



HANDBOOK OF POWER BI

AN ILLUSTRATIVE GUIDE



By: Shahzad Z Khan

TABLE OF CONTENTS

1. INTRODUCTION

- 1.1 What is Power BI?
- 1.2 Benefits of Power BI
- 1.3 Power BI Architecture
- 1.4 Components of Power BI

2. GETTING STARTED

- 2.1 Power BI desktop
- 2.2 Sample data.
- 2.3 Power BI development stages

3. CONNECTING TO DATA SOURCES

- 3.1 Launch Power BI
- 3.2 Connect / Get data.

4. DATA TRANSFORMATION WITH POWER QUERY

- 4.1 Adding / removing columns
- 4.2 Custom & conditional columns
- 4.3 Column from examples
- 4.4 Splitting & merging columns
- 4.5 Pivoting / unpivoting columns
- 4.6 Merging two queries or tables
- 4.7 Append two or more queries.
- 4.8 Case study

5. BASIC DATA MODELING

- 5.1 Relationships in data modelling
- 5.2 Identifying key columns
- 5.3 Creating calculated columns
- 5.4 Calculated Measures
- 5.5 Calculated tables

6. MEASURES USING DAX

- 6.1 Quick measure / Implicit measure
- 6.2 Explicit measure using DAX.

7. DATA VISUALIZATION

- 7.1 Basic data visualization
- 7.2 Visualization interactivity
- 7.3 Building our first report

8. CUSTOM VISUALIZATION

- 8.1 Getting Custom Visuals
- 8.2 Key visuals

9. PUBLISHING THE REPORT

- 9.1 Power BI Service

10. FUNDAMENTALS OF DAX

- 10.1 Introduction to DAX
- 10.2 DAX syntax
- 10.3 Row / Filter context.
- 10.4 Key DAX formulas

11. ERRORS HANDLING

- 11.1 Common Errors
- 11.2 Troubleshooting

12. CASE STUDY

- 12.1 ESG Dashboards
- 12.2 Environmental
- 12.3 Social
- 12.4 Governance

13. SAMPLE FILES

- 13.1 Superstore Sales Data
- 13.2 HSBC ESG Data Pack

14. FINANCIAL CONTROL DASHBOARDS

- 14.1 Income Statement Analysis
- 14.2 CFO Dashboard
- 14.3 Working Capital Analysis
- 14.4 Price, Volume, Mix (PVM) Analysis
- 14.5 Variance analysis

Preface to the First Edition

This illustrative guide is written for users of Power Query in Excel and Power BI at an intermediate level. It will give you an introduction to Power BI architecture and components. Data transformation with Power Query, data modeling, relationships and visualizations covered in detail. The most important part that is creating Measures using DAX (Data Analysis Expressions) has been covered with illustrations.

About the Author / Expert

Shahzad Khan is the founder of [Data Synergy](#), a global training & consulting firm. He is a qualified finance professional and a data enthusiast. He has overall experience of around two decades in Corporate Finance, Financial Modeling, Internal Audit, Sustainability, Data Analytics, and Business Intelligence in various sectors including Big4 Accounting Firm (Ernst & Young), Investments (KFH) and Kuwait Petroleum.

He is a [Chartered Management Accountant](#) with a degree in Master's in economics. He has also completed CFA Investment Certificate and has done Financial Markets, Corporate Sustainability & Risk Management and Data Analytics & Visualization courses using R and Power BI from Harvard University, edX.

He has worked with KFH RE Investment Company, a wholly owned subsidiary of Kuwait Finance House. KFH is one of the pioneer Sharia compliant banks in Kuwait. Presently he is working on various consulting projects related to data strategy, risk data analytics, business intelligence and ESG / sustainability.

He has been part of Advisory and Business Community Training at E&Y and conducted professional trainings at various institutes in Kuwait. He has also conducted various workshops & online courses / webinars in ESG and Data Analytics using Power BI and R language for Banking & Finance.

A handwritten signature in blue ink, appearing to read "SZKhan", with a stylized underline.

Shahzad Z Khan

Email: fintraining.info@gmail.com

Updated: November 23, 2025

BY: SHAHZAD Z KHAN

I. INTRODUCTION

WHAT IS POWER BI?

Power BI is a business intelligence tool provided by Microsoft that lets you connect to various data sources, visualize your data, and share insights. It gets data from different sources to build interactive dashboards and Business Intelligence reports.

Business analytics / Intelligence is a technical term referring to the methods and systems for collecting, storing, processing, analyzing, and visualizing data for business purposes.

Power BI is not simply a generalized software, it is really a gigantic collection of services offered by Microsoft for transforming, modeling, analyzing, and visualizing data.

BENEFITS OF POWER BI

1. Access to Volumes of Data from Multiple Sources

Power BI can access vast volumes of data from multiple sources. It allows you to view, analyze, and visualize vast quantities of data that cannot be opened in Excel. Some of the important data sources available for Power BI are Excel, CSV, XML, JSON, pdf, etc. Power BI uses powerful compression algorithms to import and cache the data within the PBIX file.

2. Interactive UI/UX Features

Power BI makes things visually appealing. It has an easy drag and drops functionality, with features that allow you to copy all formatting across similar visualizations.

3. Exceptional Excel Integration

Power BI helps to gather, analyze, publish, and share Excel business data. Anyone familiar with Office 365 can easily connect Excel queries, data models, and reports to Power BI Dashboards.

4. Accelerate Big Data Preparation with MS Azure

Using Power BI with Azure, Microsoft's cloud platform, allows you to analyze and share massive volumes of data. An azure data lake can reduce the time it takes to get insights and increase collaboration between business analysts, data engineers, and data scientists.

5. Turn Insights into Action

Power BI allows you to gain insights from data and turn those insights into actions to make data-driven business decisions.

6. Real-time Stream Analytics

Power BI will enable you to perform real-time stream analytics. It helps you fetch data from multiple sensors and social media sources to get access to real-time analytics, so you are always ready to make business decisions.

POWER BI ARCHITECTURE

Power BI architecture is a service built on top of Azure. There are multiple data sources that Power BI can connect to. Power BI Desktop allows you to create reports and data visualizations on the dataset. Power BI gateway is connected to on-premises data sources to get continuous data for reporting and analytics. Power BI services refer to the cloud services that are used to publish Power BI reports and data visualizations. Using Power BI mobile apps, you can stay connected to their data from anywhere. Power BI apps are available for Windows, iOS, and Android platforms.

POWER BI SERVICE

Power BI service is the Software as a Service (SaaS) part of Power BI. It is also known as Power BI Online. To access Power BI Service, you need to log in to Power BI service.

It allows you to connect to data, create reports and dashboards, and ask questions about your data.

POWER BI DASHBOARD

Power BI Dashboard is a single page visualization to tell a story. The visualizations on a dashboard are generated from reports, and each report is based on one dataset. A single page dashboard is known as a [Canvas](#).

COMPONENTS OF POWER BI

1. Power Query

Power Query is the data transformation and mash up engine. It enables you to discover, connect, combine, and refine data sources to meet your analysis need. It is an ETL tool available in Excel or can be used as part of the Power BI Desktop.

2. Power Pivot

Power Pivot is a data modeling technique that lets you create data models, establish relationships, and create calculations. It uses [Data Analysis Expression \(DAX\) language](#) to model simple and complex data.

3. Power View

Power View is a technology that is available in Excel, Share point, SQL Server, and Power BI. It lets you create interactive charts, graphs,

maps, and other visuals that bring your data to life. It can connect to data sources and filter data for each data visualization element or the entire report.

4. Power Map

Microsoft's Power Map for Excel and Power BI is a 3-D data visualization tool that lets you map your data and plot more than a million rows of data visually on Bing maps in 3-D format from an Excel table or Data Model in Excel. Power Map works with Bing maps to get the best visualization based on latitude, longitude, or country, state, city, and street address information.

5. Power BI Desktop

Power BI Desktop is a development tool for Power Query, Power Pivot, and Power View. With Power BI Desktop, you have everything under the same solution, and it is easier to develop BI and data analysis experience.

6. Power Q&A

The Q&A feature in Power BI lets you explore your data in your own words. It is the fastest way to get an answer from your data using [natural language](#). An example could be what was the total sales last year? Once you have built your data model and publish using the Power BI service, then you can ask questions and get answers quickly using [Quick Insights feature](#).

FEATURES OF POWER BI

Some of the key features of Power BI are:

1. Power BI Desktop

Power BI Desktop is free software that you can download and install, and it allows you to build reports by accessing data easily. For using Power BI desktop, you do not need advanced report designing, or query skills to build a report.

2. Stream Analytics

Power BI's primary advantage is it supports stream analytics. From factory sensors to social media sources, Power BI assists in real-time analytics to make timely decisions.

3. Multiple Data Sources

Support for various data sources is one of the vital features of Power BI. You can access various sources of data such as Excel, CSV, SQL Server, Web files, etc. to create interactive visualizations.

4. Custom Visualization

While dealing with complex data, Power BI's default standard might not be enough in some cases. In that case, you can access the custom library of visualization that meets your needs.

2. GETTING STARTED

2.1 Power BI Desktop

Power BI Desktop is a free desktop application from Microsoft that can be installed on your Windows machine, and it is the primary tool for creating data models and reports in Power BI.

From Power BI Desktop, you can save your reports locally to your hard drive as a document, which has a file extension of PBIX. Power BI developers often refer to these locally stored reports as PBIX files.

2.2 Sample Data

- Find the sample files at the end of this guide.
- Save the file to your computer and get data through the Power BI Desktop.

2.3 Power BI Development Stages

Power BI is designed to be user friendly. Once you connect to a data source, you can shape and transform the data (remove columns, change data types, and so on), do data modelling (create relations) and visually present that data.

This guide steps you through the following stages:

- Connect to data sources.
- Data transformation using Power Query.
- Data modeling or creating relationships.
- Report on the data / create visuals.

3. CONNECTING TO DATA SOURCES

Power BI can connect to hundreds of data sources, with over 200 connectors available for sources like SQL Server, Excel, Google Analytics, and Azure services. While the number of available data connectors is vast and constantly expanding.

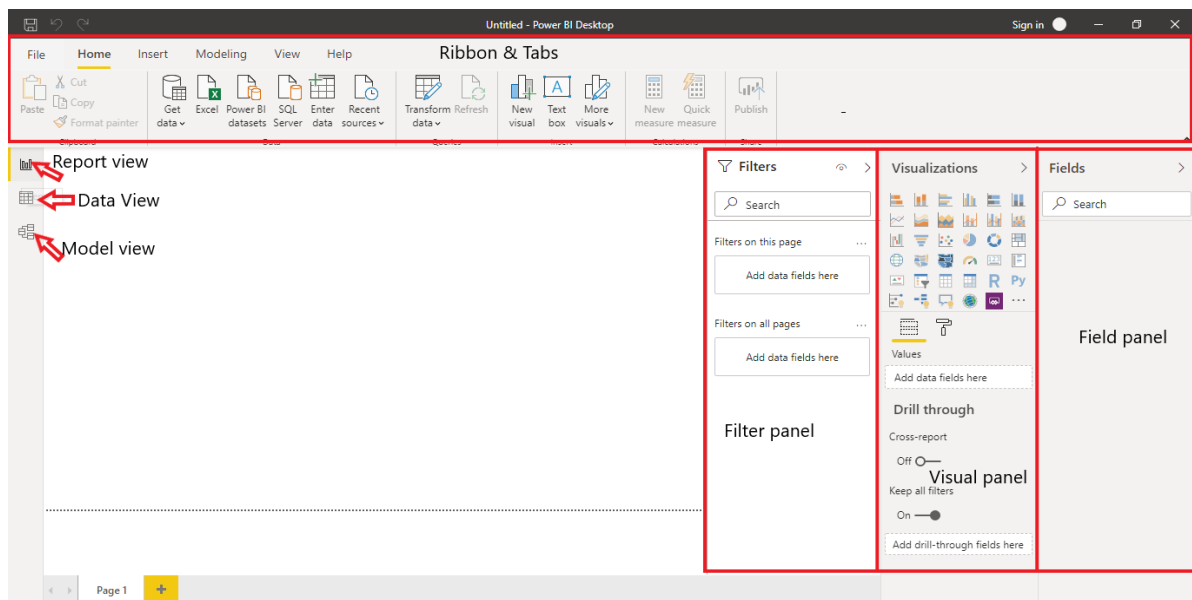
Commonly used Power BI data sources include:

- File (Excel, Text/CSV, XML, JSON, PDF).
- Database (SQL Server, Oracle, IBM DB2, MySQL etc.).
- Power Platform (Power BI datasets, Common Data Services)
- Cloud Services (MS Azure, SQL Database, Analysis Services, MS Fabric)
- Online Services (SharePoint, Dynamics 365, Dynamics Business Central, Salesforce, Google Analytics, GitHub, etc.).
- Others (Web, ODBC, Hadoop, Spark, R script, Python script, etc.).

In this section, we will see how to retrieve data for the first time and introduce you to Power Query, Power BI Desktop and how they relate to each other and complement each other. And you will be developing the practical skills you need to become an effective Power BI developer.

Get Data

The available data sources can be accessed via the [Get data](#) button in Power BI.



The data sources are broken down by category such as files or databases. Files can be things like Excel spreadsheets, text / csv files, or even PDFs.

We will select the file data source that category to get a filtered list of the possible data sources we can connect to.

Let's select the text CSV option. CSV stands for “comma separated values”. It's a kind of plain text file that allows you to separate your data into rows and columns by use of a particular character known as a delimiter.

When we select text CSV, a window opens to find and choose file from your computer.

You can find the [sample files](#) at the end of this guide, and you just need to navigate to the folder where you need to save them. First, select the file named Superstore data.csv.

When you open it, you will see a preview window that gives you a preview of the data along with some buttons in the bottom right corner, where we can either cancel, transform data or load.

4. DATA TRANSFORMATION WITH POWER QUERY

[Power Query](#) is a [magic](#) tool from Microsoft which is used for data transformation and data preparation. Power Query comes with a graphical interface for getting data from sources and a [Power Query Editor](#) for applying transformations. Using Power Query, you can perform the [extract, transform, and load](#) (ETL) processing of data.

If we click load, it will take us straight into the [Report Editor](#), bypassing Power Query without doing any transformations, you can use the load button if you know your data is clean and precise. It doesn't require any transformation.

But mostly the data sets require transformation or cleanup first, so we are going to click the [transform data button](#), and this will open [Power Query](#).

So now, we have created our first query.

A query is a connection to a data source for the purpose of gathering information. Whether the data source is a database or a file or a web page, our query is asking that data source to give us its data.

Once we have queried that data from our source, Power Query then gives us the tools we need to transform that data before loading it into our data model.

Now, let's take an overview of where everything locates in Power Query.

The first thing you will see is the [data preview](#) in the center of your screen.

If your data set is large enough, this won't show you all the rows, but it will show you all your columns and at least gives you a preview to help you in your data modeling. Also, notice how each of our column headers has a drop-down menu icon, just like the column headers in an Excel worksheet. These are very important to manipulate our dataset columns.

Superstore Analysis Report

File	Home	Transform	Add Column	View	Tools	Help
Close	New Source Recent Sources Enter Data	Data source settings Manage Parameters Refresh Preview Advanced Editor Manage	Choose Columns Remove Columns Keep Rows Remove Rows Split Column Group By Data Type: Date Use First Row as Headers Replace Values	Merge Queries Append Queries Combine Files		

2	04-Jan-20	East	Boston	Crackers	Whole Wheat	87
3	07-Jan-20	West	Los Angeles	Cookies	Chocolate Chip	58
4	10-Jan-20	East	New York	Cookies	Chocolate Chip	82
5	13-Jan-20	East	Boston	Cookies	Arrowroot	38
6	16-Jan-20	East	Boston	Bars	Carrot	54
7	19-Jan-20	East	Boston	Crackers	Whole Wheat	149
8	22-Jan-20	West	Los Angeles	Bars	Carrot	51
9	25-Jan-20	East	New York	Bars	Carrot	100
10	28-Jan-20	East	New York	Snacks	Potato Chips	28
11	31-Jan-20	East	Boston	Cookies	Arrowroot	36
12	03-Feb-20	East	Boston	Cookies	Chocolate Chip	31
13	06-Feb-20	East	Boston	Crackers	Whole Wheat	28
14	09-Feb-20	West	Los Angeles	Bars	Carrot	44
15	12-Feb-20	East	New York	Bars	Carrot	23
16	15-Feb-20	East	New York	Snacks	Potato Chips	27
17	18-Feb-20	East	Boston	Cookies	Arrowroot	43

Query Settings

PROPERTIES

Name: Sales_Data (2)

All Properties

APPLIED STEPS

- Source
- Navigation
- Changed Type

There is a **ribbon**, at the top of the screen which is the main navigation in tools menu just like we have in our Microsoft Office applications and in Power BI Desktop.

Home tab gives us options to add more data sources to our model, filter the rows and columns in our data set and merge data sets with each other.

The **transform tab** lets us reformat and restructure our data in a variety of ways.

In languages such as Python and R, you can insert scripts to manipulate your data sets.

The **add column tab** shows us ways we can supplement our data model with new columns, whether that's by deriving them from existing columns, writing custom functions, or simply creating a row index.

Finally, the [view tab](#) has a few options for examining the content of our data model with some cool features that will allow us to see if our data sources contain any unclean data.

There is a [queries pane](#) on the left side of your screen in Power Query. This provides you a list of all the data sources your data model is connected to.

You can use this pane to add new queries, group sets of queries together or create query parameters. You can also create [custom functions](#) in Power BI, and they will also be visible on this pane.

There is a [query settings pane](#) on the right side of your screen. In query settings, you can rename your data set if you want and give it a description.

The main thing in this section is called [applied steps](#). For each query in your data model, the applied steps show all the steps you have taken to transform your data before loading it into the data model.

We can only view three steps till now.

The first one [source](#) just shows Power Query connecting to our CSV file and you can click on any one of these steps to see what the data set looked like once that step was applied.

Here, Power Query is giving us the ability to replay our changes to see how our data transformations are having an effect. So, if you click on the source step, notice how the data preview changes.

Our column headers are in row one. This is how Power Query initially reads the data.

And this is not ideal because we need that first row to be the column headers.

However, Power Query is quite intelligent. So, after it connected to the file, it automatically created the second step called [promoted headers](#).

If we click on this step, you will see Power Query moved all the data up a row to get those column headers in the right place.

In the last step, [change type](#) Power Query scan each of the columns in our dataset and once again, being helpful in intelligent recognize the underlying data type in each column, and automatically transformed each column to fit its data type.

In the first few steps, we will pull data from a CSV file. And then we can use Power Query to add more transformations store data set, manipulating the recipe ingredients to build out a better and complex method that will ultimately lead us to our finished data model.



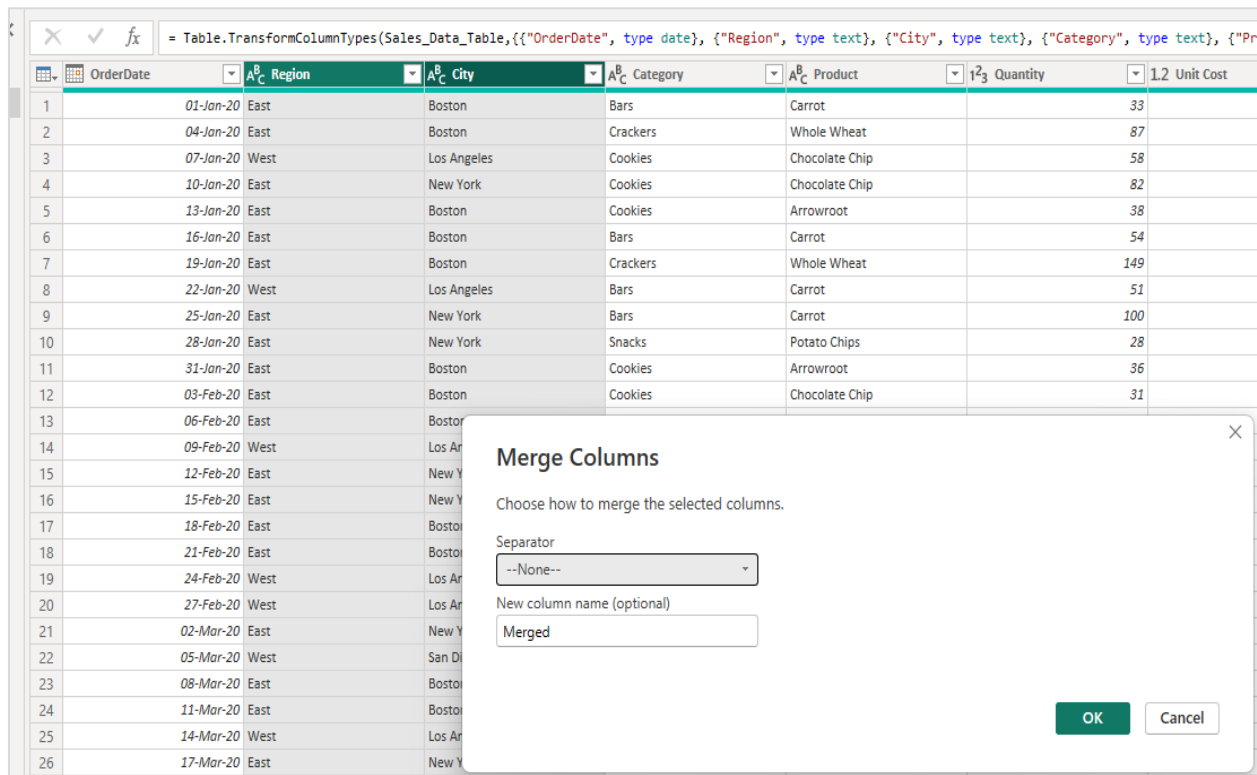
Tip: With Power Query, we can automate our regular reports with a click of the refresh button monthly. All we need to do is build the query once and reuse it, saving an hour of work every month.

DATA WRANGLING WITH POWER QUERY

Merge Columns

Method 1: Using Power Query UI

1. Go to **Power Query Editor** (Home → Transform Data).
2. Select the columns you want to merge (hold **Ctrl** to select multiple).
3. On the **Transform** tab, click **Merge Columns**.
4. Choose a **separator** (e.g., space, comma, custom text).
5. Provide a **new column name**.
6. Click **OK**.



The screenshot shows the Power Query Editor interface. The background table has columns: OrderDate, Region, City, Category, Product, Quantity, and Unit Cost. A 'Merge Columns' dialog box is open in the foreground, allowing the user to choose how to merge selected columns. The dialog includes a 'Separator' dropdown set to '--None--', a 'New column name (optional)' text box containing 'Merged', and 'OK' and 'Cancel' buttons.

	OrderDate	Region	City	Category	Product	Quantity	Unit Cost
1	01-Jan-20	East	Boston	Bars	Carrot		33
2	04-Jan-20	East	Boston	Crackers	Whole Wheat		87
3	07-Jan-20	West	Los Angeles	Cookies	Chocolate Chip		58
4	10-Jan-20	East	New York	Cookies	Chocolate Chip		82
5	13-Jan-20	East	Boston	Cookies	Arrowroot		38
6	16-Jan-20	East	Boston	Bars	Carrot		54
7	19-Jan-20	East	Boston	Crackers	Whole Wheat		149
8	22-Jan-20	West	Los Angeles	Bars	Carrot		51
9	25-Jan-20	East	New York	Bars	Carrot		100
10	28-Jan-20	East	New York	Snacks	Potato Chips		28
11	31-Jan-20	East	Boston	Cookies	Arrowroot		36
12	03-Feb-20	East	Boston	Cookies	Chocolate Chip		31
13	06-Feb-20	East	Boston				
14	09-Feb-20	West	Los Angeles				
15	12-Feb-20	East	New York				
16	15-Feb-20	East	New York				
17	18-Feb-20	East	Boston				
18	21-Feb-20	East	Boston				
19	24-Feb-20	West	Los Angeles				
20	27-Feb-20	West	Los Angeles				
21	02-Mar-20	East	New York				
22	05-Mar-20	West	San Diego				
23	08-Mar-20	East	Boston				
24	11-Mar-20	East	Boston				
25	14-Mar-20	West	Los Angeles				
26	17-Mar-20	East	New York				

= Table.CombineColumns(#"Changed Type",{ "Region", "City"},C			
	OrderDate	A ^B _C Merged	A ^B _C Category
1	01-Jan-20	East Boston	Bars
2	04-Jan-20	East Boston	Crackers
3	07-Jan-20	West Los Angeles	Cookies
4	10-Jan-20	East New York	Cookies
5	13-Jan-20	East Boston	Cookies
6	16-Jan-20	East Boston	Bars
7	19-Jan-20	East Boston	Crackers
8	22-Jan-20	West Los Angeles	Bars
9	25-Jan-20	East New York	Bars
10	28-Jan-20	East New York	Snacks
11	31-Jan-20	East Boston	Cookies
12	03-Feb-20	East Boston	Cookies
13	06-Feb-20	East Boston	Crackers

Splitting Columns

Power Query allows us to split a column into one or more other columns. There are various options available under the Split Column dropdown.

We will be using By Delimiter to split the Merged column into Region and City.

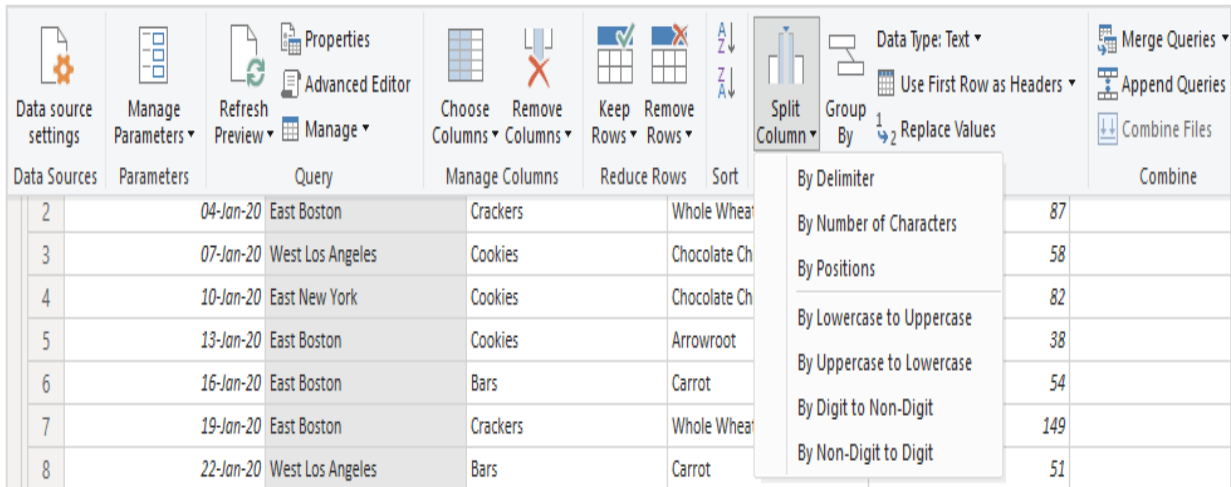
Method 1: Using Power Query UI

1. Open **Power Query Editor** (Home → Transform Data).
2. Select the column you want to split.
3. Go to **Home → Split Column** or **Transform → Split Column**.
4. Choose one of these options:
 - **By Delimiter** (e.g., comma, space, custom text)
 - **By Number of Characters**
 - **By Positions**

- **By Lowercase to Uppercase**
 - **By Digit to Non-Digit**
5. Choose whether to split:
- **Into Columns** (creates new columns)
 - **Into Rows** (creates new rows)
6. Click **OK**.

In the dialog box, ensure Select or enter delimiter is set to Space.

- Select Each occurrence of the delimiter and click OK.



Data Sources		Parameters		Query		Manage Columns		Reduce Rows		Sort		Combine	
2	04-Jan-20	East Boston	Crackers	Whole Wheat									
3	07-Jan-20	West Los Angeles	Cookies	Chocolate Chip									
4	10-Jan-20	East New York	Cookies	Chocolate Chip									
5	13-Jan-20	East Boston	Cookies	Arrowroot									
6	16-Jan-20	East Boston	Bars	Carrot									
7	19-Jan-20	East Boston	Crackers	Whole Wheat									
8	22-Jan-20	West Los Angeles	Bars	Carrot									

×

Split Column by Delimiter

Specify the delimiter used to split the text column.

Select or enter delimiter

Space

Split at

☐ Left-most delimiter

☐ Right-most delimiter

☒ Each occurrence of the delimiter

▸ Advanced options

Quote Character

"

☐ Split using special characters

Insert special character

OK

Cancel

The merged column is split now into two columns, showing Region and city separately.

OrderDate	A ^B _C Merged.1	A ^B _C Merged.2
01-Jan-20	East	Boston
04-Jan-20	East	Boston
07-Jan-20	West	Los Angeles
10-Jan-20	East	New York
13-Jan-20	East	Boston
16-Jan-20	East	Boston
19-Jan-20	East	Boston
22-Jan-20	West	Los Angeles
25-Jan-20	East	New York
28-Jan-20	East	New York
31-Jan-20	East	Boston
03-Feb-20	East	Boston
06-Feb-20	East	Boston

Method 2: Using M Code

You can also do this in the **Advanced Editor** using `Table.AddColumn` or `Text.Combine`.

Example:

```
= Table.AddColumn(
    PreviousStep,
    "MergedColumn",
    each Text.Combine({[FirstName], [LastName]}, " ")
)
```

Tips

- If you want to keep original columns, use `Table.AddColumn`.
- If you want to replace them, use `Table.CombineColumns`.

Adding Columns

1. Conditional Column
2. Custom Column
3. Column from Examples

Conditional Column

Why Use Conditional Columns in Power BI

In Power Query - Excel or Power BI, you can add a Conditional Column to create new columns based on logic, same as an IF statement in Excel. The use of conditional columns in Power BI can provide valuable insights and context that might not be immediately apparent when viewing the raw data. With conditional columns, users can create new data based on conditions that meet the criteria set by the user. This can help to better understand trends, patterns, and correlations in the dataset. Conditional columns can also be used to group and categorize data, making it easier to analyze and visualize.

Now, write the DAX formula for your conditional column. The syntax for a basic DAX formula is as follows:

COLUMN NAME = IF (CONDITION, VALUE_IF_TRUE, VALUE_IF_FALSE)

Once you have written your DAX formula, click "Enter" to create your new conditional column. Your new column will now appear in the "Fields" list, where you can drag and drop it into your visualizations.

It is important to note that the "CONDITION" in your DAX formula can be any logical expression that evaluates to either true or false. This can include comparisons between columns, mathematical operations, or even more complex expressions using functions like "AND" and "OR".

 You can use the "SWITCH" function in place of the "IF" function to create more complex conditional columns with multiple conditions and

corresponding values. This can be especially useful when dealing with large datasets or complex business logic.

Applications of Conditional Columns in Power BI

Conditional columns can be incredibly useful when working with complex data sets. Here are a few examples of how conditional columns can be used in real-world applications:

- **Grouping sales by region:** A conditional column can be used to group sales by region, based on the location of the sale.
- **Calculating customer age:** A conditional column can be used to calculate the age of each customer, based on their date of birth. This can then be used to group customers by age group.
- **Categorizing customers by spending habits:** A conditional column can be used to categorize customers based on their spending habits, such as high-spending, medium-spending, and low-spending customers.
- **Calculating Discount etc.:** A conditional column can be used to calculate discount based on the sales revenue. For example, a 10% discount will be given to customers if sales revenue is greater than \$100 else no discount.

Method 1: Using Power Query UI

1. Open Power Query Editor (Home → Transform Data).
2. Go to **Add Column → Conditional Column**.
3. In the dialog:
 - **Column Name:** Enter the name for the new column.
 - **Column to Evaluate:** Select the column you want to apply conditions on.
 - **Operator:** Choose (equals, greater than, etc.).
 - **Value:** Enter the value to compare.
 - **Output:** Enter what should appear if the condition is true.
4. Add more conditions using **Add Clause**.
5. Set the **Else** value for cases that don't match any condition.
6. Click **OK**.

✕

Add Conditional Column

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

New column name

	Column Name	Operator	Value ①		Output ①
If	Profit	is greater than or...	ABC 123 300	Then	ABC 123 Good
Else If	Profit	is greater than or...	ABC 123 200	Then	ABC 123 Average ...

Else ①
 ABC 123

Method 2: Using M Code

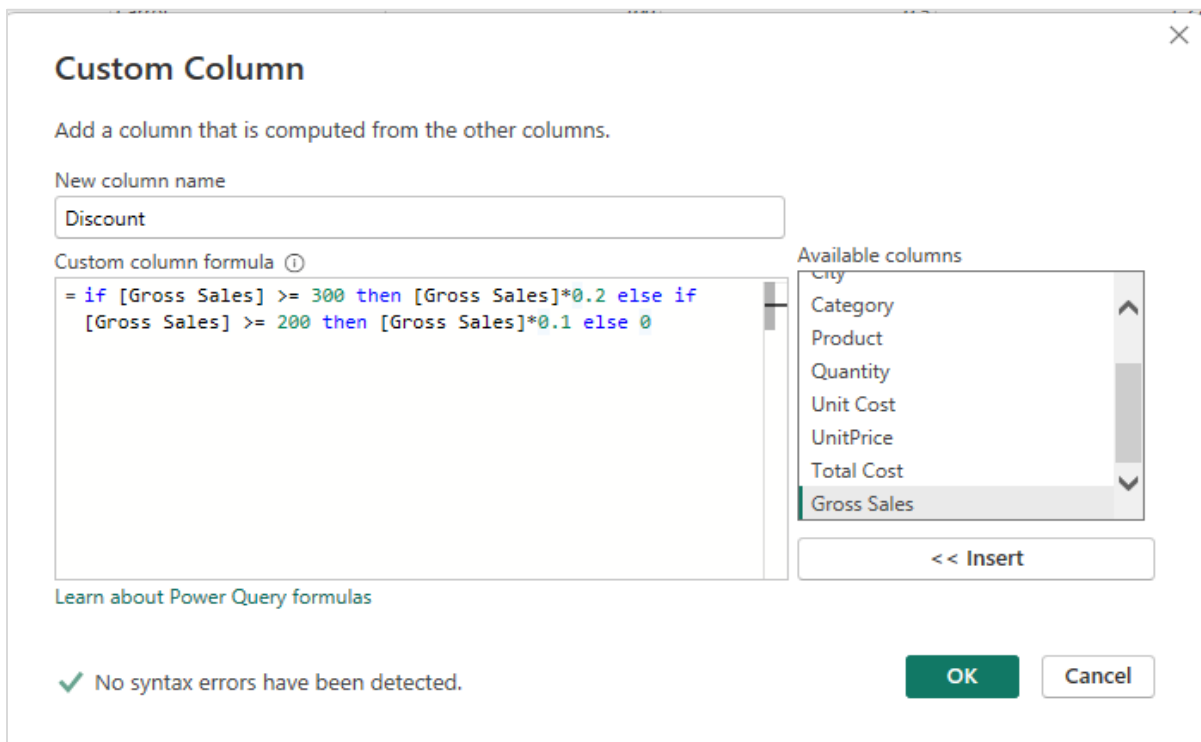
```
Table.AddColumn (PreviousStep,
    "Performance",
    each if [Profit] >= 300 then "Good"
        else if [Profit] >= 200 then "Average"
        else "Poor")
```

```
fx = Table.AddColumn(
    #"Added Custom4",
    "Performance",
    each if [Profit] >= 300 then "Good"
    else if [Profit] >= 200 then "Average"
    else "Poor")
```

	ABC 123	Net Sales		ABC 123	Profit		ABC 123	Performance	
0		58.41			41.91			Poor	
6		242.904			242.904			Average	
0		108.46			21.46			Poor	
0		153.34			30.34			Poor	
0		82.84			25.84			Poor	
0		95.58			68.58			Poor	
2		416.008			416.008			Good	
0		90.27			64.77			Poor	
0		177			127			Poor	
0		37.8			37.8			Poor	
0		78.48			24.48			Poor	
0		57.97			11.47			Poor	
0		97.72			97.72			Poor	
0		77.88			55.88			Poor	
0		40.71			29.21			Poor	
0		36.45			36.45			Poor	

Custom Column

In Power BI Desktop, you can add a new custom column of data to your model by using Power Query Editor. You can create and rename your custom column to create Power Query [M formula](#) queries to define your custom column.



