# Mastering Power BI

Build Business Intelligence Applications Powered with DAX Calculations, Insightful Visualizations, Advanced BI Techniques, and Loads of Data Sources

CHANDRAISH SINHA

# *Mastering Power BI*

*Build Business Intelligence Applications Powered with DAX Calculations, Insightful Visualizations, Advanced BI Techniques, and Loads of Data Sources*

**Chandraish Sinha**

www.bpbonline.com

**FIRST EDITION 2022**

**Copyright © BPB Publications, India**

**ISBN: 978-93-91030-728**

**Dedicated to**

*Ishie*

*My daughter, who taught me that patience is a virtue.*

## *About the Author*

**Chandraish Sinha** is the Founder/President of Ohio Computer Academy, a company dedicated to IT education.

An IT trainer at heart, Chandraish resonates with his company's slogan - Inspire, Educate and Evolve. He is a Business Intelligence learner and explorer. He has implemented multiple large and medium scale BI solutions.

In his 22 years of career, Chandraish has worked with a variety of dashboarding applications such as Power BI, Tableau, Qlik View, Qlik Sense, IBM Cognos, Business Objects and Actuate.

He is passionate about data and explores applications that provide better data insights. He has also authored multiple books on Tableau and Qlik View. His Amazon author profile can be viewed at the URL

He blogs regularly on BI applications such as Power BI, Tableau and QlikView. These are the links to his blogs:

https://ohiocomputeracademy.com/category/power-bi/

https://www.learntableaupublic.com/

https://www.learnallbi.com/

LinkedIn  Profile:  [www.linkedin.com/in/chandraishsinha](www.linkedin.com/in/chandraishsinha)

## *About the Reviewer*

**Saravanan Shanmugam (Sara)** is a certified trainer for a plethora of Data Science and Data Visualization technologies viz., PowerBI, Snowflake, Tableau, Machine learning and Python. He has over 20 years experience in managing and delivering software projects. His data platform experience includes design and development of Business Intelligence Reporting solution with Power BI, Microsoft SQL Server, SharePoint, SSIS and SSAS.

Sara has a Master's degree in Business Administration with a specialization in Technology Management from Anna University and bachelor's degree in Electronics and Communication. He has worked in various implementation projects and products across the globe on functionalities like ERP, EDI, Data Analytics using Microsoft Technologies.

## *Acknowledgement*

My sincere thanks to my readers who have supported my books so far and inspired me to go on. I personally read all their reviews and feedback.

My gratitude to Saravanan Shanmugam, Technical Reviewer of this book for painstakingly going through the entire book and providing valuable suggestions.

Finally, I would like to thank BPB Publications for giving me this opportunity. Book writing involves a lot of people and thanks to them for providing all the necessary assistance.

# *Preface*

Power BI visualization is gaining popularity in the business world due to its capability in modeling and effectively presenting the data. Through the years, Power BI has evolved into a powerful suite of products.

In my career, I worked on different reporting and visualization applications such as Tableau, Qlik, IBM Cognos, Business Objects and Actuate. I feel Power BI is at par and sometimes more advantageous as compared to others.

The functionalities included in Power BI are as vast as an ocean. Power BI desktop, Query editor, Visualizations, DAX formulas and Power BI Service make the application robust but at the same time can become a little difficult to understand. In this book, an attempt has been made to introduce the readers to the basics and then take them to more deeper concepts. This book covers all aspects of creating a project in Power BI from start to finish.

As Power BI is getting adopted by more and more organizations, it is generating good career options. The purpose of writing this book is to equip readers with the essentials of Power BI in a nutshell. This book is useful for the novice IT aspirants ready to take a plunge and also for the seasoned IT professionals who want to switch their career to Power BI.

In writing this book, I have followed the same approach as observed in my other books. It is simple to follow and provides concepts followed by hands-on exercises. All the data and solution files are provided. The readers can follow the steps mentioned in the book and create their own application.

The primary challenge I faced in writing this book are the changes introduced by Power BI. Few times, the look-and-feel of the application has changed. I have tried my best to keep up with the changes, but, in case some discrepancies are found remember, the changes are only in the interface and not in the concepts. A menu item may have shifted in the ribbon but it will be there. The way the functionality is implemented has not changed and remains the same.

The primary goal of this book is to provide information and skills that are necessary to create a Power BI application. This book contains steps that will show readers how to install and create Power BI application.

**Over the 7 chapters in this book, you will learn the following:**

Chapter 1 covers the basics of Business Intelligence and explains all the important terms and definitions. It also explains different components of Power BI and what they do. It will assist in installing Power BI desktop and provides an overview of the data tables used in the book.

Chapter 2 explains how Power BI connects to data from disparate sources such as database tables, XLS files, relational database and many more. It also introduces Query editor and describes how to use it to shape the data.

Chapter 3 deals with creating and optimizing a data model in Power BI. It covers relationships and how to create them in Power BI. It also explains how to create joins by using merge and append functionalities.

Chapter 4 is a key chapter which explains Data Analysis Expression (DAX). DAX formulas are important for any Power BI implementation. This chapter covers all the main functions and shows how to implement them.

Chapter 5 visualizations are important as they help in understanding the data. This chapter covers all the visualizations available in Power BI and explains when to use them.

Chapter 6 describes components of Power BI Service. In this chapter, you will deploy the application created in the Power BI desktop to Power BI Service. This chapter also teaches how to connect to data and create visualizations in Power BI Service.

Chapter 7 introduces to the concept of Row-level security (RLS). It is important to keep the application secured and users see only the data they are authorized to view.

***Downloading the code***

***bundle and coloured images:***

Please follow the link to download the
***Code Bundle*** and the ***Coloured Images*** of the book:

[***https://rebrand.ly/5d8ca8***](https://rebrand.ly/5d8ca8)

***Errata***

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

*Table of Contents*

CHAPTER 1

# Understanding the Basics

## ***Introduction***

In this chapter, we will learn about the basics of Power BI. Power BI is growing in popularity due to the functionality it provides to the business users. This chapter will cover the basics of Power BI as a Business Intelligence application. It will start with the Business Intelligence fundamentals and explain the terms and technologies in the BI paradigm.

We will learn about Power BI and understand how it works. This chapter is important as it will lay the foundation of all the subsequent chapters in this book.

## *Structure*

In this chapter, we will discuss the following topics:

Understanding Business Intelligence (BI)?

Concepts of the Star and Snowflake schema

Power BI and its components

Installation of Power BI Desktop

Power BI Desktop Interface

Overview of the data used in the book

Folder setup for learning

## ***Objectives***

Understanding of the Business Intelligence concepts is a key to success in Power BI. After completing this chapter, you will be able to explain what is Business Intelligence and its terminology like the Star schema, and the Snowflake schema. You will also be able to differentiate between the Dimension and Fact tables, which is the key to designing powerful data models and visualizations. You will also learn what is Power BI and how it works.

## ***Understanding Business Intelligence***

Before defining the term **Business Intelligence** let’s understand the terms data and information. In the world of **Information Technology** data can be anything – text, numbers, or images in a digital format. The data is raw, unorganized, or arbitrary, but should be in a format that is understandable to a computer system. Once loaded, the data is transformed, processed, and interpreted by the system to produce meaningful and contextual information.

In the business world, data and information are closely related and thus used interchangeably.

BI relates to the set of technologies and techniques that collect and categorize an organization's data and presents meaningful information in a format that helps in better decision making. The BI applications allow the developers to collect vast amount of data from diverse sources, transform the data according to the business requirements, and present it in a visual format – tables and charts. BI does not make decisions for an enterprise, but eases the analysis of data to arrive at actionable results.

## *Advantages of a Business Intelligence (BI) system*

An enterprise can drive huge benefits by implementing a BI System, which are as follows:

**Data** BI system facilitates the collection of data from diverse sources. This data is stored in an enterprise-wide data warehouse or a data mart. Since the data is centrally stored, it helps in producing a single version of the truth.

**Information** The information is delivered in a visual format that is understandable to the users. It helps in the quick delivery of information in the form of interactive dashboards, tables, charts, and maps. The users can get to the data faster and collaborate with the information.

**Secure** BI System also supports secure information delivery, -that is, the data is contextual and is delivered on a need-to-know basis. The visualizations can be developed that display the different data based on the organization's role or organization structure.

**Adhoc** The business users can use the self-service BI applications to perform their own data analysis. Doing so will reduce the dependency on the IT technical team.

## ***Business Intelligence components***

Before plunging into a BI application like the Power BI, it is important to learn about some of the following BI components:

## ***Data sets***

The core use of a Business Intelligence application is to enhance the understanding of data. The data can come from disparate sources. The data can be sourced from legacy systems, relational database, cloud, or from various file-based applications, such as Excel, CSV, or flat files. The data can be unstructured, such as emails or webpages.

## ***Extract, Transform, and Load (ETL)***

ETL is a process of data integration and is used to combine disparate data arriving from multiple sources. In an ETL process, the data is extracted from the source, transformed to aggregate or to implement business rules, and then loaded into the target system. The data loaded in the target system is used for analysis. The ETL process is used to build a data warehouse.

## ***Data warehouse***

A data warehouse is a process of managing large amounts of data in an organization. It is designed to assist in the BI tasks, especially in analytics. Given a large amount of historical data, a data warehouse enables faster data query and analysis. A typical data warehouse contains historical data, which is derived from a variety of sources, such as operational or transactional databases.

A data warehouse works as a central repository of the aggregated data from multiple sources and provides an organization with a single version of truth. Since it contains historical records, it empowers the data scientists and data analysts in improved decision making and predictive analysis.

A typical data warehouse contains the following:

A relational database to store and manage the data. This relational database is created in any of the applications, such as Oracle, SQL Server, or db2, etc.

An ETL process to extract data from the multiple sources, transform, and aggregate the data according to the organizational needs and load the data in the data warehouse.

Data analysis and visualization applications, such Power BI to assist in the analysis of the data.

The data warehouse is a core component of a BI implementation.

## Data mart

Data mart is similar to the data warehouse but contains only the specific business data within an organization. A data warehouse is a central repository of an enterprise-wide data, while a data mart contains the subset of data pertaining to a specific business or user function. Both the data warehouse and the data marts are used for reporting and analysis. A data mart can be sourced from a data warehouse.

A typical data warehousing environment is shown in the following diagram:

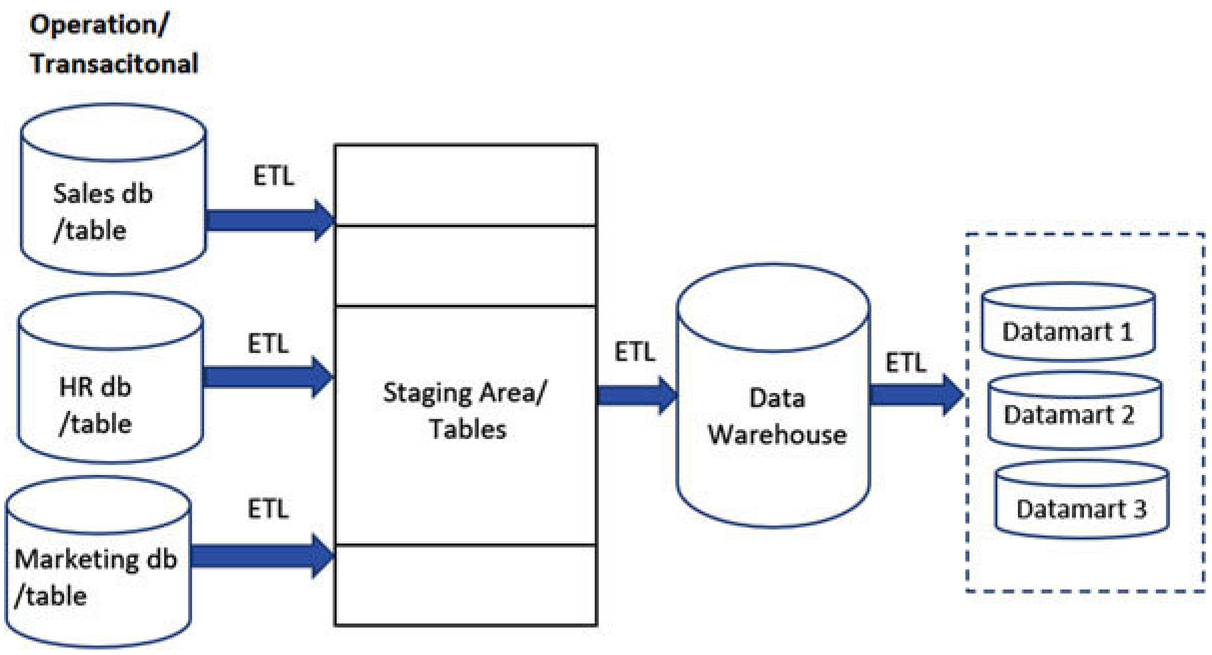


***Figure 1.1:*** *Data warehousing environment*

The preceding *figure 1.1* shows how the data warehouse and the data marts are created, and are explained in words as follows:

An ETL process is the run process to extract the data from the various operational or transactional databases or tables.

This data is stored in the staging tables.

A different ETL process is created to extract and transform this data, which is loaded in the data warehouse database.

From the data warehouse, separate business or user specific data marts are created.

A data warehouse is typically created for the reporting and analytical needs of the organization. It helps in the data analyses by reducing the number of tables and joining the ones that are atypical of an operational database.

## ***Data model***

A data model displays how the different data entities are related in a data warehouse environment. It presents a pictorial format, showing different tables and the relationships between them. In BI, a data model represents the organization's data and should be designed for a faster data access. It should contain all the data categories, hierarchies, and filters. The most popular data model used in a data warehouse is the dimensional model, which is also called the Star schema. The two kinds of tables in a star schema are the Dimension and the Facts table.

**Dimensions and facts:** Dimension and Facts tables are the main ingredient of any Business Intelligence implementation. These tables are used to form the Star or the Snowflake schemas, which are designed as part of building a data warehouse or a data mart.

**Dimension** The dimension tables contain the descriptive or qualitative attribute of the data. For example, the customer dimension may contain information about the customer, such as the name, address, contact number, and so on. The dimension fields usually contain the characters or the textual type of data. The dimension tables are constructed from the operational or transactional relational database. The dimension tables contain the primary key with the respective foreign key in the fact table. A dimension table provides context to a fact table.

There are different types of dimension tables, some of the commonly used ones are as follows:

**Slowly Changing dimensions** It is a dimension table where the row of the data in the table varies with time. It is used to track the current and historical data. SCD is implemented in the following 3 ways:

In type1 SCD, the existing row of data is simply overwritten. No history is maintained, and the existing data is lost.

For example, consider the following employee record:

| record: |
|---|
| record: |

***Table 1.1:*** *Employee record*

If employee John changes his department to HR, no history will be maintained. The record will simply be overwritten:

| overwritten: |
|---|
| overwritten: |

***Table 1.2:*** *Employee changes department*

The type2 SCD keeps the complete history of the data by creating a new record with the Start date and the End date. Only one record will be active at a time. This is the most popular way of storing the historical data.

For example, consider the following employee record:

| record: |
|---|
| record: |

***Table 1.3:*** *Employee history with Active indicator*

If employee John moves to a new department, HR, a new record will be added to keep the history:

| history: |
|---|
| history: |
| history: |

***Table 1.4:*** *Employee history with only one department Active*

In type3 SCD, the history of the data is maintained by using the **Current_Value** and the **New_Value** columns. It is cumbersome to maintain the history, as it is limited by the number of columns needed to store the historical data. This technique is not frequently used.

For example, consider the following employee record:

| record: |
|---|
| record: |

*Table 1.5: Employee history using current and previous value columns*

If employee John moves to a new department, HR, the history is maintained by putting the new department under the **Current_Value** column:

| column: |
|---|
| column: |

*Table 1.6: Employee record with changed current value*

**Conformed** A dimension table is said to be conformed if it has the same context and content when used with the different fact tables. The two conformed dimension tables will be exactly the same even if used in the different data marts. Such tables, when used in visualizations, provide a single version of the truth to the users. The time dimension is the best example of the conformed dimension, because the definition of the attributes such as year, month, quarter, etc., will be the same across the organization.

**Role playing** A single dimension table can be joined multiple times to a fact table. This can be done by creating multiple copies of the dimension. These copies of the dimension tables can connect to the fact table based on the context. A good example of the role playing dimension will be the time dimension, which will join with the customer order table to get the order date, ship date, and delivery date.

**Fact** The fact table contains the foreign keys of all the dimension tables. It stores the measurable or the quantitative attribute of the data. For example, a fact table may contain products purchased by the customer. The fact fields are the metric fields and generally contain data of the type number. You can aggregate on the fact fields, such as sum (sales).

Apart from storing measures, some of the other common fact tables are as follows:

**Fact-less-fact** This fact table contains only the foreign keys of the dimension tables and does not contain any measure values.

**Conformed fact** Similar to the conformed dimensions, the conformed fact tables are used across multiple dimension models.

The dimension and fact tables are the basis of a star-schema. Such tables are designed by consolidating multiple tables from an operational database.

## Star schema

Star schemas are created in a data warehouse and data mart environments. It consists of the facts and dimension tables. The shape of a star schema is such that the fact table is in the middle, surrounded by multiple dimension tables. The schema assumes the shape of a star and hence the name.

A star schema supports querying huge amount of data stored in a typical data warehouse storage system. The queries run against the star schema are faster due to the reduced number of joins used to query the data.

An example of a star schema is as follows:

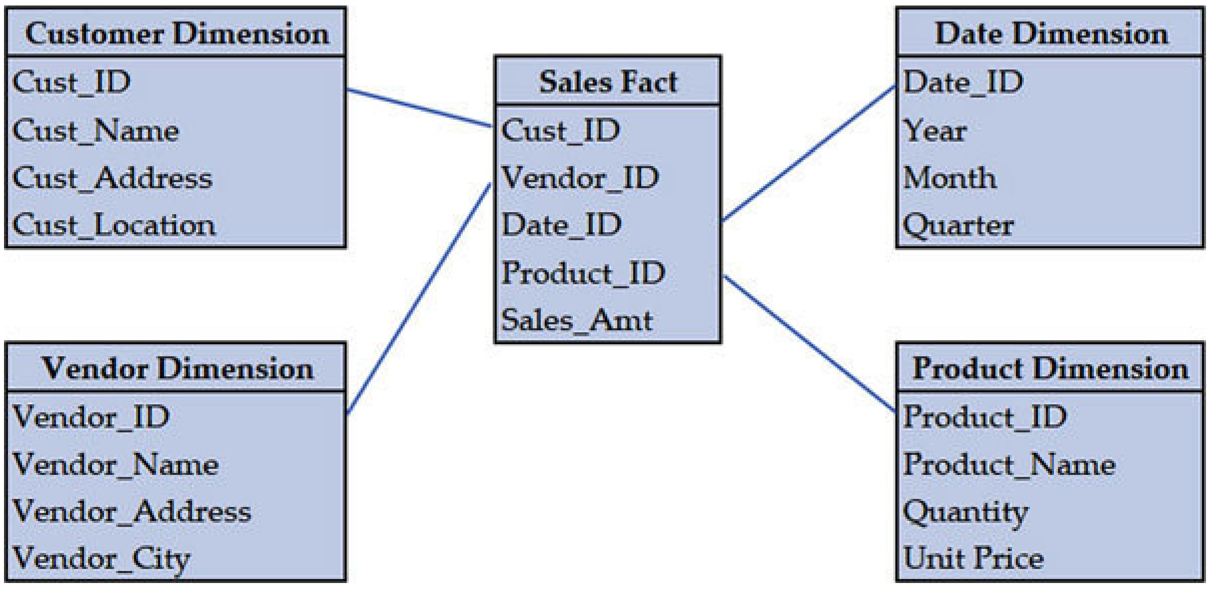


***Figure 1.2:*** *Star schema*

Since the fact table contains the foreign keys of the dimensions tables, during the ETL process, the dimension tables are loaded before the fact tables.

## Snowflake schema

A snowflake schema is also used in the data warehousing and data marts. It is an extension of the star schema. In a star schema, each dimension is stored in a single dimension table, whereas in a snowflake schema, a dimension explodes or has a lookup table. This further extension of the dimension tables gives a picture of a snowflake, and hence the name. In the following diagram, the product dimension is connected to another dimension table,

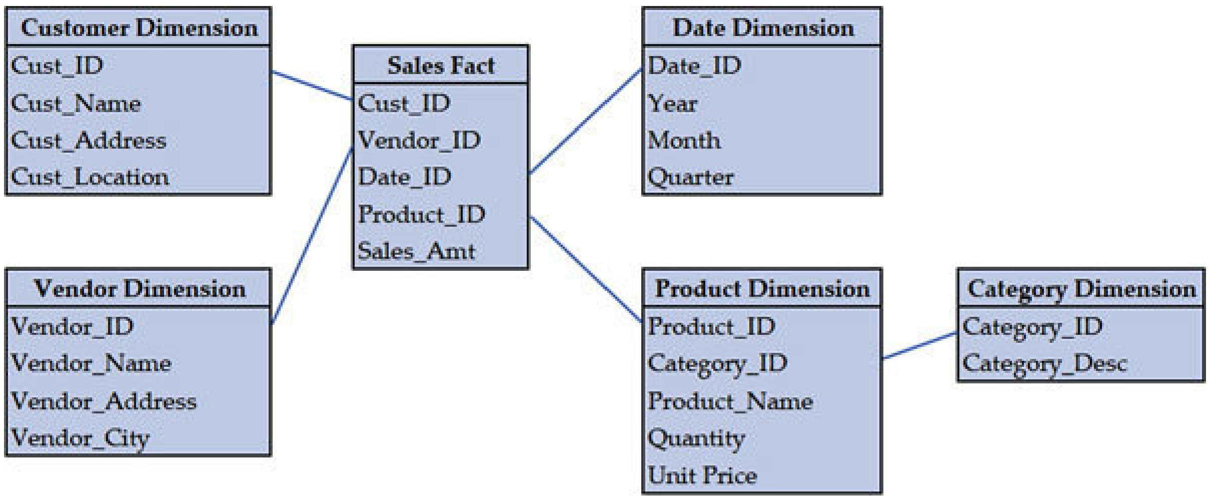


***Figure 1.3:*** *Snowflake schema*

In the Power BI, the developer connects to a data source and creates a data model. Care should be taken to create a data model close to a star schema or a snowflake schema.

## *Key Performance Indicator (KPI)*

**Key Performance Indicators (KPI's)** are performance pointers that depict the overall health of an organization. A good example of KPIs can be of a car; before driving a car, you want to check all the indicators, such as air pressure, fluids, heat, brakes etc., to ensure a smooth journey.

The business executives gauge the KPI's to track the health of the company, and base their business decisions on it. KPI's may be different for each business unit; for example, for a Sales department, the KPI's may be Product Sales Amount, the variance between Actual and Budgeted amounts, etc.

KPIs play an important role in Business Intelligence. The executives at various levels are usually interested in knowing the KPI of their departments and sections. KPI's can be displayed in the form of visualizations and dashboards.

## *Visualization*

Human brain understands pictures faster than text. Visualization represents data in a pictorial format and aids in faster surfing and consumption of data. Interactive visualizations are created in the form of tables, charts, and maps. Visualization helps the users in understanding complex data. Some of the examples of visualizations are bar, line, or pie charts.

The following is an example of a visualization displaying the different categories in color for a better understanding:

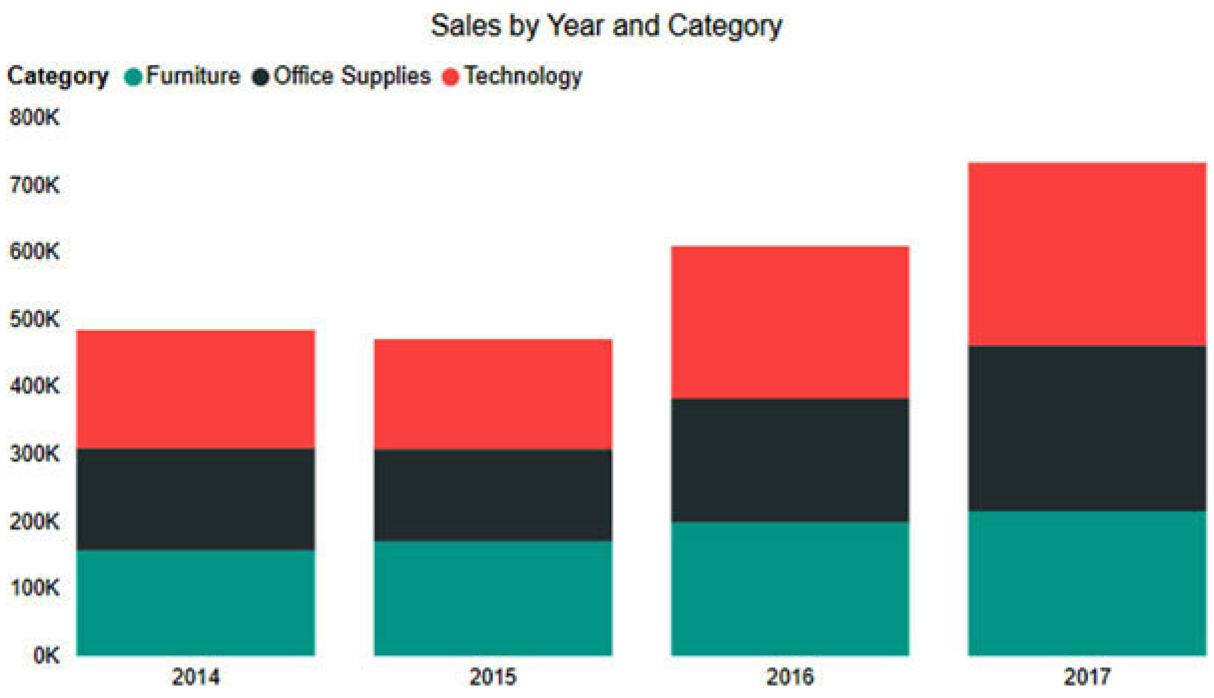


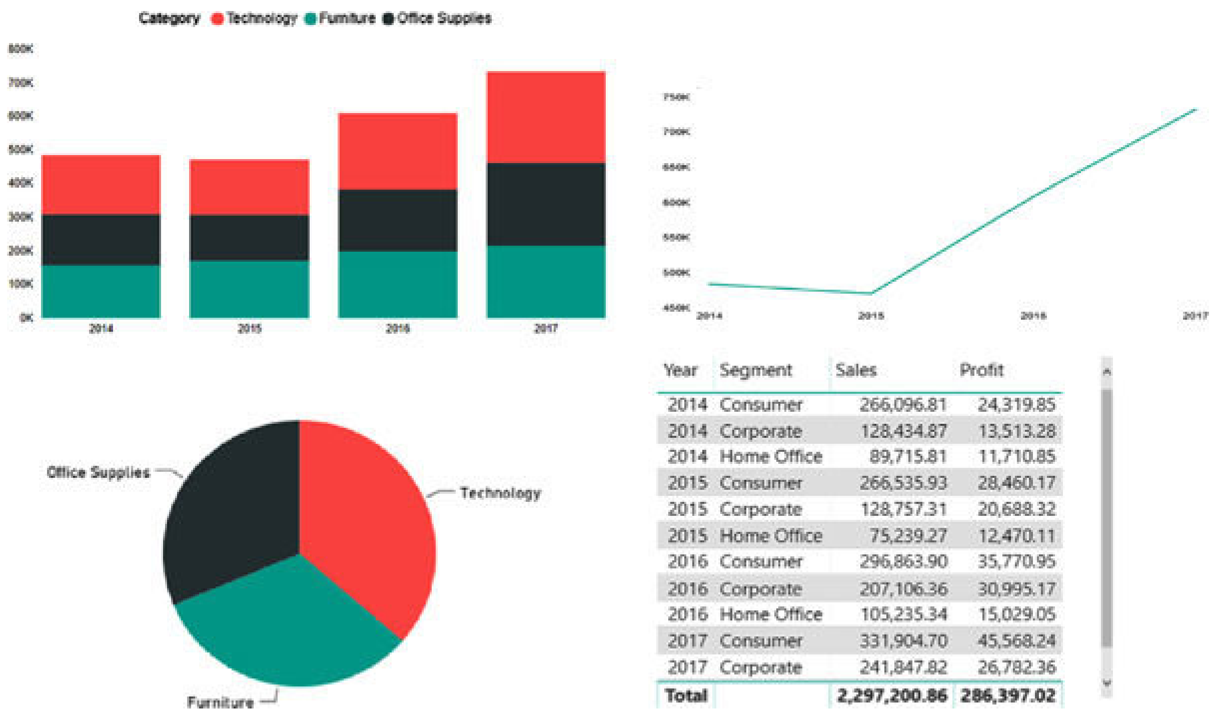
***Figure 1.4:*** *Stack chart*

The preceding chart is a stack chart for the **Sales by Year and** In the absence of this chart, the users would have to spend considerable amount of time in going through the several data sheets to arrive at this data.

## Dashboard

A data dashboard can be compared to a car's dashboard. It is a visual snapshot of data. It is usually a combination of one or more visualization used to display the key performance indicators of a company. The visualizations in a dashboard are usually related and display a key metric of the company.

The following is an example of a dashboard; the sales of the organization is displayed in the form of a stack chart, trend chart, pie chart, and table:



| Year | Segment | Sales | Profit |
|---|---|---|---|
| 2014 | Consumer | 266,096.81 | 24,319.85 |
| 2014 | Corporate | 128,434.87 | 13,513.28 |
| 2014 | Home Office | 89,715.81 | 11,710.85 |
| 2015 | Consumer | 266,535.93 | 28,460.17 |
| 2015 | Corporate | 128,757.31 | 20,688.32 |
| 2015 | Home Office | 75,239.27 | 12,470.11 |
| 2016 | Consumer | 296,863.90 | 35,770.95 |
| 2016 | Corporate | 207,106.36 | 30,995.17 |
| 2016 | Home Office | 105,235.34 | 15,029.05 |
| 2017 | Consumer | 331,904.70 | 45,568.24 |
| 2017 | Corporate | 241,847.82 | 26,782.36 |
| **Total** | | **2,297,200.86** | **286,397.02** |

***Figure 1.5:*** *A typical dashboard*

Visualizations give better insights to the users, but care should be taken while creating a visualization. A visualization should be data centric and should represent meaningful and contextual data.

## ***Power BI as a Business Intelligence application***

Microsoft Power BI is a Business Intelligence application. It is a suite of tools designed for a better understanding of data. These tools help in extracting the data from multiple sources, shaping or transforming the data according to the business requirement, and presenting the data in the form of interactive graphs, tables, maps, and dashboards.

## ***Functions of Power BI***

Power BI can do the following:

Connect and extract large and diverse data sources. Power BI compresses the data and hence is able to load data faster.

Powerful data transformation capabilities facilitate in creating a robust data model. The tables can be joined based on the common fields and various cardinalities. Data can be combined from different sources, such as Excel, CSV, or a database.

Power BI can connect to the cloud-base or on-premises data.

The data model is used to create interactive reports and dashboards. These reports and dashboards contain powerful visualizations. Power BI has an array of charts to represent the data.

The Power BI interface is intuitive and equips the business users to gain better insights of data by creating their own aggregations and visualizations.

Power BI reports and visualizations can be shared with the other users.

Using row-level security, the reports in Power BI can be secured.

## ***Power BI components***

Power BI comes with a collection of different applications. These applications have evolved over time and become useful, as more functionalities were added to them. Some of these components were integrated to excel as an add-on, but now they are also available with the Power BI desktop. The following are the core components of Power BI:

**Power BI** The Power BI desktop is an interface that interacts with all the other tools in the Power BI environment. It is an authoring tool and is used to connect and shape the data, write powerful code for calculations, and create reports containing visualizations. It uses the ribbon as a navigational menu to navigate to different available tools. The Power BI desktop contains tabs such as Home, Insert, Modeling and View.

**Power** Power Query is used for connecting and preparing the data to create a dashboarding application. It enables the users to connect and combine the data from hundreds of data sources. It can be invoked using the **Get Data** option in the Power BI desktop ribbon.

Get Data is used to connect and load the source data, as shown in the following screenshot:

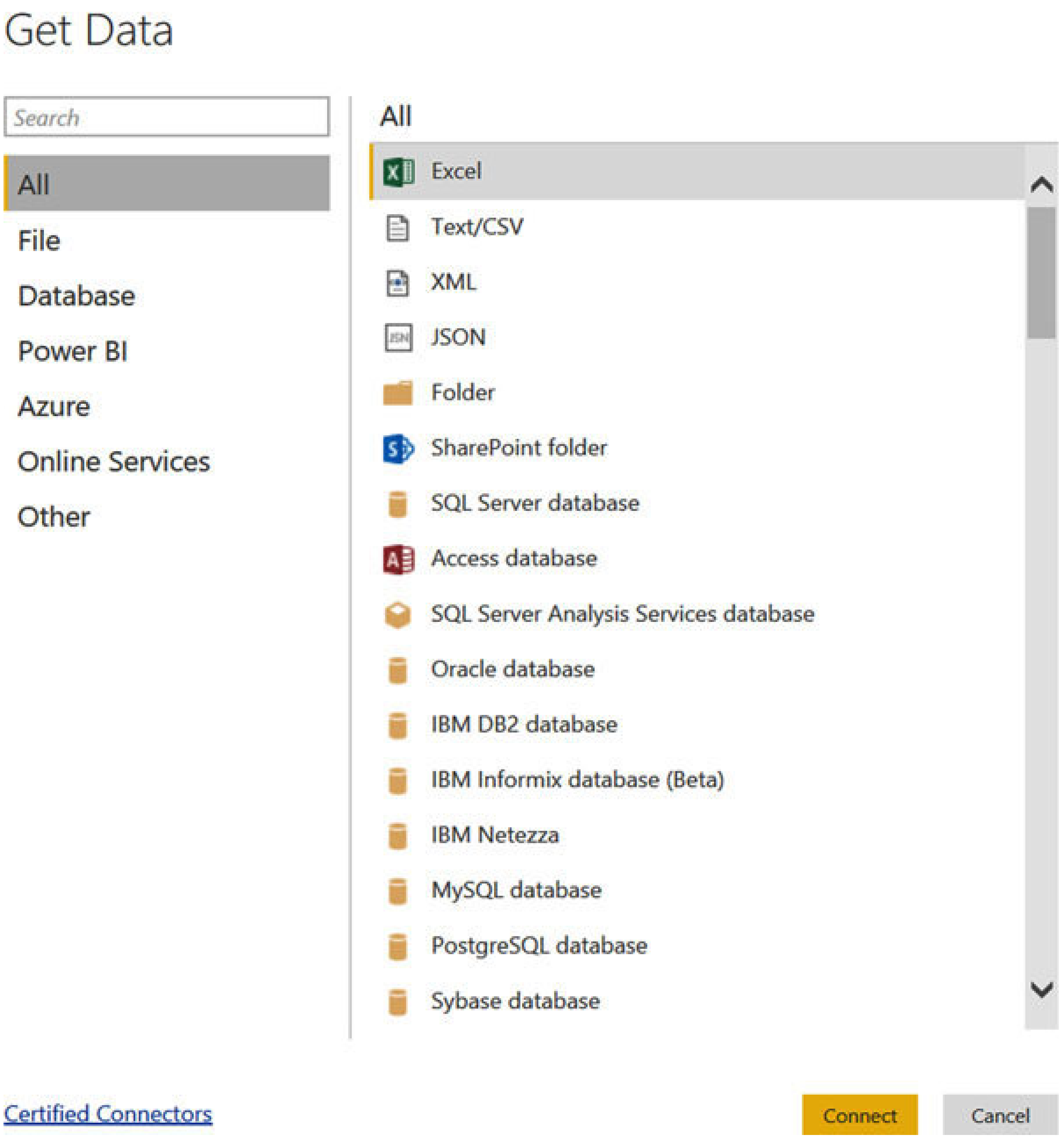
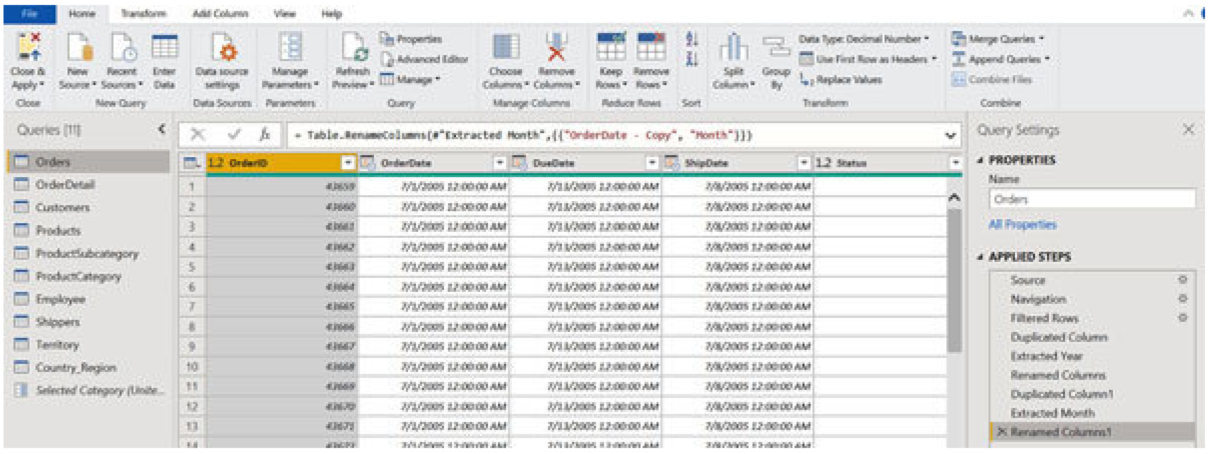


***Figure 1.6:*** *Get data dialogue*

The Power Query Editor is used to edit the loaded queries or tables. It allows the users to apply different transformations on the tables. The available data transformation options are consistent across all the data sources. When transformations are created, Power Query uses the M code language in the background.

Any operations performed on the query or field is stored in the applied steps.



Queries [11]

- Orders
- OrderDetail
- Customers
- Products
- ProductSubcategory
- ProductCategory
- Employee
- Shippers
- Territory
- Country_Region
- Selected Category (Unite...

```
= Table.RenameColumns(#"Extracted Month",{{"OrderDate - Copy", "Month"}})
```

| | 1.2 OrderID | OrderDate | DueDate | ShipDate | 1.2 Status |
|---|---|---|---|---|---|
| 1 | 43659 | 7/1/2005 12:00:00 AM | 7/13/2005 12:00:00 AM | 7/8/2005 12:00:00 AM | |
| 2 | 43660 | 7/1/2005 12:00:00 AM | 7/13/2005 12:00:00 AM | 7/8/2005 12:00:00 AM | |
| 3 | 43661 | 7/1/2005 12:00:00 AM | 7/13/2005 12:00:00 AM | 7/8/2005 12:00:00 AM | |
| 4 | 43662 | 7/1/2005 12:00:00 AM | 7/13/2005 12:00:00 AM | 7/8/2005 12:00:00 AM | |
| 5 | 43663 | 7/1/2005 12:00:00 AM | 7/13/2005 12:00:00 AM | 7/8/2005 12:00:00 AM | |
| 6 | 43664 | 7/1/2005 12:00:00 AM | 7/13/2005 12:00:00 AM | 7/8/2005 12:00:00 AM | |
| 7 | 43665 | 7/1/2005 12:00:00 AM | 7/13/2005 12:00:00 AM | 7/8/2005 12:00:00 AM | |
| 8 | 43666 | 7/1/2005 12:00:00 AM | 7/13/2005 12:00:00 AM | 7/8/2005 12:00:00 AM | |
| 9 | 43667 | 7/1/2005 12:00:00 AM | 7/13/2005 12:00:00 AM | 7/8/2005 12:00:00 AM | |
| 10 | 43668 | 7/1/2005 12:00:00 AM | 7/13/2005 12:00:00 AM | 7/8/2005 12:00:00 AM | |
| 11 | 43669 | 7/1/2005 12:00:00 AM | 7/13/2005 12:00:00 AM | 7/8/2005 12:00:00 AM | |
| 12 | 43670 | 7/1/2005 12:00:00 AM | 7/13/2005 12:00:00 AM | 7/8/2005 12:00:00 AM | |
| 13 | 43671 | 7/1/2005 12:00:00 AM | 7/13/2005 12:00:00 AM | 7/8/2005 12:00:00 AM | |

Query Settings

PROPERTIES

Name: Orders

All Properties

APPLIED STEPS

- Source
- Navigation
- Filtered Rows
- Duplicated Column
- Extracted Year
- Renamed Columns
- Duplicated Column1
- Extracted Month
- Renamed Columns1

***Figure 1.7:*** *Query Editor Interface*

**Power** Power Pivot is the calculation engine of Power BI. It is used to model the relationships between the tables and create calculations. Power pivot uses the **DAX Analysis** to build formulas and expressions.

**Power** Power View is the visualization technology that is used to create tables, graphs, and maps. It is embedded in the Power BI desktop and uses the drag-drop feature for faster creation of data visualization.

Following types of visualizations are available in Power BI:

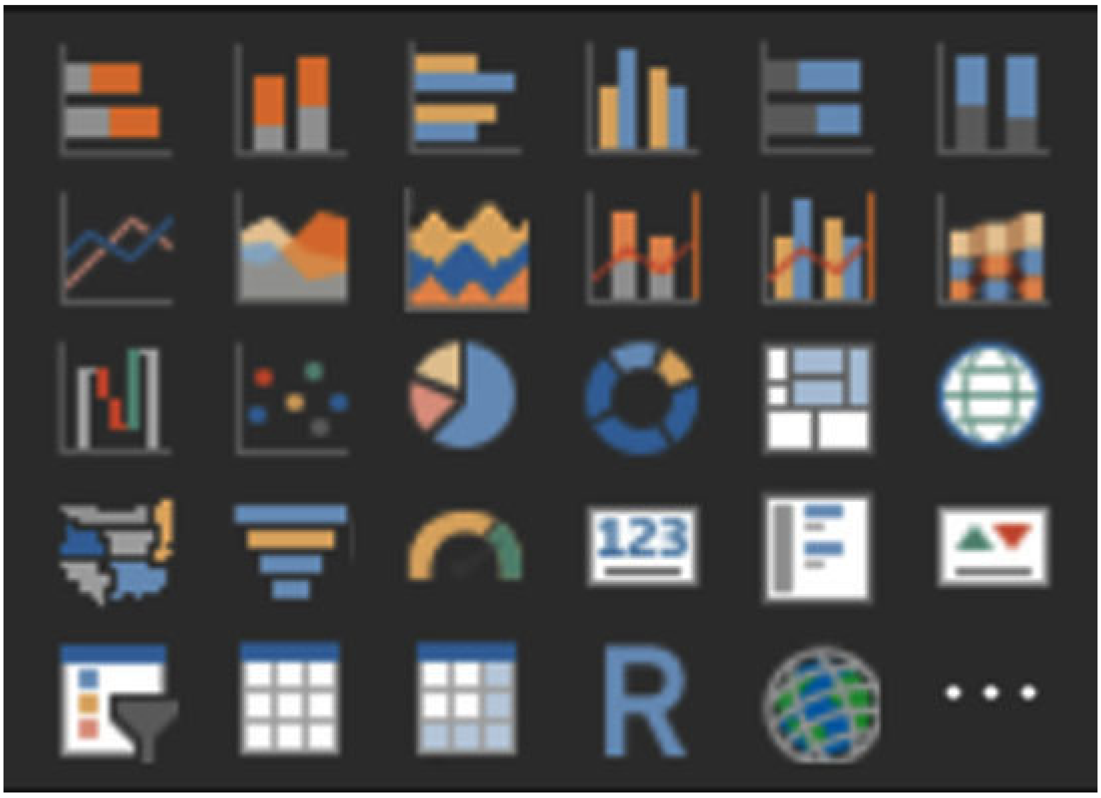


***Figure 1.8:*** *List of Visualizations*

**Power BI Service:** The Power BI Service is the cloud-based service of Power BI that allows publishing of reports, and datasets which are created using the Power BI desktop, to the cloud. The Power BI service will let you share visualizations with the other users and perform data refresh as well. Visualizations are published to the Service from the Power BI desktop.

**Power BI Report Server:** Power BI report server is similar to the Power BI Service, except that it is on-premises and not on a cloud service. It will interact with the on premise data, that is, within the firewall.

**Power BI mobile** Power BI offers a set of mobile apps for all kinds of mobile devices. Using these apps, users can interact with the visualizations and data located on the Power BI Service or report server.

## Power BI Environment

The Power BI desktop can load and transform any data and create visualizations. These visualizations can be published to the Power BI Service or Report Server. The users can access these visualizations from their desktop or mobile devices.

Power BI environment works as depicted in the following diagram. It connects to the various data sources, helps in creating visualizations, and these visualizations are deployed on the Power BI service or Report server for user consumption.

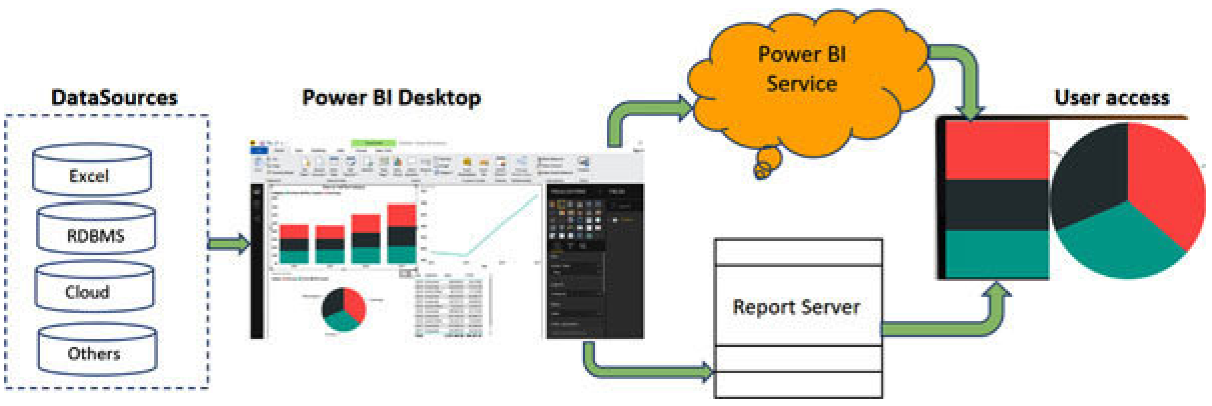


***Figure 1.9:*** *Power BI Architecture*

Power BI Service is used for the cloud-based data sources and Report Server is used for the on premise data.

## Different users of Power BI

Power BI can be used by different users in their day-to-day job. These users may have various responsibilities within the organization.

## *Power BI desktop developer*

As a Power BI desktop developer, we have strong skills in one or more databases. We love data and like to spend time in writing queries to understand the underlying data. We devote our time in understanding the relationships and cardinalities between the tables. We are proficient in using DAX (Data Analysis Expressions) to build formulas and expressions. The Power BI desktop developers are good at representing data in the form of charts, tables, and maps. By looking at a dataset, we can quickly figure out which chart will be best suited to represent this data.

**Responsibilities:**

We use the Power BI desktop free edition or Pro for authoring. We will be responsible for extracting the data from multiple data sources and create relationships between them. We will transform the data based on the organizational needs. We will create reports and publish them on the Power BI Service or report server. We will schedule data refreshes and will be responsible for the entire application. A Power BI desktop developer will maintain the application and if something goes wrong with the application, he/she will be the main contact for any fixes or answers.

## ***Power BI Analyst***

A Power BI Analyst has strong data skills and is someone who has been working in the organization for quite some time. With a good understanding of the organization's data, he works closely with the business users and executives. He is not concerned with tables, relationships, and joins, but is more interested in the output of the Power BI reports.

**Responsibilities:**

A Power BI Analyst will use the published reports on the Power BI Service or Report server. He is responsible to provide answers to the questions posed by his manager. He looks at the data for trends and outliers. He provides answers to questions such as – What is the trend of sales over the past five years; what is the difference in sales from the current year to the previous year; how will sales be impacted if more discounts are offered?

Before providing the answers to the higher management, a Power BI Analyst ensures the quality and accuracy of the data.

For faster analysis, sometimes he desires to develop his own Power BI reports instead of relying on the developers.

## *Power User*

A Power User has strong technical skills. In his previous job, he was responsible for running and managing the reports. In the past, he might have worked in an application similar to Power BI with an advanced knowledge of Microsoft Excel. He works with the business users but is interested in learning the new applications. He experiences that Power BI is intuitive, and he can use it as a self-service to get to quick data insights.

**Responsibilities:**

A Power User will need the Power BI pro license or work in Power service. He will use the existing data model created by the developers to create his own visualization. He also enhances the visualizations by adding or removing the filters. A Power User enhances the business team as he possesses both technical and the business knowledge.

## ***Executive User***

Executive Users are usually the head of a department or a business unit. They are not concerned about the underlying technology, but are more anxious to know about the overall health of their business unit. Analysts and Power users work in an executive user's team to provide him with answers, but he uses Power BI to know the truth himself.

**Responsibilities:**

An Executive user consumes dashboards published on the Power BI Service or Report server. Dashboards provide a quick glance of the business unit data and inform him on how the organization is doing. They look at the high-level data and are interested in seeing the charts with colors – red or green – red being bad and green being good. If something causes a concern, an executive user collaborates with his team.

## *Power BI licensing*

Power BI comes with different licensing options, which are as follows:

**Power BI Desktop free edition**

This edition is used by the users responsible for preparing/modeling the data and also creating the data visualizations. The amount of data that can be used with this version is 1 GB.

It provides access to the Power BI Service but does not allow the sharing of the dashboard with the other users.

**Power BI Pro**

The users require the Power BI pro licensing for collaboration, data modeling, content authoring, dashboard sharing, ad hoc analysis, and report publishing. It is named User licensing and allows the users to both create and consume content. Power BI Pro allows up to 10 GB of data.

There is a monthly subscription for $9.99 per user. A trial version for 60 days is available.

## Power BI Premium

This license option provides the user access to the content, i.e., pre-published dashboards and reports. It licenses the number of users having access to the content. These users can view the content, but for content creation, the Power BI pro licensing is required. The Power BI Premium allows the users to work with 50 GB of data. Larger implementations require such kind of licenses.

Power BI's competitive licensing and pricing structure encourages new and existing users to adopt the application.

## ***Power BI desktop installation***

The following system requirements are needed for the successful installation of the Power BI desktop:

**Operating** Windows 10, Windows 7, Windows 8, Windows 8.1, Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2

At least 1 GB

Internet Explorer version 10 or greater

Microsoft Power BI Desktop is available for 32-bit (x86) and 64-bit (x64) platforms. Check your operating system configuration before installing.

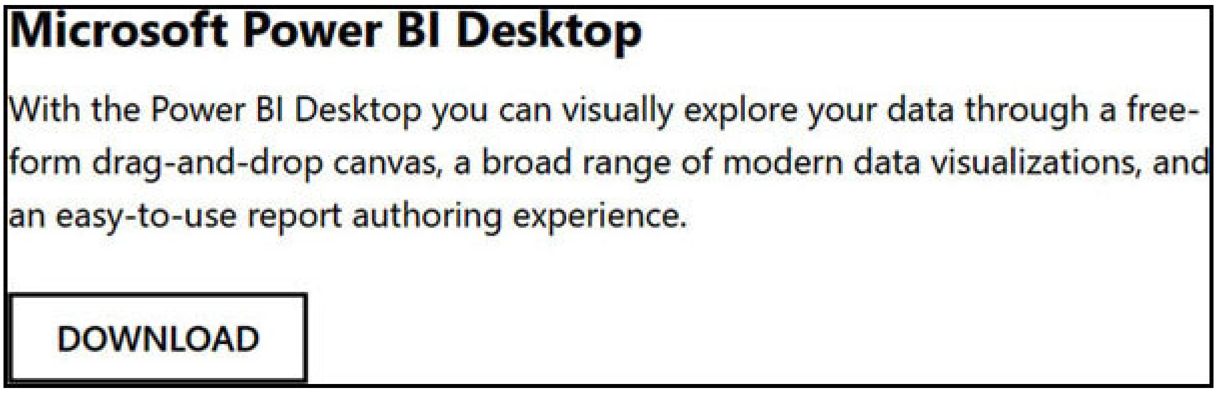
## *Installation*

Please complete the following installation steps to install the Power BI desktop:

To install the Power BI desktop, click on the following link:

https://powerbi.microsoft.com/en-us/downloads/

Several options will be displayed; select the one as shown below and hit

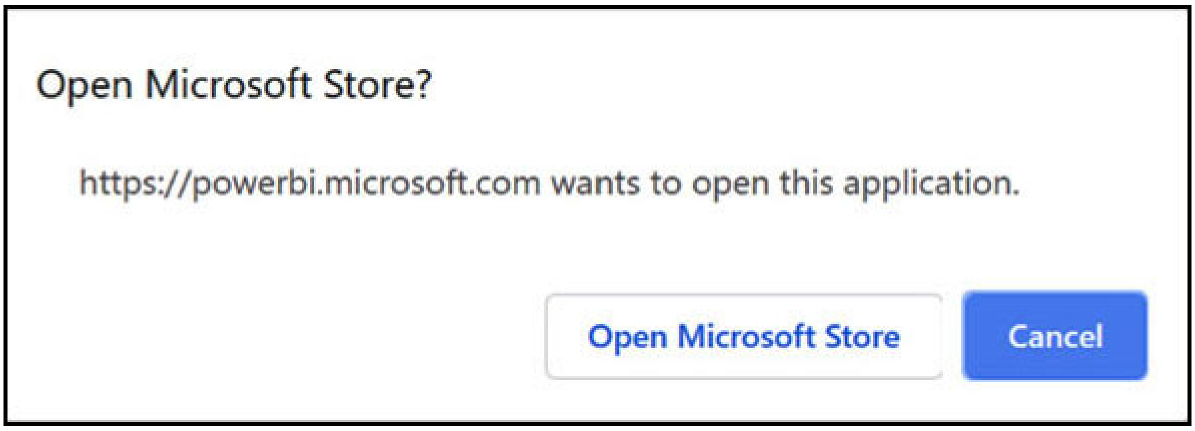


**Microsoft Power BI Desktop**

With the Power BI Desktop you can visually explore your data through a free-form drag-and-drop canvas, a broad range of modern data visualizations, and an easy-to-use report authoring experience.

***Figure 1.10:*** *Power BI desktop download*

In the subsequent screen, select **Open Microsoft**


Open Microsoft Store?

https://powerbi.microsoft.com wants to open this application.

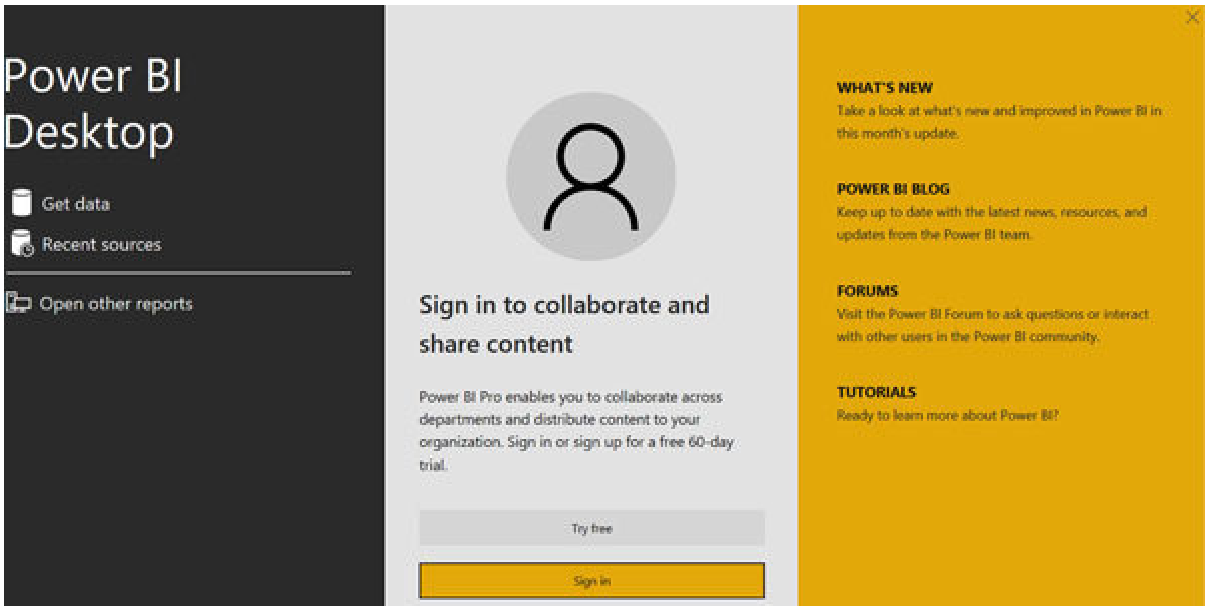
***Figure 1.11:*** *Microsoft Store*

On the next screen, click on Install. It will prompt a sign in. If you don't have an account with Microsoft, you can create one.

Once you are signed in, the Microsoft Power BI desktop will download and run on your machine. Once the download is completed, you will be asked if you want to launch the application. Click on

The Power BI welcome screen will be displayed. On this screen, you will see **What's new on Power** blogs, and tutorial.

If you want to try the Power BI Pro, you can sign in, or else just close this welcome screen by clicking on **X** at the right-hand corner.



Sign in to collaborate and share content

Power BI Pro enables you to collaborate across departments and distribute content to your organization. Sign in or sign up for a free 60-day trial.

WHAT'S NEW
Take a look at what's new and improved in Power BI in this month's update.

POWER BI BLOG
Keep up to date with the latest news, resources, and updates from the Power BI team.

FORUMS
Visit the Power BI Forum to ask questions or interact with other users in the Power BI community.

TUTORIALS
Ready to learn more about Power BI?

***Figure 1.12:*** *Power BI Desktop Welcome Screen*

## *Overview of Power BI desktop*

Power BI desktop provides a range of functionalities to connect to the data, edit query, create relationships, and build reports. The following figure explains the Power BI desktop interface:

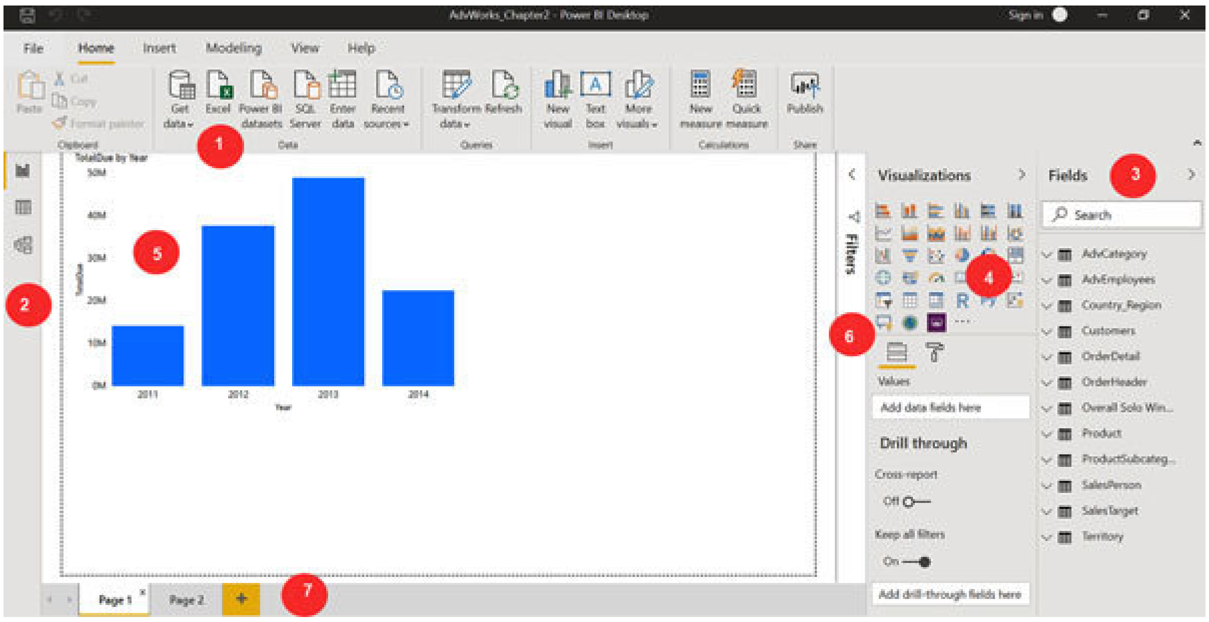


***Figure 1.13:*** *Power BI Desktop Interface*

Depicted in numbers, the different options available are as follows:

**The menu or** The ribbon on the Power BI desktop is contextual. We will see different options depending on which view is selected from the left pane– Report, Data, and Model. These views are explained in #2. Typically, menu has four tabs – **Modeling,** and

**Home** provides the data related options, such as connecting to the data, edit query, creating new Measure and column. It also gives the option to publish and share the visualizations.

The **Insert** option is contextual and shows up when Report View is selected. It provides options for laying out visualizations, such as showing gridlines, locking objects etc.

**Modeling** helps in creating a complex data model, New Measure, roles etc.

**Help** provides help related to Power BI, such as blogs and community.

**These are 3** The first one is the Report view and is selected by default when Power BI desktop is launched, second is the Data, and third is the Relationship view. Hover and click on each to see what do they display.

**Report view** displays the visualization created using the available charts and fields on the right.

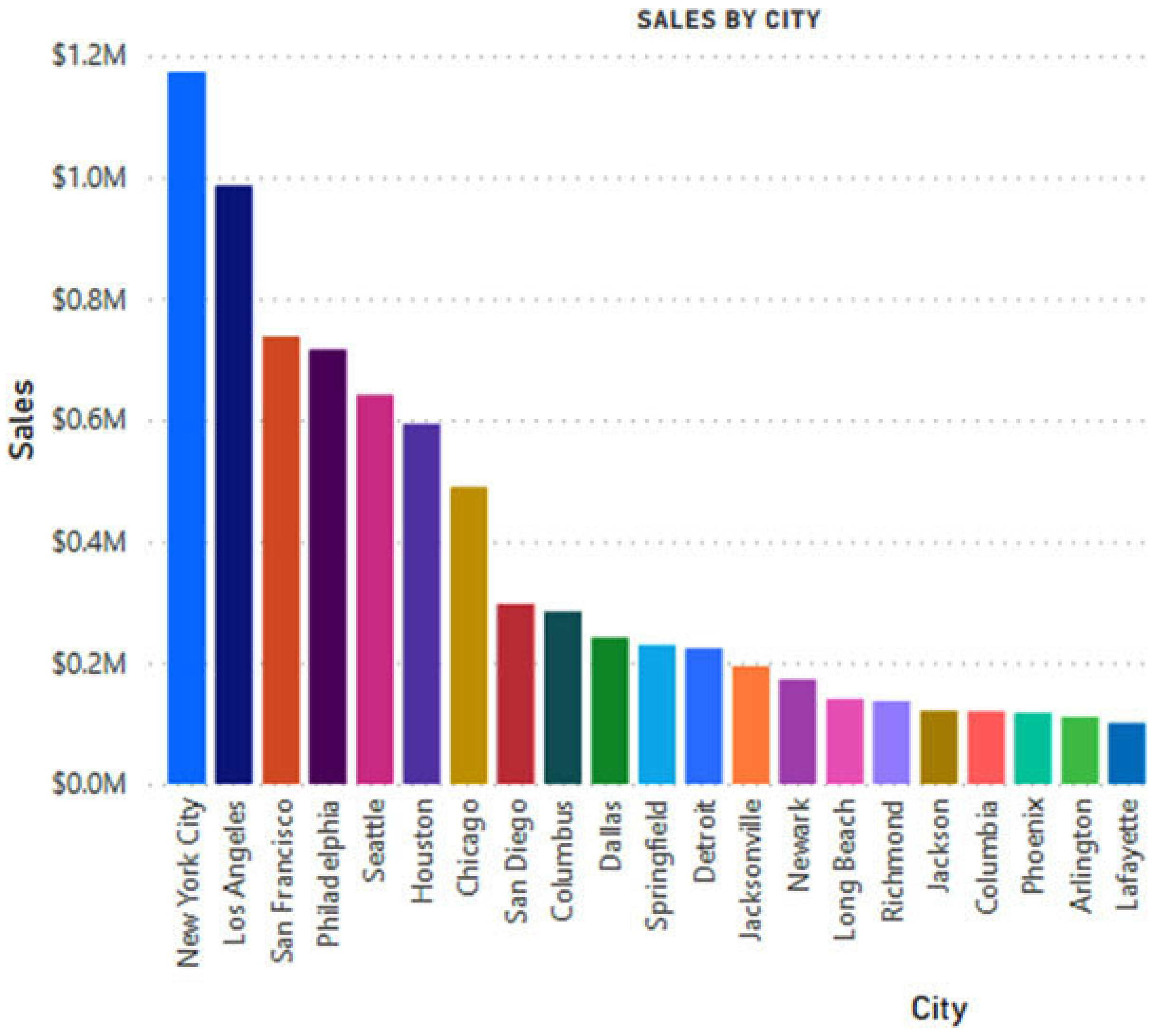
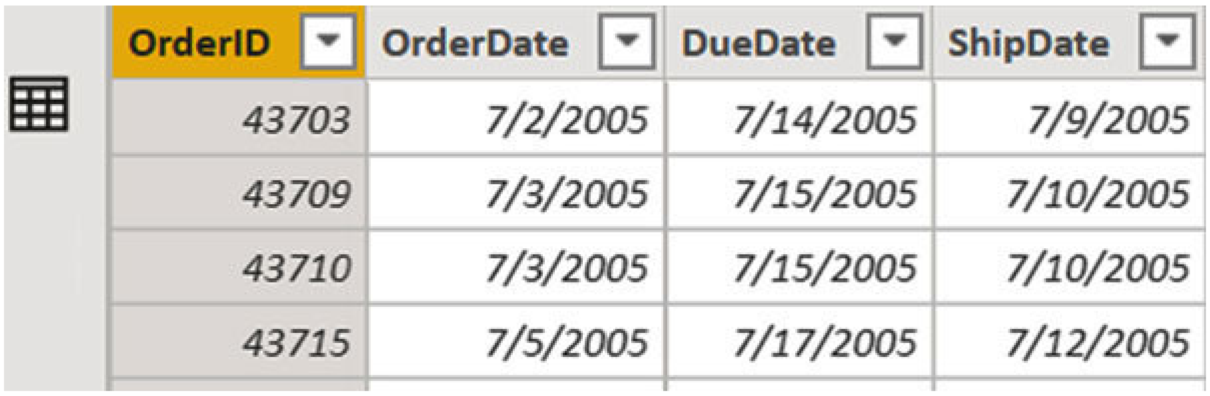


***Figure 1.14:*** *Visualization example*

**Data view** displays the data of the loaded queries/tables. To display the data, select the table from right, under the Fields.



| OrderID | OrderDate | DueDate | ShipDate |
|---|---|---|---|
| 43703 | 7/2/2005 | 7/14/2005 | 7/9/2005 |
| 43709 | 7/3/2005 | 7/15/2005 | 7/10/2005 |
| 43710 | 7/3/2005 | 7/15/2005 | 7/10/2005 |
| 43715 | 7/5/2005 | 7/17/2005 | 7/12/2005 |

*Figure 1.15: Data preview*

**Relationship view** displays the relationship between the queries.

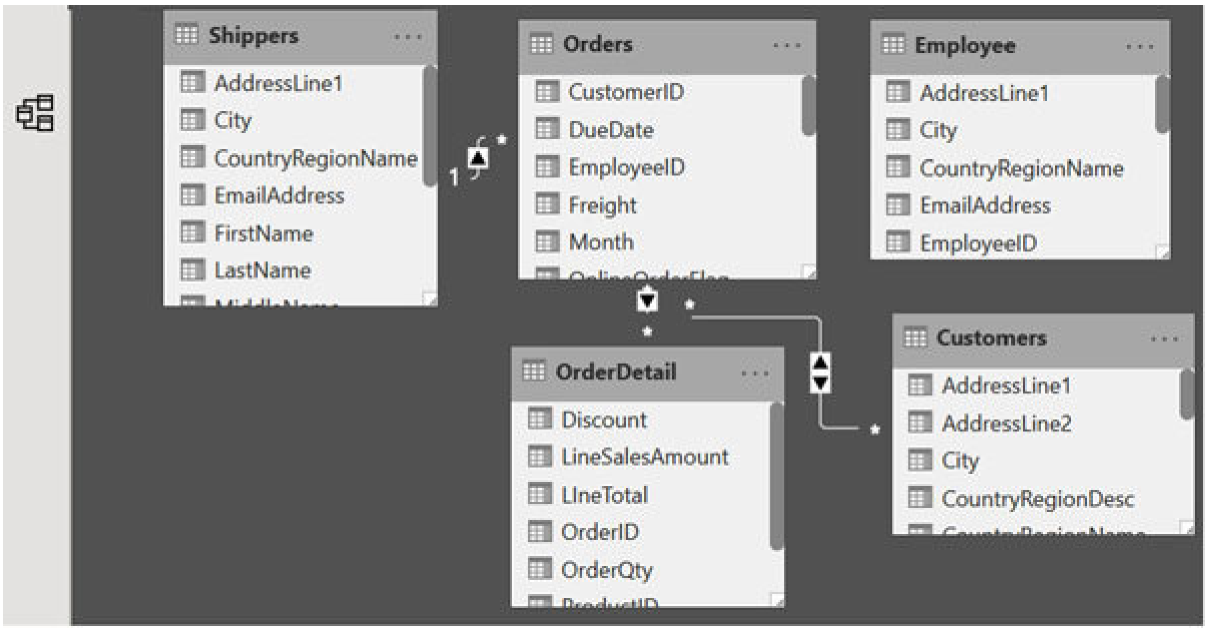


*Figure 1.16: Sample data model*

**Fields:** Display the queries or tables loaded in the application. It will display all the available tables and fields.

Power BI provides a wide range of charts and tables to display the data. These visualizations show up when the Report view (mentioned in 2) is selected from the left. To create a chart, click on a chart and it will show up on the canvas area. To display the data on the chart, select one or more Fields.

This is a canvas area, the charts, tables, and the other visualizations will be displayed in this section. One or more filters can also be applied to the visualizations. Power BI allows creation of multipage reports. Reports are made up of one or more visualizations. A new page is created from the Report view, by clicking on the + sign at the bottom of the screen.

We will use and learn more about these options in the subsequent chapters.

## *Initiate a Power BI implementation as a developer*

If you have been hired as a Power BI developer, these tasks will help in initiating and completing the project successfully. Every BI project is initiated because the business wants to gain insights into their data to take actionable decisions. As a developer, you should be able to do the following:

Understand the organizational structure and stakeholders. These are the key people with whom you will be working closely.

Confirm the Power BI infrastructure – type of licenses, use of Power Service or Report server, etc.

Participate in requirement meetings. Meet with the users to understand their requirements. Have an understanding on what they are looking for; what KPIs they are most interested in; what answers they are looking for; what are their pain points; do they use any existing reports, if yes then, what are they not getting from those reports that they want to get by using Power BI?

Since data is the key here, gain access and understand the data. Look at the type of available data sources – relational, excel, text, or any other. Consider the number of tables in the data set and the relationships between them. Make sure the tables contain the data that the users want to see in the reports.

Create mock-ups in Excel or some other tool to show the users, how visualizations will appear in Power BI and set their expectations.

If the users have not used Power BI before, then give them a quick tour.

Connect and extract the data into Power BI. Follow the best data modeling practices to create a robust and usable data model.

Follow the best visualization practices to create reports. During development, be in touch with the users for previewing your application.

Once the application is ready, publish it on Power BI Service or Report Server. Schedule data refreshes as needed.

A developer has the responsibility of the overall Power BI application. He should understand all the aspects of the implementation.

## *Analyze dataset used in this book*

This book uses the data which is a combination of relational database, excel, and text files. It uses the following tables:

| tables: |
|---|
| tables: tables: tables: tables: tables: tables: tables: tables: |
| tables: tables: tables: tables: tables: tables: |
| tables: tables: tables: tables: tables: tables: tables: |
| tables: tables: tables: tables: tables: tables: tables: tables: tables: tables: tables: |
| tables: tables: tables: tables: tables: |
| tables: tables: tables: tables: tables: tables: tables: tables: tables: |

***Table 1.7:*** *List of Tables used in the book*

We will be working with these tables throughout the book to create data model and visualizations. To explain additional concepts, we may be using some other tables and connections, the details of which you will see in the following chapters.

The following figure shows the structure of the **Orders** table:

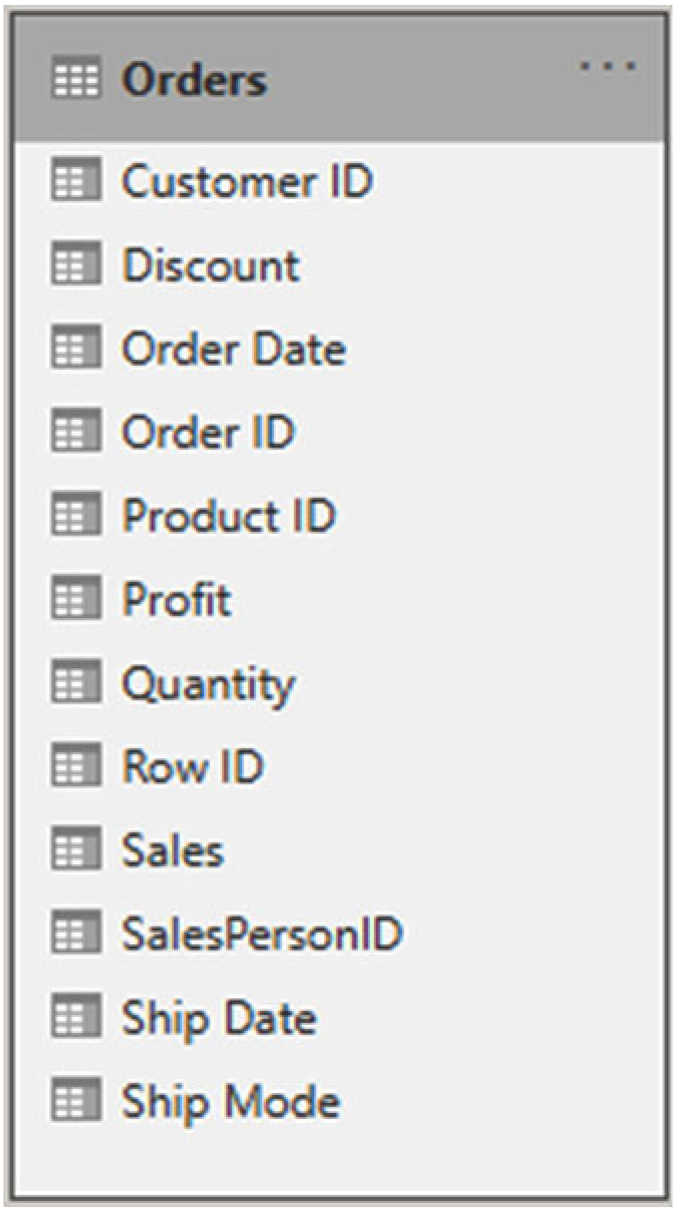


***Figure 1.17:*** *Order table*

Apart from the **Orders** table, there are other tables in our data model, which are shown in the following figure:

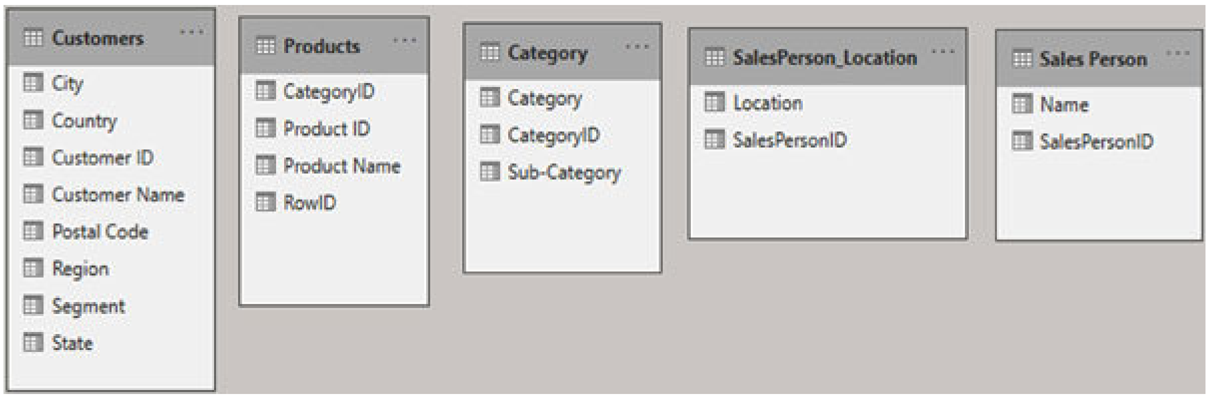


***Figure 1.18:*** *Other tables*

From the concepts, we learned so far in this chapter on Dimensions and Fact tables that we can identify the dimension and fact tables in the data source.

The dimension tables are as follows:

Customers

Products

Category

Sales Person Location

Sales Person

The fact table is as follows:

Orders

We will create a data model by creating a relationship between the fact and dimension tables.

**Dimension tables contain the descriptive or qualitative attribute of the data, such as customer information, etc. Fact tables contain measures or something we can aggregate, such as amount etc.**

Our objective through the course of this book will be to load these tables, create relationships between them, create transformations, and build a robust data model that can be used to create visualizations.

## Power BI development setup

To setup the development environment, download the accompanied ZIP code for this book and unzip the downloaded file in **C:\** of your laptop. Name the folder as the name of the ZIP file, that is, PowerBI. The unzipped folder should contain the following:

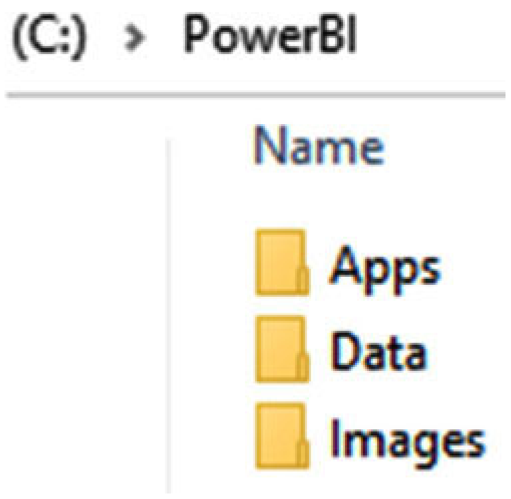


***Figure 1.19:*** *Development folder structure*

This folder will contain chapter-wise sample Power BI applications files.

The data files required by the application are stored in this folder. It includes the MS access database **PowerBI_Data.mdb** file. It also contains the excel and text files.

To make maximum use of this book, I would suggest readers to follow the exercises given in this book and create their own Power BI application files. Name your self-created files as Refer to the sample files in case of any issues.

**As a good practice, always analyze the data before you start working on it. Navigate to your data folder and check the tables and data contained in each table and file.**

## Conclusion

This chapter familiarized us with the basics of Business Intelligence. We gained knowledge on the key components of data warehouse and understood how the star schema and the snowflake schema use dimensions and measures. We are now equipped with the fundamentals of the Power BI application.

In this chapter we learned the following:

Key terminology of a Business Intelligence environment

The design of star schema and snowflake schema

Basics of the Dimension and Fact tables

What is Power BI and how it works

Overview of the underlying data sets used in this book

In the next chapter, we will load data from different sources, such as relational database, Excel, Text files, and many more.

We will also be creating relationships to create a data model.

## ***Questions***

What is Business Intelligence?

What functions are performed by an ETL application?

Why is star schema called a ‘Star'?

What information is stored in a fact table?

Why is star schema preferred for reporting and analysis?

What is the advantage of a dashboard?

Why is Power BI considered a Business Intelligence application?

What is a query tool in Power BI?

## *Answers*

Business Intelligence is a concept that deals with technology and infrastructure pertaining to the extraction, transformation, and presentation of data.

The ETL application performs Extract, Transform, and Load of data. It extracts the data from the various sources, mostly operational or transactional sources, transforms, and loads into the data warehouse tables.

In star schema, a fact table is surrounded by the dimension tables, giving it a star-like shape, hence the name.

Fact table stores measurable or metric fields.

Star schema is preferred for reporting and analysis because it reduces the number of tables and relationships between them, and therefore it is optimized for reporting.

The biggest advantage of a dashboard is that it gives quick snapshot of your organization's health by displaying all the Key Performance Indicators (KPIs).

Power BI is considered a BI application because it can connect and load diverse data. This data can be transformed according to the visualization's requirement. Power BI has a large pool for visualizations that help in better presentation of the data.

Power query is used for querying the data.

# CHAPTER 2

# Connect and Shape

## *Introduction*

As we understood in the previous chapter, the power of any Business Intelligence implementation lies in managing the data appropriately. Power BI can connect to disparate data sources, manage relationships between the different tables, and create a powerful data model.

In this chapter, we will load the data from the different data sources and transform them into the process of creating a data model. The loaded tables will be used in the subsequent chapters to create relationships, reports, and dashboards.

## Structure

In this chapter, we will discuss the following topics:

Data connections in Power BI

Overview of Query Editor

Loading data from the database files

Loading data from Excel and CSV files

Loading data from SQL Server database

Loading data from Web

Loading data from SharePoint

Loading data from Azure

Performing transformations

## Objectives

The objective of this chapter is to learn how the Power BI connects to the different data sources. We will gain knowledge on how to load the data from the different sources, such as database tables, XLS, and CSV files. We will also study how to create our own table in Power BI. As a bonus, we will understand how Power BI connects to the Web data, SharePoint Lists, and Azure.

## *Data connections in Power BI*

Power BI can connect and load the data from a wide range of data sources. The data modeling capabilities of Power BI are not restricted to the type of data source. We can perform any operation with any of the data source. While loading the tables, we can transform the data to match our requirements.

To check the available data sources, launch the Power BI desktop and select **Get Data** from the **Home** ribbon as shown in the following screenshot:

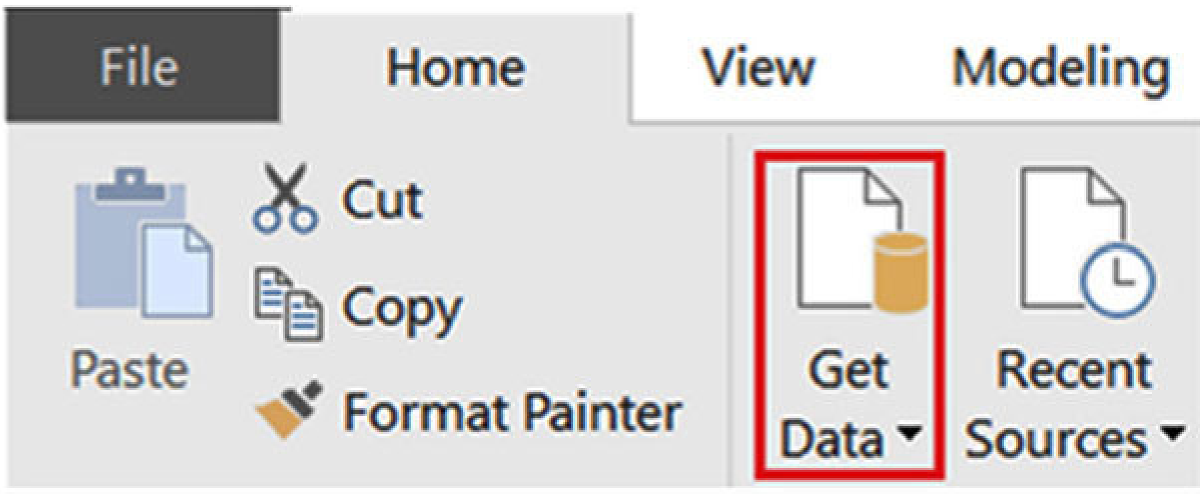


***Figure 2.1:*** *Using Get Data*

The **Get Data** option will display the list of commonly used data sources. Navigate all the way down in the list and click on it will display different data sources, as shown in the following figure:

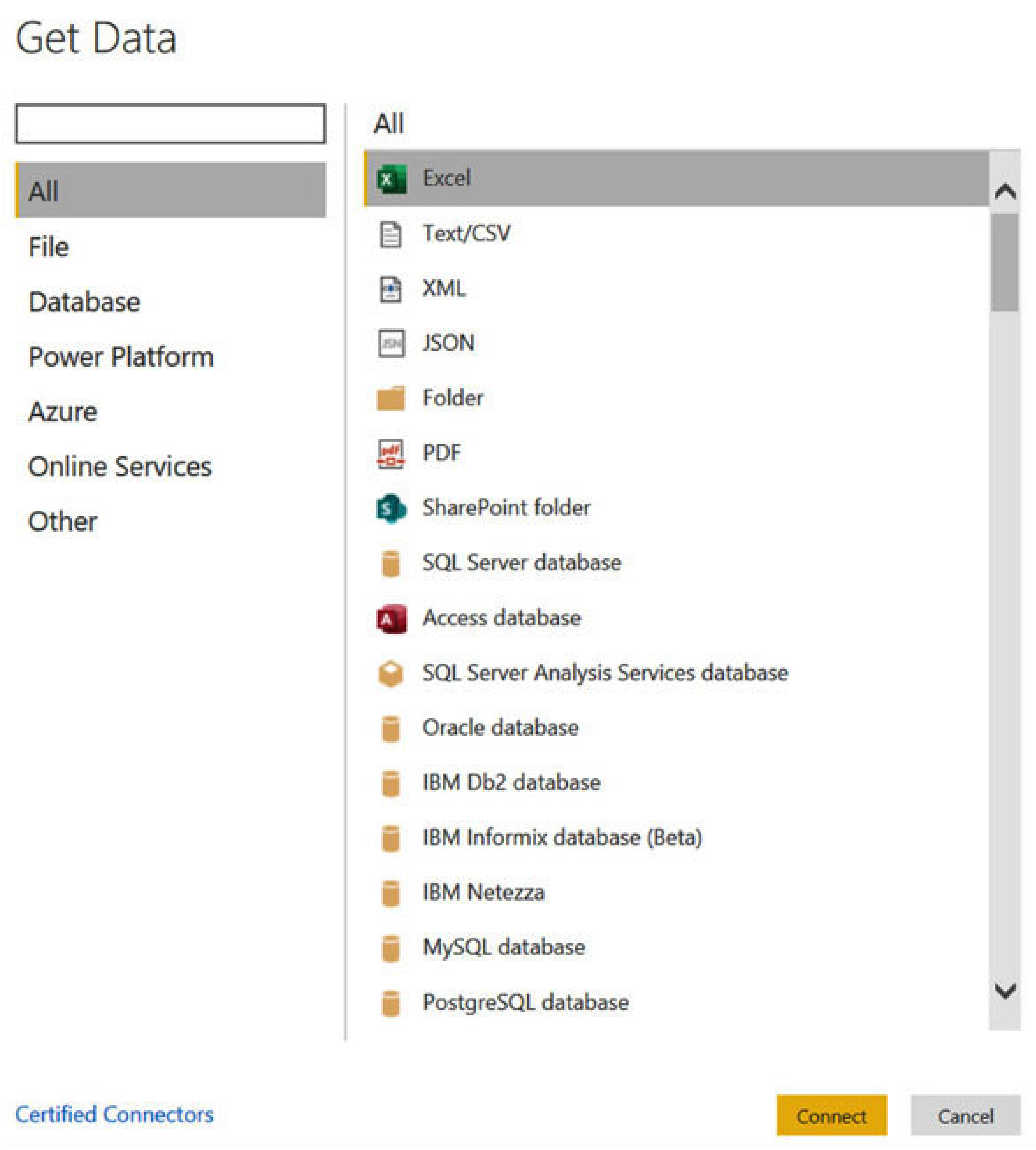


***Figure 2.2:*** *Available Data Sources*

Power BI desktop segregates the data sources (to which it can connect) into the following different categories:

File

Database

Power Platform

Azure

Online Services

Other

| Other Other |
|---|
| Other Other Other Other Other Other Other Other |

| Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other |
|---|
| Other Other Other Other Other Other Other Other Other Other Other Other |

Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other

Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other

Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other

| |
|---|
| Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other Other |

***Table 2.1:*** *List of all the data sources under each category*

Power BI has a large number of available data sources. The organizations can connect to a variety of data and create a data model.

