistent model without unnecessary data refreshes.

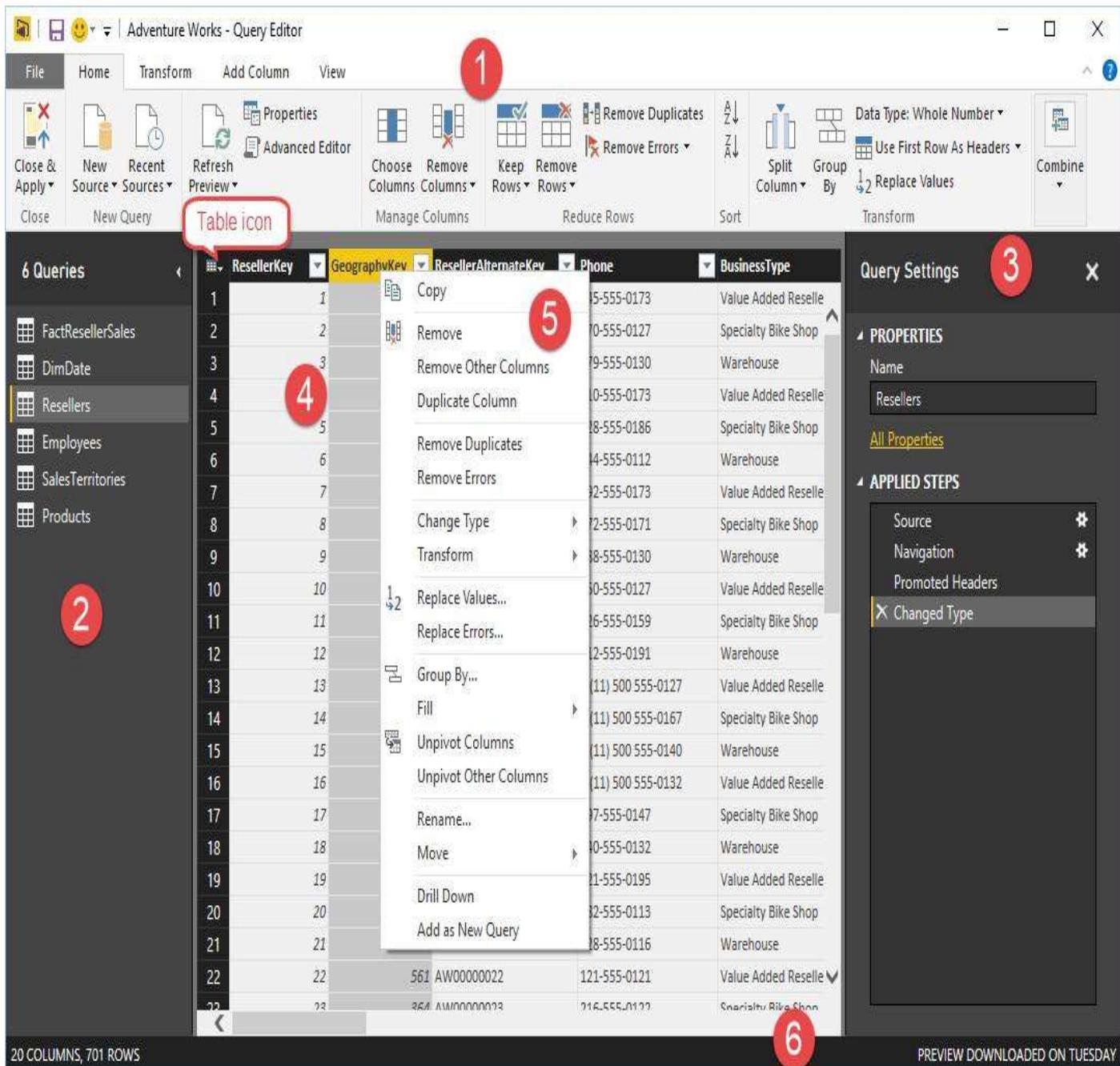


Figure 6.1 The Query Editor opens in a new window to give you access to the queries defined in the model.

The New Query ribbon group is another starting point for creating new queries if you prefer to do so while you’re in the Query Editor as opposed to Power BI Desktop. The Properties button opens a Query Properties window (**Figure 6.2**) that allows you change the query name (another way to change the query name is to use the Query Settings pane).

Sometimes, you might not want to load the query data in the data model, such as when you plan to append the query results to another query. If you don’t want the query to generate a table in the data model, uncheck the “Unable load to report” checkbox. And, if you don’t want to refresh the query results when the user initiates Power BI Desktop table refresh, uncheck “Enable refresh of this query”. Continuing on the list of Home tab’s buttons, the Advanced Editor button gives you access to the query source.

NOTE Queries are described in a formula language (informally known as “M”). Every time you apply a new transformation, the Query Editor creates a formula and adds a line to the query source. For more information about the

query formula language, read “Microsoft Power Query for Excel Formula Language Specification” at <http://go.microsoft.com/fwlink/p/?linkid=320633>.

Figure 6.2 Use the Query Properties pane to change the query name, to enable data load to report, and to enable data refresh.

The rest of the buttons on the Home tab let you perform common transformations, including removing columns, reducing rows, grouping by and replacing values, and combining queries. We’ll practice many of these in the lab exercise that follows.

Understanding the ribbon’s Transform tab

The Transform tab (see **Figure 6.3**) includes additional table and column transformations. Many of the column-level transformations from the context menu (see item 5 in **Figure 6.1**) are available when you right-click a column in the data preview pane. And, many of the table-level transformations are available when you expand or right-click the Table icon (grid) in the top-left corner of the data preview pane.

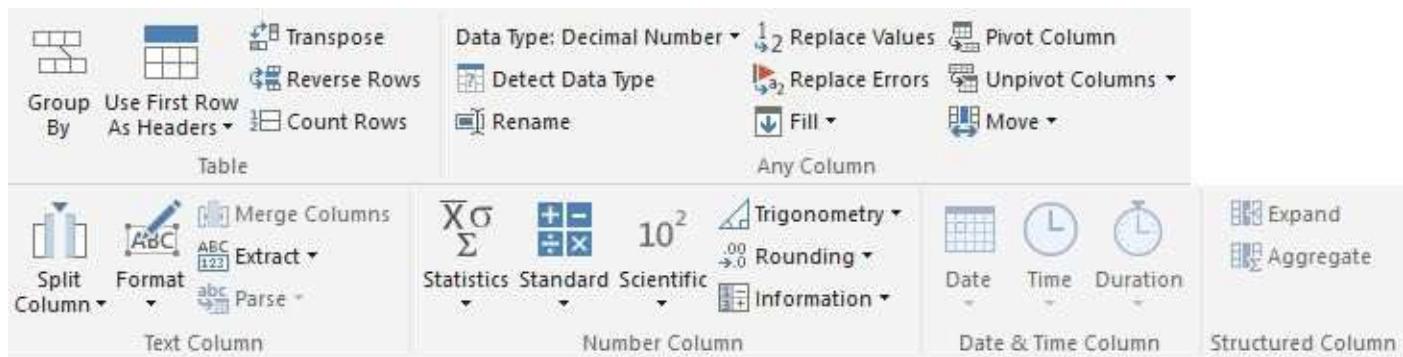


Figure 6.3 The Transform ribbon (split in the screenshot to reduce space) includes many table and column transformations.

Some transformations apply to columns that have specific data types (see the second row in **Figure 6.3**). For example, the Split Column transformation applies only to text columns, while the Rounding transformation applies only to number columns.

Understanding the ribbon’s Add Column tab

The Add Column tab (see **Figure 6.4**) lets you create custom columns. For example, I’ll show you later how you can create a custom column that returns the end of the month from a given date column.



Figure 6.4 Use the Add Column tab in the ribbon to create custom columns.

NOTE Don’t confuse query custom columns with data model calculated columns. Added to the query, query custom columns are created using the Power Query formula language called “M” and they can’t reference fields in the data model. On the other hand, calculated columns in the data model are described in DAX and they can reference other fields in the model.

Understanding the ribbon’s View tab

The Query Settings button in the View tab toggles the visibility of the Query Settings pane

(item 3 in **Figure 6.2**). The Advanced Editor button does the same thing as the button with the same name (also called Advanced Editor) in the Home tab. It shows the source code of the query and allows you to change it. The Formula Bar checkbox toggles the visibility of the formula bar that shows you the “M” formula behind the selected transformation step.

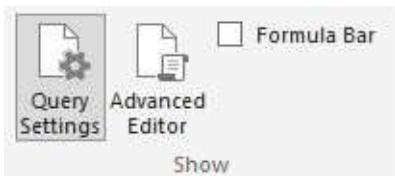


Figure 6.5 The View tab gives you access to the query source.

So what does this “M” query language do for you anyway? It allows you to implement more advanced data manipulation. For example, Martin needs to load multiple Excel files from a given folder. Looping through files isn’t an out-of-box feature. However, once Martin applies the necessary transformations to a single file, he can use the Advanced Editor to modify the query source to define a query function. Now Martin can automate the process by invoking the query function for each file, passing the file path. I’ll show you some of these capabilities later in this chapter.

6.1.2 Understanding Queries

I mentioned in the previous chapter that there’s a query behind every table you import or access live via DirectQuery. The whole purpose of the Query Editor is to give you access to these queries so that you can add additional transformation steps if needed.

Understanding the Queries pane

The Queries pane (see item 5 in **Figure 6.1**) shows you all the queries that exist in the Power BI Desktop file. In the Adventure Works model, there are six queries. In general, the number of queries correspond to the number of tables you use in the model, unless you’ve created queries for other more advanced tasks, such as to merge a query with results from other queries.

You can right-click a query to open a context menu with additional tasks, such as to delete, duplicate, reference (reuse a base query so you can apply additional steps), enable load (same as “enable load to report” in the Query Properties window), move the query up or down, and organize queries in groups.

Understanding the Query Settings pane

As you’ve seen, the Query Settings pane allows you to change the query name. Renaming the query changes the name of the table in the data model and vice versa. The more significant role of Query Settings is to list all the steps you’ve applied to load and shape the data (see **Figure 6.6**).

The query shown in the screenshot is named Products and it has four transformation steps. The Source step represents the connection to the data source. You can click the cog icon (⚙️) next to it to view and change the source settings, such as the name of the file or server. If the Get Data flow used the Navigator window, such as to let you select a database table or an Excel sheet, the second step would be Navigation so that you can

view or change the source table if needed. However, you imported the Products table from an SSRS report, and you didn't use the Navigator window.

Although you didn't specifically do this, Power BI Desktop applied the Promoted Headers step to promote the first row as column headers. Power BI Desktop applies the Change Type step when it discovers that it needs to overwrite the column data types. Finally, you applied the “Removed Other Columns” step when you removed some of the source columns when you imported the Product Catalog report.

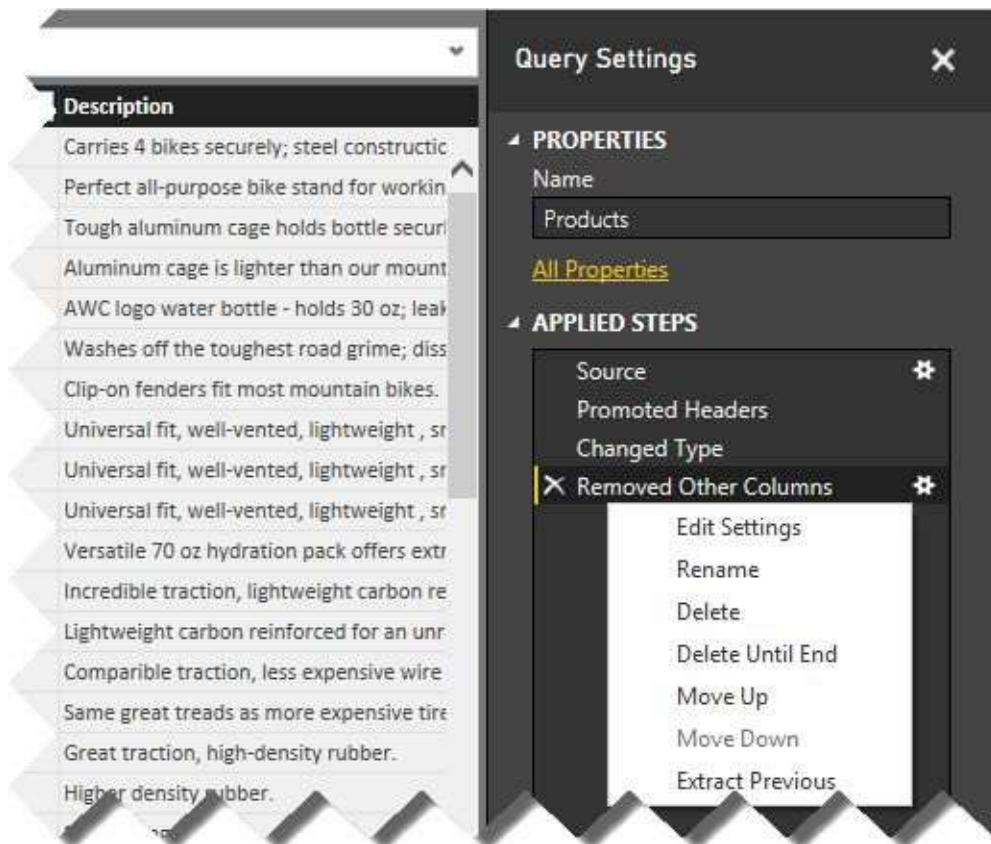


Figure 6.6 The Applied Steps section of Query Setting show all the steps applied to load and shape the data.

You can click a step to select it. If the formula bar is enabled (you can check the Formula checkbox in the ribbon's View tab to enable it), you'll see the query language formula behind the step. In addition, selecting a step updates the data preview to show you how the step affected the source data. When you select a step in the Applied Steps list, the Data Preview pane shows the transformed data after that step is applied. So you can always go back in the step history to check the effect of every step!

If the step is not needed, you can click the (x) button to the left of the step name or press the Delete key to remove a step. The Query Editor will ask you to confirm deleting intermediate steps because there might be dependent downstream steps and removing a prior step might result in breaking changes. You can also right-click a step to get additional options, such as to rename a step to make its name more descriptive, to delete it, to delete the current steps and all subsequent steps, to move the step up or down in the Applied Steps list, and to extract the previous steps in a separate query.

6.1.3 Understanding Data Preview

The data preview pane (see item 4 back in **Figure 6.1**) shows a read-only view of the source schema and the data as of the time the query was created or the data preview was last refreshed. Each column has a drop-down menu that lets you sort and filter the data before it's imported.

Understanding data filtering

Filtering allows you to exclude rows so you don't end up with more data than you need. The filtering options differ, based on the data type of the column. **Figure 6.7** shows the filtering options available for date/time columns. It's important to understand that unlike sorting in the Data View in Power BI Desktop, sorting and filtering in the Query Editor affects how the data is loaded in the data model. For example, if I filter the ShipDate column in FactResellerSales for the last year, this will create a new transformation step that will load only the data for last year based on the system date.

NOTE Power BI queries have smarts to push as much processing as possible to the data source. This is called query folding. For example, if you filter a table column, the query would append a WHERE clause that the query will pass on to the data source. This is much more efficient than filtering the results after all the data is loaded. Filters, joins, groupings, type conversions, and column removal are examples of work that gets folded to the source. What gets folded depends on the capabilities of the source, internal logic, and the data source privacy level.

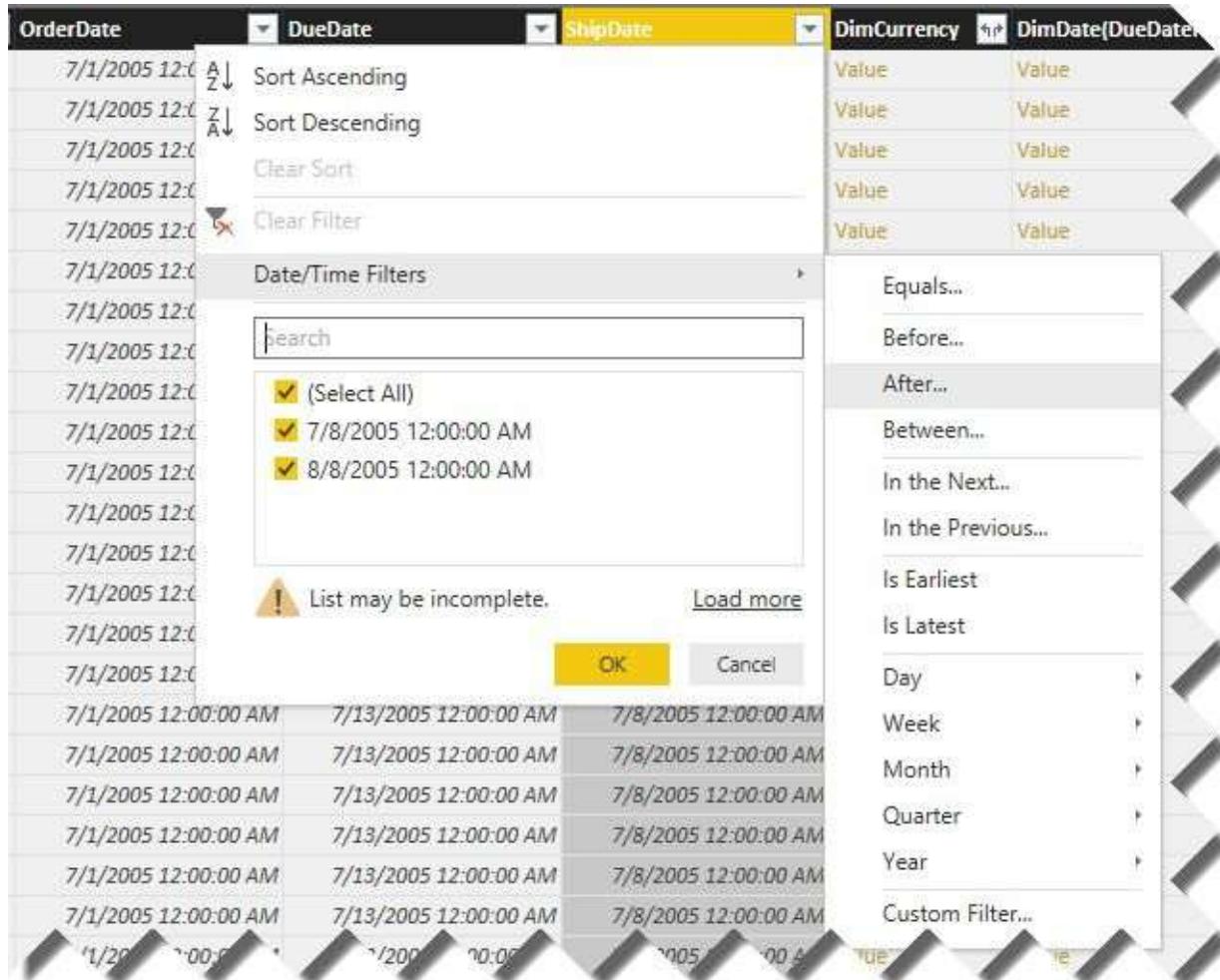


Figure 6.7 The column dropdown allows you to sort and filter the source data before it's loaded in the model

Understanding preview caching

The Query Editor status bar (see item 6 in the figure) informs you when the data preview of the selected query was last refreshed. A cached copy of the query preview results is stored on your local hard disk for faster viewing later. You can control the size of the data cache from File → Options and Settings → Options (Data Load tab). Because the preview results are cached, the data preview might get out of sync with schema and data changes in the data source. You can click the Refresh Preview button to update the data preview.

If the data preview hasn't been refreshed for more than two days, a warning will be displayed above the preview pane. Don't confuse data preview refresh with table refresh in the data model (the Refresh button in the Power BI Desktop ribbon). The data model refresh executes the queries and reloads the data.

TIP Do you want to export the data shown in the preview pane? While waiting for Microsoft to implement the "Export to Excel" feature, you can expand the Table icon in the top-left corner of the preview pane and click "Copy Entire Table". Then you can paste the data in Excel.

Auto-discovering relationships

When you import data from a database that has relationships defined between tables, the query discovers these relationships and adds corresponding columns to let you bring columns from the related tables. For example, if you select FactResellerSales and scroll the data preview pane all the way to the right, you'll see "split" columns for DimCurrency, DimDate (three columns for each relationship), DimEmployee, and all the other tables that FactResellerSales has relationships to in the AdventureWorksDW database. If you click the split button () in the column header, Query Editor opens a list that allows you to add columns from these tables. This handy feature saves you steps to create relationships or custom queries.

NOTE The relationships in the database and in the Query Editor are different than the relationships among the tables in the model (discussed in detail in the next chapter). However, when you import tables, Power BI Desktop checks the query for database relationships during the auto-discovery process and it might add corresponding relationships in the data model.

As you've seen, you can also rename columns in the Query Editor and in the Data View interchangeably. No matter which view you use to rename the column, the new name is automatically applied to the other view. However, I encourage you to check and fix the column data types (use the Transform group in the ribbon's Home tab) in the Query Editor so you can address data type issues before your data is loaded.

For example, you might expect a sales amount field to be a numeric column. However, when Query Editor parsed the column, it changed its data type to Text because of some invalid entries, such as "N/A" or "NULL". It would be much easier to fix these data type errors in the Query Editor, such as by using the Remove Errors, Replace Errors, and Replace Values column-level transformations.

6.2 Shaping and Cleansing Data

Suppose that the Adventure Works Finance department gives Martin periodically (let's say every month or year) an Excel file that details accessories and parts that Adventure Works purchases from its vendors. Martin needs to analyze spending by vendor. The problem is that the source data is formatted in Excel tables that the Finance department prepared for its own analysis. This makes it very difficult for Martin to load the data and relate it to other tables that he might have in the model. Fortunately, the Query Editor component of Power BI Desktop allows Martin to transform and extract the data he needs. For the purposes of this exercise, you'll use the Vendor Parts.xlsx file that's located in the \Source\ch06 folder.

6.2.1 Applying Basic Transformations

Figure 6.8 shows the first two report sections in the Vendor Parts file. This format is not suitable for analysis and requires some preprocessing before data can be analyzed.

Specifically, the data is divided in sections and each section is designed as an Excel table. However, you need just the data as a single Excel table, similar to the Resellers Excel file that you imported in the previous chapter. Another issue is that the data is presented as crosstab reports, making it impossible to join the vendor data to a Date table in the data model.

Vendor Parts - 2008

Category	Manufacturer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	2014 Total
Wheels	Acme 1	341	442	703	699	772	697	555	518	539	521	434	521	6,742
	Acme 2	230	334	617	819	955	829	596	603	411	272	251	340	6,257
	Acme 3	407	665	1,307	1,511	1,608	1,317	1,039	902	680	587	471	593	11,087
	Acme 4	202	293	545	520	504	419	356	364	310	246	184	239	4,182
	Acme 5	467	612	1,320	1,333	1,453	1,329	942	972	806	559	577	464	10,834
	Acme 6	4	6	10	7	13	11	11	9	4	11	5	4	95
	Acme 7	284	293	410	495	410	353	280	254	194	209	238	318	3,738
	Acme 8	18	27	74	64	67	75	51	45	33	33	23	20	530
Mix Total		1,953	2,672	4,986	5,448	5,782	5,030	3,830	3,667	2,977	2,438	2,183	2,499	43,465

Category	Manufacturer	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	2014 Total
Tires	Acme 1	13	26	31	42	18	10	20	25	18	21	9	33	266
	Acme 2	7,074	8,256	12,808	14,375	14,495	11,358	8,724	9,236	7,588	7,418	9,032	21,971	132,335
	Acme 3	18	22	79	414	3,257	2,800	2,75	73	63	701	580	352	3,522

Figure 6.8 The Vendor Parts Excel file includes crosstab reports, which present a challenge for relating this data to other tables in the model.

Exploring source data

Let's follow familiar steps to connect to the Excel file. However, this time you'll launch the Query Editor before you import the data.

- 1.If the Vendor Parts file is open in Excel, close it so that Excel doesn't lock the file and prevent importing.
- 2.Open a new instance of Power BI Desktop. Save the file as *Query Examples*.
- 3.Expand the Get Data menu, and click Excel because you'll import an Excel file.
- 4.Navigate to the \Source\Ch06 folder and select the Vendor Parts.xlsx file. Then click Open.
- 5.In the Navigator window, check Sheet1 to select it and preview its data. The preview shows how the data would be imported if you don't apply any transformations. As you see, there are many issues with the data, including mixed column content, pivoted data by month, and null values.
- 6.Click the Edit button to open the Query Editor.

Removing rows

First, let's remove the unnecessary rows:

1. Right-click the first cell in the Column1 column, and then click Text Filters ð “Does Not Equal” to exclude the first row.
2. Locate the *null* value in the first cell of Column3, and apply the same filter (Text Filters ð “Does Not Equal”) to exclude all the rows that have NULL in Column3.
3. Promote the first row as headers so that each column has a descriptive column name. To do so, in the ribbon's Transform tab, click the “Use First Row as Headers” button. Alternatively, you can expand the table icon (in the top-left corner of the preview window), and then click “User First Row as Headers”. Compare your results with **Figure 6.9**.

The screenshot shows the Microsoft Query Editor interface. On the left, a sidebar lists 7 Queries: FactResellerSales, DimDate, Resellers, Employees, SalesTerritories, Products, and Sheet1, with Sheet1 selected. The main area displays a table with columns: Category, Manufacturer, Jan, Feb, Mar, Apr, May, Jun. The data includes rows for Wheels (Manufacturer Acme 1), Tires (Manufacturer Acme 1, 2, 3), and a Mix Total row. The Query Settings pane on the right shows the Name is set to Sheet1 and the Applied Steps list includes Source, Navigation, Changed Type, Filtered Rows, Filtered Rows1, and Promoted Headers.

	Category	Manufacturer	Jan	Feb	Mar	Apr	May	Jun
1	Wheels	Acme 1	341	442	703	699	772	
2		null Acme 2	230	334	617	819	955	
3		null Acme 3	407	665	1307	1511	1608	
4		null Acme 4	202	293	545	520	504	
5		null Acme 5	467	612	1320	1333	1453	
6		null Acme 6	4	6	10	7	13	
7		null Acme 7	284	293	410	495	410	
8		null Acme 8	18	27	74	64	67	
9	Mix Total		1953	2672	4986	5448	5782	
10	Category	Manufacturer	Jan	Feb	Mar	Apr	May	Jun
11	Tires	Acme 1	13	26	31	42	18	
12		null Acme 2	7074	8256	12808	14375	14495	
13		null Acme 3	1854	2473	3479	3414	3257	
14		null Acme 4	2	2	11	11	11	

Figure 6.9 The source data after filtering unwanted rows.

4. Note that the first column (Category) has many null values. These empty values will present an issue when relating the table to other tables, such as Product. Click the first cell in the Category column (that says Wheels), and then click the ribbon's Transform tab. Click the Fill Δ Down button. This fills the null values with the actual categories.
5. Let's remove rows that represent report subtotals. To do so, right-click a cell in the Category column that contains the word Category. Click Text Filters Δ "Does Not Equal" to remove all the rows that have "Category" in the first column.
6. Next, you will need to filter all the rows that contain the word "Total". Expand the column dropdown in the column header of the Category column. Click Text Filters Δ "Does Not Contain". In the Filter Rows dialog box, type "Total", and then click OK.
7. Hold the Ctrl key and select the last two columns, Column15 and 2014 Total. Right-click the selection, and then click Remove Columns.

Un-pivoting columns

Now that you've cleansed most of the data, there's one remaining task. Note how the months appear on columns. This makes it impossible to join the table to a Date table because you can't join on multiple columns. To make things worse, as new periods are added, the number of columns might increase. To solve this problem, you need to un-pivot the months from columns to rows. Fortunately, this is very easy to do with the Query Editor!

1. Hold the Shift key and select all the month columns, from Jan to Dec.

2.Right-click the selection, and then click Unpivot Columns. The Query Editor un-pivots the data by creating new columns called Attribute and Value, as shown in **Figure 6.10**.

8. Double-click the column header of the Attribute column, and rename it to *Month*.
Rename the Value column to *Units*.

The screenshot shows the Power BI Query Editor interface. On the left, there's a list of 7 Queries: FactResellerSales, DimDate, Resellers, Employees, SalesTerritories, Products, and Sheet1 (which is selected). The main area displays a table with four columns: Category, Manufacturer, Attribute, and Value. The rows show data for 'Wheels' from January to November, with values ranging from 341 to 772. To the right, the 'Query Settings' pane is open, showing the 'Name' is set to 'Sheet1'. Under 'APPLIED STEPS', 'Source' is listed with a gear icon, followed by 'Navigation', 'Changed Type', and 'Filtered Rows'.

	Category	Manufacturer	Attribute	Value
1	Wheels	Acme 1	Jan	341
2	Wheels	Acme 1	Feb	442
3	Wheels	Acme 1	Mar	703
4	Wheels	Acme 1	Apr	699
5	Wheels	Acme 1	May	772
6	Wheels	Acme 1	Jun	697
7	Wheels	Acme 1	Jul	555
8	Wheels	Acme 1	Aug	518
9	Wheels	Acme 1	Sep	539
10	Wheels	Acme 1	Oct	521
11	Wheels	Acme 1	Nov	

Figure 6.10 The un-pivoted dataset includes Attribute and Value columns.

Adding custom columns

If you have a Date table in the model, you might want to join this data to it. As I mentioned, a Data table is at a day granularity. So that you can join to it, you need to convert the Month column to a date. Click the ribbon’s Add Column tab, and then click “Add Custom Column”.

1.In the “Add Custom Column” dialog box, enter *FirstDayOfMonth* as the column name. Then enter the following formula (see **Figure 6.11**):

=Date.FromText([Month] & " 1," & "2008")

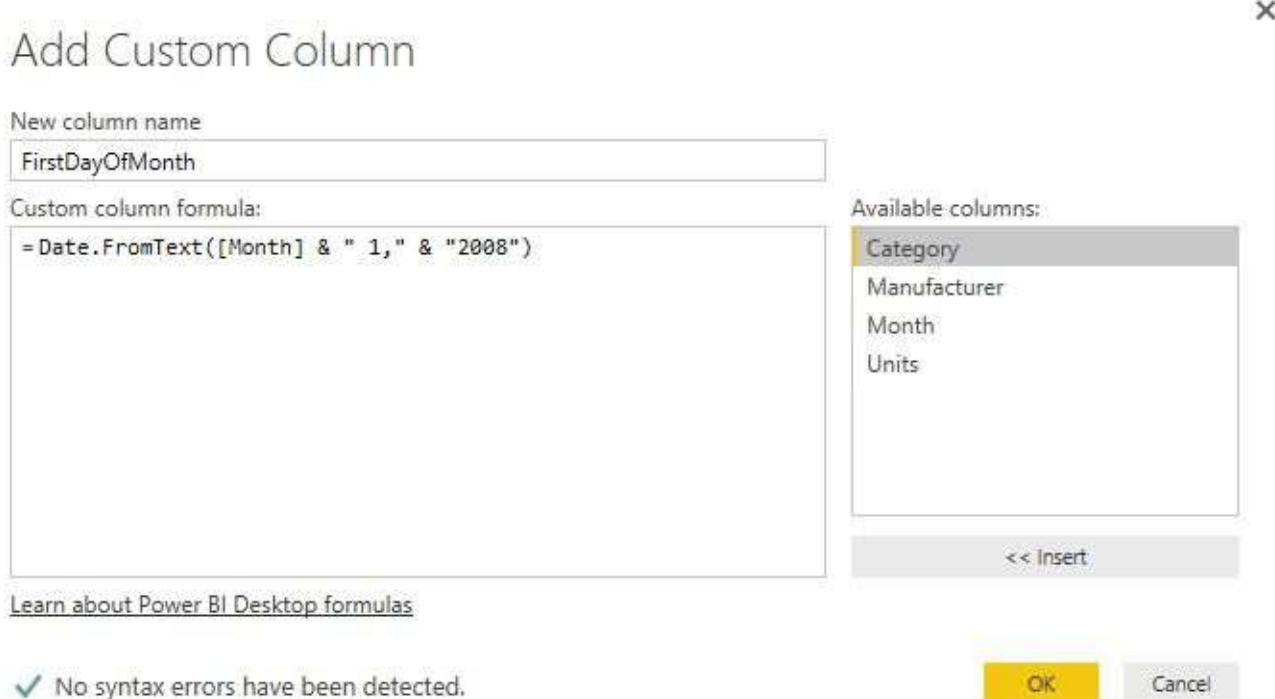


Figure 6.11 Create a custom column that converts a month to a date.

This formula converts the month value to the first day of the month in year 2008. For example, if the Month value is Jan, the resulting value will be 1/1/2008. The formula hardcodes the year to 2008 because the source data doesn't have the actual year. If you need to default to the current year, you can use Date.Year(DateTime.LocalNow()) formula.

NOTE Unfortunately, as it stands the “Add Custom Column” window doesn't have IntelliSense or show you the formula syntax, making it difficult to work with formulas and forcing a “trial and error” approach. If you've made a mistake, the custom column will display “Error” in every row. You can click the Error link to get more information about the error. Then in the Applied Steps pane, click the Settings next to the step to get back to the formula, and try again.

2.Assuming you need the month end date instead of the first date of the month, select the FirstDayOfMonth column. In the ribbon's Transform tab, expand the Date dropdown button, and then select Month ð “End of Month”.

3.Rename the new column to *Date*.

6.2.2 Loading Transformed Data

Now that you've cleaned and transformed the raw data, you can load the query results to the model. As I explained, you have three options to do so that are available by expanding the Close & Apply button. Close & Apply closes the Query Editor and applies the changes to the model. You can close the Query Editor without applying the changes, but the model and queries aren't synchronized. Finally, you can choose to apply the changes without closing the Query Editor so that you can continue working with it.

Renaming steps and queries

Before loading the data, consider renaming the query to apply the same name to the new table. You can also rename transformation steps to make them more descriptive. Let's rename the query and a step:

- 1.In the Query Settings pane, rename the query to *VendorParts*. This will become the name of the table in the model.
- 2.In the Applied Steps pane, right-click the last step and click Rename. Change the step name to “Renamed Column to Date”, and click Enter.

Loading transformed data

Let's load the transformed data into a new table:

- 1.Click the Close & Apply button in the ribbon. Power BI Desktop imports the data, closes the Query Editor, and adds the VendorParts table to the Fields pane.
- 2.In the ribbon's Home tab, click the Edit Queries button. This brings you to the Query Editor in case you want to apply additional transformation steps.
- 3.(Optional) You can disable loading query results. This could be useful if another query uses the results from the VendorParts query and it doesn't make sense to create unnecessary tables. To demonstrate this, right-click the VendorParts query in the Queries pane and then uncheck “Enable Load”. Accept the warning that follow that disabling the query load will delete the table from the model and break existing reports. Click Close & Apply and notice that the VendorParts table is removed from the Fields list.

6.3 Using Advanced Features

The Query Editor has much to offer than just basic column transformations. In this section, I'll walk you through more advanced scenarios that you might encounter so that you handle them yourself instead of asking for help. First, you'll see how you can join and merge datasets. Then I'll show you how query functions can help you automate mundane data processing tasks. Finally, you'll find how to use the "M" query language to auto-generate date tables.

6.3.1 Combining Datasets

As I mentioned previously, if relationships exist in the database, the query will discover these relationships. This allows you to expand a table and reference columns from other tables. For example, if you open the Adventure Works model you created in the previous chapter and examine the data preview pane of the FactResellerSales query, you'll see columns that correspond to all the tables that are related to the FactResellerSales table in the AdventureWorksDW database. All of these column show the text "Value" for each row in FactResellerSales and have an expansion button () in the column header (see **Figure 6.12**).

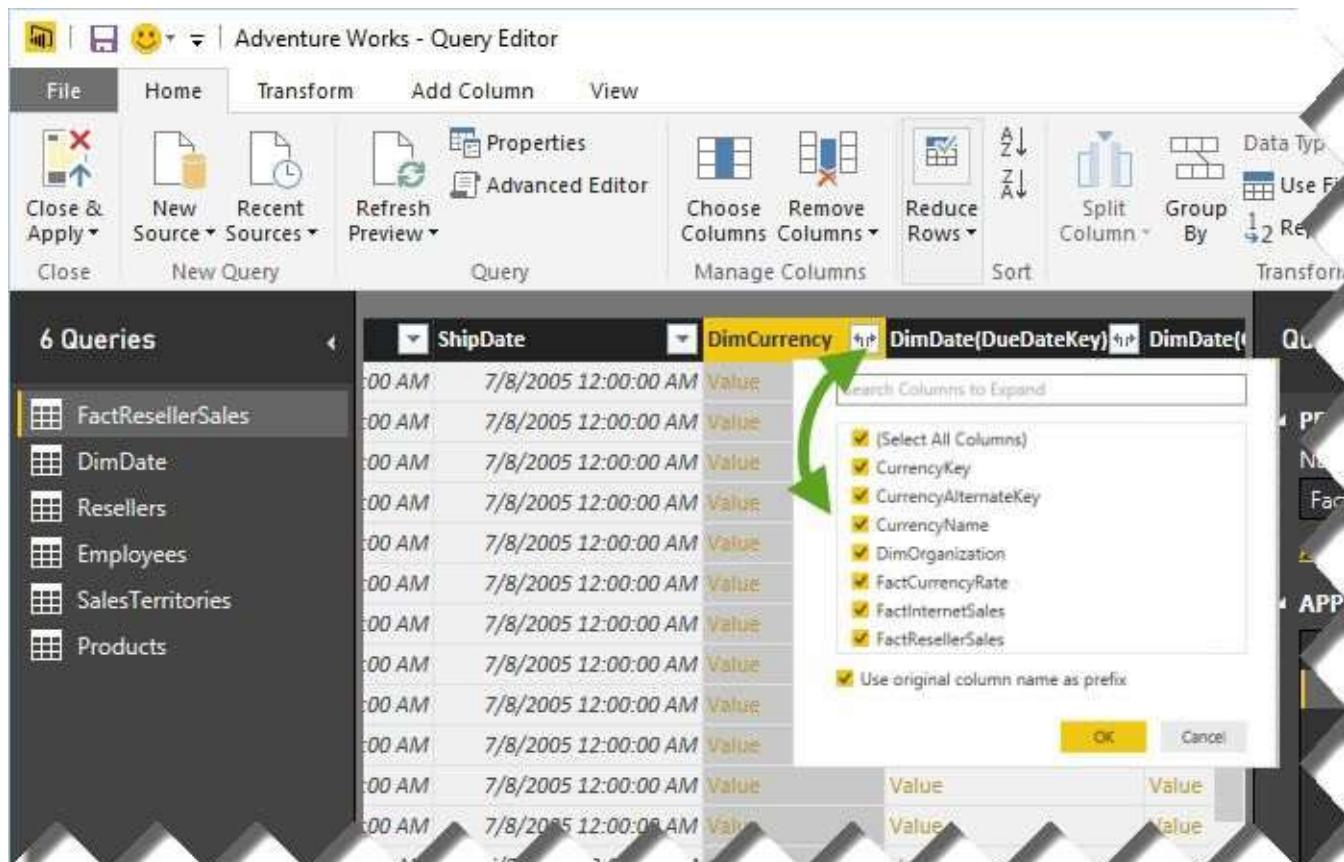
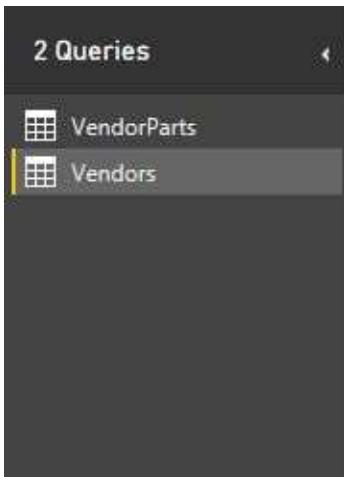


Figure 6.12 Create a custom column that converts a month to a date.

When you click the expansion button, the Query Editor opens a window that lists all the columns from the related table so that you can include the columns you want in the query. This is a useful feature, but what if there are no relationships in the data source? As it turns out, as long as you have matching columns, you can merge (join) queries.

We are back to the VendorParts.pbix model. Suppose that you have another query that returns a list of vendors that Adventure Works does business with. In the Vendor Parts model, you'll find a Vendors query that I created by importing the Vendors sheet from the Excel Vendor Parts file (see **Figure 6.13**).



2 Queries			
	Name	City	State
1	Acme 1	Atlanta	GA
2	Acme 2	Norcross	GA
3	Acme 3	Alpharetta	GA
4	Acme 4	New York	NY
5	Acme 5	Charlotte	NC
6	Acme 6	Nashville	TN
7	Acme 7	Tampa	FL
8	Acme 8	Montgomery	AL
9	Acme 9	Birmingham	AL
10	Acme 10	Boston	MA

Figure 6.13 The Vendors query returns a list of vendors imported from the Vendors sheet in the Vendor Parts Excel file.

Now I'd like to join the VendorParts query to the Vendors query so that I can add some columns from the Vendor query in the VendorParts table. If two queries have a matching column(s) you can join them just like you can join two tables in SQL:

- 1.In the Queries pane, select the VendorParts query because this will be our base query.
- 2.In the ribbon's Home tab, click Merge Queries (in the Combine group).
- 3.Configure the Merge window as shown in **Figure 6.14**. This setup joins the Manufacturer column from the VendorParts query to the Name column of the Vendors query.

Merge

Select a table and matching columns to create a merged table.

VendorParts



Category	Manufacturer	Month	Units	FirstDayOfMonth	Date
Wheels	Acme 1	Jan	341	1/1/2008	1/31/2008
Wheels	Acme 1	Feb	442	2/1/2008	2/29/2008
Wheels	Acme 1	Mar	703	3/1/2008	3/31/2008
Wheels	Acme 1	Apr	699	4/1/2008	4/30/2008
Wheels	Acme 1	May	772	5/1/2008	5/31/2008

Vendors



Name	City	State
Acme 1	Atlanta	GA
Acme 2	Norcross	GA
Acme 3	Alpharetta	GA
Acme 4	New York	NY
Acme 5	Charlotte	NC

Join Kind

Left Outer (all from first, matching from second)

Left Outer (all from first, matching from second)

Right Outer (all from second, matching from first)

Full Outer (all rows from both)

Inner (only matching rows)

Left Anti (rows only in first)

Right Anti (rows only in second)

OK

Cancel

Figure 6.14 You can merge queries by one or more matching columns.

Notice that the Join Kind list that has different types of joins. For example, a Left Outer Join will keep all the rows from the first query and only return the matching values from the second query, or null if no match is found. By contrast, the Inner Join will only return matching rows.

- 4.Click OK. The Query Editor adds a new column to the end of the VendorParts query.
- 5.Expand the new column. Notice that you can add columns from the Vendors query (**Figure 6.15**). You can also aggregate these columns, such as by using the Sum or Count aggregation functions.

Category	Manufacturer	Month	Units	FirstDayOfMonth	Date	NewColumn
1 Wheels	Acme 1	Jan				
2 Wheels	Acme 1	Feb				
3 Wheels	Acme 1	Mar				
4 Wheels	Acme 1	Apr				
5 Wheels	Acme 1	May				
6 Wheels	Acme 1	Jun				
7 Wheels	Acme 1	Jul				
8 Wheels	Acme 1	Aug				
9 Wheels	Acme 1	Sep				
10 Wheels	Acme 1	Oct				
11 Wheels	Acme 1	Nov				
12 Wheels	Arme 1	Dec				

Figure 6.15 Once you merge queries, you can add columns to the source query from the merged query.

Appending queries

Suppose that some time has passed and Martin gets another Vendors Parts report, for the year 2009. Instead of overwriting the data in the data model, which will happen if Martin refreshes the data, Martin wants to append the second dataset to the VendorParts table so he can analyze data across several years. If Martin is given a new file once a year, he can use the “Append Queries” feature to append queries manually. To simulate a second dataset, you’ll clone the existing VendorParts query.

- 1.In the Queries pane, right-click the VendorParts query, and then click Copy.
- 2.Right-click an empty area in the Queries pane, and then click Paste. The Query Editor adds two queries: VendorParts (2) and Vendors (2). It clones the Vendors query because it discovers that VendorParts depends on it.
- 3.In the Queries pane, select the VendorParts (2) query. In the Query Settings pane, delete the last step, “Merged Queries”.
- 4.Open the settings of the “Added Custom” step and change the formula to use year 2009. You do this to simulate that this dataset has data for the year 2009.

Date.FromText([Month] & " 1," & "2009")

- 5.In the Query Settings pane, rename the VendorsParts (2) query to *VendorParts 2009*.
- 6.In the Queries pane, delete the Vendors (2) query.
- 7.Right-click *VendorParts 2009* and turn off Enable Loading because you’ll append this query and you don’t want it to create a new table when you click Close & Apply.
- 8.In the Queries pane, select the VendorParts query.
- 9.In the ribbon’s Home tab, click Append Queries (in the Combine group). In the Append window, select the VendorParts 2009 query, and then click OK.

The Query Editor appends VendorParts 2009 to VendorParts. As a result, the VendorParts query returns a combined dataset for years 2008 and 2009. As long as two queries have the same columns, you can append them.

6.3.2 Using Functions

Appending datasets manually can get tedious quickly as the number of files increase. What if Martin is given Excel reports every month and he needs to combine 100 files? Well, when the going gets tough, you write some automation code, right? If you have a programming background, your first thought might be to write code that loops through files, to check if the file is an Excel file, to load it, and so on. However, the “M” language that Power BI queries are written in is a functional language, and it doesn’t support loops and conditional logic. What it does support is functions and they can be incredibly useful for automation.

Creating query functions

Think of a query function as a query that can be called repeatedly from another query. Similar to functions in programming languages, such as C# and Visual Basic, a function can take parameters to make it generic. Let’s create a query function that will load an Excel file given a folder path and file name.

- 1.The easiest way to create a function is to start with an existing query that already produces the results you need. In the Queries pane, copy the VendorParts query and paste it.
- 2.In the Queries pane, right-click the VendorsParts (2) query, and then click Advanced Editor. Another way to open the Advanced Editor is to click Advanced Editor in the ribbon’s Home tab or View tab. The Advanced Editor shows the M code behind the query. Each line represents a transformation step you applied.
- 3.Insert a new line at the beginning of the query and type in the following M code (see **Figure 6.16**):

0 =>

The empty parenthesis signifies that the function has no parameters. And the “goes-to” => operator precedes the function code.

- 4.Click Done. In the Applied Steps list of the Query Settings pane, you should see a single step called *Vendor Parts (2)*. In the Queries pane, you should see *fx* in front of *Vendor Parts (2)*. The Preview pane has an Invoke button.
- 5.Click the Invoke button. This executes function.
- 6.In the Applied Steps pane, delete the “Invoked FunctionVendorParts (2)” step created by the Invoke action because you won’t need it.

The screenshot shows the Power BI Advanced Editor window. The title bar says "Advanced Editor". The main area contains the following M query code:

```

() =>
let
    Source = Excel.Workbook(File.Contents("D:\Users\Teo\Books\POWERBI\Source\ch06\Vendor Parts.xlsx"), null, true),
    Report_Sheet = Source{[Item="Sheet1", Kind="Sheet"]}[Data],
    #"Changed Type1" = Table.TransformColumnTypes(Report_Sheet,{{"Column1", type text}, {"Column2", type any}, {"Column3", type any}}),
    #"Changed Type" = Table.TransformColumnTypes(#"Changed Type1",{{"Column1", type text}, {"Column2", type any}, {"Column3", type any}}),
    #"Filtered Rows" = Table.SelectRows(#"Changed Type", each [Column1] <> "Vendor Parts - 2008"),
    #"Filtered Rows1" = Table.SelectRows(#"Filtered Rows", each [Column3] <> null),
    #"Promoted Headers" = Table.PromoteHeaders(#"Filtered Rows1"),
    #"Filled Down" = Table.FillDown(#"Promoted Headers",{"Category"}),
    #"Filtered Rows2" = Table.SelectRows(#"Filled Down", each [Category] <> "Category"),
    #"Filtered Rows3" = Table.SelectRows(#"Filtered Rows2", each not Text.Contains([Category], "Total")),
    #"Removed Columns" = Table.RemoveColumns(#"Filtered Rows3",{"Column15", "2014 Total"}),
    #"Unpivoted Columns" = Table.UnpivotOtherColumns(#"Removed Columns", {"Category", "Manufacturer"}, "Attribute", "Value"),
    #"Renamed Columns" = Table.RenameColumns(#"Unpivoted Columns",{{"Attribute", "Month"}, {"Value", "Units"}}),
    #"Added Custom" = Table.AddColumn(#"Renamed Columns", "FirstDayOfMonth ", each Date.FromText([Month] & " 1," & "2008")),
    #"Inserted End of Month" = Table.AddColumn(#"Added Custom", "EndOfMonth", each Date.EndOfMonth([#"FirstDayOfMonth "]), type date),
    #"Renamed Column to Date" = Table.RenameColumns(#"Inserted End of Month",{{"EndOfMonth", "Date"}})
in
    #"Renamed Column to Date"

```

Below the code, a message says "No syntax errors have been detected." At the bottom right are "Done" and "Cancel" buttons.

Figure 6.16 Use the Advanced Editor to work with the query code and create a function.

7.Rename the query to *fnProcessFiles*. A function can be called anything but it's good to have a consistent naming convention.

Working with function parameters

The next step will be to make the function generic.

1. In the Queries pane, right-click the *fnProcessFiles* query, and then click Advanced Editor. Notice that the Query Editor has changed the function code somewhat when you invoked the function.

1.Change the first let statement as follows (see **Figure 6.17**) to define two parameters:

let

fnProcessFiles = (FolderPath, FileName) =>

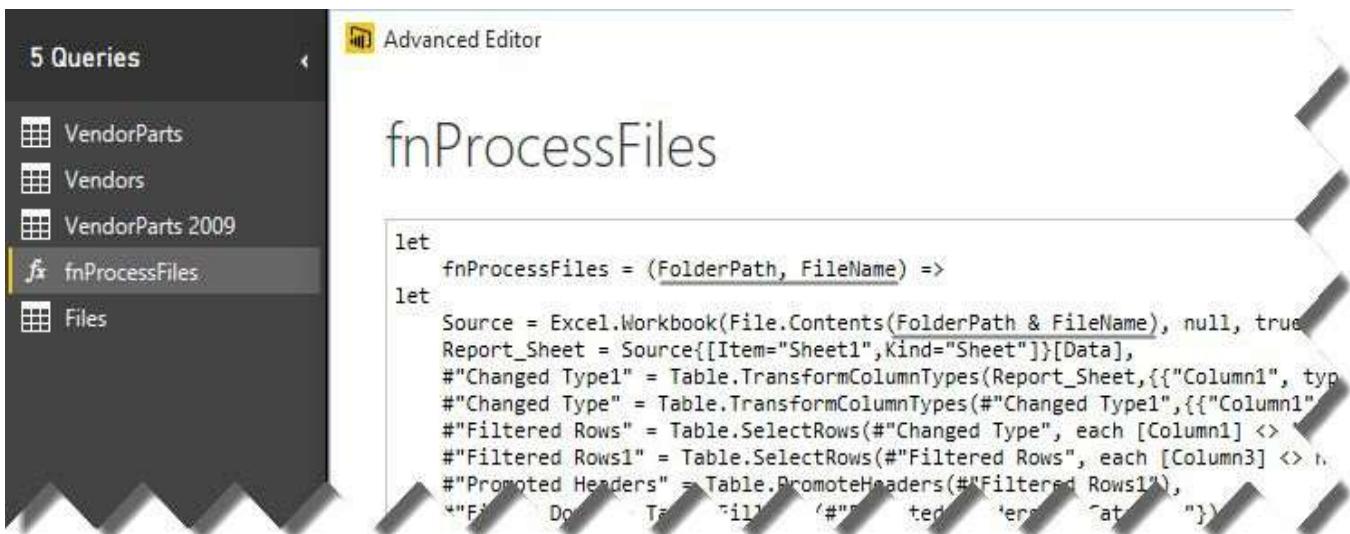


Figure 6.17 The fnProcessFiles function takes two parameters.

2. Now that the function allows you to specify the file location, instead of hardcoding the path to the Excel file, change the “Source =” line as follows:

```
Source = Excel.Workbook(File.Contents(FolderPath & FileName), null, true),
```

This code uses the ampersand (&) operator to concatenate the folder path and file name parameters. The reason why I don’t pass the full path as a single parameter will become obvious in a moment.

3. Click Done to close the Advanced Editor.

Invoking functions

Now that the function is ready, you need another query to invoke it. In our case, the outer query will loop through all files in a given folder and call the fnProcessFiles function to load the data from each file.

1. In the Query Editor, expand the New Source button (in the ribbon’s Home tab), and then click More.

1. In the Get Data window, click the File tab and select Folder. The Folder data source returns a list of files in a specific folder. Click Connect.

2. In the Folder window, specify the folder path where the Excel source files are located. For your convenience, I saved two Excel files in the \Source\ch06\Files folder. Click OK to create the query.

3. Rename the new query to *ProcessExcelFiles*.

TIP If the folder contains different types of files and you want to process files with a specific file extension(s), you can filter the Extension column in the data preview pane of the Query Editor. For example, to only process Excel 2007 or later files, filter the Extension column to include only the *.xlsx” file extension.

4. Each row in the ProcessExcelFiles query represents a file. You need to add a custom column that invokes the fnProcessFile function for each file.

5. In the Queries pane, select the ProcessExcelFiles query. In the ribbon’s Add Column tab, click “Add Custom Column”. Configure the custom column as shown in **Figure 6.18**, and the click OK.

Add Custom Column



Figure 6.18 Add a custom column to the outer query to invoke the fnProcessFiles query function.

6.The Query Editor adds the custom column to the ProcessExcelFiles query. Because the Query Editor understands that the function returns a table, it allows you to expand the column. In the data preview pane, expand the Custom column. In the Expand Custom window, leave all the columns selected. Assuming you don't want each column to be prefixed, in the "Default column name prefix" field, delete "Custom" so that the field is empty. Click OK.

For each file in the folder, the ProcessExcelFiles query calls fnProcessFiles which appends the results to a single dataset.

NOTE If you expand the dropdown of the Date column, you'll only see dates for year 2008, which might let you believe that you have data from one file only. This is actually not the case, but it's a logical bug where year 2008 is hardcoded in the query. If the year is specified in the file name, you can add another custom column that extracts the year, passes it to a third parameter in the fnProcessFiles function, and uses that parameter instead of hardcoded references to "2008".

6.3.3 Generating Date Tables

Now that you know about query functions, I'm sure you'll think of many real-life scenarios where you can use them automate routine data crunching tasks. Let's revisit a familiar scenario. As I mentioned in Chapter 5, even if you import a single dataset, you should strongly consider a separate date table. I also mentioned that there are different ways to import a date table, and one of them was to generate it in the Query Editor. The following code is based on an example by Matt Masson, as described in his "Creating a Date Dimension with a Power Query Script" blog post (<http://www.mattmasson.com/2014/02/creating-a-date-dimension-with-a-power-query-script>).

Generating dates

The Query Editor has useful functions for manipulating dates, such as for extracting date parts (day, month, quarter), and so on. The code uses many of these functions.

1. Start by creating a new blank query. To do so, in the Query Editor, expand the New Source button (the ribbon's Home tab) and click Blank Query. Rename the blank query to *GenerateDateTable*.
2. In the Queries pane, right-click the *GenerateDateTable* query and click Advanced Editor.
3. In the Advanced Editor, paste the following code which you can copy from the *GenerateDateTable.txt* file in the \Source\ch06 folder:

```
let GenerateDateTable = (StartDate as date, EndDate as date, optional Culture as nullable text) as table =>
    let
        DayCount = Duration.Days(Duration.From(EndDate - StartDate)),
        Source = List.Dates(StartDate, DayCount, #duration(1,0,0,0)),
        TableFromList = Table.FromList(Source, Splitter.SplitByNothing()),
        ChangedType = Table.TransformColumnTypes(TableFromList, {{"Column1", type date}}),
        RenamedColumns = Table.RenameColumns(ChangedType, {{"Column1", "Date"}}),
        InsertYear = Table.AddColumn(RenamedColumns, "Year", each Date.Year([Date])),
        InsertQuarter = Table.AddColumn(InsertYear, "QuarterOfYear", each Date.QuarterOfYear([Date])),
        InsertMonth = Table.AddColumn(InsertQuarter, "MonthOfYear", each Date.Month([Date])),
        InsertDay = Table.AddColumn(InsertMonth, "DayOfMonth", each Date.Day([Date])),
        InsertDayInt = Table.AddColumn(InsertDay, "DateInt", each [Year] * 10000 + [MonthOfYear] * 100 + [DayOfMonth]),
        InsertMonthName = Table.AddColumn(InsertDayInt, "MonthName", each Date.ToText([Date], "MMMM", Culture), type text),
        InsertCalendarMonth = Table.AddColumn(InsertMonthName, "MonthInCalendar", each
            try(Text.Range([MonthName],0,3)) otherwise [MonthName] & " " & Number.ToText([Year])),
        InsertCalendarQtr = Table.AddColumn(InsertCalendarMonth, "QuarterInCalendar", each "Q" &
            Number.ToText([QuarterOfYear]) & " " & Number.ToText([Year])),
        InsertDayWeek = Table.AddColumn(InsertCalendarQtr, "DayInWeek", each Date.DayOfWeek([Date])),
        InsertDayName = Table.AddColumn(InsertDayWeek, "DayOfWeekName", each Date.ToText([Date], "dddd", Culture), type text),
        InsertWeekEnding = Table.AddColumn(InsertDayName, "WeekEnding", each Date.EndOfWeek([Date]), type date)
    in
        InsertWeekEnding
    GenerateDateTable
```

This code creates a *GenerateDateTable* function that takes three parameters: start date, end date, and optional language culture, such as “en-US”, to localize the date formats and correctly interpret the date parameters. The workhorse of the function is the *List.Dates* method, which returns a list of date values starting at the start date and adding a day to every value. Then the function applies various transformations and adds custom columns to generate date variants, such as Year, QuarterOfYear, and so on.

Invoking the function

You don't need an outer query to generate the GenerateDateTable function because you don't have to execute it repeatedly. In this case, you simply want to invoke the function for a range of dates.

- 1.In the Queries pane, select the GenerateDateTable query, and then click the Invoke button in the preview pane.
- 2.In the Enter Parameters window (see **Figure 6.19**), enter StartDate and EndDate parameters. Click OK to invoke the function.
- 3.Watch how the Query Editor produces a date table in a second. If you want to regenerate the table with a different range of values, simply delete the “Invoked FunctionGenerateDateTable” step in the Applied Steps list, and then invoke the function again with different parameters.



Figure 6.19 Invoke the GenerateDateTable function and pass the required parameters.

6.4 Summary

Behind the scenes, when you import data, Power BI Desktop creates a query and there's a query for every table you import. Not only does the query give you access to the source data, but it also allows you to shape and transform data using various table and column-level transformations. To practice this, you applied a series of steps to shape and clean a crosstab Excel report so that its results can be used in a self-service data model.

You also practiced more advanced query features. You learned how to join, merge, and append datasets. Every step you apply to the query generates a line of code described in the M query language. You can view and customize the code to meet more advanced scenarios and automate repetitive tasks. You learned how to use query functions to automate importing files. And you saw how you can use custom query code to generate date tables if you can't import them from other places.

Next, you'll learn how to extend and refine the model to make it more feature-rich and intuitive to end users!

Chapter 7

Refining the Model

In the previous two chapters, you learned how to import and transform data. The next step is to explore and refine the model before you start gaining insights from it. Typical tasks in this phase include making table and field names more intuitive, exploring data, and changing the column type and formatting options. When your model has multiple tables, you must also set up relationships to relate tables.

In this chapter, you'll practice common tasks to enhance the Adventure Works model. First, you'll learn how to explore the loaded data and how to refine the metadata. Next, I'll show you how to manage schema and data changes, including managing connections and tables, and refreshing the model data to synchronize it with changes in the data sources. Finally, I'll walk you through the steps needed to set up table relationships and I'll discuss their limitations.

The screenshot shows the Power BI Desktop application window titled "Adventure Works - Power BI Desktop". The ribbon at the top has tabs for File, Home, and Modeling, with "Modeling" currently selected. A callout labeled "Modeling ribbon" points to the ribbon area. Below the ribbon is a toolbar with icons for Manage Relationships, New Measure, New Column, New Table, Sort By Column, Sort, and Formatting. The main area is the Data View, which displays a table of data from the DimDate table. The columns shown are DateKey, FullDateAlternateKey, DayNumberOfWeek, EnglishDayNameOfWeek, DayNumberOfMonth, DayNumberOfYear, and WeekNumberOfYear. The EnglishDayNameOfWeek column is highlighted with a yellow background and a red border, and a callout labeled "Column header" points to its header. The first few rows of the EnglishDayNameOfWeek column show values like Friday, Saturday, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday, Monday, Tuesday, Wednesday, and Thursday. To the right of the Data View is the Fields pane, which lists fields from the DimDate table and other tables like Employees and FactResellerSales. A callout labeled "Fields" points to the pane. At the bottom left, a status bar says "TABLE: DimDate (2,191 rows) COLUMN: EnglishDayNameOfWeek (7 distinct values)". At the bottom right, there is an "UPDATE AVAILABLE" button.

DateKey	FullDateAlternateKey	DayNumberOfWeek	EnglishDayNameOfWeek	DayNumberOfMonth	DayNumberOfYear	WeekNumberOfYear
20050701	Friday, July 1, 2005	6	Friday	1		
	Saturday, July 2, 2005	7	Saturday	2		
	Sunday, July 3, 2005	1	Sunday	3		
20050704	Monday, July 4, 2005	2	Monday	4		
20050705	Tuesday, July 5, 2005	3	Tuesday	5		
20050706	Wednesday, July 6, 2005	4	Wednesday	6		
20050707	Thursday, July 7, 2005	5	Thursday	7		
20050708	Friday, July 8, 2005	6	Friday	8		
20050709	Saturday, July 9, 2005	7	Saturday	9		
20050710	Sunday, July 10, 2005	1	Sunday	10		
20050711	Monday, July 11, 2005	2	Monday	11		
20050712	Tuesday, July 12, 2005	3	Tuesday	12		
20050713	Wednesday, July 13, 2005	4	Wednesday	13		
20050714	Thursday, July 14, 2005	5	Thursday	14		
20050715	Friday, July 15, 2005	6	Friday	15		
20050716	Saturday, July 16, 2005	7	Saturday	16		
20050717	Sunday, July 17, 2005	1	Sunday	17		
20050718	Monday, July 18, 2005	2	Monday	18		
20050719	Tuesday, July 19, 2005	3	Tuesday	19		
20050720	Wednesday, July 20, 2005	4	Wednesday	20		
20050721	Thursday, July 21, 2005	5	Thursday	21		

Figure 7.1 In the Data View, you can browse the model schema and data.

7.1 Understanding Tables and Columns

Power BI stores imported data in tables. Although the data might originate from heterogeneous data sources, once it enters the model, it's treated the same irrespective of its origin. Similar to a relational database, a table consists of columns and rows. You can use the Data View to explore a table and its data, as shown in **Figure 7.1**.

7.1.1 Understanding the Data View

The Power BI Desktop navigation bar (the vertical bar on the left) has three view icons: Report, Data, and Relationships. As its name suggests, the Data View allows you to browse the model data. In contrast, the Relationships View only shows a graphical representation of the model schema. And the Reports View is for creating visualizations that help you analyze the data. In Chapter 4, I covered how the Data View shows the imported data from the tables in the model. This is different from the Query Editor data preview, which shows the source data and how it's affected by the transformations you've applied.

Understanding tables

The Fields pane shows you the model metadata that you interact with when creating reports. When you select a table in the Fields pane, the Data View shows you the first rows in the table. As it stands, the Adventure Works model has six tables. The Data View and the Fields pane (when are in Data View) shows the metadata in the order they appear in the source dataset. However, to help you locate table and fields easier when you create reports, the Fields pane in the Reports View sorts the metadata (table names and column names) in an ascending order by name.

NOTE What's the difference between a column and a field anyway? A field in the Fields pane can be a table column or a calculated measure, such as SalesYTD. A calculated measure doesn't specifically map to a table column.

The table name is significant because it's included in the model metadata, and it's shown to the end user. In addition, when you create calculated columns and measures, the Data Analysis Expressions (DAX) formulas reference the table and field names. Therefore, spend some time to choose suitable names and to rename tables and fields accordingly.

TIP When it comes to naming conventions, I like to keep table and column names as short as possible so that they don't occupy too much space in report labels. I prefer camel casing where the first letter of each word is capitalized. I also prefer to use a plural case for fact tables, such as ResellerSales, and a singular case for lookup (dimension) tables, such as Reseller. You don't have to follow this convention, but it's important to have a consistent naming convention and to stick to it. While I'm on this subject, Power BI supports identical column names across tables, such as SalesAmount in the ResellerSales table and SalesAmount in the InternetSales table. However, some reporting tools, such as Power BI reports, don't support renaming fields on the report, and you won't be able to tell the two fields apart if a report has both fields. Therefore, consider renaming the columns and adding a prefix to have unique column names across tables, such as ResellerSalesAmount and InternetSalesAmount. Or you can create DAX calculations with unique names and then hide the original columns.

The status bar at the bottom of the Data View shows the number of rows in the selected table. When you select a field, the status bar shows the number of its distinct values. For example, the EnglishDayNameOfWeek field has seven distinct values. This is useful to know because that's how many values the users will see when they add this field to the

report.

Understanding columns

The vertical bands in a table open in the Data View represent the table columns. You can click any cell to select the entire column and to highlight the column header. The Formatting group in the ribbon's Modeling tab shows the data type of the selected column. Similar to the Query Editor data preview, Data View is read-only. You can't change the data – not even a single cell. Therefore, if you need to change a value, such as when you find a data error that requires a correction, you must make the changes either in the data source or in the query. As I mentioned before, I encourage you to make data changes in the query.

Another way to select a column is to click it in the Fields pane. The Fields pane prefixes some fields with icons. For example, the sigma (S) icon signifies that the field is numeric and can be aggregated using any of the supported aggregate functions, such as Sum or Average. If the field is a calculated measure, it'll be prefixed with a calculator icon (⌚). Even though some fields are numeric, they can't be meaningfully aggregated, such as CalendarYear. The Properties group in the ribbon's Modeling tab allows you to change the default aggregation behavior, such as to change the CalendarYear default aggregation to "Do not aggregate". This is just a default; you and other users can still overwrite the aggregation on reports.

The Data Category property in the Properties group (ribbon's Model tab) allows you to change the column category. For example, to help Power BI understand that this is a geospatial field, you can change the data category of the SalesTerritoryCountry column to Country/Region. This will prefix the field with a globe icon. More importantly, this helps Power BI to choose the best visualization when you add the field on an empty report, such as to use a map visualization when you add a geospatial field.

7.1.2 Exploring Data

If there were data modeling commandments, the first one would be "Know thy data". Realizing the common need to explore the raw data, the Power BI team has added features to both the Query Editor and Data View to help you become familiar with the source data.

Sorting data

Power BI doesn't sort data by default. As a result, Data View shows the imported data as it's loaded from the source. You can right-click a column and use the sort options (see **Figure 7.2**) to sort the data.

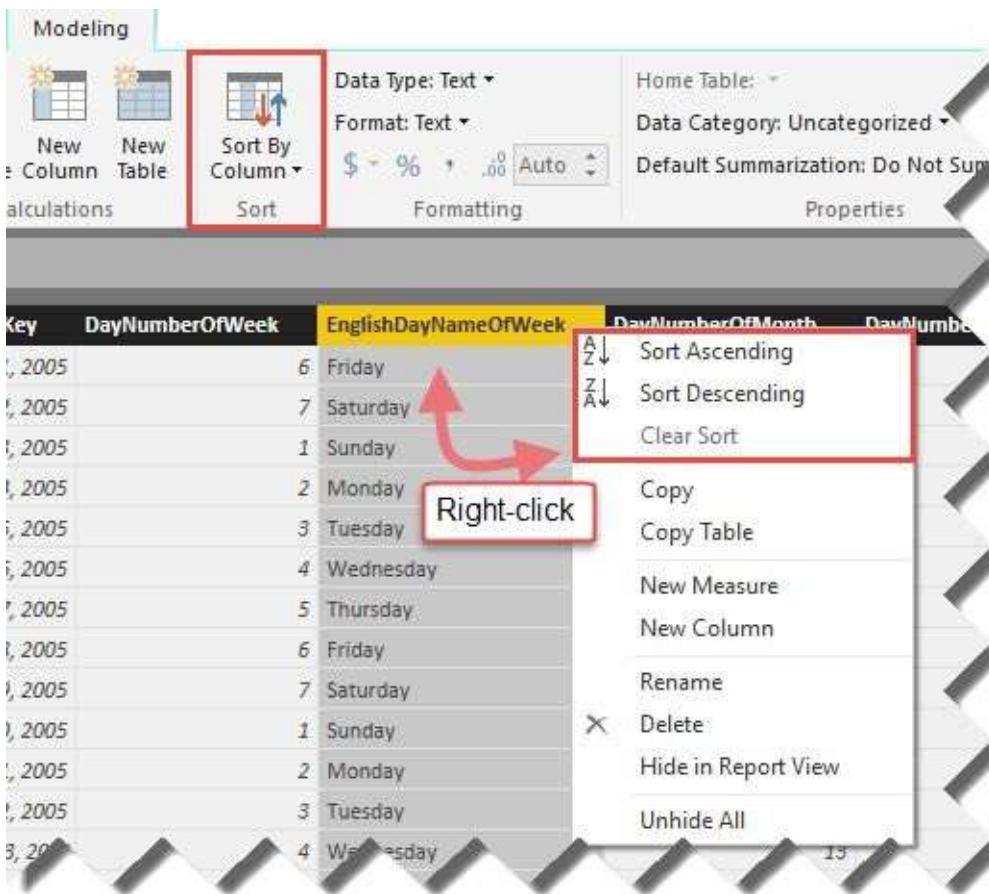


Figure 7.2 You can sort the field content in ascending or descending order.

You can sort the content of a table column in an ascending or descending order. This type of sorting is for your benefit, because it allows you to get familiar with the imported data, such as to find what's the minimum or maximum value. Power BI doesn't apply the sorting changes to the way the data is saved in the model, nor it propagates the column sort to reports. For example, you might sort the EnglishDayNameOfWeek column in a descending order. However, when you create a report that uses this field, the visualization would ignore the Data View sorting changes and it will sort days in an ascending order (or whatever sort order the reporting tool prefers).

When a column is sorted in the Data View, you'll see an up or down arrow in the column header, which indicates the sort order. You can sort the table data by only one column at a time. To clear sorting and to revert to the data source sort order, right-click a column, and then click Clear Sort.

NOTE Power BI Desktop automatically inherits the data collation based on the language selection in your Windows regional settings, which you can overwrite in the Options and Settings ð Option ð Data Load (Current File section). The default collations are case-insensitive. Consequently, if you have a source column with the values "John" and "JOHn", then Power BI Desktop imports both values as "John" and treats them the same. While this behavior helps the xVelocity storage engine compress data efficiently, sometimes a case-sensitive collation might be preferable, such as when you need a unique key to set up a relationship, and you get an error that the column contains duplicate values. However, currently there isn't an easy way to change the collation and configure a given field or a table to be case-sensitive. So you'll need to try to keep the column names distinct.

Custom sorting

Certain columns must be sorted in a specific order on reports. For example, calendar months should be sorted in their ordinal position (Jan, Feb, and so on) as opposed to

alphabetically. To accomplish this, Power BI supports custom sorting. Custom sorting allows you to sort a column by another column, assuming the column to sort on has one-to-one or one-to-many cardinality with the sorted column.

Let's say you have a column MonthName with values Jan, Feb, Mar, and so on, and you have another column MonthNumberOfYear that stores the ordinal number of the month in the range from 1 to 12. Because every value in the MonthName column has only one corresponding value in MonthNumberOfYear column, you can sort MonthName by MonthNumberOfYear. However, you can't sort MonthName by a Date column because there are multiple dates for each month.

Compared to field sorting for data expiration, custom sorting has a reverse effect on data. Custom sorting doesn't change the way the data is displayed in the Data View, but it affects how the data is presented in reports. **Figure 7.3** shows how changing custom sorting will affect the sort order of the month name column on a report.

EnglishMonthName	EnglishMonthName
April	January
August	February
December	March
February	April
January	May
July	June
June	July
March	August
May	September
November	October
October	November
September	December

Figure 7.3 The left table shows the month with the default alphabetical sort order while the right table shows it after custom sorting was applied by MonthNumberOfYear.

Copying data

Sometimes you might want to copy the content of a column (or even an entire table) and paste it in Excel or send it to someone. You can use the Copy and Copy Table options from the context menu (see **Figure 7.2** again) to copy the content to Windows Clipboard and paste it in another application. You can't paste the copied data into the data model. Again, that's because the data model is read-only.

The Copy Table option is also available when you right-click a table in the Fields pane. Copying a table preserves the tabular format, so pasting it in Excel produces a list with columns instead of a single column.

7.1.3 Understanding the Column Data Types

A table column has a data type associated with it. In Chapter 5, I mentioned that a query

column already has a data type. When a query connects to the data source, it attempts to infer the column data type from the data provider and then maps it to one of the data types it supports. Although it seems redundant to have data types in two places (query and storage), it gives you more flexibility. For example, you can keep the inferred data type in the query but change it in the table.

Currently, there isn't a one-to-one mapping between query and storage data types. Instead, Power BI Desktop maps the query column types to the ones that the xVelocity storage engine supports. Table 7.1 shows these mappings. Queries support a couple of more data types (Date/Time/Timezone and Duration) than table columns.

Table 7.1 This table shows how query data types map to column data types.

Query Data Type	Storage Data Type	Description
Text	String	A Unicode character string with a max length of 268,435,456 characters
Decimal Number	Decimal Number	A 64 bit (eight-bytes) real number with decimal places
Fixed Decimal Number	Fixed Decimal Number	A decimal number with four decimal places of fixed precision useful for storing currencies.
Whole Number	Whole number	A 64 bit (eight-bytes) integer with no decimal places
Date/Time	Date/Time	Dates and times after March 1 st , 1900
Date	Date	Just the date portion of a date
Time	Time	Just the time portion of a date
Date/Time/Timezone	Date	Universal date and time
Duration	Text	Time duration, such as 5:30 for five minutes and 30 seconds
TRUE/FALSE	Boolean	True or False value
Binary	Binary data type	An image or blob

How data types get assigned

The storage data type has preference over the query. For example, the query might declare a column date type as Decimal Number, and this type might get carried over to storage. However, you can overwrite the column data type in the Data View to Whole Number. Unless you change the data type in the query and apply the changes, the column data type remains Whole Number.

The storage engine tries to use the most compact data type, depending on the column values. For example, the query might have assigned a Fixed Decimal Number data type to a column that has only whole numbers. Don't be surprised if the Data View shows the column data type as Whole Number after you import the data. Power BI might also perform a widening data conversion on import if it doesn't support certain numeric data types. For example, if the underlying SQL Server data type is tinyint (one byte), Power BI will map it to Whole Number because that's the only data type that it supports for whole numbers.

Power BI won't import data types it doesn't recognize and won't import the corresponding columns. For example, Power BI won't import a SQL Server column of a geography data type that stores geodetic spatial data. The Binary data type can be used to import images. Importing images is useful for creating banded reports, such as to allow you to click a product image and filter data for that product.

If the data source doesn't provide schema information, Power BI imports data as text and uses the Text data type for all the columns. In such cases, you should overwrite the data types after import when it makes sense.

Changing the column data type

As I mentioned, the Formatting group in ribbon's Modeling tab and the Transform group in the Query Editor indicate the data type of the selected column. You should review and change the column type when needed, for the following reasons:

- Data aggregation – You can sum or average only numeric columns.
- Data validation – Suppose you're given a text file with a SalesAmount column that's supposed to store decimal data. What happens if an 'NA' value sneaks into one or more cells? The query will detect it and might change the column type to Text. You can examine the data type after import and detect such issues. As I mentioned in the previous chapter, I recommend you react to such issues in the Query Editor because it has the capabilities to remove errors or replace values. Of course, it's best to fix such issues at the data source, but probably you won't have write access to the source data.

NOTE What happens if all is well with the initial import, but a data type violation occurs the next month when you are given a new extract? What really happens in the case of a data type mismatch depends on the underlying data provider. The text data provider (Microsoft ACE OLE DB provider) replaces the mismatched data values with blank values, and the blank values will be imported in the model. On the query side of things, if data mismatch occurs, you'll see "Error" in the corresponding cell to notify you about dirty data, but no error will be triggered on refresh.

- Better performance – Smaller data types have more efficient storage and query performance. For example, a whole number is more efficient than text because it occupies only eight bytes irrespective of the number of digits.

Sometimes, you might want to overwrite the column data type in the Data View. You can do so by expanding the Data Type drop-down list in the Formatting ribbon group and then select another type. Power BI Desktop only shows the list of the data types that are applicable for conversion. For example, if the original data type is Currency, you can convert the data type to Text, Decimal Number, and Whole Number. If the column is of a Text data type, the Data Type drop-down list would show all the data types. However, you'll get a type mismatch error if the conversion fails, such as when trying to convert a non-numeric text value to a number.

Understanding column formatting

Each column in the Data View has a default format based on its data type and Windows regional settings. For example, my default format for Date columns is MM/dd/yyyy hh:mm:ss tt because my computer is configured for English US regional settings (such as 12/24/2011 13:55:20 PM). This might present an issue for international users. However, they can overwrite the language from the Power BI Desktop File → Options menu and

change the formatting. Changing column formatting has no effect on how data is stored because the column format is for visualization purposes only. Use the Formatting group in the ribbon's Modeling tab to change the column format settings, as shown in **Figure 7.4**.

