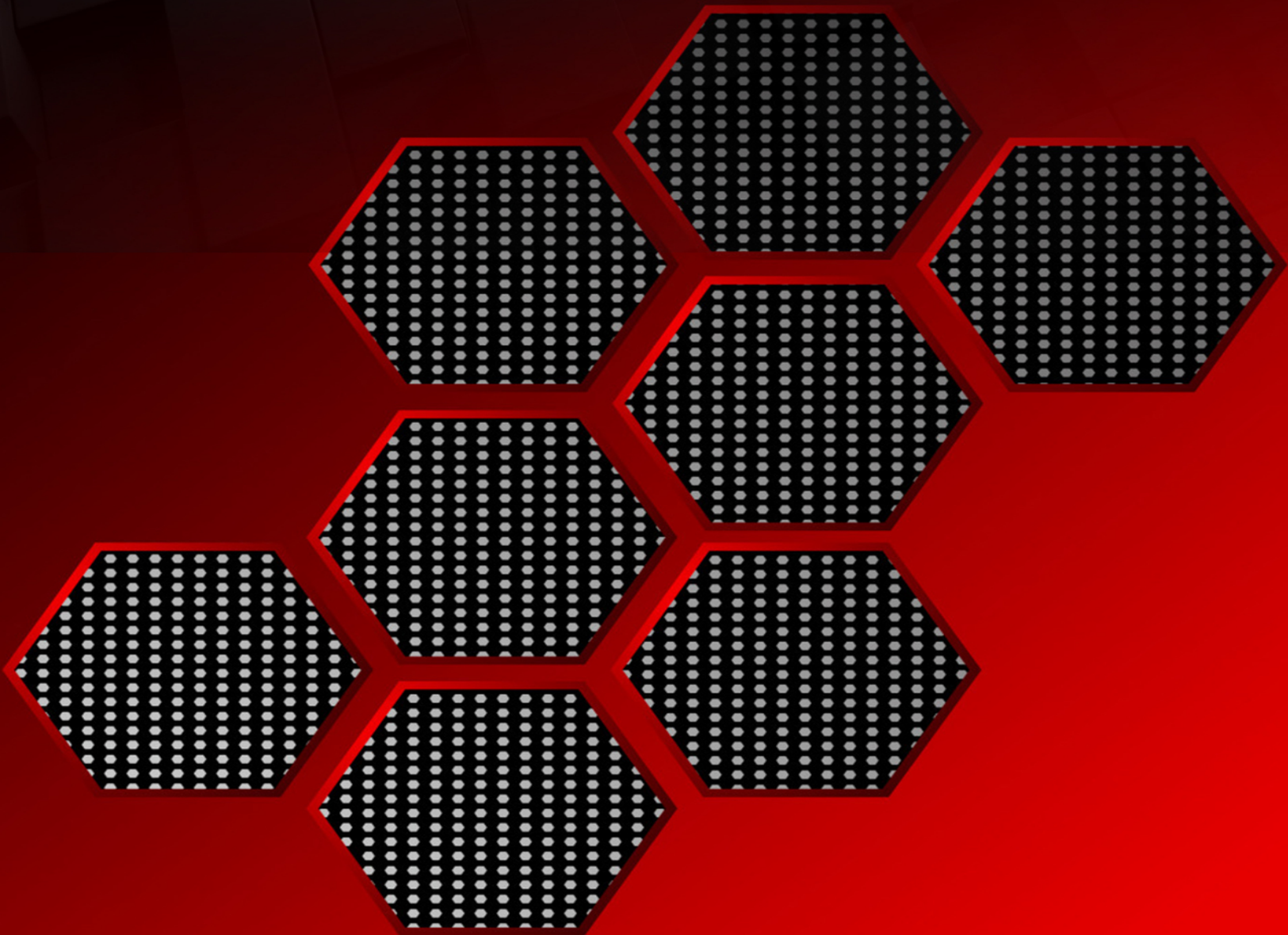


# POWER BI

---

MOVING BEYOND THE BASICS OF POWER BI AND  
LEARNING MORE ABOUT DAX LANGUAGE

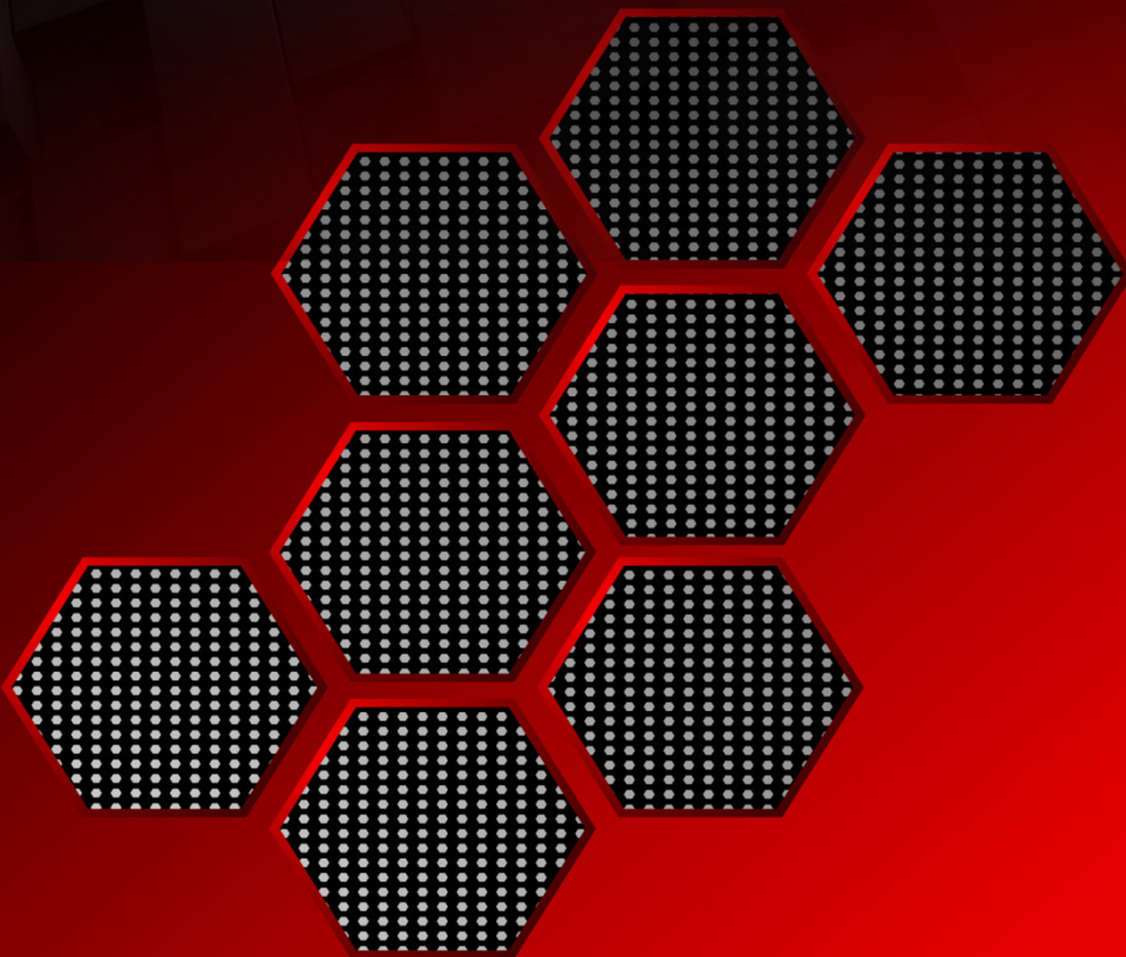


MIKE MORRIS

# POWER BI

---

MOVING BEYOND THE BASICS OF POWER BI AND  
LEARNING MORE ABOUT DAX LANGUAGE



MIKE MORRIS

# **Power BI**

*Moving Beyond the Basics of  
Power BI and Learning More  
about DAX Language*

**© Copyright 2020 - All rights reserved.**

The contents of this book may not be reproduced, duplicated, or transmitted without direct written permission from the author.

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.

**Legal Notice:**

This book is copyright protected. This is only for personal use. You cannot amend, distribute, sell, use, quote, or paraphrase any part of the content within this book without the consent of the author.

**Disclaimer Notice:**

Please note the information contained within this document is for educational and entertainment purposes only. Every attempt has been made to provide accurate, up to date, and reliable, complete information. No warranties of any kind are expressed or implied. Readers acknowledge that the author is not engaging in the rendering of legal, financial, medical, or professional advice. The content of this book has been derived from various sources. Please consult a licensed professional before attempting any techniques outlined in this book.

By reading this document, the reader agrees that under no circumstances is the author responsible for any losses, direct or indirect, which are incurred as a result of the use of the information contained within this document, including, but not limited to, —errors, omissions, or inaccuracies.

## Table of Contents

### [Introduction](#)

### [Chapter One: An Introduction To Power BI](#)

#### [An Overview](#)

#### [Components Of Power BI](#)

#### [Types Of Connection In Power BI](#)

#### [Power BI For All](#)

#### [Concept Of Power BI](#)

#### [Installation of Power BI](#)

### [Chapter Two: Why Should You Use Power BI?](#)

#### [Access and Store a Vast Amount of Data \(with No Issue\).](#)

#### [Spot Data Trends Quickly and Easily.](#)

#### [Intuitive UX Features](#)

#### [Incredible Cloud-Based Features](#)

#### [Row Level Security Features](#)

### [Chapter Three: Power BI Data Sources](#)

#### [All](#)

#### [File](#)

#### [Database](#)

#### [Import Option Versus DirectQuery Option](#)

#### [Azure](#)

#### [Online Services](#)

#### [Other](#)

## [Chapter Four: Comparison Of Power BI And Other Tools](#)

[Power BI Versus Tableau](#)

[Power BI Versus SSRS](#)

## [Chapter Five: Data Modeling In Power BI](#)

[Using Navigation And Data Modeling](#)

[Creating Calculated Columns](#)

[Creating Calculated Tables](#)

[Managing Time-Based Data](#)

## [Chapter Six: Power BI Dashboard Options](#)

[Exploring Different Datasets](#)

[Creating Dashboards](#)

[Sharing Dashboards](#)

[Tiles In Dashboards](#)

[Data Gateway](#)

## [Chapter Seven: Power BI Visualization Options](#)

[Creating Simple Visualizations](#)

[Creating Map Visualizations](#)

[Using Combination Charts](#)

[Using Tables](#)

[Modify Colors In Charts](#)

[Adding Images, Shapes And Text Boxes](#)

[Styling Reports](#)

[Duplicating Reports](#)

## Chapter Eight: How To Share Power BI Dashboards

Using Power BI Desktop To Share Reports

Printing Power BI Dashboards

Export Options

Publishing Report to Web

Using Content Pack

Editing Content Pack

## Chapter Nine: Introduction To DAX

An Introduction To DAX

Importance Of DAX

DAX Functions

DAX Calculation Types

## Chapter Ten: DAX Parameter Naming Languages

Parameter Names

Prefixing Parameter Names or Using the Prefix Only

## Chapter Eleven: Description Structure Of DAX Functions

Description

Syntax

Parameters

Return Value

Remarks

## Chapter Twelve: Aggregate Functions

ADDCOLUMNS

[AVERAGE](#)

[COUNT](#)

[CROSSJOIN](#)

[DISTINCTCOUNT](#)

[GENERATE](#)

[MAX](#)

[MIN](#)

[PRODUCT](#)

[ROW](#)

[SELECTCOLUMNS](#)

[SUM](#)

[SUMMARIZE](#)

[SUMMARIZE With Options](#)

[SUMX](#)

[TOPN](#)

## [Chapter Thirteen: Filter Functions](#)

[ADDMISSING ITEMS](#)

[ALL](#)

[ALLEXCEPT](#)

[CALCULATE](#)

[CALCULATETABLE](#)

[CROSSFILTER](#)

[DISTINCT](#)



EARLIER

EARLIEST

FILTER

FILTERS

HASONEFILTER

HASONEVALUE

ISCROSSFILTERED

ISFILTERED

KEEPFILTERS

RELATED

RELATEDTABLE

USERELATIONSHIP

VALUES

## Chapter Fourteen: Time Intelligence Functions

CLOSINGBALANCEMONTH

CLOSINGBALANCEQUARTER

CLOSINGBALANCEYEAR

DATEADD

DATESBETWEEN

DATESINPERIOD

ENDOFMONTH

FIRSTDATE

FIRSTNONBLANK

## [Chapter Fifteen: Tips To Using Power BI](#)

[Keep The Visualization Simple](#)

[Identify The Context](#)

[Slice, Dice, and Filter](#)

[Hierarchies](#)

[Think About The Message](#)

[Conclusion](#)

[References](#)

# Introduction

It was in the year 2009 that Microsoft introduced the concept of Self-Service Business Intelligence, and in the year 2010, Microsoft launched the application Power Pivot as an add-in for Microsoft Excel. At that time, the company did not market the product, but business intelligence users who came across this tool began to use it extensively for their research. They could gather insights about the data set from the Power Pivot add-in. Since Microsoft did not promote this product, numerous users still did not have access to the tool. Microsoft, however, used this time to gather insights and feedback from users, and it developed the Power BI application that is available to you now.

Power BI is an extension of the Microsoft Excel Add-ins Power Query, Power Pivot, and Power View. Now, you can use Power BI either with or without Excel. Based on the various insights and feedback that Microsoft obtained, it developed a mobile application and one that did not depend on the version of Microsoft Office installed in your system. They also included features that allowed users to import data from a variety of sources, both online and on-premises. In summary, Microsoft did what it does best – it listened to the users and developed a tool that catered to the needs of all users in the business intelligence field .

Power BI has suddenly gained a lot of popularity, and many users, including yourself, are doing their best to learn more about this amazing tool. This book is an effective resource that will shed light on Power BI. You will learn about the various features of the tool and also understand the capabilities of this tool. This book will not only shed light on Power BI but will also provide information about the DAX language, which is used to run queries and perform other

functions in Power BI. We will also look at some tips you can use to change the way you visualize the data in the data set. Make sure that you stick to these tips if you want to simplify the visualizations that you create for any data set. You will also gather some tips on how to filter the data, which will make it easier for you to work with large volumes of data.

# **Chapter One: An Introduction to Power BI**

If you frequently work with data and are looking for a way to visualize your output, Microsoft Power BI is the tool you must use. In this chapter, we will look at some concepts and components of Microsoft Power BI. Microsoft Power BI is one of the most powerful business intelligence tools present in the market. This tool Enable you to visualize data in multiple ways. You can also use a programming language called DAX language, to improve the way you visualize your data.

## **An Overview**

Like every other company, Microsoft also came up with a business analytics service that is situated in the cloud. This tool is Power BI. Power BI allows any user to interpret and visualize data with a higher perception, read, and performance. You can view your data differently because of the different content packs that are built into the dashboard. Power BI also has built-in reports and other formats present in the dashboard. These reports are developed by popular service providers like Google Analytics or Salesforce.com.

You can also obtain a 360-degree view of your business using simply one dashboard. Power BI allows you to display real-time data across all major operating systems on any device. You can set alerts to your emails or mobile phone to inform you about any changes in your data. This will allow you to share your data and update the dashboard easily.

Power BI also provides a drag and drop feature that enables you to delve deeper with your data. You can also ask new questions and further explore the data to see how the variables work together. You do not need to learn a new language if you want to use Power BI. However, the DAX language will make it easier for you to improve your analysis. Power BI allows you to clean, model, visualize, and analyze data. This will make it easier for you to share reports with the business.

Power BI allows you to connect to fifty different applications. You can use the Power BI desktop to create a visualization of your data. You can then publish this visualization to the Power BI service and allow users to connect to the dashboard to access the data. This data can be accessed using tablets, smartphones, laptops, and computers.

## **Components of Power BI**

Since Power BI is a Microsoft application, it consists of a windows desktop application. This application is called the Power BI desktop. There is also an online service called the Power BI service, and Microsoft has also developed applications for Android, iOS, and Windows devices. Let us now understand the components of Power BI .

## **Power Query**

The Power Query will allow you to explore, access easily, and also convert internal and public data sources. This is the first component of the Microsoft business intelligence system. Microsoft also came up with the Power Pivot and Power View immediately after the power query. You can add the Power Query to your Microsoft Excel application on your system if the version of the application is either 2010 or 2013. Microsoft Excel 2016 has a built-in tool that you can install to extract and transform any data. Power Query will allow you to search and extract required information or data, in the form of excel data, SQL service data, JSON files, text files, etc., from numerous sources. These sources can be internal or external.

You obtain the data; you can process it by applying the Power Query like you would in SQL. You can then pull back just the required information from that data set. You can also transform, process, clean, and combine data with other sources that are relevant to your analysis. Power BI will then store this data in the connection and load it in a model or Excel for any further analysis in Power Pivot, visualization with Power View, or in Excel.

Microsoft Excel is the foundation of most businesses. Microsoft Excel uses very complex software. People are willing to import data into Excel and use different functions in Excel to manipulate that data. This is very time consuming since it takes some time to import data from external sources into excel. Using a power query, you can import data in a matter of seconds .

As mentioned earlier, Power Query will allow you to obtain data from different sources, including open data, big data, and social media, and then quickly process, transform or clean that data into a form that can be used in Excel. Microsoft Excel is a tool that almost every user is familiar



with. Power Query will also help to split and merge columns, replace text, find text, delete duplicates, and so much more. You can save these queries in Power BI and use them repeatedly.

## **Power Pivot**

Power Pivot will allow you to model data to perform in-memory analytics. It is difficult to use large volumes of data or large datasets to perform analysis. This component of Power BI is a powerful data modeling feature. This is an invaluable tool that will take large quantities of data and process it so that you can analyze it with ease. This component will also allow you to customize various data models with the data you have obtained in a flexible and familiar Excel environment.

In just a few clicks, you can create custom measures, hierarchies, relationships, and KPIs to build your models. You can also build your models by dragging and dropping the required data variables from the data set into your diagonal view. Since Power Pivot uses the in-memory technology, you can perform analytics very quickly. This technology makes it easier to process and analyze millions of rows of data at once.

Power pivot also works very well with Microsoft Excel. This means that you can refresh your model with the real-time data from Power Query. This will allow you to visualize various aspects of your data in the reports and power view. You can also create some models that will enable you to query the data using SQL, natural languages, Power BI Q&A, and DAX language. You will love Power Pivot if you have a large volume of data to analyze.

## **Power View**

Power View will allow you to analyze, illustrate, and visualize data using interactive data visualization methods. Before we understand how you can analyze data using Power View, let us understand it as a feature and the product. Power View is very simple. You can just click once or twice to do anything using Power View. It was designed so that it can be used everywhere.

The objective of Power BI is data exploration. In power view, anything you do will have an impact directly on the data set that you send as an input into power bi. Since you can see how the different variables in the data set work together and can shape or change the data, it will make it easier for you to analyze and explore the data better. So, this tool is an ad-hoc exploration tool and not a batch reporting tool. The outputs that you see are very visual since this component enables you to use graphs, animation, and charts. You don't just dump the data from rows and columns into a chart.

Power View is the data visualization component of Power BI, and it is focused on active graphical exploration and visualization. It is also easy to use this component to develop presentations. You can use the DAX language to change the way you look at dates and also tell some stories from data. You don't just have to dump the output. Power View is a relatively new feature, and it fills a product gap by enabling a single-click data explosion and providing a presentation environment that allows users to perform explorative data analytics.

## **Power Map**

You can use Power Map to use interactive geographical visualization to bring data to life. It is easy to download this component and work with it. Let us look at the data model. Some of these fields are:

- Age group
- District
- City code and more

In this analysis, we will predict the snowfall in India based on the districts in different states. These districts are close to the sandstorm origins. Another sheet in the data extract will also provide information about the population in each of these districts. When you compare these data, you can identify which state will be hit worse.

You can create these maps using the command Launch Geoflow under the Insert tab. This command will launch the map and give you an initial view of the Bing maps. The data views directly come from the power pivot workbook, and it is organized in different layers. You should start with each layer and give it a name .

## **Power BI Service**

You can use the Power BI service to share workbooks and data viewpoints. You can use this service to refresh the data either from cloud-based data and on-site sources. There are some important concepts that you should understand when you use this service. The important things to consider when it comes to Power BI are dashboards, workspaces, datasets, and reports. A workspace is a platform or container that can have numerous dashboards within them. You can include numerous tiles within a dashboard, and each of these tiles can be pulled from numerous reports. These reports will only be attached to one data set alone.

For instance, when you are logged into Power BI, you can share information present on one dashboard with other people. The dashboard is the place you work from. There are numerous maps, tiles, doughnut charts, bar charts, and other visualization tools that you can use to perform your analysis.

## **Power BI Q&A**

Through Power BI Q&A, you can ask the right questions and get answers quickly using the natural language query. Using the Power BI Q&A, you can create a visual space of data whenever you look at the dashboard. You can ask questions about the data set that you need answers to. All you need to do is check the box “Ask a question about data” to answer the right questions about the data set. When you check the box, Power BI will list the questions that you can ask about the data set .

For instance, if you have a data set that includes information about products and customers, you can click on the option “Products,” and you will see a list of the products in the data set. You will also see a list of questions that you could ask about the data set. In this list, you will also find

questions with an underline, and exchange the questions based on what you are looking for. The chart on the dashboard will show you all the information based on your selection. The Power BI Q&A will show you the best visual of the data set that you are looking at.

## **Data Management Gateway**

When you use the data management gateway, you can get recurrent expose tables, view data feeds, and also data refreshers. Microsoft Data Management Gateway will allow you to join the data sources on-premises to the cloud services. This will allow you and anybody else to use the data from these sources for their analysis. Microsoft offers numerous cloud services like Azure Data Factory and Power BI for Office 365. These services will give you numerous benefits, including:

- Fast deployment of data models
- Low sustaining costs
- An adjustable business models

These features also enable you to keep your data on-premises.

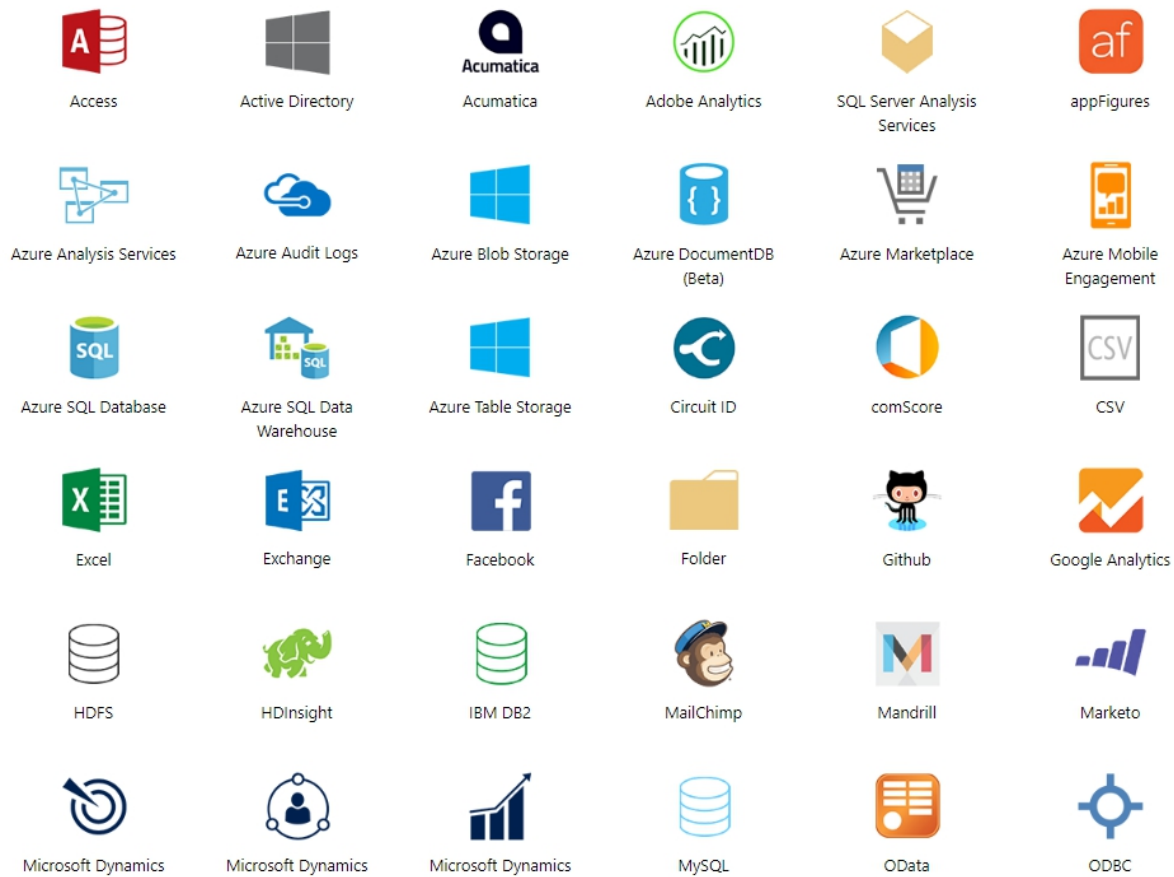
The Data Management Gateway allows you to attach any data that you may have on-premises to the various cloud services you use. You can do this in a controlled and safe way. This will enable you to react quickly to any changes in the market needs on a hybrid and adjustable cloud platform. You can continue to analyze the data while you still run your businesses.