**Power BI connects to the data using the native connectors. The developers can also create custom connectors. Some custom connectors are certified and distributed by Microsoft as certified connectors. The Power BI desktop displays the certified third-party connectors in the Get Data list of data sources (refer *figure***

## ***Connecting to data***

In the subsequent sections, we will see how to connect to the various forms of data in Power BI. We will use the same set of tables to show an example of how to connect to the database files, excel files, text files, etc.

## ***Connecting to database tables***

In this exercise, we will load the database tables that are available in the accompanied dataset We have used the MS-Access database as it comes with the MS-Office installation and is readily available on most laptops.

**As a good practice, always analyze the data before loading it. Navigate to your downloaded files' folder, that is, C:\PowerBi\Data and analyze the various tables in**

## *Loading the Orders table*

The **Orders** table is a database table and is present in the MS-Access database As the name suggests, the **Orders** table stores the details of specific order, such as, order id, order date, shipment details, sales, profit, quantity, and discount amounts.

**The steps to connect and load the Orders table are as follows:**

Launch the Power BI desktop. Cancel the welcome screen.

Save your application.

In the Power BI desktop, navigate to the menu **File** and select **Save** Navigate to **C:\PowerBI\Apps** and save your file in the **Apps** folder.

You can provide any name. In this book, the application is named as

If you are following the exercise, it is better to save the file with your initials, so that you don't accidentally overwrite the accompanied files.

From the **Home** tab, click on **Get Data** and at the bottom of the drop-down list, click on

On the **Get Data** dialog box, select **Database** as we are loading the database table. From the list of the data sources on the right, select the **Access** database as shown in the following screenshot:

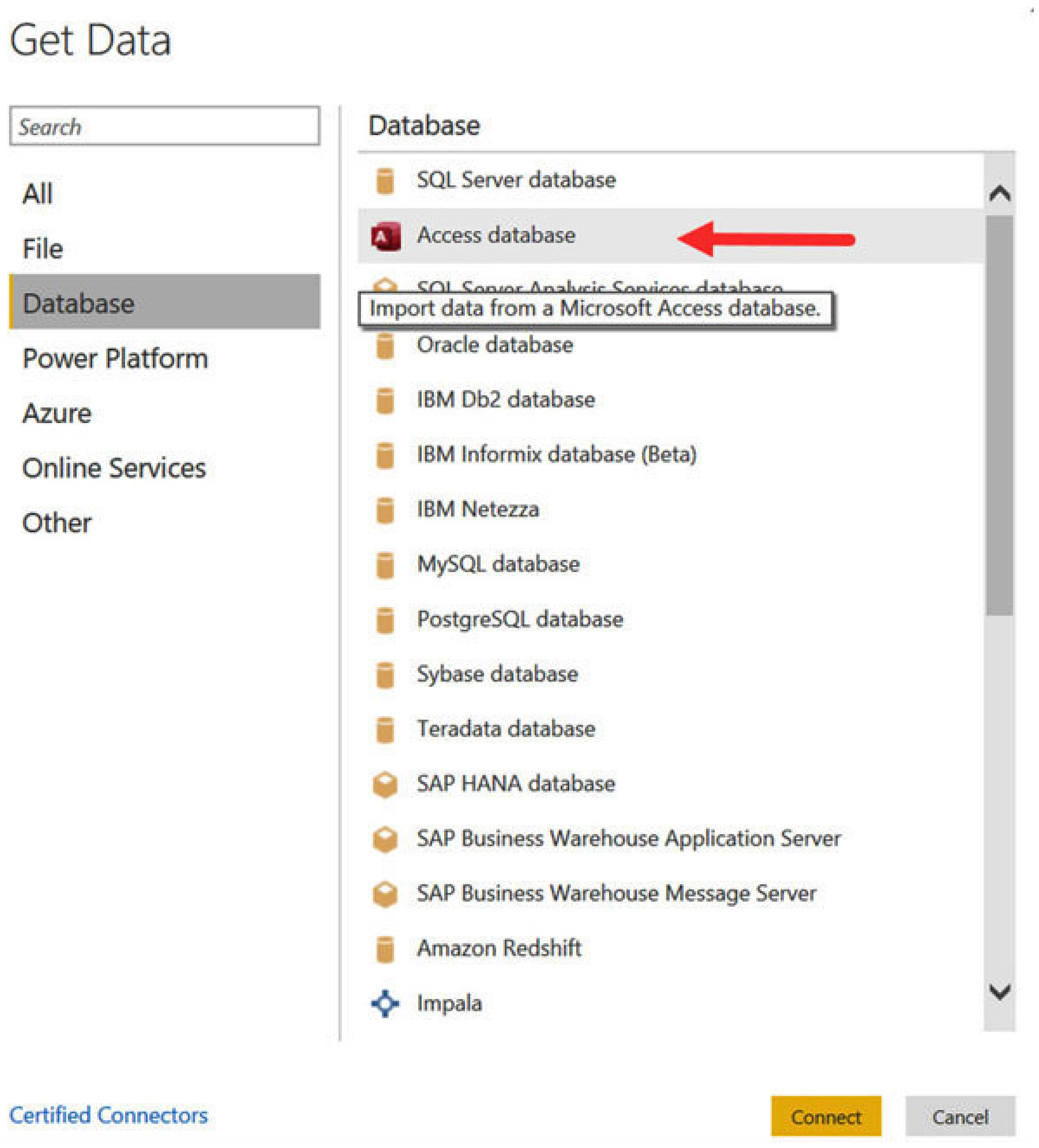
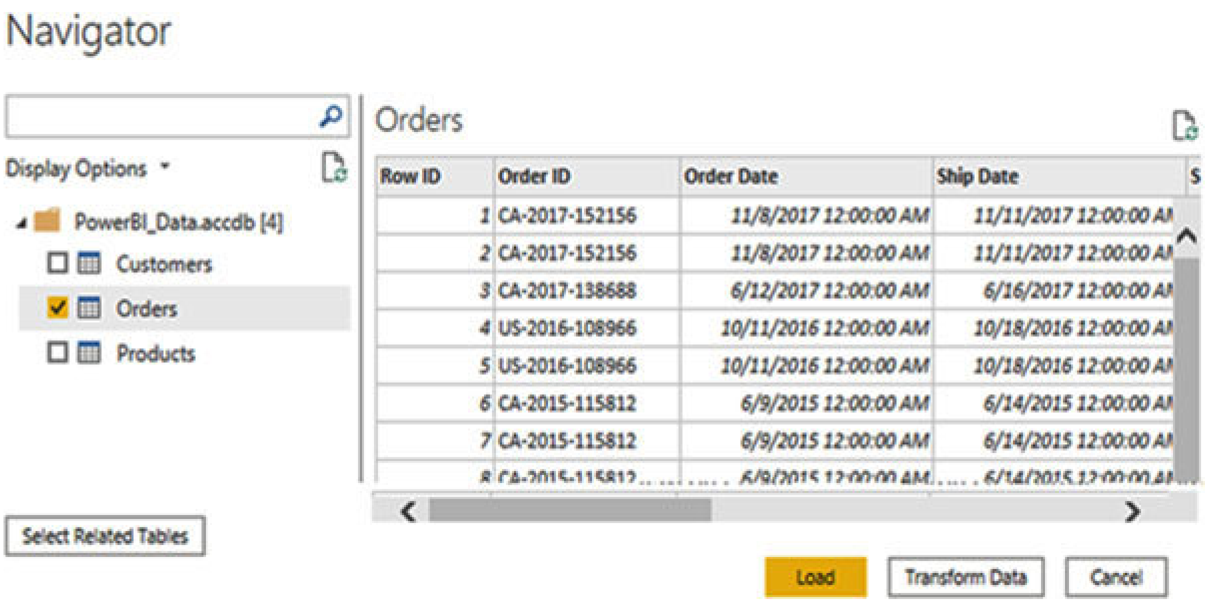


***Figure 2.3:*** *Selecting Access database*

Click on Connect at the bottom. Browse to the location of the downloaded that is,

Select the **mdb** file and open it.

The next dialog box, **Navigator** will list all the tables contained in the database. Select/check the **Orders** table from the left. Selecting the table will also display the limited data in the preview window as shown in the following screenshot:



| Row ID | Order ID | Order Date | Ship Date | S |
|---|---|---|---|---|
| 1 | CA-2017-152156 | 11/8/2017 12:00:00 AM | 11/11/2017 12:00:00 A | |
| 2 | CA-2017-152156 | 11/8/2017 12:00:00 AM | 11/11/2017 12:00:00 A | |
| 3 | CA-2017-138688 | 6/12/2017 12:00:00 AM | 6/16/2017 12:00:00 A | |
| 4 | US-2016-108966 | 10/11/2016 12:00:00 AM | 10/18/2016 12:00:00 A | |
| 5 | US-2016-108966 | 10/11/2016 12:00:00 AM | 10/18/2016 12:00:00 A | |
| 6 | CA-2015-115812 | 6/9/2015 12:00:00 AM | 6/14/2015 12:00:00 A | |
| 7 | CA-2015-115812 | 6/9/2015 12:00:00 AM | 6/14/2015 12:00:00 A | |
| 8 | CA-2015-115812 | 6/9/2015 12:00:00 AM | 6/14/2015 12:00:00 A | |

***Figure 2.4:*** *Navigator dialog box lists the tables*

At the bottom, as shown in the preceding figure, there are two options – **Load** and **Transform** The **Load** option will simply load the data. The **Transform Data** option will navigate to the Power Query Editor and let us perform the transformations before loading the data.

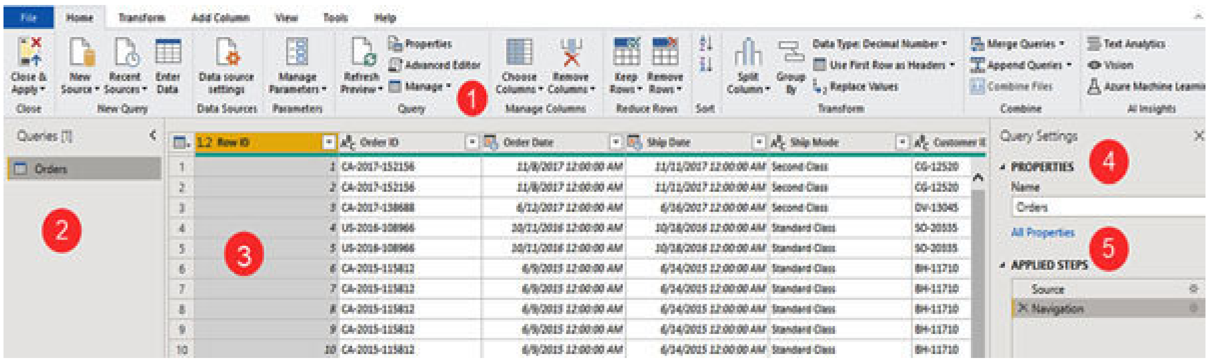
Choose the option **Transform** You will be navigated to the Power Query Editor.

There are different ways to connect to the data. In the preceding exercise, we used MS-Access and loaded only one table. We can also load multiple tables at one time.

## Understanding Power Query Editor

The query editor connects to one or many data sources and helps in transforming the data according to the organizational needs, and loads it in the Power BI desktop. It uses the mashup query language (M). When a data is imported in Power Query, it uses M in the background. It also provides the developers an editor window/interface to write powerful formulas and expressions.

The Query Editor can also be invoked from the Power BI desktop, **Home** tab | **Transform Data** option. In the following figure, we will explore the different options available in the Query Editor:



| | 1.2 Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID |
|---|---|---|---|---|---|---|
| 1 | 1 | CA-2017-152156 | 11/8/2017 12:00:00 AM | 11/11/2017 12:00:00 AM | Second Class | CG-12520 |
| 2 | 2 | CA-2017-152156 | 11/8/2017 12:00:00 AM | 11/11/2017 12:00:00 AM | Second Class | CG-12520 |
| 3 | 3 | CA-2017-138688 | 6/12/2017 12:00:00 AM | 6/16/2017 12:00:00 AM | Second Class | DV-13045 |
| 4 | 4 | US-2016-108966 | 10/11/2016 12:00:00 AM | 10/18/2016 12:00:00 AM | Standard Class | SO-20335 |
| 5 | 5 | US-2016-108966 | 10/11/2016 12:00:00 AM | 10/18/2016 12:00:00 AM | Standard Class | SO-20335 |
| 6 | 6 | CA-2015-115812 | 6/9/2015 12:00:00 AM | 6/14/2015 12:00:00 AM | Standard Class | BH-11710 |
| 7 | 7 | CA-2015-115812 | 6/9/2015 12:00:00 AM | 6/14/2015 12:00:00 AM | Standard Class | BH-11710 |
| 8 | 8 | CA-2015-115812 | 6/9/2015 12:00:00 AM | 6/14/2015 12:00:00 AM | Standard Class | BH-11710 |
| 9 | 9 | CA-2015-115812 | 6/9/2015 12:00:00 AM | 6/14/2015 12:00:00 AM | Standard Class | BH-11710 |
| 10 | 10 | CA-2015-115812 | 6/9/2015 12:00:00 AM | 6/14/2015 12:00:00 AM | Standard Class | BH-11710 |

***Figure 2.5:*** *Query Editor Interface*

In the preceding figure, depicted in numbers are the different options available, which are explained as follows:

**The menu or** The ribbon in the Query Editor contains four tabs – **Add** and

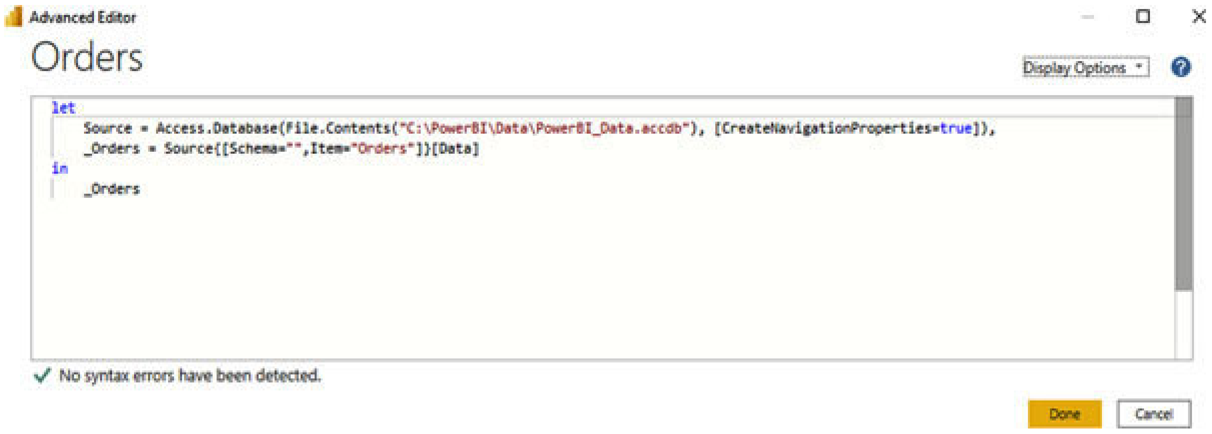
**Home** provides the common data related functionality like connecting to a new data source, enter data, splitting columns, or merging queries.

**Transform** provides the options for the data transformation tasks such as group by, transpose, changing data types, splitting columns, and so on.

**Add Column** contains column related tasks such as adding new custom or conditional column.

**View** displays the advanced editor. The advanced editor generates the code for each step taken within the Query Editor. It also provides with an option to write the custom code to perform transformations. We can see the code written for the operations performed so far.

Go to **View** and select **Advance Editor** as shown in the following figure



```
let
    Source = Access.Database(File.Contents("C:\PowerBI\Data\PowerBI_Data.accdb"), [CreateNavigationProperties=true]),
    _Orders = Source{[Schema="",Item="Orders"]}[Data]
in
    _Orders
```

✓ No syntax errors have been detected.

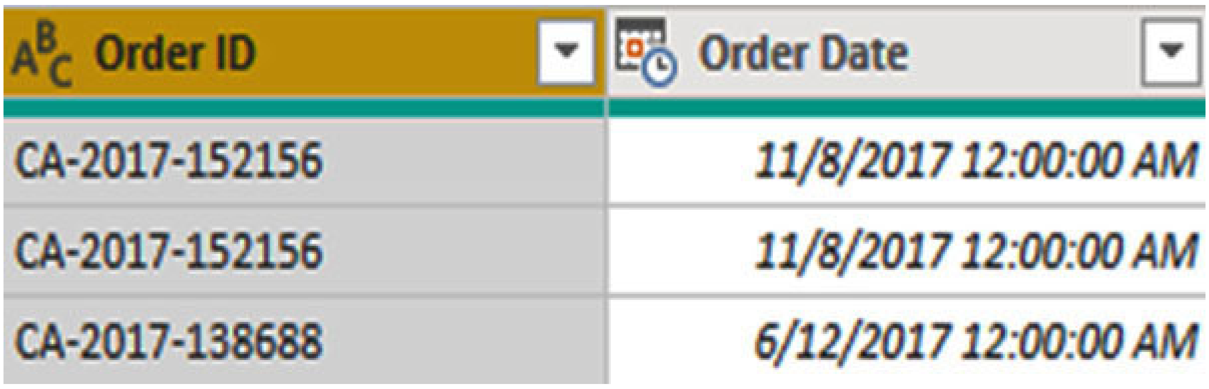
***Figure 2.6:*** *Advanced Editor*

Since no transformation steps are performed, the code only displays the source information of the database.

This lists the queries available for transformations. In our case, it is Orders as this is the only query we have connected to so far.

This is the preview of the data from the selected queries. This data is available for shaping and cleansing. When Query Editor connects to the data, it automatically assigns the data types to each of the columns based on the source data. The datatype can be seen in the column header itself.

For example, take a look at the following figure:



| Order ID | Order Date |
|---|---|
| CA-2017-152156 | 11/8/2017 12:00:00 AM |
| CA-2017-152156 | 11/8/2017 12:00:00 AM |
| CA-2017-138688 | 6/12/2017 12:00:00 AM |

***Figure 2.7:*** *Data Preview*

In the preceding figure, the column header of **Order ID** is prefixed with characters, such as which shows that the datatype of the column is Text. Similarly, the **Order Date** column header shows a small calendar icon to display that it has a datatype of

Datatypes can also be viewed and changed from the ribbon option **Home/Data** Click on the **Order ID** column and navigate to the **Home/Data Type** to view the associated data type as shown in the following screenshot:

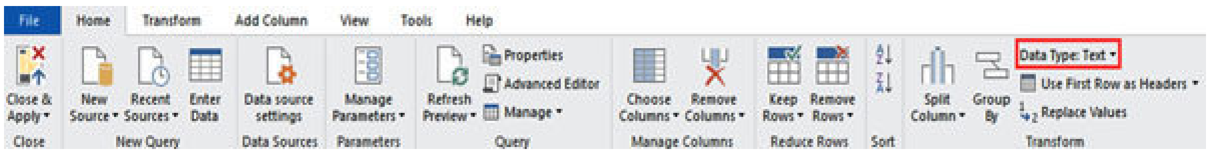


***Figure 2.8:*** *Data Type of fields*

This displays the **Query Settings** and shows the properties of the query.

**APPLIED STEPS** is also part of query setting. Any transformation step applied to the query is displayed here. We can cancel any of the applied step to undo our transformations.

**Transformations will involve changing the data type and summarization of the columns so that data is displayed in the correct format.**

Applying the transformations on the

Continuing with our orders query, complete the following steps:

Change the data types. It will be beneficial to change the data types of the various columns according to our reporting needs.

We can remove the time from the **Date** columns, as we are not interested in the time part of the dates.

Select **Order Date** and *Shift* + click on **Ship** Navigate to the **Home** ribbon, and from the **Data Type** drop down list, change it to

Take a look at the following figure on how to change the datatype to date:

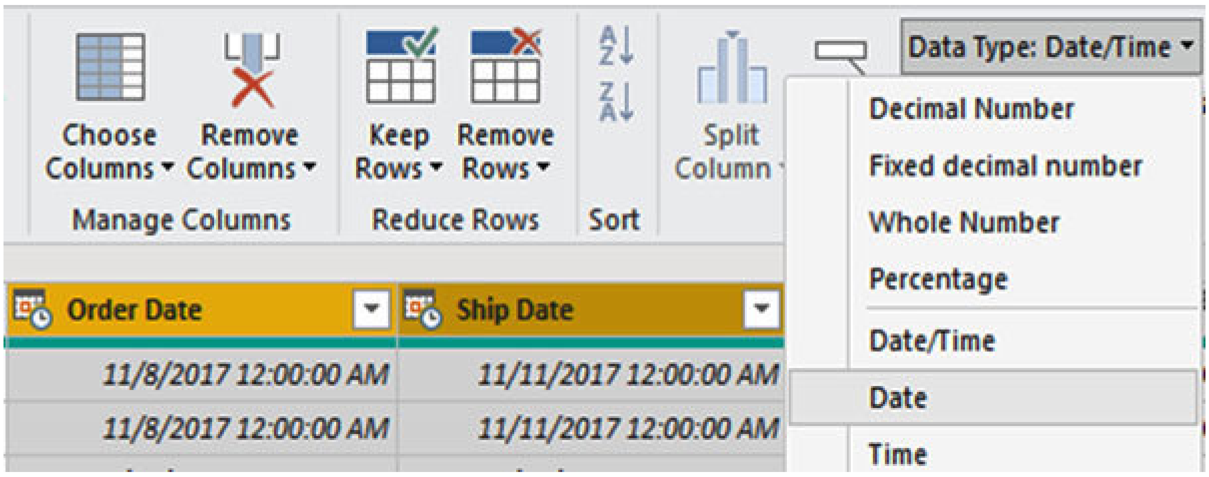
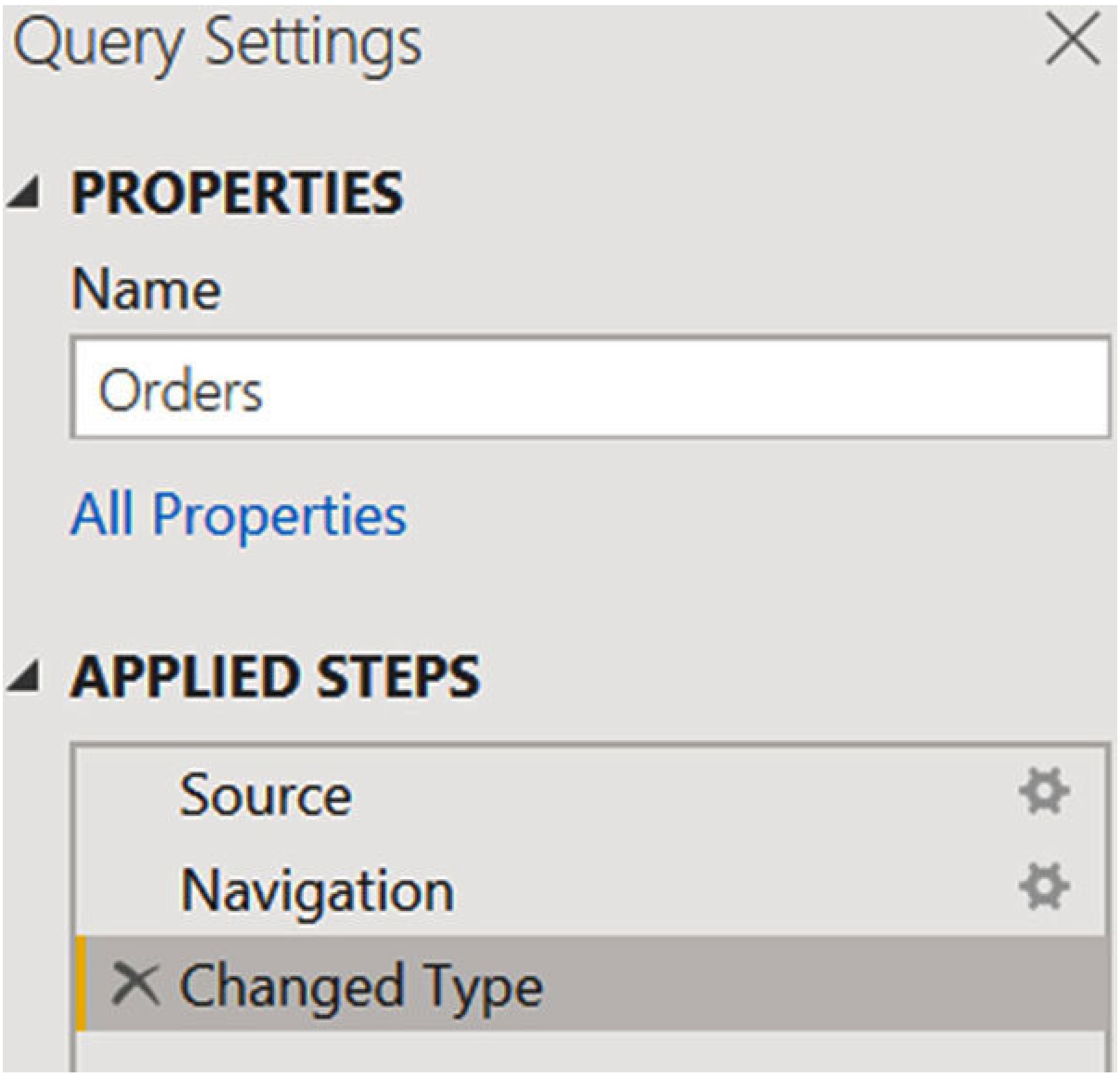


***Figure 2.9:*** *Changing data type to date*

Also notice **APPLIED STEPS** on the right under **Query** it shows **Changed Type** referring to the data type changed for the columns as shown in *figure* To revert the changes, just close the specific **APPLIED STEPS** and it will undo the changes.


Query Settings

PROPERTIES

Name

Orders

All Properties

APPLIED STEPS

Source

Navigation

Changed Type

***Figure 2.10:*** *Applied Steps*

The datatype of a column can also be changed by right-clicking on the column and selecting **Change** Here, many more options are available, which can be used as and when needed. Take a look at the following figure to note the options:

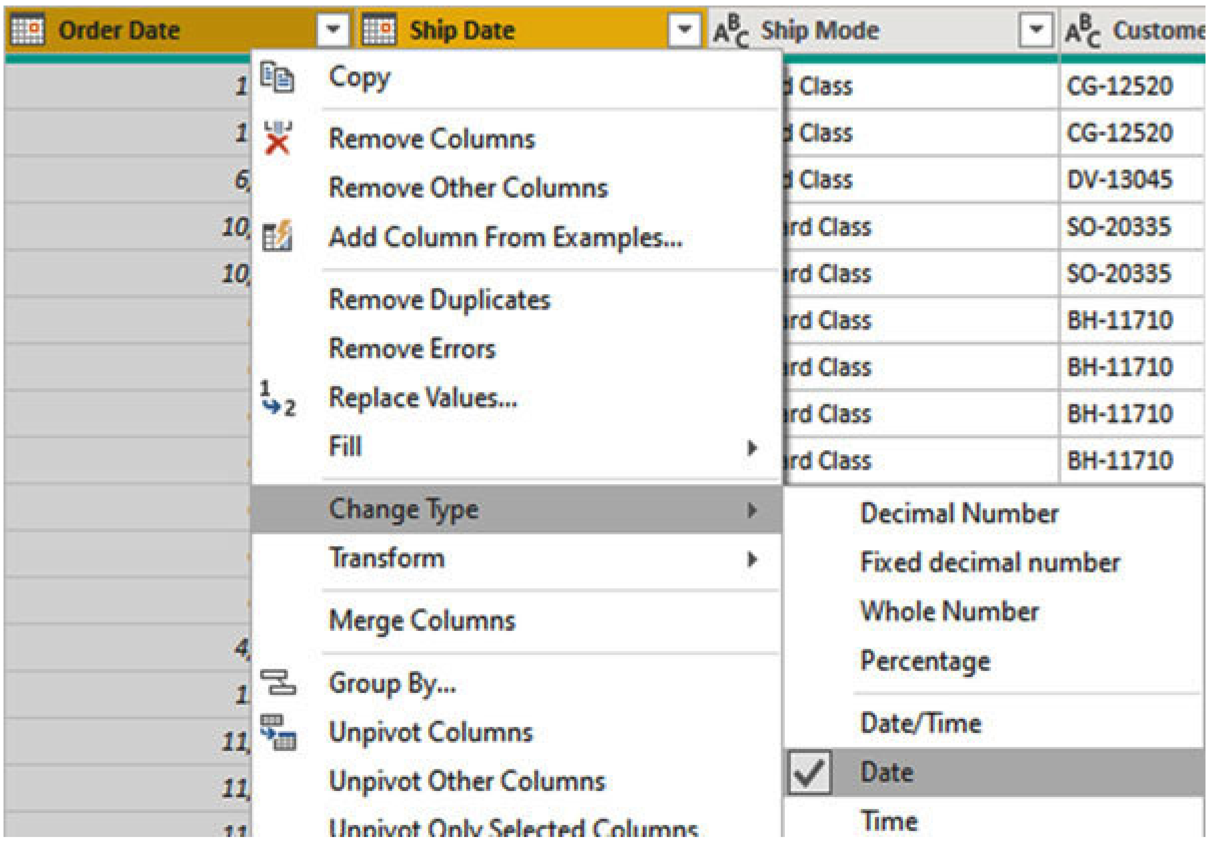


*Figure 2.11:* *Alternative method of changing Data Type*

After making changes in the **Query** navigate to the **File** menu and select **Close &** This will apply the query changes and navigate to the Power BI desktop.

Save your application.

Query editor has a lot of options to transform the queries. In the section 'More of Query Editor', we will learn about it in more detail.

## *Verifying the loaded query*

In the Power BI desktop, in the data view, we can preview the data and see all the loaded fields on the right, as shown in the following screenshot:

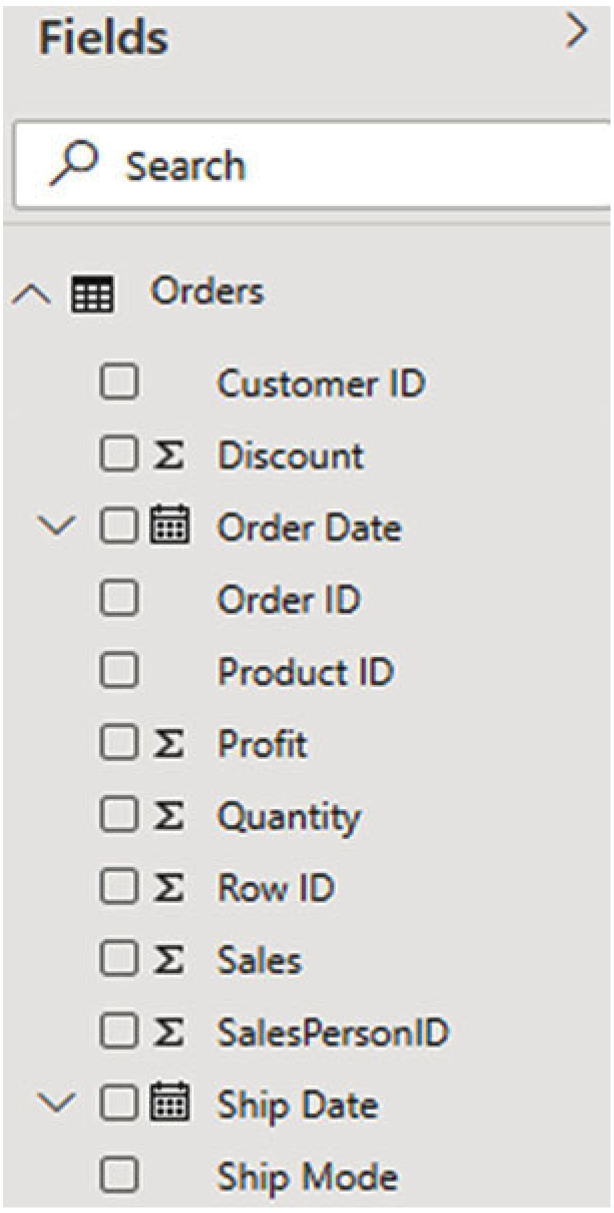


***Figure 2.12:*** *Loaded fields*

After reviewing the fields, we can decide which fields are needed for the visualizations and if they are in the right format.

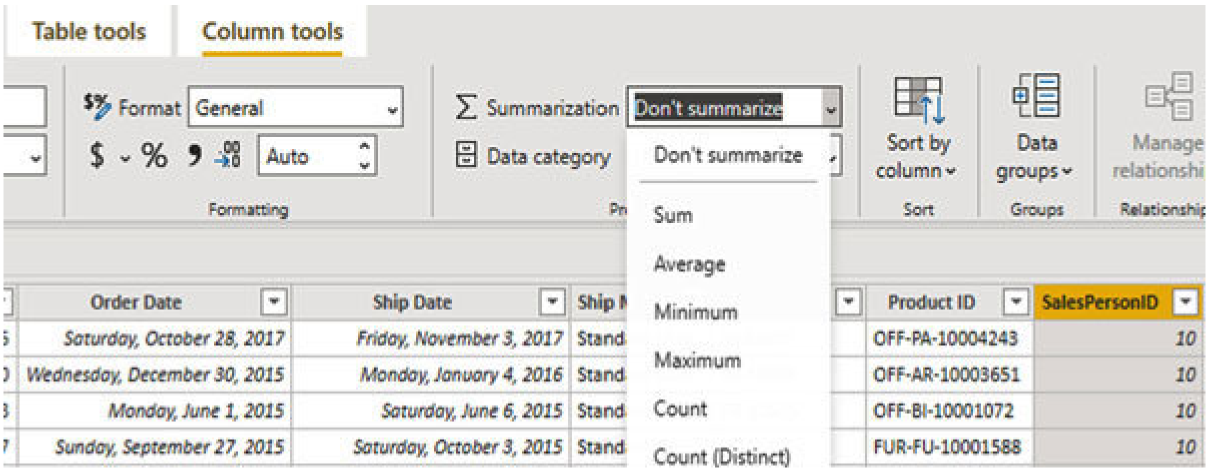
**Changing the default summarization:**

In the preceding field list of **Orders** table, notice the columns with the summarization symbol (∑). Power BI has detected these columns as measures, as they have the numeric data type. Power BI will perform the summarization or aggregation on them.

While loading the tables, it is always a good practice to review all the columns and change their summarization property as required. Not all the columns need to be summarized, some can be keys and will be used only for joins. Some of the columns are also numeric fields but are used as identifiers or flags. We should change the summarization property of any such columns in a table.

After a quick review, we will notice that Power BI has identified **SalesPersonID** as a measure, but this column is a key field and will be used to join with the **SalesPerson** table (this table will be loaded in the subsequent sections). We will change the summarization of this column.

To change the summarization property, select **SalesPersonID** from the fields (or data preview) navigate to **Column tools** ribbon and change the default **Summarization** to **Don't Summarize** as shown in the following screenshot.



Format: General

Summarization: Don't summarize

| Order Date | Ship Date | Ship M | Product ID | SalesPersonID |
|---|---|---|---|---|
| Saturday, October 28, 2017 | Friday, November 3, 2017 | Stand | OFF-PA-10004243 | 10 |
| Wednesday, December 30, 2015 | Monday, January 4, 2016 | Stand | OFF-AR-10003651 | 10 |
| Monday, June 1, 2015 | Saturday, June 6, 2015 | Stand | OFF-BI-10001072 | 10 |
| Sunday, September 27, 2015 | Saturday, October 3, 2015 | Stand | FUR-FU-10001588 | 10 |

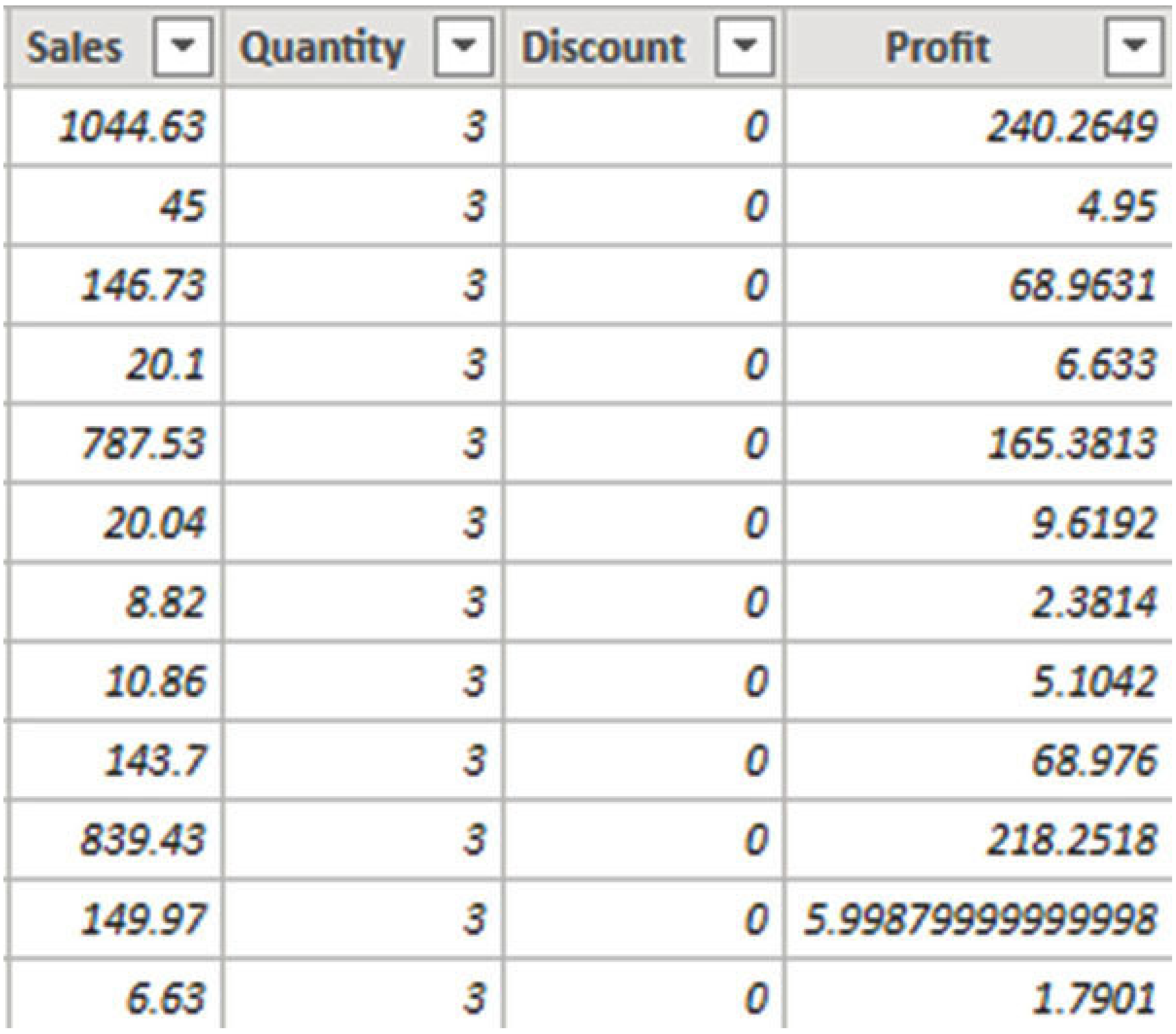
***Figure 2.13:*** *Changing the default field summarization*

Summarization helps in aggregating the results, but if the column is not required for aggregation, its summarization property should be changed.

**The summarization of a column can also be changed to Sum, Average, or any other type.**

**Changing the display format of the numbers:**

While the data view is still selected, look at the data preview. Observe the columns and See if the values in these columns display appropriate results. For example, Profit is displayed in multiple decimal places as shown in the following screenshot:



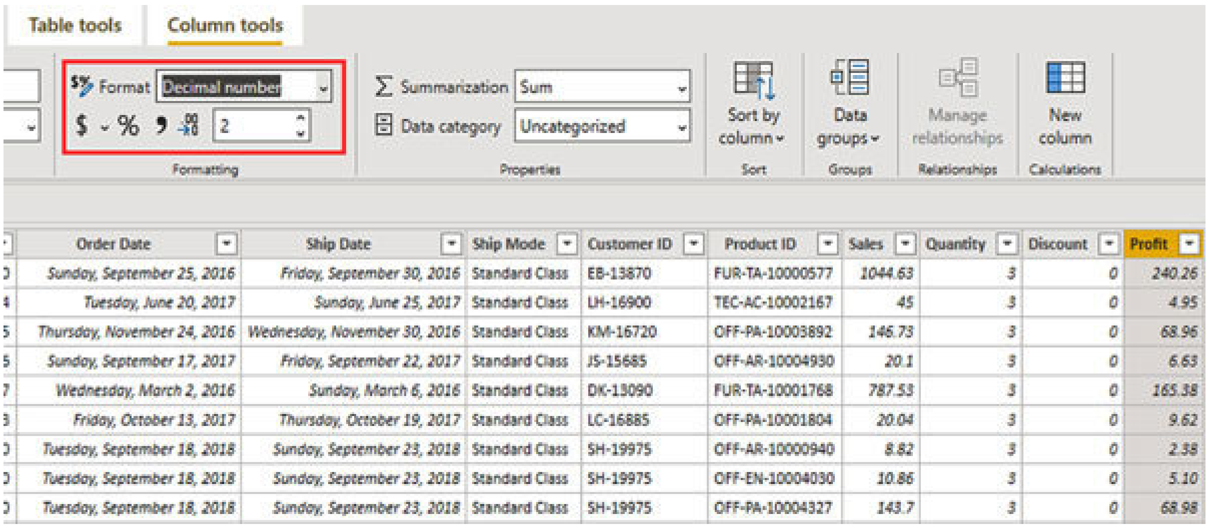
| Sales | Quantity | Discount | Profit |
|---|---|---|---|
| 1044.63 | 3 | 0 | 240.2649 |
| 45 | 3 | 0 | 4.95 |
| 146.73 | 3 | 0 | 68.9631 |
| 20.1 | 3 | 0 | 6.633 |
| 787.53 | 3 | 0 | 165.3813 |
| 20.04 | 3 | 0 | 9.6192 |
| 8.82 | 3 | 0 | 2.3814 |
| 10.86 | 3 | 0 | 5.1042 |
| 143.7 | 3 | 0 | 68.976 |
| 839.43 | 3 | 0 | 218.2518 |
| 149.97 | 3 | 0 | 5.99879999999998 |
| 6.63 | 3 | 0 | 1.7901 |

***Figure 2.14:*** *Columns with 3 places of decimals*

We can change this display format to 2 places of decimals.

The option to change the decimal places is available from the **Column tools** ribbon. Navigate to the **Format** option and change the values.

To change the number of decimals, select **Profit** from the data preview or fields and from the **Column tools** ribbon under change the value of the decimal number to 2 places as shown in the following screenshot:



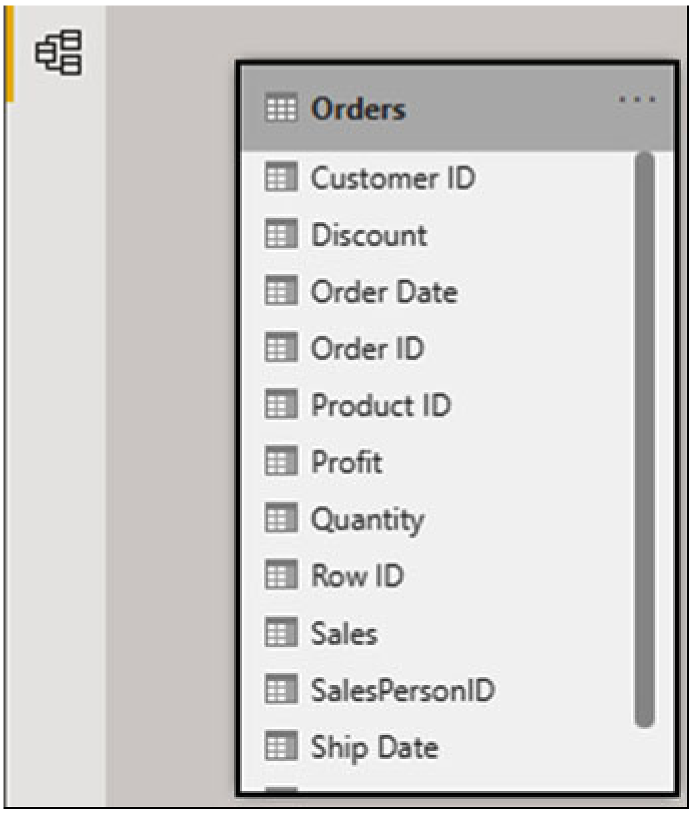
| Order Date | Ship Date | Ship Mode | Customer ID | Product ID | Sales | Quantity | Discount | Profit |
|---|---|---|---|---|---|---|---|---|
| Sunday, September 25, 2016 | Friday, September 30, 2016 | Standard Class | EB-13870 | FUR-TA-10000577 | 1044.63 | 3 | 0 | 240.26 |
| Tuesday, June 20, 2017 | Sunday, June 25, 2017 | Standard Class | LH-16900 | TEC-AC-10002167 | 45 | 3 | 0 | 4.95 |
| Thursday, November 24, 2016 | Wednesday, November 30, 2016 | Standard Class | KM-16720 | OFF-PA-10003892 | 146.73 | 3 | 0 | 68.96 |
| Sunday, September 17, 2017 | Friday, September 22, 2017 | Standard Class | JS-15685 | OFF-AR-10004930 | 20.1 | 3 | 0 | 6.63 |
| Wednesday, March 2, 2016 | Sunday, March 6, 2016 | Standard Class | DK-13090 | FUR-TA-10001768 | 787.53 | 3 | 0 | 165.38 |
| Friday, October 13, 2017 | Thursday, October 19, 2017 | Standard Class | LC-16885 | OFF-PA-10001804 | 20.04 | 3 | 0 | 9.62 |
| Tuesday, September 18, 2018 | Sunday, September 23, 2018 | Standard Class | SH-19975 | OFF-AR-10000940 | 8.82 | 3 | 0 | 2.38 |
| Tuesday, September 18, 2018 | Sunday, September 23, 2018 | Standard Class | SH-19975 | OFF-EN-10004030 | 10.86 | 3 | 0 | 5.10 |
| Tuesday, September 18, 2018 | Sunday, September 23, 2018 | Standard Class | SH-19975 | OFF-PA-10004327 | 143.7 | 3 | 0 | 68.98 |

***Figure 2.15:*** *Formatting Profit*

Similarly, change the decimal places for any columns as required.

**Review your Model:**

Click on the *Model* view to see the loaded query. Right now, only one table is displayed, as shown in the following screenshot:



Orders

Customer ID
Discount
Order Date
Order ID
Product ID
Profit
Quantity
Row ID
Sales
SalesPersonID
Ship Date

***Figure 2.16:*** *Model view with Orders table*

As more queries/tables will be loaded, additional tables will be displayed in this view.

Save your application.

## *Loading Customers and Products tables*

In this section, we will load multiple tables such as the **Customers** and the **Products** tables.

The **Orders** table contains the details of the orders purchased. The **Customers** table contains the details of the customers who made the purchase. The **Products** table contains the product details.

**Query editor can also be used to load tables. It is a better way of loading the data, as you can transform the data too in the same place. Going forward, for rest of the tables, we will use only the Query editor.**

The steps to load the Customers and Product tables are as follows:

On the Power BI desktop, from select **Transform**

The following figure shows the Transform data with Query Editor:

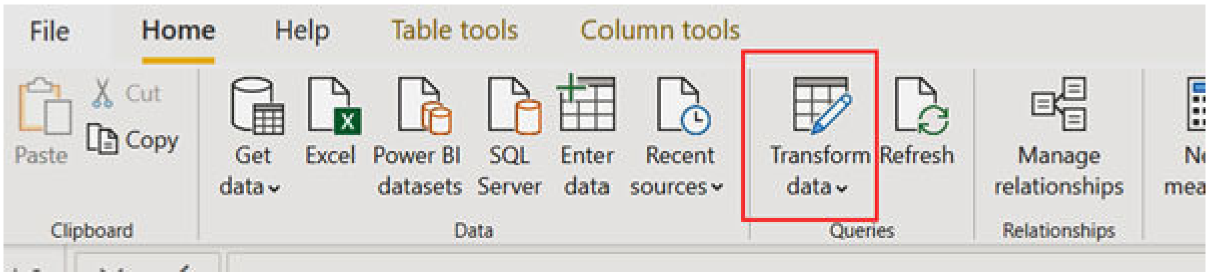


***Figure 2.17:*** *Select Transform data to launch Query Editor*

We will get navigated to the Query Editor. Click on **Recent** since we have previously used the **PowerBI_Data** Microsoft Access database, it will show up in the drop-down list. Select this dataset as shown in the following screenshot:

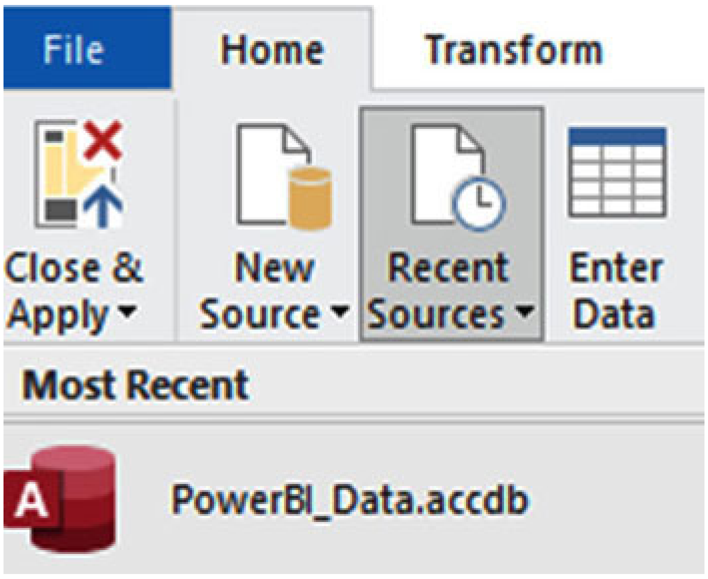
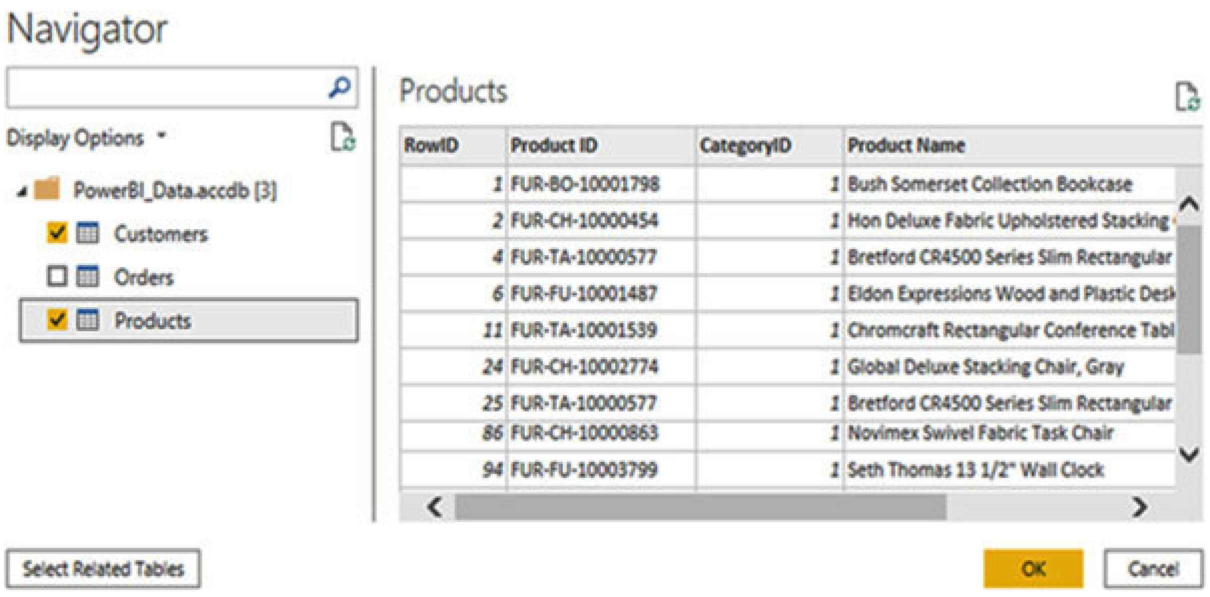


***Figure 2.18:*** *Select PowerBI_Data database*

From the Navigator, select the **Customers** and **Products** tables and click on **OK** as shown in the following screenshot:



| RowID | Product ID | CategoryID | Product Name |
|---|---|---|---|
| 1 | FUR-BO-10001798 | 1 | Bush Somerset Collection Bookcase |
| 2 | FUR-CH-10000454 | 1 | Hon Deluxe Fabric Upholstered Stacking |
| 4 | FUR-TA-10000577 | 1 | Bretford CR4500 Series Slim Rectangular |
| 6 | FUR-FU-10001487 | 1 | Eldon Expressions Wood and Plastic Desk |
| 11 | FUR-TA-10001539 | 1 | Chromcraft Rectangular Conference Tabl |
| 24 | FUR-CH-10002774 | 1 | Global Deluxe Stacking Chair, Gray |
| 25 | FUR-TA-10000577 | 1 | Bretford CR4500 Series Slim Rectangular |
| 86 | FUR-CH-10000863 | 1 | Novimex Swivel Fabric Task Chair |
| 94 | FUR-FU-10003799 | 1 | Seth Thomas 13 1/2" Wall Clock |

***Figure 2.19:*** *Selecting multiple tables*

We will get connected to the tables **Customers** and In the list of queries on the left, we will see three queries – **Customers,** and

Review Customers Query for any Click on **Customers** from the Queries, review the data.

Notice that **Postal Code** is loaded as a decimal number. To verify, click on the **Postal Code** and check the **Data Type** property in the ribbon. Though **Postal Code** contains numerical digits, it is a dimension field and should not be a number.

Change its data type property to text.

Review **Product query** for any Click on the **Products query** from the left and check the data preview. No transformations are required at this time.

Once you are done with the Query Editor. Navigate to the **File** menu and select **Close &** This will apply the query changes and navigate to the Power BI desktop.

Save your application.

Loading multiple tables is a good option if you have large number of tables.

Verifying the loaded

In Power BI desktop, select the data view, to preview the data tables, as shown in the following screenshot:

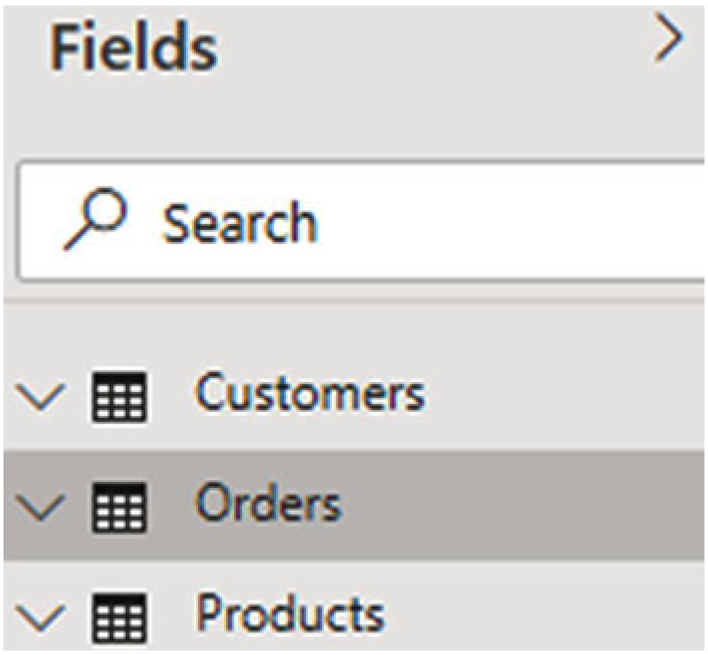


***Figure 2.20:*** *Loaded tables*

Under **Fields** on the right, we can see all the loaded tables.

**Changing the Default summarization for the Products table:**

In the preceding field list of **Products** table, notice the **CategoryID** field with the Summarization symbol (∑). This is a key field and will only be used to create a join. There will be no summarization or aggregation needed on this field. We can change the summarization property of

Change the default summarization property of **CategoryID** by completing the following steps:

In Power BI desktop, select the data view from the left and then from under **Products** table, select navigate to the ribbon **Column tools** and change **Summarization** to **Don't** Notice that this will remove the ∑ symbol for

Similarly, review the other tables as well and remove the **Summarization** where it is not needed.

Save your application.

In the next section, we will load the data from an excel file.

**Review your model:**

Click on the model view to see the loaded queries. We can see three loaded tables, that is, and We can also see a line between **Orders** and **Products** table, which depicts a relationship between the tables, as shown in the following screenshot:

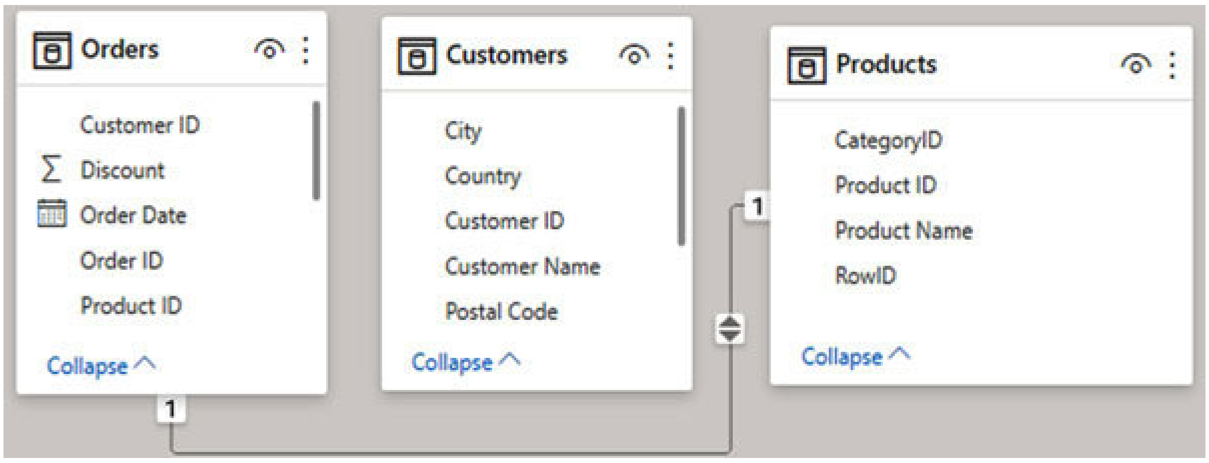


***Figure 2.21:*** *Loaded tables and a relationship between Orders and Products*

Power BI has auto detected a relationship between **Orders** and leave it in place right now. We will cover relationships in the next chapter.

## *Loading data from an Excel file*

In the earlier exercises, we loaded data from the MS-Access database. Power BI also allows combining the data from the other sources.

### LOADING CATEGORY DATA FROM AN EXCEL FILE

Categories and subcategories of various products are stored in an Excel file This file is available in your folder location. We will load this file, following the same process as in the earlier exercises.

The steps to load Category.xls are as follows:

In the Power BI desktop, select **Transform Data** from the ribbon and navigate to Query Editor.

From the Query editor **Home** ribbon, select **New** and from the dropdown list, select Since we are using this data source for the first time, it is not available in the **Recent**

Select **New** as this data source has not been used before, as shown in the following screenshot:

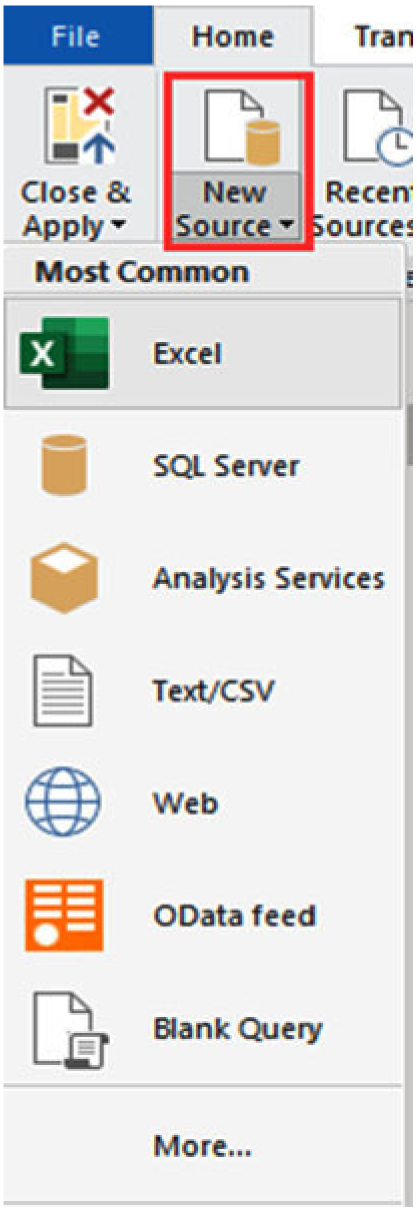


***Figure 2.22:*** *Selecting New Source*

In the browser window, navigate to your enclosed that is,

In the Navigator window, select **Category** and then click on

If due to some reason, Query Editor is unable to recognize the headers in your file, from the **Home** ribbon, select **Use First Row**

**as** This option is available below **Data** as shown in the following screenshot:

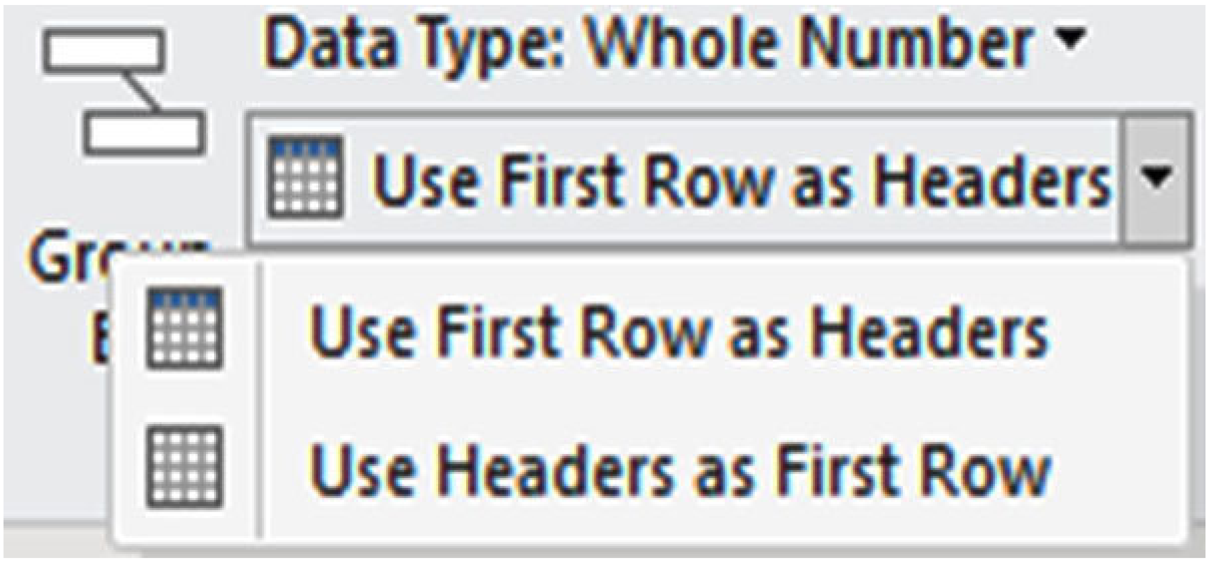


***Figure 2.23:*** *Option to fix headers in Excel file*

Review the loaded query. Notice, even though it is an Excel file, it is loaded just like the database tables.

From the menu, select **File** and select **Close &** This will apply the query changes and navigate to the Power BI desktop.

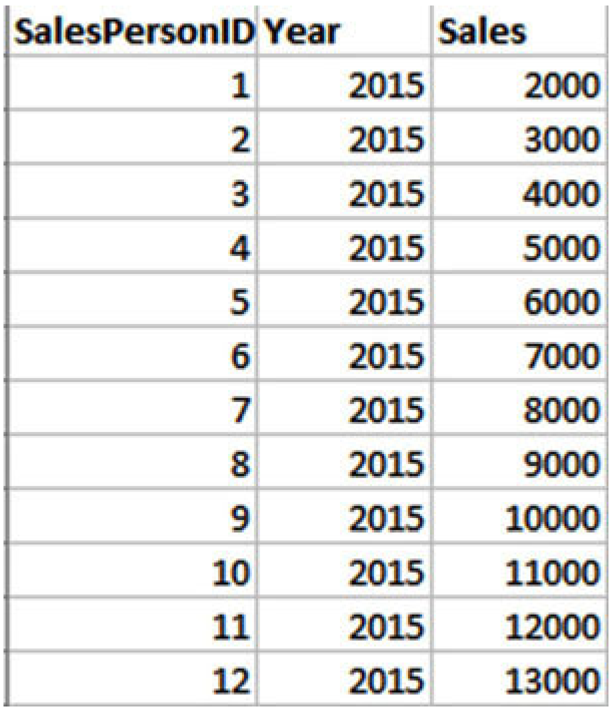
Change the default summarization of **CategoryID** field in the **Category** table to **Don't**

Save your application.

In this exercise, we loaded one Excel file; we can also load multiple Excel files at the same time.

## Loading multiple CSV files from a folder

Power BI can also load multiple files located in a folder. This can be done for the files having the same structure, that is, number of columns, column format, etc. To understand this concept, we will use four CSV files containing the yearly sales targets for the sales persons. These sales targets are created for each of the years, 2015 to 2018 as shown in the following screenshot:



| SalesPersonID | Year | Sales |
|---|---|---|
| 1 | 2015 | 2000 |
| 2 | 2015 | 3000 |
| 3 | 2015 | 4000 |
| 4 | 2015 | 5000 |
| 5 | 2015 | 6000 |
| 6 | 2015 | 7000 |
| 7 | 2015 | 8000 |
| 8 | 2015 | 9000 |
| 9 | 2015 | 10000 |
| 10 | 2015 | 11000 |
| 11 | 2015 | 12000 |
| 12 | 2015 | 13000 |

***Figure 2.24:*** *SalesTarget_2015.csv*

In the enclosed dataset, there are three more similar year-wise files, that is, and These files are stored in a folder named

The steps for loading multiple CSV files from a folder are as follows:

From the Power BI desktop **Home** ribbon, click on **Transform** You will be navigated to Query Editor.

In Query Editor, select **New Source and** In the subsequent, **Get Data** dialog box, select **File and** Select **Connect** as shown in the following screenshot:

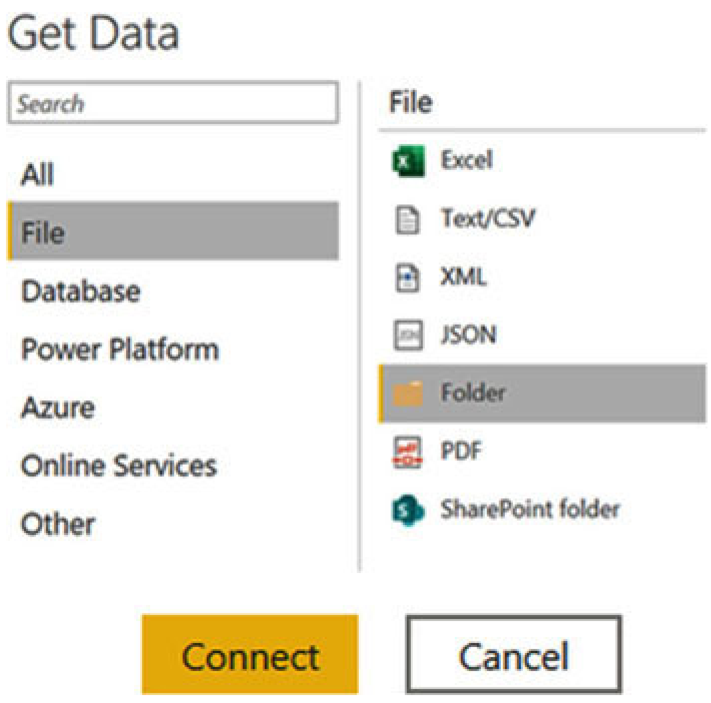
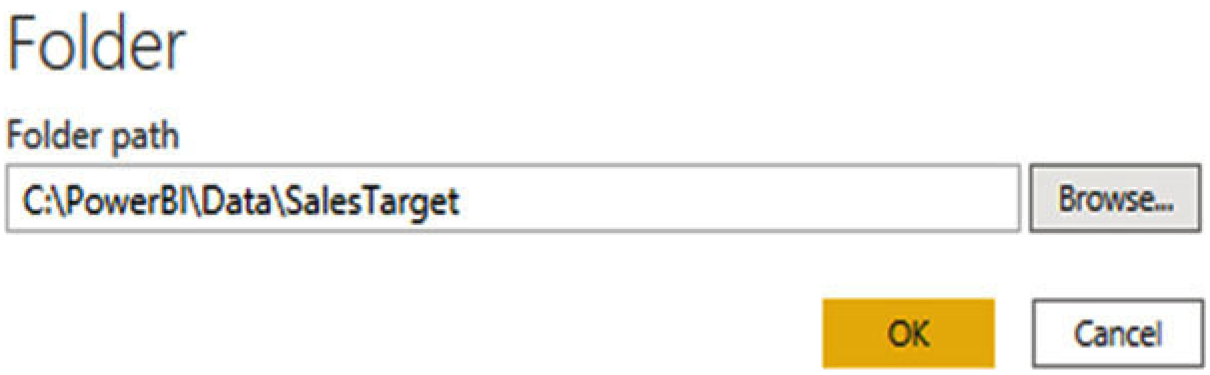


***Figure 2.25:*** *Selecting multiple files in a folder*

Browse to the **SalesTarget** folder. The path should be **C:\PowerBI\Data\SalesTarget** as shown in the following screenshot:

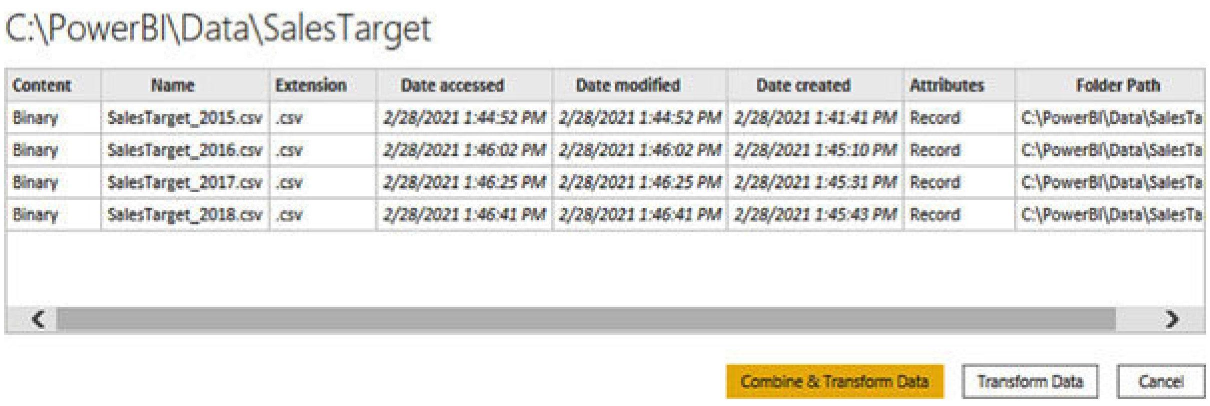

Folder

Folder path

C:\PowerBI\Data\SalesTarget

***Figure 2.26:*** *Specifying folder path*

After the folder path is specified, Power BI lists all the files with a folder path, as shown in the following screenshot. We can see it in the data preview. This helps in identifying the source of the files.


C:\PowerBI\Data\SalesTarget

| Content | Name | Extension | Date accessed | Date modified | Date created | Attributes | Folder Path |
|---|---|---|---|---|---|---|---|
| Binary | SalesTarget_2015.csv | .csv | 2/28/2021 1:44:52 PM | 2/28/2021 1:44:52 PM | 2/28/2021 1:41:41 PM | Record | C:\PowerBI\Data\SalesTa |
| Binary | SalesTarget_2016.csv | .csv | 2/28/2021 1:46:02 PM | 2/28/2021 1:46:02 PM | 2/28/2021 1:45:10 PM | Record | C:\PowerBI\Data\SalesTa |
| Binary | SalesTarget_2017.csv | .csv | 2/28/2021 1:46:25 PM | 2/28/2021 1:46:25 PM | 2/28/2021 1:45:31 PM | Record | C:\PowerBI\Data\SalesTa |
| Binary | SalesTarget_2018.csv | .csv | 2/28/2021 1:46:41 PM | 2/28/2021 1:46:41 PM | 2/28/2021 1:45:43 PM | Record | C:\PowerBI\Data\SalesTa |

***Figure 2.27:*** *Files and path displayed*

At the bottom, it provides with two options – **Combine & Transform Data** or **Transform** Let's select **Combine & Transform** In the **Combine Files** dialog box, we can verify the format of each of the file.

The **Delimiter** option in the **Combine Files** dialog box displays the different file delimiters, you can select any one, depending on your type of the file.

Since we have loaded the CSV files from a folder, in Query Editor, it displays the multiple transformation options for these files.

You should see five queries loaded, that is, **Category,** and

Select the **SalesTarget** query from the left. In the data preview, observe that **Year** is loaded as a number. Select the **Year** column and change the **Data Type** to

In the **Change Column Type** dialog box, select **Replace**

From the **File** menu, select **Close &**

In the Power BI desktop, in Data view, preview the data for You will observe that it has combined all the files and loaded them as one.

Loading the files from the folder is a good option to load multiple similar files.

## ***Loading data from MS SQL server***

In the previous sections, we loaded the tables from the MS-Access database. In this section, we will load the table from the relational database SQL Server. It deals with some important concepts; we will first discuss those concepts.

## *Import and DirectQuery*

When connecting to the relational databases, such as SQL server, we get two options on **Data Connectivity mode** – **Import** and **DirectQuery** as shown in the following screenshot:

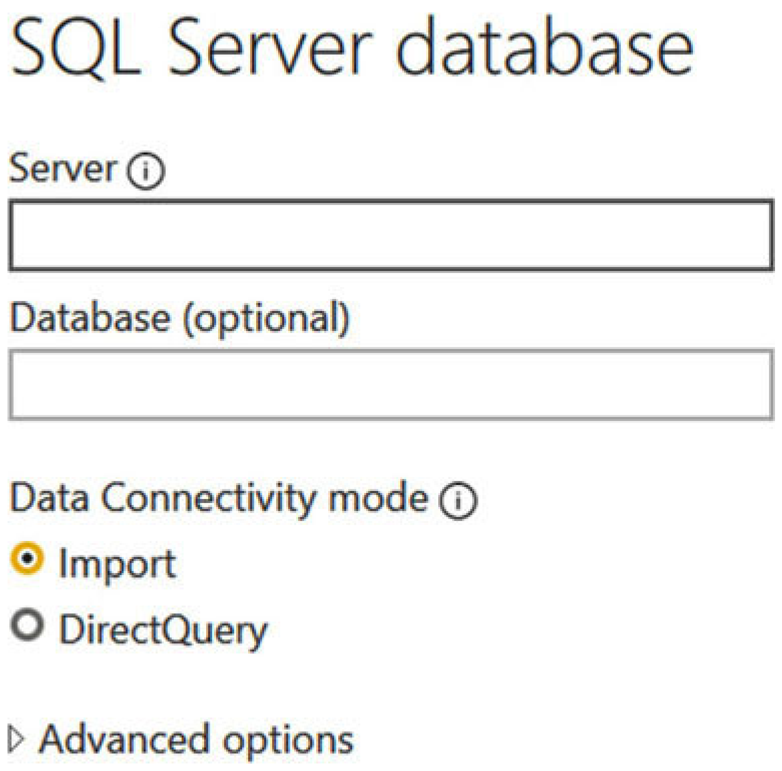


***Figure 2.28:*** *Query modes in SQL Server*

When the import query option is selected, the tables and columns from the source data are imported in the Power BI desktop. The Power BI developer and users work with this imported data. If the underlying data changes, the import needs to be refreshed. Import is the default option. MS-Excel and the other file-based data sources are always imported.

In this mode, the tables and columns are not imported in the Power BI desktop. The connection is made directly with the data source. When connecting with the larger datasets, the DirectQuery option should be used.

This option does not utilize the full capability of Power BI.

Loading the Sales Person table from SQL

We will load **Sales Person** table which is stored in the SQL Server database. This table contains the Sales Person's details such as **SalesPersonID** and

**To perform this exercise, you will need SQL Server. You can install the express edition from the following link:**

**If you don't wish to use SQL Server, this table is also present in your enclosed Excel files. We have provided two versions of the data model – Chapter2_DbConn uses the Excel version of the Sales Person table.**

**If you have SQL Server installed and configured, you can use Chapter2_SQLServerConnection.**

**In the subsequent chapters, we will be using the excel version of the Sales Person table, so that even if you don't have SQL Server, you can still follow the chapters.**

The steps to loading the Sales Person table from SQL Server are as follows:

On the Power BI desktop, from the **Home** tab, select **Transform**

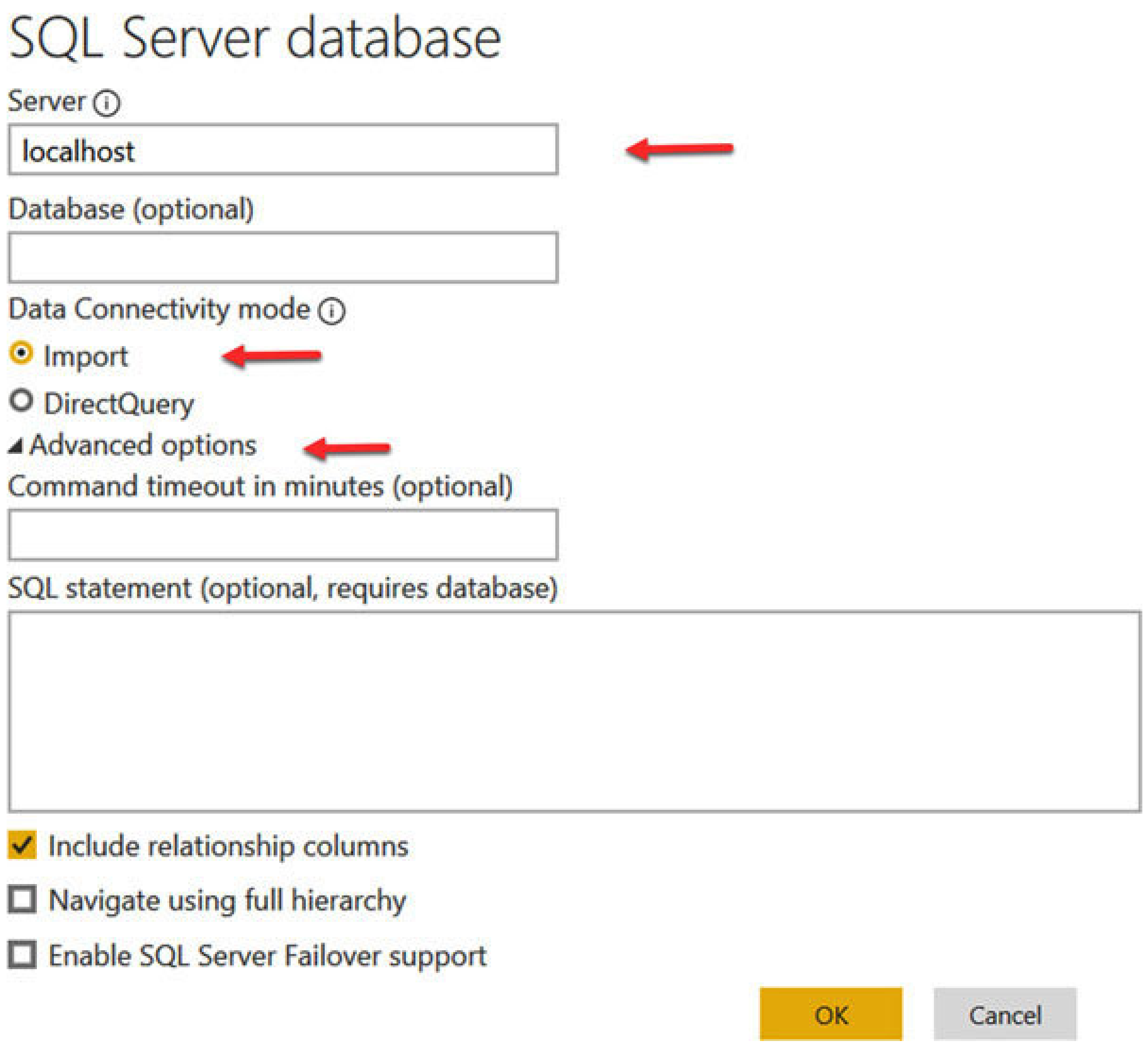
From the Query Editor ribbon, select **New Source** and select **Sql**

In the SQL Server database screen, specify your SQL Server credentials:

Provide the **Server Database name** is optional.

In **Data Connectivity** select

The **Advanced** options in this screen can be used to write the custom SQL query, as shown in the following screenshot:


SQL Server database

Server: localhost

Database (optional):

Data Connectivity mode: [✓ Import] DirectQuery

Advanced options

Command timeout in minutes (optional):

SQL statement (optional, requires database):

[✓] Include relationship columns

[ ] Navigate using full hierarchy

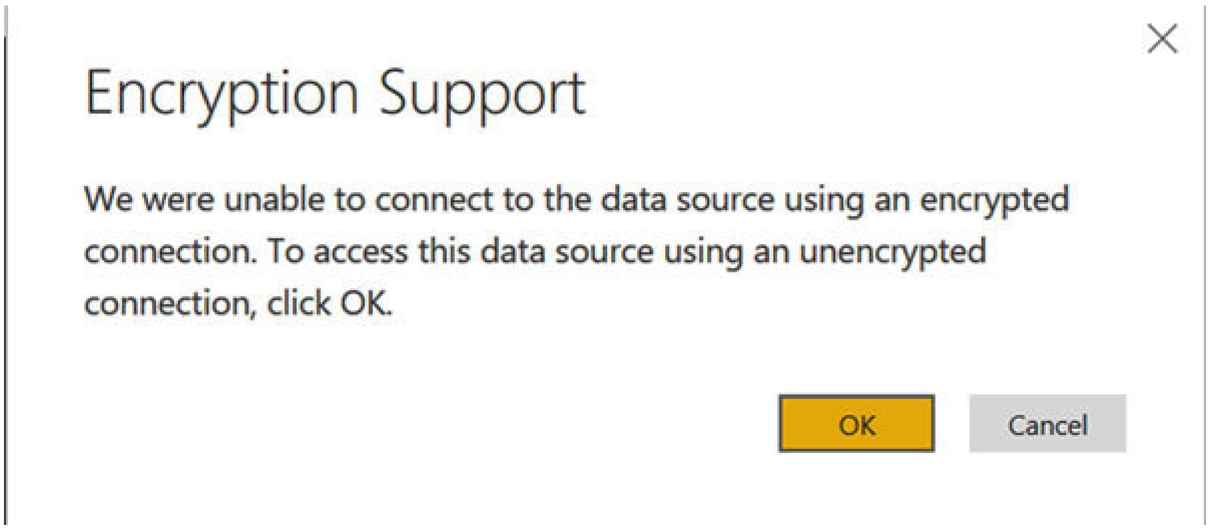
[ ] Enable SQL Server Failover support

***Figure 2.29:** SQL Server connection and query parameters*

Click on

In the next SQL Server database screen, review your credentials and select

You may get a warning on the encryption, as shown in the following figure; click on


Encryption Support

We were unable to connect to the data source using an encrypted connection. To access this data source using an unencrypted connection, click OK.

