



Data Analytics Using Power BI

▼ Contents

- Introduction to Power BI
- Data Preparation, Modelling and Visualization
- Data Transformations
- Interactive Visualizations
- Advance Transformations & Visualizations
- Advance Dashboard Enhancements
- DAX Essentials
- DAX - CALCULATE & Filtering
- DAX - Time Intelligence
- Publishing Reports & Power BI Services
- Scheduling & Data Gateway
- Row Level & Page Level Security
- M Language Fundamentals

▼ 1- Introduction to Power BI

▼ What is Power BI ?

What is power bi.pdf

Before we dive into the details, it's important to first answer the question, "What exactly is Power BI?" Power BI is essentially a data visualization tool. However, it's not just any data visualization tool; it is the most commonly used tool in the industry and is a clear leader in Gartner's Magic Quadrant. This means Power BI is recognized as the best in terms of its ability to execute and its completeness of vision.

With Power BI, we can visualize and analyze our data, but before we can do that, a significant amount of data transformation work is usually required. This is an essential first step in any reporting process. We often need to fix errors in the data, rename columns, or replace certain values, so a large portion of our effort is spent on data transformation before we can visualize our data.

Even after the data transformation is complete, we may not be ready to visualize our data just yet because we often work with multiple data sources. This requires us to load multiple tables into Power BI, which then need to be connected through relationships. These relationships allow us to combine and interactively use the tables together, which is why building a data model is crucial. All of this work is done within Power BI Desktop, a local application that we will download and install on our machine.

When we want to share our reports and collaborate with team members, we need to publish the report to the cloud. There are different plans available for this, which we will explore in detail later in this course. Additionally, there's a Power BI mobile app that we can download and install on our smartphone, enabling us to access reports from anywhere directly from our mobile device.

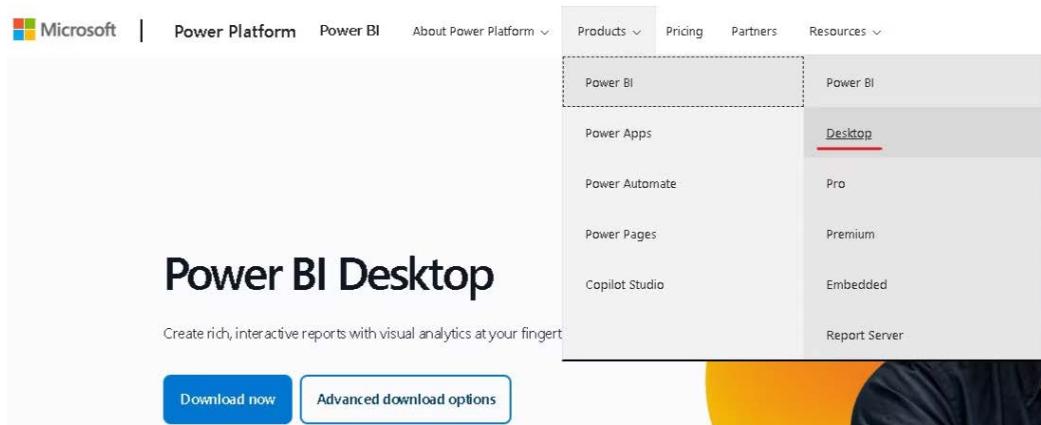
As you can see, most of the heavy lifting is done in Power BI Desktop. Therefore, if you want to master Power BI, you need to master Power BI Desktop. This is where the bulk of the work happens, and that's why we will start with Power BI Desktop. In the next section, we will download and install it on our local machine.

▼ Power BI Desktop Download and Installation

To install **Power BI Desktop** on your local machine, follow these steps:

1. Visit the Website:

- Go to powerbi.microsoft.com and navigate to the **Products** section to find **Power BI Desktop**.

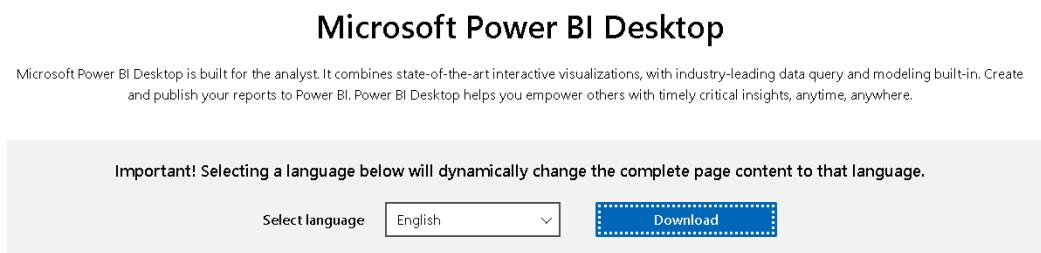


2. Download Options:

- You have two download options:
 - **Microsoft Store:** Click **Download Free** to install via the Microsoft Store, which will enable automatic updates.
 - **Manual Download:** For more flexibility, choose **Advance Download Options or Language Options** to manually download the installer.

3. Choose the Installer:

- Select your preferred language (e.g., English) and click **Download**.



- Choose the **64-bit version** (recommended for most systems) or the 32-bit version if needed.

Choose the download you want

<input type="checkbox"/> File Name	Size
<input type="checkbox"/> PBIDesktopSetup.exe	483.6 MB
<input type="checkbox"/> PBIDesktopSetup_x64.exe	525.1 MB

[Download](#) Total size: 0 bytes

4. Install Power BI Desktop:

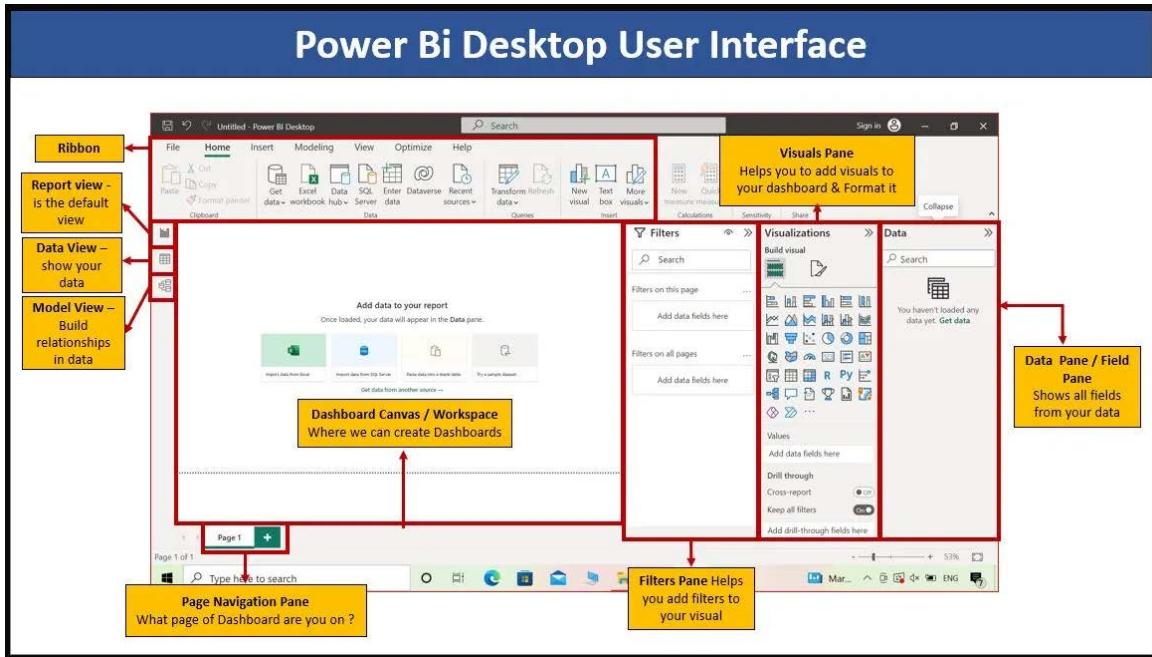
- Once downloaded, locate the installation file and double-click to start the installation.
- Follow the installation wizard, accept the terms, and select the default settings.
- Check the option to **Create a Desktop Shortcut** for easy access.

5. Launch Power BI Desktop:

- After installation, Power BI Desktop will launch automatically, ready for use.

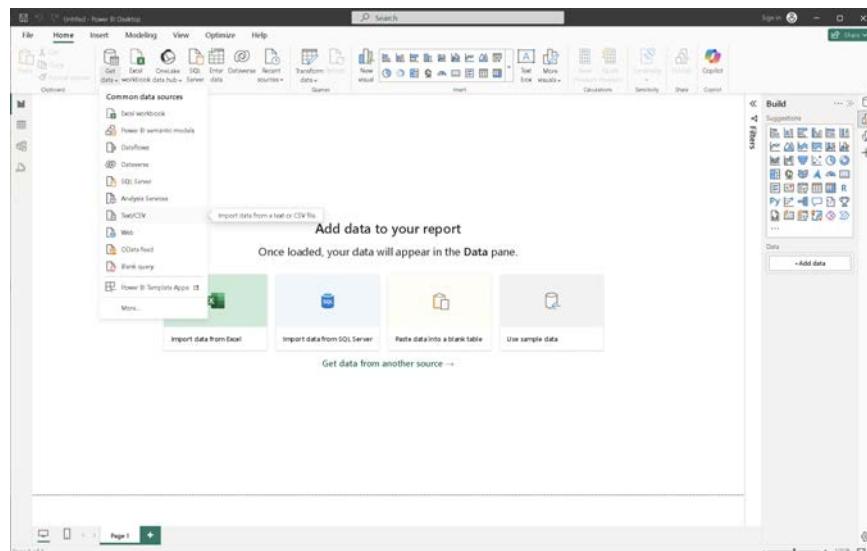
▼ Navigate Interface

Let's take a quick look at the Power BI interface by loading some data to help illustrate its various components. We will explore these steps in greater detail later, but for now, the focus is on understanding the interface.



1. Loading Data:

- First, go to **Get Data** and select **Text/CSV** to load a CSV file.



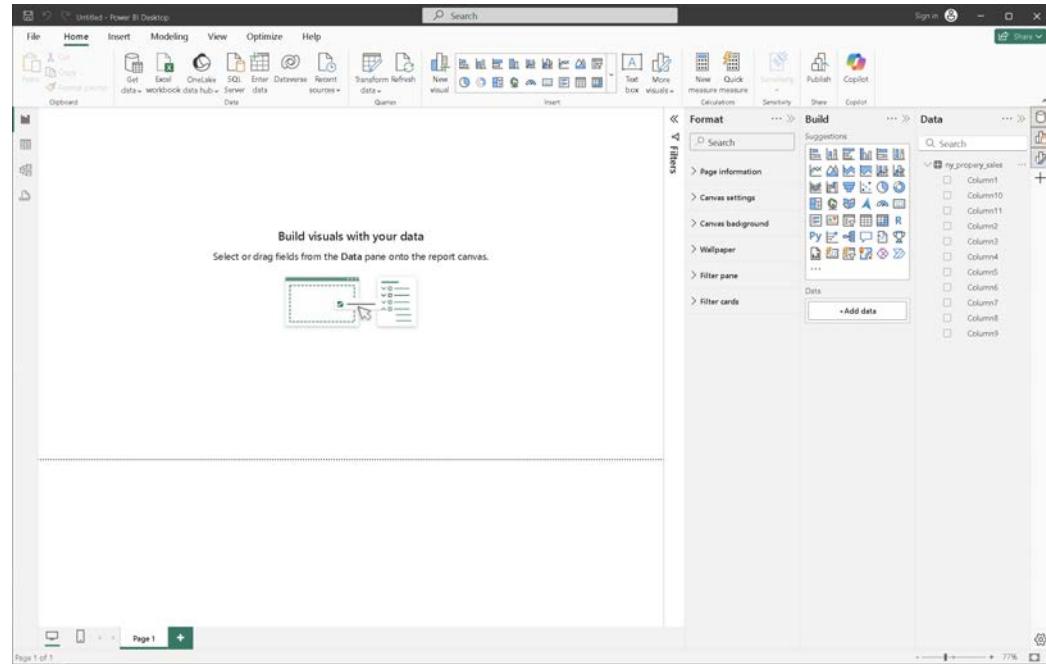
- Navigate to the file location, select it, and click **Open**. You will see a preview of the data, which Power BI detects based on the first 200 rows.

The screenshot shows the Power BI Desktop application window. In the center, a 'ny_property_sales.csv' file is being loaded into the Power Query Editor. The editor displays a preview of the data with the following columns:

ID	Area	NEIGHBORHOOD	BUILDING CLASS CATEGORY	ADDRESS	ZIP CODE	LAND SQUARE FEET
14108	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	520 EAST 86TH STREET, AC	10028	
14109	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	530 EAST 86TH STREET, BE	10028	
14110	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	530 EAST 86TH STREET, BD	10028	
14111	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	510 EAST 86TH STREET, BA	10028	
14112	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	510 EAST 86TH STREET, BC	10028	
14113	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	510 EAST 86TH STREET, SC	10028	
14114	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	525 EAST 86TH STREET, 2ND/F	10028	
14115	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	525 EAST 86TH STREET, BG	10028	
14116	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	525 EAST 86TH STREET, BD	10028	
14117	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	525 EAST 86TH STREET, 11A	10028	
14118	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	525 EAST 86TH STREET, 10	10028	
14119	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	525 EAST 86TH STREET, 10C	10028	
14120	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	525 EAST 86TH STREET, SC	10028	
14121	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	535 EAST 86TH STREET, 2AM	10028	
14122	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	535 EAST 86TH STREET, 1B	10028	
14123	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	530 EAST END AVENUE, 1E	10028	
14124	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	530 EAST END AVENUE, 1B	10028	
14125	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	530 EAST END AVENUE, 1B	10028	
14126	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	515 EAST 86TH STREET, 5BC	10028	
14127	Manhattan	UPPER EAST SIDE (2996)	10 COOPS ELEVATOR APARTMENTS	525 E 86TH ST, 3D	10028	

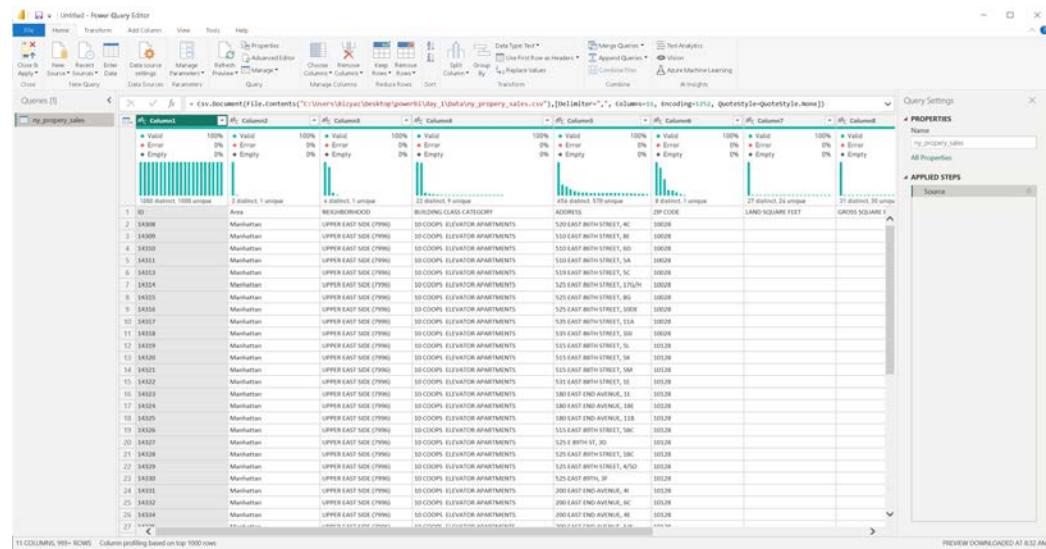
At the bottom of the editor, there are 'Extract Table Using Examples', 'Load', 'Transform Data', and 'Cancel' buttons.

- You can either directly load the data or choose to transform it using the **Power Query Editor**. For now, let's load the data without transformation.
- Once the data is loaded, it will appear on the right side of the screen under **Fields**. You can expand this to view the different columns available in the dataset.



2. Power Query Editor:

- This is an integrated tool in Power BI that allows you to transform and restructure your data.



3. Report View:

- This is the default view where you will build your visualizations. The main area is the canvas where visualizations are placed, and the data fields are on the right.

4. Other Views:

- **Data View:** Provides a tabular view of your data, allowing you to examine the data structure and format.

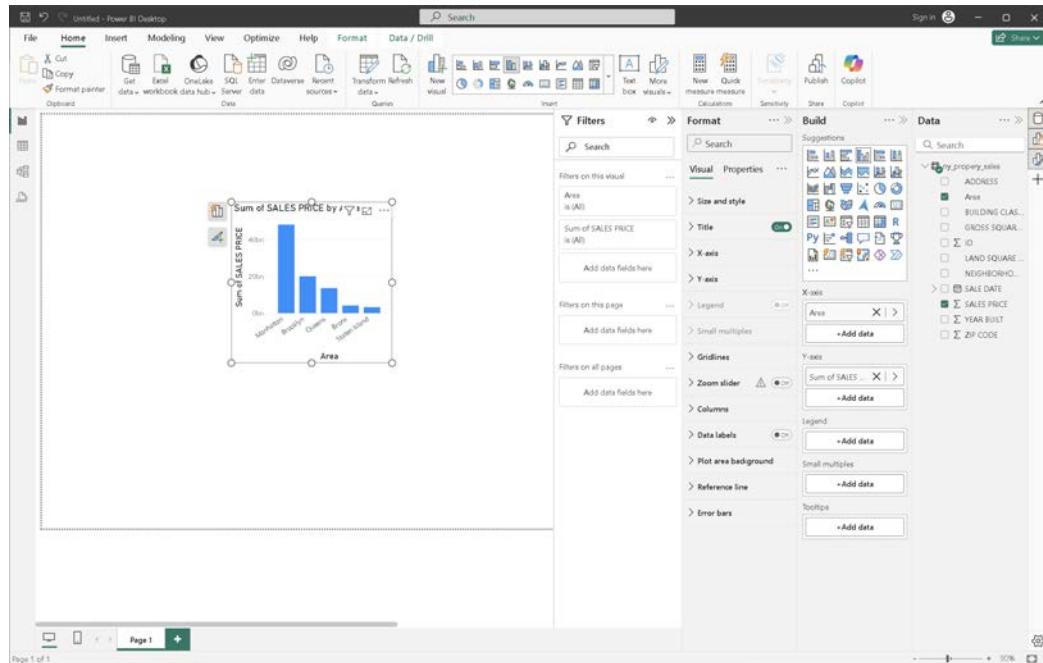
The screenshot shows the Power BI Desktop interface with the 'Data' view selected. The table is titled 'my_property_sales'. The columns are: ID, AREA, NEIGHBORHOOD, BUILDING CLASS CATEGORY, ADDRESS, ZIP CODE, LAND SQUARE FEET, GROSS SQUARE FEET, YEAR BUILT, SALE DATE, and SALES PRICE. The table contains approximately 150 rows of real estate sales data. The data includes various addresses in Manhattan, such as 100 Fifth Avenue, 720 Park Avenue, and 11 East 77th Street, across different building classes and years.

- **Model View:** Used to create and manage relationships between different tables in your dataset.