## Data Catalog

Users can identify the right queries to use on a data set and reuse those queries using the Data Catalog option. You can use Metadata to improve search functionality.

## Types of Connection in Power BI

It is very simple to create a dashboard using Power BI since there are close to 50 connections that you can use to import numerous data sets. Let us look at some of the popular services from the official website of Microsoft.



This is, however, not an exhaustive list of applications that Power BI is connected to. When you have these data connections available, it will make your life easier to analyze large volumes of data.



# Power BI for All

## **For Analysts**

An analyst can use Power BI to work on the data using the “Action” option. You can connect a hundred different sources to Power BI, prepare data with ease, develop beautiful reports, and more. You can do this in a matter of minutes.

## **For Business Users**

As mentioned earlier, you can view a dashboard on your phone or web. This will help you always be in the know about what is happening with your data. You can also set an alert notification on your phone or email to send you an alert if there is some change in the dataset. You can also manipulate and control the data in the set in a matter of minutes.

## **For Information Technology**

You can always give users access to numerous insights about the data set. You can give them access to the analysis and also keep the data secure. This can be done easily .

## **For Developers**

Power BI allows you to create interactive visuals, and you can easily embed these reports on any device. This will make your application come alive using the data.

## **Concept of Power BI**

You should have a good understanding of the following if you want to create a complex and detailed report about the data set.

## **Data Sets**

You have heard the term data set being used repeatedly across this chapter. A data set is the accumulation of information or data. Power BI can harness all this information about the data set and create a visualization of the data set. These visualizations can always be a mix of pure data sets or different channels. You can screen these visualizations and blend them together to obtain a completely different data set. This data set will be an aggregation of data. Power BI will use this new data set to create a visualization. For instance, you can gather information about the data set from numerous channels like databases, excel tables, social media platforms, and various email surveys.

## **Visualizations**

A visualization is a visual depiction of the data set that you are analyzing. For instance, you can use a graph or chart to interpret different variables in a data set visually. Power BI allows you to use different types of visualization, and there are new types of visualization being introduced in every new version of Power BI. Some great visualizations are:

- Card Visualization
- Tree map
- Map representation
- Pie chart
- Stacked area chart

You can either use visually detailed or plain visualizations.



## **Reports**

You can aggregate different visualizations on one web page or many web pages using a Power BI report. This report is an accumulation or a combination of unknown correlations, hidden insights, and patterns. You can use the different variables in the data set to create visualizations on more than one page if required. You can also arrange those visualizations in a way that will depict your analysis in the best way.

## **Dashboards**

Power BI allows you to use dashboards, and these are single-page interfaces. Most people call dashboards canvases since these will include the visualizations that will talk about the data in the same way that a picture tells a story. Since a dashboard is limited to only one page, a well-designed dashboard will contain only the most important variables in the data set.

The visualizations that you will see on the dashboard are termed as tiles. These tiles link the reports to the dashboard. Tiles, in Power BI, will always have a single visualization that is found in a dashboard or report. A tile is a rectangular box that will hold each visualization. Power BI will also give you the liberty to arrange or move tiles. This will make it easier for you to exhibit the data and variables in the exact way you want. This is especially true if you are building a dashboard or report. You can also adjust the height and width of the tile or rearrange them if needed.

## **Cost of Using the Tool**

Power BI has a free and professional version, and you can use either of these to analyze different data sets. Both the professional and free versions allow you to create a dashboard and share it on the cloud so users can access the data set from various platforms. This means that one can access the dashboard on numerous devices, including mobile devices. You can also publish your analysis on the web and import data from different files. Let us take a look at the differences between these versions. The basic difference is the limit on the data capacity. The free version is only 1GB per user, but the professional version will allow users 10GB per user. The cost of the professional version is \$9.99.

## **Installation of Power BI**

You need to select the “Advanced download options” if you want to obtain the details of the installation files or the system requirements. Let us look at some of the system requirements that you need to look at if you want to download the Power BI tool.

## Supported Operating Systems

- Microsoft Power BI Desktop requires Internet Explorer 9 or higher
- Microsoft Power BI Desktop is available on both the 64-bit and 32-bit platform
- Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, Windows 7, Windows 8, Windows 8.1 and Windows 10

A user can also select the language in which they will want to install Power BI. You can then download the following files.

You can download the different Power BI files from the following link: <https://www.microsoft.com/en-us/download/details.aspx?id=45331> .

You will see that the file will show you a 64-bit OS file. You should select the file that you want to install based on the operating system and then click next. Now, save the installation file on the local drive.

Now, you can run the installation file.

Now, accept the license agreement. You will need to follow all the instructions that are present on the screen and finish the installation process.

Once Power BI is installed in the system, it will launch a welcome screen. This will enable you to launch and use the different options on Power BI to get data, use the existing data models, create a report, publish the report, and share it.

## **Chapter Two: Why Should You Use Power BI?**

For businesses, Microsoft Excel has always been the best reporting tool, but Power BI provides more powerful reporting and analytical features. You get much faster visualization experimentation, and the ability to do calculations and statistical functions across much broader datasets, not to mention recombining fields to get the answers you need on the fly all that adds up to Power BI being a far more powerful tool with better business insights than Excel is.

To give you an idea, let's compare Excel and Power BI.

### **Access and Store a Vast Amount of Data (with No Issue)**

Power BI makes it easier for you to compress large volumes of data and files. This will make it easier for you to work with large data sets. This is not possible to do if you use Microsoft Excel. Power BI will make it easier for you to collect, view, analyze, interpret, and visualize large volumes of data. You can open a 300MB file with ease when you use Power BI. All you need to do is convert the file into the .PBIX format which will allow you to work with large volumes of data. You also do not need to cut the large data sets into smaller sizes or aggregated. Power BI allows you to look at summarizations and granular details in the data. You can do this by accessing drilldowns. We will look at this feature in detail in the last chapter of the book. Power BI also enables you to modify and prepare data:

- You can edit and transform data through changes in format, removing rows, transposing data and the addition of columns

- If you use a snowflake or star schema, you can use different relationships between the data sets to obtain a granular understanding of data
- When you discover additional data, you can easily add this data to the table. You do not have to recombine the data since you can integrate numerous data sets on Power BI with ease. Since you can integrate these data sets with ease, you have the flexibility to work with large data sets.

When you use Power BI, you can ensure that the software will remember the steps that you follow to prepare the data. You can simply repeat these steps if there are changes made to the data. This helps to reduce and eliminate the repetition of data preparation.

## **Spot Data Trends Quickly and Easily**

You can spot different trends in the data set that you import into Power BI. You will only need to look at the data set for a few minutes before you use the DAX functions and other features in Power BI. You can look at the data set through different attributes and dimensions. You can also include date-time dimensions .

If there is no schema structure, which means that the data does not have a date, you can create this easily. This will allow you to use different time intelligence functions. You no longer have to work with different macros across large files. You only need to identify the right functions to use.

## **How to Create a Time Trend Analysis Using Power BI**

1. You will first need to create a date dimension table. To do this, you should use the function “New Table” in the Modeling tab in the Power BI desktop. You can set the table as Date = calendarauto(). When you

have the table with the dates from the data set, you can create the date table. You can do this only if you use the DirectQuery option.

2. Now, you need to establish a relationship between the variables in the data and the data table. You will need to do this using a query, and you can develop one using the Query Editor. All you need to do is drag the fields from the tables and establish the relationship between the data variables. Remember, the date dimension will connect different data sets.
3. You can also go back to the report tab to help you create a visual that will allow you to place the date on the X-Axis and the measure, like sales or revenue on the Y-Axis, which depicts the value.
4. You can use additional measures using time intelligence functions like Power BI. These have been detailed later in the book. When you click on the table, you can select a new measure and enter it in the following format: YTD\_Measure = TOTALYTD (sum(Table\_Name[ Value]),Dim\_Date[ Date])



## **Intuitive UX Features**

Power BI also offers UX skills and options that will make the visualizations neat and visually appealing. If you have used the Microsoft Office suite, you will easily be able to use this feature since you only need to use the drag and drop functionality.

When you use Power BI themes, it will be easier for you to create a consistent color scheme. If you work for an organization, you can create a branding theme based on the rules of the branding department. This format will be stored in a .json file. Once you develop these formats, you can easily import the theme into your dashboard. From here on, you can create the necessary charts and graphs.

Themes are simply .json files. These files contain numerous colors that are saved in the HEX format, and you can create these formats and modify them using a notepad file. This file can also be imported using Theme tabs in the Home Ribbon in the Power BI service. You can use the following information to learn more about how you can format the data: <https://docs.microsoft.com/en-us/power-bi/desktop-report-themes> .

The Format Painter option in Microsoft Power BI will allow you to copy the formatting across different visualizations. This will make it easier for you to format the changes to individual visuals. You can also turn markers on inline charts, place the data labels on the right axis in the chart, give people access to create some custom format changes, and more. This will make the chart visually appealing.

## **Incredible Cloud-Based Features**

Now that you have created the dashboard, you need to publish it onto the cloud service. You can also publish the reports that you create onto the service. You can do this after you finish your analysis. When you use Microsoft Excel,

you will need to prepare the file, move it to SharePoint, and share the link to the file on the cloud via email. This process has been changed with the introduction of Power BI.

The data set is now on the Power BI service, which is an online cloud service. It is here that the data can be refreshed easily. You can automate the refreshing of the data set with ease. All you need to do is publish the data set onto the cloud using the publish button.

Power BI not only allows for an easy publication and distribution of data but also gives you different tools and methods that you can use to publish the data and reports you have created to the cloud.

## **Quick Insights**

Power BI offers you an easy way to gather some insights about the data that you are looking at. You can use the quick insights option to do this. You can also obtain a thorough analysis of the data set you have without opening the data set in the application. You can also create a dashboard by clicking on the data set on the Power BI Service. A dialog window will open up where you can see the option called quick insights. Power BI will automatically create the model and also establish a relationship between the variables in the data set.

## **Natural Language Query**

This is an incredible feature that will allow you to type a question and answer. You can either use a user-specified or a default form. This feature is useful for any business user who is unfamiliar with how they can use Power BI. You can also use this to identify the right tools you can use to represent data. This feature will also save you from creating numerous visualizations on the dashboard to answer a variety of questions. If you want to access this feature, you will need to type the question in the box and then use the right tools to visualize the data.

## **Personalized Dashboards**

When you use Power BI, you can create a dashboard that will allow you to develop different visualizations from numerous reports. You can also use a whole report if you want to. When you use these reports, you can customize the layout of the page and the visuals as per your needs. You can lay the dashboard in a way that will allow you to understand the data better. You can also connect the dashboard to a different data source. You can share these dashboards with different users in the organization, allowing them to view these visualizations. If you want to improve a dashboard, check the tips mentioned in the last chapter in the book.

## **Alerts**

When you have created a dashboard, you can set up an email alert in the KPI on the online service. When you right-click on the dashboard, a dialog window will come up on the screen where you can choose to manage alerts. This will allow you to set up an alert for when there are any changes in the data. This is a useful tool for an employee since it will allow them to track a specific change in the data. For example, if you check on inventory data, you can set up an alert that will tell you when the inventory level increases or decreases.

## **Row Level Security Features**

It is easy for users to add row-level security features in Power BI, and you can do this easily even if you do not have coding experience. This was a complex feature in Microsoft Excel. For example, an employee who uses RLS can look at data that is specific to the local geography. You can always add these filters to the report. This also makes it easier to protect the data used in the report from being mailed to another user. When you want to set up the RLS, you should filter for various roles that you can set up within Power BI. You can add the remaining group or individual filters to the report using the Power BI online service.

You will need to select the option “Manage Roles” in the Modeling tab. You can find this option in the security tab. You can choose the groups and create a role for each of these groups using a DAX filter function (covered later in the book). For instance, if you want to look at the data only for a specific country, you should set the filter on the country to the country you are looking at.

Power BI makes it easier for organizations to function in a data-driven world. A business can no longer focus only on IT to collect, collate, clean, transform, and analyze data. They will need to have some powerful self-service abilities. Organizations need to use Power BI so they can integrate different data sets, expand into using newer data sources, visualize data easily, and deploy numerous RLS filters. It is also easy to govern how the data is shared or used since the user can determine how they want other people to view the data when they share it on the cloud. This can be done through different features, including the Natural Language Query. It is important that analytics companies and teams always work with the data in the right way and also intelligently collect the data. This is the only way they can

evolve with the changes in consumer habits and business needs.

Since the introduction of numerous data visualization tools, companies are doing their best to incorporate various visualization methods and tools into their analysis. You can also incorporate this tool for your organization.



# Chapter Three: Power BI Data Sources

As mentioned earlier, Power BI does support numerous data sources. You can simply use the Get Data option on Power BI, and this will give you a list of sources and data connections that are available to you. You can connect to different SQL databases, flat files, cloud services like Azure, or even social media platforms like Google Analytics, Salesforce objects, and Facebook. These data sources also include an ODBC connection that will allow you to connect to other data sources. These sources are not listed in this chapter.

Some of these data sources are:

- Azure Cloud platform
- SQL Database
- Flat Files
- Blank Query
- OData Feed
- Blank Query
- Online Services
- Other data sources such as Hadoop, Exchange, or Active Directory

If you want to move data into the Power BI desktop, you can choose the Get Data option on the home screen of the Power BI desktop. This will show you the common data sources that people use to import data. You can click on 'More Options' to view the full list of the data sources available.

When you obtain this screen on your desktop, click on the "More..." option. This will send you to a new window. On the left side of the window, you will see a category of all the

data sources that you can use to obtain the data from different sources. Alternatively, you can perform a search to obtain these data sources.

You will see the following data sources listed:

## **All**

In this category, you will look at the different options and data sources that are available for you to use. This will list the sources present in the Power BI desktop.

## **File**

When you click on this option, you will find a list of all the file types that the Power BI desktop will support. If you want to connect to a specific data type, all you need to do is select a file type from the list and then choose 'Connect.' You only need to provide the source of the file .

## **Database**

When you click on this option, it will show you a list of the database connections that Power BI will allow you to connect to.

If you want to connect to a database, you should select the database type from the list. Now, click on connect. You should pass the name of the server, username, and password if you want to connect to the database. Alternatively, you can connect to the database directly using an SQL query through the Advance options. You can also use the DirectQuery or Import option. Remember, you cannot combine these two options in one report.

## **Import Option Versus DirectQuery Option**

The DirectQuery option will limit the option of manipulating the data. Another point to keep in mind is that the data will only stay in the SQL database. Since this option is a live option, you do not need to schedule the refresh option,

unlike the Import method. The Import method, on the other hand, will allow you to perform both data manipulation and transformation. When you publish this data to the Power BI service, the limit of data you can transfer will be 1GB only. This option will consume the data and push it directly into the Power BI Azure backend. You can also refresh the data at least eight times a day. Since you need to refresh the option every time you open the query, you can schedule the refresh.

## **Advantages of Using DirectQuery**

- DirectQuery will allow you to work with large data sets and also build a data visualization. This is not a feasible option if you use the Power BI desktop
- The DirectQuery will not set a limit on the data that you can import
- You can always show current data when you use this option

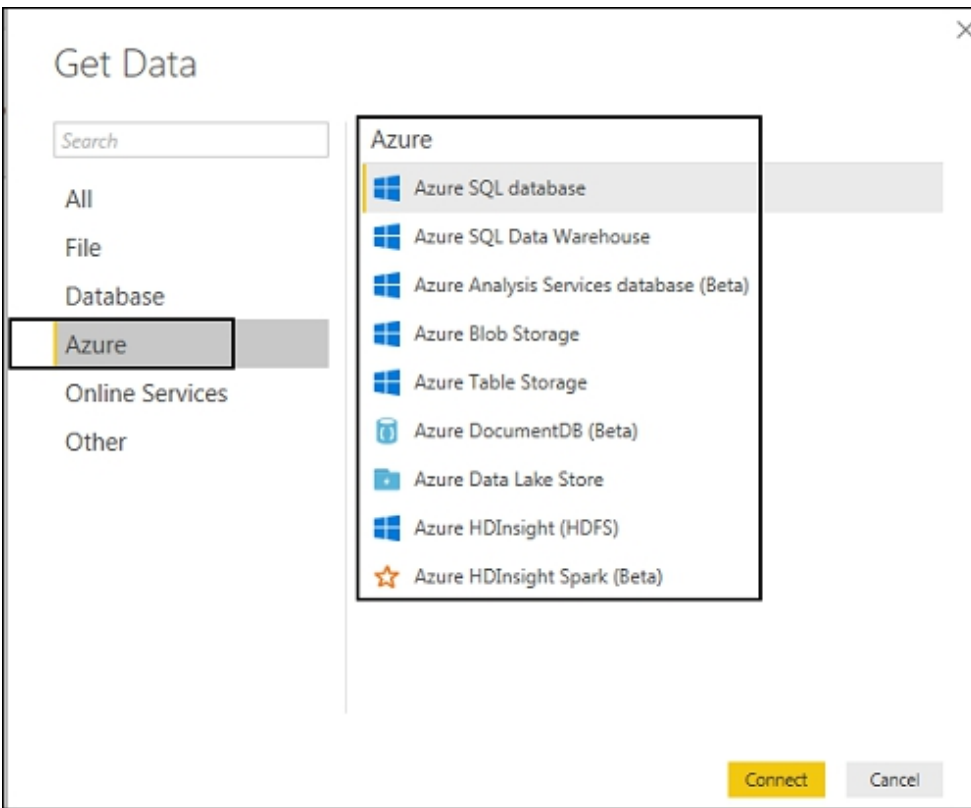
## **Limitations of Using DirectQuery**

- DirectQuery sets a limitation of only 1 million rows to return any data. You can also perform an aggregation of different rows. You must, however, ensure that the result rows will not be more than 1 million
- Every table you use should only come from one database
- If you use a complex query in the editor, you will receive an error. If you want to run the query, you must remove the error from that query
- You can only filter using the relationship option in one direction
- You cannot perform any queries on time-related tables

## **Azure**

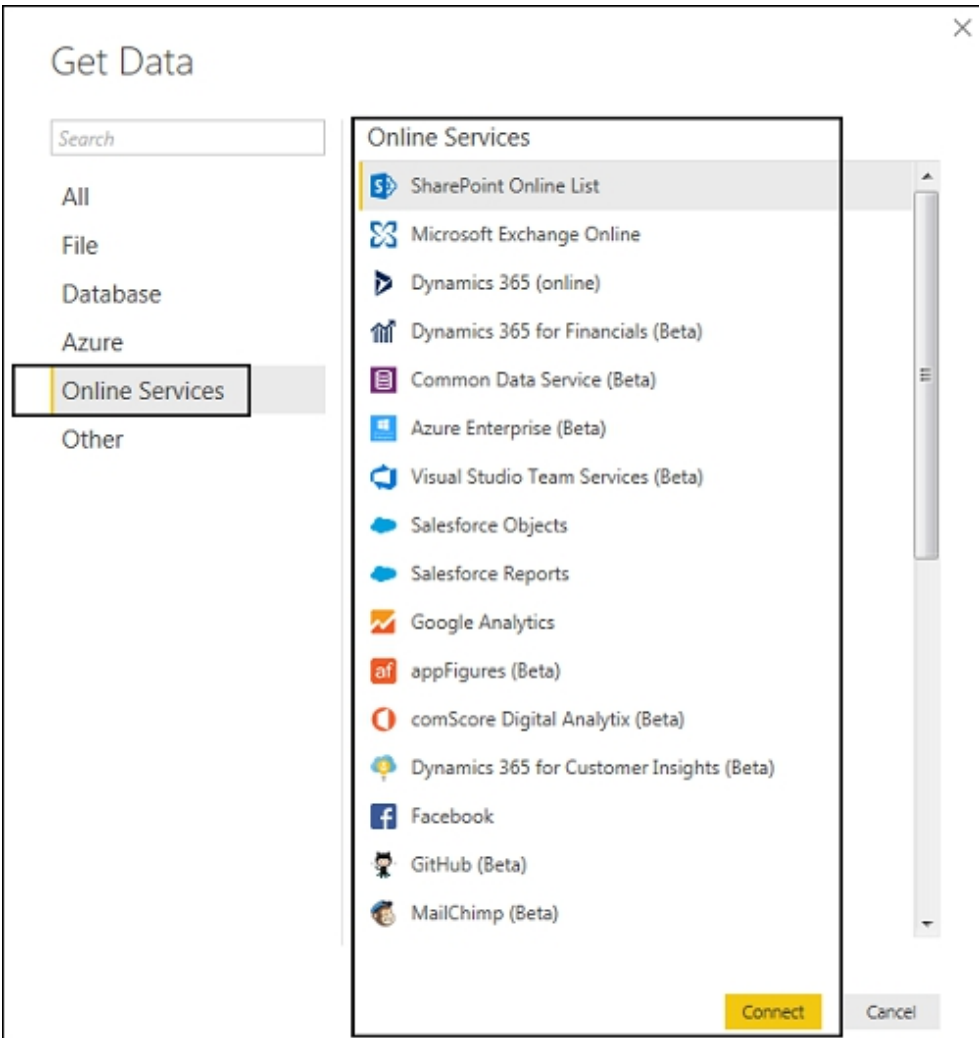
As mentioned earlier, Power BI allows you to connect the data to a cloud service. The Azure option will allow you to

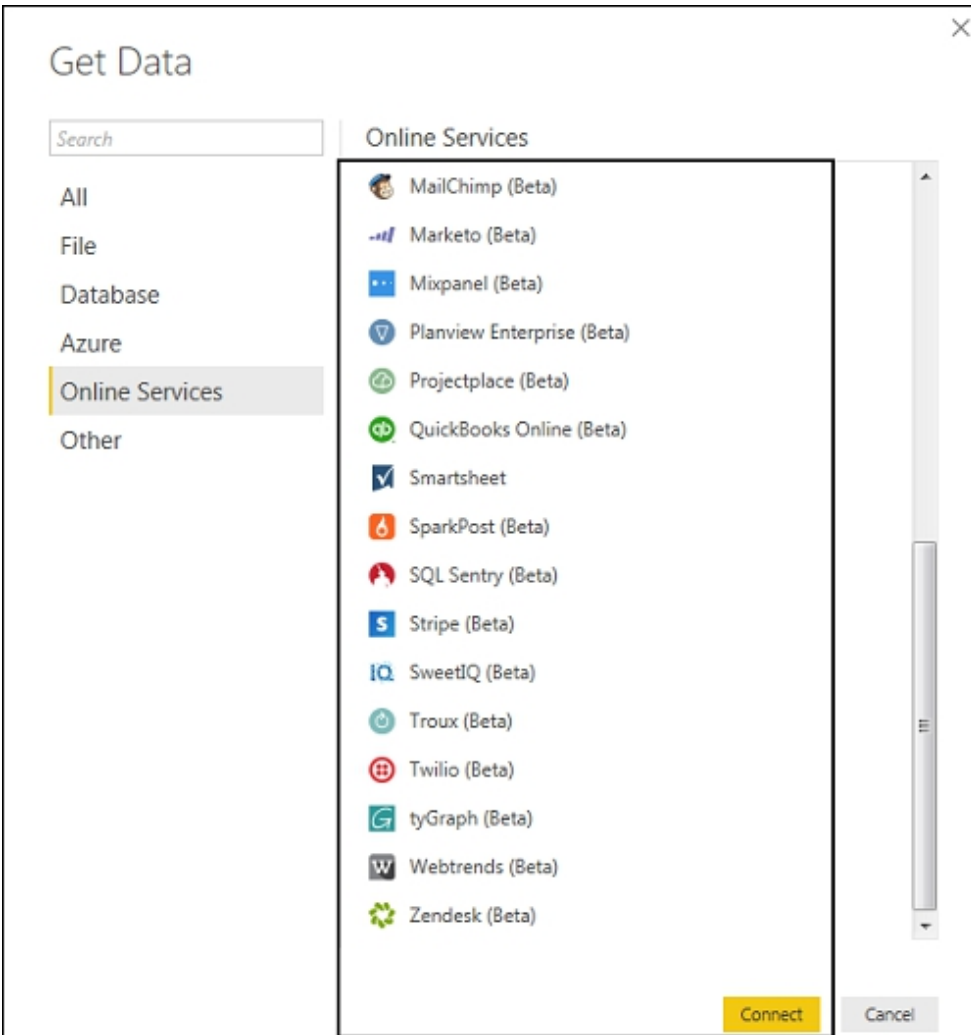
connect the database to the Azure cloud. The screenshot below will show you the different options that you can choose from this category.



