

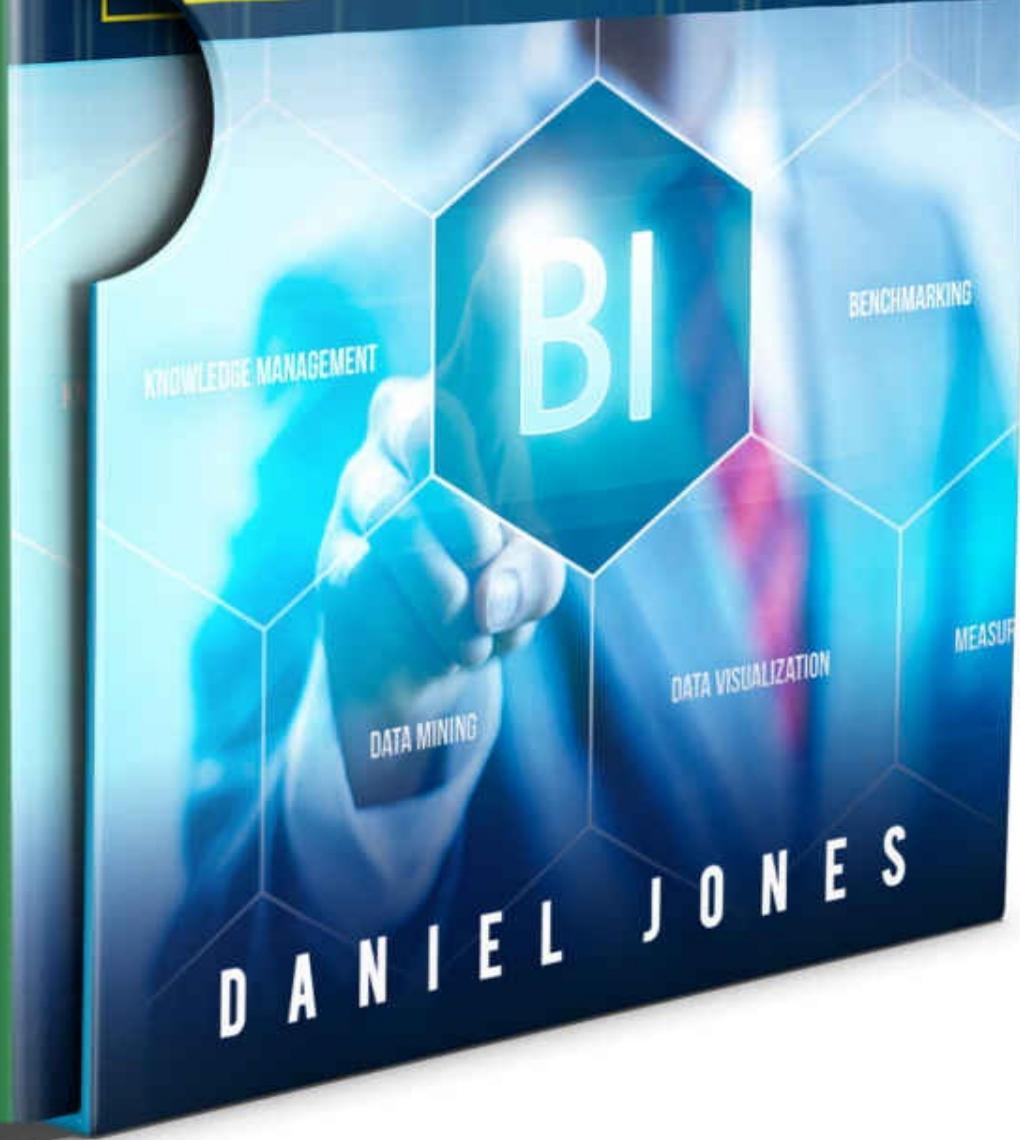
# POWER BI

THIS BOOK INCLUDES

A COMPREHENSIVE GUIDE OF TIPS AND TRICKS  
TO LEARN THE FUNCTIONS OF POWER BI

SIMPLE AND EFFECTIVE STRATEGIES TO LEARN  
THE FUNCTIONS OF POWER BI AND POWER QUERY

AN ADVANCED GUIDE TO LEARN  
THE ADVANCED REALMS OF POWER BI



POWER BI

AN ADVANCED GUIDE TO LEARN THE  
ADVANCED REALMS OF POWER BI

DANIEL JONES

BOOK 3

POWER BI

SIMPLE AND EFFECTIVE STRATEGIES  
TO LEARN THE FUNCTIONS OF  
POWER BI AND POWER QUERY

DANIEL JONES

BOOK 2

POWER BI

A COMPREHENSIVE GUIDE OF TIPS  
AND TRICKS TO LEARN THE  
FUNCTIONS OF POWER BI

DANIEL JONES

BOOK 1

---

# **POWER BI**

---

**DANIEL JONES**

**Copyright 2021 by Daniel Jones - All rights reserved.**

This document is geared towards providing exact and reliable information in regards to the topic and issue covered. The publication is sold with the idea that the publisher is not required to render accounting, officially permitted, or otherwise, qualified services. If advice is necessary, legal or professional, a practiced individual in the profession should be ordered.

- From a Declaration of Principles which was accepted and approved equally by a Committee of the American Bar Association and a Committee of Publishers and Associations.

In no way is it legal to reproduce, duplicate, or transmit any part of this document in either electronic means or in printed format. Recording of this publication is strictly prohibited, and any storage of this document is not allowed unless with written permission from the publisher. All rights reserved.

The information provided herein is stated to be truthful and consistent, in that any liability, in terms of inattention or otherwise, by any usage or abuse of any policies, processes, or directions contained within is the solitary and utter responsibility of the recipient reader. Under no circumstances will any legal responsibility or blame be held against the publisher for any reparation, damages, or monetary loss due to the information herein, either directly or indirectly.

Respective authors own all copyrights not held by the publisher.

The information herein is offered for informational purposes solely, and is universal as so. The presentation of the information is without a contract or any type of guarantee assurance.

The trademarks that are used are without any consent, and the publication of the trademark is without permission or backing by the trademark owner. All trademarks and brands within this book are for clarifying purposes only and are owned by the owners themselves, not affiliated with this document.

# TABLE OF CONTENTS

---

## POWER BI

### A Comprehensive Guide of Tips and Tricks to Learn the Functions of Power BI

#### Introduction

Business Intelligence Software (BI)

Treatment Of Data In Power Query

The Query Editor Interface

Data Processing

Data Relationship And Modeling

Calculations And Dax

Visualization

Principles For Creating Visuals And Reports

Showing Compositions And Flows

Power BI

Sharepoint

Power Bi For Mobile

Power BI And Excel

#### Chapter One: Review Of Getting Started With Microsoft Power BI

Basic Theoretical Concepts

Explanation Of The Practical Example Of Power Bi

Connection With Excel File

Connection To A Web Page

Data Transformation in Power BI

Apply The Transformations Performed In Power Query

Creating A Report With Graphics

Dynamic Filters

MAPS

#### Chapter Two: Introduction to Power Pivot

Power Pivot Ribbon in Excel

[Steps to Enable Power Pivot in Excel](#)  
[Power Pivot Management Window](#)  
[Excel Power BI Components - Power Pivot](#)

### **Chapter Three: Introduction to DAX**

[Where Are We On The Road?](#)  
[What is DAX?](#)  
[DAX Syntax](#)  
[DAX Functions](#)  
[Aggregation Functions](#)

### **Chapter Four: DAX in Practice**

[Understanding the Contexts](#)  
[Row Context](#)  
[Query Context](#)  
[Filter Context](#)  
[DAX Functions](#)  
[Aggregation Functions Ending in "X"](#)  
[Practical Example of DAX](#)  
[Using DAX Studio and Excel as Power BI Measurement Verification Tools](#)  
[DAX Studio: A Really Useful Tool](#)  
[Power BI and the SSAS Service](#)

### **Chapter Five: Power BI and Power Query (M Language)**

[The Role of BI Self-Service in the Development of a DataWarehouse](#)  
[Power Query much more than Self-Service BI](#)  
[Population Register. The Source Data Source](#)  
[Population File Record Design](#)  
[Data Import with Power BI](#)  
[Query Editor. The Power Query Development Environment](#)  
[Modifying the Default Names](#)  
[Optimization of Data Load Times](#)  
[Creation of Columns from the Source Data](#)

## **Chapter Six: Data Model Using Power Query**

Nationality. Import from Excel

## **Chapter Seven: Power Pivot Engine To Design A Population Data Model**

Data Designer Introduction

Data Designer Population Count Measure

Report Designer Elementary Operations in the Use of Controls

Report Designer Results Display

Measurement Control Test

Tables Need Relationships

Relationships Designer. Relationship Creation

Data Designer Hierarchy Creation

Demographic Structure Indicators

Demographic Structure Indicators

## **Conclusion**

## **References**

## **POWER BI**

### **Simple and Effective Strategies to Learn the Functions of Power BI and Power Query**

## **Power BI: A Disruptive Reporting Platform**

Introduction to Power BI

Power BI Online Service

Power BI Desktop

Power BI Mobile

Manage Data Sources in Power BI Desktop

Benefits of Power BI

Manage Data Source

Q&A (Natural Language Query)

## **Strategies to Learn Power BI Functions**

Introduction

Power BI Architecture

[Creating, Publishing and Scheduling a Dashboard](#)

[Creating Parameters in Power BI](#)

[Calling Oracle Package Function in Power BI](#)

[Limitations in Power BI](#)

## **Power BI Desktop Functionalities**

[Power BI Desktop Introduction](#)

[Data Connectivity Modes](#)

[Pbit Template File Implementation](#)

[Assigning Parameter Values Dynamically](#)

[Maps in Power BI Desktop](#)

## **Power BI Real-Time Dashboard**

[Introduction](#)

[Power BI Preview](#)

[Navigation Pane](#)

[Dashboards](#)

[Dashboard Tiles](#)

[Q&A Question Box](#)

[Functionalities of Power BI Preview](#)

[Share a Power BI Dashboard](#)

[Various Data Sources](#)

[Power BI Designer File](#)

[Refresh your Data](#)

[Steps to Refresh Schedule](#)

## **Power BI for Report Generation and Mail**

[Introduction](#)

[Setup](#)

[Report Generation Using Visualization](#)

[Publishing the Report](#)

[Subscription and Mailing of the Report](#)

[Report Creation in Power BI Desktop with AX 2012](#)

[Developing Power BI Reports with NAV 2016](#)

[Integration of Power BI Modules with MS Excel](#)

[Steps to Enable Plugins](#)

[Tips for Using the Power BI Dashboard](#)

## **[Dynamic Row Level Security in Power BI](#)**

[Introduction to Power BI](#)

[Benefits of Power BI](#)

[Steps to Define a Role in Power BI Desktop](#)

## **[Toggle Button and Tooltip Features in Power BI](#)**

[Introduction](#)

[Designing a Dashboard using the Imported Dataset](#)

[Toggle Button with Bookmark Feature](#)

[Report Page Tooltip in Power BI Desktop](#)

[Publishing Reports on Apps in Power BI](#)

[App Workspace](#)

[Sourcing an Existing Power BI Report into a New Power BI Report Using Live Connection](#)

## **[Power BI and Share Point](#)**

[Introduction](#)

[Power BI Desktop](#)

[Views in Power BI Desktop](#)

[Build Reports](#)

[Creating Charts For Sharepoint List Data Using Power BI](#)

[Integrating Power BI with SharePoint](#)

[Create Relationships between Two Charts](#)

[Create Custom and Calculated Columns in Power BI](#)

[Create a Calculated Column](#)

[Share the SharePoint Page with PowerBI Report Embedded in It](#)

[Hosting of Power BI Reports in SharePoint](#)

[Publish the Power BI Report on Dynamics 365](#)

## **[Power Query for Report Generation](#)**

[Introduction](#)

[Setting up Power Query](#)

[Power Query for Excel](#)

[ETL Process](#)

[Synoptic Panel in Power BI Desktop](#)

[Synoptic Panel by SQLBI](#)

## **[Filters and Slicers in PowerBI Report](#)**

[Filters in PowerBI](#)

[Different Types of Filters in PowerBI](#)

[Slicers](#)

[Slicers vs. Filters](#)

[How to Toggle in Power BI Report Using Images](#)

## **[Cognos Self Service BI](#)**

[Self Service BI: Overview](#)

[Self Service BI: Challenges](#)

[Why Self Service is Required in Today's World](#)

[The Business Case for Self Service BI](#)

[Need for BI Users to be More Empowered and BI Tools to be More Exhaustive](#)

[Features and Functions](#)

[The Tradeoffs](#)

## **[Conclusion](#)**

# **POWER BI**

***An Advanced Guide to Learn  
the Advanced Realms of Power BI***

## **Introduction**

[Know Your Role in Power BI](#)

[Know More about Power BI](#)

## **Chapter 1 : Power BI Building Blocks**

[Visualizations](#)

[Datasets](#)

[Reports](#)

[Dashboards](#)

[Tiles](#)

[Combined Together](#)

[Using the Power BI Service](#)

[Creating Dashboards using Cloud Services](#)

[Updating Data from Power BI Service](#)

[Power BI Desktop](#)

## **[Chapter 2 : Power BI Reports](#)**

[Copy-Paste Reports](#)

[Hiding the Report Pages](#)

[Different Filter Types in Reports](#)

[How to Add Filters to Reports](#)

[Role of Filters and Highlighters](#)

[Filters Pane](#)

## **[Chapter 3 : High-Density Line Sampling](#)**

[How Does Line Sampling Actually Work?](#)

[Tooltips in Displayed Data](#)

[Limitations in the Algorithm](#)

## **[Chapter 4 : Connecting with Data Sources in Power BI](#)**

[Steps to Connect Data in Power BI](#)

[Import Data from Microsoft Excel](#)

[Excel Workbook Types Supported by Power BI](#)

[Connecting to Excel workbook using Power BI](#)

[Fetching Data from Power BI Desktop](#)

[Editing Parameter Settings](#)

## **[Chapter 5 : Power BI Real-Time Streaming](#)**

[Types of Real-Time Datasets](#)

[How to Push Data in Datasets](#)

[Setting Up Real-Time Streaming Dataset](#)

## **[Chapter 6 : Publish Using Microsoft Excel into Power BI](#)**

[How to Publish the Workbook](#)

[Publishing a Local File](#)

[Upload Option](#)

[Export Option](#)

[Reducing Workbook Size to Display in Power BI](#)

[Removing the Data from the Worksheet](#)

## **[Chapter 7 : Working with R Scripts](#)**

[Importing Data Using R Scripts](#)

[Transforming Data Using R Scripts](#)

[Creating R Visualizations Using R Scripts](#)

[Importing Custom Visuals Using R](#)

[Getting the Best Out of R in Power BI](#)

## **[Chapter 8 : Working with Parameters](#)**

[Connection-Specific Parameters](#)

[Connecting to Data Sources](#)

[Add Parameters to Your Filter Data](#)

[Add Parameters to Control Statement Logic](#)

[Naming Dataset Objects Using Parameters](#)

[Using Parameters in Data View](#)

## **[Chapter 9 : Working with SQL Server Data](#)**

[Retrieving an SQL Server Table](#)

[Work with Relationships](#)

[How to Merge Datasets](#)

[Retrieving SQL Server Data Using T-SQL Queries](#)

[Make the Most of SQL Server Data](#)

## **[Chapter 10 : Working with Power Query M Formula Language](#)**

[How to Retrieve Data from CSV Files](#)

[How to Remove Columns from Datasets](#)

[Promote the First Row to Headers](#)

[Renaming the Dataset Columns](#)

[Filtering Dataset Rows](#)

[Replacing A Dataset's Column Values](#)

[Changing the Column Values Case](#)

[Adding Calculated Columns to Datasets](#)

[Navigate the Power Query M Formula Language](#)

## **Chapter 11 : How to Visualize SQL Server Audit Data**

[Setting Your Test Environment Up](#)

[Generate the Test Audit Data](#)

[Creating the Connection to SQL Server Audit Data in Power BI Desktop](#)

[Adding Tables](#)

[Adding a Matrix](#)

[Adding Visualizations](#)

[Adding a Gauge](#)

[Using Power BI to Visualize Audit Data](#)

## **Conclusion**

# **Power BI**

---

***A Comprehensive Guide of Tips and Tricks to Learn the Functions of Power BI***

**DANIEL JONES**

# Introduction



## Business Intelligence Software (BI)

Business intelligence software is capable of turning large volumes of data into analytical and summary dashboards. They are configurable and capable of automatically executing the routine of data collection, treatment<sup>1</sup>, analysis, and visualization. The term BI began to be used for technology in mid-1958. From the initial letters of business intelligence, it was tied to software capabilities of transforming the way much information was consumed in Hans Peter Luhn's article "[A Business Intelligence System](#)." The computer scientist, who was currently working for IBM, described BI software as automatic systems designed to propagate information to various sectors of the economy. His research, among the developments of more than 80 patents, was initially applied in IBM analysis software.

Nowadays, there are many business intelligence software in the world that provides the basis for the functionalities that software of this category should offer, such as data analysis through calculations using different languages, connection to external sources in various ways and data visualization with specific graphics and with its own design and configuration. Given all of its similarities and differences, one of the most relevant in Power BI is the learning curve.

Perhaps because of the strategy of making some of its resources available in Excel before its release (Power Pivot and Power Query), but certainly not only because of that but because of the very way it was logically architected, the learning of languages and the process of building projects in Power BI is considered by us one of the fastest and most consistent, enabling results at a higher speed than many software in the same category.

Research that supports and shows some of the reasons why we, as business intelligence experts, chose Power BI as the leading BI software is published by Gartner, a leading global company focused on research and consulting in several areas of information technology. Every year an extensive report and graph summarizing it, called the "Magic Quadrant of Analytics and Business

Intelligence Platforms," are published with analysis of BI software, being analyzed 15 critical aspects involving infrastructure, data management, analysis and content creation, discovery and sharing, ease of use and return on investment. Power BI is identified as "Microsoft" and is in the top direct quadrant, i.e., the market leaders today.

After downloading, open Power BI Desktop and you will observe each numerical legend commented below.

1. This is the Power BI ribbon containing the Home Page, Display, Modeling, and Help. It is through this interface that the database connections, the Query Editor (Power Query), the data modeling options, the panel layout and others are accessed.
2. The three buttons located on the left vertical are called the Report, which is the blank panel displayed in the previous image — data, which displays the data imported into the Query Editor. And Relationships, which show the tables and allows the relationship of their key fields.
3. Set of Power BI views. Initially, only the default views will be shown, but hundreds of others can be added by the Marketplace, and can also be created with the R language.

## **Treatment Of Data In Power Query**

Power Query Editor is a free add-on created by Microsoft in 2013 for Excel 2010 and 2013 and later also included in Power BI. It is the intuitive tool that allows you to transform, maintain, manipulate, consolidate, merge, enrich, and do much more with your data.

It acts like the ETL (Extract, Transform and Load) tool, extracting data from nearly any source, transforming it and then loading it elsewhere - in this case, Power BI or Excel. The Power Query Editor interface is separate from Power BI, and its window opens when you click Edit Queries. The function language M is its base, coming from the term data mashup, i.e., obtaining and combining data.

There are over 600 functions in M language that, when combined, provide comprehensive variation for data collection, processing, and joining. Its main concept is in the automation of this whole process, being necessary to connect only once with the data source and treat it so that the next updates follow the same pattern. Among the categories of functions available, we have:

Data retrieval	Date and time	Expressions
Binary	Date, time and time zone	Lists
Combination	Duration	Lines
Date	Error handling	Logic
Replacement	Record	Numerical
Type	Uri	Text
Separation	Table	Weather

## The Query Editor Interface

In Power BI, we have a separate window, exclusive for data processing. Look below at each of the regions commented on to begin familiarizing yourself with your organization.

Another way to import data into Power BI is by using the Insert Data option, located next to getting Data. In the window shown below, you can copy data from other locations and paste it in, as well as declare it (by typing it in the columns and rows). As this data has no connection to external sources, it will be static and normally only used for fast data entry purposes that need not be changed.

When connecting with data in Power Query, lines of code of language M will be automatically created with the configuration of the connection made. As a Power BI user, you do not need to know the M language to use Power Query. Its interface was developed with the end-user in mind and is filled with buttons that automate transformations, processing, and data creation.

## Data Processing

When we need to present a report at a meeting or do some kind of analysis, it is not uncommon to search for the data in various sources such as spreadsheets, text files, databases, or reports exported from systems. Then you have to start processing that data. When done in Excel, for example, this work is done by copying from here and pasting their several formulas in Excel to extract a part of the text, convert the data correctly, macro in VBA to remove unwanted rows, join information from one report with another to this information

Instead of filtering by a manually typed text, we will choose the filter by a parameter. Note that we could have created a new parameter in this window, too, by choosing the last option. Then, just select the parameter in the list next to it and click OK.

The column will be immediately filtered for the value that is stored in the Country parameter. Note the formula in Power Query: each [Country] = Country, i.e., each item in the Country column that matches the text entered in the Country parameter.

We can modify the current parameter by clicking on it and writing, for example, "CA" in the Current Value box. Afterward, just press Enter on the keyboard to save this new text in the parameter.

## **Data Relationship And Modeling**

Before starting the technical studies in DAX, it is important to emphasize the concept of data modeling. This type of project usually involves databases and how to delimit them, obtain them, transform them, and model them. Data modeling simplifies a natural complexity of data and big data and can be defined as a logical process of relationships between different data, enhancing its possible links to meet and respond to business needs.

Data in modeling projects are usually presented in tables. Some of them have detailed operation data, de facto calls (such as sales orders, production orders, invoices). They are the result of pro- cessation of business and usually store numbers that can be calculated and summarized into scalar values that will be distributed in the contexts of visual filters in Power BI.

The fact tables correspond to a specific event and should contain only columns related to that event. For example, sales data for a company can have the granularity daily or even temporal (with the time of purchase) and contain information about the sale, such as quantity, values, method of payment, product purchased, the customer who bought that product and the seller who sold it. It is irrelevant to insert, for example, the date of birth of the customer in this table, since here we are talking about sales facts and not customers.

Other tables in the data model will have descriptive and unique data, the so-called dimension tables. Basically, they describe why, when, where, who, what, and how the events in the tables were recorded. They are able to group several

facts into specific attributes or sub-dimensions, categorizing them and making it possible to filter through these similar descriptions among several records in the fact table. In addition to this attribute, they are essential for making the "connection" between fact tables, since they contain descriptions that may be common to them, through the relation of the similar record (in the same row). Size tables can be created from data standardization techniques, which will be explored in the next chapter.

Note that in the customer registration table above, we have the location of this customer in the columns Country, Province, and City, and the employee (seller) responsible for it. These four columns do not need to compose the sales fact table because we will have more than one record (row) of the same customer, repeating the information of the four columns several times and using hardware resources without need. Instead of having all the information in just one table, two have been created: one fact and another dimension, and both need to have a primary and foreign key to connect. In this case, the key (column) that connects the tables is the Client Code.

## **Calculations And Dax**

DAX is an expression language used to perform calculations and queries on created data models, being very similar to Excel functions, however, with a particularly different concept. DAX is an acronym for Data Analysis Expressions. Some experts have given the evolution of the MDX (Multidimensional Expressions) language.

Before we start exploring the language, let's first get to know the platforms where she is present. Besides Power BI, it's also in:

- Power Pivot is an Excel add-in released in 2010 used for data model creation, measurement creation, and table relationship. Its functionalities expand the usability of Excel for data analysis, through new self-service BI concepts introduced.
- Microsoft SQL Server Analysis Service, also known as SSAS, is a tool of SQL Server to analyze data in OLAP cubes, that is, it is capable of working in several dimensions in databases using concepts of facts, dimensions, relationships, measures, hierarchies, among others.

Created by the SSAS team of experts, with the Power Pivot, it is possible to

analyze more than 20 million lines inside Excel, something completely impossible to execute previously. And all this, without harming performance, by storing the data and calculations in computer memory. Donald Farmer (2010), one of the top product managers at the time of Power Pivot development, considers the technology to be a new type of Analysis Service designed to put the power of SSAS calculation in the hands of an Excel user.

Due to the popularity of Excel and the robustness of a solution used in SSAS, the DAX language was quickly spread and accepted after the release of Power BI. Its functions are classified into ten categories. Between them:

- Date and Time
- Time Intelligence
- Filter
- Information
- Logic
- Mathematics and Trigonometry
- Hierarchy
- Statistics
- Text
- Other

The tables are stored using xVelocity in-memory technology, also called VertiPaq, which compresses the data in the best possible compression according to its variation, redundancy, classification, and type. Some important features should be observed in Power BI:

- The tables must have different names in the same data model.
- Columns within the same table must also have different names.
- Columns in different tables may have the same names since their address will be the table name + the column name.
- Unlike the language M, in DAX, all objects are case-insensitive; that is, DATE and Date are considered equal names.

Rank Products = RANKX(ALLSELECTED(dCategoriaProduct);[Sum Sales])

The previous expression classified the categories from 1 to 9, according to their ranking based on sales. We use the ALLSELECTED function to consider the ex-suit filter next to the table.

Now let's use this ranking for the accumulated sum of each category. With the help of the CALCULATE function, we will calculate the sales according to the return table of the TOPN function.

```
Sum Products = CALCULATE([Sum Sales];
    TOPN([Rank Products]; ALL(dCategoriaProduct); [Sum Sales]; DESC
    )
)
```

## Visualization

In the previous chapters, we have taken the necessary steps to reach this moment: the visualization and the application of the data. We go through the data collection and processing, we load and model it, then we create the necessary calculations, all this process to show coherent, consistent, clean data that answer business questions. Power BI offers dozens of visuals between columns, bars, and stacked, horizontal and vertical, line, area, and combination, ribbon, cascade, and scatter graphics.

Pizza charts, thread, and treemap. Real geographical maps, choleric and area maps. Funnel, indicators, cards, tables, matrices and segmenters.

Not only those, but there is also a Marketplace with hundreds of visuals created by companies and experts and validated by Microsoft. Most are free, and some have subscription services, like one of the most famous: ArcGIS Maps. And you also have your own environment to create visuals and program yourself in the R language.

Marketplace visuals are installed per Power BI file. This means that when you open a new file, only the default looks will be displayed, and if you constantly use a custom look, you must add it again (or use Power BI template files).

## Principles For Creating Visuals And Reports

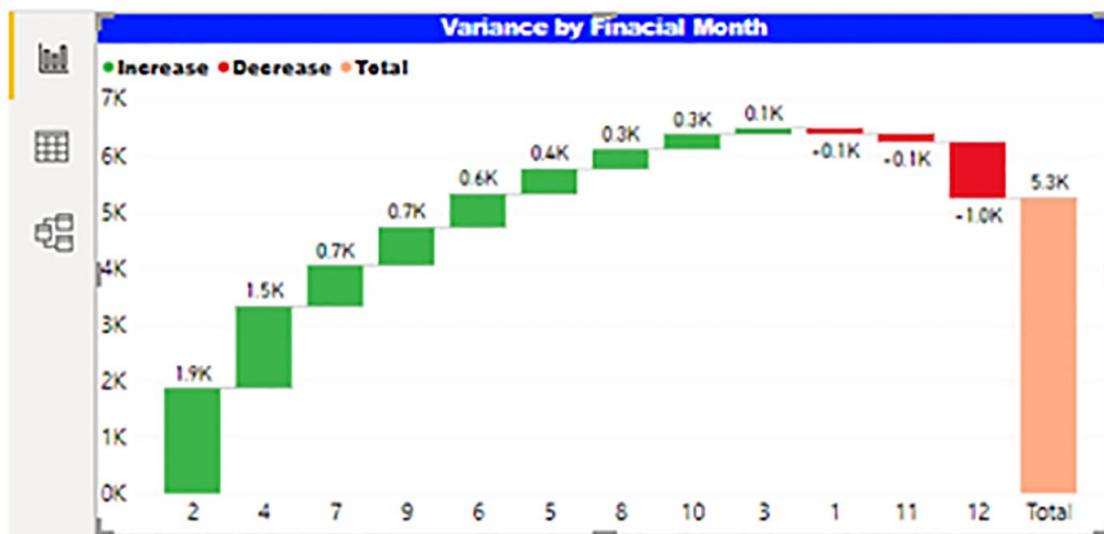
The choice of the correct look, the fields to be displayed on it, the colors, and their elements are determining factors for a data visualization project. Its main objective should be to provide the fastest and simplest reading that meets the informational needs of its customers. Some principles can be used as a guide for building your reports.

## Showing Compositions And Flows

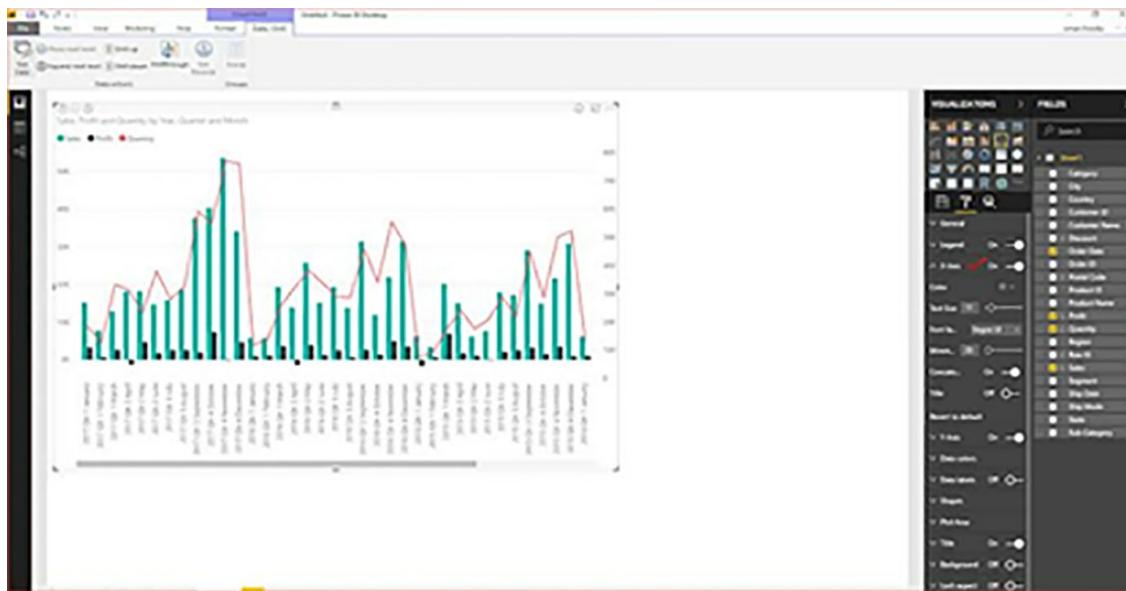
When we analyze the total of two figures, for example, sales in 2016 and 2017, we do not immediately know the influences that these figures have suffered to be positive or negative in relation to the previous one, but there are ideal visuals to avoid this composition.

The first of these is the cascade graph, which aims to show the variation between two or more points. For example, if a cascade graph shows the year 2016 and 2017 and the variation in performance over the target. We can interpret that X and Y had great positive participation, but A and B compensated negatively, resulting in 117 thousand in 2016 to 118 thousand in 2017. Looking at these two figures in isolation, we can think that there were only 1,000 sales, but in fact, there was a large internal variation in the country segment.

The waterfall graph has an interesting feature that allows you to use it for many categories. For example, if this company operated in more than 40 countries, we would have 40 columns on the chart, and this would certainly hinder its reading and interpretation.



To get around this problem, you can format the look (by clicking on the "ink roller" symbol) and choose the Split formatting to show only one or more values of those with higher representativity and still add the rest in a category called **Other**. In the following example, only two of the most representative values are shown.



After performing the previous procedure, to run an automatic filter on the tooltip according to the axis of the visual (where this tooltip will be configured), it is important to select "Keep all filters" in the panel below Filters, called Tooltip (it will only be displayed after defining the page as Tooltip).

After this procedure, go to the main report page and select the look where you want this tooltip to appear. For example, a column graph containing sales values by country was selected. In the Format, the Look panel, under Tool Tip, enable the report page and select the page name with the tooltip just below. In this example, it is called "Details." When performing this procedure, by positioning the mouse over Y, the **mini-report** will appear with the data filtered for Y.

With the panel displayed, we will navigate to the page created with the order details and click Add. The current stage of the page will be "saved," considering current filters and displays. The name of the indicator created will be "Requests."

With the indicator created, after clicking the "View Orders" button, in the Action configuration, we must inform that the type of it will be the Indicator and that the indicator will be the "Orders."

## ***Classifying By Columns***

The values in the axis of the visuals will be classified according to the standard order of the column. If it is formatted as text, it will be rated from A-Z, but if it is numeric or date, from the smallest to the largest. It is very common to have to change the classification from one text column to another than the alphabetical one. The nation of your classification can be modified in modeling.

## **Power BI**

Power BI Service is a cloud platform used to publish and share reports and dashboards, enabling you to access them over the Internet, configure your automatic update, manage shared workspaces, measure the usability of published reports and dashboards, and many other features.

Your access is via app.powerbi.com, and a user and password will be required to access your workspace. Its interface is completely different from Power BI Desktop because they are not the same version. In fact, they complement each other with several different usabilities. However, they also have similarities, and for example, you can create reports, connect to (limited) databases, and create visuals. Despite these similarities, it is not possible to perform data processing in Power Query or create measurements with DAX.

Its interface can be divided as follows:

1. **Favorites:** reports and panels can be bookmarked to be stored in this section.
2. **Recent:** displays recently used reports and panels.
3. **Applications:** shows applications (workspaces transformed into applications) created or obtained.
4. Shared with me: displays the reports and panels that are not owned by this user, but that was shared with him.
5. Workspaces: are containers that store data sets, data folders, reports, and panels.
6. The selected workspace shows the currently selected workspace and its components.
7. Notifications: displays account notifications, dataset update, configured

alerts, etc.

Reports and shared dashboards can be viewed in the application to **cellular** Power BI normally, without limitations or need for special sharing.

To view or remove the access given to users, in the same panel, click on Access and then remove the access by clicking on the options of each user who has access. This setting can also be performed by clicking on the options in this report's dataset and choosing the Manage Permissions option.

## **Sharepoint**

A third way accessible only for Pro accounts is to insert reports into SharePoint1. Its use is recommended for those who already use SharePoint in the organizational environment or will start using it for Power BI reports. To start it, the first step is to get the report link to SharePoint (different from the web publishing link), in the File menu 'Insert in SharePoint Online.'

## **Power Bi For Mobile**

Power BI has applications for Android, Windows, and iOS smartphones, and you can connect to your Power BI account (free or Pro) and consume the created or shared relations (Pro accounts only). They can be found in the official store of each platform (<https://powerbi.microsoft.com/mobile/>). With them, you can have quick access to reports, receive comparison notifications, update reports, share annotations with others, and typically use filters and pages.

Annotations can be made on the graphs, and you can create callsigns with symbols, use the free brush, and add annotations with text.

## ***Optimizing Reports For The Application***

As explained in the chapter on Power BI Service, there are two types of data collection: reports and panels. In the Power BI application, it is possible to visualize both, but the panel tends to a better visualization because it automatically organizes the visuals on top of each other, while the report displays exactly the layout developed in Power BI Desktop (horizontal). And while fully functional, its visibility can get too small for smaller screens.

## **Power BI And Excel**

One of the most used software in the world is Excel. And the reason for this is

quite clear: its resources and flexibility are extraordinary. Many analyses are performed in Excel, and if structured correctly, we encourage their use for the purposes that the software was created. We see a real movement of many analyses that were previously performed in Excel now being transferred to Power BI. We evaluate that this change occurs naturally since the language of the two software is similar, and their integration is and has been increasingly consolidated.

Power BI is a business intelligence software with all the layers that the BI process needs. Precisely because of its objective of usability, it has its business intelligence stages very well defined and with the flexibility in the necessary measure. Excel, on the other hand, is a spreadsheet. Its flexibility is one of the highest, depending a lot on its user. Both can be used to analyze data, but there are differences in productivity, storage, availability, management, and sharing.

## **Chapter One:**

### **Review Of Getting Started With Microsoft Power BI**



One of the great virtues of Power BI is the learning curve, which allows you to transform data into information almost immediately and perform very complex analyzes if you learn to master the tool.

This chapter is aimed at people who are faced for the first time with the blank Power BI canvas and overwhelmed by its many options that need a guide to get started.

The text is divided into four sections with practical guidance, and at the end of its reading, you will know how to connect Power BI with the data sources, transform the data and create a display panel.

On the Internet, there is more content on data visualization with Power BI than on the transformation and cleaning of these. For this reason, I have found it useful that the data transformation section is the most extensive and detailed.

### **Basic Theoretical Concepts**

#### ***ETL***

ETL is the acronym for extraction, transformation, and load (in English: extraction, transformation, and load), which is the process that allows us to obtain data from multiple sources to transform and clean them and then upload them to Power BI.

The ETL process is necessary so that the data we use to mount our graphics are in the right conditions (suitable formats, without errors or blank fields, etc.).

#### ***Power Bi Elements***

- Power Query: is the ETL tool integrated into Power BI.

- Power BI Desktop: the desktop version of Power BI to mount the reports.
- Power BI in the Cloud: view, share, and modify models created in Power BI Desktop.

### ***Power Bi Languages***

Language M: it is the programming language of Power Query, but do not worry, the editor that includes the program allows you to do almost everything without knowing how to program with this code.

DAX language: it is the equivalent to Excel formulas within Power BI, but much more powerful. Even if you do not master it, it is advisable to learn the use of formulas that are useful for improving your reports.

### ***Measurements***

In a very simplified way, we can say that the measurements are the calculations made using the “formulas” of Power BI (DAX language), with the advantage that once created a measure, we can use it as many times as we want in our report.

We will see it more clearly in the example.

## **Explanation Of The Practical Example Of Power Bi**

During the following sections, we will make a visualization panel using data on the World population of the World Bank.

The objective is that you learn to create a display panel from scratch while following the steps in the practical example.

At the end of the example, you will know how to do the following:

- Connect data with Power BI.
  - From Excel files.
  - From a web page.
- Transform and clean data with Power Query.
  - Rows and table headers.

- Attach tables
- Rename tables and columns.
- Enable and disable table loading.
- Remove columns
- Cancel column dynamization.
- Modify column formats.
- Merge tables
- Filter rows
- Upload the transformed data to Power BI Desktop.
- Create a panel with graphics.

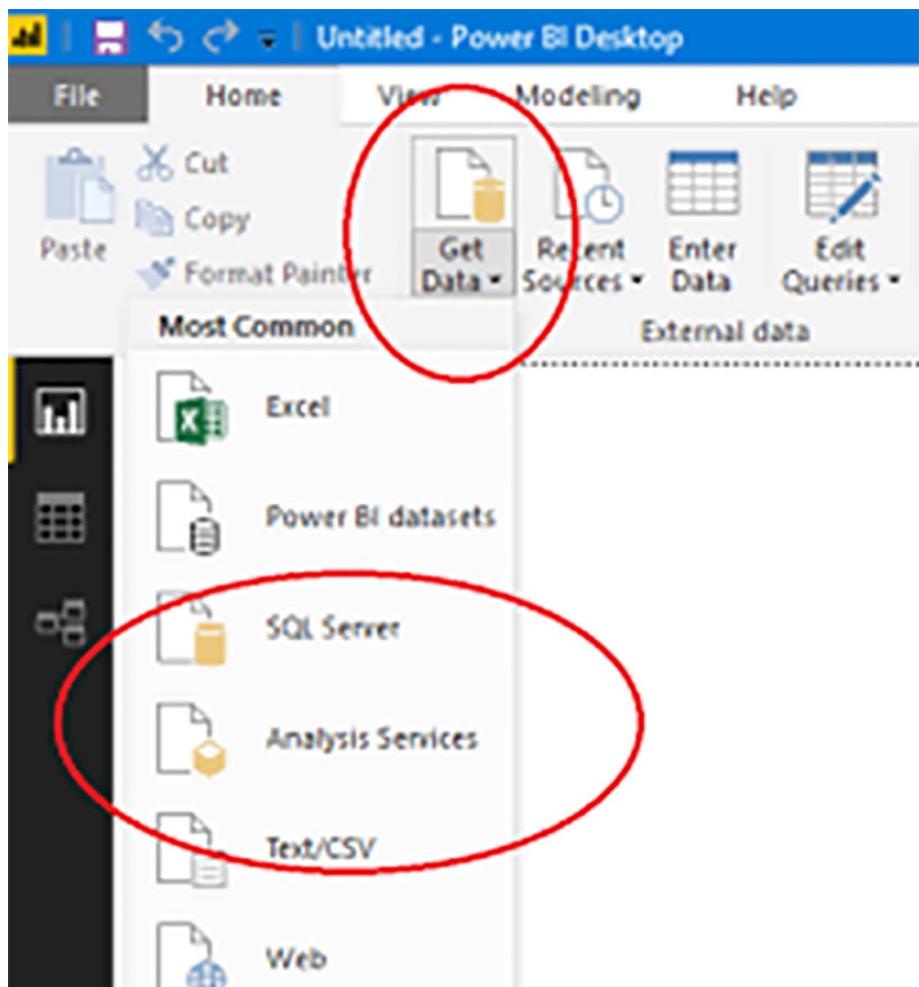
### ***Data Connection***

There are multiple options for connecting to cloud services, databases, web pages, etc. In this case, we will connect to an Excel file and a web page.

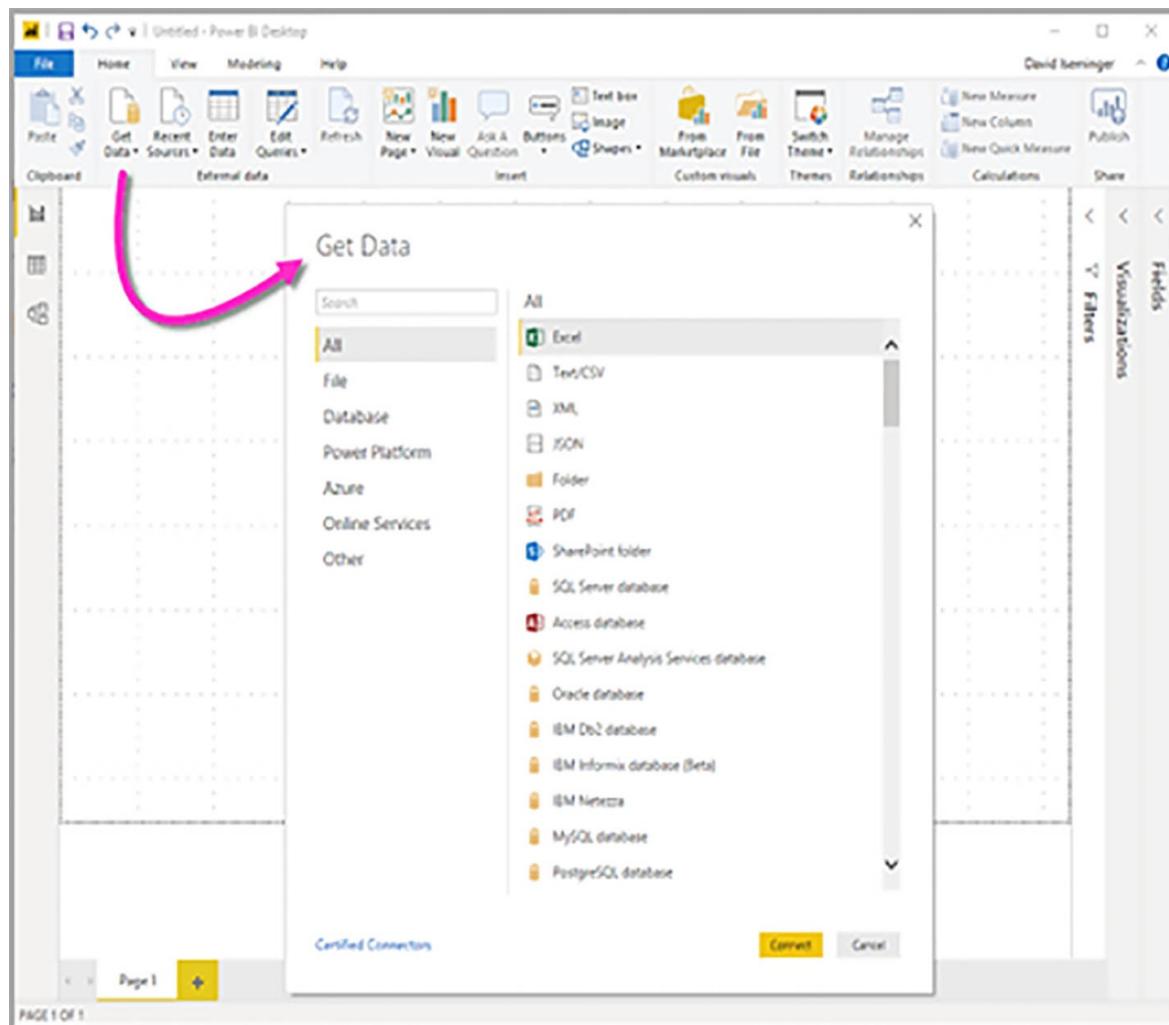
### **Connection With Excel File**

To start, we will use a file that you can download the file from the World Bank website.

Now that we have the Excel file on the computer, let's connect it with Power BI. To do this, press the «Get data» button found in the «Start» tab, as shown in the following image.

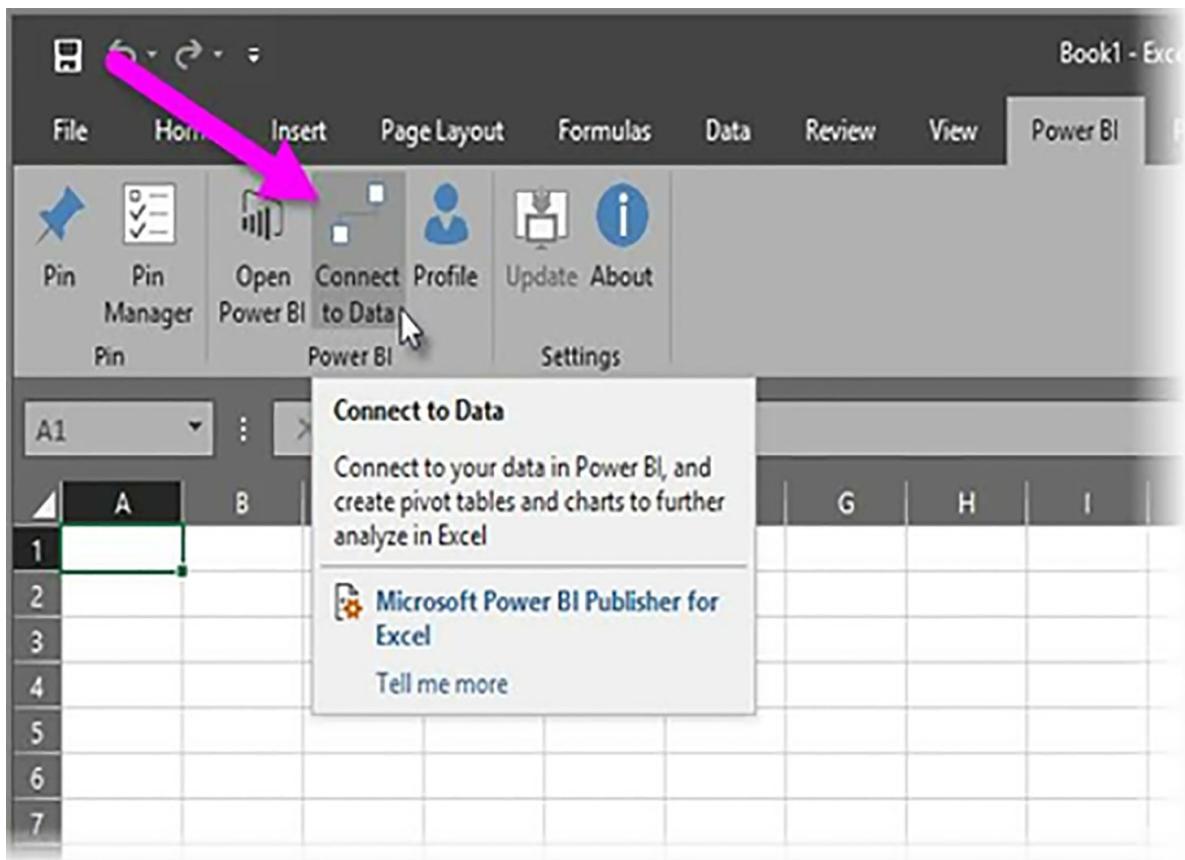


In the window that opens, you must select the Excel option and click on the "Connect" button, as we show you below.



A new window will appear to search and select the file, and you will have to search for it in the path where you have downloaded it. Once located, select it and press the open button.

Next, you must indicate to Power BI the Excel tabs that you want to connect. In this case, select the three available tabs and click on “Load.”



During the upload process, the progress will appear on a screen.

You will be able to verify that the upload was successful because the three tables appear in the Power BI field selector.

A screenshot of the Power BI field selector in Excel. On the left, the 'Get Data' pane shows a tree view of available data sources, with 'Excel' selected. On the right, the 'FIELDS' pane shows the three tables ('Sheet1', 'Sheet2', 'Sheet3') that were uploaded from Excel. The interface includes various Power BI-specific tools like 'Visualizations', 'Values', 'Filters', and 'Drillthrough'.

## **Connection To A Web Page**

Now that you have the data from the Excel file loaded in Power BI, let's get what we are missing from a web page. To do this, click on the "Get data" button on the home tab again, look for the "Web" option in the window that opens and click on "Connect."

In the new window that opens, enter the following Wikipedia address:

[https://en.wikipedia.org/wiki/ISO\\_3166-1](https://en.wikipedia.org/wiki/ISO_3166-1)

Later I will explain why we connect to this page, and the interesting thing so far is that you see how easy it is to import from different data sources.

After pasting the web address and clicking "Accept," a new window will open to select the data we want to load in Power BI.

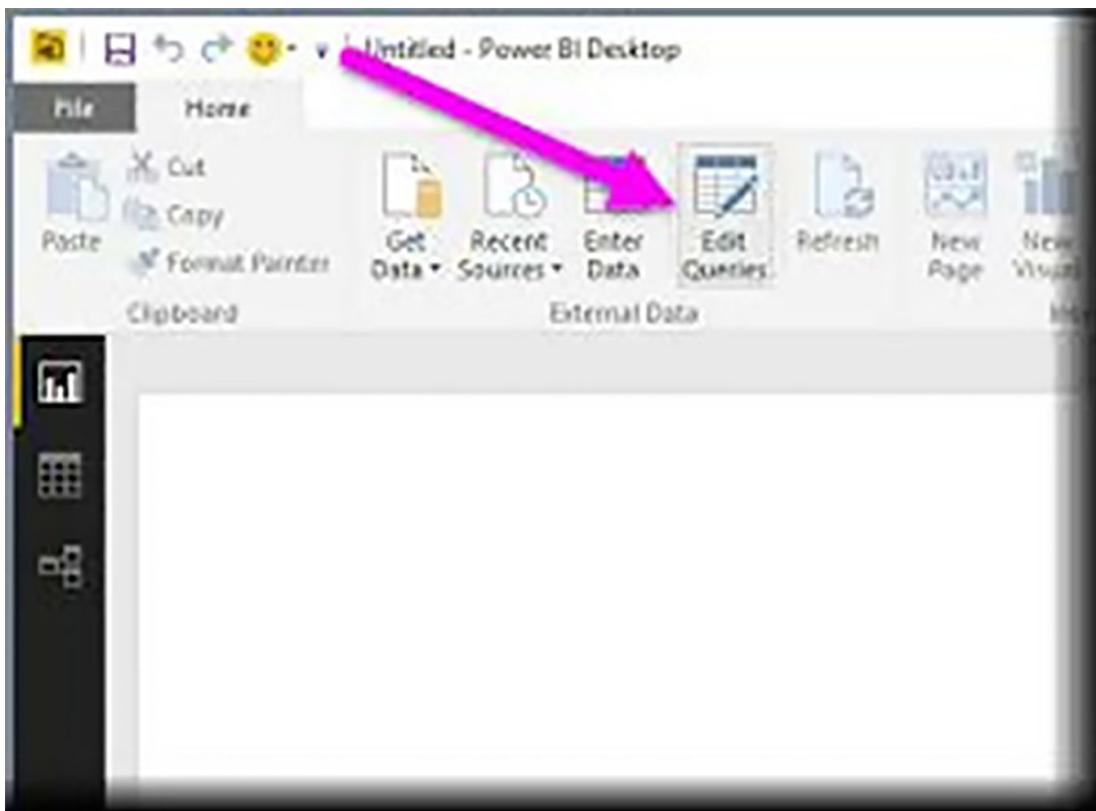
The web page is made up of multiple tables, and the one that interests us in this exercise is what is called "Officially assigned codes [edit]."

When selected, a preview of the table will appear on the screen, and you can press the "Load" button.

At this point, you already have the four tables loaded in Power BI, and you are ready to perform the data transformation.

## **Data Transformation in Power BI**

To start the transformation of the data, click on the "Edit queries" button on the home tab, as shown in the following image.



A new window will open with the Power BI query editor, which you will use to clean and transform the data.

### ***Rows And Headings***

First, select the "Active population" table and look at the rows.

- Row 1 contains the date the Excel file was updated and is not useful for our exercise.
- Row 2 contains empty cells ("null") and is not useful for exercise either.
- Row 3 contains the table headings (country code, indicator name, etc.), and you will need it.
- From row 4, there is the data, and you will also need it.

To eliminate unnecessary rows, you must press the “Remove rows” button on the “Start” tab (remember that we are working in the Power Query window) and choose the option “Remove upper rows.”

In the window that will open, we must indicate that we want to delete the top 2 rows (remember that we want to delete rows 1 and 2 that did not contain data of

interest).

You will be able to observe how the first two rows of the table have disappeared and that in the right panel, the action “Upper rows removed” has been added.

If you make a mistake by doing something, you can delete it by clicking on the X or edit it if you click on the gearwheel.

The next step is to correctly assign the headings, which are now in row 1, as can be seen in the following image.

Country Code	Indicator Name	Indicator Code	2000	2001	2002
AFG	Población activa, total	SL.TUP.TOTL.IN	Null	Null	Null
SL	Población activa, total	SL.TUP.TOTL.IN	Null	Null	Null

To do this, click on the “Use the first row as header” button on the “Start” tab.

With this step, you have already achieved that the table has the correct headings and is ready for the next transformation.

Reproduce these same steps in the “World Population” and “Urban Population” tables before continuing with the practice.

*A reminder of the steps to reproduce:*

- Delete the first two rows.
- Use the first row as a header.

## Attach Tables

The next step is to join the three tables to which you have eliminated the unnecessary rows. All three tables must have the same number of columns, and their names must match.

Click on the "Append queries" button on the "Start" tab and select the option "Attach queries to create a new one."

In the pop-up window, choose the "Three or more tables" option and select the "Active population," "World population," and "Urban population" tables by pressing the "Add" button.

When the three selected tables are in the “Tables to append” area, click on “OK.”

This action creates a new table that contains all the rows of the previous tables.

*Recommendation:*

This function is very useful if you have an Excel file for each month of the year because it allows you to put them all together in the same table.

***Renaming Tables And Columns***

To rename a table or column, double click on it. By default, the new table that we have created is called “Append1,” and we will modify it by a name that is easier to interpret and thus facilitates its identification when there are many tables.

For this example, we will rename it as “World Bank Data.”

***Enable And Disable Table Loading***

Sometimes we will not need to load all our tables to the Power BI graphics creation panel, and it is advisable to disable the loading of those that we will not use to mount graphics. For this reason, we are going to disable the “Active population,” “World population” and “Urban population” tables, since we already have all their data in “World Bank Data.”

Press the right button on each of the tables to access the menu and uncheck the option "Enable load."

A pop-up window will warn us of the risk of removing the tables from the report, but you should not worry, because we will not use that data to assemble the graphics, so you can click on “Continue.”

***Delete Columns***

Our table “World Bank Data contains the column “Indicator Code” that we will not use and, therefore, we can eliminate.

Select the column and click on the "Remove" option from the pop-up menu.

## ***Transform Columns Into Rows***

The correct way to work on the Power BI model is that the same data can only appear in one column. This rule is not met in our table, as the number of inhabitants appears in column 1960, 1961, 1962, etc.

To transform the columns into one with the year and another with the number of inhabitants you must select all the columns of dates (from 1960 to 2018) as you would in Excel, holding down the “Control” key and selecting them one by one or selecting column 1960 and holding down the “Shift” key when clicking on column 2018.

Once all the columns have been selected, you must click on the “Cancel dynamization of the columns” option of the button with the same name found on the “Transform” tab.

Our table will transform the columns into rows automatically and will have one column with the years and another with the number of inhabitants corresponding to that year.

The next step will be to rename the columns, "Attribute" as "Year" and "Value" as "No. of inhabitants" by double-clicking on the column name.

## ***Modify Column Format***

You will see that the year column is represented with the letters "ABC" and the number of inhabitants with the number "1.2". This is because a text format has been assigned by default to the first and numeric to the second.

The format can be modified by clicking on the “ABC” or “1.2” symbol, although in this example, we will not modify them.

## ***Merge Tables (Hypervitamine Search)***

If you use the Excel VLOOKUP formula, be prepared to discover a Power Query function that will be very useful.

In the table "World Bank Data" is the "Country Code" column that contains the country's code, but not their name. To add the name of the country, we will use the table “Officially assigned codes [edit]” that we have connected from Wikipedia.

First, select the “World Bank Data” table and click on the “Merge queries”

option from the “Start” menu.

Then follow these steps:

1. Select the "Country Code" column from the "World Bank Data" table.
2. Choose the “Officially assigned codes [edit] table from the drop-down menu.
3. Choose the “Alpha-3 code” column of this last table, as it corresponds to the country codes.
4. Press the "Accept" button.

After performing these actions, a new column will appear in the table with the World Bank data, and you will have to press the button with the two arrows.

In the pop-up menu, only the option "Common name" should be checked and unchecked "Use the original column name as a prefix."

Once this is done, press the "Accept" button, and you will have the name of the country in the table.

### ***Filter Rows***

Looking closely at the table, you will see that some rows do not have the name of the country. For example, rows with code "ARB" have a country name "null."

This is because “ARB” is not the code of a country and does not appear in the Wikipedia table. The Excel files with the World Bank data contain subtotals, and “ARB” is the sum of all the countries that belong to the Arab World.

As we want to work with country data, we will filter all the rows that are not. Stop it by pressing the filter button in the "Common Name" column and unchecking the "(null)" value.

This will hide rows that do not correspond to countries and will not appear on the Power BI Desktop graphics display screen.

## **Apply The Transformations Performed In Power Query**

At this point, we have completed the transformation of the data, and you can move on to the next stage by clicking on the "Close and apply" button on the "Start" tab.

# **Creating A Report With Graphics**

## **Main Sections Of Power Bi**

Power BI contains three main sections:

- The visualization of the model in which we can establish, among other things, relationships between tables.
- The visualization of the data that we have transformed and loaded from Power Query.
- And finally, the reporting area where we will paint the graphics, and that is now empty.

The options offered are very wide, and deepening them requires a lot of time, so in this section, we will focus on the creation of graphs from the data we have prepared previously.

## **Calculations With Dax (Use Calculate)**

During the transformation process, we have created a column with country data on the world population, the active population, and the urban population.

If we want to calculate the total urban population, we cannot use a formula that adds up the entire column “Number of inhabitants,” since we would also be adding the active population and the total population. To solve this situation, there is the CALCULATE function, which is one of the most useful when learning Power BI is explained in this example.

To start, right-click on the “World Bank Data” table found in the “Fields” section of Power BI Desktop and select “New measure” from the pop-up menu.

You will notice that the formula bar has been enabled to write the DAX function.

We will write a function that will add the number of inhabitants if the “Indicator Name” column is equal to the “Urban population.” This will not add the active population or the total population.

Type the following expression in the formula bar:

Urban population = CALCULATE (SUM ('World Bank Data' [Number of inhabitants]); 'World Bank Data' [Indicator Name] = »Urban population»)

This is what our function will do:

CALCULATE warns the system that we are going to perform a calculation by applying a filter (we just want to add the urban population).

SUM ('World Bank Data' [Number of inhabitants]) will add the values of the column "Number of inhabitants" in the "World Bank Data" table.

The 'World Bank Data' filter [Indicator Name] = »Urban population» forces only the values of the urban population to be added, ignoring the rest.

When closing the last parenthesis, we finish the calculation.

You can repeat this same procedure to calculate the total population and the active population. Here you can see the expressions you should write:

Active population = CALCULATE (SUM ('World Bank Data' [Number of inhabitants]); 'World Bank Data' [Indicator Name] = »Active population, total»)

Total population = CALCULATE (SUM ('World Bank Data' [Number of inhabitants]); 'World Bank Data' [Indicator Name] = »Population, total»)

At this point, you will have in the Power BI field selector the three measures that we have created marked with the icon of a calculator.

## Graphics

Now that we have the basic calculations, we are going to create the first graphics in Power BI Desktop.

Click the left mouse button on the line chart icon, and an empty graphic will appear inside the Power BI canvas.

Before continuing, be sure to select the graphic you just created and make it wider to see it better.

The next step is to take the measurements we have created to the chart. Stop it, and we must have selected the graph and click on the field of selection of the measures we want to use and the year.

They will have a graphic similar to this on the screen, and you can enlarge it by clicking on the "Focus mode" button.

The dates may appear untidy because the graph has been ordered according to the number of inhabitants and not by year.

To order it correctly, click on the "More options" button on the graph and choose the options sort by year and ascending order.

## **Dynamic Filters**

The next step is to add a filter that allows you to select countries. Click on an empty place on the canvas to make sure the graphic is not selected and then choose the "Data segmentation" display.

The empty display will appear on the canvas, and you must select the "Common Name" field from the "World Bank Data" table to transform it into a country filter.

Click on different countries to see how the graph behaves. You can also select several countries at once if you press and hold the keyboard control key.

## ***Add Visualizations From The Power Bi Marketplace***

As the list of countries is very long, we are going to add an object that helps us find a particular country.

Click on the button with the three points in the visualization box and select the option "Import from Marketplace."

Choose the "Filters" option from the pop-up window and add the "Text Filter" object.

We already have the object in our Power BI Desktop and can use it to create a country search engine.

Place the visual object on the canvas and add the "Common name" field as you have done before.

Now you have a search engine where you can write directly the name of the country you want to analyze.

With what has been learned so far, you can build a graph with similar characteristics to the one that appears on the World Bank website.

### ***Add Measurements To A Chart***

We will enrich our panel with information that provides value for the analysis. For this, we will create a new measure.

$$\% \text{ Active population} = \text{DIVIDE} ([\text{Active population}]; [\text{Total population}])$$

Next, bring the chart of grouped columns and lines to the canvas.

Select the graph and add the year in the "Shared axis," in "Column values" the total population and in "Line values" the% of the active population.

With this selection, you will obtain a graph in which you will be able to observe, among other things, how the world population has been growing continuously and how the crisis of 2008 caused a decrease in the% of the active population.

### ***Creating A Page In The Report***

Create a new page to place the maps by clicking on the button with the + symbol at the bottom of Power BI.

To create a map that shows us the population by countries, we will inform the Power BI of the field that contains the names of the countries.

First, double-click on the "Common Name" field and change the name to "Country" to make more sense.

### ***Transform The Data Type***

Select the country field and choose the "Country or region" option from the drop-down menu that appears when you click on the "Data category" button of the "Modeling" tab, as shown in the following image.

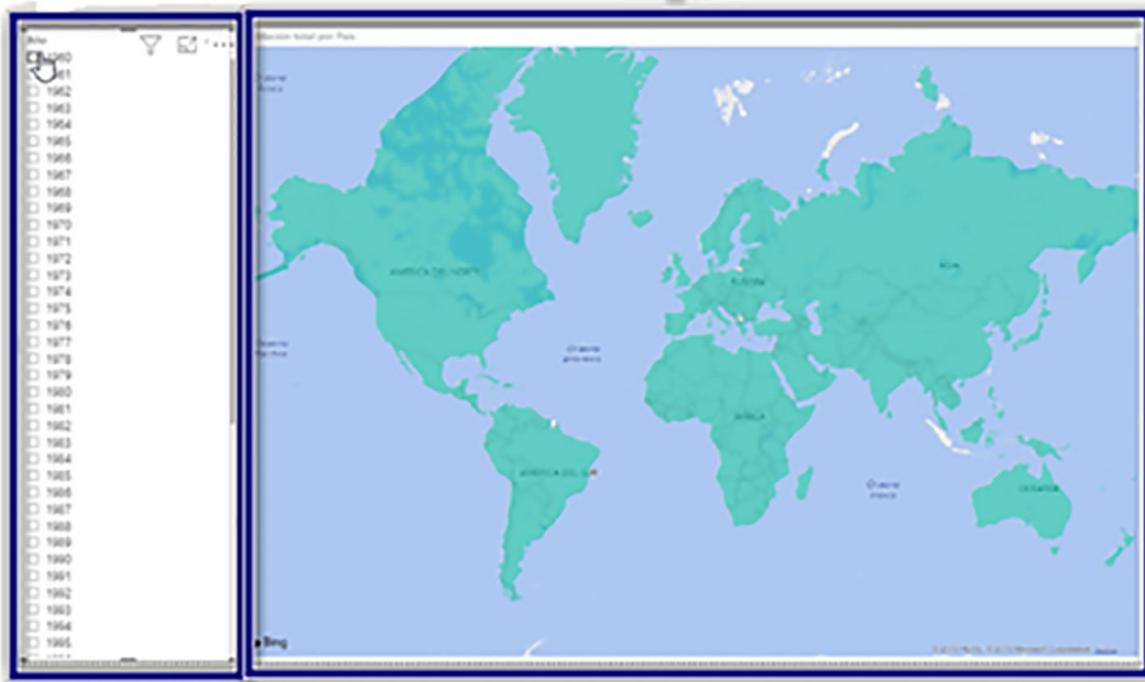
## **MAPS**

Now take the visual object "Choropletic map Map" to the canvas.

Then place the country in the field "Location" and% of the urban population in the field "Information on tools."

Then create a filter with the year field as you did before with the name of the countries.

You should have a structure similar to the one in the following image.



Finally, we will format the map, selecting it and clicking on the roll of the visualizations section (remember to select the map, or you will be formatting the filter of years).

Left-click on the symbol with three vertical dots in the “Data colors” menu and click on the “Conditional formatting” button.