Custom Column

Add a column that is computed from the other columns.

New column name
Discount

Custom column formula ⓘ
= if [Gross Sales] >= 300 then [Gross Sales]*0.2 else if [Gross Sales] >= 200 then [Gross Sales]*0.1 else 0

Available columns
City
Category
Product
Quantity
Unit Cost
Unit Price
Total Cost
Gross Sales

<< Insert

[Learn about Power Query formulas](#)

✓ No syntax errors have been detected.

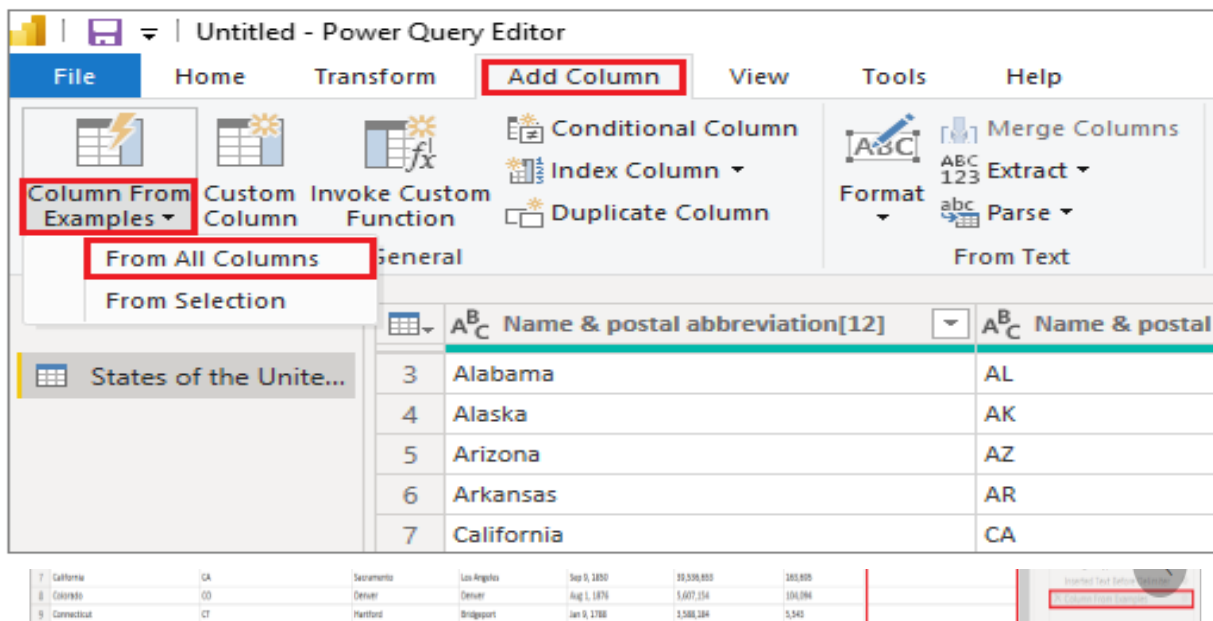
OK Cancel

Tips

- You can nest multiple if ... then ... else statements for complex logic.
- For better readability, use let ... in structure:

Column From Examples

Column from Examples in Power Query (Excel & Power BI) lets you create a new column by typing a few example outputs, and Power Query infers the transformation and generates the **M code** for you—similar to Flash Fill, but it writes reusable steps.



As you continue to provide examples, Power Query Editor adds to the transformations. When you are satisfied, select OK to commit your changes. You can rename the new column by double-clicking the column heading or by right-clicking it and selecting Rename.

How it works

- Type the desired output for a few rows in the preview column.
- Power Query detects a pattern (e.g., extract text before a dash, combine two columns, format dates).
- It proposes a transformation and shows the generated **M code**. You can accept and later edit this code in **Advanced Editor** or the formula bar.

Examples

1) Combine first & last name

Input columns: FirstName, Last Name

Type examples: Aisha Khan, Omar Salem

Resulting M code:

```
= Table.AddColumn("#Previous Step", "Full Name", each  
Text.Combine({[FirstName], [LastName]}, " "), type text)
```

2) Extract text before a delimiter

Input column: ProductCode values like ELEC-001, FURN-044

Example output: ELEC, FURN

Generated M:

```
Table.AddColumn("#Previous Step", "Category", each  
Text.BeforeDelimiter([ProductCode], "-"), type text)
```

3) Extract numeric suffix after dash

Input: INV-2025-000123 → **Output examples:** 000123

4) Format a date

Input column: OrderDate (type date)

Desired output: 2025/11 (YYYY/MM)

Generated M:

```
= Table.AddColumn("#Previous Step", "YearMonth", each  
    Text.Combine({Text.From(Date.Year([OrderDate])),  
    Text.PadStart(Text.From(Date.Month([OrderDate])), 2, "0")}, "/"),  
    type text)
```

Pivoting & Unpivoting Columns

Pivoting and unpivoting are essential techniques for reshaping data in Power Query in Excel or Power BI.

Pivoting

Pivoting means **turning rows into columns**. It's useful when you have categorical values in a column and want them as separate columns.

Example:

Product	Attribute	Value
A	Price	10
A	Qty	5
B	Price	20

After Pivoting on Attribute:

Product	Price	Qty
A	10	5
B	20	3

Steps in Power Query:

1. Select the column to pivot (e.g., Attribute).
2. Go to Transform → Pivot Column.
3. Choose the Values Column (e.g., Value).
4. Select an aggregation function (e.g., Sum, Average).

Unpivoting

Unpivoting means **turning columns into rows**. It's useful when you have multiple columns representing categories and want them in a single column.

Example:

Product	Price	Qty
A	10	5
B	20	3

After Unpivoting:

Product	Attribute	Value
A	Price	10
A	Qty	5
B	Price	20
B	Qty	3

Steps in Power Query:

1. Select the columns you don't want to unpivot (e.g., Product).
2. Go to Transform → Unpivot Columns.
3. Power Query creates two new columns: Attribute and Value.

When to Use Each?

- **Pivot:** When you want a wide format (categories as columns).
- **Unpivot:** When you want a long format (categories as rows).

Example Scenario

Product	Attribute	Value
A	Price	10
A	Qty	5
B	Price	20
B	Qty	3

We will **pivot** and then **unpivot** it.

In Excel Power Query

1. Load Data into Power Query

- Select your table → **Data** tab → **Get & Transform** → **From Table/Range**.

2. Pivot Columns

- In Power Query Editor:
 - Select **Attribute** column.
 - Go to **Transform** → **Pivot Column**.
 - In the dialog:
 - **Values Column** = Value.
 - **Aggregate Function** = Don't Aggregate (or Sum if needed).

Product	Price	Qty
A	10	5
B	20	3

Result.

3. Unpivot Columns

- Select columns you **don't want to unpivot** (e.g., Product).
- Go to **Transform → Unpivot Columns**.

In Power BI

1. Load Data

- Home → **Get Data → Excel** → Load your table.
- Click **Transform Data** to open Power Query.

2. Pivot Columns

- Same steps as Excel: **Transform → Pivot Column**.

3. Unpivot Columns

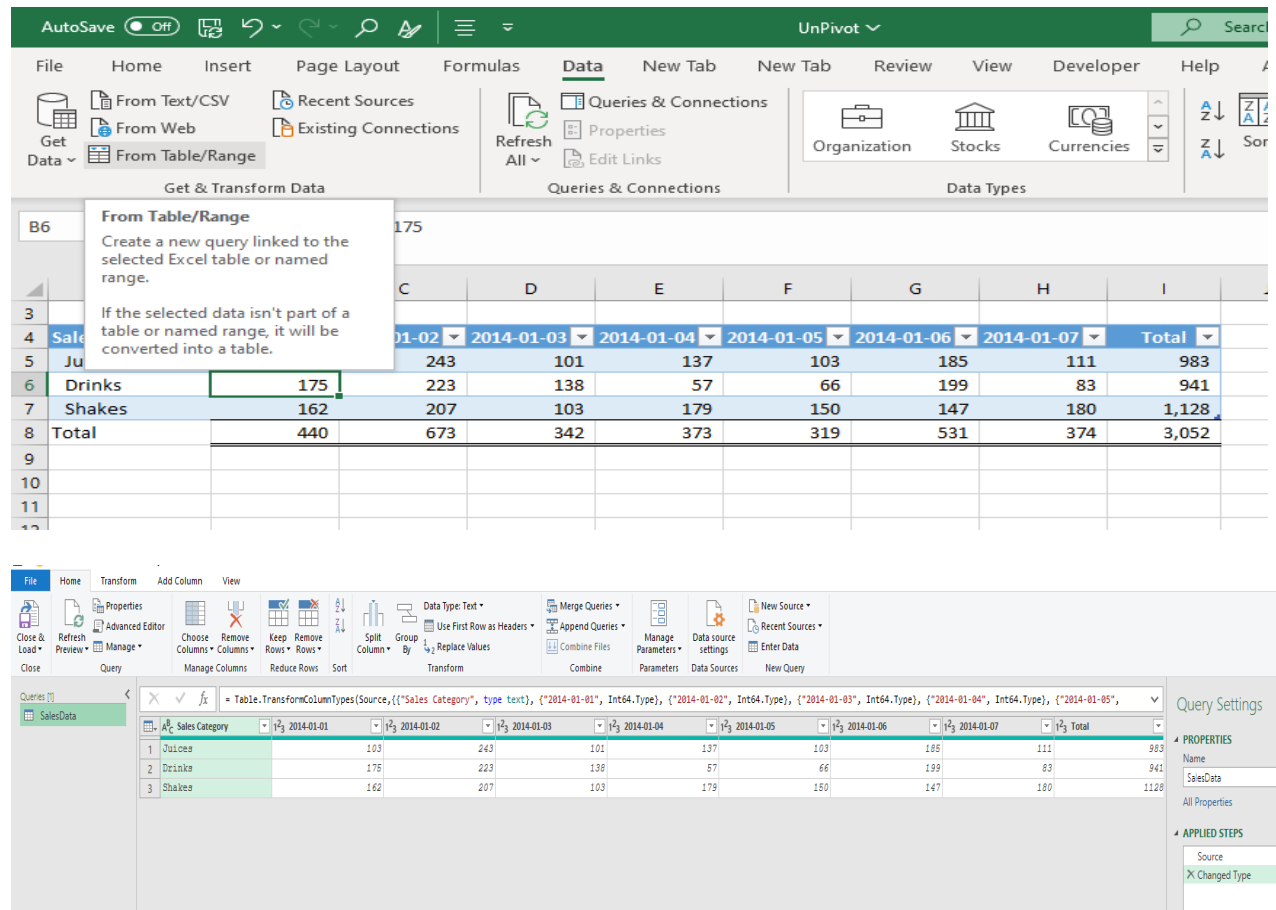
- Same steps: **Transform → Unpivot Columns**.

Product	Attribute	Value
A	Price	10
A	Qty	5
B	Price	20
B	Qty	3

★ Practical Example

Get Data from Table

Let's clean up this data:



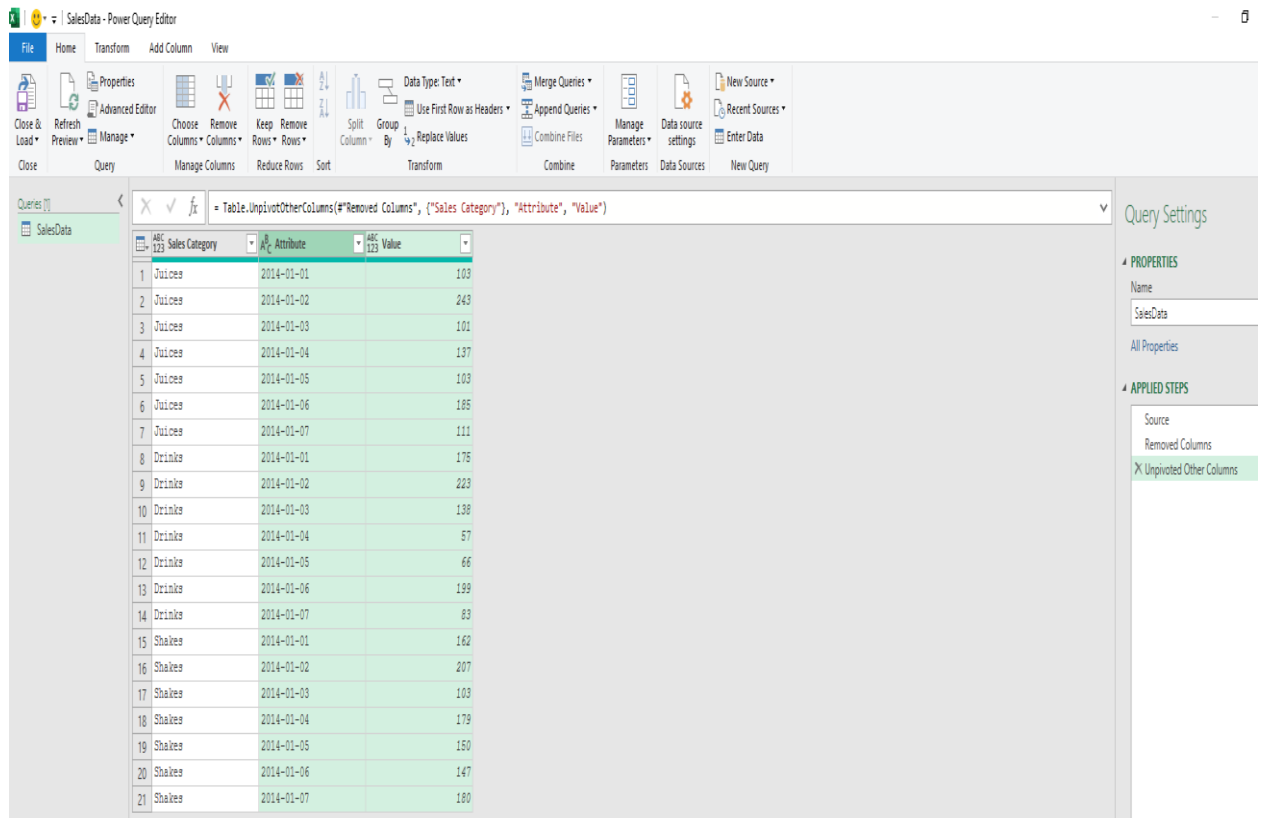
The screenshot shows the Microsoft Excel interface with the 'Data' tab selected. The 'Get Data' dropdown menu is open, and the 'From Table/Range' option is highlighted. A tooltip is displayed, explaining that this option creates a new query linked to the selected Excel table or named range. The background shows a table with columns for dates (2014-01-01 to 2014-01-07) and a 'Total' column, with rows for 'Juices', 'Drinks', 'Shakes', and a 'Total' row.

- Delete the 'Changed Type' step.
- Select the 'Total' column and remove it.

Un-pivoting Other Columns

Right click the sales category column and then unpivot other columns.

We can select multiple columns before unpivoting our data.



The screenshot shows the Power Query Editor interface. The formula bar at the top displays the M code: `= Table.UnpivotOtherColumns(#\"Removed Columns\", (\"Sales Category\"), \"Attribute\", \"Value\")`. The main data view shows a table with three columns: Sales Category, Attribute, and Value. The data is as follows:

Sales Category	Attribute	Value
Juices	2014-01-01	103
Juices	2014-01-02	243
Juices	2014-01-03	101
Juices	2014-01-04	137
Juices	2014-01-05	103
Juices	2014-01-06	165
Juices	2014-01-07	111
Drinks	2014-01-01	175
Drinks	2014-01-02	223
Drinks	2014-01-03	138
Drinks	2014-01-04	57
Drinks	2014-01-05	66
Drinks	2014-01-06	199
Drinks	2014-01-07	83
Shakes	2014-01-01	162
Shakes	2014-01-02	207
Shakes	2014-01-03	103
Shakes	2014-01-04	179
Shakes	2014-01-05	150
Shakes	2014-01-06	147
Shakes	2014-01-07	180

The right-hand pane shows the 'Query Settings' for 'SalesData'. Under 'APPLIED STEPS', the sequence is: Source, Removed Columns, and Unpivoted Other Columns (which is currently selected).

Following changes are required:

1. Change the name of Attribute and value columns as Date and units.
2. Change the data types for sales category, date and units to text, date and whole number.
3. Rename the query to sales.

Power Query Editor - Sales

Query: Sales

Formula Bar: = Table.TransformColumnTypes(#"Renamed Columns",{{("Sales Category", type text), ("Date", type date)}})

	Sales Category	Date	Units
1	Juices	01-Jan-14	103
2	Juices	02-Jan-14	243
3	Juices	03-Jan-14	101
4	Juices	04-Jan-14	137
5	Juices	05-Jan-14	103
6	Juices	06-Jan-14	185
7	Juices	07-Jan-14	111
8	Drinks	01-Jan-14	175
9	Drinks	02-Jan-14	223
10	Drinks	03-Jan-14	138
11	Drinks	04-Jan-14	57
12	Drinks	05-Jan-14	66
13	Drinks	06-Jan-14	199
14	Drinks	07-Jan-14	83
15	Shakes	01-Jan-14	162
16	Shakes	02-Jan-14	207
17	Shakes	03-Jan-14	103
18	Shakes	04-Jan-14	179
19	Shakes	05-Jan-14	150
20	Shakes	06-Jan-14	147
21	Shakes	07-Jan-14	180

Query Settings

PROPERTIES

Name: Sales

APPLIED STEPS

Source

Removed Columns

Unpivoted Other Columns

Renamed Columns

Changed Type

Re-pivoting thru a PivotTable

- Load the clean data into a new sheet and then build pivot tables.
- Put the cursor in any cell inside the Table and press Insert then PivotTable.

Excel - Juices

	Sales Category	Date	Units
1	Juices	01-01-14	103
2	Juices	02-01-14	243
3	Juices	03-01-14	101
4	Juices	04-01-14	137
5	Juices	05-01-14	103
6	Juices	06-01-14	185
7	Juices	07-01-14	111
8	Drinks	01-01-14	175
9	Drinks	02-01-14	223
10	Drinks	03-01-14	138
11	Drinks	04-01-14	57
12	Drinks	05-01-14	66
13	Drinks	06-01-14	199
14	Drinks	07-01-14	83
15	Shakes	01-01-14	162
16	Shakes	02-01-14	207
17	Shakes	03-01-14	103
18	Shakes	04-01-14	179
19	Shakes	05-01-14	150
20	Shakes	06-01-14	147
21	Shakes	07-01-14	180

PivotTable from table or range

Select a table or range

Table/Range: Sales

Choose where you want the PivotTable to be placed

☐ New Worksheet

☒ Existing Worksheet

Location: Sales!\$F\$1

Choose whether you want to analyze multiple tables

☐ Add this data to the Data Model

OK Cancel

Queries & Connections

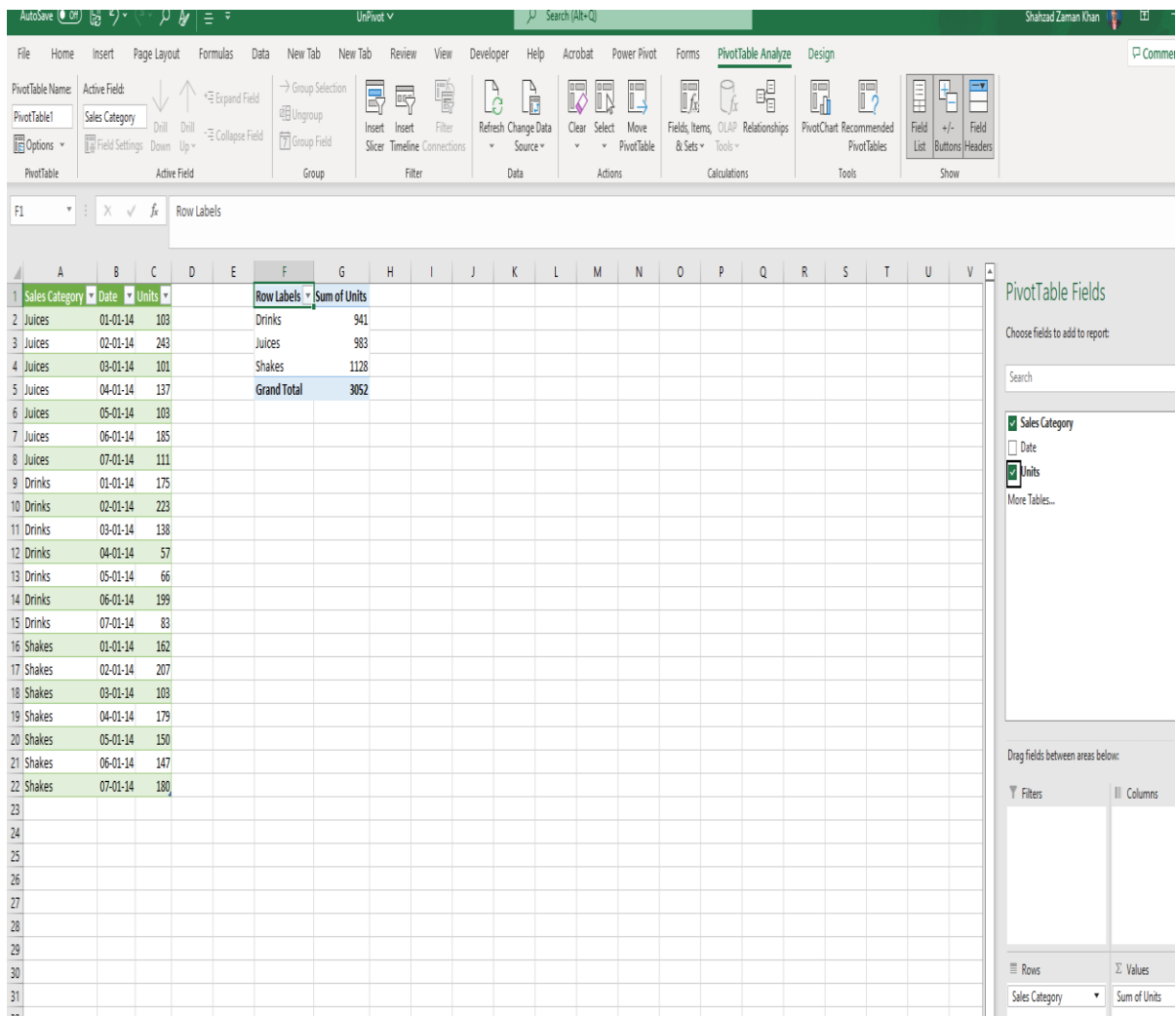
Queries

Connections

1 query

Sales

21 rows loaded.



The screenshot shows the Microsoft Excel interface with the 'PivotTable Analyze' tab selected. A PivotTable is displayed with 'Sales Category' as the Row Labels and 'Sum of Units' as the Values. The data is summarized by date and sales category.

	Sales Category	Date	Units
1	Juices	01-01-14	103
2	Juices	02-01-14	243
3	Juices	03-01-14	101
4	Juices	04-01-14	137
5	Juices	05-01-14	103
6	Juices	06-01-14	185
7	Juices	07-01-14	111
8	Drinks	01-01-14	175
9	Drinks	02-01-14	223
10	Drinks	03-01-14	138
11	Drinks	04-01-14	57
12	Drinks	05-01-14	66
13	Drinks	06-01-14	199
14	Drinks	07-01-14	83
15	Shakes	01-01-14	162
16	Shakes	02-01-14	207
17	Shakes	03-01-14	103
18	Shakes	04-01-14	179
19	Shakes	05-01-14	150
20	Shakes	06-01-14	147
21	Shakes	07-01-14	180
22			
23			
24			
25			
26			
27			
28			
29			
30			
31			

The PivotTable Fields task pane on the right shows the following configuration:

- Choose fields to add to report:** Sales Category, Date, Units.
- Drag fields between areas below:**
 - Filters:** (Empty)
 - Columns:** (Empty)
 - Rows:** Sales Category
 - Σ Values:** Sum of Units

Now we can return the file to the user and let them continue to update it as power query solutions can be refreshed at any time.

Once we get the sheet back, a new day and sales category has been added to data.

Go to sales worksheet and under data click 'Refresh All' button two times (once for query and once for PivotTables).

	A	B	C	D	E	F	G	H	I	J
3										
4	Sales Category ▼	2014-01-01 ▼	2014-01-02 ▼	2014-01-03 ▼	2014-01-04 ▼	2014-01-05 ▼	2014-01-06 ▼	2014-01-07 ▼	2014-01-08 ▼	Total ▼
5	Juices	103	243	101	137	103	185	111	120	983
6	Drinks	175	223	138	57	66	199	83	50	941
7	Smoothies	60	80	100	120	50	45	32	120	487
8	Shakes	162	207	103	179	150	147	180	40	1,128
9	Total	500	753	442	493	369	576	406	330	3,539
10										
11										
12										
13										
14										

Pivoting Data

When the data is fully unpivoted but what if we want to get it into a format that had distinct columns for Actual and Budget. That's where the Pivot function comes in play.

- Create a new query From sheet.
- Change the data type on the Dates column.... Date.....Replace.
- Update the query name to Sales.

All cleaning work is done, now it is time to switch it up so that we could get distinct columns for the Actual and Budget values.

- Select the Measure column.

- Go to Transform.....Pivot Column

Query Editor - Transform tab

Formula Bar: `= Table.TransformColumnTypes(Data_Table,{{"Category", type text}, {"Date", type date}, {"Measure", type text}, {"Units", Int64.Type}})`

	Category	Date	Measure	Units
1	Juices	01-Jan-21	Actual	200
2	Juices	01-Jan-21	Budget	150
3	Juices	02-Jan-21	Actual	50
4	Juices	02-Jan-21	Budget	200
5	Juices	03-Jan-21	Actual	100
6	Juices	03-Jan-21	Budget	0
7	Drinks	01-Jan-21	Actual	200
8	Drinks	01-Jan-21	Budget	200
9	Drinks	02-Jan-21	Actual	100
10	Drinks	02-Jan-21	Budget	0
11	Drinks	03-Jan-21	Actual	100
12	Drinks	03-Jan-21	Budget	0
13	Shakes	01-Jan-21	Actual	250
14	Shakes	01-Jan-21	Budget	150
15	Shakes	02-Jan-21	Actual	250
16	Shakes	02-Jan-21	Budget	0
17	Shakes	03-Jan-21	Actual	250
18	Shakes	03-Jan-21	Budget	100

Pivot Column

Use the names in column "Measure" to create new columns.

Values Column: Units

Advanced options
[Learn more about Pivot Column](#)

OK Cancel

Query Settings

- PROPERTIES
 - Name: Sales
 - All Properties
- APPLIED STEPS
 - Source
 - Navigation
 - Changed Type

Query Editor - Transform tab

Formula Bar: `= Table.Pivot(#"Changed Type", List.Distinct(#"Changed Type"[Measure]), "Measure", "Units", List.Sum)`

	Category	Date	Actual	Budget
1	Drinks	01-Jan-21	200	200
2	Drinks	02-Jan-21	100	50
3	Drinks	03-Jan-21	100	50
4	Juices	01-Jan-21	200	150
5	Juices	02-Jan-21	50	200
6	Juices	03-Jan-21	100	0
7	Shakes	01-Jan-21	250	150
8	Shakes	02-Jan-21	250	0
9	Shakes	03-Jan-21	250	100

Query Settings

- PROPERTIES
 - Name: Sales
 - All Properties
- APPLIED STEPS
 - Source
 - Navigation
 - Changed Type
 - Pivoted Column

Merging Two Queries / Tables in Power Query

We might need to merge two or more queries or tables in Power Query in Excel or Power BI when working with multiple data tables. The process of merging queries in Power Query is usually initiated by the need to combine related data. Often, relevant data is spread across different sources or tables.

Whether the data in each query comes from different sources, such as SQL Server and Excel, does not matter but having a common identifier is necessary. The queries can be combined based on the common key or identifier (like IDs or dates), allowing you to conduct a comprehensive analysis. For example, you might merge customer's information with their purchase history. In this case the common key could be customer ID.

In addition to a common key or identifier, you should also have a good understanding of the join type you want to apply before merging queries. Power Query gives you six join types to choose from and selecting the correct one is crucial, as each serves different purposes and affects the outcome of the merge.

Join Types

Following are the join types in power Query:

- Inner join.
- Left outer join.
- Right outer join.
- Full outer join.
- Left anti join.
- Right anti join.

1. Inner Join

- Keeps only matching rows from both tables based on the key column(s).
- If a row exists in Table A and Table B with the same key, it will appear in the result.
- **Result:** Intersection of both tables.

2. Left Outer Join

- Keeps all rows from the first (left) table and adds matching rows from the second table.
- If there's no match in the second table, columns from that table will show **null**.
- **Result:** All from left + matches from right.

3. Right Outer Join

- Opposite of Left Outer.
- Keeps all rows from the second (right) table and adds matching rows from the first table.
- Non-matching rows from the first table will show **null**.
- **Result:** All from right + matches from left.

4. Full Outer Join

- Keeps all rows from both tables, matching where possible.
- If a row exists only in one table, the other table's columns will show **null**.
- **Result:** Union of both tables.

5. Anti Joins

- **Left Anti Join:** Keeps rows from the first table that do NOT have a match in the second table.
- **Right Anti Join:** Keeps rows from the second table that do NOT have a match in the first table.
- **Result:** Non-matching rows only.

Summary Table

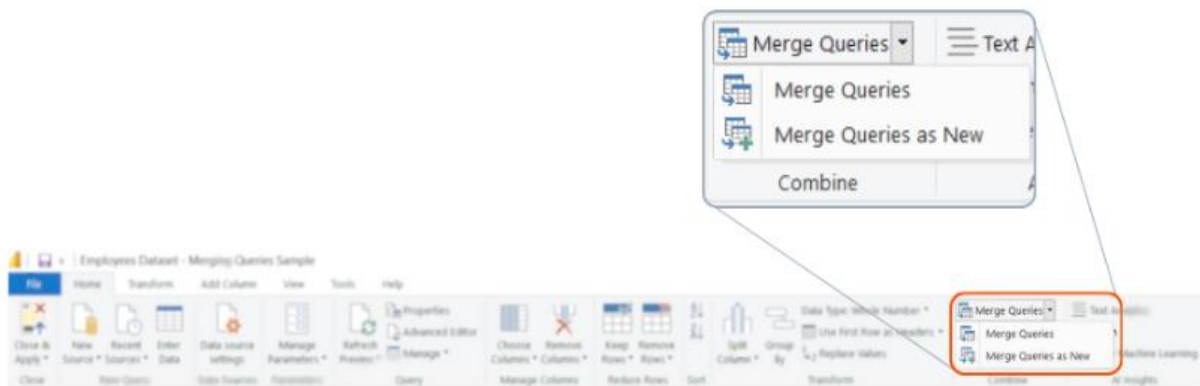
Sr	Join Type	Result
1	Inner Join	Matching rows (common) in both tables.
2	Left Outer	All from left + match from right
3	Right Outer	All from right + match from left (opposite of left outer)
4	Full Outer	All rows from both tables (union of both tables)
5	Left Anti	Left rows with no match in right.
6	Right Anti	Right rows with no match in left.

★ Practical Example

- **Scenario:** You have a customers table and a Sales table.
 - Customers
 - CustomerID
 - Name
 - Region
 - Sales
 - SaleID
 - CustomerID
 - Amount
 - Date
- Relationship: Customers [CustomerID] → Sales [CustomerID]
- **Options in Power BI:**
 - Use **Manage Relationships** to create a **one-to-many** relationship.
 - When building visuals:
 - **INNER JOIN behavior** happens by default when you use fields from both tables in a visual.
 - To simulate **LEFT JOIN**, use fields from Customers and add Sales measures (blank if no sale).
 - For **FULL OUTER JOIN**, you need to use **DAX** or merge queries in Power Query.

Steps to Merge Queries in Power Query

To merge two queries in Power Query, navigate to the **Home tab** and expand the **Merge Queries** option under the Combine section. This will present the option to Merge Queries or Merge Queries as New. The choice between these options depends on whether you prefer to merge the data into an existing query or to create a new, separate query for the merged data.



Next, you will need to select one of the six Join kinds available from Power Query. Finally, click OK to finalize the merge of the two queries in Power Query.

The last step when merging queries in Power Query is to expand the resulting table to display the desired columns from the joined query. After clicking OK in the Merge dialog box, you will notice that the new 'merged column' displays results as a table. The final step is to expand this table.

To expand the table, click the icon on the top right of the column to choose which columns to expand. By default, all columns are selected but you can use this dialog box to choose only the columns you need for your analysis.