The screenshot shows the Power BI Desktop interface with the 'Model' view selected. It displays the relationships between tables: ADDRESS, AREA, BUILDING CLASS CATEGORY, LAND SQUARE FEET, NEIGHBORHOOD, SALE DATE, and SALES PRICE. A card for the 'my_property_sales' table is open, showing its schema. The card lists the columns: ADDRESS, AREA, BUILDING CLASS CATEGORY, LAND SQUARE FEET, NEIGHBORHOOD, SALE DATE, and SALES PRICE. The Power BI ribbon is visible at the top, showing options like File, Home, Help, Table tools, and Transform.

5. Creating Visualizations:

- To create a visualization, simply drag and drop a field (e.g., sales price) onto the canvas. Power BI will apply a default visualization, which you can change from the visualization pane.



This overview provides the foundational steps to navigate the Power BI interface and begin working with your data. As we proceed, we will delve deeper into each of these components to fully leverage Power BI's capabilities.

▼ Important Initial Settings

Before starting our first project, it's essential to configure some initial settings in Power BI to ensure that your interface matches mine. This alignment will help you follow along smoothly with the examples provided.

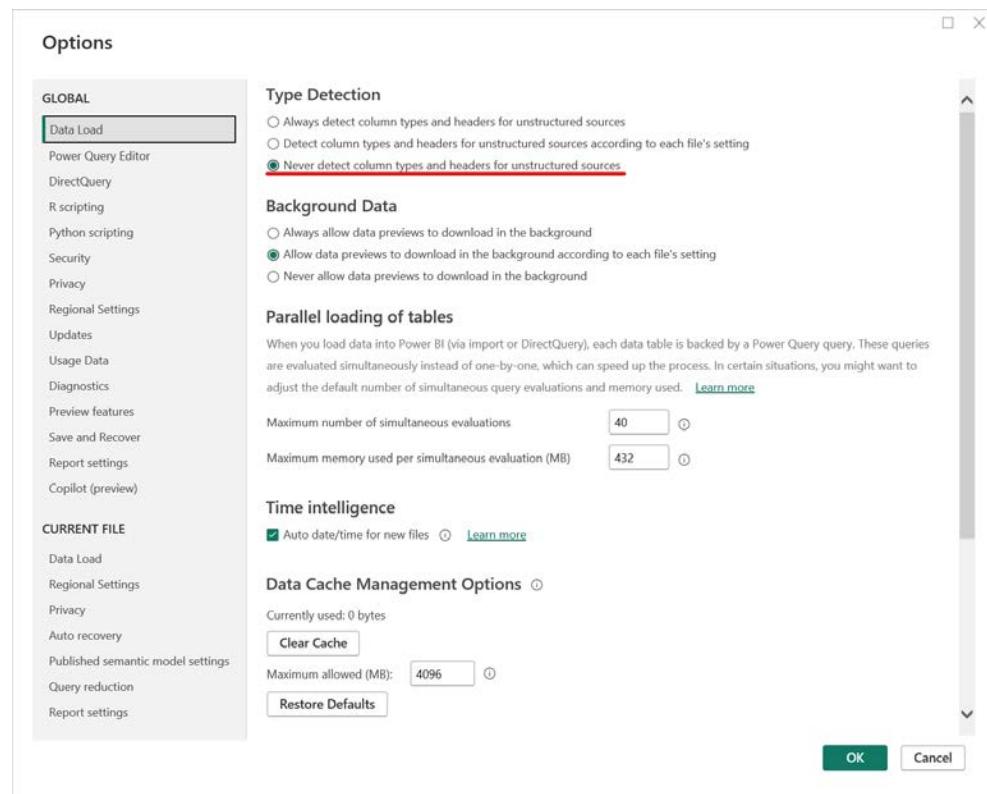
1. Creating a New Project File:

- Start by going to **File > New** to create a new project file. This file will serve as the foundation for our work.

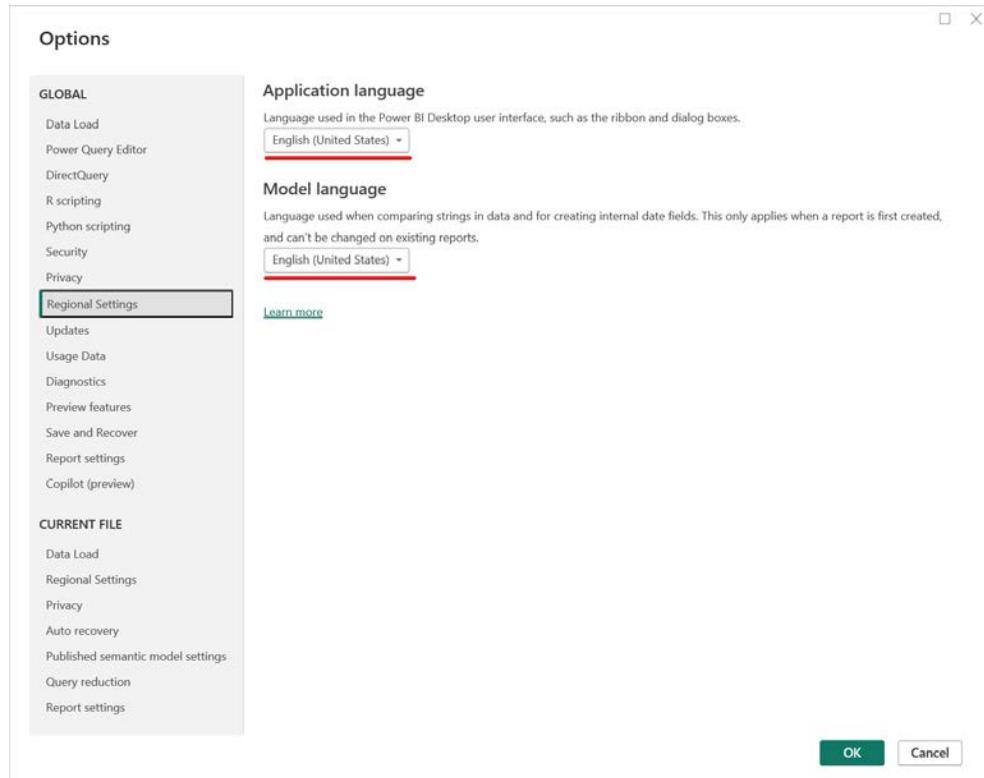
2. Global Settings:

- Go to **File > Options and Settings > Options**.
- In the **Global** section:

- **Data Load:** Set the type detection to "Never detect column types and headers for unstructured sources." This allows us to manually manage column types, ensuring consistency in the learning process.

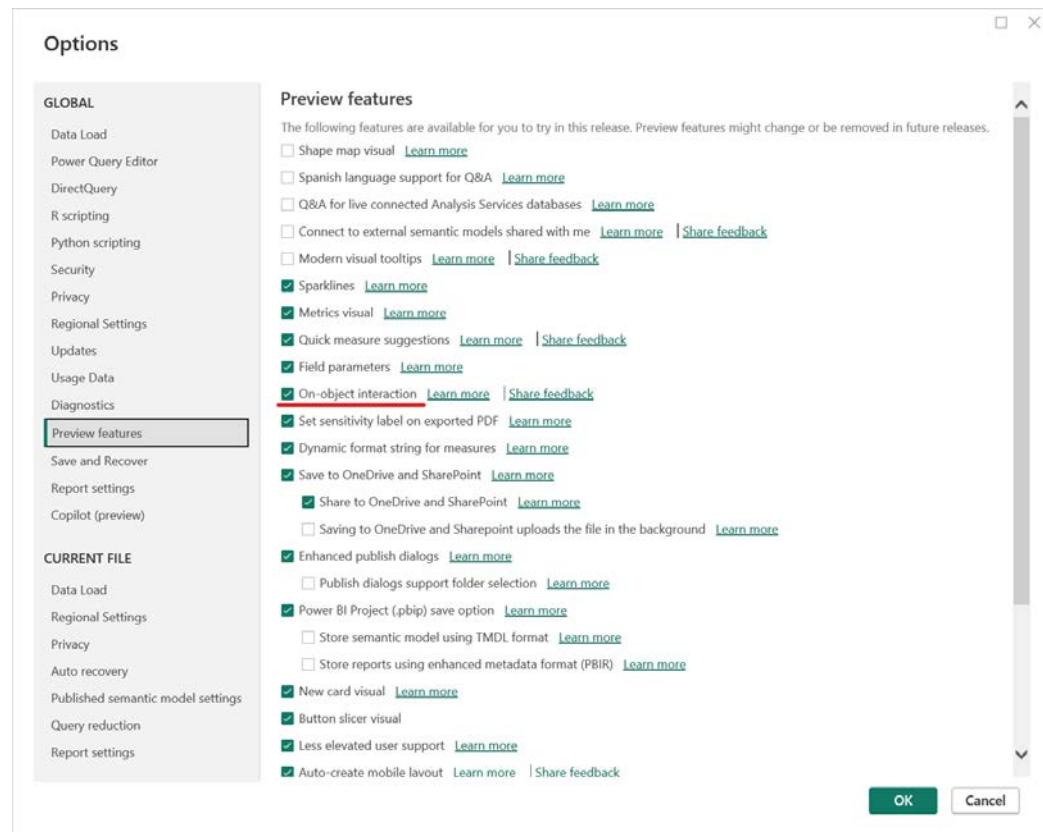


- **Regional Settings:** Ensure that both the **Application Language** and **Model Language** are set to "English (United States)." This is crucial for handling date fields and formatting.



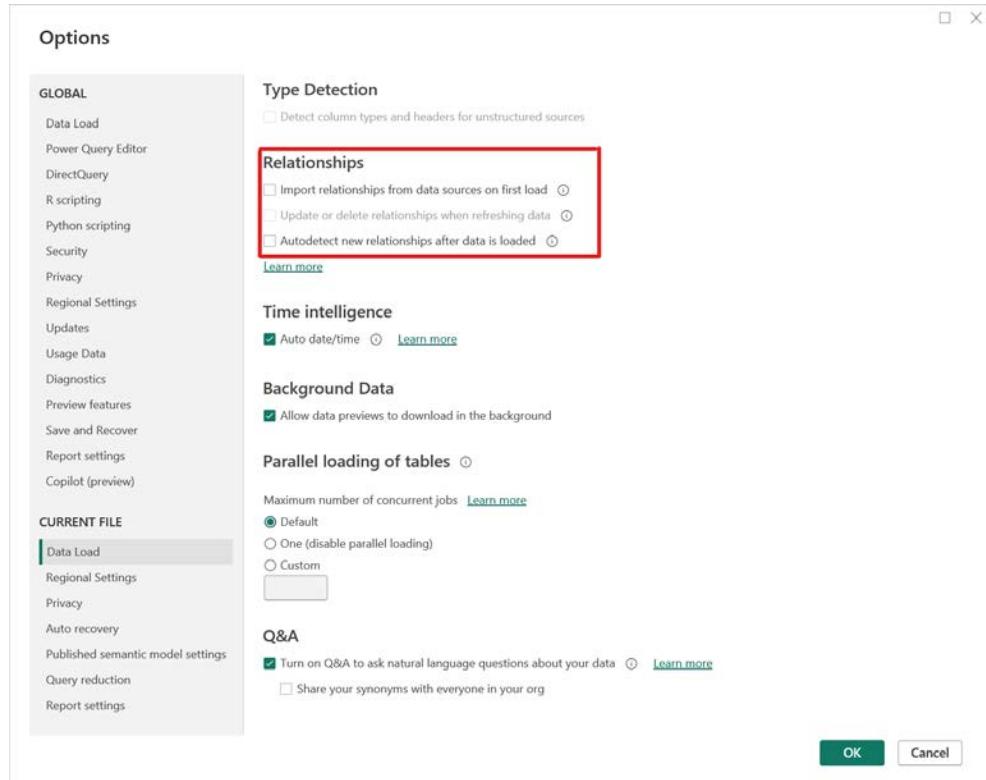
3. Preview Features:

- If available, enable the **On-Object Interaction** feature under **Preview Features**. If you don't see this option, it may already be a public feature.

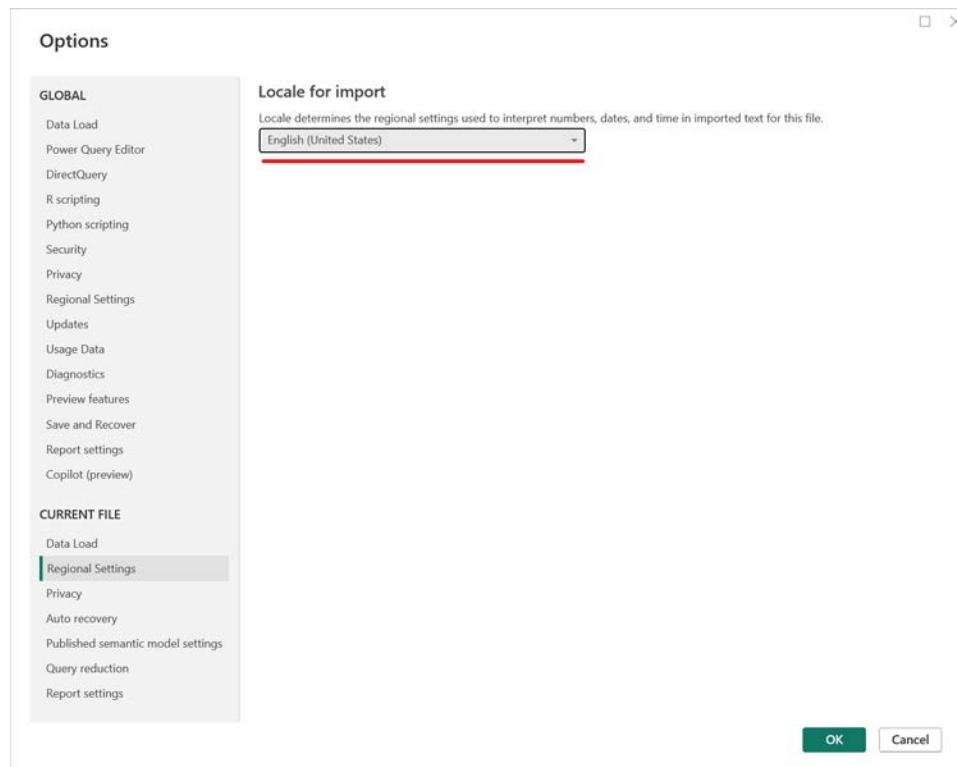


4. Current File Settings:

- In the **Current File** section:
 - **Data Load > Relationships:** Uncheck all options to prevent automatic relationship detection. This will allow us to manually create relationships, enhancing our understanding of how they work.



- **Regional Settings:** Set the locale to "English (United States)" to ensure consistent interpretation of numbers, dates, and time formats.



5. Saving and Reopening the Project File:

- Save your project file with a name like "Initial Project."
- Close Power BI, then reopen the saved file to apply all settings.

With these settings configured, you're ready to begin working on your first project. In the next section, we will introduce the project, examine the data we'll be using, and load it into our Power BI file.

▼ 2- Data Preparation, Modelling and Visualization

▼ Chapter 1

▼ Goals

So, today we are again focusing on our course project. And in this project we now have a couple of tasks and in the end we need to do a lot of things. So we want to extract values. That means we want to extract information from the data that has not been there before. We want to

split columns to structure the data so that we have all of the information in the right place and some other data transformations. And in the end, we are then also going to create some visualizations and this will be a super important part where we are going to learn about the different and the most important visualizations already, how we can customize them, and some tips how we can use them. So a very special topic is the pie chart. I know this is a super controversial topic and many people even advise that this should not be used at all, this pie chart. And therefore, I don't think that this is really true, but you really need to follow some really strict guidelines, I would say, to really not use the pie chart in a bad way. And therefore, this will be a really deep dive into the pie chart also, where you learn how you can use the pie chart in the right way. And, by the end of today, we are able to create those visualizations and the important data transformation tasks.

▼ Our Project Data

[Our data project.pdf](#)

Let's now take a look at our first course project. In this project, we will cover everything needed in Power BI from start to finish. You will take on the role of a data analyst working for an E-commerce company, tasked with creating a comprehensive sales report from start to finish.

We will begin by working with a raw data file, which contains messy sales data from an E-commerce store. This is beneficial because it gives us the opportunity to practice the necessary tools for data transformation. The goal is to create a detailed report that visualizes all the key metrics, such as sales categories, promotion performance, sales trends over time, and more.

Although this is just a sample project, we will develop our own complete report. To visualize all these different metrics, we will need to combine multiple data sources. We will then create a data model by establishing relationships between the various tables, enabling us to produce a comprehensive report.

In every realistic project, there are three essential steps that we will follow here:

1. **Data Import:** First, we import the data into Power BI.
2. **Data Transformation:** Next, we clean and transform the data as needed.
3. **Data Modeling and Visualization:** After that, we connect the tables by creating relationships to build the data model, and finally, we visualize the data to generate our report.

Once the report is complete, we will also publish it to the cloud.

▼ Getting Data

[sales2017_raw.csv](#)

Let's first take a look at how we can connect data with Power BI. This is, of course, the first and very important step. You can also download this data from the resources section of this course. We will start with the first file that we need to load, which is called sales2017_raw.

Before we start restructuring and cleaning the data, let's first get an overview of what we have. The dataset includes several key columns:

- **Order ID:** Each row represents a single order.
- **Product ID:** This can be linked to a product table later, allowing us to pull in additional product details by creating a relationship based on this column.
- **Store ID:** Similar to Product ID, this can be used to connect with a store table.
- **Order Date and Delivery Date:** These columns include date information, though you might notice the dates appear in different formats.
- **Revenue and Price:** Financial data is provided, but keep in mind these are sample figures, so they don't need to be taken literally.

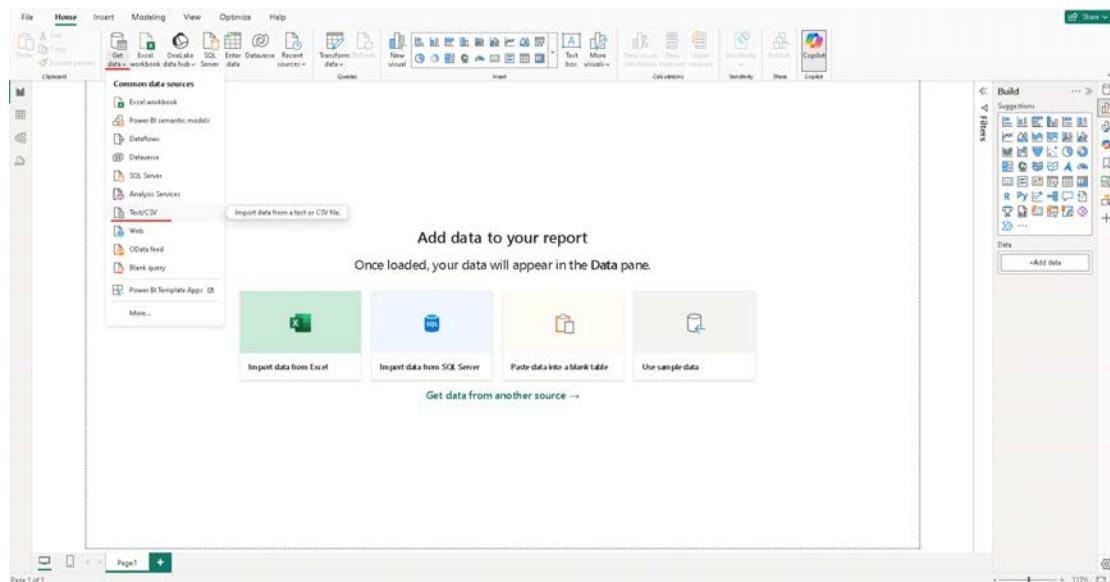
- **Promotions:** Details about the types of promotions related to each order are also included.