In the pop-up window, configure the following fields:

- Format by: Color scale.

This way we will make the colors of the countries vary according to the number of inhabitants.

- Depending on the field:% Urban population

- Minimum: Lowest value à white

The country with the least urban population will appear on the map in white.

- Default format: Specific color to black.

Thus we will see in black the countries that do not have data in our Power BI table, and we will be able to detect possible errors.

- Maximum: Highest value to red

The country with the most urban population will appear in red.

You can now click on “Accept.”

To test the graph, select the year 2017, and you will see that there is little color contrast that facilitates the interpretation of the map.

To achieve greater visual contrast, press the conditional format found in “Data colors” again.

In the format window, check the box "Divergent," and the color scale will be modified by adding the yellow color.

Pressing on the control key and moving the mouse wheel, you can zoom in the map to enlarge specific areas.

Power BI is a very powerful tool with countless options, and this book intends to be a guide so you can get started in application management.

The consultancy Gartner carries out a classification of the main analytics and business intelligence providers every year, and Microsoft has been leading it for several years, so you can rest assured that all the effort you dedicate to learning to manage Microsoft Power BI will be well spent.

## **Chapter Two:**

### **Introduction to Power Pivot**



**P**ower Pivot is a technology that allows complex information to be processed and analyzed very efficiently and quickly. Its main feature and the one that gives it enormous speed is that it is an "In-Memory" technology, that is to say, it has all the data in memory with high compression, to decrease the space it takes up and increase the speed of analysis. These data models are based on tables and relationships.

It has three "flavors" or presentations:

1. **Power Pivot for Excel:** is an add-in for Excel that contains all this technology and allows you to manage these models from new menus integrated into Excel itself. Its use is oriented to the Personal BI.
2. **Power Pivot for SharePoint:** Allows you to manage Power Pivot worksheets on a SharePoint server. Both the data processing and the display are managed on this server. Its use is oriented to the departmental BI.
3. **SQL Server Analysis Services Tabular Model (SSAS):** these are "In-Memory" databases managed by Analysis Services through its analytical engine, a service-oriented to servers and their administration by qualified IT personnel. Its objective is to have centralized corporate information and to serve all users in the company simultaneously with very fast response times. Supports larger storage volumes, management is optimized for simultaneous use by many users. Finally, it should be noted that it allows for the automation and scheduling of the loading of information from the data sources to the model. Its use is oriented to Corporate BI.

At this point, we will focus on the Excel add-in, which allows business users to

create their own analytical models and populate them with information from a variety of data sources, independent of the IT department. In addition, a model created in Excel can be shared in SharePoint or imported from SSAS (with the help of qualified from the IT department), reusing everything that the business user has developed in Excel.

Power Pivot for Excel is an add-in for Excel that allows us to integrate and structure in tables the information from various data sources, and this information is stored in the files ".xlsx" with a high degree of compression, and at the moment they are opened they are read and loaded completely into memory, which allows for a speed of calculation not seen until now in Excel, as well as the possibility of storing large quantities of rows, many millions of rows, thus overcoming the limitations (approximately 1 million rows) and slow speeds that we have when there are a large number of rows in traditional Excel sheets.

To give you an idea of the high compression that can be achieved, let's take an example with real data, made by SolidQ. We have a model with four boards:

Clients	<b>18,848 rows</b>	<b>29 columns</b>
Date	2,191 rows	19 columns
Products	606 rows	36 columns
Sales	6,341,790 rows	26 columns

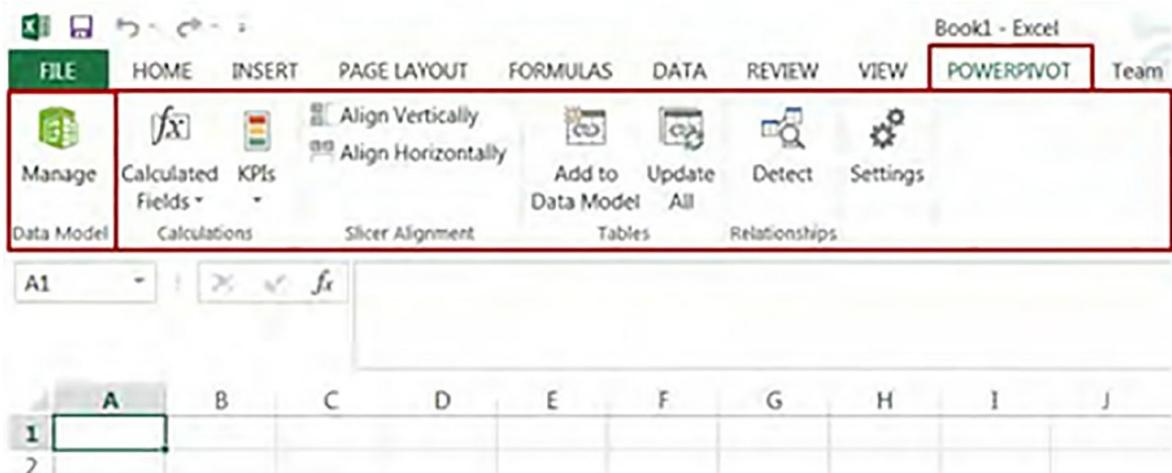
It occupies a size of only 26.3 MB. Note that the compression may be lower, as it depends on the redundancy of values in each column, and in this case, the redundancy is quite high.

In addition, it includes a powerful formula language called DAX (Data Analysis Expressions) that greatly increases the analytical calculation capacity and response speed.

Whether you are a mid-level Excel user and are getting your company data by exporting reports and queries from your applications and/or by copy and paste operations. And if you are an advanced user who discovers and struggles every day with the VBA language to try to obtain analytical solutions with some dynamism and without having to do tedious and repetitive manual tasks to obtain updated data from your company and transform it for analysis, Power Pivot is

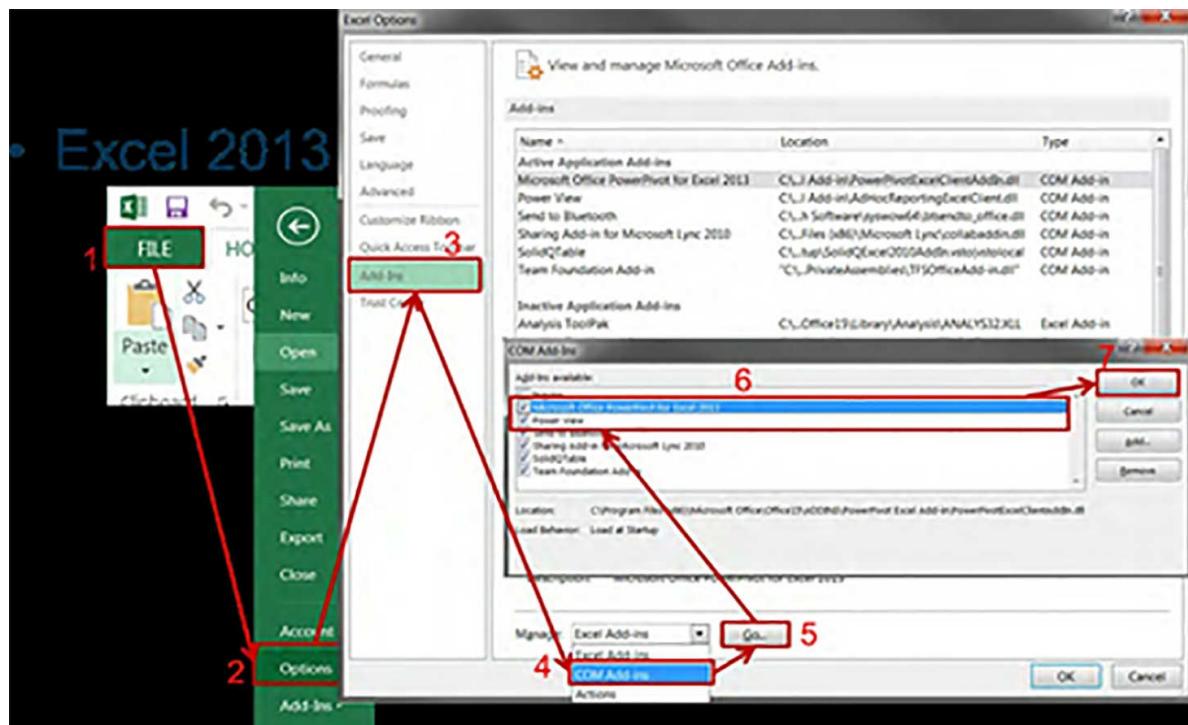
the tool you need to make your life easier, spending less time on tedious tasks and more time analyzing your information and making better decisions.

In short, more storage, easier access to external data, greatly increased calculation speed and a new formula language with much more computing power. Power Pivot is the new end-user-oriented information storage and analysis tool that is fully integrated with Excel. As you can see in the following image, its interface is a new menu in the Excel ribbon.



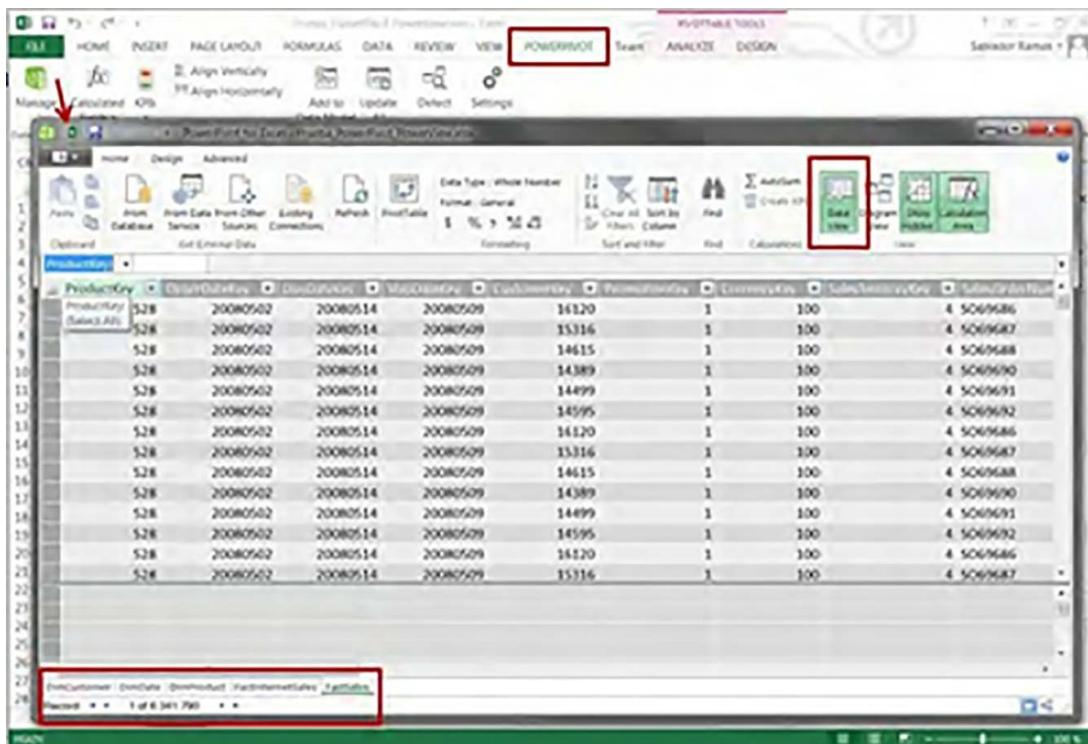
## Power Pivot Ribbon in Excel

The Power Pivot add-in is not enabled by default in Excel 2013. To enable it, you must follow a series of steps that are summarized in the following image:



## Steps to Enable Power Pivot in Excel

One of the main elements is the "Manage" option that gives access to a new window with all the necessary elements for its management. In the following image, you can see this window.



## Power Pivot Management Window

Note at the bottom left, where it indicates that there are 6,341,790 records in that table, something impossible in a traditional Excel sheet.

All the information, both from Power Pivot and traditional Excel, is stored in the same file ".xlsx". Note that Excel's autosave feature does not apply to the Power Pivot window, so you will need to save changes manually to avoid losing them in the event of an unexpected shutdown.

Finally, we will place Power Pivot along with the rest of Power BI for Excel components in the following image, to continue working on the understanding from a global perspective. Note that we have circled its location in red:

## Excel Power BI Components - Power Pivot

Power Pivot is the core of the Power BI for Excel components, without it, the existence of the rest of the components does not make sense since all the other tools need it as a starting point or destination:

1. **Power Query:** reads data from sources, transforms it and stores it in
2. Power Pivot.
3. **Power View:** is a tool to visualize information stored in

4. Power Pivot and on servers with SQL Server Analysis Services
5. **Power Map:** is a tool to visualize and analyze geolocated information in different types of maps, from information stored in Power Pivot and servers with SQL Server Analysis Services.

### **Create Analysis Models with Power Pivot**

From this moment on, already focused on Power Pivot for Excel, we are going to get to know all the details necessary for the creation of analytical models.

Power Pivot has the peculiarity, with respect to other similar tools on the market, that we do not create as such and from scratch a structure of tables and columns, with their data types and characteristics from a specific editor for this purpose. For the reader to understand what kind of tools I am referring to, I am going to put a Microsoft Access screen where the characteristics of each table in the database, and the columns that make it up, are defined visually:

The screenshot shows the Microsoft Access 'Table Tools' ribbon tab selected. Below the ribbon, the 'Design' view is open, displaying the structure of the 'Amount of Sale Export Table'. The table has nine fields: Customer, Salesman, Product, Area, Date of Sale, Amount of Sale, Comments, Mean - The Average, and Median - The middle. The 'Customer' field is currently selected. The 'Field Properties' pane on the right shows various properties for the selected field, including 'Field Size' (30), 'Format' (General), and 'Required' (Yes). A note in the pane states: 'A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.'

Field Name	Data Type	Description
Customer	Text	First and last name of the customer.
Salesman	Text	First and last name of the salesman
Product	Text	A simple description of the item sold.
Area	Text	The sales area (North, South, East or West)
Date of Sale	Date/Time	That date the sale was made
Amount of Sale	Currency	The dollar value of the sale
Comments	Memo	Salesman's notes for future leads and sales
Mean - The Average	Text	
Median - The middle	Text	

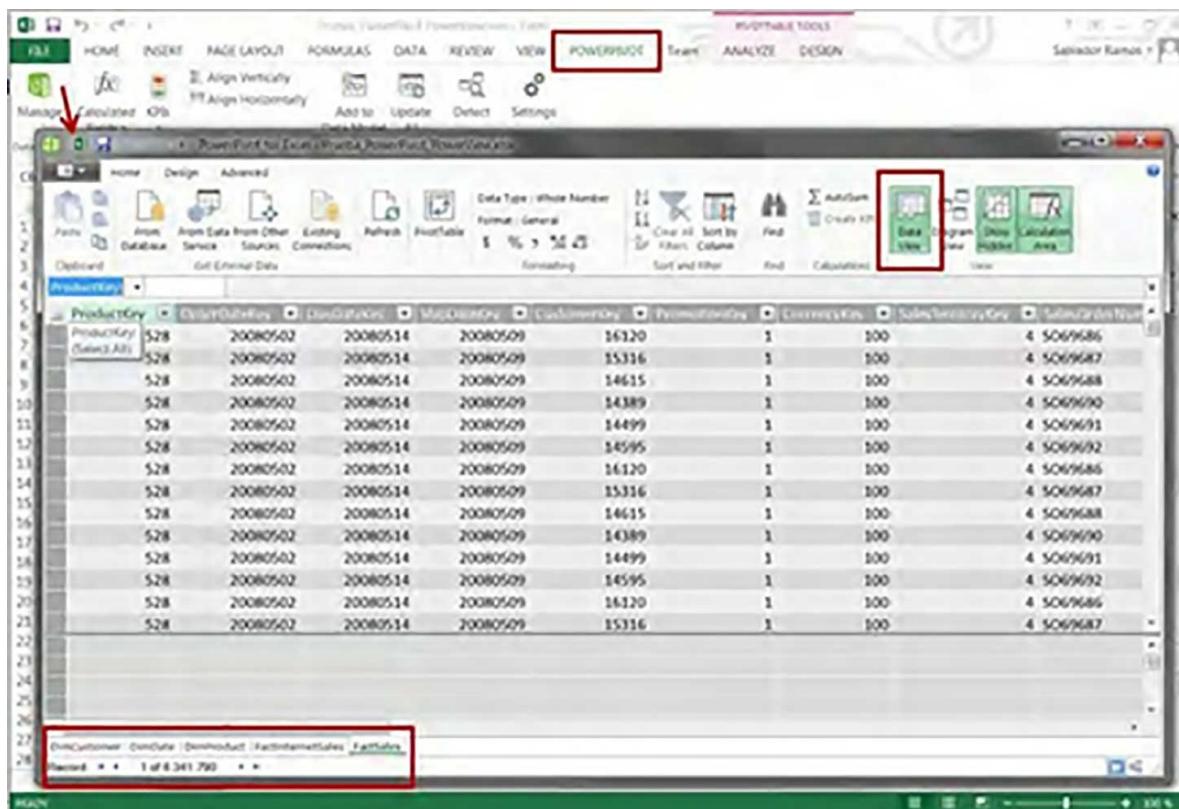
## Table design in Microsoft Access

At Power Pivot, we have two ways of generating these storage structures:

- Using Power Query, a tool that we have studied previously, which allows us to extract data from the sources, apply a series of transformations to them and save them at the destination. Well, when we indicate that the process uses Power Pivot as a destination, what happens is that a storage structure is created, as we have defined it in Power Query, and once created, it imports the data selected in that process.
- Using the options for "Get external data," you provide

Power Pivot (we haven't studied them yet, they will be studied later)

Once these structures have been created, we can access them by opening the Power Pivot administration window. To do this, we click on the "Manage" option, which is located on the left side of the "POWERPIVOT" menu, from which we have access to all the options it provides.

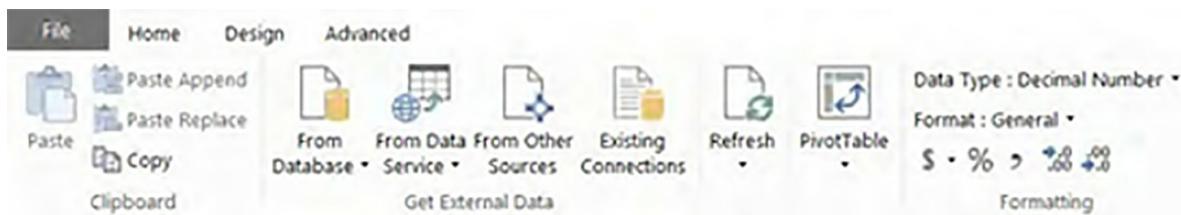


Power Pivot Management Window. Data view

We will now study each of these "ribbon" menus and each of the groups of

options and choices that make up the ribbon.

### **Home Menu**



*Ribbon, 'Home' menu*

Group of options '**Clipboard**': these are options related to obtaining data from the clipboard. It will be studied later.

Group of options '**Get External Data**': these are options related to obtaining data from various sources from the Power Pivot itself, without using Power Query. It will be studied later.

Group of options '**Refresh**': these are options related to the maintenance of data of the model, it allows to update the data of the whole model (option **Refresh All**) or only of the selected table (option **Refresh**). Completely removes the content and re-extracts it from the source, transforms it, and loads it: no incremental loading, only full loads.

Group of options '**PivotTable**': allows us to create dynamic tables and/or dynamic graphics connected to the model with a single click, opening the Excel window again and showing us the tables and graphics ready to make queries about the model. It is simply an option that improves usability and avoids us having to do all of the above step by step.

Group of options '**Formatting**': these are options relating to the consultation of the origin of data.

- **Data Type** : allows to change the data type that has been assigned to each column. It is very important to apply the right data types, it is good practice, and it helps to optimize storage and response times. The types of data available are:
  - Text
  - Date

- Decimal Number
- Whole Number
- Currency
- TRUE/FALSE
- **Format** : allows you to choose from a series of formats that will depend on the type of data in that column. For numerical data types, you can choose from the following formats:
  - General
  - Decimal Number
  - Whole Number
  - Currency
  - TRUE/FALSE

For data types Date, Date/Time, and Time you can choose from a variety of formats, more or less extensive, to represent that date/time.

- Currency symbol assignment. This and the following icons behave in the same way as in traditional Excel sheets.
- Display as a percentage, note that it multiplies the value stored in Power Pivot by 100.
- Format: format with a thousand separator.
-  The number of decimals is increased or decreased.

Let's go a little deeper into the types of data. Each column is assigned a data type, which by default is automatically assigned according to the data type of each column of the origin, but you should review it manually, to improve it if possible, and reassign a more appropriate data type if you consider it appropriate and, of course, if it is compatible and allows you to store the values obtained from the origin. By this, I mean that, for example, if you have a column with customer names, you can't turn it into a type of numerical data. These data type changes have to be done manually in Power Pivot once the table and its columns have been created after the first import of the data (which is when the data types

are assigned).

You should take into account the following aspects in terms of data types and format:

- The type of data affects storage, while the format only affects the display. You cannot assign data types that do not allow you to store the values that the data in that column has.
- Do not assign inconsistent formats for the chosen data type, although in some cases, this is allowed. It doesn't make sense to have a type of data
- Integer' and in format to assign two decimals (although always worth ".00" is confusing).
- The format 'Currency' does not make conversions: 142.84?

Group of options ' **Sort and Filter** ': these are options related to sorting and filtering of the data.

- **A-Z** : Sort in ascending order by the selected column.
- **Z-A** : Sort down the column by choosing
- **Clear Sort** : removes any sorting that we have enabled with the previous options.
- **Clear All Filters** : eliminates the established filters, leaving all the rows of the table visible.
- **Sort by Column** : allows sorting the data of a column by the values of another different column. Although at first sight, it may seem very strange, it is actually very useful, let's see an example of use: we have a column with the name of the month, in ascending order, so the first month would be April and the second would be August (alphabetical order of the names of the months), but we need the chronological order and the first month to be January, the second February, and so on. How do we solve it? So we should have a column 'MonthName' with the names of the months and another 'Month' with the numbers of the months from 1 to 12; then we use this option and tell you to sort the 'MonthName' column by the values in the 'Month' column. **Find**' option: allows you to search the contents of the model tables. It is equivalent to

that option in Excel.

- Group of options '**Calculations**': these are options that allow us to create calculated fields quickly and KPIs. We will study them in part, dedicated to DAX.
- Group of options '**View**': gives access to the different display options:
- **Data View** shows the data view (see figure above showing Power Pivot Management Window).
- **Diagram View** : shows the diagram view (you can see it in the following image). It shows the tables and relations in the form of a diagram. This is also where the Hierarchies are managed through the icon at the top left of each table (remember that it only appears when this table is selected).
- **Show Hidden** : shows or hides the columns we have marked as "hidden in client tools." These columns are those we need internally, usually for calculations, and establishing relationships, but they do not contribute anything to the user during the analysis, but on the contrary, introduce "noise" as they are columns that he does not know and will not use.
- **Calculation Area** : shows or hides the "calculation area," which is that grid at the bottom where we define the fields calculated by DAX (we will study this area later).

### ***Creation of Hierarchies***

To create a hierarchy you must be in the diagram view and click on the specific table you want to create the hierarchy on, and you will see that it is marked in a different color and that two icons appear at the top, to the right of the table name, you must click on the icon, then you must assign a name to this hierarchy and drag columns from the same table to create the different levels we want it to have. Note that a hierarchy cannot use columns that belong to another table than the one you created.

### ***Menu 'Design'***

Group of options '**Columns**': they are options related to the treatment of the columns of a table in the Power Pivot Management window.

- **Add** : allows you to add calculated columns to a table. This option will be seen in detail when we study the DAX part.
- **Delete** : it allows you to delete columns from a table.
- **Freeze** : allows fixing columns, so that they remain visible when *scrolling* horizontally. It does not affect at all how users will see it in the pivot tables, and it only affects the display of the table in the Power Pivot Management window.
- **Width** : allows you to adjust the width of the column in pixels. It only affects the display of the table in the Power Pivot Management window.

Group of options '***Calculations***' : these are options related to the execution of calculations using DAX expressions.

- **Insert Function** : applies to the formula bar, it helps us when writing DAX expressions, allowing us to use a search form about the available functions.
- **Calculation Options** : Allows calculations to be performed automatically, i.e., every time you enter or change a DAX expression, the values are recalculated. The advantage is that you have the result immediately, but the more DAX expressions you have in the model, the slower any change is made (it recalculates all of them, not just the one you have modified). Therefore, from here, you can change the option to 'Mode Manual calculation,' so the calculations will only be made when you decide, that is, when you press the option on that same button, called 'Calculate now.' It is common practice to remove the 'Calculation automatic' while we are designing the models, thus achieving shorter waiting times when writing DAX.

Group of options '***Relationships***' : these are options related to the management of relationships between tables.

- **Create Relationship** : a relationship is established based on a column in a table that is related to another column in another table. Both have to be the same type of data. This is the form that allows us to create a relationship.
- **Manage Relationships** : allows you to create, modify, or delete

relationships.

You can also create relationships from the '**Diagram View**' by clicking on the source column of the relationship and dragging and dropping on the target column of the relationship, as shown in the image:



Releasing the mouse will create the relationship between these columns in both tables.

**Relations** - besides being between columns of different tables and having the same type of data, we must coherently create them. For example, even if the product code is numerical and the supplier code is also numerical, there is no point in establishing a relationship between the two (even if the tool allows me to), and we would get totally false and inconsistent analytical data.

Therefore, when establishing relationships, we are going to do so based on the relationships defined in our documentation of the model, which is where it is already studied in detail and set down in writing what relationships there should be between the tables of the model and through which columns they are to be carried out, all of this applying the good modeling practices studied. Here we will merely do the operational part of physically creating them in Power Pivot based on the documentation.

Next, we are going to mention a series of limitations that we have when creating these relationships, and that partly affect the design of the model, although if we follow the good modeling practices studied, we will be given this casuistry in fewer occasions:

- Only one column can be used for each table involved in the relationship. Although this aspect is reflected in the form, where you can only select one column, we want to highlight it here because other tools allow several columns from each table to be involved in the relationship.

- There can only be one active relationship between two tables. For example, if I have a sales table with three dates (sale date, shipment date, and collection date) I will have to establish a relationship for each of these columns with the table of the dimension 'Date,' but only one of them can be active, leaving the other two as inactive. An active relationship is the one Power Pivot uses by default to obtain information from two related tables. If we need to use an inactive relationship, we will have to do it using DAX expressions in which the inactive relationship to be used is expressly indicated.
- It only allows for one-to-one or one-to-many relationships.
- Relationships between two columns in the same table are not allowed (self-referenced relationships).
- Circular relations are not allowed, that is, if table1 is related to table2 and table2 to table3, table3 cannot be related again to table1.

Option '**Table Properties**': allows you to edit the table properties. As we explained before, there is no designer for this, but the structure of the table comes from the result of obtaining and transforming data from Power Query or from the options to 'obtain external data' included in Power Pivot. If we have used Power Query, in the form, you cannot make any modifications to the process made with Power Query, but you have to go to Power Query and make the appropriate changes. If the process was done with any of the options in the menu '*Get External Data*' of Power Pivot, from here, we would have access to consult and modify it.

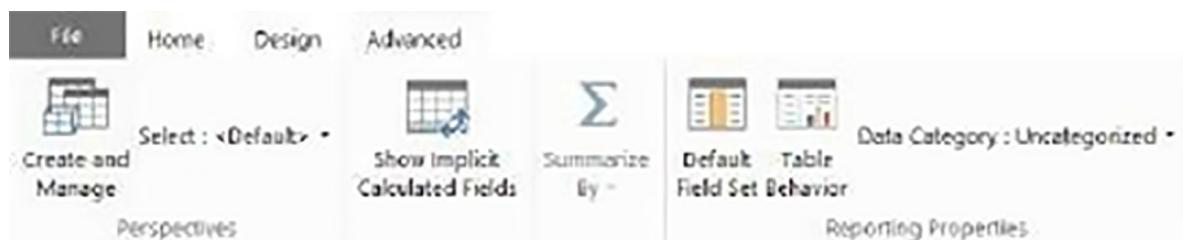
Option '**Mark as Date Table**': allows you to select which is the Date table. This table corresponds to the Date dimension, which we studied in detail previously. With this option, we must assign this table and mark which column has the data type Date. From that moment on, the time intelligence functions that are part of the DAX language are enabled. *Basically, in Power Pivot, at the model creation level, you only have to make this assignment. The bulk of the work is to include in the dimensional model a 'Date' dimension that compiles all the time treatment needs of each business and, later, to include the calculations through DAX expressions that help to enrich the analysis.*

Group of options '**Edit**': these are options to redo and undo.

- **Undo** : undo.

- **Redo** : redo.

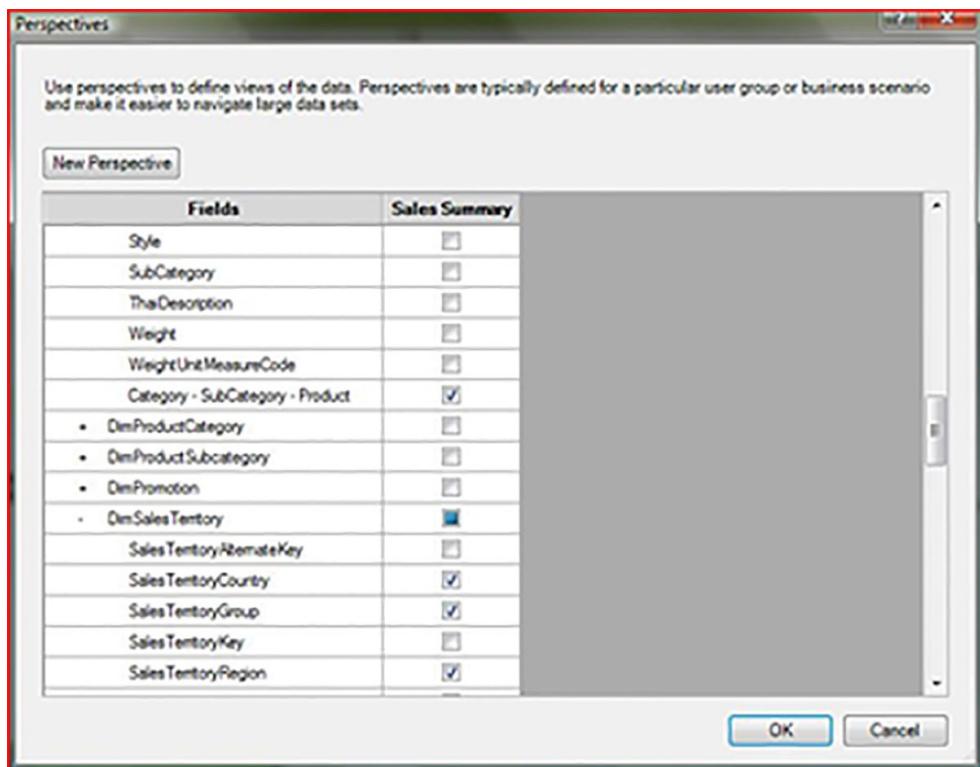
### **'Advanced' Menu**



*Ribbon, 'Advanced' menu*

Group of options '**Perspectives**': these are options relating to the management of perspectives. **Perspectives** are subsets of model elements that allow for simplified navigation. Let's suppose that we have a sales model with diverse information related to the stores, but not all of it provides value for the analysis that our sales team is going to carry out, although it is useful for other departments.

Therefore we can make a personalized view that only shows the selected elements that the commercials need, avoiding them to have a hard time finding what they need among a lot of information that they will never use. In the following image, you can see that the *perspective* called 'Commercial,' where instead of having access to all the data of the stores, only the fields will be offered: Category – SubCategory - SalesTerritoryCountry, SalesTerritoryGroup, SalesTerritoryRegion:



### Selection of fields from a Perspective

- **Create and Manage** : allows you to create and manage *perspectives* .
- **Select** : allows you to select one of the existing perspectives. When you select a perspective, you only see the fields that are marked in it.
- **Show implicit Calculated Fields** : shows or hides the calculated implicit fields. These fields are those that are created when you drag a numerical column into the value zone of a pivot table or pivot chart, using the summation aggregation function by default. From their first use, they appear in the grid of the administration window automatically. The appearance of *Implicit Calculated Fields* can be avoided to a great extent by applying a good practice that consists of creating all the calculated fields that we need and assigning them the appropriate format, and then marking the columns from which they are calculated as not visible in client tools (as they are not visible we will not be able to use them and, therefore, no implicit calculated fields will be created from them).
- **Summarize By** : indicates the aggregation function (sum, min, max, ...) that you will use by default when you create a field calculated implicitly

by that column.

- Default **Field Set** : it is the default set of fields that will be shown in the client tools when we directly select a table of the model, besides it will do it in the order that they appear in the box on the right side of the form. If we don't define this set of fields, it will show all the fields in that table, in the same order as they appear in it.
- **Table Behavior** : Allows you to define certain advanced visualization options in client tools. For example, we can configure the behavior of some features of this table when we use it in Power View.

As a summary we show the way we have walked so far with what we have seen on Power Pivot, to give you an idea of the road we still have to walk, and that we will cover it in the section dedicated to the DAX Language:



## Evolution and learning process of Power Pivot and DAX

All these advanced options, to understand in detail all their possibilities, need to be defined and then check their behavior in the client tools, mainly pivot tables, dynamic graphics, and Power View.

Here is a summary of the steps we recommend you follow to create an analysis model with Power Pivot:

1. To have the logical design of the dimensional model to be created. It is very important to have applied the studied good practices, and it will avoid many problems, as well as unnecessary complexities in the DAX

expressions that will be made later.

2. Use Power Query and/or Power Pivot's own data collection tools to extract the data from the sources, transform it according to the established model and save it in Power Pivot.
3. Use the design options offered by Power Pivot, applying the good design practices studied, to improve the model, i.e.:
  - Assign suitable data types and define formats that improve the display.
  - Manage the relationships between tables (based on the model documentation)
  - Create hierarchies (based on what is defined in the model documentation).
  - Make adequate time treatment (Date Table).
4. Enrich the model with DAX, creating calculated columns, calculated fields (measurements), and KPIs.

So far we have studied how to perform most of the process, we know how to create dimensional models and make the logical design, we know how to perform ETL processes (extraction, transformation, and loading) using Power Query and make the appropriate improvements to the model.

We still have to obtain external data from the Power Pivot itself and know the DAX language to enrich the model. These points will be the ones we see below.

### ***Uploading Data to Power Pivot***

At this point, the reader will wonder *why I have a menu available to obtain external data within the Power Pivot itself if I have Power Query ?*

The truth is that it is a totally logical question and that we have also asked ourselves, the main reason we found is that when Power Pivot came out in 2010, there was no Power Query (which appeared in 2014). Therefore the tool itself incorporated the possibility of obtaining external data from different sources.

Nowadays, there are a lot of Excel files that have used these options in them, and on the other hand, many users that have not installed Power Query, either

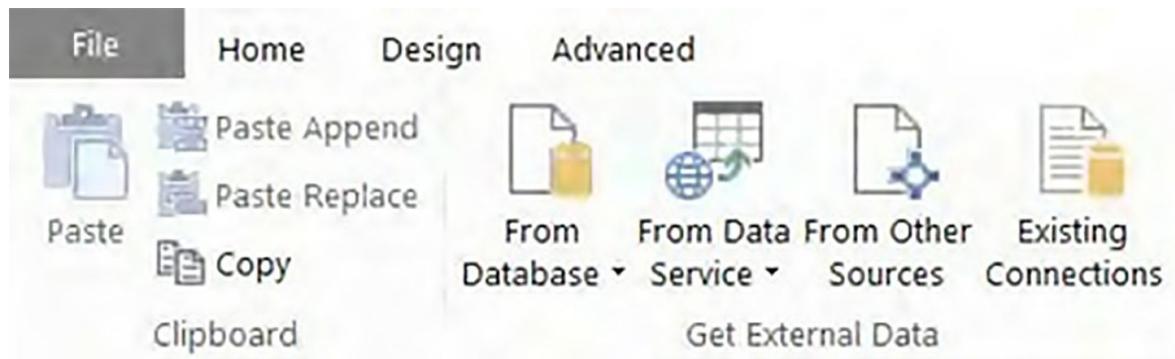
because of ignorance (you have to download and install it separately) or to avoid new learning.

Therefore, at the moment, and this is a personal opinion: *it is complicated to eliminate these options, although we believe that in the future they will be eliminated and they will be aligning everything in the same place* because basically, the functionalities available in this menu are a subset of those that are available in Power Query; with some small exceptions, there are some data sources (like Informix and some other) that are in Power Pivot and not in Power Query.

If you already know Power Query, have experience with the tool, and have it installed on the computers you work with, you do not need to use any of these features from Power Pivot. However, if you decide to use them, with the knowledge acquired about Power Query, there won't even be a learning curve, just click on the menu options and follow the instructions, you won't have any problem understanding what the tool will ask you for. The only thing you will find is that there will be fewer steps and fewer options in the process of obtaining data, which will limit you, and you will not have as much flexibility as to when you use Power Query.

There will always be occasions, in both Power Pivot and Power Query, when you need to rely on IT staff to provide you with the connection data to the corporate servers.

Although we have commented that its use is optional, we will be looking at the options available.



Ribbon, 'Home' menu, 'Clipboard,' and 'Get External Data' option groups.

Group of options '**Clipboard**': allows you to paste data tables from

The 'clipboard' is stored as a table inside the Power Pivot. It is an option that is available, and that may seem useful and comfortable for you to use, but I advise you not to use it because it has more disadvantages than advantages.

Once the table has been created, you have no chance to make changes to it (I remind you that this is a feature of Power Pivot), the way to update it is by refreshing it with the data that is currently in origin. But there is an exception, which is precisely when they come from the clipboard since the clipboard is only stored in RAM, and therefore, there is no link to the origin once the table has been created. If you want to update the table, you must copy the data back to the clipboard and use one of the following options:

- **Paste** : creates a new table from the data in the clipboard; in it, we can only choose its name and if the first row contains headers or not. Later we can make improvements just like on any other table (change column names, data types, assign formats, etc.)
- **Paste Append** : **paste** the data, but adding it and keeping the data that was already in the table. It only works on boards that were created by pasting from the clipboard.
- **Paste Replace** : **Paste** the data, leaving only the data, and removing all the data that was previously in the table. It only works on boards that were created by pasting from the clipboard.
- **Copy** : allows you to copy the selected cells to the clipboard. It works like in Excel, or like in any other application that allows copying to the clipboard.

Group of options '**Get External Data**':

- **From Database** : it has a drop-down menu with the following options:
  - **From SQL Server** : allows obtaining data **from** Microsoft SQL Server.
  - **From Access** - allows you to obtain data from Microsoft Access.
  - **From Analysis Services or PowerPivot** : allows you to obtain data from SQL Server Analysis Services (SSAS) or from SharePoint PowerPivot models. It is important to remember that

data stored in the PowerPivot cannot be obtained from another Excel file.



- **From Data Service** : has a drop-down menu with the following options:
  - **From Windows Azure Marketplace** : allows obtaining data from Microsoft SQL Server.
  - **Suggest Related Data** : Suggests **related data** sources that exist in the Azure Marketplace.
  - **From OData Data Feed** : allows obtaining data stored in Open Data sources.
- **From Other Sources** : a form appears with a variety of sources, including relational databases, Analysis Services databases, '*Data Feeds*', 'text files, etc. This includes all previously viewed sources and additional ones.
- **Existing Connections** : shows a list of connections that we have already created on our computer so that we can reuse them without having to enter all the data necessary for the connection again. This data is stored in files with the extension ".odc."

## Chapter Three:

### Introduction to DAX



#### Where Are We On The Road?

At this point, before tackling a subject, which is certainly the one with the highest degree of complexity and at the same time the one that contributes the most value to our business analysis, we would like to take a little break from the evolution of the learning that we are doing daily, getaway like a bird and take an overview of the road we have traveled and the one we have yet to travel. To do this, we will review the development cycle of a Personal BI solution:

We start by identifying a Business Process and documenting it. From there, we design the Dimensional Model that will allow us to answer the business questions. As this model is empty (it still does not contain information), we must design an ETL (Extraction, Transformation, and Loading) process that incorporates the information into the model. From this point on, and depending on the tool we are using (in our case, Power Pivot), we must make the improvements we consider appropriate to the model.

We have already studied and practiced everything mentioned above with different real case studies. So we have two more phases to discover: *Enrich the information through the DAX expression language and Perform the Visualization layer*. Once put in context, we will focus on Power Pivot, and we will also see the road we have traveled and what remains to be done.

With what we have seen so far, we have made the dimensional model, which is the basis of our entire solution, we have created the ETL process (either with Power Query or Power Pivot) that has allowed us to load the information in our storage in Power Pivot, we have been able to improve the model (data types, relationships, hierarchies, etc.) and it is time to enrich the model through DAX.

#### What is DAX?

DAX (Data Analysis Expressions) is a language of expressions that we can find inside Power Pivot, and it is based on formulas that allow defining personalized calculations. It consists of a library of functions and operators that can be combined to generate formulas and expressions. It is characterized by its simplicity of syntax and its similarity to Excel formulas and has even inherited many functions with the same name and functionality (except for small variations in a few exceptions).

## DAX Syntax

Excel's formulas are oriented to calculations *based on cells and ranges of these*, for example, in cell A21, we can write the following expression:

=SUM(C1:G1)

We also have a new syntax, incorporated in the latest versions of Excel, that applies in the case that we have formatted a set of cells as a table, which allows us to reference columns in the table using the format "[@ColumnName]." Note that in tables, you can use both the traditional cell-based syntax and the new column-based syntax.

Why are we talking about these two syntaxes here? So to facilitate the introduction to the DAX syntax, which is more similar to the syntax we just saw, based on tables, although with some peculiarities:

- It is only allowed to reference columns in tables, and *there is no cell concept*.
- It can be applied to calculated columns, calculated fields (measurements), and KPIs.
- In the case of calculated fields (measurements), the syntax includes the name of the new field.
- KPIs do not allow the use of table columns or calculated columns, and they only allow the use of calculated fields (measurements).

Calculated fields are the new name given to what used to be called Measurements as of the 2012 version. It is simply a name change; the concept, functionality, and features remain the same. Here we will use the new name and in parentheses the old one, as you should know both, and you will find a lot of documentation on the internet using the term "Measures."

## In DAX: Calculated fields = Measures

Where are the DAX formulas written? Within the Power Pivot window, although it is located in a different place and with certain nuances in the syntax, depending on whether it is a calculated column or a calculated field (measurement).

### The Syntax for Calculated Columns

It is similar to sets of cells formatted as tables:

```
= "Expression."  
= FUNCTION ("Expression1" ; "Expression2" ; ... )
```

To refer in any expression to the columns of a table, we will use the format 'Table' [Column], and follow the following rules:

- Single quotes " " are optional when indicating the table, they are only mandatory if the table name contains spaces or special characters:
  - Customers' table, Customers
- If the table column used in the calculated column expression belongs to that same table, it is optional to enter the table name:
  - 'Sales Table' [SalesAmount], Sales [SalesAmount], [SalesAmount]
- Brackets are mandatory. It is the form that identifies that we are referring to a column in the table.
  - 'Sales Table' [SalesAmount], Sales [SalesAmount], [SalesAmount]

The input of the calculated column expressions is done in a text box that is right at the top of the table, very similar to where you enter formulas in Excel. Once the expression has been entered, to change the name of the calculated column we right-click on it and choose the option "Change name of column" (this is the way to do it, since the name of the calculated column is not part of the DAX expression).

When we are writing an expression for a calculated column, we always get a contextual help that makes the task easier.

## ***The Syntax for Calculated Fields (Measurements)***

Calculated field := "Expression."

Calculated field := FUNCTION("Expression1" ; "Expression2" ; ... )

The same rules as above are used for the calculated columns, except that in this case, the name of the calculated field (measurement) does form part of the DAX expression and is indicated on the left side of the " := " symbols.

The process of entering calculated fields consists of positioning yourself in any free cell of the area of the calculated fields (the grid at the bottom of the table), clicking on the DAX expression area or the "F2" key, writing the DAX expression and pressing the "Enter" key.

Once a calculated column or a calculated field (measurement) is created, it is available to be used in any DAX expression we create. In fact, *we recommend that whenever possible, you reuse them and avoid rewriting that part of the expression.*

Whether you are using Excel in English or another language, only the original English function names will be valid. For example, we will always write SUM(... AVERAGE(..., etc. Translations such as SUM(..., AVERAGE(..., etc.) will not be valid.

## ***Calculated Columns***

A *Calculated Column* is a column that is added to an existing Power Pivot table, using a DAX expression that defines it. The value of the expression used is calculated for each row of the table at the same time it is created (at model design time) and then updated when the source data is updated (by pressing the "Refresh" button).

You can create calculated columns based on other calculated fields (measurements) and other calculated columns.

The calculated columns are added to the right of the columns we have extracted from the data source and automatically assigned the names CalculatedColumn1, CalculatedColumn2, and so on. Although you can rearrange the order of the columns by clicking on them and dragging them to the desired position, you can also rename any calculated column by right-clicking on it and choosing the option "Rename column."

Its values are stored on disk and in memory in the same way as any other column in the table. Technically they are known as "materialized" or "persistent." They are designed to be static.

The calculated columns can be used later in Excel:

- Segmenters ( *Slicers* )
- Filters
- Rows
- Columns
- Values

Let's look at some examples of use:

- Calculate the year (or any other date-related calculation) in a new column=YEAR ([Date])
- Concatenate the serial number and invoice=[Serial] & "#" [Invoice]
- Calculate the Margin=[NetPrice] - [PriceCost]
- Get a column from another table=RELATED (Province [Provincial])

*It's one more technique to denormalize*

When we drag a column into the value area, something happens internally that we must understand, really what Power Pivot does is to create an "Implicit Calculated Field," this happens whenever we drag a column of a table, whether it is calculated or not, to the value area.

An **Implicit Calculated Field** is one that does not originally exist but is automatically created when you drag a column into the value area. From then on it exists as one more field which by default is assigned the aggregation function SUM and its default name is "sum of

Calculated\_Column\_Name". For example, if you have the SalesAmount column and drag it into the value area for the first time, the calculated field "Sum of SalesAmount" will be created automatically.

An **Explicit Calculated Field** is one that we create in design time in Power Pivot in the grid designed for it. Next, we will study in detail the calculated fields (measurements).

At this point, we ask ourselves the following question that has several answers, with its pros and cons: Where should I perform the calculations, with SQL when asking for information from the source, with Power Query, with calculated columns?

With SQL, we have two disadvantages, we need to know SQL language (it is evident), and in addition, the information is calculated in origin consuming resources of the server and traveling by the network once calculated. We do not recommend it in this case.

With Power Query, it is calculated during the execution of the data update process, it is done already at the destination, in the computer where it is running, and the column is stored like any other column in the source (we are not really able to distinguish when we are looking at the column information in Power Pivot if a column exists in the source or is calculated with Power Query). This is a good alternative, leaving all that calculation logic in Power Query, it has the advantage that if I need certain columns only for the calculation I can use them in the process and they don't have to get to the model in Power Pivot.

Using calculated columns, the DAX calculations are quite efficient, but on the other hand, it forces me to bring the columns involved in the calculation into the model and thus take up unnecessary space. What we can do is hide them from the user so that he does not see them.

### ***Calculated Fields (Measurements)***

*Calculated Fields*, also known as *Measures* in previous versions, are the calculations used for information analysis.

They are created when you write a DAX Provincement after you select a cell in the calculation area (this process has been seen in detail above).

Their values are not stored on disk (as is the case with the calculated columns) but are calculated by the engine when it receives the queries. Thanks to them, we will have almost infinite computing power. No, I'm not exaggerating. To make you aware from the beginning of the importance and scope of this incorporation,

I am going to quote one of the greatest gurus in the field, **Bill Jelen (Mr. Excel)** :

"The Calculated Fields (Measurements) is the best thing that has happened to Excel in the last 20 years."

We may find that formulas are added and can be used in a dynamic table or graph, and so they are, but what is really relevant is that they offer unprecedented calculation power and flexibility.

If you have already used the pivot tables and you know the formulas that can be created in them, the new message is: "you will stop using them," it is no longer necessary, you have DAX that does everything those formulas and many other things. If you have not used this feature, it is not a problem, it is not well known, and you will not need it from now on either.

### **Use of Calculated Columns vs. Calculated Fields (Measurements)**

We consider it a fundamental question to know when to write a calculated column, and when a calculated (measured) field should be written. Although there is no single answer, we will give you a series of guidelines to help you in your decision. We will show you what to use in the following cases:

#### **Use Calculated Columns when only Values from the Row where it is going to be Represented are Involved in the Calculation**

If for each row of the table, we have the number of units sold and the total amount of the row, but not the net unit price, and we want to calculate it.

Another common example is when we have a date, and we want to extract from its new columns with the year, month, quarter, day of the week, etc.

#### **Use Calculated Columns if you Need to Use them in *slicers* , Filter Area, Rows or Columns**

As we have seen above, calculated fields (measurements) can only be used in the value area (there is no way to place them in another area), while calculated columns can be used in addition to the value area, in segmenters (*slicers*), filter area, rows or columns. We don't always have this need, and calculated columns have less flexibility than calculated fields, so think carefully when making your decision whether you really need to use it beyond the value area.

#### **Use Calculated Fields (Measurements) In All Other Cases**

Calculated fields (measurements) are especially useful and flexible for performing calculations that affect sets of rows. For example, knowing the active customers (those we have sold to them over some time)

The next step is to get to know some fundamental concepts and the functions that make up the DAX language, to practice with them, and to show examples of use.

## DAX Functions

We are going to start with the study of some of the basic and most common DAX functions, and then go into more detail and additional concepts that we need to be clear about to obtain the best analytical result.

### *Legacy Excel Functions*

Let's start with the **legacy Excel functions**. If you see the name of a function that you already know from Excel, you can use it directly, since, in Power Pivot, it will have the same behavior and parameters that you have in traditional sheets. There will only be very few exceptions that we will see when the time comes to study each function in depth. Another difference, already mentioned above, is that there will be no translation but only the function with its English name.

The following image shows a list of these functions:

Date and Time	Information	Math and Trig	Math and Trig	Text
DATE	ISBLANK	ABS	ROUND	CONCATENATE
DATEVALUE	ISERROR	CEILING	ROUNDDOWN	EXACT
DAY	ISLOGICAL	EXP	ROUNDUP	FIND
EDATE	ISNONTEXT	FACT	SIGN	FIXED
EOMONTH	ISNUMBER	FLOOR	SQRT	LEFT
HOUR	ISTEXT	INT	SUM	LEN
MINUTE		LN	SUMSQ	LOWER
MONTH	Logical	LOG	TRUNC	MID
NOW	AND	LOG10		REPLACE
SECOND	IF	MOD	Statistical	REPT
TIME	IFERROR	MROUND	AVERAGE	RIGHT
TIMEVALUE	NOT	PI	AVERAGEA	SEARCH
TODAY	OR	POWER	COUNT	SUBSTITUTE
WEEKDAY	FALSE	QUOTIENT	COUNTA	TRIM
WEEKNUM	TRUE	RAND	COUNTBLANK	UPPER
YEAR			MAX	VALUE
YEARFRAC			MAXA	
			MIN	
			MINA	

### *Legacy Excel functions*

## Aggregation Functions

There is another group of functions, the **Aggregation Functions**, which come from the relationship of functions inherited from Excel and will make it easier for us to begin to understand the differences between traditional Excel and Power Pivot.

Aggregations allow you to contract, summarize, or group data. They are in charge of calculating a column from the maximum level of detail of its rows, affecting in principle all of them unless we indicate some kind of filter or descriptive breakdown.

Let's see an example, and if we want to know the total sales of the company, we will create a column calculated using the SUM() function.

So far, nothing new, we have already seen this function in previous examples, and it gives us the sum of all the values in the column "[SalesAmount]."

Let's now see what we have achieved by writing such a simple function for

visualization and use in a pivot table. Note that we have also applied to format so that it always shows two decimals and the thousands of separators (how to do this was studied earlier).

If we create a pivot table connected to our model in Power Pivot and drag "TotalSales" to the value area, it shows us the result of adding "[SalesAmount]" for all the rows of the table and already formatted.

Now we break it down into a row for each store, and we break it down again with a column for each year.

And finally, we added a filter to have only the sales of "single" customers (Marital Status=Single):

*How many DAX formulas have you had to perform to obtain all this information regarding the company's sales amount?*

JUST ONE!!! And keeping the same value in the

But, if you had used the SUM function in traditional Excel, how many times would you have had to change the cell ranges of the parameter that passes to that function to get that result? It's not even worth counting them, obviously one for each change.

That is where one of the elements that give that computing power and flexibility to perform analysis resides. But this is not the only element; later on, we will go deeper and understand why this behavior is so simple after creating a calculated (measured) field in DAX.

We will now explain the different **types of aggregations** that exist and the functions in each of them:

- **Additives** : They add without restrictions.
  - Functions: **SUM, COUNT, COUNTROWS, AVERAGE**
- **Semi-additives** : They only add-in part, let's see an example: we need to know the total number of different customers who buy in our stores, we know that in January there have been 200 customers, in February there have been 225 customers and in March 210; on the other hand, if what we want to know is the total number of different customers who buy in our stores in the first quarter, we cannot add it up as it is, because there

will be customers who have bought from us during the three months of this quarter and others who have not, therefore it will be a number between 225 and 635.

- Functions: **DISTINCTCOUNT**
- **Non-additive** : Cannot add.
  - Functions: **MIN, MAX**
- **Customized** : we decide through various DAX expressions how they should act. For example, if we want to calculate a balance sheet, there will be accounts that add and accounts that subtract.
  - Functions: there are no specific ones; we will have to create more complex DAX expressions that solve each need individually.

The following table shows the syntax of these functions and a brief description. For more details, you can see the product's help.

SUM(<Column>)	Add up all the rows in a column. = SUM([SalesAmount])
COUNT(<Column>)	Count the number of rows in a column that has numerical data or dates. = COUNT([SalesAmount])
COUNTROWS(<Table>)	Count the number of rows in a table. = COUNTROWS('Client')
	= COUNTROWS( RELATEDTABLE( Sales' ))
AVERAGE(<Column>)	It returns the average of all the numbers in a column. =

	AVERAGE([SalesAmount])
DISTINCTCOUNT(<Column>)	<p>Count the number of rows with different values in a column.</p> <p>=</p> <p>DISTINCTCOUNT(Sales [Ticket])</p>
MIN(<Column>)	<p>Returns the smallest numerical value in a column that has numerical data or dates.</p> <p>= MIN(Sales [StoreSK])</p>
MAX(<Column>)	<p>Returns the largest numerical value in a column that has numerical data or dates.</p> <p>= MAX(Sales [StoreSK])</p>

## Navigation Function Between Tables through Relationships

These functions allow us to navigate and obtain column values from various tables in the model as long as there are relationships between them.

The following table shows the syntax of these functions and a brief description. For more details, you can see the product's help.

RELATED(<Column>)	<p>It returns a related value from another table column following the M-&gt;1 relationship.</p> <p>Example: add a column calculated with the "[Population]" to the "Store" table (Many Stores -&gt; 1 Population):</p> <p>=RELATED(Geography(Population))</p>
	<p>It returns a table in a context specified by the filters indicated, following the relation 1-&gt;M.</p> <p>Example: add a column with the number of</p>

RELATED(<Table>)	customers to the table "Geography":
	=
	COUNTROWS(RELATEDTABLE('Client'))

The RELATED function can be used, for example, to un-normalized, creating calculated columns that allow columns from several related tables of the origin to be added to a single table.

We will come back to the RELATED function later, as it is used in conjunction with other functions, and we need to understand what contexts are and how they work.

# Chapter Four:

## DAX in Practice



### Understanding the Contexts

Previously we have seen that with a very simple DAX expression, such as the calculated field (measurement).

It has allowed us to analyze in a dynamic table, the total sales, the sales by store and year, the sales to single customers by store and year, ..., and thus we could have continued showing dozens of reports based on that calculated field.

But you will also sometimes find that the DAX expression you perform does not return the value you expect, much of the blame lies with the **contexts**. Therefore, we are going to study them next, since it is essential to know them and how they behave to understand any DAX function.

Contexts allow for dynamic analysis, where the results of expression vary according to the selections made in the row, column, and filter areas. We must know them and use them effectively, both avoid getting the wrong results and to generate efficient expressions.

We have three types of contexts:

- *Row Context*
- *Query Context*
- *Filter Context*

### Row Context

It applies to the elements in the row in question, the "current row." It is used, among others, in the calculated columns. It covers the row being calculated and the rows of other tables related to that row.

It behaves like the Excel formulas applied inside a table, automatically referring to values in the same row. If the table is related to others, you can access any value in those tables that are related to the current row. To access these related tables, we have the RELATED function explained above.

There are other types of cases for which DAX has a series of functions that iterate, in a kind of internal loop, the calculations on a table. For the time being, it is sufficient to understand the above and to lay down a good foundation.

## Query Context

It evaluates a dataset for a specific cell. It refers to a subset of data that is implicitly retrieved for formula and allows elements to be added to the filter, row, and column areas.

When you place a calculated field or a column with numerical values in the value area of a pivot table, the Power Pivot engine examines the column and row headers, data segments, and report filters to determine the context. Power Pivot then performs the necessary calculations to fill each cell in the pivot table. The data set that is retrieved is the context of the query for each cell.

For a calculated field (measurement), it is the one defined by the row and column, plus the filters applied from segmenters (*slicers*) and filter areas.

Let's look at a simplified sales example, with a very reduced number of rows and columns to understand it better. To analyze the sales, we have put in rows the dates, in values the calculated field "Sum of the amount" (implicit calculated field), and we have added two segmenters, one for Customer and another for Product.

In the table, you will see the cells read by the Power Pivot engine to obtain the value that appears calculated in the dynamic table on the right (light blue and white).

## Filter Context

It is applied to the row and query contexts to generate exceptions and allow us to vary the evaluated cells based on our needs. That is, it allows you to reduce or extend the row context (*Row context*) and *query context* (*Query context*) using additional filters applied in the DAX functions

There are many cases in which we must apply additional filter contexts, but to

understand this we are going to make a simple example in which we are going to calculate the sales ratio of each client concerning the total sales of the company, for this, we are going to define the following calculated fields:

TotalImport:=SUM([Amount])

TotalCompanyImport:=CALCULATE([TotalImport];ALL(Sales2))

RatioSales:=DIVIDE([TotalImport]; [TotalCompanyImport])

Note that "TotalImportEnterprise" is always 656. Later on, we will study the CALCULATE and ALL functions, as well as other functions that allow us to perform filter contexts.

As you can see, we have applied the good practices of creating fields calculated with simple calculations and reusing them. It is preferable to do this with the three calculated fields shown above than with the following expression:

RatioSales:=DIVIDE(SUM([Amount]);

CALCULATE([TotalAmount];ALL(Sales2)) )

Both for the simplicity of understanding them, and for their subsequent reuse, if we do it as we initially proposed, we will be able to reuse the calculated fields "TotalImport" and "TotalImportEnterprise" in any other calculation we need to do, and if there were any error just correct it there would be automatically corrected all the calculated fields that include them.

## DAX Functions

Once we have laid the foundations, we can learn any DAX function and apply it correctly. Here are some frequently used ones, which are also often used in combination with other existing functions.

### Aggregation Functions Ending in "X"

We have several aggregation functions ending in "X," such as SUMX, COUNTX, AVERAGEX, MINX, MAXX, etc. These functions iterate row by row on the table. We pass to it as the first parameter and do the indicated operation as the second parameter, to finally apply the indicated aggregation function to the result.

Let's look at something that seems very simple to use a priori, which is also an expression that we can use in the calculated columns (remember, row context),

but which is not allowed to be used in the calculated fields (measurements).

In the following table, we have "Quantity" and "SalesAmount," but not "NetPrice," this is not a problem is a simple division to which we will also apply rounding to four digits.

If we later need to know the total sum of "NetPrice" we can create the next calculated field:

```
TotalNetPrice:=SUM([NetPrice])
```

Now we put ourselves in the situation that we need to make that calculation in a calculated field (measurement), dispense with the calculated column and save that disk space and memory; therefore, we go to the grid and create it:

```
NetXPrice:=ROUND([SalesAmount]/[Quantity];4)
```

But we came up against the mistake:

Semantic Error: *The value of ImportSale* cannot be determined

Well, what we really want is a calculated field that gives us the sum, so we include the SUM function as follows:

```
TotalNetPrice:=  
SUM(ROUND([SalesAmount]/[Quantity];4))
```

And again we find a mistake, which is different now:

Semantic Error: The SUM function only accepts a column reference as an argument.

Can't we really make a calculated field that adds up the result of an operation? Well, we can, but not with the SUM function, for that we must use the SUMX function.

```
TotalNetPrice:=SUMX('Sales'; ROUND([AmountSales]/[Quantity];4))
```

Be very careful about making incorrect calculations, which we tend to do when we start in the DAX language. Let's see the next calculated field:

```
TotalMalPrice:=  
ROUND(SUM([AmountSales])/SUM([Quantity]); 4)
```

This second case, what it does is, first add "SalesAmount" for all the rows, then

add "Amount" for all the rows and finally divide both results.

Let us now handle the result of the three calculations on the pivot table; the calculation error made in the last case is abysmal.

This is an extreme case, but you should keep in mind that as we make more sophisticated calculated fields (measurements), you may get grand totals and subtotals that do not equal the sum or average of their detail. In these cases, the aggregation functions ending in "X" will help you.

The following table shows the syntax of these functions and a brief description. For more details, you can see the product's help.

SUMX(<Table>; <Expression>)	Returns the sum of an expression evaluated by each row of a table.
COUNTX(<Table>; <Expression>)	Returns the number of rows that meet the evaluated expression.
AVERAGEX(<Table>; <Expression>)	Calculates the average of an expression evaluated by each row in the table
MINX(<Table>; <Expression>)	Evaluates one expression per row of the table and returns the smallest numerical value.
MAXX(<Table>; <Expression>)	It evaluates one expression for each row in the table and returns the highest numerical value.

## ALL() and ALLEXCEPT() Functions

The ALL function is used to remove filters from the context. It has two syntaxes:

`ALL(<Table>)`

`ALL(<Column1>;<Column1>;...;<Column1>)`

You cannot combine both options. If a table is indicated, remove all filters from the specified table. If one or more columns are indicated, remove the filters from the specified columns, keeping the filters applied for the rest of the table columns.

This function is not used alone but is used as a parameter for other functions to change the set of results on which other calculations will be performed.

The ALLEXCEPT function is the opposite in terms of context filters to be removed, i.e., it keeps the context filters applied to the specified table or columns and removes all others. It also has two syntaxes:

`ALLEXCEPT(<Table>)`

`ALLEXCEPT(<Column1>;<Column1>;...;<Column1>)`

And on it applies everything explained for the ALL function.

Let's see an example of when to use one or the other since, in many cases, it is more a matter of writing less. If we have a table with 15 columns and we need to apply ALL on 14 of them, it is more practical to use ALLEXCEPT on that one column I do not want you to apply ALL on. The difference would be in the behavior it would have if a new column was included in the table, in the first case ALL would continue applying on the 14 columns indicated as parameters and would not apply now on 2 of them, while in the second case ALLEXCEPT would make it apply on 15 columns (the new one would be included) and would not apply on the only column we include as a parameter.

## **FILTER() Function**

The FILTER function allows us to make a filter on a table. Its syntax is:

`FILTER(<Table>;<Filter>)`

`<Table>`: is the table to be filtered. Note that it can either be a table directly or a Provincement that returns a table as a result.

`<Filter>`: It is a Boolean expression, that is, the result of evaluating it has to be *True* or *False*. For example: `[AmountSale] > 0`,

[Product]="P2", ...

This function is not used alone but is used as a parameter for other functions to change the set of results on which calculations will be performed.

The FILTER function can be used as a parameter in functions such as CALCULATE, COUNTROWS, SUMX, etc.

If you know the SQL language, you should know that it has certain similarities with the WHERE condition.

Let's see an example of use:

FILTER(Sales2;[Product]="P2")

Later on, we will see additional examples using FILTER as a parameter for other functions that we will study.

## CALCULATE() Function

The CALCULATE function evaluates an expression in a context that has been modified by the filters passed to it as parameters. Its syntax is:

CALCULATE(<Expression>;<Filter1>;<Filter2>;...;<FilterN>)

All the "Filter" parameters applied for work as if they were within a Y() (*AND()* function, that is, the expression will only apply to the rows that comply with all the filters.

As this definition can tell you little at this time, let's see with other words more understandable for any Excel user. CALCULATE is like the SUM.SI() function, but its power and flexibility has been increased almost inexhaustibly. But what does this really mean? So the function CALCULATE instead of applying only to "ADD" applies to any DAX expression we pass to it as the first parameter, and instead of the condition "YES," it applies to all the filters we pass to it as a parameter. Remember that the concept of the filter is used only to select a part of the selected values, but it is a switch to filter context, so you can select several rows that is higher than the filtered ones at a given time.

We are going to show a very basic example by emulating the function SUM.SI(), which although not very useful, will help us to understand better the function CALCULATE:

```
SumImportP2:=CALCULATE(SUM([Amount]);  
Sales2[Product]="P2")
```

```
SumImportP2:=CALCULATE(SUM([Amount]); FILTER(Sales2;  
[Product]="P2"))
```

The two expressions above are equivalent. We have used an addition expression and applied a filter so that it only adds up the amounts of the product "P2".

Let's take up the example we saw earlier when we explained the Filter Context, and now we will be able to understand it in detail:

```
TotalImport:=SUM([Amount])
```

```
TotalImportCompany:  
=CALCULATE([TotalImport];ALL(Sales2))
```

We have used an addition expression and applied a filter context, so that instead of applying the filters defined by the user in the pivot table, always apply to all the rows of the table 'Sales2', since the function ALL() here means "return all the rows of the table sales2 without filtering".

Next, we will explain the keys to the operation of CALCULATE, to do this, we must understand how the filters behave:

The "Filter" parameters change the context of the pivot table, by changing it to filter context:

- If the filter argument acts on a field that "Already" is in the pivot table, it overrides that context and activates the filter context.
- If the filter argument acts on a field that "No" is being used in the pivot table, it adds it to the filter context.