You can use the format buttons in the Formatting ribbon group to apply changes interactively, such as to add a thousand separator or to increase the number of decimal places. Formatting changes apply automatically to reports the next time you switch to the Reports View. If the column width is too narrow to show the formatted values in Data View, you can increase the column width by dragging the right column border. Changing the column width in Data View has no effect on reports.

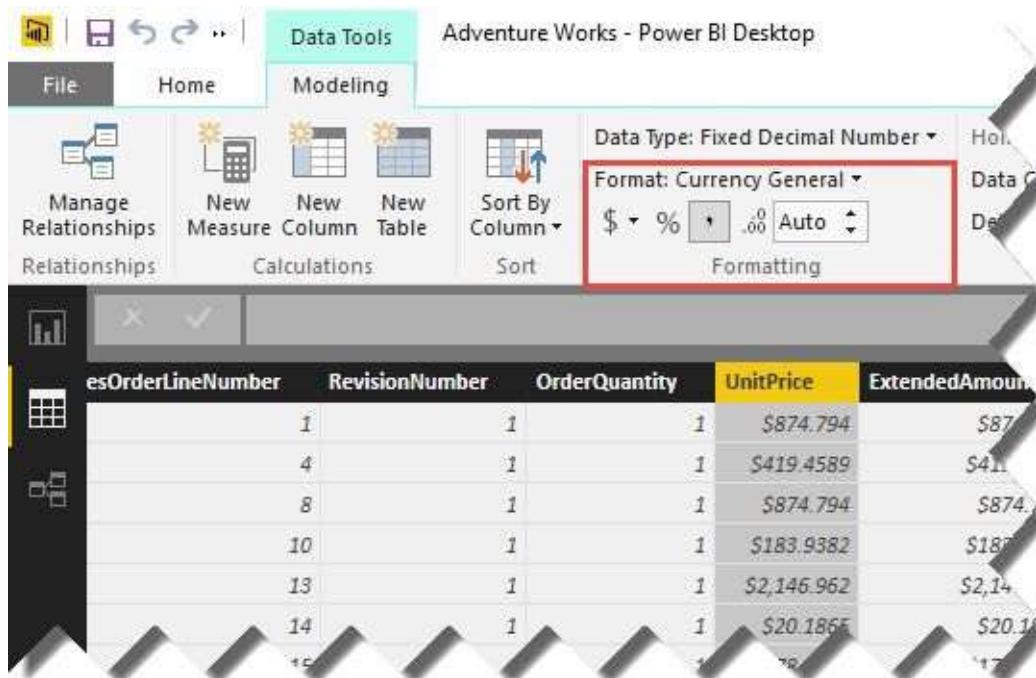


Figure 7.4 Use the Formatting ribbon group to change the column format.

7.1.4 Understanding Column Operations

You can perform various column-related tasks to explore data and improve the metadata visual appearance, including renaming columns, removing columns, and hiding columns.

Rename columns

Table columns inherit their names from the underlying query that inherits them in turn from the data source. These names might be somewhat cryptic, such as TRANS_AMT. The column name becomes a part of the model metadata that you and the end users interact with. You can make the column name more descriptive and intuitive by renaming the column. You can rename a column interchangeably in three places: the Data View, Query Editor, and Fields pane. For example, if you rename a column in the Data View and then switch to the Query Editor, you'll see that Power BI Desktop has automatically appended a Rename Column transformation step to apply the change to the query.

NOTE No matter where you rename the column, the Power BI “smart rename” applies throughout all the column references, including calculations and reports to avoid broken references. You can see the original name of the column in the data source by inspecting the Rename Column step in the Query Editor formula bar or by looking at the query source.

To rename a column in the Data View, double-click the column header to enter edit mode, and then type in the new name. Or, right-click the column, and then click Rename Column (see **Figure 7.2** again). To rename a column in the Fields pane (in the Data and Report Views), right-click the column and click Rename (or double-click the column).

Removing and hiding columns

In Chapter 5, I advised you to not import a column that you don't need in the model. However, if this ever happens, you can always remove a column in the Data View, Query Editor, and Fields pane. I also recommended you use the Choose Columns transformation in Query Editor as a more intuitive way to remove and add columns. If the column participates in a relationship to another table in the data model, removing the column removes the associated relationship(s).

Suppose you need the column in the model, but you don't want to show it to end users. For example, you might need a primary key column or foreign key column to set up a relationship. Since such columns usually contain system values, you might want to exclude them from showing up in the Fields pane by simply hiding them. The difference between removing and hiding a column is that hiding a column allows you to use the column in the model, such as in hierarchies or custom sorting, and in DAX formulas.

To hide a column in Data View, right-click any column cell click "Hide in Report View". A hidden column appears grayed out in Data View. You can also hide a column in the Fields pane by right-clicking the column and clicking Hide. If you change your mind later on, you can unhide the column by toggling "Hide in Report View" (see **Figure 7.5**). Or, you can click Unhide All to unhide all the hidden columns in the selected table.

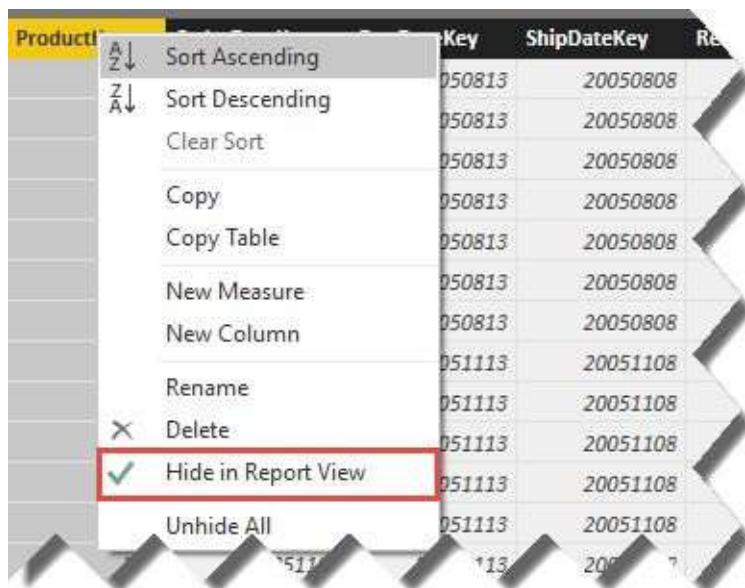


Figure 7.5 Toggle the "Hide in Report View" menu to hide or unhide a column.

Finally, the Copy operation allows you to copy the content of the selected columns and paste it in another application, such as in Microsoft Excel. You can't paste the column content inside the Data View because the model is read-only.

7.1.5 Working with Tables and Columns

Now that you're familiar with tables and columns, let's turn our attention to the Adventure Works model and spend some time exploring and refining it. The following steps will help you get familiar with the common tasks you'll use when working with tables and columns.

Sorting data

You can gain insights into your imported data by sorting and filtering it. Suppose that you want to find which employees have been with the company the longest:

- 1.In Power BI Desktop, open the Adventure Works.pbix file.

NOTE If you haven't completed the Chapter 5 exercises, you can use the Adventure Works file from the \Source\Ch05 folder. However, remember that my samples import data from several local data sources, including the Adventure Works cube and the Product Catalog report. You need to update all the data sources to reflect your specific setup. To do so, open the Query Editor, and then double-click the Source step in the Query Settings pane for each data source that fails to refresh. Then, change the server name and database name as needed.

- 2.Click Data View in the navigation bar. Click the Employees tab in the Fields pane.
- 3.Right-click the HireDate column, and then click Sort Ascending. Note that Guy Gilbert is the first person on the list, and he was hired on 7/31/1997.
- 4.Right-click the HireDate column again, and then click the Clear Sort menu to remove the sort and to revert to the original order in which data was imported from the data source.

Implementing a custom sort

Next, you'll sort the EnglishMonthName column by the MonthNumberOfYear column so that months are sorted in their ordinal position on reports.

1. In the Fields pane, click the DimDate table to select it.
- 1.Click a cell in the EnglishMonthName column to select this column.
- 2.In the ribbon's Modelling tab, click the "Sort by Column" button, and then select MonthNumberOfYear.
- 3.(Optional) Switch to the Reports View, and create a visualization that uses EnglishMonthName. The months should be sorted in their ordinal position.

Renaming tables

The name of the table is included in the metadata that you'll see when you create reports. Therefore, it's important to have a naming convention for tables. In this case, I'll use a plural naming convention for fact tables (tables that keep a historical record of business transactions, such as ResellerSales), and a singular naming convention for lookup tables.

1. Double-click the DimDate table in the Fields pane and rename it to *Date*.
- 1.To practice another way for renaming a table, click the Edit Queries button to open the Query Editor. In the Queries pane, select the Employees query. In the Query Settings pane, rename the query to *Employee*. Click the "Apply & Close" button to return to the Data View.
- 2.Rename the rest of the tables using the Fields pane. Rename FactResellerSales to *ResellerSales*, Products to *Product*, Resellers to *Reseller*, and SalesTerritories to *SalesTerritory*.

Working with columns

Next, let's revisit each table and make column changes as necessary.

1. In the Fields pane (in the Data View), select the Date table. Double-click the column header of the FullDateAlternateKey column, and then rename it to *Date*. Decrease the Date column width by dragging the column's right border so it's wide enough to accommodate the content in the column. Rename the EnglishDayNameOfWeek column to *DayNameOfWeek* and EnglishMonthName to *MonthName*. Right-click the DateKey column and click "Hide in Report View" to hide this column.
1. You can also hide columns in the Fields pane. In the Fields pane, expand the Employee table. Right-click the EmployeeKey column and then click "Hide in Report View". Also hide the ParentEmployeeKey and SalesTerritoryKey columns. Using the Data View or Fields pane, delete the EmployeeNationalIDAlternateKey and ParentEmployeeNationalIDAlternateKey columns because they're sensitive columns that probably shouldn't be available for end-user analysis.
2. Click the Product table tab. If you've imported the Product table from the Product Catalog report, rename the ProdCat2 column to *ProductCategory*. Increase the column width to accommodate the content. Rename ProdSubCat column to *ProductSubcategory*, ProdModel to *ProductModel*, and ProdName to *ProductName*. Hide the ProductKey column. Using the ribbon's Modeling tab (Data View), reformat the StandardCost and ListPrice columns as Currency \$ English (United States). Hide the ProductKey column.
3. Select the Reseller table. Hide the ResellerKey and GeographyKey columns. Rename the ResellerAlternateKey column to *ResellerID*.
4. Select the ResellerSales table. The first nine foreign key columns (with the "Key" suffix) are useful for data relationships but not for data analysis. Hide them.
5. To practice formatting columns, change the format of the SalesAmount column to two decimal places. To do so, select the column in the Data View, and then click the Increase Decimal button twice in the Formatting group on the ribbon's Modeling tab.
6. Click the SalesTerritory table tab. Hide the SalesTerritoryKey column. Rename the SalesAmount column to Revenue. Format the Revenue column as Currency \$ English (United States).
7. Save the Adventure Works data model.

7.2 Managing Schema and Data Changes

To review, once Power BI Desktop imports data, it saves a copy of the data in a local file with a *.pbix file extension. The model schema and data is *not* automatically synchronized with changes in the data sources. Typically, after the initial load, you'll need to refresh the model data on a regular basis, such as when you receive a new source file or when the data warehouse is updated. Power BI Desktop provides features to keep your model up to date.

7.2.1 Managing Connections and Tables

It's not uncommon for a model to have several tables connected to different data sources so that you can integrate data from multiple places. As a modeler, you need to understand how to manage connections and tables.

Managing data sources

Suppose you need to import additional tables from a data source that you've already set up a connection to. One option is to use Get Data again. However, increasing the number of connections will increase the effort required to manage them. For example, if the security credentials change, you'll need to update multiple connection definitions. A better way is to use the Recent Sources button in the ribbon's Home tab (see **Figure 7.6**).

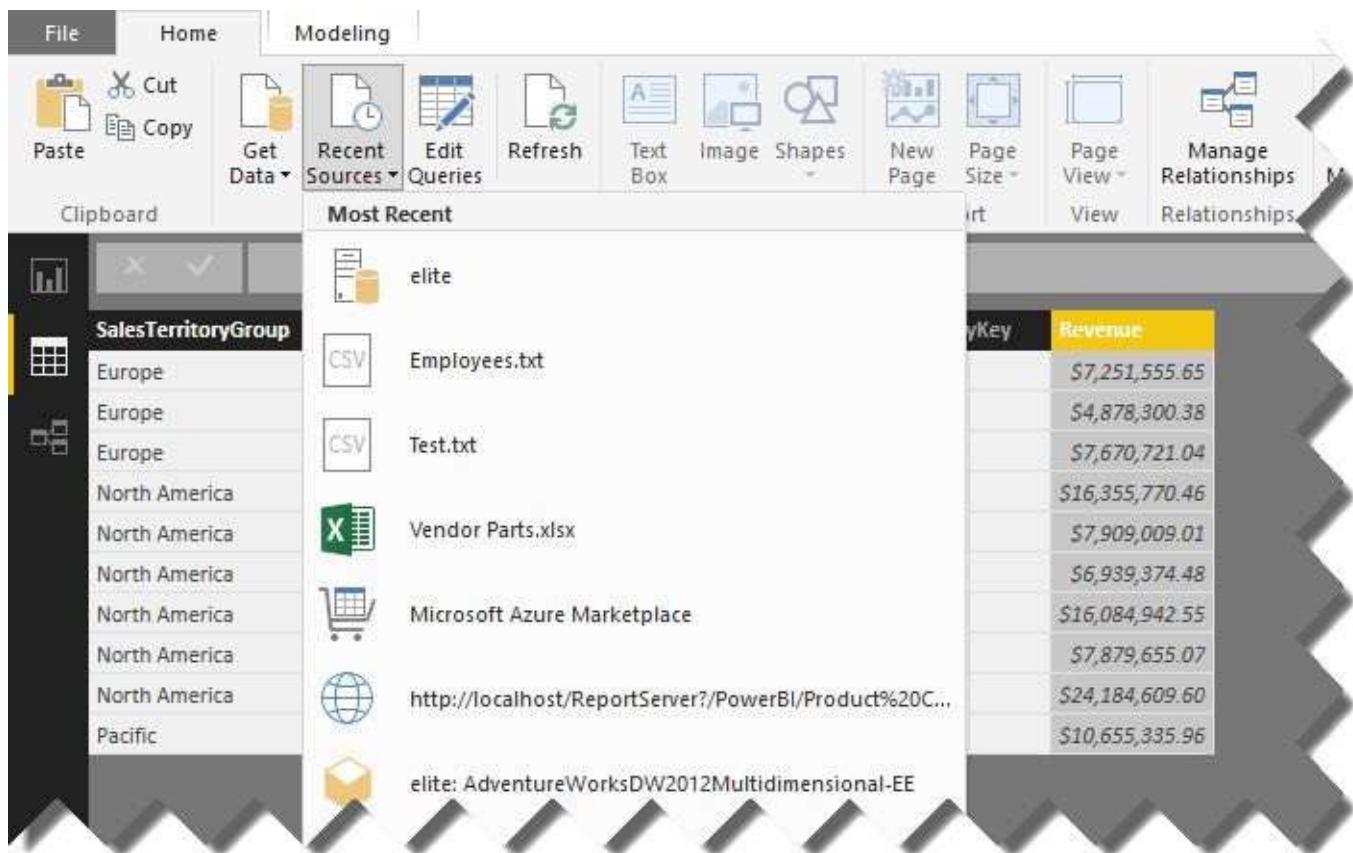


Figure 7.6 Use the Data Source Settings window to manage the data source credentials and encryption options in one place.

If you connect to a data source that has multiple entities, such as a relational database, when you click the data source in Recent Sources, Power BI Desktop will bring you straight to the Navigator window so that you can select and import another table. And you

can use the File → Options and Settings → Data Source Settings menu to manage the security credentials in one place. The Data Source Settings window (see **Figure 7.7**) shows the data sources that you connected to from Power BI Desktop on your computer.

NOTE Readers familiar with Excel data modelling might recall that Power Pivot has an Existing Connections window that allow you to manage connections to data sources in one place. The Data Source Settings window fulfills a similar purpose but it doesn't allow you to change the server and database names. If the server name or database name changes, you need to update the queries in the Query Editor accordingly.

You can select a data source and click the Edit button to change connection credentials or encryption options if the data source supports encryptions. These changes will apply to all the queries and Power BI Desktop models that use this data source. That's because Power BI Desktop encrypts the connection credentials and stores them in the local AppData folder on your computer.

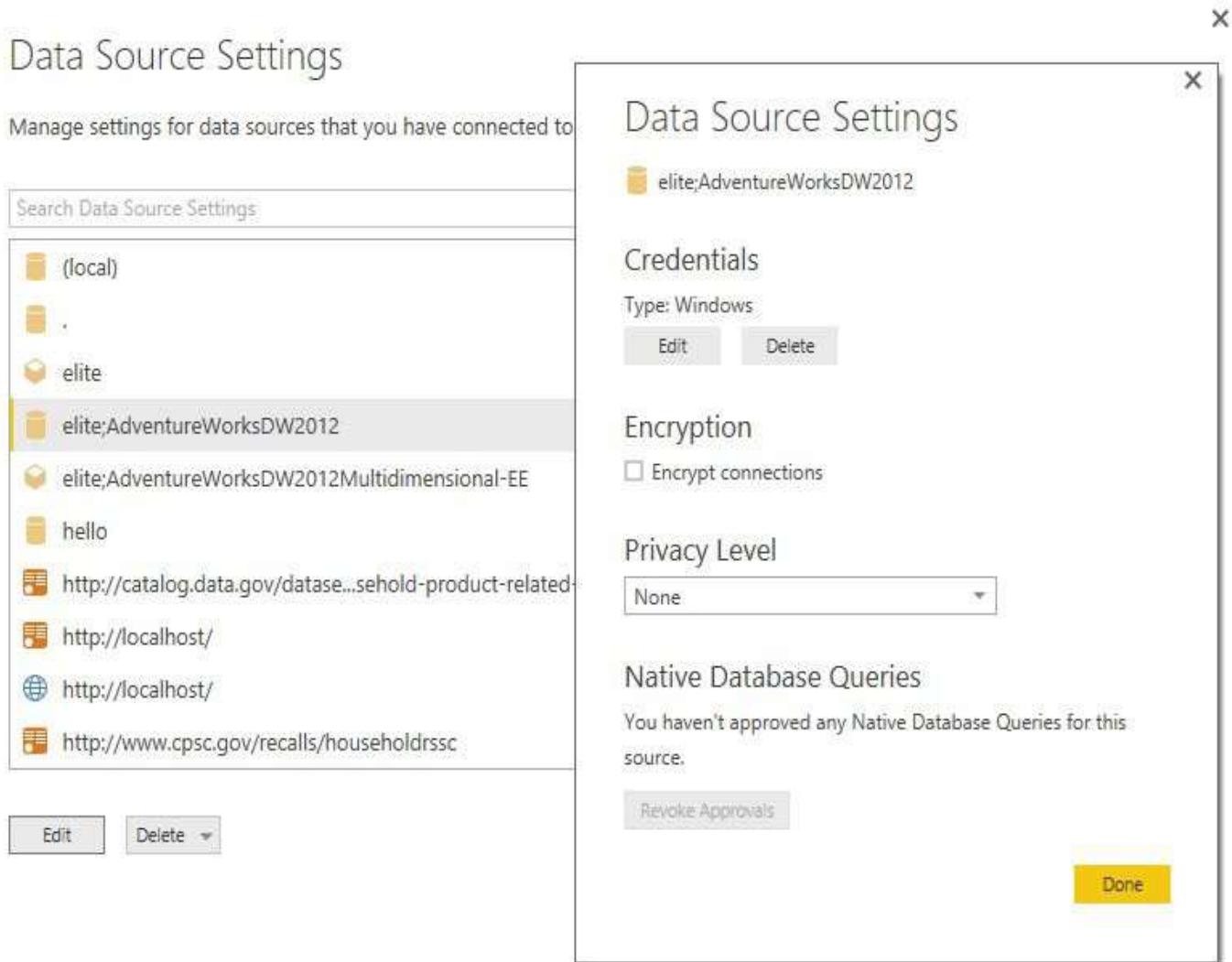


Figure 7.7 Click the Recent Sources button to access the data sources you've already used.

Deleting data sources

Every time you use Get Data and connect to a data source, the data source is added to the list for each server and database name. For example, if you don't use a database name, a data source with only the server name is added. If you use both localhost and (local) to connect, you'll end up with two data sources (although only one of them might be used by a query). This can clutter the list of data sources in the Data Sources Settings window.

To tidy the list up, you can delete data sources. When you do so, Power BI Desktop deletes the associated encrypted credentials. Unfortunately, the Data Source Settings window doesn't show which data sources are used by queries. If you delete a data source that's used by a query, nothing gets broken. However, the next time you refresh, you'll be asked to specify credentials and encryption options as you did the first time you used Get Data to connect to that data source. Then the data source will be added to the Recent Sources list.

Changing server and database names

Another limitation of the Data Source Settings window is that it doesn't allow you to change the server and database names, such as when you move from Development to Production environment. To do so, you must go to Query Editor and double-click the Source step in the Query Settings pane, as shown in

Figure 7.8. If you have left the database name empty, you can also use the Navigation step to change the database if you need to.

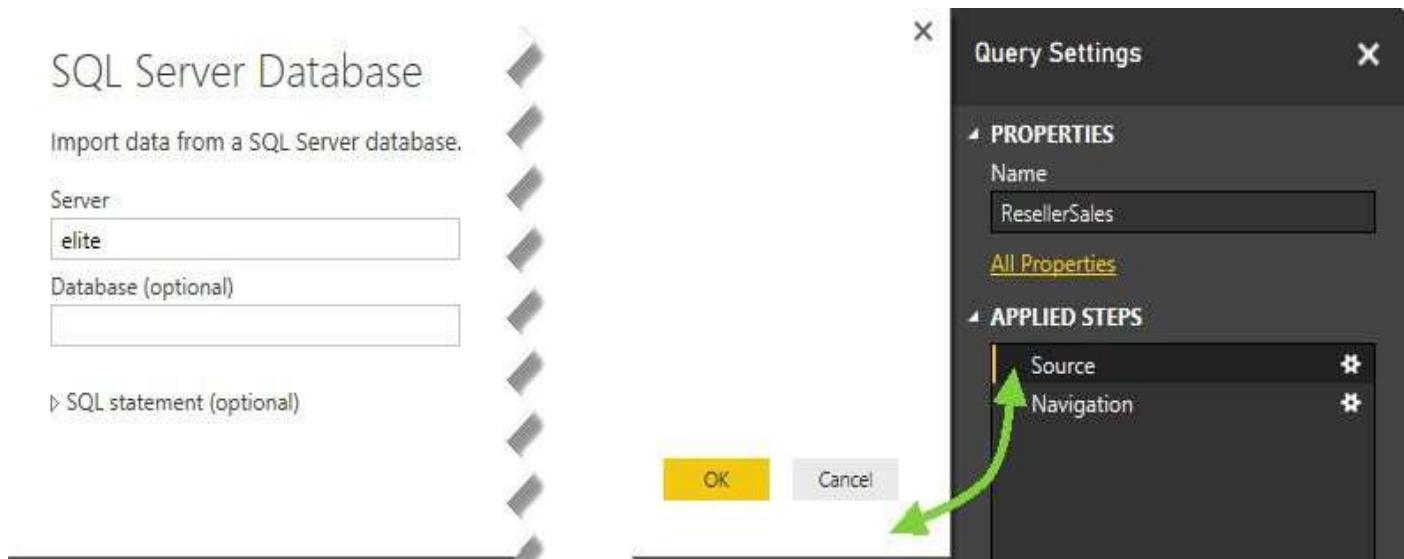


Figure 7.8 You can double-click the Source query step to change the server and database names.

Importing additional tables

Besides wholesale data, the Adventure Works data warehouse stores retail data for direct sales to individual customers. Suppose that you need to extend the Adventure Works model to analyze direct sales to customers who placed orders on the Internet. Follow these steps to import three additional tables:

NOTE Other self-service tools on the market restrict you to analyzing single datasets only. If that's all you need, feel free to skip this exercise as the model has enough tables and complexity already. However, chances are that you might need to analyze data from different subject areas side by side. This requires you to import multiple fact tables and join them to common dimensions. And this is where Power BI excels because it allows you to implement self-service models whose features are on a par with professional models. So, I encourage you to stay with me as the complexity cranks up and master these features so you never say "I can't meet this requirement".

- 1.In the ribbon's Home tab, expand the Recent Sources button, and then click the SQL Server instance that hosts the AdventureWorksDW database.
- 2.In the Navigator window, expand the AdventureWorksDW database, and then check the DimCustomer, DimGeography, and FactInternetSales tables. In the AdventureWorksDW

database, the DimGeography table isn't related directly to the FactInternetSales table. Instead, DimGeography joins DimCustomer, which joins FactInternetSales. This is an example of a snowflake schema, which I covered in Chapter 5.

- 3.Click the Edit button. In the Queries pane of the Query Editor, select DimCustomer and change the query name to *Customer*.
- 4.In the Queries pane, select DimGeography and change the query name to *Geography*.
- 5.Select the FactInternetSales query and change its name to InternetSales. Use the Choose Columns transformation to exclude the RevisionNumber, CarrierTrackingNumber, and CustomerPONumber columns.
- 6.Click “Close & Apply” to add the three tables to the Adventure Works model and to import the new data.
- 7.In the Data View, select the Customer table. Hide the CustomerKey and GeographyKey columns. Rename the CustomerAlternateKey column to *CustomerID*.
- 8.Select the Geography table, and hide the GeographyKey column.

Select the InternetSales table, and hide the first eight columns (the ones with “Key” suffix).

7.2.2 Managing Data Refresh

When you import data, Power BI Desktop caches it in the model to give you the best performance when you visualize the data. The only option to synchronize data changes on the desktop is to refresh the data manually.

Refreshing data

As it stands, Power BI Desktop refresh is simple. You just need to click the Refresh button in the Reports View or in the Data View. This executes all the table queries, discards the existing data, and imports all the tables from scratch. Currently, there isn't an option to process specific tables or to process only a portion of a table.

NOTE Analysis Services Tabular, which also uses xVelocity as a storage engine, supports more processing options, including processing specific partitions of a large table. You should consider Tabular when your data volumes exceed a few millions rows.

Suppose that you've been notified about changes in one or more of the tables, and now you need to refresh the data model.

- 1.In Power BI Desktop, click the Data View icon (or the Reports View icon) in the navigation bar.
- 2.In the ribbon's Home tab, click the Refresh button to refresh all tables.
- 3.Press Ctrl-S to save the Adventure Works data model.

When you initiate the refresh operation, Power BI Desktop opens the Refresh window to show you the progress, as shown in see **Figure 7.9**.

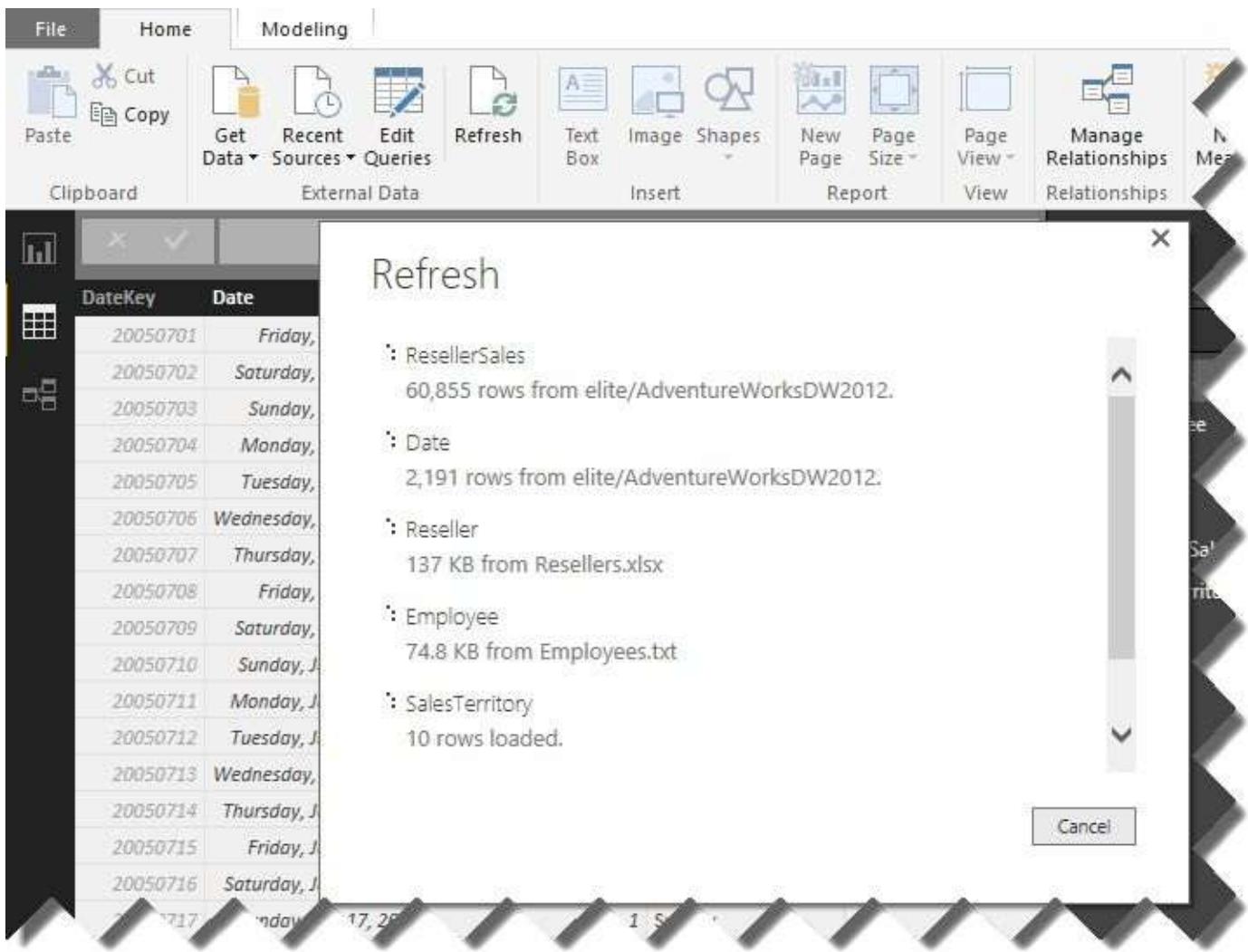


Figure 7.9 Power BI Desktop refreshes tables sequentially and cancels the entire operation if a table fails to refresh.

Power BI Desktop refreshes tables sequentially, one table at the time. The Refresh window shows the number of rows imported. You can't cancel the refresh once it has started.

NOTE Based on usability feedback, Microsoft decided on the sequential data refresh in Power Pivot and Power BI Desktop for easier failure analysis. If a table fails to refresh, the entire refresh operation stops so that the user can more easily identify which table failed.

The xVelocity storage engine is capable of importing more than 100,000 rows per second. The actual data refresh speed depends on many factors, including how fast the data source returns rows, the number and data type of columns in the table, the network speed, your machine hardware configuration, and so on.

REAL LIFE I was called a few times to troubleshoot slow processing issues with Analysis Services and Power Pivot. In all the cases, I've found that the external factors impacted the processing speed. For example, a large fact table might be missing an index, or the data warehouse server could be overwhelmed. In one case, it turned out that the IT department had decided to throttle the network speed on all non-production network segments in case a computer virus takes over.

Troubleshooting data refresh

If a table fails to refresh, such as when there's no connectivity to the data source, the Refresh window shows an error indicator and displays an error message, as shown in **Figure 7.10**. When a table fails to refresh, the entire operation is aborted because it runs in

a transaction, and no data is refreshed. At this point, you need to troubleshoot the error.

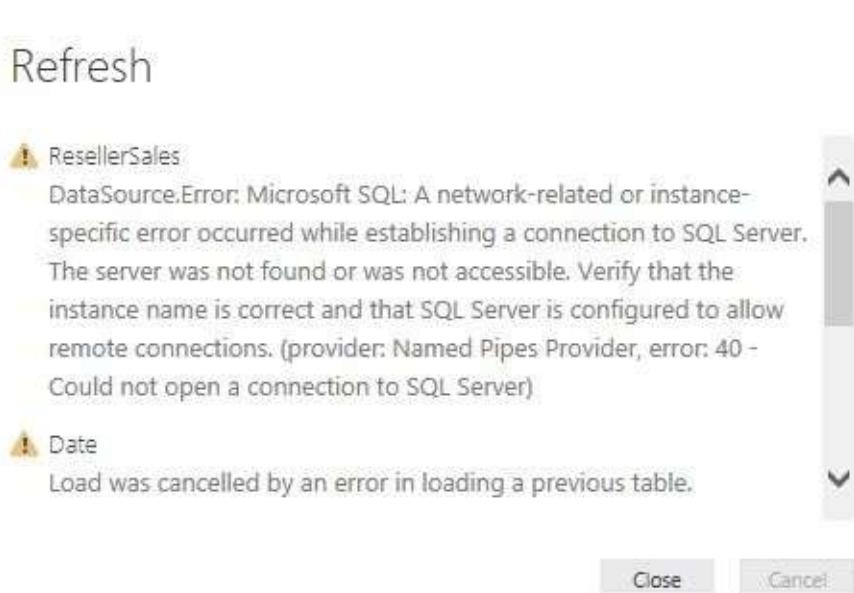


Figure 7.10 If the refresh operation fails, the Refresh window shows which table failed to refresh and shows the error description.

7.3 Relating Tables

One of the most prominent Power BI strengths is that it can help an analyst analyze data across multiple tables. Back in Chapter 5, I covered that as a prerequisite for aggregating data in one table by columns in another table, you must set up a relationship between the two tables. When you import tables from a relational database that supports referential integrity and has table relationships defined, Power BI Desktop detects these relationships and applies them to the model. However, when no table joins are defined in the data source, or when you import data from different sources, Power BI Desktop might be unable to detect relationships upon import. Because of this, you must revisit the model and create appropriate relationships before you analyze the data.

7.3.1 Relationship Rules and Limitations

A relationship is a join between two tables. When you define a table relationship with one-to-many cardinality, you're telling Power BI that there's a logical one-to-many relationship between a row in the lookup (dimension) table and the corresponding rows in the fact table. For example, the relationship between the Reseller and ResellerSales tables in **Figure 7.11** means that each reseller in the Reseller table can have many corresponding rows in the ResellerSales table. Indeed, Progressive Sports (ResellerKey=1) recorded a sale on August 1st, 2006 for \$100 and another sale on July 4th 2007 for \$120. In this case, the ResellerKey column in the Reseller table is the primary key in the lookup (dimension) table. The ResellerKey column in the ResellerSales table fulfills the role of a foreign key in the fact table.

Reseller Table		
ResellerKey	ResellerName	BusinessType
1	Progressive Sports	Warehouse
2	Bikes and Motorbikes	Catalog Store
3	Exotic Bikes	Warehouse
4	Every Bike Shop	Value Added Reseller

ResellerSales Table		
ResellerKey	OrderDateKey	SalesAmount
1	20060801	\$100.00
1	20070704	\$200.00
3	20060801	\$120
4	20060801	\$56

Figure 7.11 There's a logical one-to-many relationship between the Reseller table and the ResellerSales table because each reseller can have multiple sales recorded.

Understanding relationship rules

A relationship can be created under the following circumstances:

- The two tables have matching columns, such as a ResellerKey column in the Reseller lookup table and a ResellerKey column in the ResellerSales table. The column names don't have to be the same but the columns must have matching values. For example, you can't relate the two tables if the ResellerKey column in the ResellerSales table has

reseller codes, such as PRO for Progressive Sports.

- The key column in the lookup (dimension) table must have unique values, similar to a primary key in a relational database. In the case of the Reseller table, the ResellerKey column fulfills this requirement because its values are unique across all the rows in the table. However, this doesn't mean that all fact tables must join the lookup table on the same primary key. As long as the column is unique, it can serve as a primary key. And some fact tables can use one column while others can use another column. If you attempt to establish a join to a column that doesn't contain unique values in a lookup table, you'll get the following error:

The relationship cannot be created because each column contains duplicate values. Select at least one column that contains only unique values.

Interestingly, Power BI doesn't require the two columns to have matching data types. For example, the ResellerKey column in the Reseller table can be of a Text data type while its counterpart in the fact table could be defined as the Whole Number data type. Behind the scenes, Power BI resolves the join by converting the values in the latter column to the Text data type. However, to improve performance and to reduce storage space, use numeric data types whenever possible.

Understanding relationship limitations

Relationships have several limitations. To start, only one column can be used on each side of the relationship. If you need a combination of two or more columns (so the key column can have unique values), you can add a custom column in the query or a calculated column that uses a DAX expression to concatenate the values, such as =[ResellerKey] & “|” & [SourceID]. I use the pipe delimiter here to avoid combinations that might result in the same concatenated values. For example, combinations of ResellerKey of 1 with SourceID of 10 and ResellerKey of 11 and SourceID of 0 result in “110”. To make the combinations unique, you can use a delimiter, such as the pipe character. Once you construct a primary key column, you can use this column for the relationship.

Moving down the list, you can't create relationships forming a closed loop (also called a diamond shape). For example, given the relationships Table1 → Table2 and Table2 → Table3, you can't set an active relationship Table1 → Table3. Such a relationship probably isn't needed anyway, because you'll be able to analyze the data in Table3 by Table1 with only the first two relationships in place. Power BI will actually let you create the Table1 → Table3 relationship, but it will mark it as inactive. This brings to the subject of role-playing relationships and inactive relationships.

As it stands, Power BI doesn't support role-playing relationships. A role-playing lookup table is a table that joins the same fact table multiple times, and thus plays multiple roles. For example, the InternetSales table has the OrderDateKey, ShipDateKey, and DueDateKey columns because a sales order has an order date, ship date, and due date. Suppose you want to analyze sales by these three dates. One approach is to import the Date table three times with different names and to create relationships to each date table. This approach gives you more control because you now have three separate Date tables and their data doesn't have to match. For example, you might want the ShipDate table to

include different columns than the OrderDate table. On the downside, you increase your maintenance effort because you know have to maintain three tables.

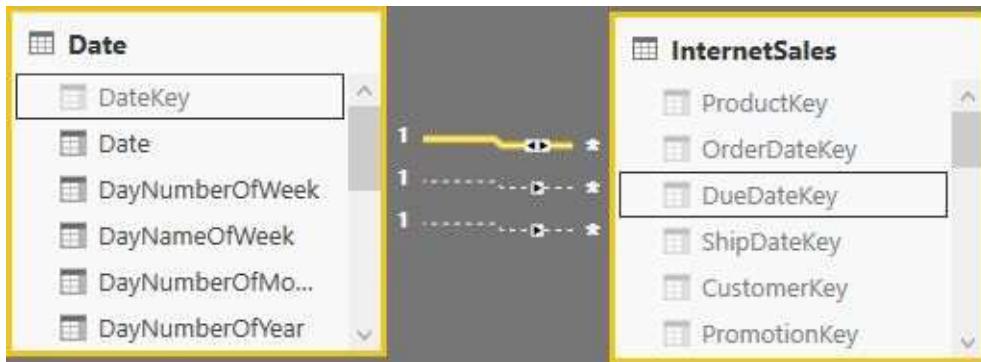


Figure 7.12 Power BI supports only one active relationship between two tables and marks the other relationships as inactive.

REAL WORLD On the subject of date tables, AdventureWorksDW uses a “smart” integer primary key for the Date table in the format YYYYMMDD. This is a common practice for data warehousing, but I personally prefer to use a date field (Date data type). Not only is it more compact (3 bytes vs. 4 bytes for Integer) but it’s also easier to work with. For example, if a business user imports ResellerSales, he can filter easier on a Date data type, such as to import data for the current year, than to parse integer fields. And DAX time calculations work if a date column is used as a primary key. That’s why in the practice exercises that follows, you’ll recreate the relationships to the date table.

Understanding active and inactive relationships

Another approach is to join the three date columns in InternetSales to the Date table. This approach allows you to reuse the same date table three times. However, Power BI supports only one active role-playing relationship. An active relationship is a relationship that Power BI follows to automatically aggregate the data between two tables. A solid line in the Relationships View indicates an active relationship while a dotted line is for inactive relationships (see **Figure 7.12**). You can also open the Manage Relationships window (click the Manage Relationships button in ribbon’s Home or Modeling tabs) and inspect the Active flag.

When Power BI Desktop imports the relationships from the database, it defaults the first one to active and marks the rest as inactive. In our case, the InternetSales[DueDateKey] → DimDate[DateKey] relationship is active because this happens to be the first of the three relationships between the DimDate and FactInternetSales tables that you imported. Consequently, when you create a report that slices Internet dates by Date, Power BI automatically aggregates the sales by the due date.

NOTE I’ll use the TableName[ColumnName] notation as a shortcut when I refer to a table column. For example, InternetSales[DueDateKey] means the DueDateKey column in the InternetSales table. This notation will help you later on with DAX formulas because DAX follows the same syntax. When referencing relationships, I’ll use a right arrow (→) to denote a relationship from a fact table to a lookup table. For example, InternetSales[OrderDateKey] → DimDate[DateKey] means a relationship between the OrderDateKey column in the InternetSales table to the DateKey column in the DimDate table.

If you want the default aggregation to happen by the order date, you must set InternetSales[OrderDateKey] → DimDate[DateKey] as an active relationship. To do so, first select the InternetSales[ShipDateKey] → DimDate[DateKey] relationship, and then click Edit. In the Edit Relationship dialog box, uncheck the Active checkbox, and then click OK. Finally, edit the InternetSales[OrderDateKey] → DimDate[DateKey] relationship, and then check

the Active checkbox.

What if you want to be able to aggregate data by other dates without importing the Date table multiple times? You can create DAX calculated measures, such as ShippedSalesAmount and DueSalesAmount, that force Power BI to use a given inactive relationship by using the DAX USERELATIONSHIP function. For example, the following formula calculates ShippedSalesAmount using the ResellerSales[ShipDateKey] ð DimDate[DateKey] relationship:

```
ShippedSalesAmount=CALCULATE(SUM(InternetSales[SalesAmount]),  
USERELATIONSHIP(InternetSales[ShipDateKey],‘Date’[DateKey]))
```

Cross filtering limitations

In Chapter 5, I mentioned that a relationship can be set to cross-filter in both directions. This is a great out-of-box feature that allows you to address more advanced scenarios that previously required custom calculations with Power Pivot, such as many-to-many relationships. However, bi-directional filtering doesn't make sense and should be avoided in the following cases:

- When you have two fact tables sharing some common dimension tables – In fact, to avoid ambiguous join paths, Power BI Desktop won't let you turn on bi-directional filtering from multiple fact tables to the same lookup table. Therefore, if you start from a single fact table but anticipate additional fact tables down the road, you may also consider a uni-directional model (Cross Filtering set to Single) to keep a consistent experience to users, and then turn on bi-directional filtering only if you need it.

NOTE To understand this limitation better, let's say you have a Product lookup table that has bi-directional relations to ResellerSales and InternetSales tables. If you define a DAX measure on the Product table, such as Count of Products, but have a filter on a Date table, Power BI won't know how to resolve the join: count of products through ResellerSales on that date, or count of products through InternetSales on that date.

- Relationships toward the date table – Relationships to date tables should be one-directional so that DAX time calculations continue to work.
- Closed-loop relationships – As I just mentioned, Power BI Desktop will automatically deactivate one of the relationships when it detects a closed loop, although you can still use DAX calculations to navigate inactive relationships. In this case, bi-directional relationships would produce meaningless results.

7.3.2 Auto-detecting Relationships

When you create a report that uses unrelated tables, Power BI Desktop can auto-detect and create missing relationships. This behavior is enabled by default, but you can disable it by turning it off from the File ð Options and Settings ð Options menu, which brings you to the Options window (see **Figure 7.13**).

Options

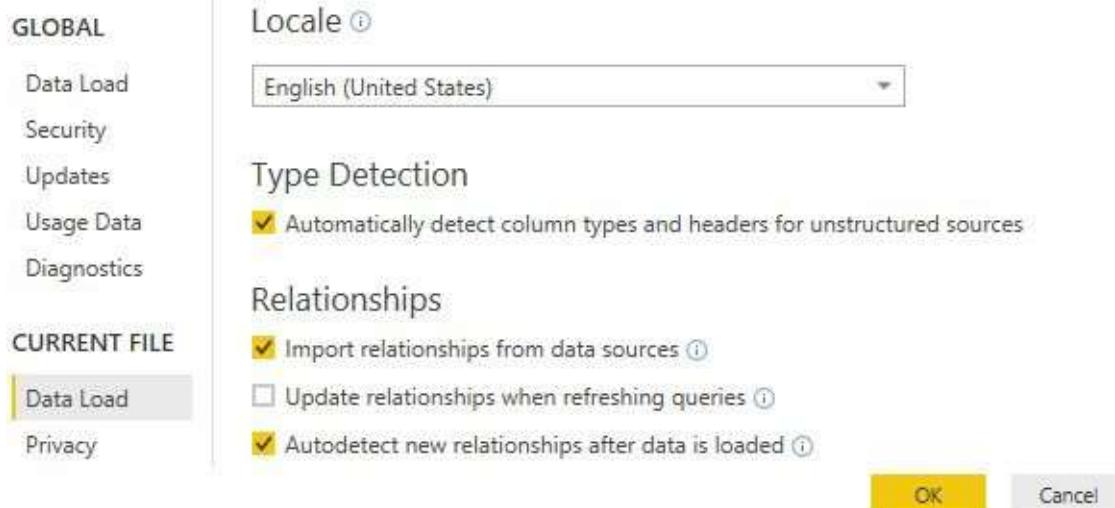


Figure 7.13 You can control how Power BI Desktop discovers relationships.

Configuring relationships detection

There are three options that control how Power BI desktop detects relationships. When checked, the “Import relationships from data sources” option instructs Power BI Desktop to detect relationships from the data source *before* the data is loaded. When this option is enabled, Power BI Desktop will examine the database schema and probe for existing relationships.

The “Update relationships when refreshing queries” option will attempt to discover missing relationships when refreshing the imported data. Because this might result in dropping existing relationships that you’ve created manually, this option is off by default. Finally, “Autodetect new relationships after data is loaded” will attempt to auto-detect missing relationships *after* the data is loaded. Because this option is on by default, Power BI Desktop was able to detect relationships between the InternetSales and Date tables, as well as between other tables. The auto-detection mechanism uses an internal algorithm that considers column data types and cardinality.

Understanding missing relationships

What happens when you don’t have a relationship between two tables and attempt to analyze the data in a report? You’ll get repeating values. In this case, I attempted to aggregate the SalesAmount column from the ResellerSales table by the ProductName column in the Product table, but there’s no relationship defined between these two tables. If reseller sales should aggregate by product, you must define a relationship to resolve this issue.

ProductName	SalesAmount
	\$80,450,596.98
All-Purpose Bike Stand	\$80,450,596.98
AWC Logo Cap	\$80,450,596.98
Bike Wash - Dissolver	\$80,450,596.98
Chain	\$80,450,596.98
Classic Vest, L	\$80,450,596.98
Classic Vest, M	\$80,450,596.98

Figure 7.14 Reports show repeating values in the case of missing relationships.

Autodetecting relationships

The lazy approach to handle missing relationships is to let Power BI Desktop create them by clicking the Autodetect button in the Manage Relationship window. If the internal algorithm detects a suitable relationship candidate, it creates the relationship and informs you, as shown in **Figure 7.15**.

Manage Relationships

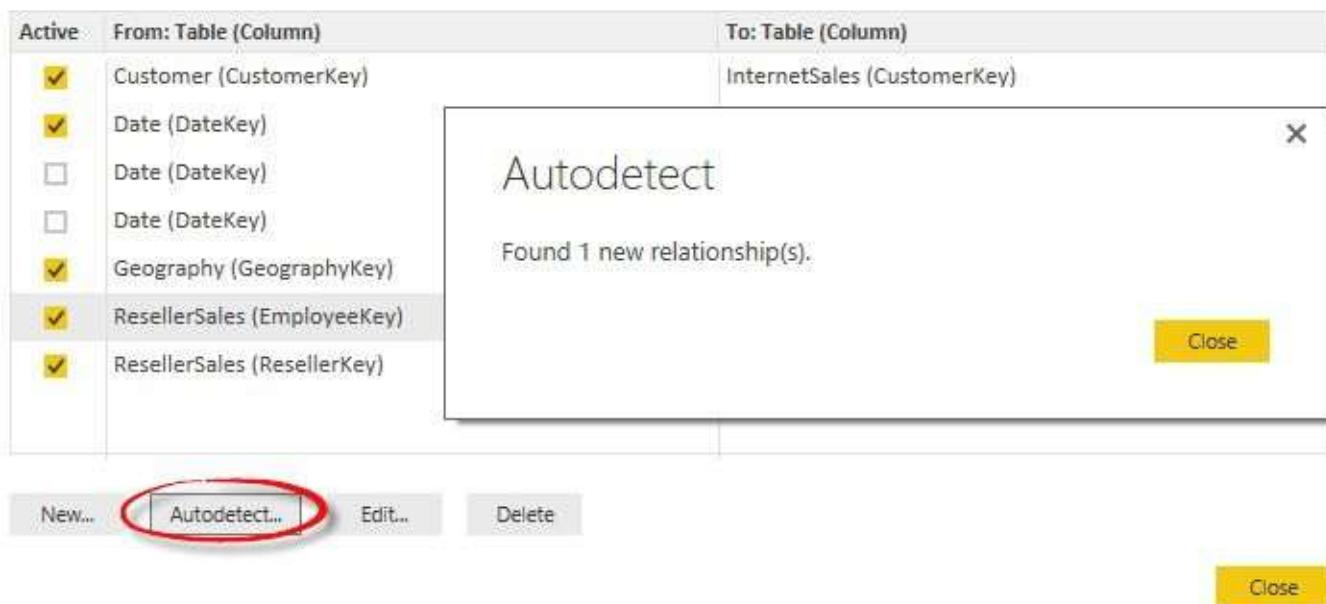


Figure 7.15 The Autodetect feature of the Manage Relationship window shows that it has detected and created a relationship successfully.

In the case of an unsuccessful detection process, the Relationship dialog box will show “Found no new relationships”. If this happens and you’re still missing relationships, you need to create them manually.

7.3.3 Creating Relationships Manually

Since table relationships are very important, I’d recommend that you configure them manually. You can do this by using the Manage Relationships window or by using the Relationships View.

Steps to create a relationship

Follows these steps to set up a relationship:

- 1.Identify a foreign key column in the table on the Many side of the relationship.
- 2.Identify a primary key column that uniquely identifies each row in the lookup (dimension) table.
- 3.In the Manage Relationship window, click the New button to open the Create Relationship window. Then create a new relationship with the correct cardinality. Or you can use the Relationships View to drag the foreign key from the fact table onto the primary key of the lookup table.

Understanding the Create Relationship window

You might prefer the Create Relationship window when the number of tables in your model has grown and using the drag-and-drop technique in the Relationships View becomes impractical. **Figure 7.16** shows the Create Relationship dialog box when setting up a relationship between the ResellerSales and Product tables.

Create Relationship

X

Select tables and columns that relate to one another.

ResellerSales							
ProductKey	OrderDateKey	DueDateKey	ShipDateKey	ResellerKey	EmployeeKey	PromotionKey	UnitPrice
317	20050801	20050813	20050808	403	282	1	10.99
322	20050801	20050813	20050808	403	282	1	10.99
316	20050801	20050813	20050808	403	282	1	10.99
270	20050801	20050813	20050808	403	282	1	10.99
314	20050801	20050813	20050808	403	282	1	10.99

Product				
ProductModel	Description	ProductKey	ProductNumber	ProductName
Hitch Rack - 4-Bike	Carries 4 bikes securely; steel construction, fits 2" receiver.	483	RA-H123	Hitch Rack - 4
All-Purpose Bike Stand	Perfect all-purpose bike stand for working on your bike...	486	ST-1401	All-Purpose E
Mountain Bottle Cage	Tough aluminum cage holds bottle securely on tough ter...	478	BC-M005	Mountain Bo
Road Bottle Cage	Aluminum cage is lighter than our mountain version; pe...	479	BC-R205	Road Bottle C
Water Bottle	AWC logo water bottle - holds 30 oz; leak-proof.	477	WB-H098	Water Bottle

Advanced options

Cardinality

Many to One (*:1)

Cross filter direction

Both

Make this relationship active

OK Cancel

Figure 7.16 Use the Create Relationship window to specify the columns used for the relationship, cardinality and cross filter direction.

When defining a relationship, you need to select two tables and matching columns. The Create Relationships window will detect the cardinality for you. For example, if you start with the table on the many side of the relationship (ResellerSales), it'll choose the Many to One cardinality; otherwise it selects One to Many. If you attempt to set up a relationship with the wrong cardinality, you'll get an error message ("The Cardinality you selected isn't valid for this relationship"), and you won't be able to create the relationship. And if you choose a column that doesn't uniquely identify each row in the lookup table, you'll get the error message "You can't create a relationship between these two columns because one of the columns must have unique values".

Managing relationships

You can view and manage all the relationships defined in your model by using the Manage

Relationships window (see **Figure 7.15** again). In this case, the Manage Relationships window shows that there are seven relationships defined in the Adventure Works model from which two are inactive. The Edit button opens the Edit Relationship window, which is the same as the Create Relationship window, but with all the fields pre-populated. Finally, the Delete button removes the selected relationship.

Understanding unknown members

Consider the model shown in **Figure 7.17**, which has a Reseller lookup table and a Sales fact table. This diagram uses an Excel pivot report to demonstrate unknown members but a Power BI Desktop report will behave the same. The Reseller table has only two resellers. However, the Sales table has data for two additional resellers with keys of 3 and 4. This is a common data integrity issue when the source data originates from heterogeneous data sources and when there isn't an ETL process to validate and clean the data.



Figure 7.17 Power BI enables an unknown member to the lookup table when it encounters missing rows.

Power BI has a simple solution for this predicament. When creating a relationship, Power BI checks for missing rows in the lookup table. If it finds any, it automatically configures the lookup table to include a special unknown member. That's why all unrelated rows appear grouped under a blank row in the report. This row represents the unknown member in the Reseller table.

7.3.4 Understanding the Relationships View

Another way to view and manage relationships is to use the Relationships View. You can use the Relationships View to:

- Visualize the model schema
- Create and manage relationships
- Make other limited schema changes, such as renaming, hiding, deleting objects

One of the strengths of the Relationships View is that you can quickly visualize and understand the model schema and relationships. **Figure 7.18** shows a subset of the Adventure Works model schema open in the Relationships View. Glancing at the model,

you can immediately see what relationships exist in the model!

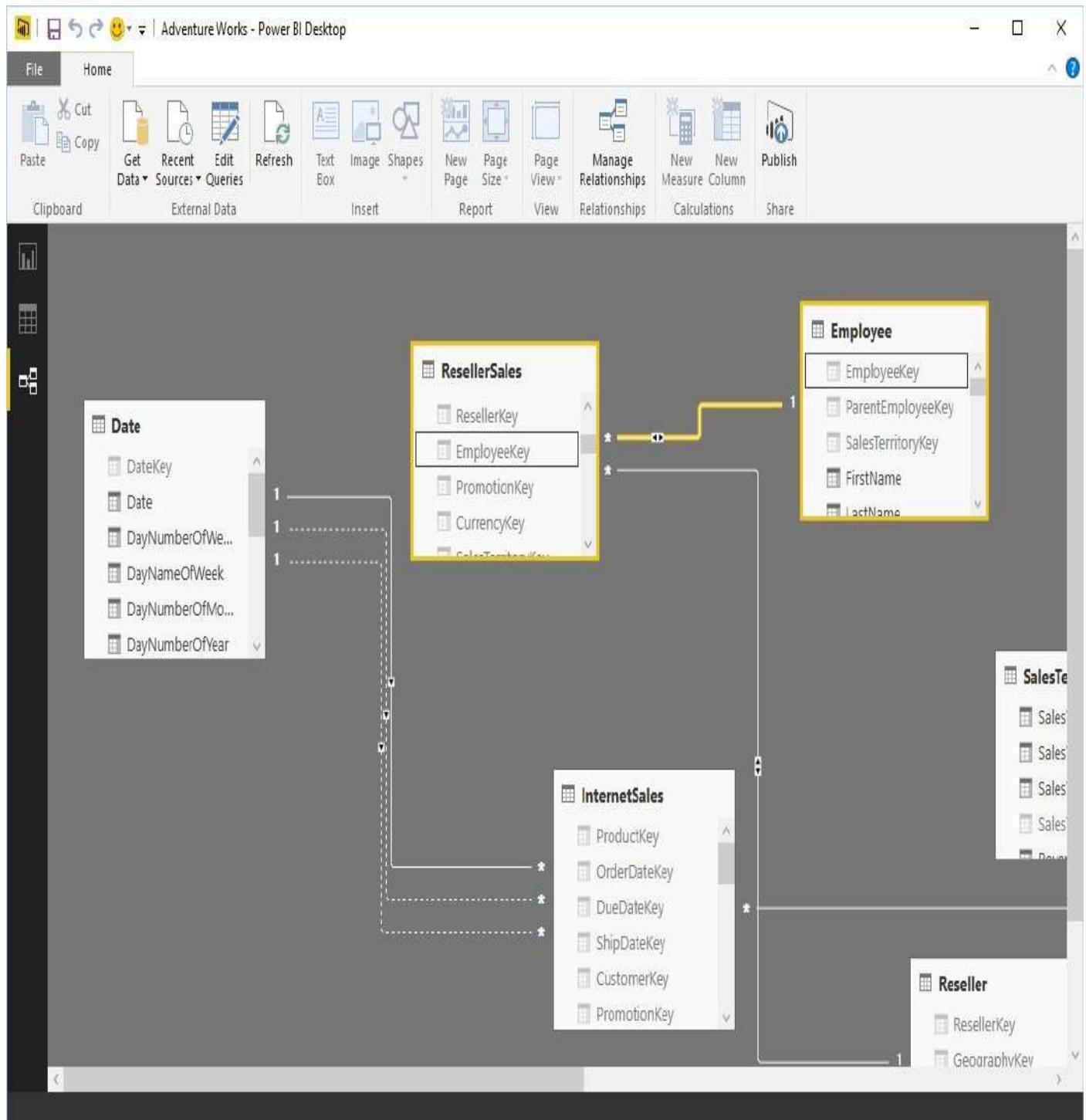


Figure 7.18 The Relationships View helps you understand the model schema and work with relationships.

Making schema changes

You can make limited schema changes in the Relationships View. When you right-click an object, a context menu opens up to show the supported operations, as shown in **Table 7.2**.

Table 7.2 This table shows the schema operations by object type.

Object Type	Supported Operations	Object Type	Supported Operations
Table	Delete, Hide, Rename, Maximize	Measure	Delete, Hide, Rename
Column	Delete, Hide, Rename		
Relationship	Delete		

Managing relationships

Since I'm on the subject of relationships, let's take a closer look at how the Relationships View represents relationships. A relationship is visualized as a connector between two tables. Symbols at the end of the connector help you understand the relationship cardinality. The number one (1) denotes that the table on the One side of the relationship, while the asterisk (*) is shown next to the table on the Many side of the relationship. For example, after examining **Figure 7.18**, you can see that there's a relationship between the Employee table and ResellerSales table and that the relationship cardinality is One to Many with the Employee table on the One side of the relationship and the ResellerSales table on the many.

When you click a relationship to select it, the Relationships View highlights it in an orange color. When you hover your mouse over a relationship, the Relationships View highlights columns in the joined tables to indicate visually which columns are used in the relationship. For example, pointing the mouse to the highlighted relationship between the ResellerSales and Employee tables reveals that the relationship is created between the ResellerSales[EmployeeKey] column and the Employee[EmployeeKey] column (see **Figure 7.18** again).

As I mentioned, Power BI has a limited support of role-playing relationships where a lookup table joins multiple times to a fact table. The caveat is that only one role-playing relationship can be active. The Relationships View shows the inactive relationships with dotted lines. To make another role-playing relationship active, first you need to deactivate the currently active relationship. To do so, double-click the active relationship, and then in the Edit Relationship window uncheck the “Make this relationship active” checkbox. Next, you double-click the other role-playing relationship and then check its “Make this relationship active” checkbox.

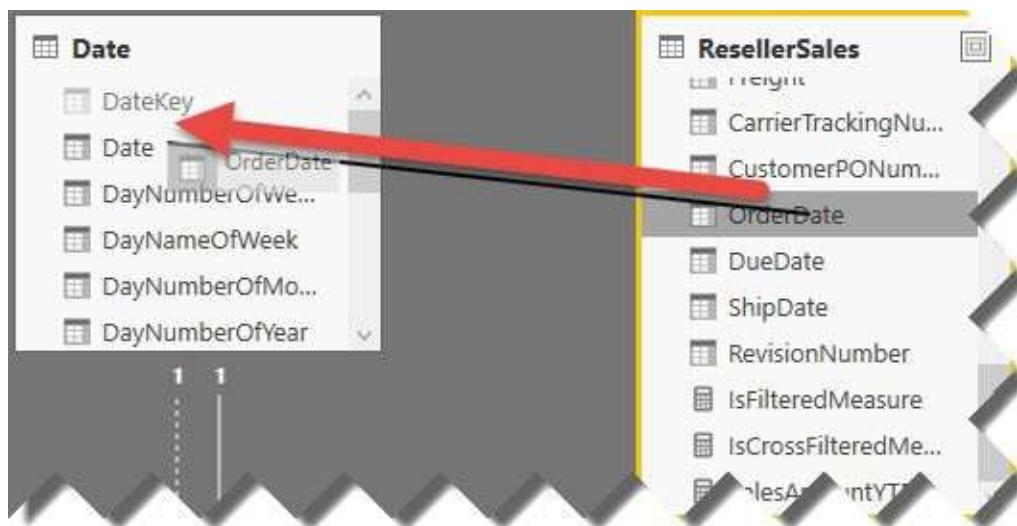


Figure 7.19 The Relationships View lets you create a relationship by dragging a foreign key column onto a primary key column.

A great feature of Relationships View is creating relationships by dragging a foreign key column and dropping it onto the primary key column in the lookup table. For example, to create a relationship between the ResellerSales and Date tables, drag the OrderDate

column in the ResellerSales table and drop it onto the Date column in the Date table. The Relationships View won't allow you to create a relationship in the reverse direction. To delete a relationship, simply click the relationship to select it, and then press the Delete key. Or right-click the relationship line, and then click Delete.

7.3.5 Working with Relationships

As it stands, the Adventure Works model has nine tables and seven auto-detected relationships. There are many tables that aren't related. Next, you'll set up some more relationships using different techniques.

Auto-detecting relationships

Let's see how far you can get by letting Power BI Desktop auto-detect relationships:

- 1.In the Data View (or Reports View), click the Manage Relationships button in the ribbon's Home tab to open the Manage Relationships window.
- 2.Click the Autodetect button. If there's no relationship between the Customer and Geography tables, the auto-detection finds one new relationship. Click OK. Notice that a relationship Reseller[GeographyKey] ð Geography[GeographyKey] is added. This relationship reflects the underlying snowflake schema consisting of the ResellerSales, Reseller, and Geography tables. The new relationship allows us to analyze reseller sales by geography using a cascading relationship spanning three tables (ResellerSales ð Reseller ð Geography).

NOTE Usually a byproduct of a snowflake schema, a cascading relationship joins a fact table to a lookup table via another referenced lookup table. Power BI supports cascading relationships that can traverse multiple lookup tables.

Now let's clean up some existing relationships. As it stands, the InternetSales table has three relationships to the Date table (one active and two inactive) which Power BI Desktop auto-discovered from the AdventureWorksDW database. All these relationships join the Date table on the DateKey column. As I mentioned before, I suggest you use a column of a Date data type in the Date table. Luckily, both the Reseller Sales and InternetSales tables have OrderDate, ShipDate, and DueDate date columns. And the Date table has a Date column which is of a Date data type.

- 3.Delete the two inactive relationships (the ones with an unchecked Active flag) from the InternetSales table to the Date table. If it exists, delete also the relationship ResellerSales[DueDateKey] ð Date[DateKey].

Creating relationships using the Manage Relationships window

The Adventure Works model has two fact tables (ResellerSales and InternetSales) and seven lookup tables. Let's start creating the missing relationships using the Manage Relationships window:

TIP When you have multiple fact tables, join them to common dimension tables. This allows you to create consolidated reports that includes multiple subject areas, such as a report that shows Internet sales and reseller sales side by side grouped by date and sales territory.

- 1.First, let's rebind the InternetSales[OrderDateKey] ð Date[DateKey] relationship to use another set of columns. In the Manage Relationship window, double-click the

InternetSales[OrderDateKey] ð Date[DateKey] relationship (or select it and click Edit). In the Edit Relationship window, select the OrderDate column (scroll all the way to the right) in the InternetSales table. Then select the Date column in the Date table and click OK.

NOTE When joining fact tables to a date table on a date column, make sure that the foreign key values contain only the date portion of the date and not the time portion. Otherwise, the join will never find matching values in the date table. If you don't need it, the easiest way to discard the time portion is to change the column data type from Date/time to Date. You can also apply query transformations to strip the time portion or to create custom columns that have only the date portion.

2. Back in the Manage Relationship window, click New. Create a relationship ResellerSales[OrderDate] ð Date[Date]. If the new relationship ends up being inactive (the Active column is unchecked) and you attempt to activate it by checking the Active flag, you might get the error shown in **Figure 7.20**.

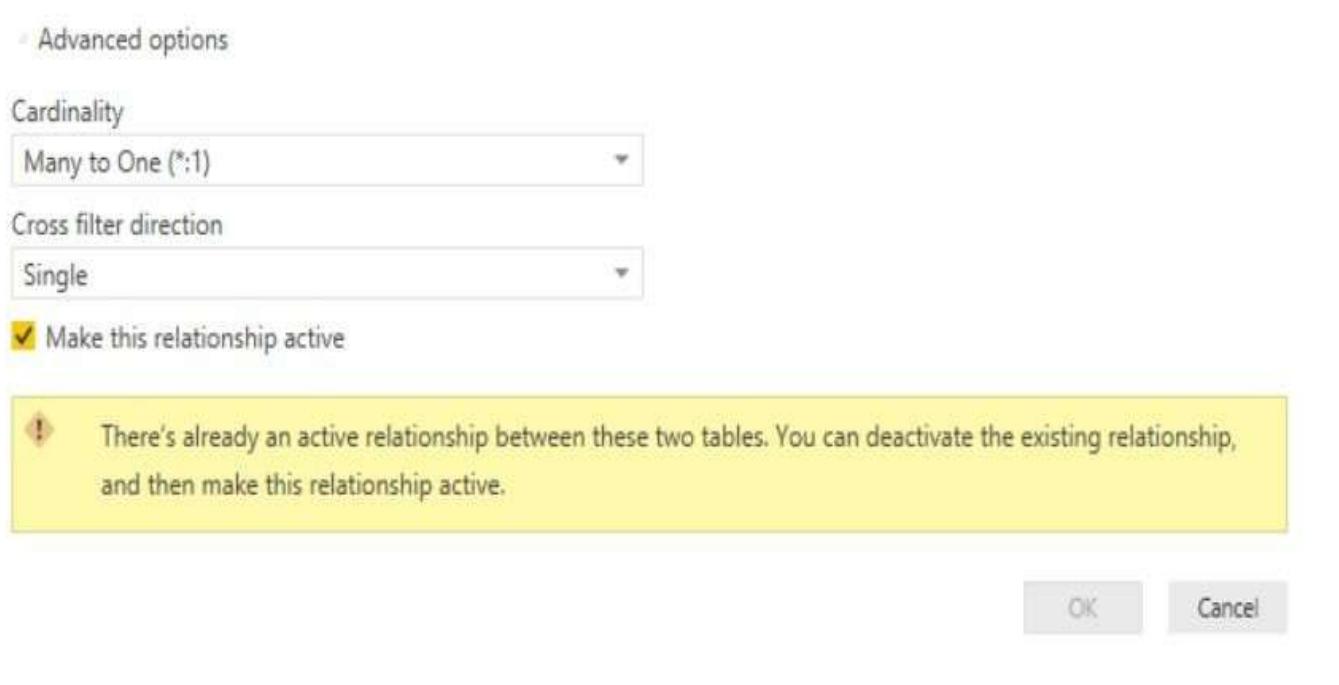


Figure 7.20 This error is caused by bi-directional relationships to the Geography lookup table.

This error message is misleading because there isn't a direct relationship between ResellerSales and Date. The problem is that during the auto-discovery process, Power BI Desktop might have created bi-directional relationships from the InternetSales and ResellerSales tables to the Geography lookup table. As I mentioned before, this causes an issue because it leads to ambiguous paths.

NOTE Based on the feedback I got from Microsoft, this issue is caused by a bug in the current relationship auto-discovery process, which should be fixed in a subsequent release. The correct behavior should have been to turn on uni-directional relationships first and make them active before trying bi-directional filtering. Also, relationships to a date table should always be uni-directional. Still, when you create relationships manually, you need to pay extra attention. You may need to demote some existing bi-directional relationships to uni-directional prior to creating a new relationship so that the new relationship doesn't end up in an inactive state.

3. If this issue happens, there's how to fix it. In the Manage Relationships window, edit the ResellerSales[ResellerKey] ð Reseller[ResellerKey], Reseller[GeographyKey] ð Geography[GeographyKey], InternetSales[CustomerKey] ð Customer[CustomerKey] and

Customer[GeographyKey] → Geography[GeographyKey] relationships one by one and change their “Cross filtering direction” property to Single. Then activate the ResellerSales[OrderDate] → Date[Date] relationship.

- 4.Create ResellerSales[SalesTerritoryKey] → SalesTerritory[SalesTerritoryKey] and ResellerSales[ProductKey] → Product[ProductKey] relationships.

Creating relationships using the Relationships View

Next, you’ll use the Relationships View to create relationships for the InternetSales table.

1. Click the Relationships View icon in the navigation bar.
- 1.Drag the InternetSales[ProductKey] column and drop it onto the Product[ProductKey] column.
- 2.Drag the InternetSales[SalesTerritoryKey] column and drop it onto the SalesTerritory[SalesTerritoryKey] column.
- 3.Click the Manage Relationships button. Compare your results with **Figure 7.21**. As it stands, the Adventure Works model has 11 relationships. For now, let’s not create inactive relationships. I’ll revisit them in the next chapter when I cover DAX.

The screenshot shows the 'Manage Relationships' dialog box. At the top, there's a title bar with a close button (X) on the right. Below the title bar, the main area is a table with two columns: 'From: Table (Column)' and 'To: Table (Column)'. The 'From' column lists various columns from different tables, and the 'To' column lists the corresponding columns they relate to. All 11 relationships listed are currently active, indicated by checked checkboxes in the 'Active' column. The relationships are:

Active	From: Table (Column)	To: Table (Column)
<input checked="" type="checkbox"/>	Customer (GeographyKey)	Geography (GeographyKey)
<input checked="" type="checkbox"/>	InternetSales (CustomerKey)	Customer (CustomerKey)
<input checked="" type="checkbox"/>	InternetSales (OrderDate)	Date (Date)
<input checked="" type="checkbox"/>	InternetSales (ProductKey)	Product (ProductKey)
<input checked="" type="checkbox"/>	InternetSales (SalesTerritoryKey)	SalesTerritory (SalesTerritoryKey)
<input checked="" type="checkbox"/>	Reseller (GeographyKey)	Geography (GeographyKey)
<input checked="" type="checkbox"/>	ResellerSales (EmployeeKey)	Employee (EmployeeKey)
<input checked="" type="checkbox"/>	ResellerSales (OrderDate)	Date (Date)
<input checked="" type="checkbox"/>	ResellerSales (ProductKey)	Product (ProductKey)
<input checked="" type="checkbox"/>	ResellerSales (ResellerKey)	Reseller (ResellerKey)
<input checked="" type="checkbox"/>	ResellerSales (SalesTerritoryKey)	SalesTerritory (SalesTerritoryKey)

At the bottom of the dialog box, there are several buttons: 'New...', 'Autodetect...', 'Edit...', 'Delete', and a yellow 'Close' button.

Figure 7.21 The Manage Relationships dialog box shows 11 relationships defined in the Adventure Works model.

- 4.If there are differences between your relationships and **Figure 7.21**, make the necessary changes. Don’t be afraid to delete wrong relationships if you have to recreate them to use

different columns.

5. Once your setup matches **Figure 7.21**, click Close to close the Manage Relationships window. Save the Adventure Works file.

7.4 Summary

Once you import the initial set of tables, you should spend time exploring the model data and refining the model schema. The Data View supports various column operations to help you explore the model data and to make the necessary changes. You should make your model more intuitive by having meaningful table and column names. Revisit each column and configure its data type and formatting properties.

Relationships are the cornerstone of self-service data modelling that involves multiple datasets. You must set up table relationships in order to integrate data across multiple tables. In some cases, Power BI Desktop might be capable of auto-detecting missing relationships or importing them from the data source. However, as a best practice, you should set up relationships manually by using the Manage Relationship window or by using the drag-and-drop technique in the Relationships View.

You've come a long way in designing the Adventure Works model! Next, let's make it even more useful by extending it with business calculations.

Chapter 8

Implementing Calculations

Power BI promotes rapid personal business intelligence (BI) for essential data exploration and analysis. Chances are, however, that in real life you might need to go beyond just simple aggregations. Business needs might require you to extend your model with calculations. Data Analysis Expressions (DAX) gives you the needed programmatic power to travel the “last mile” and unlock the full potential of Power BI.

DAX is a big topic that deserves much more attention, and this chapter doesn’t aim to cover it in depth. However, it’ll lay down the necessary fundamentals so that you can start using DAX to extend your models with business logic. The chapter starts by introducing you to DAX and its arsenal of functions. Next, you’ll learn how to implement custom calculated columns and measures. I’ll also show you how to handle more advanced scenarios with DAX. And you’ll create various visualizations to test the sample calculations.

8.1 Understanding Data Analysis Expressions

Data Analysis Expressions (DAX) is a formula-based language in Power BI, Power Pivot, and Tabular that allows you to define custom calculations using an Excel-like formula language. DAX was introduced in the first version of Power Pivot (released in May 2010) with two major design goals:

- Simplicity – To get you started quickly with implementing business logic, DAX uses the Excel standard formula syntax and inherits many Excel functions. As a business analyst, Martin already knows many Excel functions, such as SUM and AVERAGE. When he uses Power BI, he appreciates that DAX has the same functions.
- Relational – DAX is designed with data models in mind and supports relational artifacts, including tables, columns, and relationships. For example, if Martin wants to sum the SalesAmount column in the ResellerSales table, he can use the following formula:
=SUM(ResellerSales[SalesAmount]).

DAX also has query constructs to allow external clients to query organizational Tabular models. As a data analyst, you probably don't need to know about these constructs. This chapter focuses on DAX as an expression language to extend self-service data models. If you need to know more about DAX in the context of organizational BI, you might find my book "Applied Microsoft SQL Server 2012 Analysis Services: Tabular Modeling" useful.

You can use DAX as an expression language to implement custom calculations that range from simple expressions, such as to concatenate two columns together, to complex measures that aggregate data in a specific way, such as to implement weighted averages. Based on the intended use, DAX supports two types of calculations: calculated columns and measures.

8.1.1 Understanding Calculated Columns

A calculated column is a table column that uses a DAX formula to compute the column values. This is conceptually similar to a formula-based column added to an Excel list, although DAX formulas reference columns instead of cells.

How calculated columns are stored

When a column contains a formula, the storage engine computes the value for each row and saves the results, just like it does with a regular column. To use a techie term, values of calculated columns get "materialized" or "persisted". The difference is that regular columns import their values from a data source, while calculated columns are computed from DAX formulas and saved after the regular columns are loaded. Because of this, the formula of a calculated column can reference regular columns and other calculated columns.

Unlike regular columns, however, the storage engine doesn't compress calculated columns. This is important because if you have a large table with millions of rows and a calculated column that has many unique values, this column might have a large memory footprint.

Understanding row context

Every DAX formula is evaluated in a specific context. The formulas of calculated columns are evaluated for each row (row context). Let's look at a calculated column called FullName that's added to the Customer table, and it uses the following formula to concatenate the customer's first name and last name:

```
FullName=[FirstName] & " " & [LastName]
```

Because its formula is evaluated for each row in the Customer table (see **Figure 8.1**), the FullName column returns the full name for each customer. Again, this is very similar to how an Excel formula works when applied to multiple rows in a list, so this should be easy to understand.

Title	FirstName	MiddleName	LastName	FullName
Larry		Gill		Larry Gill
Geoffrey			Gonzalez	Geoffrey Gonzalez
Blake				Blake
Alexa				Alexa
Jacquelyn				Jacquelyn
Casey		Gutierrez		Casey Gutierrez
Colleen		Lu		Colleen Lu
Jeremiah			Stewart	Jeremiah Stewart

Figure 8.1 Calculated columns operate in row context, and their formulas are evaluated for each table row.

In terms of reporting, you can use calculated columns to group and filter data, just like you can use regular columns. For example, you can add a calculated column to any area of the Visualizations pane.

When to use calculated columns

In general, use a calculated column when you need a formula-based column to use on a report or to use in related calculations. Because DAX formulas can reference other tables, a good usage scenario might be to look up a value from another table, just like you can use Excel VLOOKUP to reference values from another sheet. For example, to calculate a profit for each line item in ResellerSales, you might need to look up the product cost from the Product table. In this case, using a calculated column might make sense because its results are stored for each row in ResellerSales.