## Online Services

You can connect to different online services like Google Analytics, Facebook, Exchange, and Salesforce to obtain data. The screenshot below shows you the different options available under this category.





## Other

The screenshot below will show you the different options under this category.

# Get Data

Search

- All
- File
- Database
- Azure
- Online Services
- Other**

## Other

- Web**
- SharePoint list
- OData Feed
- Active Directory
- Microsoft Exchange
- Hadoop File (HDFS)
- Spark (Beta)
- R script
- ODBC
- OLE DB
- Blank Query

**Connect** Cancel

# Chapter Four: Comparison of Power BI and Other Tools

This chapter will compare Power BI with other tools like SPSS and Tableau.

## Power BI Versus Tableau

Most businesses in the business intelligence market use Tableau as a leading data visualization tool. Power BI is an emerging tool and is giving Tableau strong competition. Power BI allows you to connect with numerous data sources and allows you to manipulate data within the application itself. It is for this reason that this data visualization tool is gaining popularity. Having said that, medium and large enterprises still continue to use Tableau as their data visualization tool. As mentioned earlier, Power BI is integrated with the Office 365 suite comma and is compatible with numerous sources like Microsoft Excel, SharePoint, etc.

Feature	Tableau	Power BI
Data Visualization	As mentioned earlier, Tableau is one of the leading data visualization tools in the market.	Power BI, on the other hand, provides access to simple visualizations and allows you to manipulate data at the back end.
Size of Dataset	Tableau allows you to connect with large volumes of data whenever necessary.	Power BI limits the number of records you can include in the data set and also sets a limit of 1GB on the amount of data you can import into the system if you use the free version.
Data Sources	Tableau, like Power BI, allows you to connect to numerous data sources to import data for your analysis. Having said that, in Tableau, you first need to select the data set before you visualize the variables to perform an analysis.	Since Power BI is integrated with Office 365, it also provides access to cloud services like SharePoint and Azure. Power BI also provides connectivity to



		<p>various data sources that one can connect within Tableau. The online version of Power BI also supports visualization on cloud-based applications like Azure and SharePoint. You can also create direct visualizations on search engines, but Bing is the only Search Engine that is compatible with this tool.</p>
Cost	<p>Unlike Power BI, Tableau is very expensive. Small and medium enterprises still find it difficult to purchase this software.</p>	<p>As mentioned earlier, Power BI also provides a free version. The only issue with that version is that you can import 1GB of data into the tool to perform your analysis. The professional version is a cheaper solution when compared to the tableau.</p>
License and Pricing	<p>Tableau Desktop Profession: USD70/user/month. This version allows you to connect to numerous data sources.</p> <p>Tableau Desktop Personal: USD35/user/month. This version allows you to connect to web-based applications like Google Sheets and Excel files.</p> <p>Tableau Server: Minimum ten users with the cost of USD35/user/month</p>	<p>Power BI: Free</p> <p><b>1 GB storage</b></p> <p>10k rows/hour data streaming</p> <p>Power BI Pro: USD9.99/user/month</p>

	Tableau Online with private cloud: USD 42/user/month	<b>10 GB storage</b>  1 million rows/hour
Implementation	The implementation of tableau will vary based on the needs of the organization. The installation can take a few weeks or a couple of hours.	Power BI includes a very simple implementation process, and it uses cloud storage.

**Power BI Versus SSRS**

# **Chapter Five: Data Modeling in Power BI**

This chapter will shed some light on how you can use data modeling in Power BI.

## Using Navigation and Data Modeling

This is one of the most popular features in Power BI, and you can use this feature to connect to numerous data sources using a relationship. You can use this relationship to define how the different data sources can be connected to each other. You can also create data visualizations on numerous data sources. When you use this modeling feature, you should build a custom calculation on the existing data sets. This means that you can perform numerous calculations on the existing columns and tables. You can present this directly in the Power BI data visualization section of the tool. You can also define new metrics and custom calculations to calculate those metrics for visualization.

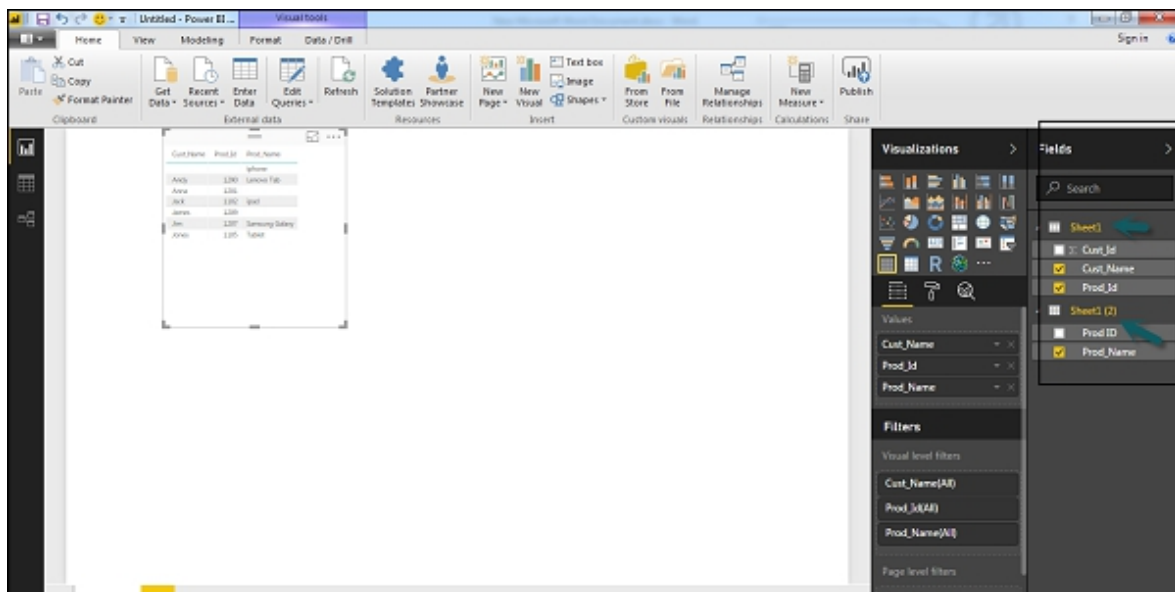
The screenshot displays the Power BI data modeling interface. Two tables are shown: 'salesbycustomer' and 'salesbycustomer\_products'. A relationship line connects the 'id' column of 'salesbycustomer' to the 'id' column of 'salesbycustomer\_products', labeled 'id = id'. The 'salesbycustomer' table has columns: (Add All Columns), \*(Wildcard), id (checked), CustomerDetails\_customerFirstName, and CustomerDetails\_customerLastName. The 'salesbycustomer\_products' table has columns: (Add All Columns), \*(Wildcard), id, ProductName (checked), AmountSold, and QuantitySold (checked). Below the tables is a summary table with the following data:

Field	id	CustomerData...	ProductName	QuantitySold
Table	salesbycustomer	salesbycustomer	salesbycustom...	salesbycustom...
Database	localCassandra	localCassandra	localCassandra	localCassandra
Aggregate Function				
Where Condition		...		

In the image above, we are looking at a common data model. In this model, you can see that there is a relationship between two tables. These tables can be combined together using the primary column name "ID." In the same way, you can assign a relationship between two objects in the data set. You can set this relationship by dragging a line between the columns that are common to both tables. You can view

these relationships using a data model in Power BI. Before you create a data model in Power BI, you will need to choose the data source that you want to use to perform the analysis. You can do this using the new report option. If you want to add any data source, you should use the “Get Data” option, and then select the source from where you want to obtain the data. Now, click on the “Connect” button.

Once you choose the data source and add it to your application, you will see it on the right side of the screen. In the image below, you can see that we have used two Excel files - “Customer” and “Product” - to import the data into Power BI.



On the left side of your screen, you will see the following tabs:

- Report
- Data
- Relationships

If you navigate to the Report tab, you will see the chart that you selected for the analysis and the dashboard. This is what you will need to use for data visualization. You can also select different chart types or other visualization tools as

per your analysis. In the example used in this chapter, we have chosen the method “Table” from the “Visualizations” tab.

You would have created a relationship between the data sets when you imported the data source. When you navigate to the Data Tab, you will see the different data sources based on the defined relationships that you have set.

In the relationship tab, you will see that there is a relationship between the tables from the data sources. If you add numerous sources to the visualization in Power BI, you can define the relationship between the different tables using a simple relationship by establishing a link between the columns. When you move to the Relationship tab, you will see the relationship that exists between the columns in the tables. You can use the option “Create Relationships” to create a relationship between various columns in the tables.

You can also add various relationships and remove them if needed in the data visualization tab in the application. If you want to remove any relationship, you should right-click on the relationship and choose the “Delete” option. If you now want to create a new relationship, you need to drag the fields and drop them on the tabs that you want to link.

You can also hide a specific column in the report that you create using the “Relationship” view. If you want to hide a column, all you need to do is right click on that column and then choose the option “Hide in report view.”

## Creating Calculated Columns

When you analyze data, you may need to combine two columns to understand how the data changes or moves. You can create a calculated column in Power BI by combining elements in the existing data set. You can also define one or more new columns in the data set by applying a calculation on a column that is present in the data set. You can also create calculated columns to establish the relationship between different tables sourced from the data sources. You can also use this relationship to establish a relationship between these tables. If you want to create a new calculated column, move to the “Data View” tab on your Power BI screen and then select “Modeling.”

On the Power BI screen, you should navigate to the Modeling tab. In this tab, you will see a “New Column” option. This will also open the Formula Bar, and you can enter any DAX formula you need to use for this calculation. We will learn more about these DAX functions later in the book. DAX refers to Data Analysis Expression language that is powerful to use. You can rename these columns by changing the Column Text in the formula bar.

Let us use a DAX function to create a new column in this data set. The column is the Product Code (Product\_C). This table is created based on the last three characters in the column Prod\_Id. You can write the following formula: `Product_C = RIGHT( Sheet1[Prod_Id],3)`.

You can also choose from a long list of formulae provided in the application. You can use these formulae to create a calculated column. For this, you need to enter the first character in the formula so you can use it in the formula. Look at the screenshot below to understand this better.

Relationships Calculations Sort Formatting Properties

Product\_C = R Sheet1[Prod\_Id],3)

Cust_Id	Cust_Name	Pi
101	Jack	
102	Jones	
111	Anna	
125	Jim	
135	James	
137	Andy	

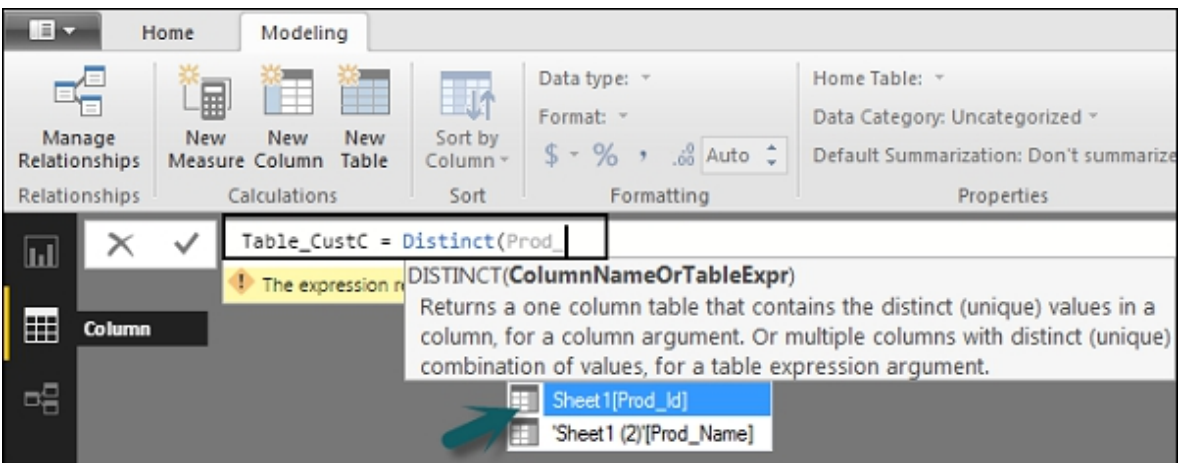
- RADIANS (Converts degrees to radians.)
- RAND
- RANDBETWEEN
- RANK.EQ
- RANKX
- RELATED
- RELATEDTABLE
- REPLACE
- REPT
- RIGHT
- ROUND



## Creating Calculated Tables

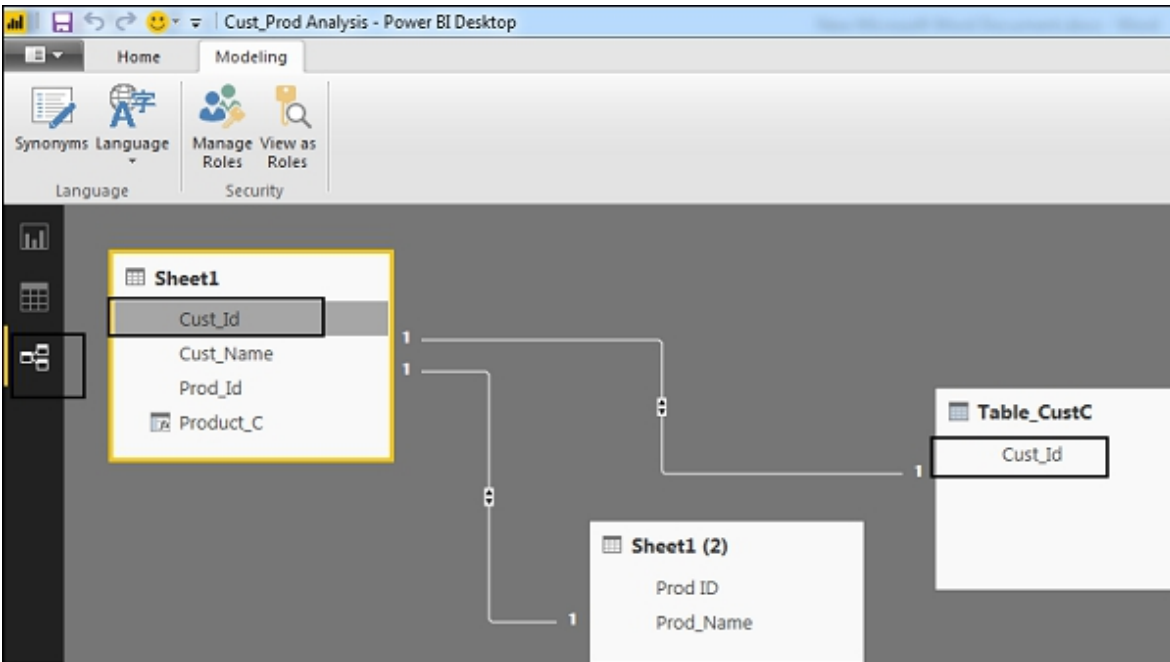
Power BI allows you to create a new calculated table using the data modeling tab. If you want to create new tables, you should navigate to the Data View tab on the left side of the Power BI screen. You can then move to the Modeling option that is present at the top of the Power BI screen.

You can use the DAX expression to create new tables. You should also enter the name of the new table on the Power BI screen. You need to use the same format of formulae that you will use in Microsoft Excel. All you need to do is enter the name of the table on the left side of the “=” sign. You should then include the required DAX formula to perform the required calculation. When you finish writing this calculation, you can create a new table that will appear on your field pane. In the example below, let us now define a new table, “Table\_CustC,” which will return the column that contains a unique value.



You will find a new table under the “Fields” section on the screen. This is shown in the screenshot below. When you create the calculated table and column as per the requirement, you will see the fields in the report tab in the Power BI application. If you want to add these objects, you will need to select the right checkbox. You will then see the relationship between these objects. Otherwise, you can drag

between the columns that you want to build a relationship between.



If you want to view these reports correctly, you should navigate to the Report tab in the Power BI tab. You can then see the “Calculated Columns” and the “Calculated Tables” options in the report view of the Power BI.

## Managing Time-Based Data

Power BI also allows you to drill through data based on time. If you add the data field that you need to look at to the analysis, you can enable this drill version in your data visualization. This will take you to another level in the time-based data. Let us now consider the column “Time-based Table” that we have added to this visualization. From the screenshot below, you can see that we have added the tables “Year” and “Revenue.”

D7	:	X	✓	<i>fx</i>						
	A	B	C	D	E	F	G	H	I	
1	Prod ID	Year	Quarter	Revenue						
2	1101	2017	1	10000						
3	1102	2016	1	25000						
4	1105	2017	2	15000						
5	1207	2016	3	20000						
6	1290	2016	4	14000						

You can also enable this drill feature in any visualization using the option at the top of the Power BI screen. Once you enable this drill feature and click on the lines or bars in this chart, you can drill it down to the next level or feature in the time hierarchy. For example, you can drill down from years to quarters to months. You can also use this to move to the next level in the option and perform a deeper drill.

# **Chapter Six: Power BI Dashboard Options**

In this chapter, we will look at the different dashboard options that you can use in the Power BI Application.

## **Exploring Different Datasets**

The Power BI tool will provide you with numerous options that you can use to explore the data set. When you work on a dashboard on Power BI or a report, you can use the Power BI Look to obtain some quick insights about the data set. You should navigate to the data sets section in the User interface. Click on the three dots and choose the option. Get insights for the same.

When you choose this option, the Power BI user interface will run all the algorithms in your data set. Only when this is complete Power BI will send you a notification that will show you that your data set is ready for analysis.

You can then choose the “View Insights” option to obtain the necessary tools. This will give you a representation of the data in the form of charts or bars. You can view this option at any point during your analysis of the data set. You will also get the option to obtain a Quick Insight about your data set when you publish the report in Power BI .

## **Creating Dashboards**

Power BI allows you to create a dashboard where you can pin the visualizations that you obtain from the Power BI report that you publish on the Power BI desktop. You can create these visualizations using the Power BI service. You can pin these visualizations to your dashboard as well. If you want to pin a visual to your dashboard in Power BI, you

should open the report in the Power BI service. You can then select the pin icon that is present at the top of the visual.

When you choose the pin option, you will see a dialog box appear on your screen. This dialog box will appear in the following manner. You will be asked if you want to create a new dashboard or if you want to use an existing dashboard to put the visual on your screen. You can choose the visual from the dropdown list. If you do not have any existing dashboards, Power BI will grey out the second option.

When you choose the pin button, Power BI will throw you a confirmation that you have pinned your visualization to the dashboard. You can then click on the option “My Workspace” to check the dashboard and view the visualization. When the dashboard is created, you can use numerous options to configure or redesign that dashboard.

## **Sharing Dashboards**

When you publish a report in Power BI using the Power BI Service, you can either share the dashboards or reports with other users either within or outside the organization. This makes it easier for you to share the dashboard in Power BI. You can then open the dashboard using the Power BI Service and also click on the Share option that is present at the top of the screen.

Having said that, you only have the sharing feature if you use the Professional version. If you want to see how this works, you can use the 60-days trial.

All you need to do is click on the Try Pro for free to begin this trial version. Now, select the Start Trial and follow the steps to click on Finish. You will then receive a confirmation that the trial period has started. Now, enter the email ID of the person with whom you wish to share this dashboard. You can also decide if you want to allow these users to send email notifications about the changes made to this

dashboard or share this dashboard with other users. You can also provide the URL to these users and access the dashboard directly.

## **Tiles in Dashboards**

When you choose the more option in the dashboard, you will see the option “Focus Mode” and many other options in this dashboard. Apart from this option, you will also see other options on your screen. The focus mode will allow you to take a close look at the data present in your dashboard. When you have numerous values in the dashboard, you can use this mode to obtain a better view of the objects present in this dashboard. If there are numerous columns in this database, Power BI may not show you those tables. You can view these tables using this Focus Mode .

The focus mode is used to view all the data in the report and dashboard. You can also pin this visual directly onto the dashboard and other dashboards using the Focus Mode. You can do this by selecting the pin icon. If you want to move out of the focus mode, you should select the option “Exit Focus Mode.”

The Tile Details option will help you edit the format of the tiles on the dashboard. This option will let you change the title, subtitle, the last refresh date and time, and some other details about the tiles. This will allow you to create a custom link to your dashboard.

## **Data Gateway**

As mentioned earlier, you can connect to various on-premises data sources to the Power BI service. This can be done using the data gateway option. You can use the Personal Gateway, which will not include any administration configuration. This is another version of the data gateway. You can set this gateway up by logging into the service offered by Power BI. You can also select the download icon

on the top corner of the screen. You can do this by clicking on the Data Gateway.

When you use on-premises data for your analysis, you can always refresh the data if there are any changes made to the data. You also do not need to move this data from the Power BI service. You can source the data using the Data Gateway option. You can query a large volume of data using Power BI service. These gateways will make it easier for you to provide all the flexibility you need to meet any individual or organizational needs .

If you want to set up the download gateways, you will need to run the setup until it is fully downloaded, and the installation process is complete.

You can choose from the following options:

- Personal gateway (Power BI only)
- On-premise data gateway

When you launch the gateway, you will need to log into that service. You can also enable Power BI to send you automatic updates or schedule the refresh if needed.

# **Chapter Seven: Power BI Visualization Options**

Let us now look at the different visualization options in Power BI.

## **Creating Simple Visualizations**

You can create a visualization to present your data effectively. Visualizations are the basic building blocks of any business intelligence tool used by organizations. Power BI contains numerous default visualization components that you can use. These components include pie charts, maps, bar charts, graphs, and some complex models like funnels, gauges, waterfalls, and other components.

Power BI allows you to create a visualization using two methods. The first method is to add the visualization from the report canvas present on the right-side pane of the user interface. The default visualization is a table type visualization that is selected by Power BI. The second method is to drag the necessary fields from the table on the right side of the bar to the axis. You can then include multiple fields against every axis as per the business requirement.

You can also move your visualization to the reporting canvas in Power BI. To do this, you need to click on the reporting canvas and then drag it onto the screen. You can also switch between different types of visualization using the visualization pain. Power BI does its best to visualize any data that you import into the tool.