As you can see, this is a very messy sales dataset, which will be the first data we load. There is quite a bit of work to do here we need to restructure and clean the data. But first, let's get an overview of the dataset.

We can see that there's an `order_id`, meaning each row represents one order. In the second column, we have the `product_id`. This column allows us to connect the sales data with a product table to retrieve additional information about the products by creating a relationship using the `product_id` column. This concept can also be applied to the `store_id`.

Additionally, there's information about the `order_date`, although the date format may vary in some cases. We also have columns for revenue and price. Keep in mind that the numbers in this dataset may not always be exact, as it's just sample data. However, overall, this is a realistic project, and we will build a realistic report from it.

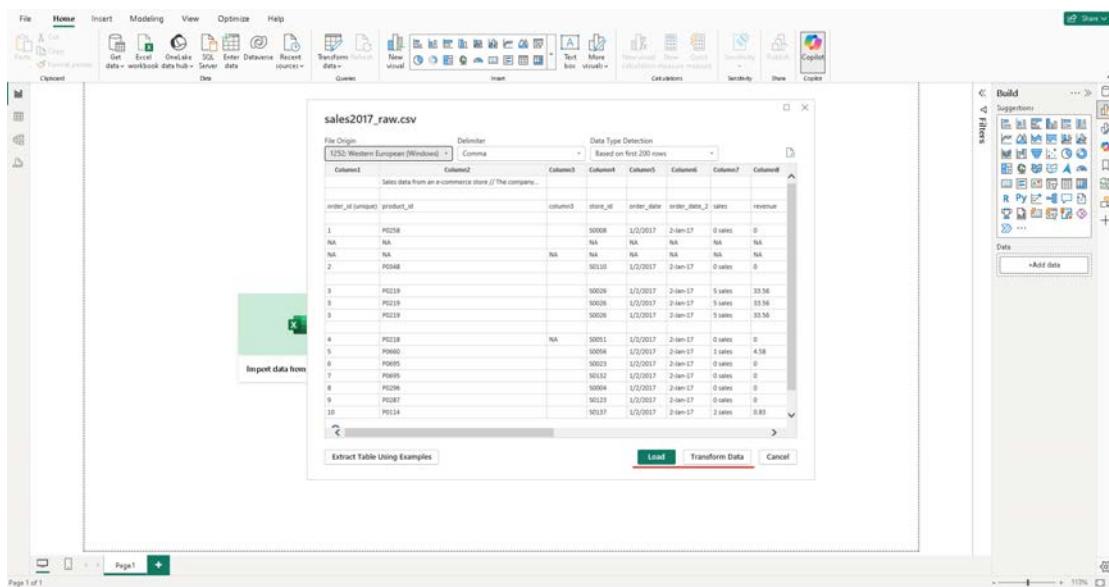
We also have data about the types of promotions used and information regarding the delivery date following the order date.



Now, let's load this data into Power BI. To import the data, navigate to the Home ribbon and click Get Data. By clicking the upper part of the Get Data button, you'll open a menu with different data sources categorized by type. You can find CSV files under the File section, or you can type "CSV" in the search box for quicker access.

Instead of browsing through all available connectors, you can use a shortcut. If you click on the lower part of the button (text instead of the icon), you'll see the most commonly used data sources, including text/CSV.

To load the CSV file, select the appropriate data source, navigate to the file's location on your local machine, select it, and click Open. You'll then see a preview of the data.



At this point, you have two options: you can either load the data directly into the model, or you can click on Transform Data. This will open the Query Editor, where you can make changes to clean and restructure the data.

When you load a CSV file into Power BI, you'll notice several important options appear at the top of the preview window, such as File Origin, Delimiter, Data Type Detection, and at the bottom, Extract Table Using Examples. Let's break down what each of these options means:

Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8
	Sales data from an e-commerce store // The company...						
order_id (unique)	product_id	column3	store_id	order_date	order_date_2	sales	revenue
1	P0258	S0008	1/2/2017	2-Jan-17	0 sales	0	
NA	NA	NA	NA	NA	NA	NA	NA
NA	NA	NA	NA	NA	NA	NA	NA
2	P0348	S0110	1/2/2017	2-Jan-17	0 sales	0	
3	P0219	S0026	1/2/2017	2-Jan-17	5 sales	33.56	
3	P0219	S0026	1/2/2017	2-Jan-17	5 sales	33.56	
3	P0219	S0026	1/2/2017	2-Jan-17	5 sales	33.56	
4	P0218	NA	S0051	1/2/2017	2-Jan-17	0 sales	0
5	P0660		S0056	1/2/2017	2-Jan-17	1 sales	4.58
6	P0695		S0023	1/2/2017	2-Jan-17	0 sales	0
7	P0695		S0132	1/2/2017	2-Jan-17	0 sales	0
8	P0296		S0004	1/2/2017	2-Jan-17	0 sales	0
9	P0287		S0123	1/2/2017	2-Jan-17	0 sales	0
10	P0114		S0137	1/2/2017	2-Jan-17	2 sales	0.83

Extract Table Using Examples **Load** **Transform Data** **Cancel**

File Origin:

sales2017_raw.csv

File Origin	Delimiter	Data Type Detection					
1252: Western European (Windows)	Comma	Based on first 200 rows					
874: Thai (Windows)		Column3	Column4	Column5	Column6	Column7	Column8
857: Turkish (DOS)		Company...					
28599: Turkish (ISO)							
10081: Turkish (Mac)							
1254: Turkish (Windows)							
10017: Ukrainian (Mac)							
1200: Unicode							
1201: Unicode (Big-Endian)							
12001: Unicode (UTF-32 Big-Endian)							
12000: Unicode (UTF-32)							
65000: Unicode (UTF-7)							
65001: Unicode (UTF-8)							
20127: US-ASCII							
1258: Vietnamese (Windows)							
20005: Wang Taiwan							
850: Western European (DOS)							
20105: Western European (IA5)							
28591: Western European (ISO)							
10000: Western European (Mac)							
1252: Western European (Windows)							
9	P0287						
10	P0114						

Extract Table Using Examples Load Transform Data Cancel

This option allows you to specify the encoding of the CSV file. For example, you might see options like "65001: Unicode (UTF-8)" or other character encodings depending on how the file was created. This is important to ensure that characters and symbols are correctly displayed when loading the data. If you notice any unusual characters, changing the file origin can help resolve the issue.

Delimiter:

sales2017_raw.csv

File Origin	Delimiter	Data Type Detection
1252: Western European (Windows)	Comma	Based on first 200 rows
Column1	Colon	Column4
	Comma	Column5
Sales data from an e-comm	Equals Sign	Column6
order_id (unique)	Semicolon	Column7
product_id	Space	Column8
1	Tab	store_id
P0258	--Custom--	order_date
NA	--Fixed Width--	order_date_2
NA	NA	sales
2	NA	revenue
P0348		
3		
P0219		S0008
3		1/2/2017
P0219		2-Jan-17
3		0 sales
P0219		0
4		
P0218	NA	NA
5		S0051
P0660		1/2/2017
6		2-Jan-17
P0695		1 sales
7		4.58
P0695		NA
8		S0023
P0296		1/2/2017
9		2-Jan-17
P0287		0 sales
10		0
P0114		NA
		S0132
		1/2/2017
		2-Jan-17
		0 sales
		0
		S0004
		1/2/2017
		2-Jan-17
		0 sales
		0
		S0123
		1/2/2017
		2-Jan-17
		0 sales
		0
		S0137
		1/2/2017
		2-Jan-17
		2 sales
		0.83

Extract Table Using Examples Load Transform Data Cancel

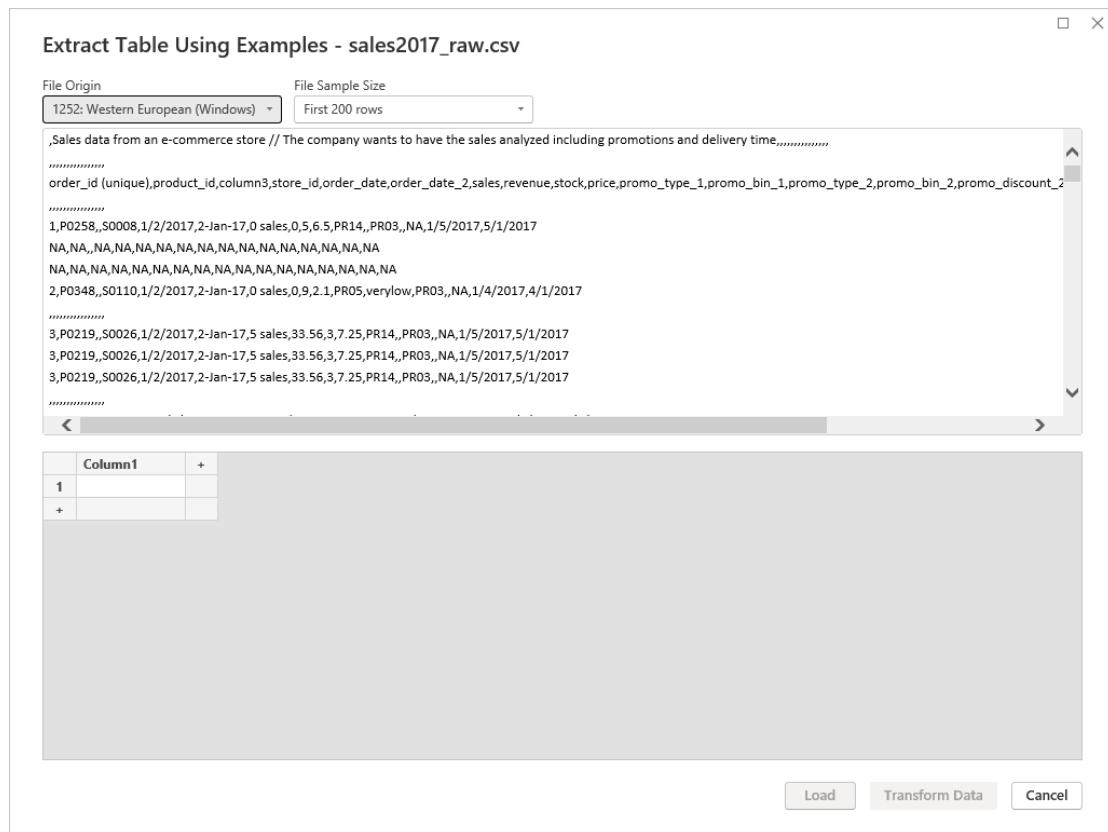
The delimiter is the character that separates values within the CSV file. Common delimiters include commas (","), semicolons (";"), tabs, or spaces. Power BI automatically tries to detect the correct delimiter, but you can manually select the appropriate one from the dropdown if needed. Choosing the right delimiter ensures that the data is split into the correct columns.

Data Type Detection:

The screenshot shows the 'Get Data' screen in Power BI. A dropdown menu labeled 'Data Type Detection' is open, with the option 'Based on first 200 rows' selected. Other options include 'Based on first 2000 rows', 'Based on entire dataset', and 'Do not detect data types'. Below the dropdown is a preview of the 'sales2017_raw.csv' data. The preview shows columns: order_id (unique), product_id, column3, store_id, order_date, order_date_2, sales, and revenue. The data consists of 10 rows of sales information. At the bottom of the screen are three buttons: 'Extract Table Using Examples' (disabled), 'Load' (highlighted in green), and 'Transform Data'.

This option controls how Power BI detects and assigns data types (e.g., text, date, number) to each column. The default setting, "Based on the first 200 rows," analyzes the first 200 rows of your data to determine the data type of each column. You can also choose "Based on the entire dataset" if you want a more thorough analysis, or "Do not detect data types" if you prefer to manually assign data types later.

Extract Table Using Examples:



This feature allows you to create a new table by providing examples of the data you want to extract. Based on the examples you provide, Power BI tries to understand the pattern and extracts similar data automatically. This is useful when you need to filter or reshape data without writing complex code, making it a more user-friendly way to extract specific rows or columns from the dataset.

▼ Working with Query Editor

Let's take a closer look at data transformation using the Power Query Editor, a key tool within Power BI for cleaning, restructuring, and transforming data.

Power Query Editor Overview:

- **Interface:** At the top of the screen, you'll see a variety of tools like "Keep Rows," "Remove Rows," and "Change Data Type." These are

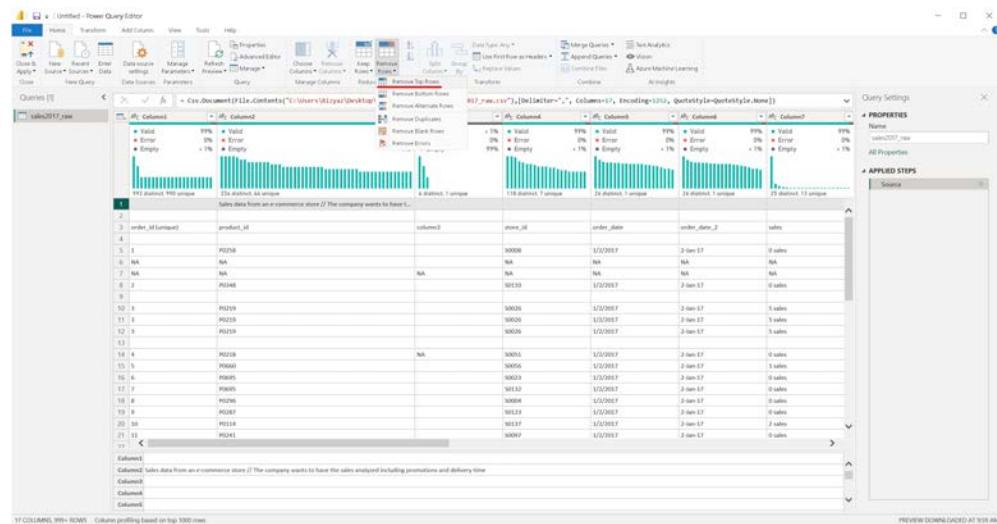
commonly used tools found under the **Home** ribbon. Other ribbons like **Transform** and **Add Columns** offer additional functionalities, which we'll explore later.

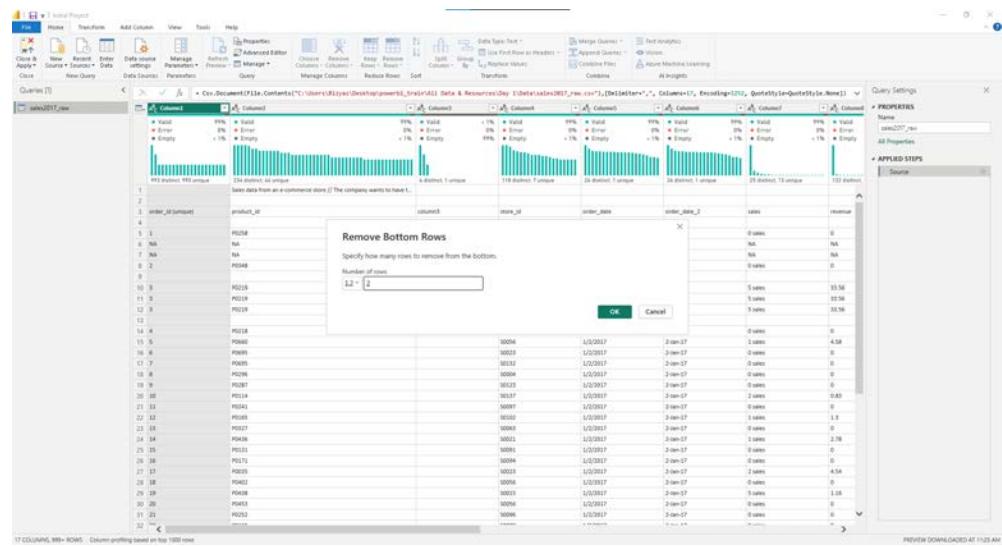
- **Data Preview:** The central part of the screen shows a preview of your data as it currently stands, reflecting any transformations applied.

Basic Data Transformation Steps:

1. Removing Unnecessary Rows:

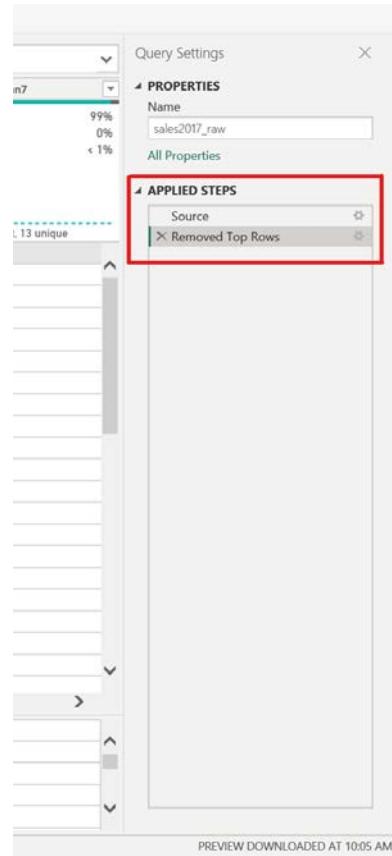
- In the data preview, if you see unnecessary rows (e.g., the first two rows), you can remove them.
- Go to **Remove Rows > Remove Top Rows**. Specify the number of rows to remove (e.g., 2), and click **OK**. The preview will update to reflect these changes.





2. Understanding Applied Steps:

- On the right side of the screen, the **Applied Steps** pane tracks each transformation you apply. You can click on any step to view the data at that stage.
- If you need to undo a step, click the "X" next to the step to remove it. To redo a step, you'll need to reapply the transformation.

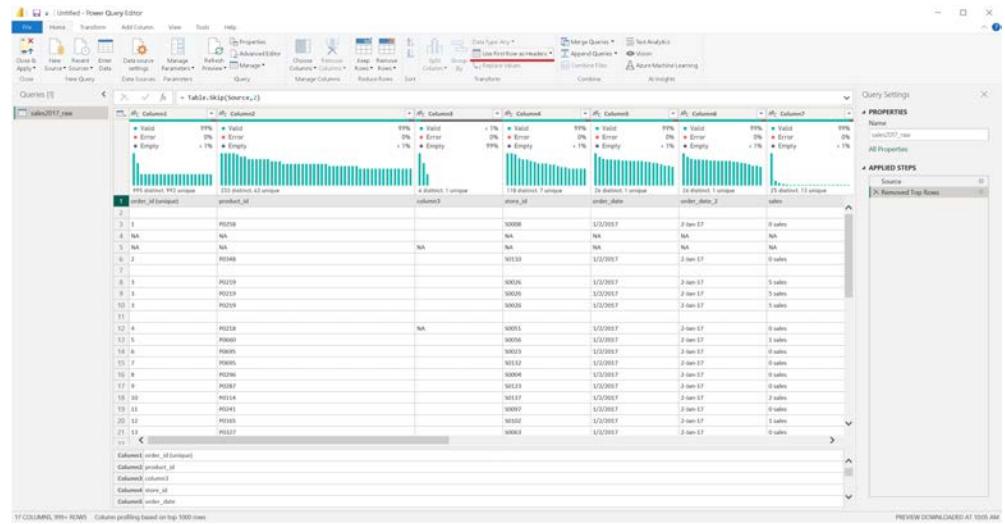


3. Modifying Steps:

- If you make a mistake (e.g., removing the wrong number of rows), you can modify the step. Click the gear icon next to the step to adjust its settings, such as changing the number of rows to remove.

