

THE EXPERT'S VOICE® IN BUSINESS INTELLIGENCE

Pro Power BI Desktop

Free interactive data analysis with
Microsoft Power BI

Adam Aspin

EXTRAS ONLINE

Apress®

Pro Power BI Desktop



Adam Aspin

Apress®

Pro Power BI Desktop

Adam Aspin
Stoke-on-Trent
Staffordshire, United Kingdom

ISBN-13 (pbk): 978-1-4842-1804-4
DOI 10.1007/978-1-4842-1805-1

ISBN-13 (electronic): 978-1-4842-1805-1

Library of Congress Control Number: 2016938680

Copyright © 2016 by Adam Aspin

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director: Welmoed Spahr

Lead Editor: Jonathan Gennick

Development Editor: Douglas Pundick

Technical Reviewer: Ian Rice

Editorial Board: Steve Anglin, Pramila Balen, Louise Corrigan, Jim DeWolf, Jonathan Gennick,

Robert Hutchinson, Celestin Suresh John, Michelle Lowman, James Markham, Susan McDermott,

Matthew Moodie, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Gwenan Spearing

Coordinating Editor: Jill Balzano

Copy Editor: Kim Burton-Weisman

Compositor: SPi Global

Indexer: SPi Global

Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springer.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a Delaware corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales-eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this text is available to readers at www.apress.com. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/.

Printed on acid-free paper

To the memories of my mother and grandmother

Contents at a Glance

About the Author	xxiii
About the Technical Reviewer	xxv
Acknowledgments	xxvii
Introduction	xxix
■ Chapter 1: Introduction to Power BI Desktop	1
■ Chapter 2: Discovering and Loading Data with Power BI Desktop	35
■ Chapter 3: Transforming Datasets	79
■ Chapter 4: Data Cleansing	109
■ Chapter 5: Data Mashup	131
■ Chapter 6: Creating a Data Model.....	175
■ Chapter 7: Extending the Data Model with Calculated Columns	205
■ Chapter 8: Adding Measures to the Data Model.....	239
■ Chapter 9: Analyzing Data over Time	267
■ Chapter 10: Text-Based Visualizations	301
■ Chapter 11: Charts in Power BI Desktop	333
■ Chapter 12: Other Types of Visuals.....	371
■ Chapter 13: Filtering Data	401

■ CONTENTS AT A GLANCE

■ Chapter 14: Using Slicers	429
■ Chapter 15: Enhancing Dashboards	451
■ Chapter 16: PowerBI.com	471
■ Appendix A: Sample Data	501
Index.....	503

Contents

About the Author	xxiii
About the Technical Reviewer	xxv
Acknowledgments	xxvii
Introduction	xxix
■ Chapter 1: Introduction to Power BI Desktop	1
The Microsoft Self-Service Business Intelligence Solution.....	2
Import Data from Diverse Sources	2
Model Your Data.....	3
Creating Dashboards and Reports.....	3
Power BI Desktop Files.....	3
Installing Power BI Desktop	4
Removing Power BI Desktop	9
Running Power BI Desktop	10
A First Power BI Desktop Dashboard.....	10
Loading the Source Data	11
The Data Load Process.....	14
The Navigator Window.....	15
The Navigator Data Preview	15
Modifying Data	16
The Power BI Desktop Window	16
Your First Visualizations	17
Display Available Fields	18
Add a Matrix of Sales per Country by Year	19

■ CONTENTS

Add a Column Chart of Delivery Charge by Model.....	21
Add a Map of Labor Cost by Country	23
Add a Card Showing the Total Cost of Spare Parts.....	24
Add a Slicer by Make	25
Arranging the Dashboard.....	26
Interactivity in Dashboards.....	28
Creating and Modifying Reports.....	29
Adding Pages.....	30
Renaming Pages.....	30
Deleting Pages.....	30
Moving Pages	31
Duplicating Pages.....	31
Scrolling through Collections of Pages.....	31
Importing Excel and Power View Items.....	31
Conclusion.....	34
■ Chapter 2: Discovering and Loading Data with Power BI Desktop	35
Data Sources	36
File Sources.....	37
Databases.....	37
Azure	38
Other Sources	39
Loading Data	40
Web Pages.....	40
CSV Files.....	44
Text Files.....	47
XML Files	49
Excel	51
Microsoft Access Databases.....	52
Relational Databases: SQL Server	53
Automatically Loading Related Tables	55
Database Options	56

Microsoft SQL Server Analysis Services Data Sources	63
Analysis Services Cube Tools	66
Microsoft SQL Server Analysis Services Tabular Data Sources.....	68
Other Data Sources	71
Reusing Recent Data Sources	71
Reusing a Data Source	71
Pinning a Data Source	73
Refreshing Data.....	74
Refreshing a Query	74
Refreshing All the Queries in a Power BI Desktop File	74
Old Data.....	74
Connection Security	75
Conclusion.....	77
■ Chapter 3: Transforming Datasets.....	79
Power BI Desktop Queries.....	80
Editing Data After a Data Load.....	80
Transforming Data Before Loading	82
Query or Load?	82
The Power BI Desktop Query Editor	83
The Applied Steps List	84
The Power BI Desktop Query Editor Ribbons.....	85
Dataset Shaping	91
Renaming Columns.....	91
Reordering Columns	92
Removing Columns.....	93
Merging Columns	93
Removing Records.....	96
Removing Duplicate Records.....	99
Sorting Data.....	100
Reversing the Row Order.....	101

Filtering Data.....	101
Selecting Specific Values.....	101
Finding Elements in the Filter List.....	102
Filtering Text Ranges	103
Filtering Numeric Ranges	104
Filtering Date and Time Ranges.....	104
Filtering Data.....	105
Conclusion.....	106
■ Chapter 4: Data Cleansing	109
Viewing a Full Record.....	109
Power BI Desktop Query Editor Context Menus	110
Changing Data Type	111
Detecting Data Types.....	113
Replacing Values	113
Transforming Column Contents	115
Filling Down.....	123
Using the First Row As Headers	125
Grouping Records.....	126
Conclusion.....	129
■ Chapter 5: Data Mashup	131
The Power BI Desktop View Ribbon	132
Extending Data	132
Duplicating Columns	133
Splitting Columns	133
Splitting Column by a Delimiter	134
Splitting Columns by Number of Characters.....	136
Merging Columns	137
Custom Columns	138
Index Columns.....	140
Merging Data.....	142

Adding Data	142
Aggregating Data During a Merge Operation.....	145
Types of Join.....	149
Joining on Multiple Columns	150
Preparing Datasets for Joins	152
Correct and Incorrect Joins	152
Examining Joined Data.....	153
The Expand and Aggregate Buttons	154
Appending Data.....	156
Adding the Contents of One Query to Another	156
Adding Multiple Files from a Source Folder	157
Changing the Data Structure	161
Unpivoting Tables.....	161
Pivoting Tables.....	163
Transposing Rows and Columns.....	164
Managing the Transformation Process.....	164
Modifying a Step.....	165
Renaming a Step	165
Deleting a Step or a Series of Steps.....	166
Adding a Step	167
Altering Process Step Sequencing	167
An Approach to Sequencing	167
Error Records.....	168
Removing Errors	168
Managing Queries	168
Organizing Queries	169
Grouping Queries	169
Duplicating Queries	171
Referencing Queries	171
Add a Column As a New Query	171

■ CONTENTS

Enable Data Load.....	172
Enable Data Refresh.....	172
Pending Changes	173
Copying Data from Power BI Desktop	174
Conclusion.....	174
■ Chapter 6: Creating a Data Model.....	175
Data Modeling in the Power BI Desktop Environment.....	175
The Power BI Desktop Data View	176
Data Model or Query?.....	176
The Power BI Desktop Data View Ribbons	177
The Modeling Ribbon.....	177
Managing Power BI Desktop Data.....	178
Manipulating Tables.....	179
Manipulating Columns.....	180
Power BI Desktop Data Types.....	183
Formatting Power BI Desktop Data.....	184
Currency Formats	185
Preparing Data for Dashboards	187
Categorize Data	187
Apply a Default Summarization	188
Define Sort by Columns	189
Sorting Data in Power BI Desktop Tables	190
Adding Hierarchies	190
Designing a Power BI Desktop Data Model	191
Data View and Relationship View	192
Relationship View Display Options	193
Creating Relationships.....	194
Creating Relationships Manually	196
Creating Relationships Automatically	199
Deleting Relationships.....	199

Managing Relationships	200
Deactivating Relationships	201
Advanced Relationship Options	201
Conclusion.....	203
■ Chapter 7: Extending the Data Model with Calculated Columns	205
Types of Calculations.....	206
New Columns	206
Naming Columns	207
Concatenating Column Contents.....	208
Tweaking Text.....	210
Simple Calculations.....	212
Math Operators.....	213
Rounding Values	214
Calculating Across Tables.....	216
Choosing the Correct Table for Linked Calculations	217
Cascading Column Calculations	217
Refreshing Data	218
Using Functions in New Columns.....	219
Safe Division.....	219
Counting Reference Elements	220
Statistical Functions	221
Applying a Specific Format to a Calculation.....	222
Simple Logic: the IF() Function	226
Exception Indicators	226
Creating Alerts	227
Flagging Data	228
Nested IF() Functions.....	229
Creating Custom Groups Using Multiple Nested IF() Statements	230
Multiline Formulas.....	232

Complex Logic	232
Formatting Logical Results.....	236
Making Good Use of the Formula Bar.....	237
Conclusion.....	237
■ Chapter 8: Adding Measures to the Data Model	239
A First Measure: Number of Cars Sold	239
Basic Aggregations in Measures	241
Using Multiple Measures.....	243
Cross-Table Measures.....	245
More Advanced Aggregations.....	248
Filter Context.....	251
Row Context	251
Query Context.....	251
Filter Context	252
Filtering Data in Measures	252
Simple Filters	252
Text Filters	253
Numeric Filters	254
More Complex Filters	256
Multiple Criteria in Filters	256
Using Multiple Filters.....	257
Calculating Percentages of Totals	258
A Simple Percentage	258
Removing Multiple Filter Elements.....	259
Visual Totals.....	261
The ALLEXCEPT() Function.....	262
Filtering on Measures.....	263
Displaying Rank.....	265
A Few Comments and Notes on Using Measures.....	266

Calculation Options	266
Conclusion.....	266
■Chapter 9: Analyzing Data over Time	267
Simple Date Calculations	267
Date and Time Formatting	270
Calculating the Age of Cars Sold	273
Calculating the Difference Between Two Dates.....	274
Adding Time Intelligence to a Data Model.....	276
Creating and Applying a Date Table.....	277
Creating the Date Table	277
Adding Sort By Columns to the Date Table	280
Date Table Techniques	281
Adding the Date Table to the Data Model	282
Applying Time Intelligence	284
YearToDate, QuarterToDate, and MonthToDate Calculations	284
Analyze Data As a Ratio over Time	286
Comparing a Metric with the Result from a Range of Dates	288
Comparisons with Previous Time Periods	291
Calculating Sales for the Previous Year	291
Comparison with a Parallel Period in Time.....	293
Comparing Data from Previous Years	293
Comparing with the Same Date Period from a Different Quarter, Month, or Year	296
Rolling Aggregations over a Period of Time.....	298
Conclusion.....	300
■Chapter 10: Text-Based Visualizations	301
Power BI Desktop Dashboards	302
Switching to Report View	302
Tables	302
Creating a Basic Table	303
Deleting a Table	305

■ CONTENTS

Copying a Table.....	306
Changing the Table Size and Position.....	306
Changing Column Order.....	307
Removing Columns from a Table	307
Types of Data.....	308
Enhancing Tables	309
Row Totals	309
Formatting Columns of Numbers.....	312
Font Sizes in Tables	312
Changing Column Widths.....	312
Adding and Formatting Titles.....	313
Modifying the Table Background	315
Table Borders.....	316
Sorting by Column	317
Table Granularity	318
Creating a Matrix.....	319
Row Matrix	319
Column Matrix	322
Sorting Data in Matrices.....	323
Cards	324
Formatting Cards.....	325
Multirow Cards	328
Formatting Multirow Cards.....	330
Switching Between Table Types	331
Conclusion.....	332
■ Chapter 11: Charts in Power BI Desktop	333
A First Chart	333
Creating a First Chart.....	334
Deleting a Chart.....	337
Basic Chart Modification.....	337

Basic Chart Types	338
Column Charts	338
Line Charts	339
Pie Charts	339
Essential Chart Adjustments	340
Resizing Charts.....	341
Repositioning Charts	342
Sorting Chart Elements.....	342
Donut Charts	344
Funnel Charts	344
Multiple Data Values in Charts	345
100% Stacked Column and Bar Charts	348
Scatter Charts	350
Scatter Charts to Display Flattened Hierarchies.....	351
Bubble Charts.....	352
Waterfall Charts.....	354
Dual-Axis Charts.....	355
Line and Clustered Column Chart.....	355
Line and Stacked Column Chart	356
Data Details	357
Drill Down.....	358
Enhancing Charts	361
Chart Legends	361
Chart Title	363
Chart Data Labels	363
Chart Background.....	364
Data Colors	365
Axis Modification	365

■ CONTENTS

Chart Borders	367
Chart Aspect Ratio	367
Bubble Chart Play Axis	368
Conclusion.....	369
■ Chapter 12: Other Types of Visuals.....	371
Maps.....	371
Bing Maps.....	372
Maps in Power BI Desktop.....	372
Using Geographical Data.....	374
Geographical Data Types	376
Adjusting the Map Display in Power BI Desktop	376
Positioning the Map Elements.....	376
Zooming In or Out.....	377
Multivalue Series.....	377
Highlighting Map Data	378
Filled Maps	379
Formatting Maps	379
Tree Maps.....	382
Formatting Tree Maps.....	384
Gauges	385
Formatting Gauges	386
Additional Visuals	387
The Power BI Visuals Gallery	388
Loading Custom Visuals	390
Enabling Custom Visuals	393
A Rapid Overview of a Selection of Custom Visuals.....	394
Aster Plots	394
Radar Charts.....	394
Bullet Charts	395
Word Clouds.....	396

Sunburst Charts	396
Streamgraphs	397
Tornado Charts	397
Histograms	398
Chord Charts	399
Sankey Diagrams	399
Conclusion	400
Chapter 13: Filtering Data	401
Filters	402
Visual-Level Filters	402
The Filters Well	402
Adding Filters	403
Applying Filters	404
The (All) Filter	406
Clearing Filters	407
Filtering Different Data Types	407
Numeric Data	407
Numeric Filter Options	409
Date and Time Data	410
Date and Time Filters	411
Date Filter Options	414
Other Data Types	414
Advanced Text Filters	415
Applying an Advanced Text Filter	415
Clearing an Advanced Filter	416
Reverting to Basic Filtering	417
Text Filter Options	417
Specific Visualization-Level Filters	417
Multiple Filters	419
Page-Level Filters	419

■ CONTENTS

Report-Level Filters.....	420
 Removing Filters	421
Filter Field Reuse.....	423
 Using the Filter Hierarchy.....	425
 Filtering Tips.....	425
Don't Filter Too Soon.....	425
Annotate, Annotate, Annotate.....	426
Avoid Complex Filters	426
 Conclusion.....	427
■ Chapter 14: Using Slicers	429
 Slicers	429
Adding a Slicer	430
Applying Slicers.....	431
Clearing a Slicer	432
Deleting a Slicer	432
Modifying a Slicer.....	432
 Formatting Slicers.....	433
Slicer Orientation.....	433
Modifying the Outline	434
Adjusting Selection Controls.....	434
Setting the Exact Size and X and Y coordinates of a Slicer	435
Slicer Header	435
Slicer Items.....	435
 Using Charts As Slicers	436
Charts As Slicers.....	436
Highlighting Chart Data	438
Cross-Chart Highlighting	439
Highlighting Data in Bubble Charts.....	442
 Charts As Filters	444
Column and Bar Charts As Filters.....	446

Specifying Visual Interactions	448
Choosing the Correct Approach to Interactive Data Selection.....	450
Conclusion.....	450
Chapter 15: Enhancing Dashboards	451
Adding Text Boxes to Annotate a Report	451
Adding a Text Box	451
Moving Text Boxes.....	452
Formatting a Text Box.....	452
Adding a Hyperlink	454
Removing a Hyperlink.....	454
Deleting Text Boxes	454
Modifying the Page Background Color	454
Images.....	455
Image Sources.....	456
Adding an Image.....	457
Resizing Images	458
Formatting Images	459
Background Images.....	459
Images in Charts.....	460
Some Uses for Independent Images.....	461
Adding Shapes	461
Formatting Shapes	462
Removing Shapes.....	465
Standardizing Shapes.....	465
Organizing Visuals on the Page	465
Layering Visuals.....	466
Aligning Visuals	467
Distributing Visuals.....	468
Exporting Data from a Visualisation	469
Conclusion.....	469

■ Chapter 16: PowerBI.com	471
Publishing Reports to PowerBI.com	471
Creating a Power BI Account	472
Using Power BI Desktop Files in PowerBI.com	475
Logging on to PowerBI.com.....	476
Adding a Power BI Desktop File.....	477
Interacting with a Report on PowerBI.com.....	478
Printing PowerBI.com Reports.....	479
Creating PowerBI.com Dashboards.....	479
Creating a New Dashboard.....	479
Adding Tiles to PowerBI.com Dashboards.....	480
Editing Dashboard Tiles	482
Modifying Dashboards	484
Sharing Dashboards	488
Creating New Reports in PowerBI.com	489
The Power BI App on Tablet Devices	491
PowerBI.com Gateways.....	494
Downloading a Gateway	495
Configuring a Gateway	498
Ad Hoc Data Refresh.....	498
Scheduled Data Refresh	499
Conclusion.....	500
■ Appendix A: Sample Data	501
Sample Data.....	501
Downloading the Sample Data	501
Images	501
Sample Databases.....	501
Index.....	503

About the Author



Adam Aspin is an independent business intelligence consultant based in the United Kingdom. He has worked with SQL Server for nearly 20 years. During this time, he has developed several dozen reporting and analytical systems based on the Microsoft BI product suite.

A graduate of Oxford University, Adam began his career in publishing before moving into IT. Databases soon became a passion, and his experience in this arena ranges from dBase to Oracle, and Access to MySQL, with occasional sorties into the world of DB2. He is, however, most at home in the Microsoft universe when using SQL Server, SQL Server Analysis Services, SQL Server Reporting Services, SQL Server Integration Services, SharePoint, DataZen, and Power BI.

Business Intelligence has been his principal focus for the last 15 years. He has applied his skills for a range of clients in industry sectors across finance, utilities, telecoms, insurance, retail, and luxury goods.

Adam is a frequent contributor to SQLServerCentral.com and Simple-Talk. He is a regular speaker at SQL Server User Groups, SQL Saturdays, and conferences such as SQL Bits. He has written numerous articles for various French IT publications. A fluent French speaker, Adam has worked in France and Switzerland for many years.

He is the author of *SQL Server Data Integration Recipes* (Apress, 2012), *High Impact Data Visualization with Power View, Power Map, and Power BI* (Apress, 2014), and *Business Intelligence with SQL Server Reporting Services* (Apress, 2015).

About the Technical Reviewer



Ian Rice is a business intelligence and data warehouse consultant, specializing in the Microsoft suite of technologies. In his 12 years of consultancy, he has designed and delivered multiple data warehouses, including supporting report environments across finance, insurance, retail, and service sectors. He has also used his experience to assist organizations in recovering failed data warehouse implementations. Ian is currently based in the UK. He accepts clients both nationally and internationally.

Acknowledgments

Writing a technical book can prove to be a daunting challenge. So I am all the more grateful for all the help and encouragement that I have received from so many friends and colleagues.

First, my heartfelt thanks go, once again, to Jonathan Gennick, the commissioning editor of this book. Throughout the publication process, Jonathan has been an exemplary mentor. He has shared his knowledge and experience selflessly and courteously and provided much valuable guidance.

My deepest gratitude goes yet again to Jill Balzano, the Apress coordinating editor for managing this volume through the rocks and shoals of the publication process. She has succeeded in the Sisyphean task of making a potentially stress-filled trek into a pleasant journey filled with light and humor. Her team also deserves thanks for their efforts. So my gratitude also goes to Dhaneesh Kumar for the hours that he has spent formatting and indexing the text, and to Kim Burton-Weisman for editing and polishing my prose.

When delving into the arcane depths of technical products it is all too easy to lose sight of the main objectives of a book. Fortunately, my good friend and former colleague Ian Rice, the technical reviewer, has worked unstintingly to help me retain focus on the objectives of this book. He has also shared his considerable experience of Power BI Desktop in the enterprise and has helped me immensely with his comments and suggestions.

Finally, my deepest gratitude has to be reserved for the two people who have given the most to this book. They are my wife and son, who have always encouraged me to persevere while providing all the support and encouragement that anyone could want. I am very lucky to have both of them.

Introduction

Business intelligence (BI) is a concept that has been around for many years. Until recently, it has too often been a domain reserved for large corporations with teams of dedicated IT specialists. All too frequently, this has meant developing complex solutions using expensive products on timescales that did not meet business needs. All this has changed with the advent of self-service business intelligence.

Now a user with a reasonable knowledge of Microsoft Office can leverage their skills to produce their own analyses with minimal support from central IT. Then they can deliver their insights to colleagues safely and securely via the cloud. This democratization has been made possible by a superb free product from Microsoft-Power BI Desktop that revolutionizes the way in which data is discovered, captured, structured, and shaped so that it can be sliced, diced, chopped, queried, and presented in an interactive and intensely visual way.

Power BI Desktop provides you with the capability to analyze and present your data and to shape and deliver your results easily and impressively. All this can be achieved in a fraction of the time that it would take to specify, develop, and test a corporate solution. To cap it all off, self-service BI with Power BI Desktop lets you produce reports at a fraction of the cost of more traditional solutions, with far less rigidity and overhead.

The aim of this short book is to introduce the reader to this brave new world of self-service business intelligence. This will involve a complete tour of Power BI Desktop. Although a basic knowledge of the MS Office suite will help, this book presumes that you have little or no knowledge of Power BI Desktop. This product is therefore explained from the ground up with the aim of providing the most complete coverage possible of the way in which its components work together to deliver user-driven business intelligence. Hopefully if you read the book and follow the examples given, you will arrive at a level of practical knowledge and confidence that you can subsequently apply to your own BI requirements. This book should prove invaluable to business intelligence developers, MS Office power users, IT managers, and finance experts—indeed anyone who wants to deliver efficient and practical business intelligence to their colleagues. Whether your aim is to develop a proof of concept or to deliver a fully-fledged BI system, this book can, hopefully, be your guide and mentor.

If you wish, you can read this book from start to finish, as it is designed to be a progressive self-tutorial. However, as Power BI Desktop is composed of a set of interdependent BI functions the book is broken down into several chapters that correspond to the various facets of the product:

- Chapter 1: This chapter introduces Power BI Desktop to newcomers to the product.
- Chapters 2 through 5: These chapters cover data integration using the Power BI Desktop Query Editor.
- Chapter 6: This chapter describes data modeling using Power BI Desktop.
- Chapters 7 through 9: These chapters are an introduction to DAX, the Power BI Desktop function language that you use to create metrics for your BI delivery.

■ INTRODUCTION

- Chapters 10 through 15: These chapters explain how to create interactive dashboards and reports.
- Chapter 16: This chapter takes you through the ways that you can share your insights using the cloud-based Power BI service.

This book comes with a small sample data set that you can use to follow the examples that are provided. It may seem paradoxical to use a tiny data sample when explaining a product suite that is capable of analyzing medium and large data sets. However, I prefer to use an extremely simplistic data structure so that the reader is free to focus on the essence of what is being explained, and not the data itself. Inevitably, not every question can be answered and not every issue can be resolved in one book. I truly hope that I have answered many of the essential self-service BI questions that you will face and have provided ways of solving a reasonable number of the challenges that you may encounter. I wish you good luck in using Power BI Desktop to prepare and deliver your insights. And I sincerely hope that you have as much fun with it as I had writing this book.

CHAPTER 1



Introduction to Power BI Desktop

If you are reading this book, it is probably because you need to use data. More specifically, you want to take a journey from data through to insight where quantities of facts and figures need to be shaped into comprehensible information and given clear and visual meaning.

This book is all about that journey. It covers the many ways that you can transform raw data into high-impact analyses using Power BI Desktop—the new self-service business intelligence (BI) application that Microsoft is making freely available. This fresh approach to self-service BI presumes minimal central IT intervention or dependency. It is based on giving the user the ability to use simple yet powerful tools to handle industrial-strength quantities of data and to share stunning output in the shortest possible timescales.

The following are keywords in this universe:

- Speed
- Delivery
- Empowerment
- Decentralization
- Disintermediation

Once you have mastered the tools and techniques described in this book, you will be able to

- Discover, structure, and load your data from a wide range of sources
- Add all the calculations you need to enhance information and extract accurate analysis
- Create stylish interactive presentations
- Share your insights with your colleagues and clients

It follows that this book is written from the perspective of the user. Essentially, it is all about giving users the tools and knowledge to define their own requirements and satisfy their own needs simply and efficiently through developing new and existing skills.

This chapter assumes that you have no prior knowledge of Power BI Desktop or even business intelligence. Consequently, it starts from the very beginning by explaining what exactly self-service business intelligence really is. Then it takes you through the necessary steps to download and install Power BI Desktop. Finally, it takes you on a whirlwind tour of Power BI Desktop, where you see just how quickly and easily you can go from raw data to polished insight with this new and amazing tool.

Electronic supplementary material The online version of this chapter ([doi:10.1007/978-1-4842-1805-1_1](https://doi.org/10.1007/978-1-4842-1805-1_1)) contains supplementary material, which is available to authorized users.

If you intend to follow the dashboard example that you find in this chapter (and I hope that you will), then you need to download the source material for this book from the Apress web site. Appendix A describes how to do this so that you can prepare the terrain for your upcoming adventure with Power BI Desktop.

The Microsoft Self-Service Business Intelligence Solution

It is important to understand from the start that Microsoft's Self-Service Business Intelligence solution is a constantly evolving process. It has been assembled from a series of parallel technologies and it is in a continuous state of flux. Fortunately, this perpetual motion is now at a peak of readiness, and while it is still undergoing some enhancements and revisions, it is ready for immediate use.

So what exactly is Power BI? At its heart, it is a cloud-based service that lets you store and share essential business data in the form of dashboards and reports. However, before you can share dashboards, you need to create them—and this is where Power BI Desktop comes in. This easy-to-use tool is completely free. It is used to find, cleanse, and mash up data so that you can then develop telling metrics and deliver them in the form of stylish visualizations. Once your tables, charts, and maps are assembled into reports, you can then share them with a selected audience in Azure—the Microsoft Cloud (should you want to, of course). Yet the good news does not stop there. Your public can view your insights on just about any Windows, iOS, or Android device using the free Power BI apps that Microsoft has made available for these platforms.

So all that you have to do is to create dashboards with Power BI Desktop, share the output in Azure, and then view and interact with the results using the Power BI apps. It really is that simple. Moreover, up to a certain limit on file sizes, it is completely free.

There is more—much more—in the Power BI universe, but this short description will suffice to get you started. In any case, Chapter 16 provides more detail on the way that Power BI Desktop fits into the Microsoft Self-Service Business Intelligence solution. In the meantime, let's move the focus back to Power BI Desktop. To begin, what exactly will you be using this application for? There are three answers:

- Import data
- Model data
- Create dashboards

Let's take a quick look at some of the things that these may entail.

Import Data from Diverse Sources

The first step on the path to delivering concrete business intelligence is to find and import all the data that you need for your analysis. Power BI Desktop lets you

- Import data from a wide variety of sources. This covers corporate databases to desktop files, social media to big data.
- Merge data from multiple sources and shape it into a coherent structure.
- Cleanse your data to make it reliable and easy to use.
- Break down the data into the columns and records that suit your requirements.

There was a time when these tasks required dedicated teams of IT specialists. In fact, it was considered so complex that it earned its own acronym, ETL, short for Extract, Transform, and Load. Well, this process no longer needs specialists. With Power BI Desktop, you can mash up your own data so that it is ready to use as part of your self-service BI solution.

Handling data is discussed in Chapters 2 through 5.

Model Your Data

Power BI Desktop is not just a data store for your information. It also lets you extend and develop the cleansed data. More specifically, it allows you to

- Create a data model by joining tables to develop a coherent data structure from multiple separate sources of data. This data model is then used in dashboards.
- Enrich the data model by applying coherent names and data types.
- Create calculations and prepare the core metrics that you want to use in your analyses and presentations.

It is worth noting that you can load data into Power BI Desktop directly without going through the data cleansing and modeling stage. If the source data is already in good shape, then you can start using it straightaway. Modeling data and adding calculations is discussed in Chapters [6](#) through [9](#).

Creating Dashboards and Reports

I think of creating dashboards and reports as the “jewel in the crown” of self-service business intelligence. A truly dynamic analysis and presentation approach lets you deliver business intelligence composed of

- tables
- matrixes
- charts
- maps
- gauges
- images

and many other types of visualization.

Not only that, but it is incredibly fast and highly intuitive. It provides advanced interactivity so that you and your users can “slice and dice” the data “on the fly” in real time using

- slicers
- filters

Creating dashboards and reports is discussed in Chapters [10](#) through [15](#).

Power BI Desktop Files

Power BI Desktop lets you create multiple pages in a single file. Each collection of pages that is based on the same underlying data is called a *report*. A Power BI Desktop file therefore contains all the dashboards and all the data that is needed by each element (called a *visualization*) on each page. So, a Power BI Desktop file is completely self-contained.

Power BI Desktop is built to handle vast quantities of data. Fortunately, however, it compresses the data that you load in an extremely efficient way. This means that Power BI Desktop files often take up only a fraction of the space that they would if they contained only the raw source data.

This compression also applies to the data that Power BI Desktop uses when you are modeling data and creating dashboards. This is because Power BI Desktop loads all the data that you are using into memory, where it is compressed to make the most of the available memory. This means that Power BI Desktop is extremely fast to use and normally shows you the results of any changes that you make or any filters that you apply in fractions of a second. This instantaneous interactivity also applies to dashboards that you display in Windows, iOS, or Android apps.

Installing Power BI Desktop

The first thing that you have to do to create dashboards (or reports or pages of visualisations) is download and install Power BI Desktop. Although this process is really easy, you will save time if you ensure that the computer where you want to install Power BI Desktop has the capability to run this application. Currently, the minimum requirements are as follows:

- Windows 10, Windows 7, Windows 8, Windows 8.1, Windows Server 2008 R2, Windows Server 2012, or Windows Server 2012 R2
- Internet Explorer 10 or greater

Note Power BI Desktop Designer works on both 32-bit and 64-bit computers. However, if you intend to analyze large datasets, a 64-bit workstation with several gigabytes of memory could very well prove necessary.

Microsoft does not specify a minimum memory requirement, but you need to be aware that although the application itself is not a memory hog, it can let you load huge amounts of data into Power BI Desktop. Given that all of this data will be loaded into memory, you need to ensure that you have enough available memory if you intend to analyze large amounts of data—even if the data is compressed.

So, if you are sure that your PC or laptop is ready for Power BI Desktop, you can install it by following these steps:

1. Go to the Power BI Desktop download page on the Microsoft web site. This is currently <https://www.microsoft.com/en-us/download/details.aspx?id=45331>. You can easily find the right page by entering **Power BI Desktop download** in your favorite search engine. You should see a web page containing something like the information shown in Figure 1-1.

Microsoft Power BI Desktop

Microsoft Power BI Desktop is built for the analyst. It combines state-of-the-art interactive visualizations, with industry-leading data query and modeling built-in. Create and publish your reports to Power BI. Power BI Desktop helps you empower others with timely critical insights, anytime, anywhere.

Details

System Requirements

Install Instructions

Figure 1-1. The Power BI Desktop download page

2. Select the download version (32-bit or 64-bit).
3. Click the Download button. You will be taken to the next page, where you should choose the type of download (32-bit or 64-bit), as shown in Figure 1-2.

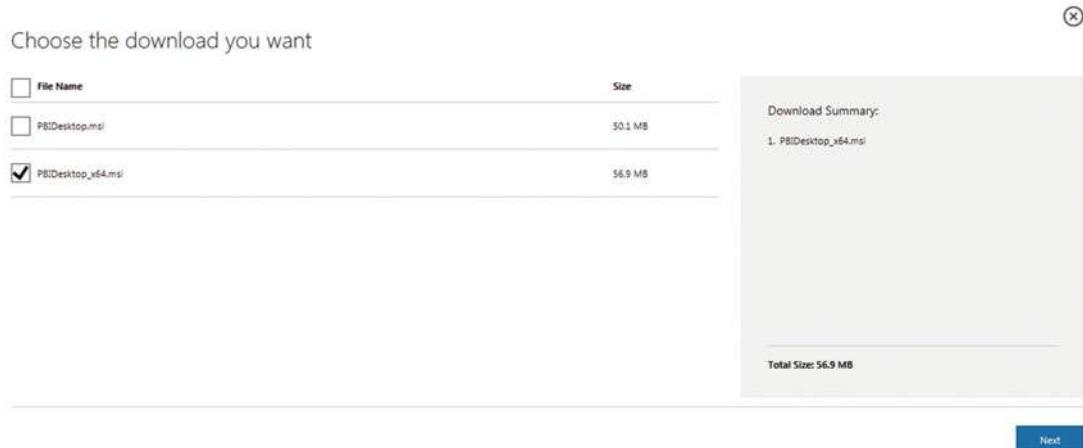


Figure 1-2. The download selection page

4. Click Next (this button only appears once you have selected the type of download that you want). The final download page is displayed. A pop-up appears, as shown in Figure 1-3.



Figure 1-3. The save or run download pop-up

5. Click Run. The Power BI Desktop installation package will download (probably in under a minute) and the initial is displayed, as shown in Figure 1-4. If you do not see this dialog once the download has completed, then click the toolbar icon (this will have appeared in the toolbar to make it show on top of any other open windows).



Figure 1-4. The initial Power BI Desktop setup dialog

6. Click Next. The Setup Licensing dialog will appear, as shown in Figure 1-5.



Figure 1-5. The setup license dialog

7. Ensure that the check box accepting the license agreement is checked and click Next. The setup destination dialog will appear, as shown in Figure 1-6. If you prefer to install the Power BI Desktop files in a different directory, then you can enter it here (or click the Change button and browse to select it).

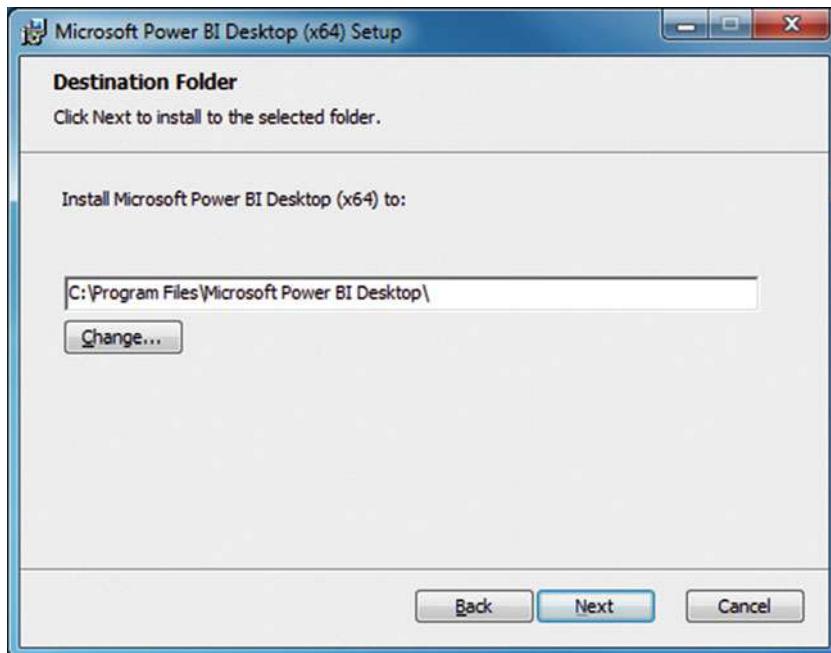


Figure 1-6. The setup destination dialog

8. Click Next. The final confirmation dialog will appear, as shown in Figure 1-7.

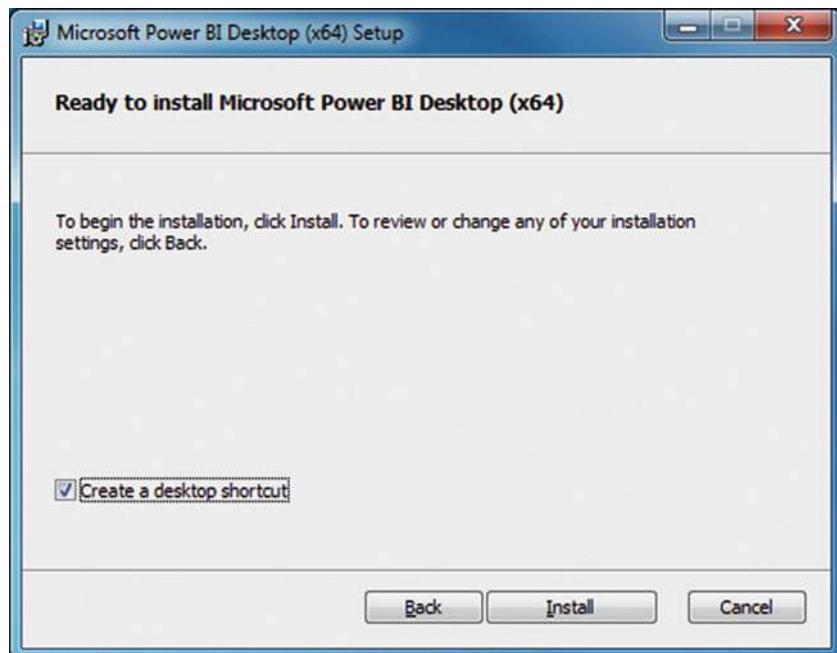


Figure 1-7. The setup final confirmation dialog

9. Click **Install**. The Power BI Desktop installation package will run and will complete the installation in a few seconds. You will see a progress dialog, as shown in Figure 1-8.

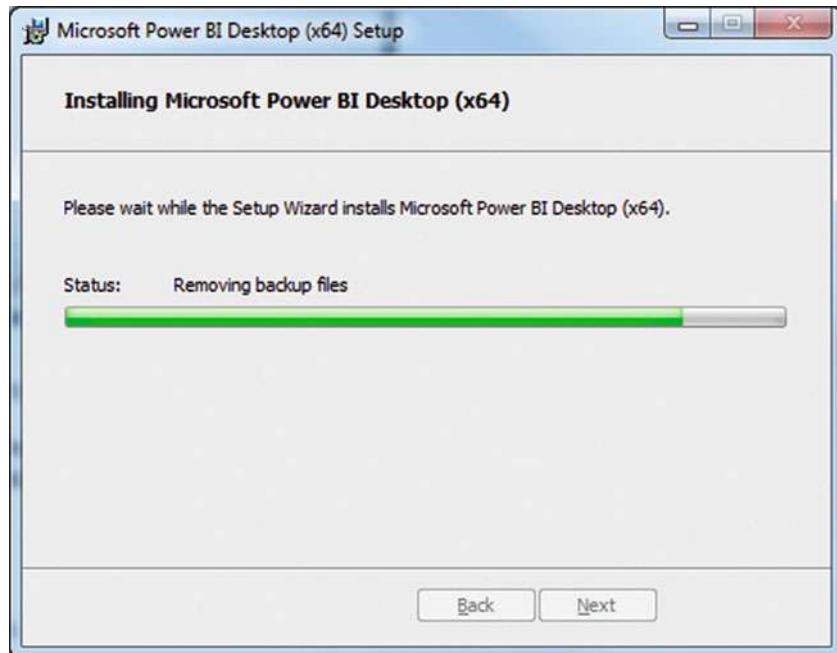


Figure 1-8. The installation progress dialog

10. Once the installation process has finished successfully, you will see the completion dialog, as shown in Figure 1-9.



Figure 1-9. The final Power BI Desktop installation dialog

11. If you want to run Power BI Desktop immediately, then leave the Launch Power BI Desktop check box ticked; otherwise, uncheck it and click Finish. The dialog will close and Power BI Desktop is now installed on your computer.

Removing Power BI Desktop

Should you ever want to remove Power BI Desktop from a computer where it has been installed, you have a couple of choices:

- Run the web-based installation process as described earlier. At step 4, you see a dialog asking you if you want to repair or remove Power BI Desktop from your computer. Click Remove and follow the process that is indicated to delete the application from your machine.
- Open the Windows Control Panel. In the Programs section, click Uninstall a program. Select Power BI Desktop from the list of currently installed programs to uninstall it.

Running Power BI Desktop

Once you have installed Power BI Desktop successfully, you are ready to start creating dashboards and analyzing your data. You can start your Power BI Desktop experience the program as follows:

Click the Power BI Desktop icon that was created on the Desktop as part of the installation process. You will see the Power BI Desktop splash screen, as shown in Figure 1-10.

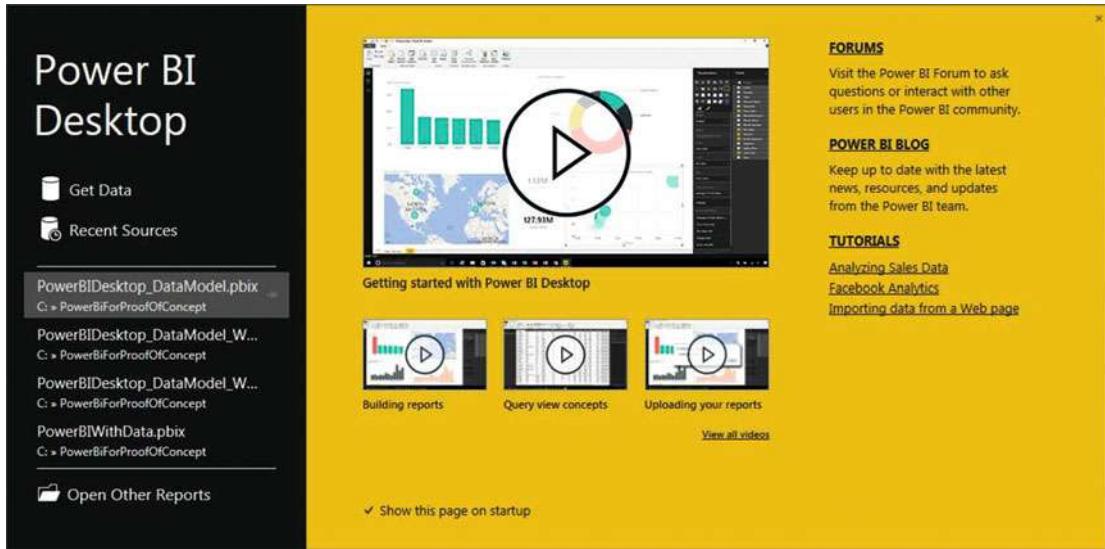


Figure 1-10. The Power BI Desktop splash screen

For the moment, however, it is time to stop and draw breath. You have successfully installed Power BI Desktop and you are ready to create your first dashboard with this exciting and revolutionary application.

A First Power BI Desktop Dashboard

This book takes you through an immense amount of detail that explains how to import, cleanse, and shape data from a multitude of different sources. You then learn how to carry out a variety of calculations that will help you to tease out the meaning from the data that you are analyzing. Finally, you see how to transform this analysis into telling visuals that make your insights intuitively comprehensible to your audience.

Yet before delving into all of this detail, it is perhaps more important to appreciate the really fundamental qualities of this amazing application. Despite the depth and reach of this piece of software, there are other qualities that make it stand out and that are possibly even more fundamental, including

- *Simplicity:* Anyone can learn to create stunning visualizations and carry out in-depth analysis of data without having to endure a steep or arduous learning curve.
- *Power:* Data from virtually any source can be loaded, manipulated, and combined with other data elements, and then modeled and extended without needing advanced knowledge of IT systems or data management.

Consequently, it is important to see just how easy it is to use the Power BI Desktop dashboard. Indeed, the fastest way to get you “hooked” on this particular tool is to let you see for yourself how fast you can go from zero to hero in delivering compelling dashboards. So let’s see just how quick and easy it can be to take a data source (an Excel file in this instance) and transform it into a Power BI Desktop dashboard.

Loading the Source Data

Once you have launched Power BI Desktop, you are faced with the startup screen that you saw in Figure 1-10. Given that you are working with an application that lives and breathes data, it is not really surprising that the first step in a new analytical challenge is to find and load some data. So the following explains what you have to do (assuming that you have downloaded the sample data that accompanies this book from the Apress web site).

1. Click Get Data in the startup screen. The Get Data dialog will appear, as shown in Figure 1-11.

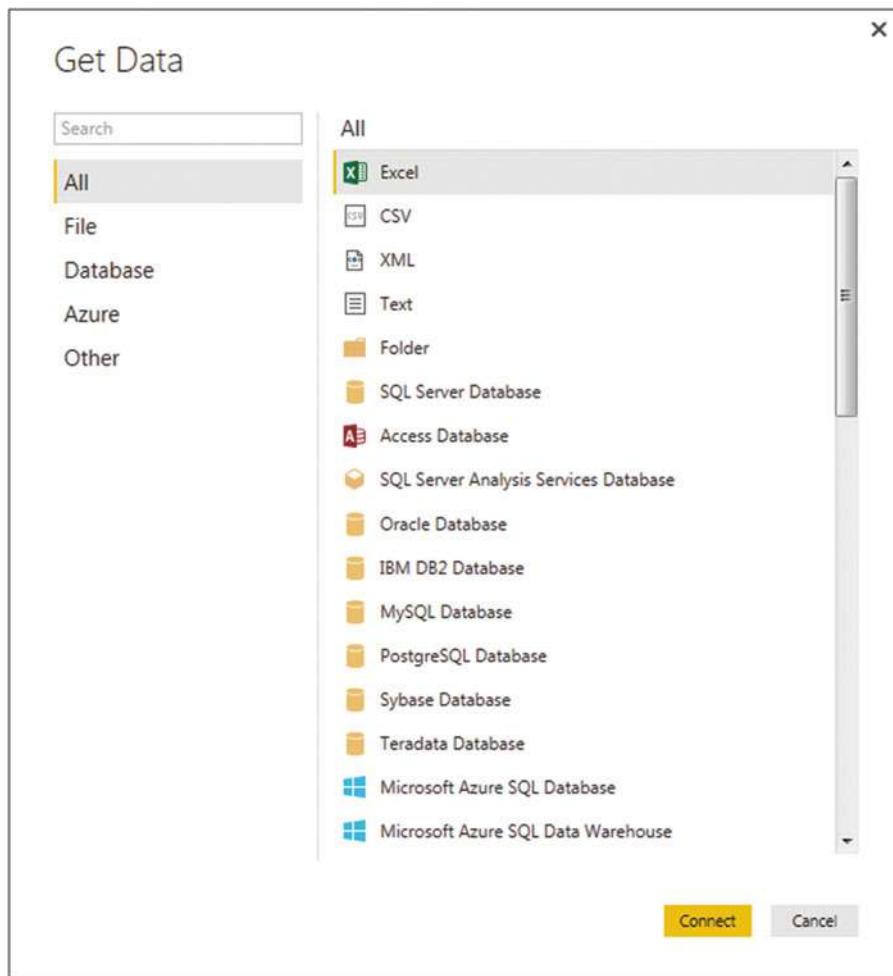


Figure 1-11. The Get Data dialog

2. In the list of all the possible data sources on the right of this dialog, click Excel, and then click Connect. The Windows Open File dialog will appear, as shown in Figure 1-12.

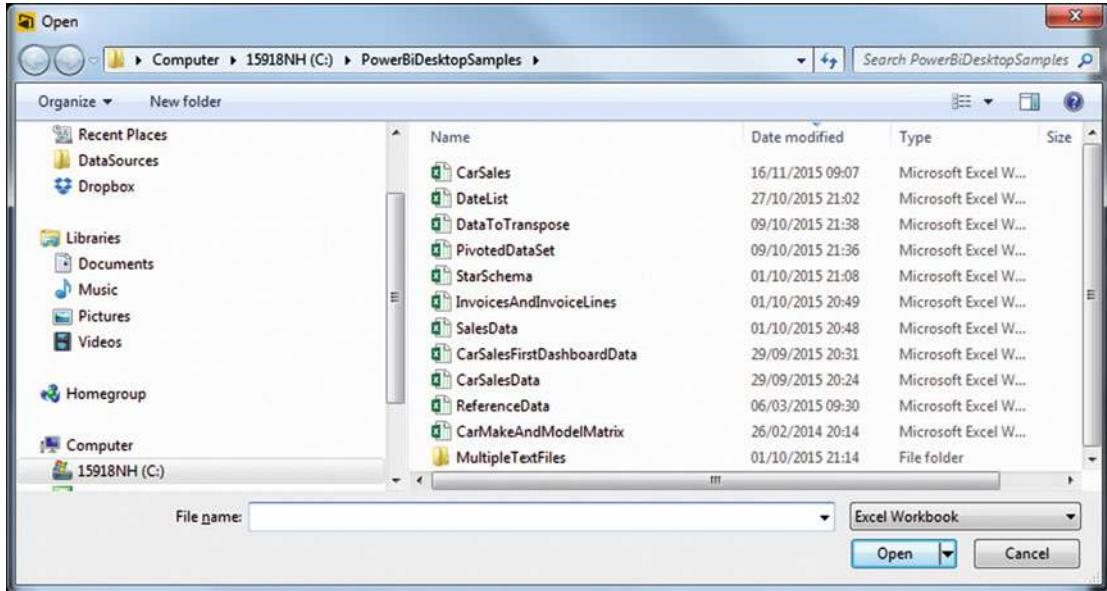


Figure 1-12. The Windows Open File dialog when loading data from a file source

3. Click the file C:\PowerBiDesktopSamples\CarSales.xlsx.
4. Click the Open button. The Connecting dialog will appear for a second or two and then the Navigator dialog will appear.
5. You will see that the CarSales.xlsx file appears on the left of the Navigator dialog and that any workbooks, named ranges, or data tables that it contains are also listed. Click the BaseData worksheet name that is on the left. The contents of this workbook will appear in the data pane on the right of the Navigator dialog.
6. Click the check box for the BaseData worksheet on the left. The Load and Edit buttons will be activated. The Navigator dialog should look like Figure 1-13.

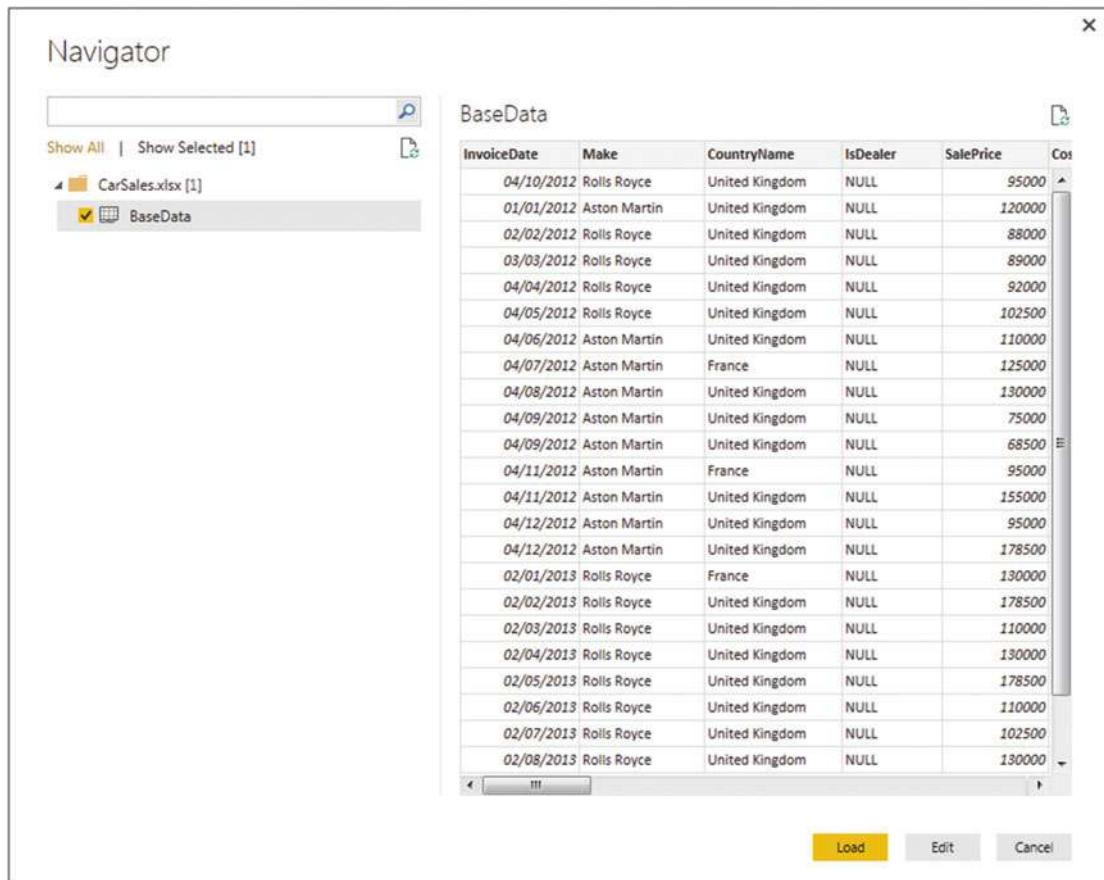


Figure 1-13. The Navigator dialog with data selected

- Click Load. The data will be loaded from the Excel file into Power BI Desktop. You will see the Power BI Desktop report window, like the one shown in Figure 1-14.

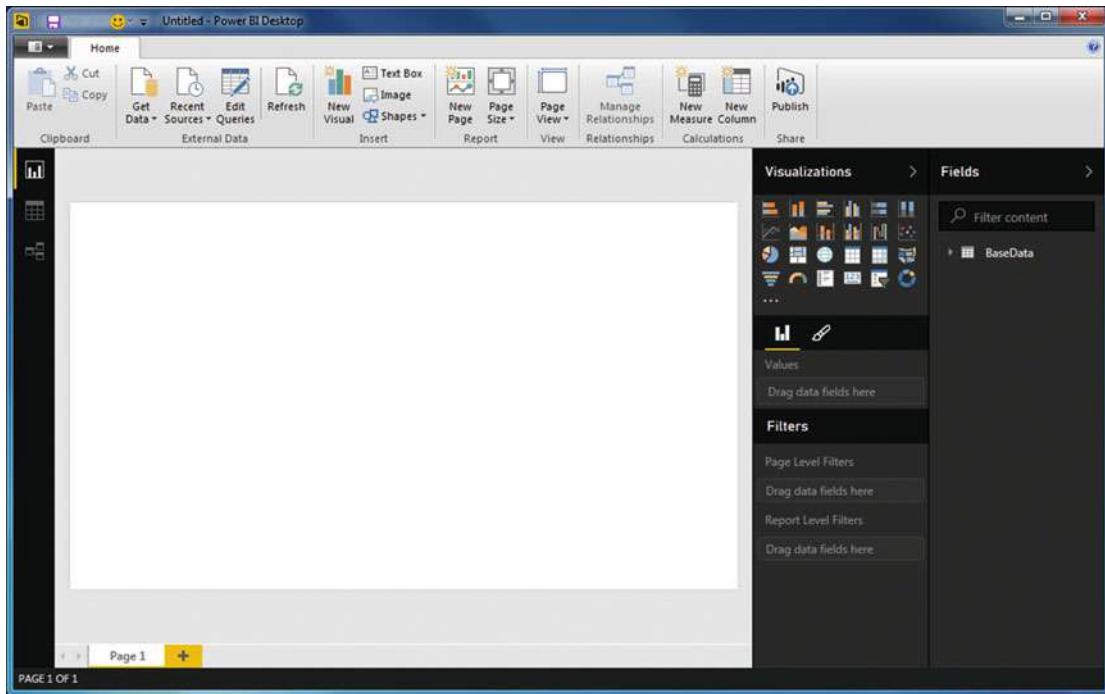


Figure 1-14. The Power BI Desktop report window

I imagine that loading this data took under a minute. Yet you now have a fully operational data model in Power BI Desktop that is ready for action. However, before moving forward and creating a dashboard, I would like to pause for an instant and explain exactly what you have seen so far. Of course, if you are itching to race ahead and actually create a couple of tables and charts, then you can always jump ahead to the “Your First Visualizations” section.

The Data Load Process

What you have seen so far is an extremely rapid dash through a Power BI Desktop data load scenario. In reality, this process can range from the blindingly simple (as you just saw) to the more complex (where you join, filter, and modify multiple datasets from different sources). However, it will always be the first step in any data analysis scenario when you are using Power BI Desktop.

In this short example, you nonetheless saw many of the key elements of the data load process. These included

- Accessing data that is available in any of the source formats that Power BI Desktop can read.
- Taking a first look at the data before loading it into Power BI Desktop.

What you did not see here is how Power BI Desktop can add an intermediate step to the data load process and edit the source data in Power BI Desktop Query Editor. This aspect of data manipulation is covered extensively in the following few chapters.

The Navigator Window

One key aspect of the data load process is using the Navigator window correctly. The Navigator window appears when connecting to many, but not all, data sources. It is there to let you

- Take a quick look at the available data tables in the data source.
- Filter multiple data elements that are available in a single data source.
- Look at the data in individual tables.
- Select one or more data tables to load into Power BI Desktop.

Depending on the data source to which you have connected, you might see only a few data tables in the Navigator window, or hundreds of them. In any case, what you can see are the structured datasets that Power BI Desktop can recognize and is confident that it can import. Equally dependent on the data source is the level of complexity of what you will see in the Navigator window. If you are looking at a database server, for instance, then you may start out with a list of databases and you may need to dig deeper into the arborescence of the data by expanding databases to list the available data tables and views.

You will see much, much more of the Navigator window in the following chapter.

The Navigator Data Preview

The Navigator Data Preview pane is, as its name implies, a preview of the data in a data source. It provides

- A brief overview of the top few records in any of the datasets that you want to look at. Given that the data you are previewing could be hundreds of columns wide and millions of rows long, there could be scroll bars for the data table visible inside the Navigator Data Preview.
- A list of the available columns in the data table. These are shown at the bottom of the Navigator Data Preview.

Power BI Desktop can preview and load data from several different sources. Indeed (as you can see from the list of possible data sources in the Get Data dialog in Figure 1-11), it can read most of the commonly available enterprise data sources as well as many, many others. What is important to appreciate is that Power BI Desktop applies a common interface to the art and science of loading data, whatever the source. So whether you are examining an SQL Server or an Oracle database, an XML file or a text file, a web page or a big data source, you will always be using a standardized approach to examining and loading the data. This makes the Power BI Desktop data experience infinitely simpler—and extremely reassuring. It means that you spend less time worrying about technical aspects of data sources and you are free to focus on the data itself.

Note The Navigator Data Preview is a brilliant data discovery tool. Without having to load any data, you can take a quick look at the data source and any data that it contains that can probably be loaded by Power BI Desktop. You can then decide if it is worth loading, so that you do not waste time on a data load that could be superfluous to your needs.

Modifying Data

Once you have one or more queries in Power BI Desktop that can connect to data sources and bring the data into this environment, you can start thinking about the next step—transforming the data so that it is ready for use. Depending on the number of data sources that you are handling and the extent of any modifications that are required, this could vary from the simple to the complex. To give a process some structure, I advise that you try to break down any steps into the following main threads:

- *Shape the dataset:* This covers filtering out records to reduce the size of the dataset, as well as removing any extraneous columns. It may also involve adding columns that you create by splitting existing columns, creating calculated columns, or even joining queries.
- *Cleanse and modify the data:* This is also known as *data transformation* (the T in ETL). It encompasses the process of converting text data to uppercase and lowercase, as well as (for instance) removing nonprinting characters. Rounding numbers and extracting date parts from date data are also possible (among many other eventual transformations).

For the moment, however, it is only important to understand that Power BI Desktop can do all of this if you need it to. Transforming data is explained in detail in Chapters 2 through 5.

The Power BI Desktop Window

Before we go any further, I would like to explain the Power BI Desktop window, since it is something that you will use a lot in this chapter from this point onward. The Power BI Desktop window contains the elements that are outlined in Figure 1-15.

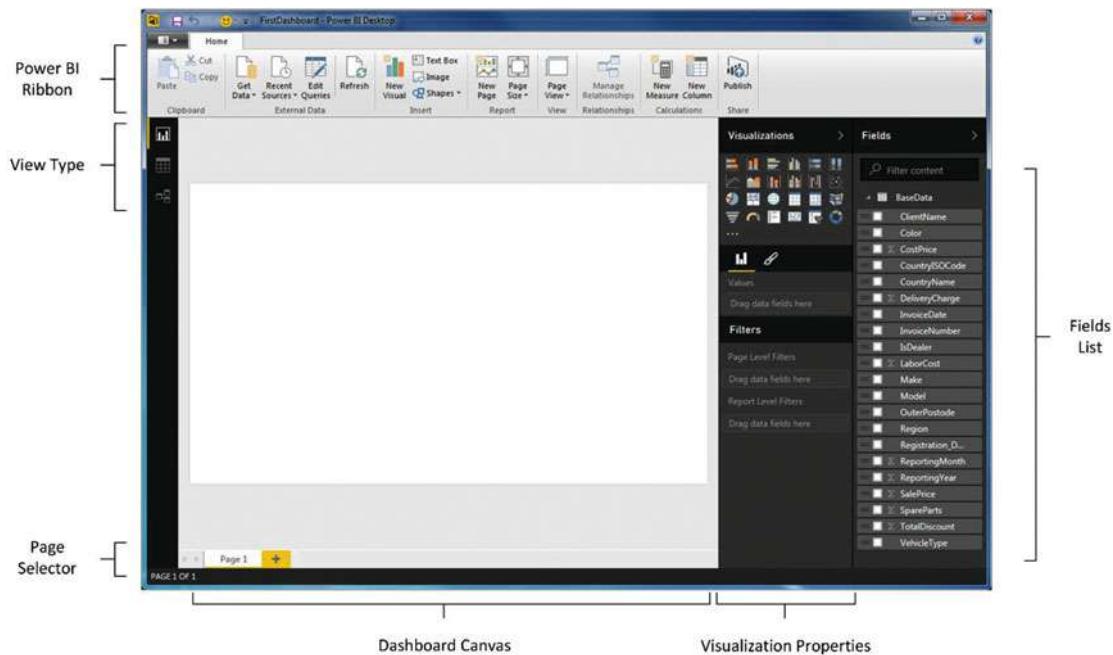


Figure 1-15. The Power BI Desktop

As you can see, the Power BI Desktop screen is simple and uncluttered. The various elements that it contains are explained in Table 1-1.

Table 1-1. Power BI Desktop Options

Option	Description
Power BI Ribbon	This contains the principal options that are available to you when developing dashboards with Power BI Desktop.
View Type	These three icons let you flip between Dashboard view (where you create dashboards and reports), Data view (where you add calculations), and Relationships view (where you join data from different sources).
Dashboard Canvas	This is the main area, where you add visualizations and design your dashboards.
Visualization Properties	This area of the application is specific to each type of visualization and lets you set the specific attributes of each element on a dashboard. It also allows you to filter dashboards, pages, and individual visualizations.
Fields List	Here you can see all the available fields from the source data that you can use to build your visualizations.
Visualization Palette	This area contains all the currently available types of visualization that you can add to a dashboard.
Page Selector	These are tabs that let you switch from page to page in a report.

Power BI Desktop—like most Microsoft applications—has several available ribbons. These are explained in the course of this book.

Your First Visualizations

With your data safely in place inside Power BI Desktop, you can now begin to create the tables, charts, maps, and other elements that you want to add to a dashboard, which you can use to present your first insights into Brilliant British Cars. As this is a first “taster” exercise, I am not looking at explaining all that can be done using Power BI Desktop. All I want to do is to show you how easy it is to create dashboards in minutes rather than hours. Indeed, I only hope that this first simple dashboard will leave you hungry to learn more—and so to move on to the rest of this book.

Before creating a few simple visualizations, let me clarify some of the terms that you will meet when working with Power BI Desktop.

- *Visualization:* Also known as *visuals*, these are the individual presentation elements that you create based on the underlying data. A visual can be a table, a chart, a gauge, a map, or many things indeed.
- *Dashboard:* A Power BI Desktop dashboard is a collection of visualizations on a single page. Indeed, I use the terms *page* and *dashboard* interchangeably.
- *Report:* This is a collection of pages (or dashboards) in a single file, all using the same dataset.

Display Available Fields

One of the first things to do is make sure that you can see the data that you will be working with in dashboards and reports. If you look at the right of the Power BI Desktop Report view, you see a vertical pane with the label *Fields* at the top. This is the Fields pane. It is from here that you access all the data that you will use in your visualizations and dashboards.

For the moment, however, all that you can see is probably the name of the data table that you imported previously—the BaseData table. Do the following to see all the fields that this table contains.

1. Click the small triangle to the left of the table name. The table will expand to reveal all the available fields that it contains. You can see what this looks like in Figure 1-16.

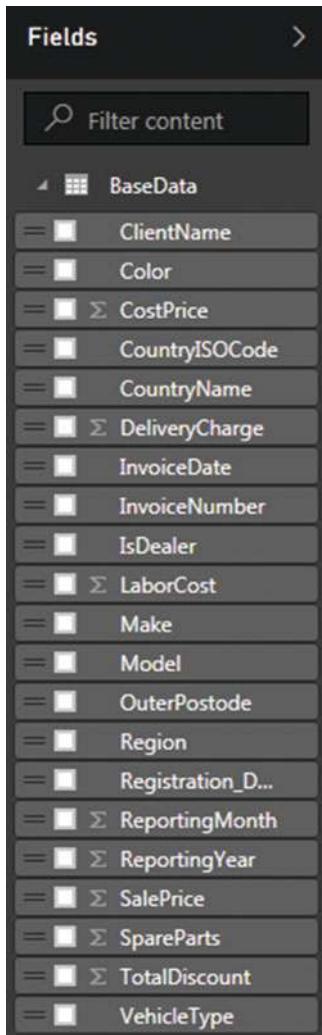


Figure 1-16. The Power BI Desktop Fields list

You can see that some of the fields have a sigma (Σ) icon to their left. This indicates that the data in the field is numeric. As you progress through this book, you will see that there are other icons that Power BI Desktop uses to flag different types of fields.

Add a Matrix of Sales per Country by Year

It is now time to draw on the blank canvas that is your first dashboard. To begin, with let's start with a simple matrix of sales per country for each year that Brilliant British Cars has been trading.

1. In the Visualizations pane, click the matrix icon, as highlighted in Figure 1-17.
A blank matrix will appear on the dashboard canvas.



Figure 1-17. The matrix icon in the Visualizations pane

2. Leaving the freshly created matrix selected, click the check box to the left of the CountryName field in the Fields list. The list of countries where cars have been sold will appear as the left-hand column of the matrix.
3. Drag the ReportingYear field into the Visualizations pane over the Columns fields area (this is called the *field well*). Figure 1-18 shows how to do this. This adds the model years as column headers in the matrix.

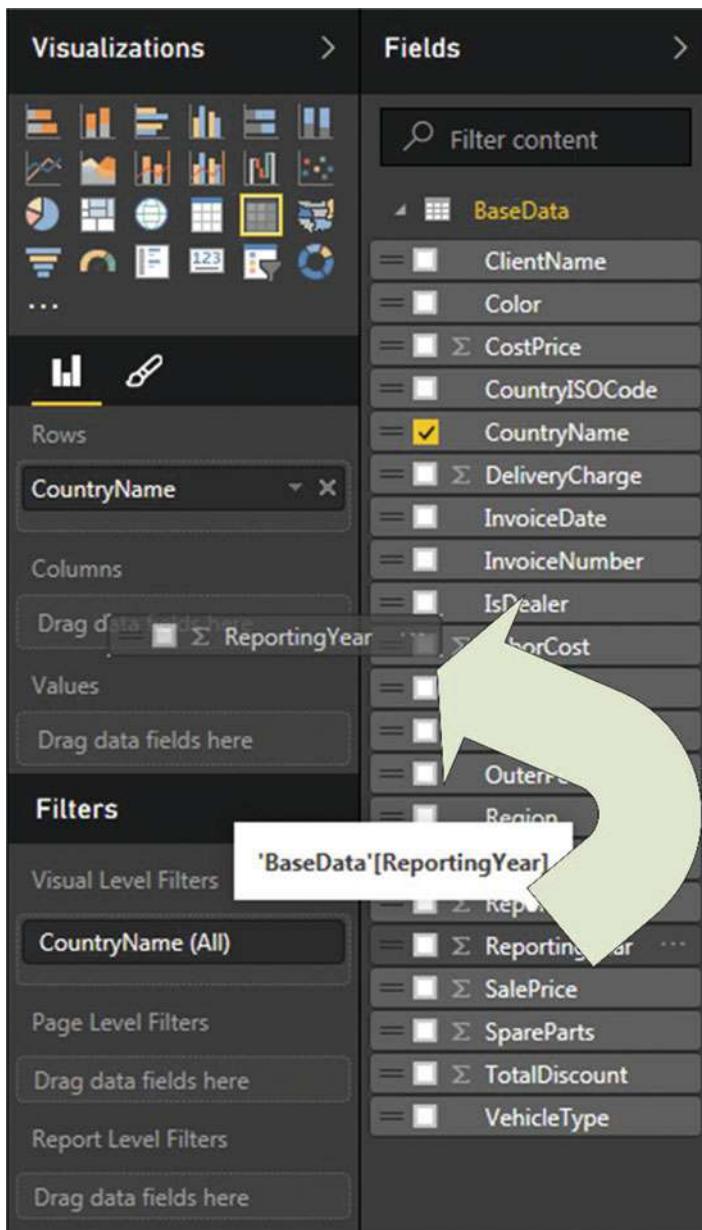


Figure 1-18. Adding the Columns fields to a matrix

4. Leaving the matrix selected, click the check box to the left of the SalePrice field in the Fields list. The aggregated sale price for all vehicles sold by country and by year will appear in the matrix.
5. Drag the corner handle of the matrix to resize it so that there is no spare white space inside the matrix itself. It will look like Figure 1-19.

CountryName	2012	2013	2014	2015	Total
France	248000	446950	193200	571950	1460100
Germany		75000		85000	160000
Spain		92000		86700	178700
Switzerland	88200	233625	103200	276125	701150
United Kingdom	1702890	3169000	1738190	3583800	10193880
USA	113200	162000	2548490	4892375	7716065
Total	2152290	4178575	4583080	9495950	20409895

Figure 1-19. A matrix of sales per country

It would be hard to make this any simpler. Within seconds, you have created a matrix of sales by year and country and the totals have been added automatically. Of course, there are many ways of extending and developing a matrix in Power BI Desktop—and you can discover them all in Chapter 10—but for now, it is time to press on and add a chart to your fledgling dashboard.

Note In this short exercise, you saw that you can both select fields from the Fields list or drag them to the field well to add them to a selected visual. An alternative is to drag a field from the Fields list onto the visualization itself.

Add a Column Chart of Delivery Charge by Model

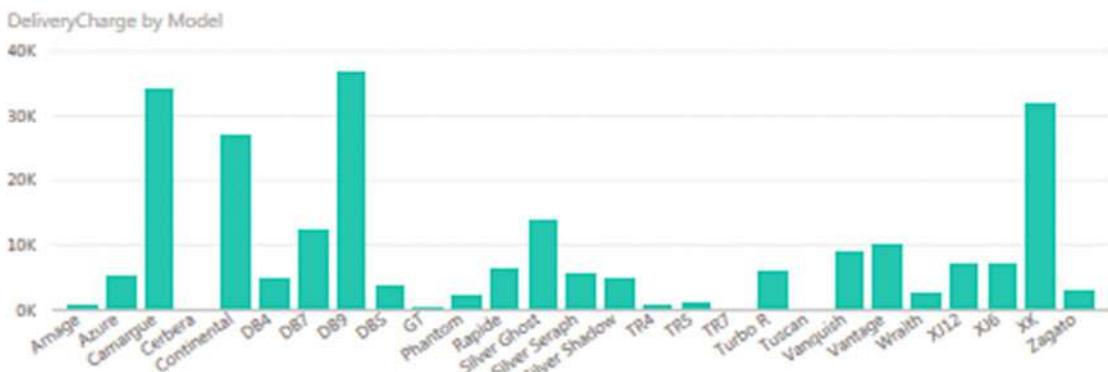
Now that you have seen how easy it is to create a matrix in Power BI Desktop, the time has come to add some visual impact to your analysis. This time, you will use the available data to display the total delivery charge for each model of car sold.

1. Click an empty area of the dashboard canvas to unselect any visualizations.
2. Drag the Model field onto an empty area of the dashboard canvas. Power BI Desktop automatically creates a table displaying all the vehicle models sold.
3. Drag the DeliveryCharge field from the Fields pane onto the table that you just created. Power BI Desktop will calculate the total DeliveryCharge for each available make. The table will look like Figure 1-20.

Model	DeliveryCharge
Arnage	975
Azure	5175
Camargue	34140
Cerbera	150
Continental	27175
DB4	4850
DB7	12550
DB9	36950
DBS	3950
GT	550
Phantom	2225
Rapide	6450
<i>Grand Chassis</i>	112840
Total	239970

Figure 1-20. A table of aggregated delivery charge per make

- Leaving the table selected, click the column chart icon in the Visualizations pane. This is the second icon on the left on the upper row of the selection of visualizations. Power BI Desktop will switch the table to a chart.
- Drag the corner handle of the chart to resize it so that all the makes are visible on the bottom axis. The chart will look like Figure 1-21.

**Figure 1-21.** A column chart of delivery charge by model

Equally simple, don't you think? Yet, believe me, the fun has only just begun. While you will see lots more about how to create and enhance charts in Chapter 11, you can always try a few basic tweaks now. For instance, if you select the chart and then click any of the other charting icons in the Visualizations pane, you can change the type of chart instantaneously. Moreover, if you are not sure which icon does what, then all you have to do is hover the mouse pointer over an icon in the Visualizations pane to display a tooltip that will guide you further.

Add a Map of Labor Cost by Country

Tables and charts are all very well, but nothing beats a good picture when it comes to illustrating a point or highlighting an insight. So, as we have a dataset that includes information for a range of countries, why not display some of our analysis as a map?

1. Click any empty part of the dashboard canvas to unselect any visualizations.
2. Click the filled map icon in the Visualizations pane. You can see this icon in Figure 1-22.

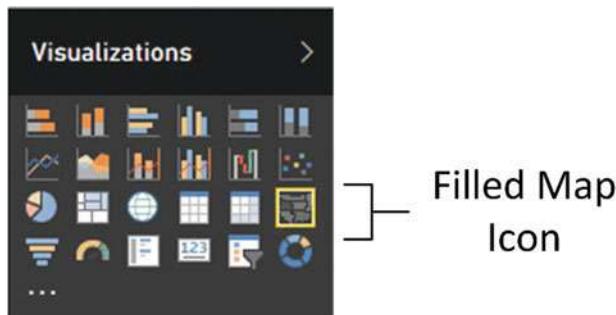


Figure 1-22. Adding a Filled Map visualization to a dashboard

3. Leaving the empty map visualization selected, click the check box to the left of the CountryName field in the Fields list. This will display a map of the world.
4. Leaving the map selected, drag the LaborCost field onto the map. This will highlight any countries where there are labor costs relating to vehicles sold.
5. Drag the colored European countries to the center of the map.
6. Using the mouse wheel, zoom in to the colored European countries. The finished map will look like Figure 1-23.



Figure 1-23. A map of labor cost by country

This time, and in only a few clicks, you have used your data to create a map that clearly illustrates the geography of your sales. Once again, this is only a rapid overview of all that Power BI Desktop can do when it comes to displaying mapping data. You will learn more about creating and modifying maps in Power BI Desktop in Chapter 12.

Add a Card Showing the Total Cost of Spare Parts

Sometimes you do not want to show a large amount of data but quite the opposite. You want to highlight a single figure to give it prominence on the dashboard. Power BI Desktop has a really effective way of doing just this. It consists of adding visualizations called *cards*, which are what you will now add to your dashboard.

1. Click the dashboard canvas to unselect any visualizations.
2. Click the card icon in the Visualizations pane, as shown in Figure 1-24.



Figure 1-24. Adding a card visualization to a dashboard

3. Leaving the (slightly clunky) empty card visualization selected, click the check box to the left of the SpareParts field in the Fields list. This displays the spare parts total in the source data.
4. Drag the corner handle of the matrix to resize it so that there is no spare white space inside the matrix itself. It will look like Figure 1-25.

495K

SpareParts

Figure 1-25. A card showing the total cost of spare parts

That is all you have to do. Three or four clicks and you have a clear visualization of a key metric for your audience. This is not the only way that you can create this particular visualization, but you have to wait for Chapter 10 to get all the details on adding cards to Power BI Desktop dashboards.

Add a Slicer by Make

As a final tweak, I want you to add some interactivity to the dashboard that you are building. What you will do now is to add a slicer (an interactive selection tool) that will let you—or any user of this dashboard—filter by any or all car models sold. Here is how you can do this.

1. Drag the Make field to a blank area on the dashboard canvas. Power BI Desktop will create a list of vehicle models.
2. Click the slicer icon in the Visualizations pane, as shown in Figure 1-26.



Figure 1-26. Adding a Slicer to a dashboard

3. Drag the corner handle of the slicer to resize it so that there is no spare white space inside the slicer. It will look like Figure 1-27.



Figure 1-27. A slicer on the model of vehicle

You can now test the slicer by selecting—or deselecting—any car model that is listed in the slicer. The other visualizations on the dashboard will instantly be updated to reflect the choice of models. You will soon get a first look at how this slicer can be used to filter data.

Arranging the Dashboard

Now that you have created a few visualizations, it is time to coordinate them on the page so that you can deliver a meaningful dashboard that adds power to your insights.

Moving a Visualization

Moving a visualization is impressively easy.

1. Click the visualization that you want to move.
2. Drag the visualization elsewhere on the dashboard canvas.

Resizing a Visualization

Resizing a visualization is also extremely easy.

1. Click the visualization that you want to resize.
2. Move the mouse pointer over any of the corner or side handles of the visualization. The pointer will become a double-headed arrow.
3. Drag the edge of the visualization to increase or decrease its current size.

After a little effort, your dashboard could look like the one in Figure 1-28.

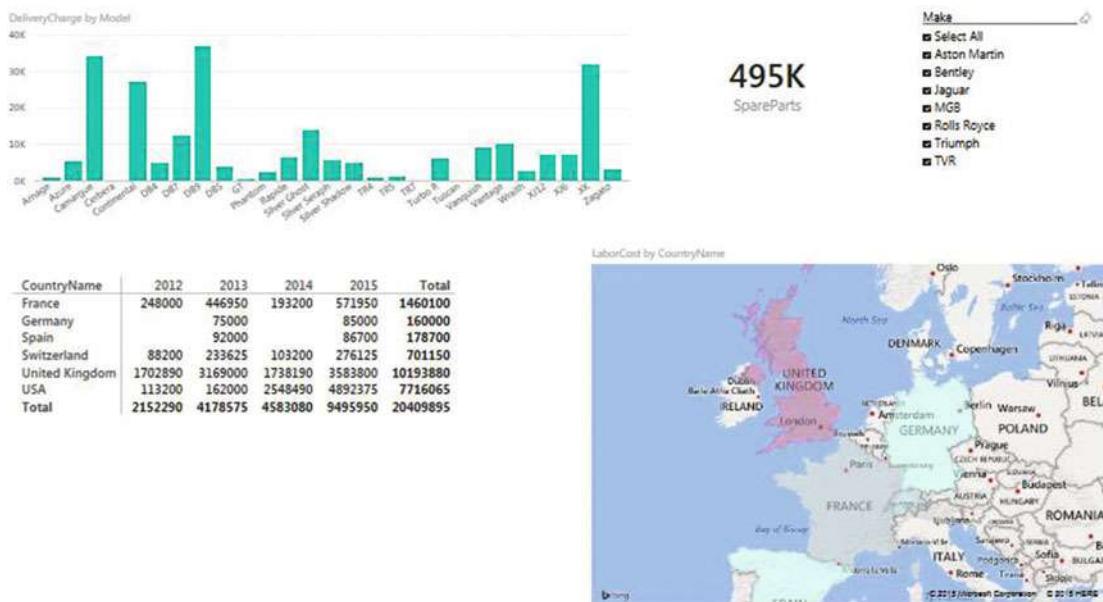


Figure 1-28. The final version of your first dashboard

So here you have your first Power BI dashboard. I have to admit that this first stab at self-service business intelligence was not concentrating overmuch on the aesthetics of the output. I preferred to let you appreciate the speed and simplicity with which you have created an entire dashboard, from scratch and with no prior knowledge of the tool that you have used.

How long did it take you to build this dashboard, do you think? Fifteen minutes? Thirty minutes? Indeed, however long it took, you have also learned the basics of dashboarding with Power BI Desktop. You can now build on this knowledge as you progress through this book.

In fact, extending a dashboard by adding further visualizations is so intuitive that it is too easy to miss the salient points of what you have seen so far. So, to resume, what you have just learned is that

- You can place any visualization anywhere on the dashboard canvas
- You can resize an element quickly and easily
- You can convert any type of visualization to any other type in a single click

Interactivity in Dashboards

Building a dashboard was only the start, as far as Power BI Desktop is concerned. For a Power BI dashboard is never set in stone. In fact, quite the opposite is true, because every dashboard that you create is instantly and intuitively interactive. This means that you can use it to highlight salient points and drill down to expose the key insights that your analysis has led you to.

Even a simple dashboard like the one that you just created is immediately interactive. As an example, suppose that you want to use this dashboard to display data for only a couple of the makes that the company has sold. The following explains how to do it.

1. In the slicer (on the top right of the dashboard), click Bentley and Rolls-Royce. The dashboard will instantly update to show only data for these car models, as shown in Figure 1-29.



Figure 1-29. Interactively filtering a dashboard using a slicer

2. In the slicer, click Select All to clear the filter.
3. In the map, click France. The dashboard will instantly update to show only data for the selected country, as shown in Figure 1-30.

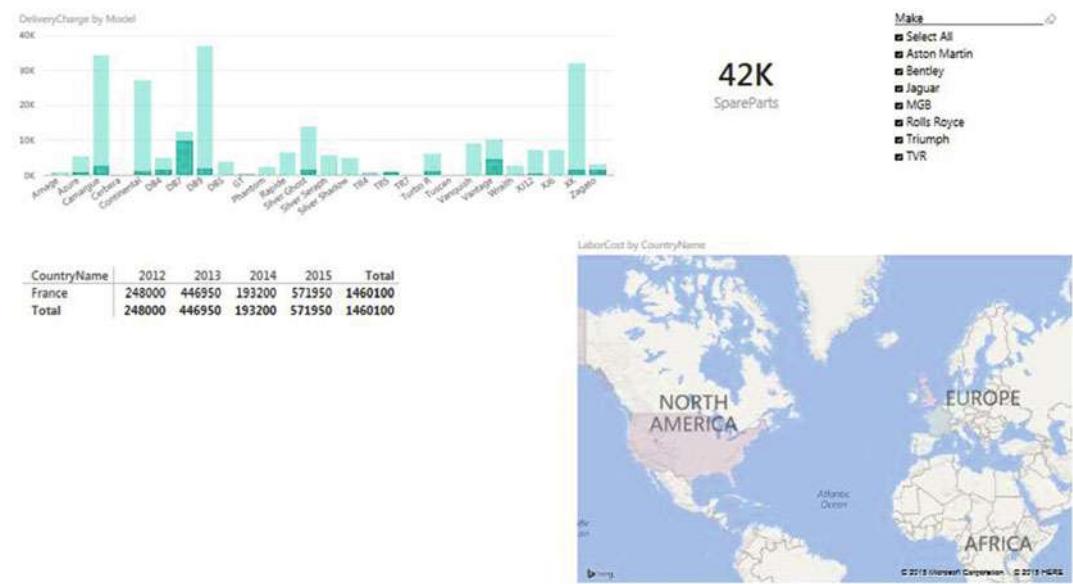


Figure 1-30. Interactively filtering a dashboard using a map

Now it is your turn. Rather than explain here all that you can do to filter and view your data, I suggest that you try clicking parts of the map or the column chart and see what happens! In any case, all the detail concerning filters and slicers is explained in Chapters 13 and 14.

Creating and Modifying Reports

So far in this chapter, we have treated the Power BI Desktop file as if it consisted of only a single page. In practice, you are likely to need to base several pages of analysis and information on the same underlying dataset. Consequently, Power BI Desktop makes it easy to add, copy, and delete the pages in your original file so that you can create complex data stories that all use the same dataset.

If (as I presume is probably the case for many Power BI users) you are used to using Excel, then you will likely find the way that pages are handled in self-service BI incredibly simple, because in Power BI Desktop, each page is very similar to an Excel worksheet. To make matters clearer, look at Figure 1-31, where you can see the page tabs of Power BI Desktop.

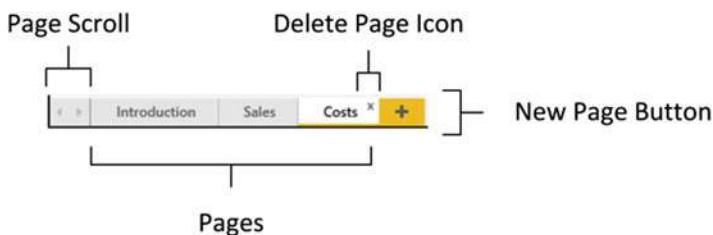


Figure 1-31. The Power BI Desktop page tabs

Adding Pages

When you open a new Power BI Desktop file, it always defaults to having a single page, thoughtfully named Page 1. You can add a new page as follows:

In the Home ribbon, click the New Page button. A new blank page named Page *n* will be added to the existing collection of pages in the report.

Tip As an alternative to the New Page button, you can always click the small plus-sign tab at the bottom of the screen, as seen in Figure 1-31.

Renaming Pages

If a page contains a set of elements that you want to reuse (it may be a template page containing a logo and the background for a series of pages in a report, for instance), then you can make duplicates of pages, as follows.

1. Double-click the tab of the page that you want to rename. The existing name will be highlighted.
2. Enter a new name for the page.
3. Press Enter. Click inside the dashboard canvas for the page or click another tab to confirm your changes.

Deleting Pages

If a page is no longer any use to you, then you can delete it, of course.

1. Hover the mouse pointer over the tab for the page that you want to delete. A small cross appears at the top right of the page name, as you can see in Figure 1-31.
2. Click the cross. A warning dialog will appear, as shown in Figure 1-32.

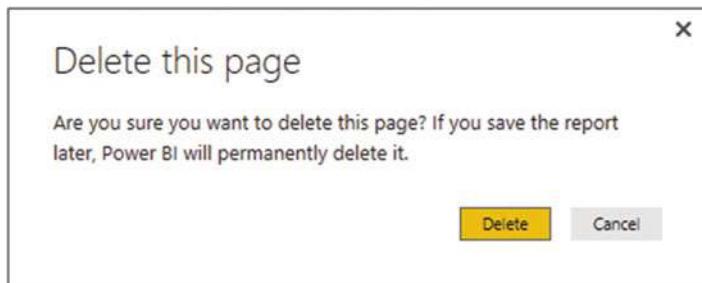


Figure 1-32. The page delete dialog

3. Click Delete. The page will be deleted and *all* visuals on the page are removed from the file.

Moving Pages

To alter the sequencing of the pages in your report, do the following.

1. Click the tab corresponding to the page that you want to move.
2. Drag the page tab left or right to a new position in the set of pages.

Duplicating Pages

If a page contains a set of elements that you want to reuse (it may be a template page containing a logo and the background for a series of pages in a report, for instance), then you can make duplicates of pages, as follows.

1. Hover the mouse pointer over the tab for the page that you want to delete.
2. Right-click the tab. A pop-up menu will appear.
3. Select Duplicate Page.

An identical copy of the page will appear to the right of any existing pages. There is also a Duplicate Page option in the popup menu for the New Page button if you prefer.

Scrolling through Collections of Pages

If your report contains dozens of pages, then it can get very tiring to trawl through the set of pages one at a time. Instead, you can click the page scroll buttons that (see Figure 1-31) to scroll through the set of pages in a Power BI Desktop file.

Importing Excel and Power View Items

Power BI Desktop is not the first incarnation of Power BI; the data model that it uses has been around for some years now. So, you may already be an accomplished Power View expert using Power View for Excel—or you may have advanced data models that you have built using Power Pivot in Excel that you want to transfer into Power BI Desktop.

Fortunately, the team at Microsoft has thought of this, and the result is that you can transfer all your effort from Excel (Power View dashboards, Power Pivot data models, and DAX metrics) into Power BI Desktop with remarkable ease. Here is how.

1. Open a new, blank Power BI Desktop file.
2. In the File menu, select Import ▶ Excel Workbook Contents, as shown in Figure 1-33.

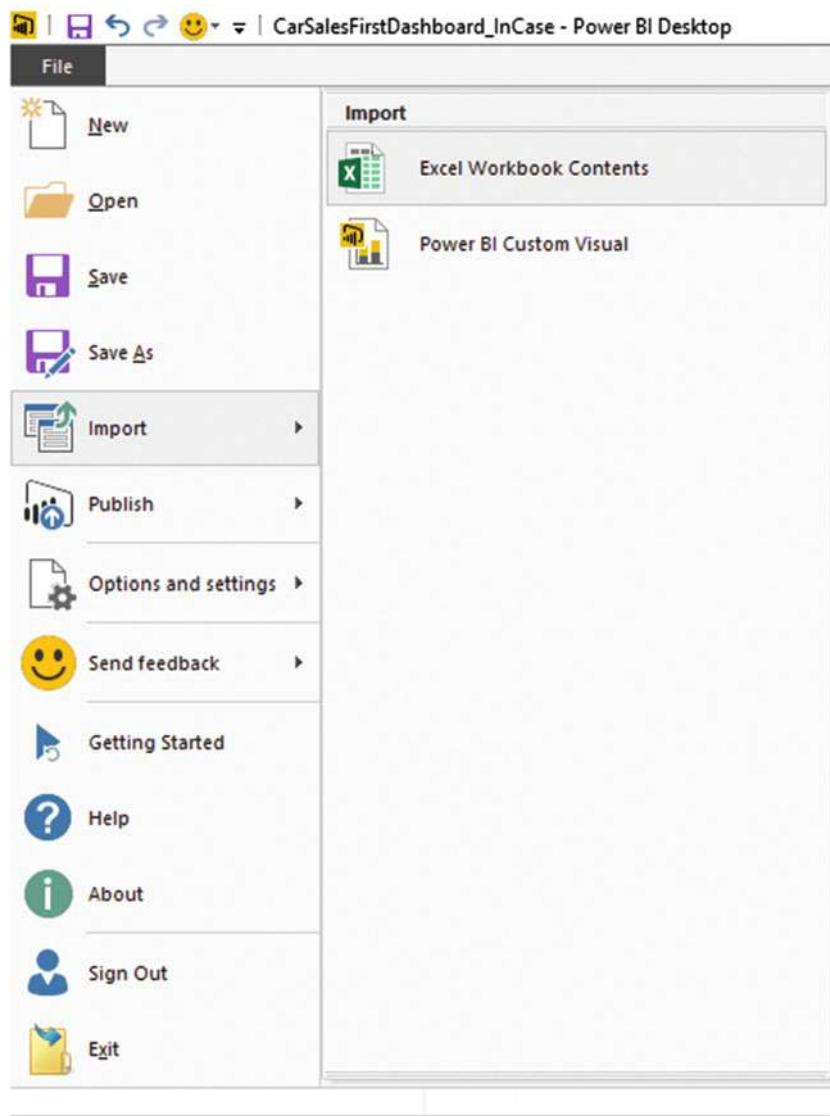


Figure 1-33. Importing existing Power View or Power Pivot items from Excel

3. The Windows Open dialog appears, from which you can select an existing Excel file containing Power View or Power Pivot items. In this example, you can use the file KeySalesData.xlsx from the sample files.
4. Click Open.
5. Power BI Desktop will import any compatible items and display the import screen (as shown in Figure 1-34) during the import process.

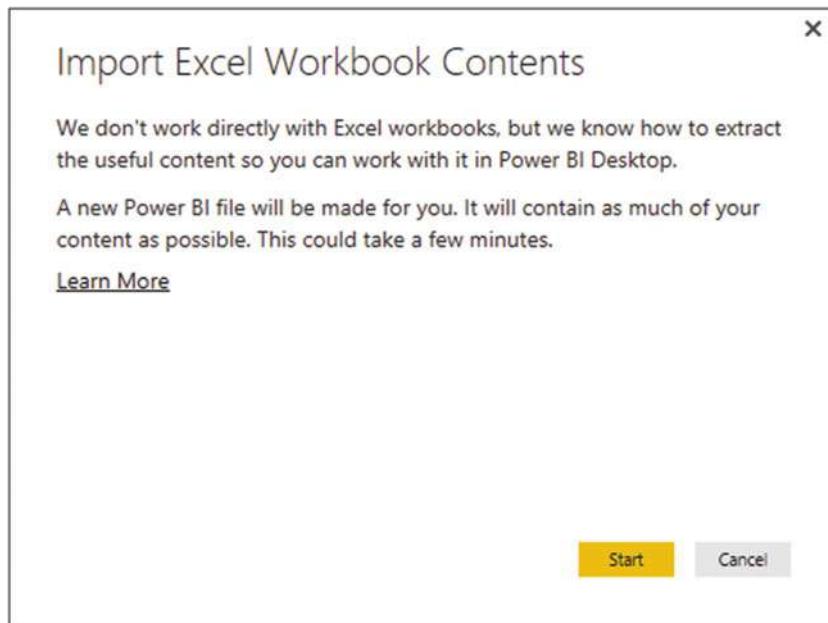


Figure 1-34. Importing Power View and Power Pivot elements from Excel

6. Click Start. Power BI Desktop will begin to load and convert data and elements from Excel. Indeed, you could see further specific questions. Then the import will continue, showing the progress dialog that you see in Figure 1-35.

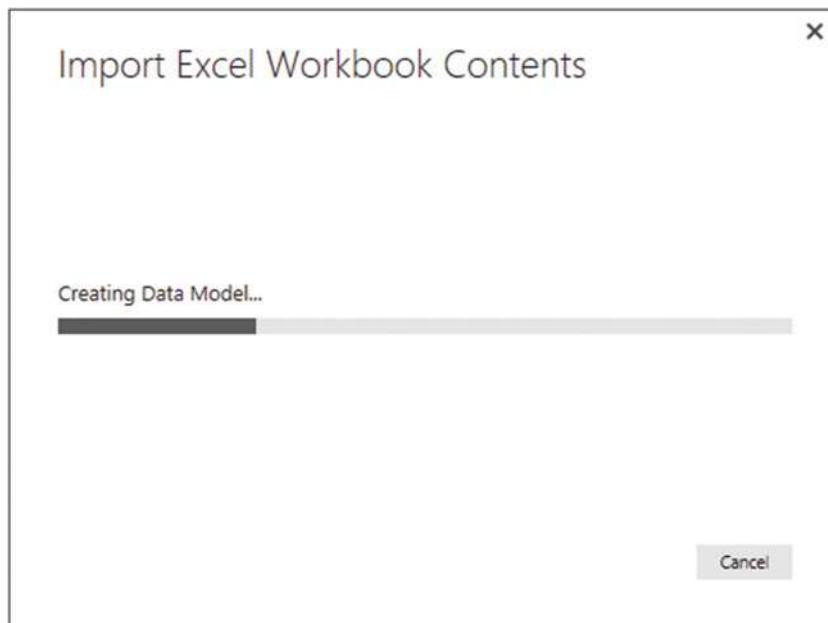


Figure 1-35. The import progress dialog

- Once the import process has successfully finished, Power BI Desktop will display the summary dialog that you see in Figure 1-36.

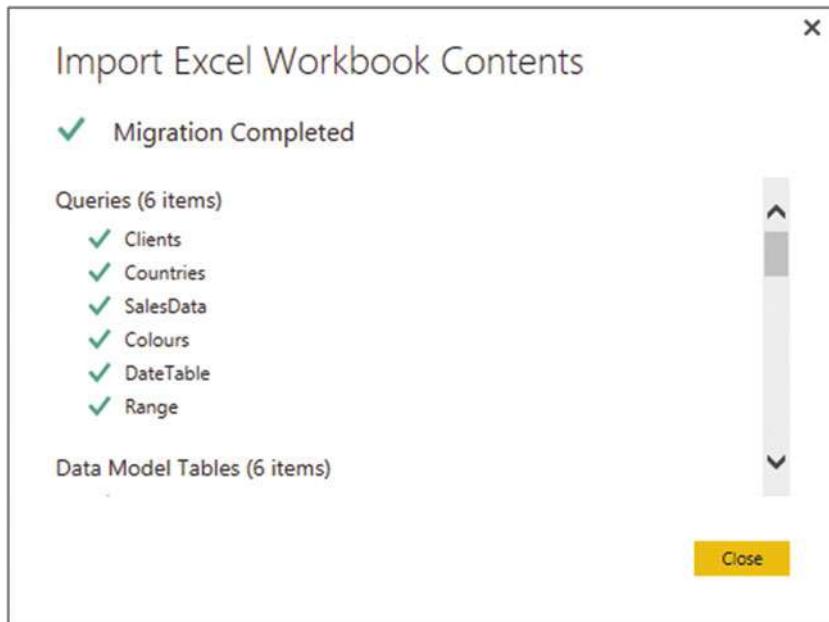


Figure 1-36. The import summary dialog

- Click Close. The items become a Power BI Desktop report.

As this book went to press, there were a few aspects of some Power View visualizations that were not imported, and certain elements (such as KPIs or hierarchies) were not loaded into the Power BI Desktop data model. However, as this technology is currently developing at a rapid pace, you could well find that these minor limitations have been resolved by the time that you read this book. In any case, I advise you to consult the Power BI web site for up-to-date details on any remaining limitations concerning the conversion of Excel objects to Power BI Desktop reports.

Conclusion

Welcome to Power BI Desktop. In a short chapter, you have seen just how amazingly simple and intuitive it is to use this free, self-service business intelligence tool from Microsoft. You have seen how to load data from an external source. This chapter has also given you an idea of the wealth of potential sources of data that Power BI Desktop can handle. You saw how to take data and use it to create tables, charts, maps, and slicers in an interactive dashboard that you can now share with co-workers and friends, if you want to.

Yet all of this merely scratched the surface of the vast potential of this amazing application. As you will discover as you progress through this book, you are on the cusp of discovering a veritable treasure trove of analytical possibilities and stunning visualizations that will help you drive your data analysis and presentation skills to the next level.

CHAPTER 2



Discovering and Loading Data with Power BI Desktop

Before you can present any analysis or insight, you need data. Your sources could be in many places and in many formats. Nonetheless, you need to access them, look at them, select them, and quite possibly clean them up to some extent. You may also need to join many separate data sources before you shape the data into a coherent model that you can use as the foundation for your dashboards and reports. The amazing thing is that you can do all of this using Power BI Desktop without needing any other tools or utilities.

Discovering, loading, cleaning, and modifying source data is one of the areas where Power BI Desktop really shines. Using this latest addition to the Microsoft self-service business intelligence offering, you can carry out

- *Data discovery:* Find and connect to a myriad of data sources containing potentially useful data. This can be from both public and private data sources.
- *Data loading:* Select the data you have examined and load it into Power BI Desktop for shaping.
- *Data modification:* Modify the structure of each data table that you have imported, filter and clean the data itself, and then join any separate data sources (we will look at this in detail in Chapters 3 through 6).

Although I have outlined these three steps as if they are completely separate and sequential, the reality is that they often blend into a single process. Indeed, there could be many occasions when you will examine the data *after* it has been loaded into Power BI Desktop—or join data tables *before* you clean them. The core objective will, however, always remain the same: find some data and then load it into Power BI Desktop where you can tweak, clean, and shape it.

This process could be described simplistically as “First, catch your data.” In the world of data warehousing, the specialists call it ETL, which is short for Extract, Transform, and Load. Despite the reassuring confidence that the acronym brings, this process is rarely a smooth logical progression through a clear-cut series of steps. The reality is often far messier than that. You may often find yourself importing some data, cleaning it, importing some more data from another source, combining the second dataset with the first one, removing some rows and columns, and then repeating many of these operations several times over.

In this chapter and the following one, I will try to show you how the process can work in practice using Power BI Desktop. I hope that this will make the various steps that comprise an ETL process clearer. All I am asking is that you remain aware that the range of options that Power BI Desktop includes make it a multifaceted and tremendously capable tool. The science is to know *which* options to use. The art is to know *when* to use them.

This chapter extends the data load process that you saw briefly in Chapter 1. In the previous chapter, you loaded data directly into Power BI Desktop—or more precisely, into the Power BI Desktop data model. In this chapter, you extend this approach with an additional step. You will load data into the Power BI Desktop Query Editor *before* adding it to the data model. This “detour” is the part of the process that allows you to cleanse and transform the data before it is added to the data model. Of course, if your data is perfect, then you can add it straight into the data model and start building reports. Indeed, if you are connecting to cleansed corporate data, you may want to jump straight to Chapter 6 and learn how to structure the data model. However, if your data needs any adjustment at all, then the Power BI Desktop Query Editor will likely soon become a trusted tool.

This chapter begins by seeing how to find and load data from a variety of sources. Once again, I will be using a set of example files that you can find on the Apress web site. If you have followed the instructions in Appendix A, then these files will be in the C:\PowerBIDesktopSamples folder. Once loaded into Power BI Desktop, this data will become the basis for the dashboards that you will create in the rest of the book.

Data Sources

In the first chapter, you saw how quickly and easily you can load data into Power BI Desktop and create stunning dashboards. It is now time to take a wider look at the *types* of data that Power BI Desktop can ingest and manipulate.

As the sheer wealth of possible data sources can seem overwhelming at first, Power BI Desktop groups potential data sources into the following categories:

- *File*: Includes Excel, CSV (comma-separated values) files, text files, JSON and XML files.
- *Database*: A fairly comprehensive collection of relational databases that are currently in the workplace and in the cloud, including (among others) SQL Server, MS Access, and Oracle. The full list of those available when this book went to press is given later in this chapter.
- *Azure*: This option lets you see all available data that is hosted in the Microsoft Cloud. This covers a range of data formats, from SQL Server through to big data sources.
- *Other*: A considerable and ever-growing range of data sources, from Facebook to Microsoft Exchange. The full list is given later in this chapter.

This list is changing all the time and you need to be aware that you have to look closely at the version of Power BI Desktop that you are using if you want an exhaustive list of the available data sources. I expect that several more will have been added by the time that you read this book.

You can also list the contents of folders on any available local disk or network share (even if it is not always a data source) and then leverage this to import several files at once. Similarly (if you have the necessary permissions), you can list the databases and data available on the database servers you connect to. This way, Power BI Desktop can provide not only the data, but also the metadata—or data about data—that can help you to take a quick look at potential sources of data and only choose those that you really need.

Unfortunately, the sheer range of data sources from which Power BI Desktop can read data is such that we do not have space here to examine the minutiae of every one. Consequently, we will take a rapid tour of *some* of the most frequently used data sources in the next few pages. Fortunately, most of the data sources that Power BI Desktop can read are used in a similar way. The Power BI Desktop interface does a wonderful job of making the arcane connection details as unobtrusive as possible. So even if you are not using the data sources that are described in this chapter, you will nonetheless see a variety of techniques that can be applied to virtually any of the data sources that Power BI Desktop can connect to.

File Sources

Sending files across networks and over the Internet or via email has become second nature to most of us. As long as the files that you have obtained conform to some of the widely recognized standards currently in use (of which you will learn more later), you should have little difficulty loading them into Power BI Desktop.

The file sources that Power BI Desktop can currently read and from which it can load data are given in Table 2-1.

Table 2-1. File Sources

File Sources	Comments
Excel	Allows you to read Microsoft Excel files (versions 97 to 2016) and load worksheets, named ranges, and tables.
CSV	Lets you load text files that conform to the CSV (comma-separated values) format.
XML	Allows you to load XML data.
Text	Lets you load text files using a variety of separators.
Folder	Lets you load the information about all the files in a folder.

Databases

Much corporate data currently resides in relational databases. It follows that being able to look at this data is essential for much of today's business intelligence. In the real world, connecting to corporate data could require you to have a logon name and possibly a password that will let you connect (unless the database can recognize your Windows login). I imagine that you will also require permissions to read the tables and views that contain the data. So the techniques described here are probably the easy bit. The hard part is convincing the guardians of corporate data that you actually *need* the data and you should be allowed to see it.

The databases that Power BI Desktop can currently connect to, and can preview and load data from, are given in Table 2-2.

Table 2-2. Database Sources

Database	Comments
SQL Server	Lets you connect to a Microsoft SQL Server on-premises database and import records from all the data tables and views that you are authorized to access.
Access database	Lets you connect to a Microsoft Access file on your network and load queries and tables.
SQL Server Analysis Services database	Lets you connect to a SQL Server Analysis Services (SSAS) database. This can be either an online analytical processing (OLAP) cube or an in-memory tabular data warehouse.
Oracle database	Lets you connect to an Oracle database and import records from all the data tables and views that you are authorized to access.
IBM DB2 database	Lets you connect to an IBM DB2 database and import records from all the data tables and views that you are authorized to access.

(continued)

Table 2-2. (continued)

Database	Comments
MySQL database	Lets you connect to a MySQL database and import records from all the data tables and views that you are authorized to access.
PostgreSQL database	Lets you connect to a PostgreSQL database and import records from all the data tables and views that you are authorized to access.
Sybase database	Lets you connect to a Sybase database and import records from all the data tables and views that you are authorized to access.
Teradata database	Lets you connect to a Teradata database and import records from all the data tables and views that you are authorized to access.
SAP Hana database	Lets you connect to a SAP Hana in-memory database and import records from all the data tables and views that you are authorized to access.

Connecting to Oracle, DB2, MySQL, PostgreSQL, Sybase, or Teradata requires not only that the database administrator has given you the necessary permissions, but also that connection software (known as *drivers* or *providers*) has been installed on your PC. Given the “corporate” nature of the requirements, it may help if you talk directly to your IT department to get this set up in your enterprise IT landscape.

Azure

Azure is the Microsoft Cloud. The Azure data sources that Power BI Desktop can currently connect to, and can preview and load data from, are given in Table 2-3.

Table 2-3. Azure Sources

Source	Comments
Microsoft Azure SQL Database	Lets you connect to a Microsoft SQL Server cloud-based database and import records from all the data tables and views that you are authorized to access.
Microsoft Azure SQL data warehouse	Lets you connect to Microsoft’s cloud-based, elastic, enterprise data warehouse.
Microsoft Azure Marketplace	Lets you load data that you are authorized to access on the Microsoft Azure Marketplace. It requires a Microsoft Azure Marketplace subscription.
Microsoft Azure HDInsight	Reads cloud-based Hadoop files in the Microsoft Azure environment.
Microsoft Azure Blob Storage	Reads from a cloud-based unstructured data store.
Microsoft Azure Table Storage	Reads from Microsoft Azure tables.
Azure HDInsight Spark	Lets you connect to Microsoft’s parallel-processing framework in the Cloud.
Microsoft Azure DocumentDB	Lets you connect to Microsoft’s NoSQL database.
Microsoft Azure Data Lake Store	Lets you connect to Microsoft’s raw data cloud storage.

Other Sources

Up until now, you have seen some of the more traditional sources of data that you might need for your analysis. As the world changes, the available sources evolve. The current trend is toward less “structured” (and controlled) data sources and more varied and often less corporate sources.

Power BI Desktop can connect to, and read data from, a whole host of these less classic sources. Some of the most used other sources that it can currently connect to are listed in Table 2-4.

Table 2-4. Other Sources

Source	Comments
Web	Connects to a web page and reads tables of data.
SharePoint list	Loads a SharePoint list as a data table. You will need SharePoint permissions to access SharePoint data.
OData feed	Connects to an OData feed to read and load the data it contains. OData is a standardized protocol for creating and consuming data, especially over the Internet.
Hadoop Distributed File System (HDFS)	Reads Hadoop (“big data”) files.
Active Directory	Reads data from the Enterprise Active Directory. This will probably require custom access rights.
Dynamics CRM Online	Reads data from the Microsoft Dynamics Business Solutions.
Facebook	Connects to a Facebook profile and downloads the data.
MS Exchange	Reads data from the Microsoft Exchange Email system.
Facebook	Reads Facebook data.
Google Analytics	Connects to Google Analytics so that you can monitor your web site usage.
Salesforce objects	Connects to Salesforce objects and downloads data for analysis.
Salesforce reports	Connects to Salesforce reports and downloads data for analysis.
ODBC	Connects to an ODBC driver. This is a standard data interface that can connect to an extremely wide range of data sources.

These data sources are so varied, and so often customized or uniquely personal, that I will not be going through anything other than web page sources in this chapter.

Note The list of data sources that Power BI Desktop can access is growing all the time. Consequently, when you read this book you will probably find even more sources than those described so far.

Loading Data

It is time to start looking at the heavy-lifting aspect of Power BI Desktop and how you can use it to load data from a variety of different sources. I will begin on the bunny slopes with a simple example of “scraping” data from a web page. Then, given the plethora of available data sources, and to give the process a clearer structure, we will load data from several of the ubiquitous data sources that are found in most workplaces. These data sources are the basis of the data that you will learn to tweak and “mash up” in the following chapters. This data will also become the basis of many of the dashboards that you will create in Chapters 10 to 15. These sources are as follows:

- *CSV*: This file type will be the source of the Countries table.
- *Text*: Here we will use the Stock table.
- *XML*: We will use an XML file containing the Colors data.
- *Excel*: We will use the Invoices and InvoiceLines tables from an Excel file.
- *Access*: We will load the Clients table from an Access database file.

After we have loaded these six tables, we will look into getting data from a relational database (SQL Server will be the example here), because in my experience, databases are a frequent source of core data for analysis. Here you will see how to apply several tricks and techniques to database sources.

Finally, we will load data from a couple of “all-in-one” sources that contain *all* the data that you need in one place to create dashboards without any extensive data preparation. These sources are

- *SQL Server Analysis Services OLAP cube*: We will load a much simplified subset of data from an Enterprise data warehouse “cube.”
- *SQL Server Analysis Services Tabular data warehouse*: We will read data directly from an in-memory data warehouse.

Finally, we will look at a technique for loading multiple files of the same format, as this can be a frequent requirement in the real world.

Web Pages

As a first and extremely simple example, let’s grab some data from a web page. Since I want to concentrate on the method rather than the data, I will use a web page that has nothing to do with the sample data in the book. We will not be using this other than as a simple introduction to the process of loading data from web pages using Power BI Desktop.

Assuming that you have launched Power BI Desktop and closed the splash screen...

1. Click the small triangle at the bottom of the Get Data button in the Home ribbon.
2. Select Web from the menu that appears, as shown in Figure 2-1.

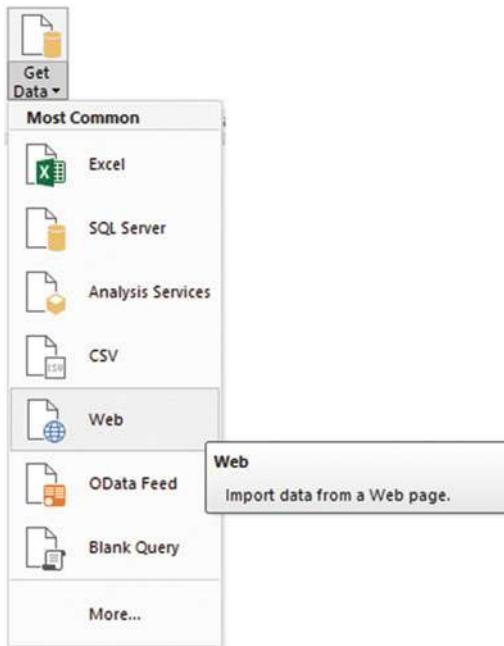


Figure 2-1. The Get Data menu

3. Enter the following URL (it is a Microsoft help page for Power BI Desktop that contains a few tables of data): <http://office.microsoft.com/en-gb/excel-help/guide-to-the-power-query-ribbon-HA103993930.aspx>. I am, of course, hoping that it is still available when you read this book. Of course, if you have a URL that you want to try out, then feel free! The dialog will look something like Figure 2-2.

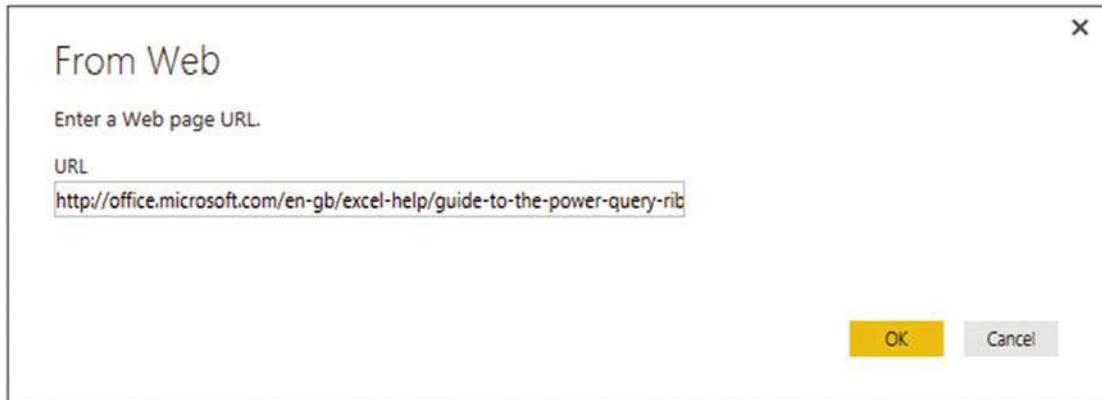


Figure 2-2. The From Web source dialog

4. Click OK. The Navigator dialog will appear. After a few seconds, during which Power BI Desktop is connecting to the web page, the list of available tables of data in the web page will be displayed.
5. Click one of the table names on the left of the Navigator dialog. The contents of the table will appear on the right of the Navigator dialog to show you what the data in the chosen table looks like, as shown in Figure 2-3.

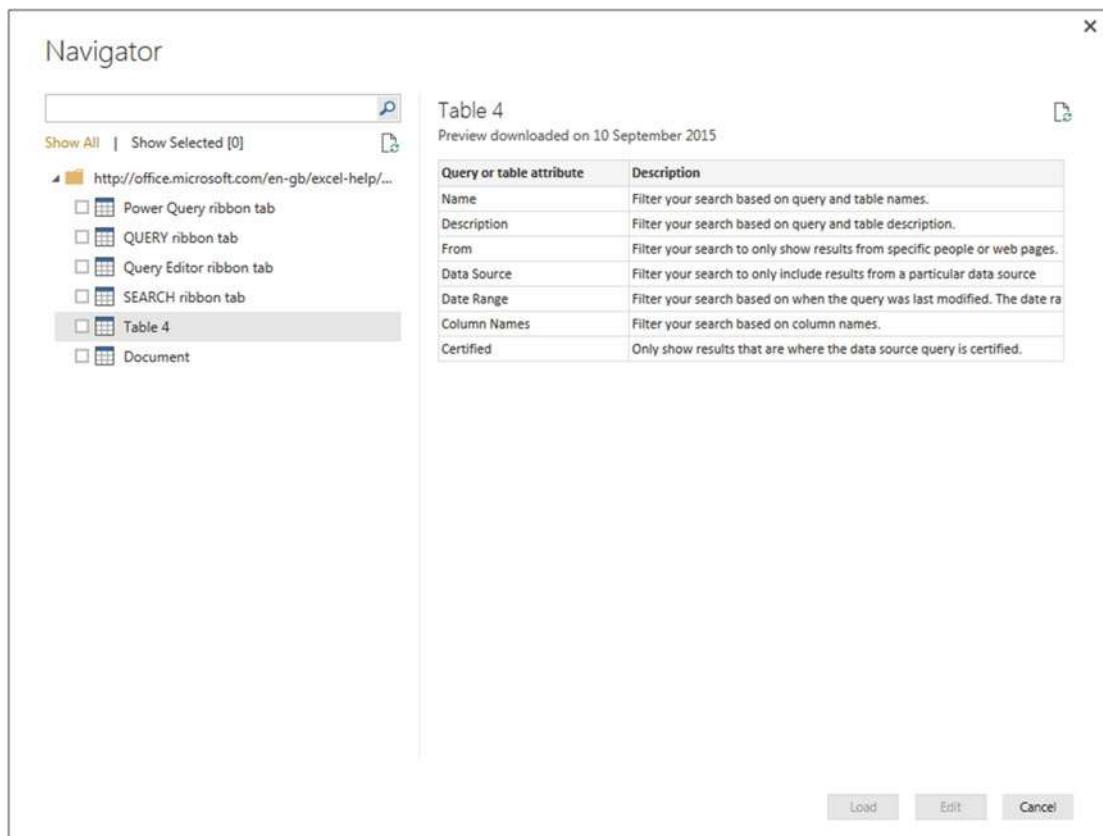


Figure 2-3. The Navigator dialog previewing the contents of a table on a web page

6. Select the check box in the Navigator dialog (shown to the left of Table 4 in Figure 2-3).
7. Click Edit at the bottom of the window (or double-click the table name). The Power BI Desktop Query window opens to display the table of data. It should look like Figure 2-4. (We will look at this window in detail in the next chapter.)

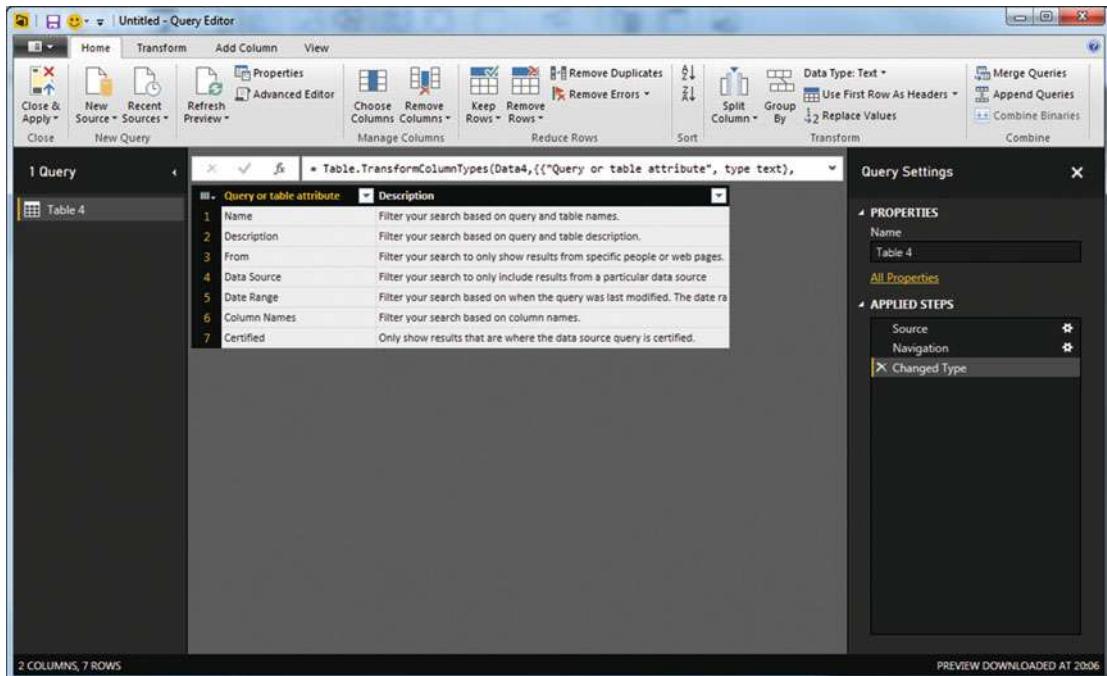


Figure 2-4. The Power BI Desktop Query window

8. Click Close & Apply in the Power BI Desktop ribbon. The Power BI Desktop Query window will close and copy the data into the Power BI Desktop data model.
9. Because we will not be using this data (it was only an example of how to load data from a table on a web page), click the Power BI Desktop System menu, click Exit to close Power BI Desktop Query, and then click Don't Save. You will return to the Power BI Desktop window.

Tip Another way of accessing web pages is to click Get Data ▶ More ▶ File and select Other in the Get Data dialog. You can then select Web in the list on the right of the Get Data dialog.

This simple example showed how you can load data from a supported data source and load it into Power BI Desktop Query. What is new here (compared to what you saw in Chapter 1) is the Power BI Desktop Query window, where you can modify and transform the original data.

CSV Files

The scenario is as follows: you have been given a comma-separated text file (also known as a *CSV file*) containing a list of data. You now want to load this into Power BI Desktop so that you can look at the data and consider what needs to be done (if anything) to make it useable. The following explains what you have to do.

1. In the Power BI Desktop ribbon, click the small triangle at the bottom of the Get Data button, and then click CSV. The Open dialog will appear.
2. Navigate to the folder containing the file that you want to load and select it (C:\PowerBIDesktopSamples\Countries.csv, in this example).
3. Click Open. A dialog will display the initial contents of the file, as shown in Figure 2-5.

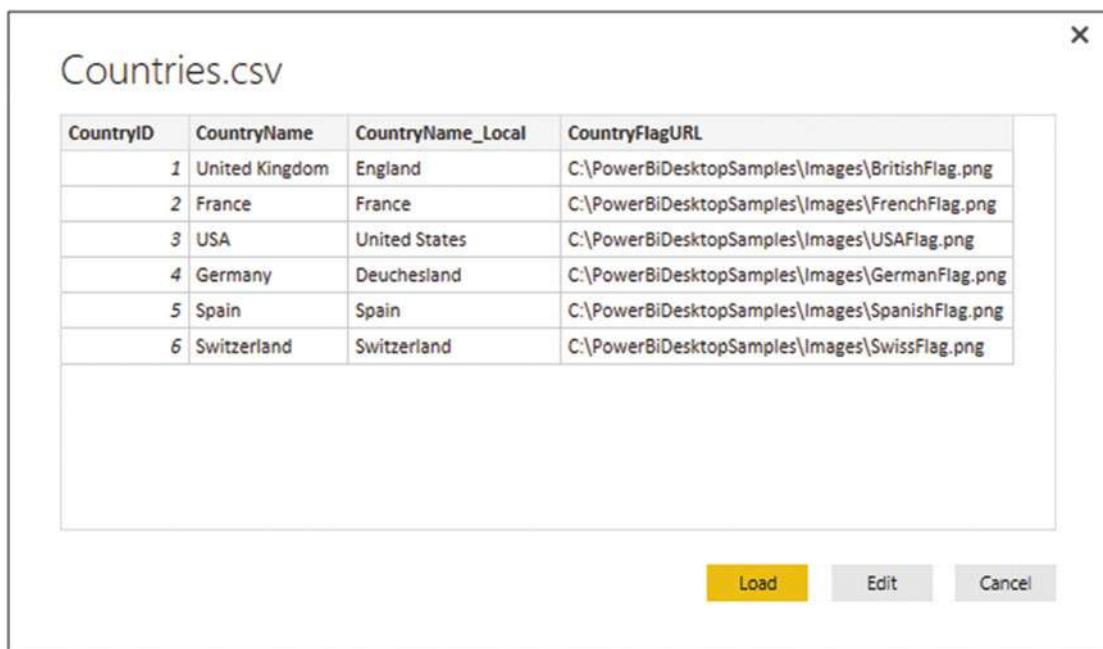


Figure 2-5. The Power BI Desktop file dialog

4. Click the Edit button. The Power BI Desktop Query window appears; it contains a sample of the contents of the CSV file—or possibly the entire file if it is not too large.
5. Click the Close and Apply button in the Power BI Desktop Query window (you can see this at the top left in Figure 2-4). You will see that the Countries table appears in the Fields list on the right of the screen, as shown in Figure 2-6.

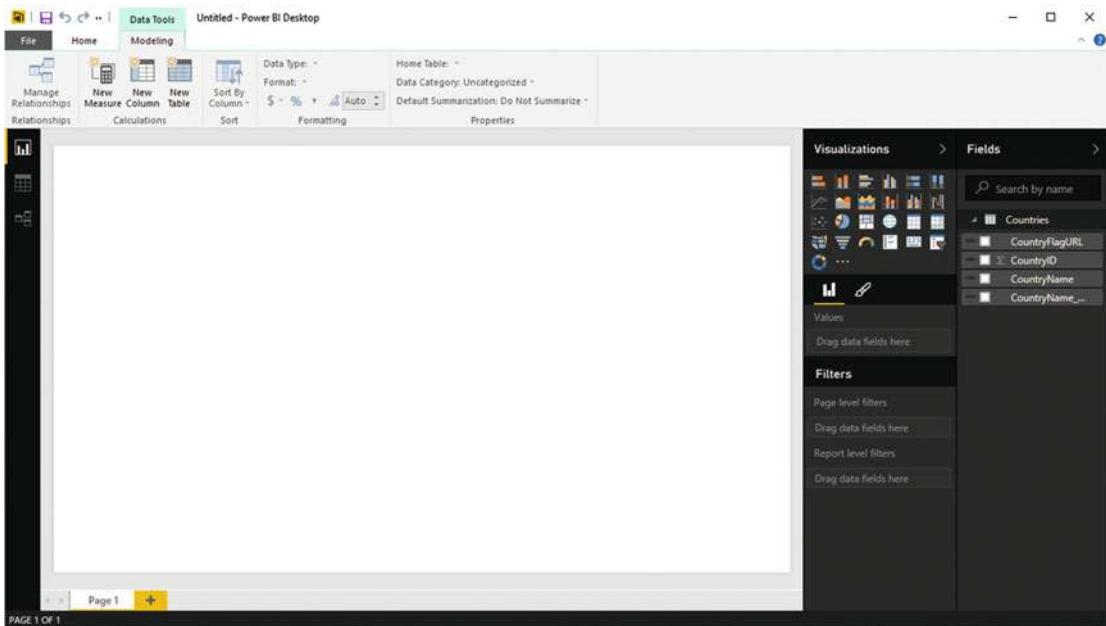


Figure 2-6. The Power BI Desktop Query window with a file loaded

And that, for the moment, is that. You have loaded the file into Power BI Desktop in a matter of a few clicks and it is ready for use in dashboards and reports. If you expand the Countries table in the Fields pane of Power BI Desktop, you see the fields that this table contains—just as they were in the dialog shown in Figure 2-5.

The important thing to note here is that although you used two completely different data sources, the process that you applied was virtually identical. This really is one of the great strengths of Power BI Desktop.

In the upcoming chapters, you will learn how to shape the data. For the moment, however, let's continue looking at some other data sources.

What Is a CSV File?

Before we move on to other file types, there are a few comments I need to make about CSV files. There is a technical specification of what a “true” CSV file is, but I won’t bore you with that. What’s more, many programs that generate CSV files do not always follow the definition exactly. What matters is that Power BI Desktop can handle text files that

- Have a .csv extension (it uses this by default to apply the right kind of processing).
- Use a comma to separate the elements in a row. This, too, is a default that can be overridden.
- End with a line feed, carriage return, or line feed/carriage return.
- Can, optionally, contain double quotes to encapsulate fields. These will be stripped out as part of the data load process. If there are double quotes, they do not have to appear for every field, nor even for every record in a field that can have occasionally inconsistent double quotes.

- Can contain “irregular” records; that is, rows that do not have every element found in a standard record. However, the first row (whether or not it contains titles) must cover every element found in all the remaining records in the list. Put simply, any other record can be shorter than the first one but cannot be longer.
- Do not contain anything other than the data table. If the file contains header rows or footer rows that are not part of the data, then Power BI Desktop cannot load the data table without further work. There are workarounds to this all-too-frequent problem; one is given in Chapters 3 and 4.

Note Another way of accessing CSV files is to click Get Data ▶ More ▶ File and select Text in the Get Data dialog. You can then select CSV on the right of the Get Data dialog.

Text Files

If you followed the process for loading a CSV file in the previous section, then you will find that loading a text file is very similar. This is not surprising. Both are text files and both should contain a single list of data. The following are the core differences:

- A text file can have something *other* than a comma to separate the elements in a list. You can specify the delimiter when defining the load step.
- A text file should normally have the extension .txt (though this, too, can be overridden).
- A text file *must* be perfectly formed; that is, every record (row) must have the same number of elements as every other record.
- A text file, too, *must not* contain anything other than the data table if you want a flawless data load the first time.
- If a text file encounters difficulties, it should import the data as a single column that you can then try and split up into multiple columns, as described in Chapter 5.

Here, then, is how to load a text file into Power BI Desktop:

1. In the Power BI Desktop ribbon, click Get Data ▶ More ▶ File. The Get Data dialog will look like Figure 2-7.

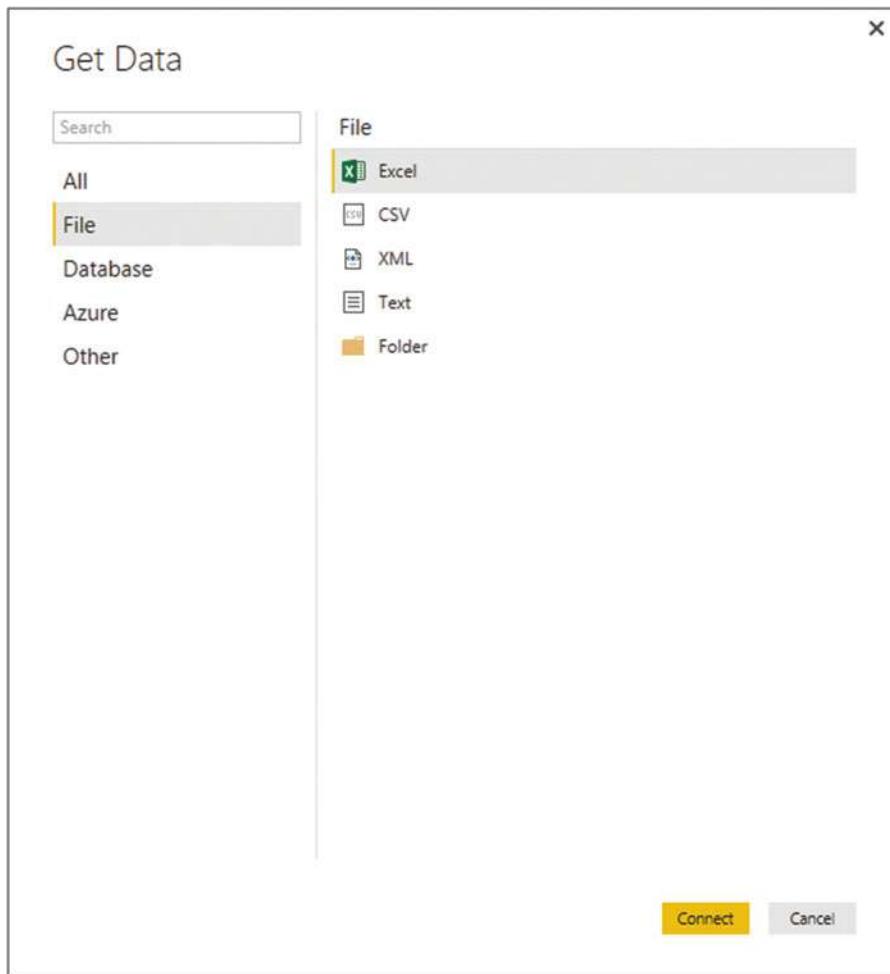


Figure 2-7. The Get Data dialog

2. Select Text from the list on the right and then click Connect. The Open dialog will be displayed.
3. Navigate to the folder containing the file and select the file (C:\PowerBIDesktopSamples\CountryList.txt, in this example).
4. Click Open. A dialog will display the contents of the file.
5. Click the Cancel button (because after a quick look at the contents of the file you have decided that you do not really need it).

Where Power BI Desktop is really clever is that it can make a very educated guess as to how the text file is structured; that is, it can nearly always guess the field separator (the character that isolates each element in a list from the other elements). And so not only will it break the list into columns, but it will also avoid importing the column separator. If it does not guess correctly, then don't despair. You will learn how to correct this in Chapter 3.

Looking at the contents of a file and then deciding not to use it is part and parcel of the *data discovery* process that you will find yourself using when you work with Power BI Desktop. The point of this exercise is to show you how easy it is to glance inside potential data sources and then decide whether to import them into the data model or not. What's more, this short exercise introduced you to the Get Data dialog, where you can always refer (in the All pane) to the complete list of data sources that Power BI Desktop can handle.

Tip At the risk of stating the obvious, you can press Enter to accept a default choice in a dialog and press Esc to cancel out of the dialog.

XML Files

XML, or Extensible Markup Language, is a standard means of sending data between IT systems. Consequently, you have every chance of having to load an XML file one day. Although an XML file is just text, it is text that has been formatted in a very specific way, as you can see if you ever open a XML file in a text editor such as Notepad. Do the following to load an XML file.

1. In the Power BI Desktop ribbon, click Get Data and then click More. Next, in the Get Data dialog, select File and XML.
2. Click Connect. The Open dialog will appear.
3. Navigate to the folder containing the file and select the file (C:\PowerBIDesktopSamples\ColoursTable.xml, in this example).
4. Click Open. The Navigator dialog will open.
5. Click the Colors table in the left-hand pane of the Navigator dialog. The contents of the XML file will be displayed on the right of the Navigator dialog, as shown in Figure 2-8.

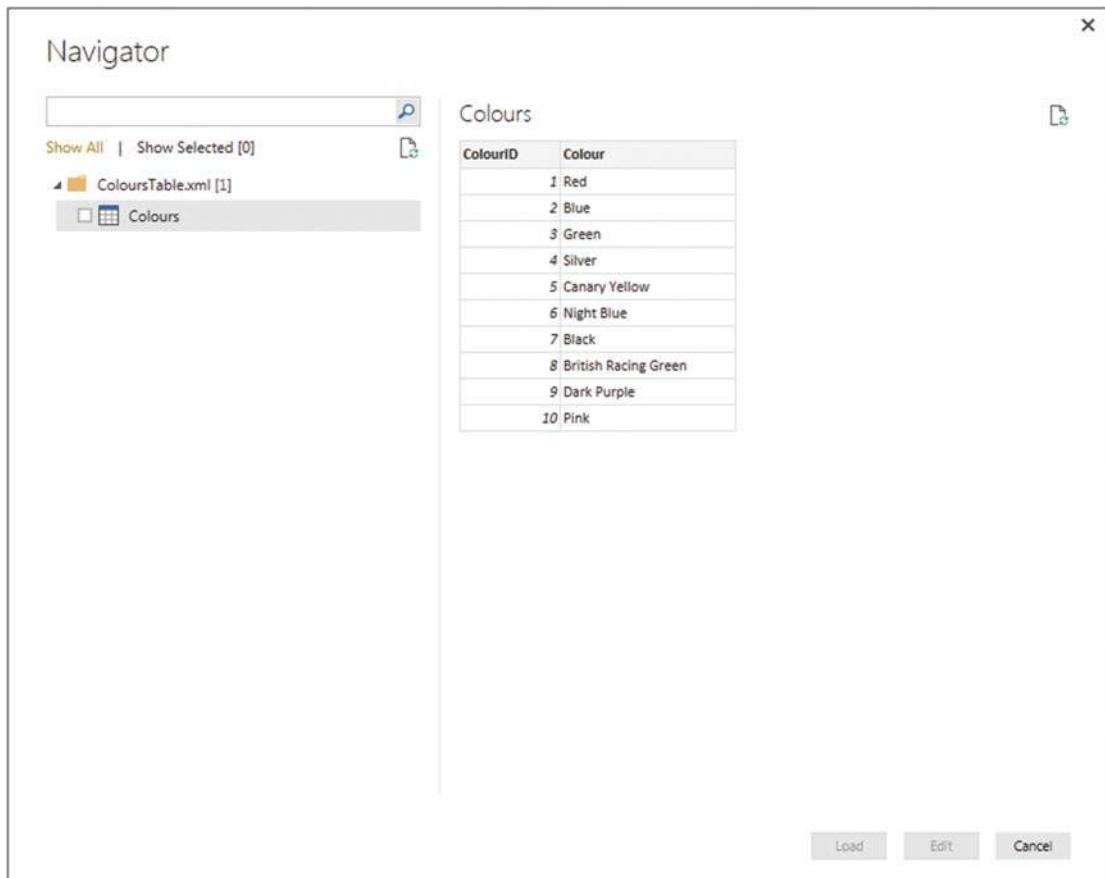


Figure 2-8. The Navigator dialog before loading an XML file

6. Click the check box to the left of the Colors table on the left.
7. Click the Edit button. The Power BI Desktop Data window will display the contents of the XML file.
8. Click the Close and Apply button in the Power BI Desktop Data window. You will see that the Colors table appears in the Fields list on the right of the screen.

The actual internal format of an XML file can get extremely complex. Sometimes an XML file will contain only one data table; sometimes it will contain many data separate tables. On other occasions, it will contain one table whose records contain nested levels of data that you need to handle by expanding a hierarchy of elements. These techniques are described later in this chapter in the context of database sources.

Note Certain types of data source allow you to load multiple sets of data simultaneously. XML files (unlike CSV and text files) can contain multiple independent datasets. You can load several “tables” of data simultaneously by selecting the check box to the left of each dataset that you want to load from the XML file.

Excel

You are probably already a major Excel user and have many, many spreadsheets full of data that you want to rationalize and use for analysis and presentation. So, let's see how to load the contents of an Excel file.

1. In the Power BI Desktop ribbon, click the small triangle at the bottom of the Get Data button and then click Excel. The Open dialog will appear.
2. Navigate to the directory containing the file that you want to look at (C:\PowerBIDesktopSamples, in this example).
3. Select the source file (InvoicesAndInvoiceLines.xlsx, in this example) and click OK. The Navigator dialog will appear, showing the worksheets, tables, and ranges in the workbook file, as shown in Figure 2-9.