*Figure 2.30: Encrypted connection warning*

In the Navigator screen, select the database and the table that you wish to connect to. Select the table **Sales**

Select

From the **File** menu, select **Close &**

In the model view, we can see that Power BI has auto-detected a relationship between, **Sales Person** and **SalesTarget** based on Leave it the way it is for now. We will discuss relationships in the next chapter.

Save the application.

Similar steps can be used to connect to any other relational database, such as Oracle.

## ***Creating static table in Power BI***

Sometimes we may have to use the data which is not present in the database or files, in such cases, create a table in Power BI and enter the data into this table. Such tables are called Static tables.

We can take the data from the existing tables in our model and create such tables.

The characteristics of a static table are as follows:

Static tables contain static data and cannot be refreshed as they are not connected to the data source.

Static tables are created using the **Enter Data** functionality in the Power BI desktop.

Static tables are created by either typing the values manually or by copying and pasting the data from an existing table.

A static table, once loaded, can be in a relationship with the other tables/queries in the data model.

Static tables are usually smaller tables and can be created on the fly.

## ***Why do we need a static table in our model?***

For the sake of an example, consider that **SalesPersonIDs** in the **Sales Person** table doesn't have Locations associated with them, and this information is not available in the database, but client has provided it in an email.

We will create a static table with two columns – **SalesPersonID** and We will use **SalesPersonID** from the **Sales Person** table and enter **Location** for each of these manually.

This table will be in a relationship with the **Sales Person** table.

The steps to create a static table for the Sales Person and his respective location are as follows:

As a first step, we will get the **SalesPersonIDs** from the **Sales Person** table.

In the Power BI desktop, select the **Report view** from the left. From the bottom, create a new report page by clicking on the **+** symbol.

Navigate to **Visualizations** and select From select **Sales Person** and then select **SalesPersonID,** as shown in the following screenshot:

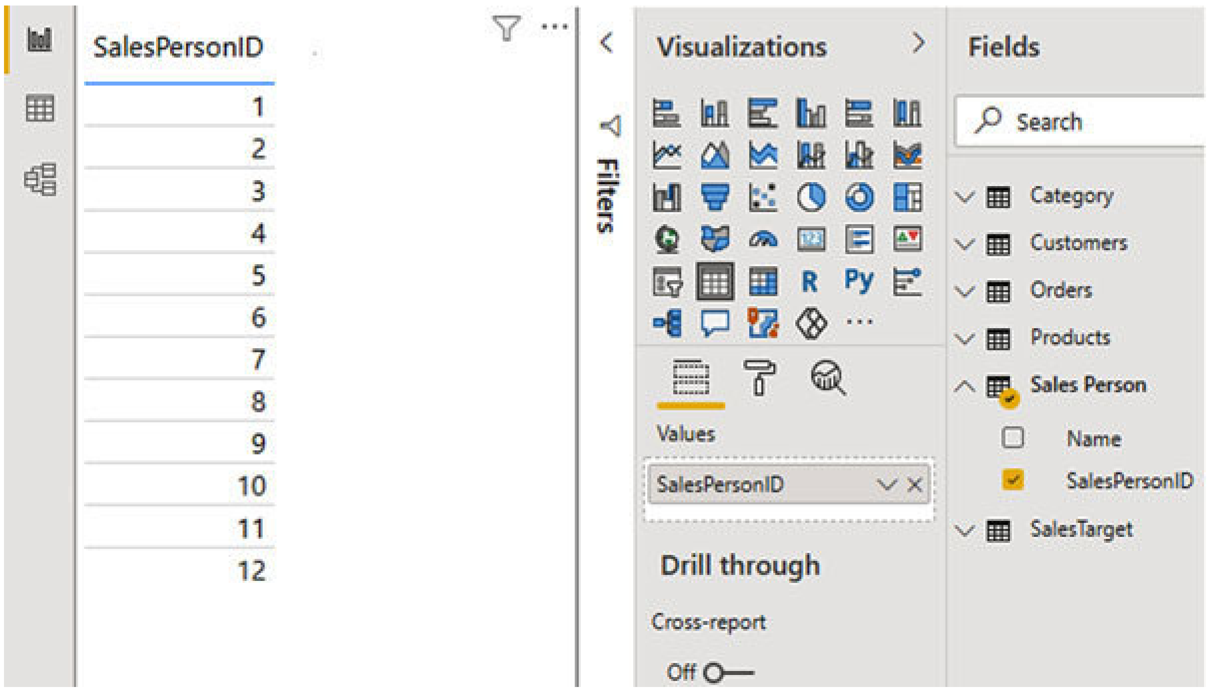


***Figure 2.31:*** *Displaying SalesPersonID from Sales Person*

This will display the unique **SalesPersonIDs** in the table visualization.

Export the **SalesPersonIDs** from the visualization to a CSV or Excel file as shown in the following screenshot:

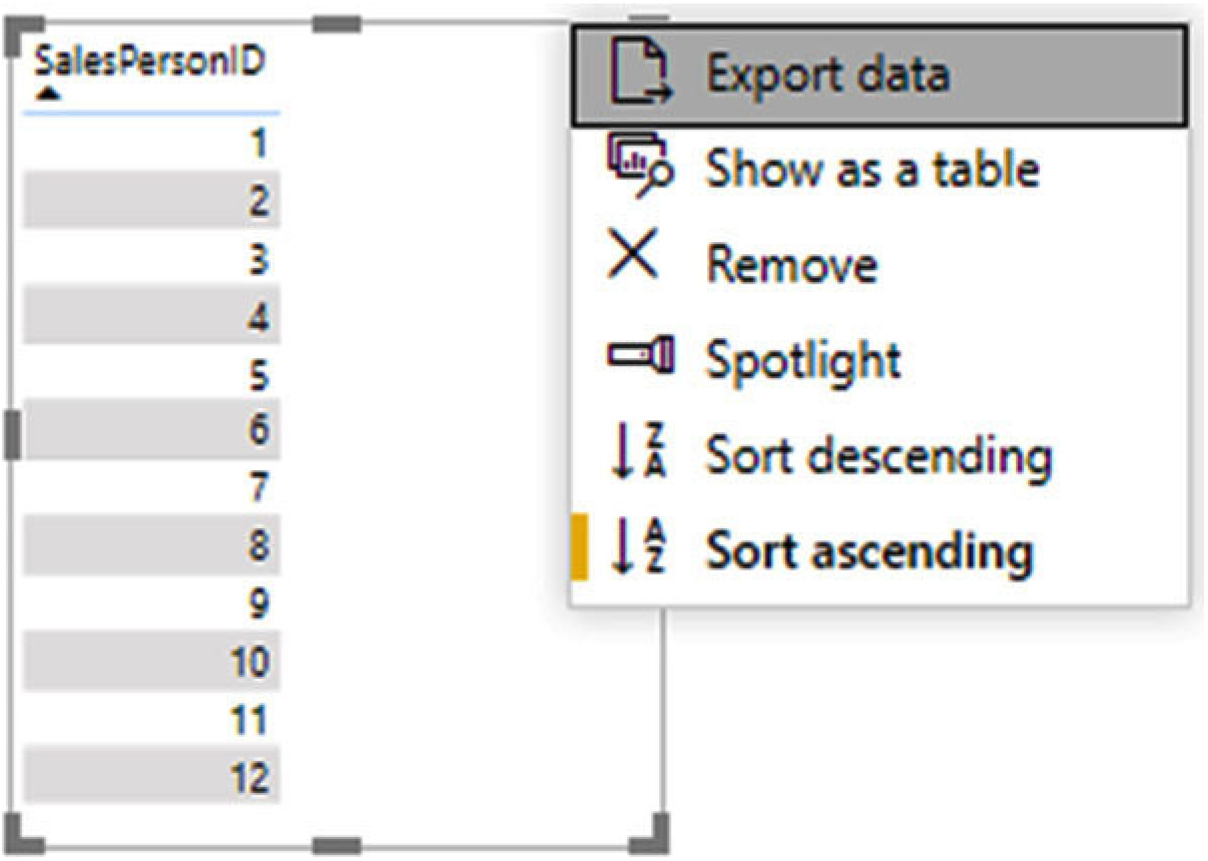


***Figure 2.32:*** *Exporting to file*

Save this file with your other data files. Save this file as **SalesPersonID_Exporteddata.csv** under

Navigate to the preceding path in Windows Explorer. Open the CSV file and copy all the rows of the data (don't copy the column header).

Now, to create the static table, complete the following steps:

On the Power BI desktop, from the **Home** tab, You will see the **Create Table** dialog box, as shown in the following screenshot:

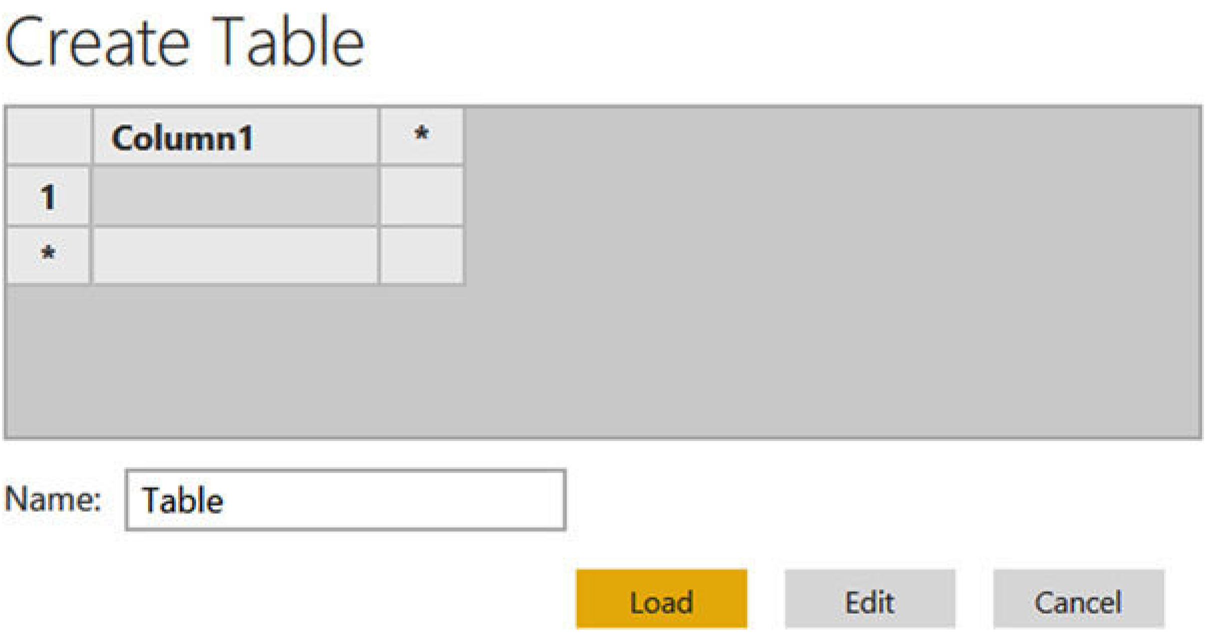


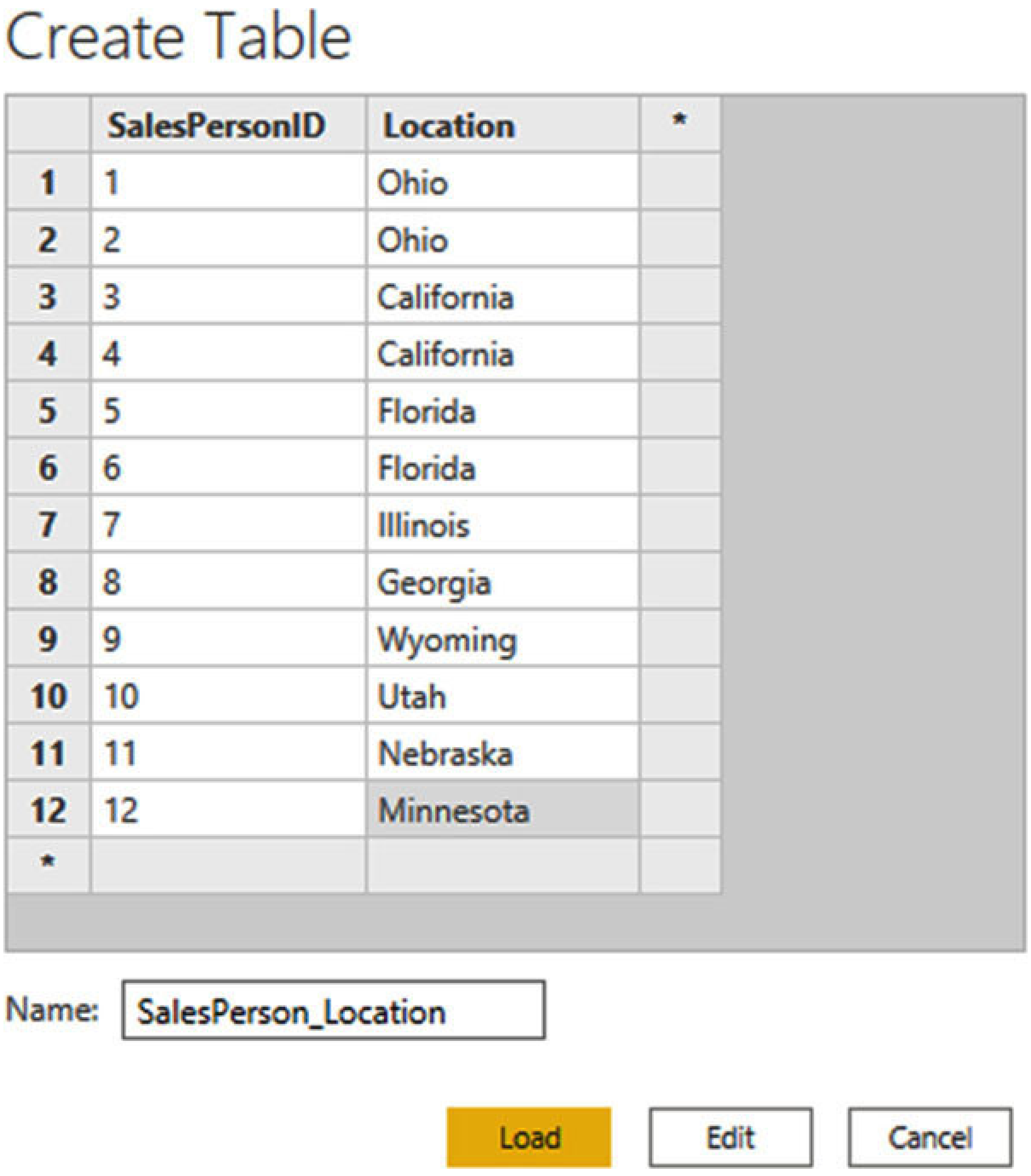
***Figure 2.33:*** *Create Table dialog box*

Double click on **Column1** and name it

Double click on the next column displayed as * and name it as

Right click on the first cell below the SalesPersonID header and paste the data you have copied from the

In the **Location** column, enter the values shown in the following screenshot; name the table as



|  | SalesPersonID | Location | * |
|---|---|---|---|
| 1 | 1 | Ohio |  |
| 2 | 2 | Ohio |  |
| 3 | 3 | California |  |
| 4 | 4 | California |  |
| 5 | 5 | Florida |  |
| 6 | 6 | Florida |  |
| 7 | 7 | Illinois |  |
| 8 | 8 | Georgia |  |
| 9 | 9 | Wyoming |  |
| 10 | 10 | Utah |  |
| 11 | 11 | Nebraska |  |
| 12 | 12 | Minnesota |  |
| * |  |  |  |

Name: SalesPerson_Location

***Figure 2.34:*** *SalesPerson_Location Static table*

Click on This will create a new table We can see it under

In the model view, Power BI has auto detected the relationship between the **SalesPerson_Location** and **Sales Person** table.

**SalesPerson_Location table is also enclosed as an excel file in your enclosed data file. You can use it as needed.**

Static tables are a useful way to add smaller datasets in your Power BI model.

**Review the data model:**

After loading all the preceding tables, the data model will look like the following screenshot:

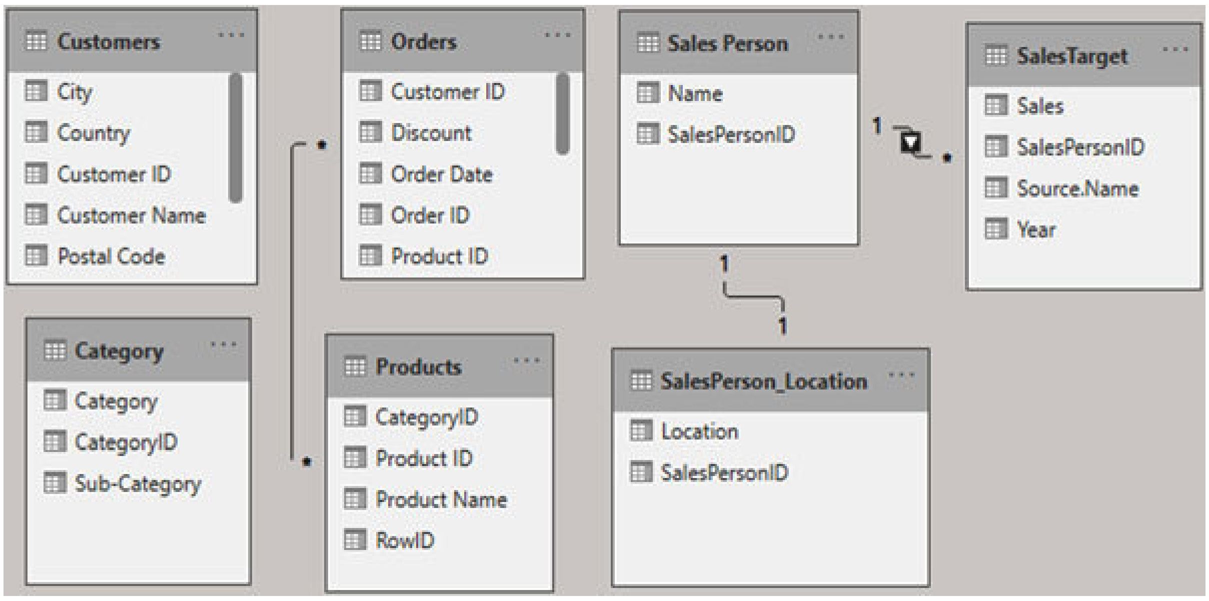


***Figure 2.35:*** *Final data model*

The data model displays the pictorial representation of the loaded tables and relationships. It displays all the tables and the relationships auto-detected by Power BI between **Orders** and **Products** and between the Sales Person's tables.

## ***Bonus section***

In this section, we will learn about some more additional ways to connect to the data in Power BI. The tables loaded in this section will not be used in the data model that we created earlier.

These exercises will be presented as separate applications in the enclosed files.

## ***Connecting to web data source***

Sometimes we may use web data to display in Power BI reports. To load the data from the web, Power BI allows connecting to the web data using the basic authentication.

In this section, we will connect to the list of Overall Solo winners of the Race Across America. This data is available on the Web at the following URL:

https://en.wikipedia.org/wiki/Race_Across_America

The steps to load the web data are as follows:

(This example is only for demonstration purposes to show how to load the data from the Web. Refer to enclosed file

Launch the Power BI desktop.

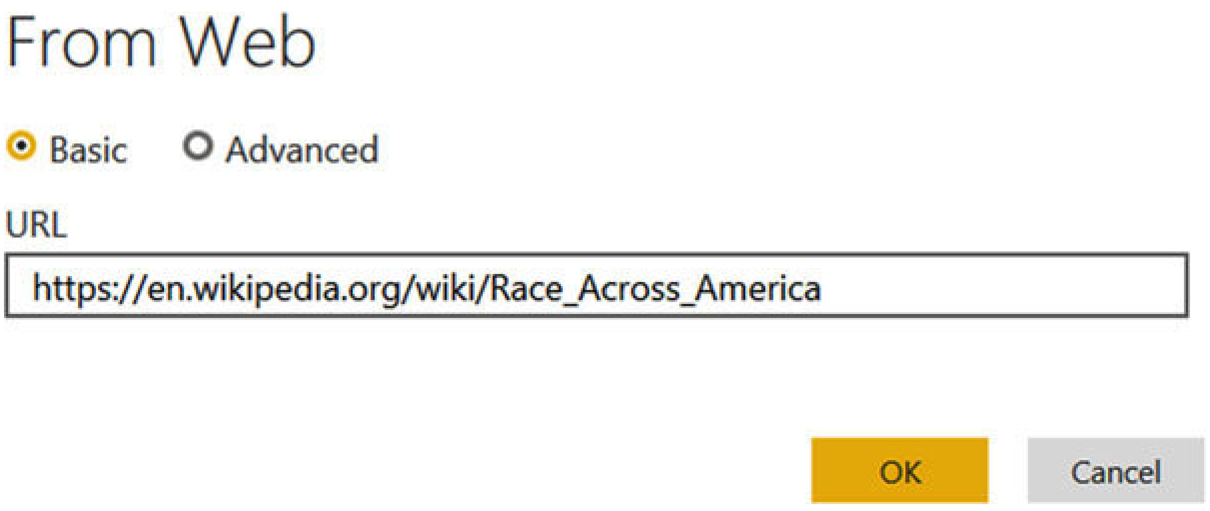
On the Power BI desktop, select **Transform Data** from You will get navigated to Query Editor.

From the Query Editor ribbon, select **New Source** and select

On the **From Web** dialog box, paste the URL:

https://en.wikipedia.org/wiki/Race_Across_America

Click on as shown in the following screenshot:



From Web

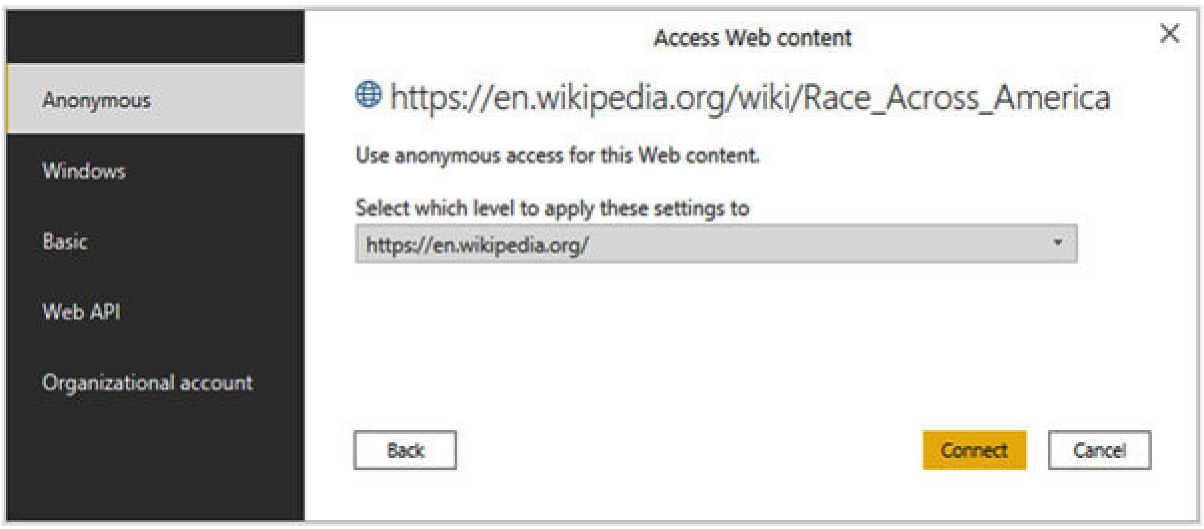
Basic [✓] Advanced

URL

https://en.wikipedia.org/wiki/Race_Across_America

***Figure 2.36:*** *URL path of the web data*

In the **Access Web Content,** choose the access method of your choice or select **Anonymous** and select as shown in the following screenshot:



Access Web content

https://en.wikipedia.org/wiki/Race_Across_America

Use anonymous access for this Web content.

Select which level to apply these settings to

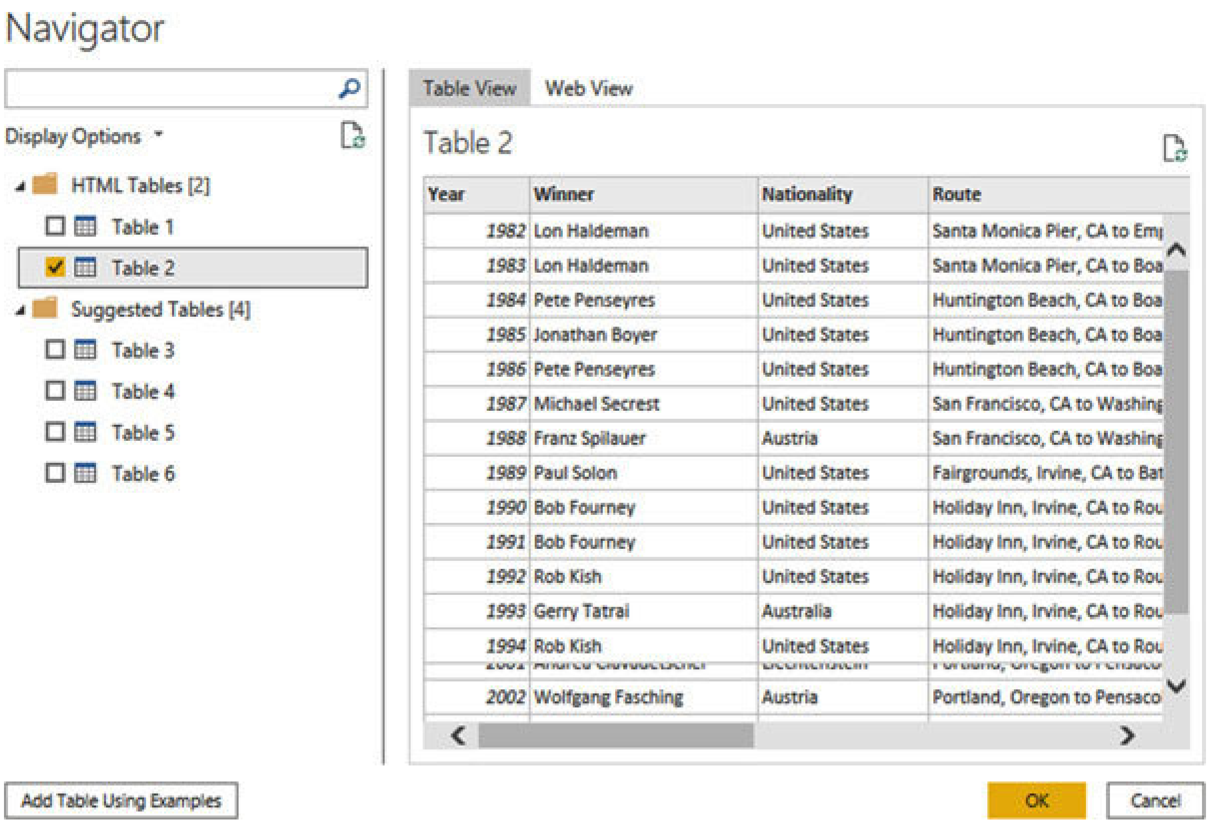
https://en.wikipedia.org/

*__Figure 2.37:__ Access methods for web data*

The Navigator screen will show the list of tables available from this URL.

You can select any table of interest. For the sake of this example, we will select **Table2,** as shown in the following screenshot:

Click on


Navigator

Display Options

- HTML Tables [2]
  - Table 1
  - ✓ Table 2
- Suggested Tables [4]
  - Table 3
  - Table 4
  - Table 5
  - Table 6

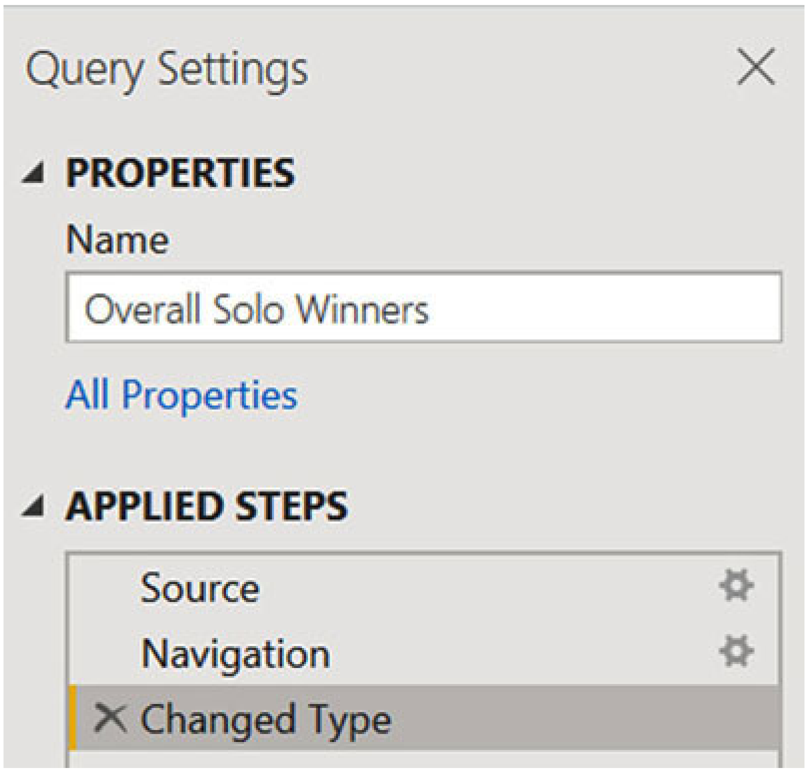
Table View | Web View

Table 2

| Year | Winner | Nationality | Route |
|---|---|---|---|
| 1982 | Lon Haldeman | United States | Santa Monica Pier, CA to Em |
| 1983 | Lon Haldeman | United States | Santa Monica Pier, CA to Boa |
| 1984 | Pete Penseyres | United States | Huntington Beach, CA to Boa |
| 1985 | Jonathan Boyer | United States | Huntington Beach, CA to Boa |
| 1986 | Pete Penseyres | United States | Huntington Beach, CA to Boa |
| 1987 | Michael Secrest | United States | San Francisco, CA to Washing |
| 1988 | Franz Spilauer | Austria | San Francisco, CA to Washing |
| 1989 | Paul Solon | United States | Fairgrounds, Irvine, CA to Bat |
| 1990 | Bob Fourney | United States | Holiday Inn, Irvine, CA to Rou |
| 1991 | Bob Fourney | United States | Holiday Inn, Irvine, CA to Rou |
| 1992 | Rob Kish | United States | Holiday Inn, Irvine, CA to Rou |
| 1993 | Gerry Tatrai | Australia | Holiday Inn, Irvine, CA to Rou |
| 1994 | Rob Kish | United States | Holiday Inn, Irvine, CA to Rou |
| [illegible] | [illegible] | [illegible] | [illegible] |
| 2002 | Wolfgang Fasching | Austria | Portland, Oregon to Pensaco |

*__Figure 2.38:__ Data preview of web data*

From the **Query Settings** (on the right), change the name of the Query to **Overall Solo Winners,** as shown in the following screenshot:


Query Settings

PROPERTIES

Name: Overall Solo Winners

All Properties

APPLIED STEPS

- Source
- Navigation
- Changed Type

***Figure 2.39:*** *Renaming the Query*

Review the columns. The year is loaded as text, change the data type of the year to date.

From the **File** menu, select **Close &** You will get navigated to the Power BI desktop.

In the model view, see the table loaded.

Save this file as

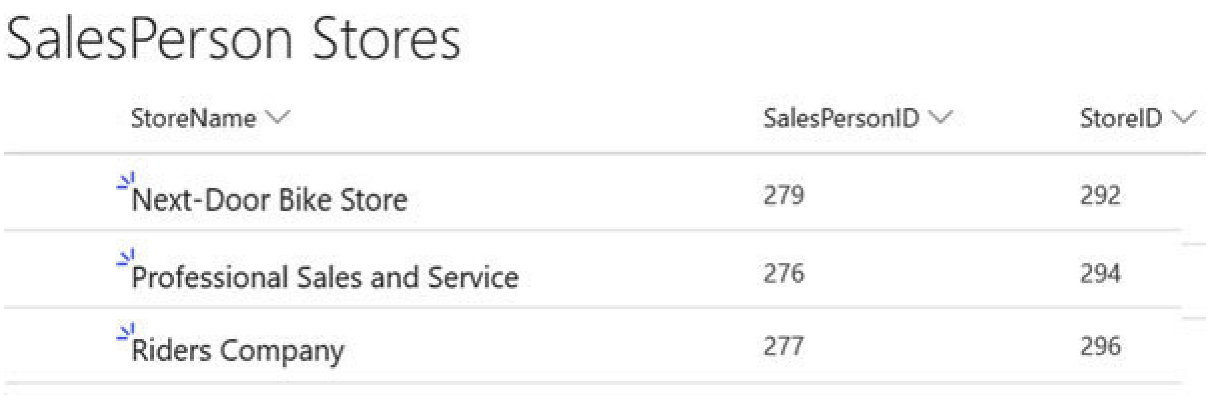
Using the preceding steps, we can connect to any data on the Web.

## *Connecting and loading the data from Microsoft SharePoint*

In this section, we will see how Power BI connects with SharePoint. We will be creating a separate Power BI application to demonstrate this example. Refer to **Chapter2_SharePointConnection.pbix** in your apps folder.

*The explanation of Microsoft SharePoint is beyond the scope of this book. If you are interested to learn, checkout Microsoft's documentation that is available online.*

We will be using the list in SharePoint, which is created in the following format, as shown in the following screenshot:



SalesPerson Stores

| StoreName | SalesPersonID | StoreID |
|---|---|---|
| Next-Door Bike Store | 279 | 292 |
| Professional Sales and Service | 276 | 294 |
| Riders Company | 277 | 296 |

***Figure 2.40:*** *SharePoint list created using SharePoint application.*

In SharePoint, a List is a collection of data that provides a flexible way to organize the information.

If you have SharePoint available through your organization, create a similar list to follow this example. The connection to SharePoint can be made in the following two ways:

**OData Feed:** The **Open Data protocol** accesses a data source or a database, by using a specially constructed URL. It permits for an easier method for connecting and working with the data sources hosted within an organization. It is a protocol for accessing data over the web.

**SharePoint List:** A list in SharePoint is a collection of data that gives us a flexible way to organize information. It is similar to a table in database or Excel.

Microsoft SharePoint is used by many corporations to organize their documents.

**Connecting to SharePoint using ODataFeed:**

For this exercise, we will be using the list created in SharePoint. This list is called **Sales Person Stores** and contains the details such as and

**Getting the OData URL from**

We need to get the OData URL, which we will be connecting to from Power BI. To get the URL, navigate to your SharePoint site and use the path similar to the following:

**https:///_vti_bin/ListData.svc/**

Here, **servername** is your SharePoint server name and **listname** is the name of the list present on your server.

In my instance, it looks like the following:

https://techkonceptsolutions.sharepoint.com/_vti_bin/ListData.svc/SalesPersonStores

The steps to connect to the SharePoint List using ODataFeed are as follows:

Launch the Power BI desktop.

From the **Home** tab, select **Get Data** and select From the **Get Data** dialog box, select **Other** and choose **OData Feed,** as shown in the following screenshot:

Select

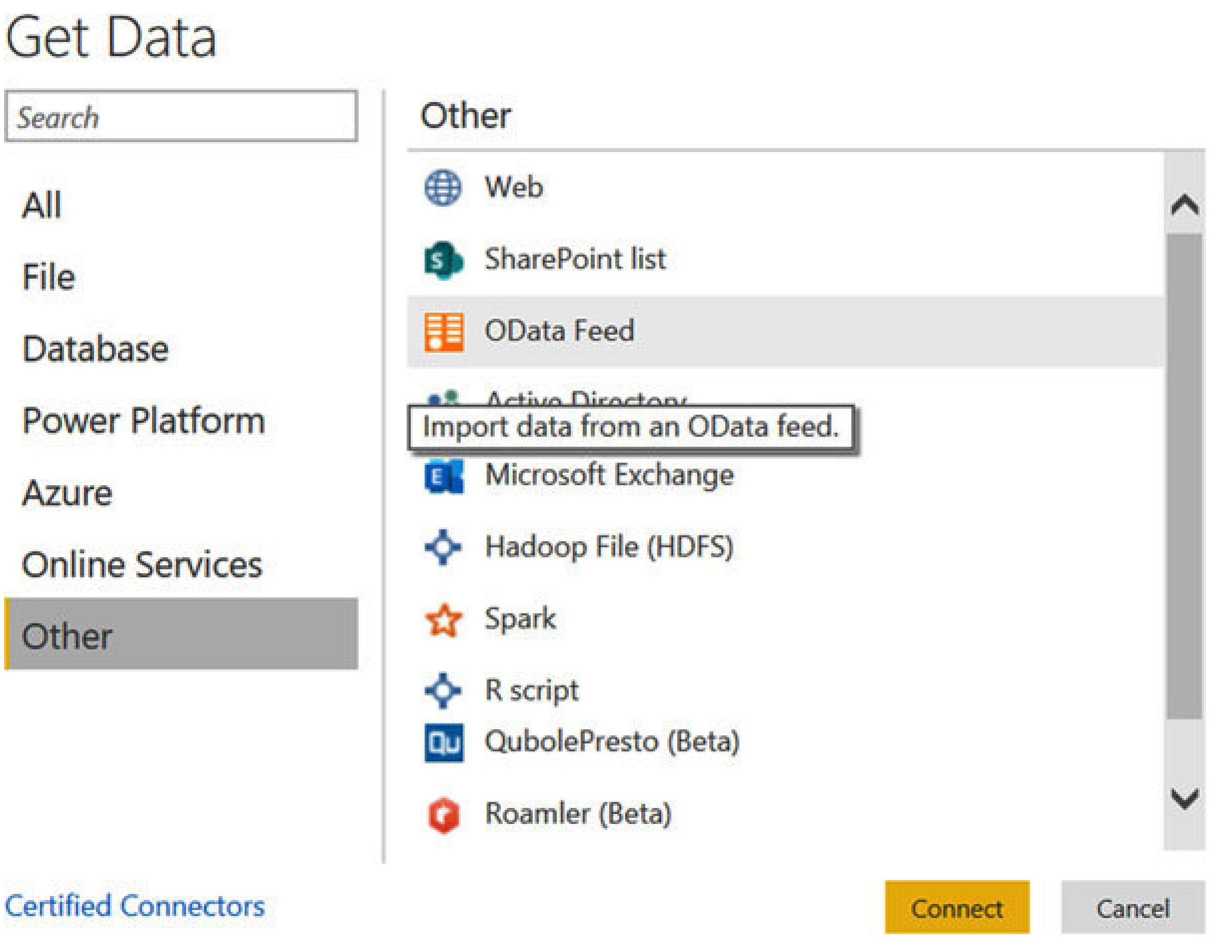
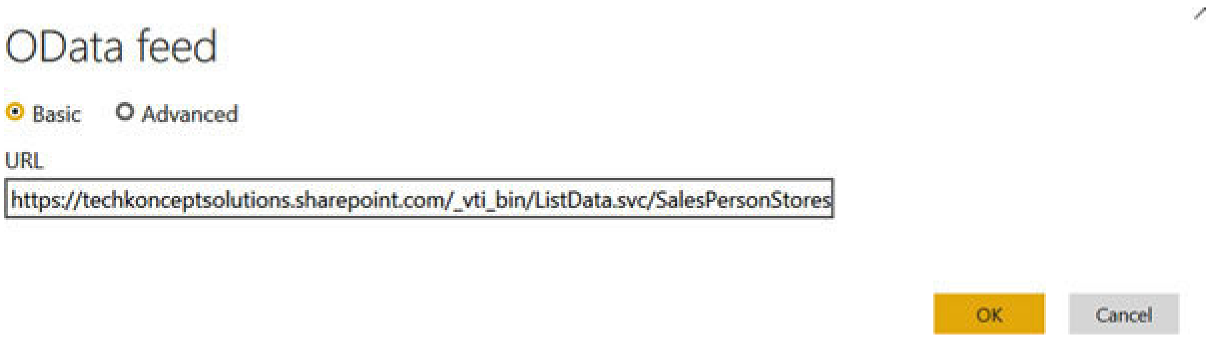


***Figure 2.41:*** *Connecting to SharePoint*

Provide the URL generated earlier and click on as shown in the following screenshot:



URL
https://techkonceptsolutions.sharepoint.com/_vti_bin/ListData.svc/SalesPersonStores

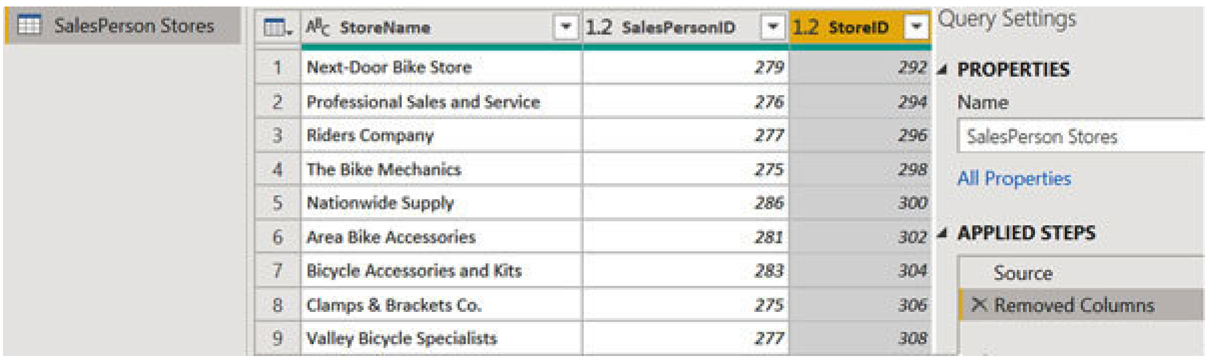
***Figure 2.42:*** *URL specified in OData feed*

If you get a connection error, follow the steps mentioned at the end of this section.

Once connected, the preview of the data will be displayed. Click on **Transform** if you wish to transform the data in Power Query Editor.

Remove the extra columns, so that you have only and Change the Query name under **Properties** to **SalesPerson**

**Stores** query after removing the extra columns, is shown in the following screenshot:



SalesPerson Stores

| | StoreName | SalesPersonID | StoreID |
|---|---|---|---|
| 1 | Next-Door Bike Store | 279 | 292 |
| 2 | Professional Sales and Service | 276 | 294 |
| 3 | Riders Company | 277 | 296 |
| 4 | The Bike Mechanics | 275 | 298 |
| 5 | Nationwide Supply | 286 | 300 |
| 6 | Area Bike Accessories | 281 | 302 |
| 7 | Bicycle Accessories and Kits | 283 | 304 |
| 8 | Clamps & Brackets Co. | 275 | 306 |
| 9 | Valley Bicycle Specialists | 277 | 308 |

Query Settings

PROPERTIES

Name: SalesPerson Stores

All Properties

APPLIED STEPS

Source

Removed Columns

***Figure 2.43:*** *SalesPerson Stores data preview*

From the **File** menu, select **Close &**

In the model view, you can see that the **SalesPerson Stores** table is loaded.

Save your application.

Many organizations use the SharePoint applications; you can use the preceding steps to connect to the SharePoint list.

**Resolving the OData Feed connection error:**

Complete the following steps, if you get a connection error after the SharePoint OData URL is provided:

On the Power BI desktop, from navigate to **Transform Data,** and from the dropdown list, select **Data source** as shown in the following screenshot:

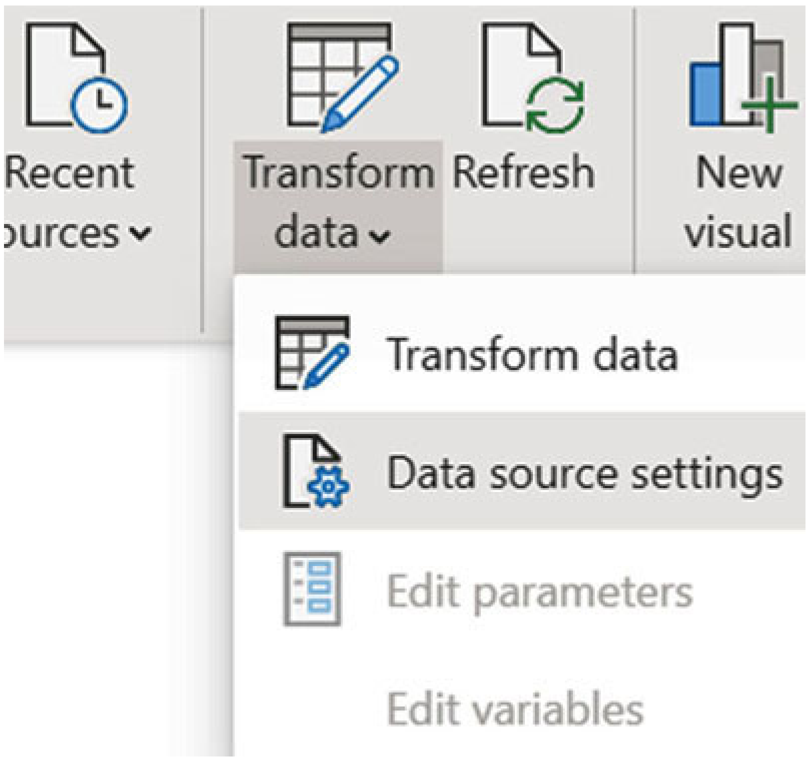


***Figure 2.44:*** *Data source settings*

In the **Data source** select your URL and click on **Edit** In the **Edit Permissions** box, provide your SharePoint connection credentials, as shown in the following screenshot:

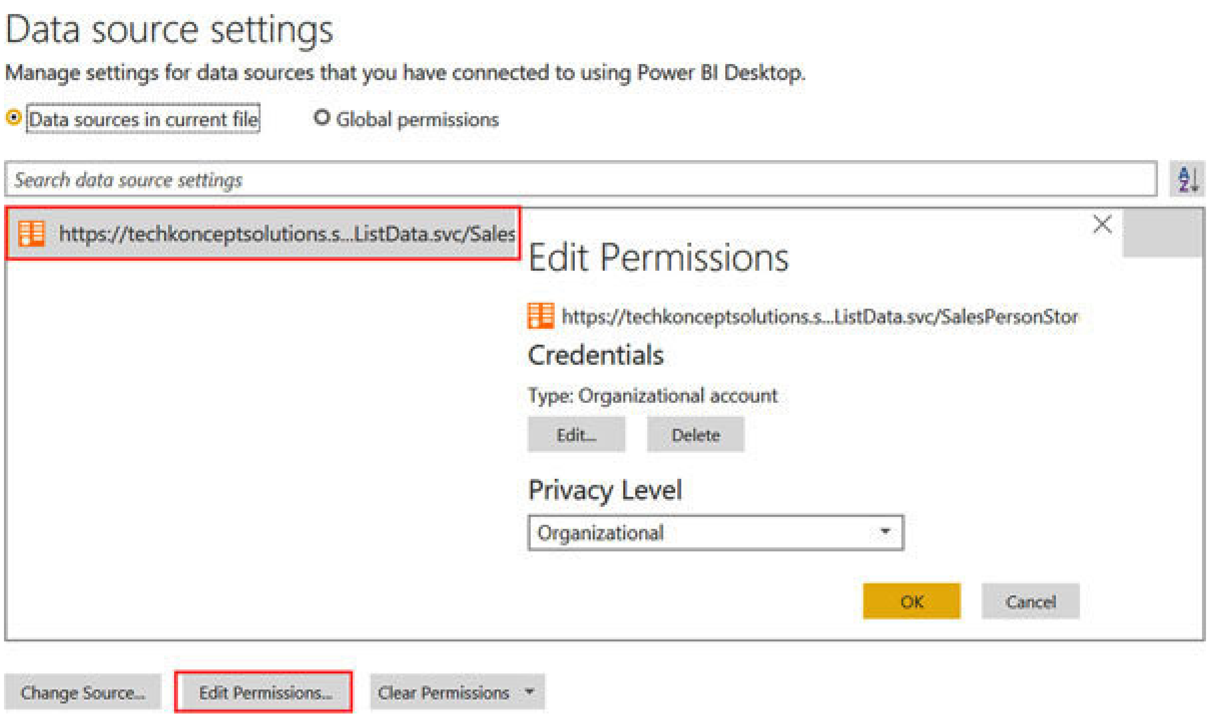

Data source settings
Manage settings for data sources that you have connected to using Power BI Desktop.
Data sources in current file
Global permissions
Search data source settings
https://techkonceptsolutions.s...ListData.svc/Sales
Edit Permissions
https://techkonceptsolutions.s...ListData.svc/SalesPersonStor
Credentials
Type: Organizational account
Edit...
Delete
Privacy Level
Organizational
OK
Cancel
Change Source...
Edit Permissions...
Clear Permissions

***Figure 2.45:*** *Providing permission Credentials*

Error in connection usually occurs due to the incorrect or missing credentials. If you fix the permissions, the error will be resolved.

**Connecting to the SharePoint Lists using the SharePoint connector:**

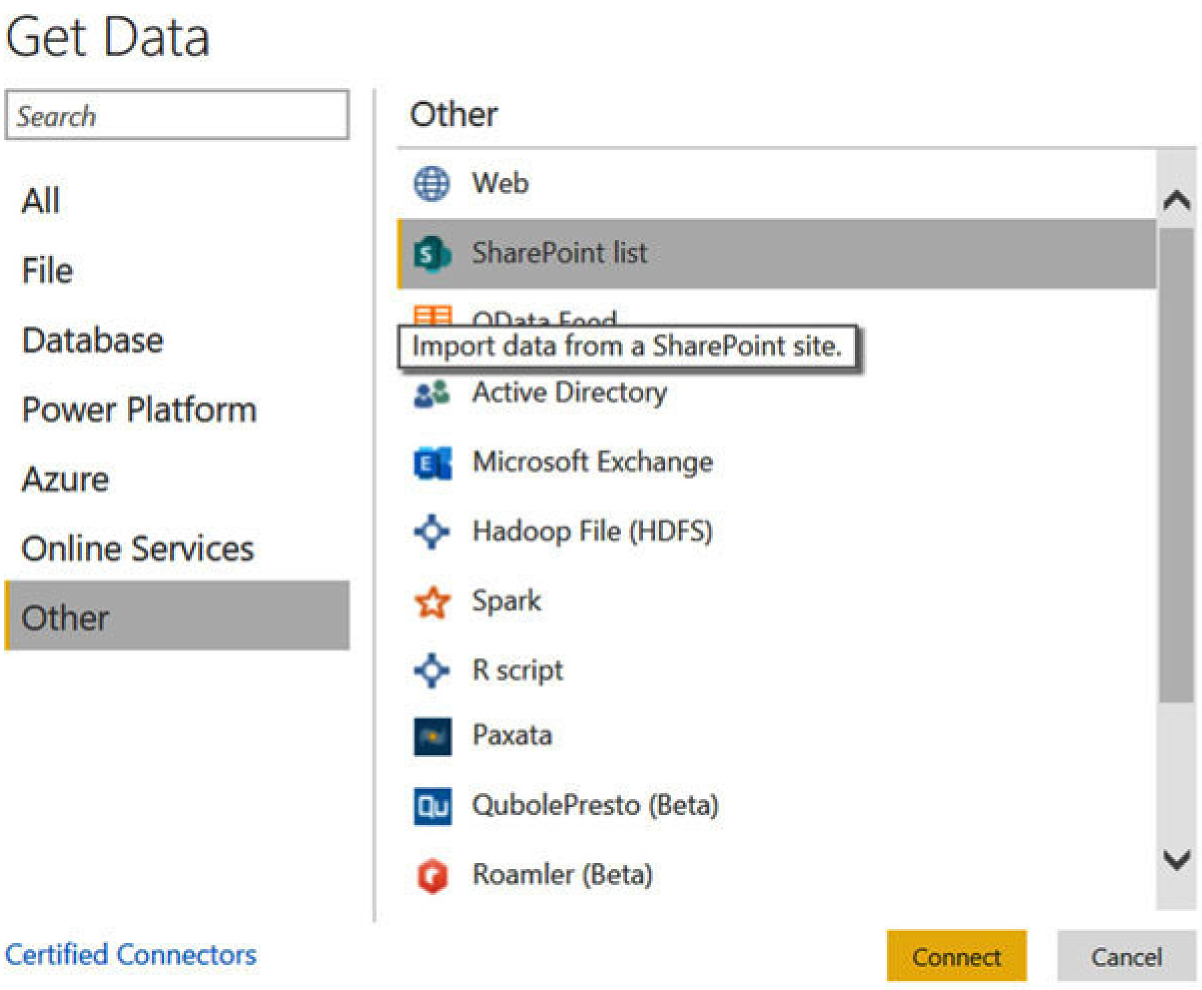
In this example, we will connect to the SharePoint lists using the SharePoint connector.

The steps to connect to the SharePoint List using SharePoint connector are as follows:

Use the same application created in the previous section.

From the **Home** tab, select **Get Data** and select From the **Get Data** dialog box, select **Other** and choose **SharePoint** as shown in the following screenshot:

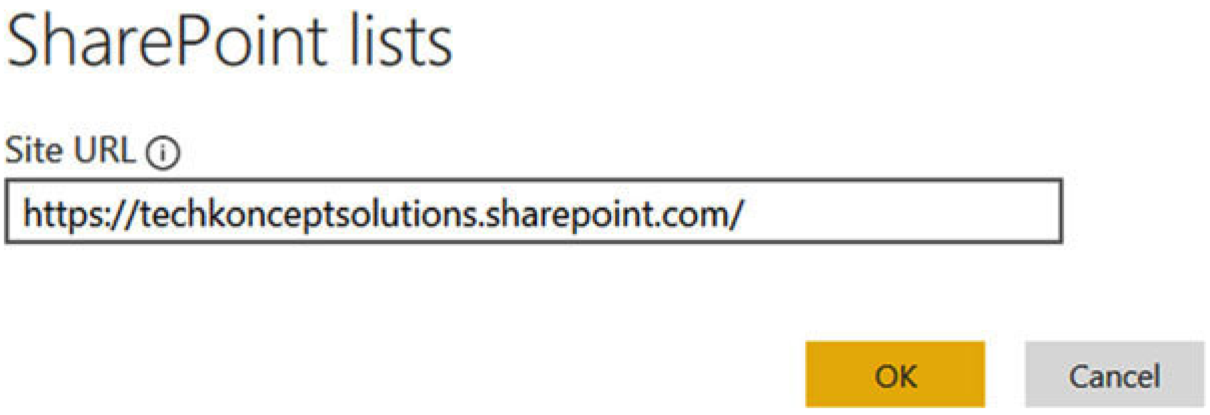
Select



Import data from a SharePoint site.

***Figure 2.46:** SharePoint List data connection*

In the site URL, specify the URL of your SharePoint site and select as shown in the following screenshot. The following URL is my organization's site.


SharePoint lists

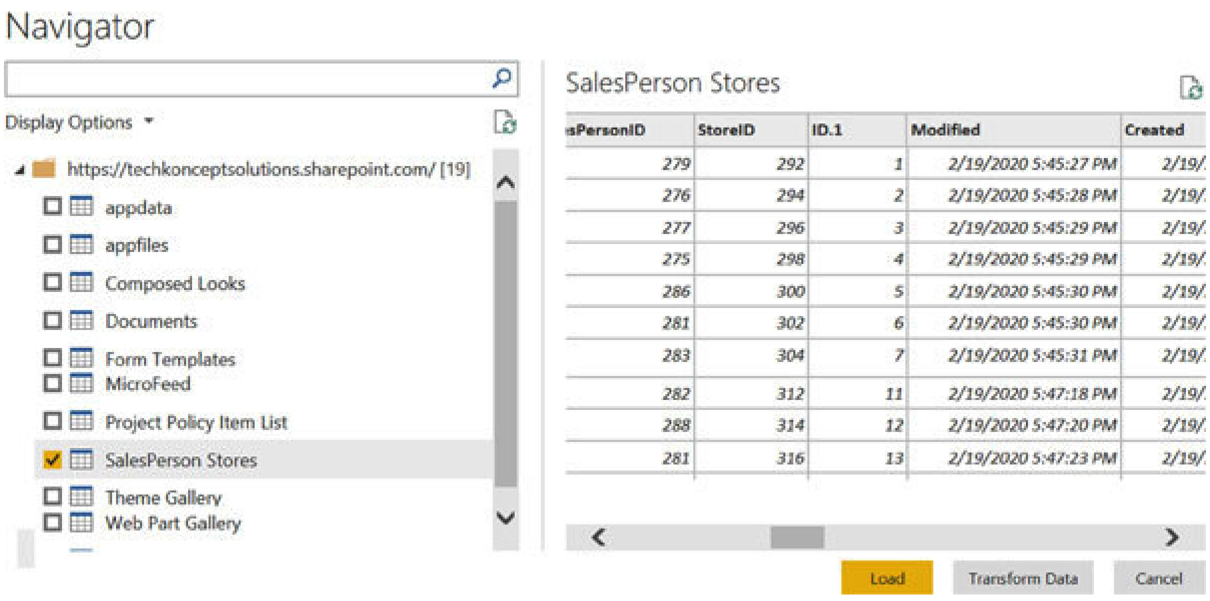
Site URL ⓘ

https://techkonceptsolutions.sharepoint.com/

***Figure 2.47:*** *Site URL*

If you get a connection error, follow the steps mentioned at the end of this section.

Once you are connected, select your list in the **Navigator** dialog box, as shown in the following screenshot:


Navigator

Display Options ▾

https://techkonceptsolutions.sharepoint.com/ [19]

- ☐ appdata
- ☐ appfiles
- ☐ Composed Looks
- ☐ Documents
- ☐ Form Templates
- ☐ MicroFeed
- ☐ Project Policy Item List
- ✓ SalesPerson Stores
- ☐ Theme Gallery
- ☐ Web Part Gallery

SalesPerson Stores

| sPersonID | StoreID | ID.1 | Modified | Created |
|---|---|---|---|---|
| 279 | 292 | 1 | 2/19/2020 5:45:27 PM | 2/19/ |
| 276 | 294 | 2 | 2/19/2020 5:45:28 PM | 2/19/ |
| 277 | 296 | 3 | 2/19/2020 5:45:29 PM | 2/19/ |
| 275 | 298 | 4 | 2/19/2020 5:45:29 PM | 2/19/ |
| 286 | 300 | 5 | 2/19/2020 5:45:30 PM | 2/19/ |
| 281 | 302 | 6 | 2/19/2020 5:45:30 PM | 2/19/ |
| 283 | 304 | 7 | 2/19/2020 5:45:31 PM | 2/19/ |
| 282 | 312 | 11 | 2/19/2020 5:47:18 PM | 2/19/ |
| 288 | 314 | 12 | 2/19/2020 5:47:20 PM | 2/19/ |
| 281 | 316 | 13 | 2/19/2020 5:47:23 PM | 2/19/ |

***Figure 2.48:** Selecting the SharePoint List*

Follow the steps for **Load** or **Transform Data** as needed.

In the model view, you will see another table loaded **SalesPerson**

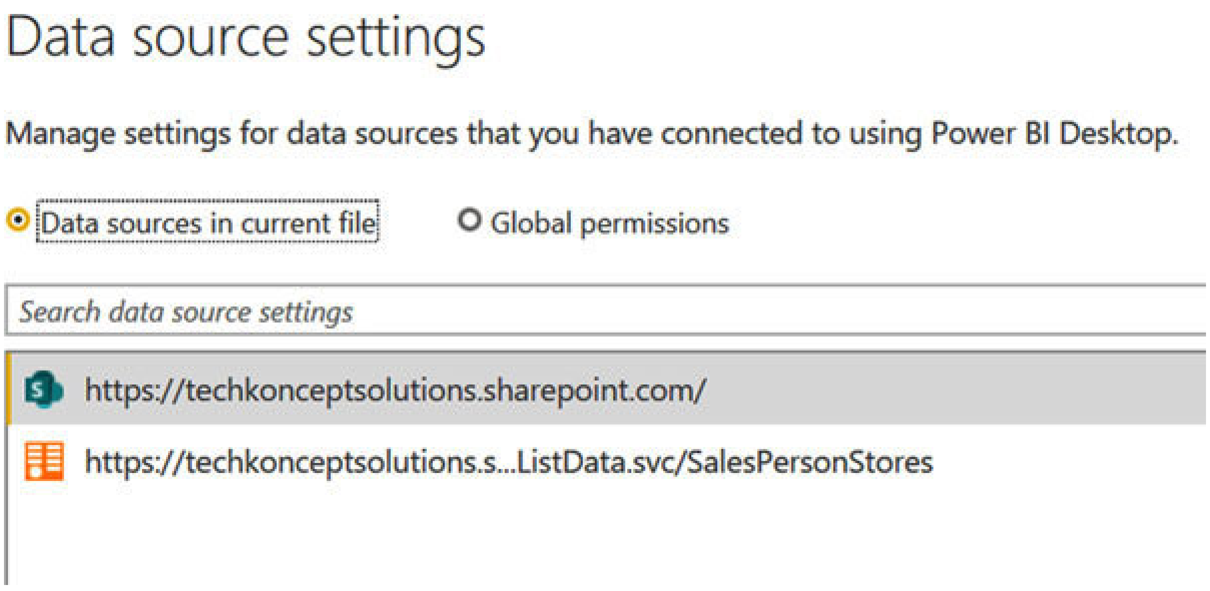
Power BI provides two options to connect to SharePoint – the ODataFeed and SharePoint connector – you can use any one method.

**Resolving the SharePoint list connection error:**

Complete the following steps, if you get a connection error after the SharePoint site URL is provided:

In the Power BI desktop, from the **Home** tab, navigate to **Transform** and from the dropdown list, select the **Data source**

In the **Data source** select your SharePoint site URL and click on **Edit Permissions** and provide your SharePoint connection credentials, as shown in the following screenshot:


Data source settings

Manage settings for data sources that you have connected to using Power BI Desktop.

(● Data sources in current file) ○ Global permissions

Search data source settings

https://techkonceptsolutions.sharepoint.com/

https://techkonceptsolutions.s...ListData.svc/SalesPersonStores

***Figure 2.49:*** *Providing credentials for SharePoint List*

This is similar to how we resolved the SharePoint connection error in the previous example.

## *Connecting and loading the from Azure SQL*

In this section, we will see how to connect and load the data from Azure SQL. This example shows how to connect to Azure SQL Server database. Refer to **Chapter2_AzureConnection.pbix** in your enclosed files.

For this example, we will use the Azure portal. A sample database **AdventureWorksLT** is created in Azure. To follow this example, create a free account with the Azure portal and create an SQL database.

*The steps to create the Azure portal and Azure SQL database is out of the scope for this book. If you are interested in more, follow the Microsoft Azure documentation that is available online.*

The steps to connect to the Azure SQL database are as follows:

Launch the Power BI desktop.

Save this file as

In the Power BI desktop, from the **Home** tab, select **Get** and click on

In the next **Get Data** dialog box, select Azure and choose **Azure SQL database** and select as shown in the following screenshot:

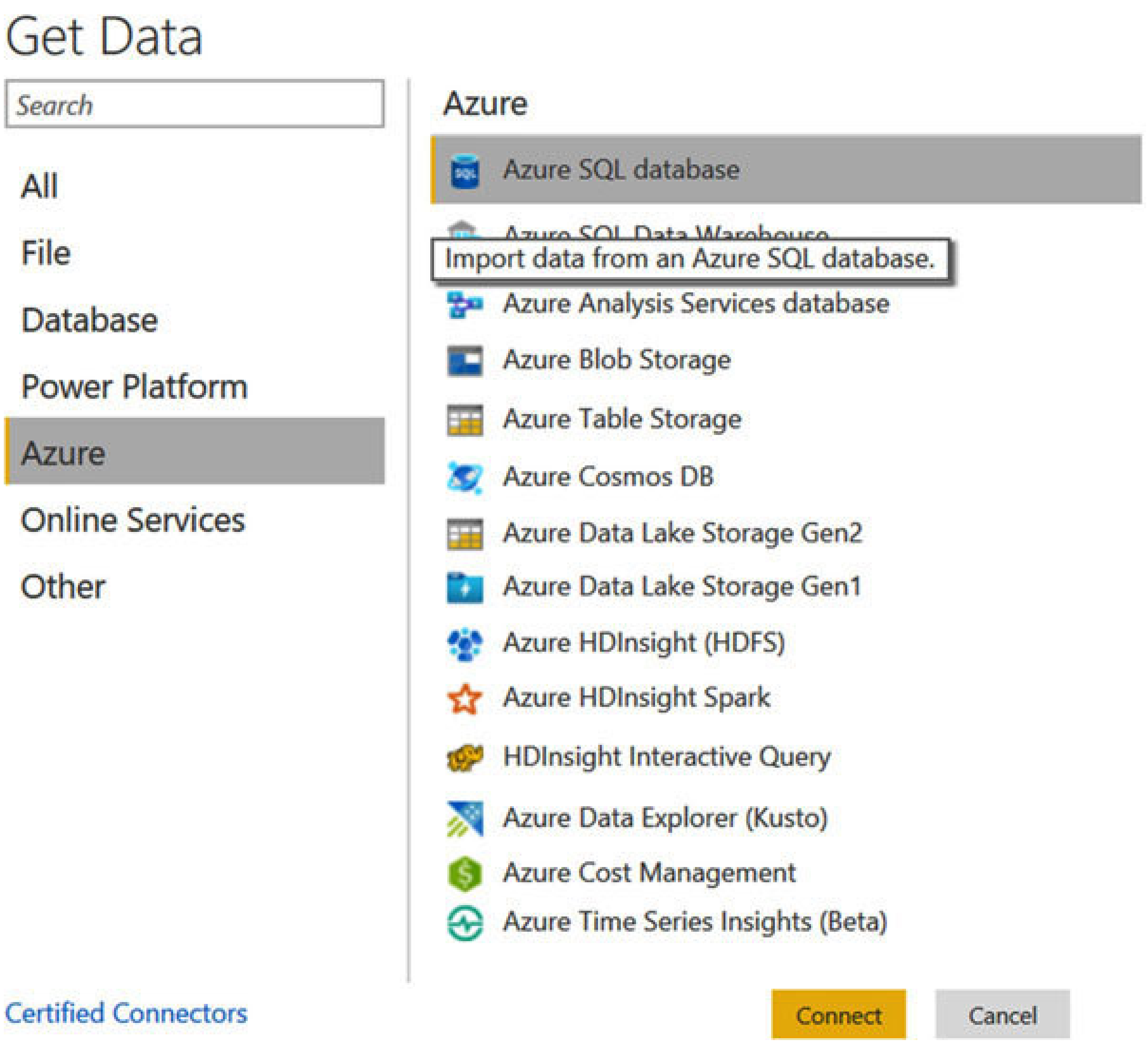


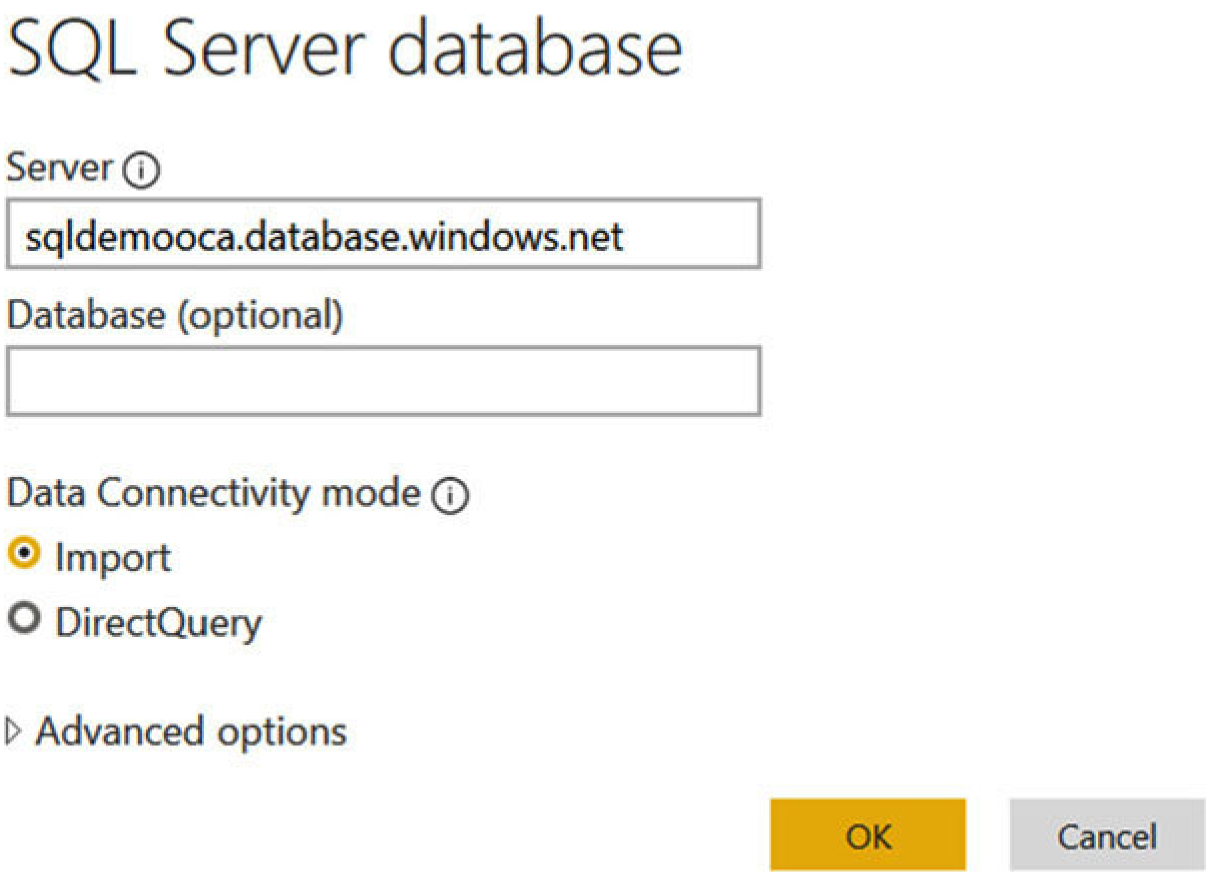
***Figure 2.50:*** *Connecting to Azure SQL Database*

In the SQL Server database dialog box, specify the **Server name** as mentioned in your Azure portal.

In the **Data Connectivity** you can select **Import** or Use the **Advanced** options if you wish to write an SQL query to extract the data.

Select

Provide the Azure SQL Server database credentials, as shown in the following dialog box, as shown in the following screenshot:


SQL Server database

Server: sqldemooca.database.windows.net

Database (optional):

Data Connectivity mode: [✓ Import] DirectQuery

***Figure 2.51:*** *Connecting to Azure SQL Server*

In the **Navigator** window, you can select one or more tables based on your requirement. In this example, four tables are selected.

Click on

This will create a data model, which can be used to create visualizations, as shown in the following screenshot:

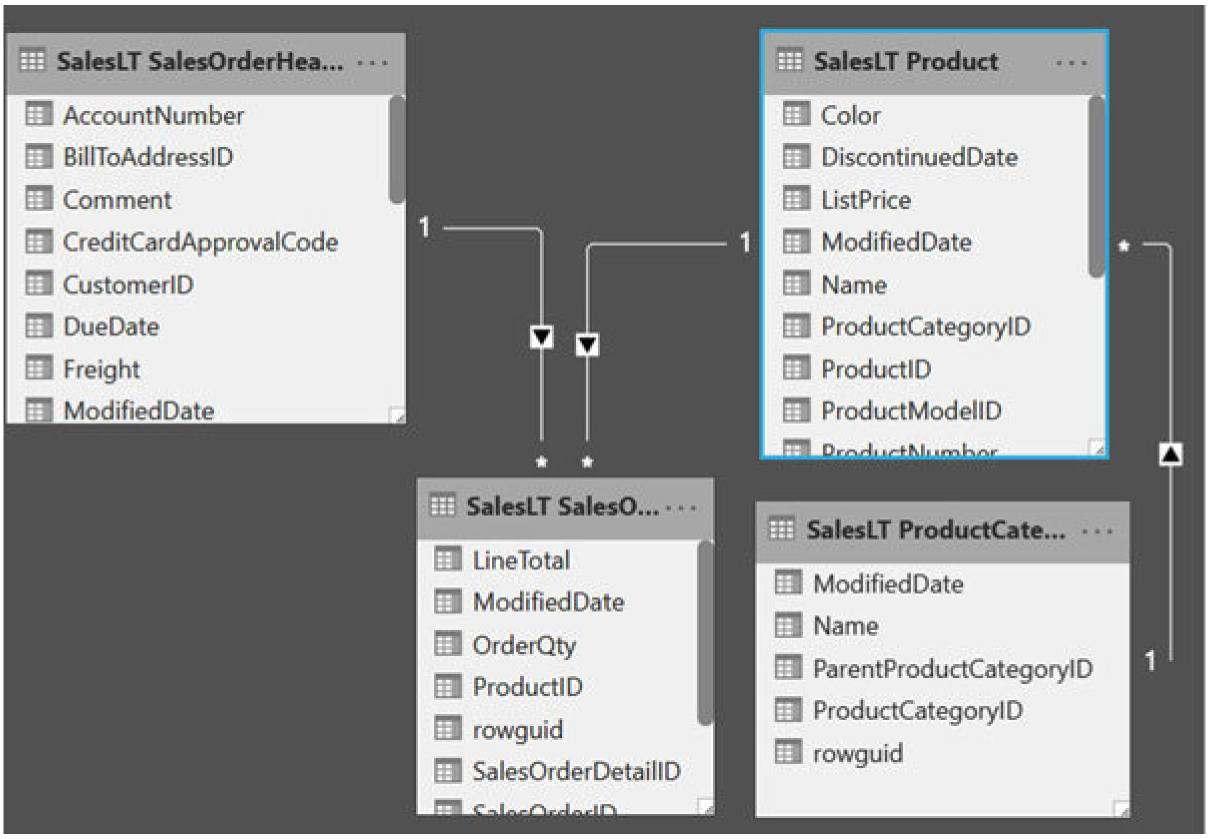


***Figure 2.52:*** *Data model depicting Azure SQL DB tables*

The database in Azure is no different from any other database. It is just available on the Azure cloud.

## ***More on Query Editor***

Since Query Editor is very useful and a powerful feature of Power BI, we will look at it in more detail. In the previous section, we have seen what Query Editor can do. In this section, we will dive deep into understanding the other functionalities offered by Query Editor.

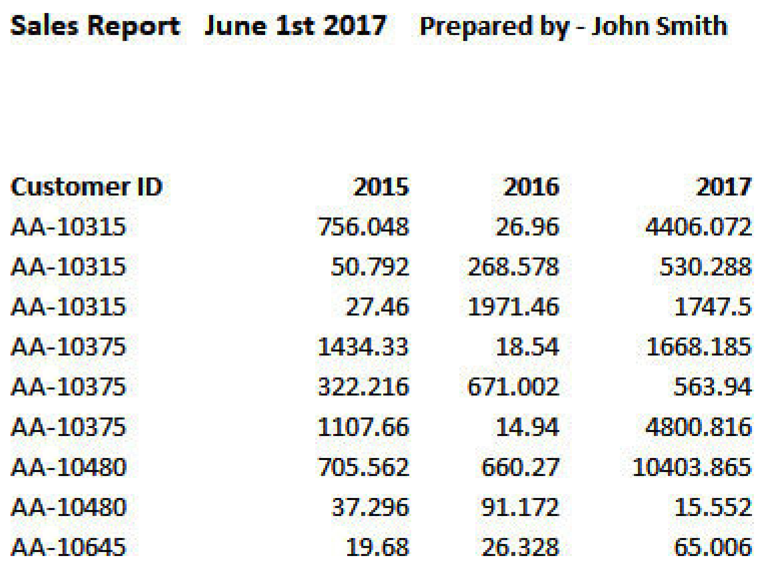
A new dataset CustomerSales_Report.xls is used to explain the various concepts of Query Editor. This XLS is available in the enclosed data files. This data is not part of our primary dataset used elsewhere in the book. Please review the data in the XLS before proceeding.

In real time, data is never perfect. We will receive data in all the different formats. Our objective is to take any data and transform it according to how it is expected by Power BI and the organization's reporting needs.

The **CustomerSales_Report.xls** file is a report prepared by a fictitious employee, John Smith on the Sales generated from the different customers between the years, 2015 and 2017. It is in the cross-tab format. In this example, we will see how to load this data using Query Editor and, in the process, we will learn about the different functionalities.

**Review of CustomerSales_Report.xls:**

The XLS contains two sheets – **CustomerSales_Report** and The structure of **CustomerSales_Report** sheet is shown in the following screenshot:



Sales Report June 1st 2017 Prepared by - John Smith

| Customer ID | 2015 | 2016 | 2017 |
|---|---|---|---|
| AA-10315 | 756.048 | 26.96 | 4406.072 |
| AA-10315 | 50.792 | 268.578 | 530.288 |
| AA-10315 | 27.46 | 1971.46 | 1747.5 |
| AA-10375 | 1434.33 | 18.54 | 1668.185 |
| AA-10375 | 322.216 | 671.002 | 563.94 |
| AA-10375 | 1107.66 | 14.94 | 4800.816 |
| AA-10480 | 705.562 | 660.27 | 10403.865 |
| AA-10480 | 37.296 | 91.172 | 15.552 |
| AA-10645 | 19.68 | 26.328 | 65.006 |

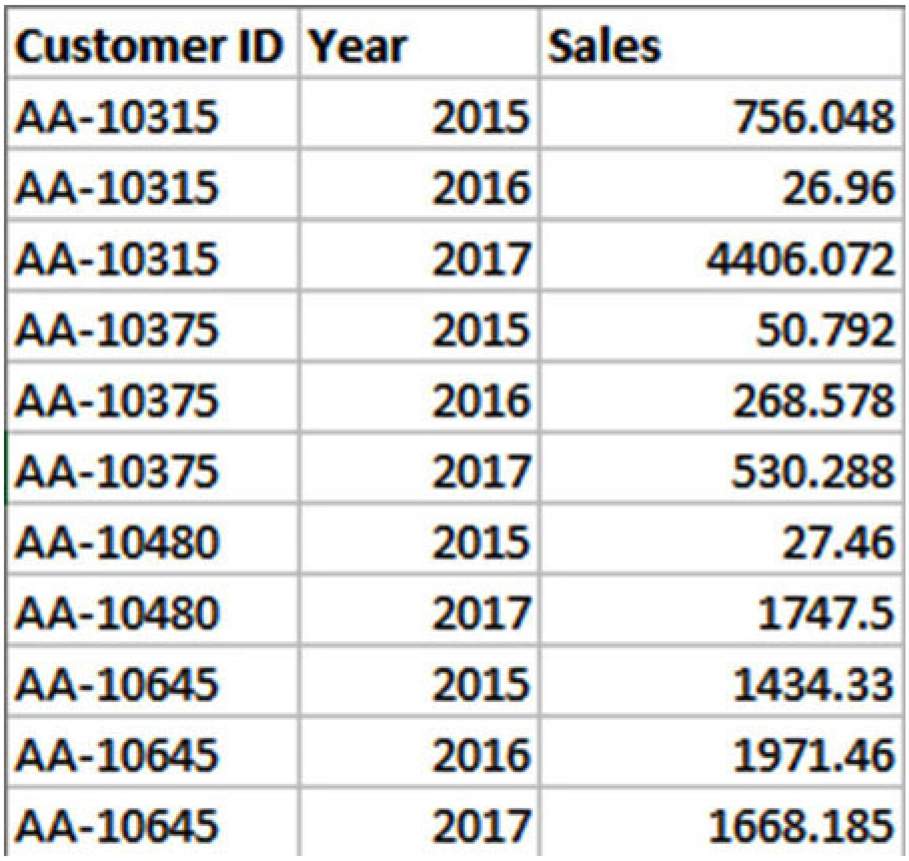
***Figure 2.53:*** *Customer Sales Report*

In the preceding figure, we can see that it has a report title, the customer IDs are listed as a column, and the Sales for each of the year are listed under the specific Year columns, that is, 2015, 2016, and 2017.

**Problem with the preceding data:**

If the preceding data is loaded without any transformation, it will be difficult to derive useful information from it. Aggregating this data will be an issue, and getting a year-by-year trend will be problematic. Also, if we have more years in the future, it will make the data load more cumbersome as each of the year will be loaded separately.

The expected structure of the data is provided in the **ExpectedFormat** sheet. We should apply the transformations and convert the preceding **CustomerSales_report** in the correct format. Any database will store the data in this way and it should be loaded similarly in Power BI too, as shown in the following screenshot:



| Customer ID | Year | Sales |
|---|---|---|
| AA-10315 | 2015 | 756.048 |
| AA-10315 | 2016 | 26.96 |
| AA-10315 | 2017 | 4406.072 |
| AA-10375 | 2015 | 50.792 |
| AA-10375 | 2016 | 268.578 |
| AA-10375 | 2017 | 530.288 |
| AA-10480 | 2015 | 27.46 |
| AA-10480 | 2017 | 1747.5 |
| AA-10645 | 2015 | 1434.33 |
| AA-10645 | 2016 | 1971.46 |
| AA-10645 | 2017 | 1668.185 |

***Figure 2.54:*** *Expected Format of the data*

As we can see from the preceding figure, in this data sheet, the customer ID, year, and sales are listed as separate columns. Now, it is easier to see which year had the maximum sales. We can also find out the year-by-year sales trend.

In the following exercise, our objective is to use Query Editor to transform **CustomerSales_Report.xls** into the correct format.

The steps to load the CustomerSales_Report.xls format are as follows:

(This data will not be used in our primary data model and will only be used for demonstration purposes. Refer to

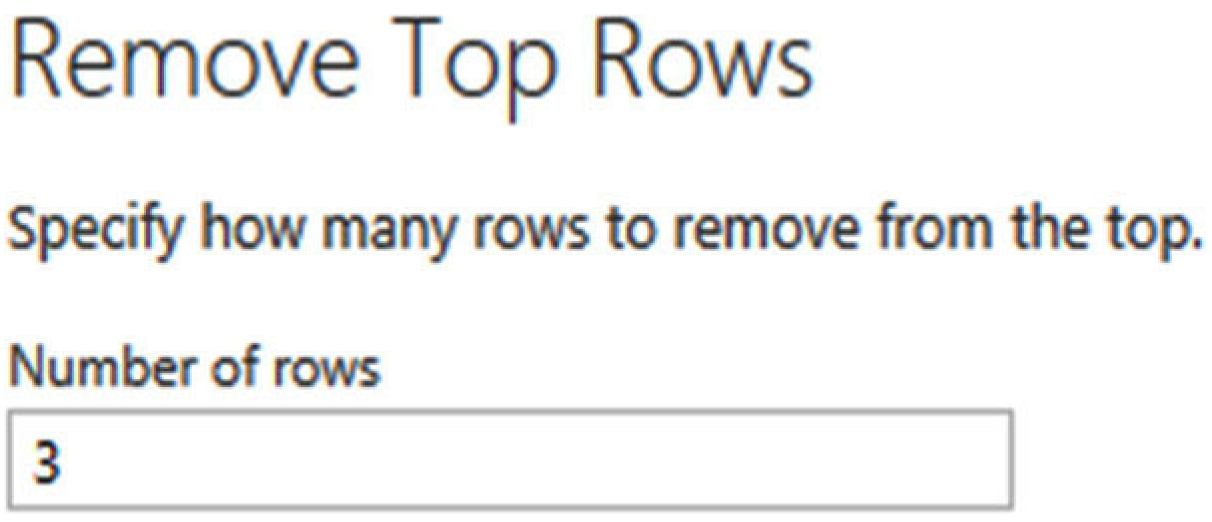
Launch the Power BI desktop, from the **Home** tab, select **Transform** We will be navigated to Query Editor.

From the **Home** ribbon, select **New Source** and then Browse to **CustomerSales_Report.xls** and open it.

In the **Navigator** dialog box, select **CustomerSales_Report.xls** and select

In the loaded data, observe that it has loaded the report title as the column headers and the three top rows are displayed as nulls. We will fix these blank rows and headers.

From the **Home** ribbon, select **Remove** and from the drop-down, select In the **Remove Top Rows** dialog box, specify the number of rows that needs to be removed – here it is three – and select as shown in the following screenshot:


Remove Top Rows

Specify how many rows to remove from the top.

Number of rows

3

***Figure 2.55:*** *QE_3 Remove Top rows*

This will remove the three rows with nulls.

Notice, the first row contains the actual column headers of the Excel. To get the correct column header, navigate to the **Transform** ribbon and select **Use First Row as** This will remove the incorrect column headers and use the first row as headers which contains the correct names.

Look at **APPLIED STEPS** on the right. It records the transformation steps that we take on the loaded data. If at any time, we need to revert our changes, simply delete the step, as shown in the following screenshot:

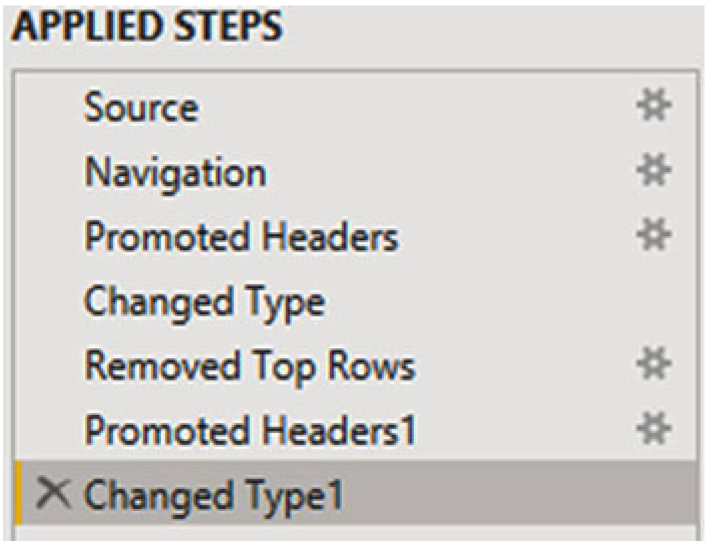


***Figure 2.56:*** *Transformation Steps*

The preceding steps have loaded the data the way it is in the first XLS sheet, that is, Now we will transform it, so that it looks like the **ExpectedFormat** sheet.