Let's look at a third example of the use of the CALCULATE function. In this case, something that has a certain complexity and poor performance in relational database systems, but here it is done very easily and very well. We need to know the accumulated sales from the beginning of time to a given date; for this, we use the following expression DAX:

```
Accumulated Sales:=CALCULATE([TotalSales]; FILTER(ALL(Date);  
Date[DateSK] <= MAX(Sales[DateSK])) )
```

## Time Intelligence Functions

Most of today's analytical systems incorporate several advanced time management features. In this case, Power Pivot and DAX will not be less, also incorporate them. This set of functions allows us to manipulate periods, which is a very common requirement in BI.

There is a simple but important requirement that we must take into account, to apply them, we need to have a column whose data type is "Date."

The time functions allow us:

- Get specific dates.
- Get a set of dates.
- Evaluate an expression over a specified period
- There's a group that returns on a particular day, some of these are:  
FIRSTDATE(DateColumn)
- LASTDATE(DateColumn)
- FIRSTNONBLANK(DateColumn, Expression)
- LASTNONBLANK(DateColumn, Expression) S
- TARTOFMONTH(DateColumn)
- ENDOFMONT(H(DateColumn) Etc.

We are not going to study them one by one since they are functions that either by their name or because they exist in Excel, are very simple to consult in the help and start using them.

There's another group that returns a set of dates, some of which are

- DATEADD(DateColumn, Interval number, interval)
- DATESBETWEEN(DateColumn, StartDate, EndDate)
- DATESINPERIOD(DateColumn, Interval number, interval)
- PARALLELPERIOD(DateColumn, Interval number, interval) Etc.

Here is an example that will help us understand the usefulness of this type of

function. It is very common that when we are working with a period, we want to see the result that was obtained in the same period of the previous year, for this, one of the functions that we can use is PARALLELPERIOD:

TotalSales:=SUM([AmountSales])

- SalesPreviousYear:=CALCULATE([TotalSales]);
- PARALLELPERIOD(Date[Date];-1;year))
- SalesPreviousQuarter:=CALCULATE([TotalSales]);
- PARALLELPERIOD(Date[Date];-1;quarter))

And finally, we'll see a third group that evaluates an expression over some time:

TOTALYTD(Expression, DateColumn, Filter) -- Year

TOTALQTD(Expression, DateColumn, Filter) -- Quarter

TOTALMTD(Expression, DateColumn, Filter) – Month Etc.

Let's see an example of the use of TOTALYTD and TOTALQTD, which allows us to make accumulated sales within the current year, initializing to zero automatically for each year.

TotalSales:=SUM([AmountSales])

TotalCurrentYear:=TOTALYTD([TotalSales]; Date[Date])

In the following image, you can see the result obtained:

## Practical Example of DAX

Here are some examples which allow us to define an analysis report of the average ticket in our company.

These are the calculations we have used:

TotalSales:=SUM(Sales [AmountSales]) T

otalQuantity:=SUM([Amount])

NoOperations:=COUNTROWS(Sales [Ticket]))

AverageTicket(Amount):=DIVIDE([TotalSales]; [NoOperations])

Average Ticket(Quantity):=DIVIDE([TotalAmount]; [No.Operations])

Accumulated Sales:=CALCULATE([TotalSales]; FILTER(ALL(Date); Date[Date] <= MAX(Date[Date])))

As you can understand, it is not enough to do the calculations in DAX. Once we get to this point, we have to be able to show them with the greatest possible visual impact. We'll go deeper into these visualization techniques later on, knowing the possibilities of the tools at our disposal and showing multiple examples of use.

## **Using DAX Studio and Excel as Power BI Measurement Verification Tools**

If we have previously worked on the development of data models using Power Pivot or SSAS Tabular when we have to do this same task, but from Power BI, we will surely notice the lack of an important feature such as the possibility of reviewing the results of the measures that we create in the model from an Excel pivot table.

Take as an example; the data model based on the Population Register outlined in chapter 7 on the design of a population data model of this book.

This chapter set out the creation of a set of measures for the model, which allowed us to carry out the population analysis by the absolute number of inhabitants and through a series of structure indicators. For the example that concerns us at the moment, let us keep the Total Population measure and the structure indicators by Age, assuming that we have not yet added them to the model.

Total Population =

COUNTROWS ( PopulationCensus )

Youth Population =

CALCULATE ( [Total Population], Age [AgeID]> = 0 && Age [AgeID] <= 14 )

Adult Population =

```
CALCULATE ( [Total Population], Age [AgeID]> = 15 && Age  
[AgeID] <= 64 )
```

Major Population =

```
CALCULATE ( [Total Population], Age [AgeID]> = 65 )
```

Let's start by adding the Total Population and Young Population measures.

Then, using the means offered by Power BI, if we wanted to verify the validity of the measurements, we would use Visual Card and Multi-row card controls, to observe the total values, or Table or Matrix controls, to analyze the numbers disaggregated by the elements of some of the search tables of the model.

However, this procedure will seem somewhat cumbersome to carry out if we already have experience in the development of multidimensional cubes or tabular models, in which, simply by opening Excel and connecting to the model, we already have a dynamic table with which to check the measures we have created.

For this reason, we propose a technique explained by [Matt Allington](#) in his blog, where in addition to Power BI, the use of Excel and DAX Studio is required, working all the elements in a coordinated way to achieve our goal.

## DAX Studio: A Really Useful Tool

If the reader usually works with tabular models, whether from SSAS Tabular, Power Pivot or Power BI, this utility can't be missing in your toolbox, and not only by the technique that we will explain below, but by a large number of features it offers, and that will surely make your work a little easier.

First, we will download DAX Studio from the [link](#) available on its website and proceed to install it on our machine.

Then we will open the Power BI file on which we are developing our data model, and in which we have already added the measures PopulationTotal and PopulationYoungBoys.

Next, we will run DAX Studio, which, as a first step, will ask us to connect to a

valid data source. In our case, having already opened a Power BI file, DAX Studio will detect this situation, offering to connect with it.

Regarding the other connection modes, if it is an SSAS Tabular data source, it is very evident, since we only have to enter the server and instance name.

Finally, the connection against a Power Pivot model is made directly in the Excel file that contains the model, from the Add-ons tab, by clicking on the DAX Studio option.

## **Power BI and the SSAS Service**

Returning to the current scenario, Power BI acts as an SSAS server, which we can check from the Windows Task Manager.

This service is connected to DAX Studio to access the contents of the data model, which allows us, among other features, to execute DAX queries, obtain information about the execution times of the queries, execute dynamic administration views (DMV - Dynamic Management Views), and a very important fact: the address of connection to the service, which is shown in the lower right part of the window.

The next step will be to open Excel and establish a connection against the Analysis Services data source represented by the Power BI file, entering in the window of the data connection wizard the address of the service obtained from DAX Studio.

Accepting the rest of the steps in this wizard will result in a dynamic table, connected to the Power BI model, from which it will be easier to review your measurements.

Let us now continue with the development of the model by adding the two measures of structure indicators pending to include: Adult Population and Major Population.

Now we will return to Excel and select the Update option, located in the Data options group, PivotTable Tools tab | Analyze, which will produce an update of the contents of the pivot table, incorporating the new measures to it.

## **Reconnect with the Data Model in a New Work Session**

It is important to mention that, after finishing our work session, closing Power BI, DAX Studio, and Excel, when reopening Power BI, a new address will be

generated for the SSAS service, which we can verify when running DAX Studio. Due to this, if we try to validate the measures of the model using the same Excel file with which we had previously worked, we will get an error.

Accepting this warning window will open a connection window in which we can re-enter the new address. After confirming this and the next step, we will reconnect successfully from Excel, and the pivot table will be operational again.

## Get Model Measurements from DAX Studio

Another small but useful trick of DAX Studio lies in the possibility of obtaining a list of the measures that make up our model, with diverse information among which is its source code. For this, in the query panel, we will write the following sentence.

```
SELECT * FROM $ SYSTEM.MDSHEMA_MEASURES
```

We can also drag and drop this dynamic management view from the DMV tab located in the left pane of the DAX Studio window.

Then we will click on the Run option (or press F5), obtaining in the results panel all the information related to the model measurements.

The screenshot shows the DAX Studio interface version 2.7.4. In the top-left corner, there's a toolbar with various icons: Run, Cancel, Clear Cache, Output, Edit, Format, Query, Find, All Queries, Plan, Traces, Scan, Cache, Server Timings, Right Layout, Bottom Layout, Internal, Server Timings, Connect, Refresh Metadata, and Connection. Below the toolbar is a dropdown menu labeled 'DMV' containing several items: MDSHEMA\_MEASUREGROUP\_DIMENSION, MDSHEMA\_MEASUREGROUPS, and MDSHEMA\_MEASURES. The 'MDSHEMA\_MEASURES' item is highlighted with a red box and has a red arrow pointing to it from the left. To the right of the dropdown is a query editor window titled 'Query1.dax'. It contains the following DAX query:

```
SELECT * FROM $ SYSTEM.MDSHEMA_MEASURES
```

Below the query editor is a results grid. The columns are: MEASURE\_UNITS, DESCRIPTION, EXPRESSION, MEASURE\_IS\_VISIBLE, and LEVELS\_LIST. The data in the grid is as follows:

MEASURE_UNITS	DESCRIPTION	EXPRESSION	MEASURE_IS_VISIBLE	LEVELS_LIST
1	COUNTROWS([PoblacionPadron])		True	
1	CALCULATE([PoblacionTotal], Edad[EdadID] >= 0 && Edad[EdadID] <= 14)		True	
1	CALCULATE([PoblacionTotal], Edad[EdadID] >= 15 && Edad[EdadID] <= 64)		True	
1	CALCULATE([PoblacionTotal], Edad[EdadID] >= 65)		True	
1			False	

The MDSHEMA\_MEASURES view returns a large number of columns, of

which we may only need some very specific ones, such as the name of the measure and the expression used to create it. Also, we also obtain a row corresponding to the default measure of the model, which we can exclude using the MEASURE\_AGGREGATOR column. Therefore, we can refine the previous query as follows.

```
SELECT  
    MEASURE_NAME, EXPRESSION  
FROM $ SYSTEM. MDSHEMA_MEASURES  
WHERE MEASURE_AGGREGATOR = 0
```

The Power BI model that we are using for our example consists of a single table of data or measurements, but if we develop more complex models, in which several measurement tables intervene, it is advisable to add the MEASUREGROUP\_NAME column to the previous query, which we reports the table name to which the measure belongs.

or weekly basis. This saves a lot of time and manual effort.

## **Chapter Five:**

### **Power BI and Power Query (M Language)**



**S**ince its entry into the arena of business intelligence tools, Power BI has rapidly climbed positions within this segment of data analysis and decision-making systems development, to position itself as one of the most interesting options that they exist today thanks to a numerous set of qualities, among which a powerful capacity for graphical representation of the resident information in the data model, as well as the integration into the Power Query product, the extraction, transformation and loading engine (ETL) also included in Excel.

Precisely, the data mentioned above cleaning and debugging processes represent one of the main pillars of the phases that make up the development of an information system, so having a powerful and versatile tool to perform these operations is essential. Power Query, thanks to its rich user interface, which allows us to carry out the most varied operations of transformation and adaptation of the source data, gives Power BI a great power in the preparation of the data that we will later model using Power Pivot (DAX), to convert them into accessible information through dashboards, reports, etc.

In the following figure, we can see a broad graphical representation of the different phases that constitute the development of an information system using Power BI.



And if this were not enough, where we cannot reach the operations available in the user interface, Power Query also offers us the possibility to schedule tasks through M, its formula language, which considerably expands the range of options at our disposal.

## **The Role of BI Self-Service in the Development of a DataWarehouse**

The term self-service BI is usually associated with the ability of certain users of a DataWarehouse to connect to the system, obtain a subset of their information and analyze it independently to the main system, creating a smaller BI structure focused on certain aspects of the main.

These are users with an advanced profile and great knowledge of the nature of the data handled by the system, which, together with the appropriate tools, can analyze the information with a different degree of precision than is normally required by the bulk of end-users.

This type of user frequently works with the developers of the DataWarehouse responsible for the design and creation of databases, cubes (multidimensional or tabular), dashboards, etc., since it provides very valuable information for the proper optimization of processes involved in its creation, that is why having quality self-service BI tools such as Power BI helps to improve the final result of

the information system.

## **Power Query much more than Self-Service BI**

Since its emergence in the BI market, Power Pivot, Power View, and ultimately, Power Query, all add-ins of Excel, have been presented as self-service tools BI, intended for advanced users mentioned above. However, the evolution and improvement in performance of this entire suite of add-ons, and the subsequent appearance of Power BI as an integrator of all of them (Power Pivot analytical engine, Power View display engine, and Power Query transformation engine) in a only product, they turn them not only into software for information systems analysts but also very useful tools for the developers of such systems.

The objective will be to approach the development of a Power BI model, focusing mainly on the extraction, transformation, and loading (ETL) operations that we will carry out through Power Query, but also paying attention to the important volume of data that we can handle through this application.

It is not about explaining Power Query in all its details but about making an introduction to it and M, its programming language, addressing some of the most common techniques in data processing, which serve the reader as an initiation, at the same time as we observe the management capacity you have working with datasets of several million records.

It is assumed that the reader has Power BI installed on your machine to follow the examples described; however, in the following [link](#), it is possible to download the installer.

## **Population Register. The Source Data Source**

As a source of the model that we are going to build, we will use imaginary data on the imaginary population registered in the Register of inhabitants of the Cities of a country.

The said registry is formed by the information related to the residents of the populations that make up the national territory, sent by the different cities to the authority, which offers an anonymous version of this information to the citizens through the so-called microdata files.

Each of these files, available in the Statistics of the Continuous Register, contains, in plain text format, information on the population registered in the

country on January 1 of each year, where each row of the file represents an individual.

As we have just said, these data are adequately anonymized to comply with the data protection regulations, and among them, we can obtain the place of residence and birth by Province, city, and country, as well as the age, sex and size of the city by the number of inhabitants.

Also complying with the requirements of the authority regarding the use of these files, we must inform the reader that the degree of accuracy of the information elaborated from these files does not depend on the INE, but in this case, the processes and operations carried out with Power BI (Power Query) as software used to perform its operation.

For the example that we are going to develop, we will use the file corresponding to the statistics data as of January 1, 2016 (for example, 46 million records.)

Once we have accessed the download page, we will click on the “Go” button to obtain the micro\_2016.txt text file with the population data. We will also click on the link Registration design and valid values of the variables, which will provide us with the file Design and values of the variables of the Census\_2016.xlsx Microdata File, containing the guide of the population file registration design, as well as the data to feed the search tables or catalogs that we will use in the model: list of Provinces, municipalities, countries, etc. From now on, we will refer to this file in an abbreviated form as an Excel source or Excel source file. Due to space issues, both files are downloaded compressed in zip format, and subsequent decompression is necessary.

## **Population File Record Design**

As we have advanced previously, each row of the population file contains a string of digits with the data of an individual. In the following example, we see a random sample of one of these rows.

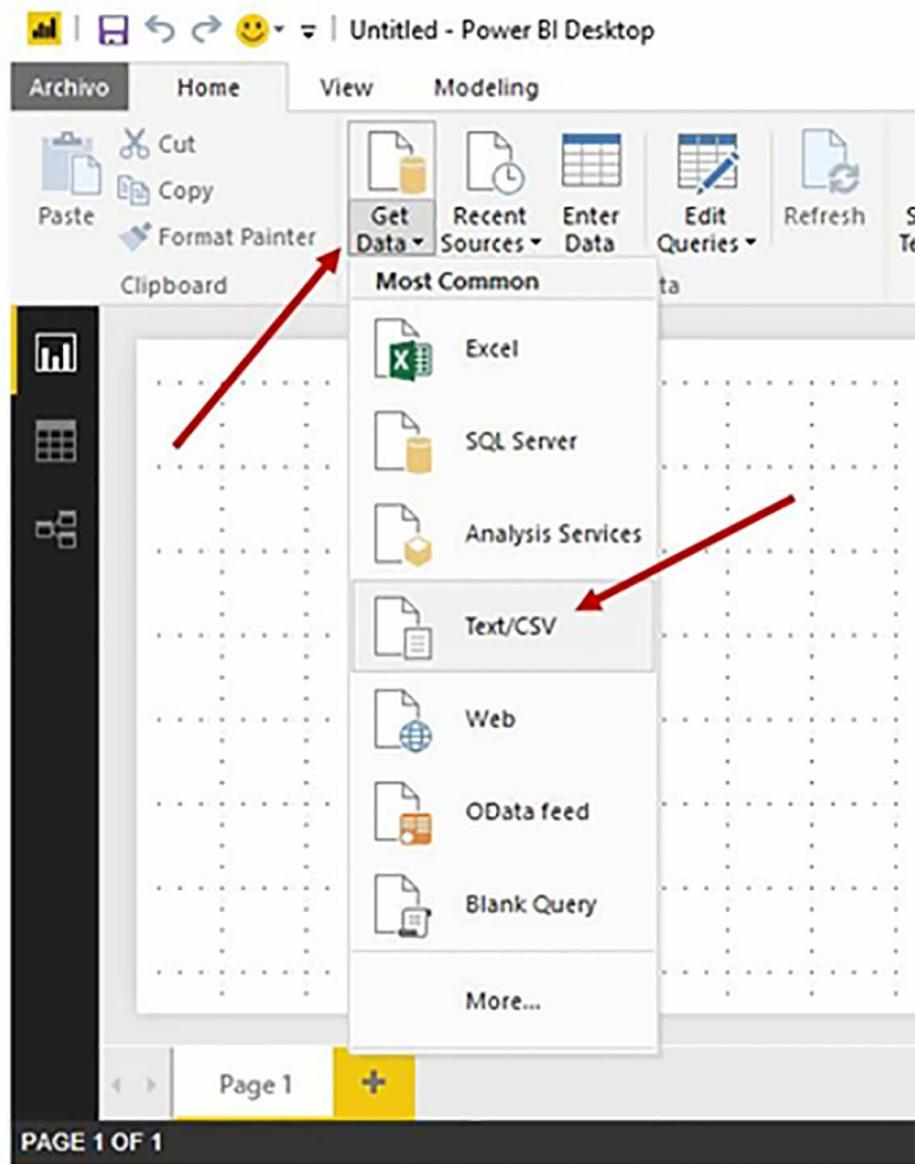
030141010021080840009

To extract each of the particles that identify the different properties of the person, we will use as a guide the Design sheet of the source Excel file, which shows us the meaning and positions that correspond to it within the total string of digits.

Using these indications, the following figure shows a brief scheme of the distribution of the values for the next process that we will implement with Power Query, to segment them so that they make sense in the subsequent modeling phase.

## Data Import with Power BI

We finally arrived at the moment to start the fieldwork. First, we will run Power BI and select the Get Data option from the External Data group, which will show a list of the most common types of data sources, from which we will choose Text / CSV.



Next, a dialog box will open in which we will navigate to the path in which we have deposited the population data file micro\_2016.txt. After selecting it, Power BI will display a preview of its first rows.



Next, we will click on Edit, and with that, Power BI will give way to the Power Query engine, represented by the query editor window or Query Editor, which will contain a query with the data of the file that we just incorporated.

## Query Editor.

### The Power Query Development Environment

Query Editor will be our main window of work throughout, and it is recommended that we familiarize ourselves with the elements.

You want Container panel of the queries used in the data transformation operations, which will later be used as a basis in the construction of the model from the Power BI designers. The term Query (query) serves the Query Editor to identify mainly tables, lists of values, or other types of objects that store data in a structured way. The work to be developed in this here will be focused mainly on the management of tables, so we will use both terms, query, and table to refer to the same type of object: a structure of data organized in rows and columns.

Data. This area shows the content of the query currently selected in the Queries panel.

Applied Steps Each of the operations we carry out on a query represents a step in the set of transformations performed, being registered with a name in this block

of the Query Settings panel. This allows us to place ourselves in any of the steps to check the status of the query at that precise moment, modify it, delete it, etc., as if it were a macro recorder.

Formula Bar. Expression in M language corresponding to the currently selected step. If this element is not visible, we will activate it by checking the Form Bar box of the Layout group on the View tab.

Operations menu It is an impressive toolbox where we will find all kinds of debugging, cleaning, transformation, etc. We need to apply to the data. Let us also keep in mind that if we do not find the solution we are looking for, the possibility of entering our own code in M language is always available.

## **Modifying the Default Names**

The file just imported generates a query called micro\_2016, which contains a column named Column1, both automatically assigned in the import operation.

Although we can use these default names in the following transformation tasks, we will change them to others that are more meaningful for the rest of the operations to be performed, since Power Query assigns names to all the elements of the transformation procedures that your engine needs have identified. Therefore it is a good practice that in those on which we have to perform some special manipulation, such as its use in a formula, we assign custom descriptors.

In the case that currently concerns us, we will right-click both the name of the query and the column, selecting the Rename option and modifying their names respectively by PopulationCensus and Individual (by pressing F2, we will also achieve the same result). PopulationCensus will represent the future table of data of the model that we are going to develop.

## **Optimization of Data Load Times**

At this point, we can update the work developed so far in the Query Editor window on the Power BI data model by clicking on the Close & Apply menu option, located in the Close group.

In our case, this operation may entail a significant process time penalty due to the high volume of data contained in PopulationCensus, since by default, all queries of the Query Editor are loaded into memory for later use in the data model of Power BI

As an alternative to be able to work with the data without actually loading it, we will right-click on the query and select the Properties option to open its properties window. Once located in it, we will uncheck the Enable load to report box and accept the changes; In this way, we maintain a reference to the data in its location of origin without incorporating them until it is necessary, thus reducing the process times. This [article](#) by Reza Rad on performance issues in Power BI provides additional information about this particular.

After executing the Close & Apply option, the Query Editor window will close, and we will return to the main Power BI window, where we will click on the File | Save to save the work developed so far in a file that we will call PopulationCensus.pbix.

## **Creation of Columns from the Source Data**

### **Preparation of the Data Table**

The next task that we will deal with will be the division into columns of the identification data of each individual from the existing column in the PopulationCensus table, following the guidelines of the Design sheet of the source Excel file.

We will start with the Province's code of residence of the person, which corresponds to the first two digits of the Individual column.

First, we will click on the header of the Individual column. Next, in the Add Column menu tab, From the Text group, we will select the Extract option to obtain a substring of the column, choosing Range as the type of extraction.

As a result of this action, a dialog box will be opened wherein the Starting Index parameter we will enter 0. This parameter represents the index of the position from which the extraction of the substring will begin. On the other hand, in the Number of Characters parameter, we will enter 2 as the number of characters to obtain.

Accepting this dialogue will create a new column with the Province's code of residence of the individual, to which we will change the name to ProvinceResidentID.

The reason for using 0 to indicate the starting position of the extraction is because Power Query works with zero-based indexes, which means that the

index number or position of the first element is 0, the index of the second is 1 and so on. The following figure shows schematically the operation that we just performed.

If, for example, we wanted to obtain, from the string in the previous figure, a substring of 7 characters starting at position 12, the values to be used would be 11 for the index of the extraction start, and 7 for the number of characters to recover. Then we show the execution scheme for this operation again.

The function of the M language that internally executes the Extract-Range menu option is `Text.Range`, whose use we will discuss in later examples. In the following block of code, we see the extraction of the previous substrings, but using this function just mentioned.

```
Text.Range ([Individual], 0.2)
```

```
Text.Range ([Individual], 12.7)
```

Taking into account the considerations that we have just explained, for those operations related to the manipulation of character strings, it will be necessary to subtract 1 from the positions of the string with which we need to work, to make them coincide with the index that Power Query internally uses in Text processing functions.

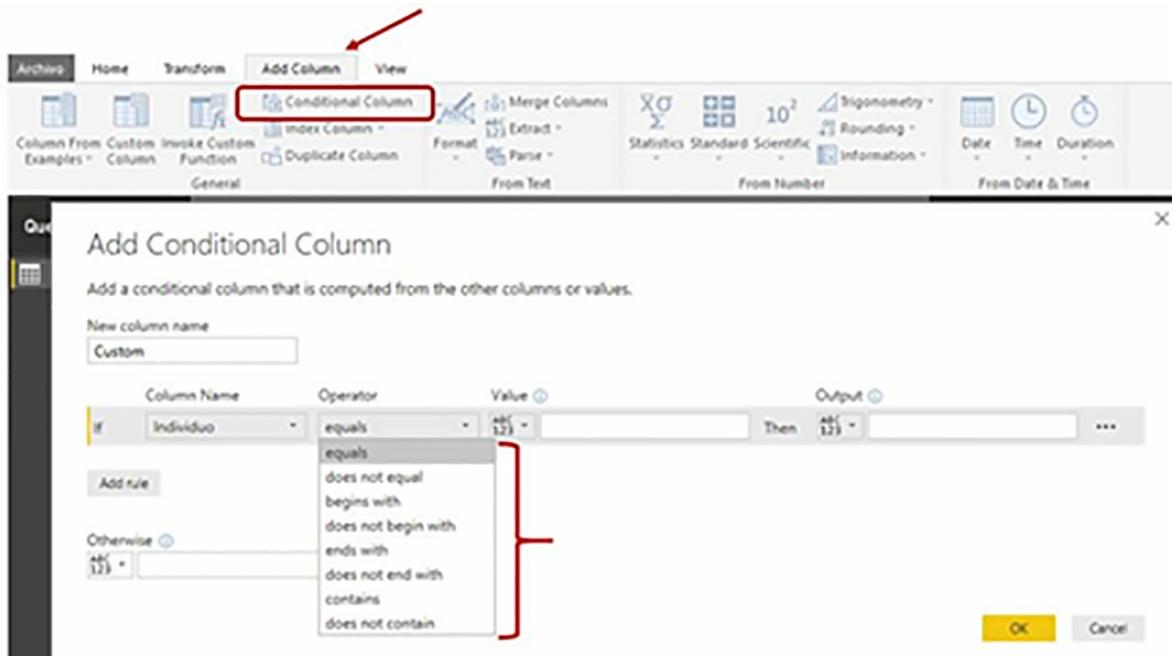
### ***Value Column Based on a Condition***

The municipality code of residence is the next column that we are going to add to the table, finding its value located in positions 3, 4, and 5 of the Individual column.

Additionally, we must bear in mind that for those municipalities with a population of fewer than 10,000 inhabitants, this value is bleached (empty chain) due to data protection issues. Since, as we will see later, the way to identify a municipality is based on the union of Province codes plus municipality, for cases where the latter is not available, we will use a code that does not previously exist as 999, more appropriate in this circumstance than an empty chain, since it will provide more consistency to the data in the phase of creation of the post-transformation model that we are currently carrying out with Power Query.

Reviewing in the Query Editor window the menu options of the Add Column tab, we will notice the existence of the Conditional Column option, through

which we can create a column based on one or more conditions. However, among the available operators, none allows us to extract a substring by positions.



This leads us to look for the solution in the Custom Column option, also located in the Add Column menu tab, which, as the name implies, enables the creation of a custom column by entering an expression written in M language.

```
if Text.Length(Text.Trim(Text.Range([Individual], 2,3))) = 0 then
```

```
    "999"
```

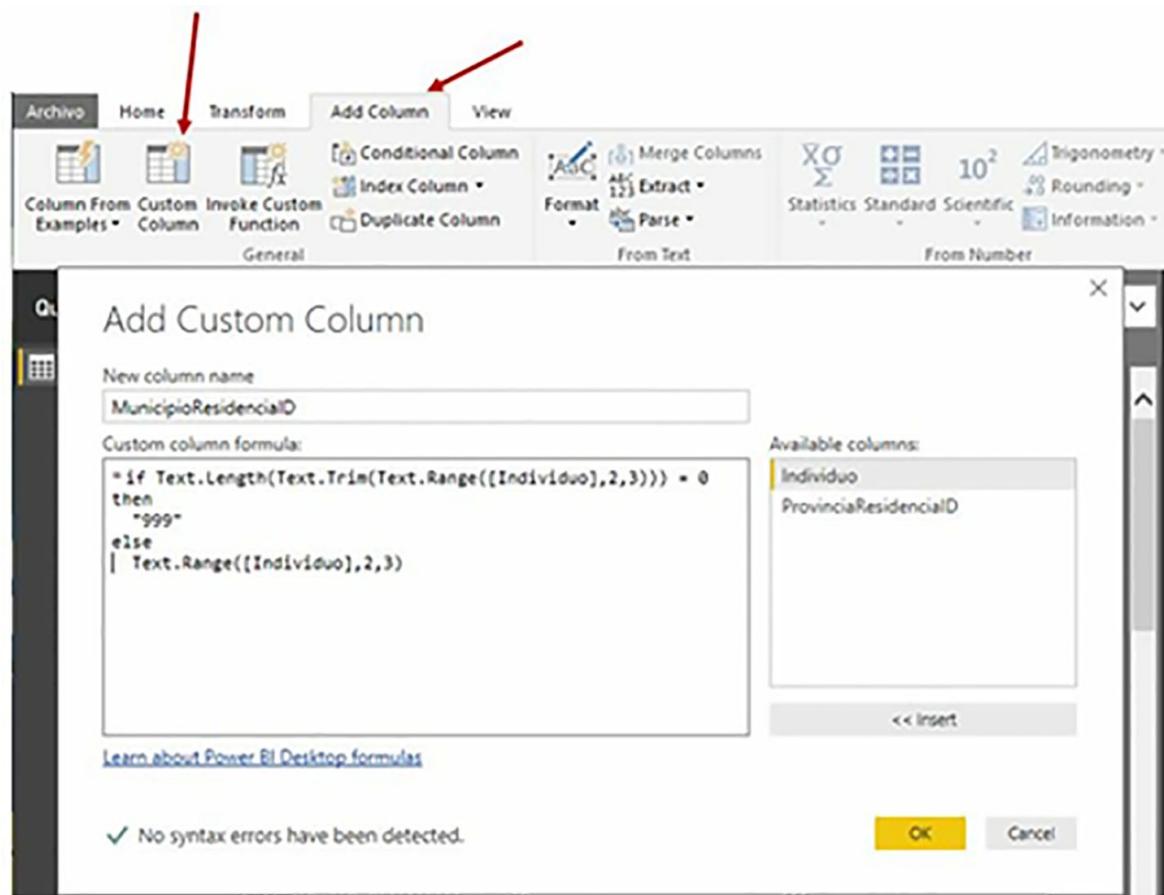
```
else
```

```
    Text.Range ([Individual], 2,3)
```

In the previous block of code, we use the decision structure if... then... else and functions belonging to the Text category to check if there is value in the 3-character substring that starts at index 2. Note that in a case like the current one, where we perform several functions in a nested way, the order of evaluation of the same begins with the one that is located in the innermost position continuing outwards, as explained below.

First, we extract the substring using the `Text.Range` function, then we eliminate the blank spaces that could be with the `Text.Trim` function, and check the resulting length with the `Text.Length` function; if this is 0 (empty string) it is one of the bleached municipalities, so we return the 999 string as a result (the

double-quote character is used as a string delimiter); otherwise, it is a municipality with value, and we return the substring corresponding to its code through `Text.Range`.



For the remaining columns to be created, we will continue to use the Custom Column menu option, since in all cases, we will need to extract a substring from the Individual column.

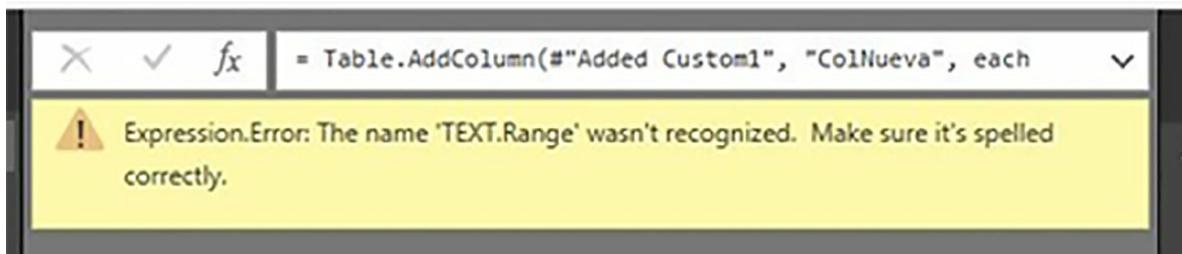
### Attention to Upper and Lower Case in M

Now that we have begun to write M code, we must be cautious when creating our expressions, due to the distinction between upper and lower case letters that this language makes between its elements, such as reserved words, function names, etc.

If for example, we try to create a new column with the following expression.

`TEXT.Range ([Individual], 2, 3)`

Having written the name of the TEXT function category in capital letters instead of Text, which would be the correct form, we would get the error we see in the following figure.



How do I find the most appropriate function for each moment? The answer is inside

M is a language composed of functions, which organized into categories (Table, Text, Number, List, etc.) offer a wide spectrum of possibilities for handling tables, lists, character strings, etc.

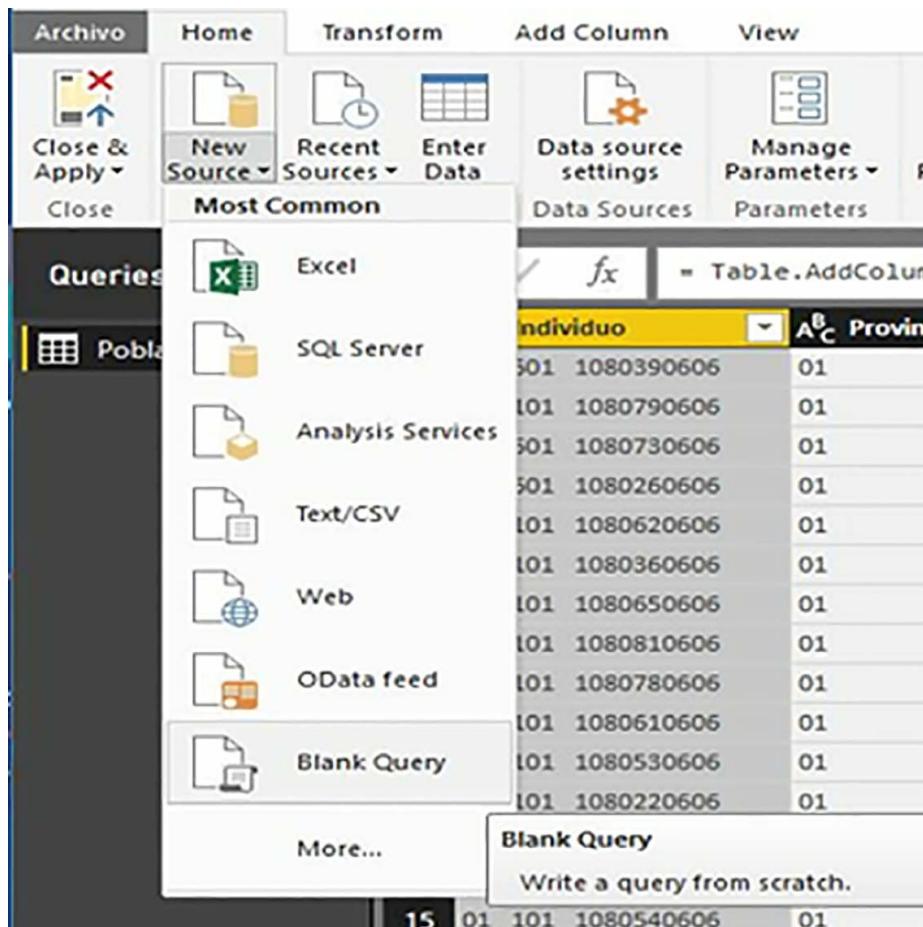
But in the face of such an offer, we need some type of documentation to find the one that best suits our requirements in each situation.

As resources on the Internet, we have the official Microsoft reference page, from which we can download a document with the language specifications. For the functions, there is also a page where they are grouped by category, and we can also find many sites dedicated not only to Power Query, but also to Power BI, Power Pivot and other Microsoft BI technologies, of which in the end some featured links are provided.

However, all of the above resources are based on our availability of an Internet connection, but what happens if, under certain circumstances, we cannot connect to the Network?

This situation should not worry us since from the Power Query itself, and we can access information about the functions of the M language through a small trick.

First, we will create a new query using the New Source | Blank Query, located in the Home menu, New Query group.



A new query with the default name Query1 will be created. In the formula bar, we will write the following expression.

```
= #shared
```

After pressing Enter, a result set consisting of records (Record objects) containing the language functions, as well as types, constant values, etc. will be generated. To be able to handle this list more comfortably, we will click on the Into Table menu option of the Convert group, belonging to the Record Tools | Convert, to convert it into a table type query.

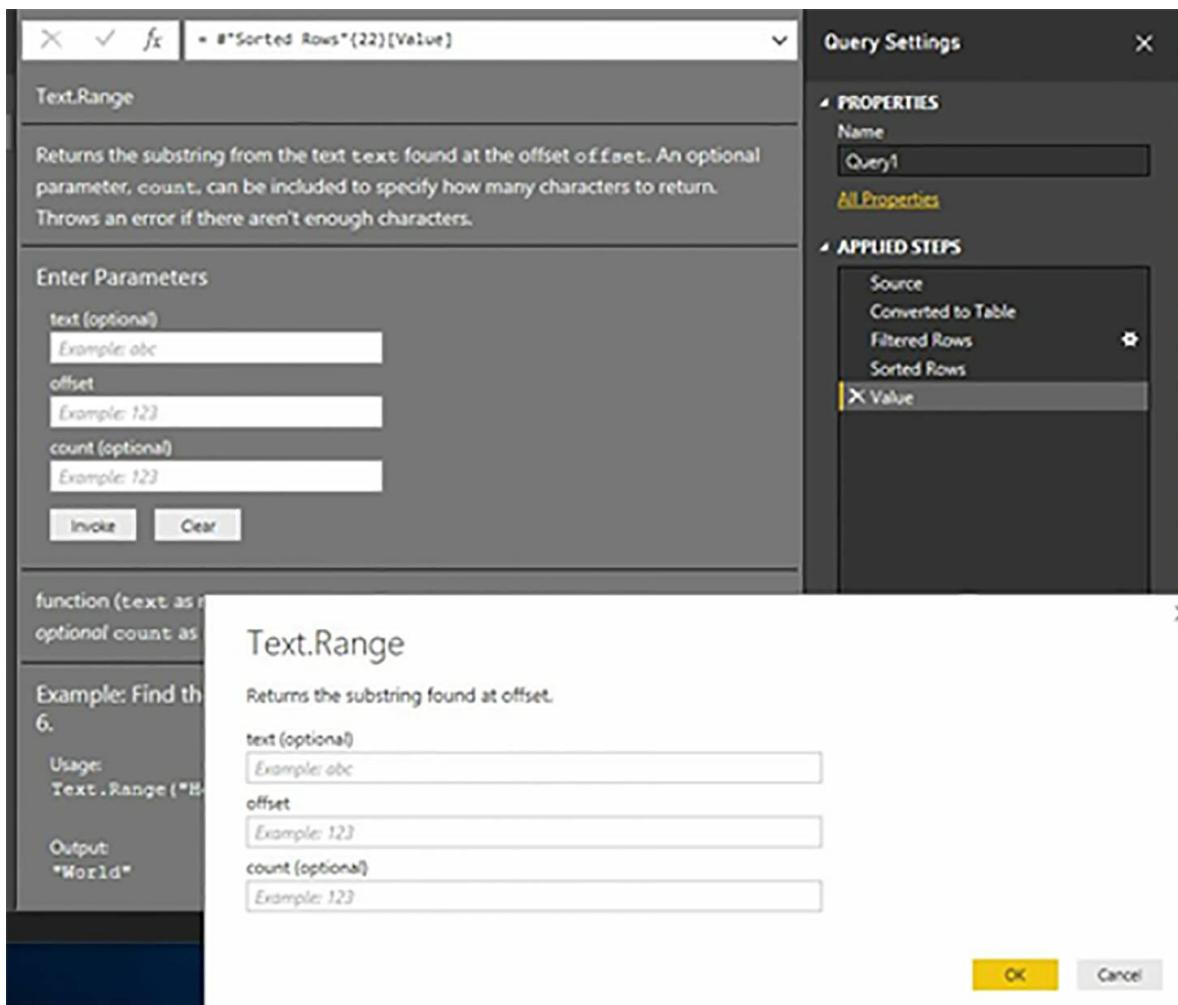
Once the table is obtained, we can find information about the functions by applying a filter on the Name column. For example, if we want to see only the functions of the Text category, we will click on the filter/order button of that column, we will enter the Text value. (including the point to narrow the search better), and we will click, OK.

As a result, the table will show the names of the filtered functions. We can

reopen this filter box and apply it for ascending order.

But if the aid remained only in a list of functions, this utility would be impractical. We are going to look for in this list one of the functions that we have previously used: Text.Range, and in the Value column, we will click but not on the Function value but in an empty space within that box. A panel will open at the bottom of the window with detailed information about the selected function: signature (parameters and return with their corresponding types), description of the operation performed, and examples of use.

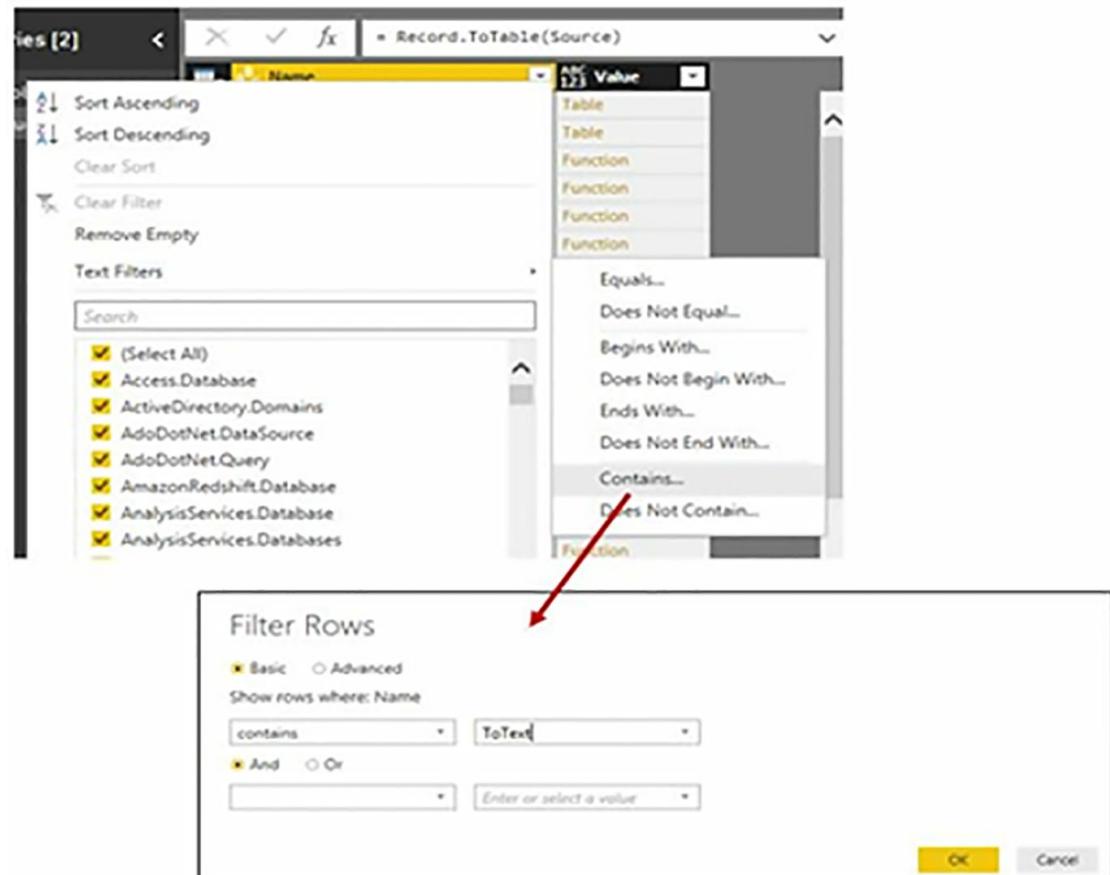
If we click again, this time on the Function value, the information on the function will be expanded, covering the entire window and opening the user interaction dialog box so that we can evaluate its operation.



We can also use other filter variants, such as the search for functions that contain a specific substring within their name. First of all, we will remove the last steps

created in the Applied Steps panel until we remain in Converted to Table. Deleting a step is very simple, just select it and press Delete or click on the delete icon next to its name.

Then we will open the filter box again, and in the Text Filters option, we will select Contains, entering the ToText value in the filter box. After accepting this dialog box, we will obtain it. As a result, all the functions whose name contains the introduced particle.



#	Name	Type
1	Duration.ToString	Function
2	Number.ToString	Function
3	Binary.ToString	Function
4	Date.ToString	Function
5	DateTime.ToString	Function
6	DateTimeZone.ToString	Function
7	Time.ToString	Function
8	Logical.ToString	Function
9	Lines.ToString	Function

Once we finish consulting the documentation using these techniques, we can select the Query1 table in the Queries pane and delete it by pressing the Delete key.

### ***Column to Identify the Sex of the Individual***

The digit located in position 6 of the Individual column identifies the sex of the person, the value being 1 for men and 6 for women. We will transfer this value to the new column SexID, where in addition to the Text.Range function we will use Number.FromText from text, to convert the result to a type of numerical data.

```
Number.FromText (Text.Range ([Individual], 5,1))
```

### ***Data Types in Power Query. Choose Between Any or a Specific Type***

When creating the SexID column in the previous section, despite having used the Number.FromText function to convert its content to a number, the data type of the generated column is Any (any or generic type) instead of Whole Number (number integer) as was our intention, which explicitly obliges us to change the data type of the column by clicking on its name, and then, in the Home menu, Transform group, Data Type option, select the Whole Number value.

Whenever we are faced with a situation of these characteristics, in which Power Query has assigned the default Any type to a column, but we know what is the appropriate type of data that it should have, the recommendation is that we change its type to which it really corresponds, so that operations with it can be carried out with greater precision and reliability.

To quickly know the type of data that a column has, we will look at the icon located on the left side of its title.

Another technique that we can also use to change the type of data is to click on the icon in the column header and choose the new type in the list that appears.

From here, in those operations that involve the creation of a column, we will specify next to its name the type of data that should have

### ***Place of Birth. Province***

The next column, which will have the name Province BirthBirth (Text), is obtained from positions 7 and 8 of the Individual column, and represents, as its name indicates, the Province of birth of the person. For the population born

outside the country, value 66 will be used, and in the case of those born in former American territories, code 53 is used.

ProvinceBirthID

=====

Text.Range ([Individual], 6.2

### ***Place of Birth. City or Country***

The municipality or country of birth is obtained from positions 9, 10, and 11, moving to the column CityPaisBirthID (Text). For people who were not born in the country, the value will correspond to the code of the country of birth. As was the case with the place of residence, in municipalities with fewer than 10,000 inhabitants, this value is bleached for data protection, so that for these cases, we will use code 999.

Municipality Country Birth ID

=====

if Text.Length (Text.Trim (Text.Range ([Individual], 8.3))) = 0 then

"999"

else

Text.Range ([Individual], 8.3)

Nationality

The nationality code data, which we will transfer to a column with the name NationalityID (Text), is in positions 12, 13, and 14.

Nationality ID

=====

Text.Range ([Individual], 11.3)

## Age

The age of the individual will be transferred to a column called AgeID (Whole Number) obtained from positions 15, 16, and 17.

### Age ID

=====

Text.Range ([Individual], 14,3)

## ***Municipality Size According to Inhabitants***

And we finish with the creation of the columns that indicate the size of the municipality of residence and birth according to its number of inhabitants: TamMunicipleResidantID (Text) and TamMunicipalBirthID (Text). Both have a length of 2 digits and are respectively in positions 18 and 19 as well as in 20 and 21.

For reasons of statistical confidentiality, in municipalities with less than 10,000 inhabitants, if for a Province, there is only one municipality in a population section, it has been added to the next or previous section, maintaining the code of the latter.

As was the case with the Province of birth, for column TamMunicipalBirthID, in those people born abroad, code 66 is assigned, while those born in former American territories are coded with the value 53. The following code block shows the expressions to use to build these columns.

### TamMunicipalityResidenceID

=====

Text.Range ([Individual], 17,2)

TamMunnoverBirthID

=====

Text.Range ([Individual], 19,2)

### ***Composition of the Place of Residence Identifier***

CityResidantID is not a column that can be used autonomously to select the place of residence of the individual. The most palpable sample of this is in the fact that if we review the CMUN sheet of the source Excel file, code 004, for example, corresponds to municipalities in 5 different Provinces, so to refer uniquely to one locality, we need both codes: Province and municipality.

For this reason, we will create a new column called Resident Place (Text), in which we will concatenate the values of ProvinceResidantID and CityResidantID. This will allow us to later build a query with an identifier of the same characteristics, which we will use as a search table when implementing the data model in Power BI.

PlaceResidenceID

=====

[ProvinceResidantID] & [MunicipalityResidantID]

### ***Composition of the Birthplace Identifier***

Suppose that in the phase of creating the data model, we want to design the information of the place of birth as a hierarchy formed by the country, Province, and municipality levels. One way to solve this problem would be the creation, in the current table, of a new column that acts as an identifier, which we will call Birth PlaceID (Text), formed by three different codes corresponding to the levels just mentioned. The way to compose this identifier will vary depending on whether the individual was born in the country or abroad, data that we will find out by consulting the column ProvinceBirthID.

If the value of ProvinceBirthID is 66, it is a foreigner, which means that the

column MunicipalityPaisBirthID contains the code of the country of birth instead of a municipality. Of course, in this case, we lack the municipality code, so we assign the value 999 to indicate that it is an unknown or unavailable location.

[Municipality CountryBirthID] & [ProvinceBirthID] & "999"

If the value of ProvinceBirthID is not 66, we will begin to construct the identifier with the value 101, which corresponds to the country code of USA, for example, followed by ProvinceBirthID and CityPaisBirthID.

"108" & [ProvinceBirthID] & [MunicipalityPaisBirthID]

The entire expression to create the column is shown in the following block of code.

```
if [Birth ProvinceID] = "66" then
    // if he was born abroad
    [Municipality CountryBirthID] & [ProvinceBirthID] & "999"
else
    // if he was born in the USA
    "108" & [ProvinceBirthID] & [MunicipalityPaisBirthID]
```

After the creation of CityBirthID, we will eliminate those other columns used as support in the preparation of this table, which we will not need to use from now on for the development of the data model. The columns in question will be: Individual, ProvinceResidantID, CityResidantID, ProvinceBirthID and CityCountryBirthID.

Next, we will execute the Close & Apply menu option again in the Query Editor, and in the Power BI window, we will save the changes in the .pbix file.

At this point we conclude this chapter in which we have made an introduction to Power BI from an ETL tool perspective, making use of the data transformation operations belonging to the Power Query engine integrated in this tool, with the aim of putting together the data table for a model that we will build with Power BI designers.

In the next chapter, we will continue with the development of this information

system, addressing the creation, also with the Query Editor, of the search tables, which once related to the data table, allow users to perform their analysis tasks through the application of filters and the creation of metrics.

# **Chapter Six:**

## **Data Model Using Power Query**



In the last chapter on the development of an information system based on the Population Register, we explained the steps to follow to create the data table of an analysis model using the Power Query tools (language M) integrated with Power BI. The next stage in the development of this model is to create the lookup tables, a task for which we will also use the Query Editor. The source Excel file described in the previous delivery will be this time the primary source of data for the new tables that we will study in this chapter.

Once these search tables have been created, we will have all the elements to build the analytical model: relationships, measures, reports, dashboards, etc. through the designers of Power BI, this aspect will be addressed in an upcoming chapter.

### **Nationality. Import from Excel**

We will start by importing the data located on the NACI sheet of the source Excel file, which corresponds to the nationality of the individual. From the main Power BI window, in the Home menu tab, External data group, Get Data | Excel, we will select the source Excel file.

Next, the Navigator window will offer us the possibility of choosing one or several sheets of this file. After selecting NACI, we will click on Edit, thus passing the data of the sheet as a new query to the Queries pane of the Query Editor window.

#### ***Nationality. Set Column Headings from the First Row of the Query***

After changing the name of the query for Nationality in the Queries panel, we have to replace the current column titles (Column1 and Column2) with names that adequately describe their content.

One possibility is to reuse existing names in the Excel sheet from which we have imported this data. These names are currently in the third row of the table, so we will remove the first two rows (which do not contain valid information) using the menu option Remove Rows | Remove Top Rows, from the Reduce Rows group, Home tab, which will show a dialog box in which we will enter the number of rows to delete.

As a next step, we will transfer the values of the first row of the table to the column titles, using the Use First Row as Headers option, of the Transform group, Home menu tab.

We have explained this technique of assigning names to the headings of a table simply for demonstration purposes so that the reader knows the existence of this possibility. However, we will not reuse the column titles of the source Excel file, but we will assign new names explicitly; so in this table, we will change the names of the columns by NationalityID (Text) and Nationality (Text), by right-clicking on each title and selecting the Rename option.

This table also contains a significant amount of empty records at the end, which we will delete using the menu option Remove Rows | Remove Blank Rows, from the Reduce Rows group, Home tab.

### ***Creating the Sex Table from Dynamic Records***

In the steps carried out to build the PopulationCensus table, we saw that the values used to identify the sex of the individual are 1 for men and 6 for women, so the base query for the search table of these data, being very simple, we will create it dynamically through an expression in M language.

First, we will select the menu option New Source | Blank Query, from the New Query group, Home tab, to create an empty query in which we will write the following block of code.

```
= Table.FromRecords ( { [SexID = 1, Sex = "Man"], [SexID = 6, Sex = "Woman"] } )
```

```
}
```

```
)
```

Let us analyze each of the parts of the previous expression: first, to define a record in M, we must enclose in square brackets the field names that compose it next to their values, using the format shown in the following figure.

Next, the records that will be part of the table will be included, separated by commas, in a list (List object). The key characters are those used in M to define a list of elements.

Finally, we will use the FromRecords function of the Table category, to which we will pass as a parameter the list we just created, obtaining a table as a result.

Once the table is created, we will change its default name to Sex; for the SexID and Sex columns, we will assign as data type Whole Number and Text, respectively.

### ***Place of Residence. Query Combination (Merge Queries)***

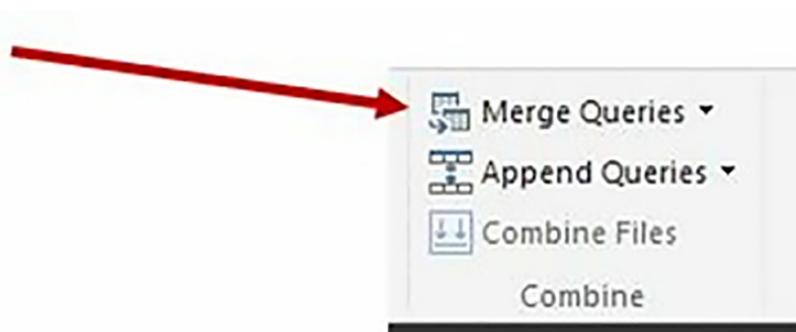
During the creation stage from the data table, we explained that the residence of the individual is established from the combination of the province and municipality codes; These values are found in the CPRO and CMUN sheets of the Excel source, so to assemble the table of the place of residence we will first have to import these sheets into Power Query, just as we did before with the nationality table, changing the name of the CPRO table by ProvinceResidence, and the columns CPRO and Province of Residence by ProvinceResidenceID (Text) and ProvinceResidence (Text). Regarding the CMUN table, we will do the same name change operation for MunicipalityResidence, and the columns CPRO, CMUN, and NAME for ProvinceResidenceID (Text), MunicipalityResidenceID (Text) and MunicipalityResidence (Text).

Next, in the Queries panel, we will click on ProvinceResidence, and we will delete the first three rows with the menu option Remove Rows | Remove Top Rows. For the MunicipalityResidence query, we will delete the first five rows.

Reviewing the contents of these tables, we will appreciate that in MunicipalityResidence we have almost everything necessary; at the moment a column with the name of the province is missing, an issue that we will solve by crossing with ProvinceResidence, to transfer the data of the column of the same

name.

The first requirement for a crossing of these characteristics is to be positioned in the table to which we are going to add data, so in the Queries panel we will click on MunicipalityResidence, and then we will select the Merge Queries menu option, from the Combine group, on the Home tab.



Then the Merge window will open, which in its upper part shows the origin table of the crossing: MunicipalityResidence. We will also have to select the ProvinceResidence table at the bottom, and in both the crossing and union field, which has the same name in the two tables: ProvinceResidenceID. The type of crossing or Join Kind, which in this case is Left Outer, indicates that for the operation, all the rows of the first table, MunicipalityResidence, and those rows of the second, ProvinceResidence, in which there is a match will be used. The last element of this window is an informational message with the number of rows between which there has been a match, and which in this example has been for all rows.

The consequence of this union operation when accepting the Merge window will be the aggregation, in the MunicipalityResidence table, of an additional column with the name NewColumn, containing the value Table in its cells. If we click on an empty area of any of these cells, the record obtained from the ProvinceResidence table, the result of the crossing operation, will be displayed at the bottom of the data area.

However, we have not yet transferred any data between the tables. Now that we have established a relationship between them thanks to the Merge Queries option, we must select what data from the ProvinceResidence table we want to take to MunicipalityResidence; what we will achieve by clicking on the icon next to the header of the NewColumn column, selecting the Expand option (default) and checking only the box in the ProvinceResidence column.

When you click OK, the selected action (Expand) expands the ProvinceResidence table to which each cell in the NewColumn column points, and as a result, brings the MunicipalityResidence a column with the name NewColumn.ProvinceResidence containing the names of the provinces.

After changing the name of the new column to ProvinceResidence, we will move its position until it is located next to ProvinceResidenceID. We will use the Move | Left, from the Any Column group, on the Transform menu tab. We will have to execute this option a couple of times until the column is placed in the desired position.

**Place of residence. Creation of the code for bleached municipalities and union of consultations**

When generating the code of the place of residence in the PopulationCensus table, we saw that for the municipalities with less than 10,000 inhabitants, the municipality code is not provided due to data protection issues. This had led us to decide to use the code 999 for such cases, which means that, for example, for province 08 (Barcelona), the code of the place of residence will be 08999.

The MunicipalityResidence table that we have just created would currently lack all this set of records that we do know about the province, but we do not know the municipality, and that is necessary for the measures that we are going to implement in the modeling phase to correctly perform their calculations. Then we explain a technique that will help us solve this problem.

In the ProvinceResidence table, we will add two new columns: MunicipalityResidenceID (Text) and MunicipalityResidence (Text), containing respectively the code 999 and the literal Not available.

Next, we will position ourselves in the MunicipalityResidence table and select the Append Queries menu option, from the Combine group, Home tab. In the Append window, we will select the ProvinceResidence table, and when you click OK, the records of the latter will be added to MunicipalityResidence, completing the code combinations necessary for this table.

### ***Place of Residence. Column for the Table Identifier***

We will finish the creation of this table by adding a new column that represents its identifier, to which we will give the name PlaceResidenceID (Text), composed of the union of the columns ProvinceResidenceID and

MunicipalityResidenceID.

Once created, we will place this column in the first position of the table using the menu option Move | To Beginning, from the Any Column group, on the Transform tab.

### ***Age. Creation of Numerical Intervals Using Lists***

The table with the age of the individuals will contain the values in the range of 0 to 99 years; for those with 100 years or more, the value 999 will be used for data protection reasons.

The M language allows us to generate a sequence of numbers using the Numbers function of the List category, passing as parameters the initial value and the number of numbers to be created.

```
= List.Numbers (0,100)
```

However, this is not enough, since we also have to add the number 999 to these values. To solve this problem, we will use the Combine function, also of the List category, which receives as a parameter an object of type List containing a combination of lists this function will return as a single list.

In this way, in addition to the list with the range 0-99, we will create another composed only by the number 999, which we will also pass to combine.

```
= List.Combine (  
    {  
        List.Numbers (0,100),  
        {999}  
    }  
)
```

The previous block of code will be included in a new empty query (New Source | Blank Query menu option, from the New Query group, Home tab), which will create a unique list by adding it to the Queries panel.

In the next step, we will convert the list into a table with the To Table menu option, from the Convert group, located on the List Tools | Transform.

In the To Table window that opens next, we will accept the default values. Once the table is created, we will change its name to Age and the name of the column to AgeID (Whole Number). Next, we will add a new column with the name Age (Text) from the values of AgeID, but converted to character, using the following expression.

```
Number.text ([AgeID])
```

An aspect widely used in work with demographic data consists of analyzing the population's age in ten-year, five-year, or other range intervals. For this example we will define the following intervals: 0, 1-4, 5-14, 15-24, 25-34, 35-44, 45-54, 55-64, 65-74, 75-84, 85 and more years.

The way to implement these intervals in the age table will be through the incorporation of a new column, which we will call Decennial Age (Text), in which we will include the following block of code, where based on the interval in which the age is placed, We will assign the corresponding literal.

```
if [AgeID] = 0 then
    "0"
else
    if [AgeID]> = 1 and [AgeID] <= 4 then
        "1-4"
    else
        if [AgeID]> = 5 and [AgeID] <= 14 then
            "5-14"
        else
            if [AgeID]> = 15 and [AgeID] <= 24 then
                "15-24"
            else
                if [AgeID]> = 25 and [AgeID] <= 34 then
                    "25-34"
                else
```

```

if [AgeID]> = 35 and [AgeID] <= 44 then
    "35-44"
else
    if [AgeID]> = 45 and [AgeID] <= 54 then
        "45-54"
    else
        if [AgeID]> = 55 and [AgeID] <= 64 then
            "55-64"
        else
            if [AgeID]> = 65 and [AgeID] <= 74 then
                "65-74"
            else
                if [AgeID]> = 75 and [AgeID] <= 84 then
                    "75-84"
                else
                    if [AgeID]> = 85 then
                        "85+"
                    else
                        ""

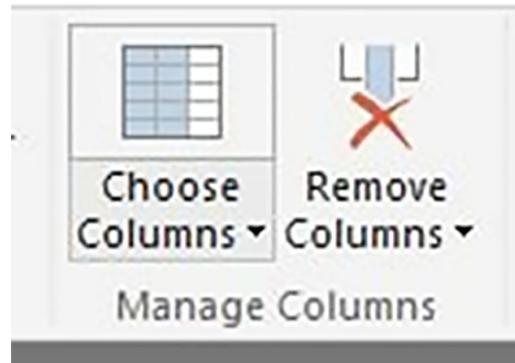
```

### ***Size of the Municipality of Residence***

Our next operation will be to create the table with which we can classify individuals according to the size by inhabitants of the municipality in which they reside.

To do this, we will import the TAMU sheet from the source Excel file. Once the data has been incorporated into Power Query, we will remove the last column of

the table, which is empty, using the menu option Remove Columns, from the Manage Columns group, Home tab.



Next, we will change the name of the table to TamMunicipalityResidence, and the column names to TamMunicipalityResidenceID (Text) and TamMunicipalityResidenceOrigin (Text), eliminating the first three and last four rows using the Remove Top Rows and Remove Bottom Rows suboptions respectively, depending on the Remove Rows menu option.

Since the objective of this table will be to inform whether the municipality is Provincial capital, and otherwise, the range of inhabitants it has, the descriptors obtained from the Excel file may be excessively long and repetitive. For this reason, we will try to summarize them, so that we simply show the literal "Province Capital" when it is a Provincial capital city or the range of inhabitants in the rest of the cases.

We will create a new column with the name TamMunicipalityResidence (Text) using the following block of code.

```
if Text.Contains ([TamMunicipalityResidenceOrigin], "provincia") then
    "Province capital."
else
    if Text.Contains ([TamMunicipalityResidenceOrigin], "hasta") then
        "Up to 100."
    else
        if Text.Contains ([TamMunicipalityResidenceOrigin], "de") then
```

```
Text.RemoveRange ([TamMunicipleResidenceOrigin], 0,24)  
else
```

```
Text.RemoveRange ([TamMunicipleResidenceOrigin], 0,21)
```

The technique used in the previous code is to check the value of the TamMunicipalityResidenceOrigin column using the Text.It contains a function, which returns true or false, depending on whether or not, within the text passed as the first parameter, the character string of the second parameter. In cases where the word "province" exists, we will know that it is a Provincial capital city and we will return the literal "Province Capital"; If it contains the word "up to" it will be a municipality with a maximum of 100 inhabitants, and we will return "Up to 100".

For the rest of the occurrences, we are only interested in taking the text fragment that indicates the range of inhabitants, eliminating the rest with the Text.RemoveRange function, which is responsible for eliminating, in the text of the first parameter, a substring whose start index we indicate in the second parameter, while its length is specified in the third parameter. In these values, the amount of text to be deleted will depend on the existence of the "de" particle in the value of the column.

Regarding the size by inhabitants of the municipality of birth (sheet ALSO of the Excel source file), the creation of your query is equal to the process just described, so we will avoid repeating it here.

### ***Place of Birth. Data Distribution by Country, Province, and Municipality***

As we had explained in the previous chapter, the table with the information of the place of birth that we will build next will have a structure that will allow us, during the design of the data model, to create a hierarchy based on the levels of country, province, and municipality.

First, we will import the CPRON and CMUNN sheets from the source Excel file, which in the Queries panel, we will change their name to Province of Birth and Municipality of Country of Birth, respectively.

Placing ourselves in the Province of Birth table, we will rename its columns by Province Birth Birth (Text) and Birth Province (Text); then, we will delete the first three rows.

Then we will position ourselves in the MunicipalityCountryBirth table, where we will remove its empty columns, and then, from the Queries panel, we will right-click on this same table, selecting the Duplicate option. We will change the name to the new duplicate table by Country.

We will deal with this table a little later, now we will return to MunicipalityCountryBirth, and first, we will delete the last three columns, corresponding to the country data, using the Remove Columns menu option

Continuing with the delete operations, then we will use the menu sub-options dependent on Remove Rows to delete the first eight rows, as well as those rows with null values. We will also change the default names of the columns to ProvinceBirthID (Text), MunicipalityBirthID (Text), and MunicipalityBirth (Text).

### ***Place of Birth. Completing Provinces Information***

As it happened during the preparation of the table of the municipality of residence, the MunicipalityCountryBirth table has the province code but not the names, so we will make a combination (Merge Queries menu option) with the ProvinceBirth table, to transfer from this Last column with the province names to the table of municipalities. We will not repeat the steps on how to make a combination of tables since this operation was described earlier during the preparation of the tables relating to the place of residence. Once the crossing is finished, and the positions of the columns are organized.

As indicated in the source Excel file, information on births in municipalities with less than 10,000 inhabitants is bleached for data protection, which means that for these cases, we will know the province of birth but not the municipality. The technique that we will use to incorporate these data in the MunicipalityCountryBirth table will be the same as that previously used for the MunicipalityResidence table.