## **Creating Map Visualizations**

There are two types of map visualizations in Power BI, namely shape maps and bubble maps. If you want to create



the latter, you should select the map option present in the visualization pane.

If you want to use the bubble map, you should drag the map to the report canvas from the visualization pane. If you want to display the values for your analysis, you need to add the location object to the axis.

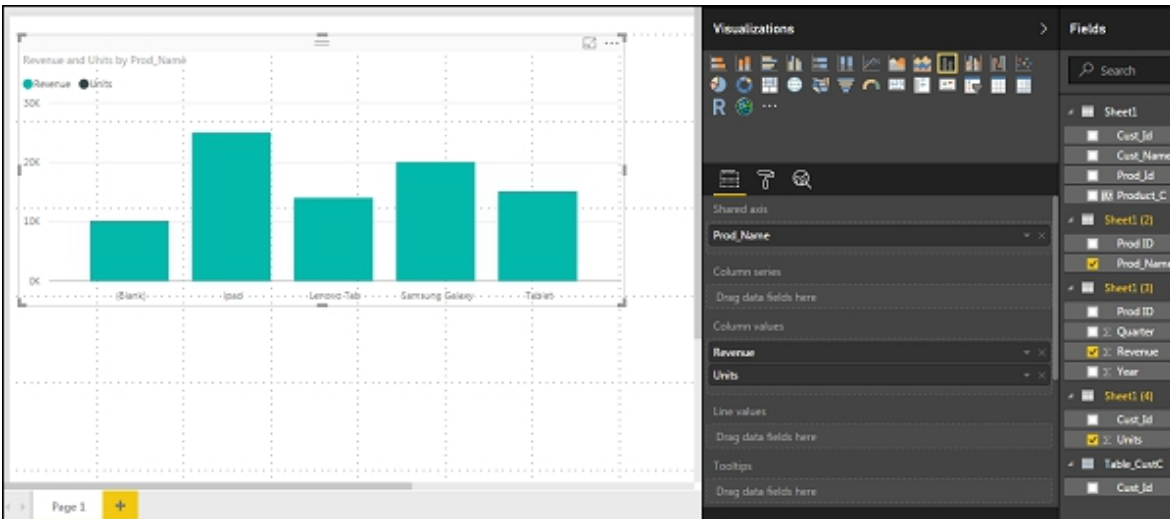
In the value field, you will see that the map allows you to include values like a state or city. You can also include values like latitude and longitude. If you want to change the size of the bubble, you need to add a field to the value axis. If you want, you can use a filled map for your visualization. To use a filled map, drag it to the report canvas from the visualization pane.

If you see a warning symbol at the top of the map visualization, this means that you need to include more locations to the chart.

## **Using Combination Charts**

At times, you will need to plot multiple measures in one map or chart. Power BI allows you to combine different chart types to plot numerous values for analysis. Let us assume that you want to plot the unit sold and the revenue on one chart. It is a good idea to use a combination chart if you want to meet this requirement. One of the most common combination charts that are used by analysts in Power BI is the Line and Stacked column chart. Let us assume that you have a revenue field, and you have included a new data source that has information about the number of units purchased by each customer. Let us now plot this information in the chart.

Add the data source; you will see that it is added to the list of fields on the right side of your user interface. You can then add the unit soul to the column axis, as seen below.



You can also use other types of combination charts like the line and clustered column chart.

## Using Tables

When you add date-assist your visualization and Power BI, you will see that a table chart has automatically been added to the report canvas. As mentioned earlier, the table visualization is the default visualization chart used by Power BI. You can always drag those fields that you want to use for your analysis to the table. Alternatively, you can also include these fields to the table simply by clicking on the checkbox present against the field names. If you have any numerical values in the table, you will see the sum of these values at the bottom of the column.

You are also allowed to start the data in your table using the sort option that doesn't appear in the interface. This option is present at the top of the column in the form of an arrow key. If you want to perform the ascending or descending sort, you will just need to click on the respective arrow mark. The values in the column will automatically be sorted.

Remember that the order is columns in every table in Power BI is determined by order of the value on the right side of that column. If you want to change the order of the column,

you can either add another column or delete the column on the right side of that column.

You can summarize, apply different aggregate functions, undo, or perform other actions on numerical values present in the data table. If you want to change the aggregation type you are using, all you need to do is Navigate to the value bucket and click on the arrow in that bucket. You will then see a list of functions that you can use to change the aggregation of the values in the data table.

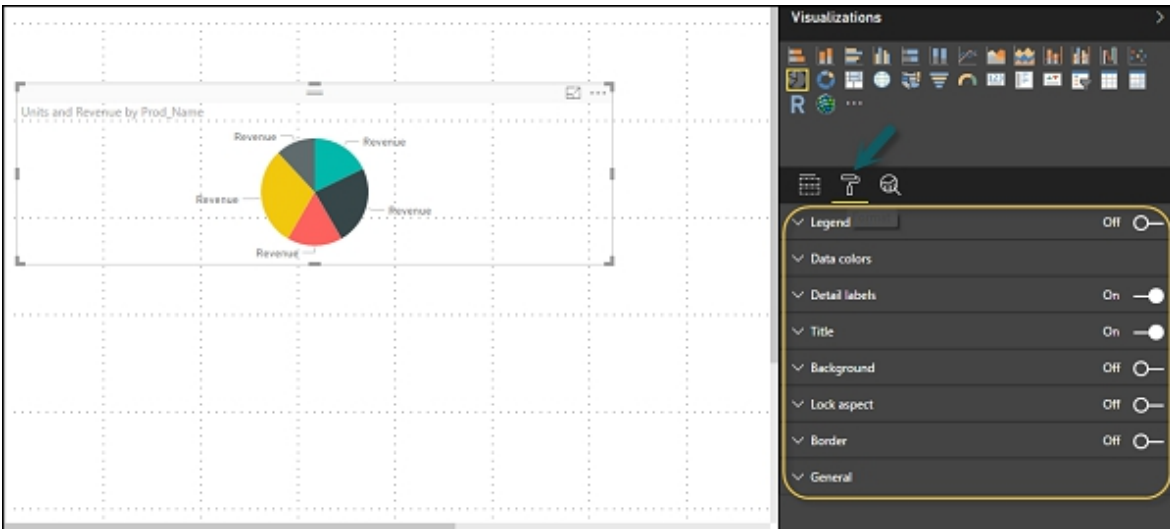
You can also use a matrix table type in Power BI to perform an analysis. This table provides numerous features like column tables, setting colors, auto-sizing, etc .

## **Modify Colors in Charts**

Power BI allows you to modify various colors in charts. If you select any visualization, there are numerous ways in which you can change the color of the axis and other variables present in this chart. The following options are available in the format tab for a chart:

- Data Colors
- Legend
- General
- Detail Label
- Background
- Border
- Lock Aspect
- Title

You can view all these options in the Format tab. The screenshot below shows you where you can find all the options available to you.



If you expand the legend field in the above options, you will have an option where you can choose to display the legend, Ford, your visualization. You can select the following options:

- Color
- Position
- Legend Name
- Title
- Font Family
- Text Size

You also have the option to choose numerous data colors. In case you want to change the color of a column representing a data field in your table or chart, you can use this option first up. This will show you all the objects in your chart and the corresponding formatting options.

You can also use the analytics feature in the tool. This is where you can draw lines or the analysis as per the requirement. You can use the following line types in this tab:

- Min Line
- Constant Line
- Percentile Line
- Median Line
- Max Line

- Average Line

Alternatively, you can also choose a dotted, solid, or dashed line. You can change the transparency level of the line comma, the position of the line, and the color. You can also switch off or dealing the data label for this line.

## **Adding Images, Shapes and Text Boxes**

There are times when you will need to include images, static text, shapes, and other components to the data visualization. To do this, you will need to use the header footer option. You can include static signatures or messages to the visualization depending on your requirement. You can also add a URL to the text box. Power BI will then make those links active. If you want to add images, text boxes, or shapes to the visualization, move to the home tab and choose the option to add images.

You can also include different shapes to the visualization that you have developed. If you want to look at the available shapes, you should navigate to the shapes button and click on the arrow next to it.

When you select the text box, you will see that a text box has been created on the report canvas. You can either enter some text in that box or use the rich text editor to format the box and make any changes.

You can also add images to the data visualization in a similar manner. You can add images like logos to the visualization. To do this, you need to click on the image option an enter the path to pass the file that contains the image. You can also add various shapes by selecting the shapes that are present in the drop-down list. Power BI also allows you to resize these shapes and images using various options.

## **Styling Reports**

Power BI gives you the option to adjust the page layout for stop. You can also format the layout by changing the page size and the orientation. To do this, you should move to the page view menu in the Home tab where you will find the following options:

- Actual Size
- Fit to Width
- Fit to Page

In Power BI, the default page size is 16: 9. You can, however, change the size of that page for the report. If you want to change the page size, you should move to the visualization pane in the report canvas and click on the paintbrush option. Remember, you need to change the page size before you add any visualization to the report canvas. You can choose from the following options under page layout:

- Page Size
- Page Background
- Page Information

Under the page information tab, you only have the options Q&A and name. under page size, you can choose from the following options:

- Width
- Type
- Height

Under the page background option, you can choose from the following options:

- Transparency
- Add Image
- Color

## **Duplicating Reports**

In some scenarios, you can choose to use the same layout and visualizations for various other pages. Power BI allows you to create a copy of the page with the report in it. When you use the option “Duplicate Page,” you will see that a new page has been created with this same visuals and layout. If you want to create a duplicate of any page, right-click on the page and choose the option “Duplicate Page.” This will create a page with the name duplicate of Page 1.

If you want to delete the existing page or rename it, you can use the other options that are present in the dialog box.

# Chapter Eight: How to Share Power BI Dashboards

In this chapter, we will learn how you can share a power be I dashboard printing, report sharing, publishing, etc.

## Using Power BI Desktop to Share Reports

When you create a Power BI report, you can share that report with other users only if you have created that report on a Power BI desktop. All Power BI dashboards, data, and reports can be shared with business users and colleagues in the organization. You can use the following methods to share reports with these users:

- Use content packs to combine reports, data sets and dashboards that are created in the Power BI desktop tool
- You can publish the report directly using the Power BI service
- You can use the Power BI mobile application to access any shared reports or dashboards
- You can create a group and share the dashboard, report, data and other information with this group will stop Having said that, you can assign specific rights to those users

Let us now look at how you can publish a Power BI report using the desktop tool. When the report is created in the Google Power BI desktop, you should navigate to the publish button. This button is present in the home tab on the desktop.

When you select the publish service option, the custom measures, reports, and visuals are packaged and published to the service. These files will have the extension “.pibx.” While the upload is still in process, you will receive a notification from Power BI with the status of the upload.



Once Power BI completes this upload, you will receive a confirmation message which will tell you about the success or failure of the upload. You can also obtain some quick insights about the reports or the data. To view these quick insights, open the shared report from the dialog box.

## **Printing Power BI Dashboards**

You will also need to take some printouts of your dashboards or reports. Power BI allows you to do this easily. If you want to take a printout of the report, navigate to the Power BI Service and click on the option "..."

This will open a print dialog box that you can use. You can then choose the printer on which you want to take a printer off that report. You can also choose different print options like margins, header, scale, portrait, or landscape .

## **Export Options**

Power BI will also allow you to export the report, dashboards, or the data set directly from the Power BI Report. If you want to use the export option, you should navigate to the Power BI service and choose the power bear report that you wish to export.

When you choose the export option, Power BI will generate a CSV file. You can also view or export the report directly using the print option.

## **Publishing Report to Web**

Power BI also allows you to publish the report to a cloud service or on the web. It also allows you to share this report via email. If you want to publish a report to a cloud service or web, you should navigate to the workspace in the Power BI service.

When you open the report you wish to publish, you should navigate to the publish to web option in the File tab. Once you select this option, Power BI will open a new dialog box, which will create an embedded code for this report. You can embed this code in an email or on a website. When you click on this option, a dialog box will open, which will say the following: *“Get a link or embed code that you can include on a public website. You may use publish to web functionality to share content on a publicly available website. You may not use this functionality to share content internally, which includes your email, your internal network, or intranet site. Publish a live version that will remain synchronized with the source report in Power BI. Any changes you make to the report will immediately be reflected in the published public version.”*

When you choose the option to create embed code, Power BI will prompt you to share the data with everybody on the Internet. When you select this option comma, the following message will be displayed: *“You are about to create an embed code for this report. Once published, anyone on the Internet will be able to access the report and the data it contains, and Microsoft may display the report on a public website or a public gallery.”*

Before you publish this report, you must ensure that you have the right to share this data publicly. You should never publish any proprietary or confidential information. You also cannot share anybody's personal information on the web. If you are in doubt, you should check the policies of your

organization before you publish that data set. You can also publish the report that you create in the form of a web page. You can share this link with any user. Power BI will share this link via email so that it can be used as an iframe.

## **Deleting an Embed Code**

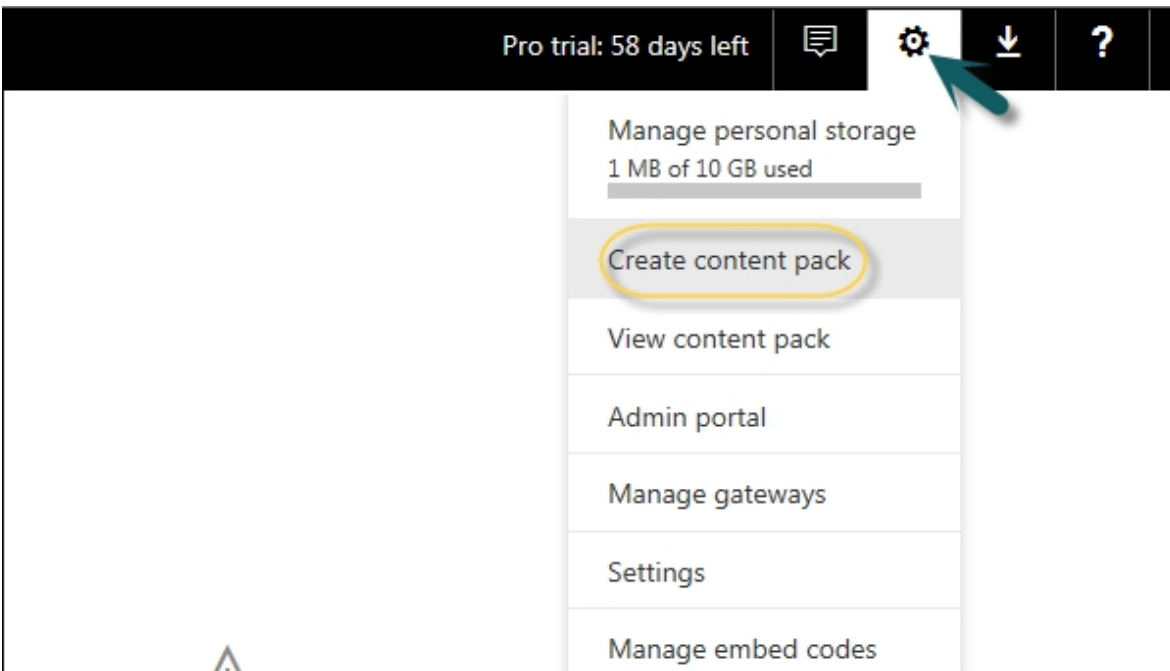
Let us assume that you want to delete the embedded code created for the rept. For this, you should navigate to the gear icon present the Top of the screen. You can then go to the option “Manage embed codes.”

If you want to remove the embedded code, click on the ellipsis mark, which is present in front of the report name. You can then delete that option .

Choose the delete option, and Power BI will ask if you want to delete the embedded code that is published on a web page. If you are sure, you can click on delete.

## Using Content Pack

You can also share reports, datasets, and dashboards in power behind the form of a package. You can share these packages with your colleagues or other users. If you want to create a content pack, navigate to the gear box in the workspace, as shown in the screenshot below.



When you choose the create content pack in the dialog box, Power BI will prompt you with a new dialog box. You can select the option to distribute this path with the entire organization or specific groups alone. If you want to share the report only with specific people, you will need to enter their email addresses will stop you can also add a description and a title of the content package that you are sharing .


If you look at the screenshot below, you will notice that you can choose the components that you want to publish. You can choose from the following options:

- Reports



- Dashboards

- Datasets



**Upload** an image or company logo  
Image size: 45 KB or less, 4:3 aspect ratio, JPG or PNG format

Select items to publish

Dashboards	Reports	Datasets
<input type="checkbox"/> Prod Analysis	<input type="checkbox"/> Cust_Prod Analysis	<input type="checkbox"/> Cust_Prod Analysis

The content pack will be available in your organization's content gallery. [Learn more](#)

**Publish** Cancel



## **Editing Content Pack**

When you create a content pack, you can also go back and edit those options that you're willing to share in that content pack. If you update a report or a dashboard, you are asked by Power BI if you want to update the objects in the shared content. To do this, navigate to the gear box icon in your workspace and choose the option you content pack.

When you look at the content pack, you will see a small icon present next to the name of the content pack. This will show that the pack has been updated. If you select the edit button, you will be sent to the home screen where you can either create a new content pack or update the existing content pack.

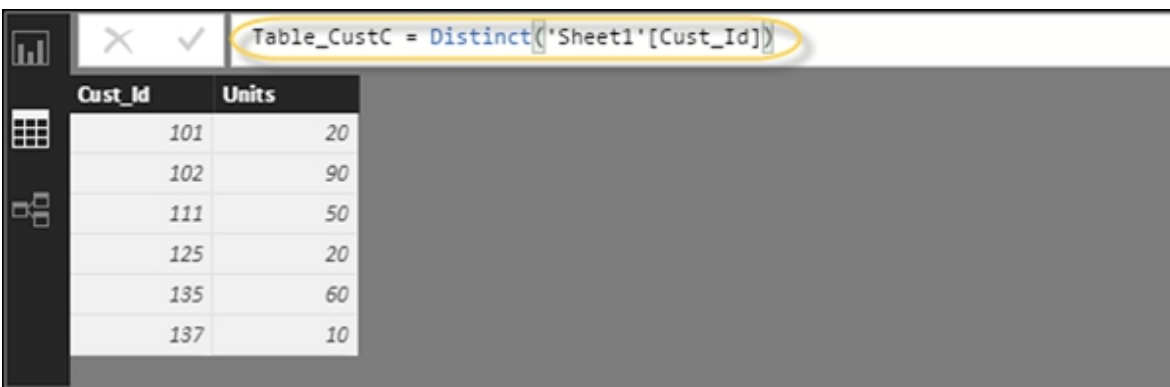
Power BI will accept every change that you make with the content pack and will publish the updated content packs to the Gallery. It will also send notes to the users about the update me to the content pack.

# Chapter Nine: Introduction To DAX

Now that we have covered the basics of Power BI let us look at how we can use the DAX language in Power BI.

## An Introduction To DAX

The Data Analysis Expressions or DAX language is a formula expression language. This is used in most visualization tools, including Power BI. This language is also called a function language where the full code will be kept within a function. This programming language has two data types - Other and Numeric. The latter will include integers, decimals, and currency, while the former will include binary and string objects. In the screenshot below, we will use a DAX formula, which will use a function to calculate the column of the table.



The screenshot shows the Power BI interface. At the top, the formula bar contains the DAX formula: `Table_CustC = Distinct('Sheet1'[Cust_Id])`. Below the formula bar, a table is displayed with two columns: **Cust\_Id** and **Units**. The table contains the following data:

Cust_Id	Units
101	20
102	90
111	50
125	20
135	60
137	10

The DAX function will also include conditional statements, functions, and value references. Let us look at some key points about DAX that will enable you to understand these concepts in an easier way.

- Since the code in DAX is always a function, the language is termed a functional language. A DAX statement that is executable will contain nested functions, value references, conditional statements and more

- As mentioned earlier, there are two primary data types used in DAX formulae
- A DAX expression will first evaluate the innermost functions before it looks at the outermost function. It is for this reason that you should format the formula correctly

You can also use mixed data types as inputs in any DAX formula, and the conversion will automatically take place. You can execute a formula easily in the DAX. The output values will always be converted into a data type that you have instructed in the formulae and functions.

## **Importance Of DAX**

You may wonder why it is necessary to use the DAX language in Power BI. You must understand that the DAX Language will make it easier for you to work with Power BI efficiently. As we have read earlier, we know that Power BI is used to make reports using different functionalities like transforming, visualizing, and data importing. You must need to have a basic knowledge of the Power BI desktop if you want to create a report that is easy for people to understand with all the available data. If you want to improve the way you project your reports, you should learn DAX. Let us assume that you want to make a report that will help you analyze the growth percentage of sales across different cities, states, or districts in a country. You can also use this analysis to compare the growth to sales every year. You can import these fields from the data table that you import into Power BI.

To do this, you will need to create a new measure in Power BI using the DAX language. In this manner, you can always create new measures and use those to create some exclusive visualizations. You can also obtain some unique insights into the data. It is only when you have these unique insights into the data that you can develop the required

solutions. DAX, thus, makes it easier to perform data analysis in Power BI.

## **DAX Functions**

DAX functions are predefined formulae stored in Power BI. These functions are used to perform some calculations on values in a table in an argument. These arguments will need to be added in a specific order, and they can be numbers, text, column references, logical values like TRUE or FALSE, another function or formula, or constants. Functions will also perform a specific operation on those values that are included in the argument. You can use one or more arguments in a DAX function .

## **Points About DAX Functions**

Let us look at some unique facts about a DAX function that you should know if you want to understand them better.

- A DAX function will always need to refer to a field or column in a table. You cannot use individual values in a DAX function. If you want to apply functions on different values in a column, you can apply the necessary filters in the DAX formula
- A DAX function will give you the flexibility to create the formula which is applied on every record in the table. These formulae or calculation will be applied to each row based on the arguments in the function
- In a few cases, DAX functions will return the full table, and you can use these values in other functions or formulae. You cannot display the contents of the table
- DAX functions will have a category that is known as a time intelligence function. These functions are used to calculate date/time ranges and periods.

Power BI allows you to use different types of functions that will allow you to analyze the data. You can also create new measures and columns. The list of functions will include functions from different categories like:

- Information
- Date
- Text
- Counting
- Logical
- Aggregate

Power BI will also provide a very easy way to list the various functions you can use in Power BI. When you start typing the formula in the formula bar in the Power BI screen, you will see the list of the functions that start with that alphabet. This is similar to Microsoft Excel.