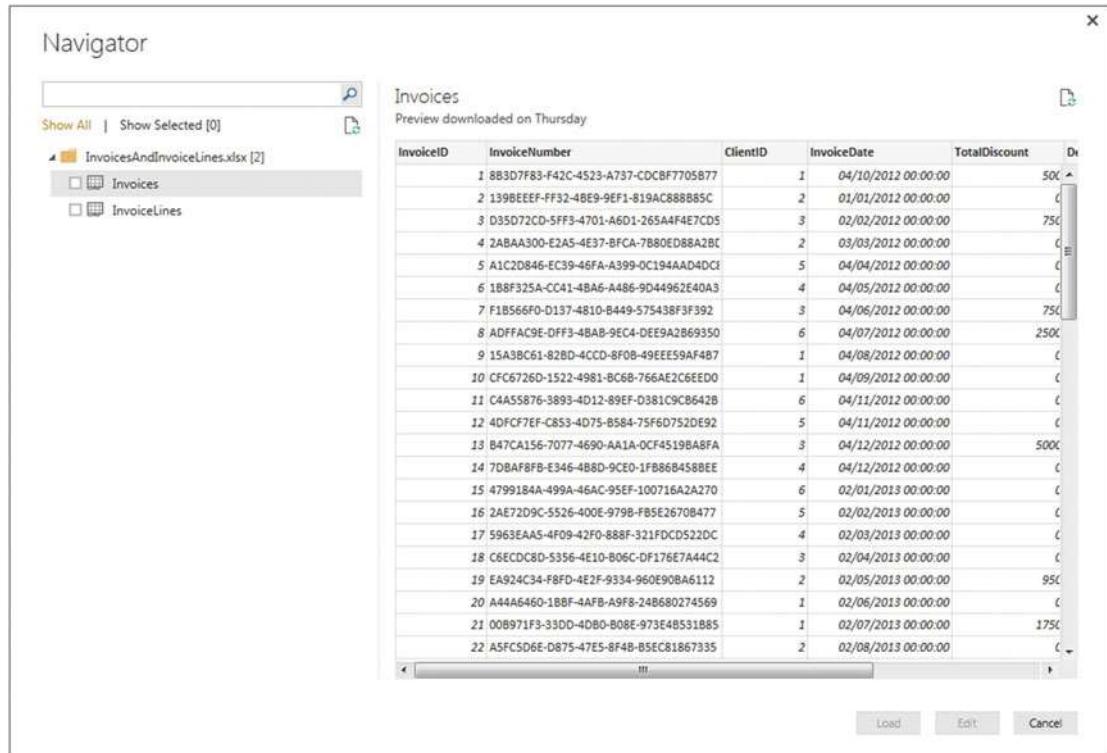


Figure 2-9. The Navigator dialog before loading data from an Excel Workbook

4. Click one of the tables listed on the left of the Navigator dialog. The top few rows of the selected spreadsheet will appear on the right of the dialog to show you what the data in the chosen table looks like.
5. Click the check boxes to the left of the Invoices and InvoiceLines tables on the left.
6. Click Load. The selected worksheets will be loaded into the Power BI Desktop data model and will appear in the Fields list in the Report window.

As you can see from this simple example, having Power BI Desktop read Excel data is really not difficult. You could have edited this data in Power BI Desktop Query before loading it, but as the data seemed clean and ready to use, I preferred to load it straight into Power BI Desktop (or rather the Power BI Desktop data model). As well, you saw that Power BI Desktop can load multiple tables at the same time from a single data source. However, you might still be wondering about a couple of things that you saw during this process, so here are some anticipatory comments:

The Navigator dialog displays

- worksheets
- named ranges
- named tables

Each of these elements is represented by a different icon in the Navigator dialog. Sometimes these can, in effect, be duplicate references to the same data, so you should really use the most precise data source that you can. For instance, I advise using a named table or a range name rather than a worksheet source, as the latter could easily end up containing “noise” data (that is data from outside the rows and columns that interest you), which would make the load process more complex than it really needs to be.

Power BI Desktop will list and use data connections to external data sources in a source Excel workbook *if* the data connection is active and has returned data to the workbook. Once a link to Power BI Desktop has been established, you can delete the data table itself in the source Excel workbook—and still use Power BI Desktop to load the data over the data connection in the source workbook.

Power BI Desktop will not take into account any data filters on an Excel data table. Consequently, you will have to reapply any filters (of which you’ll learn more in the next chapter) in Power BI Desktop if you want to subset the source data.

Microsoft Access Databases

Another well-used data repository that proliferates in many corporations today is Microsoft Access. It is a powerful relational desktop database and can contain multiple tables, each containing millions of records. So we need to see how to load data from this particular source. Because this process uses the same dialogs that you saw previously, I will not show them again here.

1. In the Power BI Desktop ribbon, click Get Data ▶ More ▶ Database and select Access Database in the Get Data dialog.
2. Click Connect and navigate to the MS Access database containing the data that you want to load (C:\PowerBIDesktopSamples\ClientsDatabase.accdb in this example).
3. Select the Access file and click OK. The Navigator dialog appears; it lists all the tables and queries in the Access database.
4. Check the check box for the ClientList table and click Load. The Power BI Desktop window opens and displays the table in the Fields list in the Report window.

I am sure that you can see a pattern emerging here. Indeed, this pattern will continue as you progress to loading tables from relational databases in a few pages time. The process is nearly always

- Knows the type of source data that you want to look at
- Finds the file, database, or connection that lets you access the data
- Allows you to examine the data table(s) and to select the ones that you want to load

Note Power BI Desktop cannot see linked tables, only imported tables or tables that are actually in the Access database. It can, however, read queries overlaid upon native, linked, or imported data.

Once you have loaded the Excel, XML, CSV, and text data, save the Power BI Desktop file as CarSalesFirstDashboard.pbix.

Relational Databases: SQL Server

Enterprise-grade relational databases still hold much of the world's data, so you really need to know how to tap into the vast mines of information that they contain. The bad news is that there are many, many databases out there, each with their intricacies and quirks. The good news is that once you have learned to load data from *one* of them, you should be able to use *any* of them.

Here I will use the Microsoft enterprise relational database—SQL Server—as an example to show you how to load data from a database into Power BI Desktop. The first advantage of this setup is that you probably do not need to install any software to enable access to SQL Server (although this is not always the case, so talk this through with your IT department). A second advantage is that the techniques are pretty similar to those used and applied by Oracle, DB2, and the other databases to which Power BI Desktop can connect. Furthermore, you can load multiple tables or views from a database at once. To see this in action (and presuming that you have created the database CarSalesData as described in Appendix A), take the following steps:

1. Open a new Power BI Desktop application.
2. In the Power BI Desktop ribbon, click the small triangle at the bottom of the Get Data button and then click SQL Server. The SQL Server Database dialog will appear.
3. Enter the server name in the Server text box. This will be the name of your SQL Server or one of the SQL Server resources used by your organization.
4. Enter the database name; if you are using the sample data, it will be CarSalesData. The dialog will look like Figure 2-10.

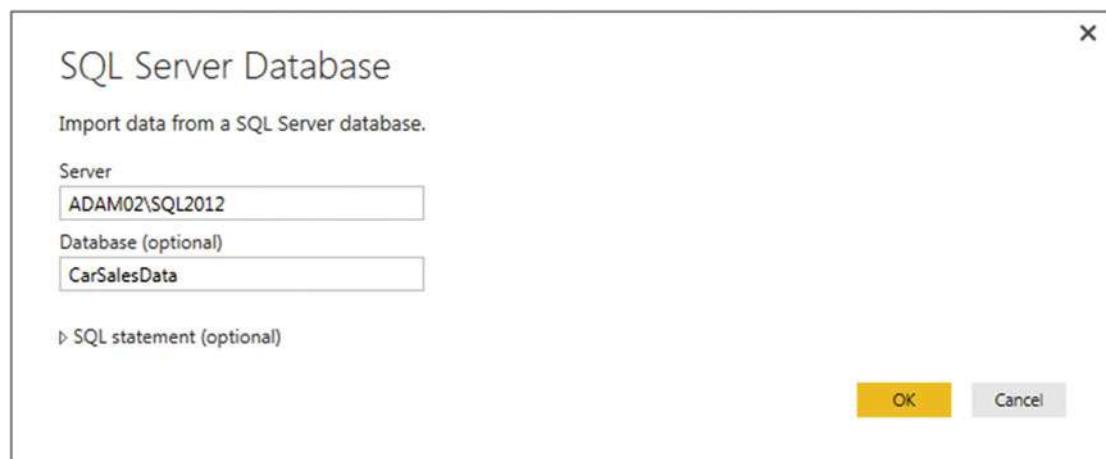


Figure 2-10. The Microsoft SQL Database dialog

5. Click OK. The Access a SQL Server Database dialog will appear. Assuming that you are authorized to use your Windows login to connect to the database, leave “Use my current credentials” selected, as shown in Figure 2-11.

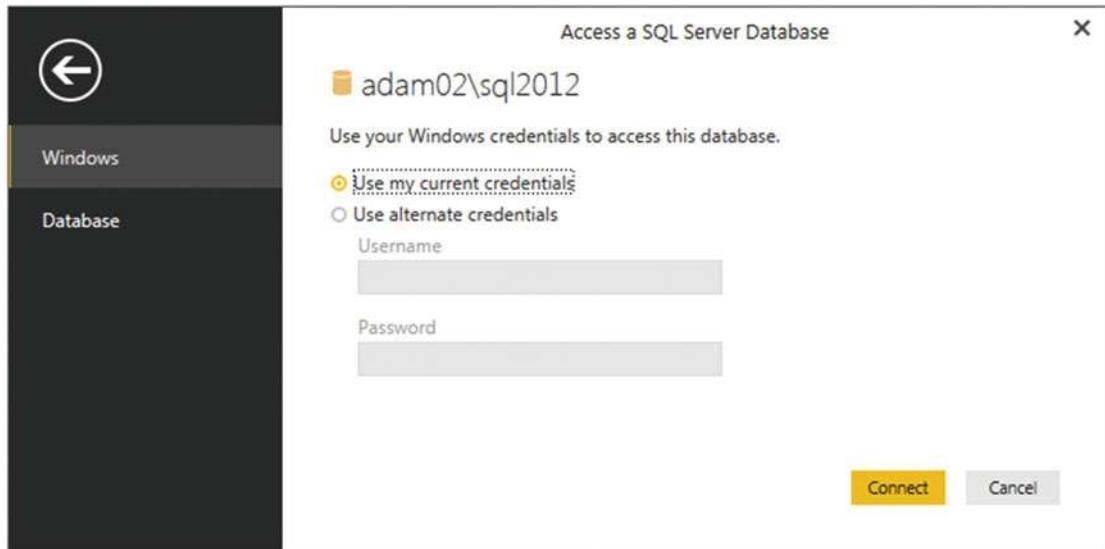


Figure 2-11. The Access a SQL Server Database dialog

6. Click Connect. Power BI Desktop will connect to the server and display the Navigator dialog containing all the tables and views in the database that you have permission to see on the server you selected. In some cases, you could see a dialog saying that the data source does not support encryption. If you feel happy with an unencrypted connection, then click the OK button for this dialog.
7. Click the check boxes for the Clients, Colors, Countries, Invoices, InvoiceLines, and Stock tables. The data for the most recently selected data appears on the right of the Navigator dialog, as shown in Figure 2-12.

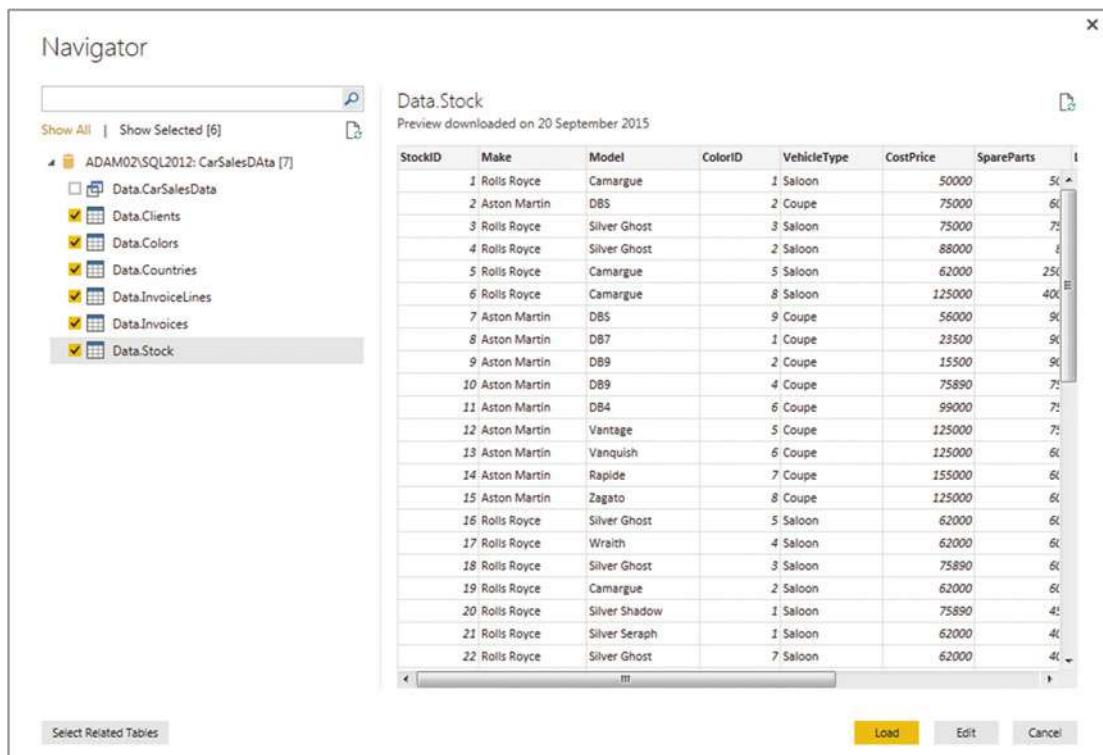


Figure 2-12. The Navigator dialog when selecting multiple items

8. Click Load. The connection Settings dialog will appear. Leave Import selected.
9. The Power BI Desktop window will open and display the tables that you selected in the Fields list in the Report window when you click OK.

Since this is very similar to the way in which you loaded data from Access, I imagine that you are getting the hang of things by now.

Tip When selecting multiple tables or views, you will only ever see the contents of a single data source in the Navigator dialog. However, you can preview the contents of any of the selected data sources (or even any that are not selected) simply by clicking the table or view name. This will not affect the choice of selected tables and views that you want to load into Power BI Desktop.

Automatically Loading Related Tables

Relational databases are nearly always intricate structures composed of many interdependent tables. Indeed, you will frequently need to load several tables to obtain all the data that you need.

Knowing which tables to select is not always easy. Power BI Desktop tries to help you by automatically detecting the links between tables in the source database; this way, you can rapidly isolate the collections of tables that have been designed to work together.

Do the following to see a related group of tables.

1. Connect to the source database as described in the previous section.
2. In the Navigator dialog, click a table that contains data that you need.
3. Click the “Select related tables” button.

Any tables that are linked in the database are selected. You can deselect any tables that you do not want, of course. More importantly, you can click the names of the selected tables to see their contents.

Database Options

The world of relational databases is—fortunately or unfortunately—a little more complex than the world of files or MS Access. Consequently, there are a few comments to make about using databases as a data source; specifically, how to connect to them.

First, let's cover the initial connection to the server. The options are explained in Table 2-5.

Table 2-5. Database Connection Options

Option	Comments
Server	You cannot browse to find the server and you need to type or paste the server name. If the server has an instance name, you need to enter the server and the instance. Your IT department will be able to supply this if you are working in a corporate environment.
Database	If you know the database, then you can enter (or paste) it here. This restricts the number of available tables in the Navigator dialog and makes finding the correct table or view easier.
SQL statement	You can enter a valid snippet of T-SQL that returns data from the database.

Note These options apply to most databases and certainly to SQL Server. However, they vary depending on the make of database that you are connecting to.

These options probably require a little more explanation. So let's look at each one in turn.

Server Connection

It is fundamental that you know the exact connection string for the database that you want to connect to. This could be the following:

- The database server name
- The database server name, a backslash, and an instance name (if there is one)
- The database server IP address
- The database server IP address, a backslash, and an instance name (if there is one)

Note A database instance is a separate SQL Server service running alongside others on the same physical or virtual server. You will always need both the server and this instance name (if there is one) to successfully connect. You can also specify a timeout period if you wish.

Most SQL Server instances host many, many databases. Sometimes these can number in the hundreds. Sometimes, inevitably, you cannot remember which database you want to connect to. Fortunately, Power BI Desktop can let you browse the databases on a server. To do this, do the following:

1. In the Power BI Desktop ribbon, click the small triangle at the bottom of the Get Data button and then click SQL Server. The SQL Server Database dialog will appear.
2. Enter the server name in the Server text box and click OK. The Navigator window opens, as shown in Figure 2-13. Of course, the actual contents depend on the server that you are connecting to.

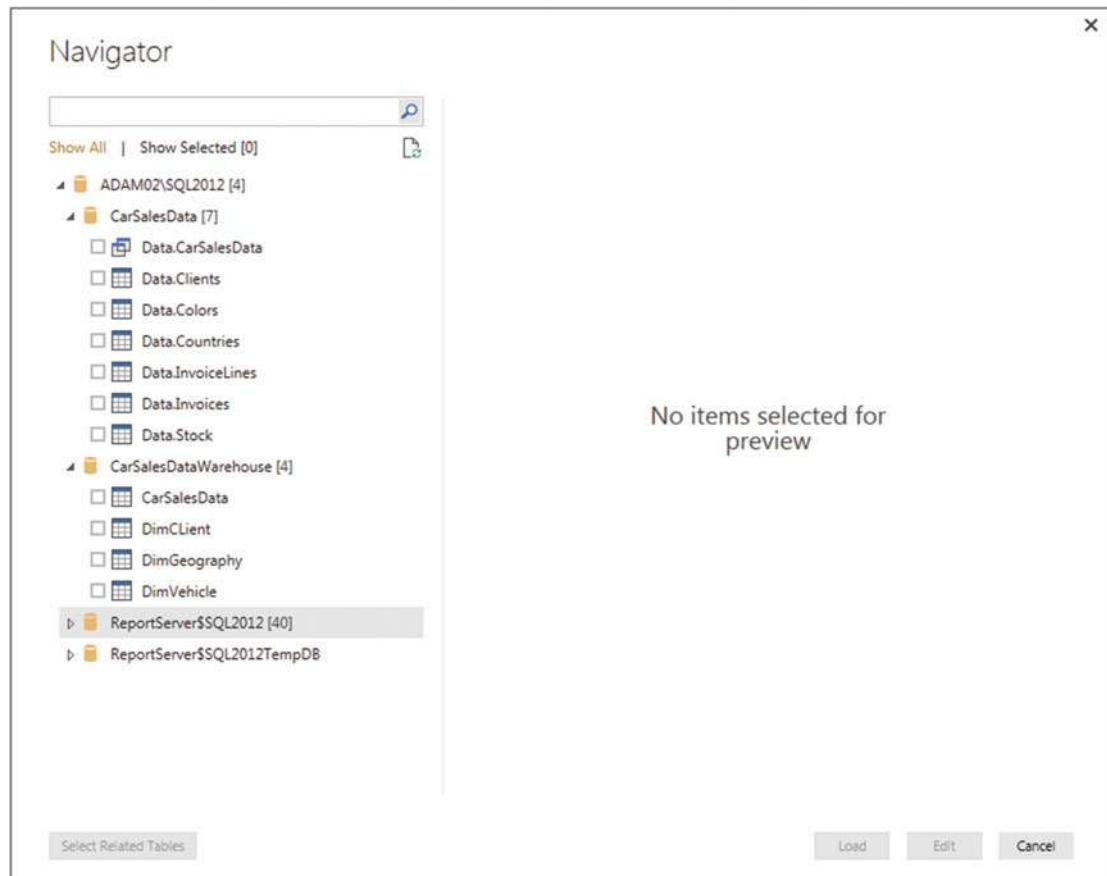


Figure 2-13. The Navigator dialog when selecting databases

You can see from Figure 2-11 that if you click the small triangle to the left of a database, then you are able to see all the tables and views that are accessible to you in this database. Although this can mean an overabundance of possible choices when looking for the table(s) or view(s) that you want, it is nonetheless a convenient way of reminding you of the name of dataset that you require.

Tip If you are swamped by the sheer number of tables in the Navigator dialog, then you can always click “Show selected” above the list of tables. This displays only selected tables in the Navigator window. Inversely, clicking Show All reverts to the display of all the available tables in the database.

Searching for Databases, Tables, and Views in Navigator

If you are overwhelmed by the sheer volume of table(s) and view(s) that appear in the left panel of the Navigator dialog, then you can use Navigator’s built-in search facility to help you to narrow down the set of potential data sources. To do this, do the following:

1. Carry out steps 1 and 2 in the earlier “Relational Databases: SQL Server” section to connect to a SQL Server instance *without* specifying a database.
2. Enter a few characters that you know are contained in the name of the table or view that you are looking for in the empty field on the top left of the Navigator dialog. This is called the *search box*. Entering, for example “cli” (without the quotes) on my server gives the result that you see in Figure 2-14.

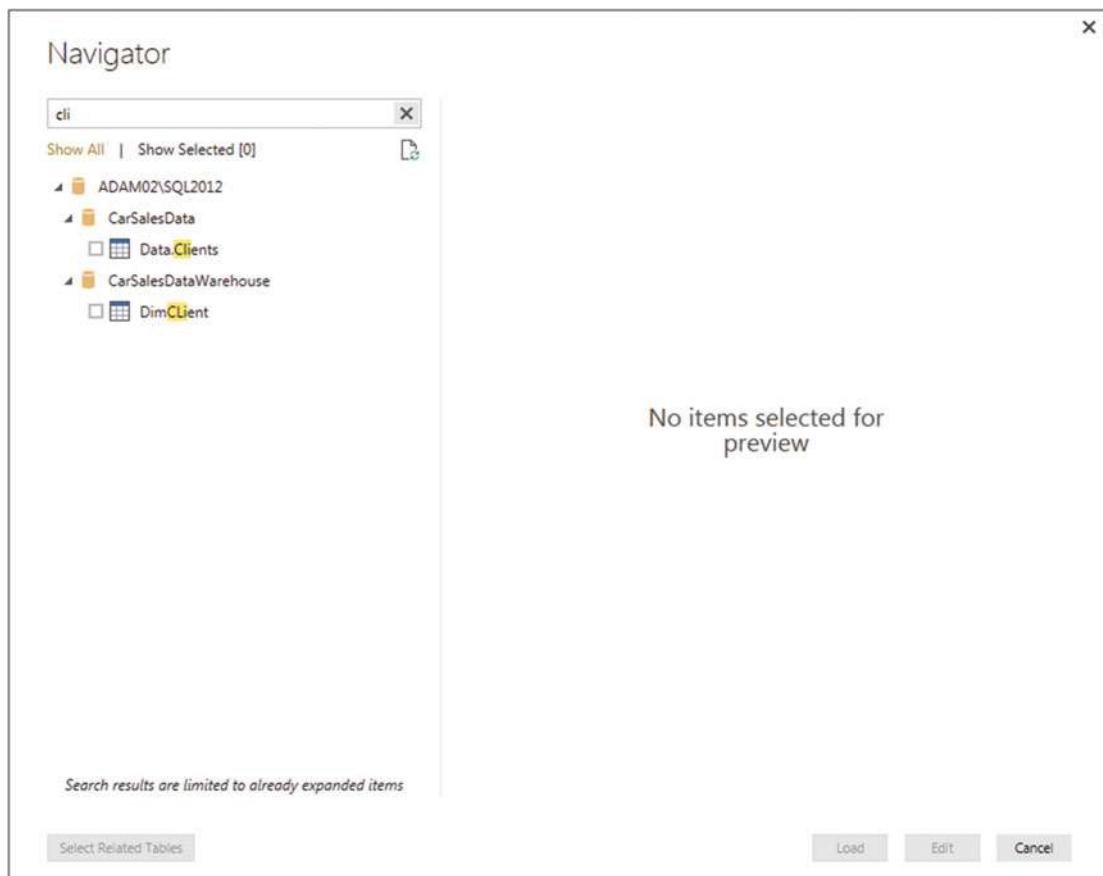


Figure 2-14. Using Search with Navigator

You can enter the text in uppercase or lowercase, and the text can appear anywhere in the names of the tables or views—not just at the start of the name. With every character that you type, the list of potential matches get shorter and shorter. Once you have found the dataset that you are looking for, simply proceed as described earlier to load the data into Power BI Desktop.

If your search does not return the subset of tables in any views that you were expecting, all you have to do is to click the cross at the right of the search box. This cancels the search and displays all the available tables, as well as clears the search box.

If you are not convinced that you are seeing all the tables and views that are in the database, then click the small icon at the bottom right of the Search box (it looks like a small page with two green circular arrows). This is the Refresh button, which refreshes the connection to the database and displays all the tables and views that you have permission to see.

Note Navigator search also looks for database names.

Database Security

Remember that databases are designed to be extremely secure. Consequently, you only see servers, databases, tables, and views if you are authorized to access them. You might have to talk to your IT department to ensure that you have the required permissions; otherwise, the table that you are looking for could be in the database, but remain invisible to you.

Tip If you experience a connection error when first attempting to connect to SQL Server, simply click the Edit button to return to the Microsoft SQL Database dialog and correct any mistakes. This avoids having to start over.

Using an SQL Statement

If there is a downside to using a relational database such as SQL Server as a data source, it is that the sheer amount of data that the database stores—even in a single table—can be dauntingly huge. Fortunately, all the resources of SQL Server can be used to filter the data that is used by Power BI Desktop before you even load the data. This way, you do not have to load entire tables of data at the risk of drowning in information before you have even started to analyze it.

The following are SQL Server techniques that you can use to extend the partnership between SQL Server and Power BI Desktop:

- SQL SELECT statements
- Stored procedures
- Table-valued functions

These are, admittedly, fairly technical solutions. Indeed, if you are not a database specialist, you could well require the services of your IT department to use these options to access data in the server. Nonetheless, it is worth taking a quick look at these techniques in case they are useful one day.

Any of these options can be applied from the SQL Server Database dialog. Here is an example of how to filter data from a database table using a SELECT statement.

1. In the Power BI Desktop ribbon, click the small triangle at the bottom of the Get Data button and then click SQL Server. The SQL Server Database dialog will appear.
2. Enter the server name and the database.
3. Click the triangle to the left of SQL Statement (optional). This opens a box where you can enter an SQL command.
4. Enter the SQL Command that you want to apply. In this case, it is `SELECT ClientName, Town, Region FROM Data.Clients ORDER BY ClientID.` The dialog will look like Figure 2-15.

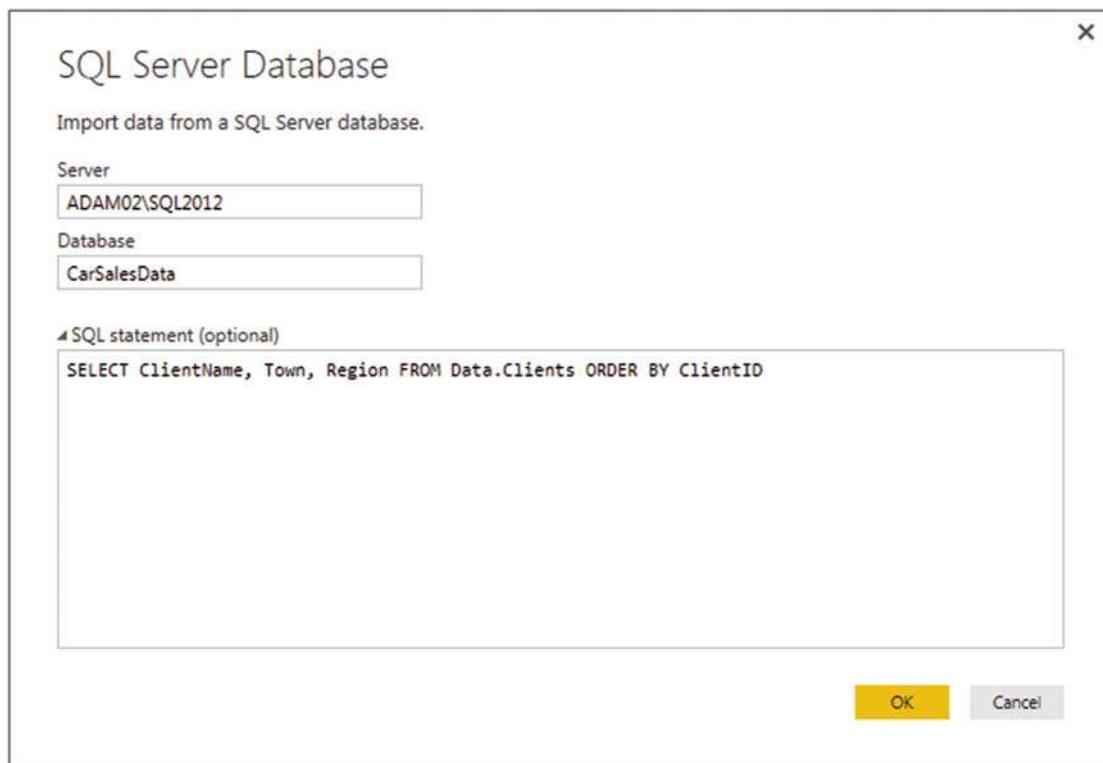


Figure 2-15. Using SQL to select database data

5. Click OK. A sample of the corresponding data is displayed in a dialog like the one shown in Figure 2-16.

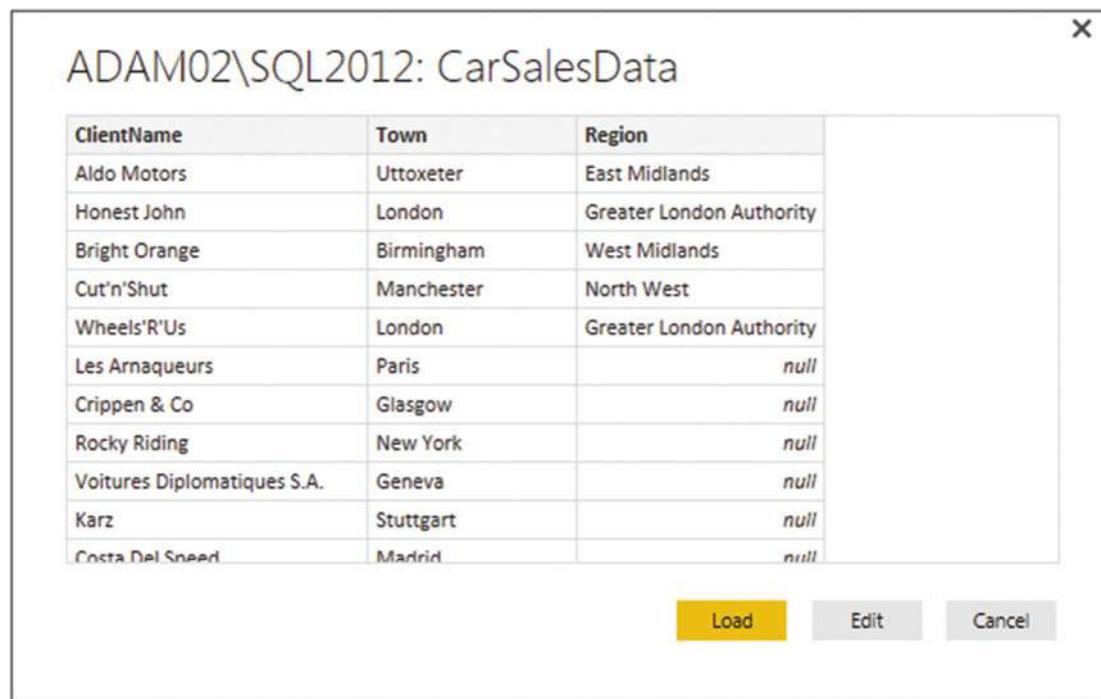


Figure 2-16. Database data selected using the SQL Statement option

6. Click Load or Edit to continue with the data load process. Alternatively, you can click Cancel and start a different data load.

The same principles apply when using stored procedures or functions to return data from SQL Server. You will always use the SQL Statement option to enter the command that will return the data. Unfortunately, the world of relational databases is huge; much, much more could be said at this point. I am afraid that I simply do not have the space to devote to all the subtleties of how you can use the available relational database sources, however. Just remember that to call a SQL Server stored procedure or function, you would enter the following elements into the Microsoft SQL Database dialog:

- Server: <your server name>
- Database: <the database name>
- SQL Statement: EXEC <enter the schema (if there is one, followed by a period) and the stored procedure name, followed by any parameters>

This way, either you or your IT department can create complex and secure ways to allow data from the corporate databases to be read into Power BI Desktop from enterprise databases.

Microsoft SQL Server Analysis Services Data Sources

An Analysis Services database is a data warehouse technology that can contain vast amounts of data that has been optimized to enable decision making. *SSAS cubes* (as these databases are also called) are composed of facts (measures or values) and dimensions (descriptive attributes). If your enterprise uses Analysis Services databases, you can access them by doing the following steps.

1. In the Power BI Desktop ribbon, click Get Data ▶ More ▶ Database and select SQL Server Analysis Services Database in the Get Data dialog.
2. Click Connect. The SQL Server Analysis Services Database dialog will appear.
3. Enter the Analysis Services server name and the database (or “cube”) name, if you know it. If you are using the sample data from the Apress site for this book, the database is CarSalesOLAP; otherwise, you have to specify your own SSAS server name.
4. Click “Select items and get data from the Multidimensional or Tabular model”. The dialog will look like Figure 2-17.

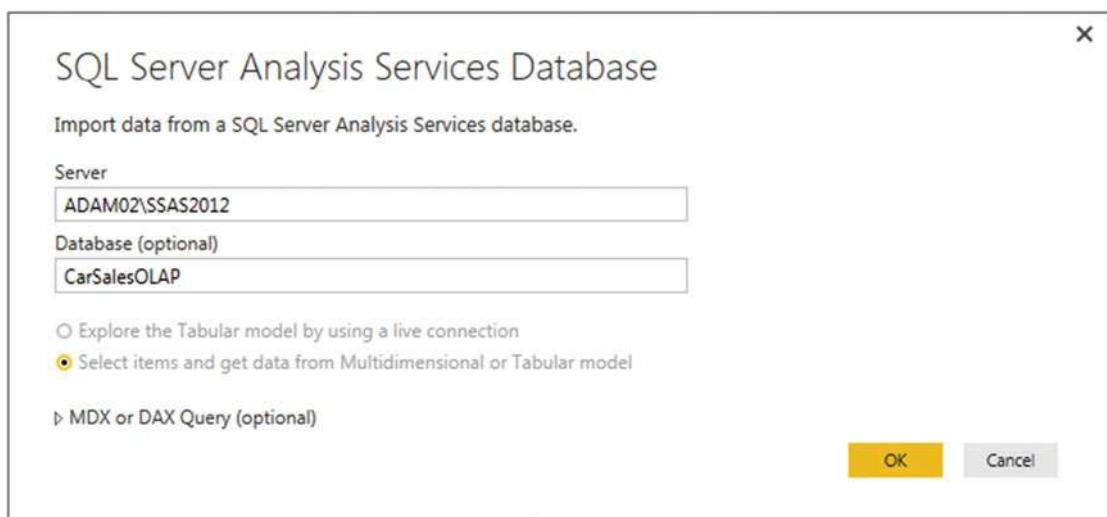


Figure 2-17. Connecting to an SSAS (multidimensional) database

5. Click OK. If this is the first time that you are connecting to the cube, then the Access SQL Server Analysis Service dialog will appear so that you can define the credentials that you are using to connect to the Analysis Services database, as shown in Figure 2-18.

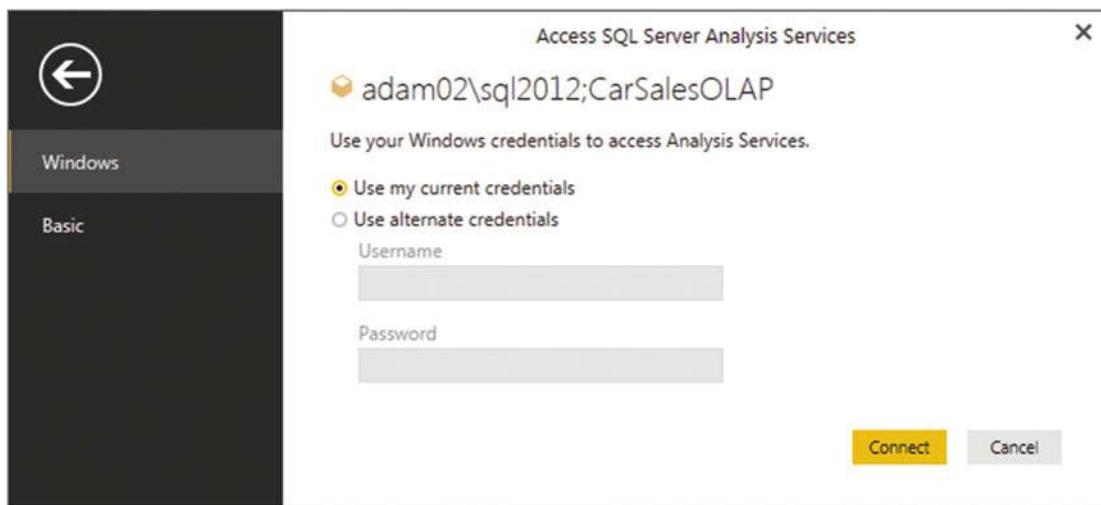


Figure 2-18. SQL Server Analysis Services Credentials dialog

6. Accept or alter the credentials and click Connect. The Navigator dialog will appear.
7. Expand the folders on the left pane of the dialog. This way, you can see all the fact tables and dimensions contained in the data warehouse.
8. Select the fact tables, dimensions, or even only the dimension elements and measures that you want to load. The dialog will look something like Figure 2-19.

The screenshot shows the Power BI Desktop Navigator window. On the left, there's a tree view of the cube structure under 'ADAM02\SSAS2012: CarSalesOLAP [1]'. The selected path is 'Car Sales Data Warehouse [1] > Car Sales Data [7] > Sale Price'. To the right is a preview of the 'Car Sales Data Warehouse' table with columns: Dim Client.Client Name, Dim Geography.Country Name, Dim Vehicle.Color, Dim Vehicle.Make, and Sale Price. The table data includes rows for Aldo Motors vehicles like Jaguar, Rolls Royce, Aston Martin, Bentley, and MGB across various colors and countries. At the bottom right are 'Load', 'Edit', and 'Cancel' buttons.

Dim Client.Client Name	Dim Geography.Country Name	Dim Vehicle.Color	Dim Vehicle.Make	Sale Price
Aldo Motors	United Kingdom	Black	Jaguar	44000
Aldo Motors	United Kingdom	Black	Rolls Royce	102500
Aldo Motors	United Kingdom	Blue	Aston Martin	392750
Aldo Motors	United Kingdom	British Racing Green	Bentley	110000
Aldo Motors	United Kingdom	British Racing Green	Jaguar	46750
Aldo Motors	United Kingdom	British Racing Green	MGB	22500
Aldo Motors	United Kingdom	Canary Yellow	Bentley	426250
Aldo Motors	United Kingdom	Dark Purple	Bentley	149500
Aldo Motors	United Kingdom	Dark Purple	Jaguar	81750
Aldo Motors	United Kingdom	Green	Aston Martin	251750
Aldo Motors	United Kingdom	Green	MGB	25250
Aldo Motors	United Kingdom	Night Blue	Aston Martin	68500
Aldo Motors	United Kingdom	Night Blue	Bentley	46750
Aldo Motors	United Kingdom	Night Blue	MGB	28000
Aldo Motors	United Kingdom	Red	Bentley	42250
Aldo Motors	United Kingdom	Red	Rolls Royce	499000
Aldo Motors	United Kingdom	Silver	Aston Martin	334000
Aldo Motors	United Kingdom	Silver	Jaguar	81750
Aldo Motors	United Kingdom	Silver	MGB	25250
Aldo Motors	United Kingdom	Silver	Rolls Royce	97750
Ambassador Cars	USA	Black	Jaguar	42250
Ambassador Cars	USA	Blue	Aston Martin	132750
Ambassador Cars	USA	British Racing Green	Jaguar	47750

Figure 2-19. Selecting attributes and measures from an SSAS cube

- Click Load. The Power BI Desktop window will open and display the measures and attributes that you selected in the Fields list in the Report window.

SSAS cubes are potentially huge. They can contain dozens of dimensions, many fact tables, and literally hundreds of measures and attributes. Understanding multidimensional cubes and how they work is beyond the scope of this book. Nonetheless, it is important to understand that for Power BI Desktop, a cube is just another data source. This means that you can be extremely selective as to the cube elements that you load into Power BI Desktop, and only load the elements that you need for your analysis. You can load entire dimensions or just a few attributes, just like you can load whole fact tables or just a selection of measures.

Note You can filter the data that is loaded from an SSAS cube by expanding the MDX or DAX query (optional) item in the SQL Server Analysis Services Database dialog. Then you can enter an MDX query in the box that appears before clicking OK. Be warned that SSAS cubes use queries written in MDX—a specialist language that is considered not always easy to learn. The good news is that if an Analysis Services expert has set up a cube correctly you can see SSAS display folders in PowerBI Desktop Query.

Analysis Services Cube Tools

Analysis Services data sources allow you to tweak the selection of source elements in a way that is not available with other data sources. Essentially, you have two extra options:

- Add Items
- Collapse Columns

Add Items

When using an SSAS data source, you can at any time add any attributes or measures that you either forgot or thought that you would not need when setting up the initial connection.

1. In the Query pane, click the SSAS query that you have previously established. (The sample data for this connection is displayed in the center of the Query window.) The Manage ribbon will appear, as shown in Figure 2-20.

The screenshot shows the Power BI Desktop interface with the 'Query Editor' open. The ribbon at the top has the 'Cube Tools' tab selected. On the left, there's a toolbar with icons for 'Add Items' and 'Collapse Columns'. The main area contains a table with 22 rows of data, representing sales records. The columns are labeled: Dim Client.Client Name, Dim Geography.Country Name, Dim Vehicle.Color, Dim Vehicle.Make, and Sale Price. The data includes various car models like Jaguar, Rolls Royce, and Bentley across different colors and countries. To the right of the table is a 'Query Settings' pane. Under the 'PROPERTIES' section, the 'Name' is set to 'Car Sales Data Warehouse'. Under 'APPLIED STEPS', there are three steps listed: 'Source', 'Navigation', and 'Added Items', with 'Added Items' being the current selection. The bottom right corner of the interface shows the text 'PREVIEW DOWNLOADED AT 21:09'.

Figure 2-20. Cube Tools

2. In the Manage ribbon, click the Add Items button. The Add Items dialog will appear, as shown in Figure 2-21.

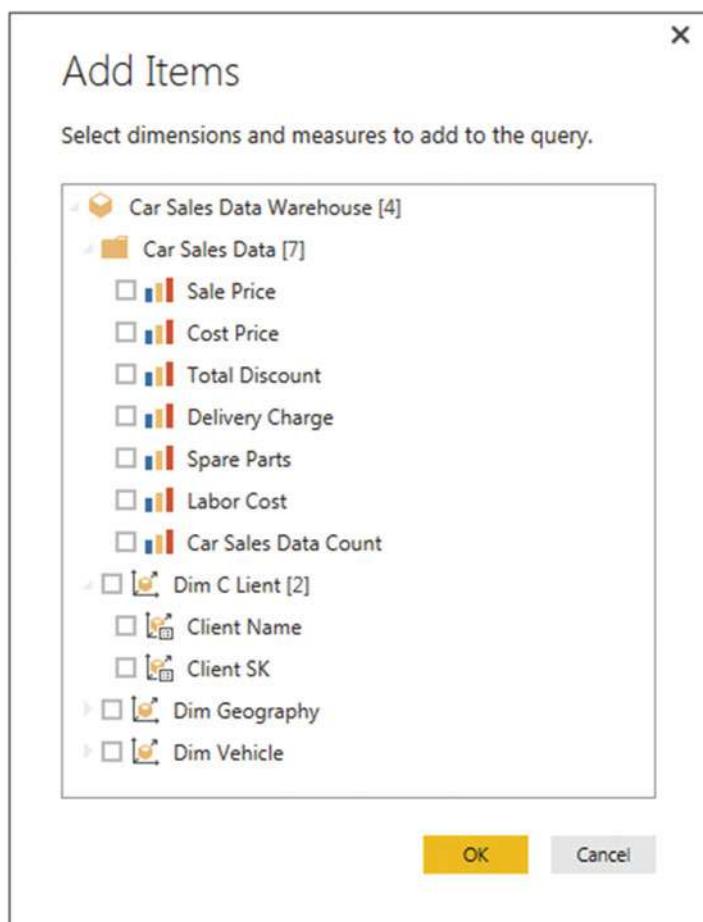


Figure 2-21. The Add Items dialog

3. Expand any measure groups and select all the measures and attributes that you want to add.
4. Click OK.

The selected measures and attributes are added as new columns at the right of the dataset.

Note Power BI Desktop Query will not detect if any new measures and attributes that you add are already in the dataset. So if you add an element twice, it will appear *twice* in the query.

Collapse Columns

Do the following to remove any columns that you no longer require from the data source (which can accelerate data refresh).

1. In the Query pane, click the SSAS query that you have previously established. (The sample data for this connection will be displayed in the center of the Query window.) The Manage ribbon will appear.
2. In the Manage ribbon, click the Collapse Columns button.

The columns are removed from the connection to the SSAS cube and, consequently, from the Fields list at the right of the Power BI Desktop window. They are also removed from any visualizations that use them.

Note Removing columns from Power BI Desktop Query can have a serious domino effect on reports and dashboards. Consequently, you need to be very careful when removing them.

Microsoft SQL Server Analysis Services Tabular Data Sources

A SQL Server Analysis Services tabular database is another technology that is used for data warehousing. It is different from the more traditional dimensional data warehouse in that it is entirely stored in the server memory, and consequently, is usually very much faster to use.

Connecting to a tabular data warehouse is identical to connecting to a traditional SSAS cube, so I will not explain it here. All you have to do is follow the steps that you used in the previous section to connect to a SQL Server Analysis Services data source, only you enter the tabular database server and database. If you want to filter the data that is returned, then you can use the DAX language to return only a subset of the available data.

There is, however, one tabular option that is available over and above the “classic” connection to a data source. This is the possibility of connecting *directly* to the tabular source using a live connection. This option is different for the following reasons:

- You do not load the data into Power BI Desktop. Instead, you use the data directly from the tabular database server.
- You are consequently not loading a copy of the data into Power BI Desktop. This means that you *cannot* work offline. You need to be able to connect to the tabular database to use the data.
- The connection to the tabular database—and the consequent availability of the data for analysis—is usually extremely fast, if not instantaneous.
- It follows that you *not* need to refresh the data source if ever the data is updated in the tabular database. The data that is available in Power BI Desktop is always the latest version of the data.
- You will have all the data that is in the tabular database available.
- You cannot filter the source data so that only a subset of the database is visible in Power BI Desktop.

- You cannot connect to any other data source if you are using a live (or direct) connection to an SSAS tabular database. So this data source has to contain all the data that you need for your analysis.
- You cannot transform or mash up the data in any way using a live connection to an SSAS tabular database. Consequently, none of the possibilities concerning data transformation that are explained in the next three chapters is available.

To see this in action, you need to set up a direct connection to a tabular database. This is how it can be done:

1. In the Power BI Desktop ribbon, click Get Data ▶ More ▶ Database and select SQL Server Analysis Services Database in the Get Data dialog.
2. Click Connect. The SQL Server Analysis Services Database dialog will appear.
3. Enter the Analysis Services server name and the tabular database name, if you know it. If you are using the sample data from the Apress site for this book, the database is CarSalesOLAP.
4. Select “Explore the Tabular model by using a live connection”.
5. Click OK. The Navigator dialog will appear.
6. Expand the folders on the left and select Model, as shown in Figure 2-22.

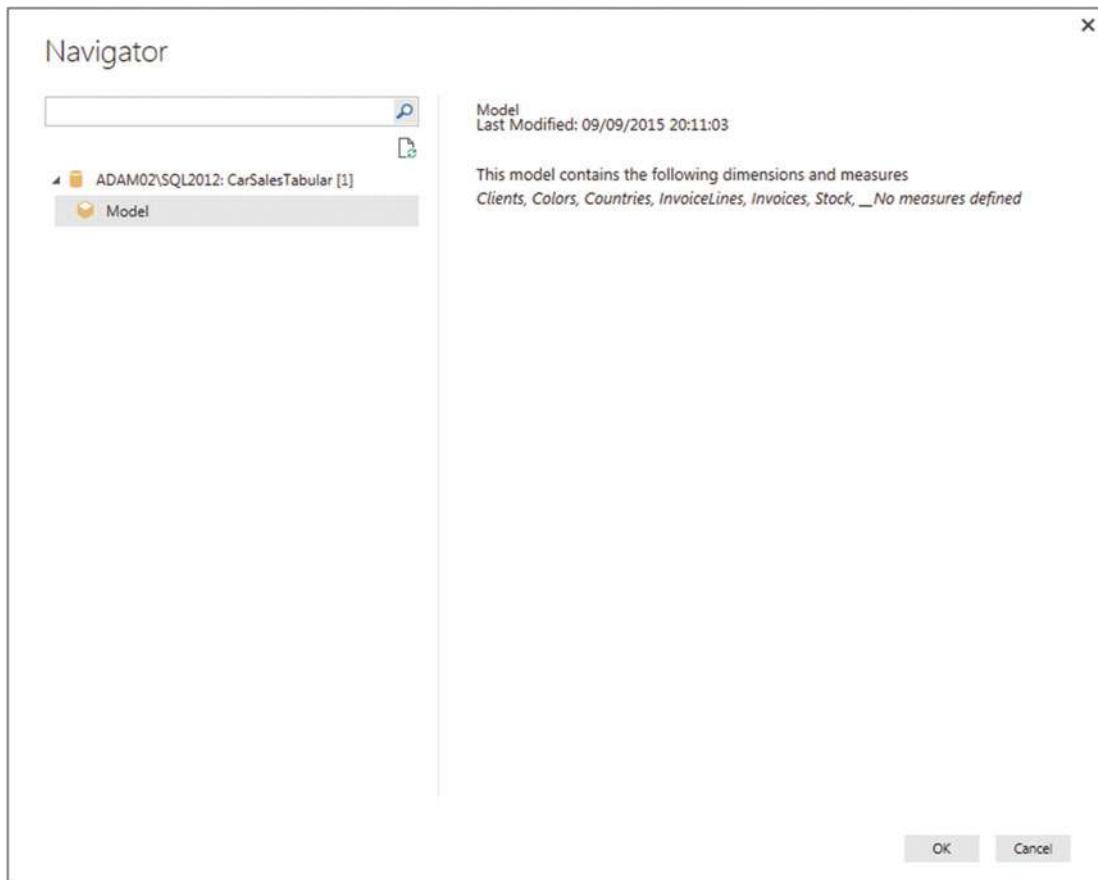


Figure 2-22. Establishing a live connection to a tabular data source

7. Click OK. The Power BI Desktop window opens and displays all the tables that are in the tabular model.

Note When you set up a live connection to a tabular data source, you *cannot* filter the data that is loaded from an SSAS cube by expanding the MDX or DAX query (optional) item in the SQL Server Analysis Services database dialog and entering a DAX query.

It is worth noting that there is no real Power BI Desktop query for a live connection to an Analysis Services tabular data warehouse. If you click the Edit Queries button, all you will see is the connection dialog for the Analysis Services database.

Other Data Sources

So far in this chapter, you have seen how to load data from a handful of the more frequently used available sources. The good news is that Power BI Desktop can read data from dozens more sources. The bad news is that it would take a whole book to go into all of them in detail.

So I will not be describing any other data sources in this book. This is because now that you have come to appreciate the core techniques that make up the extremely standardized approach that Power BI Desktop takes to loading data, you can probably load any possible data type without needing much more information from me. It is worth noting that you can also connect to Oracle and Teradata databases using DirectQuery mode. While on the subject of Direct Query, remember that you can change the data types of source columns and add calculated columns in Direct Query mode - whatever the data source.

Should you need any specific information on other data sources, then your best port of call is the Microsoft Power BI web site. This is currently at <http://support.powerbi.com>.

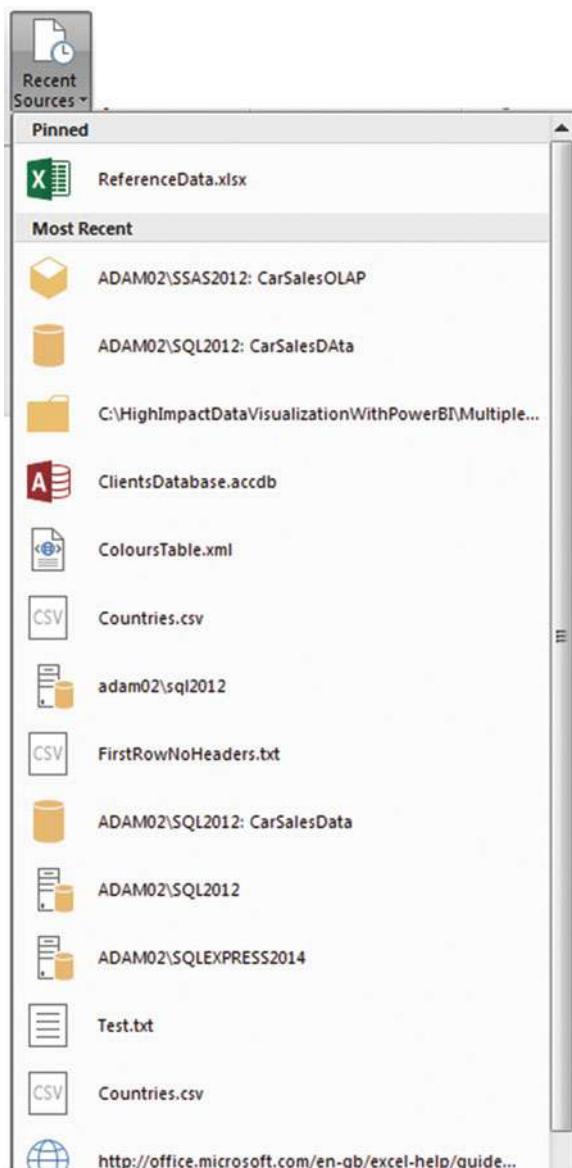
Reusing Recent Data Sources

Given the myriad of data sources that Power BI Desktop can connect to, it is easy to lose track of which dataset is contained in what database. This could make returning to a previous connection a matter of trial and error. Fortunately, the Power BI Desktop team has sought to make your life easier. Power BI Desktop remembers the last couple of dozen connections that you have made, whatever the data source, and allows you to reconnect to a data source from any Power BI Desktop application.

Reusing a Data Source

Do the following to reuse a data source that you have connected to recently.

1. Click the Recent Sources button in the Power BI Desktop Home ribbon. A menu of recently used data sources will appear. On my laptop, the list looks like the one shown in Figure 2-23.

**Figure 2-23.** The Recent Sources list

2. Click the data source that you want to use again. The Navigator dialog for that particular data source will be displayed.

From here, you can continue to select the data that you want from the data source, just as if you had reestablished a connection. As you can see, Power BI Desktop has memorized any security information and will not ask you to reenter passwords or choose a security method. It is worth noting that it is only the data source that is memorized by Power BI Desktop. The application does not remember the selected tables, views, spreadsheets, named ranges, fact tables, or dimensions that you used previously. Consequently, you have to reselect these.

If you cannot see the connection that you were hoping for in the menu of recent sources, then do not despair. If you scroll to the bottom of the list of recent sources, you will find a More option. Clicking this displays the Recent Sources dialog, which contains a much bigger list of recently used data sources. An example of this is given in Figure 2-24.

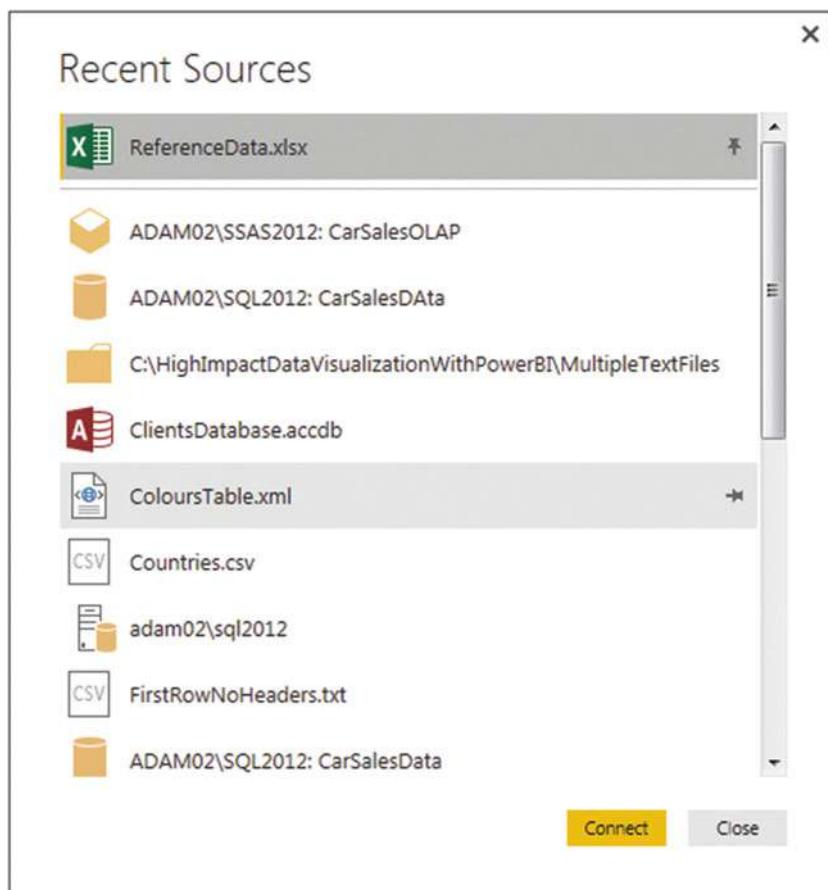


Figure 2-24. The Recent Sources dialog

Pinning a Data Source

If you look closely at Figures 2-23 and 2-24, you see that the Excel file ReferenceData.xlsx is pinned to the top of both the Recent Sources menu and the Recent Sources dialog. This allows you to make sure that certain data sources are always kept on hand and ready to reuse.

Do the following to pin a data source that you have recently used to the menu and dialog of recent sources.

1. Click the Recent Sources button in the Power BI Desktop Home ribbon.
2. Scroll down to the bottom of the menu and click More. The Recent Sources dialog will appear.

3. Hover the mouse over a recently used data source. A pin icon will appear at the right of the data source name.
4. Click the pin icon. The data source is pinned to the top of both the Recent Sources menu and the Recent Sources dialog. A small pin icon remains visible at the right of the data source name.

Note To unpin a data source from the Recent Sources menu and the Recent Sources dialog, all you have to do is click the pin icon for a pinned data source. This unpins it and it reappears in the list of recently used data sources.

Refreshing Data

Source data can change. Indeed, the very definition of data implies continuous modification. It follows that you will need to update the preview data that Power BI Desktop Query uses so that you can see the latest version of the information that you are querying.

Refreshing a Query

Refreshing the data preview is as simple as doing the following steps.

1. Click the query that you want to refresh in the Queries pane on the left of the Power BI Desktop Query editor.
2. In the Home ribbon, click the Refresh Preview button.

Power BI Desktop Query connects to the source data and displays the latest data for this query.

Refreshing All the Queries in a Power BI Desktop File

If you have several queries that you are using in a single Power BI Desktop file, then you can refresh them all at once like this:

1. In the Home ribbon, click the small down-facing triangle at the bottom right of the Refresh Preview button.
2. Select Refresh All from the menu.

All the queries in the Power BI Desktop file are refreshed and display the latest data.

Old Data

When you open a Power BI Desktop file that you have not used in some time, you could well see a message above the data pane like the one shown in Figure 2-25.



Figure 2-25. The Refresh data warning

Simply click the Refresh button (if that is what you want to do). The preview data is refreshed and the latest data is displayed.

Connection Security

Power BI Desktop is not just about creating awesome dashboards. It is also about speed. One way that it can help you to work faster is by remembering the security information that you entered when connecting to secure data sources such as databases and data warehouses.

Secure credentials can evolve, and consequently, you may need to change your login or password occasionally. You could even want to remove security information from Power BI Desktop from time to time. Here is how you can do this:

1. In the Power BI Desktop System menu, select Options and Settings ► Data Source Settings. The Data Source Settings dialog will appear, as shown in Figure 2-26.

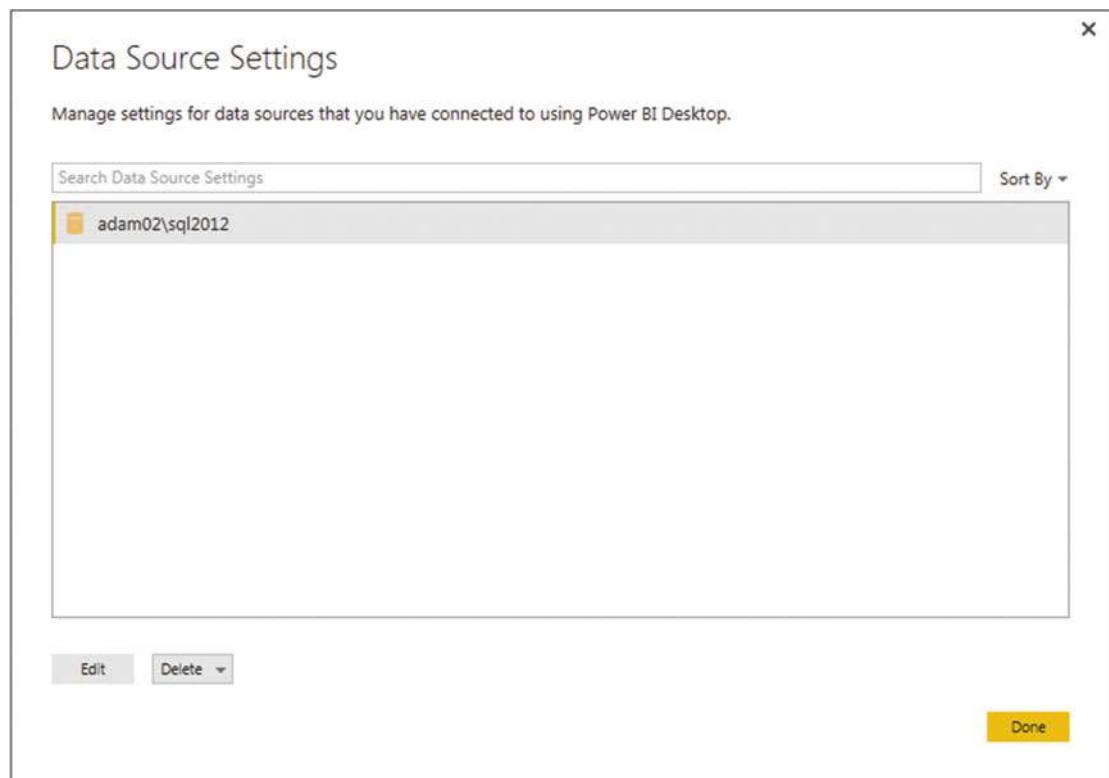


Figure 2-26. The Data Source Settings dialog

2. Select a data source that you want to modify and click Edit. The Edit Data Source Settings dialog will appear, as shown in Figure 2-27.

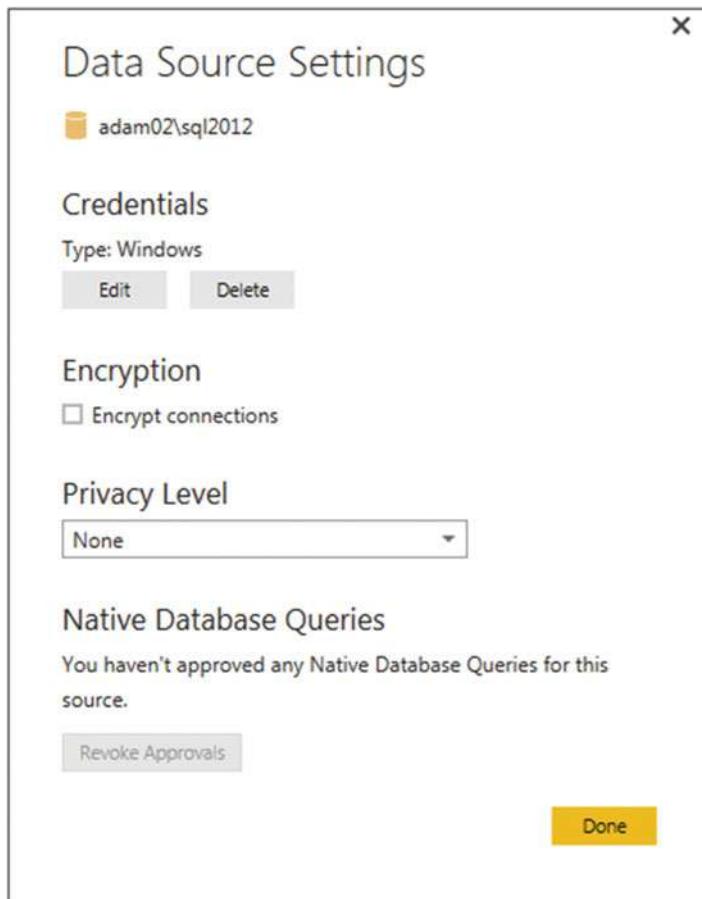


Figure 2-27. The Edit Data Source Settings dialog

3. Click Edit to display the credentials dialog (as shown in Figure 2-16). You can modify your login and password here.

The Edit Data Source Settings dialog also lets you

- Delete (revoke, if you prefer) the existing credentials
- Force encryption of the data over the connection
- Define a privacy level
- Revoke approval for native database queries

The Data Source Settings dialog also lets you delete the data source settings for any selected data source—or all of them.

Conclusion

In this chapter, you have seen how the latest addition to the Microsoft business intelligence toolset, Power BI Desktop, can help you find and load data from a variety of sources. These sources can be more traditional—such as Access, Excel, or text files—or they could come from corporate databases, or data warehouses, or even big data repositories or social media sources. Indeed, data could even be found in public data repositories or from commercial cloud-based sources. If you are already a statistics guru then you can connect to R scripts and run them. Power BI Desktop can even help you find available data (from both inside and outside the enterprise) and remember any recent searches that you have made. Yet this is only the first part of the story. Now you need to learn how to shape and tweak the data to prepare it for further use. This is the subject of the next three chapters.

CHAPTER 3



Transforming Datasets

In the previous chapter, we saw some of the ways in which you can find and load data into Power BI Desktop Query, and from there into the Power BI Desktop data model. Inevitably, this is the first part of any process that you create to extract, transform, and load data. Yet it is quite definitely only a first step. Once the data is in Power BI Desktop, you need to know how to adapt it to suit your requirements in a multitude of ways. This is because not all data is ready to be used immediately. Quite often, you have to do some initial work on the data to make it more easily useable in Power BI Desktop. Tweaking source data is generally referred to as *data transformation*, which is the subject of this chapter.

The transformations that you can apply to the raw data are many and varied. Given the vast range of ways that you can chop, change, and tweak your data, we will concentrate on the following transformations in this chapter:

- Renaming, removing, and reordering columns
- Removing groups or sets of rows
- Deduplicating datasets
- Excluding records by filtering the data
- Sorting the data

These transformations, or modifications, if you prefer another term, are potentially extensive and varied. Learning to apply the techniques that Power BI Desktop makes available enables you to take data as you find it and push it as a series of coherent and structured data tables into the Power BI Desktop data model so that it is ready to be used to create Power BI Desktop dashboards.

As it is all too easy to be overwhelmed (at least initially) by the extent of the data transformation options that Power BI Desktop has to offer, I have grouped the possible modifications into four categories. These are my own, and are merely a suggestion to facilitate understanding.

These categories are:

- *Transforming datasets.* This includes adding and removing columns and rows, renaming columns, as well as filtering data.
- *Data modification.* This covers altering the actual data in the rows and columns of a dataset.
- *Extending datasets.* By this, I mean adding further columns, possibly expanding existing columns into more columns or rows, and adding calculations.
- *Joining datasets.* This involves combining multiple separate datasets—possibly from different data sources—into a single dataset.

This chapter covers the first option in this list and introduces you to Power BI Desktop Query and how you can use it to chop and change source data. In the next chapter, you learn how to cleanse and modify data. In Chapter 5, you learn how to subset columns to extract part of the available data in a column, calculate columns, merge data from separate queries, and add further columns containing different types of calculations, and you learn about pivoting and unpivoting data. So, if you cannot find what you are looking for in this chapter, there is a good chance that the answer is in the next one or the one after that.

In this chapter, I will also use a set of example files that you can find on the Apress web site. If you have followed the instructions in Appendix A, then these files are in the C:\PowerBiDesktopSamples folder.

Power BI Desktop Queries

In the Chapter 1, you saw how to load source data directly into Power BI Desktop and use it immediately to create dashboards. Clearly, this approach presumes that the data that you are using is perfectly structured, clean, and error-free. Source data is nearly always correct and ready to use in dashboards when it comes from “corporate” data sources such as data warehouses (held in relational, dimensional, or tabular databases). This is not always the case when you are faced with multiple disparate sources of data that have not been precleansed and prepared by an IT department. The everyday reality is that you could have to cleanse and transform much of the source data that you will use for your Power BI Desktop dashboards.

The really good news is that the kind of data transformation that used to require expensive servers and industrial-strength software is now available for free. Yes, Power BI Desktop comes with an awesome ETL (Extract, Transform, and Load) tool that can rival many applications that cost hundreds of thousands of dollars.

Power BI Desktop data transformation is carried out using *queries*. As you saw in Chapter 1, you do not have to modify source data. You can load it directly if it is ready for use. Yet if you need to cleanse the data, you add an intermediate step between connecting the data and loading it into the Power BI Desktop data model. This intermediate step uses the Power BI Desktop Query Editor to tweak the source data.

So how do you apply queries to transform your data? You have two choices:

- Load the data first from one or more sources, and then transform it later.
- Edit each source data element in a query before loading it.

Power BI Desktop is extremely forgiving. It does not force you to select one or the other method and then lock you in to the consequences of your decision. You can load data first and then realize that it needs some adjustment, switch to the query editor and make changes, and then return to creating your dashboard. Or you can first focus on the data and try to get it as polished and perfect as possible before you start building reports. The choice is entirely up to you.

To make this point, let’s take a look at both of these ways of working.

Note At risk of being pedantic and old-fashioned, I would advise you to make notes when creating really complex transformations, because going back to a solution and trying to make adjustments later can be painful when they are not documented at all.

Editing Data After a Data Load

In Chapter 1, you saw how to load the Excel workbook CarSales.xlsx directly into the Power BI Desktop data model and use it to create a starter dashboard. Now let’s presume that you want to make some changes to the data structure of the data that you have already loaded. Specifically, you want to

rename the CostPrice column. Hopefully, you saved this as the CarSalesFirstDashboard.pbix file in the C:\PowerBiDesktopSamples directory. If you did not, then it is available in the sample file set as CarSalesFirstDashboard_InCase.pbix.

1. Launch Power BI Desktop.
2. Open the sample file C:\PowerBiDesktopSamples\CarSalesFirstDashboard_InCase.pbix.
3. In the Power BI Desktop Home ribbon, click the Edit Queries button. The Power BI Desktop Query Editor will open and display the source data as a table. The window will look like Figure 3-1.

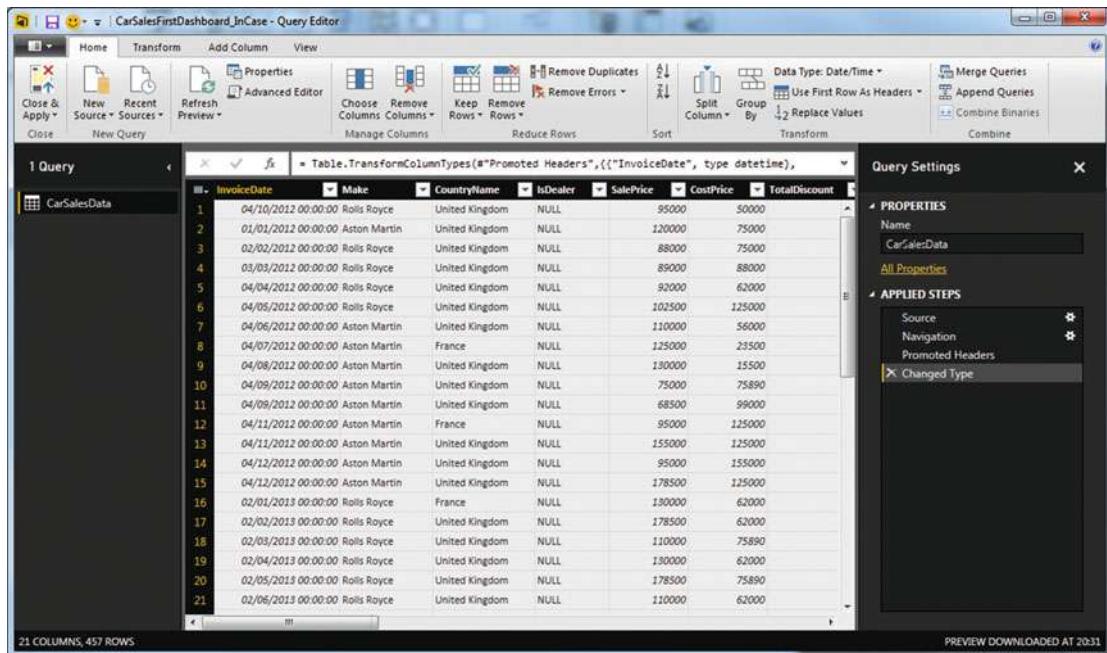


Figure 3-1. The Power BI Desktop Query Editor

4. Right-click the title of the CostPrice column. The column will be selected and the title will appear in yellow.
5. Select Rename from the context menu.
6. Type **VehicleCost** and press Enter. The column title will change to VehicleCost.
7. In the Power BI Desktop Query Editor Home ribbon, click the Close and Apply button. The Power BI Desktop Query Editor will close and return you to the Power BI Desktop window. VehicleCost has replaced CostPrice anywhere that it was used in the dashboard.

I hope that this simple example makes it clear that transforming the source data is a quick and painless process. The technique that you applied—renaming a column—is only one of many dozens of possible techniques that you can apply to transform your data. However, it is not the specific transformation that is

the core idea to take away here. What you need to remember is that the data that underpins your dashboard is always present and it is only a single click away. At any time, you can “flip” to the data and make changes, simply by clicking the Edit Queries button in the Power BI Desktop Window. Any changes that you make and confirm will update your dashboards and reports instantaneously. If you want you can edit the query behind any table that is visible in the Report or Data views simply by right-clicking in the table and selecting Edit Query from the context menu.

Transforming Data Before Loading

On some occasions, you might prefer to juggle with your data before you load it. This is a variation on the approach that you have used so far when creating dashboards. Do the following to transform your data before it appears in the Power BI Desktop Window:

1. In the Power BI Desktop Home ribbon, click the tiny triangle in the Get Data button.
2. Select Excel in the menu and Open the Excel file CarSalesFirstDashboardData.xlsx.
3. In the Navigator window, select the CarSalesData worksheet.
4. Click the Edit button (*not* the Load button). The Power BI Desktop Query Editor will open and display the source data as a table.
5. Carry out steps 4 through 6 from the previous example to rename the CostPrice column.
6. In the Power BI Desktop Query Editor Home ribbon, click the Close and Apply button. The Power BI Desktop Query Editor will close and return you to the Power BI Desktop window.

This time, you have made a simple modification to the data *before* loading the dataset into the Power BI Desktop data model. The data modification technique was exactly the same. The only difference between loading the data directly and taking a detour via the Query window was clicking Edit instead of Load in the Navigator dialog.

Query or Load?

Power BI Desktop always gives you the choice of loading data directly into its data model or taking a constructive detour via Power BI Desktop Query. The path that you follow is entirely up to you and clearly depends on each set of circumstances. Nonetheless, there are a few basic principles that you might want to consider when faced with a new dashboarding challenge using unfamiliar data.

- Are you convinced that the data is ready to use? That is, is it clean and well structured? If so, then you can try loading it directly into the Power BI Desktop data model.
- Are you faced with multiple data sources that need to be combined and molded into a coherent structure? If this is the case, then you really need to transform the data using Power BI Desktop Query.
- Does the data come from an enterprise data warehouse? This could be held in a relational database, a SQL Server Analysis Services cube, or even an in-memory tabular data warehouse. As these data sources are nearly always the result of many hundreds—or even thousands—of hours of work cleansing, preparing, and structuring the data, you can probably load these straight into the data model.

- Does the data need to be preaggregated and filtered? Think Power BI Desktop Query.
- Are you faced with lots of lookup tables that need to be added to a “core” data table? Then Power BI Desktop Query is your friend.
- Does the data contain many superfluous or erroneous elements? Then use Power BI Desktop Query to remove these as a first step.
- Does the data need to be rationalized and standardized to make it easier to handle? In this case, the path to success is via Power BI Desktop Query.

These kinds of questions are only rough guidelines. Yet they can help to point you in the right direction when you are working with Power BI Desktop. Inevitably, the more that you work with this application, the more you will develop the reflexes and intuition that will help you make the correct decisions. Remember, however, that Power BI Desktop is there to help, and that even a directly loaded dataset is based on a query. So you can always load data and then decide to tweak its structure later if you need to. Alternatively, editing data in a query window can be a great opportunity to take a closer look at your data before loading it into the data model—and it only adds a couple of clicks.

So feel free to adopt a way of working that you feel happy with. Power BI Desktop will adapt to your style easily and almost invisibly, letting you switch from data to dashboards so fluidly that it will likely become second nature.

The remainder of this chapter will take you through some of the techniques that you need to know to cleanse and shape your data. However, before getting into all the detail, let’s take a quick, high-level look at the Power BI Desktop Query Editor and the way that it is laid out.

The Power BI Desktop Query Editor

All of your data transformation will take place in the Power BI Desktop Query Editor. It is a separate window from the one where you create your dashboards and it has a slightly different layout.

The Power BI Desktop Query Editor consists of six main elements:

- The four ribbons: Home, Transform, Add Column, and View
- The Query list pane containing all the queries that have been added to a Power BI Desktop file
- The Data window, where you can see a sample of the data for a selected query
- The Query Settings pane that contains the list of steps used to transform data
- The formula bar above the data that shows the code (written in the Power BI “M” language) that performs the selected transformation step
- The status bar (at the bottom of the window) that indicates useful information, such as the number of rows and columns in a query table, and the date when the dataset was downloaded

The callouts for these elements are shown in Figure 3-2.

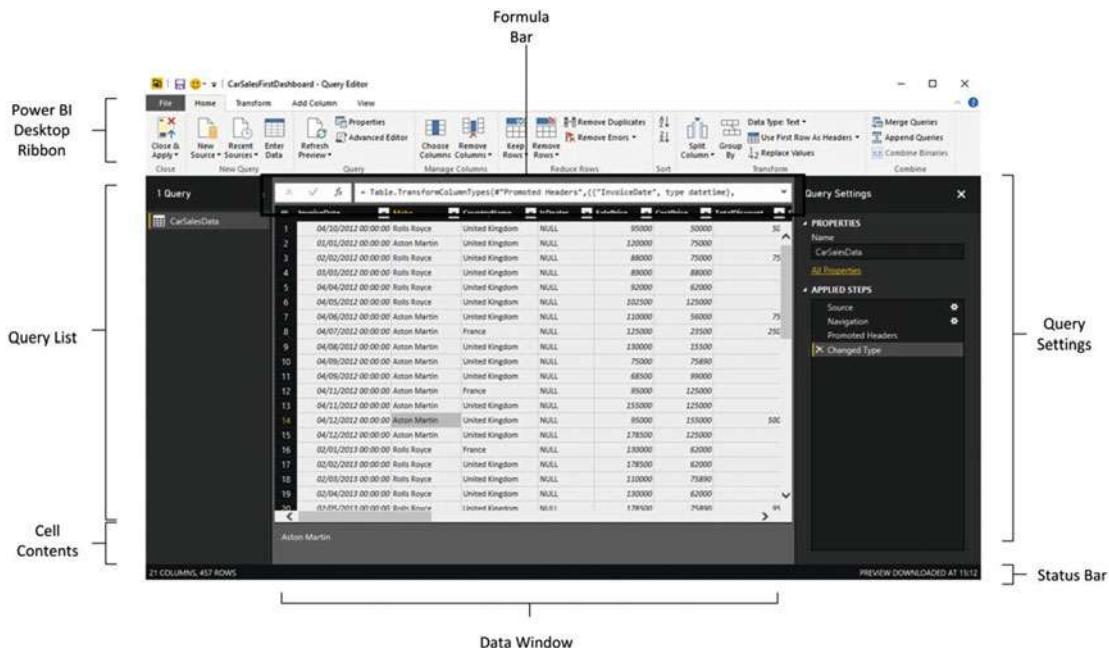


Figure 3-2. The Power BI Desktop Query Editor, explained

The Applied Steps List

Data transformation is by its very nature a sequential process. So the Query window stores each modification that you make when you are cleansing and shaping source data. The various elements that make up a data transformation process are listed in the Applied Steps list of the Query Settings pane in the Query Editor.

The Power BI Desktop Query Editor does not number the steps in a data transformation process, but it certainly remembers each one. They start at the top of the Applied Steps list (nearly always with the Source step) and can extend to dozens of individual steps that trace the evolution of your data until you load it into the data model. Indeed, as you click each step in the Applied Steps list, the data in the Data window changes to reflect the results of each transformation, giving you a complete and visible trail of all the modifications that you have applied to the dataset.

The Applied Steps list gives a distinct name to the step for each and every data modification option that you cover in this chapter and the next. As it can be important to understand exactly what each function actually achieves, I will always bring to your attention the standard name that Power BI Desktop Query applies.

There are just a couple of fundamental functions that you need to know when using the Applied Steps list, as explained next.

Renaming a Step

You can rename any step really simply:

1. Right-click the step that you want to rename.
2. Select Rename from the context menu.
3. Change the existing name.
4. Press Enter.

Deleting a Step

Simply click the cross to the left of the step name in the Applied Steps list.

Note Be aware that deleting a step can have serious consequences for any steps that follow the step that you delete. Indeed, Power BI Desktop Query warns you if removing a step will disrupt the processing sequence. If you need to alter a series of steps to change a sequence of data modifications, it is often easiest to delete steps from the bottom upward, in reverse order, and then rebuild a sequence of transformations from a stable starting point.

The Power BI Desktop Query Editor Ribbons

Power BI Desktop Query Editor uses (in the January 2016 version, at least) four ribbons. They are fundamental to what you learn in the course of this chapter. They are as follows:

- The Home ribbon
- The Transform ribbon
- The Add Column ribbon
- The View ribbon

I am not suggesting for a second that you need to memorize what all the buttons in these ribbons do. What I hope is that you are able to use these brief descriptions of the query ribbon buttons to get an idea of the amazing power of Power BI Desktop in the field of data transformation. So if you have an initial dataset that is not quite as you need it, you can take a look at the resources that Power BI Desktop has to offer and how they can help. Once you find the function that does what you are looking for, you can jump to the relevant section for the full details on how to apply it.

The Home Ribbon

Since we will be making intense use of the Power BI Desktop Home ribbon to transform data, it is important to have an idea of what it can do. I explain the various options in Figure 3-3 and in Table 3-1.

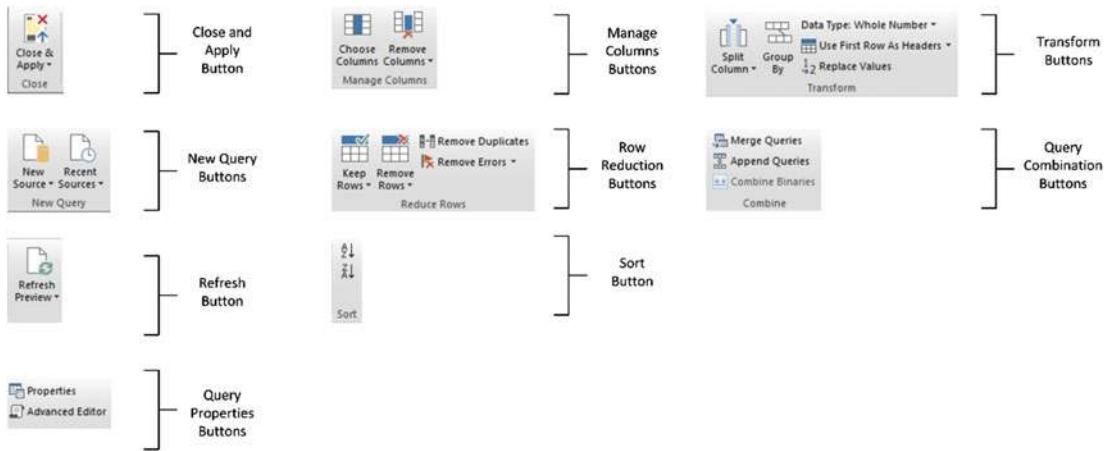


Figure 3-3. The Query Editor Home ribbon

Table 3-1. Query Editor Home Ribbon Options

Option	Description
Close & Apply	Finishes the processing steps; saves and closes the query.
New Sources	Lets you discover and add a new data source to the set of queries.
Recent Sources	Lists all the recent data sources that you have used.
Refresh Preview	Refreshes the source data and reprocesses all the steps.
Properties	Displays the query properties and lets you add a description and inhibit or enable data load and refresh.
Advanced Editor	Displays the complete “M” language code for all the transformation steps that have been used in a query.
Choose Columns	Lets you select the columns to retain from all the columns available in the source data.
Remove Columns	Removes one or more columns.
Keep Rows	Keeps the specified number of rows at the top of the table.
Remove Rows	Removes a specified number of rows from the top of the data table.
Remove Duplicates	Removes all duplicate rows, leaving only unique rows.
Remove Errors	Removes all rows that contain processing errors.
Sort	Sorts the table using the selected column as the sort key.
Split Column	Splits a column into one or many columns at a specified delimiter or after a specified number of characters.

(continued)

Table 3-1. (continued)

Option	Description
Group By	Groups the table using a specified set of columns and aggregates any numeric columns for this grouping.
Data Type	Applies the chosen data type to the column.
Use First Row As Headers	Use the first row as the column titles.
Replace Values	Carries out a search-and-replace operation on the data in a column or columns. This only affects the complete data in a column.
Merge Queries	Joins a second query table to the current query results and aggregates or adds data from the second to the first.
Append Queries	Adds the data from another query to the current query in the current Power BI Desktop file.
Combine Binaries	Merges all the binary data from another column into a single binary column.

The Transform Ribbon

The Transform ribbon, as its name implies, contains a wealth of functions that can help you to transform your data. The various options it contains are explained in Figure 3-4 and Table 3-2.

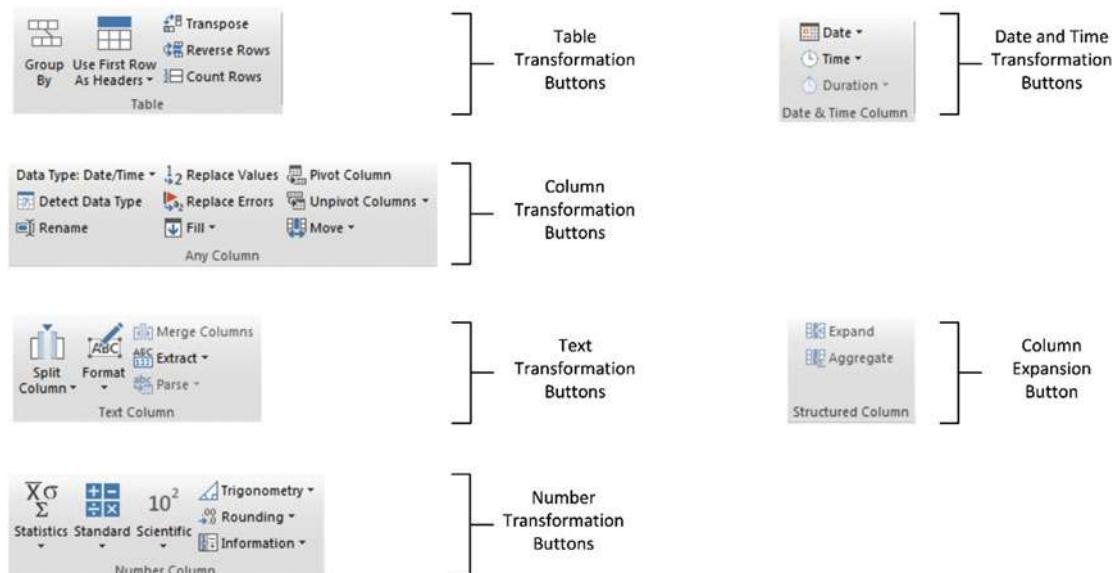
**Figure 3-4.** The Query Editor Transform ribbon

Table 3-2. Query Editor Transform Ribbon Options

Option	Description
Group By	Groups the table using a specified set of columns; aggregates any numeric columns for this grouping.
Use First Row As Headers	Uses the first row as the column titles.
Transpose	Transforms the columns into rows and the rows into columns.
Reverse Rows	Displays the source data in reverse order, showing the final rows at the top of the window.
Count Rows	Counts the rows in the table and replaces the data with the row count.
Data Type	Applies the chosen data type to the column.
Rename	Renames a column.
Detect Data Type	Detects the correct data type to apply to multiple columns.
Replace Values	Carries out a search-and-replace operation inside a column, replacing a specified value with another value.
Replace Errors	Replaces error values with a specified value.
Fill	Copies the data from cells above or below into empty cells in the column.
Pivot Column	Creates a new set of columns using the data in the selected column as the column titles.
Unpivot Columns	Takes the values in a set of columns and unpivots the data, creating two new columns using the column headers as the descriptive elements.
Move	Moves a column.
Split Column	Splits a column into one or many columns at a specified delimiter or after a specified number of characters.
Format	Modifies the text format of data in a column (uppercase, lowercase, capitalization) or removes trailing spaces.
Merge Columns	Takes the data from several columns and places it in a single column, adding an optional separator character.
Extract	Replaces the data in a column using a defined subset of the current data. You can specify a number of characters to keep from the start or end of the column, set a range of characters beginning at a specified character, or even list the number of characters in the column.
Parse	Creates an XML or JSON document from the contents of each cell in a column.
Statistics	Returns the Sum, Average, Maximum, Minimum, Median, Standard Deviation, Count, or Distinct Value Count for all the values in the column.
Standard	Carries out a basic mathematical calculation (add, subtract, divide, multiply, integer-divide, or return the remainder) using a value that you specify applied to each cell in the column.
Scientific	Carries out a basic scientific calculation (square, cube, power of n, square root, exponent, logarithm, or factorial) for each cell in the column.

(continued)

Table 3-2. (continued)

Option	Description
Trigonometry	Carries out a basic trigonometric calculation (Sine, Cosine, Tangent, ArcSine, ArcCosine, or ArcTangent) using a value that you specify applied to each cell in the column.
Rounding	Rounds the values in the column either to the next integer (up or down) or to a specified factor.
Information	Replaces the value in the column with simple information: Is Odd, Is Even, or Positive/Negative.
Date	Isolates an element (day, month, year, etc.) from a date value in a column.
Time	Isolates an element (hour, minute, second, etc.) from a date/time or time value in a column.
Duration	Calculates the duration from a value that can be interpreted as a duration in days, hours, minutes, and so forth.
Expand	Adds the (identically structured) data from another query to the current query.
Aggregate	Calculates the sum or product of numeric columns from another query and adds the result to the current query.

The Add Column Ribbon

The Add Column ribbon does a lot more than just add columns. It also contains functions to break columns down into multiple columns, and to add columns containing dates and calculations based on existing columns. The various options it contains are explained in Figure 3-5 and Table 3-3.

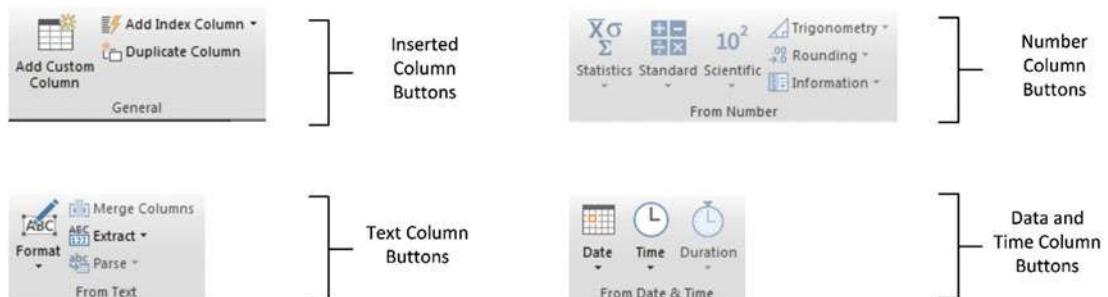
**Figure 3-5.** The Query Editor Add Column ribbon

Table 3-3. *Query Editor Add Column Ribbon Options*

Option	Description
Add Custom Column	Adds a new column using a formula to create the columns' contents.
Add Index Column	Adds a sequential number in a new column to uniquely identify each row.
Duplicate Column	Creates a copy of the current column.
Format	Modifies the text format of data in a column (uppercase, lowercase, capitalization) or removes trailing spaces.
Merge Columns	Takes the data from several columns and places it in a single column, adding an optional separator character.
Extract	Replaces the data in a column using a defined subset of the current data. You can specify a number of characters to keep from the start or end of the column, set a range of characters beginning at a specified character, or even list the number of characters in the column.
Parse	Creates an XML or JSON document from the contents of each cell in a column.
Statistics	Returns the Sum, Average, Maximum, Minimum, Median, Standard Deviation, Count, or Distinct Value Count for all the values in the column.
Standard	Carries out a basic mathematical calculation (add, subtract, divide, multiply, integer-divide, or return the remainder) using a value that you specify applied to each cell in the column.
Scientific	Carries out a basic scientific calculation (square, cube, power of n, square root, exponent, logarithm, or factorial) for each cell in the column.
Trigonometry	Carries out a basic trigonometric calculation (Sine, Cosine, Tangent, ArcSine, ArcCosine, or ArcTangent) using a value that you specify applied to each cell in the column.
Rounding	Rounds the values in the column either to the next integer (up or down) or to a specified factor.
Information	Replaces the value in the column with simple information: Is Odd, Is Even, or Positive/Negative.
Date	Isolates an element (day, month, year, etc.) from a date value in a column.
Time	Isolates an element (hour, minute, second, etc.) from a date/time or time value in a column.
Duration	Calculates the duration from a value that can be interpreted as a duration in days, hours, minutes, and so forth.

The View Ribbon

The View ribbon lets you alter some of the Query settings and see the underlying data transformation code. The various options that it contains are explained in the next chapter.

Dataset Shaping

So you are now looking at a data table that you have loaded into Power BI Desktop. For argument's sake, let's assume that it is the C:\PowerBiDesktopSamples\CarSalesFirstDashboard.pbix file from the sample data directory, and that you have clicked the Edit Queries button to display the Power BI Desktop Query editor. What can you do to the CarSalesData data table that is now visible? It is time to take a look at some of the core techniques that you can apply to shape the initial dataset. These include the following:

- Renaming columns
- Reordering columns
- Removing columns
- Merging columns
- Removing records
- Removing duplicate records
- Filtering the dataset

I have grouped these techniques together as they affect the initial size and shape of the data. Also, it is generally not only good practice, but also easier for you, the data modeler, if you begin by excluding any rows and columns that you do not need. I also find it easier to understand datasets if the columns are logically laid out and given comprehensible names from the start. All in all, this makes working with the data easier in the long run.

Renaming Columns

Although we took a quick look at renaming columns in the first pages of this chapter, let's look at this technique again in more detail. I admit that renaming columns is not actually modifying the form of the data table. However, when dealing with data, I consider it vital to have all data clearly identifiable. This implies meaningful column names being applied to each column. Consequently, I consider this modification to be fundamental to the shape of the data and also as an essential best practice when importing source data.

To rename a column:

1. Click inside the column that you want to rename.
2. Click Transform to activate the Transform ribbon.
3. Click the Rename button. The column name will be highlighted.
4. Enter the new name or edit the existing name.
5. Press Enter or click outside the column title.

The column will now have a new title. The Applied Steps list on the right will now contain another element, Renamed Columns. This step will be highlighted.

Note As an alternative to using the Transform ribbon, you can right-click the column title and select Rename.

Reordering Columns

Power BI Desktop will load data as it is defined in the data source. Consequently, the column sequence will be entirely dependent on the source data (or by a SQL query if you used a source database, as described earlier). This need not be definitive, however, and you can reorder the columns if that helps you understand and deal with the data. Do the following to change column order:

1. Click the header of the column you want to move.
2. Drag the column left or right to its new position. You will see the column title slide laterally through the column titles as you do this, and a thicker gray line will indicate where the column will be placed once you release the mouse button.
Reordered Columns will appear in the Applied Steps list.

Figure 3-6 shows this operation.

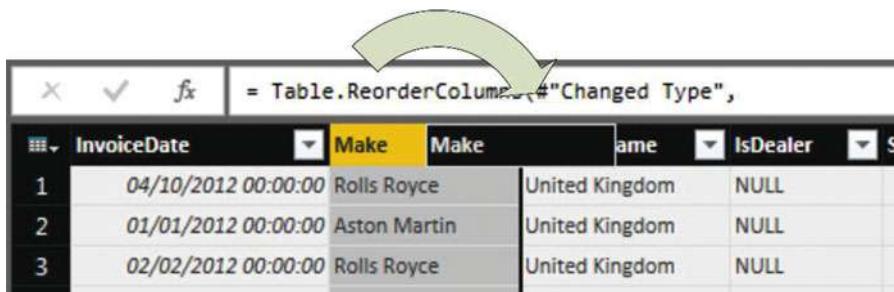


Figure 3-6. Reordering columns

If your query contains dozens—or even hundreds—of columns, you may find that dragging a column around can be slow and laborious. Equally, if columns are extremely wide, it can be difficult to “nudge” a column left or right. Power BI Desktop can come to your aid in these circumstances with the Move button in the Transform ribbon. Clicking this button gives you the menu options that are outlined in Table 3-4.

Table 3-4. Move Button Options

Option	Description
Left	Moves the currently selected column to the left of the column on its immediate left.
Right	Moves the currently selected column to the right of the column on its immediate right.
To Beginning	Moves the currently selected column to the left of all the columns in the query.
To End	Moves the currently selected column to the right of all the columns in the query.

The Move command also works on a set of columns that you have selected by Ctrl-clicking and/or Shift-clicking. Indeed, you can move a selection of columns that is not contiguous if you need to.

Note You need to select a column (or a set of columns) before clicking the Move button. If you do not, then the first time that you use Move Power BI Desktop query selects the column(s)—but not move them.

Removing Columns

So how do you delete a column or series of columns? Like this:

1. Click inside the column you want to delete, or if you want to delete several columns at once, Ctrl-click the titles of the columns that you want to delete.
2. Click the Remove Columns button in the Home ribbon. The column(s) will be deleted and Removed Columns will be the latest element in the Applied Steps list.

When working with imported datasets over which you have had no control, you may frequently find that you only need a few columns of a large data table. If this is the case, you will soon get tired of Ctrl-clicking on numerous columns to select those you want to remove. Power BI Desktop has an alternative method. Just select the columns you want to keep and delete the others. To do this:

1. Ctrl-click the titles of the columns that you want to keep.
2. Click the small triangle in the Remove Columns button in the Home ribbon. Select Remove Other Columns from the menu. All unselected columns will be deleted and Removed Other Columns will be added to the Applied Steps list.

When selecting a contiguous range of columns to remove or keep, you can use the standard Windows Shift-Click technique to select from the first to the last column in the block of columns that you want to select.

Note Both of these options for removing columns are also available from the context menu, if you prefer. It shows Remove (or Remove Columns, if there are several columns selected) when deleting columns, as well as Remove Other Columns if you right-click a column title.

Merging Columns

Source data is not always exactly as you wish it could be (and that is sometimes a massive understatement). Certain data sources could have data spread over many columns that could equally well be merged into a single column. So it probably comes as no surprise to discover that Power BI Desktop query can carry out this kind of operation too. Here is how to do it:

1. Ctrl-click the headers of the columns that you want to merge (Make and Model in the CarSalesData table in this example).
2. In the Transform ribbon, click the Merge Columns button. The Merge Columns dialog will be displayed.
3. From the Separator pop-up menu, select one of the available separator elements. I chose Colon in this example.

4. Enter a name for the column that will be created from the two original columns (I am calling it MakeAndModel). The dialog should look like Figure 3-7.



Figure 3-7. The Merge Columns dialog

5. Click OK. The columns that you selected will be replaced by the data from all the columns, as shown in Figure 3-8.

			= Table.CombineColumns(#"Reordered Columns",
#	InvoiceDate	MakeAndModel	CountryName
1	04/10/2012 00:00:00	Camargue:Rolls Royce	United Kingdom
2	01/01/2012 00:00:00	DBS:Aston Martin	United Kingdom
3	02/02/2012 00:00:00	Silver Ghost:Rolls Royce	United Kingdom
4	03/03/2012 00:00:00	Silver Ghost:Rolls Royce	United Kingdom
5	04/04/2012 00:00:00	Camargue:Rolls Royce	United Kingdom
6	04/05/2012 00:00:00	Camargue:Rolls Royce	United Kingdom
7	04/06/2012 00:00:00	DBS:Aston Martin	United Kingdom
8	04/07/2012 00:00:00	DB7:Aston Martin	France
9	04/08/2012 00:00:00	DB9:Aston Martin	United Kingdom
10	04/09/2012 00:00:00	DB9:Aston Martin	United Kingdom
11	04/09/2012 00:00:00	DB4:Aston Martin	United Kingdom
12	04/11/2012 00:00:00	Vantage:Aston Martin	France
13	04/11/2012 00:00:00	Vanquish:Aston Martin	United Kingdom
14	04/12/2012 00:00:00	Rapide:Aston Martin	United Kingdom

Figure 3-8. The result of merging columns

I need to make a few comments about this process.

- You can select as many columns as you want when merging columns.
- If you do not give the resulting column a name in the Merge Columns dialog, it will simply be renamed Merged. You can always rename it later if you want.
- The order in which you select the columns affects the way that the data is merged. So, always begin by selecting the column whose data must appear at the left of the merged column, then the column whose data should be next, and so forth. You do not have to select columns in the order that they initially appeared in the dataset.
- If you do not want to use any of the standard separators that Power BI Desktop query suggests, you can always define your own. Just select --Custom-- in the pop-up menu in the Merge Columns dialog. A new box will appear in the dialog, in which you can enter your choice of separator. This can be composed of several characters if you really want.
- Merging columns from the Transform ribbon removes all the selected columns and replaces them with a single column. The same option is also available from the Add Column ribbon—only in this case, this operation adds a new column and leaves the original columns in the dataset.

Note This option is also available from the context menu if you right-click a column title.

The available merge separators are described in Table 3-5.

Table 3-5. *Merge Separators*

Option	Description
Colon	Uses the colon (:) as the separator.
Comma	Uses the comma (,) as the separator.
Equals Sign	Uses the equals sign (=) as the separator.
Semi-Colon	Uses the semicolon (;) as the separator.
Space	Uses the space () as the separator.
Tab	Uses the tab character as the separator.
Custom	Lets you enter a custom separator.

Tip You can split, remove, and duplicate columns using the context menu if you prefer. Just remember to right-click the column title to display the correct context menu.

Removing Records

You may not always need *all* the data that you have loaded into a Power BI Desktop query. There could be several possible reasons for this:

- You are taking a first look at the data and you only need a sample to get an idea of what the data is like.
- The data contains records that you clearly do not need and that you can easily identify from the start.
- You are testing data cleansing and you want a smaller dataset to really speed up the development of a complex data extractions and transformation process.
- You want to analyze a reduced dataset to extrapolate theses and inferences, and to save analysis on a full dataset for later, or even using a more industrial-strength toolset such as SQL Server Integration Services.

To allow you to reduce the size of the dataset, Power BI Desktop proposes two basic approaches out of the box:

- Keep certain rows
- Remove certain rows

Inevitably, the technique that you adopt will depend on the circumstances. If it is easier to specify the rows to sample by inclusion, then the keep-certain-rows approach is the best option to take. Inversely, if you want to proceed by exclusion, then the remove-certain-rows technique is best. Let's look at each of these in turn.

Keep Rows

This approach lets you specify the rows that you want to continue using. It is based on the application of one of the following three choices:

- Keep the top *n* records.
- Keep the bottom *n* records.
- Keep a specified range of records—that is, keep *n* records every *y* records.

Most of these techniques are very similar, so let's start by imagining that you want to keep the top 50 records in the sample C:\PowerBiDesktopSamples\ CarSalesFirstDashboard.pbix file.

1. In the Home ribbon of the Power BI Query Editor, click the Keep Rows button. The menu will appear.
2. Select Keep Top Rows. The Keep Top Rows dialog will appear.
3. Enter **50** in the “Number of rows” box, as shown in Figure 3-9.



Figure 3-9. The Keep Top Rows dialog

4. Click OK. All but the first 50 records are deleted and Kept First Rows is added to the Applied Steps list.

To keep the bottom n rows, the technique is virtually identical. Follow the steps in the previous example, but select Keep Bottom Rows in step 2. In this case, the Applied Steps list displays Kept Last Rows.

To keep a range of records, you need to specify a starting record and the number of records to keep from then on. For instance, suppose that you wish to lose the first 10 records but keep the following 25. This is how to go about it:

1. In the Home ribbon, click the Keep Rows button.
2. Select Keep Range of Rows. The Keep Range dialog will appear.
3. Enter **11** in the “First row” box.
4. Enter **25** in the “Number of rows” box, as shown in Figure 3-10.



Figure 3-10. The Keep Range dialog

5. Click OK. All but records 1–10 and 36 to the end are deleted and Kept Range Of Rows is added to the Applied Steps list.

Remove Rows

Removing rows is a nearly identical process to the one you just used to keep rows. As removing the top or bottom n rows is highly similar, I will not go through it in detail. All you have to do is click the Remove Rows button in the Home ribbon and follow the process as if you were keeping rows. The Applied Steps list will read Removed Top Rows or Removed Bottom Rows in this case, and rows will be removed instead of being kept in the dataset, of course.

The remove rows approach does have one very useful option that can be applied as a sampling technique. It allows you to remove one or more records every few records to produce a subset of the source data. To do this, you need to do the following:

1. Click the Remove Rows button in the Query window Home ribbon. The menu will appear.
2. Select Remove Alternate Rows. The Remove Alternate Rows dialog will appear.
3. Enter **10** as the First row to remove.
4. Enter **2** as the Number of rows to remove.
5. Enter **10** as the Number of rows to keep.

The dialog will look like Figure 3-11.



Figure 3-11. The Remove Alternate Rows dialog

6. Click OK. All but the records matching the pattern you entered in the dialog are removed. Removed Alternate Rows is then added to the Applied Steps list.

Note If you are really determined to extract a sample that you consider to be representative of the key data, then you can always filter the data before subsetting it to exclude any outliers. Filtering data is explained later in this chapter.

Remove Blank Rows

If your source data contains completely blank (empty) rows, you can delete these as follows:

1. Click the Remove Rows button in the Query window Home ribbon. The menu will appear.
2. Select Remove Blank Rows.

This results in empty rows being deleted. Removed Blank Rows is then added to the Applied Steps list.

Removing Duplicate Records

An external source of data might not be quite as perfect as you might hope. One of the most annoying features of poor data is the presence of duplicates. These are insidious since they falsify results and are not always visible. If you suspect that the data table contains strict duplicates (that is, where every field is identical in two or more records), then you can remove the duplicates like this:

Click the Remove Duplicates button in the Home ribbon. All duplicate records are deleted and Removed Duplicates are added to the Applied Steps list.

Note I must stress that this approach will only remove *completely* identical records. If two records have just one different character or a number but everything else is identical, then they are *not* considered duplicates by the Power BI Desktop Query Editor. Alternatively, if you want to isolate and examine the duplicate records, then you can display only completely identical records by selecting Keep Duplicates from the popup menu for the Remove Duplicates button.

So if you suspect or are sure that the data table you are dealing with contains duplicates, what are the practical solutions? This can be a real conundrum, but there are some basic techniques that you can apply:

- Remove all columns that you are sure you will not be using later in the data-handling process. This way, Power BI Desktop will only be asked to compare essential data across potentially duplicate records.
- Group the data on the core columns (this is explained in the next chapter).

Note As you have seen, Power BI Desktop Query can help you to home in on the essential elements in a dataset in just a few clicks. If anything, you need to be careful that you are *not excluding* valuable data—and consequently skewing your analysis—when excluding data from the query.

Sorting Data

Although not strictly a data modification step, sorting an imported table will probably be something that you want to do at some stage, if only to get a clearer idea of the data that you are dealing with. Do the following to sort the data:

1. Click inside the column you wish to sort by.
2. Click Sort Ascending (the A/Z icon) or Sort Descending (the Z/A icon) in the Home ribbon.

The data is sorted in either alphabetical (smallest to largest) or reverse alphabetical (largest to smallest) order. If you want to carry out a complex sort operation (that is, first by one column and then by another if the first column contains the same element over several rows), you do this simply by sorting the columns one after another. Power BI Desktop Query Editor adds a tiny 1, 2, 3, and so on to the right of the column title to indicate the sort sequence. You can see this in Figure 3-12.

Make	Model
Aston Martin	DB4
Aston Martin	DB7
Aston Martin	DB7
Aston Martin	DB7

Figure 3-12. Sorting multiple columns

As sorting data is considered part of the data modification process, it also appears in the Applied Steps list as Sorted Rows.

Note An alternative technique for sorting data is to click the pop-up menu for a column (the downward-facing triangle at the right of a column title) and select Sort Ascending or Sort Descending from the pop-up menu.

Reversing the Row Order

If you find that the data that you are looking at seems upside-down (that is, with the bottom rows at the top and vice versa), you can reverse the row order in a single click, if you want. To do this, do the following:

In the Transform ribbon, click the Reverse Rows button.

The entire dataset will be reversed and the bottom row will now be the top row.

Filtering Data

The most frequently used way of limiting a dataset is, in my experience, the use of filters on the table that you have loaded. Now, I realize that you may be coming to Power BI Desktop after years with Excel, or after some time using PowerPivot, and that the filtering techniques that you are about to see probably look much like the ones you have used in those two tools. However, because it is fundamental to include and exclude appropriate records when loading source data, I will thoroughly handle Power BI Desktop filters, even if this means that certain readers will experience a strong sense of *déjà vu*.

Here are two basic approaches for filtering data in Power BI Desktop:

- Select one or more specific values from the unique list of elements in the chosen column.
- Define a range of data to include or exclude.

The first option is common to all data types, whether they are text, number, or data/time. The second approach varies according to the data type of the column that you are using to filter data.

Selecting Specific Values

Selecting one or more values present in a column of data is as easy as this (assuming that you are still using the Power BI Desktop file CarSalesFirstDashboard.bix and are in the Query Editor):

1. Click a column's pop-up menu. (I used Make in the sample dataset in this example.) The filter menu appears.
2. Check all elements that you want to retain and uncheck all elements that you wish to exclude. In this example, I kept Bentley and Rolls-Royce, as shown in Figure 3-13.

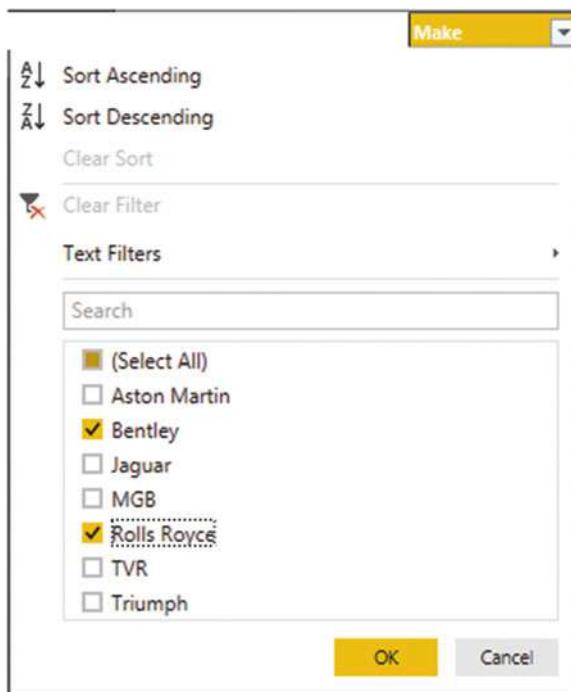


Figure 3-13. A Filter menu

3. Click OK. The Applied Steps box adds Filtered Rows.

Note You can deselect all items by clicking the (Select All) check box; reselect all the items by selecting this box again. It follows that, if you want to keep only a few elements, it may be faster to unselect all of them first and then only select the ones that you want to keep. If you want to exclude any records without a value in the column that you are filtering on then select Remove Empty from the filter menu.

Finding Elements in the Filter List

Scrolling up and down in a filter list can get extremely laborious. A fast way of limiting the list to a subset of available elements is to do the following:

1. Click the pop-up menu for a column. (I use Model in the sample dataset in this example.) The filter menu appears.
2. Enter a letter or a few letters in the Search box. The list shortens with every letter or number that you enter. If you enter **ar**, then the filter popup will look like Figure 3-14.

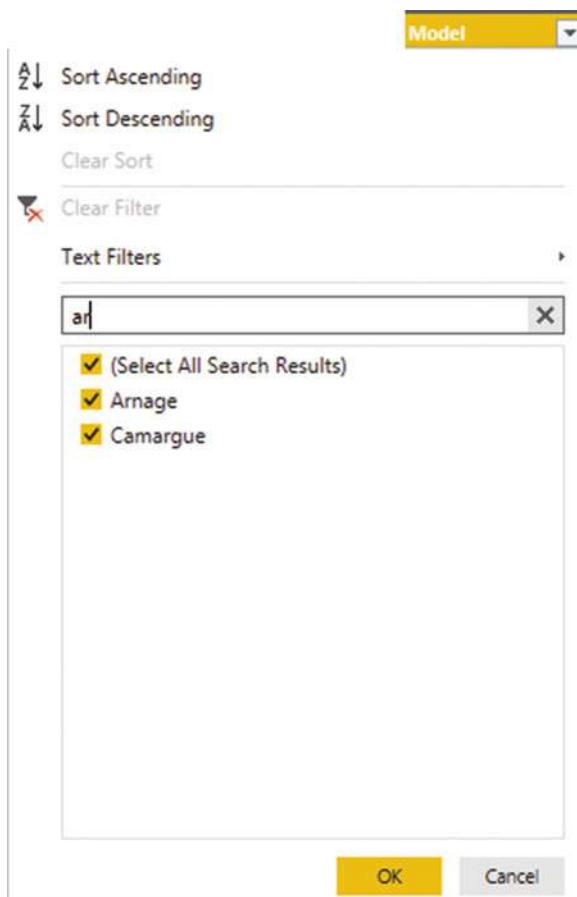


Figure 3-14. Searching the filter menu

To remove a filter, all that you have to do is click the cross that appears at the right of the Search box.

Filtering Text Ranges

If a column contains text, then you can apply specific options to filter the data. These elements are found in the filter pop-up of any text-based column in the Text Filters submenu. The choices are given in Table 3-6.

Table 3-6. *Text Filter Options*

Filter Option	Description
Equals	Sets the text that must match the cell contents.
Does Not Equal	Sets the text that must not match the cell contents.
Begins With	Sets the text at the left of the cell contents.
Does Not Begin With	Sets the text that must not appear at the left of the cell contents.
Ends With	Sets the text at the right of the cell contents.
Does Not End With	Sets the text that must <i>not</i> appear at the right of the cell contents.
Contains	Lets you enter a text that will be part of the cell contents.
Does Not Contain	Lets you enter a text that will <i>not</i> be part of the cell contents.

Filtering Numeric Ranges

If a column contains numbers, then there are also specific options that you can apply to filter the data. You'll find these elements in the filter pop-up of any text-based column in the Number Filters submenu. The choices are given in Table 3-7.

Table 3-7. *Numeric Filter Options*

Filter Option	Description
Equals	Sets the number that must match the cell contents.
Does Not Equal	Sets the number that must not match the cell contents.
Greater Than	Cell contents must be greater than this number.
Greater Than Or Equal To	Cell contents must be greater than or equal to this number.
Lesser Than	Cell contents must be less than this number.
Lesser Than Or Equal To	Cell contents must be less than or equal to this number.
Between	Cell contents must be between the two numbers.

Filtering Date and Time Ranges

If a column contains dates or times (or both), then specific options can also be applied to filter the data. These elements are found in the filter pop-up of any text-based column in the Date/Time Filters submenu. The choices are given in Table 3-8.

Table 3-8. Date and Time Filter Options

Filter Element	Description
Equals	Filters data to include only records for the selected date.
Before	Filters data to include only records up to the selected date.
After	Filters data to include only records after the selected date.
Between	Lets you set an upper and a lower date limit to exclude records outside that range.
In the Next	Lets you specify a number of days, weeks, months, quarters, or years to come.
In the Previous	Lets you specify a number of days, weeks, months, quarters, or years up to the date.
Day ▶ Tomorrow	Filters data to include only records for the day after the current system date.
Day ▶ Today	Filters data to include only records for the current system date.
Day ▶ Yesterday	Filters data to include only records for the day before the current system date.
Week ▶ Next Week	Filters data to include only records for the next calendar week.
Week ▶ This Week	Filters data to include only records for the current calendar week.
Week ▶ Last Week	Filters data to include only records for the previous calendar week.
Month ▶ Next Month	Filters data to include only records for the next calendar month.
Month ▶ This Month	Filters data to include only records for the current calendar month.
Month ▶ Last Month	Filters data to include only records for the previous calendar month.
Quarter ▶ Next Quarter	Filters data to include only records for the next quarter.
Quarter ▶ This Quarter	Filters data to include only records for the current quarter.
Quarter ▶ Last Quarter	Filters data to include only records for the previous quarter.
Year ▶ Next Year	Filters data to include only records for the next year.
Year ▶ This Year	Filters data to include only records for the current year.
Year ▶ Last Year	Filters data to include only records for the previous year.
Year ▶ Year To Date	Filters data to include only records for the calendar year to date.
Custom Filter	Lets you set up a specific filter for two possible date ranges.

Filtering Data

Filtering data uses a very similar approach, whatever the type of filter that is applied. As a simple example, here is how to apply a number filter to the sale price to find vehicles that sold for less than £5,000.00.

1. Click the pop-up menu for the SalePrice column.
2. Click Number Filters. The submenu will appear.
3. Select Less than. The Filter Rows dialog will be displayed.
4. Enter **5000** in the box next to the “is less than” box, as shown in Figure 3-15.

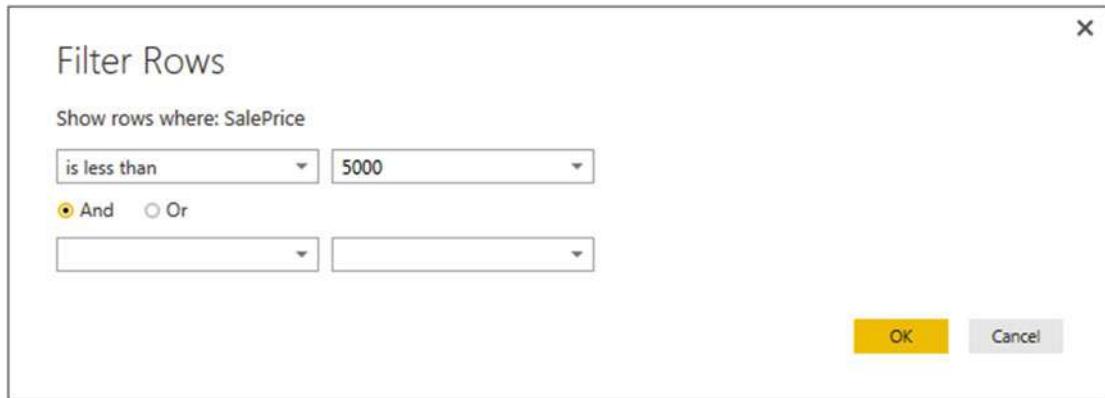


Figure 3-15. The Filtered Rows dialog

5. Click OK. The dataset only displays rows that conform to the filter that you have defined.

Although extremely simple to apply, filters do require a few comments:

- You can combine up to two elements in a filter. These can be mutually inclusive (an AND filter) or they can be an alternative (an OR filter).
- You should not apply any formatting when entering numbers.
- Any text that you filter on is not case-sensitive.
- If you choose the wrong filter, you do not have to cancel and start over. Simply select the correct filter type from the pop-up in the left-hand boxes in the Filter Rows dialog.

Note If you set a filter value that excludes all the records in the table, Power BI Desktop displays an empty table except for the words “This table is empty”. You can always remove the filter by clicking the cross to the left of Filtered Rows in the Applied Steps list.

Conclusion

This chapter started you on the road to transforming datasets with Power BI Desktop. You saw how to trim datasets by removing rows and columns. You also saw how to subset a sample of data from a data source by selecting alternating groups of rows.

You saw how to move columns around in the dataset and how to rename columns so that your data is easily comprehensible when you use it later in dashboards and reports. Finally, you saw how to filter and sort data, as well as how to remove duplicates to ensure that your dataset only contains the precise rows that you need for your upcoming visualizations.

Preparing raw data for use in dashboards and reports is not always easy and can take a while to get right. However, Power BI Desktop Query can make this task really easy with a little practice.

So now that you have grasped the basics, it is time to move on and discover some further data transformation techniques. Specifically, you will see how to cleanse the data that you have imported. This is the subject of the next chapter.

CHAPTER 4



Data Cleansing

Once a dataset has been shaped and filtered, it probably still needs a good few modifications to make it ready for consumption. Many of these modifications are, at their heart, a selection of fairly simple yet necessary techniques that you apply to make the data cleaner and more standardized. I have chosen to group these approaches under the heading *data cleansing*.

The sort of things that you may be looking to do before finally loading source data into the data model can include the following:

- Change the data type for a column—by telling Power BI Desktop that the column contains numbers, for example
- Replace the values in a cell with other values
- Transform the column contents—by making the text uppercase, for instance, or by removing decimals from numbers
- Apply math or statistical (or even trigonometric) functions to columns of numbers
- Convert date or time data into date elements such as days, months, quarters, years, hours, or minutes
- Fill data down or up over empty cells to ensure that records are complete
- Ensure that the first row is used as headers (if this is required)
- Group record sets and aggregate numbers

This chapter will take you on a tour of these kinds of essential data transformations. Once you have finished reading it, you should be confident that you can take a rough and ready data source as a starting point and convert it into a polished and coherent data table that is ready to become a pivotal part of your Power BI Desktop data model. Not only that, but you will have carried out really heavy lifting much faster and more easily than you could have done using enterprise-level tools.

Viewing a Full Record

Before even starting to cleanse data, you probably need to take a good look at it. While the Power BI Desktop Query Editor is great for scrolling up and down columns to see how data compares for a single field, it is often less easy to appreciate the entire contents of a single record.

So to avoid having to scroll frenetically left and right across rows of data, the Query Editor has another brilliantly simple solution. If you click a row (or more specifically, on the number of a row in the grid on the left), the Power BI Desktop Query Editor will display the contents of an entire record in a single window under the dataset. You can see an example of this in Figure 4-1.

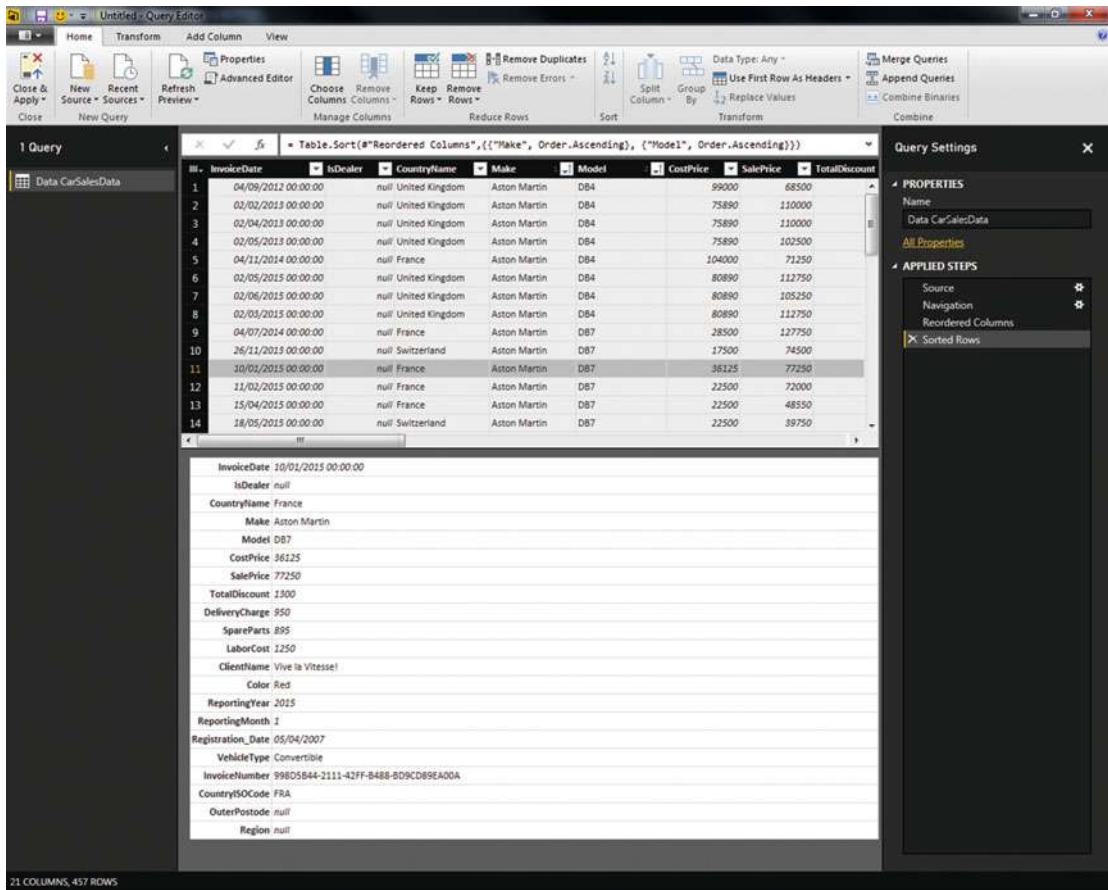


Figure 4-1. Viewing a full record

Power BI Desktop Query Editor Context Menus

As is normal for Windows programs, Power BI Desktop Query Editor makes full use of context (or “right-click”) menus as an alternative to using the ribbons. When transforming datasets, there are three main context menus that you will probably find yourself using:

- *Table menu:* This menu appears when you right-click the top corner of the grid containing the data
- *Column menu:* This menu appears when you right-click a column title
- *Cell menu:* This menu appears when you right-click a data cell

While I have referred copiously to the context menus when explaining how to transform data, it is probably easier to take a quick look at them now so that you can see the various options. Figure 4-2 gives you a quick overview of these three context menus.

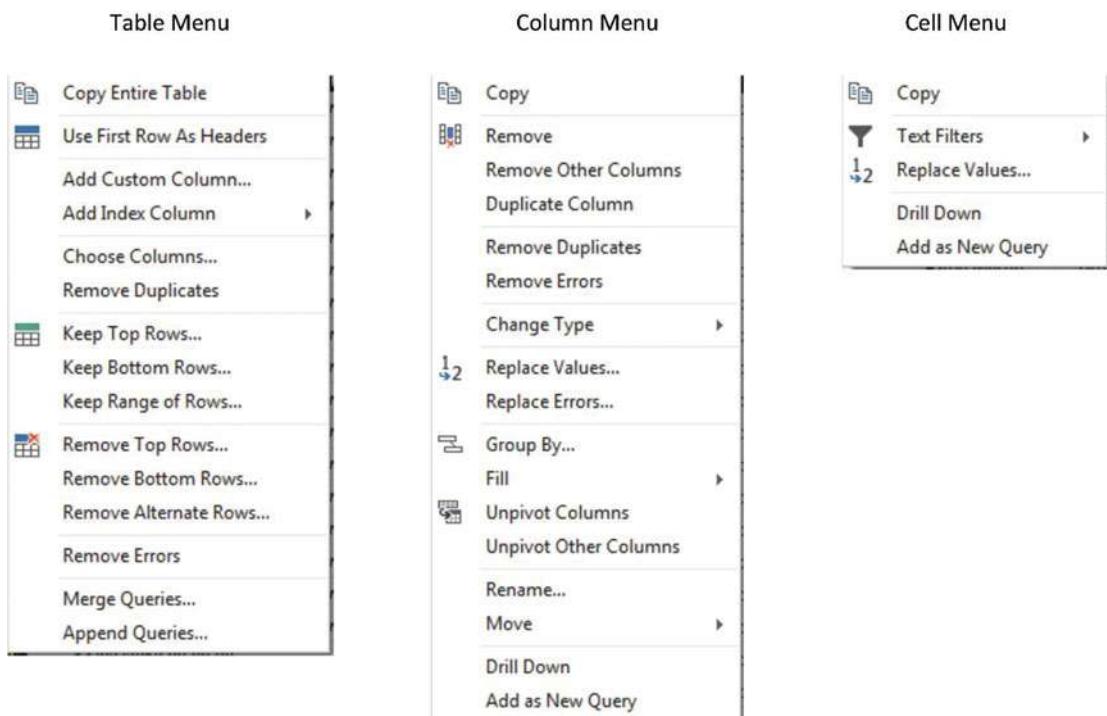


Figure 4-2. The Power BI Desktop Query Editor context menus

Changing Data Type

A truly fundamental aspect of data modification is ensuring that the data is of the appropriate type; that is, if you have a column of numbers that are to be calculated at some point, then the column should be a numeric column. If it contains dates, then it should be set to one of the date or time data types. I realize that this can seem arduous and even superfluous; however, *if you want to be sure that your data can be sliced and diced correctly further down the line*, then setting the right data types at the outset is *vital*. An added bonus is that if you validate the data types early on in the process of loading data, you can see from the start if the data has any potential issues—dates that cannot be read as dates, for instance. This allows you to decide what to do with poor or unreliable data early in your work with a dataset.

The good news here is that for many data sources, Power BI Desktop applies an appropriate data type. Specifically, if you have loaded data from a database, then Power BI Desktop will recognize the data type for each column and apply a suitable native data type. Unfortunately, things can get a little more painful with file sources, specifically CSV, text, and Excel files, as well as some XML files. In the case of these file types, Power BI Desktop often tries to guess the data type, but there are times when it does not succeed. If it has made a stab at deducing data types, then you see a Changed Type step in the Applied Steps list. Consequently, if you are obtaining your data from these sources, then you could well be obliged to apply data types to many of the columns manually.

Note In some cases, numbers are not meant to be interpreted as numerical data. For instance, a French postal code is five numbers, but it will never be calculated in any way. So it is good practice to let Power BI Desktop know this by changing the data type to Text.

Do the following to change data type for a column or a group of columns:

1. Open the sample file C:\PowerBiDesktopSamples\CarSalesFirstDashboard.pbix.
2. Click the Edit Queries button in the Home ribbon. The Query Editor will open.
3. Click inside the column whose data type you wish to change. If you want to modify several columns, then Ctrl-click the requisite column titles. In this example, you could select the CostPrice and TotalDiscount columns.
4. Click the Data Type button in the Transform section of the Power BI Desktop Query Home ribbon. A pop-up menu of potential data types will appear.
5. Select an appropriate data type. If you have selected the CostPrice and TotalDiscount columns, then Whole Number is the type to choose.

After a few seconds, the data type will be applied. Changed Type will appear in the Applied Steps pane. The data types that you can apply are outlined in Table 4-1.

Table 4-1. Data Types in Power BI Desktop

Data Type	Description
Decimal Number	Converts the data to a decimal number.
Fixed Decimal Number	Converts the data to a decimal number with a fixed number of decimals.
Whole Number	Converts the data to a whole (integer) number.
Date/Time	Converts to a date and time data type.
Date	Converts to a date data type.
Time	Converts to a time data type.
Duration	Sets the data as being a duration. These are used for date and time calculations.
Text	Sets to a text data type.
True/False	Sets the data type to Boolean (True or False).
Binary	Defines the data as binary, and consequently, it is not directly visible.

Inevitably, there will be times when you try to apply a data type that simply cannot be used with a certain column of data. Converting a text column (such as Make in this sample data table) into dates will simply not work. If you do this, then Power BI Desktop will replace the column contents with Error. This is not definitive or dangerous, and all you have to do to return the data to its previous state is to delete the Changed Type step in the Applied Steps list using the technique described in the previous chapter.

It can help to alter data types at the same time for a *set* of columns where you think that this operation is necessary. There are a couple of good reasons for this approach:

- You can concentrate on getting data types right, and if you are working methodically, you are less likely to forget to set a data type.
- Applying data types for many columns (even if you are doing this in several operations, to single or multiple columns) will only add a single step to the Applied Steps list.

Note Don't look for any data formatting options in Power BI Desktop Query; there aren't any. This is deliberate since this tool is designed to structure, load, and cleanse data, but not to present it. You carry out the formatting in the Power BI Desktop data view, as you will see in Chapter 6.

Detecting Data Types

Applying the correct data type to dozens of columns can be more than a little time-consuming. Fortunately, Power BI Desktop now contains an option to apply data types automatically to a whole table.

1. In the Transform ribbon, click the Detect Data Type button.
2. Changed Type will appear in the Applied Steps list. Most of the columns will have the correct data type applied.

This technique does not always give perfect results, and there will be times when you want to override the choice of data type that Power BI Desktop has applied. Yet it is nonetheless a welcome addition to the data preparation toolset that can save you considerable time when preparing a dataset.

Replacing Values

Some data that you load will need certain values to be replaced by others in a kind of global search-and-replace operation—just as you would in a document. For instance, perhaps you need to standardize spellings where a make of car (to use the current sample dataset as an example) has been entered incorrectly. To carry out this particular data cleansing operation, do the following:

1. Click the title of the column that contains the data that you want to replace. The column will become selected. In this example, I used the Model column as an example.
2. In the Home ribbon, click the Replace Values button. The Replace Values dialog will appear.
3. In the Value To Find box, enter the text or number that you want to replace. I used Ghost in this example.
4. In the Replace With box, enter the text or number that you want to replace. I used Fantôme in this example, as shown in Figure 4-3.

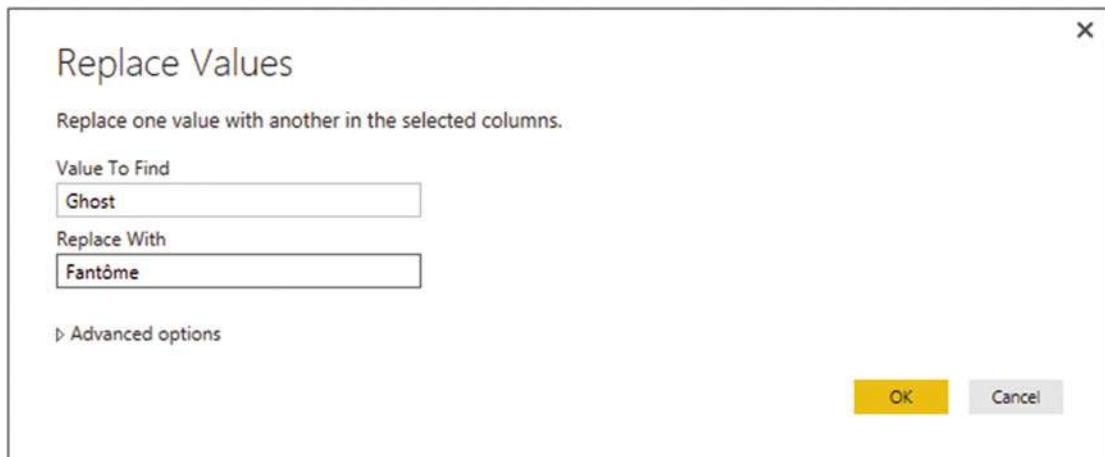


Figure 4-3. The Replace Values dialog

5. Click OK. The data is replaced in the entire column. Replaced Values are added to the Applied Steps list.

I only have a few comments about this technique:

- The Replace Values process searches for every occurrence of the text that you are looking for in each record of the selected columns. It does not look for the entire contents of the cell unless you specifically request this by checking the Match Entire Cell Contents check box in the advanced options.
- If you click on a cell containing the contents that you want to replace (rather than the column title, as we just did), before starting the process, Power BI Desktop automatically places the cell contents in the Replace Values dialog as the value to find.
- You can only replace text in columns that contain text elements. This does not work with columns that are set as a numeric or date data type.
- If you really have to replace parts of a date or figures in a numeric column with other dates or numbers, then you can
 - Convert the column to a text data type
 - Carry out the replace operation
 - Convert the column back to the original data type

The Replace Values dialog also has a few advanced options that you can apply. You can see these if you expand the "Advanced options" item by clicking the triangle to its left. These options are explained in Table 4-2.

Table 4-2. Advanced Replace Options

Option	Description
Match entire cell contents	Only replaces the search value if it makes up the entire contents of the column for a row.
Replace using special characters	Replaces the search value with a nonprinting character.
Tab	Replaces the search value with a Tab character.
Carriage Return	Replaces the search value with a Carriage Return character.
Line Feed	Replaces the search value with a Line Feed character.
Carriage Return and Line Feed	Replaces the search value with a Carriage Return and Line Feed.

Note Replacing words that are subsets of another are dangerous. When replacing any data, make sure that you don't damage elements other than the one you intend to change.

As a final and purely spurious comment, I must add that I would never suggest rebranding a Rolls-Royce, as it would be close to automotive sacrilege.

Transforming Column Contents

Power BI Desktop has a powerful toolbox of automated data transformations that allow you to standardize the contents of a column in several ways. These include

- Setting the capitalization of text columns
- Rounding numeric data or applying math functions
- Extracting date elements such as the year, month, or day (among others) from a date column

Power BI Desktop is very strict about applying transformations to appropriate types of data. This is because transforms are totally dependent on the data type of the selected column. This is yet another confirmation that applying the requisite data type is an operation that should be carried out early in any data transformation process—and certainly *before* transforming the column contents. Remember, you will only be able to select a numeric transformation if the column is a numeric data type, and you will only be able to select a date transformation if the column is a date data type. Equally, the text-based transformations can only be applied to columns that are of the text data type.

Text Transformation

Let's look at a simple transformation operation in action. As an example, I will get Power BI Desktop to convert the Make column into uppercase characters.

1. Still using the file CarSalesFirstDashboard.pbix, click anywhere in the column whose contents you wish to transform (Make, in this case).
2. In the Transform ribbon, click the Format button. A pop-up menu will appear.
3. Select UPPERCASE, as shown in Figure 4-4.