When shouldn't you use calculated columns? I mentioned that because calculated columns don't compress, they require more storage than regular columns. Therefore, if you can perform the calculation at the data source or in the model query, I recommend you do it there instead of using calculated columns. This is especially true for high-cardinality calculated columns in large tables because they require more memory for storage. For example, you might need to concatenate a carrier tracking number from its distinct parts in a large fact table. It's better to do so in the data source or in the table query before the data is imported. Continuing this line of thought, the example that I gave for using a calculated

column for the customer's full name should probably be avoided in real life because you can perform the concatenation in the query.

Sometimes, however, you don't have a choice, such as when you need to reference a column from a table imported from another source. Or you might need a more complicated calculation that can be done only in DAX, such as to calculate the rank for each customer based on sales history. In these cases, you can't easily apply the calculation at the data source or the query. These are good scenarios for using calculated columns.

8.1.2 Understanding Measures

Besides calculated columns, you can use DAX formulas to define measures. Measures are typically used to produce aggregated values, such as to summarize a SalesAmount column or to calculate a distinct count of customers who have placed orders. Although measures are associated with a table, they don't show in the Data View's data preview pane, as calculated columns do. Instead, they're accessible in the Fields pane. When used on reports, measures are typically added to the Value area of the Visualizations pane.

Understanding measure types

Power BI Desktop supports two types of measures:

- Implicit measures – To get you started as quickly as possible with data analysis, Microsoft felt that you shouldn't have to write formulas for basic aggregations. Any field added to the Value area of the Visualizations pane is treated as an implicit measure and is automatically aggregated, based on the column data type. For example, numeric fields are summed while text fields are counted.
- Explicit measures – You'll create explicit measures when you need an aggregation behavior that goes beyond the standard aggregation functions. For example, you might need a year-to-date (YTD) calculation. Explicit measures are measures that have a custom DAX formula you specify.

Table 8.1 summarizes the differences between implicit and explicit measures.

Table 8.1 Comparing implicit and explicit measures.

Criterion	Implicit Measures	Explicit Measures
Design	Automatically generated	Manually created
Accessibility	Can be changed on the report	Become a part of the model
DAX support	Standard aggregation formulas only	Can use DAX functions

Implicit measures are automatically generated by Power BI Desktop when you add a field to the Value area of the Visualizations pane. By contrast, you must specify a custom formula for explicit measures. Once the implicit measure is created, you can use the Fields list to change its aggregation function. By contrast, explicit measures become a part of the model, and they can't be changed on the report. Implicit measures can only use the DAX standard aggregation functions: Sum, Count, Min, Max, Average, DistinctCount, Standard Deviation, Variance, and Median. By contrast, explicit measures can use any DAX function, such as to define a custom aggregation behavior.

Understanding filter context

Unlike calculated columns, DAX measures are evaluated *at run time* for each report *cell* as opposed to once for each table row. DAX measures are always dynamic and the result of the measure formula is never saved. Moreover, measures are evaluated in the filter context of each cell, as shown in **Figure 8.2**.

The screenshot shows a Power BI report with a table. The table has columns for SalesTerritoryCountry, Date[CalendarYear], and Total SalesAmount. A specific cell in the table is highlighted with a red oval, containing the value '\$722,502.00'. This cell is part of a row for Germany in 2008. A callout bubble points to this cell from the formula 'Sum(ResellerSales[SalesAmount])'. To the right of the table is a 'ProductCategory' filter pane. The 'Bikes' category is selected, indicated by a checked checkbox. Other categories listed are (All), Accessories, Clothing, and Components.

SalesTerritoryCountry	2007	2008	Total
Australia	\$43,174.77	\$1,323,820.73	
Canada	\$9,709.62	\$4,370,334.95	
France	\$1,341,320.07	\$1,111,858.69	\$2,453,178.76
Germany	\$820,513.65	\$722,502.00	\$1,543,015.65
United Kingdom	\$1,230,915.70	\$1,060,662.55	\$2,291,578.25
United States	\$8,933,163.90	\$7,951,335.55	\$16,884,499.45
Total	\$15,467,184.61	\$13,399,243.18	\$28,866,427.79

Figure 8.2 Measures are evaluated for each cell, and they operate in filter context.

This report summarizes the SalesAmount measure by countries on rows and by years on columns. The report is further filtered to show only sales for the Bikes product category. The filter context of the highlighted cell is the Germany value of the SalesTerritory[SalesTerritoryCountry] fields (on rows), the 2008 value of the Date[CalendarYear] field (on columns), and the Bikes value of the Product[ProductCategory] field (used as a filter).

If you're familiar with the SQL language, you can think of the DAX filter context as a WHERE clause that's determined dynamically and then applied to each cell on the report. When Power BI calculates the expression for that cell, it scopes the formula accordingly, such as to sum the sales amount from the rows in the ResellerSales table where the SalesTerritoryCountry value is Germany, the CalendarYear value is 2008, and the ProductCategory value is Bikes.

When to use measures

In general, measures are most frequently used to aggregate data. Explicit measures are typically used when you need a custom aggregation behavior, such as for time calculations, aggregates over aggregates, and weighted averages. Suppose you want to calculate year-to-date (YTD) of reseller sales. As a first attempt, you might decide to add a SalesAmountYTD calculated column to the ResellerSales table. But now you have an issue because each row in this table represents an order line item. It's meaningless to calculate YTD for each line item.

As a second attempt, you could create a summary table in the database that stores YTD sales at a specific grain, such as product, end of month, reseller, and so on. While this might be best for report performance, it presents issues. What if you need to lower the grain to include other dimensions? What if your requirements change and now YTD needs to be calculated as of any date? A better approach would be to use an explicit measure that's evaluated dynamically as users slice and dice the data. And don't worry too much about performance. Thanks to the memory-resident nature of the storage engine, most

DAX calculations are instantaneous!

NOTE The performance of DAX measures depends on several factors, including the complexity of the formula, your knowledge of DAX (whether you write inefficient DAX), the amount of data, and the hardware of your computer. While most measures, such as time calculations and basic filtered aggregations, should perform very well, more involved calculations, such as aggregates over aggregates or the number of open orders as of any reporting date, are more intensive.

8.1.3 Understanding DAX Syntax

As I mentioned, one of the DAX design goals is to look and feel like the Excel formula language. Because of this, the DAX syntax resembles the Excel formula syntax. The DAX formula syntax is case-insensitive. For example, the following two expressions are both valid:

=YEAR([Date])

=year([date])

That said, I suggest you have a naming convention and stick to it. I personally prefer the first example where the function names are in uppercase case and the column references match the column names in the model. This convention helps me quickly identify functions and columns in DAX formulas, and so that's what I use in this book.

Understanding expression syntax

A DAX formula for calculated columns and explicit measures has the following syntax:

Name=expression

Name is the name of the calculated column or measure. The expression must evaluate to a scalar (single) value. Expressions can contain operators, constants, or column references to return literal or Boolean values. The FullName calculated column that you saw before is an example of a simple expression that concatenates two values. You can add as many spaces as you want to make the formula easier to read.

Expressions can also include functions to perform more complicated operations, such as aggregating data. For example, back in **Figure 8.2**, the DAX formula references the SUM function to aggregate the SalesAmount column in the ResellerSales table. Functions can be nested. For example, the following formula nests the FILTER function to calculate the count of line items associated with the Progressive Sports reseller:

=COUNTRROWS(FILTER(ResellerSales, RELATED(Reseller[ResellerName])="Progressive Sports"))

DAX supports up to 64 levels of function nesting, but going beyond two or three levels makes the formulas more difficult to understand. When you need to go above two or three levels of nesting, I recommend you break the formula into multiple measures. This also simplifies testing complex formulas.

Understanding operators

DAX supports a set of common operators to support more complex formulas, as shown in **Table 8.2**. DAX also supports TRUE and FALSE as logical constants.

Table 8.2 DAX supports the following operators.

Category	Operators	Description	Example
----------	-----------	-------------	---------

Arithmetic	$+, -, *, /, ^$	Addition, subtraction, multiplication, division, and exponentiation	$=[\text{SalesAmount}] * [\text{OrderQty}]$
Comparison	$>, >=, <, <=, <>$	For comparing values	$=\text{FILTER}(\text{RELATEDTABLE}(\text{Products}), \text{Products}[\text{UnitPrice}]>30))$
Logical	$\ \, \&\&$	Logical OR and AND	$=\text{FILTER}(\text{RELATEDTABLE}(\text{Products}), \text{Products}[\text{UnitPrice}]>30 \&\& \text{Products}[\text{Discontinued}]=\text{TRUE}())$
Concatenation	$\&$	Concatenating text	$=[\text{FirstName}] \& " " \& [\text{LastName}]$
Unary	$+, -, \text{NOT}$	Change the operand sign	$= - [\text{SalesAmount}]$

Referencing columns

One of DAX's strengths over regular Excel formulas is that it can traverse table relationships and reference columns. This is much simpler and more efficient than referencing Excel cells and ranges with the VLOOKUP function. Column names are unique within a table. You can reference a column using its fully qualified name in the format <TableName>[<ColumnName>], such as in this example:

ResellerSales[SalesAmount]

If the table name includes a space or is a reserved word, such as Date, enclose it with single quotes:

'Reseller Sales'[SalesAmount] or 'Date'[CalendarYear]

When a calculated column references a column from the same table, you can omit the table name. The AutoComplete feature in the formula bar helps you avoid syntax errors when referencing columns. As **Figure 8.3** shows, the moment you start typing the fully qualified column reference in the formula bar, it displays a drop-down list of matching columns. The formula bar also supports color coding, and it colors the function names in a blue color.

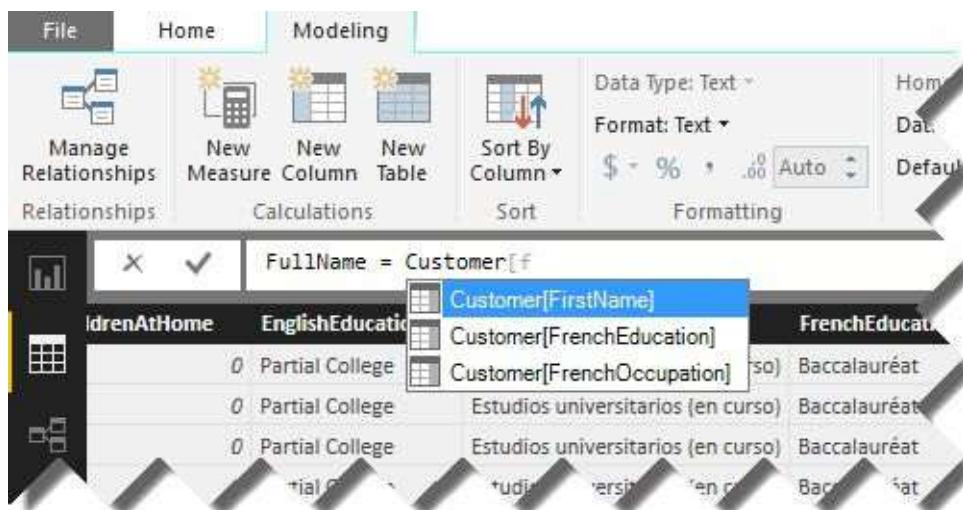


Figure 8.3 AutoComplete helps you with column references in the formula bar.

8.1.4 Understanding DAX Functions

DAX supports over a hundred functions that encapsulate a prepackaged programming logic to perform a wide variety of operations. If you type in the function name in the formula bar, AutoComplete shows the function syntax and its arguments. For the sake of

brevity, this book doesn't cover the DAX functions and their syntax in detail. For more information, please refer to the DAX language reference by Ed Price (this book's technical editor) at <http://bit.ly/daxfunctions>, which provides a detailed description and example for most functions. Another useful resource is "DAX in the BI Tabular Model Whitepaper and Samples" by Microsoft (<http://bit.ly/daxwhitepaper>).

Functions from Excel

DAX supports approximately 80 Excel functions. The big difference is that DAX formulas can't reference Excel cells or ranges. References such as A1 or A1:A10, which are valid in Excel formulas, can't be used in DAX functions. Instead, when data operations are required, the DAX functions must reference columns or tables. **Table 8.3** shows the subset of Excel functions supported by DAX with examples.

Table 8.3 DAX borrows many functions from Excel.

Category	Functions	Example
Date and Time	DATE, DATEVALUE, DAY, EDATE, EOMONTH, HOUR, MINUTE, MONTH, NOW, SECOND, TIME, TIMEVALUE, TODAY, WEEKDAY, WEEKNUM, YEAR, YEARFRAC	=YEAR('Date'[Date])
Information	ISBLANK, ISERROR, ISLOGICAL, ISNONTEXT, ISNUMBER, ISTEXT	=IF(ISBLANK('Date'[Month]), "N/A", 'Date'[Month])
Logical	AND, IF, NOT, OR, FALSE, TRUE	=IF(ISBLANK(Customers[MiddleName]),FALSE(),TRUE())
Math and Trigonometry	ABS, CEILING, ISO.CEILING, EXP, FACT, FLOOR, INT, LN, LOG, LOG10, MOD, MROUND, PI, POWER, QUOTIENT, RAND, RANDBETWEEN, ROUND, ROUNDDOWN, ROUNDUP, SIGN, SQRT, SUM, SUMSQ, TRUNC	=SUM(ResellerSales[SalesAmount])
Statistical	AVERAGE, AVERAGEA, COUNT, COUNTA, COUNTBLANK, MAX, MAXA, MIN, MINA	=AVERAGE(ResellerSales[SalesAmount])
Text	CONCATENATE, EXACT, FIND, FIXED, LEFT, LEN, LOWER, MID, REPLACE, REPT, RIGHT, SEARCH, SUBSTITUTE, TRIM, UPPER, VALUE	=SUBSTITUTE(Customer[Phone], "-", "")

Aggregation functions

As you've seen, DAX "borrows" the Excel aggregation functions, such as SUM, MIN, MAX, COUNT, and so on. However, the DAX counterparts accept a table column as an input argument instead of a cell range. Since only referencing columns can be somewhat limiting, DAX adds X-version of these functions: SUMX, AVERAGEX, COUNTAX, MINX, MAXX. These functions take two arguments. The first one is a table and the second is an expression.

Suppose you want to calculate the total order amount for each row in the ResellerSales table using the formula [SalesAmount] * [OrderQuantity]. You can accomplish this in two ways. First, you can add an OrderAmount calculated column that uses the above expression and then use the SUM function to summarize the calculated column. However, a better approach is to perform the calculation in one step by using the SUMX function, as follows:

```
=SUMX(ResellerSales, ResellerSales[SalesAmount] * ResellerSales[OrderQuantity])
```

Although the result in both cases is the same, the calculation process is very different. In the case of the SUM function, DAX simply aggregates the column. When you use the

SUMX function, DAX will compute the expression for each of the detail rows behind the cell and then aggregate the result. What makes the X-version functions flexible is that the table argument can also be a function that returns a table of values. For example, the following formula calculates the simple average (arithmetic mean) of the SalesAmount column for rows in the InternetSales table whose unit price is above \$100:

```
=AVERAGEX(FILTER(InternetSales, InternetSales[UnitPrice] > 100), InternetSales[SalesAmount])
```

This formula uses the FILTER function, which returns a table of rows matching the criteria that you pass in the second argument.

Statistical functions

DAX adds new statistical functions. The COUNTROWS(Table) function is similar to the Excel COUNT functions (COUNT, COUNTA, COUNTX, COUNTAX, COUNTBLANK), but it takes a table as an argument and returns the count of rows in that table. For example, the following formula returns the number of rows in the ResellerSales table:

```
=COUNTROWS(ResellerSales)
```

Similarly, the DISTINCTCOUNT(Column) function, counts the distinct values in a column. DAX includes the most common statistical functions, such as STDEV.S, STDEV.P, STDEVX.S, STDEVX.P, VAR.S, VAR.P, VARX.S, and VARX.P, for calculating standard deviation and variance. Similar to Count, Sum, Min, Max, and Average, DAX has its own implementation of these functions for better performance instead of just using the Excel standard library.

Filter functions

This category includes functions for navigating relationships and filtering data, including the ALL, ALLEXCEPT, ALLNOBLANKROW, CALCULATE, CALCULATETABLE, DISTINCT, EARLIER, EARLIEST, FILTER, LOOKUPVALUE, RELATED, RELATEDTABLE, and VALUES functions. Next, I'll provide examples for the most popular filter functions.

You can use the RELATED(Column), RELATEDTABLE(Table), and USERELATIONSHIP(Column1, Column2) functions for navigating relationships in the model. The RELATED function follows a many-to-one relationship, such as from a fact table to a lookup table. Consider a calculated column in the ResellerSales table that uses the following formula:

```
=RELATED(Product[StandardCost])
```

For each row in the ResellerSales table, this formula will look up the standard cost of the product in the Product table. The RELATEDTABLE function can travel a relationship in either direction. For example, a calculated column in the Product table can use the following formula to obtain the total reseller sales amount for each product:

```
=SUMX(RELATEDTABLE(ResellerSales), ResellerSales[SalesAmount])
```

For each row in the Product table, this formula finds the corresponding rows in the ResellerSales table that match the product and then it sums the SalesAmount column

across these rows. The USERELATIONSHIP function can use inactive role-playing relationships, as I'll demonstrate in section 8.4.1.

The FILTER (Table, Condition) function is useful to filter a subset of column values, as I've just demonstrated with the AVERAGEX example. The DISTINCT(Column) function returns a table of unique values in a column. For example, this formula returns the count of unique customers with Internet sales:

```
=COUNTRROWS(DISTINCT(InternetSales[CustomerKey]))
```

When there is no table relationship, the LOOKUPVALUE (ResultColumn, SearchColumn1, SearchValue1

[, SearchColumn2, SearchValue2]...) function can be used to look up a single value from another table. The following formula looks up the sales amount of the first line item bought by customer 14870 on August 1st, 2007:

```
=LOOKUPVALUE(InternetSales[SalesAmount],[OrderDateKey],"20070801",[CustomerKey],"14870", [SalesOrderLineNumber],"1")
```

If multiple values are found, the LOOKUPVALUE function will return the error "A table of multiple values was supplied where a single value was expected". If you expect multiple values, use the FILTER function instead.

The CALCULATE(Expression, [Filter1],[Filter2]..) function is a very popular and useful function because it allows you overwrite the filter context. It evaluates an expression in its filter context that could be modified by optional filters. Suppose you need to add a LineItemCount calculated column to the Customer table that computes the count of order line items posted by each customer. On a first attempt, you might try the following expression to count the order line items:

```
=COUNTRROWS(InternetSales)
```

However, this expression won't work as expected (see the top screenshot in **Figure 8.4**). Specifically, it returns the count of all the rows in the InternetSales table instead counting the line items for each customer. To fix this, you need to force the COUNTRROWS function to execute in the current row context. To do this, I'll use the CALCULATE function as follows:

```
=CALCULATE(COUNTRROWS(InternetSales))
```

The CALCUATE function determines the current row context and applies the filter context to the formula. Because the Customer table is related to InternetSales on CustomerKey, the value of CustomerKey for each row is passed as a filter to InternetSales. For example, if the CustomerKey value for the first row is 11602, the filter context for the first execution is COUNTRROWS(InternetSales, CustomerKey=11602).

The screenshot shows two Power BI visual cards side-by-side. Both cards have a header bar with a 'LineItemCount' measure and a dropdown menu.

Card 1 (Top):

```
LineItemCount = COUNTROWS(InternetSales)
```

Card 2 (Bottom):

```
LineItemCount = CALCULATE(COUNTROWS(InternetSales))
```

Both cards display a table with columns: AddressLine2, Phone, DateFirstPurchase, CommuteDistance, and LineItemCount. The data is identical in both cases:

AddressLine2	Phone	DateFirstPurchase	CommuteDistance	LineItemCount
1 (11) 500 555-0125	Friday, January 11, 2008	2-5 Miles	60398	
1 (11) 500 555-0131	Friday, July 21, 2006	0-1 Miles	60398	
1 (11) 500 555-0140	Thursday, July 13, 2006	0-1 Miles	60398	
1 (11) 500 555-0191	Monday, April 21, 2008	2-5 Miles	60398	
1 (11) 500 555-0134	Saturday, February 2, 2008	2-5 Miles	60398	
1 (11) 500 555-0115	Monday, November 12, 2007	0-1 Miles	60398	
1 (11) 500 555-0174	Sunday, January 20, 2008	2-5 Miles	60398	

Figure 8.4 This calculated column in the second example uses the CALCULATE function to pass the row context to the InternetSales table.

The CALCULATE function can also take one or more filters as optional arguments. The filter argument can be a Boolean expression or a table. The following expression returns the transaction count for each customer for the year 2007 and the Bikes product category:

```
=CALCULATE(COUNTROWS(InternetSales), 'Date'[CalendarYear]=2007, Product[ProductCategory]="Bikes")
```

The following expression counts the rows in the InternetSales table for each customer where the product category is “Bikes” or is missing:

```
=CALCULATE(COUNTROWS(InternetSales), FILTER(Product, Product[ProductCategory]="Bikes" || ISBLANK(Product[ProductCategory])))
```

The FILTER function returns a table that contains only the rows from the Product table where ProductCategory=“Bikes”. When you pass the returned table to the CALCULATE function, it’ll filter away any combination of column values that don’t exist in the table.

TIP When the expression to be evaluated is a measure, you can use the following shortcut for the CALCULATE function: =MeasureName(<filter>). For example, =[SalesAmount1]('Date'[CalendarYear]=2006)

Time intelligence functions

One of the most common analysis needs is implementing time calculations, such as year-to-date, parallel period, previous period, and so on. The time intelligence functions require a Date table. The Date table should contain one row for every date that might exist in your data. You can add a Date table using any of the techniques I discussed in Chapter 5. DAX uses the Data table to construct a set of dates for each calculation depending on the DAX formula you specify. For more information about how DAX uses a date table, read the blog post, “Time Intelligence Functions in DAX” by Microsoft’s Howie Dickerman

(<http://bit.ly/daxtfunctions>).

NOTE Readers familiar with Excel data modeling might know that you need to explicitly mark a date table. Power BI Desktop doesn't require you to do so as long as the relationships to the date table join on a column of a Date data type. There is a planned enhancement to support more advanced cases, such as data tables using "smart" Integer keys in the format YYYYMMDD.

As I mentioned in the previous chapter, Power BI doesn't limit you to a single date table. For example, you might decide to import three date tables so you can do analysis on order date, ship date, and due date. Then, you can implement calculations such as:

```
SalesAmountByOrderDate = TOTALYTD(SUM(ResellerSales[SalesAmount]), 'OrderDate'[Date])
```

```
SalesAmountByShipDate = TOTALYTD(SUM(ResellerSales[SalesAmount]), 'ShipDate'[Date])
```

DAX has about 35 functions for implementing time calculations. The functions that you'll probably use most often are TOTALYTD, TOTALQTD, and TOTALMTD. For example, the following formula calculates the YTD sales. The second argument tells DAX which Date table to use a reference point:

```
= TOTALYTD(SUM(ResellerSales[SalesAmount]), 'Date'[Date])
```

— or the following expression to use fiscal years that end on June 30th

```
= TOTALYTD(SUM(ResellerSales[SalesAmount]), 'Date'[Date], ALL('Date'), "6/30")
```

Another common requirement is to implement variance and growth calculations between the current and previous time period. The following formula calculates the sales amount for the previous year using the PREVIOUSYEAR function:

```
=CALCULATE(SUM(ResellerSales[SalesAmount]), PREVIOUSYEAR('Date'[Date]))
```

There are also to-date functions that return a table with multiple periods, including the DATESMTD, DATESQTD, DATESYTD, and SAMEPERIODLASTYEAR. For example, the following measure formula returns the YTD reseller sales:

```
=CALCULATE(SUM(ResellerSales[SalesAmount]), DATESYTD('Date'[Date]))
```

Finally, the DATEADD, DATESBETWEEN, DATESINPERIOD, and PARALLELPERIOD functions can take an arbitrary range of dates. The following formula returns the reseller sales between July 1st 2005 and July 4th 2005.

```
=CALCULATE(SUM(ResellerSales[SalesAmount]), DATESBETWEEN('Date'[Date], DATE(2005,7,1), DATE(2005,7,4)))
```

Ranking functions

You might have a need to calculate rankings. DAX supports ranking functions. For example, the RANK.EQ(Value, Column, [Order]) function allows you to implement a calculated column that returns the rank of a number in a list of numbers. Consider the Rank calculated column in the SalesTerritory table (see **Figure 8.5**).

Rank = RANK.EQ([Revenue], SalesTerritory[Revenue])

SalesTerritoryGroup	SalesTerritoryCountry	SalesTerritoryRegion	SalesTerritoryKey	Revenue	Rank
North America	United States	Southwest	4	\$24,184,609.60	1
North America	Canada	Canada	6	\$16,355,770.46	2
North America	United States	Northwest	1	\$16,084,942.55	3
Pacific	Australia	Australia	9	\$10,655,335.96	4
North America	United States	Central	3	\$7,909,009.01	5
North America	United States	Southeast	5	\$7,879,655.07	6
Europe	United Kingdom	United Kingdom	10	\$7,670,721.04	7
Europe	France	France	7	\$7,251,555.65	8
North America	United States	Northeast	2	\$6,939,374.48	9
Europe	Germany	Germany	8	\$4,878,300.38	10

Figure 8.5 The RANK.EQ function ranks each row based on the value in the REVENUE column.

The formula uses the RANK.EQ function to return the rank of each territory, based on the value of the Revenue column. If multiple territories have the same revenue, they'll share the same rank. However, the presence of duplicate numbers affects the ranks of subsequent numbers. For example, had Southwest and Canada had the same revenue, their rank would be 1, but the Northwest rank would be 3. The function can take an Order argument, such as 0 (default) for a descending order or 1 for an ascending order.

Creating calculated tables

An interesting Power BI Desktop feature is creating calculated tables using DAX. A calculated table is just like a regular table, but it's populated with a DAX function that returns a table instead of using a query. You can create a calculated table by clicking the New Table button in the ribbon's Modeling tab. For example, you can add a SalesSummary table (see **Figure 8.6**) that summarizes reseller and Internet sales by calendar year using the following formula:

```
SalesSummary = SUMMARIZE(ResellerSales, 'Date'[CalendarYear], "ResellerSalesAmount",  
SUM(ResellerSales[SalesAmount]),  
"InternetSalesAmount", SUM(InternetSales[SalesAmount]))
```

The screenshot shows the Power BI Desktop ribbon with the 'Modeling' tab selected. In the 'Calculations' group, the 'New Table' button is highlighted with a large green arrow pointing towards it. To the right of the ribbon, a preview window displays a calculated table named 'SalesSummary' with three columns: 'ResellerSalesAmount', 'InternetSalesAmount', and 'CalendarYear'. The table contains four rows of data corresponding to the years 2005, 2006, 2007, and 2008.

ResellerSalesAmount	InternetSalesAmount	CalendarYear
8065435.3053	6978.26	2005
24144429.654	8349.6628	2006
32202669.4252	11101.8121	2007
16038062.5978	9413.42	2008

Figure 8.6 You can use the New Table button in the ribbon's Modeling tab to create a

calculated table that uses a DAX formula.

This formula uses the SUMMARIZE function which works similarly to the SQL GROUP BY clause. It summarizes the ResellerSales table by grouping by Date[CalendarYear] and computing the aggregated ResellerSales[SalesAmount] and InternetSales[SalesAmount]. Unlike SQL, you don't have to specify joins because the model has relationships from the ResellerSales and InternetSales tables to the Date table.

Now that I've introduced you to the DAX syntax and functions, let's practice creating DAX calculations. You'll also practice creating visualizations in the Reports View to test the calculations but I won't go into the details because you've already learned about visualizations in Chapter 3. If you don't want to type in the formulas, you can copy them from the dax.txt file in the \Source\ch08 folder.

8.2 Implementing Calculated Columns

As I previously mentioned, calculated columns are columns that use DAX formulas for their values. Unlike the regular columns you get when you import data, you add calculated columns after the data is imported, by entering DAX formulas. When you create a report, you can place a calculated column in any area of the Visualizations pane, although you'd typically use calculated columns to group and filter data on the report.

8.2.1 Creating Basic Calculated Columns

DAX includes various operators to create basic expressions, such as expressions for concatenating strings and for performing arithmetic operations. You can use these operators to create simple expression-based columns.

Concatenating text

Suppose you need a visualization that shows sales by employee (see **Figure 8.7**). Since you'd probably need to show the employee's full name, which is missing in the Employee table, let's create a calculated column that shows the employee's full name:

The screenshot shows the Power BI Desktop interface with the Adventure Works - Power BI Desktop file open. The ribbon is visible at the top, showing the Data Tools tab is selected. In the center, there is a table view showing several rows of data. The formula bar at the top has the formula `LineTotal = ([UnitPrice] * (1-[UnitPriceDiscountPct])) * [OrderQty]`. A tooltip message "Column 'OrderQty' cannot be found or may not be used in this expression." is displayed below the formula bar. The table has columns: CustomerPONumber, OrderDate, DueDate, ShipDate, RevisionNumber, and LineTotal. All rows in the LineTotal column currently display "#ERROR".

Figure 8.7 This visualization shows sales by the employee's full name.

1. Open the Adventure Works file with your changes from Chapter 6.
2. Click the Data View icon in the navigation bar. Click the Employee table in the Fields pane to select it.
3. In the Modeling bar, click the New Column button. This adds a new column named "Column" to the end of the table and activates the formula bar.
4. In the formula bar, enter the following formula:

`FullName = [FirstName] & " " & [LastName]`

This formula changes the name of the calculated column to *FullName*. Then, the DAX

expression uses the concatenation operator to concatenate the FirstName and LastName columns and to add an empty space in between them. As you type, AutoComplete helps you with the formula syntax, although you should also follow the syntax rules, such as that a column reference must be enclosed in square brackets.

5. Press Enter or click the checkmark button to the left of the formula bar. DAX evaluates the expression and commits the formula. The FullName column now shows the employee's full name.

TIP Although this exercise is good for demonstrating simple calculated columns, as I discussed before, I recommend you favor query custom columns for simple formulas. The Adventure Works.pbix model in the \Source\ch08 folder includes an FullNameFromQuery column in the Employee table that was produced by adding a custom column to the Employee query. In this case, the query language formula has the same syntax as its DAX calculated column counterpart.

Working with date columns

Instead of DAX calculated columns, you can implement simple expression-based columns in table queries. This is the technique you'll practice next. Some columns in the Date table don't have user-friendly values. For example, you might want to show Q1 2007 as opposed to 1 when the user slices data by quarters:

1. Click the Edit Queries button (in the ribbon's Home tab) to open the Query Editor. In the Queries pane, select the Customer query.

1. Add a FullName custom column to the Customer query with the following formula:

```
=[FirstName] & " " & [LastName]
```

2. In the Queries pane, select the Date query. Add the custom columns shown in **Table 8.4** to assign user-friendly names to months, quarters, and semesters. In case you're wondering, the Text.From() function is used to cast a number to text. An explicit conversion is required because the query won't do an implicit conversion to text, and so the formula will return an error.

3. Click the "Close & Apply" button to apply the changes to the data model.

Table 8.4 Add the following calculated columns in the Date query.

Column Name	Expression	Example
MonthNameDesc	= [MonthName] & " " & Text.From([CalendarYear])	July 2007
CalendarQuarterDesc	= "Q" & Text.From([CalendarQuarter]) & " " & Text.From([CalendarYear])	Q1 2008
FiscalQuarterDesc	= "Q" & Text.From([FiscalQuarter]) & " " & Text.From([FiscalYear])	Q3 2008
CalendarSemesterDesc	= "H" & Text.From([CalendarSemester]) & " " & Text.From([CalendarYear])	H2 2007
FiscalSemesterDesc	= "H" & Text.From([FiscalSemester]) & " " & Text.From([FiscalYear])	H2 2007

4. In the Fields pane, expand the Date table, and click the MonthNameDesc column to select it in the Data View. Click the "Sort By Column" button (ribbon's Modeling tab) to sort the MonthNameDesc column by the MonthNumberOfYear column.

5. To reduce clutter, hide the CalendarQuarter and CalendarSemester columns in the Date table. These columns show the quarter and semester ordinal numbers, and they're not that useful for analysis.

6. In the Reports tab, create a bar chart visualization using the SalesAmount field from the

ResellerSales table (add it to Value area) and the FullName field from the Employee table (add it to the Axis area).

7.Sort the visualization by SalesAmount. Compare your results with **Figure 8.8** to verify that the FullName calculated column is working. Save the Adventure Works model.

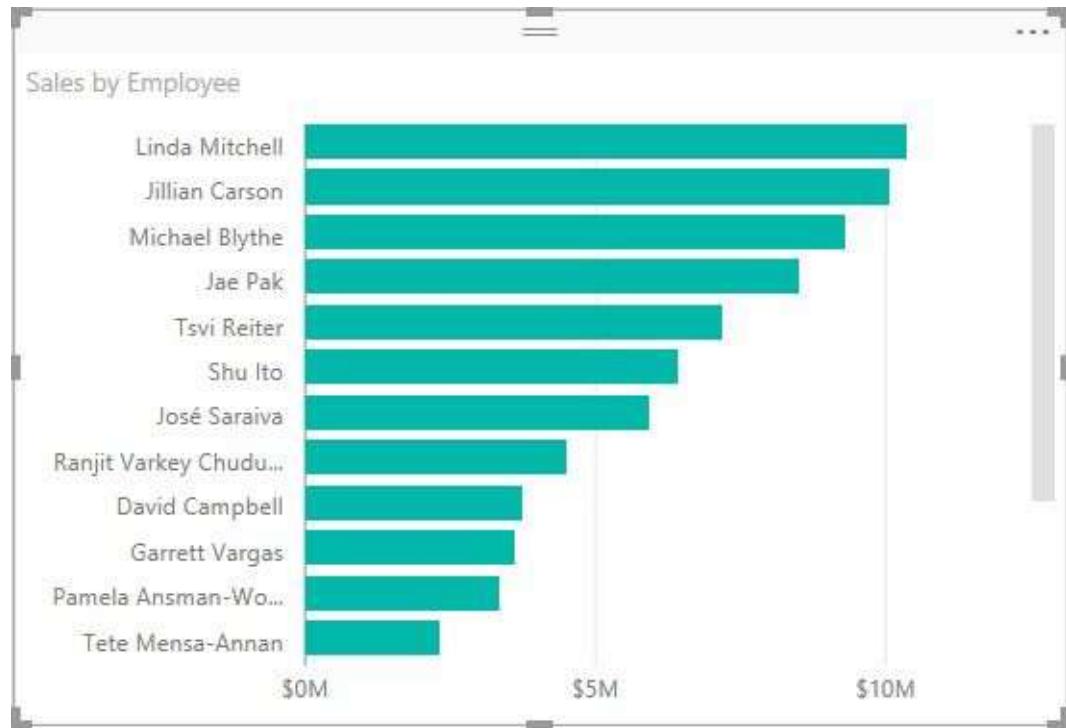


Figure 8.8 The formula bar displays an error when the DAX formula contains an invalid column reference.

Performing arithmetic operations

Another common requirement is to create a calculated column that performs some arithmetic operations for each row in a table. Follow these steps to create a LineTotal column that calculates the total amount for each row in the ResellerSales table by multiplying the order quantity, discount, and unit price:

1. Another way to add a calculated column is to use the Fields pane. In the Fields pane, right-click the ResellerSales table, and then click New Column.

1.In the formula bar, enter the following formula and press Enter. I've intentionally misspelled the OrderQty column reference to show you how you can troubleshoot errors in formulas.

```
LineTotal = ([UnitPrice] * (1-[UnitPriceDiscountPct])) * [OrderQty]
```

This expression multiplies UnitPrice times UnitPriceDiscountPrc times OrderQty. Notice that when you type in a recognized function in the formula bar and enter a parenthesis “(“, AutoComplete shows the function syntax. Notice that the formula bar shows an error “Column ‘OrderQty’ cannot be found or may be used in this expression”. In addition, the LineTotal column shows “Error” in every cell.

2.In the formula bar, replace the OrderQty reference with OrderQuantity as follows:

```
=([UnitPrice] * (1-[UnitPriceDiscountPct])) * [OrderQuantity]
```

3. Press Enter. Now, the column should work as expected.

8.2.2 Creating Advanced Calculated Columns

DAX supports formulas that allow you to create more advanced calculated columns. For example, you can use the RELATED function to look up a value from a related table. Another popular function is the SUMX function, with which you can sum values from a related table.

Implementing a lookup column

Suppose you want to calculate the net profit for each row in the ResellerSales table. For the purposes of this exercise, you'd calculate the line item net profit by subtracting the product cost from the line item total. As a first step, you need to look up the product cost in the Product table.

1. In the Fields pane, add a new NetProfit calculated column to the ResellerSales table that uses the following expression:

```
NetProfit = RELATED(Product[StandardCost])
```

This expression uses the RELATED function to look up the value of the StandardCost column in the Product table. Since a calculated column inherits the current row context, this expression is evaluated for each row. Specifically, for each row DAX gets the ProductKey value, joins to the Product table following the ResellerSales[ProductKey] → Product[ProductKey] relationship, and then retrieves the standard cost for that product from the Product[StandardCost] column.

2. To calculate the net profit as a variance from the line total and the product's standard cost, change the expression as follows:

```
NetProfit = [LineTotal] - RELATED(Product[StandardCost])
```

Note that when the line item's product cost exceeds the line total, the result is a negative value.

Aggregating values

You can use the SUMX function to aggregate related rows from another table. Suppose you need a calculated column in the Product table that returns the reseller sales for each product:

1. Add a new ResellerSales calculated column to the Product table with the following expression:

```
ResellerSales = SUMX(RELATEDTABLE(ResellerSales), ResellerSales[SalesAmount])
```

The RELATEDTABLE function follows a relationship in either direction (many-to-one or one-to-many) and returns a table containing all the rows that are related to the current row from the specified table. In this case, this function returns a table with all the rows from the ResellerSales table that are related to the current row in the Product table. Then, the SUMX function sums the SalesAmount column.

2. Note that the formula returns a blank value for some products because these products

don't have any reseller sales.

Ranking values

Suppose you want to rank each customer based on the customer's overall sales. The RANKX function can help you implement this requirement:

1.In the Fields pane, right-click the Customer table and click New Column.

2.In the formula bar, enter the following formula:

SalesRank = RANKX(Customer, SUMX(RELATEDTABLE(InternetSales), [SalesAmount])),,,Dense)

This function uses the RANKX function to calculate the rank of each customer, based on the customer's overall sales recorded in the InternetSales table. Similar to the previous example, the SUMX function is used to aggregate the [SalesAmount] column in the InternetSales table. The Dense argument is used to avoid skipping numbers for tied ranks (ranks with the same value).

8.3 Implementing Measures

Measures are typically used to aggregate values. Unlike calculated columns whose expressions are evaluated at design time for each row in the table, measures are evaluated at run time for each cell on the report. DAX applies the row, column, and filter selections when it calculates the formula. DAX supports implicit and explicit measures. An implicit measure is a regular column that's added to the Value area of the Visualizations pane. An explicit measure has a custom DAX formula. For more information about the differences between implicit and explicit measures, see Table 8.1 again.

8.3.1 Implementing Implicit Measures

In this exercise, you'll work with implicit measures. This will help you understand how implicit measures aggregate and how you can control their default aggregation behavior.

Changing the default aggregation behavior

I explained before that by default Power BI Desktop aggregates implicit measures using the SUM function for numeric columns and the COUNT function for text-based columns. When you add a column to the Value area, Power BI Desktop automatically creates an implicit measure and aggregates it based on the column type. For numeric columns Power BI Desktop uses the DAX SUM aggregation function. If the column date type is Text, Power BI Desktop uses COUNT. Sometimes, you might need to overwrite the default aggregation behavior. For example, the CalendarYear column in the Date table is a numeric column, but it doesn't make sense to sum it up on reports.

1. Make sure that the Data View is active. In the Fields pane, click the CalendarYear column in the Date table. This shows the Date table in the Data View and selects the CalendarYear column.
2. In the ribbon's Modeling tab, expand the Default Summarization drop-down and change it to "Do Not Summarize". As a result of this change, the next time you use CalendarYear on a report, it won't get summarized.

Working with implicit measures

Suppose you'd like to check if there's any seasonality impact to your business. Are some months slower than others? If sales decrease, do fewer customers purchase products? To answer these questions, you'll create the report shown in **Figure 8.9**. Using the Line and Clustered Column Chart visualization, this report shows the count of customers as a column chart and the sales as a line chart that's plotted on the secondary axis. You'll analyze these two measures by month.

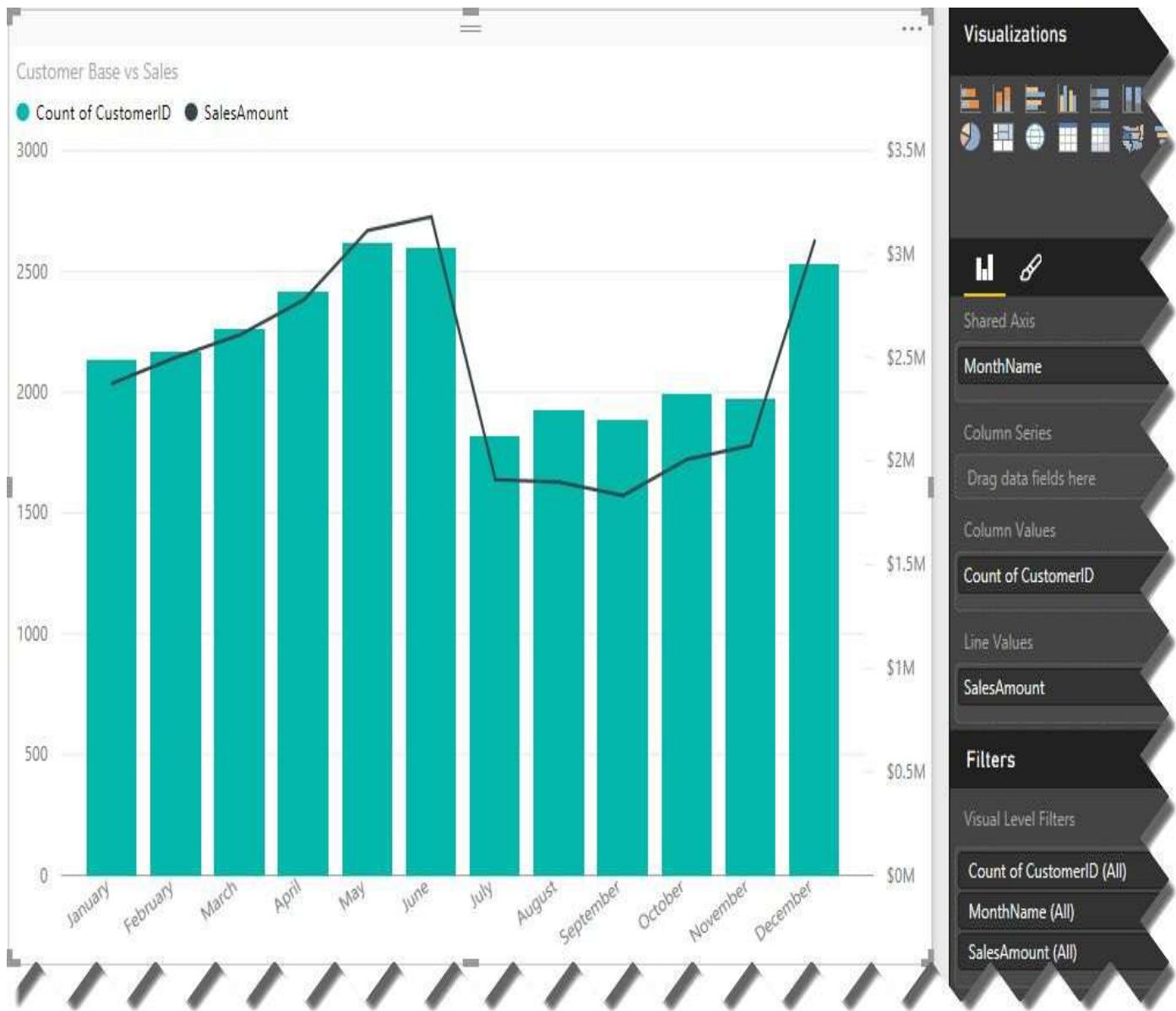


Figure 8.9 Implemented as a combo chart, this visualization shows the correlation between count of customers and sales.

Let's start with visualizing the count of customers who have purchased products by month. Traditionally, you'd add some customer identifier to the fact table and you'd use a Distinct Count aggregation function to only count unique customers. But the InternetSales table doesn't have a CustomerID column, and your chances to get IT to add it to the data warehouse are probably slim. Can you count on the CustomerID column in the Customer table?

NOTE Why not count on the CustomerKey column in InternetSales? This will work if the Customer table handles Type 1 changes only. A Type 1 change results in an in-place change. When a change to a customer is detected, the row is simply overwritten. However, chances are that business requirements necessitate Type 2 changes as well, where a new row is created when an important change occurs, such as when the customer changes its address. Therefore, counting on CustomerKey (called a surrogate key in dimensional modeling) is often a bad idea because it might lead to overstated results. Instead, you'd want to do a distinct count on a customer identifier that is not system generated, such as the customer's account number.

- 1.Switch to the Reports View. From the Fields pane, drag the CustomerID column from the Customer table, and then drop it on an empty area in the report canvas.
- 2.Power BI Desktop defaults to a table visualization. Switch the visualization type to “Line”

and Clustered Column Chart". In the Visualizations pane, drag the CustomerID field from the Shared Axis area to the Column Values area.

- 3.Expand the dropdown in the "Count of CustomerID" field. Note that it uses the Count aggregation function, as shown in **Figure 8.10**.

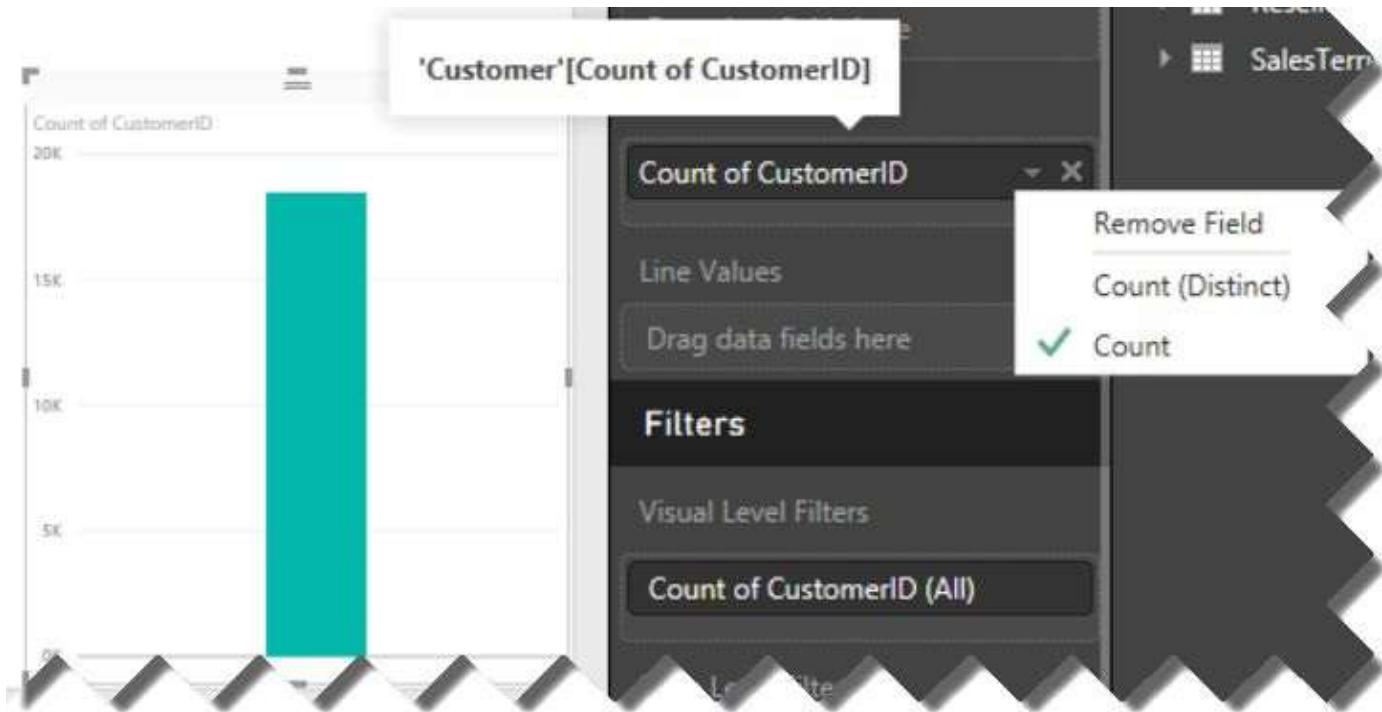


Figure 8.10 Text-based implicit measures use the Count function by default.

- 4.A product can be sold more than once within a given time period. If you simply count on CustomerID, you might get an inflated count. Instead, you want to count customers uniquely. Change the aggregation function of "Count of CustomerID" to Count (Distinct).
- 5.(Optional) Use the ribbon's Modeling tab to change the CustomerID default summarization to Count (Distinct) so you don't have to overwrite the aggregation function every time this field is used on a report.
- 6.With the new visualization selected, check the MonthName column of the Date table in the Fields pane to add it to the Shared Axis area of the Visualizations pane.

At this point, the results are incorrect. Specifically, the count of customers doesn't change across months. The issue is that counting happens over the InternetSales fact table via the Date \texttt{i} InternetSales \texttt{o} Customer relationships. Furthermore, the cardinality of the Date and Customer tables is Many-to-Many (there could be many customers who purchased something on the same date, and a repeating customer could buy multiple times). So you need a bi-directional relationship to the Customer table.

- 7.Switch to the Relationships View. Double-click the InternetSales \texttt{o} Customer relationship. In the Advanced Options properties of the relationship, change the cross filter direction to Both.
- 8.Switch to the Reports View. Note that the visualization now varies by month.
- 9.Drag the SalesAmount field from the InternetSales table to the Line Values area of the Visualizations pane.

Note that because SalesAmount is numeric, Power BI Desktop defaults to the SUM aggregation function. Note also that indeed seasonality affects sales. Specifically, the customer base decreases during the summer. And as number of customers decreases, so do sales.

8.3.2 Implementing Explicit Measures

Explicit measures are more flexible than implicit measures because you can use custom DAX formulas. Similar to implicit measures, explicit measures are typically used to aggregate data and are usually placed in the Value area in the Visualizations pane.

TIP DAX explicit measures can get complex and it might be preferable to test nested formulas step by step. To make this process easier, you can test measures outside Power BI Desktop by using DAX Studio. DAX Studio (<http://daxstudio.codeplex.com>) is a community-driven project to help you write and test DAX queries connected to Excel Power Pivot models, Tabular models, and Power BI Desktop models. DAX Studio features syntax highlighting, integrated tracing support, and exploring the model metadata with Dynamic Management Views (DMVs). If you're not familiar with DMVs, you can use them to document your models, such as to get a list of all the measures and their formulas.

Implementing a basic explicit measure

A common requirement is implementing a measure that filters results. For example, you might need a measure that shows the reseller sales for a specific product category, such as Bikes. Let's implement a BikeResellerSales measure that does just that.

- 1.In Fields pane (Data View or Reports View), right-click the ResellerSales table, and then click New Measure.
- 2.In the formula bar, enter the following formula and press Enter:

BikeResellerSales = CALCULATE(SUM(ResellerSales[SalesAmount]), 'Product'[ProductCategory] = "Bikes")

Power BI Desktop adds the measure to the ResellerSales table in the Fields pane. The measure has a special calculator icon in front of it.

- 3.(Optional) Add a map visualization to show the BikeResellerSales measure (see **Figure 8.11**). Add both SalesTerritoryCountry and SalesTerritoryRegion fields from the SalesTerritory table to the Location area of the Visualizations pane. This enables the drill down buttons on the map and allows you to drill down sales from country to region!

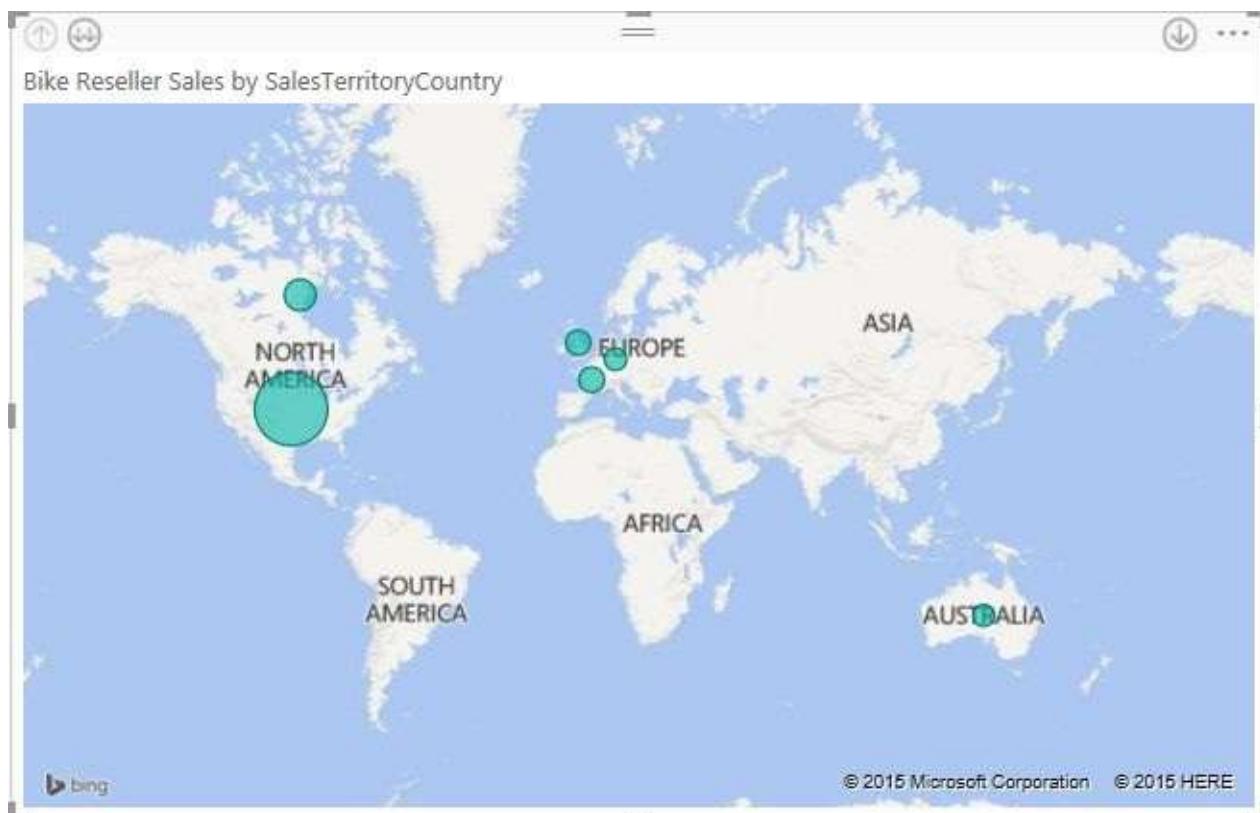


Figure 8.11 This map visualization shows the “Sum of Bike Reseller Sales” measure.

Implementing a percent of total measure

Suppose you need a measure for calculating a ratio of current sales compared to overall sales across countries. For example, for “United States”, the measure calculates the ratio between the sales for United States and the sales across all the territories (see **Figure 8.12**).

SalesTerritoryCountry	2005	2006	2007	2008	Total
Australia			2.63 %	4.66 %	1.98 %
Canada	18.76 %	19.98 %	17.55 %	14.90 %	17.87 %
France		3.55 %	7.37 %	8.58 %	5.73 %
Germany			3.41 %	5.52 %	2.47 %
United Kingdom		3.49 %	6.71 %	7.96 %	5.32 %
United States	81.24 %	72.99 %	62.33 %	58.37 %	66.63 %
Total	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %

Figure 8.12 The PercentOfTotal measure shows the contribution of the country sales to the overall sales.

1. Another way to create a measure is to use the New Measure button. Make sure that the Data View is selected. In the Fields pane, click the ResellerSales table.
1. Click the New Measure button in the Modeling ribbon.
2. In the Formula field, enter the following formula:

PercentOfTotal = DIVIDE (SUM(ResellerSales[SalesAmount]), CALCULATE (SUM(ResellerSales[SalesAmount]), ALL(SalesTerritory))))

To avoid division by zero, the expression uses the DAX DIVIDE function, which performs a safe divide and returns a blank value when the denominator is zero. The SUM function sums the SalesAmount column for the current country. The denominator uses the

CALCULATE and ALL functions to ignore the current context so that the expression calculates the overall sales across *all* the sales territories.

- 3.Click the Check Formula button to verify the formula syntax. You shouldn't see any errors. Press Enter.
- 4.In the Formatting section of the ribbon's Modeling tab, change the Format property to Percentage, with two decimal places.
- 5.(Optional). In the Reports View, add a matrix visualization that uses the new measure (see **Figure 8.12** again). Add the SalesTerritoryCountry to the Rows area and CalendarYear to the Columns area to create a crosstab layout.

Implementing a YTD calculation

DAX supports many time intelligence functions for implementing common date calculations, such as YTD, QTD, and so on. These functions require a column of the Date data type in the Date table. The Date table in the Adventure Works model includes a Date column that meets this requirement. Remember also that for the DAX time calculations to work, relationships to the Date table must use the Date column and not the DateKey column. Let's implement an explicit measure that returns year-to-date (YTD) sales:

1. In the Fields pane, right-click the ResellerSales table, and then click New Measure.
- 1.In the formula bar, enter the following formula:
`SalesAmountYTD =TOTALYTD(Sum(ResellerSales[SalesAmount]), 'Date'[Date])`

This expression uses the TOTALYTD function to calculate the SalesAmount aggregated value from the beginning of the year to date. Note that the second argument must reference the column of the Date data type in the date table.

- 2.To test the SalesAmountYTD measure, create the matrix visualization shown in **Figure 8.13**. Add CalendarYear and MonthName fields from the Date table in the Rows area and SalesAmount and SalesAmountYTD fields in the Values area.

CalendarYear	MonthName	SalesAmount	SalesAmountYTD
2005	July	\$489,328.58	\$489,328.5787
	August	\$1,538,408.31	\$2,027,736.8909
	September	\$1,165,897.08	\$3,193,633.9687
	October	\$844,721.00	\$4,038,354.965
	November	\$2,324,135.80	\$6,362,490.7625
	December	\$1,702,944.54	\$8,065,435.3053
2006	Total	\$8,065,435.31	\$8,065,435.3053
	January	\$713,116.69	\$713,116.6943
	February	\$1,900,788.93	\$2,613,905.6247
	March	\$1,455,20.41	\$4,068,186.0183

Figure 8.13 The SalesAmountYTD measure calculates the year-to-date sales as of any date.

If the SalesAmountYTD measure works correctly, its results should be running totals. For example, the SalesAmountYTD value for 2005 (\$8,065,435) is calculated by summing the

sales of all the previous months since the beginning of year 2005. Moreover, notice that the formula works as of any date. It also works on any field from the Date table. For example, if you add the Date field to the report, then YTD will be calculated as of any day. This brings a tremendous flexibility to reporting!

8.4 Implementing Advanced Relationships

Besides regular table relationships where a lookup table joins the fact table directly, you might need to model more advanced relationships, including role-playing, parent-child, and many-to-many relationships. Next, I'll show you how to meet such requirements with DAX.

8.4.1 Implementing Role-Playing Relationships

In Chapter 5, I explained that a lookup table can be joined multiple times to a fact table. The dimensional modeling terminology refers to such a lookup table as a role-playing dimension. For example, in the Adventure Works model, both the InternetSales and ResellerSales tables have three date-related columns:

OrderDate, ShipDate, and DueDate. However, you only created relationships from these tables to the

OrderDate column. As a result, when you analyze sales by date, DAX follows the InternetSales[OrderDate] → Date[Date] and ResellerSales[OrderDate] → Date[Date] paths.

Creating inactive relationships

Suppose that you'd like to analyze InternetSales by ship date:

1. Click the Manage Relationships button. In the Manage Relationships window, click New.
2. Create the InternetSales[ShipDate] → Date[Date] relationship, as shown in **Figure 8.14**. Note that this relationship will be created as inactive because Power BI Desktop will discover that there's already an active relationship (InternetSales[OrderDate] → Date[Date]) between the two tables.

Create Relationship

X

Select tables and columns that relate to one another.

InternetSales

ctStandardCost	TotalProductCost	SalesAmount	TaxAmt	Freight	OrderDate	DueDate	ShipDate
\$1.8663	\$1.8663	\$4.99	\$0.3992	\$0.1248	8/1/2007	8/13/2007	8/8/2007
\$1.8663	\$1.8663	\$4.99	\$0.3992	\$0.1248	8/2/2007	8/14/2007	8/9/2007
\$1.8663	\$1.8663	\$4.99	\$0.3992	\$0.1248	8/4/2007	8/16/2007	8/11/2007
\$1.8663	\$1.8663	\$4.99	\$0.3992	\$0.1248	8/4/2007	8/16/2007	8/11/2007
\$1.8663	\$1.8663	\$4.99	\$0.3992	\$0.1248	8/5/2007	8/17/2007	8/12/2007

Date

DateKey	Date	DayNumberOfWeek	DayNameOfWeek	DayNumberOfMonth	DayNumberOfYear
20050701	Friday, July 1, 2005	6	Friday		1
20050702	Saturday, July 2, 2005	7	Saturday		2
20050703	Sunday, July 3, 2005	1	Sunday		3
20050704	Monday, July 4, 2005	2	Monday		4
20050705	Tuesday, July 5, 2005	3	Tuesday		5

Advanced options

Cardinality

Many to One (*:1)

Cross filter direction

Single

Make this relationship active

OK Cancel

Figure 8.14 The InternetSales[ShipDate] ð Date[Date] relationship will be created as inactive because there is already a relationship between these two tables.

3.Click OK and then click close.

4.In the Relationships View, confirm that there's a dotted line between the InternetSales and Date tables, which signifies an inactive relationship.

Navigating relationships in DAX

Let's say that you want to compare the ordered sales amount and shipped sales amount side by side, such as to calculate a variance. To address this requirement, you can implement measures that use DAX formulas to navigate inactive relationships. Follow these steps to implement a ShipSalesAmount measure in the InternetSales table:

1. Switch to the Data View. In the Fields pane, right-click InternetSales, and then click New Measure.

1.In the formula bar, enter the following expression:

```
ShipSalesAmount = CALCULATE(SUM([SalesAmount]), USERELATIONSHIP(InternetSales[ShipDate], 'Date'[Date]))
```

The formula uses the USERELATIONSHIP function to navigate the inactive relationship between the ShipDate column in the InternetSales table and the Date column in the Date table.

2.(Optional) Add a Table visualization with the CalendarYear, SalesAmount and ShipSalesAmount fields in the Values area. Notice that the ShipSalesAmount value is different than the SalesAmount value. That's because the ShipSalesAmount measure is aggregated using the inactive relationship.

8.4.2 Implementing Parent-Child Relationships

A parent-child relationship is a hierarchical relationship formed between two entities. Common examples of parent-child relationships include an employee hierarchy, where a manager has subordinates who in turn have subordinates, and an organizational hierarchy, where a company has offices and each office has branches. DAX includes functions that are specifically designed to handle parent-child relationships.

Understanding parent-child relationships

The EmployeeKey and ParentEmployeeKey columns in the Employee table have a parent-child relationship, as shown in **Figure 8.15**. Specifically, the ParentEmployeeKey column points to the EmployeeKey column for the employee's manager. For example, Kevin Brown (EmployeeKey = 2) has David Bradley (EmployeeKey=7) as a manager, who in turn reports to Ken Sánchez (EmployeeKey=112). (Ken is not shown in the screenshot.) Ken Sánchez's ParentEmployeeKey is blank, which means that he's the top manager. Parent-child hierarchies might have an arbitrary number of levels. Such hierarchies are called *unbalanced* hierarchies.

The diagram illustrates a parent-child relationship in the Employee table. A dashed arrow points from the 'EmployeeKey' column to the 'ParentEmployeeKey' column, indicating that the ParentEmployeeKey column contains the identifier of the employee's manager. The table shows 13 rows of data with the following structure:

EmployeeKey	ParentEmployeeKey	SalesTerritoryKey	FirstName	LastName	MiddleName
1	18		Guy	Gilbert	R
2	7		Kevin	Brown	F
3	14		Roberto	Tamburello	NULL
4	112		Rob	Walters	NULL
5	3		Rob	Walters	NULL
6	267		Thierry	D'Hers	B
7	112		David	Bradley	M
8	112		David	Bradley	M
9	23		Jolynn	Dobney	M
10	189		Ruth	Ellerbrock	Ann
11	3		Gail	Erickson	A
12	189		Barry	Johnson	K
13	3		Jossef	Goldberg	H

Figure 8.15 The ParentEmployeeKey column contains the identifier of the employee's manager.

Implementing a parent-child relationship

Next, you'll use DAX functions to flatten the parent-child relationship before you can create a hierarchy to drill down the organizational chart:

1.Start by adding a Path calculated column to the Employee table that constructs the parent-child path for each employee. For the Path calculated column, use the following formula:

Path = PATH([EmployeeKey],[ParentEmployeeKey])

NOTE At this point, you might get an error that the PATH function requires Integer or Text columns. The issue is that the ParentEmployeeKey column has a Text data type. This might be caused by a literal text value “NULL” for Ken Sánchez’s while it should be a blank (null) value. To fix this, open the Query Editor and use the Replace transformation for the ParentEmployeeKey column to replace NULL with blank. Then, in the Query Editor, change the column type to Whole Number and click the “Close & Apply” button.

The formula uses the PATH DAX function, which returns a delimited list of IDs (using a vertical pipe as the delimiter) starting with the top (root) of a parent-child hierarchy and ending with the current employee identifier. For example, the path for Kevin Brown is 112|7|2, where the rightmost part is the ID of the employee on that row, and each segment to the right follow the organizational chain.

The next step is to flatten the parent-child hierarchy by adding a column for each level. This means that you need to know beforehand the maximum number of levels that the employee hierarchy might have. To be on the safe side, add one or two more levels to accommodate future growth.

2.Add a Level1 calculated column that has the following formula:

Level1 = LOOKUPVALUE(Employee[FullName], Employee[EmployeeKey], VALUE(PATHITEM([Path],1)))

This formula uses the PATHITEM function to parse the Path calculated column and to return the first identifier, such as 112 in the case of Kevin Brown. Then, it uses the LOOKUPVALUE function to return the full name of the corresponding employee, which in this case is Ken Sánchez. The VALUE function casts the text result from the PATHITEM function to Integer so that the LOOKUPVALUE function compares the same data types.

3.Add five more calculated columns for Levels 2-6 that use similar formulas to flatten the hierarchy all the way down to the lowest level. Compare your results with **Figure 8.16**. Note that most of the cells in the Level 5 and Level 6 columns are empty, and that's okay because only a few employees have more than four indirect managers.

```
Level16 = LOOKUPVALUE(Employee[FullName],Employee[EmployeeKey],VALUE(PATHITEM([Path],6)))
```

Employee	FullNameFromQuery	Path	Level1	Level2	Level3	Level4	Level5	Level6
Flood	Kathie Flood	112 23 201 13	Ken Sánchez	Peter Krebs	Lori Kane	Kathie Flood		
McAskill	Katie McAskill-White	112 23 186	Ken Sánchez	Peter Krebs	Katie McAskill			
Myer	Ken Myer	112 23 214 19	Ken Sánchez	Peter Krebs	Brenda Diaz	Ken Myer		
Sánchez	Ken Sánchez	112	Ken Sánchez					
Keil	Kendall Keil	112 23 16 31	Ken Sánchez	Peter Krebs	Taylor Maxwe	Kendall Keil		
Ju	Kevin Liu	112 23 214 58	Ken Sánchez	Peter Krebs	Brenda Diaz	Kevin Liu		
Homer	Kevin Homer	112 23 27 232	Ken Sánchez	Peter Krebs	Zheng Mu	Kevin Homer		
Brown	Kevin Brown	112 7 2	Ken Sánchez	David Bradley	Kevin Brown			
Abercrombie	Kim Abercrombie	112 23 18 239	Ken Sánchez	Peter Krebs	Jo Brown	Kim Abercrom		
Ralls	Kim Ralls	112 23 87 74	Ken Sánchez	Peter Krebs	Pilar Ackerman	Kim Ralls		
Zimmerman	Kimberly Zimmerman	112 23 66 238	Ken Sánchez	Peter Krebs	Cristian Petcu	Kimberly Zimm		
en	Kirk Langley	112 23 14 54	Ken Sánchez	Peter Krebs	Jeff H	Kirk Langley		

Figure 8.16 Use the PATHITEM function to flatten the parent-child hierarchy..

- 4.Hide the Path column in the Employee table as it's not useful for analysis.
- 5.(Optional) Create a table visualization to analyze sales by any of the Level1-Level6 fields.
- 6.(Optional) Deploy the Adventure Works model to Power BI Service. To do this, click the Publish button in the ribbon's Home tab, or use the File → Publish → "Publish to Power BI" menu. This will add an Adventure Works dataset and Adventure Works report to the navigation bar in the Power BI portal. Once the model is published, go to Power BI Service (powerbi.com) and test the visualizations you've created. Use your knowledge from Part 1 of this book to explore and visualize the Adventure Works dataset.

8.4.3 Implementing Many-to-Many Relationships

Typically, a row in a lookup table relates to one or more rows in a fact table. For example, a given customer has one or more orders. This is an example of a one-to-many relationship that most of our tables have used so far. Sometimes, you might run into a scenario where two tables have a logical many-to-many relationship. As you've seen in the case of the Customer distinct count example, handling many-to-many is easy, thanks to the DAX bi-directional relationships. But what if you need to report closing balances (common for financial reporting)?

Understanding many-to-many relationships

The M2M.pbix model in the \Source\ch08 folder demonstrates a popular many-to-many scenario that you might encounter if you model joint bank accounts. It consists of five tables, as shown in **Figure 8.17**. The Customer table stores the bank's customers. The Account table stores the customers' accounts. A customer might have multiple bank accounts, and a single account might be owned by two or more customers, such as a personal or business savings account.

The CustomerAccount table is a bridge table that indicates which accounts are owned by which customer. The Balances table records the account balances over time. Note that

the relationships CustomerAccount[AccountNo] ð Account[AccountNo] and CustomerAccount[Customer] ð Customer[Customer] are bi-directional.



Figure 8.17 The M2M model demonstrates joint bank accounts.

Implementing closing balances

If the Balance measure was fully-additive (can be summed across all lookup tables that are related to the Balances table), then you're done. However, semi-additive measures, such as account balances and inventory quantities, are trickier because they can be summed across all the tables except for the Date table. To

understand this, take a look at the report shown in **Figure 8.18**.

Quarter Customer	Q1 2011				Q2 2011			Total
	1/1/2011	2/1/2011	3/1/2011	Total	4/1/2011	5/1/2011	Total	
Alice	100	200	300	300				300
Bob	600	700	300	300				300
John	100	200		200				200
Sam		100	100	100	200	50	50	50
Total	700	1000	400	400	200	50	50	50

Figure 8.18 This report shows closing balances per quarter.

If you create a report that simply aggregates the Balance measure (hidden in Report View), you'll find that the report produces wrong results. Specifically, the grand totals at the customer or account levels are correct, but the rest of the results are incorrect. Instead of using the Balance column, you'll add a new measure that aggregates balances correctly:

1. Add a ClosingBalance explicit measure to the Balances table that uses the following formula:

```
ClosingBalance = CALCULATE(SUM(Balances[Balance]), LASTNONBLANK('Date'[Date],  
CALCULATE(SUM(Balances[Balance]))))
```

This formula uses the DAX LASTNONBLANK function to find the last date with a recorded balance. This function travels back in time, to find the first non-blank date. For John and Q1 2011, that date is 2/1/2011 when John's balance was 200. This becomes the quarter balance for John.

2.Create a Matrix report, such as the one shown in **Figure 8.17**, to verify that the ClosingBalance measure is working as expected.

8.5 Summary

One of the great strengths of Power BI is its Data Analysis Expressions (DAX) language, which allows you to unlock the full power of your data model and implement sophisticated business calculations. This chapter introduced you to the DAX calculations, syntax, and formulas. You can use the DAX formula language to implement calculated columns and measures.

Calculated columns are custom columns that use DAX formulas to derive their values. The column formulas are evaluated for each row in the table, and the resulting values are saved in the model. The practices walked you through the steps for creating basic and advanced columns.

Measures are evaluated for each cell in the report. Power BI Desktop automatically creates an implicit measure for every column that you add to the Value area of the Visualizations pane. You can create explicit measures that use custom DAX formulas you specify.

More complex models might call for role-playing, parent-child, and many-to-many relationships. You can use DAX formulas to navigate inactive relationships, to flatten parent-child hierarchies, and to change the measure aggregation behavior.



Power BI for Pros

BI and IT pros have much to gain from Power BI. Information technology (IT) pros are concerned with setting up and maintaining the necessary environment that facilitates self-service and organizational BI, such as providing access to data, managing security, data governance, and other services. On the other hand, BI pros are typically tasked to create backend services required to support organizational BI initiative, including data marts and data warehouses, cubes, ETL packages, operational reports, and dashboards.

We're back to Power BI Service now. This part of the book gives IT pros the necessary background to establish a trustworthy and collaborative environment! You'll learn how Power BI security works. You'll see how you can create workspaces and groups to promote team BI where multiple coworkers can work on the same BI artifacts. Finally, you'll learn how to create organizational content packs so that you can push BI content to a larger audience and even to the entire company. And you'll discover how the Enterprise Power BI Gateway can help you centralize data management to on-premises data sources.

I'll show BI pros how to integrate popular organizational BI scenarios with Power BI. If you've invested into on-premises Analysis Services models, you'll see how you can create reports and dashboard connected to these models without moving data to the cloud. If you plan to implement real-time BI solutions, I'll show how you can do that with Azure Stream Analytics Service and Power BI. Most of the features discussed in this part of the book require the Power BI Pro edition. And if you're interested in predictive analytics, you'll see how you can integrate Power BI with Azure Machine Learning!

Chapter 9

Enabling Team BI

We all need to share information and this is even more true with BI artifacts that help an organization understand its business. To accomplish this, an IT department (referred to as “IT” in this book) must establish a trustworthy environment where users have secure access to the BI content and data they need. While traditionally Microsoft has promoted SharePoint for sharing all your documents, including BI artifacts, Power BI doesn’t have dependencies on SharePoint Server or SharePoint Online. Power BI has its own sharing and collaboration capabilities! Although these capabilities are available to all users, establishing a cooperative environment should happen under the guidance and supervision of IT. This is why sharing and collaboration are discussed in this part of the book.

Currently, Power BI doesn’t have the SharePoint data governance capabilities, such as workflows, taxonomy, self-service BI monitoring, versioning, retention, and others. Over time, the Enterprise Power BI Gateway is expected to add information management and data governance capabilities to Power BI. Although a large organization might be concerned about the lack of such management features now, Power BI gains in simplicity and this is a welcome change for many who have struggled with SharePoint complexity and for organizations that haven’t invested in SharePoint. This chapter starts by introducing you to Power BI workspaces and groups, and then explains how members of a department or a group can share Power BI artifacts. Next, it shows you how IT can leverage Power BI content packs to bundle and publish content across your organization, and how to centralize data management.

9.1 Understanding Workspaces and Groups

Oftentimes, BI content needs to be shared within an organizational unit or with members of a project group. Typically, the group members require write access so that they can collaborate on the artifacts they produce and create new content. This is where Power BI workspaces and groups can help. Remember that Power BI workspaces and groups are features of Power BI Pro.

For example, now that Martin has created a self-service data model with Power BI Desktop, he would like to share it with his coworkers from the Sales department. Because his colleagues also intend to produce self-service data models, Martin approaches Elena to set up a workspace for the Sales department. The workspace would only allow the members of his unit to create and share BI content. To support similar department or project needs, Elena can set up groups and add the appropriate members to these groups. Then she can create workspaces and grant the groups access to their workspaces.

9.1.1 Understanding Groups

Power BI groups offer a simplified sharing and collaborative experience built on Office 365 groups. Groups bring together people, information and applications to facilitate sharing and collaboration in a secure environment.

What is a group?

From an implementation standpoint, a Power BI group is a record in Azure Active Directory (AAD) that has a shared membership. To understand why you need groups consider that today every Office 365 online application (Exchange, SharePoint, OneDrive for Business, Skype for Business, Yammer, and others) has its own security model, making it very difficult to restrict and manage security across applications. Microsoft introduced groups in Office 365 to simplify this and to foster sharing and collaboration.

NOTE Conceptually, an Office 365 group is similar to a Windows AD group; both have members and can be used to simplify security. However, an Office 365 group has shared features (such as a mailbox, calendar, task list, and others) that Windows groups don't have. Unlike Windows groups, Office 365 groups can't be nested. To learn more about Office 365 groups, read the "Find help about groups in Office 365" document by Microsoft at <http://bit.ly/1BhDecS>.

Although Power BI groups use Office 365 groups behind the scenes, they don't require an Office 365 plan. However, if you have an Office 365 subscription or if you're an administrator of your organization's Office 365 tenant, you can use both Power BI and the Office 365 portal to create and manage groups. Currently, you have to use Office 365 to manage some aspects of the group, such as the group description, and to see the group email address.