The steps to transform CustomerSales_Report.xls are as follows:

In Query Editor, press *Shift* and select the columns 2015, 2016, and 2017. From the **Transform** ribbon, select **Unpivot Columns** and then select **Unpivot Only Selected** as shown in the following screenshot:

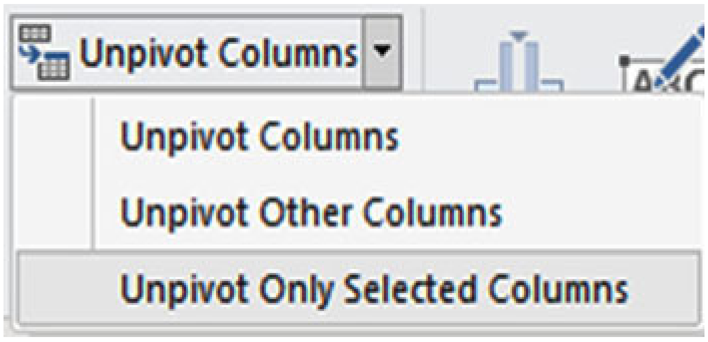
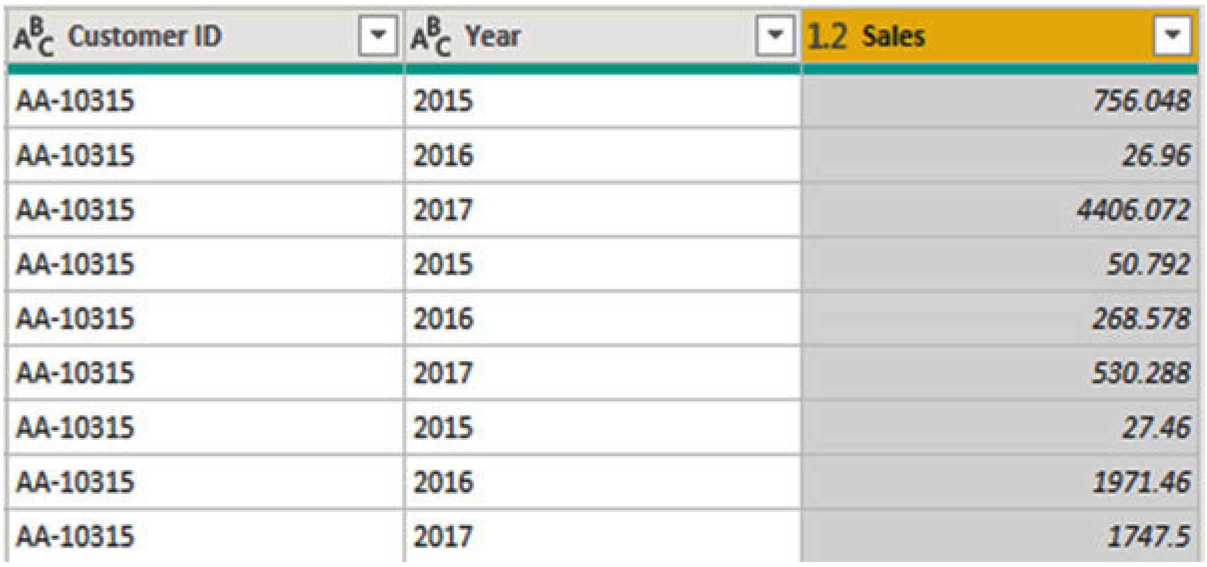


*Figure 2.57: Unpivoting Year columns*

This will unpivot the 2015, 2016, and 2017 columns. In place of those, we will see two new columns – one for **Year** and one for **Sales** in each of the It will show some generic column titles, such as attribute and value. We can right-click on these and rename them.

The data will now be transformed in the expected format. After the transformations, the correct format of the data is achieved, as shown in the following screenshot:



| Customer ID | Year | 1.2 Sales |
|---|---|---|
| AA-10315 | 2015 | 756.048 |
| AA-10315 | 2016 | 26.96 |
| AA-10315 | 2017 | 4406.072 |
| AA-10315 | 2015 | 50.792 |
| AA-10315 | 2016 | 268.578 |
| AA-10315 | 2017 | 530.288 |
| AA-10315 | 2015 | 27.46 |
| AA-10315 | 2016 | 1971.46 |
| AA-10315 | 2017 | 1747.5 |

*Figure 2.58: Expected format of the data*

The other transformation, such as changing the data types, can be applied as discussed in the previous sections.

From the **File** menu, select **Close & Apply** to load this data in Power BI.

We can view this table loaded in the Data view or Model view.

**Split columns by delimiter:**

If in the loaded data, we have a column which is a concatenation of two values, we can split the column into two, based on certain conditions. In our data, the customer ID column has a text and a number such as AA-10315. We can split such data if it is needed for reporting purposes.

The steps are as follows:

In Query Editor, select the query we have loaded. Select the **Customer ID** column.

From select **Split** There are different options to split a column; for this example, select **By** as shown in the following screenshot:

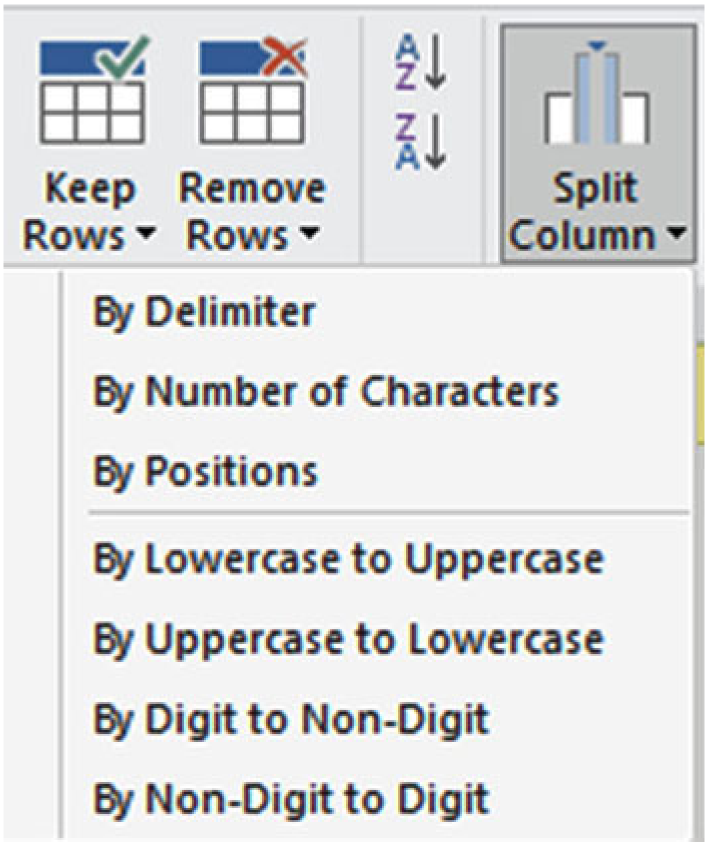
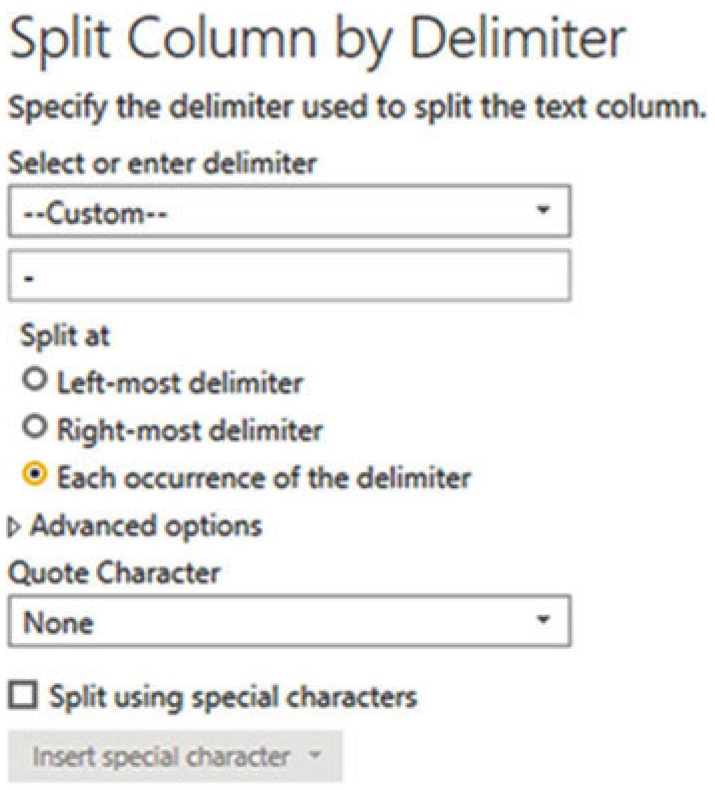


***Figure 2.59:*** *Split column*

In the **Split Column by Delimiter** dialog box, specify how you would like to split the column. Explore the different options under Select or enter delimiter. We can select the default custom because it uses the delimiter that we need, because the **Customer ID** column uses – (hyphen) between the text **AA** and the number, as shown in the following screenshot:



Split Column by Delimiter

Specify the delimiter used to split the text column.

Select or enter delimiter

--Custom--

-

Split at

○ Left-most delimiter

○ Right-most delimiter

● Each occurrence of the delimiter

▷ Advanced options

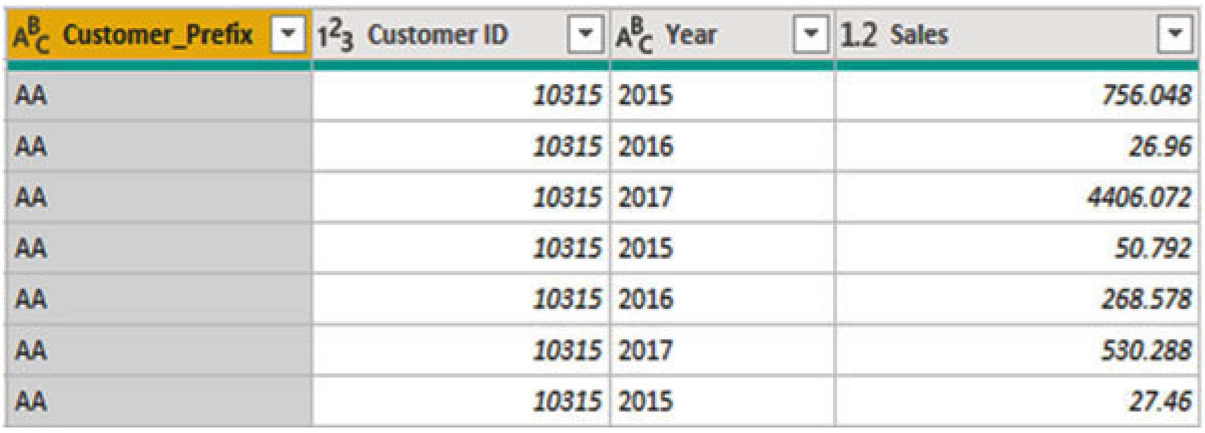
Quote Character

None

☐ Split using special characters

***Figure 2.60:*** *Specifying the delimiter*

This will split the **Customer ID** column into two columns. We can rename these columns according to our preference, as shown in the following screenshot:



| Customer_Prefix | Customer ID | Year | Sales |
|---|---|---|---|
| AA | 10315 | 2015 | 756.048 |
| AA | 10315 | 2016 | 26.96 |
| AA | 10315 | 2017 | 4406.072 |
| AA | 10315 | 2015 | 50.792 |
| AA | 10315 | 2016 | 268.578 |
| AA | 10315 | 2017 | 530.288 |
| AA | 10315 | 2015 | 27.46 |

***Figure 2.61:*** *Columns Split using a delimiter*

## Group By in Query Editor:

Group by is an important functionality in Query Editor. It can be used to check the totals in the table to make sure that the data is correct. It comes in handy when a complex dataset is used with multiple columns or when the tables are merged.

We will use the **CustomerSales_Report** data that we loaded earlier and perform grouping on it. Before jumping to it, let's understand the two important concepts of reusing the tables/query, that is, *Reference vs*

## Reference vs. duplicate:

Right-click on the **CustomerSales_Report** query loaded in Query Editor. We will have two options – **Duplicate** and as shown in the following screenshot:

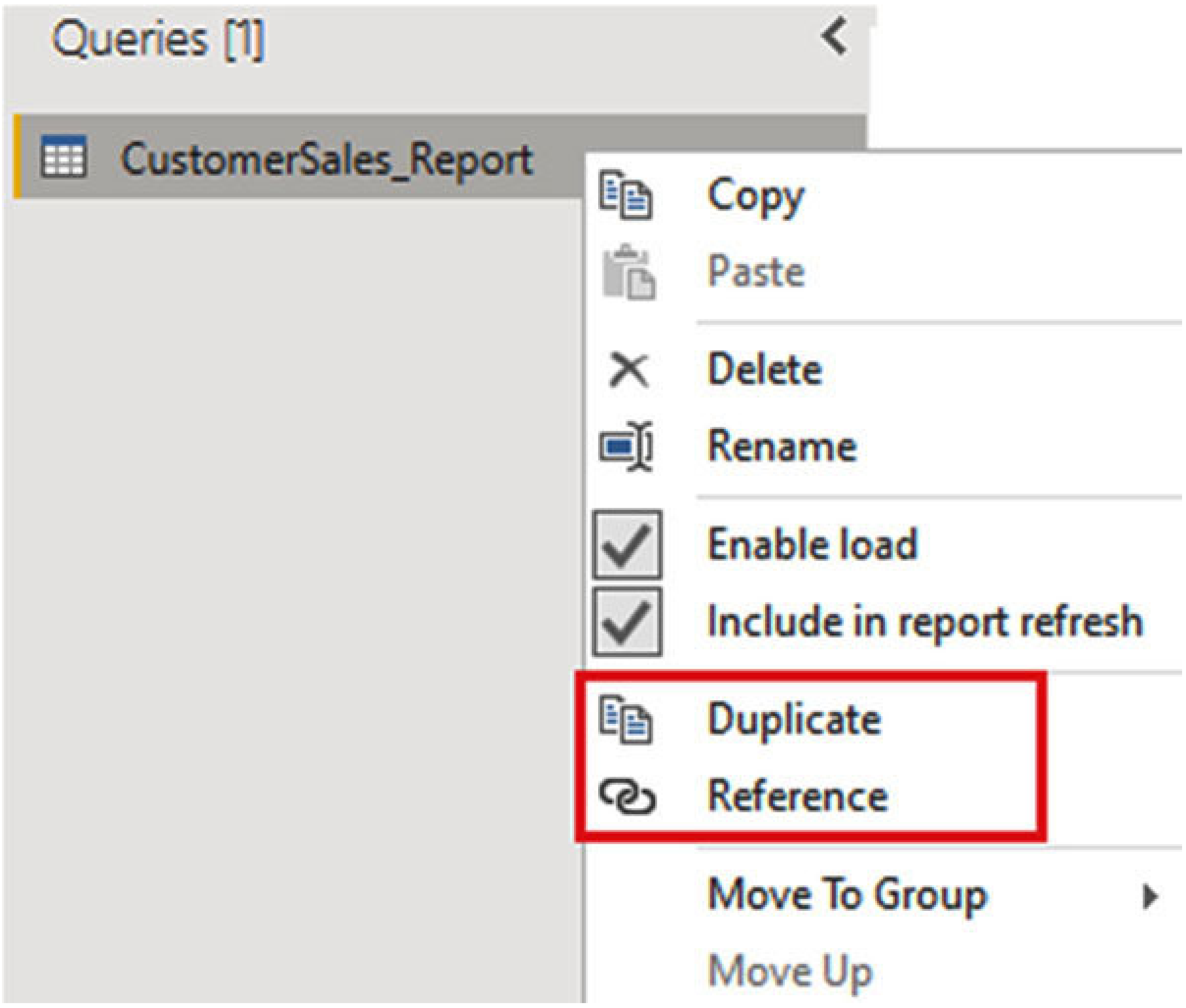


***Figure 2.62:*** *Duplicate and Reference options*

When the **Duplicate** option is selected, Query Editor makes an exact copy of the query/table with all the 'applied steps' to the query. This new query is independent of the original query. If we perform any changes to the original query, the new query will not be impacted.

The reference option simply references the original query/table. It will have only one applied step, that is, It sources the data from the original query. If we make changes to the original query, then this referenced query will also change.

To explore, select **Duplicate** and reference and observe the **APPLIED**

**Applying Group By to CustomerSales_Report:**

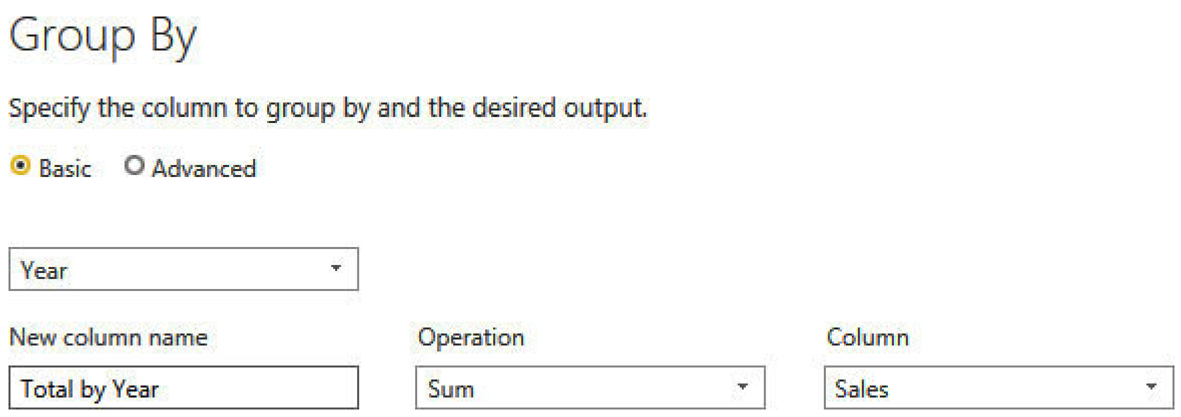
In this section, we will see how to perform **Group By** to calculate the **Total Sales by**

The steps to perform Group By are as follows:

In Query Editor, right-click on the **CustomerSales_Report** query and select

Rename the new query as **Total by**

With the new query selected, navigate to the **Transform** ribbon, and select the **Group By** option. In the **Group By** dialog box, change the defaults to calculate the **Totals by** as shown in the following screenshot:



Group By

Specify the column to group by and the desired output.

[✓ Basic] Advanced

Year

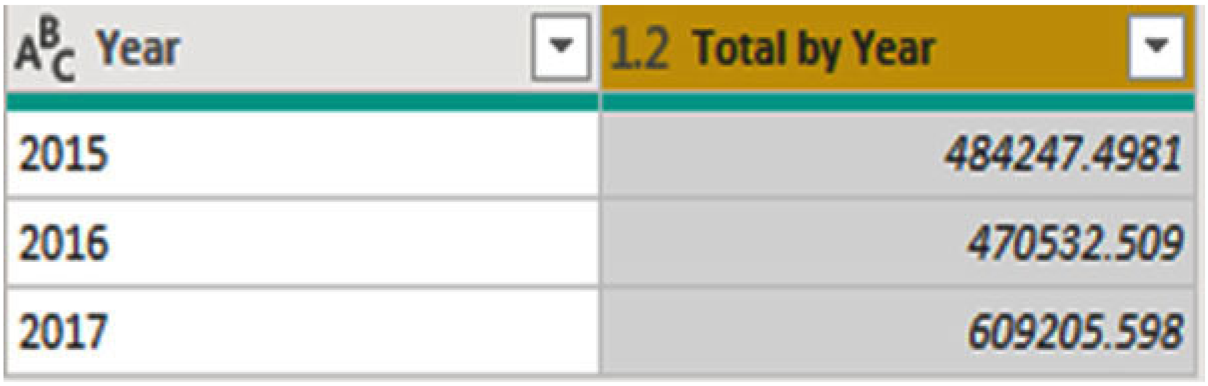
New column name: Total by Year

Operation: Sum

Column: Sales

***Figure 2.63:*** *Group by Operation*

This will give us a table for **Total by** as shown in the following screenshot:



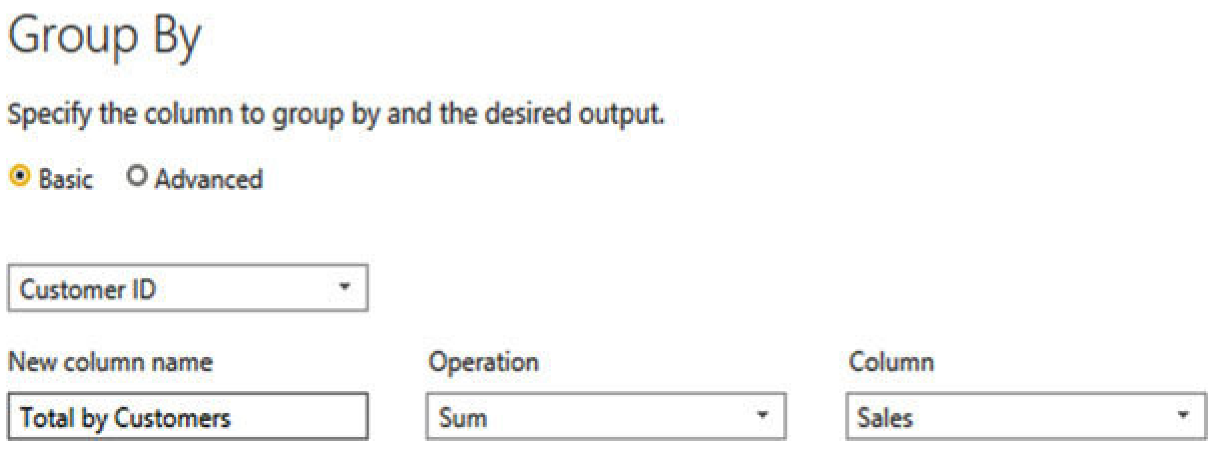
| Year | Total by Year |
|---|---|
| 2015 | 484247.4981 |
| 2016 | 470532.509 |
| 2017 | 609205.598 |

***Figure 2.64:*** *Group By calculates Total by Year*

**Group By to calculate Total by Customer:**

Now, we will calculate **Total by** Complete the same steps as earlier. Right-click on **CustomerSales_Report** and select Rename the new query as **Total By**

In the **Group By** dialog box, select **Customer Operation** as and **Column** as **Sales** as shown in the following screenshot:


Group By

Specify the column to group by and the desired output.

[✓ Basic] Advanced

Customer ID

| New column name | Operation | Column |
|---|---|---|
| Total by Customers | Sum | Sales |

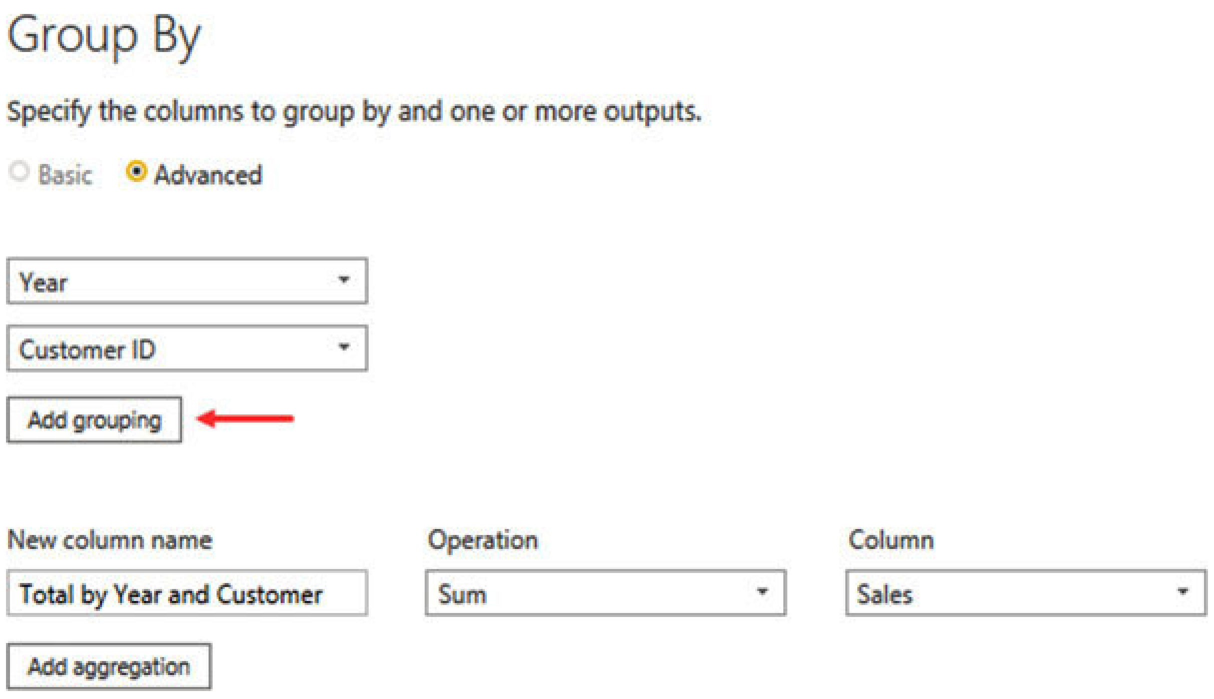
***Figure 2.65:*** *Group By to calculate Total by Customers*

**Advanced Group By to calculate Totals by Year and Customer:**

By using the advanced option in **Group** we can perform grouping on multiple dimensions, that is, **Year** and

Complete the same steps that were given earlier. Right-click on **CustomerSales_Report** and select Rename the new query as **Total By Year and**

In the **Group By** dialog box, select the **Advanced** option. This option will let us pick multiple dimensions and aggregations. Click on **Add grouping** to add more dimensions, as shown in the following screenshot:



Group By

Specify the columns to group by and one or more outputs.

Basic [✓ Advanced]

Year

Customer ID

Add grouping

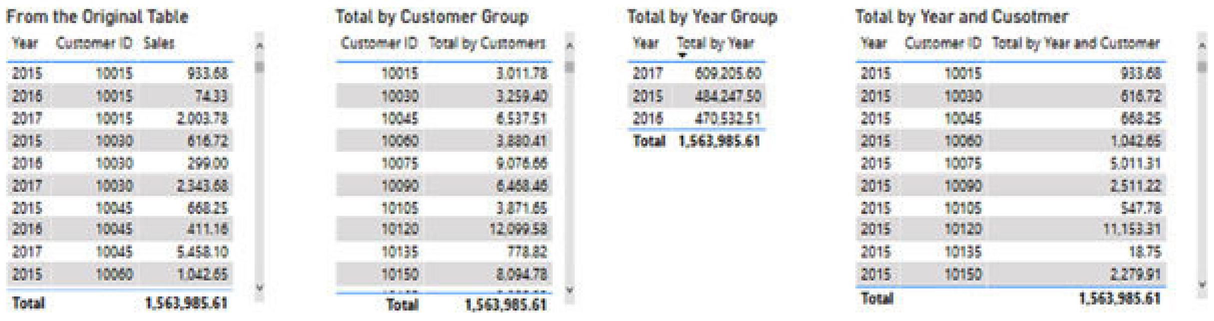
| New column name | Operation | Column |
|---|---|---|
| Total by Year and Customer | Sum | Sales |

Add aggregation

*Figure 2.66: Advanced Grouping*

In the preceding exercises, we have loaded one table and created three groupings on that data.

If we take the sales total of all the four, it should match. This shows that our loaded data is correct. We got the same total, that is, 1,563,985.61 in all the queries, as shown in the following screenshot:



From the Original Table

| Year | Customer ID | Sales |
|---|---|---|
| 2015 | 10015 | 933.68 |
| 2016 | 10015 | 74.33 |
| 2017 | 10015 | 2,003.78 |
| 2015 | 10030 | 616.72 |
| 2016 | 10030 | 299.00 |
| 2017 | 10030 | 2,343.68 |
| 2015 | 10045 | 668.25 |
| 2016 | 10045 | 411.16 |
| 2017 | 10045 | 5,458.10 |
| 2015 | 10060 | 1,042.65 |
| Total | | 1,563,985.61 |

Total by Customer Group

| Customer ID | Total by Customers |
|---|---|
| 10015 | 3,011.78 |
| 10030 | 3,259.40 |
| 10045 | 6,537.51 |
| 10060 | 3,880.41 |
| 10075 | 9,076.66 |
| 10090 | 6,468.46 |
| 10105 | 3,871.65 |
| 10120 | 12,099.58 |
| 10135 | 778.82 |
| 10150 | 8,094.78 |
| Total | 1,563,985.61 |

Total by Year Group

| Year | Total by Year |
|---|---|
| 2017 | 609,205.60 |
| 2015 | 484,247.50 |
| 2016 | 470,532.51 |
| Total | 1,563,985.61 |

Total by Year and Cusotmer

| Year | Customer ID | Total by Year and Customer |
|---|---|---|
| 2015 | 10015 | 933.68 |
| 2015 | 10030 | 616.72 |
| 2015 | 10045 | 668.25 |
| 2015 | 10060 | 1,042.65 |
| 2015 | 10075 | 5,011.31 |
| 2015 | 10090 | 2,511.22 |
| 2015 | 10105 | 547.78 |
| 2015 | 10120 | 11,153.31 |
| 2015 | 10135 | 18.75 |
| 2015 | 10150 | 2,279.91 |
| Total | | 1,563,985.61 |

*Figure 2.67: Totals from all the queries to compare results*

Group by is an important way to check if the data is loaded correctly. It is very useful when loading multiple tables or creating joins between the tables.

**Creating a Column in Query Editor:**

In Query Editor, from the **Add Column** ribbon, we can create the different types of columns, such as the custom column, conditional column, etc. In this section, we will create a conditional column based on sales, to display when the Sales is Low, High, or Medium.

The column will be based on the following conditions:

*If Sales is less than 1500, then display Low.*

*If Sales is greater than 1500, then display Medium.*

*If Sales is greater than 2000, then display Good.*

**The steps to create a conditional column are as follows:**

(In this exercise, we will also see how to create a Duplicate of the query.)

In Power Query Editor, right-click on the query **CustomerSales_Report** and select Rename the new query as

Observe the **APPLIED** it will contain all the transformation steps of the original query.

Navigate to the **Add Column** ribbon and select **Conditional**

In the **Add Conditional Column** dialog box, specify the following condition:

**New column name:** This will add a new named column in the query.

**Column Name:** Select the column on which the condition will be applied.

**Operator:** Select the comparison operator, such as *equal to* or *is less than* etc.

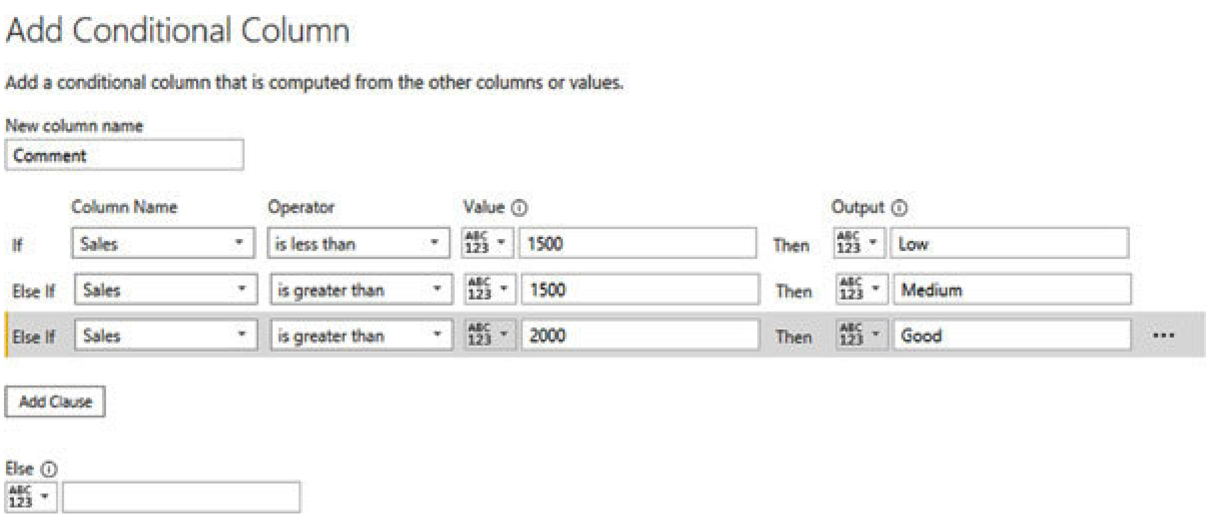
**Value:** It is used to input the value that we want to compare against. It can be a text or a numeric value.

**Output:** Specify the output to be displayed if the condition is true.

**Add Clause:** It is used to specify more conditions.

An else condition can be specified to display the values if the condition evaluates as false.

Take a look at the following screenshot for the specified conditions



Add Conditional Column

Add a conditional column that is computed from the other columns or values.

New column name: Comment

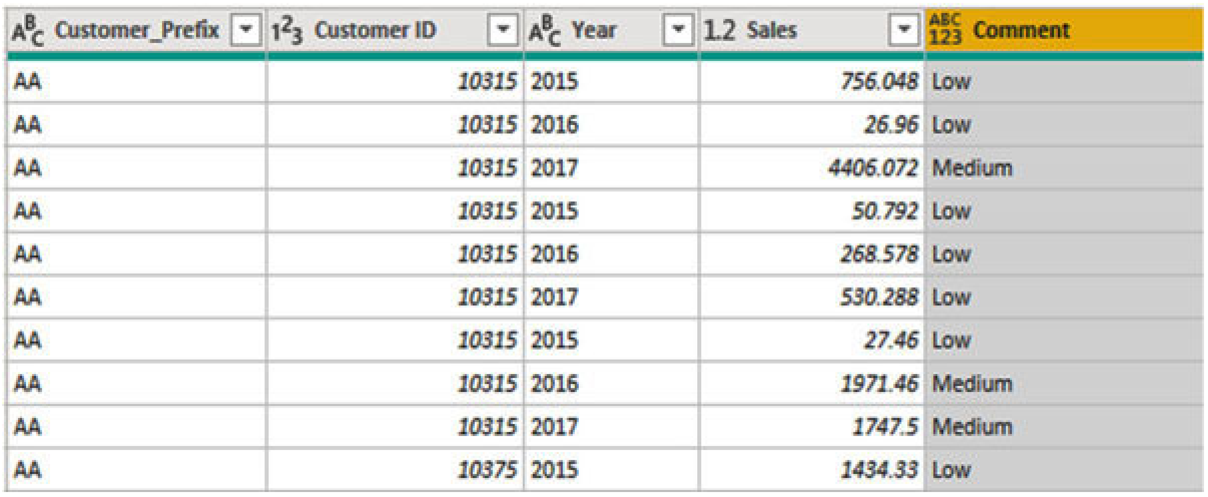
| | Column Name | Operator | Value | | Output |
|---|---|---|---|---|---|
| If | Sales | is less than | 1500 | Then | Low |
| Else If | Sales | is greater than | 1500 | Then | Medium |
| Else If | Sales | is greater than | 2000 | Then | Good |

Add Clause

Else

*Figure 2.68: Adding a Conditional Column*

This will create a column **Comment** in our query with the values and **Good** based on the **Sales** amount, as shown in the following screenshot:



| Customer_Prefix | Customer ID | Year | Sales | Comment |
|---|---|---|---|---|
| AA | 10315 | 2015 | 756.048 | Low |
| AA | 10315 | 2016 | 26.96 | Low |
| AA | 10315 | 2017 | 4406.072 | Medium |
| AA | 10315 | 2015 | 50.792 | Low |
| AA | 10315 | 2016 | 268.578 | Low |
| AA | 10315 | 2017 | 530.288 | Low |
| AA | 10315 | 2015 | 27.46 | Low |
| AA | 10315 | 2016 | 1971.46 | Medium |
| AA | 10315 | 2017 | 1747.5 | Medium |
| AA | 10375 | 2015 | 1434.33 | Low |

*Figure 2.69: Conditional column Comment displayed in the query*

Query Editor has many other useful functionalities such as merging or appending two queries. We will cover them in the following chapter.

## ***Conclusion***

Power BI can connect and load the data in different ways. As we saw in this chapter, data is loaded from the Relational database, Excel, CSV files, SharePoint, Azure, and Web. We can also create tables within Power BI. The advantage of Power BI is that data can be loaded from the diverse sources but it can still be combined together to provide a single version of truth.

This chapter also pays more attention to Query Editor; a good knowledge of Query Editor is essential to transform the data in Power BI.

In this chapter, we learned how to load and transform data; in the next chapter, we will learn how to create relationships between these tables to create a data model.

## ***Questions***

Which option is selected to load the data in the Power BI desktop?

What is the difference between Get Data and Transform data?

What is APPLIED STEPS in Query Editor?

What is the default Data connectivity option when connected to a database such as SQL Server?

What are the two ways to connect to SharePoint?

What is the difference between Duplicate and Reference in Query Editor?

Which method is used to see if the loaded data is correct?

## ***Answers***

Get Data from the Home ribbon.

Get Data option is used to connect to the data source and load data. The transform data option navigates you to Query Editor and helps in transforming the data. It can also be used to load the data.

APPLIED STEPS shows the steps performed in transforming the data in the table.

Import.

The two ways to connect to SharePoint are as follows:

OData Feed

SharePoint Lists

The Duplicate and Reference options refer to the previously loaded query. The Duplicate option creates an exact copy of the query with all the applied steps. The new query is disconnected from the original query, and any new changes made to the original query will not be reflected in the new query.

The Reference option references or links to the original query. The new query contains only one applied step, that is, source. Any new changes made to the original query will be displayed in the new query.

Group By.

CHAPTER 3

# Optimize Your Data Model

## ***Introduction***

In the previous chapter, we learned how to extract, load, and transform the data from the different sources, and create a data model.

In this chapter, we will learn how to create a data model and use joins. It is important that while creating the data model best practices followed, so that the data model is simple, maintainable, and reusable. It should also be easy for the other developers and power users to follow.

## ***Structure***

In this chapter, we will discuss the following topics:

Concepts of Data modeling

Creating different types of relationships in Power BI

Different ways to combine the tables

## *Objectives*

The objective of this chapter is to learn what is a data model and the best practices of data modeling. Relationships are an important part of a data model; we will learn all about relationships in Power BI and create relationships between loaded tables. We will also learn how to combine the tables using the Power BI merge and append functionalities.

## ***Introduction to data modeling***

Data modeling is the most important aspect of a Business Intelligence implementation. Without a proper data model, the business will not get a true picture of the data. The organizations spend considerable resources in creating a useful model.

Data model or data modeling is a process of organizing the data elements and defining how they relate with each other. Data model is created, taking into consideration the business requirements. The questions to be asked before creating a model should be – what data or KPI's are most important to the business and how do we arrive at this data? Given a set of data, can we answer all the business queries effectively? A good data model adheres to the business rules and assist in quick decision making.

For example, from the tables and if the business wants to know, in different years, the total sales by the customer region, what type of relationship/s should exist between these tables to answer this question? A good data model will help us in answering such queries. The data model feeds structured data to the visualizations, and thus helps in efficient reporting.

## ***Best practices of data modeling***

A data model is a basis for visualizations. It is a must that an optimized data model is created as it will facilitate easy creation of reports and dashboards.

The following best practices of data modeling will ensure that the data model is robust, reusable, and performs in an optimized way.

Create a star schema. A star schema is good for reporting and analysis due to a smaller number of joins. The data model approach should be to create a schema closer to a star schema.

Load only the data required. If the source data contains ten years of data but the users are interested in seeing only five years of data, use filters and load only five years of data.

Reduce the number of tables and relationships. Load only the tables required to create visualizations. If some tables are needed to create relationships with the other tables, we can use them in the data model but hide them from the report view.

Avoid using time portion in the date. When dates are present in the table, avoid using the time portion in the date. Convert the **Date** column data type to only date.

One fact table approach. Sometimes it may not be possible, but an attempt should be made to have only one fact table in the model.

Hide the tables and columns not required in the visualizations. The report view should show a lesser number of tables and columns.

Remove the fields which are present in more than one table. In the Power BI desktop, we use the search function in fields. It will show the occurrence of a field in different tables.

The tables with two columns in the format of an ID and description should be hidden. These tables can be loaded to the main table.

Combine the tables wherever necessary to reduce the number of tables in the data model.

While designing a data model, always follow the best practices and design as per the visualization needs.

## ***A word about relationships***

Since relationship is an important activity performed while creating a data model, let’s learn about it. The relationship between two tables work by matching the data in the key columns. In a typical environment, two tables are connected based on the Primary key on one table and a foreign key on another table. Usually the columns/fields having the same name between the tables are assumed to be related, though this may or may not be true.

For example, the two tables can have an ID field to uniquely identify the row in the table, but these two tables may not be connected/related to each only because of the common field name. If such scenario occurs, the relationship should be reviewed because a proper relationship will ensure accuracy of data.

There are three kinds of relationships or cardinality between the tables, which are as follows:

**Many to one (*:1):** This is the default relationship and means that many rows in one table relates to one row in another table.

**One to One (1:1):** In this relationship, one row in the first table is related to one row in the second table.

**Many to Many:** In this relationship, many rows in the first table relates to many rows in the second table. For example, **Orders**

and **Customers** tables.

In the Power BI desktop, creating relationship is an easy task. In many cases, Power BI auto detects the relationship between the column names in the two tables. If Power BI cannot detect a relationship, we can create one by using the **Home** ribbon option, **Manage Relationships,** as shown in the following screenshot:

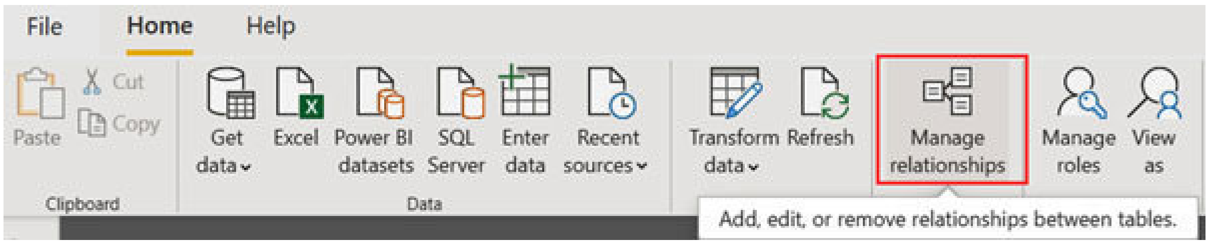


***Figure 3.1:*** *Manage relationships option*

Using the Manage relationships option, you can edit or review the existing relationships. We use these concepts when we define the relationships between our loaded tables.

## *Review the loaded tables*

In *Chapter 2, Connect and* we have loaded various tables. The loaded tables are as shown in the following screenshot:

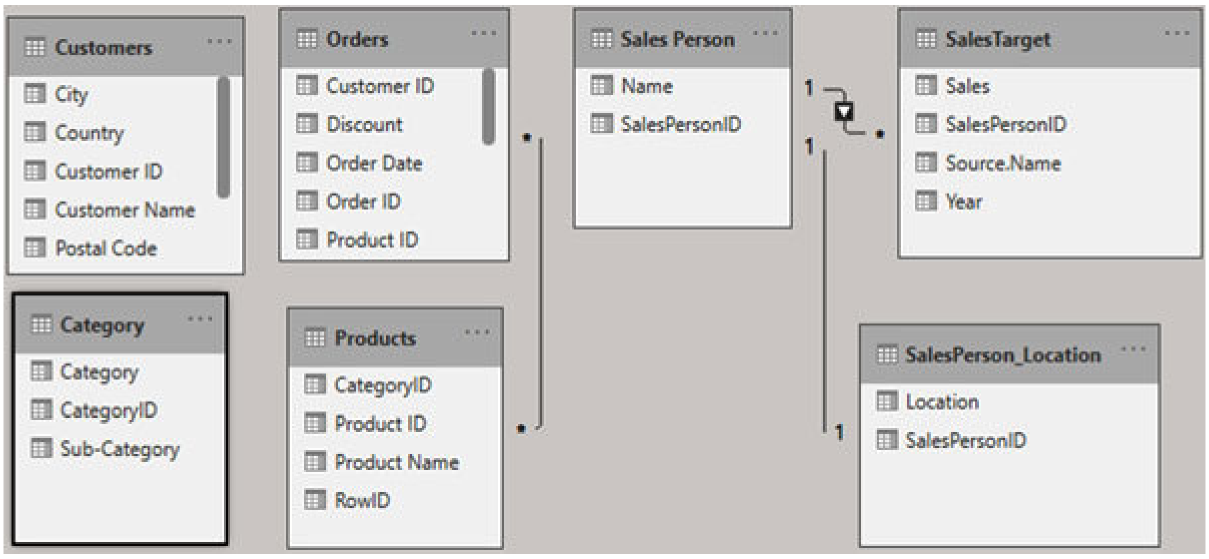


***Figure 3.2:*** *Tables loaded in Chapter 2*

This model can be viewed by launching **Chapter2_DbConn.pbix** and navigating to the model view. It displays the loaded tables and the relationships between the and **Sales Person** tables. These tables were auto-detected by Power BI.

In the subsequent sections, we will learn about the relationships in detail.

Review the relationship between the Orders and **Products** tables.

Save **Chapter2_DbConn.pbix** as

In the model view, we can see that Power BI has auto detected a relationship between the **Orders** and **Products** table. These tables should be in relationship using the keys in these tables, that is, **Product** We will review the relationship to see if Power BI has detected it correctly.

Hover over the relationship and right click. Select as shown in the following screenshot:

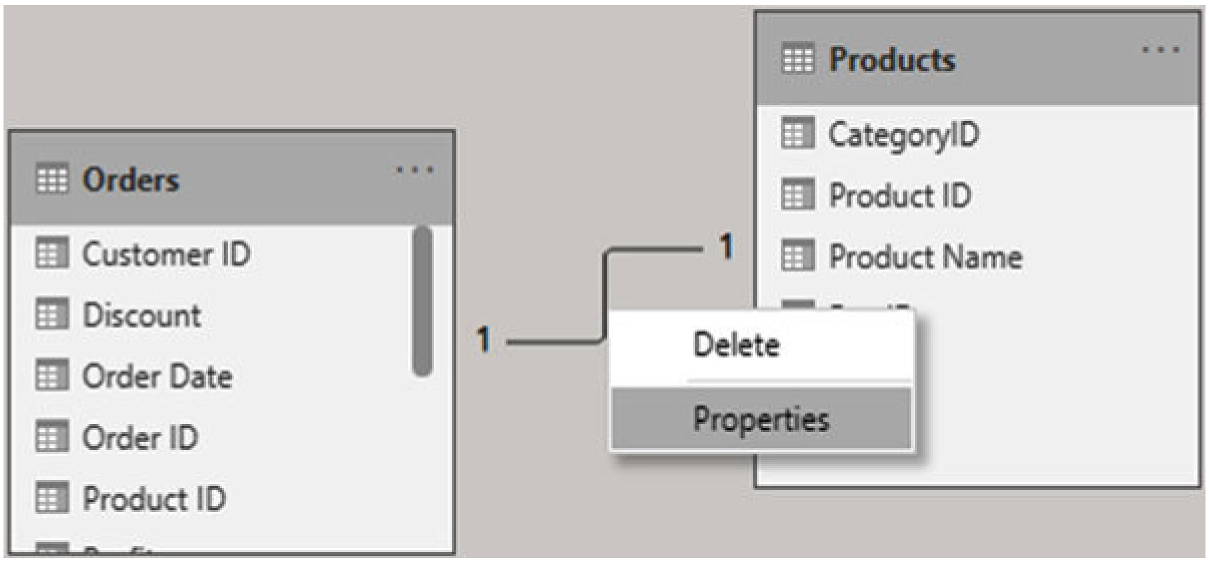
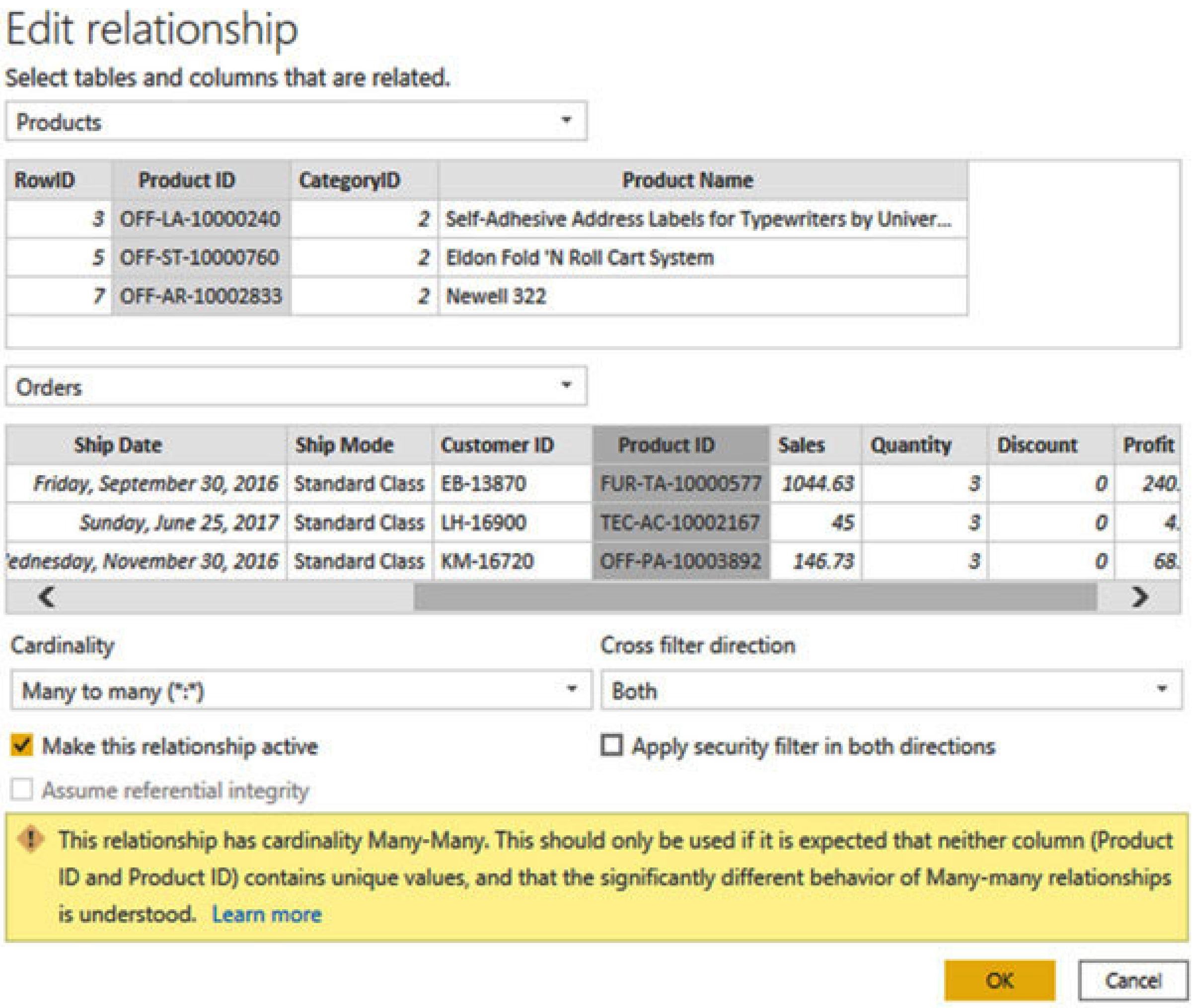


***Figure 3.3:*** *Review relationship between Orders and Products*

This will take you to the **Edit relationship** dialog box. Notice the highlighted/greyed column, Power BI has auto relationship based on these columns, which is correct, as shown in the following screenshot:


Edit relationship

Select tables and columns that are related.

Products

| RowID | Product ID | CategoryID | Product Name |
|---|---|---|---|
| 3 | OFF-LA-10000240 | 2 | Self-Adhesive Address Labels for Typewriters by Univer... |
| 5 | OFF-ST-10000760 | 2 | Eldon Fold 'N Roll Cart System |
| 7 | OFF-AR-10002833 | 2 | Newell 322 |

Orders

| Ship Date | Ship Mode | Customer ID | Product ID | Sales | Quantity | Discount | Profit |
|---|---|---|---|---|---|---|---|
| Friday, September 30, 2016 | Standard Class | EB-13870 | FUR-TA-10000577 | 1044.63 | 3 | 0 | 240. |
| Sunday, June 25, 2017 | Standard Class | LH-16900 | TEC-AC-10002167 | 45 | 3 | 0 | 4. |
| 'ednesday, November 30, 2016 | Standard Class | KM-16720 | OFF-PA-10003892 | 146.73 | 3 | 0 | 68. |

Cardinality: Many to many (*:*)

Cross filter direction: Both

☑ Make this relationship active

☐ Apply security filter in both directions

☐ Assume referential integrity

This relationship has cardinality Many-Many. This should only be used if it is expected that neither column (Product ID and Product ID) contains unique values, and that the significantly different behavior of Many-many relationships is understood. Learn more

***Figure 3.4:*** *Relationship between Orders and Products*

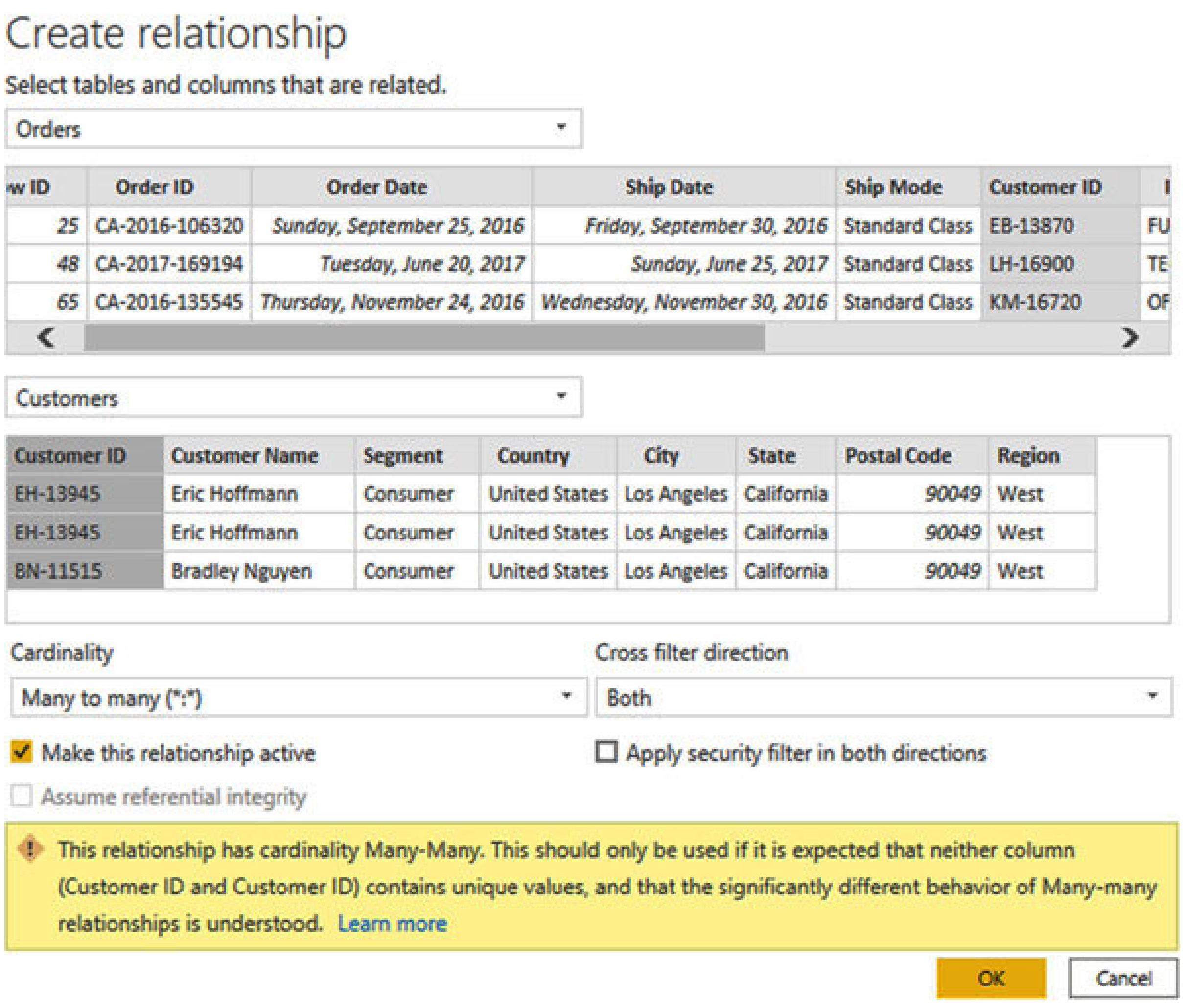
Using the edit relationship, we can modify any relationship or cardinality between the two tables.

## ***Creating a manual relationship between the Orders and Customers table***

Power BI was unable to detect the relationship between these tables, so we will create one manually, the steps to which are as follows:

From select **Manage** The **Manage relationships** dialog box is displaying one **Active** that is, between **Orders** and **Products** table. From the buttons at the bottom, click on

In the **Create relationship** dialog box, select the tables and columns that are related. Select **Orders** and **Customers** for tables. Select **Customer ID** in both the tables as this field will be used to create the relationship. Power BI automatically detects this relationship to be **Many to** as shown in the following screenshot:


Create relationship

Select tables and columns that are related.

Orders

| w ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | I |
|---|---|---|---|---|---|---|
| 25 | CA-2016-106320 | Sunday, September 25, 2016 | Friday, September 30, 2016 | Standard Class | EB-13870 | FU |
| 48 | CA-2017-169194 | Tuesday, June 20, 2017 | Sunday, June 25, 2017 | Standard Class | LH-16900 | TE |
| 65 | CA-2016-135545 | Thursday, November 24, 2016 | Wednesday, November 30, 2016 | Standard Class | KM-16720 | OF |

Customers

| Customer ID | Customer Name | Segment | Country | City | State | Postal Code | Region |
|---|---|---|---|---|---|---|---|
| EH-13945 | Eric Hoffmann | Consumer | United States | Los Angeles | California | 90049 | West |
| EH-13945 | Eric Hoffmann | Consumer | United States | Los Angeles | California | 90049 | West |
| BN-11515 | Bradley Nguyen | Consumer | United States | Los Angeles | California | 90049 | West |

Cardinality: Many to many (*:*)

Cross filter direction: Both

✓ Make this relationship active

☐ Apply security filter in both directions

☐ Assume referential integrity

This relationship has cardinality Many-Many. This should only be used if it is expected that neither column (Customer ID and Customer ID) contains unique values, and that the significantly different behavior of Many-many relationships is understood. Learn more

***Figure 3.5:*** *Creating a manual relationship*

Click on The next screen will show two active relationships.

In the model view, we can see the relationship between the **Orders** and **Customers** table.

Creating a manual relationship is a good option when Power BI has not auto detected the relationship or when you want to check and fix an existing relationship.

The cross-filter direction in the preceding Many to Many cardinality shows how the data in the tables will be filtered. For the Orders and Customers tables, it has three values – Customers filters Orders, Orders filters Customers, or Both. For the Products and Orders tables, it has values such as Orders filters Products, Products filters Orders, and Both. Default is Both.

## *Creating manual relationship between Products and Category tables*

In the model view, we can verify the loaded tables. Power BI is unable to detect the relationship between the **Products** and **Category** tables, so we will create one manually, the steps to which are as follows:

From the **Home** ribbon, click on **Manage** In the next dialog box, select

Select **Products** and **Category** for tables. Select **CategoryID** in both the tables since the relationship will be based on

Based on the data in the tables, Power BI will auto-detect the **Many to Many** relationship between the tables.

Select **OK** and then

Save your application.

Power BI auto-detects the relationship but we always have the option to change the default relationship.

**Review the data model:**

After making the necessary relationship changes, we can review the data model, as shown in the following figure:

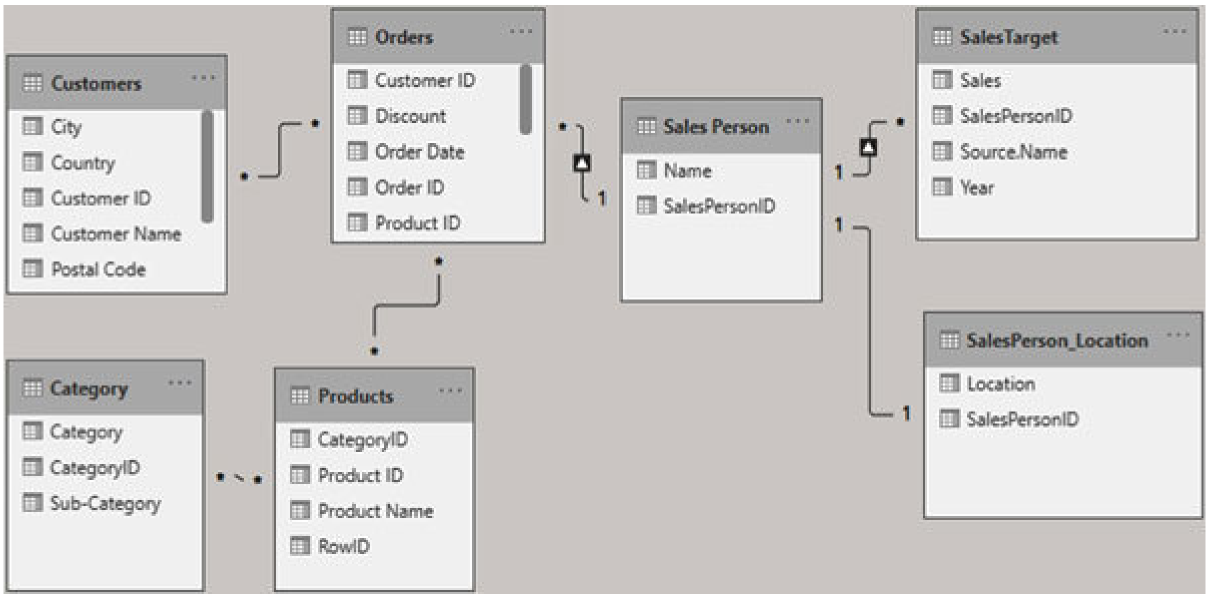


***Figure 3.6:*** *Model view after relationship changes*

It is always a good practice to review the data model after each change.

## ***Combining queries Using joins***

Power BI supports combining the queries using the merge queries option in Query Editor. It supports all the different types of SQL joins. Joins are used to display the data from multiple tables or when we have to merge two tables based on a join condition.

The following join types are present in Power BI and SQL:

**Left outer join:** Returns all the rows from the first table and only the matching rows from the second table.

**Right outer join:** Right join is opposite to the left join. Returns all the rows from the second table and only the matching rows from the first table.

**Full outer join:** Returns all the rows present in the first and the second table.

**Inner join:** Only the matching rows between the two tables are returned, else it returns zero rows.

**Left anti:** Returns all the rows from the first table which do not have a match in the second table.

**Right anti:** Returns all the rows from the second table which do not have a match in the first table.

Joins help in combining the data from multiple tables. It will also help in reducing the tables from the report view. Joins in Power BI are implemented using the **Merge Queries** in Query Editor, as shown in the following screenshot:

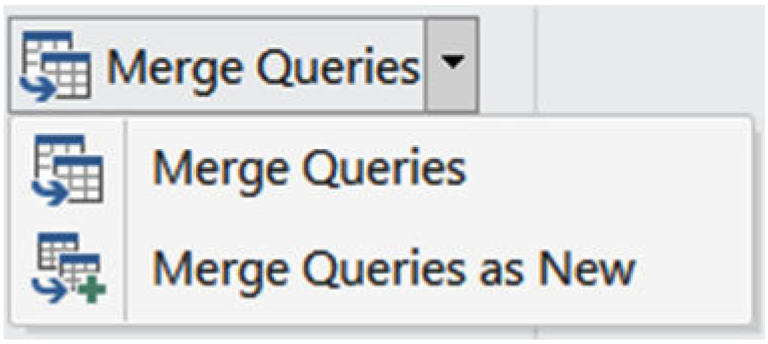


***Figure 3.7:*** *Merge Query*

Merge Queries have the following two options:

**Merge Queries:** It merges the second table with the first table.

**Merge Queries as New:** It creates a new table which is a result of the Join between the first and the second table.

We can use either option but the **Merge Queries** option should be preferred, as it will keep the original table/s and maintain the relationships with the other tables.

If **Merge Queries** as the **New** option is used, it will create a new table and we have to recreate the relationships.

## ***Joins***

To understand joins, we will be using As our current data model doesn’t require any joins, a few more tables will be introduced in this section to show how joins work. The first such table will be This table is present in the enclosed dataset.

## *Loading OrderDetails.xlsx*

For this exercise, save the previously created **Chapter3_CreatingRelationships** as Complete the following steps to load We learned how to load XLS file in *Chapter 2, Connect and*

The steps to load OrderDetails.xlsx are as follows:

From **Home** tab, select Browse to the enclosed data files and select In the navigator dialog box, select **OrderDetails** and select

From the **Home** tab, use the **Manage relationships** to create a relationship between **Orders** and **OrderDetail** based on the **Order ID** field.

The relationship between these tables will appear like the one in the following figure:

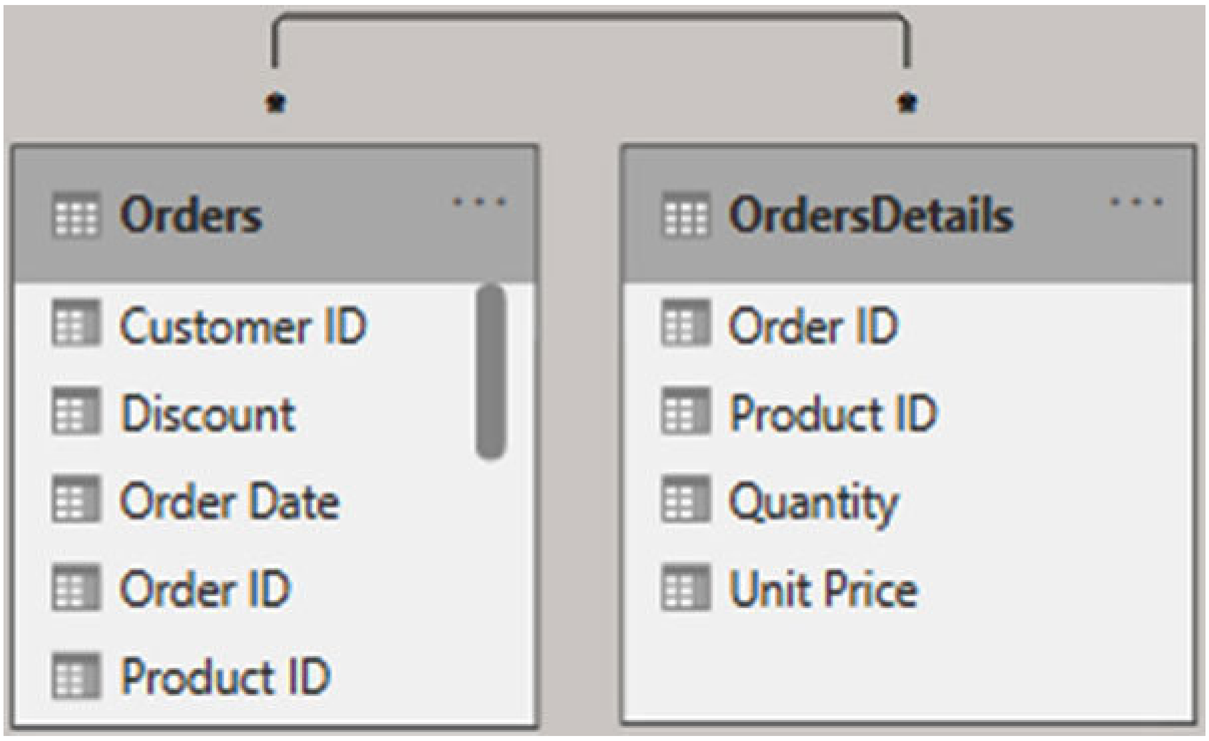


***Figure 3.8:*** *Orders and OrderDetails relationship*

There are other tables in the data model, but here only these two tables are displayed because we will be dealing with these tables in this section.

## ***Left outer join***

Orders and OrderDetail tables:

The **Orders** and **OrderDetail** tables can be joined, as **OrderDetails** contain the details of the orders present in the **Orders** table. The join operation to be performed on these tables will be a left outer join with orders as the first table, that is, we want to take all the records from the **Orders** table and only the matching **Order ID** records from the **OrderDetails** table. There may be a scenario where **OrderDetails** contains orders which are not present in the **Orders** table, the left outer join will avoid those orders.

The steps to the left join Orders and OrderDetails tables are as follows:

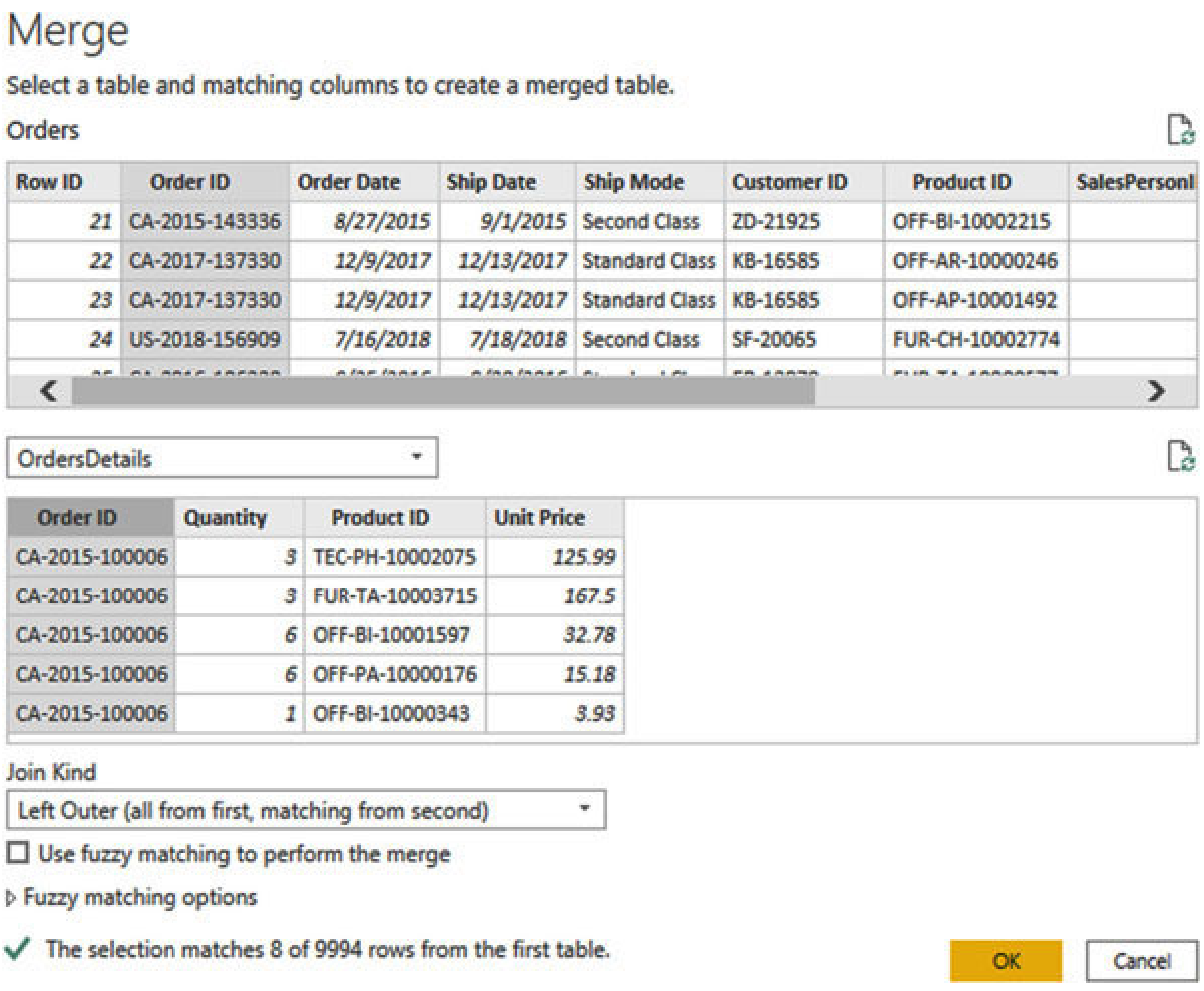
On the Power BI desktop, from the **Home** tab, select **Transform**

In Query Editor, from the queries on the left, select the **Orders** table and from the **Home** tab, select the **Merge Queries** option and then select **Merge**

In the merge dialog box, **Orders** is the first table and select **OrderDetails** as the second table.

Select **Order ID** in both the tables, as join will be performed based on the **Order**

Based on the data in the two tables, the join kind is already selected as left outer. We can change this join type, if needed, as shown in the following screenshot:



Merge

Select a table and matching columns to create a merged table.

Orders

| Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Product ID | SalesPersonI |
|---|---|---|---|---|---|---|---|
| 21 | CA-2015-143336 | 8/27/2015 | 9/1/2015 | Second Class | ZD-21925 | OFF-BI-10002215 | |
| 22 | CA-2017-137330 | 12/9/2017 | 12/13/2017 | Standard Class | KB-16585 | OFF-AR-10000246 | |
| 23 | CA-2017-137330 | 12/9/2017 | 12/13/2017 | Standard Class | KB-16585 | OFF-AP-10001492 | |
| 24 | US-2018-156909 | 7/16/2018 | 7/18/2018 | Second Class | SF-20065 | FUR-CH-10002774 | |

OrdersDetails

| Order ID | Quantity | Product ID | Unit Price |
|---|---|---|---|
| CA-2015-100006 | 3 | TEC-PH-10002075 | 125.99 |
| CA-2015-100006 | 3 | FUR-TA-10003715 | 167.5 |
| CA-2015-100006 | 6 | OFF-BI-10001597 | 32.78 |
| CA-2015-100006 | 6 | OFF-PA-10000176 | 15.18 |
| CA-2015-100006 | 1 | OFF-BI-10000343 | 3.93 |

Join Kind

Left Outer (all from first, matching from second)

☐ Use fuzzy matching to perform the merge

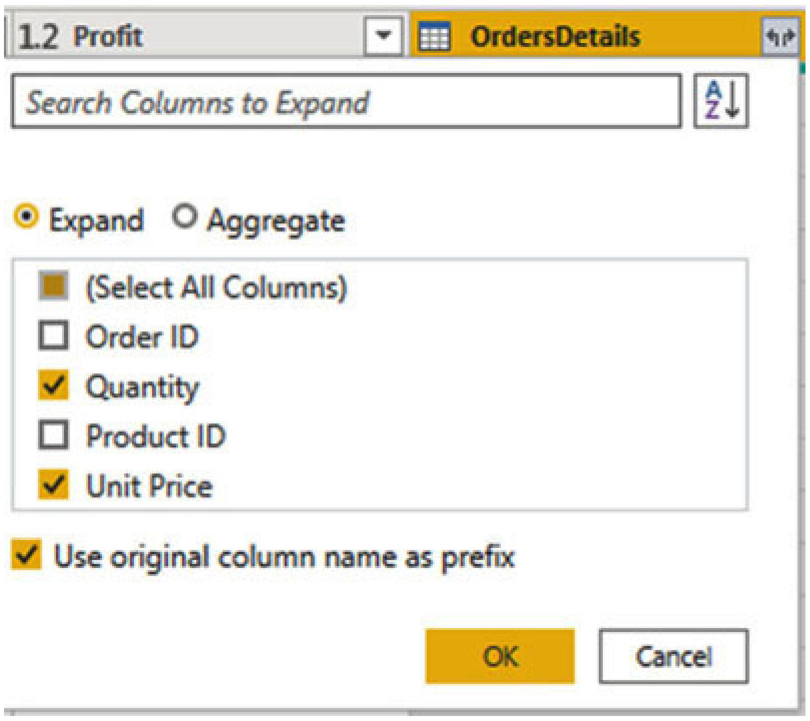
▷ Fuzzy matching options

✓ The selection matches 8 of 9994 rows from the first table.

***Figure 3.9:** Left Outer Join Orders and OrderDetails*

Click on

In the next data preview window, scroll all the way to the right and we will see **OrderDetails** as the last column. Click on the double arrows on the right to select the columns you want add to the **Merge1** table and click on **OK,** as shown in the following screenshot:



Expand [✓] Aggregate

- ▨ (Select All Columns)
- ☐ Order ID
- ✓ Quantity
- ☐ Product ID
- ✓ Unit Price

✓ Use original column name as prefix

***Figure 3.10:** Selecting columns from OrderDetails*

From the **File** menu, select **Close &**

In the Power BI desktop, under **Data** verify the fields in and the fields that came from the **OrderDetails** table. These fields will be prefixed by such as **OrderDetails.Quantity** and

We can rename these fields to remove the prefix **OrderDetail** as they are now part of the **Orders** table. Under right-click on

**OrderDetails.Quantity** and **OrderDetails.UnitPrice** and rename by removing The fields are renamed as **Quantity1** and **Unit**

In the data model, right click on the **OrderDetails** table and select **Hide in Report** The report view will show less tables as compared to earlier.

Save your application.

The merging or joining of table will combine the data in the two tables.

**Query Editor provides the option for Fuzzy matching when trying to find matches across joining the table columns. Fuzzy matching is a string comparison logic that compares the items in a separate list and joins them if they're close/match to each other. If performing the string comparison, you can specify Similarity threshold which indicates how similar the two values need to be in order to match.**

## ***Right outer join***

The right outer join is similar to the left outer join, the only difference being that the right outer join will take all the rows from the right table and the matching records from the left table.

The steps are similar to the left outer join, described earlier, only the join type will change.

The steps to the Right outer join **SalesTarget** and **Sales Person** tables are as follows:

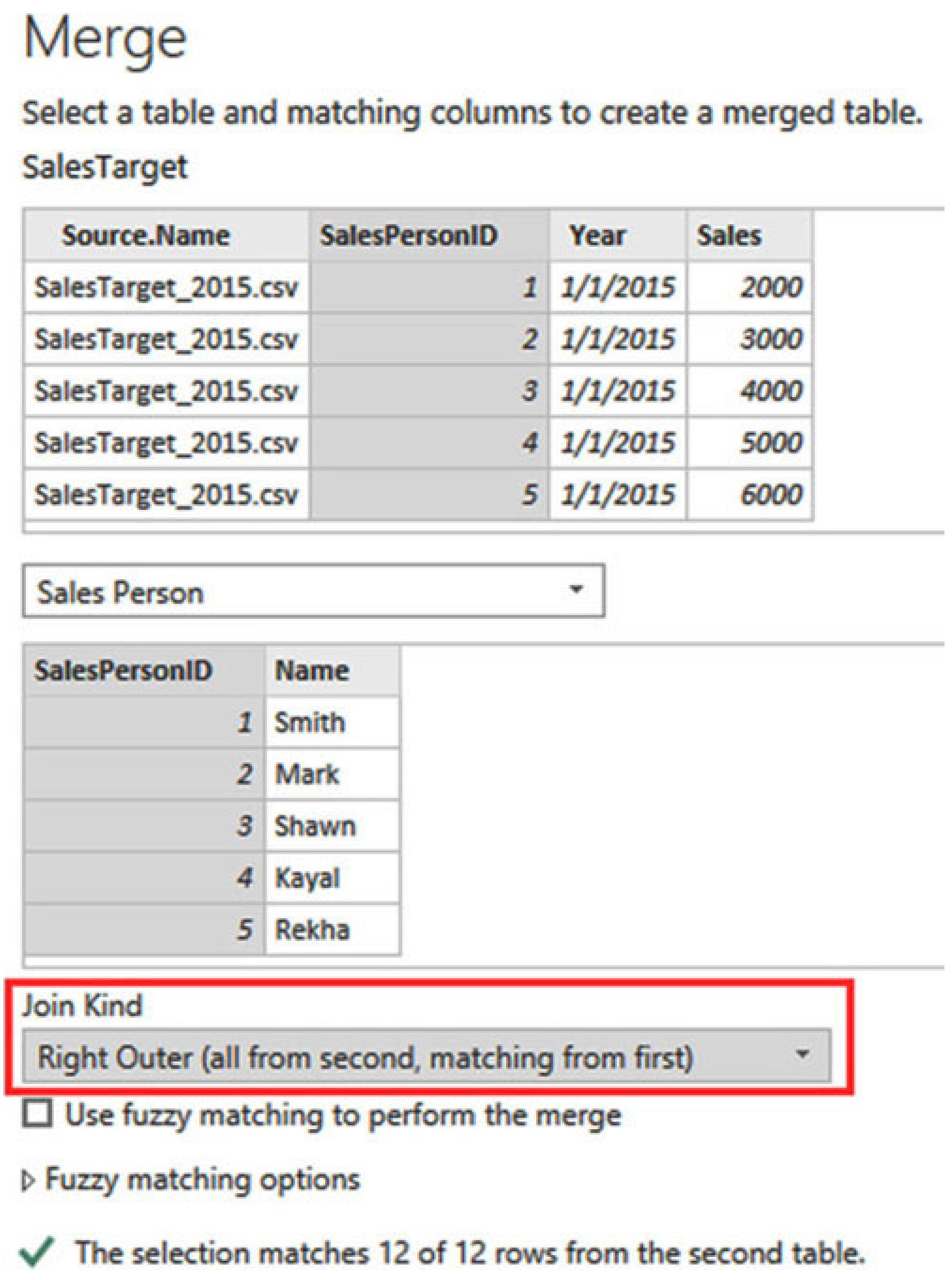
On the Power BI desktop, from the **Home** tab, select **Transform**

In Query Editor, from the **Queries** on the left, select the **SalesTarget** table and from the select **Merge Queries** and then select **Merge**

In the merge dialog box, **SalesTarget** is the first table and select **Sales Person** as the second table.

Select **SalesPersonID** in both the tables, as join will be performed based on the

Change the Join kind to **Right Outer,** as shown in the following screenshot:.



Merge

Select a table and matching columns to create a merged table.

SalesTarget

| Source.Name | SalesPersonID | Year | Sales |
|---|---|---|---|
| SalesTarget_2015.csv | 1 | 1/1/2015 | 2000 |
| SalesTarget_2015.csv | 2 | 1/1/2015 | 3000 |
| SalesTarget_2015.csv | 3 | 1/1/2015 | 4000 |
| SalesTarget_2015.csv | 4 | 1/1/2015 | 5000 |
| SalesTarget_2015.csv | 5 | 1/1/2015 | 6000 |

Sales Person

| SalesPersonID | Name |
|---|---|
| 1 | Smith |
| 2 | Mark |
| 3 | Shawn |
| 4 | Kayal |
| 5 | Rekha |

Join Kind

Right Outer (all from second, matching from first)

☐ Use fuzzy matching to perform the merge

▷ Fuzzy matching options

✓ The selection matches 12 of 12 rows from the second table.

***Figure 3.11:*** *Right Outer Join SalesTarget and SalesPerson*

In the next data preview window, scroll all the way to the right and we will see **Sales Person** as the last column. Click on the double arrows on the right to select the columns that we want add to the merged table.

Let’s select **Name** as it is not present in the **SalesTarget** table.

Select **OK** and **Close & Apply** to close Query Editor.

In the Power BI desktop, under **Data** verify the fields in the **SalesTarget** table; the fields that came from the **Sales Person** table is **Sales**

## ***Full outer join***

This returns all the rows present in the first and the second table. The process of performing the full outer join is similar to the previously mentioned joins. We have to change the join kind in Query Editor to full outer.

It will return all the rows from both the tables.

The steps to perform the Full Outer Join between Products and Category tables are as follows:

Follow the same steps as earlier, to create a join between the two tables. Select table and then select **Merge** In the **Merge** dialog box, select the first table as the **Products** table, and select the second table as the **Category** table. Select **CategoryID** in both the tables.

Change to **Join kind** to **Full** as shown in the following screenshot:

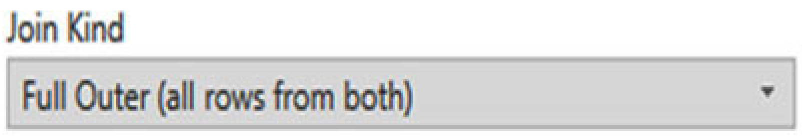

Join Kind
Full Outer (all rows from both)

***Figure 3.12:*** *Full Outer Join selection*

In the data preview window, we can select the columns we need from the **Category** table, that is, **Category** and **SubCategory** as these columns are not present in the **Products** table.

**Full outer joins should be avoided as it returns the data from both the tables and can take a long time to run.**

### ***Left_anti***

This returns all the rows from the first table which do not have a match in the second table.

## ***Right_anti***

This returns all the rows from the second table which do not have a match in the first table. These joins are self-explanatory and can be worked in the similar fashion as the previous joins.