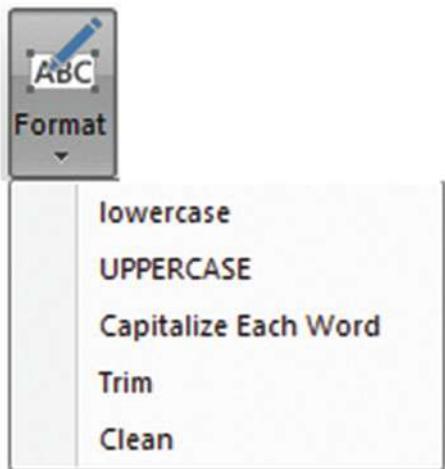


Figure 4-4. Transforming a text to uppercase

The contents of the entire column will be converted to uppercase. Uppercased Text will be added to the Applied Steps list.

As you can see from the menu for the Format button, you have five possible options when formatting (or transforming) text. These options are explained in Table 4-3.

Table 4-3. Text Transformations

Transformation	Description	Applied Steps Definition
Lowercase	Converts all the text to lowercase.	Lowercased Text
Uppercase	Converts all the text to uppercase.	Uppercased Text
Capitalize Each Word	Converts the first letter of each word to a capital.	Capitalized Each Word
Trim	Removes all spaces before and after the text.	Trimmed Text
Clean	Removes any nonprintable characters.	Cleaned Text

Note I realize that Power BI Desktop Query calls text transformations Formatting. Nonetheless, these options are part of the overall data transformation options.

Removing Leading and Trailing Spaces

There will inevitably be occasions when you inherit data that has extra spaces before, after, or before *and* after the data itself. This can be insidious, as it can cause

- Data duplication, because a value with a trailing space is *not* considered identical to the same text without the spaces that follow
- Sort issues, because a leading space causes an element to appear at the *top* of a sorted list
- Grouping errors, because elements with spaces are not part of the same group as elements without spaces

Fortunately, Power BI Desktop Query has a ruthlessly efficient solution to this problem.

1. Click anywhere in the column whose contents you wish to transform (Make, in this case).
2. In the Transform ribbon, click the Format button. A popup menu will appear.
3. Select Trim from the menu.

All superfluous leading and trailing spaces will be removed from the data in the column. This should help with sorting, grouping, and deduplicating records.

Number Transformations

Just as you can transform the contents of text-based columns, you can also apply transformations to numeric values. As an example, suppose that you want to round up all the figures in a column to the nearest whole number.

1. Click anywhere in the column whose contents you wish to transform (TotalDiscount, in this case).
2. In the Transform ribbon, click the Rounding button. A pop-up menu will appear.
3. Select Round Up.

The values in the entire column will be rounded up to the nearest whole number. Rounded Up will be added to the Applied Steps list.

The other possible numeric transformations that are available are described in Table 4-4. Because these numeric transformations use several buttons in the Transform ribbon, I have indicated which button to use to get the desired result.

Table 4-4. Number Transformations

Transformation	Description	Applied Steps Definition
Rounding ► Round Up	Rounds each number to the specified number of decimal places.	Rounded Up
Rounding ► Round Down	Rounds each number up.	Rounded Down
Round...	Rounds each number to the number of decimals that you specify. If you specify a negative number, you round to a given decimal.	Rounded Off
Scientific ► Absolute Value	Makes the number absolute (positive).	
Scientific ► Power ► Square	Returns the square of the number in each cell.	Calculated Square
Scientific ► Power ► Cube	Returns the cube of the number in each cell.	Calculated Cube
Scientific ► Power ► Power	Raises each number to the power that you specify.	Calculated Power

(continued)

Table 4-4. (continued)

Transformation	Description	Applied Steps Definition
Scientific ► Square Root	Returns the square root of the number in each cell.	Square Root
Scientific ► Exponent	Returns the exponent of the number in each cell.	Calculated Exponent
Scientific ► Logarithm ► Base 10	Returns the base 10 logarithm of the number in each cell.	Calculated Base 10 Logarithm
Scientific ► Logarithm ► Natural	Returns the natural logarithm of the number in each cell.	Calculated Natural Logarithm
Scientific ► Factorial	Gives the factorial of numbers in the column.	Calculated Factorial
Trigonometry ► Sine	Gives the sine of the numbers in the column.	Calculated Sine
Trigonometry ► Cosine	Gives the cosine of the numbers in the column.	Calculated Cosine
Trigonometry ► Tangent	Gives the tangent of the numbers in the column.	Calculated Tangent
Trigonometry ► ArcSine	Gives the arcsine of the numbers in the column.	Calculated ArcSine
Trigonometry ► ArcCosine	Gives the arccosine of the numbers in the column.	Calculated ArcCosine
Trigonometry ► ArcTangent	Gives the arctangent of the numbers in the column.	Calculated ArcTangent

Note Power BI Desktop Query will not even let you try to apply numeric transformation to texts or dates. The relevant buttons remain grayed out if you click inside a column of letters or dates.

Calculating Numbers

Power BI Desktop Query can also apply simple arithmetic to the figures in a column. Suppose, for instance, that you want to multiply all the sale prices by 110% as part of your forecasts. This is how you can do just that:

1. Click inside any column of numbers. In this example, I used the column SalePrice.
2. Click the Standard button in the Transform ribbon. The menu will appear.
3. Click Multiply. The Multiply dialog will appear.
4. Enter **1.1** in the Value box. The dialog will look like the one shown in Figure 4-5.

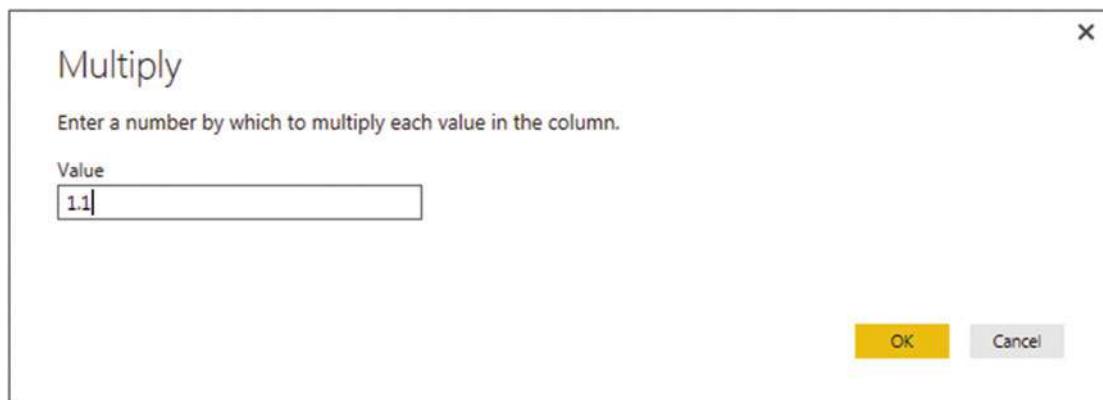


Figure 4-5. Applying a calculation to a column

5. Click OK.

All the numbers in the selected column will be multiplied by 1.1. In other words, they are now 110% of the original value. Table 4-5 describes the possible math operations that you can carry out in Power BI Desktop Query.

Table 4-5. Applying Basic Calculations

Transformation	Description	Applied Steps Definition
Add	Adds a selected value to the numbers in a column.	Added to Column
Multiply	Multiplies the numbers in a column by a selected value.	Multiplied Column
Subtract	Subtracts a selected value from the numbers in a column.	Subtracted from Column
Divide	Divides the numbers in a column by a selected value.	Divided Column
Integer-Divide	Divides the numbers in a column by a selected value and removes any remainder.	Integer-Divided Column
Modulo	Divides the numbers in a column by a selected value and leaves only the remainder.	Calculated Modulo

Note You can also carry out many types of calculations in Power BI Desktop data view and avoid carrying out calculations in the Query Editor. Indeed, many Power BI Desktop purists seem to prefer that anything resembling a calculation should take place inside the data model rather than at the Query stage. As ever, I will let you decide which approach you prefer. Yet I would advise you to read Chapters 7 through 9 to get a clearer understanding of how to add calculated elements to Power BI Desktop using DAX. This is because some transformations need to adjust to the situation (the context) in which they are used, and consequently need to be done using DAX. Be aware that some heavy transforms can slow the reports down if calculated at run time, whereas others can only be effective as part of a well-thought-out calculation process.

Finally, it is important to remember that you are altering the data when you carry out this kind of operation. In the real world, you might be safer duplicating a column before profoundly altering the data it contains.

Date Transformations

Transforming dates follows similar principles to transforming text and numbers. As an example, here is how to isolate the month from a date.

1. Click inside the InvoiceDate column.
2. In the Transform ribbon, click the Date button. The menu will appear.
3. Click Year. The submenu will appear.
4. Select Year. The year part of the date will replace all the dates in the InvoiceDate column.

The other possible date transformations that are possible are given in Table 4-6.

Table 4-6. Date Transformations

Transformation	Description	Applied Steps Definition
Age	Calculates the date and time difference (in days and hours) between the original date and the current local time.	Calculated Age
Date Only	Converts the data to a date without the time element.	Calculated Date
Year ▶ Year	Extracts the year from the date.	Calculated Year
Year ▶ Start of Year	Returns the first day of the year for the date.	Calculated Start of Year
Year ▶ End of Year	Returns the last day of the year for the date.	Calculated End of Year
Month ▶ Month	Extracts the number of the month from the date.	Calculated Month
Month ▶ Start of Month	Returns the first day of the month for the date.	Calculated Start of Month
Month ▶ End of Month	Returns the last day of the month for the date.	Calculated End of Month
Month ▶ Days in Month	Returns the number of days in the month for the date.	Calculated Days in Month
Day ▶ Day	Extracts the day from the date.	Calculated Day
Day ▶ Day of Week	Returns the weekday as a number (Monday is 1, Tuesday is 2, etc.).	Calculated Day of Week
Day ▶ Day of Year	Calculates the number of days since the start of the year for the date.	Calculated Day of Year
Day ▶ Start of Day	Transforms the value to the start of the day for a date and time.	Calculated Start of Day
Day ▶ End of Day	Transforms the value to the end of the day for a date and time.	Calculated End of Day
Quarter ▶ Quarter	Returns the calendar quarter of the year for the date.	Calculated Quarter
Quarter ▶ Start of Quarter	Returns the first date of the calendar quarter of the year for the date.	Calculated Start of Quarter

(continued)

Table 4-6. (continued)

Transformation	Description	Applied Steps Definition
Quarter ▶ End of Quarter	Returns the last date of the calendar quarter of the year for the date.	Calculated End of Quarter
Week ▶ Week of Year	Calculates the number of weeks since the start of the year for the date.	Calculated Week of Year
Week ▶ Week of Month	Calculates the number of weeks since the start of the month for the date.	Calculated Week of Month
Week ▶ Start of Week	Returns the date for the first day of the week (Monday) for the date.	Calculated Start of Week
Week ▶ End of Week	Returns the date for the last day of the week (Monday) for the date.	Calculated End of Week

Time Transformations

You can also transform date/time or time values into their component parts using Power BI Desktop Query. This is extremely similar to how you apply date transformations, but in the interest of completeness, the following explains how to do this.

1. Click inside the InvoiceDate column.
2. In the Transform ribbon, click the Time button. The menu will appear.
3. Click Hour. The hour part of the time will replace all the values in the InvoiceDate column.

The range of Time transformations is given in Table 4-7.

Table 4-7. Time Transformations

Transformation	Description	Applied Steps Definition
Time Only	Isolates the time part of a date and time.	Extracted Time
Local Time	Converts the date/time to local time from date/time and timezone values.	Extracted Local Time
Parse	Extracts the date and/or date/time elements from a text.	Parsed DateTime
Hour	Isolates the hour from a date/time or date value.	Extracted Hour
Minute	Isolates the minute from a date/time or date value.	Extracted Minute
Second	Isolates the second from a date/time or date value.	Extracted Second
Earliest	Returns the earliest time from a date/time or date value.	Calculated Earliest
Latest	Returns the latest time from a date/time or date value.	Calculated Latest

Note In the real world, you could well want to leave a source column intact and apply number or date transformations to a copy of the column. You learn how to apply these transformations to copies of columns in the next chapter.

Duration

If you have values in a column that can be interpreted as a duration (in days, hours, minutes, and seconds), then Power BI Desktop Query can extract the component parts of the duration as a data transformation. For this to work, however, the column *must* be set to the duration data type. This means that the contents of the column have to be interpreted as a duration by Power BI Desktop. Any values that are incompatible with this data type will be set to error values.

If you have duration data, you can extract its component parts like this:

1. Click inside the column.
2. In the Transform ribbon, click the Duration button. The menu will appear.
3. Click Hour. The hour part of the time will replace all the values in the InvoiceDate column.

The range of duration transformations is given in Table 4-8.

Table 4-8. Duration Transformations

Transformation	Description	Applied Steps Definition
Days	Isolates the day element from a duration value.	Extracted Days
Hours	Isolates the hour element from a duration value.	Extracted Hours
Minutes	Isolates the minutes element from a duration value.	Extracted Minutes
Seconds	Isolates the seconds element from a duration value.	Extracted Seconds
Total Days	Displays the duration value as the number of days and a fraction representing hours, minutes, and seconds.	Calculated Total Days
Total Hours	Displays the duration value as the number of hours and a fraction representing minutes and seconds.	Calculated Total Hours
Total Minutes	Displays the duration value as the number of minutes and a fraction representing seconds.	Calculated Total Minutes
Total Seconds	Displays the duration value as the number of seconds and a fraction representing milliseconds.	Calculated Total Seconds
Multiply	Multiplies the duration (and all its component parts) by a value that you enter.	Multiplied Column
Divide	Divides the duration (and all its component parts) by a value that you enter.	Divided Column
Statistics ► Sum	Returns the total for all the duration elements in the column.	Calculated Sum
Statistics ► Minimum	Returns the minimum value of all the duration elements in the column.	Calculated Minimum
Statistics ► Maximum	Returns the maximum value of all the duration elements in the column.	Calculated Maximum
Statistics ► Median	Returns the median value for all the duration elements in the column.	Calculated Median
Statistics ► Average	Returns the average for all the duration elements in the column.	Calculated Average

Note If you multiply or divide a duration, Power BI Desktop Query displays a dialog so that you can enter the value to multiply or divide the duration by.

Filling Down

Imagine a data source where the data has come into Power BI Desktop from a matrix-style structure. The result is that some columns only contain a single example of an element and then a series of empty cells until the next element in the list. If this is difficult to imagine, then suppose that you have loaded the sample file CarMakeAndModelMatrix.xlsx into Power BI Desktop and you are looking at the table shown in Figure 4-6 in the Query Editor.

	Make	Marque	Sales
1	Aston Martin	DB4	391000
2		null DB7	500740
3		null DB9	915070
4		null DBS	230000
5		null Rapide	225000
6		null Vanquish	746500
7		null Vantage	320850
8		null Zagato	178500
9	Bentley	Arnage	44000
10		null Azure	239250
11		null Continental	991250
12		null Turbo R	347500
13	Jaguar	XJ12	303500
14		null XJ6	602000
15		null XK	1092250
16	MGB	GT	315000
17	Rolls Royce	Camargue	810300
18		null Phantom	178500
19		null Silver Ghost	649500
20		null Silver Seraph	288500
21		null Silver Shadow	308500
22		null Wraith	178500
23	Triumph	TR4	140500
24		null TR5	98250
25		null TR7	47750
26	TVR	Cerbera	89250
27		null Tuscan	112250

Figure 4-6. A matrix data table

All these blank cells are a problem since we need a full data table—or rather, they would be, if PowerPivot did not have a really cool way of overcoming this particular difficulty. Do the following to solve this problem.

1. Click in the column that contains the empty cells; make sure that you click where you want to replace the empty cells with the contents of the first non-empty cell above.
2. In the Transform ribbon, click Fill. The menu will appear.
3. Select Down. The blank cells will be replaced by the value in the first non-empty cell above. Filled Down will be added to the Applied Steps list.

The table will now look like Figure 4-7.

	Make	Marque	Sales
1	Aston Martin	DB4	391000
2	Aston Martin	DB7	500740
3	Aston Martin	DB9	915070
4	Aston Martin	DBS	230000
5	Aston Martin	Rapide	225000
6	Aston Martin	Vanquish	746500
7	Aston Martin	Vantage	320850
8	Aston Martin	Zagato	178500
9	Bentley	Arnage	44000
10	Bentley	Azure	239250
11	Bentley	Continental	991250
12	Bentley	Turbo R	347500
13	Jaguar	XJ12	303500
14	Jaguar	XJ6	602000
15	Jaguar	XK	1092250
16	MGB	GT	315000
17	Rolls Royce	Camargue	810300
18	Rolls Royce	Phantom	178500
19	Rolls Royce	Silver Ghost	649500
20	Rolls Royce	Silver Seraph	288500
21	Rolls Royce	Silver Shadow	308500
22	Rolls Royce	Wraith	178500
23	Triumph	TR4	140500
24	Triumph	TR5	98250
25	Triumph	TR7	47750
26	TVR	Cerbera	89250
27	TVR	Tuscan	112250

Figure 4-7. A data table with empty cells replaced by the correct data

Note This technique is built to handle a fairly specific problem and only really works if the imported data is grouped by the column containing the missing elements.

Although rare, you can also use this technique to fill empty cells with the value from below. If you need to do this, just select Fill ► Up from the Transform ribbon. In either case, you need to be aware that the technique is applied to the entire column.

Using the First Row As Headers

Power BI Desktop is very good at guessing if it needs to take the first record of a source dataset and have it function as the column headers. This is fundamental for two reasons:

- You avoid leaving the columns named Column1, Column2, and so on. Leaving them named generically like this would make it needlessly difficult for a user (or even yourself) to understand the data.
- You avoid having a text element (which should be the column title) in a column of figures, which can cause problems later on. This is because a whole column needs to have the same data type for another data type to be applied. Having a header text in the first row prevents this for numeric and data/time data types, for instance.

Yet there could be—albeit rare—occasions when Power BI Desktop guesses incorrectly and assumes that the first record in a dataset is data when it is really the header information. So instead of headers, you have a set of generic column titles such as Column1, Column2, and so forth. Fortunately, correcting this and using the first row as headers is simple.

Click Use First Row As Headers in the Transform section of the Power BI Desktop Query window Home ribbon.

After a few seconds, the first record disappears and the column titles become the elements that were in the first record. On the right now contains a Promoted Headers element, indicating which process has taken place. This step is highlighted.

Note Power BI Desktop is often able to apply this step automatically when the source is a database. It can often correctly guess when the source is a file. However, it cannot always guess accurately, so sometimes you have to intervene. You can see if Power BI Desktop has had to guess this if it has added a Promoted Headers element to the Applied Steps list.

In the rare event that Power BI Desktop gets this operation wrong and presumes that a first row is column titles when it is not, you can reset the titles to be the first row by clicking the tiny triangle to the right of the Use First Row As Headers button. This displays a short menu where you can click the Use Headers As First Row option. The Applied Steps list on the right now contains a Demoted Headers element and the column titles are Column1, Column2, and so forth. You can subsequently rename the columns as you see fit.

Grouping Records

At times, you will need to transform your original data in an extreme way—by grouping the data. This is very different from filtering data, removing duplicates, or cleansing the contents of columns. When you group data, you are altering the structure of the dataset to “roll up” records where you do the following:

- Define the attribute columns that will become the unique elements in the grouped data table
- Specify which aggregations are applied to any numeric columns included in the grouped table

Grouping is frequently an extremely selective operation. This is inevitable, since the more attribute (that is, non-numeric) columns you choose to group on, the more records you are likely to include in the grouped table. However, this will always depend on the particular dataset you are dealing with, and grouping data efficiently is always a matter of flair, practice, and good, old-fashioned trial and error. As an example, you could try out the following:

1. Load the SalesData2012_2013 worksheet from the Excel workbook C:\PowerBiDesktopSamples\SalesData.xlsx into Power BI Desktop.
2. Click Edit Queries to open the Query Editor.
3. Select the following columns (by Ctrl-clicking on the column headers):
 - a. Make
 - b. Model
4. In the Power BI Desktop Home ribbon, click the Group By button. The Group By dialog will appear.
5. In the New Column Name box, enter **TotalSales**.
6. Select Sum as the operation.
7. Choose SalePrice as the source column in the Column pop-up list.
8. Click the plus (+) icon to the right of the new column elements that you just entered and repeat the operation; only this time, use the following:
 - a. New Column Name: AverageCost
 - b. Operation: Average
 - c. Column: CostPrice

The dialog should look like the one in Figure 4-8.

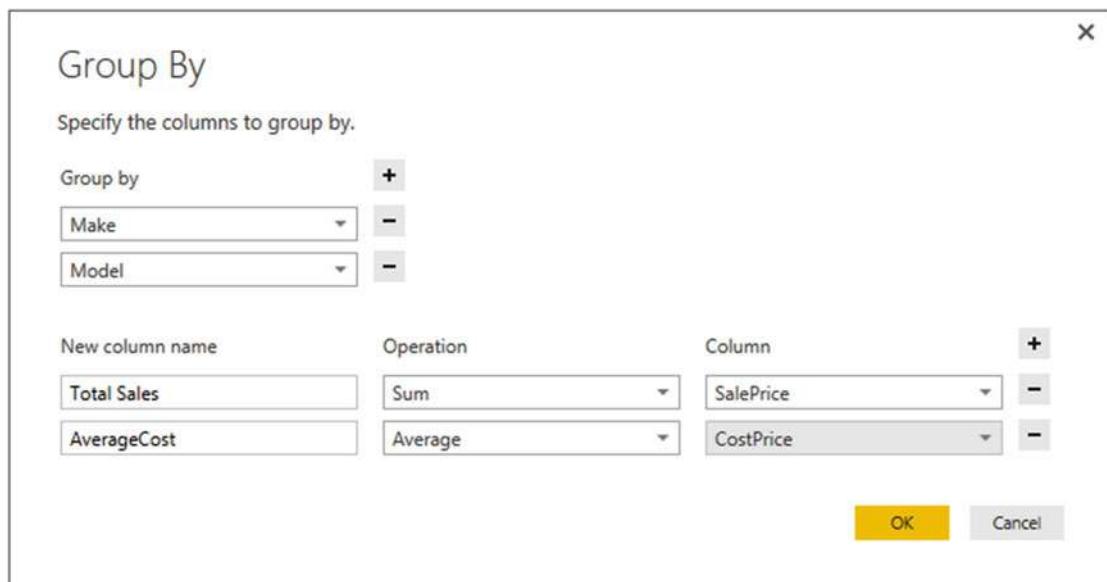


Figure 4-8. The Group By dialog

9. Click OK. All columns, other than those that you specified in the Group By dialog, are removed, and the table is grouped and aggregated, as shown in Figure 4-9. Grouped Rows will be added to the Applied Steps list. I have also sorted the table by the Make and Model columns to make the grouping easier to comprehend.

	Make	Model	Total Sales	AverageCost
1	Rolls Royce	Camargue	4116900	61002.69231
2	Aston Martin	DBS	465500	68000
3	Rolls Royce	Silver Ghost	1315500	75630
4	Aston Martin	DB7	1023480	23703.125
5	Aston Martin	DB9	5423860	59132.96296
6	Aston Martin	DB4	793000	84167.5
7	Aston Martin	Vantage	658200	38750
8	Aston Martin	Vanquish	1506750	89700
9	Aston Martin	Rapide	455500	142500
10	Aston Martin	Zagato	359750	127500
11	Rolls Royce	Wraith	359750	64500
12	Rolls Royce	Silver Shadow	622500	71445
13	Rolls Royce	Silver Seraph	582500	71445
14	Rolls Royce	Phantom	359750	64500
15	Jaguar	XK	4431500	39827.65957
16	Jaguar	XJ6	1239750	32630.76923
17	Jaguar	XJ12	618000	33750
18	Bentley	Continental	3662250	49256.86275
19	Bentley	Arnage	90750	28200
20	Bentley	Azure	489500	28200
21	Bentley	Turbo R	708750	35460
22	TVR	Tuscan	374250	41000
23	TVR	Cerbera	154250	39500
24	MGB	GT	1011000	9500
25	Triumph	TR4	566500	18704.54545
26	Triumph	TR5	207500	19500
27	Triumph	TR7	101000	15250

Figure 4-9. Grouping a dataset

Note You do not have to Ctrl-click to select the grouping columns. You can add them one by one to the Group By dialog by clicking the plus button to the top right of the list of grouped columns. Equally, you can remove grouping columns (or added and aggregated columns) by clicking the minus icon to the right of a column name.

Conclusion

In this short chapter, you learned some essential techniques that you can use to cleanse and standardize datasets. You saw how to round numbers up and down, how to deliver conformed text presentation, and how to remove extraneous spaces from columns of data.

You also saw how to replace values inside columns, as well as ways of applying mathematical, statistical, and trigonometric functions to numbers. Other techniques covered extracting date, time, and duration elements from date/time and duration columns. Finally, you saw how to group and aggregate data that is ready to load into the data model.

It is now time to see how you can join hitherto separate datasets into single queries, and more generally, how to manage the multiple queries that often underpin a data model. This will be the subject of the next chapter.

CHAPTER 5



Data Mashup

In the previous two chapters, you saw how to hone your dataset so that you only used the rows and columns of data that you really need and then how to cleanse and complete the data that they contain. In this chapter, you will learn how to build on these foundations to deliver data that is ready to be molded into a structured and useable data model.

The generic term for this kind of data preparation in Power BI Desktop is *data mashup*. It covers the following:

- *Extending datasets* by adding further columns. These will nearly always be derived from existing columns of data. You could extract years or months from a date column into separate columns, for instance. Alternatively, you could create calculated columns based on existing data—or even take lists of data from a single column and present the data across several columns.
- *Joining datasets* (or queries, if you prefer). This involves taking two queries and linking them so that you display the data from both sources as a single dataset.
- *Pivoting and unpivoting data*. If you need to switch data in rows to display as columns—or vice versa—then you can get the Power BI Desktop Query Editor to help you do exactly this. This means that you can guarantee that the data in all the tables that you are using conforms to a standardized tabular structure.

I want to be clear that separating data preparation into these two apparently contradictory approaches—reducing datasets only to augment them later—is not necessarily the way that you will work in practice. Given the range of features available in Power BI Desktop Query, I have simply tried to apply some structure to the way that they are explained. Hopefully, this will make the data-handling toolkit that is the Power BI Desktop Query Editor a little easier to understand. I imagine that when you are delving into data with Power BI Desktop you will mix and alternate many of the techniques that are outlined in Chapters 2 through 5 in any order. After all, one of the great strengths of Power BI Desktop Query is that it does not impose any strict way of working and lets you experiment freely. So remember that you are at liberty to take any approach you want when transforming source data. The only thing that matters is that it gives you the result that you want.

Creating a good data model can mean using a large range and variety of source data queries. So you will also need to know how to manage the queries that you create to use them efficiently. Consequently, we will end this chapter with a short overview of query management so that you can get an idea of some of the techniques that you might need to keep your queries under control in real-world situations.

The Power BI Desktop View Ribbon

Until now, we have concentrated our attention on the Power BI Desktop Home, Transform, and Add Columns ribbons. This is for the good and simple reason that this is where nearly all the action takes place. There is, however, a fourth Power BI Desktop ribbon—the View ribbon. The buttons that it contains are shown in Figure 5-1 and the options are explained in Table 5-1.



Figure 5-1. The Power BI Desktop View ribbon

Table 5-1. Power BI Desktop View Ribbon Options

Option	Description
Query Settings	Displays or hides the Query Settings pane at the right of the Power BI Desktop window. This includes the Applied Steps list.
Advanced Editor	Displays the Advanced Editor dialog containing all the code for the steps in the query.
Formula Bar	Shows or hides the formula bar.
Monospaced	Displays data in a monospaced (courier) font.

Possibly the only option that is not immediately self-explanatory is the Advanced Editor button. It displays the code for all the transformations in the query as a single block of “M” language script.

■ Tip Personally, I find that the Query Settings pane and the Formula Bar are too vital to be removed from the Power BI Desktop Query window when transforming data. Consequently, I tend to leave them visible. If you need the screen real estate, however, then you can always hide them for a while.

Extending Data

Transforming data does not only consist of reducing it. Sometimes you may have to *extend* the data to make it useable. This normally means adding further columns to a data table. The techniques to do this cover the following:

- Duplicating columns and possibly altering the format of the data in the copied column.
- Extracting part of the data in a column into a new column.

- Parsing the data in a column so that each data element appears in a separate column.
- Merging columns into a new column.

Let's take a look at these techniques and see how you can apply them to prepare data for the reports and dashboards that you will create using Power BI Desktop.

- Adding custom columns that possibly contain calculations or extract part of a column's data into a new column, or even concatenate columns. This can be crucial given that Power BI tries to aggregate anything remotely looking like a number. Consequently, you must ensure that you have provided it with the columns of numbers that you need to work with.
- Adding "index" columns to ensure uniqueness or memorize a sort order.

Duplicating Columns

Sometimes you just need a simple copy of a column, with nothing added and nothing taken away. This is where the Duplicate Column button comes into play.

1. Load the C:\PowerBiDesktopSamples\CarSalesTables.pbix sample file.
2. Open the Power BI Desktop Query Editor and click the Stock query in the Queries pane on the left.
3. Click inside (or on the title of) the column that you want to duplicate. I will use the Make column in this example.
4. In the Add Column ribbon, click the Duplicate Column button. After a few seconds, a copy of the column is created at the right of the existing table. Duplicated Column will appear in the Applied Steps list.
5. Scroll to the right of the table and rename the existing column; it is currently named Make-Copy.

Note The duplicate column is named Original Column Name-Copy. I find that it helps to rename copies of columns sooner rather than later in a data mashup process.

Splitting Columns

Sometimes a source column contains data that you really need to break up into smaller pieces across two or more columns. The following are classic cases where this happens:

- A column contains a list of elements, separated by a specific character (known as a *delimiter*).
- A column contains a list of elements, but the elements can be divided at specific places in the column.
- A column contains a concatenated text that needs to be split into its composite elements (a bank account number or a Social Security number are examples of this).

The following short sections explain how to handle such eventualities.

Splitting Column by a Delimiter

Here is another requirement that you may encounter occasionally. The data that has been imported has a column that needs to be further split into multiple columns. Imagine a text file where columns are separated by semicolons, and these subdivisions each contain a column that holds a comma-separated list of elements. Once you have imported the file, you then need to further separate the contents of this column that uses a different delimiter.

Here is what you can do to split the data from one column over several columns:

1. Load the C:\PowerBiDesktopSamples\CarSalesData.pibx sample file.
2. Open the Power BI Desktop Query Editor, and click the *Invoices* query in the Queries pane on the left.
3. Click inside the InvoiceNumber column.
4. In the Transform ribbon, click the Split Column button. The menu will appear.
5. Select By Delimiter in the pop-up menu. The Split A Column By Delimiter dialog appears.
6. Select Custom from the list of available options in the Select Or Enter Delimiter pop-up. A new box will appear in the dialog.
7. Enter a hyphen (-) in the new box.
8. Click At Each Occurrence Of The Delimiter. The dialog should look like Figure 5-2.

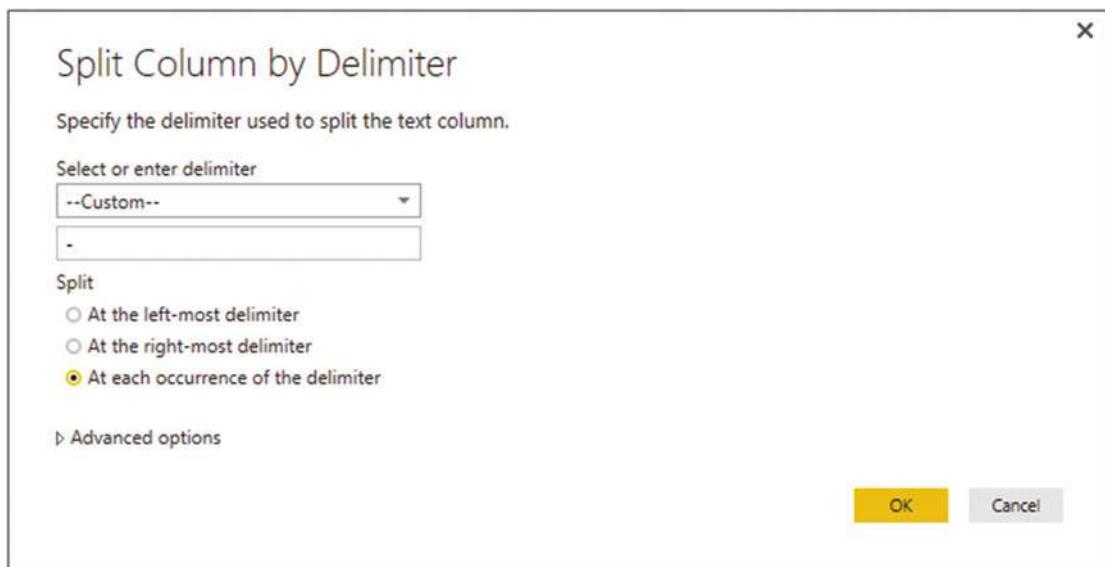


Figure 5-2. Splitting a column using a delimiter

9. Click OK. Split Column by Delimiter will appear in the Applied Steps list.

The initial column is replaced and all the new columns are named InvoiceNumber.1, InvoiceNumber.2, and so forth. As many additional columns as there are delimiters are created; each is named (Column.n) and is sequentially numbered. The result of this operation looks like Figure 5-3.

InvoiceNumber.1	InvoiceNumber.2	InvoiceNumber.3	InvoiceNumber.4	InvoiceNumber.5
8B3D7F83	F42C	4523	A737	CDCBF7705B77
139BEEEF	FF32	4BE9	9EF1	819AC888B85C
D35D72CD	5FF3	4701	A6D1	265A4F4E7CD5
2ABAA300	E2A5	4E37	BFCA	7B80ED88A2BD
A1C2D846	EC39	46FA	A399	0C194AAD4DC8
1B8F325A	CC41	4BA6	A486	9D44962E40A3
F1B566F0	D137	4810	B449	575438F3F392
ADFFAC9E	DFF3	4BAB	9EC4	DEE9A2869350
15A3BC61	82BD	4CCD	8FOB	49EEE59AF4B7

Figure 5-3. The results of splitting a column

This particular process has several options, and their consequences can be fairly far-reaching as far as the data is concerned. Table 5-2 contains a description of the available options followed by a few comments.

Table 5-2. Delimiter Split Options

Option	Description
Colon	Uses the colon (:) as the delimiter.
Comma	Uses the comma (,) as the delimiter.
Equals Sign	Uses the equals sign (=) as the delimiter.
Semi-Colon	Uses the semicolon (;) as the delimiter.
Space	Uses the space () as the delimiter.
Tab	Uses the Tab character as the delimiter.
Custom	Lets you enter a custom delimiter.
At the Left-Most Delimiter	Splits the column once only at the first occurrence of the delimiter.
At the Right-Most Delimiter	Splits the column once only at the last occurrence of the delimiter.
At Each Occurrence of the Delimiter	Splits the column into as many columns as there are delimiters.
Advanced Options ▶ Number of Columns to Split Into	Allows you to set a maximum number of columns into which the data is split in chunks of the given number of characters. Any extra columns are placed in the rightmost column.
Advanced Options ▶ Quote Style ▶ CSV	Separators inside a text that is contained in double quotes are not used to split the text into columns.
Advanced Options ▶ Quote Style ▶ None	Quotes are not taken into consideration when splitting text into columns.

Splitting Columns by Number of Characters

Another variant on this theme is when text in each column is a fixed number of characters and needs to be broken down into constituent parts at specific intervals. Suppose, for instance, that you have a field where each group of (a certain number of) characters has a specific meaning, and you want to break it into multiple columns. Alternatively, suppose you want to extract the leftmost or rightmost n characters and leave the rest. A bank account or Social Security number are examples of this. This is where splitting a column by the number of characters can come in useful. As the principle is very similar to the process that we just saw, I will not repeat the whole thing again. All you have to do is choose the By Number Of Characters menu option at step 5 in the previous exercise. Options for this type of operation are given in Table 5-3.

Table 5-3. Options When Splitting a Column by Number of Characters

Option	Description
Number Of Characters	Lets you define the number of characters of data before splitting the column.
Once, As Far Left As Possible	Splits the column once only at the given number of characters in from the left.
Once, As Far Right As Possible	Splits the column once only at the given number of characters in from the right.
Repeatedly	Splits the column as many times as necessary to cut it into segments every defined number of characters.
Advanced Options ► Number Of Columns To Split Into	Allows you to set a maximum number of columns into which the data is split in chunks of the given number of characters. Any extra columns are placed in the rightmost column.

There are a couple of things to note when splitting columns:

- When splitting by a delimiter, Power BI Desktop makes a good attempt at guessing the maximum number of columns into which the source column must be split. If it gets this wrong (and you can see what its guesstimate is if you expand the Advanced Options box), you can override the number here.
- If you select a Custom Delimiter, Power BI Desktop displays a new box in the dialog where you can enter a specific delimiter.
- Not every record has to have the same number of delimiters. Power BI Desktop simply leave the rightmost column(s) blank if there are fewer split elements for a row.

Note You can only split columns if they are text data. The Split Column button remains grayed out if your intention is to try to split a date or numeric column.

Merging Columns

You may be feeling a certain sense of *déjà-vu* when you read the title of this section. After all, we saw how to merge columns (that is, how to fuse the data from several columns into a single, wider column) in a previous chapter, did we not?

Yes, we did indeed. However, this is not the only time in this chapter that you will see something that you have tried previously. This is because Power BI Desktop Query repeats several of the options that are in the Transform menu in the Add Column menu. While these functions all work in much the same way there is one essential difference. If you select an option from the Transform menu then the column(s) that you selected is *modified*. If you select a similar option from the Add Column menu then the original column(s) will not be altered, but a *new column* is added containing the results of the data transformation.

Merging columns is a case in point. Now, as I went into detail as to how to execute this kind of data transformation in the previous chapter I will not describe it all over again here. Suffice it to say, if you Ctrl-click the headings of two or more columns and then click Merge Columns in the Add Column ribbon, you will still see the data from the selected columns concatenated into a single column. However, this time the original columns *remain* in the dataset. The new column is named *Merged*, exactly as was the case for the first of the columns that you selected when merging columns using the Transform ribbon.

The following are other functions that can either overwrite the data in existing columns *or* display the result as a new column:

- *Format*: Trims or changes the capitalization of text.
- *Extract*: Takes part of a column and creates another column from this data.
- *Parse*: Adds a column containing the source column data as JSON or XML strings.
- *Statistics*: Creates a new column of aggregated numeric values.
- *Standard*: Creates a new column of calculated numeric values.
- *Scientific*: Creates a new column by applying certain kinds of math operations to the values in a column.
- *Trigonometry*: Creates a new column by applying certain kinds of trigonometric operations to the values in a column.
- *Rounding*: Creates a new column by rounding the values in a column.
- *Information*: Creates a new column indicating arithmetical information about the values in a column.
- *Date*: Creates a new column by extracting date elements from the values in a date column.
- *Time*: Creates a new column by extracting time elements from the values in a time or datetime column.
- *Duration*: Creates a new column by calculating the duration between two dates or date/times.

When transforming data, the art is to decide whether you want or need to keep the original column before applying one of these functions. Yet, once again, it is not really fundamental if you later decide that you made an incorrect decision as you can always backtrack. Alternatively, you can always decide to insert new columns as a matter of principle and delete any columns that you really do not need at a later stage in the data transformation process.

Custom Columns

Another way to extend the original data table is to add more columns. Although these are known as *custom columns* in Power BI Desktop, they are also known more generically as *derived columns* or *calculated columns*. Although they can do many things, their essential role is to

- Concatenate (or join, if you prefer) existing columns.
- Add calculations to the data table.
- Extract a specific part of a column.
- Add flags to the table based on existing data.

The best way to understand these columns is probably to see them in action. You can then extend these principles in your own processes.

Initially, let's perform a column join and create a column named Vehicle, which concatenates the Make and Model columns with a space in between.

1. Load the C:\PowerBiDesktopSamples\CarSalesFirstDashboard.pibx sample file.
2. Click Edit Queries.
3. Select the CarSalesData query.
4. In the Add Column ribbon, click Add Custom Column. The Add Custom Column dialog is displayed.
5. Click the Make column in the column list on the right, then click the Insert button; =[Make] will appear in the Custom Column Formula box at the left of the dialog.
6. Enter & “ “ & in the Custom Column Formula box after =[Make].
7. Click the Model column in the column list on the right, and then click the Insert button. The dialog will look like Figure 5-4.

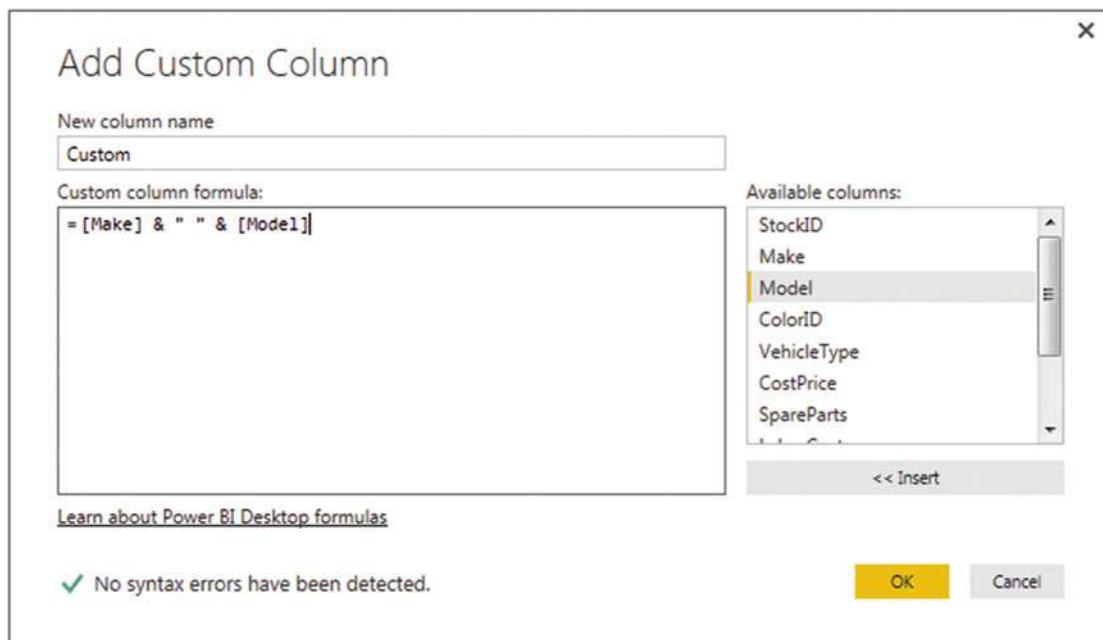


Figure 5-4. The Add Custom Column dialog

8. Click inside the “New column name” box and enter a name for the column. I call it CarType.
9. Click OK. The new column is added to the right of the data table; it contains the results of the formula. InsertedColumn appears in the Applied Steps list.

You can always double-click a column to insert it into the Custom Column Formula box if you prefer. To remove a column, simply delete the column name (including the square brackets) in the Custom Column Formula box.

Note You must always enclose a column name in square brackets.

Rather than take you step by step through other examples, I prefer to show you some of the formulas that you can use to calculate columns and extract data into a new column. These code snippets are given in Table 5-4. As an Excel user, you can probably see a distinct similarity with how you build formulas in Excel and PowerPivot, except that here (as in PowerPivot) you use column names rather than cell references.

Table 5-4. Custom Column Code Examples

Output	Code Snippet	Description
Column Calculations	= [SalePrice]-[CostPrice]	Subtracts the Cost Price from the Sale Price to give the Gross Margin
Column Arithmetic	=[SalePrice] * 1.2	Adds the UK sales tax (20%) to the Net Sale Price
Left	Text.Start([Make],3)	Returns the first three characters from the Make column
Right	Text.End([Make],3)	Returns the last three characters from the Make column
Up to a specific character	Text.Start([Make], Text.PositionOf([Make], " "))	Returns the leftmost characters up to the first space

If you look ahead to Chapter 7, then you are probably wondering why you carry out operations like this in Power BI Desktop Query when you can do virtually the same thing in the data model. Well, it is true that there is some overlap; so you have the choice of which to use. You can perform certain operations at multiple stages in the data preparation and analysis process. It all depends on how you are using the data and with what tool you are carrying out the analyses.

The last three examples in Table 5-4 probably seem a little abstruse for the moment. This is because they are examples of how to use the Power BI data transformation language. This language is normally referred to as “M”. Because an in-depth description of this language is beyond the scope of this book, I suggest that you refer to the Microsoft documentation available at <https://msdn.microsoft.com/en-us/library/mt211003.aspx> if you require further information.

Index Columns

An index column is a new column that numbers every record in the table sequentially. This numbering scheme applies to the table, because it is currently sorted and begins at zero. There are many situations where an index column can be useful. The following are some examples:

- Reapply a previous sort order.
- Create a unique reference for every record.
- Prepare a recordset for use as a dimension table in a Power BI Desktop data model. In cases like this, the index column becomes what dimensional modelers call a *surrogate key*.

This list is not intended to be exhaustive in any way; you will almost certainly find other uses as you work with Power BI Desktop. Whatever the need, here is how to add an Index column:

1. In the Add Column ribbon, click Add Index Column. The new, sequentially numbered column is added at the right of the table, and Added Index is added to the Applied Steps list.
2. Scroll to the right of the table and rename the index column; it is currently named Index.

You have a fairly free hand when it comes to deciding how to begin numbering and index column. The choices are as follows:

- Start at 0 and increment by a value of 1 for each row.
- Start at 1 and increment by a value of 1 for each row.
- Start at any number and increase by any number.

As you saw in step 1, the default is for Power BI Desktop query to begin numbering rows at 0. However, you can choose another option by clicking the small triangle to the right of the Add Index Column button. This displays a menu with the three options outlined.

Selecting the third option, Custom, displays the dialog that you see in Figure 5-5.



Figure 5-5. The Add Index Column dialog

This dialog lets you specify the start number for the first row in the dataset as well as the increment that is added for each record.

Note It is a good idea to sort the dataset before adding an index column if you have a specific need for the records to be numbered in a certain order. Indeed, you can add multiple indexes to varied columns in differing sort orders. This lets you switch sort orders by sorting on different index columns.

Merging Data

Until now, we have treated each individual query as if it existed in isolation. The reality, of course, is that you will frequently be required to use the output of one query in conjunction with the output of another to join data from different tables in various ways. Assuming that the results of one query share a common field with another query, you can “join” queries into a single data table. Power BI Desktop calls this a merge operation, and it enables you, among other things, to

- Look up data elements in another “reference” table to add lookup data. This could be where you want to add a client name where only the client code exists in your main table, for instance.
- Aggregate data from a “detail” table (such as invoice lines) into a higher-grained table, such as a table of invoices.

Here, again, the process is not difficult. The only fundamental factor is that the two tables, or queries, that you are merging must have a shared field or fields that enable the two tables to match records coherently. Let’s look at a couple of examples.

Adding Data

First, let’s try looking up extra data that we will add to a query.

1. In a new, empty Power BI Desktop file, load all the worksheets in the SalesData.xlsx Excel file.
2. Click the query named Sales in the Queries pane of the Power BI Desktop Query window.
3. Click the Merge Queries button in the Home ribbon. The Merge dialog will appear.
4. In the upper part of the dialog—where an overview of the output from the current query is displayed—click the ClientName column title. This column is highlighted.
5. In the pop-up under the upper table, select the Clients query. The output from this query will appear in the lower part of the dialog.
6. In the lower table, select the column title for the column—the join column—that maps to the column that you selected in step 4. This will also be the ClientName column. This column is then selected in the lower table. You may be asked to set privacy levels for the data sources. If this is the case, set them to Public.
7. Select the Join Kind Inner (only matching rows) from the pop-up. The dialog will look like Figure 5-6.

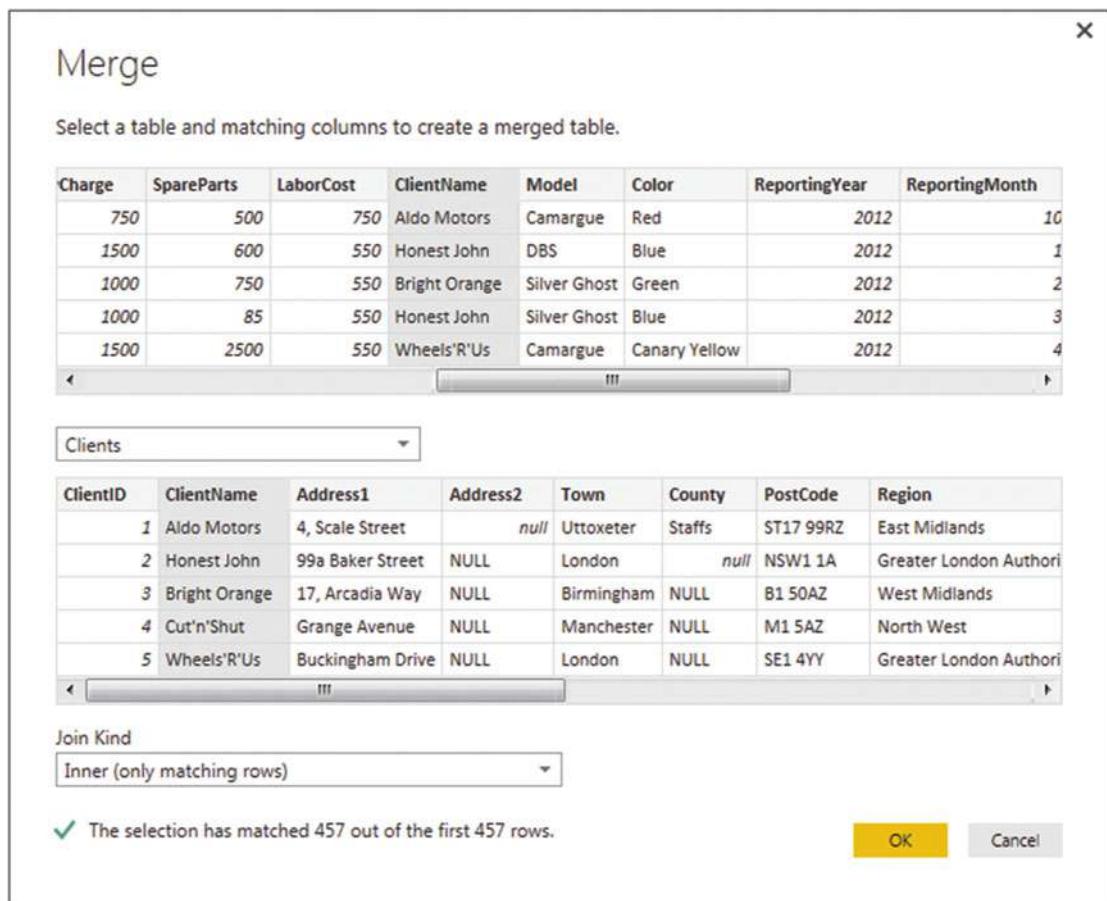


Figure 5-6. The Merge dialog

8. Click OK. A new column is added to the right of the existing data table.
9. Scroll to the right of the existing data table and click the Expand icon to the right of the column name (it has probably been named New Column and every row contains the word *Table*). The pop-up list of all the available fields in this data table (or query, if you prefer) is displayed, as shown in Figure 5-7.

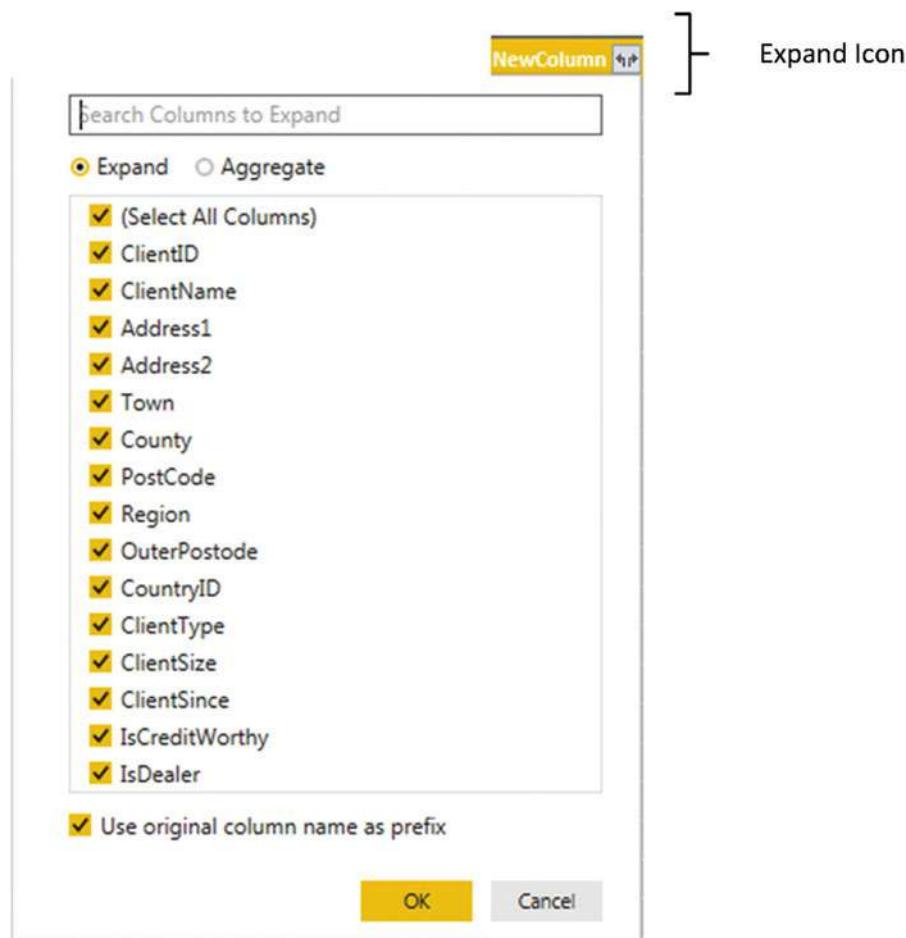


Figure 5-7. The fields available in a joined query

10. Ensure that the Expand radio button is selected.
11. Clear the selection of all the columns by unchecking the (Select All Columns) check box.
12. Select the following columns:
 - a. ClientName
 - b. ClientSize
 - c. ClientSince
13. Click OK. The selected columns from the linked table are merged into the main table, and the link to the reference table (New Column) is removed.
14. Rename the columns that have been added, and apply and close the query.

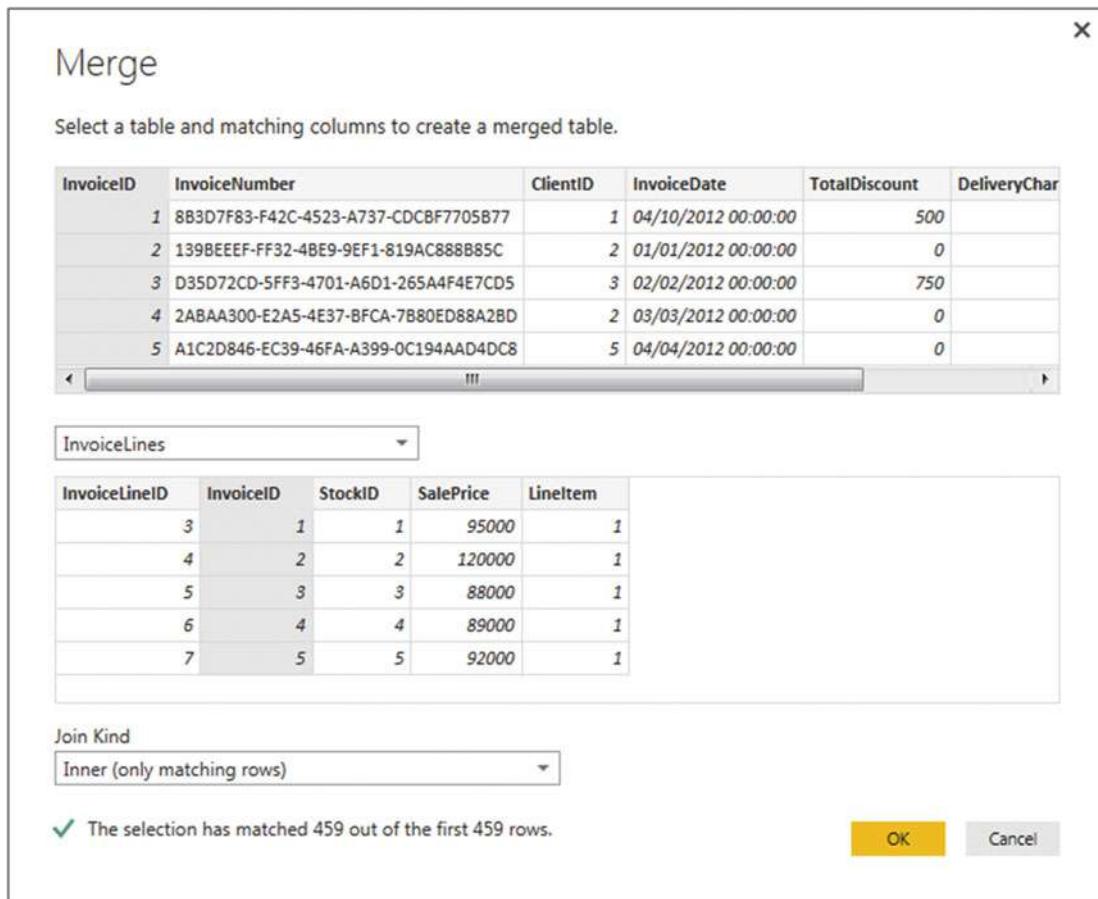
You now have a single table of data that contains data from two linked data sources. Reprocessing the Sales query will also reprocess the dependent *clients* query and result in the latest version of the data being reloaded.

Note You probably noticed that the Merge dialog indicated how many matching records there were in the two queries. This can be a useful indication that you have selected the correct column(s) to join the two queries.

Aggregating Data During a Merge Operation

If you are not just looking up reference data but need to aggregate data from a separate table and then add the results to the current query, then the process is largely similar. This second approach, however, is designed to suit another completely different requirement. Previously, you saw the case where the current query had many records that mapped to a *single* record in the lookup table. This second approach is for when your current (or main) query has a single record where there are *multiple* linked records in the second query. Consequently, you need to aggregate the data in the second table to bring the data across into the first table. Here is a simple example, using some of the sample data from the C:\PowerBiDesktopSamples folder.

1. Find the InvoicesAndInvoiceLines.xlsx Excel source file in the C:\PowerBiDesktopSamples folder. Load the two worksheets it contains (Invoices and InvoiceLines) into two queries in the Power BI Desktop Query Editor.
2. Click the query named Invoices in the Queries pane on the left.
3. In the Home ribbon, click the Merge Queries button. The Merge dialog will open. You will see some of the data from the Invoices dataset in the upper part of the dialog.
4. Click anywhere inside the InvoiceID column. This column is selected.
5. In the pop-up, select the InvoiceLines query. You will see some of the data from the Invoices dataset in the lower part of the dialog.
6. Click anywhere inside the InvoiceID column for the lower table. This column is selected.
7. Select Inner (only matching rows) from the Join Kind pop-up menu. The dialog will look like Figure 5-8.

**Figure 5-8.** The Merge dialog when aggregating data

8. Click OK.
9. Scroll to the right of the existing data table. You will see a new column (named NewColumn) that contains the word *Table* in every cell. This column will look something like Figure 5-9.

The screenshot shows a data grid with three columns. The first two columns are dropdown menus labeled 'DeliveryCharge' and 'InvoiceDateKey'. The third column is a dropdown menu labeled 'NewColumn' with a yellow background. Below the grid, there is a vertical scroll bar.

DeliveryCharge	InvoiceDateKey	NewColumn
750	20121004	Table
1500	20120101	Table
1000	20120202	Table
1000	20120303	Table
1500	20120404	Table
1000	20120504	Table
500	20120604	Table
1000	20120704	Table
1000	20120804	Table
1000	20120904	Table

Figure 5-9. A merged column

- Click the Expand icon to the right of the new column title (the two arrows facing left and right). The pop-up list of all the available fields in the InvoiceLines query is displayed.
- Select the Aggregate radio button. The dialog will look like Figure 5-10.

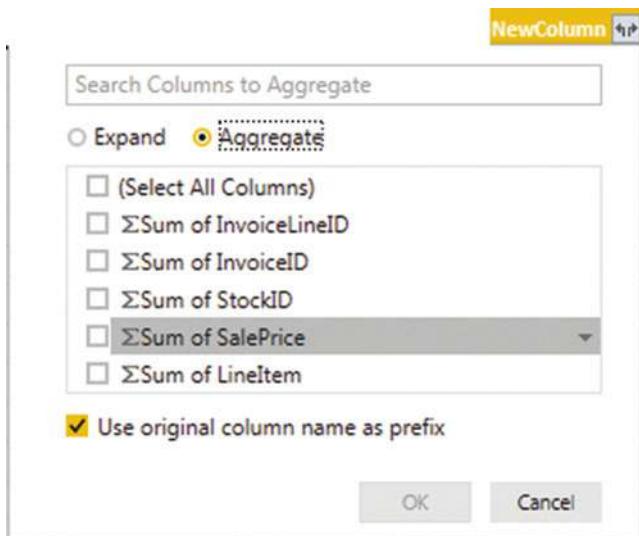


Figure 5-10. The available fields from a merged dataset

- Select the Sum Of SalePrice field.
- Uncheck "Use original column name as prefix" and click OK.

Power BI Desktop will add up the total sale price for each invoice and add this as a new column. Naturally, you can choose the type of aggregation that you wish to apply (before clicking OK), if the sum is not what you want. To do this, place the cursor over the column that you want to aggregate (see step 11 in the preceding exercise) and click the pop-up menu at the right of the field name. Power BI Desktop will suggest the following options:

- Sum
- Average
- Minimum
- Maximum
- Count (all)
- Count (Not Blank)

An example of this is shown in Figure 5-11.

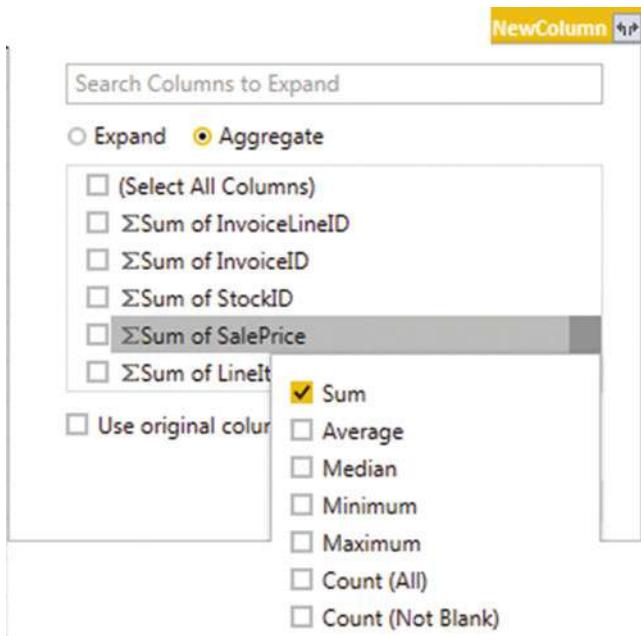


Figure 5-11. Aggregation options when merging datasets

The merge process that you have just seen, while not complex in itself, suddenly opens up many new horizons. It means that you can now create multiple separate queries that you can then use together to expand your data in ways that allow you to prepare quite complex datasets.

Here are a couple of comments I need to make about the merge operation:

- Only queries that have been previously created in the Power BI Desktop Query window can be used when merging datasets. So remember to get all the datasets that you require before attempting a merge operation.
- Refreshing a query will cause any other queries that are upstream of this query to be refreshed also. This way you will always get the most up-to-date data from all the queries in the process.

Types of Join

When merging queries—either to join data or to aggregate values—you were faced with a choice when it came to how the two queries could be linked. The choice of join can have a profound effect on the resulting dataset. Consequently, it is important to understand the six join types that are available. These are described in Table 5-5.

Table 5-5. Join Types

Join Type	Explanation
Left Outer	Keeps all records in the upper dataset in the Merge dialog (the dataset that was active when you began the Merge operation). Any matching rows (those that share common values in the join columns) from the second dataset are kept. All other rows from the second dataset are discarded.
Right Outer	Keeps all records in the lower dataset in the Merge dialog (the dataset that was not active when you began the Merge operation). Any matching rows (those that share common values in the join columns) from the upper dataset are kept. All other rows from the upper dataset (the dataset that was active when you began the Merge operation) are discarded.
Full Outer	All rows from both queries are retained in the resulting dataset. Any records that do not share common values in the join field(s) contain blanks in certain columns.
Inner	Only joins queries where there is an exact match on the column(s) that are selected for the join. Any rows from either query that do not share common values in the join column(s) are discarded.
Left Anti	Keeps only rows from the upper (first) query.
Right Anti	Keeps only rows from the lower (second) query.

When you are expanding the column that is the link to a merged dataset, you have a couple of useful options that are worth knowing about:

- Use original column name as prefix
- Search columns to expand

Use the Original Column Name As the Prefix

You will probably find that some columns from joined queries can have the same names in both source datasets. It follows that you need to identify which column came from which dataset. If you leave the check box selected for the “Use original column name as prefix” merge option (which is the default), any merged columns will include the source query name to help you identify the data more accurately.

If you find that these longer column names only get in the way, you can unselect this check box. This will leave the added columns from the second query with their original names. However, because Power BI Desktop cannot accept duplicate column names, any new columns will have .1, .2, and so forth, added to the column name.

Search Columns to Expand

If you are merging a query with a second query that contains a large number of columns then it can be laborious to search for the columns that you want to include. To narrow your search you can enter a few characters from the column that you are looking for. The more characters you type, the fewer matching columns are displayed in the Expansion pop-up dialog.

Note When you use any of the *outer* joins you are keeping records that do not have any corresponding records in the second query. Consequently, the resulting dataset contains empty values for some of the columns.

Joining on Multiple Columns

In the examples so far, you only joined queries on a single column. While this may be possible if you are looking at data that comes from a clearly structured source (such as a relational database) you may need to extend the principle when joining queries from diverse sources. Fortunately, Power BI Desktop allows you to join queries on multiple columns when the need arises.

As an example of this, the sample data contains a file that I have prepared as an example of how to join queries on more than one column. This sample file contains data from the sources that you saw in Chapter 2. However, they have been modeled as a data warehouse star schema. To complete the model, you need to join a dimension named Geography to a fact table named Sales so that you can add the field GeographySK to the fact table. However, the Sales table and the Geography table share three fields (Country, Region, and Town) that must correspond for the queries to be joined. The following explains how to perform a join using multiple fields.

1. Open the C:\PowerBiDesktopSamples\StarSchema.xlsx Excel file and load the two worksheets that it contains into a new Power BI Desktop Query window.
2. In the Home ribbon, click the Edit Queries button.
3. Select the Sales query from the list of existing queries on the left of the Power BI Desktop Query window.
4. In the Home ribbon, click the Merge Queries button. The Merge dialog will appear.
5. In the pop-up list of queries, select Geography as the second query to join to the first (upper) query.
6. Select Inner (only matching rows) as the join kind.

7. In the upper list of fields (taken from the Sales table), Ctrl-click the fields Country, Region, and Town, in this order. A small number will appear to the right of each column header indicating the order that you selected the columns.
8. In the lower list of fields (taken from the Sales table) Ctrl-click in the fields Country, Region, and Town, in this order. A small number will appear to the right of each column header indicating the order that you selected the columns.
9. Verify that you have a reasonable number of matching rows in the information message at the bottom of the dialog. The dialog will look like Figure 5-12.

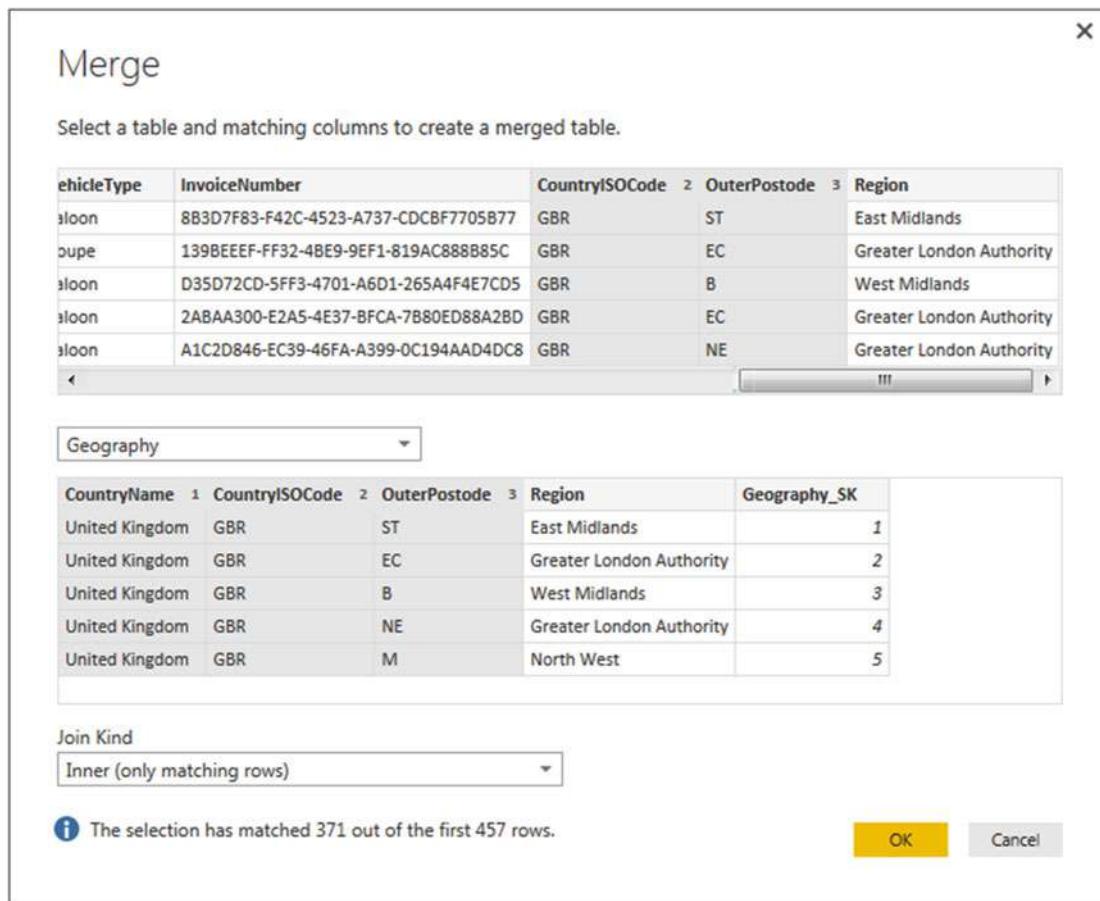


Figure 5-12. Joining queries using multiple columns

10. Click OK.

You can then continue with the data mashup. In this example, that would be adding the GeographySK field to the Sales query and then removing the Country, Region, and Town fields from the Sales query.

There is no real limit to the number of columns that can be used when joining queries. It will depend entirely on the shape of the source data.

■ Tip In the current version of Power BI Desktop Query, there is a small bug. You cannot click the title of any column in the second (lower) table in the Merge dialog, but only inside the columns when selecting join columns. Hopefully, this is fixed by the time that this book is published.

Preparing Datasets for Joins

You could have to carry out a little preparatory work on real-world datasets before joining queries. More specifically, any columns that you join have to be the same basic data type. Put simply, you need to join text-based columns to other text columns, number columns to numbers and dates to dates. If the columns are *not* the same data type, you receive a warning message when you try to join the columns in the Merge dialog.

Consequently, it is nearly always a good idea to take a look at the columns that you will use to join queries *before* you start the merge operation itself. Remember that data types do not have to be identical, just similar. So a decimal number type can map to a whole number, for instance.

You might also have to cleanse the data in the columns that are used for joins before attempting to merge queries. This could involve the following:

- Removing trailing or leading spaces in text-based columns
- Isolating part of a column (either in the original column or as a new column) to use in a join
- Verifying that appropriate data types are used in join columns

Correct and Incorrect Joins

Merging queries is the one data mashup operation that is often easier in theory than in practice, unfortunately. If the source queries were based on tables in a relational or even dimensional database them joining them could be relatively easy, as a data architect will (hopefully) have designed the database tables to allow for them to be joined. However, if you are joining two completely independent queries, then you could face several major issues:

- The columns do not map
- The columns map, but the result is a massive table with duplicate records

Let's take a look at these possible problems.

The Columns Do Not Map

If the columns do not map (that is, you have joined the data but get no resulting records) then you need to take a close look at the data in the columns that you are using to establish the join. The questions you need to ask are:

- Are the values in the two queries the same data type?
- Do the values really map—or are they different?
- Are you using the correct columns?
- Are you using too many columns and so specifying data that is not in both queries?

The Columns Map, but the Result Is a Massive Table with Duplicate Records

Joining queries depends on isolating *unique* data in both source queries. Sometimes a single column does not contain enough information to establish a unique reference that can uniquely identify a row in the query.

In these cases, you need to use two or more columns to join queries—or else rows will be duplicated in the result. Therefore, once again, you need to look carefully at the data and decide on the minimum number of columns that you can use to join queries correctly.

Examining Joined Data

Joining data tables is not always easy. Neither is deciding if the outcome of a merge operation will produce the result that you expect.

So Power BI Desktop Query includes a solution to these kinds of dilemma. It can help you more clearly see what a join has done. More specifically, it can show you or each record in the first query exactly which rows are joined from the second query.

Do the following to see this in action:

1. Carry out steps 1 through 8 in the Merging Data section.
2. Scroll to the right of the data table. You will see the new column named NewColumn (as shown in Figure 5-7).
3. Click to the *right* of the word *Table* in the row where you want to see the joined data. Note that you must *not* click the word *Table*. A second table will appear under the main query's data table containing the data from the second query that is joined for this particular row. Figure 5-13 shows an example of this.

The screenshot shows a Power BI Desktop Query interface. The main data table has columns: ClientID, InvoiceDate, TotalDiscount, DeliveryCharge, InvoiceDateKey, and NewColumn. The NewColumn column contains the word 'Table' for most rows, except for one row where it contains a small secondary table. This secondary table has columns: InvoiceLineID, InvoiceID, StockID, SalePrice, and LineItem. The value in the NewColumn cell for the highlighted row is a screenshot of this secondary table, showing data for LineItem 1 with values 122, 117, 118, 25250, and 1 respectively.

Figure 5-13. Joined data

This technique is as simple as it is useful. There are nonetheless a few comments that need to be made.

- You can resize the lower table (and consequently display more or less data from the second joined table) by dragging the bottom border of the top data table up or down.
- Clicking to the right of the word *Table* in the NewColumn will enable the Expand and Aggregate buttons in the Transform ribbon.
- Clicking the word *Table* in the NewColumn adds a new step to the query that replaces the source data with the linked data. You can also do this by right-clicking inside the NewColumn and selecting Drill Down.

Note Drilling down into the merged table in effect limits the query to the row(s) of the subtable. Consequently, you have to delete this step if you want to access all the data in the merged tables.

The Expand and Aggregate Buttons

Power BI Desktop Query offers an alternative to using the NewColumn pop-up menu to expand or aggregate data when merging queries. If you have clicked to the right of the word *Table* in the NewColumn, you can then click the newly activated Expand or Aggregate buttons to display the Expand or Aggregate dialogs in the Transform ribbon.

The only real difference between the dialogs and the pop-ups is that the Expand dialog also has an option where you can add a new column prefix that you choose for the additional columns from the second query. You can see this in Figure 5-14.

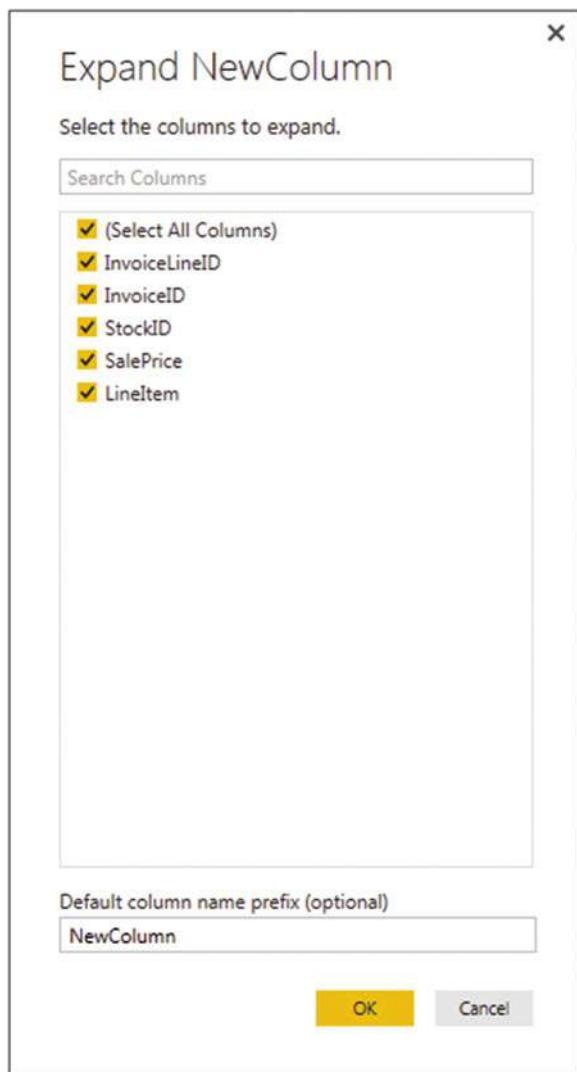


Figure 5-14. The Expand New Column dialog

Appending Data

Not all source data is delivered in its entirety in a single file or as a single database table. You may be given access to two or more tables or files that have to be loaded into a single table in Excel or PowerPivot. In some cases, you might find yourself faced with hundreds of files—all text, CSV, or Excel format—and the requirement to load them all. Well, Power BI Desktop can handle these eventualities, too.

Adding the Contents of One Query to Another

In the simplest case, you could have two data sources that are structurally identical (that is, they have the same columns in the same order), and all that you have to do is add one to another to end up with a query that outputs the amalgamated content of the two sources. This is called *appending data*, and it is easy, provided that the two data sources have *identical* structures; this means

- They have the same number of columns.
- The columns are in the same order.
- The data types are identical for each column.
- The columns have the same names.

As long as all these conditions are met, you can append the output of queries (which Power BI Desktop calls *Tables*) onto another. The queries do not have to have data that comes from identical source types, so you can append the output from a CSV file to data that comes from an Oracle database, for instance. As an example, we will take two text files and use them to create one single output.

1. Create queries to load each of the following text files into Power BI Desktop—without the final load step, which would output them to Excel or the Excel data model. Both files are in the C:\PowerBiDesktopSamples\MultipleTextFiles folder:
 - a. Colors_01.txt
 - b. Colors_02.txt
2. Name the queries Colors_01 and Colors_02.
3. Open one of the queries (I use Colors_01, but either will do).
4. Click the Append Queries button in the Power BI Desktop Query Editor Home ribbon. The Append dialog will appear.
5. From the Select The Table To Append pop-up, choose the query Colors_02.
6. Click OK.

The data from the two output tables is placed in the current query. You can now continue with any modifications that you need to apply. You will notice that the column names are not repeated as part of the data when the tables are appended one to the other.

Adding Multiple Files from a Source Folder

Now let's consider another possibility. You have been sent a load of files, possibly downloaded from an FTP site or received by e-mail, and you have placed them all into a specific directory. However, you do not want to have to carry out the process that we just saw and load files one by one if there are several hundred files. So here is a way to get Power BI Desktop to do the work of trawling through the directory and only loading files that correspond to a file name specification you have indicated.

1. Create a new Power BI Desktop file.
2. In the Power BI Desktop Home ribbon, click New Source ▶ More ▶ File in the Get data dialog. Then select Folder to the right of the dialog.
3. Click Connect. The Folder dialog is displayed.
4. Click the Browse button and navigate to the folder that contains the files to load. In this example, it is C:\PowerBiDesktopSamples\MultipleTextFiles. You can also paste in, or enter, the folder path if you prefer. The Folder dialog will look like Figure 5-15.

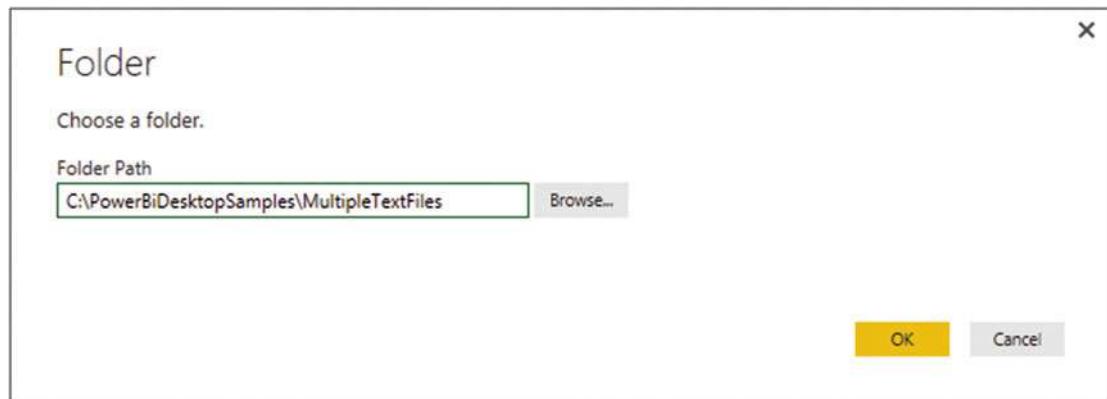


Figure 5-15. The Folder dialog

5. Click OK. The Power BI Desktop window opens. The contents of the folder are listed as a table, as shown in Figure 5-16.

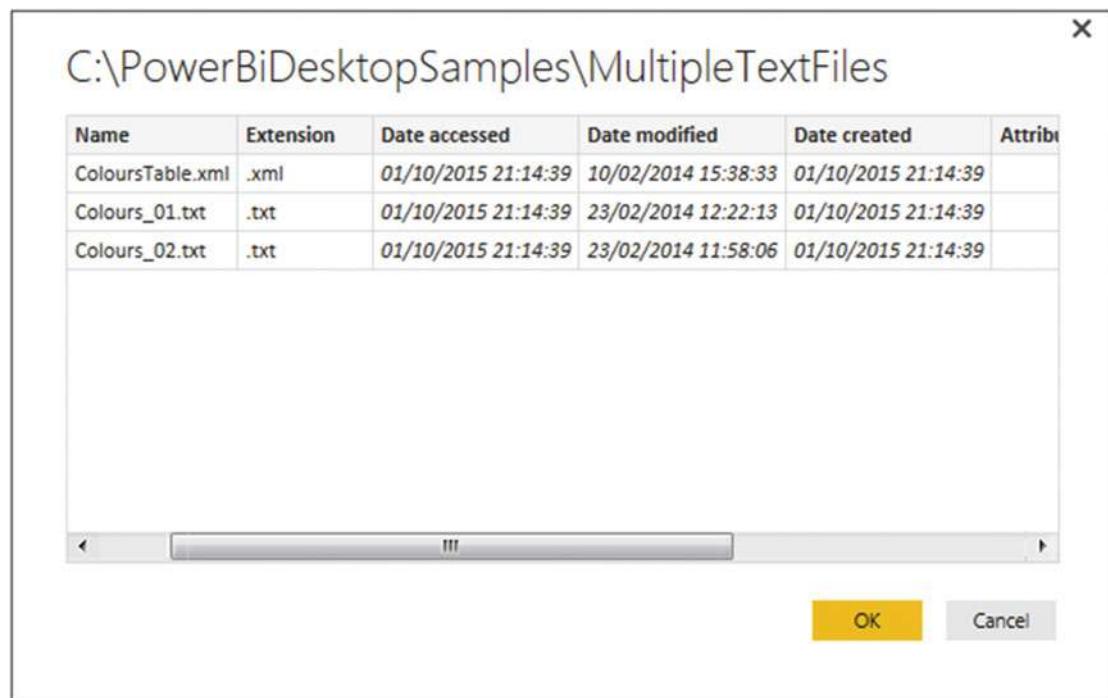


Figure 5-16. The folder contents in Power BI Desktop

6. Click OK. The Power BI Desktop Query Editor will display the list of files. This is shown in Figure 5-17.

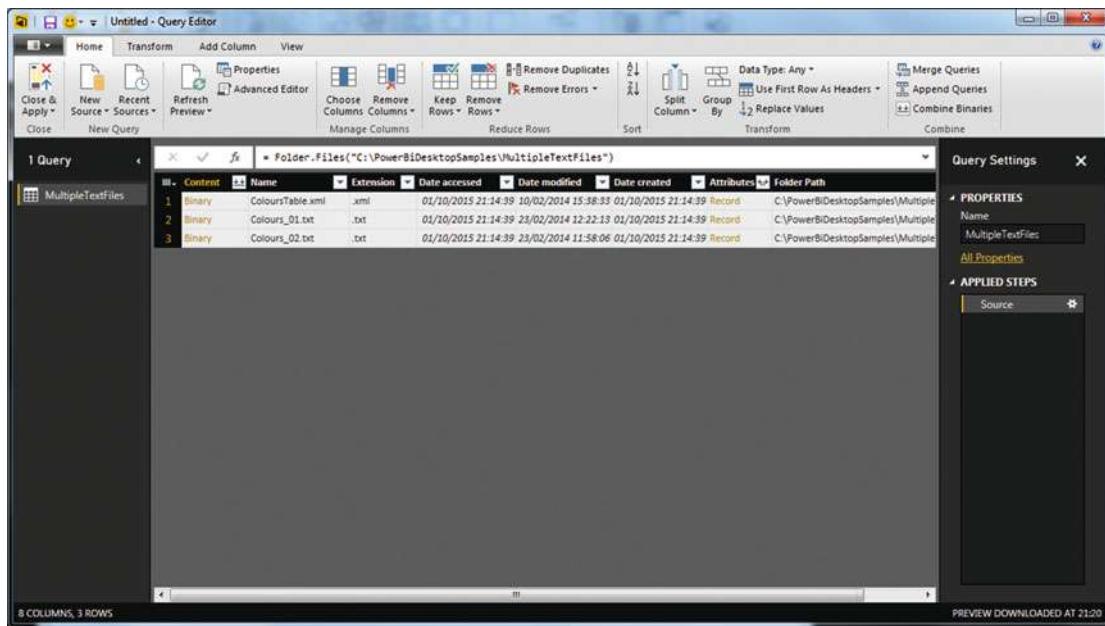


Figure 5-17. Files listed in the Power BI Desktop Query Editor

- As we want to load only text files, and avoid files of any other type, click the filter pop-up menu for the column title Extension and uncheck all elements *except .txt*. This is shown in Figure 5-18.

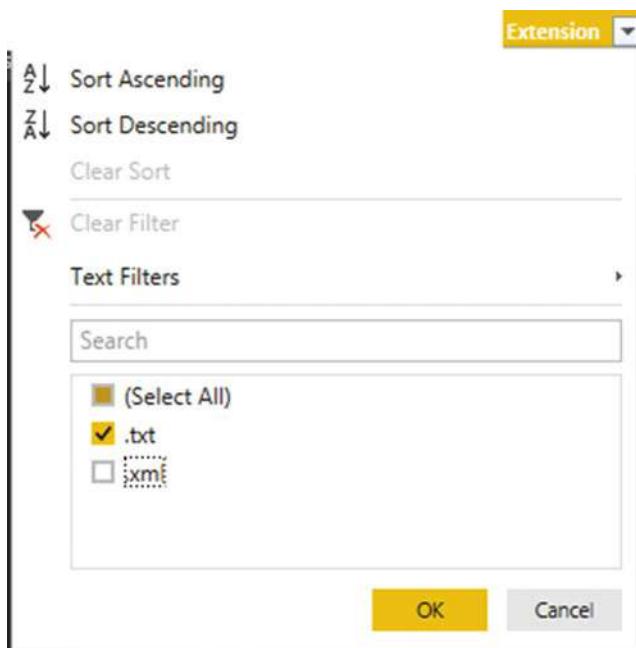


Figure 5-18. Filtering file types when loading multiple identical files

- Click the Expand icon (two downward-facing arrows) to the right of the first column title; this column is called Content and every row in the column contains the word *Binary*. Power BI Desktop loads all the files and displays the result, as shown in Figure 5-19.

	Column1	Column2
1	ColourID	Colour
2	1	Red
3	2	Blue
4	3	Green
5	4	Silver
6	5	Canary Yellow
7	ColourID	Colour
8	6	Night Blue
9	7	Black
10	8	British Racing Green
11	9	Dark Purple
12	10	Pink

Figure 5-19. All files loaded from a folder

- If (but only if) each file contains header rows, then scroll down through the resulting table until you find a title element. In this example, it is the word *ColourID* in the *ColourID* column.
- Right-click *ColourID* and select Text Filters ► Does Not Equal. All rows containing superfluous column titles are removed.

The contents of all the source files are now loaded into the Power BI Desktop Query Editor and can be transformed and used like any other dataset. What is more, if ever you add more files to the source directory, and then click Refresh in the Home ribbon, *all* the source files are reloaded, including any new files added to the directory.

Note If your source directory only contains the files that you want to load, then step 4 is unnecessary. Nonetheless, I always add steps like this in case files of the “wrong” type are added later, which would cause any subsequent process runs to fail. Equally, you can set filters on the file name to restrict the files that is loaded.

Changing the Data Structure

Sometimes your requirements go beyond the techniques that we have seen so far when discussing data cleansing and transformation. Some data structures need more radical reworking, given the shape of the data that you have acquired. I include in this category the following:

- Unpivoting data
- Pivoting data
- Transforming rows and columns

Each of these techniques is designed to meet a specific, yet frequent, need in data loading, and all are described in the next few pages.

Unpivoting Tables

From time to time, you may need to analyze data that has been delivered in a “pivoted” or “denormalized” format. Essentially, this means that information that really should be in a single column has been broken down and placed across several columns. An example of the first few rows of a pivoted dataset is given in Figure 5-20 and can be found in the C:\PowerBiDesktopSamples\PivotedDataSet.xlsx sample file.

InvoiceDate	Aston Martin	Bentley	Jaguar	MGB	Rolls Royce	Triumph	TVR
02/01/2013	75890	25700	88200	4500	62000	8500	
09/01/2013	31125						
10/01/2013	17500						
02/02/2013	75890	25700	63200	8500	62000	17000	37500
11/02/2013	22500						
02/03/2013	75890	25700	88200	4500	75890	8500	
12/03/2013	17500						
13/03/2013					31125		
14/03/2013	17500						
02/04/2013	75890	25700	99500	8500	62000	17000	37500
15/04/2013					22500		
16/04/2013	17500						
02/05/2013	75890	62000	124500	4500	75890	8500	
17/05/2013	17500						
18/05/2013	17500						
19/05/2013	22500						
02/06/2013	62000	62000	63200	8500	62000	17000	37500
20/06/2013	17500						
02/07/2013	62000	25700	88200	4500	62000	17000	
21/07/2013					17500		
22/07/2013	22500						
02/08/2013	62000	62000	38200	8500	62000	17000	37500
02/09/2013	62000	62000	124500	4500	75890	17000	
23/09/2013	17500						
02/10/2013	62000	62000	63200	8500	75890	17000	37500

Figure 5-20. A pivoted dataset

To analyze this data correctly, we really need the makes of the cars to be switched from being column titles to becoming the contents of a specific column. Fortunately, this is not hard at all:

1. In a new Power BI Desktop file, load the table PivotedCosts from the C:\PowerBiDesktopSamples\PivotedDataSet.xlsx file into Power BI Desktop. Ensure that the first row is set to be the table headers.
2. Select all the columns that you want to unpivot. In this example, this means all columns except the first one.
3. In the Transform ribbon, click the Unpivot button (or right-click with the columns selected and choose Unpivot from the context menu). The table is reorganized and the first few records look as they do in Figure 5-21. Unpivoted Columns is added to the Applied Steps list.

	InvoiceDate	Attribute	Value
1	02/01/2013	Aston Martin	75890
2	02/01/2013	Bentley	25700
3	02/01/2013	Jaguar	88200
4	02/01/2013	MGB	4500
5	02/01/2013	Rolls Royce	62000
6	02/01/2013	Triumph	8500
7	09/01/2013	Aston Martin	31125
8	10/01/2013	Aston Martin	17500
9	02/02/2013	Aston Martin	75890
10	02/02/2013	Bentley	25700
11	02/02/2013	Jaguar	63200
12	02/02/2013	MGB	8500
13	02/02/2013	Rolls Royce	62000
14	02/02/2013	Triumph	17000
15	02/02/2013	TVR	37500
16	11/02/2013	Aston Martin	22500
17	02/03/2013	Aston Martin	75890
18	02/03/2013	Bentley	25700
19	02/03/2013	Jaguar	88200

Figure 5-21. An unpivoted dataset

4. Rename the columns that Power BI Desktop Query has named Attribute and Value.

The data is now presented in a standard tabular way, and so it can be used to create a data model and then produce reports and dashboards.

Pivoting Tables

On some occasions, you may have to switch data from columns to rows so that you can use it efficiently. This kind of operation is called *pivoting data*. It is—perhaps unsurprisingly—very similar to the unpivot process that you saw in the previous section.

1. Follow steps 1 through 3 of the previous section so that you end up with the table of data that you can see in Figure 5-18.
2. Click inside the column *InvoiceDate*.
3. In the Transform ribbon, click the Pivot Column button. The Pivot Column dialog will appear.
4. Select Value (the column of figures) as the values column that is aggregated by the pivot transformation. The Pivot Column dialog will look like Figure 5-22.

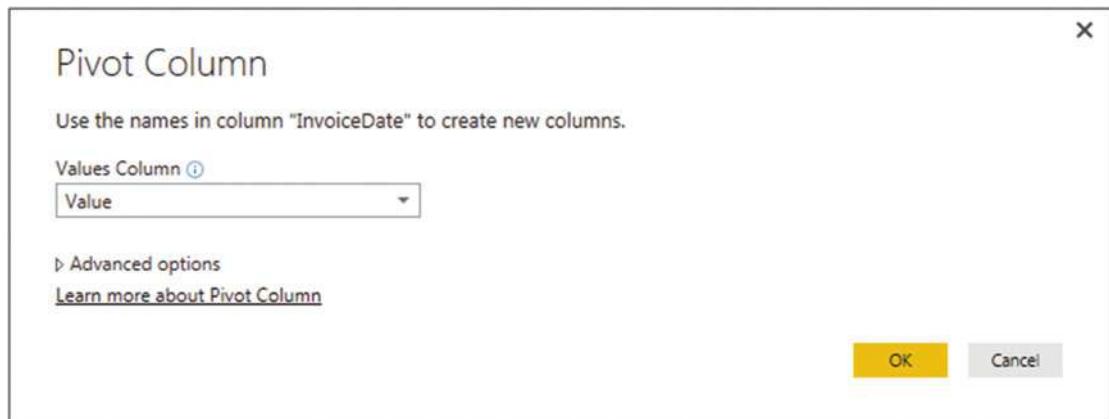


Figure 5-22. The Pivot Column dialog

5. Click OK. The table is pivoted and looks like Figure 5-23. Pivoted Column is added to the Applied Steps list.

	Attribute	02/01/2013	09/01/2013	10/01/2013	02/02/2013	11/02/2013	02/03/2013	12/03/2013
1	Aston Martin	75890	31125	17500	75890	22500	75890	17500
2	Bentley	25700	null	null	25700	null	25700	null
3	Jaguar	88200	null	null	63200	null	88200	null
4	MGB	4500	null	null	8500	null	4500	null
5	Rolls Royce	62000	null	null	62000	null	75890	null
6	TVR	null	null	null	37500	null	null	null
7	Triumph	8500	null	null	17000	null	8500	null

Figure 5-23. Pivoted data

Note The Advanced Options section of the Pivot Column dialog lets you choose the aggregation operation that is applied to the values in the pivoted table.

The Unpivot button contains another menu option that is displayed if you click the small triangle to the right of the Unpivot button. This is the Unpivot Other Columns option that will switch the contents of columns into rows for all the columns that are not selected when you run the transformation.

Transposing Rows and Columns

On some occasions, you may have a source table where the columns need to become rows and the rows columns. Fortunately, this is a one-click transformation for Power BI Desktop. Here is how to do it.

1. Load the C:\PowerBiDesktopSamples\DataToTranspose.xlsx Excel file into the Power BI Desktop Query Editor. You will see a data table like the one in Figure 5-24.



	Column1	Column2	Column3	Column4	Column5	Column6
1	1	2	3	4	5	6
2	United Kingdom	France	USA	Germany	Spain	Switzerland

Figure 5-24. A dataset needing to be transposed

2. In the Transform ribbon, click the Transpose button. The data is transposed and appears as two columns, just like the CountryList.txt file that you saw in Chapter 2.
3. Rename the columns.

Managing the Transformation Process

Pretty nearly all the transformation steps that we have applied so far have been individual elements that can be applied to just about any data table. However, when you are carrying out even a simple data load and transform process, you are likely to want to step through several transformations in order to shape, cleanse, and filter the data to get the result you want. This is where the Power BI Desktop approach is so clever, because you can apply most data transformation steps to just about any data table. The art is to place them in a sequence that can then be reused any time that the data changes to reprocess the new source data and deliver an up-to-date output.

The key to appreciating and managing this process is to get well acquainted with the Applied Steps list in the Query Settings pane. This list contains the details of every step that you applied, in the order in which you applied it. Each step retains the name that Power BI Desktop gave it when it was created, and each can be altered in the following ways:

- Modified
- Renamed
- Deleted
- Moved (in certain cases)

Many steps can also be modified, so you are not stuck with the choices that you initially apply.

Note Remember that before tweaking the order in which the process is applied, clicking any process step causes the table in the Power BI Desktop window to refresh to show you the state of the data up to and including the selected step. This is a very clear visual guide to the process and how the ETL is carried out.

Modifying a Step

How you alter a step will depend on how the original transformation was applied. This becomes second nature after a little practice and will always involve first clicking the step that you wish to modify and then applying a different modification. If you invoke a ribbon option, such as altering the data type, for instance, then you change the data type by simply applying another data type directly from the ribbon. If you used an option that displayed a dialog (such as splitting a column, among others), then you can right-click the step in the Applied Steps list and select Edit Settings from the context menu. Alternatively, and if you prefer, you can click the “gear” icon to the right of any step that was created using a dialog box to display the dialog for modification. This will cause the original dialog to reappear; in it, you can make any modifications that you consider necessary.

A final step that makes it easy to alter the settings for a process is to edit the formula that appears in the formula bar each time you click a step. We will look at this method in greater detail in a later chapter.

Note If you can force yourself to organize the process that you are writing with Power BI Desktop, then a little forethought and planning can reap major dividends. For instance, certain tasks, such as setting data types, can be carried out in a single operation. Not just that, but if you need to alter a data type for a column at a later stage, I suggest that you click the ChangedType step before you make any further alterations. This way, you extend the original step, rather than creating other steps, which can make the process more confusing and needlessly voluminous.

Renaming a Step

Because Power BI Desktop names steps using the name of the transformation that was applied, and then, if another similar step is applied later, uses the same name with a numeric increment, you may prefer to give more user-friendly names to process steps. This is done as follows:

1. Right-click the step that you want to rename.
2. Select Rename from the context menu.
3. Type in the new name.
4. Press Enter.

The step is renamed and the new name will appear in the Applied Steps list in the Query Settings pane. This way you can ensure that when you come back to a data transformation process days, weeks, or months later you are able to understand more intuitively the process that you defined, as well as why you shaped the data like you did.

Deleting a Step or a Series of Steps

Deleting a step is all too easy, but doing so can have serious consequences. This is because an ETL process is often an extremely tightly coupled series of events, where each event depends intimately on the preceding one. So deleting a step can make every subsequent step fail. Knowing which events you can delete without drastic consequences will depend on the types of process that you are developing as well as your experience with Power BI Desktop. In any case, this is what you should do if you need to delete a step:

1. Right-click the step that you want to delete.
2. Select Delete. The Delete Step dialog might appear, as shown in Figure 5-25.

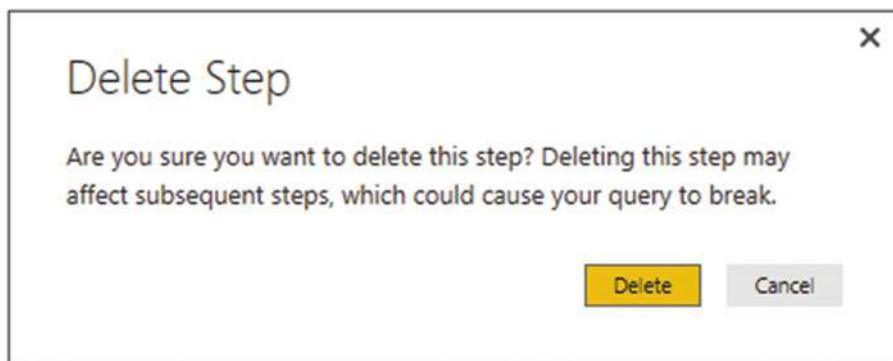


Figure 5-25. The Delete Step dialog

3. Confirm by clicking the Delete button. The step is deleted.

If—and it is highly possible—deleting this step causes issues for the rest of the process, you will see that the data table is replaced by an error message. This message will vary depending on the type of error that Power BI Desktop has encountered.

When describing this technique, I was careful to state that you *might* see the Delete dialog. If you are deleting the final step in a sequence of steps, then you will probably not see it, since there should not be any potentially horrendous consequences; at worst, you will have to re-create the step. If you are deleting a step in the middle of a process, then you might want to think seriously about doing so before you cause a potentially vast number of problems. Consequently, you are asked to confirm the deletion in these cases.

Note If you realize at this point that you have just destroyed hours of work, then (after drawing a deep breath) click the File menu in the Power BI Desktop window (the downward-facing triangle at the top left) and select Discard And Close. You will lose all work up until the last time you clicked Apply And Close, however. Don't count on using an undo function as you can in other desktop applications. To lower your blood pressure, you may prefer to save a copy of a complex file before deleting any steps.

An alternative technique is to place the pointer over a process step and click the cross (x) icon that appears. You may still have to confirm the deletion.

If you realize that an error in a process step has invalidated all your work up until the end of the process, rather than deleting multiple elements one by one, click Delete Until End from the context menu at step 2 in the preceding exercise.

Adding a Step

You can add a step anywhere in the sequence. All you have to do is to click the step that *precedes* the new step that you want to insert *before* clicking the icon in any of the ribbons that corresponds to the new step. As is the case when you delete a step, Power BI Desktop will display an alert warning you that this action *could* cause problems with the process from this new step on.

Altering Process Step Sequencing

It is possible—technically—to resequence steps in a process. However, in my experience, this is not always practical, since changing the order of steps in a process can cause as much damage as deleting a step. Nonetheless, you can always try it like this:

1. Right-click the step that you want to resequence.
2. Select Move Up or Move Down from the context menu.

I remain pessimistic that this can work miracles, but it is good to know that it is there.

An Approach to Sequencing

Given the array of available data transformation options, you may well be wondering how best to approach a new ETL project using Power BI Desktop. I realize that all projects are different, but as a rough and ready guide, I suggest attempting to order your project like this:

1. First, of course, load the data into Power BI Desktop.
2. Second, promote or add correct column headers. For example, you really do not want to be looking at step 47 of a process and wondering what Column29 is, when it could read ClientName.
3. Third, remove any columns that you do not need. The smaller the dataset, the faster the processing. What is more, you will find it easier to concentrate on, and understand, the data if you are only looking at information that you really need. Any columns that have been removed can be returned to the dataset simply by deleting or editing the step that removed them.
4. Fourth, alter the data types for every column in the table. Correct data types are fundamental for many transformation steps, and are essential for filtering, so it's best to get them sorted out early on.
5. Next, filter out any records that you do not need. Once again, the smaller the dataset, the faster the processing. This includes deduplication.
6. Then, carry out any necessary data cleansing and transforms.
7. Finally, carry out any necessary column splits or adding custom columns.

Once again, this is not a definitive guide, but I hope that it will help you to see the wood for the trees.

Error Records

Some data transformation operations will cause errors. This can be a fact of life in when mashing up source data. For instance, you could have a few rows in a large dataset where a date column contains a few records that are texts or numbers. If you convert the column to a date data type then any values that cannot be converted will appear as error values.

Removing Errors

Assuming that you do not need records that Power BI Desktop has flagged as containing an error, you can have all such records removed in a single operation:

1. Click inside the column containing errors; or if you want to remove errors from several columns at once, Ctrl-click the titles of the columns that contain the errors.
2. Click Remove Errors in the Home ribbon. Any records with errors flagged in the selected columns are deleted. Removed Errors is added to the Applied Steps list.

You have to be very careful here not to remove valid data. Only you can judge, once you have taken a look at the data, if an error in a column means that the data can be discarded safely. In all other cases, you would be best advised to look at cleansing the data or simply leaving records that contain errors in place. The range and variety of potential errors is as vast as the data itself. You could see errors due to invalid data types, for instance.

Managing Queries

Once you have used Power BI Desktop for any length of time you will probably become addicted to creating more and deeper analyses based on wider-ranging data sources. Inevitably, this will mean learning to manage the data sources that feed into your data models efficiently and productively.

Fortunately, Power BI Desktop Query comes replete with a small arsenal of query management tools to help you. These include

- Organizing queries
- Grouping queries
- Duplicating queries
- Referencing queries
- Adding a column as a new query
- Enabling data load
- Enabling data refresh

Let's take a look at these functions, one by one.

Organizing Queries

When you have a dozen or more queries that you are using in the Power BI Desktop Query Editor, you may want to exercise some control over how they are organized. To begin with, you can modify the order in which queries appear in the Query List pane on the left of the Power BI Desktop Query window. This lets you override the default order, which is that the most recently added data source appears at the bottom of the list.

Do the following to change the position of a query in the list:

1. Right-click the query that you want to move.
2. Select Move Up (or Move Down) from the context menu.

You have to carry out this operation a number of times to move a query up or down a number of places.

Grouping Queries

You can also create custom groups to better organize the queries that you are using in a Power BI Desktop file. This will not have any effect on how the queries work. Grouping queries is simply an organizational technique.

Create a New Group

The following explains how to create a new group.

1. Right-click the query that you want to add to a new group.
2. Select Move To Group ► New Group from the context menu. The New Group dialog will appear.
3. Enter a name for the group and (optionally) a description. The dialog will look something like Figure 5-26.

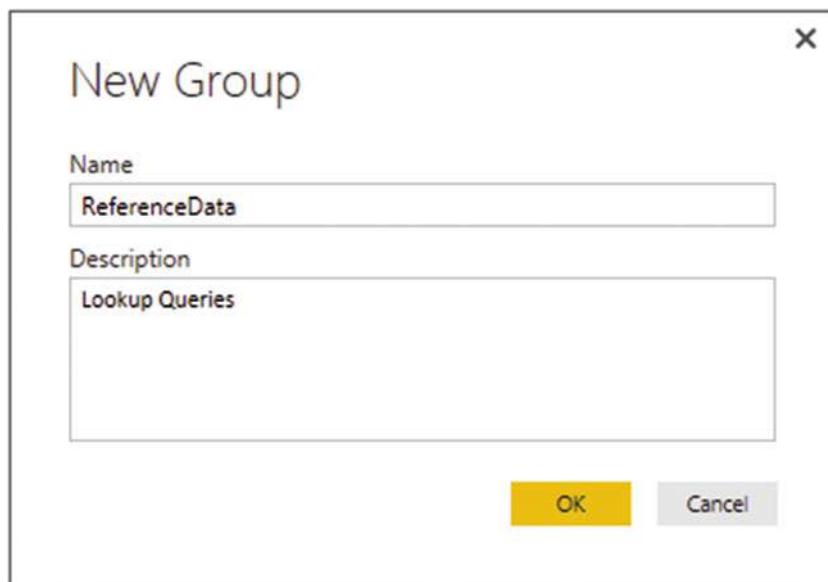


Figure 5-26. The New Group dialog

4. Click OK.

The new group is created and the selected query will appear in the group. The Query pane will look something like Figure 5-27.

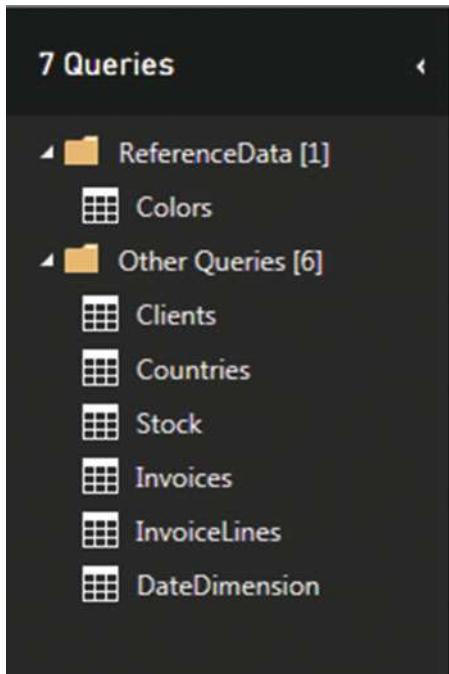


Figure 5-27. The Query pane with a new group added

Note By default, all other queries are added to a group named Other Queries.

Add a Query to a Group

To move a query from its current group to another group, you can carry out the following steps:

1. Right-click the query that you want to add to another existing group.
2. Select Move To Group ► *Destination Group Name* from the context menu.

The selected query is moved to the chosen group.

Duplicating Queries

If you have done a lot of work transforming data, you could well want to keep a copy of the original query before trying out any potentially risky alterations to your work. Fortunately, this is extremely simple.

1. Right-click the query that you want to copy.
2. Select Duplicate from the context menu.

The query is copied and the duplicate appears in the list of queries. It has the same name as the original query, with a number in parentheses appended. You can always rename it in the Query Settings pane or in the Queries Pane on the left of the Query Editor window.

Referencing Queries

If you are building a complex ETL (Extract, Transform, Load) routine, you might conceivably organize your work in stages to better manage the process. To help you with this, the Power BI Desktop Query Editor allows you to use the output from one query as the source for another query. This enables you to break down different parts of the process (structure, filters, then cleansing, for example) into separate queries so that you can concentrate on different aspects of the transformation in different queries.

To use the output of one query as the source data for another you need to *reference* a query. The following explains how to do it:

1. Right-click the query that you want to use as the source data for a new query.
2. Select Reference from the context menu.

A new query is created in the list of queries in the Query pane. The new query has the same name as the original query, with a number in parentheses appended. If you click the new query, you see exactly the same data in the referenced query as you can see if you click the final step in the source query.

From now on, any modifications that you make in the referenced (source) query produces an effect on the data that is used as the source for the second query. It is worth noting that any queries that are “intermediate” queries (that is, queries that you use to modify data but that do not show in Data view or Report view are in italics).

Add a Column As a New Query

There are occasions when you might want to extract a column of data and use it as a separate query. It could be that you need the data that it contains as reference data for another query, for example. The following explains how you can do this.

1. In the Query list on the left, select the query containing the column that you want to isolate as a new query.
2. Right-click the title of the column containing the data that you want to isolate.
3. Select Add As New Query from the context menu. A new query is created. It is named after the original query and the source column.
4. In the Transform ribbon, click To Table. The To Table dialog will appear.
5. Click OK.
6. Rename the query.

Enable Data Load

You may have gained the impression that any query that you create will always become part of the Power BI Desktop data model. This is emphatically *not* the case. You can create queries that are in effect only “staging” queries that are part of a more complex sequence of transformations, or queries that contain only lookup data that is added to another table but not needed in the data model, for instance. In cases like these, you certainly do not want these tables adding clutter to the data model.

To prevent a query being added to the data model, do the following:

1. In the Query list on the left, select the query that you want to keep in the Query Editor—but not in the data model.
2. Right-click the query and uncheck Enable Load.

The query will no longer be loaded into the data model with the other queries.

To reset a query as a candidate for loading into the data model merely carry out the same operation and ensure that Enable load is checked. You can see from the check mark in the context menu if the query is due to be loaded or not. As an alternative you can right-click the query and display the query properties, where you can check (or uncheck) the Enable load to report check box.

Enable Data Refresh

By default, all queries can be refreshed. This lets you gather the very latest data from the source into the Power BI Desktop Query Editor and then into the data model.

There could be times when you do not want to refresh a query. Perhaps the data source is unobtainable, or is slow, or you want to return later for the latest data. Power BI Desktop Query Editor lets you set the refresh option for each query like this:

1. In the Query list on the left, right-click the query whose refresh status you want to modify.
2. Select Properties from the context menu. The Query Properties dialog will appear.
3. Uncheck Enable refresh of this query. The dialog will look like Figure 5-28.

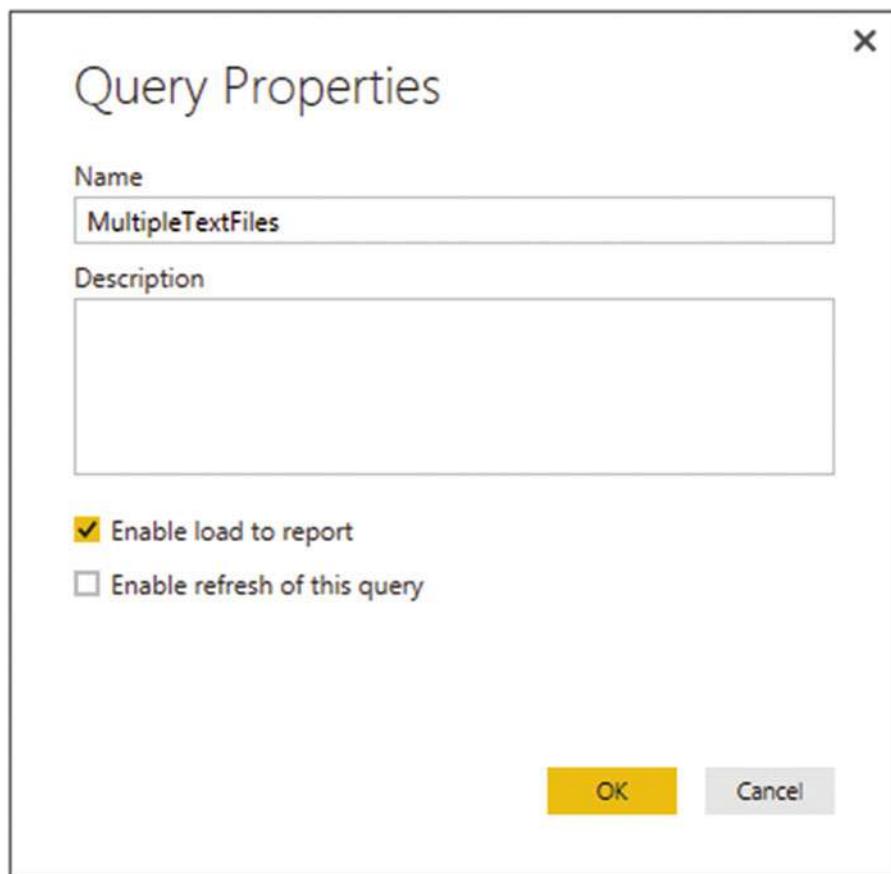


Figure 5-28. The Query Properties dialog

4. Click OK.

Note Only queries that are enabled for load can have the refresh property modified.

Pending Changes

When you are dealing with data sources and switch from the Query Editor to the Power BI Desktop View, normally, you then want to load the full data from the source into the data model.

The downside to this approach is that a huge set of source data can take a long time to load, or reload, when you move back to creating and modifying visualizations. This is why Power BI Desktop will let you select Close from the Close and Apply button in the Query Editor Home ribbon. Doing this will return you instantly to the Data View, but will not apply any changes that you have made. As a reminder, you will see an alert like the one shown in Figure 5-29 at the top of the Data View.

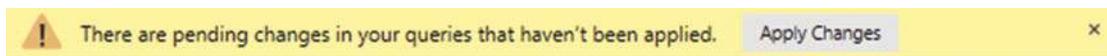


Figure 5-29. The pending changes alert

You can continue to work in Data View as long as you like. Click the Apply Changes button when you have time to reload the modified source data into the data model.

Copying Data from Power BI Desktop

Power BI Desktop is designed as a data destination. It does not have any data export functionality as such. You can manually copy data from the Power BI Desktop Query Editor, however. More precisely, you can copy any of the following:

- The data in the query
- A column of data
- A single cell

In all cases, the process is the same:

1. Click the element to copy. This can be
 - a. The top left square of the data grid
 - b. A column title
 - c. A single cell
2. Right-click and select copy from the context menu.

You can then paste the data from the clipboard into the destination application.

Note This process is somewhat limited because you cannot select a range of cells. And you must remember that you are only looking at sample data in the Query Editor.

Conclusion

This chapter showed you how to structure your source data into a valid data table from one or more potential sources. You saw how to pivot and unpivot data, to fill rows up and down with data, as well as how to create new columns of data.

Possibly the most important thing that you have learned is how to join individual queries so that you can add the data from one query into another. This can involve looking up data from a separate query or carrying the aggregated results from one query into another.

Finally, you saw some of the basic techniques for managing and organizing queries. This way an ETL process can be polished into a structured and maintainable process.

In this chapter and the three previous chapters, you have seen essentially a three-stage process: first, you find the data, then you load it into Power BI Desktop Query, and from there, you cleanse and modify it. The techniques that you can use are simple but powerful and can range from changing a data type to merging multiple data tables. Now that your data is prepared and ready for use, you can add it to the Power BI Desktop data model and start creating your Power BI Desktop dashboards.

CHAPTER 6



Creating a Data Model

You need only one thing to create stunning visualizations and that is good data. Specifically, you need data loaded into the Power BI Desktop data model where you can handle tens of millions of rows from multiple data sources and hone them into a coherent and powerful framework on which you can build your analyses and dashboards.

Finding, editing, and loading the source data using queries (as you learned in Chapters 2 through 5) is fundamental to preparing your data for the data model. However, the process does not stop there. Once you have made accessible the data that you need you still have a few more tasks to carry out. Specifically you need to assemble these queries into a coherent data model and ensure that the underlying structure is clean, ready for use, and contains all the metrics that your dashboards require.

To guarantee that you are at ease when creating a data model in Power BI Desktop for later analysis in your dashboards, this chapter introduces you to some of the techniques that you need to apply to model your data. You will discover how to take the data tables that you loaded and convert them into a coherent dataset in the Power BI Desktop data model. This dataset enables you to deliver information, insight, and analysis from the data in the tables. This learning curve covers

- Specifying data types.
- Formatting data in the data model.
- Categorizing data.
- Adding “sort by” columns that ensure the correct sort order in dashboard elements.
- Establishing relationships between the tables so that Power BI Desktop understands how the data in one table is linked to the data contained in another table. Chaining one table to another will let you use the data to deliver accurate and cogent results.

Once again, all the sample files used in this chapter are available on the Apress web site. Once downloaded, they should be in the C:\PowerBiDesktopSamples folder.

Data Modeling in the Power BI Desktop Environment

Before leaping into the detail of what can be done to enhance a data model I think that it is best if you first familiarize yourself with the tool itself so that you can feel at home in this new environment.

The Power BI Desktop Data View

To start using Power BI Desktop for data modeling, you need to follow these steps:

1. Run Power BI Desktop and load any queries that you need to access your data. To follow the examples in this chapter, you need to load the C:\PowerBiDesktopSamples\DataModeling.pbix file.
2. If you are using your own data, you need to close the Power BI Desktop Query window, applying any changes that you have made. You will revert to the Power BI Desktop environment.
3. Click the Data View icon on the left. You will switch to the Data View, where you can see all the available tables in the Fields list on the right. One of the tables will be selected and its data will be visible in the Data pane as shown in Figure 6-1.

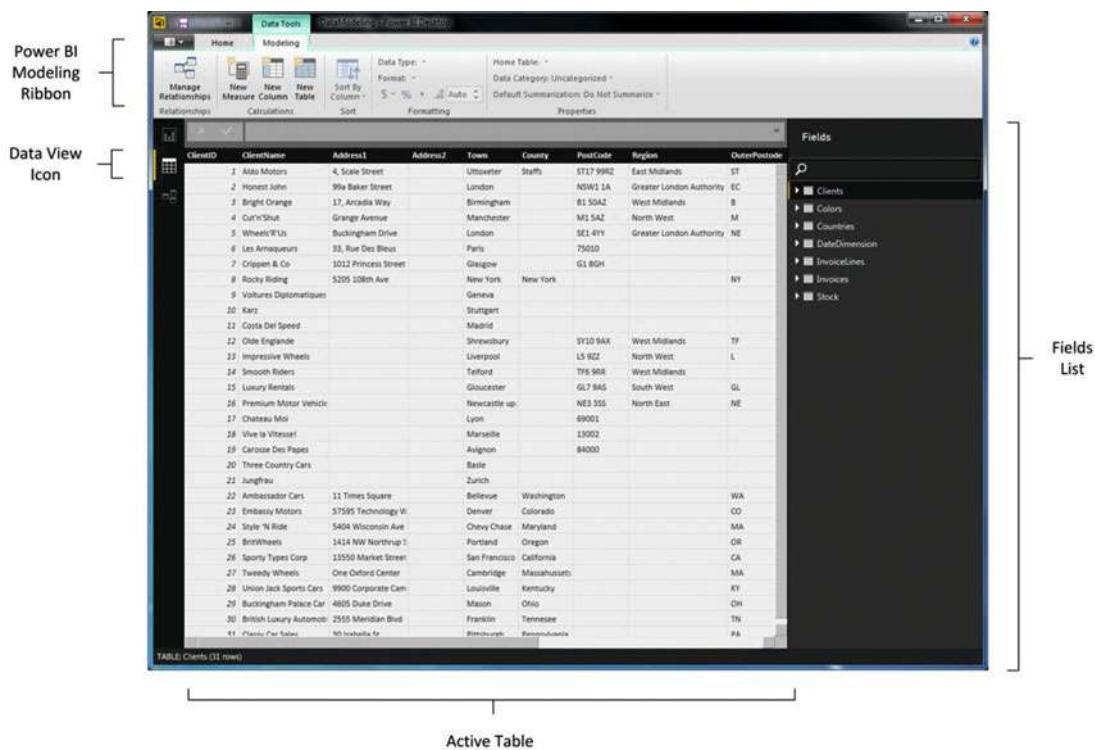


Figure 6-1. Data View

Data Model or Query?

You could well be thinking that this does not look like very much at all yet. If anything, it looks like an extension of the Power BI Desktop Query window. However, be reassured, you will soon see what you can do in Data View, and exactly how powerful a tool Power BI Desktop really is when it comes to data modeling. For the moment, it is essential to remember that you have (so to speak) opened a door into the engine room of Power BI desktop. Although this new world is part of Power BI Desktop (and you can return to Power BI

visualizations instantaneously just by clicking the Report View icon above the Data View icon), it is best if you consider it a kind of parallel universe for the moment. This universe has its own ribbons and buttons and is separate from the dashboard window so that you can concentrate on enhancing the data without getting distracted.

The Power BI Desktop Data View is also different from the Power BI Desktop Query Editor. For instance, one major difference between the Power BI Desktop Query window and the Data View is that in Data View you are looking at *all* your data. The Query window only ever shows a sample of data. Once you load the data into the data model, you finally have access to the *entire* dataset (all the data in all the tables) and this is what you can see in the Data View.

A second difference is that once you have left the Query window, you are always working with the entire dataset and only filtering it as required by specific dashboards or visualizations. So you need to be aware that any filters that you apply in the Query window limits the data that can be displayed in Power BI Desktop. By contrast (and as you will see later in this book), you can apply subsequent filters to entire dashboards or specific visualizations once you have defined your core data.

There are some areas where Power BI Desktop Query can do some of the things that can also be done in the Data or Relationships Views. For instance, you can create calculated columns in both (you will see how to do this in Data View in the following chapter). My advice is to try to remember that Power BI Desktop Query is for finding, filtering and mashing up data, whereas Data View is for refining and calculating the metrics in your data model. However, each user can take the approach that they prefer and carry out any necessary calculations in either the Query Editor or the Data View. After all, it is the result that counts.

The Power BI Desktop Data View Ribbons

So what, exactly, are you looking at when you start Power BI Desktop? Essentially, you can see two ribbons, which are all devoted to data management:

- The Home ribbon
- The Modeling ribbon

Since the Home ribbon is principally used when creating reports and dashboards, I will explain it in Chapter 8. I prefer to explain the Power BI Desktop Modeling ribbon as we start out in this chapter, because you will use it extensively in the next few pages. This should make understanding the Power BI Desktop environment easier. However, if you prefer to skip this section (or possibly use it as reference later), then feel free to jump ahead.

The Modeling Ribbon

Because I cover the Home ribbon in Chapter 10 (as it essentially concerns dashboard creation), I will only explain the Modeling ribbon here. The Modeling ribbon is used to categorize and organize data and tables as well as adding calculated columns and additional metrics. The buttons that it contains are shown in Figure 6-2 and explained in Table 6-1.

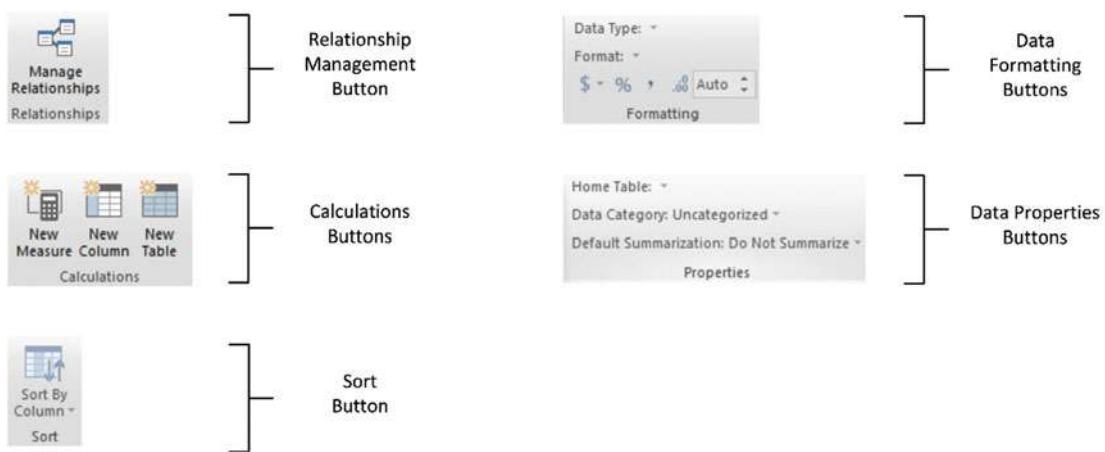


Figure 6-2. Buttons in the Modeling ribbon

Table 6-1. The Modeling Ribbon Buttons

Button	Description
Manage Relationships	Lets you join tables as well as delete these joins (called <i>relationships</i>) and modify them.
New Measure	Used to add a new value or calculation to a table.
New Column	Adds a new calculated column to a table.
Sort By Column	Specifies that the data in one column be used as the basis for sorting data in another column.
Data Type	Lets you define the data type for a column.
Format	Lets you specify how numbers are formatted.
Home Table	Lets you select the table that will contain a measure.
Data Category	Allows you to define that a certain column is of a specific category. It is mostly used with geographical data for mapping.
Default Summarization	Lets you define the default aggregation that is applied in dashboard visualizations.

Managing Power BI Desktop Data

Assuming that all has gone well, you now have a series of tables from various sources successfully added to your Power BI Desktop data model. You can see these data tables in the Fields list at the right of the Power BI Desktop window. Clicking on a table will display the data from that table in the central area of the Data View window. It will soon be time to see what we can do with this data, but first, to complete the roundup of overall data management, you need to know how to do the following:

- Rename tables
- Delete tables
- Move tables

- Move around a table
- Rename a column
- Delete columns
- Set column width

Manipulating Tables

Let's begin by seeing how you can tweak the tables that you have imported.

Renaming Tables

Suppose that you wish to rename a table that you previously imported using Power BI Desktop Query. These are the steps to follow:

1. Right-click the table name in the Fields list at the right of the Power BI Desktop window.
2. Select Rename. The current name will be highlighted in the Fields list.
3. Enter the new name or modify the existing name.
4. Press Enter.

This also renames the query on which the table is based. In essence, Power BI will let you rename datasets either as queries in the Power BI Desktop Query window or in the Data window. Because the query *is* the table, renaming one renames the other.

Deleting a Table

Deleting a table is virtually identical to the process of renaming one—you just right-click the table name tab and select Delete instead of Rename. As this is a potentially far-reaching operation, Power BI Desktop will demand confirmation.

Note When you delete a table, you are removing it from Power BI Desktop completely. This means that it is *also* removed from the set of queries that you may have used to transform the data. So you need to be careful, because you could lose all your carefully wrought transformation steps as well.

Selecting a Column from the List of Available Column Names

If you want to leap straight to a column of data (and presuming you know which column contains the data that interests you), all you have to do is

1. In the Fields list, click the expand triangle to the left of the table containing the field that you want to jump to. A list of the fields in the table appears, as shown in Figure 6-3.

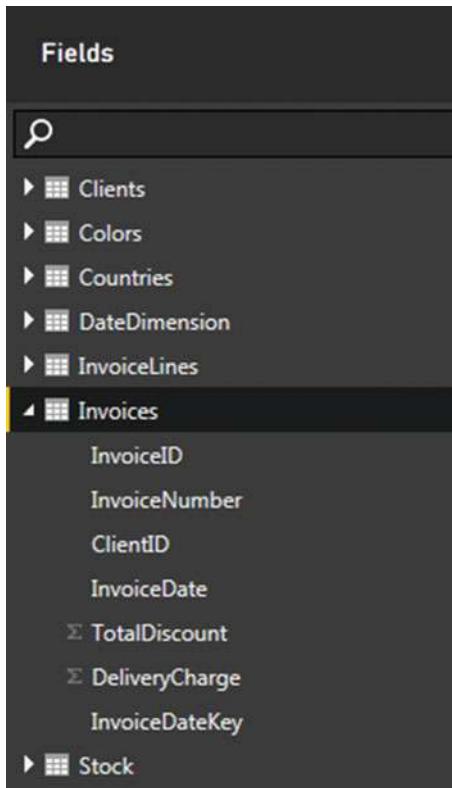


Figure 6-3. The list of columns in a table

2. Click the name of the field that contains the data that you want to study. The field will be selected in the table.

Manipulating Columns

Now let's see how to perform similar actions—but this time inside a table—to the columns of data that make up the table.

Renaming a Column

Renaming a column is pretty straightforward. All you have to do is

1. In the Fields list, right-click the title of the column that you wish to rename. In this example, it is the field ClientType in the Data Clients table. The column will be selected and the context menu will appear. This is shown in Figure 6-4.

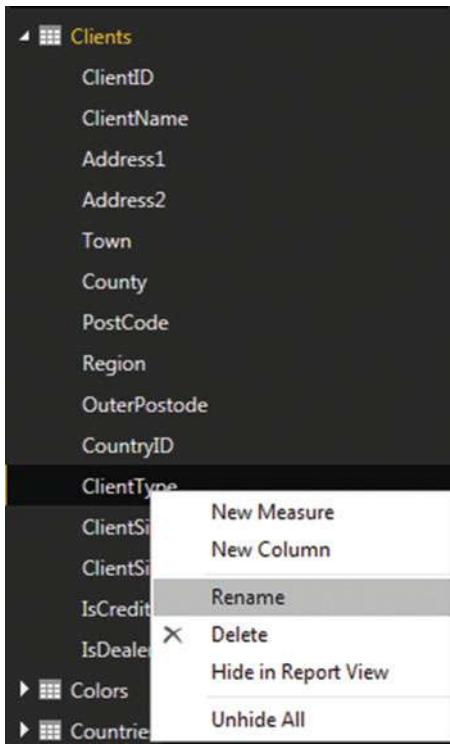


Figure 6-4. The Power BI Desktop context menu

2. Select Rename from the context menu. The current column name will be highlighted.
3. Type in the new name for the column (**TypeOfClient**) and press Enter.

And that is it. Your column has been renamed; it is also selected. You cannot use the name of an existing column in the same table, however. If you try, Power BI Desktop will name the column *ExistingColumnName2*, for example.

Although renaming a column may seem trivial, it can be important. Consider other users first; they need columns to have instantly understandable names that mean something to them. Then there is the PowerBI.Com Q&A natural language feature. This only works well if your columns have the sort of names that are used in the queries—or ones that are recognizable synonyms. Finally, you cannot rename columns individually in separate visualizations, so you really need to give your columns the names that you are happy seeing standardized across all the dashboards that are based on this dataset.

You can also rename a column in the Fields list. After all, a column *is* a field. So all you have to do is to expand the table containing the column that you want to rename by clicking the triangle at the left of the table name. Then right-click the field that interests you and select Rename from the context menu.

Note Power BI Desktop is very forgiving when it comes to renaming columns (or, indeed, tables and calculations too). You can rename most elements in the Query window, the Data View, the Report View or the Relationships View (depending where they were added) and the changes will ripple through the entire Power BI Desktop data model. Better still, renaming columns, calculations, and tables generally do not cause Power BI Desktop any difficulties if these elements have already been applied to dashboards, which are also updated to reflect the change of name. Just be sure that if you customize a lot of names and don't document the columns for reference, you will find that reverse engineering a report can be quite a task. In practice, it can save you a lot of effort if you always keep a reference of alias columns back to source in the query.

Deleting Columns

Deleting a column is equally easy. You will probably find yourself doing this when you bring in a column that you did not mean to import or when you find that you no longer need a column. So, to delete a column, you need to do the following:

1. Right-click the title of the column that you wish to delete. Alternatively, right-click the field name in the Fields list. The column will be selected and the context menu will appear.
2. Select Delete Columns from the context menu. Unless the column is empty, the confirmation dialog will appear as shown in Figure 6-5.

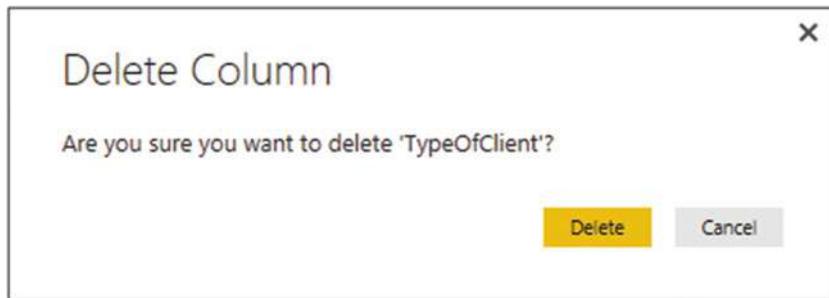


Figure 6-5. Deleting a column

3. Click OK. The column will be deleted from the table.

Deleting unused columns is good practice because you will

- Reduce the memory required for the dataset.
- Speed up data refresh operations.
- Reduce the size of the Power BI Desktop file.

■ **Note** Deleting a column really is permanent. You cannot use the undo function to recover it. Indeed, refreshing the data will not add the column back into the table either. If you have deleted a column by accident, you can choose to close the Power BI Desktop file without saving and reopen it, thus reverting to the previous version. Otherwise, you can return to the source query and add the column name back into the query.

Moving Columns

Once in the data model, you cannot move columns around. So if you want to change the column order for any reason, you have to switch to the Power BI Desktop Query Editor and move the column there. Once you save and apply your changes, the modified column order is visible in the data model.

Setting Column Widths

One final thing that you may want to do to make your data more readable—and consequently easier to understand—is to adjust the column width. I realize that as an Excel or Word user, you may find this old hat, but in the interests of completeness, this is how you do it:

1. Place the mouse pointer over the right-hand limit of the column title in the column whose width you want to alter. The cursor will become a two-headed arrow.
 2. Drag the cursor left or right.
-

■ **Note** You cannot select several adjacent columns before widening (or narrowing) one of them to set them all to the width of the column that you are adjusting. You can double-click the right-hand limit of the column title in the column whose width you want to alter so that Power BI Desktop can set the width to that of the longest element in the column.

Power BI Desktop Data Types

When you are importing data from an external source, Power BI Desktop tries to convert it to one of the nine data types that it uses. These data types are described in Table 6-2.

Table 6-2. Power BI Desktop Data Types

Data Type	Description
Decimal Number	Stores the data as a real number with a maximum of 15 significant decimal digits. Negative values range from -1.79E +308 to -2.23E -308. Positive values range from 2.23E -308 to 1.79E + 308.
Fixed Decimal Number	Stores the data as a number with a specified number of decimals.
Whole Number	Stores the data as integers that can be positive or negative but are whole numbers between 9,223,372,036,854,775,808 (-2^63) and 9,223,372,036,854,775,807 (2^63-1).
Date/Time	Stores the data as a date and time in the format of the host computer. Only dates on or after the 1st of January 1900 are valid.
Date	Stores the data as a date in the format of the host computer.
Time	Stores the data as a time element in the format of the host computer.
Text	Stores the data as a Unicode string of 536,870,912 bytes at most.
True/False	Stores the data as Boolean—true or false.
Binary	Stores the data as binary (machine-readable) data.

Formatting Power BI Desktop Data

Power BI Desktop allows you to apply formatting to the data in the tables that it contains. When you format the data in the Power BI data model, you are defining the format that will be used in all visualizations in all the dashboards that you create using this metric. So it is probably worth learning to format data for the following reasons:

- You will save time and multiple repetitive operations when creating reports and presentations by defining a format once and for all in Data View. The data will then appear using the format that you applied in multiple visualizations in this Power BI Desktop file.
- It can help you understand your data more intuitively if you can see the figures in a format that has intrinsic meaning.

This explains how to format a column (of figures in this example):

1. Assuming that you are working in the Power BI Desktop Data View, click the table name in the Fields list that contains the metric you wish to format.
2. Click inside the column that you want to format (SalePrice from the InvoiceLines table in the DataModeling.pbix sample file).
3. In the Modeling ribbon, click the Thousands Separator icon (the comma in the Formatting section). All the figures in the column will be formatted with a thousands separator and two decimals.

The various formatting options available are described in Table 6-3.