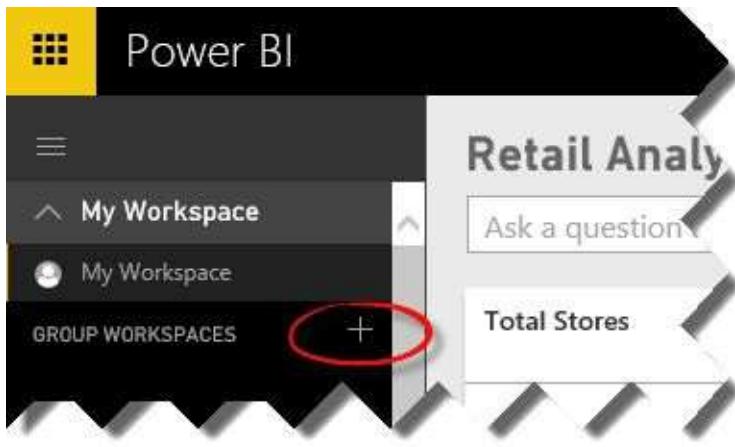


Figure 9.1 Click the Create Group (+) button to create a new group.

Creating groups

Currently, any Power BI Pro user can create a group. The user who creates the group becomes the administrator of the group. The administrator can add other Power BI Pro users as members of the group. Power BI groups have two roles: Admins and Members. Admins can add or remove group members. Creating a group from the Power BI Portal only takes a few mouse clicks:

1. Once you log in to Power BI, expand the My Workspace section in the navigation pane.
2. Click the Create Group (+) button next to the Group Workspaces section, as shown in **Figure 9.1**.
3. In the “Create a Group” window (see **Figure 9.2**), enter the group details and members, and click Save.

When creating a group, you need to specify the group name, such as *Sales Department*. Every group has a unique identifier which Power BI auto-generates from the group name but you can change it if you need some more descriptive. A group can have one of the following privacy settings:

- Private (default) – Only the group members can access the group content.
- Public – Every Power BI Pro user can become a member and gain access to the group content.

Note that you can also specify a privacy level: edit or only view content. It's important to understand that the privacy level is granted at a group level and not at the user level. If you change the group privacy to “Members can only view Power BI content”, all the group members will get read-only rights to the group's content. Now all the members of this group can only view the Power BI dashboards and reports that the group shares. They can't create or edit content, and they can't set any scheduled refreshes. However, admin users always have the rights to create and edit content. It's important to consider content security when you plan your groups.



Figure 9.2 Click the Create Group (+) button to create a new group.

4.If you want to add members right away, enter a name or email alias in the “Add group members” field. You can separate multiple members with a comma (,) or a semi-colon (;). Once the group is created, a welcome page opens that’s very similar to the Get Data page, so that you can start adding datasets the group can work on.

NOTE It might take a while for Office 365 to fully provision a group and enable its collaboration features. When the group is ready, the members will receive an email explaining that they’ve joined the new group. The email includes useful links to get the members started with the collaboration features, including access to group files, a shared calendar, and conversations.

Disabling group creation

One of the benefits of groups is that it enables users to create shared workspaces without relying on IT. At the same time, IT might want to prevent indiscriminate use of this feature. Currently, the only way to disable creating new groups is to use a PowerShell script that applies a policy to the user Exchange mail account. For more information and the steps required to disable groups across the board or for specific users, read the “Disable Group creation” document by Microsoft at <http://bit.ly/1MvPGF0>.

9.1.2 Understanding Workspaces

A Power BI workspace is a container of BI content (datasets, reports, and dashboards) that the group members share and collaborate on. By default, all the content you create goes to the default workspace called “My Workspace”. Think of My Workspace as your private

desk – no one can see its content unless you share it. By contrast, a group workspace is shared by all the group members.

Once you create a group, Power BI creates a workspace that has the same name as the group. Currently, there's a one-to-one relationship between groups and workspaces: one Power BI group has one corresponding workspace. Members who belong to the group will see the group workspace added to the Power BI navigation bar. A Power BI user can be added to multiple groups and can access their workspaces. For example, **Figure 9.3** shows that besides My Workspace, I'm a member of two groups, Finance and Sales Department, and I can access their workspaces.

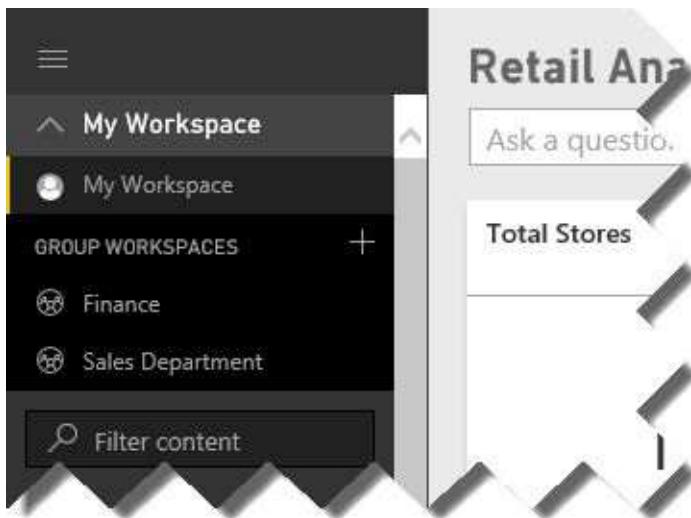


Figure 9.3 Each group has a workspace associated with it.

Besides creating an organizational content pack, currently there isn't another way to move or copy content from one workspace, such as My Workspace, to another. Therefore, it makes sense to create a workspace before you start adding content to Power BI. If you a Power BI Pro user, think of who you want to share the content with and create a group that includes these people. You can use the default My Workspace to publish content that you want to share with people that don't have Power BI Pro rights (remember that simple dashboard sharing works with the Power BI Free edition).

Understanding collaboration features

Because the primary goal of Power BI groups is to facilitate communication and collaboration, groups go beyond BI and support collaborative features. Group members can access these features from the context menu by clicking the ellipsis (...) next to your group name, as shown in **Figure 9.4**. Let's review the available collaboration features:

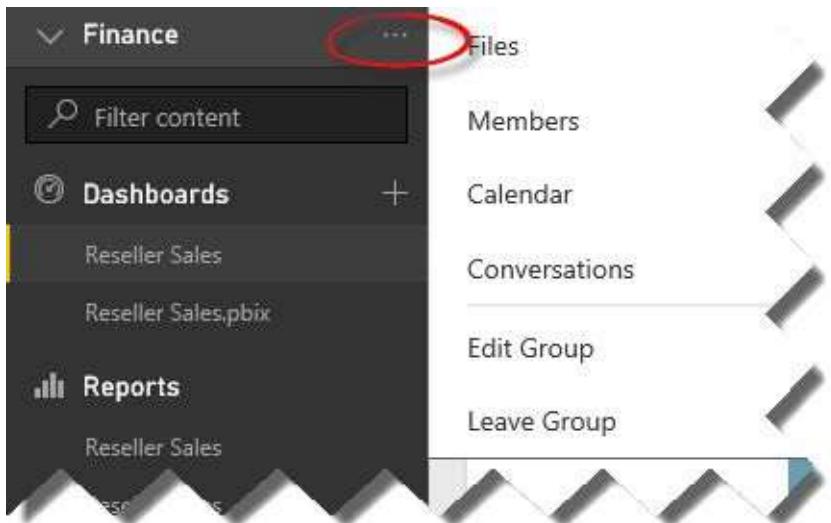


Figure 9.4 Group members have access to a set of features that allows them to communicate and share files.

- Files – Brings you to the OneDrive for Business file storage that's dedicated to the group. While you can save all types of files to OneDrive, Excel workbooks used to import data to Power BI are particularly interesting. As I mentioned, that's because Power BI automatically refreshes the datasets you import from Excel files stored to OneDrive every ten minutes or when the file is updated.
- Members – This menu allows you to manage the group membership, such as to view, add, or remove members.
- Calendar – This brings you to a shared group calendar that helps members coordinate their schedules. Everyone in the group sees meeting invites and other events posted to the group calendar. Events that you create in the group calendar are automatically added and synchronized with your personal calendar. For events that other members create, you can add the event from the group calendar to your personal calendar. Changes you make to those events automatically synchronize with your personal calendar.
- Conversations – Think of a conversation as a real-time discussion list. The conversations page displays each message. If you use Outlook, conversations messages are delivered to a separate folder dedicated to the group. You can either use Outlook or the conversation page to reply to messages and you can include attachments.
- Edit Group – This allows you to change the group name and manage the group members. You can also delete the group. Deleting a group deletes the workspace and its content so be very careful!
- Leave Group – Use this menu if you want to be removed from the group.

Understanding data security

The main goal of workspaces is to allow all group members to access the same data. When the group members explore datasets with imported data, everyone sees the same data. An interesting scenario exists when the group member connects to an on-premises Analysis Services model. In this case, the user identity is carried over to Analysis Services. If the Analysis Services model applies data security based on the user identity, the data will still

be secured and the user can only see the data he's authorized to see.

Remember that when you import data, Power BI datasets cache a copy of the data. However, a user can schedule a data refresh to periodically synchronize the Power BI dataset with changes that occur in the data source. What happens if you need to make changes to the data refresh but this user goes on vacation or leaves the company? Fortunately, another user can take over maintaining the data refresh. To do so, the group member would click the ellipsis button (...) next to the dataset to go to the dataset properties, and then click Schedule Refresh. Then the user clicks Take Over (see **Figure 9.5**) to take ownership of the data refresh and to overwrite the refresh settings if needed.

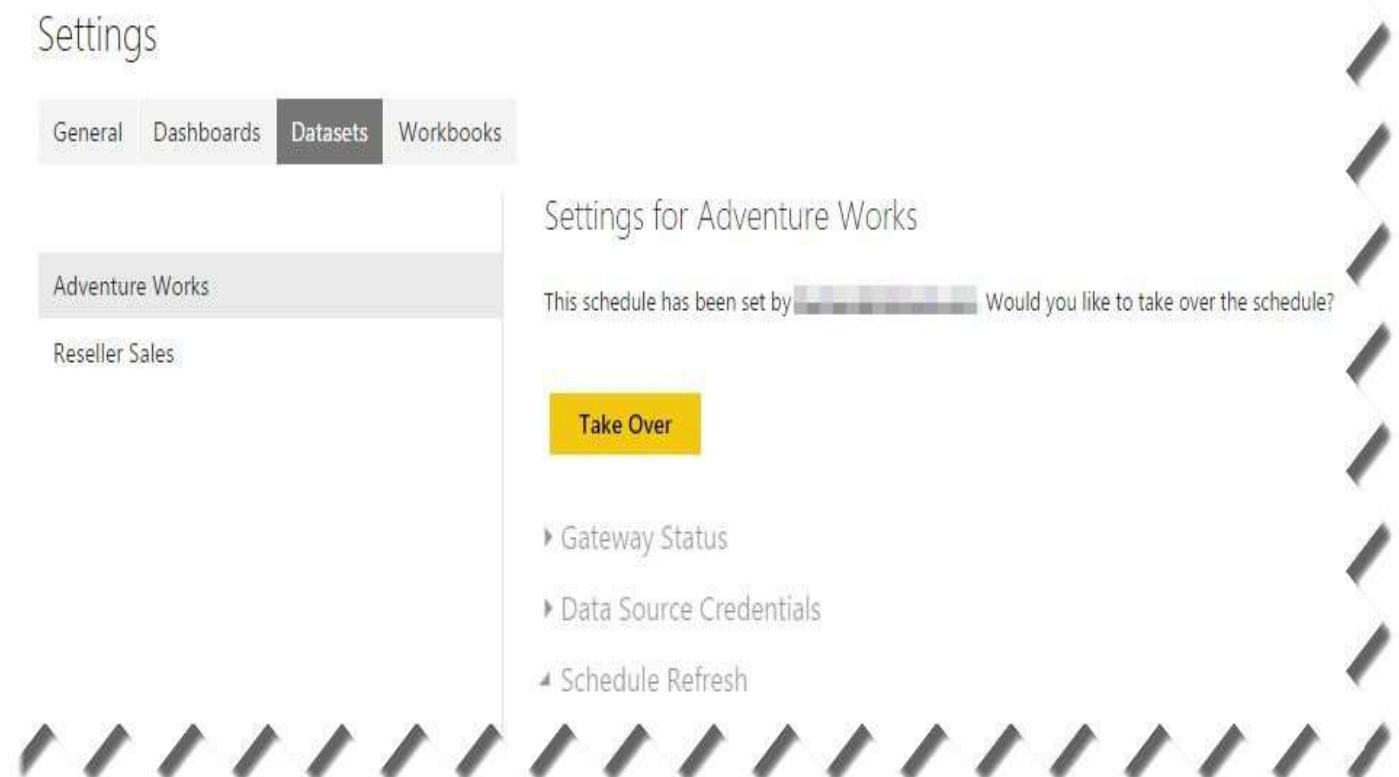


Figure 9.5 A group member can take over the data refresh settings.

9.1.3 Working with Group and Workspaces

We're back to Martin, a data analyst from Adventure Works. Martin approached Elena, who oversees data analytics, to help him set up a workspace for the Sales Department. This workspace will be accessed only by members of his unit and, it'll contain BI content produced by Martin and his colleagues.

NOTE Unless IT has disabled Office 365 group creation, currently there's nothing stopping Martin from creating a group on his own (if he has a Power BI Pro subscription) and becoming the new group's admin. However, I do believe that someone needs to coordinate groups and workspaces, because creating groups indiscriminately could quickly become as useful as having no groups at all. It would be impossible for IT to maintain security over groups they didn't know existed.

Creating a group

As a first step, you need to create a Sales Department group:

1. Open your web browser and navigate to <http://powerbi.com>. Log in to Power BI Service.

- 2.In the left navigation bar, expand My Workspace.
- 3.Click the plus (+) sign next to the Group Workspaces section.
- 4.In the “Create a Group” window, enter *Sales Department* as a group name.
- 5.In the “Add group members” field, enter the email addresses of some of your coworkers, or leave the field blank for now. You don’t have to add your email because your membership is applied when you create the group, and you’ll be the group admin.
- 6.Click Save to create the group.

In the navigation bar, the Group Workspaces section now includes the Sales Department workspace.

Uploading content

As you create a group, Power BI opens a “Welcome to the Sales Department group” page so that you can start adding content immediately. Let’s add the Adventure Works model (that you previously created) to the Sales Department workspace:

The screenshot shows the Power BI interface. At the top, there's a navigation bar with icons for settings, download, help, and user profile. Below the navigation bar is a sidebar containing a tree view of workspaces: Sales Department (expanded), My Workspace, GROUP WORKSPACES (Finance), and another Sales Department entry. A search bar labeled 'Filter content' is also in the sidebar. The main content area features a large heading 'Welcome to the Sales Department group'. Below the heading, a message says 'You're on your way to exploring your data and monitoring what matters with all your group members.' followed by 'Let's start by getting some data.' To the right of this message is a link 'Need more guidance? Try this tutorial'. The main content area is divided into two sections: 'Content Pack Library' and 'Import or Connect to Data'. The 'Content Pack Library' section contains four cards: 'My Organization' (Browse content packs from other people in your organization have published), 'Services' (Choose content packs from online services that you use), 'Files' (Bring in your reports, workbooks, or data from Excel, Power BI Desktop or CSV files), and 'Databases' (Connect to live data in Azure SQL Database and more). Each card has a 'Get' button with a downward arrow. Below the 'Content Pack Library' section is a link 'Samples' with a folder icon.

Figure 9.6 This page allows you to add content to a workspace.

1. In the welcome page, click the Get button in the Files section (**Figure 9.6**). If you close your web browser and go back to Power BI, make sure that you expand My Workspace and select Sales Department so that content is added to this workspace and not to your personal workspace.
 - 1.In the Files page, click Local File.
 - 2.Navigate to the Adventure Works.pbix file you worked on in the previous part of the book, and upload it in Power BI.
 - 3.Using your knowledge from reading this book, view and edit the existing reports, and create some new reports and dashboards. For example, open the Adventure Works.pbix dashboard. Click the Adventure Works.pbix tile to navigate to the underlying report, and then pin some tiles to the dashboard. Back to the dashboard, delete the Adventure Works.pbix tile, and rename the dashboard to *Adventure Works Sales Dashboard*.

Scheduling data refresh

Once you've uploaded the Adventure Works model, you might want to schedule an automatic data refresh. As a prerequisite, you need to install and configure the Power Personal BI Gateway:



NOTE As its name suggests, the Personal Power BI Gateway is for your personal use. You can install it on your computer or another machine that has access to the data source you want to refresh the data from. Each personal gateway installation is tied to the user who installs it and it can't be shared with other users. By contrast, the Enterprise Gateway (discussed in section 9.3) allows IT to centralize and manage access to on-premises data. However, currently the Enterprise Gateway doesn't support refreshing data so you'll use the Personal Gateway.

1. From the Power BI portal, expand the Downloads menu in the top-right corner, and click "Power BI Gateways". In the next page, click the Download button below the "For personal Use" section.
1. Once the setup program starts, follow the steps to install the gateway. When the gateway is installed, a configuration utility pops up. It asks you to specify what Windows account the gateway will run under. By default, it'll use your account so that it can access the data sources on your behalf. The configuration utility will ask you to enter your Windows password. Remember that if your password expires, you'll need to reconfigure the gateway. To do so, click the Power BI Personal Gateway icon in the Windows taskbar or open the application with the same name.



NOTE Depending on the account permissions (admin versus regular user), the Personal Gateway installs either as a service or as an application. If you have administrator access to your computer, it will install as a service (Data Management Gateway Service) and run unattended, even if you log out of the computer. If you change your password, you need to update the service password as well. If you don't have administrator rights, the gateway will run as an application using your current account and password, but you need to be logged in. For more information about these setup scenarios, read the "Power BI Personal Gateway" topic in the Power BI documentation at <https://support.powerbi.com/knowledgebase/articles/649846-power-bi-personal-gateway>.

2. Back in Power BI Service, click the ellipsis (...) button next to the Adventure Works dataset. In the properties window, click Schedule Refresh to open the Settings page.
3. In the Settings page (Datasets tab), expand the Data Source Credentials section, which should look like the one shown in **Figure 9.7**.

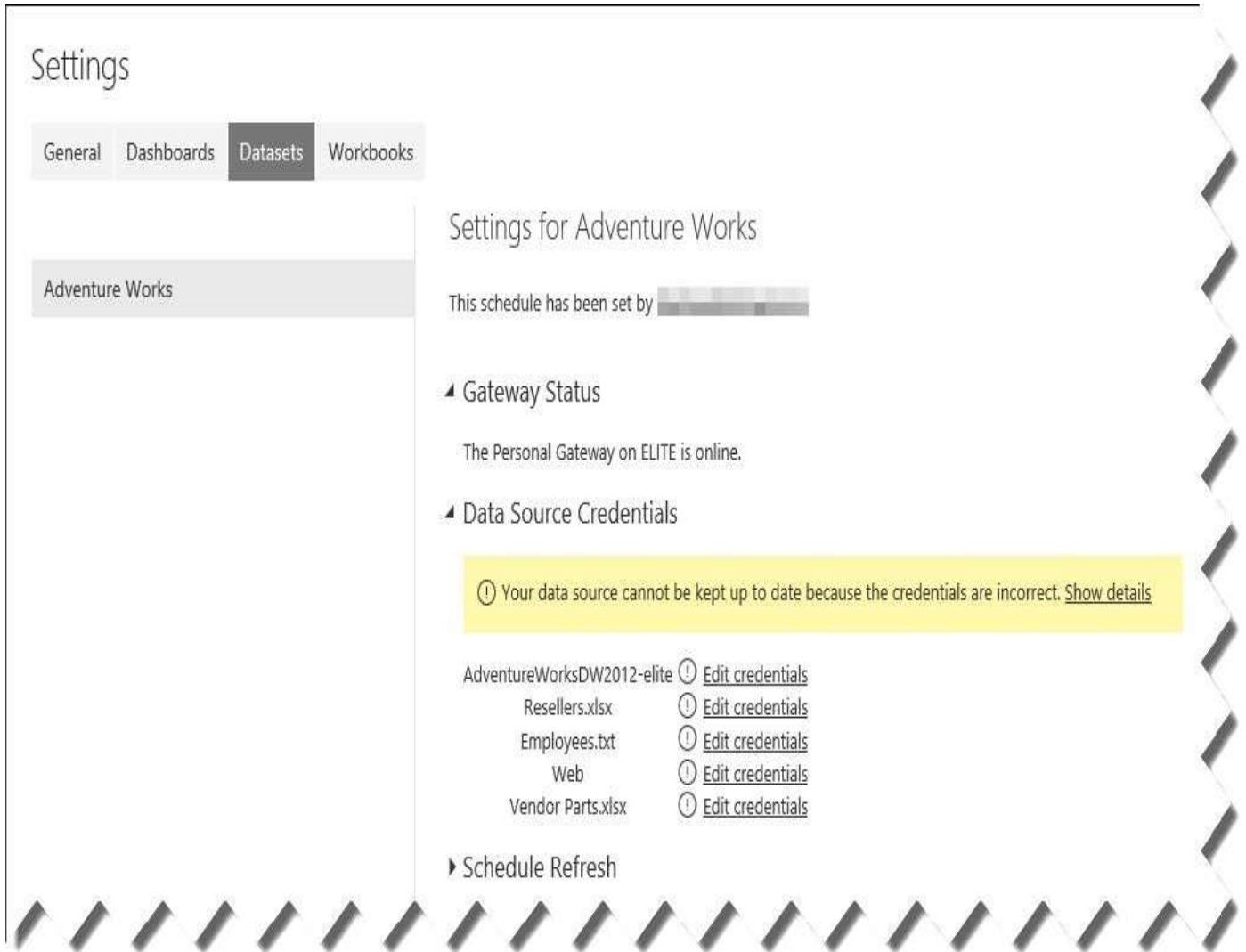


Figure 9.7 The data settings page allows you to configure data refresh.

- 4.The Gateway Status should show that the personal gateway is online on the computer where it's installed.
- 5.The “Data Source Credentials” section shows that the credentials are incorrect. Although this might look alarming, it's easy to fix, and you only need to do it once per data source. Power BI simply requires you to restate how you want to connect to the data source. Click the “Edit Credentials” link for each data source, specify Windows authentication as the authentication method, and then click the Sign In button (see **Figure 9.8**).

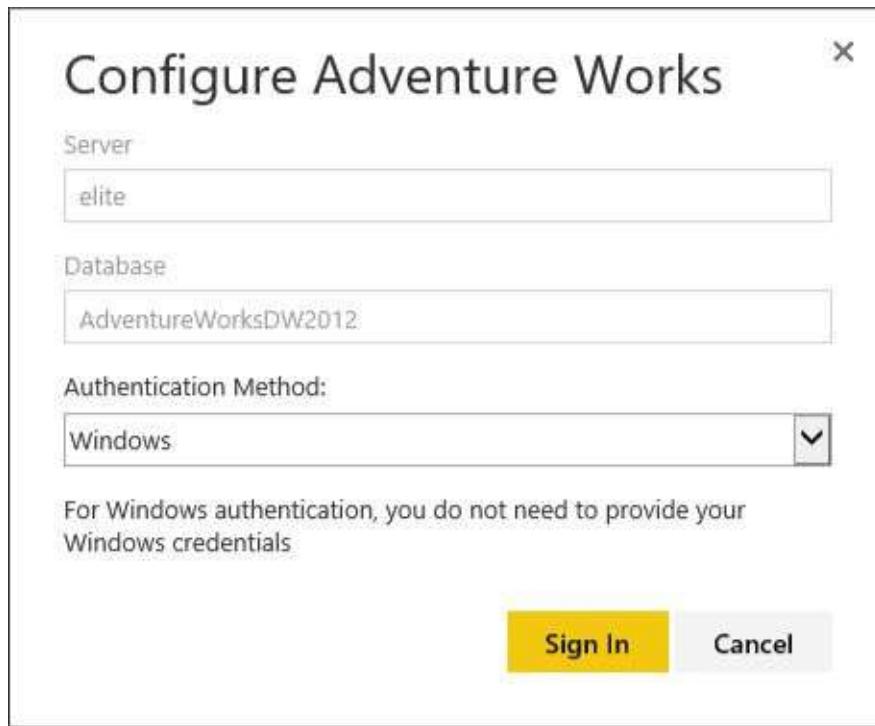


Figure 9.8 The first time you configure scheduled data refresh, you need to reset the authentication method.

6. (Optional) Specify a schedule interval. Back in the Datasets tab of the Settings page (see **Figure 9.7** again), expand the Schedule Refresh section, and update your settings, such as turning on the refresh, refresh frequency, your time zone, time of the refresh, and whether you want refresh failure email notifications. When you're finished, click Apply.

Now the Adventure Works is scheduled for an automatic refresh. When the schedule is up, Power BI will connect to the Personal Gateway, which in turn will connect to all the data sources in the model and will reload the data. Currently, there isn't an option to refresh specific data sources or to specify data source-specific schedules. Once a model is enabled for refresh, Power BI will refresh all the data on the same schedule.

7.(Optional) If another member in your group has scheduled a dataset refresh, go to the dataset Settings page and discover how you can take over the data refresh when you need to. Once you take over, you can overwrite the data source and the schedule settings.

9.2 Packaging and Publishing Content

You saw how workspaces foster collaboration across members of a group. But what if you want to package and publish content to a broader audience, such as across multiple departments or even to the entire organization? Enter Power BI organizational content packs. Your users no longer need to wait for someone else to share content and will no longer be left in the dark without knowing what BI content is available in your company! Instead, users can discover and open content packs from a single place - the Power BI Content Gallery.

9.2.1 Understanding Organizational Content Packs

In Chapter 2, I explained that Power BI comes with content packs that allow your information workers to connect to a variety of online services, such as Google Analytics, Dynamics CRM, QuickBooks, and many more. Microsoft and its partners provide these content packs to help you analyze data from popular cloud services. Not only do content packs allow you to connect easily to external data, but they also include prepackaged reports and dashboards that you can start using immediately! Similar to Power BI content packs, organizational content packs let data analysts and IT/BI pros package and distribute BI content within their organization.

What's an organizational content pack?

An organizational content pack is a way for any company to distribute Power BI content to anyone who's interested in it. A content pack includes specific dashboards, reports, and datasets. Packaging an item publishes dependent items as well, so that the dependencies are included and the package content is functional.

For example, if you publish a dashboard, the content pack will also include the dependent reports and datasets. And if you publish a report, the content pack will include the report dataset. You create a content pack from the Power BI Settings → Create Content Pack menu. Use the “Create Content Pack” to specify which items you want to publish from the workspace you selected.

For example, if you're in your private workspace (My Workspace), you can publish any dashboard, report, or dataset that exists there. However, if you select the Sales Department workspace, you can only publish content from that workspace. Because I was in the Sales Department workspace before creating the content pack the Create Content (see **Figure 9.9**), the page limits me to choose items from this workspace only.

Create Content Pack

Choose who will have access to this content pack:

- Specific Groups My Entire Organization

salesdepartment@prologika.onmicrosoft.com;finance@prologika.onmicrosoft.com

Title

Adventure Works Sales

Description

Summary reports that include results from direct and reseller channels



Select items to publish

Dashboards

Adventure Works.pbix

Reports

Adventure Works

Adventure Works Refresh

Datasets

Adventure Works

Adventure Works Refresh

The content pack will be available in your organization's content gallery. [Learn more](#)

Publish

Cancel

Figure 9.9 An organizational content pack can include dashboards, reports, and dataset references.

I've checked the Adventure Works report. The content pack will automatically include the Adventure Works dataset because that's the dataset that this report uses. If I've decided to publish the Adventure Works dashboard that uses the Adventure Works report, the content pack will include that report and the Adventure Works dataset. If I decide to publish only the datasets, users won't get access to my reports but they can create their own reports from these datasets.

An organizational content pack packages *links* to the original content that you publish. Going back to the example, the consumers of my content pack will gain read-only access by default to the Adventure Works report and its dataset. But what if they want to make

changes?

Understanding content access

In the process of creating an organizational content pack, you specify which users will have access to it. You can publish the content pack to the entire organization or restrict it to specific Power BI, Azure Active Directory, or Exchange distribution groups. Note that you can only publish to groups and not to individuals. As shown back in **Figure 9.9**, I've decided to publish the "Adventure Works Sales" content pack to the Sales Department and Finance groups, and I entered their email aliases in the top field.

NOTE The Power BI portal doesn't show the email group alias of a Power BI/Office 365 group. You can use the Admin section of Office 365 to find the email associated with a group.

Any Power BI Pro user can create an organizational content pack. The content pack author can view and manage the content packs he published by going to the "Power BI Settings" ð "View Content Pack" menu. This opens a page that shows the content packs you published for the active workspace. For example, if you have selected the Sales Department workspace, you'll see the content packs you've published from that workspace (see **Figure 9.10**). The Edit link sends you to the "Update Content Pack" page, which is similar to the "Create Content Package" page. You can click the Delete link to delete a content pack.

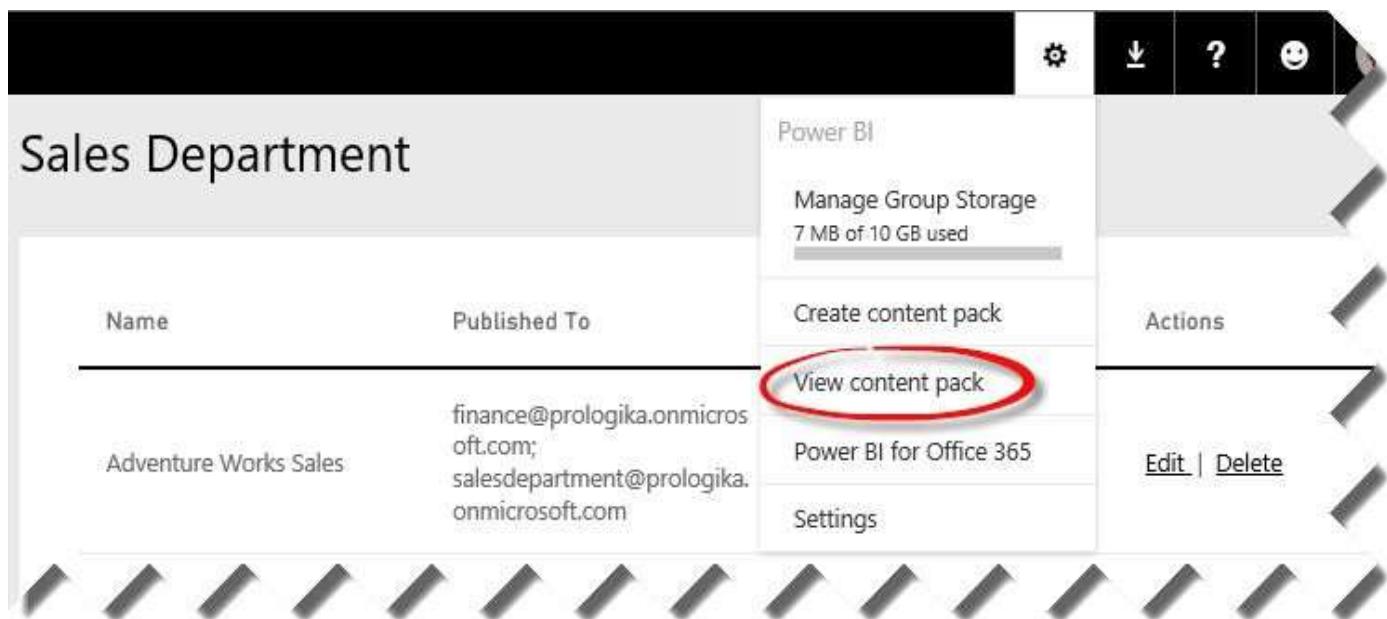


Figure 9.10 Click Power BI Settings ð View Content Pack to view and manage the content packs that you've published.

On the consumer side of things, any Power BI Pro user can consume a content pack. If the content pack is restricted to specific groups, the user must be a member of one or more of these groups. By default, consumers get read-only access to the content. However, a consumer can customize the included dashboards and reports without affecting other users and the original content. When a consumer indicates that he wants to change a BI artifact that's included in a content pack, Power BI creates a copy of the content pack. This copies the original report or dashboard. However, consumers can never modify datasets or change the dataset refresh settings. Only the content pack author can make dataset

changes.

Understanding data security

Content pack data security is similar to workspace data security. If Elena creates and publishes the Adventure Works self-service model created by Martin, every consumer of the content pack will have access to all the data that Martin imported. In other words, Martin and the content pack consumers have the same access to the content pack data. If Martin has scheduled the Adventure Works model for data refresh, the consumers will also see the new data.

What if you want consumers to have different access rights to the data? For example, you might want Martin to have unrestricted access but want Maya to see data for only a subset of your customers. Currently, the only way to address this requirement is to implement an Analysis Services model that applies data security, based on the user identity. In Power BI, you need to create a dataset that connects live to the Analysis Services model. You can create this dataset by using Get Data in the Power BI Portal or by using Power BI Desktop and publishing the model to Power BI Service. Now when Maya uses the content pack, her identity will be passed to Analysis Services, and she'll get restricted access depending on how the SSAS security is set up. I'll discuss this scenario in more detail in the next chapter.

Understanding the Content Gallery

The Power BI Content Gallery allows users to discover and consume organizational content packs. You can access the Content Gallery from the Get Data → My Organization section. The Content Gallery shows the organizational content packs and allows users to search on the content pack name and description. Users can only see content packs that they have permissions to access, via their group membership.

Figure 9.11 shows that the user has permissions to use two content packs. When the user selects a content pack, a popup section is displayed with more information about the content pack, including the author, the created date, and the description. Once the user connects to the pack, the user can access its content.

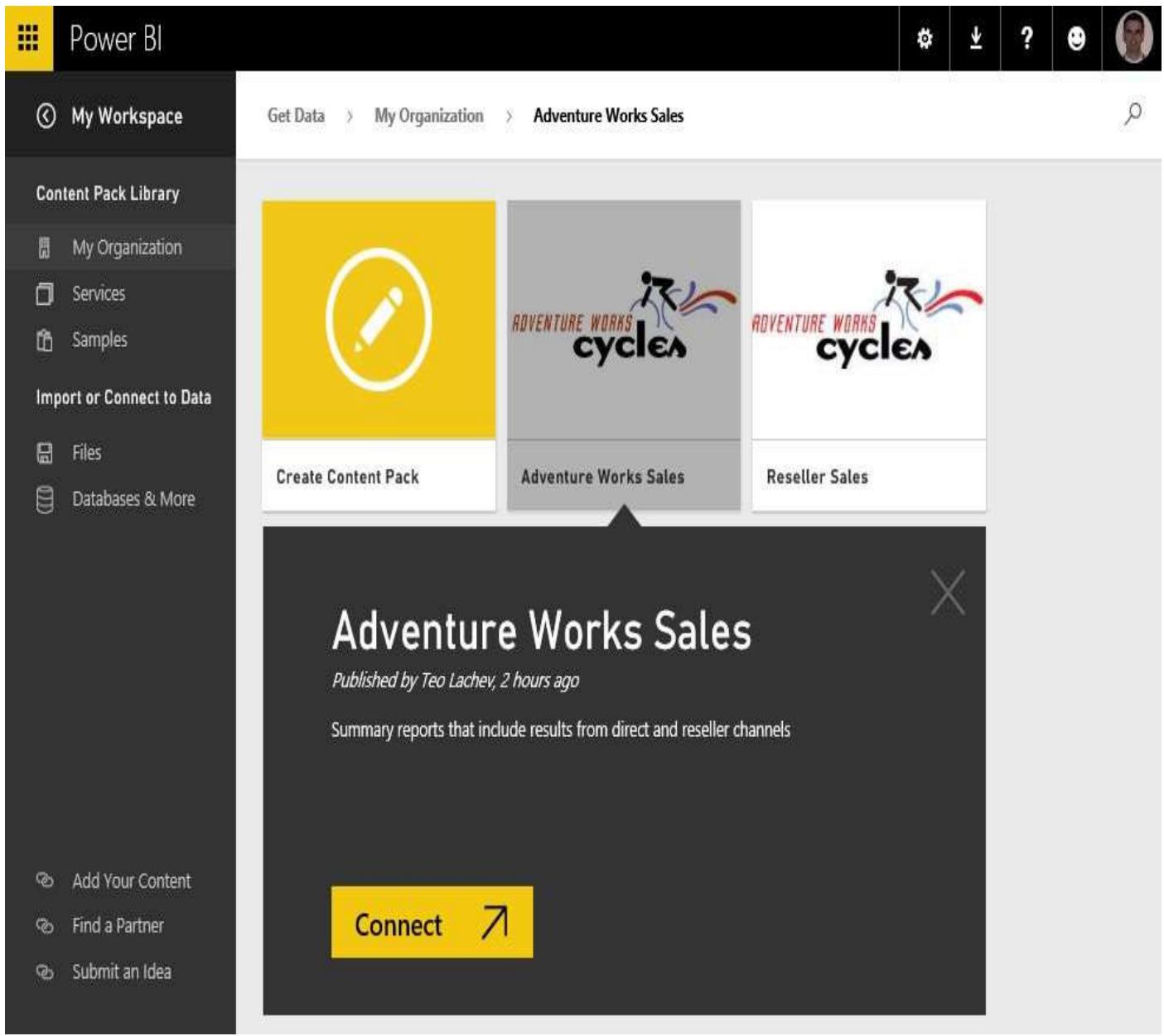


Figure 9.11 The Power BI Content Library allows consumers to discover and use content packs.

9.2.2 Understanding the Content Pack Lifecycle

Let's go through a few scenarios that will help you understand the lifecycle of a content pack. The lifecycle includes initial publishing, consumer access and changes, as well as changes by the content pack author. In this scenario, Elena is the content pack author, and Maya is the consumer.

Initial publishing and access

The content pack lifecycle starts when the content pack is published and available for access:

- 1.Elena belongs to the Sales Department workspace via a Power BI group membership. Martin has informed Elena that the Sales Department has added some BI content that could benefit the finance group and even the entire organization!
- 2.Elena creates and publishes an Adventure Works Sales content pack that includes some

dashboards, reports, and datasets. When Elena uses the “Create Content Pack” page, she grants access to a group that Maya belongs to.

3. Maya logs in to Power BI and navigates to the Content Gallery. Maya discovers the Adventure Works Sales content pack and connects to it. Now Maya has read-only access to the dashboard and reports that are packaged in the Adventure Works Sales content pack.

Changes by consumers

Consumers can personalize content that’s included in an organizational content pack. However, consumers never modify the original content. Instead, when they make a change, Power BI creates a personal copy of the content pack. At this point, the content exists in two places: the original content as published by the report author and a personal copy for the consumer. Let’s follow Maya as she makes some changes:

1. Maya analyzes the reports and dashboards in the Adventure Works Sales dashboard.
2. Maya wants to change a dashboard (or a report), such as to add a new dashboard tile. If Maya attempts to add a tile to the dashboard, she won’t be able to do so because the dashboard name will be grayed out. Upon further inspection, Maya sees a padlock icon next to the dashboard name.
3. Maya clicks the icon, and she’s prompted whether or not she wants to get a personal copy of the content pack (see **Figure 9.12**). If Maya wants to change a report that’s included in the content pack, she’ll get the same prompt when she opens the report in Reading View, and clicks the “Edit report” menu to switch to Editing View.



Figure 9.12 Consumers can personalize dashboards and reports included in a content pack.

4. Maya clicks the Save button. Power BI clones the content of the organizational content pack. Maya looks at the navigation bar and sees another set of reports, dashboards, and datasets with identical names as the ones included in the content pack. Now she can change her copy of the pack content. Her changes never affect other users or the original content.

TIP Personalizing a content pack clones all the content but keeps the same item names. At this point Maya can get confused about which ones are the personalized clones and which are the original versions. She can click an item and learn about its origin by inspecting the menus and icons, but I’d recommend she renames the personalized copies to

make it clear. For example, she can rename the cloned reports to *My Adventure Works Dashboard* or *My Sales Report*.

Changes by the content pack publisher

Content packs are likely to undergo changes. For example, as the business evolves, Elena might want to make changes to the content pack reports and dashboards:

1.Elena opens the Adventure Works dashboard (or report) and makes a change. For example, she changes an existing visualization or adds a visualization.

2.Power BI pops up the message shown in **Figure 9.13**



Figure 9.13 Power BI shows this message when the publisher changes content included in an organizational content pack.

3.Now Elena has to decide if she wants to push the changes to all the consumers. If she wants to do that, she can click the “View Content Packs” button or go to “Power BI Settings” ð “View Content Pack”.

4.This action shows all content the packs she created (see again **Figure 9.10**). However, this time the Adventure Works Sales content pack has a warning icon next to its name. When Elena hovers her mouse over the warning icon, a message pops up to let her know that she needs to update the content pack, as shown in **Figure 9.14**.



Figure 9.14 Power BI informs the content pack publisher about pending changes.

5.Elena updates the content pack.

6.The next time that Maya views the original dashboard or report, she sees Elena’s changes. However, Maya’s personalized content doesn’t change.

Deleting content packs

A content pack reaches the end of its lifecycle when it’s no longer needed, and so it’s removed:

1.The Adventure Works Sales content pack is outdated, and Elena needs to remove it. When she attempts to delete it, she’s prompted to confirm her intention and warned that anyone who’s connected to the content pack will no longer be able to access it.

2.Elena confirms the prompt, and Power BI removes the content pack.

3.When Maya opens the Power BI portal, she notices that the original reports and datasets are gone. The dashboards are still there but all the original tiles are removed. However, Maya’s personal copy remains unaffected as long as the datasets are still functional.

9.2.3 Comparing Sharing Options

To recap, Power BI supports three ways of sharing BI content: simple dashboard sharing, workspaces, and organizational content packs. Because having that many options could be confusing, Table 9.1 summarizes their key characteristics and usage scenarios.

Table 9.1 This table compares the sharing options supported by Power BI.

	Dashboard Sharing	Group Workspaces	Organizational Content Packs
Purpose	Ad hoc dashboard sharing	Team collaboration	Broader content delivery
Discovery	Invitation email or direct sharing to another user's workspace	Workspace content	Content Gallery
Target audience	Selected individuals (like your boss)	Groups (your team)	Anyone who might be interested
Content permissions	Read-only dashboards	Read/edit to all workspace content	Read only by default and edit for a personalized copy
Communication		Calendar, conversations, files	
License	Power BI Free	Power BI Pro	Power BI Pro

Dashboard sharing

The primary purpose of dashboard sharing is the ad hoc sharing of dashboards only by sending an email to selected individuals or direct sharing to their workspace. For example, you might want to share your dashboard with your boss or a teammate. Consumers can't edit the shared dashboards. Dashboard sharing is the only sharing option supported by the free version of Power BI (Power BI Free).

Group workspaces

Group workspaces foster team collaboration and communication. They're best suited for departments or project teams. Based on Office 365 groups, group workspaces allow all the members of a group to edit and view the workspace content. Group workspaces are the only option that supports shared communication features, including OneDrive for Business file sharing, a shared calendar, and conversations.

Organizational content packs

Organizational content packs are designed for broader content delivery, such as across groups or even across the entire organization. Consumers discover content packs in the Power BI Content Gallery. By default, consumers get read-only access to the published content, but they can create personalized copies.

9.2.4 Working with Organizational Content Packs

Several departments at Adventure Works have expressed interest in the content that the Sales Department has produced, so that they can have up-to-date information about the company's sales performance. Elena decides to create an organizational content pack in order to publish these artifacts to a broader audience.

Creating an organizational content pack

As a prerequisite, Elena needs to discover if there are any existing Active Directory, Exchange Distribution Lists, or Office 365 groups that include the authorized consumers.

This is an important step from a security and maintenance standpoint. Elena doesn't want the content pack to reach unauthorized users. She also doesn't want to create a new group if she can reuse what's already in place. Elena needs to be a member of the Sales Department Power BI group so that she has access to this workspace.

- 1.Elena discusses this requirement with the Adventure Works system administrator, and she discovers that there's already an AD group for the Finance Department. Since the rest of the users come from other departments, the system administrator recommends Elena creates an Office 365 group for them.
- 2.Elena uses Power BI Service (she can also use the Office 365 Admin Center) to set up a new group. Now she has two groups to distribute the content to. Elena records the group email aliases. At this point, Elena might want to make sure the target users have Power BI Pro licenses so that they can consume the pack.
- 3.In Power BI Service, Elena expands "My Workspace" in the navigation bar and selects the Sales Department workspace. She does this so that, when she creates a content pack, she'll only see the Sales Department workspace content.
- 4.In Power BI Service, Elena expands the Settings menu in the top-right corner, and then she clicks "Create Content Pack".
- 5.In the "Create Content Pack" page (see **Figure 9.9** again), Elena enters the group email alias. She fills in the content pack name and enters a useful description about the content pack's purpose. Then she selects the dashboards and reports that she wants to distribute, and she clicks Publish.

At this point, the Adventure Works Sales content pack is published and ready for authorized consumers.

Consuming an organizational content pack

Maya learns about the availability of the sales content pack, and she wants to use it. Maya belongs to one of the groups that's authorized to use the pack.

- 1.Maya logs in to Power BI and clicks Get Data → My Organization.

TIP If Maya wants to share the content pack with members of a group workspace she belongs to, Maya can expand My Workspace and select the workspace. Then when she imports the content pack, all the group members will have access to the content pack.

- 2.If there many content packs available, Maya uses the search box to search for "sales".
- 3.Maya discovers the Adventure Works Sales content pack. She selects it and clicks Connect. Power BI adds links to the navigation bar for all dashboards, reports, and datasets included in the content pack. Maya can now gain insights from the prepackaged content.
- 4.To review and practice the different steps of the content pack lifecycle, follow the instructions in section 9.2.2.

9.3 Centralizing Data Management

Because it's a cloud platform, Power BI requires special connectivity software to access on-premises data. You've seen how the Personal Power BI Gateway allows end users to refresh datasets connected to on-premises data sources, such as relational databases and files. And the Analysis Services connector lets users connect Power BI Service and Power BI Desktop directly to Analysis Services models. However, these connectivity mechanisms don't give IT the ability to centralize and sanction data access. The Enterprise Power BI Gateway (currently in preview), will eventually fill in this gap.

9.3.1 Understanding the Enterprise Gateway

Think of the Enterprise Power BI Gateway as a successor of the Office 365 Data Management Gateway that you might be familiar with if you have used the previous version of Power BI. Currently, it supports the following features:

- Serving many users – Unlike the Personal Gateway which serves the needs of individuals, the administrator can configure one or more enterprise gateways for entire team and even the organization.
- Centralizing management of data sources, credentials, and access control – The administrator can use one gateway to delegate access to multiple databases and can authorize individual users or groups to access these databases.
- Providing DirectQuery access from Power BI Service to on-premises data sources – Once Martin creates a model that connects directly to a SQL Server database, Martin can publish the model to Power BI Service and its reports will just work.

Understanding limitations

As the rest of Power BI, the Enterprise Gateway is evolving rapidly. Microsoft released the first preview build in the beginning of December 2015. As it stands, the Enterprise Gateway only supports DirectQuery connections to SQL Server databases. Recall from Chapter 5, that DirectQuery allows analysts to create data models that connect directly to a subset of data sources, including SQL Server. This feature was prioritized because the Personal Gateway doesn't support DirectQuery connections.

As it stands, the Enterprise Gateway doesn't allow you to refresh data from on-premises data sources. It also doesn't let you connect live to Analysis Services models (Microsoft announced that this feature is coming soon). All users that connect to on-premises data sources via the Enterprise Gateway require Power BI Pro subscriptions.

Comparing connectivity options

Because having different connectivity options to on-premises data can be confusing, Table 9.2 compares them to help you decide if you need the Enterprise Gateway. However, remember that Power BI is evolving at a rapid space, so the Enterprise Gateway is expected to add more compelling features over time and to unify the capabilities of the Personal Gateway and Analysis Services Connector.

Table 9.2 This table compares connectivity options to on-premises data.

	Enterprise Gateway	Personal Gateway	Analysis Services Connector
Purpose	Centralized data management	Isolated access to data by individuals	Connect live to SSAS models
Audience	IT	Business users	IT
DirectQuery	Yes (SQL Server only)	No	SSAS only
Data refresh	Coming soon	Yes	Not needed
User access	Users and groups managed by IT	Individual user	Controlled by SSAS security
Data sources	Multiple data sources (SQL Server for now, more coming soon)	All refreshable data sources	One SSAS instance
User license	Power BI Pro	Power BI Pro	Power BI Pro

9.3.2 Getting Started with the Enterprise Gateway

Next, I'll show you how to install and use the Enterprise Gateway. While you can install the gateway on any machine, you should install it on a dedicated server within your corporate network. You can install multiple gateways if needed, such as to assign department-level admin access.

Installing the Enterprise Gateway

Follow these steps to download the gateway:

1. Remote in to the server where you want to install the gateway. This server should have a fast connection to the on-premises databases that the gateway will access. Verify that you can connect to the target databases.
2. Open the web browser and log in to Power BI Service.
3. In the Application Toolbar located in the upper-right corner of the Power BI portal, click the Download menu, and then click "Power BI Gateways".
4. In the next page, click Download below the "For enterprise deployments" section.
5. Once you download the setup executable, run it. Select the installation folder, read and accept the agreement, and then click Install. The gateway installs and runs as a Windows service called Power BI Enterprise Gateway Service (PBIEgwService) and its default location is the \Program Files\Power BI Enterprise Gateway folder.

Configuring the Enterprise Gateway

Next, you need to configure the gateway:

1. Once the setup program installs the gateway, it'll ask you to sign in to Power BI.
1. Specify the gateway name and a recovery key (see **Figure 9.15**). Save the recovery key in a safe place. Someone will need it to restore the gateway if admin access is lost.

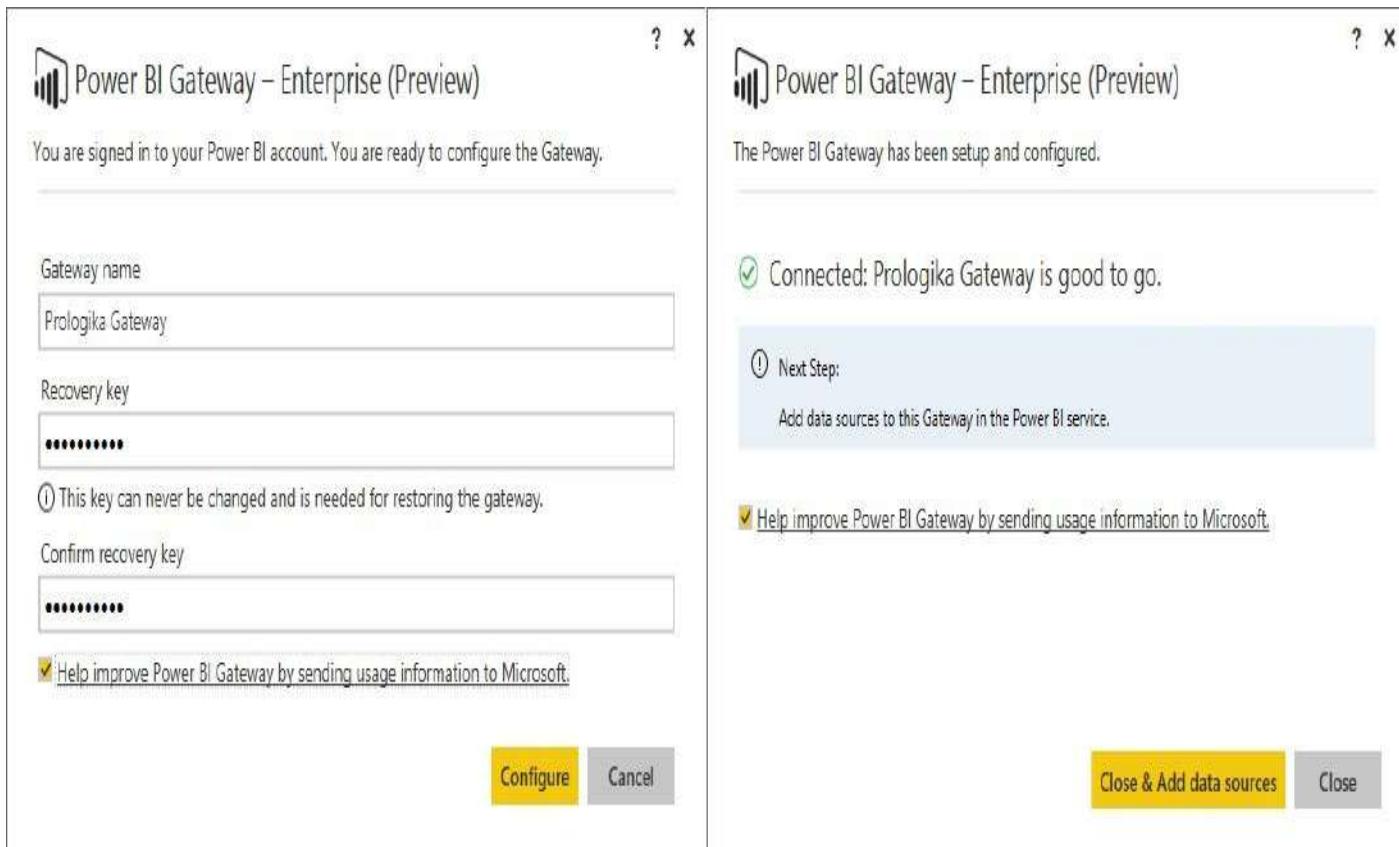


Figure 9.15 When you configure the Enterprise Gateway, you need to give it a name and provide a recovery key.

- 2.Click Configure. This registers the gateway with Power BI. You should see a message that the gateway is connected.
- 3.Click “Close & Add data sources”.

NOTE Data transfer between the Power BI service and the gateway is secured through Azure Service Bus. The gateway communicates on outbound ports TCP 443 (default), 5671, 5672, 9350 thru 9354. The gateway doesn't require inbound ports, so you don't have to open any ports in your company firewall.

Gateways

+ ADD DATA SOURCE

Prologika Gateway

New data source

Data Source Settings

Data Source Name

Adventure Works DW (Prod)

Data Source Type

SQL Server

Server

ELITEVM1

Database

AdventureWorksDW2012

Authentication Method

Windows

The credentials are encrypted using the key stored on-premises on the Gateway server. [Learn more](#)

Username

PROLOGIKA\powerbi

Password

Add

Discard

Figure 9.16 The enterprise Gateway can provide access to many databases.

Managing the Enterprise Gateway

The next phase of configuring the gateway needs to be done in Power BI Service, as follows:

1. The setup program should navigate you to the Gateways page in Power BI Service. In future, to access this page, click the Settings menu in the Application Toolbar in the upper-right corner, and then click “Manage gateways”.
- 1.In the Gateways page, select your gateway and notice that you can enter additional gateway settings, such as the department and description. Moreover, you can specify additional administrators who can manage the gateway (the person who installs the gateway becomes the first administrator).
- 2.Next, add one or more data sources that the gateway will delegate access to. Suppose you want to set up DirectQuery to the AdventureWorksDW2012 database. In the Gateways page, click “Add Data Source” (see **Figure 9.16**).

- 3.Fill in the data source settings to reflect your database setup. Currently, the only data source type that the Enterprise Gateway supports is SQL Server but more types will be added in future.
- 4.The Authentication method allows you to specify the credentials of a trusted Windows account or a standard SQL Server login that has access to the database. Remember to grant this account at least read credentials to the database, such by assigning it to the SQL Server db_reader role. Note that all queries from all users will use these credentials to connect to the database, so grant the account only the minimum set of permissions it needs to the SQL Server database. This might result in different data permissions than the permissions a data analyst had when he used Power BI Desktop to connect to SQL Server under *his* credentials.

NOTE The gateway uses asymmetric encryption to encrypt the credentials so that they cannot be decrypted in the cloud. Power BI sends the credentials to the gateway server which decrypts the credentials when the data sources are accessed.

- 5.By default, only the administrators can access the gateway. Once you add a data source and click the data source in the Gateways page, you'll see a new Users tab. You can use this tab to specify the email addresses of individual users or groups that can use the gateway.

TIP If you have issues with the Enterprise Gateway setup or data source access, you can configure it for troubleshooting. You can find the troubleshooting steps in the “Troubleshooting the Power BI Gateway – Enterprise” page at <https://powerbi.microsoft.com/en-us/documentation/powerbi-gateway-enterprise-tshoot>.

9.3.3 Using the Enterprise Gateway

Once the Enterprise Gateway is set up and functional, you can use it from published reports that access on-premises SQL Server databases with DirectQuery. Setting up DirectQuery connections to SQL Server is currently only available in Power BI Desktop, so you must create a data model.

Connecting directly to SQL Server

Follow these steps to create a simple data model that you can use to test the gateway:

- 1.Open Power BI Desktop and expand the Get Data button in the ribbon's Home tab. Select SQL Server.
- 2.In the SQL Server Database prompt, specify the name of the SQL Server instance.
- 3.In the Navigator Window, expand the database the gateway delegates access to, select one or more tables, and then click Load.
- 4.In the Connection Settings window, select DirectQuery and click OK.
- 5.Publish the model to Power BI by clicking the Publish button in the ribbon's Home page.

Testing connectivity

Next, test that the Enterprise Gateway is functional from Power BI Service:

- 1.Log in to Power BI.
- 2.In the left navigation bar, click the dataset to explore it. Note that it's not enough to see the model metadata showing in the Fields pane because the list comes from the Tabular

backend database where the model is hosted. You need to visualize the data to verify that the gateway is indeed functional.

3.In the Fields pane, check a field to create a visualization. If you see results on the report, then the gateway works as expected. You can also use the SQL Server Profiler to verify that the report queries are sent to the SQL Server database.

9.4 Summary

Power BI allows teams to collaborate and share BI artifacts via dashboard sharing, group workspaces, and organizational content packs. Dashboard sharing is available on the free version of Power BI, but you'll need the Power BI Pro license for group workspaces and organizational content packs. This chapter started by introducing you to group workspaces. Based on Office 365 groups, workspaces allow a team to collaborate and share Power BI content. In addition, group workspaces include communication features, such as calendar, files, and conversations, which allow the group to share information.

Organizational content packs are designed to complement group workspaces by letting content producers share their insights with other teams and even with the entire organization. Authorized users can discover content packs in the Power BI Content Gallery. When consumers connect to a content pack, they can view all the published content, but they can't edit the content. However, they can create personalized copies without affecting other content pack consumers or the original content.

This chapter compared the three sharing and collaboration options and recommended usage scenarios. It also walked you through a few exercises to help you practice the new concepts. It also showed you how the Enterprise Power BI Gateway is positioned to centralize and simplify access to on-premises data.

Now that you know about sharing, Chapter 10 will show you how you can create more advanced BI solutions that integrate with Power BI!

Chapter 10

Organizational BI

So far, the main focus of this book has been the self-service and team flavors of Power BI, which empower business users and data analysts to gain insights from data and to share these insights with other users. Now it's time to turn our attention to BI pros who implement organizational BI solutions. Back in Chapter 2, I compared self-service and organizational BI at a high level. I defined organizational BI as a set of technologies and processes for implementing an end-to-end BI solution where the implementation effort is shifted to BI professionals.

This chapter shows BI pros how to build common organizational BI solutions. You'll understand the importance of having an organizational semantic layer, and you'll learn how to integrate it with Power BI. Implementing real-time BI solutions is one of the fastest growing BI trends. I'll show you how this can be done using the Azure Stream Analytics Service, while you use Power BI for real-time dashboards. Because predictive analytics is another popular scenario, I'll show you how you can integrate Power BI with predictive models that you publish to Azure Machine Learning.

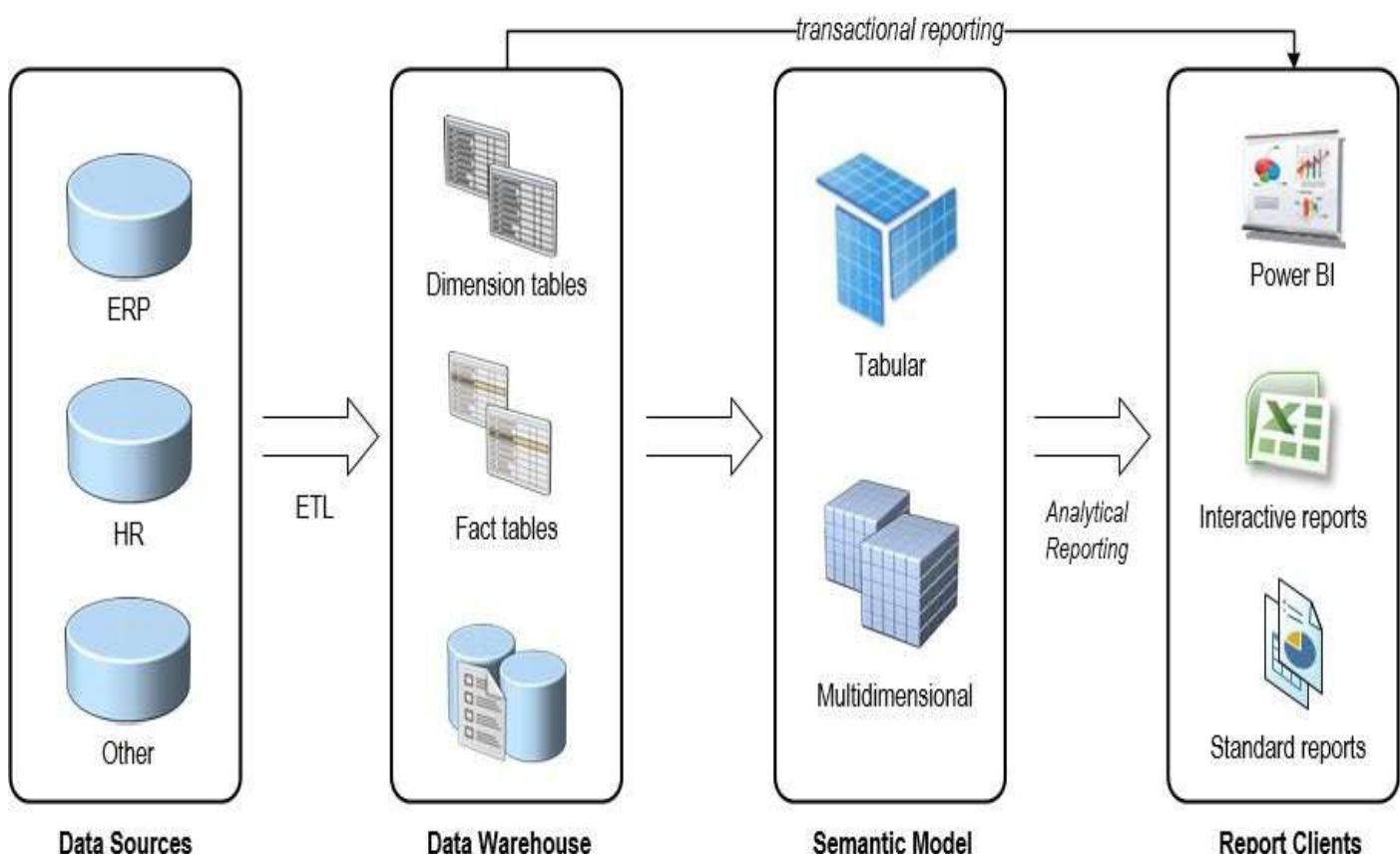


Figure 10.1 Organizational BI typically includes ETL processes, data warehousing, and

a semantic layer.

10.1 Implementing Classic BI Solutions

In Chapter 2, I introduced you at a high level to what I refer to as a classic BI solution (see **Figure 10.1**). This diagram should be familiar to you. Almost every organization nowadays has a centralized data repository, typically called a data warehouse or a data mart, which consolidates cleaned and trusted data from operational systems.

REAL LIFE Data warehousing might mean different things to different people. In my consulting practice, I've seen data warehouse "flavors" ranging from normalized operational data stores (ODS) to hub-and-spoke architectures. If they work for you then that's all that matters. I personally recommend and implement a consolidated database designed in accordance with Ralph Kimball's dimensional modeling (star schema), consisting of fact and dimension tables. For more information about dimensional modeling, I recommend the book "The Data Warehouse Toolkit" by Ralph Kimball and Margy Ross.

You might not have an organizational semantic layer that sits between the data warehouse and users, and you might not know what it is. In general, semantics relates to discovering the meaning of the message behind the words. In the context of data and BI, semantics represents the user's perspective of data: how the end user views the data to derive knowledge from it. As a BI pro, your job is to translate the machine-friendly database structures and terminology into a user-friendly semantic layer that describes the business problems to be solved. In Microsoft BI, the role of this layer is fulfilled by Microsoft BI Semantic Model (BISM).

10.1.1 Understanding Microsoft BISM

Microsoft BISM is a unifying name for several Microsoft technologies for implementing semantic models, including self-service models implemented with Excel Power Pivot or Power BI Desktop, and organizational Microsoft Analysis Services models. From an organizational BI standpoint, BI pros are most interested in Microsoft Analysis Services modeling capabilities.

NOTE Besides the necessary fundamentals, this chapter doesn't attempt to teach you Multidimensional or Tabular. To learn more about Analysis Services, I covered implementing Analysis Services Multidimensional models in my book "Applied Microsoft Analysis Services 2005" and Tabular models in "Applied Microsoft SQL Server 2012 Analysis Services: Tabular Modeling".

Introducing Multidimensional and Tabular

Since its first release in 1998, Analysis Services has provided Online Analytical Processing (OLAP) capabilities so that IT professionals can implement Multidimensional OLAP cubes for descriptive analytics. The OLAP side of Analysis Services is referred to as Multidimensional. Multidimensional is a mature model that can scale to large data volumes. For example, Elena can build a Multidimensional cube on top of a large data warehouse with billions of rows, while still providing an excellent response time where most queries finish within a second!

Starting with SQL Server 2012, Microsoft expanded the Analysis Services capabilities by adding a new path for implementing semantic models, where entities are represented as relational-like constructs, such as two-dimensional tables, columns, and relationships. Referred to as Tabular, this technology uses the same xVelocity engine that powers Power BI Desktop, Excel Power Pivot, Power BI, and SQL Server columnstore indexes.

Although not as scalable as Multidimensional, Tabular gains in simplicity and flexibility. And because Tabular uses the same storage engine, if you know how to create self-service data models in Power BI Desktop or Excel, you already know 90% of Tabular!

Understanding implementation paths

Microsoft organizational BISM can be visualized as a three-tier model that consists of data access, business logic, and data model layers (see **Figure 10.2**). The data model layer is exposed to external clients. Clients can query BISM by sending Multidimensional Expressions (MDX) or Data Analysis Expressions (DAX) queries. For example, Excel can connect to both Multidimensional and Tabular and send MDX queries, while Power BI and Power View send DAX queries.

The business logic layer allows the modeler to define business metrics, such as variances, time calculations, and key performance indicators (KPIs). In Multidimensional, you can implement this layer using Multidimensional Expressions (MDX) constructs, such as calculated members, named sets, and scope assignments. Tabular embraces the same Data Analysis Expressions (DAX) that you've learned about in Chapter 8 when you added business calculations to the Adventure Works model.

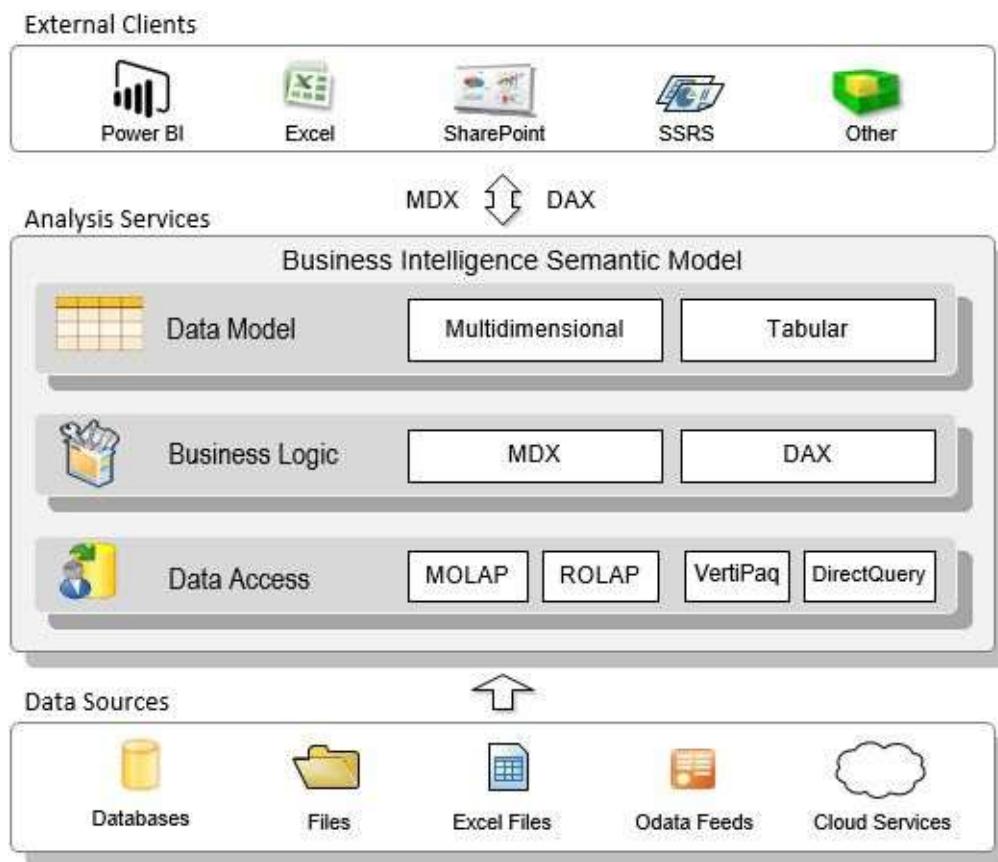


Figure 10.2 BISM has two organizational BI implementation paths: Multidimensional and Tabular.

The data access layer interfaces with external data sources. By default, both Multidimensional and Tabular import data from the data sources and cache the dataset on the server for best performance. The default multidimensional storage option is Multidimensional OLAP (MOLAP), where data is stored in a compressed disk-based format. The default Tabular storage option is xVelocity, where data is initially saved to

disk but later loaded in memory when users query the model.

Both Multidimensional and Tabular support real-time data access by providing a Relational OLAP (ROLAP) storage mode for Multidimensional and a DirectQuery storage mode for Tabular. When a Multidimensional cube is configured for ROLAP, Analysis Services doesn't process and cache the data on the server. Instead, it auto-generates and sends native queries to the database. Similarly, when a Tabular model is configured for DirectQuery, Analysis Services doesn't keep data in xVelocity; it sends native queries directly to the data source. As I mentioned in Chapter 5, the DirectQuery mode of Tabular enables DirectQuery connections in Power BI Desktop.

Understanding the BISM advantages

Having a semantic model is very valuable for organizational BI for the following main reasons:

- Larger data volumes – Remember that Power BI Service currently limits imported files to 250 MB each, including Power BI Desktop models. This size limit won't be adequate for organizational solutions that are typically built on top of corporate data warehouses. By contrast, BISM can scale to billions of rows and terabytes of data!
- Great performance – BISM is optimized to aggregate data very fast. Queries involving regular measures and simple calculations should be answered within seconds even, when aggregating millions of rows!
- Single version of the truth – Business logic and metrics can be centralized in the semantic model instead of being defined and redefined in the database or in reports.

REAL WORLD The BI department of a retail company included some 20 report developers on staff whose sole responsibility was creating operational reports from stored procedures. Over time, stored procedures have grown in complexity, and developers have defined important business metrics differently. Needless to say, operational reports didn't tally. Although the term "a single version of the truth" is somewhat overloaded, a centralized semantic model can get pretty close to it.

- Data security – Currently, Power BI Desktop and Excel data models don't support data security. If the user has access to the model, the user can see all the data in the model. By contrast, BISM models can apply data security based on the user's identity, such as to allow Maya to only see the data for the customers she's authorized to access.
- Implicit relationships – In the process of designing the semantic layer, the modeler defines relationships between entities, just like a data analyst does when creating a self-service model. As a result, end users don't need to join tables explicitly because the relationships have already been established at design time. For example, you don't have to know how to join the Product table to the Sales table. You simply add the fields you need on the report, and then the model knows how to relate them!
- Interactive data analysis – From an end-user perspective, data analysis is a matter of dragging and dropping attributes on the report to slice and dice data. A "smart" client, such as Power BI, auto-generates report queries, and the server takes care of aggregating data, such as to summarize sales at the year level.
- Good client support – There are many reporting tools, including Power BI, Excel,

Reporting Services, and third-party tools, that support BISM and address various reporting needs, including standard reports, interactive reports, and dashboards.

TIP After reading all of this, still not convinced that you should invest in Analysis Services models? Remember that this is not the only path for accessing on-premises data. In Chapter 5, I showed you how you can implement Power BI Desktop models that connect directly to on-premises SQL Server databases. For example, you can implement a model that connects directly to your data warehouse.

When should you upgrade to organizational BI?

You might have started your BI journey with a self-service model you created in Excel or Power BI Desktop. At what point should you consider switching to an organizational solution? What would you gain?

REAL WORLD Everyone wants quick and easy BI solutions, ideally with a click of a button. But the reality is much more difficult. I often find that companies have attempted to implement organizational BI solutions with self-service tools, like Power BI Desktop, Excel, or some other third-party tools. The primary reasons are cutting cost and misinformation (that's why it's important to know who you listen to). The result is always the same – sooner or later the company finds that it's time to “graduate” to organizational BI. Don't get me wrong, though. Self-service BI has its important place, as I explain in Chapter 2. But having trusted organizational-level solutions will require the right architecture, toolset, and investment.

You should consider engaging BI pros when the following happens:

- Data integration – The requirements call for centralizing and consolidating data instead of implementing isolated self-service data models.
- Data complexity – You realize that the data complexity exceeds the capabilities of the Power BI Desktop queries. For example, the integration effort required to clean and transform corporate data typically exceeds the simple transformation capabilities of self-service models.
- Data security – Recall that self-service models currently don't support data security, such as to allow Martin to access all the customers, but to restrict Maya to only see a subset of the customers. Unlike Power Pivot models, where security is an all-or-nothing proposition, organizational BI models can have row-level data security, such as to allow a sales representative to see only the data for resellers that he's responsible for.
- Enterprise scalability – Power BI Desktop and Excel models import data in files. Once you get beyond a few million rows, you'll find that you stretch the limits of these tools. For example, it'll take a while to save and load the file. It'll take even longer to upload the file to Power BI. By contrast, organizational models must be capable of handling larger data volumes. Just by deploying the model to an Analysis Services instance, you gain better scalability that's boosted by the hardware resources of a dedicated server. Also, to reduce the data load window and to process data incrementally, you can divide large tables in partitions.
- Faster data refresh – Unlike the Power BI Desktop sequential data refresh, organizational models support processing tables and partitioned tables in parallel, in order to better utilize the resources of a dedicated server.

NOTE Currently, Tabular supports upgrading from Power Pivot only. There's no upgrade path from Power BI Desktop models. That's because Microsoft updates Power BI Desktop on a monthly basis, and it can get ahead of Tabular, which is included in the SQL Server box product and only releases a new version every few years.

Comparing self-service and organizational models

Since you're reading this book, the logical transition path is from Power BI Desktop (or Excel) to Tabular. Then you'll discover that you can seamlessly transition your knowledge to organizational projects. Indeed, an organizational Tabular model has the same foundation as Power Pivot or Power BI Desktop, but it adds enterprise-oriented and advanced features, such as options for configuring security, scalability, and low latency. Table 10.1 provides a side-by-side comparison between self-service and organizational features.

Table 10.1 This table highlights the differences between self-service and organizational semantic models.

Feature	Self-service BISM	Organizational BISM
Target users	Business users	Professionals
Environment	Power BI Desktop or Excel	Visual Studio (SSDT)
xVelocity Engine	Out of process (local in-memory engine)	Out of process (dedicated Analysis Services instance)
Size	One file	Large data volumes, table partitions
Refreshing data	Sequential table refresh	Parallel table refresh, incremental processing. Parallel partition refresh in SQL Server 2016.
Development	Ad-hoc development	Project (business case, plan, dates, hardware, source control)
Lifespan	Weeks or months	Years

How Power BI Service connects to on-premises Analysis Services

Now that you understand the benefits of a semantic model, let's see how Power BI integrates with Analysis Services. Currently, the Power BI Analysis Services Connector supports only connections to Tabular. It doesn't support connecting to Multidimensional and Tabular in SharePoint integration mode (Power Pivot for SharePoint uses this mode). I hope this limitation is lifted soon so that you can connect Power BI to cubes and to Power Pivot models deployed to SharePoint.

From the Power BI standpoint, the BISM integration scenario that will inspire the most interest is the hybrid architecture shown in **Figure 10.3**. This architecture will be also appealing for companies that prefer to keep data on premises but want to host only report and dashboard definitions to on Power BI.

Elena has implemented an Analysis Services Tabular model layered on top of the Adventure Works data warehouse, and she deployed it to an on-premises server. "On premises" could mean any server that's external to Power BI, such as a server hosted in the company's data center or on an Azure virtual machine. So Power BI can connect to the model, Elena installs the Analysis Services Connector (or the Enterprise Power BI Gateway when it supports SSAS connections) on an on-premises computer that can connect to Analysis Services. Elena has granted Maya access to the model.