4. Using First Row as Headers:

- If your first row contains headers, you can convert it into column names by selecting **Use First Row as Headers** under the Home ribbon. This moves the first row data into the header position, organizing your data correctly.

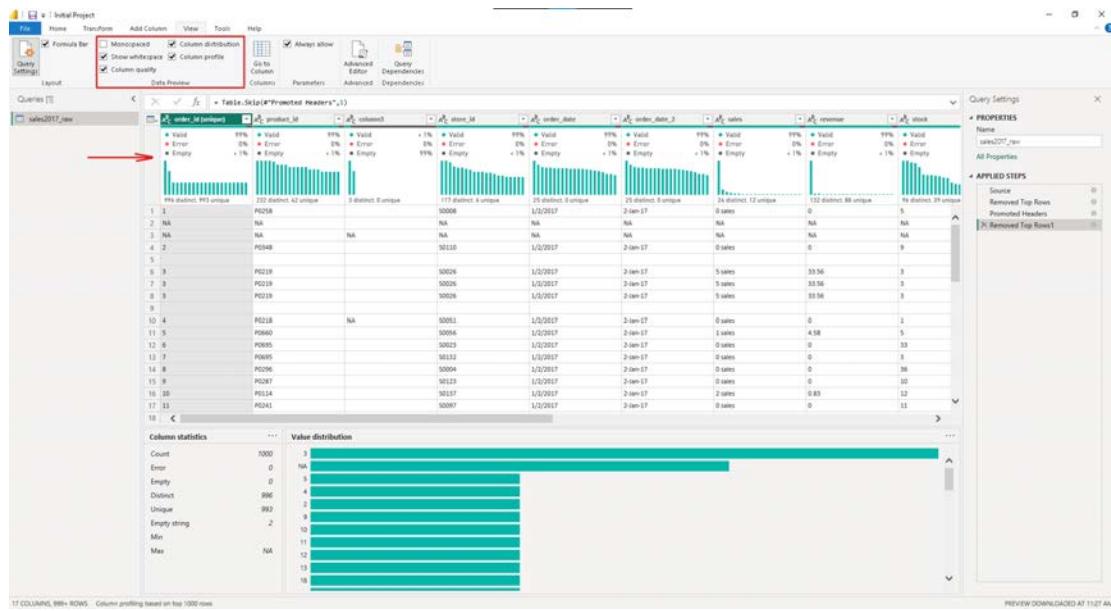


Finalizing Transformations:

- Once you've completed your data transformations, click **Close & Apply** to save your changes and load the transformed data into the model. You'll see a loading window, and the data will then appear on the right side of the Power BI interface, ready for visualization.

Tip

When working with data in Power BI, you'll often see several key metrics or indicators displayed beneath each column, which help provide insights into the quality and characteristics of your data. You can turn these features on as shown.

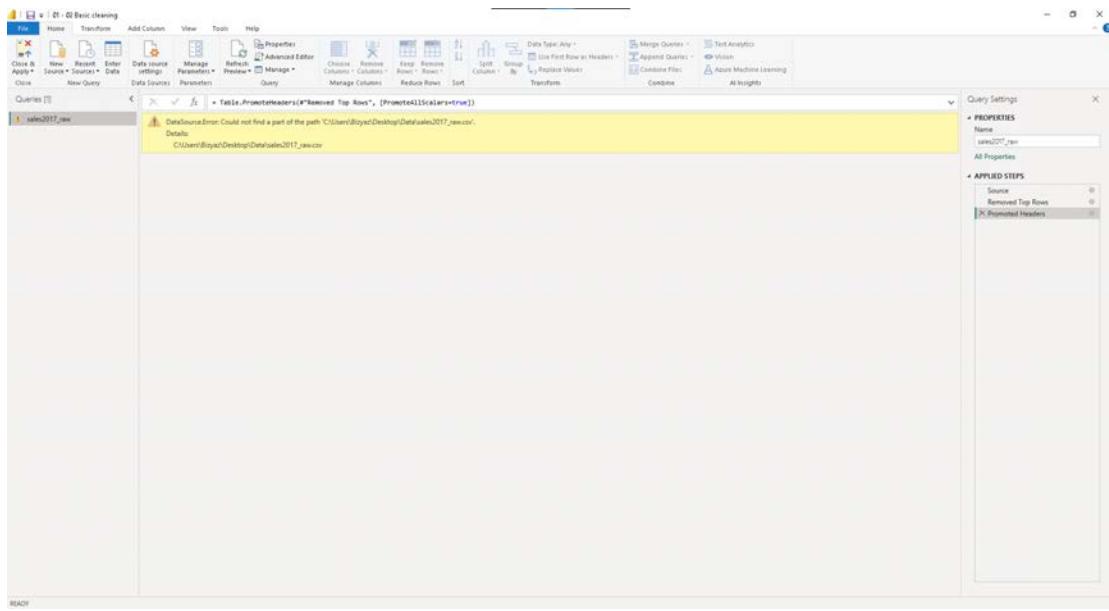


Before continuing, make sure to save your project. In the next section, we'll discuss how to work with the attached project files provided throughout the course.

▼ Using Attached Project Files

Throughout the course, I provide the report we are working on as a resource. You can use these files optionally; most of the time, you might prefer to work along in your own file, which is a more common scenario. However, there may be instances where you want to use my file. For example, if you're encountering issues, need to jump into a specific part of the project, or if you didn't follow along with previous videos and want to start immediately from where the video begins.

You can easily download the file and open it. For instance, I've already opened the Basic Cleaning file that we worked on in the previous lecture. While the data appears to load without any issues, upon opening the Query Editor, you might encounter an error message:
"Could not find part of the path..."



This error is pointing to the data file that was loaded into the project file, and it is linked to a path on my local machine. Since this file was created on my machine, the path is unique to my environment, and therefore, the file cannot be found on your local system. This issue is unavoidable, but it can be solved quite easily.

This type of error can happen in many situations, so it's important to understand how to resolve it. The error occurs because Power BI cannot locate the data file on your machine (the file path might have changed, or the file may not be present anymore).

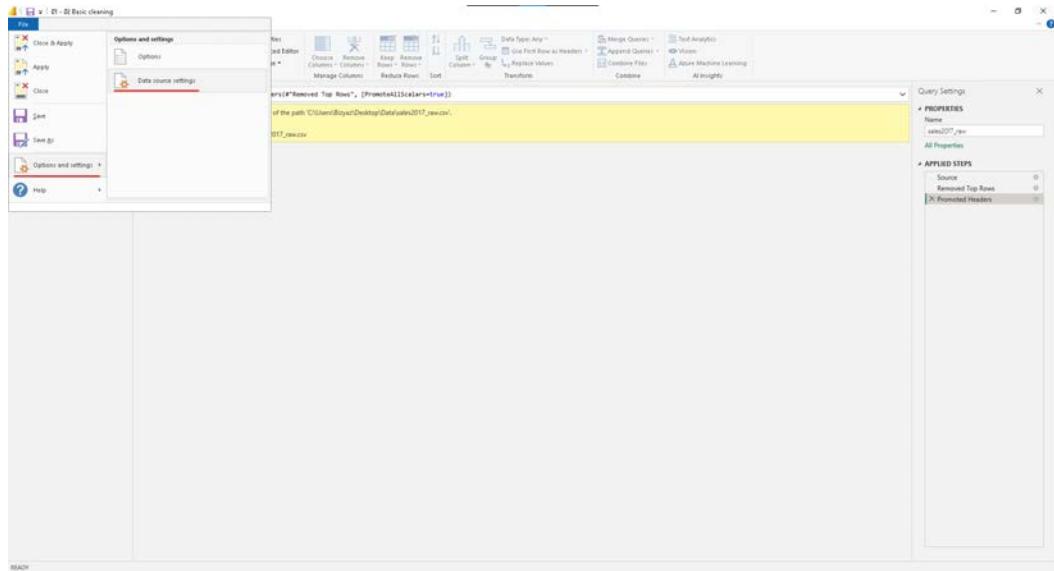
Here's how to solve the issue:

1. Close the Query Editor

You can resolve the issue by updating the data source file path in the report view.

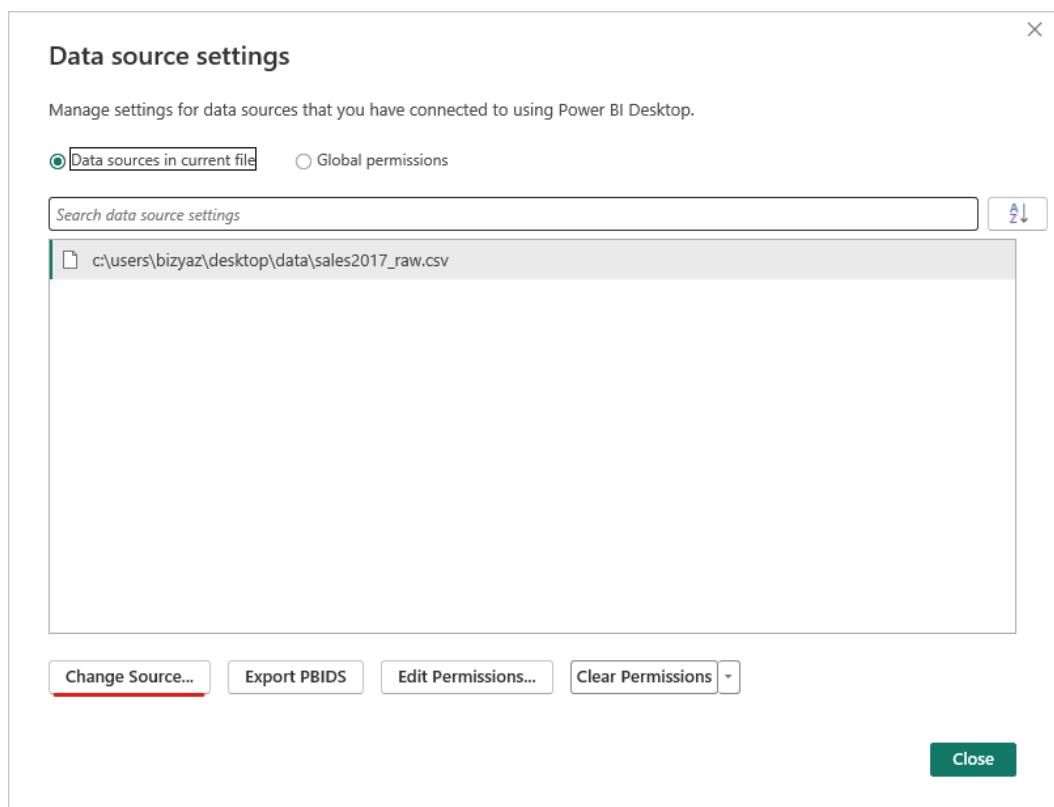
2. Open Data Source Settings

In the Home tab, go to **Transform Data** and then click on **Data Source Settings**. This opens a window showing all the data sources and paths loaded into your project.



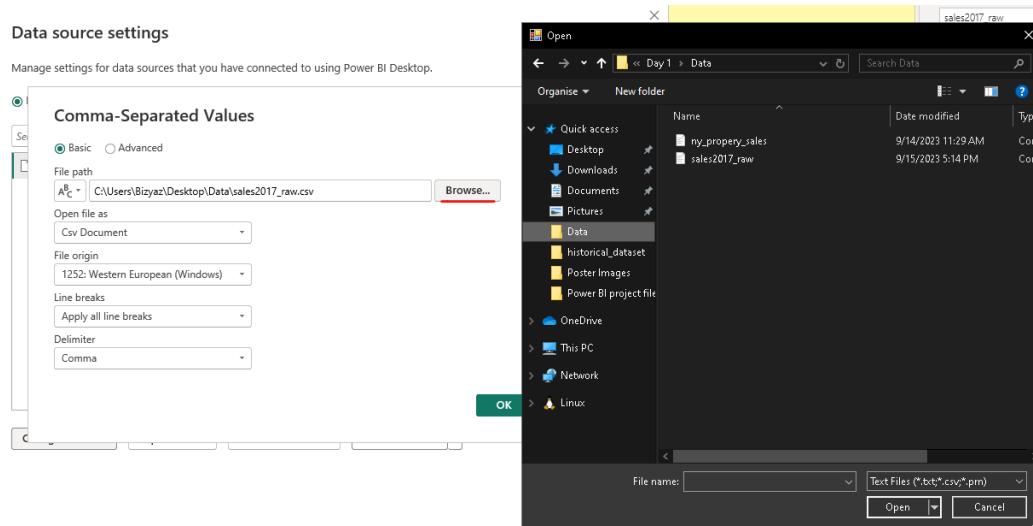
3. Change Source Path

To fix the path, click on **Change Source**. You will see a window where you can edit the file path. You can either manually update the file path or browse for the correct file location on your system.



4. Browse for File

If you prefer, you can use the **Browse** button to navigate to the folder where the file is saved, select the file, and click **Open**.



5. Apply Changes

Once the new path is entered, click **OK** and close the data source settings. Now, if you return to **Transform Data**, the error should no longer be present. Click **Close and Apply** to apply the changes, as updating the data source is considered a transformation that needs to be applied.

Now that you know how to fix this common issue with attached project files, it will be easier to continue working without interruptions.

▼ Rows & Columns

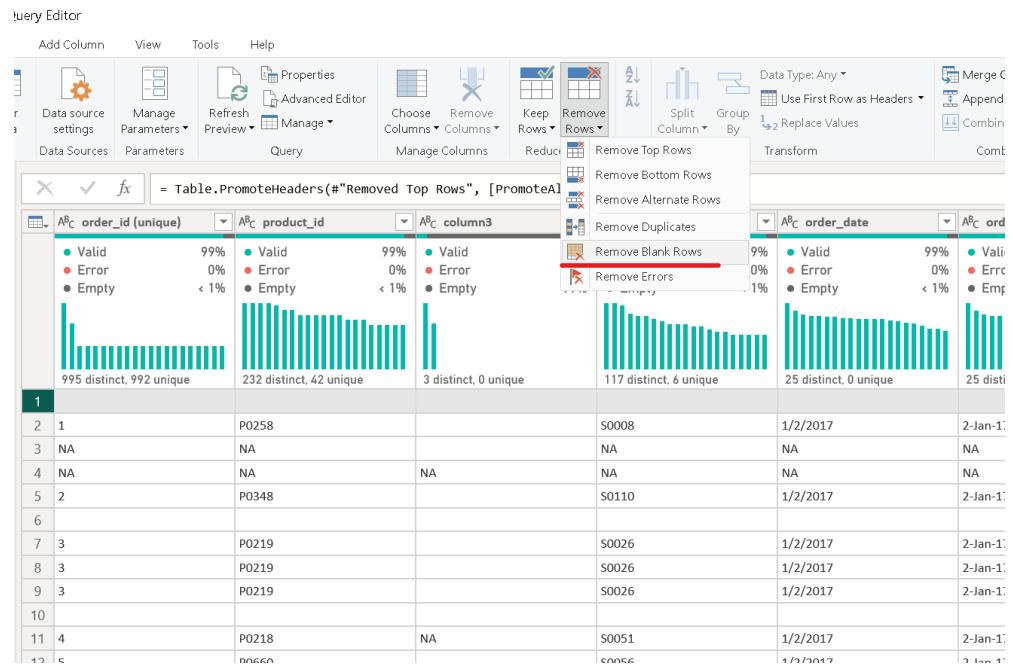
Let's explore how to edit rows and columns in Power BI, focusing on addressing common data issues such as blank rows, NA values, and unnecessary columns.

Editing Rows:

1. Removing Blank Rows:

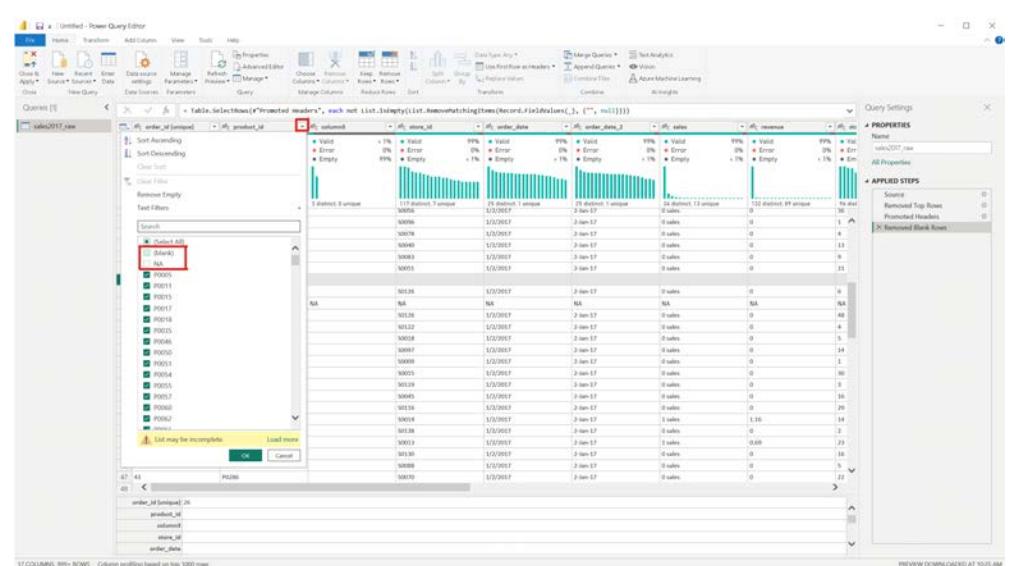
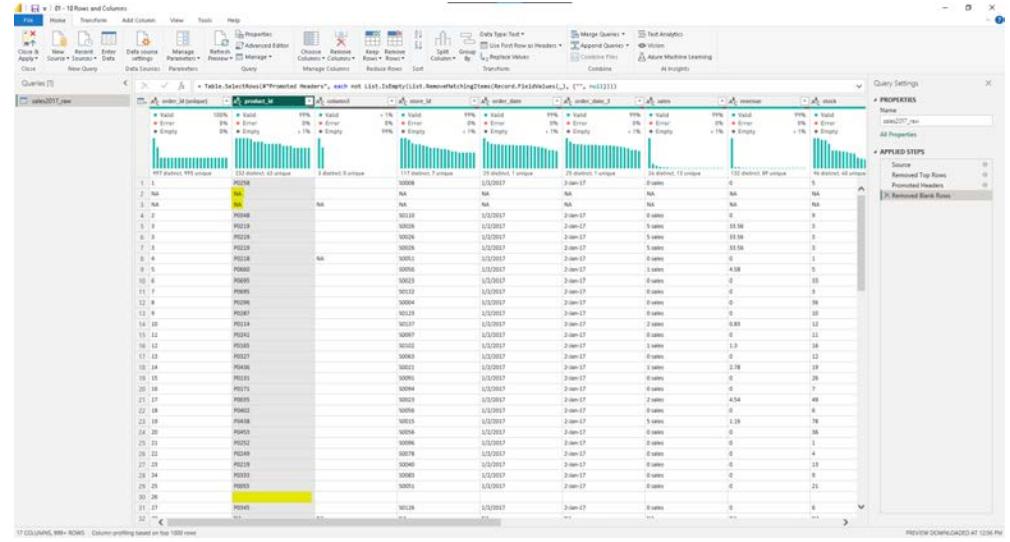
- If your dataset contains completely blank rows, they need to be filtered out. Under the **Home** ribbon, use the **Remove Rows** tool

and select **Remove Blank Rows**. This will eliminate all rows without any data.



2. Filtering Specific Rows:

- Sometimes, rows may appear blank except for one or two columns. To remove these, apply a filter to specific columns. For example, you can filter out rows where the **Product ID** is blank by clicking the filter icon next to the column header and selecting the appropriate filter options. This method can also be used to remove rows with NA values.