First, we will add two new columns to the ProvinceBirth table: MunicipalityBirthID (Text) and MunicipalityBirth (Text), containing respectively the values 999 and Not available.

Then, using the Append Queries menu option, we will add to the MunicipalityCountry table the content of the ProvinceBirth table. As a result, MunicipalityCountryBirth will now contain all possible combinations of

province and municipality, including those of bleached municipalities (not available) because they have a population of fewer than 10,000 inhabitants.

### ***Place of Birth. Completing Country Information***

The next step will be to create two new columns: CountryBirthID (Text) and CountryBirth (Text), to host the country data. Currently, the table only has information on provinces and municipalities in the USA, so the values of the two new columns will be country code 108 (see the reader for the Nationality table) and the literal USA, respectively. After rearranging the columns in a more logical order (country - province - municipality).

Now it is necessary to add the data of the rest of the countries, so we will place ourselves in the CountryBirth table to prepare its structure and contents so that we can combine it with MunicipalityCountryBirth.

The columns of the CountryBirth table that we need to work with are the last 3, which correspond to the province code 66, indicative of being born abroad, as well as the country code and name.

As an alternative way to delete columns, we will select the columns that contain the data we need, and we will execute the Remove Columns | Remove Other Columns from the Manage Columns group, Home menu tab; thus deleting the columns of the table that we had not selected.

This table contains a large number of empty records (null, blank, etc.) that we will delete with the Remove Rows | Remove Blank Rows, used on other occasions throughout for similar situations. After this deletion, we will also delete the first two rows, to keep only the data of the countries.

The next step will be to change the name of the columns by Province Birth Birth (Text), Country Birth ID (Text), and Country Birth (Text).

Now we must match the structure of this table with the MunicipalityCountryBirth table, adding the columns ProvinceBirth (Text), MunicipalityBirthID (Text), and MunicipalityBirth (Text). Since the CountryBirth table contains only the data for the population born abroad, we do not have, of course, data on province or municipality, so we will assign values to these new columns according to this situation: 999 for NacimientoID Municipality and the literal Not available for ProvinceBirth and MunicipalityBirth.

Next, we will add the records of this table to MunicipalityCountryBirth through the menu option Append Queries, obtaining. As a result of all combinations of codes that will allow us to locate the population geographically.

### ***Place of Birth. Deletion (filter) of Invalid Rows***

After adding this data, we need to remove from the MunicipalityCountryBirth table the row that contains the foreign value in the ProvinceBirth field, since it corresponds to a non-existent case in the PopulationCensus data table: individual born in the country but classified as a foreigner by province.

The operation that we are going to carry out to achieve this objective will not consist, however, in the deletion of a row, but we will apply a filter on the table, which excludes the record in question; for what we will click on the filtering icon in the Birth Province column, selecting the Text Filters | Does Not Equal. In the Filter Rows window shown below, we will enter the foreign value and accept; this will apply the filter, excluding said row from the rest of the rows of the MunicipalityCityBirth table.

### ***Place of Birth. Column Creation with a Table Identifier***

As the last task, we will add a column that will act as the identifier of the table, which we will call Birth PlaceID (Text). This column will be formed by the concatenation of the CountryBirthID, ProvinceBirthID, and MunicipalityBirthID columns; Once created, we will place it as the first column of the table.

[CountryBirthID] & [ProvinceBirthID] & [MunicipalityBirthID]

### ***Disable Loading Auxiliary Tables***

At this point, we can conclude the process of creating the tables that will be part of the data model. However, we will not need to use in the modeling process all the tables that the Query Editor currently contains, so we will disable their loading to achieve a cleaner design of the model and optimize the processing of your data.

Located in the Queries panel, to disable the loading of a table, we will right-click on it, unchecking the Enable load option box. The tables on which we will carry out this operation are ProvinceResidence, ProvinceBirth, and CountryBirth.

On the other hand, we will re-enable data loading for the PopulationCensus

table, since we had previously disabled it during the preparation phase of the model search tables, also for performance reasons.

In this and the previous chapter, we have addressed the creation of the search tables or lookup tables of an analysis system built using Power Query ETL tools integrated into Power BI. In the next chapter, we will discuss those aspects of the information system that concern the design of the data model: relationships between tables, creation of hierarchies, measures, etc., as a previous step to the visual representation of the information contained in the system.

# **Chapter Seven:**

## **Power Pivot Engine To Design A Population Data Model**



**I**n the past two chapters ( chapter 5 - chapter 6 ), Query Editor was the Power BI tool that completely occupied our attention, showing us how to build the tables of a model from the functionalities of extraction, transformation, and load (ETL) that it has.

Once the creation of the tables is finished, in this section we will continue our task of developing the data model, using on this occasion the Power BI designers most directly related to the Power Pivot engine: Data and Relationships, responsible for the development of measures, hierarchies and relationships between the tables of the model. We will also pass briefly through the Designer Report, just to check the values of the measurements generated in the data table.

### **Data Designer Introduction**

Finished the work of preparing tables from the Query Editor we will place in the Data designer, which among other operations, we will use to create the measures of the model through expressions written in DAX, the query language of the Power Pivot engine integrated in Power BI, as well as to format the values of the columns and create hierarchies in the tables.

The following figure shows an overview of this designer, with a central area dedicated to the display of the selected table in the Fields pane on the right, and the main work options in the Modeling menu tab.

### **Data Designer Population Count Measure**

As we have already indicated, from this designer we will create the main measure of the model, to which we will give the name of Total Population (to simplify, we will avoid the use of tilde and other special characters in the names

of the measures), and whose objective will be to make a count of the individuals that make up the Register of inhabitants that we are analyzing.

To do this, we will select the PopulationCensus table and click on the New Measure menu option, from the Calculations group, Modeling tab, which will position us in the formula bar of the table, where we will write the following DAX expression.

Total Population = COUNTROWS (PopulationCensus)

As we have already guessed, in the previous expression, the COUNTROWS function is responsible for counting the rows of the PopulationCensus table that we pass as a parameter, to obtain the total number of inhabitants.

Then we will format the measure by adding a thousand separator characters, which we will select in the Modeling menu tab, Formatting group, Thousands separator option.

## **Report Designer Elementary Operations in the Use of Controls**

To check the operation of the measure that we have just created, we will move to the Designer Report, whose purpose is to build a graphic representation of the information in the data model, through the visual objects contained in its palette of controls.

As a general rule, the steps used to add a control with a minimum of functionality to the Report designer are basically the following:

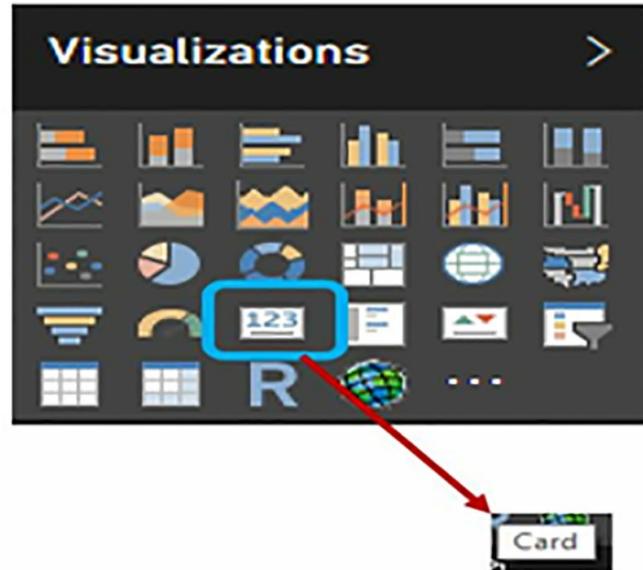
- 1) Select the control in the palette of the Visualizations panel.
- 2) The previous action places a copy of the control in the design area.
- 3) Select from the fields of the Fields panel the measure (s) that we want to represent graphically, and place it in the properties of the Fields section of the control.
- 4) Depending on the power and visual richness of the control, use some of the properties of the Format section to improve its appearance.

## **Report Designer Results Display**

In this first approach to the Power BI report designer, we are not going to create a very complex or elaborate presentation, and we just need to verify that the data

table count is correct through visual control.

For the demonstration, we will use the Card control, whose operation consists of a simple visualization of the value of a measurement.



Once the control is selected in the Visualizations panel, we will click on the PopulationCensus table of the Fields panel to display its elements (fields, measurements, hierarchies) and check the box of the Total Population measure, which will automatically go to the Fields property of the control, belonging to the section of the same name, within the Visualizations panel. As a result, the control will display the measurement value using an abbreviated numerical format.

To show the value of the measurement with the complete number format, we will click on the Format section of the Visualizations pane, and within the Data label section, we will change the value of the Display units property from Auto to None.

## Measurement Control Test

To ensure that the number of individuals is correct, we will return to the Query Editor by clicking on the Edit Queries menu option, External data group, Home tab. Once there, in the PopulationCensus table, we will select the Count Rows menu option, from the Table group, Transform tab, which will also count the records in the table, returning the same value (without formatting) as the measurement used in the Report designer.

Once this verification is finished, we will delete the counting operation from the Applied Steps list in the Query Settings panel, and close the Query Editor window, applying the changes to return to the report designer.

## **Tables Need Relationships**

The fact that the previous rapid test of verification of the value of the measure has given the expected value does not mean, at all, that the model preparation work has been completed. As evidence of this, if we add a control to the designer of the Report panel, in which in addition to the Total Population measure, fields from any other table of the model intervene, the results will not be as desired.

As an example, we are going to add a new page to the pbix file we are working with, by clicking on the New Page button at the bottom of the Power BI window. On this page, we will add a Stacked column chart control, to measure the number of inhabitants per age group, showing the representation of each section in a column. Once the control is placed in the design zone, we will select the Total Population measure, which will be assigned to the Value property of the Fields section. In the Age table, we will choose the Decennial Age field, which will be assigned to the Axis property; As a result, the control should show the columns corresponding to each age range with different sizes, thus reflecting the number of existing people by range.

However, this is not the behavior obtained, since the control shows all the columns with the same size, without correctly reflecting the population for each age group, which is a symptom of some type of lack in the model configuration sections.

The origin of this problem lies in the lack of relationships between the data table and the model search since without a relationship between the Population and Age tables, the Power Pivot engine does not know the amount of population that should be assigned to each age group, and therefore, the control is not drawn correctly.

## **Relationships Designer. Relationship Creation**

To solve the problem posed in the previous section, we will locate ourselves in the Relationships designer, who, as indicated by his name, has the purpose of establishing the relationships between both types of table of the model: data and

search, so that the visual representation of the data from a Power BI report, correctly reflect the information that you want to transmit to the user.

Within the work area of this designer, we see an outline of the tables that make up the data model. If, due to their quantity, they were not all insight, we can adjust their size by clicking on the Fit to screen button or the Zoom controls, located at the bottom.

We can also drag the tables to place them more properly, according to the relationships we need to create.

Next, we will proceed to create the relationship between PopulationCensus and Age, for which we will select the column AgeID in the PopulationCensus table, and drag it to the column of the same name in the Age table, establishing the relationship with a cardinality of many to one.

If we double click on the line that represents the relationship, the Edit relationship window will open, through which we can see its characteristics.

Upon returning to the Report Designer, the Stacked column chart control will now reflect the information properly, although the order of the age ranges will not be totally correct, which we will solve in the next installment dedicated to the development of reports with the Designer Report.

The list will show the rest of the relationships that we must establish between the PopulationCensus table and the other tables, so that the controls that we incorporate into our reports, dashboards, etc., adequately show the calculations of the model metrics.

Alternatively, we can also manage the model relationships from the Manage Relationships window, accessible from the menu option of the same name, located on the Home tab, Relationships group.

## **Data Designer Hierarchy Creation**

A hierarchy is an element, which, once defined in a table of the model, will allow us to analyze its information through levels, which represent various categories of related data within that table. As an example, we will create a hierarchy based on the MunicipalityCountryBirth table, which will consist of three levels: country, province, and municipality.

In the first place, we will place ourselves in the Data designer, and in the Fields

panel, we will display the list of fields of the MunicipalityCountryBirth table. Then we will right-click on the Country Birth field, choosing the New hierarchy option.

This action will create a hierarchy with the default name Country Hierarchy. As a next step, we will successively drag the ProvinceBirth and MunicipalityBirth fields until they are released within the name of the hierarchy, now being composed of the three levels mentioned above.

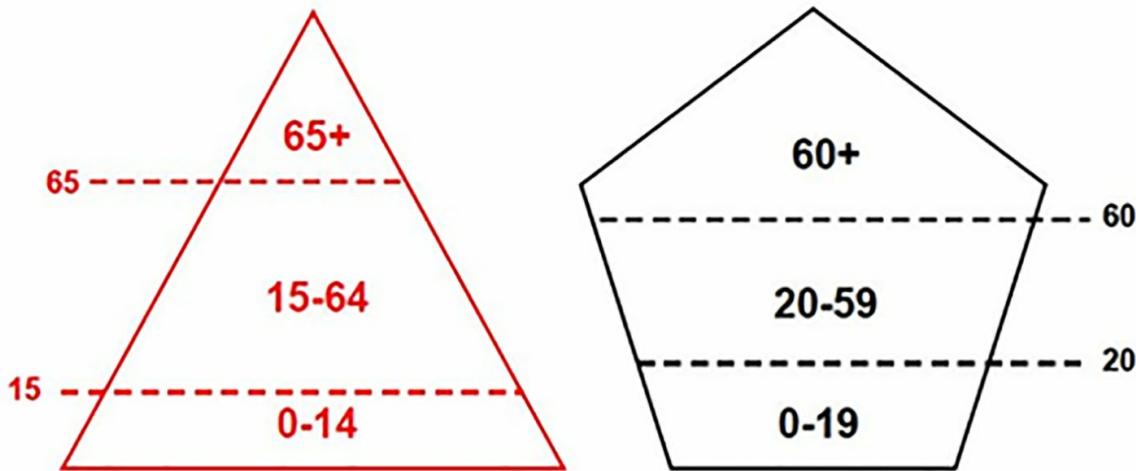
Finally, we will change the name of both the hierarchy (CountryProvinceMunicipality) and its levels (Country, Province, and Municipality), by right-clicking on each element and choosing the Rename option.

## **Demographic Structure Indicators**

In addition to the measurements performed by the count of individuals, the demographic analysis employs a series of additional indicators, whose purpose is to reflect the distribution of the population through the variables of sex and age; they are called Demographic Structure Indicators.

Starting from three large age groups, with intervals 0-14, 15-64 and 65+ (65 years and over), the structure indicators measure aspects such as the proportion of a certain group over the total population; the relationship of the groups of elderly or young people with respect to adults; and the reason between old and young.

The demographic change processes experienced in certain populations (low fertility, high life expectancy, etc.) have caused that, in addition to the traditional grouping indicated in the previous paragraph, some organizations such as Eurostat use a second grouping based on intervals 0- 19, 20-59 and 60+ (60 and older). The following figure shows these two types of age groups.



## Demographic Structure Indicators

We will begin by creating the three measures that represent the number of inhabitants for each age group (young, adults and older). As we did previously with the Total Population measurement, we will return to the PopulationCensus table of the Fields pane, and select the New Measure menu option, to create the measures Young Population, Adult Population and Major Population, whose source code we see below.

```
Youth Population = CALCULATE (
    [Total population],
    Age [AgeID]> = 0 && Age [AgeID] <= 14
)
```

```
Adult Population = CALCULATE (
    [Total population],
    Age [AgeID]> = 15 && Age [AgeID] <= 64
)
```

```
Major Population = CALCULATE (
```

```
[Total population],  
Age [AgeID]> = 65  
)
```

The CALCULATE function, used to create these measures, evaluates the last expression as the first parameter (Total Population measurement of the PopulationCensus table) by modifying its original evaluation context by using filter expressions (selection of intervals on the AgeID column of the Age table with DAX operators ) located in the second and subsequent parameters.

Following the recommendation of the reference sheet available in PowerPivotPro, when writing the code of a measure within which we have to refer to another previously created measure, for that existing measure, we will only use its name without first putting the name of the table.

To check the values that we have just created, we will return to the Report Designer, and on a new page, we will add a Multi-row card control, which will facilitate the simultaneous visualization of several measures by including them in their Fields property.

### ***Demographic Structure Indicators. Proportions over the Total Population***

This type of indicator calculates the percentage that represents each age group (youth, adults, and the elderly) with respect to the total population, so we will have to create three measures, one for each group. To obtain it, we will use the DIVIDE function, as we see in the following block of code, where the measure of the corresponding population group acts as the numerator and the total population as the denominator.

```
Young Proportion = DIVIDE ([Young Population], [Total Population])
```

```
ProportionAdultos = DIVIDE ([Adult Population], [Total Population])
```

```
Major Proportion = DIVIDE ([Major Population], [Total Population])
```

Additionally, we will format each of these measures as a percentage with two decimals.

And we will also use a Multi-row card control to visualize its values.

Demographic structure indicators. Dependency Indices

These are indicators that establish a relationship between the groups of young and / or older population (numerator) with respect to the adult group (denominator), so we will use the following expressions to calculate them.

Youth Dependency Index = DIVIDE ([Youth Population], [Adult Population])

Major Dependency Index = DIVIDE ([Major Population], [Adult Population])

### ***Demographic Structure Indicators. Aging Rates***

And we finish the calculation of indicators with this measure, which allows us to observe the aging process of the population through the ratio between the population of elderly and young people using the following formula.

Aging Index =  
DIVIDE ([Older Population], [Young People])

And with the creation of demographic indicators, we finalize this chapter, in which we have shown the role that Data and Relationships designers play in the process of developing an information system with Power BI, specifically in the section on creating the model of data, through the elaboration of the metrics, relationships, and hierarchies.

## Conclusion



In the past, companies were stuck with tables, spreadsheets, and static charts. Today, the use of joint and connector tools makes it possible to use and generate data more intelligently. Flexibility and personalization of data storage allow you to analyze and guide decisions in companies and corporations.

Data pinned to offline spreadsheets does not allow correct and fast sharing. We know that business data and information has increased. On the other hand, the speed at which companies need to make decisions also increases.

Using the right tools for the data processing task increases the speed for processing, manipulation, and understanding. Saving time is one of the main benefits. Regardless of the area of business, corporations and new businesses need to adopt more dynamic and powerful analytics tools.

Large, medium and small businesses must choose this path urgently. Information cannot be lost and the company cannot waste time. Microsoft's Power Bi program is critical to new endeavors.

All the best companies want to download and use this tool that connects and expands access for the workforce. The tool has four elements and enables you to turn a huge amount of data into optimized views.

This information can be absorbed and understood in moments. Being dynamic, they adapt to any need, which improves the decision-making process. The company will change its organizational and administrative culture. We can say that this tool is the present and future of enterprise data organization, allowing to overcome the old difficulties of the past.

Some companies may not see an advantage in being Microsoft, a very traditional brand that has already launched outdated services like MSN and Outlook, but the company has adopted a policy of launching more specific products for different situations.

The company is updating itself in the market and strives to make significant

changes never before seen in the company. Following the launch of innovative products, Power Bi is part of this new trend. The company listened to end-users, and they identified what they thought was best and did not develop the program based solely on what they understood.

We know the company is a leader in operating systems, personal and corporate program development. After all, who has never used Office? Despite its large portfolio, and being a superpower, the company needed to innovate in other sectors, such as management and information management.

Although new, Power Bi is inherited from SQL Server Analysis Services. The SSAS team developed the foundation for the new software from Power Pivot (the data modeling engine), one of Excel's first extensions. Most current Microsoft software can integrate with this joint tool depending on the skill of the user.

In this book, we seek to present the main features and applications of the functions of this tool that has grown a lot in the data and information management market. Data sharing and simplicity are the best reasons for companies to invest in this tool that makes it possible to optimize data and Excel connections.

Today, companies are looking for time savings, flexibility, and real-time data refresh as a way to unify the efforts of businesses and corporations. It is worth researching, installing, and studying this tool and its four interface and application elements.

## References



<https://powerbi.microsoft.com/en-us/blog/>

<https://exceleratorbi.com.au/blog/>

<https://datachant.com/>

[https://www.academia.edu/36243555/Analyzing\\_Data\\_with\\_Power\\_BI.pdf](https://www.academia.edu/36243555/Analyzing_Data_with_Power_BI.pdf)

<https://www.powerpivotpro.com/wp-content/uploads/2015/10/PowerPivotPro-Reference-Card.pdf>

<https://yodalearning.com/tutorials/power-bi-dax-basics-power-bi-dax-function-list/>

[http://www.powerpivot-info.com/public\\_files/cheat-sheet/powerpivot-dax-functions-long.pdf](http://www.powerpivot-info.com/public_files/cheat-sheet/powerpivot-dax-functions-long.pdf)

<http://projanco.com/Library/The%20Definitive%20Guide%20to%20DAX-Business%20intelligence%20with%20Microsoft%20Excel,%20SQL%20Server%20and%20Power%20BI.pdf>

# **Power BI**

---

***Simple and Effective Strategies to Learn the Functions  
of Power BI and Power Query***

**DANIEL JONES**

# Power BI: A Disruptive Reporting Platform



## Introduction to Power BI

Power BI is a cloud-based analytics service that provides a unified view of the mission-critical business data. It's Software as a Service (SaaS) offered by Microsoft in the reporting and visualization domain. It offers three basic components, which are:

- Power BI ( Online Service)
- Power BI Desktop
- Power BI Mobile

Here you will see the basics of Power BI and a critical feature termed as Natural Language Query. Before you understand each of these components separately, let's have a look at the Architecture of Power BI. The architecture will explain how Power BI components are related to each other. If not, I am here to throw more light on it. Let's understand the sequence as:

- **Create Reports using Power BI Desktop (On-Premise)** : Power BI desktop is a report authoring and designing tool that can be installed on your machine. You can use this tool to connect to more than 60 data sources and create a data model for further creating your reports.
- **Publish Reports to Power BI online service (On-Cloud)** : Once you create the reports, you need to show it to the world. But how? That's where the Power BI online service comes into the picture. The reports created using Power BI desktop can be published to Power BI online (<https://powerbi.microsoft.com>). For publishing the reports, you need to sign in to the online service using your official ID only.
- **Consumer Reports on various devices** : Once published, the reports can be shared with the world. The shared reports can be viewed on desktop browsers and even through the Power BI Mobile App.

Now, you have a fair idea about the Power BI architecture. Let us explore the three Power BI components separately.

## Power BI Online Service

Power BI helps you to stay informed about the progress of your business and keep you up to date with the business-critical information. With Power BI **dashboards**, you can keep an eye on the health of your business. The dashboards display the tiles that you can click to view the detailed **reports** for exploring more details. You can connect to multiple **datasets** to bring all the necessary data together in one place.

Did you notice the highlighted keywords in the above section? Yes, you are right; the **Dashboard**, **Report**, and **Dataset** are the major components of Power BI service. Let's understand them in detail.

### Dataset

A dataset is a collection of relevant data coming from a data source or multiple data sources. One dataset is also used to create multiple reports and dashboards.

Datasets can be created in two ways. It can be created by connecting to data sources using the Power BI Desktop tool. Alternatively, you can go to Power BI online and use the **Get Data** feature to connect to data sources for creating datasets.

Dataset is the most important building block for developing reports and dashboards. When you publish the reports Power BI Desktop to Power BI online service, the dataset also gets published to make reports intuitive and responsive.

### Report

A report is a form of visualizing data into single or multiple pages like charts, graphs, trees, images, and pie charts. All these visualizations are created using a singular dataset. All the reports can be generated from scratch, or you can also import them with the help of dashboards shared among the team members. You can also import the reports when you connect to online third-party software such as Microsoft Excel, Salesforce, Power BI Desktop, content packages, and backend databases.

A report can be viewed and interacted in two different modes: **Reading View**

and **Editing View**. The owners, co-owners, and users who are granted permission can view the reports in the Editing view. Others can view the reports in the form of a Reading view only; they can interact with reports but can't modify them.

A report can only be created using a single dataset. However, it can be pinned to multiple dashboards.

## **Dashboard**

The dashboard is a canvas than contains single or multiple tiles and widgets. These tile and widgets show the visualizations. And the visualizations are nothing but reports. So, the dashboards display reports and other components.

Dashboards can be created or shared by users among the group that has access to the shared content. Dashboards can also pin the reports coming from different sources like SSRS, Salesforce, etc.

## **Power BI Desktop**

Power BI Desktop is a report designing and authoring tool with advance data shaping and modeling capabilities that helps you to get your data ready for analysis in a short span of time.

Through Power BI Desktop, you can connect to more than 60 data sources and then make a data model using the Query Editor. Once the data model is built, you can develop your reports on it. After you create reports, you need to share them with peers. There are two ways the reports can be shared. One, the reports can be sent to peers in the form of a file with the extension .pbix, which is the default file extension for Power BI project files. Alternatively, you can publish your reports to Power BI online service and then share it with your peers. For sharing online, all your peers must have authorized access to the Power BI online service.

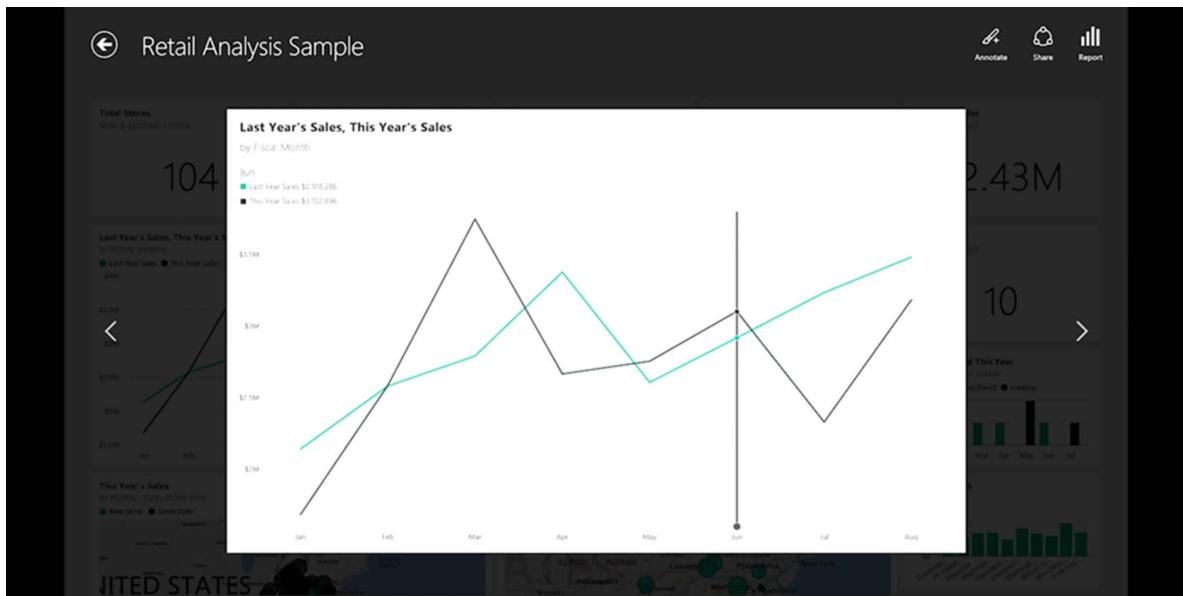
## **Power BI Mobile**

Power BI mobile is a very useful and important part of Power BI. If you want to keep track of your data while you're on the move, you can use one of Power BI's mobile applications for iOS, Android, and Windows devices. You can view all your reports, data, and all analytics on your mobile screen.

You need to install Power BI Mobile App on your mobile device or Tab. Once the app is installed, you can log in to the app using the credentials that you use for Power BI Online service. After logging in, the app takes you to the dashboards that were created by you or shared by your peers. This way, you are always connected to your business.

Power BI Mobile app allows you to set the alerts to notify you whenever there is a data modification at the source of data. This app is suitable for users who are always on the move.

You can open any of the dashboards by tapping on them. Within a dashboard, you can tap on a dashboard tile to focus on it in a larger view, as shown in the following figure.



You can also annotate any insights that you discover by tapping the Annotate button in the top right corner. This allows you to draw on a focused tile to highlight particular areas of interest. The annotation tools are found along the bottom of the screen.

Share your annotated tile by tapping the Share link in the top right-hand corner. Tap the Report link in the top right-hand corner to view the report for a tile. This will display the same visualizations that you would see in a web browser or Power BI Desktop. Also, it will include the ability to interact with the report by tapping on bars, drilling down, or using slicers.

**Please Note:** The details/demonstration of Power BI Mobile is out of the scope

of this chapter.

Now you are aware of multiple components of Power BI and ready to move on with more digging into details. However, we would like to stop here and think about why we are calling Power BI as a disruptive reporting platform. Yes, it is disruptive, and it's challenging the other tools that are available in the market. I would discuss a very significant feature of Power BI that is making it different from others. This feature is called Q&A or Natural Language Query.

## **Manage Data Sources in Power BI Desktop**

Power BI, a product of Microsoft, is self-service business intelligence. It provides highly interactive visualizations, where a user can create reports and dashboards by themselves. Power BI Desktop is an application that can be installed and run on your desktop. In Power BI Desktop, you can create data models or reports and can share them with others on the Power BI Service.

It provides a cloud-based Services –known as Power Bi Services, along with Power BI Desktop interface.

The Power BI service provides fast and easy access to data, supporting inputs from most of the data sources, and the data can be analyzed and visualized by anyone.

## **Benefits of Power BI**

1. Accessibility: Data is more accessible in Power BI. Power BI can connect to a huge number of Microsoft as well as non-Microsoft solutions.
2. Implementation: It's very easy to implement power BI.
3. Robust access control and Security: Row Level Security is the feature of Power BI, which makes it different than others. It allows a team to grant and rescind access on a very controlled level.
4. Drag and Drop functionality: Through familiar drag and drop functionality, an end-user can create his/her own report easily—Drill Down Functionality.

## **Import and Manage Data Sources in Power BI Desktop**

Here you will understand the various ways to manage your data sources in your Power BI desktop. Let's go through the steps to import a data source first.

## **Import Data Source to Power BI Desktop**

To Import a data source into your Power BI Desktop, follow the below steps:

1. From the ‘Home’ ribbon, select ‘Get data’ or click on the down arrow next to ‘Get Data.’
2. The drop Down will show some frequently used data sources along with an option ‘More’ at the bottom.
3. Click on the ‘More’ will give the ‘Get Data’ menu. The ‘Get data’ menu can be brought up by clicking on the ‘Get Data’ button in the Home ribbon.
4. The data sources have been categorized, as shown in the left pane of the above snapshot.
5. The required data source can be selected just by selecting the proper category and data source name. The data source ‘Oracle database’ has been selected which comes under ‘Database.’
6. By selecting the button ‘Connect,’ a pop up will appear, which will have a label where the Server name to be put.
7. After putting the server name / global database name as per the above snapshot, the window will appear once ‘ok’ is clicked.
8. Put the credentials in the Username and Password boxes as shown above and click ‘Connect.’ Once the data source is connected, a window will appear where you need to select the required table/tables.
9. After selecting the required tables, click on the ‘Load’ button.

## **Manage Data Source**

These steps will show you how to manage the data sources that you have already connected to your Power BI Desktop.

1. From the ‘File’ drop-down, select ‘Options and Settings’ -> ‘Data Source Settings.’
2. The ‘Data Source Settings’ window will display all the Data sources as

below:

**Data sources in the current file:** this will have the list of all data sources which have been connected in the current file.

**Global permissions :** this will have the list of all data sources which have been connected previously in all the files.

## Change Source

The below steps will show you how to change the source/ sources. Please note that the ‘Change Source’ option is available only for the current file.

1. After selecting the source name, which you want to change for the current file, Once you click on the ‘Change Source’ button at the bottom left corner of the window.
2. Replace the old Source name ( System Identifier ) with the new one and click on the ‘OK’ button.
3. If the new source which has been added is not connected previously in your Power BI Desktop, then after replacing the new source following the steps mentioned above, once you click on the refresh button ( Home ribbon), a pop up window will appear. Because you are connecting this particular data source for the first time in your Power BI Desktop, you need to give the credentials there.

## Edit Permissions

In case of situations where:

1. password has been modified for a particular user or
2. (2) User needs to be changed to connect to a particular source.

The below steps can be followed:

1. In the ‘Data Source Settings’ window ( shown above), select the particular data source, and hit the ‘Edit Permissions’ button.
2. The ‘Edit’ button there in the Edit permission window will allow you to change/modify the credentials for the current data source.

## Clear Permissions

The credentials to connect to a data source can be removed following the below steps:

In the ‘Data Source Settings’ window, select the particular data source and hit the ‘Clear Permissions’ button.

From the drop-down of ‘Clear Permissions’ button, you can choose:

- (1) Clear Permissions: credentials will be removed only for the selected data source
- (2) Clear all permissions: credentials will be removed for all the data sources.

## **Q&A (Natural Language Query)**

Do you know that you can ask questions from your data, not only using programming queries but even through the natural language? Yes, natural means a natural language that you often use while talking about data. Let’s understand with an example. You often ask, “What is your total sale for June?” or “Which product is the top seller?” and so on. Now imagine you are typing such statements, and as an answer, you are getting the stunning visualizations showing the sought information.

Let’s understand this concept by following examples.

The preceding figures are showing the results of two very generic questions, what is the total profit? And what is the total sale? As a result, you get two widgets showing the sought information.

You can use the Q&A feature to find more about your data using the intuitive, NLP capabilities and get the answers back with the help of graphs and charts. Please note that Q&A is different from a search engine capability. Q&A only provides results about the data in Power BI.

## **How to use Power BI Q&A**

Q&A feature is available on Power BI online service and Power BI Mobile. But not on Power BI desktop. The Q&A question box is a text box where you can easily type your question with the help of NLP. Q&A can recognize the typed words and can figure out the dataset to be used to find answers to your questions. Q&A is also capable of forming your question with the help of auto-prediction, textual aids, and restatement.

**Step 1** . Click inside the question box. Now, the Q&A will show a dropdown with helpful suggestions even before you type anything in the question box.

The dropdown list would display the following:

- All the questions which are used in creating the tiles already present in the dashboard.
- The name of the tables present inside the dataset underlying it.

You can type your question and keep on refining it to reach to specific answer that you want an answer for. Alternatively, you can also use the name of the table that can help you to form a completely new question.

**Step 2** . Select from the displayed dropdown, or you can also start typing the question of your own choice of question.

**Step 3** . While you are typing a question, Power BI Q&A selects the most suitable visualization to show your answer; and this visualization would change dynamically as you refine the question, as shown in the following figures.

- The question here is: total units of 2020 by month
- The question here is modified to total units of 2020 by month by manufacturer

**Step 4** . While you type your question, the Q&A feature searches for an answer in any dataset, which is having a tile placed in its dashboard. If each tile is from the Sales and Marketing Sample, the answer is going to come from Sales and Marketing Sample. If there are tiles from Sales and Marketing Sample and Other Dataset, then the Q&A feature will look for the best answer present in both these datasets.

**Please note:** You need to be cautious if only one tile is present in the Sales and Marketing Sample, and by chance, you are removing it from the dashboard, the Q&A feature will not be having any access to Sales and Marketing Sample.

**Step 5.** Once you are satisfied with the overall result, you can now pin the visualization to any dashboard with the help of a pin icon present at the top right corner. If you have not created the dashboard yourself, or it has been shared with you from any other member, you are not allowed to pin the visualization.

**Step 6** . You can also tell Power BI Q&A functionality the type of visualization

you want to view. If you noticed, the preceding graph is a bar chart. If you want a line chart, you just need to add word line after your query, and you will get the visualization changed to a line chart.

# Strategies to Learn Power BI Functions



## Introduction

Here you will see the information about Power BI architecture and how to create, publish, and schedule a Power BI dashboard, how to create parameters, and also limitations of Power BI.

## Power BI Architecture

**Data Sources:** In power BI data sources can be broadly classified into two types cloud data sources and On-premise data sources.

- **Cloud data sources:** Azure SQL Database, Azure SQL Data Warehouse, Azure Blob Storage, Excel files in OneDrive, Online Services as Facebook, Mail Chimp, Salesforce, Google Analytics.
- **On-Premise Data sources:** premise data sources can be further classified into two types as On-premise databases and On-premise files.

**On-premise databases:** Oracle, My SQL, Teradata, SAP HANA, SQL Server Analysis Services(SSAS), Sybase, IBM DB2.

**On-premise files:** Excel, CSV, XML, Text files.

**Power BI Authoring PC:** This authoring machine should have two components installed they are:

- **Power BI Desktop:** It is a windows application which can be downloaded from Microsoft's Power BI portal. By using this application, you can design interactive reports and dashboards. Report files can be saved on local disk with .pbix extension. The reports and datasets can be shared in the Power BI portal.
- **Power BI Personal Gateway:** Needs to be installed separately on

authoring PC along with Power BI Desktop and is a bridge providing quick and secure transfer of data in between the Power BI portal and on-premises data sources. Available only with Power BI Pro Subscription. Azure Service Bus provides security for the data transferred between the gateway and the Power Bi Portal, providing a secure channel between your computer and the Power Bi service. This secure channel means that there is no need for you to open any ports in the computer firewall.

**Power BI Portal:** It is where users publish their datasets and reports, create dashboards and content packs, and share data with team members of their organization. Users can sign in to the portal using their organization account. Each user will have a personal workspace in the portal named My Workspace, which comprises dashboards, reports, and datasets published by the user.

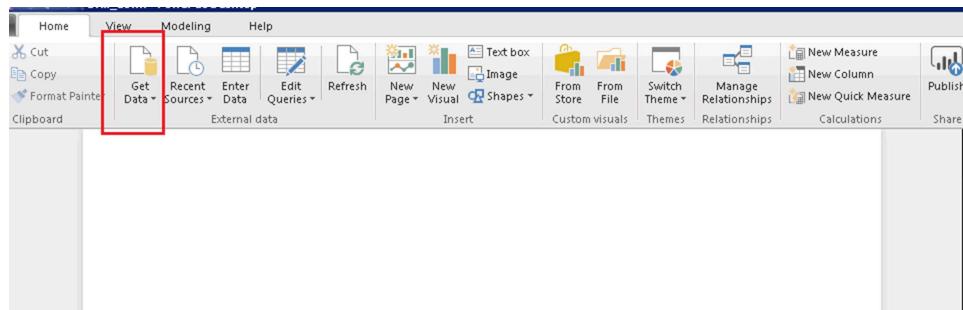
**End Users:** Dashboards can be shared via email with multiple stakeholders. Power BI offers a free mobile app for IOS, Android, and Windows devices. Reports and dashboards automatically adjust their sizes to fit the screen of the device, so users need not worry about creating mobile versions of the work. Users can interact with reports and dashboards through filters, slicers, drilling, etc.

## Creating, Publishing and Scheduling a Dashboard

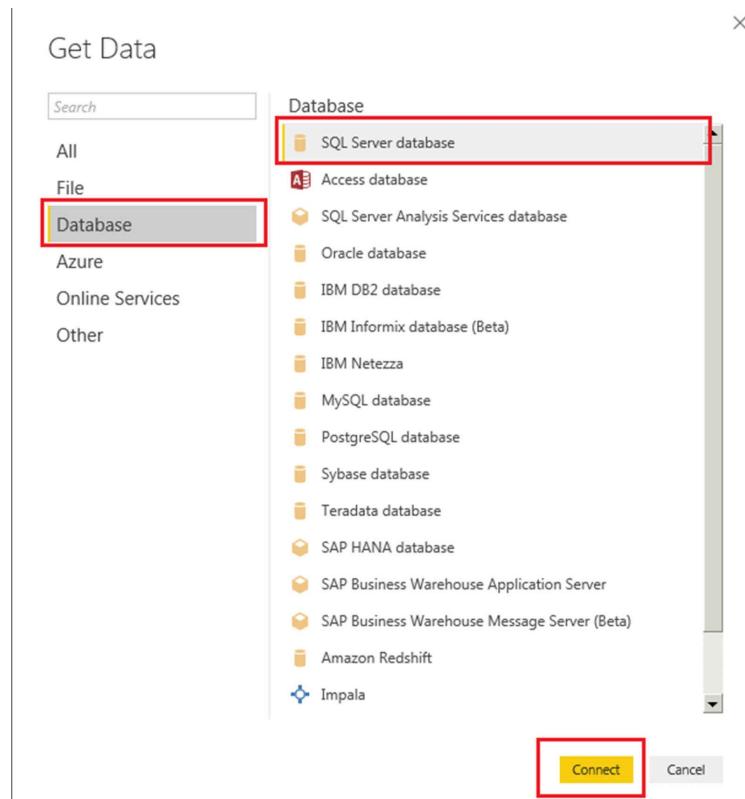
### Creating a Dashboard

#### Importing data source

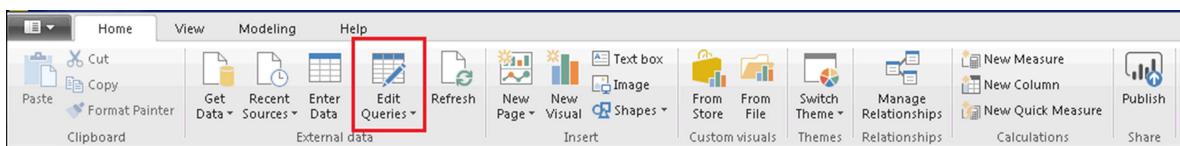
- AdventureWorks database needs to be imported from the Microsoft SQL Server. Click on get data icon in the home bar.



- Go to the database and select the SQL Server database and click on connect.



- Now mention the server name and click on ok.
- Select the AdventureWorks database and check the boxes for the tables that need to be imported and click on load.
- After importing tables will be shown in the fields pane.
- Click on the edit queries icon in the home bar to view the data for all the tables.



- A new edit queries window will open, which will show data for all tables where a lot of transformations can be done on the data using the edit Queries window. Click on close and apply after changes have been applied.

## Designing the Dashboard

Below are the requirements that should be achieved while designing a dashboard

**Requirement-1:** To display overall total sales of the company.

**Requirement-2:** To create drill down on SalesTerritoryGroup> SalesTerritoryCountry> SalesTerritoryRegion and compare total sales.

**Requirement-3:** To show sales for product subcategories in the selected product category for both internet and reseller sales.

**Requirement-4:** Locate countries and their respective sales on a geographical map.

**Requirement-5:** Create a filter on the sales territory group.

Now let us understand each requirement one by one.

**Requirement-1:** To display the overall total sales of the company.

- To fulfill this requirement, card visualization can be used.
- Drag and drop the card visualization from the visualization pane.
- Right-click on fact internet sales and select a new measure.
- In the measure window, type the name of the measure as **TotalSalesbyDAX**.
- Type in the DAX query as:

TotalSalesbyDAX = SUM(FactInternetSales[SalesAmount]) +  
SUM(FactResellerSales[SalesAmount])

- Now drag the measure into value field of the card. And do some formatting by using the format pane of card visualization.

**Requirement-2:** To create drill down on SalesTerritoryGroup> SalesTerritoryCountry> SalesTerritoryRegion and compare total sales.

- To fulfill the requirement, a bar chart can be used.
- Select the **Bar** chart from the visualization panel.
- Expand the table DimSalesTerritory and select the SalesTerritoryGroup column and drop it on the **Axis** section of the Fields pane.
- To create the hierarchy, drag the SalesTerritoryCountry column and drop it below SalesTerritoryGroup in the Axis section.

- Also, drag the SalesTerritoryRegion column and drop it below SalesTerritoryCountry. This creates the hierarchy.
- In the value section, drag the measure TotalSalesbyDAX.
- Now some formatting is required for the bar chart, so maximize the chart by clicking on the icon.
- Go to format pane and enable the data label property now the values will appear and change the font color, size, background color of data labels.
- As hierarchy got created, data can be drill down from SalesTerritoryGroup> SalesTerritoryCountry> SalesTerritoryRegion. Right-click on the bar and click on drill down or use the drill down arrows, which are appearing at the top right corner.
- On drilling down Europe, total sales with respect to countries in Europe are shown.
- And this creates drill down on SalesTerritoryGroup> SalesTerritoryCountry> SalesTerritoryRegion and compares total sales.

**Requirement 3:** To show sales for the product subcategories in the selected product category for both internet and reseller sales.

### Steps to Create a treemap for Internet Sales

- To fulfill this requirement, a treemap can be used.
- Select a treemap from the visualization panel.
- To build the hierarchy and enable drill-down, drag, and drop.
- EnglishProductCategoryName and EnglishProductSubCategoryName columns from DimProductCategory and DimProductSubCategory tables, respectively, on **group** section of the Fields pane.
- Drag and drop SalesAmount column from FactInternetSales table in the **Value** section.
- In this tree, chart drill-down is created from EnglishProductCategoryName> EnglishProductSubCategoryName so right-click

EnglishProductCategoryName> EnglishProductSubCategoryName so right-click

on bikes and do drill down the sales for different subcategories of bikes can be seen.

### **Steps to Create a treemap for Reseller Sales**

- Repeat the steps 1-2, as mentioned in the above chart.
- Select the **SalesAmount** column from the **FactResellerSales** table instead of the FactInternetSales table while following the third step.

This creates treemaps that show the sales made through the internet and reseller in various product categories.

**Requirement 4:** Locate countries and their respective sales on a geographical map.

### **Steps to Create a Map Visualization**

To fulfill this requirement, map visualization can be used.

- Select Map from Visualization pane.
- To enable the drill-down feature on Map drag and drop SalesTerritoryGroup and SalesTerritoryCountry columns from the DimSalesTerritory table in the location section of Fields pane.
- In the size section, drag the measure TotalSalesbyDAX.
- Formatting can be done by changing the bubble colors using the data color option and increase the bubble size by using the bubble option.
- As the drill-down feature is enabled from SalesTerritoryGroup> SalesTerritoryCountry. On drilling down, European sales made by the individual country in Europe can be seen.

**Requirement 5:** Create a filter on the sales territory group.

## Steps to Add a Slicer

- To fulfill this requirement, the slicer can be used.
- Select slicer from the Visualizations pane.
- Select the DimSalesTerritory table and drag and drop the column SalesTerritoryGroup on to **Field** section. This adds the data of the SalesTerritoryGroup column in the slicer.
- Go to format pane of the slicer to enable multi selection to disable the single select in selection controls. If the select all option is required, it can be enabled. And select either list or dropdown of slicer by clicking down arrow at the top left corner.
- Slicer on SalesTerritoryGroup is created.
- Done with all the requirements of creating a dashboard.

## Publishing a Dashboard

- Click on the publish icon, which is on the top right corner.
- A sign-in dialogue box will appear where the Power Bi account email id needs to be entered.
- Give a password and click on sign in.
- Select my workspace.
- A dialogue box will appear, saying that the report is published. Click on the report name; it will navigate to the Power BI portal.
- Now the report in the portal will appear. Click on the icon pin live page to create a dashboard for the report.
- Give the name of the dashboard and click on pin live.
- The dashboard view can be switched between the web view and phone view.

## Scheduling a Dashboard

- Click on subscribe to receive the email.
- A dialogue box will appear, saying that an email will be sent whenever the data is updated. Click on save and close.
- Now data refresh for AdventureWorks needs to be scheduled. Go to AdventureWorks and select a scheduled refresh.
- Install power bi personal gateway if the data source is on-premise and mention the gateway connections and enter the credentials of the data source.
- Data refresh can be done daily or weekly and set time zone and mention at what time the data needs to be refreshed if there is a failure in data refresh a notification mail will be sent. Click on apply .
- As a data refresh is scheduled whenever the refresh happens, the subscription will trigger an email to us.

## Creating Parameters in Power BI

**Requirement:** Create parameter on sales territory group

- Go to edit queries in the home bar.
- Go to DimSalesTerritory table right-click on column SalesTerritoryGroup and select add as a new query.
- Now the column SalesTerritoryGroup will be considered as a new table. Right-click on the column and select remove duplicates as you need distinct SalesTerritoryGroup.
- Now SalesTerritoryGroup Query needs to be passed as parameter.
- Click on manage parameters.
- A parameter dialogue box will appear where the name, description, and data type of parameter need to be mentioned. If the suggested value

option is any value, only one value can be passed if it is a list of values multiple values can be passed in the current scenario query should be passed as parameter. A select query is suggested values, and in query option select SalesTerritoryGroup\_query and the current value, you can mention any among the list of values.

In the current scenario, Europe is taken.

- Click on Ok. Europe will appear as the current value.
- This parameter needs to be applied to the table DimSalesTerritory table. Go to DimSalesTerritory table click on down arrow of SalesTerritoryGroup column select text filters click on equals.
- Filter rows dialogue box will appear—select parameter.
- Now automatically, the parameter will be mapped. Click on OK.
- Now the query data will be filtered on Europe as Europe is mentioned default parameter.
- Click on close and apply it to the query editor. Export the report to Power BI templet. The report will be saved in .pbix format.
- When the templet file is opened, it prompts for the Sales Territory Group where the value should be selected. North America is selected as a filter and click on load then all the data related to North America will be loaded in the report.
- Data related to North America is loaded.
- The difference between .pbix and .pbix files is that a .pbix file will have the data for all the queries, but a .pbix file has only schemas initially. It doesn't have any data, and data will be loaded only after selecting the parameter; this is the reason .pbix file has less size when compared to .pbix.

## **Calling Oracle Package Function in Power BI**

Packages that have functions and cursors can be created in oracle, and when that particular package and function is called, it gives the data. In power BI w oracle packages can be called, which will load the data.

**Requirement:** create an oracle package on the customer table when 'M' is passed as an input value. It should show the male customers details as output, and if 'F' is passed, it should show the female customers details as output.

The package code looks like below:

```
CREATE OR REPLACE
PACKAGE customer
AS
CURSOR c_tr_customer (p_gender    string default null)
is
select * from "customer" where "Gender"=(p_gender);
TYPE customer_type IS TABLE OF c_tr_customer%ROWTYPE;
FUNCTION Fn_customer (p_gender        string default null) RETURN
customer_type PIPELINED ;
END customer
;
/
```

```
CREATE OR REPLACE
PACKAGE BODY customer as
FUNCTION Fn_customer (p_gender        string default null) RETURN
customer_type pipelined AS
v_c_tr_type customer_type;
v_count    PLS_INTEGER := 0;
BEGIN
FOR r IN c_tr_customer(p_gender)
```

```

LOOP
    v_c_tr_type := customer_type(r);
    PIPE ROW(r);
    v_count := v_count + 1;
END LOOP;

RETURN ;

```

END Fn\_customer;

END customer;

/

- When package function is called by passing ‘M’ as input, male customer details were shown as output.
- Click on get data in Power BI go to databases select oracle database and click on connect. Give the server name and write the query in the SQL statement.
- ‘M’ is passed as input male customer details will be loaded.

This is how you can call oracle packages in power BI.

## **Limitations in Power BI**

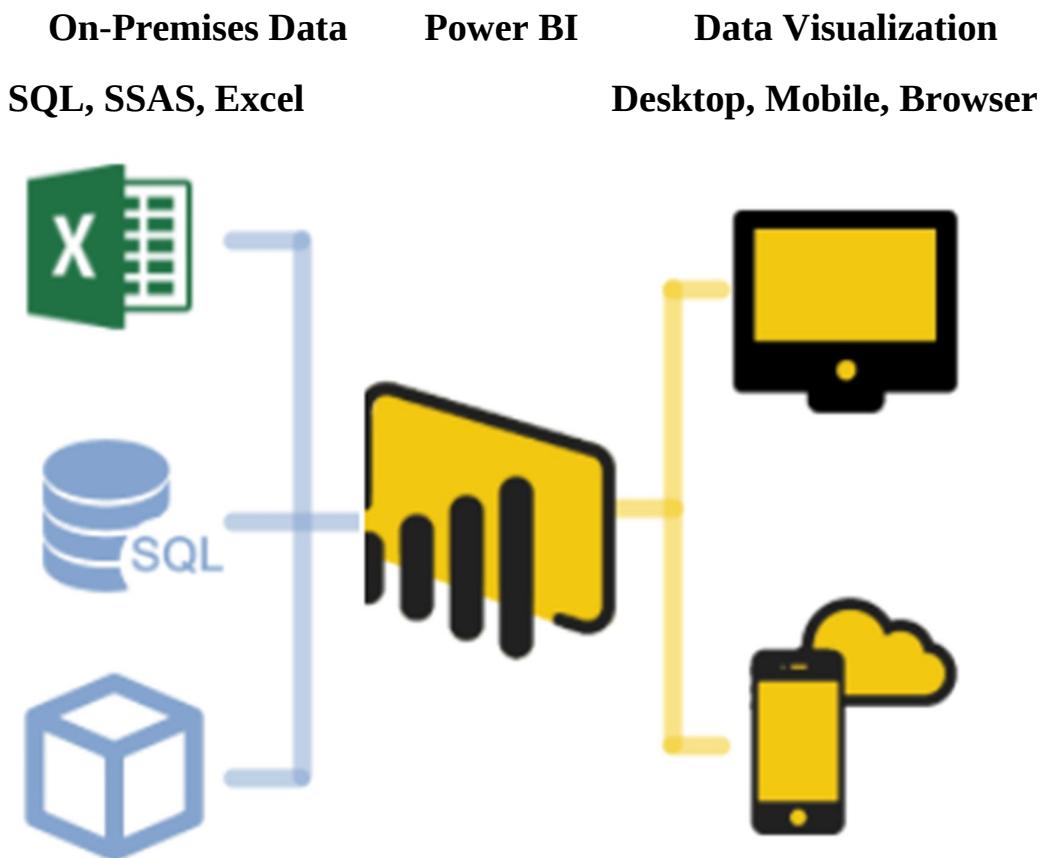
- In Power BI, the size of the table does not increase or decrease relatively along with the data, and when data is more scroll bar should be used to scroll down, and when the data is less, the empty space will pop up in the table .
- .Pbit file can't be uploaded in the Power BI portal you can only upload .Pbix files where the .Pbit file prompts for a parameter so you cannot upload reports with parameters in the portal.
- There is no visibility property in Power BI, like in SSRS, to hide the visualization depending on the condition given.

- Conditional formatting can't be applied to fields when the data type is other than integers. The conditional formatting pane has only employee key, so conditional formatting can be applied only in numerical fields.
- Dashboards and reports can only be shared with users who have the same email domains or email domains listed with Office 365 tenant.
- Microsoft Power BI is still in the early days. The tool will be getting updates every month by considering the requests from the users.
- Scheduling of reports in power BI portal is not as flexible as Scheduling SSRS reports using share point.

# Power BI Desktop Functionalities



In this chapter, you will understand the basic and detailed concepts of Power BI Desktop implementation with clear examples. Also, you will see the advantage of Power BI Desktop in data visualization.



## Power BI Desktop Introduction

Microsoft provides **Power BI** as a free service for business analytics, providing

fully interactive visualizations, allowing end-users to create dashboards and reports by themselves without depending on anyone.

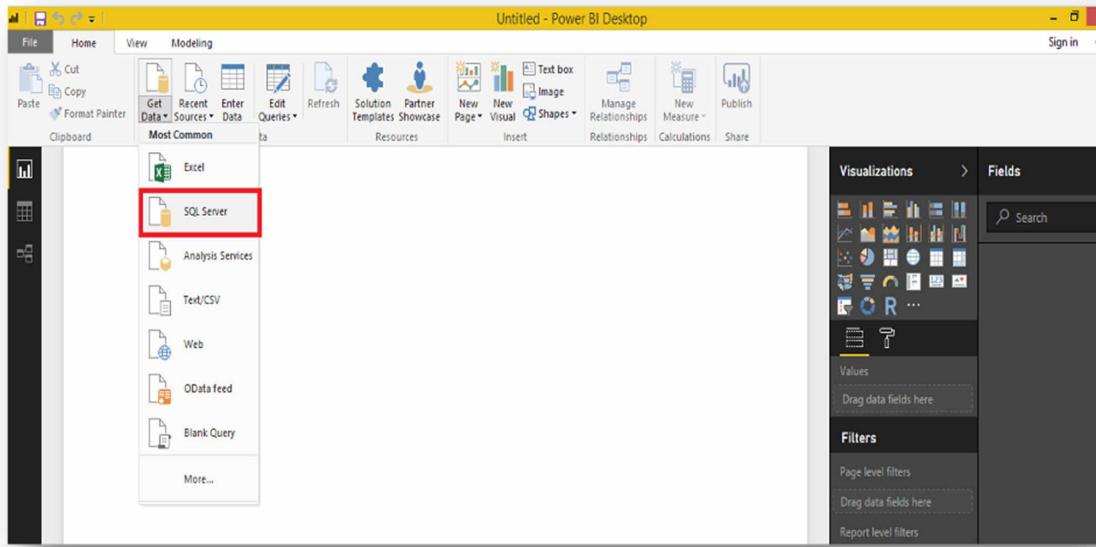
**Power BI Desktop** is a free desktop application that provides report creation & generation, modeling, and advanced query features, which helps you to create reports, data models, and collaborating with team members by sharing your work with each other.

You can share the work in two ways:

- 1) **Power BI desktop file ( .pbix/.pbit )** : Share the file to the user; if the user wants to see your report, they need to install Power BI desktop in their system.
  - .**pbix**- ( Power BI Desktop File ) Report with static output.
  - .**pbit**- ( Power BI Desktop Template ) Report should accept the changes dynamically and display the output based on the input parameters.
- 2) **Power BI service** : Publishing the PBI desktop file directly to Power BI Service, so that users can see the report and interact with it from the web browser. You need a license to connect the Power BI service.

### **Starting Power BI Desktop**

1. The first step is to launch the Power BI Desktop.
2. Connect the SQL Server objects to get the data loaded into PBI Desktop.



3. If you are the admin of the server, then connect the SQL Server using “localhost\SQLEXPRESS” in my Local Server Instance.
4. If you want to connect the Generic server with Username and password, then connect the SQL Server by providing the Username and Password.

## Data Connectivity Modes

### Import

It will import all the selected tables/columns into the Power BI Desktop.

Whenever a user wants to create/interact with a visualization, the ‘Import’ won’t pull the latest data from DB; if you want to see any changes to the underlying data, then you need to do a refresh every time in Power BI desktop.

See the detailed example mentioned below by loading the “TrendData” table.

Column_name	Type
WorkYear	Int
Workweek	Nvarchar
USA	Int
SNO	Int

**File Name = Trend.pbix**

Slicer Tool will act as a filter tool. You have two-year values (2019 & 2020).

In the example, I selected the value 2019 from the slicer tool, which is coming from Query1 Dataset. So, the designed report will also show only 2019 year values information.

Add a new value to the Trend Data table to see whether it is reflecting in the Power BI Report or not.

```
SQL> Insert into [dbo].[TrendData] values(2019, 'SNO4', 140, 138, 123, 251, 106)
```

Close the Power BI Desktop file Trend.pbix and open again; you won't see the newly inserted 'WW07' values in the Power BI Desktop until you do a "Refresh."

## **Direct Query**

It won't copy/import any data to the Power BI Desktop.

Whenever a user wants to create/interact with a visualization, 'Direct Query' queries, only the underlying data from the DB. Meaning always you will view the current data.

## **File Name = Trend\_Direct.pbix**

```
SQL> Insert into [dbo].[TrendData] values(2020, 'SNO5', 145, 213, 126, 174, 148)
```

Close the Power BI Desktop file Trend\_Direct.pbix and open again, you can see the newly inserted 'WW07' 2018 values in the Power BI Desktop without a Refresh because Direct Query will always give us the current data from DB.

1. Suppose instead of pulling data from a single table; if you want to join with some other table to get the result set, then you need to right-click on the Dataset 'Query1' and select the 'Edit Query' option highlighted in the screen.
2. Double click on the Source option highlighted on the screen.
3. Database server connection popup will open, and you can modify the query by joining with other tables and put the statement again in the 'SQL Statement' window highlighted on the screen.

**Note:** While modifying the query statement, think you have already designed the Power BI Desktop with some attributes, make sure that the same attribute names must come again along with other new additional attributes (if any). Otherwise, you will lose the design which you had already designed before.

Example: You designed the Power BI Desktop using (Work Year, Workweek, USA, SNO) attributes while modifying the query to make sure these attributes must come in the new query also otherwise, you will lose design.

## Pbit Template File Implementation

All the above explained the **Pbix** extension file would give only the static result. Suppose if you want to see the data based on your input parameters (Date, text, Number, etc.) in Power BI reports, you need to use the **Pbit** template file.

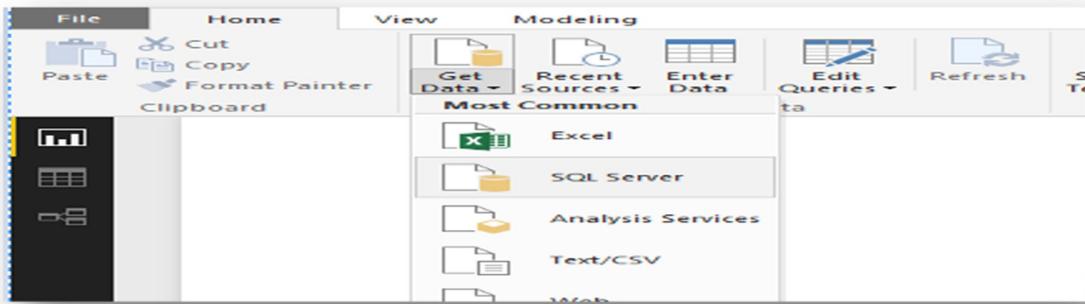
### Parameters Syntax

Date Parameter = ""& Date.ToString(Column\_Name) & ""

Text Parameter = ""& ToString(Column\_Name) & ""

Number Parameter = " & ToString(Column\_Name) &"

1. You are going to get the Missed Records status report for all the GEOS using **Pbit** template file.
2. I have maintained three tables to load the missed records data counts every weekly from 2019 to till now from the daily transaction OLTP.
3. Now you are going to create a Self-service Analytical Power BI report (pbit) to allow business users to leverage analytics.
4. Open the Power BI Desktop and save the file in the below-mentioned format (\*.pbit).
5. Get the data loaded from SQL Server by following the below step.



1. Load the table data with hardcoded parameter value into Power BI desktop.
2. Data Loaded into the Power BI Desktop with dataset name ‘Query1’.

## Create Parameters

Add Parameters in the Power BI desktop to pass the dynamic parameter value while loading the data into Power BI Desktop.

1. Right-click on the ‘Query1’ and select the ‘Edit Query’ option.
2. Select Manage Parameter drop-down and click on the ‘New Parameter’ option.

Now, create two Parameter with the names “Year” and “Workweek.”

- Year Parameter Creation.
- WorkWeek Parameter Creation.

After the creation of two parameters, click on the “Close & Apply” option to save.

## Assigning Parameter Values Dynamically

Change the hardcoded Parameter values to the created Parameter names to get only the specific data loaded into Power BI Desktop based on the I/P Parameters.

1. Right-click on the ‘Query1’ and select the ‘Edit Query’ option.
2. Click on the drop-down arrow nearby Query Settings.
3. You will get the hardcoded value for YearName and Week.

4. Now try to replace the hardcoded values with the created parameters (Year & WorkWeek). The tick mark(Parse) will tell you whether your modified query is correct or not.
5. For better understanding, you can rename the ‘Query1’ dataset to whichever name you want. “MissedRecords table loaded into PBI with dynamic parameters.”
6. You need to follow the same procedure which you did for the 1<sup>st</sup> table dataset ”MissedRecord” to load the 2<sup>nd</sup> table “Missed Percentage” by passing the dynamically created parameter values.
7. For the 3<sup>rd</sup> table, I don’t want to pass any parameters; instead, I want to load all the data. Creating visualization for the Loaded DataSet. Select whichever objects needed from the visualization menu.
8. Select whichever data fields which you want the user to see in the Power BI Report.
9. I have pulled data fields from two Dataset “MissedRecords” & MissedPercentage” and created a tabular view of data.

Once you have completed the above steps, you can start the below ones.

1. For formatting the text and good design, click on the red circle highlighted to see the available formatting option to alter the text format and design format.
2. You can also change the Data Type of any attribute, for example, click on the Missed\_Record field and click on the Modeling menu to change the data type.
3. Create a new tab to create the Missed records Trend using Visualization in PBI.
4. As I told before, I haven’t set up any dynamic parameter value for this Dataset MissedTrendData. Then pull all the data from this table from SQL DB.
5. Copy the created design and paste on the same tab.
6. Click on the pasted design and select the Line graph, then your design will change. Click on the yellow color highlighted option to change the

level of data to the lowest with an ordered format.

7. Change the Y-axis scale type to ‘Log’ to arrange the data in proper order. Enable the ‘Data Label’ Option to see the values for the corresponding Year & Work Week.
8. You can filter the data by using the slicer tool ‘Year Name,’ then the report will also reflect based on the selected value. Save and close the Power BI desktop .
9. Reopen the Power BI Desktop; it will ask for the parameter values.

Now, data got loaded, and you can see the same tabs.

- 1<sup>st</sup> Tab - You set dynamic parameter values for the two datasets “MissedRecords” & “MissedPercentage.”
- 2<sup>nd</sup> Tab - You loaded the “MissedTrend Data” dataset without setting up any dynamic parameters.

For testing purposes, I have pulled Week, and YearName attributes from the loaded “MissedRecords” dataset to verify the data loaded only for the selected Parameters.

### **Publishing Power BI Desktop to the Web (Power BI Service)**

1. Click on the Publish tool.
2. Sign in to Power BI Login.
3. Select Destination place where to save the Power BI report.
4. Once the above step is done, you will receive a link to your report.
5. Click on the link to open the report in your Power BI publishing site.

### **Minimum System Requirements to Run Power BI Desktop**

Windows 7 / Windows Server 2008 R2, or later.

.NET Framework 4.5 or later.

### **Maps in Power BI Desktop**

Here you will understand how to connect to the data sources, create dashboards,

availability of different kinds of visualizations, especially map visualizations and their usage in Power BI Desktop. Now let us get started.

## **Creating a Dashboard**

Open Power BI Desktop, it will show the home screen.

Then click on Get Data, it shows the following screen showing a wide range of source connections.

Select the required option. Here I am selecting the Excel. Click on connect and browse the path where the file is located. Select the sheet which you want to load and use. Click on Edit if you need to change the dataset like adding a calculated column, deleting the unwanted column, etc. else click on Load.

Then it will load the data. To check the data loaded, click on the icon Data on the top left side on the screen.

Now you will develop dashboards using map visualizations available.

Note: If the visualization is not available in the list available, you can download and import it by clicking on the three dots (...)

## **Dashboard Using Map**

For this, click on the empty or white screen then click on Map under the visualizations tab.

Then check the country and sales columns available under the Country\_vs\_Sales dataset. Maximize the map.