Queries [2]

Sales Transactions

Inventory

Merge

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

Sales Transactions

Date	No	Item	Units
3/12/2022	510005	Bars	50
3/13/2022	510010	Cookies	40
3/14/2022	510014	Crackers	60
3/15/2022	510019	Snacks	50
3/16/2022	510019	Snacks	20

Inventory

No	Item	Type	Unit	Size	Price	Cost	Profit
510005	Bars	VIP	Packs	25	28.4	20.4	8
510010	Cookies	VIP	Packs	5	12.5	8.5	4
510014	Crackers	Economical	Packs	5	10.5	9.5	1
510019	Juices	VIP	Bottles	5	10.2	10.25	-0.05
510021	Bars	VIP	Bottles	2	4.5	3.7	0.8

Join Kind

Left Outer (all from first, matching from second)

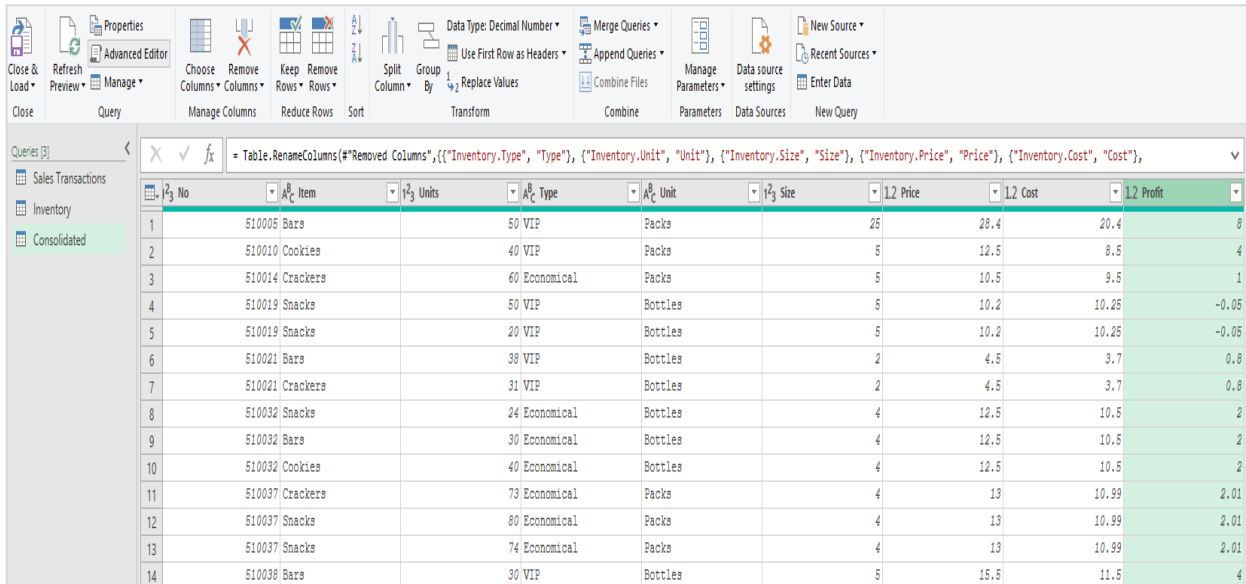
☐ Use fuzzy matching to perform the merge

> Fuzzy matching options

✓ The selection matches 20 of 20 rows from the first table.

OK

Cancel



No	Item	Units	Type	Unit	Size	Price	Cost	Profit
1	510005 Bars	50	VIP	Packs	25	28.4	20.4	8
2	510010 Cookies	40	VIP	Packs	5	12.5	8.5	4
3	510014 Crackers	60	Economical	Packs	5	10.5	9.5	1
4	510019 Snacks	50	VIP	Bottles	5	10.2	10.25	-0.05
5	510019 Snacks	20	VIP	Bottles	5	10.2	10.25	-0.05
6	510021 Bars	38	VIP	Bottles	2	4.5	3.7	0.8
7	510021 Crackers	31	VIP	Bottles	2	4.5	3.7	0.8
8	510032 Snacks	24	Economical	Bottles	4	12.5	10.5	2
9	510032 Bars	30	Economical	Bottles	4	12.5	10.5	2
10	510032 Cookies	40	Economical	Bottles	4	12.5	10.5	2
11	510037 Crackers	73	Economical	Packs	4	13	10.99	2.01
12	510037 Snacks	80	Economical	Packs	4	13	10.99	2.01
13	510037 Snacks	74	Economical	Packs	4	13	10.99	2.01
14	510038 Bars	30	VIP	Bottles	5	15.5	11.5	4

Merging datasets in Power Query in Excel and Power BI is a powerful technique that allows analysts to combine related data and uncover deeper insights by providing a holistic view of the data.

Append Query

Steps in Power Query in Excel / Power BI:

1. **Load your tables** into Power Query (e.g., Sales tables Jan-25 and Mar-25).
2. Go to **Home → Append Queries**:
 - **Append Queries** (adds to existing query)
 - **Append Queries as New** (creates a new combined query)
3. Select the tables you want to append:
 - For **two tables**, choose them in the dialog.
 - For **three or more tables**, select **Three or more tables** and pick all.
4. Click **OK** → You'll get a combined table with all rows stacked.

Case Study 1

Match the customer invoices to find duplicate invoices.

Suppose you have two excel sheets, detail of customer invoices from company's ledger and from head office. Now you want to match the two sheets to find the duplicate invoices or to reconcile difference in the amounts.

 **Tip:** Use the merge queries option and apply following steps in Power Query.

- 1) Load both sheets into Power Query.
- 2) Merge queries on Invoice Number.
- 3) Choose Inner Join for duplicates or Full Outer Join for all records.
- 4) Add a custom column to compare amounts.

Case Study 2

Automate Monthly TB using Power Query.

Power Query in Excel and Power BI is a powerful tool for automating the import, transformation, and loading of monthly trial balance data. Here is how you can use it to streamline your accounting workflow:

Step 1: Connect to Your Data Source

If your trial balance is saved monthly in a workbook. You can connect Power Query using:

- ✓ Data Tab
- ✓ Get Data
- ✓ From Other Sources
- ✓ Blank Query

Step 2: Transform the Data

- ✓ Remove unnecessary columns.
- ✓ Change data types (e.g., dates, numbers).
- ✓ Rename columns for consistency.

Step 3: Complete the Process

- ✓ Click the small arrow icon at right in Content column.
- ✓ Uncheck “use original column name as prefix”.

Step 4: Load to Excel or Power BI

- ✓ In Excel, load the cleaned data into a table or PivotTable.
- ✓ In Power BI, load it into your data model for reporting and visualization.

Step 5: Refresh Monthly

- ✓ Add the monthly trial balance in Excel workbook.
- ✓ Refresh it.

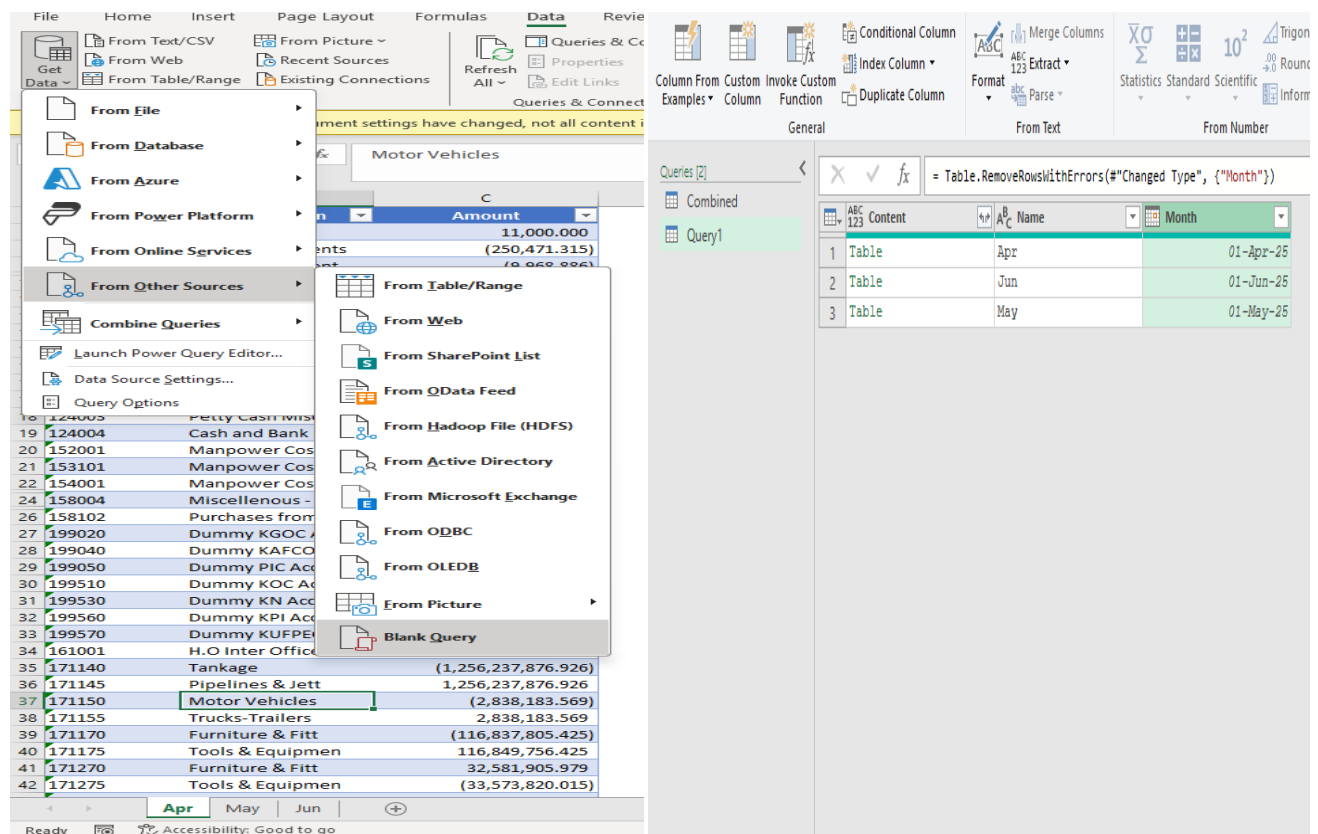
Step 6: Monthly Reporting & Analysis

- ✓ Monthly Pivot Table Reports using Slicers.
- ✓ Perform Data Analysis.

Step 1 CONNECT TO DATA SOURCE

- ✓ Data Tab
- ✓ Get Data
- ✓ From Other Sources

Blank query



The screenshot shows the Microsoft Excel interface with the 'Data' tab selected. The 'Get Data' dropdown menu is open, showing various data sources. The 'From Other Sources' option is highlighted, and the 'Blank Query' option is selected. The background shows a trial balance table with columns for Date, Description, and Amount.

Date	Description	Amount
124005	Petty Cash	11,000.000
124005	Cash and Bank	(250,471.315)
124005	Manpower Cos	(9,968.895)
124005	Manpower Cos	
124005	Manpower Cos	
124005	Miscellaneous -	
124005	Purchases from	
124005	Dummy KGOCA	
124005	Dummy KAFCA	
124005	Dummy PIC Acc	
124005	Dummy KOC Acc	
124005	Dummy KN Acc	
124005	Dummy KPI Acc	
124005	Dummy KUFPEI	
124005	H.O Inter Office	
124005	Tankage	(1,256,237,876.926)
124005	Pipelines & Jett	1,256,237,876.926
124005	Motor Vehicles	(2,838,183.569)
124005	Trucks-Trailers	2,838,183.569
124005	Furniture & Fitt	(116,837,805.425)
124005	Tools & Equipmen	116,849,756.425
124005	Furniture & Fitt	32,581,905.979
124005	Tools & Equipmen	(33,573,820.015)

Note: I have loaded the trial balance for the three months, Apr25, May-25 & Jun-25 to automate the monthly trial balance reports.

Step 2 Data Transformation

- ✓ Add custom column.
- ✓ Click the small arrow icon at right in Content column.
- ✓ Uncheck “use original column name as prefix”.

Custom Column

Add a column that is computed from the other columns.

New column name

Custom column formula ①

Available columns

Content

Name

<< Insert

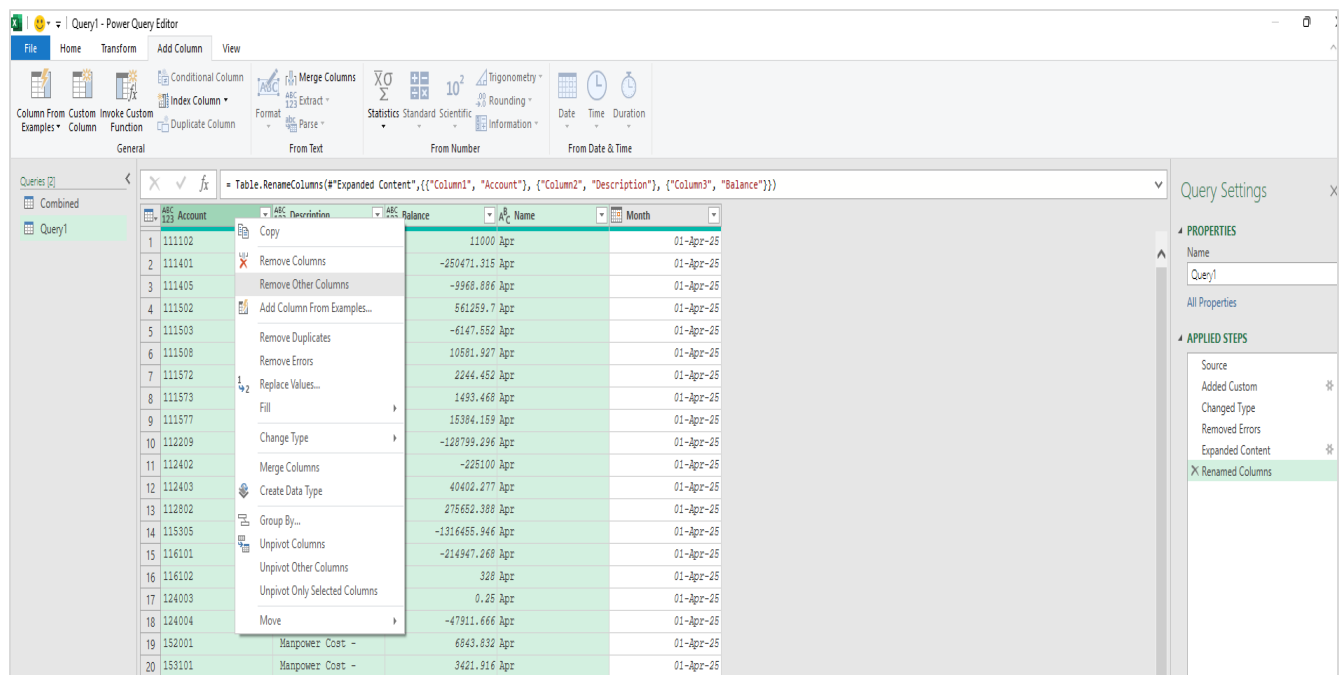
[Learn about Power Query formulas](#)

✓ No syntax errors have been detected.

OK Cancel

Step 3

- ✓ Select all columns except Month then click “Remove Other Columns”.



Query1 - Power Query Editor

Table: `Table.RenameColumns(#Expanded Content,{{("Column1", "Account"), ("Column2", "Description"), ("Column3", "Balance")}})`

	Account	Description	Balance	Name	Month
1	111102		11000	Apr	01-Apr-25
2	111401		-250471.315	Apr	01-Apr-25
3	111405		-9968.886	Apr	01-Apr-25
4	111502		561259.7	Apr	01-Apr-25
5	111503		-6147.552	Apr	01-Apr-25
6	111508		10581.927	Apr	01-Apr-25
7	111572		2244.452	Apr	01-Apr-25
8	111573		1493.468	Apr	01-Apr-25
9	111577		15384.159	Apr	01-Apr-25
10	112209		-128799.296	Apr	01-Apr-25
11	112402		-225100	Apr	01-Apr-25
12	112403		40402.277	Apr	01-Apr-25
13	112802		275652.388	Apr	01-Apr-25
14	115305		-1316455.946	Apr	01-Apr-25
15	116101		-214947.268	Apr	01-Apr-25
16	116102		328	Apr	01-Apr-25
17	124003		0.25	Apr	01-Apr-25
18	124004		-47911.666	Apr	01-Apr-25
19	152001	Manpower Cost -	6843.832	Apr	01-Apr-25
20	153101	Manpower Cost -	3421.916	Apr	01-Apr-25

Query Settings

PROPERTIES
 Name
 Query1

APPLIED STEPS
 Source
 Added Custom
 Changed Type
 Removed Errors
 Expanded Content
 X Renamed Columns

Step 4 Close & Load

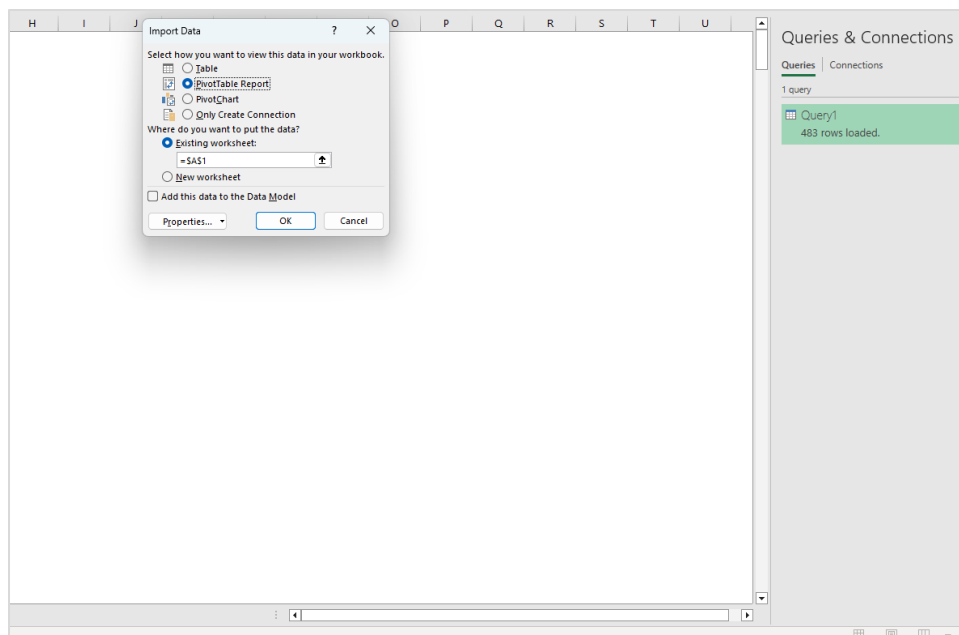
- ✓ Select Close & Load.
- ✓ Load to Table

	A	B	C	D
1	Account	Description	Balance	Name
3	111401	H.O Payments	-250471.315	Apr
11	112209	Advances Against	-128799.296	Apr
12	112402	Travel Advances	-225100	Apr
13	112403	Employees Person	40402.277	Apr
14	112802	Accrued Intrerst	275652.388	Apr
15	115305	Inventory Final	-1316455.946	Apr
16	116101	Stock Spares and	-214947.268	Apr
17	116102	Direct Charge Ma	328	Apr
18	124003	Petty Cash Misc.	0.25	Apr
19	124004	Cash and Bank Mi	-47911.666	Apr
20	152001	Manpower Cost -	6843.832	Apr
23	158002	Training - KPC C	-93814.12	Apr
24	158004	Miscellenous - K	-4153.099	Apr
26	158102	Purchases from K	-291081372.4	Apr
34	161001	H.O Inter Office	316741917.2	Apr
35	171140	Tankage	-1256237877	Apr
36	171145	Pipelines & Jett	1256237877	Apr
37	171150	Motor Vehicles	-2838183.569	Apr
38	171155	Trucks-Trailers	2838183.569	Apr
39	171170	Furniture & Fitt	-116837805.4	Apr
40	171175	Tools & Equipmen	116849756.4	Apr
41	171270	Furniture & Fitt	32581905.98	Apr
42	171275	Tools & Equipmen	-33573820.02	Apr
43	211203	Interest Payable	-925941.837	Apr
44	214001	Employer Contrib	-2844.316	Apr
45	214002	Employer Contrib	-216.719	Apr
46	216101	Workmen Compensa	-68248.996	Apr
47	216301	Electricity - Ac	-437947.44	Apr
48	216302	Fresh Water - Ac	-31162.065	Apr
49	216303	Desalinated Wate	-120033.54	Apr
50	216902	Contracts Retent	-19221.178	Apr
51	216903	Retention For Ta	-31818.945	Apr
52	216905	Insurance Claim	-1647.469	Apr
53	216913	Penalty On Contr	-370	Apr

Ready Accessibility: Investigate

Step 5 Load to Pivot Table Report

- ✓ Right click on the Combined query in the queries & Connections Pane.
- ✓ Load to Pivot Table Report



The screenshot shows the 'Import Data' dialog box open over an Excel spreadsheet. The dialog box has the following options:

- Select how you want to view this data in your workbook:**
 - ☐ Table
 - ☒ PivotTable Report
 - ☐ PivotChart
 - ☐ Only Create Connection
- Where do you want to put the data?**
 - ☒ Existing worksheet:
 - Sheet: \$A\$1
 - ☐ New worksheet
- ☐ Add this data to the Data Model

Buttons at the bottom: Properties..., OK, Cancel.

On the right, the 'Queries & Connections' pane shows a single query named 'Query1' with the status '483 rows loaded.'

Step 6 Select Pivot Table Fields

- ✓ Drag Amount in Values, Description in Rows and Month in Columns.

	A	B	C	D	E
1	Sum of Balance	Column Labels			
2	Row Labels	Apr	May	Jun	Grand Total
11	Allowances		7220	25120	32340
12	AP Suspense/Maxi		-72657.902		-72657.902
13	Bank Charges	54,578	-54,578		0
14	Basic Pay		2522955.441		2522955.441
15	Building		480		480
16	Building and Roa		0		0
17	Buses	26896	26896		53792
18	Call Account - D		-11472.941	-11088.681	-22561.622
19	Call Account (US		-5560.776	-7577.659	-13138.435
20	Call Account Eur	-481.814	-795.063	-1612.355	-2889.232
21	Capex WIP - Hea		56614	-944038	-887424
22	Capex WIP - AZ		377198.073		377198.073
24	Capex WIP - P		34531.091	26508.555	61039.646
26	Catalyst		-3999487.517		-3999487.517
27	Catalyst Loading		357570		357570
28	Catering Service		56156.468		56156.468
29	Chemical and Gas		-13990.85		-13990.85
30	Chemicals	348393.164	52616.12		401009.284
31	Child Allowance	110578.679	108736.452		219315.131
32	Civil - Cable Wo			40361.594	40361.594
33	Clearing - Curre		-28725.127		-28725.127
34	Communication Se		42362.147		42362.147
35	Computer System		173275.462		173275.462
36	Computers and Co	27025.287	0	122384.086	149409.373
37	Consultant Fees		91884.515		91884.515
38	Contra A-C Req.		3820536.72	24266642.84	28087179.56
39	Contracts Retent	-19221.178	928973.866		909752.688
40	Cost of Living A		125920.299		125920.299
41	CPQ Revenue Acco		-18721.565		-18721.565
42	Credit And Savin		0		0
43	Demurrage - Paya	54809.846	361299.501		416109.347
44	Deposit Interest	-275652.388	-237384.292	478157.786	-34878.894
45	Desalinated Wate	-120033.54	0	126709.605	6676.065

PivotTable Fields

Choose fields to add to report:

Search

- ☐ Account
- ☒ Amount
- ☒ Description
- ☒ Month

More Tables...

Drag fields between areas below:

Filters

Columns

Month

Rows

Description

Values

Sum of Amount

☐ Defer Layout Update
 Update

Step 7

- ✓ Remove Grand Total Column.
- ✓ Take Amount again in values.
- ✓ Pres CTRL and Select Apr and May then Right click on Sum of Amount.

	A	B	C	D	E	F	G	H	I	J	K
1	Sum of Balance	Column Labels							Month		
2	Row Labels	Apr	May	Jun					Apr		
11	Allowances		7,220	25,120					May		
12	AP Suspense/Maxi		(72,658)						Jun		
13	Bank Charges	55	(55)								
14	Basic Pay		2,522,955								
15	Building		480								
16	Building and Roa		-								
17	Buses	26,896	26,896								
18	Call Account - D		(11,473)	(11,089)							
19	Call Account (US		(5,561)	(7,578)							
20	Call Account Eur	(482)	(795)	(1,612)							
21	Capex WIP - Hea		56,614	(944,038)							
22	Capex WIP - AZ		377,198								
24	Capex WIP - P		34,531	26,509							
26	Catalyst		(3,999,488)								
27	Catalyst Loading		357,570								
28	Catering Service		56,156								
29	Chemical and Gas		(13,991)								
30	Chemicals	348,393	52,616								
31	Child Allowance	110,579	108,736								
32	Civil - Cable Wo			40,362							
33	Clearing - Curre		(28,725)								
34	Communication Se		42,362								
35	Computer System		173,275								
36	Computers and Co	27,025	-	122,384							
37	Consultant Fees		91,885								
38	Contra A-C Req.		3,820,537	24,266,643							
39	Contracts Retent	(19,221)	928,974								
40	Cost of Living A		125,920								
41	CPQ Revenue Acco		(18,722)								
42	Credit And Savin		-								
43	Demurrage - Paya	54,810	361,300								
44	Deposit Interest	(275,652)	(237,384)	478,158							
45	Desalinated Wate	(120,034)	-	126,710							

Show Values As (Sum of Amount2)
?
X

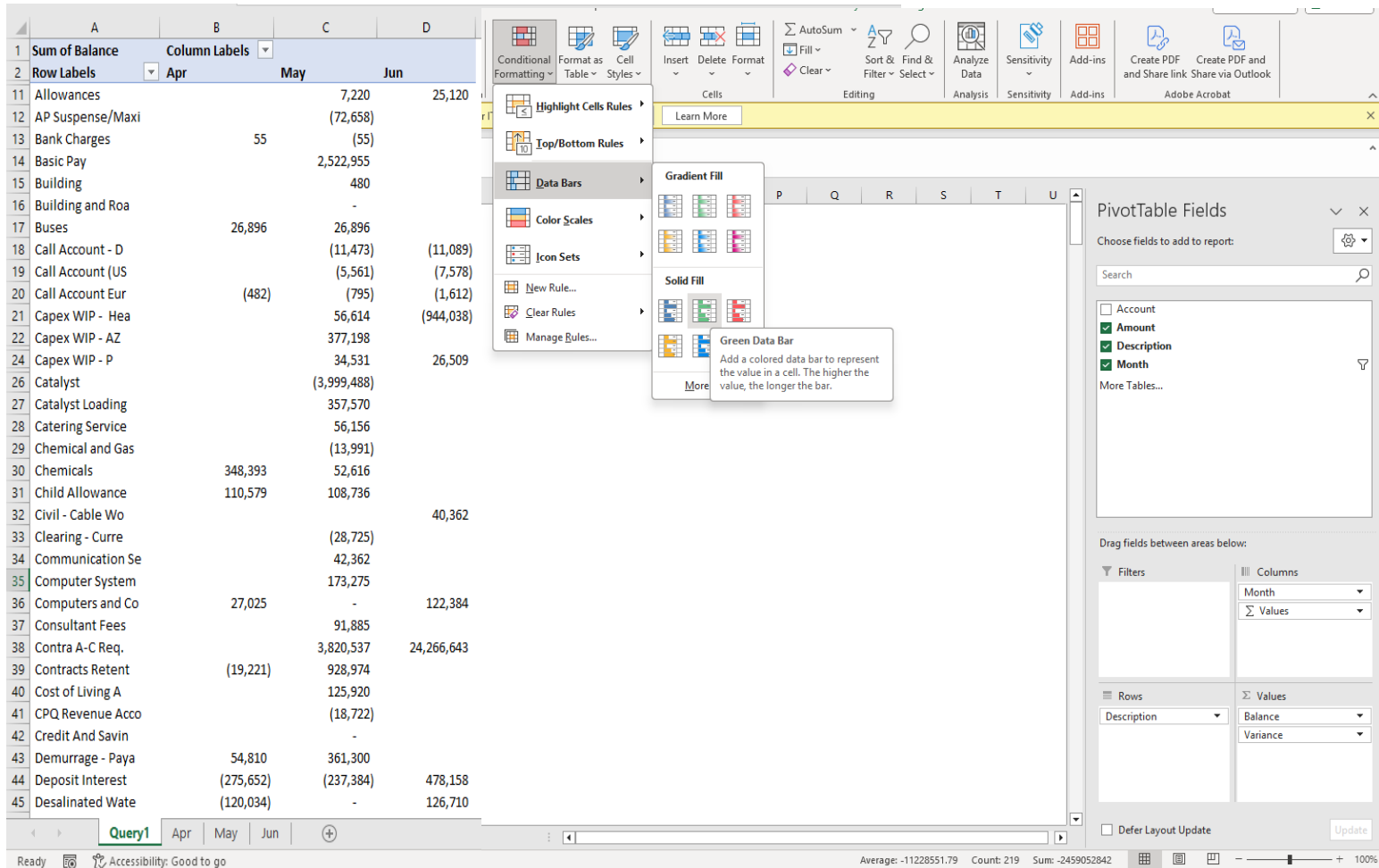
Calculation: Difference From

Base Field: Month

Base Item: (previous)

OK Cancel

Step 8: Apply Conditional Formatting.



The screenshot displays the Microsoft Excel interface. The main area shows a PivotTable with the following data:

Row Labels	Apr	May	Jun
Allowances		7,220	25,120
AP Suspense/Maxi		(72,658)	
Bank Charges	55	(55)	
Basic Pay		2,522,955	
Building		480	
Building and Roa		-	
Buses	26,896	26,896	
Call Account - D		(11,473)	(11,089)
Call Account (US		(5,561)	(7,578)
Call Account Eur	(482)	(795)	(1,612)
Capex WIP - Hea		56,614	(944,038)
Capex WIP - AZ		377,198	
Capex WIP - P		34,531	26,509
Catalyst		(3,999,488)	
Catalyst Loading		357,570	
Catering Service		56,156	
Chemical and Gas		(13,991)	
Chemicals	348,393	52,616	
Child Allowance	110,579	108,736	
Civil - Cable Wo			40,362
Clearing - Curre		(28,725)	
Communication Se		42,362	
Computer System		173,275	
Computers and Co	27,025	-	122,384
Consultant Fees		91,885	
Contra A-C Req.		3,820,537	24,266,643
Contracts Retent	(19,221)	928,974	
Cost of Living A		125,920	
CPQ Revenue Acco		(18,722)	
Credit And Savin		-	
Demurrage - Paya	54,810	361,300	
Deposit Interest	(275,652)	(237,384)	478,158
Desalinated Wate	(120,034)	-	126,710

The Conditional Formatting ribbon is open, showing options for Highlight Cells Rules, Top/Bottom Rules, Data Bars, Color Scales, and Icon Sets. A 'Green Data Bar' tooltip is visible, explaining that it adds a colored data bar to represent the value in a cell. The PivotTable Fields task pane on the right shows the following configuration:

- Filters:** None
- Columns:** Month, Σ Values
- Rows:** Description
- Σ Values:** Balance, Variance

The status bar at the bottom shows: Ready, Accessibility: Good to go, Average: -11228551.79, Count: 219, Sum: -2459052842, 100%.

Step 9 Apply CF for Month over Month Analysis

- ✓ Go to Conditional Formatting.
- ✓ Manage Rules then Edit Rule.
- ✓ Select last option so that CF will be applied for other months variance too.

	A	B	D	E
64	Employee Housing	811.666	1100	288.334
65	Employee Leave S		372.689	372.689
66	Employee Loans -		17334.141	17334.141
67	Employees Car Fu		20552.92	20552.92
68	Employees Car Re		109833.935	109833.935
69	Employees Person	40402.277	7310.936	-33091.341
70	Employer Contrib	3061.035	273.494	-2787.541
71	Ex Gratia Paymen		218598	218598
72	Exchange Rate Pr		1626111.047	1626111.047
73	Exchange Rate Va		5156.146	5156.146
74	Financial Suppor	68995.048	67852.365	-1142.583
75	Freight - Keros	1732595.952	1733381.158	785.206
76	Freight - Diesel	3365617.953	4048651.585	683033.532
77	Freight - Sulphu	342616.489		-342616.489
78	Fresh Water		33229.68	33229.68
79	Fresh Water - Ac	31162.065	33229.68	2067.515
80	Fuel - KNPC Curr		27906.215	27906.215
81	Furniture & Fitt	149437054.5	34644.342	-14940240.2
82	Furniture Grant		8750	8750
83	General - Consol		167534.371	167534.371
84	General Statione	450.925		-450.925
85	Gifts		4848.35	4848.35
86	Gross Revenue -	40826170.78	386181692.3	34535552.15
87	Group Term Life	68248.996	66028.223	-2220.773
88	H.O Inter Office	316741917.2	397942966.8	81201049.64
89	H/O Deferred Stu		7437.205	7437.205
90	Head Office Inte	313124651.7	315376865.7	2252213.935
91	Housing Allowanc		213754.184	213754.184
92	Housing Grant	142618.292	150977.548	8359.256
93	Incentive Allowa	976777.328	956467.152	-20310.176
94	Increase In COLA	168518.672	165095.678	-3422.994
95	Instrument Maint		338334.87	338334.87
96	Insurance - Dies	8833.802	12959.585	4125.783
97	Insurance - Kero	3886.337	5571.259	1684.922
98	Insurance Claim	1647.469	58066.754	56419.285

Conditional Formatting Rules Manager

Show formatting rules for: This PivotTable

New Rule... Edit Rule... Delete Rule Duplicate Rule

Rule (applied in order shown)	Format	Applies to	Stop If True
Data Bar		=SE\$4:SE\$222	☐

Edit Formatting Rule

Apply Rule To: =SE\$4

☐ Selected cells
☐ All cells showing "Variance" values
☒ All cells showing "Variance" values for "Description" and "Month"

Select a Rule Type:

Format all cells based on their values
Format only cells that contain
Format only top or bottom ranked values
Format only values that are above or below average
Use a formula to determine which cells to format

Edit the Rule Description:

Format all cells based on their values:
Format Style: Data Bar ☐ Show Bar Only

Type: Automatic Minimum Automatic Maximum Automatic

Value: (Automatic)

Bar Appearance:
Fill: Solid Fill Color: Border: No Border Color:

Negative Value and Axis... Bar Direction: Context

Preview:

OK Cancel

Step 10 Update Report by Adding New Month

- ✓ Add a new sheet as July.
- ✓ Go to Data Tab and click Refresh All.
- ✓ The July sheet will be added in the data and slicer.

1	Column Labels		
2	May	Jun	
3	Row Labels	Balance	Variance
5	Absence Day Adju	6252.651	-6252.651
6	Accrued Interest	172434.942	-172434.942
7	Accrued Interst	64949.35	-64949.35
8	Advances Against	256821.819	-256821.819
9	Advertisements	40	-40
10	Air Fares - BT	418.75	-418.75
11	All Payments - A	749336.642	-749336.642
12	Allowances	7220	25120 17900
13	AP Suspense/Maxi	72657.902	-72657.902
14	Bank Charges	54.578	-54.578
15	Basic Pay	2522955.441	-2522955.441
16	Building	480	-480
17	Building and Roa	6905814.32	-6905814.32
18	Buses	26896	-26896
19	Call Account - D	11472.941	11088.681 -384.26
20	Call Account (US	5560.776	7577.659 2016.883
21	Call Account Eur	795.063	1612.355 817.292
22	Capex WIP - Hea	56614	944038 887424

Note: Once you click Refresh, a new month July TB will be added to the data model and slicer. So, by using Power Query in Excel or Power BI, monthly trial balance can be updated on a just single click.