3. Removing Duplicates:

- Duplicates can skew your data analysis. Use the **Remove Duplicates** option under **Remove Rows**. Make sure no specific column is selected when applying this step, as it will otherwise only remove duplicates within that column. To remove duplicates across all columns, ensure that no columns are highlighted before applying the step.

Note ↪

Note that the row is not selected. Perform the operation with the ID column selected.

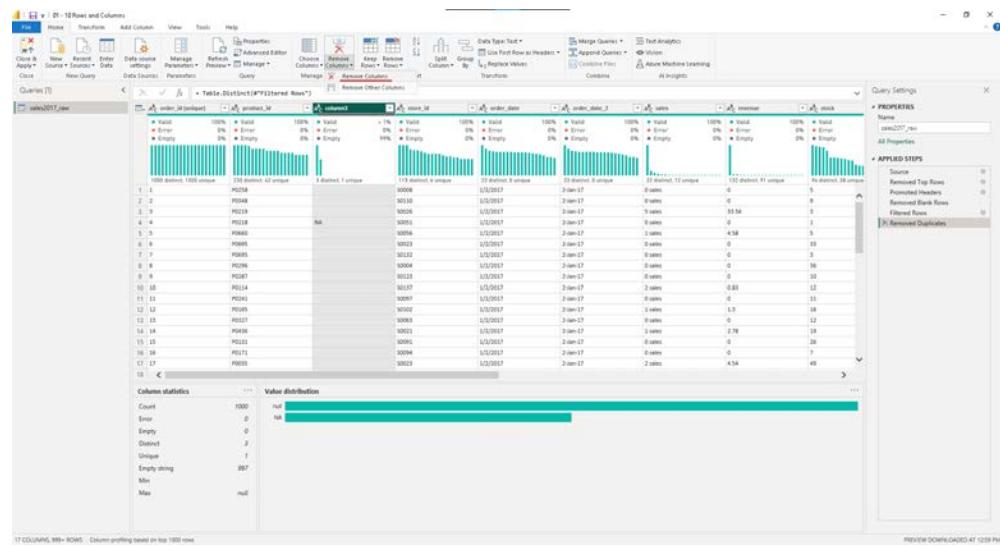
The screenshot shows the Power Query Editor interface with the 'sales2017.xlsx' file open. The 'Applied Steps' pane on the right lists the step 'Removed Blank Rows'. The main table view shows a dataset with columns: product_id, sales_id, order_date, sales, and revenue. A red box highlights the first two rows of the table, which are not selected. The status bar at the bottom indicates '17 COLUMNS, 999+ ROWS'.

This screenshot is similar to the previous one, but the 'Applied Steps' pane now lists the step 'Filtered Rows'. The table view shows the same dataset, but the first two rows are now highlighted with a blue selection bar, indicating they are the current focus. The status bar at the bottom indicates '17 COLUMNS, 999+ ROWS'.

Editing Columns:

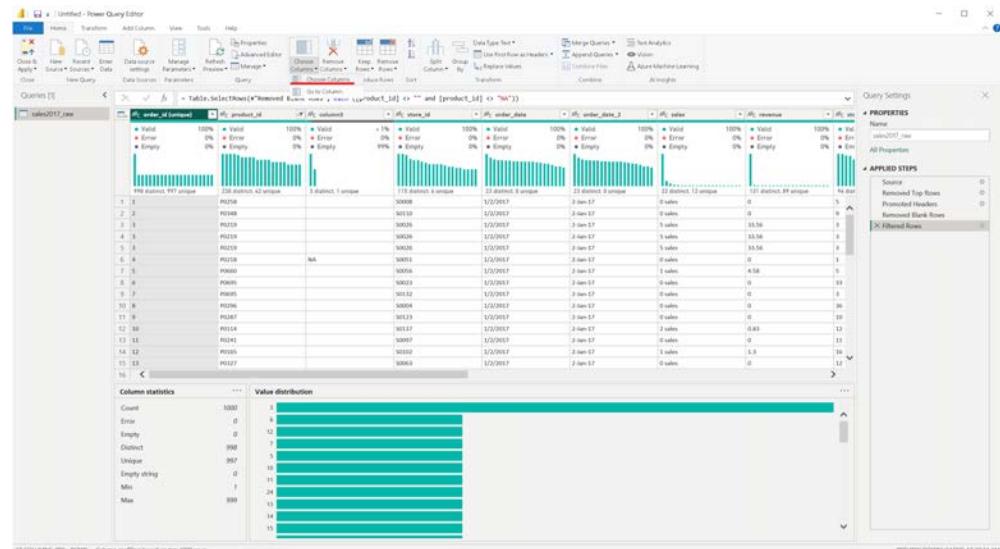
1. Removing Unnecessary Columns:

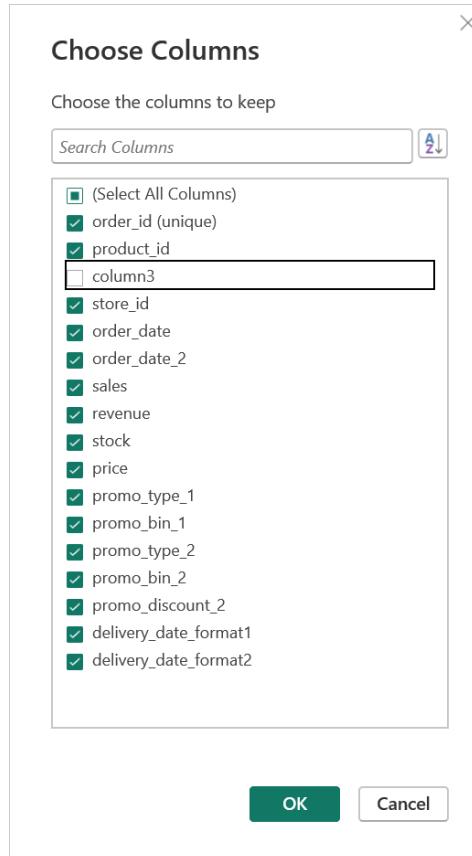
- If there's a column that adds no value to your analysis (e.g., it's empty or irrelevant), select it and use the **Remove Columns** option. This will delete the column from your dataset.



2. Choosing Specific Columns:

- If your dataset contains many columns and you want to remove several, use the **Choose Columns** option. This allows you to quickly select or deselect multiple columns. You can always go back and modify your selection later by clicking the gear icon next to the step in the **Applied Steps** pane.





With these steps, you've now cleaned up much of your data by removing unwanted rows and columns.

▼ Data Types

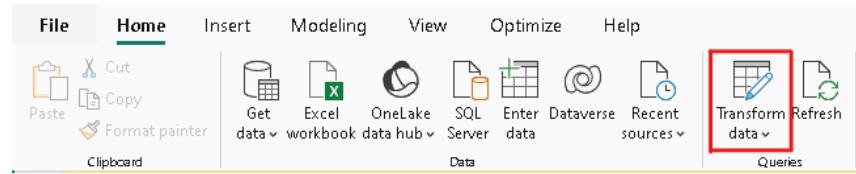
▼ Detecting Data Types

In Power BI, detecting and managing the data types of your columns is a crucial step in ensuring that the data is processed correctly and efficiently. Data types define the kind of data that each column holds, and setting them properly helps in transforming, analyzing, and visualizing the data accurately.

Accessing the "Transform Data" Section

1. Opening Power Query Editor

To begin working with your data, you need to go to the **Home** tab in Power BI Desktop and click on **Transform Data**. This opens the **Power Query Editor**, where you can view and transform your data before loading it into the Power BI model.



Reviewing Data Types

In the Power Query Editor, each column will have a small icon next to its name, representing its detected data type:

Icon	Description
1.2	Decimal Number
\$	Fixed decimal number
1 ² 3	Whole Number
%	Percentage
⌚	Date/Time
📅	Date
🕒	Time
🌐	Date/Time/Timezone
🕒⌚	Duration
Text	
True/False	
Binary	
Using Locale...	

1. Decimal Number

- Description:** The **Decimal Number** data type is used to store numbers with decimal points. It supports floating-point numbers and is ideal for precise calculations.
- Common Use Cases:** Prices, percentages, measurements, etc.
- Example:** `12.5`, `200.75`, `3.14159`

Icon: Displayed as **1.2**.

2. Fixed Decimal Number

- **Description:** The **Fixed Decimal Number** (also known as Currency) type is used for financial data that requires a fixed number of decimal places, typically two.
- **Common Use Cases:** Currency values, financial figures.
- **Example:** `12.34`, `$200.00`, `150.99`

Icon: Displayed as a **currency symbol** like \$ or £.

3. Whole Number

- **Description:** The **Whole Number** data type represents integer values without any decimal points. These are used for numbers that do not require fractional parts.
- **Common Use Cases:** Counts, quantities, IDs, etc.
- **Example:** `5`, `200`, `1050`

Icon: This is displayed as **123**.

4. Percentage

- **Description:** The **Percentage** data type formats numbers as percentages. It is generally used for ratios and proportion-based calculations.
- **Common Use Cases:** Discount rates, growth percentages, etc.
- **Example:** `25%`, `0.25`

Icon: Displayed as a **percentage symbol** (%).

5. Date/Time

- **Description:** The **Date/Time** data type stores both the date and the time (down to seconds). This allows you to track specific timestamps along with the date.
- **Common Use Cases:** Transaction timestamps, event times, etc.

- **Example:** `2024-10-17 10:30:00 AM`

Icon: Displayed as a **clock** along with a calendar.

6. Date

- **Description:** The **Date** data type stores calendar dates. It is used for tracking daily data points and time-related data.
- **Common Use Cases:** Birth dates, order dates, deadlines, etc.
- **Example:** `2024-10-17`, `2022-03-01`

Icon: Displayed as a **calendar** symbol.

7. Time

- **Description:** The **Time** data type is for storing time values without the associated date. It captures time in hours, minutes, and seconds.
- **Common Use Cases:** Opening/closing times, durations, etc.
- **Example:** `10:30:00 AM`, `15:45:00`

Icon: Displayed as a **clock**.

8. Date/Time/Zone

- **Description:** This data type stores date and time information, including the time zone in which the data was captured. It is useful for global operations where data spans multiple time zones.
- **Common Use Cases:** International meetings, worldwide transaction logs.
- **Example:** `2024-10-17 10:30:00 AM +03:00`

Icon: Calendar and clock with a **zone offset**.

9. Duration

- **Description:** The **Duration** data type represents a period of time. This is used to calculate time intervals between two dates or times.

- **Common Use Cases:** Time between two events, project durations.

- **Example:** `5 days`, `3 hours`, `45 minutes`

Icon: Displayed as a **clock** icon with an arrow.

10. Text (String)

- **Description:** The **Text** data type is used to store alphanumeric characters. This can include letters, numbers, and special characters.
- **Common Use Cases:** Names, product codes, addresses, etc.
- **Example:** `"John Doe"`, `"P001"`, `"New York"`

Icon: The data type icon is displayed as **ABC** in Power Query.

11. True/False (Boolean)

- **Description:** The **True/False** data type (Boolean) stores logical values, typically used in conditional logic or filters. The possible values are either `True` or `False`.
- **Common Use Cases:** Flags, binary choices, active/inactive status.
- **Example:** `True`, `False`

Icon: Displayed as a **checkbox**.

12. Binary

- **Description:** The **Binary** data type is used for binary objects, such as images or files, that cannot be easily represented as text or numbers.
- **Common Use Cases:** Images, files stored as data in Power BI.
- **Example:** File blobs, image data

Icon: Displayed as a **file** icon.

Note ↪

Using locale will be explained later.

Changing Data Types Manually

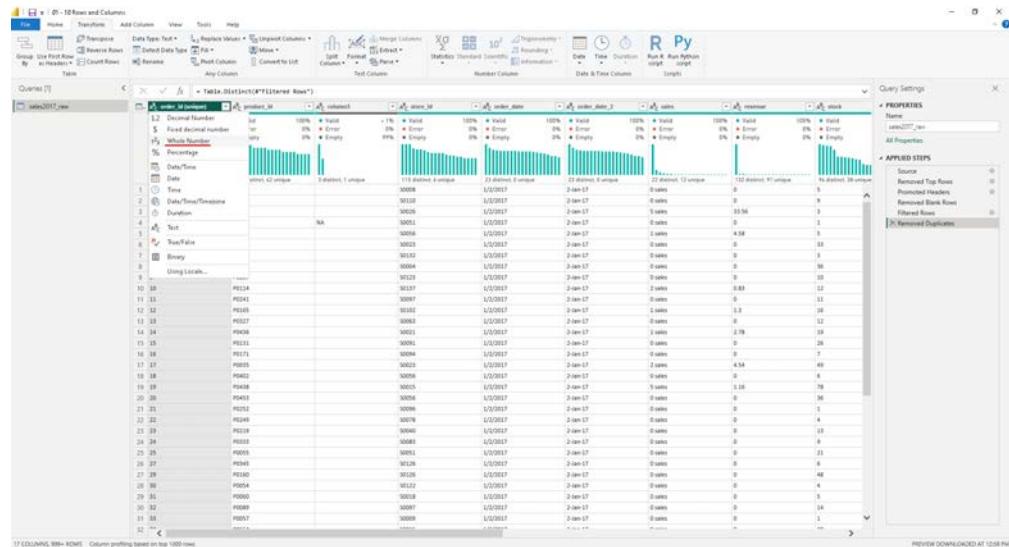
Sometimes Power BI may not correctly detect the data type of a column, especially if the data is messy or inconsistent. Here's how you can manually change the data type:

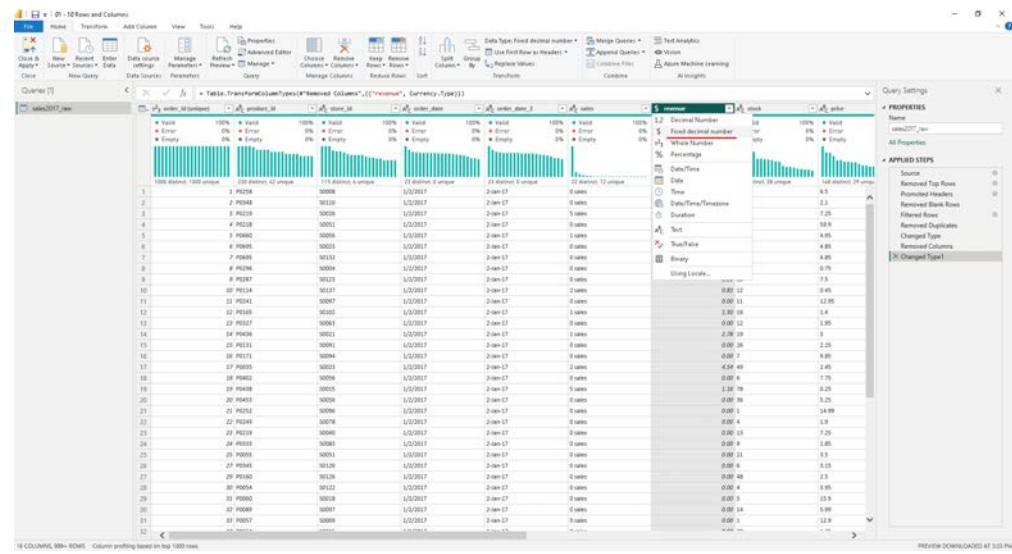
1. Select a Column

In the **Power Query Editor**, click on the column whose data type you want to change.

2. Change Data Type

In the ribbon at the top, under the **Transform** tab, there is a section called **Data Type**. Click on the dropdown arrow, and you will see a list of data types you can choose from, such as:





3. Confirming Data Type Changes

Once you select the appropriate data type, Power BI will apply the change. It's essential to ensure that the data type matches the column's content, as this affects how Power BI interprets and processes the data during transformations and calculations.

Using "Detect Data Type" Feature

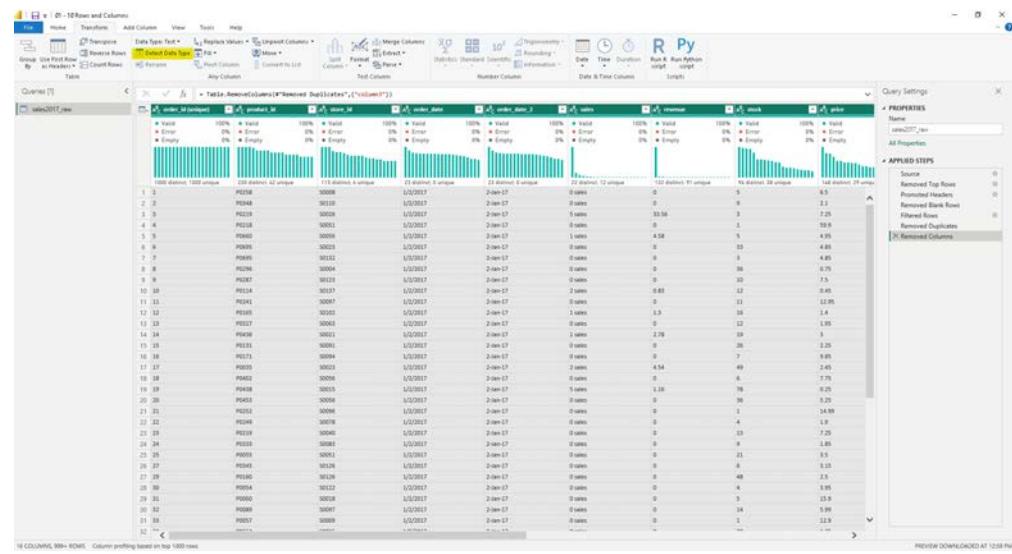
Power BI provides an automatic way to detect and set the data type for each column using the **Detect Data Type** feature. To use this:

1. Select the Columns

In the Power Query Editor, you can select one or multiple columns by holding down the **Ctrl** key.

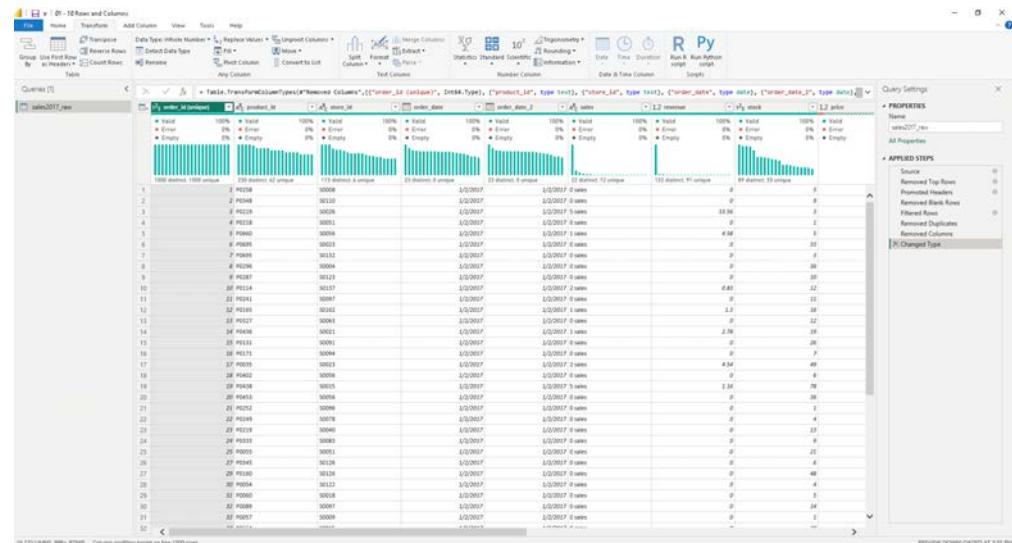
2. Click "Detect Data Type"

In the **Home** tab of the Power Query Editor, you will find the **Detect Data Type** button. Clicking this will prompt Power BI to reassess the data and set the most appropriate data type for each selected column.