Now you will try to include other visualizations like a bar chart, a pie chart for a better understanding. For this also, click on the empty white screen available, then click on the desired visualization, then drag the fields to the desired appropriate values or check the columns.

And then, by clicking on any of the countries, you can filter only for that particular country value throughout the dashboard.

Suppose I clicked on the value New Zealand, then the total dashboard will highlight the details of New Zealand only.

Give the proper name to the dashboard and save the file.

Based on the requirements, you can use different visualizations on the dashboard.

The above dashboard consists of the **Visualization Map**. Now you will try to implement different kinds of maps available.

### **Filled Map**

Now select **Filled Map** under Visualizations tab, then check the boxes against country and sales columns.

You can also go for the individual coloring for each country by dragging the country column to legend.

You can also specify the color saturation from one color to another, depending on the values in the data.

E.g., it will display the colors from red to green, i.e., red indicate lower values and green indicate higher values.

You can also include here one more color as the center value by setting the value of Diverging to On. The values for the Minimum, Center, and Maximum are taken by default. You can also explicitly mention the range of these values according to your needs.

If you give the consecutive values to them, they display the exact colors with no saturation.

And if consecutive values are not given to them, they display the saturated colors.

### **Shape Map**

You use a shape map to achieve your own desired map or when you want only to specify particular regions of the map. For this, you can select the maps available under it. Or if the map which you are looking for is not available, you can create one yourself and import it. This file will be in .json format.

.g.: You shall compare the sales among the different countries of the African continent.

Now click on the shape map under visualizations.

If it is not available, enable Shape Map by navigating to File -> Options and Settings.

Then go to Options -> Preview Features.

Finally, select the Shape Map, and you have to restart the Power BI Desktop once you finalize the selection.

Now develop a map file in JSON(JavaScript Object Notation) format. (You can get from internet of desired geographical regions). I created here the countries of the African continent.

As soon as you enable the Shape Map, click on Shape Map control present in the Visualizations pane. Check a few columns and go to the Format tab, where you can see the predefined groups of countries under the Shape tab.

You can select one among the given list, or you import the map file by clicking on the tab +Add Map.

Now it will direct to browse the file option, then select the file and import.

You are allowed to use the custom maps along with Shape Map only if their format is in TopoJSON. If the map is not in this format, use the online converters like Map Shaper (<http://mapshaper.org>) to convert it to TopoJSON format. Generally, the JSON files downloaded from the internet should be converted to TopoJSON format using Map Shaper.

Now the map of desired regions is visible. You can compare the sales values among these countries by having different colors for each of the sales value by having Sales in legend.

# **Power BI Real-Time Dashboard**



## **Introduction**

This chapter provides details about the process required to implement Real-Time Dashboards using the Power BI Preview feature of Office 365. It describes best practices for building a business intelligence (BI) environment by using a combination of Microsoft Power BI, Microsoft Office 365(O365), and on-premises data sources.

This chapter will help you understand designing and sharing a real-time dashboard. It will make you aware of its functionalities and benefits as well. It also provides you with information about how you can connect to various data sources to fetch your data and use the same for building your dashboards.

To start with, you should have the data, or Power BI reports that will help to build your dashboard in some data sources like Excel Workbook, SSAS, GitHub, etc. You should also have some basic knowledge on how to create Power BI reports, though it is not mandatory. It will help you to learn how to create Power BI reports and will help you to create real-time dashboards having tiles made of visualization (like charts, map, combo charts, graphs, etc.)

## **Power BI Preview**

Microsoft Power BI dashboard helps you to stay up to date with the information that is important to you. Your dashboards have tiles that lead to the reports which you can explore on just a single click. You can bring all your relevant data together in one place by connecting to multiple datasets.

## **Reasons for Creating the Dashboards**

- To see all the information needed to make decisions in one place and one look.
- To increase efficiency and support the decisions based on the most important facts.

- To ensure all the colleague's views and use the same information.
- To help the business determine goals and strategies and monitor the health of a business.
- To allow you to identify and correct negative trends using metrics and trends that matter a lot for business.

When you first open Power BI Preview, you'll see:

- Navigation pane
- Dashboard with tiles
- Q&A question box
- Help and feedback buttons
- Dashboard title
- Home button
- Version

## **Navigation Pane**

The three objects of Power BI Preview include dashboards, datasets, and reports. You can use the Navigation pane to explore them. You can have no data in dashboards and reports, but that is not useful until you have the data with you. So for creating dashboards, you need to have data.

## **Datasets**

You need to have data for *dashboards*, and for getting that, you can use *datasets* to connect to various data sources. After gathering all your data in one place, you can start by creating reports or dashboards. You can even bring your already created reports using *datasets*.

In the navigation pane, you can find all the datasets that you have connected to present in the heading Datasets. Each dataset listed is only having a single data source. Let's say that you have an Excel sheet in OneDrive, your computer, or any of your on-premises datasets, Salesforce database, etc.

ONE dataset is useful in creating many reports, or you can say the same dataset

is used many times in different reports. You can pin as many visualizations from your dataset to any number of the dashboard.

To connect to a Dataset, you can either click GetData from the top of Navigation Pane or Click on  next to Datasets.

In some cases,

- When Power BI Preview imports a copy of the dataset, then, in that case, the changes you make won't affect your original dataset since it is a copy of your dataset.
- When Power BI Preview connects to your dataset like if the workbook was imported from OneDrive for Business, then you can refresh your dataset with the latest data from the workbook on OneDrive.

## **Dashboards**

A dashboard is a visualization tool that shows you the current status of metrics, graphs, charts, KPI's, etc. related to business on a single screen for an enterprise. It is created by the co-workers and shared with the concerned members in an organization to stay up to date. Dashboards consolidate tiles. Each tile contains visualization created from the underlying data in your datasets.

You can find your dashboard listed in Navigation Pane under Dashboards heading.

ONE dashboard can show visualizations from many different datasets, and the same goes for reports as well. After creating a dashboard, you can share it within your team as well.

You can even import your dashboards from other SAAS services with the dataset like the Get Data popup for Salesforce has an option of getting dashboard and/or report to be created from your dataset.

## **Reports**

The Power BI Preview report contains page(s) of visualizations. You can create reports from scratch using the PowerBI Preview itself, or you can also import them using datasets. Once you add a dataset, it automatically gets added under the reports heading in Navigation Pane, but only if your data is formatted properly, and it has visualization in Power View sheets and with shared

dashboards. For example, if you are connecting to the Microsoft Excel workbook, which has the Power View sheets present in it, then the Power BI Preview will automatically create a new report as per the visualization present in those sheets. You can even create Reports based on the data you have using Excel tool-kit like Power Query (Discover and Connects to data), Power Pivot (Transform and models data), Power View (Create Visualization (charts & graphs))

You can find your reports in the navigation pane listed under the Reports heading. Each listed report has a page(s) of visualization. You have two ways to view as well as interact with reports. They are:

**Reading View:** There is no need to worry using this view as you cannot modify or update anything. All you can do is explore and browse the data and visualization in the report, and temporarily pin any of the visualizations you want in your dashboard. (The filters applied while interacting with your reports also won't be saved, and neither the pinned tiles will be saved when you close and reopen Power BI Preview).

**NOTE:** To edit the report and save your changes, you have to open it in Editing View.

**Editing View:** As compared to Reading View, In editing View, you can look into your data by creating new visualizations, changing visualization type, adding and deleting visualizations, adding and removing fields, and adding pages from the report.

**NOTE:** To edit a report, you must be the owner of the report

**NOTE:** If a dashboard has been shared with you, then you won't be able to see the report in the navigation pane. You will be able to open the report only if the report owner pinned that particular tile from the reports or otherwise if the owner created from the Q&A, then the Q&A page opens.

ONE report can be used in multiple dashboards. Tiles selected from the different reports can appear on the same dashboard. ONE report can be created from only one dataset. There is an *exception* to this - you can use Power BI Designer, which is capable of combining more than one dataset, to build a report. There is also an option in GetData 'Power BI Designer File' to import a copy of it and build your dashboard.

## Dashboard Tiles

Dashboards are made up of tiles containing visualization. They are created either in report Editing View or Q&A. The owner of the report pins the tiles that appear on a dashboard.

You have now created the dashboard.

## Benefits

- Shows the holistic summary view for all the metrics in the current fiscal year.
- Real-time status of Product Launches launched and Hot-fix information.
- Single view to show everything that's needed for the business.

### Information on Dashboard

**Product Launches:** The Product Launches metrics show the Standard Product Launches count and the Non-Standard Product Launches, count.

**Quality:** The Quality metrics will show the data for both Production defects and Hot-Fixes. The Fields shown under this category are below:

- No. of Production Defects
- Standard Defects & Non-Standard Defects
- Bad Hot-Fixes & Good Hot-Fixes

**Drill-down views:** Drilldown views for each of the categories are shown in the report.

- **Product Launches Drilldown:** You can show the Product Launches in different dimensions. Product Launches against Business Groups, Product launches against Launch lead, and Product Launches against Functional Areas.
- **Quality Drilldown:** You are showing the quality metrics in multiple dimensions based on the need. You can show the defects against the functional Area, Defects against launch type, and Defects against the environment—Hot-Fixes against good or Bad Hot-fixes, hotfixes.

## **Q&A Question Box**

In Power BI, the Q&A uses Natural Language as a query being able to answer a question in the form of charts, graphs, metrics, etc. in the browser within a second. The visualizations are all dynamic and interactive. Any change in your query will automatically alter the results accordingly. This helps in creating the reports or getting calculations while giving presentations instantly. You can modify the reports as per your requirement.

You can add content to your dashboards only in the form of tiles. Q&A helps them who are new to create Power View reports from the basic level. When you write a query in the natural language, it looks for an answer in the dataset(s) connected to the dashboard.

When you start typing your question, you are directed to the Q&A page. As you type your question, it gives you the options to ask the right question with the help of IntelliSense, suggestions, and finds the best answer in the form of visualization. Once you have the expected visualization, you can pin it to your dashboard.

And Work Request according to the State, which gives us information on How many work Request is closed or pending, which is very useful for business.

## **Help and Feedback Buttons**

The icons in the top right corner contain a Download menu containing the direct link for downloading the resources, getting help for any problem that you face, and providing feedback to the Power BI team. It also contains the support link for learning all about Power BI preview in a very detailed manner.

## **Dashboard Title**

It's difficult to figure out which dashboard is active, always. The dashboard title appears in the following places:

- Report Editing View
- Dashboard view page
- Q&A page
- Report Reading View and when you open a dataset.

This is the way to know which dashboard you're pinning your visualization into.

## **Home Button**

To return to your dashboard, click on the home button that appears on the top left corner of all screens in Power BI Preview. You may also return to the navigation pane using this button.

## **The Retail Analysis Sample Dashboard**

The first time you visit <https://app.powerbi.com> to open Power BI Preview, you will find a sample dashboard, i.e., **Retail Analysis Sample**. Unless you manually remove this sample, it will always be there. You will not see any reports under Reports heading even if the sample dashboard because of the earlier mentioned reason that it is shared with you. You will only be able to view the report through tiles if the owner pinned the visualization from the report. You can explore and learn from this sample until you remove it.

## **Functionalities of Power BI Preview**

### **Delete a Dashboard**

If you Delete or remove a dashboard, that does not mean that any reports or datasets that are associated with it will also be deleted. You have to delete them separately.

- If you have created the dashboard on your own, then only you will be able to delete it. Once you have shared it with someone else, then deleting from your Power BI Preview won't delete or remove from their Power BI Preview.
- You can remove only those dashboard which is shared with you. This doesn't mean that the same will be removed from other people's Power BI Preview.

### **How to Remove a Dashboard**

1. Navigate to the navigation pane, then right-click on the dashboard to delete it.
2. Click **Yes**. The dashboard will be removed.

You can identify a shared dashboard with you, by lock icon prefixed to the dashboard name. To delete a dashboard, click **Delete**, following the same steps as removing.

## Create an Empty Dashboard

1. Click the plus sign that appears against the **Dashboards** heading on the navigation pane.
2. Give a name for your new dashboard.

Your dashboard will be blank until you get some data. You can get data from either of these options mentioned below:

- Pin the answer(Visualization) from Q&A
- Pin visualization from Reports

## How to Pin Visualizations from Q&A

In the question box type, what you want to know about data. For example:

To make the metric ‘Bug Count’ display on your dashboard as a tile, click on the Pin Icon. When you click it. The above-shown visualization will be displayed as a tile in your newly created dashboard.

## How to Pin Visualizations from Report

1. Open the report from which you want to pin the visualization from in the Navigation Pane.
2. When you open the report, it opens in the reading view. You can pin a tile from both the Reading View and Editing View. In the reading view, you can either interact with the visualization and filter but cannot edit the report.
3. The report you opened has one or more visualization. Click the visualization you want to pin and click the pin icon. A Success message will be shown in the top right corner that tells you the visualization has been added to your dashboard.
4. Click the Home button to see the changes to your dashboard.
5. Now you can customize (rename, resize and remove) the tile in your

dashboard.

## **Customizing Your Tile**

Rename the tile : To edit the tile title, click the pencil icon by hovering over a dashboard tile.

Move the tile : Hold and drag your tile to a new location on the dashboard.

Resize the tile : The tiles can be re-sized to small, medium, or large. To resize the tile, click and drag the arrow in the bottom right corner.

Delete the tile : Click this icon to delete the tile. Deleting a tile does not delete the related report or dataset.

## **Share a Power BI Dashboard**

Now you have the dashboard ready. As I mentioned earlier, you can share the reports amongst co-workers. Let's see the steps on how to do it.

You can share the Dashboards with your colleagues by simply clicking the Share icon and specifying their email addresses.

1. Click the Share Icon after you open the dashboard that you want to share.
2. When you click on the **Share** icon, a dialog box appears. Click on **Invite** and enter the co-worker's email ID and the description of the dashboard on the box below. If you want your co-workers to share your dashboards with others, then check the option. **Allow recipients to share your dashboard.** Click **Share**. Your co-workers will receive an email invitation with a link to the shared dashboard.
3. To see with whom you have shared the dashboard, click **Shared With**.
4. If you want to unshare the dashboard with your co-worker, click on **Shared With** and then Click on **Cancel Invite** corresponding to the email address.

Power BI is a very user-friendly BI tool that helps us to create, share, and access reports just by using a browser.

## **Various Data Sources**

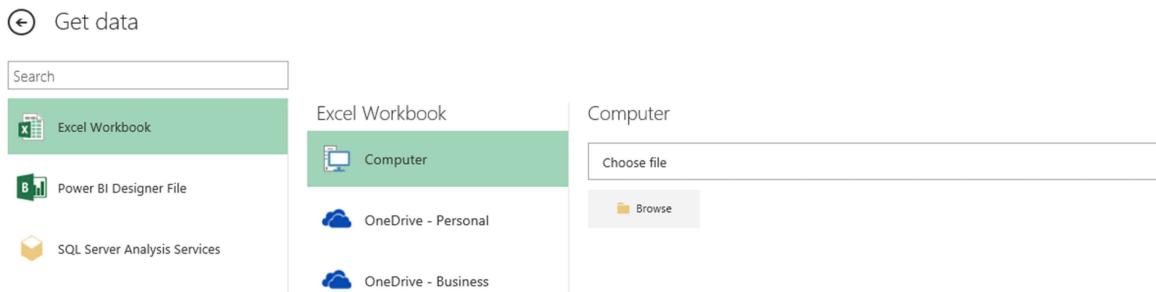
The GetData button helps you to get your own data into the dashboard, which has various data sources available to connect to get your data and build powerful dashboards for your business.

Various data sources are Excel Workbook, PowerBI Designer File, SQL Server Analysis Service, Microsoft Dynamics CRM Online. Customers can also connect to their data in SAAS services like Salesforce, Zendesk, Marketo, SendGrid, and GitHub and Microsoft Organizational Content like Microsoft: IT New Technology Adoption, Windows User Voice, Digital Crimes Unit and Peoples View.

## Starting with an Excel Workbook

First, you have to find the data to visualize. Select the Get Data option present on the left side of the screen. Now you will be able to view all the data sources.

Click on Connect to find the excel workbook by clicking on browse. When you select One Drive (Cloud Service), then you can update the changes automatically by configuring the **Data Management Gateway**.



Select the workbook that you need and click the Connect button. When you are done connecting, you are ready to start.

You can create visualizations like charts, graphs, a Filled Map, TreeMap, Funnel, Gauge, Cards, Combo Charts, etc. You can explore and learn from your own.

## Now select the visualization you want

Once the dashboard is saved, to see the workbook that you uploaded under the Dashboard heading, you can return to the main Power BI Dashboard pane.

The Power BI Q&A and connection to your workbook you uploaded into your dashboard will appear when your dashboard opens.

## **Power BI Designer File**

This application is a combination of Power Query, Power View, and Power Pivot in a single application, which allows you to build your Power BI report offline. You can later upload it to Power BI Preview to build powerful dashboards. It's just another option for people who do not have Microsoft Office 2013 Add-Ins for Power BI.

You can click on the 'Power BI Designer Preview' option in Download to get the application.

Once you are done with the installation load your data in Power BI Designer, click on Get Data Ribbon (top left corner) to get your data using Power Query and load it.

The extension for the output file is **PBIX**.

## **Limitations of Microsoft Office 2013 Power BI Add-Ins**

You will not get complete control over the data model. You can add relationships only. You will not have access to add any synonyms in the Power BI Designer.

You cannot Sign-In in Power Query. If you already have a data model present in your MS Excel, then you cannot use the same to create reports.

Features will be added to overcome a limitation in the future. You may even have an option to create a dashboard offline.

## **SQL Server Analysis Service (SSAS)**

There are two ways to connect to SQL Server Analysis Services tabular model data:

- Go to Get data, Click on SQL Server Analysis Services. Click on Connect. Select the server and get started.
- Connect to an Excel workbook that already connects to an Analysis Services server. But if your reports are in a tabular data model, then only you will be able to explore and edit in Power BI Preview.

Download the App, Install it, and configured it by an administrator.

To download and install the latest Analysis Services Connector, you can click on it from the DOWNLOAD menu.

### **Salesforce, Zendesk, Marketo, SendGrid, and GitHub**

These are applications from which you can bring your data in Power BI. After connecting to your accounts (by providing credentials) in each of those applications, you will be able to import a variety of data to explore and monitor in your Power BI dashboards.

Once you connect to these data sources, you have your data; then, you can create a visualization, pin it to your dashboards and share it with your co-workers.

### **Refresh your Data**

Most of the features of the Power BI was either already available or could be achieved by some or other means. But what was impossible until now was to have an excel workbook in Office 365 that updates automatically and provides you with the latest data. It matters a lot since data is changing, and you want the data to be up to date for your reports. **Data Management Gateway** does this for you.

The **Data Management Gateway** is a Windows service, and it needs Windows 7 or above for its installation. You can run multiple gateways within your organization. The gateways can help us in the following manners:

- With the installation done, the gateways get connected to a Power BI service (existing in “the cloud”) and receive the request for a refresh.
- On getting a request, the gateways act as an intermediary between the cloud services and on-premises data.

Most of the configuration described above exists in the Power BI Admin Centre “in the cloud.” So all the activity relating to the gateway machine can be monitored using “Resource Monitor” or “Task Manager.”

There are two ways to refresh your data, i.e., **Schedule Refresh** and **Refresh now**. This feature is only available for Excel workbooks where power query is used to connect data from sources, such as SQL (Basic), SQL Azure (Basic), Blob store (Account Key), Table store(Account Key), HD Insight(Account Key), Azure Marketplace (Marketplace Key), Facebook (OAuth), Salesforce (OAuth)

& Blank Query(N/A). In Brackets, you have the Authentication Method, which will come into use as you proceed.

Important: Apart from data sources, other restrictions might impact on Schedule Refresh option. They are:

- You have to build your queries by selecting the tables or views UI. It means you cannot enter the SQL query manually to execute.
- You have to connect to your data using Power Query. It means if you cannot connect directly from Power pivot for your data.
- When you have your data in power query, then you got to select the ‘Load to Data Model’ option. Otherwise, if you have loaded the data to the datasheet, then you cannot use the Schedule Refresh option.

## Steps to Refresh Schedule

- Select a Dataset and Click on Open Menu.
- Now you can click on Refresh Now to refresh your data right away. Or, you can click Schedule Refresh. When you do so, you will be directed to the page where you can schedule your refresh to keep your data up to date.
- Then you have to Edit Credentials to apply the changes. Choose the **Authentication Method** as I mentioned above and enter your Credentials and click on Apply, and your data will be updated as per schedule.

This is just a start in Power BI; there is still much more to come. So keep on exploring and stay up to date with the information that matters to you most.

# **Power BI for Report Generation and Mail**



## **Introduction**

Power BI can be used to build reports adhering to the business requirements. With Power BI, user can:

- Connect to a variety of data sources.
- Can bring in the data from these sources and transform the data which matches the requirement or prep it for further analysis using tools like Power Pivot or Power view.
- Create custom views over the transformed data.
- Perform data cleansing/modelling/integration operations.
- Data can be imported from multiple log files.
- Create a visualization of data using numerous graphical designs that are already available.

You will mainly focus on connecting to the SQL database, generating required reports along with visualizations and mailing of reports to specific groups or individuals.

## **Setup**

Power BI can be downloaded from the following link [Power BI download](#). The link also has other requirements that specify the software requirements for installation.

1. After installation, the Power BI Desktop icon appears on the desktop.
2. Sign in needs to be done.
3. Power BI account needs to be created if not already present.

4. After providing the correct credentials, sign-in is successful.

## **Getting Started with Power BI**

You will consider the SQL database as your source.

There are mainly six parts to it –

- Connect to the database
- Data Load
- Data Refresh
- Report Generation using visualization
- Publishing the report
- Subscription and mailing of the report.

### **Connect to the database**

Click on the ‘Get Data’ tab and a pop up showing all the data sources come up.

Click on the ‘Database’ option, choose the SQL Server database, and click on connect.

Provide the required SERVER (SID/database name) and QUERY in the highlighted text box.

Then provide the credentials of the database you want to connect in the highlighted text box.

### **Data Load**

The user can see the data in a popup and load the data into the desired window.

### **Data Refresh**

Then change the source to some other environment and edit the permissions by giving the credentials.

## **Report Generation Using Visualization**

Select the report tab from the left. Select the objects from the right-side ‘Fields’ panel based on which the report needs to be generated.

Select the type of visualizations. One can also view the details for a particular object from the graph, which depicts that the report is user-friendly and interactive as well.

## **Publishing the Report**

Once the report creation step is complete, just use the Publish button present in the Home tab to start the process.

The data and report, including all the queries, visualizations, and other features used are combined together in a package. All of them are then uploaded to the Power BI service.

As soon as the upload gets completed, a dialog box will open, which mentions that the publishing is completed. Also, there will be a link present in the dialog box, which helps in navigating to the report directly inside your web browser.

## **Report Sharing**

The report can be directly shared with an individual or a certain group. Just fill in the details (Email Id and any optional message) and then Click on the share button.

## **Subscription and Mailing of the Report**

1. Open the dashboard or report.
2. From the menu bar, Select SUBSCRIBE or select the envelope icon (present on the right side).
3. The yellow slider can be used to turn the subscription on and off. The trashcan icon can be used to delete the subscription entirely.
4. You can fill in the message details. The current mail id that you are using is pre-populated, but you can add others to the subscription as well.

NOTE: Email addresses in the same domain can be added.

## **Report Creation in Power BI Desktop with AX 2012**

For creating a Power BI Desktop report with AX 2012 R2, one needs to follow

the below steps.

Before you proceed with report creation, make sure that the correct version of Power BI Desktop is installed based on the machine's configuration, either 32 or 64 bit.

1. First of all, go to AX -> Organization administration.
2. Then navigate to Setup -> Document management.
3. Finally, move to Document data sources. Open document data source form.

Document data source form fetches data from three services such as Service, Query reference, and Custom query from AX and which can be displayed in reports at the end.

Here you will create a new document data source based on the required data from the related query.

Now that you have added an existing query of AOT "**CustTable**" as Query reference as the data source type and **Customer Details** as description.

Once the required query is selected and added on document data source form as above, go to browser/internet explorer and enter the URL of the AX Odata query service. Make sure this URL is running and displaying the XML code associated with it. This verifies the Q data query service is running fine from the AX perspective without any issue.

One can verify against query wise AX Odata query service URL by running it in the browser.

Once the AX Odata query service is successfully verified on IE, the next stage will cover report design steps in Power BI Desktop as below.

Open Power BI Desktop. Now click on Get Data option. This will open a screen which contains various available source data file supported by Power BI.

As you need to get data from AX using query service, you will select **Odata Feed** from the list and click on connect. On the next screen, it will ask to provide the URL of the AX Odata query service.

Once URL is entered on clicking OK, it will fetch and display all the query

service registered in the AX document data source form.

Here you will select **CustTable** Odata feed as a data source. In the right side window, one can preview the data populated from the query. Once the required query is selected, click on the **Load** button.

Once data is loaded, it will appear in the Power BI report designer window.

Now, based on the selected visualization-basically report format like Table, Matrix, Pie chart, BAR Chart- you can display the data in the report. Kindly note here you can make various field related and report format related changes in detail as and when required.

Once data is added, you can see it with a preview page.

Also, you can publish this report on the cloud if you have azure subscription active, which is provided at an organization level.

Below are a few other types of reports created as a sample.

- Pie chart report - Currency wise Amount.
- Donut Chart report – Customer group-wise amount.

## **Developing Power BI Reports with NAV 2016**

Using the Power BI, you can develop the reports with NAV 2016 data with pre-defined visualizations. You can easily retrieve the data from NAV and show it in reports. Before you develop the Power BI report with NAV 2016, you should verify the version of Power BI. Power BI version should be the same as OS configuration like 32 or 64 bit.

### **Steps to Develop a Power BI report with NAV 2016:**

- Configure the Web service in NAV 2016.
- Configure the NAV 2016 Web service in Power BI.
- Creating visualization with NAV 2016 data using Web service.

#### **Configure the Web service in NAV 2016:**

1. Open the NAV 2016, then go the below path to configure the Web service.

## **Administration/IT Administration/Services**

2. From here, open the Web services page. This page contains base web services.
3. By clicking the New button in the action page, you can create new web service with object type: Page
4. Add the below details to the page to create the Web service.
  - Object Type: Page
  - Object ID: 38
  - Object Name: It will populate automatically
  - Service Name: Any meaningful name
  - Published: check this flag to update OData and SOAP URL's
  - OData URL: It will update automatically
  - SOAP URL: It will update automatically

## **Configure the NAV 2016 Web service in Power BI.**

1. Open the Power BI Desktop to start the configuration of the NAV web service.
2. Select the **OData Feed** from the **Get Data** option.
3. Now enter the NAV OData web service URL in the OData Feed and click on Ok.
4. You have to give authentication on **Access an OData feed** window as shown below then click on the connect button. Here you can give two types; one is present windows credentials or any other alternate credentials like different users' credentials.
5. Now Power BI retrieves all the data from Item Ledger Entry table and display them on the screen. Click **Load**.
6. Now Query will be added to the Power BI with all the fields from the Item Ledger Entry table.
7. From the Query, you can select the fields to get the data in the Power BI

report. (Here I selected Item No., Location Code, Document No., Quantity, and Document Type.

8. You can see the Power BI report with selected fields.
9. You can change the font size also. You can add the filters to the Power BI report.
10. Now you have the final Power BI report with NAV data after using the filters and table style.

### **Creating Visualization with NAV 2016 Data using Web Service:**

Using Power BI, you can create pre-defined visualizations like Stacked Column chart, Line Chart, Stacked area chart, Pie chart. Now you will create Item quantity Location wise.

### **Below are the steps to create charts using NAV data**

1. Firstly select any chart type from visualizations.
2. Here you have to assign fields to X-Axis, Y-Axis, and values to be shown in the chart. Select Item No from Fields; it will move to under Axis. Now select the Location Code, it will move under Legend then select the quantity. (Here Axis represents X-Axis, Legend represents Y-Axis)
3. After selecting the required fields, the chart will be shown below. The X-Axis represents Item No., and Y-Axis represents Quantity.

### **Power BI Report Using SQL Server**

Create and design a Microsoft Power BI Report using data source as SQL Server. You can use the Power BI report to manipulate the data and virtualize in many graphical and table representations.

You need a minimum version of SQL Server is SQL Server2005 and above. Consider the Power BI desktop is installed and configured to allow Power BI reports in the system.

### **Steps to create Power BI reports using SQL Server**

1. Open the Power BI desktop application and click on the sign-in button.

2. Provide the User Id and password, which has power BI access, and click sign in to access the Power BI desktop.
3. You will be able to see the Power BI Screen after successful login.
4. In the Power BI desktop, click on the “Get Data” option in the action pane and then click SQL Server from the options listed.
5. A window will pop up, enter server and database details, and select data connectivity mode.
  - If you want to use existing tables as a data source, select Import as data connectivity mode.
  - If you want to write your own query instead of selecting tables, you can opt DirectQuery option as data connectivity mode. Here you are using the Import option as data connectivity mode.
  - Select data connectivity mode as per your requirement and click on OK.
6. Once the Power BI desktop is connected to the SQL Server, it lists all the available tables. Select tables that are required to design reports and click on the Load button.
  - Now the Power BI desktop application loads the table and all the columns within it on the right side under Fields section.
  - Select a graph pattern from the visualization section by clicking on it.
  - Drag and drop the fields into Axis and values section. The graph gets updated based on the fields and values selected for the visualization.
  - You can add multiple graph patterns for data visualization in your reports.
  - Now save the report and click on Publish.
  - You will get Success popup, once the report gets published to power BI.
  - To check the published report, you need to sign in to Power BI

online with the same credentials used for Power BI desktop and go to the Reports tab under “My workspaces,” and the published report could be seen here.

You can use this application to create and design quick reports with various gateways to get data. Directly you can access the SQL Server data into this application to get data. Manipulate with data and get the expected result to display in the report and design as required with the predefined graphs and formats.

## **Integration of Power BI Modules with MS Excel**

### **Components of Power BI**

#### **Power BI QnA**

Sometimes the fastest way to get an answer from your data is to ask a question using natural language. In this quickstart, we'll look at two different ways of creating the same visualization: first, building it in a report and, second, asking a question with Q&A. We'll use Power BI service, but the process is almost identical using Power BI Desktop.

#### **Query and Data Management**

Having a considerable amount of Pros, Power BI still has lots of challenges when to be implemented in small applications or accounts.

E.g.:

- a) It's still not the most powerful tool to handle the bulk data.
- b) It has lots of in tool options hence making its learning a bit complex.
- c) It is not a freeware and requires an Individual account cost, based on no. of persons using it.

**Solution:** Well, the possible solution to attain all the great features Power BI provides is to use the Excel embedded Power BI functions using the add-ins.

#### **Power Bi Implementation over Excel Through Plugins**

Below are the add-ons which you can use to attain the corresponding Power Bi Desktop functions:

For instance, you can use Power Pivot Plugins to attain Power BI Sites functionalities.

Power Query can be used to get equivalent features to Power BI Q and A and So on,

### **How to Enable these Power Plugins**

Now, before adding the plugin, they must be downloaded and installed:

Power Map: <https://www.microsoft.com/en-in/download/details.aspx?id=38395>

Power Query: <https://www.microsoft.com/en-in/download/details.aspx?id=39379>

Power Pivot : [https://msdn.microsoft.com/en-us/library/gg413462\(v=sql.110\).aspx](https://msdn.microsoft.com/en-us/library/gg413462(v=sql.110).aspx)

Once The Add-Ins have been installed. There are several steps to be followed to enable the plugins over Excel 2013

### **Steps to Enable Plugins**

1. Go to **File > Options > Add-Ins .**
2. In the **Manage** box, click **COM Add-ins > Go .**
3. Check the Plugins which you want to Enable and click “OK.”

Once Done, Notice that you will get a Power Pivot option over the Excel Tabs.

If you see carefully, you now have the option of Power Pivot with features such as

- a) Pivot Chart
- b) Power Map
- c) Power View
- d) Power Query

### **Tips for Using the Power BI Dashboard**

Here you will understand Power-BI charts and dashboard. You will get to know

some of the tips in sorting month names and displaying data in the geographical charts to show distribution. And it also provides an insight on configuring schedule refresh to update chart periodically.

You intend to work on Power-BI charts and sorting month names based on month sequence and geographically representing data to show distribution way.

Before you perform the steps, make sure that your environment meets the following prerequisites:

- Installed Power-BI desktop in your machine ( <https://powerbi.microsoft.com/en-us/desktop/> )
- Have Power-BI [site](#) access to share reports

## **Sorting Month Name**

The month name column is a string column. If you add month name to any of the filters like “Filter” or “Bar Chart,” then the month names will be sorted based on alphabets. But as a result, you expect month names should order by months which comes first.

## **Steps to Follow**

1. If you have a data column to pull data from, then create two columns, Month and Month Name directly in Power-BI. To get options to pull Month and Name, you must select the date column and then click on add column tab. These operations must perform on “Edit Queries Edit Queries.”
2. Once you created new columns with the name “Month” and “Month Name,” replace the “Month” column name with “Month Number” and “Month Name” with “Month.” And close edit queries window and apply changes.
3. Create a filtered chart for testing and add a Month column for filtering the data.
4. Now, select the Month column from rightmost columns and select Sorting by column as “Month Number.” This operation performs the sorting of month column based on month number.
5. After updating the sorting, the filter chart that you created earlier

represents data in the way months come.

## **Showing Data in a Geographical Way**

If your data is having geographical information like country, latitude, and longitude, then a chart can be displayed with few clicks to show information based on geographical location.

### **Steps to Follow**

1. In Power-BI desktop application from Visualization section on the left select “Map” chart.
2. After adding a chart to the page, you will see the properties of the map chart below the charts. Add location property with location related column from the data set, Size property with the data column like sales, count, etc. Below are the sample property values. If your data set has only longitude and latitude, then add those properties and leave location property blank.
3. After adding the above properties, the data will be represented on the maps based on the geographical location and the size.
4. To represent the data based on a specific column, then the column should be added to the “Legend” property. After adding legend property to the chart, each circle will represent data based on legend column property.

## **Schedule Refreshing Data on Power BI Dashboard**

Power-BI web site provides different operations to show reports on the dashboard and sharing with different groups of users, both internal and external organization users. And also, the most important is the data on the dashboard to refresh periodically so that the end-users see always updated information.

### **Steps to Follow**

1. Connect to your power-BI site.
2. Select appropriate data set from the available data sets.
3. Click on the ... on the left side of the data set, and it shows available properties.

4. Click on “Refresh Now” if you want to do manually.
5. Click on “Schedule Refresh” to schedule the data refresh periodically. This property allows refreshing the data based on time zone at a time, either daily or weekly. And it required authentication for the data sources.
6. Set credentials under “data source credentials” to access the data source. If there are any issues in the connectivity, that error will be listed there. To schedule an auto-refresh, there should not be any authentication issue.
7. Click on “Schedule Refresh” and provide “Refresh frequency” either daily or weekly. Specify “Time Zone” based on your time zone. Provide Time only if required else can be left empty or can be removed. By default, the time will be 12:00 AM.
8. After setting all the required properties, then click on “Apply.”
9. Now data refresh happens daily or weekly based on the setting automatically.

So this is how you can schedule data refreshing on the Power BI dashboard automatically on a daily or weekly basis. This saves a lot of time and manual effort.

# **Dynamic Row Level Security in Power BI**



## **Introduction to Power BI**

Power BI is self-service business intelligence. It provides highly interactive visualizations, where a user can create reports and dashboards by themselves. It has an interface that is very much similar to Microsoft Excel but is more powerful in terms of reporting and dashboards.

It provides a cloud-based Services –known as Power Bi Services, along with Power BI Desktop interface. The Initial version was unveiled in September 2014.

Power BI service provides fast and easy access to data, supporting inputs from most of the data sources, and the data can be visualized and analyzed by anyone.

Here we will discuss points to publish a Power BI report to Power Bi online dashboard with Row-level security and how to embed this report to the SharePoint site. This is an open-source Power BI Desktop and Power BI Service online to allow to build simple to complex report to monitor the health of the system. You can provide different facilities for the end-users to visualize the data.

## **Benefits of Power BI**

1. Fast Deployment, secure, and can be integrated with any existing systems.
2. Pre Build Dashboard templates.
3. Supports both live connections, on-premise, and cloud.

## **Dynamic Row Level Security**

Power BI automatically updates the data in real-time and providing options to

refresh the data by schedule refresh promptly and which will depend on data sources that you are binding to the Power BI report. Real-time data options will be available through Azure Stream Analytics, integration, and the REST API for Power BI. With both mentioned ways, the data can be moved into Power BI directly.

## **Power BI Desktop**

Power BI desktop is a visual data exploration and reporting tool. Using this reporting tool, users can connect different data sources and can generate reports and dashboards at a single location. Power BI desktop tool can connect a variety of data sources like Excel, SQL, OData feed, etc. There are already many supporting formats like charts, tables which can help the user in analyzing the data.

Power BI desktop can connect different data sources like files (Excel and CSV), Azure (Azure Services), Databases (SQL, and Oracle Server DB), and also Facebook and Google analytics.

To restrict data at report or dataset level, Row Level Security (RLS) is used.

To implement dynamic RLS in Power BI, you need to follow the below steps.

1. Create a role and maintain security logic (using DAX Expression) in the Power BI Desktop tool.
2. Create a workspace & members to it for whom RLS has to be enabled (Optional)
3. Upload the report to Power BI services and assign users under the security tab of the dataset.

Here you will use the data from an Excel source, which has two tables, Sales\_Products, and Country.

Sales\_Product Table

Country Table

In this example, you have three users Mary (Manager of US & Canada), Dan (Manager of UK) and John (Admin User)

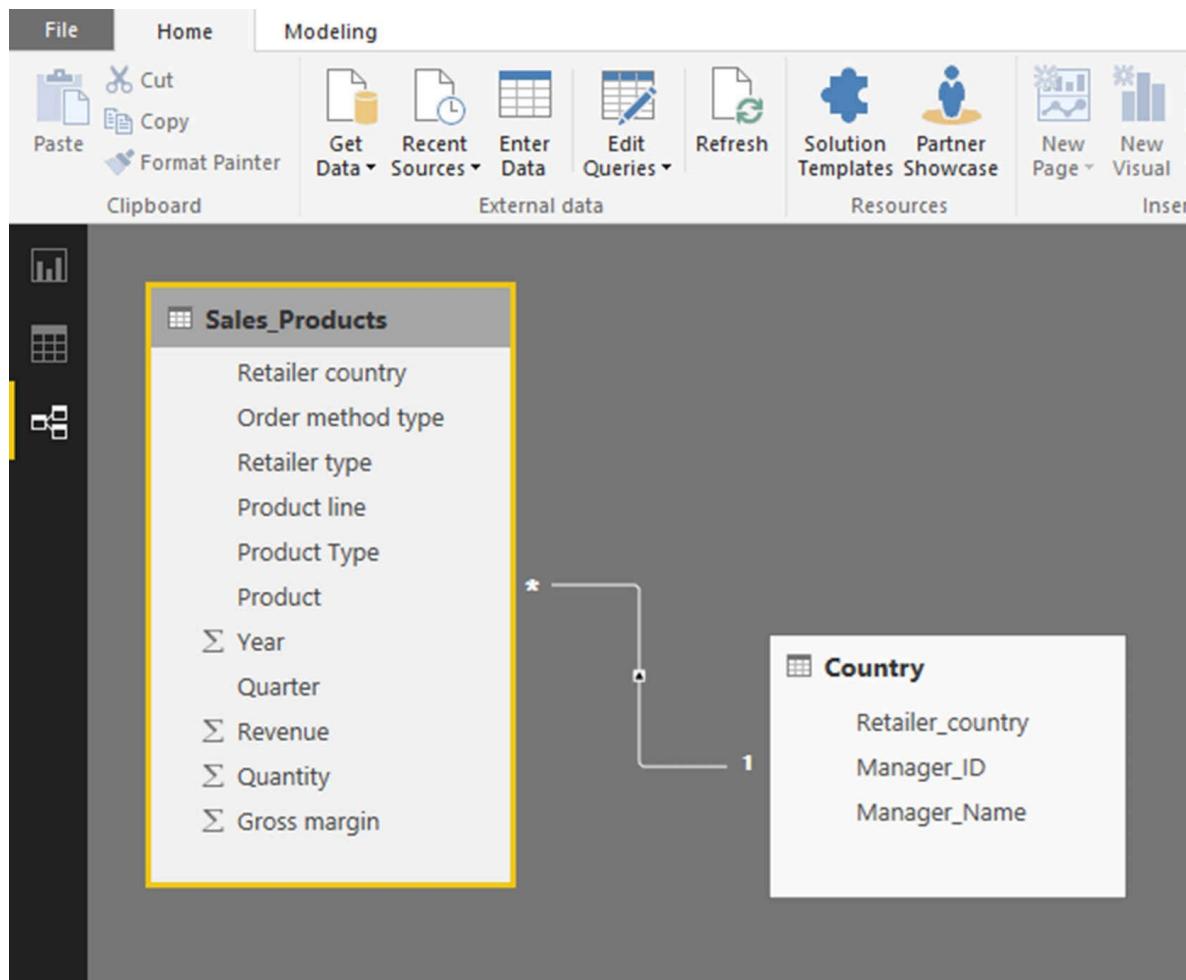
You will create a report and apply dynamic security. The Manager will be seeing

only those countries which belong to him (or) her.

## Steps to Implement Dynamic RLS

Import tables in Power BI and create the relationship by joining the “Retailer\_country” column from the country table with the “Retailer country” column in Sales\_Products.

This relationship is associated with the Manager Id.



Create the report as per requirement:

In example. You have created three slicers (Product Line, Retail Country & manager) and one detailed report (table view).

Once the report is created and then you will implement security.

### Step 1

Create a role and add filter condition.

To do this, go to the modeling tab. Then create a role(Manager) and add filter condition on the respective table. In the example you have added “Manager\_ID = Username()” condition on country table.

Username() is the DAX function, which will return the username of the person who is logged in.

By using this filter, the user who has logged in will only be able to see their own records present in the complete dataset. Here in the example, the Manager\_Id field in Country table defined exactly like the Power BI usernames. Sales\_Products table is the replica of Retailer\_Country, therefore changing any of the tables will also reflect in the other table automatically.

## **Step 2 :**

Create an app workspace in Power BI service as Implement\_RLS\_Group and add the members into it.

Log into Power BI services (using admin login), click on “Create app workspace” under workspaces.

Then provide Name, Privacy settings, and members mentioned as below example.

Example:

Name: Implement\_RLS\_Group

Members:

John@xyz.com – Admin (The person who creates the app workspace)

Dan@xyz.com – Member (Manager for the UK)

Mary@xyz.com – Member (Manager for US & Canada)

### **Step 3:**

Assign users to Power BI Security.

First, save, and then publish the report under “Implement\_RLS\_Group” in Power BI. Then, navigate to the “Security” tab of the dataset, which has been published just now (In your case, the name of the dataset is Store\_RLS).

Click on the “security tab,” you will see the role you created in power BI Desktop. Add the user under that role for whom RLS should be implemented.

The users added on this page will see only those records, i.e., which are associated with their own username. Whereas for the user who publishes (admin), the report can see whole revenue details since he/she is the admin.

### **Step 4 :**

Test the RLS by logging into BI services using Mary or John login.

You should only see those records that belong to them.

This is how Dynamic RLS is implemented in Power BI.

### **Simple Row Level Security in Power BI**

Power BI Row-level security can be used to limit the report access for specified users. Also, you have Power BI filters to restrict data at the row-level; the filters within the roles can be defined.

The security roles can be defined by following these steps:

1. Open the Power BI report and select the Modeling tab.
2. Then choose **Manage Roles** option from the **Modeling** tab.
3. Select **Create** .
4. You should name the role after clicking on the **create** button as above.
5. Choose the table in which you are going to restrict and choose a DAX rule to apply.
6. Enter a DAX expressions - this will return true/false. E.g.: [Entity ID] = “Value”.

7. You can select the checkmark after applying the DAX formula to validate above the expression content box to the expression as below.

8. Choose **Save**.

Users can only be assigned roles through the Power BI service, not through the Power BI desktop. However, the Desktop can be used for creating the role.

9. After creating a role, the results of that role can be validated in Power BI Desktop. For this, you should choose **View As Roles** option from the **Modeling** tab as below shown below.

10. You can see and change the view roles **using View as Roles** dialog as below.

11. After that, you can select the checkbox for the respective role you have created and click on **OK** to apply modifications. The reports will pull the data appropriate for that selected role only. You can select other given user checkbox also.

You can bring the data from the below datasets to Power BI.

1. SaaS Data Sources (Software as Service): GitHub, Microsoft Dynamics CRM, SendGrid
2. Azure Resident Systems: Azure SQL/DW
3. On-Premise systems: SQL, Tabular, Excel, PBI Desktop

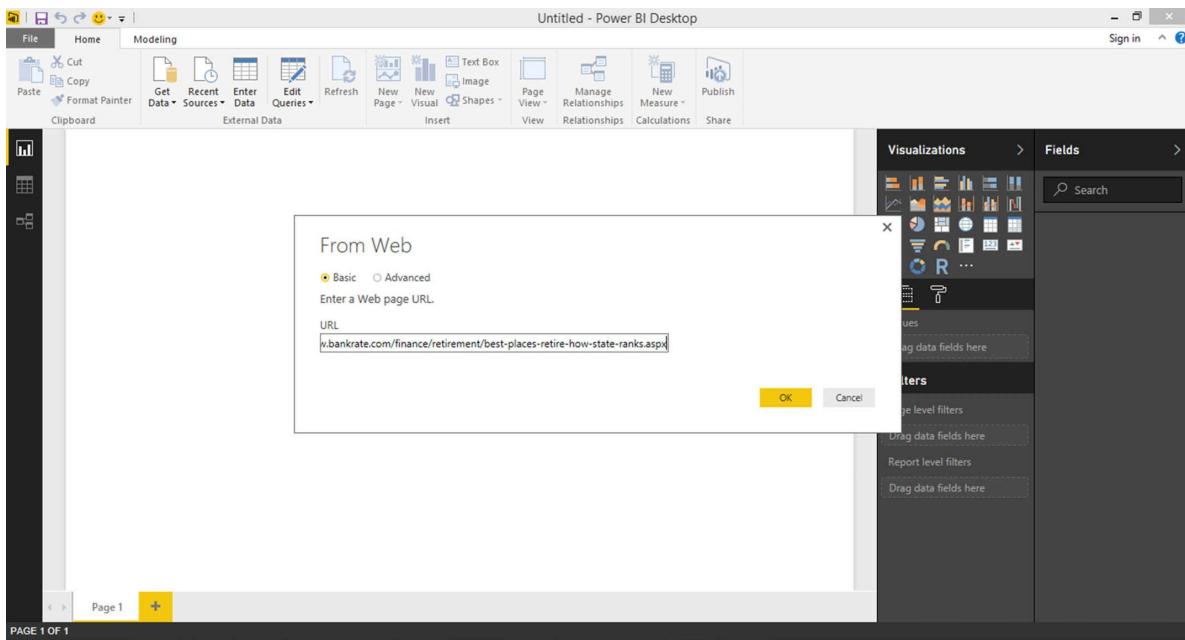
## **Row-Level Security in Power BI**

You have to create roles within the Power BI Desktop. You cannot create the roles in the Power BI service.

When you publish this to Power BI Service, the role definition will also be published. You can add the users in the Roles created at Power BI Service.

## **Steps to Define a Role in Power BI Desktop**

### **Step 1: Import Data to Power BI Desktop**



From the above web source, I have selected the data from Table(0).

## Step 2:

Now select the Modeling tab and click on the Manage Roles.

I'm creating a Role Texas to enable the user to view data only in that State.

You have to create the DAX expression for the same and can validate by clicking on the tick mark to avoid further errors.

As mentioned above, you cannot assign users within Power BI Desktop; this can be done only at Power BI Service.

Instead, you can have dynamic security in Power BI Desktop using the username() and lookup function as in Tabular Cube.

## Step 3:

Publish this report and log onto the Power BI Service. Under the datasets, click(...) on the Model that you have created.

Now you can add the User ID under the members of the Roles created. You can also test the Role by clicking on Test as Role. (Now you will be able to see the

data only for Texas).

## Publishing the Report

1. Select the **Publish** button.
2. Once Publish has done, it will show below popup for success confirmation.
3. By using the Link in the above popup, you can open Power BI online service.
- 4.

## Manage Security on Model

To manage security on your data model, you should do the following steps:

1. Select the security under the **dataset** .
2. You can add a member to the role by giving an email address or name of the user. This member should exist within your organization. You cannot add Groups created within the Power BI desktop.
3. You can remove members also by clicking on X next to their names.

## Embed in SharePoint Online

1. Go to My Workspace-> Report-> File -> “ **Embed in SharePoint Online** ”.
2. You will get a URL. Now you should log-in to the SharePoint site and create a new page.
3. After that, you can see a PLUS button as below to add/embed the Power BI report to SharePoint.
4. You should place the URL (Which you copied from Power BI online My Workspace-> Report-> File -> “ **Embed in SharePoint Online** ”) in the below box to embed.
5. Once you are done with the above steps, you can see the Power BI report in SharePoint online.

## **Power BI Mobile Development Steps**

1. The Power BI mobile applications are where you can view Power BI content only, but you cannot create it.
2. Power BI Mobile development will be done in Power BI desktop; then, it will have converted into a Mobile view.
3. Once you are done with required functionalities in Power BI desktop.
4. Then you can change the view into the phone view by selecting the option “Phone Layout” from the VIEW tab.
5. Then it will open the visual and Mobile view. You need to arrange the visuals as you wanted in the Power BI Mobile screen by drag and drop the visuals into the Mobile view.
6. Then publish the above changes into power BI service as below. From there, you can access the Power BI service and Power BI Mobile app as well.
  - a. Select the Publish button.
  - b. Once Publish has done, it will show below popup for success confirmation.
7. You should install the Power BI Mobile app from the App Store to access the same.

## **Cons in Power BI Row Level Security**

- Only Direct Query connections are supported. This doesn't support live the connections to Analysis Services. You have to have them on

the on-Premise Model itself.

- You have to recreate the roles in Power BI Desktop if you have them in Power BI services.
- Cortana is not supported.
- The roles can only be defined in Power BI Desktop.

Therefore, Row-Level Security is just a way to restrict the access of the user. This can be achieved by applying filters at the row level.

# **Toggle Button and Tooltip Features in Power BI**



## **Introduction**

In this chapter, we will discuss the Bookmarks in Power Bi Desktop, how to implement the Toggle button with the Bookmark feature, and how to implement report Tooltip in Power BI desktop.

## **Designing a Dashboard using the Imported Dataset**

### **Metrics to be Displayed in the Report**

1. To Display gross profit by year
  2. To Display gross profit by country
  3. To Display gross profit by product type
  4. To Display revenue, planned revenue and gross profit by year in Tabular format
  5. To Display revenue, planned revenue and gross profit by product type in Tabular format
1. To Display gross profit by year
- Drag and drop the clustered column chart from the visualizations pane to full fill this requirement.
  - Drag and drop the year from fields pane to the axis field of clustered column chart and drag and drop the gross profit from fields pane into the value filed of a clustered column chart.
  - To customize the chart, change the following format options from the format tab.
    - To show the data labels turn the Data labels slider to On, then in the

Display units drop-down, select the Millions to show the values as millions.

## 2. To Display gross profit by country

- Drag and drop the Treemap from the visualizations pane to full fill this requirement.
- Drag and drop retailer country from fields pane to the group field of treemap and drag and drop the gross profit from fields pane into the values filed of the treemap.
- To customize the treemap, change the following format options from the format tab.
  - To show the data labels turn the Data labels slider to On, then in the Display units drop-down, select the Millions to show the values as millions.

## 3. To Display gross profit by product type

- Drag and drop the pie chart from the visualizations pane to full fill this requirement.
- Drag and drop product type from fields pane to the legend field of pie chart and drag and drop the gross profit from fields pane into the values filed of a pie chart.

## 4. To Display revenue, planned revenue and gross profit by year in Tabular format

- Drag and drop the table from the visualizations pane to full fill this requirement.
- Drag and drop year, revenue, planned revenue, and gross profit from fields pane into the values filed of the table.
- To customize the table format, change the following format options from the format tab.
  - To change the table style, select the format section, and expand the Table Style card, then Style drop-down, select the table style whichever is needed.

5. To Display revenue, planned revenue and gross profit by product type in Tabular format

- Drag and drop the table from the visualization pane to full fill this requirement.
- Drag and drop product type, revenue, planned revenue, and gross profit from fields pane into values filed of the table.
- To customize the table format, change the following format options from the format tab.
  - To change the table style, select the format section, and expand the Table Style card, then Style drop-down, select the table style whichever is needed.

**Use Case :** The customer wants to display different views on a page when you click on a button and show multiple metrics on a single tile.

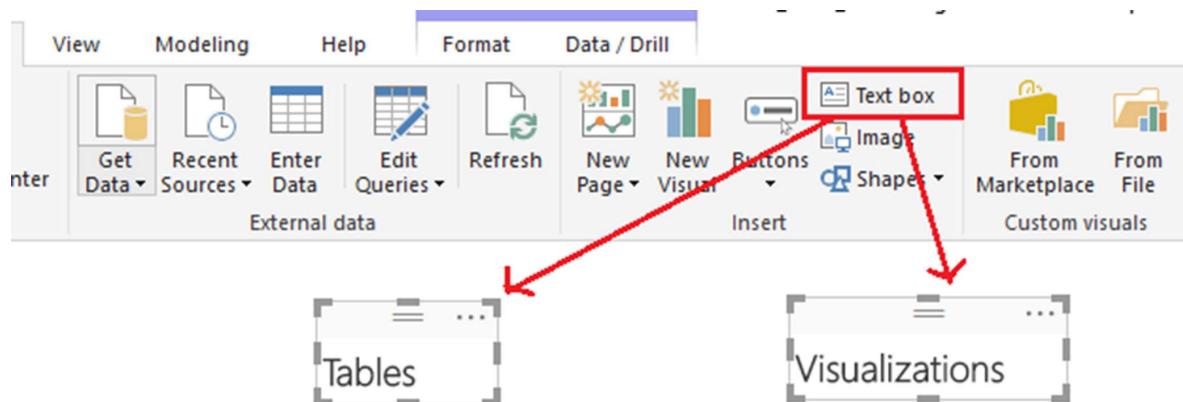
Feature Requirements:

1. Toggle between different visuals in a single page
2. Report Page Tooltip

## Toggle Button with Bookmark Feature

### Create Bookmark in Power BI Desktop

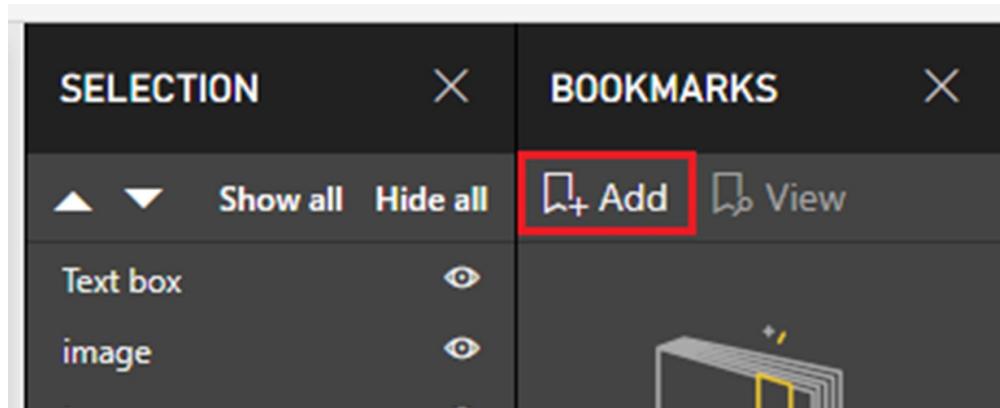
- Add two textboxes from the home tab and name it as Tables and Visualizations.



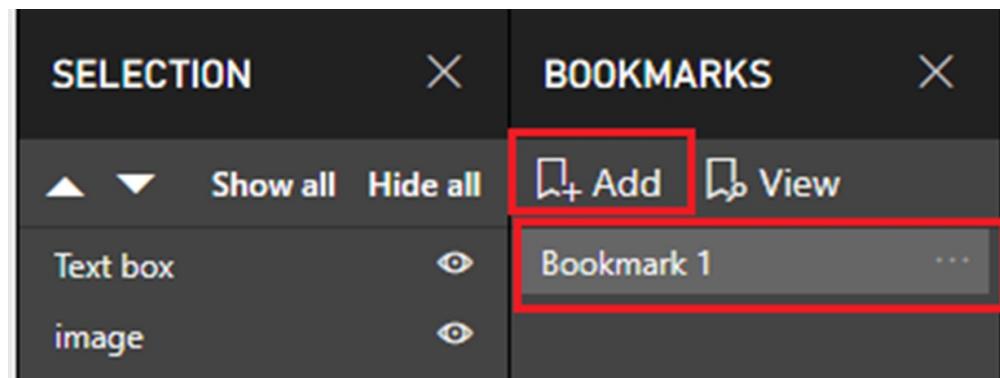
- Import toggle button images from the home tab to implement a

bookmark feature with a toggle button.

- To get the bookmarks pane and selection pane to go to view, check the bookmarks pane and selection pane.
- Click on the Add button to add the new bookmark.



- You will get the bookmark.



- Click on three dots you will get the options of the bookmark, then select rename and type Tables.
- Add another bookmark and rename it as Visualizations.
- Select the Tables to configure tables bookmark.
- Configure the tables bookmark.
  - Select tables, then in the selection pane, click on the eye button to hide the visuals of graphical visualizations and toggle button images.
  - Click on three dots then select an update to update the bookmark.

- Configure the visualizations bookmark.
- Select visualizations, then in the selection pane, click on the eye button to hide the visuals of tables and toggle button images.
- Click on three dots then click on Update to update the bookmark.
- Configure the Tables Toggle Button.
  - Select the Toggle button image, turn the Action slider to On, in the format image section, then select type as a bookmark, and select bookmark as Visualizations
- Configure the Visualizations Toggle Button.
  - Select the Toggle button image, turn the Action slider to On, in the format image section, then select type as a bookmark, and select bookmark as Tables.
- Arrange both the toggle button images one under another to feel like toggle button in the dashboard.
  - Click on show all in the selection pane.
  - After the arrangement of images, the toggle button shown in the below image.



- While clicking the toggle button, press the Control key in Power BI Desktop; there is no need to press the control key while clicking the toggle button in Power BI Online Services.
  - When you click on the below button, you will get the tables.
  - When you click on the below button, you will get the visualizations.



## Report Page Tooltip in Power BI Desktop

- Before you can create a tooltip, you need to enable the tooltip feature.
  - To enable the tooltip feature, go to File -> Options and Settings. Then move to Options -> Preview Features. Finally, navigate to Power BI Desktop. Checkmark the checkbox beside the tooltips.
  - You are required to restart the Power BI Desktop after making the changes to enable the preview of the report tooltips.

Note: To implement Report tooltip feature in Power BI Desktop install March 2018 (2.56) or later versions.

- Create a report tooltip page.
  - First, create a report page with the help of the + button present at the bottom of the screen.
  - A new template called Tooltip will be visible in the formatting pane. It will help you to view the report page canvas size.
  - You need to modify the Page View to its original size to get a better view of the tooltip page.
  - Now choose the View option from the top ribbon. Now go to Page View -> Actual Size.
  - Choose the information card present on the Format page. You can provide the name in that particular field.
  - Now you can easily create any visuals of your choice to be shown up in the tooltip.
  - There is one Stacked bar chart present on the tooltip page. It also has two other values as Revenue and Planned Revenue with customized colors, to provide the look you wanted.
- Configure the automatic report tooltip.
  - You need to turn on the Tooltip slider and then move to the Page Information card. There you can mark the page as a tooltip.
  - For visuals in the report, to auto-enable the tooltip to drag the fields

into the Tooltip fields bucket, so that the fields can be specified for all the reports appearing in the tooltip.

- When the mouse is hovering over the sections in the pie chart, it shows the product type and gross profit in normal visuals, as shown in the below image.
- But with the tooltip configured visual, when the mouse is hovering over the sections in the pie chart, it shows the tooltip as Revenue and Planned revenue to that particular product type, as shown in the below image.
- Manually setting a report tooltip.
  - For setting up the tooltip manually, you can choose the visual of your own choice. Then navigate to visualizations pane, followed by format section. Expanding the tooltip card will show you the tooltip page as per the selected visual.
  - When the mouse is hovering over the sections in the treemap, it shows the retailer country and gross profit in normal visuals, as shown in the below image.
  - But with the tooltip configured visual, when the mouse is hovering over the sections in the treemap, it shows the tooltip as Revenue and Planned revenue to that particular country.

## **Publishing Reports on Apps in Power BI**

Microsoft Power BI is a reporting tool that is used for data visualization. It can connect to different data sources, produce reports, and publish them for various organizations. It brings data to life with complete visualization. This section provides detailed steps for publishing the reports/dashboard for the members of various organizations or specific individuals using the Apps Workspace.

For the benefit of the organization, we have tested the scenario and attached the screenshots of the sample data and the dashboard that comes in the app once it is published. This can be a potential use case to share the reports like monthly profit dashboards, quarterly results of the company, etc. with identified stakeholders.

## **App Workspace**

Apps Workspace will act as a placeholder for published dashboards and reports. Apps are bound to App Workspace; that is, before you create an app, you have to have an app workspace. App Workspace is fairly similar to a group workspace.

- The App workspace has dashboards, reports, datasets, excels.
- Apps are the published version of whatever is inside of the workspace.
- Apps Workspace acts as a Potential Staging Environment for the apps.
- With Potential Staging Environment, we mean that you can modify the dashboards, and then the dashboards can be published to apps.
- After the apps are published, you can still make modifications to the existing dashboards in the apps workspace and can publish the updated app.
- You can manage the users as to who can view the apps.

## **Creating Apps**

- Apps are a collection of reports dashboards and which are sent across to multiple organizations or a group of people or even to specific people.
- Earlier Organizational content Packs were being used for publishing. However, now Apps have replaced them.
- As you create the app in the apps workspace, you grant permissions on the dashboards.

Step 1: To Create the App Workspace, click on Create app workspace.

Step 2: Fill in the details for creating the apps workspace. You can add members to this workspace.

Private: Only approved members of the workspace can view the app.

Public: All members of the organization can view the app.

Members can be granted permission:

1. View: Members can only view Power BI Content
2. Edit: Members can edit Power BI Content

Step 3: Go to your Power BI Desktop Version and publish the report to Power BI Service.

Step 4: Select the destination to be published to:

Visit the report published by Power BI desktop in the Power BI Service and pin this report to the dashboard.

### **Publishing App**

Go to Workspace -> The app workspace you created -> Click on Dashboards

Click on Publish App.

This will take you through certain steps:

1. Fill in Details, Content, and Access
2. And click on Finish to end publishing the app

This is the App Icon generated through App Workspace. On clicking the app icon user can view the dashboard.

The end-user/business user can install the app from Microsoft AppSource, or the admin/creator of the app can directly send the URL link to the app.

### **Updating the App**

For any changes made to the dashboards in the Apps Workspace. The app needs to be updated.

1. Modify the dashboard: You will have edit report option on the top of the report to make any changes
2. In the app workspace, you will find the update app option.
3. Update all the options: Details, Content, and Access.
4. Click on Update App.

### **Unpublish the App**

For unpublishing the app, you can click on the option **Unpublish App**. Below is the snapshot for the same.

**Note:**

- APPS are created in the APPS WORKSPACE.
- To create the app, you will need the Power BI Pro License.
- All users need Power BI Pro Licenses to view the app.

## **Sourcing an Existing Power BI Report into a New Power BI Report Using Live Connection**

Let us understand in detail how to source an existing Power BI report into a new Power BI report. If there is a requirement of creating a new Power BI reports by making use of the existing set of the report, then Power BI provides an option to do this. With this option, the effort of recreating the data model, and the process of data refresh multiple times is eliminated.

The steps are explained below in detail.

**Step 1:** Open a new Power BI desktop file and log in using your Microsoft account.

**Step 2:** In the navigation bar under ‘Get Data,’ select ‘Power BI datasets.’ This connects to the workspace created in the Power BI server.

**Step 3:** Select the required report from the workspace and click on ‘Load.’

**Step 4:** This step adds all the tables which are available in the existing power BI report to the newly created report. These tables are available on the right-hand side of the report. And at the bottom right corner, there is a message displayed saying that the report is connected live to Power BI dataset, report name, and the workspace name.

**Step 5:** This newly created file can then be used to create the required visualizations for analysis.

## **Advantages**

- Rework in creating the data model is reduced.
- If multiple reports are using a common dataset, then this is one of the efficient approaches.
- Refreshing the data for source. pbix file is sufficient; refreshed data will be available in the new reports as well. This eliminates data extraction from the same data set multiple times.

## **Limitations**

- Source. pbix file, which is to be used, must be published in the workspace; any local files cannot be used as a source for a new. pbix file.
- Any changes made in the source. pbix file will be replicated in the new files too.
- If the source. pbix file is deleted in the workspace; the new file will no longer work.

# **Power BI and Share Point**



## **Introduction**

This chapter gives a brief introduction on using Power BI Desktop to generate reports and dashboards and integrating the reports in SharePoint to achieve Business Intelligence.

Business Intelligence in SharePoint is vast and is achieved with the combination of SharePoint and Microsoft Power BI in particularly Excel. This combination of the PBIX file and SharePoint enables us to create dashboards, reports, and scorecards with simple as well as complex data.

The above-said process is two-fold.

- Creation of Power BI Desktop file
- Integration with the SharePoint

Power BI is a cloud-based Analytics Service that allows users to analyze and visualize data with greater speed and efficiency. The data is transformed into interactive reports, dashboards, or Compelling visualizations.

The Power BI Service is built upon Azure. The Power BI service architecture consists of 2 clusters.

1. WFE Cluster
2. Back end Cluster

WFE Cluster manages the initial connection and authentications process. Whereas BE Cluster handles the user interactions. Power BI uses Azure Active directory for the purpose of user validation & authentication, whereas Azure Blob and Azure SQL manage the storage of data and Metadata.

Power BI Service is not available as a private internal cloud. However, using Power BI and Power BI Desktop, you can connect to your on-premises data sources using on-premises Data Gateway.

Power BI is a Microsoft product which is of twofold:

- Power BI downloadable Desktop program
- Cloud service (online program).