Table 6-3. Currency Format Options

Format Option	Icon	Description	Example
General		Leaves the data unformatted	100000.01
Currency	\$ ▾	Adds a thousands separator and two decimals as well as the current monetary symbol	\$100,000.01
Date/Time		Formats a date and/or time value in one of a selection of date and time formats	
Decimal Number	,	Adds a thousands separator and two decimals	100,000.01
Whole Number		Adds a thousands separator and truncates any decimals	100,000
Percentage	%	Multiplies by 100, adds two decimals, and prefixes with the percentage symbol	28.78%
Scientific		Displays the numbers in Scientific format	1.00E+05
Decimal Point		Increases or reduces the number of decimals	

If you wish to return to “plain vanilla” data, then you can do this by selecting the General format. Remember that you are not in Excel, and you cannot format only a range of figures—it is the whole column or nothing. Also, there is no way to format nonadjacent columns by Ctrl-clicking to perform a noncontiguous selection. And, you cannot select multiple adjacent columns and format them in a single operation.

By now, you have probably realized that power BI Desktop operates on a “Format once/apply everywhere” principle. However, this does not mean that you have to prepare the data exhaustively before creating dashboards and reports. You can flip between the data model and the Report View at any time to select another format, secure in the knowledge that the format that you just applied is used throughout all of your dashboards wherever the relevant metric is used.

Note Numeric formats are not available for selection if the data in a field is of text or data/time data type. Similarly, date and time formats are only available if the column contains data that can be interpreted as dates or times.

Currency Formats

Power BI Desktop will propose a wide range of currency formats. To choose the currency that you want:

1. Click the pop-up (the downward-facing triangle) to the right of the currency format icon. This will display a list of available formats.
2. Select the currency symbol that you want, or scroll through the list to view all the available currency formats, as shown in Figure 6-6, then click OK.

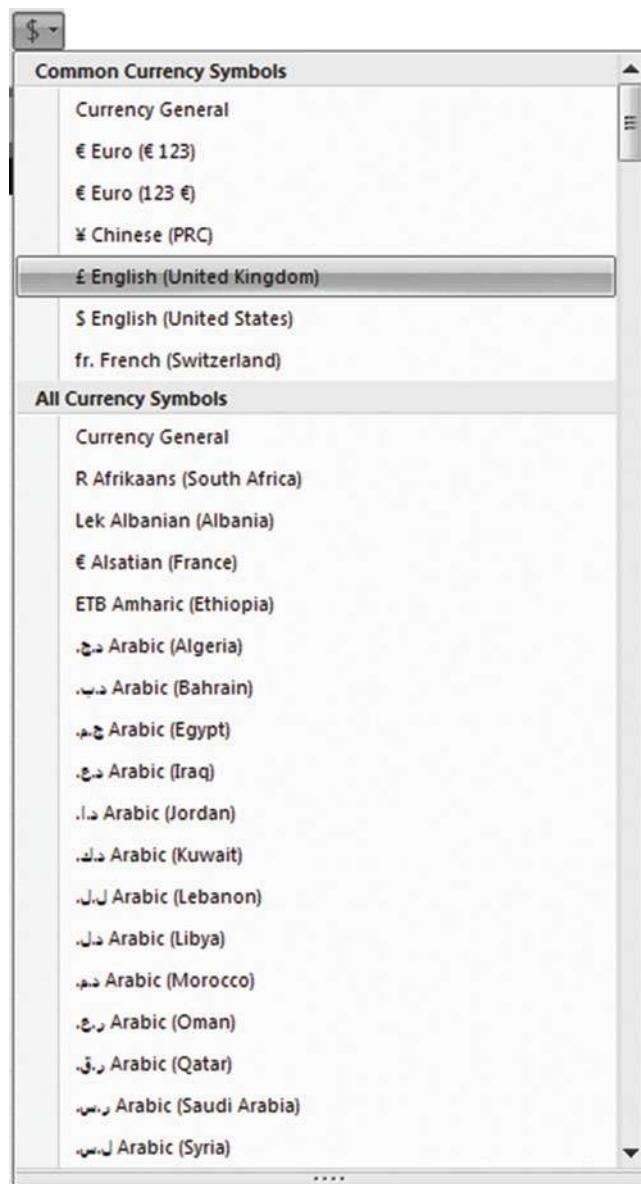


Figure 6-6. The currency format pop-up list

Note The thousands separator that is applied, as well as the decimal separator, depends on the settings of the PC on which the formatting is applied.

Preparing Data for Dashboards

Corralling data into a structure that can power your dashboards necessitates a good few tweaks above and beyond specifying data types and formats for final presentation. As part of the groundwork for your dashboards you could also have to

- Categorize data
- Apply a default summarization
- Define Sort By columns

These ideas probably seem somewhat abstruse at first sight, so let's see them in action to make it clear why you need to add these touches to your data model.

Categorize Data

Power BI dashboards are not just made up of facts and figures. They can also contain geographical data or hyperlinks to web sites or documents. While we humans can recognize a URL pretty easily and we can guess that a column with postcodes contains, well, postcodes, such intuitions may not be quite as self-evident for a computer.

So, if you want Power BI to be able to add maps or hyperlinks (for instance), you will make life easier for both you and the application if you categorize any columns that contain the types of data that are used for maps and links.

For instance, suppose that you want to prepare the Country table a potential data source for a dashboard map (and assuming that you have loaded the DataModeling.pbix sample file).

1. In the Data View, select the Country table in the Fields list. The Country data will appear in the center of the Power BI Desktop window.
2. Click inside the CountryName column. The column will be highlighted.
3. In the Modeling ribbon, click the pop-up to the right of the Data Category button and select Country/Region from the menu.

The ribbon will show Data Category: Country/Region. This means that Power BI Desktop now knows to use the contents of this field as a country when generating maps in dashboards.

The data category options that are available are described in Table 6-4.

Table 6-4. Data Category Options

Data Category Option	Description
Uncategorized	Applies to data that is not used for hyperlinks or creating maps
Address	Specifies an address for mapping
City	Specifies for mapping
Continent	Specifies a continent for mapping
Country/Region	Specifies a country or region for mapping
County	Specifies a county for mapping
Latitude	Specifies a latitude for mapping

(continued)

Table 6-4. (continued)

Data Category Option	Description
Longitude	Specifies a longitude for mapping
Place	Specifies a location or place for mapping
Postal Code	Specifies a postal (Zip) code for mapping
State or Province	Specifies a state or province for mapping
Web URL	Indicates an URL for a hyperlink

Note Not specifying a data category does not mean that Power BI Desktop cannot create maps in dashboards or recognize URLs. However, the results cannot be guaranteed of a reasonable chance of success unless you have indicated to the application that a column contains a certain type of data.

Apply a Default Summarization

When you are creating dashboards, you are often aggregating numeric data. Most times, this means adding up the figures to return the column total. However, there could be columns of data where you want another aggregation applied. Do the following to set your own default aggregation (assuming that you have loaded the DataModeling.pbix sample file).

1. In the Data View, select the Colors table in the Fields list. The Colors data will appear in the center of the Power BI Desktop window.
2. Click inside the Color column. The column will be highlighted.
3. In the Modeling ribbon, click the pop-up to the right of the Default Summarization button and select Count from the menu.

The ribbon will show Default Summarization: Count.

The default summarization options that are available are described in Table 6-5.

Table 6-5. Default summarization Options

Default Summarization Option	Description
Do Not Summarize	The data in this column is not summarized
Sum	The data in this column is added (summed)
Average	The average value for the data in this column is returned
Minimum	The minimum value for the data in this column is returned
Maximum	The maximum value for the data in this column is returned
Count	The number of elements in the column is returned
Count (Distinct)	The number of individual (distinct) elements in the column is returned

Obviously, you can only apply mathematical aggregations to numeric values. However, you can apply counts to any type of data.

Note Specifying a default aggregation does not prevent you overriding the default in dashboards. It merely sets a default that is applied as a standard when aggregating data from a column. In the real world, this can be really useful because it reduces the time you spend building dashboards.

Define Sort by Columns

Sometimes you will want to sort data in a dashboard visualization based not on the contents of the selected column, but by the contents of another column. As an example, imagine that you have a table of data that contains the month for a sale. If you sort by month, you probably do not want to see the months in alphabetical order, starting with April. In cases such as this, you can tell Power BI Desktop that you want to sort the month *name* element by the month *number* that is contained in another column.

1. Load the C:\PowerBiDesktopSamples\DataModeling.pbix sample file (unless you have already loaded it, of course).
2. In the Data View, select the DateDimension table in the Fields list. The date data will appear in the center of the Power BI Desktop window.
3. Click inside the MonthFull column. The column will be highlighted.
4. In the Modeling ribbon, click the Sort By Column button and select MonthNum from the list of available columns.

Now if you sort by MonthFull in a visualization, you see the months in the order that you probably expect—from January to December. Had you *not* applied a Sort By column then calendar months would have been sorted in alphabetical order, which is from April to September! Once again, this choice applies to *any* visualization that you create in a Power BI Desktop dashboard that is based on this data model. So remember to add numeric sort column alongside textual columns, such as dates and so forth, at the source.

Tip If you want to see which column has been set as a Sort By column then all you have to do is to click inside the column to be sorted and then click the Sort By Column button. The pop-up list of columns shows a tick to the left of the column that is being used to sort the selected column.

The sample file for this chapter (DataModeling.pbix) already contains columns that you can use as Sort By columns. In the real world, your source data might not always be this instantly useable. So remember that you can always switch to the Power BI Desktop Query Editor and enhance source tables with extra columns that you can then use to sort data in Data View.

Sorting Data in Power BI Desktop Tables

A Power BI Desktop table could contain millions of rows, so the last thing that you want to have to do is to scroll down through a random dataset. Fortunately, ordering data in a table is simple.

1. Right-click inside the column you want to order the data by. I will choose the Make column in the Stock table.
2. Click the Sort Ascending option in the context menu to sort this column in ascending (alphabetical) order.

The table is sorted using the selected column as the sort key and even a large dataset appears correctly ordered in a very short time. If you want to sort a table in descending order (reverse alphabetical or largest to smallest order), click the Sort Descending option in the context menu.

At this juncture, you need to remember that the data model is not really designed for interactive data analysis. That is what dashboards are for. Consequently, you should not expect to use the data tables in Power BI Desktop as if they were vast Excel spreadsheets.

Tip If you need a visual indication that a column is sorted, look at the right of the column name. You will see a small arrow that faces upward to indicate a descending sort or downward to indicate an ascending sort.

Power BI Desktop alters the text in the sort icon slightly depending on the data type of the column in which you have clicked. This makes the result even more comprehensible, if anything.

- For a numeric column, the icons read Sort Smallest To Largest and Sort Largest To Smallest.
- For a date column, the icons read Sort Oldest To Newest and Sort Newest To Oldest.

If you want to remove the sort operation that you applied and return to the initial dataset as it was imported, all you have to do is click the Clear Sort icon in the context menu for the column.

Note You cannot perform complex sort operations; that is, you cannot sort first on one column, then—carry out a secondary sort in another column (if there are identical elements in the first column). You also cannot perform multiple sort operations sorting on the least important column and then progressing up to the most important column to sort on to get the effect of a complex sort. This is because Power BI Desktop always sorts the data based on the dataset as it was initially loaded. Remember that you can add index columns in Power BI Desktop Query and then sort on these if you want to reapply an initial sort order.

Adding Hierarchies

Organising data can be fundamental when you want to “see the wood for the trees”. Consequently Power BI Desktop lets you create any hierarchy on the fly so that you can better appreciate the structure of the information that you are presenting.

To create a hierarchy:

1. Switch to Report View.
2. Right-click on the field that will become the top-level element in the new hierarchy (Make in the Stock table in this example).
3. Select New Hierarchy. A new hierarchy named Make Hierarchy will appear in the Stock table.
4. Drag the Model field onto the new hierarchy.

This is all that you have to do! You can now drag the hierarchy on to the report canvas to create a table that is based on the multiple elements that make up the hierarchy. The data can be used anywhere-and in any visual-where the multiple fields that make up the hierarchy would be used. They are particularly useful as the basis for matrices and drill-down charts.

You can, of course, rename or delete the hierarchy just as you would any standard data field.

Designing a Power BI Desktop Data Model

Congratulations! You are well on the way to developing a high-performance data model for self-service business intelligence (BI). You have imported data from one or even from several sources into the Power BI Desktop data model. You have taken a good look at your data using Power BI data model window and you can carry out essential operations to rename tables and columns. The final step to ensure that your dataset is ready for initial use as a self-service BI data repository is to create and manage relationships between tables. This is a fundamental part of designing a coherent and useable dataset in Power BI Desktop.

Before leaping into the technicalities of table relationships, we first need to answer a couple of simple questions:

- What are relationships between tables?
- Why do we need them?

Table relationships are links between tables of data that allow columns in one table to be used meaningfully in another table. If you have opened the Power BI Desktop example file DataModeling.pbix, then you can see that there is a table of stock data that contains a ColorID column, but not the actual color itself. As a complement to this, there is a reference table of colors. It follows that, if we want to say what color a car was when it was bought, we need to be able to link the tables so that the stock table can look up the actual color of the car that was sold. This requires some commonality between the two tables and, fortunately, both contain a column named ColorID. So if we are able to join the two tables using this field, we can see which color is represented by the color ID for each car sold, which figures in the sales data.

You can see another example of linking tables if you take a look at the Invoices and InvoiceLines tables. These two tables have been designed using a technique called relational modeling. Essentially this means that two tables have been created to avoid pointless data duplication. So any data that is used to describe an invoice (such as the invoice date or invoice number) is stored in the Invoices table, whereas all the details concerning the vehicles sold are held in a separate table named InvoiceLines. The two tables then share a field that allows them to be joined so that users can see the data from the two tables together if they need to.

It is possible to store the data from these two tables as one table. However, this would mean repeating elements such as the invoice date or invoice number each time that an invoice contained more than one item.

Clearly, these examples are extremely simple. However, they are not unduly contrived. They represent the way many relational databases store data. So there is every chance that you will see potential links or relationships like this in the real-world data that you import from corporate databases. In any case, if you want to use data from multiple sources in your data analysis, you will have to find a way to link the tables using a common field, like the ColorID field that I just mentioned. The reality may be messier (the fields may not have the same name in the two tables, for instance), but the principle always applies.

Tip If you have the necessary permissions as well as the SQL knowledge, then you can join tables directly in the source database using a query. This way you can create fewer “flattened” tables in Power BI Desktop from the start.

Data View and Relationship View

Power BI Desktop lets you see your data model in two different yet complementary ways. When you need a high-level overview of all the data tables, you should use Relationship View, as this allows you to step back from the detail and look at the dataset as a whole. The following explains how to do this.

1. Click the Relationship View button on the left of the Power BI Desktop window.
Power BI Desktop will display the tables in Relationship View, as shown in Figure 6-7.

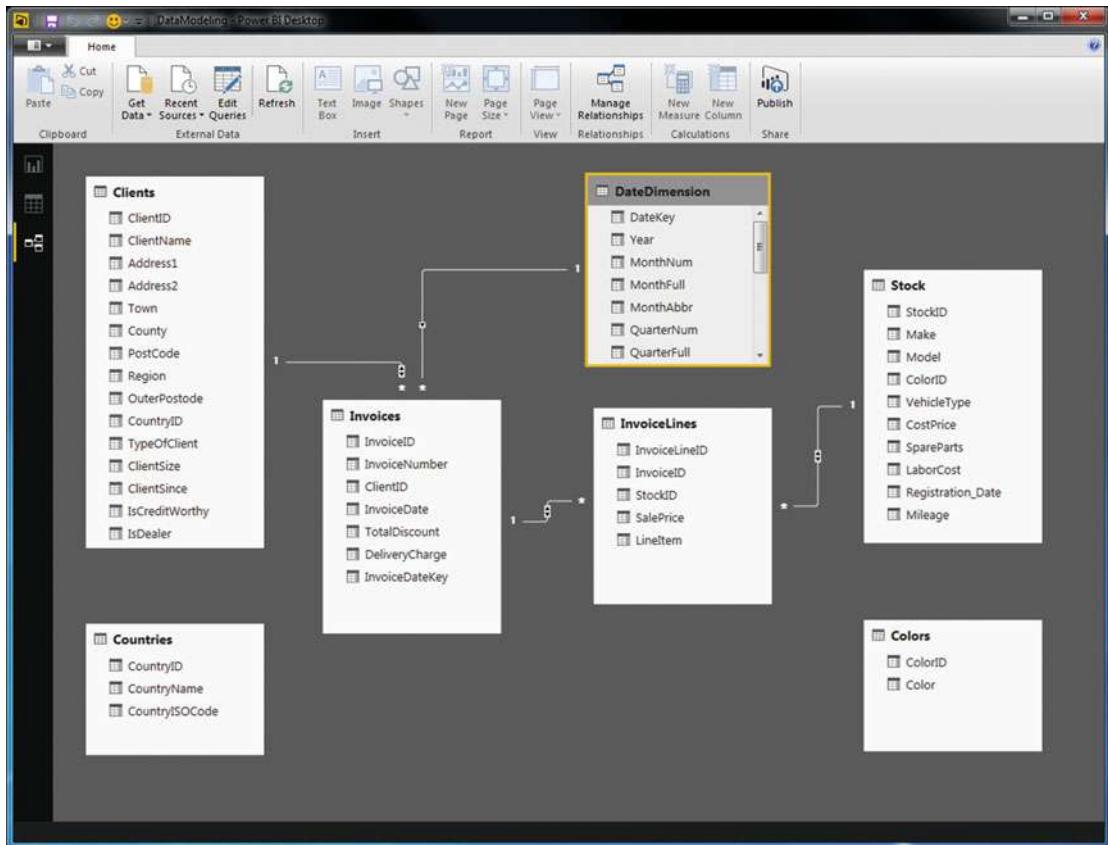


Figure 6-7. Relationship View

As you can see from Figure 6-1, you are now looking at most or all of your tables, and although you can see the table and column names, you cannot see the data. Not only that, but you can also see that some tables are already joined—though not all of them. This is because Power BI Desktop always attempts to guess any possible relationships between tables and creates relationships automatically, if it can, to save you time and effort.

This is exactly what we need, because now it is time to think in terms of overall structures rather than the nitty gritty. You can use this view to move and resize the tables. Moving a table is as easy as dragging the table's title bar. Resizing a table means placing the pointer over a table edge or corner and dragging the mouse.

Although repositioning tables can be considered pure aesthetics, I find that doing so is really useful. A well laid out dataset design helps you understand the relationships between the tables and the inherent structure of the data.

Relationship View Display Options

The whole point of Relationship View is to let you get a good look at the entire dataset and, if necessary, modify the layout in order to see the relationships between tables more clearly.

Maximizing a Table

If you have many, many fields in a table, then you may occasionally need to take a closer look at a single table. Fortunately, the creators of Power BI Desktop have thought of this. So, to zoom in on a specific table:

1. Click the table that you wish to examine more closely. In this example, it will be the DateDimension table.
2. Click the Maximize button at the top right of the table. The table will expand to give you a clearer view of the fields in the table. You can see an example of this in Figure 6-8.

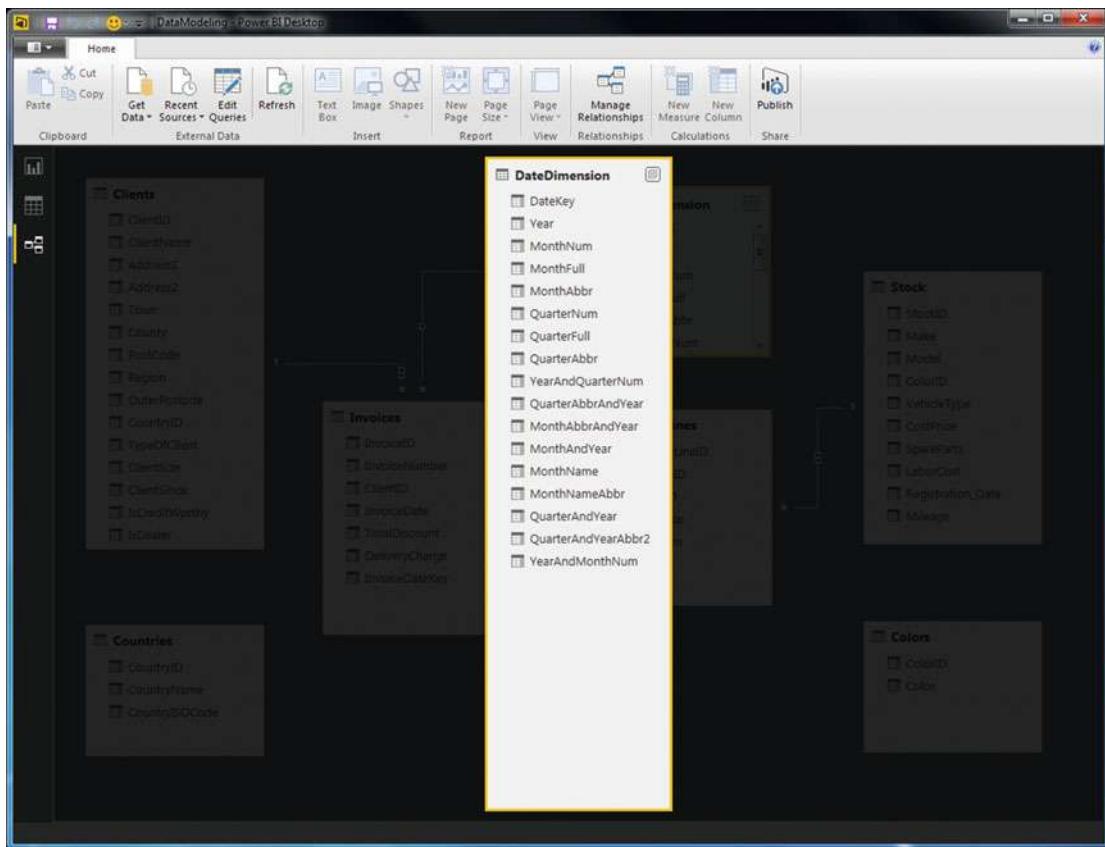


Figure 6-8. Maximizing a table

Minimizing a Table

To reset the table to its previous size, click the same icon—now called the Restore button—at the top right of the table.

Creating Relationships

Creating relationships is easy once you know which fields are common between tables. Since we already agreed that we need to join the Colors table to the SalesData table. Let's look at how to do this.

1. Drag the ColorID field from the Stock table over the ColorID field in the Colors table as shown in Figure 6-9.

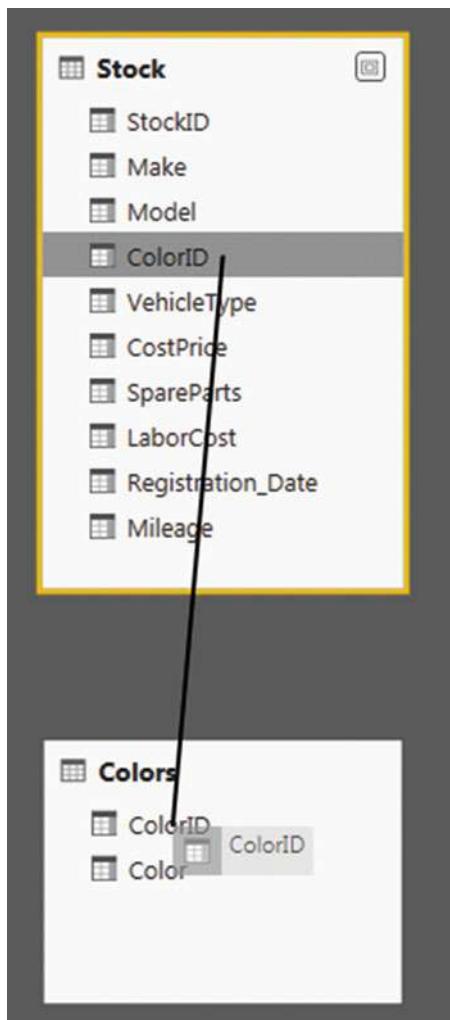


Figure 6-9. A table relationship

This is all that you have to do. The two tables are now joined and the data from both tables can be used meaningfully in reports and dashboards. What is important when creating relationships in this way is that you *always* drag the field that appears many times in a table (the multiple colors for vehicles that are stored in the Stock table) over the *single* example of the field that is held in a “reference” table such as the Colors table. If you try to create a table join by dragging a field that appears in a reference table over a field in another table, you will get an error message like the one shown in Figure 6-10.



Figure 6-10. The relationship error dialog

Normally, you can correct this by dragging the field between the tables in reverse order to the direction that you just attempted.

Note Currently, in the Power BI Desktop data model, you can only join tables on a single field. You may need to take this into account when preparing queries in the Power BI Desktop Query Editor for later use in the data model.

Creating Relationships Manually

You do not have to drag and drop field names to create relationships. If you prefer, you can specify the tables and fields that will be used to create a relationship between tables. What is more, you do not have to be in Relationship View to do this. So, just to make a point and to show you how flexible Power BI Desktop can be, in this example, you will join the Countries and Clients tables on their common CountryID field.

1. Click the Manage Relationships button in the Home ribbon. The Manage Relationships dialog will appear. It should look like Figure 6-11 at the moment.

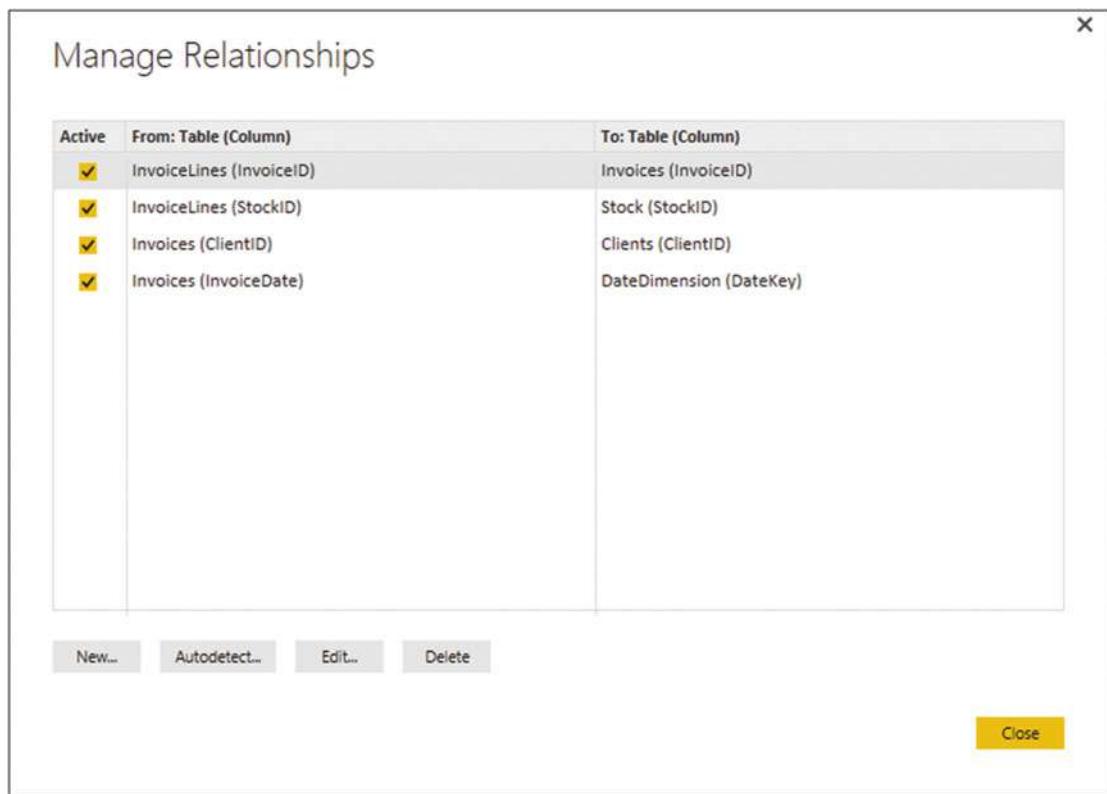
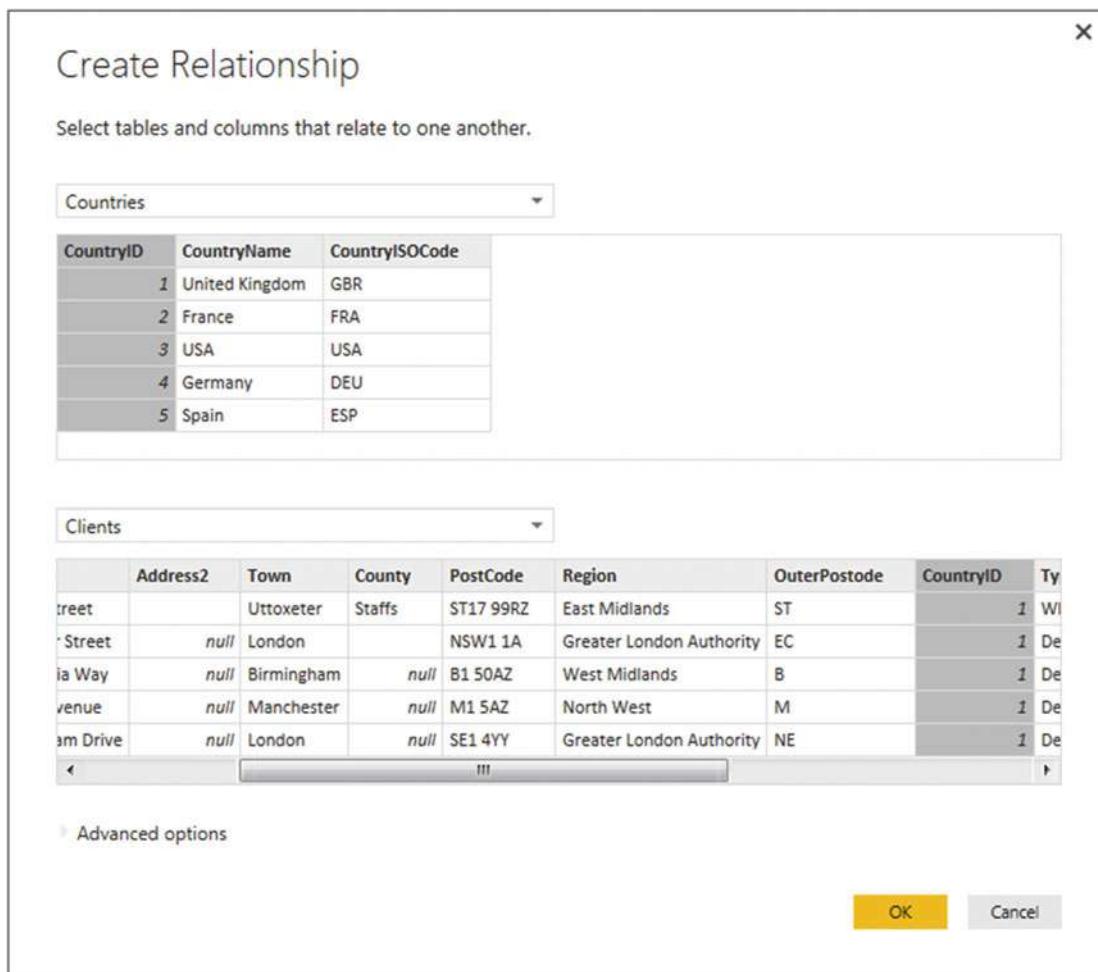


Figure 6-11. The Manage Relationships dialog

2. Click New. The Create Relationship dialog will appear.
3. In the upper part of the dialog, select the Countries table from the pop-up list of tables.
4. Select the CountryID field from the Countries fields.
5. On the lower row, select the Clients table as the Related Lookup Table. The CountryID field should appear automatically as the field to join on (in the Related Lookup Column field). If Power BI Desktop has guessed the field, it will appear selected. If it does not, or if it has guessed incorrectly, you can always select the correct field in the lower part of the dialog. The Create Relationship dialog should look like Figure 6-12.

**Figure 6-12.** The Create Relationship dialog

- Click OK. The Create Relationship dialog will close and return you to the Manage Relationships dialog.
- Click Close. The Manage Relationships dialog will close and the relationship will be created.

Note Creating relationships manually can be easier when you have hundreds of fields or if you want Power BI Desktop to guess which fields to use.

Creating Relationships Automatically

If you are importing several tables from a relational database, then you can have Power BI Desktop to create the relationships during the import process. This approach has a couple of advantages:

- You avoid a lot of manual work.
- You reduce the risk of error (that is, creating relationships between tables on the wrong fields, or even creating relationships between tables that are not related).

This technique is unbelievably easy. You carry out an import from a relational database source (say, using SQL Server). When faced with the list of available tables in the source database (as described in Chapter 9), you do the following:

1. Select the major source table.
2. Click the Select Related Tables button. Any tables that are linked to the table you have selected will be selected.
3. Continue the import process as described previously.

Once you have completed the import, switch to Relationship View. You will see that the tables you just imported already have the relationships generated in Power BI Desktop.

Note If you choose to select related tables, be aware that doing so only selects tables linked to the table(s) that you have already selected. As a result, you may have to carry out this operation several times, choosing a different starting table every time, to force all the existing relationships to be imported correctly.

Deleting Relationships

In addition to creating relationships, you will inevitably want to remove them at some point. This is both visual and intuitive.

1. Click the Design View button in the Home ribbon. Power BI Desktop will display the tables in Relationship View.
2. Select the relationship that you want to delete. The arrow joining the two tables will become a double link, and the two tables will be highlighted.
3. Right-click and choose Delete (or press the Delete key). The Confirmation dialog will appear, as shown in Figure 6-13.

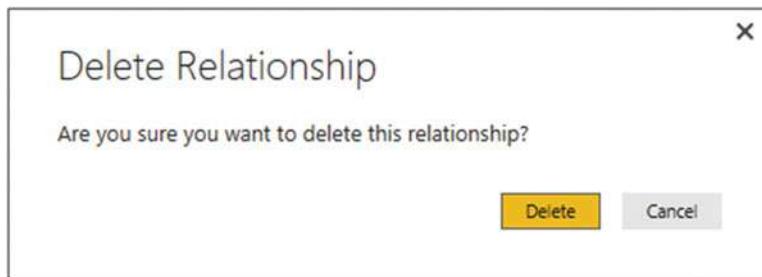


Figure 6-13. The Delete Relationship dialog

4. Click Delete.

The relationship will be deleted and the tables will remain in the data model.

Managing Relationships

If you wish to change the field in a table that serves as the basis for a relationship, then you have another option. You can use the Manage Relationships dialog. This approach may also be useful if you want to create or delete several relationships at once. If you want to use this

1. In the Design tab, click Manage Relationships. The Manage Relationships dialog appears, as was shown previously in Figure 6-11.
2. Click the relationship you wish to modify.
3. Click Edit. The Edit Relationship dialog appears (it is virtually identical to the Create Relationship dialog shown in Figure 6-12).
4. Continue modifying the relationship as described previously.

As you can see from this dialog, you also have the option of creating or deleting relationships. Since the processes here are identical to those I have already described, I will not repeat them.

Note If you delete a set of related tables and subsequently reimport them without importing the relationships, then Power BI Desktop will *not* remember the relationships that existed previously. Consequently, you will have to re-create any relationships manually. The same is true if you delete and reimport any table that you linked to an existing table in Power BI Desktop—once a relationship has been removed through the process of deleting a table, you will have to re-create it.

The techniques used to create and manage relationships are not, in themselves, very difficult to apply. It is nonetheless *absolutely fundamental* to establish the correct relationships between the tables in the dataset. Put simply, if you try to use data from unconnected tables in a single Power BI Desktop dashboard, you will not only get an alert warning you that relationships need to be created, you will also get visibly inaccurate results. Basically, all of your analysis will be false. So it is well worth it to spend a few minutes upfront designing a clean, accurate, and logically coherent dataset.

Deactivating Relationships

If you do longer need a relationship between tables but do not want to delete it you also have the option of deactivating the relationship. This means that the relationship no longer functions, but that you can reactivate it quickly and easily.

Deactivating—or reactivating—a relationship as simple as selecting or unselecting the box in the Active column of the Manage Relationships dialog.

Advanced Relationship Options

The Edit Relationship dialog contains a few advanced options that you could find useful on occasion. To access these, you need to expand the Advanced options section in this dialog by clicking either the small triangle to the left of Advanced options or the title Advanced options itself. Doing this extends the Edit Relationship dialog, with the further possibilities that you can see in Figure 6-14.

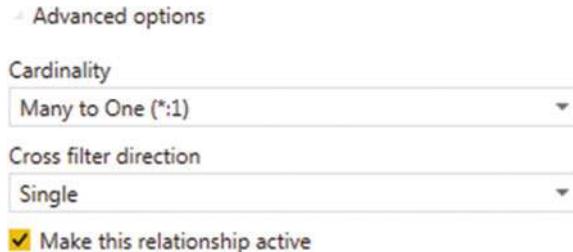


Figure 6-14. Advanced options for relationships

These various options are described in Table 6-6.

Table 6-6. Power BI Desktop Relationship Types

Relationship Option	Type	Description
Many-to-One	Cardinality	Specifies that there are many records in one table for a single record that maps in the table that is joined
One-to-One	Cardinality	Specifies that there is a single record in one table for a single record that maps in the table that is joined
One-to-Many	Cardinality	Specifies that there is a single record in one table for a many records that map in the table that is joined
Single	Cross filter direction	Lets you apply a filter to the second table in a join relationship and see the effect on the main table
Both	Cross filter direction	Lets you apply a filter to either table in a relationship and filter the results in the other table

Managing relationships in Power BI Desktop is often the key to creating an efficient data model. It is, however, outside the scope of this book to provide a complete course in data modeling. Nonetheless, here are a few tips to bear in mind when creating your initial data models.

- It can help to think in terms of *main* tables and *lookup* tables. In the data model that you have looked at in this chapter, you can consider certain tables as lookup tables for the main data, such as the Colors, Countries, and Clients tables.
- Lookup tables generally contain a series of values that are not repeated in the table (a list of countries, for instance). These values are called the “one” side of a relationship and they are linked to another table where they are referred to on many occasions. Hence the table that contains the multiple references is called the “many” side of the relationship. This is also called the *cardinality* of a relationship by database and data warehouse designers; it is the term that Power BI Desktop uses in the Edit Relationship dialog, as you can see in Figure 6-14.
- If your source data contains many lookup tables that cascade down through a series of relationships (a classic case is the Category ► Subcategory ► Product hierarchy that you find in many retail environments). To avoid over-complicating the data model you may prefer to merge multiple tables into a single table using Power BI Desktop Query *before* developing the data model in Power BI Desktop itself.
- Sometimes—and this can be the case when importing data from relational databases—you need to join tables on more than one field. This is not possible in Power BI Desktop. However, you can often find workarounds to this, again using Power BI Desktop Query before modelling the data. In cases like this, you can merge tables by creating joins using multiple columns (as you saw in the previous chapter), for instance.
- Data imported from data warehouses can have a built-in structure of facts (main tables containing metrics) and dimensions (containing lookup elements). However, you may want to “flatten” complex hierarchies of dimensions and create single-level tables of lookup elements here, too, using Power BI Desktop Query.
- Sometimes you may want to use the same table twice in different contexts. For instance, a date table may be useful to join to a sale date and a purchase date. In cases like this, you can reimport the date table a second time (and give it another name) and then create two separate joins from a lookup table to the two different lookup tables. This allows you to filter and aggregate data by separate date criteria. A lookup table like this is often called a *role-playing dimension*.
- It is possible to create multiple relations between one table and another, and to specify that only one of them is active. You can then force a calculation to apply a nonactive relationship using the `USERELATIONSHIP()` function in DAX. However, we are already starting to reach more complex levels of data modeling, so I will only mention the possibility here.

Conclusion

This chapter was all about taking the clean data that you had prepared using Power BI Desktop Query and molding it into a structured and coherent data model that will be the basis for your dashboards and reports.

To begin with, you saw how to look at the whole dataset that was now available in the data model. This included sorting the data and adjusting column widths.

Then you learned how to ensure that each column was defined as having the appropriate data type. After that you applied any number formats that would be required in future dashboard elements directly to the data model. You also saw how to prepare certain types of column for use in maps or to provide hyperlinks.

Finally, you learned how to pull all the disparate data sources together in a joined-up data model that has now become the basis for some in-depth analytics. All that you have to do now is add any calculated metrics that your reports need. This is the subject of the next chapter.

CHAPTER 7



Extending the Data Model with Calculated Columns

This chapter further develops the data model that you created in the previous chapter. It explains how to augment the existing tables by adding new columns containing calculations to the tables that you have imported. You can then apply these additional metrics to the dashboards that you create using Power BI Desktop.

Admittedly, not every data model in Power BI Desktop needs extensive calculations. Frequently, the data can speak for itself without much polishing. Yet business intelligence is, at its heart, based on figures. Consequently, sooner or later, you need to apply simple math, calculate percentages, or compare figures over time. You may even want to develop more complex formulas that enable you to extend your analyses and illustrate your insights. Fortunately, Power BI Desktop makes these—and many, many other calculations—amazingly easy. What's more, if you are an Excel user, you will probably find most of the techniques explained in this chapter to be totally intuitive.

In some cases, you only need to cherry-pick techniques from the range of available options to finalize a dataset. So, it probably helps to know what Power BI Desktop can do and when to use the techniques outlined in this chapter. Therefore, I leave it to you to decide what is fundamental and what is useful. The objective in this chapter and the next two is to present a tried and tested suite of calculation solutions so you are empowered to deal with the range of the potential challenges that you may encounter in your data analysis.

All calculations in the Power BI Desktop data model are written using a simple language named DAX. This stands for **D**ata **A**nalysis **X**pressions. As you will see, DAX is not in any way a complex programming language. Indeed, it is known as a *formula language* because it is a set of some 260 formulas that you can use and combine to extend data models and to create metrics to underpin the visualizations in your dashboards. Fortunately, DAX formulas are loosely based on the formulas in Excel (indeed a good third of them are identical) so the learning curve for an Excel user is really quite short.

Given the vast horizons that DAX opens up to Power BI Desktop users, a single chapter could never be enough to give you a decent idea of the practical uses of this formula language. So the introduction to DAX in this book is spread over three chapters. To apply some structure to a potentially huge and amorphous area, I have broken down DAX into the following areas:

- This chapter covers *column-based calculations* (where the formula appears as a new column in a table)
- Chapter 8 describes *measures* (calculations that are added to a table but that do not add a column of calculated data)
- Chapter 9 describes *time calculations* (measures that are used to aggregate data over time periods or to compare data over time)

If you want to continue enhancing the data based on the kinds of data transformations that you saw in the previous few chapters, download the PowerBIDataModel.pbix file from the Apress web site. This file lets you follow the examples as they appear in this chapter.

Types of Calculations

If you are lucky, then the data that you have imported contains everything that you need to create all the visualizations you can dream up in Power BI Desktop. Reality is frequently more brutal than that, however, and it necessitates adding further metrics to one or more tables. These calculated metrics will extend the data available for visualization. This is fundamental when you are using tools such as Power BI Desktop that do not allow you to add calculated elements to the output, but insist that all metrics—whether they are source data or calculated metrics—exist in the dataset. This is less of a constraint and more of a nod toward good design practice, because it forces you to develop calculations once and to place them in a single central repository. It also reduces the risk of error, because users cannot develop their own (possibly erroneous) metrics and calculations and so distort the truth behind the data.

When creating DAX metrics, you are defining elements that are of practical use for your dashboard visualizations. This can include

- Creating derived metrics that will appear in visualizations.
- Adding elements that you use to filter pages or visualizations.
- Creating elements that you use to segment or classify data. This can include creating your own groupings.
- Defining new metrics based on existing metrics.
- Adding your own specific calculations (such as accounting or financial formulas).
- Adding weightings to values.
- Ranking and ordering data.

And many, many more...

New Columns

Adding new columns is one of the two ways in which you can extend a dataset with derived metrics that you can use in Power BI Desktop dashboards. There are multiple reasons why you may need further columns, including:

- Concatenating data from two existing columns into one new column
- Performing basic calculations for every row in the table, such as adding or subtracting the data in two or more columns
- Extracting date elements such as the month or year from a date column and adding them as a new column
- Extracting part of the data in a column into another column
- Replacing part of the data in a column with data from another column
- Creating the column needed to apply a visually coherent sort order to an existing column
- Showing a value from a column in a linked table inside the source table

Indeed, the list could go on...

Before you start wondering exactly what you are getting yourself into, I want to add a few words of reassurance about the ways in which a data model can be extended.

- First, extending a table with added columns is designed to be extremely similar to what you would do in Excel. Consequently, you are in all probability building on your existing knowledge as a data and dashboard power user.
- Second, the functions that you will be using are, wherever possible, similar to existing Excel functions. This does not mean that you have to be an Excel super user to add a column, but that knowledge gained using Excel will help with Power BI Desktop and vice versa.
- Finally, most of the basic table extension techniques follow similar patterns and are not complex. So the more you work at adding columns, the easier it will become as you reuse and extend techniques and formulas.

Creating columns is a bit like creating a formula in Excel that you then copy down over the entire column. They are even closer to the derived columns that you can add to queries in Access. The key thing to note is that any formula will be applied to the *entire* column.

It is worth noting from the start that a formula that you add to a new column is calculated and applied to a column when it is created. It is only recalculated if you recalculate the entire table or file or if you refresh the source data.

Note In this chapter and the next, I frequently emphasize that you need to prepare your metrics *before* building visualizations for dashboards and reports. In practice, Power BI Desktop is extremely forgiving and immensely supple. It lets you switch from Dashboard View to Data View at any time so that you can add any new columns or missing metrics. Indeed, you can even add measures from the Fields list in Dashboard View. However, it can be more constructive to think through all your data requirements before rushing into the fun part that is creating dashboards. This approach can save you creating duplicate measures and can help you to adopt a clear and coherent approach to naming metrics as well.

Naming Columns

If you create new columns, then you need to give them names. Inevitably, there are a few minor limitations on the names that you can apply. So, rather than have Power BI Desktop cause problems, I prefer to explain the overall guidelines on the Power BI Desktop naming conventions earlier rather than later in the course of this chapter.

The first thing to remember is that column names have to be *unique* inside each table. Therefore, you *cannot* have two columns with the same name inside the *same* table. You *can*, however have two columns that share a name if they are in separate tables. However, I generally advise that you try to keep column names unique across all the tables in a Power BI Desktop file if you possibly can. This can make building visualizations easier and safer because you do not run the risk of using a column from the “wrong” table in a chart, for instance, and getting entirely inappropriate results as a consequence.

The essential point of note is that columns cannot contain any of the following:

- Spaces (unless the spaces are enclosed by brackets or single apostrophes).
- The characters: . , ; ' : / \ * | ? & % ! + = () [] { } < >

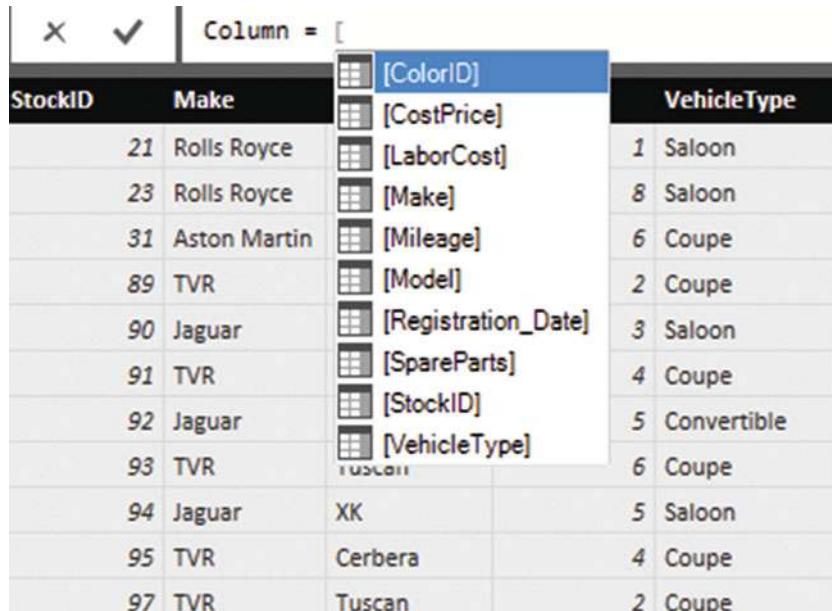
Fortunately, column names are *not* case sensitive.

Note All the restrictions that concern column names also apply to measures (which you will learn about in the next chapter).

Concatenating Column Contents

As an initial example of a New Column, I will presume that, when working with Power BI Desktop to create a dashboard for Brilliant British Cars, you have met a need for a single column of data that contains both the make and model of every car sold. Because the data we imported contains this information as separate columns, we need to add a new column that takes the data from the columns Make and Model, and joins them together (or concatenates them, if you prefer) in a new column. The following explains how this is done.

1. In the Power BI Desktop window, make sure that you are in Data View.
2. Click the Stock table to select it.
3. In the Modeling ribbon, click the New Column button. A new empty column will appear to the right of the final column of data. This column is currently entitled Column and is highlighted.
4. The formula bar above the table of data will display Column = .
5. In the formula bar, click to the right of the equals sign. Press [. A list of the fields in the current table will appear, as shown in Figure 7-1.



The screenshot shows the Power BI Desktop interface with the formula bar open. The formula bar displays "Column = [". To the right of the equals sign is a dropdown menu containing a list of fields from the current table: [ColorID], [CostPrice], [LaborCost], [Make], [Mileage], [Model], [Registration_Date], [SpareParts], [StockID], and [VehicleType]. The field [ColorID] is highlighted in blue, indicating it is selected for use in the formula.

StockID	Make	VehicleType
21	Rolls Royce	1 Saloon
23	Rolls Royce	8 Saloon
31	Aston Martin	6 Coupe
89	TVR	2 Coupe
90	Jaguar	3 Saloon
91	TVR	4 Coupe
92	Jaguar	5 Convertible
93	TVR	6 Coupe
94	Jaguar	5 Saloon
95	TVR	4 Coupe
97	TVR	2 Coupe

Figure 7-1. Selecting a field for a formula

6. Click the [Make] field. The formula bar now reads =[Make].
7. In the formula bar, add & " " &. The formula bar now reads =[Make] & " " &.
8. Press [. Select **[Model]** from the list of fields. The formula bar now reads =[Make] & " " & [Model].
9. Press Enter (or click the tick icon in the formula bar). The column is automatically filled with the result of the formula and it shows the make and model of each car sold.
10. Right-click the column header for the new column and select Rename.
11. Type the word **Vehicle** and press Enter.

The table will now look something like Figure 7-2. Moreover, the new column has been added as a field to the Stock table in the Fields list.

StockID	Make	Model	ColorID	VehicleType	CostPrice	SpareParts	LaborCost	Registration_Date	Mileage	Vehicle
21	Rolls Royce	Silver Seraph	1	Saloon	62000	400	£951.00	01 January 1999	52500	Rolls Royce Silver Seraph
23	Rolls Royce	Silver Shadow	8	Saloon	62000	400	£750.00	08 January 1985	52500	Rolls Royce Silver Shadow
31	Aston Martin	DB4	6	Coupe	75890	400	£147.00	08 September 2006	52500	Aston Martin DB4
89	TVR	Tuscan	2	Coupe	37500	400	£325.00	20 September 2006	52500	TVR Tuscan
90	Jaguar	XK	3	Saloon	37500	400	£325.00	09 May 2007	52500	Jaguar XK
91	TVR	Cerbera	4	Coupe	37500	400	£325.00	20 September 2006	52500	TVR Cerbera
92	Jaguar	XK	5	Convertible	37500	400	£325.00	09 May 2007	52500	Jaguar XK
93	TVR	Tuscan	6	Coupe	37500	400	£325.00	20 September 2006	52500	TVR Tuscan
94	Jaguar	XK	5	Saloon	37500	400	£325.00	20 September 2006	52500	Jaguar XK
95	TVR	Cerbera	4	Coupe	37500	400	£325.00	20 September 2006	52500	TVR Cerbera
97	TVR	Tuscan	2	Coupe	37500	400	£325.00	09 May 2007	52500	TVR Tuscan
98	Jaguar	XK	2	Coupe	37500	400	£325.00	20 September 2006	52500	Jaguar XK
103	Jaguar	XK	6	Coupe	62000	400	£250.00	20 September 2005	52500	Jaguar XK
104	Jaguar	XK	5	Saloon	62000	400	£250.00	20 September 2005	52500	Jaguar XK
110	Jaguar	XK	2	Saloon	62000	400	£250.00	20 September 2005	52500	Jaguar XK
112	MGB	GT	3	Coupe	4500	400	£325.00	20 September 2005	52500	MGB GT
118	MGB	GT	2	Coupe	4500	400	£325.00	20 September 1974	52500	MGB GT
119	MGB	GT	3	Coupe	8500	400	£325.00	20 September 1974	52500	MGB GT

Figure 7-2. An initial calculated column

I imagine that if you have been using Excel for any length of time, then you might have a strong sense of déjà vu after seeing this. After all, what you just did is virtually what you would have done in Excel. All you have to remember is that

- Any additional columns are added to the *right* of the existing columns. You *cannot* move them elsewhere in the table once they have been created.
- All functions begin with the equals sign.
- Any function can be developed and edited in the formula bar at the top of the table.
- Reference is always made to *columns*, not to cells (as you would in Excel).
- Column names are always enclosed in square brackets.
- You can nest calculations in parentheses to force inner calculations before outer calculations—again, just as you would in Excel.

Once a new column has been created, it remains at the right of any imported columns in the table where you added it. It is not possible to move the new column elsewhere in the table. The field that it represents is always added to the bottom of the collection of fields for this table in the Fields list. This way the available fields will appear in the order that they were created. However, fields always appear in alphabetical order in the Fields list in Dashboard View.

If you look closely at the field that was added, you will notice that there is a tiny Fx icon to its left. This is how you can distinguish new columns from other fields such as numeric fields (which have a sigma (Σ) icon to their left) or measures (that you will meet in the next chapter) that have a small calculator icon to their left.

Note In this example, you selected columns from the pop-up list of the available columns in the table. You can enter the column name in the formula bar if you prefer, but if you do then you *must* enclose the column name in square brackets. You must also enter it *exactly* as it appears in the Fields list and column title.

Tweaking Text

In Chapter 4, you learned many techniques that you can apply to text-based columns. If you remember, these included changing the capitalization and removing extra spaces (among other things).

DAX also lets you clean up and modify the text in the tables that you have imported into your data model. Indeed, it offers a wide range of functions that you can apply to standardize and cleanse text in tables. As an example, let's imagine that you want to create a column in the Clients table that contains a shortened version of each town. In fact, what you want to do is extract a three-letter acronym from the first letters of the town name, which you can use later in charts.

1. In the Power BI Desktop window, make sure that you are in Data View.
2. Click the Stock table to select the Clients table.
3. In the Modeling ribbon, click the New Column button. A new empty column will appear to the right of the final column of data. This column is currently entitled Column and is highlighted.
4. The formula bar above the table of data will display Column = .
5. In the formula bar, click to the right of the equals sign.
6. Click to the right of the equals sign.
7. Type **LEFT()**. Once the function appears in the pop-up menu, you can select it if you prefer.
8. Click inside the Town column of the Clients table. Clients[Town] will appear in the formula bar. Alternatively, you can type a left square bracket and select the [Town] field.
9. Enter a comma.
10. Enter the number **3**. This indicates to the LEFT() function that it is the *three* characters on the left that you want to isolate for each row in this column.
11. Add a right parenthesis.

12. Still inside the formula bar, replace Column with **TownAbbreviation**. The formula Bar will read

```
TownAbbreviation = LEFT(Clients[Town],3)
```

13. Press Enter (or click the tick icon in the formula bar). The column is automatically filled with the result of the formula and it shows the first three letters of every town's name.

As you can see, the LEFT() function takes two parameters:

- First, the field from which you want to extract the leftmost characters.
- Second, the number of characters to extract.

And that is all that you have to do. By applying a simple text formula, you have prepared a column of text for effective use in a visualization.

DAX contains a couple of dozen functions that you can apply to the text in columns. Most of them follow the same principle as the LEFT() function in that they take at least two parameters, the first of which is the column that you want to take as the basis for your new column and the second (or even third) parameters provide information about how the modification is to be applied. Since I do not have space to explain each and every one of these functions, Table 7-1 contains a succinct overview of a selection of some of the most useful text functions. This table does not explain all the subtleties of every function, but is destined to be both a brief introduction and a starting point for your DAX formulas that rework the text elements of your data tables.

Table 7-1. Core Power BI Desktop Text Functions

Function	Description	Example
LEFT()	Extracts a specified number of characters from the left of a column.	LEFT(Clients[Town], 3)
RIGHT()	Extracts a specified number of characters from the right of a column.	RIGHT(Invoice[InvoiceNumber], 12)
MID()	Extracts a specified number of characters (the second parameter) from a specified position defined by the number of characters from the left (the first parameter) inside a column.	RIGHT(Invoice[InvoiceNumber], 10, 4)
UPPER()	Converts the data to uppercase. This function takes no parameters.	UPPER(Clients[ClientName])
LOWER()	Converts the data to lowercase. This function takes no parameters.	LOWER(Clients[ClientName])
TRIM()	Removes any extra spaces (trailing or leading) from the text inside a column. This function takes no parameters.	TRIM(Clients[Address1])
LEN()	Counts the number of characters in a column. This is often used with the MID() function. This function takes no parameters.	LEN(Clients[Address1])

(continued)

Table 7-1. (continued)

Function	Description	Example
FIND()	Gives the starting point (as a number of characters) of a string inside a column. This function is case sensitive. Interestingly the first parameter is the text to find and the second is the column.	FIND('Car',Clients[ClientName])
SEARCH()	Gives the starting point (as a number of characters) of a string inside a column. This function is not case sensitive and disregards accents. Interestingly the first parameter is the text to find and the second is the column.	SEARCH('Car',Clients[ClientName])
SUBSTITUTE()	Replaces one text with another inside the column. This is a bit like the search and replace function in a word processor.	SUBSTITUTE(Clients[ClientName], 'Car', 'Vehicle')
VALUE()	Converts a figure in a text column to a numeric data type. This function takes no parameters.	VALUE(Colors[ColorID])
FIXED()	Takes a number and rounds it to a specified number of decimals then converts it to a text. The second parameter indicates the number of decimals to apply.	FIXED(Stock[LaborCost], 2)

There are a couple of points to note now that you have seen how to use DAX formulas in Power BI Desktop:

- Functions need not apply to entire columns; they can be applied to specific texts as part of a more complex formula.
- You can enter functions in uppercase or lowercase.

Simple Calculations

To extend the basic principle and to show a couple of variations on a theme, let's now add a calculation to the Stock table. More precisely, I assume that our Power BI Desktop visualizations frequently need to display the figure for the direct costs relating to all vehicles purchased, which I define as being the purchase price plus any related costs. You obtain this by applying a variation on a technique that you have used before,

1. In Data View, select the Stock table.
2. Click New Column in the Modeling ribbon.
3. In the formula bar, replace the word Column with the word **Direct Costs**. (Notice that there is a space, because column names can contain spaces.)
4. To the right of the equals sign, enter a left square bracket: [. The list of the fields available in the Stock table will appear in the formula bar.
5. Type the first few characters of the column that you want to reference—**CostPrice** in this example. The more characters you type, the fewer columns are displayed in the list.
6. Click the column name. It will appear in the formula bar (including the right bracket).

7. Enter the minus sign.
8. Enter a left parenthesis.
9. Enter a left square bracket and select the SpareParts column.
10. Enter a plus sign.
11. Enter a left square bracket and select the LaborCost column.
12. Enter a right parenthesis. This corresponds to the left parenthesis before the SpareParts field. The formula should read

$$\text{DirectCosts} = [\text{CostPrice}] - ([\text{SpareParts}] + [\text{LaborCost}])$$
13. Click the tick box in the formula bar (or press Enter). The new column will be created. It will also appear as a new field in the Fields list for this table.

As you can see, using arithmetic in calculated columns in Power BI Desktop is almost the same as using calculating cells in Excel. If anything, it is easier because you do not have to copy the formula over hundreds or even thousands of rows as the formula is automatically applied to every row in the table.

When selecting fields in steps 5, 9, and 11 you can (if you prefer) click the field name in the Fields list rather than enter a left bracket and scroll through a list of fields. Of course, this assumes that you have not hidden the Fields list and that you have expanded the table name so that you can see the fields that it contains.

If you are a PowerPivot user, then the sense of *déjà-vu* is probably so total as to be overwhelming. In fact most PowerPivot users will probably only need to skim through this chapter as the Data View of Power BI Desktop is virtually identical to the PowerPivot window in Excel. Except for the tables that now appear on the right (instead of tabs at the bottom of the window) and the absence of a formula button, there are few differences. So feel free to jump over any sections (in this chapter and the next) that you already know by heart if you are a PowerPivot expert.

Note You can give a new column an appropriate name either when you create the formula initially (by replacing the word Column with the new column name) or by renaming the column once the formula is correct and confirmed. You can include spaces in column names if you want. After all, this is how the name will appear in your dashboards.

Math Operators

For the sake of completeness, and in case there are any newcomers to the world of Microsoft products out there, I prefer to recapitulate the core math functions that are available in Power BI Desktop. These are given in Table 7-2.

Table 7-2. Core Power BI Desktop Math Operators

Operator	Description	Example
+	Adds two elements	[SpareParts] + [LaborCost]
-	Subtracts one element from another	[CostPrice] - [SpareParts]
/	Divides one element by another	[CostPrice] / [SpareParts]
*	Multiplies one element by another	[CostPrice] * 1.5
^	Raises one element to the power of another	[CostPrice] ^ 2

If you are working in BI then you are certainly able to perform basic math operations. Consequently, I will not explain things you most likely already know. Just use the same arithmetical operators as you would use in Excel and, after a little practice, you should be able to produce calculated columns with ease. Remember that you have to enclose in parentheses any part of a formula that you want to have calculated before the remainder of the formula. This way, you will avoid any unexpected results in your dashboards.

Rounding Values

You already saw in Chapter 4 that Power BI Desktop Query can round and truncate values when you are preparing data ready for loading into a data model. In practice, of course, you might not yet be aware that you need to tweak your data at this stage. Fortunately, DAX also contains a range of functions that can be used to round values up and down, or even to the nearest hundred, thousand, or million, if need be.

As an example of this, I take the column Direct Costs that you created previously and round it to the nearest integer. This way, you also learn how to modify a formula in DAX.

1. In Data View, select the Stock table.
2. Click inside the Direct Costs column. The column will be selected and the formula that you created previously will appear in the formula bar.
3. Click inside the formula bar to the right of the equals sign.
4. Enter **Round(**. You can also select the formula from the pop-up if you prefer.
5. Click at the right of the formula in the formula bar.
6. Enter a comma.
7. Enter a **0**.
8. Add a right parenthesis to complete the **ROUND()** function. You will see that the corresponding left parenthesis is highlighted in the formula bar to help you track which pair of parentheses is which.
9. Click the tick box in the formula bar (or press Enter). The formula will be modified and any decimals removed from the data in the column.

This example introduced the **ROUND()** function. It will round a value (whether calculated or loaded from a data source) to the number of decimals specified as the second parameter of the function, which is zero in this example.

ROUND() is only one of the functions that you can choose when truncating or rounding values. The DAX functions that carry out rounding and truncation are given in Table 7-3.

Note Remember that using the ROUND() function *modifies* the data, whereas formatting numbers only changes their appearance.

Table 7-3. DAX Rounding and Truncation Functions

Function	Description	Example
ROUND()	Rounds the value to 0 if the second parameter is zero. If the second parameter is greater than zero, the function rounds the value to the number of decimals indicated by the second parameter. If the second parameter is less than zero, the figure to the left of the decimal is rounded to the nearest 10 (for a second parameter of -1, 100 (for a second parameter of -2, and so forth).	ROUND([CostPrice], 2)
ROUNDDOWN()	Rounds the value down to 0 if the second parameter is zero. If the second parameter is greater than zero, the function rounds the value down to the number of decimals indicated by the second parameter. If the second parameter is less than zero, the figure to the left of the decimal is rounded down to the nearest 10 (for a second parameter of -1, 100 (for a second parameter of -2, and so forth). The value is always rounded down; never up.	ROUNDDOWN([CostPrice], 2)
ROUNDUP()	Rounds the value up to 0 if the second parameter is zero. If the second parameter is greater than zero, the function rounds the value up to the number of decimals indicated by the second parameter. If the second parameter is less than zero, the figure to the left of the decimal is rounded up to the nearest 10 (for a second parameter of -1, 100 (for a second parameter of -2, and so forth). The value is always rounded down; never up.	ROUNDUP([CostPrice], 2)
MROUND()	Rounds the value to the nearest multiple of the second parameter.	MROUND([CostPrice], 2)
TRUNC()	Removes the decimals from a value.	TRUNC([CostPrice])
INT()	Rounds down (or up, if the number is negative) to the nearest integer.	INT([CostPrice])
FLOOR()	Rounds down to the nearest multiple of the second parameter.	FLOOR([CostPrice], .2)
CEILING()	Rounds up to the nearest multiple of the second parameter.	CEILING([CostPrice], .2)
FIXED()	Rounds a value to the number of decimals indicated by the second parameter and converts the result to text.	FIXED([CostPrice], 2)

Calculating Across Tables

If your data model is not complex (particularly if it consists of a single table), then most calculations should be simple. All you have to do is follow the principle of building math expressions using column names and arithmetic operators.

The real world of data analysis is rarely this uncomplicated. In most cases, you have metrics on one table that you need to apply in a calculation in a completely different table. Power BI Desktop makes these “cross-table” calculations really easy, *if* you have defined a coherent data model. (This can be done even if tables are not joined in a data model, as you will discover in the next chapter.)

As an example, let’s look at how to subtract the Stock table’s Direct Costs column from the InvoiceLines table’s SalePrice column to calculate the margin on sales. To do this, we will add a new column, called Gross Margin, to the InvoiceLines table.

1. In Data View, select the InvoiceLines table.
2. Click New Column in the Modeling ribbon.
3. In the formula bar, replace the word Column with the words **Gross Margin**.
4. To the right of the equals sign, enter a left square bracket: [. The list of the fields available in the InvoiceLines table will appear in the formula bar.
5. Select the SalePrice field. (Remember that you can type the first few characters to limit the pop-up list of fields to those most closely resembling the field that you are looking for.)
6. Enter a minus sign (you can add spaces before and/or after it, if you want).
7. Start typing the keyword **RELATED**, and select this function once you have limited the selection of functions in the pop-up list. (Alternatively, you can type the whole word and a left parenthesis.) The pop-up list will list all the fields of all the tables that can be joined to the current table in the data model. The pop-up list will look like Figure 7-3.

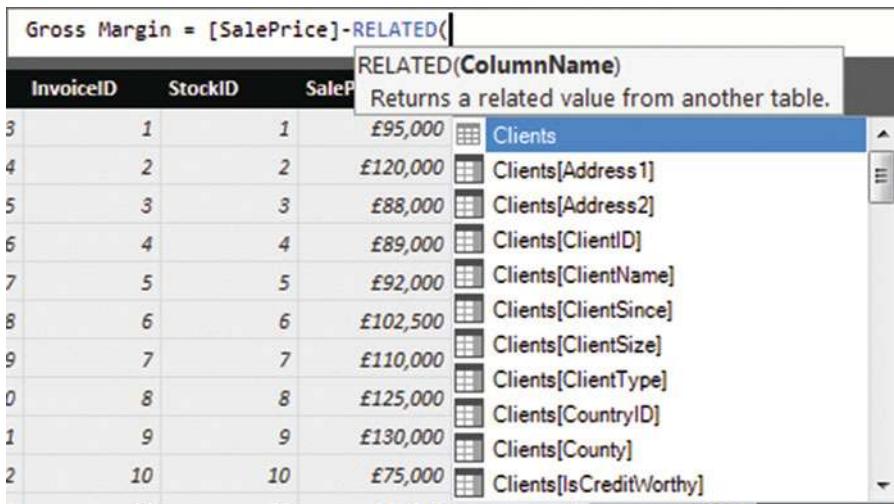


Figure 7-3. The pop-up list of related tables and fields

8. Scroll down through the list and select the field Direct Costs in the Stock table. You can jump directly to the Stock table by typing the first few characters of the table name.
9. Enter a right parenthesis. The formula bar should contain the following formula:

$$\text{Gross Margin} = [\text{SalePrice}] - \text{RELATED}(\text{Stock}[\text{Direct Costs}])$$
10. Click the check icon in the formula bar or press Enter to complete the definition of the calculated column.

You can now see a new column added to the right of the InvoiceLines table. This column contains the gross margin for every vehicle sold, even if the sale price is on one table and the sum of the costs is in a separate table. This is all thanks to the RELATED() function, which links fields from different tables using the joins that you defined in the data model.

If you are an Excel user who has spent hours, or even days, wrestling with the Excel LOOKUP() function, then you are probably feeling an immense sense of relief. For it really is this easy to lookup values in another table in Power BI Desktop. Once again (and at risk of laboring the point), if you have a coherent data model, then you are building the foundations for simple and efficient data analysis further down the line using DAX.

Choosing the Correct Table for Linked Calculations

The nature and structure of a Power BI Desktop data model controls where you can add new columns from another table. In essence, you can only bring data into a table if it is from another table that

- Contains reference data
- Contains many records that are “sub” elements of the current table

So tables such as Countries, Clients, and Colors cannot pull back data from another table using the RELATED() function. This is because they are lookup tables and contain reference data that appears only once, but is used many times in other tables. In database terms, these tables are the “one” side of a relationship; whereas tables such as InvoiceLines are on the “many” side of a relationship. In Power BI Desktop (as is the case in a relational database), you can only look up data from the “many” side.

Equally, you can add data to the InvoiceLines table from the Invoices table, because the former is considered a “parent” to the latter. However, you cannot pull data into the Invoices table from the InvoiceLines table. Quite simply, when an invoice contains many lines, Power BI Desktop does not know which row to select and return to the destination table.

In a data model like the sample Brilliant British Cars example, some tables can return data from several tables. For instance, the Stock table can reach into the InvoiceLines table (because there is a single record in the Stock table for each record in the InvoiceLines table). Since the InvoiceLines table is a child of the Invoices table, the Stock table can reach “through” the InvoiceLines table into the Invoices table. Indeed, as the Colors table is a lookup table for the Stock table, and the Invoices table looks up data from the Clients table, these are also accessible to the Stock table.

So essentially, the table where you add a new column has the potential to reach through most, if not all, of the data model and return data from many other tables—providing that the data model has been constructed in a coherent manner, of course.

Cascading Column Calculations

New columns can refer to previously created new columns. This apparently anodyne phrase hides one of the most powerful features of Power BI Desktop: the ability to create spreadsheet-like links between columns where a change in one column ripples through the whole data model.

This implies that you help yourself if you build the columns in a logical sequence, so that you always proceed step by step and do not find yourself trying to create a calculation that requires a column that you have not created yet. Another really helpful aspect of new columns is that if you rename a column, Power BI Desktop automatically updates all formulas that used the previous column name, and uses the new name in any visualizations that you have already created. This makes Power BI Desktop a truly pliant and forgiving tool to work with.

So if we take the DAX formulas that you have created so far, you have the Gross Margin column that depends on the data for the sale price and the calculation of the Direct Costs column, which itself is based on the data for the cost price, spare parts, and labor cost. As a spreadsheet user, you probably won't be surprised to see that any change to the source data for the four elements causes both the Direct Costs and Gross Margin columns to be recalculated.

Refreshing Data

Do the following to force Power BI Desktop to recalculate the data model:

1. Activate the Home ribbon in Data View (or in Dashboard View).
2. Click the Refresh button. The Refresh dialog will appear, looking something like Figure 7-4.

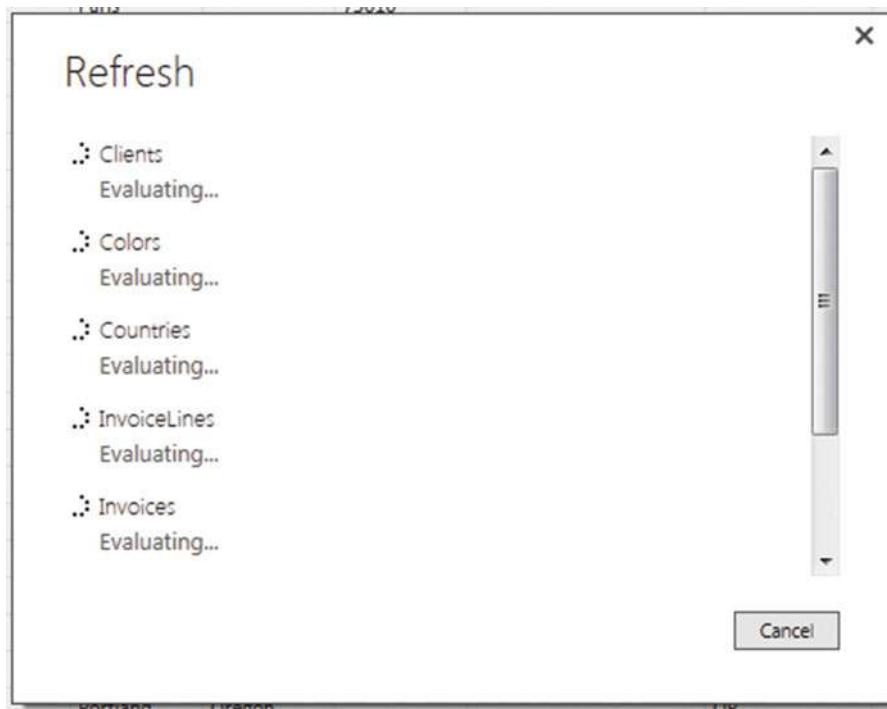


Figure 7-4. The Refresh dialog

After a short while (depending on the amount of data that has to be reloaded from the source(s) into the data model), the dialog closes and the data reappears with all calculated columns updated.

Using Functions in New Columns

You have seen just how easy it is to extend a data model with some essential metrics in Power BI Desktop. Yet we have only performed simple arithmetic to achieve our ends. Power BI Desktop can do much more than just carry out simple sums, of course.

Safe Division

I imagine that if you are an Excel user, you have seen your fair share of DIV/0 (divide by zero) errors in spreadsheets. Fortunately, the Power BI Desktop team shares your antipathy to this particular issue and they have endowed Power BI Desktop with a particularly elegant solution to the problem. This solution also serves as a simple introduction to the world of DAX functions in Power BI Desktop.

Suppose that you want to add a new column that divides the contents of one column by the contents of another. Still using the PowerBiDataModel.pbix sample file, you can implement safe division like this:

1. With the InvoiceLines table selected, activate the Modeling ribbon and click the New Column button.
2. To the right of the equals sign, type **DIVIDE** followed by a left parenthesis.
3. Type the formula **RELATED** and a left parenthesis.
4. Select the Stock[DirectCosts] field from the list. This will be the numerator (the value that will be divided by another value).
5. Add a right parenthesis (this is to finish the RELATED() function).
6. Enter a comma.
7. Enter a left square bracket: [.
8. Select the SalePrice field from the list. This will be the denominator (the value that divides the first value).
9. Enter a comma.
10. Enter a **0**. This is the figure that will appear if there is a division by zero error.
11. Add a right parenthesis to end the DIVIDE() function.
12. To the left of the equals sign, replace Column with **SalePriceToSalesCostsRatio**. The formula bar will look like Figure 7-5.



Figure 7-5. The DIVIDE() function

13. Click the tick icon in the formula bar (or press the Enter key). The ratio of sales cost to the sale price will be calculated for every row in the table. The column will fill with zeros and will remain highlighted.
14. In the Modeling ribbon, click the percent button, and then add a couple of decimals by using the decimals button. As this metric is a ratio, it is best presented as a percentage to be more easily comprehensible, not only in the current table but also in any visualizations that it appears in.

In this example, you have used a function at required multiple parameters.

- The *numerator*: The number that is divided by another number.
- The *denominator*: The number that is used to divide the first value.
- The *error value*: The number that is used if a divide by zero error is encountered.

I imagine that by now that you are feeling that DAX formulas are not only relatively easy, but also made easier by their close relationship to Excel formulas. So let's move on to see a few more.

Counting Reference Elements

Data models are often assembled to make the best use of reference elements. The Brilliant British Cars data model has a couple of lookup tables (Clients and Colors) that contain essential information that you could need to analyze the underlying data. So as an example of how the data model can be put to good use, let's look at how DAX can calculate the number of clients per country.

This challenge introduces two new DAX elements:

- The COUNTROWS() function
- The RELATEDTABLE() function

As its name implies, the COUNTROWS() function counts a number of rows in a table. While you can use it simply to return the number of records in the current table, it is particularly useful when used with the RELATEDTABLE() function. Once again, because the data model has been set up coherently, and the Countries table is joined to the Clients table, using the COUNTROWS() and RELATEDTABLE() functions together does not just return the number of records in a table, but also calculates the number of records for each element in the table where it is applied. This means that any elements from the Countries table that exist in the Clients table can be identified because the Clients table is using the Countries table as a lookup table.

Here is an example of how you can use these two functions to count reference elements:

1. In Data View, select the Countries table.
2. Click New Column in the Modeling ribbon.
3. In the formula bar, replace the word Column with the words **Clients Per Country**.
4. To the right of the equals sign, type **COUNTROWS(**. Once the keyword appears in the pop-up list, you can select it to save time (and keystrokes), if you want.
5. Enter (and/or select) **RELATEDTABLE(**.
6. The list of the tables that are related to the Countries table will appear in the formula bar.
7. Select Clients.
8. Add *two* right parentheses. One will close the RELATEDTABLE() function and the other will end the COUNTROWS() function.
9. The formula bar will contain the following:

$$\text{Clients Per Country} = \text{COUNTROWS}(\text{RELATEDTABLE}(\text{Clients}))$$
10. Click the tick icon in the formula bar (or press the Enter key). The number of customers for each country will appear in the new column named Clients Per Country. The Countries table now looks like what's shown in Figure 7-6.

CountryID	CountryName	CountryISOCode	Clients Per Country
1	United Kingdom	GBR	11
2	France	FRA	4
3	USA	USA	11
4	Germany	DEU	1
5	Spain	ESP	1
6	Switzerland	CHE	3

Figure 7-6. Using the COUNTROWS() function to calculate the number of clients per country

Statistical Functions

As an intrinsic part of the Microsoft BI offering, DAX can calculate aggregates. After all, analyzing totals, averages, minima, and maxima (among others) is a core aspect of much business intelligence.

However, I do not want just to show you how to create a column containing the average sale price of all vehicles in the dataset. This would hardly be instructive. So instead of this, let's begin with a slightly more interesting requirement. Suppose that, for each vehicle, you want to see how the net profit compares to the average net profit.

1. Select the InvoiceLines table in the Fields list.
2. In the Modeling ribbon, click the New Column button.
3. In the formula bar to the left of the equals sign, replace Column with **DeltaToAvgNetProfit**.
4. Click to the right of the equals sign.
5. Enter a left square bracket.
6. Select the Gross Margin field.
7. Enter a minus sign.
8. Start typing the word **Average**. Power BI Desktop will display the list of available fields. When enough of the function name Average appears in the list of functions, select the AVERAGE function.
9. Select the field Gross Margin.
10. Add a right parenthesis. The formula will look like this:

```
DeltaToAvgNetProfit = [Gross Margin]-AVERAGE([Gross Margin])
```

11. Confirm the formula by pressing Enter or clicking the tick icon in the formula bar. The new column will display the difference between the net margin for each row and the average net margin.

Once again, if you are an Excel or Microsoft Access user, you are probably feeling quite at ease with this way of working. Even if you are not a spreadsheet or database expert, you must surely be feeling reassured that creating calculations that apply instantly to an entire column is truly easy.

Now that you have seen the basic principles, look at some of the more common available aggregation functions described in Table 7-4.

Table 7-4. DAX Statistical Functions

Function	Description	Example
AVERAGE()	Calculates the average (the arithmetic mean) of the values in a column. Any non-numeric values are ignored.	AVERAGE([Mileage])
AVERAGEA()	Calculates the average (the arithmetic mean) of the values in a column. Empty text, non-numeric values, and FALSE values count as 0. TRUE values count as 1.	AVERAGEA([Mileage])
COUNT()	Counts the number of cells in a column that contain numeric values.	COUNT([Mileage])
COUNTA()	Counts the number of cells in a column that contain any values.	COUNTA([Mileage])
COUNTBLANK()	Counts the number of blank cells in a column.	COUNTBLANK([Mileage])
COUNTROWS()	Counts the number of rows in a table.	COUNTROWS(Stock)
DISTINCTCOUNT()	Counts the number of unique values in a table.	DISTINCTCOUNT([Vehicle])
MAX()	Returns the largest numeric value in a column.	MAX([Mileage])
MAXA()	Returns the largest value in a column. Dates and logical values are also included.	MAXA([Mileage])
MEDIAN()	Returns the median numeric value in a column.	MEDIAN([Mileage])
MIN()	Returns the smallest numeric value in a column.	MIN([Mileage])
MINA()	Returns the smallest numeric value in a column. Dates and logical values are also included.	MINA([Mileage])

There are many more statistical functions in DAX, and you can take a deeper look at them in the Power BI online documentation. However, for the moment the intention is not to blind you with science, but to introduce you more gently to the amazing power of DAX. For the moment, then, rest reassured that all your favorite Excel functions are present when it comes to calculating aggregate values in Power BI Desktop.

Applying a Specific Format to a Calculation

Sometimes you will want to display a number in a particular way. You may need to do this to fit more information along the axis of a chart for instance. In cases like these, you can, in effect, duplicate a column and reformat the data so that you can use it for specific visualizations. As an example of this, and to show how functions can be added to formulas that contain math, you will convert the cost of vehicles and any spare parts from pounds sterling to US dollars and then format the result in dollars.

1. In the Power BI Desktop window, make sure that you are in Data View.
2. Click the Invoices table.
3. In the Modeling ribbon, click the New Column button. A new empty column will appear to the right of the final column of data. This column is currently highlighted and titled Column.
4. The formula bar above the table of data will display Column = .
5. In the formula bar, click to the right of the equals sign.
6. Type **FORMAT(**. Once the function appears in the pop-up menu, you can select it if you prefer.
7. Click inside the Cost Plus Spares column. Clients[Cost Plus Spares] will appear in the formula bar. Alternatively, you can type a left square bracket and select the [Cost Plus Spares] field.
8. Enter *** 1.6**.
9. Enter a comma.
10. Enter the text “**Fixed**” (include the double quotes).
11. Add a right parenthesis.
12. Still inside the formula bar, replace Column with **Invoice In Dollars**.
13. To the right of the equals sign, enter “\$” & (include the double quotes). The formula bar will read as follows:
`Invoice In Dollars = "$ " & FORMAT([Cost Plus Spares] * 1.6, "Fixed")`
14. Press Enter (or click the tick icon in the formula bar). The column will contain the same number as the one in the Cost Plus Spares column. However, it is formatted as a text and preceded by a dollar sign. Figure 7-7 shows a few records from this new column.

InvoiceInDollars
\$ 71200.00
\$ 71040.00
\$ 19600.00
\$ 1464.00
\$ 44000.00
\$ -42400.00
\$ 84960.00
\$ 160960.00
\$ 181760.00
\$ -52624.00

Figure 7-7. Applying a custom numeric format

The `FORMAT()` function can be applied equally well to dates and times as to numeric columns, as you will see later in this chapter. Indeed, it offers a wealth of possibilities. So many in fact, that rather than illustrate all of them, Tables 7-5 and 7-6 contain the essential predefined formats that you can apply to columns of numbers.

Table 7-5. Predefined Currency Formats

Format Code	Description	Example	Comments
Currency	Currency	FORMAT(Stock[CostPrice], "Currency")	The currency indicator will depend on the PC's settings and language used.
Scientific	Exponential or scientific notation	FORMAT(Stock[CostPrice], "Scientific")	This is also called the <i>scientific format</i> .
Fixed	Fixed number of decimals	FORMAT(Stock[CostPrice], "Fixed")	Displays at least one figure to the left of the decimal (even if it is a zero) and two decimals.
General Number	No format	FORMAT(Stock[CostPrice], "General Number")	Displays the number with no thousand separators.
Percent	Percentage (and divided by 100)	FORMAT(Stock[SalePriceToSalesCostsRatio], "Percent")	Displays the number as a percentage.

Table 7-6. Custom Number Formats

Format Code	Description	Comments
0	The zero placeholder	Adds a zero even if no number is present
#	The digit placeholder	Represents a number if one is present
.	The decimal character	Sets the character that is used before the decimals
,	The thousands separator	Defines the thousands separator
%	Percentage symbol	Adds a percentage symbol

Should you wish to create your own highly specific date and number formats, you can assemble them using the format code elements found in Tables 7-5 and 7-6.

Using the custom number formats is not difficult; but rather than explain all the permutations laboriously, here are a couple of examples to help you to see how they work.

- `FORMAT([Cost Plus Spares], "#,#.00")` gives you 44,500.00 (and any figure less than 1 will have a nothing to the left of the decimal)
- `FORMAT([Cost Plus Spares], "0.0")` gives you 12250.0 (and any figure less than 1 will have a 0 to the left of the decimal)

Remember that if you want to abandon a formula while you are creating it, all you have to do is click the cross icon in the formula bar or press Escape.

Note The `FORMAT()` function actually converts a number to text. Consequently, you might not be able to use a calculated column that is the result of a `FORMAT()` operation in further calculations.

Simple Logic: the IF() Function

Having data available is always a prerequisite for analysis; however, the raw data may not always lend itself to being used in dashboard visualizations in an ideal way.

DAX can help you to see “the wood for the trees” in the thicket of data that underlies your data model. Let’s begin by looking at a series of practical examples that extend your data in ways that use the resources of Power BI Desktop to do the heavy lifting and let you focus on items that need your attention.

Exception Indicators

As a first example of how to use the IF() function, suppose that you want to highlight any records where the cost of spare parts is over £2,000.00. This means comparing the contents of the column PartsCost to a fixed value (3500). If this test turns out to be true (that is, the parts cost is over the threshold that you have set), then you want to display the words *Too High!*

The following explains how to add a column that applies this test to the data.

1. In the PowerBiDataModel.pbix file, click the Data View icon and then click the Stock table in the Fields list.
2. Click the New Column button in the Modeling ribbon. A new column named Column will appear at the right of any existing columns.
3. To the right of the equals sign, enter **IF(**. You will see that as you enter the first few characters the list of functions will list all available functions beginning with these characters.
4. Press the [key. The list of available fields will appear.
5. Scroll down through the list of fields and click the SpareParts field.
6. Enter the greater than symbol: >.
7. Enter **2000**.
8. Enter a comma.
9. Enter the following text (including the double quotes): **“Too Much!”**.
10. Enter a closing parenthesis: **)**. The code in the formula bar will look like this:

```
Column = IF([SpareParts]>2000,"Too Much!")
```

11. Press Enter or click the tick icon in the formula bar. The new column will display Too Much in any rows where the cost of spares is over £2,000.00.
12. Rename the column **Excessive Parts Cost**.

Like the DIVIDE() function, the IF() function can take up to three arguments (as the separate elements that you enter between the parentheses are called). The first two are compulsory:

- A *test* in this case comparing the contents of a column to a fixed value.
- The outcome if the test is *positive* (or TRUE in programming terms).

The IF() function can also have a third argument, although this is optional, as you can see in the example in this section.

- The outcome if the test is *negative* (or FALSE in programming terms)

This was a simple test to help you to isolate certain records. Later, when building dashboards, you can use the contents of this new column as the basis for tables, charts, and indeed just about any Power BI Desktop visualization.

Creating Alerts

When using the IF() function, the major focus is nearly always on the first argument—the test. After all, this is where you can apply the real force of Power BI Desktop. So here is another example of an IF() function being used, only this time it is to create an alert based on a slightly more complex calculation. This time, the objective is to detect records where the selling price of the vehicle is less than half the average sale price for all cars.

1. In the PowerBiDataModel.pbix file, click the Data View icon, and then click the Stock table in the Fields list.
2. Click the New Column button in the Modeling ribbon. A new column named Column will appear at the right of any existing columns.
3. To the left of the equals sign, replace the word Column with **Price Check**.
4. To the right of the equals sign, enter **IF**. You will see that as you enter the first few characters the list of functions will list all available functions beginning with these characters.
5. Press the [key. The list of available fields will appear.
6. Scroll down through the list of fields and click the CostPrice field.
7. Enter >= (it represents “greater than or equals to”).
8. Start typing the word **Average**. Power BI Desktop will display the list of available fields. When enough of the function name Average appears in the list of functions, select the AVERAGE() function.
9. Enter a left square bracket.
10. Start typing the name of the SalePrice field.
11. When the [SalePrice] field is visible in the pop-up list of fields, select it.
12. Add a right parenthesis. This ends the AVERAGE() function.
13. Enter *2.
14. Enter a comma.
15. Enter the following text (including the double quotes before and after the text): **“Price too High”**.
16. Enter a comma.
17. Enter **“Price OK”** (including a pair of double quotes).

18. Enter a closing parenthesis. This ends the IF() function. The code in the formula bar will look like this:

```
PriceCheck = IF([CostPrice] >= AVERAGE([CostPrice]) *2,"Price too high",
"Price OK")
```

19. Press Enter or click the tick icon in the formula bar. The new column will display Price too High or OK, depending on whether the cost price is more than half the average cost price or not.

You can then use the results of the cost price test as the basis for a visualization to compare the expensive purchases with the others.

Comparison Operators

When carrying out tests like this, you need to compare values. You may be familiar with the standard comparison operators that many programs and languages use (such as Excel), but for the sake of completeness, Table 7-7 provides a list of the most frequently used operators.

Table 7-7. DAX Comparison Operators

Operator	Description
=	Equals (exactly!)
<>	Not equals to
<	Less than
>	Greater than
<=	Less than or equals to
>=	Greater than or equals to

Flagging Data

As a practical example of how you might use an IF() function to validate data, imagine that Brilliant British Cars is embarking on a “know your customer” program and you envisage a chart that compares the clients that have reliable postcodes with those that do not. This way you can make a business case for cleansing the data and potentially rooting out certain clients.

For the moment, the Clients table either has or does not have a postcode (Zip code) for each customer. What you want is a clear extra column that contains either HasPostCode or NoPostCode to indicate whether there is a postcode present.

In this example, you will not only test numeric values, but you will also look at whether a record contains a value for a row. This means introducing a new DAX function. This is the ISBLANK() function. It allows you to see if a column contains any data or not. Technically, this DAX function returns TRUE if the column is empty, and FALSE if it contains data. So you can nest it inside an IF() function to detect the presence of data, rather than looking at the data itself.

The following explains how to create a clear indicator of the presence or absence of a postcode.

1. In the PowerBiDataModel.pbix file, click the Clients table in the Fields list.
2. Click the New Column button in the Modeling ribbon.
3. To the left of the equals sign, replace Column with IsPostCode.

4. To the right of the equals sign, enter IF(. You will see that as you enter the first few characters, the list of functions lists all available functions beginning with these characters.
 5. Type **IsB**. The list of functions will show ISBLANK().
 6. Click ISBLANK() or press the Tab key to select this function. Power BI Desktop will place the function in the formula bar and add the left parenthesis automatically.
 7. Press the [key. The list of available fields will appear.
 8. Select [PostCode].
 9. Enter a right parenthesis. (This finishes the ISBLANK() function.)
 10. Enter a comma and then type “**NoPostCode**”, “**HasPostCode**”. These are the outputs that the IF() function will return, depending on whether the column is blank or not.
 11. Enter a final right parenthesis. The formula bar will display this:
- ```
IsPostCode = IF(ISBLANK([PostCode]), "NoPostCode", "HasPostCode")
```
12. Press Enter or click the tick icon in the formula bar. The new column will display either NoPostCode or HasPostCode for every Client.

You can now use this new field to filter data in dashboards or in tables and charts to separate the clients that do or do not have postcodes.

## Nested IF() Functions

A frequent requirement in data analysis is to categorize records by ranges of values. Suppose, for instance that you want to break down the stock of cars into low-, medium-, and high-mileage models. This requires more than a simple IF() function. However, it is not very difficult, as all that is needed is to “nest” one IF() function inside another, thereby extending the test that is applied to cover three possible outcomes.

Here, then, is how to create a simple nested IF() function.

1. In the PowerBiDataModel.pbix file, click the Stock table in the Fields list.
2. Click the New Column button in the Modeling ribbon.
3. To the left of the equals sign, replace Column with **Mileage Range**.
4. To the right of the equals sign, enter **IF**. You see that as you enter the first few characters, the list of functions lists all available functions beginning with these characters.
5. Press the [ key. The list of available fields will appear.
6. Select [Mileage]. You can type it fully if you prefer, but remember to add the right square bracket if you do.
7. Enter <= **50000**.
8. Enter a comma.
9. Enter “**Low**” (including the double quotes).

10. Enter a comma.
11. Enter **IF(**.
12. Enter **< 100000**.
13. Enter a comma.
14. Enter “**Medium**”, “**High**”. This must include the double quotes and the comma separating the two words.
15. Enter two right parentheses, one for each of the IF() functions.  
The formula bar will display

Mileage Range = IF([Mileage] <= 50000, "Low", IF([Mileage] < 100000, "Medium", "High"))

16. Press Enter or click the tick icon in the formula bar. The new column will display Low, Medium, or High for every vehicle in the new Mileage Range column.

You have now categorized all the cars in stock by their mileage, and can use the category flag in the Mileage Range column to create, for instance, a chart that shows the number of vehicles corresponding to each mileage category.

A nested IF() function works like this:

1. You set up a first test. In this example, the test is to flag all cars that have less than 50,000 miles “on the clock.”
2. You specify what the outcome is if this initial test is positive. In this example, the word *Low* appears in the column.
3. You then add a second test. By definition, this will *only* apply to cars that have travelled more than 50,000 miles; otherwise, the formula returns the word *Low*. So you add a higher threshold for the second test, which is 10,000 miles in this example.
4. If the record passes the test and the vehicle has traveled less than 100,000 miles, then the word *Medium* will appear in the column. In all other cases (that is for all mileage over 100000 miles), the word *High* is displayed in the column.

When writing nested IF() statements, the essential trick is to use a sequence of tests that follow a logical order, from lowest to highest (or in some cases, from highest to lowest). This way, the succession of IF() statements acts like a series of hoops that catch the values and return an appropriate result.

You can nest up to 64 IF() statements in a single DAX expression. In fact, you can nest a maximum of 64 DAX expressions, whatever they are. However, more than half a dozen can be painful to write correctly, and getting the correct number of right parentheses in place can be tricky. However, there may be many occasions when you need to segment your data for your visualizations and dashboards, even if it means grappling with complex nested IF() statements. So let's take a look at one of these to whet your appetite.

## Creating Custom Groups Using Multiple Nested IF() Statements

To give you another example of a slightly more complex DAX function (but one that can be very necessary), consider the following requirement. Our data now has the car age, but we want to group the cars by age segments (or buckets if you prefer). So we will use a nested IF() function to do this. Then, to allow us to sort the column in a more coherent way, we will create a Sort By column for the new Vehicle Age Category column that we created. If you remember, we saw how to create and use Sort By columns in the previous chapter.

In this example, I will not explain every step, as you have seen how to select functions and fields from pop-ups in the previous examples. Instead, I prefer to concentrate on the logic itself and explain how complex IF() statements can be built.

The code for the Vehicle Age Category column is as follows:

```
Vehicle Age Category=IF(
 [VehicleAgeInYears] <=5,"Under 5",
 IF(AND([VehicleAgeInYears]>=6,[VehicleAgeInYears]<=10),"7-10",
 IF(AND([VehicleAgeInYears]>= 11,[VehicleAgeInYears]<=15),"7-15",
 IF(AND([VehicleAgeInYears]>=16,[VehicleAgeInYears]<=20),"17-20",
 IF(AND([VehicleAgeInYears]>=21,[VehicleAgeInYears]<=25),"21-25",
 IF(AND([VehicleAgeInYears]>=26,[VehicleAgeInYears]<=30),"27-30",
 ">30"
)
)
)
)
)
)
```

The only slight problem with a great technique for segmenting data is that if you sort the Vehicle Age Category column, you will find that the category that corresponds to the highest age appears at the top of the list. So you need to add a second column that can be used as a sort order column for the new column that you just created. The following is the code for this Vehicle Age Category Sort column.

```
Vehicle Age Category Sort=IF([VehicleAgeInYears]<=5, "1",
 IF(AND([VehicleAgeInYears]>=6, [VehicleAgeInYears]<=10),"2",
 IF(AND([VehicleAgeInYears]>= 11, [VehicleAgeInYears]<=15), "3",
 IF(AND([VehicleAgeInYears]>=16, [VehicleAgeInYears]<=20), "4" ,
 IF(AND([VehicleAgeInYears]>=21 , [VehicleAgeInYears]<=25), "5",
 IF(AND([VehicleAgeInYears]>=26, [VehicleAgeInYears]<=30),"6","7"
)
)
)
)
)
```

These formulas could have come straight from an Excel spreadsheet. Indeed, some 80 of the DAX functions are nearly identical to their Excel cousins. So experience and imagination combined have shown me that you have many ways to extend the data you imported by adding calculated columns. Even better, all calculated columns are updated when you refresh the data from the source. The only major caveat is that when you are tweaking the data connection, you must be careful *not* to delete any source columns on which a calculated column depends, or else you will get errors in the Power BI Desktop table.

---

**Tip** You could have created the formula in this example without creating the VehicleAge column first, as you could have used the formula that calculates the age of the car each time that you need the vehicle age. However, as you can imagine, it is easier to create a column that contains the vehicle age first and then refer to this in the Vehicle Age Category formula. This makes the more complex formula easier to read. It also helps you to break down the analytical requirement into successive steps, which is good DAX development practice. Moreover, you can always hide any “intermediate” columns if you do not need them in dashboards and reports. Indeed, you could even do this kind of calculation in Power BI Desktop Query.

---

## Multiline Formulas

By default, all formulas that you create in Power BI Desktop will be on a single line that overflows onto the next line when there is no more room in the formula bar. This can become an extremely tedious way of working, so it is worth knowing that you can tweak long formulas to force them to display over more than one line. All you have to do is force a line return inside the formula bar by pressing Shift+Enter where you want to force a new line. My experience is that Power BI Desktop will not let you create line breaks everywhere in a formula. Nonetheless, with a bit of trial and error, a more complicated formula, such as the CarAgeBucket column that you created, can look like the multiline formula in Figure 7-8.

Just in case you were wondering, you do *not* have to write formulas over multiple lines as I did just. Indeed, the two formulas used to create complex nested IF() statements could be written as follows:

```
Vehicle Age Category=IF([VehicleAgeInYears] <=5,"Under 5",
IF(AND([VehicleAgeInYears]>=6,[VehicleAgeInYears]<=10),"7-10",
IF(AND([VehicleAgeInYears]>= 11,[VehicleAgeInYears]<=15),"11-15",
IF(AND([VehicleAgeInYears]>=16,[VehicleAgeInYears]<=20),"17-20",
IF(AND([VehicleAgeInYears]>=21,[VehicleAgeInYears]<=25),"21-25",
IF(AND([VehicleAgeInYears]>=26,[VehicleAgeInYears]<=30),"27-30","Over 30")))))
```

And

```
Vehicle Age Category Sort=IF([VehicleAgeInYears] <=5,"1",
IF(AND([VehicleAgeInYears]>=6,[VehicleAgeInYears]<=10),"2",
IF(AND([VehicleAgeInYears]>= 11,[VehicleAgeInYears]<=15),"3",
IF(AND([VehicleAgeInYears]>=16,[VehicleAgeInYears]<=20),"4",
IF(AND([VehicleAgeInYears]>=21,[VehicleAgeInYears]<=25),"5",
IF(AND([VehicleAgeInYears]>=26,[VehicleAgeInYears]<=30),"6","7")))))
```

I chose to write the formulas over multiple lines, hoping that by doing so, I'd make the nested logic clearer. You can write your formulas in any way that suits you and that does not cause Power BI Desktop a problem.

## Complex Logic

Categorizing data can sometimes involve applying logic that is more complex than a single simple comparison. You could need to apply two or more conditions when evaluating a record, and this more intricate logic could require you to test the contents of more than one column.

Once again, DAX can help you in circumstances like these. To explain by example, consider the following analytical challenge. You want to flag any vehicle that is a red or blue coupe. This example will show the basics of applying complex logic to data analysis with DAX.

1. In the PowerBiDataModel.pbix file, click the Stock table in the Fields list.
2. Click the New Column button in the Modeling ribbon.
3. To the left of the equals sign, replace Column with **Special Sales**.
4. To the right of the equals sign, enter **IF()**. Since we are dealing with multiple parentheses in this formula, I prefer to enter both the opening and the closing parentheses for each function when adding the function.
5. Click inside the parentheses.
6. Type the **AND()** function and then click inside the parentheses. This function ensures that multiple logical conditions are applied and that all must be satisfied for the test to be successful.
7. Type **[VehicleType] = "Coupe"**, (including the comma). This is one of the conditions that has to be true for the test to be successful.
8. After the comma, type **OR()** and then click inside the parentheses. This is a second condition (it is still part of the **AND()** function), but it can be one of many different tests.
9. Type (or use the pop-up menus to select as well as partially typing) **RELATED(Colors[Color]) = "Red", RELATED(Colors[Color]) = "Blue"**. This is the second part of the test. However, because the color is in another table, you have to use the **RELATED()** function to find the color of the vehicle because it is not in the Stock table. Also there are two alternative conditions inside the parentheses for the **OR()** expression.
10. Click just inside the final parenthesis at the right of the DAX expression (this is the one that ends the **IF()** function) and type **"Special", "Normal"**. Be sure to include the commas. The formula should read:

```
Special Sales = IF(AND([VehicleType] = "Coupe", OR(RELATED(Colors[Color]) = "Red", RELATED(Colors[Color]) = "Blue")), "Special", "Normal")
```

11. Press Enter or click the tick icon in the formula bar. The new column will display either Special or Normal for every vehicle in the new Special Sales column

I realize that a formula like this can seem daunting at first sight. So let's take another look at this DAX expression formatted a little differently.

```
Special Sales = IF(
 AND(
 [VehicleType] = "Coupe",
 OR(RELATED(Colors[Color]) = "Red",
 RELATED(Colors[Color]) = "Blue")
)
 , "Special"
 , "Normal"
)
```

As you can see, at its heart, the expression is an IF() expression. As such, it consists of three parts:

- A *test* (The car is a coupe that is either red or blue)
- An *outcome for a positive result* (displays “Special”)
- An *outcome for a negative result* (displays “Normal”)

The only tricky bit now is the test itself. Since it is built on logic that is more complex, it requires a little explanation.

- First, you have stated that the test is in several parts, all of which must be true for a record to pass the test. This is done by using the AND() function and then separating each individual test (of which there are two in this example—the vehicle type and the color).
- Second, you have told DAX that the second test (on the color) can be any of several possibilities (two in this example). You did this using the OR() function and separating each individual test by a comma.

Although I prefer to build complex functions from the inside out (that is, by adding all the required parentheses first and adding what goes inside them second), this is not an obligation. You are free to build DAX formulas in any way that works.

**Note** This example only showed two alternatives for the AND() and OR() functions. This is because these functions are limited to only two parameters. What is important to remember is that you will have *to repeat the field* (and possibly the table name if it is not the current table) for *each comparison*, just as you did here when testing the colors of the cars. If you need more than two alternatives for an AND or an OR operation then you will have to use the logical *operators* (&&, ||, and !) that are described below.

Armed with this knowledge, you can now build extremely complex logical tests on your data. If you are an Excel or Access power user, then the learning curve should be quite short as the principles and functions are similar to those that you are using already. If you have come from the world of programming, then the concepts are probably familiar. If you are just starting out, then just be prepared to spend a little time practicing, and above all, analyze the question that you want DAX to answer *before* starting to write the statement.

## DAX Logical and Information Functions

So far in this chapter, you have seen three of the DAX logical functions. In practice, you may need to build formulas that use some of the other functions that DAX provides to apply logic and to test the state and type of information in columns. Table 7-8 describes the essential functions for creating complex data models.

**Table 7-8.** DAX Logical Functions

| Operator    | Description                                                                                             | Example                                                                               |
|-------------|---------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| IF()        | Tests a condition and applies a result if the test is true, and possibly a result if the test is false. | IF([PartsCost]> 500,<br>"Check Parts", "OK")                                          |
| AND()       | Extends the logic to include several conditions <i>all</i> of which must be met.                        | IF(AND([PartsCost]><br>500,[LaborCost]>1000),<br>"Repair Cost Excessive", "OK")       |
| OR()        | Extends the logic to include several conditions <i>any</i> of which must be met.                        | IF(OR([PartsCost]><br>500,[LaborCost]>1000),<br>"Repair or Labor Cost issue", "OK")   |
| NOT()       | Extends the logic to include several conditions <i>none</i> of which must be met.                       | IF(NOT([PartsCost]><br>500,[LaborCost]>1000),<br>"No Repair or Labor Cost issue", "") |
| ISERROR()   | Tests a value and returns TRUE if there is an error value.                                              | IF(ISERROR([PartsCost]),<br>"Check parts", "")                                        |
| TRUE()      | Returns TRUE.                                                                                           | IF(Stock[Mileage] > 100000,<br>TRUE(), FALSE())                                       |
| FALSE()     | Returns FALSE.                                                                                          | IF(Stock[Mileage] > 100000,<br>TRUE(), FALSE())                                       |
| ISNUMBER()  | Detects if a column value is numeric.                                                                   | IF(ISNUMBER([PartsCost]), "",<br>"Data Error")                                        |
| ISTEXT()    | Detects if a column value is a text.                                                                    | IF(ISTEXT([PartsCost]),<br>"Data Error", "")                                          |
| ISNONTEXT() | Detects if a column value is not a text and is not a blank.                                             | IF(ISNONTEXT([PartsCost]), "",<br>"Data Error")                                       |
| ISODD()     | Detects if a value is an odd number.                                                                    | IF(ISODD([PartsCost]),<br>"Data Error", "")                                           |
| ISEVEN()    | Detects if a value is an even number.                                                                   | IF(ISEVEN([PartsCost]), "",<br>"Data Error")                                          |
| ISLOGICAL() | Detects if a column value is a true or false.                                                           | IF(ISLOGICAL([PartsCost]),<br>"Data Error", "")                                       |

## Logical Operators

If you are writing more complex logical statements when specifying intricate conditions that must be met, then DAX has an alternative to the AND(), OR(), and NOT() functions. These are called *logical operators*, which are explained in Table 7-9.

**Table 7-9.** DAX Logical Operators

| Operator | Description | Example                                    |
|----------|-------------|--------------------------------------------|
| &&       | AND         | [Color] = "Red" && [VehicleType] = "Coupe" |
|          | OR          | [Color] = "Red"    [Color] = "Blue"        |
| !        | NOT         | [Color] = "Red" ! [VehicleType] = "Coupe"  |

As a simple example, you could want a choice of three possible colors in a logical operation. The code for this would read:

```
[Color] = "Red" || [Color] = "Blue" Color] || [Color] = "Green"
```

## Formatting Logical Results

Sometimes a logical function might exist only to return a simple true or false. For instance, you could want to test a value and indicate if it is over a certain threshold, using a formula like the following (added to the Stock table):

```
High Mileage = IF(Stock[Mileage] > 100000, TRUE(), FALSE())
```

This formula simply tests the mileage figure for each record and returns TRUE() if the mileage is greater than 100,000 miles; it returns FALSE() in all other cases.

However, you might not want to display simply TRUE or FALSE in the column. So DAX also lets you format logical output, whether it is calculated like it is here or imported as a TRUE or FALSE value from a data source. The formula that you just saw can be formatted like this:

```
High Mileage = FORMAT(IF(Stock[Mileage] > 100000, TRUE(), FALSE()), "Yes/No")
```

There are only three logical formats available in DAX. These are explained in Table 7-10.

**Table 7-10.** DAX Logical Operators

| Format Code | Description                                |
|-------------|--------------------------------------------|
| Yes/No      | Formats the output as either Yes or No     |
| True/False  | Formats the output as either True or False |
| On/Off      | Formats the output as either Yes or No     |

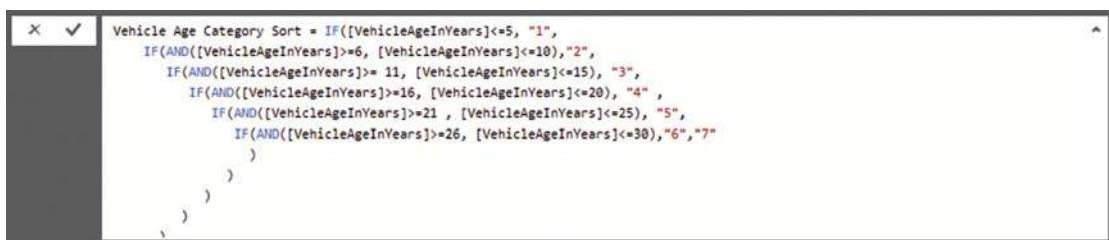
**Note** Different data sources represent True in different ways; however, nearly all represent False as a zero. Consequently, DAX interprets a logical column as a number, any zeros as a False, and anything else as a True.

## Making Good Use of the Formula Bar

If you only ever enter simple formulas, then not only will you be extremely lucky, but you can also content yourself with a single line in the formula bar. I doubt that this is likely to be the case, however, because you will want to do great things with Power BI Desktop. It follows that you may soon be tired of creating long and complex DAX formulas in a limited space. So here is how to expand the formula bar—pretty much as you would in Excel.

1. Click the Expand icon at the right of the formula bar (the downward-facing chevron).

The formula bar increases in height to allow you to type and see several lines of text. To reduce the height of the formula bar and reset it to a single line, just click the Reduce icon at the right of the formula bar (which has now become an upward-facing chevron). You can see this icon in Figure 7-8.



**Figure 7-8.** Multiline formulas

## Conclusion

This chapter introduced you to some of the core techniques that you can apply to extend a Power BI Desktop data model with further metrics. These additional elements were in the form of new columns that you added to many of the data tables that you had previously loaded, cleansed, and assembled into a structured data model.

All the added columns were based on DAX, the Power BI Desktop formula language. As you saw, this language is not especially difficult and it is fairly close to the Excel formula language.

You also learned how to concatenate fields and how to perform basic arithmetic. Then you practiced carrying out calculations that involve multiple tables. Finally, you learned how to segment records using logical functions.

This brief introduction is nonetheless only a quick foretaste of the power of DAX. There is much, much more that can be accomplished to prepare the quantitative analyses that you are likely to need to produce telling visuals with Power BI Desktop. It is time to move on to the next chapter and take a look at the next feature of DAX: creating measures.

## CHAPTER 8



# Adding Measures to the Data Model

Adding new columns can provide much of the extra data that you want to output in tools like Power BI Desktop. It is unlikely, however, that this approach can deliver *all* the analyses that you need. Specifically, calculated columns can *only* work on a row-by-row basis; they cannot contain formulas that have to apply to all or part of the records in a table. For instance, counting the number of cars sold for a year, a quarter, or a month has nothing to do with the data in a single row in the Stock table. It does, however concern the table as a whole.

Generally, you need to add a second type of formula to your tables when you have to look at subsets of the data. These formulas are called, simply, *measures*. These calculations (or measures or metrics—call them what you will) also use DAX. They are applied differently, though, and they can produce some extremely powerful results to help you analyze your data. This is because measures do things that calculated columns simply cannot do. So if you need to work with aggregate values and not on a row-by-row basis then you will have to create measures to achieve the correct result.

As with so many aspects of Power BI Desktop and self-service business intelligence in general, measures are probably best introduced through a few examples. Unfortunately, it is impossible to do anything other than scratch the surface of measures in only a few pages, because they are arguably the most powerful element in Power BI Desktop—one that deserves an entire book to itself. Nonetheless, I hope that this short introduction will whet your appetite, and that you will then continue to learn all about DAX and its more advanced application from the many excellent resources currently available.

In this chapter, we continue to develop the PowerBIDataModel.Pbix file that you began in the previous chapter. The file with all the columns that were added in the previous chapter is available for download as PowerBIDataModelExtended.Pbix.

## A First Measure: Number of Cars Sold

Suppose that you want to be able to display the number of cars sold in Power BI Desktop. Not only that, but you want this figure to adjust when it is filtered or sliced by another criterion, such as country or color. Put simply, you want this metric to be infinitely sensitive to how it is displayed, yet always give the right answer.

So how are we going to achieve this? The following explains how:

1. In Power BI Desktop, ensure that you are in Data View (the middle of the view icons on the left below the ribbon).
2. Select the table to which you wish to add a measure. I chose Stock here.
3. In the Modeling ribbon, click the New Measure button. The formula bar will look like Figure 8-1.

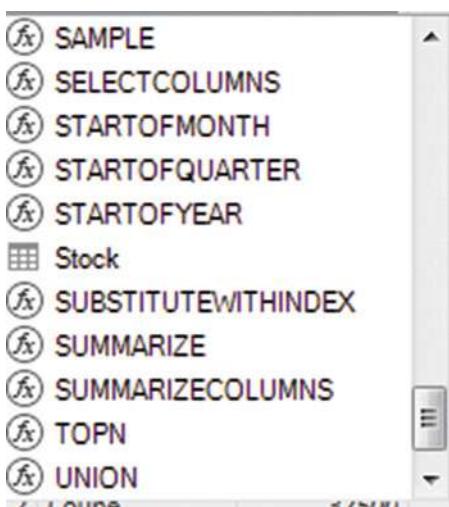


**Figure 8-1.** The formula bar when creating a measure

- Add the following formula to the formula bar:

```
NumberOfCarsSold=COUNTROWS(
```

You will see that the pop-up will then suggest a list of DAX formulas interspersed with the names of tables in the current data model. If you scroll down the list, it will look like Figure 8-2.



**Figure 8-2.** The pop-up menu showing functions and tables

- Select the table Stock and add the right parenthesis. The formula will look like this:

```
NumberOfCarsSold = COUNTROWS(Stock)
```

- Confirm the creation of the formula by pressing Enter or by clicking the check mark icon in the formula bar. A new field will appear in the Fields pane after any existing fields for the Stock table.

Assuming that you have just read Chapter 7, the first thing that will strike you in comparison with creating a new column is that no column is created for this measure. The only indication that it exists is its presence in the Fields list once you expand the Stock table. If you look closely at the field (NumberOfCarsSold) that you have just created, you will also see that there is a tiny icon of a calculator to the left of the field name. This allows you to distinguish measures from New Columns (which have a small Fx icon to the left of the field name).

Not difficult, I am sure you will agree. Yet the best is yet to come. Suppose that you now use this field in a Power BI Desktop card (you saw these briefly in Chapter 1, and can see them on more detail in Chapter 10), which is also filtered to show the results for 2014 only. The result is filtered so that only the number of sales for 2014 is displayed. In other words, the formula is completely separate from the data in a column, but applies any filters that are selected. You can see this applied to a visualization in Figure 8-3.



**Figure 8-3.** Using a measure in a card visualization

The key thing to take away is that a correctly applied measure can be used in a Power BI Desktop table, chart, or indeed any type of visualization, and always shows the correct result of any and all filters and slicers that you have applied. Also, the figures are correct for each intersection of rows and columns in tables. All in all, it is well worth ensuring that you have all the measures that you need for your analytical output in place and that they are working correctly in Power BI Desktop, because you can then rely on these calculations in the dataset in so many different visualizations.

**Tip** You can rename measures by either right-clicking the name of the measure in the Fields list and selecting Rename, or by clicking the name of the measure in the Fields list and altering the name in the formula bar.

When you start out creating measures, it can be a little disconcerting at first not to see the results of a formula immediately, as you can when adding new columns. If this worries you (or if you want to test the result of a new measure), then one approach is to create a table in Dashboard View and add the new measure plus any other useful measures that allow you to verify that everything works as you expected. This technique was outlined briefly in Chapter 1 and is explained in detail in the Chapter 10.

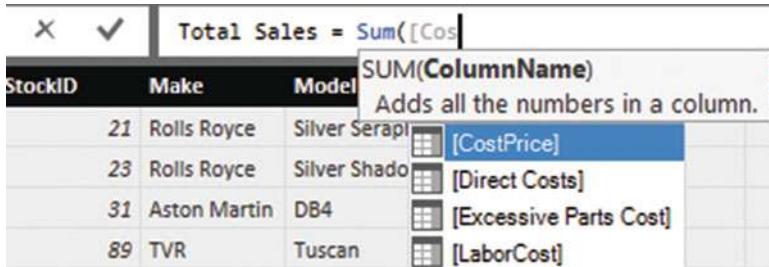
## Basic Aggregations in Measures

Measures are DAX formulas, so in learning to use measures you will have to become familiar with some more DAX functions. My intention here, though, is definitely not to take you through all that DAX can offer. Instead, I would like to show you a few basic formulas that can be useful in real-world dashboards and give you some initial DAX recipes that should prove practical.

So, as a second example, let's calculate total costs of vehicles purchased. Although you can just type in a simple DAX formula, I prefer to show you how you can extend the knowledge that you gained when creating calculated columns and apply many of the same techniques to creating measures.

1. In Power BI Desktop, ensure that you are in Data View.
2. Select the Stock table and click the New Measure button in the Modeling ribbon.
3. Replace Measure with the name that you want to use (**Total Sales**).
4. Click to the right of the equals sign (=).
5. Enter **SUM** as the function, followed by a left parenthesis. A list of all the tables and fields in the data model will appear (including any columns and measures that you have added).
6. Enter a left bracket to restrict the pop-up list to fields in the current table.

7. Start typing the field name (**SalePrice** in this example). After a couple of characters, any tables or fields with these characters will be listed, as shown in Figure 8-4.



**Figure 8-4.** Creating a measure containing an aggregation

8. Scroll down and select the [CostPrice] field.
9. Add a right parenthesis. The formula should read:

TotalSales=SUM(Stock[CostPrice])

10. Press Enter (or click the check mark icon in the formula bar).

The measure is created and it appears in the Fields list. This particular function gives you the total of the SalePrice column. However, when you use it in Power BI Desktop, it is filtered and applied (or sliced and diced if you prefer) to take into account how the data is subset.

One important thing to note when creating measures is that you should use the table name as well as the field name if there are fields that have the same name in several tables. This is so that Power BI is certain that it is using the right field from the right table. What's more, if a table name contains spaces, then the table name needs to be in single quotes. In all cases, the field name has to be enclosed in square brackets.

To practice a little (and to prepare the ground for some eye-catching visualizations in the next few chapters), try creating the average, maximum, and minimum sale price using the formulas in Table 8-1.

**Table 8-1.** A Few Elementary DAX Measures

| Name               | DAX Code                     |
|--------------------|------------------------------|
| Average Cost Price | AVERAGE(Stock[CostPrice])    |
| Maximum Sale Price | MAX(InvoiceLines[SalePrice]) |
| Minimum Sale Price | MIN(InvoiceLines[SalePrice]) |

## Using Multiple Measures

As you can well imagine, not all metrics are likely to be as simple as those that you just saw. You can also create measures that are the result of combining several DAX functions.

For a little practice, you could try adding the ratio of gross margin to sale price to the data model. This measure will then be used in the upcoming chapters on creating dashboards with Power BI Desktop.

1. In Power BI Desktop, ensure that you are in Data View.
2. Select the InvoiceLines table and click the New Measure button in the Modeling ribbon.
3. Replace Measure with the name that you want to use (**RatioNetMargin**).
4. Click to the right of the equals sign (=).
5. Enter **SUM** as the function, followed by a left parenthesis.
6. Enter a left bracket to limit the list to fields in the current table.
7. Select the [Gross Margin] field.
8. Enter a right parenthesis.
9. Enter a forward slash (the divide by operator).
10. Enter **SUM** as the function, followed by a left parenthesis.
11. Enter a left bracket to limit the list to fields in the current table.
12. Select the [SalePrice] field.
13. Enter a right parenthesis. The formula should read:

`RatioNetMargin = SUM([Gross Margin])/SUM([SalePrice])`

14. Press Enter (or click the check mark icon in the formula bar).
15. In the Modeling ribbon, click the percentage button to apply a percentage format.

The easiest way to see the output of a measure like this is to create a table that uses the measure and another attribute so that you can see how the measure works in practice. Figure 8-5 shows a simple example of this, also for 2014.

| Color                | RatioNetMargin▼ |
|----------------------|-----------------|
| Red                  | 47.91%          |
| Blue                 | 46.46%          |
| Dark Purple          | 43.71%          |
| Canary Yellow        | 30.57%          |
| British Racing Green | 25.89%          |
| Silver               | 25.54%          |
| Green                | 19.36%          |
| Black                | 15.37%          |
| Night Blue           | -43.86%         |
| <b>Total</b>         | <b>31.37%</b>   |

**Figure 8-5.** Applying a measure in a table

This is an extremely simple example of a composite DAX function, of course. Indeed, you can probably see a distinct resemblance to an Excel formula. However, the key point to take away is that once you have created the measure it will work in just about any Power BI visualization and using most, if not all, of the attributes from the data model. Power BI can also apply the measure intelligently to hierarchies of data. So, for instance, if you add the Make field to Table 8-1 and switch the visualization to a matrix, you will instantly see the calculations that are shown in Figure 8-6. Here, Power BI Desktop has automatically calculated the net margin ratio for each vehicle sold without you having to alter the formula in any way.

| Color                | Make         | RatioNetMargin |
|----------------------|--------------|----------------|
| Red                  | Bentley      | 66.13%         |
|                      | Aston Martin | 35.82%         |
|                      | Total        | 47.91%         |
|                      |              |                |
| Blue                 | Aston Martin | 61.76%         |
|                      | Jaguar       | 53.93%         |
|                      | Bentley      | 36.38%         |
|                      | Rolls Royce  | -0.67%         |
| Dark Purple          | Total        | 46.46%         |
|                      | Aston Martin | 47.18%         |
|                      | Jaguar       | 39.32%         |
|                      | Total        | 43.71%         |
| Canary Yellow        | Jaguar       | 46.02%         |
|                      | Bentley      | 34.83%         |
|                      | Aston Martin | -17.55%        |
|                      | Total        | 30.57%         |
| British Racing Green | Bentley      | 75.93%         |
|                      | Jaguar       | 16.39%         |
|                      | Rolls Royce  | -19.19%        |
|                      | Total        | 25.89%         |
| Silver               | Bentley      | 40.61%         |
|                      | Rolls Royce  | 38.86%         |
|                      | Aston Martin | 11.66%         |
|                      | Total        | 25.54%         |
| Green                | Jaguar       | 25.57%         |
|                      | Rolls Royce  | 13.28%         |
|                      | Total        | 19.36%         |
|                      |              |                |
| Black                | Bentley      | 75.61%         |
|                      | Jaguar       | 34.76%         |
|                      | Aston Martin | -62.49%        |
|                      | Total        | 15.37%         |
| Night Blue           | Aston Martin | -43.86%        |
|                      | Total        | -43.86%        |
| Total                |              | 31.37%         |

**Figure 8-6.** A hierarchy using a measure

When you use measures like this one in visualizations, you may well find that some calculations are displayed to many decimal places. If you find this distracting, then you can format measures in the same way that you format Power BI Desktop columns. Any formats that you apply are used in Power BI Desktop by default whenever you use this measure.

A final point. When you insert a table or a field from the pop-up list shown in Figure 8-4, you see three types of icons to the left of the table or field: the icon with a table outline denotes a Power BI Desktop table; the table with a selected column icon indicates a column of data or a column that you have added; a calculator icon indicates an existing measure.

## Cross-Table Measures

You are not limited to creating measures that refer to the fields in a single table. If anything, measures are designed to apply across *all* the fields in a data model. To start out with a simple example, suppose that you want to create a custom measure that displays the margin for each vehicle once the cost price and any cost of spares have been deducted.

1. In Power BI Desktop, ensure that you are in Data View (the middle of the view icons on the left below the ribbon).
2. Select the InvoiceLines table and click the New Measure button in the Modeling ribbon.
3. Replace Measure with the name that you want to use (**Cost Plus Spares Margin**).
4. Click to the right of the equals sign (=).
5. Enter **SUM** as the function, followed by a left parenthesis. A list of all the tables and fields in the data model will appear (including any columns and measures that you have added).
6. Select InvoiceLines[SalePrice].
7. Add a minus sign after the formula (you can add spaces before and afterward if you want).
8. Enter **SUM** as the function, followed by a left parenthesis. A list of all the tables and fields in the data model will appear (including any columns and measures that you have added).
9. Select InvoiceLines[CostPrice].
10. Add a minus sign after the formula (you can add spaces before and afterward if you want).
11. Enter **SUM** ( and then expand the Stock table in the Fields list and click the SparesCost field.
12. Add a right parenthesis. The formula will look like this:

```
Cost Plus Spares Margin = SUM(InvoiceLines[SalePrice]) - SUM(Stock[CostPrice]) -
SUM(Stock[SparesCost])
```

13. Press Enter (or click the check mark icon in the formula bar).

You could then add this new measure to the matrix that you saw previously. If you do, you see something like Figure 8-7.

| Color                | Make         | Cost Plus Spares Margin | RatioNetMargin |
|----------------------|--------------|-------------------------|----------------|
| Red                  | Bentley      | 185,920.00              | 66.13%         |
|                      | Aston Martin | 158,300.00              | 35.82%         |
|                      | Total        | 344,220.00              | 47.91%         |
| Blue                 | Aston Martin | 307,000.00              | 61.76%         |
|                      | Jaguar       | 110,360.00              | 53.93%         |
|                      | Bentley      | 17,960.00               | 36.38%         |
|                      | Rolls Royce  | -2,670.00               | -0.67%         |
|                      | Total        | 432,650.00              | 46.46%         |
| Dark Purple          | Aston Martin | 101,700.00              | 47.18%         |
|                      | Jaguar       | 48,960.00               | 39.32%         |
|                      | Total        | 150,660.00              | 43.71%         |
| Canary Yellow        | Jaguar       | 190,160.00              | 46.02%         |
|                      | Bentley      | 242,880.00              | 34.83%         |
|                      | Aston Martin | -48,955.00              | -17.55%        |
|                      | Total        | 384,085.00              | 30.57%         |
| British Racing Green | Bentley      | 158,960.00              | 75.93%         |
|                      | Jaguar       | 9,960.00                | 16.39%         |
|                      | Rolls Royce  | -57,500.00              | -19.19%        |
|                      | Total        | 111,420.00              | 25.89%         |
| Silver               | Bentley      | 28,200.00               | 40.61%         |
|                      | Rolls Royce  | 135,000.00              | 38.86%         |
|                      | Aston Martin | 46,520.00               | 11.66%         |
| Green                | Total        | 209,720.00              | 25.54%         |
|                      | Jaguar       | 23,960.00               | 25.57%         |
|                      | Rolls Royce  | 20,000.00               | 13.28%         |
| Black                | Total        | 43,960.00               | 19.36%         |
|                      | Bentley      | 160,200.00              | 75.61%         |
|                      | Jaguar       | 20,600.00               | 34.76%         |
| Night Blue           | Aston Martin | -125,700.00             | -62.49%        |
|                      | Total        | 55,100.00               | 15.37%         |
|                      | Aston Martin | -67,000.00              | -43.86%        |
| Total                |              | -67,000.00              | -43.86%        |
| Total                |              | 1,664,815.00            | 31.37%         |

**Figure 8-7.** Cross-table measures

Creating cross-table measures in DAX is easier than creating new columns that use values from more than one table. From this example, you can see the following:

- You do not need to use the RELATED() function.
- You only have to specify the table name before entering the field name (or select the combination of table and field from the pop-up).
- You can use the Fields list to refer to fields in other tables (or even in the same table).
- You *must* use aggregation functions on numeric fields. If you do not, you will get an error message.

---

**Note** A measure is attached to a table so that it appears as a field in the specific table. The measure does not have to use any of the fields in the table that “hosts” it. This means that you can attach measures to any table in your data model, which allows for a considerable organizational freedom when extending the model with further metrics. Moreover you can move measures between tables if you want.

---

## More Advanced Aggregations

Now that you have seen how to create basic measures, it is time to move on to some more advanced concepts. More precisely, I want to outline a couple of ways to aggregate data on a row-by-row basis, yet return the result with any filters and slicers applied. There are many cases where this *cannot* be done using a calculated column and then returning the aggregate of the column data. After all, you need to return the *ratio of the sum* of any values and *not the sum of the ratio*. Think of calculating a ratio for each row and then averaging the results to get the average ratio; it is arithmetically false.

Fortunately, Power BI Desktop has some simple yet powerful solutions to this kind of conundrum. One principal tool is the use of the “X” functions—AVERAGEX, COUNTX, SUMX, MAXX, and MINX, among others. These functions allow you to specify

- The table in which the calculations apply
- The row-by-row calculation that is to be applied

As an example, consider the requirement for the ratio of the cost of any parts compared to the purchase price of each vehicle. Not only do we need this potentially at the finest level of granularity—the individual record—but we may need it sliced and diced by any number of criteria. To extend your knowledge, we will also create this measure directly in the Report View, without stepping sideways into the Data View. The following explains how to create the formula that you could use in Power BI Desktop reports:

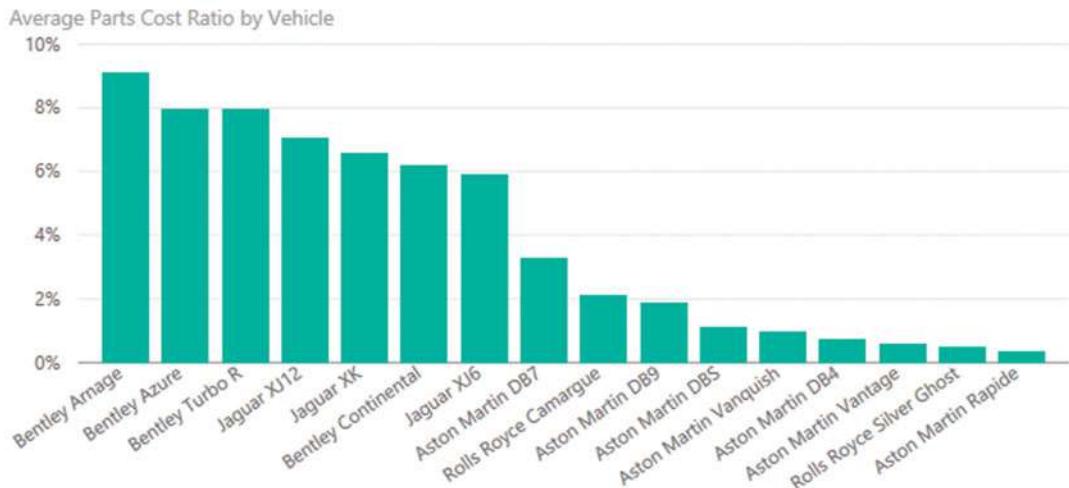
1. In Power BI Desktop, ensure that you are in Report View (the topmost of the view icons on the left below the ribbon).
2. Select the Stock table in the Fields list and click the New Measure button in the Modeling ribbon.
3. Replace Measure with the name that you want to use (**Average Parts Cost Ratio**).
4. Click to the right of the equals sign (=).
5. Enter **AVERAGEX** as the function, followed by a left parenthesis. A list of all the tables and fields in the data model will appear (including any columns and measures that you have added).
6. Select the Stock table.
7. Enter a comma.
8. Add a left parenthesis. This is to ensure that the subtraction is carried out before the division.
9. Enter a left bracket and select the [CostPrice] field or enter the field name, including the square brackets.
10. Add a minus sign after the field name (you can add spaces before and afterward if you want).

11. Enter a left bracket and select the [PartsCost] field or enter the field name including the square brackets.
12. Enter a right parenthesis. This matches the opening left parenthesis in step 7.
13. Add a forward slash (the divide by operator).
14. Enter a left bracket and select the [PartsCost] field or enter the field name, including the square brackets.
15. Enter a right parenthesis. This finishes the AVERAGEX function. The formula will look like this:

Average Parts Cost Ratio = AVERAGEX(Stock,([CostPrice]-[PartsCost]) / [PartsCost])

16. Press Enter (or click the check mark icon in the formula bar).

Just creating the formula is pretty meaningless. So take a look at the chart in Figure 8-8, where you can see how this ratio instantly shows you which models are the most costly as far as spare parts are concerned (for 2014 again).



**Figure 8-8.** Using the AVERAGEX() function in DAX

As you can see from this example—and unlike the AVERAGE function—AVERAGEX takes two inputs (or parameters as they are technically known):

- The *table* to which the formula is applied
- The *formula* to use, which is just as you would apply it to a calculated column

This formula deducts the CostPrice from the SalePrice for every row in the table, and then returns the average dependent on the filters and selections currently applied. This way, you always get the mathematically accurate result in your visualizations.

The following are the essential points to take away from this example:

- It is essential to wrap any field references in an aggregate function, such as SUM, AVERAGE, or COUNT, for an aggregated result to work. This is because the calculation (depending on the filters used) is not applied to only one record, but potentially several records, so data must be aggregated. Hence, the use of the SUM function in this example.
- You can, and indeed, must, nest calculations inside parentheses to force Power BI Desktop to calculate elements in the correct order. This functions exactly as it does in Excel, so I will not labor the point here.

There are many more of these “X” functions (which are generally known as *iterative functions*, as they iterate over an entire table) in Power BI. Table 8-2 outlines those that are currently available.

**Table 8-2.** DAX Iterative Functions

| Formula           | Description                                                                                                                   | Example                                                                 |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------|
| MINX()            | Calculates a value for each row in a table and displays the minimum value.                                                    | MINX(Stock, [PartsCost]+[LaborCost])                                    |
| MAXX()            | Calculates a value for each row in a table and displays the maximum value.                                                    | MAXX(Stock, [PartsCost]+[LaborCost])                                    |
| SUMX()            | Calculates a value for each row in a table and displays the total.                                                            | SUMX(Stock, [PartsCost]+[LaborCost])                                    |
| AVERAGEX()        | Calculates a value for each row in a table and displays the average of these values.                                          | AVERAGEX(Stock, [PartsCost]+[LaborCost])                                |
| COUNTX()          | Calculates a value for each row in a table and counts the resulting rows, including non-blank results of the calculation.     | COUNTX(Stock, [PartsCost]+[LaborCost])                                  |
| COUNTAX()         | Calculates a value for each row in a table and counts the resulting rows, not including non-blank results of the calculation. | COUNTAX (Stock, [PartsCost]+[LaborCost])                                |
| GEOMEANX()        | Calculates a value for each row in a table and displays the geometric mean of these values.                                   | GEOMEANX(Stock, [PartsCost]+[LaborCost])                                |
| MEDIANX()         | Calculates a value for each row in a table and displays the median value of these values.                                     | MEDIANX(Stock, [PartsCost]+[LaborCost])                                 |
| PERCENTILEX.EXC() | Returns the percentile of a record relative to the dataset.                                                                   | PERCENTILEX.EXC(Stock, [PartsCost]+[LaborCost])                         |
| PERCENTILEX.INC() | Returns the percentile of a record relative to the dataset.                                                                   | PERCENTILEX.INC(Stock, [PartsCost]+[LaborCost])                         |
| RANKX()           | Orders the rows by progressive rank.                                                                                          | RANKX(ALL(Stock[Make]), SUMX (RELATEDTABLE(InvoiceLines), [SalePrice])) |

(continued)

**Table 8-2.** (continued)

| Formula    | Description                                                                                                              | Example                                  |
|------------|--------------------------------------------------------------------------------------------------------------------------|------------------------------------------|
| STDEVX.P() | Calculates a value for each row in a table and displays the standard deviation of the entire population of these values. | STDEVX.P(Stock, [PartsCost]+[LaborCost]) |
| STDEVX.S() | Calculates a value for each row in a table and displays the standard deviation of a sample population of these values.   | STDEVX.S(Stock, [PartsCost]+[LaborCost]) |
| VARX.S()   | Calculates a value for each row in a table and displays the variance of the entire population of these values.           | VARX.S(Stock, [PartsCost]+[LaborCost])   |
| VARX.P()   | Calculates a value for each row in a table and displays the variance of the entire population of these values.           | VARX.P(Stock, [PartsCost]+[LaborCost])   |

## Filter Context

When working with DAX (at least if you want to develop any complex formulas), you need to understand *filter context*. This is the basis of the dynamic data analysis using Power BI Desktop. It is the basis of the approach where the results of a formula can change to reflect the current row or cell selection and any related filters.

Filter context can become an extremely complicated subject. However, since this is not a book on DAX, I am deliberately simplifying some of the ideas explained later in this chapter. After all, the aim is to get you started with DAX, not to scare you off right at the start.

The following are three key elements that you need to understand:

- Row context
- Query context
- Filter context

Let's take a brief look at these in turn.

## Row Context

*Row context* is essentially the values from the current row. You saw this when creating new columns. This means that any fields that you used in a calculation always used other fields from the same record—or from a linked table. Therefore, this is largely automatic and typically handled by DAX without any intervention on your part.

## Query Context

*Query context* is the combination of the following factors that produce a calculated result:

- Report-level filters
- Page-level filters

- Visualization-level filters
- Slicers
- Interactive selection
- Row and column filters

The first three are cumulative (report-level filters, page-level filters, and visualization-level filters), and reduce the available data that a visualization can show. They are explained in detail in Chapter 13. For the moment, just consider them as a set of filters that only allows certain data to be used.

Slicers are any interactive filtering that you add for a dashboard. These restrict even further the dataset resulting from any report-level filters, page-level filters and visualization-level filters to reduce even further the data that can be displayed. Interactive selection is a method of filtering data by selecting an element in another visualization. Both of these techniques are explained in detail in Chapter 14.

Row and column filters are best thought of as the row and column headers in a pivot table (or a cross-tab if you prefer). These define the intersections of data that can be shown.

Query context is the cumulative effect of any filters that you apply when creating Power BI reports using the Power BI Desktop interface.

## Filter Context

*Filter context* is added when you specify filter constraints on the set of values allowed in a column or table, by using formulas that expand or reduce the dataset that is used to obtain a result. Filter context applies on top of other contexts, such as row context or query context. This is the focus of the next few pages.

## Filtering Data in Measures

Inevitably, there will be times when the filters that you apply using the Power BI Desktop user interface (the query-level filters that were described earlier) are not quite what you are looking for. There could be several reasons for this, including the following:

- You want to apply a highly specific filter to a single metric.
- You want to override the natural result of the query-level filter.
- You are creating a highly complex formula and it has to be tailored to a specific use.

Any of these reasons (and there are many others that you will discover as you progress with DAX) could require you to filter the data in a measure. Let's look at a few circumstances where this could prove necessary. Given the wide-ranging possibilities of DAX, I do not intend to explain anything more than the basics of DAX filtering using a few simple examples that I hope you find practical when building your own dashboards.

## Simple Filters

There will probably be many occasions in your career when you need to home in on a specific subset of data. Maybe you need to compare and contrast one sales stream with another. Perhaps you need to highlight one cost compared to a total. Whatever the actual requirement, you need to apply a specific filter to a metric.