Live dashboards and exploration

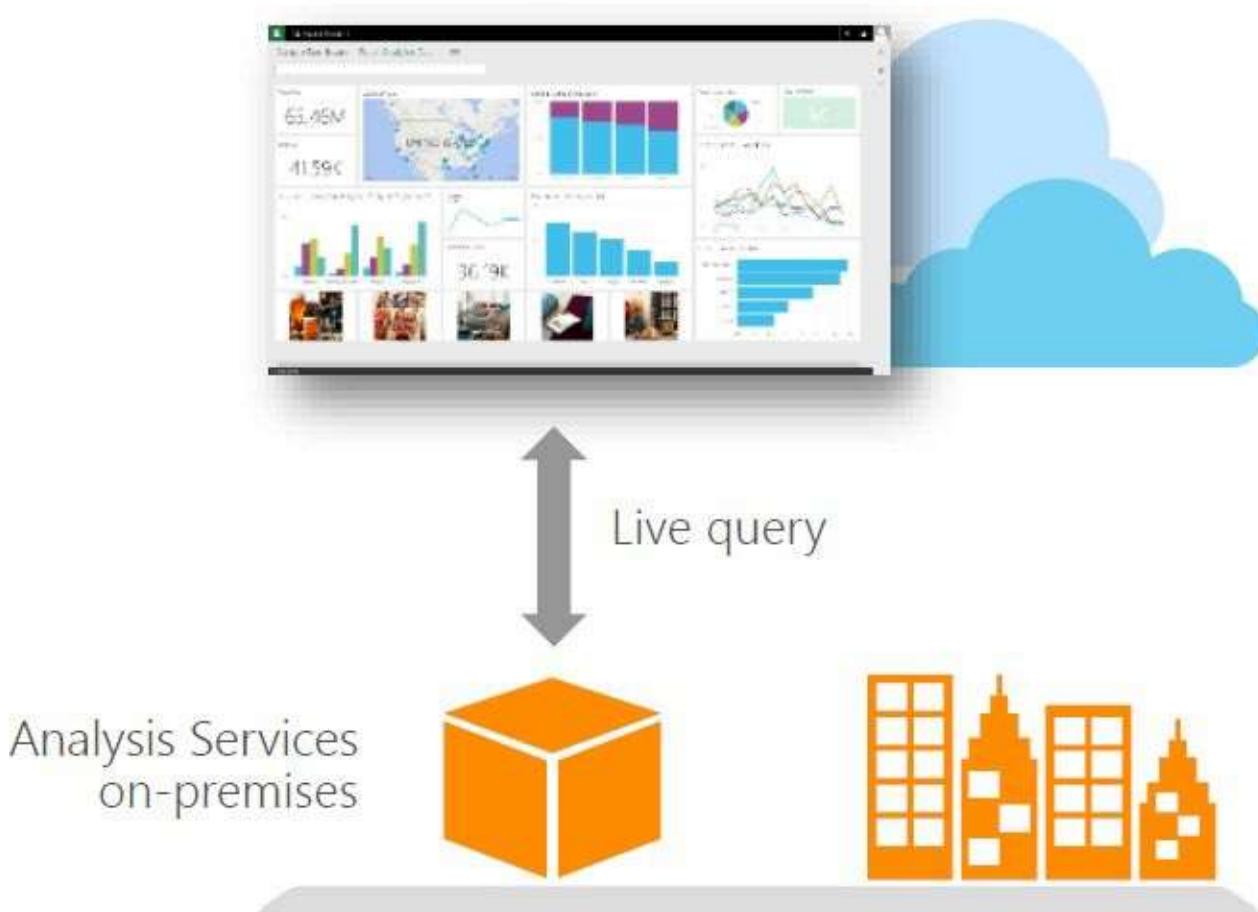


Figure 10.3 Power BI can connect to on-premises Analysis Services models.

Maya logs in to Power BI and uses Get Data to connect to Analysis Services. Maya selects the Adventure Works model. Power BI establishes a live connection. Now Maya can create reports and dashboards, as I demonstrated in Chapter 2. If the model is configured for data security, Maya can only see the data that the model security authorizes her to access. When she explores the data, Power BI auto-generates queries and sends them to the Adventure Works model. The model sends results back. No data is hosted on Power BI!

10.1.2 Understanding Setup Requirements

Because Power BI needs to connect to the on-premises model and pass it the user identity, IT needs to take care of certain prerequisites for both on premises and in the cloud. The exact setup steps depend on how your company is set up for Microsoft Azure.

Checking your domain setup

Remember that Power BI adds registered users to Azure Active Directory. In general, both the users who will access the on-premises Analysis Services model and the SSAS server, need to be on the same domain. Let's say the Adventure Works corporate domain is `adventureworks.com` and the SSAS model is installed on a server that's joined to `adventureworks.com`. If Adventure Works has federated the `adventureworks.com` domain to Azure, then no additional configuration is required!

If Adventure Works hasn't federated its domain to Azure, users can still connect to Analysis Services as long as they register and sign in to Power BI using their work emails that use the same domain as the Analysis Services server. So if Maya signs in to Power BI as maya@adventureworks.com, then Maya can connect to the Analysis Services server. What if Elena wants to install Analysis Services on an Azure virtual machine that doesn't belong to adventureworks.com? Elena has several options:

1. She can install Active Directory on that server and create a standalone adventureworks.com domain. As long as the domain names of emails and the server match, she's fine.
2. If there are other machines that require domain memberships, Elena can set up a pair of Azure virtual machine to implement a redundant domain controller.
3. Elena can use Azure Active Directory Domain Services (currently in preview) to join the VM to a domain. Pricing depends on your Active Directory, but if you have a small number of objects, Azure Active Directory Domain Services may be only \$37 per month.

NOTE Behind the scenes, Power BI appends a special EffectiveUserName connection setting when connects to an on-premises SSAS. If Maya connects to the Analysis Services server, then EffectiveUserName will pass Maya's email, such as EffectiveUserName=maya@adventureworks.com. The passwords don't need to match.

Dealing with multiple domains

What if some users are on one domain while others are on a different domain? For example, Adventure Works might have acquired Acme and the Acme employees might still be on the acme.com domain. This will cause an issue when these employees attempt to connect to Analysis Services and the connection will fail. You can see the error if you use SQL Server Profiler to monitor connections to Analysis Services. Specifically, if the server is joined to a domain, the error will read "The following system error occurred: The user name or password is incorrect." but if the server isn't on a domain, then your error may read "The following system error occurred: The name provided isn't a properly formed account name." To fix this issue, your system administrator can set an alternative UPN suffix for the Acme domain in Active Directory as follows:

1. Open the Active Directory Domains and Trust tool.
1. Right-click the top node and then click Properties.
2. In the UPN Suffixes tab, enter the other domain, as shown in **Figure 10.4**.

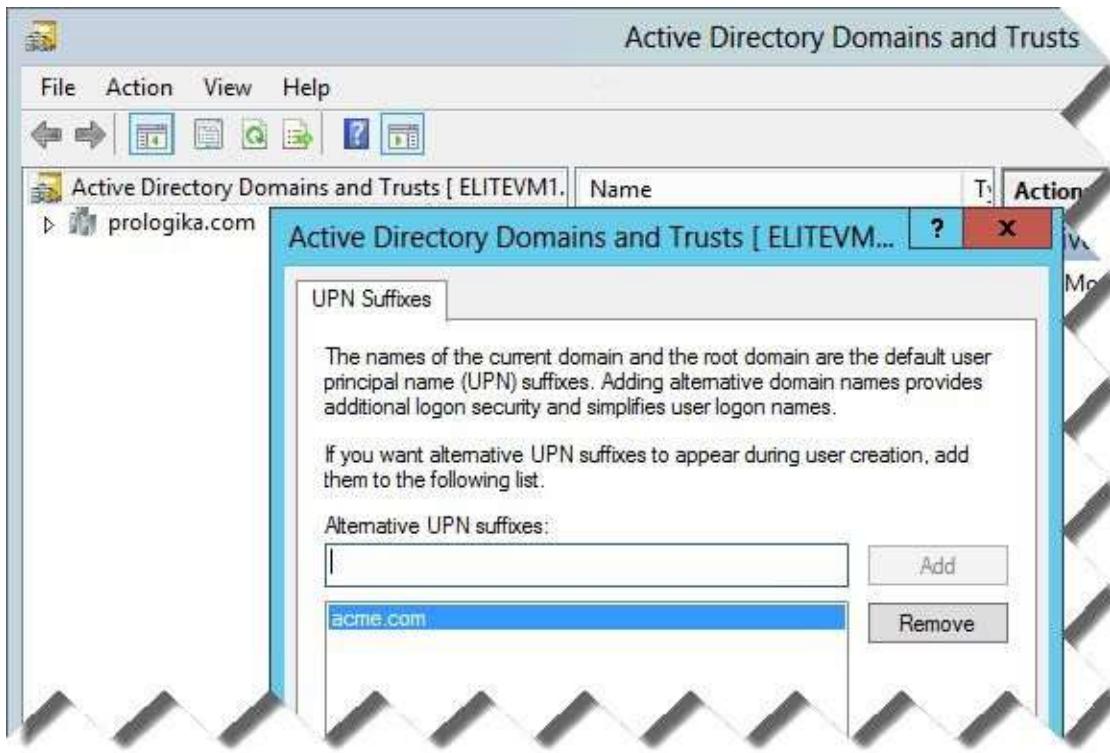


Figure 10.4 The system administrator can specify an alternative UPN suffix in Active Directory so that users on another domain can connect to the SSAS server.

Installing Analysis Services Connector

After you verify the domain setup, the next step is to set up the Analysis Services Connector. Installing and configuring a connector is usually done by an administrator. It requires special knowledge of your Analysis Services servers and it requires Server Administrator permissions to the Analysis Services instance.

NOTE As I discussed in the previous chapter, IT can use the Enterprise Power BI Gateway (a preview build was released in December 2015) to centralize data access and security. Once the Enterprise Power BI Gateway supports SSAS connectivity (according to Microsoft this feature won't take long), I recommend it because it will allow you to centrally manage connections to multiple SSAS instances. The Analysis Services Connector supports only one SSAS instance.

The Analysis Services Connector acts as a bridge between Power BI and SSAS. Power BI connects to the Analysis Services Connector, which in turns connects to the on-premises Analysis Services model. This is conceptually similar to how Power BI uses the Personal Gateway to refresh datasets from on-premises data sources. However, there are important differences, which I summarize in Table 10.2.

Table 10.2 This table highlights the differences between Analysis Services Connector and Personal Gateway.

Feature	Analysis Services Connector	Personal Gateway
Purpose	Live connections to on-premises Analysis Services	Data refresh from on-premises data sources
Data sources	Analysis Services	All refreshable data sources
Service account	Any account that has admin access to SSAS	Defaults to your account; used to access sources with Windows security
Initial setup	Performed by a server administrator	Performed by a business user or administrator

Currently, you can't install the Analysis Services Connector and Personal Gateway on the same machine. In addition, each install instance of the Analysis Services Connector is

associated with a specific SSAS instance. So if you need to connect to multiple SSAS instances, you need to install multiple connectors on different computers. Typically, you'd install the connector on the SSAS server to avoid Windows security and firewall issues, but it can be hosted on another machine as long as the machine has connectivity to the SSAS server. The setup steps are as follows:

1. Remotely access the SSAS server (or another machine that has access to the SSAS instance).
 - 1.Log in to Power BI. Expand the Downloads menu  and click “Power BI gateways”.
 - 2.Scroll all the way down the next page, and click “Analysis Services Connector”. Download and run the setup program.
 - 3.Once the setup program installs the connector, it's ask you to launch and configure the connector. This starts a wizard that helps you configure it. First, it'll ask you to sign in to Power BI.

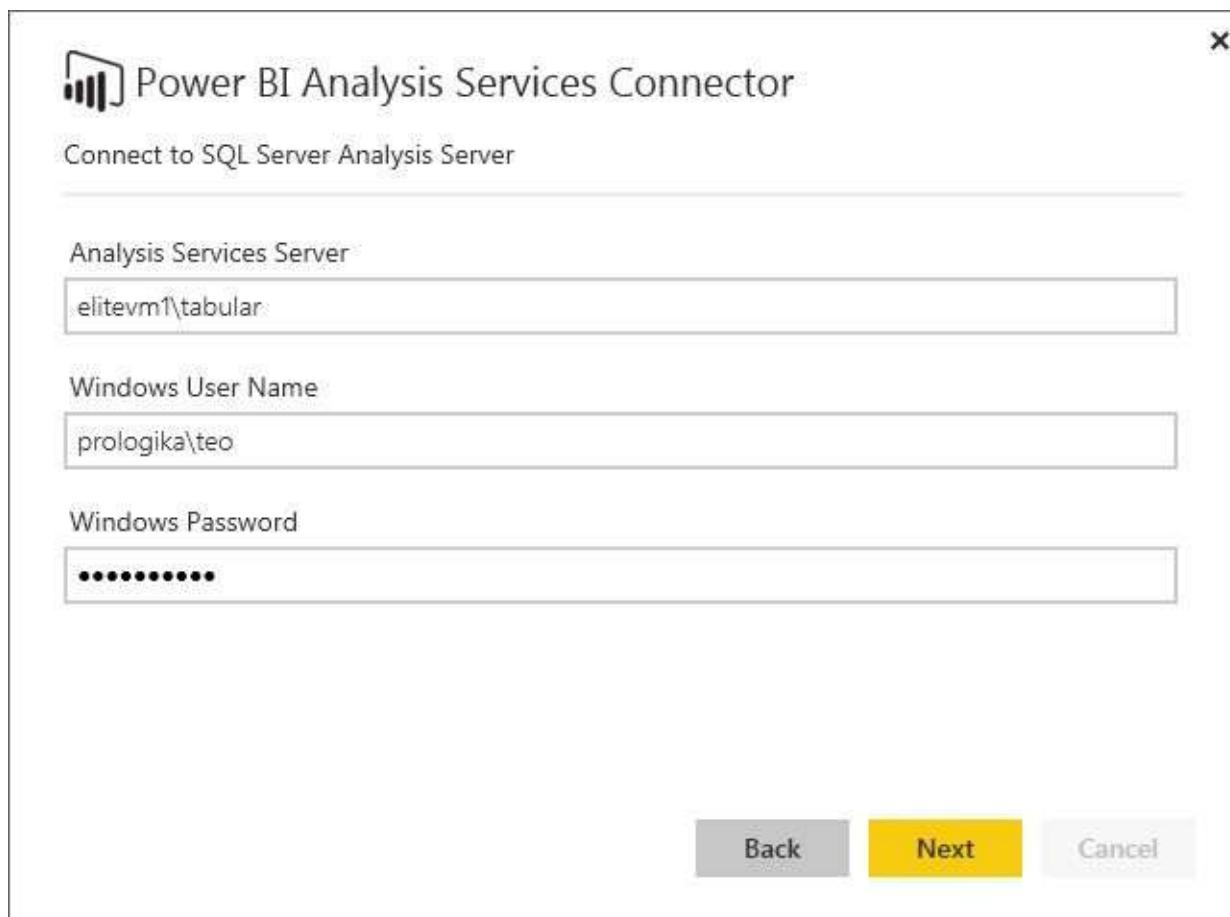


Figure 10.5 When configuring the connector, you must specify an account that has admin rights to the SSAS instance.

- 4.The next step is to establish a connection to Analysis Services (see **Figure 10.5**). In the Analysis Services Server field, enter the SSAS instance that you want to connect to. Next, specify a Windows account that has *administrator* rights to that SSAS instance. It must have admin rights so that it can impersonate the user successfully using the EffectiveUserName mechanism. When you click Next, the connector will attempt to connect to the instance using the configuration details you specify.

NOTE To verify or grant the account admin rights to SSAS, open SQL Server Management Studio (SSMS) and

connect to your SSAS instance. Right-click on the instance, and then click Properties. In the Security tab of the Analysis Services Properties page, add the account to the Server Administrator list.

5.If all goes well, the next step will ask you to enter a friendly name and description of the connector. This is what the user will see when he uses Power BI to connect to Analysis Services. You can also specify a friendly error message to inform the user when the connection fails, such as “We couldn’t connect right now to Analysis Services. Contact helpdesk@adventureworks.com.”

That’s all it takes to configure the connector! You can make subsequent configuration changes by opening the Power BI Analysis Services Connector application.

10.1.3 Connecting Users to Analysis Services

Once the Analysis Services Connector is installed and configured, users can go to Power BI and connect live to Analysis Services. However, users might not have the rights to access the Analysis Services models.

Granting user access

To grant users access, you need an Analysis Services database role. For the purposes of this exercise, you’ll use SQL Server Management Studio (SSMS) to add users to a role. Let’s grant user access to the Adventure Works Tabular model:

TIP As a best practice, the BI developer should use SQL Server Data Tools (SSDT) to define role membership in the Analysis Services project, instead of using SSMS. This way the role membership becomes a part of the project and can be deployed together with the project. But we’re using SSMS here for the sake of simplicity.

- 1.Open SQL Server Management Studio (SSMS) and connect to the Analysis Services instance.
- 2.In the Object Explorer in the left pane, expand the Analysis Services instance, and then expand the Databases node.
- 3.Expand the “AdventureWorks Tabular Model SQL 2012” database, and then expand the Roles folder.
- 4.The Adventure Works SSAS database includes an Analysts role that grants access to the model. For the purposes of this exercise, you’ll use this role. Double-click the Analysts role.
- 5.In the Role Properties window (see **Figure 10.6**), select the Membership tab and add the users.

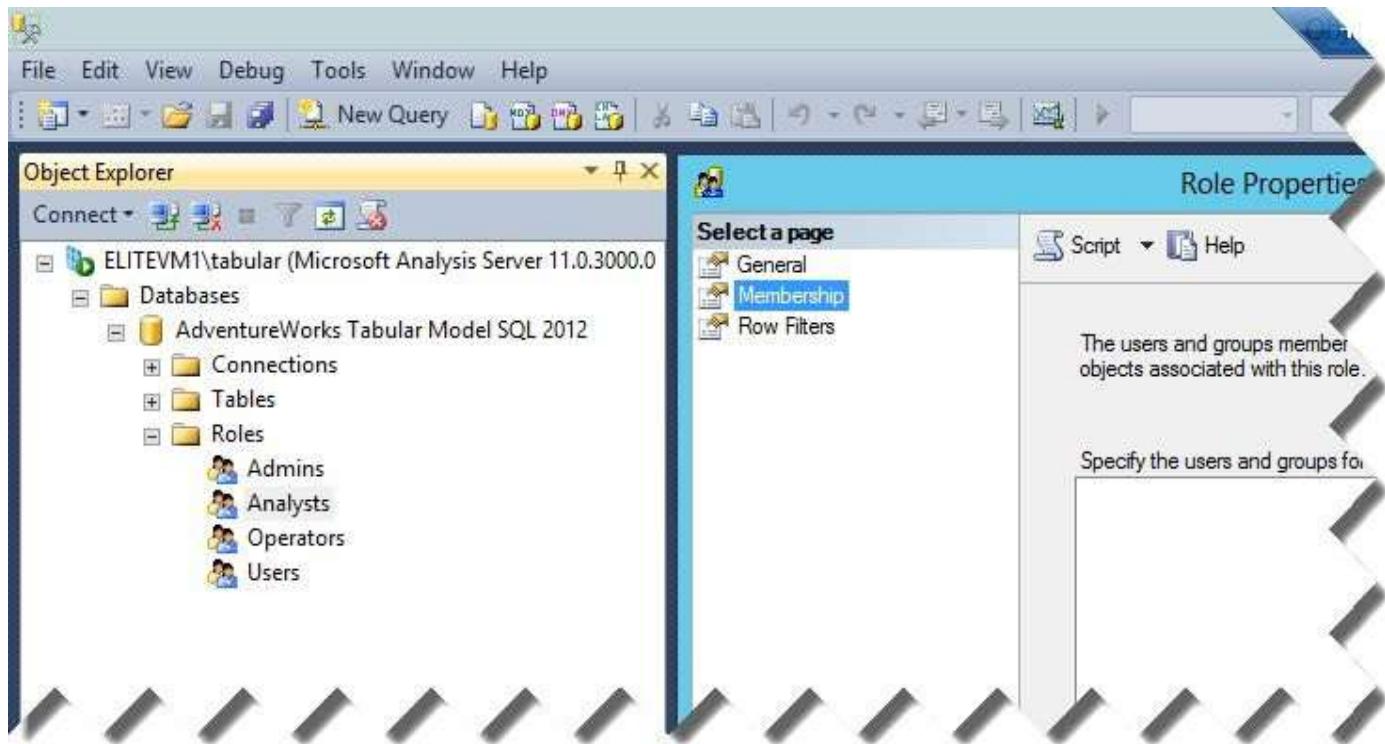


Figure 10.6 You must grant users access to the SSAS model by assigning users to a database role.

Verifying user connectivity

Once you grant the users access to the model, they should be able to connect from Power BI Service. I showed you how they do this back in Chapter 2. From an administrator standpoint, you need to know how to troubleshoot connectivity issues.

When the user goes to Power BI Service and connects to SSAS (Get Data → Databases → SQL Server Analysis Services), they'll see all the registered connectors (see **Figure 10.7**), even though they might not have rights to access any of the databases on some of the SSAS instances. Power BI Service will make the actual connection only when they select a gateway and click Connect. At this point, the user will only see the SSAS databases they are allowed to access on that instance.

SQL Server Analysis Services

Servers

Search

Name	Description	Server Name
AZVM-SSAS-MSTR_SSAS_TAB	tabular instance running on AZVM-SSAS-MSTR.byobi.local	AZVM-SSAS-MSTR.byobi
ELITEVM1 TABULAR	Adventure Works Tabular Model	ELITEVM1\TA
SSAS_TAB	SQL-DEV-01	\ssas_tab

Figure 10.7 Power BI lists all the registered connectors whether or not the user has permissions to see any of the databases hosted on that SSAS instance.

The best way to verify what identity the user connects under is to use the SQL Server Profiler connected to the SSAS instance. Many events, such as Discover Begin or Query Begin, have a PropertyList element that includes the EffectiveUserName property (see **Figure 10.8**).

SQL Server Profiler - [Untitled - 1 (EL)]

File Edit View Replay Tools Window Help

EventClass EventSubclass TextData

EventClass	EventSubclass	TextData
Session Initialize		*
Discover End	26 - DISCO...	<RestrictionList xmlns="urn:schemas-microsoft-com:xml-analysis">
Discover Begin	26 - DISCO...	<Property>DbpropMsmdCurrentActivityID</Property>
Discover End	26 - DISCO...	</RestrictionList>
Discover Begin	26 - DISCO...	<RestrictionList xmlns="urn:schemas-microsoft-com:xml-analysis">
Discover End	26 - DISCO...	<Property>DbpropMsmdCurrentActivityID</Property>
Discover Begin	26 - DISCO...	</RestrictionList>
Discover End	26 - DISCO...	<RestrictionList xmlns="urn:schemas-microsoft-com:xml-analysis">
Discover Begin	26 - DISCO...	<Property>DbpropMsmdCurrentActivityID</Property>
Session Initialize		*

```
<RestrictionList xmlns="urn:schemas-microsoft-com:xml-analysis">
    <Property>DbpropMsmdCurrentActivityID</Property>
</RestrictionList>

<PropertyList xmlns="urn:schemas-microsoft-com:xml-analysis">
    <Catalog>AdventureWorks Tabular Model SQL 2012</Catalog>
    <Cube>Model</Cube>
    <EffectiveUserName>[REDACTED]</EffectiveUserName>
    <SspropInitAppName>PowerBI</SspropInitAppName>
    <LocaleIdentifier>127</LocaleIdentifier>
    <ClientProcessID>13417</ClientProcessID>
    <OpenList>[REDACTED]</OpenList>
</PropertyList>
```

Figure 10.8 The EffectiveUserName property includes the identity of the interactive user.

The EffectiveUserName should match the work email that the user typed to log in to Power BI, such as maya@adventureworks.com. If you need data security, you can set up row filters in your Tabular model to filter the SSAS model data, based on the user identity.

10.2 Implementing Real-time BI Solutions

The growing volume of real-time data and the reduced time for decision making are driving companies to implement real-time operational intelligence systems. You might have heard the term “Internet of Things” (IoT), which refers to sensors, devices, and systems that constantly generate data. If you need to analyze this data in real-time, you need a real-time BI solution. The Microsoft Data Platform includes services and tools to help you implement solutions for real-time BI and complex event processing (CEP). Microsoft StreamInsight, which ships with SQL Server, allows you to implement on-premises real-time BI solutions. And Microsoft Azure Stream Analytics is its cloud-based counterpart.

10.2.1 Understanding Real-time BI

Unlike the classic descriptive BI architecture (which is all about analyzing the past), real-time data analytics is concerned about what happens now. For example, Adventure Works might be interested in sentiment analysis, based on customer feedback that was shared on popular social networks, such as Twitter. A descriptive approach would require you to import customer data, transform it, and then analyze it.

But what if you’re interested in what customers are saying *now*? Your implementation approach might depend on your definition of “now” and what data latency is acceptable. Perhaps, you can run ETL more often and still address your requirements with the classic BI architecture? However, if you need to analyze data as it streams, then you’ll need a stream analytics solution. Enter Microsoft Azure Stream Analytics!

Understanding Microsoft Azure Stream Analytics

Traditionally, implementing a complex event processing (CEP) solution has been difficult because...well, it’s complex, and it requires a lot of custom code. Microsoft sought to change this by introducing Azure Stream Analytics as a fully managed stream-processing service in the cloud. Stream Analytics provides low latency and real-time processing of millions of events per second in a highly resilient service (see its architecture in **Figure 10.9**).

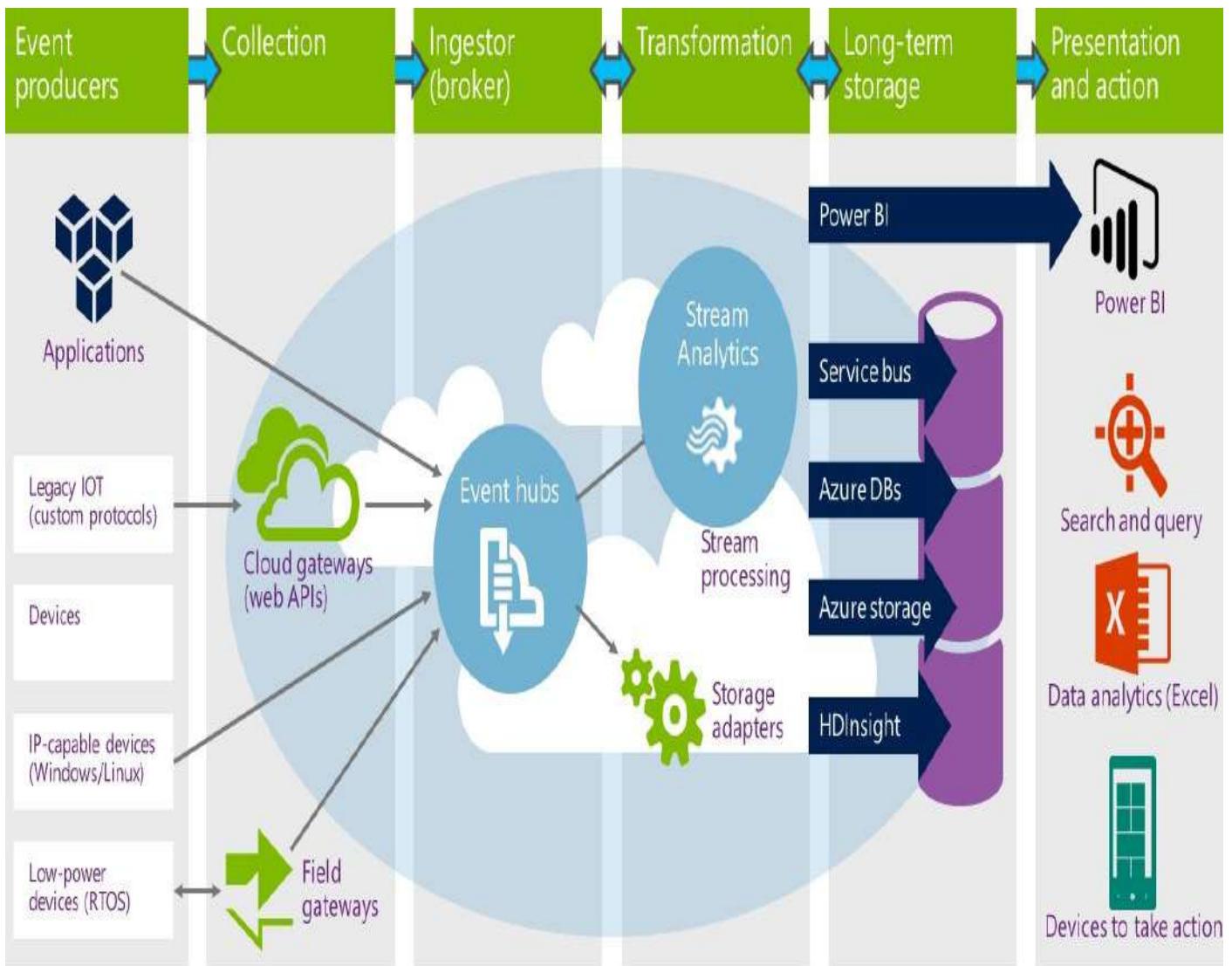


Figure 10.9 Stream Analytics is a cloud-based service and it's capable of processing millions of events per second.

Common real-time BI scenarios that will benefit from Stream Analytics include fraud detection, identity protection, real-time financial portfolio analysis, click-stream analysis, and energy smart grid utilization. Stream Analytics integrates with Azure Event Hubs, which is a highly scalable service for ingesting data streams. It enables the collection of event streams at high throughput from a diverse set of devices and services.

For example, Event Hubs is capable of processing millions of events per second via HTTP(S) or Advanced Message Queuing Protocol (AMQP) protocols. Once data is brought into Event Hubs, you can then use Stream Analytics to apply a standard SQL-like query for analyzing the data as it streams through, such as to detect anomalies or outliers. The query results can also be saved into long-term storage destinations, such as Azure SQL Database, HDInsight, or Azure Storage, and then you can analyze that data.

Currently in preview, another interesting scenario is configuring Stream Analytics to output query results directly to Power BI! This allows you to implement a real-time dashboard that updates itself constantly when Stream Analytics sends new rows. Behind the scenes, this scenario uses the Power BI Dataset Push APIs that I'll discuss in Chapter 11.

About Cortana Analytics Suite

While on the subject of the Stream Analytics Service, you should know that it's a part of a much broader vision that Microsoft has for building intelligent applications. During the 2015 World Partner Conference, Microsoft announced Cortana Analytics Suite – a cloud-based data analytics platform (see **Figure 10.10**).

Cortana Analytics Suite was built on years of Microsoft's research in perceptual intelligence, including speech recognition, natural user interaction, and predictive analytics. The key benefit is that over time, Cortana Analytics will let you roll out prepackaged analytics solutions, reducing time to market and project costs over do-it-all-yourself approaches. For example, there will be prepackaged solutions for Sales and Marketing (customer acquisition, cross-sell, upsell, loyalty programs, and marketing mix optimization), Finance and Risk (fraud detection and credit risk management), Customer Relationships Management (lifetime customer value, personalized offers, and product recommendation), and Operations and Workspace (operational efficiency, smart buildings, predictive maintenance, and supply chain).

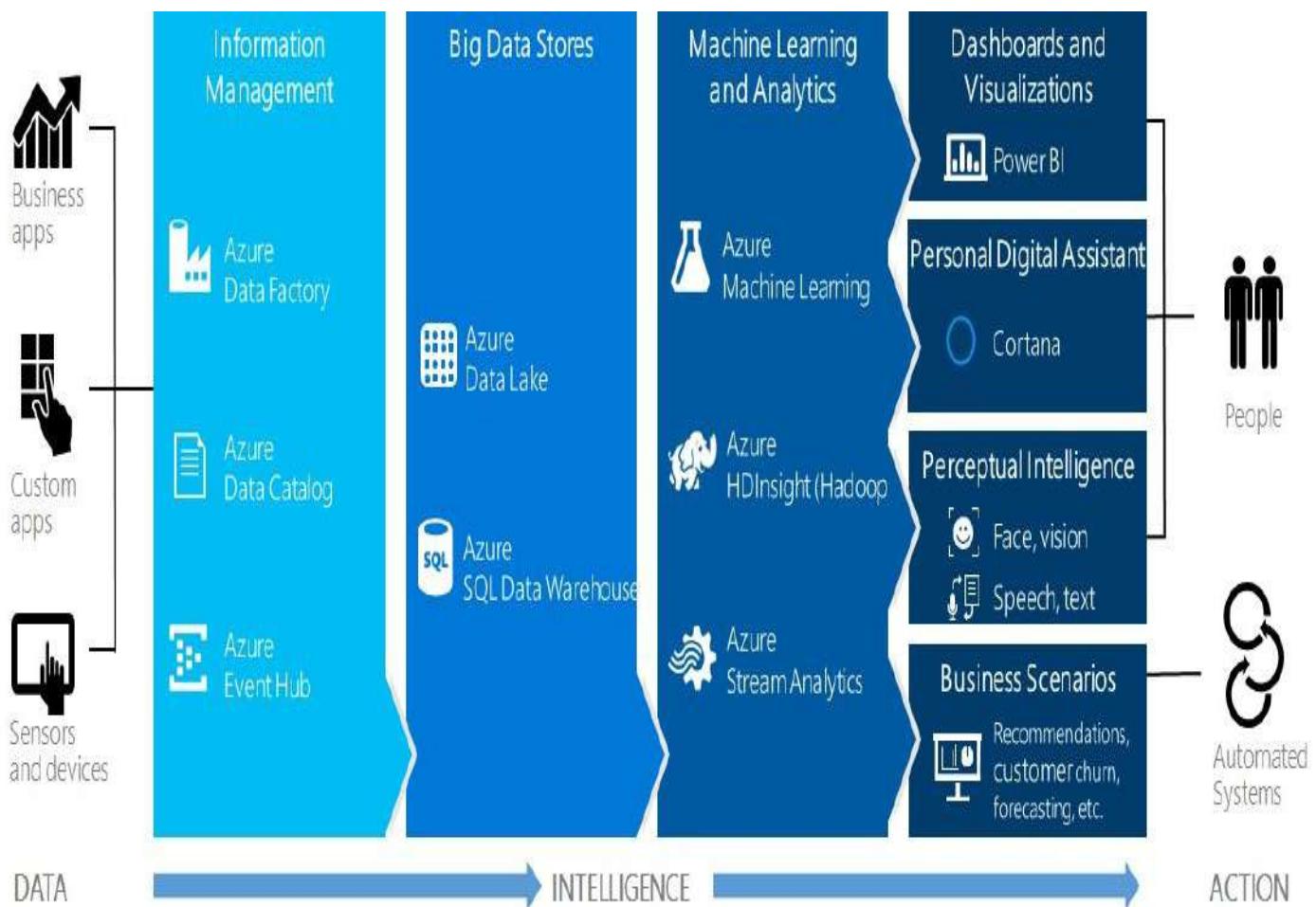


Figure 10.10 Cortana Analytics Suite is a set of tools and services for building intelligence applications.

Cortana Analytics Suite provides services to bring data in so that you can analyze it. For example, you can use Azure Data Factory (a cloud ETL service) so that you can pull data from any source (both relational and non-relational data sources), in an automated and scheduled way, while performing the necessary data transforms. As I mentioned, Event Hubs ingests data streams. The incoming data can be persisted in Big Data storage

services, such as Data Lake and Azure SQL Data Warehouse.

You can then use a wide range of analytics services from Azure Machine Learning and Stream Analytics to analyze the data that are stored in Big Data storage. This means you can create analytics services and models that are specific to your business needs, such as real time-demand forecasting. The resulting analytics services and models that you create by taking these steps, can then be surfaced as interactive dashboards and visualizations powered by Power BI.

These same analytics services and models can also be integrated with various applications (web, mobile, or rich-client applications), as well as via integrations with Cortana Personal Digital Assistant (demonstrated in Chapter 3). This way, end users can naturally interact with them via speech. For example, end users can get proactively be notified by Cortana if the analytics model finds a new data anomaly, or whatever deserves the attention of the business users. For more information about Cortana Analytics Suite, visit <http://www.microsoft.com/en-us/server-cloud/cortana-analytics-suite/overview.aspx>.

10.2.2 Integrating Stream Analytics and Power BI

Now that you've learned about real-time BI, let me walk you through a sample solution that demonstrates how Azure Stream Analytics and Power BI can help you implement real-time dashboards. Suppose that Adventure Works is interested in analyzing customer sentiment from messages that are posted on Twitter. This is immediate feedback from its customer base, which can help the company improve its products and services. And so Adventure Works wants to monitor the average customer sentiment about specific topics in real time. **Figure 11.11** shows you the process flow diagram.

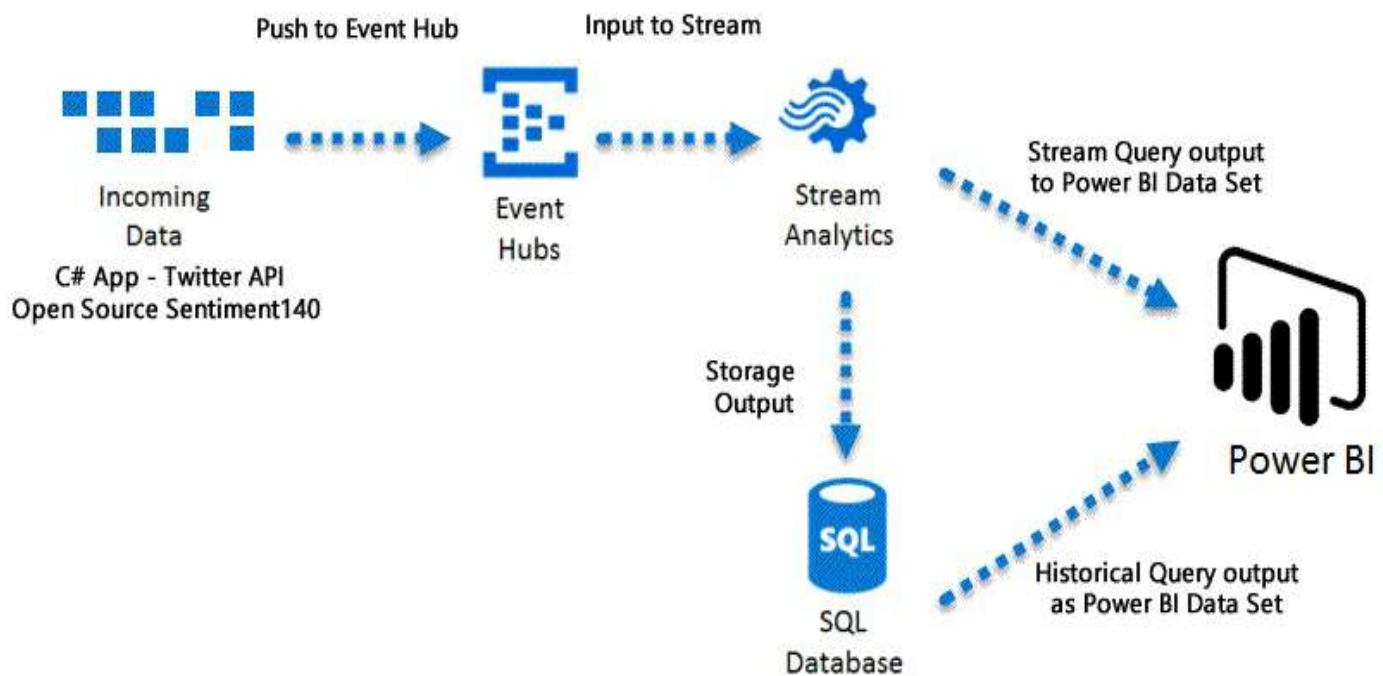


Figure 10.11 This solution demonstrates how you can integrate Stream Analytics with Power BI.

Instead of building the entire solution from scratch, I decided to use the Real-time Twitter

sentiment analysis sample by Microsoft. You can download the code from GitHub (<https://github.com/Azure/azure-stream-analytics/tree/master/DataGenerators/TwitterClient>) and the read the documentation from <https://github.com/Azure/azure-content/blob/master/articles/stream-analytics/stream-analytics-twitter-sentiment-analysis-trends.md>. You can also read the documentation online at <https://azure.microsoft.com/en-us/documentation/articles/stream-analytics-twitter-sentiment-analysis-trends>.

NOTE This sample demonstrates how remarkably simple it is to implement real-time cloud solutions with Stream Analytics. You only need to write custom code to send events to Events Hub. By contrast, a similar StreamInsight-based application would require much more coding on your part, as the Big Data Twitter Demo (<http://twitterbigdata.codeplex.com>) demonstrates. That's because you'd need to write the plumbing code for observers, adapters, sinks, and more.

Understanding the client application

Designed as a C# console application, the client app uses the Twitter APIs to filter tweets for specific keywords that you specify in the app.config file. To personalize the demo for our fictitious bike manufacturer, (Adventure Works), I used the keywords “Bike” and “Adventure”. In the same file, you must specify the Twitter OAuth settings that you obtain when you register a custom application with Twitter. For more information about registering an application with Twitter and about obtaining the security settings, read the “Tokens from dev.twitter.com” topic at <https://dev.twitter.com/oauth/overview/application-owner-access-tokens>.

Note that the client app (as coded by Microsoft) doesn't have any error handling. If you don't configure it correctly, it won't show any output and won't give you any indication what's wrong. To avoid this and to get the actual error, I recommend that you re-throw errors in every catch block in the EventHubObserver.cs file.

```
catch (Exception ex)
{
    throw ex;
}
```

The application integrates with an open source tool (Sentiment140) to assign a sentiment value to each tweet (0: negative, 2: neutral, 4: positive). Then the tweet events are sent to the Azure Event Hubs. Therefore, to test the application successfully, you must first set up an event hub and configure Stream Analytics. If all is well, the application shows the stream of tweets in the console window as they're sent to the event hub.

Understanding Stream Analytics setup

The documentation that accompanies the sample provides step-by-step instructions to configure the Azure part of the solution. You can perform the steps using old Azure portal (<https://manage.windowsazure.com>) or the new Azure portal (<http://portal.azure.com>). Instead of reiterating the steps, I'll just emphasize a few points that might not be immediately clear:

- 1.Before setting up a new Stream Analytics job, you must create an event hub that ingests that data stream.

- 2.After you create the hub, you need to copy the connection information from the hub registration page and paste it in the EventHubConnectionString setting in the client application app.config file. This is how the client application connects to the event hub.
- 3.When you set up the Stream Analytics job, you can use sample data to test the standing query. You must have run the client application before this step so that the event hub has some data in it. In addition, make sure that the date range you specify for sampling actually returns tweet events.

4.This is the standing query that I used for the Power BI dashboard:

```
SELECT System.Timestamp as Time, Topic, COUNT(*), AVG(SentimentScore), MIN(SentimentScore),  
Max(SentimentScore), STDEV(SentimentScore)  
FROM TwitterStream TIMESTAMP BY CreatedAt  
GROUP BY TUMBLINGWINDOW(s, 5), Topic
```

This query divides the time in intervals of five seconds. A tumbling window is one of the windows types that's supported by both Stream Analytics and SQL Server StreamInsight (read about at <https://msdn.microsoft.com/en-us/library/azure/dn8350110.aspx>). Within each interval, the query groups the incoming events by topic and calculates the event count, minimum, maximum, average sentiment score, and the standard deviation. Because stream analytics queries are described in a SQL-like grammar, you can leverage your SQL query skills.



NOTE Unlike SQL SELECT queries, which execute once, Stream Analytics queries are standing. To understand this, imagine that the stream of events passes through the query. As long the Stream Analytics job is active, the query is active and it's always working. In this case, the query divides the stream in five-second intervals and calculates the aggregates on the fly.

Understanding Power BI setup

A Stream Analytics job can have multiple outputs, such as to save the query results to a durable storage for offline analysis and to display them on a real-time dashboard. **Figure 10.12** shows the available outputs that Stream Analytics currently supports. Power BI is just one of these outputs. Follow these steps to configure Stream Analytics to send the output to Power BI.

1. In the “Add an output to your job” step, select Power BI.
- 1.In the “Authorize Connection” step, sign in to Power BI.

ADD AN OUTPUT

x

Add an output to your job

- SQL Database ?
- Blob storage ?
- Event Hub ?
- Power BI PREVIEW ?
- Table storage ?
- Service Bus Queue ?
- Service Bus Topic ?
- DocumentDB ?

Missing an output sink? [Let us know!](#) (Powered by [UserVoice](#) - [Privacy Policy](#))



2

Figure 10.12 Stream Analytics supports different types of outputs for sending query results.

2.In the “Microsoft Power BI Settings” step, name the output and specify the name of the Power BI dataset and table names (see **Figure 10.13**). You can send results from different queries to separate tables within the same dataset. If the table with the same name exists, it’ll be overwritten.

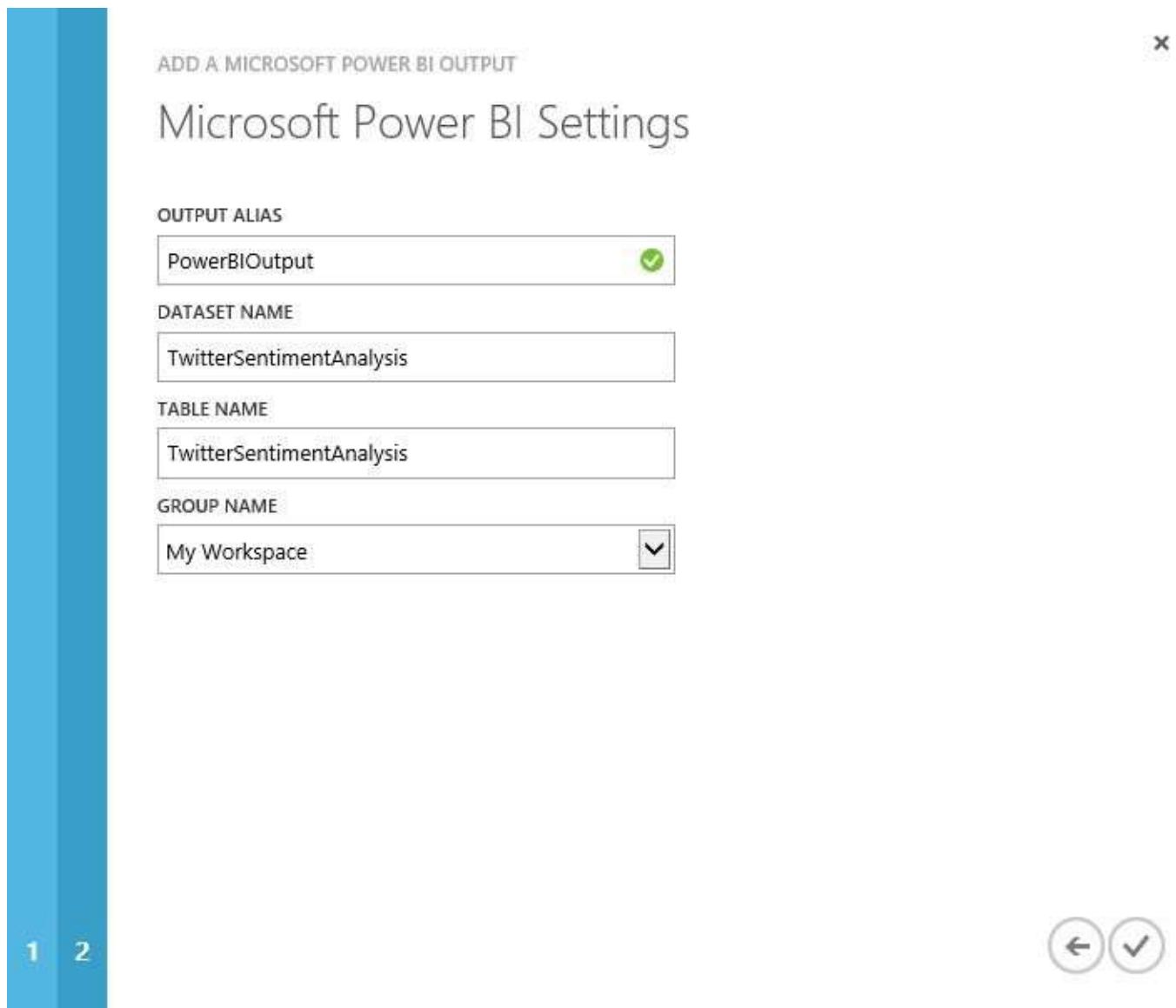


Figure 10.13 Configure which dataset and table the query results will be sent to.

Creating a real-time dashboard

Now that all the setup steps are behind us, we're ready to have fun with the data:

1. Run the Stream Analytics job. It might take a while for the job to initiate, so monitor its progress on the Stream Analytics dashboard page.
1. Once the job is started, it'll send the query results to Power BI, assuming there are incoming events that match the query criteria. Power BI will create a dataset with the name you specified.
2. Now you log in to Power BI Service and explore the dataset.
3. To show the data in real time, you need to create a dashboard by pinning the report visualization. The dashboard tile in **Figure 10.14** is based on a report visualization that uses a Combo Chart. It shows the count of events as columns and the average sentiment score as a line. Time is measured on the shared axis.

Real-time Sentiment Analysis



Share Dashboard

Ask a question about the data on this dashboard

[How to ask](#)

Avg Sentiment Score by Count

topic ● Bike ● avg

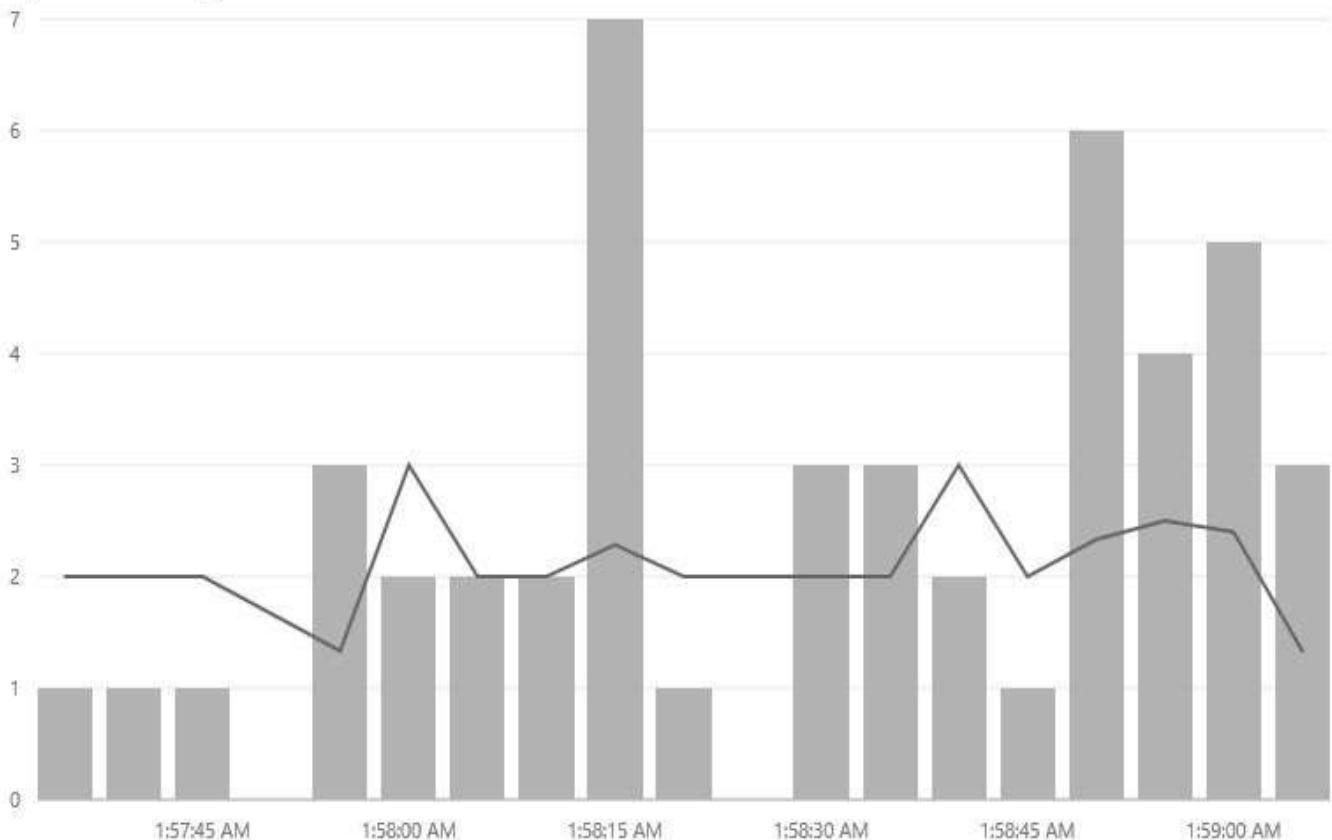


Figure 10.14 Power BI generates a real-time dashboard when Stream Analytics Service streams the events.

4. Watch the dashboard updates itself as new data is coming in, and you gain real-time insights!

10.3 Implementing Predictive Solutions

Predictive analytics (based on data mining and machine learning algorithms) is another popular requirement. It also happens to be one of the least understood because it's usually confused with slicing and dicing data. However, predictive analytics is about discovering patterns that aren't easily discernible. These hidden patterns can't be discovered with traditional data exploration, because data relationships might be too complex or because there's too much data for a human to analyze.

Typical data mining tasks include forecasting, customer profiling, and basket analysis. Data mining can answer questions, such as, "What are the forecasted sales numbers for the next few months?", "What other products is a customer likely to buy along with the product he or she already chose?", and "What type of customer (described in terms of gender, age group, income, and so on) is likely to buy a given product?"

10.3.1 Understanding Azure Machine Learning

Predictive analytics isn't new to Microsoft. Microsoft extended Analysis Services with data mining back in SQL Server 2000. Besides allowing BI pros to create data mining models with Analysis Services, Microsoft also introduced the Excel Data Mining add-in to let business users perform data mining tasks in Excel.

Introducing Azure Machine Learning

In 2014, Microsoft unveiled a cloud-based service for predictive analytics called Azure Machine Learning or AzureML (previously known as project Passau), which also originated from Microsoft Research. In a nutshell, Azure Machine Learning makes it easy for data scientists to quickly create and deploy predictive models. The Azure ML value proposition includes the following appealing features:

- It provides an easy-to-use, comprehensive, and scalable platform without requiring hardware or software investments.
- It supports workflows for transforming and moving data. For example, AzureML supports custom code, such as R or Python code, to transform columns. Furthermore, it allows you to chain tasks, such as to load data, create a model, and then save the predictive results.
- It allows you to easily expose predictive models as REST web services, so that you can incorporate predictive features in custom applications.

Figure 10.15 shows a typical AzureML implementation flow.

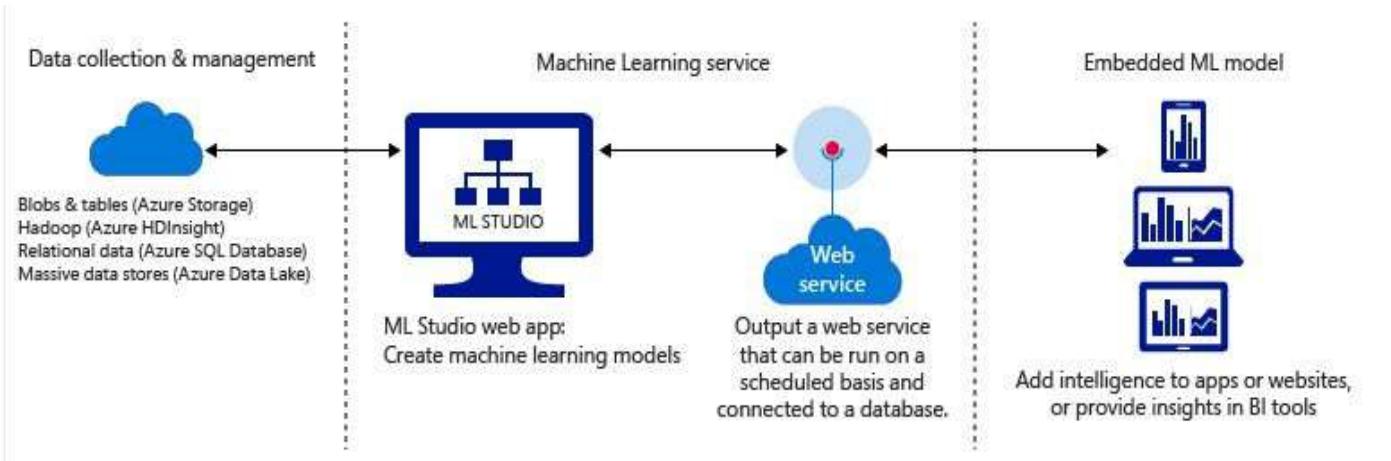


Figure 10.15 The diagram demonstrates a common flow to implement predictive models with AzureML.

Understanding workflows

Because it's a cloud service, AzureML can obtain the input data directly from other cloud services, such as Azure tables, Azure SQL Database, or Azure HDInsight. Alternatively, you can export on-premises data as a file and upload the dataset to AzureML.

Then you use ML Studio to build a workflow for creating and training a predictive model. ML Studio is a browser-based tool that allows you to drag, drop, and connect the building blocks of your solution. You can choose from a large library of Machine Learning algorithms to jump-start your predictive models! You can also extend the workflow with your own custom R and Python scripts.

Similar to Stream Analytics and Power BI, AzureML is an integral component of the Cortana Analytics Suite, although you can also use it as a standalone service. The Cortana Analytics Gallery (<http://gallery.cortanaanalytics.com>) is a community-driven site for discovering and sharing predictive solutions. It features many ready-to-go predictive models that have been contributed by Microsoft and the analytics community. You can learn more about Azure Machine Learning at its official site (<https://azure.microsoft.com/en-us/services/machine-learning>).

10.3.2 Creating Predictive Models

Next, I'll walk you through the steps to implement an AzureML predictive model for a familiar scenario. Adventure Works is planning a marketing campaign. The marketing department approaches you to help them target a subset of potential customers who are the most likely to purchase a bike. You need to create a predictive model that predicts the purchase probability, based on past history.

Understanding the historical dataset

To start, you'd need to obtain an order history dataset that you'd use to train the model:

- 1.In SQL Server Management Studio (SSMS), connect to the AdventureWorksDW2012 database and execute the following query:

SELECT

```
Gender, YearlyIncome, TotalChildren, NumberChildrenAtHome,
```

```
EnglishEducation,EnglishOccupation,HouseOwnerFlag,  
NumberCarsOwned,CommuteDistance,Region,Age,BikeBuyer  
FROM [dbo].[vTargetMail]
```

This query returns a subset of columns from the vTargetMail view that Microsoft uses to demonstrate the Analysis Services data mining features. The last column, BikeBuyer, is a flag that indicates if the customer has purchased a bike. The model will use the rest of the columns as an input to determine the most important criteria that influences a customer to purchase a bike.



NOTE The Adventure Works Analysis Services cube includes a Targeted Mailing data mining structure, which uses this dataset. The mining models in the Targeted Mailing structure demonstrate how the same scenario can be addressed with an on-premises Analysis Services solution.

2.Export the results to a CSV file. For your convenience, I included a Bike Buyers.csv file in the \source\ch09 folder.

Creating a predictive model

Next, you'll use AzureML to create a predictive model, using the Bike Buyers.csv file as an input:

1.Go to <https://azure.microsoft.com/en-us/services/machine-learning> and sign in. If you don't have an Azure subscription, you can start a trial subscription, or just use the free version!

2.Once you open ML Studio, click the New button on the bottom of the screen, and then create a dataset by uploading the Bike Buyers.csv file. *Name the dataset Bike Buyers.*

3.Click the New button again and create a new experiment. An Azure ML experiment represents the workflow to create and train a predictive model (see **Figure 10.16**).

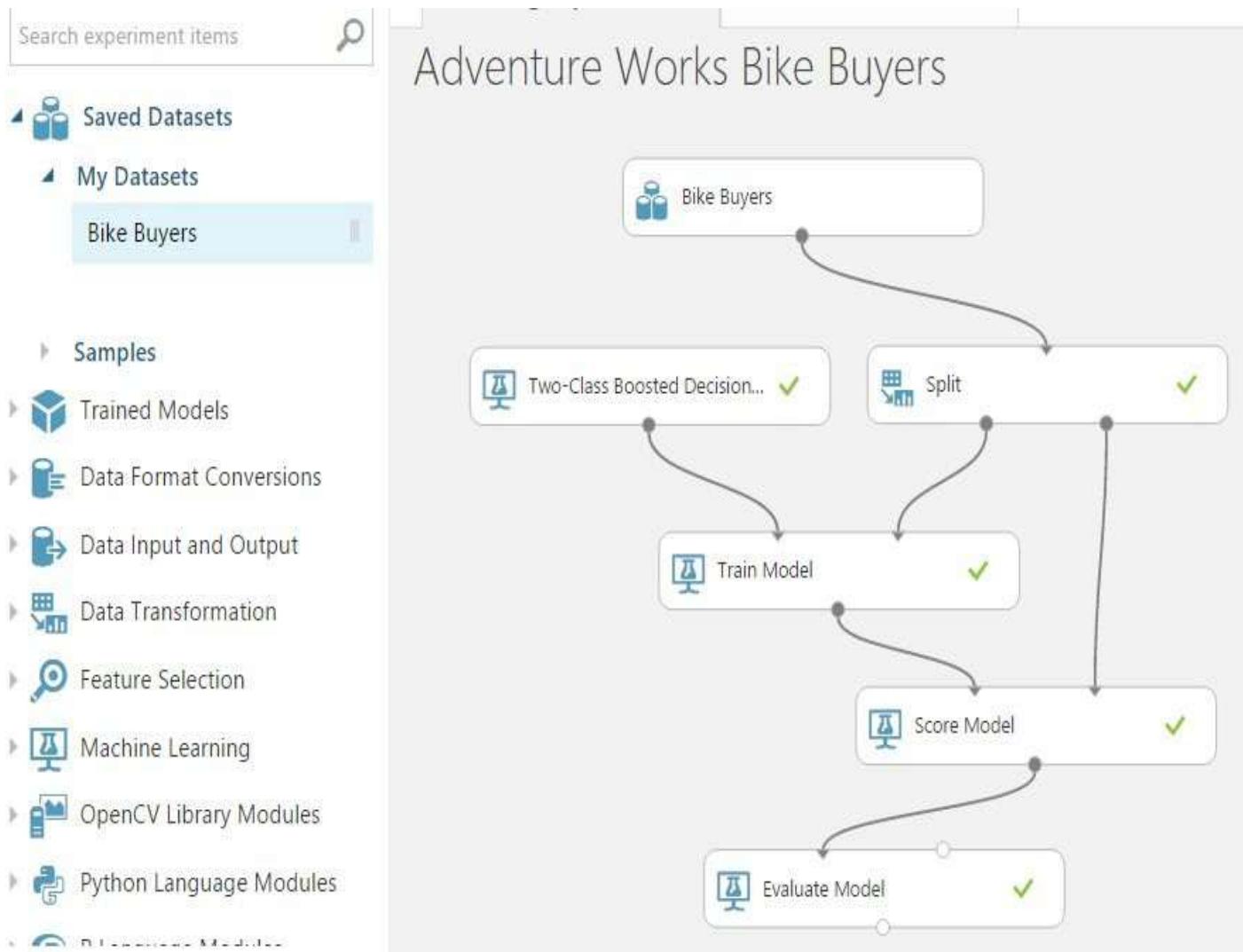


Figure 10.16 The Adventure Works Bike Buyer predictive model.

- 4.From the Saved Datasets section in the navigation bar, drag and drop the Bike Buyers dataset.
- 5.Using the search box, search for each of the workflow nodes shown in **Figure 10.16** by typing its name, and then drop them on the workflow. Join them as shown in **Figure 10.16**. For example, to find the Split task, type “Split” in the search box. From the search results, drag the Split task and drop it onto the workflow. Then connect the Bike Buyer dataset to the Split task.
- 6.The workhorse of the model is the Two-Class Boosted Decision Tree algorithm which generates the predictive results. Click the Split transformation and configure its “Fraction of rows in the first output dataset” property for a 0.8 split. That’s because you’ll use 80% of the input dataset to train the model and the remaining 20% to evaluate the model accuracy.

Setting up a web service

A great AzureML feature is that it can easily expose an experiment as a REST web service. The web service allows client applications to integrate with AzureML and use the model for scoring. Setting up a web service is remarkably easy.

1. Click the Run button to run the experiment.
- 1.(Optional) Right-click the output (the small circle at the bottom) of the Evaluate Model

task, and then click Visualize to analyze the model accuracy. For example, the Lift graph shows the gain of the model, compared to just targeting all the customers without using a predictive model.

2. At the bottom of the screen, click “Set up Web Service”. AzureML creates a new experiment because it needs to remove the unnecessary tasks from the web service, such as the task to train the model. **Figure 10.17** shows the workflow of the predictive experiment after AzureML sets up the web service.

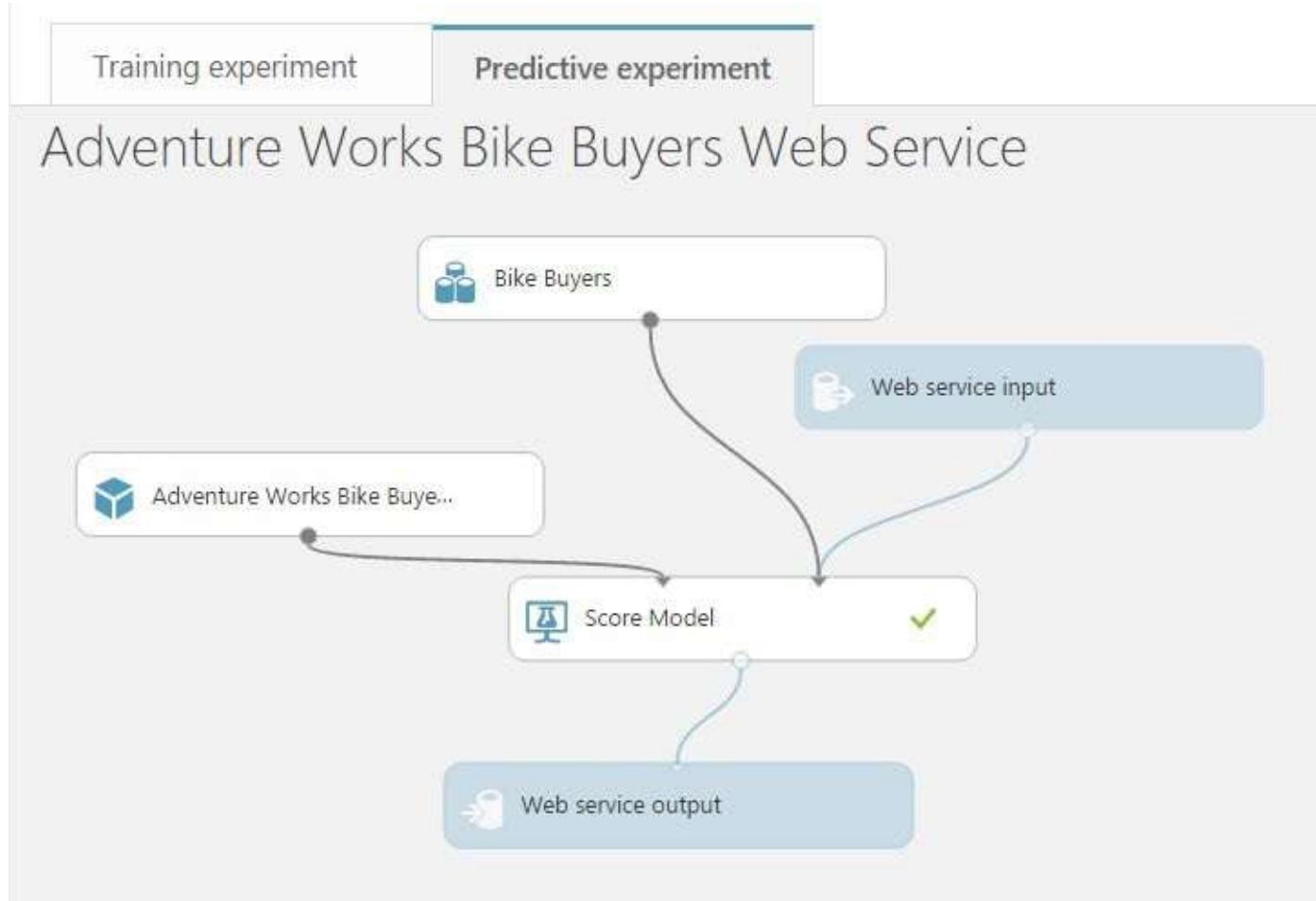


Figure 10.17 The predictive experiment is optimized for scoring.

3. Run the predictive experiment.

4. At the bottom of the screen, click “Deploy Web Service” to create the REST API endpoint.

5. In the navigation bar, click Web Services, and then click the new web service that you just created.

6. AzureML opens the web service page (see **Figure 10.18**). Your application will use the API key to authenticate against ML. Copy the API key because you’ll need it later.

7. Click the Request/Response link to see the web service developer documentation and to test the web service. The documentation page includes the web service endpoint, which should look like this following example (I excluded the sensitive information):

<https://ussouthcentral.services.azureml.net/workspaces/<...>/execute?api-version=2.0&details=true>

NOTE AzureML upgraded the web service API to version 2.0. Many of the samples on the Internet use the previous

APIs. If you want to see what the old API details were, click the “Previous version of this API” link.

adventure works bike buyers web service

DASHBOARD CONFIGURATION

General

Published experiment

[View snapshot](#) [View latest](#)

Description

No description provided for this web service.

API key



Default Endpoint

[API HELP PAGE](#)

TEST

[REQUEST/RESPONSE](#)

Test

[BATCH EXECUTION](#)

Figure 10.18 The web service page allows you to obtain the API key and test the web service.

8.Click the Request/Response link to see the web service developer documentation and to test the web service. The documentation page includes the web service endpoint, which should look like this following example (I excluded the sensitive information):

<https://ussouthcentral.services.azureml.net/workspaces/<...>/execute?api-version=2.0&details=true>

NOTE AzureML upgraded the web service API to version 2.0. Many of the samples on the Internet use the previous APIs. If you want to see what the old API details were, click the “Previous version of this API” link.

The page also shows a simple POST request and response described in JSON (see **Figure 10.19**).

Sample Request

```
{  
  "Inputs": {  
    "input1": {  
      "ColumnNames": [  
        "Gender",  
        "YearlyIncome",  
        "TotalChildren",  
        "NumberChildrenAtHome",  
        "EnglishEducation",  
        "EnglishOccupation",  
        "HouseOwnerFlag",  
        "NumberCarsOwned",  
        "CommuteDistance",  
        "Region",  
        "Age",  
        "BikeBuyer"  
      ],  
      "Values": [  
        [
```

Sample Response

```
{  
  "Results": {  
    "output1": {  
      "type": "DataTable",  
      "value": {  
        "ColumnNames": [  
          "Gender",  
          "YearlyIncome",  
          "TotalChildren",  
          "NumberChildrenAtHome",  
          "EnglishEducation",  
          "EnglishOccupation",  
          "HouseOwnerFlag",  
          "NumberCarsOwned",  
          "CommuteDistance",  
          "Region",  
          "Age",  
          "BikeBuyer",  
          "Scraped_Labels"  
        ]  
      }  
    }  
  }  
}
```

Figure 10.19 This figure shows a simple request and response from the Adventure Works Bike Buyers predictive web service.

10.3.3 Integrating Machine Learning with Power BI

Currently, Power BI doesn't include a connector to connect directly to Azure Machine Learning. This leaves you with two integration options:

- Instead of publishing to a web service, save the predictive results to Azure Storage, such as to an Azure SQL Database. Then use Power BI Service's Get Data to visualize the predictive results.
- Publish the predictive model as a web service, and then use Power BI Desktop or Power Query to call the web service.

The second option allows you to implement flexible integration scenarios, such as to call the web service for each customer from a dataset you imported in Power BI Desktop. Moreover, it demonstrates some of the advanced capabilities of Power BI Desktop queries and Power Query. This is the integration option I'll demonstrate next.

EMailAddress	Gender	YearlyIncome	TotalChildren	NumberCars	EnglishEducation	EnglishOccupation	HouseOwner	NumberCabinets	CommuteDistance	Region	Age	BikeBuyer
jon24@adventureworks.com	M	90000	2	0	Bachelors	Professional	1	0-1 Miles	Pacific	49	0	
eugene10@adventureworks.com	M	60000	3	3	Bachelors	Professional	0	1-2 Miles	Pacific	50	0	
ruben35@adventureworks.com	M	60000	3	3	Bachelors	Professional	1	2-5 Miles	Pacific	50	0	
christy12@adventureworks.com	F	70000	0	0	Bachelors	Professional	0	5-10 Miles	Pacific	47	0	
elizabeth5@adventureworks.com	F	80000	5	5	Bachelors	Professional	1	1-2 Miles	Pacific	47	0	
julio1@adventureworks.com	M	70000	0	0	Bachelors	Professional	1	5-10 Miles	Pacific	50	0	
janet9@adventureworks.com	F	70000	0	0	Bachelors	Professional	1	5-10 Miles	Pacific	49	0	
marco14@adventureworks.com	M	60000	3	3	Bachelors	Professional	1	0-1 Miles	Pacific	51	0	
rob4@adventureworks.com	F	60000	4	4	Bachelors	Professional	1	10+ Miles	Pacific	51	0	
shannon38@adventureworks.com	M	70000	0	0	Bachelors	Professional	0	5-10 Miles	Pacific	51	0	
jacquelyn20@adventureworks.com	F	70000	0	0	Bachelors	Professional	0	5-10 Miles	Pacific	51	0	
curtis9@adventureworks.com	M	60000	4	4	Bachelors	Professional	1	10+ Miles	Pacific	51	0	
lauren41@adventureworks.com	F	100000	2	0	Bachelors	Management	1	1-2 Miles	North America	47	0	
ian47@adventureworks.com	M	100000	2	0	Bachelors	Management	1	0-1 Miles	North America	47	0	
sydney23@adventureworks.com	F	100000	3	0	Bachelors	Management	0	1-2 Miles	North America	47	0	
chloe23@adventureworks.com	F	30000	0	0	Partial College	Skilled Manual	0	5-10 Miles	North America	36	0	
wyatt32@adventureworks.com	M	30000	0	0	Partial College	Skilled Manual	1	5-10 Miles	North America	36	0	
shannon1@adventureworks.com	F	20000	4	0	High School	Skilled Manual	1	5-10 Miles	Pacific	71	0	
clarence32@adventureworks.com	M	30000	2	0	Partial College	Clerical	1	5-10 Miles	Pacific	71	0	
luke18@adventureworks.com	M	40000	0	0	High School	Skilled Manual	0	5-10 Miles	Europe	37	0	

Figure 10.20 The New Customers Excel file has a list of customers that need to be scored.

Understanding the input dataset

Let's get back to our Adventure Works scenario. Now that the predictive web service is ready, you can use it to predict the probability of new customers who could purchase a bike, provided that you have the customer demographics details. The marketing department has given you an Excel file with potential customers which they might have downloaded from the company's CRM system.

- 1.In Excel, open the New Customers.xlsx file, which includes 20 new customers (see **Figure 10.20**).
- 2.Note that the last column BikeBuyer is always zero. That's because at this point you don't know if the customer is a potential bike buyer.

The predictive web service will calculate the probability for each customer to purchase a bike. This requires calling the web service for each row in the input dataset by sending a predictive query (a data mining query that predicts a single case is called a singleton query). To see the final solution, open the Predict Buyers.pbix file in Power BI Desktop.

Creating a query function

You already know from Chapter 6 how to use query functions. Similar to other programming languages, a query function encapsulates common logic so that it can be called repeatedly. Next, you'll create a query function that will invoke the Adventure Works predictive web service:

1. In Power BI Desktop, click Edit Queries to open the Query Editor.
- 1.In Query Editor, expand the New Source button and click Blank Query. Then in the

ribbon's View tab, click Advanced Editor.

2.Copy the function code from the \source\ch10\fnPredictBuyer.txt file and paste in the query. The query function code follows:

```
1. let
2. PredictBikeBuyer = (Gender, YearlyIncome, TotalChildren, NumberChildrenAtHome, EnglishEducation,
   EnglishOccupation, HouseOwnerFlag, NumberCarsOwned, CommuteDistance, Region, Age, BikeBuyer) =>
3. let
4. //replace service Uri and serviceKey with your own settings
5. serviceUri="https://ussouthcentral.services.azureml.net/workspaces/.../execute?api-
version=2.0&details=true",
6. serviceKey=<your service key>
7. RequestBody =
8. {
9.   "Inputs": {
10.     "input1": {
11.       "ColumnNames": [
12.         "Gender", "YearlyIncome", "TotalChildren", "NumberChildrenAtHome", "EnglishEducation",
           "EnglishOccupation",
13.         "HouseOwnerFlag", "NumberCarsOwned", "CommuteDistance", "Region", "Age",
           "BikeBuyer"
14.       ],
15.       "Values": [
16.         [
17.           "&Gender&", "&Text.From(YearlyIncome)&", "&Text.From(TotalChildren)&",
           "&Text.From(NumberChildrenAtHome)&", "&EnglishEducation&", "&EnglishOccupation&",
           "&Text.From(HouseOwnerFlag)&", "&Text.From(NumberCarsOwned)&",
           "&Text.From(CommuteDistance)&",
18.           "&Region&", "&Text.From(Age)&", "&Text.From(BikeBuyer)&"
19.         ]]}
20.     },
21.   }
22.   ,
23.   Source=Web.Contents(serviceUri,
24. [Content=Text.ToBinary(RequestBody),
25. Headers=[Authorization="Bearer "&serviceKey,#"Content-Type"="application/json; charset=utf-8"]]),
26. #“Response” = Json.Document(Source),
27. #“Results” = Record.ToTable(#“Response”)
28. in
29. #“Results”
30. in
31. PredictBikeBuyer
```

The code in line 2 defines a PredictBikeBuyer function with 12 arguments that correspond to the columns in the input dataset. Remember to update lines 5 and 6 with your web

service URI and service key. Then the code creates a request payload described in JSON, as per the web service specification (see **Figure 10.19** again). Notice that the code obtains the actual values from the function arguments. Lines 23-25 invoke the web service. Line 26 reads the response, which is also described in JSON. Line 27 converts the response to a table.