This document elaborates the steps to install and create Power BI Reports using Power BI Desktop program

## **Power BI Desktop**

Power BI Desktop integrates the powerful query engine with the online Power BI Service. It is a Powerful analytics tool designed for windows. Powerful data connections, models, and reports are created using the Power BI Desktop. Whereas Power BI Service is used to shares the power BI Desktop reports to the users to view and interact with them

It is a powerful and highly accessible tool to build robust models and craft reports.

### **How Power BI Desktop Works**

Power BI Desktop enables the user to connect to different data sources data, shape or refine the data and represent the data model in the form of reports. This centralizes the scattered and disconnected data and represents them in the form of reports. The user can save the work to a file, which is Power BI Desktop file format (.pbix) file format. This can be shared with other users by uploading the file to the Power BI SERVICE

### **Power BI Desktop Installation**

Go to PowerBI.Microsoft.com and click on Products and click on Power BI Desktop. Click on the download button. A PBI Desktop MSI file will be downloaded to the computer.

Execute the MSI file by following the instructions. A shortcut icon will be created on the desktop.

## **Requirements /Limitations**

To use the Power BI, few requirements need to be satisfied.

1. Power BI Account – The user has to register to PowerBI.COM and get created the account
2. Power pivots and power queries are used to generate reports.
3. A website is needed to upload the report. Website can be either SharePoint / WordPress / any website / web page.

## **Power BI Report Creation**

To start up with, double click the Desktop icon, a welcome screen is displayed.

Click on Get Data Icon to select the raw input data. It supports many input data formats like files, Databases, Azure, or Online services.

In this current context, you will take up the SQL Server database as input data. Select the SQL Server database and click on the connect button. Enter the database details.

Click on the Load button to load the data to Report.

## **Views in Power BI Desktop**

- Report View
- Data View
- Relationship view

There are three views found towards the left of the Power BI Desktop.

**Report View** is the View, which shows the actual representation of the data based on the visualizations of the chart selected.

**Data View** depicts the data in table format. You can cross verify the data to be mapped to the report. Intermediate changes are allowed to adjust the data using the query editor window and need to refresh the data for the same to get refreshed to the report. The below diagram depicts the sample data

**Relationship View** depicts the pictorial representation of the relationship between columns.

The user can also Edit / Combine more than one table using query editor. The query editor is used to shape and combine the data from multiple tables. However, the original data source is not getting altered. Based on the requirement, only the view of this data source is getting altered. This also has an option of merging queries.

All data manipulation activities are carried out in the query editor. Suppose the current data type of the column is decimal and need to convert to the whole number just right-click the column header and select change type and again select the whole number.

Finally, to apply the query Editor changes, click on the close & apply button in the Home Ribbon Tab. The Power BI Table is added to the report.

## **Build Reports**

The Report View has five sections.

- The ribbon panel displays the tasks associated with the report.
- Report View pane enables the user to view the Visualizations and customize the report.
- Pages pane enables you to add a report page.
- Visualizations pane to customize the report like applying the different kinds of charts and axis fields and filters.
- Fields pane enables the user to drag query elements and filters to the report view.

In this context, as the Power BI Table data contains state data, let us create a Map report. Based on the input data and requirement of the user can create different types of visualizations like Pie chart, Bar diagram, Line chart, etc.

To create the visualization, Drag and drop the fields column to the location field, i.e., drag the state column to the report view, A map with the states defined in the table.

In this example, as you created the selected Map Report Chart in the visualization, the data available in the table column state is mapped to the Map control Chart. An in-depth view of the states can be obtained by applying filters

for the states.

## **Creating Charts For Sharepoint List Data Using Power BI**

If the source data changes, you need to update the Excel sheet. The issue mentioned above can be rectified by using Microsoft Power BI by analyzing and building charts.

### Step 1 – Power BI Desktop Activities and publishing

1. Please download the Power BI desktop version. Once the Power BI desktop version is installed successfully, open the same using the Office365 credentials.
2. Select Get Data SharePoint Online List. Then a pop up will open. Provide the Site link and then log in using your credentials for that application.

**Note:** Apart from SharePoint online, there are so many options available for which you have to analyze the data and create the charts. I have listed out a few of the options available there once you click on “Get Date.”

SharePoint List – Choose this if your application is an on-premise SharePoint application.

SharePoint folder

Azure, SQL

Oracle

ODBC

SAP HANA DB

Microsoft Exchange

Microsoft Exchange Online

Dynamic 365

3. Select the list for which you need to represent that data (list items) in a chart.

4. Once the list is loaded successfully, select the Pie Chart option on Visualizations (right-hand panel). Drag & drop parameters (list columns) for which you analyze the data.
5. After completing your chart, you can save the file with the .pbix extension.
6. In the top ribbon, you can find the “Publish” button. Click on it.

Note: Once you published the chart using the desktop version, the same will be pushed to the Office 365 Power BI web version, where you need to perform some actions to set the Refresh time.

## Step 2 - Connection update:

1. Open the Power BI web version. Log in using your Office365 credentials.
2. Once logged in successfully, click on Datasets Data Connection. On that page, click on Data Source Credentials Edit Credentials OAuth2. Provide your/admin credentials on the authentication page.
3. Select “Schedule Refresh” and set the below refresh factors as per your requirement.

Refresh Frequency

Time Zone

Refresh between

4. Click Apply.

Note: At this point in time, you are almost done with the chart preparation in which the data will be refreshed automatically according to the settings made above—the only thing you need to do to make use of this chart on the application page.

## Step 3 - Sharing

You can share the chart with your organization or a particular set of people or

use it in the application.

### **1. Share the Report via Dashboard**

- a. In the web version Power BI page, you can find “Dashboard” in the left panel. Click on + plus sign near Dashboard.
- b. **In the left panel under “Report,” click on the report which you want to share. Select the “Pin Live” page.**
- c. **In the Pine Live page, select the Dashboard where you want to share and finally click on “Pin Live.”**

### **2. Share Report using Content Pack**

- a. In the web Power Bi page, click on the gear icon at the top panel.
- b. **From the various option listed, select “Create Content Pack.”**
- c. **Provide the list of users’ email addresses for whom you want to share the data analysis and select the respective Report.**
- d. **Click OK.**

### **3. Share the Report using the URL or embedded URL:**

- a. In the web version Power BI page, you can find “Report” in the left panel. Click on the report you want to publish.
- b. **In the top panel, click on the “File” option.**
- c. **Then click on “Publish to Web.”**
- d. **A pop up will appear, which will provide you the two options.**
- e. **The link you can send in an email the direct URL which you can share with users.**
- f. **Html, you can paste into your blog or website the code which you can add in “Embed content” of your HTMLK**

page to display the data analysis.

- g. **Size** You can select the required size. Once you select the size, the above “Html” code will change accordingly.

## Integrating Power BI with SharePoint

Power BI is one of the Microsoft tools. Here you will explain the process of publishing a Power BI report to the Power BI online dashboard and to embed the report to the SharePoint site. Microsoft Power BI Desktop can be downloaded from [here](#). After installing Power BI desktop, sign in with your enterprise office 365 account for accessing Power BI Application, which is also available for a free trial version, also paid one as per business requirement.

Power BI desktop can connect a variety of data sources like files such as Excel and CSV, Azure (Azure Services), Databases (SQL, and Oracle Server database) and SharePoint Online, etc. and create simple and complex customized business reports.

When the installation is complete, open Power BI Desktop and connect to the data source and create reports. Save the report in Power BI Desktop format, which is the “.pbix” extension. Finally, publish reports to Power BI Application and embed it to SharePoint.

### Data Refresh

Power BI frequently updates the data in real-time, if you are using a paid version of Power BI and providing options to refresh the data by schedule refresh in a timely manner. If you are using a free version of Power BI, can schedule refresh by timely manner (eight times per day) not frequently.

### Steps to Create Power BI Reports

This section explains how to connect to SharePoint online data for creating reports.

#### Connect to SharePoint Online

- i. Open Power BI Desktop and click on ‘Get Data.’
- ii. Click on “More...” and Select Online Services -> SharePoint Online List and click on “Connect.”

- iii. Enter the SharePoint site URL and click on “OK.”
- iv. Select Microsoft account -> Select a site -> Sign in.
- v. A login window will open. Provide Microsoft credentials and click on “Connect.”
- vi. Select the lists to generate reports and click on “Load.”
- vii. Selected lists get loaded on the right side under “FIELDS.”

## **Create Reports using Different Visualizations**

You can create different types of reports using different visualizations. Ex. Pie chart, Donut chart, stacked column chart, etc. You can import charts from the store, from the file. Also, it is possible to import custom visuals to Power BI Desktop. Below are some of the sample charts.

ChicletSlicer: This chiclet slicer has to import from the store.

- a. Click on ... under visualizations and select Import from the store.
- b. Search for ChicletSlicer and click on “ADD.”
- c. Click on the ChicletSlicer, it gets added to the page.
- d. Drag and drop fields from lists as per requirement. Here the selected field is dragged and drop into “Category.”
- e. Select the “Format” tab and add the header, change the background color, font, etc. as per requirement.

In the same way, you will create other reports as per business needs.

## **Create Relationships between Two Charts**

You can create a relationship between two tables in the Power BI. So, based on the selection of one chart, other charts data get filtered in the reports. Below are the steps to create a relationship.

- i. Click on “Manage Relationships.”
- ii. Select the tables.
- iii. Double click on the selected tables and select the unique value field in

both tables and click on “OK.”

## Create Custom and Calculated Columns in Power BI

You can create calculated columns in the tables. For example, while creating a chart using two tables or one table, that particular chart needs a column value that needs to be customized for chart representation. Below are some of the columns which will be used in Power BI charts.

### Create a Calculated Column

Calculated column using the DAX (Data Analysis Expressions) formula in Power BI and visualization tools. For row-by-row calculation calculated column will be used. For adding Calculated column,

- a. Go to “FIELDS” and right-click on the table and click on “New Column.”
- b. Rename the column and write the formula as per requirement. Use that column for creating charts.

### Create Measure Column

Measures are used to calculate aggregates, such as sum or average of a column.

- a. Go to “FIELDS” and right-click on the table and click on “New measure.”

Sample Formula: Sum of XY = CALCULATE(SUM('X'[a]), ('Y'[b]))

### Conditional Column and Custom Column

The custom column uses the M language. In the custom column, you cannot perform functions like SUM, AVG, etc. So, it is recommending to use Measures and Calculated column in Power BI for calculations. For creating custom and conditional column.

- a. Go to “Edit Queries” and select “Edit Queries.”
- b. Under “Add Column,” select “Custom” or “Conditional Column.”

Publish to Power BI App and Embed in SharePoint online

This section explains how to publish the Power BI file to the Power BI app and embed it in SharePoint online.

1. Click on “Publish” and Select a Destination and click on “Select.”
2. Click on “Open ‘document name.pbix’ in Power BI.”
3. Sign in to Power BI Account; the Power BI app will open with the published file.
4. For embed into SharePoint, click on “File” and click on “Publish to the web.”
5. Click on “Create embed code.” Click on “Publish.”
6. Copy the “Html you can paste into your blog or website.”
7. Go to the SharePoint page. Click on “Edit Page.” Click on “Edit Source” and paste the HTML code.
8. After saving the page, you will get a report on the page.

## Schedule Refresh

This section explains how to refresh the data in reports. You can schedule refresh in a timely manner. Reports can be refreshed eight times per day. Follow the steps to schedule refresh.

1. Open the Power BI app and gear icon and click on “Settings.”
2. Click on Datasets and click on the file which you need to schedule refresh.
3. Go to “Scheduled refresh” and change it to “On” and add the time, click on “Apply.” You can add another scheduled time.

## Schedule an Automated Refresh

This is an important feature of PowerBI. If data source (In your case SharePoint List) gets updated, your report should get updated on SharePoint Page. You can schedule refresh for your dataset. Click on “...” of your dataset and select **Schedule refresh**.

## Schedule Refresh

- Keep your data up to Date: On
- Refresh Frequency: Data can be refreshed daily or weekly.
- Add another time: You can set your own time when you want to refresh your data.
- Send refresh failure notification email to me: Check this option for getting the mail notification when your automatic refresh gets failed.

## **Change Data Source**

This section explains how to change the data source. Consider the SharePoint site is migrated to another environment, the Power BI report also needs to be migrated, in that case, you have to do some changes in the Power BI desktop file and publish it to SharePoint. Make sure that whatever lists used for reports should be the same in the Migrated site also. Below are the steps to change the data source.

1. Open the provided “pbix” file in Power BI.
2. Click on ‘Home’ in the Ribbon menu and go to Edit queries. Choose ‘Data Source settings.’
3. Click the ‘Change Source’ button.
4. Provide the URL of the SharePoint site and click close.
5. A pop-up to sign in to the portal opens. Choose the Microsoft account. Select the URL from the dropdown and click sign-in. Provide the credentials and click Connect.
6. Click on ‘Apply changes’ in the ribbon. A popup listing all the errors will open. Click on Close.
7. Click the ‘Home’ menu again. Click on the ‘Edit Queries’ option and choose Edit Queries.
8. A page with all the lists with errors in the left navigation pane will open.
9. Click on ‘List.’ From the Query settings in the right pane, click the gear icon near “Navigation.” Select ‘List’ from the pop-up and click OK.
10. The Query Settings will now have “Renamed Columns1” in

addition to “Renamed Columns,” which was present earlier. Delete “Renamed Columns.” (If there are more than 1 Renamed columns with different suffixes, delete all the older ones and retain only the newly created one.)

11. Repeat steps 8 and 9 for Waves, Sites, and Libraries. [Note: The warning sign in the left navigation will be changed to the list icon].
12. Click on Close & Apply.
13. Once the page is loaded, click on Publish and follow the steps explained in Publish to Power BI App and Embed in SharePoint Online.

## **Share the SharePoint Page with PowerBI Report Embedded in It**

SharePoint page can be shared with office 365 users (SharePoint Online users). All users should have Office 365 account with PowerBI service. However, Power Bi report is embedded into this SharePoint page, to view these reports to SharePoint Online users must have access to PowerBI reports also.

It means that you need to manage permission on both locations, PowerBI report, and SharePoint Online Page.

**Access to SharePoint :** Open SharePoint Page where your PowerBI report is integrated. On the right hand, there is a SHARE button. Click on it. One popup will open to add users for whom you want to give permission to this page.

Enter name or mail ID's of office 365 users, select permission level as per your requirement, and click on Share .

**PowerBI Permission :** The user also needs permission in PowerBI to access the report. To handle the permission, you can use Manage Permissions. Go to Datasets and click on the manage permissions. Click on Add Users.

Enter office 365 user's name or mail ID whom you want to share the PowerBI report. Click on Add. After assigning permissions on both locations, the user can view or edit the report as per the permission level.

## **Hosting of Power BI Reports in SharePoint**

The main prerequisites to host the Power BI Report in SharePoint are:

- The SharePoint site should have been created to host on the site.
- The Power BI Account should be created

To Host the power BI Reports in SharePoint, you need to publish the report. To publish the report, open the report in Power BI and select publish to Web option. This will open up with a dialog box showing Embed code and iframe details. Make a note of these details.

- Go to the SharePoint site to which the Power BI Report needs to be hosted.
- Click on the settings icon in the site and click on Add a page option to create a web page.
- Give proper Name and open the page in Editable mode.
- Click on the insert web part option from the Ribbon.
- Go to the Media and content category and in the parts select Script Editor option.
- Script Editor is added to the page.
- Click on the Edit Snippet link in Top Right of the page.
- Now copy the HTML iframe code and paste it to the snippet editor.
- A representation view of the image will be shown on the page.
- Finally, click on insert, save, and publish the page in SharePoint.

## **Publish the Power BI Report on Dynamics 365**

Microsoft Power BI is a suite for business analytics. You can use Power BI to analyze the data, explore data, and create rich reports quickly.

A business analyst or a developer can use open data protocol (OData) endpoints to create Power BI reports. Consider the Dynamics 365 for Operations application is configured to allow Power BI reports.

1. Open the Power BI desktop application and click on the sign-in button.
2. Provide the User Id, which has power BI access and clicks the sign in.

3. A window opens prompting for the password. Enter the password and click on the sign to access the Power BI desktop.
4. In the Power BI desktop, click on the “Get Data” option in the action pane and then click OData Feed from the options listed.
5. A window pops up for OData Feed prompting to enter the URL. Provide the URL of the Dynamics 365 application by appending “/data” at the end and click on the Ok button.
6. An Access window for OData opens where the Sign-in can be changed, or you will be prompted to sign- in case it is not done in the first step when opening the Power BI Desktop. Now click on connect.
7. Once the Power BI desktop is connected to the OData URL provided, it lists all the tables available from the dynamics365 application.
8. Select the required tables from the list and click on “Load.” Now the Power BI desktop application loads the table and all the columns within it on the right side under Fields section.
9. Select a graph pattern from the visualization section by clicking on it. Drag and drop the fields into Axis and values section. The graph gets updated based on the fields and values selected for the visualization.
10. Now Click on Publish. You will be prompted to save the report if it is unsaved. Once you save the report, it gets published to power BI.
11. Sign to Power BI online with the same credentials used for Power BI desktop. Navigate to the Reports tab under My workspaces, and the published report could be seen. Click on the report published from Power BI desktop. Once it opens, click on the Pin Visual option.
12. A window opens, asking if the report has to add to an existing dashboard or a new dashboard. Select New Dashboard and provide the name for it. Click on the Pin button.
13. Navigate to dashboards under My workspaces, and the newly created dashboard is available. Click on the SalesLine Analysis dashboard once it is available in the Power BI online dashboards, Open Dynamics 365, for operations.

14. Create a new Workspace by right click > Add Workspace. Provide a name for the New Workspace.
15. Now click on the Workspace “Power BI Report Space” that is created.
16. Now for the power BI control to be available in the workspace, navigate to the below path in the workspace created. Options> personalize this form> click on + sign> Allow Power BI control> click OK.
17. You can see a Power BI Tile in the workspace. Click on the “Get Started” option. It opens a Tile Catalog, showing all the Power BI dashboards that can be used in dynamics 365.
18. Now select the dashboard by clicking the title. Now click on the report, so it is Ticked and Click on OK. The dashboard selected is now displayed in the New Workspace created. You can add multiple graphs/reports to the same dashboard.

You can use this tool to analyze and create quick reports and remove the performance bottlenecks to improve the performance of the whole application.

These reports are not only rich and interactive and but also users can make changes without having to rely on another person. Users can pin reports to workspaces themselves.

# **Power Query for Report Generation**



## **Introduction**

Almost all of the testing projects require data to be tested/analyzed based on the requirement. Many of the testing applications require the testers to set up data, whether it might be simple or complex. This document helps to understand how a user can easily get different data Sources for Data Preparation/ Test data, modify the data as per the user's needs (like removing a column, changing a datatype, or splitting a single column) and for Data analysis.

Power queries can act as an ETL tool and also can be used to build reports adhering to the business requirements. It is a part of the larger domain provided by Microsoft, i.e., Power BI. A power Query is a user-friendly way to play with the data. Power Query is a free Excel add-in that allows the user to import data into excel from many data sources, allow data modification, refine data, and load to the data model. This applies to Excel 2016, Excel 2013, and Excel 2010.

With Power Query, user can:

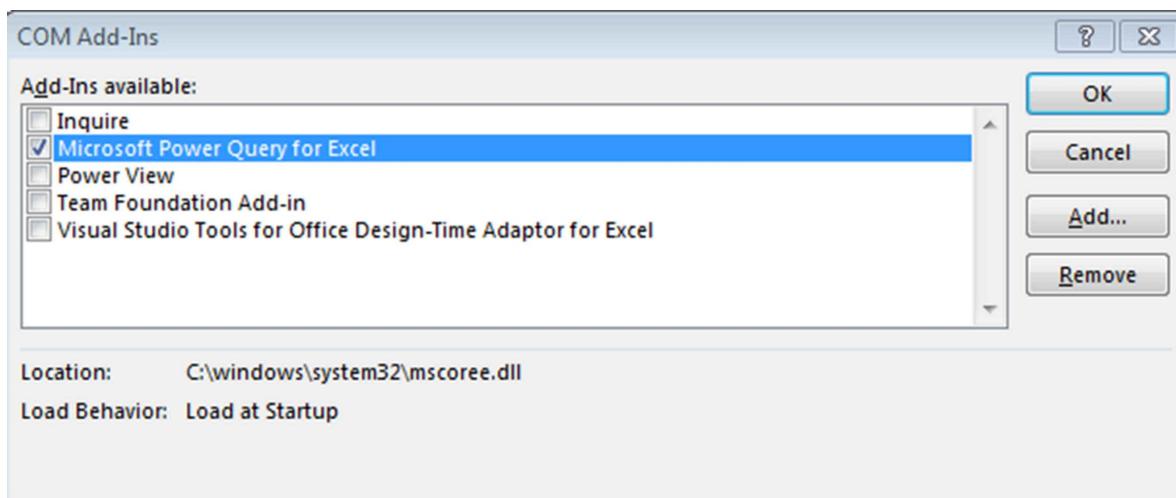
- Connect to a variety of data sources.
- Can bring in the data from these sources and transform the data which matches the requirement or prep it for further analysis using tools like Power Pivot or Power view.
- Create custom views over the transformed data.
- Perform data cleansing/modelling/integration operations.
- Data can be imported from multiple log files.

This section will mainly focus on connecting to the ORACLE database and generating excel reports.

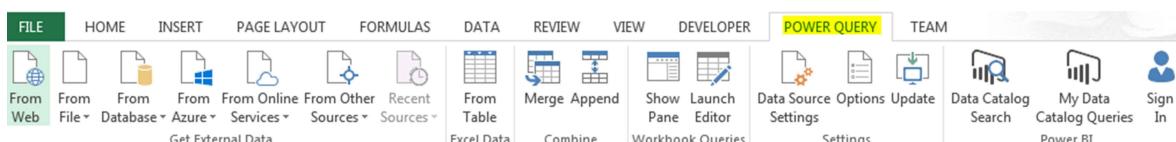
## Setting up Power Query

Power query can be downloaded from the following link [Power Query Download](#). The link also has other requirements that specify the software requirements for installation.

1. After installation, navigate to the FILE menu. Click on OPTIONS to enable the use of add-in.
2. Now select the Add-Ins tab, which is located on the left side of the screen. Go to Manage Combo Box and then click COM Add-Ins. Finally, select the GO button.
3. When you have the dialog box on your screen, check the checkbox for “Power Query for Excel.” Once done, select the OK button to continue.



4. After doing the above steps, there will be a new Power Query tab present in MS Excel, as shown in the below image.



## Power Query for Excel

You will consider the ORACLE database as your source.

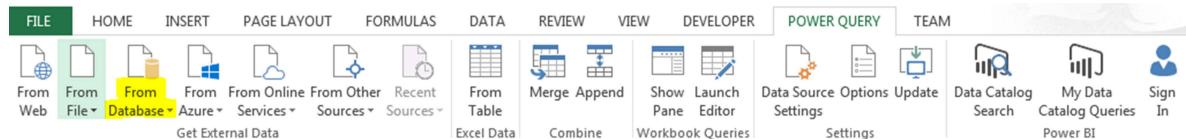
There are mainly three parts to it –

- Connect to the database

- Data Load
- Data Refresh

## Connect to the database

Under the Power Query tab, click on FROM DATABASE icon as shown below:



Click on FROM ORACLE DATABASE. Provide the SERVER (SID/database name) and QUERY.

Then provide the credentials of the database you want to connect.

## Data Load

The user can see the data in a popup and load the data into the desired excel sheet.

## Data Refresh

Right-click on the workbook query and click EDIT.

A Query Editor window pops up. Click on the Data Source settings tab.

Then change the source to some other environment and edit the permissions by giving the credentials.

After successfully providing the credentials and connecting to the database, refresh the preview and then close the query editor and load the data to the main sheet.

The user can also refresh the data from a link button present in the workbook query window tab (right side of the screen near the black arrow).

## Simple Installation

- Excel 2016 - Power Query is a built-in tab in Excel 2016. It has been renamed to the Data tab. It is known as “Get & Transform” in Excel 2016.

- Excel 2010 & 2013 – Power query has to be downloaded from the below link and installed.

<http://www.microsoft.com/en-us/download/details.aspx?id=39379>

Once the installation is completed, open excel, and the user will be able to access the Power Query tab.

Incase power query tab is missing, follow the below steps:

- 1) Go to File > Options > Add-in.
- 2) Select COM Add-ins from the Manage drop-down.
- 3) Click the Go button.
- 4) Check Microsoft Power Query for Excel checkbox and Click on OK.
- 5) Now power query tab will appear on the screen.

## **ETL Process**

ETL (Extract, Transform, and Load) is a process where data is extracted from various data sources and transformed in a proper format for analysis and querying and loaded into the destination.

### **Extract**

#### **Data Sources**

Power query can transform both structured and unstructured data from various sources. Different sources include Excel, Web Page, XML file, Text File, Folder, HDFS, Facebook, and from different databases such as SQL, Oracle, DB2, MySQL, Sybase, Teradata, etc.

If you want to import data from the web, you need to enter the URL in the textbox, so that the user is navigated to the desired web page.

The contents will be loaded in the form of tables on the extreme right of the sheet under the Navigator section. If you hover over the tables specified under the navigator, details of the table, along with the column names, will be specified in the display box for a quick view to the user.

Due to access restrictions and for easier understanding, you shall import data

from another Excel file. Say, you import data from Excel. Data imported from excel will be displayed under Navigation pane. Users can load multiple files by selecting the ‘Select multiple items’ checkbox and checking the desired files.

‘Load To option’ in the navigator pane allows viewing the data in the way the user wants. Clicking on ‘Load’ will load the data.

## **Transform**

Query and Design tabs will get enabled to the user.

- Query tab is used for the transformation/shaping of data.
- The design tab helps in improving the look and feel.

Once data is loaded, it will get displayed on excel automatically. The count of the number of records loaded will be displayed under Workbook Queries.

## **Query Tab**

Click on ‘Edit Query’ in the Query tab. Users can now modify/transform data without using the normal excel formulas and coding.

### **Operations in the Query tab**

Below are the operations that can be performed in Edit Query:

1. Choose Columns - Allows the user to select the columns that are needed by the user.
2. Remove Columns - Remove the selected columns from the table.
3. Remove Other Columns - Remove other columns from the table except for the selected ones.
4. Move - Columns can be reordered.
5. Keep Rows - Keep top rows/Keep bottom rows/ Keep a range of rows- based on the number of rows specified.
6. Remove Rows - Remove top rows/Remove bottom rows/ Remove alternate rows- based on the number of rows specified.
7. Remove Duplicates - Remove rows containing duplicate values for the selected columns.

8. Split Column -Split values in the selected column based on the delimiter/number of characters.
9. Group by - Group rows based on the selection of columns.
10. Datatype - Change datatype for the specified column(Text, Decimal, Whole Number, Logical, Date, Time, Date/Time/Timezone, etc.).
11. First Row as header - First row of the column can be used as a header.
12. Replace value - Existing value can be replaced with the specified new value.
13. Append – Consolidate data from different sources and from multiple tables into one.
14. Merge – More like a join operation to add data.

Values can also be sorted ascending/descending.

## Merging Two Tables

If you want to merge two columns together, given the need, Power query will evaluate the selection and will give the results on how many matches the selection will return.

To Merge Orders and Returns table, click on the Merge option present in the ‘Power query’ ribbon. Select the tables and matching columns to create a merged table. Matching column act as a join condition.

A new column will be added at the end of the table, which has to be expanded to see the original results.

After expanding the columns, you can see the columns that are merged. Unmatched rows are displayed with the value null.

## Load

Close and Load: Save the changes to the query, close the query editor window, and load the data to a new workbook.

## Applied Steps

The applied steps section shows the operations applied to the data in sequential order, which helps to keep track of the transformation process. These steps can also be deleted.

Thus power query excel makes it easy to get data in good shape. It has all the common transformations. This is of more use comparing to VBA macros which use coding in data transformation.

## **Synoptic Panel in Power BI Desktop**

One of the key components of Intelligence from Business Intelligence is to find trends and patterns over various periods of time and be able to compare them across. These “Periods of time” have a large generic form or structure such as Month over Month, Year over Year, Same Month Last Year Vs. This year. Power BI Visualizations require DAX’s Time Intelligence functions to calculate through them. The formulae using DAX functions take different shapes depending on the type of measure as being additive or semi-additive or non-additive. Bringing Contexts and Filters in the mix can sometimes make it very complicated to understand.

Let us see synoptic panel by SQLBI and their usage in Power BI Desktop. And also how to customize a visualization using it in Power BI Desktop.

## **Synoptic Panel by SQLBI**

A synoptic designer is a tool of Synoptic Panel by SQLBI. It will allow us to draw customized areas, maps, images, etc. and export it as SVG file to Power BI. Then using it, able to color the different areas based on your dataset data.

The Synoptic Panel connects regions of the image with attributes in the dataset and coloring the regions or with a saturation of a color related to the value of a measure. For any image, you can draw custom areas or regions using the Synoptic Designer tool, and an SVG file is generated, which you import in the Synoptic Panel.

Now you will try to implement one of the travel systems which owns cars; you need to find whether the seats are available to book the cab on a sharing basis or not.

1. Get the desired image on which the dashboard has to be created; here, it

is the car seating arrangement. Let the image look as shown.

2. Now open the URL <https://synoptic.design/>
3. Drag the image or click on the insert an image path/URL, then navigate to the image file location and import.
4. Click on Draw new areas icon. Then go clicking the areas required. Give the appropriate names to those regions as per the data in the dataset. Note: The region names and the data values in the dataset should be the same.
5. You can automatically say the designer to detect the remaining regions by clicking on the tab “automatically detect bitmap areas,” or else you can detect them manually as earlier. And then give appropriate names accordingly.
6. Now click on EXPORT TO POWER BI. Click on DOWNLOAD SVG and save it in the desired location.
7. Now download and import “Synoptic Panel by SQLBI....” visualization and click on it. Select a few of the columns as per the requirement; it will ask for a select map or Gallery. Select one of the options. I click on the Select map, and I will import the SVG file created using the designer in the earlier step.
8. The column values matched with map region names come under Matched areas.
9. You can specify the colors of the matched and unmatched regions to a single color or multiple colors in the Format tab, as shown. Enable the Labels button to show the values.

Now you will try to implement the colorization of the regions based on their availability. I.e., if the seat is available, it should display in green; it is not available or booked, show it in red and if it is the driver seat, show it in yellow.

For this, I am adding a new column as below.

Color = IF(Car[Status] = "Available", 1, IF(Car[Status] = "Booked", 2, 3))

Place the columns values as shown, and give the values for State 1 From = 1, To = 1 and color = Green, for State 2 From = 2, To = 2 and color = Red and for

State 3 From = 3, To = 3 and color = Yellow

Similarly you can achieve any desired customized visualization as per the requirement using Power BI Desktop.

# **Filters and Slicers in PowerBI Report**



## **Filters in PowerBI**

Filters in PowerBI are used to view the data only, which you want to focus on. Filtering will remove the data from the visual temporarily, and it can be viewed back once the filter is opted out. Filters enhance the visual representation of the report.

There are four types of filters in PowerBI.

- **Page filter**
- **Visual filter**
- **Report filter**
- **Drillthrough filter**

## **Different Types of Filters in PowerBI**

### **1. Page Filter**

Page filter applies to all the visuals on the specific report page. In a PowerBI report, you can create multiple pages. As the name suggests, this page filter gets applied only at the page level. Let's consider you have created a PowerBI report of multiple visuals from a single dataset on the first page.

If you want to apply the filter to another visual from the same page (populated from the different dataset), the relationship has to be created for both the datasets to enable the page filter.

Now when you apply the page filter, all the visuals get filtered based on the filter. You can also have multiple columns added to a page filter.

Example: In the right pane (VISUALIZATIONS), under the FILTERS, you will

be able to see the PAGE LEVEL FILTER. Drag the necessary column from the dataset and place it in the filter.

Now once you select a specific data from the Page-level filter (e.g., Australia), the whole page with all the visualizations gets filtered.

## **2. Report Filter**

The report filter applies to all the pages in the report page. A report filter is very much similar to the page filter. While in page filter, it applies at the page level. In the case of the report filter, it gets applied at the report level. All the other properties remain the same.

Example: In the right pane (VISUALIZATIONS), under the FILTERS, you will be able to see the REPORT LEVEL FILTER. Drag the necessary column from the dataset and place it in the filter.

Now once you select a specific data from the Report level filter, the whole report with all the visualizations gets filtered across the pages.

## **3. Visual Filter**

The visual filter applies to a single visual on a report. This filter gets applied only at the visual level, even if you have created multiple visuals from the same datasets.

When you apply the visual filter, only the specific visual gets filtered based on that filter. Similar to page filter, you can have multiple columns added to a page filter.

By default, all the columns in that visual will get added to that filter. In the right pane (VISUALIZATIONS), under the FILTERS, you will be able to see the VISUAL LEVEL FILTER. All columns in the dataset will be added by default in the filter.

Once the visual filter is applied for the country ‘Australia,’ only the specific visual is filtered, whereas all the remaining visuals remain the same, not filtering any data.

## **Generic Properties for Page, Report, and Visual Filters**

There are two filter types in the Page, Report, and Visual filters.

1. Basic Filtering
2. Advanced Filtering

Basic Filtering is filtering the data by just selecting the column values present in the dropdown. Advanced filtering consists of many options such as ‘contains,’ ‘starts with’ etc.

### **1. Drillthrough Filter**

With the drillthrough filter in Power BI Desktop, you can navigate from the one page with high-level data to the destination report page with detailed data - such as a product, or sales, or country. From any report pages, you can click on data for that entity and drillthrough to the destination page.

Drillthrough filters help us in situations where you have to navigate from the main page with high-level data to the detailed data visualization.

Let’s take the below example to see the functionality of the drillthrough filter.

Consider you are having two pages in the PowerBI report named as ‘Highlevelview’ and ‘DetailedView.’ In the DetailedView, you have all the salesdetails information, which includes the details salesperson, color, country, and salesprofit.

### **DetailedView Page**

In this DetailedView page, you are having all the sales details with respect to country, salesperson, color, etc.

### **Highlevelview**

In the Highlevelview page, you have the map visual in which you have added the ‘Country’ as location and ‘Color’ as a legend in the fields.

We have added the Country and Color columns in a high-level page, as we need

to filter the detailed data in the DetailedView through these two fields. Now to enable the drillthrough filter option, you have to add those two columns in the ‘Detailedview’ page where you have the detailed information of data.

Once you add the two columns in the destined page, a back button will be enabled and shown on the page. Any image can replace this back button. If you want to replace this image, you have to insert the new image on the same page and make the action as ‘Back.’

In the right bottom of the page, you have added the columns added in the Drillthrough filter. Now when you right-click on any color area in the map and click on the **DetailedView**, it will take you to the **DetailedView** page. (Country- Australia, Color: Green).

Now you will be able to see the DetailedView page. Your DetailedView page with all the details is filtered based on the (Country- Australia, Color: Green).

A **back arrow** on the top can be used to navigate back to the **Highlevelview** page.

## Slicers

Slicers are the types of visualization in PowerBI, which helps the readers by enhancing the readability of the report. A slicer is another way of filtering, which will narrow the data populated in the dataset.

Instead of creating multiple reports dataset to a different set of readers, you can use slicers to slice the data according to them.

### Slicers can be used to:

- Display the most often used filter on the report.
- The currently filtered state is much visible.
- Filters the necessary data and hide the unwanted in the data tables.

Though Slicers are similar to Filters, it has its own advantages and limitations. Usage of Slicer/Filter must be decided based on the user’s requirement.

There are different types of slicers available. You can import the slicer from the PowerBI visuals. Some of the examples are given below.

- Chiclet Slicer – Displays image/buttons that act as a filter.
- Timeline Slicer – Graphical data range selector for date filtering.
- Attribute Slicer – Visualize and search data attributes
- Hierarchy Slicer – Slicer with a hierarchy of different fields.

### **Assigning Slicers to Selective Visuals:**

We have an option in PowerBI to assign a slicer to selective visuals. If you do not want to filter the visual even when sliced, you can opt-out that visual from that slicer. To enable that, select the slicer visual ‘Country,’ click on the Format tab, and then select ‘Edit Interactions.’

Now you have enabled the filter for all the visuals except ‘SalesProfit by SalesPerson and Country.’ If the ‘Filter’ option is selected, that visual will be enabled for that visual. If the ‘None’ option is selected, that visual will not be enabled for that visual.

Once you are done with the changes, when you slice the data using the slicer, it will filter all the visuals except the one which you marked as disabled. In the country slicer, select ‘Australia,’ and all the visuals have been sliced other than ‘SalesProfit by SalesPerson and Country’ visuals.

### **Syncing Slicers**

In PowerBi, you can also sync the slicer across the pages. If you have to slice the data in the visuals across all the pages using a single page, you have to do the below steps.

- Click on the slicer.
- Click on the View tab and enable sync slicers.
- In the right pane, sync slicers will be enabled. Select the pages which you enable the slicers.

### **Slicers vs. Filters**

#### **Use Slicer when:**

1. Number of Values in the column is limited.

2. If there is any frequently used column to filter data.
3. The user needs to know by glance for what scenario data is filtered.
4. The dashboard is used more in mobile PowerBI applications.

### **Use Filter when:**

1. Data volume in the datasets is larger.
2. The column is not used frequently.
3. PowerBI Dashboard is used more in laptops/desktops.

## **How to Toggle in Power BI Report Using Images**

Power BI is an enhanced BI reporting tool which helps us to fetch the data from different sources to create and publish dashboards and BI reports.

### **Toggling in Power BI**

Most of the business users' requirement is to view the data in different visuals such as charts, tables, graphs, etc. With default configurations in Power BI Desktop, you are not having the option of Toggling between a chart and table, or table to graph or vice versa. But in Power BI toggling can be done with the help of Bookmark and Selection Pane.

### **Steps Involved**

Step 1: Create a Power BI report with necessary datasets from the sources.

Step 2: Create tables, charts, and graphs based on your requirements.

Step 3: Create some images in the Power BI page for toggling.

Step 4: Enable Bookmarks and selection pane in the Power BI report.

Step 5: Link the images to bookmarks.

In Toggling, you map an image with bookmarks, where bookmarks are mapped with visuals. So when you select an image in your report, it will display the visuals which are mapped through bookmarks.

Assuming you have already created a Power BI report with necessary tables and charts, let us see the process from the 3<sup>rd</sup> step.

### **Step 3: Create images in the Power BI page for toggling**

#### **How to add text box and images in Power BI Report**

To add a text image, you have click on the ‘Image’ button on the HOME tab of Power BI desktop, browse and select the image. Place it accordingly. Similarly, create a ‘Text box’ and type in whatever text you want to put in. Now, considering that you have already populated the required charts, tables, and other required images in addition to the previous step, your Power BI desktop will show everything on your page. (All the images, tables and charts).

### **Step 4: Enable Bookmarks and selection pane in the Power BI report**

GO to the VIEW tab of the Power BI desktop and enable the ‘Bookmarks Pane’ and ‘Selection Pane.’ Now in your Power BI desktop, you will be able to see the SELECTION and BOOKMARKS pane on the right-hand side. In the SELECTION pane, whichever visual you have on your page will be listed. In the BOOKMARKS, it will be empty unless you create one.

#### **How to create a bookmark**

Click on the Add button to create a BOOKMARK; a bookmark will get created with the default name. You can rename or update or do any of the options based on your requirement by clicking the three dots near the bookmark.

Once you are done with creating a bookmark, you should select the images, which have to be mapped to that bookmark.

You can do that by clicking the ‘eye’ symbol in the selection pane. Visuals with the ‘eye’ symbol will be mapped, and visuals with the ‘hyphen’ symbol will be hidden for that bookmark. Click on Using the same way.

### **Step 5: Link the images to bookmarks**

As you have created all the necessary bookmarks, now you have to link an image with the bookmark so that if you click that image, it will show you the visuals which are bookmarked.

You can select the image (left bottom) in the Power BI report and map it with the bookmark ‘HOME.’ So when you click that image (HOME), it should display the images which are mapped to the bookmark HOME.

To do that, first, select the image, do the below changes in the third pan ‘FORMAT IMAGE.’

1. Make the action type as ‘ON.’
2. Select the Type as ‘BookMark.’
3. From the bookmark dropdown, select the bookmark ‘HOME,’ which you have created earlier.
4. Now do the same linking process of the image to all the necessary bookmarks. Once you are done with linking all the images with bookmarks, save and close the other pans.

If you click the image, it will show you the visuals which are marked in the bookmark in which the image is linked.

# Cognos Self Service BI



## **Self Service BI: Overview**

“The specific function designed within the BI environment to give BI users the ability to become more capable and less relied on IT.”

To meet this emerging need and improve time to build, companies are looking for different choices for BI. One such approach is to build the infrastructure that supports the BI environment in which the Power users can collaborate specific sets of BI reports, Dashboards, Logic, and analytics with less dependence on IT.

The advantage of self Service BI is to extend the scope of BI applications to address a wider range of business users and help them solve the needs and problems of complex scenarios. At the same time, this extension must support the information workers need for a personalized and collaborative decision-making framework.

It lists down a few business scenarios where self-service capabilities of the BI platform make a difference to an organization’s ability to meet its business goals. After establishing a business case for self-service BI, the document gives a brief introduction to Self Service BI environment. Without going into technicalities, it lays down the key features that can enable a self-service Bi environment.

Self Service Business Intelligence is much under discussion these days. Users find traditional BI platforms to be too complex and technical for them to use, and at the same time, the IT team finds it difficult to meet the ever-rising information needs of business users. The self-service environment aims at removing the distance between the people who generate a report and people who actually use it.

To meet the rising information and analytical needs, there is a need for BI users to be more empowered and BI tools to be more exhaustive. From a functional point of view, three things make the base for a self-service environment - an

interactive and simple user interface, a set of tools that serve changing needs and accommodate new work practices, and a robust backend data support to allow the first two. There are several tools from various vendors that provide self-service capabilities in some form or another.

Four key objectives of Self Service BI:

- 1) Make it easy to use.
- 2) Make it easy to access data.
- 3) Make the DW solution fast to manage and deploy.

## **Self Service BI: Challenges**

Implementing Self Service BI is more difficult than BI professionals anticipate.

**Differing User Requirements:** Business requirements keep changing. BI Solutions must be implemented quickly to take advantage of the business strategy. The timely market solution is the key to Self Service BI Solution's success.

**The Balance:** Self-service BI requires clean, comprehensive, and integrated data, and a few of the key factors are:

- Know your audience: There is no “One size fits all solution in BI.” Different BI users need different types of solutions to suit their needs. Different business user interprets data differently
- Training: A significant number of Self Service BI Solutions fails because of proper training. Training doesn’t pass on the exact know-how of the Tool and data.
- BI Tools: BI Developers need the ability and functionality to create reports and dashboards; however, the BI Power users or casual users need the Tool features and functionality to be simple and easy to navigate through. In the majority of cases, Self Service BI Solutions have often implemented a good BI tool but has not able to reap the benefits of self-service BI.

## **Data Governance**

- a) Poor Data Quality: Abnormal definition of the same metric has

been a critical reason in getting poor data quality. The same metric may have various definitions across different departments of the same organization. The definitions or KPI can be created based on functional rules which are based on functional rules which are the same across the whole organization.

- b) Data Integration: Data Integration rules comes into picture when data has not been formatted before aggregating
- c) Logical Errors: Logical errors are the addition of the data integration issues where a KPI or metric definition is taken out of context to do the analysis.

User can't afford to wait:

**Solution: Get Self-Service Right!** Training should involve the Functionalities of the tool and which data elements and fields can help them with their analysis. It should also include security policy recommendations.

## Why Self Service is Required in Today's World

This trend of Self Service BI was started by Business objects with Web Intelligence and was marketed as an Ad-Hoc reporting tool. This tool did provide the business users the capability of Self Service BI; however, packed with powerful features; it became difficult for business users to understand queries and functionality.

With changing times and usage patterns, features of BI have changed. Over the past decade, BI has evolved from the crude Integrated Development environment to tools that have been easy to use and navigate.

Business users or analysts can now use these tools to analyze the data, create reports, and dashboards.

Primary reasons why Self Service BI is required:

- Next-Generation Solutions: BI tools provide excellent support for smart analytics and dashboard using the Big Data
- Time to Value: BI Solutions needs to be applied quickly to take advantage of changing business scenario

- Cost Optimization: Optimized hardware and software solutions with a lot of built for purpose Analytical solutions help businesses with less IT intervention.
- Data Understanding: Self Service BI Tools provide a lot of visualization features that help them understand the data better. They can create any number of graphs, charts, and analytical functions that help them see the data better.

Self Service BI is a BI platform that allows BI users to be more self-reliant and less dependent on the IT team of their organization.

### **What Makes the BI Self Service**

In simple words, Self Service BI is nothing but a user being able to get the data that she wants in a format she understands, and within the time frame, she has. Now there is a range of functionalities of a BI platform that enable business users to be more self-sufficient and BI tools to be more user-friendly. How many of these are achievable within the cost and a performance boundary is a matter of separate discussion, though.

If you don't go into technical details or in a layman's language, a self-service BI platform should be able to serve the following needs:

- It should have a very interactive, user-friendly, and intuitive user interface.
- It should have advance features to support evolving business needs.
- It should have a robust backend that can support the above two.

### **The Business Case for Self Service BI**

The job of IT is to support the business. In the end, the worth of any IT solution lies in the amount it can contribute to the bottom line of an organization. Hence let us go through a few business scenarios where an organization's existing Business Intelligence infrastructure finds it difficult to support day to day information needs of business users.

#### **I want it quick**

An organization 'ABC Motors,' a car manufacturer, faced a drastic reduction in

top and bottom lines during the recession, which started late 2008. Due to dropped sales and changing consumer preferences, it accumulated a huge amount of inventory, especially those of high-end multi-utility vehicles. On the one hand, it had to cut its IT spending by reducing IT headcount. On the other hand, it needed to enhance its responsiveness to changes in the volume and patterns of the demand.

For this responsiveness, its executives need to dig into data and do various kinds of analysis to find out new business trends and to respond to them quickly. Calling an already understrength IT department to fetch a few new reports in quick time was an exercise in vain.

### **I want to do it my way**

John, a marketing manager at online travel and ticketing company, wants to regularly perform varied kinds of analysis on his sales data to find the impact of various environmental conditions. Those conditions include the time of year, time of travel, weather conditions, holidays, promotional schemes, competition's promotional offers, new advertising campaigns, etc. The analysis and reports that he needs to keep changing every day. His need is to play around with data as and when he wants.

### **I want to drill-down whenever I want**

Dan is an executive assistant to the CEO of a large corporate house. While preparing presentations for regular meetings, he usually needs financial data at a very high level (Ex: Business Vertical, Territory, etc.). Huge reports with data presented at a very granular level are of little use for him. But sometimes, when a problem is detected in some part of the organization, to diagnose the root cause, he feels the need to drill down into the data to lower level granularities.

### **I want it on Excel**

ABC bank is a large investment bank with operations across the globe. Bank provides capital market research reports to investors who use its trading desk for their investments. Due to various market manipulation frauds that have occurred in the past, market regulator keeps a close watch on the research reports that investment banks publish. Hence ABC bank has established a centralized database where all the research data is stored. Governance and checks have been put at various levels to ensure that all the data and research work passes through

regulatory filters before reaching to publishing desk. The bank has instructed all its analysts to do all their research work on centralized servers. Ricky, a research analyst of ABC bank, has been working with MS Excel spreadsheets for a long time. He finds excel applications very intuitive and very convenient to use. As he keeps traveling across the globe to visit clients, he finds it a major bottleneck in connecting to centralized servers to do some analysis. It is also very convenient for him to email excel sheets to his clients and associates. For him, MS Excel is a full-fledged BI tool, and Banks' newly established BI platform seems more of a hindrance.

## **Need for BI Users to be More Empowered and BI Tools to be More Exhaustive**

All the above scenarios are not very uncommon in today's volatile market conditions. Business feels the need to respond to the challenges as swiftly as possible with the methods and approaches which are far away from being 'Standard.' With business scenarios changing every day, it is very difficult for system designers to anticipate all the possible business needs at the time of tool implementation. This leads to the following key drivers for a new set of BI tools:

- Volatile business needs.
- IT's limited ability to meet new requests in a timely manner.
- The need to be a more analytics-driven organization.
- Slow, restricted, and untimely access to information.
- Business users need to be more independent of IT.
- The complexity of traditional BI platforms.
- Lack of intuitiveness of existing tools.
- Constrained IT budget.

The solution lies in building a BI platform that is very flexible to accommodate changing business needs and in empowering users to serve their own needs with minimal intervention from or dependency on IT. In summary, it reduces the need for time and effort involved in building a report for a new idea.

These seem to be ideal features for any BI solution; at the same time, these are

unachievable without overstretching the costs and compromising the performance. Self Service BI tools endeavor to walk on the tight rope.

Let me briefly explain all the features of self-service Power BI. I am also trying to put some techno-friendly names to these business needs.

### **Interactive, user-friendly and intuitive user interface**

A business user is never concerned about what is happening at the backend. A sales manager is interested in knowing which product is selling in which market rather than in understanding the data structure of the database. Neither is he interested in going through a five-day training course for being able to use a so-called cutting edge business intelligence tool. Irrespective of what is happening in the backend, he wants a few simple things to happen on his desktop.

- Intuitive, nontechnical front- end: A user interface that is very interactive and uses a language that is familiar to business users.
- Office integration : Making BI results available on MS office interface, with which users are very conversant.
- Advanced visualization : A picture speaks more than a thousand words. The best way of making something simple to understand and use is to present it in graphs, charts, and pictures.
- Guided analytics : Proactively alert users and automatically provide pathways to additional information and insight.

### **Features and Functions**

Business expectations from BI are evolving rapidly. A traditional text-based report is no more sufficient. To support the business decisions in a desirable time frame, BI should have features and tools which make it easier to work all level of users and strong enough to support analytical and presentation needs of most sophisticated users.

1. Predefined report templates : These allow users to change content as well as the layout of a report by simple drag and drop functionalities.
2. Parameterized reports : This allows users to change the content of the fixed layout report.

3. Mash boards or Custom Dashboards : To enable power users to craft dashboards for themselves and others by selecting elements from existing reports and external Web pages.
4. Stored analyses: Allows users to store reusable analysis work to be used later by other users.
5. BI Web widgets and mash-ups : Small applications that can be installed and executed within a Web page by a user.
6. Sophisticated analytical processing and algorithms / functions .
7. Statistical functions : Allows advance statistical analysis like predictive analytics.
8. Geospatial Functions : Interactive maps and geographical presentation.
9. Workgroup portal and collaborative working: It helps in sharing results, enables better governance, and optimizes network usage by eliminating the need to publish results to individual users. Users can retrieve BI results from the portal on an as-needed basis. Also helpful are the techniques like tagging, ratings, comments, and information collections that help in identifying related content both inside and outside the organization.
10. Decision workflows: These help users to make decisions based on best practices. It not only sets the structure for decision making but might also make available necessary data and expert opinion from various sources.
11. Automated Triggers : It allows automatic trigger of some actions/ workflows based on business rules and predictive models.

### **A robust and swift backend**

There should be a robust data warehouse to support all that we have discussed so far. But this is not limited to data structured in one warehouse. It has become increasingly important for a user to understand the complete picture. To gain contextual knowledge, the user must have access to external data (e.g., weather information, geographic, demographic, or psychographic data, comments or e-mails, social media sources). Also, a data warehouse should be fast to deploy and easy to manage so that changes can be incorporated swiftly.

1. Data virtualization : Because of the diverse business needs, it is very difficult to incorporate all the required data into a single data warehouse. With the help of data virtualization and data federation technologies, it is possible to build virtual business views that allow applications to access data from various physical sources. These tools also support access to different types of data sources, including relational databases, non-relational systems, web services, Web data feeds, application package databases, flat files, etc.
2. Cloud-Based Data Platforms : DBMS might be deployed in cloud for faster deployment and lower upfront costs. It's a new trend, and many vendors provide this.
3. Browser-based dashboards : Deliver flexible access to information that has been filtered and personalized for user identity, function, or role—including enforcement of predefined security rules.

## The Tradeoffs

It might sound that SSBI is a panacea or a holy grail that can meet all the demands that businesses put in front of a BI platform. The reality, on the other hand, is far from it. On the one hand, a self-service environment empowers users to do a lot of reporting and analytics on their own. Still, on the other hand, it exposes the system to improper usage and might make it more complex for casual users.

- a. Power users might use SSBI to query very large data sets, putting huge pressure on query performance.
- b. Inspired by a new set of sophisticated tools, power users might create thousands of reports with conflicting and inaccurate data, making it difficult for casual users to find a relevant report.
- c. Many users might end up running a similar query putting unnecessary pressure on system and network.
- d. Making data available from various sources using data virtualization or shared data views helps the user to get a contextual sense of analysis. Still, this approach often harms performance.

**Available Tools :** Many vendors, both big and small, have come up with self

capabilities added to their BI offerings. Following are a few of those tools:

- Microsoft: Power Pivot
- Tibco: Spotfire
- SAP: Business Objects Explorer
- IBM: Cognos Express
- Information Builder: Web Focus Visual Discovery
- Open Source: BEE Project, Jaspersoft, Pentaho, SpagoBI

## Conclusion

---

So this was all about the strategies to learn the functions of Power BI and Power Query. By following the above approaches, you can easily learn Power BI on your own and become a master in power BI. The only thing you need to focus on while learning Power BI and Power Query is that all steps need to be followed properly. You can't skip any step; otherwise, you will not be able to utilize all the functions of Power BI simply and efficiently. You will be able to generate intuitive reports easily after going through the book.

# **Power BI**

---

*An Advanced Guide to Learn  
the Advanced Realms of Power BI*

**DANIEL JONES**

# **Introduction**

All businesses are running with the help of data in the form of emails, printouts, reports, excel sheets, etc. As the business grows, the data also gets accumulated and reaches a certain level where it is nearly impossible to find the data you need. In fact, when you have a huge amount of data, it is possible that you might miss the latest information and using the old information.

Power BI is one such tool that helps you in getting your job done in an easy and efficient manner by transforming the data in the form of charts, reports, graphs, and other visualization techniques. This will make your data more meaningful and easy to understand with a few clicks. Data insights can be viewed using intuitive and colorful visual designs instead of using old tables, lists, and bullets.

Power BI allows you and your co-workers to work simultaneously on the reports and dashboards using a web browser or smartphone. All the data gets refreshed and synced automatically, which means you are always working on the latest data. Thus, no need to worry about the manual revisions or asking your co-workers to maintain the tracker for changes. Power BI performs all these tasks for you on its own.

Since the data is now dynamic in nature, you are free to check your data in an advanced manner for insights, trends, and BI functions. Power BI is such a powerful tool that you can slice the content on your own and also ask personalized questions using your own language. All this information is available 24/7 anytime, anywhere. You will receive alerts from Power BI once the data is changed from the threshold value set by you.

## **Know Your Role in Power BI**

Your job role in the organization decides how you will be interacting with Power BI. You can be a user or a manager. As a manager or a client, you will be the one who will be receiving all the reports, dashboards, and other visualizations to review them. You will then interact with the stakeholders to make important business decisions for the growth of the organization. You might also be using the online Power BI service from your device. For this role, you don't need to be a data scientist, as you will not be processing and analyzing the complex data.

Power BI will do all that work, and it will provide you straightforward results.

The second role is of the developer or designer. If you are a developer or the designer, then you will be using Power BI extensively to deal with the huge amount of data. You will create business reports, use Power BI services, create intuitive templates & designs, collaborate data with your co-workers, and more. This is an essential role as it plays an important part in using Power BI in an advanced manner. You will be using Power BI differently for different projects because the requirements are different for every project.

## **Know More about Power BI**

It is a collection of services, mobile apps, web apps, connectors, and cloud-based framework working together that combines unrelated data into visually appealing insights. You can easily connect your data, whether it is an Excel sheet or cloud-based, in a very easy manner and then share it with your colleagues.

Power BI is very fast and can create insights using any database. It is robust in nature and works in real-time analytics. It can be customized as per the needs and requirements to suit the client's needs.

Now coming to the Power BI elements, there are mainly three elements:

1. Desktop
2. Service
3. Mobile apps.

You can easily download Power BI Desktop using the below link.

<https://go.microsoft.com/fwlink/?linkid=2101122>

To sign in to the Power BI service, you need to create an online account first. You can do it by using your email address.

To learn the advanced realms of Power BI, there is a common flow of activities to be followed. They are:

1. Submit the data in the Power BI Desktop. Then create the required report.

2. Publish the report to Service. After doing so, you will be able to create dashboards and visualizations as per your requirements.
3. Now collaborate the newly created dashboards with other team members to review, and they can also modify it simultaneously.
4. Check the shared dashboards as well as custom reports for interactions within the Mobile apps.

Now in the next chapter, we will see about the building blocks present in Power BI, which help create appealing visuals and reports from your data.

# **Chapter 1**

## **Power BI Building Blocks**

There are a few main things that Power BI performs, and they are known as building blocks. Once you have understood the building blocks of Power BI, you can easily use them to start building complex and intuitive reports. In fact, all the complex things in Power BI are built with the help of building blocks only. In real life also, building blocks are the pillars of any project such as cars, buildings, roads, etc. Now let us see the building blocks present in Power BI, which help easily create complex things.

1. Visualizations
2. Reports
3. Datasets
4. Dashboards
5. Tiles

Now let us understand each one of them one by one.

### **Visualizations**

It is a visual representation of your data intuitively by using charts, maps, and other interesting things. Power BI comes with various types of visualizations, and they are evolving with every new version released. Let us see the various visualizations present in Power BI service with the help of the below image.

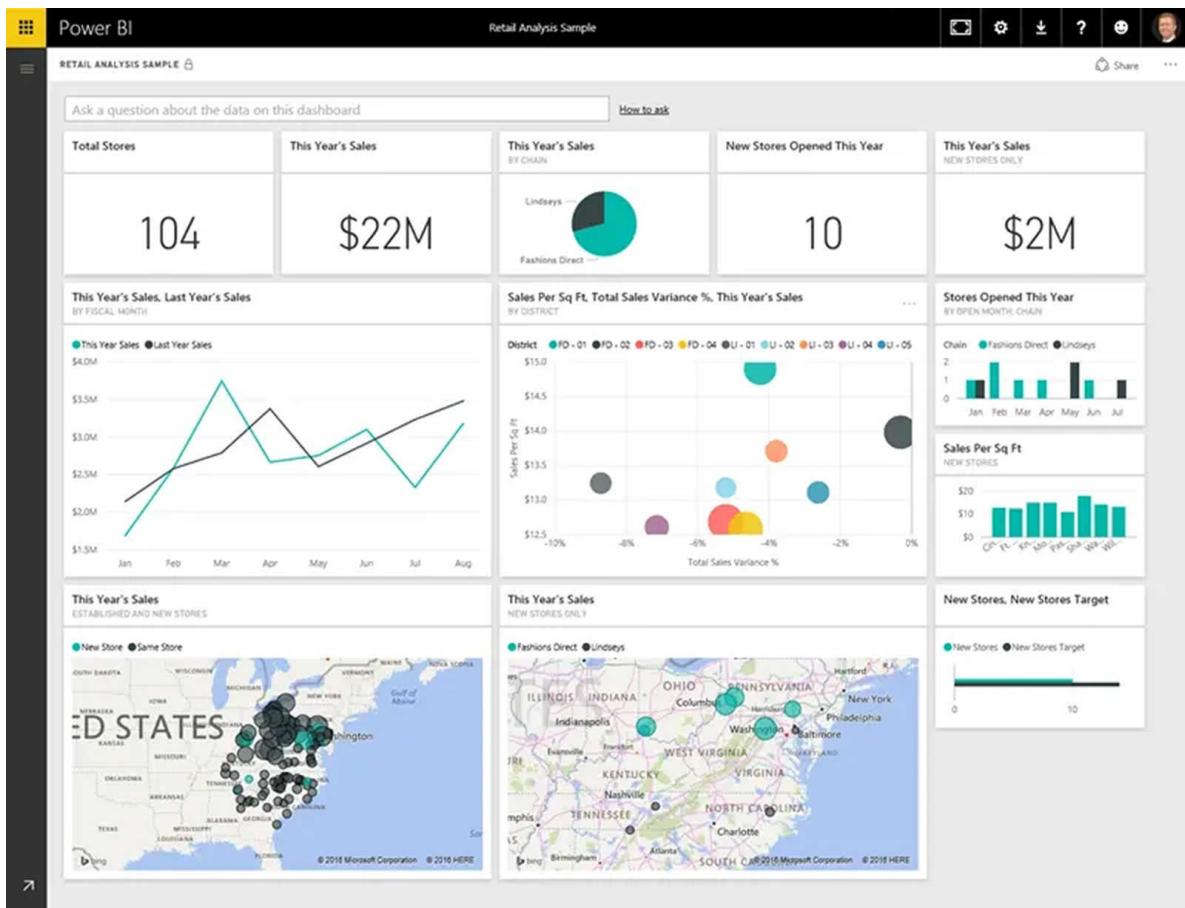


Image source: Microsoft.com

Visualizations are very simple in nature. They are like a unique number, which is representing an important thing. They can also be complex if the issues require a huge amount of data to be shown in the form of a color map. The main idea behind representing data using visuals is to show such data, which is very difficult to be shown in the form of text or tables.

## Datasets

It is a collection of data that is used by Power BI to start creating the visualizations. It can be a very simple dataset coming from an excel sheet workbook like the employee information (Name, Age, Address, Salary, Date of Joining, etc.)

Datasets are not limited to individual sources. They can be a combination of more than one data source which is filtered and then combined together. This results in a unique dataset having all the required data to be used in Power BI for

visualization.

Let us say that you have created a dataset with the help of following database fields:

1. A table from a website
2. An online survey result
3. An Excel table in your laptop or desktop

This will be a unique combination of 3 different databases. Still, it will be counted as a single dataset in Power BI, which is pulled from various data sources.

To take into account the data, which is important to you, filtering plays a very important role in Power BI. Let us say you have filtered the contact database in such a way that only those users who have received your online survey link are included in the Power BI dataset. Then you can easily create visuals with those filtered dataset collections of users whom you invited for taking the survey. Therefore, filtering will surely help you to focus on the data you need to be considered.

A very important feature in Power BI is a large number of data connectors present in it. Power BI provides the data to be connected from every major data source like Microsoft SQL, Excel workbook, Oracle, Azure, Salesforce, Facebook, MailChimp, and more. With the help of Power BI, you can easily connect to the data sources, filter them using the required options, and finally put in your Power BI dataset for further processing.

Once you have the required dataset handy, you are free to start creating visualizations to display the data in different formats and designs to understand the insights proficiently. This is where the role of Power BI Reports comes in.

## Reports

It is a combined collection of visuals created by you, and they appear together in one or multiple pages. It is like a normal report which you create for a meeting presentation with your peers. It is a collection of data that are inter-related to each other in one form or the other. You are also free to create reports with the help of the Power BI service.

Reports help you in creating various visualizations and then arrange them in your own order as per the needs & requirements. The report can be for anything from quarterly company results to the sales growth.

## **Dashboards**

Once you have created the report and it is ready to be shared, then you will create a dashboard for the same. A dashboard is a collection of the visuals taken from one page, which is then shared with others. It can be shared with those users who will be providing insights into the data.

It is a mandatory requirement that the dashboard should integrate inside a single page. That page is known as a canvas where you drop the visualizations in an empty space. You are free to share the dashboards with other users for interaction. It can be opened in the mobile device or Power BI service.

## **Tiles**

It is a single visualization of the dashboard or a report. It is a rectangle-shaped box that will hold a particular visual. A tile can be surrounded by multiple tiles as well. While creating a dashboard or report, you are free to move the tiles in any order as per your needs and requirements. A tile can be made larger, its height or width can be changed, and it can hover over other tiles.

If you are not the owner of the report or the dashboard, you will be able only to view it. It means it is only a read-only mode, and you can't modify it in any manner what so ever. You are only authorized to interact with it, that's all.

## **Combined Together**

So we saw all the building blocks of Power BI in the above sections. Therefore, Power BI is a collection of apps, services, and data connectors that help you to bring the data from various sources. Then you filter them as required and finally move it to Power BI reports or dashboards to create intuitive and appealing visualizations to be shared with other users.

Since Power BI is backed by Microsoft, you can always trust that even the complex datasets can be presented in a straightforward manner, and further expansion can be done with a few clicks. Now let us see how Power BI service works.

## **Using the Power BI Service**

Some users directly start within the Power BI service, so let us see how to create the visuals using the Power BI: apps.

First of all, an app is the collection of already created, readily available reports and visuals that are shared in the whole organization. It is like a pre-built template, and you only need to do a few clicks to see the collection on your screen in a package.

Now we will see how the apps and services work together.

## **Creating Dashboards using Cloud Services**

You can easily connect to any database using Power BI. All you need to do is click on the “Get Data” button present at the left-hand side of the screen (home page).

Then the canvas, which is the area present in the center of the screen, will display all the data sources available in Power BI service. You can choose anyone from the list, such as Excel worksheets, Microsoft Azure, SQL, GitHub, SaaS service providers, Facebook, Salesforce, Google Analytics, third party databases, and many others.

One thing to note here is that for all the software services, you will get a readily available collection of visuals that are pinned on reports and dashboards. All these visual collections are known as an app. Let's say you are using the GitHub app; the Power BI service will connect you to the account once you enter your credentials in the dialog box. Once you have done that, a new pop up will be shown on the screen having the predefined visuals as well as dashboards.

So they are useful apps for Power BI online services. You will see various online services as soon as you click on the “Get” button present inside the “Services” section. There will be lots of apps available to connect to the data source. You can choose any one as per your needs and requirements.

For example, you can connect to GitHub, which is a version control online software. You will be prompted to click on “Connect to GitHub” after clicking the button. One thing to note here is that Git Hub does not support Internet Explorer. So you need to choose any other web browser such as Google Chrome, Mozilla Firefox, Safari, Opera, etc.

Once you have provided the credentials and other information to connect to GitHub app, it will automatically start the installation process. Once the data gets loaded, you will see a predefined dashboard for GitHub on your screen. All the dataset, report, and the dashboard will be available altogether. You can choose any visual from the dashboard and start interacting with it easily. All the remaining visuals on that particular page will also start responding on their own. This means all the visuals get refreshed and updated automatically whenever you interact with a single visual.