5. BASIC DATA MODELING

Data modeling is the process of defining the data structure, properties, and relationships within a data model. A data model in Power BI is a logical representation of how data is structured and related within the tool. It is a collection of tables and relationships between them that are used to create reports and visualizations.

Data modelling is one of the most important aspects of data analysis, regardless of the tools we use. After we import data into the data model, we need to create and manage relationships, creating analytical calculations and implement the business logics available for data visualization.

When talking about data modeling, the term that is most often used is the [star schema](#). This is the recommended approach to shape and structure data in Power BI. In this schema, data tables (or fact tables) for example, Sales table are surrounded by lookup or dimension tables such as Products, Regions, Calendar etc.

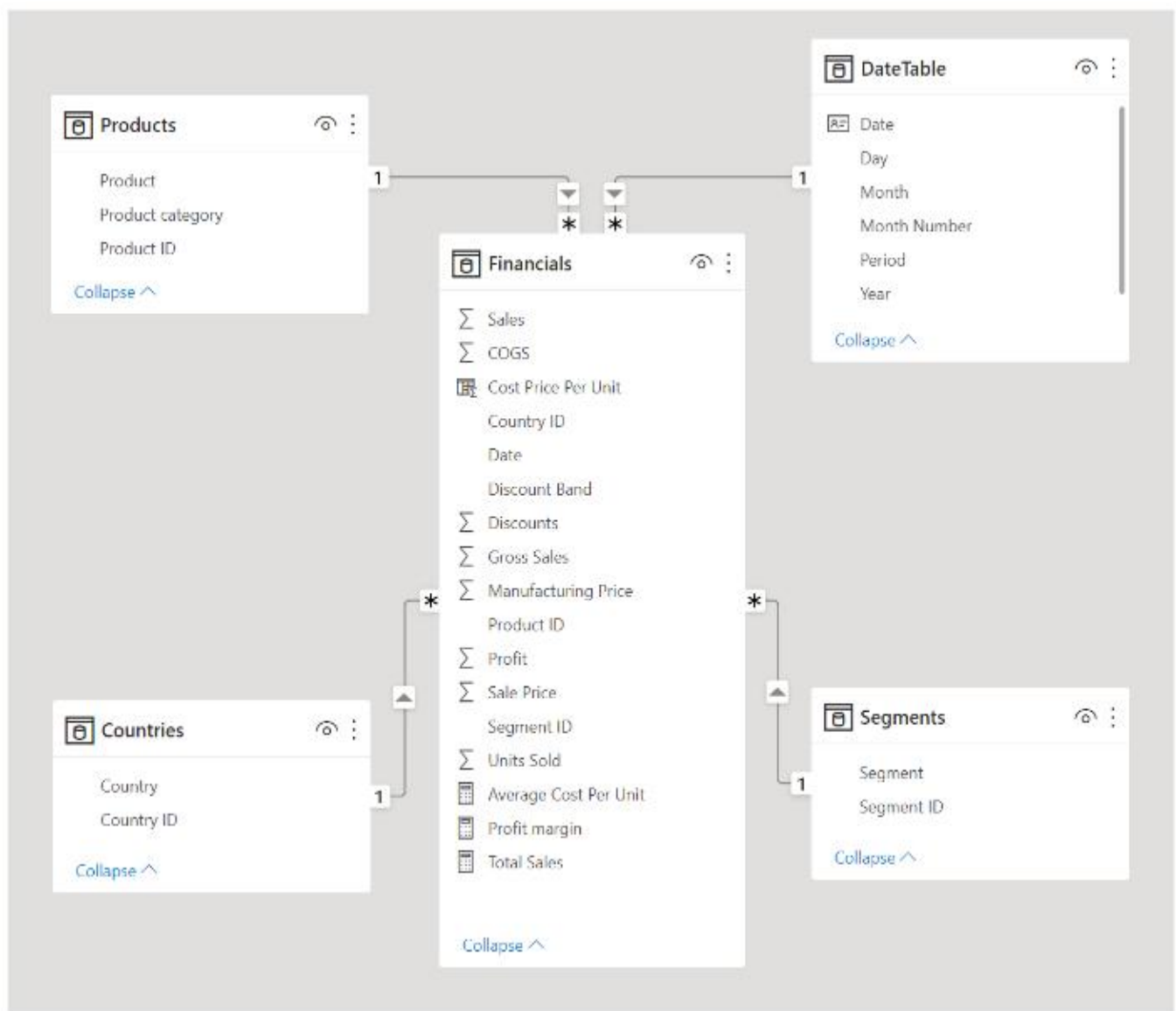
There are two main benefits of using a star schema:

- **Usability:** A star schema makes your data model cleaner and more organized, and your report will be easier to use.
- **Performance:** Star schemas allow you to easily scale your report to very large volumes of data. DAX measures also calculate faster, and the Power BI report refreshes faster overall.

Relationships in Data Models

Star schema

A star schema is made up of a central fact or data table with multiple dimension or lookup tables branching off this fact table, much like the appearance of a star. Ideally, it is good practice to have only one fact table in a data model. However, it is possible to include multiple fact tables in Power BI.



A [data or fact table](#) is made up of values that can be summarized and aggregated, as well as one or more keys that link to the dimension tables. The purpose of the key is to make connections between tables. If you have complex data models or large volumes of data, it is a good idea for the keys to be numeric values rather than text strings, as this can improve the performance of the model.

Fact tables should be very narrow and include as few columns as possible. Any columns that describe the features of a dataset, such as Product Name or Product Category, should be separated out of the fact table (if not already separated in the data warehouse) into their respective dimension tables. This process is called normalization of the data, and the goal is to avoid repetitiveness.

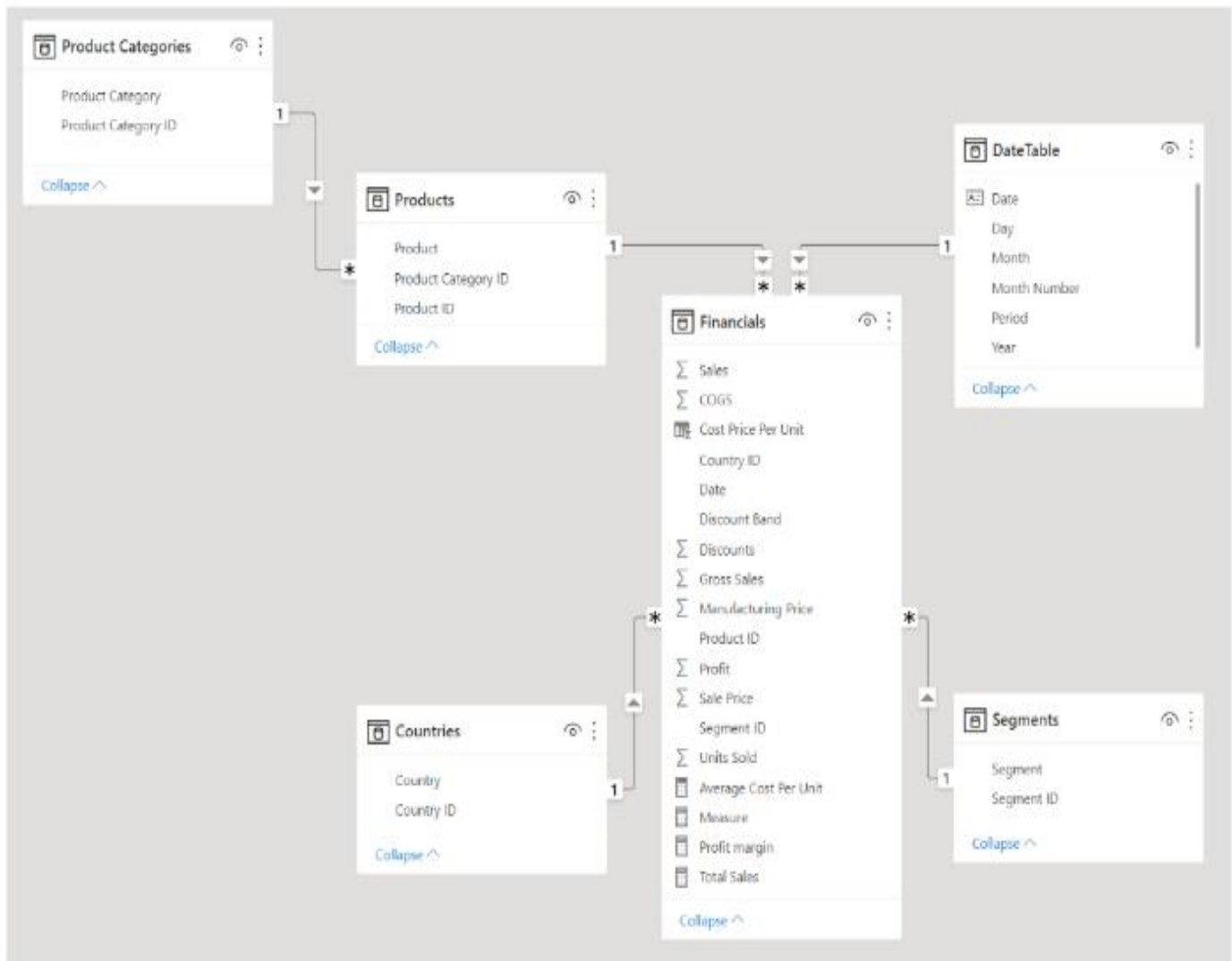
[Dimension or lookup tables](#) describe the features of the dataset, where features are logically grouped together in separate tables to avoid bloating a single table or making it difficult to create relationships. The dimension tables should contain unique values for the features they are describing, and their columns will be responsible for filtering or grouping the data in the Power BI report. For example, Product table describes the attributes such as product key, product category, product name, etc.

Snowflake Schema

An extension of the star schema is the snowflake schema. In this approach, the dimension tables are further divided if a feature has any further categories or subcategories. In our case, we have a Products table that contains the Product ID, Product Name, and Product Category. In a [snowflake schema](#), this Products table will only contain the Product ID and the Product Name, while the Product Category would be further divided into its own table with a Product Category ID and Product Category

Sometimes when normalizing data into tables there can be multiple levels of lookup tables. For example, there is a single data table, Sales and there are three lookup or dimension tables, all chained together in a row, one table is

a lookup table of other. This data structure is common in traditional transactional databases.



This is not the best way to structure data in Power BI due to the following reasons:

- The extra relationships will potentially have negative performance impacts on the database.
- Users will be building reports using your database structure or design and they will see all the tables in the data model. This structure is confusing and difficult to understand.
- Power BI is built to be very efficient in the way it stores repetitive data in columns, particularly in the smaller lookup tables.

Cardinality

It refers to the number of unique values in one table related to the number of unique values in another. In Power BI we work with data models that contain data from multiple tables that must be connected through relationships. Well prepared tables that are connected by relationships are required between these tables to function together to allow visualizations to be sliced and diced by data from different tables.

Relationships between tables are necessary to accurately calculate the results and visualize the correct information in the report.

One-One

A row of data from Table X is related to one and only one row of Table Y. This type of relationship is called a [One-to-one relationship](#). Like when we stored the personal details of a customer in a table called Customer and we stored the customers' address in a table called Address and a customer has only one address.

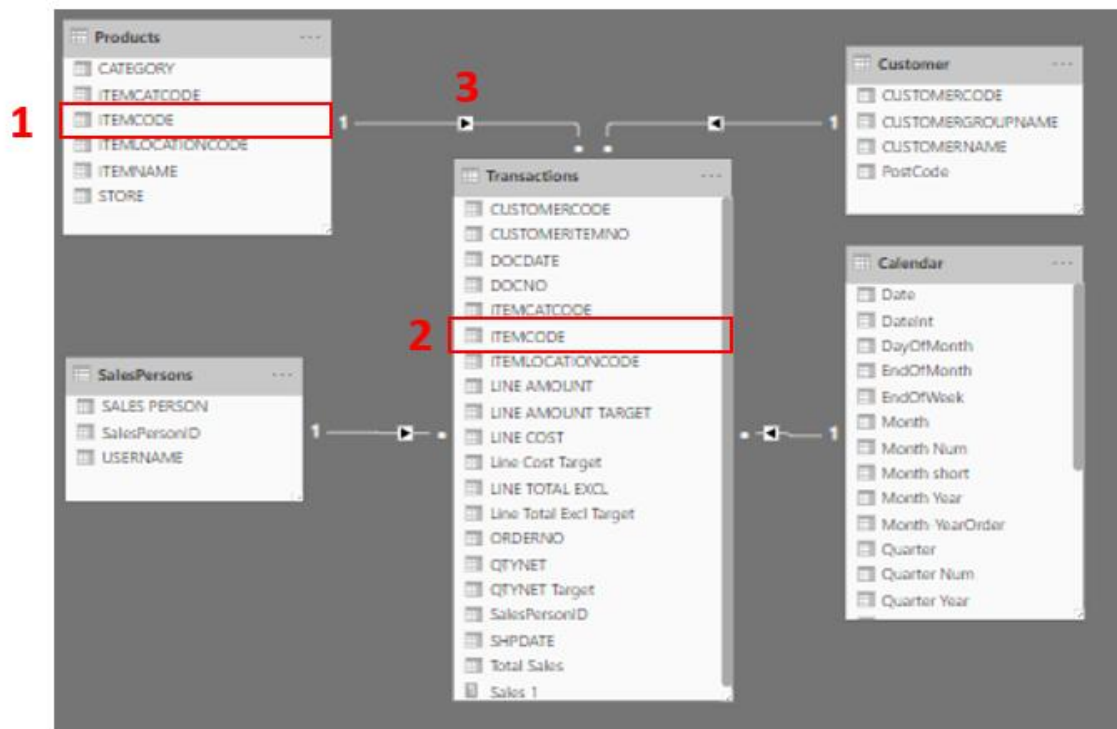
- **One-Many**

A row of data from Table X is related to many rows from Table Y. This type of relationship is called a [One-to-many relationship](#). In our previous example, a customer can have many sales orders, so if we stored the sales

orders in a Sales table, then each customer from the Customer table relates to many rows of sales data from the Sales table.

- **Many-Many**

A **many-to-many relationship** occurs when multiple records in one table are associated with multiple records in another table. This type of relationship is called a **Many-to-many relationship**. For example, a student can enroll in multiple courses, and each course can have multiple students. Another example is, an employee can work on multiple projects, and each project can have multiple employees.



the relationship can filter data across to the other table. This means filtering data in one table affects the data shown in the other table, but filtering in the other table does not affect the data shown in the first table.

Bi-Directional

When a **bi-directional cross filter** is applied, data can be filtered in either direction across the relationship between the tables. This means filtering data in one table affects the data shown in the other table and vice versa. This type is less common and can negatively impact performance and create ambiguous filter paths.

In the data model example, the Products dimension and Orders fact tables have a single direction cross filter from Products to Orders, meaning you can filter the Orders table based on the Products table. Still, you can't filter the Products table based on the Orders table.

All the relationships in this data model are defined by the single cross-filter direction except for the relationship between the Orders fact table and the Orders Details fact table, which has a bi-directional cross-filter direction. This bi-directional cross-filter direction means you can filter the Orders table by the Orders Details table, and you can filter the Orders Details table by the Orders table.

Data Loading – Best Practices

Following are best practices to shape and build your own data model:

- Keep your data tables long, wherever possible and get rid of extra columns of data by unpivoting your data, particularly if a data table is very wide (i.e., has a lot of columns).
- Create lookup tables instead of repeating attribute columns from your data tables. If a lookup table has only two columns, for example product key and product name then it is better to drop the product key column and just load the product name column directly into the data table.
- Shape and structure the table, which is coming from your company's transactional database system, for example, oracle, SQL server etc.

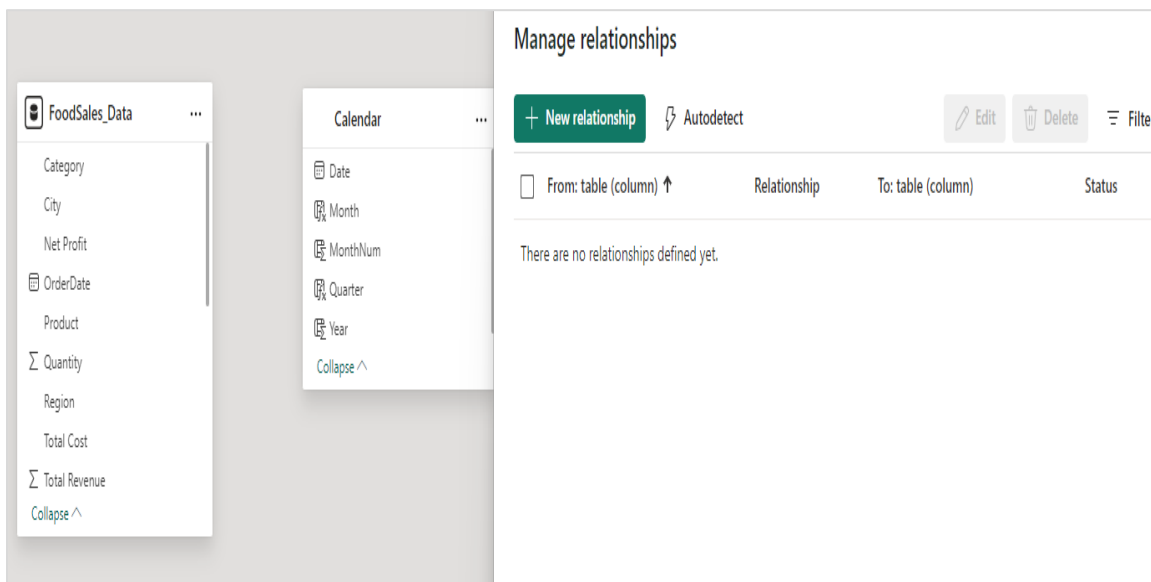
How to Create Relationships in Power BI

There are two ways you can create a relationship in Power BI:

1. Select a field from one table and drag it onto the field in the second table with which you want to make the relationship form.
2. Select Manage Relationships from the ribbon and select “New” to add a relationship using the same window that we will be discussing next (except that it will start as blank).

By default, Power BI will try to infer a relationship between tables; it doesn't always get this right, so you may wish to turn this feature off in the settings or delete any relationships that are created automatically. To edit the relationship, right-click the connecting line between them and select “Properties”.

This window has two interesting options to choose from when defining a relationship: cardinality and cross filter direction. The choices for each of these options can have a big impact on the resulting report, so choose carefully.



FoodSales_Data ...

Calendar

Category

City

Net Profit

OrderDate

Product

Σ Quantity

Region

Total Cost

Σ Total Revenue

Collapse ^

Date

Month

MonthNum

Quarter

Year

Collapse ^

Edit relationship

Select tables and columns that are related.

From table

Calendar

Date	Month	MonthNum	Quarter	Year
1/1/2020	Jan-20	1	Qtr 1	2020
1/2/2020	Jan-20	1	Qtr 1	2020
1/3/2020	Jan-20	1	Qtr 1	2020

To table

FoodSales_Data

Category	City	Net Profit	OrderDate	Product	Quantity	Region
Bars	Boston	79.38	1/16/2020	Carrot	54	East
Bars	Los Angeles	74.97	1/22/2020	Carrot	51	West
Bars	New York	147	1/25/2020	Carrot	100	East

Cardinality

One to one (1:1)

Cross-filter direction

Both

☒ Make this relationship active
☐ Assume referential integrity

Save

Cancel

Data modeling is the process of organizing and preparing data for storing it in a database. This involves a bit of data wrangling or cleaning activities such as filtering out rows we don't need, removing unnecessary columns of data, cleaning up data that is inconsistent or redundant, or completely reorganizing the data to put into a new format.

In our superstore dataset there is a list of all countries, regions, city, category, product, quantity, cost etc.

We will be combining multiple datasets together later and completely reorganize them to give us the best possible data model for building our reports.

The **applied steps** contain the recipe for our data model. After importing data from our CSV file, promoted the first row into our column headers, and changed a few data types, but all of this happened automatically when Power Query first connected to the data source.

Now we need to do some additional data wrangling or cleanup.

Let's start by removing some extra columns we don't need. The quickest way to do this is to use the menu that is available in the top left corner of our data table. When you click on it, you will see a menu of transformations pop up for our dataset. And one of these options is choose columns in Home Tab.

When we click this, it will give us a pop-up window showing all the columns in our dataset and we can just choose the ones we want.

In this way you can select columns from the list which you need, for example you may select product, region etc.

In our exercise, we will be examining how product and category relate to each other.

So, these are the only two columns we need. When I click OK, the other columns we did not select are removed from the dataset and a new step appears in the applied steps named [removed other columns](#).

The reason to remove columns is primarily to minimize the size of our data model. The more columns you have, the more data Power BI is going to import into your data model. This can affect Power BI's performance when your data model gets big enough.

But keep in mind, we can always go back and add more columns later. Suppose we forgot to add region, which we need for our report.

There are two ways we can go back and add those to our data model.

In the first method, we go to our recipe for applied steps and look at the step where we removed the other columns. Notice that there is a [gear icon](#) next to the right of the step.

If we click on the gear, the choose columns window will reopen and we can just select region from the list. When we hit OK, you will see that region has been added into our table.

The second way to edit our recipe is to use the formula bar.

If you look directly above the data table, you should see it. If you don't see the formula bar, you can enable it by going to the [view tab](#) and selecting [formula bar](#).

The formula bar shows you the code that results from everything you do in Power Query. And we can switch between the different steps in our recipe and see this code for each one.

For instance, if we click on our source step, you will see some code that is connecting Power Query to our data source. If we click on the most recent step of our recipe, removed other columns, you can see how it uses this [table select columns statement](#) to tell Power Query grab these columns.

This language is called M, the letter **M** is for data **mash-up**. It is a bit advance language and is not necessary to learn at a beginner level but as you progress into more advanced Power BI techniques, learning M can be useful, especially as you do things like conditional formulas and advanced filtering.

In this case, all you would need to do to add a column back into your data model would be to modify this simple formula using the formula bar.

In this exercise, we want to find out which categories of products are most sold, but first we need to do a little bit of calculations.

So, we are going to add a new column that calculates the sales price of each product by adding a custom column and apply the formula, unit price multiply by quantity.

Another way is to select the two columns and go to that add column tab and select the dropdown menu underneath standard and then we click multiply.

After doing this, you will see we have a new column in our dataset named Sales Price and a new final step added to our recipe.

The order in which we select the columns is very important because it tells Power Query which field is the numerator, and which is the denominator.

Our dataset is very small, so this is not a problem in this exercise but adding a column like this in Power Query is not the preferred way of handling these types of calculations.

The preferred method for performing calculations is to do it in Power BI desktop using **Measures**.

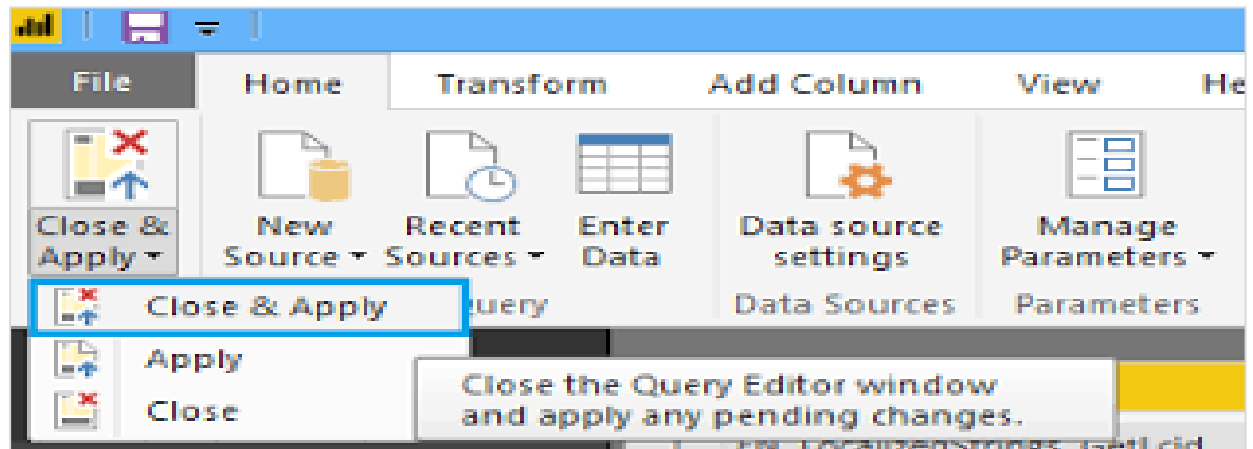
Measures are dynamically calculated in our report and do not require us to add more data to our data model. There is another method in Power BI Desktop called **calculated columns** which we will cover later.

There are three different ways to rename a column name by either double-clicking on the column header or by right-clicking on it and selecting the rename option.

When you do two or more of the same operation back-to-back, Power Query likes to combine them into one step. We renamed three separate columns, but Power Query only created one renamed column step. And the formula in the formula bar shows all three columns being renamed in one operation. This keeps the steps streamlined and efficient for you.

Now we have a quite decent data model prepared. So, we are ready to load it and move to Report Editor & start building our report.

POWER QUERY EDITOR



	A	B	C	D	E	F	G	H	I
1	OrderDate	Region	City	Category	Product	Quantity	Unit Cost	UnitPrice	Sales
2	1/1/2020	East	Boston	Bars	Carrot	33	0.5	1.77	58.41
3	1/4/2020	East	Boston	Crackers	Whole Wheat	87	0.6	3.49	303.63
4	1/7/2020	West	Los Angeles	Cookies	Chocolate Chip	58	0.4	1.87	108.46
5	1/10/2020	East	New York	Cookies	Chocolate Chip	82	0.4	1.87	153.34
6	1/13/2020	East	Boston	Cookies	Arrowroot	38	0.3	2.18	82.84
7	1/16/2020	East	Boston	Bars	Carrot	54	0.5	1.77	95.58
8	1/19/2020	East	Boston	Crackers	Whole Wheat	149	0.6	3.49	520.01
9	1/22/2020	West	Los Angeles	Bars	Carrot	51	0.5	1.77	90.27
10	1/25/2020	East	New York	Bars	Carrot	100	0.5	1.77	177
11	1/28/2020	East	New York	Snacks	Potato Chips	28	0.35	1.35	37.8
12	1/31/2020	East	Boston	Cookies	Arrowroot	36	0.3	2.18	78.48
13	2/3/2020	East	Boston	Cookies	Chocolate Chip	31	0.4	1.87	57.97
14	2/6/2020	East	Boston	Crackers	Whole Wheat	28	0.6	3.49	97.72
15	2/9/2020	West	Los Angeles	Bars	Carrot	44	0.5	1.77	77.88
16	2/12/2020	East	New York	Bars	Carrot	23	0.5	1.77	40.71
17	2/15/2020	East	New York	Snacks	Potato Chips	27	0.35	1.35	36.45
18	2/18/2020	East	Boston	Cookies	Arrowroot	43	0.3	2.18	93.74
19	2/21/2020	East	Boston	Cookies	Oatmeal Raisin	123	0.25	2.84	349.32
20	2/24/2020	West	Los Angeles	Bars	Bran	42	0.5	1.87	78.54
21	2/27/2020	West	Los Angeles	Cookies	Oatmeal Raisin	33	0.25	2.84	93.72
22	3/2/2020	East	New York	Cookies	Chocolate Chip	85	0.4	1.87	158.95
23	3/5/2020	West	San Diego	Cookies	Oatmeal Raisin	30	0.25	2.84	85.2
24	3/8/2020	East	Boston	Bars	Carrot	61	0.5	1.77	107.97
25	3/11/2020	East	Boston	Crackers	Whole Wheat	40	0.6	3.49	139.6
26	3/14/2020	West	Los Angeles	Cookies	Chocolate Chip	86	0.4	1.87	160.82
27	3/17/2020	East	New York	Bars	Carrot	38	0.5	1.77	67.26
28	3/20/2020	East	New York	Snacks	Potato Chips	68	0.35	1.68	114.24
29	3/23/2020	West	San Diego	Cookies	Chocolate Chip	39	0.4	1.87	72.93
30	3/26/2020	East	Boston	Bars	Bran	103	0.5	1.87	192.61
31	3/29/2020	East	Boston	Cookies	Oatmeal Raisin	193	0.25	2.84	548.12

Building Our First Report

Now we can build a report by dragging Sales from the field pane to the Report Canvas in Power BI Desktop. Then drag Product too so it will show the Sales by Product.

We can change the chart type as **stacked bar chart**.

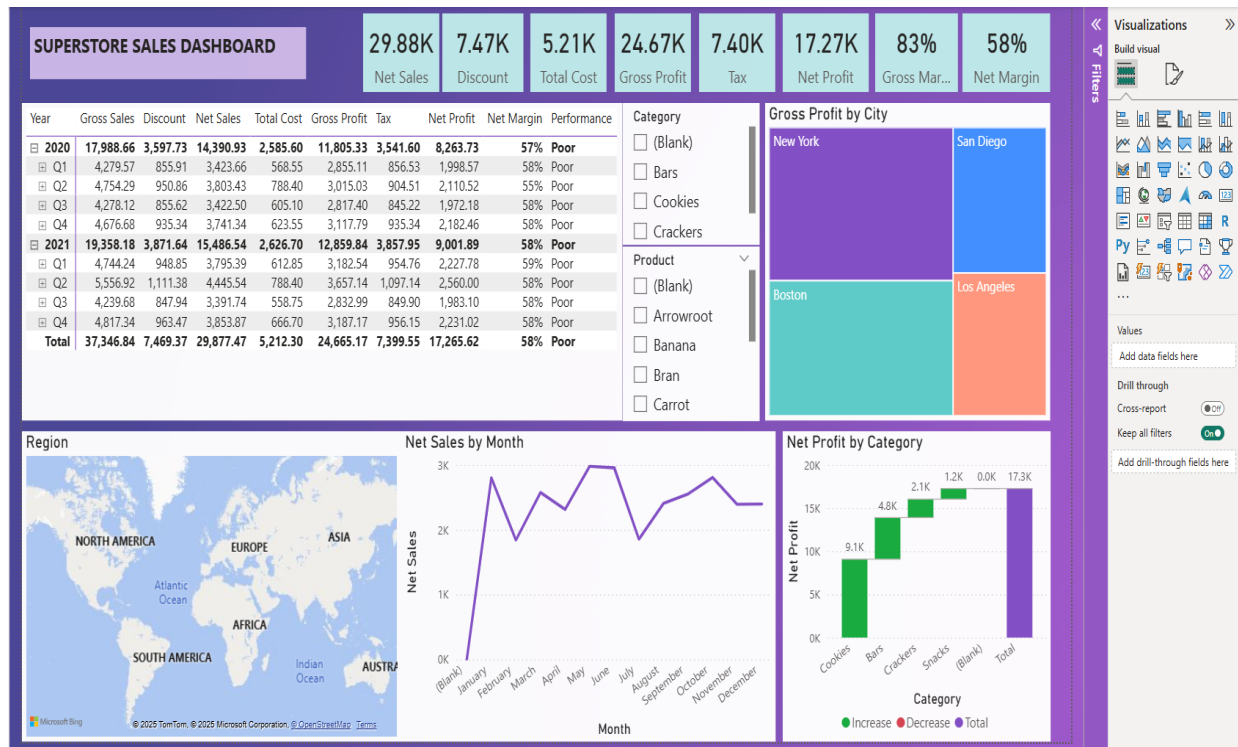
We must unselect the existing chart and can add another visual such as tree-map. Drag the Sales in values pane and Country in the Group.

Then drag Region anywhere on the report canvas to create a new visual, Power BI by default creates a map.

Drag and drop the card visual from visualization pane to the Report canvas to show the KPI's. For example, we can show Net Sales, total cost, net profit, tax, and net margin% etc.

Furthermore, we can also insert a slicer then drag and drop fiscal year into it to show the date at various granularity, such as year, quarter, and month. So that it can filter data based on the selected period.

SALES DASHBOARD



This sales dashboard is showing Sales by Product and Sales by Month. The total profit has also shown for each country using the **Treemap**.

The map is used to show the region or states and countries for sales by each category.

The formatting can be done by clicking at the format tab at the right-hand side in Visualization pane.

We can format the canvas background, font color, size and chart title, legends, color options etc. Additionally, we can apply conditional formatting by right clicking on any measure or column in the field pane. Various options are available such as data bars, gradients etc.

6. MEASURES USING DAX

Measures are dynamically calculated fields in your report that could be used for counts, sums, averages, percentiles, ranking, running totals, year to date and year over year figures, and numerous other applications.

Measures in Power BI are used to perform calculations on data, such as aggregations, ratios, and percentages. They can be created using a variety of DAX functions, including SUM, AVERAGE and COUNT.

It is important to note that measures are dynamic and will update automatically as data changes. This means that reports and visualizations using measures will always reflect the most up-to-date information.

Why Use Measures in Power BI?

Measures can be helpful for various reasons, which include:

- **Summarizing important data:** Measures are essential for summarizing large and complex data sets to obtain insights easily.
- **Performing calculations:** Measures provide a way to perform calculations on data that is not possible through regular columns.
- **Creating complex metrics:** Power BI allows users to create complex metrics that can define business performance and help users monitor KPIs.

Another reason to use measures in Power BI is that they can help with data accuracy. By using measures, you can ensure that calculations are consistent and accurate across all visualizations and reports. This is especially important when dealing with large data sets that may have inconsistencies or errors. Measures can also help with data governance by providing a standardized way to calculate and report on key metrics.

Moreover, measures can be used with other measures in DAX formulas to perform arithmetical operations such as addition, subtraction, multiplication, and division.

Understanding Measures in Power BI

A measure is defined as a calculation based on column values in a table or across multiple tables. It can perform arithmetic calculations or aggregate functions like summing, counting, averaging, or calculating the minimum/maximum values. These measures are defined using the [DAX \(Data Analysis Expressions\)](#) language, which is like the Excel formula language, making it easy for Excel users to understand. Measures can be created for use in Reports or added to the values field in visualizations.

Measures can also be used to create calculated columns, which are columns that are created based on a formula or expression. Unlike measures, calculated columns are computed during data loading and are stored in the data model. This means that they can be used in any visualization or report without the need to recalculate the formula each time.

However, [calculated columns](#) can increase the size of the data model and slow down performance if not used carefully. It is important to understand the differences between measures and calculated columns and choose the appropriate one for your needs.

Creating a Measure in Power BI

Creating a measure in Power BI involves the following steps:

1. Select the table containing the needed column(s).
2. Right click the table and select "New Measure".
3. Enter a name for the measure.
4. Define the calculation formula using the DAX syntax.
5. Drag & drop the measure in the visuals in your report.