## **Aggregate Functions**

There are numerous aggregate functions that you can use in Power BI like:

- SUM
- SUMX
- Max
- Min
- Average

## **Counting Functions**

Some of the counting functions you can use in DAX are:

- COUNT
- COUNTA
- COUNTROWS
- COUNTBLANK
- DISTINCTCOUNT

## **Logical Functions**

The following are the numerous logical functions one can use in Power BI:

- AND
- OR
- NOT
- IF
- IFERROR

## **TEXT Functions**

Some of the text functions you can use in Power BI include the following:

- REPLACE
- SEARCH
- UPPER
- FIXED
- CONCATENATE

## **DATE Functions**

Some DATE functions used in Power BI are:

- DATE
- HOUR
- WEEKDAY
- NOW
- EOMONTH

## **INFORMATION Functions**

Some functions used are:

- ISBLANK
- ISNUMBER
- ISTEXT
- ISNONTEXT



- ISERROR

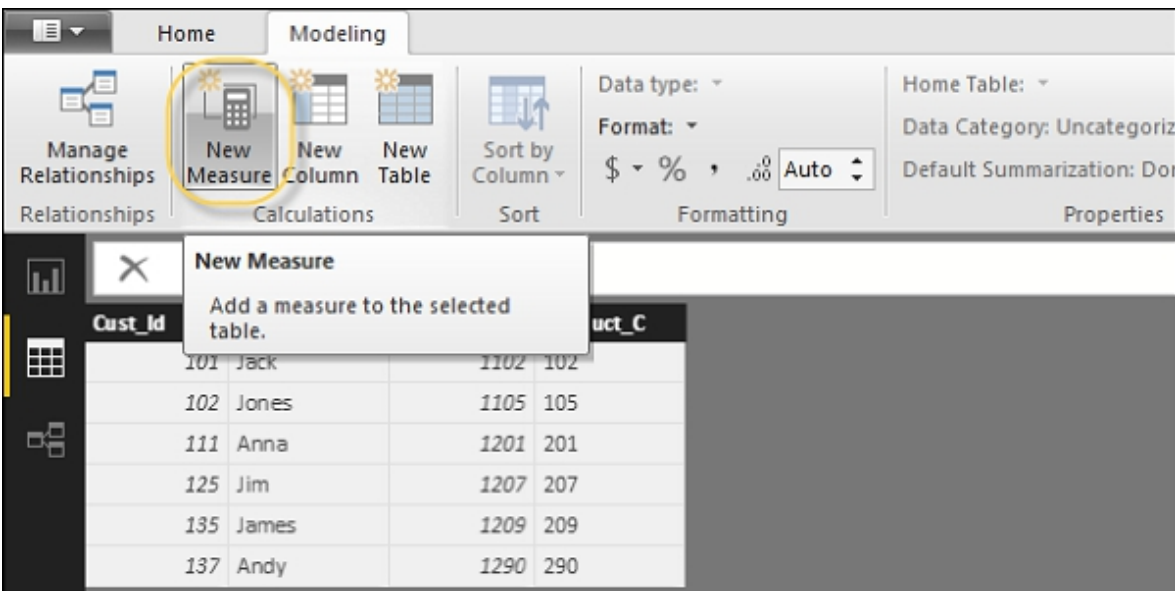
## DAX Calculation Types

You can use two types of calculations when you use DAX:

- Calculated Measures
- Calculated Columns

You can see the New Column option in the Modeling tab in the Power BI Desktop interface. This option will be present at the top of the screen. When you select this option, a formula bar will open. This is where you should enter the DAX formula to perform the required calculation. The Data Analysis Expression is a language used in Excel to perform calculations too. You can rename these columns by changing the Column text in the formula bar in the Power BI desktop.

In the example below, we will create a new column called Product Code (Product\_C). This column is derived from the last three characters in the Prod\_Id column. This is the formula used: `Product_C = RIGHT( Sheet1[Prod_Id],3)`.



If you want to create a new measure, you should move to the Modeling tab and choose the “New Measure Tab” option.

This will allow you to include a new object with the name “Measure” in the Fields tab.

DAX functions and formulae can be used to calculate the value of new measures in the same way that we calculated a new column. Experts state that DAX formulae are also called calculations since they calculate the input value to obtain a resultant value. You can always create two types of measures in Power BI using DAX: calculated measures and calculated columns.

## **Calculated Columns**

A calculated column will create a new column in the table that is present in the Report pane of the desktop. The difference between a calculated column and regular column is that you need to have at least one function in this column. You will use these values when you create a column with sorted or filtered information. Follow the steps below to create a calculated column:

1. In the Power BI desktop, move to the Modeling tab.
2. Now, choose the “New Column” option. The formula bar will appear on the screen, which will show “Column=,” where you can replace the word “Column” with the name you want to add to the data table.
3. You can then enter the expression that you want to add to the calculated column

## **Calculated Measures**

A calculated measure will create a field that will have an aggregated value like a ratio, sum, average, percentage, etc. Follow the steps below to create a calculated measure:

1. In the Power BI desktop, move to the Modeling tab.
2. Now, choose the “New Measure” option. The formula bar will appear on the screen, which will

show “Measure=,” where you can replace the word “Measure” with the name you want to add to the data table.

3. You can then enter the expression that you want to add to the calculated column
4. When you create the measure, you can modify the name using the editing options.

## Chapter Ten: DAX Parameter Naming Languages

The DAX language uses some standard parameter names to facilitate the understanding and usage of any DAX function. You can also use some prefixes to add to the parameter names. If the prefix is not very clear, you can use this prefix to help you identify the parameter name. If you want to understand the syntax of this language and function, you can use the data values in the right manner to include these function parameters. You also need to understand these parameters.

### Parameter Names

Let us look at the standard parameter names used in DAX:

<b>Sr.No.</b>	<b>Parameter Name &amp; Description</b>
1	expression A DAX expression will return only one scalar value. This is where the expression will be evaluated numerous times for each column or row.
2	value A DAX expression will return only one scalar value. This is where the expression will be evaluated before all the other operations in the formula.
3	table This is a DAX expression that will return the output in the form of a table.
4	tableName This is the name of the existing table that uses the DAX syntax. Remember, this cannot be an expression.
5	columnName The name of an existing column using standard DAX syntax, usually fully qualified. It cannot be an expression.
6	Name The name is a string constant that can be used to name any object.
7	Order The order will determine how the data needs to be sorted.
8	Ties

	Ties will determine how you can handle the tie values.
9	<p>type</p> <p>The type enumeration can be used to determine the type of data you can use for PathItemReverse or PathItem .</p>

## Prefixing Parameter Names or Using the Prefix Only

A parameter can be qualified using a prefix:

- The prefix should always be written in a way that makes it easy to read the parameter. No ambiguity should exist
- The prefix should always be descriptive of how the parameter can be used

For instance,

- Search\_ColumnName: This is a prefix that will refer to the existing column which is used to search for any value in the DAX LOOKUPVALUE() function
- Result\_ColumnName: This prefix refers to the existing column that is used to obtain the values in the DAX LOOKUPVALUE() function

You can always omit the names of the parameters and use just the prefix. If your prefix is very clear, you can use it to describe the parameter in use. When you omit the parameter name only to use the prefix, it will avoid the clutter in the function. Let us consider the function DATE (Year\_value, Month\_value, Day\_value). In this function, you can omit the names of the parameters. In this function, you can write the word “DATE” thrice in the function. This will increase the clutter in the function. To avoid this, you can use the word “DATE” as a prefix, which will make this function easy for you to read. Sometimes, however, the parameter name and prefix must be present in the function to improve clarity. For instance, consider the term Year\_columnName, the prefix is Year, and the parameter name is ColumnName. You need both of these values to make the user understand that the parameter needs the column of years to provide the required output.

# Chapter Eleven: Description Structure of DAX Functions

If you want to use DAX functions in a formula, you should understand the function that you will use in detail. You will need to learn the syntax of the function, the parameter types, return values in the function, etc. We have covered the syntax details in the ninth chapter. We will use the uniform description structure in this chapter to enable you to read and understand these DAX functions.

- Different DAX functions are grouped together by the name of the type of the DAX function, and we will cover these in detail later in the book
- Each of these chapters about these types of functions will provide you a brief description of the use of the type of DAX function
- A list of DAX functions will also follow this description
- Every DAX function will have the following description structure:

- Description

- Syntax



- Parameters

- Return Value

- Remarks

- Example

In the sections below, we will look at these headings and understand them in detail. We will also look at how you can use them in the DAX function.

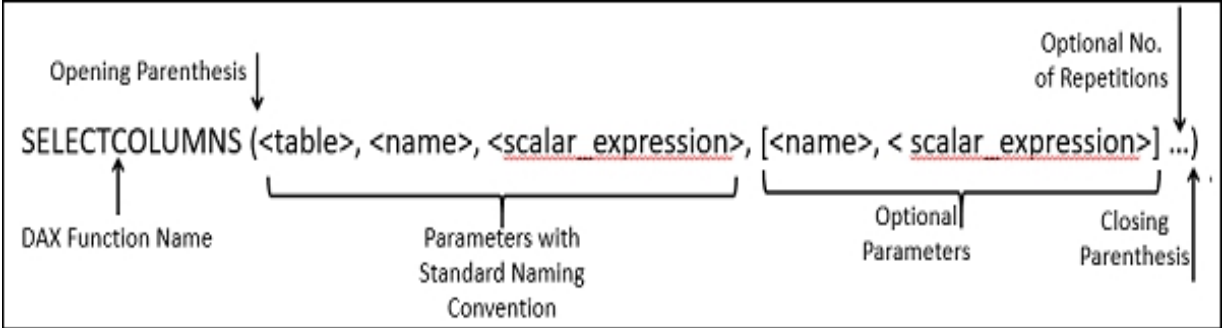
## **Description**

In this section, you will learn more about the DAX function, including information about what the function is about and where this function can be used. You will also learn more about when Microsoft introduced the functions.

# Syntax

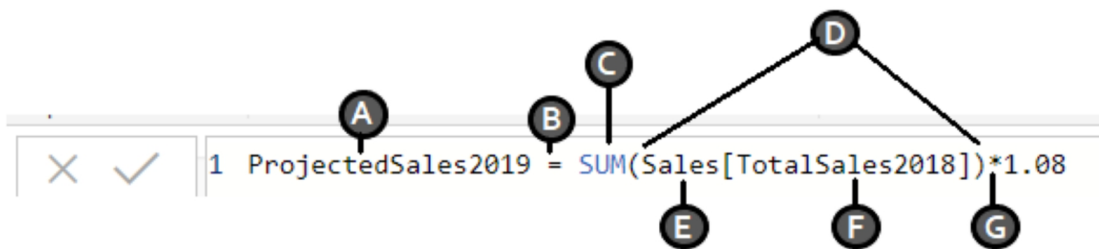
In this section, we will look at the exact name of the function and the various parameters used in the function.

## Example



## Syntax of DAX Formulae

The most important step when it comes to learning any DAX language is to break the functions down into the smaller elements and understand each of these elements. It is for this reason that you should understand the syntax of the DAX language. In the section below, we will look at an example of the DAX formula. Let us also look at each of these elements separately.



Based on the labeling in the image above, an explanation is given below about the elements in the formula:

- A: This is the name of the measure that you are adding to the visualization
- B: If you remember from Microsoft Excel, you always begin a function after the “=” sign. This operator in Power BI equates two sides of the DAX formula and is the start of the DAX formula
- C: This is the DAX function that you will use to add the values from the two tables in the data set. Since we are taking the sum of the values, we will use the SUM function
- D: The parenthesis used in the function will enclose and define the arguments that you will use in this expression. Remember every function will have at least one argument
- E: This lists the name of the table from where the column or field will be taken from

- F: This is the name of the field in the table from where the formula will use the values. For example, in this formula, we will use the function SUM to apply the values of the field or column
- G: This is another operator that is used in DAX for multiplication. The syntax elements are from A to F, and these constitute the basic syntax of the DAX function

In simple words, the DAX formula will command the system to add the values in the Total Sales 2018 column and the projection column. It will then store these values in the new measure or column called Projected Sales 2019.



## **Parameters**

We have covered the various parameters in the previous chapter. For each of the function groups that we will cover in the later sections of the book, we will look at the parameters in a table along with the description.

## **Return Value**

The return value will determine the data type of the value that the function will return after it runs through the formulae.

## **Remarks**

In the remarks section of any function, you will need to add some extra information about why you are using the DAX function. This will also help you identify any errors and also debug the formula.

# Chapter Twelve: Aggregate Functions

This chapter lists the different aggregate functions used in DAX.

## ADDCOLUMNS

This function will allow you to add columns to any table expression or table.

**Syntax** **ADDCOLUMNS** (<table>, <name>, <expression>, [<name>, <expression>] ...) **Parameters**

Sr.No.	Parameter & Description
1	table This parameter will indicate to Power BI that it should return a table
2	name This parameter will indicate the column that we will be using. This will be included in double quotes
3	Expression This parameter will return a scalar expression that is evaluated for every row in the table

**Return Value** This function will return a table with all values - both the original and added values.

**Example** **PRODUCT\_SALES = ADDCOLUMNS (**

```
Products,"East_Sales", SUMX (RELATEDTABLE(East_Sales), IF([Product] = East_Sales[Product], East_Sales[Sales Amount],0) )  
)
```

## AVERAGE

This function will return the arithmetic mean or average of the numbers in the specified column.

**Syntax** **AVERAGE** (<column>) **Parameters**

Sr.No.	Parameter & Description
1	Column This parameter will return the numbers that you want to calculate an average of .

**Return Value** This function will return a decimal number that will represent the arithmetic mean of the numbers present in the column.

## Remarks

- If there are any empty cells or logical values in the column, the values will be ignored. These rows will not be counted
- A cell with a null value will not be included, but the rows will be counted for the divisor
- If there are no rows that can be aggregated, the function will return a blank value. Having said that, if the rows in the table do not meet some criteria, the value returned will be zero

**Example = AVERAGE (Sales[Sales Amount])** Types Some of the other versions of this function are: **AVERAGEA** This function returns the average of the values in a column, including non-numeric and text values.

### Remarks :

This function will take the column and will calculate the average or the arithmetic mean of the numbers present in that column. The column will need to adhere to the following rules:

- Any value that will evaluate to "TRUE" will be assigned the number 1
- Any value that will evaluate to "FALSE" will be assigned the number 0
- Any empty text will be assigned the value zero
- The non-numeric text will be counted as zero

### Syntax :

= AVERAGEA (East\_Sales[Sales Amount]) AVERAGEX

This function will calculate the average of all the expressions used in a table.

### Remarks :

This function will enable you to evaluate different expressions across the row in any table. It will then take the results and then calculate the arithmetic mean of those values. Therefore, this function will take the table as the first argument and also the expression as the second argument in the function. This function will follow the same rules as the AVERAGE function, which means that you cannot include null cells or non-numeric cells.

### Syntax :

= AVERAGEX (East\_Sales,East\_Sales[Unit Price]\*East\_Sales[No. of Units] )  
COUNT

This function will count the number of cells that are present in a column which contains numbers.

## **Syntax COUNT (<column>) Parameters**

--

Sr.No.	Parameter & Description
1	column This parameter will indicate the column where the numbers should be counted.

**Return Value** This function will return a whole number.

**Remarks** You can use this function to count the number of records in a column, but only the numbers present in this column are taken into consideration. The following values will be counted in the column if you use this function:

- Dates
- Number

If these rows contain text, you cannot expect Power BI to translate that text into a number. This means that the row will not be counted. When the function does not find the rows to count, it will return a blank value. If there are rows and none of those rows meet the criteria, the function will return zero.

**Example = COUNT (ProductInventory[UnitsBalance])** **Types** **COUNTA** This function is used to count the number of cells present in a column which are not empty. This means that this function can be used to count those rows that include nonblank values, including dates, logical values, and text.

**Remarks :**

If the function does not find the rows in the table to count, it will return a blank value. If there are rows, but none of those rows meet any criteria, the value will be zero.

**Example :**

= COUNTA (ProductInventory[ UsageDate]) COUNTAX

This function will count the nonblank values when an expression over a table is being evaluated. This function works like the COUNTA function, which means that it can iterate through different rows in the table where there are no blanks.

**Syntax :**

COUNTAX (<table> , <expression> ) **Parameters :**

Sr.No.	Parameter & Description
1	table This parameter returns the table, which contains the row in which the expression will be evaluated.

2	expression This parameter will list the expression that should be evaluated in every row in this table.
---	---

**Remarks :**

This function will count the number of cells that contain any form of information. This information includes expressions as well. For instance, if the column you are using contains an expression which you will use to evaluate a string, this function will treat the record as a non-blank record. The COUNTAX function will not count any empty cell, but since this cell contains a formula, the cell will be included in the list. If there is no row for the function to aggregate, the function will return a blank value. Having said that, if the rows do not meet the criteria, the value of the function will be zero.

**Example :**

Medal Count Summer Sports:= COUNTAX (  
 FILTER (Results, Results[Season] = "Summer"), Results[Medal]  
 ) COUNTBLANK

This function will return the number of blank cells that are present in a column in a table.

**Example :**

= COUNTBLANK(Results[ Medal]) **Remarks :**

When you use this function, you can use columns that contain any kind of data. This function will count the number of blank cells in the table. Even those cells that have the value zero will not be counted since this number is a numeric data type and not a blank.

**Example :**

= COUNTBLANK (SalesTarget[SalesTarget]) COUNTROWS

This function is used to count the number of rows in a table or the table specified by some expression.

**Syntax :**

COUNTROWS (<table> ) **Parameters :**

Sr.N o.	Parameter & Description
1	table This parameter will indicate the name of the table that will be used by the function to count the number of rows. This parameter can also indicate the expression that will return the table.

**Remarks :**

This function is not only used to count the number of rows present in the base table but also used to count the number of rows present in a table after a context or filter is applied to that table.

**Example :**

= COUNTROWS (CALENDAR (DATE (2016,8,1), DATE (2016,10,31))) returns 92.

= COUNTROWS (Results) returns 34094.

= COUNTROWS (Events) returns 995.

***COUNTX***

This function will count the number of rows that are present in a table that contain an expression which will evaluate a number or number.

**Syntax :**

COUNTX (<table> , <expression>) **Parameters :**

Sr.No.	Parameter & Description
1	table This parameter will return the value of the table, which contains the rows that you want to count.
2	expression This parameter will return the number that you want to count.

**Remarks :**

This function will only count numeric values in the cell, including dates. The parameters which use logical values or text will not be translated into numbers, and therefore, are not counted. If this function does not find any rows that it can count, it will return a blank value. If none of the rows in the table will meet the specified criteria, the function will return the value zero.

**Example :**

= COUNTX (RELATEDTABLE (East\_Sales), IF ([Product] = East\_Sales[Product],1,0) ) CROSSJOIN

This function will return the value as a table, which contains the Cartesian product of the values that are present in the rows from the tables included in the parameter list of the function. The columns in this new table will include the columns from all the parameter tables.

**Syntax CROSSJOIN (<table1>, <table2>, [<table3>] ...)  
Parameters**

Sr.No.	Parameter & Description
1	table1



	This parameter will return the name of the first table to use in the DAX expression.
2	table2 This parameter will return the name of the second table to use in the DAX expression.
3	table3 This parameter will return the name of the third table to use in the DAX expression. This is an optional parameter .

**Return Value** This function will return the value as a Cartesian product of the values of the rows present in the tables used as a parameter in the function. The columns in the new table will return the columns in the parameter tables.

**Remarks**

- The names of the columns from the tables used as parameters should be different in all the tables. Otherwise, you will receive an error
- The number of rows in the result table will be a product of the number of rows present in the tables entered as the parameter
- The total number of columns present in the result table will be the sum of the number of columns present in the tables entered as the parameters

For instance, if the first table has r1 rows and c1 columns, the second table has r2 rows and c2 columns, and the third table has r3 rows and c3 columns, the resulting table will have: r1\*r2\*r3 rows and c1+c2+c3 columns.

**Example = CROSSJOIN (Salesperson, Products) DISTINCTCOUNT**

This function is used to count the number of distinct values present in this column .

**Syntax DISTINCTCOUNT (<column>) Parameters**

Sr.N o.	Parameter & Description
1	column This parameter will return the list of values that must be counted.

**Return Value** This function will return a whole number.

**Remarks** If you want to use this function, you can use a column that contains any type of data. If the function does not find any rows to count, it will return a blank value.

**Example = DISTINCTCOUNT (Sales[Account]) GENERATE**

This function is used to return a table that contains the Cartesian product of the rows that are present in the table entered as the parameter and the table that is returned as a result of evaluating the second table in the context of the first table.

**Syntax GENERATE (<table1>, <table2>) Parameters**

Sr.No.	Parameter & Description
1	table1 This parameter will return the value of the first table that should be used in the DAX expression.
2	table2 This parameter will return the second table, which needs to be evaluated under the pretext of the first table.

**Return Value** This will return a value as a parameter that can be passed into a DAX function.

**Remarks**

- If the resulting table when the second table is evaluated based on the first table comes up empty, the final table will not contain any rows from the first table. This is different from the GENERATEALL function. In this function, all the rows from the first table will be included even if the second table does not have any entries
- The column names in both tables should be different

**Example = GENERATE (**

SUMMARIZE(Salesperson,Salesperson[Salesperson]),  
SUMMARIZE(SalesTarget,SalesTarget[SalesTarget], "MaxTarget",MAX(SalesTarget[SalesTarget])) ) Types GENERATEALL

This function will return a table that is the Cartesian product between the rows of the first table and the table, which results from evaluating the second table in the context of the first table.

**Syntax :**

GENERATEALL (<table1> , <table2>) **Example :**

= GENERATEALL (
  
SUMMARIZE(Salesperson,Salesperson[ Salesperson]),
SUMMARIZE( SalesTarget,SalesTarget[ SalesTarget], "MaxTarget" ,MAX( Sales
Target[SalesTarget])) )

**MAX**

This function will return the largest value that is present in the column.

**Syntax MAX (<column>) Parameters**

Sr.No.	Parameter & Description
1	column This parameter will return the column within which you want to find the largest numeric value.

**Return Value** The function will return a decimal number.

**Remarks** This function will consider the following types of values when looking for the maximum value:

- Dates
- Numbers

The function will ignore logical values, text, and empty cells.

**Example = MAX (Sales[Sales Amount] ) Types MAXA** This function, like the MAX function, will return the largest value in the column.

**Remarks :**

This function will take a column as the argument and will look for the maximum value from the following data types:

- Dates
- Numbers
- Logical values like TRUE and FALSE. The row with TRUE will be valued as one, and that with FALSE is valued at zero

An empty cell is always ignored. If the column does not contain any values which adhere to the criteria, this function will return zero.

**Example :**

= MAXA (ProductInventory[ UsageDate]) MAXX

This function will evaluate the expression in every row in the table and then return the cell value with the largest number.

**Remarks :**

This function will only evaluate the following values:

- Dates
- Numbers. If the expression does not return a number, the value returned by the function is zero

Text values, logical values, and empty cells are ignored.

**Example :**

= MAXX (East\_Sales,East\_Sales[No. of Units]\*East\_Sales[Unit Price]) MIN

This function will return the smallest numeric value that is present in the column entered as the parameter.

**Syntax MIN (<column>) Parameters**

Sr.No.	Parameter & Description
1	column This parameter will return the column in which you are looking for the minimum values.

**Return Value** The function returns a decimal number.

**Remarks** The function only considers the following types of values:

- Dates
- Numbers

Text, empty cells, and logical values are ignored.

**Example = MIN (Sales[Sales Amount]) Types MINA This function will return the smallest value present in the column, including the numbers represented as text and logical values.**

**Syntax :**

MINA (<column>) **Remarks :**

The function will consider those cells with the following values:

- Dates
- Numbers
- Logical values like TRUE and FALSE. The row with TRUE will be valued as one, and that with FALSE is valued at zero
- Text that can be converted into a numeric value

As with the previous function, this function also ignores empty cells. If the cells in the columns do not contain any values, the function will return zero .

**Example :**

= MINA (ProductInventory[InventoryDate]) MINX

This function will return the smallest numeric value that is present in the column. It will either look for numeric values or look for those expressions that return numeric values.

**Remarks :**

This function will evaluate the result of the expressions based on the following rules:

- This function only counts numbers. If there are no numbers in this expression, the function will return zero
- Logical values, text values, and empty cells are always ignored. Any number that is represented as text will be treated as text

**Example :**

= MINX (East\_Sales,East\_Sales[No. of Units]\*East\_Sales[Unit Price]) PRODUCT

This function will return the product of the numbers in a column. This function was introduced in 2016 .

**Syntax PRODUCT (<column>) Parameters**

Sr.N o.	Parameter & Description
1	column This parameter will return the column that contains the numbers in the column for which the product will be calculated.

**Return Value** This function will return a decimal number.

**Remarks** This function will only look at the numbers present in the column entered as the parameter. It will ignore the text, blanks, and logical values.

**Example = PRODUCT (ProductInventory[InventoryDuration])  
Types PRODUCTX**

This function will return the product of those numbers that result from when an expression is evaluated for every row in the table entered as the parameter .

**Syntax :**

PRODUCTX (<table> , <expression>) **Parameters :**

Sr.N o.	Parameter & Description
1	table This parameter will return the table, which contains the row for which the expression must be evaluated.
2	Expression This parameter will return the expression that will need to be evaluated for every row in the table.

**Remarks :**

This function will only consider those numbers in the column. Text, blanks, and logical values will be ignored.