## Combining_queries_using_append

The other way to combine Queries in Power BI is using the **Append Queries** option. Append is similar to Union in SQL. It combines the two queries/tables having the same columns and structure. The **Append Queries** option is also available in Query Editor, below the **Merge Queries** option, as shown in the following figure:

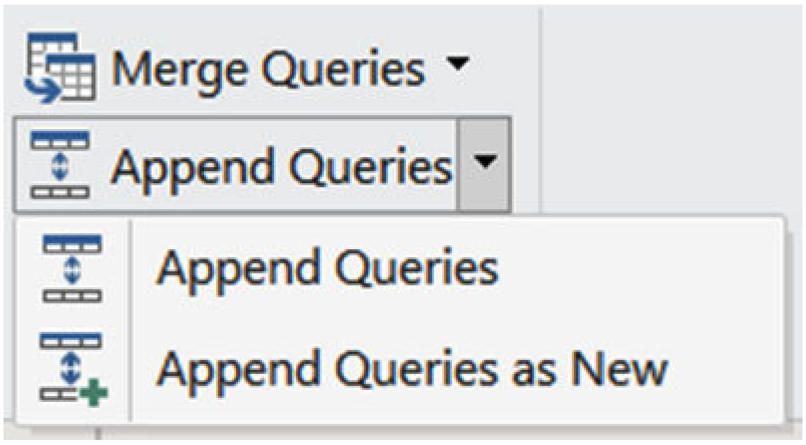


***Figure 3.13:*** *Append Queries*

It has the following two options:

**Append** It appends/combines the second table in the first table.

**Append Queries as** It appends/combines the query and creates a new table, leaving the old ones in the original format.

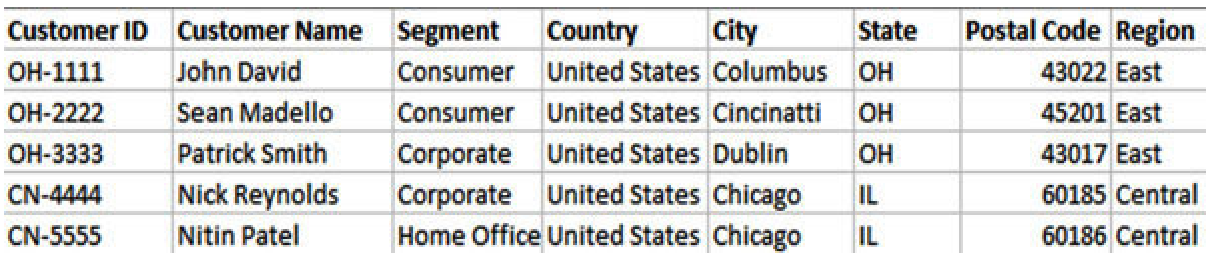
We can use any of the previously mentioned options; the first option should be preferred as it will not create a new table but simply append the rows in the existing table.

## *Appending records from NewCustomers table to Customers table*

The Append Queries option is used when in a business scenario, we have to append new records to an existing table. For example, we have an existing **Customers** table which currently has 994 rows. We receive more customer records which need to be added to this table. In such cases, the **Append Queries** option is used.

In this exercise, we will append the existing customers table with a new list of customers. We will do so, using the **Append**

For this example, we will use This file is present in the enclosed dataset. It contains five customer records in the same format as the **Customers** table. We will append these records to the existing **Customers** table, as shown in the following screenshot:



| Customer ID | Customer Name | Segment | Country | City | State | Postal Code | Region |
|---|---|---|---|---|---|---|---|
| OH-1111 | John David | Consumer | United States | Columbus | OH | 43022 | East |
| OH-2222 | Sean Madello | Consumer | United States | Cincinatti | OH | 45201 | East |
| OH-3333 | Patrick Smith | Corporate | United States | Dublin | OH | 43017 | East |
| CN-4444 | Nick Reynolds | Corporate | United States | Chicago | IL | 60185 | Central |
| CN-5555 | Nitin Patel | Home Office | United States | Chicago | IL | 60186 | Central |

***Figure 3.14:*** *NewEmployees.xls*

The steps to be followed will be, load the **NewCusotmers.xls** and then append it to **Customers** and then hide **NewCustomers** table.

**When using Append, the second table should have the same column headers and data types as the first table. If the column headers are different, the second table will create new columns and show nulls for the corresponding column headers in the first table. So before performing Append, clean your data to make sure the column headers in both the tables are the same.**

The steps to append NewCustomers to the existing Customers table are as follows:

In the Power BI desktop, from select

Browse to your data folder and select In the **Navigator** dialog box, select the **NewCustomers** and select This will create the **NewCustomers** table in the model view.

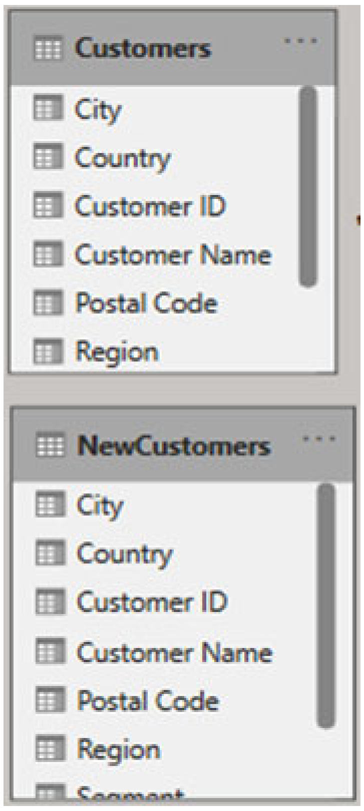


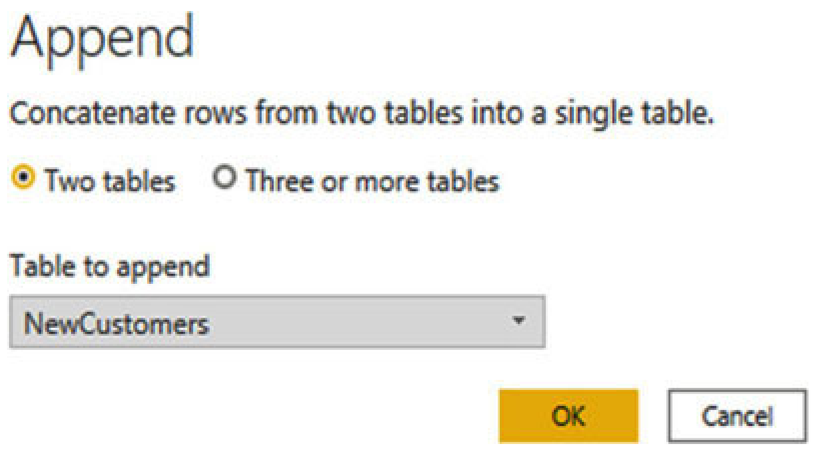
***Figure 3.15:*** *NewCustomers.xls table loaded in the data model*

Now we will append this table with the **Customers** table. From the **Home** tab, select **Transform data** to navigate to Query Editor.

In Query Editor, under select **Customers** from the left and from select **Append Queries** and then select **Append**

From the **Append** dialog box, select the two tables as we are only concatenating/ combining rows from two tables. Select the **NewCustomers** table in the table to append.

Select as shown in the following screenshot:



Append

Concatenate rows from two tables into a single table.

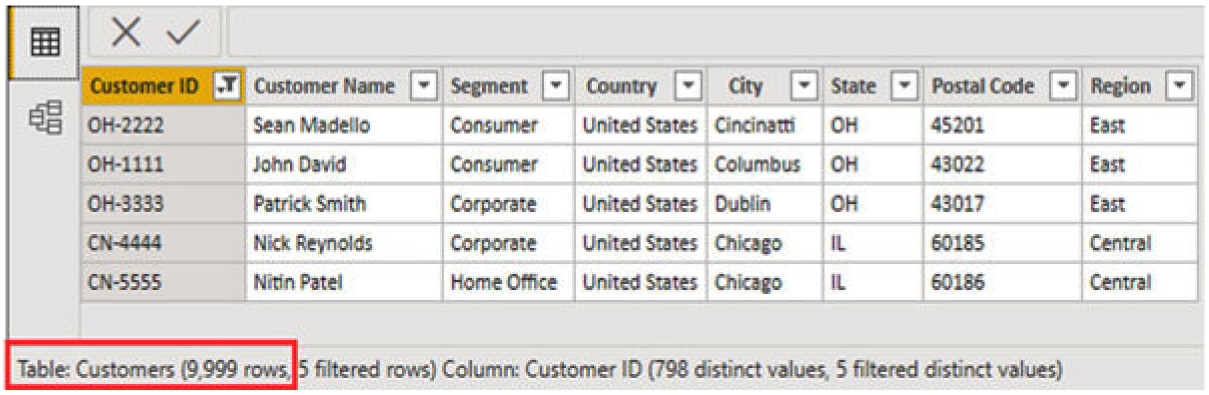
[✓ Two tables] Three or more tables

Table to append: NewCustomers

***Figure 3.16:*** *Appending the tables*

From the **File** tab, select **Close &**

On the Power BI desktop, in the **Data** select we can see that the records from the new table are appended to the **Customers** table. We can filter in **Customer ID** to see the new rows, as shown in the following screenshot:



| Customer ID | Customer Name | Segment | Country | City | State | Postal Code | Region |
|---|---|---|---|---|---|---|---|
| OH-2222 | Sean Madello | Consumer | United States | Cincinatti | OH | 45201 | East |
| OH-1111 | John David | Consumer | United States | Columbus | OH | 43022 | East |
| OH-3333 | Patrick Smith | Corporate | United States | Dublin | OH | 43017 | East |
| CN-4444 | Nick Reynolds | Corporate | United States | Chicago | IL | 60185 | Central |
| CN-5555 | Nitin Patel | Home Office | United States | Chicago | IL | 60186 | Central |

Table: Customers (9,999 rows, 5 filtered rows) Column: Customer ID (798 distinct values, 5 filtered distinct values)

***Figure 3.17:*** *Customers table with the appended rows*

After performing the preceding steps, we will notice that Customers table has 999 records.

Now we can hide the **NewCustomers** table from the data model, as all the records are added to the **Customers** table.

## ***Conclusion***

Data model is an important component of a BI implementation as all the data used in visualization is sourced from a data model. In this chapter, we learned how to create a data model by creating relationships. We learned the different kinds of relationships available in Power BI.

We also learned that data modeling requires combining the data using merge and append. Combining the tables reduces the number of joins performed while creating visualization, and thus improves the overall performance of the application.

In the next chapter, we will dive deeper into DAX and learn about creating the different kinds of aggregations and other functions using DAX.

## ***Questions***

How is a data model created?

How are relationships created in Power BI?

What are the different techniques to combine the tables?

What is the difference between Merge and Append Queries?

What are the different ways to reduce the tables and columns in the report view?

## *Answers*

Data model is created by defining the relationships between the data elements. Relationships are created based on the matching data between the two columns or based on the primary and foreign keys between the tables.

Relationships in Power BI are created using the **Manage relationship** option.

Tables can be combined using the **Merge** and **Append**

The Merge Query option is used to create join between the tables. The **Append Queries** option is similar to Union in SQL. It simply appends/concatenates one table into another.

From the data model, we can delete the table or choose **Hide in report** We can do the same for the fields. The hidden table/column will not be displayed in the Report view while creating visualization.

CHAPTER 4

# Data Analysis Expressions (DAX)

## ***Introduction***

In the previous chapter, we created relationships between the tables. In this chapter, we will learn about **Data Analysis Expressions** We will look at some of the important functions of DAX. This chapter will explain how to create calculated columns and measures. It will also cover the aggregation, string, conditional, and date functions.

As we develop a Power BI application, it is important to learn why and how to create new columns and measures. A knowledge of DAX is a must for a Power BI developer, as the formulas created using DAX will provide more context to the users.

## ***Structure***

In this chapter, we will discuss the following topics:

Introduction to DAX

Calculated Column

Calculated Measure

Mathematical Functions

Count Functions

Information Functions

Logical Functions

Filter Functions

Date Time Functions

Variables in DAX Functions

Using Variables in DAX

## ***Objectives***

The objective of this chapter is to take a deep dive into the DAX concepts and functions. We will understand the difference between the calculated columns and measures. We will learn how to create new columns and measures using the various DAX functions. We will also be creating the Date Dimension using DAX and will further use the variables to simplify the calculations.

## ***Introduction to DAX***

DAX is the formula language used in Power BI. DAX is a functional language and contains a set of functions. The DAX functions can contain other functions, conditional statements, and value references. The DAX expressions are created using the functions which are written in a single line.

DAX has only two datatypes, which are as follows:

Numeric – Integers, Decimals, and Currency

Other – Strings and Binary objects

## ***DAX functions***

DAX has some 200 functions; in this section, we will discuss some of the important functions in DAX. Before looking at the functions in detail, let's understand what kind of values they return.

Depending on the type of values returned, every DAX function is divided into the following:

**Scalar** Return single values such a number, text, or date. These values can be used in a visualization.

**Table** Return a table or values, that is, multiple rows. These functions cannot be used in visualization as visualizations, expect a single value such as etc. The table functions can be used in the DAX queries as an input into the other DAX expressions. They can also be used in creating tables in the Power BI desktop.

We will explore the different DAX functions. Each function will have a return value, such as Scalar or Table.

DAX provides a set of functions to create a formula or expression. The different types of functions are as follows:

**Mathematical and trig** These functions are similar to the mathematical and trigonometric functions available in excel. Some of the functions available are etc.

**Logical** These set of functions check a conditional expression to see if it evaluates to TRUE or FALSE. Some of the functions available are etc.

**Information** These functions look at a table/column which is provided as an argument and tells whether the value matches the expected type. Some of the functions available are **IS** etc.

**Text** These functions are string functions and work on the Table and Columns. Some of the functions available are etc.

**Date** These functions assist in creating the calculations based on dates and times. Some of the functions available in this category are etc.

**Filter** These functions work with tables and relationships. They help in manipulating the data context to create dynamic calculations. Some of the functions available are etc.

DAX helps in creating calculated columns and measures which are not present in the source data.

**Syntax:**

DAX works with Table/s and Table columns. The syntax for Table and Column used in DAX is as follows:

TableName[ColumnName]

If the table name contains a space, the table name is enclosed within single quotes.

'Table Name' [ColumnName]

**The DAX expressions are written in the Power BI desktop. These expressions can be written either in Report view or Data view.**

## ***Calculated Columns and Measures***

In DAX, two kinds of calculations are created, which are as follows:

Calculated Columns

Calculated Measures

Depending on the requirements, we can choose whether to create a column or a measure.

In this section, we will see when and how to create these calculations.

## ***Calculated Columns***

A calculated column extends the loaded table in Power BI. It is created by using the DAX formula that is evaluated for each row. It adds a physical column to the table. This column is created by using the data already present in the data model.

For example, a **Customers** table contains each customer's city and state as separate columns. In the visualization, however, we want to display (for each row) a concatenation of City-State. To do this, we will create a calculated column, say and write the following DAX formula to perform the concatenation:

CityState = CONCATENATE(Customers[City], CONCATENATE("-
",Customers[State]))

This new column will be stored in the table and will be available in the table field list. Any time the data refreshes, this column calculation will also be refreshed and provide the updated results.

The properties of a calculated column are as follows:

Calculated columns are created in the Report or Data view.

The values are computed, when the expression is defined or when the dataset is refreshed.

They are evaluated for each row within the table.

Created calculated columns appear in the field list, just like any other fields, and are displayed with a special icon.

Once created, they are part of the data model/table in which they are created, thus, increasing the size of the data model and the consumption of RAM.

Calculated columns are used to implement the business rules and use the fields present in the data source tables.

Calculated columns provide a way to add a column in a table.

**A calculated column is used when it is not present in the data source table and its physical representation is required. These columns should be avoided as they increase the size of the data model.**

**Exercise 1:**

**Create a calculated column – Unit Price**

The loaded **Orders** table present in our data model is missing Unit Price. We will create the **Unit Price** column by using the calculated column functionality; in this way, it will be physically present in the table.

**The steps to create a calculated column named UnitPrice are as follows:**

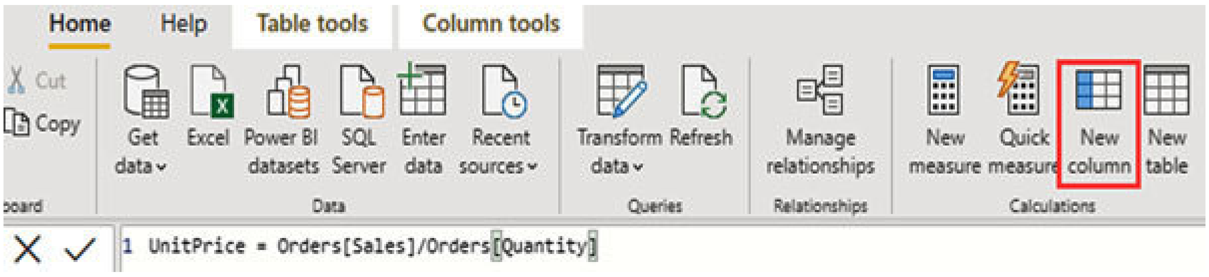
Use the application **Chap3_CreatingRelationships** and save it as

From the Power BI desktop, navigate to **Data view** (or Report view), and select the **Orders** table. In this table, we will be creating a **New**

From the **Home** tab, select **New** as shown in the following figure, and in the DAX edit window, write an expression.

On the expression window, replace column with **Unit Price** and after the = (equals to) sign, type The drop down will display the list of columns in the **Orders** table. and then type / and type **Orders** again and select

Press *Enter* after typing the expression.



```
1 UnitPrice = Orders[Sales]/Orders[Quantity]
```

***Figure 4.1:** Creating calculated column*

The expression should look like the following:

UnitPrice = Orders[Sales]/Orders[Quantity]

In the preceding calculation, **UnitPrice** is the name of the calculated column, **Orders** is the table name, and **Sales** and **Quantity** are the fields in the table

The preceding steps will create a column **UnitPrice** in the **Orders** table. This column will be physically present in the table. We can verify it under the data preview, fields, or in the data model, as shown in the following screenshot:

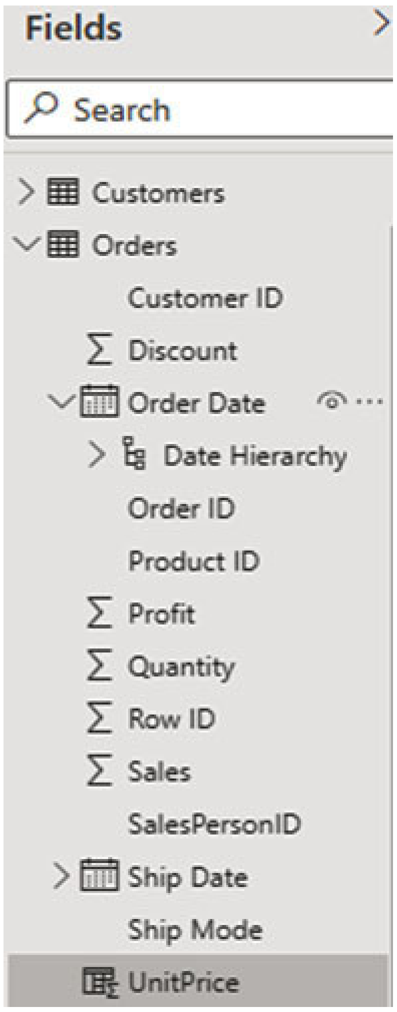


***Figure 4.2:*** *UnitPrice column created in Orders table*

The **UnitPrice** column will be displayed in the field list with a special icon. Whenever a column with such icon is displayed, we can be sure that a formula is associated with it.

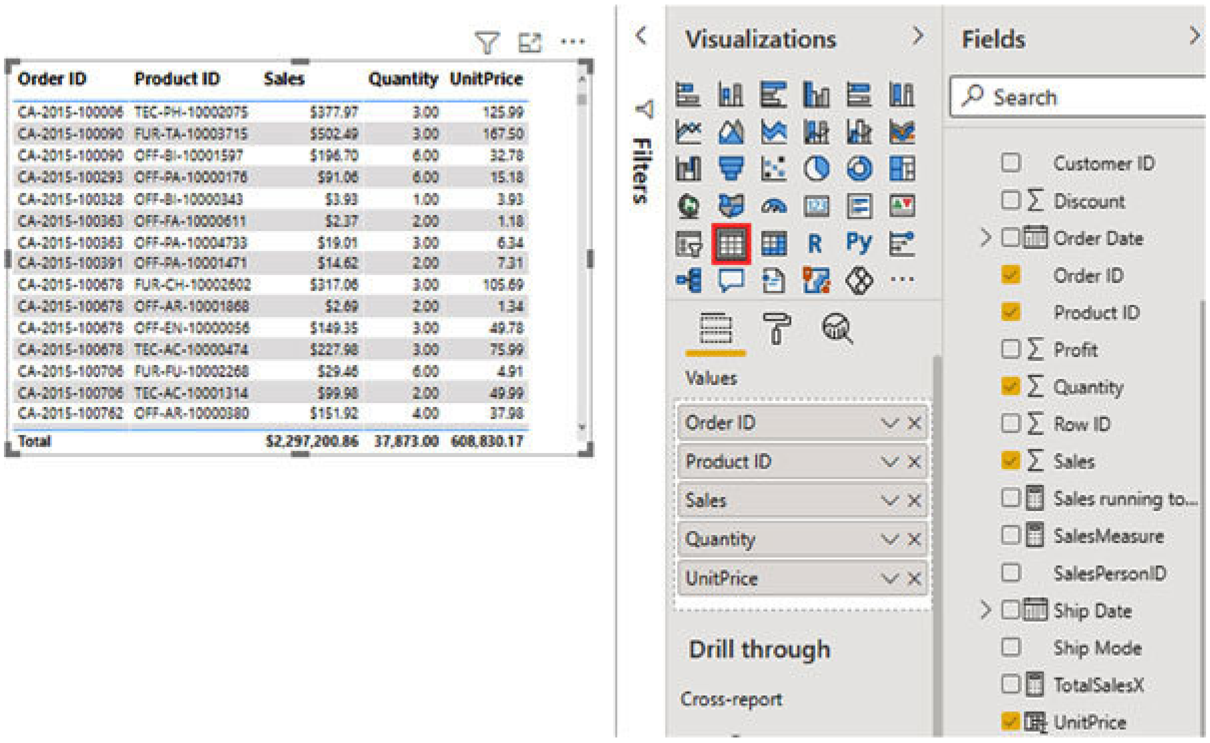
## *Verify the UnitPrice calculation*

We will cover visualizations in much detail in the next chapter. In this chapter, to verify our calculation, we will create a simple visualization table. (Refer to the page on Calculated Column in the enclosed

The steps to verify the UnitPrice calculation are as follows:

Navigate to the Report view. From the list of select

In Fields, from the **Orders** table, select and as shown in the following screenshot:



| Order ID | Product ID | Sales | Quantity | UnitPrice |
|---|---|---|---|---|
| CA-2015-100006 | TEC-PH-10002075 | $377.97 | 3.00 | 125.99 |
| CA-2015-100090 | FUR-TA-10003715 | $502.49 | 3.00 | 167.50 |
| CA-2015-100090 | OFF-BI-10001597 | $196.70 | 6.00 | 32.78 |
| CA-2015-100293 | OFF-PA-10000176 | $91.06 | 6.00 | 15.18 |
| CA-2015-100328 | OFF-BI-10000343 | $3.93 | 1.00 | 3.93 |
| CA-2015-100363 | OFF-FA-10000611 | $2.37 | 2.00 | 1.18 |
| CA-2015-100363 | OFF-PA-10004733 | $19.01 | 3.00 | 6.34 |
| CA-2015-100391 | OFF-PA-10001471 | $14.62 | 2.00 | 7.31 |
| CA-2015-100678 | FUR-CH-10002602 | $317.06 | 3.00 | 105.69 |
| CA-2015-100678 | OFF-AR-10001868 | $2.69 | 2.00 | 1.34 |
| CA-2015-100678 | OFF-EN-10000056 | $149.35 | 3.00 | 49.78 |
| CA-2015-100678 | TEC-AC-10000474 | $227.98 | 3.00 | 75.99 |
| CA-2015-100706 | FUR-FU-10002268 | $29.46 | 6.00 | 4.91 |
| CA-2015-100706 | TEC-AC-10001314 | $99.98 | 2.00 | 49.99 |
| CA-2015-100762 | OFF-AR-10000380 | $151.92 | 4.00 | 37.98 |
| Total | | $2,297,200.86 | 37,873.00 | 608,830.17 |

***Figure 4.3:*** *Table displaying Unit Price*

Using the table visualization, we can verify any calculation in this chapter.

**Exercise 2:**

**Create a calculated column – City-State**

In the **Customers** table, the City and State of the customers are displayed in separate columns, we will create a calculated column to create **CityState** which will be a concatenation of **City** and **State** and will be displayed as City-State.

The steps to create a calculated column named City-State are as follows:

(Refer to the page on Calculated Column in the enclosed

From the Power BI desktop, navigate to **Data View** (or Report view), select the **Customers** table. In this table, we will be creating a **New**

From the **Home** tab, select **New Column** and in the DAX edit window, write the following expression:

```
CityState = CONCATENATE(Customers[City], CONCATENATE("-
",Customers[State]))
```

In the preceding formula, **CONCATENATE** is a DAX function that joins two text strings into one string. The second **CONCATENATE** is used because we want to display a hyphen (-) in between **City** and

This will create a column **CityState** in the **Customers** table, as shown in the following screenshot:

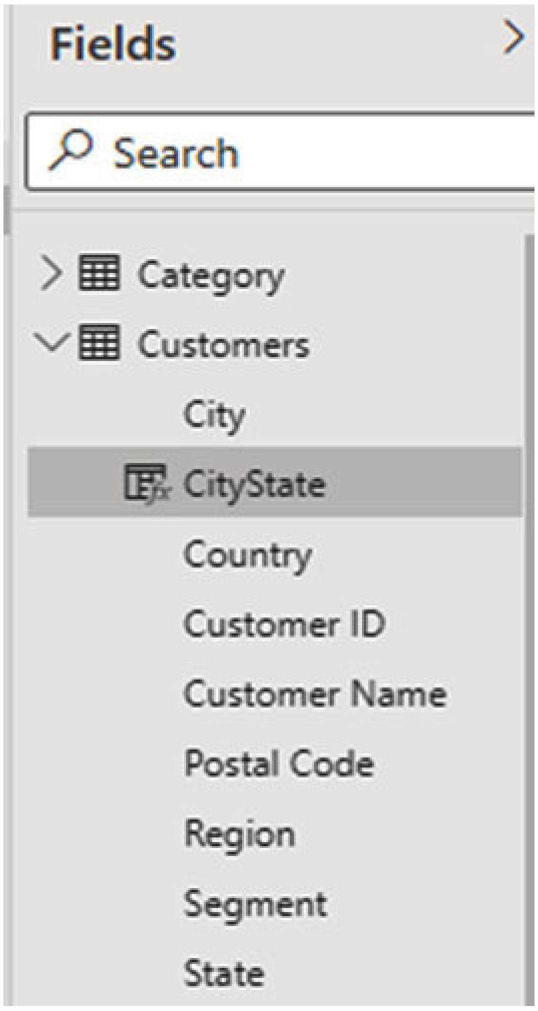


***Figure 4.4:*** *CityState column in Customers table*

The **CityState** column will be physically stored in the table, and therefore will increase the size of the data model.

## ***Calculated measure***

The calculated measures are not stored physically in the table, but rather are calculated on the fly. They are evaluated, based on the context in which they are used. The way in which they differ from the calculated column is that they are calculated based on the context where they are used. They don't depend on a specific table, but rather where they are used. A calculated column, on the other hand, is used within the table, and therefore are physically stored within a specific table.

We can put a calculated measure in any table, it will not lose its functionality.

For example, if we create a calculated measure say **TotalSales** = **SUM(Orders [Sales]).**

This measure will calculate on the fly when used in the visualization or matrix. It will provide the **Total of Sales** when used with the different tables and dimensions, such as **TotalSales** for **TotalSales** for etc.

The properties of a calculated measure are as follows:

Calculated measures are also created in the report or data view.

Unlike calculated columns, calculated measures are not stored as part of the data model. They are calculated on the fly.

Does not increase the size of the data model or RAM overhead.

It is not computed row by row but uses an aggregation function like **Sum** around the fields. If row-by-row computation is required, then the **X** function iterator, such as **SumX,** is used.

Created calculated measures appear in the field list, just like any other fields, but are displayed with a special calculator icon.

They are not displayed in the data preview in the data view.

Calculated measures are suited for calculating percentages, ratios, and complex aggregations.

**Exercise 3:**

**Create a calculated measure – SalesMeasure**

In this exercise, we will create a calculated measure **SalesMeasure** in the **Orders** table, by using **Quantity** and **UnitPrice** from the **Orders** table.

The steps to create a calculated measure are as follows:

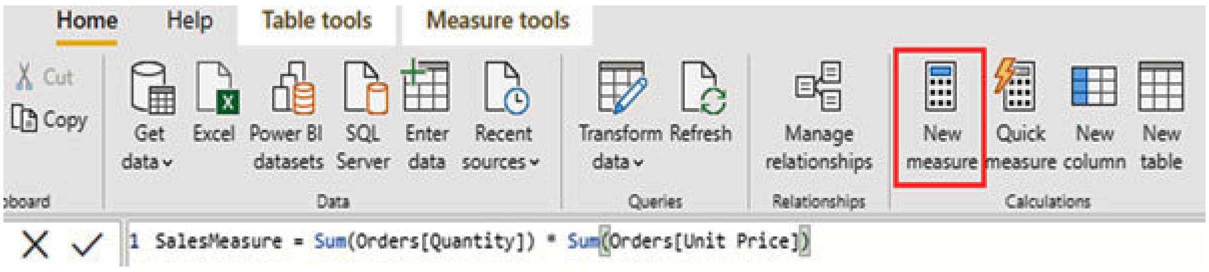
(Refer to the page on Calculated Measure in the enclosed

From the Power BI desktop, navigate to **Data View** (or Report view), and select the **Orders** table.

From select **New** as shown in the following figure, and in the DAX edit window, write an expression.

In the DAX expression for a measure, the field names are enclosed in an aggregation function such as

SalesMeasure = Sum(Orders[Quantity]) * Sum(Orders[UnitPrice])



```
SalesMeasure = Sum(Orders[Quantity]) * Sum(Orders[Unit Price])
```

***Figure 4.5:*** *Creating calculated measure*

Notice that the calculated column and measure are displayed with different icons in the following figure:

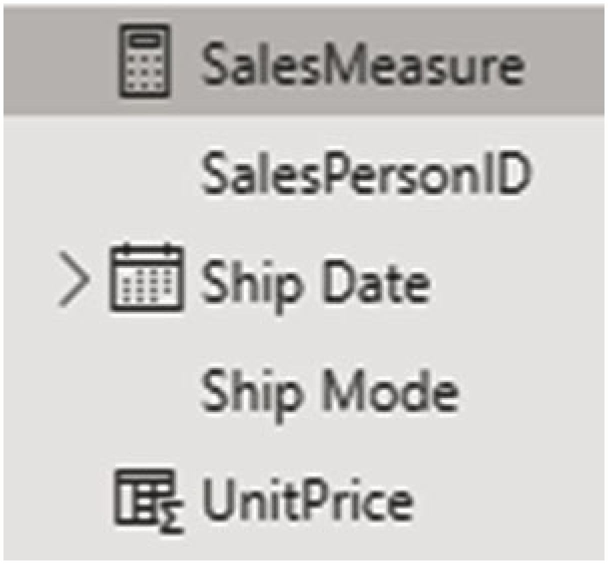


***Figure 4.6:*** *Calculated Column and Measure in Field list*

The **SalesMeasure** column will not be visible in the data preview, as it is not physically present in the table, but will be calculated on the fly. The **UnitPrice** column will be visible in the data preview.

## ***Quick measures***

Quick measures are also used to create calculations. The following are some of the important characteristics of quick measures:

Predefined calculations provided by Power BI.

Provides different categories of calculations.

We don't write the DAX expressions for these calculations; Power BI automatically writes DAX based on the input provided.

The DAX code executed for the measure can be seen.

New quick measure, once created, will be available as a column in the field list and the data model in the table where it is created.

The new quick measure is available to any visual in the reports.

Quick measures can be renamed or deleted.

Quick measures are a great way to learn DAX because we can see how the different calculations are written. Power BI provides various kinds of quick measures. These are available to us like any other measure calculation, the only difference is that DAX for these measures is predefined by Power BI, as shown in the following screenshot:

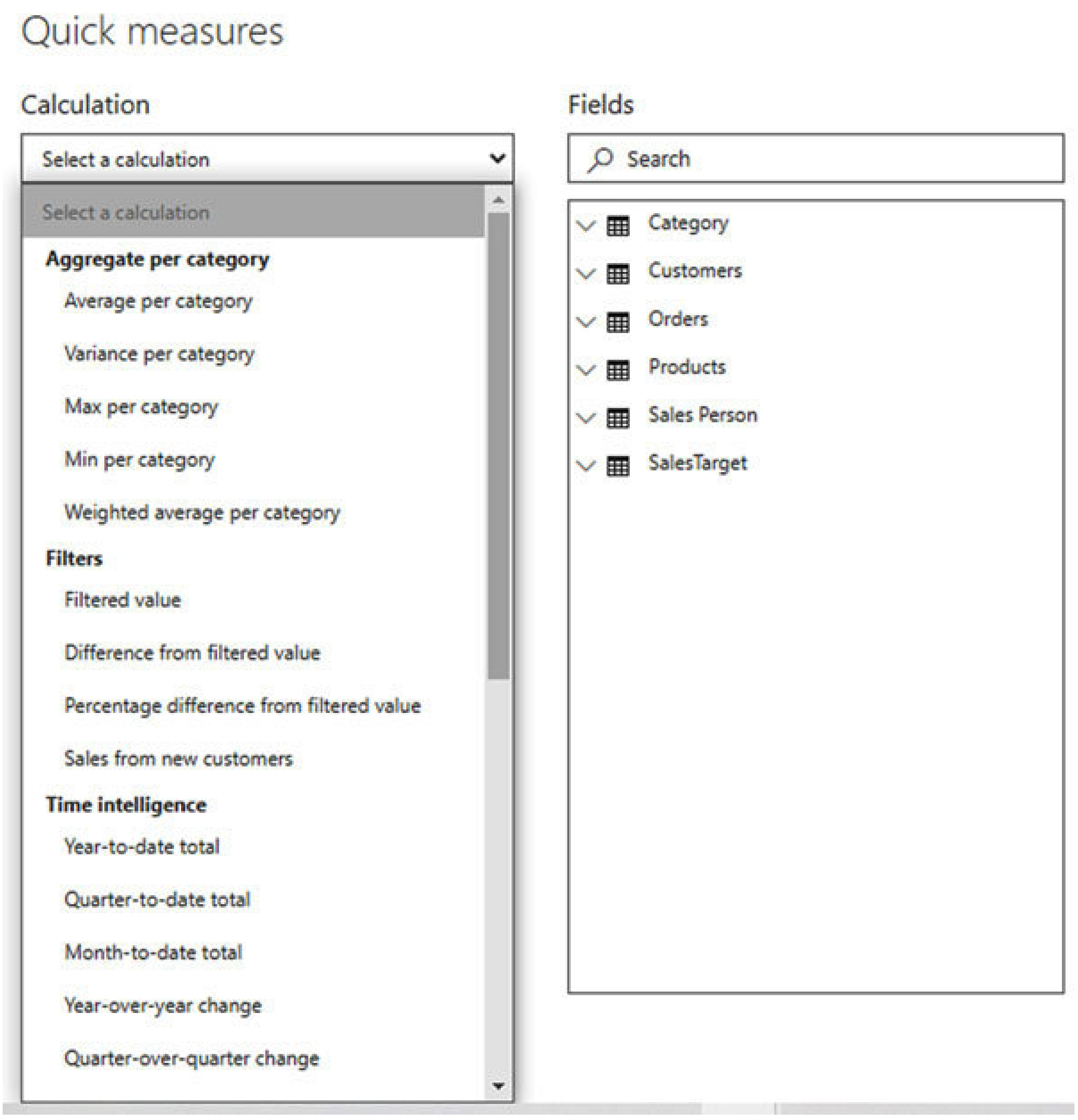


***Figure 4.7:*** *Quick measure categories*

Quick measures can also be created from the field list by right-clicking on any field or selecting the ellipsis next to any field in the **Fields** pane.

## Creating a Quick measure

From the Power BI desktop, select the table in which you want to create **New** and from the **Home** tab, select **Quick measure,** as shown in the following figure:

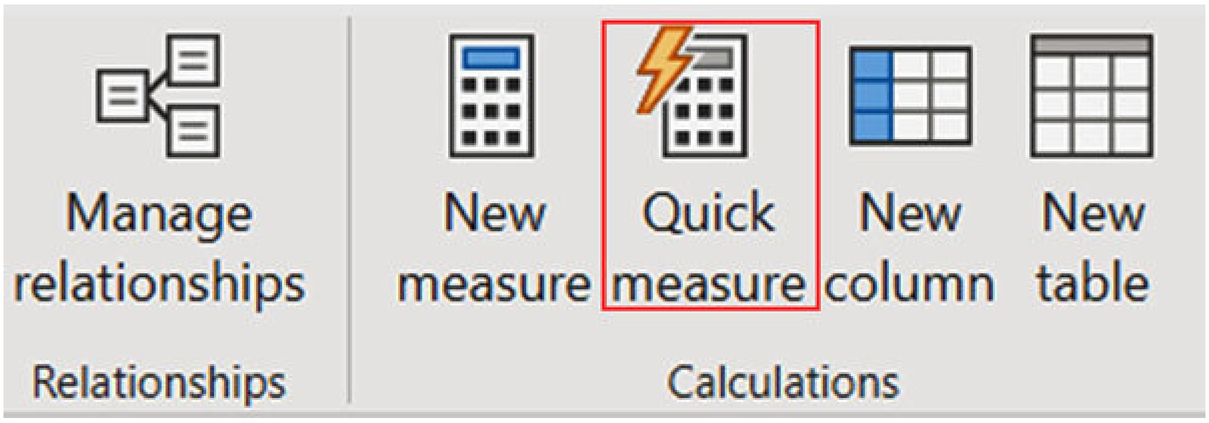


***Figure 4.8:*** *Creating Quick measure*

We can also create a **Quick Measure** from the field list or the visualization.

**Exercise 4:**

**Create a Quick Measure – Running Total of Sales**

In this exercise, we will create a **Quick Measure** to calculate the **Running total of** This calculation will be created on the **Orders** table.

The steps to create a Quick Measure for Running Total of Sales are as follows:

(Refer to the page on Quick Measure in the enclosed

On **Data** select **Orders** from the Fields. Click on the ellipsis next to the **Orders** table and select **New quick measure,** as shown in the following screenshot:

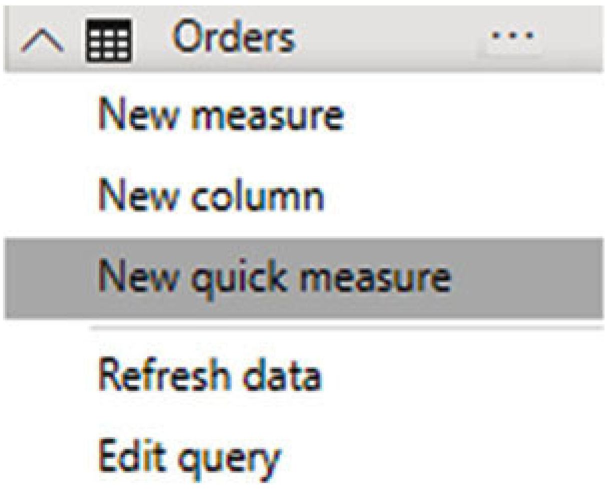
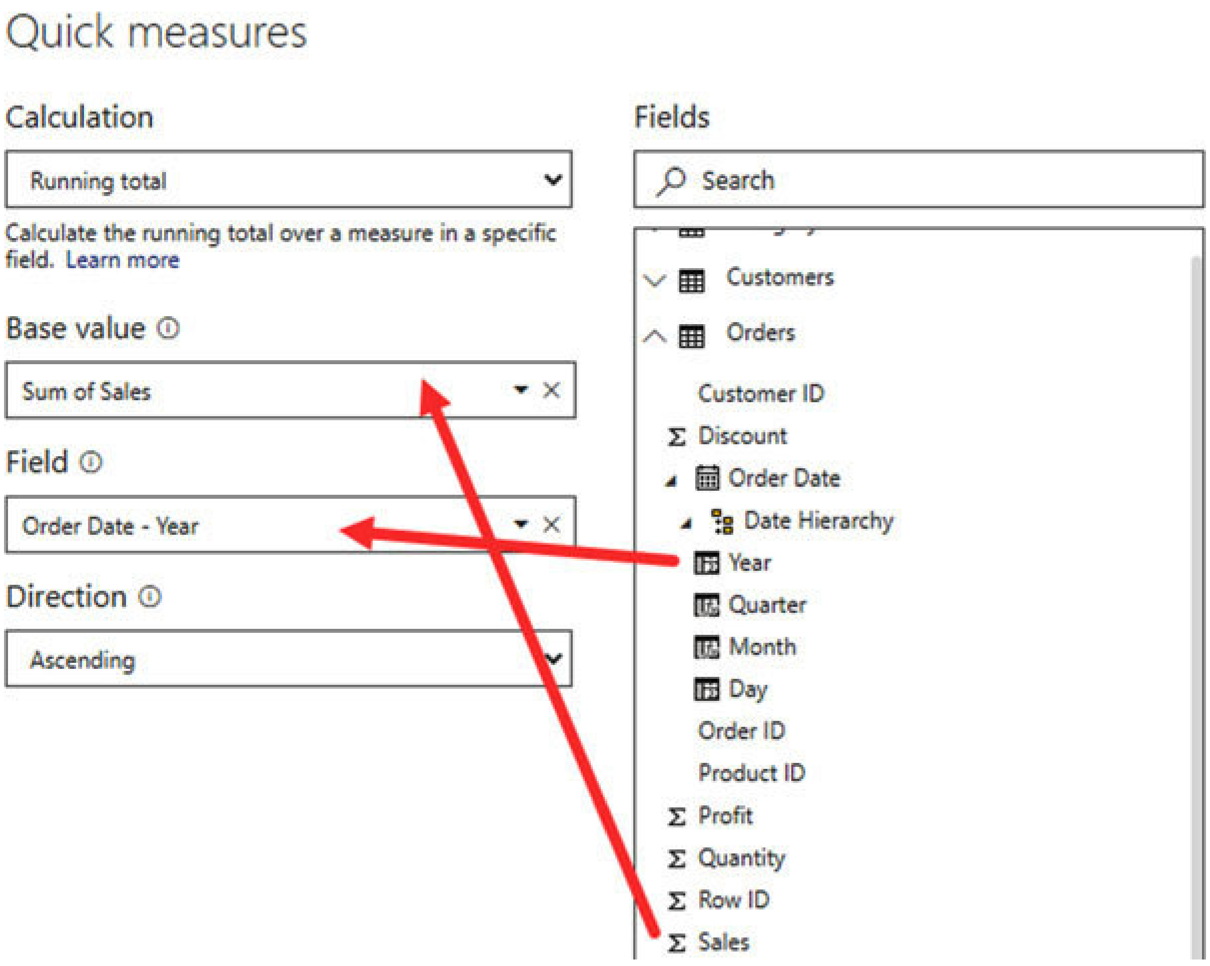


***Figure 4.9:*** *Selecting the option for New quick measure*

In the next Quick measures dialog box, select **Running total** from the **Select a calculation** drop down.

In the fields, select **Orders** and drag **Sales to the Base** value box and place year on to the field.


Quick measures

Calculation: Running total

Calculate the running total over a measure in a specific field. Learn more

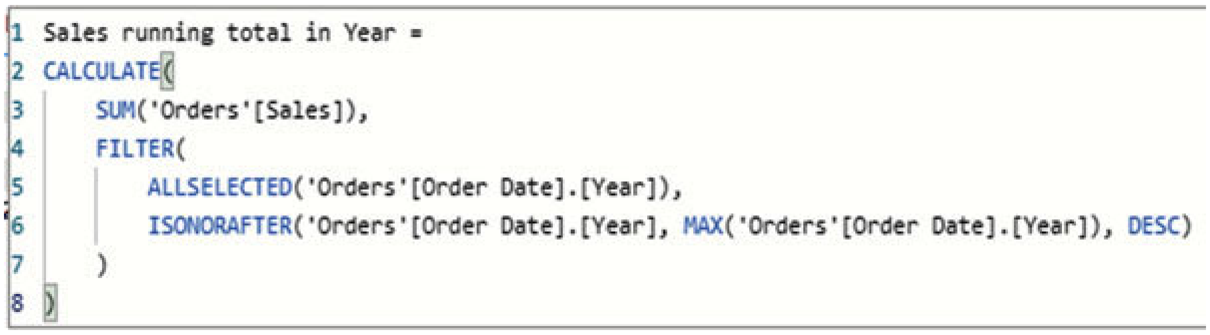
Base value: Sum of Sales

Field: Order Date - Year

Direction: Ascending

***Figure 4.10:** Creating Quick measure*

This will create a Quick measure **Sales running Total in Year** under We can also verify it in the Data model. Click on the measure, and see that Power BI has created a DAX formula. In the following screenshot, we can see how running total is calculated:



```
Sales running total in Year =
CALCULATE(
    SUM('Orders'[Sales]),
    FILTER(
        ALLSELECTED('Orders'[Order Date].[Year]),
        ISONORAFTER('Orders'[Order Date].[Year], MAX('Orders'[Order Date].[Year]), DESC)
    )
)
```

***Figure 4.11:*** *DAX formula using in Quick measure*

Quick measure will be available in any other visualization, if you want to use them. It is a good way to learn DAX too, as it shows you how the formula is created using DAX.

## ***Mathematical functions***

DAX has a range of mathematical functions and they are similar to the Excel mathematical functions. Some of these functions are etc.

These functions are also called the aggregate functions, as they aggregate the values of a column. The aggregate functions work on numeric values.

## *Sum*

This function aggregate/add all the numbers in a column. Only one column can be aggregated at a time, such as

The operations involving two columns, such as Sum(Orders[Quantity]) * Orders[Unit Price]) are not allowed.

**Syntax:**

Sum(Table [Column])

**Returns:** Scalar/single decimal value

**Exercise 5:**

**Calculate percentage (%) of Profit**

This calculation will use the **Profit** and **Sales** columns from the **Orders** table.

The steps to calculate the percentage (%) of Profit are as follows:

(Refer to the page on Sum in the enclosed **Chap4_Dax)**

Use the same application as earlier,

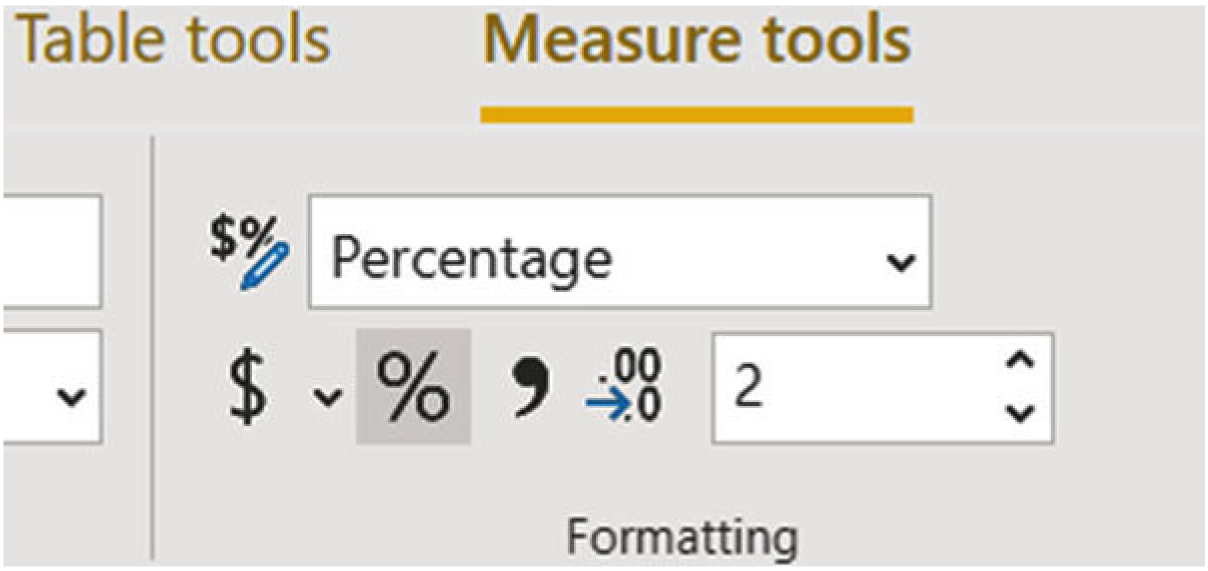
Select **Data** from and select

From the **Home** ribbon, select **New** Type the following expression in the DAX edit box:

% of Profit = Divide(Sum(Orders[Profit]), Sum(Orders[Sales]),0)

In the preceding calculation, % of profit is a **Measure** Divide is a function that divides **Orders[Profit]** by **Orders[Sales],** and returns 0, in case if divided by 0 or blank.

We will see a column % of profit created under the **Orders** table with a calculator icon. Select the % of **Profit** column, and from the ribbon, select **Measure** and change the formatting to percentage with two places of decimals, as shown in the following screenshot:

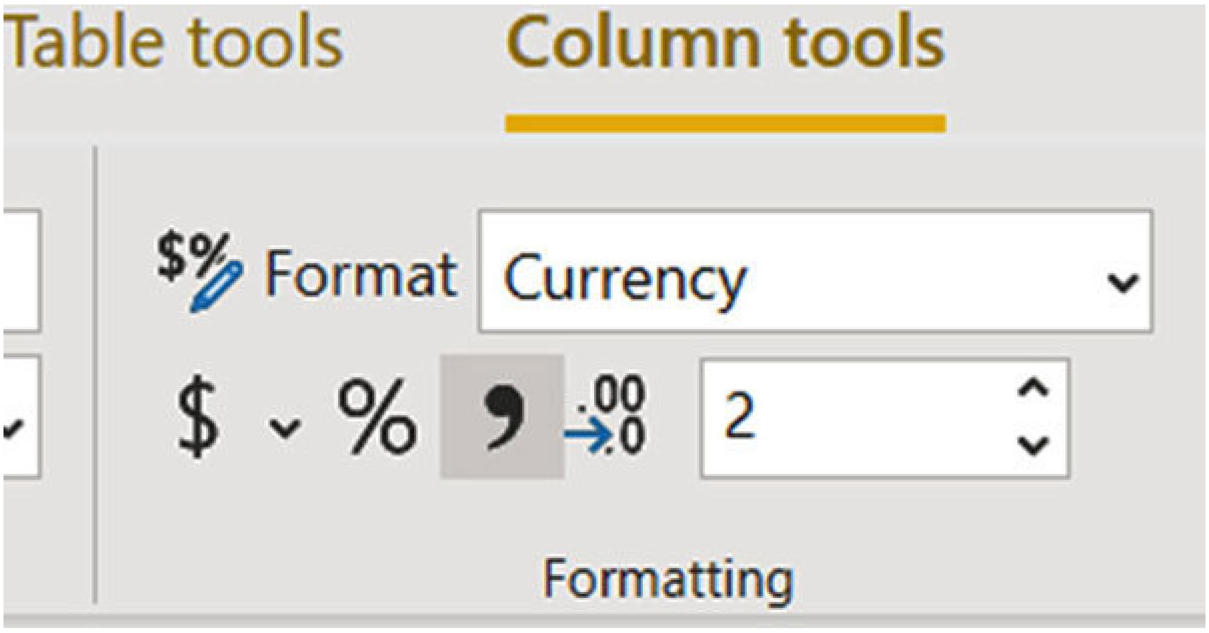


Percentage

2

***Figure 4.12:*** *Percentage formatting*

We can also change the formatting of the Profit and Sales fields. Under the Orders table, one by one, select Profit and Change the formatting for these columns to Currency and two places of decimals, as shown in the following screenshot:



Format: Currency

2

***Figure 4.13:*** *Formatting of columns*

## *SumX*

This function evaluates the sum of an expression for each row of the table. It performs row-by-row computation over the specified table. It can work on multiple columns, as follows:

SumX(Orders, Orders[Quantity] * Orders[Unit Price])

**Syntax:**

SumX(Table, )

Here, **Table** is the name of the table in the data model or an expression that returns a table.

Expression is a column that contains the numeric values to be summed or an expression that evaluates to a column.

**Returns:** Scalar decimal value.

## ***SumX verses Sum***

SumX is called an iterator function. It is similar to **Sum()** with certain distinctions, which are as follows:

Unlike **SumX** will return the sum of an expression computed for each row in a table.

It is called an iterator function as it works through the entire table row by row.

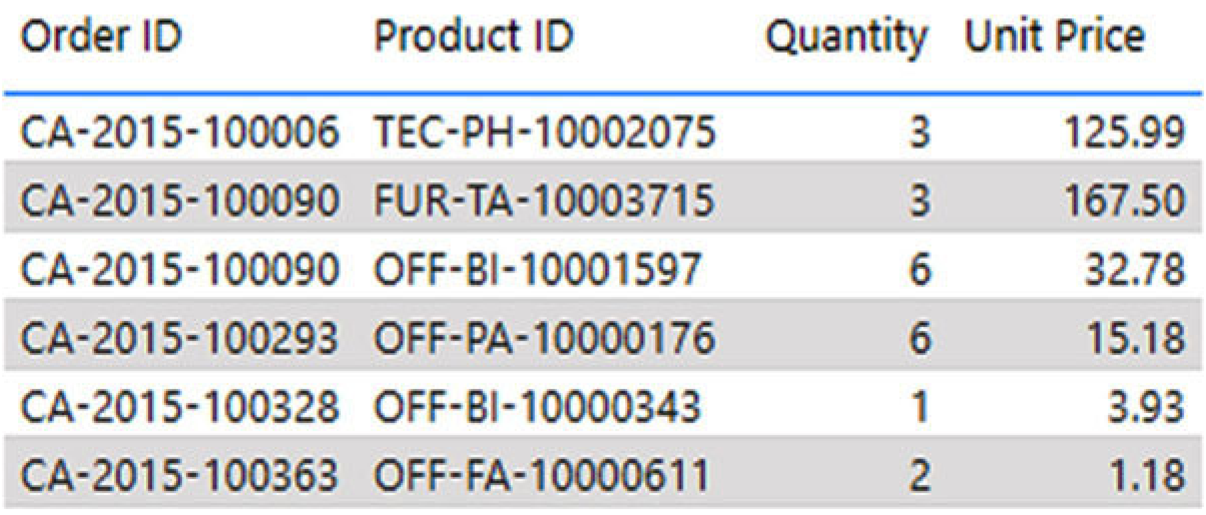
**Sum()** works on a single column but **SumX** can work on multiple columns in a table and performs row-by-row evaluation of those columns.

**Sum()** and **SumX()** can give the same or different results based on the context in which they are being used.

A calculated column performs row-level computation. Such computations are not possible in calculated measures using **Sum(),** but using we can create calculated measure and perform a row-level computation.

**When to use SumX?**

Use SumX when the table is in the format of the **Orders** table which lists **Order Product** and **Unit** as shown in the following figure:



| Order ID | Product ID | Quantity | Unit Price |
|---|---|---|---|
| CA-2015-100006 | TEC-PH-10002075 | 3 | 125.99 |
| CA-2015-100090 | FUR-TA-10003715 | 3 | 167.50 |
| CA-2015-100090 | OFF-BI-10001597 | 6 | 32.78 |
| CA-2015-100293 | OFF-PA-10000176 | 6 | 15.18 |
| CA-2015-100328 | OFF-BI-10000343 | 1 | 3.93 |
| CA-2015-100363 | OFF-FA-10000611 | 2 | 1.18 |

***Figure 4.14:*** *Data from Orders Table*

To get the **Total Sales for each Order** we need to multiply (for each row) **Quantity** with **Unit** In this case, we need to use **SumX** as we need to perform a row-by-row computation of the two columns.

**Exercise 6:**

**Calculate Total Sales for each Order ID**

For each row in the **Orders** table, we need to aggregate the multiplication of **Quantity** and **Unit**

The steps to calculate the Total Sales are as follows:

(Refer to the page on SumX in the enclosed

On the Power BI desktop, navigate to **Data** From select

With the Data view selected, from the **Home** tab, select **New Measure** and write the following expression in the DAX edit box:

```
TotalSalesX = SUMX(Orders, Orders[Quantity] * Orders[UnitPrice])
```

In the preceding expression, **TotalSalesX** is the name of the measure, **SumX** is the iterator function, **Orders** is the table on which the calculation will be performed, **Orders[Quantity]** and **Orders[Unit Price]** are the columns in the **Orders** table used in the expression.

This will create a column **TotalSalesX** under the **Orders** table.

## ***Count functions***

DAX provides different count functions which can be utilized in visualizations. Each count function works differently.

## *Count*

This function counts the number of non-blank rows of a column. It counts the rows containing values, such as and It skips the blank rows and cannot count the Boolean values.

**Syntax:**

Count(Table[Column])

**Returns:** Scalar whole number value

**Exercise 7:**

**Count the number of orders in Orders table**

The steps to count the number of orders are as follows:

(Refer to the page on Count in the enclosed

Follow the same steps that were given earlier; navigate to **Data** Select **Orders** from the fields.

Select **New Measure** from Write a calculation, as follows:

```
CountofOrders = Count(Orders[Order ID])
```

This will create a new measure in the **Orders** table.

## *CountA*

**CountA** is similar to the **Count** function. It counts the number of non-blank rows of a column. It counts the rows containing values such as and

The difference between **Count** and **CountA** is that **CountA** can count the **Boolean** values, whereas **Count** cannot.

**Syntax:**

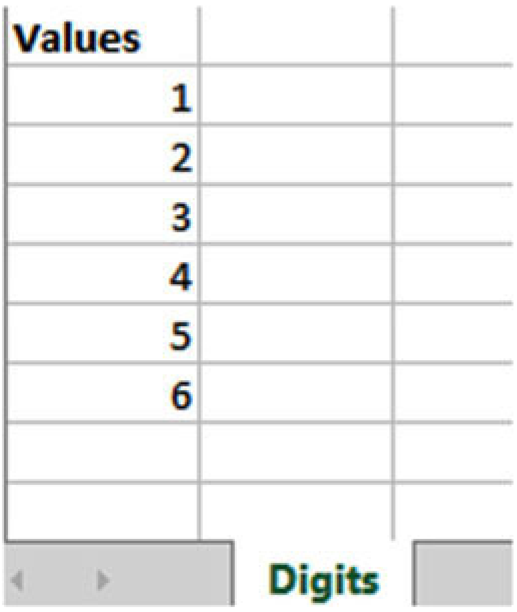
CountA(Table[Column])

**Returns:** Scalar whole number value.

**Exercise 8:**

**Using Count and CountA to check the difference between them**

To understand this concept, we will load a simple Excel sheet This XLS is available in the enclosed data files. This has a format, as shown in the following figure:



| Values |
|---|
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |

***Figure 4.15:*** *BooleanVales.xls*

The steps to load BooleanValue.xls are as follows:

Load the preceding Excel sheet in the Power BI desktop. Follow the steps mentioned in the previous chapter on how to load an Excel file.

This will create a table Digits with one column -

Now, we will create a **New BooleanValue** column in this table.

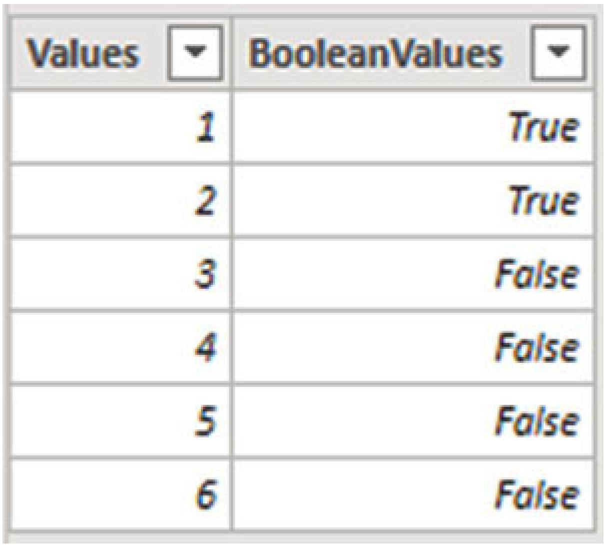
In the **Data** select **Digits from the**

From select **New** Create a simple DAX calculation to calculate (using the Boolean values of **True** and

BooleanValues = If(Digits[Values] = 1, TRUE(), If(Digits[Values] = 2, True(), If(Digits[Values] = 7, FALSE(), Blank())))

The preceding expression uses the **IF** conditional statement. It says, if the **Digits[Values]** = then set the new column as if **Digits[Values]** = then set the new column as but if **Digits[Values]** = then set the new column as false and leave the others as blank.

This will create a **BooleanValues** column in the **Digits** table with values **True** and Observe that only two values are **True** as 1, 2 values are present in the table, as shown in the following figure:



| Values | BooleanValues |
|---|---|
| 1 | True |
| 2 | True |
| 3 | False |
| 4 | False |
| 5 | False |
| 6 | False |

***Figure 4.16:*** *Table with calculated measure BooleanValues*

**Using Count and CountA**

Now we will count the values in the column, using the **Count** and **CountA** functions and see the results.

**The steps to using Count and Count A functions are as follows:**

(Refer to the page on in

With the **Digits** table selected under navigate to the **Home** tab and select **New**

Use the **Count** function as shown. This will create a **CountBoolean** column under **Digits** table:

CountBoolean = Count(Digits[BooleanValues])

Navigate to the **Home** again, and create another **New** In this expression, we will use as shown in the following screenshot:

CountBooleanA = CountA(Digits[BooleanValues])

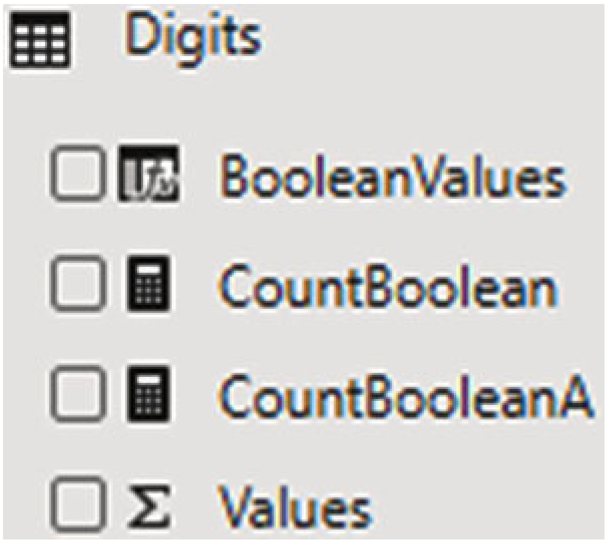
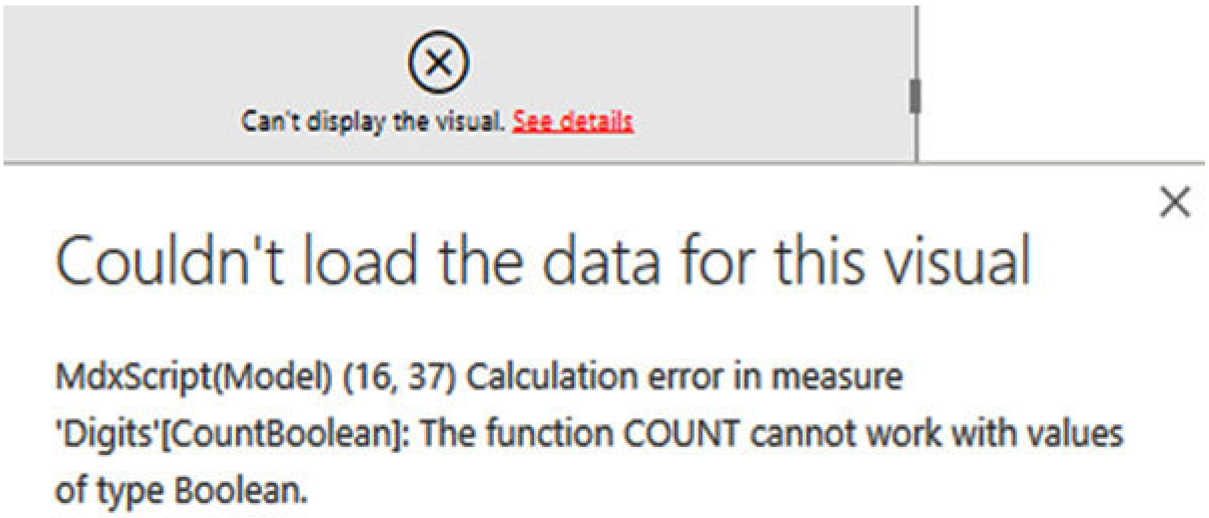


***Figure 4.17:*** *Columns in Digits table*

Now, verify the expression using

Navigate to the Report view and select a **Table** type, Visualizations. From the **Digits** table, select and Power BI will display an error, as shown in the following screenshot:



Can't display the visual. See details

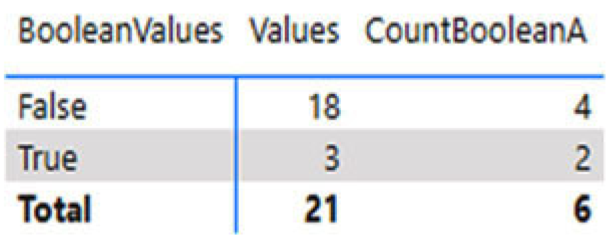
Couldn't load the data for this visual

MdxScript(Model) (16, 37) Calculation error in measure 'Digits'[CountBoolean]: The function COUNT cannot work with values of type Boolean.

***Figure 4.18:*** *Error when using Count on Boolean values*

Now, verifying the expression using Close the preceding error and remove the chart. Select the **Table** visualization again, and this time use and

You will get the correct values. This is because **CountA** can count the **Boolean** values, as shown in the following screenshot:



| BooleanValues | Values | CountBooleanA |
|---|---|---|
| False | 18 | 4 |
| True | 3 | 2 |
| **Total** | **21** | **6** |

***Figure 4.19:*** *Table visual with CountBooleanA*

## *CountX*

**CountX** is similar to the **CountA** function, but counts the non-blank values when calculating the result of an expression. It iterates the rows of the table and counts the rows where the input expression results in a non-blank output.

For example, if the table contains ten rows out of which four rows are blank, then using the **CountX** function, it will count up to six as, it will only count the non-blank rows.

**Syntax:**

CountX(Table, expression)

**Returns:** Scalar integer value.

## *CountAX*

**CountAX** is similar to the **CountX** function; it counts the non-blank results when calculating the result of an expression. It iterates the rows of the table and counts the rows where the input expression results in a non-blank output.

The difference between **CountAX** and **CountX** is that **CountAX** can operate on the Boolean datatype, whereas **CountX** cannot do that.

**Syntax:**

CountAX(Table, expression)

**Returns:** Scalar whole number value.

## *CountBlank*

**CountBlank** counts the number of blank values in a column. If the rows do not meet the criteria, blanks are returned.

**Syntax:**

CountBlank(Column)

**Returns:** Scalar whole number value.

## *CountRows*

The **CountRows** function counts the number of rows in a table.

**Syntax:**

CountRows(Table)

**Returns:** Scalar whole number value.

Power BI provides a variety of **Count** functions, which can be used in different scenarios.

## ***Information functions***

DAX contains a number of Information functions. These functions check the value or row that is passed as an argument and returns true/false or an alternate value if the value matches the expected type. In this section, we will look at some of these functions.

## ISERROR

This function checks if the value or expression results in an error.

ISERROR()

True, if the expression or the value is an error, else false.

**Exercise 9:**

**Create a calculation to divide SalesMeasure by CountofOrders**

If error occurs, return 0, or else, return the output of the expression.

On the **Orders** table, create a calculated column to divide **SalesMeasure** by **ISERROR** will check for the divide by 0 error. If the error occurs, it will return 0, otherwise it will return the output of the calculation.

The step to use the ISError function is as follows:

(Refer to the page on Information Functions, enclosed in

From the **Data** select the **Orders** table and create a **New Measure** using the following expression:

```
DivideByZero = if
(ISERROR(Orders[SalesMeasure]/Orders[CountofOrders]),0,
[SalesMeasure])
```

## USERNAME

This function returns the domain name and username of the logged in user. It can be used only with the calculated measure.

**Syntax:**

```
USERNAME()
```

User logged into the application.

**Exercise 10:**

**Identify the current user using the application**

On **Data** select the **Orders** table and select **New Measure** from the **Home** tab. Create a new measure with the following DAX expression:

```
CurrentUser = USERNAME()
```

This expression will return the current user using the application. We can test by using this expression in a table visualization.

## LOOKUPVALUE

The **LookUPValue** function is an important DAX Information function. It can be used for the Data model cleanup. One of the used cases is when we have a table with only two columns such as **ID** and and it can be merged to the main table.

**Syntax:**

LOOKUPVALUE(, , , )

**Returns:** The **result_columnName** for the row that meets the criteria specified by the **search_columnName** and search value.

If there is no match that satisfies the search values, a blank is returned.

**The** for the LookUpValue function is as follows:

The column name of a table that wished to be returned. The column name must be qualified with the table name and cannot be an expression.

The lookup column name in the same table as the **result_columnName** or in a related table. The column name must be qualified with the table name and cannot be an expression.

The column name or value used compare with
**search_columnName** to return

This is an optional result value, if no values are returned in If this is not specified, **BLANK** is returned.

We will use the **LookUpValue** function to load the columns from one table into the main table.

## ***LOOKUPVALUE – SalesPerson_Location and Sales Person Table***

The **SalesPerson_Location** table contains only two columns, **SalesPersonID** and It is in relationship with the **Sales Person** table. Based on the we can load the **Location** in the **Sales Person** table.

The steps to load Locations from SalesPerson_Location to the Sales Person table are as follows:

On the Power BI desktop, navigate to **Data** From **Fields** on the right, select the table. In this table, we will be creating a column for

From the **Home** tab, select **New Column** and complete the following steps:

In the formula bar, give the new column a name as **Location** and after the = symbol, type Power BI will display the syntax.

Enter **Result_ColumnName** as Pick the **Table.column** name from the drop down.

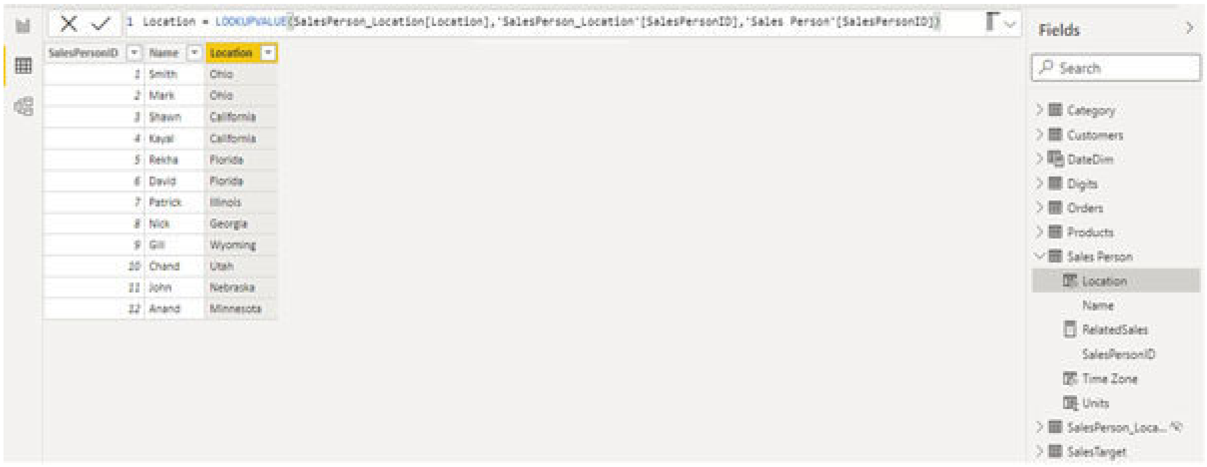
as **SalesPersonID** because we want **Location** based on that.

as **Sales Person[SalesPersonID].**

Press *Enter* after specifying the formula.

The complete formula will look like the following:

Location =
LOOKUPVALUE(SalesPerson_Location[Location],'SalesPerson_Location
'[SalesPersonID],'Sales Person'[SalesPersonID])



```
Location = LOOKUPVALUE(SalesPerson_Location[Location],'SalesPerson_Location'[SalesPersonID],'Sales Person'[SalesPersonID])
```

| SalesPersonID | Name | Location |
|---|---|---|
| 1 | Smith | Ohio |
| 2 | Mark | Ohio |
| 3 | Shawn | California |
| 4 | Kayal | California |
| 5 | Rekha | Florida |
| 6 | David | Florida |
| 7 | Patrick | Illinois |
| 8 | Nick | Georgia |
| 9 | Gill | Wyoming |
| 10 | Chand | Utah |
| 11 | John | Nebraska |
| 12 | Anand | Minnesota |

***Figure 4.20:*** *Creating Location column using LOOKUPVALUE*

This will create the **Location** column in the **Sales Person** table. We can verify it by navigating to the model view.

The preceding operation is performed using **LOOKUPVALUE** and implies the following:

Create the **Location** column in the **Sales Person** table, populate it with the Location column from the **SalesPerson_Location** table by matching the values of **SalesPersonID** in the **SalesPerson_Location** table and **SalesPersonID** in the **Sales Person** table.

**Hiding the redundant tables:**

Review the data model, notice that the **SalesPerson_Location** table is no longer required, as the field it contained is now loaded in the **Sales Person** table. This table is now redundant and can be hidden from the data model.

Right-click on the **SalesPerson_Location** table, you will see two options at the bottom, **Delete** from model and Hide in report view, as shown in the following figure:

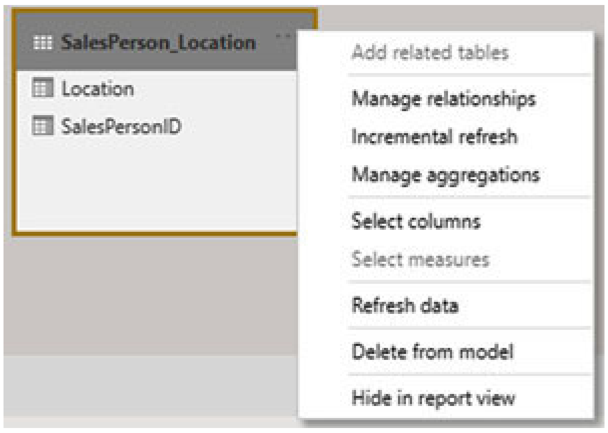


***Figure 4.21:*** *Delete and Hide options*

We can either hide the table or delete it from the model. If **Hide in report view** is selected, this table will be visible as greyed out in the Data view, but will not be visible in the Report view.

Select **Hide in report view** as this table will no longer be required; the result will be shown in the following screenshot:

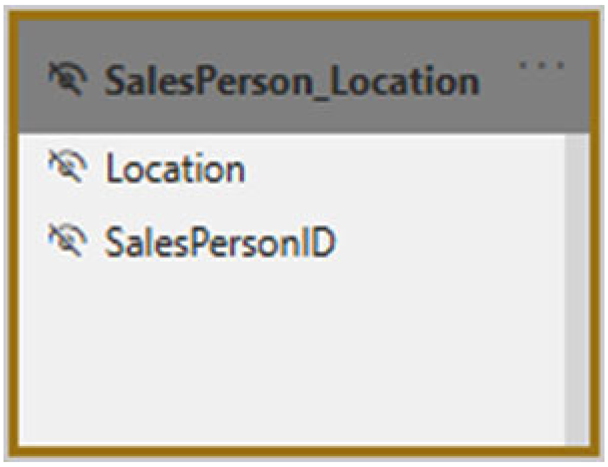


***Figure 4.22:*** *SalesPerson_Location hidden from the Report View*

The Lookup function can be used to reduce the number of tables in the report view.

## ***Logical functions***

DAX contains a variety of logical functions. These functions allow us to test if the condition in an expression evaluates as true or false. In this section, we will look at some of these functions.

## *If*

This function checks the condition in an expression. It returns one value if the condition is true and returns another value if the condition evaluates as false. For multiple conditions, you can use the nested

**Syntax:**

IF (, value>, value>)

Value of the expression.

**Exercise 11:**

**Create a Timezone column in the Sales Person table**

**Sales Persons** are located in different locations. We can create a new column in the table to show their time zones.

In this example, we will use only few time zones and will leave the others as blank.

We can store the following time zones in the new column:

Ohio, Florida in EST,
Illinois in CST,

Since we have multiple conditions, we can use the nested

The steps to create the TimeZone column in the Sales Person table are as follows:

In the Data view, select the **Sales Person** table, and from the **Home** tab, select **New**

Enter the following DAX expressions:

Time Zone = If('Sales Person'[Location] = "Ohio", "EST",If('Sales Person'[Location] = "Florida","EST", If('Sales Person'[Location] = "Illinois","CST")))

In the previous expression, the **If** condition states that if **Sales Person Location** is **Ohio,** then store **EST** as the value in the new column; if **Sales Person Location** is **Florida,** then store **EST** as the value in the new column; if **Sales Person Location** is **Illinois,** then store **CST** as the value in the new column.

After executing this expression, in the Data view, check the **Sales Person** table for the new column **Time**

## *And*

This function tests whether both the conditions are true. It returns True if both the conditions are true, or else it returns false.

**Syntax:**

AND (expression 1>,expression 2>)

**Returns:** True or false based on the tested condition.

**Exercise 12:**

**Find Good and Bad Orders by Sales**

Suppose, we want to check which **Orders** are good or bad, based on **Sales** and We want to label the **Orders** with **Sales** greater than 1000 and % of **Profit** greater than 10% as **Good** If this condition is not met, we want to label those orders as bad.

Since we are checking two conditions, we can use

The steps to Find Good and Bad Orders by Sales using AND are as follows:

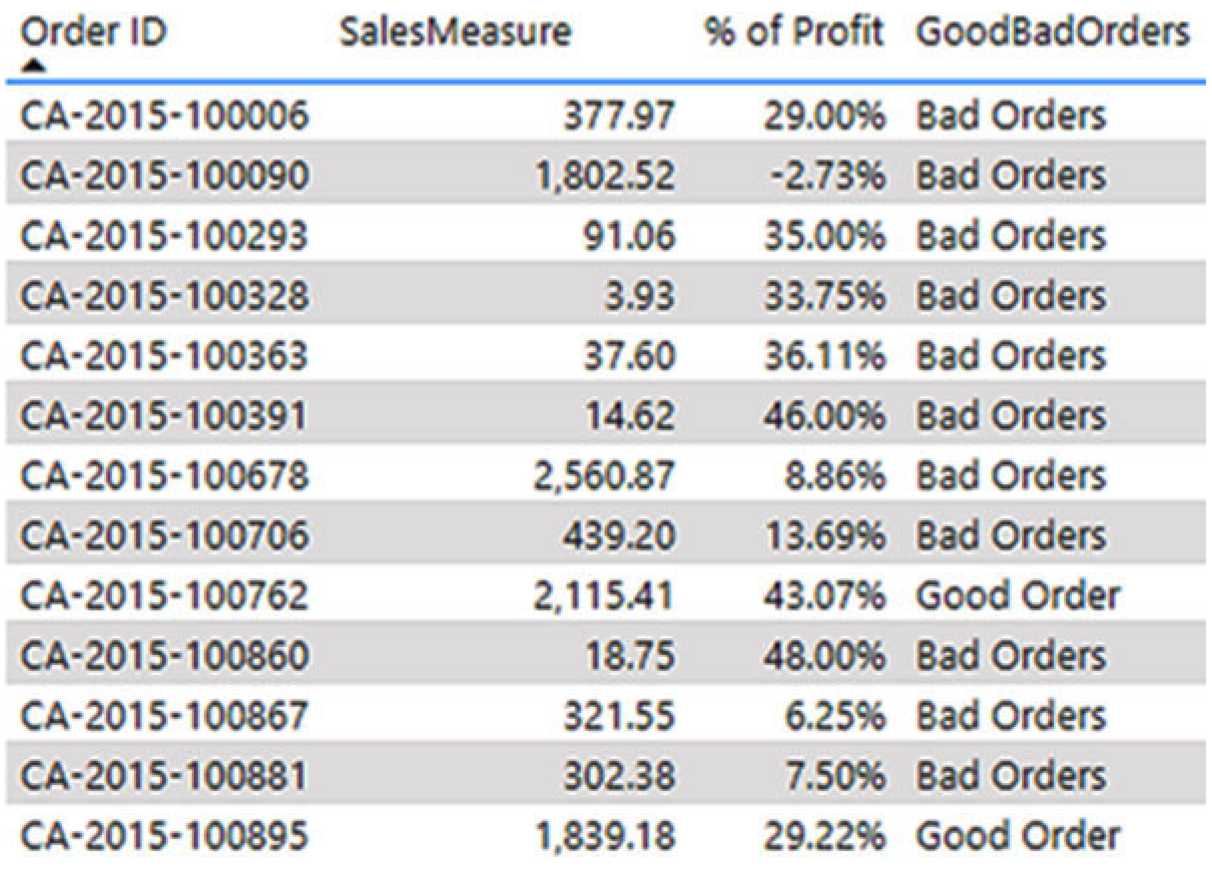
(Refer to the page on enclosed in

Create a **New Column** on the **Orders** table, and use the following DAX expression:

GoodBadOrders = If(AND(Orders[SalesMeasure] > 1000, Orders[% of Profit] >.1), "Good Order","Bad Orders")

This expression uses both **IF** and AND functions.

Create a table visualization, and from the **Orders** table, select **Order Sales % of** and the calculation we created earlier for **GoodBadOrders,** as shown in the following screenshot:



| Order ID | SalesMeasure | % of Profit | GoodBadOrders |
|---|---|---|---|
| CA-2015-100006 | 377.97 | 29.00% | Bad Orders |
| CA-2015-100090 | 1,802.52 | -2.73% | Bad Orders |
| CA-2015-100293 | 91.06 | 35.00% | Bad Orders |
| CA-2015-100328 | 3.93 | 33.75% | Bad Orders |
| CA-2015-100363 | 37.60 | 36.11% | Bad Orders |
| CA-2015-100391 | 14.62 | 46.00% | Bad Orders |
| CA-2015-100678 | 2,560.87 | 8.86% | Bad Orders |
| CA-2015-100706 | 439.20 | 13.69% | Bad Orders |
| CA-2015-100762 | 2,115.41 | 43.07% | Good Order |
| CA-2015-100860 | 18.75 | 48.00% | Bad Orders |
| CA-2015-100867 | 321.55 | 6.25% | Bad Orders |
| CA-2015-100881 | 302.38 | 7.50% | Bad Orders |
| CA-2015-100895 | 1,839.18 | 29.22% | Good Order |

***Figure 4.23:*** *Table visual with GoodBadOrders*

## *Switch*

Evaluates an expression and returns the options based on the evaluated value. It is similar to **IF** but **IF** only returns **True** or **False** unless it is a nested The **Switch** function removes the use of nested IFs and returns multiple values based on the expression.

**Syntax:**

SWITCH(, , 1>, , 2> ..., result>

**Returns:** A scalar value from the evaluated results.

**Exercise 13:**

**Create a New column for the month names from Order Date in the Orders table**

The steps to create the month names from Order Date are as follows:

(Refer to the page on **LogicalExpressions** in the enclosed

Select the **Orders** table, and create a **New Column** using the following DAX expression:

```
OrderMonth =
Switch(MONTH(Orders[Order Date]),1,"Jan",2,"Feb",3,"Mar",4,"Apr", 5,
"May", 6, "June", 7, "July", 8, "Aug", 9, "Sept", 10, "Oct", 11,
"Nov", 12, "Dec", "Invalid month number")
```

In the **Data** check the **Orders** table for the **OrderMonth** column.

Switch is similar to IFs but avoids the use of nested IFs

The other functions under this category are as follows:

This returns the first expression that evaluates to a value. If all expressions evaluate to a Blank, then Blank is returned.

This returns the logical value

This returns the logical value

This computes an expression and returns a mentioned value if the expression returns an error, otherwise returns the value of the expression.

This returns the logical opposite of the value computed by an expression. If the value in the expression returns the **NOT** function will return false and vice versa.

Tests if one of the logical expression is The function returns false if both the arguments are false.

The logical functions help is creating a new column based on conditions. They can be nested too, as we saw in the example using **IF** and

## ***Filter functions***

The filter functions in DAX are very powerful. These functions work on Table and relationships. There are a number of Filter functions that help in creating dynamic calculations.

Some of these functions are described in the following section.

## ***Filter***

The **Filter** function is also called a **Table** function as it returns a table based on the filter condition. The **Filter** function is used with the other functions that take a table as an input.

Filter is not used as a standalone, but as a function that is embedded in the other functions that require a table as an argument.

**Syntax:**

FILTER (Table, condition>)

Here, **Table** is the table to be filtered. It can also be an expression that outputs a table. Filter is a Boolean expression that filters/limits the rows of the table.

**Returns:** A table containing the filtered rows.

We will see an example of the filter function with the **Calculate** function in the following section.

## *Calculate*

It evaluates an expression in the context of the applied filters. It returns a single value of the expression modified by the filter. If the data has already been filtered, the function removes any existing filter and applies the filter provided in the filter expression.

**Syntax:**

CALCULATE (, expression 1>, expression 2>...)

Here, the first attribute is the one to be evaluated. It is basically a measure and will require an aggregate function. The second attribute, the **Filter** expression can be a Boolean expression or table expression that defines a filter. The function first evaluates the filter, then the expression.

**Returns:** The value as evaluated by the expression.

**Exercise 14:**

**Calculate Sales for the Year 2018**

This calculation will use the filter function too.