Creating Calculated Columns and Measures with DAX

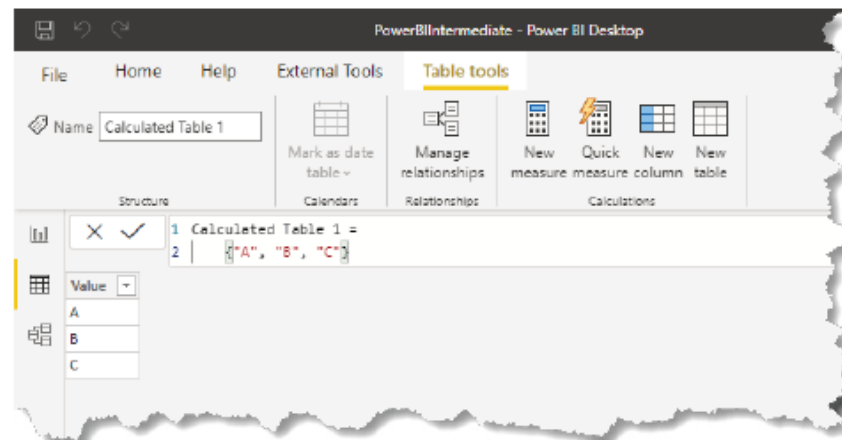
A data model consists of Tables and their Relationships. There are also other elements included in the data modeling:

1. Calculated Tables
2. Calculated Columns
3. Measures

Calculated Tables

Sometimes you need to add new tables based on data you have already loaded into the model. These tables can be created using DAX. You can also use table constructor in DAX to create a calculated table. Table constructor isn't a function; it's a set of characters that allow you to create a table in DAX. For example, the following DAX expression creates a table with one column.

Calculated Table 1 =
{ "A", "B", "C" }



Calculated Columns

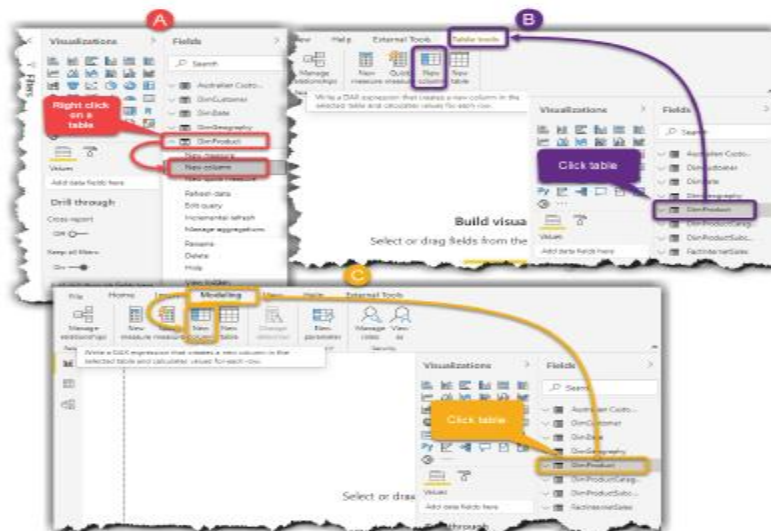
Calculated columns are the new columns created in the data model using DAX. There are many scenarios when you want to create a calculated column; the general rule of thumb is that you only create a new calculated column if:

- There is a complex scenario, and you want to create calculated columns to use them in other calculations like measures.
- You need to create a new calculated column to be used in a slicer or as a filter on a report element.

Other than that, you should avoid creating new calculated columns as there is a risk of performance degradation.

You can create a calculated column either from the Report view or the Data view (you don't need to complete this as a step):

- A. Right click on a table from the Fields pane and select New Column.
- B. Select a table from the Fields pane: select New Column from the Table Tools tab.
- C. Select a table from the Fields pane: select New Column from the Modelling tab.



Why Calculated Columns?

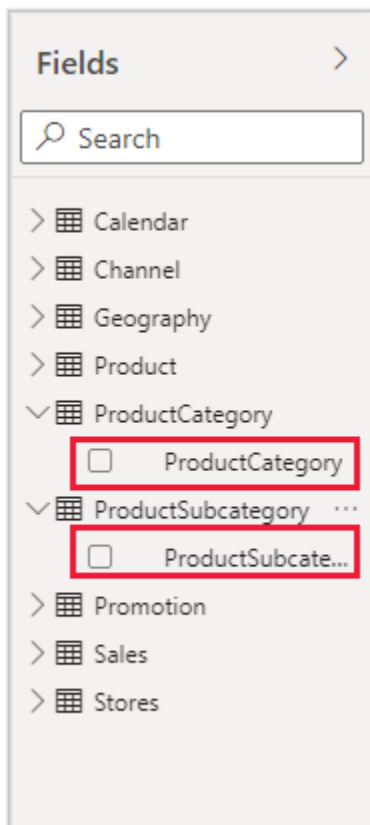
Sometimes the data you are analyzing does not contain a particular field that you need to get your desired results. Calculated columns are useful for this situation. Calculated columns use [Data Analysis Expressions \(DAX\)](#) formulas to define a column's values. This tool is useful for anything from putting together text values from a couple of different columns to calculating a numeric value from other values. For example, let's say your data has City and State fields, but you want a single Location field that has both, like "New York, NY".

Calculated columns are similar to measures in that both are based on DAX formulas, but they differ in how they are used. You often use measures in a visualization's Values area, to calculate results based on other fields. You use calculated columns as new Fields in the rows, axes, legends, and group areas of visualizations.

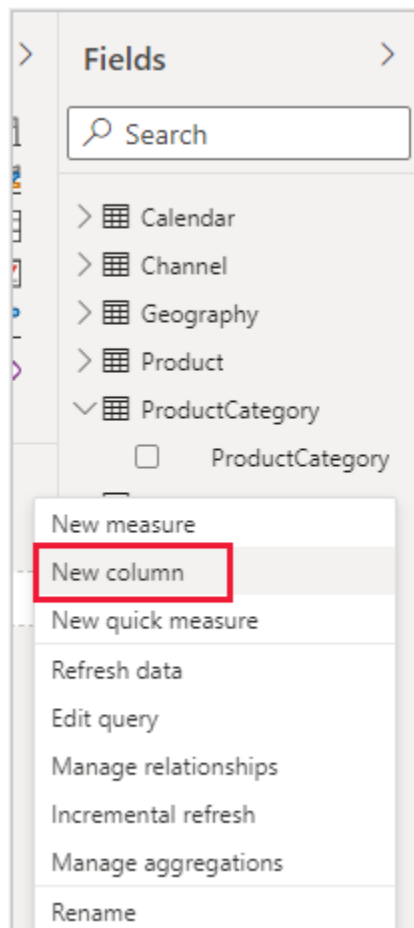
Create a calculated column with values from related tables.

For example, in our Sales Report, we want to display product categories and subcategories as single values, like "Cell phones – Accessories", "Cell phones – Smartphones", and so on. There's no field in the Fields list that gives you that data, but there is a "ProductCategory" field and a "ProductSubcategory" field, each in its own table.

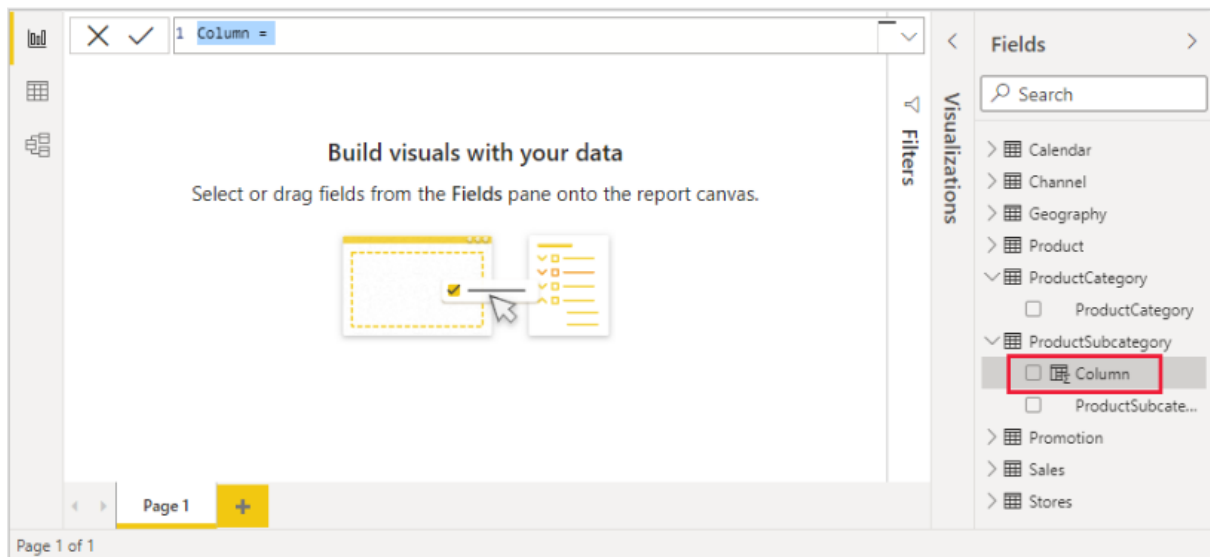
You can create a calculated column that combines values from these two columns. DAX formulas can use the full power of the model you already have, including relationships between different tables that already exist.



To create your new column in the ProductSubcategory table, right-click or select the ellipsis ... next to ProductSubcategory in the Fields pane and choose New column from the menu.

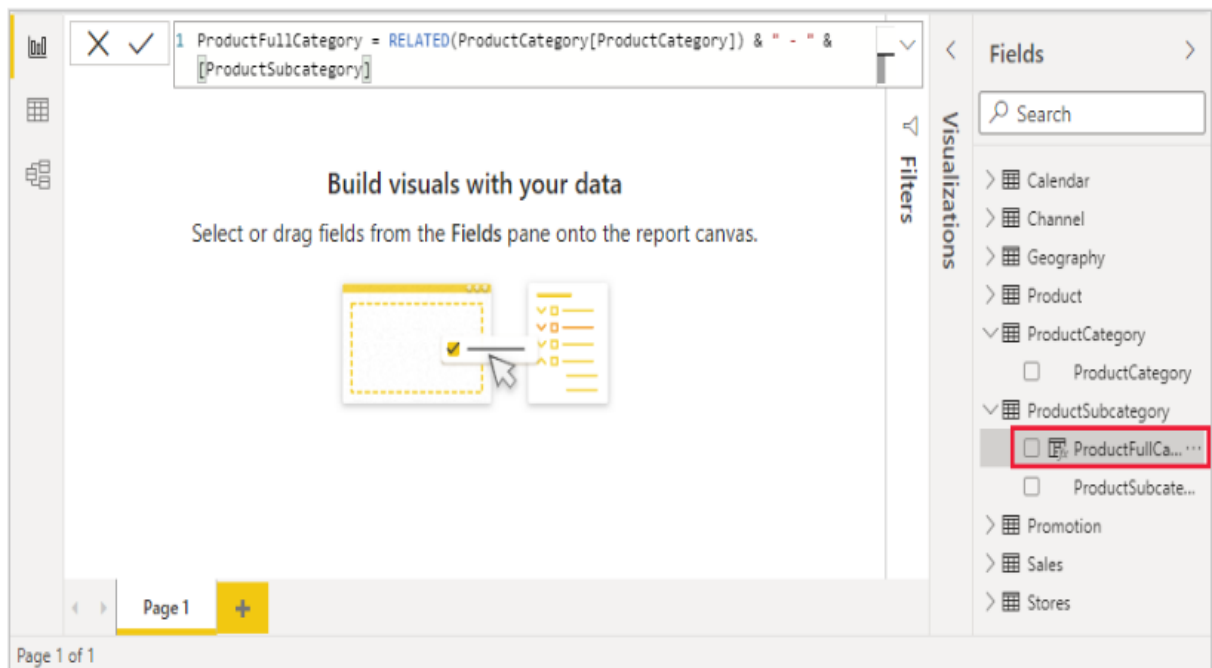


When you choose **New column**, the **Formula bar** appears along the top of the Report canvas, ready for you to name your column and enter a DAX



formula.

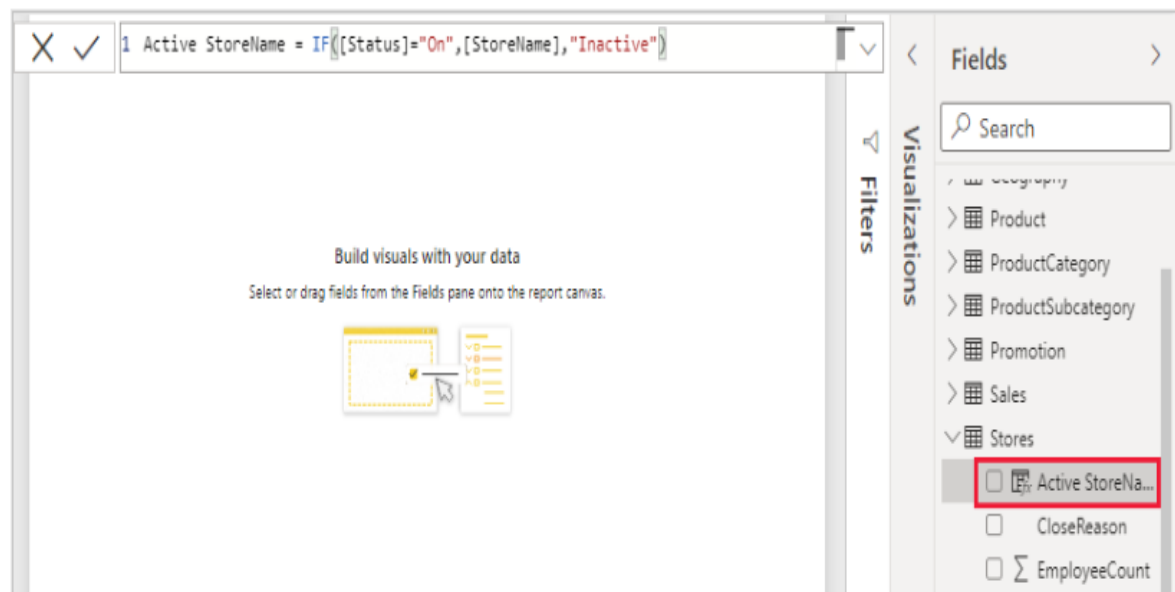
An opening parenthesis appears, along with another suggestion list of the related columns you can pass to the RELATED function, with descriptions and details of expected parameters.



USE YOUR NEW COLUMN IN A REPORT

Now you can use your new ProductFullCategory column to look at SalesAmount by ProductFullCategory.

Create a calculated column that uses an IF function.



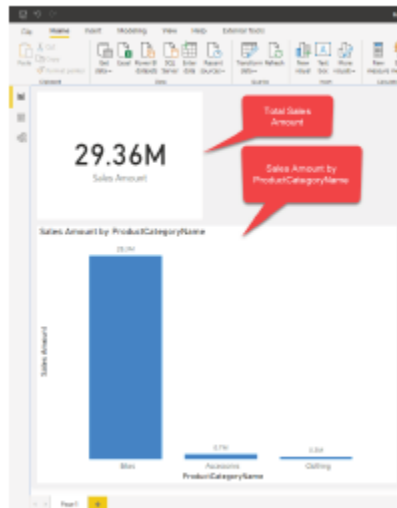
query.

Measures

Measures are normally analytical calculations: summations, calculating averages, minimum, maximum, counts and so on. You can use the measures for visuals in Power BI.

The result of the measures change depending on how we interact with them in different visuals. For instance, you create a measure to calculate Sales Amount. If you put the Sales Amount on a Card visual, it shows total sales amount over the whole data. If you use the same measure in a

Column Chart with Product Category on the Axis, the measure always calculates the correct results for each category.



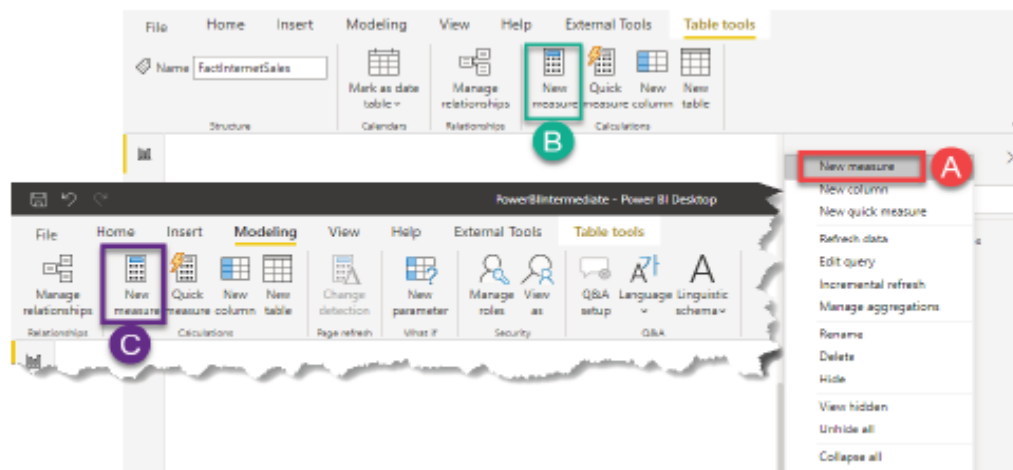
 You can create measures in various ways, just like the way you create calculated columns - either from the Report view or from the Data view.

Right click on a table from the Fields pane and select New measure.

B. Select a table from the Fields pane and select New measure from the Table

Tools tab.

C. Select a table from the Fields pane and select New measure from the Modeling tab.



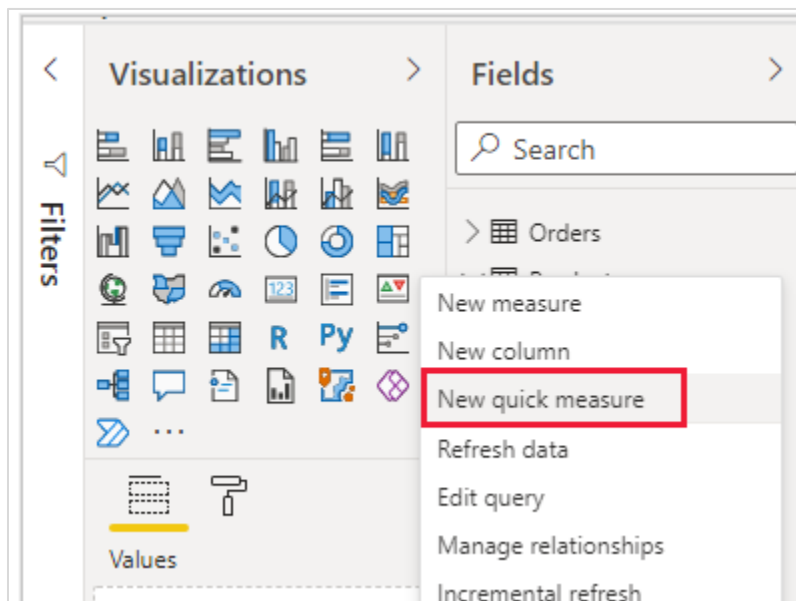
Quick Measures

We can use quick measures to perform powerful calculations quickly and easily. A quick measure creates a set of [Data Analysis Expressions \(DAX\)](#) commands at the back end, then presents the results to use in your report. You don't need to write the DAX; it's done for you based on input you provide in a dialog box. There are many available categories of calculations and ways to modify each calculation to fit your needs.

Steps to Apply Quick Measures in Power BI

1. Open Power BI Desktop and load your dataset.
2. In the Fields pane, select the table where you want the measure to be created.
3. Right-click the table name and choose New Quick Measure.
4. In the Quick Measures dialog box:
 - Select the Calculation type (e.g., Total, Average per category, Year-to-date, Running total, Percentage difference, etc.).
 - Choose the Base value (the column or measure you want to calculate on).
 - Select any additional fields required (like category or date).

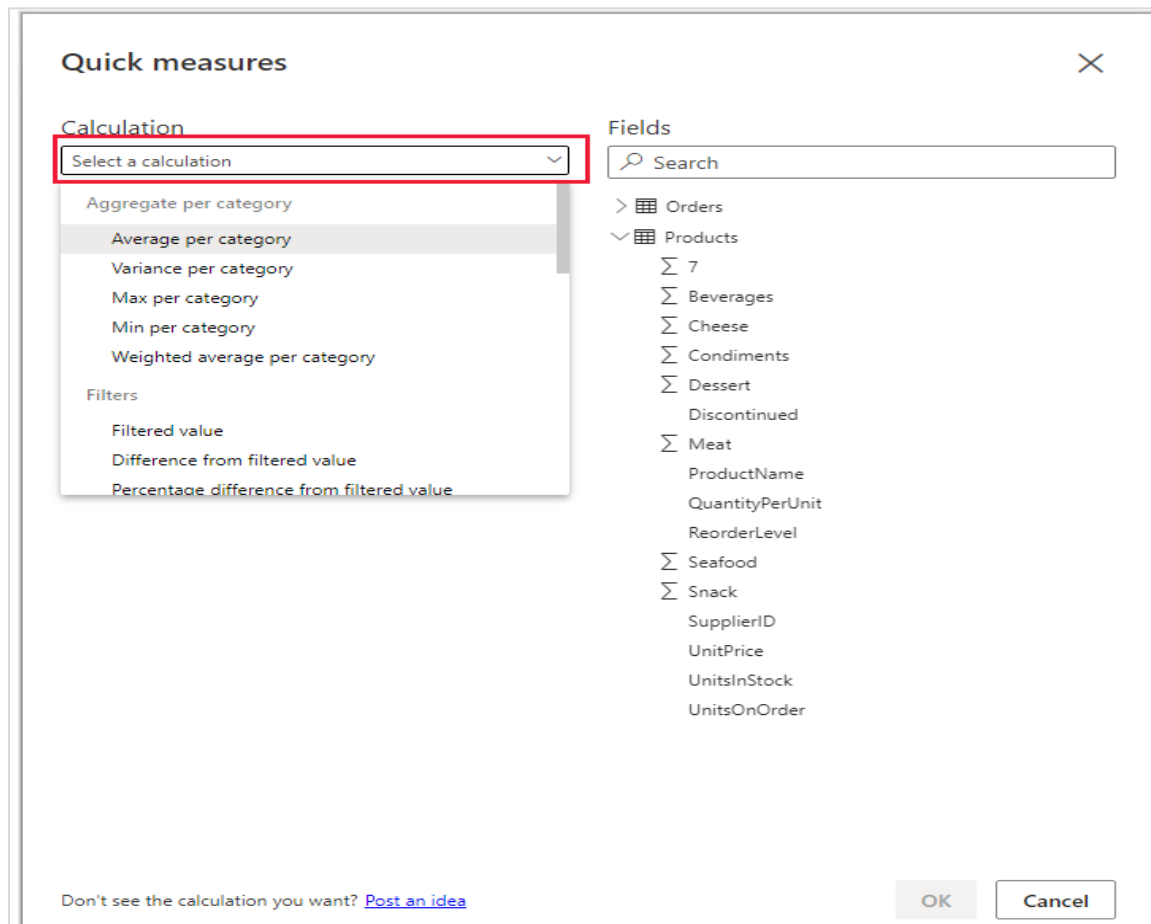
5. Click OK. Power BI will automatically generate the DAX formula for you and add the measure to your table.
6. Use this new measure in your visuals like any other field.



 You can also right-click or select the drop-down arrow next to any value in the Values wall for an existing visual and choose New quick measure from the menu.

When you select New quick measure, the Quick measures window appears, letting you choose the calculation you want and the fields to run the calculation against.

Choose the Select a calculation field to see a long list of available quick measures.



After you select the calculations and fields you want for your quick measure, choose **OK**. The new quick measure appears in the **Fields** pane, and the underlying DAX formula appears in the formula bar.

Following are the categories of quick measure calculation types:

- **Aggregate per category.**
 - Average per category.
 - Variance per category
 - Max per category
 - Min per category
 - Weighted average per category

Example: Average per Category.

Use Case: Calculate average sales per product category.

How to apply: Select your measure (e.g., Sales Amount) and category column (e.g., Product Category).

- **Filters**

- Filtered value.
- Difference from filtered value
- Percentage difference from filtered value
- Sales from new customers

Example: Filtered Value

Use Case: Show sales only for a specific region.

How to apply: Choose measure and apply filter condition.

- **Time intelligence**

- Year-to-date total
- Quarter-to-date total.
- Month-to-date total
- Year-over-year growth
- Quarter-over-quarter change.
- Month-over-month change
- Rolling average

Examples: Year-to-date total Or, Month-over-month change

Use Case: Compare current month sales vs previous month.

How to apply: Select date column and measure.

- **Totals**

- Running total
- Total for category (filters applied)
- Total for category (filters not applied)

Example: Running Total

Use Case: Cumulative sales over time.

How to apply: Select measure and date column.

- **Mathematical operations**

- Addition
- Subtraction
- Multiplication
- Division
- Percentage difference
- Correlation coefficient

Example: Difference between two measures

Use Case: Profit = Revenue – Cost.

How to apply: Pick two measures.

- **Text**

- Star rating
- Concatenated list of values

Example: Concatenate values

Use Case: Combine first name and last name.

How to apply: Select columns to concatenate.

Below are some of the ready-to-use DAX formulas for the most useful financial Quick Measures.

1. Year-to-Date (YTD) Total

- ✓ Go to Quick Measure → Time Intelligence → Year-to-date total.
- ✓ Select:
- ✓ Base Value: Sales Amount
- ✓ Date: Order Date

```
YTD_Total =  
CALCULATE (  
    SUM ('Sales'[Sales Amount]),  
    DATESYTD('Calendar'[Date]))
```

2. Month-to-Date (MTD) Total

```
MTD_Total =  
CALCULATE (  
    SUM('FactTable'[Amount]),  
    DATESMTD (' Calendar '[Date]))
```

3. Quarter-to-Date (QTD) Total

```
QTD_Total =  
CALCULATE (  
    SUM('FactTable'[Amount]),  
    DATESQTD (' Calendar '[Date]))
```

4. Variance (Actual vs Budget)

```
Variance =  
SUM('FactTable'[Actual]) - SUM('FactTable'[Budget])
```

5. Percentage Variance

```
%Variance =  
DIVIDE (  
    [Variance],  
    SUM('FactTable'[Budget]),  
    0
```

6. Year-over-Year (YoY) Growth

```
YoY_Growth =  
DIVIDE (  
    [Current Year Sales] - [Previous Year Sales],  
    [Previous Year Sales],  
    0)
```

7. Rolling 3-Month Average

```
Rolling_3M_Avg =  
CALCULATE (  
    AVERAGE('FactTable'[Amount]),  
    DATESINPERIOD (' Calendar '[Date], LASTDATE (' Calendar '[Date]), -3,  
    MONTH)  
)
```

8. Rank by Category

```
Rank_Category =  
RANKX (  
    ALL('Dimension'[Category]),  
    CALCULATE(SUM('FactTable'[Amount])),  
    ,  
    DESC  
)
```

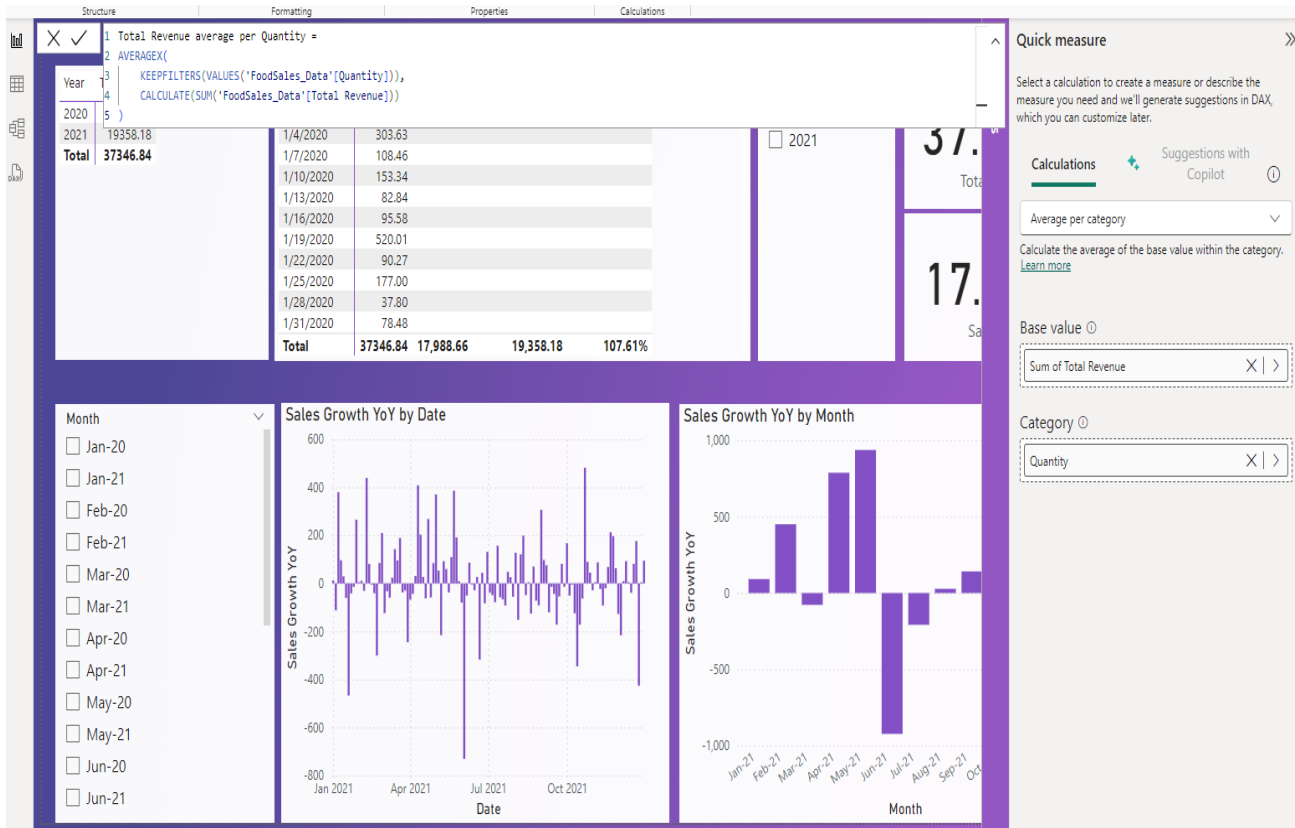
9. Percentage of Grand Total

```
% of Total =  
DIVIDE (  
    SUM('FactTable'[Amount]),  
    CALCULATE(SUM('FactTable'[Amount]), ALL('FactTable')),  
    0  
)
```

10. Running Total

```
Running_Total =  
CALCULATE (  
    SUM('FactTable'[Amount]),  
    FILTER (  
        ALL('Date'),  
        'Date'[Date] <= MAX('Date'[Date])  
    )  
,
```

★ PRACTICAL EXAMPLE – QUICK MEASURE



The formula bar not only shows you the formula behind the measure, but more importantly, lets you see how to create the DAX formulas underlying quick measures.

If you need to do a year-over-year calculation, but you are not sure how to structure the DAX formula, or you have no idea where to start. You can create a quick measure by using the [Year-over-year change calculation](#) and see how it appears in your visual and how the DAX formula works. Then you can either make changes directly to the DAX formula or create a similar measure that meets your needs and expectations.

Using DAX to Create Measures

DAX formulas are created using a combination of functions, operators, and expressions to perform calculations. Some of the common DAX functions used to create measures in Power BI include SUM, AVERAGE, MIN, MAX, COUNTROWS, and DISTINCTCOUNT. To create a measure using DAX, you need to have a good understanding of DAX syntax, functions, and calculations.

Another important aspect of using DAX to create measures in Power BI is understanding how to optimize your formulas for performance. DAX calculations can be resource-intensive, especially when working with large datasets. To improve performance, you can use techniques such as filtering, aggregation, and calculation groups to reduce the amount of data that needs to be processed.

The key difference between DAX and Excel formulas is that Excel formulas work with cells in a workbook while DAX formulas are optimized to work with the tables and columns in your Power BI data model.

DAX gives you many more potential ways to summarize and analyze your data and it is also very fast at calculations and very flexible.

One of the biggest benefits of measures is that they are reusable within other measures.

In our example, we calculated our Total Sales by just dragging those numeric fields with the Sigma symbols next to them from our dataset into our charts.

When we do that, we are creating an implicit measure.

Implicit measures are easy and straightforward, but they are not the recommended way of calculating measures in Power BI.

Instead, you want to create an explicit measure, which means learning a little bit of DAX.

Remember, explicit measures are fast, re-usable, can be calculated across multiple tables, and are far more flexible than implicit measures.

As you work with more complex data models, you will quickly realize the limitations of implicit measures and see why Power BI developers avoid them.

Looking at the far-right side of my home tab, you can see two buttons namely New Measure and Quick Measure.

If you look at your Fields pane, a new field has appeared, called Measure, with a calculator symbol next to it. The calculator symbol shows that this is an explicit measure, that is what we want.

So, we click into the formula bar, and we will write a very simple DAX formula.

When you press Enter, our new measure will rename itself, and it's ready to be used in our report.

Now we are going to create our Total Sales measure.

Click New Measure again, and this time in our formula bar, we are going to write, "Total **Sales = [Sales] quantity * [Sales] unit price**"

Press Enter.

At this point, we have a new measure, Total Sales and we need to create one more for Total Cost.

This time let's use the quick measure feature.

After you click Quick Measure button, a window will appear.

On the right side, you can see all the fields in our data model, including our new explicit measures.

And on the left, is a drop-down menu with a list of calculations. This is not the full list of calculations in DAX, it's just some basics. Let's scroll to the bottom of the list and select Multiplication. When we do that, Power BI gives us two areas or drop zones that we can drop fields.

We are going to drag Quantity and unit cost, so we are multiplying the Quantity and unit cost. Once we hit OK, the window goes away and a new measure is created, named Total Cost.

Power BI is trying to be helpful in giving us a simple English description of what we did, but the real term for what we did is calculating the Total Cost, so we can rename that measure. To rename it, double click on the measure and then type in Total Cost.

Let's look at the calculations get returned by our new measure. To do it, let's do it with a table visualization, which is a very common and extremely useful tool for summarizing your data.

Now, let's add a new table visual to our report page. And I am going to add in City, Product Category, Total Sales, and product Price.

Tables are great because you can sort them interactively by clicking on any of the column headers and tables have more hidden features that we will discuss later, but for now, I just want to look at what our measures have given us.

The numbers are not in a proper format and are a little inconsistent, so let's fix that. I like how Sales has the thousands separated by commas, and I really want Cost to look that way too, so I am going to my Fields pane and click on Total Sales, and I am going to look up my ribbon.

What we want is this new Measure Tools tab which only showed up when we selected a measure from our Fields pane.

On this tab, we have got the ability to change the data type for our columns, how they are formatted, grouped, and sorted and even how they are summarized when we add them to visuals.