**Example :**

= [PresentValue] \* PRODUCTX (AnnuityPeriods, 1 + [FixedInterestRate]) ROW

This function will return the table with only one row. This row will contain the values that will result from the expression that is given in the columns in this table.

**Syntax** ROW (<name>, <expression>, [<name>, <expression>] ...) **Parameters**

Sr.No.	Parameter & Description
1	name This parameter is the name of the column, which is enclosed in double-quotes.
2	expression This parameter or expression will return a scalar value that will help to populate the column mentioned above

**Return Value** This function will return a table with one row.

**Remarks** You need to include a pair of names and expressions when you use this function.

**Example** = ROW (Total Number of Products, COUNTA (Products, Products[Product\_key]), Total Sales Value, SUM (Sales, Sales[ExtendedAmount])) **SELECTCOLUMNS**

This function will add all the calculated columns in a table expression or a given table, and it was introduced in the year 2016.

**Syntax** SELECTCOLUMNS (<table>, <name>, <scalar\_expression>, [<name>, <scalar\_expression>] ...) **Parameters**

Sr.No.	Parameter & Description
1	table This parameter is either a DAX expression that can return a table or a table.
2	name This parameter is the name that is given to the column, and this is enclosed in double-quotes.
3	scalar_expression The DAX expression will return a scalar value, which can be an integer, string, or column reference.

**Return Value** This function will return a table with the same number of rows that are present in the table mentioned in the parameters of the function. The table that is returned will have one column that is named after each pair of scalar\_expression and name parameters. Every scalar\_expression will always be evaluated in the context of some row in the table specified as the parameter .

**Remarks** This function is similar to the ADDCOLUMNS function and behaves in the same way except that it will not start only with the table specified. This function will always begin with an empty table, and columns will be added to it based on the results provided by the function.

**Example = SELECTCOLUMNS (**

Products,"Product-NoOfUnits",Products[ Product]&" - "&Products[ Units Sold])  
SUM

This function will return the sum of the numbers present in a column.

**Syntax SUM (<column>) Parameters**

Sr.No.	Parameter & Description
1	column This parameter will contain the name of the column that contains the numbers that are used to calculate the sum.

**Return Value** This function returns a decimal value .

**Remarks** This function will return a null value if the rows contain non-numeric values.

**Example = SUM ([Sales Amount]) SUMMARIZE**

This function will return the summary table that you request over a set of columns or groups.

**Syntax SUMMARIZE (<table>, <groupBy\_columnName>, [<groupBy\_columnName>] ..., [<name>, <expression>] ...) Parameters**

Sr.No.	Parameter & Description
1	table This parameter returns a DAX expression that will return the table of data.
2	groupBy_columnName This parameter will provide the list of the

	existing column that can be used to create a group or summary based on the various values present in the column. You cannot use an expression in this parameter.
3	name This function will return the name of the total of the column. The name of the column is enclosed in double-quotes.
4	expression This parameter will return the scalar value, and the expression will need to be evaluated for every row or context .

**Return Value This function will return a table with the names of the columns written in the parameter groupBy\_columnName. The summary of these columns will be named based on the name parameters.**

### Remarks

- Every column that you define will need to have a corresponding expression. If there is no expression, an error will occur. The first parameter in this function will define the name of the column that will be used in the result. The second parameter called the expression will define the calculation that will be performed to obtain this value for every row in the selected column
- You should have the groupBy\_columnName either in a table related to a table or in a table alone
- Every name should always be enclosed in double quotes
- The function will group a selected set of rows into one set of values. These values will be grouped into groupBy\_columnName. The function will return a value for every group in that list

### Example = SUMMARIZE (

SalesTarget,SalesTarget[SalesTarget],"MaxTarget",MAX  
(SalesTarget[SalesTarget])) SUMMARIZE With Options Now that you know what the summarize function does let us look at how you can use different options within that function. You can use the following options within the summarize function:

- ISSUBTOTAL function
- ROLLUP function
- ROLLUPGROUP function

If you use these functions in the SUMMARIZE function, you will obtain different results:

- If you use the ISSUBTOTAL option within the SUMMARIZE function, you can create a column that will contain logical values. The ISSUBTOTAL functions return these values in the output table. The value will be TRUE or FALSE. The resulting value will be the former if



the rows in the table contain sub-total values of the column as a parameter to ISSUBTOTAL; otherwise, it will be the latter

- If you use the ROLLUP function, the SUMMARIZE function will react differently by adding some roll-up rows to the result table based on the groupBy\_columnName column
- The ROLLUPGROUP function can be used within the ROLLUP function which allows you to prevent the roll-up of rows in partial subtotals

**Syntax SUMMARIZE (<table>, <groupBy\_columnName>, [<groupBy\_columnName>] ..., [ROLLUP (<groupBy\_columnName>, [<groupBy\_columnName> ...])], [<name>, <expression>] ...) SUMMARIZE (<table>, <groupBy\_columnName>, [<groupBy\_columnName>] ..., [ROLLUPGROUP (<groupBy\_columnName>, [<groupBy\_columnName> ...])], [<name>, <expression>] ...) SUMMARIZE (<table>, <groupBy\_columnName>, [<groupBy\_columnName>] ..., [ROLLUP (ROLLUPGROUP (<groupBy\_columnName>, [<groupBy\_columnName> ...])], [<name>, <expression>] ... ) SUMMARIZE (<table>, <groupBy\_columnName>, [<groupBy\_columnName>] ..., [ROLLUP (<groupBy\_columnName>, [<groupBy\_columnName> ...])], [<name>, {<expression> | ISSUBTOTAL (<columnName>)}] ...) Parameters (ROLLUP / ROLLUPGROUP Function)**

Sr.No.	Parameter & Description
1	groupBy_columnName This parameter will use the qualified name of the existing group of columns and then create a summary group using the values found in those columns. You cannot use this parameter as an expression.

### Parameters (ISSUBTOTAL Function)

Sr.No.	Parameter & Description
1	columnName This parameter will return the name of the column that is present in the SUMMARIZE function or any other column that will be present in the table or the table of a table.

We looked at the other parameters used in the SUMMARIZE function in the previous section .

**Return Value** This function will return a table that has selected columns based on the `groupBy_columnName` parameter. The parameter names will design the summarized columns. It will also include the roll-up rows based on the `groupBy_columnName` column. The subtotals will not be displayed if the `ROLLUPGROUP` will not be used within the `ROLLUP`. If you use the `SUBTOTAL` function, you will obtain an additional column that contains the logical expressions `TRUE` and `FALSE`. The former will be present in the cells containing the sub-total value for the parameters in the column name, and the latter will be present otherwise.

**Remarks** The columns that are present in the `ROLLUP` function will not be referenced using the `groupBy_columnName` parameter in the `SUMMARIZE` function. The `ISSUBTOTAL` function will only use the expression parameter in the `SUMMARIZE` function, and this function can precede the matching column name. You can use the `ROLLUP` function as a parameter within the `SUMMARIZE` function, and the `ROLLUPGROUP` function as a parameter of both the `ROLLUP` and `SUMMARIZE` functions.

**Example - ROLLUP**

```
= SUMMARIZE (
    SalesTarget,          ROLLUP          (SalesTarget[SalespersonID]),
    SalesTarget[SalesTarget], "MaxTarget", MAX (SalesTarget[SalesTarget]) )
Example - ROLLUP with ROLLUPGROUP
```

```
= SUMMARIZE (
    SalesTarget, ROLLUP (ROLLUPGROUP (SalesTarget[SalespersonID])),
    SalesTarget[SalesTarget], "MaxTarget", MAX(SalesTarget[SalesTarget]) )
```

**Example - ISSUBTOTAL**

```
= SUMMARIZE (
    SalesTarget, ROLLUP (ROLLUPGROUP (SalesTarget[SalespersonID])),
    SalesTarget[SalesTarget], "MaxTarget", MAX (SalesTarget[SalesTarget]),
    "IsSubTotalSalesTarget", ISSUBTOTAL (SalesTarget[SalesTarget]) )
```

**SUMX**

This function will return the sum of the expression that will be evaluated within every row in the table.

**Syntax SUMX (<table>, <expression>) Parameters**

--	--

<b>Sr.N o.</b>	<b>Parameter &amp; Description</b>
1	table This parameter will contain the name of the table for which you will need to consider the rows where the expression will need to be evaluated.
2	expression This parameter is an expression that can be evaluated for every row in the table.

**Return Value** This function returns a decimal number as the value.

**Remarks** This function will only return the number present in the column that where the expression can be evaluated. The cells with text values, logical values, or blanks, will be ignored.

**Example USA Gold Medal Count:= SUMX(**

Results,IF(AND([Country] = "USA",[Medal] = "Gold") = TRUE(),1,0) )

## **TOPN**

This function will return the maximum number of rows that have been specified in the table .

**Syntax** TOPN (<n\_value>, <table>, <orderBy\_expression>, [<order>], [<orderBy\_expression>, [<order>]] ...)

### **Parameters**

<b>Sr.N o.</b>	<b>Parameter &amp; Description</b>
1	n_value This parameter will determine the number of rows that the function should return. This parameter will be a DAX expression that will return a scalar value, and the expression is evaluated numerous times for each context or row.
2	table This parameter will return a DAX expression that will return the table with the data where it will need to extract the top n_value number of rows present in the table.
3	orderBy_expression This parameter is a DAX expression which is used to sort the table. This parameter will be applied to every row in the table.
4	order This is an optional parameter to use. You can specify the value depending on which the rows will be sorted: FALSE or zero: This will sort the data in a descending order TRUE or one: This will sort the data in an ascending order

**Return Value** This function will return either of the following values:

- An empty table if the value of n\_value is less than or equal to zero
- A table with the top n\_value number of rows from the table if the value of n\_value is greater than zero

The rows do not have to be sorted in a specific order.

**Remarks**

- TOPN will not guarantee that the data will be sorted in the right order
- If there is a tie, then every tied row will be returned. This function will return more than n\_value number of rows if there is a tie

**Example** = **SUMX** (**TOPN**  
**(15,Sales,Sales[Salesperson],ASC),Sales[Sales Amount])**

## Chapter Thirteen: Filter Functions

In this chapter, we will look at the different filter functions that you can use in the DAX language.

### ADDMISSING ITEMS

This function will add a combination of items from numerous columns to a table. This function will look for any missing items alone, but will not duplicate items. The function will use the referencing columns to identify the right items that need to be included in the table. This function was introduced in 2016.

**Syntax**                    **ADDMISSINGITEMS**                    (**<showAllColumn>**,  
**[<showAllColumn>]**    ...,    **<table>**,    **<groupingColumn>**,  
**[<groupingColumn>]**                    ...,                    **[<filterTable>]**                    ...)  
**ADDMISSINGITEMS** (**<showAllColumn>**, **[<showAllColumn>]**  
 ..., **<table>**, **[ROLLUPISSUBTOTAL** (**<groupingColumn>**,  
**<isSubtotal\_columnName>**,                    **[<groupingColumn>**,  
**[<isSubtotal\_columnName>]**]    ...)],    **[<filterTable>]**    ...)

### Parameters

Sr.No.	Parameter & Description
1	showAllColumn This parameter is the name of the column that the function will look at to return those items with no data.
2	table This parameter will contain the items with non-empty data fields where the calculated fields option is used.
3	groupingColumn This parameter will return the column, which is used by the group of columns that are supplied in the table argument.
4	isSubtotal_columnName This parameter will return a Boolean column, which is supplied as a table argument. This will contain the ISSUBTOTAL value for the column groupingColumn.
5	filterTable A table representing filters to include in the logic for determining whether to add specific combinations of items with no data. Used to avoid having ADDMISSINGITEMS add item combinations that are not present because a filter removed them .

**Remarks** This function will evaluate the following based on a combination of items from various columns:

- Crossjoin will be applied across different tables
- AutoExist will be applied to the same table

## **ADDMISSINGITEMS with ROLLUPGROUP**

The ROLLUPGROUP function can be used in the ROLLUPGROUPSUBTOTAL function to reflect the roll-up rows present in the table.

### **Restrictions**

- If you use the ROLLUPSUBTOTAL function to define the table arguments that you supply or you add the ISSUBTOTAL columns and the equivalent rows by another method, you should use the ROLLUPSUBTOTAL function along with the ADDMISSINGITEMS function. This holds true for ROLLUPGROUP too if this function is used along with the ROLLUPSUBTOTAL function to define the arguments for the supplied table argument
- When you use the ROLLUPSUBTOTAL function in the ADDMISSINGITEMS function, you need to ensure that the ROLLUPGROUP is present in the supplied table argument and ISSUBTOTAL columns that correspond to every group in the column. The names of the ISSUBTOTAL column will need to contain the values obtained from ROLLUPSUBTOTAL function, which is present within the ADDMISSINGITEMS. These columns will also need to match the names of the columns present in the supplied table argument
- If the ROLLUPSUBTOTAL is used along with the ROLLUPGROUP to define the table argument, you need to use at least one ISSUBTOTAL column name against the ROLLUPGROUP. This column name should match the ISSUBTOTAL column name present in the table

If the ISSUBTOTAL returns blank rows or columns, the ADDMISSINGITEMS will return a blank value.

**Example = ADDMISSINGITEMS (Products, FILTER (Products,Products[Product]=" Air Purifier")) ALL**

This function is either used to return all the values or rows in a column or table, respectively. You can use this function to clear any filters or calculate any new measures on the rows in the table.

**Syntax ALL ({<table> | <column>, [<column>], [<column>] ...}) Parameters**

<b>Sr.No.</b>	<b>Parameter &amp; Description</b>
1	table This parameter will include the name of the table on which

	you want to apply filters.
2	Column This parameter will name the column where you want to clear the filters.

The arguments passed to the ALL function should either be a reference to a base column or a base table. You can never use the table expression or column expression if you use this function.

**Return Value** This table will return a table, column, or columns without any filters.

**Remarks** You cannot use the ALL function by itself. You will need to use it as an intermediate function, and you can change the set of the results obtained using a calculation.

**Example** = COUNTA (Results[Medal])/CALCULATE (COUNTA (Results[Medal], ALL ( Results))) When you use this DAX formulae, the rows in the resulting table will be considered in the CALCULATED function with the ALL function containing a filter. When you do this, you will need to include the total number of values in the denominator.

### ALLEXCEPT

This function will always remove the context filters used in the table. It will, however, not remove those columns that specifically have filters applied to them.

**Syntax** ALLEXCEPT (<table>, <column>, [<column>] ...)  
**Parameters**

Sr.No.	Parameter & Description
1	table This parameter is the name of a table which will contain no context filters. The table can, however, include filters for those columns where they have been applied specifically.
2	column This parameter will contain one or more columns which will specify the context filters, and these must be preserved.

If you use the ALLEXCEPT function, you will need to first refer to the base table. The other arguments in the function can only be references made to the base table. You can never use a column expression or table expression with the ALLEXCEPT function .

**Return Value** This function will return a table without any filters used in the table, except for those columns with filters.

**Remarks** The ALLEXCEPT function can never be used by itself. It will serve as an intermediate function that can be used to change the results over a specific calculation performed. You can also use this function if you want to remove the filters but never remove the columns in the table.

**Example** = CALCULATE (COUNTA (Results[Medal]), ALLEXCEPT (Hosts, Hosts[City])) The values in the Medal column in the above formula is present in the Results table. These values will be counted without any filters except for those filters that are present in the Hosts table in the column City.

### **Types ALLBLANKROW**

This function will return the rows in the table except for the blank rows. It will also return the distinct values of the columns present in the table but the blank rows. The function will disregard those filters that are context in nature .

#### **Syntax** :

ALLNOBLANKROW (<table>|<column>) **Return Value** :

This function will return the following values:

- A column with values where the argument is only a column
- A table where the argument will only be a table

#### **Remarks** :

This function will not consider the blank rows in a table but will consider those blank rows if they are generated in the parent table. The function will consider those blank values or non-matching values in the child tables.

#### **Example** :

= COUNTROWS (ALLNOBLANKROW (Salesperson)) This formula will return a numeric value depending on the number of rows present in the Salesperson table. Let us assume that there is a Sales table which has a list of some unaccounted Salespeople.

= COUNTROWS (ALL (Salesperson)) This formula will return a different number when compared to the Salesperson table since there are some records of unaccounted Salespeople in the Sales table .



## **ALLSELECTED**

This function will get the context that will represent the rows and columns present in the query. It will keep the explicit filters assigned to specific columns other than any context row and column filters. You can use this function to obtain a visual total in the query.

### **Syntax** :

ALLSELECTED ([<tableName> | <columnName>]) **Parameters** :

<b>Sr.N o.</b>	<b>Parameter &amp; Description</b>
1	tableName This is an optional parameter and will contain the name of the table. You cannot use an expression in this parameter.
2	columnName This is also an optional parameter, which is the name of the column. This cannot be an expression.

### **Remarks** :

- You can include either the columnName or tableName argument
- This function will either take one or zero arguments
- This function will be different from the ALL function since it will retain all the filters that are set within the query. This means that the function will retain all the context filters except for row and column filters.

### **Example** :

```
SumTotal:= CALCULATE (SUM (Sales[Sales Amount]),ALLSELECTED ())  
CALCULATE
```

This function will evaluate different expressions in the table that are in context with the modified using specific filters.

### **Syntax CALCULATE (<expression>, [<filter1>], [<filter2>] ...) **Parameters****

<b>Sr.N o.</b>	<b>Parameter &amp; Description</b>
1	expression This parameter is the expression that must be evaluated.
2	filter1, filter2, ...  This parameter is optional and is a list of comma-separated expressions that are Boolean. This can also be a table expression that will define the filter .

**Return Value** This function will return the result of the expression entered as the parameter.

**Remarks** The function will use the expression as the first parameter. This is the same as the calculated field. If you use a Boolean expression as the argument, the following rules will apply:

- You cannot use a function in the expression that will either scan or return a table, including an aggregate function
- Expressions cannot use nested CALCULATE function
- You cannot reference a calculated field in the expression

You can, however, use a Boolean expression to use a function that will look for a single value. It will return a scalar value. If you filter the data, the CALCULATE function will change how the data is filtered. It will also evaluate the expression in a new context that you define or specify. For every column that you use in the filter argument, you can use existing filters on the column, which is removed. You can also filter the arguments instead.

**Example = COUNTA (Results[Medal])/CALCULATE (COUNTA (Results[Medal], ALL (Results) ) CALCULATETABLE**

This function will evaluate the table expression based on the given filters.

**Syntax CALCULATETABLE (<expression>, [<filter1>], [<filter2>] ...) Parameters**

Sr.No.	Term & Definition
1	expression This parameter shows the expression that should be used to evaluate the table.
2	filter1, filter2 ... This parameter is a Boolean expression that will define a filter to be used in a table expression.

**Return Value** This function will return a table with values.

**Remarks** The expression which is used as the first parameter should be in the form of a function so it can return a table as the output. If you use a Boolean expression as the argument, the following conditions will hold true:

- You cannot use a function in the expression that will either scan or return a table, including an aggregate function
- Expressions cannot use nested CALCULATE function

- You cannot reference a calculated field in the expression

You can, however, use a Boolean expression to use a function that will look for a single value. It will return a scalar value. If you filter the data, the CALCULATE function will change how the data is filtered. It will also evaluate the expression in a new context that you define or specify. For every column that you use in the filter argument, you can use existing filters on the column, which is removed. You can also filter the arguments instead. This function is also used in place of the RELATEDTABLE function.

### Example = SUMX (

```
CALCULATE (East_Sales, FILTER (East_Sales, East_Sales[Product]=
[Product])), East_Sales[Sales Amount]
)
```

### CROSSFILTER

This function will specify the cross-filtering direction that Power BI will need to take to calculate a value based on a relationship that exists between two or more columns. This function was introduced in 2016.

**Syntax CROSSFILTER (<columnName1>, <columnName2>, <direction> ) Parameters**

Sr.N o.	Parameter & Description
1	columnName1 This parameter will include the name of the column that will represent the type of relationship that needs to be used - the data table side or many sides. If the arguments are not given in the right order, the function will swap the order before it uses the arguments. The arguments used cannot be expressions.
2	columnName2 This parameter is also a name of the column, which represents the type of relationship to be used - one side or lookup table side. If the arguments are not given in the right order, the function will swap the order before it uses the arguments. The arguments used cannot be expressions.
3	Direction This parameter will define the direction in which the cross-filter should be applied: One – Filters on one or lookup table side of the relationship filter with many sides. Both – Filters on either side filter the other. None – No cross-filtering occurs along with this relationship .

**Return Value** This function will not return any value, but will only set the direction in which cross-filtering should take place for any indicated relationship. This is done while the query runs.

**Remarks**

- If you use a one-to-one relationship, there will be no difference between the one or many direction cross-filtering
- You can use the CROSSFILTER function only when you want to take filters as arguments. For instance, you can use the CALCULATETABLE, CALCULATE, CLOSINGBALANCEMONTH, CLOSINGBALANCEYEAR, OPENINGBALANCEMONTH, CLOSINGBALANCEQUARTER, OPENINGBALANCEQUARTER, OPENINGBALANCEYEAR, TOTALQTD, TOTALYTD and TOTALMTD
- This function will use any existing relationship in the model and identify the relationship based on the endpoint column
- In this function, the cross-filtering setting in a relationship does not hold any importance. You do not have to worry about whether the relationship filter is set to one or two directions since this does not affect the usage of that function. This function will always override any cross-filtering settings
- You will receive an error if the columns that you have named in the arguments are not included in the relationships. You will also receive an error if the arguments belong to different relationships
- If you nest the CALCULATE expressions, and you include the CROSSFILTER function in more than one CALCULATE expression, the innermost CROSSFILTER will be the one that is used throughout if there is any ambiguity

**Example = CALCULATE (Sales[Distinct Count of Products], CROSSFILTER (Sales[Product],Products[Product],Both)) DISTINCT**

This function will return a table with only one column. This column will contain the distinct values that are present in a specified column. In simple words, this function will remove any duplicate values and will only use the unique values in the column. Remember that you will need to use the DISTINCT function within a formula if you want to pass the distinct values into another function.

**Syntax DISTINCT (<column> ) Parameters**

Sr.N o.	Parameter & Description

1	column This parameter will be the column from where you need to return unique values. You can also use an expression here that will return the column to use.
---	---

**Return Value** This function will return a column with unique values in it.

**Remarks** The results of this function are always affected if there are any context filters in the column. For instance, when you use the formula to create a calculated field, the result of that field will never change even when a filter is applied to a region.

**Example = COUNTROWS (DISTINCT (Sales[Salesperson ID])) EARLIER**

This function will return the value that is present in the specified column in any outer evaluation.

**Syntax EARLIER (<column>, <number>) Parameters**

Sr.No.	Parameter & Description
1	column This parameter is a column or a DAX expression that can be used to resolve a column.
2	number This parameter is optional and is a positive number that is sent to the outer evaluation pass. The evaluation level is represented as one, the next one as two, and so on. If you do not include this number, the default value will be one.

**Return Value** This function will return the current value in a row or column as the evaluation passes.

**Remarks** This function is always used for a nested calculation. When you use this function, you do not want to use a specific value as an input. This function will produce an output based on that input. If you use this function in Power BI or Microsoft Excel, you can perform these calculations within the context of the current row. In DAX, however, you will need first to store the value of the input before you make the right calculation. This function is often used in the context of columns. This function will succeed only when there is a row that is present before the beginning of the table. It will otherwise throw an error.

**Example** If you use a Sales table that includes sales data, you can then create a calculated column using the rank of the amounts of sales in the following manner: =**COUNTROWS** (

FILTER (Sales, EARLIER (Sales[Sales Amount])<Sales[Sales Amount]) )+ 1

### **EARLIEST**

This function will return the current value of the column that is specified in the function parameter. The function will run through the evaluation pass based on the specified column.

#### **Syntax EARLIEST (<column>) Parameters**

<b>Sr.No.</b>	<b>Parameter &amp; Description</b>
1	column This is a parameter that refers to a specific column in the data set .