## **Updating Data from Power BI Service**

It is very easy to update or modify the dataset for the app used in the Power BI service. To do so, you need to follow the below steps:

1. Click on the “Schedule update” icon present in the dataset.
2. Use the options present in the Menu which appears on the screen.
3. Use the update icon having the arrow with a circle (It is present near the schedule update icon).
4. This will automatically update the dataset to the latest values.
5. The Settings page will be shown on the screen. Choose the “Datasets” tab from that page.
6. The next step is to click on the “Scheduled refresh” arrow in order to expand it. Once it is expanded, you can see the “update settings” option. Use it to update the settings.

Now let us move forward to Power BI Desktop and see how it is different from Power BI service.

## **Power BI Desktop**

It is a free to use desktop application for laptops and personal computers which allows you to collect, transform, and finally visualize the data in Power BI. One thing worth noting here is that Power BI Desktop works along with Power BI Service. For creating the dashboards and reports, you need to work in Power BI Desktop. For publishing the created dashboards and reports, Power BI Service is used. So, in short, both of them work parallel together.

If you are wondering where you will get the sample data to create reports and dashboards, don't worry. You can easily get the sample data for the Excel workbook at this [link](#). Then you can import the data in the Power BI Desktop by selecting “Get Data” -> Excel.

The very first step is to understand the Power BI Desktop’s GUI (Graphical User Interface). Once you launch it on your computer or laptop, it will show “Getting Started” pop up on the screen. It will contain all the important links and resources for blogs, videos, forums, FAQs, and other support. You can visit the same for gaining more knowledge on the same, or you can close the pop up for now and get back to it at a later stage.

The very first step in Power BI Desktop is to build reports using the “Reports” view. Here mainly there are five areas which you need to take care of. They are Ribbon, Canvas, Pages tab, Visualization Pane, and Fields Pane. All of them are used for different functionalities while creating reports.

# **Chapter 2**

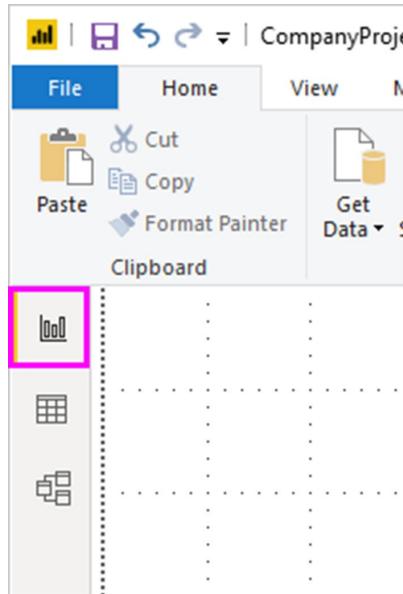
## **Power BI Reports**

Once you have started working on Power BI, it becomes very easy to create the best reports with the help of dynamic datasets and insights. There are many advanced functionalities and features present in Power BI Desktop. You can easily create advanced queries, create new relationships among different tables, mashup the huge amount of data from various data sources, etc.

There is a “Report View” section in the Power BI Desktop, which allows you to create more than one-page reports full of visualizations and charts. It provides a top-notch design experience to the users, which is pretty similar to the one that of editing view inside Power BI Service. It allows you to move the visualizations on the canvas, merge the visualizations, copy & paste, and more.

The main difference between Power BI Desktop and Power BI Service is that while working in Power BI Desktop, you are free to write your own queries and then model the data in such a way that it supports required insights for your Power BI reports. Once it is done, you have to save the file on your computer, laptop, or cloud.

When loading the data for the first time in Power BI Desktop, a blank canvas will appear where it will have the Report view. With the help of icons present at the left side of the canvas navigation pane, you can easily switch between Data, Report, and Relationship. It will be like the below image.



As soon as you are done with adding the data, fields can be added to the visualization inside the canvas. If you wish to change the visualization type, simply select it, and then choose a new type from Visualizations.

One thing to note here is that a minimum one empty page is required to create a report. All the pages are shown on the screen, which is on the left side of the canvas.

You are free to add as many visualizations as you want on a page, but you have to keep in mind that they are not over the limit as well. This is because having a huge amount of visualizations on a single page will make the page look dull and unnecessarily filled up with too much information. This is not a good practice, and you should try to avoid it as much as you can. In order to add new pages to your report, you can click on the “New Page” present in the ribbon.

If you wish to delete a page, you can do it by selecting the X (cross) button present at the bottom of the page in the Report view. One thing worth noting here is that when you are using Power BI Desktop, visualizations and reports are not allowed to be pinned on a dashboard. In order to do so, you have to go to the Power BI site and then publish it from there.

## Copy-Paste Reports

It is very simple to take a screenshot of the visual using a report from Power BI Desktop and then to paste it to any other report in the Power BI. You only need to use the command **Ctrl + C** for copying the visual report. Then move on to the

different Power BI Desktop report and use the command Ctrl + V to paste it there. You are free to select an individual visual or more than one visual present on a page while copying. Similarly, as per your decision, the paste will happen in the other report.

This copy/paste functionality is very handy if you like to create or modify multiple reports in a short span of time. This is because when you are copying the reports, all the formatting and the internal settings which are set by you in the formatting pane will automatically get applied to the destination report. Thus, if you rely on a particular formatting for your reports, you don't need to worry much, and Power BI will take care of the same.

You will get an error or a warning if the fields present in your model are different. The warning might also come if the fields do not exist anymore. This happens when you have already deleted a field in the model which another visual was using at the same time. In order to resolve this issue, you simply need to change the broken fields with the ones you have used in your actual model while pasting the visual. In case you have used a customized visual, you have to import the same customized visual at the destination report.

## **Hiding the Report Pages**

It is possible to hide the pages from a report after creating it in Power BI. This is required in scenarios where you have to use the data or visual in a report, but at the same time, you don't want others to see them. For instance, you are using tables that are being used in pages of other reports. This makes sense to hide the report pages from others who are not authorized to view those tables.

In order to hide a page report, you need to right-click on that particular report page tab. Now click on the “Hide” option from the menu on the screen.

If you want to hide a report page, you have to keep the below points in mind:

- The hidden report view will still be visible in Power BI Desktop even if the title page has been grayed out.
- The hidden report page will not be visible when the report is opened using the Power BI service.
- It is not a security-related process to hide a report page. Users can still access the page and content via various methods.

- In the View mode, if you hide the report page, then the arrows for view-mode navigations will not be visible.

## Different Filter Types in Reports

All filters are not the same and offer different functionalities. Their behavior depends on how you have created the filters in the Power BI Desktop's editing mode. There are basically two categories of filters in Power BI:

1. Automatic
2. Manual

Now let us see how they are different from each other.

**Automatic Filters:** These filters are added automatically to the visual level present in the filter pane as soon as the visual is built. They are basically dependent on the fields which are responsible for creating your visual. If you are a user having the Edit permission in Power BI reports, then you are allowed to edit, lock, clear, rename, sort, or hide the filter in a new pane. You will not be able to delete the automatic filter. This is because the visuals present in the report are referring to those fields.

**Manual Filters:** These filters can be dragged and dropped anywhere on filter pane by the owner of the report. If you are a user with the Edit permission in Power BI reports, then you are allowed to edit, lock, clear, delete, rename, sort, or hide the filter in a new pane.

Now let us see some more advanced filters present in Power BI reports. They are not commonly used, but they play a very important role in creating beautiful and intuitive reports. Using the right filter at the right time in your report will definitely help you.

### ***Include & Exclude Filters***

They are the filters which automatically gets added in the filter pane as soon as you use include or exclude feature for any visual in your report. If you are a user having the Edit permission in Power BI reports, then you are allowed to lock, delete, sort, or hide this filter present in the new pane. You won't be able to clear, rename, or edit the filter because the filter is related and linked to visual functionality.

### ***Drill Down Filters***

They are the filters that automatically get added in the filter pane as soon as you use the drill-down feature for any visual in your report. If you are a user having the Edit permission in Power BI reports, then you are allowed to clear or edit this filter present in the new pane. You won't be able to hide, rename, delete, sort, or lock the filter. This is because the filter is related and linked to drill down visual functionality. If you want to remove this filter from your report, you need to select the "drill up" button in your visual.

### ***Cross Drill Filters***

They are the filters that automatically get added in the new pane as soon as a drill-down filter gets passed to any other visual in the page report. It can be either via cross highlight or cross filter feature. If you are a user with the Edit permission in Power BI reports, then also you are not allowed to clear, lock, delete, hide, sort, or rename this filter. This is because the filter is related and linked to drill down visual functionality. Editing the filter is also not allowed as it is coming directly from the drill-down filter present in any other visual in the page report. If you want to remove the drill-down filter, you need to select the "drill up" button in your visual, which has been passed to this filter.

### ***Drillthrough Filters***

They are the filters that get passed from a page report to any other page report using the drillthrough feature. It is seen in the pane reserved for drillthrough functionality. Drillthrough filters are of two types. The first type of drillthrough filter invokes the functionality of the drillthrough. Power BI Report editor can edit, clear, lock, hide, or delete this filter.

The second drillthrough filter type is passed on to target depending upon the page level filters used in the source page. Power BI Report editor can delete, clear, or edit these filters. But this filter is not available for end-users to hide or lock it.

### ***URL Filters***

They are the filters that are added to Power BI's new pane with the help of a URL query parameter. If you are a user having the Edit permission in Power BI reports, then you are allowed to edit, clear, or delete this filter present in the new

pane. You won't be able to hide, rename, sort, or lock the filter. This is because the filter is related and linked to the URL parameter. If you want to remove this filter from your report, you need to remove the particular parameter from the URL.

### ***Pass through Filters***

They are the filters which are corresponding to the newly created visual level filters via the Q&A feature. If you are a user having the Edit permission in Power BI reports, then you are allowed to sort, hide, or delete this filter present in the new pane. But you won't be able to clear, rename, edit, or lock the filter.

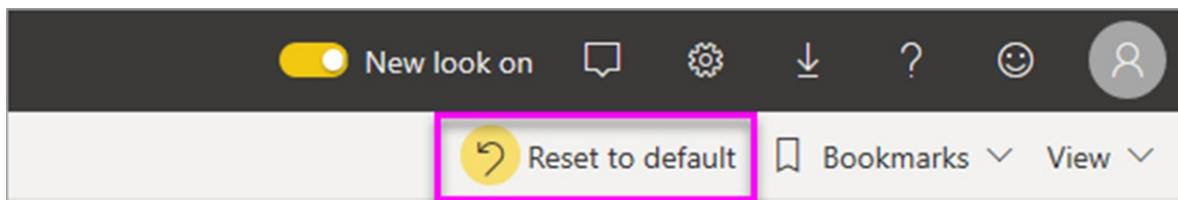
## **How to Add Filters to Reports**

Different ways of adding a page filter, report filter, visualization filter, or drill through the filter into a report in Power BI can be easily understood here. All the related examples are explained in the Power BI service. Similar steps can be followed while using it in Power BI Desktop.

### ***Editing View and Reading View Filters***

Reports can be viewed in two different modes: Reading view and Editing view. The filtering capabilities used by a person will depend on the view he is visiting.

As filters do not get deleted automatically if you navigate away to any other Power BI report, it will still retain all the data, slicers, and filters created by you. Thus, you are allowed to start from where you left earlier as soon as you get back to that Power BI report. In case you do not wish to keep the changes to be retained, you can do it by clicking on the “Reset to default” option present on the top of the screen, as shown in the below image.



### ***Filter Levels Present in the Filters Pane***

No matter you use the Power BI Service or the Power BI Desktop, you will always find the Filters pane at the right-hand side of the canvas. However, if it is

not visible due to any issue, you can click on the “>” button from the top right corner to expand the Filter pane.

Filters can be set at three different levels in the Power BI report:

1. Visual level
2. Page level
3. Report level

Note: You are also allowed to set drillthrough filters as well.

### ***Adding Visual Level Filter***

You are allowed to add a visual level filter in the report using two different methods.

1. You can filter a particular field which is already used in a report visualization.
2. You can find out a field that is not already used in a report visualization and, after that, map the particular field with the Visual level filter bucket directly.

### ***How to Filter Fields already Present in Visual***

You need to follow the below steps in order to do so.

1. Click on “More Options” and then choose “Edit Report.” It will open the report in the editing mode view.
2. If the Visualization, Fields pane and Filters pane is not already opened, you need to open them.
3. Choose any visual to keep it in an active state, including every field used by visuals in the Fields pane. It is also listed under the heading “Visual level filters.” Now you need to add the filter to a field that is already used in visualization.
4. The next step is to scroll down until the “Visual level filters.” Now click on the arrow, and it will expand the particular field which you want to filter.

5. You will get three filtering controls to choose from Top N, Basic, and Advanced. You can select any one of them as per your needs and requirements.
6. Visuals will perform the changes in order to reflect the latest filter in the report. When you save the report with the new filter, all the users will be able to see the changes, and they will also be able to interact inside the Reading view. They can either select or clear the values in the reading view.
7. If you are using the filter on a particular field which is aggregated (like a count, average, or sum), then the filter will be applied to the aggregated value at every data point present there.

### ***How to Filter Fields not Present in Visual***

You need to follow the below steps in order to do so.

1. Go to the Fields pane. Click on the particular field on which you would like to add the new visual level filter. Then drag that field and drop it in the “Visual level filters.”
2. If you are dragging and dropping a numeric value column to filter pane, then the filter will be applied to all underlying rows automatically.

### ***How to Add Drillthrough Filter in Report***

With the help of drillthrough in both Power BI Desktop and Power BI service, you are allowed to make a destination page report which will focus on a particular entity like a customer, manufacturer, or supplier. From remaining page reports, you can right click on the data point for the particular entity, then drillthrough automatically to your focused page report.

One thing to note here is that Power BI will be adding a backward arrow to the page report. If you select that arrow, you will be returned back the original page report where you chose to drillthrough. If you are in the Power BI Editing view, you can press the Ctrl Key to select that backward arrow.

### ***Adding Report Level Filter***

You need to follow the below steps in order to add a report level filter to your

entire report.

1. Click on the “Edit report.” It will open the Power BI Editing view.
2. If the Visualization, Fields pane and Filters pane is not already opened, you need to open them.
3. Go to the Fields pane. Click on the particular field on which you would like to add the new report level filter. Then drag that field and drop it in the “Report level filters.”
4. Choose the values which you would like to filter. Visual present on the active page, as well as on all pages of the report, will change in order to reflect the new filter in the report. When you save the report with the new filter, all the users will be able to see the changes, and they will also be able to interact inside the Reading view. They can either select or clear the values in the reading view.
5. You can click on the backward arrow in case you wish to go back to the previous page in the report.

## **Role of Filters and Highlighters**

By now, you have already learned that filters are used to focus on the data which you need and remove the unnecessary data. But Highlighting is different than filtering. It doesn't delete the data but actually highlights the subset of data. It means the data which is not highlighted will remain visible but in a dimmed manner.

You will find various methods for filtering and highlighting Power BI reports.

## **Filters Pane**

Filter pane is used to display the tables and the fields present in the Power BI report. It also shows the applied filters in the report. You are allowed to apply the filters directly in Filters pane, or you can also make the selections using slicers. Basically, there are four types of filters, namely: Page filters, Visual filters, Report filters, and Drillthrough filters. Each one of them covers different areas on the Power BI report.

Suppose a filter has a note “All” beside it, then all values present in that field

have been included.

The filtering capability depends upon the mode used to interact with the reports. It means filtering is different for Reading view mode and Editing view mode.

You are allowed to add all the filters in the editing view. It means as soon as you save the Power BI report, all the filters will automatically get saved and are synced to the mobile app also. Users who are reading the report in the Reading view will be able to interact with your filters, but they are not allowed to add any new filters to the report.

Whereas in the reading view, interacting with all filters already present in the report is allowed. You can also save your selections, but adding new filters to the report is not allowed. You are allowed to explore data present in the report by changing or altering the existing filters. All the changes get updated in real-time and get synced to the mobile app as well. Whenever you close the report, all the filters get saved. In order to undo your filters and go back to default settings for filters, you can click on the “Reset to default” option present at the top of the screen.

You can select an axis label or a value from the visual in order to highlight the remaining visuals present in the page report. If you want to remove the highlighted ones, the clock on the value once again and choose the unoccupied space. This is known as Ad hoc highlighting.

# **Chapter 3**

## **High-Density Line Sampling**

Since June 2017 onwards with the release of Power BI Desktop and important version updates for Power BI service, you now have a new line sampling algorithm in Power BI. It improves the visuals in Power BI, which are having high-density data. Let us say you have created a line chart using the sales figures from your retail stores, and every individual store has more than lakhs of sales invoice throughout the year. Therefore, a line chart for such data will be sampling the data into meaningful information to showcase each store's data as well as create multi-line charts to display underlying data. It is a common algorithm used to visualize high-density data in Power BI. It improves the sampling of data in an efficient manner.

### **How Does Line Sampling Actually Work ?**

Earlier Power BI used to select the collection of sample data points from the total range of underlying data with the help of the deterministic approach. Let us suppose that a high-density data for a year has around 300 sample data points shown on the visual. Each sample data has been selected to make sure that the total range of data gets represented. In order to understand this, for example, there you have plotted the stock market price for a one-year duration. It will have 365 data points creating the visual line chart, i.e., 1 data point for each day's stock price.

In this particular scenario, the stock price for the company will have many values during the share market timing for that day. The stock price changes every second, and thus there is a daily high level as well as the low level for the stock. It can be at any time during the trading hours. In high-density line sampling, suppose the data sample is taken at 9:30 AM and 2:00 PM every day. But it might not have the real high level and low level of that particular day in that time period. Therefore, sampling will not be able to capture the relevant points, which in this case was the daily high level and low level of the stock

price.

In terms of definition, high-density data needs to be sampled for creating quick visualizations that are responsive and interactive in nature. Having more than required data points in a visual can decrease the trend visibility and get diverted from its actual reason for creation. Therefore, how to perform data sampling was the need of the hour, and a sampling algorithm was created in order to provide real-time visualization to the users automatically. The algorithm is a combination of best response, data representation, and up to data sampling of data points in a slice.

### ***How New Algorithm for Line Sampling Works***

It is used for line chart visuals and area chart visuals on an X-axis model. Power BI's new algorithm is intelligent enough to slice down your data in the form of high-resolution chunks. Then it will automatically pick the important data points for representing every chunk in the visual. The process of slicing is tuned in such a manner that the final chart is almost alike and identical to all underlying data points in a fast and interactive mode. There are restrictions on the maximum and minimum values to qualify for high-density visuals.

1. The maximum number of important data points allowed to be displayed is 3500 irrespective of total underlying points. For example, you are having 20 series of 175 data points each. The visual has now already reached the maximum allowed data point limit, i.e.,  $175 * 20 = 3500$ . For only one series, you can have 3500 data points for the new high-density line sampling algorithm.
2. The maximum number of series allowed for a visual is 60. For example, you have greater than 60 series; then, you will have to cut off the data and make fewer series. You should use a slicer as a general practice for showing the data segments. By using a slicer, you can easily filter out the all-inclusive category for the same page report.

Note: The below visuals have higher limits than the 3500 data points.

- In the case of R visuals, the maximum data points limit is 1.5 Lakh.
- In the case of Power BI visuals, the maximum data points limit is 30,000.

- In the case of scatter charts, the maximum data points limit is 10,000.
- For all other visuals, the maximum data points limit is 3500.

These limits are enforced in Power BI Desktop to make sure that visuals get rendered speedily as well as are interactive in nature. They should not result in excessive computation load to render the visual in Power BI.

### ***Evaluation of Data Points in High-Density Line Charts***

As soon as the underlying data points are more than the maximum allowed limit to represent it in the visual form, the **binning** process starts automatically. It starts chunking the underlying data into various groups known as bins, followed by filtering them one by one.

The new algorithm in Power BI starts creating as many bins as it can in order to make the largest granularity in visual. For every bin, it will find the maximum and minimum data values so that the significant and major values get shown in visual. On the basis of results acquired by binning and further advanced data processing, value for a minimum resolution of the X-axis gets decided. This is to make sure that Power BI has maximum granularity in the process.

Every bin has 2 data points inside it, which are the representative points for visual. They are the maximum and minimum values, which means that important values get captured in the Power BI visual. This might look very advanced in terms of complexity, but that is why it has been created to help the users in solving the high-density data problem.

### **Toolips in Displayed Data**

One thing worth noting here is about the binning process in which maximum and minimum values are shown. It definitely affects the process of how tooltips will show data after you are hovering the mouse on those data points. In order to understand this, let's revisit the stock price example once again.

For example, you have created a visual to compare two different stocks, and both of them have high-density sampling data. You will be capturing the stock price every second, which in turn creates a huge amount of data points for each series. The Power BI algorithm will be performing the binning process for every series without affecting the other.

The first stock price has increased at 11:15 AM, and then it has moved down again after 30 seconds. It is a major data point in the visual. In the binning process, the stock will be at high at 11:15 AM to represent the data point. Coming to the second stock, its price was neither high nor low at 11:15 AM to be included in the bin. We can say that both the high and the low values for that stock came after 20 minutes.

So, in this kind of situation, after the line chart gets completed, you have hovered the mouse at 11:15 AM. You find a value for the first stock at that time because it was a high point at that time, but you won't find any value for the second stock in the tooltip. It is because there was nothing to show at that time in the tooltip. These situations will regularly arise when you use tooltips. The maximum and minimum values for one bin might not match with the X-axis data points, and this will affect the tooltip to show no results.

Note: The new algorithm is turned ON by default in Power BI. If you wish to alter this default setting, you need to go to “Formatting” pane. Then navigate to the “General” card. At the bottom of the screen, there will be an option named “High-Density Sampling.” Slide it towards the left to turn it off.

## **Limitations in the Algorithm**

There is no doubt that the new algorithm has significantly improved high data sampling in Power BI, but at the same time, there are some limitations that you need to know. Limitations are mentioned as below:

- Tooltips are affected due to the binning process and high granularity. They might only display value if and only if data points are aligned with the cursor.
- If the complete data source is too large, the algorithm will automatically eliminate the legend series elements in order to fit in the maximum data constraint. The series is arranged in alphabetical order until the maximum limit is reached, and it will discard the remaining series from Power BI processing.
- If there are more than 60 series, then the algorithm will first order them in alphabetical order and remove the ones after the 60<sup>th</sup> ordered numbered series.

- All the data needs to be in either numeric or date/time format. The algorithm will not consider any other type of value, and it will be processed by a normal algorithm, which is non-high density in nature.
- “Items having no data” is not supported in the algorithm.
- It is not possible to use this algorithm while having a live connection with SQL Server 2016 or later version. Though it will be supporting live connections in Power BI and Azure services.

# **Chapter 4**

## **Connecting with Data Sources in Power BI**

Power BI allows you to connect with various data sources such as worksheets, databased present in cloud services, local databases, Azure, Social Media databases, and more. On some occasions, the data you have collected from the data sources is not well structured and managed as you would have thought. In order to structure the data in the required format, you have to transform it by splitting columns, changing the data types, renaming some columns, creating meaningful relationships among columns, etc.

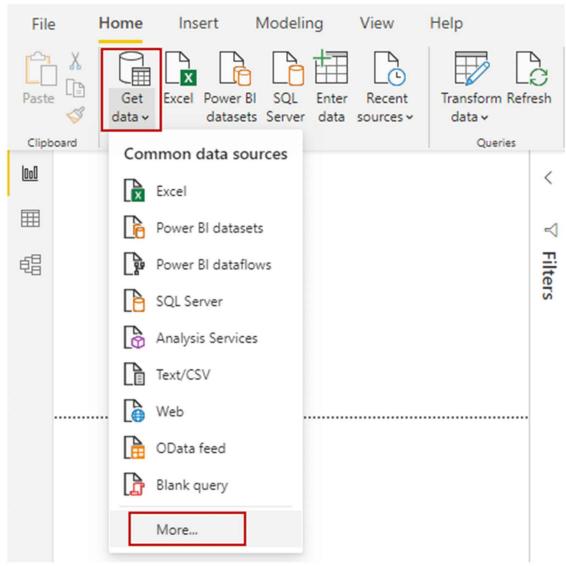
The examples shared in this chapter have been taken from the Microsoft Access database. You can download it from [here](#). Once you have downloaded it, you need to import the same into Power BI Desktop by navigating to “Get Data” -> Database -> Access Database.

Power BI supports around 60 connectors for cloud services like Marketo and GitHub. Connecting to generic sources is very easy via CSV, XML, ODBC, and text format. You can even extract all the tabular data easily from any website.

### **Steps to Connect Data in Power BI**

In order to connect to data in Power BI, you need to follow the below steps:

1. Open Power BI Desktop using shortcut link on your desktop or from the Startup Menu. Select the “Get Data” option present in the Home tab ribbon, as shown in the below image.



2. You will find various data sources to choose from. You can choose from any of these like Excel, Power BI datasets, SQL Server, Text/CSV, OData feed, Web, Analysis Services, and More. Select the required one in order to connect to the database. As per your selection, further options will be shown on the screen to find the source either on the web network or a local computer. If required, you may have to authenticate your request by signing in to the particular service.
3. Now you have to choose the data which you want to import. Once you have connected to the database, you will see a new window “Navigator” on the screen.
4. It will show all the entities or tables from your data source. When you select a particular entity or table, it will display a preview of contents present in it.
  - i. Now you have two options to proceed further: You need to import the entities or tables by clicking on the Load button.
  - ii. You can clean and transform the data before loading it in Power BI. To do so, you need to select the button “Transform Data.”
5. Once you have loaded the data in Power BI Desktop, you won't be allowed to make any changes later. So, if only a specific customer set needs to be filtered, then use the Edit button to filter out the data first and then load it.

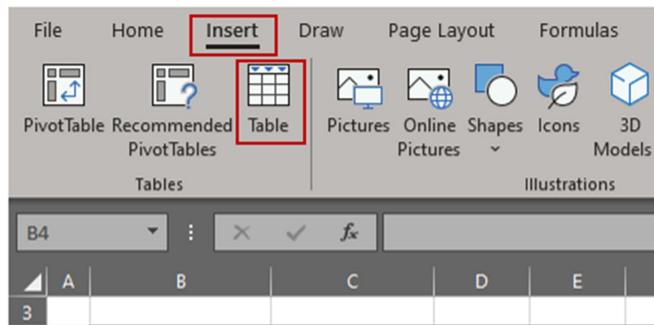
6. This will load all the data to Power BI Desktop, and you can now use it for further processing.

## Import Data from Microsoft Excel

It is very simple to import data from Microsoft Excel to Power BI. You will learn how to get data from an excel workbook having a normal table present on your computer. Till now, you have seen how to import data via Power BI Desktop. Let us now see how you can do it via Power BI service.

Here are the steps to do so:

1. The first step is to check if the data present in the excel workbook table is in the correct format or not. You need to highlight the cell ranges, navigate to the “Insert” tab on the top of the screen, and choose “Table,” as shown in the below image.



2. You need to check that every column in the table has a valid and relevant name. This is required because it makes it easy to find relevant data while creating Power BI reports.
3. You can keep your file anywhere on your computer—no need to keep in a pre-defined particular drive like C drive or D drive. Just open Power BI, navigate to “Get Data” -> Files -> Local File. A dialog box will open where you have to select the Excel workbook you want to import.
4. Once the excel file is imported in Power BI, the report creation process can start. Note: It is not necessary for the excel file to be present on your computer only. It can be on SharePoint or OneDrive as well.

Once the above steps are completed, you will find a new dataset present in Power BI under the heading Datasets. You are now free to explore the data by creating dashboards and reports.

1. Choose the “Open Menu” icon present besides the dataset.
2. Now click on Explore. It will open a blank canvas on the screen.
3. Towards the right-hand side on the screen, you will find your tables as well their columns under the “Fields” section.
4. Now you have to choose the field(s) which you would like to present in your canvas visualization. Changing the visualization type, applying filters, and modifying relevant properties can be done from the “Visualizations” section.

Note: Data can also be imported using advanced features such as Power Query, Power View, and Power Pivot.

## **Excel Workbook Types Supported by Power BI**

Connecting and importing from workbooks created in Microsoft Excel 2007 or later versions is supported by Power BI. All the workbook extensions must be in xlsx or xlsm format. The file size should be less than 1 GB.

Let us now see the different workbooks supported by Power BI.

### ***Workbooks having ranges or data tables***

Power BI supports worksheets having ranges of data, and they should be formatted as tables. You will be able to see the named tables as well as their columns in Fields pane while creating reports. It makes your work easy for visualizing data in the reports.

### ***Workbooks having Data Models***

Power BI supports workbooks having a data model containing a single table or multiple tables in it. These tables are loaded in the data model with the help of Power Pilot, Power Query, or linked tables. Power BI supports all the data model properties like hierarchies, relationships, KPIs, and measures.

One thing to note here is that you are not allowed to share the data models with other Power BI users. Let us say one user who is logged in to Power BI using abc.com account is not allowed to share the workbook with another user who is logged in to Power BI using xyz.com.

## ***Workbooks having External Database Connections***

Power BI supports workbooks using the Microsoft Excel for connecting to any external database in order to create dashboards and reports as the data is dependent on an external source. Not only this, but you are also allowed to create “Scheduled Refresh,” which automatically connects to the database for updates. This eliminates the need to refresh every single time using the Data ribbon manually. All the tiles present in the dashboards and the visualizations present in the reports will be automatically updated and synced with the data present in the data source.

## ***Workbooks having Charts, Pivot Tables, and Power View Sheets***

The location of your workbook file plays an important role in order to display, or not to display, the charts, pivot tables, and power view sheets. Let us see how the file location impacts in the below section.

### ***How Workbook File Location Makes a Difference***

The workbook file can be located anywhere, like in your local computer, OneDrive, SharePoint, etc. Let us see one by one each of them and how each one is different.

1. Local Computer: The workbook file can be loaded into Power BI from your computer. Since the original file is already present in your local hard drive, therefore not the complete file will be imported to Power BI. The actual process that happens is the creation of a new dataset. All the data and other data model(s) are loaded from the workbook straight in the dataset. If there are any Power View sheets present in your workbook, they will be shown in the “Reports” section in the Power BI site. Microsoft Excel 2016 provides the “Publish” button directly in the File menu. It is the same as the “Get Data” feature of the Power BI with the only difference that you will find it easy to update the dataset using the “Publish” feature if you are a user who makes regular changes in the workbook.
2. OneDrive (Business): If you use your OneDrive for Business and Power BI with the same account, then it is the best possible method to keep all the work in sync like dashboards, reports, and datasets. As both OneDrive and Power BI use the cloud services, Power BI will

automatically connect with the workbook file present in OneDrive in every 60 minutes. If there are any changes noticed in the workbook file, all the dashboards, reports, and datasets are updated automatically by Power BI. Similar to saving your workbook in a local drive, you are allowed to use the Publish feature for updating the reports and datasets immediately, else Power BI will do it for you in every 60 minutes.

3. OneDrive (Personal): If you have saved the workbook files in a personal OneDrive account, there will be many benefits similar to what you would have got with a Business account for OneDrive. The main difference is when you connect to the workbook file for the first time. You need to go to “Get Data” -> Files -> OneDrive (Personal) and then sign in using the Microsoft account (It will be different from the one you were using to connect to Power BI). One important point to note here is to use the “Keep me signed in” checkbox to make sure you are allowed to connect with the workbook in every 60 minutes. This will keep your reports and datasets in sync with the workbook file.
4. SharePoint Team: It is the same as saving your workbook file in OneDrive (Business). The only difference is that you need to connect to a workbook using Power BI only. Simply provide the URL for the same, or you can also connect with the root folder.

### ***Two Different Ways to Use an Excel Workbook***

If you have used OneDrive to store your workbook files, then you can use the data present in Power BI in 2 different ways. Let's see both of them one by one.

#### **1. Importing data in Power BI**

If you are using the **Import** feature, all the supporting data present inside the tables or data model(s) will get imported to a newly created dataset. If there exist any Power View sheets as well, then they will get re-created in the form of reports in Power BI.

You are allowed to make changes into your workbook, and once they are saved, it will automatically get synced with the dataset every 60 minutes. If you want to get them synced instantly, you always have the option of using the “Publish” option to get it done immediately. All the visualizations present in dashboards and reports get updated as well.

In Microsoft Excel 2016, there is an option to directly Publish and Export.

## 2. Connecting, managing, and viewing Excel

If you are using the **Connect** feature, you will see your workbook similarly you see in Excel Online. You will also get additional features to pin down elements directly into the dashboard from the worksheets. You won't be able to modify the workbook in this situation. If you want to make any changes, select the Edit button and make the changes in Excel Online or on your computer. Then all those changes will be saved in the workbook present on OneDrive.

You should go with this option only if you have got Charts, Pivot Tables, or Ranges to be pinned down in the dashboards.

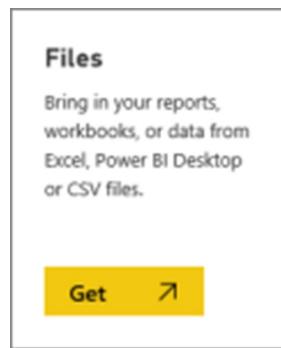
## Connecting to Excel workbook using Power BI

In order to connect or import to an Excel workbook, you need to follow the below steps.

1. Open Power BI and navigate to NAV pane. Select the “Get Data” option, as shown in the below image.



2. Go to Files. Then select the Get button as shown in the below image.



3. Choose your file from the dialog box.
4. If the workbook is present on SharePoint or OneDrive, then click on the “Import” button.

If your Excel workbook is present in your local system, you need to follow the

below steps.

1. Choose the “Local File” option from the menu. Navigate to the location where you have saved the workbook file.
2. Select the file and use the option “Upload file into Power BI.”
3. It will start uploading the workbook, and once it is done, a notification will be shown on the screen, saying that the workbook is now ready.
4. You will see your workbook in the Reports section.
5. One thing to note here is that when you use the “Import” option, it will only import the data which is present in the named table or inside the data model. It will show an error that “No data found in workbook” if the workbook does not have any named tables, data models, and Power View sheets.

## **Fetching Data from Power BI Desktop**

After you have imported data into Power BI and created some reports, you should now focus on getting the saved file into Power BI service.

Let us see how you can publish it from Power BI Desktop using the below steps.

1. Go to Power BI Desktop and select File.
2. Now navigate to Publish -> Publish to Power BI, as shown in the below image.



3. Now provide your credentials for Power BI to sign in. This is required only for the first time and is a one-time activity.
4. Once the authentication is completed, a notification will be shown on the screen. It will have a URL link which will take you to the report in Power BI site.

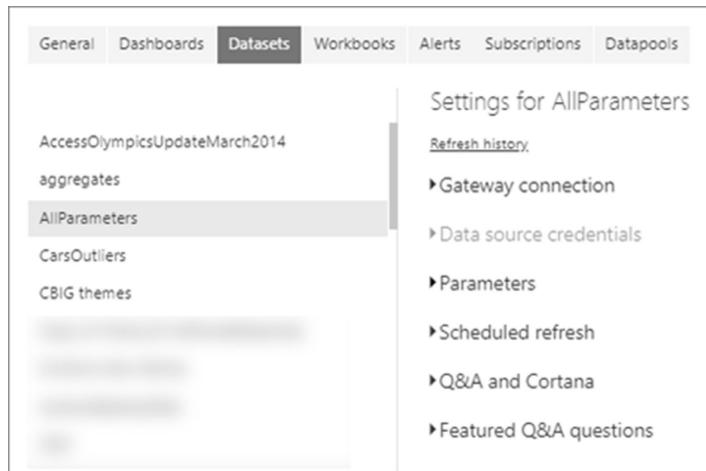
## **Editing Parameter Settings**

Query parameters are added to the reports by the report owners. The main use of parameters is to allow the report creators to create specific parts of reports depending upon a set of parameter values. Let us say a report creator has created a parameter that will be restricting the report data to an individual region or country, which takes only certain predefined field formats such as text, time, and date.

Once the parameters are defined in Power BI Desktop, they will come along with the settings when you are publishing the report in the Power BI service. You are allowed to edit a few settings which describe or define the accepted values. But you won't be able to edit the parameters restricting data to be used.

Here are the steps to do so:

1. Go to Power BI service and click the icon  . It will open the Power BI “Settings.”
2. Go to the third tab “Datasets” from the top. Now highlight any dataset present in the drop-down list, as shown in the below image.



3. Expand the “Parameters” section present on the screen. If it doesn’t have any parameters associated with the dataset, it will display a message regarding the same. If there are parameters in the dataset, it will show all the parameters when you expand it.
4. Now you can review all the settings for every parameter. If you want to perform any changes, you can do it here itself. One thing to note here is that you won’t be able to edit the fields which are greyed out.

So this was how you could review and change the parameter settings for the datasets in Power BI Desktop.

# **Chapter 5**

## **Power BI Real-Time Streaming**

Real-time streaming allows you to stream the data and get up to date dashboards instantly. Any dashboard or visual which is made in Power BI is eligible to be displayed as well as updated in real-time. Streaming data sources are social media posts, factory sensors, service metrics, and more. In fact, all the devices having time-sensitive data in it are used for transmission.

Here you will learn how to set up the streaming in real-time. Let us first see the various types of real-time datasets designed to be used for showing in dashboards and tiles.

### **Types of Real-Time Datasets**

Basically, there are three datasets which are: Push dataset, Streaming dataset, and PubNub dataset.

Let us discuss all the three datasets one by one.

#### ***Push Dataset***

This dataset is used to push the data in Power BI. When you create a dataset in Power BI, a new database gets created in the service which is used to store data. Because you have this underlying database ready to store the Power BI data, you can create reports using the same. The visuals and the reports will be similar to the normal Power BI reports and can be used to create alerts, pin down to dashboards, etc.

As soon as the report gets created with the help of a push dataset, all its visuals are eligible to be pinned down on a dashboard in Power BI. All those visuals will be updated and synced in real-time if there is any change in the data. The dashboard triggers a refresh each time new data comes in the service.

Couple of points to note here are:

1. If you pin the complete report by using the “Pin Live Page” feature, then the data will not get updated automatically.
2. When a visual gets pinned down on a dashboard, you will be able to use the Q&A feature for asking questions to the push dataset. After you have created a query for Q&A, you are allowed to pin the visual back on the dashboard so that it gets real-time updates.

### ***Streaming Dataset***

By using the streaming dataset, you will be pushing the data in Power BI service. But there is one difference with respect to push dataset. Power BI stores your data in a temporary cache, and it gets expired very quickly. The cache is mainly used to show the visuals having a transient history like a line chart, which expires after 60 minutes.

There isn't any underlying database in the streaming dataset, and thus you will not be able to create report visuals with data present in the stream. Also, you will not be able to use certain features like Power BI visuals, data filtering, etc.

There is only one option for visualizing the streaming dataset. You need to add a tile first and then make use of a streaming dataset in the form of a “custom streaming data source.” In order to display real-time data quickly, customized streaming tiles are optimized. There exists very little inactivity time in between the data getting pushed in Power BI service and visual getting updated. This is because data is not required to be read or entered into the database.

In practical scenarios where it is important to reduce the inactivity time between data getting pushed and when the data gets visualized, streaming datasets are used. As a best practice in Power BI, you need to push the data in such a format which gets visualized as it is and doesn't require any modifications. The examples for such types of data are pre-calculated averages and temperatures.

### ***PubNub Streaming Dataset***

PubNub SDK is used by the Power BI web client in order to read directly from already existing PubNub stream. This ensures that the data is not stored in the Power BI service at all. The PubNub streaming dataset doesn't have any underlying dataset associated with it, and thus you will not be able to create report visuals using the data flowing in the stream. Also, you will not be able to

use certain features like Power BI visuals, data filtering, etc.

In order to visualize the PubNub dataset, you need to add a tile on your dashboard in Power BI and then configure a data stream (PubNub). In order to display the real-time data quickly, tiles customized on the PubNum dataset are optimized. There exists very little inactivity time in between the data getting pushed in Power BI service and visual getting updated because Power BI is connected with PubNum data stream directly.

## How to Push Data in Datasets

As we have understood the real-time datasets in the above section, let us now move forward on how to push the data in datasets. You can use three different methods for pushing data in the dataset. They are mentioned as below:

1. By using Power BI REST API
2. By using Azure Streaming Analytics
3. By using Streaming Dataset UI

Now let's understand them one by one.

### ***By Using Power BI REST API***

REST APIs are used in Power BI for creating and sending the data in **push** datasets followed by sending it to **streaming** datasets. If you have created a dataset with the help of this method, a **defaultMode** flag is used to indicate if the dataset is streaming or pushing. If there is no flag set in the system, it means it is a push dataset by default.

If the value of **defaultMode** is set as **pushStreaming**, it means that the dataset is of streaming and push type. This gives a dual benefit of both of them. One thing to note here is that if the **defaultMode** is set to **pushStreaming** value, then the restrictions for both datasets apply to it. For example: In the streaming dataset, the request has to be more than 15Kb in size, whereas, in the push dataset, the request has to be less than 16MB in size. When both of them are validated and are in the required range, the request will be successful. Finally, the data will get updated in the push dataset. But at the same time, if there are any streaming tiles in the dataset, they will fail temporarily.

When the dataset is completely created, you can use the REST APIs for pushing the data. It is done with the help of PostRows API in Power BI. All requests going to the APIs are highly secured with Azure OAuth.

### ***By Using Streaming Dataset UI***

You can easily set up a dataset with the help of the API method in Power BI service.

While setting up a new streaming dataset, you need to enable the slider for “Historical data analytics,” which plays an important role, as shown in the below image.

The historical data analysis slider is by default in the disabled mode, which makes your dataset a type of streaming dataset. Once you enable the same in the slider, the dataset will function as both push dataset and streaming dataset. One thing to note here is that Azure AD authenticated is not a mandatory requirement for streaming datasets that have been created using this method. Here, the dataset creator will be receiving a link having a rowkey. It will act as an authorization method for pushing the data in the dataset.

### ***By Using Azure Streaming Analytics***

Power BI can be added like an output in the Azure Stream Analytics for visualizing the data streams in real-time. Let us see how this process actually happens.

ASA (Azure Stream Analytics) takes the help of REST APIs for creating the output data stream for Power BI. Its defaultMode value is set as pushStreaming to make the dataset use features of both streaming datasets and push dataset. Apart from this, ASA will also set another flag called **retentionPolicy** to the value basicFIFO. This setting makes sure that the push dataset has the ability to store the first 2 Lakh rows in it. As soon as the upper limit is exceeded, rows will be dropped in the order of First In First Out. It means the first row will get dropped from the dataset if there is a new record after 2 Lakh rows. The same process is repeated for any future rows coming in the dataset.

Note: If the query results from ASA are reaching to Power Bi very fast like one or two results every second, ASA will start the batching process to combine those outputs in one single request. Due to this, the streaming tile limit might

exceed, and the tiles will no longer render in Power BI. So to avoid these situations, you need to slow down the output rate reaching Power BI. You can do it by setting the maximum value to 10 seconds instead of doing it every second.

## **Setting Up Real-Time Streaming Dataset**

You just learned the three types of datasets and how you can use them for pushing data in the dataset. Now let us see how you can set up the real-time streaming datasets. In order to start the real-time streaming in Power BI, there are two options:

1. Tiles having visuals from the real-time streaming data
2. Datasets developed using the streaming data

No matter which option you choose, you are required to configure the **Streaming data** correctly in the Power BI system. In order to configure this setup, open a new or existing dashboard. Then click on “Add a tile,” followed by choosing the option “Custom data streaming.” You can also use the option “Manage data” if the real-time streaming data is not configured yet.

Now a new page will open, and you need to provide the endpoint for the streaming dataset in the text box shown on the screen. If there is no streaming dataset created by you, click on the (+) icon present at the top right corner of the screen. It will provide you all the options for creating a streaming dataset.

After clicking on the (+) icon, you will get two options to choose from. You need to choose the one which suits you best. The options are as follows:

1. PubNub
2. Power BI REST API

Let us see understand both the options one by one.

### ***Use PubNub to Setup Real-Time Streaming Dataset***

As you already know that PubNub has been integrated within the Power BI Desktop, you will be using the low-dormancy data streams. If it is not present, you can create it as well. After choosing the option PubNub, click on Next. A new window will appear.

The PubNub data streams have a huge volume. Thus, you need to see if it will be worthy in its original form to store data and use it for analytics. In order to use the Power BI Desktop for historical data analysis using PubNub data, you need to aggregate the raw stream first. Then send the same to Power BI for processing. You can do all this using the ASA tool.

### ***Use Power BI REST API to Set Up a Real-Time Streaming Dataset***

The latest updates and patches in Power BI REST API makes it a very good choice for real-time streaming datasets. First, you have to choose the API using the “New streaming dataset” window. A new window will open that has all the entries required for connecting and using the endpoint for the data stream.

You can enable the “Historical Data Analysis” option for storing the data transmitted via the data stream into Power BI. Once this data is collected, you can easily use it for reporting.

Once the data stream is created successfully, you will get a URL endpoint for REST API. This URL can be used by your application for calling the POST requests in order to push the data into streaming datasets built by you. The request body needs to match with the JSON while calling the POST requests in the Power BI service. Let us say that you can wrap up the JSON objects in the form of an array object.

### ***Example of Real-Time Streaming***

Let us see the functioning of real-time streaming with the help of an example. A public stream available for PubNub has been used here. You need to follow the below steps:

1. Go to Power BI service. Now create a new dashboard or choose an existing dashboard.
2. Click the option “Add Tile” -> Go to “Custom Data Streaming.” Click the Next button to continue.
3. Use the “Manage Data” link to create new streaming datasets if you do not have it already. The option is available just above the Next button used in the previous step. Use the plus (+) streaming data link present at the top-right corner of the screen. Now choose PubNub and click the Next button.

4. Provide an acceptable name for the dataset. Click on the Next button.
5. Select the default options present on the screen and click on Create.
6. You will be taken to the Power BI screen, where you need to set up a new dashboard. Once done, add a tile to the dashboard. When you are creating a tile and choose the option “Custom Data Streaming,” a streaming dataset will be available for you. You can now add numbers on the line charts, and do much more.
7. Now create customized datasets and stream real-time data in Power BI service.

# **Chapter 6**

## **Publish Using Microsoft Excel into Power BI**

By now, you must have understood that you can directly publish the workbooks using Microsoft Excel 2016 and later versions into Power BI. It can then be used for creating intuitive and creative dashboards as well as reports. All of them can be shared with other co-workers for collaboration.

When you publish an Excel workbook into the Power BI, you need to take care of a few things like the below ones:

1. You must be using the same account for Office, OneDrive (Business), and Power BI to publish the workbook.
2. You are not allowed to publish a workbook that is not having any data in it and is completely empty. Also, it is not allowed to publish a workbook having unsupportive content with respect to Power BI.
3. You are not allowed to publish the workbook, which is either password protected or encrypted with Protection Management.
4. You must enable the modern authentication before publishing the workbook. If it is not enabled, the Publish button will be either greyed out or not present in the menu.

### **How to Publish the Workbook**

The first step is to open Microsoft Excel. Go to Select “File” -> Publish. Now you can perform two options: Export or Upload.

If you use the **Upload** option for the workbook, you will be able to communicate with your workbook in the same way as communicating in Microsoft Excel Online. Pinning the selection to dashboards is allowed from the workbook. The complete workbook or its selected sheets can be shared with others using the Power BI interface.

If you use the Export option for the workbook, all the table data, as well as the data model, will get exported to the Power BI dataset. Then you will be able to create dashboards and reports in Power BI.

## **Publishing a Local File**

You can easily publish the local files using Microsoft Excel. It is not required that it is present in SharePoint Online or OneDrive (Business). One thing to note here is that only Microsoft Excel 2016 and later versions are compatible to perform this operation if they have the Office 365 license. Standalone installations of Excel 2016 will also be able to Publish, but in that case, the workbook has to be present in SharePoint Online or OneDrive (Business).

After selecting the option “Publish,” you will be able to choose the workspace where you would like to publish it. It can be either the personal workspace you are using or the group workspace in which you have access.

When it gets published to Power BI, all the content present in the workbook will get imported, and the local file will be kept separate as it is. In order to make any changes in the file present in Power BI, you need to publish the latest version once again. Otherwise, you need to perform a data refresh by scheduling an automatic refresh for the workbook in the Power BI interface.

If you are publishing it using the standalone installation, save the workbook in OneDrive (Business). You need to click the “Save to Cloud” option, then provide the location where you want to save in OneDrive. Once you use the Publish button, two options will be there, as mentioned earlier also.

## **Upload Option**

In this option, the workbook will be similar to how it looks in Microsoft Excel Online. But only limited features will be present for pinning the elements on the dashboards from the worksheet.

Editing the workbook is not allowed in Power BI. For making any changes in the data, click the Edit button. Now open in in Excel on your computer or online. Now those changes will be saved in OneDrive (Business). Also, datasets will not be created automatically. The uploaded workbooks will be given a unique icon to identify them like the ones that are uploaded and not imported.

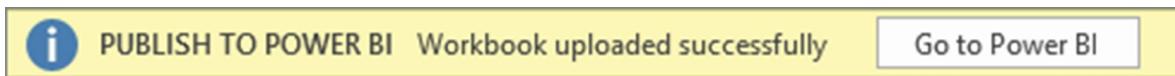
You should wisely use the Upload option if there are Charts, Pivot Tables, or data in worksheets that are required in Power BI.

## Export Option

In this option, all the table data and the data model will be exported in the newly created dataset. If there are any Power View sheets, those will be created again in the form of reports.

You are allowed to edit the workbook directly, and the changes will automatically get in sync with the dataset present in the Power BI interface in every 60 minutes. If it is an urgent update, you can use the Publish option present in Excel to get the changes updated instantly. All the visualizations will be updated as well.

So you can use any option from the above ones. Once it is done, Microsoft Excel will automatically sign in to Power BI using the current account. Then it will publish it and you will be able to check the progress. Once the process is complete, you will get a successful message, as shown in the below image.



## Reducing Workbook Size to Display in Power BI

All files having a size less than 1 GB can be uploaded easily into Power BI. You must already be aware that a workbook has two parts in it. They are:

1. The Data Model
2. The remaining report is covering the core content structure.

The following limits are placed in the system for saving the workbook in OneDrive (Business).

1. The complete workbook can't exceed 1 GB size.
2. The core content worksheet can have a maximum size of 30 MB.

Thus, if your content worksheet has more than 30 MB size, you will not be able to upload the workbook in Power BI. The elements which make your core content size greater than 30 MB are Images, Colored worksheets, Clip arts, Shaded cells, and Text boxes.

Therefore, you need to remove these elements from your workbook (if possible).

If there are Data Models, then you can remove the data from the worksheet and store the data in the Model only. You can also consider creating a data model that is memory efficient in order to decrease the size.

## **Removing the Data from the Worksheet**

If the data has been imported to Excel using the Excel Data Tab or Power Query tab, then the workbook will have similar data as present in the tables. Big tables eat up the content size and make it greater than 30 MB. Therefore, if you will remove the tables and keep the data in the model itself, the size will be decreased considerably.

Use the below points while importing data into Excel.

1. Do not use the option “Load to worksheet” from Power Query. This will ensure that the data is not imported in the worksheet at all and goes directly to the model.
2. Use the Excel Data tab to navigate to Existing Connections. Go to “Only create connection.” Now remove all the native tables that are created while the initial import was done.
3. Do not use the option “Table” present in the Import Data screen.
4. You can use the workbook size optimizer in order to decrease the overall workbook size. This will work if you have a data model present in it.

So this was how you could publish using Microsoft Excel into Power BI easily.

# **Chapter 7**

## **Working with R Scripts**

R is one of the most powerful computer languages used for data manipulation and visualization, so we need to understand how we can use it in the Power BI Desktop to import data from where we want it and use it to create some cool visualizations.

Power BI Desktop is another powerful tool that helps you create reports. These reports can easily be saved to the Power BI Report Server or published to the Power BI Service. The fact that Power BI Desktop is integrated with the R language is one of the most valuable features; we can import data and transform it using R; we can use it to create data visualizations, and there are also several pre-built visualizations based on the R language that you can import, without having to write any R or interact with it.

In this chapter, we're going to look at how we can work with the R language in the Power BI Desktop. Our focus will be on performing tasks specific to R rather than delving into the actual language. It is for this reason that I will only be using simple script examples; these are purely to demonstrate how R can be used in the Power BI Desktop. When you grasp these basic concepts, you can move on and use the extensive capabilities offered by R for the analysis, transformation, and visualization of your data. If you have no knowledge of R, now would be a good time to go and learn the basics.

To work with R in Power BI, you must ensure the language is installed with Power BI on the same computer. R can be downloaded and installed for free from several places, depending on the distribution you want. For example, you can install the CRAN distribution from the R Project website, <https://cran.r-project.org/bin/windows/base/>, or, if you want the Microsoft R Open distribution, head to

<https://mran.revolutionanalytics.com/download>. I will be using CRAN v 3.5.0.

It is also best if you install a separate R IDE (Integrated Development Environment). Doing this means your R scripts can be verified before you run them in Power BI Desktop. Plus, if you are familiar with other computing languages, you will know that using an IDE means you can easily install the packages you need to use R within the Desktop environment. I am using RStudio (the free version), which you can get from <https://www.rstudio.com/products/rstudio/download/>.

When you have installed R and set it up, the Power BI Desktop will look for the installation. To make sure it has picked up the right version, click on File>Options and Settings>Options. It should detect the right installation version and your IDE if you installed one. If the information isn't right, provide the details manually.

You only need to associate an IDE with the Desktop when you want R visualizations in the Reports view. When an IDE is associated, it can be launched from the Desktop, and you can work directly on your visualization script.

## Importing Data Using R Scripts

When you import data using an R script, that script has got to return one or more data frames; these will be the basis for the table you are importing. If you get several data frames returned, you can pick the ones you want included in the import, and Power BI will then create one table for each data frame. I do need to warn you that, if one of your data frames has a column that was configured using the vector or complex type, the values in the column will be replaced with errors by Power BI Desktop.

If you want to import data using an R script, go to the Desktop Home ribbon and click on Get Data. A dialog box opens; go to the Other category and click on R Script. Click the Connect button, and Power BI Desktop will open the dialog box for R Script. Here, you can type in your script or paste it in. We're going to get some data from the iris dataset, which is in the CRAN distribution; the following script is used, assigning the data to the variable called iris\_raw:

```
iris_raw <- iris
```

In the Power BI Desktop, the dataset has to be assigned to a variable, whether you make any modifications to the data frame or not. If you only typed in the

dataset name, you wouldn't find any data frames to import into the Desktop.

Before you put your script into the dialog box, run it through the IDE; this will allow you to check it is running okay, and you get the results you want. If you don't do this and you get an error from it in Power BI, it won't be easy to understand, and you will need to start again with the import.

When you are happy that the script is ready, type or paste it into the Script dialog box. Click OK, and Desktop will process it, opening the Navigator dialog box. From here, you can choose the data frames you want to import, and you can see some sample data for each one. In our case, we will only get the iris\_raw data frame.

Where you have multiple data frames, they are listed in the Navigator box under R[1]>Display Options. Make sure you check the box next to each data frame you want imported and then click on Load. Once the data has been loaded, click on the Data view to see the dataset.

Sometimes, your data may be in a file and not in one of the datasets built-in to the distribution. Let's say, for example, that you took the data from the iris dataset and copied it to a CSV file. You saved it to C:\DataFiles on your computer; that data can be pulled into an R data frame with the statement below:

```
iris_csv <- read.csv(file="C:/DataFiles/iris.csv", header=TRUE, sep=",")
```

This is using a function called read.csv. We set the argument for the header as TRUE, indicating that headers should be created from the first-row values. Then we used the sep argument to indicate that the data values in the file should be comma-separated. After that, the data is imported from the iris\_csv data frame using the same process as above.

R is an incredibly flexible language because it has the capability of installing packages that offer extra functionality. There are a few of these that help you with reports and analysis. For example, data.table and dplyr give you some powerful functions to use with data frames, and ggplot2 is great for visualizing data. These must be installed, and you can do this through the IDE. We used R Studio – open it and run the following commands:

```
install.packages("dplyr")
install.packages("data.table")
```

```
install.packages("ggplot2")
```

Once they have been installed, the library function in the R script can be used to call the package you want when you import the data. By doing this, you can use all the functions in the imported package.

A significant benefit of using R for data importation is that the data can be manipulated during the import process. The script below, for example, uses two functions from dplyr – group\_by and summarize – to group the data and then aggregate it is imported:

```
library(dplyr)  
  
iris_mean <- summarize(group_by(iris, Species),  
  slength = mean(Sepal.Length), swidth = mean(Sepal.Width),  
  plength = mean(Petal.Length), pwidth = mean(Petal.Width))
```

You use the group\_by function to ensure data is prepared for a different function, in our case, the summarize function. In the example above, summarize is used together with a function called mean, so that the mean can be found for the four measures. The values listed in the species column are used to group these.

Going back to the R statement we use, the data that was aggregated was saved into the variable called iris\_mean, and that is the name given to the dataset when it is imported to the Desktop.

This is just a simple script, and you can write scripts as complex as you want them to be. All I have done here is given you an idea of the simplicity of using R for importing data to the Desktop. The more experience you have with R, the more powerful this is.

## Transforming Data Using R Scripts

Sometimes, you might want to manipulate a dataset that has already been imported into the Desktop, and you can do this using Query Editor. With this, an R script can be applied to the dataset for transforming the data. Using our iris dataset, before the data is modified, look at how the dataset looks in Query Editor. In the section called Applied Steps, there are two steps:

1. **Source** – in the Source step, the main pane in Query Editor shows a small table. This table is representing the operation we did to import the dataset, and there would be one row for each data frame returned. In our case, because we only have one data frame, the table only has one row. In the column titled Value, the Table value is representing the data that goes with the data frame and, when you choose that value, step two kicks in.
2. **Navigation** – This is the actual data that we imported.

When an R script is used for importing data, both of these steps are added by the Power BI Desktop.

Now let's look at running an R script against our iris\_raw dataset. We'll keep it simple and use the exact same aggregation logic we used to import iris\_mean. There is one main difference – rather than iris being specified to reference the dataset, the dataset variable is used instead, as you can see in this script:

```
library(dplyr)  
  
iris_mean <- summarize(group_by(dataset, Species),  
  slength = mean(Sepal.Length), swidth = mean(Sepal.Width),  
  plength = mean(Petal.Length), pwidth = mean(Petal.Width))
```

To run scripts in Query Editor, go to the Transform ribbon and click on Run R Script. The relevant dialog box opens, and you can input your script to the textbox. Power BI Desktop will add a comment that tells you the input data is held in the dataset variable. That input data is the dataset active in Query Editor; in our case, that is iris\_raw.

Click on OK after you add your script and, in the Applied Steps, Query Editor will add in two steps that work much like Source and Navigation did:

1. **Run R Script** – this step is reflecting the data frames that the script returned
2. **“iris\_mean”** – this is the data frame that was selected

The ability to run R scripts against datasets gives you another powerful tool when you want to work with data you imported, be it from a text file, an online service, or a database system. Once the data has been imported into the Power BI

Desktop, it is there to do what you want with, regardless of where it came from.

## Creating R Visualizations Using R Scripts

Power BI Desktop also allows you to create visualizations using R scripts, and this is done through the Report view. Overall, this is as easy to do as using R scripts in other ways, but there is one important thing to remember – data is automatically grouped and summarized by R whether you want it or not, and there is no way of getting around the behavior. It's something of a mystery as to why Microsoft did this, and it can provide results that you really didn't want.

As with most things, though, we can get around this. How? Simply by adding a column in the dataset to identify each row uniquely. If you are familiar with the SQL Server table (more about that later), this is much like the IDENTITY column found in those.

If you are importing your data using an R script, the column can be added at the same time. Take the following script, for example; this gives our iris dataset an identifier column based on the row names or index of the dataset:

```
library(data.table)
iris_id <- iris
iris_id <- setDT(iris_id, keep.rownames=TRUE)[]
setnames(iris_id, 1, "id")
iris_id$id <- as.integer(iris_id$id)
```

We start our script by calling a package named data.table. This package gives us some useful tools to work with objects in data frames. Next, we use a function called setDT with an argument of keep.rownames, to use the index values to create our new column. Note that the iris dataset has to be assigned to the iris\_id variable before you can use setDT. The reason for this is that the setDT function directly changes the dataset, and this cannot be done with any built-in dataset.

Once the column has been created, the function called setnames is used to change the first column's name from rn, which is the default, to id. Lastly, the data type for the column is changed to integer.

Once you have the iris\_id dataset, you can use an R script to create your

visualization. In Report View, go to Visualizations and then click on R. When you do this for the first time, you are asked to enable script visualizations; just click on Enable and away you go.

On the Visualizations pane, after you click R, a graphic placeholder is added to the report by Power BI Desktop, and the R script editor pane opens. Before you can start writing your script, you must identify which columns from the dataset are being used in your visualization. The easy way to do this is simply to go to Fields (in Visualization) and drag your preferred columns into the Values section. For our example, I want you to do this with the columns title id, species, Petal\_Width, and Petal\_Length.

When the columns are added, you will see that Desktop adds some comments in the script editor box. Comments one and two are indicating that we created a data frame called a dataset, and this was based on our chosen columns. That data frame name, dataset, has to be used in your R script as a way of referencing the source data.

The next pair of comments, together with a warning message you will see at the top of the pane, are indicating that some duplicate rows have been eliminated from the dataset – this is why that identifier column was required.

Underneath those comments goes your script. The script below will produce a basic scatter plot:

```
library(ggplot2)  
ggplot(data=dataset, aes(x=Petal.Width, y=Petal.Length)) +  
  geom_point(aes(color=Species), size=3) +  
  ggtitle("Petal Widths and Lengths") +  
  labs(x="Petal Width", y="Petal Length") +  
  theme_bw() +  
  theme(title=element_text(size=15, color="blue3"))
```

The `ggplot` function from the `ggplot2` package is used for creating the visualization, using the labels and the colors specified in the script. Note that, for the X-axis, we used the `Petal_Width` column and, for the Y-axis, we used the `Petal_Length` column. The plot colors are served by the column named `Species`.

Once your R script has been defined, go to the top of the editor pane and click on Run Script. The script is then processed by Power BI Desktop, and the visualization is placed where the placeholder was inserted earlier on.

Visualizations created using R can be updated whenever you want; all you need to do is modify your script and then click on Run Script to update it.

If you wanted to use your IDE to edit your code, go to the top of the editor pane and click on Edit Script in External R IDE. The IDE will open, showing the R script with the right code for connecting to your chosen visualization data source. The script also has the code added to your visualization script in Desktop. You will still need to copy most of the code and paste it in the Power BI Desktop if any modifications are made, but this does get you out of needing to set the data source to test the script in the IDE.

When you add visualizations based on R to a report, you need to keep in mind that there are licensing restrictions with Power BI and, unless you hold a Power BI Pro license, these R visualizations cannot be used.

## **Importing Custom Visuals Using R**

Power BI Desktop also lets you import predefined visualizations that are based on R to the workspace. You can find these in the Microsoft AppSource gallery, and this is directly accessible via the Power BI Desktop. There is no need to have an understanding of R, and you don't need to build scripts or run them.

To get one of these visualizations, go to Report view and click on Visualizations. From there, click the ellipsis (...) and click on Import from Marketplace. The Power BI Visuals window opens, and you can browse what's there.

When you have found the one you want, click on Add. If the R Packages Required dialog box loads, you will know that you need other packages. The box will list the packages you need, and you have a choice – click on Install and Power BI Desktop will do it all for you, or click on Cancel and manually install them.

To continue our example, choose the Spline Chart. If you can't find it, go to the search box and type in Spline. When you have added it, the required packages box will open; just click on Install and let Power BI do the work for you.

When custom visualizations are used, you will see a new button in the Power BI

Visualizations pane; it will relate to the specific visualization. That visualization can then be included in your report and configured the same as you would the prebuilt ones. For ours, you can specify three columns from our iris\_id dataset – Species, Sepal\_Width, and Sepal\_Length.

The Microsoft AppSource gallery has several interesting, free visualizations and is certainly worth a look as it can help you to create even more engaging reports than the prebuilt visualizations allow. And Microsoft has gone the extra mile and made those visualizations dead simple to get into Power BI Desktop.

## **Getting the Best Out of R in Power BI**

The fact that R has been integrated makes Power BI even more powerful when it comes to the transformation and presentation of business intelligence data. R is one of the most comprehensive of the statistical graphics and computing languages, implemented extensively, and with one of the largest user communities. Used in Power BI, it can help you import data, modify it, and come up with many visualizations that produce deep insights into your data.

# **Chapter 8**

## **Working with Parameters**

Power BI offers you a way of making your reports even more dynamic by using parameters. These can be used to substitute query filters, connection information, and calculations in your queries.

In Power BI Desktop, parameters can be defined for use in many ways when you work with datasets. Creating them is simple, and you can add them in at the import stage or later if you want a dataset redefined or dynamic elements added. As an example, you could create parameters to provide a data source with connection information or parameters that supply predefined values for when you want to filter data.

We are going to focus on creating parameters that we can use with data that we import from an instance of a local SQL Server, using the data from a database named AdventureWorks2017.

### **Connection-Specific Parameters**

Power BI already contains some data source connections that allow you to use parameters when you are defining the properties for the connections; examples include SQL Server, Salesforce Objects, and SharePoint. Let's say that you want to get some data from SQL Server; you can use two parameters – one for the Server instance and one for the target database.

The parameters are not dependent on any dataset, and that means they can be created before or after you add a dataset. However, they must be defined, and their initial values must be set in Query Editor. When you create a parameter, you will see it in the Queries pane; here, you can also view the values, update, and reconfigure settings if needed.

We are going to retrieve some data from an SQL Server database, and, for that, we need a pair of connection parameters. The first one will list all the Server

instances that may be a host for the source data – for our example, we only need a single instance; the rest will just demonstrate having multiple instances.

First, open Query Editor and go to the Home ribbon. Click on the down arrow beside Manage Parameters and click on New Parameter. The dialog box opens, go to the Name text box and type SlqSrvInstance. Go to the Description box and type in a description for it.

Click on Type and choose Text from the drop-down menu. Then click on Suggested Values and click on List of Values. A grid will open; this is where the individual values are types for assigning to the variable. Make sure that a minimum of one value matches the name of a real SQL Server instance. The values to type in for this example are:

.\\SqlSrv17a

.\\SqlSrv17b

.\\SqlSrv17c

When the values have been typed in, choose a default from the drop-down menu, and then go to the Current Value menu and choose the current value for the variable. We'll use SqlSrv17a as both values.