That's all we are going to do now. First add commas to separate thousands in these numbers. For this just click on this little comma icon and press to, my Sales numbers now display commas. And you can do this trick with any field in your data set including regular columns.

Just click on Country and you will see a Column Tools tab on your ribbon. This can be useful when you want to change how specific columns are represented in your report.

Now, let's sort my table by product category in ascending order and we notice, "our third row doesn't have a unit cost." Because the value is zero in our dataset, and you can't divide by zero.

When we originally divided these two numbers in Power Query you probably didn't notice but it returned a tiny error message for this row saying "NaN" or "Not a Number," which is Power Query's way of telling us you can't divide a number by zero. If we had used that implicit measure for our table here, it would mess up our results.

This is an example of how explicit measures like we just created, are almost always preferable to implicit measures because there's no NaN here. Instead of space better to put a zero, so our total cost is consistent even for weird situations like this where we are dividing by zero.

Fortunately, if we click on our Total Cost measure and go back to our formula bar, we can tweak our formula just ever so slightly to make this happen. Now use the Quick Measure alternative. A divide function takes two arguments in the Quick Measure window, the numerator and denominator.

However, there is a third optional argument, that the window doesn't display, and we are going to use it now in the formula bar. It's called Alternate Result, and it allows us to substitute a value whenever we are dividing by zero.

That's a great idea.

 One of the reasons I never use the Quick Measures button is because optional arguments are hidden and in a case like this, you want to use the optional argument.

So, from my formula bar, I am just going to left click just after my Total Cost measure,

and I am going to add a comma, space, and the words, "DAX is Amazing," and hit Enter.

Now, when we look at, the blank space has been replaced by the words, "DAX is Amazing."

It's probably better if we just use a zero though, so, I will modify my formula one more time to use a zero instead of "DAX is Amazing."

And now, our table looks more consistent.

Finally, let's do a little cleanup by replacing all the implicit measures we created earlier with our explicit measures. So, we delete it and add our Total Sales measure to the values and on the map, and notice how our charts look the same, but we're no longer using those implicit measures.

Look again at my DAX formula for Total Sales. I mentioned earlier how measures are reusable and that's one of their biggest strengths. In this case, I wrote a single sum formula for Sales and another sum formula for Cost.

By defining measures, we don't need to write the same formulas repeatedly. With explicit measures, I only need to write those formulas one time and then I can use them in other measures later.

Now let's review everything and explore the data. Looking again at our map you will see that most of the circles are about the same diameter. If you look closely, you can see ones that are larger, but they are a little hard to pick out among all their neighbors.

By going to our table and sorting by product category in descending order, we can see why Bars has the highest sales. The first thing we can do is use our Filters pane.

To start with, notice that we have two sections in the Filters pane. Filters on this page and filters on all pages. We also have the filters on this visual whenever we have selected a particular visual.

This should be self-explanatory but any filters appearing in the filters on this page section will filter the data of all visuals on a particular page we are viewing. We call these Page level Filters.

Any filters appearing in the filters on all page's section will filter all visuals across all pages. We call these Report level Filters. We can create new filters by dragging columns from our fields pane into either of these two sections and dropping them on the boxes that say Add data fields here.

Let's go ahead and drag region and country into the Report level filters. This creates new filters for each of those columns. After we have created a filter, it will automatically show us its options, but we can collapse each filter and hide its options by just clicking on the top of the filter.

Let's say we don't want to change region right now, so we will just click to collapse it and focus on the country filter. Below the word country, we see a bit of text that says, is all which means we are currently not filtering anything, we are showing all regions and all countries.

When a filter is expanded, we see a list of all the possible values for that column and next to each country name is account of the number of rows in our data set with that specific value.

Our table is one row per country. So, it makes sense that each of these says one and to the left of each country is a checkbox. So, we can select our countries from the list. We will select US and Algeria and watch as everything in our report changes to focus on those two countries.

Also, notice how the filter becomes shaded in gray when we make that selection and the filter now says, country is Algeria. Then to go back to all countries, we can either click the Select All option at the top of our country list or we can just unclick the box for Algeria and UK.

If we wanted to require our users to only look at one country at a time, the check box at the very bottom of the filter gives us a way to do that. It says Require Single Selection and notice how after we've clicked it, the Select All option disappears and now we can only choose one country at a time.

One other good thing about filters is they are searchable, and you will see the search box directly above our list of countries. So, we can start typing Canada and watch as it appears in our list where we can easily select it.

Finally, at the top of our filter, we see a few icons. The up-arrow icon will collapse the filter and the X icon will delete it.

Let's talk about these other icons, the padlock, and the eyeball.

Suppose you are writing a report, and you want to prevent your audience from changing a particular filter. The padlock icon will lock the filter preventing anyone from modifying it when they run the report. They will still be able to see it, but they won't be able to change it.

Meanwhile, the eyeball icon will hide the filter from your audience entirely, so it won't be visible in the filter's pane when they run the report. There is a dropdown menu on the filter that says Basic Filtering, which just means the list-based filtering we have already been doing.

But if I click on that menu, we see there is another option for Advanced Filtering. When we enable Advanced Filtering, we get filter options that allow us to do things like, filter our countries when their name contains the word, republic.

Notice there is an Apply Filter button in the bottom right corner that we need to click to apply our new filter criteria. We also have and, or options that allow us to do logical operations like finding all countries whose names start with north or south.

We can also do negative comparisons like does not contain or does not start with. Also notice how, when my filter is shaded gray meaning it's enabled, a new icon appears at the top of the filter that looks like a rectangle that's turned 45 degrees. That's supposed to be an eraser and it just allows us to clear the filter with one click restoring it to its original state. So, let's click it so our filter is cleared.

Next, we will select our map visual from the Report Canvas and look back at the Filters pane. Notice that third category that wasn't there before, filters

on this visual and we can see three boxes underneath it, that say region, country, and category.

These are visual-level filters. This section shows the filters that only affect our map visual and any fields we add to our map visual will automatically show up here in the visual-level filters. If we change any of these, they will filter only our map, but they won't filter anything else in the report.

If we open our category filter, it looks very similar to the advanced filtering options from our country filter a moment ago. We can specify certain values and there is an and, or option. This gives us a good way to filter out our outliers.

In the first section, let's put is less than 1000 then keep the and operator and in our second section, we will say is greater than one.

So, we are telling the map to only filter on those countries that have a sale between one and 1000.

Then click apply filter and watch as our map visual updates and we can see a much greater degree of variation, all these bubbles on the map.

TIME INTELLIGENCE FUNCTIONS

Time intelligence provides measures that simplify complex time-based reporting.

For example, time-based calculations for Year-to-Date (YTD), Month-to-Date (MTD), Quarter-to-Date (QTD) and Previous Period comparisons.

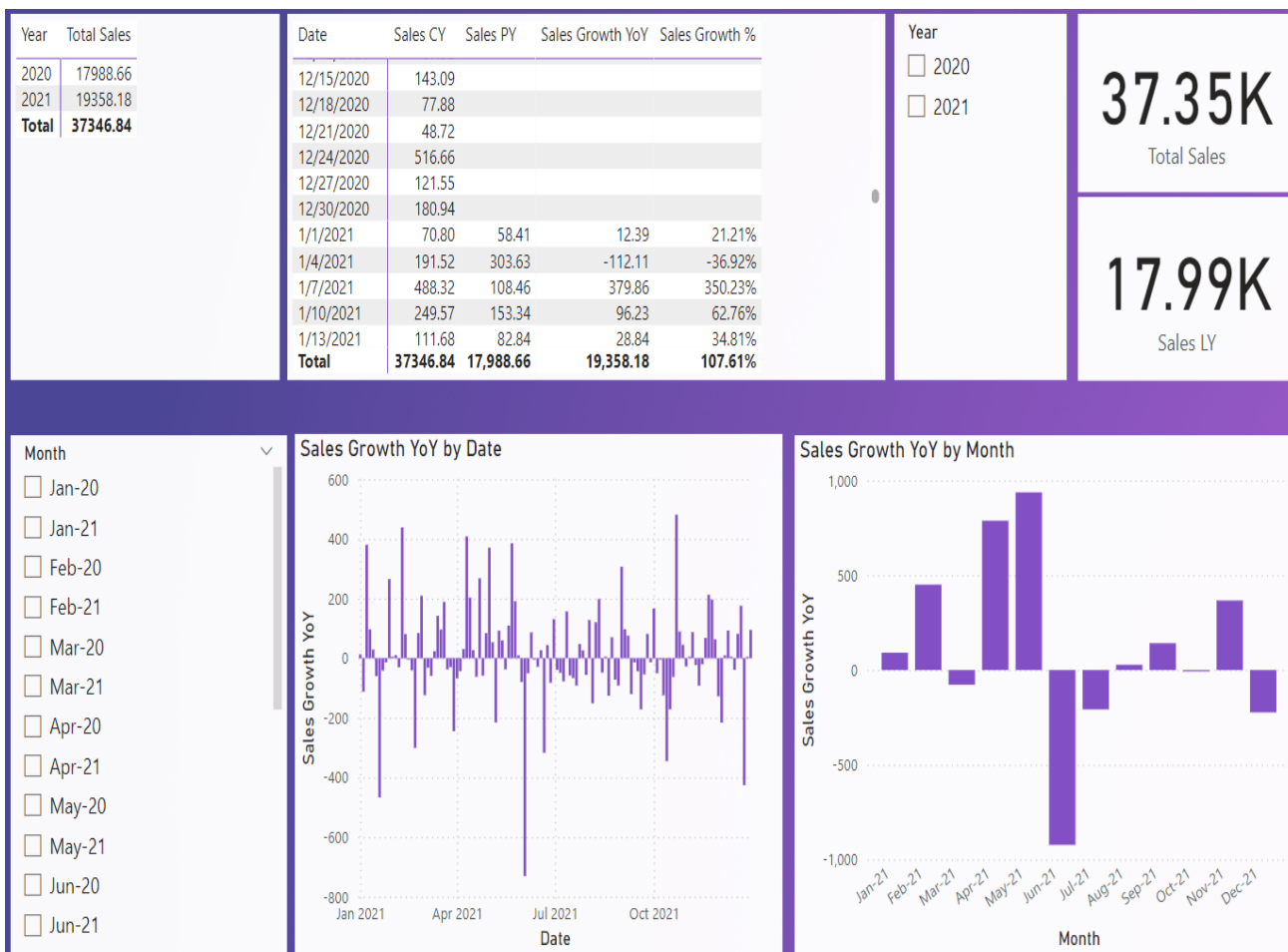
One of the advantages of using DAX to create measures in Power BI is that it allows for more complex calculations than what is possible with basic aggregation functions. For example, you can use DAX to calculate year-over-year growth rates, YTD, QTD & PYTD sales, running totals, and moving averages. These types of calculations can provide valuable insights into trends and patterns in your data.

When you calculate a Year-to-Date (YTD) metric, you are summing the values of that metric daily, starting from 1st Jan of each year up until the provided date. When you calculate a Month-to-Date (MTD) metric, you are summing the values of that metric for each day, starting from the 1st day of the month up until the provided date.

Time intelligence functions such as month-to-date (MTD), quarter-to-date (QTD), and year-to-date (YTD) in DAX operate relative to the current filter context. Their outcome depends on the filter applied, making them both adaptable for various periods and useful for comparisons.

Look at the below dashboard:

It shows Sales Current Year (CY) values, Sales Last Year or Prior Year (PY) and Sales Growth YoY and Sales Growth %.



DAX formulas used in this Dashboard.

We used YTD and PY DAX functions to create the measures used in the above dashboard.

We used the (already created) Sales measure in those functions along with the Date column from the Calendar table. The Date is a Calendar table holding date related data. The Calendar table is the basis for all time intelligence functions available in Power BI, therefore we need to have a Calendar table in our model.



What if our data source doesn't have a Date table?

We have two options:

- A. Use the Auto Date Time feature in Power BI Desktop.
- B. Generate a Date table.

Sales YTD = **Calculate** ('Sales' [SalesAmount], Dates YTD('Calendar'[Date]))

Or,

Sales YTD = **Calculate** (TOTALYTD ([Total Sales], 'Calendar'[Date]))

Sales PY = **Calculate** ('Sales' [SalesAmount],
SAMEPERIODLASTYEAR ('Calendar' [Date]))

Or,

Sales PY = **Calculate** ('Sales' [SalesAmount],

DateAdd (Calendar [Date], -1, Year)) # Moves back by 1 year.

SALES QTD Formula:

QTD Sales =

CALCULATE ([Total Sales],
DATESQTD('Date'[Date]))

Why a Date Table is Required in Power BI?

The Calendar or Date table is the basis for time intelligence functions to work correctly, but there is another reason why we need to have a Date table in a Power BI model.

The Date table provides the detail for a date. If you go to the Data view (make sure you have selected Sales on the right pane, you will see 3 date columns: OrderDate, ShipDate and InvoiceDate columns. While those columns provide date values and we can use them to analyze our data, what if we need to analyze by other date elements?

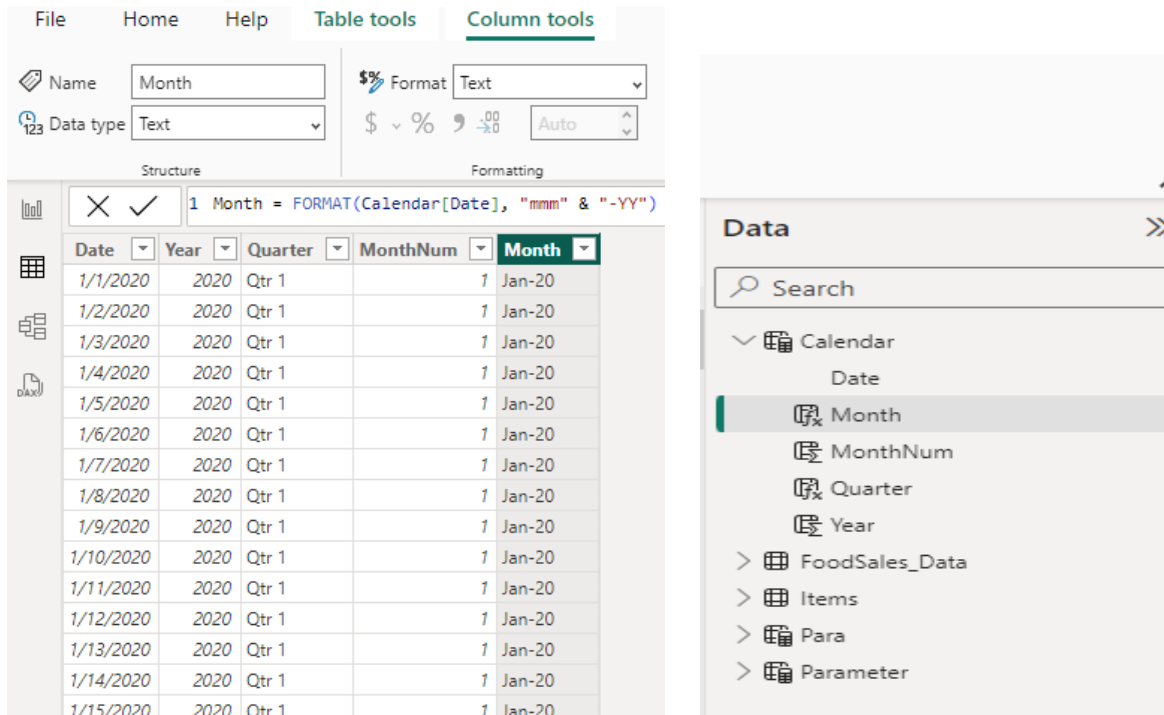
e.g., Sales Amount by month to see our bestselling month.

You can achieve this by creating a calculated column to show Month values. What if you want to analyze the Sales Amount by financial year, do you create a calculated column?

Business queries can grow over time; it's unwise to create new columns whenever the business asks new questions. Specially if you have dataset having thousands of rows.

Creating a new column to only add the Month values increases the size of the Sales table, i.e., it unnecessarily increases the size of the report file and leads to poor performance in larger reports.

To overcome this challenge, create a separate Date table by adding as many columns as you need, then create relationships to other tables to analyze their data over date elements.



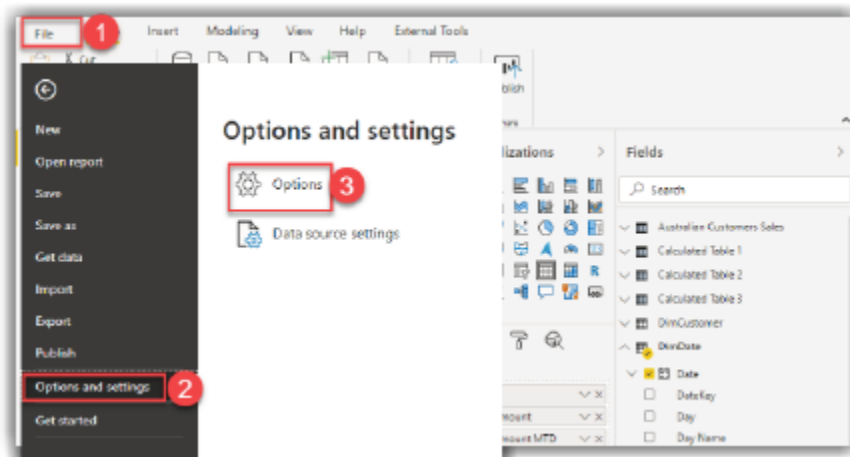
The screenshot shows the Power BI Desktop interface. The top ribbon includes 'File', 'Home', 'Help', 'Table tools', and 'Column tools'. The 'Column tools' tab is active, showing the 'Name' field set to 'Month' and the 'Format' dropdown set to 'Text'. Below this, the 'Data type' is also set to 'Text'. The 'Structure' pane shows a table with columns: Date, Year, Quarter, MonthNum, and Month. The 'Month' column is highlighted. The 'Data' pane on the right shows a search bar and a list of data sources: Calendar, Date, Month (selected), MonthNum, Quarter, Year, FoodSales_Data, Items, Para, and Parameter.

Date	Year	Quarter	MonthNum	Month
1/1/2020	2020	Qtr 1	1	Jan-20
1/2/2020	2020	Qtr 1	1	Jan-20
1/3/2020	2020	Qtr 1	1	Jan-20
1/4/2020	2020	Qtr 1	1	Jan-20
1/5/2020	2020	Qtr 1	1	Jan-20
1/6/2020	2020	Qtr 1	1	Jan-20
1/7/2020	2020	Qtr 1	1	Jan-20
1/8/2020	2020	Qtr 1	1	Jan-20
1/9/2020	2020	Qtr 1	1	Jan-20
1/10/2020	2020	Qtr 1	1	Jan-20
1/11/2020	2020	Qtr 1	1	Jan-20
1/12/2020	2020	Qtr 1	1	Jan-20
1/13/2020	2020	Qtr 1	1	Jan-20
1/14/2020	2020	Qtr 1	1	Jan-20
1/15/2020	2020	Qtr 1	1	Jan-20

Auto Date/Time Feature in Power BI

If you don't have a Date or Calendar table in your data source, don't worry. You can enable the Auto date/time feature in Power BI Desktop. Power BI automatically takes care of the Date table. To enable this feature, follow the steps:

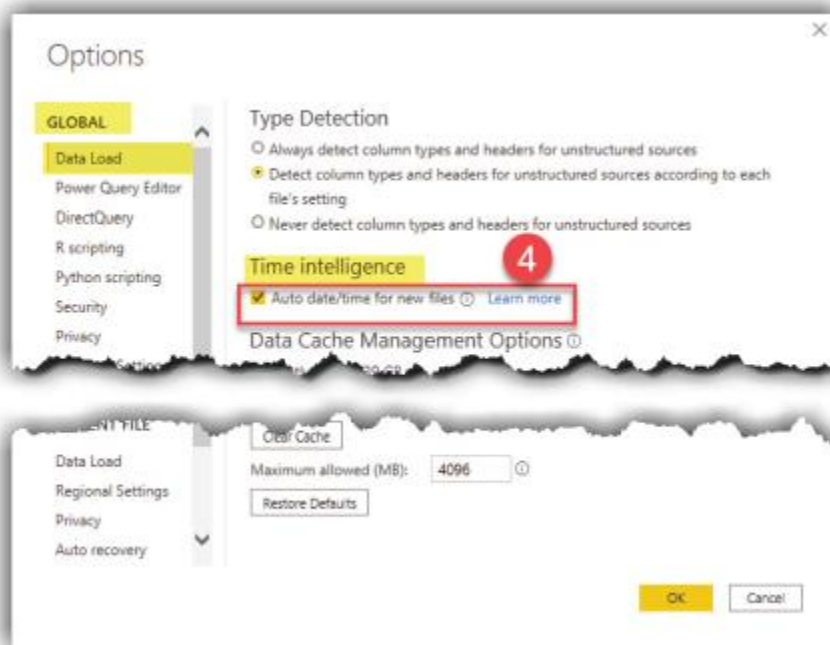
1. In Power BI Desktop click the File menu
2. Click Options and settings
3. Click Options



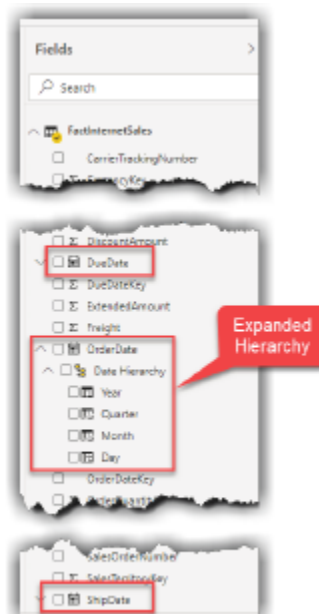
4. Under the GLOBAL section, go to the Data Load tab. Under Time intelligence - tick Auto date/time for new files (This may be already ticked).

This enables the feature globally across all Power BI files you create in future.

5. Under the CURRENT FILE, tick Auto date/time.
6. Click OK.



After enabling the Auto date/time feature, you will see that all Date or DateTime type columns have changed to Date Hierarchy.



Now that you have enabled the Auto date/time feature, you can use Quick Measures to build time intelligence measures - without having (or creating) a Date table in the model.

 It is advised to create a Date table if your source system doesn't have one, rather than relying on the Auto date/time feature. While the Auto date/time feature can be very helpful, it comes with some side effects. If you enable the Auto date/time feature, Power BI Desktop creates private Date tables in the background. Indeed, Power BI creates a Date table per column with either Date or DateTime data type regardless of whether we need to analyze them or not.

Not only is it excessive to have a Date table per column, but it can cause serious performance and storage issues. To disable this feature, just reverse back the settings we made in this section.

There are two ways to create a Date table in Power BI Desktop:

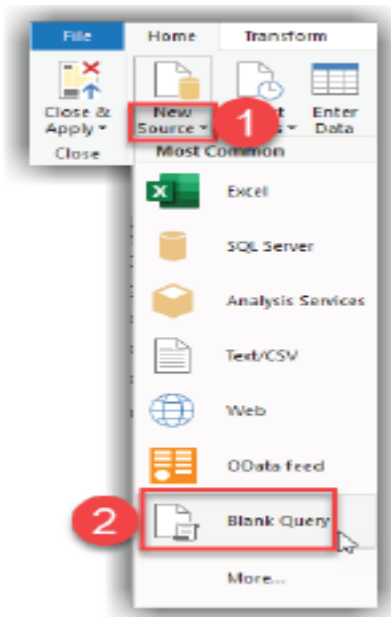
1. Create a Date table in Power Query (M) language.
2. Create a Date table using DAX.

Generating Date Table in Power Query

Open the Power Query Editor and go through the steps below to generate a Date table using Power Query (M) language.

Follow the below steps:

1. Open Power Query Editor (go to Transform Data).
2. Click New Source (or Get Data) dropdown button.
3. Click Blank Query.



4. Click the Advanced Editor button from the Home tab.
5. Copy the below script and paste in the Advanced Editor window.
6. Click Done.

```
let
    StartDate = #date(2020, 1, 1),          // Change start date
    EndDate = #date(2030, 12, 31),         // Change end date
    NumberOfDays = Duration.Days(EndDate - StartDate) + 1,
    DateList = List.Dates(StartDate, NumberOfDays, #duration(1,
0, 0, 0)),
    DataTable = Table.FromList(DateList,
Splitter.SplitByNothing(), {"Date"}, null, ExtraValues.Error),
    AddYear = Table.AddColumn(DataTable, "Year", each
Date.Year([Date]), Int64.Type),
    AddMonthNumber = Table.AddColumn(AddYear, "Month Number",
each Date.Month([Date]), Int64.Type),
    AddMonthName = Table.AddColumn(AddMonthNumber, "Month Name",
each Date.ToText([Date], "MMMM"), type text),
    AddQuarter = Table.AddColumn(AddMonthName, "Quarter", each
Date.QuarterOfYear([Date]), Int64.Type),
    AddDay = Table.AddColumn(AddQuarter, "Day", each
Date.Day([Date]), Int64.Type),
    AddDayOfWeek = Table.AddColumn(AddDay, "Day of Week", each
Date.DayOfWeekName([Date]), type text),
    AddWeekNumber = Table.AddColumn(AddDayOfWeek, "Week Number",
each Date.WeekOfYear([Date]), Int64.Type)
in
    AddWeekNumber
```

Generating Date Table using DAX

The second method to create a Date table is using DAX by creating a new calculated table. The result of the DAX formula is below:

```
1 DataTable =
2 ADDCOLUMNS(
3     CALENDARAUTO(),
4     "Year", YEAR([Date]),
5     "Month Number", MONTH([Date]),
6     "Month Name", FORMAT([Date], "MMMM"),
7     "Quarter", "Q" & FORMAT([Date], "Q"),
8     "Day", DAY([Date]),
9     "Weekday", FORMAT([Date], "dddd")
10 )
```

Date	Year	Month Number	Month Name	Quarter	Day	Weekday
01-Jul-20	2020	7	July	Q3	1	Wednesday
02-Jul-20	2020	7	July	Q3	2	Thursday
03-Jul-20	2020	7	July	Q3	3	Friday
04-Jul-20	2020	7	July	Q3	4	Saturday
05-Jul-20	2020	7	July	Q3	5	Sunday
06-Jul-20	2020	7	July	Q3	6	Monday
07-Jul-20	2020	7	July	Q3	7	Tuesday

Marking Date Table as Date

Now we have learnt what a Date table is, why we need one in the Power BI model, how to generate one if the Date table doesn't exist in the source system. We have also learnt how to configure and use the Auto date/time feature in Power BI Desktop.

But having a Date table in your data model doesn't mean that the time intelligence functions will work correctly. Your Date table must meet requirements below:

- Your Date table MUST have one column with Date 'data type'.
- The Date column MUST contain unique values.
- The Date column MUST have continuous date values - without any gaps in the dates.
- The Date column values MUST start from 1st Jan of a starting year, going up to the 31st of Dec of an ending year.
- Mark the Date table as Date.

The first four requirements are the prerequisites for the 5th requirement. Here's how to 'Mark as Date'.

1. Right click the Date table you would like to mark as Date.
2. Hover-over Mark as date table.
3. Click Mark as date table.
4. Select a Date column from the dropdown list.
5. Click OK.

7. DATA VISUALIZATION OR STORYTELLING

We can do visualization, once our data model is ready. Data visualization is the graphical representation of data. Professionals mostly use charts and graphs to put data into these visualizations because they can get insight from data which will help managers in strategic decision making.

Data visualization helps you turn all that granular data into easily understood, visually compelling and useful business information.

Data storytelling is the concept of creating a narrative using the data and analytics you have collected that help support the hypothesis of your story. Like with telling a story orally to someone else, you present a cohesive narrative that provides a final message and action to take from data insights.

Data storytelling uses several types of visuals, including scatter plots, geographic maps, timelines, line graphs, pie charts, bar charts, heat maps, and tree charts to craft a powerful story.

But when creating your data story, you first need to decide:

- What do you want your users to know?
- What do you want users to do?
- What's the narrative that will help drive that action?
- And how is your data helping drive those decisions?

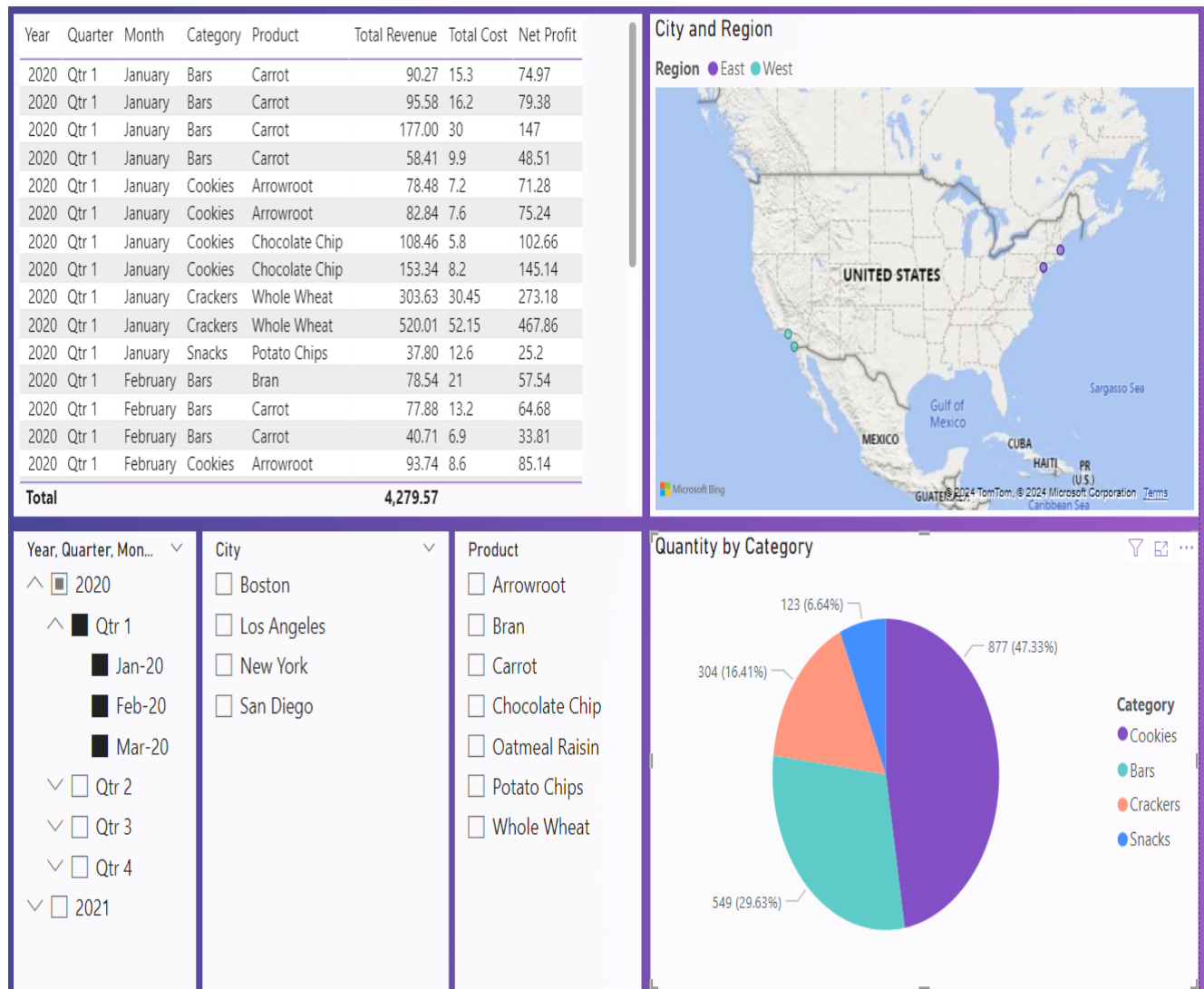
Dashboard

A dashboard is a collection of several views, enabling one to compare variety of data simultaneously. Whereas the report can encompass various pages, a Dashboard is a single page interface.

Creating a Dashboard

Once our dataset is ready with all the manipulations done, we can proceed for the Dashboard creation process. A Power BI dashboard, also known as canvas, consists of many visualizations on a single page which helps to tell a story. These visualizations called tiles are pinned to the dashboard from the reports.