There are dozens—if not hundreds—of ways of applying different filters when building measures. However, one function is definitely an essential part of your DAX toolbox: the `CALCULATE()` function. This function lets you apply a range of filters to a measure that you can then apply in the visualizations that you build into your dashboards.

## Text Filters

To begin, let's look at a fairly simple filter requirement. Brilliant British Cars sells to two types of clients: dealers and wholesalers. As part of your ongoing sales analysis, you want to isolate the dealer sales stream. The following explains how you can do this.

1. Click the InvoiceLines table in the Fields list.
2. In the Modeling ribbon, click the New Measure button.
3. In the formula bar, replace Measure with **DealerSales**.
4. To the right of the equals sign, enter (or select) **CALCULATE()**.
5. Enter (or select) **SUM()**.
6. Select the InvoiceLines[SalePrice] field.
7. Add a right parenthesis. This will terminate the **SUM()** function.
8. Enter a comma. This tells the **CALCULATE()** function that you are about to add the filters.
9. Select the Clients[ClientType] field.
10. Enter an equals sign.
11. Add the word **"Dealer"** (include the double quotes).
12. Add a right parenthesis. This will terminate the **CALCULATE()** function. The formula should now read as follows:

**DealerSales = CALCULATE(SUM(InvoiceLines[SalePrice]),Clients[ClientType]="Dealer")**

You could then add this new measure to a simple table of sales by make. If you do, you will see something like Figure 8-9 (the data is for 2015).

| Make         | SalePrice          | DealerSales        |
|--------------|--------------------|--------------------|
| Aston Martin | £4,915,940         | £3,473,310         |
| Bentley      | £1,564,500         | £1,143,500         |
| Jaguar       | £2,939,500         | £2,269,250         |
| MGB          | £696,000           | £572,500           |
| Rolls Royce  | £3,982,600         | £2,914,550         |
| Triumph      | £639,000           | £479,250           |
| TVR          | £371,000           | £218,000           |
| <b>Total</b> | <b>£15,108,540</b> | <b>£11,070,360</b> |

**Figure 8-9.** Using a simple filter

You can see from this table that the SalePrice column is not filtered in any way. However, the DealerSales column always shows a smaller figure for the sales per make, as it is displaying only the filtered subset of data that you requested. The new measure that you created can be applied to any visualization and can be filtered, sliced and diced like any other data column, calculated column, or metric.

Now let me explain. Here we are using a function in DAX called `CALCULATE()`. This function does what its name implies; it calculates an aggregation. However, the calculation is nearly always a *filter* operation. This is because of the way in which its two parameters work:

- The first parameter defines the *function* to use (`SUM()` and `AVERAGE()` here), and the table and column that is aggregated; it could have been potentially a much more complex formula.
- The second parameter is a *filter* to force DAX to show only a subset of the data. In this specific case, it returns the sum of sales *only* when the client is a car dealership.

The filter that is applied here comes from another column. Indeed, it comes from another table altogether. When using the `CALCULATE()` function, you can use just about any column (either an original data column or a calculated column) as the source for a filter.

**Note** When you are filtering on a text (such as the type of dealer in this example), you *must* always enclose the text that you are searching for in double quotes.

## Numeric Filters

You are not restricted to filtering on text-based data only in Power BI Desktop. You can also subset data by numeric values. As an example, suppose that you want to see totals for sales for lower-priced models so that you can target the higher end of the market. The following explains how you could do this.

1. Click the `InvoiceLines` table in the Fields list.
2. In the Modeling ribbon, click the New Measure button.
3. In the formula bar, replace `Measure` with **LowPriceSales**.
4. To the right of the equals sign, enter (or select) `CALCULATE()`.
5. Enter (or select) `SUM()`.
6. Select the `InvoiceLines[SalePrice]` field.
7. Add a right parenthesis. This will terminate the `SUM()` function.
8. Enter a comma. This tells the `CALCULATE()` function that you are about to add the filters.
9. Select the `InvoiceLines[SalePrice]` field again.
10. Enter a less than sign: `<`.
11. Enter **50000**.
12. Add a right parenthesis. This will terminate the `CALCULATE()` function. The formula should now read:

```
LowPriceSales = CALCULATE(SUM(InvoiceLines[SalePrice]),
 InvoiceLines[SalePrice] < 50000)
```

Comparing the low price sales with the unfiltered sales per 2015 make and model produces a table like the one in Figure 8-10.

| Make         | Model         | SalePrice          | LowPriceSales     |
|--------------|---------------|--------------------|-------------------|
| Aston Martin | DB4           | £330,750           |                   |
| Aston Martin | DB7           | £394,990           | £168,490          |
| Aston Martin | DB9           | £3,146,850         | £492,600          |
| Aston Martin | Rapide        | £132,750           |                   |
| Aston Martin | Vanquish      | £489,750           |                   |
| Aston Martin | Vantage       | £239,600           | £162,350          |
| Aston Martin | Zagato        | £181,250           |                   |
| Bentley      | Azure         | £90,750            | £90,750           |
| Bentley      | Continental   | £1,272,000         | £708,250          |
| Bentley      | Turbo R       | £201,750           | £89,000           |
| Jaguar       | XJ6           | £501,000           | £388,250          |
| Jaguar       | XK            | £2,438,500         | £2,080,250        |
| MGB          | GT            | £696,000           | £696,000          |
| Rolls Royce  | Camarque      | £2,528,600         | £148,100          |
| Rolls Royce  | Phantom       | £181,250           |                   |
| Rolls Royce  | Silver Ghost  | £483,500           |                   |
| Rolls Royce  | Silver Seraph | £294,000           |                   |
| Rolls Royce  | Silver Shadow | £314,000           |                   |
| Rolls Royce  | Wraith        | £181,250           |                   |
| Triumph      | TR4           | £476,500           | £476,500          |
| Triumph      | TR5           | £109,250           | £109,250          |
| Triumph      | TR7           | £53,250            | £53,250           |
| TVR          | Cerbera       | £65,000            | £65,000           |
| TVR          | Tuscan        | £306,000           | £306,000          |
| <b>Total</b> |               | <b>£15,108,540</b> | <b>£6,034,040</b> |

**Figure 8-10.** Using a numeric filter

In this simple example you saw how to use the less than (<) comparison operator. You can use any of the standard logical comparison operators (=, <, >, <=, >=, >>) that you saw in the previous chapter.

---

**Note** When you are filtering on a number, you must *not* enclose the number that you are searching for in quotes. Neither must you format the number in any way.

---

## More Complex Filters

In the two previous examples, you saw the basics of creating filtered measures using either a text or a number to subset the data returned by the `CALCULATE()` function. In the real world of data analysis, filters can get a lot more complex. Indeed, allowing you to create specific and complex filters for metrics is one of the ways that DAX can help you tease out real insight from your data. So without attempting to get overly complicated, next are a few examples of the ways that you can define more complex filtered metrics in your data models.

### Multiple Criteria in Filters

The `CALCULATE()` function is not limited to a single filter. Far from it. You can add multiple filters to the second part of this function each separated by a comma.

For example, imagine that you didn't just want to see dealer sales when you look at your data, but also want to see the (slightly lower) figure for dealer sales where the client has a good credit status. This means combining two filter criteria.

1. Click the `InvoiceLines` table in the Fields list.
2. In the Modeling ribbon, click the New Measure button.
3. In the formula bar, replace `Measure` with **`DealerSales`**.
4. To the right of the equals sign, enter (or select) `CALCULATE()`.
5. Enter (or select) `SUM()`.
6. Select the `InvoiceLines[SalePrice]` field.
7. Add a right parenthesis. This will terminate the `SUM()` function.
8. Enter a comma. This tells the `CALCULATE()` function that you are about to add the filters.
9. Select the `Clients[ClientType]` field.
10. Enter an equals sign.
11. Add the word “**Dealer**” (include the double quotes ).
12. Enter a comma. This indicates that you are adding another filter criterion.
13. Select the `Clients[IsCreditWorthy]` field.
14. Enter an equals sign.
15. Add the word **`TRUE()`**. This is a logical value; it does not need to be enclosed in quotes and it has an empty parenthesis added.
16. Add a right parenthesis. This will terminate the `CALCULATE()` function. The formula should now read:

```
Creditworthy DealerSales = CALCULATE(SUM(InvoiceLines[SalePrice]),
Clients[ClientType]="Dealer",Clients[IsCreditWorthy]=TRUE())
```

If this column is added to the table that you saw in Figure 8-9, you will see a result something like the one in Figure 8-11.

| Make         | SalePrice          | DealerSales        | Creditworthy DealerSales |
|--------------|--------------------|--------------------|--------------------------|
| Aston Martin | £4,915,940         | £3,473,310         | £2,110,930               |
| Bentley      | £1,564,500         | £1,143,500         | £782,250                 |
| Jaguar       | £2,939,500         | £2,269,250         | £1,380,750               |
| MGB          | £696,000           | £572,500           | £348,000                 |
| Rolls Royce  | £3,982,600         | £2,914,550         | £2,068,550               |
| Triumph      | £639,000           | £479,250           | £319,500                 |
| TVR          | £371,000           | £218,000           | £153,000                 |
| <b>Total</b> | <b>£15,108,540</b> | <b>£11,070,360</b> | <b>£7,162,980</b>        |

**Figure 8-11.** Using a more complex filter

If anything, this was a simple example. The filters that you add to any measure that uses the `CALCULATE()` function can contain multiple elements. Also, you can create a series of filters where each filter element compares data from different tables and mixes both text-based and numeric filters.

## Using Multiple Filters

For a final filter example, imagine that you want to isolate the percentage of creditworthy dealer sales relative to dealer sales. You can do this by using the two measure calculations (`DealerSales` and `CreditworthyDealerSales`) in a single measure to obtain the desired result.

Since you just saw how to create these calculations, I will only show you the formula here:

```
Creditworthy DealerSales Percent = CALCULATE(SUM(InvoiceLines[SalePrice]),
Clients[ClientType]="Dealer",Clients[IsCreditWorthy]=TRUE()) /
CALCULATE(SUM(InvoiceLines[SalePrice]),Clients[ClientType]="Dealer")
```

As you can see (as you would in Excel), you can combine functions—even complex filtered functions—in a single metric to deliver powerful analysis. Using this measure in a simple table displaying sales by make for 2015 produces the results shown in Figure 8-12.

| Make         | SalePrice          | Creditworthy DealerSales Percent |
|--------------|--------------------|----------------------------------|
| Aston Martin | £4,915,940         | 60.78%                           |
| Bentley      | £1,564,500         | 68.41%                           |
| Jaguar       | £2,939,500         | 60.85%                           |
| MGB          | £696,000           | 60.79%                           |
| Rolls Royce  | £3,982,600         | 70.97%                           |
| Triumph      | £639,000           | 66.67%                           |
| TVR          | £371,000           | 70.18%                           |
| <b>Total</b> | <b>£15,108,540</b> | <b>64.70%</b>                    |

**Figure 8-12.** Using multiple filters in a measure

Now that you have learned how to create filtered measures, you have mastered the building blocks of an extremely powerful technique that you can adapt and extend in your own data models.

## Calculating Percentages of Totals

The filters that you have applied up until now in this chapter merely delivered subsets of data. Sometimes you need filters to do the opposite, and apply a calculation to an entire dataset. In other words, you need filters that *remove* filters. This is often because calculating a total means telling DAX to aggregate a column without applying any of the filtering by row that would normally be applied. In other words, you need to *prevent* the automatic filters that have proved so useful thus far.

### A Simple Percentage

Imagine a table where you want to calculate the percentage of a total that each row represents. This could be the total of sales by make, for instance. Here you need to simply divide the sales by the total sales.

1. Click the InvoiceLines table in the Fields list.
2. In the Modeling ribbon, click the New Measure button.
3. In the formula bar, replace Measure with **MakePercentage**.
4. To the right of the equals sign, enter (or select) **DIVIDE(**.
5. Enter (or select) **SUM(**.
6. Select the **InvoiceLines[SalePrice]** field.
7. Add a right parenthesis. This will terminate the **SUM()** function.
8. Enter a comma.
9. Enter or select **CALCULATE(**.
10. Enter (or select) **SUM(**.
11. Select the **InvoiceLines[SalePrice]** field.
12. Add a right parenthesis. This will terminate the **SUM()** function.
13. Enter a comma. This tells the **CALCULATE()** function that you are about to add the filters.
14. Enter or select **ALL(**.
15. Select the **Stock[Make]** field.
16. Add a right parenthesis. This will terminate the **ALL()** function.
17. Add a right parenthesis. This will terminate the **CALCULATE()** function.
18. Add a right parenthesis. This will terminate the **DIVIDE()** function. The formula should now read as follows:

```
MakePercentage = DIVIDE(SUM(InvoiceLines[SalePrice]),
CALCULATE(SUM(InvoiceLines[SalePrice]), ALL(Stock[Make])))
```

If you create a simple table of sales per make for 2015 and add this new measure, it should look like Figure 8-13.

| Make         | SalePrice         | Make Percentage |
|--------------|-------------------|-----------------|
| Aston Martin | £2,262,440        | 35.43%          |
| Bentley      | £1,811,500        | 28.36%          |
| Jaguar       | £1,352,000        | 21.17%          |
| Rolls Royce  | £960,500          | 15.04%          |
| <b>Total</b> | <b>£6,386,440</b> | <b>100.00%</b>  |

**Figure 8-13.** Using the ALL() function to calculate a percentage per attribute

This formula and the concept behind it probably seem a little peculiar. So let me explain the ALL() function in greater detail. In essence, the ALL() function says, “Remove all the filters concerning any specified fields.” Consequently, in this example, the make is *not* filtered when calculating the total sales. This means that the unfiltered total can now be calculated—and so can the percentage of each make relative to this grand total.

It is important to note that the ALL() function only removes filters for the fields that you have specified. For instance, look at Figure 8-14, which shows a matrix for the sales and the percentage by make for two years.

| FullYear<br>Make | 2014              |                | 2015               |                | Total              |                |
|------------------|-------------------|----------------|--------------------|----------------|--------------------|----------------|
|                  | SalePrice         | MakePercentage | SalePrice          | MakePercentage | SalePrice          | MakePercentage |
| Aston Martin     | £2,262,440        | 35.43%         | £4,915,940         | 32.54%         | £7,178,380         | 33.40%         |
| Rolls Royce      | £960,500          | 15.04%         | £3,982,600         | 26.36%         | £4,943,100         | 23.00%         |
| Jaguar           | £1,352,000        | 21.17%         | £2,939,500         | 19.46%         | £4,291,500         | 19.97%         |
| Bentley          | £1,811,500        | 28.36%         | £1,564,500         | 10.36%         | £3,376,000         | 15.71%         |
| MGB              |                   |                | £696,000           | 4.61%          | £696,000           | 3.24%          |
| Triumph          |                   |                | £639,000           | 4.23%          | £639,000           | 2.97%          |
| TVR              |                   |                | £371,000           | 2.46%          | £371,000           | 1.73%          |
| <b>Total</b>     | <b>£6,386,440</b> | <b>100.00%</b> | <b>£15,108,540</b> | <b>100.00%</b> | <b>£21,494,980</b> | <b>100.00%</b> |

**Figure 8-14.** The ALL() function lets all other filters be applied

You can see here that the measure MakePercentage is correctly applied independently to sales for 2014, 2015, and the grand total. This is because all other filters (the year in this case) are applied as you have come to expect with Power BI Desktop; *only* the make is not filtered when calculating the total sales.

## Removing Multiple Filter Elements

If your visualization is more complex than the simple example that you just saw, then you have to craft your measures appropriately to handle any complexity. For instance, take the case where you want to see sales by make and color, and display the percentage of each row compared to the total. You need to calculate a total that discards the filters for make *and* color so that DAX can arrive at the correct figure for the overall total. Here is the formula that can do this:

```
MakeAndColorPercentage = DIVIDE(SUM(InvoiceLines[SalePrice]), CALCULATE(SUM(InvoiceLines[SalePrice]), ALL(Stock[Make]), ALL(Colors[Color])))
```

Since this code snippet is only an extension of the previous one, I have not explained how to construct it in detail. All you have to do is to follow the steps from the previous example and add a second filter to the second `CALCULATE()` function (the one that returns the grand total of sales). As you saw earlier in this chapter, the `CALCULATE()` function can take multiple filter parameters. It follows that it can also take multiple “unfilter” parameters, as it is doing here. The key is in the following part of the measure:

```
ALL(Stock[Make]), ALL(Colors[Color])
```

This piece of DAX is simply saying, “Don’t apply any make or color filters when calculating.” The consequence is that this measure now calculates the total sales whatever the make or color. This then becomes the basis for the percentage calculation, as you can see in Figure 8-15.

| Make         | Color                | SalePrice         | MakeAndColorPercentage |
|--------------|----------------------|-------------------|------------------------|
| Aston Martin | Black                | £195,500          | 3.06%                  |
| Aston Martin | Blue                 | £511,000          | 8.00%                  |
| Aston Martin | Canary Yellow        | £235,940          | 3.69%                  |
| Aston Martin | Dark Purple          | £225,500          | 3.53%                  |
| Aston Martin | Night Blue           | £142,500          | 2.23%                  |
| Aston Martin | Red                  | £481,000          | 7.53%                  |
| Aston Martin | Silver               | £471,000          | 7.38%                  |
| Bentley      | Black                | £225,500          | 3.53%                  |
| Bentley      | Blue                 | £84,500           | 1.32%                  |
| Bentley      | British Racing Green | £225,500          | 3.53%                  |
| Bentley      | Canary Yellow        | £863,500          | 13.52%                 |
| Bentley      | Red                  | £319,000          | 4.99%                  |
| Bentley      | Silver               | £93,500           | 1.46%                  |
| Jaguar       | Black                | £84,500           | 1.32%                  |
| Jaguar       | Blue                 | £225,500          | 3.53%                  |
| Jaguar       | British Racing Green | £189,000          | 2.96%                  |
| Jaguar       | Canary Yellow        | £497,000          | 7.78%                  |
| Jaguar       | Dark Purple          | £178,000          | 2.79%                  |
| Jaguar       | Green                | £178,000          | 2.79%                  |
| Rolls Royce  | Blue                 | £183,500          | 2.87%                  |
| Rolls Royce  | British Racing Green | £210,500          | 3.30%                  |
| Rolls Royce  | Green                | £181,500          | 2.84%                  |
| Rolls Royce  | Silver               | £385,000          | 6.03%                  |
| <b>Total</b> |                      | <b>£6,386,440</b> | <b>100.00%</b>         |

**Figure 8-15.** Using the `ALL()` function to calculate a percentage per attribute

Admittedly, preparing highly specific measures like those that you have seen in the last few pages can take a few minutes. Clearly, these measures are tightly linked to the data that you are displaying in the visuals that use them. However, the ability to compose highly focused metrics like these is often key to using your dashboards to highlight the insights that you want to deliver.

## Visual Totals

Users (meaning the target audience for your reports and dashboards) do not like anomalies or apparent contradictions. So you have to be sure that the data that they see is visually coherent. This is especially true when displaying tables and matrices with subtotals and grand totals where you want percentage totals to reflect the figures shown and not include any records that have been removed by the filter.

One technique that can help you here is the ALLSELECTED() function. This only applies any filters that have been added (either at report-, page- or visualization-level, or as slicers or cross-filters from other visuals) *without* you having to specify the fields that you do not want to filter as you did in the previous examples.

SalesPercentage is a measure that uses the ALLSELECTED() function as the filter for the CALCULATE() function that returns the total much like you did earlier.

```
SalesPercentage = DIVIDE(SUM(InvoiceLines[SalePrice]), CALCULATE(SUM(InvoiceLines[SalePrice]), ALLSELECTED())))
```

If you look at Figure 8-16, you see that the subtotals for the sales percentage (and indeed the grand total) are accurate, despite the fact that the country (USA) is selected in a slicer and the year (2014) in a filter.

| Make         | Color                | SalePrice  | SalesPercentage |
|--------------|----------------------|------------|-----------------|
| Aston Martin | Black                | £97,750    | 2.72%           |
|              | Blue                 | £255,500   | 7.10%           |
|              | Canary Yellow        | £138,190   | 3.84%           |
|              | Dark Purple          | £112,750   | 3.13%           |
|              | Night Blue           | £71,250    | 1.98%           |
|              | Red                  | £353,250   | 9.82%           |
|              | Silver               | £235,500   | 6.54%           |
|              | Total                | £1,264,190 | 35.13%          |
| Bentley      | Black                | £225,500   | 6.27%           |
|              | Blue                 | £42,250    | 1.17%           |
|              | British Racing Green | £112,750   | 3.13%           |
|              | Canary Yellow        | £431,750   | 12.00%          |
|              | Red                  | £159,500   | 4.43%           |
|              | Silver               | £46,750    | 1.30%           |
|              | Total                | £1,018,500 | 28.30%          |
| Jaguar       | Black                | £42,250    | 1.17%           |
|              | Blue                 | £225,500   | 6.27%           |
|              | British Racing Green | £94,500    | 2.63%           |
|              | Canary Yellow        | £248,500   | 6.91%           |
|              | Dark Purple          | £135,750   | 3.77%           |
|              | Green                | £89,000    | 2.47%           |
|              | Total                | £835,500   | 23.22%          |
| Rolls Royce  | Blue                 | £91,750    | 2.55%           |
|              | British Racing Green | £105,250   | 2.92%           |
|              | Green                | £90,750    | 2.52%           |
|              | Silver               | £192,500   | 5.35%           |
|              | Total                | £480,250   | 13.35%          |
| Total        |                      | £3,598,440 | 100.00%         |

Figure 8-16. Using the ALLSELECTED() function to calculate a percentage per attribute per group

The ALLSELECTED() function says to DAX “don’t filter on any filters applied by the user, whatever the technique used to apply them.” This can make creating percentage totals much easier, as this function removes the need to create highly specific measures that are tied to specific levels of totals and subtotals.

## The ALLEXCEPT() Function

In practice, you could find yourself having to write extremely targeted measures that need to remove filters from all the elements in a calculation except one or two. So, to save you writing long lists of ALL() functions, you can say “All but” a field using the ALLEXCEPT() function.

As an example of this (although it is extremely simple), suppose that you want to see the percentages of sales grouped by a subclassification. You know that you want to have Make as the main grouping element, but then you might want to use Color, Client, or even Model as the subgroup. So to save you having to write a measure specifically for each of these combinations, you can write the following:

```
AllButMakePercentage = DIVIDE(SUM(InvoiceLines[SalePrice]), CALCULATE(SUM(InvoiceLines[SalePrice]), ALLEXCEPT(Stock, Stock[Make])))
```

If you use this measure in a matrix where Make is the leftmost column, you can then add subgroups using any other field to get the kind of output that is shown in Figure 8-17.

| Make         | Color                | SalePrice          | AllButMakePercentage |
|--------------|----------------------|--------------------|----------------------|
| Bentley      | Black                | £473,000           | 9.46%                |
|              | Blue                 | £297,250           | 5.95%                |
|              | British Racing Green | £459,500           | 9.19%                |
|              | Canary Yellow        | £1,265,000         | 25.31%               |
|              | Dark Purple          | £363,250           | 7.27%                |
|              | Green                | £335,500           | 6.71%                |
|              | Night Blue           | £610,500           | 12.21%               |
|              | Red                  | £1,056,500         | 21.14%               |
|              | Silver               | £137,500           | 2.75%                |
|              | Total                | £4,998,000         | 100.00%              |
| Jaguar       | Black                | £416,000           | 6.58%                |
|              | Blue                 | £818,000           | 12.95%               |
|              | British Racing Green | £663,500           | 10.50%               |
|              | Canary Yellow        | £1,750,250         | 27.70%               |
|              | Dark Purple          | £647,000           | 10.24%               |
|              | Green                | £425,000           | 6.73%                |
|              | Night Blue           | £118,500           | 1.88%                |
|              | Red                  | £860,750           | 13.62%               |
|              | Silver               | £620,000           | 9.81%                |
|              | Total                | £6,319,000         | 100.00%              |
| <b>Total</b> |                      | <b>£11,317,000</b> | <b>100.00%</b>       |

**Figure 8-17.** Using the ALLEXCEPT() function to calculate a percentage per attribute per group

In this example, other filters (color here) are applied, but not make. So you are displaying the percentage for each color compared to the aggregate total for the make.

---

**Note** ALLEXCEPT() does what it says and removes all filters except the one that you specify. This can have the effect of preventing other filters from working as you expect.

---

## Filtering on Measures

The CALCULATE() function is without a doubt one of the most powerful functions that you will use in DAX. However, there are a few things that it cannot do. One of these is to filter data by comparing to a *measure* rather than to a column. If you cast your mind back to the examples where CALCULATE() was applied, you will remember that a data column or a calculated column was used every time that a comparison (text-based or numeric) was invoked. Indeed, if you try to use CALCULATE() with a measure rather than a column, you will get an error.

Fortunately, DAX has a solution to this conundrum, which is to use the FILTER() function. You may well wonder what the differences are between FILTER() and CALCULATE(). Well, at its simplest, FILTER() can use *measures* as part of a comparison, whereas CALCULATE() must use columns—or calculated columns. Also, FILTER() *must* use an iterator function (such as SUMX()) rather than a simple aggregation function to produce a correct result.

Let's see this in action. Suppose that you want to isolate sales where the ratio of net margin is over 50%. Fortunately, you have a measure—RatioNetMargin—that calculates the percentage. The following explains how you can use this measure in a filter so that you can display these lucrative sales.

1. Click the InvoiceLines table in the Fields list.
2. In the Modeling ribbon, click the New Measure button.
3. In the formula bar, replace Measure with **HighNetMarginSales**.
4. To the right of the equals sign, enter (or select) **CALCULATE(**.
5. Enter (or select) **SUM(**.
6. Select the InvoiceLines[SalePrice] field.
7. Add a right parenthesis. This will terminate the SUM() function.
8. Enter a comma. This tells the CALCULATE() function that you are about to add the filters.
9. Enter or select **FILTER(**.
10. Select the InvoiceLines[RatioNetMargin] field. This is the field to filter on.
11. Enter a comma. This tells the FILTER() function that you are about to enter the filter criteria.
12. Enter the greater than symbol: **>**.
13. Enter the figure **0.5**.

14. Add a right parenthesis. This will terminate the FILTER() function.
15. Add a right parenthesis. This will terminate the CALCULATE() function.  
The formula should now read:

```
HighNetMarginSales = CALCULATE(SUM(InvoiceLines[SalePrice]),
 FILTER(InvoiceLines, [RatioNetMargin]>0.5))
```

If you use this measure in a table of sales by make and model (for 2015), you should see something like Figure 8-18.

| Make         | Model         | SalePrice          | HighNetMarginSales |
|--------------|---------------|--------------------|--------------------|
| Aston Martin | DB4           | £330,750           |                    |
| Aston Martin | DB7           | £394,990           | £275,050           |
| Aston Martin | DB9           | £3,146,850         | £1,802,400         |
| Aston Martin | Rapide        | £132,750           |                    |
| Aston Martin | Vanquish      | £489,750           | £362,500           |
| Aston Martin | Vantage       | £239,600           | £174,350           |
| Aston Martin | Zagato        | £181,250           |                    |
| Bentley      | Azure         | £90,750            |                    |
| Bentley      | Continental   | £1,272,000         | £563,750           |
| Bentley      | Turbo R       | £201,750           | £112,750           |
| Jaguar       | XJ6           | £501,000           | £112,750           |
| Jaguar       | XK            | £2,438,500         | £442,750           |
| MGB          | GT            | £696,000           | £494,000           |
| Rolls Royce  | Camargue      | £2,528,600         | £2,014,100         |
| Rolls Royce  | Phantom       | £181,250           | £181,250           |
| Rolls Royce  | Silver Ghost  | £483,500           | £132,750           |
| Rolls Royce  | Silver Seraph | £294,000           | £181,250           |
| Rolls Royce  | Silver Shadow | £314,000           | £314,000           |
| Rolls Royce  | Wraith        | £181,250           | £181,250           |
| Triumph      | TR4           | £476,500           |                    |
| Triumph      | TR5           | £109,250           |                    |
| Triumph      | TR7           | £53,250            |                    |
| TVR          | Cerbera       | £65,000            |                    |
| TVR          | Tuscan        | £306,000           |                    |
| <b>Total</b> |               | <b>£15,108,540</b> | <b>£7,344,900</b>  |

**Figure 8-18.** Applying a filter to a measure

In this example, you filtered data on a measure (RatioNetMargin) rather than a column. Be aware, however, that the FILTER() function can be slow when applied to large datasets.

## Displaying Rank

DAX can do so much when it comes to preparing metrics for BI delivery that it is hard to know exactly what you need and when. The final example in this short tour of DAX measures explains how to rank sales by make. I realize that you can do this just by sorting records, but should you need a clear and unequivocal indicator of ranking, then here is how it can be done:

1. Click the InvoiceLines table in the Fields list.
2. In the Modeling ribbon, click the New Measure button.
3. In the formula bar, replace Measure with **SalesRankByMake**.
4. To the right of the equals sign, enter (or select) **RANKX()**.
5. Enter (or select) **ALL()**.
6. Select the Stock[Make] field.
7. Add a right parenthesis. This will terminate the **ALL()** function.
8. Enter a comma. This tells the **RANK()** function that you are going to enter the calculation of how to order the data.
9. Enter or select **SUMX()**.
10. Enter or select **RELATEDTABLE()**.
11. Select the InvoiceLines table. This is the table where the data is to be sourced.
12. Add a right parenthesis. This will terminate the **RELATEDTABLE()** function.
13. Enter a comma. This tells the **RELATEDTABLE()** function that you are about to enter the field to use.
14. Enter or select **[SalePrice]**.
15. Add a right parenthesis. This will terminate the **SUMX()** function.
16. Add a right parenthesis. This will terminate the **RANKX()** function. The formula should now read:

```
SalesRankByMake = RANKX(ALL(Stock[Make]),SUMX(RELATEDTABLE
(InvoiceLines), [SalePrice]))
```

If you apply this measure to a simple table that lists the makes sold (in 2014, for instance) and then sort by the SalesRankByMake field, you will see something like Figure 8-19.

| Make         | SalesRankByMake |
|--------------|-----------------|
| Aston Martin | 1               |
| Bentley      | 4               |
| Jaguar       | 3               |
| MGB          | 5               |
| Rolls Royce  | 2               |
| Triumph      | 6               |
| TVR          | 7               |

**Figure 8-19.** Using the **RANKX()** function to classify data

As its name implies, RANKX() ranks the first field using the order returned by the descending output of the second field.

## A Few Comments and Notes on Using Measures

Measures are an immense subject. The breadth and depth of the calculations that can be delivered using DAX are little short of astounding. Consequently, it is impossible in an introductory chapter on measures to do anything other than give you a taste of what can be done and provide a few useful starter functions for you to adapt to your own requirements.

As you move on with DAX, a few things might help you on your way. The first concerns the use of calculated columns. Sometimes they are such an easy solution that it is a shame not to create them. However, they are stored in the table and do take up space. This means more space on disk and more space in memory. This is particularly true for a table containing tens of millions of rows. Measures, on the other hand, are only calculated at run time, and so they take up virtually no space. So, if you are considering creating many calculated columns, perhaps some of them could become measures instead.

## Calculation Options

I imagine that you have not had to worry about recalculation of Power BI Desktop workbooks if you have been using relatively small datasets like the sample data for this book. If you are using vast amounts of data (after all, this is what Power BI Desktop was designed for), however, then recalculation could become a subject that you need to master.

By default, Power BI Desktop recalculates all calculated columns and measures when there is a change in the dataset. These are the main operations that can trigger a recalculation:

- Data from an external data source (of any kind) have been updated.
- Data from an external data source have been filtered.
- You have changed the name of a table or column.
- You have added, modified, or deleted relationships between tables.
- You have altered any formula for a calculated column or a measure.
- You have added new calculated columns or measures.

More generally, if you want to be sure that your data is up-to-date, you should probably update the data. You can do this by clicking the Refresh button in the Home ribbon.

## Conclusion

In this chapter, you took a first look at one of the most powerful features in DAX: measures. These let you develop custom calculations for the Power BI Desktop data model. You then use these metrics in your visuals to deliver specific insights based on your data.

First, you saw how to apply iterator functions so that you can apply a calculation to a set of rows and return an aggregation, be it a sum, average, or any other available aggregate function. Then you saw how to apply specific filters to your calculations. Finally, you saw how to prevent filters from being applied so that you can display percentages and calculate advanced ratios.

This chapter was only a brief introduction to DAX. Yet I hope that it has whetted your appetite and that you can now develop the analyses and metrics that you need for your own data.

It will soon be time to start applying these measures to a range of visualisations in Power BI Desktop. However before then let's take a look at one final fundamental element of the data model that you will need to set up before you can extract real value from your data. The remaining piece of the puzzle is adding time intelligence, and this is the subject of the next chapter.

## CHAPTER 9



# Analyzing Data over Time

Most data analysis—and nearly all business intelligence—Involves looking at how metrics evolve over time. You may need to aggregate sales by month, week, or year for instance. Perhaps you want to compare figures for a previous month, quarter, or year with the figures for a current period. Whatever the exact requirement, handling time (by which we nearly always mean dates) is essential in Power BI Desktop.

Initially, using time functions in Power BI Desktop may only be limited to extracting time intervals from the available data and grouping results by units of time, such as days, weeks, months, quarters, and years. As you will find out in the first part of this chapter, DAX makes this kind of analysis really simple.

However, Power BI Desktop can also add what is called *time intelligence* to data models. This massively useful capability can take your analysis to a fundamentally higher level. This approach involves adding a separate table (called a *date* or *time dimension*) to the data model and then adding DAX functions that enable you to see how data evolves over time. Consequently, this chapter also includes an introduction to time intelligence using a wide range of DAX formulas that are available to help you create time-based calculations quickly and easily.

In this chapter, we continue to develop the PowerBIDataModel.Pbix file that you extended in the two previous chapters. Should you need it, a complete version of this file is available on the Apress web site, including the extensions added in the previous chapter as PowerBIDataModelExtendedWithMetrics.Pbix.

## Simple Date Calculations

Data analysis often involves looking at how key metrics evolve over time. Power BI Desktop can help you group and isolate date and time elements in your data. Since dates (and time) are often a continuous stream of dates in a dataset, it can be useful to isolate the years, months, weeks, and days in a table alongside the date that a row contains so that you can create tables or visuals that group and aggregate records into these more comprehensible “buckets.” You can then display and compare data over years and months, for instance, in order to tease out the real insights that your raw data contains.

For the first example of how DAX can help you to categorize records using date-based criteria, let's imagine that you envisage creating a couple of charts. First, you need one that lets you track sales over the years that Brilliant British Cars has traded. Then you want a second graphic that looks at sales for each month over the years. Unfortunately (for the moment at least), your data model does not contain columns that show the year or the month of a sale, and Power BI Desktop does not let you create metrics as part of a visualization. You have to have the metric available in the data model if you plan to use it in a dashboard element. Moreover, as you will see in Chapters 10 through 15, it is best if you have all the metrics in place before you create any visualizations. Indeed, this is precisely the reason that you are learning how to extend the data model with new columns using DAX. The following explains how to create these two new columns in the Invoices table.

1. Select the InvoiceLines table in the Fields list.
2. In the Modeling ribbon, click the New Column button.
3. In the formula bar to the left of the equals sign, replace Column with **Sale Year**.
4. Click the right of the equals sign.
5. Enter (or start typing) and then select YEAR(.
6. Enter a left square bracket: [.
7. Select the InvoiceDate field.
8. Add a final right parenthesis to complete the YEAR() function.
9. Confirm the formula by pressing Enter or clicking the tick icon in the formula bar.  
The new column will display the year of each sale.
10. Repeat steps 2 through 9 only use the DAX formula MONTH() instead of YEAR(), and name the new column Sale Month.

The formula for the two new columns will be:

Sale Month = MONTH([InvoiceDate])

Sale Year = YEAR([InvoiceDate])

Figure 9-1 shows what the Invoices table looks like with these two new columns added.



The screenshot shows a Power BI Data View window with the 'Invoices' table selected. The table has 14 columns: InvoiceID, InvoiceNumber, ClientID, InvoiceDate, TotalDiscount, DeliveryCharge, InvoiceDateKey, SaleYear, and SaleMonth. The SaleYear and SaleMonth columns are highlighted in yellow, indicating they are newly created columns. The data in the table spans from April 2012 to December 2012, with various invoice numbers and client IDs.

| InvoiceID | InvoiceNumber                         | ClientID | InvoiceDate | TotalDiscount | DeliveryCharge | InvoiceDateKey | SaleYear | SaleMonth |
|-----------|---------------------------------------|----------|-------------|---------------|----------------|----------------|----------|-----------|
| 1         | 883D7F83-F42C-4523-A737-CDCBF7705B77  | 1        | 04/10/2012  | 500           | £750           | 20121004       | 2012     | 10        |
| 2         | 1398EEEF-FF32-4BE9-9EF1-819AC888B85C  | 2        | 01/01/2012  | 0             | £1,500         | 20120101       | 2012     | 1         |
| 3         | D35D72CD-5FF3-4701-A6D1-265A4F4E7CD5  | 3        | 02/02/2012  | 750           | £1,000         | 20120202       | 2012     | 2         |
| 4         | 2ABAAB300-E2A5-4E37-BFCA-7B80ED88A2BD | 2        | 03/03/2012  | 0             | £1,000         | 20120303       | 2012     | 3         |
| 5         | A1C2D846-EC39-46FA-A399-0C194AAD4DC8  | 5        | 04/04/2012  | 0             | £1,500         | 20120404       | 2012     | 4         |
| 6         | 188F325A-C4A1-4BA6-A486-9D44962E40A3  | 4        | 04/05/2012  | 0             | £1,000         | 20120504       | 2012     | 5         |
| 7         | F1B566F0-D137-4810-B449-575438F3F392  | 3        | 04/06/2012  | 750           | £500           | 20120604       | 2012     | 6         |
| 8         | ADFFAC9E-0FF3-4BAB-9ECA-DEE9A2B69350  | 6        | 04/07/2012  | 2500          | £1,000         | 20120704       | 2012     | 7         |
| 9         | 15A3B6C61-828D-4CCD-8F08-49EEE59AF4B7 | 1        | 04/08/2012  | 0             | £1,000         | 20120804       | 2012     | 8         |
| 10        | CFC6726D-1522-4981-BC68-766AE2C6EE00  | 1        | 04/09/2012  | 0             | £1,000         | 20120904       | 2012     | 9         |
| 11        | C4A55876-3893-4D12-89EF-D381C9CB642B  | 6        | 04/11/2012  | 0             | £1,500         | 20121104       | 2012     | 11        |
| 12        | 4DFCF7EF-C853-4D75-8584-75F6D752DE92  | 5        | 04/11/2012  | 0             | £1,500         | 20121104       | 2012     | 11        |
| 13        | B47CA156-7077-4690-AA1A-0CF4519B86FA  | 3        | 04/12/2012  | 5000          | £1,500         | 20121204       | 2012     | 12        |
| 14        | 7DBAF8FB-E346-4B8D-9CE0-1FB86B458BEE  | 4        | 04/12/2012  | 0             | £1,500         | 20121204       | 2012     | 12        |

**Figure 9-1.** The YEAR() and MONTH() DAX functions

In Chapters 10 through 15, you see how to use metrics like these to create visualizations that explore the way that sales have evolved over time.

---

**Note** To extract part of a date like this the column that you are using for the original data must be of the date data type or be capable of being interpreted as a date by Power BI Desktop.

---

This example illustrated two of the DAX date and time functions. Inevitably, there are many other functions that you can apply to extract a date or time element from a date field. Since they all follow the same principles as those that you have just seen (with the exception of the NOW() and TODAY() functions that are explained in a couple of pages time), it is easier to list them in Table 9-1 rather than provide a set of nearly identical examples.

**Table 9-1.** DAX Date and Time Functions

| Function         | Description                                                                                          | Example                                           |
|------------------|------------------------------------------------------------------------------------------------------|---------------------------------------------------|
| YEAR()           | Extracts the year element from a date.                                                               | YEAR([InvoiceDate])                               |
| MONTH()          | Extracts the month number from a date.                                                               | MONTH([InvoiceDate])                              |
| DAY()            | Extracts the day number from a date.                                                                 | DAY([InvoiceDate])                                |
| WEEKDAY()        | Extracts the weekday from a date. Sunday is 1, Monday is 2, and so forth.                            | WEEKDAY([InvoiceDate])                            |
| WEEKNUM()        | Extracts the number of the week in the year from a date.                                             | WEEKNUM([InvoiceDate])                            |
| HOUR()           | Extracts the hour from a time or datetime column.                                                    | HOUR([InvoiceDate])                               |
| MINUTE()         | Extracts the minutes from a time or datetime column.                                                 | MINUTE([InvoiceDate])                             |
| SECOND()         | Extracts the seconds from a time or datetime column.                                                 | SECOND([InvoiceDate])                             |
| EOMONTH()        | Returns the last day of the month from a date.                                                       | EOMONTH([InvoiceDate])                            |
| NOW()            | Returns the current date and time.                                                                   | NOW()                                             |
| TODAY()          | Returns the current date.                                                                            | TODAY()                                           |
| DATE()           | Lets you enter a date as year, month and day.                                                        | DATE(2015, 07, 25)                                |
| TIME()           | Lets you enter a time as hours and minutes.                                                          | TIME(19, 57)                                      |
| DATEDIFF()       | Calculates the difference between two dates and/or times expressed as a number of specified periods. | DATEDIFF([InvoiceDate], DATE(2025, 07, 25), YEAR) |
| STARTOFMONTH()   | Selects the first day of the month for a date.                                                       | STARTOFMONTH<br>(Invoices[SaleDate])              |
| STARTOFQUARTER() | Selects the first day of the quarter for a date.                                                     | STARTOFQUARTER<br>(Invoices[SaleDate])            |
| STARTOFTYEAR()   | Selects the first day of the year for a date.                                                        | STARTOFTYEAR<br>(Invoices[SaleDate])              |
| ENDOFMONTH()     | Selects the last day of the month for a date.                                                        | ENDOFMONTH<br>(Invoices[SaleDate])                |
| ENDOFQUARTER()   | Selects the last day of the quarter for a date.                                                      | ENDOFQUARTER<br>(Invoices[SaleDate])              |
| ENDOFTYEAR()     | Selects the last day of the year for a date.                                                         | ENDOFTYEAR<br>(Invoices[SaleDate])                |

**Note** If you define a field as having a date or datetime data type, then Power BI Desktop always creates a hierarchy of Year ▶ Quarter ▶ Month ▶ Day in the visual that you are creating. This can become annoying when all you need is the day. So it is well worth creating a well-structured date dimension table, as described later in this chapter, so that you only use the exact date element that you want, rather than have Power BI Desktop make assumptions for you.

---

## Date and Time Formatting

When dealing with dates and times in Power BI Desktop you may not need to go to the lengths of extracting a part of a date field, but may simply need to display a date in a different way. Rather like Excel, DAX can help you to do this quickly and easily.

Suppose that you want to have the InvoiceDate field displayed in a specific date format for use in certain visualizations. The following explains how you can do this.

1. Select the InvoiceLines table in the Fields list.
2. In the Modeling ribbon, click the New Column button.
3. In the formula bar to the left of the equals sign, replace Column with **UKDate**.
4. Click to the right of the equals sign.
5. Enter (or start typing and then select) **FORMAT()**.
6. Enter a left square bracket: [.
7. Select the InvoiceDate field.
8. Add a comma.
9. Enter the format code “**D-MMM-YYYY**” (include the double quotes).
10. Add a final right parenthesis to complete the **FORMAT()** function.
11. Replace the word Column with **UKDate**. The formula bar will display the following code:

```
UKDate = FORMAT([InvoiceDate], "D-MMM-YYYY")
```

12. Confirm the formula by pressing Enter or clicking the tick icon in the formula bar.  
The new column will display the sale date in a different format.

If you look at Figure 9-2, you see what the newly formatted invoice data field looks like in the new column.

| InvoiceDate | TotalDiscount | DeliveryCharge | InvoiceDateKey | SaleYear | SaleMonth | UKDate     |
|-------------|---------------|----------------|----------------|----------|-----------|------------|
| 04/10/2012  | 500           | £750           | 20121004       | 2012     | 10        | 4-Oct-2012 |
| 01/01/2012  | 0             | £1,500         | 20120101       | 2012     | 1         | 1-Jan-2012 |
| 02/02/2012  | 750           | £1,000         | 20120202       | 2012     | 2         | 2-Feb-2012 |
| 03/03/2012  | 0             | £1,000         | 20120303       | 2012     | 3         | 3-Mar-2012 |
| 04/04/2012  | 0             | £1,500         | 20120404       | 2012     | 4         | 4-Apr-2012 |
| 04/05/2012  | 0             | £1,000         | 20120504       | 2012     | 5         | 4-May-2012 |
| 04/06/2012  | 750           | £500           | 20120604       | 2012     | 6         | 4-Jun-2012 |
| 04/07/2012  | 2500          | £1,000         | 20120704       | 2012     | 7         | 4-Jul-2012 |
| 04/08/2012  | 0             | £1,000         | 20120804       | 2012     | 8         | 4-Aug-2012 |
| 04/09/2012  | 0             | £1,000         | 20120904       | 2012     | 9         | 4-Sep-2012 |
| 04/11/2012  | 0             | £1,500         | 20121104       | 2012     | 11        | 4-Nov-2012 |
| 04/11/2012  | 0             | £1,500         | 20121104       | 2012     | 11        | 4-Nov-2012 |
| 04/12/2012  | 5000          | £1,500         | 20121204       | 2012     | 12        | 4-Dec-2012 |
| 04/12/2012  | 0             | £1,500         | 20121204       | 2012     | 12        | 4-Dec-2012 |
| 02/01/2013  | 0             | £1,750         | 20130102       | 2013     | 1         | 2-Jan-2013 |
| 02/02/2013  | 0             | £1,750         | 20130202       | 2013     | 2         | 2-Feb-2013 |
| 02/03/2013  | 0             | £990           | 20130302       | 2013     | 3         | 2-Mar-2013 |
| 02/04/2013  | 0             | £500           | 20130402       | 2013     | 4         | 2-Apr-2013 |
| 02/05/2013  | 950           | £1,750         | 20130502       | 2013     | 5         | 2-May-2013 |

**Figure 9-2.** Applying the *FORMAT()* function

**Note** To reformat a date like this the column that you are using for the original data must either be of the date data type, or be capable of being interpreted as a date by Power BI Desktop.

The date format that you applied in this example was not predefined by DAX in any way. In fact, it was assembled from a set of day, month, and year codes that you can combine to create the date format that you want. Table 9-2 explains the codes that are available.

**Table 9-2.** Custom Date Formats

| Format Code | Description                                                | Example                                         |
|-------------|------------------------------------------------------------|-------------------------------------------------|
| d           | The day of the month                                       | "d MMM yyyy" produces 2 Jan 2016                |
| dd          | The day of the month with a leading zero when necessary    | "dd MMM yyyy" produces 02 Jan 2016              |
| ddd         | The three letter abbreviation for the day of the week      | "ddd d MMM yyyy" produces Sat 2 Jan 2016        |
| dddd        | The day of the week in full                                | "ddd dd MMM yyyy" produces Saturday 02 Jan 2016 |
| M           | The number of the month                                    | "dd M yyyy" produces 02 1 2016                  |
| MM          | The number of the month with a leading zero when necessary | "dd MM yyyy" produces 02 01 2016                |
| MMM         | The three letter abbreviation for the month                | "dd MMM yyyy" produces 02 Jan 2016              |
| MMMM        | The full month                                             | "dd MMMM yyyy" produces 02 January 2016         |
| yy          | The year as two digits                                     | "d MMM yy" produces 2 Jan 16                    |
| yyyy        | The full year                                              | "MMMM yyyy" produces January 2016               |

If you do not want to build your own date formats, then you can choose from the four predefined date and time formats that Power BI desktop has available. These are explained in Table 9-3.

**Table 9-3.** Predefined Date Formats

| Format Code | Description                                     | Example                             | Comments                                                                              |
|-------------|-------------------------------------------------|-------------------------------------|---------------------------------------------------------------------------------------|
| Short Date  | The short date as defined in the PC's settings. | FORMAT([InvoiceDate], "Short Date") | Formats the date using figures only.                                                  |
| Long Date   | The long date as defined in the PC's settings.  | FORMAT([InvoiceDate], "Long Date")  | Formats the date with the month as a text and the day of the week.                    |
| Long time   | The long time as defined in the PC's settings.  | FORMAT([InvoiceDate], "Long Time")  | Formats the datetime or time column with the hour and minutes of the day.             |
| Short time  | The short time as defined in the PC's settings. | FORMAT([InvoiceDate], "Long Date")  | Formats the datetime or time column with the hour and minutes and seconds of the day. |

**Note** As I explained in Chapter 7, the FORMAT() function converts a field to a text, so you need to be aware that this has the potential to restrict its use if further calculations are applied to the result produced by this formula. In my opinion, it is essentially useful when applied to modify the way that dates are displayed.

## Calculating the Age of Cars Sold

To continue with elementary DAX formulas—and as an admittedly extremely simple example—I will presume that we need to calculate the age of every car sold relative to the current date. As the source data contains the registration date for each vehicle, this will not be difficult. So, you have to do the following:

- With the Stock table selected, activate the Modeling ribbon and click the New Column button.
- Enter a left parenthesis to the right of the equals sign.
- Type **NOW()**. Make sure that you add the left and right parentheses even if these remain empty.
- Enter a minus sign then enter (or select) **[Registration\_Date]**.
- Enter a right parenthesis. This corresponds to the left parenthesis before the **NOW()** function.
- Enter a forward slash (the division symbol).
- Enter **365**.
- Replace Column with **VehicleAgeInYears**. The formula should look like this

`VehicleAgeInYears=(NOW()-[Registration_Date])/365`

- Press Enter or click the tick box in the formula bar. The formula will be added to the entire new column. Figure 9-3 shows you a small sample of the result of this operation.

| VehicleAgeInYears = (NOW()-[Registration_Date])/365 |           |            |           |                   |         |                           |              |                   |
|-----------------------------------------------------|-----------|------------|-----------|-------------------|---------|---------------------------|--------------|-------------------|
| VehicleType                                         | CostPrice | SpareParts | LaborCost | Registration_Date | Mileage | Vehicle                   | Direct Costs | VehicleAgeInYears |
| 1 Saloon                                            | 62000     | 400        | £951.00   | 01 January 1999   | 52500   | Rolls Royce Silver Seraph | 60649        | 17                |
| 8 Saloon                                            | 62000     | 400        | £750.00   | 08 January 1985   | 52500   | Rolls Royce Silver Shadow | 60850        | 31                |
| 6 Coupe                                             | 75890     | 400        | £147.00   | 08 September 2006 | 52500   | Aston Martin DB4          | 75343        | 9                 |
| 2 Coupe                                             | 37500     | 400        | £325.00   | 20 September 2006 | 52500   | TVR Tuscan                | 36775        | 9                 |
| 3 Saloon                                            | 37500     | 400        | £325.00   | 09 May 2007       | 52500   | Jaguar XK                 | 36775        | 8                 |
| 4 Coupe                                             | 37500     | 400        | £325.00   | 20 September 2006 | 52500   | TVR Cerbera               | 36775        | 9                 |
| 5 Convertible                                       | 37500     | 400        | £325.00   | 09 May 2007       | 52500   | Jaguar XK                 | 36775        | 8                 |

**Figure 9-3.** Calculated output using the *NOW()* function

Let me be clear, this is not the only way to calculate a time difference using DAX. It is probably not even the best one. It is a simple yet comprehensible introduction to a DAX function and it reminds you just how close a cousin DAX is to the Excel formula language. It also shows you an easy way to see exactly what DAX functions are available and what they do, since each function displays a brief explanation when you hover the mouse pointer over it.

## Calculating the Difference Between Two Dates

As you would probably expect, DAX can do more than just specify a number of days. As befits a formula language that is designed to aid in business analysis, it can deduce the time between two dates expressed as the following:

- Years
- Months
- Weeks
- Days
- Hours
- Minutes
- Seconds

This can be extremely useful when you want to classify records according to a duration, and can be calculated using the `DATEDIFF()` function. The `DATEDIFF()` function expects you to apply three parameters when calculating an interval. These are:

- The start date for the calculation of the interval
- The end date up until when the interval will be calculated
- The interval to calculate. This could be in years, days, or minutes, for example.

As an example, imagine that you want to display the number of weeks that each vehicle remained in stock, as this will help you to determine the fastest-selling models and consequently optimize the company's cash flow. As the data contains both the purchase date for each vehicle as well as its sale date (even if the two are in different tables) this can be done using the DAX `DATEDIFF()` function.

1. In the PowerBIDataModelExtendedWithMetrics.Pbix file, click the Data View icon and then click the Stock table in the Fields list.
2. Click the New Column button in the Modeling ribbon. A new column named Column will appear at the right of any existing columns.
3. To the right of the equals sign, enter **DATEDIFF(**. You will see that as you enter the first few characters, the list of functions will list all available functions beginning with these characters. You can click the function name to have it appear in the formula bar.
4. Press the [ key. The list of available fields from the current table will appear.
5. Select the [PurchaseDate] field.
6. Enter a comma.
7. Type **RELATED(**. The pop-up will display all the fields that can be linked via the data model to this table.
8. Scroll down the list and select `Invoices[invoiceDate]`.
9. Add a right parenthesis to close the `RELATED()` function.

10. Enter a comma. A pop-up list of available intervals will appear.

11. Select WEEK.

12. Add a final right parenthesis. The formula will look like this:

`Weeks In Stock = DATEDIFF([PurchaseDate],RELATED(Invoice[InvoiceDate]),WEEK)`

13. Press Enter or click the tick box in the formula bar. The formula will be added to the entire new column. You can now rename it **Weeks In Stock**. Figure 9-4 shows you a small sample of the result of this operation.

| Weeks In Stock = DATEDIFF([PurchaseDate],RELATED(Invoice[InvoiceDate]),WEEK) |                      |             |               |                      |                           |                 |              |                |
|------------------------------------------------------------------------------|----------------------|-------------|---------------|----------------------|---------------------------|-----------------|--------------|----------------|
| eAgeInYears                                                                  | Excessive Parts Cost | Price Check | Mileage Range | Vehicle Age Category | Vehicle Age Category Sort | Special Sales   | PurchaseDate | Weeks In Stock |
| 17                                                                           | Price OK             | Medium      | 17-20         | 4                    | Normal                    | 01 January 2012 | 74           |                |
| 31                                                                           | Price OK             | Medium      | >30           | 7                    | Normal                    | 01 January 2012 | 82           |                |
| 9                                                                            | Price OK             | Medium      | 7-10          | 2                    | Normal                    | 01 January 2012 | 65           |                |
| 9                                                                            | Price OK             | Medium      | 7-10          | 2                    | Special                   | 01 January 2012 | 56           |                |
| 8                                                                            | Price OK             | Medium      | 7-10          | 2                    | Normal                    | 01 January 2012 | 60           |                |
| 9                                                                            | Price OK             | Medium      | 7-10          | 2                    | Normal                    | 01 January 2012 | 65           |                |
| 8                                                                            | Price OK             | Medium      | 7-10          | 2                    | Normal                    | 01 January 2012 | 69           |                |
| 9                                                                            | Price OK             | Medium      | 7-10          | 2                    | Normal                    | 01 January 2012 | 74           |                |
| 9                                                                            | Price OK             | Medium      | 7-10          | 2                    | Normal                    | 01 January 2012 | 78           |                |
| 9                                                                            | Price OK             | Medium      | 7-10          | 2                    | Normal                    | 01 January 2012 | 82           |                |
| 8                                                                            | Price OK             | Medium      | 7-10          | 2                    | Special                   | 01 January 2012 | 91           |                |
| 9                                                                            | Price OK             | Medium      | 7-10          | 2                    | Special                   | 01 January 2012 | 95           |                |
| 10                                                                           | Price OK             | Medium      | >30           | 7                    | Normal                    | 01 January 2012 | 65           |                |

**Figure 9-4.** The results of a DATEDIFF() function

You can now see the number of weeks that each vehicle was in stock before being sold and use this figure in Power BI Desktop dashboards. Only *complete* intervals are displayed. In other words, if you have selected YEAR as the interval then the difference between the two dates must be fractionally more than one year for the function to return 1. Notice that you used the RELATED() function again to compare elements from separate tables. Once again, Power BI Desktop only listed the tables that were correctly linked to the destination table when you applied this function.

---

**Note** When you use the DATEDIFF() function, you must always add the lower date (the Purchase date in this example) as the first date that the function uses. If you do not, you will get an error message.

---

As you saw in the pop-up in step 10, the DATEDIFF() function lets you choose from a range of available intervals. These are explained in Table 9-4.

**Table 9-4.** Date Difference Intervals

| Function | Description                                       |
|----------|---------------------------------------------------|
| YEAR     | Returns the time difference in complete years.    |
| QUARTER  | Returns the time difference in complete quarters. |
| MONTH    | Returns the time difference in complete months.   |
| WEEK     | Returns the time difference in complete weeks.    |
| DAY      | Returns the time difference in complete days.     |
| HOUR     | Returns the time difference in complete hours.    |
| MINUTE   | Returns the time difference in complete minutes.  |
| SECOND   | Returns the time difference in complete seconds.  |

---

**Note** DAX cannot currently handle negative date differences, so you need to ensure that your data has been correctly prepared in Power BI Desktop Query *before* you calculate date and time differences.

---

## Adding Time Intelligence to a Data Model

I want now to explain the vital set of functions that concern time, or rather, the dates used in analyzing data. Power BI Desktop calls this time intelligence (even though it nearly always refers to is the use of date ranges). Applying this kind of temporal analysis can be a fundamental aspect of data presentation in business intelligence. After all, what enterprise does not need to know how this year's figures compare to last year's and what kind of progress is being made?

Time intelligence always requires a valid date table, which is one of the reasons why we will now spend a certain amount of time creating this core pillar of a successful data model. Then the date table has to be joined to the table containing the data that you want to compare over time on a date field. The good news is that once you have a valid date table, and have acquainted yourself with a handful of data and time functions in DAX, you can deliver some extremely impressive results. These kinds of calculations can cover (among other things) the following:

- YearToDate, QuarterToDate, and MonthToDate calculations
- Comparisons with previous years, quarters, or months
- Rolling aggregations over a period of time, such as the sum for the last three months
- Comparison with a parallel period in time, such as the same month in the previous year

An introduction to time intelligence in Power BI Desktop gives you a taste of some of the DAX functions that you are likely to use when analyzing data over time. To begin with, you will learn how to create a date table. Then you look at some of the different types of calculations that you can create to analyze data over time.

# Creating and Applying a Date Table

For time intelligence to work, you need a table that contains an *uninterrupted* range of dates that begins at least at the *earliest* date in your data, and that ends with a date at least equal to the *final* date in your data. In practice, this will nearly always mean creating a date table that begins on the 1st of January of the earliest date in your data and that ends on the 31st of December of the last year for which you have data. Once you have your date table, you can join it to one of the tables in your data model and then begin to exploit all the time-related analytical functions of Power BI Desktop.

The good news is that you can use Power BI Desktop itself to create a date table. It is also possible to import a contiguous range of dates from other applications such as Excel. Because I prefer you to come to appreciate the breadth and depth of DAX, I will explain how to implement a data table directly in Power BI Desktop.

## Creating the Date Table

The first requirement before starting to apply time intelligence is to have a valid date table. The following few steps explain one way to create a date table using and extending your newly acquired DAX skills.

1. In Data View, activate the Modeling ribbon and click the New Table button. The expression `Table =` will appear in the formula bar.
2. Replace the word `Table` with **DateDimension**.
3. Click to the right of the equals sign and enter the **CALENDAR()**.
4. Enter “**1/1/2012**”. This is the starting date for a table of dates. I am assuming here that your computer is configured for the UK or European date format. If this is not the case, then enter the date as you would normally using your local date format
5. Enter a comma.
6. Enter “**31/12/2016**” (or the equivalent date format that represents the 31st of December 2016 in your local date format). This is the end date for a table of dates.
7. Enter a right parenthesis. This corresponds to the left parenthesis of the `CALENDAR()` function. The formula bar will display this:

```
DateDimension = CALENDAR("1/1/2012", "31/12/2016").
```

8. Press Enter or click the tick icon in the formula bar. Power BI Desktop will create a table containing a single column of dates from the 1st of January 2012 until the 31st of December 2016.
9. In the Fields list, right-click the Date field in the DateDimension table and select Rename. Rename the Date field to **DateKey**. The date table will look like Figure 9-5.

| DateKey    |
|------------|
| 01/01/2012 |
| 02/01/2012 |
| 03/01/2012 |
| 04/01/2012 |
| 05/01/2012 |
| 06/01/2012 |
| 07/01/2012 |
| 08/01/2012 |
| 09/01/2012 |
| 10/01/2012 |
| 11/01/2012 |
| 12/01/2012 |
| 13/01/2012 |
| 14/01/2012 |
| 15/01/2012 |

**Figure 9-5.** An initial date table for a time dimension

10. Add 20 new columns containing the formulas explained in Table 9-5. Because Chapter 7 provided an exhaustive explanation covering the techniques that you need to apply when you want to add new columns, I will not repeat the process in detail here.

**Table 9-5.** DAX Formulas to Extend a Date Table

| Column Title    | Formula                          | Comments                                                                                                   |
|-----------------|----------------------------------|------------------------------------------------------------------------------------------------------------|
| FullYear        | YEAR([DateKey])                  | Isolates the year as a four-digit number.                                                                  |
| ShortYear       | VALUE(Right(Year([DateKey]),2))  | Isolates the year as a two-digit number.                                                                   |
| MonthNumber     | MONTH([DateKey])                 | Isolates the number of the month in the year as one or two digits.                                         |
| MonthNumberFull | FORMAT([DateKey], "MM")          | Isolates the number of the month in the year as two digits, with a leading zero for the first nine months. |
| MonthFull       | FORMAT([DateKey], "MMMM")        | Displays the full name of the month.                                                                       |
| MonthAbbr       | FORMAT([DateKey], "MMM")         | Displays the name of the month as a three-letter abbreviation.                                             |
| WeekNumber      | WEEKNUM([DateKey])               | Shows the number of the week in the year .                                                                 |
| WeekNumberFull  | FORMAT(Weeknum([DateKey]), "00") | Shows the number of the week in the year with a leading zero for the first nine weeks.                     |
| DayOfMonth      | DAY([DateKey])                   | Displays the number of the day of the month.                                                               |

(continued)

**Table 9-5.** (continued)

| Column Title         | Formula                                                            | Comments                                                                                                                |
|----------------------|--------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| DayOfMonthFull       | FORMAT(Day([DateKey]), "00")                                       | Displays the number of the day of the month with a leading zero for the first nine days.                                |
| DayOfWeek            | WEEKDAY([DateKey])                                                 | Displays the number of the day of the week.                                                                             |
| DayOfWeekFull        | FORMAT([DateKey], "dddd")                                          | Displays the name of the weekday.                                                                                       |
| DayOfWeekAbbr        | FORMAT([DateKey], "ddd")                                           | Displays the name of the weekday as a three-letter abbreviation.                                                        |
| ISODate              | [FullYear] & [MonthNumberFull]<br>& [DayOfMonthFull]               | Displays the date in the ISO (internationally recognized) format of YYYYMMDD.                                           |
| FullDate             | [DayOfMonth] & " " &<br>[MonthFull] & " " & [FullYear]             | Displays the full date with spaces.                                                                                     |
| QuarterFull          | "Quarter " & ROUNDDOWN(MONTH<br>([DateKey])/4,0)+1                 | Displays the current quarter.                                                                                           |
| QuarterAbbr          | "Qtr " & ROUNDDOWN(MONTH<br>([DateKey])/4,0)+1                     | Displays the current quarter as a three-letter abbreviation plus the quarter number.                                    |
| Quarter              | "Q" & ROUNDDOWN(MONTH(<br>[DateKey])/4,0)+1                        | Displays the current quarter in short form.                                                                             |
| QuarterNumber        | ROUNDDOWN(MONTH([DateKey])/<br>4,0)+1                              | Displays the number of the current quarter. This is essentially used as a sort by column.                               |
| QuarterAndYear       | DateDimension[Quarter and Year]                                    | Shows the quarter and the year.                                                                                         |
| MonthAndYearAbbr     | DateDimension[MonthAbbr] & " "<br>& [FullYear]                     | Shows the abbreviated month and year.                                                                                   |
| QuarterAndYearNumber | [FullYear] & [QuarterNumber]                                       | Shows the year and quarter numbers. This is essentially used as a sort by column.                                       |
| YearAndWeek          | VALUE([FullYear]<br>&[WeekNumberFull])                             | Indicates the year and week. The VALUE() function ensures that the figure is considered as numeric by Power BI Desktop. |
| YearAndMonthNumber   | VALUE(DateDimension[FullYear] &<br>DateDimension[MonthNumberFull]) | A numeric value for the year and month.                                                                                 |

The first few columns of the DateDimension table should now look like Figure 9-6.

| DateKey    | FullYear | ShortYear | MonthNumber | MonthFull | MonthAbbr | MonthNumberFull | WeekNumber | WeekNumberFull | DayOfMonth |
|------------|----------|-----------|-------------|-----------|-----------|-----------------|------------|----------------|------------|
| 01/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 1 01           | 1          |
| 02/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 1 01           | 2          |
| 03/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 1 01           | 3          |
| 04/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 1 01           | 4          |
| 05/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 1 01           | 5          |
| 06/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 1 01           | 6          |
| 07/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 1 01           | 7          |
| 08/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 2 02           | 8          |
| 09/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 2 02           | 9          |
| 10/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 2 02           | 10         |
| 11/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 2 02           | 11         |
| 12/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 2 02           | 12         |
| 13/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 2 02           | 13         |
| 14/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 2 02           | 14         |
| 15/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 3 03           | 15         |
| 16/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 3 03           | 16         |
| 17/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 3 03           | 17         |
| 18/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 3 03           | 18         |
| 19/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 3 03           | 19         |
| 20/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 3 03           | 20         |
| 21/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 3 03           | 21         |
| 22/01/2012 | 2012     | 12        | 1           | January   | Jan       | 01              |            | 4 04           | 22         |

**Figure 9-6.** The completed Date table

The point behind creating all these ways of expressing dates and parts of dates is that you can now use them in your tables, charts, and gauges to aggregate and display data over time. Any record that has a date element can now be expressed visually; not just as the date itself, but shown as and aggregated as years, quarters, months, or weeks. The trick is to prepare all the time groupings that you are likely to need in the date table of your dashboards. However, you do not need to worry if you find yourself needing an extra column or two further down the line, because you can always add columns that contain other date elements.

## Adding Sort By Columns to the Date Table

In Chapter 7, you saw how to sort a column using the data in another column to provide the sort order. This technique is essential when dealing with date tables, as you want to be sure that any visualizations that contain date elements appear in the right order. The classic example is months. As things stand, if you were to use the MonthFull or MonthAbbr columns in a chart or table then you would see the month names appearing on an axis or in a column in alphabetical order.

To avoid this, you have to add one final tweak to the date table and apply a Sort By column to certain of the date elements in other columns. Since you saw how this is done in Chapter 7, now I will only provide the list of columns that need this extra tweak, rather than reiterating all the details. Table 9-6 gives you the required information to extend the data table so that all date elements are sorted correctly.

**Table 9-6.** The Sort By Columns Needed for the Date Table

| Column           | Sort By Column       |
|------------------|----------------------|
| MonthFull        | MonthNumber          |
| MonthAbbr        | MonthNumber          |
| DayOfWeekFull    | DayOfWeek            |
| DayOfWeekAbbr    | DayOfWeek            |
| Quarter And Year | QuarterAndYearNumber |
| FullDate         | DateKey              |
| MonthAndYearAbbr | YearAndMonthNumber   |
| MonthAndYear     | YearAndMonthNumber   |

## Date Table Techniques

When using date tables to invoke time intelligence in DAX there are two fundamental principles that must always be applied. I realize that I mention them elsewhere, but they are so essential that they bear repetition.

- The date range must be *continuous*; that is, there must not be any dates missing in the column that contains the list of calendar days in the table of dates.
- The date range must encompass *all the dates* that you are using in other tables in the data model.

The first requirement is covered by the use of the CALENDAR() function to create a date table. The second can require that you discover the lower and upper date thresholds in one or more tables of dates. Since this can be a little laborious (not to mention error-prone), it is probably easier to get DAX to find these dates for you and apply them to the CALENDAR() function.

Consequently, once you know where the lowest and highest dates are in your data model (even if they are in separate tables), you can use these dates as the lower and upper boundaries of the CALENDAR() function. In the case of the Brilliant British Cars, data the formula could read as follows:

```
DateDimension = CALENDAR(MIN('Stock'[PurchaseDate]), MAX('Invoices'[InvoiceDate]))
```

This formula shows that you can apply two functions that you have seen already—MIN() and MAX()—with date and datetime data types. They simply tell DAX to find the earliest and latest dates in a column.

Alternatively, and as a nod to best practice, you may prefer to ensure that the date dimension always covers *entire years* of dates. In this case, the formula to create the table would be as follows:

```
DateDimension = CALENDAR(STARTOFYEAR(MIN('Stock'[PurchaseDate])), ENDOFYEAR(MAX('Invoices'[InvoiceDate])))
```

This formula extends the DAX calculation by using two functions that were explained in Table 9-1.

- STARTOFYEAR(): Deduces the first day of the year.
- ENDOFYEAR(): Deduces the last day of the year.

This formula also shows you that you can define a date range using different columns, or even different tables when you are specifying a date range. This is particularly useful when defining a date table.

**Note** The principal advantage of looking up the date boundaries like this is that they update automatically as the source data changes. So if the Stock or Invoices tables are extended in the data source to contain further rows with dates outside the existing date range, then the DateDimension table will also grow to encompass these new dates.

---

Creating a data table can be fun, but nonetheless takes a few minutes. So here's a tip that I can give you: create a Power BI Desktop file that contains nothing but a date dimension table (using a manually defined start date and end date, just as you saw at the beginning of this example) with all the other columns added. You can then make copies of this "template" file and use them as the basis for any new data models that you create. This can include replacing the fixed threshold dates with references to the data in tables, as mentioned previously.

## Adding the Date Table to the Data Model

Now that you have a date table, you can integrate it with your data model so that you can start to apply some of the time intelligence that DAX makes possible.

1. Click the Relationships View icon to display the tables in the data model.
2. In the Home ribbon, click the Manage Relationships button. The Manage Relationships dialog will appear.
3. Click the New button. The Create Relationship dialog will appear.
4. At the top of the dialog, select the DateDimension table from the pop-up list.
5. Once the sample data from the DateDimension table is displayed, click inside the DateKey column to select it.
6. Under the DateDimension table sample data, select the Invoices table from the pop-up list.
7. Once the sample data from the Invoices table is displayed, click inside the InvoiceDate column to select it. The dialog will look like Figure 9-7.

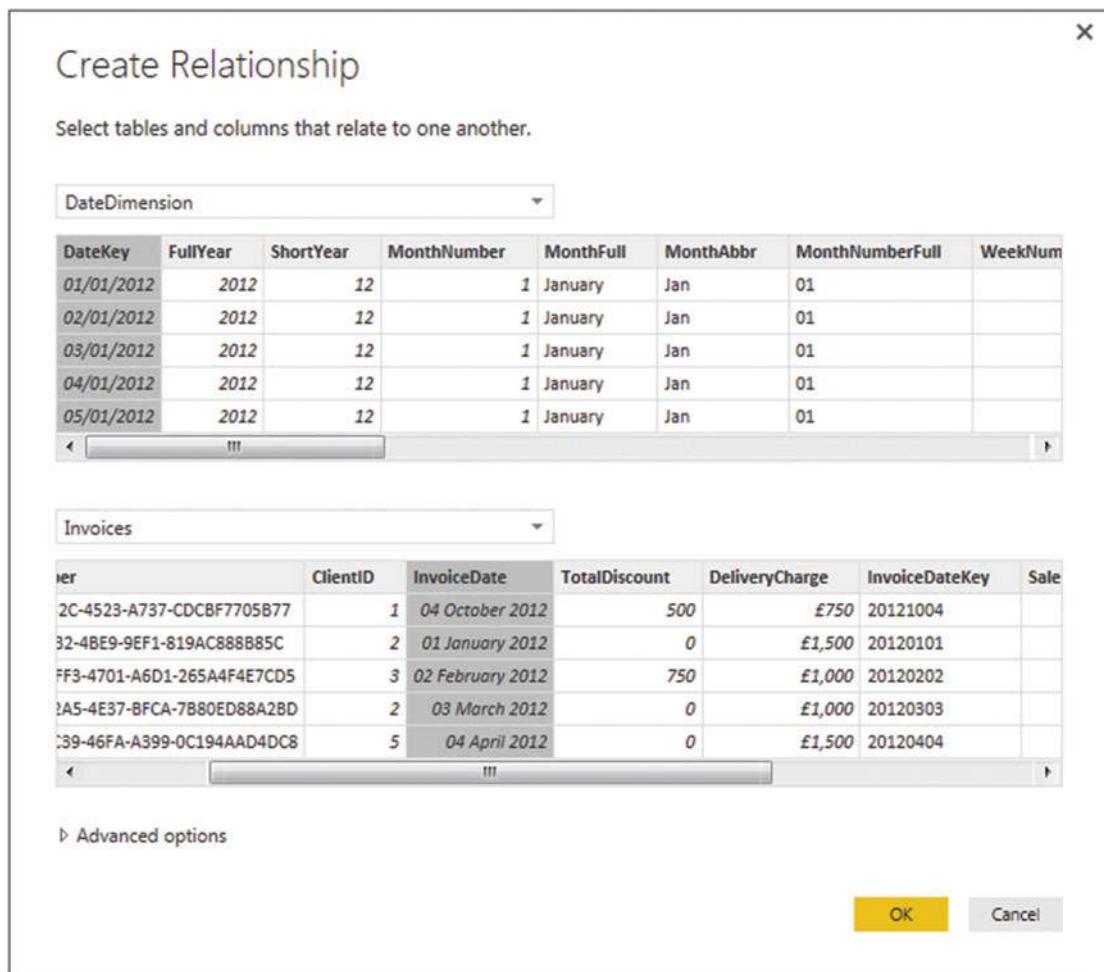


Figure 9-7. Adding a date dimension to the data model

8. Click OK. The relationship will appear in the Manage Relationships dialog.
9. Click Close. The DateDimension table will appear in the data model joined to the Invoices table.

**Note** For time intelligence in Power BI Desktop to work correctly, the fields used to join a date table and a data table must both be set to the *date* or *datetime* data type.

# Applying Time Intelligence

Now that all the preparations have been completed, it is (finally) time to see just how DAX can make your life easier when it comes to calculating metrics over time. This is possible only because you have a date table in place and it is connected to the requisite date field in the table that contains the data you want to aggregate. However, with the foundations in place, you can now start to deliver some really interesting and persuasive output.

## YearToDate, QuarterToDate, and MonthToDate Calculations

To begin, let's resolve a simple but frequent requirement: calculating month-to-date, quarter-to-date, and year-to-date sales figures.

The three functions that you will see in this example are extremely similar. Consequently, I will only explain the first one (a month-to-date calculation), and then will let you create the next two in a couple of copy, paste, and tweak operations.

1. In the Data View, select the Invoices table and click New Measure.
2. In the formula bar, replace Measure with **MonthSales**.
3. To the right of the equals sign, enter (or type and select) **TOTALMTD()**. This will apply the month-to-date aggregation for a field.
4. Enter (or type and select) **SUM()**. This specifies the actual aggregation that you want to apply.
5. Select the **InvoiceLines[SalePrice]** field from the pop-up list of available fields.
6. Enter a right parenthesis to end the **SUM()** function.
7. Enter a comma.
8. Type the first few characters of the date table (**DateDimension** in this example) and select the date key field (**DateKey** in this example).
9. Enter a right parenthesis to end the **TOTALMTD()** function. The formula should read as follows:

```
MonthSales = TOTALMTD(SUM(InvoiceLines[SalePrice]),DateDimension[DateKey])
```

10. Press Enter or click the tick mark icon to complete the measure definition.

Now copy the formula that you just created and use it as the basis for two new measures. These will be quarterly sales to date and annual sales to date, as follows:

```
QuarterSales = TOTALQTD(SUM(InvoiceLines[SalePrice]),DateDimension[DateKey])
YearSales = TOTALYTD(SUM(InvoiceLines[SalePrice]),DateDimension[DateKey])
```

The three formulas that you have used are TOTALMTD() for the month-to-date aggregation, TOTALQTD() for the quarter-to-date aggregation, and TOTALYTD() for the year-to-date aggregation. All three functions take the following two parameters:

- The *aggregate function*. Depends on the actual metric that you want to deliver and the table and column that is aggregated. (SUM() was used here, although it could have been AVERAGE(), or COUNT(), or any other of the aggregate functions that you saw in Chapter 7.)
- The *key field of the date table*. Since the Invoices table is linked to the date table in the data model using the InvoiceDate field, DAX can apply the correct calculation if—and only if—you have added a date table to the data model and then specified the key field of the date table.

Since you have created these measures, I imagine that you would like to see them in action. Figure 9-8 shows the quarter-to-date and year-to-date sales for 2014 (along with the aggregated sales from the initial SalePrice column in the InvoiceLines table) in a simple Power BI Desktop table like the one that you created in Chapter 1 (you will see a lot more of in the next chapter). To obtain this result, you have to drag the FullYear field from the DateDimension table onto the Page Level Filters area and then select only the check box for 2014. This restricts the dates to the year 2014. I realize that I am anticipating the content of Chapter 15 here; however, filtering at this level is so intuitive that it is worth learning now.

| 2014 Sales Trends |            |              |            |
|-------------------|------------|--------------|------------|
| MonthFull         | SalePrice  | QuarterSales | YearSales  |
| January           | £555,500   | £555,500     | £555,500   |
| February          | £359,500   | £915,000     | £915,000   |
| March             | £372,500   | £1,287,500   | £1,287,500 |
| April             | £367,500   | £367,500     | £1,655,000 |
| May               | £529,500   | £897,000     | £2,184,500 |
| June              | £544,500   | £1,441,500   | £2,729,000 |
| July              | £565,500   | £565,500     | £3,294,500 |
| August            | £584,500   | £1,150,000   | £3,879,000 |
| September         | £606,500   | £1,756,500   | £4,485,500 |
| October           | £554,940   | £554,940     | £5,040,440 |
| November          | £657,000   | £1,211,940   | £5,697,440 |
| December          | £689,000   | £1,900,940   | £6,386,440 |
| Total             | £6,386,440 | £1,900,940   | £6,386,440 |

Figure 9-8. The quarter- and year-to-date functions in DAX

As you can see, you have each month's sales figures along with the cumulative sales for each quarter to date (and restarting each quarter). The final column shows you the yearly sales total for each month to date. Also, note that the months appear in calendar order, because the MonthFull field has used the MonthNumber field as its Sort By column.

Conceptually, the three functions that are outlined—TOTALMTD(), TOTALQTD(), and TOTALYTD()—can be considered as CALCULATE() functions that have been extended to deliver a specific result for a time frame. You can always calculate aggregations for a period to date using the CALCULATE() function if you want to. However, since this “shorthand” version is so practical and easy to use, I see no reason to try anything more complicated when there is no real need.

**Note** In this example, I suggest starting the process of adding a new measure with the Invoices table selected, merely because this table seems a good place to store the metric. You can create the metric in virtually any table in practice—provided that you have a coherent data model to build on.

## Analyze Data As a Ratio over Time

Looking at how data aggregates over time is only one of the ways that time intelligence can enable you to deliver time-based analysis. On occasion, you may well want to see how a day’s sales relate to the total sales for a period. So in this example, we will calculate the daily percentage of sales for Brilliant British Cars relative to the total sales for the year 2014.

1. In the Data View, select the InvoiceLines table and click New Measure.
2. Replace Measure with **PercentOfYear**.
3. To the right of the equals sign, enter (or type and select) **SUM()**.
4. Select the [SalePrice] field from the pop-up list of available fields.
5. Enter a forward slash to indicate division.
6. Enter (or type and select) **CALCULATE(SUM()**. This tells DAX that you want a filtered calculation and that the specific aggregation that you want to apply is a **SUM()** function.
7. Select the [SalePrice] field from the pop-up list of available fields.
8. Enter a right parenthesis to end the **SUM()** function.
9. Enter a comma. This tells the calculate function that you have chosen the aggregation that you want and will now apply a filter.
10. Enter (or type and select) **DATESBETWEEN()**.
11. Start typing the name of the date table (**DateDimension**) and then select the DateKey field from the pop-up. This tells the **DATESBETWEEN()** function which field in the time dimension should be used as its first parameter.
12. Add a comma. This ends the first parameter for the **DATESBETWEEN()** function.
13. Enter (or type and select) **STARTOFTYEAR()**.
14. Start typing the name of the date table (**DateDimension**) and then select the DateKey field from the pop-up. This will be the lower boundary of the time span that will be used to filter the **CALCULATE()** function.
15. Add a right parenthesis to close the **STARTOFTYEAR()** function.
16. Add a comma. This ends the second parameter for the **DATESBETWEEN()** function.

17. Enter (or type and select) **ENDOFYEAR()**.
18. Once again, start typing the name of the date table (**DateDimension**) and then select the DateKey field from the pop-up. This will be the upper boundary of the time span that will be used to filter the **CALCULATE()** function.
19. Add a right parenthesis to close the **ENDOFYEAR()** function.
20. Add a comma. This ends the third and final parameter for the **DATESBETWEEN()** function.
21. Add three right parentheses. The first one ends the **ENDOFYEAR()** function, the second ends the **DATESBETWEEN()** function, and the final parenthesis closes the **CALCULATE()** function. The formula should look like this:

```
PercentOfYear = sum([SalePrice]) / CALCULATE(SUM([SalePrice]),DATESBETWEEN(DateDimension[DateKey],STARTOFTYEAR(DateDimension[DateKey]),ENDOFYEAR(DateDimension[DateKey])))
```

22. Press Enter or click the tick mark icon to complete the measure definition.
23. In the Modeling ribbon, click the percentage icon to format this measure as a percentage.

This formula was a little more complex than those that you have seen so far in this chapter. So let me explain it in a bit more detail. At its heart, this formula consists of two main elements:

- The sales total for the time element that will be used in a visualization. This could be the day, week, month, or quarter, for instance.
- The total sales for the entire year that will be used in a visualization. This has to be calculated independently of the actual date element that will be displayed. Consequently, the **CALCULATE()** function is used to extend the aggregation—the **SUM()** in this case—to the whole year. This is done by setting a range of dates as the filter for the aggregation. The date range is set using the **DATESBETWEEN()** function. This requires a lower threshold (defined by the **STARTOFTYEAR()** function) and a higher threshold (defined by the **ENDOFYEAR()** function). This way, the date range runs from the first to the last days of the year.

Finally, the calculation divides the specified time span's total by the total for the year; the percentage for that sales period is displayed.

So that you can see the outcome of your formula, you could create a table that contains the following three fields:

- DateKey
- SalePrice
- PercentOfYear

You should see something like Figure 9-9 if you have applied a page-level filter to restrict the FullYear field to 2014, as described in the previous example.

| DateKey             | SalePrice         | PercentOfYear   |
|---------------------|-------------------|-----------------|
| 01/01/2014 00:00:00 | £555,500          | 8.70 %          |
| 01/02/2014 00:00:00 | £178,000          | 2.79 %          |
| 02/02/2014 00:00:00 | £181,500          | 2.84 %          |
| 01/03/2014 00:00:00 | £189,000          | 2.96 %          |
| 03/03/2014 00:00:00 | £183,500          | 2.87 %          |
| 01/04/2014 00:00:00 | £178,000          | 2.79 %          |
| 04/04/2014 00:00:00 | £189,500          | 2.97 %          |
| 01/05/2014 00:00:00 | £319,000          | 4.99 %          |
| 04/05/2014 00:00:00 | £210,500          | 3.30 %          |
| 01/06/2014 00:00:00 | £319,000          | 4.99 %          |
| 04/06/2014 00:00:00 | £225,500          | 3.53 %          |
| 01/07/2014 00:00:00 | £310,000          | 4.85 %          |
| 04/07/2014 00:00:00 | £255,500          | 4.00 %          |
| 01/08/2014 00:00:00 | £319,000          | 4.99 %          |
| 04/08/2014 00:00:00 | £265,500          | 4.16 %          |
| 01/09/2014 00:00:00 | £451,000          | 7.06 %          |
| 04/09/2014 00:00:00 | £155,500          | 2.43 %          |
| 01/10/2014 00:00:00 | £319,000          | 4.99 %          |
| 04/10/2014 00:00:00 | £235,940          | 3.69 %          |
| 01/11/2014 00:00:00 | £319,000          | 4.99 %          |
| 04/11/2014 00:00:00 | £338,000          | 5.29 %          |
| 01/12/2014 00:00:00 | £178,000          | 2.79 %          |
| 04/12/2014 00:00:00 | £511,000          | 8.00 %          |
| <b>Total</b>        | <b>£6,386,440</b> | <b>100.00 %</b> |

**Figure 9-9.** Displaying the sales per day as a percentage of the yearly sales

You may be thinking that this was a lot of work just to get a percentage figure. Well, perhaps it is at first. Yet you can now capitalize on your effort and see how time intelligence is worth the effort. For instance, if you replace the DateKey field with the MonthFull field in the table shown in Figure 9-9, you instantly see the sales percentages for each month. The same applies if you replace MonthFull with Quarter. This is because you have created an extremely supple and fluid formula that can adapt to any time segment. The formula says to “Take the time segment (day, month, quarter, or year) and then find the date range for the whole year. Use this to calculate the total for the year, and then divide the figure for the time span by this total to display the percentage.”

## Comparing a Metric with the Result from a Range of Dates

Let’s push the time-based data analysis a little further and imagine that you need to look at how figures have evolved compared to a specific time interval in the past. By this I mean that you want to see how sales for the current month have fluctuated compared to, say, the previous month. The following formula shows you how to do just this. Once you understand the principles that this technique can be extended to, compare the data over many different time spans.

1. In the Data View, select the InvoiceLines table and click New Measure.
  2. Replace Measure with **PercentOfPreviousMonth**.
  3. To the right of the equals sign, type or select **SUM(**.
  4. Type a left bracket and select the [SalePrice] field from the pop-up list of available fields.
  5. Enter a right parenthesis to end the **SUM()** function.
  6. Enter a forward slash to indicate division.
  7. Enter (or type and select) **CALCULATE(SUM(**.
  8. Select the [SalePrice] field from the pop-up list of available fields.
  9. Enter a right parenthesis to end the **SUM()** function.
  10. Enter a comma. This tells the calculate function that you have chosen the aggregation that you want; it will now apply a filter.
  11. Enter (or type and select) **PREVIOUSMONTH(**.
  12. Start typing the name of the date table (**DateDimension**) and then select the DateKey field from the pop-up. This tells the **PREVIOUSMONTH()** function the time dimension field that should be used as its parameter.
  13. Enter a right parenthesis to end the **PREVIOUSMONTH()** function.
  14. Enter a right parenthesis to end the **CALCULATE()** function. The formula should look like this:
- ```
PercentOfPreviousMonth = SUM([SalePrice]) / CALCULATE(SUM([SalePrice]), PREVIOUSMONTH(DateDimension[DateKey]))
```
15. Press Enter or click the tick mark icon to complete the measure definition.

Figure 9-10 shows you the output that can be obtained by using the MonthFull field to give the month of a year (2014 in this example) alongside the **PREVIOUSMONTH()** function. To stress that time intelligence can adapt to different circumstances, you can also see the **PERCENTOFYEAR** function from the previous example—only it has automatically returned the percentage for the month this time.

MonthFull	SalePrice	PercentOfYear	PercentOfPreviousMonth
January	£555,500	8.70 %	110.60 %
February	£359,500	5.63 %	64.72 %
March	£372,500	5.83 %	103.62 %
April	£367,500	5.75 %	98.66 %
May	£529,500	8.29 %	144.08 %
June	£544,500	8.53 %	102.83 %
July	£565,500	8.85 %	103.86 %
August	£584,500	9.15 %	103.36 %
September	£606,500	9.50 %	103.76 %
October	£554,940	8.69 %	91.50 %
November	£657,000	10.29 %	118.39 %
December	£689,000	10.79 %	104.87 %

Figure 9-10. Comparing values with a previous period of time

Once again, the “magic” in the formula was the use of the `CALCULATE()` function. Yet again, this function required you to enter two elements:

- An aggregation to carry out (the sum of the Sale Price in this case).
- A filter to apply (the previous month in this example). Once again, the function uses the date key from the date table to apply the time intelligence.

With these two parameters in place, DAX was able to take the time element used in the visualization (the month) and say, “Give me the aggregation for the previous month.”

As you can probably imagine, DAX does not limit you to only comparing month-by-month data. Indeed, it can help you compare data for a wide range of time spans. Now that you understand the basic principles, it is probably easiest to appreciate the related DAX functions as they are shown in Table 9-7.

Table 9-7. DAX Date and Time Formulas to Return a Range of Dates

Formula	Description	Example
<code>PREVIOUSDAY()</code>	Finds data for the previous day.	<code>CALCULATE(SUM(InvoiceLines[SalePrice])), PREVIOUSDAY(DateDimension[DateKey]))</code>
<code>PREVIOUSMONTH()</code>	Finds data for the previous month.	<code>CALCULATE(SUM(InvoiceLines[SalePrice])), PREVIOUSMONTH(DateDimension[DateKey]))</code>
<code>PREVIOUSQUARTER()</code>	Finds data for the previous quarter.	<code>CALCULATE(SUM(InvoiceLines[SalePrice])), PREVIOUSQUARTER(DateDimension[DateKey]))</code>
<code>PREVIOUSYEAR()</code>	Finds data for the previous year.	<code>CALCULATE(SUM(InvoiceLines[SalePrice])), PREVIOUSYEAR(DateDimension[DateKey]))</code>
<code>NEXTDAY()</code>	Finds the date for the following day.	<code>CALCULATE(SUM(InvoiceLines[SalePrice])), NEXTDAY(DateDimension[DateKey]))</code>
<code>NEXTMONTH()</code>	Finds data for the following month.	<code>CALCULATE(SUM(InvoiceLines[SalePrice])), NEXTMONTH(DateDimension[DateKey]))</code>

(continued)

Table 9-7. (continued)

Formula	Description	Example
NEXTQUARTER()	Finds data for the following quarter.	CALCULATE(SUM(InvoiceLines[SalePrice]), NEXTQUARTER(DateDimension[DateKey]))
NEXTYEAR()	Finds data for the following year.	CALCULATE(SUM(InvoiceLines[SalePrice]), NEXTYEAR(DateDimension[DateKey]))
DATESMTD()	Finds data for the month to date.	CALCULATE(SUM(InvoiceLines[SalePrice]), DATESMTD(DateDimension[DateKey]))
DATESQTD()	Finds data for the quarter to date.	CALCULATE(SUM(InvoiceLines[SalePrice]), DATESQTD(DateDimension[DateKey]))
DATESYTD()	Finds data for the year to date.	CALCULATE(SUM(InvoiceLines[SalePrice]), DATESYTD(DateDimension[DateKey]))

Comparisons with Previous Time Periods

While the various DAX functions that return a “previous” time span are extremely useful, it could be argued that they are a little rigid for some types of calculation. After all, comparisons to the previous month typically require you to display data by month. So DAX has alternative methods of comparing data over time that do not require you to specify exactly which time element (day, month, quarter, or year) you want to compare with. Instead, you can merely say that you want to go back a defined period in time, and then depending on the choice of time element that you use in a visualization, DAX automatically calculates the correct figure for comparison.

Calculating Sales for the Previous Year

As a first example, let’s imagine that you want to compare current sales with sales for the current period—be it a day, week, month, quarter, or year. There might be several ways of doing this, but there is one fairly simple approach that returns the total car sales for the same time span for the previous year. Once the principle is clear, I will show you how to extend this to calculate the following:

- The average car sales price for the previous year
- The number of cars sold in the previous quarter

Initially, you should carry out the following steps to calculate sales for the previous year.

1. In the Data View, select the Invoices table and click New Measure.
2. Replace Measure with **SalesPreviousYear**.
3. To the right of the equals sign, enter (or type and select) **CALCULATE**.
4. Enter (or select) **SUM**.
5. Select the **InvoiceLines[SalePrice]** field.
6. Enter a right parenthesis to close the **SUM()** function.
7. Enter a comma.
8. Enter (or select) **DATEADD**.

9. Select the DateDimension[DateKey] field. This indicates to DAX the correct date table and date key field.
10. Enter a comma.
11. Enter **-1**. This will cause the time comparison to apply to a *previous* period of time.
12. Enter a comma.
13. Enter (or select) **YEAR**. This indicates to the DATEADD() function that it wants to compare figures with the previous year.
14. Enter two right parentheses. The first one closes the DATEADD() function; the second one closes the CALCULATE() function. The formula will look like this:

```
PreviousYearSales = CALCULATE(SUM(InvoiceLines[SalePrice]),
DATEADD(DateDimension[DateKey],-1,YEAR))
```

15. Press Enter or click the tick icon in the formula bar to finish the measure.

Figure 9-11 shows how the sale price (at the quarter level in this instance) is compared to last year's sale price when you add the PreviousYearSales measure to a table. In this example, the table shows only sales for 2014. Consequently, the previous year's sales are those for 2013. What is so impressive is that if you filter the table on another year (2015 for example), you will see that the previous year's figures are now those for 2014.

Quarter and Year	SalePrice	PreviousYearSales
Q1 2014	£1,287,500	£2,001,550
Q2 2014	£2,007,000	£2,412,090
Q3 2014	£2,402,940	£2,242,820
Q4 2014	£689,000	£502,250
Total	£6,386,440	£7,158,710

Figure 9-11. Calculating metrics for a previous year

The formula sums or averages the data in a column, but only for the previous year, compared to the date field for each row. (This is the InvoiceDate in the sample data, because the date field is linked to the date table in the sample data model). Note that you do not use the InvoiceDate field in these formulas. This is because it is the field that is linked to the DateKey field of the date table. Power BI Desktop knows which field to use in the Stock table as the basis for time comparisons. To labor the point, it was essential to create a coherent and complete data model to make time intelligence work perfectly.

In this example, you set the “time shift” to a negative value, so that DAX would go back one year in time. You can also use positive numbers if you are looking at data from a past viewpoint and want to compare with data from later dates.

Tip The DATEADD function lets you replace YEAR with MONTH or DAY if you need to compare with data from days or months previously.

Once you have mastered this technique you can extend and enhance a formula such as this to provide a multitude of metrics that will adapt to the time-based filters on tables and charts. As a second example, try creating the following measure. It will calculate the average sale price for the selected period(s) in the preceding year. I hope that by now you have become used to writing DAX formulas, so I will not explain how to create this formula, step by step. Instead, I will let you create it unaided. This should provide you with some good practice in creating DAX formulas by yourself.

```
AverageSalePricePreviousYear = CALCULATE(
    AVERAGE(InvoiceLines[SalePrice]),
    DATEADD(DateDimension[DateKey], -1, YEAR)
)
```

Note I have formatted the code for greater readability on the page and hopefully to make the logic of the functions more comprehensible. You might not be able to use the code formatted like this in Power BI Desktop without simplifying the presentation.

Alternatively, perhaps you want to see the number of sales for the previous quarter relative to the date that is used to filter a visualization. This formula would be as follows:

```
NumberOfSalesPreviousQtr = CALCULATE(
    COUNT(InvoiceLines[InvoiceID]),
    DATEADD(DateDimension[DateKey],
        -1, QUARTER)
)
```

Now that you have seen the principle, you are free to adapt it to your specific requirements. You can use any of the DAX aggregation functions that were described in the previous chapter. You can mix these with the four interval types (Year, Quarter, Month, and Day) that the DATEADD() function uses to deliver a truly wide-ranging set of time comparison metrics that automatically adapt to the time span of your Power BI Desktop visualization.

Comparison with a Parallel Period in Time

Looking at metrics from the past can be key indicator of how a business is progressing. Clearly identifying the extent (or lack of) progress is even more telling. There are several ways that you can perform these types of calculation in DAX. In this section, you will see a couple of techniques that you may find useful.

Comparing Data from Previous Years

So the following two measures (YearOnYearDelta and YearOnYearDeltaPercent) calculate the increase or decrease in sales compared to a previous year and also that change expressed as a percentage. These measures extend the logic of the last few formulas using functions that you have already met. I will presume that after two chapters on DAX, you do not really need a step-by-step explanation on how to enter a formula. So I will only present and then explain the code from now on. The following is the code to add YearOnYearDelta and YearOnYearDeltaPercent as new measures to the Invoices table.

```

YearOnYearDelta=IF(
    ISBLANK(
        SUM(InvoiceLines[SalePrice])
    ),
    BLANK(),
    IF(
        ISBLANK(
            CALCULATE(
                SUM(InvoiceLines[SalePrice]),
                DATEADD(DateDimension[DateKey],
                    -1, YEAR)
            )
        ),
        BLANK(),
        SUM(Stock[SalePrice])
        - CALCULATE(
            SUM(InvoiceLines[SalePrice]),
            DATEADD(DateDimension[DateKey], -1, YEAR)
        )
    )
)

YearOnYearDeltaPercent=IF(
    ISBLANK(
        SUM(InvoiceLines[SalePrice])
    ),
    BLANK(),
    IF(
        ISBLANK(
            CALCULATE(
                SUM(InvoiceLines[SalePrice]),
                DATEADD(DateDimension[DateKey],
                    -1, YEAR)
            )
        ),
        BLANK(),
        (
            SUM(InvoiceLines[SalePrice])
            - CALCULATE(
                SUM(InvoiceLines[SalePrice]),
                DATEADD(DateDimension[DateKey],
                    -1, YEAR)
            )
        )
        /CALCULATE(
            SUM(InvoiceLines[SalePrice]),
            DATEADD(DateDimension[DateKey],
                -1, YEAR)
        )
    )
)

```

These two formulas are a lot easier than they look, believe me.

The formula for YearOnYearDelta is this:

```
SUM(Stock[SalePrice])  
- CALCULATE(SUM(InvoiceLines[SalePrice]), DATEADD(DateDimension[DateKey], -1, YEAR))
```

All the code says is “Subtract last year’s sales from this year’s sales.” Everything else is wrapper code to prevent a calculation if either this year’s or last year’s data is zero.

Equally, this is the core code for the YearOnYearDeltaPercent formula:

```
SUM(InvoiceLines[SalePrice]) -  
CALCULATE(SUM(InvoiceLines[SalePrice]),DATEADD(DateDimension[DateKey], -1, YEAR)) /  
CALCULATE(SUM(InvoiceLines[SalePrice]),DATEADD(DateDimension[DateKey], -1, YEAR))
```

In other words, “Subtract last year’s sales from this year’s sales and divide by last year’s sales.” Everything else in the complete formula that is given in full earlier exists to prevent divide-by-zero errors or unwanted results for the first year where there is no previous year’s data!

The logic “wrapper” around the core formula uses two functions that you saw in Chapter 7, but it is worth taking another look at them here. They are:

- ISBLANK(): This function tests if a calculation returns nothing and allows you to specify what to do if this happens. This is a bit like an IF function that only tests for blank data.
- BLANK(): Returns a blank (or Null). This is useful for overriding unwanted results and replacing them with a blank.

Using these functions lets you handle the case where there is no data for a previous period of time. Because you are using the ISBLANK() function to test for nonexistent data, you are able to replace any missing data with a BLANK()—rather than letting Power BI Desktop display an unsightly error.

We have seen a couple of fairly complex formulas in a short section. So I think that it is a good idea to see how they look when you apply them. In this case, I will use a Power BI Desktop table to show the results, as shown in Figure 9-12. As you can see, the appropriate formats have been applied to each metric to enhance readability.

2014 Sales Trends				
MonthFull	SalePrice	PreviousYearSales	YearOnYearDelta	YearOnYearDeltaPercent
January	£555,500	£737,500	-£182,000	-24.68 %
February	£359,500	£610,750	-£251,250	-41.14 %
March	£372,500	£653,300	-£280,800	-42.98 %
April	£367,500	£642,800	-£275,300	-42.83 %
May	£529,500	£691,240	-£161,740	-23.40 %
June	£544,500	£558,000	-£13,500	-2.42 %
July	£565,500	£520,050	£45,450	8.74 %
August	£584,500	£455,000	£129,500	28.46 %
September	£606,500	£534,690	£71,810	13.43 %
October	£554,940	£638,250	-£83,310	-13.05 %
November	£657,000	£614,880	£42,120	6.85 %
December	£689,000	£502,250	£186,750	37.18 %
Total	£6,386,440	£7,158,710	-£772,270	-10.79 %

Figure 9-12. Power BI Desktop output for year on year comparisons

Once again, these formulas only scratch the surface of the myriad possibilities that DAX has on offer. However, you can adapt them to create comparisons by quarter, month, or day if you prefer simply by changing the specified time interval from YEAR to QUARTER, MONTH, or DAY.

Comparing with the Same Date Period from a Different Quarter, Month, or Year

In the last couple of sections, you saw various ways to compare data from a prior year or month. DAX offers one alternative method for these kinds of calculations that can be both easy to implement and extremely powerful. Moreover, it can serve as the basis for comparison with years, quarters, or months. This is the PARALLELPERIOD() function. Suppose that you want to use it to find average sales for the previous quarter.

Because this function is fairly similar to the DATEADD() function that you saw previously, I will not explain it step by step; instead, I prefer to give you three examples of measures that you can add to the InvoiceLines table directly.

Here are the sales for the preceding month:

```
SalesPrevMth = CALCULATE(SUM(InvoiceLines[SalePrice]), PARALLELPERIOD(DateDimension
[DateKey], -1, MONTH))
```

Here are the sales for the preceding quarter:

```
SalesPrevQtr = CALCULATE(SUM(InvoiceLines[SalePrice]), PARALLELPERIOD(DateDimension
[DateKey], -1, QUARTER))
```

Here are the sales for the preceding year:

```
SalesPrevYr = CALCULATE(SUM(InvoiceLines[SalePrice]), PARALLELPERIOD(DateDimension
[DateKey], -1, YEAR))
```

You see two examples of these functions displayed as simple visualizations in Figure 9-13.



Figure 9-13. Using the PARALLELPERIOD() function

Note You need to be aware that the PARALLELPERIOD() function compares the current data, not just up to the same data in the previous period (be it a year, a quarter, or a month), but for the entire previous time.

There are further DAX functions that you can also use when comparing data over time. These are outlined in Table 9-8.

Table 9-8. DAX Date and Time Formulas to Compare Values over Time

Formula	Description	Example
PARALLELPERIOD()	Finds dates from a “parallel” timeframe defined by a certain number of set intervals. The first parameter is the source data; the second is the number of years, quarters, months, or days; and the third is the definition of the interval: years, quarters, months, or days. A positive number of intervals looks forward in time and a negative number goes backward in time.	CALCULATE(SUM(InvoiceLines [SalePrice]), PARALLELPERIOD (DateDimension[DateKey], -1, MONTH))
SAMEPERIODLASTYEAR()	Finds the date(s) for the same time range one year before.	CALCULATE(SUM(InvoiceLines [SalePrice]), SAMEPERIODLASTYEAR (DateDimension[DateKey]))
DATEADD()	Used to return data from a past or future period in time compared to a specified date. The date difference can be in years, quarters, months, or days.	CALCULATE(SUM(InvoiceLines [SalePrice]), DATEADD(Date Dimension[DateKey], -1, YEAR))
DATESBETWEEN()	Calculates a list of dates between two dates. The first parameter is the start date and the second parameter is the end date.	DATESBETWEEN(Stock[Purchase Date], Invoices[SaleDate])
DATESINPERIOD()	Calculates a list of dates beginning with a start date for a specified period.	DATESINPERIOD(DateDimension [DateKey], DATE(2015,06,30), -90,day))

Rolling Aggregations over a Period of Time

We are now getting into the arena of more complex DAX formulas. So, since returning the rolling sum (or average) of a specified period to date necessitates several DAX functions and some in-depth nesting of these functions, I will take this as an example of a more complicated DAX formula. I will begin by outlining some of the functions that are used to deliver a result that is reliable and efficient:

- **DATESBETWEEN():** Lets you select a range of dates. The three parameters are the date key field from the date table, then the starting date, then the ending date.
- **FIRSTDATE():** Allows you to get the first date from a range. Since we are using this momentarily to go back a defined number of months, it will get the first day of the month.
- **LASTDATE():** Allows you to get the last date from a range. Since we are using this momentarily to go back a defined number of months, it will get the last day of the month.

You can now create two measures (3MonthsToDate and Previous3Months) using some fairly sophisticated logic to ensure that only blank cells are returned if there is no previous year's data using the following formulas:

```
3MonthsToDate=IF(
    ISBLANK(SUM(Stock[SalePrice])),
    BLANK(),
    CALCULATE(
        SUM(Stock[SalePrice]),
        DATESINPERIOD(
            DateDimension[DateKey],
            LASTDATE(DateDimension[DateKey]), -3, MONTH
        )
    )
)

Previous3Months=IF(
    ISBLANK(
        CALCULATE(
            SUM(Stock[SalePrice]),
            DATEADD(DateDimension[DateKey], -1, MONTH)
        )
    ),
    BLANK(),
    CALCULATE(
        SUM(Stock[SalePrice]),
        DATESBETWEEN(
            DateDimension[DateKey],
            FIRSTDATE(DATEADD(DateDimension[DateKey], -4, MONTH)),
            LASTDATE(DATEADD(DateDimension[DateKey], -1, MONTH))
        )
    )
)
```

The 3MonthsToDate formula essentially evaluates the code that is in boldface. This says, “Add up the sales for a time period ranging from three months ago to now,” using the InvoiceDate field as the date to evaluate. The IF function detects if there are sales for the current date, and if there are none (ISBLANK), then the calculation is not attempted, and a BLANK is returned.

The Previous3Months formula is pretty similar, except that the time span uses the DATESBETWEEN() function to set a range of dates—from the first day of the month four months ago to the last date in the preceding month.

DAX contains a couple of functions that you can use to return a specific date when aggregating data over time. These are shown briefly in Table 9-9.

Table 9-9. DAX Date and Time Formulas to Return a Date

Formula	Description	Example
FIRSTDATE()	Finds the first date that an event took place.	FIRSTDATE(Invoice[InvoiceDate])
LASTDATE()	Finds the last date that an event took place.	LASTDATE(Invoice[InvoiceDate])

Conclusion

This chapter has taken you on a short tour of some of the ways that you can use DAX in Power BI Desktop to extract meaning from your data by analyzing its evolution over time.

First, you saw how to extract date and time elements from columns that contain dates. The new columns that you create based on dates can then be used to filter or group data in your visualizations. This way you can provide a daily, weekly, monthly, quarterly, and yearly breakdown of your source data. These date elements can also help you to filter the datasets that you are using by dates, date elements, and date ranges.

Then you saw how to prepare the dataset for time intelligence by adding a date table and joining this table to the other tables in the data model. Finally, you saw how to start adding formulas to the dataset to prepare all the time-based metrics that your Power BI Desktop reports could need. This can include analyzing sales to date or comparing data from previous time periods with current data.

Explaining all the possibilities of DAX would take an entire book, so all I wanted to do in this and the previous two chapters was explain how you can use core DAX functions in a handful of useful calculations. I sincerely hope that this brief overview helps you on your road to mastery of DAX and that you are able to apply these formulas to deliver stunning visualizations using Power BI Desktop.

Your data is now ready for output. It can be used as the basis for multiple dashboards and reports. This is the subject of the next six chapters.

CHAPTER 10



Text-Based Visualizations

You are now entering the final straight on your race to deliver clear, powerful, and visually compelling analysis. The time has come to transform data into attention-grabbing dashboards that capture the imagination of your audience. In this chapter, you will learn how to use Power BI Desktop to

- Delve deeply into data and produce valuable information from the mass of facts and figures available.
- Create interactive views of your insights, where you can test your analyses quickly and easily.
- Enhance the presentation of your results to grab your audience's attention.

This chapter takes you through the process of creating text-based visualizations (or visuals if you want to call them that) in Power BI Desktop. You will learn how to create and enhance

- Tables
- Matrices
- Cards
- Multirow cards

What these types of visuals all have in common is that they are designed to display text to convey information rather than using the more graphical display types, which you will discover in the next two chapters. These elements are essential when it comes to

- Presenting lists of information
- Displaying crosstabs of key data
- Focusing the audience's attention on a single essential figure
- Delivering a clear overview of a few key metrics

The techniques that you will learn in this chapter will use a Power BI Desktop file that results from the application of much of what you saw in the last four chapters. This file is called DataModelWithMetricsForVisualizations.pbix. If you need a ready-prepared version of this file then it is available on the Apress web site as part of the downloadable material that accompanies this book.

Power BI Desktop Dashboards

So far in this book, you have used two of the three core interface modes of Power BI Desktop: Data View and Relationships View. Now that you are moving on to create dashboards, you will use the final part of the trilogy: Report View.

You always use Report View to build your dashboards (or reports if you prefer). It is here that you add and configure visuals that use the data that you have already loaded, cleansed, structured, and enhanced in the data model.

Switching to Report View

Although Report View is the default mode when you open Power BI Desktop, you need to switch back to it if you have spent any time working on the underlying data model. The following explains how to activate Report View.

1. Click the Report View icon at the top left of the Power BI Desktop screen.

This icon is shown in Figure 10-1.

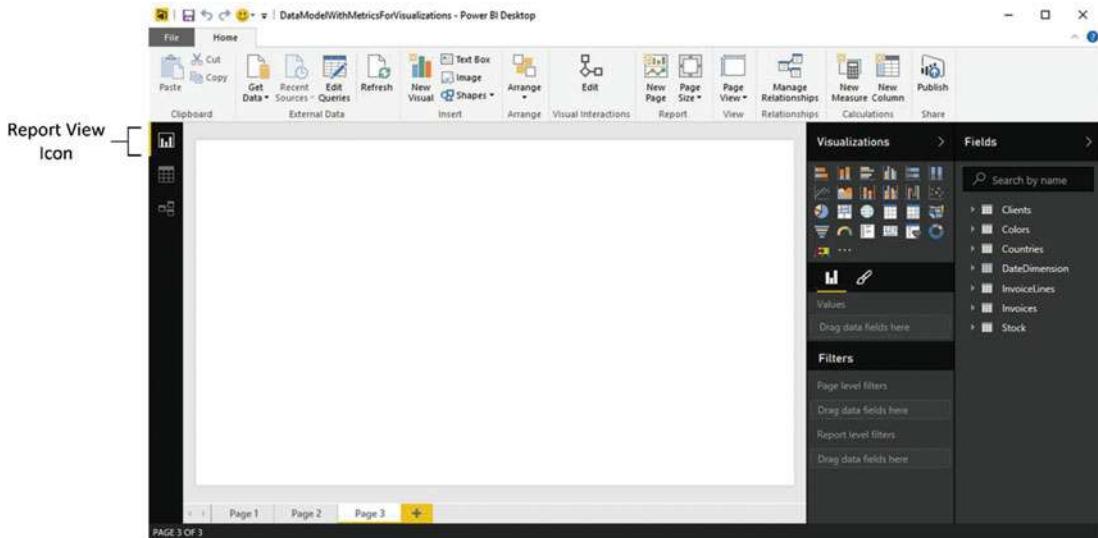


Figure 10-1. The Power BI Desktop Report View icon

Note You learn how to create and modify complex multipage reports in Chapter 15.

Tables

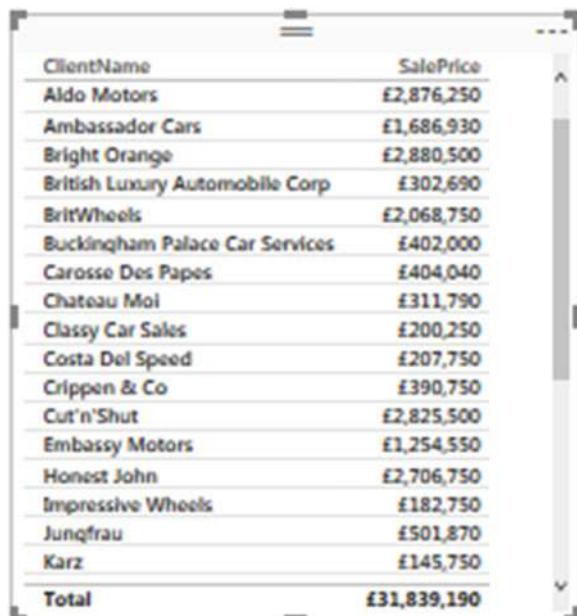
Tables are probably the simplest and most elementary way of displaying data. However, this simplicity should not detract from their usefulness. Indeed, for Power BI Desktop the humble table is the default visualization. Moreover, and as you will see in this section, the table is the default presentation style for non-numeric data.

I realize, of course, that it may seem contradictory to spend time on a subject that is generally described as intuitive. In answer to this I can only say to this that while getting up and running is easy, attaining an in-depth understanding of all of the potential of this powerful tool does require some explanation. Consequently, the approach that I am taking is to go through all the possibilities of each item as thoroughly as possible. So feel free to jump ahead (and back) if you don't need all the detail just yet.

Creating a Basic Table

The following example will introduce you to tables in Power BI Desktop by creating an initial table that will display the list of clients and their total sales.

1. Open the C:\PowerBiDesktopSamples\DataAdapterWithMetricsForVisualizations.pbix file from the downloadable samples.
2. In the Fields list, expand the Clients table.
3. Drag the ClientName field on to the dashboard canvas. A table displaying all the clients of Brilliant British Cars will appear.
4. Leave this new table selected and, in the Fields list expand the InvoiceLines table.
5. Click the check box to the left of the SalePrice field. This will add the cumulative sales per client to the table. It should look like Figure 10-2.



A screenshot of a Power BI desktop dashboard. On the left, there is a visual representation of a table with two columns: 'ClientName' and 'SalePrice'. On the right, the actual table data is displayed in a scrollable window. The table contains 20 rows of data, plus a summary row at the bottom labeled 'Total' with a value of £31,839,190. The columns are labeled 'ClientName' and 'SalePrice'.

ClientName	SalePrice
Aldo Motors	£2,876,250
Ambassador Cars	£1,686,930
Bright Orange	£2,880,500
British Luxury Automobile Corp	£302,690
BritWheels	£2,068,750
Buckingham Palace Car Services	£402,000
Carosse Des Papes	£404,040
Chateau Moi	£311,790
Classy Car Sales	£200,250
Costa Del Speed	£207,750
Crippen & Co	£390,750
Cut'n'Shut	£2,825,500
Embassy Motors	£1,254,550
Honest John	£2,706,750
Impressive Wheels	£182,750
Jungfrau	£501,870
Karz	£145,750
Total	£31,839,190

Figure 10-2. A basic table of sales per client

This is a very tiny table. In the real world, you could be looking at tables that contain thousands, or tens of thousands (or even millions) of records. Power BI Desktop accelerates the display of large datasets by only loading the data that is required as you scroll down through a list. So you might see the scroll bar advance somewhat slowly as you progress downward through a large table.

If a table contains fields that you have used in a visual, then the table name appears in yellow in the Fields list when you select the visualization. You can always see which fields have been selected for a table by expanding the table in the Fields list. The fields used are instantly displayed in both the Fields list (as selected fields) and in the field well of the Visualizations pane. To get you used to this idea, see Figure 10-3, which shows both these elements for the table that you just created.

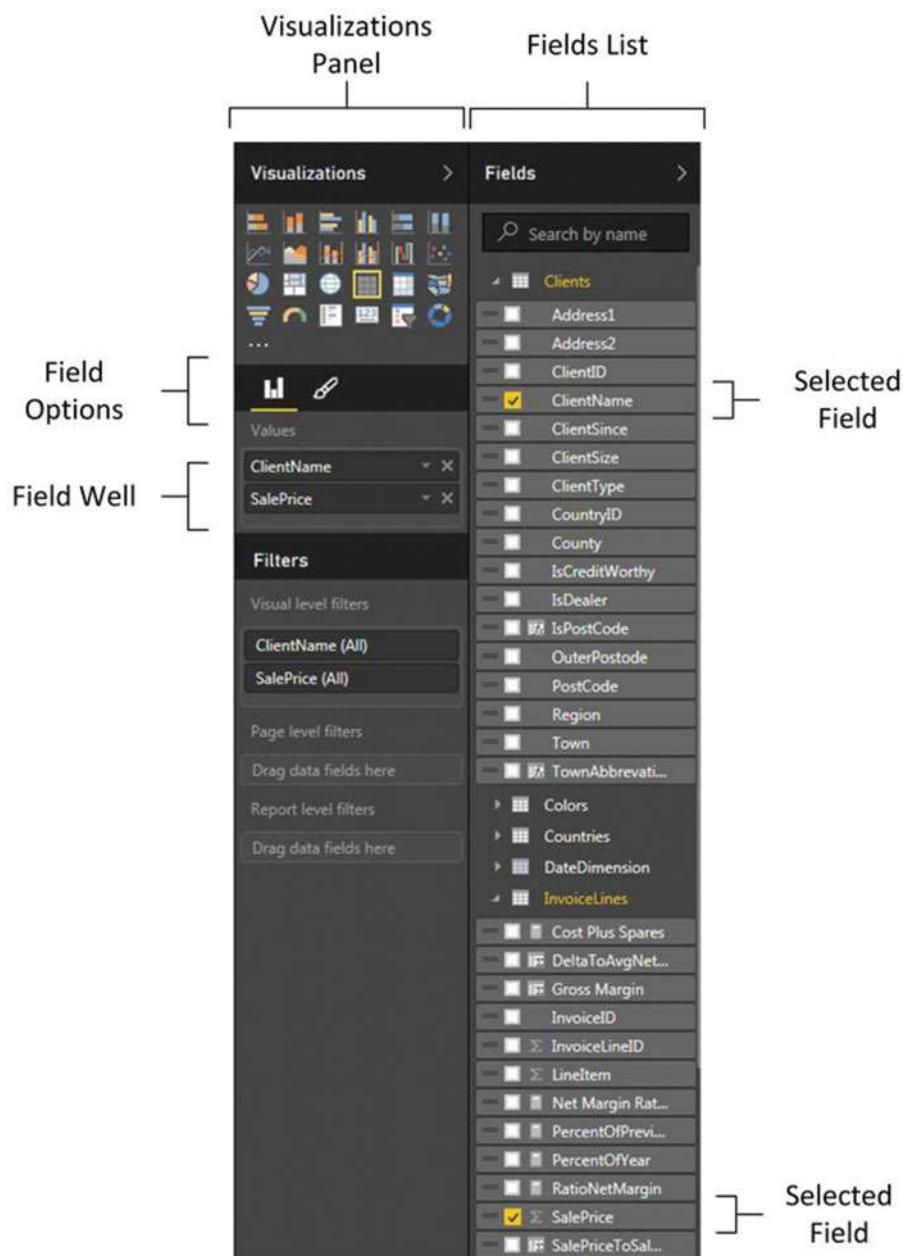


Figure 10-3. The fields used in a visual

As befits such a polished product, Power BI Desktop does not limit you to just one way of adding fields to a table. The following are ways in which you can add fields to a table:

- By dragging the field name into the Fields section at the bottom of the Fields list.
- By selecting a field (which means checking the check box to the left of the field) in the Fields list.
- By dragging a field on to an existing visualization.

You can add further fields to an existing table at any time. The key thing to remember (if you are using any of the three techniques described) is that you must select the table that you want to modify first. This is as simple as clicking inside it. After you click, you instantly see that the table is active because tiny handles appear at the corners of the table as well as in the middle of each side of the table.

Note If you do not select an existing table before adding a field, Power BI Desktop will create a new table using the field that you are attempting to add to a table.

To create another table, all you have to do is click outside any existing visuals in the Power BI Desktop report and begin selecting fields as described earlier. A new table is created as a result. Power BI Desktop always tries to create new tables in an empty part of the canvas. You will see how to rearrange this default presentation shortly.

Deleting a Table

Suppose that you no longer need a table in a Power BI Desktop report. Well, that is simple, just

1. Select the table. You can do this by clicking anywhere inside the table.
2. The table borders will appear, even if you move the mouse pointer away from the table.
3. Press the Delete key.

Another way to select a table is to click the options button that appears at the top right of any visual once it is selected. The options for the visual (the table in this example) will appear. Clicking the Remove option will delete the table. You can see the context menu in Figure 10-4.

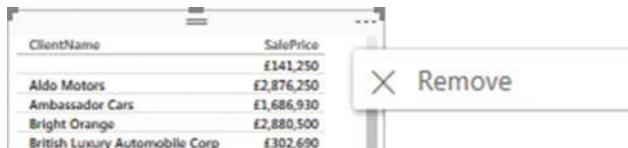


Figure 10-4. The table visualization options menu

Deleting a table is so easy that you can do it by mistake, so remember that you can restore an accidentally deleted table by pressing Ctrl+Z or using the Undo arrow at the top left of the Power BI Designer window.

Copying a Table

You need to copy tables on many occasions. There could be several reasons for this:

- You are creating a new visual on the Power BI Desktop report and need the table as a basis for the new element, such as a chart, for instance.
- You are copying visuals between reports.
- You want to keep an example of a table and try some fancy tricks on the copy, but you want to keep the old version as a failsafe option.

In any case, all you have to do is

1. Select the table (as described previously).
2. In the Home ribbon, select Copy (or press Ctrl+C).

To paste a copy, click outside any visual in a current or new Power BI Desktop report, and select Paste from the Home ribbon (or press Ctrl+V).

Changing the Table Size and Position

A table can be resized and moved just like any other visual in a Power BI Desktop report.

Resizing a Table

Resizing a table is mercifully easy. All you have to do is to click any of the table handles and drag the mouse. Lateral handles will alter the table width; top and bottom handles will change the table's height; corner handles modify both height and width by changing the size of the two sides that touch on the selected corner.

Power BI also has one formatting option that you may find useful when resizing tables. You can lock a table's aspect ratio; that is, the ratio of the height to width. Doing this means that you cannot alter the relative height and width of a table when resizing it.

To lock the aspect ratio

1. Click the table to select it.
2. In the Visualizations pane, click the Format icon under the collection of available visuals (the small paintbrush). You can see this icon in Figure 10-3.
3. In the Lock Aspect tab, click Off to turn the lock on, or alternatively, click to the right of the empty circle indicating that the lock is off so that it slides to the right and becomes a filled circle.

Now, when you resize a table using the corner handles, it will keep its height to width ratio.

Moving a Table

Moving a table is as easy as placing the pointer over the table so that the edges appear, and once the cursor changes to the hand shape, dragging the table to its new position.

Changing Column Order

If you have built a Power BI Desktop table, you are eventually going to want to modify the order in which the columns appear from left to right. The following explains how to do this.

1. Activate the Fields list (unless it is already displayed).
2. In the Visualizations pane, ensure that the Fields options are displayed (by clicking the small bar chart icon under the collection of available visuals).
3. Click the name of the field that you wish to move.
4. Drag the field vertically to its new position. This can be between existing fields, at the top or at the bottom of the Fields list. A thick yellow line indicates where the field will be positioned.

Figure 10-5 shows how to drag a field from one position to another.

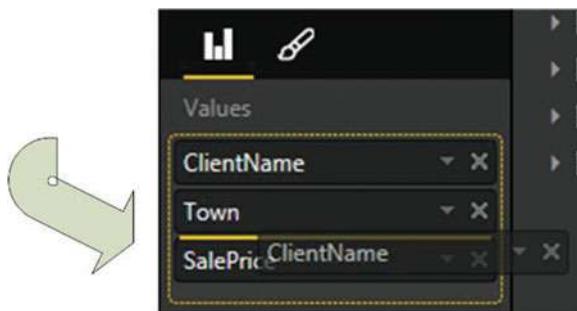


Figure 10-5. Changing column order by moving fields

■ **Note** You cannot change the position of a column in a table by dragging it sideways inside the table itself.

Removing Columns from a Table

Another everyday task in Power BI Desktop is removing columns from a table when necessary. As is the case when rearranging the order of columns, this is not done directly in the table but is carried out using the Fields list. There are, in fact, several ways of removing columns from a table, so I will begin with the way that I think is the fastest and then describe the others.

1. Activate the Fields list—unless it is already displayed.
2. Uncheck the field name in the Fields list.

Assuming that a visual is selected, the following are other ways to remove a field:

- In the Visualizations pane, click the cross icon at the right of the field name.
- In the Visualizations pane, click the small triangle at the right of the name in the field well (remember that these are the fields used in the visualization). When the pop-up menu appears select Remove Field.

Note Do not click the pop-up menu icon (the ellipses at the right of the field name) and select Delete in the Fields list to remove a field from the table. Doing this deletes the field from the data model.

Types of Data

Not all data is created equal, and the data model that underlies Power BI Desktop will provide you with different types of data. The initial two data types are

- Descriptive (non-numeric) attributes
- Values (or numeric measures)

Power BI Desktop indicates the data type by using a descriptive icon beside many of the fields, which you can see when you expand a data table in the Fields list. These data types are described in Table 10-1.

Table 10-1. Data Types

Data Type	Icon	Comments
Attribute	None	A descriptive element and is non-numeric. It can be counted but not summed or averaged.
Aggregates		A numeric field whose aggregation type can be changed.
Calculated Column		The result of a formula applied to create a new, calculated column.
Calculation		A numeric field whose aggregation type cannot be changed as it is the result of a specific calculation.
Geography		This field can potentially be used in a map to provide geographical references.
Binary Data		This field contains data, such as images.

Numeric fields are not the only ones that can be added as aggregates. If you add an attribute field to a table and then display the pop-up menu for this field by clicking the small triangle to the right of the field name in the Visualizations pane, you can select Count. You will display the *number* of elements for this attribute in the column of the table instead of the text of the element. Figure 10-6 shows you a sample pop-up field menu for a text field.

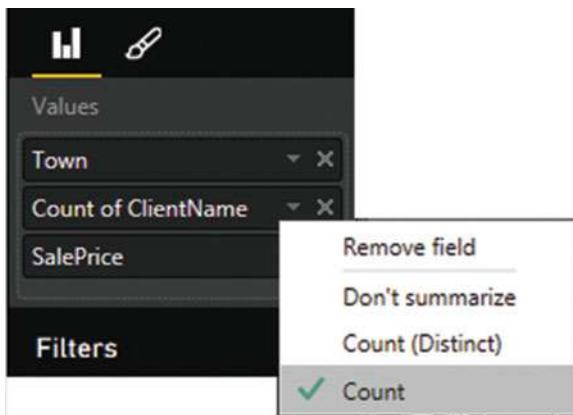


Figure 10-6. The pop-up menu for a text field

Enhancing Tables

So you have a basic table set up and it has the columns you want in the correct order. Quite naturally, the next step is to want to spice up the presentation of the table a little. So let's see what Power BI Desktop has to offer here. Specifically, we will look at

- Adding and removing totals
- Formatting columns of numbers
- Changing columns widths
- Sorting rows by the data in a specific column
- A few other aspects of table formatting

Row Totals

Row totals are added automatically to all numeric fields. You may wish to remove the totals, however. Conversely, you could want to add totals that were removed previously. In any case, to remove all the totals from a table

1. Select the table. In this example, I use the table that you saw in Figure 10-2.
2. In the Visualizations pane, click the Format icon (the small paintbrush beneath the palette of available visual types). The formatting options for the selected table will be displayed.
3. Expand the General tab by clicking the downward-facing chevron to the left of the word *General*.
4. Drag the full circle button (to the right of Totals) to the left. It will become an empty circle and On will become Off in the formatting options. The Visualizations pane will look what is shown in Figure 10-7.

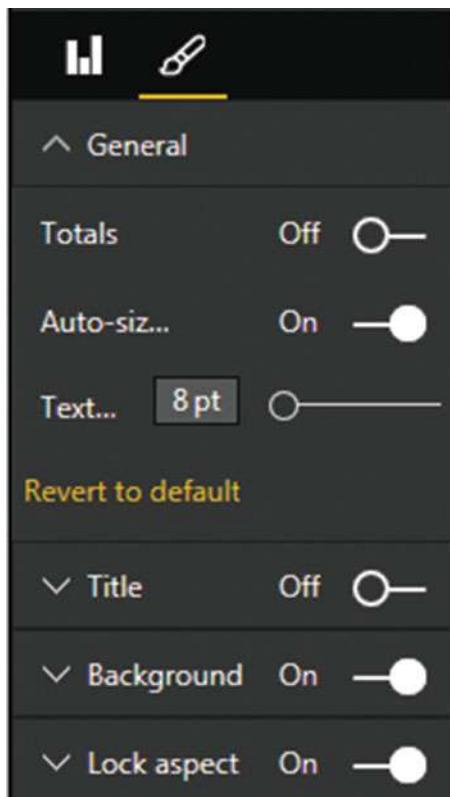


Figure 10-7. Formatting options for tables in the Visualizations pane

To add totals where there are none, follow the process described (with the table selected) and at step 4, drag the Off button to the right for Totals to set the totals to On. You can see the table you created previously—without totals—in Figure 10-8.

The screenshot shows a table with two columns: 'Town' and 'SalePrice'. The 'Town' column lists 30 different locations, and the 'SalePrice' column lists their respective sale prices in British Pounds (£). The table is presented in a clean, modern style with a light gray background and white text. The 'SalePrice' column uses a standard currency format with commas as thousands separators.

Town	SalePrice
Avignon	£141,250
Basle	£404,040
Bellevue	£233,600
Birmingham	£1,686,930
Cambridge	£2,880,500
Chevy Chase	£641,000
Denver	£1,714,550
Franklin	£1,254,550
Geneva	£302,690
Glasgow	£705,500
Gloucester	£390,750
Liverpool	£214,750
London	£182,750
Louisville	£5,839,500
Lyon	£358,690
Madrid	£311,790
Manchester	£2,825,500
Marseille	£442,430
Mason	£402,000
New York	£1,027,500
Newcastle upon Tyne	£85,250
Paris	£1,366,250
Pittsburgh	£200,250
Portland	£2,068,750
San Francisco	£1,997,050
Shrewsbury	£297,500
Stuttgart	£145,750
Telford	£132,250
Uttoxeter	£2,876,250
Zurich	£501,870

Figure 10-8. The initial table without totals

Note You can only add or remove totals if a table displays multiple records. If a table is displaying the highest level of aggregation for a value, then no totals can be displayed, as you are looking at the grand total already. In this case, the Totals button is grayed out.

Formatting Columns of Numbers

You cannot format numbers in tables in a Power BI Desktop report. This is because all number formatting is centralized in the data model. So if you want to apply a different numeric format to the one that appears when you add a value to a table, you will have to switch to Data View and apply the formatting there. If you need to remind yourself exactly how this is done then just flip back to Chapter 6 for a quick refresher course on how to format numbers in Power BI Desktop.

Font Sizes in Tables

You may prefer to alter the default font size that Power BI Desktop applies when a table is first created. This is easy to do.

1. Select the table.
2. In the Visualizations pane, click on the Format icon.
3. Expand the General tab by clicking the downward-facing chevron to the left of the word *General*.
4. Drag the full circle button to the right of Text Size to the right. The new text size in points will be displayed in the Text Size box and the actual size of the text in the table will increase.

I really should add a few of points to conclude on font sizes in tables:

- You *cannot* select a font size; you can only use the Text slider until you have found a size that suits you. My impression is that the available range is from 6 to 40 point.
- As you probably have guessed, dragging the Text Size slider button to the left will decrease the font size.
- Altering the font size will not cause the table to grow or shrink, as Power BI Desktop will continue to display the same number of characters per column as were visible using the previous font size. So you may end up having to alter the column widths by setting Auto-size to On or adjusting the table size (as described previously) to make your table look exactly the way you want it.

Changing Column Widths

Power BI Desktop will automatically set the width of a column so that all the data is visible. You cannot manually adjust the size of individual columns in a table. However, you can prevent Power BI Desktop from resizing columns once they have been added.

1. Select the table. In this example, I use the table that you saw in Figure 10-8.
2. In the Visualizations pane, click the Format icon (the small paintbrush beneath the palette of available visual types). The formatting options for the selected table will be displayed.
3. Expand the General tab by clicking the downward-facing chevron to the left of the word *General*.
4. Drag the full circle button to the right of Auto-size column width to the left. It will become an empty circle and On will become Off in the formatting options. The Visualizations pane will look like the one in Figure 10-9.

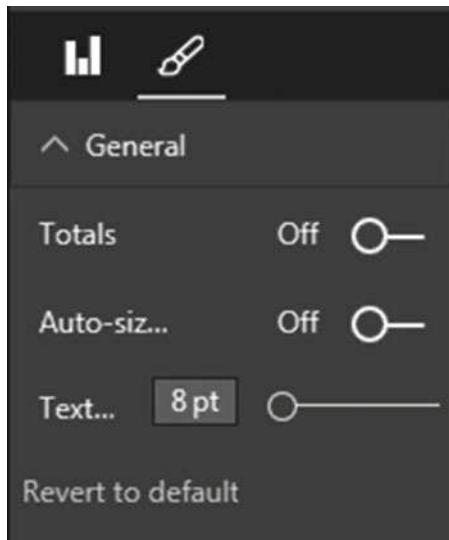


Figure 10-9. Inhibiting automatic column resizing for tables in the Visualizations pane next, you can alter the size of the font used in the table without causing column widths to grow or shrink. Inversely, resetting The Auto-size option to On causes the column width to be readjusted automatically to display the widest element in the column

Note If you are formatting a table and you start to get a little confused as to which options are active and how best to start over, you can always click “Revert to default” (visible in Figure 10-9) to reapply the standard “factory settings” for tables—or for any visual.

Adding and Formatting Titles

Like most visualizations in Power BI Desktop, tables can have titles. Once you have added a title, you can set its

- Font color
- Background color
- Text alignment

As simple as this is, I prefer to explain all the options for the sake of completeness.

1. Select the table. In this example, I use the table that you saw in Figure 10-8.
2. In the Visualizations pane, click the Format icon (the small paintbrush beneath the palette of available visualization types). The formatting options for the selected table will be displayed.
3. Expand the Title tab by clicking the downward-facing chevron to the left of the word Title.

4. Click to the right of the empty circle to the right of the word Off. This will activate a title for the selected table.
5. In the empty box to the right of Title Text, enter a title for the table. I suggest **Sales by Town**.
6. Click the pop-up menu triangle to the right of Font color and select a color from the palette of available hues.
7. Click the pop-up menu triangle to the right of Background color and select a color from the palette of available tones.
8. Click the middle icon to the right of Alignment. This will center the text relative to the width of the table.

The formatting options and the table will look like they do Figure 10-10.

The screenshot shows a data visualization application. On the left, there is a table titled "Sales by Town" with two columns: "Town" and "SalePrice". The table lists various towns and their corresponding sales prices. On the right, a floating window displays the following options:

- Title**: A switch labeled "On" is turned on, and the "Title Text" input field contains the text "Sales by Town".
- Font color**: A dropdown menu is open, showing a purple square as the current selection.
- Background color**: A dropdown menu is open, showing a yellow square as the current selection.
- Alignment**: A set of three icons for alignment: left, center (selected), and right.
- Revert to default**: A button at the bottom of the panel.

Figure 10-10. Options for titles in tables

Modifying the Table Background

The table title is not the only aspect of a table that you can modify for visual effect. You can also change a couple of elements of the table itself to add pizazz to your dashboards. The following are the two things that you can modify:

- The background color
- The table transparency

The following example illustrates both of these possibilities.

1. Select the table. In this example, I use the table that you saw in Figure 10-10.
2. In the Visualizations pane, click the Format icon (the small paintbrush beneath the palette of available visual types). The formatting options for the selected table will be displayed.
3. Expand the Background tab by clicking the downward-facing chevron to the left of the word Background.
4. Ensure that the switch to the right of the word Background is set to On (if it is not, then click to the right of the empty circle to change this).
5. Click the pop-up menu triangle to the right of Color and select a color from the palette of available colors.
6. Click the Transparency slider switch and slide it to the left to intensify the background color by reducing the percentage of transparency. The formatting options and the table will look like Figure 10-11.



Figure 10-11. Setting the background color for a table

Table Borders

As a final tweak, you can add an outside border to any table. The following explains how to do this.

1. Select the table. In this example, I use the table that you saw in Figure 10-10.
2. In the Visualizations pane, click the Format icon (the small paintbrush beneath the palette of available visual types). The formatting options for the selected table will be displayed.
3. Slide the Border button to the right. This will apply a border to the table.
4. Expand the Border tab.
5. Select a border color from those available in the pop-up palette.

Sorting by Column

Any column can be used as the sort criterion for a table, whatever the type of table. To sort the table, merely click the column header. The rows in the table are sorted according to the elements in the selected column. The sort order is A to Z (or lowest to highest for numeric values) the first time that you sort a column.

Once a table has been sorted, you cannot unsort it. You can use another column to resort the data, however. Alternatively, you can click again on the column title to sort in the opposing order—Z to A (or highest to lowest for numeric values).

For an example of sorting a column, look at Figure 10-12 (once again, I will use the table we created at the very beginning of this chapter). You can see that the data is sorted by Sale Price from highest to lowest.

Sales by Town	
Town	SalePrice
London	£5,839,500
Birmingham	£2,880,500
Uttexeter	£2,876,250
Manchester	£2,825,500
Portland	£2,068,750
San Francisco	£1,997,050
Chevy Chase	£1,714,550
Bellevue	£1,686,930
Paris	£1,366,250
Denver	£1,254,550
New York	£1,027,500
Geneva	£705,500
Cambridge	£641,000
Zurich	£501,870
Marseille	£442,430
Avignon	£404,040
Mason	£402,000
Glasgow	£390,750
Louisville	£358,690
Lyon	£311,790
Franklin	£302,690
Shrewsbury	£297,500
Basle	£233,600
Gloucester	£214,750
Madrid	£207,750
Pittsburgh	£200,250
Liverpool	£182,750
Stuttgart	£145,750
	£141,250
Telford	£132,250
Newcastle upon Tyne	£85,250

Figure 10-12. Sorting a table by a numeric column

Sometimes you are sorting a column on one field (as was the case in all the examples so far), but the actual sort uses another column as the basis for the sort operation. For example, you could sort by month name but see the result by the month number (so that you are not sorting months alphabetically, but numerically). You saw how this is configured in Chapter 9.

Table Granularity

A Power BI Desktop table will automatically aggregate data to the lowest available level of grain. Put simply, this means that it is important to select data at the lowest useful level of detail but not to add pointlessly detailed elements.