3.Rename the query to *fnPredictBuyer* and save it.

Creating a query

Now that you have the query function, let's create a query that will load the input dataset from the New Customers Excel file, and then call the function for each customer:

1. In Query Editor, expand New Source again and then click Excel.

1.Navigate to New Customers.xlsx and load Table1. Table1 is the Excel table that has the new customers.

2.Rename the query to *PredictedBuyers*.

3.Add a custom column called PredictedScore, as shown in **Figure 10.21**. This column calls the fnPredictBuyer function and passes the customer demographic details from each row in the input dataset.

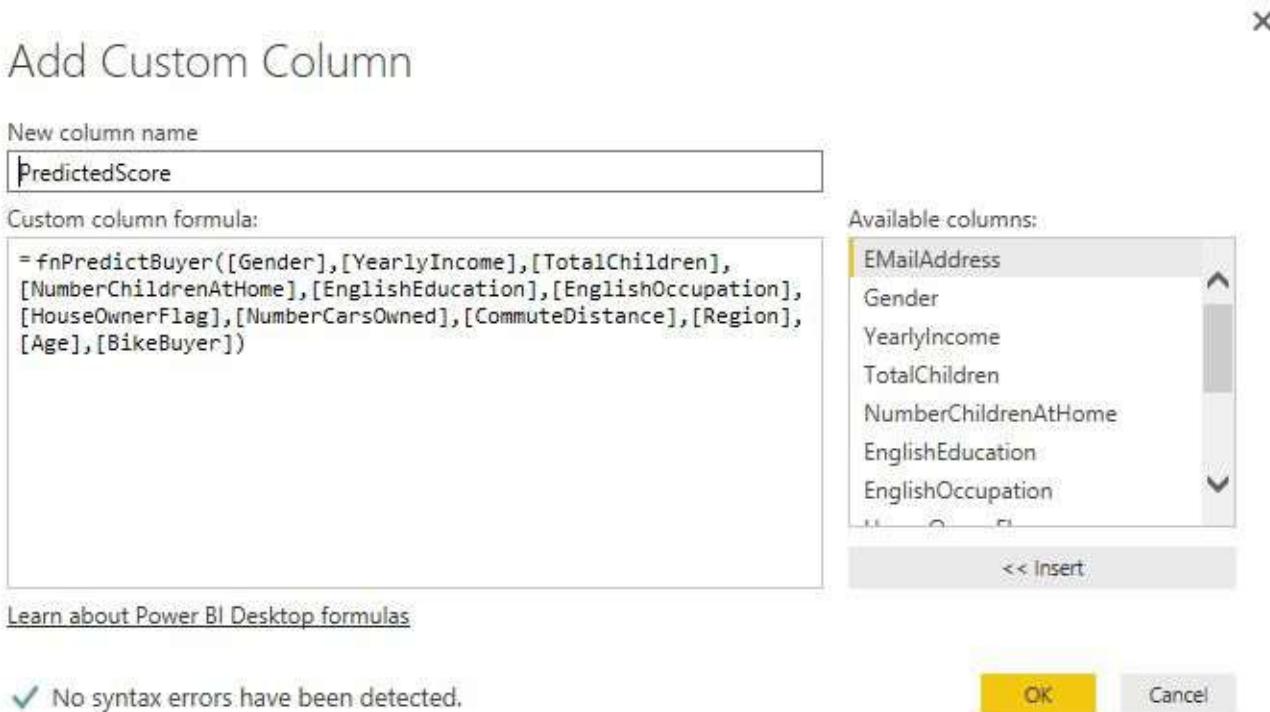


Figure 10.21 The PredictedScore custom column calls the fnPredictBuyer function.

4.The results of the custom column will be a table, which includes other nested tables, as per the JSON response specification (see **Figure 10.22**). You need to click the “expand” button four times until you see “List” showing in the PredictedScore column.

Region	Age	BikeBuyer	PredictedScore
Pacific	49	0	Table
Pacific	50	0	Table
Pacific	50	0	Table
Pacific	47	0	Table
Pacific	47	0	Table
Pacific	50	0	Table

Figure 10.22 Expand the PredictedScore column four times until you get to the list of values.

5. Once you get to the list, add another custom column to get the last value of the list. Because you have 12 input columns, the last value will be always at position 13. This value represents the score:
- ```
[PredictedScore.Value.output1.value.Values]{0}{13}
```
6. The final steps are to rename the column, change the column type to Decimal Number, and remove any errors.
  7. Create a table report that shows the customer e-mail and the probability for each customer to purchase a bike, as shown in **Figure 10.23**.

| EMailAddress                    | PredictedScore |
|---------------------------------|----------------|
| jon24@adventure-works.com       | 0.97           |
| julio1@adventure-works.com      | 0.94           |
| janet9@adventure-works.com      | 0.94           |
| clarence32@adventure-works.com  | 0.88           |
| lauren41@adventure-works.com    | 0.71           |
| christy12@adventure-works.com   | 0.48           |
| ian47@adventure-works.com       | 0.40           |
| jacquelyn20@adventure-works.com | 0.32           |

**Figure 10.23** A table report that shows the predicted results.

## 10.4 Summary

As a BI pro, you can meet more demanding business requirements and implement versatile solutions with Power BI and Azure cloud services. You can preserve the investments you've made in classic BI by integrating Power BI with Analysis Services. This allows you to implement a hybrid scenario, where data remains on premises, and Power BI reports and dashboards connect live to Analysis Services.

If you need to implement real-time solutions that analyze data streams, you'll save significant effort by using the Azure Stream Analytics Service. It's capable of ingesting millions of events per second. You can create a standing query to analyze the data stream and send the results to a real-time Power BI dashboard.

Your Power BI solutions can do more than descriptive analytics. If you need to predict future outcomes, you can implement on-premises or cloud-based predictive models. Azure Machine Learning lets business users and professionals build experiments in the cloud. You can save the predictive results to Azure, or you can publish the experiment as a predictive REST web service. Then Power BI Desktop and Power Query can call the predictive web service so that you can create "smart" reports and dashboards that transcend traditional data slicing and dicing!

Thanks to its open architecture, Power BI supports other integration scenarios that application developers will find very appealing, as you'll see in the next part of this book.



## *Power BI for Developers*

One of Power BI's most prominent strengths is its open APIs, which will be very appealing to developers. When Microsoft architected the Power BI APIs, they decided to embrace popular industry standards and formats, such as REST, JSON, and OAuth2. This allows any application on any platform to integrate with the Power BI APIs. This part of the book shows developers how they can extend Power BI and implement custom solutions that integrate with Power BI!

I'll start by laying out the necessary programming fundamentals for programming with Power BI. Once I introduce you to the REST APIs, I'll walk you through the Power BI developer site, which is a great resource to learn and test the APIs. I'll explain how Power BI security works. Then I'll walk you through a sample application that uses the REST APIs to programmatically retrieve Power BI content and to load datasets with application data.

Chances are that your organization is looking for ways to embed insightful BI content on internal and external applications. Thanks to the embedded APIs, Power BI supports this scenario, and I'll walk you through the implementation details of embedding dashboards and reports.

As you've likely seen, Power BI has an open visualization framework. Web developers can implement custom visuals and publish them to the Power BI visuals gallery. If you're a web developer, you'll see how you can leverage your experience with popular JavaScript-visualization frameworks, such as D3, WebGL, Canvas, or SVG, to create custom visuals that meet specific presentation requirements, and if you wish, you can contribute your custom visualizations to the community!



## Chapter 11

# Programming Fundamentals

Developers can build innovative solutions around Power BI. In the previous chapter, I've showed examples of descriptive, predictive, and real-time BI solutions that use existing Microsoft products and services. Sometimes, however, developers are tasked to extend custom applications with data analytics features, such as to implement real-time dashboards for operational analytics. Because of the open Power BI APIs, you can integrate any modern application on any platform with Power BI!

If you're new to Power BI development, this chapter is for you. I'll start by introducing you to the Power BI REST APIs. To jump start your programming journey, you'll see how to use the Power BI Developer Center to learn and test the APIs. You'll find how to register your custom applications so that they can access the Power BI APIs. I'll demystify one of the most complex programming topics surrounding Power BI programming: OAuth authentication. Finally, I'll walk you through a sample application that demonstrates how a custom application can integrate with the Power BI APIs.

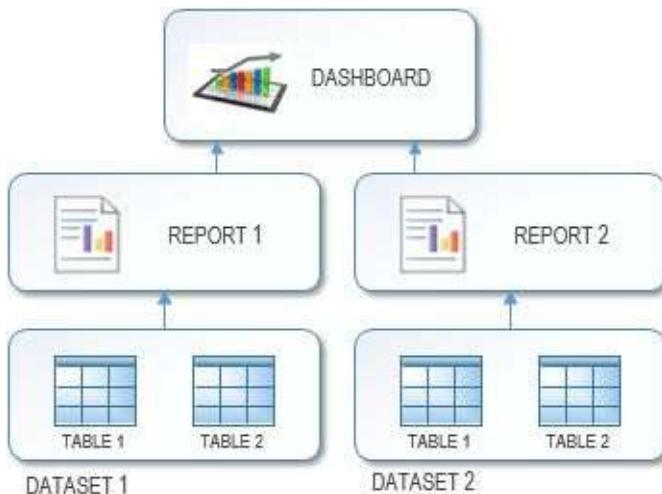
## 11.1 Understanding Power BI APIs

When the Power BI team was out to architect the programming interface, they adopted the following design principles:

- Embrace industry standards – Instead of implementing proprietary interfaces, the team looked at other popular cloud and social platforms, and decided to adopt already popular open standards, such as Representational State Transfer (REST) programming ([https://en.wikipedia.org/wiki/Representational\\_state\\_transfer](https://en.wikipedia.org/wiki/Representational_state_transfer)), OAuth authorization (<https://en.wikipedia.org/wiki/OAuth>), and JavaScript Object Notation (JSON) for describing object schemas and data (<https://en.wikipedia.org/wiki/JSON>).
- Make it cross platform – Because of the decision to use open standards, the Power BI APIs are cross platform. Any modern native or web application on any platform can integrate with Power BI.
- Make it consistent – If you have experience with the above-mentioned industry specifications, you can seamlessly transfer your knowledge to Power BI programming. Consistency also applies within Power BI so that the APIs reflect the same objects as the ones the user interacts with in the Power BI portal.
- Easy to get started – Nobody wants to read tons of documentation before writing a single line of code. To help you get up to speed, the Power BI team created a Power BI Developer Center site where you can try the APIs as you read about them!

### 11.1.1 Understanding Object Definitions

As I've just mentioned, one of the design goals was consistency within Power BI. To understand this better, let's revisit the three main Power BI objects: *datasets*, *reports*, and *dashboards*. Please refer to the diagram shown in **Figure 11.1**, which shows their relationships.



**Figure 11.1** The main Power BI objects are datasets, reports, and dashboards.

#### *Understanding dataset definitions*

As you might know by now, the dataset object is the gateway to Power BI. A dataset connects to the data and the other two main objects (reports and dashboards) get data from

datasets. Specifically, a report can reference a single dataset. A dashboard can include visualizations from multiple reports, and thus display data from multiple datasets. From a developer standpoint, the dataset object has a simple JSON definition:

```
{
 "id": "<guid>",
 "name": "<dataset name>"
}
```

As you can see, a dataset has a unique identifier of a GUID data type and a name. The JSON name property returns the dataset name you see in the Power BI portal's navigation bar.

#### ***Understanding table definitions***

In the previous chapter, I briefly touched on the fact that a dataset is just a logical container. The actual data is stored in tables. A dataset could have multiple tables, such as when you use Get Data in Power BI Service or Power BI Desktop to import multiple Excel sheets. Each table has a collection of columns. Here's what the JSON definition of a hypothetical Product table might look like (the definition is split into three columns to conserve space):

```
{
 "name": "Product",
 "columns": [
 {
 "name": "ProductID",
 "dataType": "Int64"
 },
 {
 "name": "Name",
 "dataType": "string"
 },
 {
 "name": "IsCompete",
 "dataType": "bool"
 },
 {
 "name": "ManufacturedOn",
 "dataType": "DateTime"
 },
 {
 "name": "Category",
 "dataType": "string"
 }]}
```

The Product table has five columns. Each column has a name and a data type. As far as column data types are concerned, Power BI supports a subset of the Entity Data Model (EDM) data types but with some restrictions, as shown in Table 11.1.

**Table 11.1 Power BI supports the following subset of EDM data types for table columns.**

| Data type | Purpose                    | Restrictions                                                              |
|-----------|----------------------------|---------------------------------------------------------------------------|
| Int64     | A whole integer number     | Int64.MaxValue and Int64.MinValue aren't allowed                          |
| Double    | A decimal number           | Double.MaxValue and Double.MinValue are not allowed. NaN isn't supported. |
| Boolean   | A Boolean TRUE/FALSE value |                                                                           |
| Datetime  | A datetime value           | Precision to 3.33 ms                                                      |
| String    | A text value               | Up to 128K characters                                                     |

## 11.1.2 Understanding the REST APIs

The Power BI APIs are evolving! Because of the importance of datasets, the initial set of APIs was centered around dataset manipulation. Later on, the team added group, dashboard, and report APIs, with more APIs coming in the future. In this section, you'll learn about the capabilities of the existing methods.

**NOTE** Currently, some REST APIs are in preview, and their method signatures show you this, such as "beta" in the URL: <https://api.powerbi.com/beta/myorg/dashboards>. Remember to update your code when the beta methods move to production. For example, when the "Get All Dashboards" method is production ready, its method signature needs to be changed to <https://api.powerbi.com/v1.0/myorg/dashboards>.

### *Understanding the verbs*

As I mentioned, the Power BI APIs are based on a programming specification for method invocation over HTTP, called Representational State Transfer (or REST for short). Because the same API can serve different purposes, REST supports a set of HTTP verbs to indicate the purpose of the method invocation. For example, to get the list of existing datasets, you'll use the GET verb. Power BI supports the most common HTTP verbs, as shown in Table 11.2.

**Table 11.2 Power BI supports four HTTP verbs.**

| Verb   | Operation | Success Response Codes                                          | Error Response Codes                                       |
|--------|-----------|-----------------------------------------------------------------|------------------------------------------------------------|
| GET    | Read      | 200 (OK)                                                        | 404 (Not Found) or 400 (Bad Request).                      |
| POST   | Create    | 201 (Created)                                                   | 404 (Not Found), 409 (Conflict) if resource already exists |
| PUT    | Update    | 200 (OK) (or 204 if not returning any content in response body) | 404 (Not Found) if ID is not found                         |
| DELETE | Delete    | 200 (OK)                                                        | 404 (Not Found) if ID is not found                         |

The HTTP GET method is used to retrieve a representation of a resource, such as a dataset collection. When executed successfully, GET returns the JSON representation in the response body. POST is used to create a new resource, such as a new dataset. Upon successful completion, it returns the HTTP status 201 and a Location header with a link to

the newly-created resource.

You typically use the PUT verb to update an existing resource by using the unique identifier of the resource. Upon a successful update, you'll get a response code of 200 (or 204 if the API doesn't return any content in the response body). Finally, given an identifier, DELETE removes the specified resource, such as a given row in a dataset table.

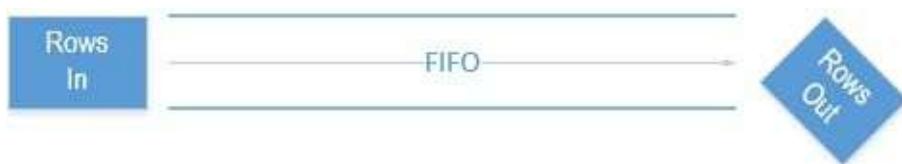
#### **Understanding the dataset methods**

The dataset-related methods are centered around the manipulation of datasets and tables, as shown in Table 11.3. Using the Power BI Rest API's, you can programmatically create new datasets, and you can add or remove rows to and from the tables in those datasets.

**Table 11.3 This table shows the dataset-related methods.**

| Purpose           | Verb   | Method Signature                                                                                                                                                   | Parameters                                  |
|-------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| List all datasets | GET    | <a href="https://api.powerbi.com/v1.0/myorg/datasets">https://api.powerbi.com/v1.0/myorg/datasets</a>                                                              |                                             |
| Create a dataset  | POST   | <a href="https://api.powerbi.com/v1.0/myorg/datasets?defaultRetentionPolicy=None">https://api.powerbi.com/v1.0/myorg/datasets?<br/>defaultRetentionPolicy=None</a> | defaultRetentionPolicy (None,<br>basicFIFO) |
| List all tables   | GET    | <a href="https://api.powerbi.com/v1.0/myorg/datasets/id/tables">https://api.powerbi.com/v1.0/myorg/datasets/id/tables</a>                                          | id (the dataset identifier)                 |
| Change table      | PUT    | <a href="https://api.powerbi.com/v1.0/myorg/datasets/id/tables/TableName">https://api.powerbi.com/v1.0/myorg/datasets/id/tables/TableName</a>                      | id (the dataset identifier),<br>TableName   |
| Add table rows    | POST   | <a href="https://api.powerbi.com/v1.0/myorg/datasets/id/tables/TableName/rows">https://api.powerbi.com/v1.0/myorg/datasets/id/tables/TableName/rows</a>            | id (the dataset identifier),<br>TableName   |
| Remove all rows   | DELETE | <a href="https://api.powerbi.com/v1.0/myorg/datasets/id/tables/TableName/rows">https://api.powerbi.com/v1.0/myorg/datasets/id/tables/TableName/rows</a>            | id (the dataset identifier),<br>TableName   |

As you can see, the verb indicates the intended action. For example, if I use the GET verb when I call the dataset API, I'll get a list of all the datasets, exactly as they're listed in the Power BI Service navigation bar under My Workspace. If I call the same API with a POST verb, I instruct Power BI to create a new dataset. In the latter case, I can specify an optional *defaultRetentionPolicy* parameter. This controls the store capacity of the dataset. The retention policy becomes important when you start adding rows to the dataset, such as to supply data to a real-time dashboard. By default, the dataset will accumulate all the rows. However, if you specify *defaultRetentionPolicy=basicFIFO*, the dataset will store up to 200,000 rows. Once this limit is reached, Power BI will start purging old data as new data comes in (see **Figure 11.2**).



**Figure 11.2** The basic FIFO dataset policy stores up to 200,000 rows and removes old data as new data flows in.

#### **Understanding the dashboard methods**

Currently in preview, the dashboard methods allow you to retrieve dashboards and tiles. This is useful if you need to embed a dashboard tile on a custom application, as I'll cover in Chapter 12. Table 11.4 shows you the dashboard methods.

**Table 11.4 This table shows the dashboard-related methods.**

| Purpose             | Verb | Method Signature                                                                                                                            | Parameters                |
|---------------------|------|---------------------------------------------------------------------------------------------------------------------------------------------|---------------------------|
| List all dashboards | GET  | <a href="https://api.powerbi.com/beta/myorg/dashboards">https://api.powerbi.com/beta/myorg/dashboards</a>                                   |                           |
| List tiles          | GET  | <a href="https://api.powerbi.com/beta/myorg/dashboards/&lt;id&gt;/tiles">https://api.powerbi.com/beta/myorg/dashboards/&lt;id&gt;/tiles</a> | id (dashboard identifier) |

Currently, there are only two methods. The first method returns a list of dashboards as they appear in the Power BI portal's navigation bar. Every dashboard element in the response body has a GUID identifier, name, and an *isReadOnly* property that indicates whether the dashboard is read-only, such as when it's shared with you by someone else, and you're given read-only access. The "List Tiles" method takes the dashboard identifier, and returns a collection of the dashboard tiles. Here's a sample JSON format that describes the "This Year's Sales" tile of the Retail Analysis Sample dashboard:

```
{
 "id": "b2e4ef32-79f5-49b3-94bc-2d228cb97703",
 "title": "This Year's Sales",
 "subTitle": "by Chain",
 "embedUrl": "https://app.powerbi.com/embed?dashboardId=<dashboard id>&tileId=b2e4ef32-79f5-49b3-94bc-2d228cb97703"
}
```

From a content embedding standpoint, the most important element is *embedUrl*, because your application needs it to reference the tile.

#### ***Understanding the report methods***

Currently in preview, the report API allows you to retrieve reports. This is useful if you need to embed a report in a custom application, as I'll cover in Chapter 12. Table 11.5 shows the "List All Reports" method that allows you to programmatically retrieve your reports.

**Table 11.5 This table shows the dashboard-related methods.**

| Purpose          | Verb | Method Signature                                                                                    | Parameters |
|------------------|------|-----------------------------------------------------------------------------------------------------|------------|
| List all reports | GET  | <a href="https://api.powerbi.com/beta/myorg/reports">https://api.powerbi.com/beta/myorg/reports</a> |            |

The "List All Reports" method returns a list of reports as they appear in the navigation bar. Every report element in the response body has a GUID-based identifier (*id*), name, *webUrl*, and *embedUrl*. The *webUrl* link navigates to the report in Power BI Service. The *embedUrl* link is used to embed the report in a custom application. Here's a sample JSON format that describes the Retail Analysis Sample report that you implemented in Chapter 3.

```
{
 "id": "22454c20-dde4-46bc-9d72-5a6ea969339e",
 "name": "Retail Analysis Sample",
 "webUrl": "https://app.powerbi.com/reports/22454c20-dde4-46bc-9d72-5a6ea969339e",
 "embedUrl": "https://app.powerbi.com/reportEmbed?reportId=22454c20-dde4-46bc-9d72-5a6ea969339e"
}
```

#### ***Understanding the group methods***

As you probably know by now, a user can access other workspaces. Therefore, to present

the user with all the objects he has access to, a custom app must enumerate not only the user’s My Workspace but also all other groups he’s a member of. Currently, Power BI defines only one group method (see Table 11.6).

**Table 11.6 This table shows the List all Groups method.**

| Purpose         | Verb | Method Signature                                                                                  | Parameters |
|-----------------|------|---------------------------------------------------------------------------------------------------|------------|
| List all groups | GET  | <a href="https://api.powerbi.com/v1.0/myorg/groups">https://api.powerbi.com/v1.0/myorg/groups</a> |            |

This method returns all groups the sign-in user belongs to. Each group element has the id and name properties. Here’s an example that returns the Sales Department and Finance groups that I belong to:

```
“value”: [
 {
 “id”: “42e3a472-2855-4a78-aec7-387d2e8722ae”,
 “name”: “Sales Department”
 },
 {
 “id”: “e6e4e5ab-2644-4115-96e0-51baa89df249”,
 “name”: “Finance”
 }
]
```

**TIP** What if you want to retrieve objects from another workspace that you’re a part of through your group membership, such as to list the datasets from the Finance workspace, or to create a new dataset that’s shared by all the members of a group? Fortunately, the dataset, dashboard, and report methods support specifying a group. For example, to list all of the dataset for a specific group, you can invoke

[https://api.PowerBI.com/v1.0/myorg/groups/{group\\_id}/datasets](https://api.PowerBI.com/v1.0/myorg/groups/{group_id}/datasets), where *group\_id* is the group unique identifier that you can obtain by calling the “List all groups” method. As it stands, the developer’s console doesn’t allow you to change the method signature, so you can’t use the group syntax in the Power BI Developer Center, but you can use this syntax in your applications.

### *Understanding the import methods*

The import APIs lets you programmatically upload a Power BI Desktop file (\*.pbix) or an Excel file to a user’s workspace. This could be useful if you want to automate the deployment process of uploading datasets and reports but remember that organizational content packs might be an easier and more flexible option to distribute content to many users. Table 11.7 shows the method signatures.

**Table 11.7 This table shows the Import File method signatures.**

| Purpose     | Verb | Method Signature                                                                                                      | Parameters                         |
|-------------|------|-----------------------------------------------------------------------------------------------------------------------|------------------------------------|
| Import File | POST | <a href="https://api.powerbi.com/beta/myorg/imports">https://api.powerbi.com/beta/myorg/imports</a>                   | datasetDisplayName<br>nameConflict |
|             | GET  | <a href="https://api.powerbi.com/beta/myorg/imports/importID">https://api.powerbi.com/beta/myorg/imports/importID</a> |                                    |

To import a file, you must create a “POST” request that takes two parameters:

- *datasetDisplayName* – The name of the resulting dataset after the import is complete.
- *nameConflict* – Specifies the behavior if the name of that dataset already exists. Specify “Abort” to fail the method, “Overwrite” to replace existing datasets and reports, and

“Ignore” (default) to ignore the conflicting names.

The request must conform to the multipart/form-data specification (<http://bit.ly/1OOwF4l>). **Figure 11.3** shows a sample request:



**Figure 11.3** A sample request body for importing a file.

The Content-Disposition header specifies “form-data” and the source file name. The Content-Type header is set to “application/octet-stream”. Next, the request body includes the entire contents of the file, byte for byte. Don’t base-64 encode it, just serialize it as a stream.

The actual import operation isn’t blocking and executes asynchronously. When you invoke the POST signature, you’ll get back a 202 accepted response. The response body will include a GUID identifier for the import task, such as:

```
{"id": "12e3a412-2855-4a78-aec7-387d2e1234ae"}
```

You can use this identifier to check the status of the import task by using the GET verb signature and passing it the *importID* identifier. You’ll get back a JSON response, such as the one shown in **Figure 11.4**.

```
{
 "id": "1db51a3d-94ea-4a43-9df2-10ab08c1922a",
 "importState": "Succeeded",
 "createdDateTime": "2015-08-10T16:44:35.087",
 "updatedDateTime": "2015-08-10T16:44:35.087",
 "reports": [
 {
 "id": "a53ad8ae-7c7f-4c28-8de4-f1c4e58f2482",
 "name": "AdventureWorks",
 "webUrl": "https://api.powerbi.com/reports/a53ad8ae-7c7f-4c28-8de4-f1c4e58f2482"
 }
],
 "datasets": [
 {
 "id": "c6ee6c94-08a4-40a1-a34a-21e20f8e189e",
 "name": "AdventureWorks",
 "tables": [
 ...
]
 }
],
 "name": "AdventureWorks"
}
```

**Figure 11.4** The response of the GET verb invocation shows the status of the operation.

Your code can inspect the *importState* property to determine if the import task has completed.

### 11.1.3 Using Power BI Developer Center

Now that you’ve learned about the Power BI REST APIs, you’re probably eager to try them out. But hold off writing code in Visual Studio. As it turns out, Microsoft created a very useful site, Power BI Developer Center, to get you started with Power BI programming!

## Getting started with Power BI Developer Center

You can access the Power BI Developer Center in three different ways:

1. Navigate to <http://powerbi.com>. Expand the Support menu and click Documentation. Scroll to the bottom of the page, and click Developers in the Information section.
2. Log in to Power BI. Expand the “Help & Support” menu and then click “Power BI for Developers”.
3. Navigate directly to <https://powerbi.microsoft.com/developers>.

The screenshot shows the Power BI Developer Center homepage. At the top, there's a yellow header bar with the text "Power BI Developer Center" and a search icon. Below the header is a large image of a person wearing headphones, looking at a computer screen. The main title "Power BI REST API" is prominently displayed. A descriptive text block below the title explains the API's purpose: "Use the Power BI REST API to push data directly from your application into a dataset in Power BI. Your dashboards will be updated in real-time when the data changes. No more waiting or having to press the Refresh button!" A yellow "Get started now" button is located on the left side of the main content area. Below the main title, there are four main sections with icons: "Client app" (monitor icon), "Web" (globe icon), "REST API and real time" (refresh/clock icon), and "Extend Power BI" (chart icon). Each section has a "Register your app" link, a "Documentation" link, and a "Sample" link. The "Try the API" link under the "REST API and real time" section is highlighted with a red oval.

**Figure 11.5** Power BI Developer Center is your one-stop resource for programming with Power BI.

The Power BI Developer Center (see **Figure 11.5**) is your one-stop resource for all your Power BI developer needs. It has links to useful resources, including developer documentation and step-by-step guides of how to accomplish specific tasks.

4. Since I'm on the subject of REST APIs, click the “Try the API” link. This opens the Power BI REST API site, where you can read the API documentation.

### Testing the APIs

A great feature of the Power BI REST API site is that it includes a testing console where you can try the APIs without writing a single line of code! Follow these steps to test the dataset APIs:

- 1.In the Power BI REST API site, click the Datasets link in the navigation page to expand the section.
- 2.Click the Datasets Collection item. This shows you the documentation of the dataset-related APIs.
- 3.Click the “List all Datasets” section. The right pane shows an example of the method call, including a sample request and a sample raw JSON response (see **Figure 11.6**).

The screenshot shows the Power BI REST API documentation interface. The top navigation bar includes the Power BI logo, 'Power BI REST API', 'Documentation', 'Inspector', and 'Apiary Powered Documentation' with a 'Sign in with Apiary account' button. The main content area is titled 'Datasets Collection'. On the left, there's a sidebar with 'INTRODUCTION' and 'REFERENCE' sections, and a list of dataset-related endpoints: 'Dashboards - Preview', 'Datasets', 'Datasets Collection', 'Tables Collection', 'Table', 'Table Rows', 'Groups', 'Create a Dataset', and 'List all Datasets'. The 'List all Datasets' section is highlighted with a red box and has a red arrow pointing to the 'Request' section on the right. The 'Request' section shows a 'GET' method call to 'https://api.powerbi.com/v1.0/myorg/datasets'. Below the request, there's a 'Response' section showing a '200' status.

**Figure 11.6** The Power BI REST API site shows an example of the method invocation.

- 4.To get a code example written in your preferred programming language, expand the drop-down menu to the left of the Try button (that says “Raw”), and then select your targeted language, such as C#, Visual Basic, JavaScript, Node.js, and others. The default Raw option shows only the request and response payloads that your application needs to invoke the API and process the results. Another good use of the Raw option is that you copy the Request code, and then use it to make direct method invocations with network tools, such as Fiddler, when you want to test or troubleshoot the APIs.
- 5.To try the API, click the Try button. This switches you to an interactive console, as

shown in **Figure 11.7**.

The screenshot shows the Power BI API console interface. On the left, there's a navigation bar with 'Datasets / Datasets Collection / List all Datasets'. Below it, a 'GET' button points to the URL <https://api.powerbi.com/v1.0/myorg/datasets>. A 'Headers' section contains a single entry: 'Add a new header'. At the bottom, there are buttons for 'Reset Values', 'Production' (with a dropdown arrow), and a prominent green 'Call Resource' button. Below these are three tabs: 'Sent' (selected), 'Compare', and 'Code Example'. On the right, the interface is divided into 'Request' and 'Response' sections. The 'Request' section shows the headers sent: 'Authorization: Bearer eyJ0-XXXXXX-XXXXXX-XXXXXX--cGw' and 'Content-Length: 0'. The 'Response' section shows the status code '200' and a list of response headers: 'connection: keep-alive', 'x-apiary-transaction-id: 5637dd8cd78117070f2afda', 'cache-control: no-store, must-revalidate, no-cache', 'transfer-encoding: chunked', 'content-type: application/json; odata.metadata=minimal; odata.streaming=true', 'server: Microsoft-HTTPAPI/2.0,Microsoft-HTTPAPI/2.0 Microsoft-HTTPAPI/2.0', 'strict-transport-security: max-age=31536000; includeSubDomains', 'x-frame-options: deny', and 'x-content-type-options: nosniff'. There are also several small decorative icons on the right side of the response pane.

**Figure 11.7** The console lets you invoke APIs and shows the request and the response.

- 6.Click the Get Resource button. If you haven't signed in to Power BI, this is when you'll be asked to do so. That's because all the API calls happen under the identity of the signed user.
- 7.After some delay, you'll get the actual request/response pair as a result of the method invocation.

If you examine the request body, you'll see an Authorization header. The obfuscated setting after "Bearer" is where the OAuth authorization token will go with an actual method invocation. The Response pane shows the result of the call. As you know already, the "200" response code means a successful invocation. Next, the Response section displays the response headers and the response body (not shown in **Figure 11.7**). The response body includes the JSON definition of the dataset collection, with all the datasets in My Workspace.

## Datasets / Tables Collection / List all Tables

GET https://api.powerbi.com/v1.0/myorg/datasets/c5d22140-ec85-4b0c-acb4-1db3b30a8c1e/tables

URI Parameters

Headers

id

c5d22140-ec85-4b0c-acb4-1db3b30a8c1e

x

Reset Values

Production

Call Resource

**Figure 11.8** The console let's you pass parameters to the method, such as the dataset identifier in this example.

### **Working with parameters**

Now let's use the console to get a list of tables in a given dataset:

1. Scroll down the response pane and copy the GUID of the id property of the Retail Analysis Sample dataset.
- 2.In the navigation pane, click the Tables Collection tab. On the Tables Collection page, click the “List all Tables” link.
- 3.In the Console pane (see **Figure 11.8**), click the URI Parameters button. This method takes an id parameter, which in this case is the dataset identifier.
- 4.Paste the dataset identifier next to the id parameter, and then click Call Resource.
- 5.If the method invocation succeeds, you should get a “200” response. Then the response body should show the JSON definition of a table collection with five tables.
- 5.(Optional) Experiment with calling other methods, such as to create a new dataset, create a new table, and add to rows to a table.

## 11.2 Understanding OAuth Authentication

I showed you how the console makes it easy to understand and test the Power BI REST APIs. However, it hides an important and somewhat complex part of Power BI programming, which is security. You can't get much further beyond the console if you don't have a good grasp of how Power BI authentication and authorization works. And security is never easy. In fact, sometimes implementing security can be more complicated than the application itself. The good news is that Power BI embraces another open security standard, OAuth2, that greatly reduces the plumbing effort to authenticate users with Power BI! As you'll see in this chapter and the next one, OAuth2 is a flexible standard that supports various security scenarios.

**REAL LIFE** I once architected a classic BI solution consisting of a data warehouse, cube, and reports, for a card processing company. After a successful internal adoption, their management decided to allow external partners to view their own data. One of the security requirements was federating account setup and maintenance to the external partners. This involved setting up an Active Directory subdomain, a web application layer, and countless meetings. At the end, the security plumbing took more effort than the actual BI system!

### 11.2.1 Understanding Authentication Flows

OAuth2 allows a custom application to access Power BI Service on the user's behalf, after the user consents that this is acceptable. Next, the application gets the authorization code from Azure AD, and then exchanges it for an access token that provides access to Power BI.

#### *Understanding the OAuth parties*

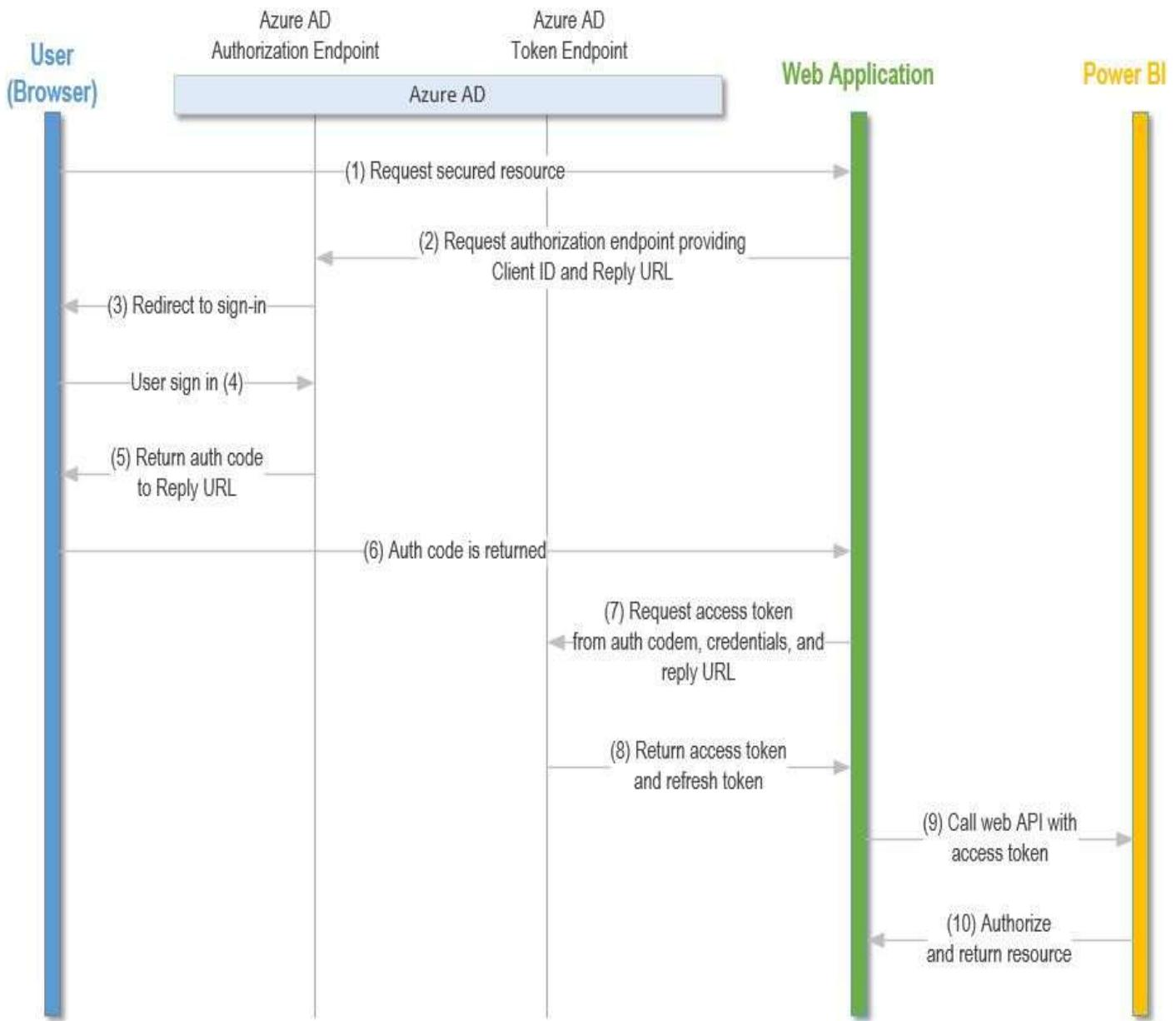
There are three parties that are involved in the default three-leg authentication flow:

- User – This is the Power BI user.
- Application – This is a custom native client application or a web application that needs to access Power BI content on behalf of the user.
- Resource – In the case of Power BI, the resource is content, such as a dataset or a dashboard, that the user has access to.

With the three-leg flow, the custom application never has the user's credentials because the entire authentication process is completely transparent to the application. However, OAuth opens a sign-in window so that the user can log in to Power BI. In the case when the application knows the user credentials, OAuth2 supports a two-leg flow where the application directly authenticates the user to the resource. The two-leg flow bypasses the sign-in window, and the user isn't involved in the authentication flow. I'll demonstrate the two-leg scenario in the next chapter.

#### *Understanding the web application flow*

**Figure 11.9** shows the OAuth flow for custom web applications. By “web application”, I mean any browser-based application, such as a custom ASP.NET application.



**Figure 11.9** This sequence diagram shows the OAuth2 flow for web applications.

The typical authentication flow involves the following steps:

- 1.The user opens the Web browser and navigates to the custom application in order to request a Power BI resource, such as to view a report.
- 2.The web application calls to and passes the application client ID and Reply URL (AD). Azure Active Directory uses the client ID to identify the custom application. You obtain the client ID when you register your application with Azure AD. The Reply URL is typically a page within your application where Azure AD will redirect the user after the user signs in to Power BI.
- 3.Azure AD opens the Power BI sign-in page.
- 4.The user signs in using valid Power BI credentials. Note that the actual authentication is completely transparent to the application. The user might sign in using his Power BI credentials or, if the organization policy requires it, the user might use a smart card to authenticate. Azure AD determines the authentication mechanism depending on the AD corporate policy.

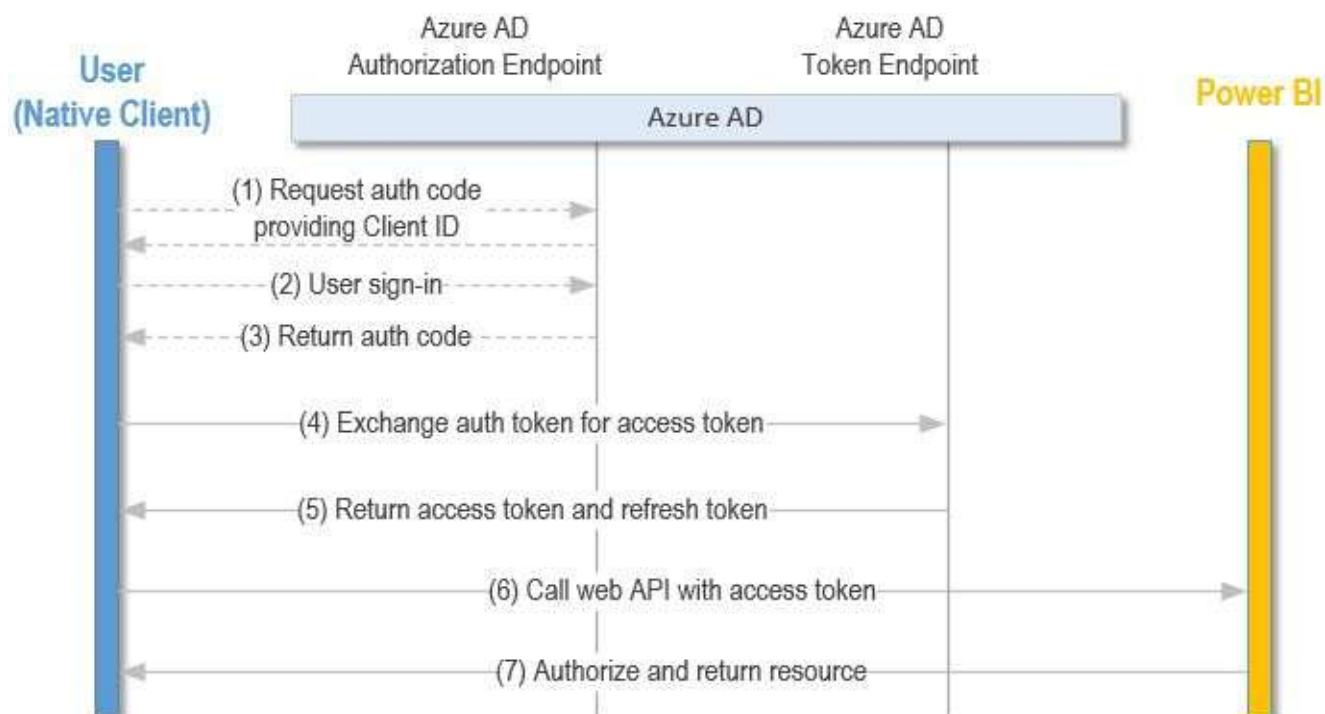
**NOTE** The first time the user signs in with the OAuth flow, the user will be asked to authorize the custom application

for all the permissions you register the app with the Azure AD. To see the authorized custom applications, the user can open the Power BI portal, and then click the Office 365 Application Launcher button (at the top-left corner of the portal) ð “View all my apps”. The user can use this menu to remove authorized custom applications at any point, and then see the granted permissions.

- 5.The Azure AD Authorization Endpoint service redirects the user to the page that's specified by the Reply URL, and it sends an authorization code as a request parameter.
- 6.The web application collects the authorization code from the request.
- 7.The web application calls down to the Azure AD Token Endpoint service to exchange the authorization code with an access token. In doing so, the application presents credentials consisting of the client id and client secret (also called a key). You can specify one or more keys when you register the application with Azure AD. The access token is the holy grail of OAuth because this is what your application needs in order to access the Power BI resources.
- 8.The Azure AD Token Endpoint returns an access token and a refresh token. Because the access token is short-lived, the application can use the refresh token to request additional access tokens instead of going again through the entire authentication flow.
- 9.Once the application has the access token, it can start making API calls on behalf of the user.
- 10.When you register the application, you specify an allowed set of Power BI permissions, such as “View all datasets”. Power BI will evaluate these permissions when authorizing the call. If the user has the required permission, Power BI executes the API and returns the results.

#### ***Understanding the native client flow***

A native client is any installed application that doesn't require the user to use the Web browser. A native client could be a console application, desktop application, or an application installed on a mobile device. **Figure 11.10** shows the OAuth sequence flow for native clients.



**Figure 11.10** This sequence diagram shows the OAuth2 flow for native clients.

As you can see, the authentication flow for native clients is somewhat simpler.

1. Before making a Power BI API call, the native client app calls to the Azure AD Authorization Endpoint and presents the client id. A native client application probably won't have a redirect page. However, because Azure AD requires a Redirect URI, a native client app can use any URI, as long as it matches the one you specified when you register the app with Azure AD. Or, if you're running out of ideas what this artificial URI might be, you can use the Microsoft suggested URI for native clients, which is [https://login.live.com/oauth20\\_desktop.srf](https://login.live.com/oauth20_desktop.srf).
1. As with the web application flow, Azure AD will open the Web browser so that the user can sign in to Power BI. Again, the sign-in experience depends on how the organization is set up with Azure AD but typically the user is asked to enter credentials in the Power BI sign-in page.
2. The Azure AD Authorization Endpoint returns an authorization code. The dotted lines in the **Figure 11.10** represent that this handshake happens within a single call. All a .NET application needs to do is call the *AcquireToken* method of the *AuthenticationContext* class to get the user to sign in, and then acquire the authorization code. In fact, the *AcquireToken* method also performs the next step (step 4) on the same call.
3. A handshake takes place between the native client and the Azure AD Token Endpoint to exchange the authorization code for an access token.
4. The Azure AD Token Endpoint returns an access token and a refresh token.
5. The client calls the Power BI resource passing the access token.
6. Power BI authorizes the user and executes the call if the user has the required permissions.

## 11.2.2 Understanding Application Registration

As a prerequisite of integrating custom applications with Power BI, you must register the app with Azure AD. In the process of registering the application, you specify the OAuth2 details, including the type of the application (web or native client), keys, Redirect URL, and permissions. The easiest way to register a custom application is to use the Power BI registration page. You can also use the Azure Management Portal to register your application but the process is more involved.

Note that although you can use the Power BI registration page for the initial registration process, you still need to use the Azure Management Portal to view the registered applications and to make subsequent changes, such as to change the Redirect URL. Next, I'll walk you through the steps to register web and native client applications using the Power BI registration page and Azure Management Portal.

### *Getting started with registration*

The application registration page helps you create a new application in Azure AD that has all the information required to connect to Power BI. Anyone can register a custom application. You can find the Power BI registration page in the Power BI Developer

Center, as follows:

1. Open your Web browser, navigate to Power BI Service (<http://powerbi.com>), and log in.
2. In the Application Toolbar at the top-right corner of the screen, click the Help & Support menu, and then click “Power BI for developers”.
3. In the Power BI Developer Center, click the “Register your app” link.
4. Or instead of these three steps, you can go directly to the Power BI Registration Page at <http://dev.powerbi.com/apps>.

***Registering your application using the Power BI registration page***

**Figure 11.11** shows the Power BI registration page (to preserve space, steps 3 and 4 are shown on the right). Registering a custom application takes four simple steps:

1. Sign in to Power BI.
1. Specify the application details (see Table 11.8).

**Table 11.8 This table describes the application registration details.**

| Setting       | Explanation                                                                                                                       | Example                                                                                                                                                                                                                                             |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| App Type      | Choose “Server-side Web app” for browser-based web apps, and choose “Native app” for an installed application, such a console app |                                                                                                                                                                                                                                                     |
| Redirect URL  | The web page where Azure AD will redirect the user after the Power BI sign-in completes                                           | <a href="http://localhost:999/powerbiwebclient/redirect">http://localhost:999/powerbiwebclient/redirect</a> (for web apps)<br><a href="https://login.live.com/oauth20_desktop.srf">https://login.live.com/oauth20_desktop.srf</a> (for native apps) |
| Home Page URL | The URL for the home page of your application (used by Azure AD to uniquely identify your application)                            | <a href="http://prologika.com/powerbiwebclient">http://prologika.com/powerbiwebclient</a>                                                                                                                                                           |

# Register an Application for Power BI

Register a new application that can be used to call Power BI APIs

## Step 1 Login to your Power BI account

Welcome, Teo Lachev! (Wrong account? No problem, logout and try again.)

## Step 2 Tell us about your app

Let's start with some basic details.

**App Name:**

PowerBIWebClient

**App Type:**

Specify the type of app. Use 'Server-side Web app' for web apps or Web APIs, or 'Native app' for apps that run on client devices (Android, iOS, Windows, etc.).

Server-side Web app

**Redirect URL:**

A URL within your web application that will be redirected to when user login completes in order for your app to receive an authorization code for that user.

http://www.prologika.com/powerbiwebclient/redirect

**Home Page URL:**

The URL for the home page of your application.

http://www.prologika.com/powerbiwebclient

## Step 3 Choose APIs to access

Select the APIs and the level of access your app needs.

**Dataset APIs**

Read All Datasets

Read and Write All Datasets

**Report and Dashboard APIs**

Read All Dashboards (preview)

Read All Reports (preview)

**Other APIs**

Read All Groups

## Step 4 Register your app

Once you've set everything the way you want it, click the button below and we'll register your app. Your client ID and secret (for web apps only) will appear below. Be sure to copy the values into your app. By clicking the Register App button, you have accepted the terms of use.

**Register App**

**Client ID:**

**Client Secret:**

**Figure 11.11** Register your custom app in four steps, using the Power BI registration page.

- 2.Specify which Power BI REST APIs your application needs permissions to call. For example, if the custom app needs to push data to a dataset table, you'll have to check the "Read and Write All Dataset permissions". As a best practice, grant the app the minimum set of permissions it needs. You can always change this later if you need to.
- 3.Click the Register App button to register your app. If all is well, you'll get back the client ID. For web apps, you'll get also a client secret (also called a key). As I mentioned, your custom web app needs the client secret to exchange the authorization code for an access token.

**TIP** Because Power BI currently doesn't let you update the registration details (you need to use the Azure Management Portal for this), if you develop a web app I recommend you put some thought in the Redirect URL. Chances are that during development, you would use the Visual Studio Development Server or IIS Express. The

Redirect URL needs to match your development setup, such as to include localhost. Once the application is tested, your Azure administrator can use the Azure Management Portal to change your web app and to add a production Redirect URL, such as <http://www.prologika.com/powerbiwebclient/redirect>.

Because managing registered applications can't be done in Power BI, next I'll show you how you can use the Azure Management Portal to view, register, and manage custom apps. You must have Azure AD admin rights to your organization's Active Directory in order to register an application. In addition, although the basic features of Azure Active Directory are free, you need an Azure subscription to use the portal and this subscription must be associated with your organizational account.

#### **Registering web applications using Azure Management Portal**

Follow these steps to register a custom web application in the Azure Management Portal:

1. Navigate to the Azure Management Portal (at <https://manage.windowsazure.com>) and sign in with your organizational account. Again, the account you use must be associated with an Azure subscription.
2. In the navigation bar on the left, select Active Directory. In the right pane, click your organization's name.
3. In your organization's page, select the Applications tab to view the registered applications (**Figure 11.12**).

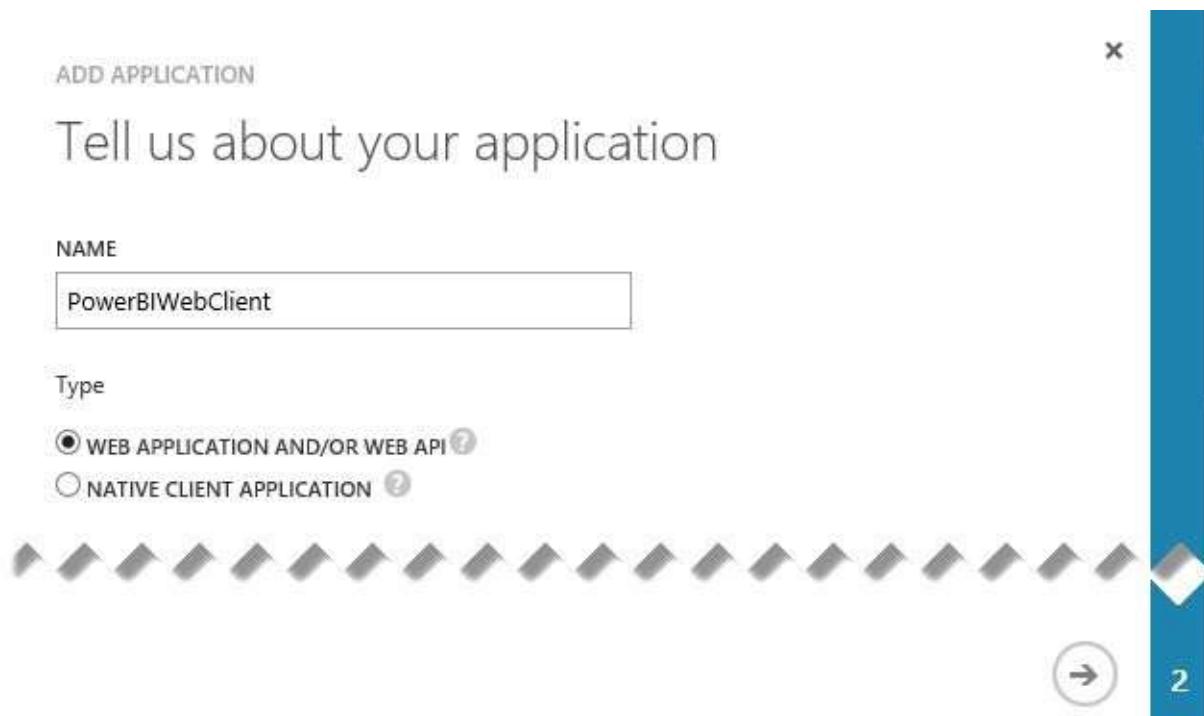
The screenshot shows the Microsoft Azure Management Portal interface. On the left, there is a vertical navigation bar with icons for Cloud, Groups, Applications, Domains, and Reports. The 'Applications' icon is highlighted. The main content area has a header 'prologika' with tabs for USERS, GROUPS, APPLICATIONS, DOMAINS, DIRECTORY INTEGRATION, CONFIGURE, and REPORTS. The 'APPLICATIONS' tab is selected. A search bar at the top says 'Show Applications my company owns'. Below the search bar is a table with columns: NAME, PUBLISHER, and TYPE. The table lists several applications:

| NAME                                    | PUBLISHER | TYPE                      |
|-----------------------------------------|-----------|---------------------------|
| PBIWebApp                               | Prologika | Web application           |
| PowerBICromeNative                      | Prologika | Native client application |
| PowerBICromeSample                      | Prologika | Web application           |
| PowerBIClient                           | Prologika | Native client application |
| ssim-agented717983-58a6-48ef-bb80-bb... | Prologika | Web application           |

**Figure 11.12** Select the Applications tab to view and register custom applications.

3. On the bottom of the page, click Add to register a new app. In the "What do you want to do?" window, and then click the default option of "Add an application my organization is developing".

4.In the “Tell us about your application page”, enter the name of your application, such as PowerBIWebClient (see **Figure 11.13**). The name must be unique across the registered application within your organization. Because this is a web application, leave the default type of “Web Application and/or Web API” selected.



**Figure 11.13** Select the Applications tab to view and register custom applications.

5.On the next page, enter the application Sign-on URL and App ID URL (same as “Home Page URL” in the Power BI registration page). Azure AD doesn’t validate these URLs but they’re required. The sign-on URL is the address of a web page where users can sign in to use your app, such as <http://prologika.com/powerbiwebclient/sign>. App ID is a unique URL that Azure AD uses to identify your app. These two properties have nothing to do with OAuth2 and you can modify them later on if needed.

Once the application is created, you can configure the rest of the application registration details using the Configure tab on the application properties page. **Figure 11.14** shows the configuration details that are relevant to OAuth.

6.The tenant scope specifies if users from other domains can access the application. By default, the application isn’t multi-tenant, meaning that only users on your domain can access the app. Leave the default option selected.

7.Use the Keys section to create a key(s) (same as the client secret in the Power BI registration page) that you can use as a client secret when the application exchanges the authorization code for an access token. Currently, the maximum duration for the key validity is two years.

APPLICATION IS MULTI-TENANT  YES  NO

CLIENT ID: 85be2343-05b3-455d-a052-1833730f2cea

keys

|        |            |            |                                                    |
|--------|------------|------------|----------------------------------------------------|
| 1 year | 10/18/2015 | 10/18/2016 | *****                                              |
| 1 year | 11/3/2015  | 11/3/2016  | THE KEY VALUE WILL BE DISPLAYED AFTER YOU SAVE IT. |

REPLY URL: (ENTER A REPLY URL)

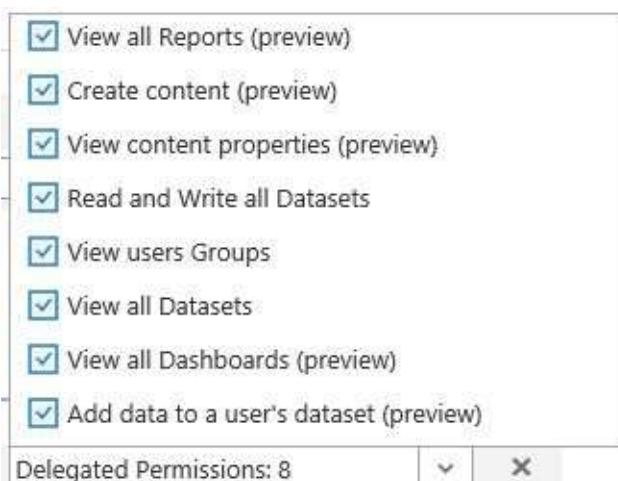
permissions to other applications

|                                |                            |                          |
|--------------------------------|----------------------------|--------------------------|
| Power BI Service               | Application Permissions: 0 | Delegated Permissions: 7 |
| Windows Azure Active Directory | Application Permissions: 0 | Delegated Permissions: 1 |

Add application

**Figure 11.14** In the process of configuring your application, you need to specify keys, Reply URLs, and Power BI permissions.

- 8.The Reply URL (same as Redirect URL in the Power BI registration page) is the page where Azure AD will redirect the user after the user signs in to Power BI. Enter a valid URL, such as <http://www.prologika.com/powerbiwebclient/redirect>.
- 9.So that Power BI can authorize your application, add Power BI Service to the “Permissions to other applications” section by clicking the “Add application” button. If you don’t see “Power BI Service” in the list of applications, make sure that at least one user has signed up for and accessed Power BI. **Figure 11.15** shows you the permissions that Power BI currently supports or that are currently in preview.



## **Figure 11.15** You must grant your application permissions to Power BI .

Some of these permissions correspond to the available REST APIs that you see in the Power BI registration page. For example, if the application wants to get a list of the datasets available for the signed user, it needs to have the “View all Datasets” permission. As a best practice, grant the application the minimum permissions it needs.

### ***Registering native clients***

The native client registration is much simpler. The only required properties are the application name and Redirect URI. As I mentioned, you can use [https://login.live.com/oauth20\\_desktop.srf](https://login.live.com/oauth20_desktop.srf) as a Redirect URI. **Figure 11.16** shows a sample configuration for a native client application. Similar to registering a web app, you need to grant the application permissions to Power BI using the “Permissions to Other Applications” section (not shown in **Figure 11.16**).



**Figure 11.16** Registering a native client application requires a name and Redirect URI.

### ***Viewing and removing registered applications***

The user can view which applications are registered to use Power BI and what permissions they have by following these steps:

1. Log in to Power BI Service (<http://www.powerbi.com>).
1. Click the Office 365 Application Launcher button in the top-left corner.
2. At the bottom of the window that shows the Office 365 applications, click the “View all my apps” link.
3. On the “My apps” screen, find your application, hover on it, and then click the ellipsis

(...) button. A menu will pop up and show you the permissions granted to the application.

Sometimes, the application might stop working. You can use the ellipsis menu to remove the application from the tenant. The next time, you launch the application, it'll ask you to register it with Power BI.

## 11.3 Working with Power BI APIs

Now that you know about the Power BI REST APIs and how to configure custom applications for OAuth, let's practice what's you've learned. Next, I'll walk you through a sample application that will help you get started with Power BI programming. While I was considering writing a sample from scratch, as it turned out Microsoft has already provided a sample app.

Implemented as a .NET console application, the PBIGettingStarted application demonstrates how a native client can authenticate and integrate with Power BI. You can download it from GitHub at <https://github.com/PowerBI/getting-started-for-dotnet/tree/master/PBIGettingStarted>, or by clicking the Sample link on the Power BI Developer Center. For your convenience, I included the version I worked with in the \Source\ch11 folder. To run the sample, you'll need Visual Studio 2010 or a higher version (and valid Power BI credentials).

### 11.3.1 Implementing Authentication

Let's start with understanding how the application uses OAuth2 to authenticate the user before she makes calls to Power BI. As a prerequisite, you'll need to register a native client application in Azure AD.

#### *Configuring the application*

Before running the application, you need to change a few class-level variables to reflect your setup.

1. Open the PBIGettingStarted application in Visual Studio.

2. In Solution Explorer, double click Program.cs to open it.

3. Change the following variables:

```
private static string clientID = "<client_id>";
private static string redirectUri = "https://login.live.com/oauth20_desktop.srf";
private static string datasetName = "SalesMarketing";
private static string groupName = "Finance";
```

Replace the *clientID* variable with the Client ID of your application when you register it with Azure AD (see **Figure 11.14** again). Replace the *redirectUri* variable with the Redirect URI of your application or leave it set to

[https://login.live.com/oauth20\\_desktop.srf](https://login.live.com/oauth20_desktop.srf) if this is the Redirect URI that you register. One of the tasks that the application demonstrates is programmatically creating a new “SalesMarketing” dataset. If the dataset name isn't what you want, change it accordingly. The application demonstrates working with groups. To try this feature, use Power BI Service to create a workspace group, such as “Finance”. Then change the *groupName* variable to the name of the group. If you use Power BI Free, which doesn't support groups, you can just skip this part of the application.

#### *Implementing OAuth*

Every API invocation calls the *AccessToken* method, which performs the actual

authentication. **Figure 11.17** shows the *DatasetRequest* method that is used by all dataset-related examples, such as when the application lists datasets (the *GetDatasets* method). The *DatasetRequest* method creates a web request object as required by the Power BI API specification. It adds an Authorization header with a Bearer property that specifies the access token. In case you’re wondering, the access token is a base64-encoded hash that looks something like this “eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1N...”.

```

543 static string AccessToken()
544 {
545 if (token == String.Empty)
546 {
547 //Get Azure access token
548 // Create an instance of TokenCache to cache the access token
549 TokenCache TC = new TokenCache();
550 // Create an instance of AuthenticationContext to acquire an Azure access token
551 authContext = new AuthenticationContext(authority, TC);
552 // Call AcquireToken to get an Azure token from Azure Active Directory token issuance endpoint
553 AuthenticationResult result = authContext.AcquireToken(resourceUri, clientID, new Uri(redirectUri), PromptBehavior.RefreshSession);
554 token = result.AccessToken;
555 }
556 else
557 {
558 // Get the token in the cache
559 token = authContext.AcquireTokenSilent(resourceUri, clientID).AccessToken;
560 }
561
562 return token;
563 }
593 private static HttpWebRequest DatasetRequest(string datasetsUri, string method, string accessToken)
594 {
595 HttpWebRequest request = System.Net.WebRequest.Create(datasetsUri) as System.Net.HttpWebRequest;
596 request.KeepAlive = true;
597 request.Method = method;
598 request.ContentLength = 0;
599 request.ContentType = "application/json";
600 request.Headers.Add("Authorization", String.Format("Bearer {0}", accessToken));
601
602 return request;
603 }
604 }
605 }
```

**Figure 11.17** The *AccessToken* method performs the OAuth authentication flow.

Remember that you need the access token to authenticate successfully with OAuth. The application obtains the token by calling the *AccessToken* method. Line 551 constructs a .NET *AuthenticationContext* class, passing the URL of the Azure AD Token Endpoint and an instance of a *TokenCache* class. I separated the original code to call *AcquireToken* into two lines, so you can gain a better understanding what happens next. Line 553 calls *AcquireToken*. It passes the URL of the resource that needs to be authorized. In your case, the resource is Power BI and its resource URI is <https://analysis.windows.net/powerbi/api>. The code also passes the Client ID and the redirect URI.

When the application calls *AcquireToken*, you’ll be required to sign in to Power BI. Once you type in your credentials and Power BI authenticates you successfully, you’ll get back an instance of the *AuthenticationResult* class. *AuthenticationResult* encapsulates important details, including the access token, its expiration date, and refresh token. On line 554, the application stores the access token so that it can pass it with subsequent API calls

without going through the authentication flow again. As it stands, the application doesn't use the refresh token flow, but you can enhance it to do so. For example, you can check if the access token is about to expire and then call the *authContext.AcquireAccessTokenByRefreshToken* method.

### 11.3.2 Invoking the Power BI APIs

Once the application authenticates the interactive user, it's ready to call the Power BI APIs. The next sample demonstrates calling various APIs to create a dataset, adding and deleting rows to and from a dataset table, changing the dataset table schema, and getting the groups that the user belongs to. I'll walk you through the PBIGettingStarted code for creating datasets and loading a dataset table with data.

#### *Creating datasets*

A custom application can create a Power BI dataset to store data from scratch. The sample application demonstrates this with the *CreateDataset* method (see **Figure 11.18**). Line 213 constructs the API method signature for creating datasets by using the POST verb. Line 216 calls the *GetDatasets* method (not shown in **Figure 11.18**), which in turns calls the "List all datasets" Power BI API.

```
207 static void CreateDataset()
208 {
209 //In a production application, use more specific exception handling.
210 try
211 {
212 //Create a POST web request to list all datasets
213 HttpWebRequest request = DatasetRequest(String.Format("{0}/datasets", datasetsUri), "POST", AccessToken());
214
215 //Get a list of datasets
216 dataset ds = GetDatasets().value.GetDataset(datasetName);
217
218 if (ds == null)
219 {
220 //POST request using the json schema from Product
221 Console.WriteLine(PostRequest(request, new Product().ToDatasetJson(datasetName)));
222 }
223 else
224 {
225 Console.WriteLine("Dataset exists");
226 }
227 }
228 catch (Exception ex)
229 {
230 Console.WriteLine(ex.Message);
231 }
232 }
```

**Figure 11.18** The *CreateDataset* method demonstrates how to programmatically create a dataset.

Then the code checks if the dataset with the name specified in the *datasetName* variable (the default name is "SalesMarketing") already exists. If it doesn't exist, line 221 calls the *PostRequest* method to create the dataset. The *PostRequest* method is a helper method that wraps the *HttpWebRequest* class. The request must specify the dataset definition in a JSON format, which must include at least one table. The application defines a Product

object whose definition will be used for the dataset table schema.

Line 221 passes an instance of the Product object, and it serializes the object to the JSON format. As a result, the request body contains the JSON representation of a dataset that has a single table called Product with five columns (ProductName, Name, Category, IsComplete, and ManufacturedOn). Once the call completes, you should see the SalesMarketing dataset in the Power BI Service navigation bar.

At this point, the dataset Product table contains no data. However, I recommend you take a moment now to create a table report for it, and then pin a visualization from the report to a dashboard. This will allow you to see in real time the effect of loading the dataset with data. Next, the code loads some data.

#### **Loading data**

The custom application can programmatically push rows to a dataset table. This scenario is very useful because it allows you to implement real-time dashboards, similar to the one we've implemented with Azure Stream Analytics Service in the previous chapter. The difference is that in this case, it's your application that pushes the data and you have full control over the entire process, including how the data is shaped and how often it pushes data to Power BI. The *AddRows* method demonstrates this (see **Figure 11.19**).

```
375 static void AddRows(string datasetId, string tableName)
376 {
377 //In a production application, use more specific exception handling.
378 try
379 {
380 HttpWebRequest request = DatasetRequest(String.Format("{0}/datasets/{1}/tables/{2}/rows", datasetsUri, datasetId, tableName), "POST", AccessToken());
381
382 //Create a list of Product
383 List<Product> products = new List<Product>
384 {
385 new Product{ProductID = 1, Name="Adjustable Race", Category="Components", IsCompete = true, ManufacturedOn = new DateTime(2014, 7, 30)},
386 new Product{ProductID = 2, Name="LL Crankarm", Category="Components", IsCompete = true, ManufacturedOn = new DateTime(2014, 7, 30)},
387 new Product{ProductID = 3, Name="HL Mountain Frame - Silver", Category="Bikes", IsCompete = true, ManufacturedOn = new DateTime(2014, 7, 30)},
388 };
389
390 //POST request using the json from a list of Product
391 //NOTE: Posting rows to a model that is not created through the Power BI API is not currently supported.
392 // Please create a dataset by posting it through the API following the instructions on http://dev.powerbi.com.
393 Console.WriteLine(PostRequest(request, products.ToJson(JavaScriptConverter<Product>.GetSerializer())));
394 }
395 catch (Exception ex)
396 {
397 Console.WriteLine(ex.Message);
398 }
399 }
```

**Figure 11.19** The *AddRows* method demonstrates how your custom app can load a dataset table.

On line 380, the code creates the API method signature for creating rows using the POST verb. Then the code creates a list collection with three products. This collection will be used to add three rows to the dataset. On line 393, the code calls the *PostRequest* method and passes the collection (serialized as JSON). This JSON output will be included in the request body. Once this method completes, three rows will be added to the Product table in the *SalesMarketing* dataset. If you watch the Power BI dashboard, you should see its tile

data updating in real time. Now you have a real-time BI solution!

## 11.4 Summary

The Power BI APIs are based on open industry standards, such as REST, JSON, and OAuth. These APIs allow you to automate content management and data manipulation tasks, including creating and deleting datasets, loading dataset tables with data, changing the dataset schema, and determining the user's group membership. You can use the Power BI Developer Center to learn and try the APIs.

As a trustworthy environment, Power BI must authenticate users before authorizing them to access the content. The cornerstone of the Power BI authentication is the OAuth2 protocol. By default, it uses a three-leg authentication flow that asks the user to sign in to Power BI. As a prerequisite to integrating a custom app with Power BI, you must register the custom app with Azure AD.

The PBIGettingStarted sample app demonstrates how a native client can authenticate and call the Power BI REST APIs. I walked you through the code that creates a new dataset and loads it with data. This allows you to implement real-time dashboards that display data as the data streams from your application.

Another very popular integration scenario that the Power BI APIs enable is embedding reports in custom applications. This is the subject of the next chapter.



## Chapter 12

# Embedding Reports

I mentioned before that because Power BI generates HTML5, users can enjoy insightful reports on any platform and on any device. Wouldn't it be nice to bring this experience to your custom apps? Fortunately, Power BI includes REST APIs that allow you to embed reports and dashboards in any web-enabled application, so that you don't have to navigate your users out of your application and into Power BI Service. Moreover, embedded reports preserve their interactive features and offer the same engaging experience as viewing them in Power BI Service!

As with the previous chapter, this is a code-intensive chapter, so be prepared to wear your developer's hat. I'll start by introducing you to the Power BI dashboard REST APIs that let you embed dashboard tiles, and then I'll walk you through a sample that demonstrates this feature. Next, I'll demonstrate how you can embed interactive reports. You'll also learn how to apply your knowledge about OAuth from the previous chapter to secure content, and you'll learn how to avoid authenticating the user twice. Finally, I'll walk you through sample code that demonstrates these features. You can find the sample ASP.NET application in the \Source\ch12 folder.

## 12.1 Understanding the Embedded Tile API

Embedded reporting is a common requirement for both internal and external (customer-facing) applications. Typically, the application presents a list of reports or dashboards to the user, so the user can pick which one to view. Then the application embeds the content in the presentation layer, so that reports and dashboards appear to be a part of the application itself (as opposed to redirecting the user to Power BI).

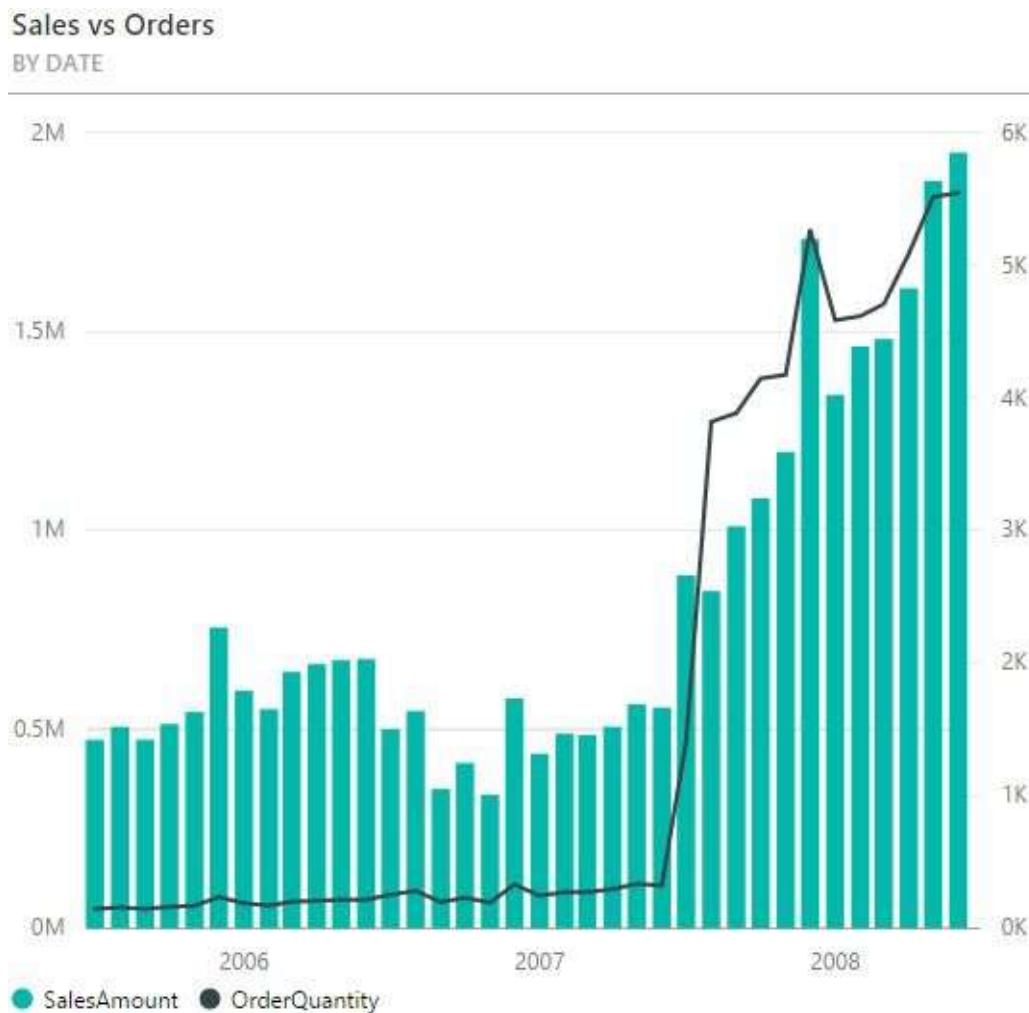
### 12.1.1 Understanding Embedded Tiles

Power BI includes dashboard APIs that allow you to embed specific dashboard tiles in your apps. The benefit of tile embedding is that the end users can decide what tiles they want to see on a consolidated dashboard, without have to use Power BI Service. So your app can mimic the dashboard features of Power BI Service. It can allow the user to compile its own dashboards from Power BI dashboard tiles, and even from tiles that are located in different Power BI dashboards!

#### *Understanding embedded features*

From the end user standpoint, embedded tiles look the same as the ones hosted in Power BI Service.

**Figure 12.1** shows the “Sales vs Orders” tile from the Internet Sales dashboard, which you implemented in Chapter 3.



**Figure 12.1** This tile is embedded in an `iframe` element that's hosted by a web page.

The difference between Power BI Service dashboards and embedding tiles in your app is that, by default, clicking a tile doesn't do anything. You need to write code for something to happen, such as to navigate the user to the underlying report (from which the visualization was pinned). On the upside, your custom code has more control over what happens when the end user clicks a tile.

#### ***Understanding implementation steps***

At a high level, the embedding tile workflow consists of the following steps:

- Obtain the tile identifier – You need to gain access programmatically to a dashboard before you get to a tile. So you'd need to find the dashboards available for the user, and then enumerate the dashboard tiles. Once the user chooses a tile, then you work with the tile identifier.
- Embed a tile – Embed the tile in your application, such by using an iframe.
- (Optional) Handle tile interaction – For example, you might want to open the underlying report when the user clicks a tile.

Of course, security needs to be addressed as well. I'll explain special considerations related to integrating custom application security with Power BI when I walk you through the sample code in section 12.3. Now let's dig deeper into the implementation steps.

### **12.1.2 Enumerating Dashboards and Tiles**

I mentioned in the previous chapter that Power BI includes APIs for enumerating dashboards and tiles. Your application can call these APIs to present the users with a list of tiles that they might want to see in your application.