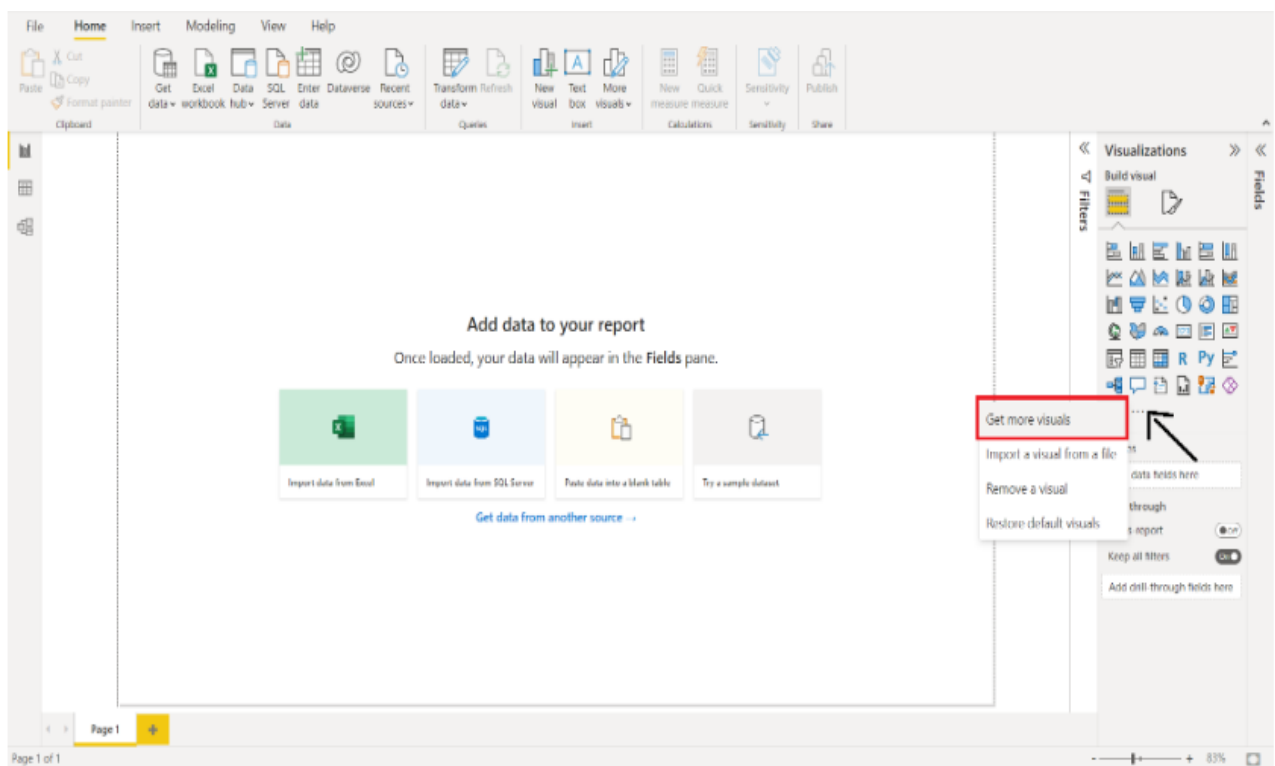
Let's now try to understand what insights we can get using our superstore data set:



8. Custom Visualization

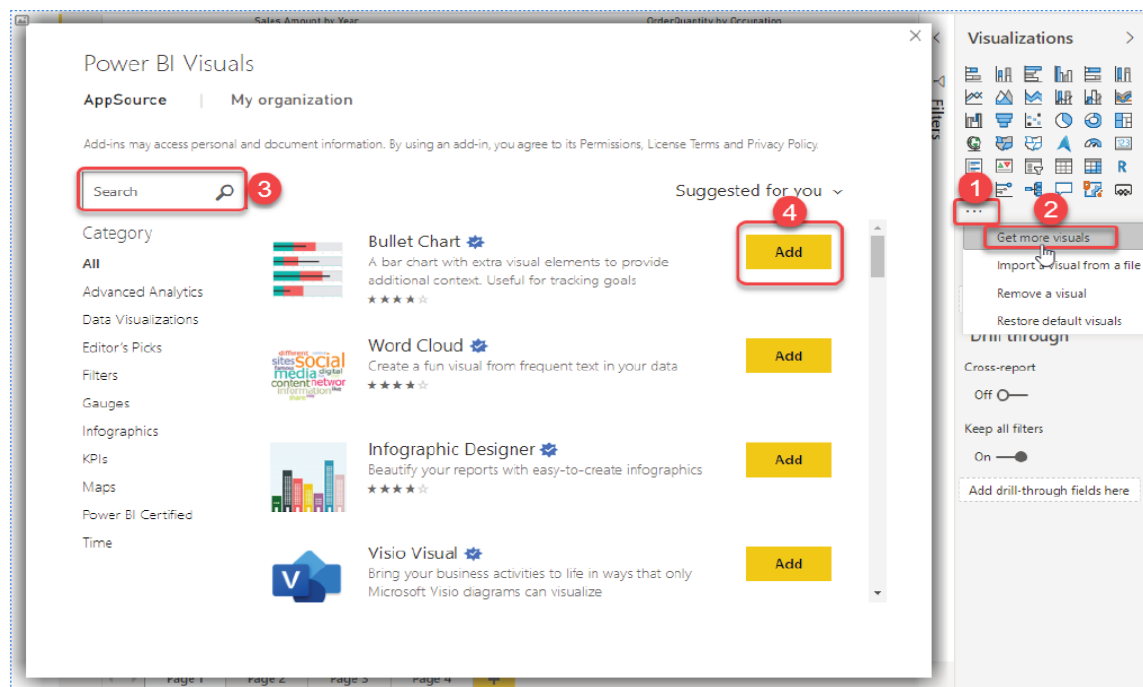
Step: I Open the Power BI Desktop.

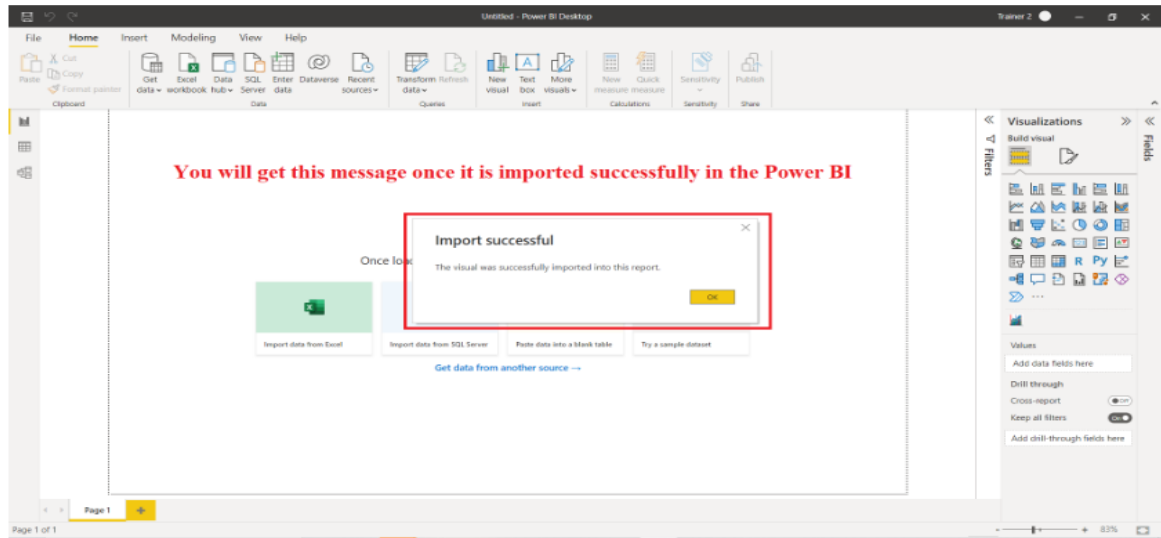
Step: II Click the three dots (...) in the visualization pane and Click “Get more visuals”.



 Custom Visuals are created by third parties, so they are NOT available within the Power BI Desktop by default. Just because they are on AppSource, it doesn't mean they are all safe to use.

Step: III You can search the visuals in the search bar and click the visual that you want.





9. Publishing to Power BI Service

Overview

Power BI Service is the cloud-based platform where you can:

- Publish reports created in Power BI Desktop.
- Share dashboards and datasets with others.
- Schedule data refreshes.
- Collaborate with your team.

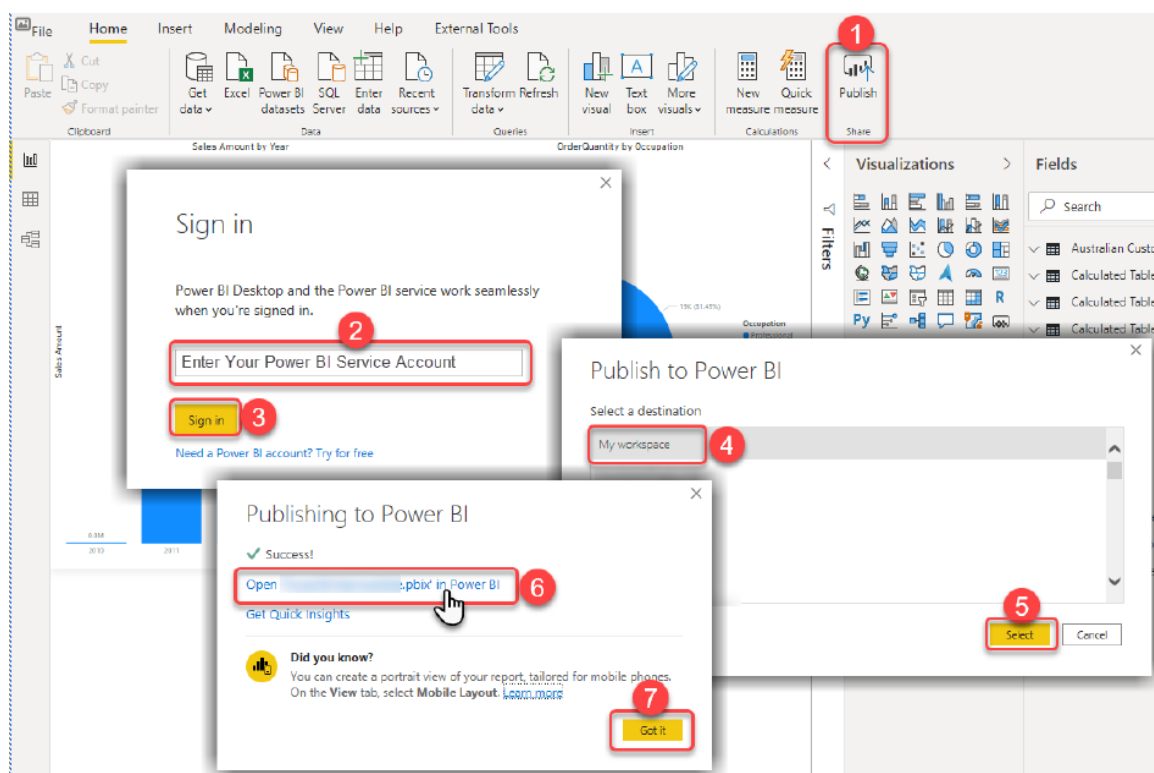
After you have finished your data visualizations in Power BI Desktop, it's time to Publish the report to Power BI Online Service.

If you have permission to publish reports to the service, it's easy. If you don't have access/permission, you won't be able to publish and share your

reports in the cloud. You can still use your reports in Power BI Desktop or share it with others by sending them the .pbix file.

Follow the steps below to publish your report to Power BI Service:

1. Click the Publish button from the ribbon bar.
2. Type in your Power BI Service credentials.
3. Click Sign in.
4. Select a workspace you want to publish your report to.
5. Click Select.
6. After your report is successfully published, you can click the report link on the Publishing to Power BI window.
7. Click Got it.



10. Fundamental Concepts of DAX

The best way to learn DAX is to write basic formulas, use them with actual data, and see the results. We will develop our understanding of DAX based on three fundamental concepts: *Syntax*, *Functions*, and *Context*. Understanding these three concepts will provide the best foundation on which to build your DAX skills.

Syntax

Syntax includes the various elements that make up a formula, or more simply, how the formula is written. For example, here is a simple DAX formula for a measure:

Total Sales = SUM (Sales [SalesAmount])

This formula includes the following syntax elements:

- 1) The measure name, *Total Sales*.
- 2) The equals sign operator (=), which indicates the beginning of the formula. When calculated, it will return a result.
- 3) The DAX function *SUM*, which adds up all the numbers in the *Sales [SalesAmount]* column.

- 4) Parenthesis (), which surround an expression that contains one or more arguments. Most functions require at least one argument. An argument passes a value to a function.
- 5) The referenced table, [Sales](#).
- 6) The referenced column, [\[SalesAmount\]](#), in the Sales table. With this argument, the SUM function knows on which column to aggregate a SUM.

The best way is to read a DAX formula in plain English. For example, you can read this formula as:

For the measure named Total Sales, calculate (=) the SUM of values in the [SalesAmount] column in the Sales table.

You might be thinking, "Isn't this measure doing the same thing as if I were to just add the SalesAmount field to my report?" Well, yes. But there's a good reason to create our own measure that sums up values from the SalesAmount field: We can use it as an argument in other formulas. This may seem a little confusing now, but as your DAX formula skills grow, knowing this measure will make your formulas and your model more efficient.

Now use a function, [SUM](#). Functions are pre-written formulas that make it easier to do complex calculations and manipulations with numbers, dates, time, text, and more.

You can see that the column name [SalesAmount] was preceded by the Sales table in which the column belongs. It's a good practice to include the table name in your measure formulas, even when in the same table.

It's important your formulas have the correct syntax. In most cases, if the syntax isn't correct, a syntax error is returned. In other cases, the syntax may be correct, but the values returned might not be what you're expecting.

 Power BI has a very good user interface that has an [auto complete feature](#). For example, when you type a DAX formula, [IntelliSense](#) prompts you with the full name: Table Name [Column Name].

Let's create a simple formula. This task will help you further understand formula syntax and how the suggestions feature in the formula bar can help you.

Task: Create a measure formula

1. Open the Superstore Power BI Desktop file.
2. In Report view, in the field list, right-click the Sales table, and then select [New Measure](#).

3. In the formula bar, replace **Measure** by entering a new measure name, *Previous Quarter Sales*.
4. After the equals sign, type the first few letters *CAL*, and then double-click the function you want to use. In this formula, we want to use the **CALCULATE** function.

We will use the CALCULATE function to filter the amounts we want to sum by an argument we pass to the CALCULATE function. This is referred to as nesting functions. The CALCULATE function has at least two arguments. The first is the expression to be evaluated, and the second is a filter.

5. After the opening parenthesis (for the CALCULATE function, type *SUM* followed by another opening parenthesis.

Next, we will pass an argument to the SUM function.

6. Begin typing *Sal*, and then select Sales [SalesAmount], followed by a closing parenthesis.

This is the first expression argument for our CALCULATE function.

7. Type a comma (,) followed by a space to specify the first filter, and then type *PREVIOUSQUARTER*.

We will use the PREVIOUSQUARTER [time intelligence function](#) to filter SUM results by the previous quarter.

8. After the opening parenthesis (for the PREVIOUSQUARTER function, type *Calendar [Date Key]*.

The [PREVIOUSQUARTER](#) function has one argument, a column containing a continuous range of dates. In our case, that's the Date Key column in the Calendar table.

9. Close both the arguments being passed to the PREVIOUSQUARTER function and the CALCULATE function by typing two closing parenthesis)).

Your formula should now look like this:

```
Previous_Quarter_Sales = CALCULATE(SUM(Sales[SalesAmount]),  
PREVIOUSQUARTER(Calendar[date Key]))
```

10. Select the checkmark in the formula bar or press Enter to validate the formula and add it to the model.

Wow! You have done it! You just created a complex measure by using DAX. This formula will calculate the total sales for the previous quarter, depending on the filters applied in a report. For example, if we put Sales Amount and

our new Previous Quarter Sales measure in a chart, and then add Year and Quarter of Year as Slicers, we would get something like this.

We have covered some important aspects of DAX formulas:

- This formula included two functions. **PREVIOUSQUARTER**, a time intelligence function, is nested as an argument passed to **CALCULATE**, a filter function.

In this formula, we also used filters. Filters narrow down what will be calculated. In this case, we selected one filter as an argument, which is the result of another function.

The **CALCULATE** function is one of the most powerful functions in DAX. It is used to modify the current filter context.

 **DAX formulas can contain up to 64 nested functions.**

Functions

Functions are predefined formulas that perform calculations by using specific values, called arguments, in a particular order or structure. Arguments can be other functions, another formula, expression, column references, numbers, text, logical values such as TRUE or FALSE, or constants.

DAX includes the following categories of functions: **Aggregation & iterator** functions, **Date and Time**, **Time Intelligence**, **Information**, **Logical**,

Mathematical, Statistical, Text, Parent/Child, and Other functions. Many of the functions in DAX are similar to Excel formulas; however, DAX functions are unique in the following ways:

- A DAX function always references a complete column or a table. If you want to use only particular values from a table or column, you can add filters to the formula.
- If you need to customize calculations on a row-by-row basis, DAX provides functions that let you use the current row value or a related value as a kind of argument, to perform calculations that vary by context.
- DAX includes many functions that return a table rather than a value. The table isn't displayed but is used to provide input to other functions. For example, you can retrieve a table and then count the distinct values in it or calculate dynamic sums across filtered tables or columns.
- DAX includes a variety of time intelligence functions. These functions let you define or select date ranges, and perform dynamic calculations based on them. For example, you can compare sums across parallel periods.
- Excel has a popular function, VLOOKUP. DAX functions don't take a cell or cell range as a reference like VLOOKUP does in Excel. DAX functions take a column or a table as a reference.

 In Power BI Desktop you are working with a relational data model. Looking up values in another table is easy, and in most cases, you don't need to create any formulas at all. In Power BI we create relationships between tables to lookup values.

One of the best places to learn details about each of the DAX functions is in the [DAX Function Reference](#). There are currently over 250 functions in DAX.

Context

Context is one of the most important DAX concepts to understand. There are two types of contexts in DAX: row context and filter context.

Row Context

📖 Definition: The context of a single row when evaluating an expression.

📖 Where it comes from:

- Iterators like SUMX, FILTER, ADDColumns.
- Calculated columns (because they evaluate row by row)

Row context is most easily thought of as the current row. It applies whenever a formula has a function that applies filters to identify a single row in a table. The function will inherently apply a row context for each row of the table over which it is filtering. This type of row context most often applies to measures.

Filter Context

📖 Definition: The set of filters applied to a calculation that determines which rows are included.

Where it comes from:

- Slicers, visual filters, page/report filters.
- Relationships between tables.
- Functions like CALCULATE that modify filters.

In Power BI, [filter context](#) refers to the set of filters applied to data when a calculation is performed. It determines which rows of data are considered during evaluation. This concept is crucial for understanding how measures and DAX formulas work.

Filter context doesn't exist in place of row context; rather, it applies in addition to row context. For example, to further narrow down the values to include in a calculation, you can apply a filter context, which not only

specifies the row context, but also specifies a particular value (filter) in that row context.

Filter context is easily seen in your reports. For example, when you add Total Cost to a visualization, and then add Year and Region, you are defining a filter context that selects a subset of data based on a given year and region.

What is Filter Context?

- It's the combination of all filters applied to a calculation, including:
 - **Visual filters** (e.g., slicers, page filters, report filters)
 - **Row context** converted to filter context (when using CALCULATE)
 - **Implicit filters** from relationships between tables

Types of Filter Context

1. **Visual Filter Context**
Comes from slicers, filters, and visuals on the report page.
2. **Row Context Converted to Filter Context**
Happens when you use CALCULATE or similar functions in DAX.
3. **Context from Relationships**
Filters propagate through relationships between tables.

Example

If you have a measure like:

Total Sales = SUM (Sales [Amount])

- If you apply a slicer for Year = 2024, the filter context includes only rows where Year = 2024.
- If you add another slicer for Region = Europe, the filter context becomes Year = 2024 AND Region = Europe.

Why is filter context so important to DAX?

Because while filter context can most easily be applied by adding fields to a visualization, filter context can also be applied in a DAX formula by defining a filter using functions such as ALL, RELATED, FILTER, CALCULATE, by relationships, and by other measures and columns.

- Filter context affects measures, CALCULATE, and ALL functions.
- Understanding it helps you write correct DAX formulas.

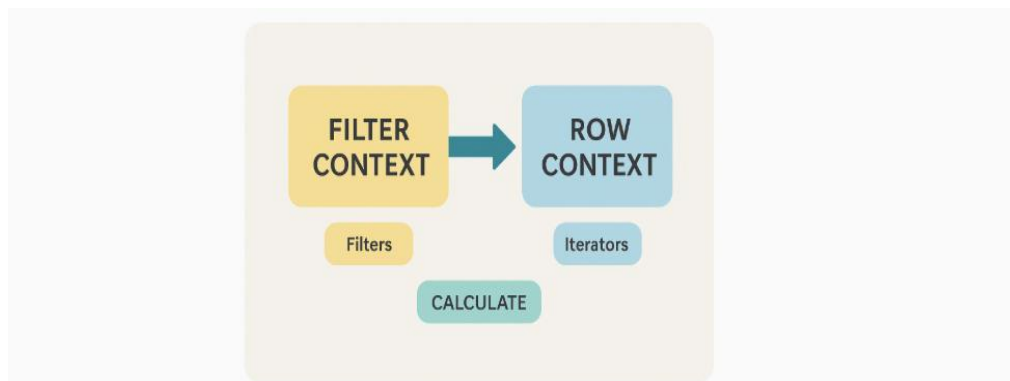
Key Differences

Aspect	Filter Context	Row Context
Scope	Applies to a set of rows (filtered)	Applies to one row at a time
Created by	Filters, slicers, CALCULATE, visuals	Iterators, calculated columns
Conversion	Row context can become filter context using CALCULATE	Filter context does not become row context

Example: Combining Both

```
CALCULATE (  
    SUM (Sales [Amount]),  
    FILTER (Sales, [Amount] > 1000)  
)
```

- CALCULATE changes filter context.
- FILTER introduces row context inside its expression.



- Filter Context (yellow) comes from slicers, filters, and relationships.
- Row Context (blue) is created by iterators or calculated columns.
- The CALCULATE function converts row context into filter context, allowing you to apply additional filters dynamically.

Example Formula:

High Value Sales =
CALCULATE (
 SUM (Sales [Amount]),
 FILTER (Sales, Sales [Amount] > 1000)
)

How it flows:

1. FILTER (Sales, Sales [Amount] > 1000) introduces row context (evaluates each row).
2. CALCULATE takes that condition and converts it into filter context.
3. The final SUM uses the new filter context (only rows where Amount > 1000).

CONDITIONS

IF vs SWITCH FUNCTION

IF Function

Purpose: Used for simple conditional logic.

Syntax:

IF(<logical_test>, <value_if_true>, <value_if_false>)

Example:

```
IF (Sales [Amount] > 1000, "High", "Low")
```

Best for:

- When you have one condition or nested conditions (though nesting too many IFs can get messy).
- Simple true/false checks.

SWITCH Function

Purpose: Used for multiple conditions without heavy nesting.

Syntax:

SWITCH(<expression>, <value1>, <result1>, <value2>, <result2>, ..., <else>)

Example:

```
SWITCH (TRUE (),  
    Sales [Amount] > 1000, "High",  
    Sales [Amount] > 500, "Medium",  
    "Low"  
)
```

Summary

DAX can be a little tricky to learn, but there are many resources available online, such as [Data Analysis Expressions \(DAX\) Reference](#). After reading through this guide and experimenting with a few of your own formulas, you can learn more about other DAX concepts and formulas that can help you solve your own business problems.

Tips and Tricks for Measuring Data in Power BI

Some tips and tricks for measuring data in Power BI include:

- Creating hierarchy in your data model.
- Using Parent-Child functions to create smart hierarchies.
- Creating time-based metrics using Date tables.
- Using IF, SWITCH, and CALCULATE functions to refine calculations.

 Another useful tip for measuring data in Power BI is to use the [Quick Measure](#) feature. This feature allows you to quickly create common calculations such as running totals, moving averages, and year-over-year comparisons without having to write complex formulas. Additionally, you

can customize these calculations to fit your specific needs. Quick Measures can save time and effort when creating reports and dashboards in Power BI.

How to Display Measures in Visualizations

Adding measures to a visualization in Power BI is relatively straightforward. Simply select the visualization, then add the measure to the values field. You can also customize how the measure is displayed in the visualization by selecting various visual components like bars, lines, or colors.

It is important to note that the type of measure you add to a visualization can greatly impact the insights you gain from the data. For example, if you add a measure that calculates the total sales for a product, you may see a spike in sales during a certain time-period. However, if you add a measure that calculates the average sales per day for that same product, you may notice a different trend in the data.

Managing and Organizing Measures in Power BI As the number of measures in a report grows, it becomes essential to organize and manage them. Some useful ways to manage measures in Power BI include:

- Grouping measures based on specific requirements.
- Creating a folder structure to store different measures together.
- Using meaningful names and descriptions to make measures easy to identify.

Another way to manage measures in Power BI is to use the "[Display Folder](#)" feature. This allows you to group measures together and display them in a collapsible folder structure, making it easier to navigate and find specific measures.

It's also important to regularly review and clean up your measures. This involves removing any unused or redundant measures, as well as consolidating similar measures into a single, more efficient measure. This not only helps to keep your report organized, but also improves its performance.

Troubleshooting Common Issues with Measures

Some of the common issues that users may face when creating measures in Power BI include:

- Calculations returning inaccurate results.
- Errors when creating DAX formulas.
- Calculation taking a long time to execute.

To troubleshoot these issues, you may need to review the DAX language syntax, check the data model for errors, or upgrade your data source to improve performance.

Incorporating Advanced Features with Measures in Power BI

Power BI has several advanced features that can be used with measures to create more meaningful insights. These features include:

- Advanced Analytics by integrating with R and Python
- Using AI and Machine Learning tools
- Dataflows for preparing and cleansing data
- Connecting to cloud services like MS Azure
- Measuring Data Across Multiple Platforms with Power BI

Using Natural Language Queries to Create Measures

Power BI allows users to create measures using natural language queries. This feature makes it easy for users to create measures without knowing the DAX language syntax. Natural language queries are created by simply typing out a question or calculation formula in plain English into the search bar, and Power BI will convert it into DAX.

11. ERRORS HANDLING & TROUBLESHOOTING

Power BI errors can occur at different stages—data loading, modeling, DAX calculations, or visualization.

Types of Errors in Power BI

1. Data Connectivity Errors

- Occur when connecting to a source (e.g., SQL, Excel).
- Examples: *Invalid credentials, Timeout, Gateway not reachable.*

2. Data Transformation Errors (Power Query)

- Issues during data shaping in Power Query.
- Examples: *Column not found, Data type mismatch, Formula.*
- *Firewall error.*

3. Modeling Errors

- Problems in relationships or calculated columns.
- Examples: *Circular dependency, Ambiguous relationships.*

4. DAX Calculation Errors

- Syntax or logical issues in measures.
- Examples:
 - The expression refers to multiple columns. Multiple columns cannot be converted to a scalar value.
 - Cannot find name 'ColumnName'.

5. Refresh Errors

- Occur during scheduled or manual refresh.
- Examples: *Gateway offline, Data source credentials expired.*

6. Visualization Errors

- Issues in visuals due to missing fields or filters.
- Examples: *Field not found; Blank visuals due to filter context.*

Error Handling Techniques

- Use IFERROR () or IF (ISBLANK ()) in DAX

Example:

Safe Division = DIVIDE (SUM (Sales [Amount]), SUM (Sales [Quantity]), 0)

 DIVIDE () avoids divide-by-zero errors.

Validate Data Types in Power Query

- Use Table.TransformColumnTypes() to ensure consistency.

Fallback Values

Example:

Sales Amount = IF (ISBLANK (SUM (Sales [Amount])), 0, SUM (Sales [Amount]))

Error Indicators

- Use conditional formatting or KPI visuals to highlight anomalies.

Troubleshooting Steps

1. **Check Query Dependencies**
 - In Power Query, use *View* → *Query Dependencies*.
2. **Enable Error Details**
 - In Power Query, click on the error cell → *View Error*.
3. **Validate Relationships**
 - Ensure correct cardinality and active relationships.
4. **Test DAX in a Table Visual**
 - Helps isolate filter context issues.
5. **Monitor Refresh History**
 - In Power BI Service → *Dataset* → *Refresh History*.
6. **Use Performance Analyzer**
 - Identify slow or failing visuals.

DAX Error Handling Patterns

1. Handle Divide-by-Zero Errors

Instead of:

Ratio = SUM (Sales [Amount]) / SUM (Sales [Quantity])

Use:

Ratio = DIVIDE (SUM (Sales [Amount]), SUM (Sales [Quantity]), 0)

Note: DIVIDE () automatically returns 0 (or a custom value) if the denominator is zero.

2. Replace Blank Values

Sales Amount = IF (ISBLANK(SUM(Sales [Amount])), 0, SUM(Sales[Amount]))

Or shorter:

Sales Amount = COALESCE (SUM (Sales [Amount]), 0)

✓ COALESCE () returns the first non-blank value.

3. Catch Errors in Calculations

Safe Measure = IFERROR ([Complex Measure], 0)

Note: IFERROR () returns a fallback value if the expression fails.

4. Validate Conditions Before Calculation

Valid Sales = IF(SUM(Sales [Quantity]) > 0, SUM(Sales[Amount]), BLANK())

It prevents invalid calculations when conditions aren't met.

5. Handle Missing Relationships

Customer Sales = IF(HASONEVALUE(Customer [CustomerID]), SUM(Sales[Amount]), BLANK())

6. Use TRY Functions (Newer DAX)

Power BI now supports TRY functions:

Safe Conversion = TRY(CONVERT([Column], INTEGER))

Returns blank if conversion fails instead of throwing an error.

7. Combine Multiple Checks

Safe KPI = IFERROR (IF(SUM(Sales [Quantity]) > 0, DIVIDE(SUM(Sales[Amount]), SUM(Sales[Quantity]), 0), 0))

DAX FORMULAS LIST

1. Date and Time Functions

Sr	Function	Description	Syntax
1	CALENDAR	Returns a table with a single column containing a set of dates.	<code>=CALENDAR ("01/01/2020","12/31/2021")</code>
2	TODAY		<code>=TODAY ()</code>
3	CALENDARAU TO	Returns a table with a single column that contains a set of dates.	<code>CALENDARAU TO ()</code> Or, <code>CALENDARAU TO ([fiscal_year_end_month])</code> <code>CALENDARAU TO (3)</code> if the fiscal year ends in March.
4	DATE	Returns the specified date in datetime format.	<code>DATE(<year>, <month>,<day>)</code>
5	DATEDIFF	Calculates the time interval between two dates.	<code>DATEDIFF(<start_date>, <end_date>, <interval>)</code>
6	DATEVALUE	Converts a date in the form of text to a date in datetime format.	<code>DATEVALUE (date text)</code>
7	EOMONTH	To calculate maturity dates or due dates that fall on the last day of the month.	<code>EOMONTH ('Table'[Date],0)</code>

8	DATEADD	To add or subtract a specified time interval from a date. For example, you can use DateAdd to calculate a date 30 days from today.	DATEADD(<dates>,<number_of_intervals>,<interval>) This formula calculates dates that are either forward or backward from the dates in the current context. For example, Sales PY = Calculate ([Total Sales], DateAdd (Calendar [Date],-1,Year))
---	----------------	--	---

2. Time Intelligence Functions

Sr	Function	Description	Syntax
1	CLOSINGBALANCEMONTH	Evaluates the expression at the last date of the month in the current context.	CLOSINGBALANCEMONTH (<expression>,<dates>[,<filter>])
2	CLOSINGBALANCEYEAR	Evaluates the expression at the last date of the year in the current context.	CLOSINGBALANCEYEAR (<expression>,<dates>[,<filter>][,<year_end_date>])
3	DATESINPERIOD	Returns a table that contains a column of dates that begins with a specified start date and continues for the specified number and type of date intervals.	DATESINPERIOD (<dates>, <start_date>, <number_of_intervals>, <interval>)
4	DATESMTD	Returns a table that contains a column of the dates for the month to date, in the current context.	DATESMTD (<dates>)
5	DATESYTD	Returns a table that contains a column of the dates for the year to date, in the current context.	DATESYTD (<dates> [,<year_end_date>])
6	ENDOFMONTH	Returns the last date of the month in the current context for the specified column of dates.	ENDOFMONTH (<dates>)

7	FIRSTNONBLANK	Returns the first value in the column, column, filtered by the current context, where the expression is not blank	FIRSTNONBLANK(<column>,<expression>)
8	WEEKDAY	This function returns a number from 1 to 7 identifying the day of the week of a date. By default, 1 (Sunday) to 7 (Saturday)	WEEKDAY(<date>, <return_type>) For example, to get the weekday number where the week starts on Monday, you can use: WEEKDAY('Date_Table'[Date], 2)
9	OPENINGBALANCEMONTH	Evaluates the expression at the first date of the month in the current context.	OPENINGBALANCEMONTH(<expression>, <dates>[,<filter>])
10	DayOfWeekName	If you want to get the name of the day, you can use the FORMAT function.	DayOfWeekName = FORMAT('Table'[Date], "dddd")
11	PREVIOUSQUARTER	Returns a table that contains a column of all dates from the previous quarter, based on the first date in the dates column, in the current context.	PREVIOUSQUARTER(<dates>)
12	PREVIOUSYEAR	Returns a table that contains a column of all dates from the previous year, given the last date in the dates column, in the current context.	PREVIOUSYEAR(<dates>[,<year_end_date>])
13	STARTOFMONTH	Returns the first date of the month in the current context for the	STARTOFMONTH(<dates>)

		specified column of dates.	
1 4	TOTALMTD	Evaluates the value of the expression for the month to date, in the current context.	TOTALMTD(<expression>,<dates>[,<filter>])
1 5	TOTALQTD	It calculates the total for a measure from the start of the quarter to the current date in the quarter.	TOTALQTD(<expression>, <dates> [, <filter>]) Example: QTD_Sales = TOTALQTD(SUM(Sales[SalesAmount]), 'Calendar'[Date])
1 6	TOTALYTD	Evaluates the year-to-date value of the expression in the current context.	TOTALYTD (SUM ('Table' [Sales]), 'Calendar'[Date])

3. Filter or Context Functions

Sr.	Function	Description	Syntax
1	ALL	Returns all the rows in a table or all the values in a column, ignoring any filters applied.	<code>ALL({<table> [<column>[, <column>...[]] })</code>
2	ALLEXCEPT	Removes all context filters in the table, except filters that have been applied to the specified columns.	<code>=CALCULATE(SUM(ExpSales_USD[SalesAmount_USD]), ALLEXCEPT (Date Time, Date Time [CalendarYear]))</code>
3	ALLSELECTED	Removes context filters from columns and rows in the current query, while retaining all other context filters or explicit filters.	<code>ALLSELECTED([<tableName> <columnName>])</code>
4	CALCULATE	Evaluates an expression	<code>CALCULATE(<expression>,<filter1>,<filter2>...)</code>

		in a context that is modified by the specified filters	
5	FILTERS	To apply filters.	<code>FILTER(<table>, <filter>)</code> Example: <code>Filtered Sales = FILTER (Sales, Sales[Amount] > 1000)</code>
6	USERELATIONSHIP	This function is particularly useful in scenarios where you have inactive relationships that you want to activate temporarily for a specific calculation.	Suppose you have a Sales table and a Date table with two relationships: one active relationship based on Order Date and one inactive relationship based on Delivery Date. To calculate the total sales based on the Delivery Date, you can use the USERELATIONSHIP function within CALCULATE: Example: Total Sales by Delivery Date = <code>CALCULATE (</code> <code> SUM (Sales [SalesAmount]),</code> <code> USERELATIONSHIP (Sales [DeliveryDate], Date [Date])</code> <code>)</code>

7. Logical Functions

Sr	Function	Description	Syntax
1	IF FUNCTION	Checks whether a condition is met, and returns one value if TRUE, and another value if FALSE	IF (logical_test>, <value_if_true>, value_if_false) Example: =IF('Sales'[Amount] > 10, "Yes", "No")
2	IFERROR	To avoid an error.	IFERROR (value, value_if_error) Example: =IFERROR (20/0,888)
3	AND		=AND('Table'[Column1] > 5, 'Table'[Column2] < 10)
4	OR		=OR('Table'[Column1] > 5, 'Table'[Column2] < 10)
5	NOT		=NOT('Table' [Flag])
6	SWITCH	Returns different results depending on the value of an expression.	SWITCH(<expression>,<value>,<result>[,<value>,<result>]...[,<else>])
7	TRUE	Returns the logical value TRUE.	=IF(SUM('Sales'[Amount])>1000000, TRUE (), false ())