This is probably easier to understand if I use an example. Suppose you start with a high level of aggregation—the country for instance. If you create a table with CountryName and Sales columns, it will give you the total sales by country. If you use the sample data given in the examples for this book (the DataModelWithMetricsForVisualisations.pbix file), this table only contains half a dozen or so lines.

Then add the ClientName after the country. When you do this, you obtain a more finely grained set of results, with the aggregate sales for each client in each country. If you (finally) add the InvoiceNumber, you get a very detailed level of data. Indeed, adding such a fine level of grain to your table could produce an extremely large number of records—as indicated by the appearance of a vertical scroll bar in this table. These progressive levels of granularity are shown in Figure 10-13.

CountryName	SalePrice	CountryName	ClientName	SalePrice	CountryName	ClientName	InvoiceNumber	SalePrice
France	£141,250	France	Carrosse Des Pages	£41,250	France	Carrosse Des Pages	080ADABBB-82F8-4DD5-815C-118AE7159C0D	£25,250
Germany	£2,524,510	France	Chateau Moi	£404,040	France	Carrosse Des Pages	67A45618-68D0-4865-9107-3F531F3157F5	£44,000
Spain	£145,750	France	Les Arnaqueurs	£311,790	France	Carrosse Des Pages	82F2E58C-1ZC8-41E6-94BE-3546F1CE50FD	£25,250
Switzerland	£1,440,970	France	Vive la Vitesse!	£366,250	France	Carrosse Des Pages	C2FC2EB7-E5B8-4741-8688-2AAACF735094	£46,750
United Kingdom	£15,725,000	Germany	Karz	£145,750	France	Carrosse Des Pages	38B5A0EF-EFDA-4248-B721-A9A7977A0F46C	£72,000
USA	£11,653,960	Spain	Costa Del Speed	£207,750	France	Carrosse Des Pages	45275E14-7807-8D7D-1E5058AB515	£77,250
Total	£31,839,190	Switzerland	Jungfrau	£501,870	France	Carrosse Des Pages	45D9282D-9E7-49CF-93C5-FDA9E736B84D	£77,250
		Switzerland	Three Country Cars	£233,600	France	Carrosse Des Pages	4D4E5593-7DAA-4FFF-A493-2F6F5CA625A1	£37,690
		Switzerland	Voitures Diplomatiques S.A.	£705,500	France	Carrosse Des Pages	6FA0ABC3-18A1-41E6-96D0-D1DF1C60C79C	£25,500
		United Kingdom	Aldo Motors	£876,250	France	Carrosse Des Pages	78273430-FD2E-4E06-847C-6E3B1165FD07	£22,750
		United Kingdom	Bright Orange	£2,880,500	France	Carrosse Des Pages	8288A324-C314-41B8-AA67-C16D8117869C	£45,800
		United Kingdom	Crippen & Co	£390,750	France	Carrosse Des Pages	84C95EFE-E0AB-4C13-BAAD-ADD268CA40A4	£45,800
		United Kingdom	Cut'n'Shot	£2,825,500	France	Chateau Moi	311AA133-E929-47E6-9E89-C39987FD0015	£74,500
		United Kingdom	Honest John	£2,706,750	France	Chateau Moi	52D515E5-9E93-42E1-9D2B-C39E9A795A56	£28,000
		United Kingdom	Impressive Wheels	£182,750	France	Chateau Moi	AFFDA41-2D10-4732-98E5-3B800C828F10	£77,250
		United Kingdom	Luxury Rentals	£214,750	France	Chateau Moi	E027A469-FDDE-4D3E-88A8-238F18645E21	£45,800
		United Kingdom	Olde Englande	£297,500	France	Chateau Moi	E34523F3-A2DC-40B9-8E7F-60E568285B09	£48,550
		United Kingdom	Premium Motor Vehicles	£85,250	France	Les Arnaqueurs	E3EBBF63-A0D3-4A0D-87A1-8105FAFA96D2	£37,690
		United Kingdom	Smooth Riders	£132,250	France	Les Arnaqueurs	064F1E61-9A9C-4708-A3A6-9859C31D321B	£42,250
		United Kingdom	Wheels'R'Us	£132,750	France	Les Arnaqueurs	0A07EB61-9E26-4E89-9E59-4887CFC08462	£46,750
		USA	Ambassador Cars	£1,686,930	France	Les Arnaqueurs	0B01B246-A95B-4817-ATF2-537F17A9C443	£44,000
		USA	British Luxury Automobile...	£302,690	France	Les Arnaqueurs	32841002-BE87-4080-8516-DCE689837C50	£44,000
		USA	BritWheels	£2,068,750	France	Les Arnaqueurs	3E106580-BF01-4CDB-A19C-4857E2A8EBE0	£39,500
		USA	Buckingham Palace Car Se...	£402,000	France	Les Arnaqueurs	4799184A-499A-46A6-95EF-1007162A270	£130,000
		USA	Classy Car Sales	£200,250	France	Les Arnaqueurs	5273F275-EFC7-4A58-9CEA-9CFEE2B0D47C	£42,250
		USA	Embassy Motors	£1,254,550	France	Les Arnaqueurs	6055A276C-B403-4A6A-AC95-55093386A316	£110,000
		USA	Rocky Riding	£1,027,500	France	Les Arnaqueurs	79203209-7C00-4161-91B8-98CC9BC0D6E1	£181,250
		USA	Sporty Types Corp	£1,997,050	France	Les Arnaqueurs	7A48B028-795B-49DE-AD0E-8CA679E3642	£127,750
		USA	Style 'N Ride	£1,714,550	France	Les Arnaqueurs	7D49FE49-27F5-4459-AE33-28710E0E7A76	£44,000
		USA	Tweedyle Wheels	£641,000	France	Les Arnaqueurs	9C4F78D0-8887-4D91-8173-817354CE5F5	£110,000
		USA	Union Jack Sports Cars	£358,690	France	Les Arnaqueurs	9E31A880-8D33-4D1E-88FB-8A47B7470009	£29,750
		Total		£31,839,190	France	Les Arnaqueurs	ADFFAC9E-DH3-4B4B-9ECA-DE9A28693350	£125,000
					France	Les Arnaqueurs	C4A55876-3893-4D12-89E7-D381C9B6428	£95,000
					France	Les Arnaqueurs	D4E22C9B-B725-42A0-8219-0520376C35E2	£71,250
					France	Les Arnaqueurs	E0479884-0C75-49E3-A166-17FF5754F2	£44,000
					France	Les Arnaqueurs	F636F5FD-BF2A-4231-BFC5-F610DD39388A6	£39,500
					France	Vive la Vitesse!	6D5EC6A1-547E-4821-9735-EF276C075D68	£69,250
					France	Vive la Vitesse!	998D5B44-2111-42FF-B48B-BD9CD89EAE00A	£77,250
					France	Vive la Vitesse!	A3CC4CA4-9AE1-472C-8C38-7127C38CE9E	£74,500
					France	Vive la Vitesse!	A454608F-9975-4DEA-89F4-BCBE8CD86985	£22,750
								£31,839,190

Figure 10-13. Progressive table granularity

Power BI Desktop always attempts to display the data using the information available to it in the underlying data model. Exactly how this can be optimized for the best possible results is described in Chapters 6 through 9.

Creating a Matrix

So far in this chapter, we have limited ourselves to tables that display the information as full columns of lists, just like the source data in an Excel spreadsheet or in a database, for instance. Lists do not always give an intuitive feeling for how data should be grouped at various levels, however. Presenting information in a neat hierarchy with multiple grouped levels is the task of a matrix-type table.

Row Matrix

When creating a matrix table, I find that it helps to think in terms of a hierarchy of information and to visualize this information flowing from left to right. For instance, suppose that we want to create a matrix with the country name as the highest level in the hierarchy (and consequently the leftmost item). Then we want the make of car to be the second level, and the next element in from the left. (In Figure 10-14, I've labeled it Make.) Finally, we want the color of car sold, followed by all the numeric fields that interest us.

Our hierarchy is shown in Figure 10-14.



Figure 10-14. An information hierarchy

When creating a matrix, it is important to have the Visualizations pane reflect the hierarchy. Put another way, you must ensure that the order of the fields that you select for the table and that you place in the field well follows the display hierarchy that you want for the matrix. Consequently, to create a matrix table like the one just described, you need to do the following:

1. Use the DataModelWithMetricsForVisualizations.pbix file as your source of data.
2. Click outside any existing visuals.
3. Drag the fields CountryName (from the Countries table), Make (from the Stock table), and Color (from the Colors table) in this order to the field well in the Visualizations pane. Then add the fields SalePrice and GrossMargin (from the InvoiceLines table). This will create a table automatically. The table will be very long, but we will not worry about that at this point. The table should look something like Figure 10-15.

CountryName	Make	Color	SalePrice	Gross Margin
	Bentley	Canary Yellow	£46,750	19106
	Triumph	British Racing Green	£25,250	12475
	Triumph	Canary Yellow	£25,250	12475
	TVR	Blue	£44,000	2225
France	Aston Martin	Blue	£141,250	105250
France	Aston Martin	British Racing Green	£181,250	52420
France	Aston Martin	Canary Yellow	£284,440	109085
France	Aston Martin	Green	£108,990	38855
France	Aston Martin	Night Blue	£165,600	26000
France	Aston Martin	Red	£450,300	322490
France	Aston Martin	Silver	£155,380	88570
France	Bentley	Blue	£44,000	16000
France	Bentley	British Racing Green	£39,500	-19444
France	Bentley	Canary Yellow	£110,000	51200
France	Bentley	Dark Purple	£44,000	-19944
France	Bentley	Red	£110,000	87000
France	Bentley	Silver	£46,750	18987
France	Jaguar	Black	£84,500	42136
France	Jaguar	Canary Yellow	£88,000	11161
France	Jaguar	Night Blue	£39,500	17200
France	Rolls Royce	Black	£48,250	2540
France	Rolls Royce	Blue	£72,000	51345
France	Rolls Royce	Canary Yellow	£207,250	112440
France	Rolls Royce	Night Blue	£45,800	30445
France	Triumph	Silver	£28,000	6600
France	TVR	Silver	£29,750	-7025
Germany	Jaguar	Green	£42,250	475
Germany	Jaguar	Red	£32,500	-9114
Germany	TVR	Blue	£41,250	4475
Germany	TVR	Silver	£29,750	-7025
Spain	Bentley	Canary Yellow	£46,750	19106
Spain	Jaguar	Green	£39,500	2725
Spain	Jaguar	Red	£29,750	-6864
Spain	Triumph	Canary Yellow	£47,750	18550
Total			£31,839,190	12139884

Figure 10-15. A table before conversion to a matrix

- With the table selected, click the Matrix icon in the Visualizations pane. This icon is immediately to the right of the Table icon that you saw in Figure 10-2. The field well in the Visualizations pane splits to add two new boxes: Rows and Columns. The table and the field well in the Visualizations pane will now look like those shown in Figure 10-16.

The screenshot shows a Power BI Desktop interface with a matrix table visualization. The matrix is organized by CountryName (France, Germany), Make (Bentley, Triumph, TVR, Aston Martin, Jaguar, Rolls Royce), Color (various colors like Blue, British Racing Green, Canary Yellow, etc.), and Sales figures (SalePrice and Gross Margin). The matrix includes totals for each row and column. The 'Insert' tab is active in the ribbon. The 'Visualizations' pane on the right lists various chart types.

CountryName	Make	Color	SalePrice	Gross Margin
France	Bentley	Canary Yellow	£46,750	19106
		Total	£46,750	19106
		British Racing Green	£25,250	12475
	Triumph	Canary Yellow	£25,250	12475
		Total	£50,500	24950
	TVR	Blue	£44,000	2225
		Total	£44,000	2225
		£141,250	46281	
	Aston Martin	Blue	£141,250	105250
		British Racing Green	£181,250	52420
Germany	Jaguar	Canary Yellow	£284,440	109085
		Green	£108,990	38855
		Night Blue	£165,600	26000
		Red	£450,300	322490
		Silver	£155,380	88570
		Total	£1,487,210	742670
		Blue	£44,000	16000
		British Racing Green	£39,500	-19444
		Canary Yellow	£110,000	51200
		Dark Purple	£44,000	-19444
Germany	Rolls Royce	Red	£110,000	87000
		Silver	£46,750	18987
		Total	£394,250	133799
		Black	£84,500	42136
		Canary Yellow	£88,000	11161
		Night Blue	£39,500	17200
		Total	£212,000	70497
		Black	£48,250	2540
		Blue	£72,000	51345
		Canary Yellow	£207,250	112440
Germany	Triumph	Night Blue	£45,800	30445
		Total	£373,300	196770
		Silver	£28,000	6600
		Total	£28,000	6600
		Silver	£29,750	-7025
		Total	£29,750	-7025
		£2,524,510	1143311	
	Jaguar	Green	£42,250	475
		Total	£42,250	475
		£2,524,500	1143314	

Figure 10-16. A matrix table

As you can see, a matrix display not only makes data easier to digest, but it automatically groups records by each element in the hierarchy and adds totals as well. What is more, each level in the hierarchy is sorted in ascending order.

You can also add fields directly to a table by dragging them onto the table. When you do this, Power BI Desktop always adds a text field to the *right* of existing fields. In a matrix, this means that any aggregate/numeric field is added to the right of existing aggregate fields (and appear in the Values box); whereas any text or date/time fields are added to the right of any existing hierarchy fields (and appear in the Rows box). However, it is always a simple matter to reorganize them by dragging the required fields up and down in the Rows and Values boxes to define the correct hierarchy and the overall type of display that you want to achieve.

When creating matrices, my personal preference is to drag the fields that constitute the hierarchy of non-numeric values into the Rows box, which means I am placing them accurately above, below, or between any existing elements. This ensures that your matrix looks right the first time, which can help you avoid some very disconcerting double takes!

Column Matrix

Power BI Desktop does not limit you to adding row-level hierarchies; you can also create column-level hierarchies, or mix the two. Suppose that we want to get a clear idea of sales and gross margin by country, make, and vehicle type, and how they impact one another. To achieve this, I suggest extending the matrix that you created previously by adding a VehicleType level as a column hierarchy.

Here is how you can do this:

1. Click inside the table that you created previously to select it. The Visualizations pane will display the fields that are used for this table in the Rows and Values boxes.
2. Drag the VehicleType field down into the Columns box in the field well in the Visualizations pane. This will add a hierarchy to the columns in the table. The table and the Visualizations pane will look like they do in Figure 10-17.

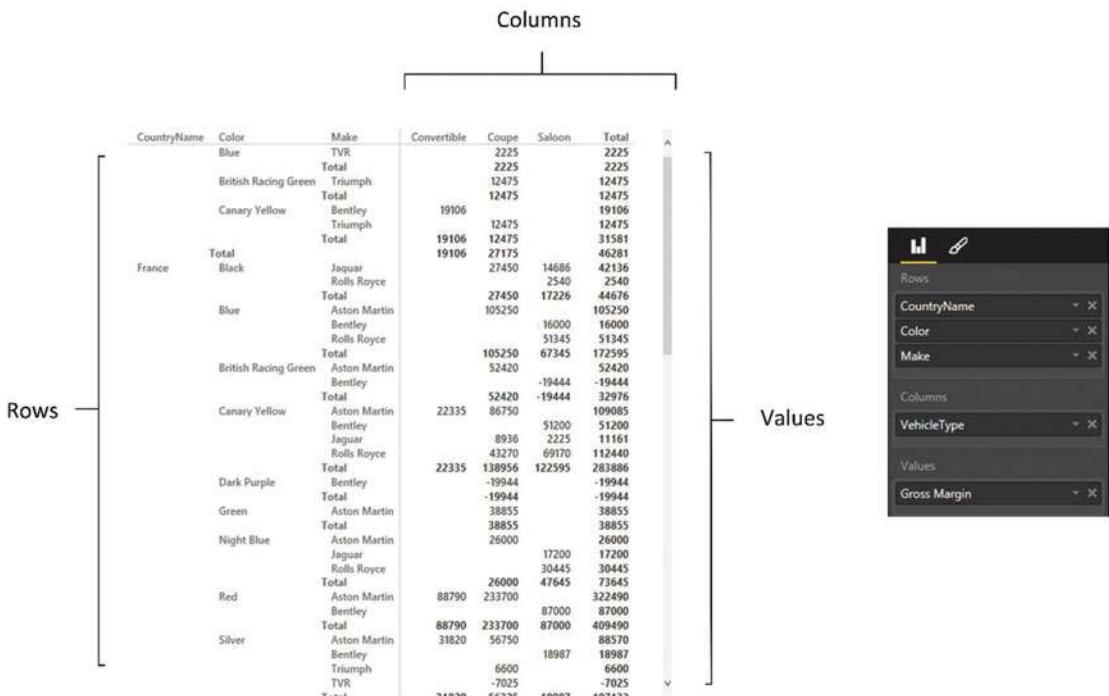


Figure 10-17. The Visualizations pane for a row and column matrix table

As you can see, you now have the sales and gross margin by country name, color, make and vehicle type, but it is in a cross-matrix, where the data is broken down by both rows and columns.

To conclude the section on creating matrices, there are a few things that you might like to note:

- If you add totals, then *every level* of the hierarchy has totals.
- Adding non-numeric data to the aggregated data makes Power BI Desktop display the Count aggregation.
- Matrices can get very wide, especially if you have a multilevel hierarchy. Power BI Desktop matrices reflect this in the way in which horizontal scrolling works. A matrix table freezes the non-aggregated data columns on the left and allows you to scroll to the right to display aggregated (numeric) data.
- Moving the fields in the field well of the Visualizations pane (using drag-and-drop, as described previously), reorders the aggregated data columns in the table.

Sorting Data in Matrices

When you sort data in a matrix table, the sort order will respect the matrix hierarchy. This means that if you sort on the second element in a hierarchy (Make, in the example table we just created) then the primary element in the hierarchy (CountryName, the leftmost column) will switch back to the initial (alphabetical) sort order as will any lower levels in the hierarchy of row elements.

If you sort by any value in a matrix, then the total for the highest level of the hierarchy is used to reorder the whole table. You can see this in Figure 10-18, where the matrix from the previous figure has been sorted on the total gross margin in descending order. This has made the country with the most sales move to the top of the table. As well, if you have a column matrix (as in this example), then you will end up sorting on the grand total of the columns (the two rightmost columns in this example) to make the matrix sort by numeric values, albeit in ascending order.

CountryName	Color	Make	Convertible	Coupe	Saloon	Total
Germany	Red	Jaguar		-9114		-9114
		Total		-9114		-9114
	Silver	TVR		-7025		-7025
		Total		-7025		-7025
	Green	Jaguar			475	475
		Total			475	475
	Blue	TVR		4475		4475
		Total		4475		4475
	Total			-11664	475	-11189
Spain	Red	Jaguar		-6864		-6864
		Total		-6864		-6864
	Blue	TVR		2225		2225
		Total		2225		2225
	Green	Jaguar			2725	2725
		Total			2725	2725
	Canary Yellow	Triumph		18550		18550
		Bentley	19106			19106
		Total	19106	18550		37656
	Total			19106	13911	2725
	Blue	TVR		2225		2225
		Total		2225		2225
	British Racing Green	Triumph		12475		12475
		Total		12475		12475
Switzerland	Canary Yellow	Triumph		12475		12475
		Bentley	19106			19106
		Total	19106	12475		31581
	Total			19106	27175	46281
	Dark Purple	Jaguar		4937		4937
		Rolls Royce			28195	28195
		Total		4937	28195	33132
	Green	Aston Martin		36605		36605
		Total		36605		36605
	Red	Jaguar			19687	19687
Silver		Aston Martin	69430			69430
		Total	69430	19687		89117
		TVR		-9275		-9275
		Aston Martin	41040	59000		100040
... ...		Total	41040	49725		90765
						44040

Figure 10-18. Sorting a matrix on values

Cards

On some occasions, you will want to display a single figure more prominently than others. Perhaps you need to attract the viewer's attention to your new sales record, or you would like the boss to appreciate the customer satisfaction figures that you have achieved.

Whatever the motivation, cards are the solution. Power BI Desktop cards are a simple and powerful way to isolate and emphasize a single figure. Suppose that you want to display all vehicle sales to date in a dashboard, for instance; the following explains how you can do it.

1. Using the DataModelWithMetricsForVisualizations.pbix file as your source of data, click outside any existing visualizations.
2. In the gallery of available visuals, click the Card icon. You can see this icon in Figure 10-19. An empty card will appear on the dashboard canvas.



Figure 10-19. The Card icon in the gallery of available visuals

3. Expand the InvoiceLines table fields in the Fields Pane and select the SalePrice field. The total for all sales will appear in the card.
4. Resize the card so that the figure fits inside the card borders without any wasted space. The final card will look like Figure 10-20.



Figure 10-20. A card visual

Formatting Cards

As you just saw, cards are an extremely effective way of focusing your audience's attention on key data. Yet there is a lot more that you can do to extend this emphasis, including the following:

- Change the units, number of decimals, color, and text size of the data that is displayed
- Remove the label that is displayed automatically, as well as alter its text size
- Add a title that you can format independently
- Change the background color of the card

Let's see how to tweak the card that you just created.

1. Select the card. In this example, I use the card that you saw in Figure 10-20.
2. In the Visualizations pane, click the Format icon (the small paintbrush beneath the palette of available visualization types). The formatting options for the selected card will be displayed.
3. Expand the Background tab by clicking the downward-facing chevron to the left of the word Background.
4. Make sure that the switch to the right of the word Background is set to On (if it is not, then click to the right of the empty circle to change this).

5. Click the pop-up menu triangle to the right of Color and select a color from the palette of available colors.
6. Click the Transparency slider switch and slide it to the left to intensify the background color by reducing the percentage of transparency.
7. Expand the Category Label tab by clicking the downward-facing chevron to the left of the word Category Label.
8. Adjust the text size by dragging the slider to the right.
9. Expand the Title tab by clicking the downward-facing chevron to the left of the word Title.
10. Set the title to On by clicking immediately to the right of the on/off button to the right of Title.
11. Enter **To Date** as the Title Text.
12. Click the pop-up menu triangle to the right of Font Color and select a color for the font from the palette of available colors.
13. Click the pop-up menu triangle to the right of Background Color and select a background color from the palette of available colors.
14. Click the middle of the Alignment icons to center the card title.
15. Expand the Data Label tab by clicking the downward-facing chevron to the left.
16. Click the pop-up menu triangle to the right of Color and select a data label color from the palette of available colors.
17. Adjust the text size by dragging the Text Size slider to the right or left.
18. Select a display unit for the figure in the card by clicking the pop-up list of Display Units.
19. Select the number of decimals to be used for the card data by clicking the up and down triangles to the right of the Precision box. You can also enter the number of decimals directly, if you prefer. The card will now look something like the one that you can see in Figure 10-21.

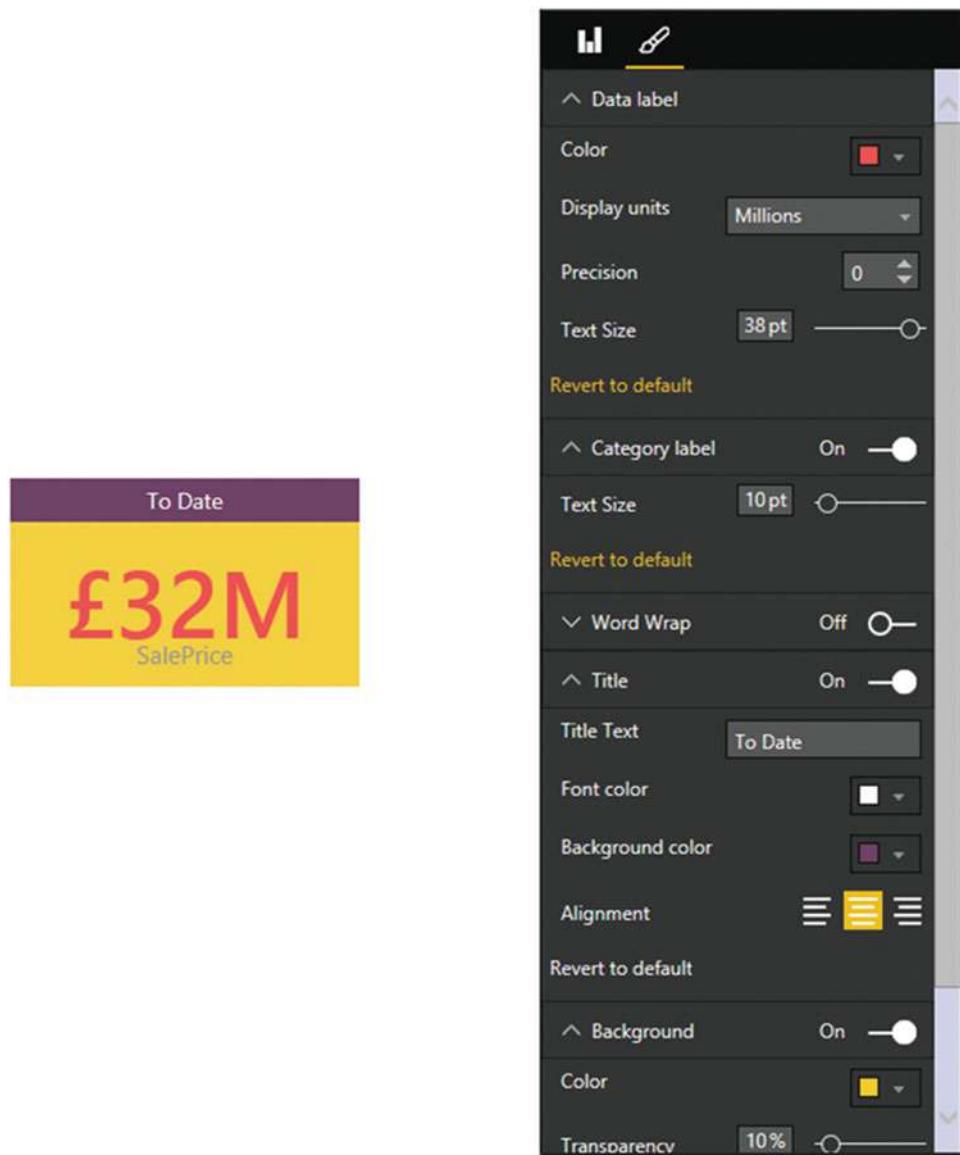


Figure 10-21. A formatted card

The available display units for formatting the value in a card are described in Table 10-2.

Table 10-2. Display Units

Display Units	Comments
Auto	Chooses an appropriate display unit from those available depending on the size of the figure
None	Displays the actual figure
Thousands	Displays the figure in thousands followed by a K
Millions	Displays the figure in millions followed by an M
Billions	Displays the figure in billions followed by a bn
Trillions	Displays the figure in trillions followed by a T

Multirow Cards

Tabular data can also be displayed in an extremely innovative way using the Power BI Desktop card style of output. As is the case with matrices, you begin by choosing the fields that you want to display as a basic table and then you convert this to another type of visual. Here is an example of how this can be done:

1. Using the DataModelWithMetricsForVisualizations.pbix file as your source of data, click outside any existing visuals.
2. In the Palette of available visuals, click the Card icon. You can see this in Figure 10-22. An empty multirow card will appear on the dashboard canvas.



Figure 10-22. The multirow card icon in the Palette of available visuals

3. Add the following fields, in this order:
 - a. CountryName (from the Countries table)
 - b. CostPrice (from the Stock table)
 - c. LaborCost (from the Stock table)
 - d. PartsCost (from the Stock table)
4. Resize the multirow card to display all the countries. The visual will look like the one in Figure 10-23. Don't worry about the (Blank) costs, these correspond to vehicles in stock that are not yet sold, and so do not have a client country yet.

(Blank)		
100,200.00 CostPrice	£1,461.00 LaborCost	3,770.00 PartsCost
France		
1,460,100.00 CostPrice	£36,471.00 LaborCost	42,430.00 PartsCost
Germany		
160,000.00 CostPrice	£1,461.00 LaborCost	1,600.00 PartsCost
Spain		
178,700.00 CostPrice	£2,272.00 LaborCost	4,420.00 PartsCost
Switzerland		
701,150.00 CostPrice	£27,207.00 LaborCost	30,315.00 PartsCost
United Kingdom		
10,193,880.00 CostPrice	£129,952.00 LaborCost	227,250.00 PartsCost
USA		
7,716,065.00 CostPrice	£113,135.00 LaborCost	189,045.00 PartsCost

Figure 10-23. A card visualization

Card-type tables will display the selected fields in the order in which they appear in the field well in the Visualizations pane, and it is here that they can be reordered, as with any table. This makes each card into a data record. The fields will flow left to right and then on to the following line in each card. What is interesting here is that adjusting the size of the table can change the appearance of the table quite radically. A very narrow table will list the fields vertically, one above the other. If you can fit all the fields onto a single row, then you will get a highly original multiple record displays.

Formatting Multirow Cards

There are a few visual aspects of multirow cards that you can tweak for added effect. These include adding a title and changing the background color. Let's try both of these.

1. Select the multirow card visualization. In this example, I use the card that you saw in Figure 10-23.
2. In the Visualizations pane, click the Format icon (the small paintbrush beneath the palette of available visual types). The formatting options for the selected multirow card are displayed.
3. Expand the Background section by clicking the downward-facing chevron to the left of the word *Background*.
4. Make sure that the switch to the right of the word *Background* is set to On (if it is not then click to the right of the empty circle to change this).
5. Click the pop-up menu triangle to the right of Color and select a color from the palette of available tones.
6. Click the Transparency slider switch and slide it to the left to intensify the background color by reducing the percentage of transparency.
7. Expand the Title tab by clicking the downward-facing chevron to the left of the word *Title*.
8. Set the title to On by clicking immediately to the right of the on/off button to the right of *Title*.
9. Enter **Key Figures By Country** as the Title Text.
10. Click the pop-up menu triangle to the right of Font Color and select a color for the font from the palette of available colors.
11. Click the pop-up menu triangle to the right of Background Color and select a background color from the palette of available colors.
12. Click the middle of the Alignment icons to center the card title.

The multirow card visual will now look like the one in Figure 10-24.

Key Figures By Country			
(Blank)			
100,200.00 CostPrice	£1,461.00 LaborCost	3,770.00 PartsCost	
France			
1,460,100.00 CostPrice	£36,471.00 LaborCost	42,430.00 PartsCost	
Germany			
160,000.00 CostPrice	£1,461.00 LaborCost	1,600.00 PartsCost	
Spain			
178,700.00 CostPrice	£2,272.00 LaborCost	4,420.00 PartsCost	
Switzerland			
701,150.00 CostPrice	£27,207.00 LaborCost	30,315.00 PartsCost	
United Kingdom			
10,193,880.00 CostPrice	£129,952.00 LaborCost	227,250.00 PartsCost	
USA			
7,716,065.00 CostPrice	£113,135.00 LaborCost	189,045.00 PartsCost	

Figure 10-24. A formatted multirow card

Note You can also fix the aspect ratio of a multirow card visual just as you did for the initial table at the start of this chapter.

Switching Between Table Types

One of the fabulous things about Power BI Desktop is that it is designed from the ground up to let you test ideas and experiment with ways of displaying your data quickly and easily. So, quite naturally, you can switch table types easily to see which style of presentation is best suited to your ideas and the message that you want to convey. To switch table types, all you have to do is select the current text-based visualization (table, multirow card, or matrix) and select one of these options from the gallery of available visuals:

- Table
- Matrix
- Multirow card

What is even more reassuring is that Power BI Desktop remembers the attributes of the previous table type you used. So, for instance, if you set up a matrix with a carefully crafted hierarchy and then switch to a card-type table, Power BI Desktop remembers how you set up the matrix should you want to switch back to it.

To see an example of how this works in practice, take a look at Figure 10-25. This shows the same initial visual that has been copied and then switched between the text-based core types of visual.

The figure displays three different ways to present the same data using text-based visualizations:

- Multi-Row Card:** On the left, this visualization shows data for various countries. Each country row contains three columns: CostPrice, LaborCost, and PartsCost. The first row is labeled '(Blank)'.
- Table:** In the center, this visualization presents the same data in a tabular format. It includes a header row with column names: CountryName, CostPrice, LaborCost, and PartsCost. Below this are several data rows for France, Germany, Spain, Switzerland, United Kingdom, and USA, followed by a 'Total' row.
- Matrix:** On the right, this visualization shows the data in a matrix format. It has a header row with column names: CountryName, CostPrice, LaborCost, and PartsCost. The data is presented in a grid structure where each cell contains the value from the corresponding row and column.

All three visualizations show the same underlying data:

	CountryName	CostPrice	LaborCost	PartsCost
(Blank)		100,200.00	£1,461.00	3,770.00
France	France	1,460,100.00	£16,471.00	42,430.00
Germany	Germany	160,000.00	£1,461.00	1,600.00
Spain	Spain	178,700.00	£2,272.00	4,420.00
Switzerland	Switzerland	701,150.00	£27,207.00	30,315.00
United Kingdom	United Kingdom	10,193,880.00	£129,952.00	227,250.00
USA	USA	7,716,065.00	£113,135.00	189,045.00
	Total	20,510,095.00	£311,959.00	498,830.00

Multi-Row Card

Figure 10-25. Switching visuals using the same data and formatting

The following are the main things to note here:

- The visual remains the same size when you switch types. Consequently, you will probably have to resize it.
- All formatting that can be retained is retained.

Conclusion

I hope that you are now comfortable with the Power BI Desktop interface and are relaxed about using it to present your data, whether you are using standard tables, matrices or the new and innovative card visuals that Power BI Desktop offers. Equally, I hope that you are at ease sorting your tables using the various available techniques.

This chapter is just a taster of the many ways in which Power BI Desktop can help you analyze and display the information that you want your audience to appreciate. Yet, as tables are the basis for just about every other form of visual, it is well worth mastering the techniques and tricks of table creation. This way, you are well on the way to a fluent mastery of Power BI Desktop, which lays the foundations for some truly impressive presentations.

Now that you have mastered the basics, it is time to move on to the next level of dashboard creation where you will discover how to create and enhance charts. This is the subject of the next chapter.

CHAPTER 11



Charts in Power BI Desktop

It is one thing to have a game-changing insight that can fundamentally alter the way your business works. It is quite another to be able to convince your colleagues of your vision. So what better way to show them—intuitively and instantaneously—that you are right than with a chart that irrefutably makes your point?

Power BI Desktop is predicated on the concept that a picture is worth many thousands of words. Its charting tools let you create clear and convincing visuals that tell your audience far more than a profusion of figures ever could. This chapter, therefore, will show you how simple it can be not just to make your data explain your analysis, but to make it seem to leap off the screen. You will see over the next few pages how a powerful chart can persuade your peers and bosses that your ideas and insights are the ones to follow.

A little more prosaically, Power BI Desktop lets you make a suitable dataset into

- Pie charts
- Bar charts
- Column charts
- Line charts
- Scatter charts
- Bubble charts
- Funnel charts
- Waterfall charts

In this chapter, we get up and running by looking at all these types of charts, and then extend them to create 100% stacked bar and stacked column charts, as well as dual-axis charts. Once you have decided upon the most appropriate chart type, you can then enhance your visualization with titles, data labels, and legends, where they are appropriate.

A First Chart

It is generally easier to appreciate the simplicity and power of Power BI Desktop by doing rather than talking. So I suggest leaping straight into creating a first chart straightaway. In this section, we will look only at “starter” charts that all share a common thread—they are based on a single column of data values and a single column of descriptive elements. The data will include:

- A list of clients
- Car sales for a given year

So, let's get charting! In this chapter, you will use the DataModelWithMetricsForVisualisations.pbix Power BI Desktop file that is available on the Apress web site for download.

Creating a First Chart

Any Power BI Desktop chart begins as a dataset. So, let me introduce you to the world of charts by showing you how to make a bar chart in a few clicks; the following explains how to begin:

1. Open the file C:\PowerBiDesktopSamples\ DataModelWithMetricsForVisualizations.pbix from the downloadable samples.
2. Click the Bar Chart icon at the top left of the Visualizations pane. This icon is illustrated in Figure 11-1. A large empty chart visual will appear on the dashboard canvas.

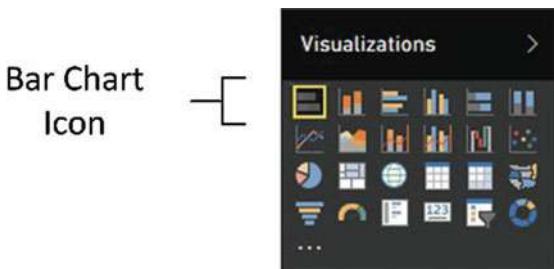


Figure 11-1. The Bar chart icon in the Visualizations pane

3. Leave the empty chart visual selected and expand the Clients table in the Fields list.
4. Select the box to the left of the ClientName field.
5. Leave this new chart visual selected. In the Fields list, expand the InvoiceLines table.
6. Click the check box to the left of the SalePrice field. This will add the cumulative sales per client to the chart.
7. Resize the chart (I suggest widening it) by dragging the handle in the middle of the right edge to the right until the axis labels are clearly visible, as shown in Figure 11-2.

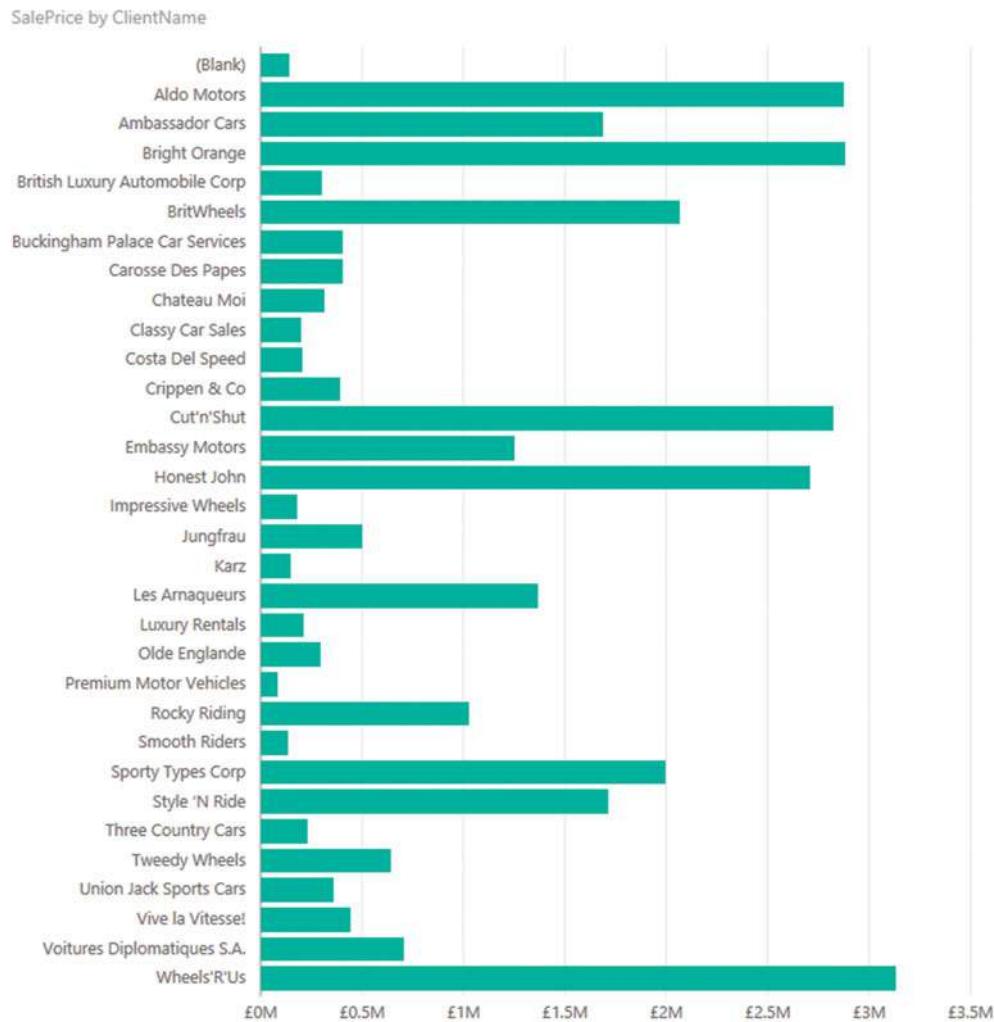


Figure 11-2. A bar chart after resizing

And that is all that there is to creating a simple starter chart. This process might only take a few seconds, and once it is complete, it is ready to show to your audience, or be remodeled to suit your requirements.

Nonetheless, a few comments are necessary to clarify the basics of chart creation in Power BI Desktop:

- First, when creating the chart, you can use any of the techniques described in Chapter 10 to add fields. You can drag fields into the Fields Well of the Visualizations pane or onto the visual directly if you prefer.
- Second, when you transform a table into a chart, the field well of the Visualizations pane changes to reflect the options available when creating or modifying a chart. If you select the chart that you just created, you will see that the ClientName field has been placed in the Axis box, and the SalePrice field has been placed in the Value box. Neither of these boxes exists if the visual is a table. This can be seen in Figure 11-3.

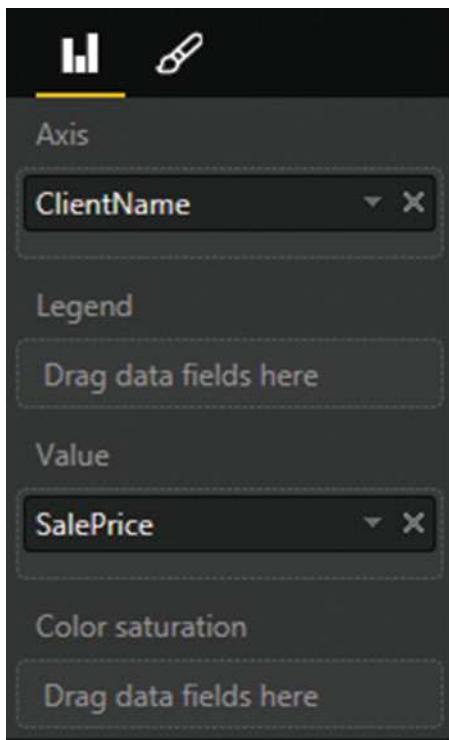


Figure 11-3. The Fields list for a bar chart

- Third, when using only a single dataset, you can choose either clustered or stacked as the chart type for a bar or column chart; the result is the same in either case. As you will see as we progress, this will not be the case when the chart is based on multiple data fields.
- Fourth, Power BI Desktop will add a title at the top left of the chart explaining what data the chart is based on. You can see an example of this in Figure 11-2.
- Finally, creating a chart is very much a first step. You can do so much to enhance a chart and accentuate the insights that it can bring. However, all of this follows further on in this chapter and in the next one.

Note You can begin by creating a table (by dragging the data fields onto the Power BI desktop canvas without first clicking a chart icon, for instance). Then select the table and click a chart icon in the visualizations palette to convert the table into a chart.

Deleting a Chart

Deleting a chart is as simple as deleting a table. All that you have to do is

1. Click inside the chart.
2. Press the Delete key.

If you remove all the fields from the Layout section of the field well of the Visualizations pane (with the chart selected), then you will also delete the chart.

Basic Chart Modification

So you have an initial chart. Suppose, however, that you want to change the actual data on which the chart is based. Well, all you have to do to change both the axis elements (the client names and the values represented) is

1. Click on or inside the chart that you created previously. Avoid clicking any of the bars in the chart for the moment.
2. In the field well of the Visualizations pane, click the pop-up menu for SalePrice in the Values box, and select Remove Field. The bars will disappear from the chart. Alternatively, you can click the small cross to the right of SalePrice to delete the field.
3. Drag the field GrossMargin from the SalesData table in the Fields list into the Values box of the field well.
4. In the field well of the Visualizations pane, click the pop-up menu for ClientName in the Axis box, and select Remove Field. The client names will disappear from the chart and a single bar will appear.
5. In the Fields list, expand the Colors table and drag the Color field from the Colors table into the Axis box. The list of colors will replace the list of clients on the axis, and a series of bars will replace the single bar. Look at Figure 11-4 to see the difference.

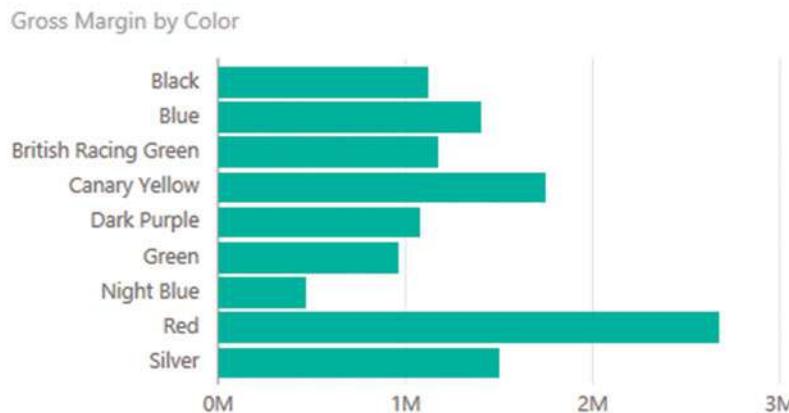


Figure 11-4. A simple bar chart with the corresponding layout section

That is it. You have changed the chart completely without rebuilding it. Power BI Desktop has updated the data in the chart and the chart title to reflect your changes.

Basic Chart Types

When dealing with a single set of values, you will probably be using the following six core chart types:

- Bar chart
- Column chart
- Line chart
- Pie chart
- Donut chart
- Funnel chart

Let's see how we can try out these types of chart using the current dataset—the colors and Gross Margin that you applied previously.

Column Charts

A column chart is, to all intents and purposes, a bar chart where the bars are vertical rather than horizontal. So, do the following to switch your bar chart to a column chart.

1. Click on or inside the bar chart that you created previously. Avoid clicking any of the bars in the chart for the moment.
2. In the Palette of possible visual in the Visualizations pane, click the Column Chart icon.
3. Resize the chart as required. Your chart should look like Figure 11-5.

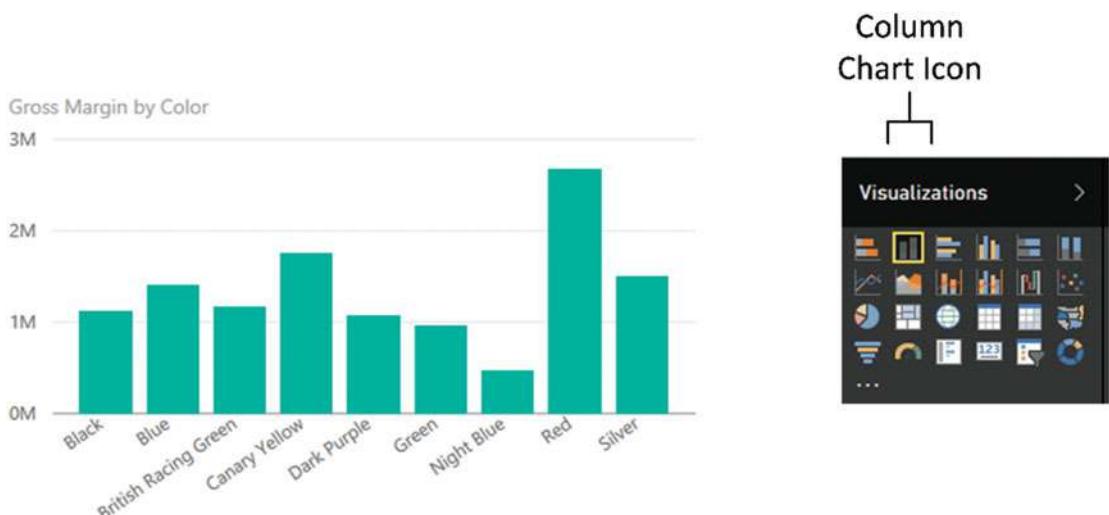


Figure 11-5. An elementary column chart

Line Charts

A line chart displays the data as a set of points joined by a line. Do the following to switch your column chart to a line chart.

1. Click the column chart that you created previously. Avoid clicking any of the columns in the chart for the moment.
2. In the gallery of available visualizations in the Visualizations pane, click the Line Chart icon. Your chart should look like Figure 11-6.

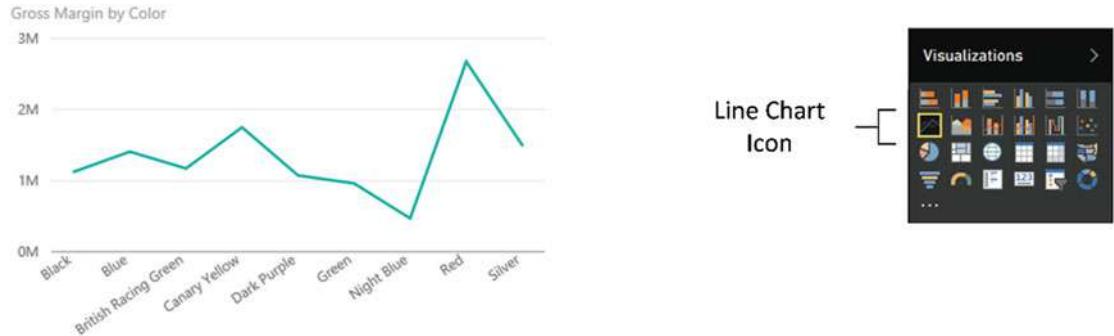


Figure 11-6. A simple line chart

Pie Charts

Pie charts can be superb at displaying a limited set of data for a single series, like we have in this example. To switch the visual to a pie chart

1. Click on, or inside, the line chart that you created previously. Avoid clicking the line in the chart for the moment.
2. In the Palette of available visual in the Visualizations pane, click the Pie Chart icon. Your chart should look like Figure 11-7. You will notice that the Layout section has changed slightly for a pie chart, and the Axis box has been replaced by a Legend box.

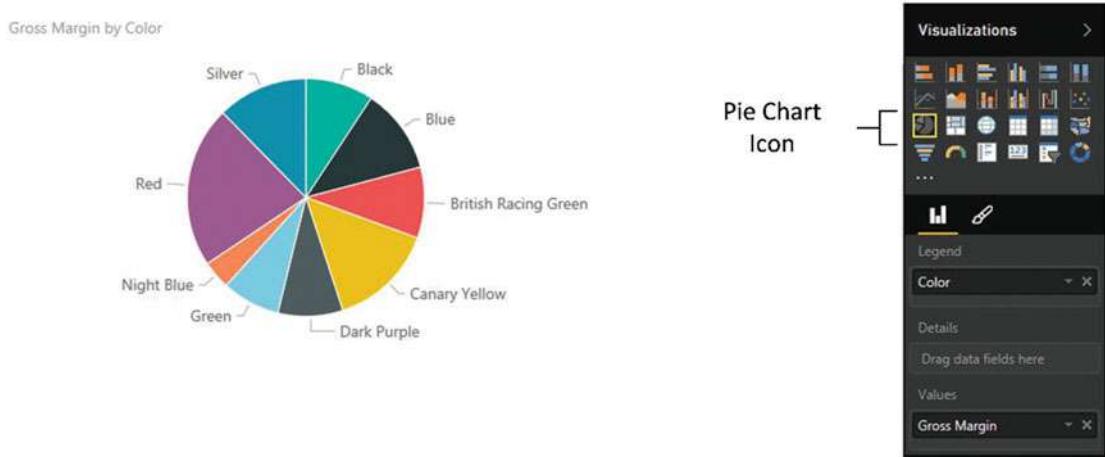


Figure 11-7. A basic pie chart

A pie chart is distorted if it includes negative values at the same time as it contains positive values. Power BI Desktop will not display the negative values. If your dataset contains a mix of positive and negative data, then Power BI Desktop displays an alert above the chart warning “Too many values. Not showing all data.” What this nearly always really means is that the pie chart contains positive *and* negative values, and that the negative values are not displayed. This also applies to donut charts. Negative values can make funnel charts appear a little peculiar, too. If you want to see this for yourself, try creating a table of makes and DeltaToAvgNetProfit. You will see that the table contains seven rows, but the corresponding pie or donut chart only displays three sections.

In practice, you may prefer not to use pie or donut charts when your data contains negative values, or you may want to separate out the positive and negative values into two datasets and display two charts, using filters (this is explained in Chapter 13).

Note Juggling chart size and font size to fit in all the elements and axis and/or legend labels can be tricky. One useful trick is to prepare “abbreviated” data fields in the source data, as has been done in the case of the QuarterAbbr field in the Date table that contains Q1, Q2, and so on, rather than Quarter 1, Quarter 2, and so on, to save space in the chart. Techniques for this sort of data preparation are given in Chapter 9.

Essential Chart Adjustments

Creating a chart in Power BI Desktop is extremely simple. I hope you will agree. Yet the process of producing a telling visualization does not stop when you take a table of data and switch it into a chart. At the very least, you want to make the following tweaks to your new chart:

- Resize the chart.
- Reposition the chart.
- Sort the elements in the chart.
- Alter the size of the fonts in the chart.

None of these tasks is at all difficult. Indeed, it can take only a few seconds to transform your initial chart into a compelling visual argument—when you know the techniques to apply.

Resizing Charts

A chart is like any other visual on the Power BI Desktop report and can be resized to suit your requirements. The following explains how to resize a chart.

1. Click inside the chart (but not on any of the bars, columns, lines, or pie segments).
2. Place the mouse pointer over any of the eight handles that appear at the corners and in the middle of the edges of the chart that you wish to adjust. The pointer becomes a two-headed arrow.
3. Drag the mouse pointer to resize the chart.

Note Remember that the lateral handles let you resize the chart only horizontally or vertically, and that the corner handles allow you to resize both horizontally and vertically.

Resizing a chart can have a dramatic effect on the text that appears on an axis. Power BI Desktop always tries to keep the space available for the text on an axis proportionate to the size of the whole chart.

For column, line, and bar charts, this can mean that the text can be truncated, with an ellipsis (three dots) indicating that not all the text is visible. With bar charts, the text may be angled at 45 degrees or even swiveled to appear vertically.

If you reduce the height (for a bar chart) or the width (for a column or a line chart) below a certain threshold, Power BI Desktop will stop trying to show all the elements on the non-numeric axis. Instead, it only shows a few elements and adds a scroll bar to allow you to scroll through the remaining data. You can see an example of this for a bar chart in Figure 11-8.

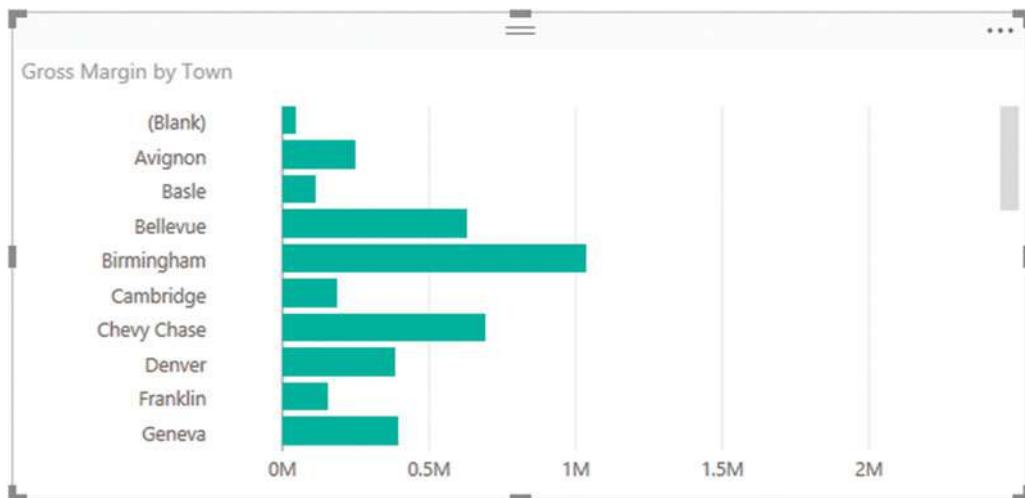


Figure 11-8. A chart with a scroll bar visible

All that this means is you might have to tweak the size and height to width ratio of your chart until you get the best result. If you are in a hurry to get this right, I advise using the handle in the bottom right corner to resize a chart, as dragging this up, down, left, and right this will quickly show you the available display options.

Repositioning Charts

You can move a chart anywhere inside the Power BI Desktop report.

1. Place the mouse pointer over the border of the chart. The pointer changes into a hand.
2. Drag the mouse pointer.

Sorting Chart Elements

Sometimes you can really make a point about data by changing the order in which you have it appear in a chart. Up until this point you have probably noticed that when you create a chart, the elements on the axis (and this is true for a bar chart, column chart, line chart, or pie chart) are in alphabetical order by default. If you want to confirm this, then just look at Figures 11-5 to 11-8.

Suppose now, for instance, you want to show the way that the gross margin is affected by the color of the vehicle. In this case, you want to sort the data in a chart from highest to lowest so that you can see the way in which the figures fall, or rise, in a clear order. The following explains how to do this.

Select the bar chart type for Color and Gross Margin, as described earlier (and shown in Figure 11-4).

1. Place the mouse pointer over the chart. You will see that the chart border and options button (the ellipses at the top right) appear.
2. Click the ellipses to display the chart menu and then click the chevron to the right of Sort By. This is shown in Figure 11-9.

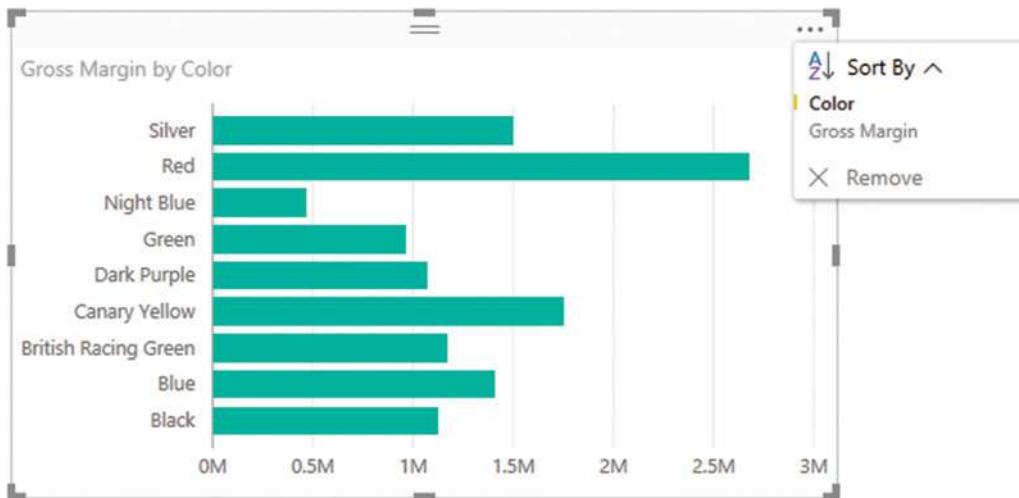


Figure 11-9. The sort area in a chart

- In the pop-up menu, click the word GrossMargin. This will change the sort order of the elements in the chart. You will now see a chart looking like the one in Figure 11-10.

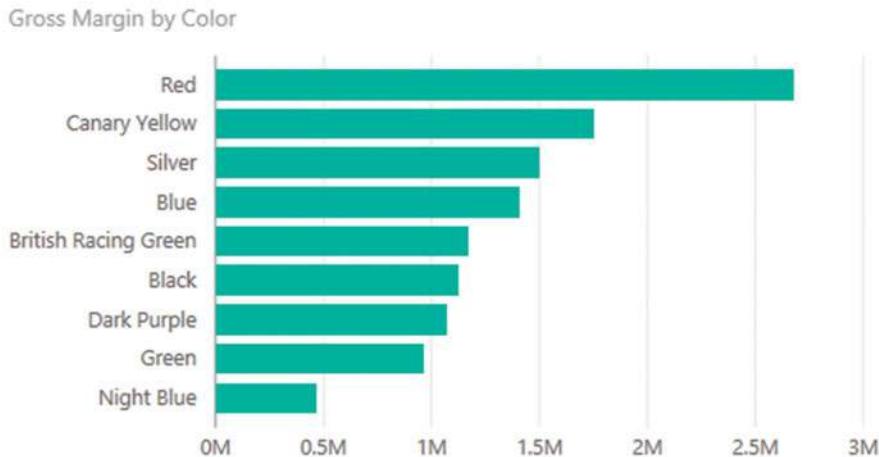


Figure 11-10. Sorting data in a bar chart

As you can see, the initial sort is in descending order when you sort by a numeric value. Let's suppose now that you want to see the sales by color in *ascending* order. All you have to do is to repeat the operation that you just carried out and the chart will be resorted. Only this time, it is sorted in ascending order. Equally, the *first* time that you sort on a *text* element the chart is sorted in *ascending* order and the *second* time it is sorted in *descending* order.

I should add just a short remark about sorting pie charts. When you sort a pie chart by a numeric field, the pie chart is sorted clockwise, starting at the top of the chart. So if you are sorting colors by GrossMargin in descending order, the top selling color is at the top of the pie chart (at 12 o'clock), with the second bestselling color to its immediate right (2 o'clock, for example) and so on. An example of this is shown in Figure 11-11.

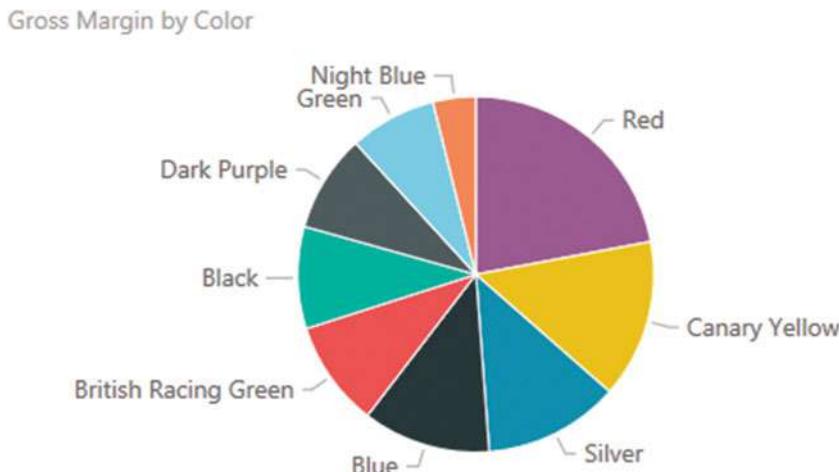


Figure 11-11. Sorting data in a pie chart

Donut Charts

Donut charts are essentially a variation of a pie chart. However, they can make a welcome presentational change from their over-exposed older sibling. Fortunately, they are equally easy to create. In this example, you are going to see how to visualize the parts cost incurred for each make of vehicle purchased.

1. Continue using (or open) the file C:\PowerBiDesktopSamples\DataAdapterWithMetricsForVisualizations.pbix.
2. Ensure that no current visual is selected.
3. Expand the Stock table in the Fields list.
4. Select the box to the left of the Make field. A table containing the list of makes purchased will appear in the dashboard canvas.
5. Leave this new table selected and click the check box to the left of the PartsCost field. This will add the cumulative cost of parts per make to the table.
6. Click the Donut icon in the Visualizations palette. This will convert the table to a donut chart.
7. Resize the chart if necessary to show the names of the makes clearly, as shown in Figure 11-12.



Figure 11-12. A donut chart

Funnel Charts

Funnel charts are excellent when it comes to comparing the relative values of a single series of figures. As you are now well-versed in the art of creating charts with Power BI Desktop, designing a funnel chart that displays parts cost per color should not be a problem for you.

1. Carry on using the file C:\PowerBiDesktopSamples\DataAdapterWithMetricsForVisualizations.pbix.
2. Expand the Colors table in the Fields list.
3. Drag the Color field onto an empty part of the dashboard canvas. A table containing the list of vehicle colors will appear.

4. Leave this new table selected and click the check box to the left of the PartsCost field in the Stock table. This will add the cumulative cost of parts per make to the table.
5. Click the Funnel icon in the Visualizations palette. This will convert the table to a donut chart.
6. Place the mouse pointer over the chart. You will see that the chart border and options button (the ellipses at the top right) appear.
7. Click the ellipses to display the chart menu and then click the chevron to the right of Sort By.
8. Select PartsCost. The funnel chart will display the color with the highest value for the cost of spare parts at the top of the visual.
9. Resize the chart if necessary to show the names of the colors clearly as well as making the relative weights for all the colors more clearly comprehensible, as shown in Figure 11-13.

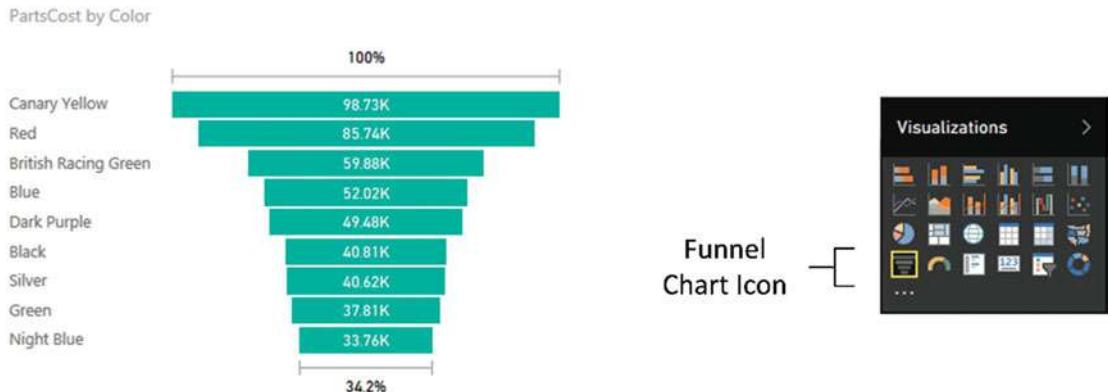


Figure 11-13. A funnel chart

Multiple Data Values in Charts

So far in this chapter, we have seen simple charts that display a single value. Life is, unfortunately, rarely that simple, and so it is time to move on to slightly more complex, but possibly more realistic, scenarios where you need to compare and contrast multiple data elements.

For this set of examples, I will presume that we need to take an in-depth look at the indirect cost elements of our car sales to date:

- Parts
- Labor

Consequently, to begin with a fairly simple comparison of these indirect costs, let's start with a clustered column chart:

1. Open the file C:\PowerBiDesktopSamples\DataModelWithMetricsForVisualizations.pbix from the downloadable samples.
2. Starting with a clean Power BI Desktop report, create a table that displays the following fields:
 - a. ClientName (from the Clients table)
 - b. PartsCost (from the Stock table)
 - c. LaborCost (from the Stock table)
3. Leaving the table selected, click the Clustered Bar icon in the Palette of chart types in the Visualization pane.
4. Resize the chart to make it clear and comprehensible, as shown in Figure 11-14. (I have included the fields from the Visualizations pane so that you can see this, too.)

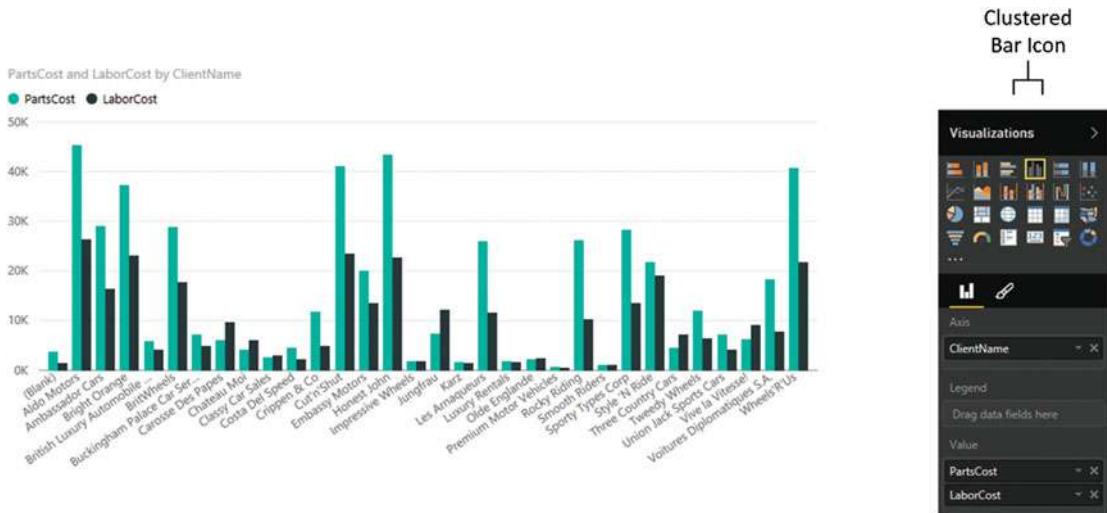


Figure 11-14. Multiple data values in charts—a clustered bar chart with the layout section shown

You will notice that a chart with multiple datasets has a legend by default, and that the automatic chart title now says PartsCost And LaborCost By ClientName.

The same dataset can be used as a basis for other charts that can effectively display multiple data values:

- Stacked bar
- Clustered column and stacked column
- Line charts
- Area charts

Since column charts are essentially bar charts pivoted 90 degrees, I will not show examples. However, in Figures 11-15, 11-16, and 11-17, you will see examples of a stacked bar chart, a line chart, and an area chart. You also see that when creating these types of visualization, the Layout section of the field well of the Visualizations pane remains the same for all of these charts.

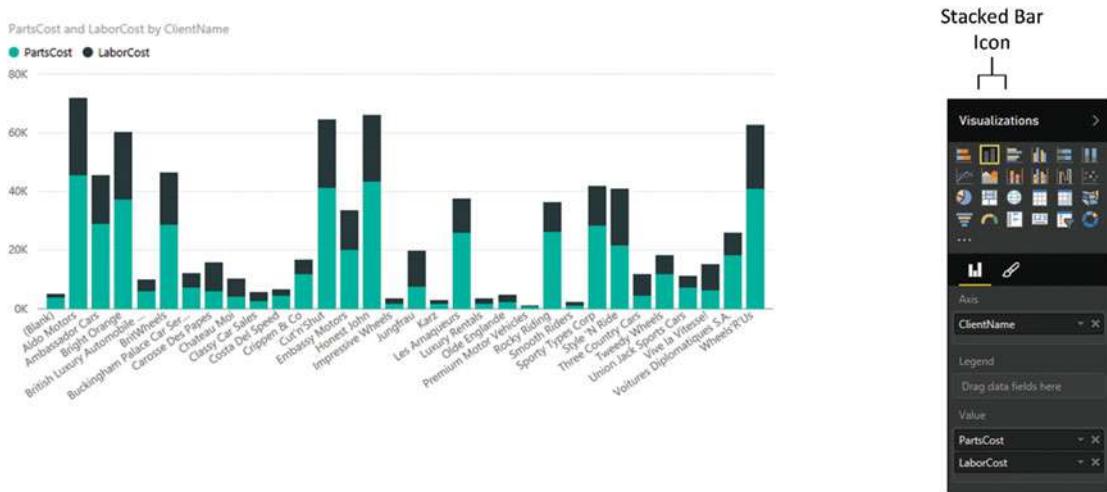


Figure 11-15. A simple stacked bar chart

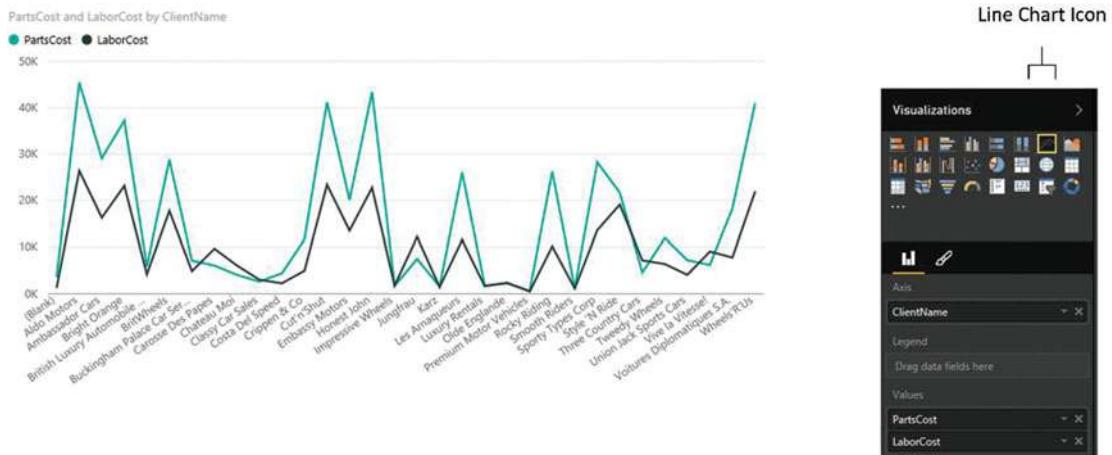


Figure 11-16. A line chart that displays multiple values

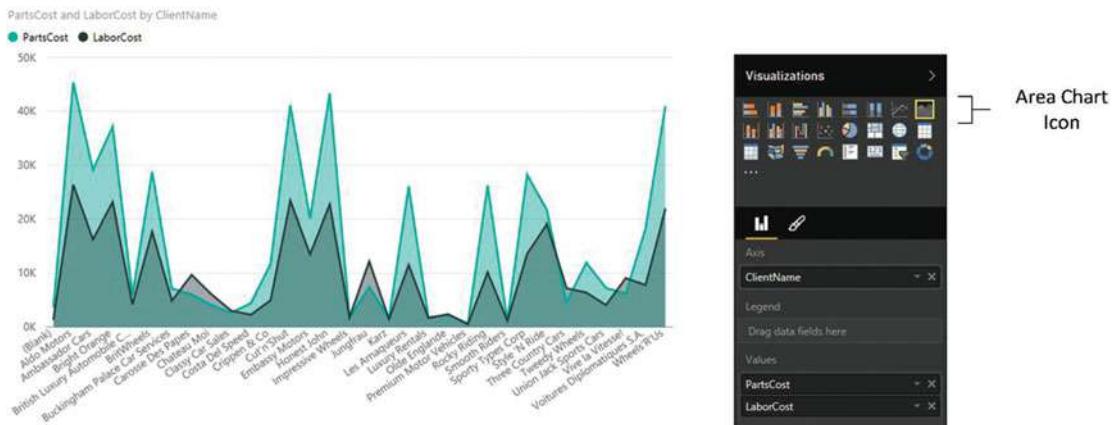


Figure 11-17. An area chart displaying multiple values

These four charts all display the same data in different ways. Knowing this, you can choose the type of chart that best conveys the information and draws your reader's attention to the point that you are trying to make.

If you are not sure which chart best conveys your message, then try them all, one after another. All it takes is a single click the relevant icon in the visuals gallery to convert a chart to another chart type.

Note Any of these chart types can be used to display a single data series if you want to. It all depends on the effect and the clarity of the insights that you are projecting using the chart.

100% Stacked Column and Bar Charts

One way to compare data from multiple datasets is to present each individual data series as a percentage of the total. Power BI Desktop includes two chart types that can do this “out of the box.” They are the 100% stacked column and 100% stacked bar charts.

Since you now know how to create charts that use multiple series of data, I will not explain how to produce these two chart types in detail; all you have to do is select the correct chart icon from the palette of available visualizations. So instead, I suggest that you look at Figures 11-18 and 11-19, which show an example of each of these charts using the same data that you used to create the clustered bar chart shown in Figure 11-15.

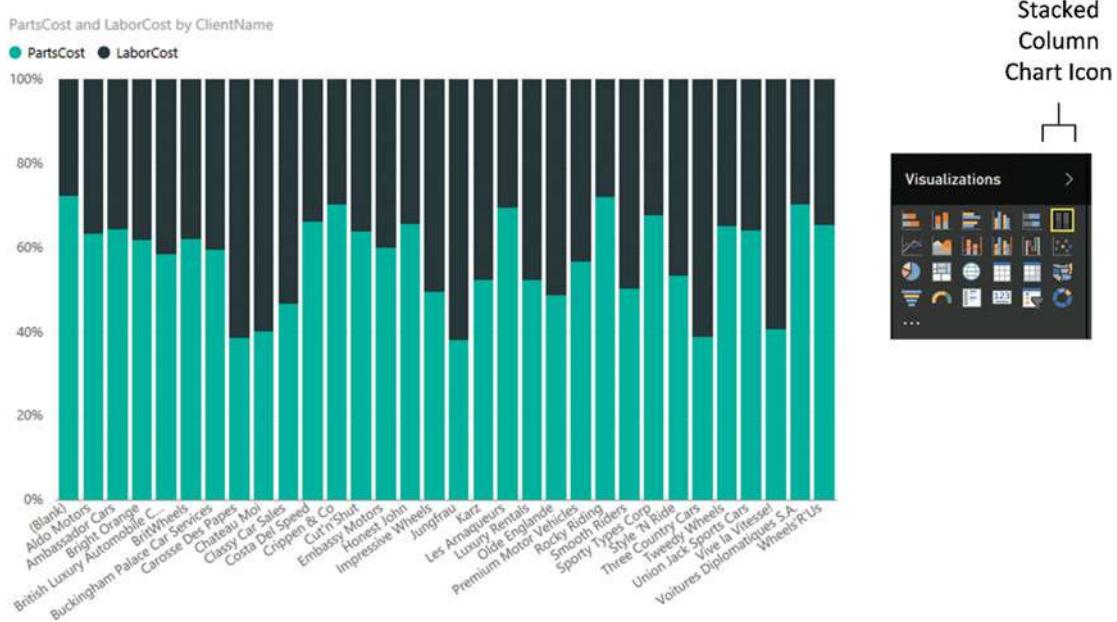


Figure 11-18. A 100% stacked column chart

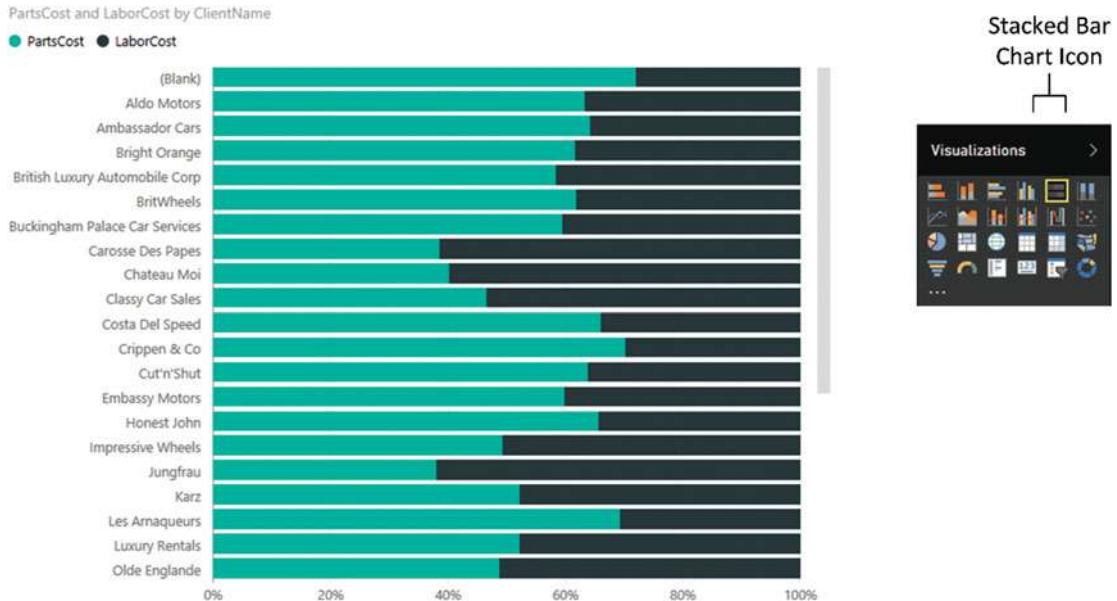


Figure 11-19. A 100% stacked column chart

Scatter Charts

A scatter chart is a plot of data values against two numeric axes, and so by definition, you need two sets of numeric data to create a scatter chart. To appreciate the use of these charts, let's imagine that you want to see the sales and margin for all the makes and models of car you sold overall. Hopefully, this allows you to see where you really made money. The following explains how to do it.

1. Using the file C:\PowerBiDesktopSamples\DataModelWithMetricsForVisualizations.pbix, delete any existing visualizations.
2. Create a table with the following fields in this order:
 - a. Vehicle (from the Stock table)
 - b. Net Margin Ratio (from the InvoiceLines table)
 - c. DirectCosts (from the InvoiceLines table)
3. Convert the table to a scatter chart by clicking the Scatter Chart icon in the palette of available visualizations. Power BI Desktop will display a scatter chart that looks like the one shown in Figure 11-20. Resize the chart to suit your taste.

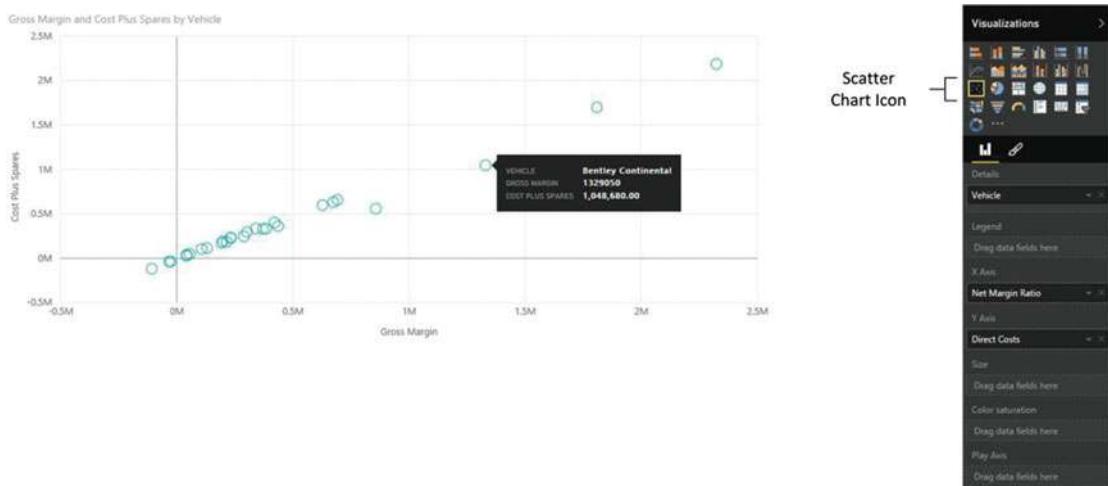


Figure 11-20. A scatter chart

If you look at the Fields area of the Visualizations pane (which is also shown in Figure 11-20), you will see that Power BI Desktop has used the fields that you selected like this:

- Vehicle: Placed in the Details box.
- GrossMargin: Placed in the X Axis box. This is the vertical axis.
- CostPlusSpares: Placed in the Y Axis box. This is the horizontal axis.

If you hover the mouse pointer over one of the points in the scatter chart, you see the data for the specific car model.

Note By definition, a scatter chart requires numeric values for both the X and Y axes. So if you add a non-numeric value to either the X Value or Y Value boxes, then Power BI Desktop converts the data to a count aggregation.

We made this chart by adding all the required fields to the initial table first. We also made sure that we added them in the right order so the scatter chart would display correctly the first time. In the real world of interactive data visualization, things may not be quite this coherent, so it is good to know that Power BI Desktop is very forgiving. And, it lets you build a scatter chart (just like any other chart) step by step if you prefer. In practice, this means that you can start with a table containing just two of the three fields that are required at a minimum for a scatter chart, convert the table to a scatter chart, and then add the remaining data field. Power BI Desktop always attributes numeric or time fields to the X and Y axes (in the order in which they appear in the Fields box) and place the first descriptive field into the Details box.

Once a scatter chart has been created, you can swap the fields around and replace existing fields with other fields from the tables in the data to your heart's content.

Scatter Charts to Display Flattened Hierarchies

Scatter charts generically are designed to show many, many data points. This makes them useful in, paradoxically, avoiding the need to drill down through a predefined hierarchy of data. Let's see how to display multiple datasets on one level, rather than drilling down for them. For this to happen, the source data must lend itself to the type of analysis that is required. Fortunately, the source data has a field named Vehicle that combines the make and model of each car and that suits this kind of analysis. To do this, you have to do the following:

1. Using the file C:\PowerBiDesktopSamples\DataModelWithMetricsForVisualizations.pbix, delete any existing visualizations.
2. Add the following fields (in this order) to the Fields List Values box:
 - a. Vehicle
 - b. Color
 - c. GrossMargin
 - d. RatioNetMargin
3. Convert the table to a scatter chart by clicking the Scatter Chart icon.

As you can see in Figure 11-21, every data point appears multiple times, once for every time there is a sale for a different color. Placing the mouse pointer over a data point lets you see exactly which model of vehicle and color is being represented. As you have probably guessed, I tweaked the size and display of the chart to show the data in its best light.

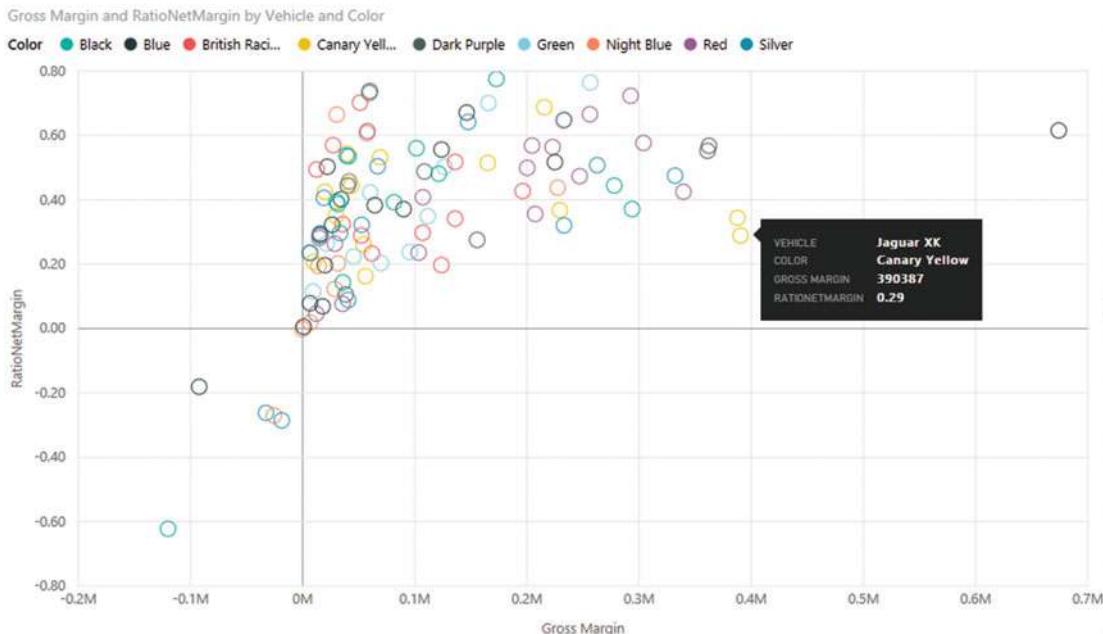


Figure 11-21. Flattened hierarchies in a scatter chart

With some scatter charts, this technique can make the chart hard to decipher. However, if your scatter chart contains relatively few data points, this technique can be useful. What is more, Power BI Desktop has the ability to highlight data by the elements that compose the legend; this is explained in Chapter 13.

Bubble Charts

A variant of the scatter chart is the bubble chart. This is one of my favorite chart types, though of course you cannot overuse it without losing some of its power. Essentially, a bubble chart is a scatter chart with a third piece of data included. So whereas a scatter chart shows you two pieces of data (one on the X axis, one on the Y axis), a bubble chart lets you add a third piece of information, which becomes the *size* of the point. Consequently, each point becomes a bubble.

The best way to appreciate a bubble chart is to create one. So here we assume that you want to look at the following for all makes of car sold in a single chart:

- The Total Sales
- The Net Margin Ratio
- The Gross Margin

This explains how a bubble chart can do this for you:

1. Using the file C:\PowerBiDesktopSamples\DataAdapterWithMetricsForVisualizations.pbix, delete any existing visualizations.
2. Create a table with the following fields from the SalesData table, in this order:
 - a. Make (from the Stock table)
 - b. Total Sales (from the InvoiceLines table)

- c. RatioNetMargin (from the InvoiceLines table)
 - d. GrossMargin (from the InvoiceLines table)
3. Convert the table to a scatter chart by clicking the Scatter Chart icon. Yes, a bubble chart is a scatter chart, with a fresh tweak added. Power BI Desktop will display a bubble chart that looks like that shown in Figure 11-22.

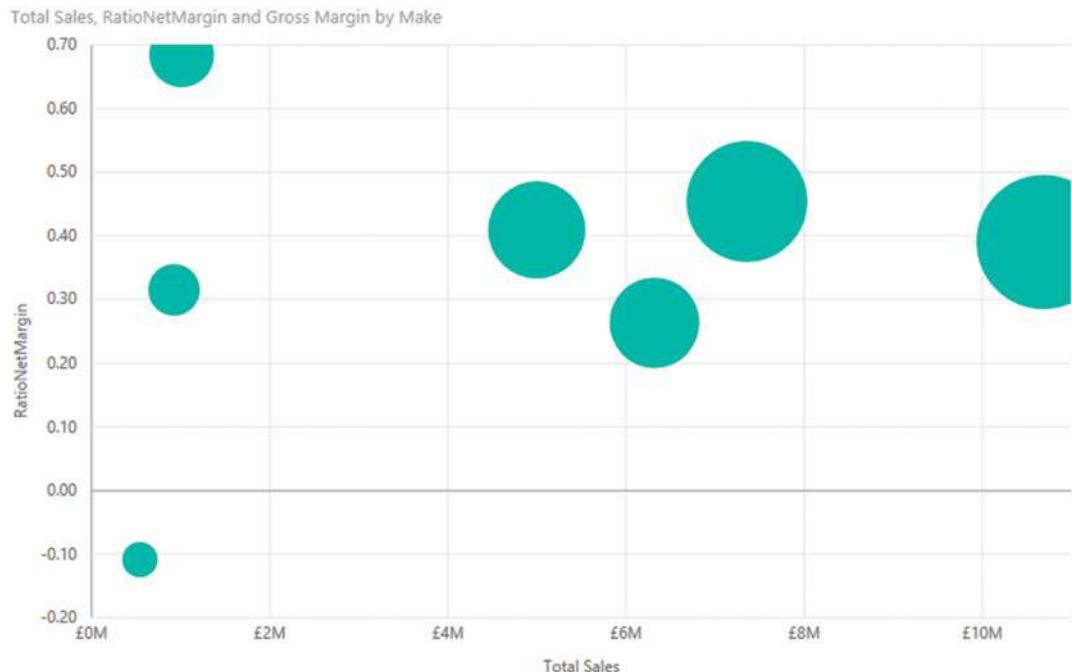


Figure 11-22. An initial bubble chart

4. Resize the chart if you need to.

If you look at the field well in the Visualizations pane (shown in Figure 11-23), you will see that Power BI Desktop has used the fields that you selected like this:

- *Make*: Placed in the Details box.
- *Total Sales*: Placed in the X Axis box. This is the vertical axis.
- *RatioNetMargin*: Placed in the Y Axis box. This is the horizontal axis.
- *GrossMargin*: Placed in the Size box. This defines the size of the points, which have consequently become bubbles.

Hover the mouse pointer over one of the points in the bubble chart. You will see all the data that you placed in the Fields list Layout section for each make, including the GrossMargin. You can see this in Figure 11-23.

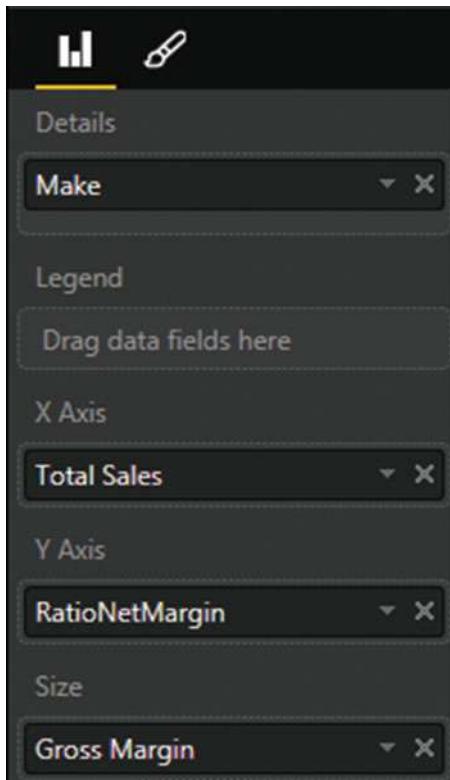


Figure 11-23. The Fields area of the Visualizations pane for a bubble chart

Waterfall Charts

We are very near the end of our tour of Power BI Desktop chart types. What I want to look at now is the penultimate chart type Power BI Desktop offers—the waterfall chart. This chart type is excellent at displaying the component parts of a final figure; in this example, it is used to show how the various makes sold make up the total sales figure.

1. Open the file C:\PowerBiDesktopSamples\DataAdapterWithMetricsForVisualizations.pbix and delete any existing visualizations.
2. Click the Waterfall chart icon. An empty waterfall chart will appear on the dashboard canvas.
3. Expand the Stock table and check the following fields:
 - a. Make
 - b. CostPrice
4. Resize the chart to suit your aesthetic requirements. It should look something like Figure 11-24.

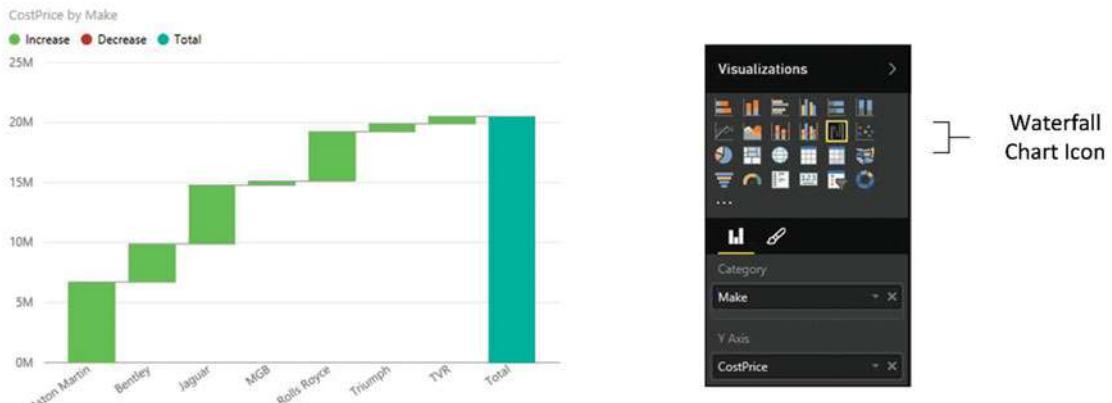


Figure 11-24. A waterfall chart

Waterfall charts can be extremely useful when you are analyzing the constituent elements of a whole. For instance, you could try using one to break down all the cost elements of a sale.

Dual-Axis Charts

To conclude our tour of chart types, let's take a look at a couple of charts that combine two of the basic chart types that you have seen previously in this chapter:

- Line and clustered column chart
- Line and stacked column chart

Let's discuss each of these in turn.

Line and Clustered Column Chart

Suppose that the CEO of Brilliant British Cars has decided to embark on an analysis of vehicle purchases. He wants to isolate any interesting correlations that could influence purchases in order to maximize profits. So, determined to satisfy his request (or possibly to humor him) you have in turn decided to take a look at indirect costs and mileage to see if there are any correlations.

1. Open the file C:\PowerBiDesktopSamples\DataAdapterWithMetricsForVisualizations.pbix and delete any existing visual.
2. Click the line and clustered column chart icon. An empty line and clustered column chart will appear on the dashboard canvas.
3. Expand the Stock table and check the following fields:
 - a. Make (this will be added to the Shared Axis box)
 - b. LaborCost (this will be added to the Column Values box)
 - c. PartsCost (this will be added to the Column Values box)
4. Also from the Stock table drag the Mileage field into the Line Values box in the field well of the Visualizations pane.

5. Resize the chart to suit your aesthetic requirements. It should look something like Figure 11-25.

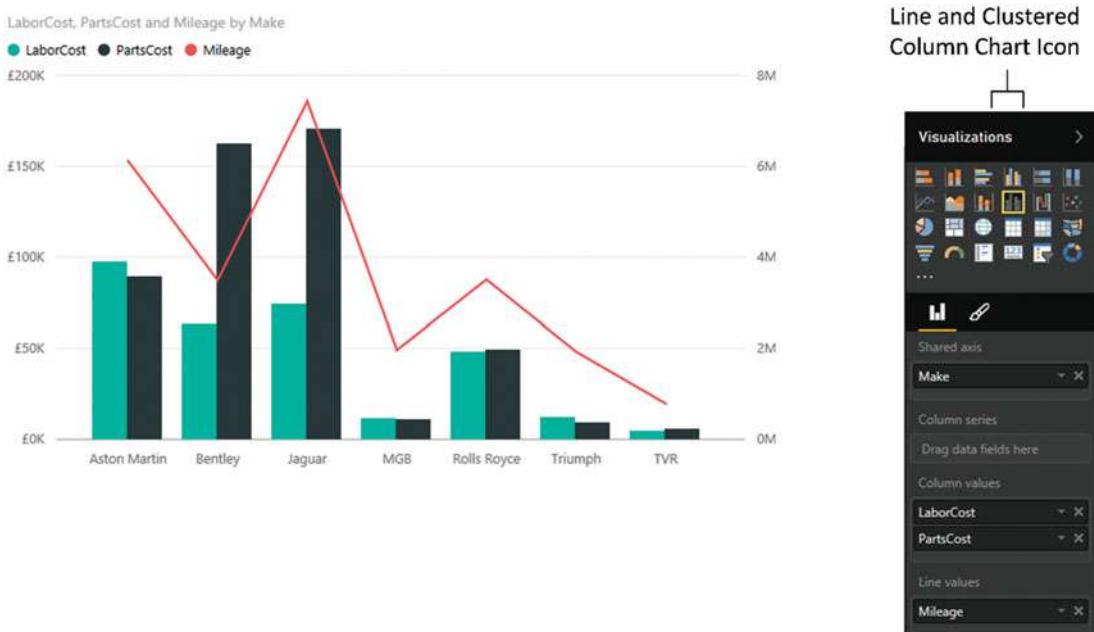


Figure 11-25. A line and clustered column chart

As you can see, this chart combines a clustered column chart that displays the labor and parts costs using the left axis with a line chart that displays the mileage using the right axis. Both of these charts share the X (or horizontal) axis.

Moreover, this kind of analysis makes it immediately clear that there is one make where low mileage does not necessarily mean lower repair costs. So the boss should be happy that you have isolated this unexpected correlation.

Line and Stacked Column Chart

One of the key advantages of dual-axis charts is that the two X axes can have vastly different scales. So for instance, when you want to see how the parts and labor cost relate to the purchase cost (which is orders of magnitude higher than the other two costs), a line and stacked column chart can be really useful to get a clearer view of the data.

1. Open the file C:\PowerBiDesktopSamples\DataMemberWithMetricsForVisualizations.pbix and delete any existing visualizations.
2. Expand the Stock table and check the following fields to display a table on the dashboard canvas:
 - a. Make
 - b. LaborCost
 - c. PartsCost

3. Click the line and stacked column chart icon. The table will be converted to a line and stacked column chart.
4. Also from the Stock table, drag the Mileage field into the Field section of the Visualizations pane. Place it in the Line Values box.
5. Resize the chart to suit your aesthetic requirements. It should look something like Figure 11-26.

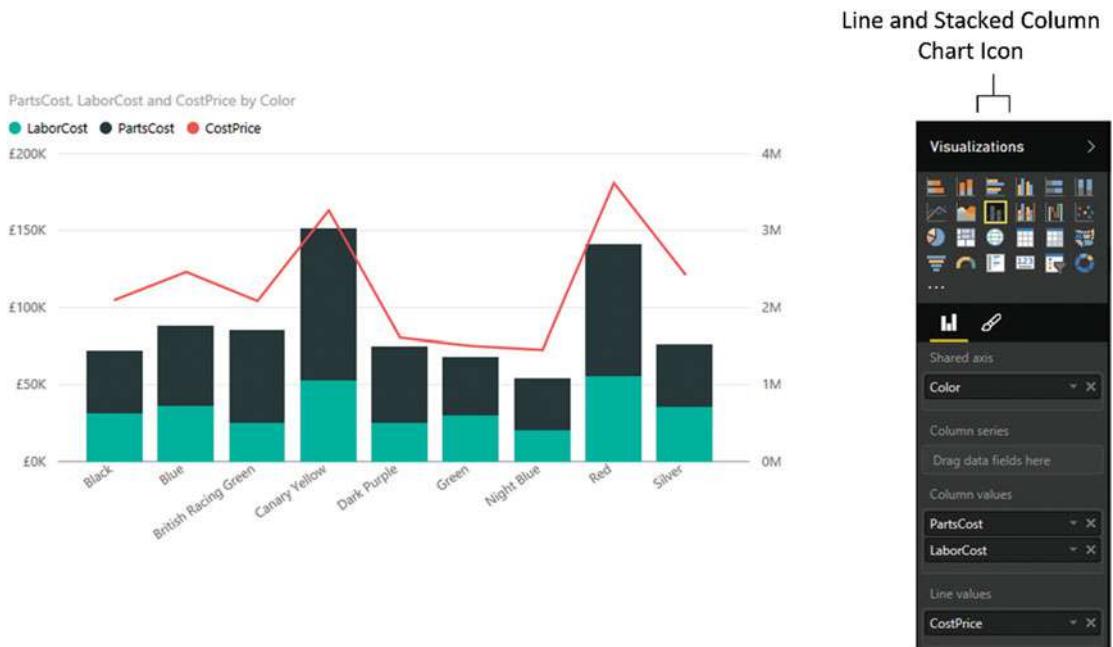


Figure 11-26. A line and stacked column chart

This way you can compare values where the use of a single chart type would lead to one value (the cost price in this example) would dwarf the other values to the point of making them unreadable.

Data Details

To conclude our tour of chart types, I just want to make a couple of comments.

- First, you can always see exactly what the figures behind a bar, column, line, point, or pie segment are just by hovering the mouse pointer over the bar (or column, or line, or pie segment). This works whether the chart is its normal size, or whether it has been popped out to cover the Power BI Desktop report area. Examples of this are given in Figures 11-19 and 11-20. However, this principle applies to any chart type equally well.

- Second, however much work you have done to a chart, you can always switch it back to a table if you want. Simply select the chart, and select the required table type from the Table button in the Design ribbon. If you do this, you see that the table attempts to mimic the design tweaks that you applied to the chart, keeping the font sizes the same as in the chart, and the size of the table identical to that of the chart. Should you subsequently switch back to the chart, then you should find virtually all of the design choices that you applied are still present—unless, of course, you made any changes to the table before switching back to the chart visualization.

Drill Down

An extremely useful aspect of Power BI Desktop charts is that you can use them as an interactive presentation tool. One of the more effective ways to both discover what your data reveals and to deliver the message to your public is to drill into layers of data. Certain Power BI chart types let you do just this. Here is a simple example of how you can analyze the hierarchy of makes, models, and colors of vehicles sold.

1. Click the Clustered Column Chart icon in the Visualizations palette. An empty chart visual will be created on the dashboard canvas.
2. Drag the following fields into the Axis area of the field well in this order:
 - a. Make (from the Stock table)
 - b. Model (from the Stock table)
 - c. Color (from the Colors table)
3. Drag the SalePrice field from the InvoiceLines table onto the chart. The chart will look like the one in Figure 11-27. As you can see, it only displays the top-level element from the Axis field well—the Make of vehicle.

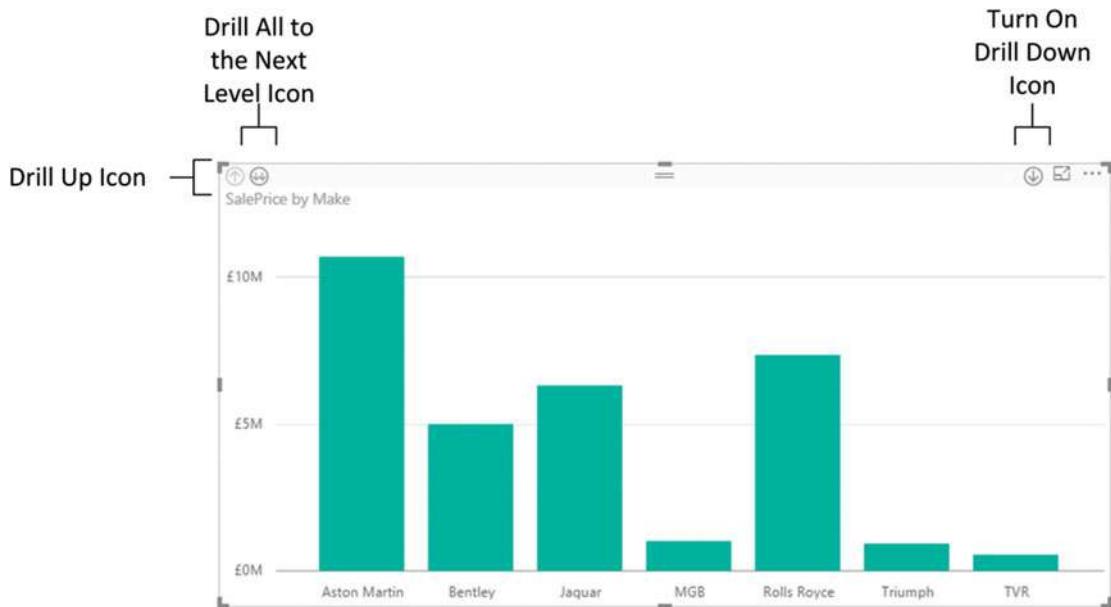


Figure 11-27. A column chart displaying the top level of data in a hierarchy

4. Click the Turn on Drill Down icon at the top right of the chart. This icon will become a white arrow on a dark background.
5. Click the column for Aston Martin. The chart will now display the makes of Aston Martin sold, as you can see in Figure 11-28.

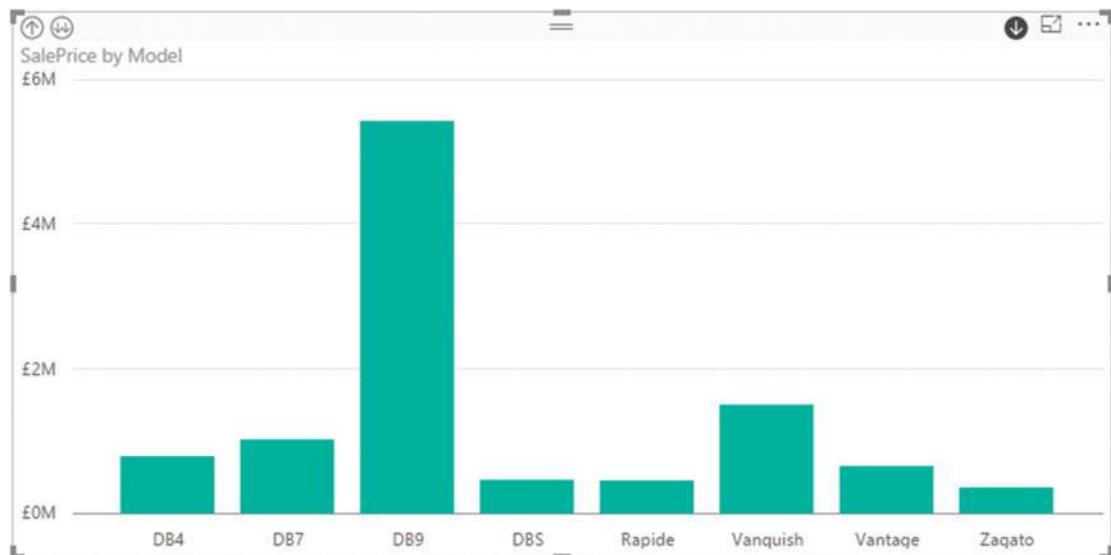


Figure 11-28. A column chart displaying the second level of data in a hierarchy

- Click the column for DB9. The chart will now display the colors of Aston Martin DB9 sold, as you can see in Figure 11-29.

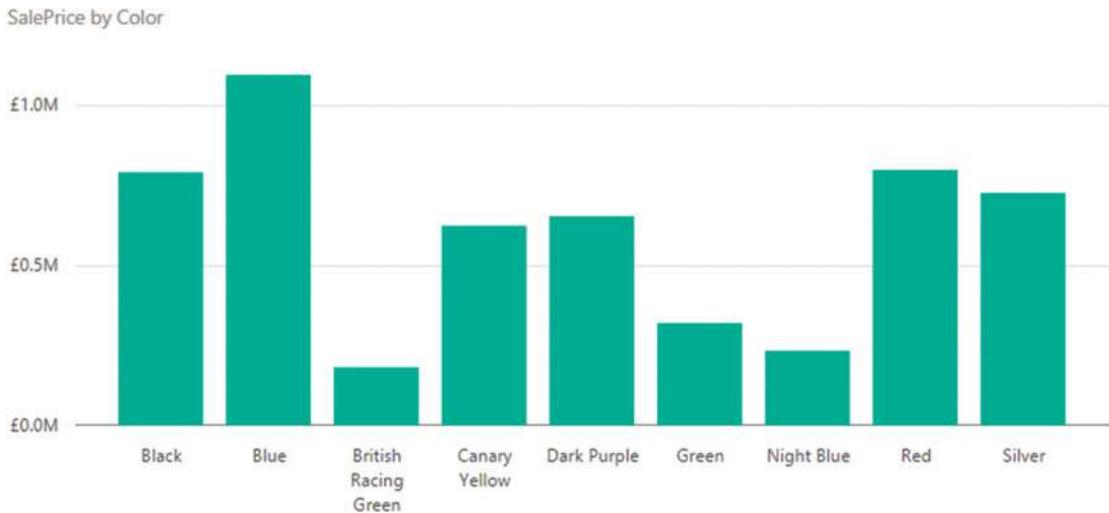


Figure 11-29. A column chart displaying the lowest level of data in a hierarchy

Once you have drilled down to a lower level in a chart you can always return to the previous level (and from there continue drilling up until you reach the top level) by clicking the Drill Up icon at the top left of the chart.

Note The Drill All to the Next Level icon produces a different result to clicking a bar in the chart. If you click the Drill All to the Next Level icon, you see *all* the elements at the next level down, not just the elements in the hierarchy that are a subgroup.

Drill down works with the following chart types:

- Clustered bar charts
- Clustered column charts
- Stacked bar charts
- Stacked column charts
- Double axis charts
- Waterfall charts
- Scatter charts
- Bubble charts
- Pie charts

- Funnel charts
- Donut charts
- Tree map charts

The key point to remember when you are creating drill-down charts is that you *must* place the fields that you wish to drill into in the axis area (the shared axis for double-axis charts, and the group area for tree map charts) of the field well, *not* in the legend area (and not in the details area for a pie chart).

Enhancing Charts

Now that you have been introduced to the Layout ribbon and you have mastered basic charts, it is time to move on to the next step and learn how to tweak your charts to the greatest effect. The next few sections are devoted to the various techniques available in Power BI Desktop to give your charts real clarity and power. Some of these enhancements apply to all chart types whereas others are specific to a single type of chart—or even one or two chart types.

Sometimes you change the configuration of a chart—and consequently enhance it—by altering the mapping of the data to the chart elements. However, on most occasions, you enhance a chart by modifying the various formatting options that are displayed when you select a chart, and then clicking the paintbrush icon (the Format icon) in the Visualizations pane. I refer to this as the “format section” for ease of reference from now on.

Note You can apply identical modifications to a set of charts on the same dashboard page by Ctrl-clicking to select multiple charts *before* you make any formatting changes.

Chart Legends

If you have a chart with more than one field that provides the values on which the chart is based (or if you are creating a pie or donut chart), then you see a legend appear automatically.

You can format a legend by modifying any of the following options:

- Legend Display
- Position
- Legend Title Display

Let's look at each of these in turn.

Legend Display

Power BI Desktop may automatically display legends in some cases, but you have the final decision as to whether a legend is required. To turn a legend off, do the following:

1. Select the chart whose legend you want to hide.
2. In the Visualizations pane, click the Format icon to display the format section.
3. Drag the Legend button to the left (or click to its left) so that it changes from a full circle to an empty circle.

The legend will disappear from the selected chart.

Legend Position

The default for the legend is for it to be placed at the top left of the chart. However, you can choose where to place the legend inside the chart area.

1. Select the chart whose legend you want to hide.
2. In the Visualizations pane, click the Format icon.
3. Expand the Legend tab.
4. Select one of the available legend positions from the Position pop-up.

The available options are given in Table 11-1.

Table 11-1. Legend Position Options

Legend Option	Comments
Top	The legend is displayed above the chart on the left
Bottom	The legend is displayed below the chart on the left
Left	The legend is displayed at the left of the chart at the top
Right	The legend is displayed at the right of the chart at the top
Top Center	The legend is displayed above the chart on the left in the center
Bottom Center	The legend is displayed below the chart on the left in the center
Left Center	The legend is displayed at the left of the chart in the middle
Right Center	The legend is displayed at the right of the chart in the middle

If one of the legend options is grayed out, it is the option that is currently active.

Legends can require a little juggling until they display their contents in a readable way. This is because the text of the legend is often truncated when it is initially displayed. If this is the case, the only real option is to modify the chart size.

Note You cannot add a legend to an area, column, bar, or line chart that only has a single data series.

Title Display

You can add a title to a legend if you want.

1. Select the chart with a legend.
2. In the Visualizations pane, click the Format icon.
3. Expand the Legend tab.
4. Switch the Title button to the On position.
5. Enter the legend title in the Legend Name box.

An example of a legend with a title is shown in Figure 11-30.

Cost Elements

- **LaborCost**
- **Direct Costs**

Figure 11-30. A legend with a title

Chart Title

Each chart is created with a title explaining what the chart is displaying; that is, the fields on which it is based. The available options are fairly simple:

- Hide or display the title
- Modify the default text of the title
- Change the font color
- Change the background color for the title
- Alter the alignment

Let's see how to apply all of these options to the title of the chart that you created in Figure 11-24.

1. Select the chart containing the title that you want to modify.
2. In the Visualizations pane, click the Format icon.
3. Expand the Title tab.
4. Ensure the Title button is in the On position. Alternatively, switch to the Off position to hide the title.
5. Modify the title text in the Title Text box. In this example, I set it to Indirect Costs.
6. Click the pop-up menu triangle to the right of Font color. Select a color from the palette of available colors.
7. Click the pop-up menu triangle to the right of Background color. Select a color from the palette of available hues.
8. Click the middle icon to the right of Alignment. This will center the text relative to the width of the table.

You can always add further annotations to a chart using free-form text boxes. This is described in Chapter 15.

Chart Data Labels

As we have seen already, you can display the exact data behind a column, bar, or point in a line chart simply by hovering the mouse pointer over the data that interests you. Yet there could be times when you want to display the values behind the chart permanently on the visualization. This is where data labels come into play.

To add data labels to a chart (in this example, I continue using the chart that you created in Figure 11-24), all you have to do is this:

1. Select the chart to which you wish to add data labels.
2. In the Visualizations pane, click the Format icon.
3. Expand the Data Labels tab.
4. Select a display unit for the figure in the card by clicking the pop-up list of display units.
5. Select a number of decimals to be used for the card data by clicking the up and down triangles to the right of the Precision box. You can also enter the number of decimals directly if you prefer.
6. Select a color for the data labels from the pop-up palette of colors.

Note When applying data labels to column, bar, and line charts, you notice that sometimes Power BI Desktop cannot physically place all the data labels exactly where the option that you have selected implies that they should appear. This is because sometimes there is simply not enough space inside a bar or column to fit the figures. In these cases, Power BI Desktop places the data outside the bar or column. On other occasions, the data cannot fit outside a line, column, or bar without being placed above the upper end of the axis. Here again, Power BI Desktop tweaks the presentation to get as close as possible to the effect that you asked for.

There are a few final points to note on the subject of data labels:

- Pie charts do not display data labels like other charts, but instead add the figure to the callout.
- Scatter charts and balloon charts do not display data labels.

Chart Background

If you need to alter the aesthetics of a chart, or should the need arise to make one chart stand out from the others in a dashboard, you can also modify the background color of the entire chart. So, continuing with the chart that you created in Figure 11-24, this is what you can do:

1. Select the chart to which you wish to add data labels.
2. In the Visualizations pane, click the Format icon.
3. Expand the Background tab.
4. Ensure that the Background ON/Off switch is set to On (a full circle on the right).
5. Click the pop-up menu triangle to the right of Background color and select a color from the palette of available colors.
6. Move the Transparency slider to the left or right to increase or decrease the transparency (and consequently the intensity) of the background color.

Data Colors

All charts allow you to specify the color of each and every data series. Of course, the amount of effort involved will depend on the number of data series that you want to alter. The way to do it is as follows.

1. Select the chart to which you wish to modify the data colors.
2. In the Visualizations pane, click the Format icon.
3. Expand the Data Colors tab. The list of chart data series (or data elements for certain chart types) will be displayed.
4. Select a color for each data element or series from the pop-up palette of available colors.

Figure 11-31 shows modified data colors for the chart that you are currently working on.

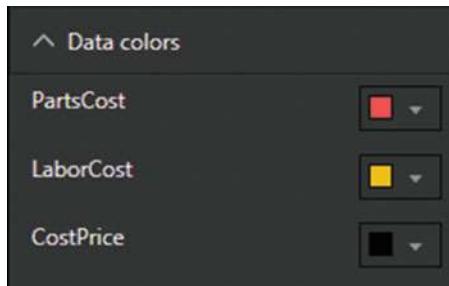


Figure 11-31. Data colors

There are a couple of things that you need to know about the way that Power BI Desktop handles data colors:

- If a chart only contains a single data series (this is always the case for pie and donut charts, and can be the case for column, line, bar, and area charts), you have the option to set a color for each element of a data series.
- Some chart types (pie, donut, column, line, bar, and area charts with only a single data series) hide the selection of data colors by default. Consequently, you have to set the Show All button to On if you want to set each color for a pie slice or donut section individually. To set a single color for all sections of the chart (and this applies to funnel charts too), you can set the Default color.

Axis Modification

Most charts—except pie, donut, and funnel charts—allow you to tweak the axis attributes. This means modifying

- The X (horizontal) axis
- The Y (vertical) axis

We will take a quick look at these two separately.

Modifying the X Axis

There are currently only two aspects of the X axis that you can alter:

- The axis display
- The axis title

A simple example explains how to do both.

1. Select the chart with an X axis that you wish to modify.
2. In the Visualizations pane, click the Format icon.
3. Expand the X Axis tab.
4. Ensure that the X axis is set to ON (this displays the axis and descriptive elements).
5. Set the Title to ON (this adds the axis title).

Modifying the Y Axis

If your chart has a Y axis (or in the case of a dual-axis chart, two Y axes), then you can modify the following elements for each axis:

- The axis position
- The axis scale type
- The lower axis value
- The upper axis value
- The title display
- The display units
- The numeric precision

Once again, it is probably easiest to see how these various formatting options can be applied using a single example. I continue using the chart that you created in Figure 11-24.

1. Select the chart to which you wish to alter the axis attributes.
2. In the Visualizations pane, click the Format icon.
3. Expand the Y Axis tab.
4. Ensure that the scale type is set to linear - or set to Log if you need a logarithmic scale.
5. Enter a Start value to set the starting figure for the axis. This can be negative, but *must* be in increments that map to the increments currently displayed on the axis.
6. Enter an End value to set the upper figure for the axis.
7. Switch Title to On to display the Y axis title.

8. Set the Style to Show Both. This way the Axis title will display both numbers and units.
9. Click the up and down triangles for the Precision box to set the number of decimals to be displayed on the axis.

If you are modifying a chart that only has a single axis, you can move the axis from the left side of the chart to the right by selecting Right from the Position pop-up menu. In the case of a chart with two axes, you can swap the axes around using this setting.

If your chart contains *two* Y axes (one on the left and one on the right), then you can repeat these settings for the other axis. You may have to scroll down in the Visualizations pane to see the settings for the second axis.

Note Remember that you can define a specific value as a measure in the data model. This can be extremely useful if you need to set reusable maximum and minimum values for the Y axis in charts.

Chart Borders

You can add a border to a chart just like you did to tables in the previous chapter. Simply click the Format icon, and set the Border button to On. You can always choose a border color by expanding the Border tab and selecting a color from the pop-up menu of available colors.

Chart Aspect Ratio

As a final comment on chart formatting, it is worth remembering that you can lock the aspect ratio of a chart. This ensures that when you resize a chart, the width and height maintain their original proportions relative to each other.

To set the aspect ratio

1. Click the chart to select it.
2. In the Visualizations pane, click the Format icon.
3. In the Lock Aspect section click Off to turn the lock on, or alternatively click to the right of the empty circle indicating that the lock is off so that it slides to the right and becomes a filled circle.

Now, when you resize a table using the corner handles, it will keep its height to width ratio.

Finally, in Figure 11-32 you can see the chart with all your modifications applied. Before rolling your eyes at my (lack of) taste, please note that I am not suggesting that you should present charts in this way, merely that you can now see the result of applying many of the available chart formatting options.

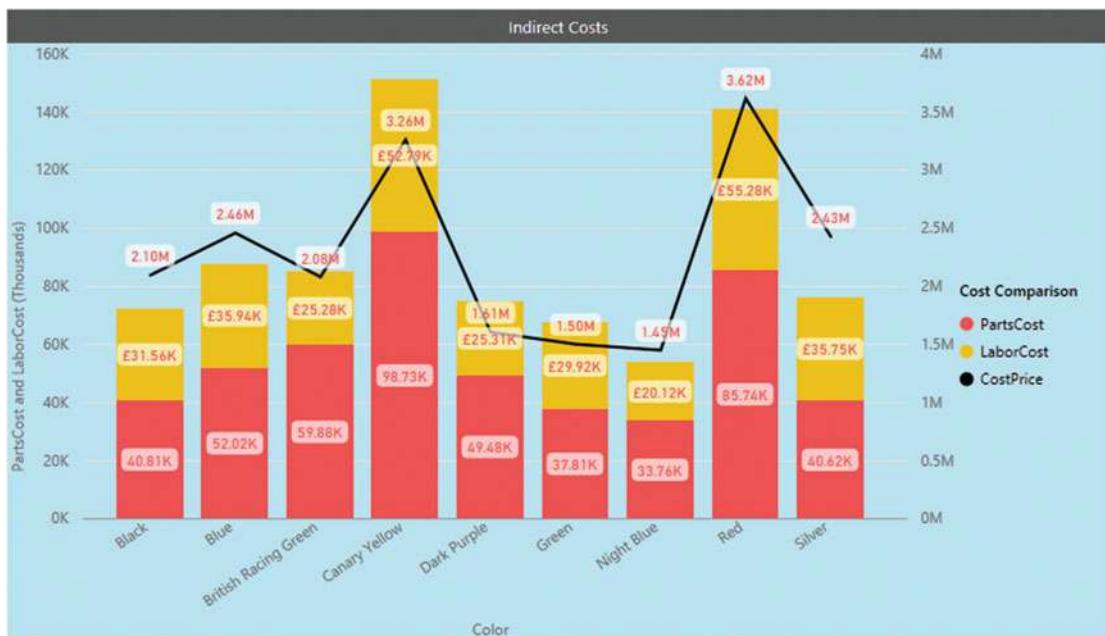


Figure 11-32. A fully formatted chart

Bubble Chart Play Axis

So far in this chapter, you have seen various ways of presenting data as charts, and how to select and compare the data using a variety of chart types. A final trick with Power BI Desktop (one that can be extremely effective at riveting your audience) is to apply a play axis to the visualization. This animates the chart, and ideally, is suited to showing how data evolves over time. Unfortunately, it is harder to get the “wow” effect using these printed pages, so this really is a technique that you have to try yourself.

You need to know that a play axis can only be applied to scatter or bubble charts. Similarly, adding a play axis will not suit or enhance all types of data. However, if you have a time-dependent element that can be added to your chart as the Y axis (such as sales to date, for instance), then you can produce some powerful and revelatory effects.

The following explains how to create a bubble chart that shows the net margin ratio for colors of car sold against the sales for the year to date.

1. Open the file C:\PowerBiDesktopSamples\ DataManagerWithMetricsForVisualizations.pbix and delete any existing visuals.
2. Create a bubble chart with the following fields using the following data:
 - a. Legend: Make
 - b. X Axis: YearSales
 - c. Y Axis: RatioNetMargin
 - d. Size: Direct Costs
 - e. Play Axis: MonthAndYearAbbr
3. Adjust the presentation (size, legend placement, data labels, etc.) to obtain the best effect.

The visual will look like that shown in Figure 11-33.

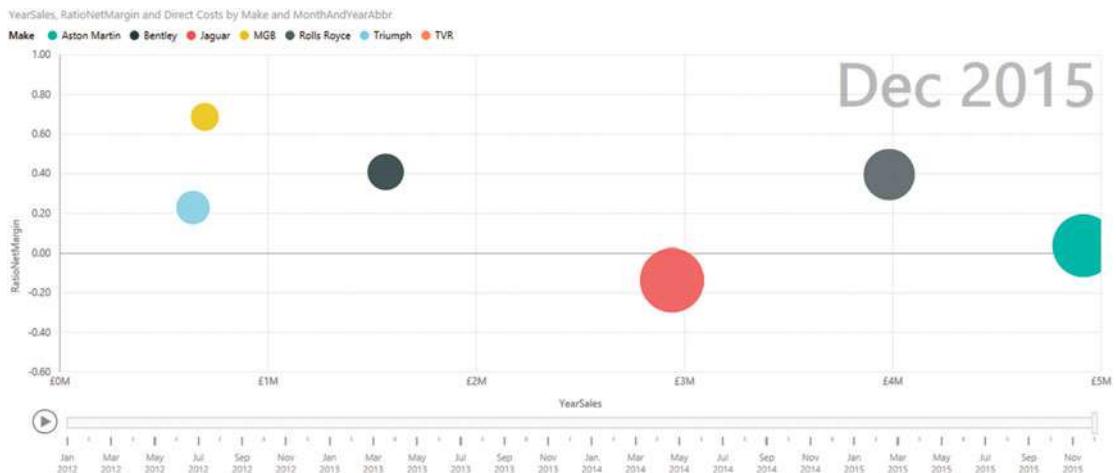


Figure 11-33. The play axis on a bubble chart

Click the Play icon to the left of the play axis. You will see the bubbles reveal how sales progress throughout the year.

There are a few points worth noting about the play axis while we are discussing it:

- You can pause the automated display by clicking the Pause icon, which the Play icon has become, while the animation is progressing. You can stop and start as often as you like.
- You can click any month (or any element) in the play axis to display the data just for that element, without playing the data before that point. This essentially means that you can use the play axis as a filter for your data.
- A play axis need not be time-based. However, it can be harder to see any coherence or progression in the data if time is not used as a basis for a play axis.
- Using cumulative data (such as the YearSales figure in this example, which is the cumulative year to date sales) is particularly effective with a time-based play axis as it lets you appreciate the growth of sales for each data item over time.
- You can use a play axis as another interactive filter for your data, but doing this makes you miss out on a fabulous animation technique!

Conclusion

This chapter took you on a tour of the available chart types that you can use in your Power BI Desktop reports and dashboards. These extend from the classic pie, line, column, and bar charts to the less common funnel, donut, area, scatter, bubble, and waterfall charts. You saw that there are charts to suit a single data series and others that can handle multiple series of data.

Then you learned how to enhance these chart types by formatting your charts for added effect. This included modifying the colors of data series and data points as well as adding or removing titles. You also saw how to add legends and data labels to certain types of chart. To add extra effect you saw how to create mixed chart types, create 100% bar and column charts and tweak axis elements.

Finally, you saw how certain types of chart can even be animated so that you can add time as a descriptive factor to help your audience understand how data evolves.

You are now well on the way to becoming a Power BI Desktop dashboard maestro. All you need to look at now are a few remaining visualization types and you will have attained a complete mastery of high-impact presentations. So now, on to the next chapter to finish your apprenticeship.

CHAPTER 12



Other Types of Visuals

While text-based visuals and charts can often make your point, there are times when you need to deliver your insights in ways that go beyond the more traditional data displays. This is where Power BI Desktop really comes to your aid. With the right data—and only a few clicks—you can revitalize your dashboards with

- Maps
- Tree maps
- Gauges

All of these visualization types are as simple to create as the text-based visuals and charts that you saw in the preceding two chapters. Yet their very ease of use should not hide you to the clarity and power that they can bring to your reports and presentations.

Yet this is only the starting point. As a freely extensible platform, Power BI also hosts a wide and growing variety of visuals developed by both Microsoft and third parties. This gallery of visual extensions is continually growing and incredibly easy to access and use. The final part of this chapter shows you some of the current visuals and how to find, add, and use them in your Power BI dashboards. This chapter will also use the file DataModelWithMetricsForVisualizations.pbix as the basis for the visuals that you will create.

Maps

Another powerful technique that you can use to both analyze and present your insights is to display the data in map form. All that this requires is that your source data contains information that can be used for geographical representation. So if you have country, state, town, postal (or Zip) code, or even latitude and longitude in the dataset, then you can get Power BI Desktop to add a map to your report and show the selected data using the map as a background.

Better yet, a Power BI Desktop map behaves just like any other visual. This means that you can filter the data that is displayed in a map, as well as highlighting it, just as you can do for charts, tables, and matrices. Not only that, but a map is an integral part of a Power BI Desktop report. So if you highlight data in a chart, a map in the same report will also be highlighted. However, you will have to wait for the next chapter to discover all this in detail.

Power BI Desktop provides you with two basic types of map visuals:

- Maps
- Filled Maps

The aim of this section is to show you some of the ways in which you can add real spice to your reports by using maps. Then, when presenting your analyses, you can interact with the maps and really—no really—impress your audience.

Bing Maps

Before adding the first map, I want to explain how mapping works in Power BI Desktop. The geographical component is based on Bing Maps. So, in order to add a map, you need to be able to connect to the Internet and use the Bing Maps service. Secondly, the underlying dataset must contain fields that are recognized by Bing Maps as geographical data. In other words, you need country, state, town, or other information that Bing Maps can use to generate the plot of the map. Fortunately, Power BI Desktop will indicate if it recognizes a field as containing data that it can use (hopefully) to create a map, as it will display a tiny icon of a globe in the field well for every field in the underlying dataset that apparently contains geographical data.

To avoid the risk of misinterpreting data, you can add metadata to the underlying PowerPivot data model, which defines geographical field types. By applying data categories to fields, Power BI Desktop maps will then use these categories to interpret geographical data for mapping. Preparing data so that any fields used by Bing Maps are not only recognizable as containing geographic data, but are also uniquely recognizable, is vital. You must help Bing Maps so that if you are mapping data for a city named Paris, Bing can see whether you mean Paris, France, or Paris, Texas. Chapter 6 explains some of the ways in which you can prepare your data for use by Bing Maps, and consequently, use it to add map visuals to Power BI Desktop.

Note There are some areas of the world that cannot use Bing Maps. So if you attempt to use Power BI Desktop mapping in these geographical zones, you will not see any map appear when you attempt to create a map.

Maps in Power BI Desktop

Let's begin by creating a map of Sales by Country. Fortunately, the sample dataset contains the country where the sale was made. This means that we can use this data to make Power BI Desktop display a map of our worldwide sales. Here is how to create an initial map:

1. Open the DataModelWithMetricsForVisualizations.pbix Power BI Desktop file.
2. Click Map in the visuals gallery. A blank map will appear (even if it looks a little peculiar, it is meant to be a map).
3. Expand the Countries table and drag the CountryName field on to the map visual. The visual will display a map showing points in all the countries for which there are values.
4. Expand the InvoiceLines table and drag the SalePrice field on to the map visual. The visual will change the size of each data point to reflect the sales value.
5. Place the cursor over the map. The pointer icon will become a hand. Click and drag Europe to the center of the map.
6. Using the mouse wheel zoom in to fit the countries with sale in the center of the visual. It will look something like Figure 12-1.



Figure 12-1. A map of sales

It is probably worth clarifying a few points about maps in Power BI Desktop before we go any further. The following are the essential points to note.

- A map is a visual like any other. You can resize and move it anywhere on the Power BI Desktop canvas.
- The map will apply any filters that have been set for the report (you will learn more about filters in the next chapter).
- Each data point (or bubble) in a map is proportional to the relative size of the underlying data.

You can hover the mouse pointer over a data “bubble” to display a pop-up showing the exact data that is represented. An example of this is shown in Figure 12-2.



Figure 12-2. Displaying the exact data in a Power BI Desktop map

Using Geographical Data

While it might seem obvious, you need geographical data if Power BI Desktop is to display your insights overlaid upon a map. So you really, absolutely must ensure that your underlying dataset contains columns of data that can be interpreted as geographical information. In the examples so far, you have seen how Power BI Desktop is capable of interpreting country names and using these to display data in maps. However, it is not limited to displaying only countries. You can use any of the following to generate map visuals:

- Country
- State
- County
- Town
- Latitude and longitude

As an example, take a look at Figure 12-3, where the Town field in the Clients table was used in the Location box in the field well. The map was then adjusted to display only the United Kingdom using the techniques described in steps 5 and 6 of the initial example.

CostPrice by Town



Figure 12-3. Displaying relative sales per town

Geographical Data Types

Power BI Desktop really is exceptionally helpful and forgiving when it comes to creating maps. If the data that you have can be interpreted geographically, then Power BI Desktop will do its best to display a map. However, you will almost certainly have to mold the dataset into a coherent data model before you start using Power BI Desktop.

In the first Power BI Desktop example, we added a single geographical data field. What is more, this field was recognized instantly for what it was—country names. In the real world of mapping data, however, you may not only have to add several fields but also specify which type of geographical data each field represents. Put simply, Power BI Desktop needs to know what the data you are supplying represents. Not only that, it needs to know what it is looking at without ambiguity. Consequently, it is up to you to define the source data as clearly and unambiguously as possible. This can involve one or more of several possible approaches.

Define the Data Category in the Data Model

As we saw in Chapter 6, you can define a data category for each column of data in PowerPivot. Although this is not an absolute prerequisite for accurate mapping with Power BI Desktop, it can help reduce the number of potential anomalies.

Add Multiple Levels of Geographical Information

The Power BI Desktop data model lets you add several levels of geographical information to a table. For instance, you can add not just a country, but also a county and a town to a record in a table. The advantage of adding as many relevant source data fields as possible is that by working in this way, you are helping Power BI Desktop dispel possibly ambiguous references. For instance, if you only had a field for town, Power BI Desktop might not know if you are referring to Birmingham, Alabama, or England's second city. If the data source has a Country field *and* a Town field, however, then Power BI Desktop has a much better chance of detecting the correct geographical location. This principle can be extended to adding states, counties, and other geographical references.

Adjusting the Map Display in Power BI Desktop

As you have seen, creating a map is extremely easy. However, the initial map is not necessarily the finalized version that you wish to show to your audience. You may wish to

- Position the map elements more precisely inside the visual.
- Zoom in or out of the map.

In the next few sections, we will look at the various modifications that you can make to Power BI Desktop maps. Hopefully, you will find these tweaks both intuitive and easy to implement. In any case, with a little practice you should find that these modifications take only a few seconds to accomplish.

Positioning the Map Elements

If the area displayed in a map is not quite as perfect as you would prefer, then you can alter the area (whether it is a country or a region) that appears in the map visual. You can do this, as follows:

1. Place the mouse pointer over the map. The pointer icon will become a hand.
2. Click and drag the pointer around to move the map elements.

Zooming In or Out

It is conceivable that the map that is displayed is not at a scale, which you would prefer. Fortunately, this is extremely easy to fix. All you have to do is the following:

1. Place the mouse pointer over the map.
2. Roll the mouse scroll wheel forward to zoom in to see part of the map in greater detail. Alternatively, roll the mouse scroll wheel backward to zoom out of the map.

Multivalue Series

So far, we have seen how you can add a single data series to a map and have the data represented as a data point. Power BI Desktop can extend this paradigm by allowing you to display the data bubble as a pie that contains a second data series—and consequently display the data broken down by a specific dataset per geographical entity.

As an example of this (and to revise the some of the map creation techniques that we have seen so far), let's try to analyze European car sales by age range.

1. Keep the map that you created in the previous section (or create it if you have not yet done so).
2. Expand the Colors table and drag the Color field into the field well onto the Legend box.
3. Filter the table to exclude the United States as described in the previous section.
4. Resize the map visual.

The map now contains a legend for the colors of vehicles sold. Each bubble is now a pie of data. The overall size of the pie represents, proportionally, the sum total data for each country compared to the other pies. The map should now look like Figure 12-4.

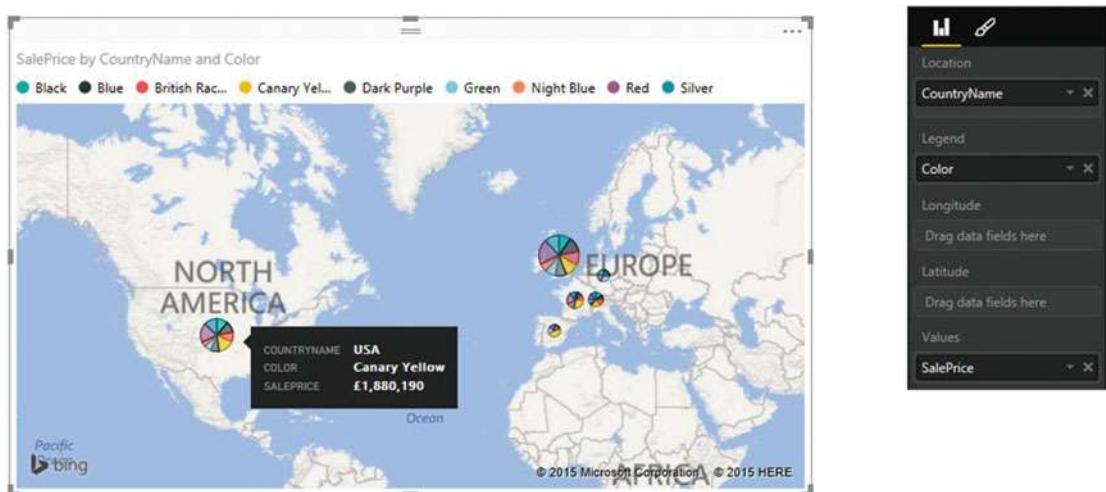


Figure 12-4. Displaying pie charts in a Power BI Desktop map

It is worth remarking that if you hover the mouse pointer over the data representation (the pie) for a country, as you pass the pointer over each pie segment, you will see a tooltip giving the details of the data, including which car color it refers to.

Note When displaying maps, you nearly always need to filter data in some way. You will discover all the details about filtering data in the next chapter.

Highlighting Map Data

If you have added data to the Legend box of the field well, and a legend is displayed, you can highlight segments of data in a map. This allows you to draw the audience's attention to specific trends in your data.