3. Review the Changes

After detecting data types, review each column to ensure that Power BI has applied the correct data type.



Automatic Detection of Data Types

When you load a dataset into Power BI, the application will automatically attempt to detect the data types of the columns (We learned that we can

turn this feature on or off.). These can include text, numbers, dates, and more. This automatic detection helps you get started quickly, but it is important to manually review and adjust the data types if necessary.

Why Correct Data Types Matter

Choosing the correct data type for each column is vital for several reasons:

- **Performance:** Ensuring the right data type improves Power BI's performance by reducing memory and processing overhead.
- **Accurate Calculations:** Data types ensure that numeric, date, and text calculations are accurate, preventing unexpected errors.
- **Data Relationships:** Correct data types help Power BI establish accurate relationships between tables, which is essential in building robust data models.
- **Data Transformation:** Data types affect how you clean and manipulate data in the Power Query Editor, as some transformations are specific to certain types.

▼ Changing Data Types

We'll explore the importance of data types in Power BI and how to change them effectively. Understanding and correctly setting data types is crucial for accurate data analysis, especially when working with numerical data like revenue.

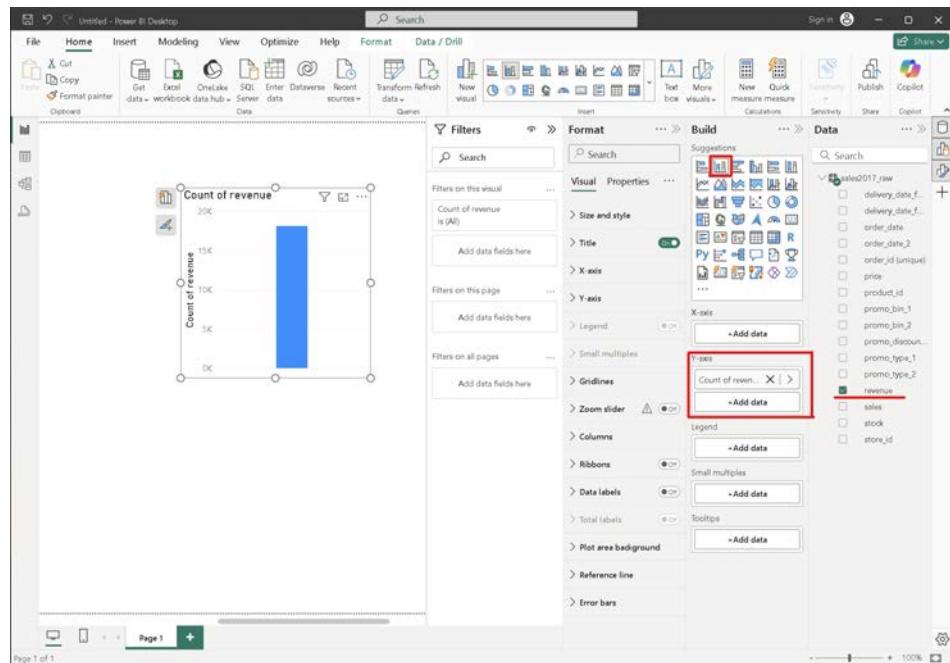
Why Data Types Matter:

Data types determine how Power BI interprets and processes your data. For instance, if a revenue column is incorrectly set as a text type instead of a number, you won't be able to sum or perform calculations on it.

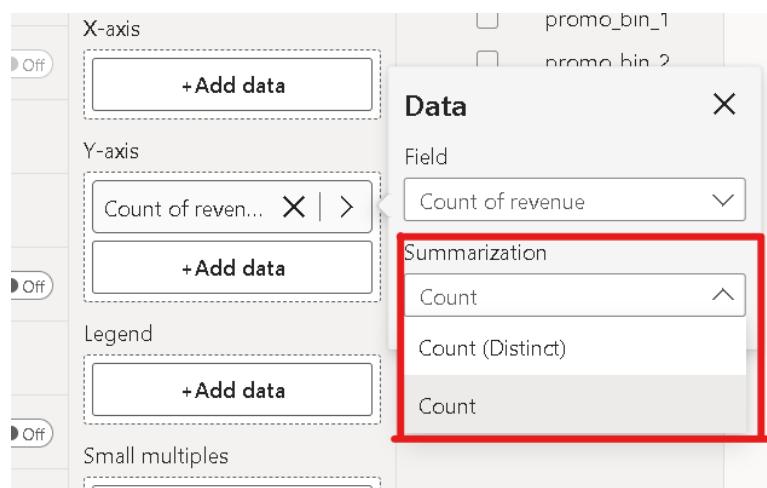
Demonstration:

1. Attempting to Summarize Revenue:

- First, let's try to summarize the revenue data. If you drag the revenue column onto the canvas in the report view, it may appear as a table by default. You can change the visual type to something like a stacked column chart.

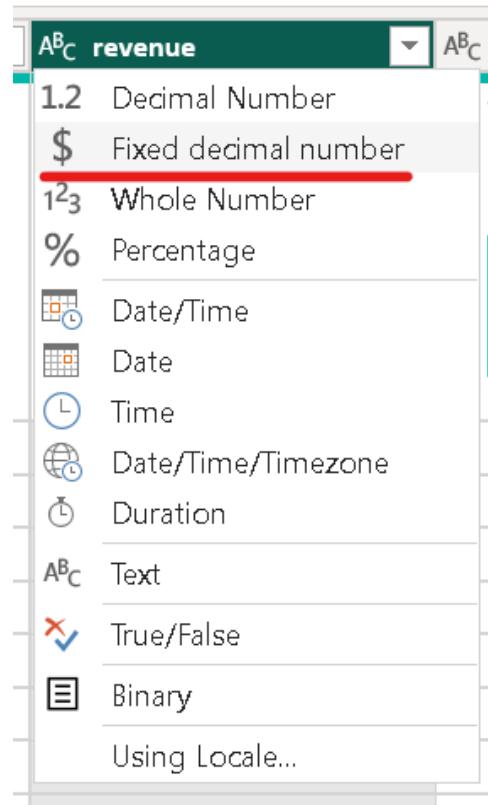


- However, if the revenue shows as a count instead of a sum, it's likely because the data type is set incorrectly.



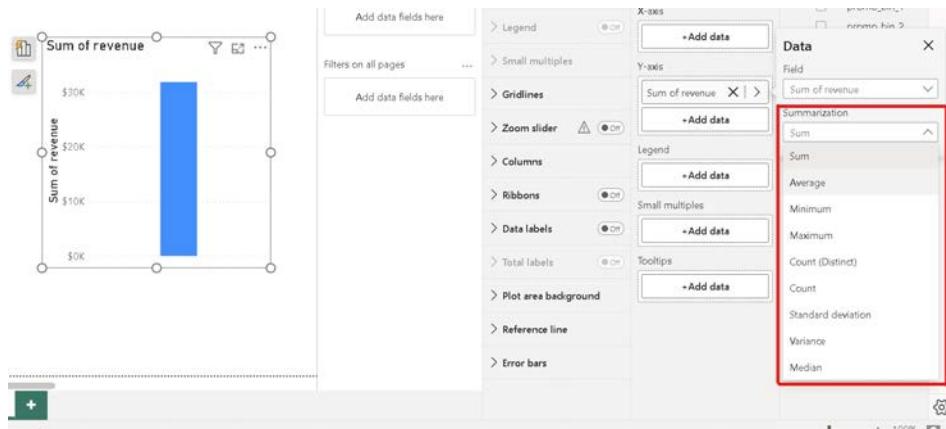
2. Identifying and Changing the Data Type:

- Go back to the **Query Editor**. You'll see that the revenue column is identified as text (indicated by the "ABC" icon). To allow Power BI to sum the revenue, we need to change this to a numerical data type.
- Click on the "ABC" icon and choose **Fixed Decimal Number**. This is a good choice for financial data like revenue because it stores values with up to four decimal places, which is typically sufficient for currency.



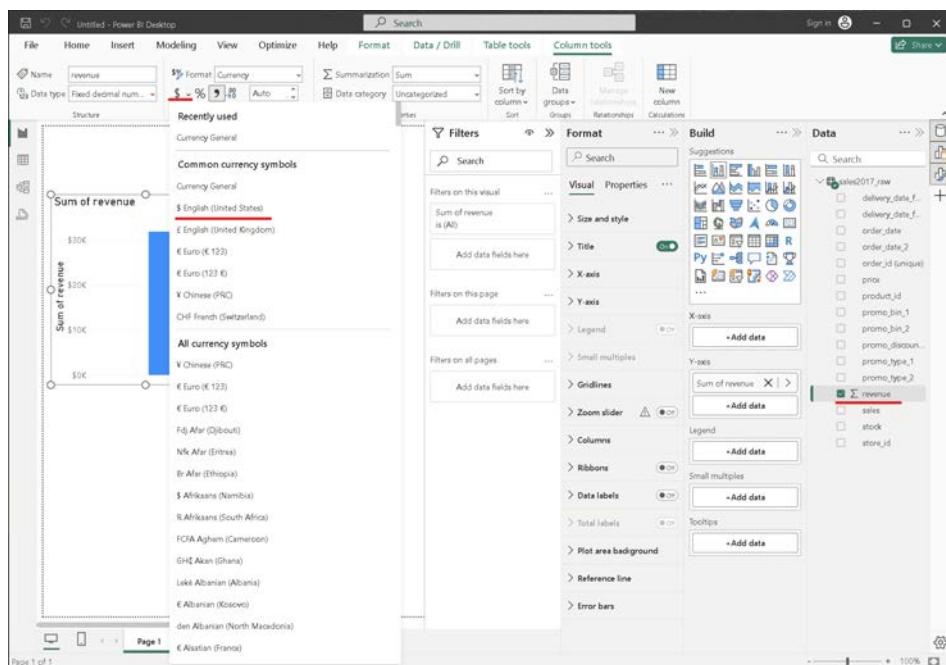
3. Applying Changes:

- After changing the data type, click **Close & Apply** to return to the report view. Now, when you drag the revenue column onto the canvas, you should be able to change the aggregation method to sum, allowing you to see the total revenue.



4. Formatting the Currency:

- If you need to change the currency format (e.g., from euros to dollars), you can do this in the report view. Select the revenue column, go to **Column Tools**, and under **Format**, choose the desired currency type.



5. Other Columns:

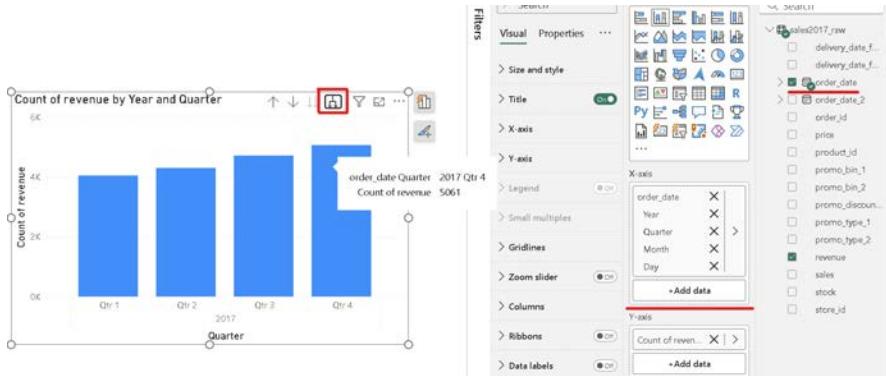
- Order ID:** This column should be a whole number. Change the data type to **Whole Number** by following a similar process.

A ^B C	order_id (unique)	A ^B C
1.2	Decimal Number	
\$	Fixed decimal number	
1 ² 3	Whole Number	
%	Percentage	
	Date/Time	
	Date	
	Time	
1	Date/Time/Timezone	2
2	Duration	3
3	Text	2
4	True/False	2
5	Binary	2
6	Using Locale...	5
7		

- **Order Date:** Dates should be set to the **Date** data type to allow Power BI to recognize them as dates rather than plain text. You can select multiple columns at once by holding the Ctrl key and applying the date type to all selected columns.

A ^B C	order_date	A ^B C	order_date_2	A ^B C
● Valid	100%	1.2	Decimal Number	V
● Error	0%	\$	Fixed decimal number	E
● Empty	0%	1 ² 3	Whole Number	E
		%	Percentage	
			Date/Time	
			Date	
			Time	d
			Date/Time/Timezone	al
			Duration	al
			Text	al
			True/False	al
			Binary	al
			Using Locale...	al
	23 distinct, 0 unique			
	1/2/2017			
	1/2/2017			
	1/2/2017			
	1/2/2017			
	1/2/2017			
	1/2/2017			
	1/2/2017			
	1/2/2017			

- Note: Allows further breakdown into quarters and other time segments (to be covered in detail later).



6. Handling Locale-Specific Date Formats:

- Sometimes, date formats differ based on locale (e.g., Germany vs. the United States). If Power BI fails to recognize a date format correctly, it may produce errors. In such cases, you can change the locale for specific columns to ensure correct interpretation.

By properly setting data types, you ensure that Power BI interprets your data correctly, allowing for accurate calculations and analysis.

▼ Data Types Using Locale

When working with data in Power BI, it's important to make sure that the data types of your columns match the expected formats and locales.

When we tried to change the data type of the

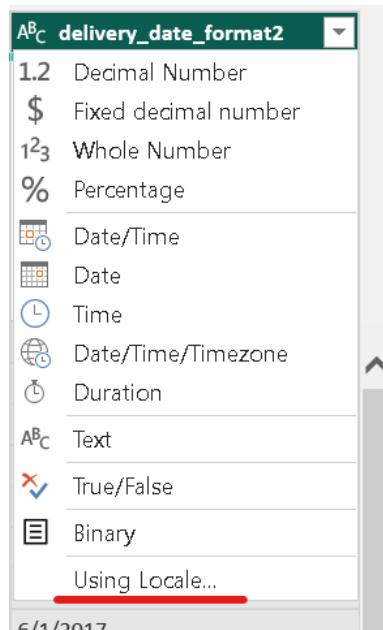
`delivery_date_format2` column, we would get errors because the date format didn't match the default `English-United States` locale set for our file.



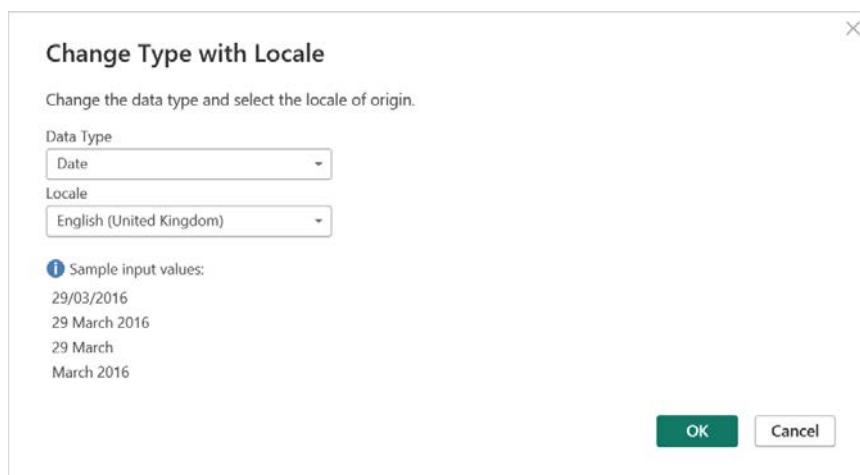
To resolve this, we needed to apply a different locale specifically for this column. Here's how to do that:

1. Change Data Type with Locale:

- Select the column and click on the data type icon.
- Instead of directly selecting a data type, choose the "Using Locale" option.



- This allows you to set both the data type and the locale. For instance, if the day is in the first position of your date format, you might choose `English-United Kingdom` as the locale instead of `English-United States`.

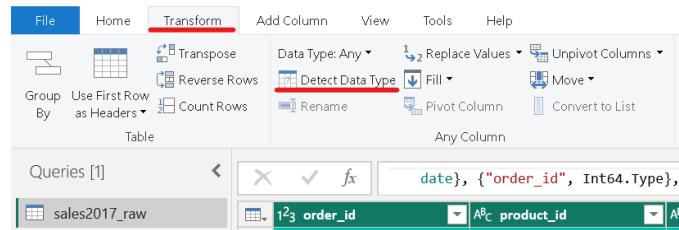


2. Verify the Data:

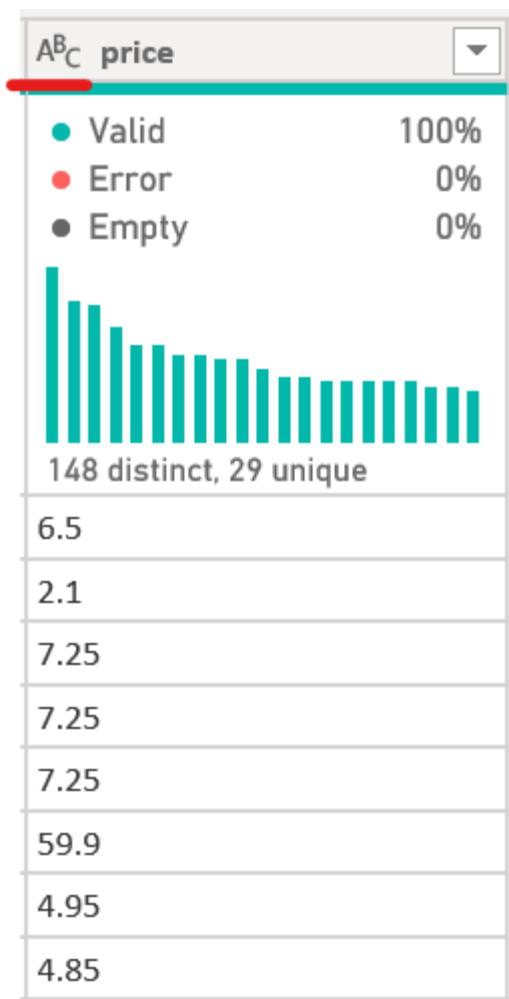
- After applying the new locale, ensure that the data is recognized correctly by hovering over the column. Power BI will show if 100% of the values are valid.
- If needed, you can still manually adjust the data type, like setting it to `Date` or other relevant types.

3. Automatically Detecting Data Types:

- Power BI also offers an option to automatically detect data types. This can be found under the **Transform** menu.
- After selecting "Detect Data Type," Power BI will attempt to identify the correct data type for each column.

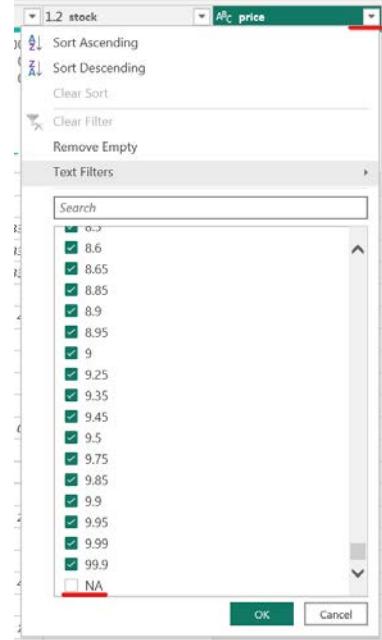


- However, always double-check critical columns, such as prices or numerical data, as they may not always be recognized correctly.
- For example, **price** is not properly recognized.