8. Maths & Statistical Functions

Sr.	Function	Description	Syntax
1	DIVIDE	It automatically handles division by zero cases.	Profit Margin = DIVIDE ([Total Profits], [Total Sales], 0) * You can give an additional parameter e.g., "NIL"
2	MINIMUM	Returns the minimum number in a column.	=MIN('Table'[Column])
3	RANDBETWEEN	Returns a random number in the range between two numbers you specify	=RANDBETWEEN (1,10)
4	CONFIDENCE.NORM	The confidence interval is a range of values.	CONFIDENCE.NORM(alpha,standard_dev,size)
5	MEDIAN	Returns the median of numbers in a column.	MEDIAN(<column>)
6	MEDIANX	Returns the median number of an expression evaluated for each row in a table.	MEDIANX(<table>, <expression>)
7	NORM.DIST	Returns the normal	NORM.S. DIST (1.96, 0, 1, TRUE)

		distribution for the specified mean and standard deviation.	
8	NORM.S. DIST	Returns the standard normal distribution (has a mean of zero and a standard deviation of one).	NORM.S. DIST (Z, Cumulative)
9	NORM.INV	Returns the value to a given probability in a normal distribution.	NORM.INV (Probability, Mean, Standard_dev)
10	STDEV.P	Returns the standard deviation of the entire population.	STDEV.P(<ColumnName>)

9. Text Functions

Sr.	Function	Description	Syntax
1	COMBINEVALUES	Joins two or more text strings into one text string.	COMBINEVALUES (<delimiter>, <expression>, <expression>[, <expression>]...)
2	CONCATENATE	Joins two text strings into one text string.	CONCATENATE (<text1>, <text2>)
3	CONCATENATEX	Concatenates the result of an expression evaluated for each row in a table.	CONCATENATEX (<table>, <expression>[, <delimiter> [, <orderBy_expression> [, <order>]]...])
4	EXACT	Compares two text strings and returns TRUE if they are the same, FALSE otherwise.	EXACT (<text1>,<text2>)
5	FORMAT	Converts a value to text according to the specified format.	FORMAT (<value>, <format_string>[, <locale_name>])
6	LEFT	Returns the specified number of characters from the start of a text string.	LEFT (<text>, <num_chars>)
7	LEN	Returns the number of characters in a text string.	LEN (<text>)

8	LOWER	Converts all letters in a text string to lowercase.	LOWER(<text>)
9	MID	Returns a string of characters from the middle of a text string, given a starting position and length.	MID(<text>, <start_num>, <num_chars>)
10	REPLACE	REPLACE replaces part of a text string, based on the number of characters you specify, with a different text string.	REPLACE(<old_text>, <start_num>, <num_chars>, <new_text>)
11	REPT	Repeats text a given number of times.	REPT(<text>, <num_times>)
12	RIGHT	RIGHT returns the last character or characters in a text string, based on the number of characters you specify.	RIGHT(<text>, <num_chars>)
13	SEARCH	Returns the number of the character at which a specific character or text string is first	SEARCH(<find_text>, <within_text>[, <start_num>][, <NotFoundValue>]])

		found, reading left to right.	
14	SUBSTITUTE	Replaces existing text with new text in a text string.	SUBSTITUTE(<text>, <old_text>, <new_text>, <instance_num>)
15	TRIM	Removes all spaces from text except for single spaces between words.	TRIM(<text>)
16	UPPER	Converts a text string to all uppercase letters.	UPPER (<text>)

10. Advanced Functions

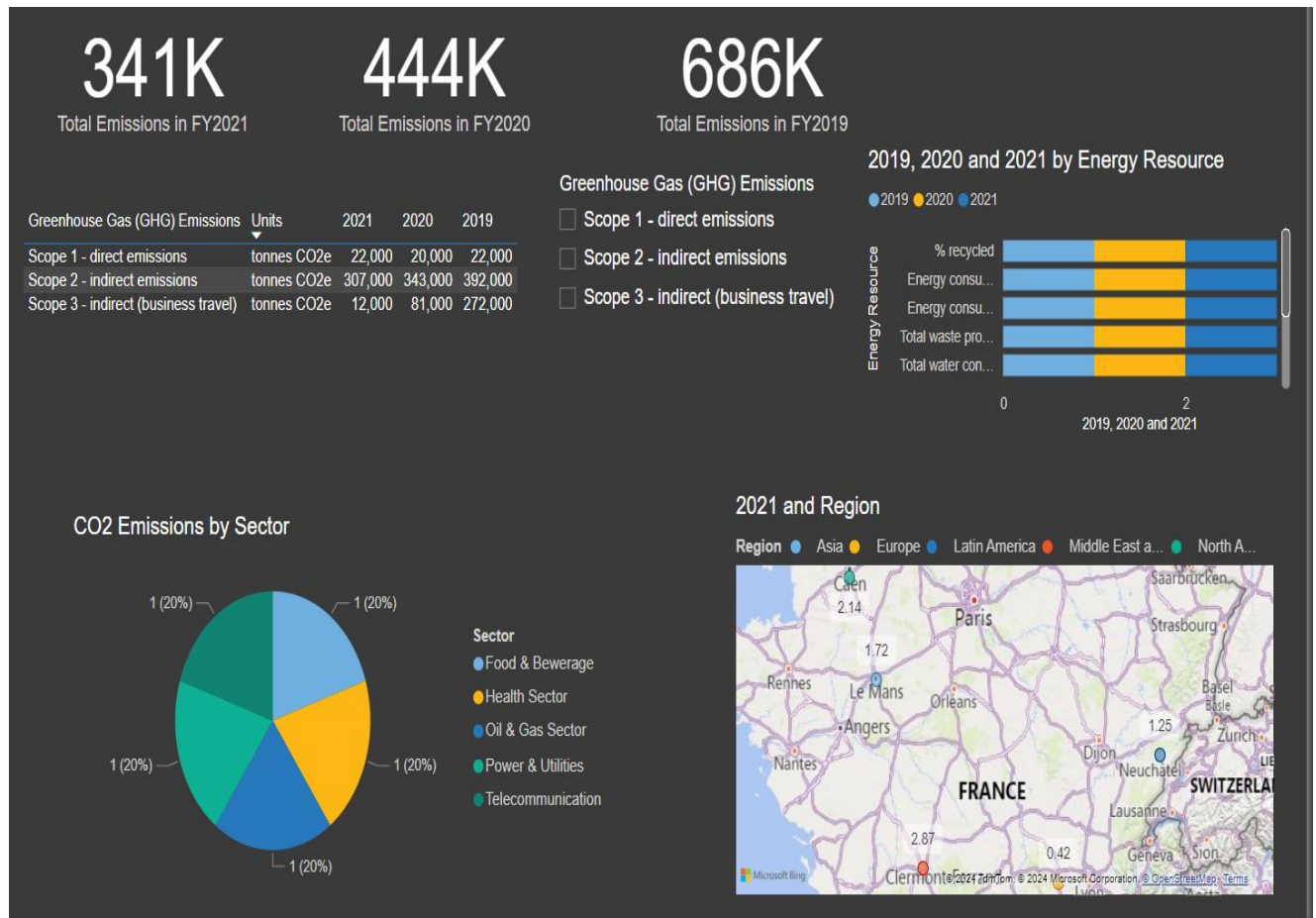
Sr	Function	Description	Syntax
1	PREDICT	The PREDICT function is used for making predictions based on historical data.	PREDICT ([Historical Data], [Future Periods], [Confidence Level])
2	FORECAST	Used for time series forecasting.	Forecast = FORECAST.ETS ([Target Value], [Timeline], [Seasonality], [Data Completion], [Aggregation])
3	ISBLANK	It is used to check whether a value is blank and returns a TRUE if the value is blank, and FALSE otherwise.	IsSalesBlank = IF(ISBLANK('SalesTable'[Sales]), TRUE, FALSE) It can be used in combination with other functions to handle missing data. Sales Growth = IF (ISBLANK ([Previous Year Sales]), BLANK (), ([Total Sales] - [Previous Year Sales]) / [Previous Year Sales])
4	ISERROR	The ISERROR function in DAX checks whether a value is an error and returns TRUE if	ErrorCheck = IF(ISERROR([Numerator] / [Denominator]), BLANK (), [Numerator] / [Denominator])

		it is, and FALSE otherwise.	
5	PATH	This is particularly useful for handling parent-child hierarchies.	PATH (<ID_ColumnName>, <Parent_ColumnName>) Example: Suppose you have an Employees table with columns EmployeeID and ManagerID. You can create a calculated column to list all the managers for each employee: EmployeePath = PATH (Employee [EmployeeID], Employee[ManagerID])
6	ISFILTERED	The ISFILTERED function in DAX checks whether a column or table is being filtered directly and returns TRUE if it is, and FALSE otherwise	Suppose you have a table Sales with a column Region. You can create a measure to check if the Region column is being filtered: IsRegionFiltered = ISFILTERED(Sales [Region])

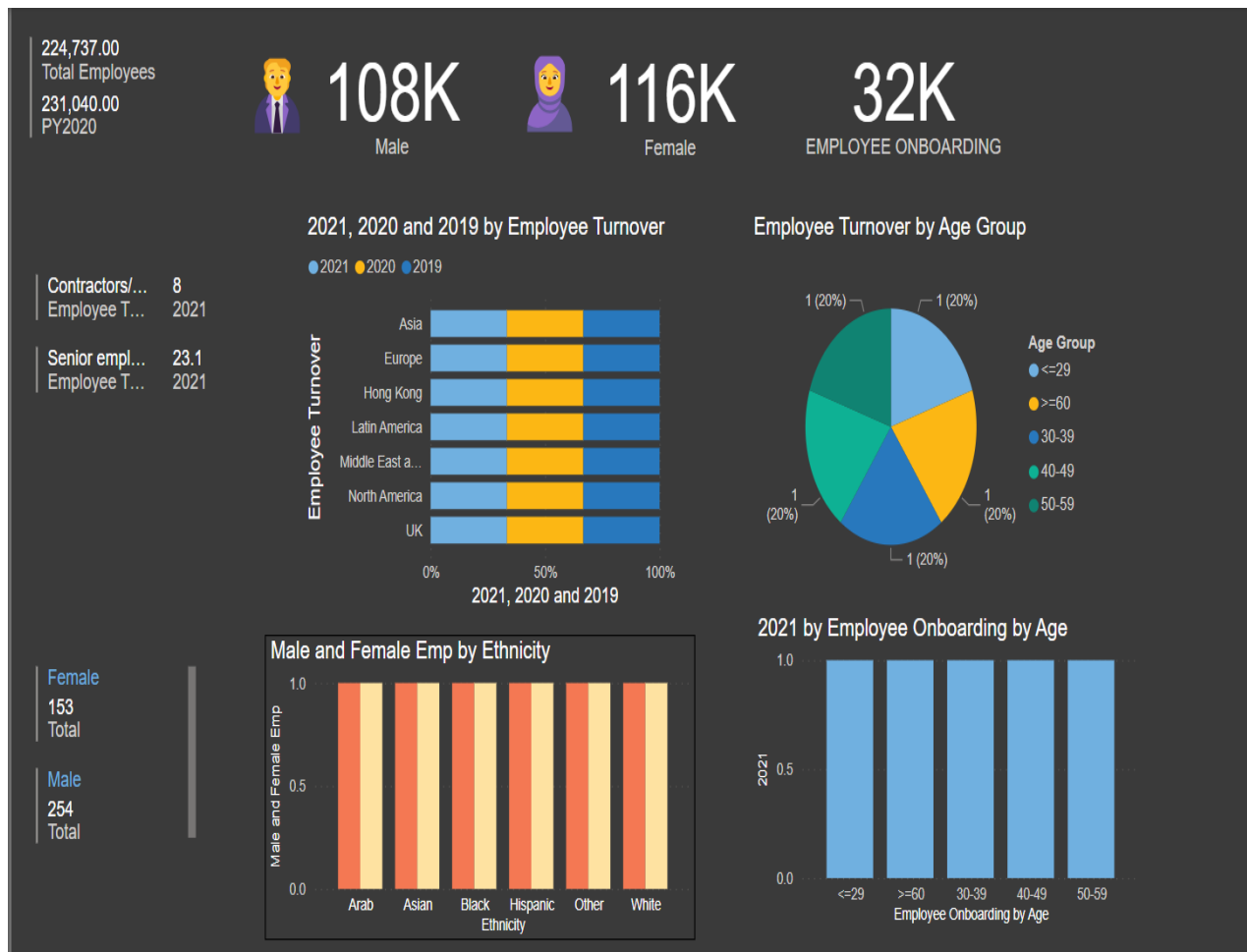
11. CASE STUDY

HSBC BANK ESG DATAPACK

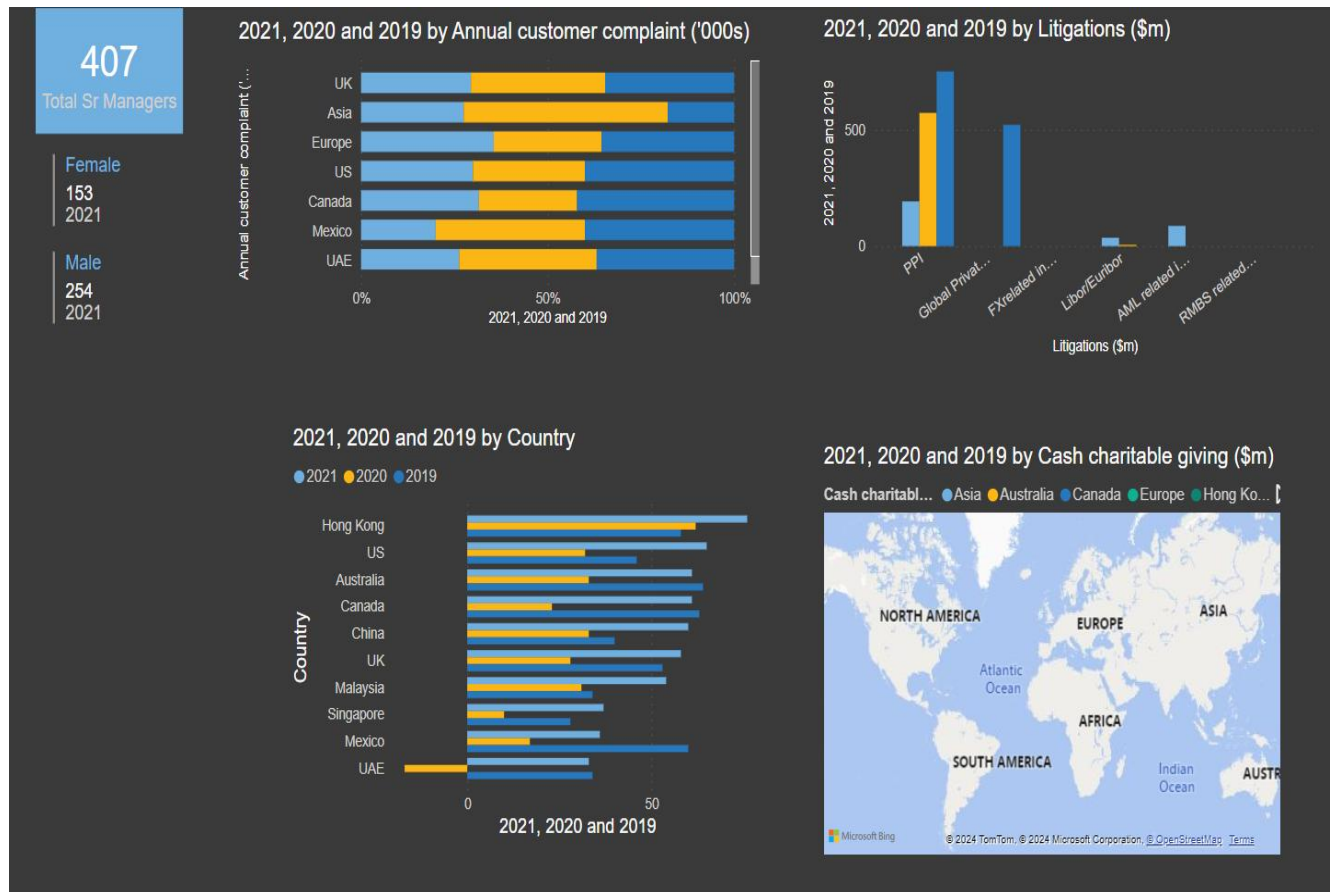
ESG Dashboard – Environmental



ESG Dashboard - Social



ESG Dashboard – Governance



12. Sample Files



HSBC-ESG-datapack
-2021.xlsx



Superstore
Data.xlsx

BY: SHAHZAD Z KHAN

FINANCIAL CONTROL DASHBOARDS

Income Statement Analysis

Business insights from Income Statement Analysis:

Profitability & Revenue Trends

- ✓ Are we generating a net profit or operating at a loss?
- ✓ What is our operating profit margin, and is it improving?
- ✓ How do our profit levels compare to the previous period or industry benchmarks?
- ✓ Are certain segments or products more profitable than others?
- ✓ Is our revenue growing steadily over months or years?
- ✓ Are we retaining customers or relying heavily on new sales?



FINANCIAL CONTROL DASHBOARDS

CFO Dashboard



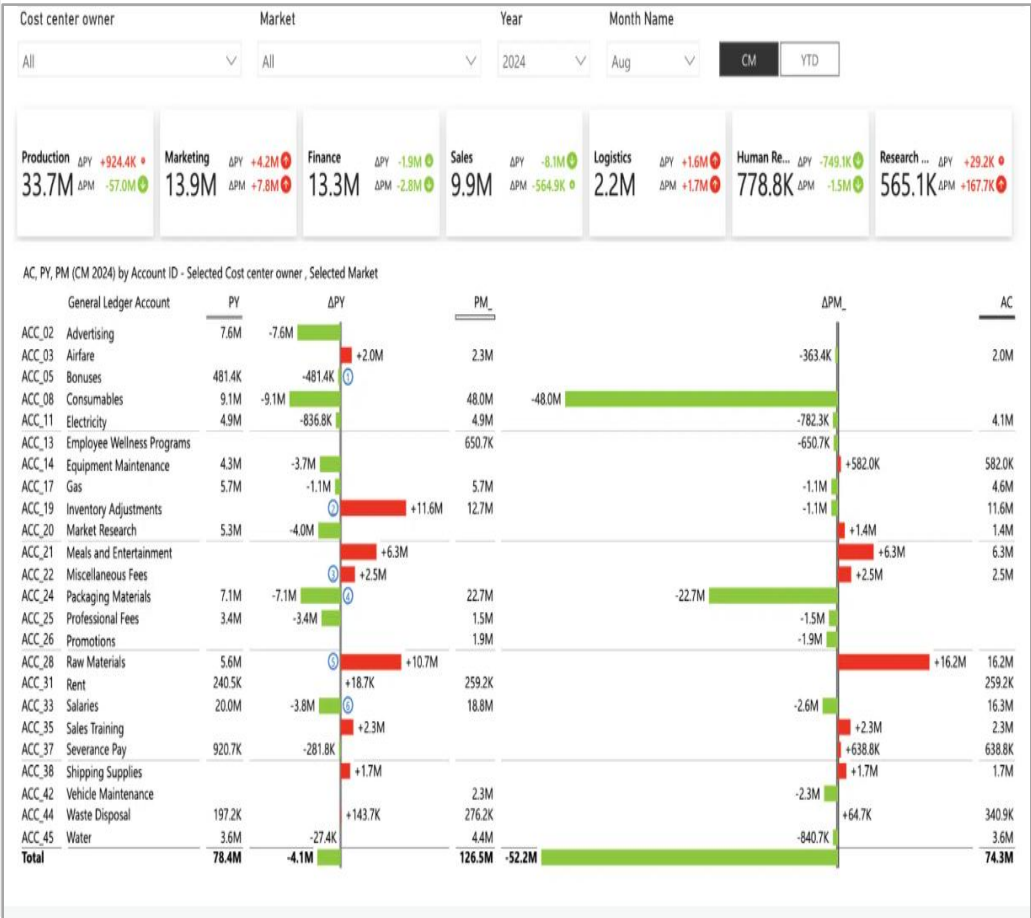
Business insights from CFO Dashboard:

Profitability & Cost Trends

- ✓ Optimize financial oversight and streamline month-end closing activities for controllers & CFOs.
- ✓ This analysis includes cost trend, profitability, forecasting, and budgeting.
- ✓ Perfect for visualizing cost trends and assessing financial impact.

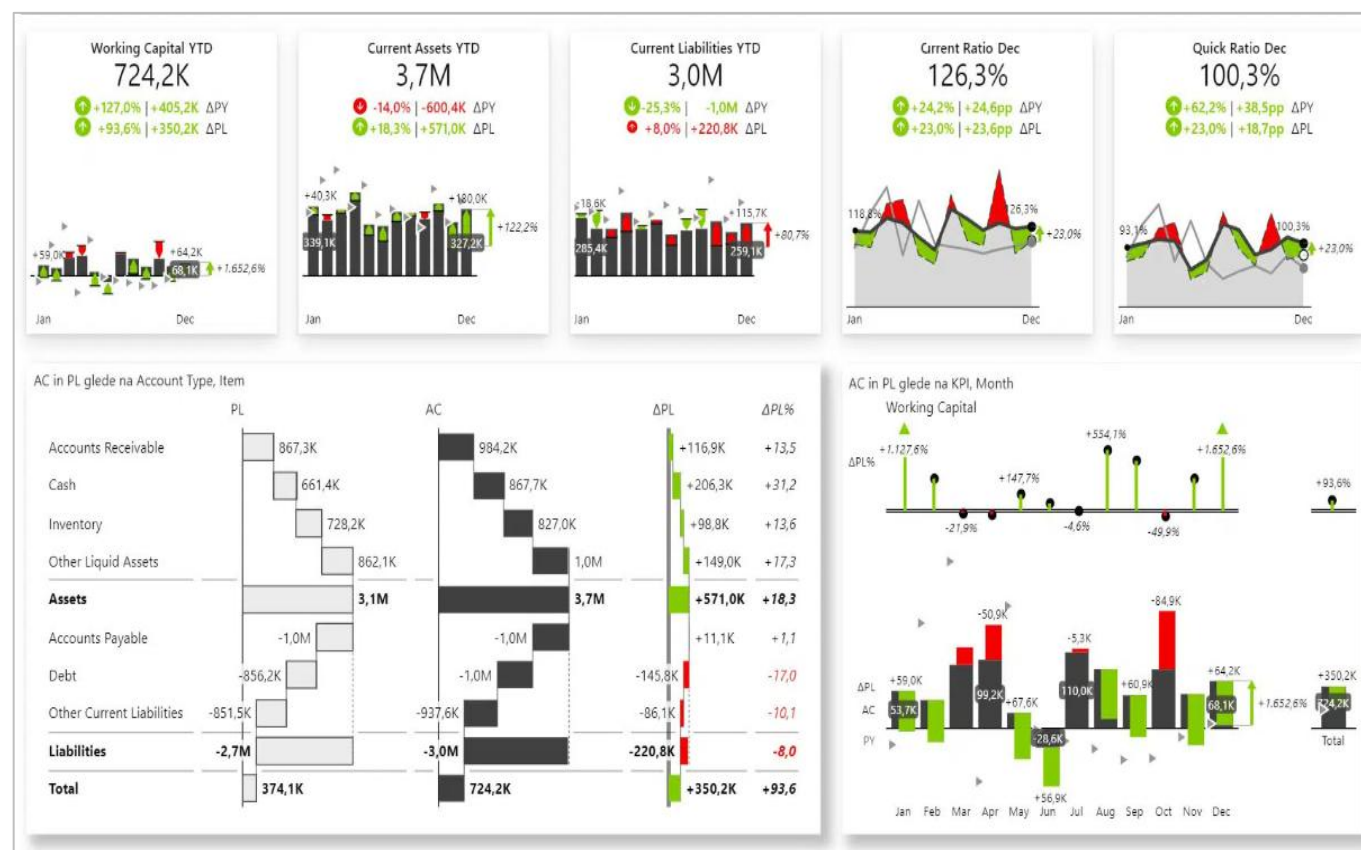
Costing Analysis

- ✓ It empowers users to track costs, identify discrepancies, and run OPEX forecast scenarios.
- ✓ It provides cost analysis and improving resource allocation within your



FINANCIAL CONTROL DASHBOARDS

Working Capital Analysis



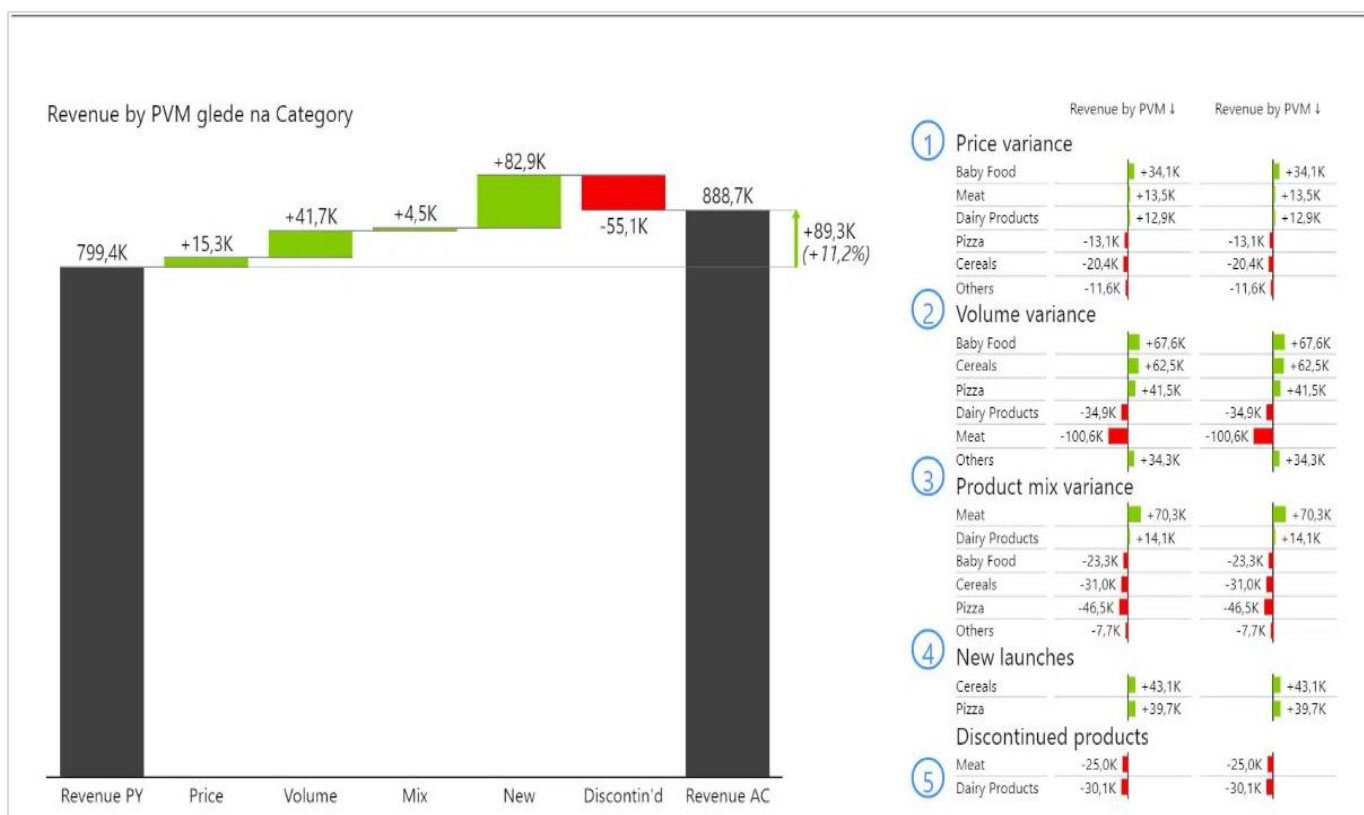
Business insights from Working Capital Analysis:

✓ Key Metrics

- ✓ Provides a clear overview of the current assets (CA) and current liabilities (CL).
- ✓ Shows the difference between CA and CL – the working capital.
- ✓ Shows the balance sheet at-a-glance, visualized with vertical waterfall charts.
- ✓ Analyze the trends of each KPI.

FINANCIAL CONTROL DASHBOARDS

Price, Volume, Mix (PVM) Analysis



Business insights from PVM Analysis:

☒ Key Metrics

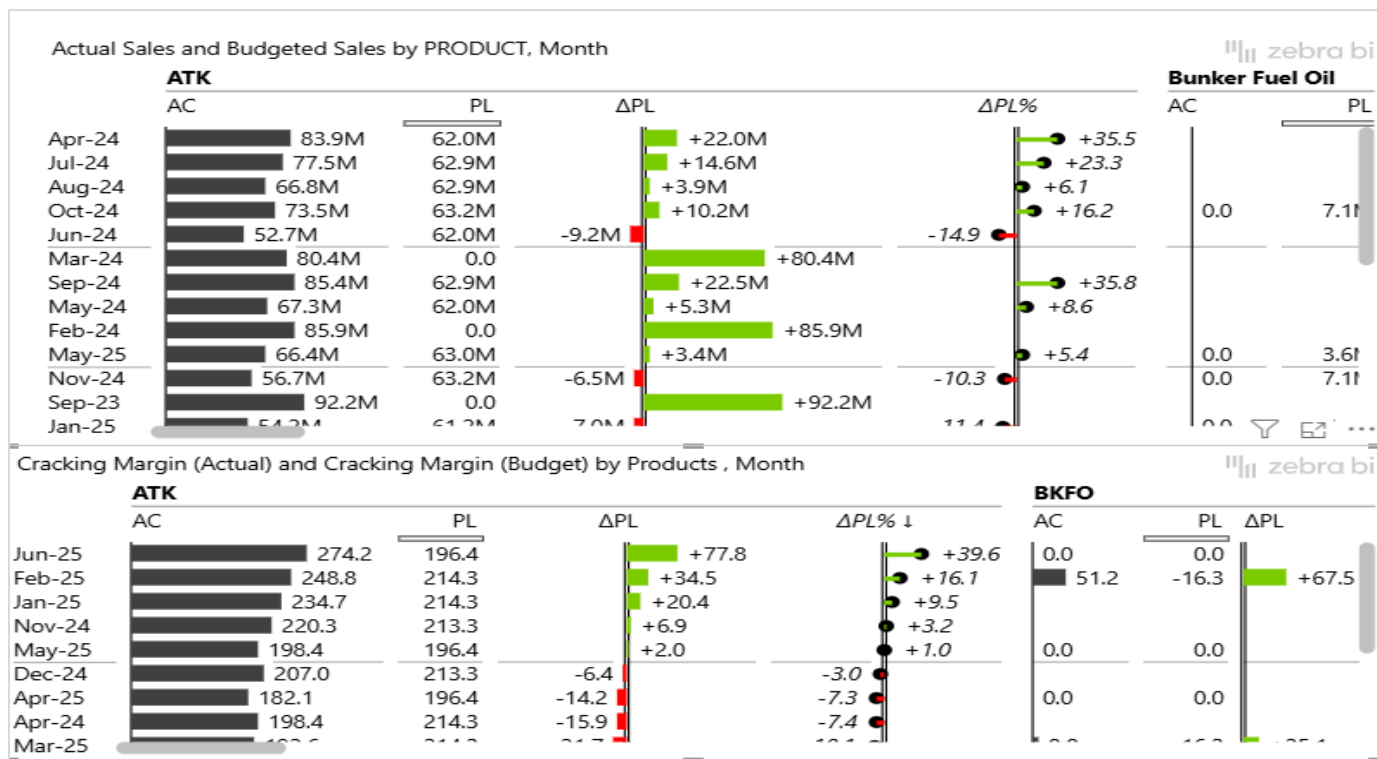
- ✓ What's driving our revenue growth or decline?
- ✓ Are we selling more, or just charging more?
- ✓ Which product categories or regions are improving, and which are underperforming?
- ✓ How are new or discontinued products affecting our results?

Beneficial for:

- ✓ Finance teams (for profitability analysis).
- ✓ Sales & marketing departments (to track price and volume trends).
- ✓ Business strategists (for informed decisions about product and pricing).
- ✓ Retail & eCommerce businesses (to analyze SKU performance).

FINANCIAL CONTROL DASHBOARDS

Variance Analysis



Business insights from Variance Analysis:

☒ Key Metrics

- ✓ It shows actual vs budgeted sales for each product by month.
- ✓ Identify performance gaps with variance analysis and clear visual indicators.
- ✓ It shows actual vs budgeted gross margin for each product by month.
- ✓ Explore key metrics interactively using filters, hierarchies, and small multiples.
- ✓ Drive strategic decision making with insights that connect business actions to financial outcomes.

FINANCIAL CONTROL DASHBOARDS

Benefits for Client

This comprehensive financial control dashboard provides a strong mechanism and deep insights about sales of each product for various granularity, total revenue, total cost, and net profit margin. Also, shows current year performance compared to last year along with % change. It gives detailed income statement analysis, comparative analysis of consolidated financials, working capital analysis and financial control or performance monitoring.

It performs variance analysis to calculate actual vs budgeted sales & gross margin for each product by month. It calculates price, volume, mix (PVM) analysis, which explains how much the change in revenue is due to price effect, how much is due to the change in the quantity sold and how much is the impact of change in product mix ratio.

Data Transformation & Analysis

Contact: +965 609 111 83

Email: fintraining.info@gmail.com

Course Overview

This course will enable participants to get expertise in the world of **data transformation** and **automation**. You will learn how to extract data from multiple sources and transform it into meaningful information to get insight and analysis. We will show you how to automate data connections & transformations and extract and consolidate data from multiple files. These skills will help you spend less time on data manipulation and more on your analysis and decision making.

Power Query in Excel / Power BI is an essential tool for any **business intelligence** analyst that gathers and processes your data, analyzes, and provides data driven insights that enables managers, business analysts and key decision makers to make informed decisions real time. It allows one to ask questions and get answers about the data.

Objectives

This program will equip you to play a pivotal role between data and the business, and help you combine your business acumen and finance knowledge with data analytics expertise.

The participants will be able to:

- Understand different data types & data extraction techniques.
- Data transformation & wrangling approaches to make it useful for analysis.
- Apply advanced analytical and statistical methods and applications for finance.
- Understand complex business data issues including query management.
- Turn raw data into meaningful information using data visualization techniques.

Data Synergy

Data Synergy aims to help the clients become data intelligent, information-driven organizations, where data-based decision-making is embedded into daily operations from the top to bottom level, which ultimately leads to better processes and profitability.

We help you decipher the language of analytics, demonstrating how insights can fuel powerful, fact-based decisions. We assist you build the structures and processes that convert raw data into meaningful information which helps in taking strategic decisions. We also help you identify hidden performance drivers in all the functional areas, from finance, production, sales, inventory, resource management, marketing and more.

We apply automation & predictive analytics using AI and machine learning tools to forecast sales, streamline processes and help the finance function to spotlight the red flags and perform better financial control and cash management. We equip you with the broad insights and new thinking that allow you to act decisively in solving challenges, seizing opportunities, and sparking innovative growth.

At Data Synergy, we specialize in helping businesses leverage the capability of Microsoft Power BI and other Power Platform solutions. Our team of experts are here to assist you and optimize your experiences, whether you need help with implementation, training, or custom solutions. We host a range of workshops and corporate trainings offering hands-on experience, valuable insights and more, covering solutions such as Data Transformation & Analysis, Power Apps, Automation using Python & R language and other cutting-edge solutions.