Using the map that you created and can see in Figure 12-3, click Blue (the second element) in the legend. The pie segments that correspond to blue cars sold are highlighted, as shown in Figure 12-5.

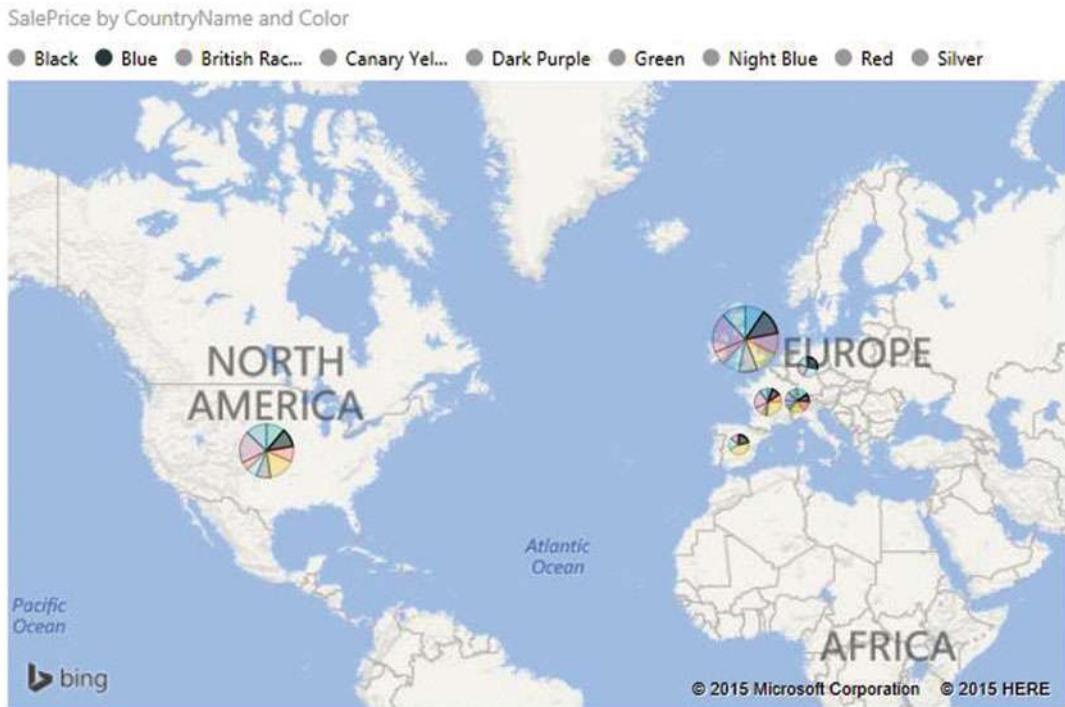


Figure 12-5. Highlighting map data

To remove highlighting from a map, all you have to do is click the legend element that you are using to highlight data, or simply click the title of the legend.

Filled Maps

Another way to display data is to fill a geographical area with color rather than to display a bubble. Power BI Desktop lets you do this, too.

1. Click in an empty part of the dashboard canvas in the DataModelWithMetricsForVisualizations.pbix Power BI Desktop file.
2. Click the Filled Map icon in the visuals gallery. A blank map will appear.
3. Expand the Countries table and drag the CountryName field on to the map visual. The visual will display a map shading in the countries for which there are values.
4. Expand the Stock table and drag the CostPrice field on to the Values box in the field well. The map will change the color of the shading for each country to reflect the sales value. A darker shade indicates a higher value. The map will now look like Figure 12-6.

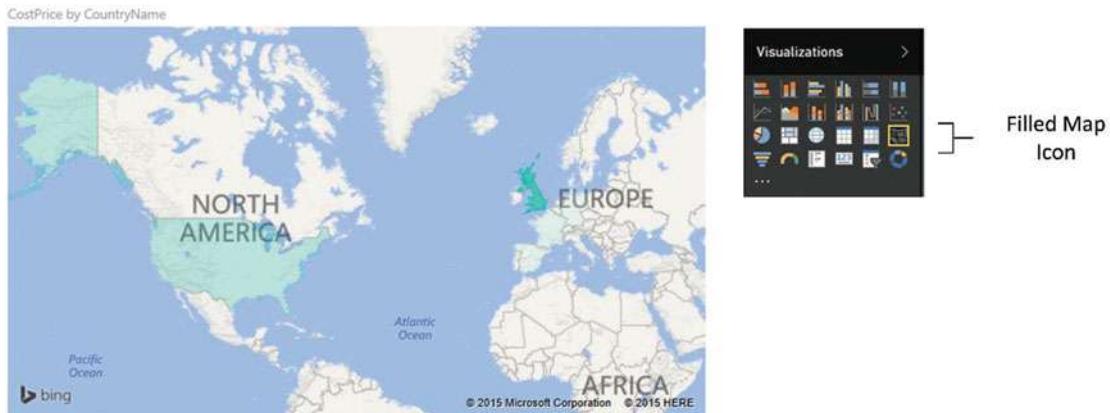


Figure 12-6. A filled map of costs per country

Formatting Maps

As you would probably expect, there are a certain number of formatting options that you can apply to maps. Indeed, the way that you can format maps is virtually identical to the way that you apply formatting to other visuals. Given that the two previous chapters covered many of these options in detail, I will only explain the possibilities in this section, and refer you back to Chapters 10 and 11 if you want all the details.

As far as maps and filled maps are concerned, you can modify the following:

- *Title:* This includes displaying or hiding the title, as well as setting the title text, font color, and background color.
- *Background:* This covers applying a background, setting its color, and defining its transparency.
- *Aspect Ratio:* If you set Lock Aspect to On, then you can resize the map while keeping it sized proportionally.

Both map types allow you to adjust the data colors. However, the way that the colors are modified is slightly different, depending on the map type.

- *Maps*: Lets you set a default color or alternatively choose the color to apply for each data bubble. It is similar to the way that you can define colors in bar or column charts, for instance.
- *Filled Maps*: Allows you to set a color for the lowest value and a color for the highest value. Power BI Desktop will then apply shading to represent the range of values.

Precisely because it is unlike any formatting that you have seen so far, let's look at how to set the colors for a filled map.

1. Return to the filled map that you created in Figure 12-6.
2. Click the map.
3. In the Visualizations pane, click the Format icon.
4. Expand the Data Colors tab.
5. Choose a color for the minimum color from the pop-up palette of colors to the right of the word Minimum.
6. Choose a color for the maximum color from the pop-up palette of colors to the right of the word Maximum.
7. Click in the Minimum box and enter a value for the minimum threshold. This will exclude any outlier values at the lower end of the data range.
8. Click in the Maximum box and enter a value for the maximum threshold. This will exclude any outlier values at the higher end of the data range.

The map should now look something like Figure 12-7.



Figure 12-7. Applying shading for values in a filled map

Note You do not have to set the minimum and maximum values for the shading that is applied, but doing this lets you control how the range of shading is applied to the data.

Finally, maps (but not filled maps) let you add category labels. You can do this as follows.

1. Create a map like the one in Figure 12-5.
2. In the Visualizations pane, click the Format icon.
3. Expand the Category Labels tab.
4. Set Category Labels to On.
5. Choose a color for the category labels from the pop-up palette of colors to the right of Category labels. The map will look something like Figure 12-8.

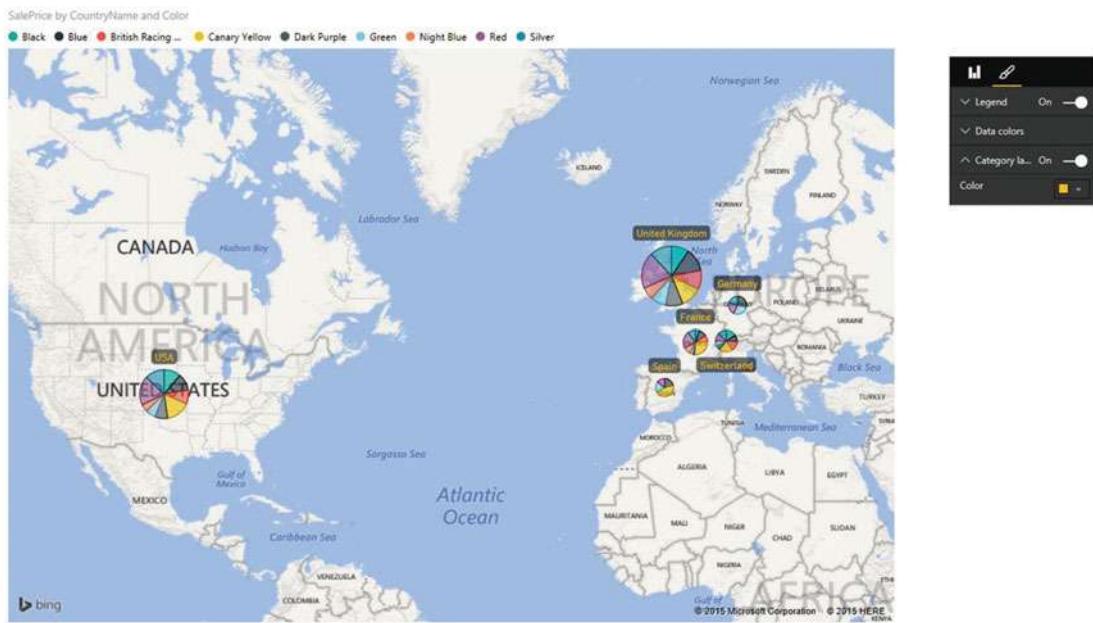


Figure 12-8. A map with category labels added

Tree Maps

One type of visual that can really help your audience to see the way that individual values relate to a total is the tree map. While this is not a map in any geographical sense, it can certainly assist users in finding their way around a set of figures.

As, yet again, seeing is believing in the world of visuals, let's take a look at a tree map showing how the labor costs stack up for each make and model of car purchased.

1. Open the DataModelWithMetricsForVisualizations.pbix Power BI Desktop file, or click an empty part of the dashboard canvas.
2. Click the Tree Map icon in the Visualizations pane. A blank tree map will appear.
3. Expand the Stock table.
4. Drag the Make field on to the Group box in the field well.
5. Drag the Model field on to the Details box in the field well.
6. Drag the LaborCost field on to the Values box in the field well. The tree map should look like the one in Figure 12-9.

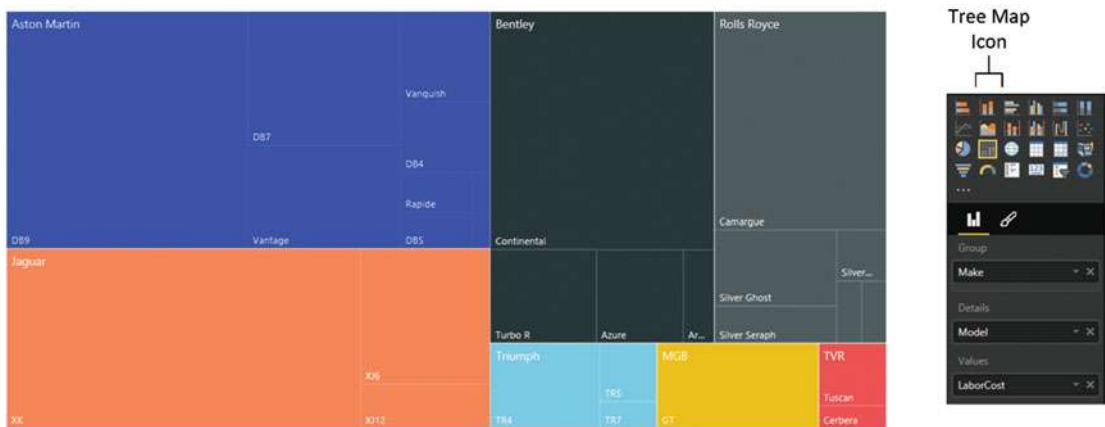


Figure 12-9. A tree map showing labor cost for each make of car

As you can see in Figure 12-9, the tree map has grouped each model of car by make, and then displayed the labor cost as the relative size of each box in the tree map. This way, your viewers can get a rough idea of

- How the labor cost for each model compares to the total labor cost for the make
- How the labor cost for each make compares to the total labor cost.

As you are probably expecting by now, you can hover the mouse over any box in the tree map and see a pop-up of the exact data.

Power BI Desktop will decide how best to organize the tree map. However, the final appearance will depend on how wide and how tall the visual is. So don't hesitate to resize this particular kind of visualization and see how it changes as you adjust the relative height and width.

There could be cases when a tree map cannot display all the available data, possibly because there are too many data points, or the range of values is too wide to be shown properly, or (as the case with the pie charts) because there are negative values in the data. In these circumstances, the tree map shows a warning triangle in the top left of the visual. If you hover the mouse pointer over this icon, you will see a message like the one shown in Figure 12-10.

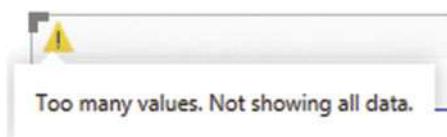


Figure 12-10. The tree map data alert

If you see this warning, then you could be advised to apply filters to the data (as described in the next chapter) to filter out the data that is causing the problem.

Formatting Tree Maps

Tree maps can be formatted just like any other visual. The following are the available options:

- Legend
- Data Colors
- Category Labels
- Title
- Background
- Lock Aspect

Since all of these options applied to visualizations in the previous two chapters, I will not explain them in detail again, but I will provide a few comments about the available options.

Legend

In my opinion, a legend is superfluous as it duplicates the information about the grouping data that is already displayed in the tree map. You may find a legend necessary if you choose not to display the category labels, however.

Data Colors

This palette lets you choose a specific color for each grouping element.

Category Labels

This option lets you decide whether or not to display the grouping elements and the detail element information inside the tree map. If your tree map contains dozens (or hundreds) of data points, then you might want to hide the category labels.

Title

As is the case with most visuals, you can choose to show or hide the title, as well as alter its text, font size, and background color.

Background

You can apply a colored background to the kind of visuals that you can see in this chapter, too.

Lock Aspect

If you set Lock Aspect to On, then you can resize the visual while keeping it proportionally sized.

Gauges

After nearly three chapters of delving into the range of visuals that Power BI Desktop has to offer, there is one final built-in visual to look at. This is the Power BI dashboard gauge. These kinds of visuals are particularly effective when you want to compare actuals to targets or see how a metric compares to a key performance indicator (KPI), for instance. In this example, you will use a gauge to see how the total cost of purchases and spares compares to the threshold set by the finance director.

1. Open the DataModelWithMetricsForVisualizations.pbix Power BI Desktop file, or click an empty part of the dashboard canvas.
2. Click the gauge icon in the Visualizations pane. A blank gauge visual will appear.
3. Expand the InvoiceLines table.
4. Drag the Cost Plus Spares field into the Value box in the field well. Alternatively, drag this field directly onto the gauge that you just created.
5. Drag the SalePrice field into the Maximum box in the field well.
6. Drag the TotalCostPlusSpares field into the Target value box in the field well.

The gauge will look like Figure 12-11.

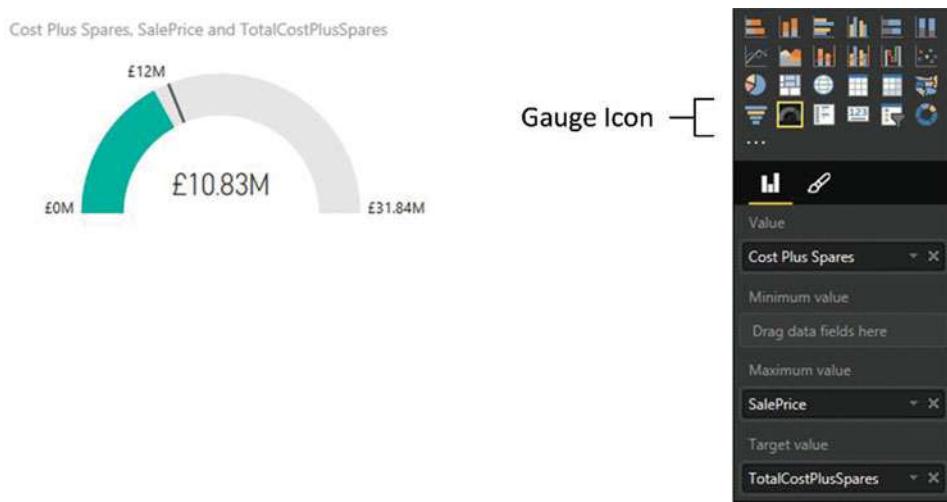


Figure 12-11. A gauge in Power BI Desktop

Gauges are designed to show progress against a target or a total. Consequently, you can set the elements described in Table 12-1 for a gauge.

Table 12-1. The specific data requirements for gauges

Element	Description
Value	The element that appears as a colored bar in the gauge.
Minimum Value	The minimum value to alter the perceptual effect of the gauge.
Maximum Value	The maximum value to alter the perceptual effect of the gauge
Target Value	A metric that is defined as something to be attained.

In fact, all that a gauge needs is a value. It will work without any of the other three elements. However, the real visual power of a gauge comes from the way that it uses the viewer's presumption of a target or value to be achieved. Consequently, it is usually best to apply a target at least, and a maximum value to convey an idea of success or failure for the reader.

Note Remember that you can define a specific value as a measure in the data model. This can be extremely useful if you need to set the maximum, minimum or target values for gauges. This way you can modify a threshold once without having to change the attributes of multiple visuals independently.

Formatting Gauges

Gauge formatting is highly specific to gauges. So here is a quick trip through the available presentation options for gauges.

1. Click the gauge that you created previously.
2. In the Visualizations pane, remove the Target value (the CostPlusSpares field).
3. Remove the Maximum value (the TotalCostPlusSpares field).
4. In the Visualizations pane, click the Format icon.
5. Expand the Gauge Axis tab.
6. Enter **2000000** as the minimum value.
7. Enter **30000000** as the maximum value.
8. Enter **10000000** as the target.
9. Expand the Data Labels tab.
10. Ensure that the Data Labels switch is set to On.
11. Choose a color from the pop-up palette for colors for the data label.
12. Expand the Callout Value tab.
13. Ensure that the Callout Value switch is set to On.
14. Choose a color from the pop-up palette for colors for the callout value. The gauge will look something like Figure 12-12.

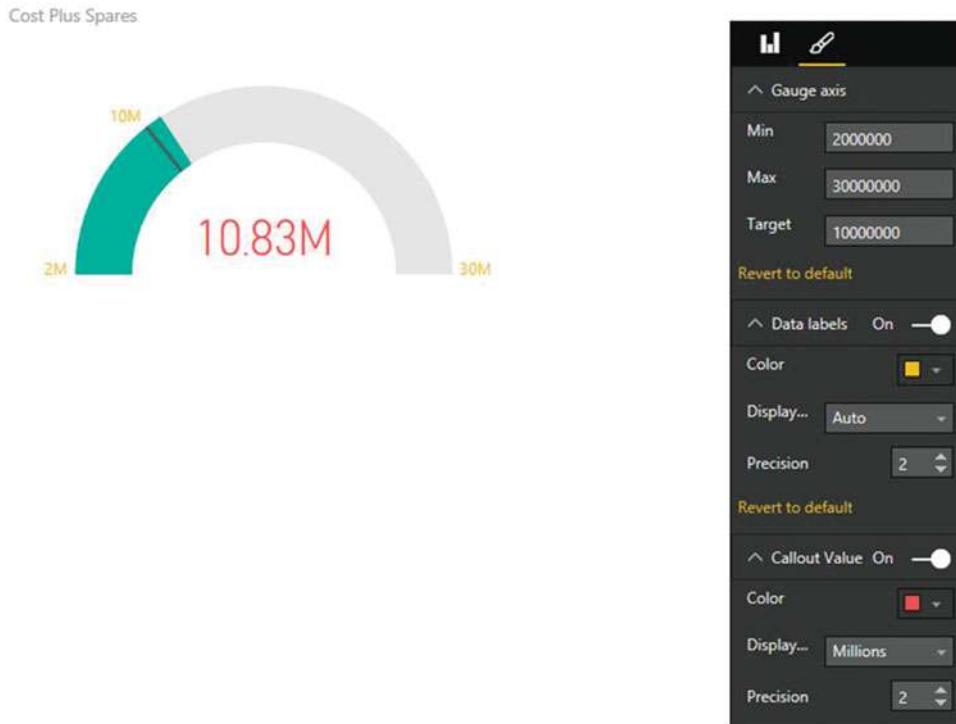


Figure 12-12. A formatted gauge

If you wish, you can also alter the background and title just as you did previously for other visuals. You can also choose not to display the data labels or the callout value (the value of the gauge) if you prefer. All you have to do is to set their respective switches to the Off position.

Note When entering figures in the format section of the Visualizations pane, you must not add any numeric formatting, such as thousands separators. If you do, the formatting will not work and Power BI will wait until you have entered the numbers in their raw form.

Gauges are a simple yet effective visual. A set of gauges can also be a truly effective way of displaying high-level key metrics on a dashboard.

Additional Visuals

By now, you must surely have come to appreciate the sheer range of visual possibilities that Power BI Desktop has to offer. From a range of chart types to gauges, tables, and cards, it delivers a wealth of easy to use ways of delivering insight into your data clearly and effectively.

Yet Power BI Desktop does not stop with the built-in visualizations that you have seen so far. It has been designed to be a completely open and extensible business intelligence application that will allow third parties (and even Microsoft) to add other visuals. This means that the core elements that you have met so

far are only a starting point. There are many other chart, text, and map visuals that you can add to Power BI Desktop in a few clicks—and then literally stun your audience with an eye-catching variety of dashboard elements.

Yet adding and using these enhancements is both quick and easy. This is because all Power BI visualizations use the same interface and they are based on the underlying data model in exactly the same way. Consequently, learning to use a new visual will most likely only take a couple of minutes, as you are always building on the knowledge and techniques that you have already acquired.

Note You need to be aware that not all the custom visuals that are available are entirely bug-free. Also, they might (or might not) have all the formatting attributes or interactive capabilities available that you might wish for. However, this does not make them any less useful—or any less fun to use. In any case, we can reasonably expect them to be enhanced and improved over time.

The Power BI Visuals Gallery

The first thing to do is become acquainted with the Power BI visuals gallery. This is a central hub where you can find a range of free visuals developed either by Microsoft or by third parties that are ready to be added to your Power BI dashboards.

The following explains how to connect to the Power BI visuals gallery.

1. Open your web browser and navigate to <https://app.powerbi.com/visuals>.
You will see a set of available visuals rather like the one shown in Figure 12-13.



Figure 12-13. Some of the currently available visuals in the Power BI visuals gallery

This page has undoubtedly evolved considerably since this book went to print, and so I imagine that what you are looking at now is very different. Hopefully, though, you should be excited to see lots of new and amazing types of data visualization that will allow you to add unparalleled pizazz to your dashboards.

Loading Custom Visuals

Now that you have found a treasure trove of extra presentation elements for your reports, you need to learn how to add them to Power BI Desktop.

1. In your web browser, navigate to <https://app.powerbi.com/visuals>.
2. Click the visual that you want to download. A dialog will appear asking you to confirm the download. It will look something like Figure 12-14.

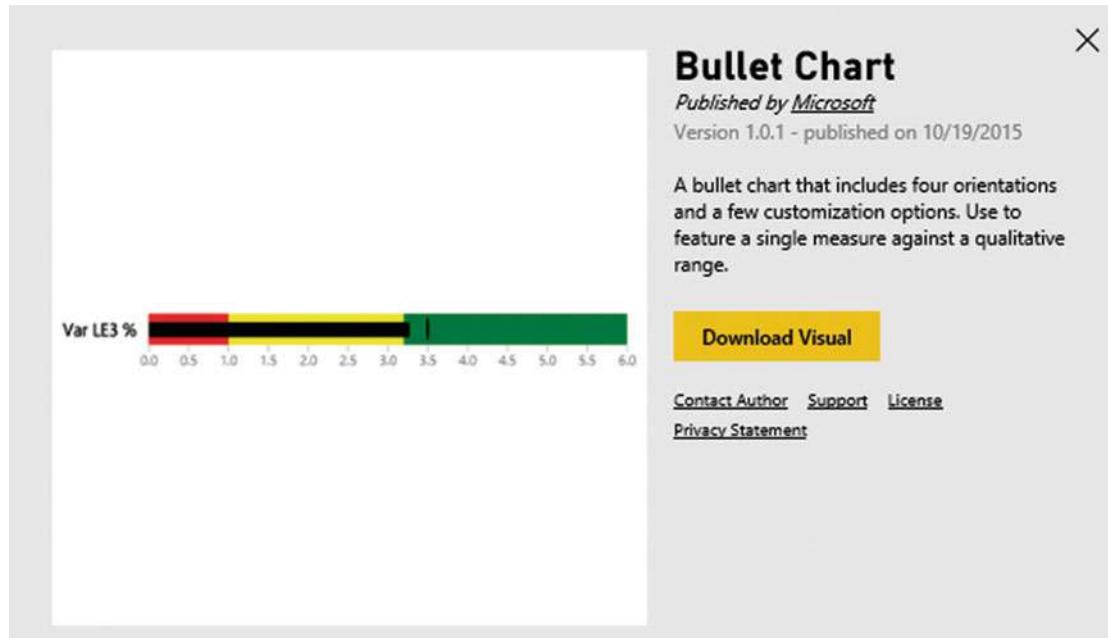


Figure 12-14. The visuals download dialog

3. Click Download Visual. The community agreement dialog will appear, as shown in Figure 12-15.

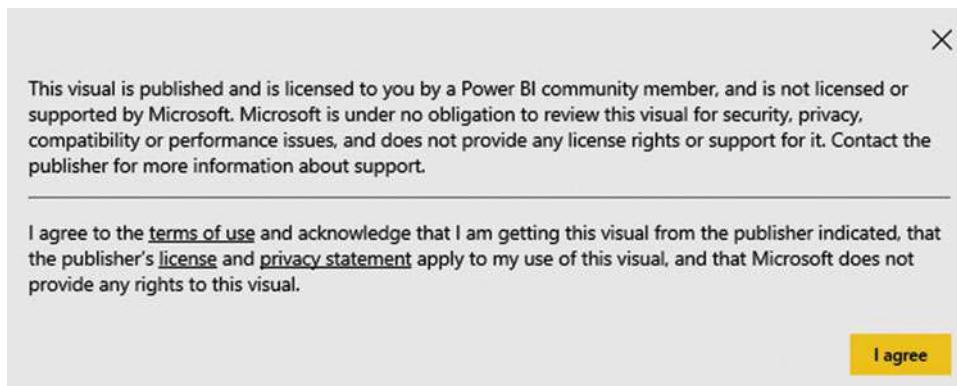


Figure 12-15. The visuals community agreement dialog

4. Click I Agree. The visual will be downloaded. Depending on your version of windows (and any machine-specific configuration) it will be placed in a specific folder or give you the choice of a folder.
5. In Power BI Desktop, click the import custom visual button (the ellipses) in the gallery of visuals in the Visualizations pane. A cautionary dialog like the one in Figure 12-16 will appear.

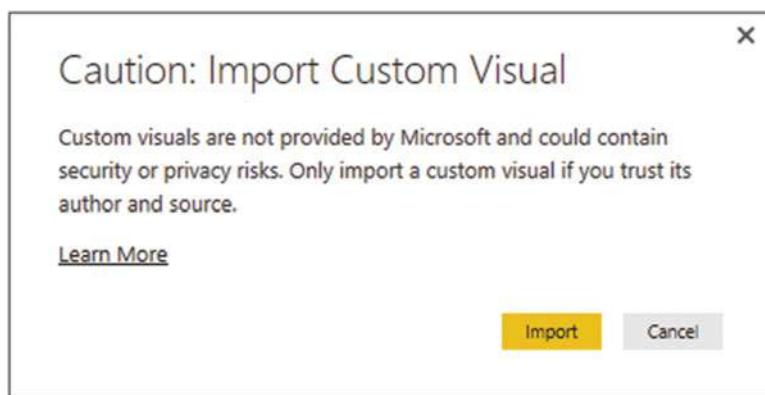


Figure 12-16. The import visuals dialog

6. Click Import. The Windows File Open dialog will appear.
7. Navigate to the folder where you downloaded the visual file in step 3. Click the visual file. You can see an example of this in Figure 12-17.

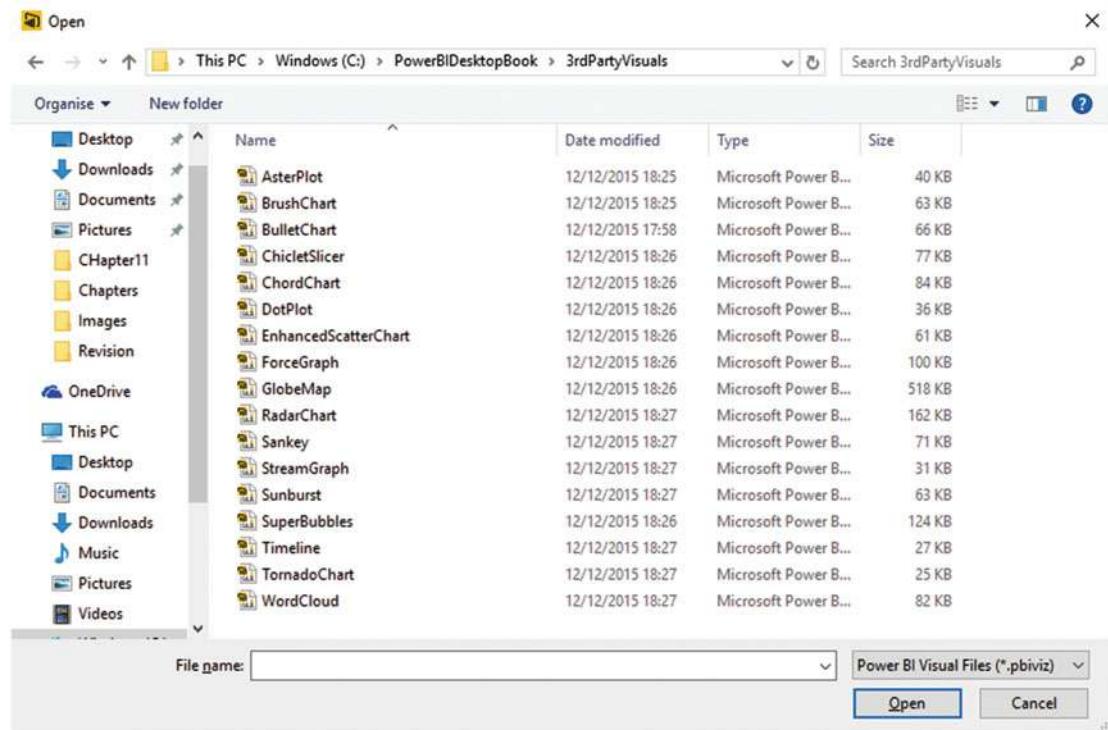


Figure 12-17. Loading a visual from file

8. Click Open. The confirmation dialog will appear, as shown in Figure 12-18.

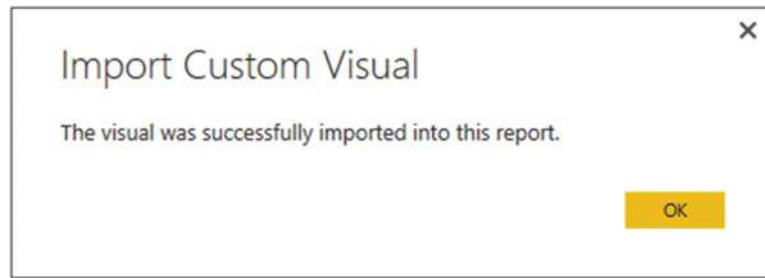


Figure 12-18. The visual import confirmation dialog

The icon for the visual will appear at the bottom of the visuals gallery in the Visualizations pane. You can then use this visual just like any other Power BI visual.

Tip Visuals are added as required to each individual Power BI Desktop file. So while you only download the visuals once, you have to load them into each Power BI Desktop (.pbix) file that you want to use them in. If you find yourself frequently using the same third-party visuals, you should create a Power BI Desktop file that contains all the visuals that you use, and then make it a template for your future dashboard development.

Enabling Custom Visuals

As an added security measure, you have to enable custom visuals each time that you open a Power BI Desktop file that contains third-party enhancements. Indeed, every time that you open a file that contains custom visuals, you will see an alert like the one shown in Figure 12-19.

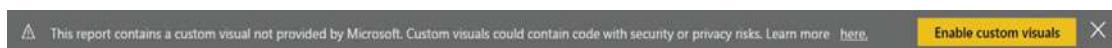


Figure 12-19. The Enable Custom Visuals alert

What is more, custom visuals in the report are not displayed. Instead, and in place of each one, you will see the blank placeholder that is shown in Figure 12-20.



Figure 12-20. The custom visual placeholder

If you are happy that the file only contains visuals that are safe, then all you have to do is to click the Enable Custom Visuals button. This does the following:

- Remove the alert from the top of the dashboard
- Allow custom visuals to be displayed in the report
- Display the icons for any installed custom visuals in the Visualizations pane

A Rapid Overview of a Selection of Custom Visuals

As the gallery of custom visuals is in a state of permanent flux, it is impossible to discuss all the currently available extensions to Power BI. However, to give you an idea of some of the possible enhancements that are available, the next few pages show some of the added visuals made available by Microsoft. Since there is no guarantee that these visuals will still be available in their current state by the time that you are reading this book, I do not explain how to create them, but merely show some examples of other chart types using the data available in the sample data that you have been using in this book.

Aster Plots

An *aster plot* is derived from the standard donut chart. However, it uses an extra data value to provide the “sweep” that you see in Figure 12-21.

SalePrice by Make

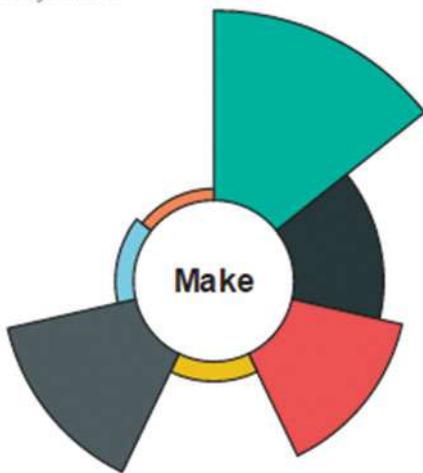
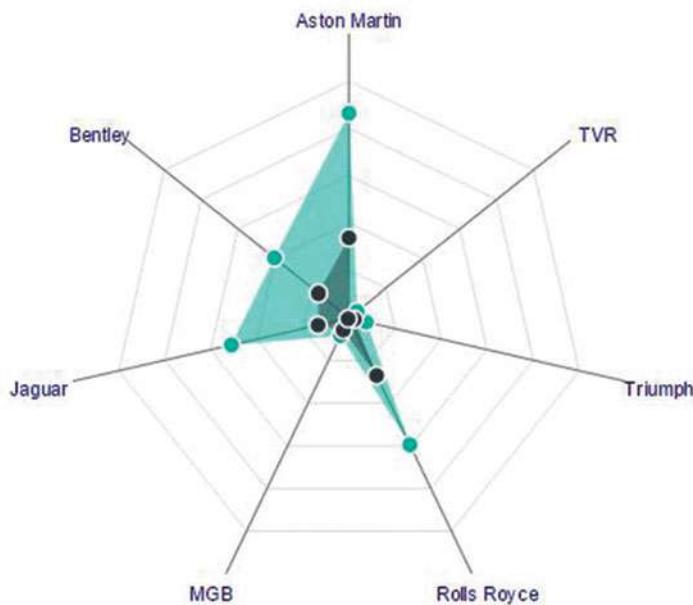


Figure 12-21. An aster plot

Radar Charts

Radar charts are often used in performance analyses to display metrics. They allow you to show multiple values relative to a shared axis, as you can see in Figure 12-22.

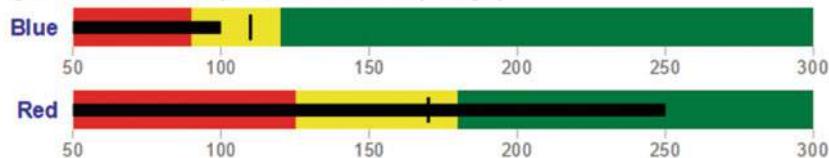
SalePrice and Gross Margin by Make

**Figure 12-22.** A radar chart

Bullet Charts

Bullet charts are extremely useful for tracking key performance indicators (KPIs) against targets. They frequently require you to add data that provides the thresholds against which performance is measured, either as absolute values or as percentages. However, they can be extremely effective at displaying how results compare to targets, as Figure 12-23 shows.

Value, Target, Minimum, Satisfactory, Good and Maximum by Category

**Figure 12-23.** A bullet chart

Word Clouds

A *word cloud* shows the number of times that words in a dataset appear relative to each other. It can be a uniquely visual way to show relative weights of values, as you can see in Figure 12-24.



Figure 12-24. A word cloud

Sunburst Charts

Sunburst charts are essentially multilevel donut charts. However, as you can see in Figure 12-25, they are good at showing hierarchies of data. The inner ring represents each car make and the corresponding segments of the outer ring show the sales for each model compared to the sales by make and the whole.



Figure 12-25. A sunburst chart

Streamgraphs

A *streamgraph* is a smoothed, stacked, area chart. As you can see in Figure 12-26, a streamgraph is good at showing how values change over time.

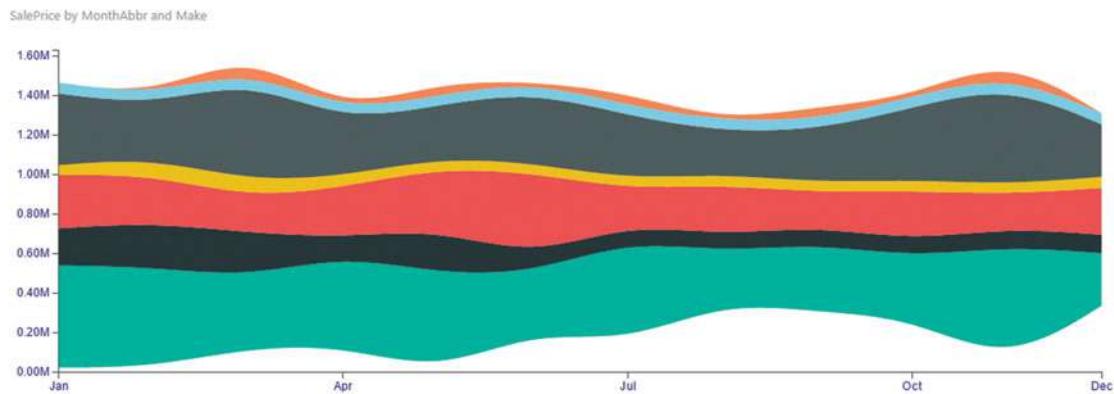
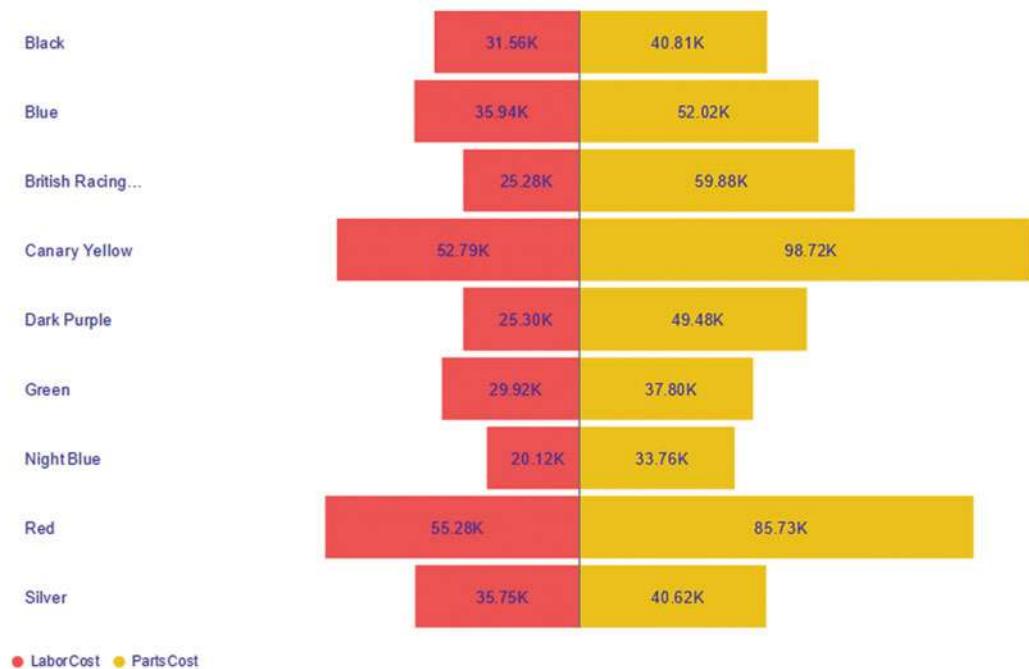


Figure 12-26. A streamgraph

Tornado Charts

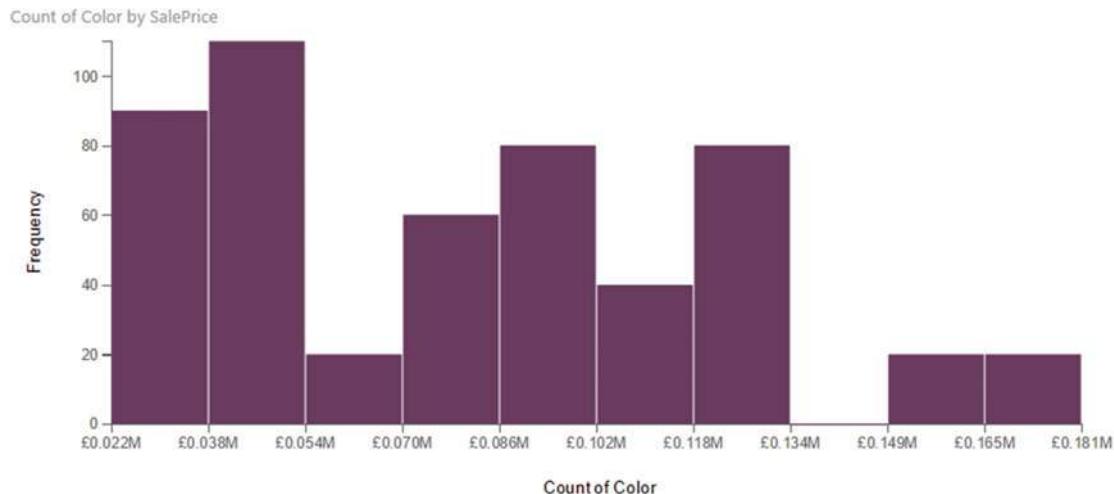
A *tornado chart* is a variation of a bar chart; only the separation between two sets of values is made clearer by the vertical separation, as you can see in Figure 12-27.

LaborCost and PartsCost by Color

**Figure 12-27.** A tornado chart

Histograms

A *histogram* is a type of chart that shows data as ranges. In Figure 12-28, you can see the sale process grouped into discrete “buckets” using a histogram chart.

**Figure 12-28.** A histogram

Chord Charts

A *chord chart* is an interesting—and somewhat less traditional—way of displaying the relationship between values in a matrix structure. Figure 12-29 shows how colors and countries relate by sales.

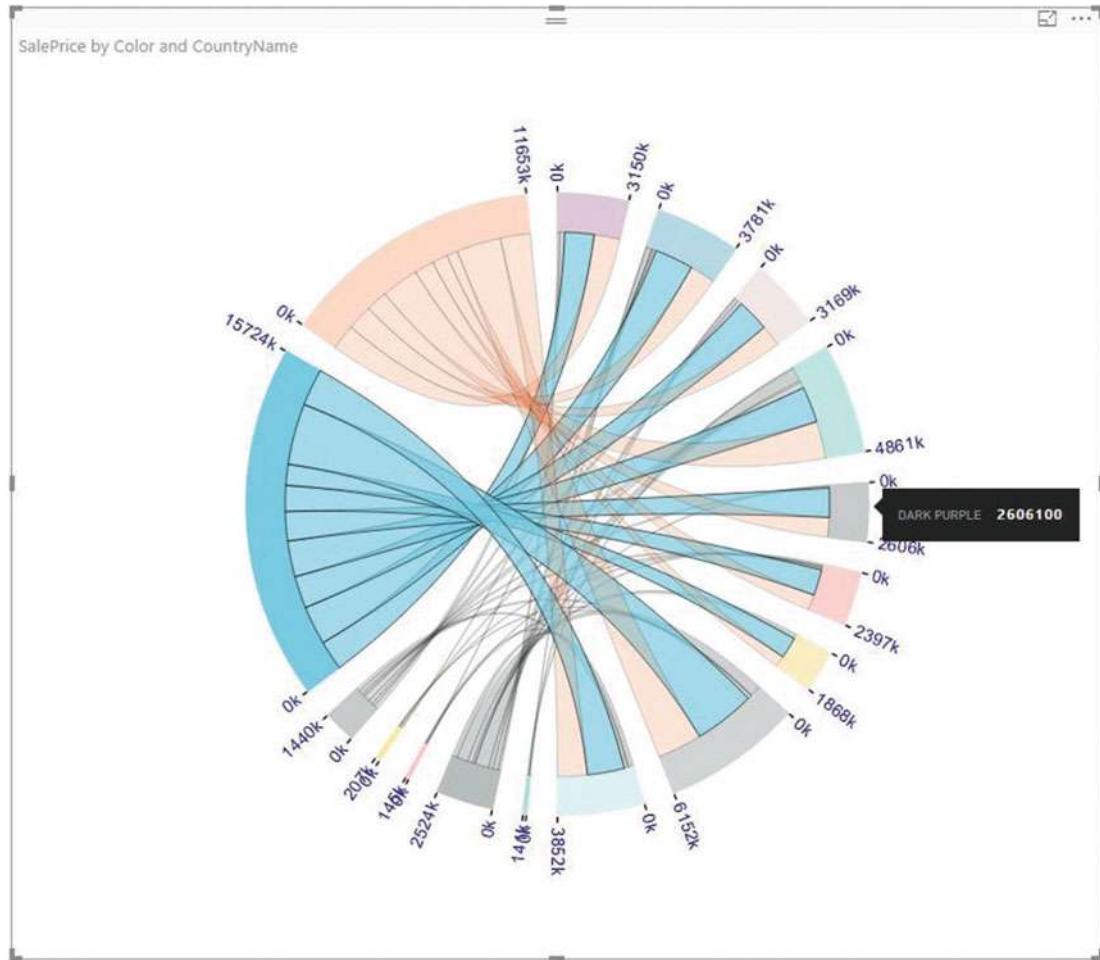


Figure 12-29. A chord chart

Sankey Diagrams

The *Sankey diagram* in Figure 12-30 also displays how colors and country sales relate.

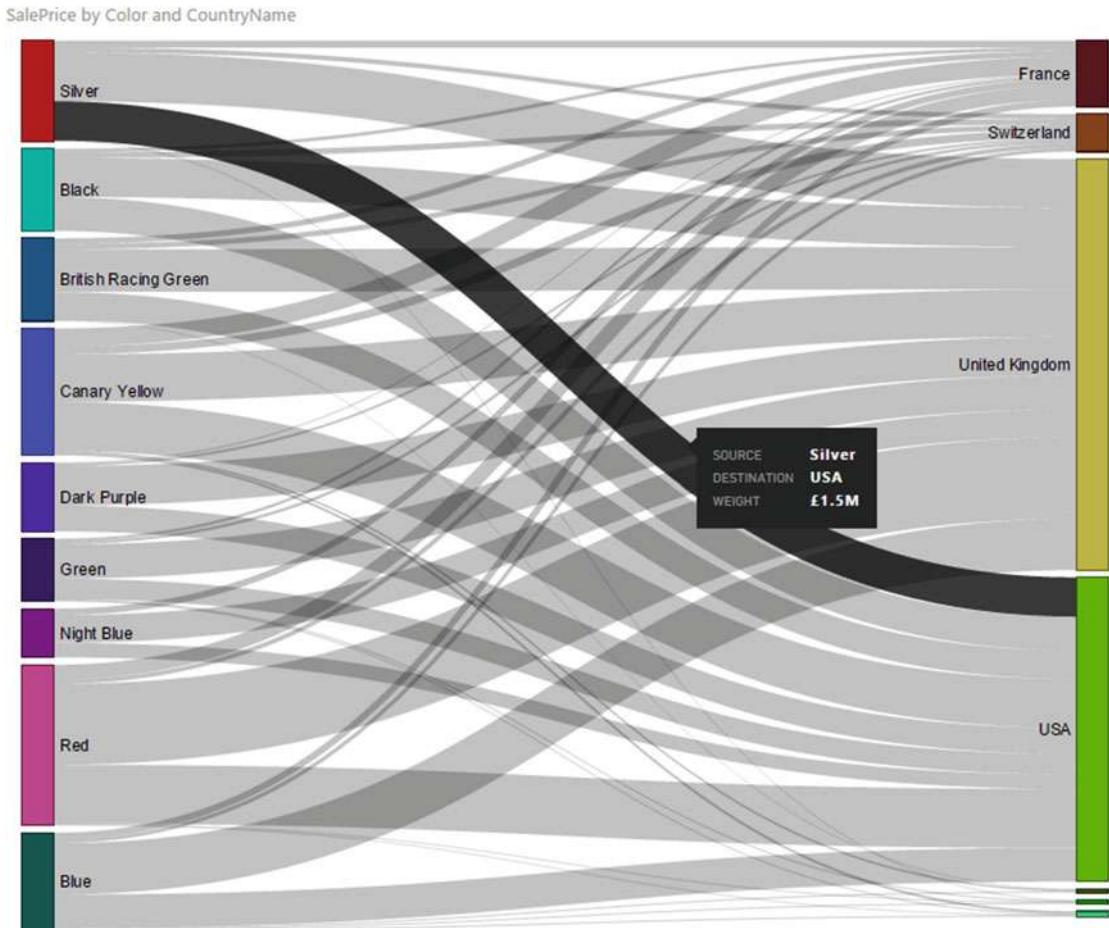


Figure 12-30. A Sankey diagram

Conclusion

Over the course of three chapters, you have met a wide range of visuals that let you express the trends hidden in your data. This chapter built on the two previous chapters, where you learned all about text-based visuals and charts by adding maps, tree maps, gauges, and custom visuals to the mix.

First, you saw how to display geographical data in the form of maps—both filled maps (where the data is represented by the intensity of an area's shading) and bubble maps, where the size of data points on a map lets the data speak for itself.

Then, you learned to create tree maps to help your public understand how groups and finely grained elements make up a whole. To complete the tour of built-in visuals, you saw how to add gauges that let you compare metrics to targets and focus on objectives.

Finally, just as you thought that it could not get any better, you discovered the jewel in the crown of Power BI Desktop: an extensive range of freely available extensions that you can add to your dashboards to deliver your insights in ways that leave other tools in the dust.

CHAPTER 13



Filtering Data

Power BI Desktop is built from the ground up to enable you, the user, to sift through mounds of facts and figures so that you can deliver meaningful insights. Consequently, what matters is being able to delve into data and highlight the information it contains quickly and accurately. This way, you can always explore a new idea or simply follow your intuitions without needing either to apply complex processes or to struggle with an impenetrable interface. After all, Power BI Desktop is there to help you come up with new analyses that could give your business an edge on the competition.

Power BI Desktop provides two main approaches to assist you in focusing on the key elements of your data:

- *Filters*: Restrict the data displayed in a report, page, or visual. This is the subject of the current chapter.
- *Interactive data selection*: Allows you to highlight key information instantly and visually for an audience. We will look at these aspects in the next chapter.

The people who developed Power BI Desktop recognize that your data is the key to delivering accurate analysis. This is why you can filter on any field or set of fields in the underlying data model to extract its real value. This approach is not only intuitive and easy, it is also extremely fast, which ensures that you almost never have to wait for results to be returned.

You can add filters before, after, or during the creation of a Power BI Desktop report. If you add filters before creating a table, then your table will only display the data that the filter allows through. If you add a filter to an existing report, then the data visualization will alter before your eyes to reflect the new filter. If you modify a filter when you have visualizations on a Power BI Desktop report, then (as you probably guessed by now), all the visualizations affected by the filter will also be updated to reflect the new filter criteria—instantaneously.

You can filter any type of data:

- Text
- Numeric values
- Dates

Each data type has its own ways of selecting elements and setting (where possible) ranges of values that can be included—or excluded. This chapter will explain the various techniques for isolating only the data that you want to display. You will then be able to create Power BI Desktop reports based only on the data that you want them to show.

We will see all how these approaches work in detail in the rest of this chapter. In any case—and as is so often the case with Power BI Desktop—it is easier to grasp these ideas by seeing them in practice than by talking about them, so let's see how tiles, slicers, and highlighting work. This chapter uses the DataModelWithMetricsForVisualizations.pbix sample file as the basis for all the filters that you will learn to apply.

Filters

Subsetting data in Power BI Desktop is based on the correct application of filters. Consequently, the first thing that you need to know about filters is that they work at three levels. You have

- Report-level filters
- Page-level filters
- Visualization-level filters

The characteristics of these two kinds of filter are described in Table 13-1.

Table 13-1. Power Bi Desktop Filters

Filter Type	Application	Comments
Report-level	Applies to every visualization in the current report.	Filters filter data for every visual in the current file.
Page-level	Applies to every visualization in the active page.	Filters data for every visual in the current page.
Visual-level	Only applies to the selected visualization.	Applies only to the selected visual (table, chart, etc.).

Filters are always applied in exactly the same way. What matters is the extent that they affect the visuals in a Power BI Desktop presentation. In practice, this makes your life much easier because you only have to learn how to apply a filter once and then you can use it in the same way at different levels in separate files.

Let's now look at how to use filters, beginning at the lowest level of the filter hierarchy: visualization-level filters.

Visual-Level Filters

Saying that there are three types of filter available in Power BI Desktop is a purely descriptive distinction. For Power BI Desktop, any filter is a filter, and all filters work in the same way. However, as there is a clear hierarchy in their application, I will begin with visual-level filters and then move on to their descendants—page-level filters and report-level filters. Given the general similarity between the three, it is probably worth noting that it is important you ensure that you are creating or modifying the appropriate filter. As this is not always obvious, not least when you are starting out with Power BI Desktop, I will try to make it clear as we proceed how exactly you can distinguish at what level you are applying a filter, as the effects can have wide-ranging consequences for the message that you are trying to convey.

The Filters Well

The first thing to note is that all filters are applied in the Filters well. This is in the lower part of the Visualizations pane, and looks (in this instance) like Figure 13-1.

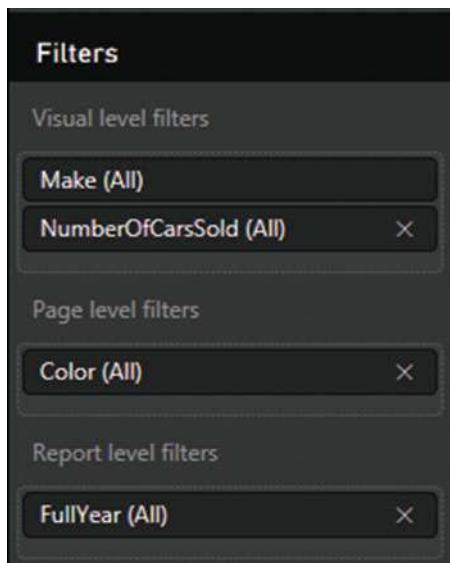


Figure 13-1. The Filters well

As you can see in this figure, different filters have been applied at all the possible levels of the filter hierarchy. If you look closely, you can also see that each filter gives an indication of how (if at all) the filter is applied. Since no filtering has been applied here, each filter shows (All) as the current selection. This tells you that all data is allowed through for the filter.

Note The Filters well only shows visual-level filters if a visual is selected.

Adding Filters

The Filters well automatically adds any fields that you use as the basis for a visualization. To see this, create the following visual.

1. Open the DataModelWithMetricsForVisualizations.pbix file.
2. Create a bar chart using these fields:
 - a. Make (from the Stock table)
 - b. Total Sales (from the InvoiceLines table)
3. The Filters well will look like Figure 13-2.

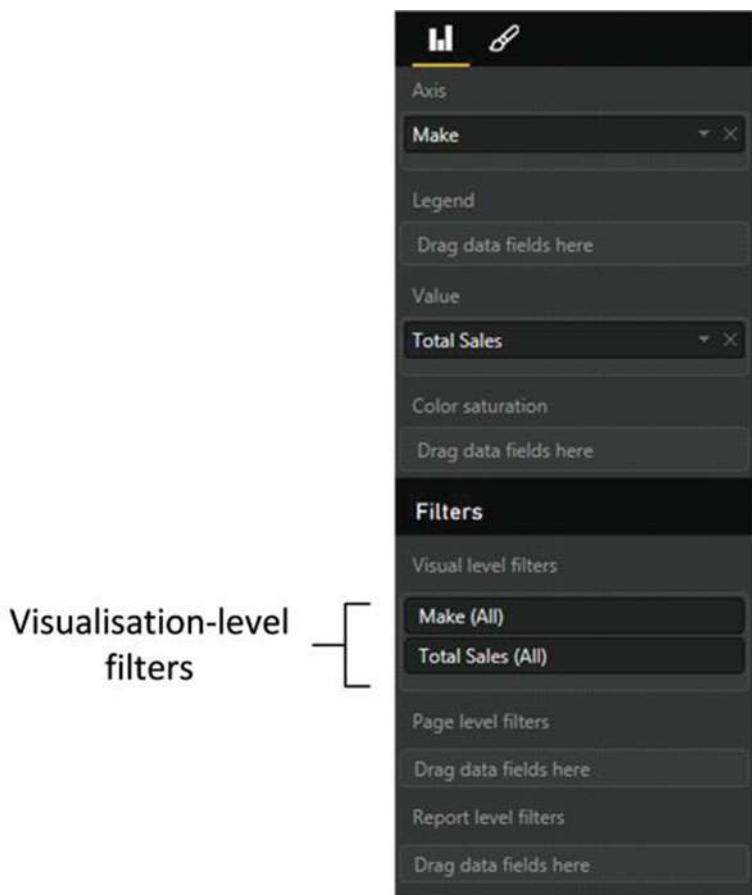


Figure 13-2. Automatic creation of visual-level filters

Figure 13-2 shows that adding a data field to a visual automatically adds the same field to the field well at the visual level.

Applying Filters

To see the filter working, let's limit the chart to displaying only a few makes:

1. Select the chart that you created previously.
2. In the Visualizations pane, hover the pointer over the Make filter in the Filters well. A downward-facing chevron will appear at the right of the field that is being used as a filter. This will be a visual-level filter.
3. Expand the Make filter by clicking the chevron. The Filters well will display all the makes that appear in the visual.

4. Select the following makes in the Filters well by selecting the check box to the left of each of the following elements:
 - a. Jaguar
 - b. Rolls-Royce
 - c. Triumph

You will see that data is only displayed for the makes of car that were selected in the filter. The resulting chart should look like Figure 13-3.



Figure 13-3. A simple filtered chart

You will have noticed that when the filter was first applied, every check box was empty, including the (All) check box. The default is (fairly logically) to set up a filter ready for tweaking, but not actually to filter any data until the user has decided what filters to apply. Once you start adding filter elements, they will be displayed in the Filters well just below the name of the field that is being used to filter data.

You modify filters the same way you apply them. All you have to do to remove a selected filter element is to click the check box with a check mark to clear it. Conversely, to add a supplementary filter element, just click a blank check box.

One final thing to note is that if you subsequently minimize the filter for a field (by clicking the upward-facing chevron that has replaced the downward-facing chevron to the right of the field name) you will now see not only the field name in the Filters well, but also a succinct description of the filter that has been applied. You can see an example of this in Figure 13-4.

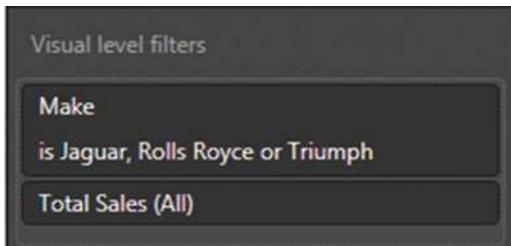


Figure 13-4. Filter description

Note You can also add fields to the Filters well before adding data fields to the visual. Simply create a blank visual and then leave it selected while you add the filter fields to the Filters well.

The (All) Filter

The only subtlety concerning simple filters is that you also have the (All) check box. This acts as a global on/off switch to select, or deselect, all the available filter elements for a given filter field. The (All) filter field has only two states:

- *Blank:* No filters are selected for this field.
- *Checked:* All filters are selected for this field.

Checking or unchecking the (All) filter field will select or deselect all filter elements for this field, in effect rendering the filter inactive. The (All) filter field is particularly useful when you want not only to remove multiple filter selections in order to start over but also want to select all elements in order to deselect certain elements individually (and avoid manually selecting reams of elements).

Note When selecting multiple elements in lists, you may be tempted to apply the classic Windows keyboard shortcuts that you may be in the habit of using in, for instance, Excel or other Windows applications. Unfortunately, Ctrl- or Shift-clicking to select a subset of elements does not work. In addition, although you can select and deselect a check box using the space bar, it is not possible to use the cursor keys to pass from one element to another in a filter list.

Clearing Filters

Setting up a finely honed filter so that you are drilling through the noise in your data to the core information can take some practice. Fortunately, the virtually instantaneous application of filters means that you can see almost immediately if you are heading down the right path in your analysis. However, there are frequent occasions when you want to start over and remove any settings for a particular filter. This can be done in one of two ways:

1. Hover the mouse pointer over the filter that you want to clear. The pointer will become a hand, and a small eraser icon will appear to the right of the filter, as shown in Figure 13-5.

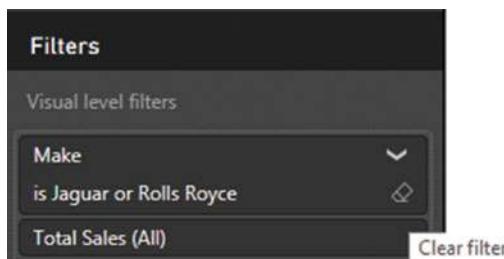


Figure 13-5. The clear filter icon

2. Click the clear filter icon (the eraser) to remove the filter settings for this field.

Alternatively, you can do the following, but *only* for basic filters:

1. Expand the filter in the Filters well by clicking the downward-facing chevron.
2. Click the (All) option to remove all filter selections.
3. Minimize the filter.

Once a filter has been cleared, the only way to get it back to its previous state is to press Ctrl+Z immediately. Otherwise, you will have to reapply the requisite criteria.

Filtering Different Data Types

So far, you have only seen how Power BI Desktop can filter text elements. Although text-based elements are a major part of many data filters, they are far from the only available type of data. There is also

- Numeric data
- Date and time data

Numeric Data

You can filter on numeric elements just as you can filter on text-based elements in Power BI Desktop. However, although the core principles are the same, there are some interface differences and tricks that you probably need to know.

Range Filter Mode

The first trick worth knowing is that, when filtering on numeric data, you do **not** have the choice of selecting elements from a list. Instead, you have a *threshold selector*, which is the only filter for numeric filters. The threshold selector allows you to set the lower and/or upper limits of the range of numbers that you want to display in a Power BI Desktop report, page, or visual.

The following explains how to set the range of figures for which data is displayed.

1. Select the chart that you created previously.
2. Clear any existing filters.
3. In the Visualizations pane, hover the pointer over the Total Sales filter in the Filters well. A downward-facing chevron will appear at the right of the field that is being used as a filter.
4. Expand the Total Sales filter by clicking the chevron. The Filters well will display two pop-ups that will enable you to set thresholds.
5. Click the upper pop-up and select the “Is greater than or equal to” option.
6. In the box under the upper pop-up, enter the value **1000000** (without any thousands separators of currency units).
7. Ensure that the And radio button is selected under the pop-up.
8. Click the lower pop-up and select the “Is less than or equal to” option.
9. In the box under the lower pop-up, enter the value **8000000** (without any thousands separators of currency units).
10. Click Apply Filter. The chart will change to show only values in the range that you set in the filter.

That is it. You have set a range for all data in the Power BI Desktop report corresponding to the selected field. It should look like Figure 13-6. It is worth noting that in this figure I have widened the Filters well so that you can see the entire text that Power BI Desktop adds to a range filter to explain what the filter does. If you leave the Visualizations pane at its default width, you will only see an abbreviated version of the filter definition.

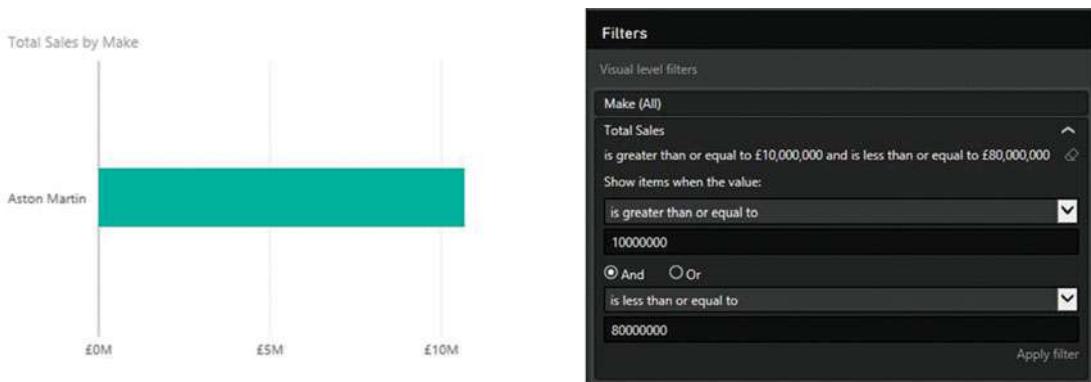


Figure 13-6. A filter range

When selecting a range of numeric data you do not have to set both upper and lower bounds. You may set one or both.

Numeric Filter Options

Numbers cannot be filtered exactly the same ways as text; the filtering options are slightly different. These filter options are described in Table 13-2.

Table 13-2. Numeric Filter Options

Filter Option	Description
Is Less Than	The selected field is less than the number you are searching for.
Is Less Than Or Equal To	The selected field is less than or equal to the number you are searching for.
Is Greater Than	The selected field is greater than the number you are searching for.
Is Greater Than Or Equal To	The selected field is greater than or equal to the number you are searching for.
Is	The selected field matches exactly the number you are searching for.
Is Not	The selected field does not exactly match the number you are searching for.
Is Blank	The selected field is blank.
Is Not Blank	The selected field is not blank.

If you are using both threshold levels to define a range of values to include or exclude, or even specific values to include or exclude, then you need to apply one of the logical filter options. These are shown in Table 13-3.

Table 13-3. Logical Filter Options

Filter Type	Comments
And	Applies both filter elements to <i>reduce</i> the amount of data allowed through the filter.
Or	Applies either of the filter elements separately to <i>increase</i> the amount of data allowed through the filter.

When applying a numeric filter you must—not altogether surprisingly—enter a numeric value. If you enter text by mistake, you will see a yellow lozenge appear at the right of the box to alert you to the fact that you entered a text by mistake, and the Apply filter text will remain grayed out.

In this case, you have to delete the characters that you entered and enter a numeric value in the place of the text.

Date and Time Data

At its simplest, date and time data is merely a list of elements or a range of numeric data. Consequently, dragging a field from a date dimension into the Filters well allows you to select one or more elements (such as years or months) or to define a range (of weeks, for instance). Take the following steps to see this in action.

1. Select the chart that you created earlier.
2. Clear any existing filters.
3. Expand the Date table in the Fields list.
4. Drag the FullYear field to the Filters well. Since Power BI Desktop assumes that the years are numbers, it switches to advanced filtering.
5. Click “Basic filtering” to revert to a list of years.
6. Click one or two years in the filter list. Figure 13-7 demonstrates this. The chart will be updated to show only data for the chosen years.

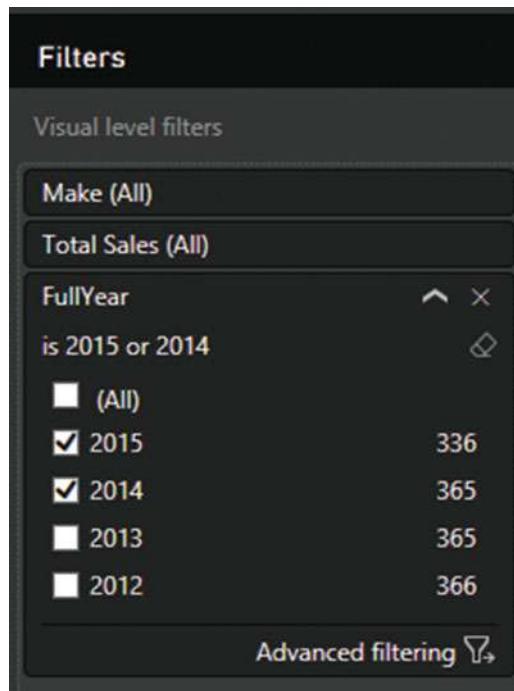


Figure 13-7. A date filter

One of the reasons that we created a date dimension is to allow you to filter on date elements this simply. So you can now use any of the fields in the DateDimension table to restrict the data that is in a visualization. In my opinion, the following are very useful fields in this dimension:

- FullYear
- MonthAndYearAbbr
- QuarterAndYear
- YearAndWeek

However, you need to remember that you can combine multiple elements in a filter to get the correct result. So there is nothing to stop you filtering on a specific year, and then adding the day of the week and calculating all the sales for the Saturdays in the year, for instance. So by combining different filter elements from a properly constructed data dimension you can look at how data varies over time incredibly easily.

This means that, when filtering by dates, you could need to apply multiple filters, where you can select elements from each of the different filters: Year, Quarter, and/or Month. Alternatively, if you will be filtering on successive elements in a date hierarchy (Year, followed by Month, for instance) you may find it more intuitive to drag the filter elements from the date hierarchy to the Filters well in the temporal order in which you will be using them (that is, Year followed by Month). This way, you can proceed in a logical manner, from top to bottom in the Filters well, to apply the date criteria that interest you.

Note If or when you want to delete filters that were added as a hierarchy, you have to delete them individually, because you cannot remove all the fields that make up the hierarchy together. Fortunately, this only takes a few seconds.

Date and Time Filters

If you are filtering on a Date or DateTime field, then you quickly notice that Power BI Desktop adds a couple of pop-up elements to the advanced filter to help you select dates and times more easily. These additions are as follows:

- A calendar pop-up that lets you click a day of the month (and scroll through the months of the year, forward and backward)
- A time series scroll filter that lets you select times to every minute throughout the day

To see this in action, imagine that you want to see all sales for a range of dates.

1. Select the chart that you created previously.
2. Clear all filters from it, as described earlier.
3. Leaving the chart selected, expand the DateDimension table and drag the DateKey field into the Filters well into the box for visual-level filters.
4. Click “Advanced filtering”.
5. From the “Show items when the value” pop-up list, select “is on or after”.
6. Click the calendar icon beneath the pop-up. The calendar pop-up is shown in Figure 13-8.

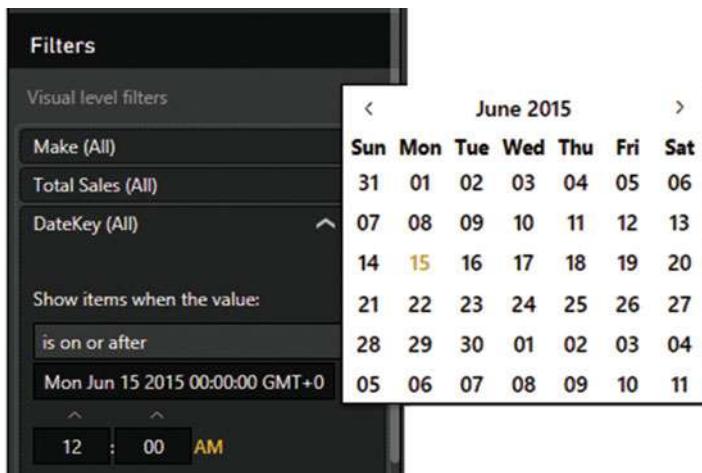


Figure 13-8. The calendar pop-up

7. Select a date from the calendar. This date will appear in the box under the calendar icon.
8. Click the And radio button.
9. In the second pop-up, select “is on or before”.
10. Click the calendar icon beneath the pop-up and select a date from the calendar. This date will appear in the box under the second pop-up.
11. Click “Apply filter”. The Filters well will look like Figure 13-9.

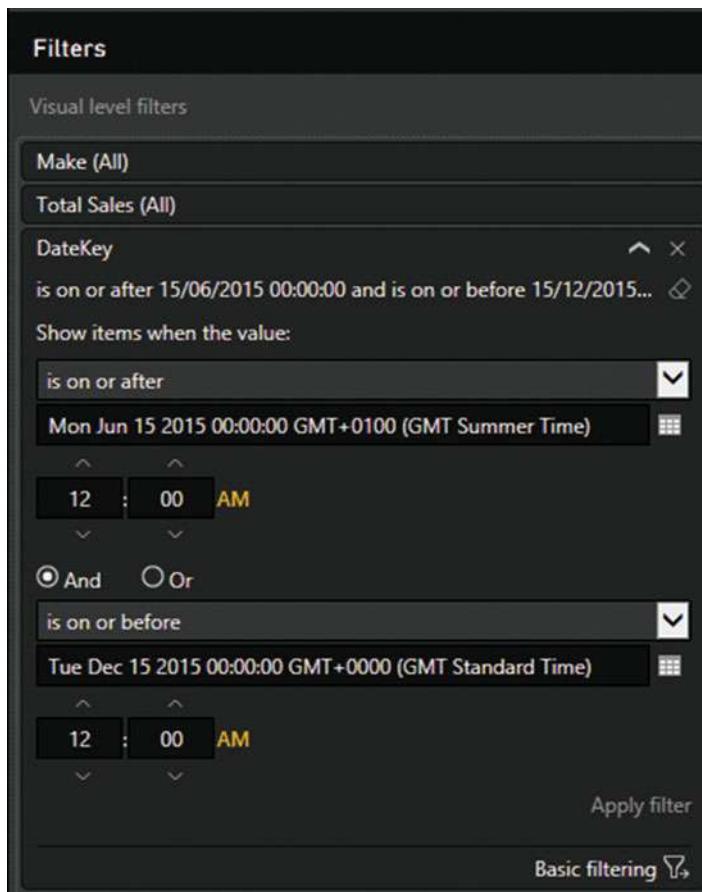


Figure 13-9. A date range filter

There are a couple of tricks that may save you time when you are selecting dates from the calendar pop-up (you may be familiar with these techniques in other desktop packages):

- When using the calendar pop-up, clicking the right-facing triangle to the right of the month and year displays the following month.
- When using the calendar pop-up, clicking the left-facing triangle to the left of the month and year displays the previous month.
- When using the calendar pop-up, clicking the month and year displays a Year pop-up, in which you can click the right-facing triangle to the right of the year to display the following year (or click the left-facing triangle to the left of the year to display the previous year), and then you can select the month from those displayed.
- When using the time pop-up, clicking inside any constituent part of the time (hour, minute, or second) and then clicking the up and down scroll triangles above and below the time field allows you to scroll rapidly through the available options.
- Clicking AM or PM to the right of the time box lets you switch from AM to PM.

If you do not want to select a date using the calendar pop-up, then you can enter a date directly in the date box of the advanced filter for a Date (or DateTime) field. Just remember that you must enter the date in the date format corresponding to the environment that you are using and that can be understood by Power BI Desktop.

Note If you enter a date where the format does not correspond to the system format, or if the date is purely and simply invalid (the 30th of February, for instance), then Power BI Desktop will not let you apply the filter. To correct this, merely select a correct date using the calendar pop-up. Similarly, if you enter a nonexistent time, the Power BI Desktop will refuse to accept it and will revert to the previous (acceptable) time that was chosen.

Date Filter Options

Dates also cannot be filtered in exactly the same ways as text or numbers. Consequently, the advanced filtering options for date filters are slightly different from those used when filtering other data types. They are described in Table 13-4.

Table 13-4. Advanced Date Filter Options

Filter Option	Description
Is	The selected field contains the date that you are searching for.
Is Not	The selected field does not contain the date that you are searching for.
Is After	The selected field contains dates after the date that you entered; that is, later dates that do not include the date you entered.
Is On Or After	The selected field contains dates beginning with the date that you entered or later.
Is Before	The selected field contains dates before the date that you entered; that is, earlier dates, not including the date you entered.
Is On Or Before	The selected field contains dates on or before the date that you entered; that is, earlier dates, up to and including the date you entered.
Is Blank	The selected field is blank.
Is Not Blank	The selected field is not blank.

Other Data Types

There are other data types in the source data that you are likely to be handling. You might have Boolean (True or False) data, for instance. However, for Power BI Desktop, this is considered, for all intents and purposes, to be a text-based filter. On the other hand, there are data types that you cannot filter on and that do not ever appear in the Filters well. Binary data (such as images) is a case in point.

So if you filter on Boolean data, Power BI Desktop displays True and False in the expanded filter for this data type. The following explains how to see this.

1. Create a chart using the following fields
 - a. IsCreditWorthy (from the Clients table)
 - b. Total Sales (from the InvoiceLines table)
2. Expand the IsCreditWorthy field in the Filters well.
3. Select True and Blank. The chart (and Filters well) will look like they do in Figure 13-10.

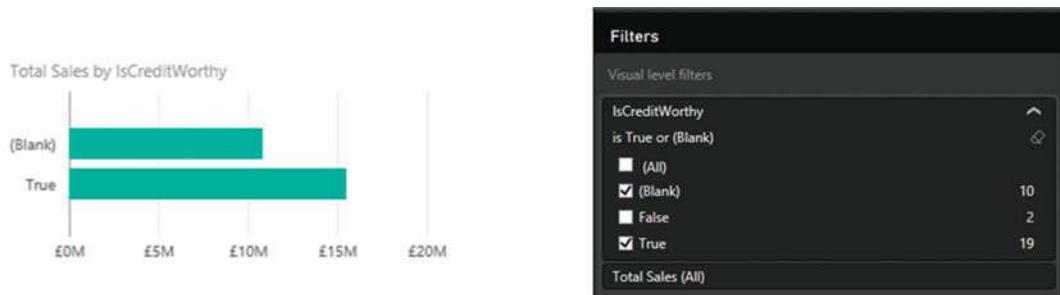


Figure 13-10. Applying a Boolean filter

Advanced Text Filters

In many cases, when you are delving into your data, merely selecting a “simple” filter will be enough to highlight the information that interests both you and your audience. There will inevitably be cases when you need to filter your data more finely in order to return the kinds of results that sort the wheat from the chaff. This is where Power BI Desktop’s advanced filtering capabilities come to the fore. Advanced filtering lets you search inside field data with much greater precision, and it is of particular use when you need to include, or exclude, data based on parts of a field if it is text.

Applying an Advanced Text Filter

Let’s begin with a simple example of how to apply an advanced filter to a text field.

1. Click the visual that you want to filter. Once again, we will use the chart that you created earlier in this chapter.
2. Clear any existing filters.
3. Expand the Make field in the Filters well (unless it has already been done).
4. Click the Advanced Filtering Mode icon at the bottom of the list of text elements for this field. The body of the filter switches to show the Advanced filter pop-ups and boxes, and the text under the filter title now reads “Show items when the value”.
5. Select Contains from the pop-up.

6. Click inside the filter text box (under the box displaying Contains) and enter the text to filter on (*aston* in this example).
7. Click Apply Filter or press the Enter key. All objects in the Power BI Desktop report will only display data where the client contains the text *aston*. The result is shown in Figure 13-11; the advanced filter used to produce it is also shown.

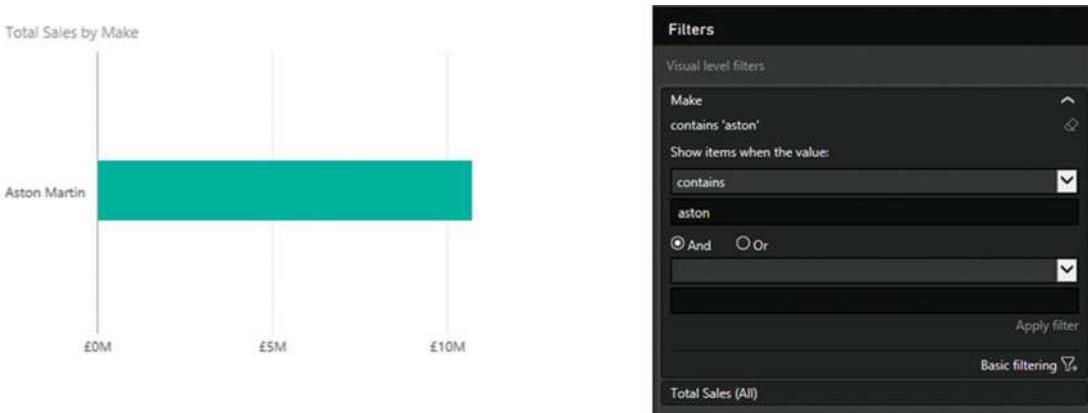


Figure 13-11. The results of applying an advanced filter

Here are several comments that it is important to make at this stage:

- Advanced filtering is *not* case sensitive. You can enter uppercase or lowercase characters in the filter box; the result is the same.
- Spaces and punctuation are important, as they are taken literally. For instance, if you enter *A* (with a space after the *A*), then you only find elements containing an *A* (uppercase or lowercase) followed by a space.
- Advanced filters, just like standard filters, are cumulative in their effect. So, if you have applied a filter and do not get the results you were expecting, be sure to check that no other filter at another level is active that might be narrowing the data returned beyond what you want.
- If your filter excludes all data from the result set, then any tables in the Power BI Desktop report displays This Table Contains No Rows.
- Similarly, if your filter excludes all data, charts will be empty.

In any case, if you end up displaying no data, or data that does not correspond to what you wanted to show, just clear the filter and start over.

Clearing an Advanced Filter

Inevitably, you will also need to know how to remove an advanced filter. The process is the same as for a standard filter; all you have to do is click the Clear Filter icon at the top of the filter for this field (just under the chevron for the field in the field well). The filter elements are removed for this filter.

Reverting to Basic Filtering

If you decide that you no longer wish to define and use a complex filter, but you wish to revert to basic filtering, then all you have to do is click “Basic filtering” at the bottom of the filter elements for the selected field.

The Filters well will switch to basic filtering for the selected field.

Text Filter Options

When filtering on the text contained in a data field, you can apply the string you are filtering on to the underlying data in several ways. These are the same for both the upper and lower of the two advanced filter options for a text field. They are described in Table 13-5.

Table 13-5. Advanced Text Filter Options

Filter Option	Description
Contains	The selected field contains the search text anywhere in the field data.
Does Not Contain	The selected field does not contain the search text anywhere in the field data.
Starts With	The selected field begins with the search text, followed by any data.
Does Not Start With	The selected field does not begin with the search text, followed by any data.
Is	The selected field matches the search text exactly.
Is Not	The selected field does not match the search text exactly.
Is Blank	The selected field is blank.
Is Not Blank	The selected field is not blank.

You are not limited to setting a single advanced text filter. Just as was the case for numeric values, you can set two filters and apply either of them (by setting the logical operator to Or). Alternatively, or you can set two complementary text filters by setting the logical operator to And.

Specific Visualization-Level Filters

So far in this chapter, we have looked at filters where the fields that were used to filter a visual were *also* visible in the actual visual—be it a chart, map, or table. Inevitably, there will be times when you will want to filter on a field that is *not* displayed in a visual. Although we touched on this earlier, it is worth explaining the concept in greater detail.

The following explains how to apply a visualization-level filter.

1. Select (or re-create) the chart of Total Sales by ClientName that you created previously.
2. Display the Filters well (unless it is already visible).
3. Expand the Colors table and drag the Color field into the Filters well. The list of colors will be expanded automatically in the Filters well. The color field will not be displayed in the visual, however.
4. Select a couple of colors, such as Night Blue and Silver. The result is shown in Figure 13-12.



Figure 13-12. A visualization-level filter without using the data field

You will notice right away that the filter(s) that you have applied only affect the selected visualization (the chart in this example). When you create complex reports that contain several visualizations, you will see that no other visualizations in the report have their underlying data modified in any way.

You can clear any filter at the visualization level by clicking the Clear icon at the top right of the filter name. Moreover, you can add multiple fields to the Filters well for a visual *and* you can add them in any order.

Note Removing a field from the field well will not remove this field from the Filters well if the filter is active. This can be deceptive because the field is no longer displayed in the visual; however, its effects can still be seen.

Multiple Filters

So far, we have treated filters as if only one was ever going to be applied at a time. Believe me, when dealing with large and intricate datasets, it is unlikely that this will be the case. As a matter of course, Power BI Desktop will let you add multiple filters to a report. This entails some careful consideration of the following possible repercussions:

- All filters are active at once (unless you have cleared a filter) and their effect is cumulative. That is, data will only be returned if the data matches *all* the criteria set by all the active filters. So, for example, if you have requested data between a specified date range and above a certain sales figure, you will not get any data in which the sales figure is lower than the figure that you specified or with a sales date before or after the dates that you set.
- It is easy to forget that filters can be active. Remember that all active filters in the Filters well remain operational whether the Filters well itself is expanded or collapsed. If you are going to collapse filters to make better use of the available space on the screen, then it is worth getting into the habit of looking at the second line below any filter title that will give you a description of the current filter state. It will display something like Contains Rolls. Of course, the exact text varies according to the filter that you have applied.

Page-Level Filters

Now that you have seen what filters are and how you can apply them to visuals, it is time to extend the concept and see how filters can be applied to multiple visuals.

The good news is that all filters are configured in exactly the same way whatever the level at which they are applied. Consequently, applying filters at page- or report-level is simply a question of choosing where in the Filters well to place a filter.

As an example, suppose that you want to filter all the visuals on a page to display data for a specific year.

1. In an open Power BI Desktop file, click the dashboard canvas outside any existing visuals.
2. Expand the DateDimension table.
3. Drag the FullYear field onto the field well into the Page level filters box. The advanced filtering options for this field will be displayed.
4. Click “Basic filtering” at the bottom of the filter details for this field.
5. Select 2014 from the list of available years. All visuals on the current page will be filtered to display only data for this year.

Figure 13-13 shows you the Filters well for this operation. You can see that a page-level filter looks identical to a visual-level filter. The only difference (apart from the effect that it produces) is the position in the Filters well.

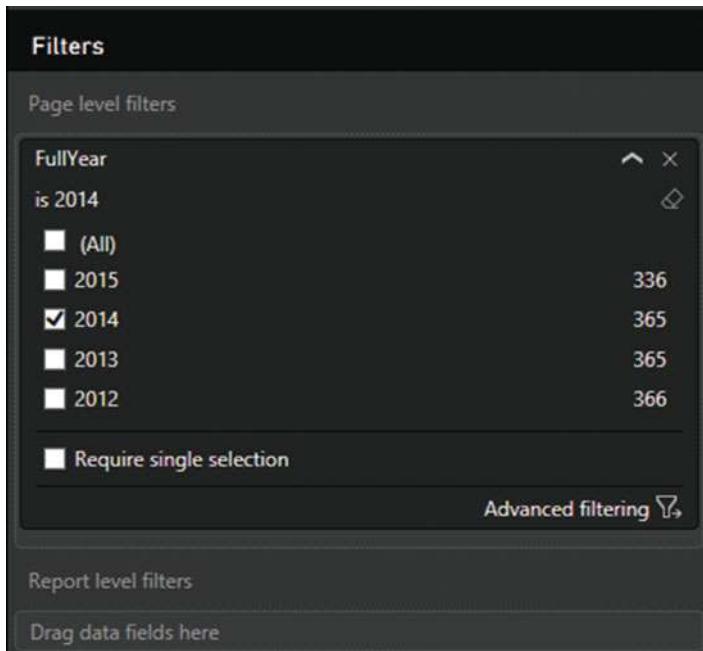


Figure 13-13. Applying a page-level filter

You can add multiple page-level filters if you wish simply by dragging further fields into the field well and into the “Page level filters” box. The order in which the fields are added is unimportant. You will notice that the page-level filters remain visible whether you have selected a visual or not.

Report-Level Filters

The highest level of filtering is applied at report level. This means that any filter set here will apply to every page (or dashboard) in a report and consequently also to every visual in the file. Applying report-level filters is virtually identical to the application of page-level filters. So, let’s suppose that your entire report covers a single country. Here is how to set a filter that will apply to every page in the Power BI Desktop report.

1. In an open Power BI Desktop file, click the dashboard canvas outside any existing visuals.
2. Expand the Countries table.
3. Drag the CountryName field onto the field well into the Report level filters box. The advanced filtering options for this field will be displayed.
4. Click “Basic filtering” at the bottom of the filter details for this field.
5. Select United Kingdom from the list of available countries. All visuals in the current report (on every page) will be filtered to display only data for this year.

Figure 13-14 shows you the Filters well for this operation.

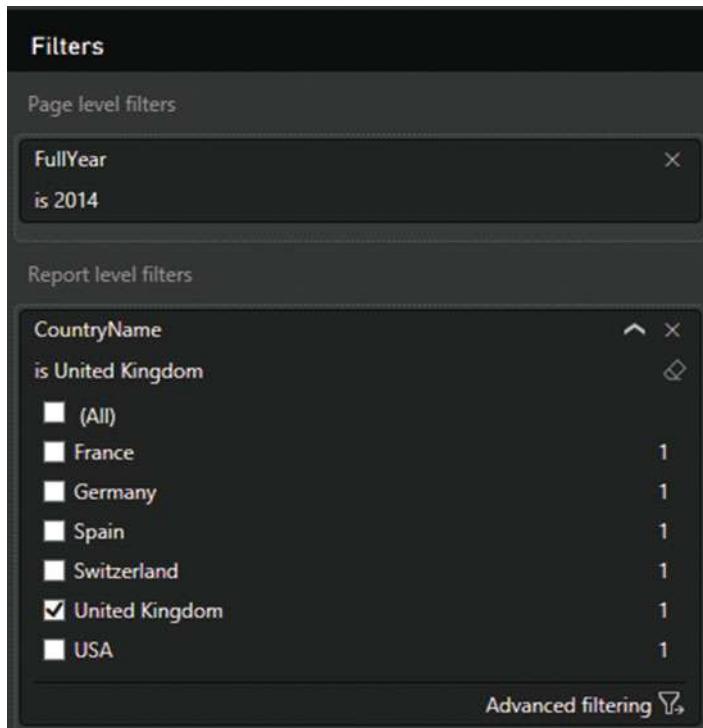


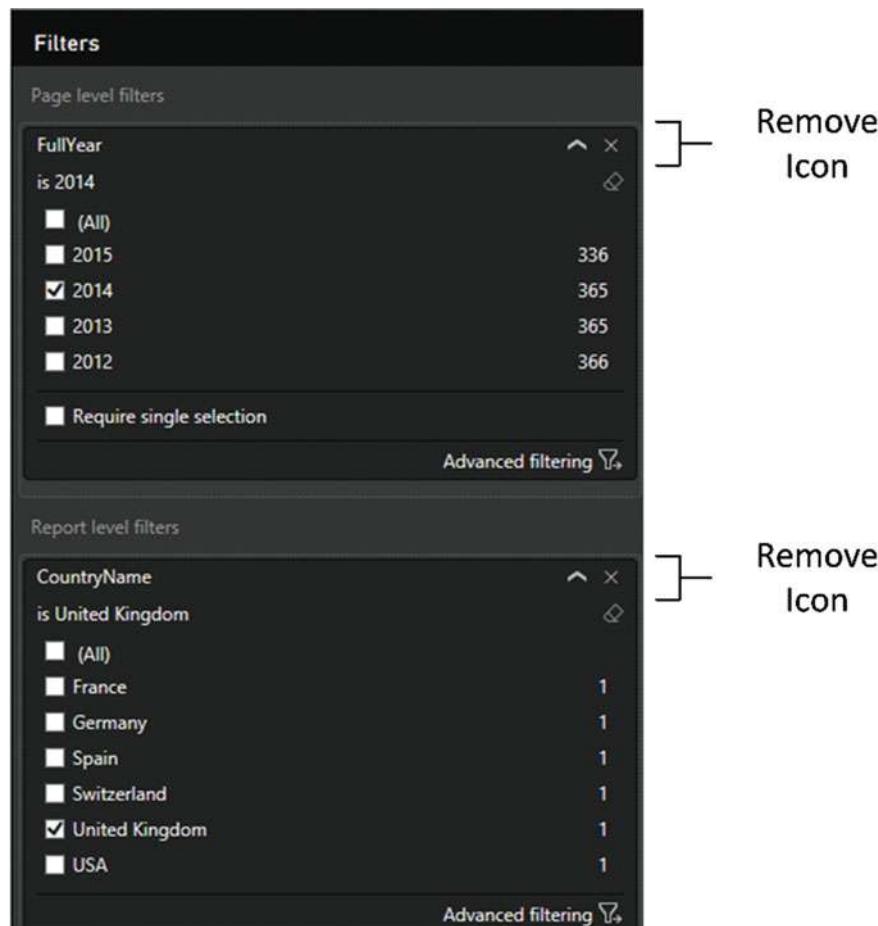
Figure 13-14. Applying a page-level filter

You can add multiple report-level filters if you wish simply by dragging further fields into the field well and into the Report level filters box. Once again, the order in which the fields are added is irrelevant. You will notice that the report-level filters remain visible whether you have selected a visual or not.

Removing Filters

When working with filters, at times you may want to clear the decks and start over. The fastest way to do this is to delete a filter; once a filter is deleted, it no longer has any effect on the data in the Power BI Desktop report. This can be done as follows.

1. Click the Remove Filter icon to the right of the selected filter. This icon is shown in Figure 13-15.

**Figure 13-15.** Removing a filter

The art here is to ensure that you have selected the correct filter to remove. The technique, however, is the same for all report- and page-level filters, as well as for visual-level filters that are not used to display data.

Once a filter has been removed, the only way to get it back is to click Undo (or press Ctrl+Z) immediately; otherwise, you will have to rebuild it from scratch. Interestingly, although you can add filters by dragging elements into the Filters well, you cannot drag them out of the Filters well to remove them.

However, you can remove visualization-level fields (that are not used as data for the visual) from the Filter well by clicking the cross at the right of the field name.

Note Visual-level filters cannot be removed, only reset to empty.

Filter Field Reuse

Although it may seem counterintuitive, you can reuse the same field at the same filter level to assist you in certain cases.

As an example of this, imagine that you want to see all the mileage for vehicles between clearly defined thresholds. Take a look at the following example.

1. Expand the Stock table.
2. Drag the Mileage field into the Field well and place it in the Page box.
3. Select “is greater than” from the first pop-up.
4. Enter **50000** in the box for the first threshold.
5. Select “is less than” from the second pop-up.
6. Enter **70000** in the box for the first threshold.
7. Click “Apply filter”.
8. Drag the Mileage field into the Field well and place it in the Page box under the filter that you just created.
9. Click “Basic filtering” for this filter. A list of available mileage for vehicles in stock (as well as the number of cars for each mileage figure) will appear. You can see this in Figure 13-16.

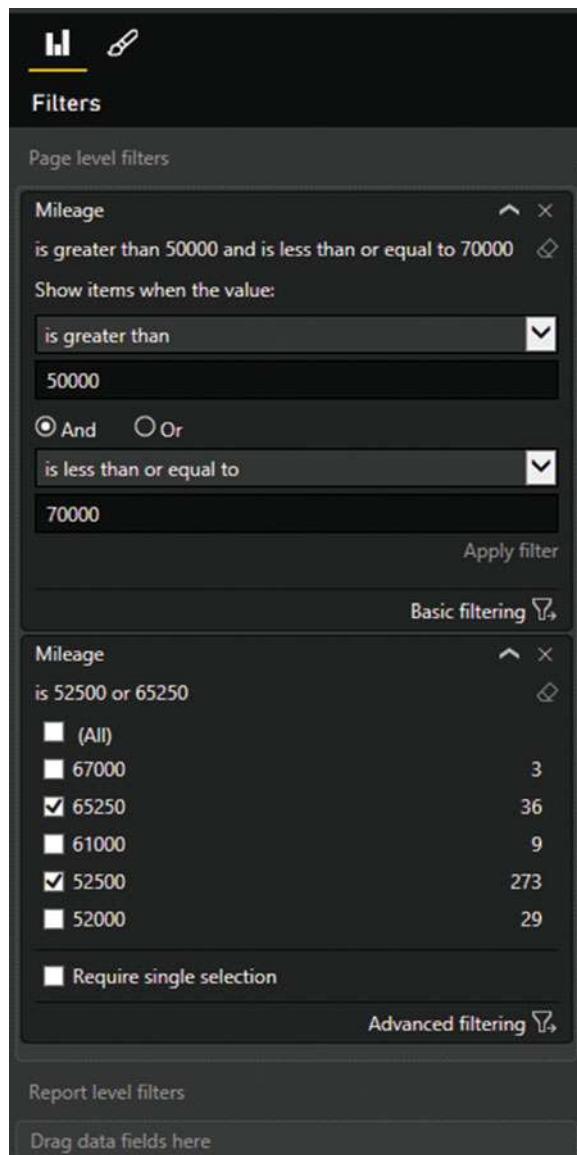


Figure 13-16. Combining multiple iterations of the same field in a filter

You can now use the more detailed filter to filter the visuals on the current page.

The interesting thing to note is that a hierarchy of filters is applied, even inside a filter box in the Filters well. Put simply, a filter that is placed above another filter will filter the available elements in the lower filter.

Note This example was set to filter at page level. It could equally well have been applied at report level.

Using the Filter Hierarchy

Given the multiple levels of filters that can be applied, a hierarchy of filters is applied in Power BI Desktop:

- First, at the data level, any selections or choices you apply to the underlying data restrict the dataset that Power BI Desktop can use to visualize your information.
- Second, at the report level, any report-level filters that you apply affect all visualizations in the report using the (possibly limited) available source data.
- Third, at the page level, any page-level filters that you apply will affect all visualizations on the view, using the (possibly limited) available source data filtered by any report-level filters.
- Finally, for each visualization, any visualization-level filters that you apply will further limit the data that is allowed through the report- and page-level filters—but only for the selected visualization.

It is worth noting the following points:

- You have no way to apply a completely different selection to a visualization filter if it has been filtered out at a higher (report or page) level. Clicking (All) will only select from the subset of previously filtered elements.
- If you apply a filter at visualization level and then reapply the same filter at report or page level, but with different elements selected, you will still be excluding all non-selected elements from the filter at visualization level. I stress this because Power BI Desktop will remember the previously selected elements at visualization level, and leave them visible even if they cannot be used in a filter, because they have already been excluded from the visualization-level filter by being ruled out at view level. In my opinion, this adds a certain visual confusion, even if the hierarchical selection logic is applied.

Hopefully, this shows you that Power BI Desktop is rigorous in applying its hierarchy of filters. Should you need to apply a filter at visualization level when the filter choice is excluded at report or page level, you have no choice but to remove the filter at the higher level and then reapply visualization-level filters to all necessary visualizations.

Filtering Tips

Power BI Desktop makes it incredibly easy to filter data and to exclude any data that you feel is not helpful in your data analysis. However, like many powerful tools, this ability to apply filters so quickly and easily can be something of a double-edged sword. So here are a few words of advice and caution when applying filters to your data.

Don't Filter Too Soon

As an initial point, I would say that a key ground rule is “Don’t filter too soon.” By this, I mean that if you are examining data for trends, anomalies, and insights, you have to be careful not to exclude data that could contain the very insights that can be game changing.

The problem is that when you first delve into a haystack of data in search of needles of informational value, you have no idea what you could be looking for. So I can only suggest the following approaches:

- Begin with no filters at all and see what the data has to say in its most elemental form.
- Apply filters one at a time and remember to delete a filter before trying out another one.
- Try to think in terms of “layers” of filters. So, once you have defined an initial set of filters, add further filters one by one.
- Go slowly. The temptation is to reach a discovery in order to shout about it from the rooftops. This can lead to excessive filtering and unreliable data.
- Always remove any filters that are not absolutely necessary.
- Be careful if you hide the Filters well. It is too easy to forget that there are active filters if they are not visible in some way.
- Remember that you can have filters specific to a visualization that might not be immediately visible in the Filters well without scrolling. So always check if any visualization filters are active for each table and chart in a report.

Annotate, Annotate, Annotate

If you are presenting a key finding based on a dataset, then it can save a lot of embarrassment if you make it clear in every case what the data does and does not contain. For example, you could be so pleased with the revelatory sales trend, that you have discovered that you forgot to note an important exclusion in the underlying data. Now, no one is suggesting that you are doing anything other than making a point, but your audience needs to know what has been excluded and why—just in case it makes a difference. After all, you don’t want a workplace rival pointing this out to invalidate your findings in the middle of a vital meeting, do you?

Annotation techniques are described in Chapter 15 if you need to jump ahead to check this out now.

Avoid Complex Filters

Power BI Desktop filters are designed to be intuitive and easy to use. A consequence of this is that they can prove to be a little limited when you need to apply very complex filters—be they text, numeric, or date filters.

If you need to create a complex filter, it’s probably best *not* to create one in the report. Instead, consider trying to filter source data using Power BI Desktop Query. Should you need a more interactive way of switching between complex filter settings in a report, then use DAX to define columns that display a text as the result of a filter (as described in Chapter 7), then use the result of the filter as a selection. As a very simple example of this, consider the MileageRange column that you defined in Chapter 6. In effect, this groups vehicle mileage into certain bandings. It follows that selecting one or more bandings in a filter will restrict the display to data that matches the predefined data ranges. With a little practice, you can extend this technique to create quite complex filters in DAX at the data level.

Conclusion

This chapter has shown you how to apply and fine-tune a series of techniques to enable you to select the data that will appear in your Power BI Desktop reports. The main thing to take away is that you can filter data at three levels: the overall report, each page, and each visualization on a page.

You have also seen a variety of selection techniques that allow you to subset data. These range from the avowedly simple selection of a few elements to the specification of a more complex spread of dates or values. Finally, it is worth remembering that you can filter data using any of the fields in the underlying dataset, whether the field is displayed in a visual in a report or not.

CHAPTER 14



Using Slicers

With your filters in place, you now have some extremely powerful and insightful dashboards ready to be paraded in front of your colleagues, bosses, and clients. Yet static illustrations can only tell a story in a certain way. What you need to clinch the deal or convince an audience is some truly telling interaction with your facts and figures. Once again, Power BI Desktop is the tool of choice, as it highlights the key metrics in your presentation with a single click—and makes your point, simply and elegantly.

Put less breathlessly, you can interact with your filtered data in Power BI Desktop reports to subset or isolate metrics. These elements have the following characteristics. They are

- Always visible in the Power BI Desktop report
- Instantly accessible
- Interactive
- Clearly indicate which selections are being applied

So what are the effects that you can add to a Power BI Desktop report to select and project your data? Essentially, they boil down to two main approaches

- Slicers
- Highlighting

These interactive elements can be considered to function as a supplementary level of filtering. That is, they take the current filters that are set in the Filters well (at any level) and then provide further fine-grained *interactive* selection on top of the dataset that has been allowed through the existing filters. Each approach has its advantages and limitations, but used appropriately, each gives you the ability not only to discover the essence of your data, but also to make your point clearly and effectively.

You will use the `DataModelWithMetricsForVisualizations.pbix` sample file as the basis for all the slicers that you create in this chapter.

Slicers

A key form of interactive filter in Power BI Desktop is the *slicer*. This is, to all intents and purposes, a standard multiselect filter, where you can choose one or more elements to filter data in a report.

The essential difference is that a slicer remains visible on the Power BI Desktop report, whereas a filter is normally hidden. So this is an overt rather than a covert approach to data selection that makes the selection criteria immediately visible. Moreover, you can add multiple different slicers to a Power BI Desktop report and consequently slice and dice the data instantaneously and interactively using multiple criteria.

Slicers can be text-based, or indeed, they can be simple charts, as you will soon see.

Adding a Slicer

To appreciate all that slicers can do, we need to see one in action. This means having at least one standard visual in a page so that you can see the result of applying a slicer. To test a slicer

1. Create a table (to show the effect of using a slicer) using the following fields:
 - a. CountryName (from the Countries table)
 - b. SalePrice (from the InvoiceLines table)
2. Resize the table so that it fits the data.
3. In the Fields list, expand the Colors table.
4. Drag the Color field to an empty part of the dashboard canvas. It will become a single-column table.
5. Click the Slicer button in the Visualizations gallery. The table of colors will become a slicer. The slicer icon is shown in Figure 14-1.



Figure 14-1. The slicer icon

6. Adjust the size of the slicer to suit your requirements using the corner or lateral handles. Power BI Desktop will add a vertical scroll bar if the slicer contains many elements to indicate that there are further elements available.

You can recognize a slicer by the small squares to the left of each element in the list. This way you know that it is not just a single-column table. Figure 14-2 shows a slicer using the Color field.

CountryName	SalePrice	Color
	£141,250	<input type="checkbox"/> Black
France	£2,524,510	<input type="checkbox"/> Blue
Germany	£145,750	<input type="checkbox"/> British Racing Green
Spain	£207,750	<input type="checkbox"/> Canary Yellow
Switzerland	£1,440,970	<input type="checkbox"/> Dark Purple
United Kingdom	£15,725,000	<input type="checkbox"/> Green
USA	£11,653,960	<input type="checkbox"/> Night Blue
Total	£31,839,190	<input type="checkbox"/> Pink
		<input type="checkbox"/> Red
		<input type="checkbox"/> Silver

Figure 14-2. A slicer

Tip If the Slicer icon is grayed out, then check that the table that you are trying to convert to a slicer only has one column (that is, one field in the Values box of the field well for this visual).

You can create *multiple* slicers for each page. All you have to do is repeat steps 2 through 5 for adding a slicer using a different field as the data for the new slicer.

Applying Slicers

To apply a slicer and use it to filter data on a page, click a single element in the slicer or Shift-click (or Ctrl-click) multiple elements.

All the objects in a Power BI Desktop page are filtered to reflect the currently selected slicer list. In addition, each element in the slicer list that is active (and consequently used to filter data by that element) now has a small rectangle to its left, indicating that this element is selected.

Figure 14-3 shows what happens when the slicer defined for Figure 14-2 is applied to the visualization shown in Figure 14-3.

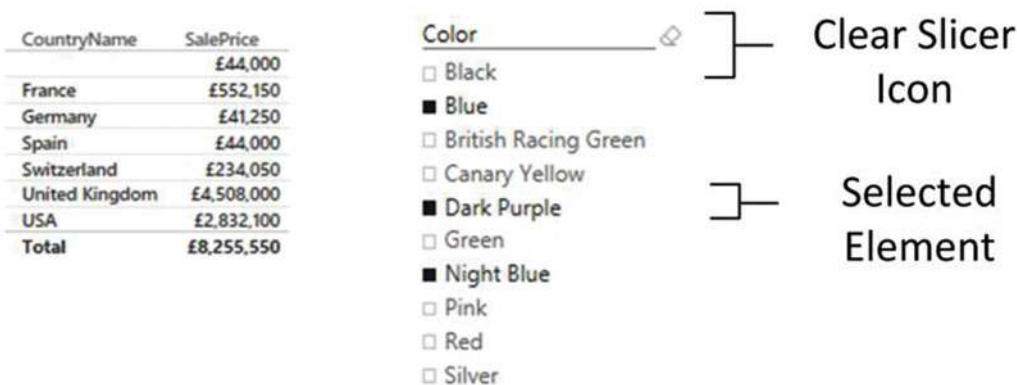


Figure 14-3. Applying a slicer

When you apply a slicer, think filter. That is, if you select a couple of elements from a slicer based on the CountryName field, as well as three elements based on the Color fields, you are forcing the two slicers (filters) to limit all the data displayed in the page to two countries that have any of the three colors that you selected. The core difference between a slicer and a filter is that a slicer is always visible—and that you have to select or unselect elements, not ranges of values.

If you experiment, you will also see that you cannot create a slicer from numeric fields in the source data. A slicer has to be based on a text field. If you need slicers based on ranges of data, then you will need to prepare these ranges in the data model. The CarAgeBucket field is an example of this. Chapter 7 explains how to add these sorts of fields to a data model.

Tip If you Shift-click or Ctrl-click all the selected elements in a slicer, you can unselect all the data it represents. This will not clear the Power BI Desktop report, however. Unselecting everything is the same as selecting everything—despite the fact that the selection squares are no longer visible to the left of each element in the slicer.

Clearing a Slicer

To clear a slicer and stop filtering on the selected data elements in a view, click the Clear Filter icon at the top right of the slicer. This icon is pointed out in Figure 14-3.

Any filters applied by the slicer to the view are now removed. You will see that each element in the slicer list now has a small empty rectangle to its left, indicating that this element is not selected. No data is now filtered out of the report.

Tip Another technique used to completely clear a slicer is to Shift-click (or Ctrl-click) the last remaining active element in a slicer. This leaves all elements inactive. Additionally, you can Ctrl-click to select every item. So, in effect, removing all slicer elements is the same as activating them all.

Deleting a Slicer

To delete a slicer and remove all filters that which it applies for a view, select the slicer and press the Delete key.

Any filters applied by the slicer to the view as well as the slicer itself are now removed.

You can even copy and paste slicers if you wish. This is very useful when you are copying slicers across different Power BI Desktop reports.

Note that if you intend to use the field that was the basis for a slicer in a table or chart you do not need to delete the slicer and re-create a table based on the same underlying field. You can merely

1. Select the slicer.
2. Click the Table button in the Design ribbon, and select the type of table (table, matrix, or card) to which you want to convert the slicer.

The instant that a slicer becomes a table, it ceases to subset the data in the Power BI Desktop report.

Modifying a Slicer

If all you want to do is replace the field that is used in a slicer with another field, then it is probably simplest to do this:

1. Select the slicer that you want to modify.
2. Drag the new field over the existing field in the field well.

The current slicer field is replaced by the new field and the slicer updates to display the contents of the field that you added. Alternatively, you can delete the slicer and re-create it.

Note When you save a Power BI Desktop file containing Power BI Desktop reports with active slicers, the slicer is reopened in the state in which it was saved.

When you start applying slicers to your Power BI Desktop reports, you rapidly notice one important aspect of the Power BI Desktop filter hierarchy. A slicer can only display data that is not specifically excluded by a report- or page-level filter. For instance, if you add a Color filter at page level and select only certain colors in this filter, you are only able to create a slicer that also displays this subset of colors. The slicer

is dynamic and reflects the elements that can be displayed once any report and page filters have been applied—just like any other visual. Consequently, adding or removing elements in a filter causes these elements to appear (or disappear) in a slicer that is based on the filtered field.

If you wish, you can apply a filter specifically to a slicer. This allows you to restrict the elements that appear in a slicer. If you want to do this, then you must apply the filter after step 3 in the previous example and *before* you convert the table to a slicer. However, be aware that the slicer itself does *not* give you any indication that it is being filtered; that is, there is no visual-level filter displayed in the Filters well.

Conversely, to remove a filter from a slicer, you need to switch it back to a table and remove the filter, and then switch back to a slicer.

Formatting Slicers

Slicers can be formatted just like any other Power BI Desktop visual. Indeed, many of the techniques that you use to format slicers are identical to those that you have already seen when formatting tables and matrices. Consequently, to avoid pointless repetition, this section concentrates on any formatting attributes that are slicer-specific, and only refers in passing to formatting approaches that you have already covered in previous chapters.

Slicer Orientation

To help you to make the best possible use of report, real estate slicers can be configured to appear vertically (as in Figure 14-2) or horizontally. To switch a slicer's orientation

1. Select the slicer that you want to modify. (I selected the colors slicer that you created previously.)
2. In the Visualizations pane, click the Formatting icon.
3. Expand the General tab.
4. In the Orientation pop-up, select Horizontal.
5. Resize the slicer to suit your dashboard. A horizontal slicer will look something like the one in Figure 14-4.

Color									
Black	Blue	British Racing Green	Canary Yellow	Dark Purple	Green	Night Blue	Pink	Red	Silver

Figure 14-4. A horizontal slicer

You probably noticed as you resized the slicer that Power BI Desktop will alter the number of rows of text as well as the width of text elements when you resize the slicer. It follows that you are best advised to experiment when altering slicer height and width in order to get the effect that suits you best.

Modifying the Outline

For Power BI Desktop, the outline is the line that separates the title from the items in a slicer. The following discusses how you can format this particular element.

1. Select the slicer that you want to modify. (I used the horizontal colors slicer that you created earlier.)
2. In the Visualizations pane, click the Formatting icon.
3. Expand the General tab.
4. Click the “Outline weight” up and down triangles to set the weight of the outline to 3 points.
5. Click the “Outline color palette” pop-up and choose a color. The separator line will change to reflect the modifications that you have made.

Adjusting Selection Controls

If you are not happy with the way that Power BI Desktop lets you select items in a slicer, then you can adjust the way that you interact with this particular visual. While the Power BI default mode of interaction is probably sufficient in most cases, it certainly does no harm to know that there are other ways of selecting elements in slicers.

Adding or Removing the Select All Box

1. Select the slicer that you want to modify. (I used the horizontal colors slicer that you created earlier.)
2. In the Visualizations pane, click the Formatting icon.
3. Expand the “Selection controls” tab.
4. Slide the “Select all” switch to the On position. This will add a Select All item to the top (or left) of the slicer.

Should you *not* want a Select All item in a slicer, all you have to do is ensure that the Select All switch is set to Off in step 4.

Enabling Single Select

A slicer’s default mode is to select only a single item, unless the user Ctrl-clicks several items. If you prefer to add items one at a time—and deactivate them individually too—you can enable Single Select to do just this.

1. Select the slicer that you want to modify. (I used the horizontal colors slicer that you created earlier.)
2. In the Visualizations pane, click the Formatting icon.
3. Expand the “Selection controls” tab.
4. Slide the Single Select all switch to the Off position. This will set the slicer interaction to single select.

Setting the Exact Size and X and Y coordinates of a Slicer

If you want to place a slicer with total accuracy on a dashboard canvas, then you can set the X and Y (horizontal and vertical) coordinates for the slicer. You can also specify its exact height and width. The following explains how.

1. Select the slicer that you want to modify.
2. In the Visualizations pane, click the Formatting icon.
3. Expand the General tab.
4. Replace the X Position, Y Position, Width, and Height values with the pixel values that define the size and position of the slicer that you wish to apply.

Slicer Header

Should you want to add some visual pizazz to a slicer, you can tweak the display of the slicer header. Here is an example of how to do this.

1. Select the slicer that you want to modify. (I used the horizontal colors slicer that we created earlier.)
2. In the Visualizations pane, click the Formatting icon.
3. Expand the “Header controls” tab.
4. Ensure that the Header switch is in the On position.
5. Click the “Font color palette” pop-up and choose a color for the header text.
6. Click the “Background color palette” pop-up and choose a color for the header background.
7. Adjust the “Text size” slider to tweak the size of the header text.

Slicer Items

Slicer items are the individual elements that make up the list of data that appear in a slicer, based on the underlying field. These, too, can be formatted to focus the attention of the reader.

1. Select the slicer that you want to modify. (I used the horizontal colors slicer that you created earlier.)
2. In the Visualizations pane, click the Formatting icon.
3. Expand the Items tab.
4. Ensure that the Header switch is in the On position.
5. Click the “Font color palette” pop-up and choose a color for the item text.
6. Click the “Background color palette” pop-up and choose a color for the background of each item.
7. Adjust the “Text size” slider to tweak the size of the text of each item.
8. Click the Outline pop-up and select Frame.

You can also format the following aspects of a slicer

- Background
- Title
- Lock aspect ratio

However, since all of these are identical to formatting attributes that apply to tables and charts (and that you have seen in previous chapters), I will not repeat them.

Finally, Figure 14-5 shows you a slicer with many of the formatting options that you just saw.

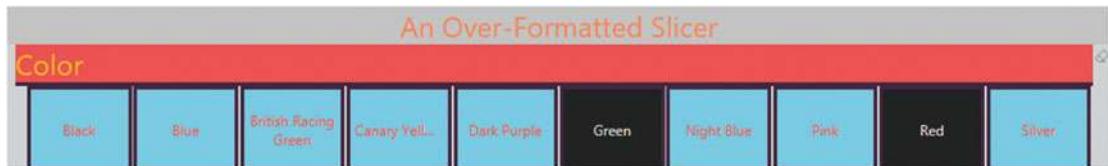


Figure 14-5. A formatted slicer

Using Charts As Slicers

You have seen how a table can become a slicer, which is a kind of filter. Well, charts can also be used as slicers. Knowing how charts can affect the data in a Power BI Desktop report can even influence the type of chart that you create, or your decision to use a chart to filter data, rather than a standard slicer. Charts can be wonderful tools to grab and hold your audience's attention—as I am sure you will agree once you have seen the effects that they can produce.

Charts As Slicers

To begin with, let's see how a chart can be used to act as a slicer to filter data interactively for any or all of the visualizations in a Power BI Desktop report. Initially, let's assume that we are aiming to produce a report using two objects:

- A Net Margin by Color table
- A Net Sales by Make column chart

Let's start with the Net Margin by Color table. It is principally used to show the effect that using a chart as a slicer in a Power BI Desktop report has on other objects. As an added extra, you will apply a filter to the page to demonstrate that filters and slicers work together, as described.

1. Open the DataModelWithMetricsForVisualizations.pbix file and delete any existing visuals. You will need an entire uncluttered report for this example.
2. Expand the DateDimension table and drag the FullYear field into the Page level filters box in the Filters well.
3. Click “Basic filtering” for this filter.
4. Check the box for the year 2014. This way, you have filtered the page to display only data for 2014.

5. Add a table based on the following fields:
 - a. Color (from the Colors table)
 - b. CostPlusSpares (from the InvoiceLines table)
6. Add a bar chart based on the following fields:
 - a. Make (from the Stock table)
 - b. PartsCost (from the InvoiceLines table)
7. Click the pop-up menu for the bar chart (the ellipses that appear at the top right of the visual when you place the mouse pointer over the chart).
8. Click the downward-facing chevron to the right of Sort by and then select PartsCost. This will sort the bar chart in descending order.
9. Adjust the layout of the two visualizations so that it looks something like Figure 14-6.

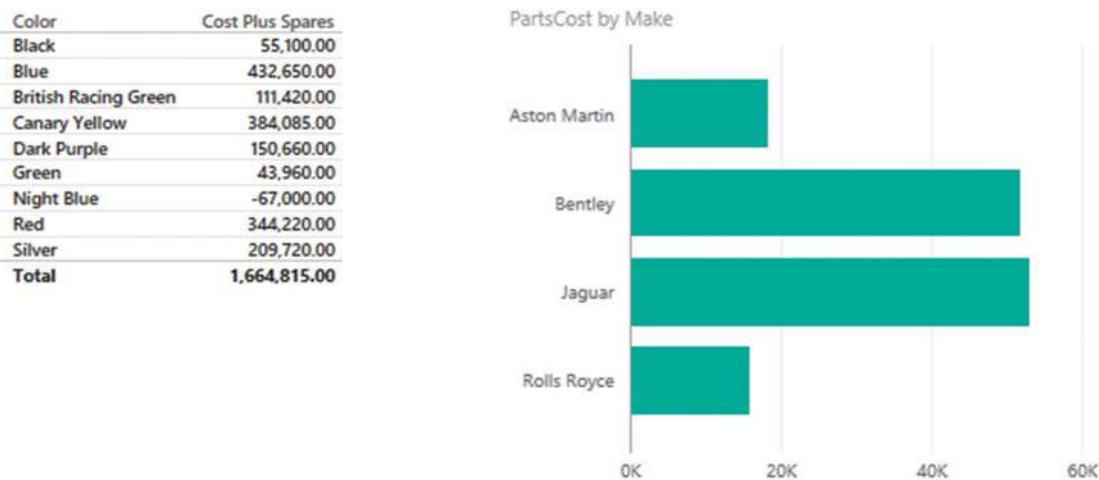


Figure 14-6. Preparing a chart for use as a slicer

Now let's see how to use a chart as a slicer.

10. Click any column in the chart of parts costs by make. I will choose Jaguar in this example.

The Power BI Desktop report will look something like Figure 14-7.

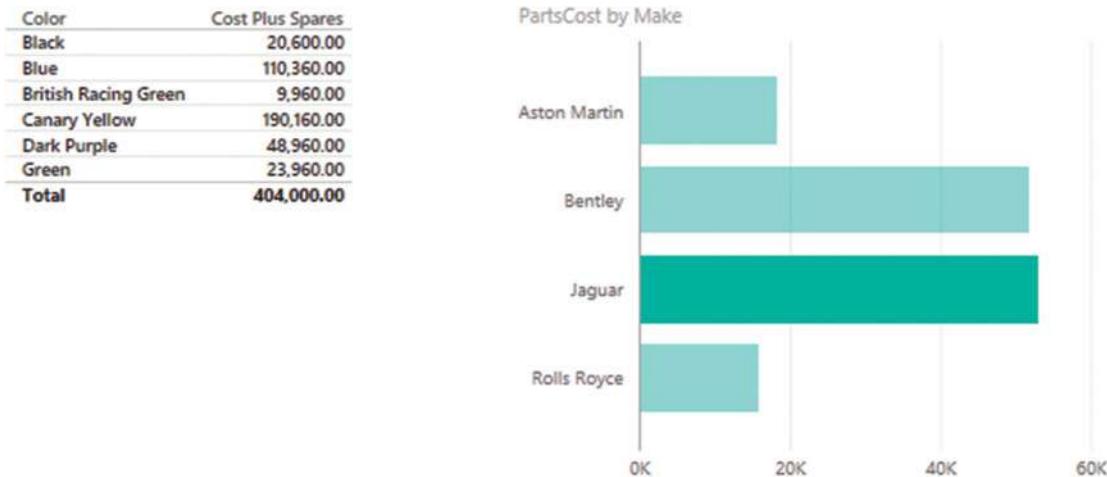


Figure 14-7. Slicing data using a chart

You will see that not only is the make that you selected highlighted in the chart (the bars for other makes are dimmed), but that the figures in the table also change. They, too, only display the cost plus spares (for each color) for the selected make.

To slice on another make, merely click the corresponding column in the column chart. To cancel the effect of the chart acting as a slicer, all you have to do is click for a second time on the highlighted column.

Any bar chart, pie chart, or column chart can act like a slicer in this way, as can funnel charts, treemaps, scatter charts, bubble charts and maps. The core factor is that for a simple slice effect, you need to use a chart that contains only one axis; that is, there will only be a single axis in the source data and no color or legend. What happens when you use more evolved charts to slice, filter, and highlight data is explained next.

Tip It is perfectly possible to select multiple bars in a chart to highlight data in the same way that you can select multiple elements in a slicer. Simply Ctrl-click to select multiple items.

Highlighting Chart Data

So far, we have seen how a chart can become a slicer for all the visualizations in a dashboard page. However, you can also use another aspect of Power BI Desktop interactivity to make data series in charts stand out from the crowd when you are presenting your findings. This particular aspect of data presentation is called *highlighting*.

Once again, highlighting is probably best appreciated with a practical example. First, we will create a stacked bar chart of costs by CountryName; and then we will use it to highlight the various costs inside the chart.

1. In a blank Power BI Desktop report (so you do not get distracted), create a clustered column chart based on the following fields:
 - a. CountryName (from the Countries table)
 - b. PartsCost (from the Stock table)
 - c. LaborCost (from the Stock table)

- Click PartsCosts in the *legend*. All the sales costs will be highlighted (that is, remain the original color) in the column for each country, whereas the other cost will be grayed out.

After highlighting has been applied, the chart will look like Figure 14-8.

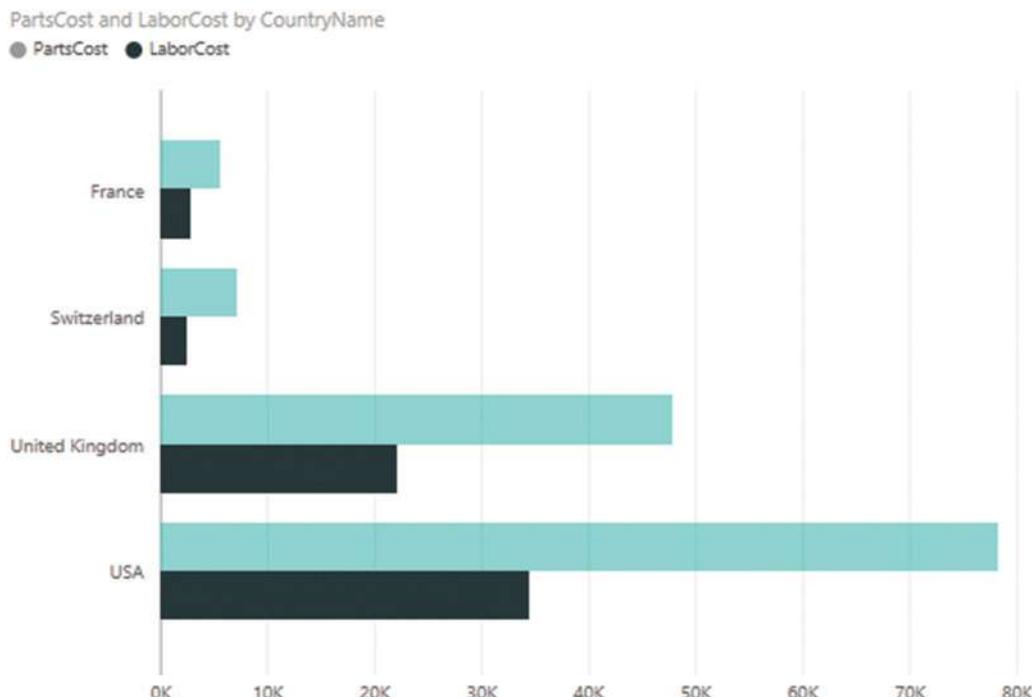


Figure 14-8. Highlighting data inside a chart

To remove the highlighting, all you have to do is click a second time on the same element in the legend. Or, if you prefer, you can click another legend element to highlight this aspect of the visualization instead. Yet another way to remove highlighting is to click inside the chart, but not on any data element.

Highlighting data in this way should suit any type of bar or column chart as well as line charts. It can also be useful in pie charts where you have added data to both the Axis and Legend boxes, which, after all, means you have multiple elements in the chart just as you can have with bar, column and line charts. You might find it less useful with scatter charts.

Cross-Chart Highlighting

Cross-chart highlighting adds an interesting extra aspect to chart highlighting and filtering. If you use one chart as a filter, the other chart is updated to reflect the effect of selecting this new filter not only by excluding any elements (slices, bars, or columns) that are filtered out, but also by showing the proportion of data excluded by the filter.

As an example of this, create a pie chart of net sales by color and a column chart of sales costs by vehicle type. We will then cross-filter the two charts and see the results. The steps to follow are

1. Create a pie chart using the following fields:
 - a. Color (from the Colors table)
 - b. CostPrice (from the Stock table)
2. Create a (clustered) column chart using the following fields:
 - a. Vehicle (from the Stock table)
 - b. DirectCosts (from the Stock table)

For charts that are this simple, Power BI Desktop automatically attributes the fields to the correct boxes in the Fields list once the source tables are converted into charts. The result is shown in Figure 14-9.

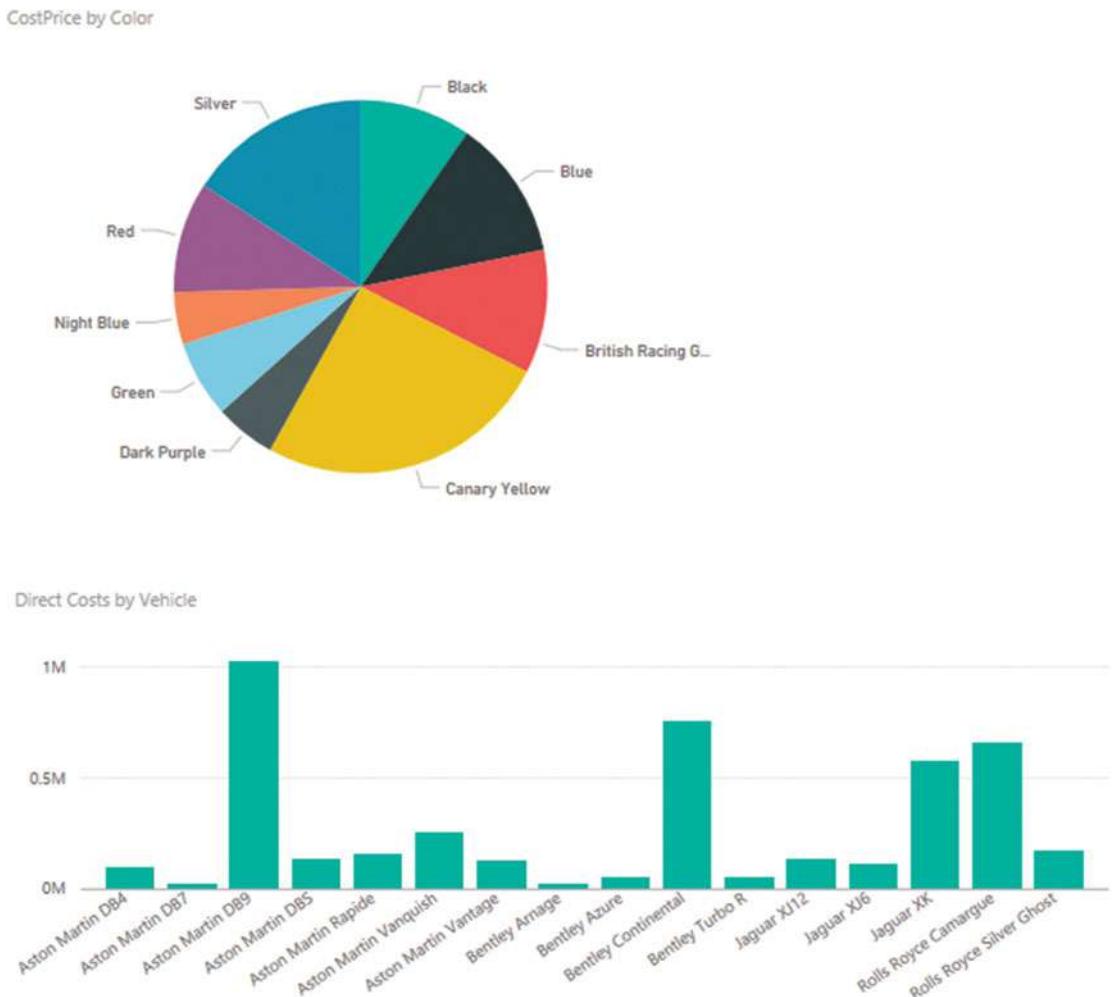
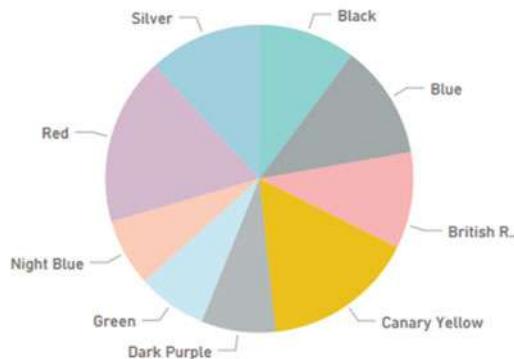


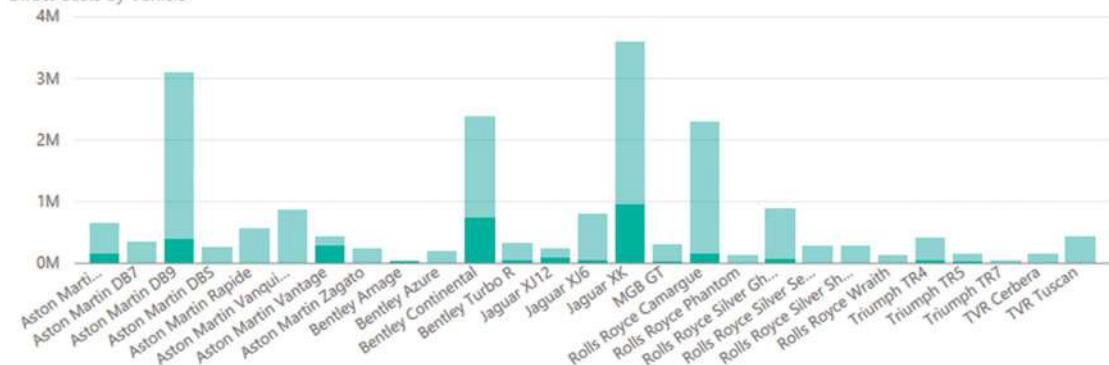
Figure 14-9. Preparing charts for cross-chart highlighting

- Now click the largest slice in the pie chart (Canary Yellow). You should see the result given in Figure 14-10. Not only have all the other segments of the pie chart been dimmed, but the bars in the bar chart have been highlighted to show the proportion of the selected color of the total sales cost per vehicle cost.

CostPrice by Color

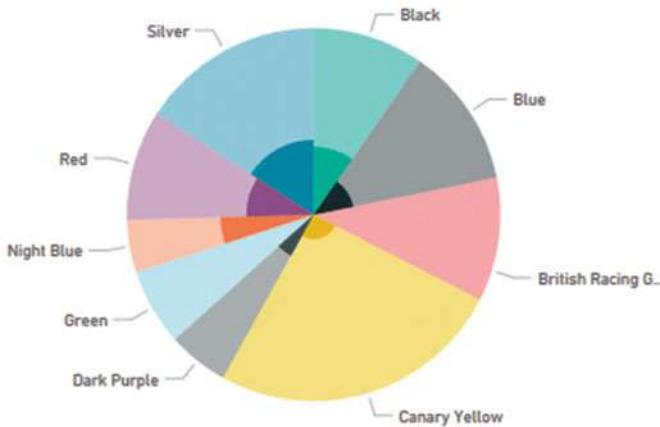


Direct Costs by Vehicle

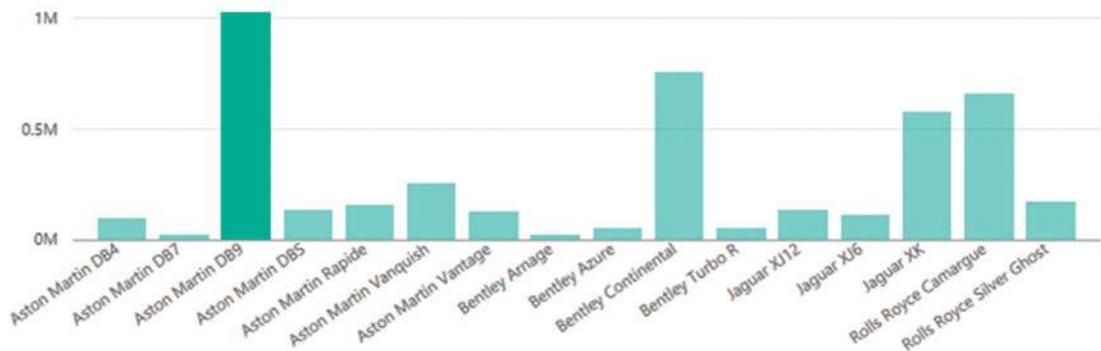
**Figure 14-10.** Cross-chart highlighting

- Now click the bar in the bar chart representing Aston Martin DB9. You are now using the bar chart as a slicer. As you can see in Figure 14-11, the pie chart displays the proportion of Aston Martin DB9 sales for each color.

CostPrice by Color



Direct Costs by Vehicle

**Figure 14-11.** Cross-chart highlighting applied to a pie chart

Note When you use a filter, you do not highlight a chart but actually filter the data that feeds into it; consequently, you remove elements from the chart. Highlighting leaves elements in a chart but accentuates certain aspects of the data relative to others.

Highlighting Data in Bubble Charts

Often when developing a visualization whose main objective, after all, is to help you to see through the fog of data into the sunlit highlands of comprehension, profit, or indeed, whatever is the focus of your analysis, you may feel that you cannot see the forest for the trees. This is where Power BI Desktop's ability to highlight data in a chart visualization can be so effective.

Let's take a visualization that contains a lot of information; in this example, it is a bubble chart of vehicle types. Indeed, in this example, an audience might think that there is so much data that it is difficult to see the bubbles for specific makes of car, and so analyze the uniqueness for sales data by make. Power BI Desktop has a solution to isolate a data series in such a chart. To see this in action and to make the details clearer, you need to do as follows.

1. Create a bubble chart using the following elements:
 - a. *X Axis:* AveragePartsCostRatio (from the Stock table)
 - b. *Y Axis:* SalePrice (from the InvoiceLines table)
 - c. *Size:* GrossMargin (from the InvoiceLines table)
 - d. *Details:* Color (from the Colors table)
 - e. *Legend:* VehicleType (from the Stock table)
2. In the legend for the chart, click a vehicle type. I used saloon in this example. The data for this vehicle type is highlighted in the chart, and the data for all the other vehicle types are dimmed, making one set of information stand out. This is shown in Figure 14-12.

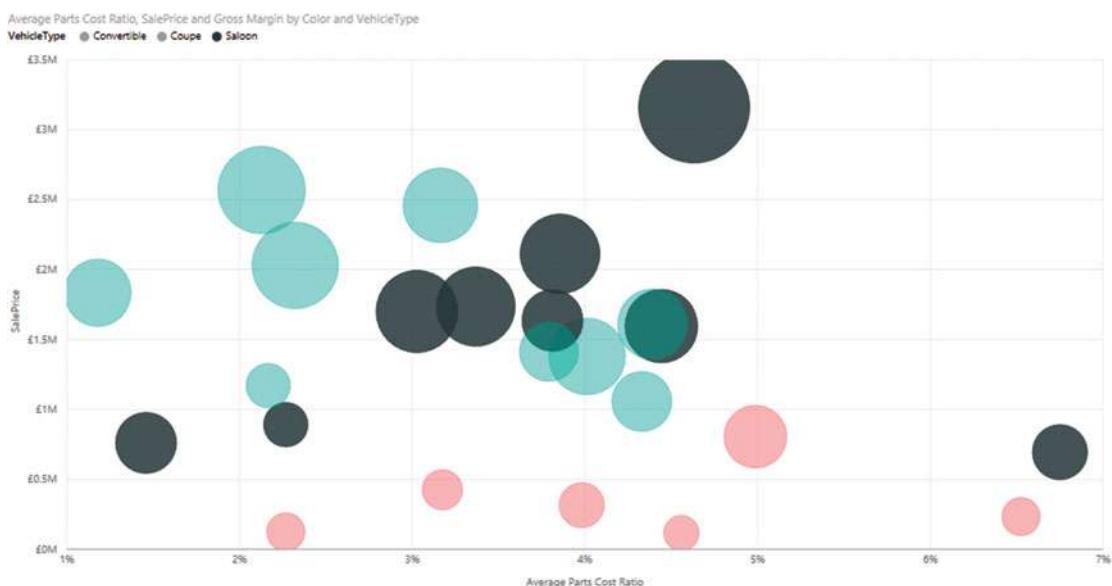


Figure 14-12. Highlighting data in bubble charts

This technique needs a few comments:

- To highlight another dataset, merely click another element in the legend.
- To revert to displaying all the data, click the selected element in the legend again.
- Highlighting data in this way also filters data in the entire page, as described previously.