#### ***Enumerating dashboards***

The “List All Dashboards” method returns the dashboards available in the user’s My Workspace. This method has the following signature:

<https://api.powerbi.com/beta/myorg/dashboards>

Similar to listing datasets, this method supports enumerating dashboards in another workspace by passing the group identifier. For example, if the group identifier of the Finance workspace is e6e4e5ab-2644-4115-96e0-51baa89df249 (remember that you can obtain the groups that the user belongs to by calling the “List All Groups” method), you can get a list of dashboards available in the Finance workspace by using this signature:

<https://api.powerbi.com/beta/myorg/groups/e6e4e5ab-2644-4115-96e0-51baa89df249/dashboards>

#### ***Enumerating tiles***

Once you retrieve the list of dashboards, you can present the user with a list of tiles from a specific dashboard by calling the “List All Tiles” method. This method takes an *id* parameter that corresponds to the dashboard identifier, which you obtain from the “List All Dashboards” method. Here is a sample “List All Tiles” method invocation:

[https://api.powerbi.com/beta/myorg/dashboards/\*id\*/tiles](https://api.powerbi.com/beta/myorg/dashboards/id/tiles)

The result of this method is a JSON collection of all the tiles hosted in that dashboard. The

following snippet shows the definition of the “Daily Sales” tile:

```
{
 "id": "255d89b5-75b4-439d-8776-40f5de0463f0",
 "title": "Daily Sales",
 "subTitle": "By day",
 "embedUrl": "https://app.powerbi.com/embed?dashboardId=d0b9e383-7b84-4bbf-8f55-
9094efdca212&tileId=255d89b5-75b4-439d-8776-40f5de0463f0"
}
```

The *embedUrl* element is what you need to embed the tile in your app.

### 12.1.3 Embedding Tiles

Once you have the embed URL, the next step is to embed the tile on a web page. When sending the tile URL, you’d need to also send the OAuth access token to the Power BI Service.

**NOTE** Currently, the tile embedded API doesn’t support real-time updates when other applications push rows to a dataset. As a workaround, you can use the tile API to refresh the tile data, and then refresh your iframe to load updated data periodically.

#### *Requesting a tile*

In the most common scenario, you’d probably report-enable a web application. You can display the tile in an iframe element. To do so, you must write client-side JavaScript to communicate with the Power BI Service. As a first step, you’d need to reference the iframe element and handle its events:

```
var iframe = document.getElementById("iFrameEmbedTile");
var embedTileUrl = document.getElementById('tb_EMBEDURL').value;
iframe.src = embedTileUrl;
// iframe.src = embedTileUrl + "&width=" + width + "&height=" + height;
iframe.onload = postActionLoadTile;
```

When making this call, you can specify the width and height that you want the tile to have. You must handle the *onload* event in your code by defining an event handler for the *iframe.onload* event. In this case, the *postActionLoadTile* function is the event handler function. If you don’t handle the *iframe.onload* event, the user will see a spinning progress image, and the tile won’t load.

#### *Handling client-side authentication*

Once the iframe triggers the load event, you have to send the access token to Power BI Service. The *postActionLoadTile* function takes care of this:

```
function postActionLoadTile() {
 // get the access token
 accessToken = '<%=Session["authResult"]==null? null:
((Microsoft.IdentityModel.Clients.ActiveDirectory.AuthenticationResult)Session["authResult"]).AccessToken%>';
```

```

if ("") === accessToken) return;

var h = height;
var w = width;

// construct the push message structure
var m = { action: "loadTile", accessToken: accessToken, height: h, width: w };
message = JSON.stringify(m);
// push the message.
iframe = document.getElementById('iFrameEmbedTile');
iframe.contentWindow.postMessage(message, "*");
}

```

First, the code obtains the access token from a session variable that stores the authentication result from the OAuth flow. Normally, the browser would reject cross-domain frame communication. However, the HTML5 specification introduces the `window.postMessage()` method (see <https://developer.mozilla.org/en-US/docs/Web/API/Window/postMessage>) as a secure mechanism to circumvent this restriction. The code creates a JSON push message structure to instruct Power BI Service to load the tile, and then it passes the access token and tile dimensions.

Next, the code posts the message. At this point, the tile should render on the page. Power BI visualizations use HTML5 and JavaScript to render visualizations on the client (read the next chapter for more detailed information about how this works). The tile will automatically scale to fit within the iframe, based on the height and width you provide.

#### ***Filtering tile content***

Dashboard tiles support limited filtering capabilities. This could be useful when you want to further filter the content based on some user-specified filter. You can pass a filter on the embed URL using this syntax:

`https://app.powerbi.com/embed?dashboardId=<dashboard_id>&tileId=<tile_id>&$filter={FieldName} eq '{FieldValue}'`

You can filter on any field in the underlying model, even though the field might not be used in the tile. Fields are contained in tables so the FieldName must be specified as TableName/FieldName. The filter is added as “AND” filter to any existing filters that are already applied to the tile, report or Q&A.

Suppose that the user has indicated that they want to see the sales for Canada only. To meet this requirement, you can filter the SalesTerritoryCountry field in the SalesTerritory table. The embed URL that specifies this filtering condition might look like this:

`embedTileUrl += "&$filter=SalesTerritory/SalesTerritoryCountry eq 'Canada'`

Currently, the filter syntax supports only a single value and you can't specify multiple filtering conditions. When you use a filter, the `navigationUrl` property of the tile will be updated to include the filter. This allows you to pass on the filter to the underlying report if your code opens the report when the user clicks the tile.

#### ***Handling user interaction***

Remember that Power BI Service allows the user to click a tile and opens the underlying report (from which the tile was pinned). You can add a similar feature to your app with the following code:

```
if (window.addEventListener) {
 window.addEventListener("message", receiveMessage, false);
} else {
 window.attachEvent("onmessage", receiveMessage);
}

function receiveMessage(event) {
 if (event.data) {
 messageData = JSON.parse(event.data);
 if (messageData.event === "tileClicked") {
 // code to do something with the report URL
 }
 }
}
```

The code defines an event handler that will listen for messages posted to the iframe content. For example, when the tile is loaded, the message is “tileLoaded”, and when the tile is clicked, the message is “tileClicked”. It’s up to you what you want to do with the *navigationUrl* property, which you obtain from the event parameter. One option is to open a separate window that shows the entire dashboard or the underlying report. Another option (demonstrated by my sample code) is to show the underlying report in the same iframe.

## 12.2 Understanding the Embedded Report API

For years, Microsoft hasn't had a good story about embedded interactive reporting. If developers wanted to distribute interactive reports with their applications, they had to use third-party components. The good news is that Power BI supports this scenario! The bad news is that, at least for now, report embedding is limited to Reading View. This means that users can enjoy report interactive features, such as filtering and highlighting, but they can't change the report layout, such as to add new fields or to change the visualizations. Regardless of this limitation, many scenarios will benefit from embedding Power BI reports.

**NOTE** Supporting Editing View for embedded reports (to allow the user to change the report layout) is high on the Power BI wish list, and Microsoft is working on implementing this feature!

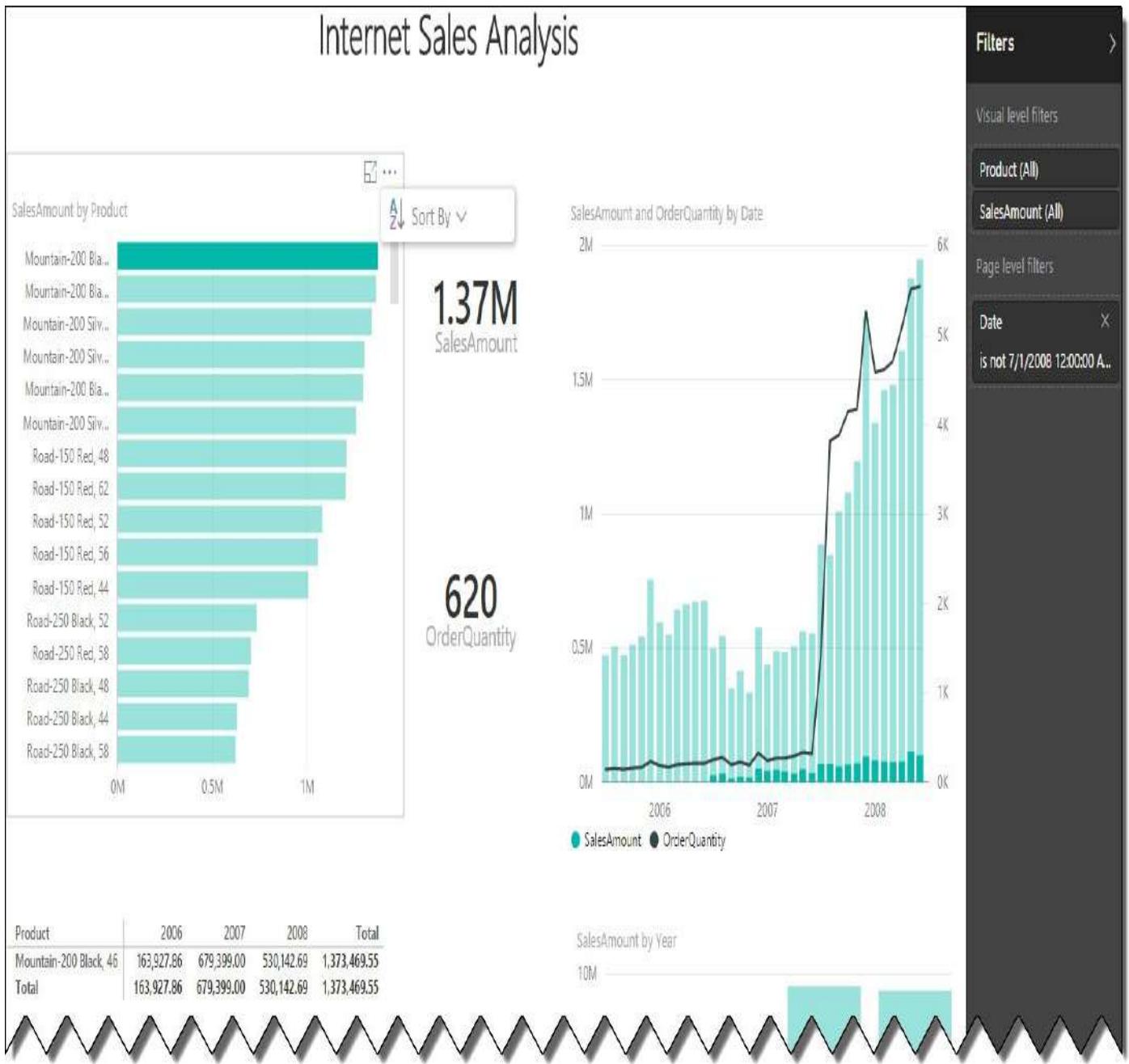
### 12.2.1 Understanding Embedded Reports

Power BI includes report APIs that allows you to embed specific reports in your applications. For example, your application can present the user with a list of reports that he's authorized to see. When the user selects a report, the application can embed the report on a web page.

#### *Understanding embedded features*

**Figure 12.2** shows the Internet Sales Report, from which the “Sales vs Orders” tile was pinned. When the user clicks the “Sales vs Orders” tile, the app embeds the report in the same iframe element. Note that the report supports the same features as when you open the report in Reading View in Power BI Service. For example, you can sort data in visualizations that support sorting.

To demonstrate that interactive highlighting works, I clicked a bar in the “Sales Amount by Product” chart, and this action cross filters the other visualizations. The Filters pane is also available to allow me to apply page-level and visualization-level filters. If the report has multiple pages, I can navigate through the report pages. Users can also pop out visualizations to examine them in more detail. Currently, embedded reports don't support interactive events, such as to do something when the user clicks a visualization on the report.



**Figure 12.2** The Internet Sales Analysis report is embedded on a web page.

#### ***Understanding implementation steps***

Embedding reports involves similar steps as embedding dashboard tiles:

- Obtain the report identifier – Your app can call the “List All Reports” method to present a list of reports to the end user. Once the user selects a report, the app has the report identifier.
- Embed the report – Once you have the report identifier, you can embed the report in your app, such as by showing the report content in an iframe.

Next, I’ll explain the two implementation steps in more details.

### **12.2.2 Enumerating Reports**

Remember from the previous chapter that Power BI includes a REST API method for

enumerating reports. Your application can call this method to show a list of reports that the user can access.

#### ***Enumerating reports in My Workspace***

The “List All Reports” method returns all the reports that are available in the user’s My Workspace. It has the following signature:

<https://api.powerbi.com/beta/myorg/reports>

The resulting JSON response is a collection of report elements. Each report element has *id* (report identifier), *name*, *webUrl*, and *embedUrl* properties. Here’s the definition of the Internet Sales Analysis report:

```
{
 "id": "b605950b-4f18-4eba-9292-82720f215693",
 "name": "Internet Sales Analysis",
 "webUrl": "https://app.powerbi.com/reports/b605950b-4f18-4eba-9292-82720f215693",
 "embedUrl": "https://app.powerbi.com/reportEmbed?reportId=b605950b-4f18-4eba-9292-82720f215693"
}
```

#### ***Enumerating reports in another workspace***

Chances are that your users will share their content. Similar to the “List All Dashboards” method, the “List All Reports” method supports enumerating reports in another workspace by passing the group identifier. For example, if the group identifier of the Finance workspace is e6e4e5ab-2644-4115-96e0-51baa89df249 (remember that you can obtain the group identifier by calling the “List All Groups” method), you can get a list of the reports available in the Finance workspace by using this signature:

<https://api.powerbi.com/beta/myorg/groups/e6e4e5ab-2644-4115-96e0-51baa89df249/reports>

**TIP** To show all the reports that the user can access, you need to enumerate the reports in the user’s My Workspace and the reports from all the groups (workspaces) that the user belongs to.

### **12.2.3 Embedding Reports**

When the user picks a report, your application can use client-side JavaScript to embed the report. This is very similar to embedding dashboard tiles.

#### ***Requesting a report***

If you report-enable a web application, you can display the report in an iframe element. You need to use client-side JavaScript to communicate with Power BI Service and to pass the access token. As a first step, you need to reference the iframe element and handle its events:

```
var iframe = document.getElementById("iFrameEmbedReport");
var embedTileUrl = document.getElementById('tb_EMBEDURL').value;
iframe.src = embedTileUrl;
// iframe.src = embedTileUrl + "&width=" + width + "&height=" + height;
iframe.onload = postActionLoadReport;
```

You must handle the *onload* event in your code by defining an event handler for

*iframe.onload*. In this case, the *postActionLoadReport* function is the event handler.

#### **Handling client-side authentication**

Once the iframe triggers the load event, you need to send the access token to Power BI Service:

```
function postActionLoadReport() {
 // get the access token.
 accessToken = '<%=Session[“authResult”]==null? null:
 ((Microsoft.IdentityModel.Clients.ActiveDirectory.AuthenticationResult)Session[“authResult”]).AccessToken%>';
 // return if no access token
 if ("'" === accessToken) return;
 // construct the push message structure
 var m = { action: “loadReport”, accessToken: accessToken};
 message = JSON.stringify(m);

 // push the message.
 iframe = document.getElementById(‘iFrameEmbedReport’);
 iframe.contentWindow.postMessage(message, “*”);;
}
```

First, the code obtains the access token from a session variable that stores the authentication result from the OAuth flow. The code creates a JSON push message structure to instruct Power BI Service to load the report, and then it passes the access token. Then the code posts the message to Power BI Service. At this point, the report should render on the page.

#### **Filtering the report data**

Similar to tiles, reports support limited filtering capabilities. This could be useful when you want to further filter the report content based on some user-specified filter, after the report filters are applied. You can pass a filter on the embed URL by using this syntax:

[https://app.powerbi.com/reportEmbed?reportId=<report\\_id>&\\$filter={FieldName} eq '{FieldValue}'](https://app.powerbi.com/reportEmbed?reportId=<report_id>&$filter={FieldName} eq '{FieldValue}')

Again, you can filter on any field in the underlying model, even though the field might not be used in the report itself.

## 12.3 Working with Embedded APIs

Now that you've learned how to embed Power BI content, let me walk through sample code to help you report-enable your custom applications. The code demonstrates the following features:

- Authenticating users – The sample code shows both a three-leg authentication flow (a Power BI sign-on page opens) and a two-leg authentication flow (the application authenticates directly with Power BI on behalf of the user).
- Embedding dashboard tiles – The sample code uses the “List All Dashboards” and “List Tiles” methods to present dashboards and tiles to the end user. Then once the user chooses a tile, JavaScript code embeds the tile on a web page.
- Embedding reports – The application shows a list of reports to the end user. Once the user picks a report, the JavaScript code embeds the report on a web page.

### 12.3.1 Understanding the Sample Application

The book source code includes a sample PBIWebApp ASP.NET application in the \Source\ch12\ folder. The code is based on Microsoft's PBIWebApp sample (<https://github.com/Microsoft/PowerBI-CSharp/tree/master/samples/webforms/get-started-web-app-asp.net>), but I've made numerous changes to it, including configuring the application for Forms Authentication (to simulate a more realistic custom app scenario), handling the two-leg OAuth flow, and navigating the user to the underlying report when the user clicks a tile.

#### *Registering the application*

I mentioned in the previous chapter that any custom application that integrates with Power BI must be registered in Azure Active Directory. Follow the steps in the previous chapter to register PBIWebApp using the settings in Table 12.1. For the sake of completeness, the table lists all settings that you need if you register the application using the Power BI registration page or Azure Management Portal (the settings you need for the Power BI registration page are suffixed with an asterisk).

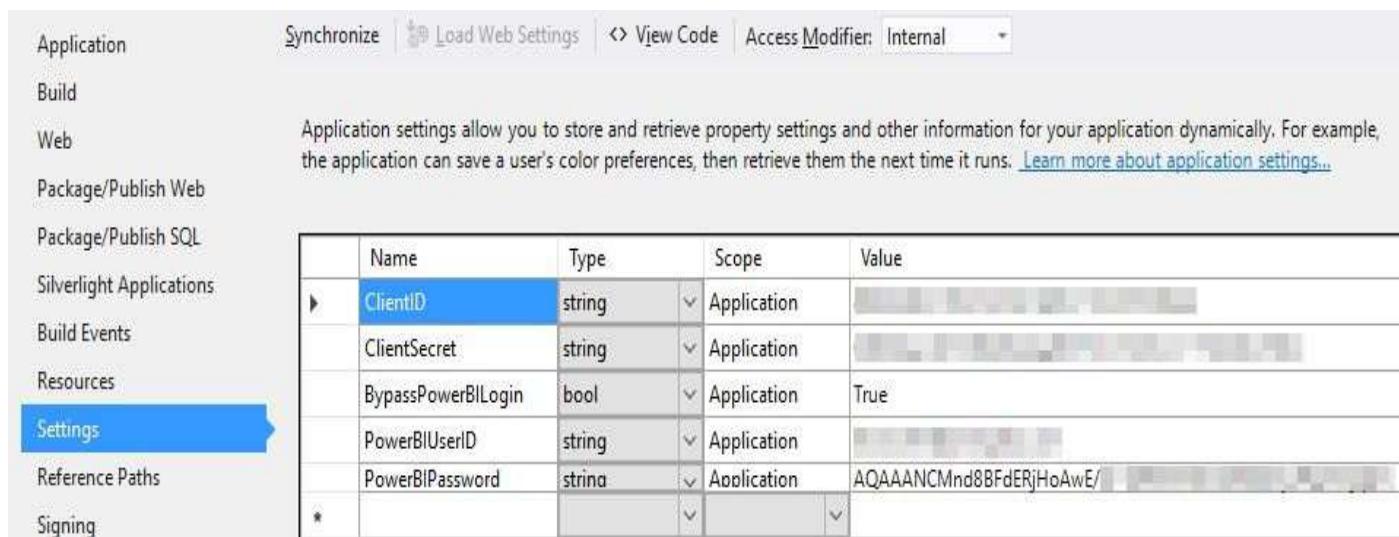
**Table 12.1 The registration settings for the PBIWebApp sample.**

| Setting                     | Value                               | Notes                                                                                        |
|-----------------------------|-------------------------------------|----------------------------------------------------------------------------------------------|
| Name*                       | PBIWebApp                           | Defines the name of the application                                                          |
| Sign-on URL                 | http://prologika.com/pbiwebapp/sign | The URL of the sign-on page (not used by PBIWebApp)                                          |
| Application is multi-tenant | No                                  | The application won't need access to data owned by other organizations                       |
| User assigned required      | No                                  | Any user is allowed to use the application                                                   |
| Keys                        | Auto-generated                      | Create a key that will be used by the two-leg OAuth                                          |
| App ID URL*                 | http://yourorg.com/pbiwebapp        | Defines a logical identifier for your application                                            |
| Reply URL*                  | http://localhost:13528/Redirect     | The redirect page for three-leg authentication (make sure it matches your development setup) |
| Permissions*                | Power BI Service (all permissions)  | Grant all Power BI Service permissions to your application                                   |

#### *Configuring the application*

Before you run the application, you need to change a few configuration settings to match your setup:

1. Open the PBIWebApp project in Visual Studio (version 2012 or higher). If you don't have Visual Studio, you can download and install the free Visual Studio 2016 Community Edition.
2. In Solution Explorer, right-click the project, and then click Properties → Settings (see **Figure 12.3**).



**Figure 12.3** Update the PBIWebApp settings to match your setup before you run the application.

3. Change the *ClientID* setting to match the client ID of your application that you obtain after you register it.
4. Change the *ClientSecret* setting to match the *Client Secret* setting of the registered application.
5. When set to True, *BypassPowerBILogin* forces the application to authenticate the user with the two-leg OAuth flow. When this setting is False, PBIWebApp will open the Power BI sign-on page. Change the setting, depending on which authentication flow you want to test.
6. To test the two-leg authentication flow, change the *PowerBIUserID* setting to the email of your Power BI account (or a test user account that can access the Power BI Service).
7. To test the two-leg authentication flow, change the *PowerBIPassword* setting to your Power BI password (or the password of a test user account). This setting is encrypted using the Windows Data Protection API (DPAPI). For your convenience, I included a helper Util class in the PBIWebApp application, and you can use its *EncryptString* method to encrypt the password.

### 12.3.2 Authenticating Users

One of the Power BI integration challenges that you'll encounter is authenticating the application users with Power BI. A custom app typically authenticates users with Forms Authentication, by showing a login form to let the users enter application credentials. Then the application verifies the credentials against a profile store, such as a table in a

SQL Server database. As it stands, Power BI doesn't support custom authentication to let custom apps inform the Power BI Service that the user is already authenticated.

This limitation presents an integration challenge, because you must register your users twice: with your application and with Power BI (Azure Active Directory). That's fine if you have a small subset of users (or customers), but both maintenance and cost perspectives it's impractical with many user accounts from (every user would probably need a Power BI Pro license, such as when reports connect to an on-premises Analysis Services model).

**NOTE** Power BI custom authentication is a frequently requested feature, and it's likely to be implemented in the near future. I hope that it won't require registering the users with Azure AD and that it'll allow the application to obtain a Power BI access token without authenticating end users with Power BI. In addition, I hope that Microsoft will offer a different business model that won't require licensing individual users.

#### ***Implementing Forms Authentication***

To demonstrate a common real-life scenario, I configured PBIWebApp for Forms Authentication. The pages that handle tile and report embedding are in the Reports folder, which is accessible by authenticated users only. When you run the application (Ctrl-F5), you'll be navigated to the Login.aspx page (**Figure 12.4**).

Welcome to Adventure Works where we bring data to life!

|                                      |                                               |
|--------------------------------------|-----------------------------------------------|
| User ID:                             | <input type="text" value="SomeUserID"/>       |
| Password:                            | <input type="password" value="SomePassword"/> |
| <input type="button" value="Logon"/> |                                               |

**Figure 12.4** The Login.aspx page simulates a custom application that uses Forms Authentication.

For the sake of simplicity, the application doesn't verify the user credentials when the user clicks the Logon button. Of course, in real life your application will verify the user name and password.

#### ***Implementing three-leg OAuth flow***

If the *BypassPowerBILogin* setting is False, the app authenticates the user with Power BI with the three-leg OAuth flow, which I explained in detail in the previous chapter. **Figure 12.5** shows the relevant code.

```

115 //perform three-leg OAuth
116 string authorityUri = "https://login.windows.net/common/oauth2/authorize/";
117 //Create a query string
118 //Create a sign-in NameValueCollection for query string
119 var @params = new NameValueCollection
120 {
121 //Azure AD will return an authorization code.
122 //See the Redirect class to see how "code" is used to AcquireTokenByAuthorizationCode
123 {"response_type", "code"},
124
125 //Client ID is used by the application to identify themselves to the users that they are requesting per
126 //You get the client id when you register your Azure app.
127 {"client_id", Properties.Settings.Default.ClientID},
128
129 //Resource uri to the Power BI resource to be authorized
130 {"resource", "https://analysis.windows.net/powerbi/api"},
131
132 //After user authenticates, Azure AD will redirect back to the web app http://localhost:13528/Redirect
133 {"redirect_uri", String.Format("{0}://{1}/Redirect", HttpContext.Current.Request.Url.Scheme,
134 HttpContext.Current.Request.Url.Authority)}
135 };
136 FormsAuthentication.RedirectFromLoginPage(userID, false);
137 //Create sign-in query string
138 var queryString = HttpUtility.ParseQueryString(string.Empty);
139 queryString.Add(@params);
140 Response.Redirect(String.Format("{0}?{1}", authorityUri, queryString));

```

**Figure 12.5** The three-leg OAuth flow redirects the user to the Power BI sign-on page.

The application creates a JSON request that includes the client id, the Power BI API authorization URL, and the Redirect URI that you specified when you registered the application. Line 140 sends the request to the authorization endpoint, which redirects the user to the Power BI sign-on page. Once the user authenticates with Power BI, the user is redirected to the Redirect.aspx page. The Redirect page saves the authentication result (the access token, refresh token, and other details) into a session variable (see line 38 in **Figure 12.6**).

```

23 string authorityUri = "https://login.windows.net/common/oauth2/authorize/";
24
25 // Get the auth code
26 string code = Request.Params.GetValues(0)[0];
27
28 // Get auth token from auth code
29 TokenCache TC = new TokenCache();
30
31 AuthenticationContext AC = new AuthenticationContext(authorityUri, TC);
32
33 ClientCredential cc = new ClientCredential(Properties.Settings.Default.ClientID, Properties.Settings.Default.ClientSecret);
34 AuthenticationResult AR = AC.AcquireTokenByAuthorizationCode(code,
35 new Uri(new Uri(HttpContext.Current.Request.Url.OriginalString).GetLeftPart(UriPartial.Path)), cc);
36
37 //Set Session "authResult" index string to the AuthenticationResult
38 Session["authResult"] = AR;
39 if (Session["user"] != null) Response.Redirect(Session["redirectUrl"].ToString());

```

**Figure 12.6** This code saves the authentication results into an AuthenticationContext object.

#### *Implementing the two-leg OAuth flow*

Your end users probably won't want to have to authenticate twice. If you develop a customer-facing application and your organization provisions users with Power BI, then

you know the Power BI login credentials and you can avoid the Power BI sign-on page. For example, if your company does business with Acme1 and Acme2, you could register acme1@yourorg.com and acme2@yourorg.com with Power BI. When an external user authenticates with your web application, your app can retrieve the user credentials from somewhere and send them to Power BI.

OAuth is a very flexible security mechanism, and it supports different flows via the *grant\_type* parameter. One of the flows that it supports is the “*grant\_type=password*” flow that allows you to avoid the Power BI sign-in step, if you know the user credentials. This is conceptually similar to how Basic Authentication works. The OAuth2 “*grant\_type=password*” scenario is also referred to as two-leg authentication. PBIWebApp demonstrates this flow when you set the *BypassPowerBILogin* setting to True. The relevant code from the Login.aspx page is shown in **Figure 12.7**.

Line 95 instructs OAuth to use the “*grant\_type=password*” flow. Besides the client ID and client secret, the code adds the Power BI user name and password to the request payload. When Power BI authenticates the user, it’ll immediately return the OAuth access token in the JSON response. So that the application can save the results in the *AuthenticationResult* object (as it does with the three-leg scenario), the code replaces some settings to match the serialized *AuthenticationResult* format (lines 84-86). Finally, the code saves the *AuthenticationResult* object in a session variable, as it does in the case of the three-leg flow.

**NOTE** Remember that the access token has a limited lifespan, and your code needs to catch errors that are caused by expired tokens. Although PBIWebApp doesn’t demonstrate this workflow, your code can use the refresh token to extend the user session when the access token expires.

```

79 Session["user"] = userID;
80 Session["redirectUrl"] = FormsAuthentication.GetRedirectUrl(userID, false);
81 bool bypassPowerBILogin = Properties.Settings.Default.BypassPowerBILogin;
82 if (bypassPowerBILogin)
83 {
84 // As it stands, Power BI doesn't support custom security so the user must be registered with Power BI
85 // After you authenticate the user, you need to retrieve the Power BI user credentials
86 string powerBIUserID = Properties.Settings.Default.PowerBIUserID;
87 System.Security.SecureString powerBIPassword = Util.DecryptString(Properties.Settings.Default.PowerBIPassword);
88
89 // perform two-leg OAuth
90 System.Net.WebRequest request = System.Net.WebRequest.Create("https://login.microsoftonline.com/e7b81d0a-a949-4103-83dc-feff");
91 request.ContentType = "application/x-www-form-urlencoded";
92 request.Method = WebRequestMethods.Http.Post;
93 using (StreamWriter streamWriter = new StreamWriter(request.GetRequestStream()))
94 {
95 string payload = String.Format("grant_type=password&client_id={0}&client_secret={1}&resource=https%3a%2f%2fanalysis.windows.net");
96 payload += WebUtility.UrlEncode(Properties.Settings.Default.ClientID), WebUtility.UrlEncode(Properties.Settings.Default.ClientSecret);
97 payload += WebUtility.UrlEncode(powerBIUserID), WebUtility.UrlEncode(Util.ToInsecureString(powerBIPassword));
98 streamWriter.Write(payload);
99 }
00 using (HttpWebResponse response = (HttpWebResponse)request.GetResponse())
01 {
02 using (StreamReader streamReader = new StreamReader(response.GetResponseStream()))
03 {
04 string payload = streamReader.ReadToEnd();
05 payload = payload.Replace("access_token", "AccessToken");
06 payload = payload.Replace("refresh_token", "RefreshToken");
07
08 AuthenticationResult ar = AuthenticationResult.Deserialize(payload);
09 Session["authResult"] = ar;

```

**Figure 12.7** The two-leg OAuth flow avoids the Power BI sign-on page.

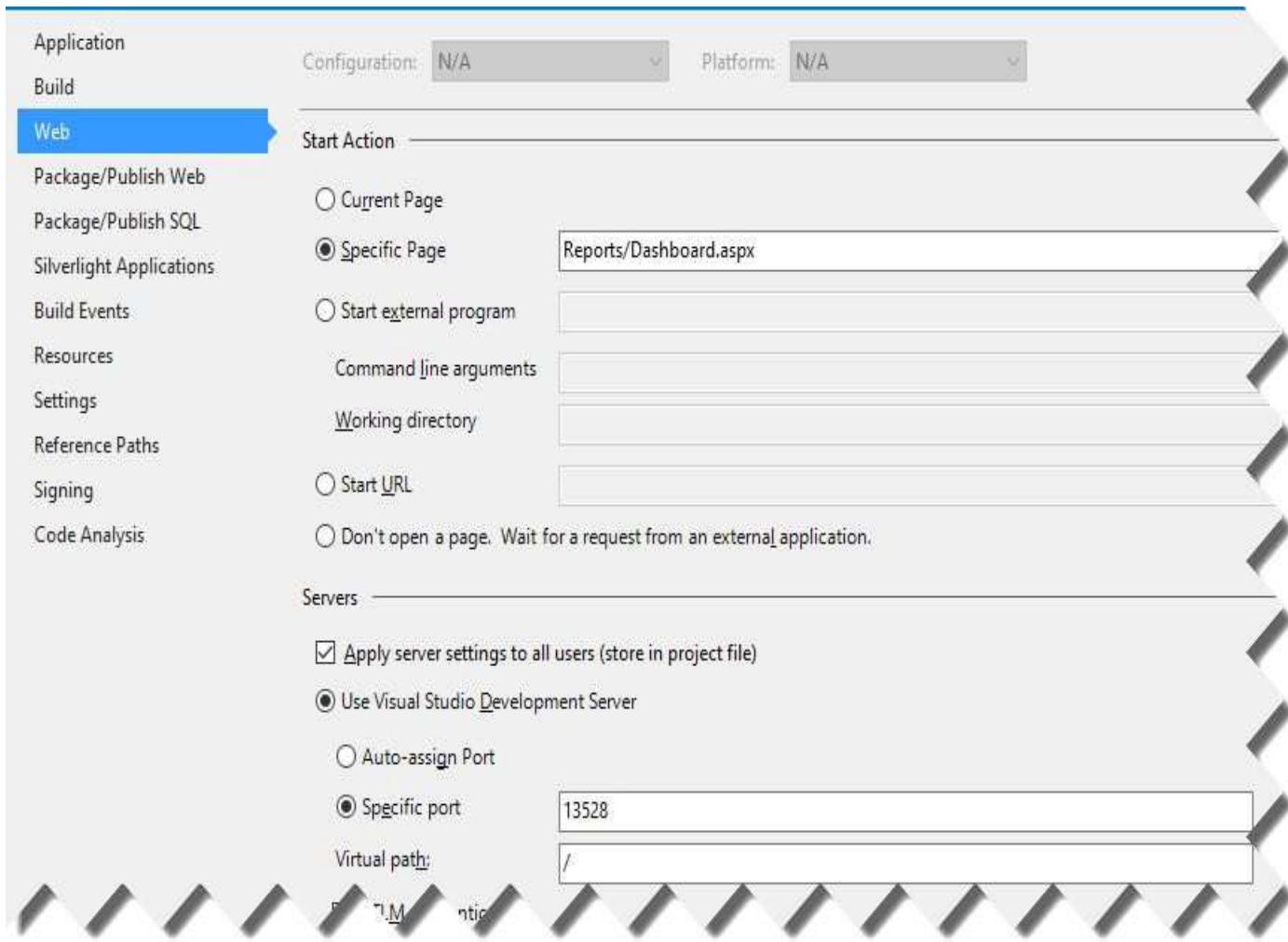
### 12.3.3 Implementing Embedded Tiles

Next, I'll walk you through the sample code for embedding tiles in a custom application. This includes retrieving the list of dashboards available to the authenticated user, getting a list of tiles in the selected dashboard, and embedding the selected tile on a web page.

#### *Configuring the Dashboard web page*

The Dashboard.aspx page in the Reports folder demonstrates how to embed tiles:

- 1.In the project properties, select the Web tab and change the “Specific Page” setting in the Start Action section to Reports/Dashboard.aspx, as shown in **Figure 12.8**.



**Figure 12.8** Configure the PBIWebApp application to start with the Dashboard.aspx page.

- 2.Run the application. You can press Ctrl-F5 to run the application or F5 to start it in Debug mode.
- 3.On the logon step, click the Logon button to authenticate with Power BI. Once the authentication flow completes, PBIWebApp opens the Dashboards.aspx page.

#### **Getting dashboards**

The Dashboard.aspx page demonstrates a typical flow that your application could adopt to embed a tile. As a first step, the application would show the users a list of dashboards that they have access to, as shown in **Figure 12.9**.

## Power BI: Embed a tile

**Signed in as:**

SomeUserID

**Access Token:**

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1Nlslng1dCI6Ik1uQ19WWmNBVGZNNXBPWWIKSE1iYTlnbC
```

**Step 1:** Get dashboards from your account.

| Get Dashboards                       |                               |
|--------------------------------------|-------------------------------|
| 21f6dc52-2199-469a-b5ec-026c06b4a2d1 | Retail Analysis Sample        |
| fbcec898-b7ea-4c13-a431-aaeb5b9724a0 | Cost Tracker.xlsx             |
| a21f77ac-c233-49f6-9d10-10c48b571f34 | Retail Analysis Sample        |
| fe5d0506-c50e-4791-9d5e-0e4f742b442b | Work Order Statistics         |
| 78580167-2d16-4120-afe5-69259f77ea17 | Sales Manager (Dynamics CRM)  |
| ac35443d-0f72-4450-883a-99348933589f | aw                            |
| 21dd560e-46d6-4469-9e88-5de4db7529d0 | Marketing                     |
| 11045dcf-1f21-44ed-b179-3dfbc8d3ceca | Test                          |
| b99c57c9-055f-4a2c-9910-0fecace3b39c | Internet Sales                |
| 97ae8f03-88ae-417f-9258-08bb1b40300e | Google Analytics              |
| 81f5d932-a159-4c94-9eb5-ab3bc1c78894 | Reseller Sales                |
| 0935c9c9-7029-4454-a5e1-7ed2aa0d30ba | Adventure Works Tabular Sales |
| 7a972cd3-1c4e-4686-bfae-32b34d76c46c | Twitter Analysis              |

**Figure 12.9** PBIWebApp shows a list of dashboards available in the user's My Workspace.

Although PBIWebApp only shows the dashboards from the user's My Workspace, you can easily extend it to enumerate all the workspaces that the user belongs to and append the dashboards from these groups. Remember that to do this, you can call the "List All Groups" method. Then you can call the "List All Dashboard" method and specify the group identifier. **Figure 12.10** shows the code that constructs the request and processes the response to show the dashboards in the list.

```

52 protected void getDashboardsButton_Click(object sender, EventArgs e)
53 {
54 string responseContent = string.Empty;
55
56 //Configure datasets request
57 System.Net.WebRequest request = System.Net.WebRequest.Create(String.Format("{0}dashboards", baseUri)) as System.Net.HttpWebRequest;
58 request.Method = "GET";
59 request.ContentLength = 0;
60 request.Headers.Add("Authorization", String.Format("Bearer {0}", authResult.AccessToken));
61
62 //Get datasets response from request.GetResponse()
63 using (var response = request.GetResponse() as System.Net.HttpWebResponse)
64 {
65 //Get reader from response stream
66 using (var reader = new System.IO.StreamReader(response.GetResponseStream()))
67 {
68 responseContent = reader.ReadToEnd();
69
70 //Deserialize JSON string
71 PBIDashboards PBIDashboards = JsonConvert.DeserializeObject<PBIDashboards>(responseContent);
72
73 tb_dashboardsResult.Text = string.Empty;
74 //Get each Dataset from
75 foreach (PBIDashboard db in PBIDashboards.value)
76 {
77 tb_dashboardsResult.Text += String.Format("{0}\t{1}\n", db.id, db.displayName);
78 }
79 }
80 }

```

**Figure 12.10** This code retrieves and shows the list of dashboards.

Lines 57-60 create the web request to invoke the “List All Dashboards” method. Line 60 adds the access token to authorize the user. Line 63 sends the request and obtains the response from the method invocation. Line 68 saves the response, which is described in JSON. Line 71 creates a collection of *PBIDashboard* objects from the response, for easier navigation through the dashboard elements. Lines 75-78 populate the list with the dashboard identifier and dashboard name.

#### Getting tiles

Next, you can specify a dashboard so that you get a list of its tiles:

1. Copy one of the dashboard identifiers, such as the one for the Internet Sales dashboard, and paste it in the text box in the Step 2 section.
1. Click the Get Tiles button to get the tile list (see **Figure 12.12**).

**Step 2:** Get tiles from your dashboards.

Enter a dashboard id from step 1:

**Get Tiles**

|                                       |                 |                                                                                                                                                           |
|---------------------------------------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 91398f7e-7832-4d3f-b45d-30125ef41c08  | Sales           | <a href="https://app.powerbi.com/embed?d=91398f7e-7832-4d3f-b45d-30125ef41c08">https://app.powerbi.com/embed?d=91398f7e-7832-4d3f-b45d-30125ef41c08</a>   |
| 9d9acc46-6694-4605-aad5-9bfef8669f8e  | Orders          | <a href="https://app.powerbi.com/embed?d=9d9acc46-6694-4605-aad5-9bfef8669f8e">https://app.powerbi.com/embed?d=9d9acc46-6694-4605-aad5-9bfef8669f8e</a>   |
| bcd1123b-c620-4511-b61b-37a2a3fe2736  | Sales vs Orders | <a href="https://app.powerbi.com/embed?d=bcd1123b-c620-4511-b61b-37a2a3fe2736">https://app.powerbi.com/embed?d=bcd1123b-c620-4511-b61b-37a2a3fe2736</a>   |
| 90edbdb8b-6da7-464a-8f30-fc88404a7389 | Sales           | <a href="https://app.powerbi.com/embed?d=90edbdb8b-6da7-464a-8f30-fc88404a7389">https://app.powerbi.com/embed?d=90edbdb8b-6da7-464a-8f30-fc88404a7389</a> |

**Figure 12.11** PBIWebApp shows a list of tiles from the selected dashboard.

**Figure 12.12** shows the code that populates the tile list. Lines 96-100 create the request to invoke the “List Tiles” method for the dashboard identifier you specified. Once the method is invoked, PBIWebApp saves the response in a collection of *PBITile* objects. Then the code enumerates through the collection to show the tile identifier, tile name, and the embed URL.

```
84 protected void getTilesButton_Click(object sender, EventArgs e)
85 {
86 string responseContent = string.Empty;
87 string dashboardId = string.Empty;
88
89 if (string.Empty == inDashboardID.Text)
90 {
91 tb_tilesResult.Text = "Please enter a dashboard id above"; return;
92 }
93 dashboardId = inDashboardID.Text;
94
95 //Configure datasets request
96 System.Net.WebRequest request = System.Net.WebRequest.Create(String.Format("{0}Dashboards/{1}/Tiles",
97 baseUri, dashboardId)) as System.Net.HttpWebRequest;
98 request.Method = "GET";
99 request.ContentLength = 0;
100 request.Headers.Add("Authorization", String.Format("Bearer {0}", authResult.AccessToken));
101
102 using (var response = request.GetResponse() as System.Net.HttpWebResponse)
103 {
104 //Get reader from response stream
105 using (var reader = new System.IO.StreamReader(response.GetResponseStream()))
106 {
107 responseContent = reader.ReadToEnd();
108 //Deserialize JSON string
109 PBITiles PBITiles = JsonConvert.DeserializeObject<PBITiles>(responseContent);
110 tb_tilesResult.Text = string.Empty;
111 foreach (PBITile tile in PBITiles.value)
112 {
113 tb_tilesResult.Text += String.Format("{0}\t{1}\t{2}\n", tile.id, tile.title, tile.embedUrl);
114 }
115 }
116 }
117 }
```

**Figure 12.12** This code retrieves the list of tiles that are hosted in the selected dashboard.

#### *Embedding a tile*

Next, you can specify which tile you want to embed on the web page:

1. Copy the embed URL (starts with https) of one of the tiles shown in Step 2. For example, copy the embed URL of the “Sales vs Orders” tile.
- 1.Paste the embed URL in the Step 3 text box, and then click Embed Tile to render the tile on the web page, as shown in **Figure 12.13**.
- 2.(Optional) Click the tile to open the underlying report. Remember that the web page has some JavaScript code that handles tile embedding and report navigation, as I’ve mentioned in section 12.1.

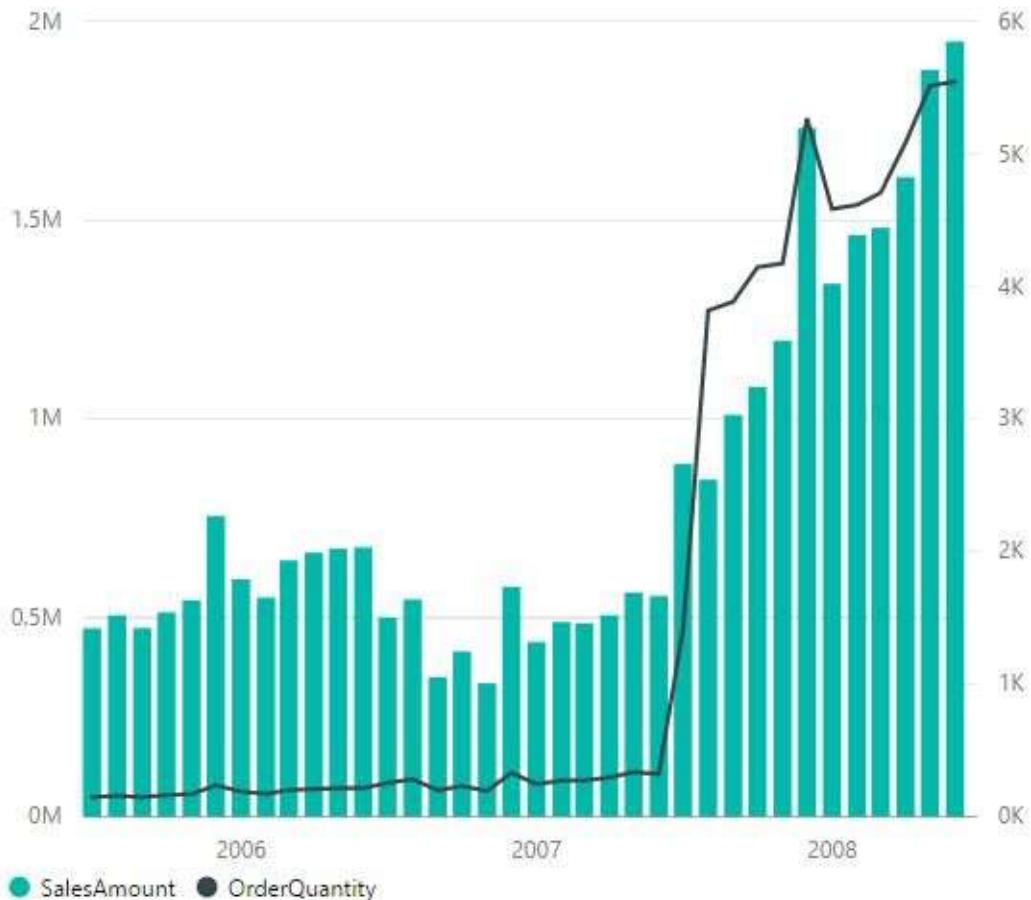
### Step 3: Embed a tile

Enter an embed url for a tile from Step 2 (starts with https://): <https://app.powerbi.com/en>

Embed Tile

Sales vs Orders

BY DATE



**Figure 12.13** Specify the tile embed URL to embed the tile on the web page.

#### 12.3.4 Implementing Embedded Reports

The Report.aspx page demonstrates the embedding report workflow. Make sure to open the project properties and change the Start Action setting to Reports/Report.aspx.

##### *Getting reports*

The Report.aspx page demonstrates how a custom app can embed a report. As a first step, the application would show the user a list of reports the user has access to, as shown in **Figure 12.14**. Although PBIWebApp only shows the reports in the user's workspace, you can easily extend it to enumerate the groups that the user belongs to and to append the reports from these groups. Remember that to do this, you can call the "List All Groups" method to obtain all the groups that the user belongs to. Then you can call the List All Reports method for each group and specify the group identifier in the method signature.

## Power BI: Embed a report

Signed in as:

SomeUserID

Access Token:

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1Nlslng1dCl6lk1uQ19WWmNBVGZNNXBPWWIKSE1iYTlnbC
```

**Step 1:** Get reports from your account.

The screenshot shows a web browser window with a title bar 'Get Reports'. Below the title bar is a scrollable list of report items. Each item has three columns: Report ID, Name, and Embed URL. The list includes: 22454c20-dde4-46bc-9d72-5a6ea969339e (Retail Analysis Sample), ca810691-ac0c-4d47-880e-6ba0e612e76c (Retail Analysis Sample), 7d2fc6b8-1985-4a69-9ad7-4baf753747dc (Work Order Statistics), e9ce4f51-d648-4382-b41e-c8fc544a1c7c (Hello World), 0c8b3e84-13f6-46f6-92ab-1dab083c123e (https://app.powerbi.com/report), ae583010-0f8b-440f-899d-e318253cc30a (Dynamics CRM Sales), 86b8bfba-8078-445b-9297-ac2666ff81ca (Contoso Sales for Power BI Designer), bb09a169-9b78-48d0-ab0b-ffe4ad22ffd1 (Internet Sales), a6f8eb91-099f-4db7-bd0f-7498bda6e264 (Google Analytics), b605950b-4f18-4eba-9292-82720f215693 (Reseller Sales), c301033a-2c55-48a5-ba7d-2846c544f493 (Adventure Works Tabular Sales), a73e9235-727f-4daf-ae75-9199cf44bd64 (Adventure Works).

**Figure 12.14** In Step 1, PBIWebApp shows the reports in the user's workspace.

**Figure 12.15** shows the code that retrieves the list of reports. Lines 49-54 create the request to invoke the “List All Reports” method. Then, the code invokes the method and saves the response as a collection of *PBIReport* objects. Finally, the code enumerates through the collection and shows the report identifier, name, and the embed URL.

```
46 protected void getReportsButton_Click(object sender, EventArgs e)
47 {
48 string responseContent = string.Empty;
49 //Configure datasets request
50 System.Net.WebRequest request = System.Net.WebRequest.Create(String.Format("{0}reports", baseUri))
51 as System.Net.HttpWebRequest;
52 request.Method = "GET";
53 request.ContentLength = 0;
54 request.Headers.Add("Authorization", String.Format("Bearer {0}", authResult.AccessToken));

55 //Get datasets response from request.GetResponse()
56 using (var response = request.GetResponse() as System.Net.HttpWebResponse)
57 {
58 //Get reader from response stream
59 using (var reader = new System.IO.StreamReader(response.GetResponseStream()))
60 {
61 responseContent = reader.ReadToEnd();

62 //Deserialize JSON string
63 PBIReports PBIReports = JsonConvert.DeserializeObject<PBIReports>(responseContent);

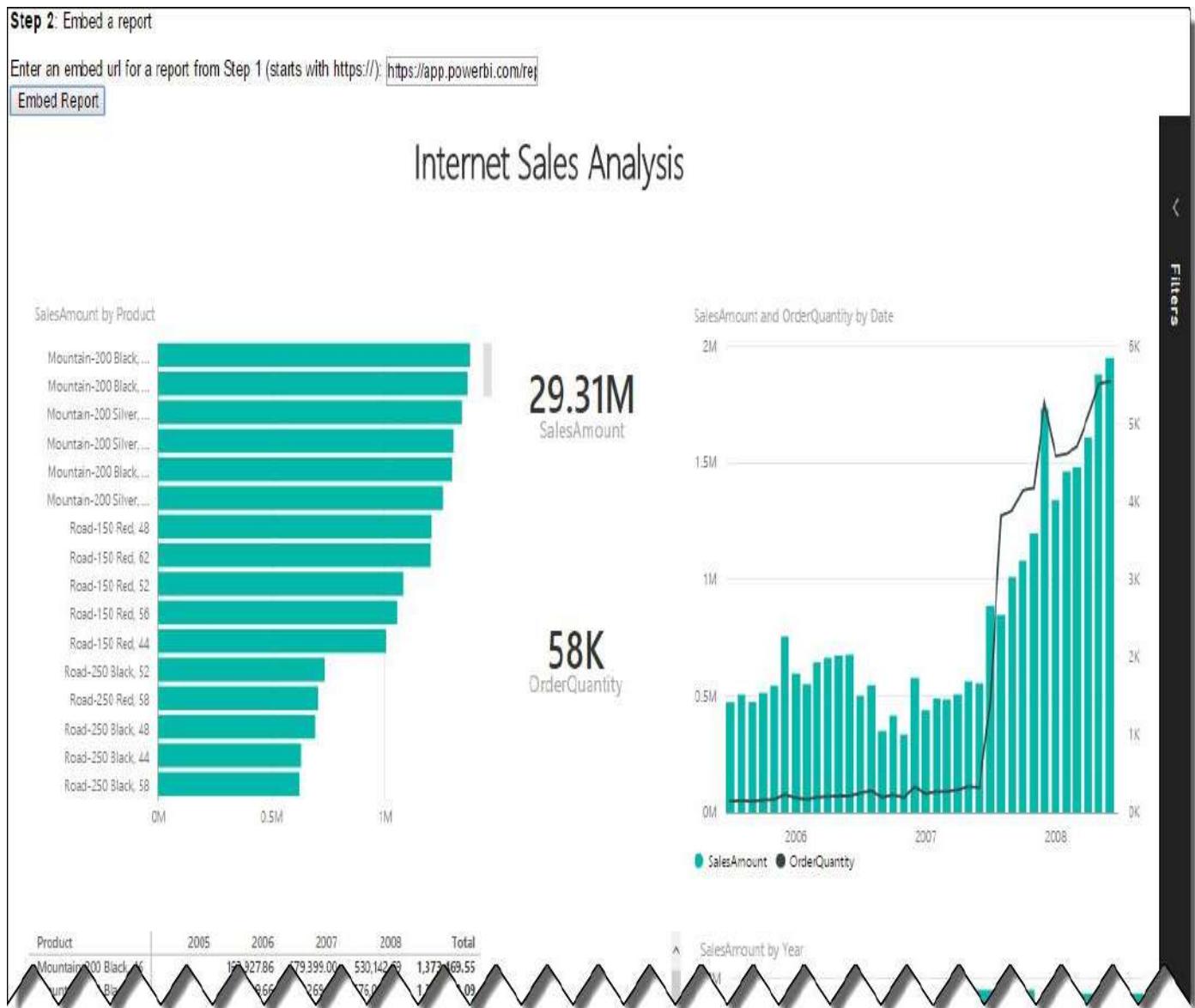
64 tb_reportsResult.Text = string.Empty;
65 //Get each Dataset from
66 foreach (PBIReport rpt in PBIReports.value)
67 {
68 //https://wabi-staging-us-east-redirect.analysis.windows.net
69 tb_reportsResult.Text += String.Format("{0}\t{1}\t{2}\n", rpt.id, rpt.name,
70 "https://app.powerbi.com/reportEmbed?reportId="+rpt.id);
```

**Figure 12.15** This code retrieves the list of reports from the user's workspace.

#### Embedding a report

Next, you can choose a report to embed it on the web page:

- 1.Copy the embed URL (starts with https) of one of the reports shown in Step 2. For example, copy the embed URL for the Reseller Sales report.
- 2.Paste the embed URL in the Step 2 text box and click Embed Report to render the report on the web page, as shown in **Figure 12.16**. Clicking the Embed Report button executes client-side JavaScript code that handles report embedding, as I've discussed in section 12.2.
- 3.(Optional) Test some of the report's interactive features, such as highlighting, sorting, and filtering. All of those features should work because embedded reports preserve their interactive features. Currently, you can't open the embedded report in Editing View to make layout changes to the report. Embedded reports are limited to Reading View.



**Figure 12.16** The Internet Sales Analysis report is embedded on a web page.

## 12.4 Summary

Developers can enrich custom applications with embedded BI content. Thanks to the Power BI open architecture, you can report-enable any web-enabled application on any platform! You can use the Power BI embedded tiles API to assemble dashboards from tiles published to Power BI. And your code can handle tile click events to open the underlying report or another web page.

You can leverage the embedded reports API to embed Power BI reports. Embedded reports preserve their interactive features, such as filtering, highlighting, and sorting. Currently, embedded reports are limited to Reading View.

One of most challenging aspects of report-enabling custom applications is security. Power BI custom security is in the works, and once it's implemented, you won't have to register users twice! As you saw, OAuth is a flexible security framework that supports different authentication flows. The default (three-leg) flow navigates the user to a sign-on page, while the two-leg flow let you authenticate the user directly with Power BI, if you know the user name and password.

Besides report-enabling custom applications, the Power BI APIs allow web developers to extend Power BI's visualization capabilities. You can read about custom visuals in the next chapter!



## Chapter 13

# Creating Custom Visuals

The Power BI visualizations can take you far when it comes to presenting data in a visually compelling and engaging way, but there is still room for the occasional requirement that simply cannot be met with the built-in visuals. For example, suppose you want to convey information graphically using a graph that Power BI does not support? Or, you might need a feature that Microsoft currently doesn't have, such as drilling through a chart data point to another report. Fortunately, web developers can extend the Power BI data visualization capabilities by implementing custom visuals. They can do this with open source JavaScript-based visualization frameworks, such as D3.js, WebGL, Canvas, or SVG.

In this chapter, I'll introduce you to this exciting extensibility area of Power BI. I'll start by explaining what a custom visual is and the developer toolset that Microsoft provides for implementing visuals. Then, I'll walk you through the steps of implementing a sparkline visual (this visual is also published to the Power BI visuals gallery) for showing data trends. Finally, I'll show you how to deploy the custom visual and use it in Power BI.

## 13.1 Understanding Custom Visuals

In Chapter 3, I introduced you to custom visuals from an end user standpoint. You saw that you can click the ellipsis (...) button in the Visualizations pane (both in Power BI Service and Power BI Desktop) and import a custom visual that you've previously downloaded from the Power BI visuals gallery. Now let's dig deeper and understand the anatomy of a custom visual before you learn how to implement your own.

### 13.1.1 What is a Custom Visual?

A custom visual is a JavaScript plug-in that extends the Power BI visualization capabilities. Because the custom visual is dynamically rendered in the Web browser, it's not limited to static content and images. Instead, a custom visual can do anything that client-side JavaScript code and JavaScript-based presentation frameworks can do. As you can imagine, custom visuals open a new world of possibilities for presenting data and new visuals are posted to the Power BI visuals gallery every week!

**NOTE** BI developers might remember that SSRS has been supporting for a while .NET-based custom report items. They might also recall that SSRS custom report items render on the server as static images with limited interactivity. By contrast, Power BI runs the custom visual JavaScript code on the client side. Because of this, custom visuals can be more interactive. To emphasize this, the sparkline visual (whose implantation I discuss in this chapter) demonstrates animated features although this might not be necessarily a good visualization practice.

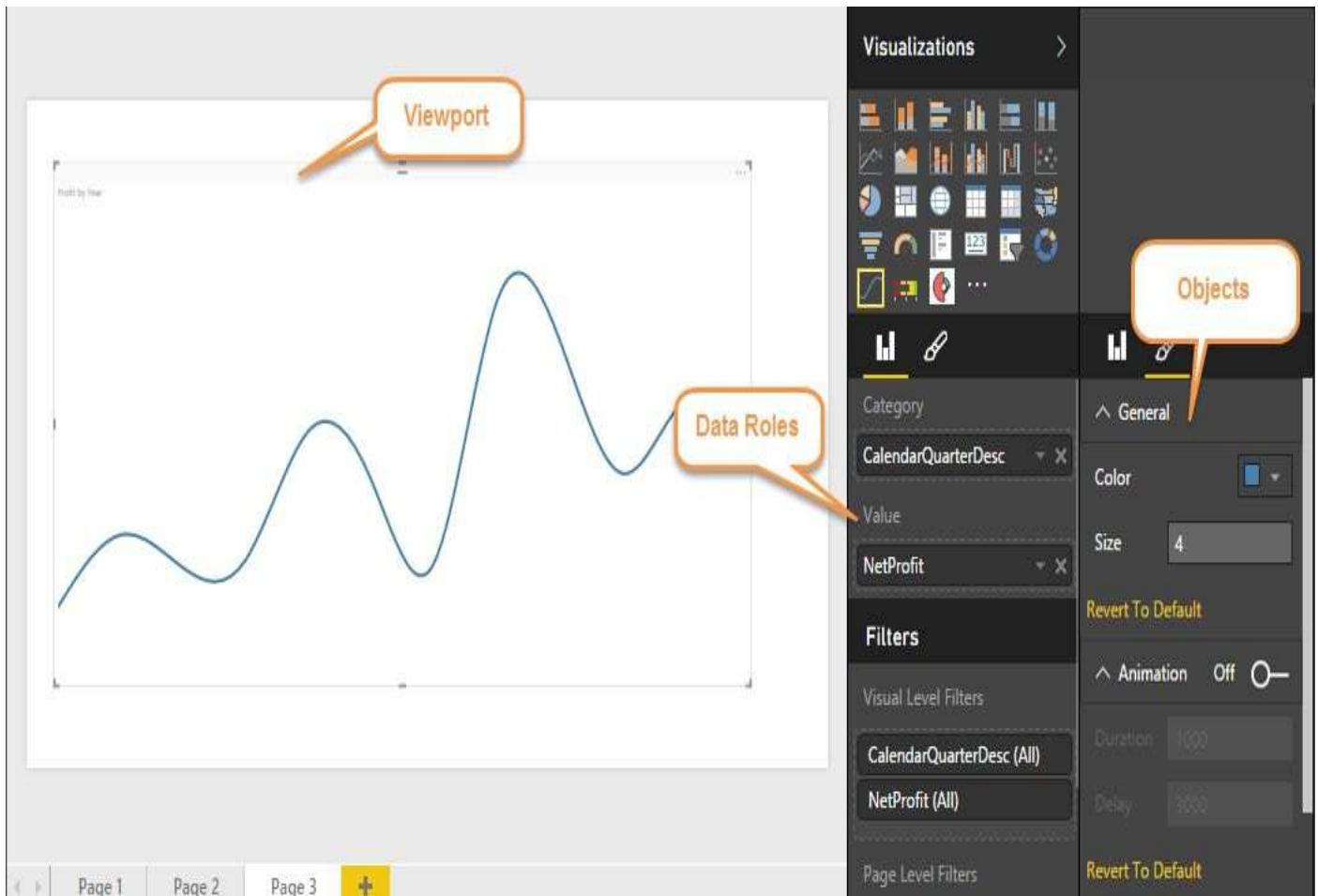
#### *Understanding the custom visual framework*

To allow developers to implement and distribute custom visuals, Microsoft provides the following toolset:

1. Support of custom visuals in Power BI reports – Users can use custom visuals in both Power BI Service and Power BI Desktop.
2. Power BI visuals gallery – A community site (<http://visuals.powerbi.com>) that allows developers to upload new Power BI visuals and users to discover and download these visuals. Custom visuals are provided by both Microsoft and the community.
3. Power BI custom visual developer tools – Developers can use Visual Studio or the Power BI Developer Tools to code and test visuals.

#### *Understanding host integration*

Power BI has different hosting environments where visuals can be used, including dashboards, reports, Q&A, native mobile applications, and Power BI Desktop. From an end user standpoint, once the user imports a custom visual, the user can use it on a report just like the visualizations that ship with Power BI. In **Figure 13.1**, the last row in the Visualizations pane shows that I've imported three custom visuals (Sparkline, Bullet Chart, and Aster Plot) and used the Sparkline custom visual on a report.



**Figure 13.1** The host takes care of the plumbing work required to configure the visual.

When a custom visual is added to a report, the user can specify the size of the visual by dragging its resize handles. The resulting area determines the boundaries of the canvas (also called a viewport) that is available to the visual to draw whatever visualization it creates. When you create a visual, you need to adhere to a specification that determines how the visual interacts with the host environment.

The hosting environment takes care of most of the plumbing work required for configuring the visual. It is the host that takes care of configuring the Fields and Format tabs of the Visualizations pane. The visual simply advertises what capabilities it supports. For example, the Sparkline visual tells the host that it supports one category field and one value field. Once the host discovers this information, it configures the Fields tab of the Visualizations pane accordingly.

The Format tab works in the same way. The visual advertises the formatting options it supports and how they should be presented. However, it is the host that configures the UI (the Format tab). For example, the Sparkline visual tells the host that it supports two properties for formatting the graph: line color and line width. It also supports an optional animation behavior that controls the delay of each redraw and the duration of how fast the graph is drawn. Given this information, the host configures the Format pane accordingly.

The host integration adds a slew of additional features that don't require any coding on your part. The host gets the data based on how the Fields tab is configured, and passes the data to the visual. Interactive highlighting that cross filters the rest of visualizations on the

page (when the user selects an element in one visual), also works without any coding. The host also takes care of report-level, page-level and visual-level filters, and adds Tile and Background settings in the Format pane.

### 13.1.2 Understanding the IVisual Interface

As I noted, a custom visual must adhere to a design specification. This specification defines an IVisual interface, which every custom visual must implement. The specification is documented and available at <https://github.com/Microsoft/PowerBI-visuals/wiki>. The IVisual interface defines three key methods that control the visual lifetime, as follows:

- *init(options: VisualInitOptions): void* – when you place a visual on a report, the host calls the *init()* method to give the visual a chance to perform some initialization tasks. The host passes an options argument, which among other things includes the viewport height and width. Microsoft recommends that you don't draw the visual in the *init()* method.
- *update (options: VisualInitOptions): void* – This is the workhorse of the visual. The *update()* method is responsible for drawing the visual presentation. Every time the host determines that the visual needs to be refreshed, such as a result of configuration changes or resizing, the host will call the *update()* method. Similar to the *init()* method, the host passes an options parameter.
- *destroy(): void* – The host calls this method when the visual is about to be disposed. This typically happens when the visual is removed from the report or the report is closed. The visual can use this method to release any resources that might result in memory leaks, such as unsubscribing event handlers.

The IVisual interface defines additional methods to support the visual configuration but I'll walk you through them when I discuss the Sparkline implementation.

## 13.2 Custom Visual Programming

How do you implement custom visuals and what development tools are available to code and test custom visuals? Microsoft provided a comprehensive toolset to assist web developers implement custom visuals. In addition, Microsoft published as open source the code of all the Power BI visualization and custom visualizations contributed by Microsoft, so there is plenty of reference material.

Creating custom visuals is not that difficult but as with any coding effort, it requires a specific skillset. First, you need to know TypeScript and JavaScript to code custom visuals. You should also have experience in Data-Driven Documents (D3.js) because this is the visualization framework that Microsoft decided to adopt for the Power BI visuals. However, you can also use other JavaScript-based frameworks if you prefer something else than D3.js. Finally, you need to have web developer experience, including experience with HTML, browser Document Object Model (DOM), and Cascading Style Sheets (CSS). If you prefer to use an integrated development environment (IDE) for coding custom visuals, some experience with Visual Studio is desired.

To get you started with custom visual programming, let me introduce you to TypeScript – the programming language for coding custom visuals.

### 13.2.1 Introducing TypeScript

So that they work across platforms and devices, custom visuals are compiled and distributed in JavaScript. But writing and testing lots of code straight in JavaScript is difficult. Instead, for the convenience of the developer, custom visuals are implemented in TypeScript.

#### *What is TypeScript?*

When you implement custom visuals, you use TypeScript to define the visual logic and interaction with the host. TypeScript is a free and open source (<http://www.typescriptlang.org>) programming language developed and maintained by Microsoft for coding client-side and server-side (Node.js) applications. Its specification ([bit.ly/1xH1m5BI](https://bit.ly/1xH1m5BI)) describes TypeScript as “a syntactic sugar for JavaScript”. Because TypeScript is a typed superset of JavaScript, when you compile TypeScript code you get plain JavaScript. So, why not write directly in JavaScript? Here are the most compelling reasons that favor TypeScript:

- Static typing – TypeScript extends tools, such as Visual Studio, to provide a richer environment for helping you code and spotting common errors as you type. For example, when you use Visual Studio and Power BI Developer Tools you get IntelliSense as you type. And when you build the code, you get compile errors if there are any syntax issues.
- Object-oriented – TypeScript is not only data-typed but it's also object-oriented. As such, it supports classes, interfaces, and inheritance.

To learn more about what led to TypeScript and its benefits, watch the video “Introducing TypeScript” by Anders Hejlsberg at <https://channel9.msdn.com/posts/Anders-Hejlsberg->

[Introducing TypeScript](#). Although he shouldn't need an introduction, Anders Hejlsberg is a Microsoft Technical Fellow and the lead architect of C# and creator of Delphi and Turbo Pascal. Anders has worked on the development of TypeScript.

The screenshot shows the TypeScript Playground interface. On the left, there's a large "TypeScript" logo. Below it is a code editor with two tabs: "TypeScript" (selected) and "JavaScript". The TypeScript tab contains the following code:

```
1 class Greeter {
2 greeting: string;
3 constructor(message: string) {
4 this.greeting = message;
5 }
6 greet() {
7 return "Hello, " + this.greeting;
8 }
9 }
10
11 var greeter = new Greeter("world");
12
13 var button = document.createElement('button');
14 button.textContent = "Say Hello";
15 button.onclick = function() {
16 alert(greeter.greet());
17 }
18
19 document.body.appendChild(button);
20
```

The JavaScript tab contains the generated JavaScript code:

```
1 var Greeter = (function () {
2 function Greeter(message) {
3 this.greeting = message;
4 }
5 Greeter.prototype.greet = function () {
6 return "Hello, " + this.greeting;
7 };
8 return Greeter;
9 })();
10 var greeter = new Greeter("world");
11 var button = document.createElement('button');
12 button.textContent = "Say Hello";
13 button.onclick = function () {
14 alert(greeter.greet());
15 };
16 document.body.appendChild(button);
```

A tooltip is visible over the "greet" method in the TypeScript code, showing its type: "greet (method) Greeter.greet(): string".

**Figure 13.2** The TypeScript Playground allows you to compare TypeScript and JavaScript.

#### Comparing TypeScript and JavaScript

To compare TypeScript and JavaScript, here's a short sample from the Typescript Playground site (<http://www.typescriptlang.org/Playground>), which is shown in **Figure 13.2**.

The TypeScript code on the left defines a Greeter class that has a member variable, a constructor and a *greet()* method. Notice that the TypeScript window supports IntelliSense. This is possible because TypeScript defines the type of member variables and method parameters. These types are removed when the code is compiled to JavaScript, but can be used by the IDE and the compiler to spot errors and help you code. TypeScript is also capable of inferring types that aren't explicitly declared. For example, it would determine the *greet()* method returns a string, so you can write code like this:

```
someMethodThatTakesString(greeter.greet());
```

### 13.2.2 Introducing D3.js

Coding the application flow in TypeScript is one thing but visualizing the data is quite

another. Again, using plain JavaScript and CSS to draw graphs would be a daunting experience. Fortunately, there are open-source visualization frameworks that are layered on top of JavaScript. Microsoft decided to adopt the Data-driven Documents (D3.js) framework to implement all the Power BI visualizations but you are not limited to it if you prefer other JavaScript-based visualization frameworks.

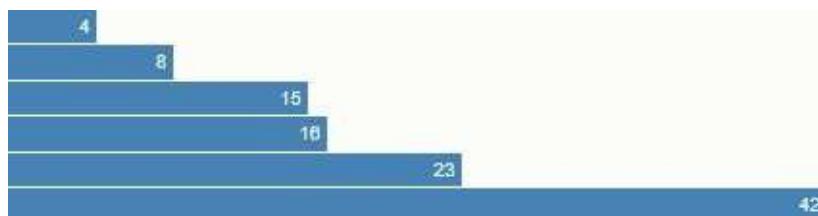
#### **What is D3.js?**

As you know, JavaScript is the de facto standard language as a client-side browser language. But JavaScript was originally designed for limited interactivity, such as clicking a button or handling some input validation. As Internet evolved, developers were looking for tools that would enable them to visually present data within Web pages without requiring reposting the page and generating visuals on the server side. There were multiple projects sharing this goal but the one that gained the most acceptance is D3.js.

According to its site (<http://d3js.org>), “D3.js is a JavaScript library for manipulating documents based on data”. Documents in this context refer to the Document Object Model (DOM) that all Web browsers use to manipulate client-side HTML in an object-oriented way. D3 uses other web standards, such as HTML, CSS, and Scalable Vector Graphics (SVG) to bind data to DOM, and then to apply data-driven transformations to visualize the data. The D3.js source code and a gallery with sample visualizations are available on GitHub (<https://github.com/mbostock/d3>).

#### **Automating visualization tasks with D3.js**

To give you an idea about the value that D3.js brings to client-side visualization, consider the bar chart shown in **Figure 13.3**.



**Figure 13.3** Although simple, this bar chart is not easy to manipulate dynamically in JavaScript and CSS.

The left section in **Figure 13.4** shows how a web developer would implement the same chart using HTML and CSS. The code has one div element for a container, and one child div for each bar. The child div elements have a blue background color and a white foreground color.

So far so good. But what if you want to bind this chart dynamically to data, such as when the report is refreshed or new fields are added? This would require JavaScript code that manipulates DOM in order to put the right values in the right div element. By contrast, the right section shows how you can do this in D3.js. Let’s break it down one line at a time.

First, the code selects the chart element using its class selector (.chart). The second line creates a data join by defining the selection to which you’ll join data. The third line binds the data to the selection. The actual data could be supplied by the application as a

JavaScript array, which may look like this:

```
var data = [4, 8, 15, 16, 23, 42];
```

```
.chart div {
 font: 10px sans-serif;
 background-color: steelblue;
 text-align: right;
 padding: 3px;
 margin: 1px;
 color: white;
}

</style>
<div class="chart">
 <div style="width: 40px;">4</div>
 <div style="width: 80px;">8</div>
 <div style="width: 150px;">15</div>
 <div style="width: 160px;">16</div>
 <div style="width: 230px;">23</div>
 <div style="width: 420px;">42</div>
</div>
```

```
d3.select(".chart")
 .selectAll("div")
 .data(data)
 .enter().append("div")
 .style("width", function(d) { return d * 10 + "px"; })
 .text(function(d) { return d; });
```

**Figure 13.4** The left section shows the chart definition in HTML/CSS while the right section shows the D3 code.

The fourth line outputs a div element for each data point. The fifth line sets the width of each div according to the data point value. The last line uses a function to set the bar label. Note that you'd still need the CSS styles (shown on the left code section) so that the chart has the same appearance. If you have experience with data-driven programming, such as using ADO.NET, you might find that D3.js is conceptually similar, but it binds data to DOM and runs in the Web browser on the client side. It greatly simplifies visualizing client-side data with JavaScript!

### 13.2.3 Understanding the Power BI Visualization Framework

Microsoft contributed the Power BI visualization framework and its complete library of visuals to the open source community. Available on GitHub (<https://github.com/Microsoft/PowerBI-Visuals>), this project has two main goals. First, it teaches developers how to build and test custom visuals. Anyone can clone the GitHub project and explore the code in Visual Studio. You can learn from the existing code how to create your own visuals. You can use a “playground” application to test Power BI and custom visuals. Second, it allows the community to extend the Microsoft-provided visualizations. Microsoft hopes that the community can add useful features to existing visualizations by submitting GitHub pull requests (a pull request is a way of submitting contributions to an open source project).

#### *Getting started with the visualization framework*

The visualization framework includes the following components:

- The source code of all the visuals used in Power BI. Microsoft also published the source code of all custom visuals that Microsoft contributed to the Power BI visuals gallery.
- A playground application to help you test the existing visuals, and experiment with the

ones you have created.

- A VisualTemplate extension for getting started with custom visuals.

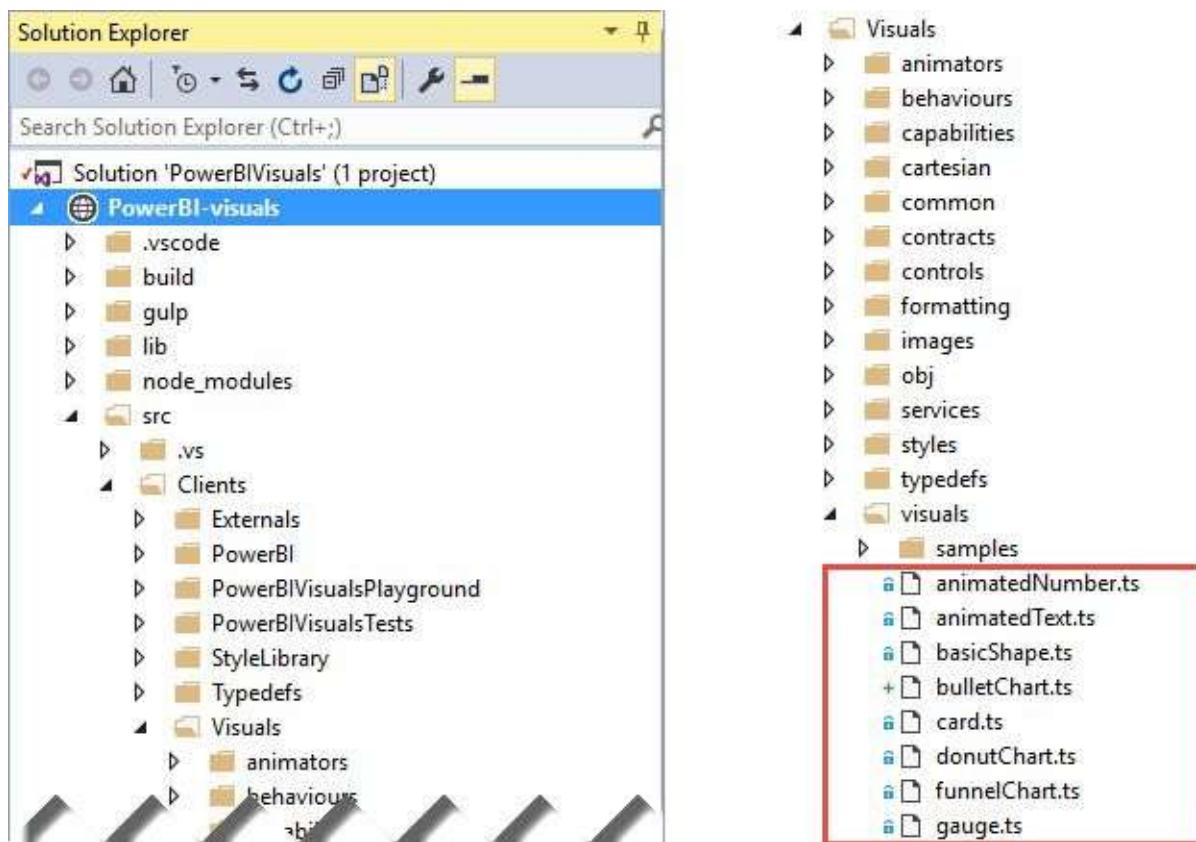
Microsoft provided the steps for cloning and setting up the project on GitHub. I recommend you use Visual Studio 2015 for two reasons. First, the setup steps and walkthroughs on GitHub refer to the Visual Studio 2015 IDE and the VisualTemplate extension is available only with Visual Studio 2015. Second, Visual Studio 2015 integrates with GitHub. If you don't have Visual Studio 2015, you can install the free community edition from <https://www.visualstudio.com/vs-2015-product-editions>.