The steps to calculate the Sales for Year 2018 using the Calculate and Filter functions are as follows:

(Refer to the page on **Filter_Calculate_All** in the enclosed

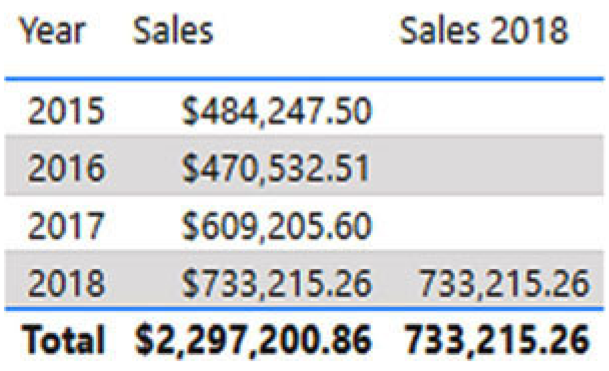
On **Data** select the **Orders** table and from the select **New**

Create a new measure with the following DAX expression:

Sales for 2018 = CALCULATE(Sum(Orders[Sales]),FILTER (Orders, Year(Orders[Order Date]) = 2018))

In the preceding expression, calculate is evaluating the expression based on the **Filter Order Date = 2018.**

To verify this function, select a **Table** From the **Orders** table, select **Sales** and **Sales for 2018**



| Year | Sales | Sales 2018 |
|---|---|---|
| 2015 | $484,247.50 | |
| 2016 | $470,532.51 | |
| 2017 | $609,205.60 | |
| 2018 | $733,215.26 | 733,215.26 |
| **Total** | **$2,297,200.86** | **733,215.26** |

***Figure 4.24:*** *Table depicting Sales 2018*

The preceding expression uses the **Filter** function to get **Year** =

## *CalculateTable*

The **CalculateTable** function is similar to **Calculate()** but returns a table. It determines a table expression modified by the given filters.

**Syntax:**

CALCULATETABLE(Table, filter 1, filter2...)

Here, **Table** is the table or the expression that returns a table. The **filter** expression can be a Boolean expression or table expression that defines a filter. It cannot use a nested calculate function or an aggregate function.

It can use any function that computes a single/scalar value. The function first evaluates the specified filter/s.

**Returns:** A table containing the values.

The **CalculateTable** function works exactly the same as the **Calculate** function but returns a table of values, that is, it will return multiple rows as compared to just one/scalar value as in the previous example using

**Exercise 15:**

**Calculate Sales for all the Years**

For this exercise, we have to use the **CalculateTable** function.

The steps to calculate Sales for all the Years are as follows:

(Refer to the page on **Filter_Calculate_All** in the enclosed

On **Data** select the **Orders** table and from the select **New**

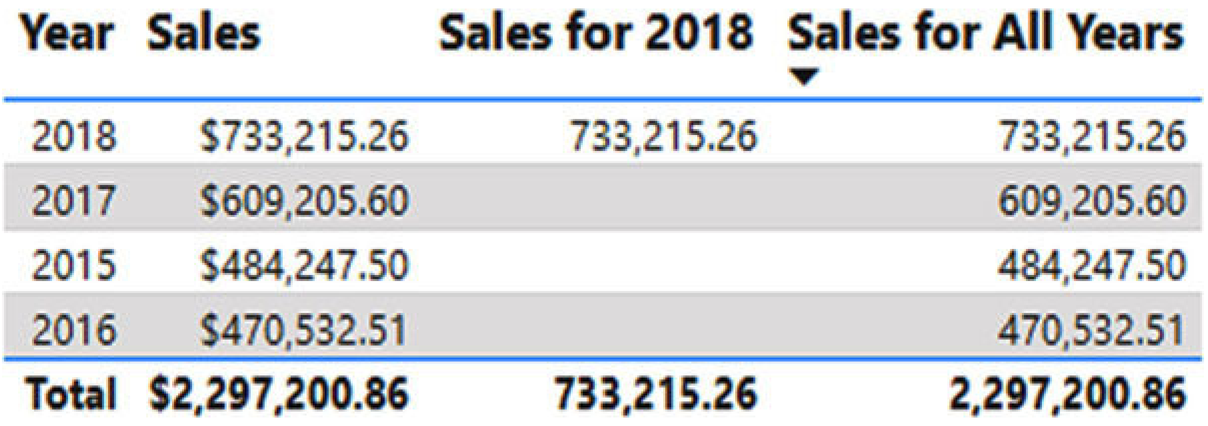
Create a new measure with the following DAX expression:

```
Sales for All Years =
SumX(CALCULATETABLE(Orders),Orders[Sales])
```

In the preceding expression, the **CalculateTable** measure is used because we want it to return multiple rows of **Sales** for all the years.

Put this column in the previous table that we created (using and see the difference.

Notice that the **CalculateTable** calculation returned multiple rows for each of the years.



| Year | Sales | Sales for 2018 | Sales for All Years |
|---|---|---|---|
| 2018 | $733,215.26 | 733,215.26 | 733,215.26 |
| 2017 | $609,205.60 | | 609,205.60 |
| 2015 | $484,247.50 | | 484,247.50 |
| 2016 | $470,532.51 | | 470,532.51 |
| **Total** | **$2,297,200.86** | **733,215.26** | **2,297,200.86** |

***Figure 4.25:*** *Calculate calculation result*

The difference between **Calculate** and **CalculateTable** is that one returns a Scalar value while another returns a table.

**Optional example:**

As another example of use the following expression:

Sales for Qty > 1 =
SumX(CALCULATETABLE(Orders,Orders[Quantity] > 1),Orders[Sales])

This expression will return **Sales for** where **Quantity** is greater than 1.

## ALL

The **ALL** function ignores any *applied filters* and returns all the rows in a table or all the values in a column. This function releases all the filters on a table or column and creates a calculation on all the rows in a table.

This function is used in conjunction with the other functions.

**Syntax:**

ALL(Table, column1,column2...)

Here, table is the one from where the filter needs to be removed. Similarly, the column is the column in the table from where the filters will be removed.

**Returns:** A table or column without any filters.

**Variations of All ()**

All functions can be applied in different ways, which are as follows:

This removes filters from the entire application.

This removes all the filters from the mentioned table.

**AllExcept(Table,** This removes all context filters in the specified table, except the filters that are applied to the specified columns.

**Exercise 16:**

**Calculate Sales of all the Orders in the dataset**

The steps to calculate Sales for all the Orders using ALL() are as follows:

(Refer to the page on **Filter_Calculate_All** in the enclosed

From **Data** select the **Orders** table and from select **New**

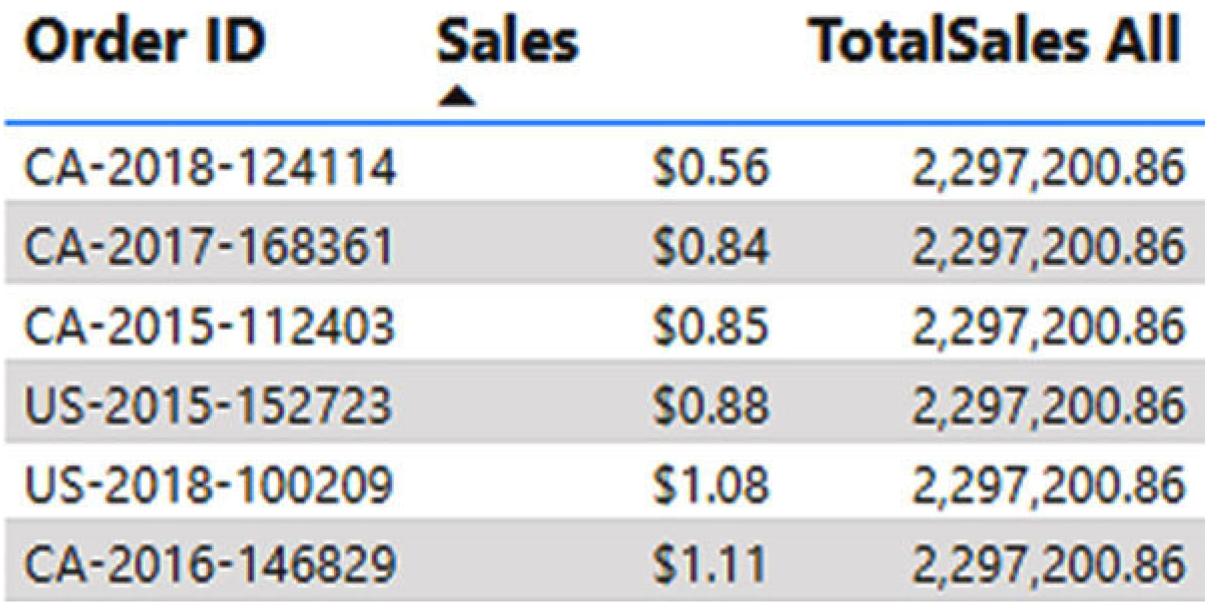
Create a new measure using the following DAX expression:

TotalSales All = CALCULATE(Sum(Orders[Sales]),All())

In this expression, **ALL()** removes all the filters if present in the visualization. It will always give the **TotalSales** of all the irrespective of the filters applied in the visualization.

To verify the preceding calculation, create a **Table**

Select **Order** and We will get different values for **Sales** but **TotalSales All** will return the same value as it is calculating the Sales for the entire dataset. The different order IDs have no effect on it, as shown in the following screenshot:



| Order ID | Sales ▲ | TotalSales All |
|---|---|---|
| CA-2018-124114 | $0.56 | 2,297,200.86 |
| CA-2017-168361 | $0.84 | 2,297,200.86 |
| CA-2015-112403 | $0.85 | 2,297,200.86 |
| US-2015-152723 | $0.88 | 2,297,200.86 |
| US-2018-100209 | $1.08 | 2,297,200.86 |
| CA-2016-146829 | $1.11 | 2,297,200.86 |

***Figure 4.26:*** *Using ALL*

## *Related*

The related function returns a column value from a related table. It is used to fetch the column that is not present in the current table but in another related table. It is similar to VLookup in Excel.

The Related function uses two tables, the current table and the second table in relationship with the current table. The function follows an existing many-to-one relationship between the tables. It requires row context, and therefore can be used with the calculated column or the **SumX** functions.

**Syntax:**

RELATED()

Here, the column is the column in a table whose value you want to retrieve.

**Returns:** A value that is related to the current row.

**Exercise 17:**

**Divide Sales in SalesTarget table with Units in SalesPerson table**

To perform this operation, the **Related** function will be used between the two related tables – **Sales Person** and **Sales**

For this example, we will first create a new column **Units** in the **Sales Person** table for each of the It will contain the Unit values for each For example,. If **SalesPersonID** = then **Unit = 100** and so on.

The steps to create Units column in Sales Person are as follows:

(Refer to the page on **Related** in the enclosed

On **Data** select the **Sales Person** table and from the select **New**

Create a new column using the following DAX expression:

```
Units = If('Sales Person'[SalesPersonID] = 1,100,If('Sales
Person'[SalesPersonID] = 2,200, If('Sales Person'[SalesPersonID] =
3,300,If('Sales Person'[SalesPersonID]=4,400,If('Sales
Person'[SalesPersonID]=5,500,If('Sales
Person'[SalesPersonID]=6,600,If('Sales
Person'[SalesPersonID]=7,700,If('Sales
Person'[SalesPersonID]=8,800,If('Sales
Person'[SalesPersonID]=9,900,If('Sales
Person'[SalesPersonID]=10,1000,If('Sales
Person'[SalesPersonID]=11,1100,If('Sales
Person'[SalesPersonID]=12,1200))))))))))))
```

This will create a new **Units** column in the **Sales Person** table.

**Using Related:**

Now we will create a **New Measure** in the **Sales Person** table. This measure will divide **Sales** in the **SalesTarget** table with **Units** in the **Sales Person** table.

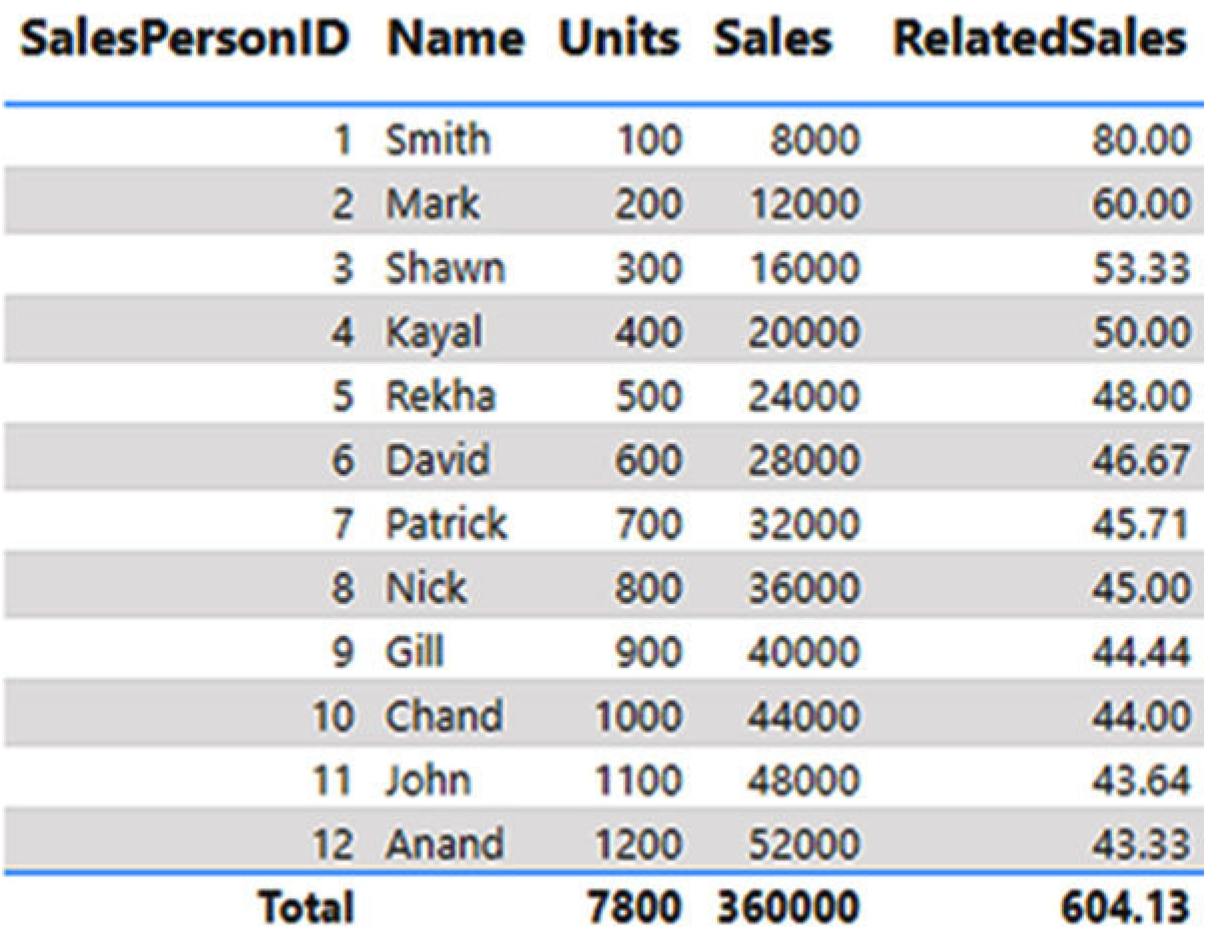
**Use the following DAX calculation:**

RelatedSales = SumX(SalesTarget,SalesTarget[Sales]/RELATED('Sales Person'[Units]))

Since the **Sales Person** and **SalesTarget** tables are in a *Many-to-One* relationship, we can use the **Related** function to get the values from these two tables.

In the previous expression, **SumX()** is used as row-by-row iteration of the table was needed.

To verify the preceding calculation, create a table visualization. From the **Sales Person** table, select and **Related** From the **Sales Target** table, select as shown in the following screenshot:



| SalesPersonID | Name | Units | Sales | RelatedSales |
|---|---|---|---|---|
| 1 | Smith | 100 | 8000 | 80.00 |
| 2 | Mark | 200 | 12000 | 60.00 |
| 3 | Shawn | 300 | 16000 | 53.33 |
| 4 | Kayal | 400 | 20000 | 50.00 |
| 5 | Rekha | 500 | 24000 | 48.00 |
| 6 | David | 600 | 28000 | 46.67 |
| 7 | Patrick | 700 | 32000 | 45.71 |
| 8 | Nick | 800 | 36000 | 45.00 |
| 9 | Gill | 900 | 40000 | 44.44 |
| 10 | Chand | 1000 | 44000 | 44.00 |
| 11 | John | 1100 | 48000 | 43.64 |
| 12 | Anand | 1200 | 52000 | 43.33 |
| **Total** | | **7800** | **360000** | **604.13** |

***Figure 4.27:*** *Output of Related function*

The Related function allowed the use of columns from the two related tables.

## ***RelatedTable***

This function evaluates to a table filtered to include the rows as per the current related table. It works in a *One-to-Many* direction of the relationships between the table.

The difference between **Related** and **RelatedTable** is that **Related** works in *Many-to-One* direction and **RelatedTable** works in *One-to-Many* direction.

**Syntax:**

RELATEDTABLE(Table)

A table of values.

## *Values*

The Value function accepts a column or a table as an input. When a column is passed, it returns a one-column table that contains the unique values of the specified column. A blank value is added to the values.

When a table is passed as an input, it returns the rows of the table. The returned rows can have duplicates and blanks.

It is used as an intermediate function or nested in a formula, to get a list of distinct values.

**Syntax:**

Values(or Column>)

**Returns:** A single column table when the input parameter is a column.

When the input parameter is a table name, a table of the same column is returned.

**Exercise 18:**

**Find the unique Customers in the Orders table**

The **Customer** table stores all the customers; there may be customers who have never made any purchase. The **Orders** table will contain the **Customers** who made a purchase. Some customers may have made multiple purchases; using the **Values** function, we can find the unique **Customers** in the **Orders** table.

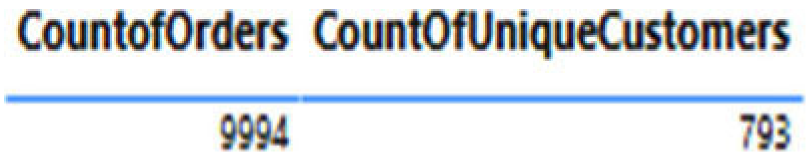
**The steps to find the unique Customers in the Orders table are as follows:**

(Refer to the page on **Values** in the enclosed

Select the **Orders** table and create **New Measure** using the following expression:

CountOfUniqueCustomers =
COUNTROWS(VALUES(Orders[Customer ID]))

Create a table visualization, from the **Orders** table, select **CountofOrders** (we have created this calculation earlier using and as shown in the following screenshot:



| CountofOrders | CountOfUniqueCustomers |
|---|---|
| 9994 | 793 |

***Figure 4.28:*** *Comparing Count and Values functions*

We can see that though the count of orders is the unique customers are only This means that some customers have made more than one order.

## Date and time functions

These functions are used in creating calculations based on dates and time. Some of these functions are described in the following sections.

## *Calendar*

This function returns a table with a single column named It creates a table with a continuous set of dates. The range of dates is between the start date and the specified end date, inclusive of those two dates.

**The advantage of the Calendar function is as follows:**

This function is useful when we want to create a separate **DateDim** table in the data model from the existing date field in the fact table.

The fact table will contain only the dates when a transaction occurs, but creating **DateDim** using **Calendar()** will give a table all the continuous dates between the specified **Min** and **Max** dates.

**Syntax:**

CALENDAR(,)

A table with a single column named

**Exercise 19:**

**Create a DateDim table in the data model using Order Date from the Orders table**

The steps to create a DateDim table are as follows:

(Refer to the page on **Date** in the enclosed

Select **Data View** and from the **Home** ribbon, select **New**

Use the following DAX expression:

```
DateDim = CALENDAR(Min(Orders[Order
Date]),Max(Orders[Order Date]))
```

In the preceding expression, **Start_Date** and **End_Date** are specified using the **Min** and **Max** of **Order**

This will create a **DateDim** table in the data model. It will have just one field Notice that Power BI automatically creates a Date hierarchy with the following columns, as shown in the following screenshot:

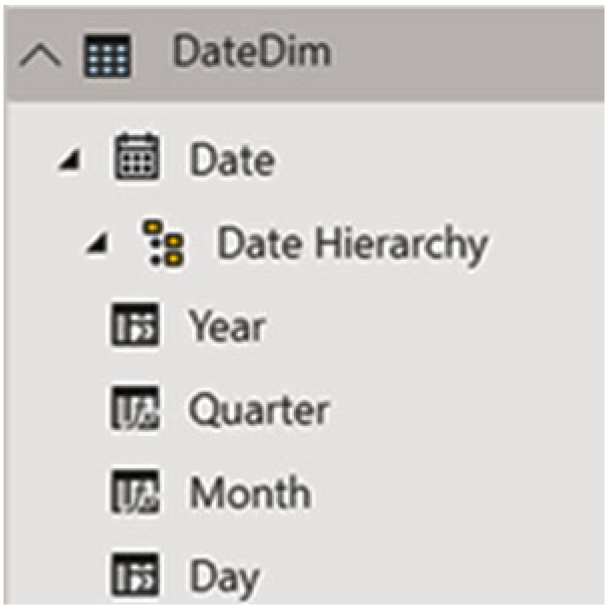


***Figure 4.29:*** *Date Hierarchy*

In the **Report** create the table visualization and select fields from

The fields under the **Date** hierarchy can be used in visualization as any other fields.

## *DateDiff*

The **DateDiff** functions return the difference between the two specified dates.

**Syntax:**

DATEDIFF(, , )

Here, **Start_Date** and **End_Date** can be any two dates and the interval is the format in which the difference of the dates is returned. It has the values such as, Second, Minute, Hour, Day, Week, Month, Quarter, and Year.

**Returns:** The count of the interval between the two dates.

**Exercise 20:**

**Find the time taken to ship the Orders**

The steps to calculate the time taken to ship the Orders are as follows:

(Refer to the page on **Date** in the enclosed

From **Data** select and from the select **New**

Use the following DAX expression to find the difference between **ShipDate** and

```
DaysToShip = DATEDIFF(Orders[Order Date],Orders[Ship
Date],DAY)
```

The previous expression returns the difference between **OrderDate** and **ShipDate** in Days as specified in the interval argument.

In the Report view, select the table visualization and from the **Orders** table, select **Order ID** and

## ***Variables in DAX expressions***

The use of variables in an expression can make the expression more efficient. The variables can be used in the nested functions where a calculation is reused. A variable helps in the following:

Performance

Readability

Debugging

The variables help in understanding a complex calculation.

## ***VAR statement***

The variables are implemented using the **VAR** statement. **VAR** declares a variable which stores the result of an expression. This variable can then be passed as an argument to the other measures.

**Syntax:**

VAR =
Return

**VAR** is the statement declaring the variable An expression is computed and stored in the named The Return statement returns the result of the expression which uses the named variable.

A named variable which holds the result of an expression

The points to remember when using **VAR** are as follows:

An expression can contain other **VAR** declarations.

The measures cannot refer to the variables defined outside the measure expression, but can refer to variables defined within the expression.

The columns cannot be referenced via the **TableName[ColumnName]** syntax.

The variables are useful in writing complex DAX expressions.

**Exercise 21:**

**Calculate of Sales for 2018 using variable**

The following calculation was mentioned earlier in the **Calculate** function:

Sales for 2018 = CALCULATE(Sum(Orders[Sales]), FILTER(Orders,Year(Orders[Order Date]) = 2018))

The steps to Calculate Sales for 2018 using Variable are as follows:

(Refer to the page on Variable in the enclosed

Select the **Orders** table and from the select **New**

Write the following DAX expression to define and use variables:

```
VarSalesAmt =
VAR FilterSales2018 = FILTER (Orders, Year (Orders [Order
Date]) = 2018)
Return
CALCULATE(Sum(Orders[Sales]),FilterSales2018)
```

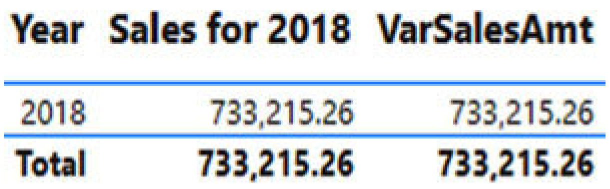
In the preceding calculation, a variable named **FilterSales2018** is defined as follows:

```
VAR FilterSales2018 = FILTER (Orders, Year (Orders [Order
Date]) = 2018)
```

It is then used in the **Calculate** function.

This will create the **VarSalesAmt** measure.

In the table visualization, use Year and both the calculations, **Sales for 2018** and They both will display the same result, as shown in the following screenshot:



| Year | Sales for 2018 | VarSalesAmt |
|---|---|---|
| 2018 | 733,215.26 | 733,215.26 |
| **Total** | **733,215.26** | **733,215.26** |

***Figure 4.30:*** *Calculation using Var*

The variables can be very useful when complex calculations are created.

## *Conclusion*

In this chapter, we learned how to use DAX to extend the functionality of the Power BI application. The DAX formulas are useful in creating new columns and measures. We also learned the use of various aggregations. These aggregations are useful in providing summarized values to the business users.

This chapter provided a direction in using DAX; as you use more of these functions in your applications, you will become more efficient. It takes practice and a deep understanding to become an expert in DAX.

In the next chapter, we will learn how to create the different types of visualizations. These visualizations will be the bar, line, pie charts, and more. The users only see the visualizations, and so whatever we have done so far will help in creating meaningful visuals.

## ***Questions***

What is the key difference between a Calculated column and a Calculated measure?

What is the difference between **Sum** and

To count the Boolean values, which function is more suitable, **Count** or

Which function is used to remove all the filters?

Which function is used to get the unique values of a column in a table?

Why are the Calculated columns not stored in the Measure table?

Given the start and the end dates, which function is used to create continuous dates?

## *Answers*

Calculate column is a physical field in the table and is part of the data model. Calculated measure is not a field in the table and is calculated on the fly.

**Sum** aggregates all the values of a column. It does not perform a row-by-row computation. **SumX** is called as an iterator function and performs a row-by-row aggregation.

**CountX.**

**All().**

**Values.**

Calculated columns are part of the table in the data model. It is a physical field in the table, and hence depends on the table where it is created. Calculated measure, on the other hand, is not part of the table and can be created anywhere.

Calendar.

CHAPTER 5

# Visualizations in Power BI

## ***Introduction***

In the previous chapter, we got an in-depth knowledge of the DAX functions and understood how they can be used. In this chapter, we will learn how to create the different visualizations in Power BI. The data model that we created in the previous chapters will be used while creating visualizations.

The knowledge of creating visualization is important because these visualizations will be consumed by the business users to understand the data.

## ***Structure***

In this chapter, we will discuss the following topics:

Understanding Visualizations

Introduction to Power BI reports

Creating multi-page report using visualizations

Filters

Creating visualizations

Formatting

Natural Language Q&A

Drill through reports

## ***Objectives***

The objective of this chapter is to understand the concepts of visualizations in a Business Intelligence application. We understand and create different kinds of visualizations to get data insights. We will also learn how to use variables to create complex calculations.

## *Review the data model*

This chapter will use the enclosed In the data model, tables, such as **Digits** and are removed as they were used only for example purposes. Before creating any visualization application, it is a good practice to get familiar with the underlying data model, as shown in the following screenshot:

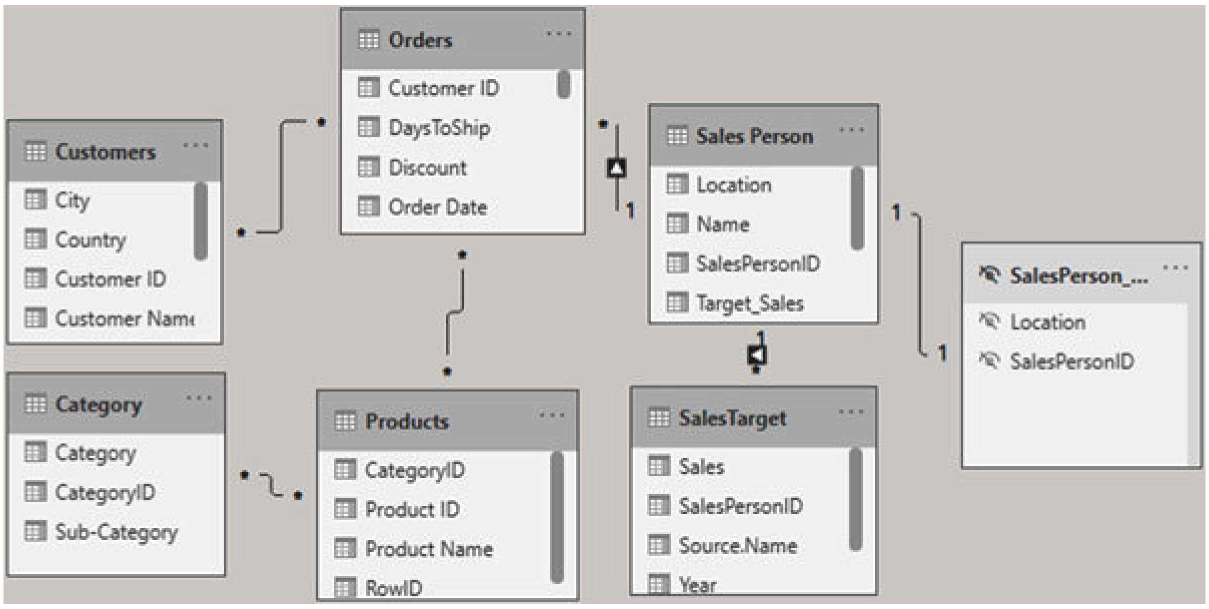


***Figure 5.1:*** *Data model to be used for Visualizations*

## *Understanding visualization*

Visualization or data visualization is a visual representation of data that helps in the better understanding of data. It is a pictorial depiction of the data in terms of charts, tables, and maps which provide insights into the data and enables the business users to make actionable decisions.

A huge amount of information can be communicated efficiently by using visualizations. It shows the relationship between one or more data elements and defines the trends and patterns. Since visualization depicts the data, it is important that the source data is clean and free of any anomalies. We performed the data loading and transformation exercises in the previous chapters.

The examples of data visualization are the bar chart, line chart, stack chart, etc. Visualizations contain dimensions or measures. In most of the cases, it will contain both the dimensions and the measures.

## ***Introduction to Power BI reports***

The report view in the Power BI desktop is used to create Reports. Reports contain one or more visualizations. These reports can be of a single page or multi-pages. The visualizations in a report are interactive and respond to filters and underlying data changes. In a report, we can move around these visualizations or copy and paste to create the other similar ones.

Sometimes reports are referred to as dashboards and vice versa. The difference between the two are as follows:

Reports can be of multiple pages, but Dashboards are of a single page.

Reports are created using the Power BI desktop and Power Service. Dashboards are created only in Power Service.

A Dashboard is created by pinning visualizations from one or more Reports.

Reports can have filters and responds to different kind of filters. A Dashboard does not contain filters.

If the underlying data changes, the report reflects the change. If the Report sourcing the Dashboard changes, the Dashboard does not change automatically.

Both the Reports and the Dashboards present the data visually. Dashboard usually contains a summarized, single page data. Reports can be detailed and multi-page.

## *How to create a Report?*

The following figure is the depiction of the various steps required to create a visualization in a report. Click on the Report view on the left to invoke the Report Editor:

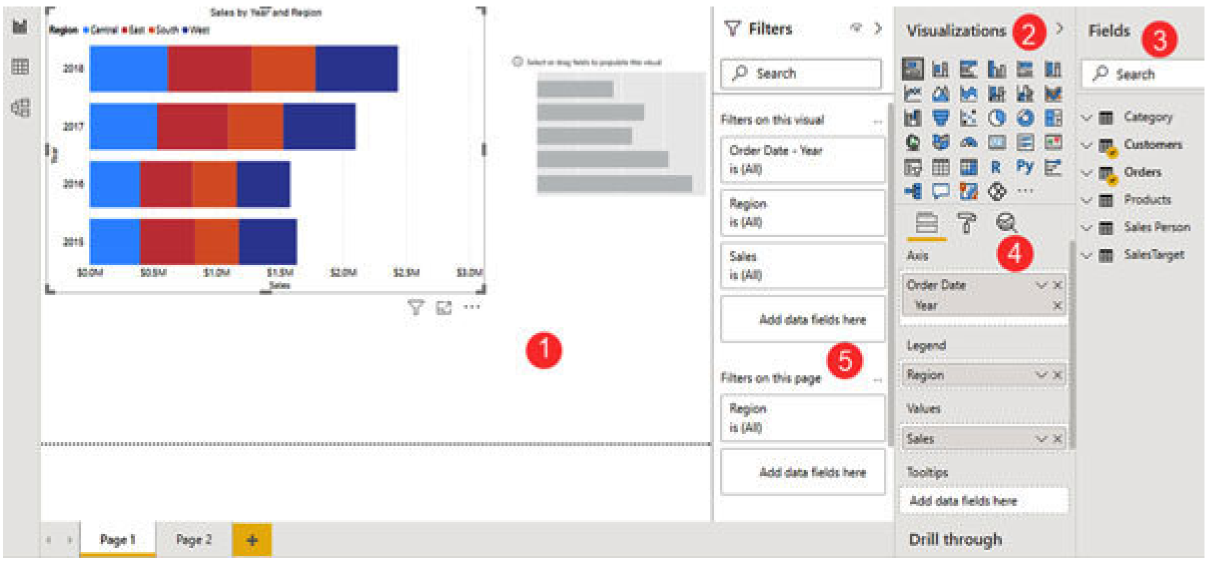


***Figure 5.2:*** *Report development interface*

The following are the descriptions for the marked numbers in the preceding figure:

**#1 Report canvas, here the visualizations are created and displayed:**

When a visualization is selected (from a placeholder of it is created on the canvas. At the bottom of the screen, the page tab **Page2** show the pages of the report. A single page report can have one of more interactive visualizations and a report can be of multiple pages. Select the visualization to see the options in the filters, fields, etc.

We can also copy and paste a visual by right-clicking on it and selecting **Copy | Copy visual,** as shown in the following figure:

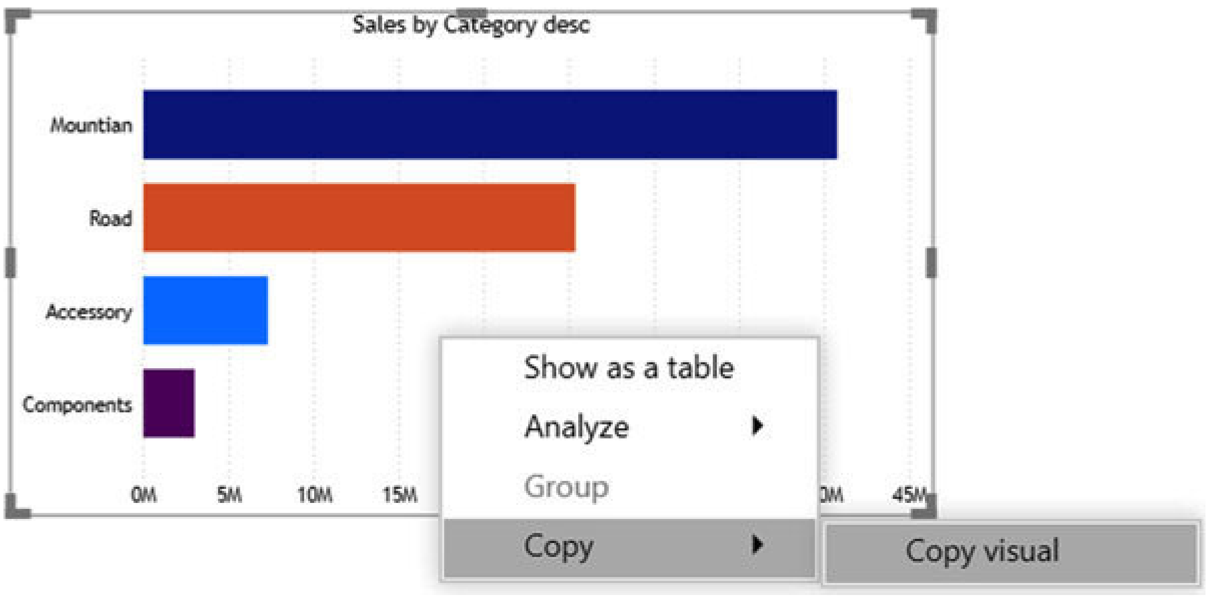


***Figure 5.3:*** *Copying a visual*

**#2** Depending on the business requirements, pick the visualizations from the list, as shown in the following screenshot: To create a visualization, select one and it will be displayed on the report canvas as a place holder. Hovering the mouse over a visual, displays its name:

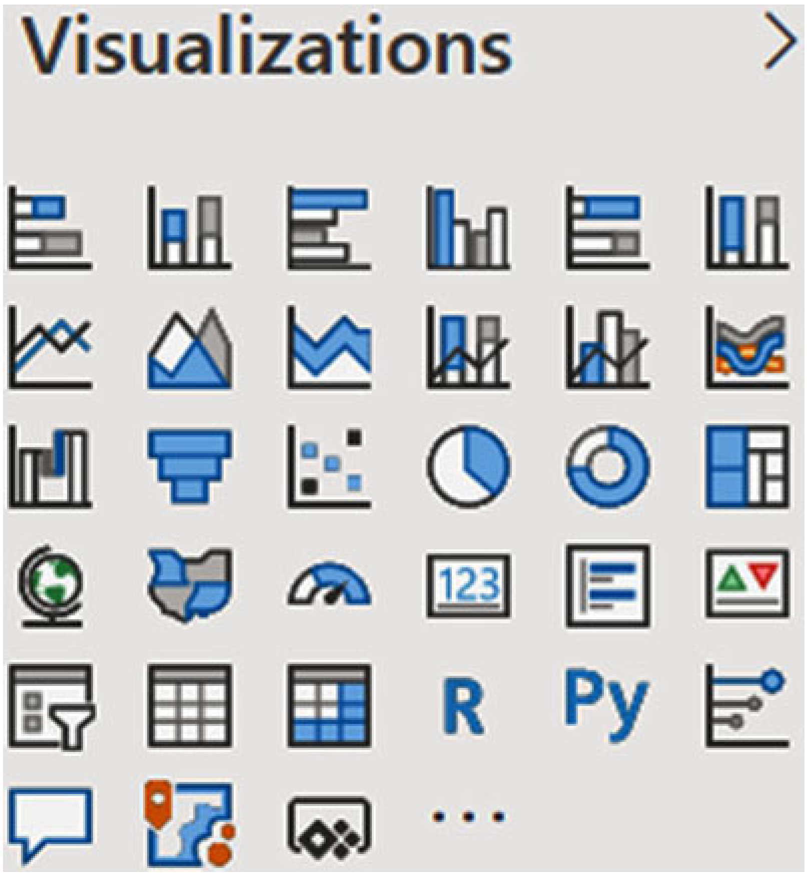


***Figure 5.4:*** *List of Visualizations*

***#3*** The various fields or calculations can be selected to be displayed on the visualization, as shown in the following screenshot:

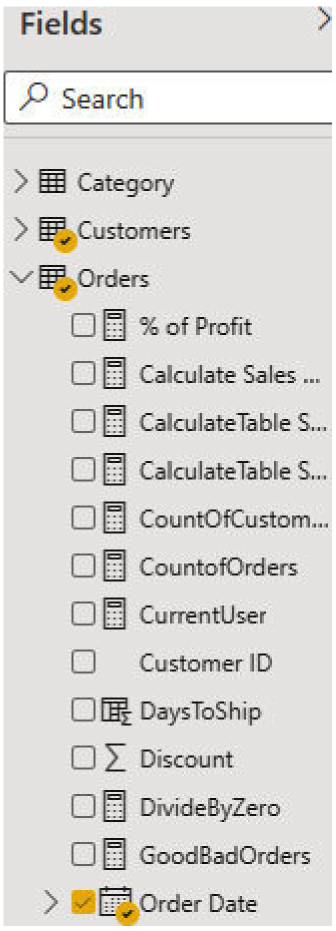
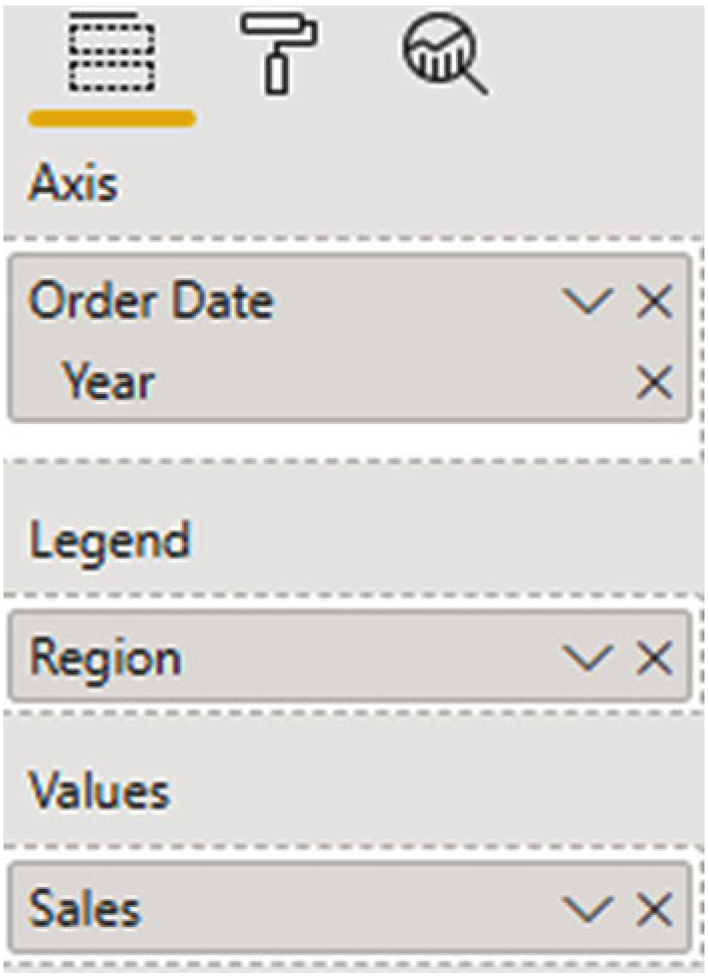


***Figure 5.5:*** *Select fields to display in Visualizations*

**#4 Fields, Format, and** The fields option is used to display the fields on the visualization. It displays the values on the visualization, such as x-axis, Y-axis, legends, etc. To remove a field from the visualization, simply remove it from here by using the X on the field.

Following is the section for the field placement on the visual. It also provides the options for formatting and analytics, as shown in the following figure:


Axis

Order Date

Year

Legend

Region

Values

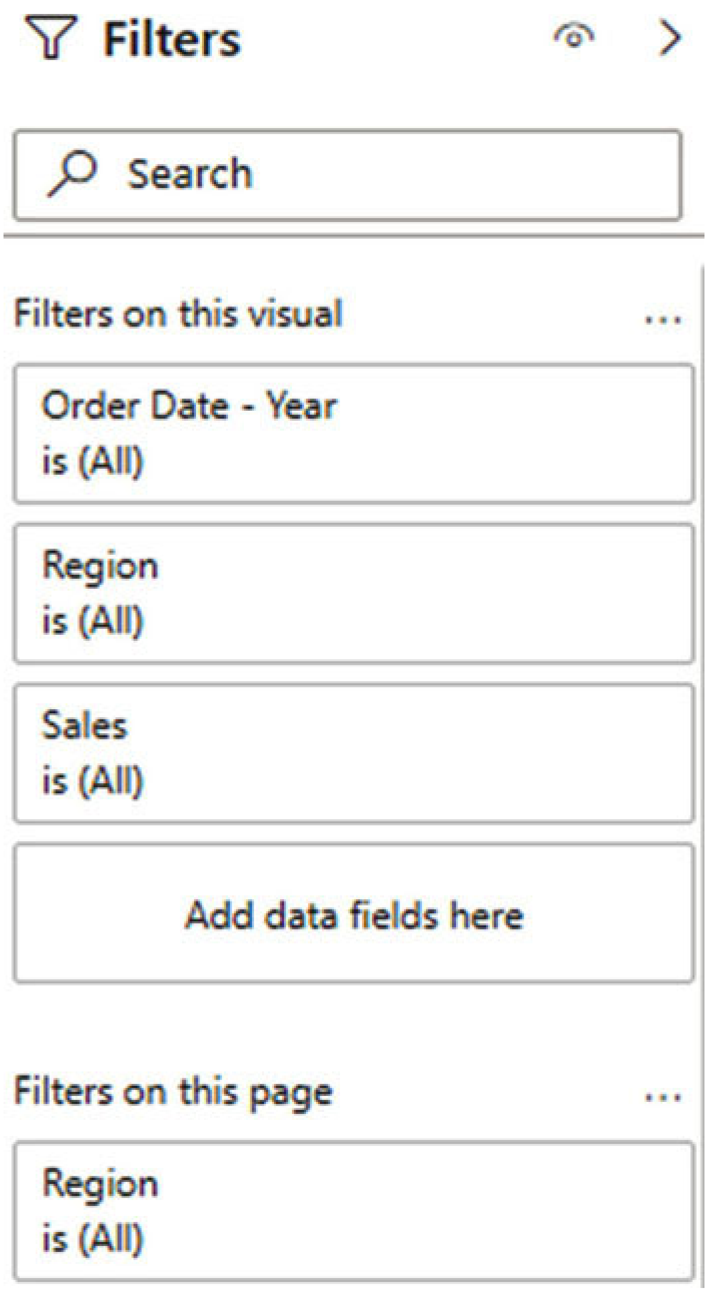
Sales

***Figure 5.6:*** *Fields, Format and Analytics*

The next roller pin option is the **Formatting** option. It is used to provide formatting to the visualization, such as color, font size, etc. The **Magnifying glass** option, is used to display the Analytics pane. It is contextual and displays the options based on the type of visualization selected.

**#5** The Filter pane is used to limit the data in the visualization. Filters can be set to filter a specific page or all the pages in the report. The fields not present in the visual can also be used as a filter. There are different types of filters. Filters can be used on a single visual or on an entire report.

The Filters pane provides the different ways to filter the data in the visual, as shown in the following figure:



Order Date - Year
is (All)

Region
is (All)

Sales
is (All)

Region
is (All)

***Figure 5.7:*** *Filters*

**The steps to create visualizations in the Power BI desktop and Power BI service are the same. You can follow the same steps to create a visualization in either application. The difference is that in the Power BI service, you will not have access to modify the data model.**

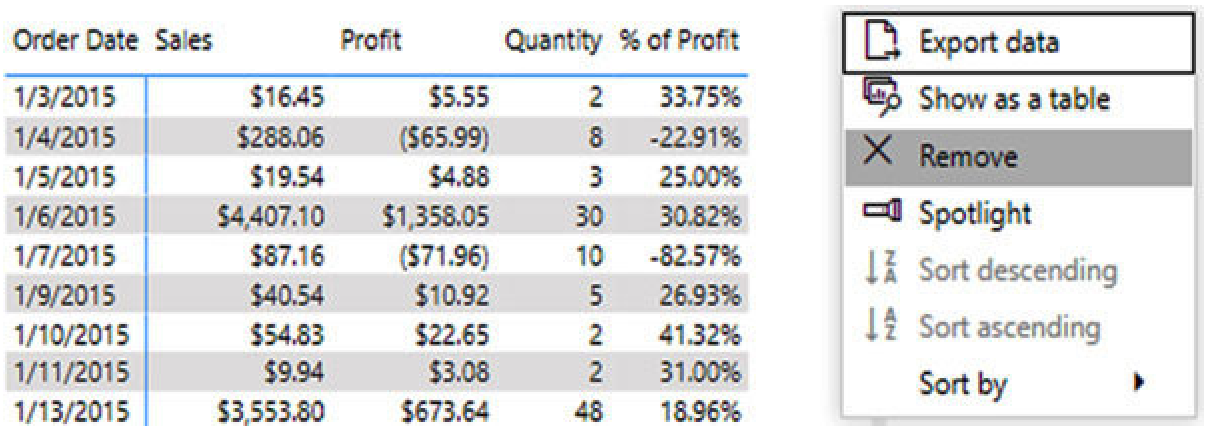
## *Creating multi-page reports using visualizations*

In this section, we will learn about creating reports in Power BI that includes filters, slicers, and the different types of visualizations.

Use the previously created **Chapter4_DAX** and save it as Remove the tables, **Digits** and **DateDim** as they were used only to explain some concepts in DAX.

## Remove any pages or visualizations

To remove a visualization, select the visual and click on the **...** symbol for more options and select as shown in the following screenshot:



| Order Date | Sales | Profit | Quantity | % of Profit |
|---|---|---|---|---|
| 1/3/2015 | $16.45 | $5.55 | 2 | 33.75% |
| 1/4/2015 | $288.06 | ($65.99) | 8 | -22.91% |
| 1/5/2015 | $19.54 | $4.88 | 3 | 25.00% |
| 1/6/2015 | $4,407.10 | $1,358.05 | 30 | 30.82% |
| 1/7/2015 | $87.16 | ($71.96) | 10 | -82.57% |
| 1/9/2015 | $40.54 | $10.92 | 5 | 26.93% |
| 1/10/2015 | $54.83 | $22.65 | 2 | 41.32% |
| 1/11/2015 | $9.94 | $3.08 | 2 | 31.00% |
| 1/13/2015 | $3,553.80 | $673.64 | 48 | 18.96% |

***Figure 5.8:*** *Removing the visualization*

Alternatively, you can also use It has a **PBIT** extension. When you open this application, it will ask for a parameter value. This parameter was created when we loaded the CSV files from a folder in *Chapter 2, Connect and* Select the default value and click on load.

This will load all the tables in your data model. The Power BI templates will be covered later in this chapter. To display the data in different ways, Power BI has a wide variety of charts. Each chart will be used for the specific data needs. To create a visualization, we will need a dimension or a metric value which will be displayed over the x-axis and y-axis.

## *Card*

A card displays a single value of the data element. It can be used to display the total value of a measure.

We will use the card visual to display the total number of and **Total** The advantage of using a card is that the business users and executives can get a quick glance of the important KPIs of the company.

**Exercise 1:**

**Display count of orders, customers, and total sales using card visualization**

The steps to create a Card visual are as follows:

(Refer to the page on Card in

Click on the **Report** From the select as shown in the following screenshot:

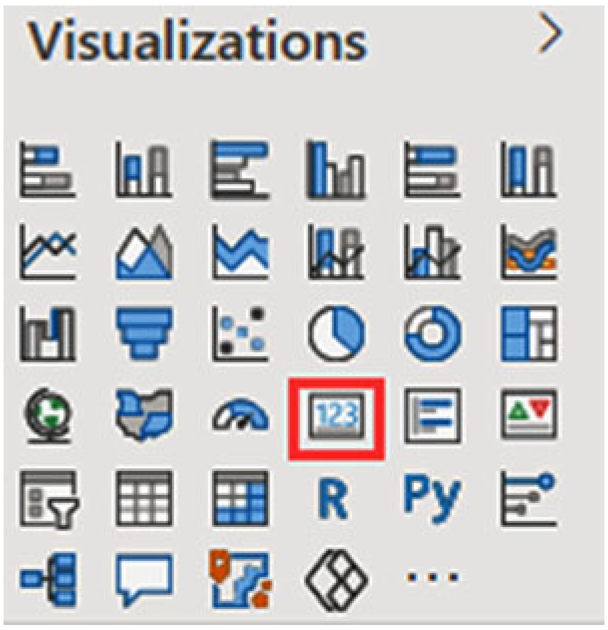


***Figure 5.9:*** *Selecting the Card*

This will create a place holder of the card.

From using the **Orders** table, select **Order**

**Fields:** The following is the list of visualizations, **Order ID** will show up under the Click on it to make sure that it uses **Count** as summarization, as shown in the following screenshot:

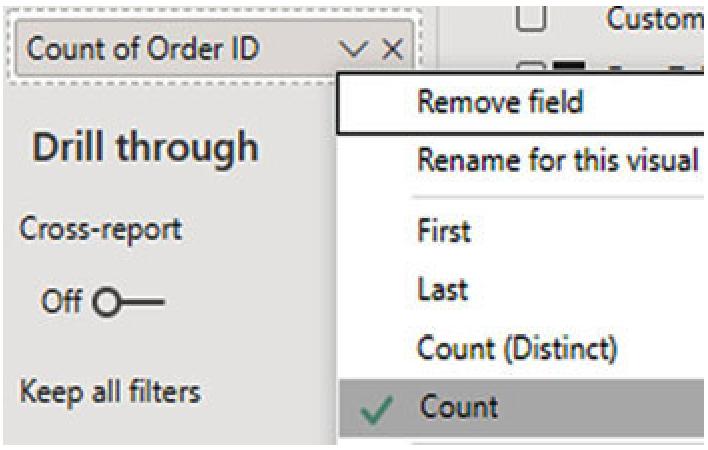


***Figure 5.10:*** *Count of Orders*

With the chart selected, next to Fields, use the roller-pin icon to set the formatting of the visual. Navigate through the different settings and change them one by one. To make any changes, remember to select the chart first.

Under **Data** set the color as black, **Display units** as the value decimal places as 0, **Text Size** as **30** and **Font family** as **Trebuchet**

Turn the **Category off** to remove the default caption of **of Order**

Turn **Title** to on by clicking on it. Use the **Title text** as **#** of **Orders** with **Text** size as 24 pt. Set its alignment to center and the **Font family** as **Trebuchet**

On the canvas, reduce the size of the card from the corner, as shown in the following screenshot:

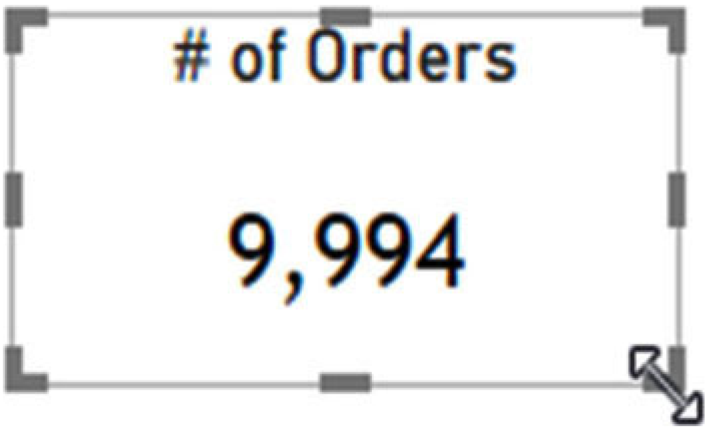


***Figure 5.11:*** *Reducing the size of the visualization*

There are many properties which you can explore and set as per your requirement.

## *Copy/Paste a visual*

Now to create the other cards for the # of **Customers** and **Total** copy and paste the preceding visualization.

Right click on the **#** of Orders visualization and select **Copy** and select **Copy** as shown in the following screenshot:

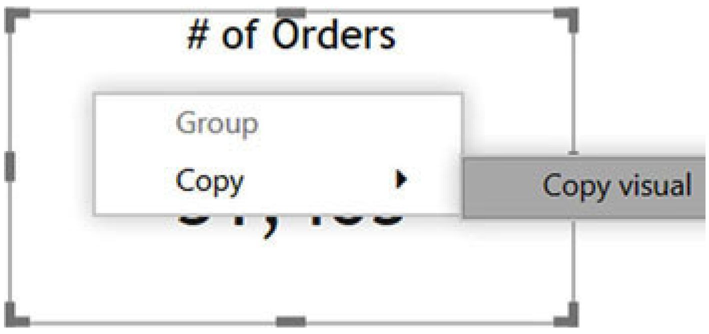


***Figure 5.12:*** *Creating copies of the visual*

From the **Home** ribbon, select Select **Paste** twice to create two copies of the same visual. Drag the copies of the visuals and lay them separately on the canvas.

**Creating # of Customers card**

Select the second visual which we just copied and pasted.

From under the **Orders** table, select **Customer ID** and place it over the fields (below the list of visualizations). Make sure that the aggregation is **Count** distinct.

Change the title to **# of**

**Creating Total Sales card**

Select the third visual which we just copied and pasted.

From under the **Orders** table, select **Sales** and place it over the fields (below the list of visualizations).

Use the roller-pin to change the formatting. Under **Data** set the **Display** units to millions and the value decimal places to 2. Change the title to **Total** as shown in the following screenshot:

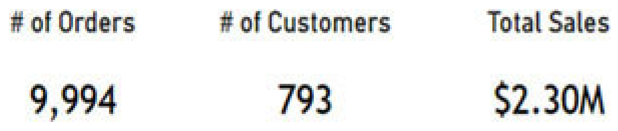


***Figure 5.13:*** *Displaying the totals.*

The cards are used to provide the summary of KPIs:

**Placing the image:** We will place an image/logo on the dashboard, for which, the steps are as follows:

Navigate to the **Insert** ribbon and select Browse to your enclosed images folder and select an image you want to put on the dashboard.

Resize the image and place it on the left-hand corner.

The visualizations created on **Page1** of the report so far will appear on your screen as shown in the following figure:

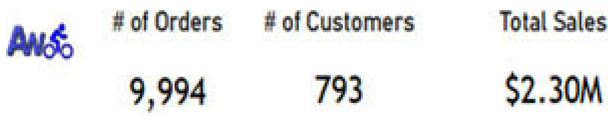


***Figure 5.14:*** *Cards and Logo*

Right now, the measure on the three cards displays the numbers for the entire data set. If we place the fields, such as **Order Date** on the these numbers will change accordingly. We will cover filters in the subsequent sections in this chapter.

**Remember, to make any changes to the visualization, it should be selected first. Once the visual is selected, you will see the properties for Fields, Formatting, etc.**

## ***Bar chart***

The Bar charts are used to compare values of unique categories. Bars can be displayed vertically, horizontally, or can be stacked. The length of the Bar displays the value of the category they represent.

The legends on the chart displays additional dimension as color.

**Exercise 2:**

**Create a Stacked Bar chart to display Sales for different Years and Region**

The steps to create a bar chart are as follows:

(Refer to the page on **Card_Stacked** Bar Chart in the enclosed

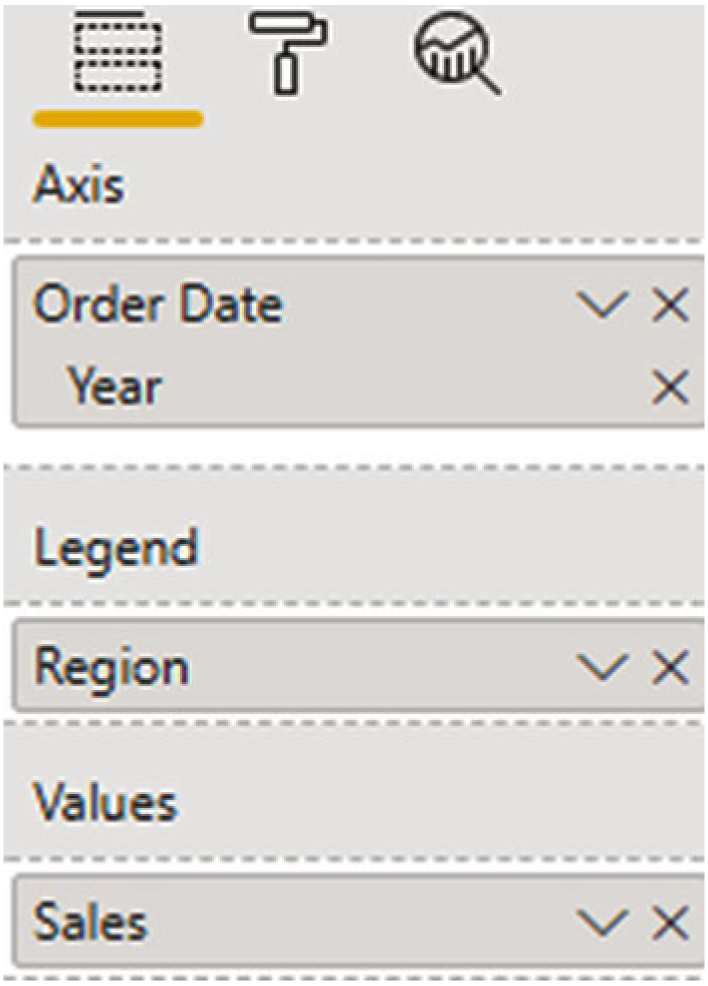
With the report view selected, on page 1, from Visualizations, select the first chart – Stacked bar chart.

With the bar chart place holder selected on the canvas, select the following fields from **Orders – Year** from the **Order Date** hierarchy, and from the **Customers** table, select and make the changes, as shown in the following screenshot:

On axis, place **Order Date/Year** from the **Orders** table.

On place **Sales** from the **Orders** table, This will create a bar chart with bars of the same color.

On from the **Customers** table, place This will produce different colors of bars for each of the


Axis

Order Date

Year

Legend

Region

Values

Sales

***Figure 5.15:*** *Fields for the bar chart*

**Formatting:** With the bar chart selected on the canvas, select the roller-pin icon to set the formatting.

Make the following changes:

Position – Right

Color – Black

Font – Trebuchet MS

Text size – 12 pt

**Y-axis**

Color – Black

Text size – 12 pt

Font family – Trebuchet MS

Turn the title off

**X-axis**

Color – Black

Text size – 12 pt

Font – Trebuchet MS

Turn the title off by clicking on it

**Data labels**

Display units – Millions

**Title**

Title text – “Sales by Year and Region". The formula section denoted by fx allows you to formulate the custom titles.

Size - 14 pt

Alignment – Center

Font – Trebuchet MS.

**Data color**

Use the data color property to choose the desired colors.

After all the formatting changes, the bar chart will appear like the one as shown in the following figure:

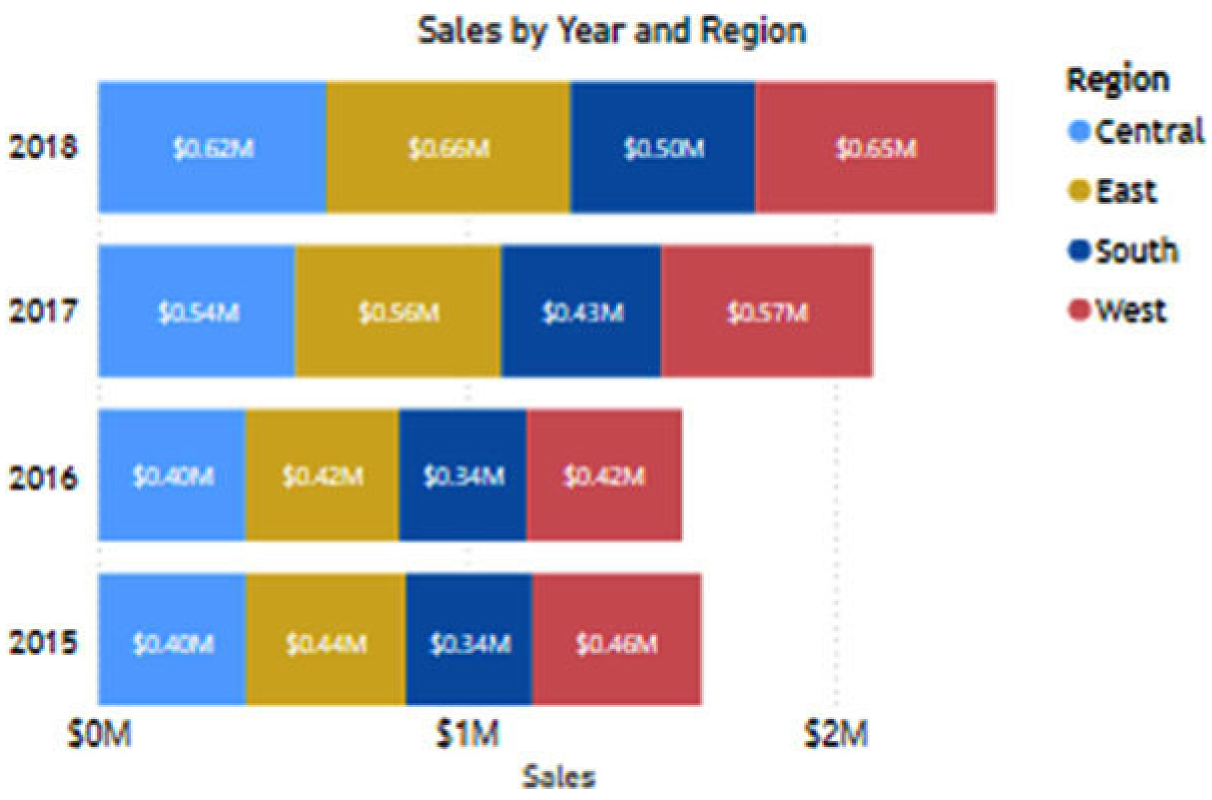


***Figure 5.16:*** *Stacked Bar chart*

There are a lot of formatting options that you can explore, and select as needed.

## *Filters and Slicers*

When creating the preceding visuals, you might have noticed that the fields used in creating the charts, automatically appear in the filter panel.

The data displayed in the charts is controlled by filters and slicers. Filters and slicers, both are used to display or limit the data in the visualizations based on the following selections made:

Filters are contextual. When visualization is selected on the canvas, it shows the visual filters and displays the fields used in the visualization as filters. If the visualization is not selected, only the page level filter option is displayed.

Filters are created by placing the fields in the Filter pane. The users make their selections in the filter pane to see their desired data.

Filters are of various types, such as visual level filters, page filters, report filters, and drill through filters.

Slicer is similar to Filters and is available in the list of visualizations. It is placed on the canvas like any other visualization.

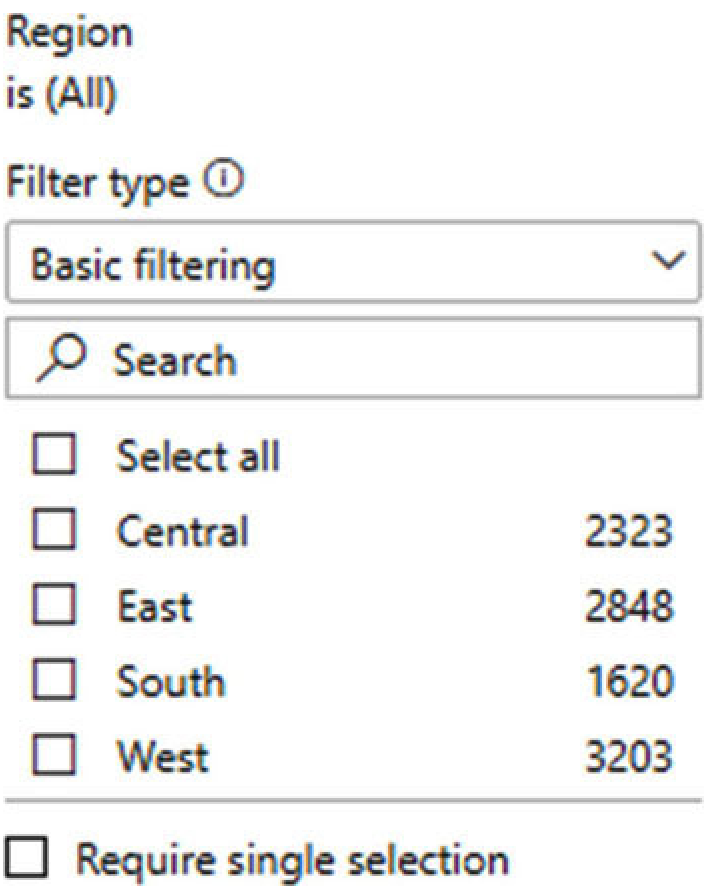
Slicers can be synced too. Navigate to the **View** ribbon and select the **Sync slicers** option.

We can use Filter or Slicer in the report. Filters and Slicers have similar functionality. Filters will work in most cases, but in situations where you want to filter the data on specific pages, slicers may be more useful.

Depending on the field used as filter, it can have different options, such as the following:

**Basic filtering:** In basic filtering, the users can select the actual values of the field. You can select all the values or the specific ones. In the preceding bar chart, **Region** is used as a filter.

Under Basic filtering, the values displayed are shown in the following figure:



Region
is (All)

Filter type

Basic filtering

Search

| | |
|---|---|
| ☐ Select all | |
| ☐ Central | 2323 |
| ☐ East | 2848 |
| ☐ South | 1620 |
| ☐ West | 3203 |

☐ Require single selection

***Figure 5.17:*** *Basic filtering*

By default, it is One or multiple values can be selected. If a single select is required, then check **Required single** This will allow the users to pick only one value of the

**Advanced filtering:** This option gives the users a wild card selection, that is, the advance filtering option is used to filter the visual with the wild card selection, such as when the data element contains or does not contain certain values, and the options can have values such as **Ea*** to get the chart for Region as shown in the following screenshot:

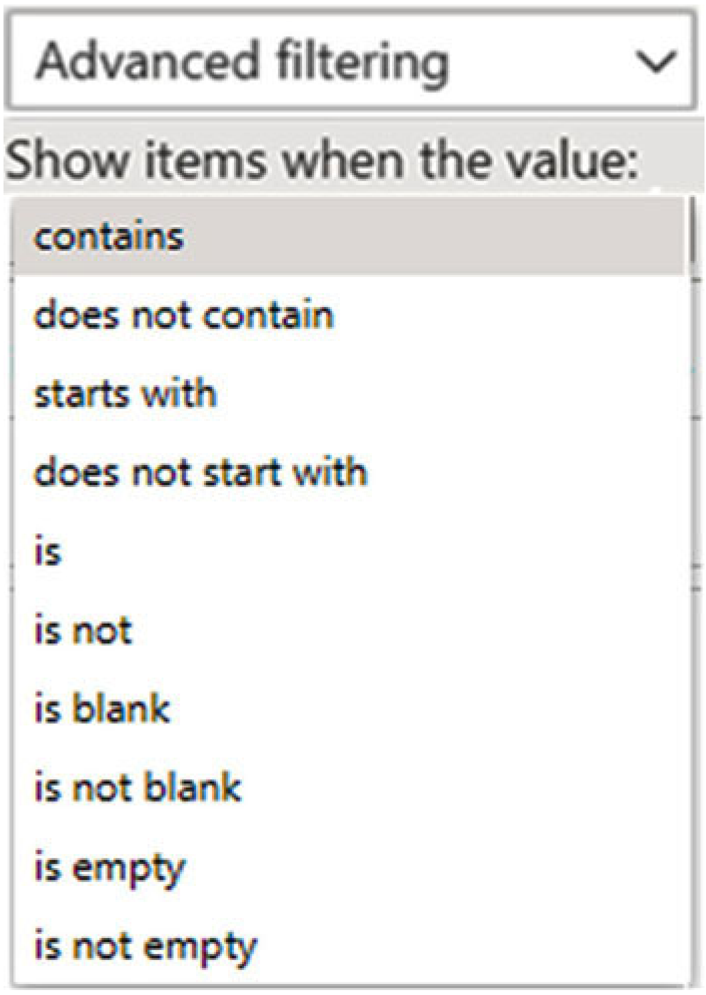
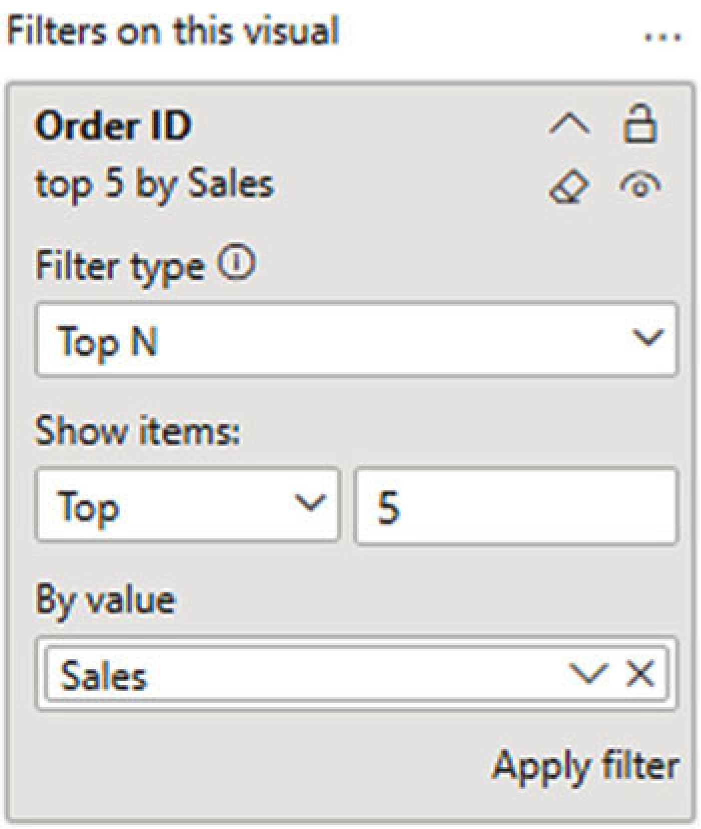


***Figure 5.18:*** *Advance filtering*

Top This filter condition is used to find the top or the bottom performing records. Say for example, we have created a table using the columns, **Order ID** and From the list of values, we are interested in finding the **Top 5** sales record. In this case, we will use the **Top N** filter and select **Apply** as shown in the following screenshot:


Filters on this visual

Order ID
top 5 by Sales

Filter type: Top N

Show items: Top 5

By value: Sales

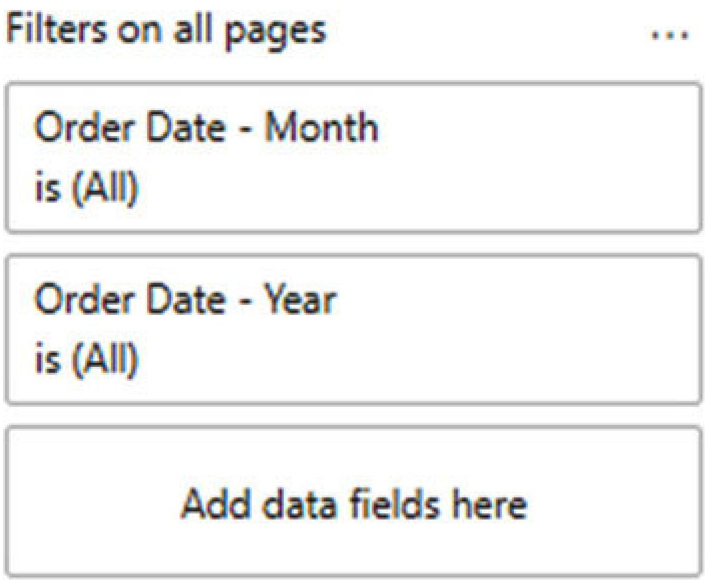
***Figure 5.19:*** *Top N filter*

Use the **Show items** dropdown to find the **Bottom** performers as well. It can have any integer; in this example, we have specified Top 5.

**Creating Filters in all the pages**

In our reports, Year and Month should be used as filters in all the pages.

To do this, we will create Filters in all the pages. To create this filter, from the drop the **Year** and **Month** fields into Filters on all the pages. Use the Basic filtering option in the filters, as shown in the following screenshot:


Filters on all pages

Order Date - Month
is (All)

Order Date - Year
is (All)

***Figure 5.20:*** *Filter on All Pages*

The filters for **Month** and **Year** will be displayed in all the report pages. Any number of filters can be created by placing the desired fields in the **Add data fields**

**We can create multiple visualizations on the same report; in this book, for the sake of clarity, we will create them on separate report pages.**

## *Line chart*

Line chart is best suited for displaying the trends of data over a period of time.

**Exercise 3:**

**UseLine Chart to display Yearly trend of Sales by Category**

The steps to create a Line Chart are as follows:

(Refer to the page on Line Chart in enclosed

With the Report view selected, click on **Page2** at the bottom. This will create another page of the report.

From select **Line**

With the line chart place holder selected on the canvas, place the following fields:

On Axis, from the **Orders** table, place

On Values, from the place

On Legend, from the **Category** table, place

This will create a line chart for Sales for the different **Years** by various

**Formatting:** With the line chart selected on the canvas, select the roller-pin icon to set the following formatting:

**Legend:** Make the following changes:

Position – Right

Color – Black

Font family – Trebuchet MS

Text size – 12 pt

**X-axis**

Color – Black

Text size – 12 pt

Font family – Trebuchet MS

Turn the title off by clicking on it

**Y-axis**

Color – Black

Text size – 12 pt

Font family – Trebuchet MS

Turn the title off by clicking on it

**Shapes**

Stroke width – 5. This will make the trend lines thicker.

Join type – Round

Line style – Solid

Turn on the show marker by clicking on it. This will show the round marker on the lines for each month.

Marker size – 7

**Title**

Title text – “Yearly Trend of Sales by Category". The formula section denoted by fx allows you to formulate the custom titles.

Size – 14 pt

Alignment – Center

Font – Trebuchet MS

The Line chart will appear like the one shown in the following figure:

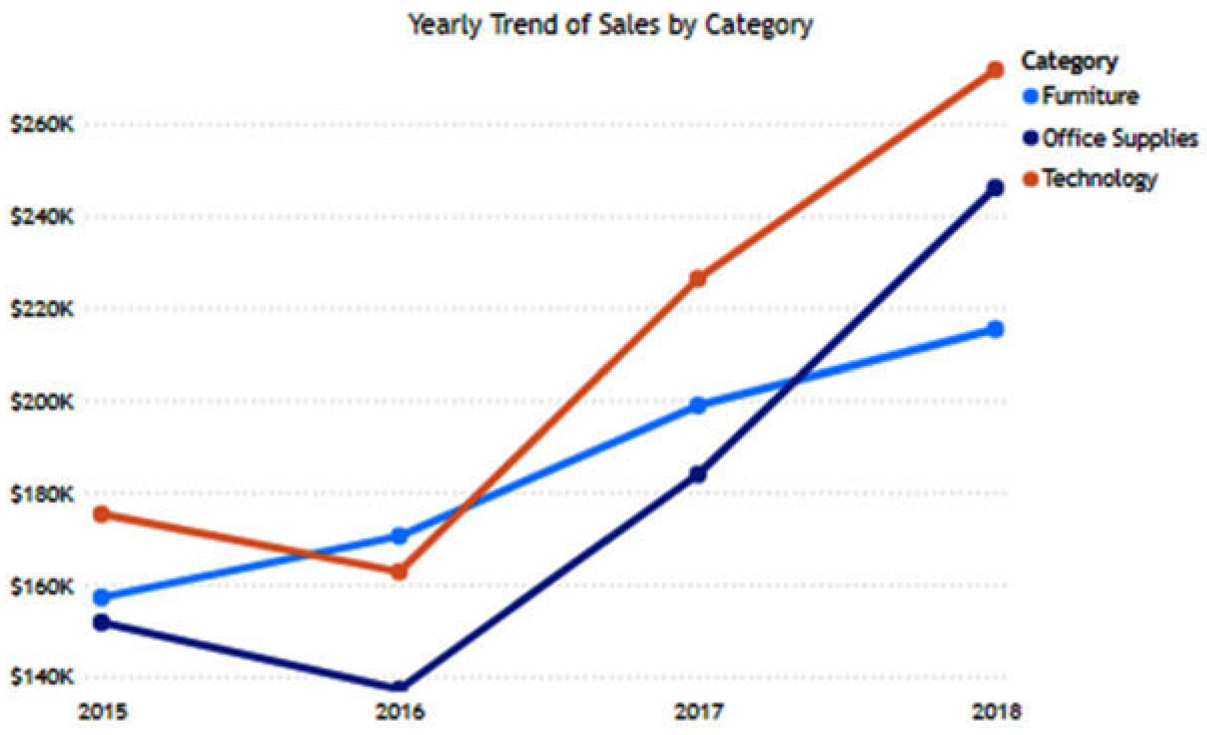


***Figure 5.21:*** *Line Chart*

## *Standardizing report development*

In the previous section, we learned how to create visualizations and use filters in Power BI. We can always follow this method for development, but in an enterprise environment, it is always good to standardize the report development process, so that the look and feel of the application is consistent. It takes less time for the other developers to follow a set of development benchmark.

In Power BI, we can standardize the report development process by using the following:

Report Themes

Templates

We will discuss these concepts, and then continue with visualizations.

## ***Report themes***

The design properties of each of the visualizations created (earlier) has to be set manually. This is time consuming and it also makes it harder to get the same look and feel for all the report pages and visualizations across the application.

Report themes in Power BI are used to standardize the visual design of the report pages. When a theme is applied, all the visuals and pages use the colors and formatting options from the selected theme.

There are two types of report themes, which are as follows:

These are the out-of-box themes and comes with the installation of the Power BI desktop.

These report themes are created by either customizing a built-in theme or by creating a new customized theme using the JSON file.

In the Power BI desktop, the report themes can be applied by either selecting a built-in theme from the View ribbon, using a customized theme (customized from the built-in), or importing a custom theme JSON file.

**Applying built-in theme**

We will apply a theme to our Power BI application We can apply the theme to our application and save the file as so that it doesn't distract the readers from the visualization exercise. But if you wish to follow in the same file, you are free to do so.

The steps to apply the theme to the Power BI application are as follows:

From the ribbon, navigate to the **View** tab and click on the drop-down next to the Themes to view all the themes and various options.

We can select any theme of our liking; for this example, we will select It will display the colors in the pages and filters, as shown in the following screenshot:

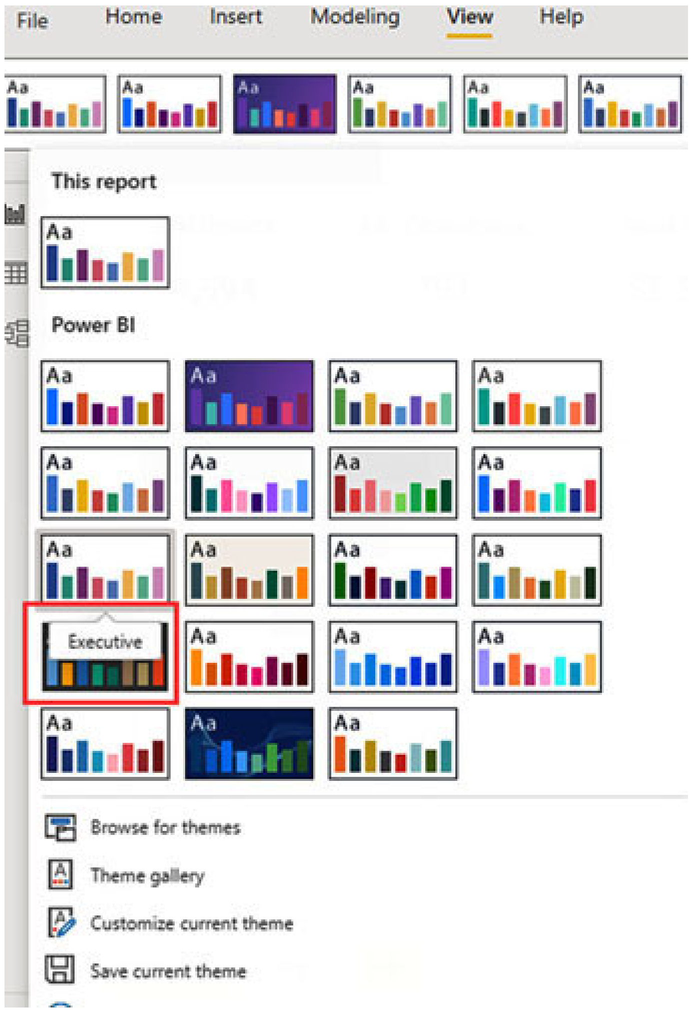


***Figure 5.22:*** *Themes*

This will change the colors of the report page, filters, etc.

If some component did not get the background color, we can change the background color to apply the theme color, as shown in the following screenshot:

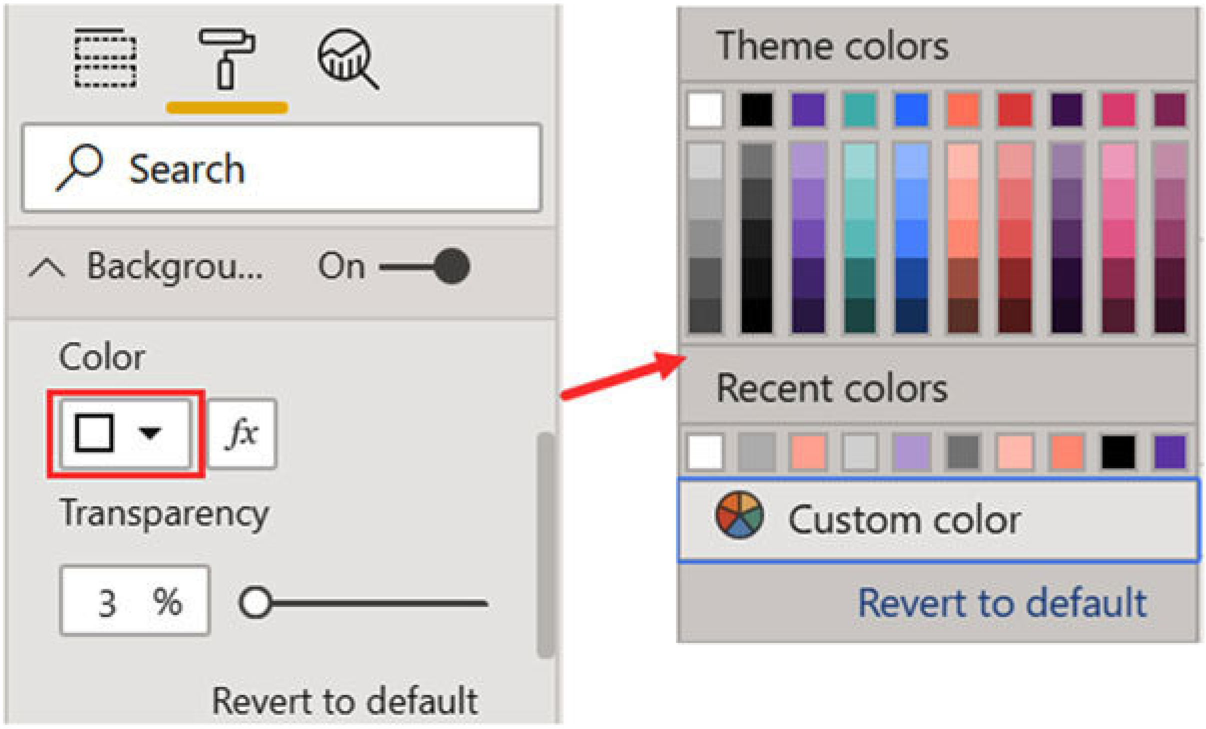


***Figure 5.23:*** *Changing the background color*

The other options available in the theme are as follows:

**Browse for** This option is used to import the downloaded JSON theme file. You can navigate to your file location.

**Theme** This option will take you to the themes created by the members of the Power BI community. You can download any JSON theme file. These downloaded JSON files are imported using the "Browse for themes" option.