Tip You can add drill down to charts and still use chart highlighting in exactly the same way as you would use it normally, provided that you have disabled drill down using a chart element, as described in Chapter 12. In this case, you have to drill down (as well as up) using the drill icons. The chart highlights an element at a drill-down level or sublevel, normally as well as apply filtering to the Power BI Desktop report.

Charts As Filters

Now that you have seen how charts can be used as slicers, let's take things one step further and see them used as more complex filters. To show this, I build on the principles shown in the previous example, but add a bubble chart that will filter on two elements at once.

Follow these steps to make the second chart.

1. Build a Power BI Desktop report with the following:
 - a. A total sales by CountryName and Color matrix
 - b. A net sales by Make chart
2. Create a bubble chart using the following data:
 - a. *X Axis:* SalePrice (from the InvoiceLines table)
 - b. *Y Axis:* PartsCost (from the Stock table)
 - c. *Size:* WeeksInStock (from the Stock table)
 - d. *Details:* CountryName (from the Stock table)
 - e. *Legend:* Color (from the Colors table)
3. Resize and tweak the bubble chart so that it is displayed under the existing column chart and table.
4. Click one of the bubbles in the bubble chart. The Power BI Desktop report should look like Figure 14-13.

CountryName	Red	Total
USA	£2,261,740	£2,261,740
Total	£2,261,740	£2,261,740

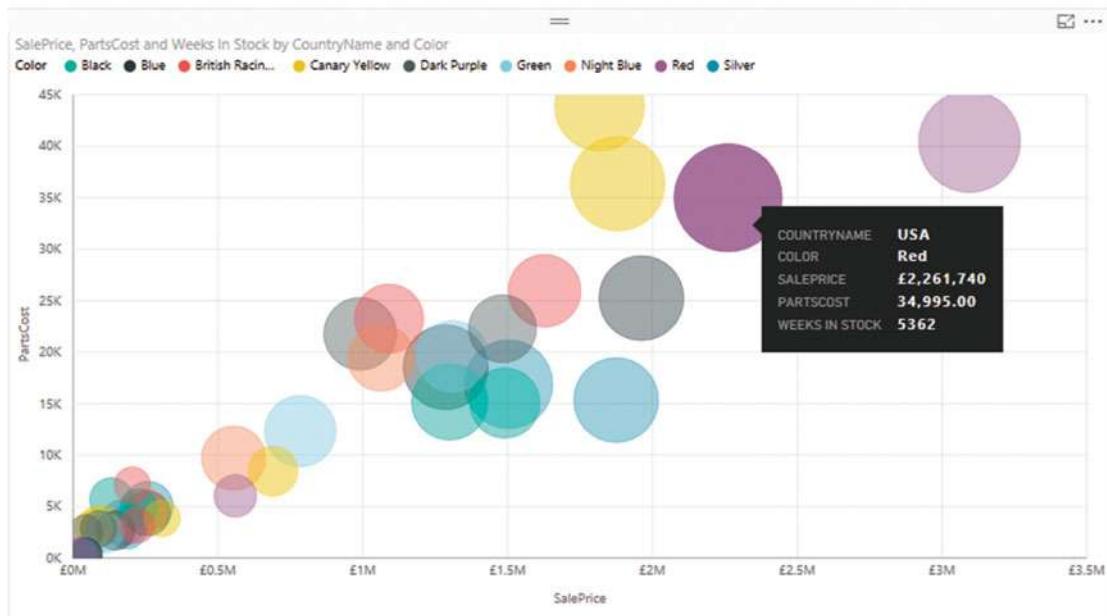


Figure 14-13. Highlighting and filtering using a chart

You can see that the other visualizations are filtered so that both the elements that make up the individual bubble (CountryName and Color) are used as filters (or double-slicers if you prefer to think of them like that). This means that

- The table only shows colors where there are sales for this country and this color.
- The chart highlights bubble data for this country.

As was the case with simple chart slicers, you can cancel the filter effect merely by clicking for a second time on the selected bubble. Or you can switch filters by clicking another bubble in the bubble chart. You will also see the chart itself has data highlighted, but this is explained a little further on.

Clearly, you do not have to display the fields on which you are filtering and highlighting in all the visualizations in a report. I chose to do it in this example to make the outcome clearer. In the real world, all other visualizations in a report are filtered on the elements in the Details and Legend boxes of the bubble chart.

Bubble charts are not the only chart type that lets you apply two simultaneous filters, however. All chart types that display multiple fields allow this. However, I am of the opinion that some charts are better suited than others to this particular technique. Specifically, I am not convinced that line charts are always suited to being used as filters for a Power BI Desktop report. Scatter charts may work—visually, that is—if you use cross-filtering, but it is just as likely that they will not. Stacked bar and stacked column charts can be ideal for cross-filtering, as you will see in the following sections.

Column and Bar Charts As Filters

Column charts and bar charts can also be used to filter a Power BI Desktop report on two elements simultaneously. The only limitation is that you can only have one set of numeric data as the values for the chart. If the bar or column chart is a stacked bar, then you can click any of the sections in the stacked bar. In addition, if the chart is a clustered bar or column, you can click any of the columns in a group to slice by the elements represented in that section.

If this limitation is not a problem, then this is how you can use bar or column charts (whether they are clustered, stacked, or 100% stacked) to apply double filters to a report.

1. Create a Power BI Desktop dashboard using the DataModelWithMetricsForVisualizations.pbix file with the following two elements:
 - a. A table based on color, country name, sales price, gross margin, and cost price
 - b. A bar chart of total sales by CountryName
2. Then create a stacked column chart using the following data:
 - a. *Values:* NetMargin (from the InvoiceLines table)
 - b. *Axis:* CountryName (from the Countries table)
 - c. *Legend:* VehicleAgeCategory (from the Stock table)

Once tweaked to clarify the appearance of the chart, the net result should look like Figure 14-14.

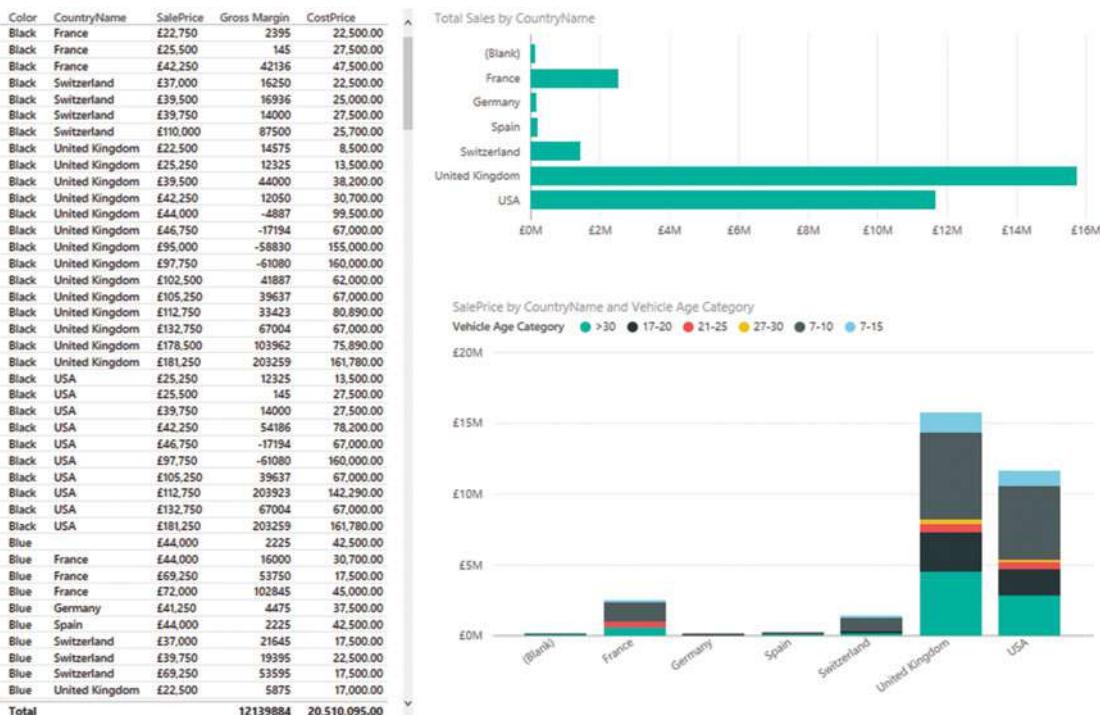


Figure 14-14. A Report ready for chart-based filtering and highlighting

Clicking any segment of a bar filters and highlights other visualizations on the same report for that country and car age range. An example of this is given in Figure 14-15, where the car age range of 17-20 has been selected for the USA bar.

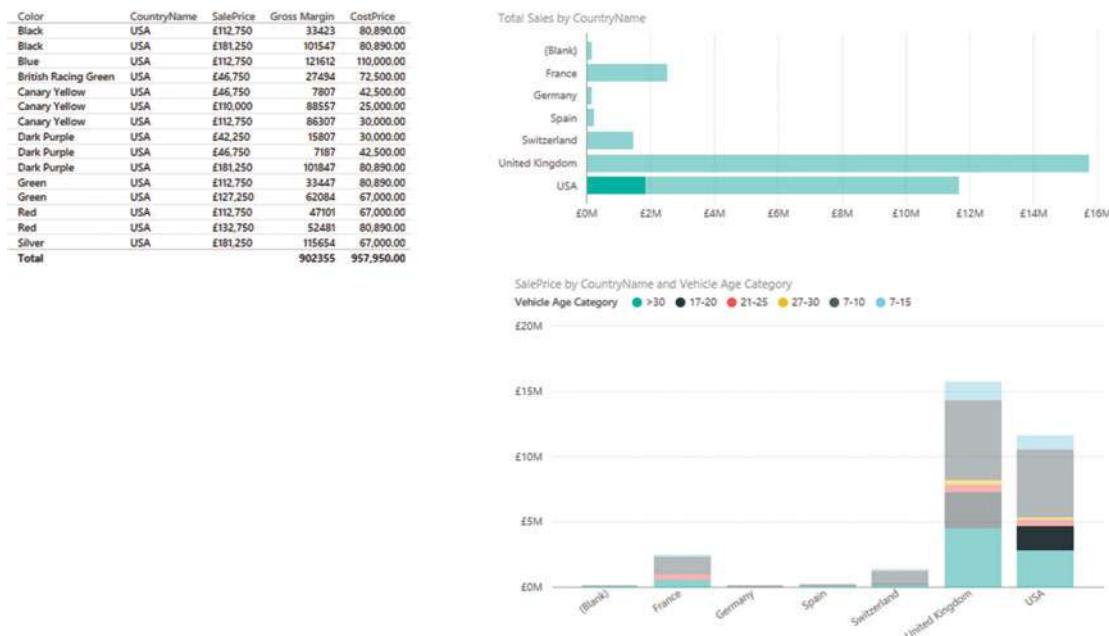


Figure 14-15. Applying filters and highlights

Clicking any car age range in the legend will filter by car age range only. You see this in Figure 14-16.

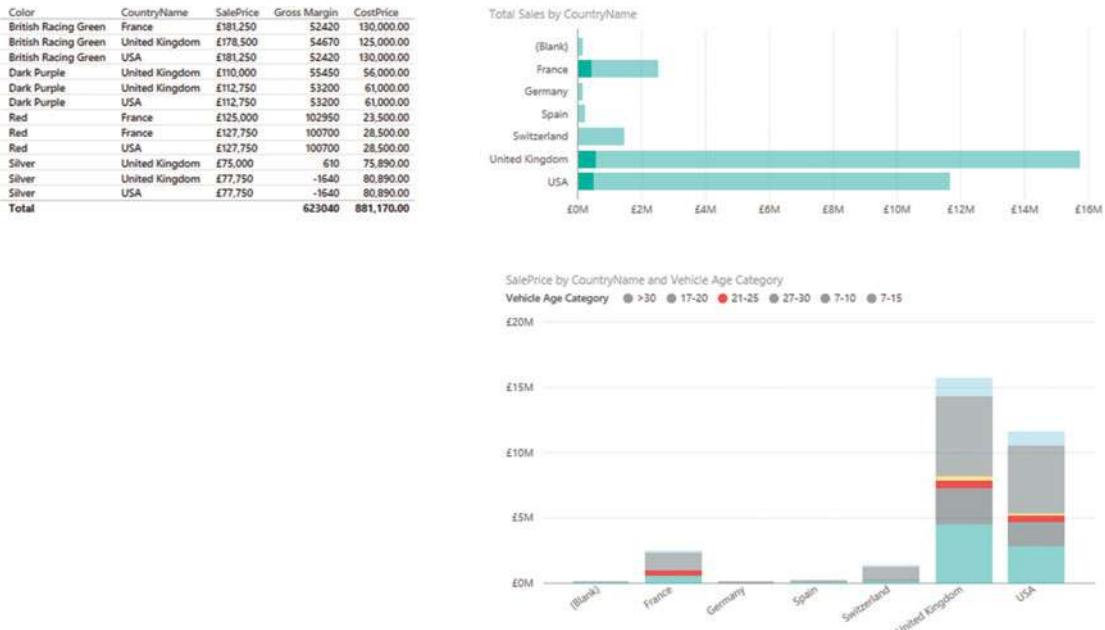


Figure 14-16. Filtering using a legend element

So in fact, you can choose to filter on a single element or multiple elements, depending on whether you use the chart or the legend as the filter source.

Note A line chart will not produce the same effect. If you click a series in a line chart, you are highlighting that series, which is numeric data, and so it cannot be used as a slicer. Similarly, if you click an element in the legend of a column or bar chart, you are selecting data series, and this, too, cannot serve as a slicer (even though it highlights the series in the chart).

Specifying Visual Interactions

In the previous few sections, you saw that the effects of selecting one or more items in a slicer are applied automatically to every visual on a page, as will the effects of using a chart as a slicer. This is indeed true, and is the default behavior of Power BI Desktop unless you specifically configure a visual *not* to react when a chart or slicer on the page has items selected.

In effect, this means that you can attain a tremendous degree of subtlety in your dashboards, because you can define which visuals are to remain interactive-and which must not change when a slicer, chart, or multiple slicers and charts are used.

The following explains how to alter the default setting, and remove interactivity from a visual.

1. In the Home ribbon, click the Edit Interactions button.
2. Click the chart or slicer whose effect you want to modify. Two tiny icons (a funnel and a “no entry” sign) will appear at the top right of every visual on the current page—including other slicers and charts.
3. These icons are shown in Figure 14-17.

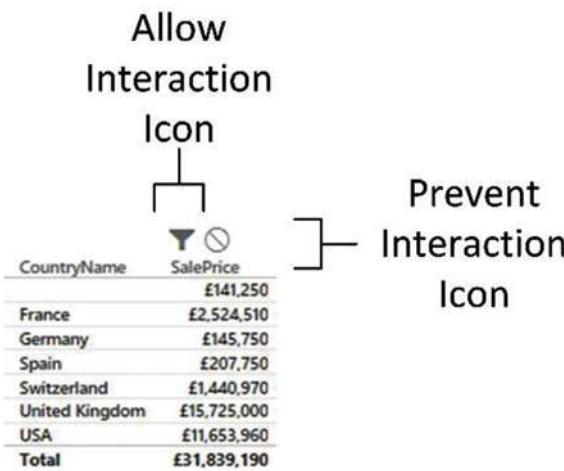


Figure 14-17. The visual interaction icons for visuals

4. Click the Stop Interaction icon (the “no entry” sign) in another visualization. I used the table of sales by country that you created previously. This icon will appear filled in, as shown in Figure 14-18.

CountryName	SalePrice
	£141,250
France	£2,524,510
Germany	£145,750
Spain	£207,750
Switzerland	£1,440,970
United Kingdom	£15,725,000
USA	£11,653,960
Total	£31,839,190

Figure 14-18. The visual interaction icons set to prevent interaction

5. Repeat steps 2 through 5 for all visuals that you want to “disconnect” from the selected visual.
6. In the Home ribbon, click the Edit Interactions button to stop configuration of visual interaction.

You can then iterate through all the charts and slicers on a page to set their dependency on another element.

Note A slicer can also be linked to or dependent on another slicer on the page. Consequently, you can set the interaction for a slicer just as you can for any other visual.

Choosing the Correct Approach to Interactive Data Selection

Now that you have taken a tour of the interactive options that Power BI Desktop offers, it is worth remembering that there is a fundamental difference between slicers and chart filters:

- Slicers and chart filters apply to all visuals on the Power BI Desktop page. Unless you have tweaked the visual interactions
- Highlighting only applies to the selected chart, although it filters data in other tables and highlights the percentage of this element in other charts.

Conclusion

In this chapter, you have seen how to use the interactive potential of Power BI Desktop to enhance the delivery of information to your audience. You saw how to add slicers to a report, and then how to use them to filter out data from the visualizations it contains. Then, you learned how to highlight data in charts. Finally, you saw how to use charts as interactive slicers to isolate specific elements in a presentation.

These techniques are powerful tools that can dramatically enhance the way that you present data to an audience. Used carefully, they will help your dashboards become more powerful and even more memorable. So all that remains is for you to start applying these techniques using your own data. Then you can see how you can impress your audiences using all the interactive possibilities of Power BI Desktop.

CHAPTER 15



Enhancing Dashboards

After spending a little time working with Power BI Desktop, I can assume that you have analyzed your data. In fact, I imagine that you have been able to tease out a few extremely interesting trends and telling facts from your deep dive into the figures—and you have created the tables, charts, maps, and gauges to prove your point. To finish the job, you now want to add the final tweaks to the look and feel of your work so that it will come across to your audience as polished and professional.

Fortunately, Power BI Desktop is on hand to help with all of these final touches, too. It can propel your effort onto a higher level of presentation—without you needing to be a graphic artist—so that your audience is captivated. With a few clicks, you can

- Add free-form text.
- Add images to a report.
- Apply a report background.
- Add basic shapes to enhance your visuals.

This chapter takes you through these various techniques and explains how to use them to add real pizzazz to your analysis. We will use the C:\PowerBiDesktopSamples\ DataModelWithMetricsForVisualizations.pbix file as the source data and as an example of all the dashboards in this chapter. You will also have to download the image files to the C:\PowerBiDesktopSamples\Images folder from the Apress web site, as described in Appendix A.

Adding Text Boxes to Annotate a Report

Let's begin at the start. You have spent quite a while digging into data and have found effective ways of drawing your audience's attention to the valuable information that it contains. However, you need the one, final, cherry on the cake—a title for the report. As a title is nothing other than a text box, this is your introduction to adding, formatting, and modifying text boxes.

A text box is a floating text entity that you can place anywhere on the Power BI Desktop report canvas. They are especially useful for annotating specific parts of a dashboard.

Adding a Text Box

Adding a text box is so easy that it takes longer to describe than to do, but nonetheless, this is how you do it:

1. In the Home ribbon, click the Text Box button. An empty text box will appear on the dashboard canvas.
2. Type in the text that you want to add; it will be a title. I entered **Sales for 2015**.

3. Place the mouse pointer over either the corner or lateral central indicators of the title box and drag the mouse to resize the title box.
4. Click the text box header bar (the gray area at the top of the text box) and drag the text box to the top center of the page.
5. Click outside the title anywhere in the blank report canvas.

Figure 15-1 shows you a text box a report. Moreover, should you want to modify a text, it is as easy as clicking inside the text box and altering the existing text just as you would in, say, PowerPoint.



Figure 15-1. Adding a text box to a report

Note If a text box is too small to hold the entire text, then scroll bars are added to the text box when you select it.

Moving Text Boxes

Text boxes are a Power BI Desktop visualization like any other, and consequently, they can be moved and resized just as if they were a table or a chart. So all you have to do to move a title is to

1. Hover the mouse pointer over the text box. A container will appear indicating the text box shape.
2. Click the top bar of the text box and drag the text box elsewhere on the dashboard canvas.

Alternatively, you can select the text box and then place the mouse pointer over the edges of the text box (but not on the corner or side handles) and drag the text box to a new position.

Formatting a Text Box

A text box can be formatted specifically so that you can give it the weight and power that you want. Even though this is completely intuitive, for the sake of completeness here is a short example of what you can do.

1. Select the text box that you created previously. The format palette will be displayed under or above the text box.
2. Select the text that you wish to modify.
3. Click the Font pop-up and select the font that you want to apply to the selected text.

4. Click the Font Size pop-up and select the size (in points) that you want to apply.
5. Apply any font attributes that you require (bold, underline, italic) by clicking the relevant icons.
6. Align the text by clicking one of the three alignment icons. This will apply to all the text in the box.
7. The formatted text box will now look like it does in Figure 15-2.



Figure 15-2. A formatted text box

Tip Remember that you must highlight the text to format it. If you select the text box itself, then you can only alter the alignment of the text in the text box.

The formatting options for text boxes are explained in Table 15-1.

Table 15-1. Formatting Options for Text Boxes

Element	Icon	Description
Font	Font Segoe UI Light <input checked="" type="button"/>	Lets you choose a font from those installed on the computer
Font Size	14 <input checked="" type="button"/>	Lets you choose a font size
Bold	B	Makes the text appear in boldface
Italic	<i>I</i>	Makes the text appear in italics
Underline	<u>U</u>	Underlines the text
Left		Places the text on the left of the text box
Center		Centers the text in the text box
Right		Places the text on the right of the text box
Hyperlink		Adds a hyperlink for the selected text

Adding a Hyperlink

Power BI dashboards certainly do not exist in isolation. It follows that you can use them as a starting point to link to other documents or web pages. Remember, however, that a hyperlink will *only* work once a Power BI Desktop file has been deployed to the Power BI Service in the cloud.

To add a hyperlink

1. Select the text box that you created previously.
2. Select the text that will be the hyperlink.
3. Click the Hyperlink icon. The Format palette will expand to display the hyperlink box.
4. Enter or paste in a URL.
5. Click the Done button in the Format palette.

Removing a Hyperlink

To remove a hyperlink that you have already added

1. Select the text box containing a hyperlink that you created previously.
2. Click inside the hyperlink. The Format palette will expand.
3. Click the Remove button.

If you prefer, you can always simply remove the text that is the link.

Deleting Text Boxes

If you want to delete a text box, then be sure to

1. Select the text box.
2. Click the pop-up menu for the text box (the ellipses at the top right of the text box).
3. Select Remove. The text box will be deleted.

An alternative way to delete a text box is to select the text box and press the Delete key.

Note Merely selecting and deleting the text inside the text box will not remove the text box itself; so to be sure that you do not leave any unnecessary clutter in a report, delete any unwanted and empty text boxes.

Modifying the Page Background Color

Power BI Desktop does not condemn you to presenting every report with a white background. To avoid monotony you can add a different color background to each page in a report individually with a couple of clicks.

To apply a background to a report, all you have to do is

1. Click inside the dashboard canvas but not on any existing visuals.
2. In the Visualizations pane, click the formatting icon.
3. Expand the Page Background tab. The available options will look like they do in Figure 15-3.

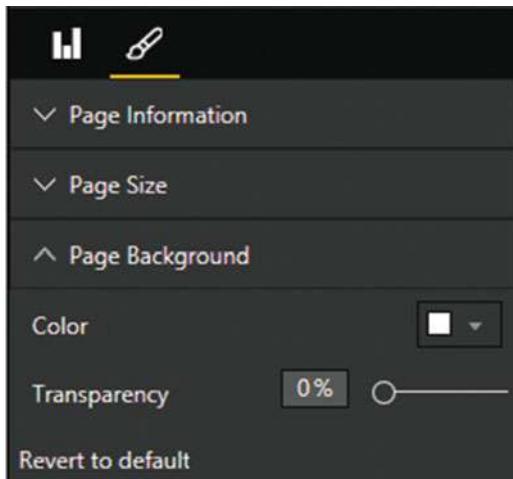


Figure 15-3. Page background options

4. Select a color from the palette of available colors.
5. Slide the Transparency button left or right to select a level of intensity for the chosen color.

Note You have to format every page in a report individually.

Images

We all know what a picture is worth. Well, so does Power BI Desktop. Consequently, you can add pictures, or images, as they are generically known, to a Power BI Desktop report to replace words and enhance your presentation. The images that you insert into a Power BI Desktop report can come from the web or from a file on a disk—either local or on an available network share. Once an image has been inserted, it is *not* linked to the source file. So if the source image changes, you will have to reinsert it to keep it up to date. So, although you can have images from databases appear in tables via PowerPivot, you cannot place these same images outside a table in Power BI Desktop.

The following are some of the uses for images in Power BI Desktop:

- As a background image for a report.
- Images in tables instead of text. An example could be to use product images.

- Images in slicers. These could be flags of countries, for instance.
- Independent images—a logo, for instance, or a complement to draw the viewer's attention to a specific point.
- Images as a chart background.

Once we have looked at the types of image formats available, we will see how images can be used in all these contexts.

Image Sources

There are multitudes of image formats. Power BI Desktop accepts all of the following industry-standard image types.

- *JPEG*: This is a venerable standard image file format.
- *PNG*: This is a standard file format for Internet images.
- *BMP*: This is a standard image type produced by MS Paint, for instance.
- *GIF*: This is a venerable image format frequently used in web pages.
- *TIFF*: This is a standard format for scanned images.

All of these formats (and their various descendant formats that Power BI Desktop can handle) can deliver reasonable quality images that should certainly suffice for Power BI Desktop reports.

If you attempt to insert an image that is not in a format that Power BI Desktop can handle, you will get the alert shown in Figure 15-4.

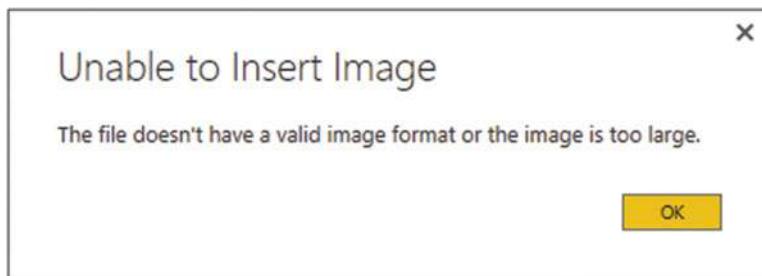


Figure 15-4. Invalid image format alert dialog

Note When you attempt to insert an image from a file, the Open dialog filters the files so that only files with one of the acceptable extensions are visible. You can force the dialog to display other file formats, but Power BI Desktop may not be able to load them.

Adding an Image

You may still want to add completely free-form floating images to a report. However, before getting carried away with all that can be done with images, remember that Power BI Desktop is not designed as a high-end presentation package. If anything, it is there to help you analyze and present information quickly and cleanly. Inevitably, you will find that there are things that you cannot do in Power BI Desktop that you are used to doing in, say, PowerPoint. Consequently, there are many presentation tricks and techniques that you may be tempted to achieve in Power BI Desktop using images to get similar results. Indeed, you can achieve many things in a Power BI Desktop report by adding images. Yet the question that you must ask yourself is “Am I adding value to my report?” I am a firm believer that less is more in a good presentation. Consequently, although I will show you a few tricks using images, many of them go against the grain of fast and efficient Power BI Desktop report creation and can involve considerable adjustment whenever the data in a visualization changes. So I advise you not to go overboard using images to enhance your presentations unless it is really necessary.

Despite these caveats, let's add a floating, independent image to a Power BI Desktop report. In this example it is a company logo—that of Brilliant British Cars, the company whose metrics we are analyzing throughout the course of this book.

Adding an image is really simple. All you have to do is

1. In the Power BI Desktop Home ribbon, click the Image button. A classic Windows dialog will appear; it lets you choose the source image, as shown in Figure 15-5.

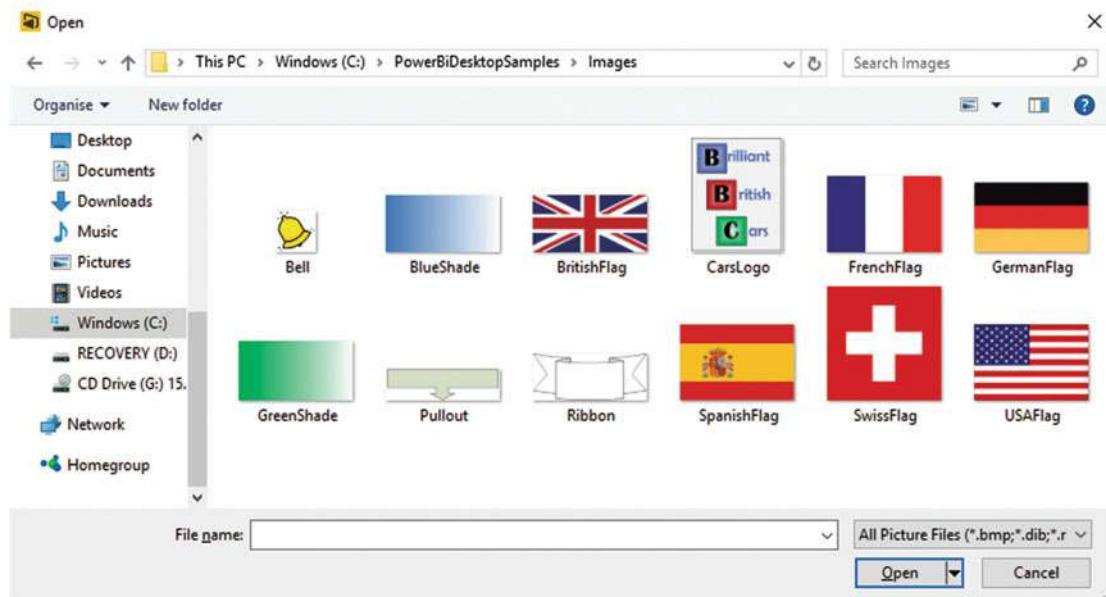


Figure 15-5. Navigating to an image file

2. Navigate to the directory containing the image that you wish to insert. There are several sample images in the C:\PowerBiDesktopSamples\Images folder, which you can install as described in Appendix A.

3. Click the image file. I use the example image CarsLogo.png in this example.
4. Click Open. The selected image will be loaded into the page.
5. Drag the image to the top left of the page. The dashboard will look like it does in Figure 15-6.



Figure 15-6. An image added to a dashboard

Removing an Image

To remove an image, all you have to do is

1. Select the image.
2. Click the pop-up menu for the image (the ellipses at the top right of the text box).
3. Select Remove. The image will be deleted.

An alternative way to delete an image is to select the image and press the Delete key. The selected image will disappear from the report.

Resizing Images

An image is just like any other visual in a Power BI Desktop dashboard in that any of these elements can be moved and resized in exactly the same way.

1. Select the image that you want to resize.
2. Place the mouse pointer over either the corner or lateral central indicators of the image and drag the mouse to resize the image.

Formatting Images

Once you have added an image, it can be tweaked to some extent in Power BI Desktop. The following are the parameters that you can modify:

- Image scaling
- Image title
- Image background
- Aspect ratio
- Exact position and size

All of these modifications except for the first are common to most of the visualizations in Power BI Desktop. Indeed, you have come across some of them several times already. Consequently, I will not waste your time repeating things that you already know, or can find elsewhere in this book.

Image scaling is completely new, however. Here is how you can alter the way that an image is altered if you resize it.

1. Select the image whose scaling you want to modify.
2. In the Visualizations pane (which has switched automatically to show only formatting options), expand the Scaling tab.
3. In the Scaling pop-up, select Fill.

The image will adjust to take up all the available space in the image placeholder.

The three available image scaling options are explained in Table 15-2.

Table 15-2. Image Scaling Options

Element	Description
Normal	The image maintains its height-to-width ratio (whatever the Lock Aspect setting) and resizes to fill the image placeholder as well as it can.
Fill	The image fills the image placeholder but distort the image.
Fit	The image fills the image placeholder but cut off parts of the image rather than distort the image.

Background Images

One major, and frequently very striking, use of images is as a background to a Report—and possibly even to a whole series of reports. So, let's take a look at how to use images for report backgrounds.

Adding a Background Image

Before anything else, you need to add a background image. This is, once again, extremely simple:

1. In the Power BI Desktop Home ribbon, click the Image button. A classic Windows dialog will appear; it lets you choose the source image.
2. Navigate to the directory containing the image that you wish to insert. In this example, it will be C:\PowerBiDesktopSamples\Images.

3. Click the image file. I will use the example image GreenShade.jpg in this example.
4. Click Open. The selected image will be loaded into the page.
5. Resize the image until it covers the entire page.
6. In the Power BI Desktop Home ribbon, click the Arrange pop-up.
7. Select “Send to back”. The image will be placed behind any other elements on the page.
8. Click outside the page (in the gray band) to unselect the image.

Frequently, an image can need a little tweaking until it truly enhances a Power BI Desktop report. To help you control the final display of a background image, Power BI Desktop offers you several ways to resize the image—both proportionally and nonproportionally—in a report; for example, to make an image cover an entire report.

Note It is advisable to add a background image once the rest of the page is as complete as possible. This is because selecting dashboard visuals without accidentally selecting the background image can become a little wearing.

Images in Charts

It is also possible to add an image as a background to certain chart types. Personally, I find this effect a little cheesy, but it is worth knowing that it can be done.

1. Create a bar chart using the following elements:
 - a. Axis: Vehicle
 - b. Value: Average Parts Cost Ratio
2. Filter the visualization as follows:
 - a. CountryName: United Kingdom
 - b. FullYear: 2014
3. With the chart selected, click the paintbrush icon in the Visualizations pane.
4. Expand the Plot Area tab.
5. Click the Add Image button.
6. Select the image C:\PowerBiDesktopSamples\Images\BritishFlag.png.
7. Select Fill from the Image Fit pop-up. The chart should look like the one in Figure 15-7.

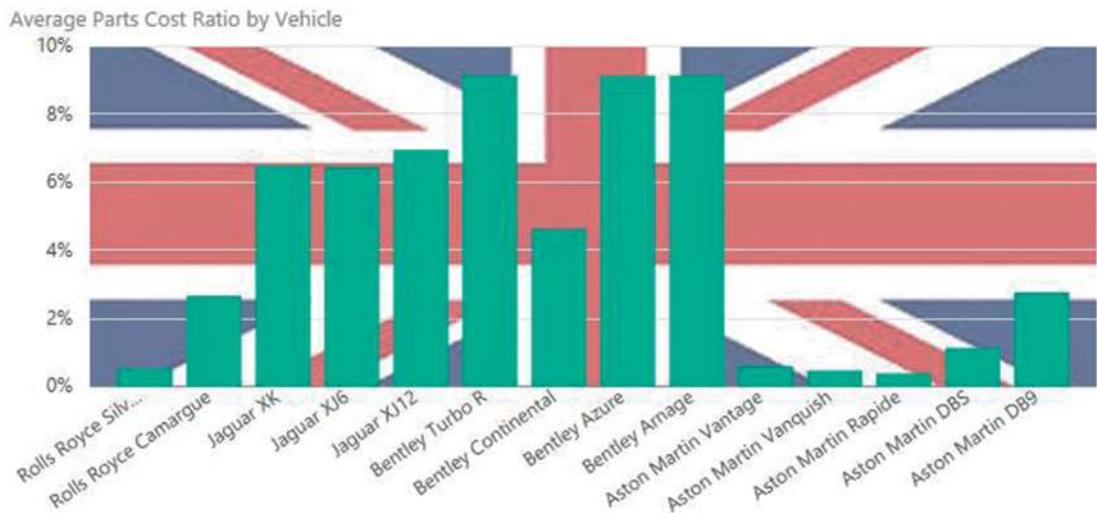


Figure 15-7. Adding an image to a chart background

Some Uses for Independent Images

The limits of what images can do to a report are only those of your imagination, so it is impossible to give a comprehensive list of suggestions. Nonetheless, the following are a few uses that I have found for free-form images:

- *Company logos*, as we have just seen.
- *Images added for a purely decorative effect*. I would hesitate before doing this at all, however, as it can distract from the analysis rather than enhance it. Nonetheless, at times, this may be precisely what you want to do (to turn attention away from some catastrophic sales figures, for instance). So add decoration if you must, but please use sparingly!
- *To enhance the text in a text box* by providing shading that is in clear relief to the underlying image or background.
- *As a background* to a specific column in a table. Be warned, however, that the image cannot be made to move with a column if it is resized.

Adding Shapes

Sometimes your figures may need just a little help to stand out from the crowd. Maybe a small set of visuals (gauges or cards perhaps) are best grouped together. Whatever the need, Power BI Desktop can add a few final touches to your dashboards by adding one or more shapes to a page.

The following are built-in shapes that you can choose from:

- Rectangle (which means squares, too)
- Oval (including circles)

- Line
- Triangle
- Arrow

Let's suppose that you want to add a decorative arrow to a dashboard to draw the reader's attention to a specific figure. The following explains how you can do this.

1. In the Home ribbon, click the Shapes button to see a pop-up list of available shapes. You can see this in Figure 15-8.

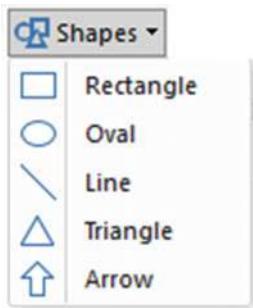


Figure 15-8. The shapes pop-up

2. Select Rectangle. A square will be added to the dashboard canvas.
3. Resize the square in the same way that you would resize a chart or an image. If you use the top, bottom or side handles then the square will become a rectangle.

As the other available shapes can be inserted in exactly the same way, I will not explain them individually, but will let you have some fun by adding different shapes to a page to see how they can enhance a dashboard.

Formatting Shapes

You can tweak the following aspects of shapes:

- Lines
- Fill
- Rotation
- Title
- Background
- Aspect Ratio
- Exact position and size (X and Y coordinates)

As it is only the first three that are new as far as formatting visuals is concerned, let's see them in action.

Lines in Shapes

When we say “lines” in shapes, we are really talking about the exterior boundary of the shape. Power BI Desktop lets you alter its

- Color
- Thickness (or weight as it is known)
- Transparency.

Here is how you can alter the characteristics of a line for any shape:

1. Select the shape to format.
2. In the Visualizations pane (that now displays only the Format Shape options), expand the Line tab.
3. Select a color for the line from the pop-up palette of available colors. The exterior line of the shape will change color.
4. Move the Weight slider to the right to adjust the line thickness. The exterior line of the shape will grow thicker.
5. Move the Transparency slider to the left to adjust the intensity of the color. The Visualizations pane will look like it does in Figure 15-9.



Figure 15-9. Formatting the outside border of a shape

Shape Fill

You can also—perhaps inevitably—modify the color and transparency of a shape. The following explains how.

1. Select the shape to format.
2. In the Visualizations pane (that now displays only the Format Shape options), expand the Fill tab.
3. Select a color for the line from the pop-up palette of available colors. The exterior line of the shape will change color.

4. Move the Transparency slider to the right to adjust the intensity of the color. The Visualizations pane will look like it does in Figure 15-10.



Figure 15-10. Formatting the interior of a shape

Shape Rotation

Some shapes need to be rotated to point in the right direction for the effect that you are trying to produce. The shapes that generally need adjusting in this way are arrows and triangles—though this technique can be applied to any shape.

1. Select the shape to rotate.
2. In the Visualizations pane (that now displays only the Format Shape options), expand the Rotation tab.
3. Drag the rotation slider left and/or right to set the arrow to point in the direction that you want. Alternatively, you can enter the exact rotation value in the rotation box to the left of the slider. The Visualizations pane will look like it does in Figure 15-11.

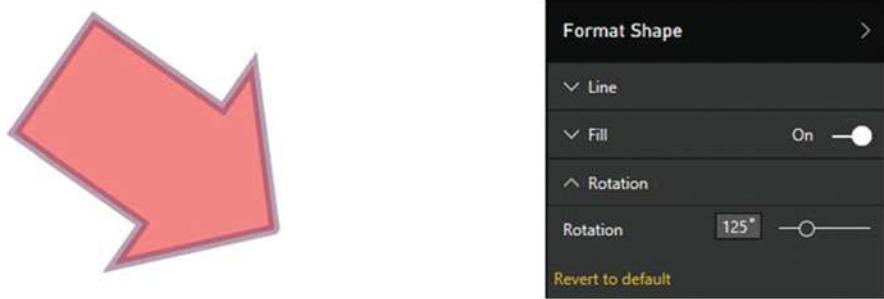


Figure 15-11. Rotating a shape

It is worth noting that you can enter a precise rotation value in the rotation box if you prefer. This is particularly useful when swiveling a shape through multiples of 45 or 90 degrees.

Note When adding an ellipse or a rectangle you will discover that Power BI Desktop begins by creating these as, respectively, circles and squares. To ensure that they stay perfectly formed (if that is what you want), ensure that the Lock Aspect formatting option is set to On before you resize the shape.

Removing Shapes

To remove a shape, all you have to do is

1. Select the shape to remove.
2. Click the pop-up menu for the shape (the ellipses at the top right of the text box).
3. Select Remove. The shape will be deleted.

An alternative way to delete a shape is to select the shape and press the Delete key. The selected shape will disappear from the report.

Standardizing Shapes

If you are adding the final decorative (and hopefully also illustrative touches) to a dashboard you might want to make a series of shapes all look identical. After all, you never want a dashboard to look like a patchwork quilt. A trusted tool from MS Office is available to help you rationalize dashboards in this way—the Format Painter.

1. Select the shape that will serve as the model for other shapes.
2. Click the Format Painter button in the Home ribbon.
3. Click the shape that you want to see formatted identically to the first shape.

Note You cannot double-click the format painter to apply the same format several times to different shapes as you can when formatting in MS Word or Excel.

Organizing Visuals on the Page

A complex dashboard can consist of many elements (though hopefully not so many that you end up confusing your public). This is where some elementary polishing of the final appearance can help. Put simply...

- An audience gives more credence to a slickly organized dashboard.
- Clarity of layout is interpreted as clarity of thought—and so adds to the credibility of the facts and figures that you are presenting.
- Good aesthetics add to, rather than detract from, the points that you are making.

So to finish our tour of the ways that you can finalize and perfect your dashboards, you need to learn how to adjust the way that visuals (whether they are tables, charts, gauges, images, or shapes) relate to one another on the page; this essentially includes the following:

- Placing elements one on top of another (or layering visuals)
- Aligning visuals, vertically and horizontally
- Distributing visuals, vertically and horizontally

Let's take a look at each of these in turn.

Layering Visuals

As a report gets more complex, you will inevitably need to arrange the elements that it contains not only side by side, but also one on top of the other. Power BI Desktop lets you do this simply and efficiently.

As an example of this, let's create a chart with another chart superposed on it.

1. Create a bar chart using the following two fields:
 - a. Make (from the Stock table)
 - b. RatioNetMargin (from the InvoiceLines table)
2. Order the bar chart by RatioNetMargin, in descending order.
3. Create a donut chart using the following two fields:
 - a. CostPlusSpares (from the InvoiceLines table)
 - b. CountryName (from the Countries table)
4. In the pie chart, and add a legend and set it to appear on the right. Also, set the Off category labels. You could also set the background transparency to 0%.
5. Place the pie chart in the top right-hand corner of the bar chart.
6. With the pie chart selected, choose Arrange ► Bring To Front from the Power BI Desktop Home ribbon.

Your composite chart should look like Figure 15-12.

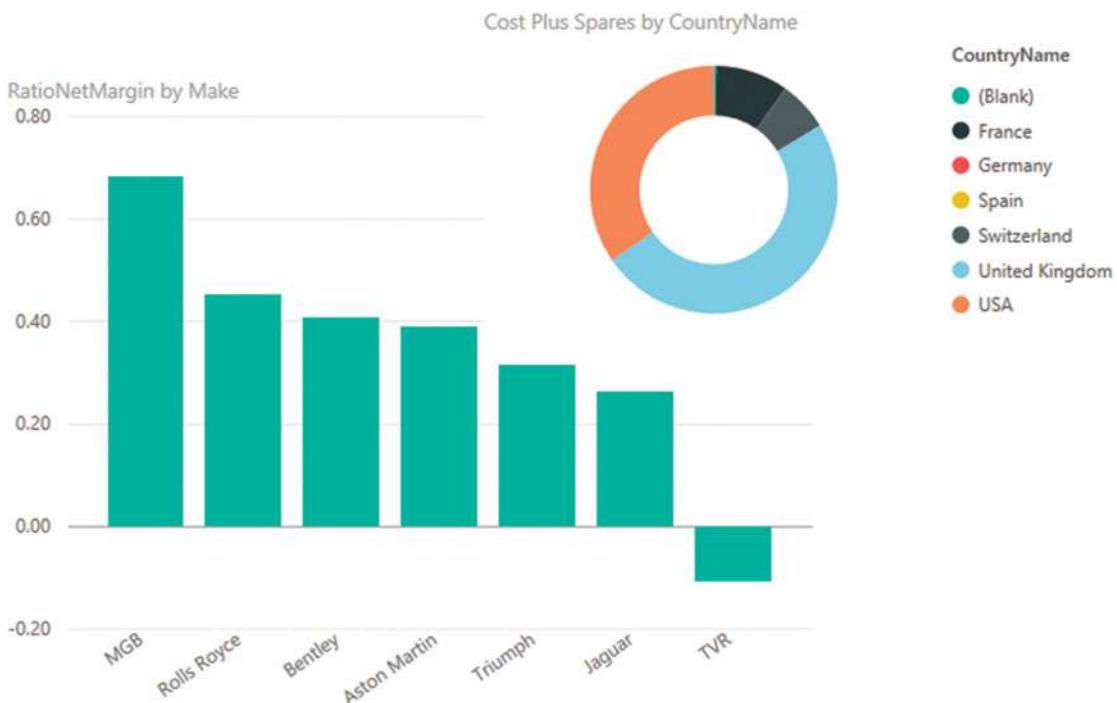


Figure 15-12. Layering charts

This technique is particularly useful when you are adding independent images as was described in the previous section. It is also handy when you are combining elements such as images and text boxes, as you will see in the next section.

Aligning Visuals

A set of neatly aligned elements on a page will always please an audience. What is more it literally only takes a couple of seconds to take a set of existing visuals and to present them harmoniously. The following explains what you do.

1. Select the visuals that you want to align (Ctrl-click each one).
2. In the Home ribbon, click the Alignment button to display the pop-up menu.
It will look like Figure 15-13.

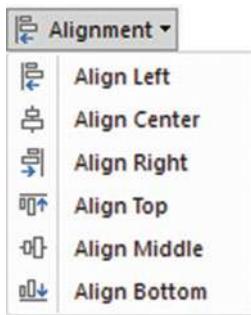


Figure 15-13. Alignment options

3. Select the required option.

The selected elements will be aligned along their tops, bottoms, left or right sides, or centered, depending on the choice that you made. The available alignment options are outlined in Table 15-3.

Table 15-3. Alignment Options

Option	Description
Align Left	Aligns all the selected visuals along their left sides
Align Center	Centers all the selected visuals
Align Right	Aligns all the selected visuals along their right sides
Align Top	Aligns all the selected visuals along their top edges
Align Middle	Aligns all the selected visuals in the middle of the elements
Align Bottom	Aligns all the selected visuals along their bottom edges

Distributing Visuals

One way to add a final professional touch to a dashboard is to make sure that all visuals are neatly distributed horizontally and/or vertically. This will simply guarantee that there is always the same amount of space between each element.

1. Select the visuals that you want to align (Ctrl-click each one).
2. In the Home ribbon, click the Alignment button to display the pop-up menu. It will look like Figure 15-14.

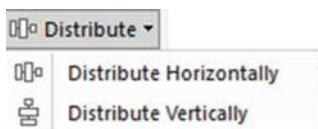


Figure 15-14. Distribution options

3. Select the required option.
4. The selected elements will be aligned along their tops, bottoms, left or right sides, or centered, depending on the choice that you made. The available alignment options are outlined in Table 15-4.

Table 15-4. Alignment Options

Option	Description
Distribute Horizontally	Distributes a selected series of visuals along a horizontal plane
Distribute Vertically	Distributes a selected series of visuals along a vertical axis

Note You may need to distribute a collection of visuals both horizontally and vertically to get the best effect.

Exporting Data from a Visualisation

Finally, it is worth noting that you can export the data that underlies a visualisation. This applies to any visual, from tables to charts. All you have to do is to click on the menu for the visual - the ellipses at the top right of the chart, table or map. Then select Export Data from the popup menu. Simply select a destination directory and file name and the relevant data, with all the filters applied, will be exported. The only current export format is a CSV file.

Conclusion

In this chapter, you saw how to push the envelope when using Power BI Desktop to deliver particularly compelling presentations. You saw how adding images can turbocharge the impression that your analysis gives when you add graphic elements to tables and slicers. And, used sparingly, images, shapes, and free-form text elements can draw your audience's attention to the most salient features of your presentation. So now it is up to you to use these powerful Power BI Desktop features to deliver some really compelling interactive analyses to your audience.

CHAPTER 16



PowerBI.com

You are now approaching the end of your journey into the world of self-service business intelligence with Power BI. Up until now in this book, you have seen how to use Power BI Desktop to prepare and visualize your data. Now, all that remains is to learn how to share your insights with your colleagues. This is where PowerBI.com comes into the frame.

PowerBI.com is a cloud-based online service from Microsoft that lets you...

- Share your Power BI Desktop BI files in the cloud. This will allow your co-workers to view and interact in real time with Power BI Desktop reports in a browser window.
- Use the new Power BI app for mobile devices to view and interact with Power BI Desktop reports on tablet devices and mobile phones.
- Configure any Power BI Desktop files that you have loaded into PowerBI.com so that they connect to on-premises data and so they refresh the data that they contain from on-site data sources at regular intervals. This way you can be sure that your colleagues are always using the most recent available data.
- Create new reports in the cloud using the data from Power BI Desktop files that you have already loaded into PowerBI.com.

However, the truly amazing thing about PowerBI.com is that it is absolutely *free* (for up to 1 GB of data). If this threshold is too low (and to access many other possibilities, especially where group collaboration are concerned) you can upgrade to the PowerBI.com Enterprise service and raise the limit to 10 GB of data (as well as obtaining other advantages) for \$9.99 per month. At least these were the prices when this book went to press.

I have to add that PowerBI.com is a vast product in its own right, and worthy of a book in itself. So I have to warn you that in this chapter, you are only taking a quick look at some of its features; you will not examine everything that this service can do in detail.

Publishing Reports to PowerBI.com

The best way to appreciate PowerBI.com is to see it in action. The following sections explain how to create a PowerBI.com account and upload a report created with Power BI Desktop.

Creating a Power BI Account

Before you can use PowerBI.com, you need to create a Power BI account.

1. In your browser, navigate to www.microsoft.com/PowerBI. You should see a web page like the one shown in Figure 16-1.



Figure 16-1. The PowerBI.com connection page

2. Click Get Started Free. You will see the How to get started page, as shown in Figure 16-2.

Choose how to get started with Power BI



Power BI Desktop for Windows

Analytics tools at your fingertips

Connect and transform data, create advanced calculations, and build stunning reports in minutes.

[Download](#)



Power BI

The easy way to see your important data in one place

With a few clicks, connect to data from applications you use and get started with pre-built dashboards from experts.

[Sign up](#)

Figure 16-2. The How to get started page

3. Click Sign Up. You will see the Sign Up page.
4. Enter your work e-mail address. The page will look like the one displayed in Figure 16-3.



Figure 16-3. The PowerBI.Com startup dialog

5. Click the arrow. PowerBI.com will send you an e-mail confirming the creation of your account. It confirms this with the screen that you can see in Figure 16-4.



Figure 16-4. The Power BI e-mail confirmation screen

6. Go to your e-mail and open the mail from Power BI. It should look like what's shown in Figure 16-5.

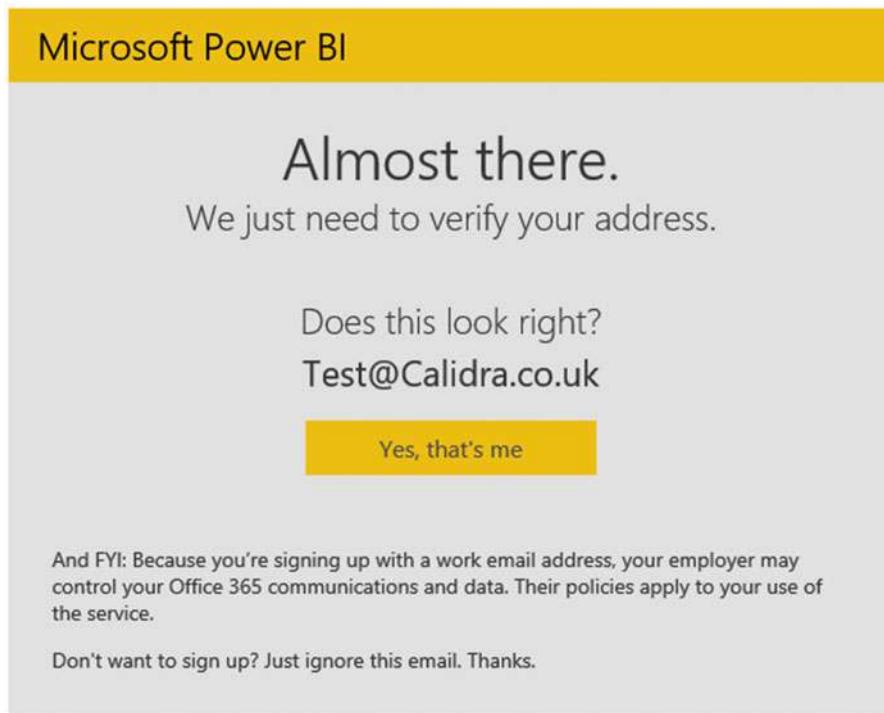


Figure 16-5. The e-mail from PowerBI.com

7. Click “Yes that’s me”. You will be connected to the PowerBI.com service and see a web page like the one in Figure 16-6.

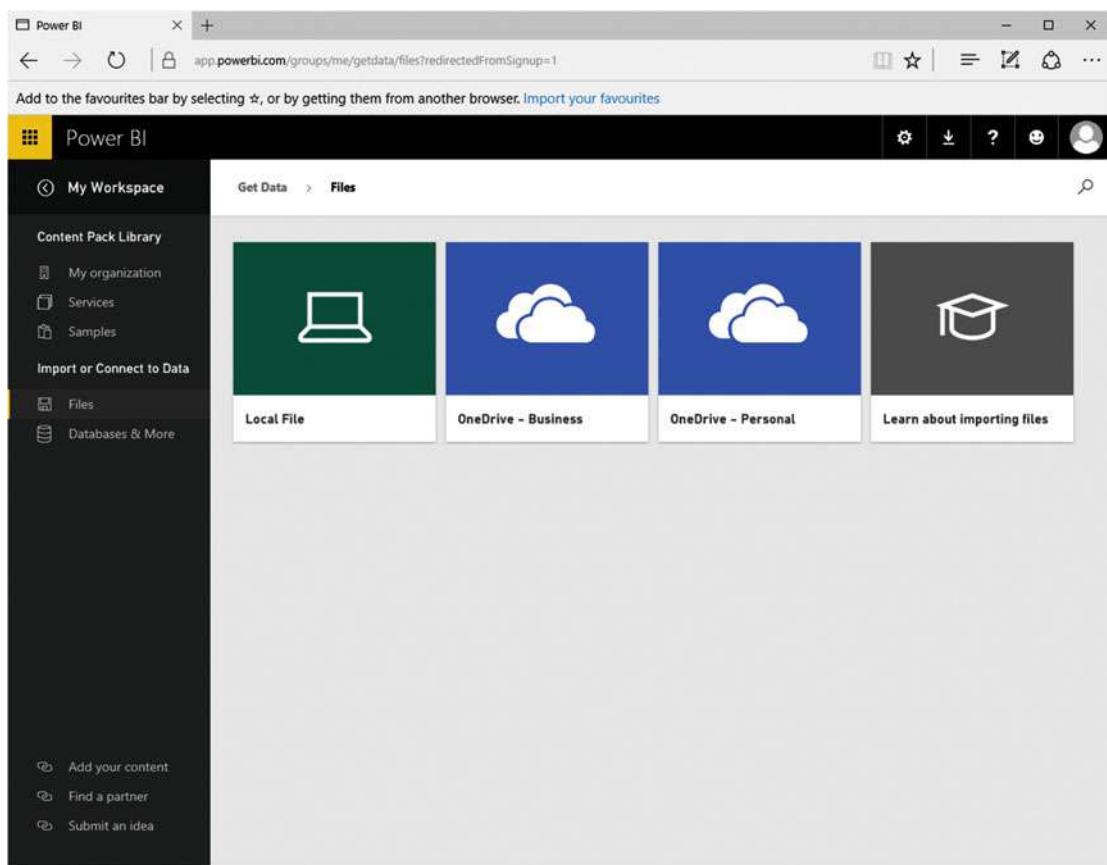


Figure 16-6. The PowerBI.com web page

That is all that you have to do. You now have a PowerBI.com account and can start sharing your insights with colleagues and friends.

Using Power BI Desktop Files in PowerBI.com

It is all very well and good if you actually have a PowerBI.com account, but what is it and what can you use it for? A PowerBI.com is essentially a collaboration environment optimized for self-service business intelligence. Given the whole self-service ethos that underlies Power BI, I think that the easiest way to understand self-service BI is to see the steps in the life cycle of a Power BI Desktop file. This will show the process, from initial load, through sharing and interaction, to deletion. Hopefully, this will give you an idea of why Power BI is so different, so visual, and so tremendously useful in practice.

Logging on to PowerBI.com

1. Enter the PowerBI.com URL in your browser to connect to the PowerBI.com start page that you saw in Figure 16-1.
2. Click Sign In. The Microsoft Sign In page will appear, as shown in Figure 16-7.

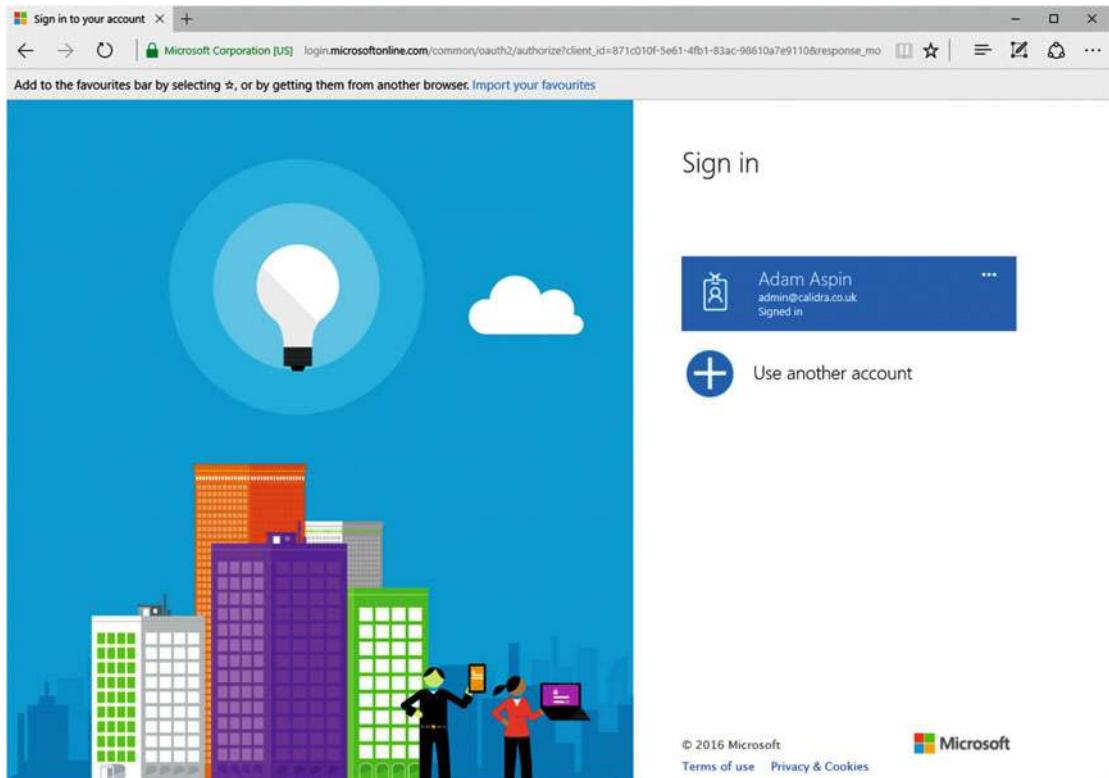


Figure 16-7. The Microsoft sign-in page

3. Click the account that you created previously. You see the PowerBI.com welcome page, as shown in Figure 16-8.

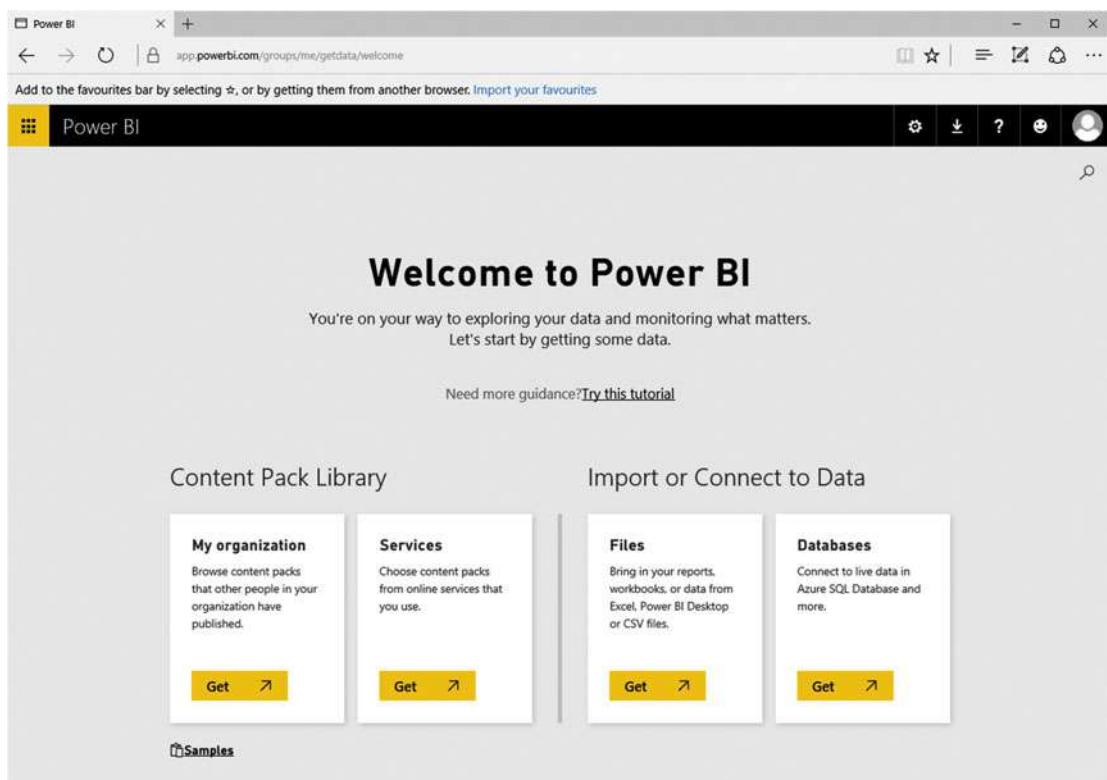


Figure 16-8. The PowerBI.com welcome page

You are now logged into PowerBI.com and ready to upload files created with Power BI Desktop.

Adding a Power BI Desktop File

As an (admittedly very simple) example, let's add a sample Power BI Desktop file directly to PowerBI.com.

1. In the Files section, click Get. You will switch to the Files page that you saw previously in Figure 16-6.
2. Click Local File. The Open dialog will be displayed.
3. Navigate to the Power BI Desktop file that you want to add to PowerBI.com. I used the C:\PowerBiDesktopSamples\FirstDashboard.pibx file from the sample data files that are available for download in this example.
4. Click OK. The file is added to PowerBI.com and appears in the Dashboards, Reports, and Datasets sections of PowerBI.com.
5. Click the FirstDashboard report. You will see the report contents exactly as they were created in Power BI Desktop. You can see this in Figure 16-9.

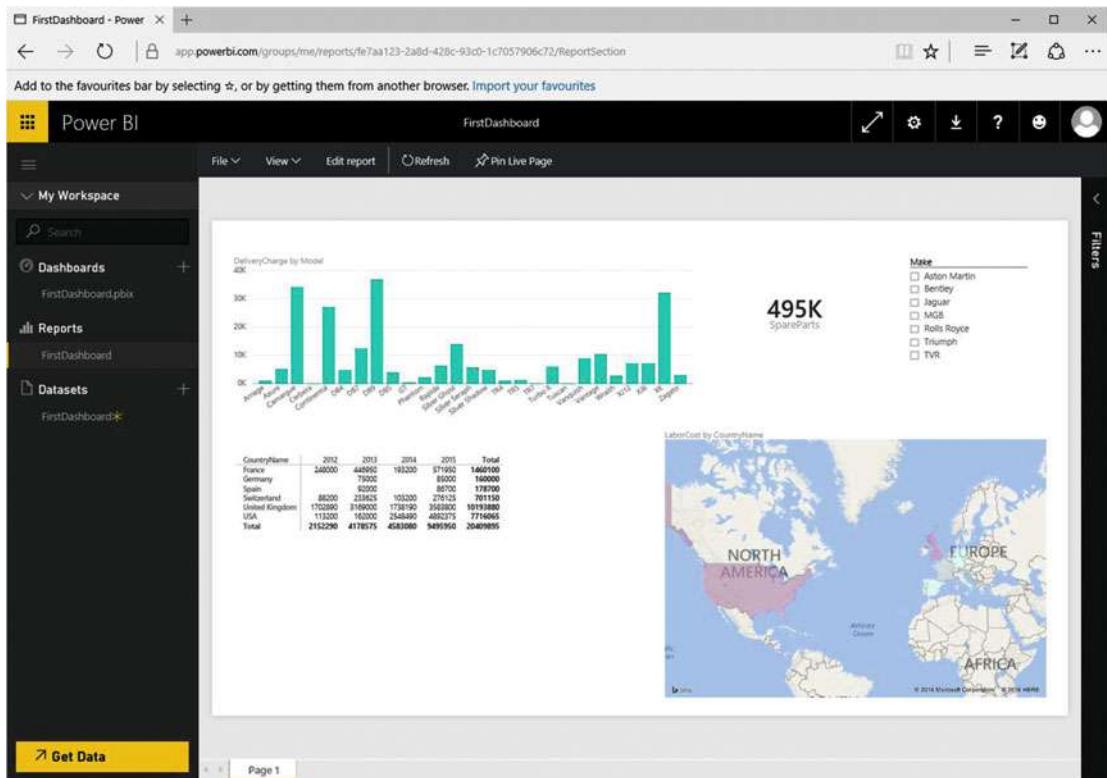


Figure 16-9. PowerBI.com with a Power BI Desktop file added

Interacting with a Report on PowerBI.com

Now that you have uploaded a report to PowerBI.com, it is worth asking what you can do with reports now that you are no longer in Power BI Desktop. The answer is quite simple: you can do nearly everything that you can do in Power BI Desktop. Specifically, you can do the following:

- *Filter data* in the filter pane on the right of the PowerBI.com report, just as you would if using Power BI Desktop. Note that you cannot add further filters, but that any existing report, page or visualization filters are accessible, and you can switch between basic filtering and advanced filtering just as you can in Power BI Desktop.
- *Highlight data* by clicking chart elements, for instance (again just as you can in Power BI Desktop).
- *Switch between pages* using the tabs at the bottom of the report.
- *Zoom in* to a specific visual by clicking the Focus icon in the top right hand corner of a visual (yes, once again just as you can in Power BI Desktop).
- *Sort data* in a visual in the same way that you can in Power BI Desktop.
- *Pan and zoom* in map visuals.
- *Use any custom visuals* that you have downloaded from the PowerBI.com site.

I imagine that by now you understand that the reports that you have loaded from Power BI Desktop are virtually identical to the originals. The only major difference is that you cannot extend an existing report by adding new data or visuals. Indeed, if you have read the chapters explaining how to filter, sort and highlight data, then you will find that PowerBI.com uses almost exactly the same techniques as Power BI Desktop, so you do not need any further explanation to help you on your way to interacting with your Power BI reports in the cloud.

Printing PowerBI.com Reports

Although PowerBI.com is built for sharing information interactively using the web, it will let you print reports directly from a PowerBI.com site. To print a report, you simply do the following:

1. In the Report menu, click File. The File menu will appear.
2. Select Print. The Print options dialog of your PC or tablet will appear from where you can select a printer and print the report.

Creating PowerBI.com Dashboards

In PowerBI.com, the term *dashboard* has a slightly different meaning to the one that we have been using so far in this book. In PowerBI.com, a dashboard is a central point of focus where you can do this:

- Add visuals from any report that you have loaded into PowerBI.com
- Jump straight from a dashboard to the report hosting a visual in a dashboard
- Enhance dashboards with specific annotations.

Let's look at how this works by creating a dashboard and adding a visual to it from the report that you loaded previously.

Creating a New Dashboard

Adding a new dashboard is very simple.

1. In the Navigation pane, click the plus icon to the right of the Dashboards heading. A new empty dashboard field will appear.
2. Enter a name for the new dashboard. I call it Car Sales in this example. The Navigation pane will look something like Figure 16-10.

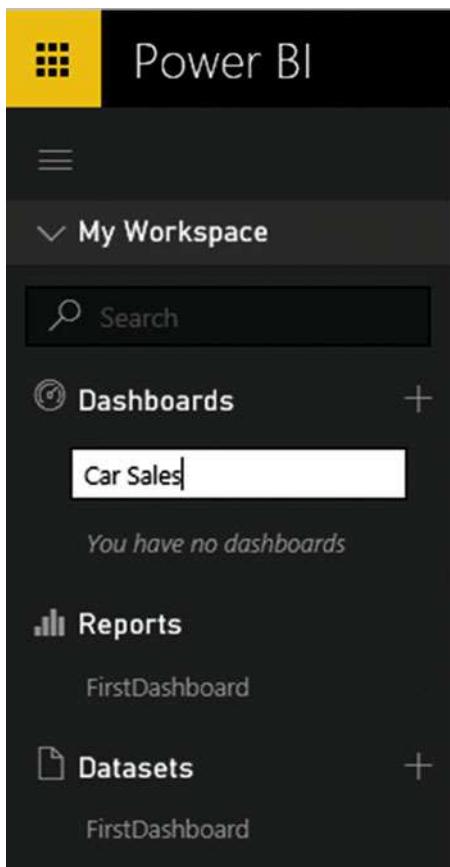


Figure 16-10. Creating a new dashboard

3. Press Enter or click outside the Navigation pane.

You have just created a new, empty dashboard. Now you will learn how to add elements (that PowerBI.com calls *tiles*) from reports that you have previously loaded into PowerBI.com.

Adding Tiles to PowerBI.com Dashboards

Now that you have a dashboard ready and waiting, the following explains how to add some content from an existing report.

1. In the Navigation pane, click the report that you have already loaded. This will be FirstDashboard, assuming that you followed the earlier example. The report will appear, as shown in Figure 16-9.
2. Hover the mouse pointer over the chart (with the title DeliveryCharge by Model) on the top left of the report. Three small icons will appear at the top right of the chart.
3. Click the pin icon. This will display the Pin to Dashboard dialog that you can see in Figure 16-11.

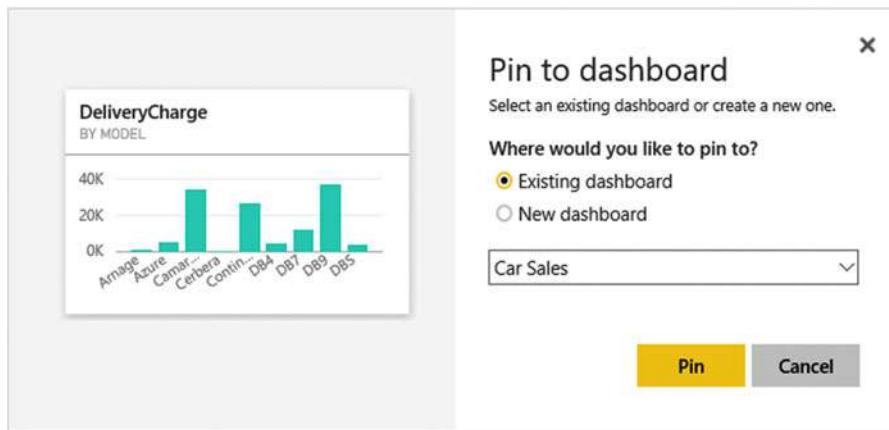


Figure 16-11. The Pin to Dashboard dialog

4. Ensure that the “Existing dashboard” radio button is selected and that the Car Sales dashboard is selected in the pop-up list of available dashboards.
5. Click Pin.

The chart is added to the Car Sales dashboard. If you click the name of this dashboard in the Navigation pane, you will see that the dashboard looks like Figure 16-12.

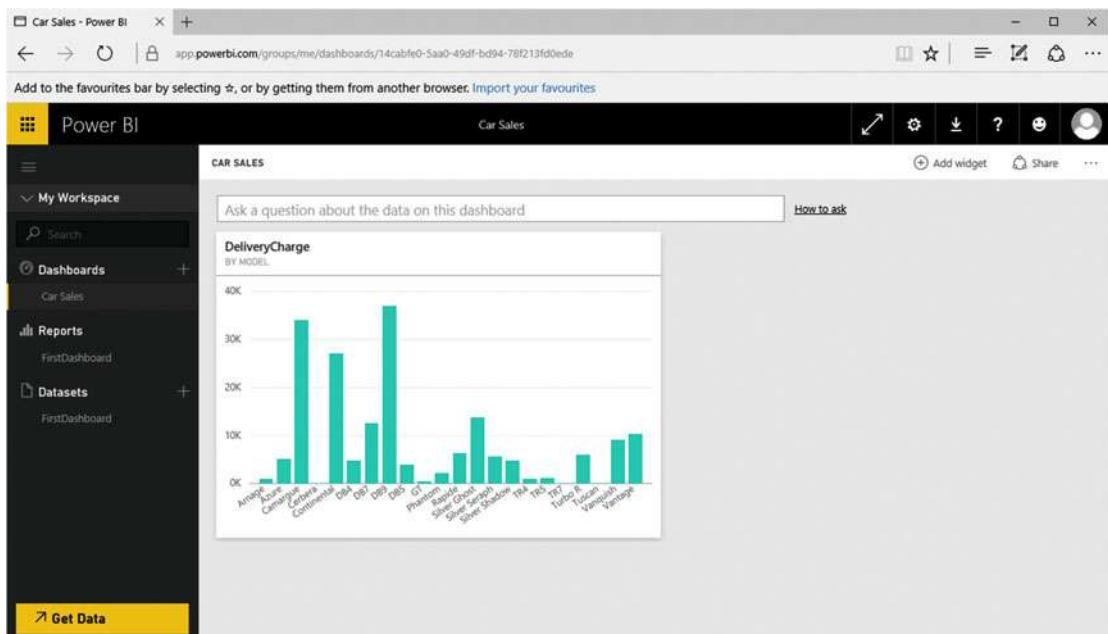


Figure 16-12. A dashboard with a visual added

If you click the chart in the dashboard, you immediately jump to the source report and page for this visual.

Note You can also create a new dashboard when you pin a visual by selecting New Dashboard as the destination in the Pin to Dashboard dialog.

Editing Dashboard Tiles

There are a few interesting things that you can do to widgets once you have added them to dashboards. The available options include

- Deleting tiles
- Modifying tile details
- Exporting the data behind the tile
- Pinning the tile to another dashboard

Let's look at each of these in turn.

Deleting Tiles

The following explains how to delete a tile from a dashboard (while leaving the source visual intact on the original report).

1. Click the tile menu icon (the ellipses at the top right of the tile). The tile options will appear, overlaying the tile data, as you can see in Figure 16-13.

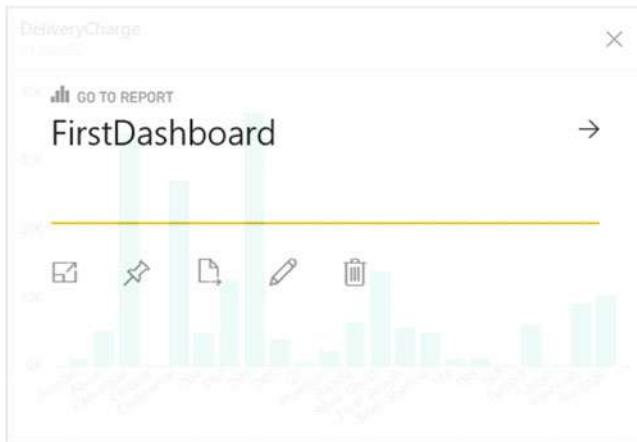


Figure 16-13. The tile menu

2. Click the waste bin icon on the right. The tile will be removed from the dashboard

Note If you delete a tile by mistake, you cannot undo the action. However, since the original visual is still in the source report, you can add it back easily enough.

Modifying Tile Details

There are a few details that you can modify when adding tiles to dashboards.

1. Click the tile menu icon to display the tile menu options.
2. Click the pencil icon. The Tile Details pane will appear on the right of the screen, looking rather like the one in Figure 16-14.

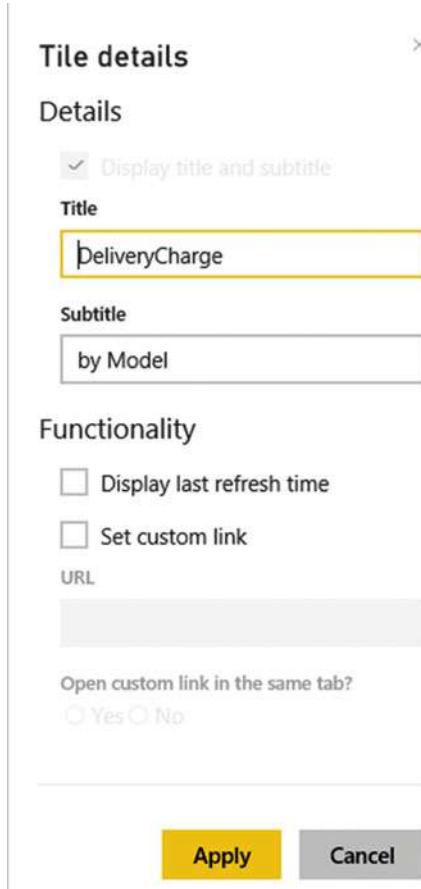


Figure 16-14. The tile details pane

The available options are largely self-explanatory, so I will not take you through all the possibilities, but simply explain what can be done to tweak a tile in a dashboard.

- *Title and Subtitle:* If you edit the elements in these fields, you can change the title and subtitle that were inherited from the source visualization.
- *Display Last refresh time:* Checking this option adds the latest refresh date and time to the tile title.
- *Set custom link:* Adding an URL to this field defines a hyperlink to a completely different web page (rather than the original report containing the source visual) when you click the tile in PowerBI.com.

Exporting the Data Behind the Tile

You can export the data that a visual is based on at any time. It is exported as a CSV file to your Downloads folder. To do this, you need to do the following:

1. Click the tile menu icon to display the tile menu options.
2. Click the page icon.

The data will be exported and a confirmation message will appear. The CSV file will have the same name as the tile title.

Note PowerBI.com does not export the entire dataset that a report is based on. It only exports the subset of data that is used in a visualization.

Pinning the Tile to Another Dashboard

Just as you pinned a visual to a dashboard, you can extend the process by pinning a tile to another dashboard.

1. Click the tile menu icon to display the tile menu options.
2. Click the Pin icon. This will display the Pin to Dashboard dialog that you saw previously in Figure 16-11.
3. Select a new or existing dashboard and click Pin. The tile will be added to the selected dashboard.

Modifying Dashboards

Apart from adding tiles, there other things that you can do with dashboards.

- Resize tiles
- Add other widgets (which can be images, text boxes, or other web content)
- Display dashboards in full screen mode
- Print dashboards

Let's take a quick look at each of these in turn.

Resizing Tiles

As you might expect, tiles can be resized (and moved) on dashboards. All you have to do to move a tile is to click inside the tile and drag it elsewhere on the dashboard pane. To resize a tile, you click the bottom right-hand corner of the tile and drag the mouse up and left (to shrink the tile) or down and right (to enlarge a tile).

Adding Other Widgets (Images, Text Boxes, or Other Web Content)

Dashboards are not limited to displaying only visuals from existing reports. To enhance your message—you can add other web content or existing images. You can even add text boxes to make a specific point.

- At the top right of the PowerBI.com screen, click Add Widget. The Add a widget dialog will appear, as you can see in Figure 16-15.

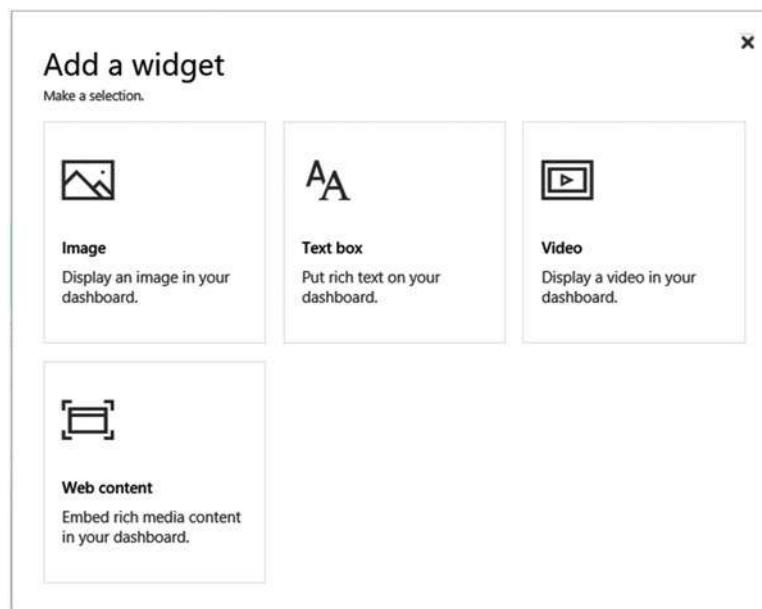


Figure 16-15. Dashboard widgets

- Select one of the available widget options. The appropriate Tile Details pane will appear at the right of the web page where you can enter any necessary information.

The currently available options for widgets are outlined in Table 16-1.

Table 16-1. PowerBI.Com widget types

Widget Type	Description
Image	Image allows you to set a URL for the image that you want to display. You cannot add images from a local disk.
Video	Video allows you to set a URL for the YouTube video that you want to display. Only YouTube videos can be added to PowerBI.com dashboards as of the time of writing.
Web Content	Web Content displays the Tile Details pane where you can add any code that you want to embed in the dashboard page.
Text Box	Text Box displays the Text Box pane where you can enter and format the text that you want to display in a dashboard.

Note If you have begun to add a widget and want to abandon it before you have finished, then simply click the close icon at the top right of the Tile Details pane.

Displaying Dashboards in Full Screen Mode

If you are using PowerBI.com as a presentation tool, or if you simply want to see the contents of a dashboard without any of the other panes or menus, you can always display the dashboard in full screen mode.

1. Click the Enter Full Screen Mode icon (the diagonal two-headed arrow) in the PowerBI.com title bar. The dashboard will appear in full screen mode, as shown in Figure 16-16.

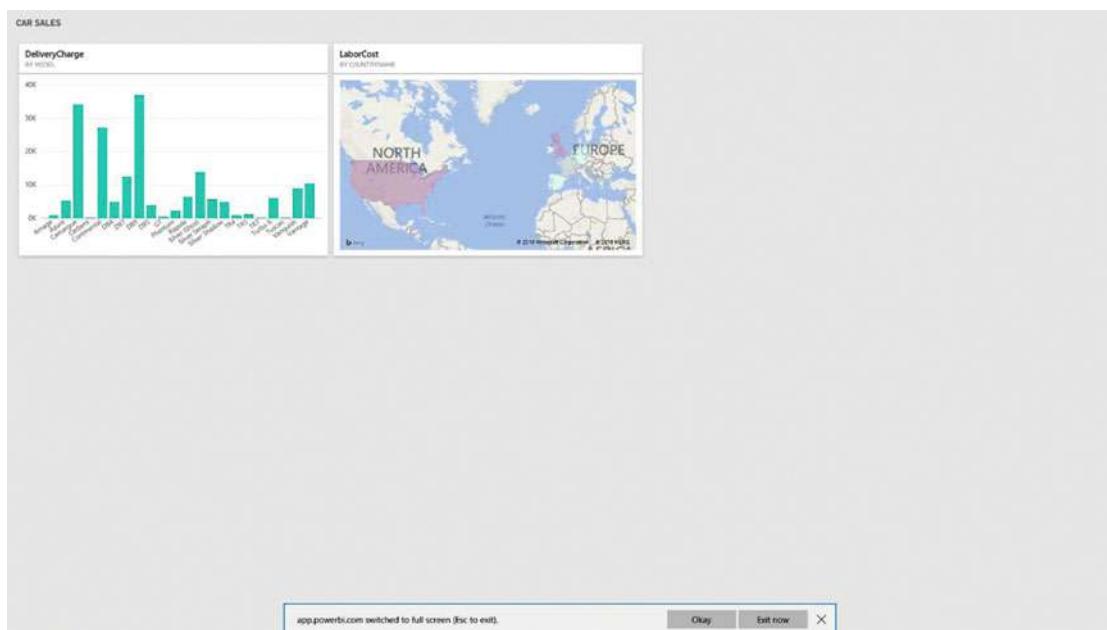


Figure 16-16. Displaying a dashboard in full screen mode

To exit full screen mode, simply press Escape or click the Exit Now button in the toolbar at the bottom of the web page.

Print Dashboards

Although PowerBI.com is built for sharing and interactivity on the Web, it also lets you make paper copies of your dashboards. The following explains how to print a dashboard.

1. Click the More Options button (the ellipses at the top right of the web page). The options menu will appear as shown in Figure 16-17.

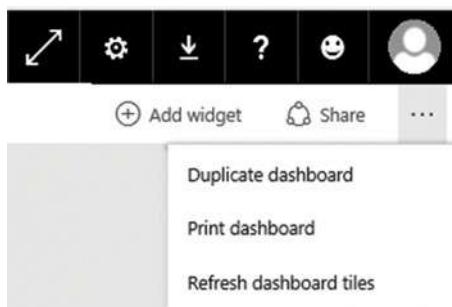


Figure 16-17. The More Options menu

2. Click Print Dashboard. The Print options dialog of your PC or tablet will appear from where you can select a printer-and print the dashboard.

Sharing Dashboards

Creating insightful dashboards may be fun, but it is only a truly meaningful activity if the information can be shared with colleagues. Perhaps inevitably, PowerBI.com will not only allow your co-workers to share your insights, it will actively help you to disseminate the information.

1. Click the Share icon at the top right of the dashboard. The Share Dashboard page will be displayed.
2. Enter the e-mail addresses of the people that you want to share your dashboard with in the upper field.
3. Add a message for the recipients in the lower field. The web page will look something like Figure 16-18.

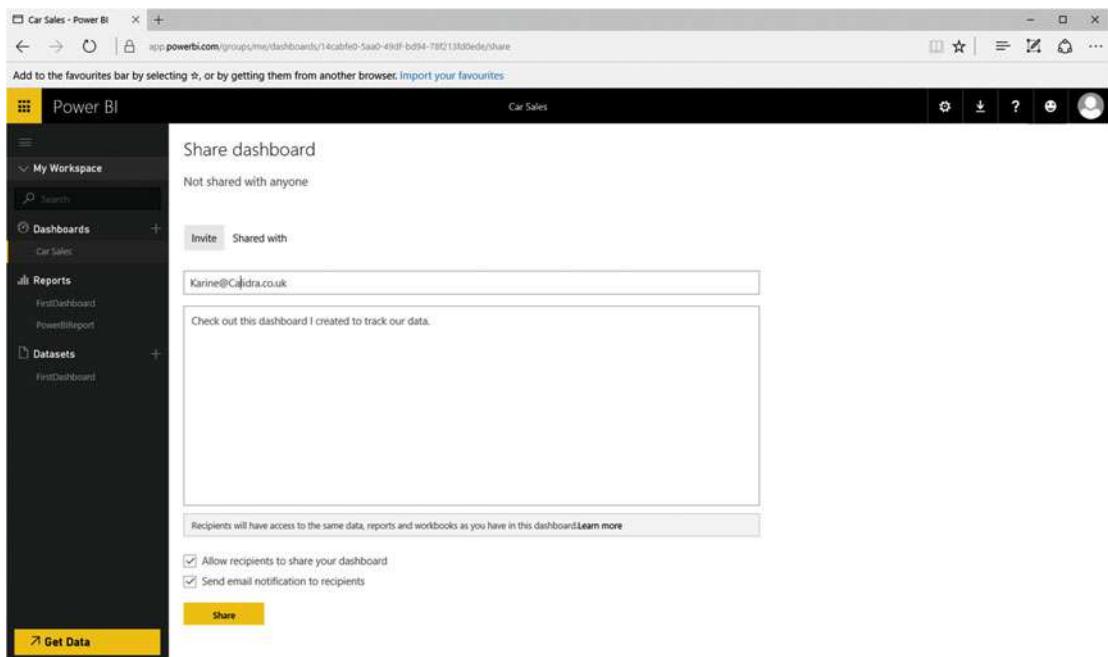


Figure 16-18. Sharing dashboards

4. Click Share.

Your colleagues will receive an e-mail containing the link to the dashboard that you have shared. Clicking this link will display the dashboard.

If you do not want to send an e-mail with a link, you can click the Shared With tab in the Share Dashboard page, and simply copy and paste the link into a file or a Microsoft Lync message (for instance) and enable your co-workers to access your dashboard in this way.

In any case, the recipient with whom you are sharing the dashboard needs a PowerBI.com account.

Creating New Reports in PowerBI.com

Nearly this entire book has been devoted to explaining how you can create powerful, interactive, analytical reports using Power BI Desktop. Moreover, as I mentioned at the start of this chapter, PowerBI.com then comes into the frame to let you share your reports. However, a large part of the knowledge that you have acquired so far in this book can also be used directly in PowerBI.com to create reports directly in the cloud.

Clearly, Power BI Desktop is the preferred solution when it comes to creating Power BI reports. Should you need to, however, you can also use existing cloud-based data or Excel, Power BI Desktop, or CSV files as data sources that you can use to create new reports. As a simple example, the following explains how to use the data in the Power BI Desktop file that you loaded earlier in this chapter to create a new report.

1. In the My Workspace tab, click the dataset FirstDashboard. A new, blank report canvas very similar to those that you have been using in Power BI Desktop will be displayed. You can see this in Figure 16-19.

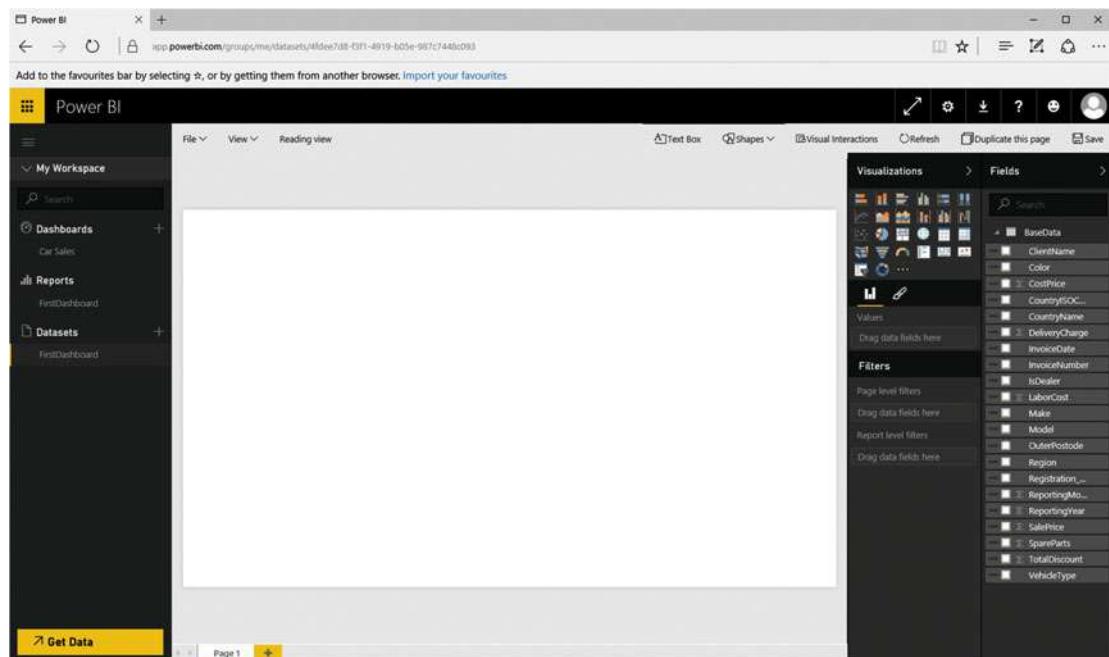


Figure 16-19. The new report canvas

2. Build your report just as you would in Power BI Desktop! This includes creating any of the visualizations that you explored in earlier chapters. You can format the visuals just as you would in Power BI Desktop as well as adding filters and shapes and defining visual interactions much as you saw previously in this book.

Well, I did warn you that it was almost identical to what you have learned so far. However, it is probably worth outlining the essential differences between creating reports directly in PowerBI.com and developing reports in Power BI Desktop.

- **Data:** This is currently limited to Excel, Power BI Desktop, or CSV files, or data that is accessible from a cloud-based source such as
 - Azure SQL database
 - Azure SQL Data Warehouse
 - SQL Server Analysis Services
- **Calculated columns and measures:** It is not currently possible to extend the dataset with calculated columns and measures. So you will have to ensure that all the metrics that you need are in the source data.
- **Custom visuals:** You can import any of the custom visuals that are available on the Power BI site, just as you can when using Power BI Desktop.

There are a few other minor differences, but the following are the key points that you need to remember if you are tempted to create reports in PowerBI.com.

One final thing that you need to know is how to save the report that you have made. In reality, you create a new report by saving the canvas in the dataset.

1. At the top right of the Datasets page, click Save.
2. Enter a name for the report. I chose PowerBIReport for this example. The save dialog will look like the one in Figure 16-20.



Figure 16-20. The save report dialog

3. Click Save. The report will be displayed in the Workspace pane in the Reports tab with an asterisk to indicate that it is a new report. You can see this in Figure 16-21.

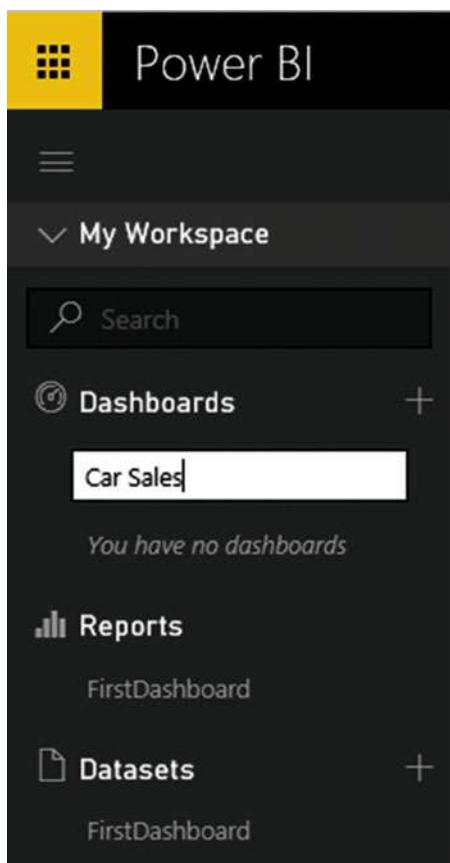


Figure 16-21. A new report created in PowerBI.com

The Power BI App on Tablet Devices

Self-service business intelligence with PowerBI.com is not limited to PCs or web browsers. Indeed, Microsoft has released a set of apps that are tailored to mobile devices on the following platforms:

- iPad and iPhone
- Android
- Windows

So if you have an Android or a Windows phone (or an iPhone for that matter), an iPad, or a Windows or an Android tablet, then you are in luck. You can download the app that is specific to the device that you are using and access reports, dashboards and data on PowerBI.com using the app. All the apps are specifically adapted to their specific platforms and make interacting with the data and visuals even easier and more intuitive.

The following explains how to download the app for your device.

1. Point the browser on your phone or tables to <https://powerbi.microsoft.com/en-us/mobile/>. The web page that you see in Figure 16-22 will appear.

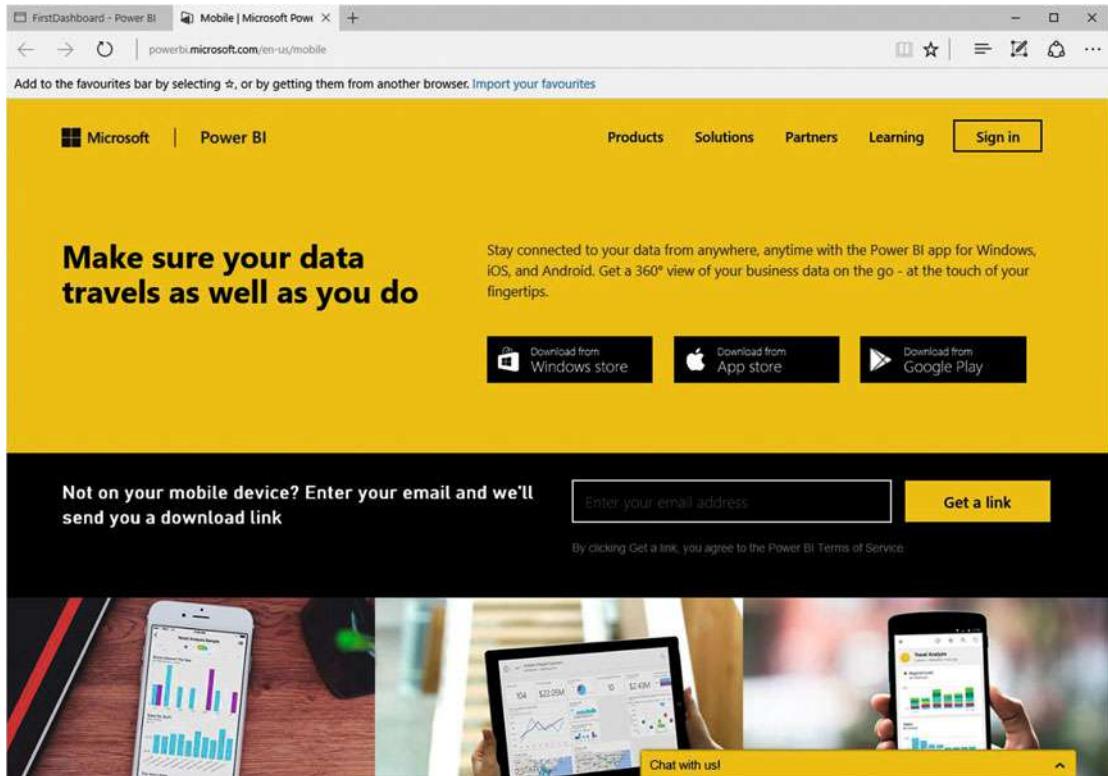


Figure 16-22. The mobile app download page

2. Download the app as you would any other app for your mobile device.
3. Run the app. You will be asked to sign in, so do this using the account that you created for PowerBI.com. Once signed in, you should see a screen rather like the one shown in Figure 16-23 (this example is on a Windows 10 tablet).

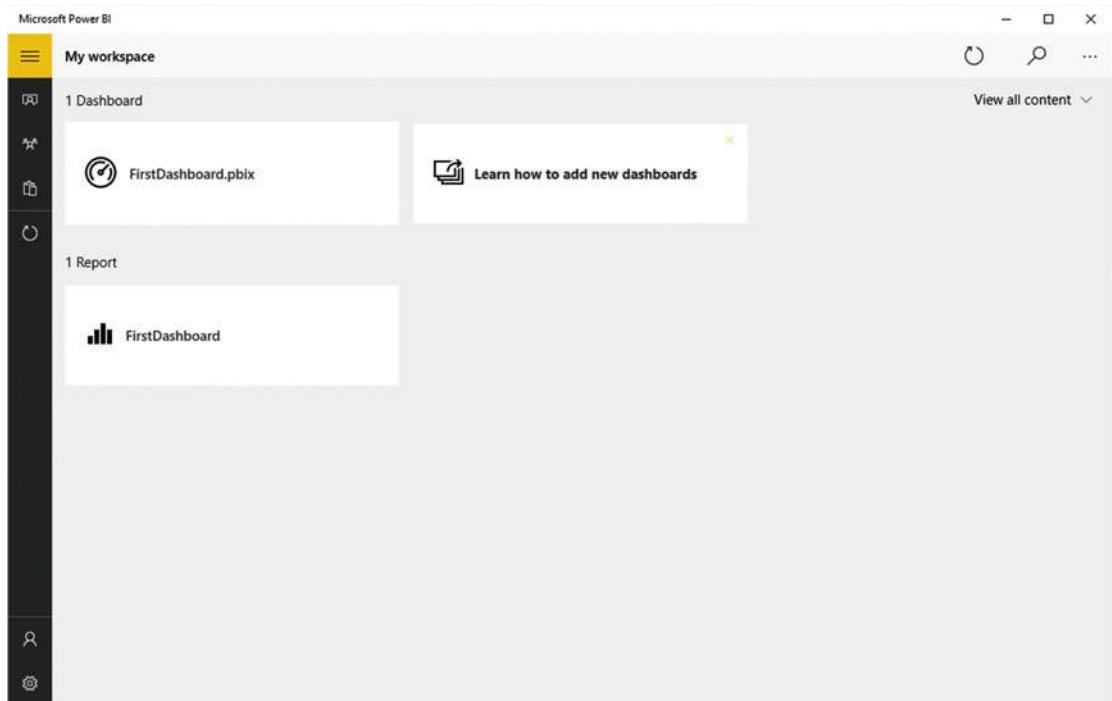


Figure 16-23. The mobile app screen

4. Click the tile FirstDashboard. You will switch to the report that you loaded previously, as shown in Figure 16-24.

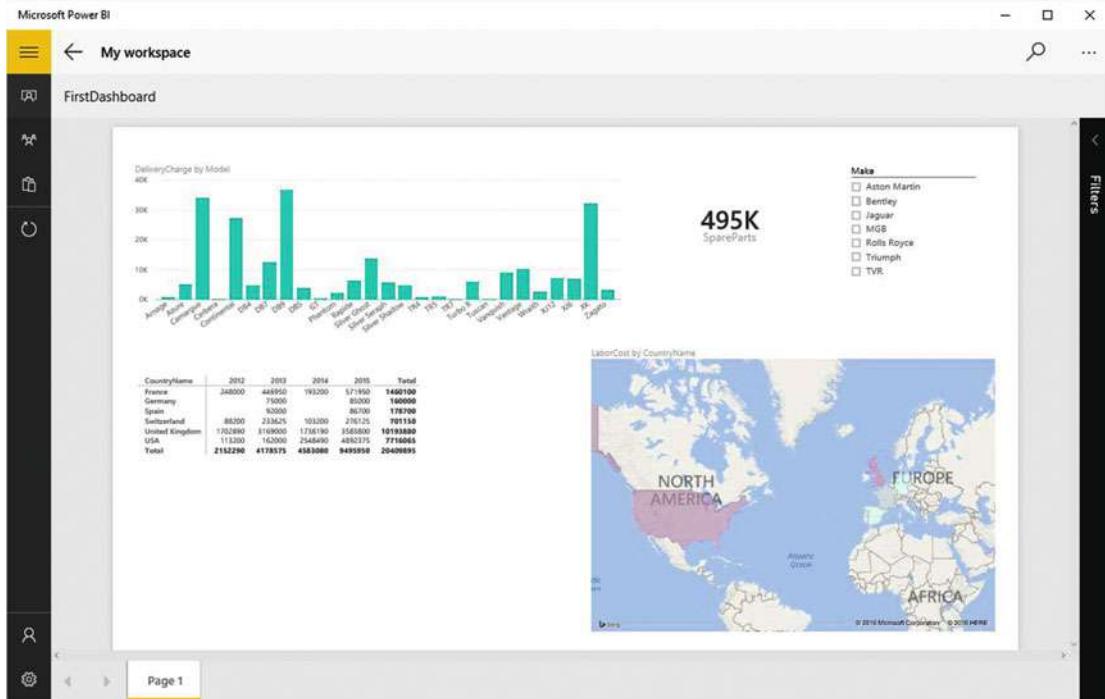


Figure 16-24. A report in a mobile app

From now on, you can interact with your reports on PowerBI.com much as you would using Power BI Desktop or PowerBI.com.

PowerBI.com Gateways

The ability to share your analyses on a platform as powerful as PowerBI.com needs only one more thing to make it an essential business tool. All that is missing is the ability to ensure that the data that you are sharing is up-to-date. Fortunately, the Power BI team has thought of this, too. Their solution is to let you create gateways from your on-premises data to the PowerBI.com site. These gateways allow you to refresh the data in your Power BI Desktop files from the data sources on which they are built. Not only that, but you can refresh the data manually or automatically on a schedule that you define.

Currently, there are two kinds of Power BI gateways:

- Personal
- Enterprise

The essential differences are that the Enterprise gateway is designed for multiple users, and consequently has fine-grained access control, monitoring and auditing. As an added bonus it allows for DirectQuery connections (that you saw in Chapter 2) to SQL Server.

PowerBI.com gateways are a large subject. Consequently, I will only give a brief introduction to the personal gateway here. However, once you understand how PowerBI.com gateways work you should have no difficulties in extending their use to suit your specific requirements.

Downloading a Gateway

A PowerBI.com gateway has to be installed on a local computer first. This is as easy as downloading the gateway application and then carrying out basic configuration.

1. In PowerBI.com, click the downloads icon at the top right of the page, as seen in Figure 16-25.



Figure 16-25. The downloads icon and menu

2. Click Gateways. A new browser tab will open, displaying the Gateways splash screen.
 3. Click Download for the gateway that you wish to install (the personal gateway in this example).
 4. Click Run to install the gateway.
 5. Confirm the User Account Control request to allow the package to run.
- The Gateway preparation dialog will be displayed as shown in Figure 16-26.

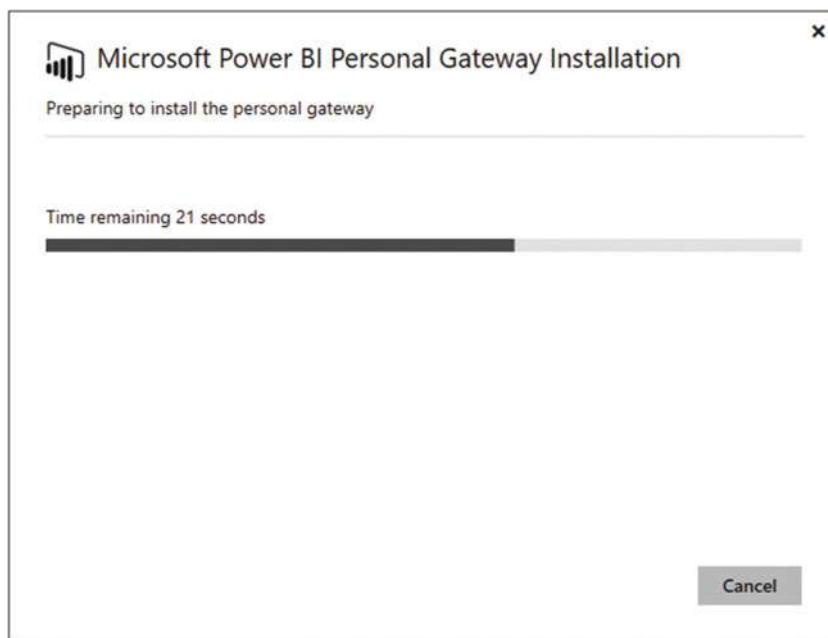


Figure 16-26. The Gateway installation dialog

6. Click Next. The warnings and alerts dialog will be displayed, as shown in Figure 16-27.

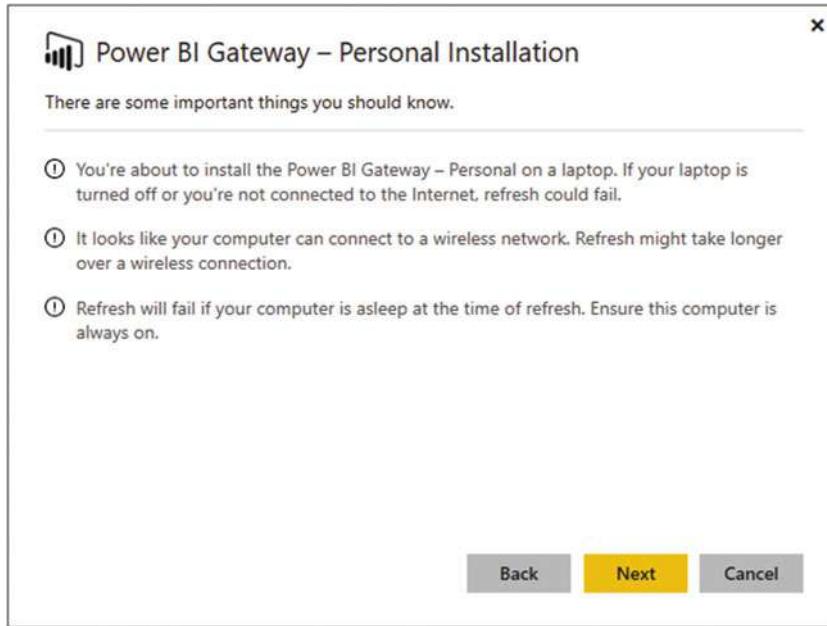


Figure 16-27. The Gateway warnings and alerts dialog

7. Click Next.
8. Accept the license terms and click Next.
9. Accept the destination folder (or change it if you wish) and click Next.
10. After a short wait the final Gateway installation dialog will appear, as shown in Figure 16-28.

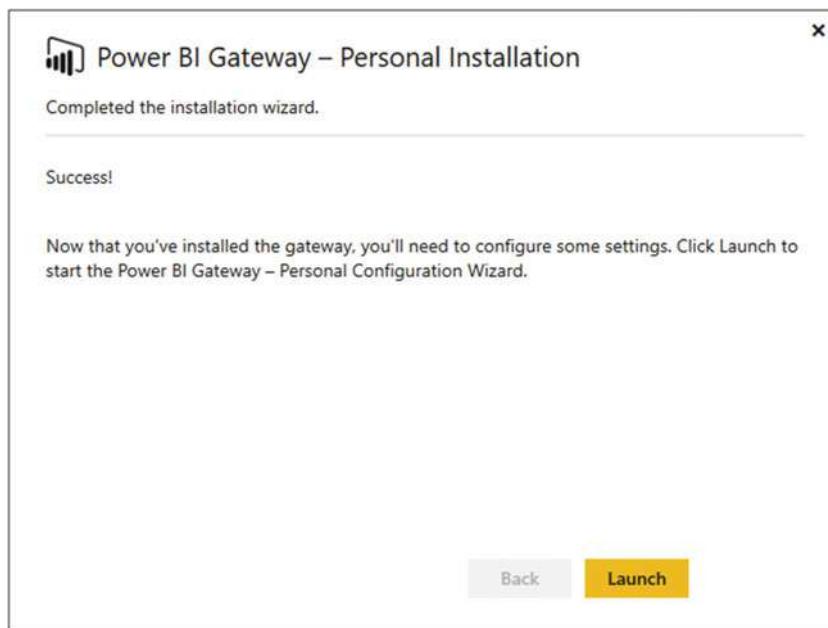


Figure 16-28. The final Gateway installation dialog

11. Click Launch. You will have to sign in to PowerBI.com unless you are already signed in.
12. After a short period of configuration, the final Gateway dialog will appear, as shown in Figure 16-29.

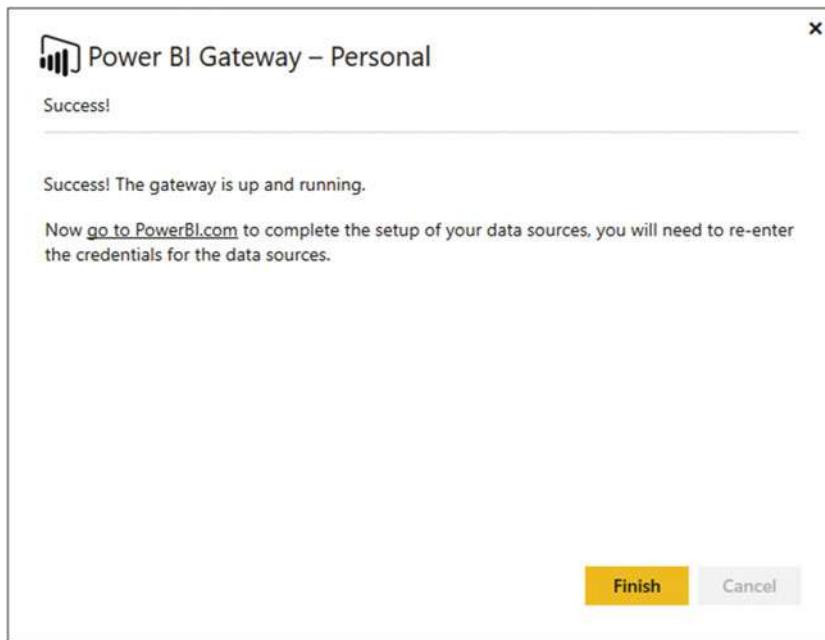


Figure 16-29. The final Gateway dialog

13. Click Finish. You are now ready to configure the gateway to use your specific data sources.

Note PowerBI.com Gateway is an app on your PC. So you can always find it (depending on your version of Windows) among the installed applications on your computer.

Configuring a Gateway

Configuring a gateway is nothing more than specifying which data source(s) you want to refresh either manually or according to a schedule.

Ad Hoc Data Refresh

As an example of how to use a gateway that you have installed on your PC, let's apply a manual refresh to the file that you loaded at the start of this chapter.

1. In PowerBI.com, click the options menu (the ellipses) for the dataset that you want to refresh (FirstDashboard in this example). The options menu will be displayed, as shown in Figure 16-30.

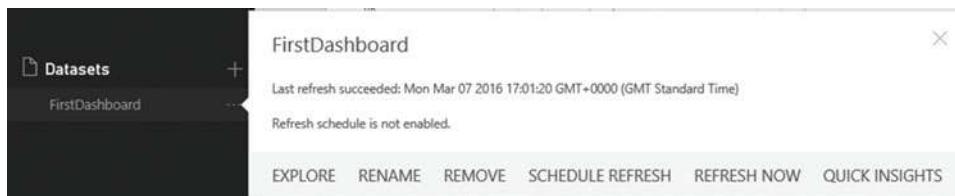


Figure 16-30. The dataset options menu

2. Click Refresh Now.

The dataset will be refreshed with the latest data from the source. This may take a minute or two. Of course, all your visuals will be updated to reflect any changes in the source data.

Scheduled Data Refresh

If you really want to ensure that your data is always up-to-date you can use a personal gateway to carry out a daily data refresh.

1. In PowerBI.com, click the options menu (the ellipses) for the dataset that you want to refresh (FirstDashboard in this example). The options menu will be displayed.
2. Select Schedule Refresh. The Datasets Settings page will be displayed.
3. Expand Schedule Refresh.
4. Set the Keep your data up to date switch to Yes.
5. Choose a refresh frequency (daily or weekly). The page will look like Figure 16-31.

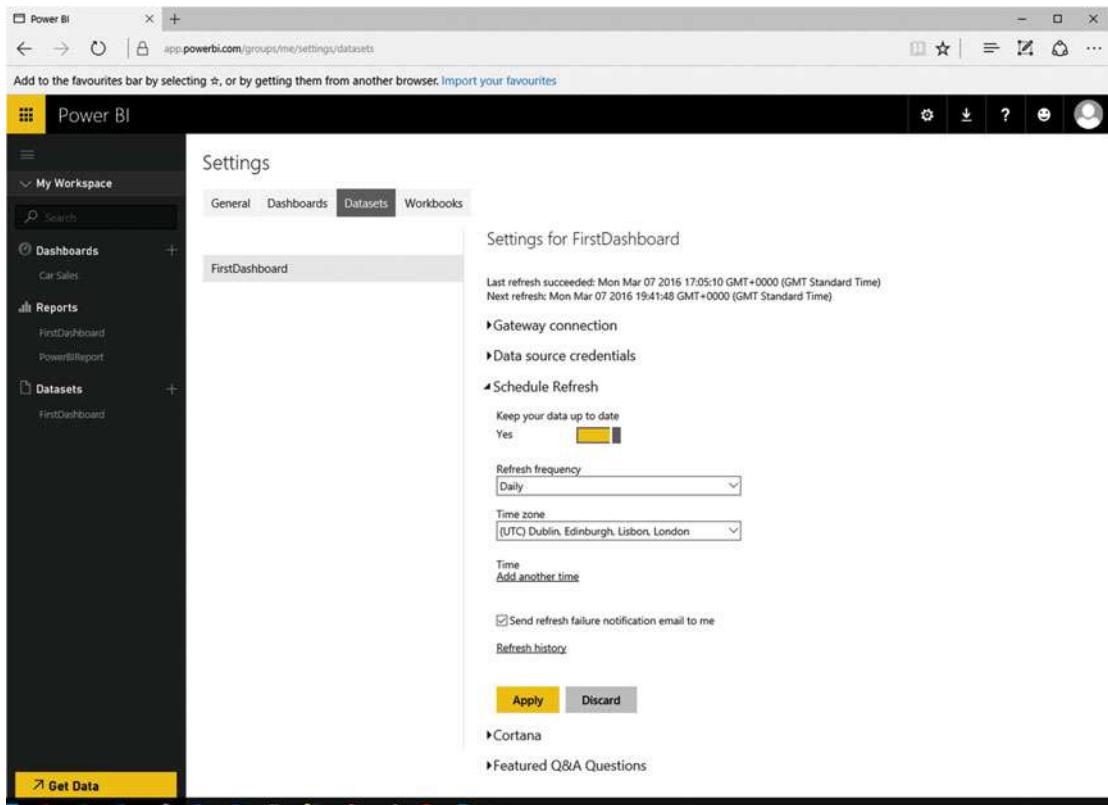


Figure 16-31. Scheduling a data refresh

6. Click Apply.

The selected dataset will now be refreshed according to the schedule that you have chosen. If you have downloaded and installed the Enterprise gateway then you can refresh your data hourly instead of daily. You can also refresh larger amounts of data.

Conclusion

Over the course of this book, you have seen how to develop the data discovery, modeling, and visualization capabilities of Power BI Desktop. As the culmination of your journey into self-service BI, this chapter has shown you the new ways you can share discoveries and collaborate from anywhere using PowerBI.com. This has included adding files to a PowerBI.com, managing the connection to on-premises source data, configuring automated data updates and allowing controlled access to corporate data if you are using the Enterprise gateway.

Once you master and implement these techniques and technologies, PowerBI.com could really become a dynamic online hub for insight and collaboration, data reuse, and interaction among your colleagues. I sincerely hope that you will have fun using Power BI and develop some awesome uses for this amazing technology.

APPENDIX A



Sample Data

Sample Data

If you wish to follow the examples used in this book—and I hope you will—you will need some sample data to work with. All the files referenced in this book are available for download and can easily be installed on your local PC. This appendix explains where to obtain the sample files, how to install them, and what they are used for.

Downloading the Sample Data

The sample files used in this book are currently available on the Apress site. You can access them as follows:

1. In your web browser, navigate to the following URL
<http://www.apress.com/9781484218044>.
2. Scroll down the page and click on the tab Source Code/Downloads.
3. Click the link Download Now, and choose a directory where you will save the file PowerBiDesktopSamples.zip.

You will then need to extract the files and directories from the zip file. How you do this will depend on which software you are using to handle zipped files. If you are not using any third party software, then one way to do this is

1. Create a directory named C:\ PowerBiDesktopSamples.
2. In the Windows Explorer navigation pane, click on the file PowerBiDesktopSamples.zip.
3. Select all the files and folders that it contains.
4. Copy them to the folder that you created in step 1.

Images

The images used in Chapter 16 can be found in the directory C:\PowerBiDesktopSamples\Images.

Sample Databases

If you wish to load data from an SQL Server 2014 database or an Analysis Services database you will need to restore the sample SQL Server and Analysis Services databases that are in the C:\PowerBiDesktopSamples\DatabaseBackups directory.

The CarSalesData database

This database is available in the sample data as the file CarSalesData.Bak in the directory C:\PowerBiDesktopSamples\DatabaseBackups. You will also need to create a directory for the database files. In the code below this is: C:\PowerBiDesktopSamples\Datasets.

Before you can load this database, you will need access to a functioning SQL Server database instance. If you need to, you can download and install the free SQL Server 2014 Express version. It is currently available at the following URL: http://www.microsoft.com/en-in/download/details.aspx?id=42299&WT.mc_id=rss_alldownloads_devresources. Once installed, you will need to restore the database backup. To do this

1. Open SQL Server Management Studio Express.
2. Open a new query window by clicking New Query in the toolbar.
3. Run the following script.

```
USE [master] RESTORE DATABASE [CarSalesData]
FROM DISK = N'C:\HighImpactDataVisualizationWithPowerBI\Database\CarSalesData.bak'
WITH FILE = 1, NOUNLOAD, STATS = 5 ,MOVE 'CarSalesData' TO 'C:\PowerBiDesktopSamples\Dataset\CarSalesData_Data.mdf' ,MOVE 'CarSalesData_Log' TO 'C:\PowerBiDesktopSamples\Dataset\CarSalesData_Log.ldf'
GO
```

The database will be restored, and can be used in the examples.

The Analysis Services Database

To restore the Analysis Services database you will first need a functioning SSAS instance. You can then restore the file CarSalesOLAP.abf or CarSalesTabular.abf (respectively the “classic” and tabular SSAS backup files for the sample data) using the standard SSAS database restoration techniques.

Index

A

Analysis Services database, 502
Appending data, 156
Applied Steps list, 84
Aster plots, 394

B

Background images

- adding, 459
- charts, 460
- free-form images, 461
- uses of, 459

Bubble charts, 352, 368

Bullet charts, 395

C

Calculated columns, 138

Calendar popup, 414

CarSalesData database, 502

Charts, 333, 357

 adjustments

- element sorting, 342–343
- repositioning charts, 342
- resizing charts, 341

bar chart

- after resize, 334
- creation, 334

Fields list, 335

 in visualizations pane, 334

bubble charts, 352, 368

column charts, 338

deletion, 337

Donut charts, 344

drill down, 360

dual-axis charts, 355

- line and stacked column chart, 356
- list and clustered column chart, 355

enhancing charts, 361

 aspect ratio, 367

 axis modification, 365

 chart background, 364

 chart borders, 367

 data colors, 365

 data labels, 363–364

 legends, 361

 titles, 363

funnel charts, 344

line charts, 339

modification, 337

multiple data values, 345

 introductory line chart, 347

 stacked bar chart, 347

pie charts, 339

scatter charts

 description, 350

 flattened hierarchies, 351

stacked column and bar charts, 348

waterfall charts, 354

Cherry-pick techniques, 205

Chord chart, 399

Column charts, 338

CSV files, 44

 definition, 46

 Power BI Desktop file dialog, 45

Custom columns, 138

D

Dashboard enhancement

 adding hyperlink, 454

 adding text boxes, 451

 deleting text boxes, 454

 formatting text boxes, 452

 images, 455

 moving text boxes, 452

 page background color, 454

 removing hyperlink, 454

- Dashboard enhancement (*cont.*)
 - shapes
 - built-in shapes, 461
 - formatting, 462
 - pop-up, 462
 - removing, 465
 - standardizing, 465
 - visuals, 465
 - alignment, 467
 - distribution, 468
 - layering, 466
- Data analysis, 267
 - date/time-based calculations, 267
 - time intelligence, 276
- Data cleansing, 109
 - filling down, 123
 - full record view, 109
 - grouping records, 126
- Power BI Desktop Query
 - Editor context menus, 110
 - column content
 - transformation, 115
 - data type changes, 111
 - data types detection, 113
 - first row as headers, 125
 - replacing values, 113
 - Data mashup
 - appending data, 156
 - identical structures, 156
 - multiple text files, 157
 - custom columns, 138
 - data structures, 161
 - pivoting tables, 163
 - transposing rows
 - and columns, 164
 - unpivoting tables, 161
 - duplicating columns, 133
 - extending datasets, 131–132
 - index columns, 140
 - joining datasets, 131
 - merging columns, 137
 - merging data, 142
 - pending changes alert, 173
 - pivoting and unpivoting data, 131
 - Power BI Desktop Query Editor, 174
 - Power BI Desktop view ribbon, 132
 - query management, 168
 - add as new query, 171
 - duplicating queries, 171
 - enable data load, 172
 - enable data refresh, 172
 - grouping queries, 169
 - organizing queries, 169
 - referencing queries, 171
 - splitting columns, 133
- transformation process, 164
 - adding a step, 167
 - altering process, 167
 - delete step, 166
 - error records, 168
 - modification, 165
 - removing errors, 168
 - rename, 165
 - sequencing, 167
- Data modeling, Power BI Desktop, 176
- Data modification, 79
- Data transformation, 79
 - dataset shaping, 91
 - merging columns, 93
 - removing columns, 93
 - removing duplicate records, 99
 - removing records, 96
 - renaming columns, 91
 - reordering columns, 92
 - row order reversal, 101
 - sorting data, 100
 - filtering data, 101, 105
 - date and time ranges, 104
 - elements, 102
 - numeric ranges, 104
 - text ranges, 103
 - values, 101
 - power BI Desktop data transformation, 80
- Date table
 - CALENDAR() function, 281
 - to data model, 282
 - DateDimension table, 280
 - DAX formulas, 278
 - requirements, 277
 - Sort By columns, 280
- Date/time-based calculations, 267
 - age of cars sold, 273
 - custom Date formats, 271
 - DATEDIFF() function, 274
 - date table creation, 277
 - DAX Date and Time functions, 269–270
 - FORMAT() function, 271
 - intervals, 275
 - NOW() function, 273
 - PARALLELPERIOD() function, 296
 - Predefined Date formats, 272
 - rolling aggregations, 298
 - with PreviousYearSales measure, 291
 - with YearOnYearDelta measure, 293
 - with YearOnYearDeltaPercent measure, 295
 - YEAR() and MONTH() DAX functions, 268
 - DAX Comparison Operators, 228
 - DAX Logical Operators, 236
 - DAX Statistical Functions, 222
 - Delimiter Split, 135

Derived columns, 138
 Donut charts, 344
 Dual-axis charts, 355
 line and stacked column chart, 356
 list and clustered column chart, 355

E

Extract, Transform, and Load (ETL), 35
 Enterprise-grade relational databases, 53
 database connection options, 56
 databases, 58
 security, 60
 server, 56
 SQL, 60
 Microsoft SQL Database dialog, 53
 Navigator dialog, 55
 related group of tables, 55
 SQL Server Database dialog access, 54
 Excel file, 51
 Extending datasets, 79

F, G

Filled maps, 379
 Filters, 401
 advanced numeric filter options, 409
 advanced text filters
 Basic filtering, 417
 Clear Filter, 416
 filter options, 417
 to text field, 415
 (All) filter field, 406
 annotation techniques, 426
 boolean data types, 414
 bubble chart, 445
 charts, 444
 clearing filters, 407
 date and time data types, 410
 date and time filters, 411
 date filter options, 414
 filter hierarchy, 425
 modification, 405
 multiple filters, 419
 numeric data types, 407
 filter options, 409
 logical filter options, 409
 range-filter mode, 408
 page-level filters, 419
 precautions, 425
 remove filters, 421
 report-level filters, 420
 visual-level filters, 402
 Formula language, 205
 Funnel charts, 344

H

Highlighting
 in bubble charts, 442–443
 cross-chart, 439–440, 442
 definition, 438
 remove, 439
 stacked bar chart of costs, 438–439
 Histograms chart, 398

I

Images
 format, 459
 independent, 457
 navigation, 457
 in Power BI Desktop, 455
 remove, 458
 report backgrounds, 459
 resize, 458
 scaling, 459
 sources, 456
 Index columns, 140

J, K

Joining datasets, 79

L

Line charts, 339

M, N, O

Map visuals
 bing maps, 372
 types, 371
 Measures, data model, 239, 266
 advanced aggregations, 248
 basic aggregations, 241
 calculation options, 266
 complex filters
 CALCULATE() function, 256
 multiple filters, 257
 cross-table measures, 245
 DAX, AVERAGEX() function, 249
 filter context, 251–252
 FILTER() function, 263
 filters, 252
 numeric filters, 254
 text filters, 253
 iterative functions, 250
 multiple measures, 243
 number of cars sold, 239
 percentage calculation, 258

- Measures, data model (*cont.*)
 - ALLEXCEPT() function, 262
 - ALL() function, 259
 - ALLSELECTED() function, 261
 - discard filters, 260
 - query context, 251
 - RANKX() function, 265
 - row context, 251
- Merging columns, 137
- Merging data
 - adding reference, 143, 145
 - aggregate data, 142, 145, 148
 - expand and aggregate buttons, 154
 - individual query, 142
- joins
 - correct and incorrect, 152
 - datasets, 152
 - data tables, 153
 - on multiple columns, 150
 - types, 149–150
- look up data, 142
- Microsoft access databases, 52
- Microsoft SQL Server Analysis Services Database, 63
 - analysis services data sources, 66
 - Add Items, 66
 - Collapse Columns, 68
 - attributes and measures selection, 65
 - credentials dialog, 64
 - Multidimensional/Tabular model, 63
- Microsoft SQL Server Analysis Services tabular database, 68
- Microsoft's Self-Service Business Intelligence solution, 2
 - dashboards and reports, 2–3
 - import data, 2
 - model data, 3
 - Power BI Desktop file, 3
- Multiple filters, 419

- P, Q
- Page-level filters, 419
- Pie charts, 339
- PowerBI.com, 471
 - dashboards, 479
 - adding tiles, 480
 - creation, 479
 - deleting tiles, 482
 - export data, 484
 - modify, 483–484
 - pinning a tile, 484
 - print, 487
 - save report dialog, 490
 - sharing, 488
- gateways, 494
 - adhoc data refresh, 498
 - configuration, 498
 - download, 495
 - installation, 495
 - scheduled data refresh, 499
 - warnings and alerts dialog, 496
- Power BI Desktop files, 475
 - adding files, 477
 - sign-in, 476
- publishing reports, 471–472
- reports, 478, 489
 - on Tablet devices, 491
- Power BI desktop, 16
 - connection security, 75
 - dashboards, 10, 28, 187
 - data category options, 187
 - default summarization, 188
 - Get Data dialog, 11
 - Navigator dialog, 13
 - power, 10
 - report window, 14
 - simplicity, 10
 - sort by columns, 189
 - Windows Open File dialog, 12
 - data loading, 40
 - CSV files, 44, 46
 - Excel file, 51
 - Microsoft Access databases, 52
 - text files, 47
 - web pages, 40
 - XML files, 49
 - data load process, 14
 - Navigator Data Preview, 15
 - Navigator window, 15
 - data modeling, 176
 - data modification, 16
 - data sources, 36, 71
 - Azure, 38
 - databases, 37
 - file, 37
 - less corporate, 39
 - data types, 183
 - data view ribbons, 177
 - Home ribbon, 177
 - Modeling ribbon, 177
 - description, 35
 - Extract Transform Load, 35
 - formatting options, 184
 - currency formats, 184–185
 - import excel and power view items, 31
 - installation
 - download page, 4
 - download selection page, 5
 - installation progress dialog, 8

- requirements, 4
- save/run download popup, 5
- setup destination dialog, 7
- setup final configuration dialog, 8
- Setup Licensing dialog, 6
- manipulating columns, 180
 - delete, 182
 - move, 183
 - rename, 180
 - setting column widths, 183
- manipulating tables, 179
 - column selection, 179
 - delete, 179
 - rename, 179
- maps
 - highlight segments, 378
 - multivalue series, 377
 - positioning, 376
- map visuals, 372
- Microsoft self-service, 35
- options, 17
- Query Editor, 177
- refreshing data sources, 74
- remove from computer, 9
- relational databases (*see* Enterprise-grade relational databases)
- reports, 29
 - adding pages, 30
 - deleting pages, 30
 - duplicating pages, 31
 - moving pages, 31
 - renaming pages, 30
- reuse data sources, 71
- sorting data, 190
- splash screen, 10
- SQL Server Analysis Services tabular data
 - (*see* Microsoft SQL Server Analysis Services tabular database)
- SSAS cubes (*see* Microsoft SQL Server Analysis Services Database)
- table relationships, 191
 - advanced options, 201
 - automatic relationships dialog, 199
 - creating relationships, 194
 - data view and relationship view, 192
 - deactivating relationships, 201
 - deleting relationships, 199
 - display options, 193
 - Manage Relationships dialog, 196
 - managing relationships, 200
- visualizations, 17
 - card visualization, 24
 - dashboard arrangement, 26
 - delivery charge, 21
- Fields pane, 18
- filled map visualization, 23
- labor cost map, by country, 24
- matrix icon, 19
- slicer icon, 25
- Power BI Desktop data model, 239
 - cascading column calculations, 217
 - column-based calculations, 205
 - concatenating column contents, 208
 - dashboard visualizations, 206
 - formula bar, 237
 - IF() function, 226
 - alerts, 227
 - comparison operators, 228
 - complex logic, 232
 - DAX logical and information functions, 234
 - exception indicators, 226
 - flagging data, 228
 - logical function, 236
 - logical operators, 235
 - multiline formulas, 232
 - multiple nested IF() statements, 230
 - nested IF() functions, 229
 - InvoiceLines table, 216
 - Gross Margin, 216
 - for linked calculations, 217
 - pop-up list, 217
 - measures, 239
 - new columns, 206
 - counting reference elements, 220
 - custom number formats, 225
 - formatting, 222
 - naming, 207
 - predefined currency formats, 224
 - safe division, 219
 - statistical functions, 221
 - simple calculations, 212
 - math operators, 213
 - rounding values, 214
 - truncation functions, 215
 - tweaking text, 210
 - DAX formulas, 212
 - text functions, 211
- Power BI Desktop data transformation, 80
 - after data load, 81
 - before data load, 82
 - choices, 80
 - query editor, 83
 - query/load, 82
- Power BI Desktop Query Editor, 83
 - Applied Steps list, 84
 - delete, 85
 - rename, 85

■ INDEX

Power BI Desktop Query Editor (*cont.*)

- ribbons, 85
 - Add Column ribbon, 89
 - Home ribbon, 86
 - Transform ribbon, 87
 - View ribbon, 90

Power BI Desktop Query Editor context menus, 110

- changing data types, 111
- column content transformation, 115
 - date transformation, 120
 - duration data, 122
 - filling down, 123
 - leading and trailing spaces removal, 116
 - number calculation, 118
 - number transformations, 117
 - text transformation, 115
 - time transformation, 121
- data types detection, 113
- first row as headers, 125
- replacing values, 113

Power BI Desktop view ribbon, 132

■ R

- Radar charts, 394
- Report-level filters, 420

■ S

- Sample data, 501–502
- Sankey diagram, 399
- Scatter charts
 - description, 350
 - flattened hierarchies, 351
- Slicers
 - add, 430
 - apply, 431
 - characteristics, 429
 - charts, 436–438
 - column and bar charts, 446
 - highlighting, 438
 - clear, 432
 - delete, 432
 - description, 429
 - element selection, 434
 - header, 435
 - interactive selection, 429
 - items, 435
 - modify, 432, 434
 - orientation, 433
 - setting X and Y coordinates, 435
 - visual interactions, 448
- Splitting columns
 - classic cases, 133
 - custom columns, 138–139

Delimiter, 134–135
number of characters, 136

Streamgraph, 397
Sunburst charts, 396

■ T, U

- Text-based visualizations
 - cards, 324
 - display units, 328
 - formatting, 325
 - multirow cards, 328, 330
- data types, 308
 - text fields,
 - pop-up menu, 308
- matrix
 - column matrix, 322
 - row matrix, 319
 - sort data, 323
- Power BI Desktop dashboards, 302
- table granularity, 318
- tables, 302
 - background modification, 315
 - borders, 316
 - column order, 307
 - column sorting, 317
 - column width, 312
 - copy tables, 306
 - delete tables, 305
 - Fields list, 304
 - font sizes, 312
 - formatting options, 313
 - for sales per client, 303
 - number formatting, 312
 - removing columns, 307
 - resize, 306
 - Row totals, 309
 - switching visuals, 331
- Text files, 47–48
- Threshold selector, 408
- Time intelligence, 267, 276, 284
 - CALCULATE() function, 290
 - month-to-date calculation, 284
 - quarter-to-date calculation, 284
 - range of dates, 288
 - time-based analysis, 286
 - year-to-date calculation, 284
- Tornado chart, 397
- Tree maps, 382
 - background, 384
 - category labels, 384
 - data colors, 384
 - legend, 384
 - lock aspect, 384
 - title, 384

V

- Visualization-level filter, 417–418
- Visual-level filters, 402
 - adding filters, 403
 - automatic creation, 404
 - filtered chart, 404
 - Filters well, 402
- Visuals, 387
 - aster plot, 394
 - bullet charts, 395
 - chord charts, 399
 - community agreement dialog, 391
 - custom visuals, 390
 - download dialog, 390
 - enable custom visuals, 393
 - filled maps, 379
 - formatting, 379
 - gauges, 385
 - elements, 385
 - gauge formatting, 386
 - histogram chart, 398
 - import confirmation dialog, 392
 - import visuals dialog, 391

maps

- bing maps, 372
- in Power BI Desktop, 372, 376
- map visuals, 371
- Power BI visuals gallery, 388
- radar charts, 394
- Sankey diagram, 399
- streamgraphs, 397
- sunburst charts, 396
- tornado chart, 397
- tree maps, 382
- using geographical data, 374
 - Country field and Town field, 376
 - data category, 376
- word clouds, 396

W

- Waterfall chart, 354
- Word clouds, 396

X, Y, Z

- XML files, 49–50