4. Handling Errors and NA Values:

- If some values are still not recognized correctly, such as `NA` values, you may need to clean the data further.
- You can load more data to identify any remaining issues and resolve them by removing or replacing incorrect values.



- After cleaning, apply the "Detect Data Type" option again or select the "Fixed Decimal Number" option to ensure correct recognition of the data.
 - Before doing the filtering process we had 18115 rows. After filtering the NA values we observe that there are 17491 rows left. Can we choose to fill with NA value 0?

5. Final Adjustments:

- Once all data types are correctly set, verify each column to ensure they are suitable for analysis. For example, numerical columns should not include text that would prevent calculations like averages or sums.

By following these steps, you can ensure that your data is properly formatted and ready for analysis in Power BI. In the next section, we will cover how to further clean your data by removing or replacing values to ensure accuracy in your reports.

▼ Replacing Values

In data preparation, it's common to encounter values that need modification to fit the desired format or data type. For instance, if we look at the sales column, we might find a mix of numbers and text, which is inconvenient when trying to convert these values into a numeric data type, as it can lead to errors. To address this, we need to replace or modify certain values in the dataset.

Here's a structured approach to replacing values in Power BI:

1. Identify the Problematic Values:

- Begin by identifying the column with the problematic values, such as the sales column where numbers are mixed with text.

2. Replace Values:

- Select the problematic column (e.g., `Promo Bin`) and navigate to the "Replace Values" option, found under the `Transform` section of the Home ribbon.
- Enter the value you want to replace (e.g., "very low") and specify the new value. In this case, you might simply want to add a space or completely remove the text.

The screenshot shows the Power BI desktop interface with the 'Transform' ribbon tab selected. A data grid is visible below the ribbon, containing columns for price, promo_type_1, promo_bin_1, and promo_type_2. The 'promo_bin_1' column contains values like 'verylow', 'moderate', and 'PRO3'. A 'Replace Values' dialog box is open in the foreground, with the 'Value To Find' field containing 'verylow' and the 'Replace With' field containing 'very low'. The 'OK' button is highlighted.

	\$ price	Avg promo_type_1	Avg promo_bin_1	Avg promo_type_2
1	100% 0% 0%	Valid Error Empty	100% 0% 0%	Valid Error Empty
2	37 unique	147 distinct, 28 unique	9 distinct, 1 unique	6 distinct, 1 unique
3	5	6.50 PR14	verylow	PRO3
4	9	2.10 PR05		PRO3
5	3	7.25 PR14		PRO3
6	3	7.25 PR14		PRO3
7	1	59.90 PR05	moderate	PRO3
8	5	4.95 PR14		PRO3

Replace Values

Replace one value with another in the selected columns.

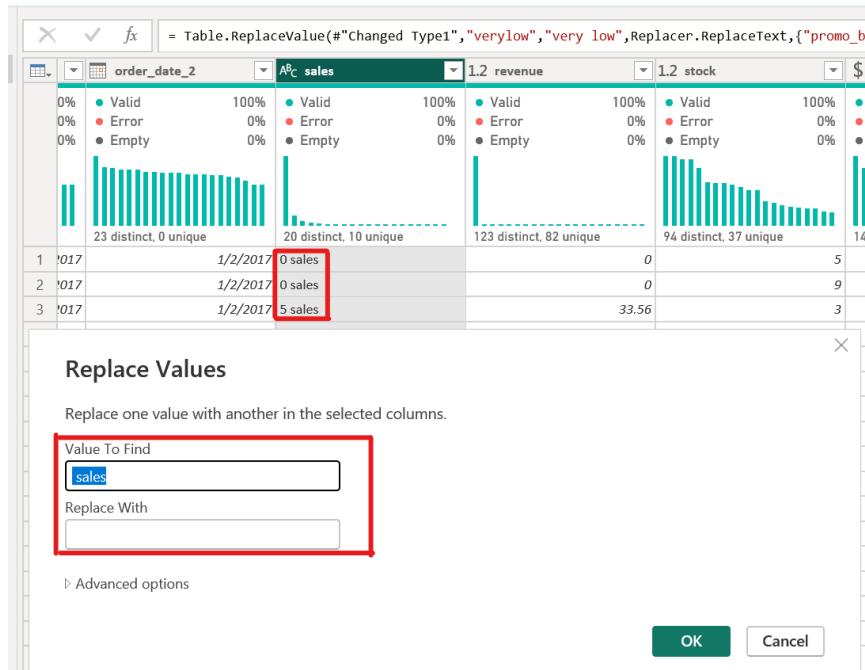
Value To Find: verylow

Replace With: very low

Advanced options

OK **Cancel**

- What we want to do for the price column is remove the text 'sales'. Notice there is a space character before 'sales'. So the values we need to find are 'sales' and we change it to nothing (i.e. delete it).



By following these steps, you can efficiently clean and prepare your data, ensuring that all values are correctly formatted for analysis in Power BI. This process is essential for accurate data modeling and reporting. In the next section, we'll apply these techniques in a practical project to reinforce what we've learned.

▼ Summary of Key Learnings So Far

As we've covered several important topics, let's recap what we've learned so far. This summary will help consolidate your understanding before you apply your knowledge in the course practice project.

1. Getting Data:

- We started by exploring how to import data into Power BI. You can connect to various data sources by simply clicking the "Get Data" option. As you progress, you'll encounter more data sources and learn how to work with them effectively.

2. Understanding Power BI Views:

- **Report View:** This is where you build and design your reports.
- **Data View:** Here, you can get an overview of the data you're working with.

- **Model View:** This view is used for creating relationships between tables and building your data model.

3. Query Editor:

- We learned how to use the Query Editor to transform data. The key features include:
 - **Applied Steps:** These allow you to track and manage each transformation. You can navigate between steps, remove, modify, or insert new steps as needed. However, be cautious when inserting steps, as this can affect subsequent transformations.

4. Replacing Values:

- We discussed how to replace specific values within a column, which is crucial for data cleaning. This process involves selecting the column and using the "Replace Values" feature.

5. Column Management:

- You learned how to rename and edit columns. This includes:
 - **Removing Columns:** You can remove unwanted columns, duplicates, blank rows, or errors.
 - **Right-Click Options:** Right-clicking on a column allows for quick removal or other actions.

6. Changing Data Types:

- Finally, we covered how to change data types by selecting the appropriate icon. Additionally, we explored how to use locales to ensure data types are correctly interpreted based on regional settings.

▼ Project - 1

[Project1.rar](#)

▼ Chapter 2

▼ Extracting Values

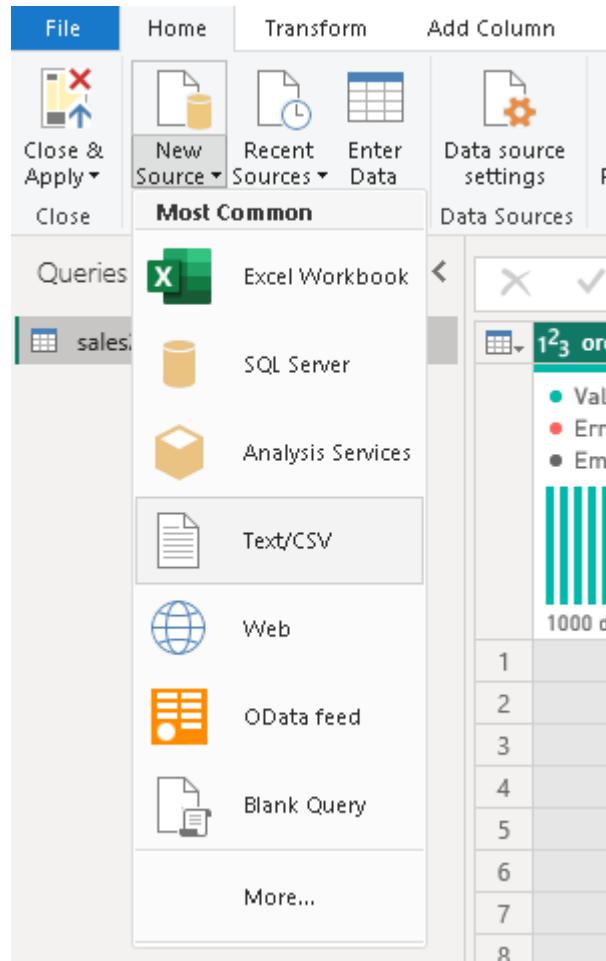
In this section, we'll explore how to extract specific values from a column in Power BI, allowing us to isolate the content we need for analysis.

Importing a New Data Source

We begin by importing a new data source directly from the Query Editor. Here's how:

1. Accessing Data:

- From the Home ribbon, select **New Source**.



- In this example, we'll import a CSV file by selecting **Text/CSV** and navigating to the file location (e.g., `store_cities.csv`).
- After previewing the data, click **OK** to load it into the Query Editor.

2. Navigating Queries:

- Once imported, the new dataset will appear in the Query Editor. You can switch between different queries, such as `sales2017` and `store_cities`, to view and manage them.

3. Renaming Queries:

- To maintain clarity, it's a good practice to rename your queries. For instance, rename `store_cities` to `stores` and `sales2017` to `Sales2017`.

Data Cleaning and Preparation

Next, we clean the imported data to ensure it's ready for use:

1. Setting Headers:

- If the first row of your data is not set as headers, go to the Home ribbon and select `Use First Row as Headers`.

2. Extracting Specific Values:

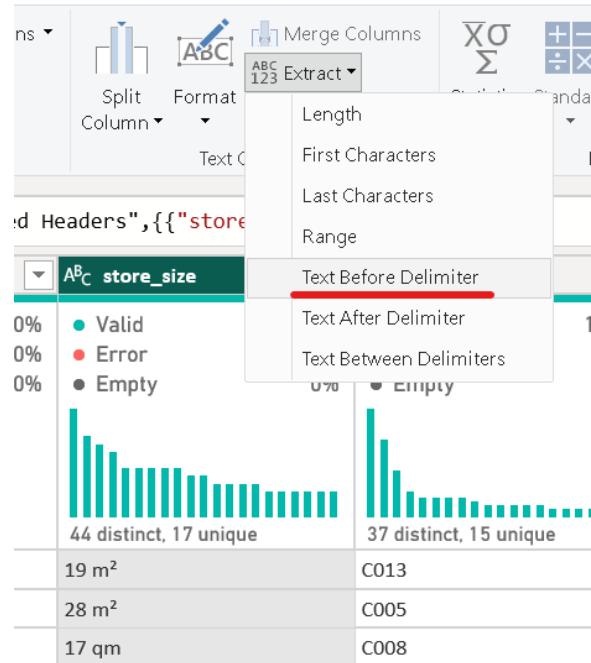
- Sometimes, you need to extract specific parts of a column's data, such as abbreviations or numeric values.
- For example, if you want to extract numbers from a `store_size` column:
 - Rename Column:** First, rename the column for clarity (e.g., `store_size`).



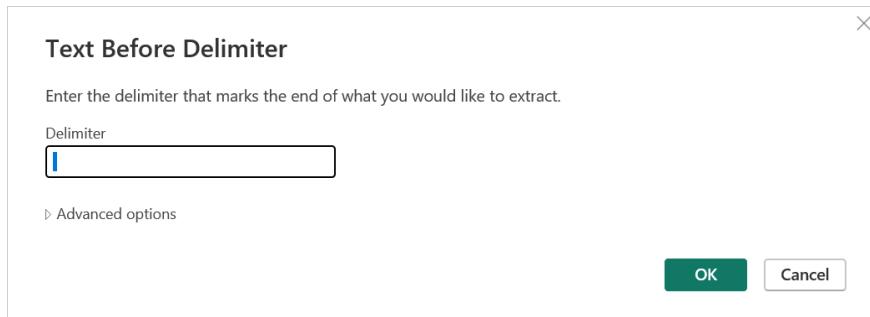
o Extract Values:

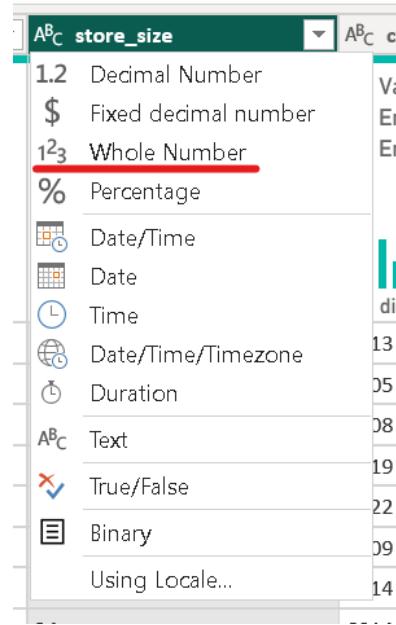
- Select the column and go to `Transform > Extract`.

- Use **Text Before Delimiter** to extract numbers before a specific character (e.g., a whitespace).



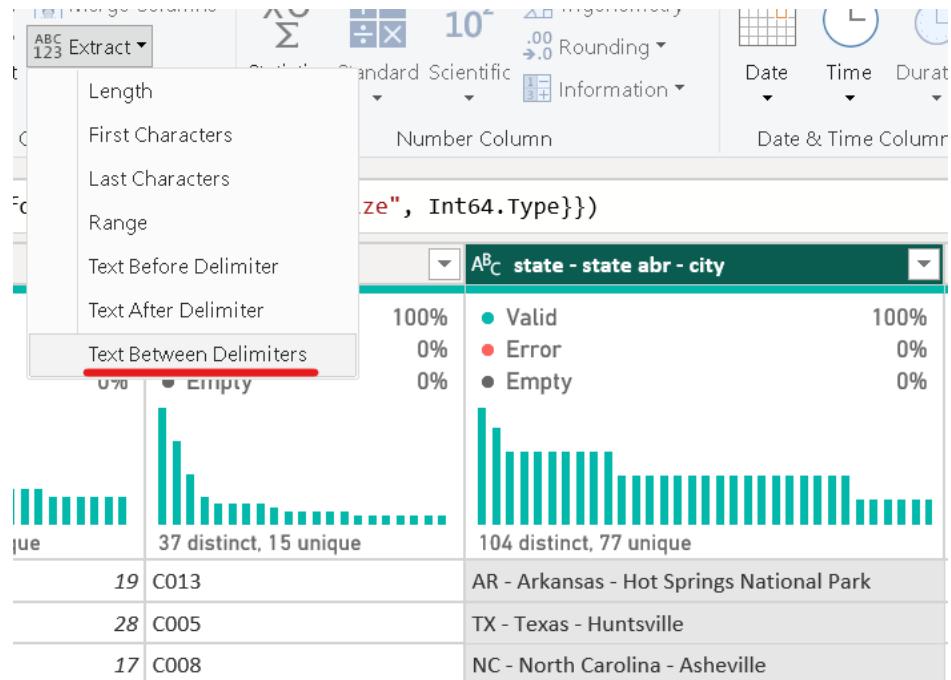
- Enter the delimiter (e.g., whitespace) and click **OK**. The numbers will be extracted and can now be transformed into a whole number data type.



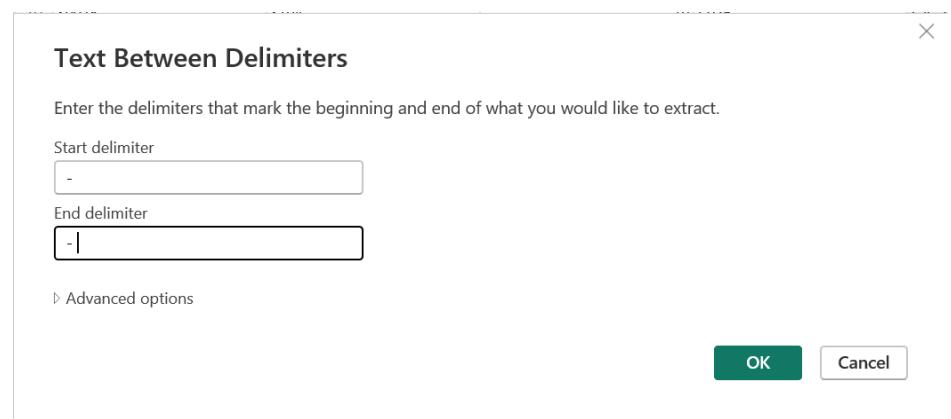


3. Advanced Extraction:

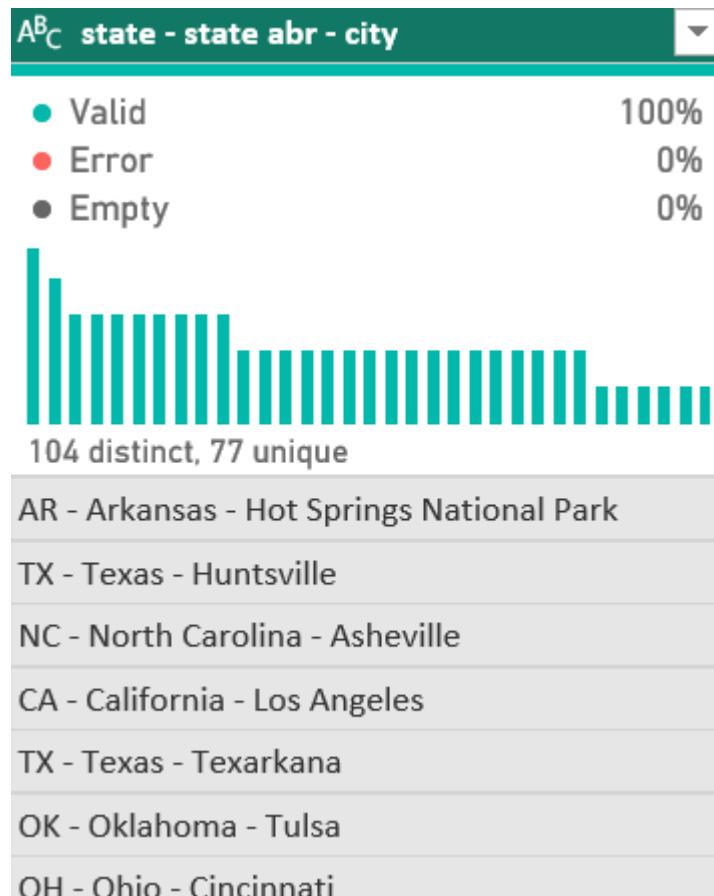
- To extract a specific part of a text, such as a state abbreviation from a combined city-state column:
 - Use **Text Between Delimiters** to specify the start and end delimiters. For example, to extract the state abbreviation between a hyphen and a space, use " - " as both the start and end delimiter.