**TIP** If you plan to contribute to the existing code by submitting GitHub pull requests, I recommend you also install the GitHub Extension for Visual Studio, which is an optional feature in the Visual Studio 2015 setup. This extension lets you clone GitHub repositories, create new repositories, and use GitHub features, such as Pull Requests, Issues, and Reports inside the Visual Studio IDE.

#### *Exploring visual code*

Once you install the project, the first thing you'd probably want to do is browse the code of the Microsoft Power BI visuals:

- 1.In the Visual Studio Solution Explorer, expand the Power BI-visuals project node.
- 2.Expand the src \ Clients \ visuals folder, as shown in **Figure 13.5**.



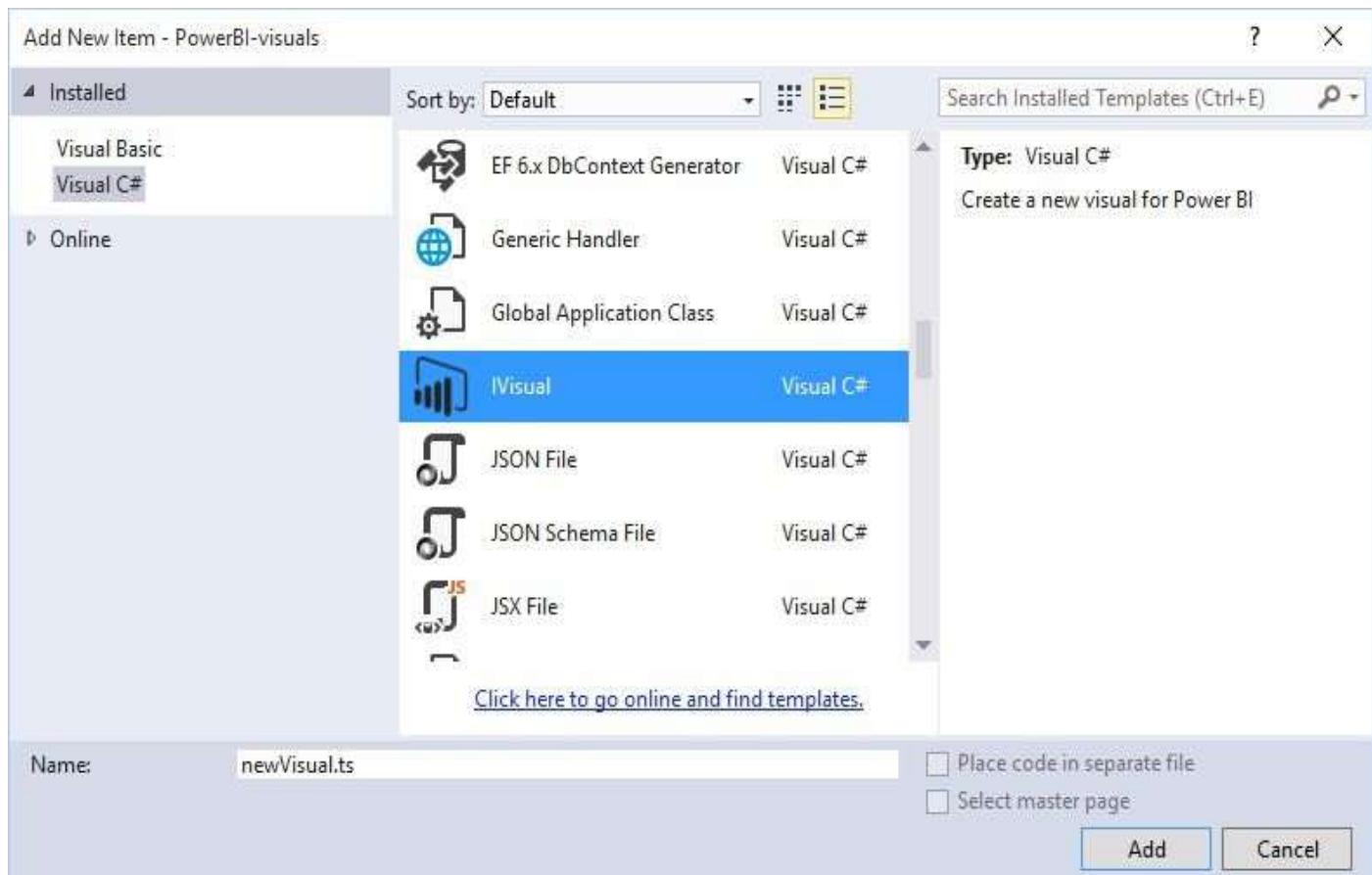
**Figure 13.5** Most of the Power BI visuals are located in the visuals folder.

The samples folder includes most of the Power BI visuals. As you can see, the files have a \*.ts file extension because visuals are written in TypeScript. Most of the Power BI charts are located in the \Visuals\cartesian folder. The \Visuals\capabilities folder includes files that are used by the visuals to advertise their capabilities to the host.

#### *Using the VisualTemplate extension*

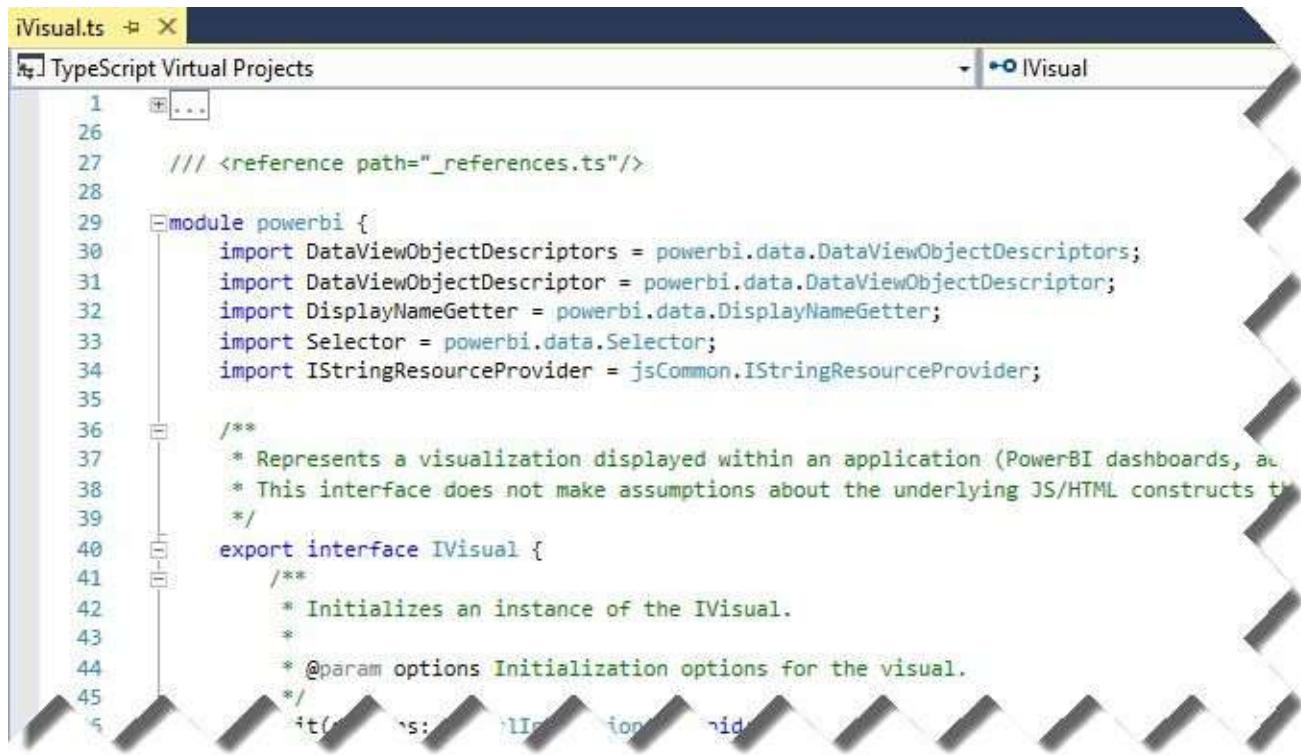
If you decide to code your visual in Visual Studio, you might find the Microsoft-provided VisualTemplate extension useful. It creates a template that you can use to jumpstart the visual implementation. Follow these steps to install the extension:

1. Install Visual Studio 2015. (Remember that the extension targets Visual Studio 2015 only.)
2. On the GitHub Power BI Visuals home page, click the “VSIX Package” link to download and install the extension. To check if the extension is installed, open Visual Studio, go to Tools menu → Extensions, and then check that the VisualTemplate extension is listed.
3. To add a visual template to the project, right-click the folder where you want to add it to, such as the \Visuals\visuals folder, and then click File → “Add New Item”.
4. In the Visual C# tab, find the IVisual template (see **Figure 13.6**), and then click Add. This adds a new IVisual.ts file to your project.



**Figure 13.6** The IVisual template is installed by the VisualTemplate extension.

The code in IVisual.ts file includes a definition of the IVisual interface (see **Figure 13.7**) with comments to help you code your visual. It also includes the definition of the IVisualPlugin interface. Implementing the IVisualPlugin interface allows the playground application to discover the visual, and add the visual to the “Select a visual” drop-down on the home page (index.html), so that you can test the visual in the playground app.



The screenshot shows a Microsoft Visual Studio code editor window. The title bar says "IVisual.ts" and "TypeScript Virtual Projects". The status bar says "IVisual". The code itself is a template for a custom visual. It starts with a reference to "\_references.ts", then imports various Power BI data-related modules. It defines an interface "IVisual" with a constructor that takes an "options" parameter. The "options" parameter is annotated with a comment: "Initialization options for the visual".

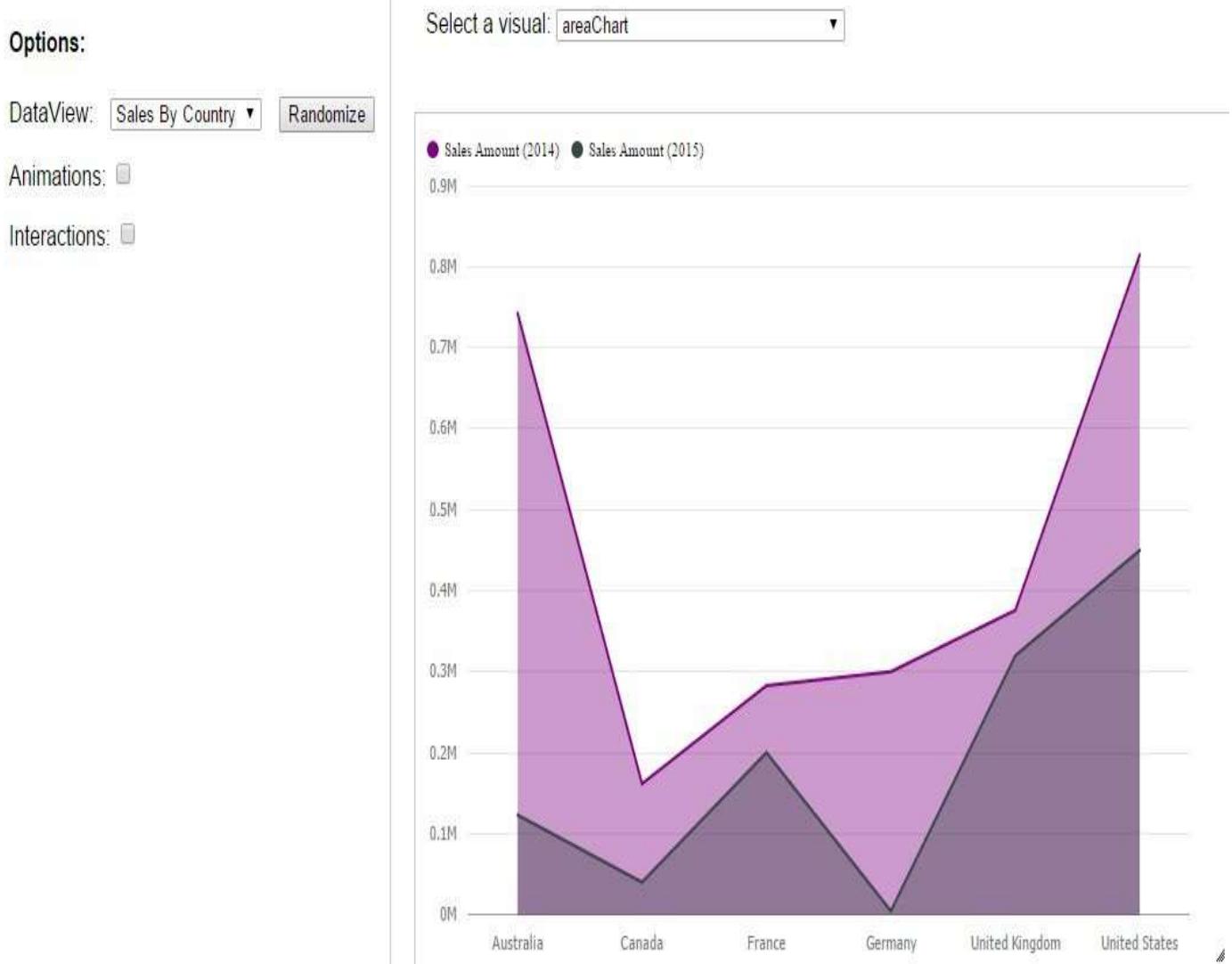
```
1 ...
26
27 /// <reference path="_references.ts"/>
28
29 module powerbi {
30 import DataViewObjectDescriptors = powerbi.data.DataViewObjectDescriptors;
31 import DataViewObjectDescriptor = powerbi.data.DataViewObjectDescriptor;
32 import DisplayNameGetter = powerbi.data.DisplayNameGetter;
33 import Selector = powerbi.data.Selector;
34 import IStringResourceProvider = jsCommon.IStringResourceProvider;
35
36 /**
37 * Represents a visualization displayed within an application (PowerBI dashboards, ac-
38 * This interface does not make assumptions about the underlying JS/HTML constructs th-
39 */
40 export interface IVisual {
41 /**
42 * Initializes an instance of the IVisual.
43 *
44 * @param options Initialization options for the visual.
45 */
46 }
47 }
```

**Figure 13.7** The IVisual template includes code to get you started coding custom visuals.

#### Testing visuals

You can use the playground app to test a visual by following these steps:

- 1.In the Solution Explorer, navigate to src → Clients → PowerBIVisualsPlayground folder.
- 2.Right-click the index.html file, and then click “Set as Start Page”.
- 3.In the project properties (Build tab), set the “Before running startup page” drop-down to “No Build” so that Visual Studio doesn’t compile the project every time you run it (this saves you time).
- 4.Press Ctrl-F5 to run the playground application and open the index.html page (see **Figure 13.8**).



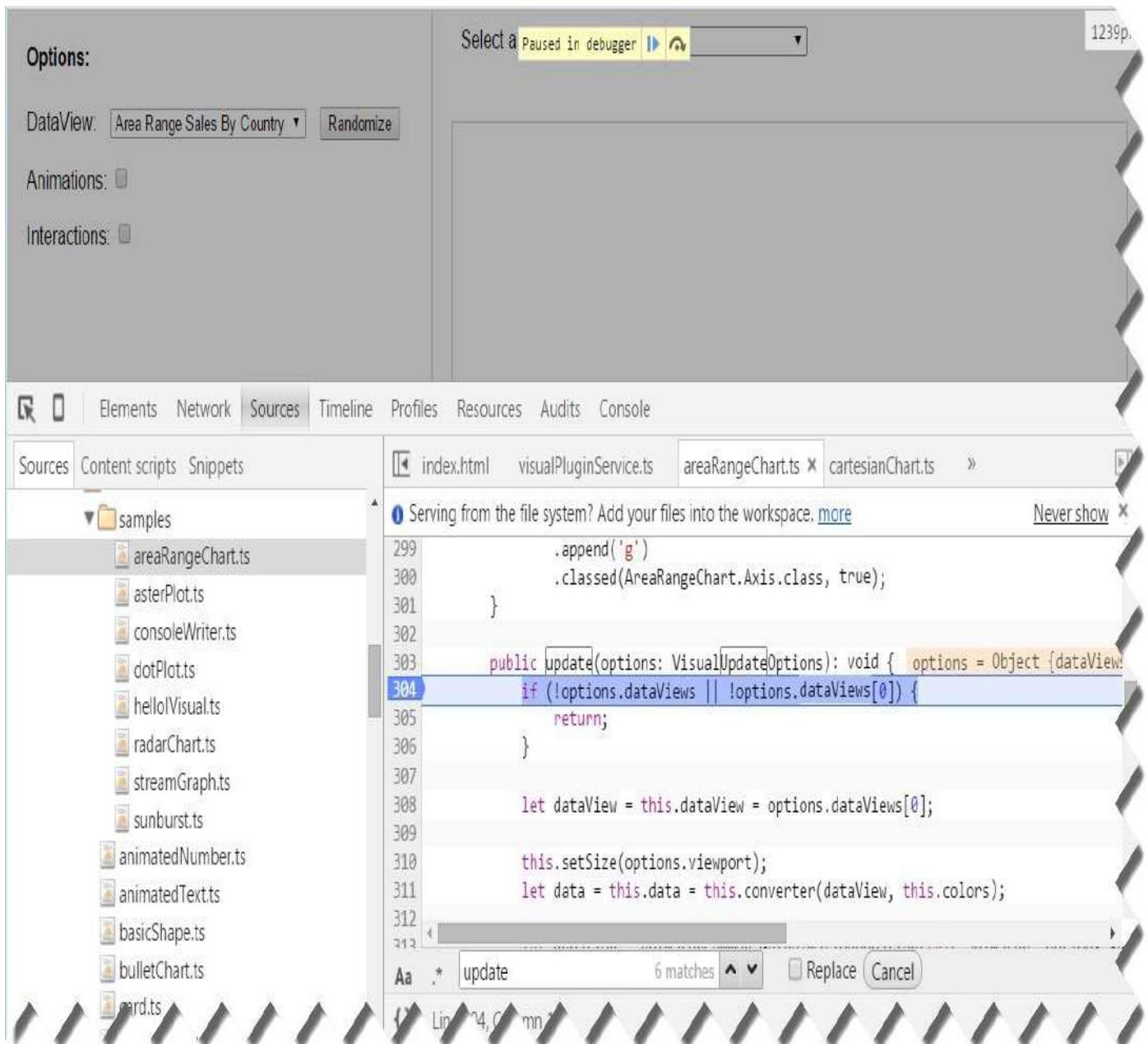
**Figure 13.8** The playground application allows you to test visuals.

5.Expand the “Select a visual” dropdown to find the visual you want to test. The page updates to show you the presentation of the visual. It also gives you options to enable animation and interaction options if the visual supports these features.

#### *Debugging visuals*

You have to use the browser debugging capabilities to debug a visual in the playground application. Here are the steps if you target Google Chrome:

- 1.With the index.html page open, press F12 to open Chrome Developer Tools.
- 2.In the Sources tab, expand the src/Clients folder and locate the TypeScript file of the visualization you want to debug. For example, to debug the areaRangeChart visual, select the areaRangeChart.ts file found in the samples folder (see **Figure 13.9**).



**Figure 13.9** Use the browser debugging capabilities to debug a visual.

- 3.In the source pane, click a line number to set a breakpoint where you want the debugger to stop. For example, to debug the *update* method, put a breakpoint by clicking the line number below the *update()* method.
- 4.Expand the “Select a visual” dropdown and select the visual you want to debug. At this point, the breakpoint should be hit and you should be able to step through the code.

**NOTE** The playground application and Power BI Developer Tools allow you to debug the IVisual interface methods, but because they don’t actually host the visual, they don’t let you step through the configuration code that reads the data roles and formatting properties. If you need to debug this code, consider debugging the visual when it’s added to a Power BI Service report. I’ll show this debugging technique in the next section.

### 13.2.4 Understanding Developer Tools

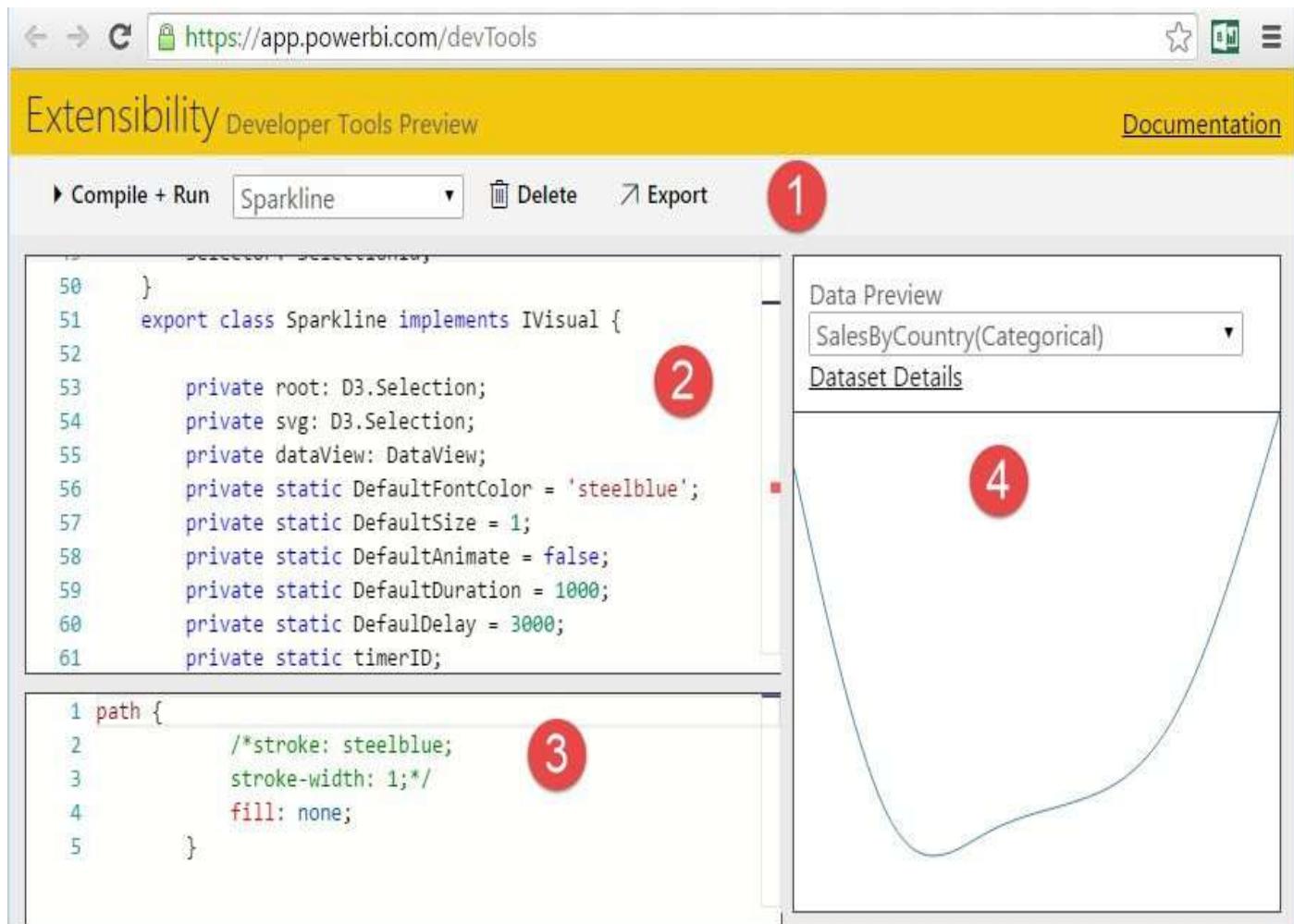
To recap, the Power BI Visuals project allows you to explore the code of the Microsoft-provided visuals, and to create and test your own visuals with Visual Studio. Visual Studio is a great IDE but it could be somewhat overwhelming for the purpose of creating a

custom visual. Fortunately, Microsoft provides another option with the Power BI Developer Tools, which I'll discuss next.

#### Getting started with Developer Tools

Currently in preview, Developer Tools is a web-based TypeScript environment for building and testing custom visuals. The main benefit of Developer Tools is that it integrates with Power BI Service. This allows you to host the visual on a report so that you can test and debug it as you code. Because of its tight integration with the Power BI portal, Developer Tools lets you test the visual and find exactly how it'll work when a Power BI user decides to use it. You can launch the Developer Tools using one of these two options:

1. Once you log in to Power BI Service, expand the Settings menu in the top-right corner and then click Dev Tools.
2. Open your browser and enter <https://app.powerbi.com/devTools>. **Figure 13.10** shows the main elements of the Developer Tools environment.



**Figure 13.10** Use the Developer Tools to code and test your custom visual.

To create a new visual, simply start writing its TypeScript code in the TypeScript Code Pane (item 2 in the diagram). When Developer Tools detects a new class that implements IVisual, it adds the class name to the dropdown in the menu bar (shown as item 1 in the **Figure 13.10**) so that you can switch between visuals. The TypeScript Code Pane is more than just a text pane! Similar to Visual Studio, it hosts the TypeScript compiler and

supports syntax checking, color coding, and IntelliSense. The “Compile + Run” button in the menu bar builds the TypeScript code and shows any errors in the Preview Pane (item 4). If all is well, this button executes the *IVisual.update()* method to render the visual in the Preview Pane.

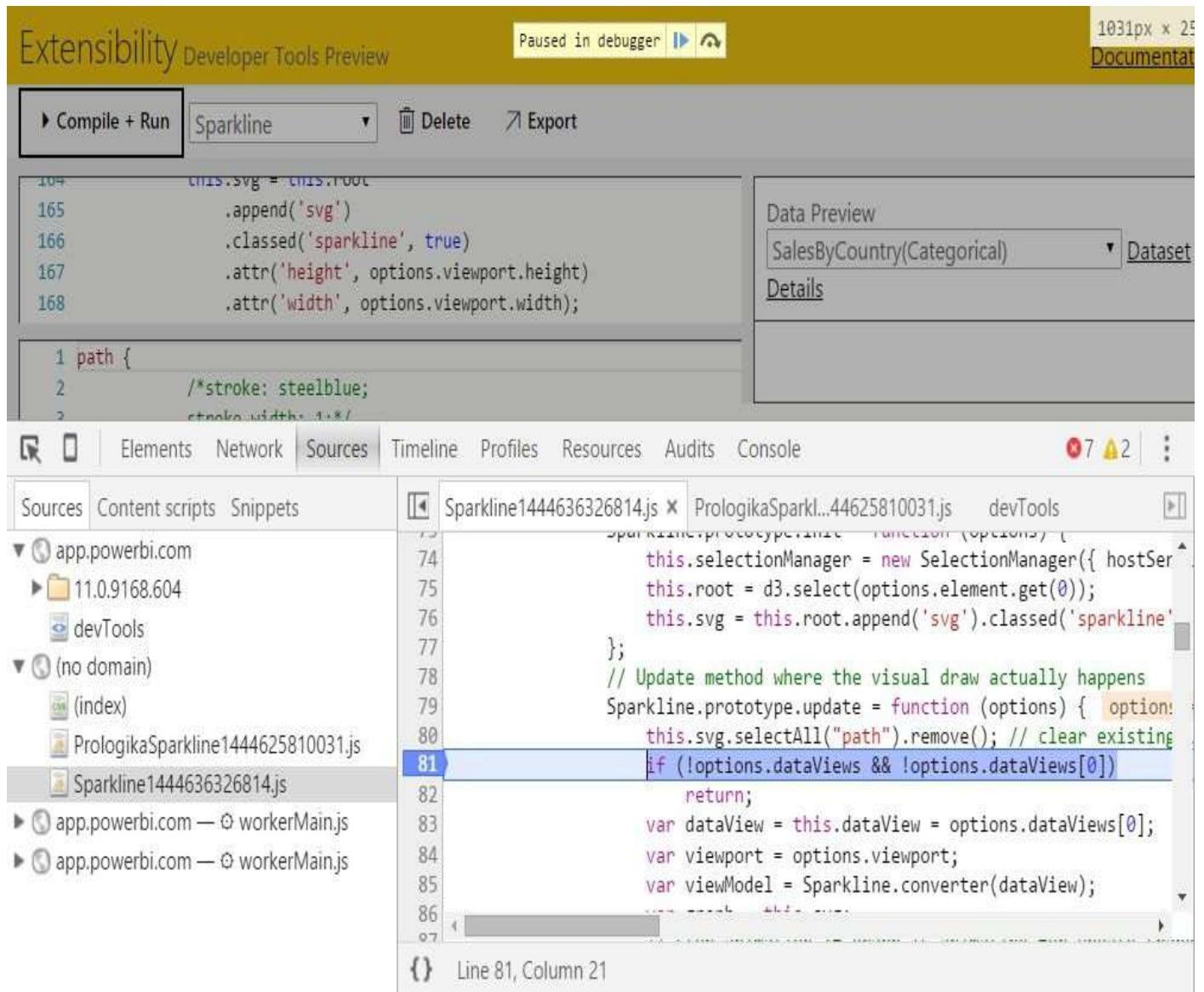
If you have the Power BI Service and Developer Tools open side by side in two tabs in the same browser session, every new visual you add to the Developer Tools will appear in the Visualizations pane on a Power BI report. This allows you to add the visual to a Power BI report and test it. To avoid cluttering the Visualizations pane with all the visuals you work with in Developer Tools, you can delete the visuals you don’t need by pressing the Delete button (see **Figure 13.10** again). The Export link allows you to package the visual code in a \*.pbviz package, so that you can import the visual in Power BI Service or Power BI Desktop, or publish it to Power BI visuals gallery.

Similar to the playground app, the Preview Pane (item 4 on **Figure 13.10**) allows you to select one of the datasets that Microsoft has provided for testing visuals. For example, the SalesByCountry(Categorical) dataset returns a table with some dates on columns and two measures on rows. You can read more about the available datasets and see their data by clicking the Dataset Details link. Finally, the CSS pane (item 3) is for writing CSS rules that control the appearance of the custom visual. The Sparkline visual uses the *fill:none* CSS style so that the drawn shape is not filled.

#### *Debugging visuals*

Similar to Visual Studio, you can use the browser debugging capabilities to step through the visual code in the Dev Tools and in the Power BI portal.

1. Have your visual loaded and compiled in the Developer Tools.
1. Assuming you use Chrome, press F12 to open its developer environment.
2. In the Chrome Sources tab, expand the “no domain” node (see **Figure 13.11**) and select your visual JavaScript file with (\*.js) extension.

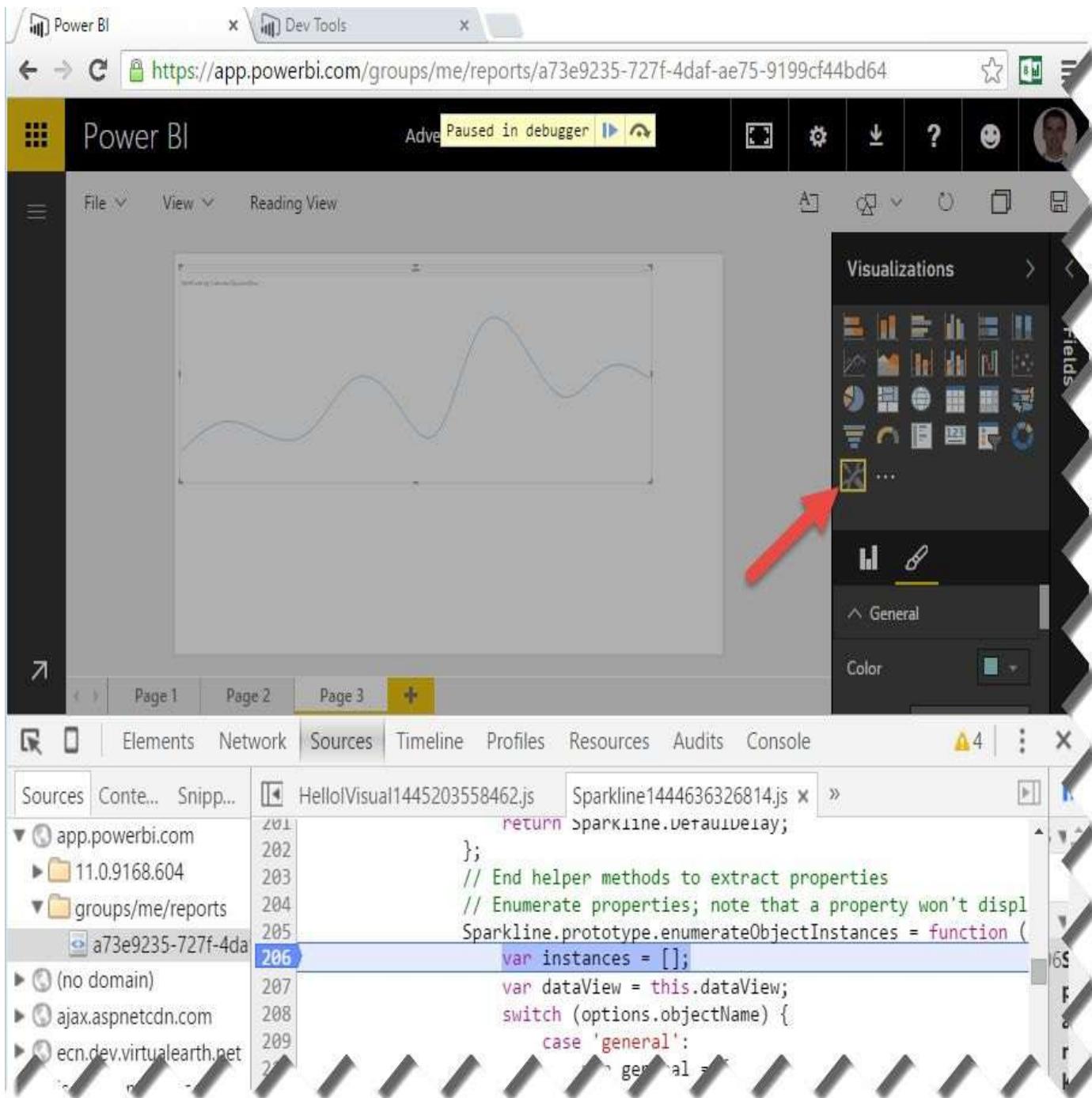


**Figure 13.11** Use the browser debugging capabilities to step through your visual code in Developer Tools.

3.Put a breakpoint somewhere in your code, and then click the “Compile + Run” button again. If your code is reachable, code execution should stop at the breakpoint.

#### *Understanding integration with Power BI Service*

As I mentioned, if you have Power BI and Developer Tools open side by side in two tabs in the same browser instance, Power BI will automatically add the custom visuals you have in Dev Tools to the Visualizations pane (see **Figure 13.12**).



**Figure 13.12** Custom visuals from the Developers Tools are added to the Visualizations pane.

This is useful for two main reasons:

- Testing with Power BI – You can add the visual to a report to do final testing.
- Debug all the code – You can use the browser debugging capabilities to debug the visual code. Unlike debugging the visual code in Visual Studio and Developer tools, however, now you can test *all* the custom visual code, including the code that enumerates the host input, such as when you make changes to the Data and Format tabs in the Visualizations pane.

**NOTE** As of the time of writing this, making changes in Developer Tools requires refreshing (F5) the Power BI page for the changes to apply to the visual you are testing. You have to remove and re-add the custom visual to the report so that the new changes apply. The visuals loaded through Dev Tools are stored in the HTML5 local storage. To clear the visual icons, start debugging in the browser, and then call `localStorage.clear()` in the browser's debug console.

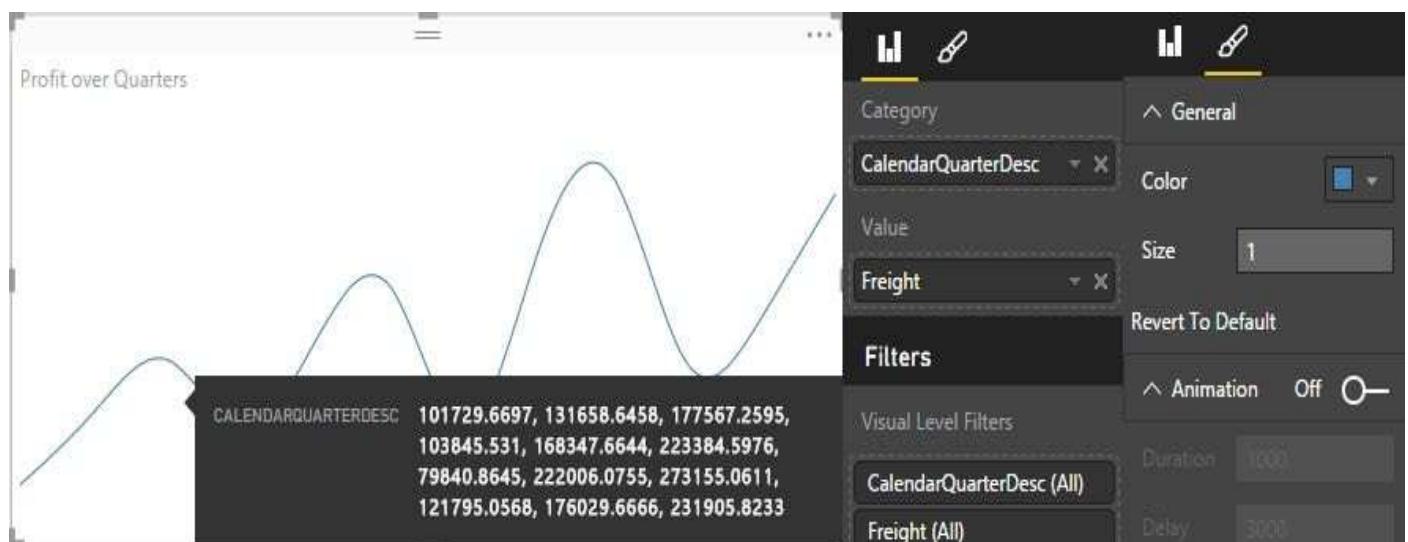
Now that you've learned about programming and testing custom visuals, let me walk you through the implementation steps of the sparkline visual.

## 13.3 Implementing Custom Visuals

A sparkline is a miniature graph, typically drawn without axes or coordinates. The term sparkline was introduced by Edward Tufte for “small, high resolution graphics embedded in a context of words, numbers, images”. Tufte describes sparklines as “data-intense, design-simple, word-sized graphics”. Sparklines are typically used to visualize trends over time, such as to show profit over the past several years. Although other Microsoft reporting tools, such as Excel and Reporting Services include sparkline elements, Power BI doesn’t have a sparkline visual. Yet, sparklines are commonly used on dashboards so I hope you’ll find my sparkline implementation useful not only for learning custom visuals but also for your real-life projects.

### 13.3.1 Understanding the Sparkline Visual

Sparklines come in different shapes and forms. To keep things simple, I decided to implement a “classic” smooth line sparkline that is shown in **Figure 13.13**.



**Figure 13.13** You can configure the sparkline using the Data and Format tabs .

#### *Understanding capabilities*

Once you import and add the sparkline to a report, you bind the sparkline to data using the Data tab of the Visualization pane. **Figure 13.13** shows that I’m aggregating a Freight field added the Value area by the CalendarQuarterDesc field added to the Category area. The resulting graph shows how freight fluctuates over quarters. You can use any field to group the data, not just a field from the Date table.

The sparkline supports several formatting options to customize its appearance. The General section lets you change the line color and width. The default properties are “steelblue” as a color and one pixel for the line width. The Animation section lets you turn on an animation effect that draws the line gradually from left to right. Although in general I advise against animations and other visual distractors in real-life reports, I wanted to emphasize the fact that Power BI visuals can support anything clients-side JavaScript can do. The Duration setting controls how fast the line draws (the default setting is 1,000 milliseconds) and the Delay settings controls the interval between redraws (the default is

3,000 milliseconds).

The sparkline also supports a tooltip. When you hover anywhere on the viewport, a tooltip pops up that shows the name of the field added to the Category area and the data point values.

#### ***Understanding limitations***

The main limitation of the current implementation is that the sparkline doesn't render multiple times in the same visualization, such as for each product category. This limitation also applies to Microsoft-provided visuals, such as the Gauge visual. Preferably, at some point Power BI would support a repeater visual, similar to the SSRS Tablix region. This would allow nesting the sparkline into other visualizations, such as a table, that could repeat the sparkline for each row. As Power BI stands now, the only way to implement this feature is to draw the visual for each category value. Although the sparkline doesn't repeat, it could be used to display multiple measures arranged either horizontally or vertically by adding it multiple times on the report.

Another limitation related to the one I've just discussed is that the sparkline supports only a single field in the Category area and a single field in the Value area. In other words, the sparkline is limited to one measure and one group. Continuing on the list of limitations, the tooltip displays the same information irrespective where you hover on the sparkline viewport, as opposed to showing just the value of the data point behind the cursor.

### **13.3.2 Implementing the IVisual Interface**

I implemented the sparkline using the Power BI Developer Tools. Let's start its implementation with the IVisual interface. Remember that IVisual has three key methods: *init()*, *update()*, and *destroy()*.

#### ***Implementing the init() method***

Power BI calls the *init()* method to give a chance to the visual to initialize itself. **Figure 13.14** shows its implementation.

---

```
165 public init(options: VisualInitOptions): void {
166 this.selectionManager = new SelectionManager({ hostServices: options.host });
167 this.root = d3.select(options.element.get(0));
168
169 this.svg = this.root
170 .append('svg')
171 .classed('sparkline', true)
172 .attr('height', options.viewport.height)
173 .attr('width', options.viewport.width);
174 }
```

**Figure 13.14** The *init()* method initializes the visual.

First, the code creates an instance of the SelectionManager which the host uses to communicate to the visual user interactions, such as clicking the graph. The sparkline doesn't do anything with user selection events but it's possible to extend it, such as to

navigate to another page or highlight a line segment. Line 167 initializes the D3.js framework with the DOM element the visual owns, which is passed to the *init()* method as a property of the VisualInitOptions parameter. Line 169 creates a *svg* HTML element and classes it as “sparkline”. It’s a good practice to create another element instead of using the root in the case when you might need to draw more elements in future. The code also sizes the *svg* element so that it occupies the entire viewport.

```

177 public update(options: VisualUpdateOptions) {
178 this.svg.selectAll("path").remove(); // clear existing line
179 if (!options.dataViews && !options.dataViews[0]) return;
180 var dataView = this.dataView = options.dataViews[0];
181 var viewport = options.viewport;
182 var viewModel: SparklineModel = Sparkline.converter(dataView);
183 var graph = this.svg;
184
185 // stop animation if graph is animating for update changes to take effect
186 this.stopAnimation();
187 // resize draw area to fit visualization frame
188 this.svg.attr({
189 'height': viewport.height,
190 'width': viewport.width
191 });
192
193 var data = viewModel.data;
194 // X scale fits values for all data elements; domain property will scale the graph width
195 var x = d3.scale.linear().domain([0, data.length-1]).range([0, viewport.width]);
196 // Y scale will fit values from min to max calibrated to the graph height
197 var y = d3.scale.linear().domain([Math.min.apply(Math, data), Math.max.apply(Math, data)]).range([0, viewport.height]);
198 // create a line
199 var line = d3.svg.line()
200 .interpolate("basis") // smooth line
201 // assign the X function to plot on X axis
202 .x(function(d,i) {
203 // enable the next line when debugging to output X coordinate
204 // console.log('Plotting X value for data point: ' + d + ' using index: ' + i + ' to be at: ' + x(i) + ' using xScale.');
205 return x(i);
206 })
207 .y(function(d) {
208 // enable the next line when debugging to output X coordinate
209 // console.log('Plotting Y value for data point: ' + d + ' to be at: ' + y(d) + " using yScale.");
210 return viewport.height-y(d); // values are plotted from the top so reverse the scale
211 })
212 // display the line by appending an svg:path element with the data line we created above
213 var path = this.svg.append("svg:path")
214 .attr("d", line(data))
215 .attr('stroke-width', function(d) { return viewModel.size })
216 .attr('stroke', function(d) { return viewModel.color});
217

```

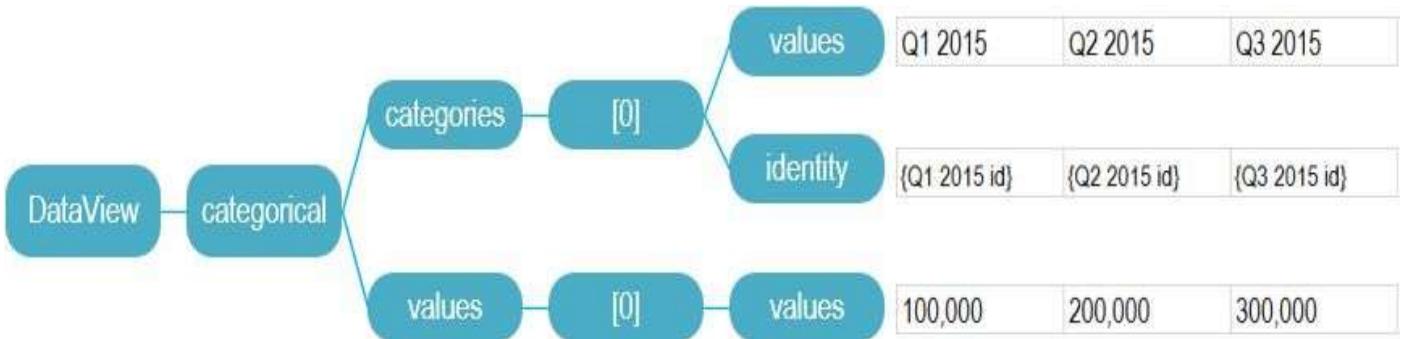
**Figure 13.15** The *update()* method draws the graph.

#### *Implementing the update() method*

The *update()* method is where the actual work of drawing the graph happens (see **Figure 13.15**). Line 178 removes the existing graph so that redrawing the sparkline doesn’t overlay what’s already plotted on the canvas and to avoid drawing new lines when the

visual is resized.

When the host calls the *update()* method, it passes the data as a *DataView* object. For example, if you add the *CalendarQuarter* field to the *Category* area and *SalesAmount* field to the *Value* area, the host will aggregate *SalesAmount* by quarter and pass the corresponding data representation and the metadata describing the columns under the *options.DataView* object. The definition of the *DataView* object is documented at <https://github.com/Microsoft/PowerBI-visuals/wiki/DataView-Introduction>. In our case, the *DataView* object might look like the example shown in **Figure 13.16**.



**Figure 13.16** When the host calls the *update()* method it passes a *DataView* object with the actual data.

Since the sparkline visual supports only one field in the *Category* area, there is only one element in the *DataView.categorical.categories* array. The *values* property returns the actual category values, such as Q1 2015. The *identity* property returns system-generated unique identifiers for each category value. The *DataView.categorical.values* property contains the values of the field added to the *Value* area. Because the sparkline visual supports only one field in the *Value* area, the *values* array has only one element.

Working directly with the *DataView* object is impractical. This is why line 182 calls a converter object, which converts the *DataView* object into a custom object for working with the data in a more suitable format. Using a converter is a recommended pattern since it allows you to organize the data just as you are to draw it, which makes your code focused on the task at hand and not on manipulating the data. For example, in our case the *converter.data* property on line 193 returns the data points as a JavaScript array.

The D3.js code starts at line 195. First, the code calibrates the X axis to plot the number of data points. Conveniently, D3.js supports quantitative scaling and the *d3.scale.linear.domain* property scales the X axis to fit the data points. Next, the code calibrates the Y axis to fit the values given the minimum and maximum data point values. Lines 199-211 plot the line. One cautionary note here is that the zero coordinate of Y axis starts at the top of the viewport. Therefore, line 210 inverts the data point Y coordinate. Line 214 draws the line using the user-specified line width and color.

#### *Animating the graph*

If the user turns on the *Animate* setting, the line constantly redraws itself using a configurable delay and redrawing speed. The code that animates the graph is shown in **Figure 13.17**. Line 233 checks if the animation effect is turned on. If so, it uses the JavaScript *setInterval()* function to call periodically the *redrawWithAnimation()* function.

D3.js and SVG make the task of animating the graph easy. Line 221 calls the SVG *getTotalLength()* function to calculate the length of the graph. The *stroke-dasharray* attribute lets you specify the length of the rendered part of the line. The *stroke-dashoffset* attribute lets you change where the *dasharray* behavior starts. Then the SVG *transition()* function is used to animate the path.

```

220 function redrawWithAnimation() {
221 var totalLength = path.node().getTotalLength();
222 graph.selectAll("path")
223 .data([data]) // set the new data
224 .attr("d", line)
225 .attr("stroke-dasharray", totalLength + " " + totalLength)
226 .attr("stroke-dashoffset", totalLength)
227 .transition()
228 .duration(viewModel.duration)
229 .ease("linear")
230 .attr("stroke-dashoffset", 0);
231 }
232
233 if (viewModel.animate)
234 {
235 Sparkline.timerID = setInterval(function() {
236 redrawWithAnimation();
237 }, viewModel.delay);
238 }
239 TooltipManager.addTooltip(this.svg, (tooltipEvent: TooltipEvent) => viewModel.toolTipInfo);
240 }
241 private stopAnimation() {
242 if (Sparkline.timerID)
243 {
244 clearInterval (Sparkline.timerID);
245 Sparkline.timerID = null;
246 }
247 }
```

**Figure 13.17** The graph supports animated line redrawing by calling repeatedly the *redrawWithAnimation* function.

The last line of the *update()* method (line 239) enables the tooltip support that allows the user to hover on top of the viewport and see the data point values in a tooltip. It calls the *converter.viewModel.toolTip* property which enumerates and concatenates the data point values.

```

var tooltipString: string = "";
var formatter = valueFormatter.create({ format:"0", value: 0});
for (var i=0; i < values.length; i++) {tooltipString += formatter.format(values[i]) + " "}// beautify tooltip values
toolTipInfo: [{

 displayName: dataViewCategorical.categories[0].source.displayName,

 value: values.join(", ") }]
```

#### **Implementing the *destroy()* method**

Remember that the host calls the *destroy()* method to give the visual a chance to release any resources that might result in memory leaks. Our implementation releases the D3.js

graph elements. It also releases the timer variable that holds a reference to timer identifier when the animation effect is used.

```
public destroy(): void {
 this.svg = null;
 this.root = null;
 this.timer = null;}
```

### 13.3.3 Implementing Capabilities

Each Power BI hosts uses the visual's capabilities to provide various extensions. For example, the report canvas uses this information to populate the Field and Formatting tabs in the Visualizations pane. For this to work, the custom visual needs to tell Power BI what data and formatting capabilities it supports.

#### *Advertising data capabilities*

**Figure 13.18** shows how the Sparkline visual advertises its data capabilities. The custom visual has a static property that inherits from the Power BI VisualCapabilities interface. The *dataRoles* property informs the host about the field areas the visual is expecting, while the *dataViewMappings* property describes how these fields relate to one another, and informs Power BI how it should construct the Fields tab areas. It can also inform the host about special conditions, such as that only one category value is supported.

On line 67, the Sparkline custom visual uses *dataRoles* to tell Power BI that it needs a Category area for grouping the data and a Value area for the measure. When the host interrogates the visual capabilities, it'll add these two areas to the Fields tab of the Visualizations pane.

```

66 public static capabilities: VisualCapabilities = {
67 dataRoles: [
68 {
69 name: 'Category',
70 kind: VisualDataRoleKind.Grouping,
71 displayName: 'Category',
72 },
73 {
74 name: 'Value',
75 kind: VisualDataRoleKind.Measure,
76 displayName: 'Value',
77 },
78]
79 ,
80 dataViewMappings: [
81 conditions: [
82 {
83 'Category': { max: 1 }, 'Value': { max: 1 }
84 },
85],
86 categorical: {
87 categories: [
88 for: { in: 'Category' },
89 dataReductionAlgorithm: { bottom: { count: 100 } }
90],
91 values: [
92 group:
93 {
94 by: "Series",
95 select: [{ bind: { to: 'Value' } }]
96 }
97]
98 }
99]
100 }

```

**Figure 13.18** The visual uses `VisualCapabilities.dataRoles` property to advertise its data capabilities.

On line 81, the custom visual uses `dataViewMappings` to instruct the host that the Category and Value areas can have only one field. To avoid performance degradation caused by plotting too many data points, line 86 specifies a bottom 100 data reduction condition to plot only the last 100 categorical values. So if the user adds the Date field from the Date table, only the last 100 dates will be displayed.

#### *Advertising formatting capabilities*

Custom visuals are not responsible for implementing any user interface for formatting the visual. Instead, they declare the formatting options they support using the `objects` property of the `VisualCapabilities` interface, and the host creates the UI for them. As it stands, Power BI supports three types of objects:

- Statically bound – These are formatting options that don't depend on the actual data, such as the line color.
- Data bound – These objects are bound to the number of data points. For example, the funnel chart allows you to specify the color of the individual data points.
- Metadata bound – These objects are bound to actual data fields, such as if you want to

color all the bars in a series of a bar chart in a particular color.

The Sparkline supports additional settings that allows the user to customize its appearance and animation behavior, as shown in **Figure 13.19**.

```
100 objects: {
101 general: {
102 displayName: data.createDisplayNameGetter('Visual_General'),
103 properties: {
104 fill: {
105 type: { fill: { solid: { color: true } } },
106 displayName: 'Color'
107 }
108 ,
109 size: {
110 type: { numeric: true },
111 displayName: 'Size'
112 }
113 },
114 },
115 labels: {
116 displayName: 'Animation',
117 properties: {
118 show: {
119 type: { bool: true },
120 displayName: data.createDisplayNameGetter('Visual_Show')
121 },
122 delay: {
123 type: { numeric: true },
124 displayName: 'Delay'
125 }
126 ,
127 duration: {
128 type: { numeric: true },
129 displayName: 'Duration'
130 }
131 }
```

**Figure 13.19** The visual uses the VisualCapabilities.objects property to advertise its formatting capabilities.

All the sparkline formatting settings are static. They are grouped in two sections: General and Animation (see **Figure 13.1** again). The *fill* property (line 104) allows the user to specify the line color. The type of this property is color. This will cause the host to show a color picker. The *displayName* property defines the name the user will see (“Color” in this case). The *Size* property is for the line width and has a numeric data type.

The labels section defines the animation settings. The *show* property (line 118) is a special Boolean property that allows the user to turn on or off the entire section. The *delay* property controls how often the line is redrawn, while the *duration* property controls the speed of redrawing the line.

#### *Enumerating capabilities*

Here is something important you need to know. The host won’t create UI for a capability until you write a code to let the host enumerates that capability! When the host discovers

the visual capabilities, it calls the *IVisual.enumerateObjectInstances()* method to obtain the values for each setting. And when the user changes a setting, the host calls this method again to push the new property values. **Figure 13.20** shows the implementation of this method.

The implementation of *enumerateObjectInstances* is straightforward. The host passes an options parameter and the *objectName* property tells us which object the host wants to enumerate. The code calls the appropriate *get* method to return the object value. When the user changes a setting the code calls the *instances.push* method to save the user selection. After the host calls *enumerateObjectInstances*, the host calls the *IVisual.update()* method so that the custom visual is redrawn with the new settings.

```
323 public enumerateObjectInstances(options: EnumerateVisualObjectInstancesOptions): VisualObjectInstance[] {
324 var instances: VisualObjectInstance[] = [];
325 var dataView = this.dataView;
326 switch (options.objectName) {
327 case 'general':
328 var general: VisualObjectInstance = {
329 objectName: 'general',
330 displayName: 'General',
331 selector: null,
332 properties: {
333 fill: Sparkline.getFill(dataView),
334 size: Sparkline.getSize(dataView)
335 }
336 };
337 instances.push(general);
338 break;
339 case 'labels':
340 var labels: VisualObjectInstance = {
341 objectName: 'labels',
342 displayName: 'Animation',
343 selector: null,
344 properties: {
345 show: Sparkline.getAnimate(dataView),
346 duration: Sparkline.getDuration(dataView),
347 delay: Sparkline.getDelay(dataView)
348 }
349 };
350 instances.push(labels);
351 break;
352 }
353 return instances;
354 }
```

**Figure 13.20** The host calls *enumerateObjectInstances* to get and set the visual capabilities .

## 13.4 Deploying Custom Visuals

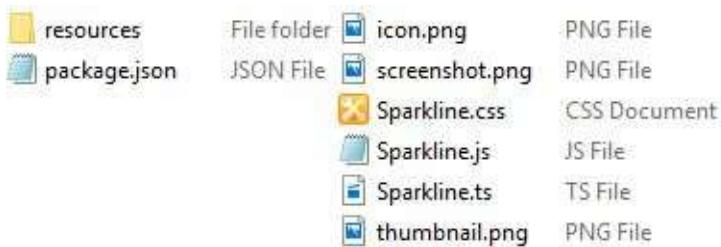
Once you test the custom visual, it's time to package and deploy it, so that your users can start using it to visualize data in new ways. If you want to make the visual publicly available, consider also submitting it to the Power BI visuals gallery.

### 13.4.1 Packaging Custom Visuals

So that end users can import your custom visual in Power BI Service and Power BI Desktop, you need to package the visual as a \*.pbviz file.

#### *Understanding visual packages*

A pbviz file is a standard zip archive. If you rename the file to have a zip extension and double-click it, you'll see the structure shown in **Figure 13.21**.



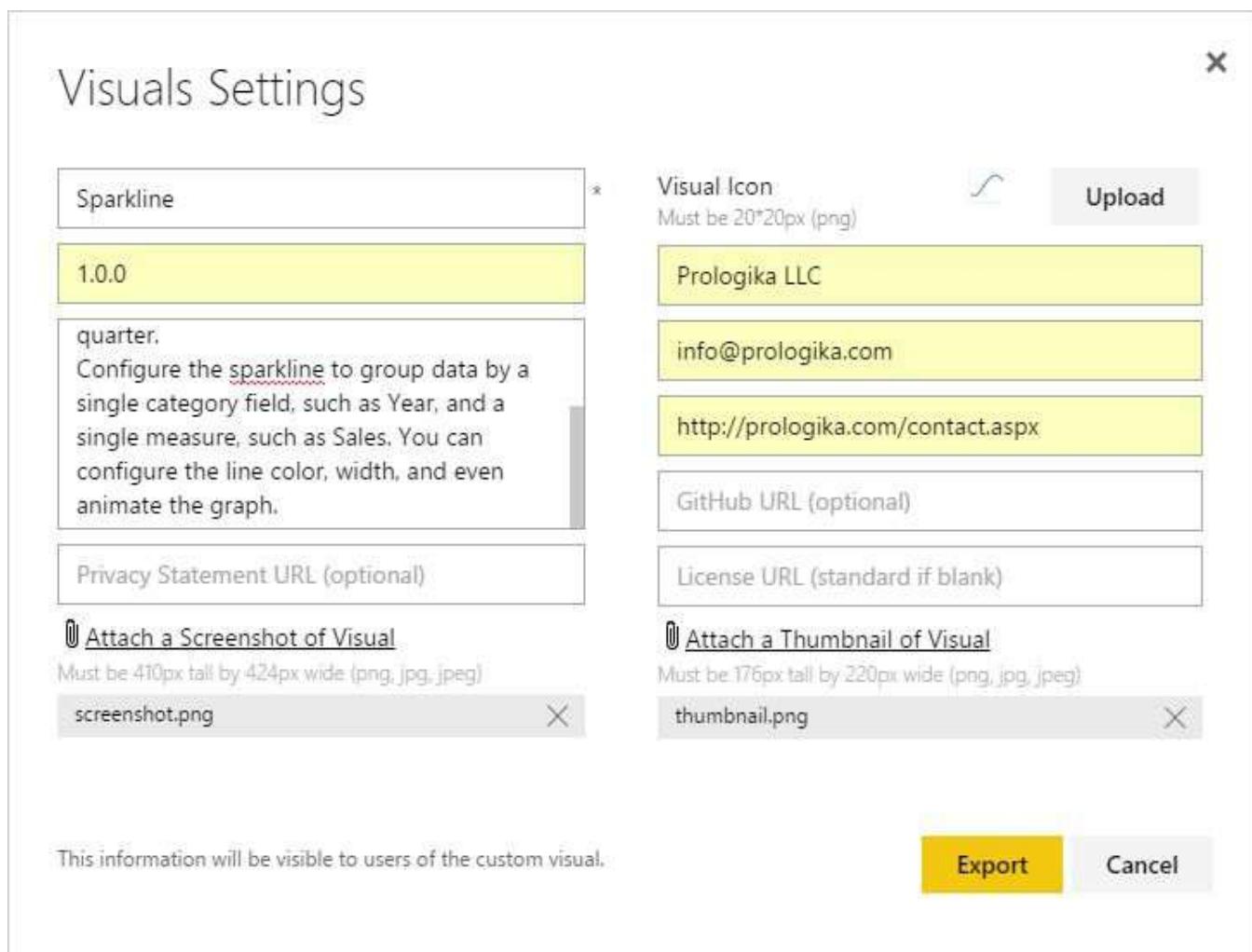
**Figure 13.21** A \*.pbviz file a zip archive file that packages the visual TypeScript and JavaScript code, CSS file and icon image.

The package.json file is the visual manifest which indicates which files are included and what properties you specified when you exported the visual. The resources folder includes six files. If you specify a visual image when you exported the visual, the icon file is the image file. The screenshot file is an image file (410 pixels tall by 424 pixels wide) that the Power BI visuals gallery shows when the user clicks the visual. The thumbnail file (176 pixels tall by 220 pixels wide) is the image the users see on the main page of the Power BI visuals gallery. The CSS file contains the CSS styles that you specified in the CSS window of the Dev Tools. The \*.ts file contains the visual TypeScript code and the \*.js file is the resulting JavaScript source that will be executed when you use the visual in Power BI.

#### *Exporting custom visuals*

The Power BI Developer Tools make it easy to export and package the visual.

- 1.In the Power BI Developer Tools, click the Export button (see **Figure 13.10** again).
- 2.Fill in the Visual Settings window. **Figure 13.22** shows the settings that I specified when I exported the Sparkline visual.



**Figure 13.22** Use the Visual Settings window to enter information that you want to distribute with the visual.

3.Click Export. Dev Tools packages the visual and downloads the package to your machine.

If only users within your organization will use the visual, you are done! You just need to distribute the \*.pbviz file to your users so they can import it in Power BI Desktop or Power BI Service.

#### ***Publishing to Power BI visuals gallery***

If you would like to make your visual publicly available, consider submitting it to the Power BI visuals gallery. To learn more about how to do so:

- 1.Open your web browser and navigate to the Power BI visuals gallery (<https://app.powerbi.com/visuals>).
- 2.Click the “Learn how to submit visuals” button.
- 3.Scroll down the next page and then click the “Submit a visual” button.

As of the time of writing this book, you can submit a visual by sending an e-mail to pbivizsubmit@microsoft.com and attaching your \*.pbviz file. Microsoft will review your visual for any issues and the Power BI team will let you know if and when your visual will be published.

## 13.4.2 Using Custom Visuals

Once downloaded, custom visuals can be added to a report in Power BI Service or Power BI Desktop. As I explained in Chapter 4, you add a custom visual to a report by clicking the ellipsis (...) button in the Visualizations pane.

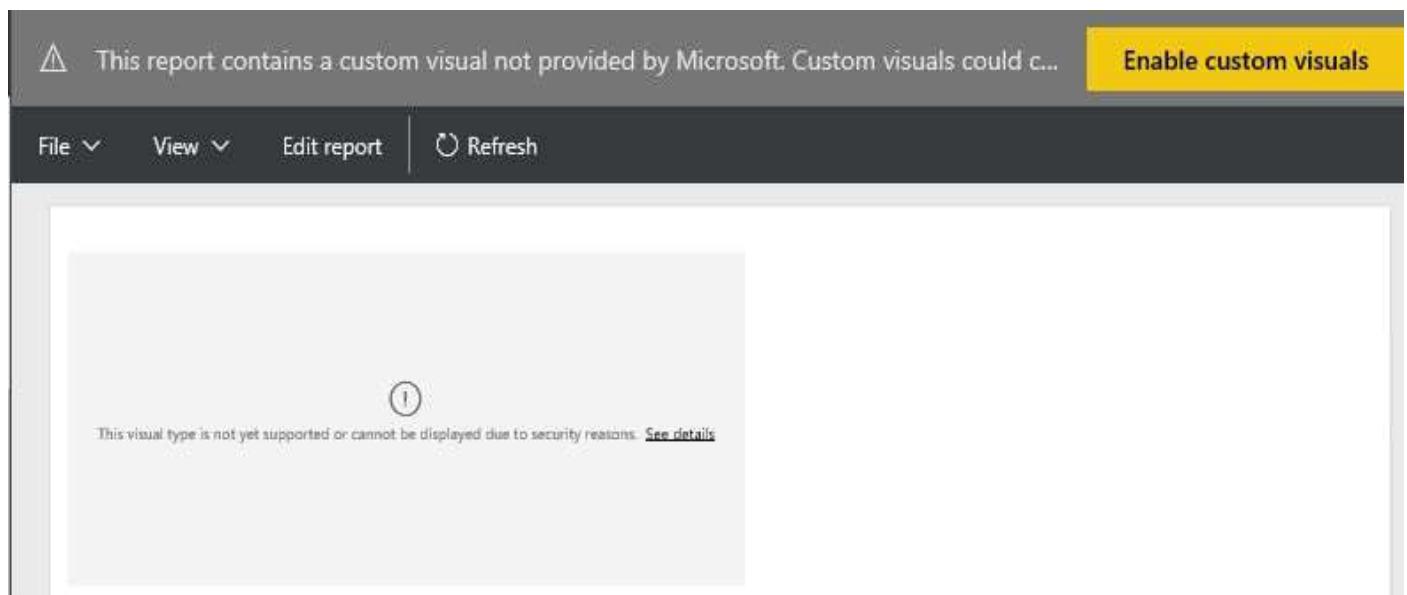
### *Understanding import limitations*

As it stands, Power BI imports the custom visual code into the report. Therefore, the visual exists only within the hosting report that imports the visual. If you create a new report, you'll find that the Visualizations pane doesn't show custom visuals. You must re-import the custom visuals that the new report needs.

**NOTE** As a best practice, you should test a custom visual for privacy and security vulnerabilities by using the Microsoft recommendations at <https://powerbi.microsoft.com/en-us/documentation/powerbi-custom-visuals-review-for-security-and-privacy>. I recommend you compare the TypeScript and JavaScript files have the same code and test the JavaScript code with an anti-virus software.

### *Understanding security restrictions*

As another security measure, Power BI doesn't trust custom visuals and it doesn't load them by default when you open a report. Instead, Power BI displays the prompt shown in **Figure 13.23** that warns you about custom code and asks you to confirm that you want to enable custom visuals on the report. Only after you confirm the prompt, the report will display the visual.



**Figure 13.23** Power BI asks you to confirm that you want to enable custom visuals.

The security prompt will be displayed each time you navigate to a report with a custom visual. Currently, there isn't a global or a report-specific setting to turn off the prompt.

## 13.5 Summary

The Microsoft presentation framework is open source to let web developers extend the Power BI visualization capabilities and to create their own visuals. Consider implementing a custom visual when your presentation requirements go beyond the capabilities of the Microsoft-provided visuals. Custom visuals help you convey information as graphics elements and images. Any data insights that can be coded and rendered with JavaScript and client-side presentation frameworks, such as D3.js and SVG, can be implemented as a custom visual and plotted on Power BI reports.

You create custom visuals by writing TypeScript code that implements the Power BI IVisual interface. You can code a custom visual in Visual Studio or in the Power BI Developer Tools. The custom visual advertises its capabilities to the host. The host is responsible for configuring the Visualizations pane to let end users configure the visual. Once the visual is ready and tested, you can export it to a \*.pbviz file, and then import it in Power BI Service or Power BI Desktop. You can also share your custom visuals with the community by submitting them to the Power BI visuals gallery.

With this chapter we've reached the last stop of our Power BI journey. I sincerely hope that this book has helped you understand how Power BI can be a powerful platform for delivering pervasive data analytics. As you've seen, Power BI has plenty to offer to all types of users who are interested in BI:

- Information worker – You can use content packs and the Power BI Service Get Data feature to gain immediate insights without modeling.
- Data analyst – You can build sophisticated BI models for self-service data exploration with Power BI Desktop or Excel. And then you can share these models with your coworkers by publishing these models to Power BI.
- BI or IT pro – You can establish a trustworthy environment that promotes team collaboration. And you can implement versatile solutions that integrate with Power BI, such as solutions for descriptive, predictive and real-time BI.
- Developer – Thanks to the Power BI open architecture, you can extend the Power BI visualization capabilities with custom visuals and integrate your custom apps with Power BI.

Of course, that's not all! Remember that Power BI is a part of a holistic vision that Microsoft has for delivering cloud and on-premises data analytics. When planning your on-premises BI solutions, consider the Microsoft public reporting roadmap at <http://bit.ly/msreportingroadmap>. Keep in mind that you can use both Power BI (cloud-based data analytics) and the SQL Server box product on-premises to implement synergistic solutions that bring your data to life!

Don't forget to download the source code from <http://bit.ly/powerbobook> and stay in touch with me on the book discussion list. Happy data analyzing with Power BI!

# Appendix A

---

## Glossary of Terms

The following table lists the most common BI-related terms and acronyms used in this book.

Term	Acronym	Description
Analysis Services Connector		Connectivity software that allows Power BI to connect to on-premises SSAS models.
Analysis Services Tabular		An instance of SQL Server 2012 Analysis Services that's configured in Tabular mode and is capable of hosting tabular models for organizational use.
Azure Marketplace		The Windows Azure Marketplace is an online market buying, and selling finished software as a Service (SaaS) applications and premium datasets.
Business Intelligence Semantic Model	BISM	A unifying name that includes both multidimensional (OLAP) and tabular (relational) features of Microsoft SQL Server 2012 Analysis Services.
Content pack		A packaged set of dashboards, reports, and datasets from popular cloud services or from Power BI content (see organizational content pack)
Corporate BI		Same as Organizational BI.
Cube		An OLAP structure organized in a way that facilitates data aggregation, such as to answer queries for historical and trend analysis.
Dashboard		A Power BI page that can combine visualizations from multiple reports to provide a summary view.
Data Analysis Expressions	DAX	An Excel-like formula language for defining custom calculations and for querying tabular models.
Data model		A BI model designed with Power BI Desktop or Analysis Services.
Dataset		The definition of the data that you connect to in Power BI, such as a dataset that represents the data you import from an Excel file.
Descriptive analytics		A type of analytics that is concerned about analyzing past history.
DirectQuery		A data connectivity configuration that allows Power BI to generate and send queries to the data source without importing the data.
Dimension (lookup) table		A table that represents a business subject area and provides contextual information to each row in a fact table, such as Product, Customer, and Date.
Enterprise Power BI Gateway		Connectivity software that allows Power BI to query directly on-premises data sources.
Extraction, transformation, loading	ETL	Processes extract from data sources, clean the data, and load the data into a target database, such as data warehouse.
Fact table		A table that keeps a historical record of numeric measurements (facts), such as the ResellerSales in the Adventure Works model.
Group		A Power BI group is a security mechanism to simplify access to content.
HTML5		A markup language used for structuring and presenting content on the World Wide Web.
Key Performance Indicator	KPI	A key performance indicator (KPI) is a quantifiable measure that is used to measure the company performance, such as Profit or Return On Investment (ROI).
Measure		A business calculation that is typically used to aggregate data, such as SalesAmount, Tax, OrderQuantity.
		The OLAP path of BISM that allows BI professionals to implement

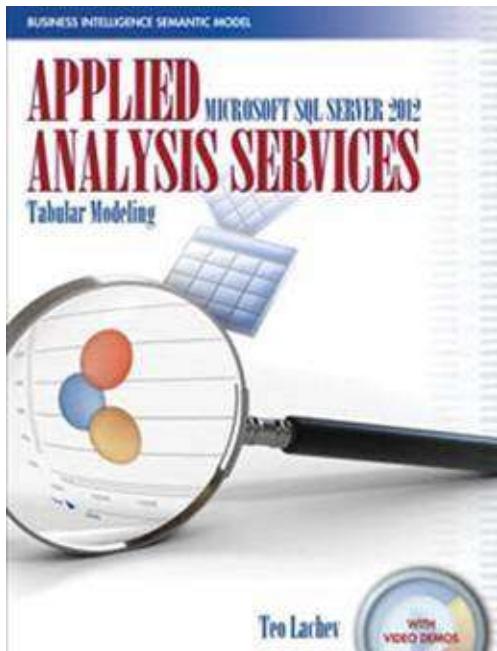
Multidimensional		multidimensional cubes.
Multidimensional Expressions	MDX	A query language for Multidimensional for defining custom calculations and querying OLAP cubes.
Office 365		A cloud-hosted platform of Microsoft services and products, such as SharePoint Online and Exchange Online.
OneDrive and OneDrive for Business		Cloud storage for individuals or businesses to upload files.
Online Analytical Processing	OLAP	A system that is designed to quickly answer multidimensional analytical queries in order to facilitate data exploration and data mining.
Organizational content pack		A packaged set of Power BI dashboards, reports, and datasets that can be distributed to many users.
Personal BI		Targets business users and provides tools for implementing BI solutions for personal use, such as PowerPivot models, by importing and analyzing data without requiring specialized skills.
Personal Power BI Gateway		Connectivity software that allows Power BI to refresh data from on-premises data sources.
Power BI		A data analytics platform for self-service, team, and organizational BI that consists of Power BI Service, Power BI Mobile and Power BI Desktop products.
Power BI Desktop		A free desktop tool for creating self-service data models that can be uploaded to Power BI Service.
Power BI Mobile		Native mobile applications for viewing and annotating Power BI content on mobile devices.
Power BI Portal		The user interface of Power BI Service that you see when you go to powerbi.com.
Power BI Service		The cloud-based service of Power BI (powerbi.com). The terms Power BI and Power BI Service are used interchangeably.
Power Map		An Excel add-in for 3D geospatial reporting.
Power View		A SharePoint-based reporting tool that allows business users to author interactive reports from PowerPivot models and from organizational tabular models.
Power Pivot for Excel		A free add-in that extends the Excel capabilities to allow business users to implement personal BI models.
Power Pivot for SharePoint		Included in SQL Server 2012, PowerPivot for SharePoint extends the SharePoint capabilities to support PowerPivot models.
Power Query		An Excel add-in for transforming and shaping data.
Predictive analytics		Type of analytics that is concerned with discovering patterns that aren't easily discernible
Questions & Answers	Q&A	A Power BI feature that allows users to type natural questions to get data insights.
Self-service BI		Same as Personal BI.
SharePoint Products and Technologies	SharePoint	A server-based platform for document management and collaboration that includes BI capabilities, such as hosting and managing PowerPivot models, reports, and dashboards.
SQL Server Analysis Services	SSAS	A SQL Server add-on, Analysis Services provides analytical and data mining services. The Business Intelligence Semantic Model represents the analytical services.
SQL Server Integration Services	SSIS	A SQL Server add-on, Integration Services is a platform for implementing extraction, transformation, and loading (ETL) processes.
SQL Server Management Studio	SSMS	A management tool that's bundled with SQL Server that allows administrators to manage Database Engine, Analysis Services, Reporting Services and Integration Services instances.
SQL Server Reporting Services	SSRS	A SQL Server add-on, Reporting Services is a server-based reporting platform for the creation, management, and delivery of standard and ad hoc reports.
Snowflake schema		Unlike a star schema, a snowflake schema has some dimension tables that relate to other dimension tables and not directly to the fact table.
Star schema		A model schema where a fact table is surrounded by dimension tables and these dimension tables reference directly the fact table.
		Tabular is the relational side of BISM that allows business users and BI

Tabular	professionals to implement relational-like (tabular) models.
Team BI	Provides tools to allow business users to share BI solutions that they create with co-workers.
Tile	A dashboard section that can be pinned from an existing report or produced with Q&A.
Visualization	A visual representation of data on a Power BI report, such as a chart or map.
Workspace	A Power BI content area that is allocated for either an individual (My Workspace) or a team
xVelocity	xVelocity is a columnar data engine that compresses and stores data in memory.

## Also by Teo Lachev

---

### Applied Microsoft SQL Server 2012 Analysis Services (Tabular Modeling)



**ISBN:** 978-0976635352

**Publisher website:** <http://bit.ly/thebismbook>

**Amazon:** <http://amzn.to/21wd3J8>

**B&N:** <http://bit.ly/1PwXWeV>

An insightful tour that provides an authoritative yet independent view of this exciting technology, this guide introduces the Tabular side of the innovative Business Intelligence Semantic Model (BISM) that promotes rapid professional and self-service BI application development. Business analysts and power users will learn how to integrate data from multiple data sources, implement self-service BI applications with Excel, and deploy them to SharePoint. Business intelligence professionals and database administrators will discover how to build corporate solutions powered by BISM Tabular, delivering supreme

performance with large data volumes, and how to implement a wide range of solutions for the entire spectrum of personal-team-organizational BI needs.

Ideal for experienced BI professionals or beginners, this book features step-by-step instructions and demos for building reports and dashboards, deploying and managing BISM, and integrating data from various sources.

Also, check out our online and onsite BI training classes at  
<http://prologika.com/training/training.aspx>.

- Analysis Services
- Reporting Services
- Power BI
- Excel data analytics
- ... and much more!