**Return Value** The function will return the value present in the current row in the column selected. This value will be the one stored at the outermost evaluation pass.

**Remarks** This function is much like the EARLIER function, but you can specify the level of recursion that you want to include. The results of the EARLIER and EARLIEST functions will be the same only if the number parameter in the EARLIER function is set to one.

**Example** If you want to use a Sales table that contains information about sales, you can create a calculated column that will rank the sales amounts in the following manner: = COUNTRROWS (

FILTER (Sales, EARLIEST (Sales[Sales Amount])>Sales[Sales Amount]) )+ 1

**FILTER**

This function will return a table that will represent a subset of an expression or another table.

**Syntax FILTER (<table>, <filter>) Parameters**

Sr.No.	Parameter & Description
1	table This parameter is a table that you want to use. You can filter this table to perform the necessary operations. You can either use a table name or an expression that will result in a table.
2	filter This parameter is a Boolean expression that you need to evaluate for every row in the table.

**Return Value** This function will return a table only with the data after a filter is applied.

**Remarks** You can use the DAX filter function if you want to reduce the number of rows present in the table you are using. You can use specific data from the columns to perform any calculation. This function cannot be used independently, but it can be used as a function that is embedded in different functions that use a table as the argument.

**Example Medal Count Summer Sports:= COUNTAX (**

FILTER (Results, Results[Season] = "Summer"), Results[ Medal]

) FILTERS

This function will return the values that are directly applied as a filter in the columnName parameter.

**Syntax FILTERS (<columnName>) Parameters**

Sr.No.	Parameter & Description
1	columnName This parameter will name the column in a table. Remember, you cannot use an expression in this parameter.

**Return Value** This function will return a value that can be used to apply a filter to the selected columnName.

**Example** You can calculate the number of direct filters on a column using the following: = COUNTROWS (FILTERS (Sales[Region])) HASONEFILTER

This function will return the value TRUE if there is only one value in the columnName that is filtered. Otherwise, this function will return the value FALSE .

**Syntax HASONEFILTER (<columnName>) Parameters**

Sr.N o.	Parameter & Description
1	columnName This parameter will take the name of a column in the table using the DAX syntax. You cannot use an expression in this parameter.

**Return Value** This function will return a Boolean value - TRUE or FALSE.

**Remarks** This function is very similar to the HASONEVALUE function, with the difference that this function will work on filters that have directly been applied to the column. HASONEVALUE is a function that will work based on any cross-filters that you use on the data.

**Example = HASONEFILTER (Sales[Product]) HASONEVALUE**

This function will return the value TRUE if there is only one distinct value in the column after a filter has been applied. Otherwise, it will return FALSE.

**Syntax HASONEVALUE (<columnName>) Parameters**

Sr.N o.	Parameter & Description
1	columnName This parameter will name the column that can be used in this function. Remember, you cannot use an expression.

**Return Value** This function will return a Boolean result - TRUE or FALSE.

**Example = HASONEVALUE (Sales[Product]) ISCROSSFILTERED**

This function will return the value TRUE when a column in the table selected or a related table is filtered.



## Syntax ISCROSSFILTERED (<columnName>) Parameters

Sr.No.	Parameter & Description
1	columnName This parameter is the name of the column in the table. You cannot use an expression for this parameter.

**Return Value** This function will return either TRUE or FALSE .

### Remarks

- Any column in the table is filtered if a filter is directly applied to the column. You can use one or more filters on a column
- The column columnName will be cross filtered if the filter is applied to a different column either in the related table or the same table

You can also use this function if you want to find a column that has been filtered directly.

## Example ISCROSSFILTERED (<columnName>) ISFILTERED

This function is used to return TRUE is the columnName is filtered directly. If there are no filters present on the columns or the column selected is filtered because of a different column in this table or a related table, this function will return FALSE.

## Syntax ISFILTERED (<columnName> ) Parameters

Sr.No.	Term & Definition
1	columnName This parameter is the name of the column present in a table, and you must remember not to use an expression.

**Return Value** This function will return the value TRUE or FALSE.

### Remarks

- Any column in the table is filtered if a filter is directly applied to the column. You can use one or more filters on a column
- The column columnName will be cross filtered if the filter is applied to a different column either in the related table or the same table

You can also use this function if you want to find a column that has been filtered directly.

## Example = ISFILTERED (Sales[Product]) KEEPFILTERS

This function is used to modify how the filters will be applied to the data while you evaluate the CALCULATETABLE or CALCULATE function.

### Syntax **KEEPFILTERS (<expression>)** Parameters

Sr.N o.	Term & Definition
1	Expression This parameter uses any DAX expression.

**Return Value** The function will not return any value.

**Remarks** This function can be used within the CALCULATE or CALCULATETABLE functions. this will allow you to override the standard behavior of these functions. If you use this function, the existing filters in the current column will be used in context with the other columns that use a similar first. The intersection of these values will be used to evaluate the expression. You must remember that both the sets of arguments will apply:

- The filters in the arguments from the KEEPFILTER function
- The filter arguments used in the CALCULATE function

When the CALCULATE filter replaces the context filters, you can use the KEEPFILTERS function to add the filters based on the context.

### **Example = SUMX (**

```

CALCULATETABLE (East_Sales, FILTER(East_Sales,East_Sales[Product] =
[Product]), KEEPFILTERS(East_Sales[Product]<>"Soap")), East_Sales[Sales
Amount]

```

)

### **RELATED**

This function will return the value from another table that is related to the value in the current column.

### Syntax **RELATED (<column>)** Parameters

Sr.N o.	Parameter & Description
1	column This parameter will take the value of the column that contains the values that you need to retrieve using this function .

**Return Value** This function will return a single value that is related to the current row.

**Remarks** This function will require that a relationship must exist between any related table and the current table based on the information required. You will need to specify those columns that contain the data that you need. This function will follow an existing relationship to help you fetch those values from any related table from the specified column. If this function is used in a LOOKUP function, it will go through all the values in the selected table regardless of the different filters that could have been applied. This function will need a row context. You can use this function in either of the following cases:

- A nested function within the expression which uses a DAX X function like COUNTX or SUMX
- Within an expression used to calculate a column where the current row being used is unambiguous

**Example** = SUMX (FILTER (Sales, RELATED (Products[Product]) <>"Soap"), Sales[Sales Amount])  
**RELATEDTABLE**

This function will evaluate the table expression based on given filters in the current context .

### **Syntax RELATEDTABLE (<tableName>) Parameters**

<b>Sr.No.</b>	<b>Parameter &amp; Description</b>
1	tableName This parameter will list the name of the table that you want to use for this function. You cannot use an expression in this.

**Return Value** This function will return a value of tables.

**Remarks** This function will change how the data is filtered, and the contexts used. This function will also evaluate the expression in the context that you define. This function is similar to the CALCULATETABLE function, but you do not use any logical expressions.

**Example** = SUMX (RELATEDTABLE (East\_Sales),East\_Sales[Sales Amount])  
**USERELATIONSHIP**

This function specifies the relationship that will be used in a calculation. This relationship can exist between two or more columns .

**Syntax**                    **USERELATIONSHIP**                    (**<columnName1>**,  
**<columnName2>**) **Parameters**

<b>Sr.No.</b>	<b>Parameter &amp; Description</b>
1	columnName1 This parameter will list the name of the column that will represent the many sides of relationships that can be used. If you list the parameters in the incorrect order, the function will swap these out before it uses the parameters. You cannot use an expression in this parameter.
2	columnName2 This parameter will list the name of the column that will represent the lookup or one side relationships that can be used. If you list the parameters in the incorrect order, the function will swap these out before it uses the parameters. You cannot use an expression in this parameter.

**Return Value** This function will not return any values. This function will only enable the user to identify the indicated relationship while the calculation is being performed .

**Remarks**

- This function is only used in DAX functions that use a filter as the parameter. For instance, you can use the CALCULATETABLE, CALCULATE, CLOSINGBALANCEMONTH, CLOSINGBALANCEYEAR, OPENINGBALANCEMONTH, CLOSINGBALANCEQUARTER, OPENINGBALANCEQUARTER, OPENINGBALANCEYEAR, TOTALQTD, TOTALYTD and TOTALMTD
- This function will use any existing relationship present in the model. You can also identify the relationship based on the columns
- When you use this function, you do not have to worry about the status of the relationship. This means that it does not matter if the relationship is active or not. This will not affect the usage of the function. The relationship will be used even if the relationship is inactive. The function will override other active relationships that can be present in this model, but not mentioned in this parameter
- You will receive an error if the columns named as the parameter are not a part of the relationship or if they belong to other relationships

- If you need to use multiple relationships to join tables in any calculation, the relationship should be indicated using the function USERELATIONSHIP
- If you use this function as a nested part of the CALCULATE function, and you use numerous CALCULATE functions with a USERELATIONSHIP function, then the innermost function will be used in case of any ambiguity
- You can use at most ten USERELATIONSHIP functions in a nested function. Your expression will, however, have a deep level of nesting

**Example Product Sales:= CALCULATE (**

**SUM (Sales[Sales Amount]), USERELATIONSHIP (Sales[Product],Products[Product]) )**

**VALUES**

This function will return a table with only one column. This column will contain the distinct values from a specific column or table. In simple words, all the duplicate values are removed from this table, and only unique values are returned.

**Syntax VALUES (<TableNameOrColumnName> ) Parameters**

Sr.No.	Parameter & Description
1	TableNameOrColumnName This parameter is either the name of a table or column from where you want to return unique values.

**Return Value This function will return a column that contains unique values alone.**

**Remarks This function can be used as an intermediate function. It can be nested within a different function to help you obtain the list of distinct values which can be counted. You can also use the function to filter the values in a table or calculate the sum of other values. If you use this function in a context where the column has been filtered, like in a Pivot Table, the unique values returned will be affected by these filters.**

**Example = COUNTROWS (VALUES (Sales[Salesperson ID]))**  
**This function will return the number of rows in the Sales table, where there are unique Salesperson IDs.**

## Chapter Fourteen: Time Intelligence Functions

The Time Intelligence functions in DAX will allow you to create the right calculations that will support the needs of any business intelligence analysis. These functions will enable you to manipulate data in the table using periods like days, weeks, months, quarters, or years.

### CLOSINGBALANCEMONTH

This function will evaluate the expression in the current context on the last day of the month.

#### Syntax

CLOSINGBALANCEMONTH (<expression> , <dates> , [<filter>])

#### Parameters

Sr.No.	Parameter & Description
1	expression This parameter is used to enter an expression that will return a scalar value.
2	dates This parameter will return a column that contains the dates.
3	filter This parameter is optional and is an expression that is used to specify the filter that you will apply to the data based on the current context.

#### Return Value

This function will return a scalar value.

#### Remarks

You can enter one of the following to define the dates parameter:

- A Boolean expression that will define a table with a single column containing information about date and time
- A table expression that will return a table with one column consisting of dates and times
- A reference to any column containing dates and times

Some constraints on Boolean expressions are:

- You cannot use any expression that will either scan a table or return a table as a value, and this includes aggregate functions
- You cannot use the CALCULATE function in this expression
- You cannot use an expression using a calculated field

Boolean expressions can, however, use functions that will either calculate a scalar value or look up a single value. You can either use a table expression or Boolean expression in the filter parameter. This will define the filter that needs to be used. If you filter the data, the context will change because of the function. This will happen since the data will be filtered differently. The function will also evaluate the expression in a new manner that you specify. For every column that you use in the filter parameter, you should remove the existing columns and apply the filter parameter instead.

### Example

```
Month End Inventory Value:= CLOSINGBALANCEMONTH (
    SUMX (ProductInventory, [UnitsBalance]*
    [UnitCost]),ProductInventory[ InventoryDate]
)
```

### CLOSINGBALANCEQUARTER

This function is used to evaluate the expression in the current context one the last date of the quarter.

### Syntax

CLOSINGBALANCEQUARTER (<expression> , <dates> , [<filter>])

### Parameters

Sr.N o.	Parameter & Description
1	expression This parameter is the expression that will return a single scalar value.
2	dates This parameter will contain the dates you are looking for.
3	filter This parameter is optional, and you can include an expression that will allow you to specify the filter you can use in the current situation.

### Return Value

This function will return the scalar value that will represent the expression that has been evaluated. This expression will be evaluated based on the last date of the quarter in the current context.

### Remarks

You can enter one of the following to define the dates parameter:

- A Boolean expression that will define a table with a single column containing information about date and time
- A table expression that will return a table with one column consisting of dates and times
- A reference to any column containing dates and times

Some constraints on Boolean expressions are:

- You cannot use any expression that will either scan a table or return a table as a value, and this includes aggregate functions
- You cannot use the CALCULATE function in this expression
- You cannot use an expression using a calculated field

Boolean expressions can, however, use functions that will either calculate a scalar value or look up a single value. You can either use a table expression or Boolean expression in the filter parameter. This will define the filter that needs to be used. If you filter the data, the context will change because of the function. This will happen since the data will be filtered differently. The function will also evaluate the expression in a new manner that you specify. For every column that you use in the filter parameter, you should remove the existing columns and apply the filter parameter instead.

### Example

```
Quarter End Inventory Value:=CLOSINGBALANCEQUARTER (
    SUMX (ProductInventory,[ UnitsBalance]*
[UnitCost]),ProductInventory[ InventoryDate]
)
```

### CLOSINGBALANCEYEAR

This function will evaluate an expression in the current context on the last day of the year.

### Syntax

```
CLOSINGBALANCEYEAR (<expression> , <dates> , [<filter>],
[<year_end_date>])
```

### Parameters

Sr.N o.	Parameter & Description



1	<p>expression</p> <p>This parameter is an expression that will return a single scalar value.</p>
2	<p>dates</p> <p>This parameter will list the name of the column that contains all the dates.</p>
3	<p>filter</p> <p>This is an optional parameter, and it is an expression that will specify which filter should be applied.</p>
4	<p>year_end_date</p> <p>This is an optional parameter and is a string literal with a date. This literal will define the end date in the year. The default date is December 31.</p>

## Return Value

This function will return a scalar value .

## Remarks

You can enter one of the following to define the dates parameter:

- A Boolean expression that will define a table with a single column containing information about date and time
- A table expression that will return a table with one column consisting of dates and times
- A reference to any column containing dates and times

Some constraints on Boolean expressions are:

- You cannot use any expression that will either scan a table or return a table as a value, and this includes aggregate functions
- You cannot use the CALCULATE function in this expression
- You cannot use an expression using a calculated field

Boolean expressions can, however, use functions that will either calculate a scalar value or look up a single value. You can either use a table expression or Boolean expression in the filter parameter. This will define the filter that needs to be used. If you filter the data, the context will change because of the function. This will happen since the data will be filtered differently. The function will also evaluate the expression in a new manner that you specify. For every column that you use in the filter parameter, you should remove the existing columns and apply the filter parameter instead. The year\_end\_date parameter is a string literal that represents the date. This parameter is found

where the workbook is created. During the debugging, the compiler will ignore the date section of the year.

### Example

```
Year End Inventory Value:= CLOSINGBALANCEYEAR (  
    SUMX (ProductInventory, [UnitsBalance]*  
    [UnitCost]),ProductInventory[InventoryDate]  
)
```

### DATEADD

This function will return a table that will contain a single column with dates. These dates will either be shifted backward or forward in time.

### Syntax

DATEADD (<dates> , <number\_of\_intervals> , <interval>)

### Parameters

Sr.N o.	Parameter & Description
1	dates This parameter will take a column that contains a list of dates.
2	number_of_intervals This parameter will contain a column that will contain a list of dates.
3	Interval This parameter will use an interval by which the dates should be shifted. The value can be any of the following: <ul style="list-style-type: none"><li>▪ Year</li><li>▪ Quarter</li><li>▪ Month</li><li>▪ Day</li></ul>

### Return Value

The function will return a table that will contain one column with dates.

### Remarks

You can enter one of the following to define the dates parameter:

- A Boolean expression that will define a table with a single column containing information about date and time

- A table expression that will return a table with one column consisting of dates and times
- A reference to any column containing dates and times

Some constraints on Boolean expressions are:

- You cannot use any expression that will either scan a table or return a table as a value, and this includes aggregate functions
- You cannot use the CALCULATE function in this expression
- You cannot use an expression using a calculated field

Boolean expressions can, however, use functions that will either calculate a scalar value or look up a single value. You can either use a table expression or Boolean expression in the filter parameter. This will define the filter that needs to be used. If you filter the data, the context will change because of the function. This will happen since the data will be filtered differently. The function will also evaluate the expression in a new manner that you specify. For every column that you use in the filter parameter, you should remove the existing columns and apply the filter parameter instead. If the parameter `number_of_intervals` is positive, the dates will move forward in time and will be shifted back if the number is negative. The interval parameter is not a set of strings but an enumeration. It is for this reason that you cannot enclose the values in these intervals in quotation marks. The values, `day`, `month`, `quarter`, or `year`, should always be written in full. This will result in the return of a table that will only include the dates specified by the parameter dates.

### Example

```
= DATEADD ( ProductInventory[InventoryDate],1, YEAR )
```

### DATESBETWEEN

This function will return a table with one column that will contain the dates that either begins with the parameter `start_date` and continue to the `end_date`.

### Syntax

```
DATESBETWEEN (<dates> , <start_date> , <end_date> )
```

### Parameters

Sr.N o.	Parameter & Description
1	dates This parameter is a column that contains dates.
2	start_date A date expression.
3	end_date

A date expression.

## Return Value

This function will return a table that contains one column with dates.

## Remarks

- The start\_date parameter is the earliest value in the date column
- The end\_date parameter is the latest or last value in the date column
- These date parameters are both inclusive. For example, let us assume that you specify November 1 and December 31 as the start and end dates, respectively. You have some sales that take place on November 1 and December 31; therefore, these sales will also be included since the start and end date parameters are inclusive.

## Example

```
= CALCULATE (  
    SUM (Sales[Sales Amount]), DATESBETWEEN (Sales[Date], DATE  
(2015,1,1), DATE (2015,3,31))  
)
```

## DATESINPERIOD

This function will return a table with one column that contains dates. These dates will begin with the start\_date and will include all dates with an interval that is specified by the parameter number\_of\_intervals.

## Syntax

```
DATESINPERIOD (<dates>, <start_date>, <number_of_intervals>,  
<interval>)
```

## Parameters

Sr.No.	Parameter & Description
1	dates This parameter is a column with dates.
2	start_date A date expression.
3	number_of_intervals This is an integer that specifies how many intervals should be added or taken away from the dates.

4	<p>interval</p> <p>This parameter is an interval using which the dates will be shifted. The value of this interval may be any of these:</p> <ul style="list-style-type: none"><li>▪ year</li><li>▪ quarter</li><li>▪ month</li><li>▪ day</li></ul>
---	--

## Return Value

This function will return one column in a table with dates.

## Remarks

You can enter one of the following to define the dates parameter:

- A Boolean expression that will define a table with a single column containing information about date and time
- A table expression that will return a table with one column consisting of dates and times
- A reference to any column containing dates and times

Boolean expressions are constrained in these ways:

- You cannot use any expression that will either scan a table or return a table as a value, and this includes aggregate functions
- You cannot use the CALCULATE function in this expression
- You cannot use an expression using a calculated field

Boolean expressions can, however, use functions that will either calculate a scalar value or look up a single value. You can either use a table expression or Boolean expression in the filter parameter. This will define the filter that needs to be used. If you filter the data, the context will change because of the function. This will happen since the data will be filtered differently. The function will also evaluate the expression in a new manner that you specify. For every column that you use in the filter parameter, you should remove the existing columns and apply the filter parameter instead. If the parameter `number_of_intervals` is positive, the dates will move forward in time and will be shifted back if negative. The interval parameter is not a set of strings but an enumeration. It is for this reason that you cannot enclose the values in these intervals in quote marks. The values, `day`, `month`, `quarter`, or `year`, should always be written in full. This will result in the return of a table that will only include the dates specified by the parameter `dates`.

## Example

= CALCULATE (

```
SUM (Sales [Sales Amount]),  
DATESINPERIOD (Sales[Date], DATE (2015,1,1),3, MONTH)  
)
```

Some similar functions are:

- DATESMTD
- DATESQTD
- DATESYTD

## **ENDOFMONTH**

This function will return the last date of the month that is in the context of the current situation for a specific set of dates.

### **Syntax**

ENDOFMONTH (< dates >)

### **Parameters**

<b>Sr.No.</b>	<b>Parameter &amp; Description</b>
1	dates This parameter will take a column of dates.

### **Return Value**

This function will return a table that contains a single row and column with a date.

### **Remarks**

You can enter one of the following to define the dates parameter:

- A Boolean expression that will define a table with a single column containing information about date and time
- A table expression that will return a table with one column consisting of dates and times
- A reference to any column containing dates and times

Some constraints on Boolean expressions are:

- You cannot use any expression that will either scan a table or return a table as a value, and this includes aggregate functions
- You cannot use the CALCULATE function in this expression
- You cannot use an expression using a calculated field

Boolean expressions can, however, use functions that will either calculate a scalar value or look up a single value. You can either use a table expression

or Boolean expression in the filter parameter. This will define the filter that needs to be used. If you filter the data, the context will change because of the function. This will happen since the data will be filtered differently. The function will also evaluate the expression in a new manner that you specify. For every column that you use in the filter parameter, you should remove the existing columns and apply the filter parameter instead.

### Example

= ENDOFMONTH (Sales [Date])

Similar functions are:

- ENDOFQUARTER
- ENDOFYEAR

### FIRSTDATE

This function will return the first date from a column of dates present in a table in the current context.

### Syntax

FIRSTDATE (<dates>)

### Parameters

Sr.N o.	Parameter & Description
1	dates This parameter is a column that will contain the dates.

### Return Value

This function will return a table with one row and column with a value that is a date.

### Remarks

You can enter one of the following to define the dates parameter:

- A Boolean expression that will define a table with a single column containing information about date and time
- A table expression that will return a table with one column consisting of dates and times
- A reference to any column containing dates and times

Some constraints on Boolean expressions are:

- You cannot use any expression that will either scan a table or return a table as a value, and this includes aggregate functions

- You cannot use the CALCULATE function in this expression
- You cannot use an expression using a calculated field

Boolean expressions can, however, use functions that will either calculate a scalar value or look up a single value. You can either use a table expression or Boolean expression in the filter parameter. If the current context is only one date, the dates returned by this function and the LASTDATE function is the same. This function will return a single row and column with the date in it. This value can be used as a parameter in other functions that will require a parameter as a table.

### Example

= FIRSTDATE (Sales [Date])

The LASTDATE function is similar to this function.

### FIRSTNONBLANK

This function will return the date in the column that is filtered based on the current context. The value will only be returned when the expression is not blank.

### Syntax

FIRSTNONBLANK (<column> , <expression> )

### Parameters

Sr.N o.	Parameter & Description
1	column This parameter is an expression to return a column.
2	expression This parameter is an expression that will evaluate a blank cell in the column entered.

### Return Value

This function will return a value that contains one row, a column, and a non-blank value.

### Remarks

You can enter one of the following to define the dates parameter:

- A Boolean expression that will define a table with a single column containing information about date and time
- A table expression that will return a table with one column consisting of dates and times



- A reference to any column containing dates and times

Some constraints on Boolean expressions are:

- You cannot use any expression that will either scan a table or return a table as a value, and this includes aggregate functions
- You cannot use the CALCULATE function in this expression
- You cannot use an expression using a calculated field

Boolean expressions can, however, use functions that will either calculate a scalar value or look up a single value. You can either use a table expression or Boolean expression in the column parameter, which will return a scalar value.

### **Example**

= FIRSTNONBLANK (Sales [Sales Amount], MIN (Sales [Date])> 3/31/2015)

The LASTNONBLANK is a similar function.