Close the Parameters box by clicking on OK, and the parameter will be placed int eh Queries pane; the parentheses show the current value. When you click to select a parameter, you will see the button for Manage Parameter and the Current Value menu in the primary pane. To change the current value, choose a new one from the menu or change the settings through the Manage Parameter.

Now we need a target database parameter; follow the same process, but make sure the parameter is named as Database. Make sure a minimum of one real database is shown in the list. After you click OK, the parameter is added to Queries again. For our database, we will use the values:

AdventureWorks2014

AdventureWorks2015

AdventureWorks2016

That's it; you have created two connection parameters. Your parameters can be configured with other data types if you require them, such as dates or decimals,

or different formats for the values. For example, the parameter can be changed, so it takes any value or one from a list query. List queries can be created manually with an M statement or based on a dataset.

## Connecting to Data Sources

Now your parameters are defined, we can retrieve our data by connecting to the SQL Server instance. Apply all your changes and shut the Query Editor window if you haven't already done it.

We are going to run a T-SQL query but, first, you need to make sure that you have disabled the property for Require User Approval for New Native Database Queries. If it hasn't been disabled when you run the query, you will get an error. To check the property, click File>Options and Settings>Options. When the dialog box opens, click on Security and, if the property checkbox is ticked, clear it.

Back to the Desktop window, click on Data view and, in the Home ribbon, click Get Data. In the dialog box, go to the Database and click on the SQL Server database.

Click on Connect, and a new dialog box appears for SQL Server Database. The top part is the Server section; go to the first option, on the left, and click it; click on Parameter. Option two changes to show a drop-down menu where you will see your newly created parameters. Choose SqlSrvInstance and then do the same for the Database section, this time choosing Database.

Click the arrow for Advanced Options and type the T-SQL statement below into the box for SQL Statement:

```
SELECT h.SalesPersonID AS RepID,  
       CONCAT(p.LastName, ', ', p.FirstName) AS FullName,  
       CAST(SUM(h.SubTotal) AS INT) AS SalesAmounts  
  FROM Sales.SalesOrderHeader h INNER JOIN Person.Person p  
    ON h.SalesPersonID = p.BusinessEntityID  
 WHERE h.SalesPersonID IS NOT NULL  
   AND YEAR(h.OrderDate) = 2013
```

```
GROUP BY h.SalesPersonID, p.FirstName, p.LastName  
ORDER BY FullName ASC;
```

Click on OK, and you will see a preview window in Power BI Desktop, showing the two parameters at the top. Click on Load, and the dataset is added to the Data view. Before you go any further, give the database another name, RepSales, or something like it.

When you define the connection property parameters, their values can be changed whenever you need to, say, for example, if you needed to get data from another database or server instance. The parameters you create are not limited to single-use either; you can use them in other datasets, eliminating the need to repeat the information whenever you need a database for the same data source.

Later, we'll look at how to work with parameters after you add them to the dataset. Now, launch Query Editor and go to Applied Steps. Click on Source and look at the M statement that goes with it. Look carefully, and you should see that the Database and SqlSrvInstance parameters are referenced by the Serve connection data. You will find them in the SQL.Database function, as the first argument. You can see the statement here:

```
= SQL.Database(SqlSrvInstance, Database, [Query="SELECT h.SalesPersonID  
AS RepID,#(lf) CONCAT(p.LastName, ', ', p.FirstName) AS FullName, #(lf)  
CAST(SUM(h.SubTotal) AS INT) AS SalesAmounts #(lf)FROM  
Sales.SalesOrderHeader h INNER JOIN Person.Person p#(lf) ON  
h.SalesPersonID = p.BusinessEntityID#(lf)WHERE h.SalesPersonID IS NOT  
NULL#(lf) AND YEAR(h.OrderDate) = 2013#(lf)GROUP BY  
h.SalesPersonID, p.FirstName, p.LastName#(lf)ORDER BY FullName ASC;"])
```

The implication here is that it is incredibly easy to reference parameters in an M statement, and this gives you a powerful, flexible way of customizing your dataset's applied steps.

## Add Parameters to Your Filter Data

The Filtered Rows step is not always easy to update and can be inflexible, so, rather than that, you may opt to use parameters to filter datasets. An example of where you could do this would be in your T-SQL statement; in the WHERE clause in that statement, 2013 is hard-coded, and you could use a parameter in

place of this:

```
YEAR(h.OrderDate) = 2013
```

Parameters can be used in place of hard-coded values, so long as the parameter has support for a range of values. To do this, you create a parameter in much the same way as we did earlier. This time though, the parameter should be called SalesYear, a list defined to contain the years 2011 to 2014 inclusive, and the current and default values set to 2011.

Once the parameter has been created, the M statement that goes with the Source step for your database should be updated to use the following code in place of the 2013 value:

```
" & SalesYear & "
```

When the code has been updated, ensure the checkmark beside the statement is ticked; this will make sure your changes are properly formatted and that the code didn't accidentally get broken.

Next, we need to test our variable. From the Queries pane, click on SalesYear and pick a year that is different from the default of 2011. Then select the dataset (RepSales) again and check the data is updated.

## Add Parameters to Control Statement Logic

Sometimes, you may want to control the logic of a query using parameters instead of filtering the data. Take the T-SQL statement from earlier, for example. The SELECT clause comes up with each reps' total sales by using the SUM aggregate function:

```
CAST(SUM(h.SubTotal) AS INT) AS SalesAmounts
```

You could add a parameter to the M statement so that another aggregate function can be applied. First, we must create a new parameter, name it AggType and then a list must be defined with an item for each individual function:

```
SUM(h.SubTotal)
```

```
AVG(h.SubTotal)
```

```
MAX(h.SubTotal)
```

`MIN(h.SubTotal)`

Set the default value and the current value as the SUM function.

The reason we have the SubTotal column is partly so the logic is clear and partly to show you that the data can be summarized using other columns that your dataset supports. For example, your dataset may have a column titled DiscountAmounts that shows total sales minus discounts applied. In a case like this, the column and function can be defined by each parameter option, including values like `AVG(h.SubTotal)` and `SUM(h.DiscountAmounts)`.

Once the parameter has been created, the M statement for the Source step should be updated by using the code below in place of the hardcoded fragment of `SUM(h.SubTotal)` – don't forget to include the quote marks:

`" & AggType & "`

We can do something much the same with the ORDER BY clause used originally, and, to do this, we would need to use a variable that offers options on ordering the data. First, we need a variable called ResultsOrder. Next, for every option:

`FullName ASC`

`FullName DESC`

`SalesAmount ASC`

`SalesAmount DESC`

We need to specify the column we want the sorting based on – SalesAmounts or FullName – and we need to specify ascending or descending order. For this example, we use FullName ASC for the current value and the default.

Again, the M statement must be updated after the parameter is created, using the code below, with the quote marks, in place of the code fragment `FullName ASC`:

`" & ResultsOrder & "`

Learning how parameters are incorporated into M statements gives you many options for data manipulation and giving your datasets more flexibility without needed to add loads of steps to the query. Just make sure that every change is saved and applied, so you don't lose it, and you know it works.

## Naming Dataset Objects Using Parameters

Something else we can do with parameters is to use them to give objects dynamic names. You could, for example, rename the SalesAmount column with a name that reflects what the sales year is and the applied aggregation type. The easiest way of doing this is to add a new step to the dataset in Query Editor, call it Renamed Columns, and then update the M statement with the parameter values.

To do that, go to the SalesAmount column header and right-click it. Click on Rename and give the column a temporary name of Sales. Press the Enter key, and the step is added to Applied Steps, together with the M statement below:

```
= Table.RenameColumns(Source,{{"SalesAmounts", "Sales"}})
```

Incorporating the two parameters, AggType and SalesYear, requires that the hardcoded Sales in the M statement be replaced with:

```
Sales (" & SalesYear & "-" & Text.Range(AggType, 0, 3) & ")
```

Note that we used the & operator. This concatenates or joins Sales with the values for the two variables, separated using a dash and contained in a set of parentheses. The function called Text. Range will only get the first three characters from the variable called AggType.

Once the statement has been updated, verify that your changes work. Note that the name of the third column in the dataset now has a set of parentheses enclosing the year and the function; this makes it far easier to see the values given to the dataset.

## Using Parameters in Data View

Earlier, we looked at changing parameter values in Query Editor. To do this, you need to go to the Queries pane and choose the parameter you want to change and then update it as needed. However, this a time-consuming process, more so when you have several parameters that you want to change, one after the other.

Thankfully, there is an easier way to do this – in Data view. To do this, go to the Home ribbon and click the arrow beside Edit Queries. Click on Edit Parameters, and a dialog box named Enter Parameters opens. Choose which values you want applied to the dataset and click on OK.

It is fair to say that you might find the dialog box name of Enter Parameters and the option Edit Parameters to be somewhat misleading, but they do represent effective features for updating the values of your parameters and the data itself. However, you should be aware that any changes made here will be applied across any dataset that uses that parameter. If you want to use a similar parameter in several datasets, but you don't want them all to have the same values, you will need to create dataset-specific parameters, giving them relevant names that you can easily tell apart.

When you know how these values are set, you can play about and try out different values, making sure you look at the dataset each time to see what effect the new settings create. For example, the changes we just applied to our dataset would bring up a list of three columns – RepID, FullName, and Sales (2013 – AVG)

The Sales column now has a new name, with the year and aggregation type enclosed in parentheses. Also, note that we sorted the data based on the values in the Sales column. If you are in Report view, you can change parameter values so you can immediately see those changes in the visualizations.

It is clear that these parameter capabilities boost the power and flexibility that Power BI Desktop already offers, and there are several ways to use these parameters, irrespective of the data source. Get to grips with using these parameters, and you will find them incredibly effective, and you will have more control over the datasets.

# **Chapter 9**

## **Working with SQL Server Data**

The data you use for Power BI can come from several different sources, but, more often than not, SQL Server will be used for hosting it. We're going to look at how you can work with SQL Server data and the relationships that exist between tables.

Power BI Desktop is far more robust than the Power BI service for business intelligence reports because it offers support for many more data sources, and it has more tools for you to use in transforming your data.

SQL Server is one of those data sources that can only be used in Power BI Desktop, not in Power BI service. Although the service does give us the connectors for SQL Data Warehouse and Azure SQL Database, it doesn't offer the ones we need for SQL Server. The Power BI Desktop lets us access SQL Server data from whole tables or allows us to run queries that will return just a subset of the data from several different tables. Stored procedures can be called, including the one that allows you to run both R and Python scripts – `sp_execute_external_script`.

We're going to delve into how we retrieve the data from an instance of the SQL Server and look at some different ways the data can be returned. As with the last chapter, we are using the AdventureWorks2017 database, and we are running it on a local SQL Server instance.

Most of what we discuss here can easily be used on any other RDBMS – Relational Database Management System – and on multiple data source types. All data in Power BI is organized into queries, which are structures similar to tables and used when we work with datasets. Because of that, once the data is in the Desktop, most of the actions you do are likely to be similar, regardless of where the data came from. You will come across features that are specific to a relational database, though, especially when you are looking at the relationships between the tables.

## **Retrieving an SQL Server Table**

Power BI Desktop lets you retrieve entire SQL Server tables or views. It will preserve the structure and, where it can, it will identify what, if any, relationship exists between them. The example we're going to work on now demonstrates how to import tables and, if you intend to follow this practically, the first thing to do is retrieve the tables we want from our AdventureWorks2017 database.

To do that, open Power BI and go to the Home ribbon. Click on Get Data, and the relevant dialog box will open. Go to the Database section and double-click on SQL Server Database. A new dialog box opens, type in the SQL Server instance name of SqlSrvInstance, and the target database name of SqlSrvDatabase.

These are the two connections created for the purpose of this example. SqlSrvInstance has the name of the Server instance, while SqlSrvDatabase has the target database name. When you provide information about the connections, you have a choice – use the name of the connection or create parameters of your own.

Once you have input the required details, click on OK, and the Navigator dialog box will open. Choose the following three tables:

- Person.Address
- Person.BusinessEntityAddress
- Person.Person

In the right pane, you can see the selected table's data. If you wanted to include the tables related to the chosen ones, you could click on Select Related Tables. If, for example, you choose Person and then click on Select Related Tables, you would get nine tables, way more than we need for our purpose. The project you are working on will determine if you use this option or not but, if you do opt for it and only want some of the related tables, use the checkboxes to include or exclude tables.

When you want to import your tables, click on Load, and every table or view that you chose gets loaded to the Power BI Desktop, and each is listed in the Data view as an individual dataset. The datasets can be modified in Query Editor, the same as you do with any other data.

Probably, one of the first things you will do in Query Editor is to rename the datasets and get rid of any extra columns:

- **Rename a Dataset** – in Queries, right-click on the dataset and choose Rename; type the new name and press the Enter key
- **Remove a Column** – in the main grid, pick the column and go to the Home ribbon; click on Remove Columns
- **Remove Multiple Columns in One Go** – choose the first column, press/hold CTRL and choose each column you want to remove; on the Home ribbon, click on Remove Columns

Following these instructions, do the following for your three imported tables:

- **Person.Address table** – change the name to Address and then remove six columns – AddressLine1, AddressLine2, PostalCode, SpatialLocation, rowguid, ModifiedDate
- **Person.BusinessEntityAddress table** – change the name to BusEntAddress and then remove three columns – AddressTypeID, rowguid, ModifiedDate
- **Person.Person table** – change the name to Person and then remove ten columns – PersonType, NameStyle, Title, MiddleName, Suffix, EmailPromotion, AdditionalContactInfo, Demographics, rowguid, ModifiedDate.

For the time being, we can ignore any of the columns that have relationship data in them; we'll come back to them later.

Once you have updated your queries, make sure the changes are applied and come out of Query Editor. You will see that every dataset now has just two or three fields in it. Filters can also be added in Query Editor if you want the data refined further, or filters can be added to the visualizations when you do your reports.

## Work with Relationships

Did you notice, when the datasets were updated in Query Editor, that the queries had several pseudo-columns in them; these were indicated by over-long names and values in gold, usually Value or Table.

These columns are used to represent a relationship between tables in the dataset. If a column has Table values, these represent foreign keys in a different table that is referencing the primary table on which we based the query. If they have Value values, these are representing foreign keys in our primary table, referencing a different table.

As an example, if you are in Query Editor and you choose the Address query, four of those relationship columns will appear:

- Person.BusinessEntityAddress
- Person.StateProvince
- Sales.SalesOrderHeader
- Sales.SalesOrderHeader(AddressID)

These columns are indicating that the Address table, as it is in the original database, has a foreign key in it, referenced by three other foreign keys:

- In the Address query, the column called Person.BusinessEntityAddress is representing a foreign key that is in the table called Person.BusinessEntityAddress; the key is referencing the table called Person.Address. The foreign key gets created in the Person.BusinessEntityAddress table, in the AddressID column.
- In the Address query, the column called Person.StateProvince is representing a foreign key that is in the table called Person.Address; the key is referencing the table called Person.StateProvince. The foreign key gets created in the Person.Address table in the StateProvinceID column.
- In the Address query, the column called Sales.SalesOrderHeader(AddressID) is representing a foreign key that is in the table called Sales.SalesOrderHeader; the key is referencing the table called Person.Address. The foreign key gets created in the Sales.SalesOrderHeader table in the BillToAddress column.
- In the Address query, the column called Sales.SalesOrderHeader(AddressID) 2 is representing a foreign key that is in the table called Sales.SalesOrderHeader; the key is referencing the table called Person.Address. The foreign key gets created on the Sales.SalesOrderHeader table in the ShipToAddress column.

Click on any one of the Table or Value values, and a new step is added by Query Editor to Applied Steps (in the right pane); the dataset is also modified, so it includes information that relates specifically to the chosen value. For example, take the first row from the Address query; you can see it has a value of 1 for AddressID and a value of 79 for StateProvinceID; if you were to click on the value for Value in that row's Person.StateProvince column, a new step, would be added to applied steps. The step is named 1, and the dataset is modified, so it only has information out of the StateProvince table that is specific to the value of 79 for StateProvinceID.

The data can be seen for any Value instance in Person.StateProvince simply by clicking on it. The step associated with it will be given the name of the selected row's AddressID value and, when you are done looking at the results, you can delete the newly added steps and put the query back to how it was.

If you go to the Person.BusinessEntityAddress column and click on the Table value in the first row, a new set would be added, named as 1, but this time containing information from the table called BusinessEntityAddress. The dataset would contain just one row after it was modified, and there would be a foreign key reference in the row to the value of 1 for AddressID in the Address table.

Play about with the Table and the Value values to see all the different data related to them; just make sure that the newly added steps are deleted from Applied steps, so the query is back to its original form. You won't be doing this in the examples here, but it is good to learn for other SQL Server-related projects.

As well as finding the relationships with foreign keys, Power BI Desktop will also try to identify any relationships that exist between the datasets we imported. These relationships can be seen by looking at Relationships in the main Desktop window. You can see each dataset represented graphically, and you can move these around as you wish. If a relationship is found between any datasets, a connector is used to join them, indicating the relationship type.

In our case, Desktop has determined that there is a one-to-many relationship between two datasets – Person and BusEntAddress. It has also detected a one-to-one relationship between two datasets – BusEntAddress and Address. Double-clicking on the connector will show you the relationship details. For example, to see the details of the one-to-many relationship between BusEntAddress and

Person, double-click on the connector between the two – the Edit Relationship box opens, and you can see the details relating to the relationship. If needed, you can also modify the details, although you will find that Power BI Desktop already gets the relationships right, most of the time, based on the data currently held in the dataset. One thing you might spot is that this is not shown as a one-to-many relationship; it is a many-to-one relationship instead. This comes down to the order the datasets are listed in; in our case, BusEntAddress is first.

With Power BI Desktop doing all this work for you, it is much easier to come up with visualizations that are based on several datasets. Later in the book, we'll be looking at creating those visualizations, so, for now, just understand that the process is made much easier because of the relationships – the data has all been mapped for you.

## How to Merge Datasets

There may be occasions when you want several datasets combined into one. You might, for example, decide that the three datasets we imported earlier should be combined into one. To do this, go to Query Editor and click on Person. Click Merge Queries on the Home ribbon and then select Merge Queries as New.

When the Merge dialog box opens, go to the first drop-down menu and make sure that Person is selected. Next, go to the middle drop-down menu and choose BusEntAddress and, in the drop-down menu for Join Kind, choose Inner (Only Matching Rows). Query Editor now knows that an inner join needs to be created between the queries. Lastly, for each of the referenced datasets, choose the column titled BusinessEntityID and click on OK.

A new query now gets created, and you can rename this as CityData. The query has the first three columns from Person, together with some relationship columns. Delete every column except for the one called BusEntAddress, as this one represents the current query.

Next, go to the BusEntAddress query and choose at least one column for the merged query. To do this, go to the icon shown in the top right of the column called BusEntAddress and clear every column, leaving just the AddressID column. There is no need for you to add BusinessEntityID as it is already in the Person dataset; the rest of the columns are just the relationship columns.

You should also clear the checkbox next to Use Original Column Name as

Prefix, so the column name is kept simple; click on OK, and the AddressID column will be added to the CityData query.

Now we want to merge our Address and CityData queries. Keeping CityData selected in the Query Editor, go to the Home ribbon and click on Merge Queries. The Merge dialog box opens, go to the second drop-down menu and choose Address. In the Join Kind menu, choose Inner (Only Matching Rows). And, for both datasets, the AddressID column should be selected; then click OK.

Now go the top right of the reference column called Address and click on the icon; clean the columns, leaving just two – City, and StateProvinceID. Also, clear the checkbox beside Use Original Column Name as Prefix, click on OK, and your datasets will merge.

You now have just one dataset you can use for other operations. As an example, you could apply transformations, like row filtering, column splitting, pivot the data, and so on, while retaining the two original datasets. And with just one dataset to work on, visualizations are easier to create because you don't have to fight your way through several sources. However, as with any Power BI feature, whether you use it will depend on whether your project will benefit from it.

## **Retrieving SQL Server Data Using T-SQL Queries**

Power BI Desktop gives you several ways to transform your data but, without a doubt, the ability to return data from SQL Server databases using T-SQL queries is one of the most powerful methods. It saves you from having to specify each individual view or table and, by using a query, you can get the data you want without the need to filter rows, remove columns, or do anything else once the data is imported. This means that you work with just one dataset that has all the data you could want instead of having to go through several datasets to extract a bit of information here and there.

For this example, we will use the SELECT statement below to join six of the AdventureWorks2017 tables together, returning the data to Desktop:

```
SELECT cr.Name AS Country, st.Name AS Territory,  
p.LastName + ' ' + p.FirstName AS FullName  
FROM Person.Person p
```

```
INNER JOIN Person.BusinessEntityAddress bea
    ON p.BusinessEntityID = bea.BusinessEntityID
INNER JOIN Person.Address a
    ON bea.AddressID = a.AddressID
INNER JOIN Person.StateProvince sp
    ON a.StateProvinceID = sp.StateProvinceID
INNER JOIN Person.CountryRegion cr
    ON sp.CountryRegionCode = cr.CountryRegionCode
INNER JOIN Sales.SalesTerritory st
    ON sp.TerritoryID = st.TerritoryID
WHERE PersonType = 'IN' AND st.[Group] = 'North America'
ORDER BY cr.Name, st.Name, p.LastName, p.FirstName;
```

Now we want to run the query so, in the Desktop window, go to the Home ribbon and click on Get Data. In the dialog box, go to Database, find SQL Server Database, and double-click on it.

A new dialog box opens, type in the SQL Server instance name as SqlSrvInstance, and the target database name as SqlSrvDatabase. Click the arrow for Advanced Options, and the dialog box expands. Go to the SQL Statement box and type in the SELECT statement from above.

Click OK, and a preview window is displayed, showing you some of the data that is being imported. Click on Load, and the data will load into Power BI Desktop. When it has loaded, click the Data view and find the new dataset. Right-click it and click on Rename; input the new name as CountryData and press the Enter key.

The dataset is now just as you want it, and you haven't needed to go through all those extra steps you have to do when you import tables one at a time. Extra transformations can still be applied if you want them but, most of the time, that won't be necessary.

Power BI Desktop lets you run all sorts of T-SQL queries when you want data

retrieved from a Server database, such as the EXECUTE statement used for called stored procedures. As an example, the stored procedure called sp\_execute\_external\_script can be called if you want an R script to run or, as you can see in the example below, a Python script:

```
DECLARE @pscript NVARCHAR(MAX);
SET @pscript = N'
# import Python modules
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from microsoftml import rx_oneclass_svm, rx_predict
# create data frame
iris = load_iris()
df = pd.DataFrame(iris.data)
df.reset_index(inplace=True)
df.columns = ["Plant", "SepalLength", "SepalWidth", "PetalLength",
"PetalWidth"]
# split data frame into training and test data
train, test = train_test_split(df, test_size=.06)
# generate model
svm = rx_oneclass_svm(
    formula=~ SepalLength + SepalWidth + PetalLength + PetalWidth",
    data=train)
# add anomalies to test data
test2 = pd.DataFrame(data=dict(
    Plant = [175, 200],
```

```

SepalLength = [2.9, 3.1],
SepalWidth = [.85, 1.1],
PetalLength = [2.6, 2.5],
PetalWidth = [2.7, 3.2]))
test = test.append(test2)

# score test data

scores = rx_predict(svm, data=test, extra_vars_to_write="Plant")

OutputDataSet = scores';

EXEC sp_execute_external_script
@language = N'Python',
@script = @pscript
WITH RESULT SETS(
(SpeciesID INT, Score FLOAT(25)));\

```

Be aware that, to run this script, Machine Learning Services (In-Database) must be installed, and the property called `sp_execute_external_script` enabled.

In our case, the stored procedure runs a Python script used for looking at sample databases with iris flower data and identifying any anomalies. Run the script, and you should get a dataset called IrisData. Each time the query is run, the values will be different because the data is random. In the dataset, you can see the IDs for the iris species and scores that indicate anomalies.

The example doesn't use SQL Server data, but it does show that Power BI Desktop can run Python scripts. And you can create scripts that include data from multiple sources or SQL Server database data.

Perhaps the best part about this is that you get to use the analytical capabilities offered by Python while still being able to import the data in the format that suits you. That lets you create your visualizations easily, providing critical insights into the data you have available.

## Make the Most of SQL Server Data

Being able to run the T-SQL queries in Desktop is a powerful way of accessing the SQL Server data. As well as gaining huge benefits from using the SQL Server query engine, you also get to reduce how many datasets are imported to Power BI Desktop and reduce the number of transformations you need to make.

You do need to have an understanding of T-SQL, but once you have that, you have way more flexibility when you want to work with SQL Server data. However, even if you still import your data one table at a time, you can still make use of some flexible tools for working with Server data and preparing the data so you can create useful visualizations for posting on Power BI service.

As I said earlier, Power BI service does not provide an SQL Server connector. The only way around it is to export the data form the SQL server into a file that would then need to be imported into the Power BI service – you cannot make a direct connection between the service and SQL Server. The more you understand how to use SQL Server data in Power BI Desktop, the better your data can be visualized in many different ways.

# Chapter 10

## Working with Power Query M Formula Language

In this section, we are going to look at how data is imported and transformed using the Power Query M formula language.

In Power BI Desktop, we define datasets by using one query to specify the data to be included and the way the data should be transformed. The query consists of several related steps, each building on the last, and resulting in the final dataset. Once the dataset has been defined, it can be used for the visualizations you might want to add to your reports, and these can then be published to Power BI Service.

At the very core of this query is the Power Query M language. This is a formula language, a bit of mashup language for Power BI Desktop, Excel 2016's Get & Transform import feature, and Power Query. It is a case sensitive language, and, like many languages, the statements from Power Query are a combination of variables, functions, expressions, values (structured and primitive), and other language elements that all come together to define the necessary logic for shaping the data.

Every Power BI Desktop query is just one Power Query expression called let; this has all the code needed for defining the dataset. These expressions consist of two statements -0 let and in – as you can see from the syntax below:

let

    <em>variable</em> = <em>expression</em> [,...]

in

    <em>variable</em>

Each let statement has at least one procedural step the helps define the query. Each procedural step is a variable assignment, and this consists of the name of the variable and the variable value, provided by an expression. That expression

also defines the required logic to add data, remove it and transform it in a particular way and this is where the majority of your work is done, as you will see throughout this chapter.

The variables may be of any type supported by Power Query and must be given a unique name; however, Power Query does offer a certain amount of flexibility in the naming, even allowing spaces. You do need to enclose the name inside a set of double quotes, and it must be preceded with a hashtag – this is a bit of a convoluted way of naming variables, but you can't avoid it. The name you give to your variable is also the name given to the relevant step in the Query Editor's Applied Steps section, so it pays to use names that make some kind of sense.

The let statement does not limit the number of procedural steps you can use, so long as they are practical and necessary. If you do have several steps, they must be comma-separated, and each step must build on the previous one – the variable in a step is used to define the logic in the next step.

Your procedural steps do not need to be defined in the physical order that matches their logical order. You could, for example, refer to a variable in step one that doesn't get defined until the final step. However, this isn't the best route to take as it could lead to your code being tough to debug, and it could cause some confusion. As such, it is best to keep the logical and physical order in synchronization.

A variable value is returned by the in statement, and this value is used for defining the final shape of the dataset. Most of the time, this is the final variable you define in the let statement. A different variable can be specified to the one in the let statement, but, in all honesty, there is rarely a reason to do this. If you want to see the value of any variable at any point in time, simply choose the associated step from Applied steps; when you select any step, you can see what is in the related variable.

## How to Retrieve Data from CSV Files

We are going to be looking at several examples over the course of the chapter, demonstrating the definition of the procedural steps that you need for defining a query. We will base our examples on the titanic dataset, which you can download from <https://vincentarelbundock.github.io/Rdatasets/datasets.html>. This dataset contains a list of the passengers on the ill-fated ship, showing those

who died and those who survived. If you want to follow along with these examples, you must first create a titanic.csv file from the dataset, saving it to where it can be accessed from the Power BI Desktop. I saved the file on my system to C:\DataFiles\titanic.csv. When you go to the page to download the dataset, make sure it is the one called Stat2Data Titanic – there are a few titanic datasets on the site.

The first step to building Power Query scripts is to put a blank query into the Desktop. In the main Power BI window, go to the Home ribbon and click on Get Data. Go to Other and then double-click on Blank Query. Query Editor opens, and you will see, in the Queries pane, a brand-new query. It will be called Query1 or something similar, depending on how many queries you created already.

Go to the Queries pane and right-click the new query. Click on Rename and type in SurvivalLog, then press Enter. Go to the View menu and click on Advanced Editor. When the editor opens, it will already have a newly defined let expression in it.

This statement has one procedural step; it has a variable called Source and an empty string for the expression – the double quotes indicate this. Note that the same variable is in the statement and in the Applied Steps as a new step. We want to change the name of this variable to GetPassengers.

The first added procedural step retrieves the data from the titanic.csv file and saves it to the variable named GetPassengers. Adding the procedure requires that the current let expression is replaced with the expression below:

```
let GetPassengers = Csv.Document(File.Contents("C:\DataFiles\titanic.csv"),
[Delimiter=",", Encoding=1252])in GetPassengers
```

The variable has been specified in both the let and the in statements in the procedural steps. Doing this ensures that the value of that variable is returned when the let expression has been run.

In the procedural step, the expression contains everything that follows the = sign. The function called Csv.Document is used to retrieve the contents of the file in the format of a Table object. Several parameters are supported by the function, but the source data is identified using just the first parameter. And, as part of that parameter, the File.Contents function is required to return just the data in the

document.

We can specify the second parameter as a record containing settings that are optional. Power Query records are sets of fields consisting of a set of brackets containing a name and value pair. In our example, the record has the Delimiter option and the Encoding option, together with their respective values. Delimiter specifies that the CSV document delimiter is a comma, while Encoding specifies that 1252 is the type of text encoding, based on the code page for Windows Western Europe.

That is all you need to do to set a let expression up. Once the code has been entered, click on Done; the Advanced Editor will close, and the procedural step runs. If you go to the Applied Steps section, you will see that GetPassengers is the first step, and this matches the variable name. The expression can also be seen in the window at the top of the dataset, and that is where you can edit the code if you need to.

Whenever you want to add a step, modify an existing one, or delete one, make sure your changes are applied and saved. To do this, just click Save in the top-left of the menu bar and, when asked if you want the changes applied, click on Apply.

## How to Remove Columns from Datasets

The second procedural step for the let statement will take Column7 out of the dataset. As you save your dataset as a Table object, there are a few Table functions in Power Query that you can choose from for updating your data. To do this, we'll be filtering the specific column using a function called Table.RemoveColumns.

To go back to the query, click on Advanced Editor. After the first of the procedural steps, insert a comma; on a new line, use the following code to define another variable and expression pair – this will remove the column:

```
let GetPassengers = Csv.Document(File.Contents("C:\DataFiles\titanic.csv"),  
[Delimiter=",", Encoding=1252]), RemoveCols =  
Table.RemoveColumns(GetPassengers, "Column7")in RemoveCols
```

Two parameters are required in the Table.RemoveColumns function. The first is used for specifying the target table you want to be updated – in this case, it is the

previous step's GetPassengers variable. The second parameter is used to specify the column or columns that are being removed. We only want to remove Column7 from our GetPassengers table.

What is important here is that step two is built based on the first one, and a new result is returned. That result is then assigned to the variable named RemoveCols; this variable has the same dataset that is in the GetPassengers variable with one difference – the data for Column7 is removed.

Once the procedural step has been added, the GetPassengers variable from the in the statement should be replaced with the RemoveCols variable. Then click on Done, and the Advanced Editor will close. Save your changes and apply them.

The Applied Steps section now has a new step, RemoveCols and Column7 has been deleted from the dataset. You can continue to build your steps as you require, using the same steps and logic from here.

## Promote the First Row to Headers

Next, we want the values from the first row in the dataset promoted to the header row, ensuring that those values become the names of the columns. To do this, go to the last defined step and insert a comma after it. On a new line, the variable/expression pair below should be added:

```
PromoteNames = Table.PromoteHeaders(RemoveCols,  
[PromoteAllScalars=true])
```

This expression is making use of a function called Table.PromoteHeaders to ensure the first row is promoted to the column headers. The first parameter is a required one, specifying which table is to be used as the source data – the variable called RemoveCols. The second parameter is an optional one and is called PromoteAllScalars. It is a good parameter to learn about, though; Power Query promotes just number and text values by default, but if you use PromoteAllScalars and set it to true, all the scalar values from the first row are promoted to headers.

The results of the expression are assigned to the variable called PromoteNames, so make sure the in statement is updated with this name.

## Renaming the Dataset Columns

Now we will rename some of the dataset columns using the function called `Table.RenameColumns`. A new `Table` object is returned by the function, containing the specified updates to the column names. Renaming the columns is done by adding the procedural step below into the let statement, making sure a comma is inserted after the previous statement:

```
RenameCols = Table.RenameColumns(PromoteNames, {{ "", "PsgrID"},  
 {"PClass", "PsgrClass"}, {"Sex", "Gender"} })
```

Two parameters are required by the function; the first is the target table name – the variable called `PromoteNames` from the last step – while the second contains a list of column names, both old and new. Power Query lists are values in an ordered sequence, comma-separated and enclosed in a set of curly brackets. In our example, each of the values is a list with two values – the old name and the new name for the column.

## Filtering Dataset Rows

Next, we want to filter out any row that has NA in the Age column, followed by any row that has an Age value of 5 or lower. When the dataset was imported, the Age column was typed as Text automatically, and that means there are three steps to filtering the data, beginning with removing any NA values. The variable/expression pair below should be added to your code to filter the NA values out:

```
FilterNA = Table.SelectRows(RenameCols, each [Age] <> "NA")
```

The function called `Table.SelectRows` will return a table that has just the rows matching the condition we defined – Age not equal to NA. As we saw previously, the first argument for the function is the variable from the previous step, the variable called `RenameCols`.

The next argument is an `each` expression, and this specifies that the value for Age must not be equal to the value for NA. The keyword, `each`, is indicating that the expression has to be applied to every one of the target table's rows. The result of this is a new table, assigned to the variable called `FilterNA`, and this will not contain any rows with an NA value for Age.

Once the NA values are removed, the Age column needs to be converted to a data type of Number; this will ensure that the data can be worked with more

efficiently and effectively; for example, using numerical ages to filter the data. When you change to Age column type, the PsgrID column type can also be changed to Int64. Doing this means the IDs are referenced by integers and not text. The expression/variable pair below must be added to your let statement to do the type conversion:

```
ChangeTypes = Table.TransformColumnTypes(FilterNA,  
    {{"PsgrID", Int64.Type}, {"Age", Number.Type}})
```

The function called Table.TransformColumnTypes is used by the expression to change the types. There are two parameters, the first being the target table name – FilterNA – and the second being the list of the columns you want updated, together with the new type. Each of the values is a list of its own containing the column name and type.

Once you have updated the Age column with the Number type, the ages can be filtered out based on numerical values, as you can see in the procedural step below:

```
FilterKids = Table.SelectRows(ChangeTypes, each [Age] > 5)
```

Now, your dataset should now only have the data required, as it is in the variable called FilterKids.

## Replacing A Dataset's Column Values

Sometimes, you may need to replace the value in a column with another one. We are going to replace two values in the column called Survived. 0 is being replaced with No and 1 is being replaced with Yes and, to do this, we will use a function called Table.ReplaceValue. The procedural step below is added to the code to replace the 0 values:

```
Replace0 = Table.ReplaceValue(FilterKids, "0", "No",  
    Replacer.ReplaceText, {"Survived"})
```

Table.ReplaceValue needs five parameters:

1. **table** – the target table. In our case, this is the variable from the previous step
2. **oldValue** – the value that we are replacing

3. **newValue** – the value replacing **oldValue**
4. **replacer** – this is a replacer function, used to do the replacement operation
5. **columnsToSearch** – the specific column/s where you want the values replaced

Most are pretty self-explanatory; the one that may not be is the replacer parameter. There are several functions that work with Table.ReplaceValue for updating values; we used Replacer.ReplaceText as we wanted to replace text values.

Once the 0 values have been replaced, you can go ahead and do the 1 values, replacing them with Yes, in much the same way:

```
Replace1 = Table.ReplaceValue(Replace0, "1", "Yes", Replacer.ReplaceText, {"Survived"})
```

Now the Survived column values are easier to read for those who don't understand 0 and 1 values. And doing this will also make it less likely that confusion will arise with the data.

## Changing the Column Values Case

You can also change the values by changing how the capitalization used. We will take the Gender column values and change the first letter to a capital letter instead of being completely lower-case. The procedural step below is added to the code in the let statement to make that change:

```
ChangeCase = Table.TransformColumns(Replace1, {"Gender", Text.Proper})
```

The Table.TransformColumns function is used by the expression for updating the values, and two parameters are required. The first parameter is the table we want to be updated, and the second lists all the operations needed. For every operation, we need the target column and the right expression to do the operation. We only have one operation in our examples, so all we need is the Gender column and the function called Text.Proper, wrapped in a set of curly braces. The first letter of every word in the Gender column is converted to a capital letter.

Now, your let expression should look something like this:

```

let
    GetPassengers = Csv.Document(File.Contents("C:\DataFiles\titanic.csv"),
        [Delimiter=",", Encoding=1252]),
    RemoveCols = Table.RemoveColumns(GetPassengers, "Column7"),
    PromoteNames = Table.PromoteHeaders(RemoveCols,
        [PromoteAllScalars=true]),
    RenameCols = Table.RenameColumns(PromoteNames, {{"", "PsgrID"}, {"PClass", "PsgrClass"}, {"Sex", "Gender"}}),
    FilterNA = Table.SelectRows(RenameCols, each [Age] <> "NA"),
    ChangeTypes = Table.TransformColumnTypes(FilterNA,
        {{"PsgrID", Int64.Type}, {"Age", Number.Type}}),
    FilterKids = Table.SelectRows(ChangeTypes, each [Age] > 5),
    Replace0 = Table.ReplaceValue(FilterKids, "0", "No",
        Replacer.ReplaceText, {"Survived"}),
    Replace1 = Table.ReplaceValue(Replace0, "1", "Yes",
        Replacer.ReplaceText, {"Survived"}),
    ChangeCase = Table.TransformColumns(Replace1, {"Gender", Text.Proper})
in
    ChangeCase

```

Included in the let statement are all the steps in your query, each one building on the last one. Note that, in the Applied Steps section, there is one step per variable, in the same order they are in the statement. Choose any step, and you will see the data in the main window showing you what is in the variable.

## **Adding Calculated Columns to Datasets**

Up to now, each of the steps your let statement has represents one discrete action, built on the previous step. Sometimes though, you may need your steps structured in a not-so linear way. For example, you might want a calculated column, showing the difference between the age of a passenger and the average

age for that gender. These are the steps you can take:

1. Take the female passengers, calculate their average age and save the result into a variable
2. Do the same with the male passengers
3. Add a column that will calculate the difference in ages using the above two variables
4. Optional step – you can round the differences up, so they are easier to read

```
// add a calculated column based on the average ages
```

```
Female = Table.SelectRows(ChangeCase, each [Gender] = "Female"),
```

```
AvgFemale = List.Average(Table.Column(Female, "Age")),
```

```
Male = Table.SelectRows(ChangeCase, each [Gender] = "Male"),
```

```
AvgMale = List.Average(Table.Column(Male, "Age")),
```

```
AddCol = Table.AddColumn(ChangeCase, "AgeDiff", each if [Gender] = "Female" then [Age] - AvgFemale else [Age] - AvgMale),RoundDiff = Table.TransformColumns(AddCol, {"AgeDiff", each Number.Round(_, 2)})
```

Let's break this down a bit so you can understand it better:

The first line with the two // (forward slashes) is indicating that this is a one-line comment. Anything that follows the // is ignored and not processed as it is for information only. You can also use multi-line comments in Power Query, using /\* to start them and \*/ to end them. You don't have to include any comments; it's entirely your choice.

The next line calculates the average female age – the two steps below are what provide the value of that average age:

```
Female = Table.SelectRows(ChangeCase, each [Gender] = "Female"),
```

```
AvgFemale = List.Average(Table.Column(Female, "Age")),
```

The first one generates a table using Table.SelectRows. This table only has rows containing a value for the Female Gender. This function is being used in much the same way as what we saw earlier, but this time, we are filtering out different

data and saving the results to a variable called Female. Note that the function used by the source table is the variable from the previous step, ChangeCase.

Step two is using the function called List.Average to calculate the average for the Female table Age values. A scalar value is returned, and this is saved in the variable called AvgFemale. Only one parameter is required, and this includes another function called Table.Column; this will pass the Age column values to the List.Average function.

Next, we want the average male passenger age, and we can do this in a similar way, with just a couple of changes:

```
Male = Table.SelectRows(ChangeCase, each [Gender] = "Male"),
```

```
AvgMale = List.Average(Table.Column(Male, "Age")),
```

Note that the ChangeCase variable must be used for the source table when the function called Table.SelectRows is called, irrespective of the fact that this is no longer the previous step. You can use any of the variables that came before in an expression, but only as long as it is sensible to do it.

Now that you have your AvgMale and AvgFemale variables, the column can be added, using the Table.AddColumn function:

```
AddCol = Table.AddColumn(ChangeCase, "AgeDiff", each if [Gender] = "Female"
```

```
[Age] - AvgFemale else [Age] - AvgMale)
```

Three parameters are required by this function; the first is ChangeCase, the target table, and the second is AgeDiff, which is new column's name.

The third is the expression that will generate the values for the column. The each keyword is used to start the expression, and this will iterate every row in the target table. An if...then...else expression follows this to calculate the value of the row based on male or female passengers. If the value for Gender equals Female, the value for AgeDiff is set as Age less the value for AvgFemale; if it is male, AgeDiff is set as Age less AvgMale.

Once the new column has been defined, the last step is to round the values for AgeDiff to two decimal points:

```
RoundDiff = Table.TransformColumns(AddCol, {"AgeDiff", each
```

```
Number.Round(_, 2)})
```

In the expression is the function called Table.TransformColumns; this time, it uses another function called Number.Round, which rounds the values, instead of altering the case. The function takes two parameters, the first of which is an underscore. This is representing the current value of the column while the second parameter, which is 2, states that two decimal places is what the value should be rounded to.

Those are the steps required for creating calculated columns; now, the let statement should look like this:

```
let
```

```
GetPassengers = Csv.Document(File.Contents("C:\DataFiles\titanic.csv"),
```

```
[Delimiter=",", Encoding=1252]),
```

```
RemoveCols = Table.RemoveColumns(GetPassengers, "Column7"),
```

```
PromoteNames = Table.PromoteHeaders(RemoveCols,  
[PromoteAllScalars=true]),
```

```
RenameCols = Table.RenameColumns(PromoteNames, {{ "", "PsgrID"},
```

```
{"PClass", "PsgrClass"}, {"Sex", "Gender"}}),
```

```
FilterNA = Table.SelectRows(RenameCols, each [Age] <> "NA"),
```

```
ChangeTypes = Table.TransformColumnTypes(FilterNA,
```

```
{ {"PsgrID", Int64.Type}, {"Age", Number.Type}}),
```

```
FilterKids = Table.SelectRows(ChangeTypes, each [Age] > 5),
```

```
Replace0 = Table.ReplaceValue(FilterKids, "0", "No",
```

```
Replacer.ReplaceText, {"Survived"}),
```

```
Replace1 = Table.ReplaceValue(Replace0, "1", "Yes",
```

```
Replacer.ReplaceText, {"Survived"}),
```

```
ChangeCase = Table.TransformColumns(Replace1, {"Gender", Text.Proper}),
```

```
// add calculated column based on average ages
```

```

Female = Table.SelectRows(ChangeCase, each [Gender] = "Female"),
AvgFemale = List.Average(Table.Column(Female, "Age")),
Male = Table.SelectRows(ChangeCase, each [Gender] = "Male"),
AvgMale = List.Average(Table.Column(Male, "Age")),
AddCol = Table.AddColumn(ChangeCase, "AgeDiff", each if [Gender] = "Female" then [Age] - AvgFemale else [Age] - AvgMale), RoundDiff = Table.TransformColumns(AddCol, {"AgeDiff", each Number.Round(_, 2)}))in RoundDiff

```

Again, going to Applied Steps, you should see all the variables we used for finding the average ages, irrespective of the fact that the AddCol step is building on the previous variable of ChangeCase. Select any step to see the variable contents but be aware that, if you choose a variable that is storing the average, you will only see the scalar value in Query Editor.

## **Navigate the Power Query M Formula Language**

It shouldn't surprise you to learn that we have only covered a fraction of what Power Query offers, but you do now have a decent base from which to build queries of your own in the Power BI Desktop. And you should also better understand that way that we construct a query using the built-in point and click features for importing and transforming data. This helps you to examine the code and gain an understanding of why the results may not be as you expect them to be. You can also use the point and click operations to build the query and use Advanced Editor to refine the datasets or bring in some logic that you can't achieve very easily via the interface.

To get the best out of Power Query, you will need to go much deeper into the elements of the language, in particular, into the built-in functions. Do be aware that Power Query was really built for Excel, so you may come up against one or two elements that simply won't transfer to the Desktop easily. That said, the language and the syntax are defined by some basic principles and, the better an understanding you have, the more power you have at your disposal when you are working in Power BI Desktop.

# **Chapter 11**

## **How to Visualize SQL Server Audit Data**

One of the more powerful features for helping you to comply with SOX, HIPAA, and other regulations is SQL Server Audit. However, viewing data collected with the feature is not easy. In our final chapter, I want to show you how you view your SQL Server Audit data and filter it using Power BI.

Business database teams are permanently involved in trying to comply with certain regulations, such as SOX (Sarbanes-Oxley Act, 2002) and HIPAA (Health Insurance Portability and Accountability Act, 1996) and these teams tend to make auditing one of the common parts of their strategies as a way of helping them track any threats to their data. As an example, if a team is running SQL Server, they might use Audit as a way of logging both database and server actions.

SQL Server Audit is already in the database engine, and, from SQL Server 2016 onwards, it is freely available in every edition of SQL Server.

If your organization is a DBA (Doing Business As) and your job is to implement Audit, you will see that it is pretty easy to set up. The hardest part is in working out what user actions need to be audited, how large amounts of data might be handled for audit, and what the best tools are for monitoring the data and reviewing it.

However, important though these considerations are, for the purpose of this chapter, we're going to focus on the final bit – monitoring the data and reviewing it. With SQL Server, you get an easy way to collect data, but you don't get any meaningful way of working with the data; all you can do is manually review it.

Perhaps the best way is to use Power BI. With this, you can quickly create reports that give you some visual insight into your data. Power BI isn't really designed for the purpose of audits and alerts, not as much as some other

dedicated management tools are, but it does allow you to track user behavior relatively easily. And Power BI Service and Desktop are both free to use.

I will be showing you, through a series of examples, how to use Power BI and SQL Server Audit data; we'll look at how to set your test environment up, how to generate some audit data to work with, how to get that data into Power BI Desktop and how to come up with a report containing both visualizations and tables.

Do bear in mind that Power BI Desktop and SQL Server Audit are very powerful, and we cannot possibly cover everything they can do in this section. What I can do is offer you a basic overview of how you can use the tools together and how to get started with reviewing audit data using Power BI.

## **Setting Your Test Environment Up**

Getting your environment set up requires you to create a test database called ImportSales. This has one schema called Sales and a table called Sales.Customers. We can then use the data in the Sales.Customer table in the database called the WideWorldImporters database to populate the table. For our purposes, the audited actions are restricted to the Customers table from the database called ImportSales.

The T-SQL code can be run to create ImportSales:

```
USE master;
```

```
GO
```

```
DROP DATABASE IF EXISTS ImportSales;
```

```
GO
```

```
CREATE DATABASE ImportSales;
```

```
GO
```

```
USE ImportSales;
```

```
GO
```

```
CREATE SCHEMA Sales;
```

```
GO
```

```
CREATE TABLE Sales.Customers(  
    CustID INT IDENTITY PRIMARY KEY,  
    Customer NVARCHAR(100) NOT NULL,  
    Contact NVARCHAR(50) NOT NULL,  
    Email NVARCHAR(256) NULL,  
    Phone NVARCHAR(20) NULL,  
    Category NVARCHAR(50) NOT NULL);
```

GO

```
INSERT INTO Sales.Customers(Customer, Contact, Email, Phone, Category)  
SELECT c.CustomerName, p.FullName, p.EmailAddress,  
    p.PhoneNumber, cc.CustomerCategoryName  
FROM WideWorldImporters.Sales.Customers c  
INNER JOIN WideWorldImporters.Application.People p  
    ON c.PrimaryContactPersonID = p.PersonID  
INNER JOIN WideWorldImporters.Sales.CustomerCategories cc  
    ON c.CustomerCategoryID = cc.CustomerCategoryID;
```

GO

If the WideWorldImporters database hasn't been installed, use data of your own to populate the Customers table. If you want a different database and table to follow these examples, forget the T-SQL statement and use what suits your needs, so long as they are not being used in a production environment. Just make sure that any ImportSales or Customers references are replaced in the rest of the examples.

Next, you need an audit object, and this needs to be created at the SQL Server instance level, and you also need a database audit specification created at the ImportSales database level. The audit object is more of a container, and it is used for organizing the audit settings for the server and the database and for the delivery of the logs at the end. We are going to save our audit data in a local

folder, but Server Audit allows you to save it to the Windows Application or Security logs.

The database audit specification needs to be created at the database level. It must be associated with an audit object, and that object must be in existence before the audit specification can be created. This specification is used to determine the actions that are going to be audited at the database level. A similar specification can also be created for the server audit for audits at the server level, but we will only use the database one for this section.

Both the object and the specification can be created using the T-SQL code below:

```
USE master;
```

```
GO
```

```
CREATE SERVER AUDIT ImportSalesAudit
```

```
TO FILE (FILEPATH = 'C:\DataFiles\audit\'');
```

```
GO
```

```
ALTER SERVER AUDIT ImportSalesAudit
```

```
WITH (STATE = ON);
```

```
GO
```

```
USE ImportSales;
```

```
GO
```

```
CREATE DATABASE AUDIT SPECIFICATION ImportSalesDbSpec
```

```
FOR SERVER AUDIT ImportSalesAudit
```

```
ADD (SCHEMA_OBJECT_CHANGE_GROUP),
```

```
ADD (SELECT, INSERT, UPDATE, DELETE
```

```
ON Object::Sales.Customers BY public)
```

```
WITH (STATE = ON);
```

```
GO
```

The object, called ImportSalesAudit, is created using the CREATE SERVER AUDIT statement and is responsible for saving data to the folder called C:\DataFiles\Audit. Then the ALTER SERVER AUDIT statement is run, so the STATE property is set to ON.

Next, the CREATE DATABASE AUDIT SPECIFICATION statement is used to define the ImportSalesDbSpec specification, which has two ADD clauses. The first one specifies SCHEMA\_OBJECT\_CHANGE\_GROUP action group, which is responsible for auditing all the ALTER, CREATE, and DROP statements that get issued against any of the database schema objects. As with all group actions, this has to be separately specified from the individual ones, like those on the other ADD clause.

There are four such actions in this ADD clause:

- **SELECT** – used for auditing SELECT statements
- **INSERT** – used for auditing INSERT statements
- **UPDATE** – used for auditing UPDATE statements
- **DELETE** – used for auditing DELETE statements

In the second ADD clause, the ON subclause is pointing to the Customers table; this means that the four actions – INSERT, SELECT, DELETE, and UPDATE – are all specifically for that table. And, because the public login is specified by the BY subclause, the auditing is applicable to every user.

Normally, you would be auditing a lot of users and actions, but what we've done here is sufficient to show you the basics of reviewing audit data using Power BI.

Once the audit structure is built, the T-SQL code below can be run; this creates three test user accounts in the database called ImportSales and assigns each user with its own set of permissions:

```
CREATE USER User01 WITHOUT LOGIN;  
GRANT ALTER, SELECT, INSERT, DELETE, UPDATE  
ON OBJECT::Sales.Customers TO user01;  
GO  
CREATE USER User02 WITHOUT LOGIN;
```

```
GRANT SELECT, INSERT, DELETE, UPDATE  
ON OBJECT::Sales.Customers TO user02;  
GO
```

```
CREATE USER User03 WITHOUT LOGIN;  
GRANT SELECT  
ON OBJECT::Sales.Customers TO user03;
```

```
GO
```

Logins are not given to the user accounts at the time they are created, just to keep things simpler. And, for that reason, all the permissions granted are specific to the table called Customers and access is defined like this:

- **User01** – has permission to access all data in the table, modify the data and update the definition of the table
- **User02** – has permission to access all data in the table, modify the data, but cannot update the definition of the table
- **User03** – has permission to access all data in the table but cannot modify the data nor can it update the definition of the table

The test users can be set up with their permissions, in whatever way you want; just make sure they are in place when you want the audit data generated for use in the Power BI Desktop.

## Generate the Test Audit Data

Generating the audit data requires several DML (data manipulation language) and DDL (data definition language) statements to be run against the table called Statement. These must be run in the execution context of each of the user accounts, and the best way to do this is:

- Specify user context using an EXECUTE AS statement
- Run at least one statement
- Go back to the original user by running a REVERT statement

The DDL and DML statements you run are entirely up to you, so long as each

account is tested, along with the account permissions. I ran several T-SQL statements, many of them multiple times, so that I could get a decent amount of data I started with these DML accounts and they were run under User01:

```
EXECUTE AS USER = 'User01';
SELECT * FROM Sales.Customers;
INSERT INTO Sales.Customers
(Customer, Contact, Email, Phone, Category)
VALUES('Wingtip Toys (Eugene, OR)', 'Flora Olofsson',
'flora@wingtiptoys.com', '(787) 555-0100', 'Gift Store');
DECLARE @LastID INT = (SELECT SCOPE_IDENTITY())
UPDATE Sales.Customers SET Category = 'Novelty Shop'
WHERE CustID = @LastID;
DELETE Sales.Customers WHERE CustID = @LastID;
REVERT;
GO
```

Do run the statements as many times as needed to generate the amount of audit data you want. I ran my DML statements around five times and the DDL statements a couple of times.

Once you have run the statements, run the next lot as User01 – these will add a new column into the Customers table:

```
EXECUTE AS USER = 'User01';
ALTER TABLE Sales.Customers
ADD Status BIT NOT NULL DEFAULT(1);
REVERT;
GO
```

Now, as User02, do the DML statements again, running them several times:

```
EXECUTE AS USER = 'User02';
SELECT * FROM Sales.Customers;
INSERT INTO Sales.Customers
(Customer, Contact, Email, Phone, Category)
VALUES('Tailspin Toys (Bainbridge Island, WA),' 'Kanti Kotadia,' 
'kanti@tailspintoys.com', '(303) 555-0100', 'Gift Store');
DECLARE @LastID INT = (SELECT SCOPE_IDENTITY())
UPDATE Sales.Customers SET Category = 'Novelty Shop'
WHERE CustID = @LastID;
DELETE Sales.Customers WHERE CustID = @LastID;
REVERT;
GO
```

As User02, attempt to add a column to Customers using the T-SQL statement below:

```
EXECUTE AS USER = 'User02';
ALTER TABLE Sales.Customers
ADD LastUpdated DATETIME NOT NULL DEFAULT(GETDATE());
REVERT;
GO
```

You should get an error generated. User02 does not have permission to modify the definition of the table. However, be aware that, when you run several statements as one user, and one of them fails, the REVERT statement doesn't run; you would need to run it again without the rest of the statements to make sure the execution context is closed. The better way of doing things would be to make sure the correct logic is written into the code to make sure that REVERT always runs.

Now, as User03, run the DML statements again:

```
EXECUTE AS USER = 'User03';

SELECT * FROM Sales.Customers;

INSERT INTO Sales.Customers

(Customer, Contact, Email, Phone, Category)

VALUES('Tailspin Toys (Bainbridge Island, WA)', 'Kanti Kotadia,'

'kanti@tailspintoys.com', '(303) 555-0100', 'Gift Store');

DECLARE @LastID INT = (SELECT SCOPE_IDENTITY())

UPDATE Sales.Customers SET Category = 'Novelty Shop'

WHERE CustID = @LastID;

DELETE Sales.Customers WHERE CustID = @LastID;

REVERT;

GO
```

This time, three statements will generate errors – INSERT, UPDATE, DELETE – because User03 does not have the correct permissions. As before, the REVERT statement will need to run without the other statements. The same would apply to the ALTER TABLE statement:

```
EXECUTE AS USER = 'User03';

ALTER TABLE Sales.Customers

ADD LastUpdated DATETIME NOT NULL DEFAULT(GETDATE());

REVERT;
```

Again, run the DDL and DML statements that you want, ensuring you have sufficient data for Power BI Desktop.

## **Creating the Connection to SQL Server Audit Data in Power BI Desktop**

When you are in Power BI Desktop, and you want to connect to SQL Server, the data can be pulled from specified views, and tables or a query can be run to return the exact data needed from several views and tables. Queries also allow

you to use system functions, like sys.fn\_get\_audit\_file. This is a function that contains table values and returns the SQL Server Audit log file results.

We will use the function in the SELECT statement below and return five things:

- User account
- Action
- Success status
- T-SQL statement
- The time of each logged event

```
SELECT f.database_principal_name [User Acct],  
(CASE  
WHEN a.name = 'STATEMENT ROLLBACK' THEN 'ROLLBACK'  
ELSE a.name  
END) [User Action],  
(CASE  
WHEN f.succeeded = 1 THEN 'Succeeded'  
ELSE 'Failed'  
END) [Succeeded],  
f.statement [SQL Statement],  
f.event_time [Date/Time]  
FROM sys.fn_get_audit_file  
('C:\DataFiles\audit\ImportSalesAudit_*.sqlaudit',  
default, default) f  
INNER JOIN (SELECT DISTINCT action_id, name  
FROM sys.dm_audit_actions) a  
ON f.action_id = a.action_id
```

```
WHERE f.database_principal_name IN ('User01', 'User02', 'User03')
```

The function called `sys.fn_get_audit_file` is joined to the function called `sys.dm_audit_actions`, returning the full name of the action instead of an abbreviated name. The statement also ensures the results are limited to our three user accounts.

This is the statement required when you set up the connection between Power BI Desktop and SQL Server. Configuring the connection requires a new report to be created in Power BI Desktop. Once that is done, go to the Home ribbon and click on Get Data, then on SQL Server. The dialog box opens; expand it by clicking on the arrow beside Advanced Options.

Configure the connection like this:

1. In the Server box, type in the Server instance - `.\sqlsrv17a`
2. In the Database box, type the database name – `ImportSales`
3. In the Data Connectivity section, choose an option – I went for `DirectQuery`
4. In the SQL Statement box, input the SELECT statement

When DirectQuery is chosen, none of the data is copied or imported into Desktop. Instead, the underlying data source is queried by Power BI Desktop whenever a visualization is created or interacted with. This ensures that you are always getting the most up to date data.

However, you should be aware that, if you want your report published to Power BI service, a gateway connection must be created so that the service can get the source data. Also, be aware that it isn't possible to create a report in Power BI Desktop with an Import SQL Server connection and a DirectQuery SQL Server connection – it must only have one of them.

Once the connection has been configured, click on OK, and a preview window opens; here, you can see a subset of the audit data. If it all looks as it should, click on Load, and the data is made available to Power BI desktop. The data can now be used to add visualizations or tables. If you chose DirectQuery, you would only be loading the schema into Desktop, not the source data.

As I said earlier, you do not have to save your data to log files. However, if you

opt to save it to the Security or the Application log, you will need to do more to extract the data from the logs and convert it to an easier format, like a .csv file.

You could, for example, export Log File Viewer or Windows Event Viewer data in SSMS – SQL Server Management Studio. However, the format won't be easy to work with, and you might find you have far more data than you need. You could also use PowerShell to get the log data; you would need to create a script to run automatically, but you would have more control over what the output was, although it still takes a little extra work to make sure it's right.

However you make your data available to Power BI Desktop, you must consider data security. Using the Security log might seem to be a safe approach to start with, but once the data has been exported to the files, you will face the same problems as you do when the data is sent straight to log files. The data has to be fully protected all the time, in-motion, and at-rest; there wouldn't be any point to an audit strategy that fully complies with regulations if the data is put at risk during the audit.

Once the data is available to Desktop, you can go ahead with your reports, presenting your data in ways that provide different types of insight.

## **Adding Tables**

One good way of showing data is to make some of it available in table form. The data can be filtered as you want, depending on what you need for your audit strategy. You can also add slicers to filter the data in a specific way, for example, by Users, Actions, and Successes and Failures.

When a filter is applied, the data in the table is updated by Power BI, based entirely on the values you chose. Doing this gives you an easy way of accessing the different data categories without having to go through it all manually or having to generate a T-SQL statement each time you want different information

## **Adding a Matrix**

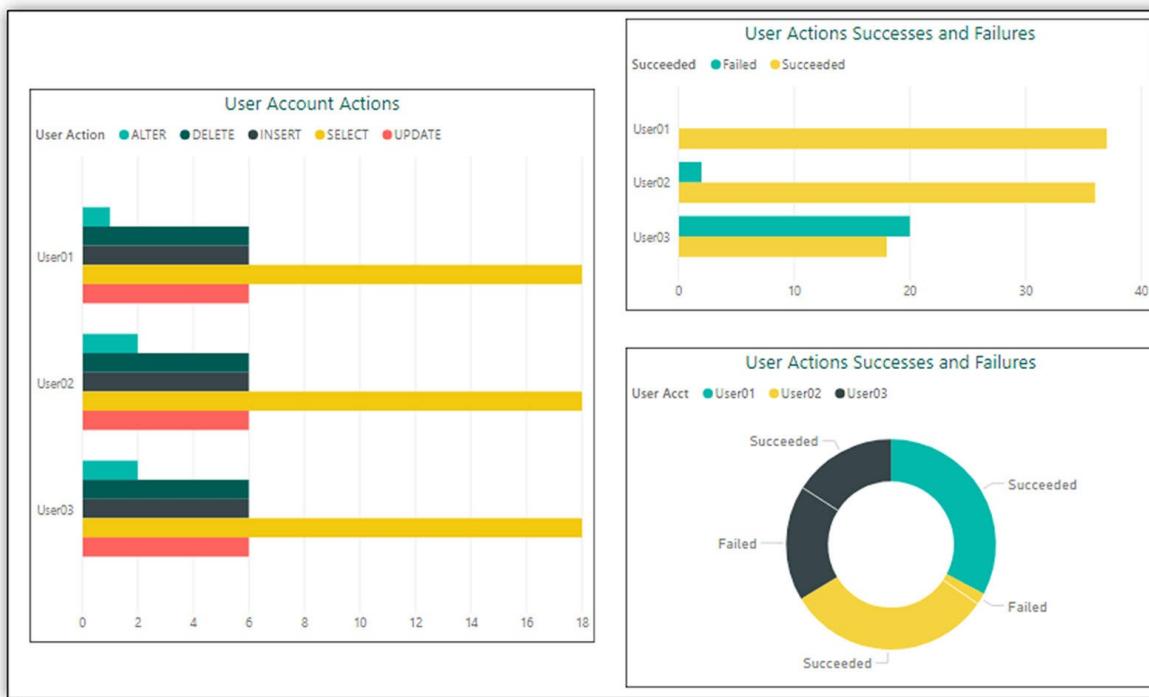
Another good way to get some quick data insights is to summarize the data using a matrix. You could, for example, add a matrix showing how many actions each user took and what those actions were. You will spot, on the matrix page, a list of user actions on the right side – one of these is called ROLLBACK; the ROLLBACK statement is required when the main statement fails, such as when

there are no permissions for a certain action.

Slicers can also be added to the matrix report page, allowing you to filter the data as you need it. The matrix is incredibly useful because it can be set up, so it drills down into your data, i.e., into how many actions were successes or failures. When you set this up, you decide the order each layer is presented in, dependent on what type of data you are using and what its hierarchical nature is.

## Adding Visualizations

Lots of different visualizations are supported in Power BI Desktop, and you also have the ability to import others. Make sure you only use the visualizations that help explain your data clearly. Say, for example, that your report page shows three visualization types; each one gives a different perspective into your data. You can filter the ROLLBACK actions out; that way, your visualizations will only reflect the user-initiated statements, instead of those that are response-generated.



Look at the figure above; on the left, the clustered bar chart shows the data grouped by the user, providing a total number of actions per user. The DML action indicates that all three user accounts have the same total – that will depend on how many times the statements were run.

At the top right, another clustered bar chart shows the data grouped by the user again, but, this time, showing how many successes and failures there were for each group. This way, you can see which users are trying to run statements they don't have permission for.

At the bottom-right, you can see a donut chart. This shows the same data as is in the clustered bar chart above but from a different perspective. Hovering over any one of the elements shows you the total action percentage.

What all this is showing you is that you can use the same data in different visualizations to see what works and what doesn't.

When elements are placed on one report page, they are tied together automatically by Power BI. By doing this, you can choose an element from one visualization, and that choice is reflected in the others.

## **Adding a Gauge**

Lastly, you can also add gauges, cards, KPIs (key performance indicators), and other elements to your reports. You could add a gauge that shows how many ROLLBACK statements were executed as a response to failed attempts at running T-SQL statements.

By adding a slicer, you can also see other data, for example, if each user were hitting the threshold specified. When you select a user in the slicer, you need to specify a target value. This is important because it means that target value is used to set alerts in Power BI Service – this cannot be done in Power BI Desktop – but you will require a Power BI Pro license to make use of these features. Alerts can also be set to generate regular email notifications, to KPI visuals and cards.

## **Using Power BI to Visualize Audit Data**

When you use Power BI to visualize your SQL Server audit data, you are using one of the more powerful tools available for reviewing information efficiently and quickly. I really have only touched on the tip of the iceberg here, but you should now have a decent understanding of how to use the two together for auditing purposes. However, you choose to use them is up to you, but you must remember to protect the integrity of the audit data, be it at-rest or in-motion, and wherever it comes from.

# **Conclusion**

---

Thank you for taking the time to read my guide. I hope you now have a better understanding of some of the advanced features in Power BI.

I have not even scratched the surface with this; Power BI is incredibly complex, and it offers so many different features that it would take an entire series of guides – long ones at that – to try and explain them all. What I have done, I hope, is to give you an idea of some of the more common advanced features you might use in Power BI to create reports and visualize your data in dynamic and interesting ways.

What you should do now is take what you have learned here and build on it. There are plenty of online courses you can take to further your knowledge, and, really, the best way to get to grips with it all is simply to use it. Create a dataset and play about with it. Apply new features, new visualizations and learn what its all about and what Power BI service and Power BI Desktop can do for you.