**Customize current** This option will let you customize the selected theme. When this option is selected, you will get a dialog screen to customize the various design aspects, such as name and color, text, visuals, page, and filters pane. You can also create or modify a JSON theme file.

**Save current** Use this option to export custom theme as a JSON file. This exported theme can be used in the other Power BI applications.

## *Creating and using templates*

The Power BI templates are used to stream-line and standardize the development process. A template can be created to include the different design functionalities, such as report layout, data model, queries, themes etc., that can be used by the other developers in an organization.

To create and use a template file, complete the following steps:

Start creating a Power BI desktop application with the data model, desired theme, sample visualization, etc.

Go to **File** and save it as **Power BI template files** as shown in the following figure:

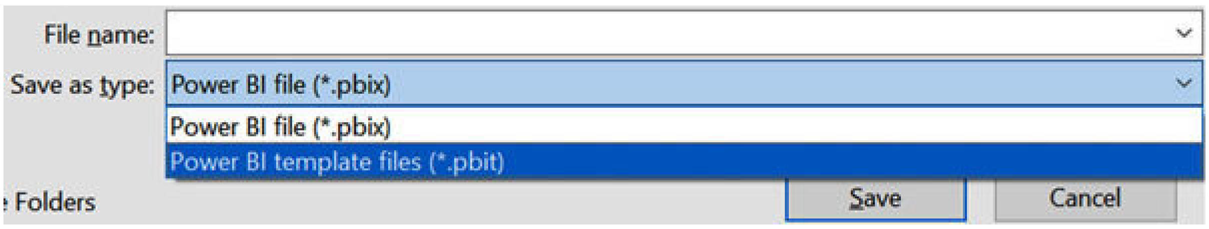

File name:
Save as type: Power BI file (*.pbix)
Power BI file (*.pbix)
Power BI template files (*.pbit)
Folders
Save
Cancel

***Figure 5.24:*** *Saving as Template*

To use a template, open the **.PBIT** file in the Power BI desktop by double clicking on it. Alternatively, launch the

Power BI desktop, and from the select **Import/Power BI**

You can easily change the **Chap5_Report** file to a Power BI template by saving it as the **.PBIT** file.

We will now continue with the rest of the visualizations using

## *Line and stacked column Chart*

The line and stacked charts are also called as a **combo** It combines the line and bar charts in one visualization. It is used when we have to compare multiple measures on the same axis.

**Usage of combo chart:**

A combo chart is used for the following requirements:

Comparing two measures.

Displaying one measure on the line chart and another measure on a column chart on the same axis.

Displaying the relationship between the two measures.

**Exercise 4:**

**Create a combo chart or line stacked chart to compare Sales and Profit in various Segments**

The steps to create a Combo or Line stacked chart are as follows:

(Refer to the page on the Line and stack Column Chart in the enclosed

With the Report view selected, click on the at the bottom to create a new page.

Select the **Line and Stacked** column chart from the visualizations.

With the placeholder of the chart selected, from the **Orders** table, select **Year** and

This will create a bar chart for the Sales by different Year.

From place the **Profit over the Line** values. This will create a line for **Profit over the Sales**

From place the **Segment over the Column** series. This will create a stack for the different **Regions over the**

Change the formatting of the chart by completing the steps covered in the previous sections in this chapter.

The Line and Stacked chart with Sales is displayed on the left axis and Profit is displayed on the right with Years on the x-axis are as follows, as shown in the following figure:

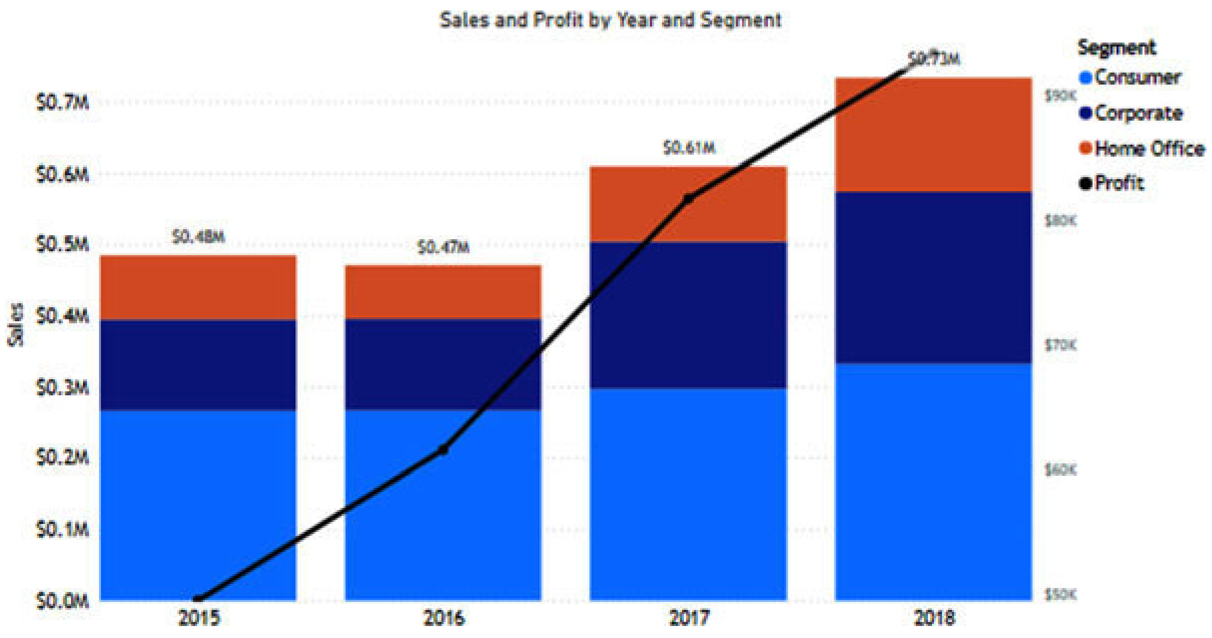


*Figure 5.25: Line and Stacked chart*

## ***Ribbon chart***

A Ribbon chart is used to discover which data category has the highest rank, that is, the Largest value. It displays the rank change effectively and always displays the highest rank/value on the top for each time period.

**Exercise 5:**

**Display how the different Product Categories rank over various months in a year**

The steps to create a Ribbon chart are as follows:

(Refer to the page on Ribbon in the enclosed

With the Report view selected, click on the + symbol at the bottom to create a new page.

Select the **Ribbon chart** from

With the placeholder of the chart selected on the canvas, from the **Orders** table, drag and place **Month** on the axis. Place **Sales** on values.

From the **Category** table, place **Category** over the

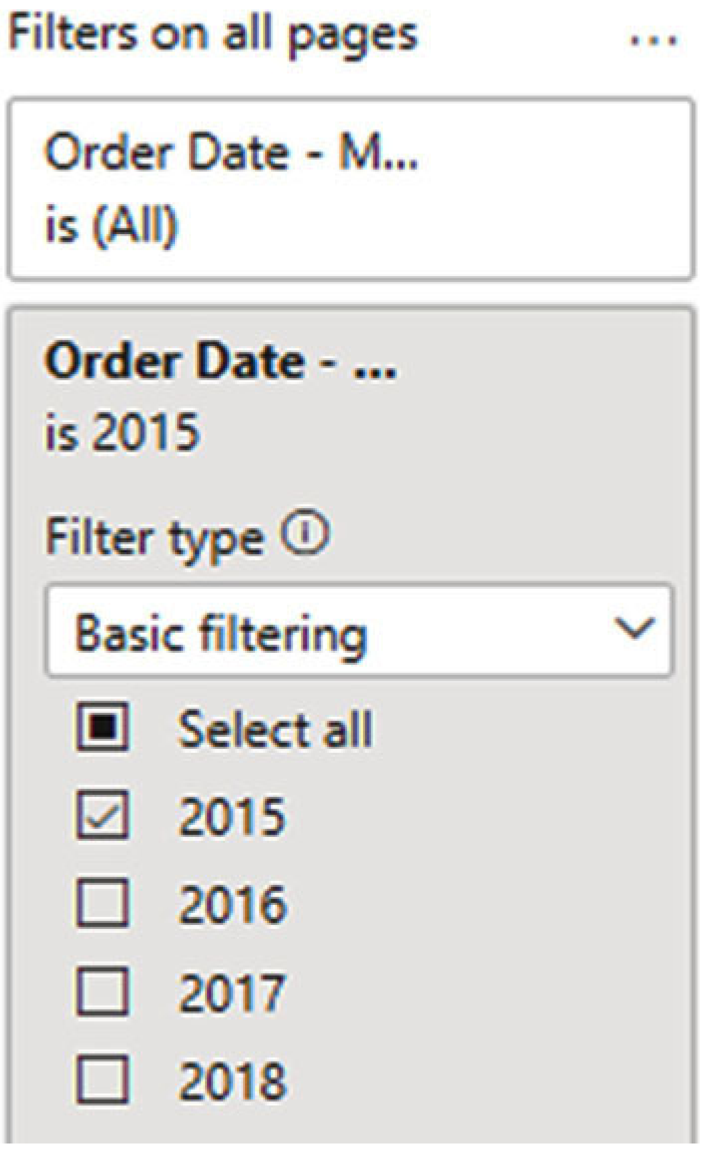
This will create a month-over-month ribbon chart.

We can select the **Year** filter using the Filter on all the pages to see the ribbon chart for the different months in a Year.

Formatting in the ribbon chart is similar to the other charts with some additional options.

Select the chart and click on the roller pin for formatting. Select This is the formatting option for the ribbons in the chart. The Spacing option adjusts the spacing between the ribbons on the chart. Change it to

After performing the formatting, as shown in the following figure, which is similar to the other charts, we will get a ribbon chart.


Filters on all pages

Order Date - M...
is (All)

**Order Date - ...**
is 2015

Filter type

Basic filtering

- [■] Select all
- [✓] 2015
- [ ] 2016
- [ ] 2017
- [ ] 2018

***Figure 5.26:*** *Selecting Year to filter the data in Ribbon chart*

The following chart displays the ribbons for the Sales by Month and Category for the Year 2015:

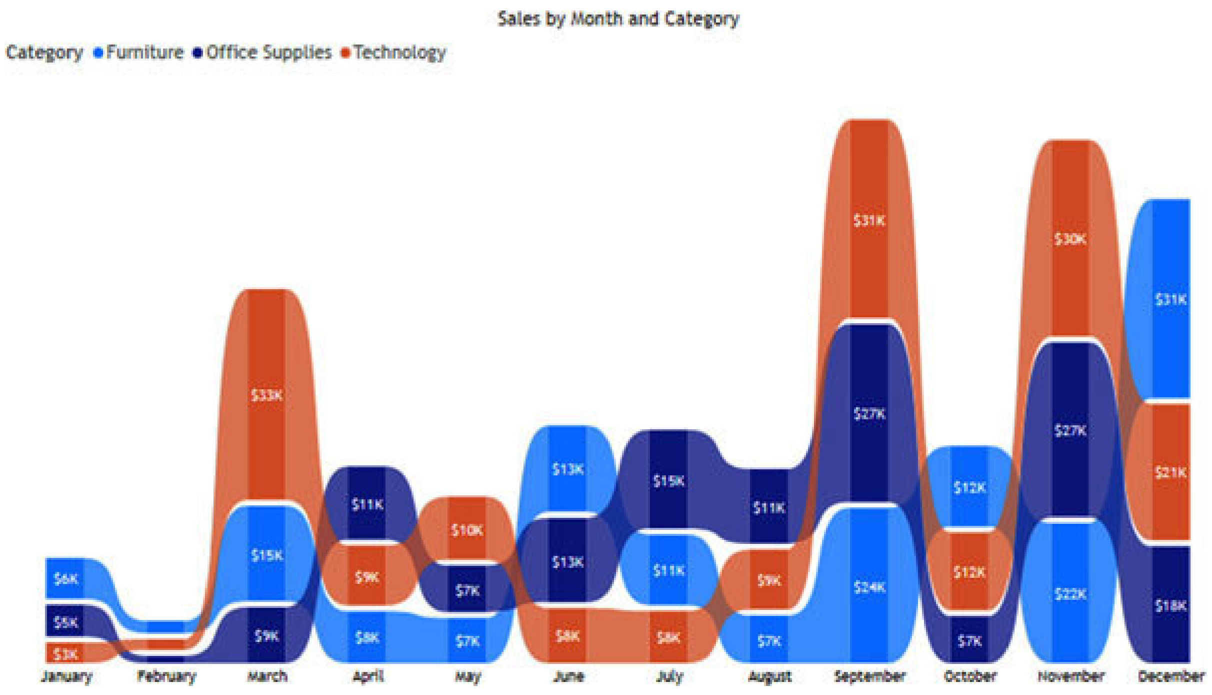


***Figure 5.27:*** *Ribbon Chart*

From this chart, just by a glance, we can say that in Jan, Furniture ranked higher, and in March, Technology ranked higher as compared to the other technologies.

## *Waterfall chart*

The Waterfall chart displays the running total of a measure. It shows how a measure value has changed over a period of time.

**Exercise 6:**

**Display how the different Product Categories rank over the various months in a year**

The steps to create a Water fall chart are as follows:

(Refer to the page on Waterfall in the enclosed

With the Report view selected, click on the + symbol at the bottom to create a new page.

Select the Waterfall chart from the visualizations.

With the placeholder of the chart selected on the canvas, from the **Orders** table, drag and place **Month on** Place **Profit** on Values.

This will create a waterfall chart. We can select a specific year from the filter to see the how the profit has increased or decreased over the different months.

Use Breakdown to add more data; this will help in displaying the contributors to the Profit's progression.

From the **Category** table, place **Category** over the Breakdown. We can also select a specific **Category** from the filter to see the effect of one category on profit.

The formatting is similar to the other charts, as shown in the following screenshot:

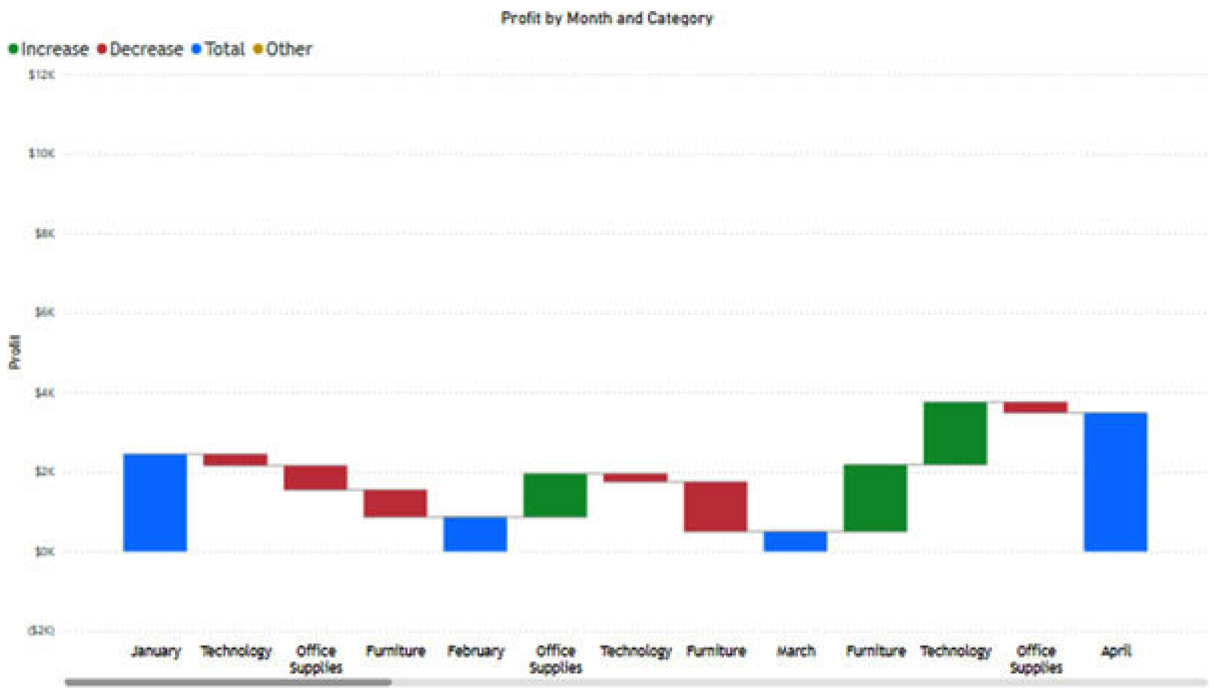


***Figure 5.28:*** *Waterfall chart*

## *Scatter chart*

A scatter chart is used to display the relationship between two measures. To create a scatter chart, the measures are placed on the x-axis and y-axis. It uses bubbles/circles or any other available shapes to represent the values of the measures. A dimension can also be used to analyze the measures over the different categories.

**Exercise 7:**

**Display the relationship between Profit and Quantity for different Customers by Count of Orders they made.**

Also, get the Top 10 Customers by Profit.

The steps to create a Scatter chart are as follows:

(Refer to the page on scatter chart in the enclosed

With the Report view selected, click on the + symbol at the bottom to create a new page.

Select the Scatter chart from the visualizations.

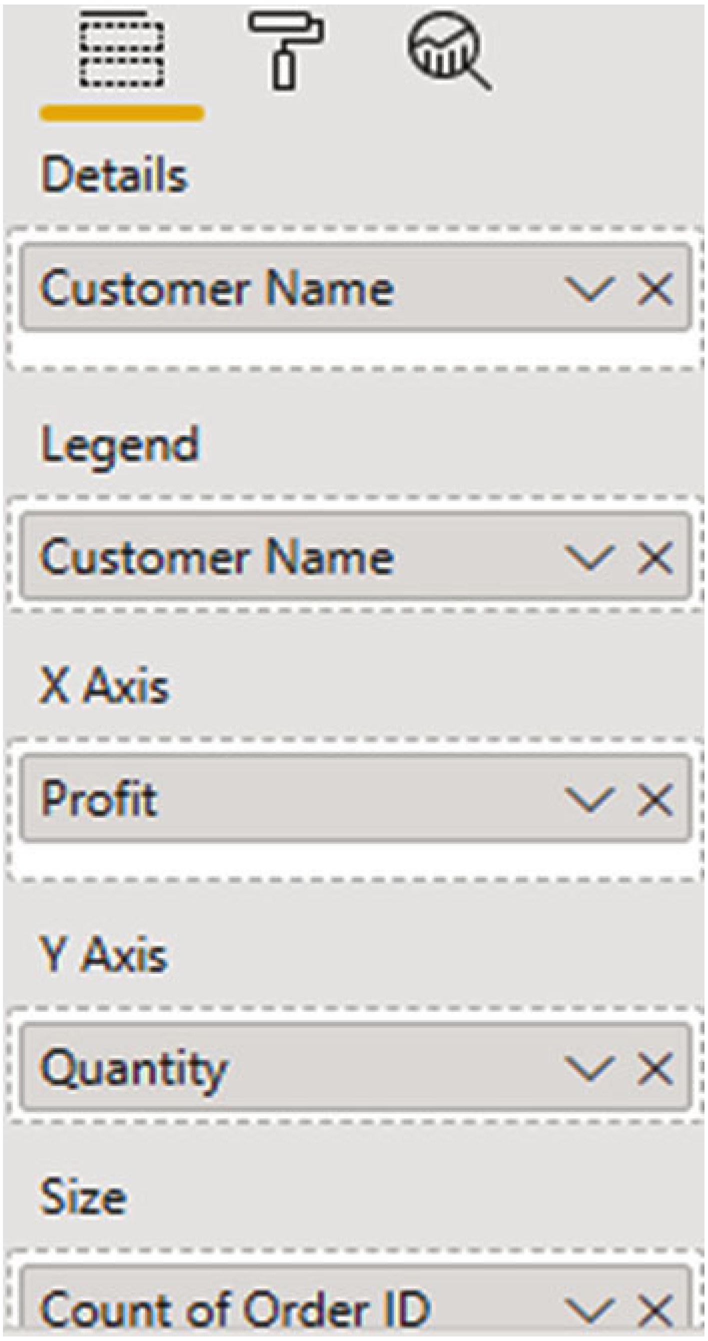
With the placeholder of the chart selected on the canvas, from the **Orders** table, drag and place **Profit** on the x-axis and **Quantity** on the y-axis.

This will display just one bubble as it is displaying the relationship between the measures for the entire data set.

From the **Customer** table, place **Customer Name** on We will get a lot of bubbles corresponding to all

To get bubbles of different colors based on the customers, from the **Customers** table, place **Customer Name** on the legend. This will display the bubbles in different colors.

Now, we will set the size of the bubble based on the number of **Orders** made. To achieve this, from place **Order ID** on as shown in the following screenshot:


Details

Customer Name

Legend

Customer Name

X Axis

Profit

Y Axis

Quantity

Size

Count of Order ID

***Figure 5.29:*** *Field placement to create a scatter chart*

Top 10 To display the top 10 customers, from the **Customer Name** filter, select the filter type as Top N and specify the value as 10. In By value, drag and place **Profit** from the **Orders** table.

This will give a scatter chart for the top 10 customers by profit and the number of orders they made and the quantity purchased.

Select the roller pin icon to set the formatting. Using the **General** option, make the following changes:

Turn off the legend's display as we don't need the legends.

On the x-axis, change the color to black and increase the font size to 12. Change the font family to Trebuchet MS, or as you desire.

On the y-axis, do the same as the x-axis.

On Data Color, change the colors, if they appear similar to each other. It will help in differentiating between them.

Align the Title to center.

We can hover over the bubble (in the chart) and get the name of the etc, as shown in the following screenshot:

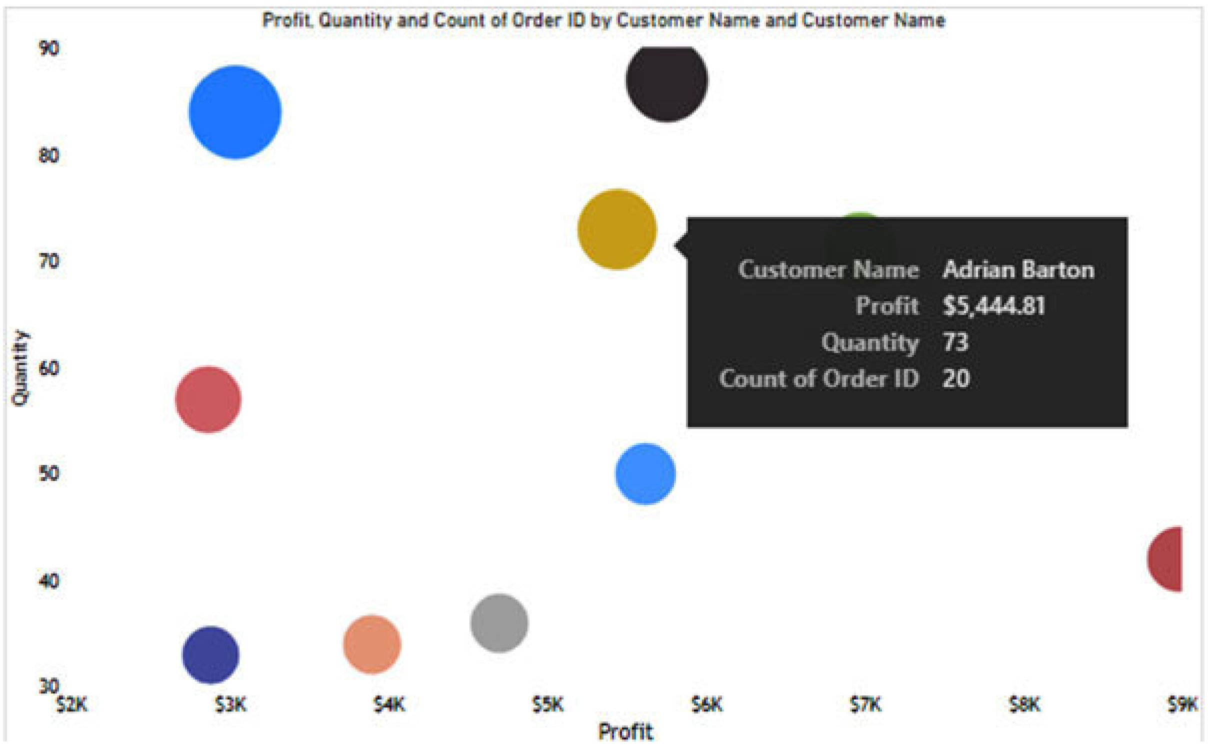


***Figure 5.30:*** *Scatter plot*

## *Donut chart*

The donut and pie charts are similar charts as they both display the percentage of the total. The different colors on the chart display the different categories that make up the total. The Donut chart has an empty space in the center, just like a donut.

**Exercise 8:**

**Display the percentage of Profit by each Segment**

The steps to create a Donut chart are as follows:

(Refer to the page on Donut Chart in the enclosed

With the Report view selected, click on the + symbol at the bottom to create a new page.

Select the donut chart from the visualizations.

With the placeholder of the chart selected on the canvas, from the **Orders** table, place **Profit** on **Values** and **Segment** on the

This will display a donut chart with **Segment** on colors and percentage of **Profit** in each Segment.

Change the formatting for and **Data**

The donut chart will appear as shown in the following figure:

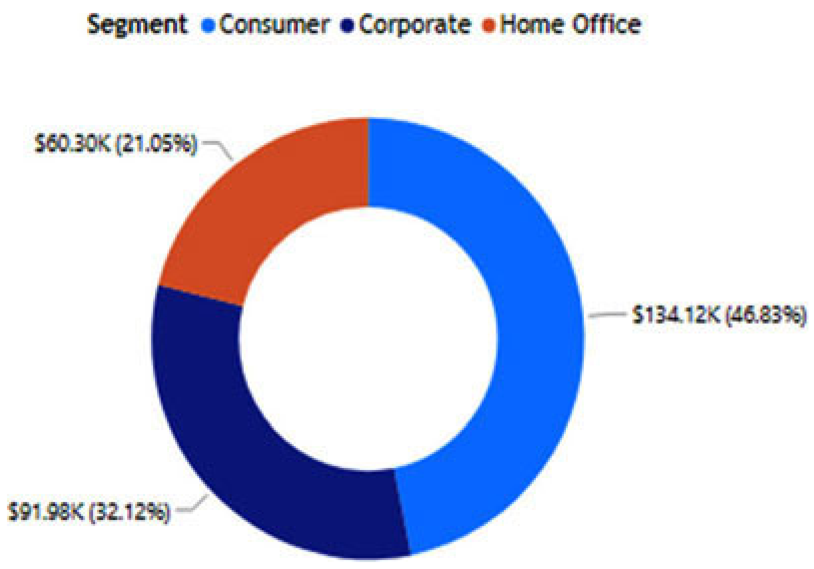


***Figure 5.31:*** *Donut Chart*

## *Treemap*

Treemap is used to display the hierarchical/tree data in a set of rectangles. The size of the rectangle is based on the value of the measure. A Treemap is used to display large amounts of hierarchical data, which is difficult to display in any other chart. The bigger values are displayed on the left.

**Exercise 9:**

**Display Sales by Sub-Category and Segment**

The steps to create a Treemap are as follows:

(Refer to the page on Treemap in the enclosed

With the Report view selected, click on the + symbol at the bottom to create a new page.

Select the **Treemap** chart from the visualizations.

With the placeholder of the chart selected on the canvas, from the **Category** table, place **Sub-Category** on the From the **Orders** table, place **Sales** on From the **Customers** table, place **Segment** on the

This will display a Treemap chart with colors on the Segment which is further divided by different Segments, as shown in the following screenshot:

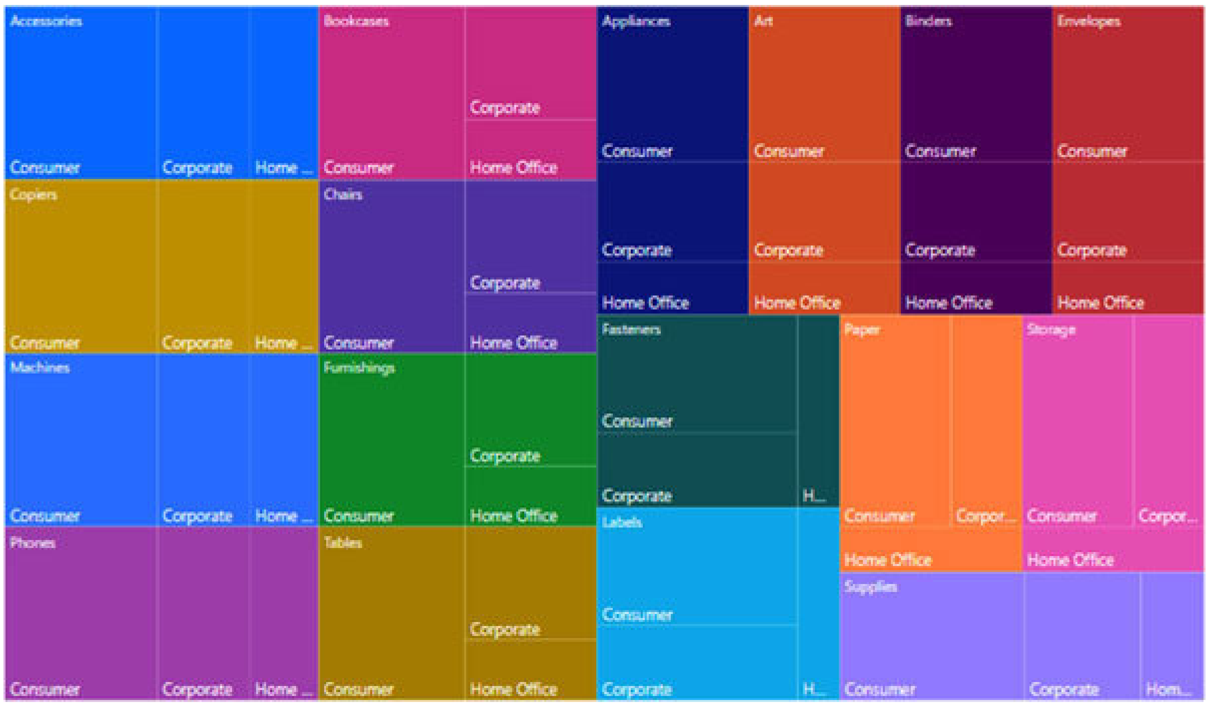


***Figure 5.32:*** *Treemap*

**Highlighting and Cross filtering:**

**Exercise 10:**

**Use Treemap as a filter for other charts**

(Refer to the page on Highlighting and Cross filtering in the enclosed **Chapter**

A Treemap is a great way of *Highlighting and Cross filtering* the other charts on a page. To try this out, copy the Treemap on a separate page and copy some other charts too on the same page. For this example, we will copy the line, column chart, and donut chart.

Now, when we click on the the other charts will filter and vice versa.

## *Map*

A Map is used to display the data based on the geographical locations. It is created by using the following field setting, as shown in the following figure:

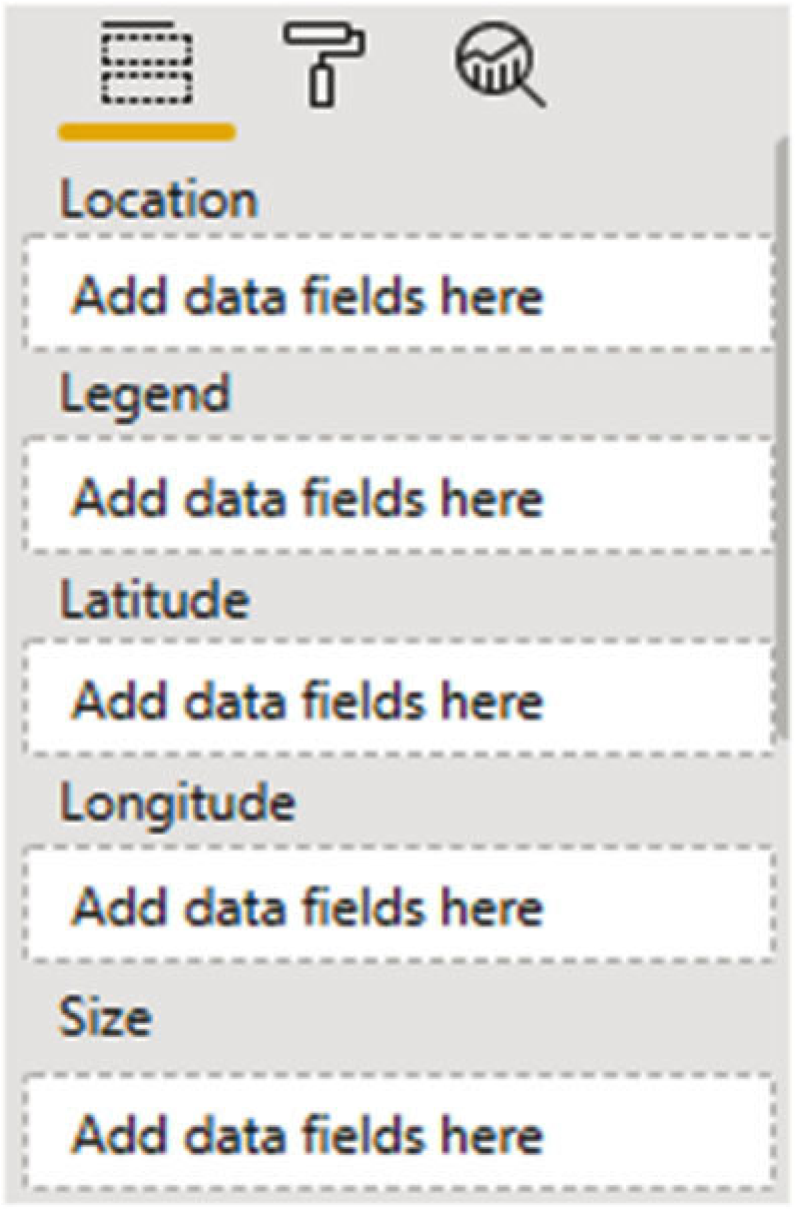


***Figure 5.33:*** *Map field Setting*

In the preceding field setting, you will find the following:

**Location:** It is the location-based field in the tables, for example, State or City. This will create circles on a Map.

**Legend:** It will have another dimension, which will display the different colors in Locations.

**Latitude and Longitude:** In case there is some ambiguity in the location data, such as duplicate names of the cities, etc., we can specify the Latitude and Longitude values.

**Size:** This will display circles with the size according to the value of measure in that location. For example, if we use it will display bigger circles if the Sales in that location is higher.

**Exercise 11:**

**Display Count of Customers in various States and Region**

The steps to create a Map are as follows:

(Refer to the page on Map in the enclosed

With the Report view selected, click on the + symbol at the bottom to create a new page.

Select the **Map chart** from the visualizations.

With the placeholder of the chart selected on the canvas, from the **Customers** table, select This will display a **Map of USA** with circles for each State.

Place **Region** on This will display the map with different colors of

From the **Orders** table, place **CountofCustomers** on

This will change the size the circles based on the amount of If we hover over the circles, we can see the tooltip display **CountofCustomers** in **State** and as shown in the following screenshot:

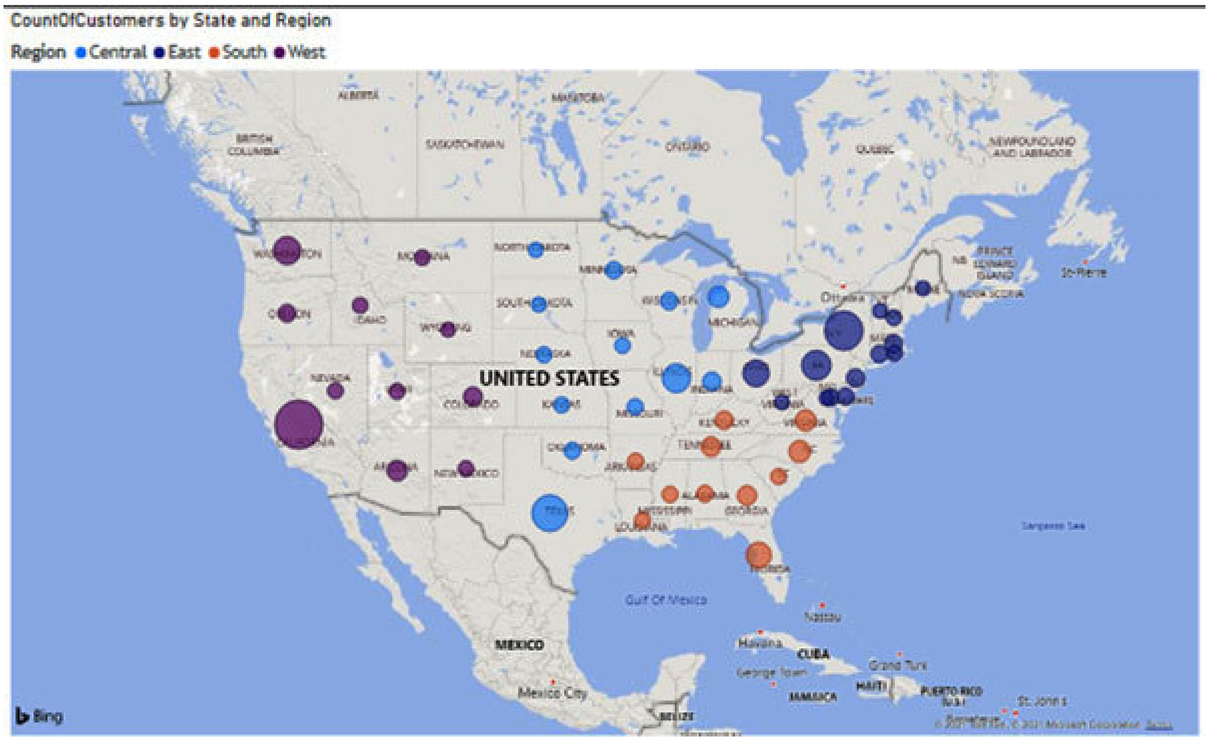

CountOfCustomers by State and Region
Region ● Central ● East ● South ● West
UNITED STATES
MEXICO
Gulf Of Mexico
CUBA
Bing

***Figure 5.34:*** *Map visualization*

Try the formatting options to change the look and feel of the Map.

One important option is Map Styles. It has themes such as Road, Aerial, Dark, Light, and Grayscale. Try the different options and see the difference.

## *Filled map*

A filled map, also known as the choropleth map, displays the filled shapes or patterns to display how a measure value differs across the geographical locations.

**Exercise 12:**

**Display Count of Customers in various States and Region**

We performed this exercise in the previous section using Map. We will do the same, using Filled Map and see the difference.

The steps to create a Filled Map are as follows:

(Refer to the page on Filled Map in the enclosed

With the Report view selected, click on the + symbol at the bottom to create a new page.

Select the **Filled Map** chart from the visualizations.

With the placeholder of the chart selected on the canvas, from the **Customers** table, select This will display a **Map of**

**USA** 'filled' with one color.

On the place This will display the map with different filled colors of the It will be easier to see which **States** fall under which

From the **Orders** table, place **CountofCustomers** on Tooltips.

The filled map will appear as shown in the following figure:

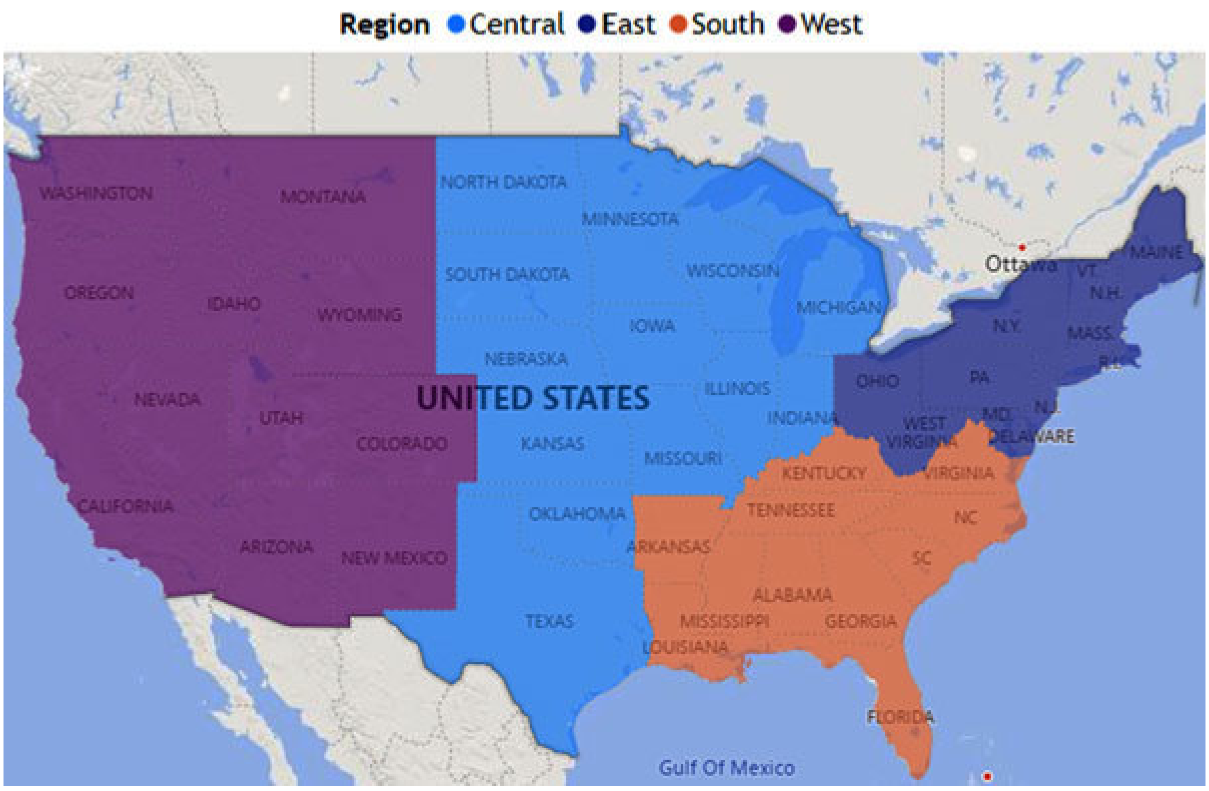


***Figure 5.35:*** *Filled Map*

## Q&A

Q&A allows the user to explore the data based on the available data set. The question can be asked in a natural language. Power BI makes a suggestion on the questions. We can pick the default questions or type a custom question.

When a question is typed, Power BI selects the best visualization to answer your question.

**Exercise 13:**

**Use Q&A and let a user get answers to the questions**

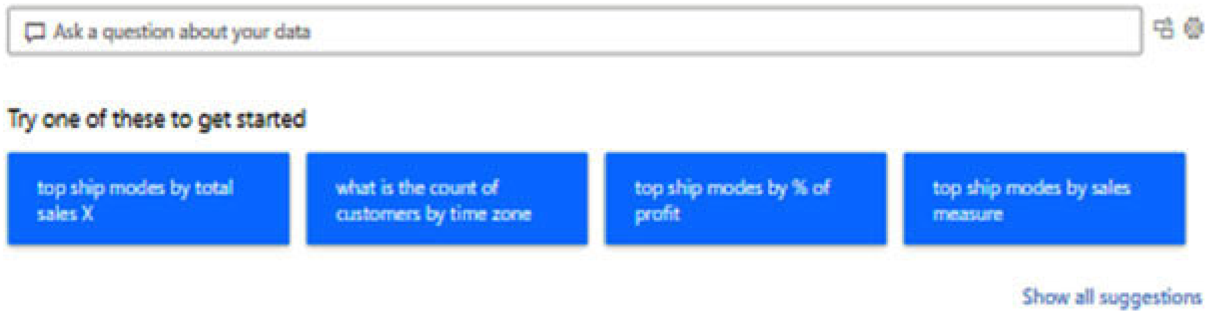
The steps to create a Q&A are as follows:

(Refer to the page on Q & A in the enclosed

Select the Report view and add a new page by clicking on the + symbol.

Select the **Q&A** type chart from the

The place holder of the chart shows a question edit box and some suggestions on the questions, as shown in the following figure:



Ask a question about your data

Try one of these to get started

top ship modes by total sales X
what is the count of customers by time zone
top ship modes by % of profit
top ship modes by sales measure
Show all suggestions

***Figure 5.36:*** *Q&A place holder with suggestions*

We can type the following question: **Top 5 order sales by customer ID.**

Power BI will automatically choose the bar chart to display this information, as shown in the following screenshot:

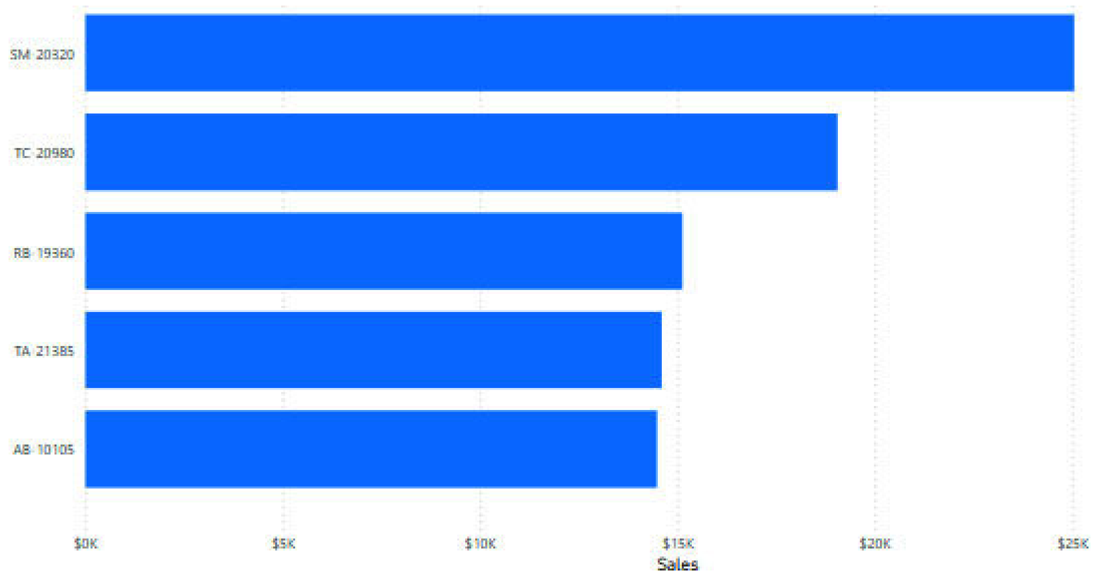


***Figure 5.37:*** *Q&A displays a Bar chart*

If we want to choose a particular visualization, mention it in the Q&A. For example, a Q&A like the preceding top 5 order sale by customer ID as Table. This will display the data as a table.

## ***Drill through reports***

Drill through reports is useful, when we want to show the summary on one page and details of it on another page. For example, the summary page displays the sales by each customer and the details page shows the details of the Customers. Click on the customer name on the summary page and it will take you to the details page.

**Exercise 14:**

**Create a Summary page to display Sales by Customers**

Clicking on a particular customer will drill through to the detail page.

The steps to create the drill through reports are as follows:

(Refer to the page on Summary and Drill Thru in the enclosed

Select the Report view and add a new page by clicking on the + symbol. Name this page as

Select the Stacked bar chart from Visualizations. From add **Customer** From select

This will create a bar chart for **Sales by**

Create a new page by clicking on the + symbol at the bottom and name it **Drill** On this page, use the table to display all the fields from the **Customers** table. This page will work as a detail page.

On the **Drill Thru** page, select the **Customers** table, click on the more option's ellipses next to **Customer Name** and select **Add to drill** as shown in the following screenshot:

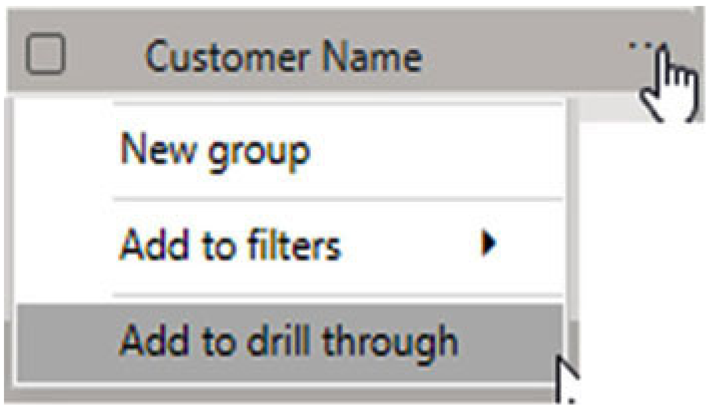
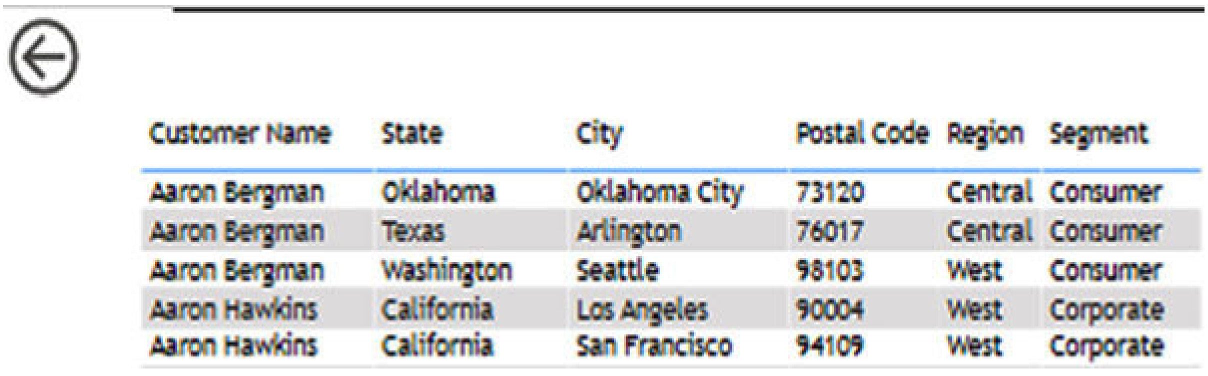


***Figure 5.38:*** *Adding Customer Name to drill through*

This will add **Customer Name** as a Drill Through field under fields. Observe that Power BI has automatically added a back button next to the table, as shown in the following figure:



| Customer Name | State | City | Postal Code | Region | Segment |
|---|---|---|---|---|---|
| Aaron Bergman | Oklahoma | Oklahoma City | 73120 | Central | Consumer |
| Aaron Bergman | Texas | Arlington | 76017 | Central | Consumer |
| Aaron Bergman | Washington | Seattle | 98103 | West | Consumer |
| Aaron Hawkins | California | Los Angeles | 90004 | West | Corporate |
| Aaron Hawkins | California | San Francisco | 94109 | West | Corporate |

***Figure 5.39:*** *Back button on drill through*

**Drill thru in Action:**

To check the drill through option, navigate to the **Summary** page, right click on any customer name/bar and select **Drill through** and then select **Drill** as shown in the following screenshot:

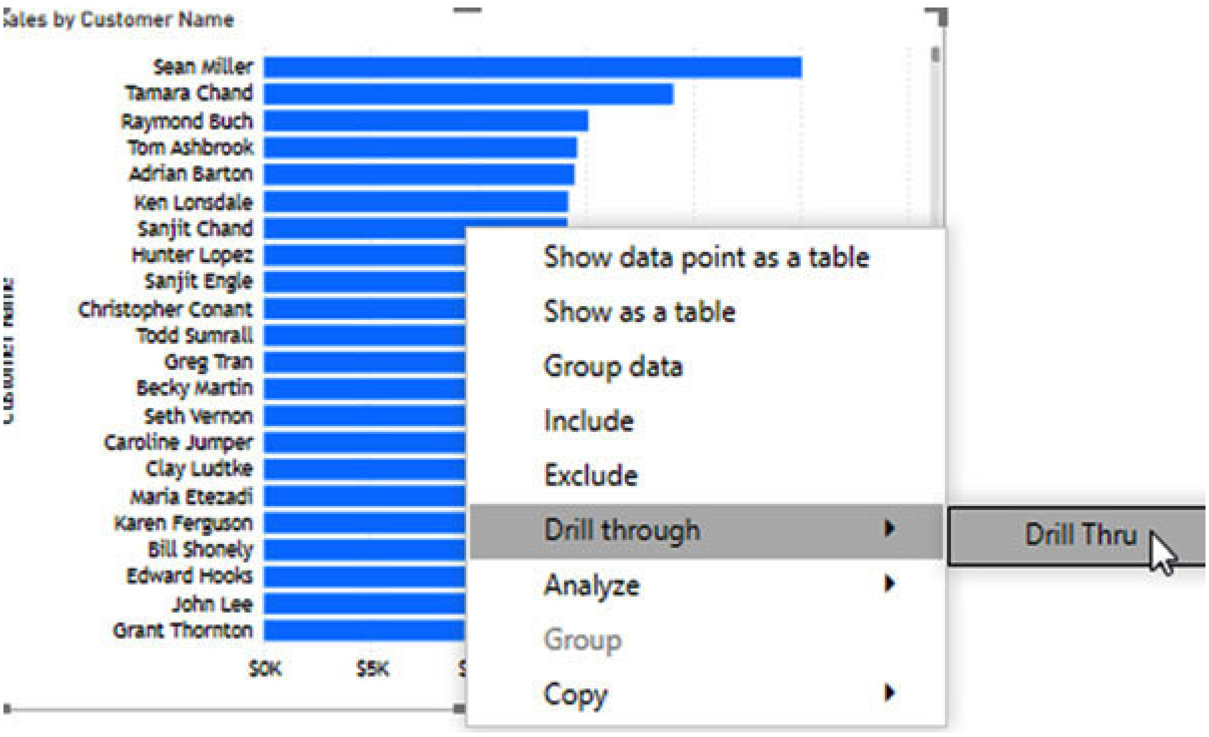
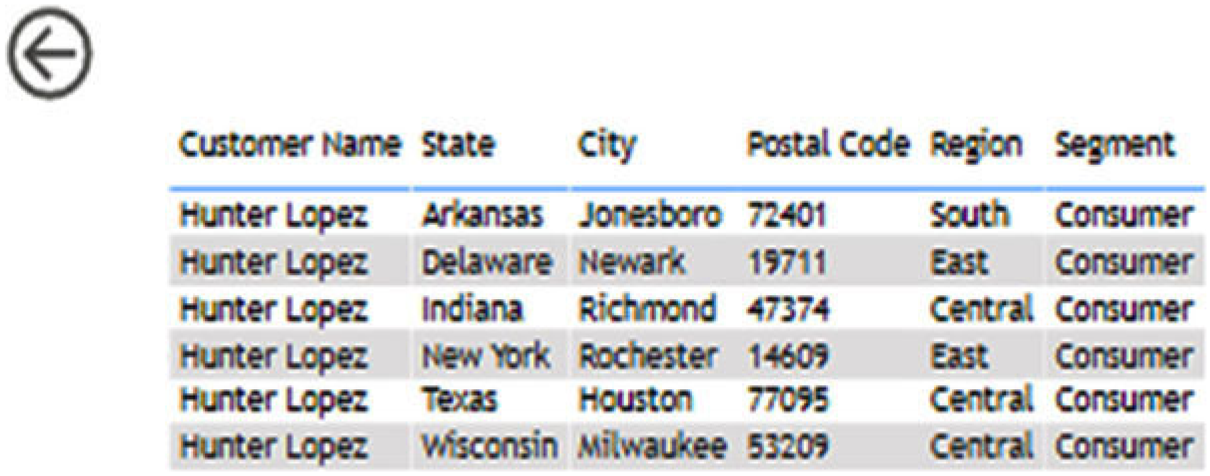


***Figure 5.40:*** *Performing drill through*

This will take you to the drill page and display the details of the customers selected in the first page, as shown in the following screenshot:



| Customer Name | State | City | Postal Code | Region | Segment |
|---|---|---|---|---|---|
| Hunter Lopez | Arkansas | Jonesboro | 72401 | South | Consumer |
| Hunter Lopez | Delaware | Newark | 19711 | East | Consumer |
| Hunter Lopez | Indiana | Richmond | 47374 | Central | Consumer |
| Hunter Lopez | New York | Rochester | 14609 | East | Consumer |
| Hunter Lopez | Texas | Houston | 77095 | Central | Consumer |
| Hunter Lopez | Wisconsin | Milwaukee | 53209 | Central | Consumer |

***Figure 5.41:*** *Details of Customer Hunter Lopez*

We can navigate back to the Summary page by using the back button.

## Conclusion

Visualizations are the most important part of any BI implementation, as the users get a quick insight into the data using the visualizations. In this chapter, we learned the different types of visualizations and when to use them. Every chart displays the data in a different way, but the overall aggregated data is the same. Visualizations are used to answer the business questions. For example, using the bar charts, we can get the Sales in different years and categories. Using the trend charts, we can get the Year trend of the sales.

We also learned some advanced charts, such as Waterfall and Ribbon. Power BI provides a ton of visualization options, but which chart is used, depends on what KPIs the users are interested in.

In the next chapter, we will learn how to deploy our application on the Power BI Service.

## ***Questions***

Which is the best visualization to use to display a single value?

In which scenarios will a bar chart be used?

Which chart should be used to display the Year-over-Year trend?

Which chart is used to display the ranking of the different data category?

Which chart is best suited to display the comparison between two measures?

Which chart is used to display the percentage of totals?

Which button automatically appears when the drill through functionality is used?

## ***Answers***

Card

To compare values of unique categories

Line chart

Ribbon

Scatter chart

Pie chart

Back button

CHAPTER 6

# Power BI Service

## *Introduction*

In the previous chapter, we got an in-depth knowledge on visualizations and understood how they are created. In this chapter, we will look into the Power BI service and its functionality. The Power BI service is a web application and is a **Software as a Service** component of the Power BI. In the Power BI service too, we can develop visualizations and share them with the other team members. A good knowledge of the Power BI service is essential for the developers and power users.

## *Structure*

In this chapter, we will discuss the following topics:

Understanding the Power BI Service

Building blocks of the Power BI Service

Creating visualizations in the Power BI Service

Creating Tiles and Dashboards

Publishing reports from the Power BI desktop

## ***Objectives***

The objective of this chapter is to understand the functionality of the Power BI Service and its various components. We will also learn how to connect to the data in the Power BI Service and create visualizations and dashboards.

## ***Understanding the Power BI Service***

As we understood in the earlier chapters, Power BI is a group of services, apps, and connectors that work together to create, share, and generate business insights.

Power BI Service, also referred to as Power BI online, and is provided as SaaS. The Power BI Service displays dashboards and reports that presents all the data in one place. In Power BI, we can connect to the data, create visuals, and share them with our team or we can consume the visualizations shared by others.

## ***Power BI Service workflow***

A typical scenario of using the Power BI Service would be as follows:

First, create a data model and the reports using the Power BI desktop.

Next, publish these reports to the Power BI Service.

Consume these published reports in the Power BI service and modify them as needed.

Create new reports in Power BI Service as needed.

And finally, create dashboards from reports in the Power BI Service.

Power BI provides a complete set of functionalities to create reports and dashboards.

## *Power BI Service interface*

Sign up to the Power BI Service from the Power BI desktop (from the sign up icon on the top) or use https://app.powerbi.com/ and sign in using your account credentials.

**To sign up for the Power BI Service, you will need a work or school email address. You will not be able to sign up using the email address like etc.**

Power BI Service provides all the capabilities to a user to connect to the data, and create visualization and dashboards. It is divided into the following different sections, as shown in the following figure:

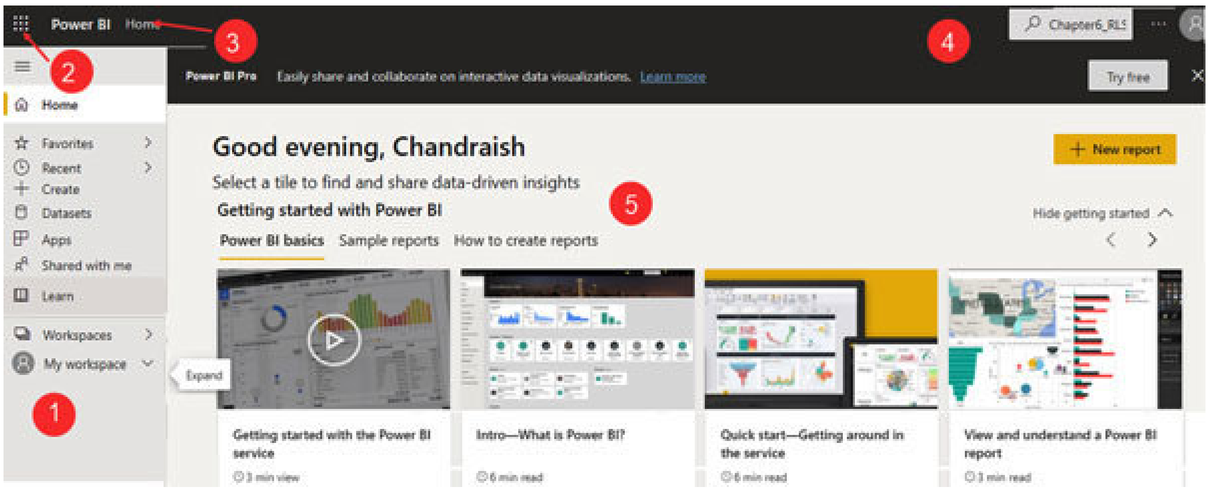


***Figure 6.1:*** *Power BI Service Interface*

#1 **Navigation pane:** In this pane, navigate to the favorites, recently opened content, published datasets, apps, and your workspace. My workspace is your personal workspace and holds your content.

#2 **Microsoft 365 app launcher:** This will help in launching the other Microsoft apps such as Outlook, PowerPoint, OneDrive, SharePoint, etc.

#3 **Home button:** This will let you navigate back to the Power BI Service home.

#4 **Search, settings, help, and feedback:** You can search for dashboards, view settings, and download.

#5 In this space, we will see the sample dashboards, favorites, frequently visited dashboards, reports, and workspace.

The Power BI Service comes with a comprehensive set of tools, in which you can explore a ton of options.

## ***Building blocks of the Power BI Service***

Power BI Service works on five important essentials, such as Dashboards, Reports, Workbooks, Datasets, and Dataflows. These essentials are categorized into workspaces, which are created on Capacities. We will look into each one of these in the following section:

**Capacities:** Capacity represents a set of resources such as storage, processor, and memory. It is used to host and deliver the Power BI content. There are two kinds of Capacities based on the client's need – shared or dedicated. By default, workspaces are created on a shared capacity.

**Workspaces:** Workspaces are created on capacities. They are the containers that hold all that you would need to create a Power BI application, such as datasets, data flows, reports, and dashboards. There are the following two types of workspaces:

**My** It is the personal workspace available to us to work with our own content. Only you have access to your workspace but you can share dashboards and reports with the other users.

In this workspace, we can collaborate and share the content such as dashboards, reports, etc. We can also add the other team members to our workspace, as long as they have the Power BI pro licenses.

**Dataflows:** Dataflows can only be created and managed in workspaces; they are not available in My Workspace. The main function of the dataflow is to combine the data from disparate sources. They are usually used in large implementations. The dataflow performs the data prep from the various data sources and makes the data available for the datasets. It is an example of the **Extract, Transform, and Load (ETL)** functionality in Power BI. Users can connect to the dataflows instead of going directly to the underlying data sources.

**Datasets:** A dataset is a collection of data that you can import or connect. In our example, a dataset is a collection of and other tables.

Dataset can also be sourced from a data flow. A single dataset can be part of multiple workspaces.

A single dataset can be used in multiple reports.

Visualization created from one dataset can be displayed in multiple dashboards.

To connect to a dataset, select **Get data** from the Similar to the Power BI Desktop, **Get data** is used in the Power BI Service to connect to data, as shown in the following figure:

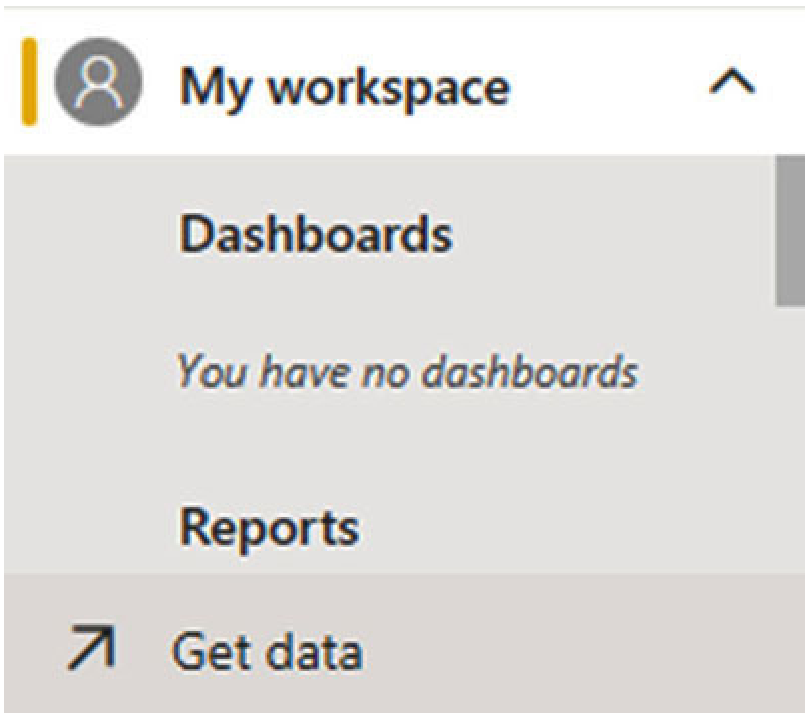


***Figure 6.2:*** *Adding datasets*

**Report:** We have seen how to create reports in the Power BI desktop. A Power BI report can be of one or more pages, containing one or more visualizations or visuals. All the visualization on a report comes from one dataset.

**Dashboards:** Dashboards are created in the Power BI service. A Power BI Dashboard is similar to a Car's dashboard, it provides all the useful information at a glance. It contains zero or more tiles. Each tile is pinned from a report. A report containing multiple pages can be pinned to a single tile.

One dashboard is associated to a single workspace.

It can display visualizations from different reports and datasets.

It can display visualizations pinned from the Excel workbook.

**Workbooks:** Using the Get Data functionality, connect to an Excel workbook. We can pin the elements from an Excel application into the dashboard.

The understanding of the preceding terms and concepts will help in using the Power BI Service efficiently.

## *Creating visualizations in the Power BI Service*

Similar to the Power BI desktop, we can create the different types of visuals in Power BI Service.

The Power BI Service doesn't provide the strong data modeling capabilities since it is a web-based interface, but you can still connect to the data and design visuals.

## ***Get data***

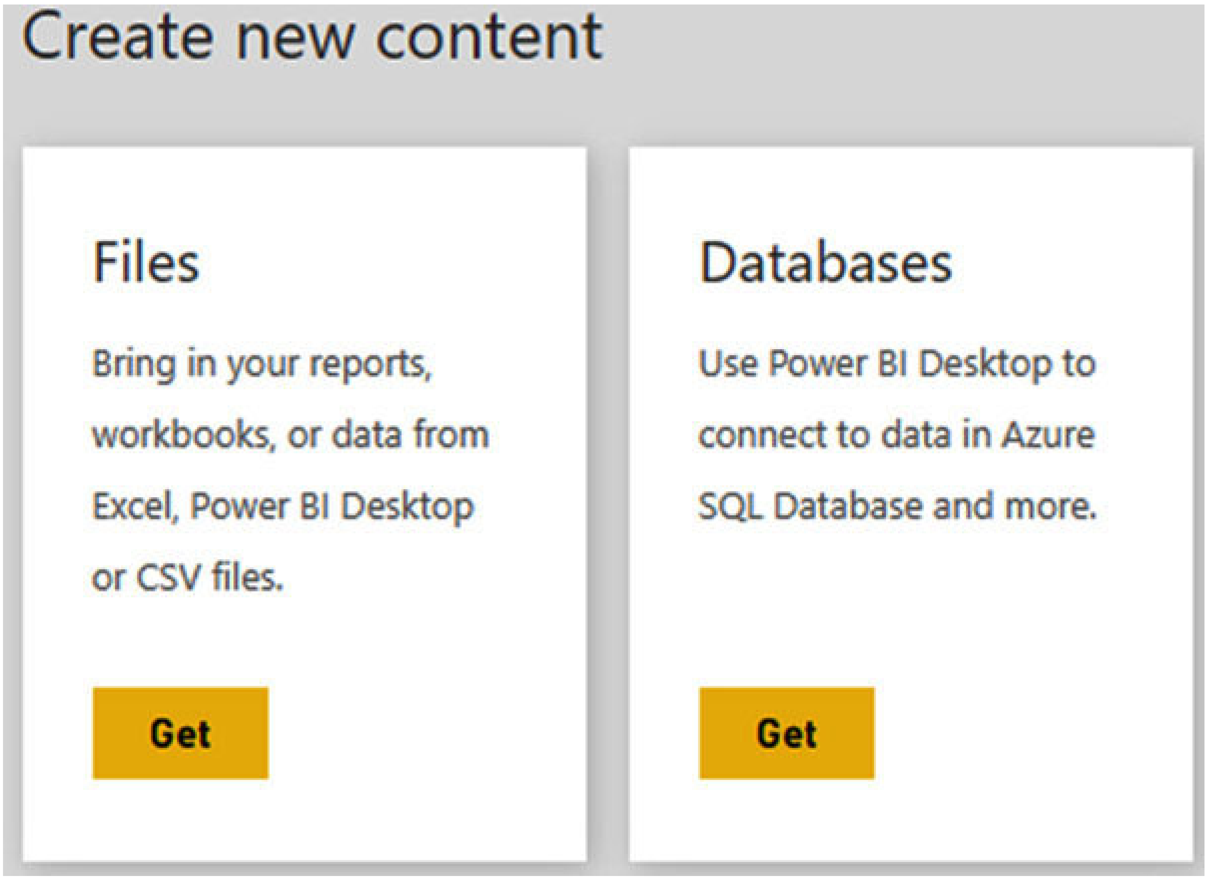
The steps to connect to the data using Get Data are as follows:

Launch the Power BI Service; sign in using the Power BI desktop or use https://app.powerbi.com/ and sign in using your account credentials.

From the navigation pane on the left, from **My** select **Upload a** You can also scroll down and select **Get**

From the **Get Data** screen, under **Create new** in the **Files** box, click on

Create new content using We can connect to **Files** based on data, such as the Excel or CSV files or databases, as shown in the following screenshot:



Create new content

Files

Bring in your reports, workbooks, or data from Excel, Power BI Desktop or CSV files.

Databases

Use Power BI Desktop to connect to data in Azure SQL Database and more.

***Figure 6.3:*** *Create new content using Files*

This will provide all the different options for the file-based data selection.

The options are available to select the different data such as Local file, OneDrive, SharePoint. Help on these data files can also be accessed from 'Learn about importing files', as shown in the following screenshot:

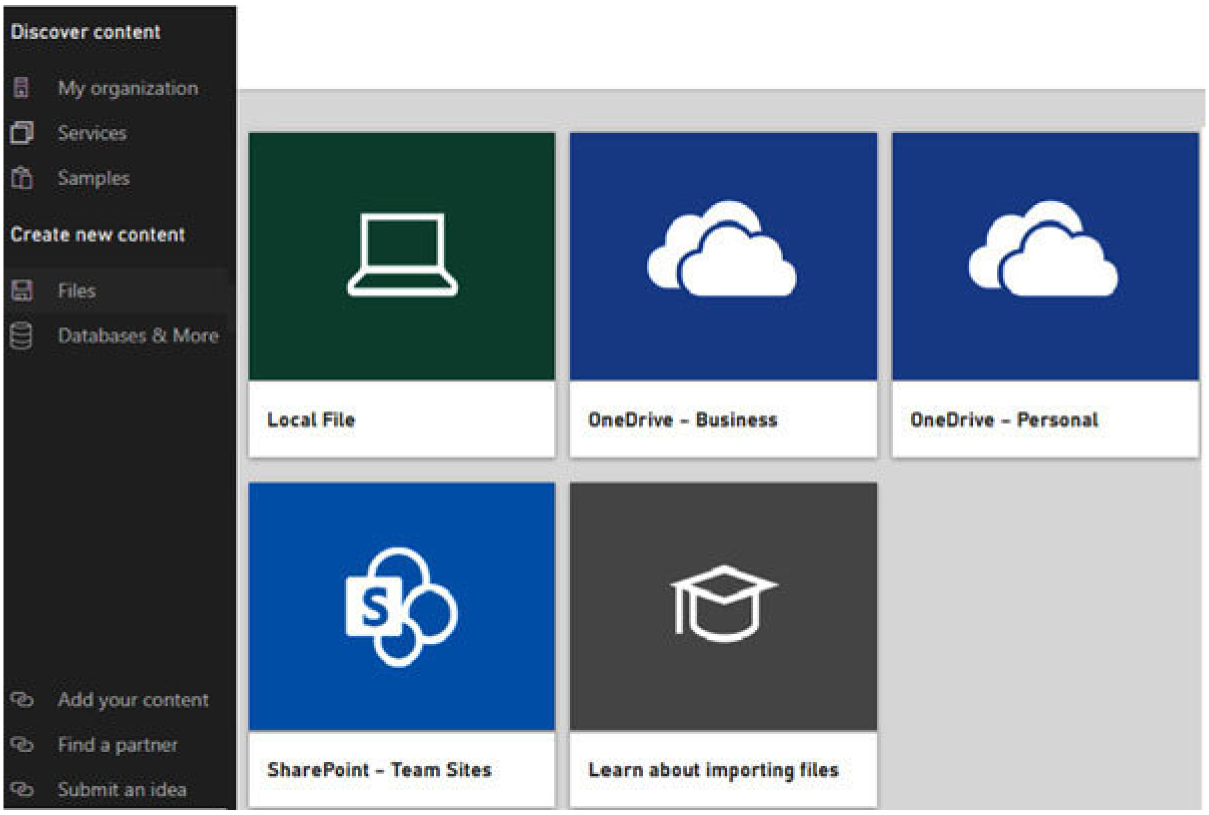


***Figure 6.4:*** *Options to select different data Files*

Select **Local File** and browse to your enclosed files Select

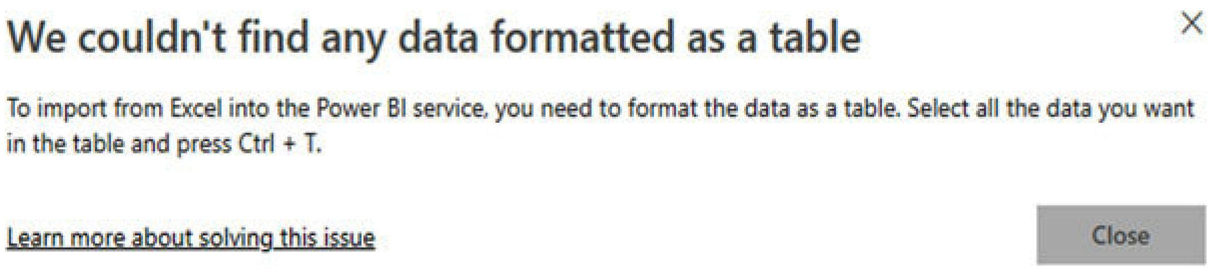
We will see the following two options in the Power BI Service:

Import Excel data into Power BI: Use this option to import data and create visualizations.

Upload your Excel file to Power BI: This option will open the entire Excel workbook which can be edited using the Excel online.

Under **Import Excel** data into Power BI, select

We will get an error, as shown in the following figure:


We couldn't find any data formatted as a table

To import from Excel into the Power BI service, you need to format the data as a table. Select all the data you want in the table and press Ctrl + T.

Learn more about solving this issue

***Figure 6.5:*** *Error in importing excel data*

This error has occurred because the Power BI service expects the XLS data to be a **Table** and our **Orders.xls** data is not created as a

**Resolution:** To resolve this issue, open

Select the range of rows containing the data, and make sure the first row selected contains the column headers.

From the **Insert** ribbon, select **Table** or use *Ctrl* + as shown in the following figure:

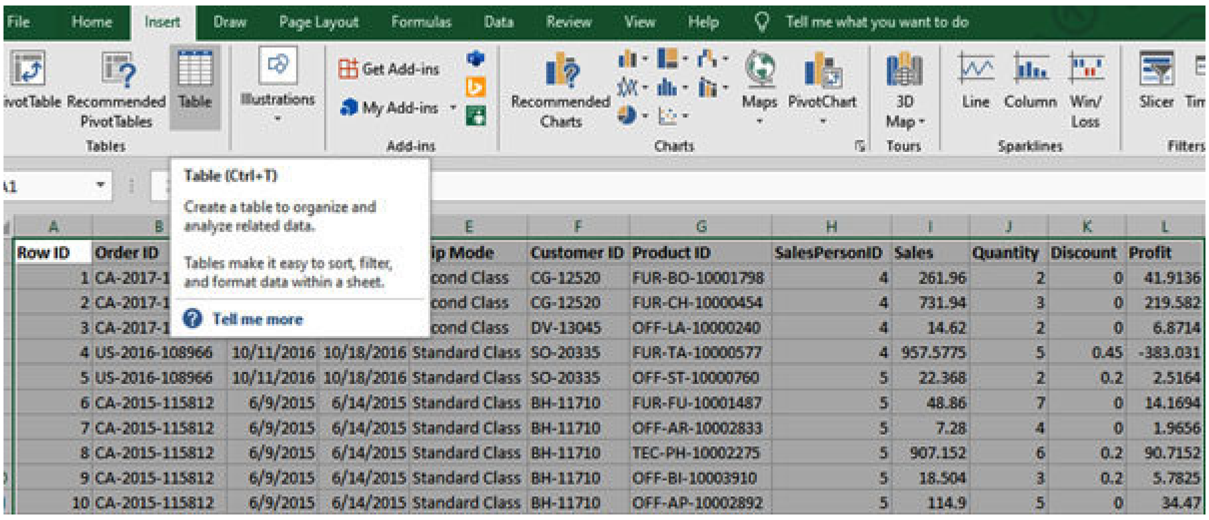


Table (Ctrl+T)

Create a table to organize and analyze related data.

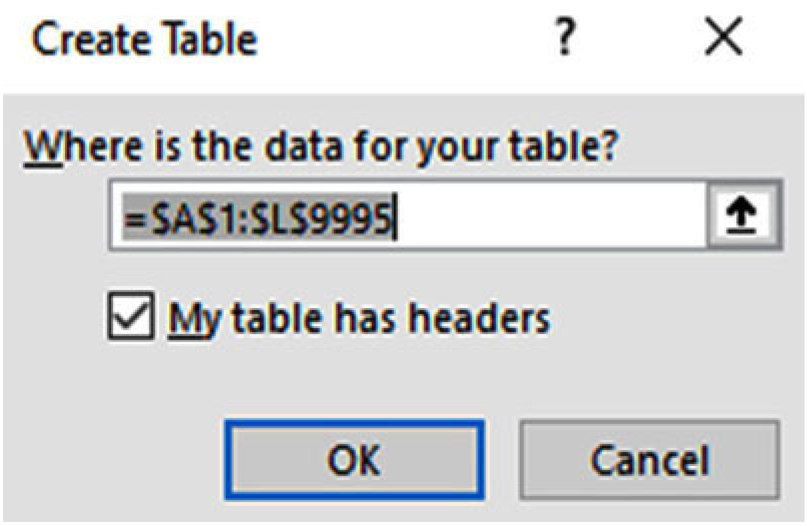
Tables make it easy to sort, filter, and format data within a sheet.

Tell me more

| Row ID | Order ID | | | ip Mode | Customer ID | Product ID | SalesPersonID | Sales | Quantity | Discount | Profit |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | CA-2017-1 | | | cond Class | CG-12520 | FUR-BO-10001798 | 4 | 261.96 | 2 | 0 | 41.9136 |
| 2 | CA-2017-1 | | | cond Class | CG-12520 | FUR-CH-10000454 | 4 | 731.94 | 3 | 0 | 219.582 |
| 3 | CA-2017-1 | | | cond Class | DV-13045 | OFF-LA-10000240 | 4 | 14.62 | 2 | 0 | 6.8714 |
| 4 | US-2016-108966 | 10/11/2016 | 10/18/2016 | Standard Class | SO-20335 | FUR-TA-10000577 | 4 | 957.5775 | 5 | 0.45 | -383.031 |
| 5 | US-2016-108966 | 10/11/2016 | 10/18/2016 | Standard Class | SO-20335 | OFF-ST-10000760 | 5 | 22.368 | 2 | 0.2 | 2.5164 |
| 6 | CA-2015-115812 | 6/9/2015 | 6/14/2015 | Standard Class | BH-11710 | FUR-FU-10001487 | 5 | 48.86 | 7 | 0 | 14.1694 |
| 7 | CA-2015-115812 | 6/9/2015 | 6/14/2015 | Standard Class | BH-11710 | OFF-AR-10002833 | 5 | 7.28 | 4 | 0 | 1.9656 |
| 8 | CA-2015-115812 | 6/9/2015 | 6/14/2015 | Standard Class | BH-11710 | TEC-PH-10002275 | 5 | 907.152 | 6 | 0.2 | 90.7152 |
| 9 | CA-2015-115812 | 6/9/2015 | 6/14/2015 | Standard Class | BH-11710 | OFF-BI-10003910 | 5 | 18.504 | 3 | 0.2 | 5.7825 |
| 10 | CA-2015-115812 | 6/9/2015 | 6/14/2015 | Standard Class | BH-11710 | OFF-AP-10002892 | 5 | 114.9 | 5 | 0 | 34.47 |

***Figure 6.6:*** *Converting excel data to Table*

We will get a dialog box showing the range of cells in the table. Make sure to check on **My table has headers** and select as shown in the following screenshot:



Create Table

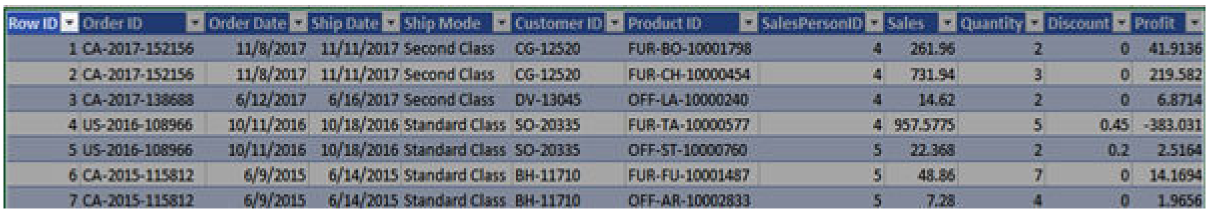
Where is the data for your table?

=$A$1:$L$9995

☑ My table has headers

***Figure 6.7:*** *Table Range*

Now the data is formatted as a as shown in the following screenshot:



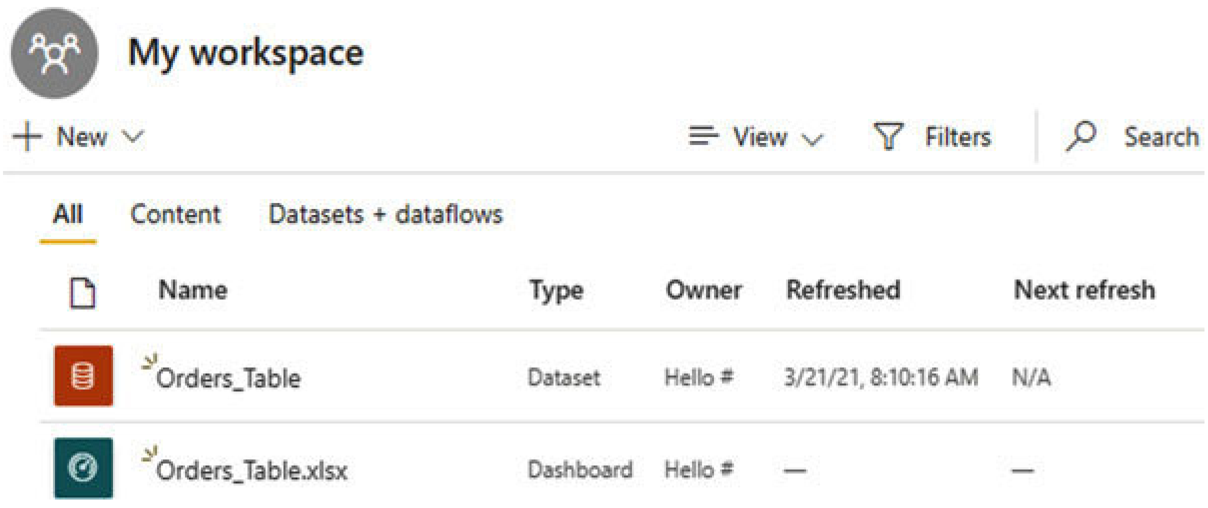
| Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Product ID | SalesPersonID | Sales | Quantity | Discount | Profit |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | CA-2017-152156 | 11/8/2017 | 11/11/2017 | Second Class | CG-12520 | FUR-BO-10001798 | 4 | 261.96 | 2 | 0 | 41.9136 |
| 2 | CA-2017-152156 | 11/8/2017 | 11/11/2017 | Second Class | CG-12520 | FUR-CH-10000454 | 4 | 731.94 | 3 | 0 | 219.582 |
| 3 | CA-2017-138688 | 6/12/2017 | 6/16/2017 | Second Class | DV-13045 | OFF-LA-10000240 | 4 | 14.62 | 2 | 0 | 6.8714 |
| 4 | US-2016-108966 | 10/11/2016 | 10/18/2016 | Standard Class | SO-20335 | FUR-TA-10000577 | 4 | 957.5775 | 5 | 0.45 | -383.031 |
| 5 | US-2016-108966 | 10/11/2016 | 10/18/2016 | Standard Class | SO-20335 | OFF-ST-10000760 | 5 | 22.368 | 2 | 0.2 | 2.5164 |
| 6 | CA-2015-115812 | 6/9/2015 | 6/14/2015 | Standard Class | BH-11710 | FUR-FU-10001487 | 5 | 48.86 | 7 | 0 | 14.1694 |
| 7 | CA-2015-115812 | 6/9/2015 | 6/14/2015 | Standard Class | BH-11710 | OFF-AR-10002833 | 5 | 7.28 | 4 | 0 | 1.9656 |

***Figure 6.8:*** *Excel data formatted as Table*

We can save this XLS as only to differentiate it from

Navigate back to the Power BI Service and follow the steps to import the Excel file, this time select

This will successfully load your XLS file. After connecting to Power BI will display the loaded Dataset, as shown in the following screenshot:



| Name | Type | Owner | Refreshed | Next refresh |
|---|---|---|---|---|
| Orders_Table | Dataset | Hello # | 3/21/21, 8:10:16 AM | N/A |
| Orders_Table.xlsx | Dashboard | Hello # | — | — |

***Figure 6.9:*** *Dataset loaded in Power BI*

We can see that the **Orders_Table** Dataset is loaded. A sample **Orders_Table.xlsx** dashboard is also created which is empty right now.