# **Chapter Fifteen: Tips to Using Power BI**

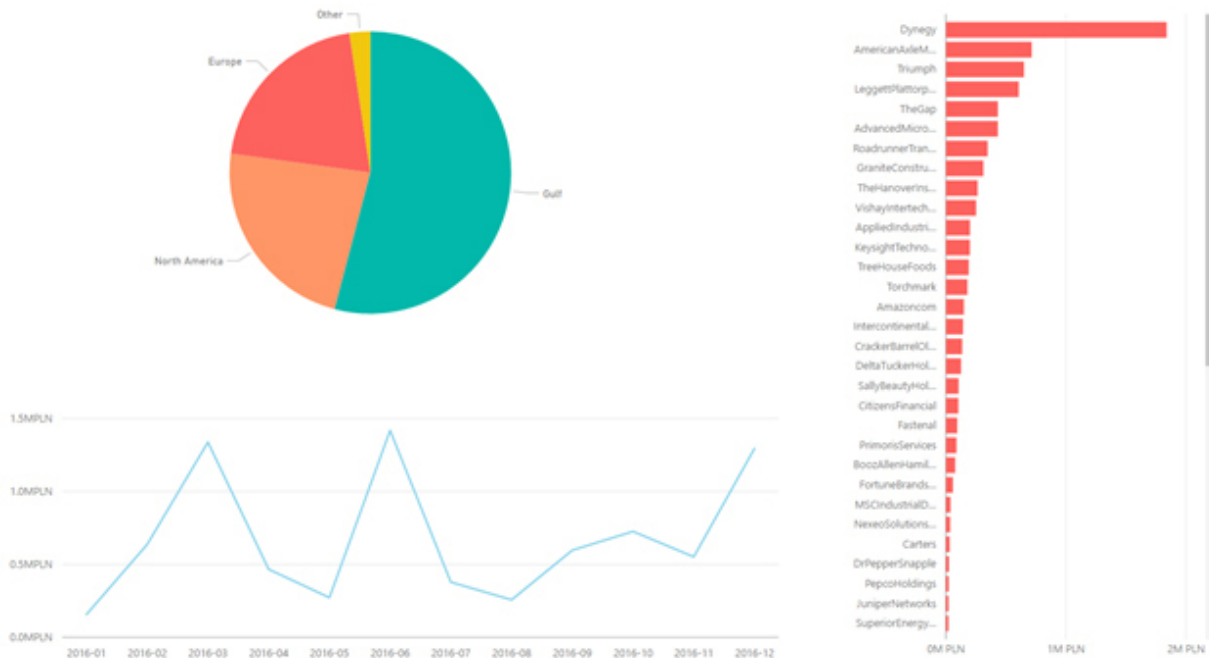
Now that we have covered the basics of Power BI and also gathered some information about DAX let us look at some tips that will help you work better with Power BI.

## **Keep the Visualization Simple**

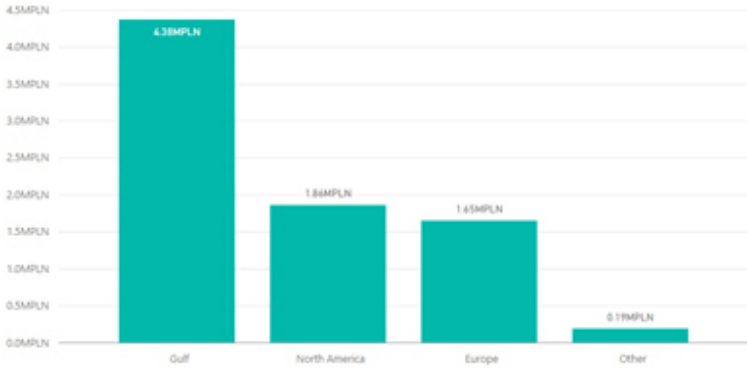
Through Power BI, you can deliver a message about the data. As mentioned earlier, there are numerous visualization tools you can use in Power BI. These visualization tools are present in the gallery. Some of these are very complex, and they can show you the relationship between data variables in a way that will make sense to you.

## Selecting the Right Chart

It is easy for an analyst to know which chart they need to select to explain the data. They can also look at complex charts and understand what is being said. When it comes to the other section of people, it is all about keeping things simple, clear, and easy to understand. So, you need to focus on simplicity. You can use a simple line or bar chart in most cases, and you can use ugly tables if needed. These are the best way to represent any raw data. This is sometimes all you need to do. For instance, some people avoid using treemaps and pie charts for the simplest of reasons - they cannot see any difference in the sections or fields where there are similar values. Let us look at the following example. We are trying to report the volume of sales per region. Look at the image below and check whether orange or red is bigger. See if you can find a difference.



If you look at the image above, you cannot find a difference in the fields representing North America (orange) and Europe (red). Now, look at the image below.



This is clearer, isn't it? This report is showing you exactly what the pie chart did, except that we have converted the pie fields into columns. You can also see the difference between the sales in North America (orange) and Europe (red) easily.

## Some Thumb Rules

- People often read any information from top to bottom, so you should always put the most relevant information at the top
- You can use vertical bars to display data. you should always use sorted data since it is easy to read
- You can use horizontal bars for ranking variables in a data set
- A line chart will work best for a time series tracker wherever you need to compare different series in data. You can also use single bars if needed
- You can use a line and bar chart to represent values of different kinds
- You can use bubble charts to present three different values

## **Identify the Context**

Power BI allows you to use the cross-filtering feature to understand data better. When you create charts, you can connect the data set to those charts. So, when you do filter one data set, you will see that the other data set has also been filtered. This will make it easier for you to compare the data since you can drill down and perform simple analysis.

## **Using Filters in Power BI**

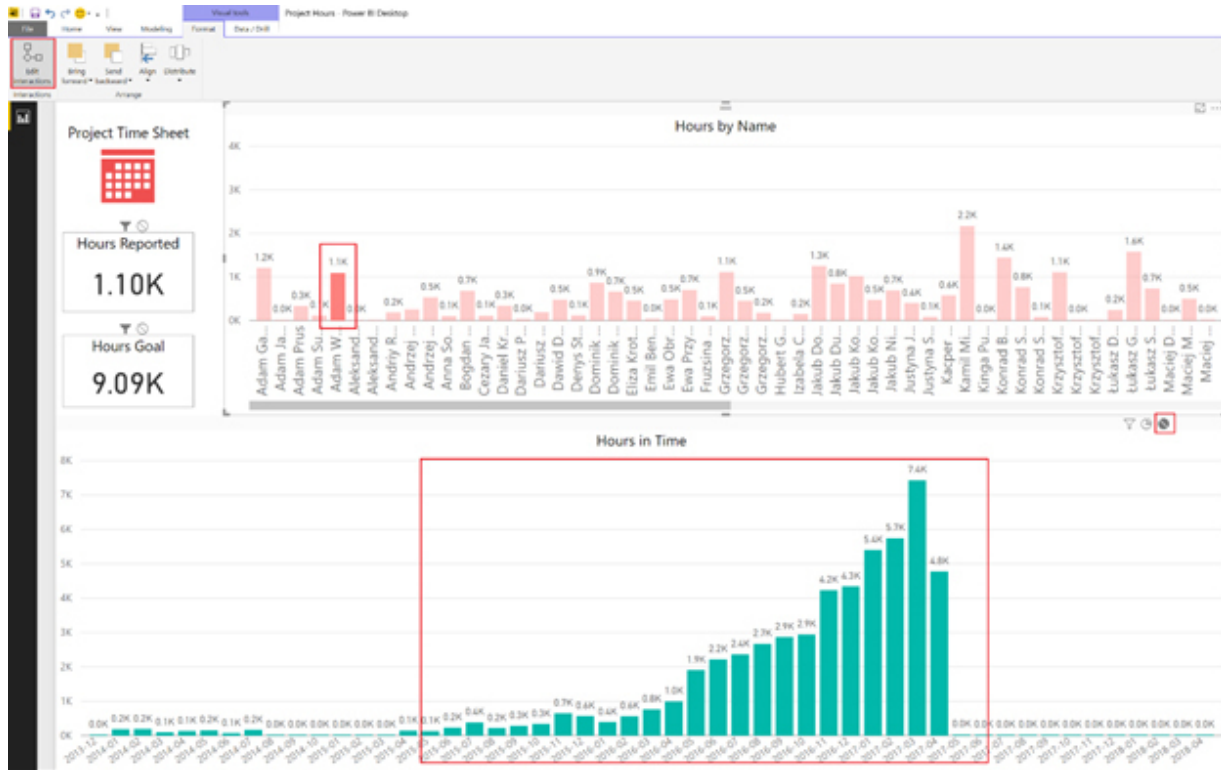
What you may not know is that there are three ways in which you can connect and filter data. This will make it easier for you to analyze data. Let us consider a project management example. You want to see the names of the people against the time reported by them and the time they have reported for every month. You can then see how the interaction will change the way the data is depicted.



# **Types of Interactions**

# None

Here you do not add any filtering to the elements in the data set. You can use this if you want to display the data to say that it is not affected by any change in the behavior of the user. In this example, you will see that you cannot see any influence data at the bottom of the chart since you do not want to look at any interactions.



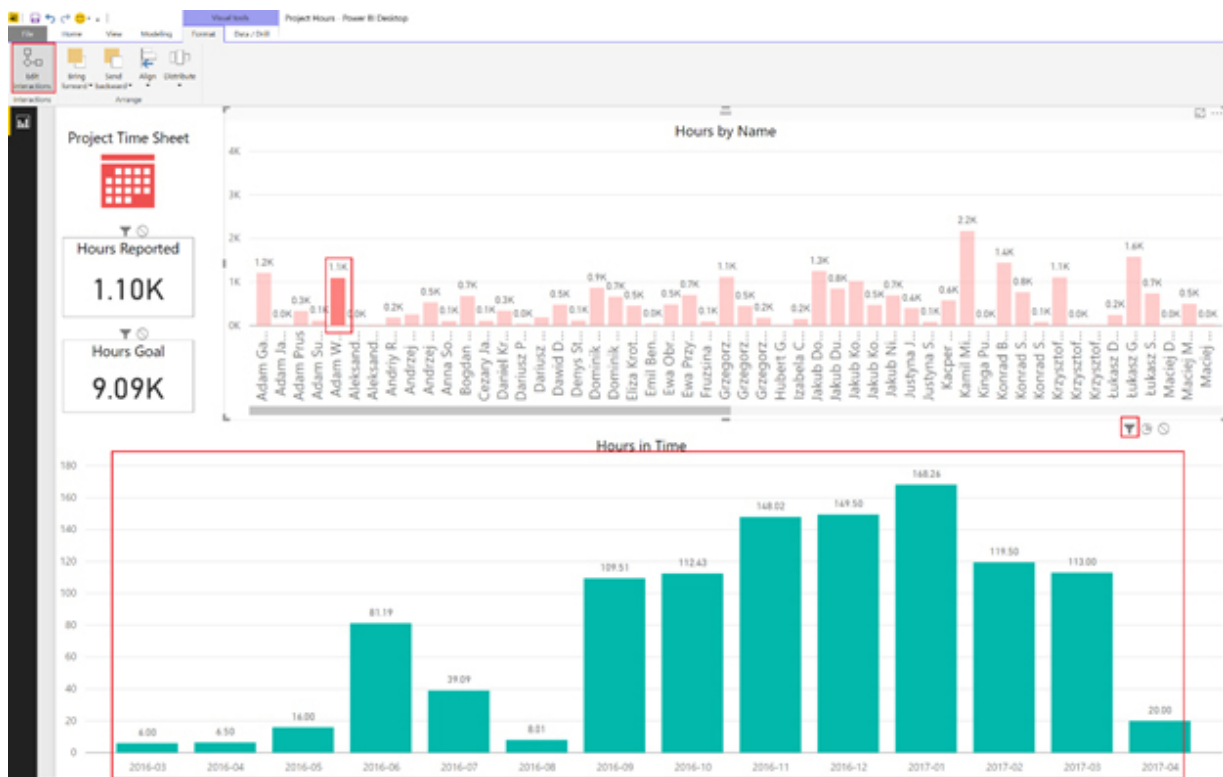
The above image shows that the data is never affected by the user's behavior. When you click on the bar at the top, it will not change the display of the bars at the bottom.

## ***Highlight***

The value that you filter on will be displayed in the form of the context. You should use this when you want to see what percentage of the total is formed by the selected element. When you click on the bar at the top, the chart will fade out at the bottom. The only part of the bar chart will be the bar that is represented by the clicked element .

## Filter

In this interaction, you will display the value that is actually filtered. You should use this only when you want to see what is hidden behind the element that you have filtered on. In this form of interaction, you only want to look at the detailed information and not how it is related to other data elements. In the example below, when you click on the bar in the chart filters for the top chart, you will see that the only data applicable in the bottom chart is for the element that you have filtered on.



You should choose the type of interaction you want to include depending on the context that you are looking at since the relationship between the data sets will have an impact on your data. If there are too many data elements, it will affect the report. You may also find it hard to create a report in Power BI .

## Slice, Dice, and Filter

This is a very basic concept of data visualization. You will still be surprised at how many different filtering options are present in Power BI. Let us look at the five most obvious ones:

# Basic Filters

### ***Visual Level Filter***

This filter will only filter the selected data at a visual level. This is a useful option to use if you want to include some background data that you can only use for filtering.

### ***Page-Level Filter***

This filter will be applied to every element present in the page.

## ***Report Level Filter***

This filter will be applied to all data elements across all pages. It is a useful filter to use if you want to see how the data works with other elements across different pages in the report. You will, however, have a different view on every page. When you select your filter and move to the next page in the report, the filter will still stay selected. This will allow you to view the data in the same context across all pages.



## ***Slicers***

A slicer is a filter that is available as a single or multiple selection in a dropdown or checkbox. Most people do not find this very useful since they take up space on the canvas. These slicers do not provide too much value addition since you can use cross-filtering to understand your data better. A slicer will only work on one page. This means that if you set up a slicer on a specific page, it will only work on that page. It is for this reason that this filtering option is limiting. When you move to a different page, you will lose the context of all the data that you worked with on the previous page.

## ***Cross-Filtering***

We have covered this in the previous point. The idea behind cross-filtering is that it can be used to look at how the data interacts with each other. It is a better idea to use this option instead of a slicer.

Let us continue with the project management example. You can use a multiple page report with different pages that will show you an overview of the number of hours or the details about the time taken by the user to complete tasks. If you use a slicer or cross-filter, you will also need to include the project that you want to look at on individual pages. If you use report filters, the project will be selected even when you move to a different page in the report. Let us assume you have seven pages in the report. Use either of these or see what is best for you.

## **Hierarchies**

A hierarchy is one of the best ways to show analysis on different levels of granularity using the same charts or tools to visualize the data. For example, if you are a program manager, you may be interested in different projects and want to know how they are progressing every month. If you are a project manager, you will be interested to know what is happening every week or every day. You want to pay close attention to what is happening. You can always create different reports for a project or program manager. You will, however, need to support a large number of these cases. If you are clever, you will want to design a report in a way that it can be used by both a program and project manager. This is where a hierarchy comes into the picture.

## Using Hierarchies in Power BI

You can use hierarchies in Power BI in three ways:

- These hierarchies can directly come from the data source, which means that the hierarchy is present in the data set
- These hierarchies can be based either on time or date data. In Power BI, you can represent the data in terms of days, months, or years. We have looked at the various DAX functions you can use in the previous chapter
- You can add more dimensions to the data set and to the visualization. This will not make these dimensions visible, but it will allow you to drill down from one to the other

When you have a chart ready, you should look at the small arrows at the corner of the chart, which will enable you to go through the hierarchy levels.

You can use this report and visualization to understand the data from different views. Since one can create reports quickly in Power BI, you will be tempted to create many reports since you can do this easily. You should, however, think of those users who will use these reports. Think about how confused they will be when you need to obtain tons of pages or reports to show similar information.

## **Think About the Message**

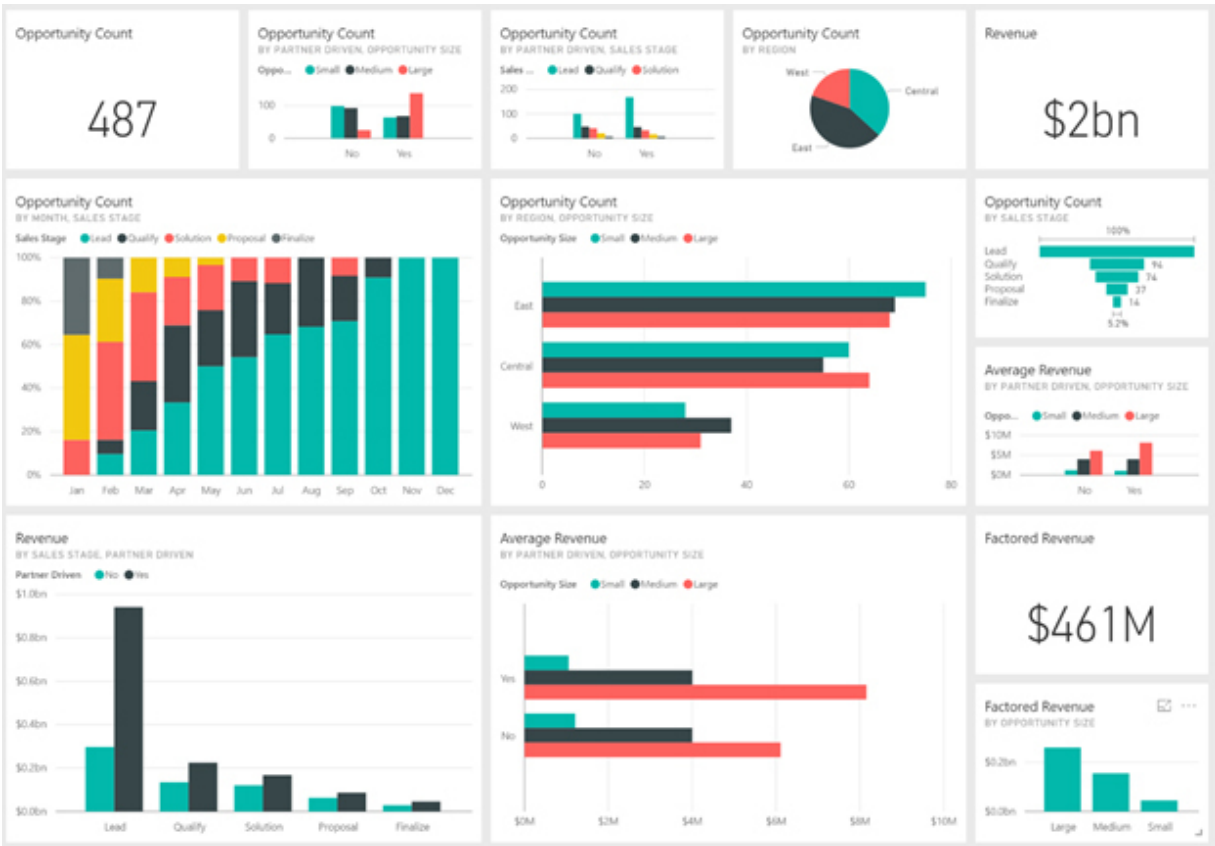
When you share the report with people and allow them to edit the report, you may end up with an analytical painting that has too many colors but absolutely no information in it. Power BI allows you to produce a variety of charts and tables to show your data set. This will make people want to create numerous reports. The reporting canvas in Power BI is much like a slide in PowerPoint. There is no pagination and scrolling. This is, however, the whole point. When you begin to work on a report in Power BI, you should look at how you can fit the charts on the reporting page and how you can visualize the information in that space. You should remember that the message should be clear and easy for anybody to digest. This is especially important when you look at the two display areas in Power BI.

## **Dashboard**

A dashboard is the primary point where you will go and work on your analysis. You cannot work with interactions or filters. A tile in the dashboard will link one report to another to present the data easily.

# Reports

A report is an analytical space with various capabilities. The purpose is to dig into the details and also understand why specific things happen using the data. Let us look at the following example shared by Microsoft.

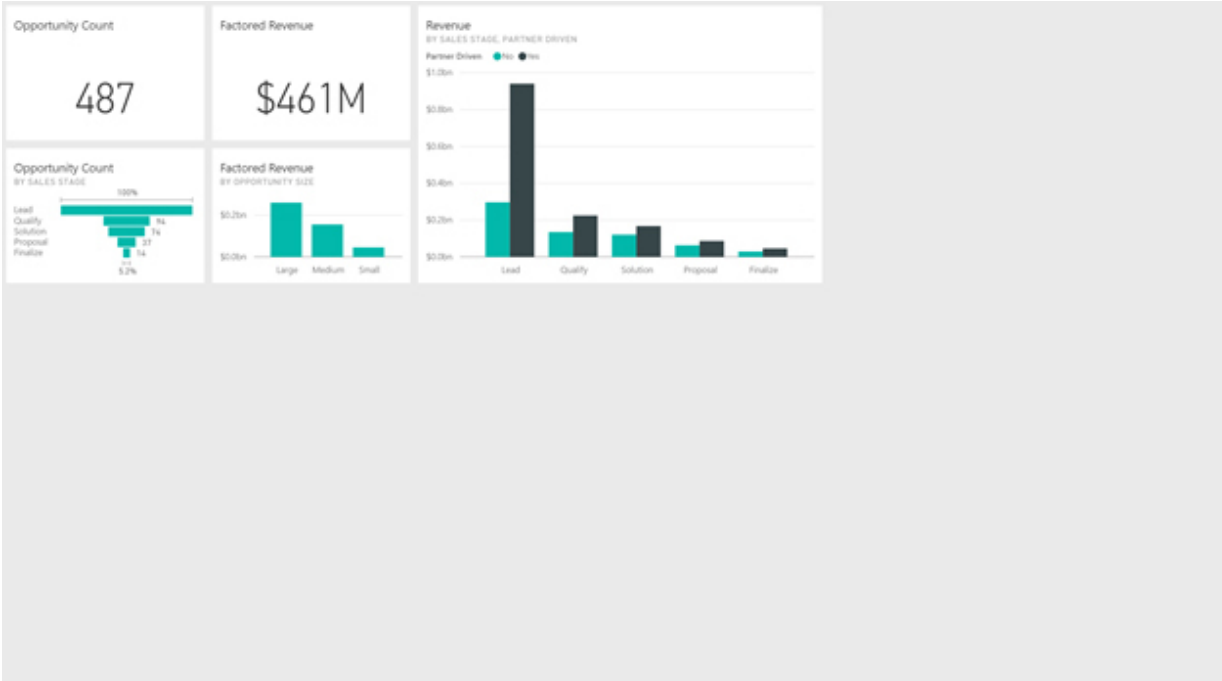


This dashboard uses some sample information about sales opportunities, and it will show the same data in numerous ways.

## What Are We Looking At?

Remember, this dashboard is only a demo dashboard that Microsoft developed. It cannot be used anywhere in particular. This is an example of bad practices. Every tile in the dashboard shows the same data from a different perspective. This will make the dashboard an analytical space and not the status space. Think about how you can

simplify this based on the main concepts - volume and actual opportunities number.



This dashboard will only show the number and volume of actual opportunities. This is the most important data that you should consider. You can not only see this data better, but you also have enough space that will allow you to add more meaningful information about the data set. If you want to learn more about the data displayed on the dashboard, you should only click on the tile to obtain the report. This will allow you to see the data in the original dashboard.

## **Conclusion**

Power BI is a data visualization tool that is used by numerous businesses across the globe. This tool has replaced data visualization tools like Tableau since it is cheaper to use. Businesses can connect to a variety of data sources and also allows you to work with the cloud. Throughout this book, you will learn everything you need to about Power BI. You will also learn more about the DAX language and how you can use it to automate the visualization process.



# References

<https://www.tutorialspoint.com/>

<https://www.digitalvidya.com/blog/introduction-to-microsoft-power-bi/>

[https://us.hitachi-solutions.com/blog/8-reasons-why-you-should-shift-reporting-from-excel-to-power-bi /](https://us.hitachi-solutions.com/blog/8-reasons-why-you-should-shift-reporting-from-excel-to-power-bi/)