The steps to create a Report are as follows:

Now we can create report from the preceding loaded dataset.

Click on the **More** options next to the dataset and select **Create** as shown in the following screenshot:

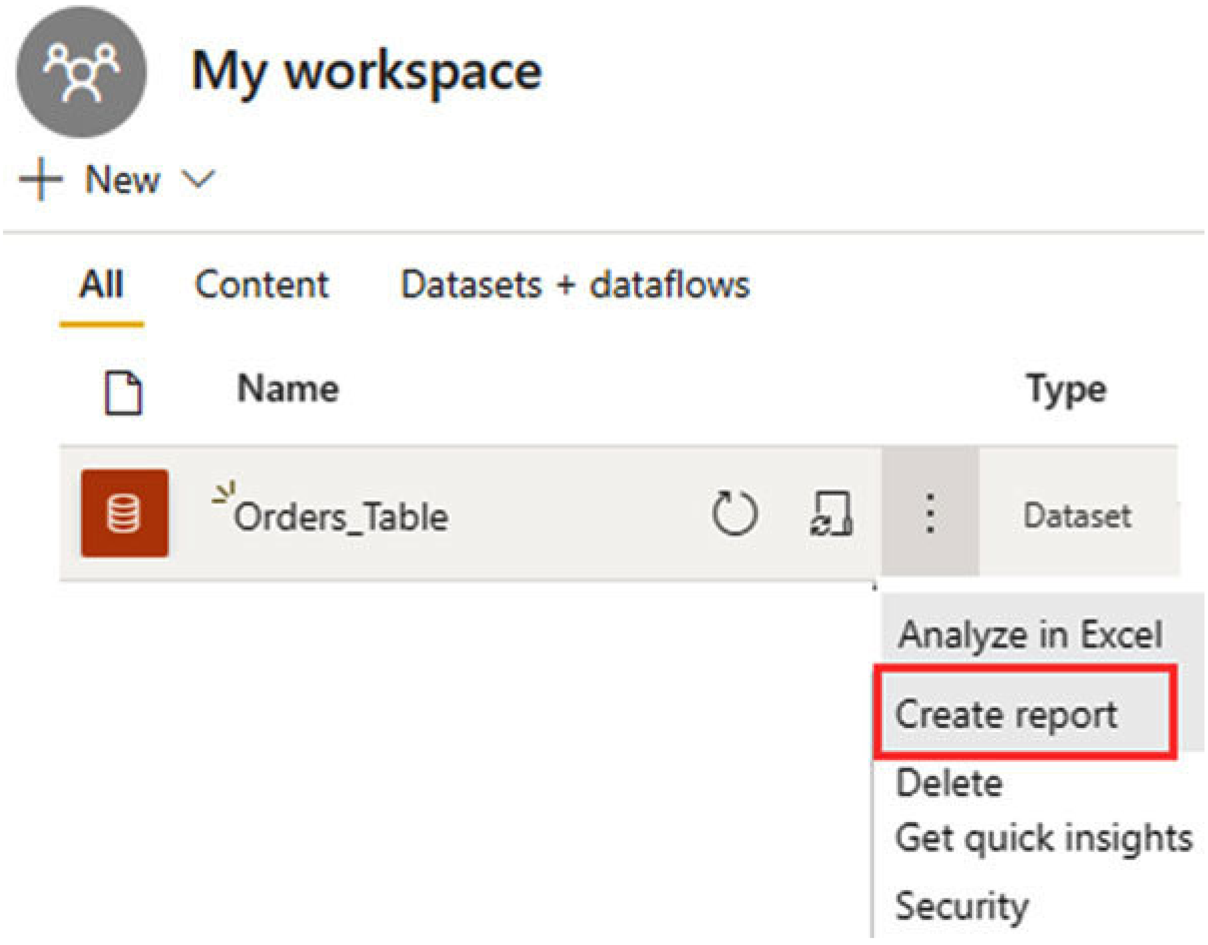


***Figure 6.10:*** *Create report selection*

The interface is similar to the Power BI desktop report view with Filters, Visualization, and fields.

Select the stacked bar chart from the visualizations. Select **Order Year** and **Sales** from the list of fields.

The fields may appear slightly different as this is **Orders_Table** loaded in the Power BI Service and not coming from the Power BI desktop where we had Query Editor to modify the table. **Order Year** and **Order Month** appear as separate fields as compared to the Power BI desktop where they appeared as the hierarchical data elements under **Order**

This will create a bar chart for **Year** and We can format the chart according to our preference. If you need more canvas space, click on the navigation pane on the left to hide the navigation pane.

The steps to create a Bar chart in the Power BI Service are similar to the bar chart created in the Power BI desktop. If both the charts connect to the same data and displays the same data elements, then the charts in both the applications will look the same, as shown in the following screenshot:

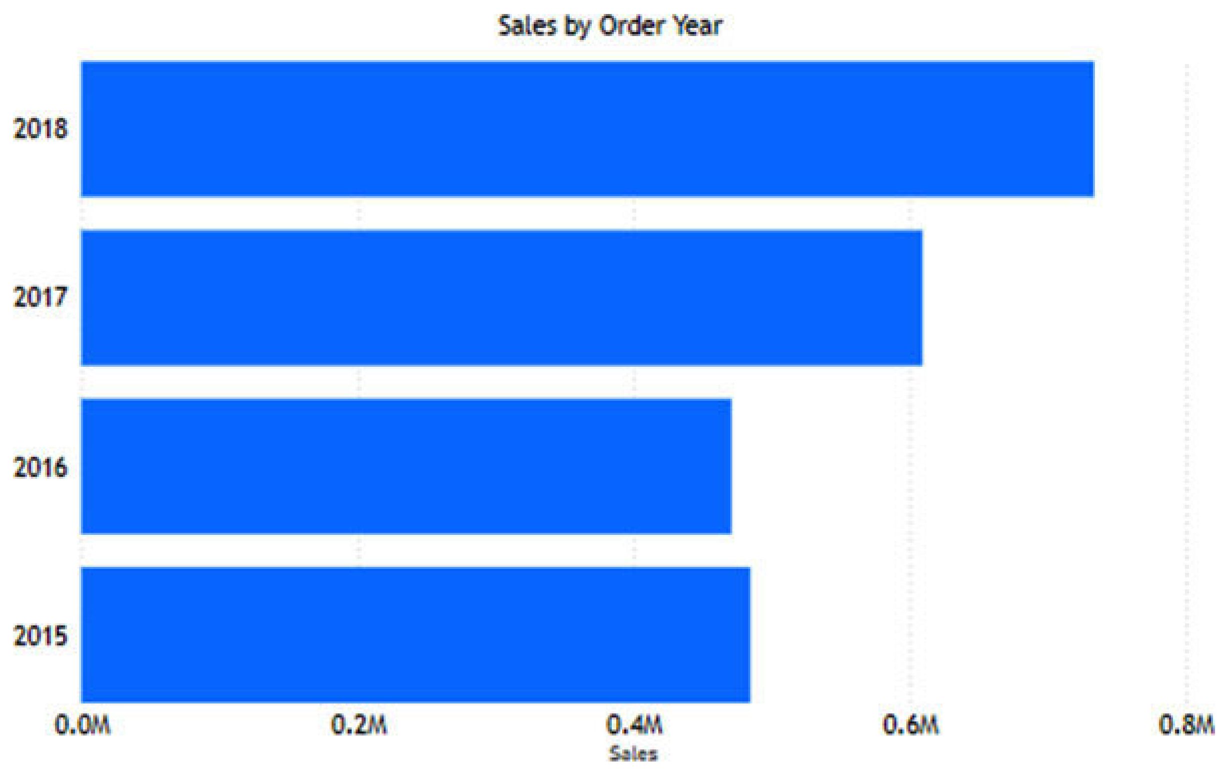


***Figure 6.11:*** *Bar chart in Power BI Service*

Select the **Save** option from the extreme right. Save your report as **Sales by**

The menu at the top of the chart shows **Reading** this is a contextual button. Since you are in the editing mode (as you are the owner of this report), it displays the Reading view. Once you share report with others, it will show them in the Reading view, that is, they can only view it and cannot modify.

The steps to create a Dashboard are as follows:

We have created a visual in the previous section, we will now pin this visual to create a dashboard:

Hover over the chart and select the pin icon.

A dashboard is created by pinning the visualization. Multiple visualizations can be pinned to create a single dashboard, as shown in the following screenshot:

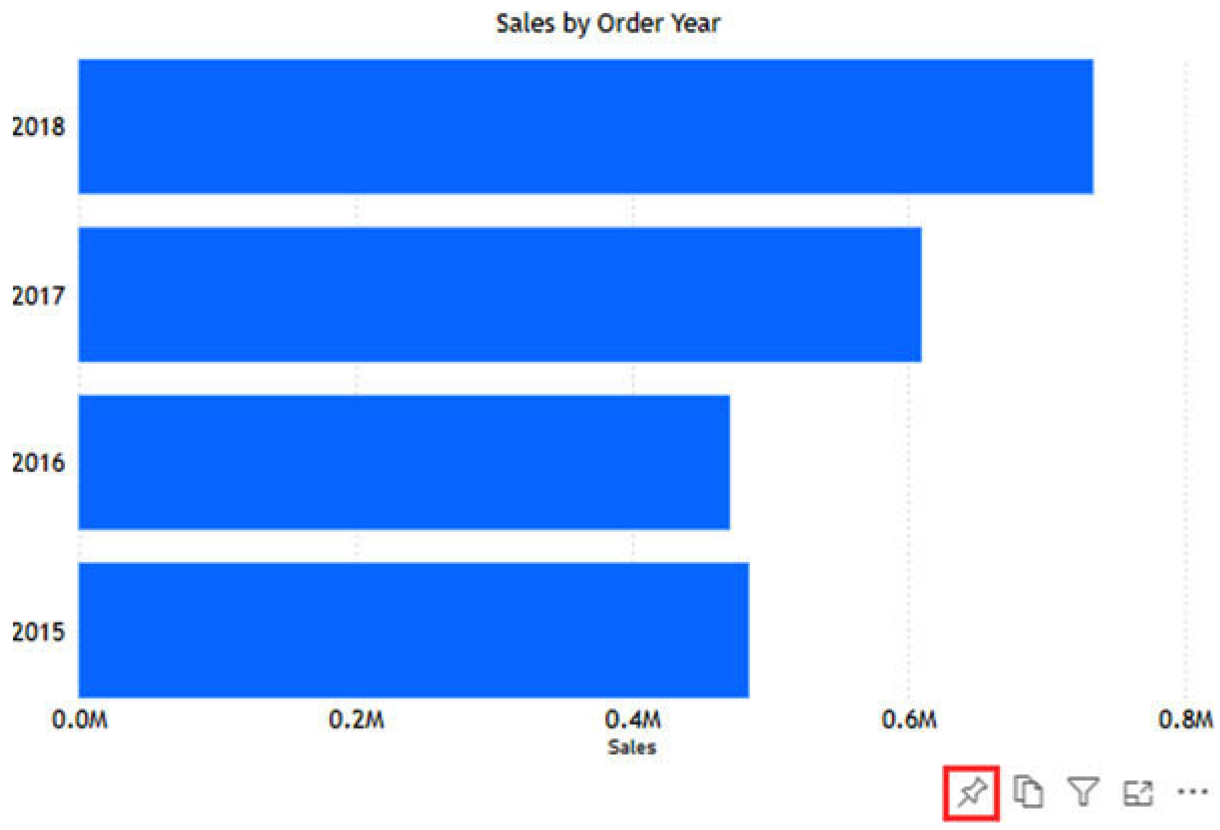
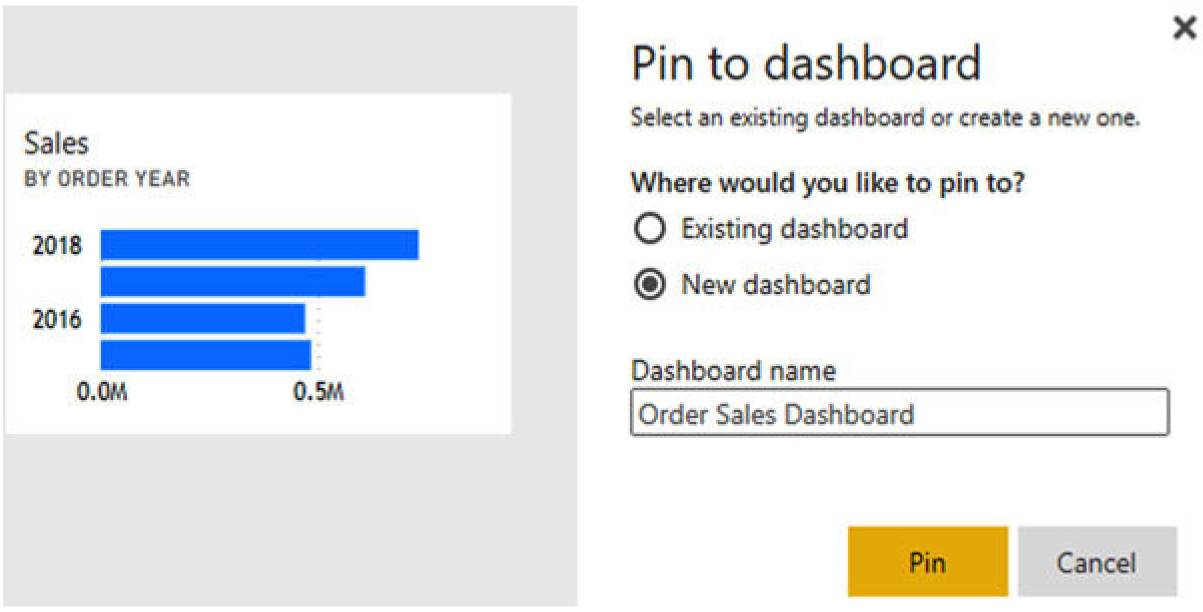


***Figure 6.12:*** *Pinning the visual*

A Pin to dashboard dialog box will appear. Select **New dashboard** and provide a name to the dashboard and select as shown in the following screenshot:



Pin to dashboard

Select an existing dashboard or create a new one.

Where would you like to pin to?

○ Existing dashboard

● New dashboard

Dashboard name

Order Sales Dashboard

***Figure 6.13:*** *Creating a dashboard from the visual*

We will get a success dialog box displaying that visualization has been pinned to the dashboard. It will display the following two options:

Create phone view, **Go to**

Select **Go to**

You can see the dashboard under **My Workspace** too, as shown in the following screenshot:

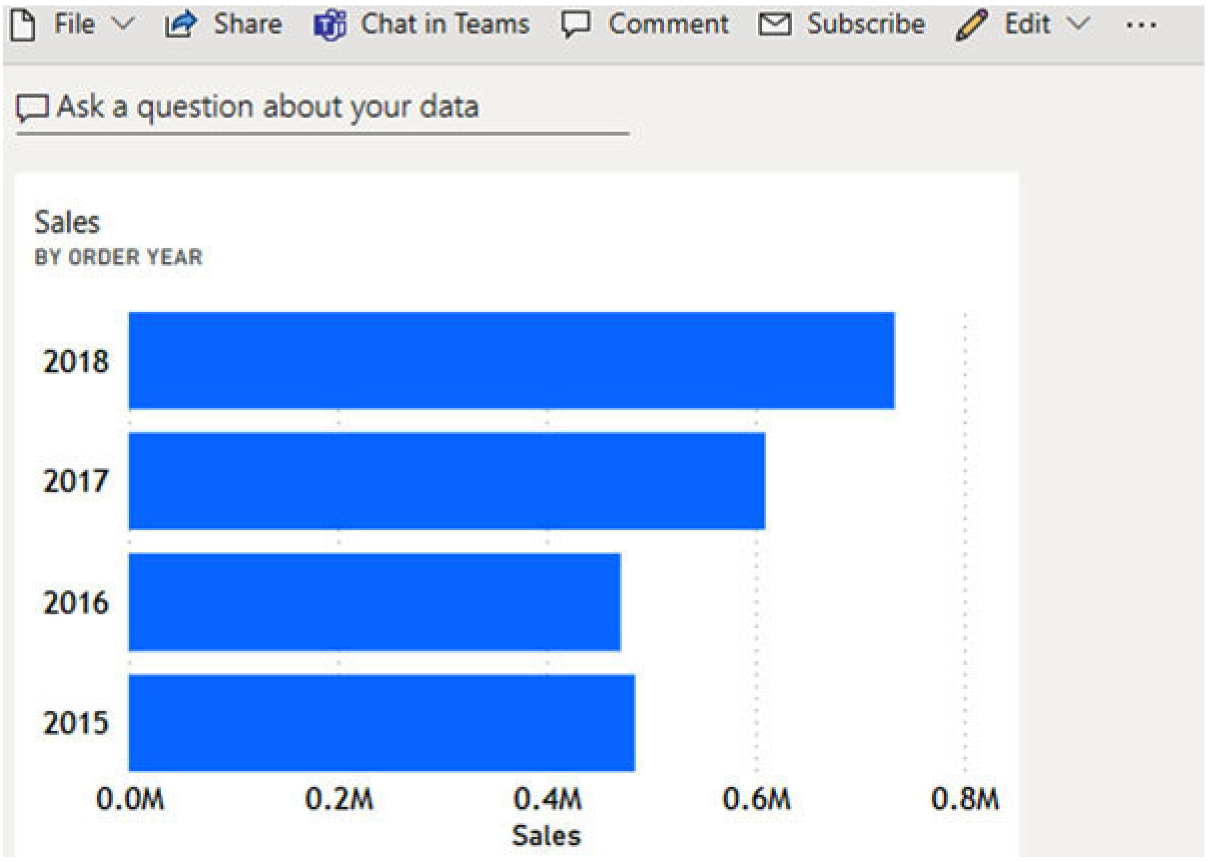


***Figure 6.14:** Dashboard*

The menu on the dashboard provides the options to share/collaborate with your team or add a tile.

Use the **Edit** option to **Add a** Adding a tile provides the options to add the web content, image, text box, or a video to the dashboard.

The steps to create another Report are as follows:

Now we will create a simple line chart from the loaded dataset **Orders_Table** and pin it to the dashboard.

Navigate to **My Workspace** and click on the **More** options next to the dataset and select **Create**

From the visualizations, select **Line** Select **Order Year** and Do the required formatting. You will get a line chart showing the trend of Sales over the years, as shown in the following screenshot:

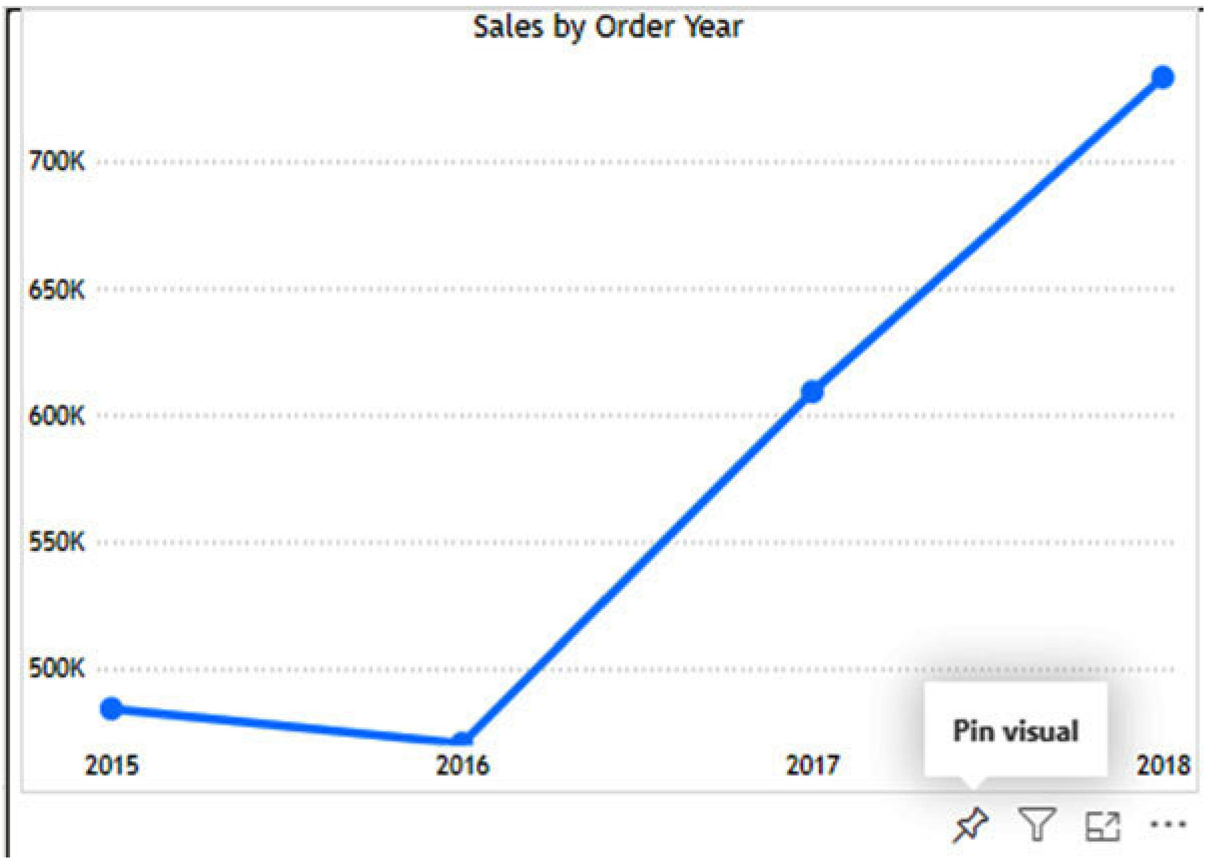
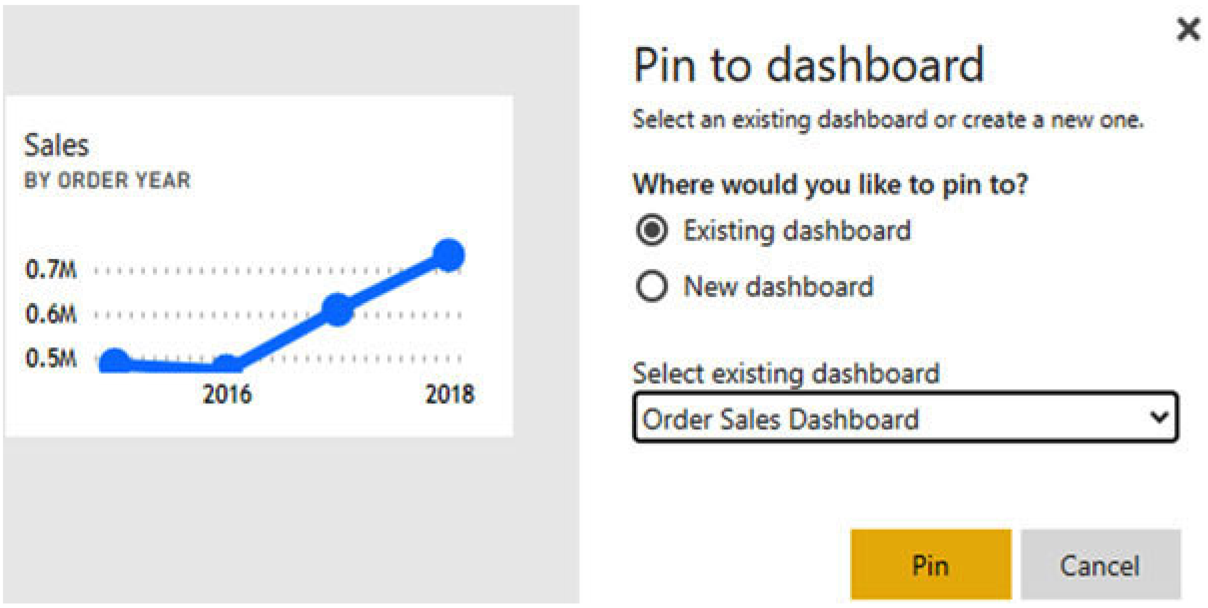


***Figure 6.15:*** *Trend of Sales by Year*

Save the report by clicking on the save icon on the extreme right. We can name the report as **Trend of Sales by**

Hover over the chart and select **Pin** Select the existing dashboard and select as shown in the following screenshot:



Pin to dashboard

Select an existing dashboard or create a new one.

Where would you like to pin to?

[✓ Existing dashboard]

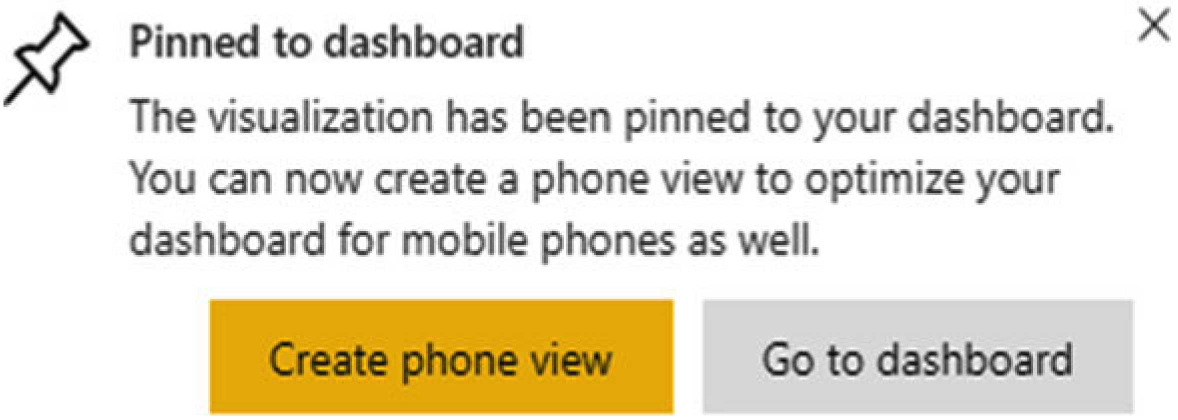
New dashboard

Select existing dashboard: Order Sales Dashboard

Pin | Cancel

***Figure 6.16:*** *Pin Line chart to the existing dashboard*

A dialog box will appear displaying that the visualization has been pinned. Select **Go to** as shown in the following screenshot:



Pinned to dashboard

The visualization has been pinned to your dashboard. You can now create a phone view to optimize your dashboard for mobile phones as well.

Create phone view | Go to dashboard

***Figure 6.17:*** *Visualization successfully pinned.*

We will be navigated to the dashboard, where the two tiles on the dashboard are displayed, one bar chart and one line chart.

If at any time we wish to delete a tile, hover over the chart and select the **More** options and select **Delete** as shown in the following screenshot:

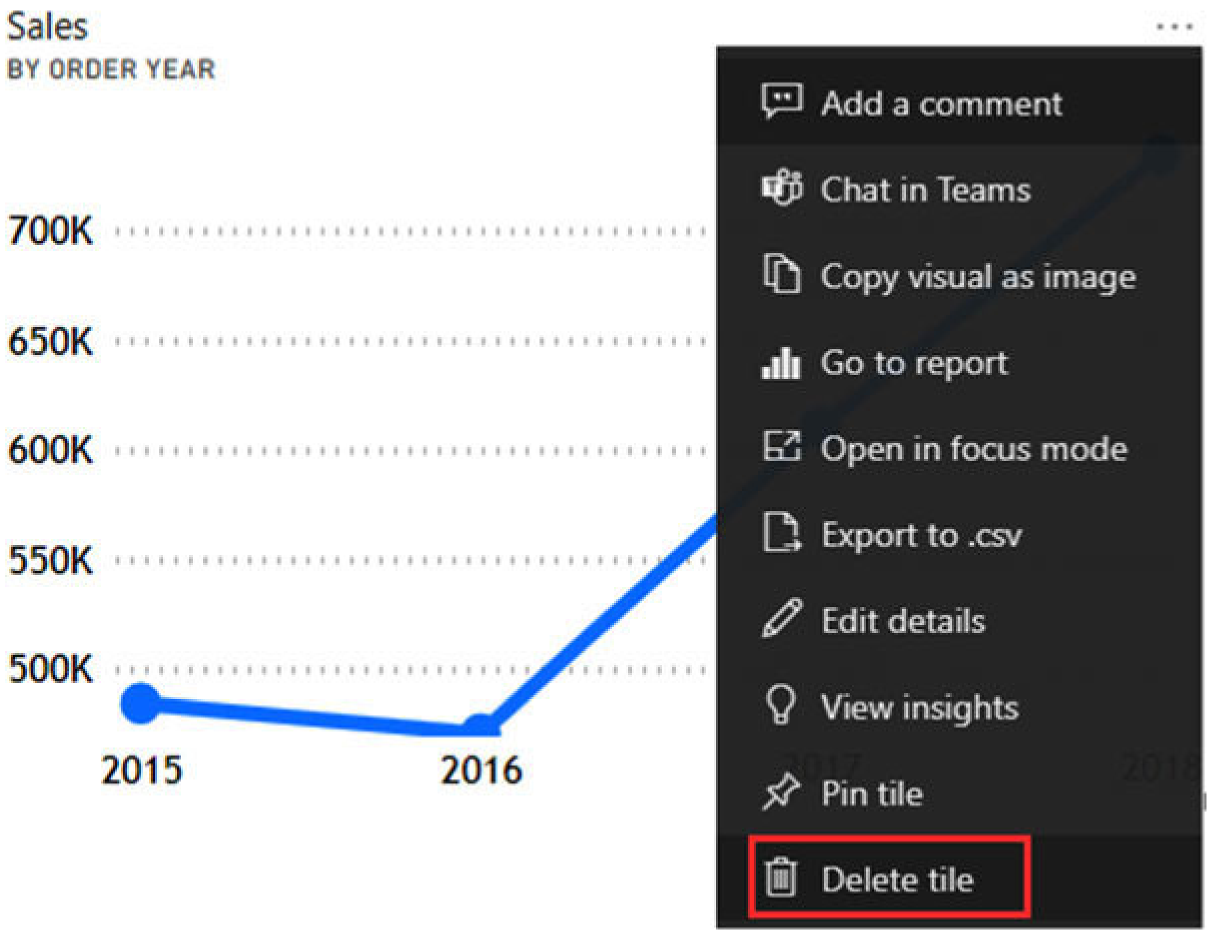


***Figure 6.18:*** *Deleting a tile*

We can also move a tile around by simply selecting it and moving around. The dashboard will appear as shown in the following figure:

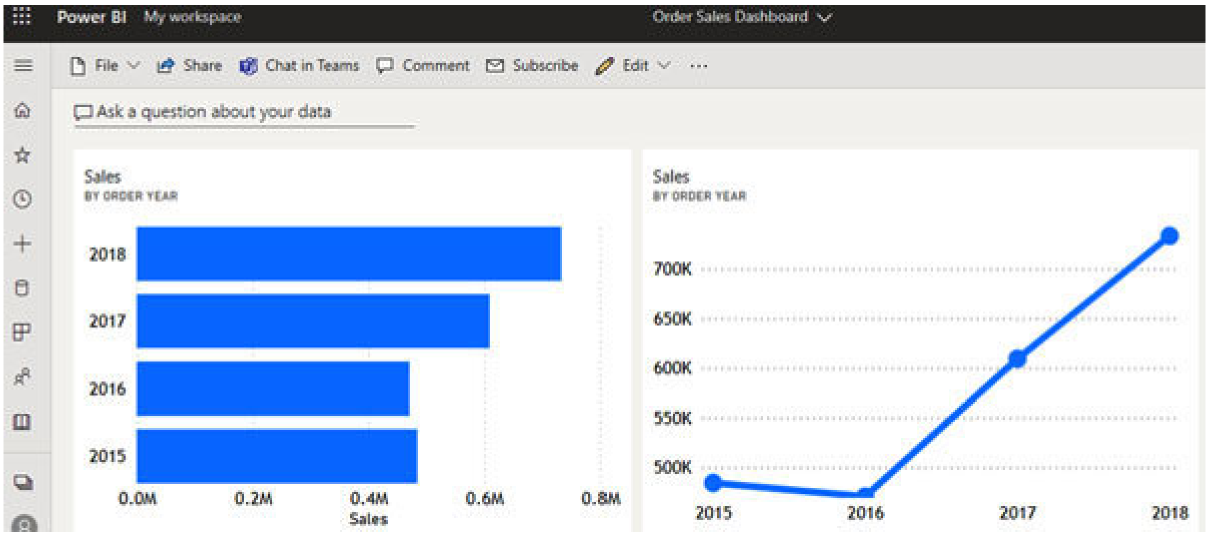


***Figure 6.19:*** *Final Dashboard*

We can have any number of tiles on a dashboard. Make sure the information is related, so that it presents useful information to the users with whom the dashboard will be shared.

If we select My Workspace on the navigation pane, a list of our datasets, reports, and dashboards will be displayed.

## Publishing reports from the Power BI desktop

Reports developed in the Power BI desktop can be published to the Power BI Service. Reports get published with the dataset.

The steps to publish a report from the Power BI desktop are as follows:

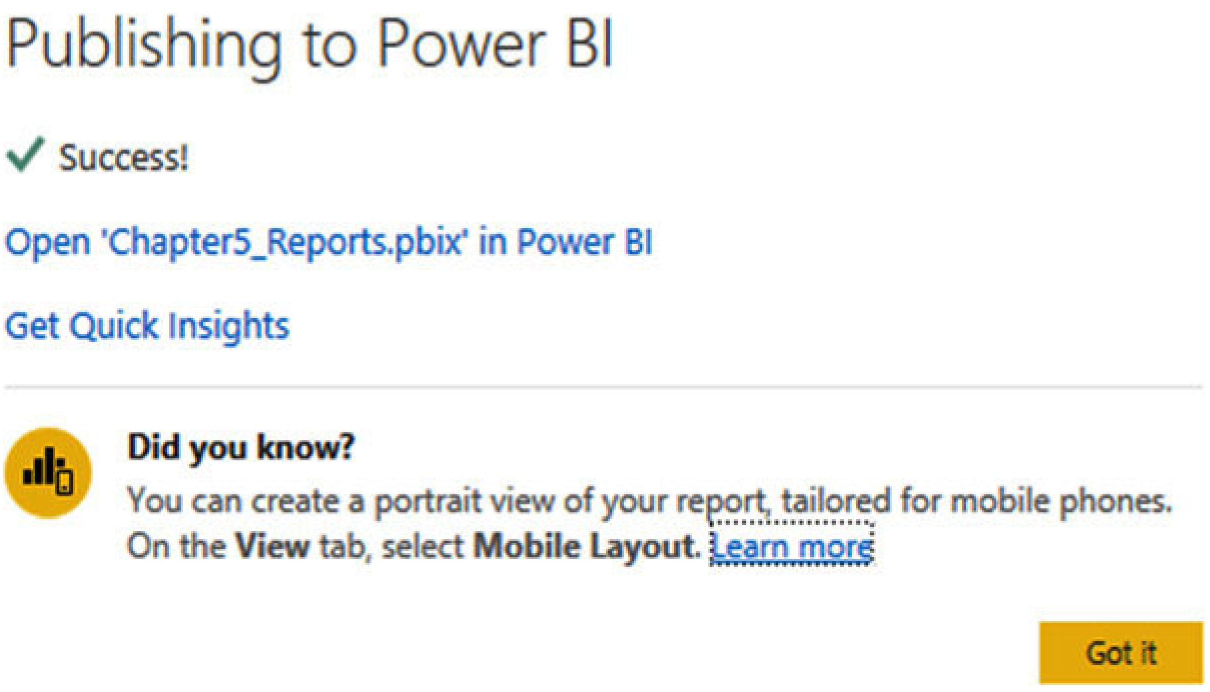
Navigate to the Power BI desktop, where you have created the visualizations. If you have followed all the chapters in this book, then use

Sign in to your account from the **Sign In** icon at the top.

In the ribbon, navigate to **Home** and select

A dialog box for selecting a workspace will be displayed. Select **My**

You will get a success dialog box. Multiple reports developed using the Power BI desktop can be published to the Power BI service, as shown in the following screenshot:


Publishing to Power BI

✓ Success!

Open 'Chapter5_Reports.pbix' in Power BI

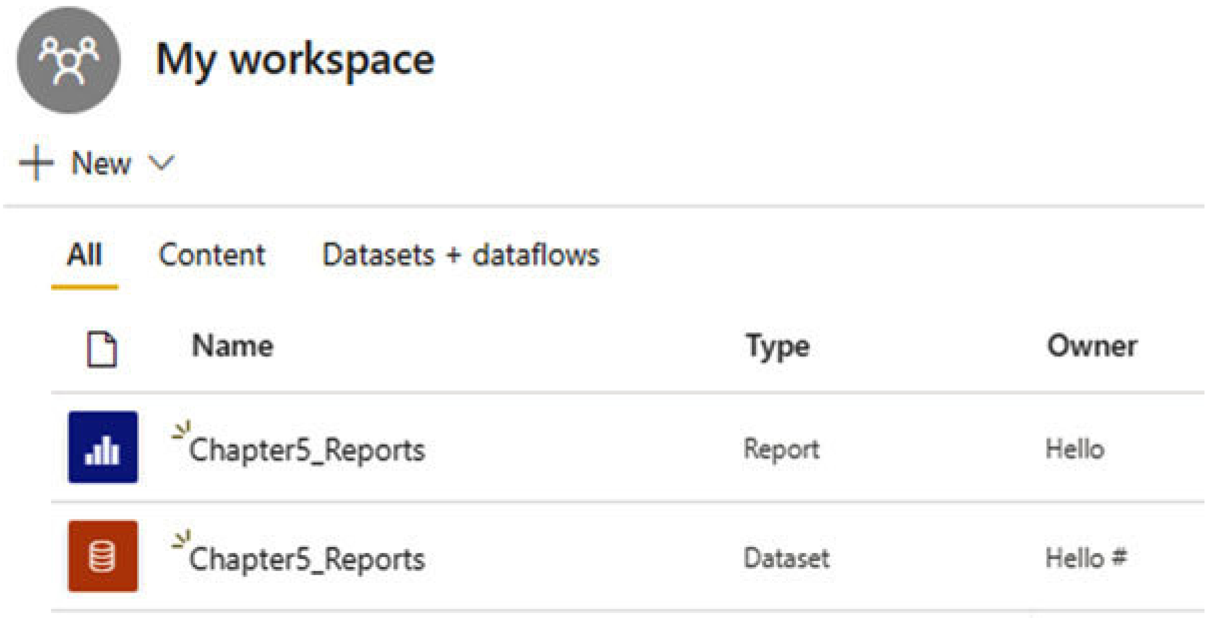
Get Quick Insights

**Did you know?**

You can create a portrait view of your report, tailored for mobile phones. On the **View** tab, select **Mobile Layout**. Learn more

***Figure 6.20:*** *Publishing Chapter5_Reports*

Navigate to the Power BI service, and go to **My** You will see your report and dataset loaded, as shown in the following screenshot:


My workspace

| Name | Type | Owner |
|---|---|---|
| Chapter5_Reports | Report | Hello |
| Chapter5_Reports | Dataset | Hello # |

***Figure 6.21:*** *Report and dataset published*

Click on **Chapter5_Reports** which is of **Type** Report. It will provide the list of pages in the report.

Click on any page and pin it to the new or existing dashboard by completing the steps mentioned in the earlier sections, as shown in the following screenshot:

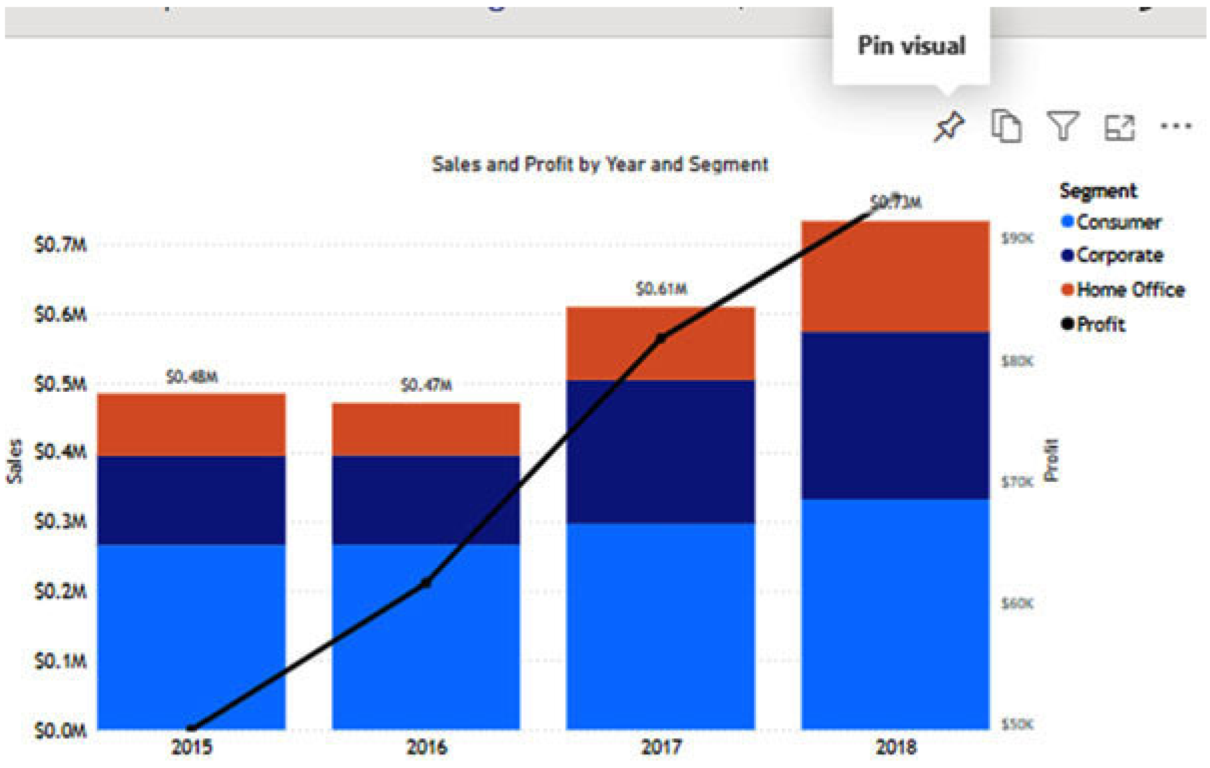


***Figure 6.22:*** *Page from the published report*

By following the preceding steps, we can publish reports from the Power BI desktop to the Power BI service, create tiles and dashboards, and share them with the users.

## Conclusion

In this chapter, we got familiarized with the Power BI Service. It is an important application for publishing, developing, and sharing the content from the Power BI desktop. Workspace in the Power BI service is a useful concept. The users have their own personal workspace where they can develop the content and share with the other users. The Development environment of the Power BI Service is similar to the Power BI desktop; the only difference is that we can create dashboards in the Power BI Service. A dashboard is made of one or multiple sheets.

In the next chapter, we will see how to secure a Power BI application by implementing a row-level security.

## ***Questions***

What is a Power BI Service and how does it differ from the Power BI desktop?

What are the important components of the Power BI Service?

What is the difference between Dataflow and Dataset?

How are dashboards created?

Can we create dashboards in the Power BI desktop?

## ***Answers***

Power BI service is a Software as a Service (SaaS) component of Power BI. It provides development and collaboration functionalities. A Development environment is similar to the Power BI desktop but it does not have Query Editor and DAX editor as in the Power BI desktop.

Important components of the Power BI Service are dashboards, Reports, Workbooks, Datasets, and Dataflows.

Dataflow is the Extract, Transform, and Load (ETL) component of Power BI. It extracts the data from the datasets and performs ETL. It unifies the data from the disparate data sources. The users need not understand the complexity of the underlying datasets, instead they can connect to the dataflow and design their visualizations.

Dashboards are created by pinning the visualizations in a report.

Dashboards BI Service.

CHAPTER 7

# Securing Your Application

## ***Introduction***

In the previous chapter, we got an in-depth knowledge on visualizations and how they are created. In this chapter, we will learn how to secure the application using the row-level security in Power BI. The data model and visuals that we created in the previous chapter will be used to explain the security concept.

It’s important to learn about security and implementation because it saves your application from unauthorized access, and the users see only the relevant data.

## ***Structure***

In this chapter, we will discuss the following topics:

Security concepts

Managing and creating roles in Power BI desktop

Creating the DAX filter to implement RLS

Configuring roles in the Power BI Service

## ***Objectives***

After studying this chapter, we will be able to define the row-level security and why it is needed. We will also be able to create security roles in Power BI and DAX filter to implement the row-level security.

## ***Row-level security***

Power BI uses the row-level security to restrict data access to the users. Row-level security is implemented using the user roles defined in the Power BI application. It restricts the user to see the data pertaining to them and not the entire data.

Row-level security is applied by enabling the different user roles and giving data access permission to the users assigned to these roles.

## ***Implementing the row-level security***

To implement the row-level security, perform the following steps:

Create visualization/s on which the security will be implemented.

Create the user roles.

Create the security/table filter using DAX.

Test the roles in the Power BI desktop.

Power BI Service configuration.

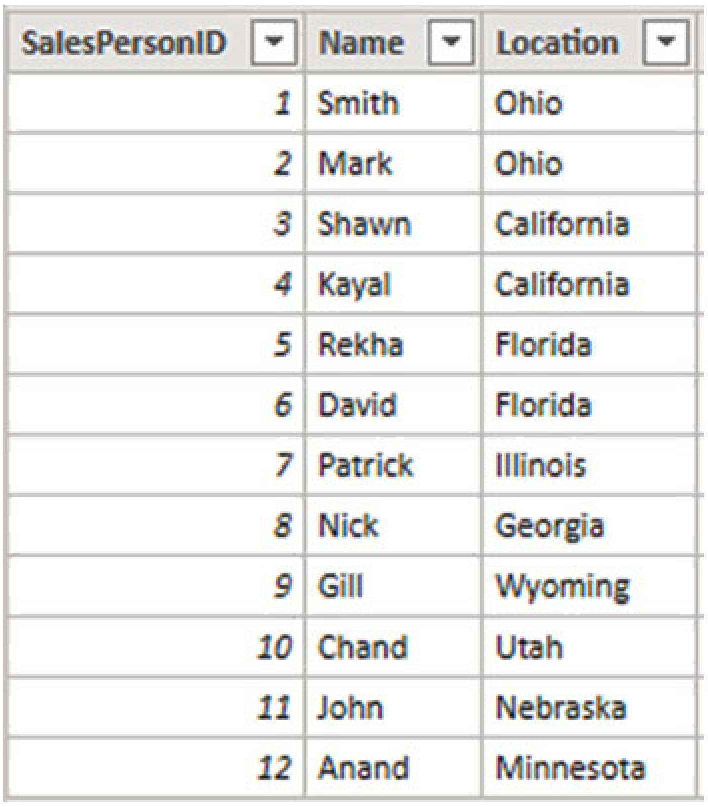
Row-level security is easy to implement, it is a powerful way to secure your data.

**Exercise:**

**Implement row-level security that restricts the Salespersons to view the data only for their assigned regions.**

For this exercise, **Chapter7_RLS.pbix** will be used. It contains the same data model as used in the previous chapter. We will review the **Sales Person** table.

Navigate to **Data View** and from the select **Sales** The **Sales Person** table contains the other data elements, but for this exercise, we are interested only in the following:



| SalesPersonID | Name | Location |
|---|---|---|
| 1 | Smith | Ohio |
| 2 | Mark | Ohio |
| 3 | Shawn | California |
| 4 | Kayal | California |
| 5 | Rekha | Florida |
| 6 | David | Florida |
| 7 | Patrick | Illinois |
| 8 | Nick | Georgia |
| 9 | Gill | Wyoming |
| 10 | Chand | Utah |
| 11 | John | Nebraska |
| 12 | Anand | Minnesota |

***Figure 7.1:*** *Sales Person data*

Our objective is that when a Sales Person views the report visuals, he should be seeing the data pertaining to his Location, that is, the relevant Customer Regions.

## *Create visualization*

For this exercise, we will be using some of the visuals from You can either copy (from and paste them into a new application or recreate them.

We will use the stacked bar chart, map, and filled map from the previous chapter. The following figure displays the original chart, without any security applied.

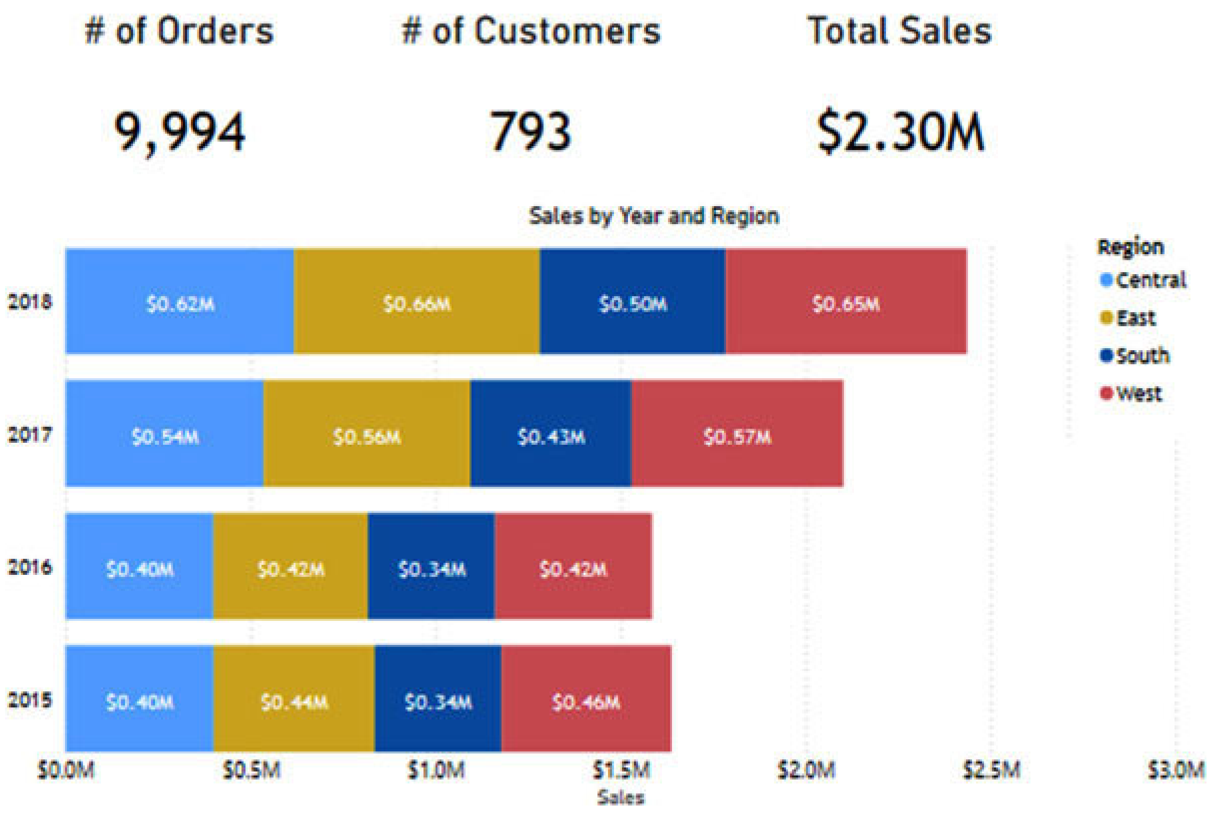


***Figure 7.2:*** *Stack bar chart and totals*

## ***Creating the role – Sales Person East***

Now, we will create the appropriate roles for the **Sales** The first one will be Sales Person East. This sales person will be allowed to see the data only for the Customers located in the Eastern region.

From the menu, navigate to the **Modeling** ribbon and select **Manage** as shown in the following screenshot:

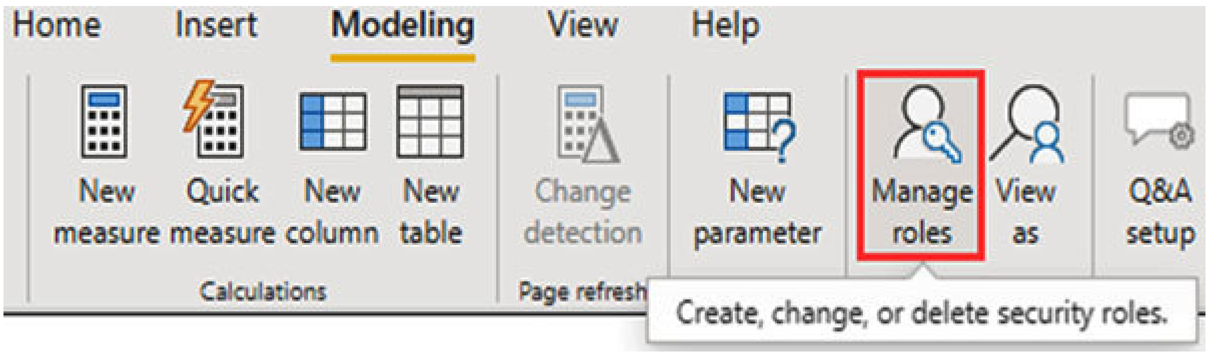


***Figure 7.3:*** *Manage roles option*

In the **Manage roles** dialog box, under select

Specify the name of the role, Tables, and Table filter DAX Expression, and click on as shown in the following screenshot:

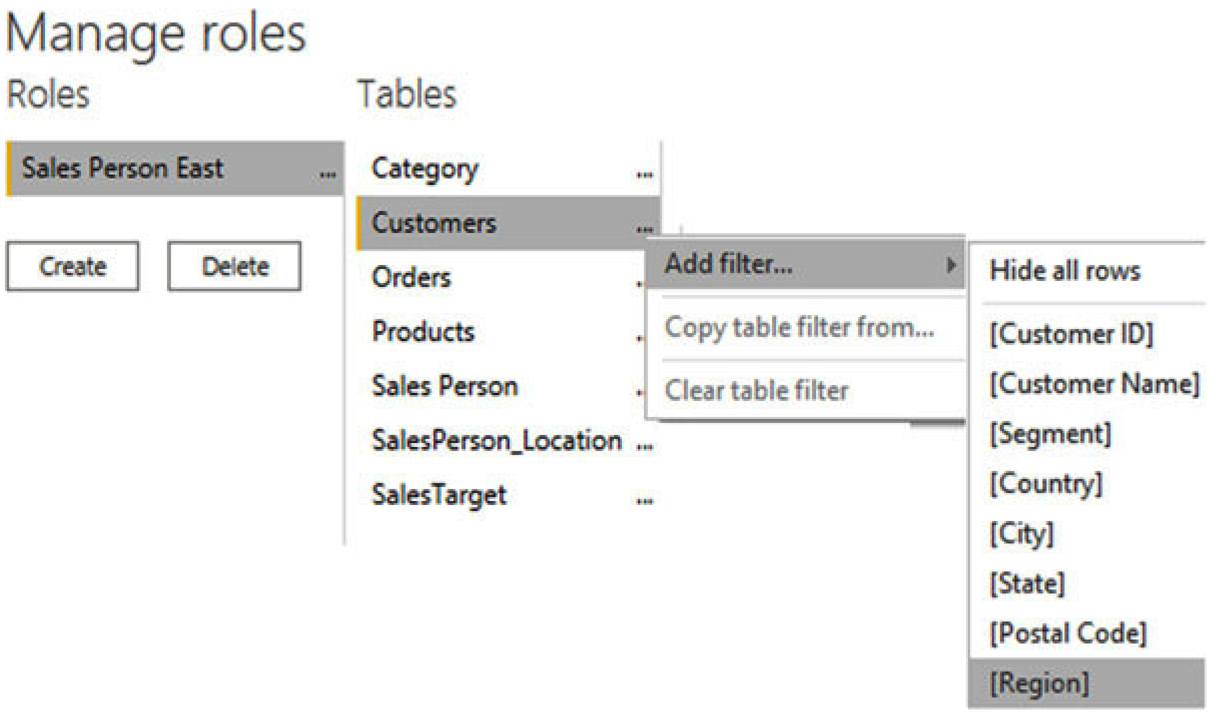
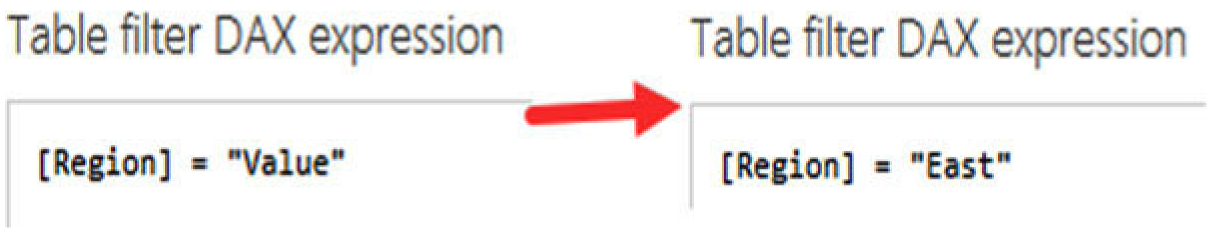


***Figure 7.4:*** *Manage roles settings*

In the next **Table filter DAX expression** box, change the **Value** to as shown in the following screenshot:



```
[Region] = "Value"
```

```
[Region] = "East"
```

***Figure 7.5:*** *Table filter DAX expression*

After specifying the expression, select

**East is the value of Region. Make sure that it has the same spelling and case as stored in the Customer table.**

Similarly, create the roles for Sales Person West, Sales Person South, and Sales Person Central.

For Sales Person West, specify the Table filter DAX expression as follows:

```
[Region] = "West"
```

For Sales Person South, specify the Table filter DAX expression as follows:

```
[Region] = "South"
```

For Sales Person South, specify the Table filter DAX expression as follows:

```
[Region] = "Central"
```

After creating all the roles, under **Manage** you will see all the four roles with the filter icon on the **Customers** table, as shown in the following screenshot:

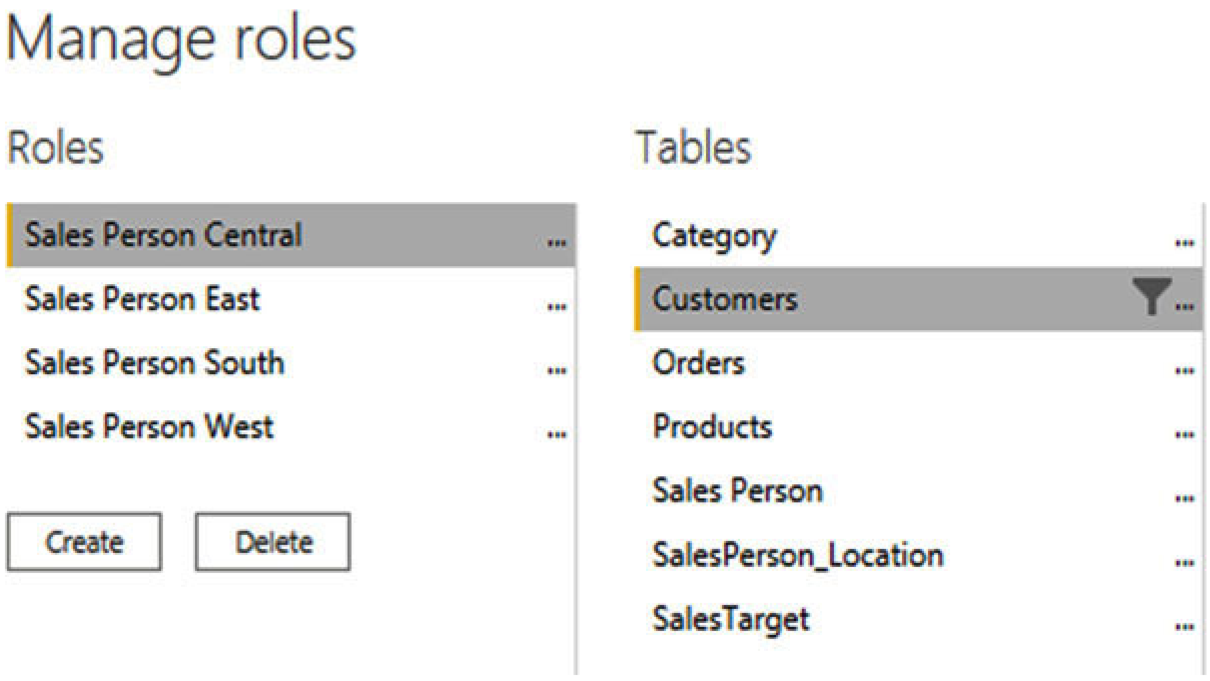
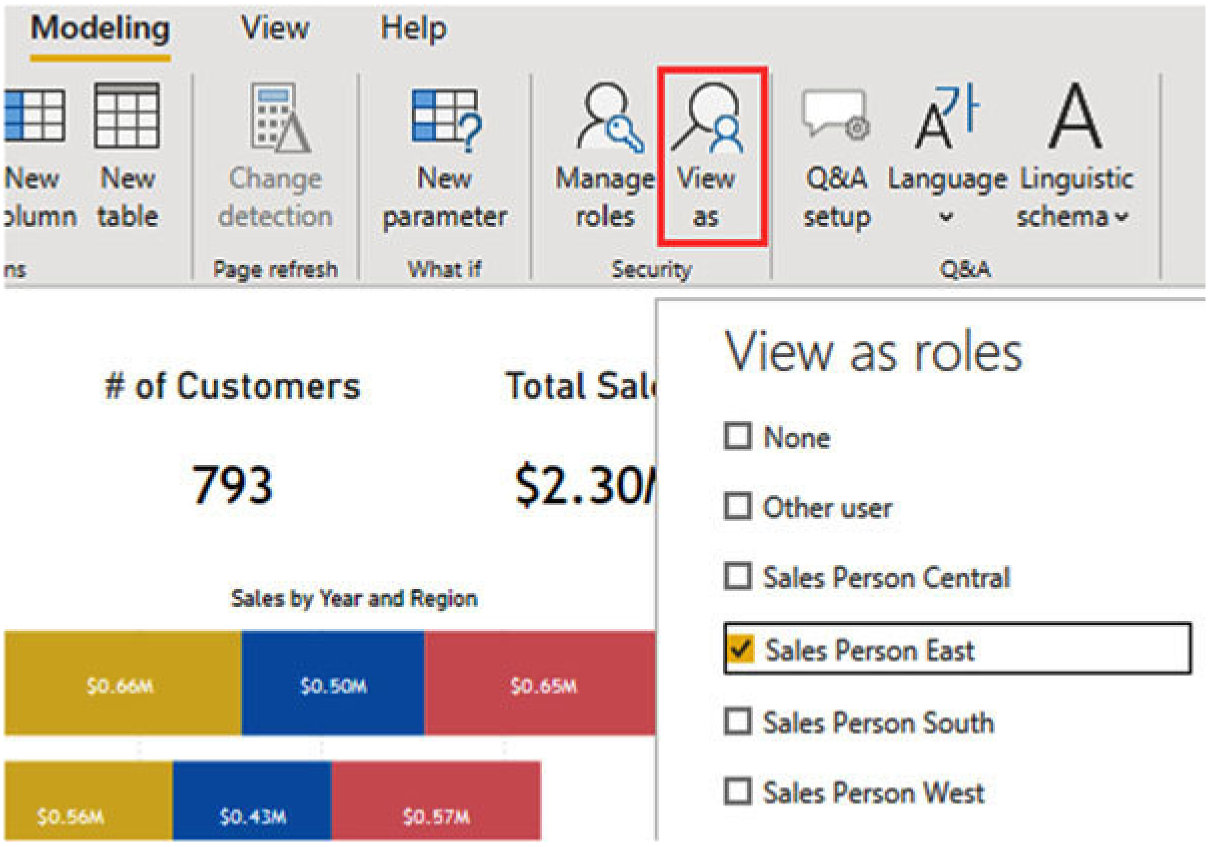


***Figure 7.6:*** *Roles created*

## Verifying the roles in the Power BI desktop

After setting up the roles, we can verify if the roles see the relevant data. Navigate to the **Modeling** view again, select **View as** and select one of the roles, as shown in the following figure, for example, Sales Person East, and click on



View as roles

- ☐ None
- ☐ Other user
- ☐ Sales Person Central
- ✓ Sales Person East
- ☐ Sales Person South
- ☐ Sales Person West

***Figure 7.7:*** *Testing the roles in Power BI desktop*

This will display the data in the Stacked bar report for only the Eastern region, as shown in the following figure:

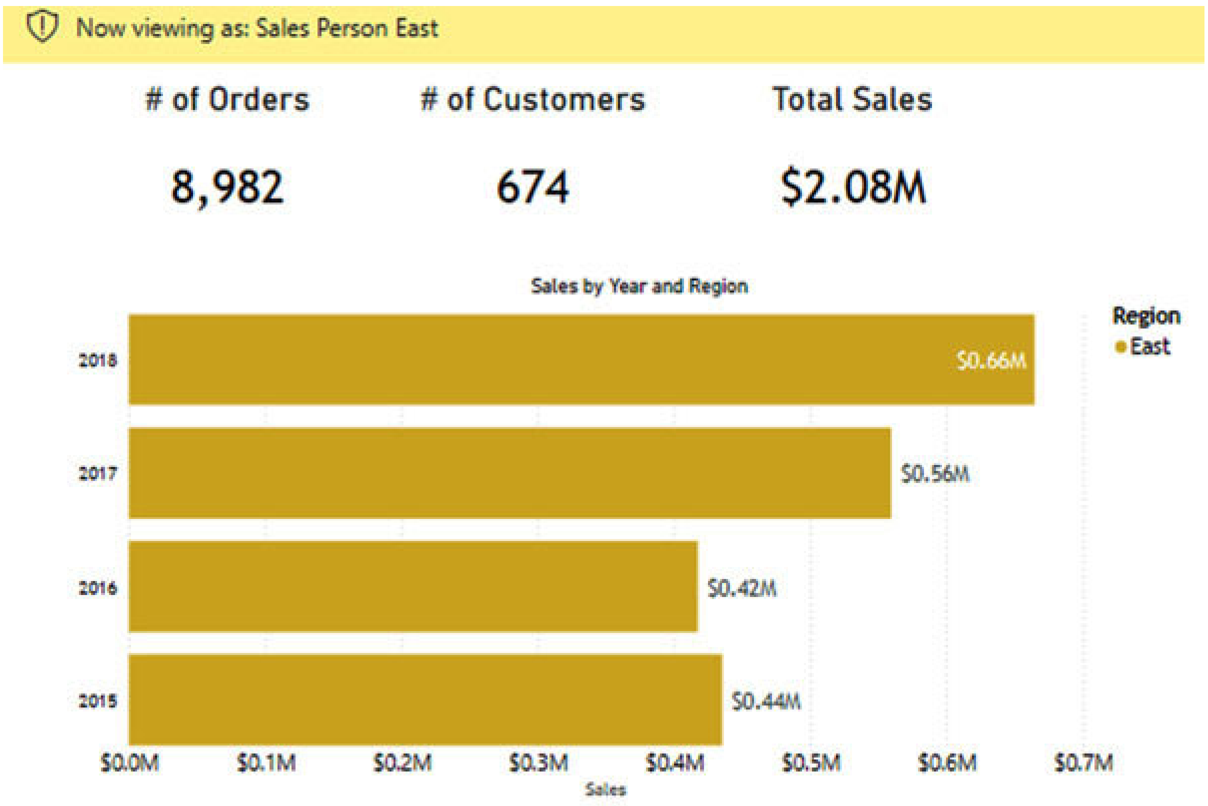


***Figure 7.8:*** *Chart display for only Eastern region*

It will also display the role for which the data is displayed at the top. If there are multiple visuals in the application, the data for all the visuals will be filtered for the role selected.

## *Power BI Service configuration*

After performing all the security tasks, we need to publish this application on the Power BI Service and do the configuration on the Power BI Service.

To access the Power BI Service, you need to sign up for it.

**To sign up for the Power BI Service, you will need a work or school email address. You will not be able to sign up using email addresses like or etc.**

To sign up to the Power BI Service, click on the small signup icon on the top of Power BI desktop window:

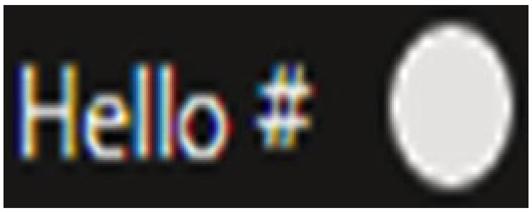


***Figure 7.9:*** *Power BI sign in*

I have already signed up and that’s why it displays my user ID, After you sign into your account, from the **Home** ribbon, select as shown in the following screenshot:

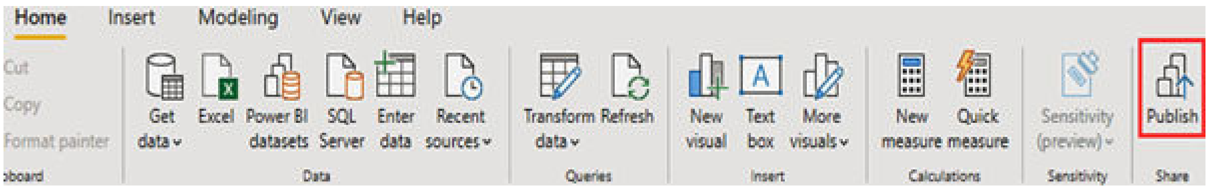
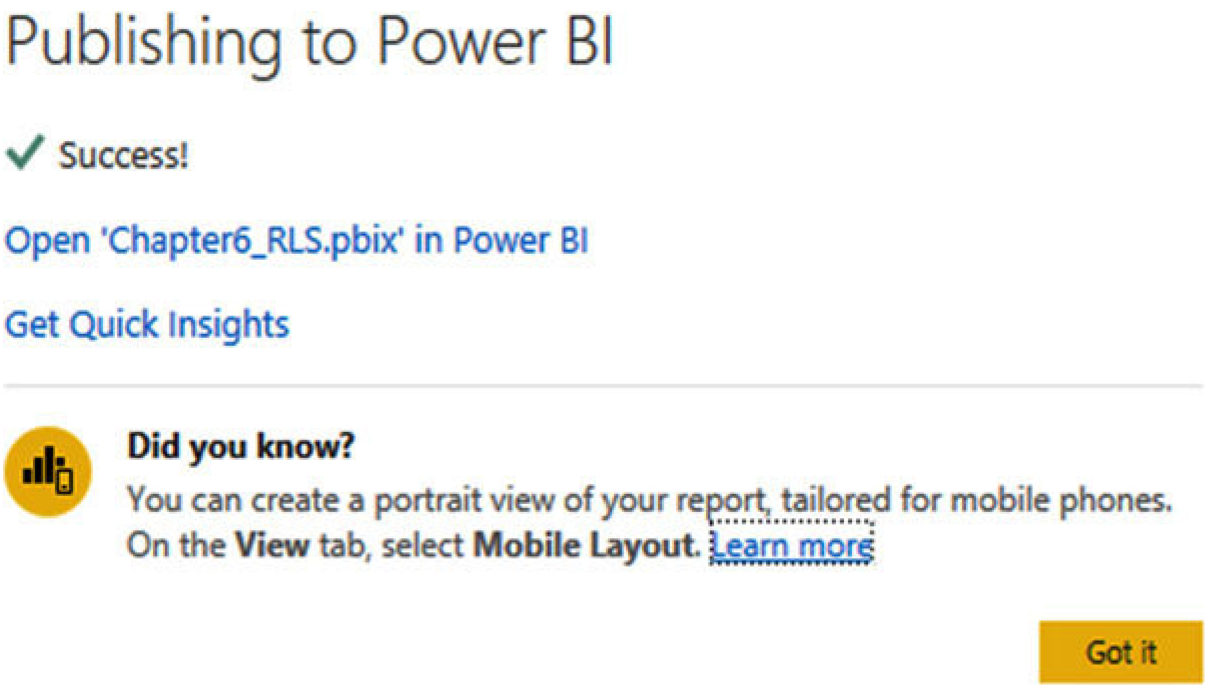


***Figure 7.10:*** *Publish your application*

Select a destination **My** This will publish the application, as shown in the following screenshot:


Publishing to Power BI

✓ Success!

Open 'Chapter6_RLS.pbix' in Power BI

Get Quick Insights

**Did you know?**

You can create a portrait view of your report, tailored for mobile phones. On the **View** tab, select **Mobile Layout**. Learn more

Got it

***Figure 7.11:*** *Published application successfully*

Click on your application; in my case, it is It will further verify your account and ask if you want to share your application with others. You can skip the step, if you don't have anyone to share your application with.

It will then show a request access box, to access your request. Click on **Send** request.

Once you have successfully logged in and granted permission, you can see the report under **My** as shown in the following screenshot:

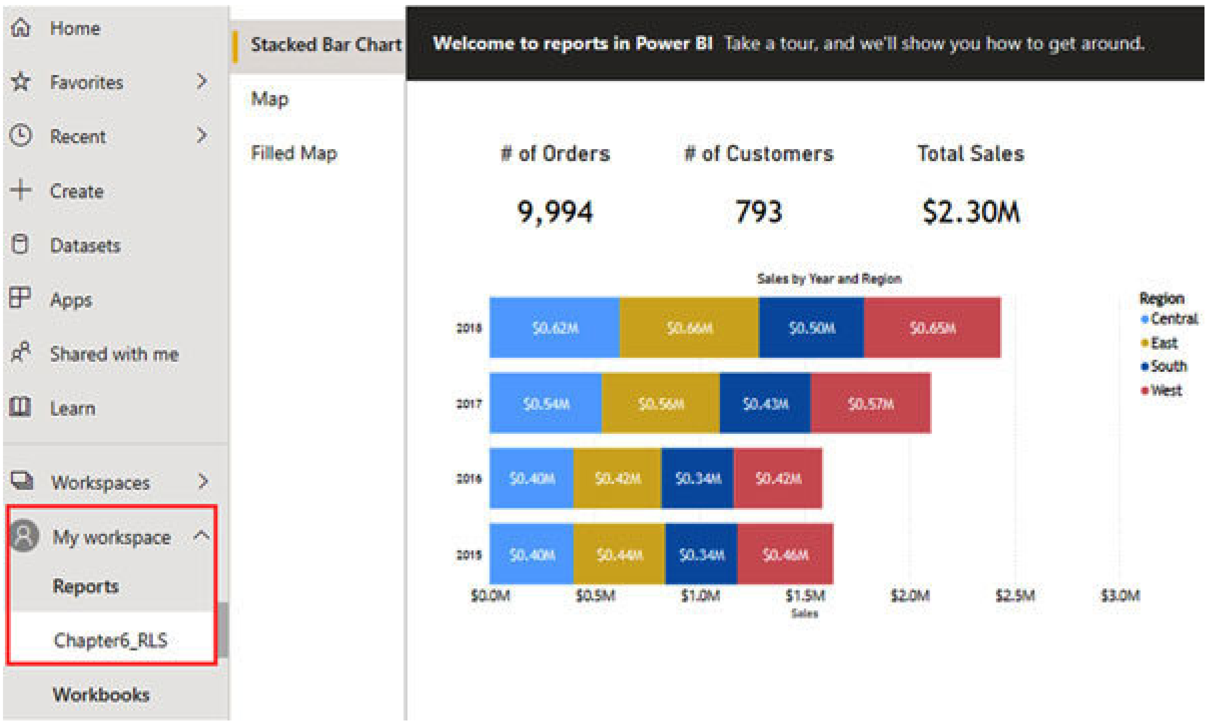


***Figure 7.12:*** *Accessing your report on Power BI Service*

**In case you don't see the report, make sure to clear the cache of your browser and try again.**

Scroll down and select the **Datasets** and then select as shown in the following screenshot:

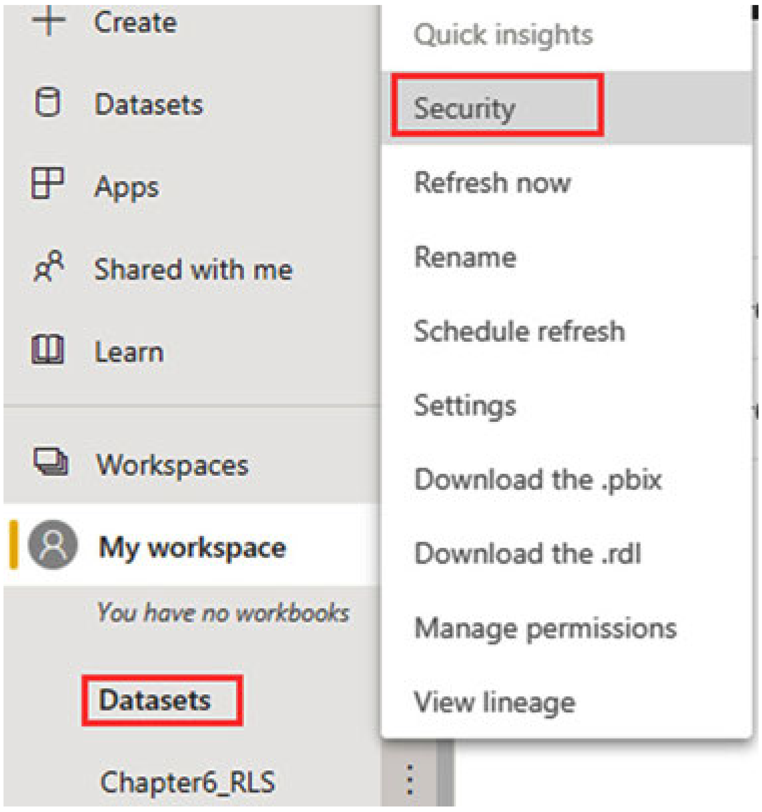
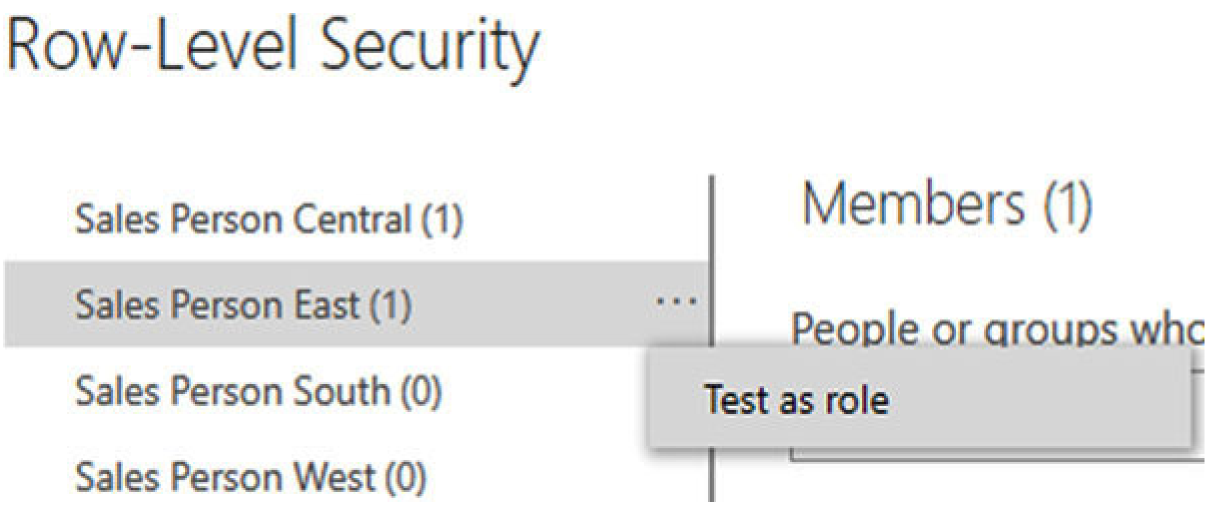


***Figure 7.13:*** *Security option under Datasets*

On the row-level security screen, add people or groups who belong to this role. For this exercise, I have added one Sales Person in Sales Person Central and one in Sales Person East.

## *Testing created roles in the Power BI Service*

We can test the users and the assigned roles in the Power BI Service. Click on the ellipsis next to the role and select **Test as** as shown in the following screenshot:



Row-Level Security

Sales Person Central (1)
Sales Person East (1)
Sales Person South (0)
Sales Person West (0)

Members (1)

People or groups whc

***Figure 7.14:*** *Testing the roles*

Click on **Test as role** for Sales Person East (1), you will get the chart for the eastern region, as shown in the following screenshot: This is the same chart that we saw on the Power BI desktop:

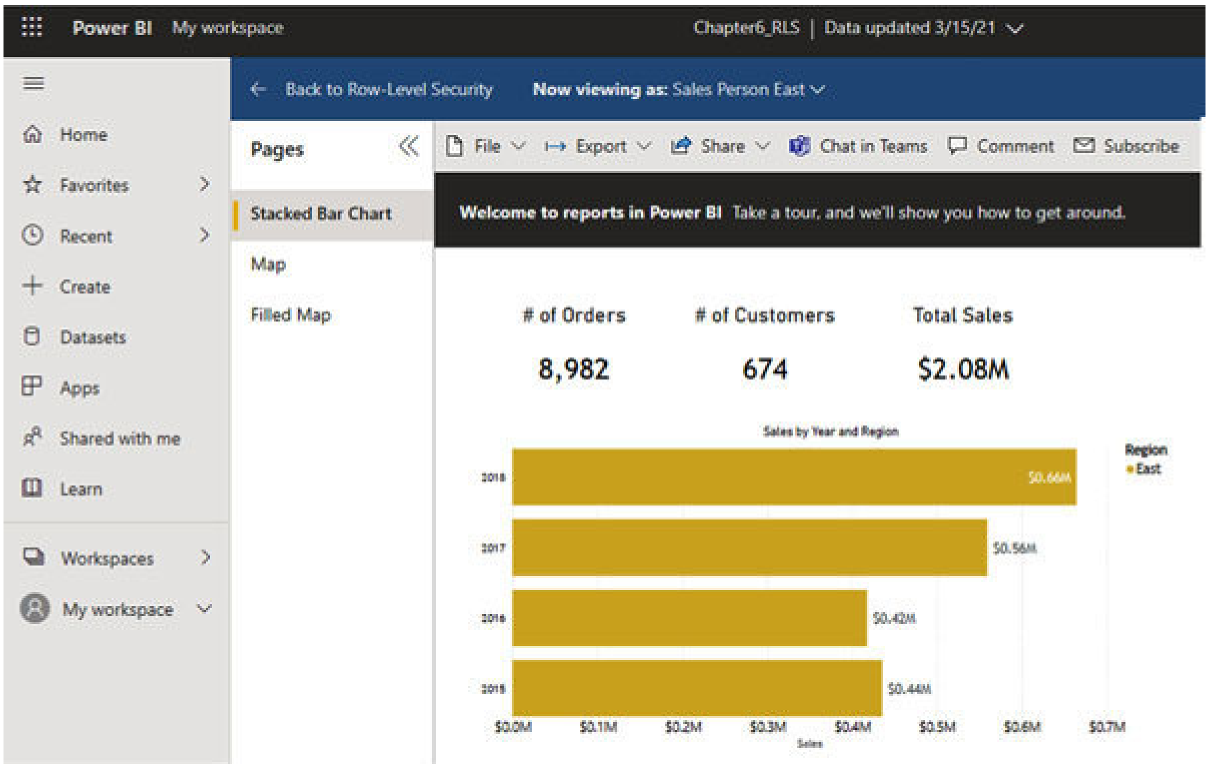


***Figure 7.15:*** *Chart display for Sales Person East*

Using the row-level security, when the user having the role of Sales Person East or Sales Person Central signs into Power BI, he will be restricted to only his data.

## Conclusion

In this chapter, we learned about the row-level security in Power BI. Row-level security provides the users with different views of the data based on their role. To implement RLS, we need the Power BI desktop application with visuals, roles, the tables to be used, and the Table filter. Once these tasks are completed, perform the Power BI Service configuration.

This completes all the concepts in Power BI. Hope you had a good learning experience.

## ***Questions***

How is data level Security implemented in Power BI?

What is the objective of implementing RLS?

What are the three important aspects of RLS?

What option is used to test the user role in the Power BI desktop?

Can I signup in the Power BI Service using my Gmail account?

## ***Answers***

Data level security is implemented using Row-level Security (RLS).

RLS is implemented to restrict the user to his view of data.

Create roles, select tables on which security will be applied, and provide the Table filter DAX expression.

The user roles can be tested in the Power BI desktop by using View as the role option under the Modeling ribbon.

The Power BI Service allows only work or school email accounts.

# Index