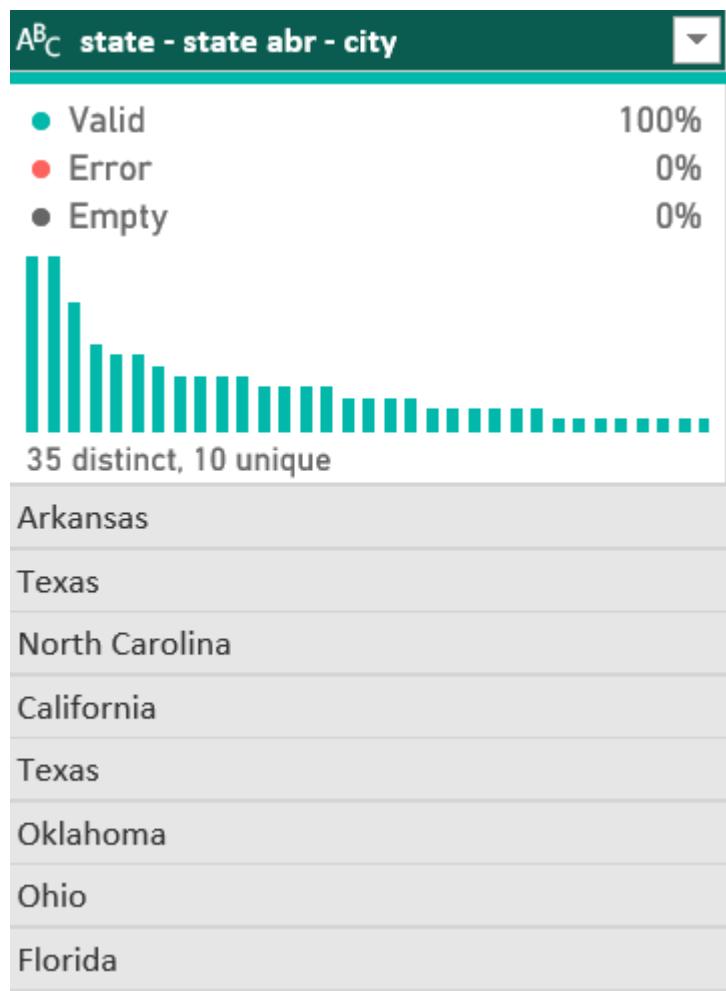
- Click **OK**, and the text between the delimiters will be extracted.



- Notes:
 - As you may have noticed, this process may cause us to lose data (State, State Abr, City). Since we do not want this situation, we need to apply different processes.
 - Before:



- After:



What do we do if we need to keep other parts of the data after extracting a value (for example, keeping both city and state information)? I'll explain in the next section.

▼ Split Columns

In the previous section, we discussed how to extract values from a column. However, there are cases where a single column contains multiple pieces of information that would be more useful if separated. In such scenarios, splitting the column into multiple columns is the best approach.

Splitting State and City Data

Let's consider a column that includes the state abbreviation, the full state name, and the city name all in one. Splitting this into separate columns will make the data easier to analyze.

1. Selecting the Column:

- First, select the column you want to split. In our example, this would be the column containing state abbreviations, state names, and city names.

2. Using the Split Column Tool:

- Go to the Home ribbon and click on **Split Column**.
- The **Split Column** functionality in Power BI allows users to divide a single column into multiple columns based on different criteria. This feature is available under the "Transform" tab in Power BI's Query Editor. Below are the seven primary methods for splitting a column:

1. **By Delimiter:** Splits the column based on a chosen character, like a comma or space.

- Example: Splitting the column `Full Name` by space into `First Name` and `Last Name`.

2. **By Number of Characters:** Splits after a specified number of characters.

- Example: Splitting a phone number `1234567890` into two columns as `12345` and `67890`.

3. **By Positions:** Splits at specific character positions you define.

- Example: Splitting an `ID` such as `123-456-789` at positions 3 and 7 to get separate parts `123`, `456`, and `789`.

4. **By Lowercase to Uppercase:** Splits where a lowercase letter is followed by an uppercase letter.

- Example: Splitting `FirstNameLastName` into `First Name` and `Last Name`.

5. By Uppercase to Lowercase: Splits where an uppercase letter is followed by a lowercase letter.

- Example: Splitting `HELLOWorld` into `HELLO` and `world`.

6. By Digit to Non-digit: Splits where a number is followed by a non-number character.

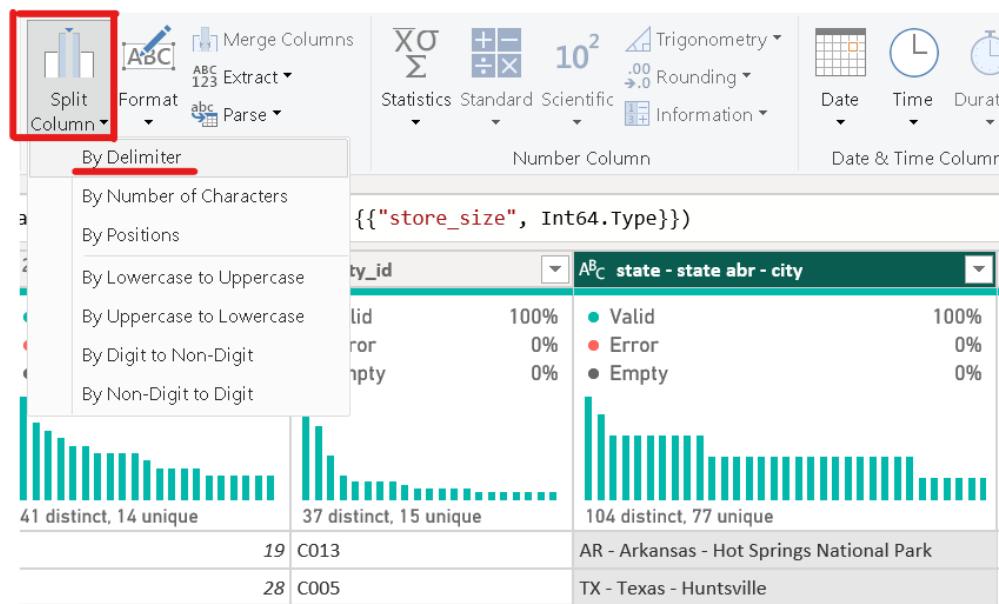
- Example: Splitting `Product123Code` into `Product`, `123`, and `Code`.

7. By Non-digit to Digit: Splits where a non-number character is followed by a number.

- Example: Splitting `Invoice1234` into `Invoice` and `1234`.

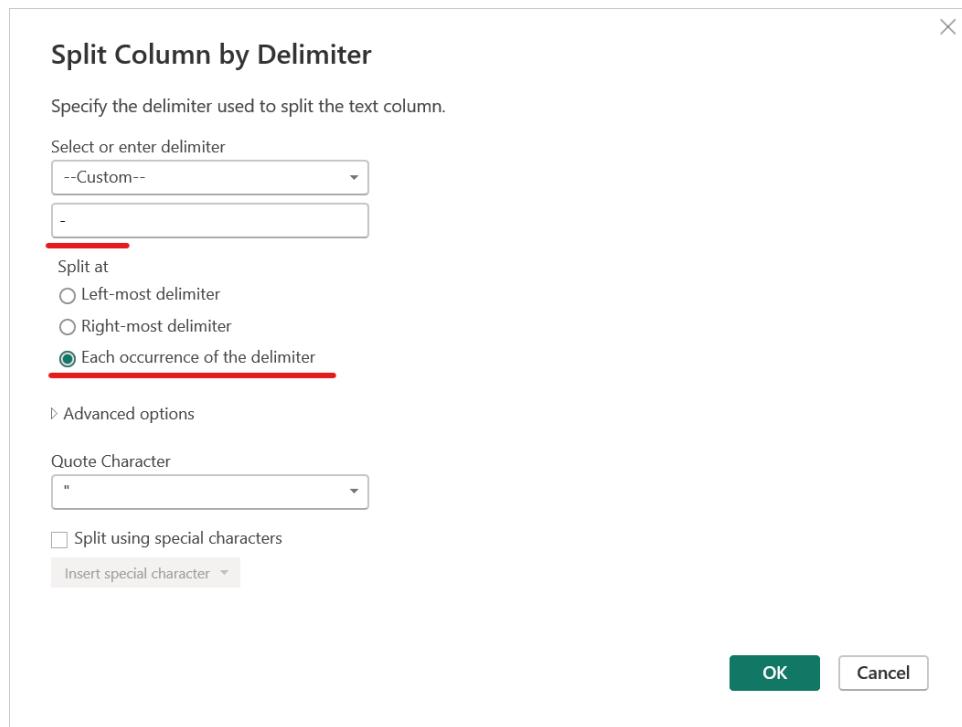
3. Splitting by Delimiter:

- For our example, we would use `By Delimiter`.



- You can choose a delimiter from the default options (e.g., comma, colon) or define a custom one. For instance, if the state and city are separated by a hyphen, select `Custom` and enter the hyphen as the delimiter.
- Ensure that you select to split at each occurrence of the delimiter, not just the first or last one, to create the necessary

columns.



4. Renaming the Columns:

- After splitting, rename the new columns for clarity (e.g., `state`, `abbr`, `state`, `city`).

A ^B _C state abr	A ^B _C state	A ^B _C city
<ul style="list-style-type: none">Valid 100%Error 0%Empty 0%  35 distinct, 10 unique	<ul style="list-style-type: none">Valid 100%Error 0%Empty 0%  35 distinct, 10 unique	<ul style="list-style-type: none">Valid 100%Error 0%Empty 0%  100 distinct, 72 unique
AR	Arkansas	Hot Springs National Park
TX	Texas	Huntsville
NC	North Carolina	Asheville
CA	California	Los Angeles

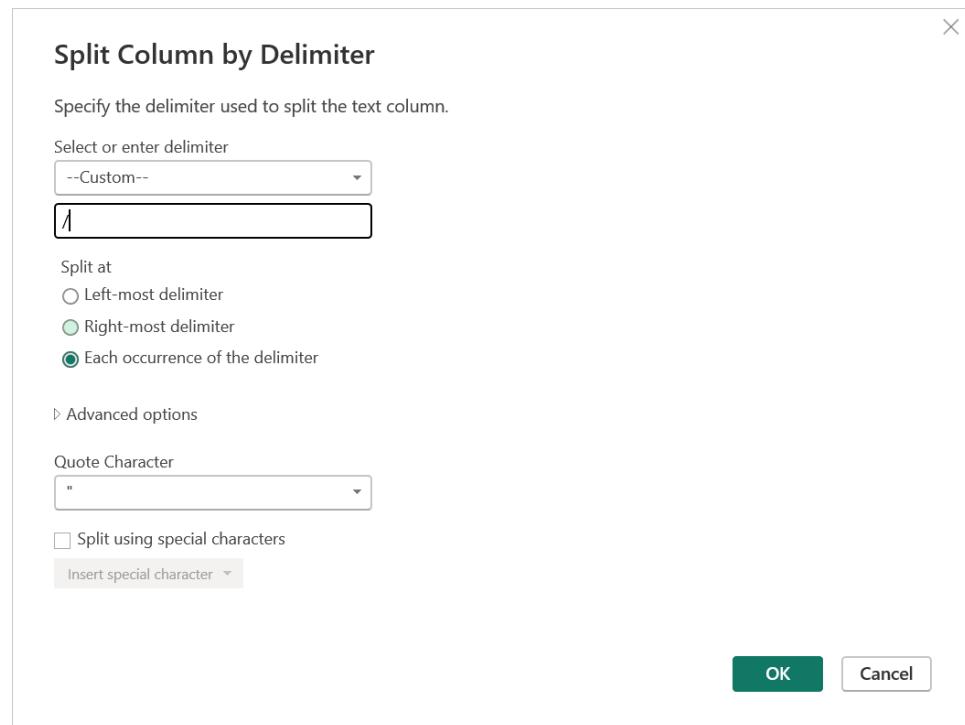
Splitting Latitude and Longitude:

1. Using the Split Column Tool:

- Similarly, if you have a column containing both latitude and longitude, you can split it by a delimiter such as a slash (/).

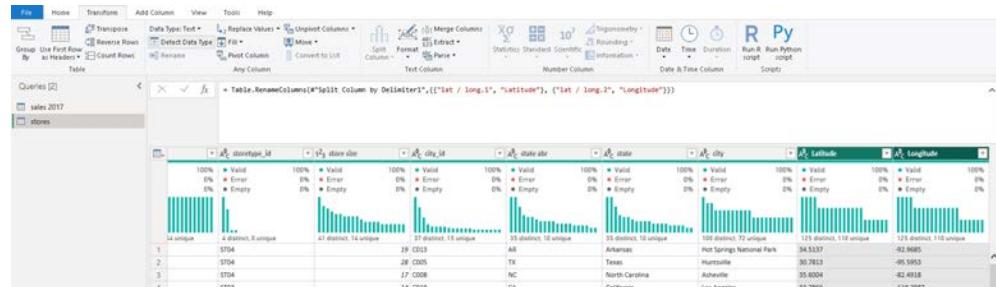


- Choose `Split Column` by `Delimiter`, use a whitespace and slash (/) as the delimiter, and rename the resulting columns to `Latitude` and `Longitude`.

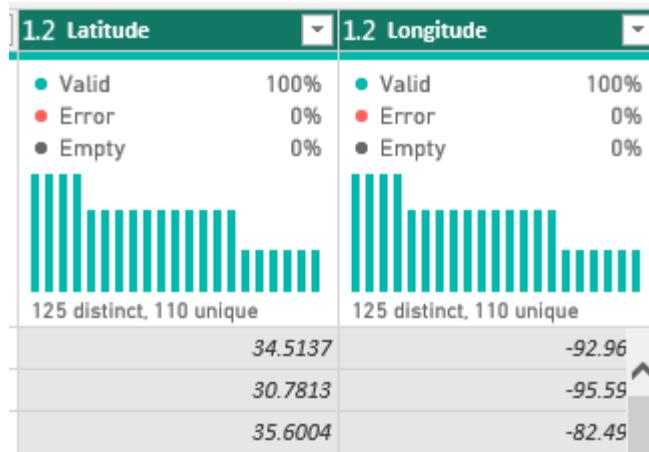


2. Setting the Correct Data Types:

- After splitting the columns, it's important to ensure that each column is assigned the correct data type. For example, `Latitude` and `Longitude` should be set to a numeric data type.



- Select all relevant columns, go to the `Transform` ribbon, and click `Detect Data Type` to automatically set the appropriate data types. Always double-check to ensure accuracy.

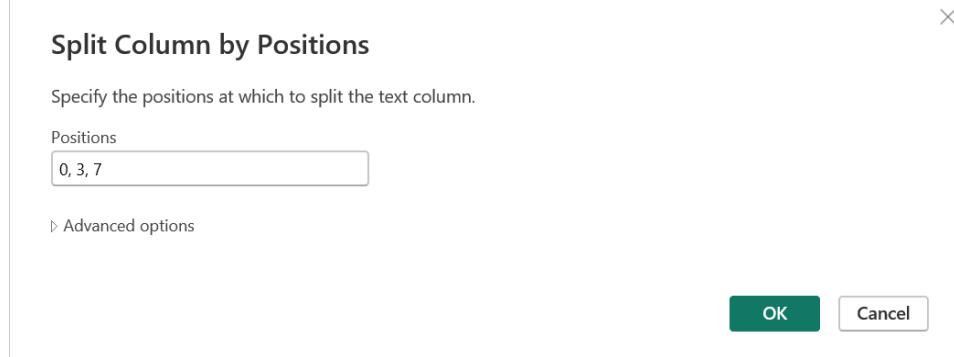


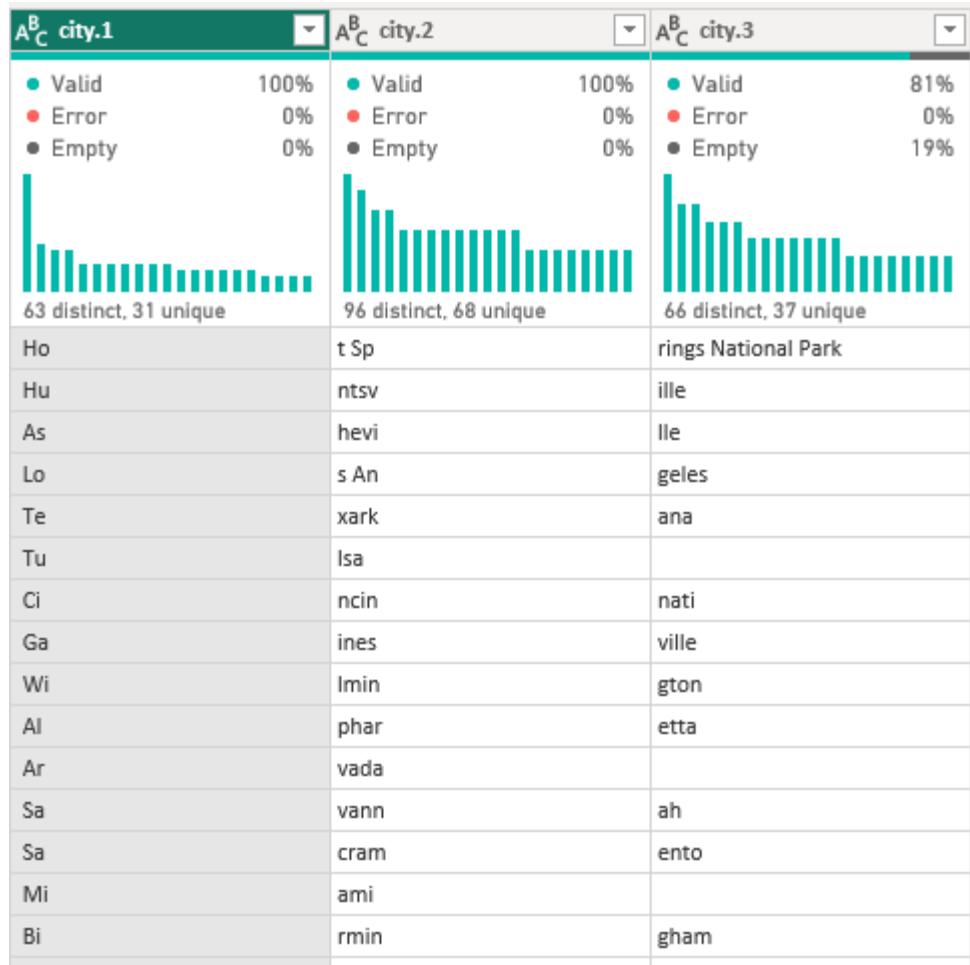
3. Advanced Splitting by Position:

- If your data requires splitting at specific character positions, use the `Split by Position` option. For example, you can split a column after the 3rd and 7th characters by specifying these positions.
- Note that the last column will contain the remaining data. This method is useful for highly structured data but may not be necessary for all datasets.

Attention !!

This example is purely for demonstration purposes.
This usage is not correct for this example.





4. Handling Leading Whitespace:

- After splitting, you may notice that some columns contain leading space. This can be cleaned up using text operations, which will be discussed in the next section, or we can prevent this with a small change to the delimiter.

By splitting columns effectively, you can enhance the clarity and usability of your data in Power BI, making it easier to perform detailed analysis and create meaningful reports.

▼ Text Operations

In the previous section, we split columns into multiple parts, which sometimes led to unwanted leading or trailing white spaces in the text. Additionally, text data can often contain hidden or non-printable characters, known as control characters, that need to be cleaned for

better data quality. In this section, we will explore the tools available in Power BI for cleaning and formatting text data.

Removing White Spaces with Trim

1. Identifying White Spaces:

- After splitting columns, you might notice white spaces at the beginning or end of the text. These spaces can interfere with data accuracy and appearance.

2. Using the Trim Function:

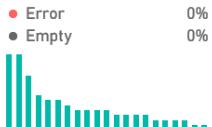
- To remove these unwanted spaces, select the column and either:
 - Go to the **Transform** tab and select **Format**, then choose **Trim**.
 - Or, more conveniently, right-click on the column, go to **Transform**, and select **Trim**.
- The **Trim** function removes all leading and trailing white spaces, leaving the text clean and properly formatted.

A screenshot of the Power BI desktop interface. On the left, there are two tables: 'state' and 'city'. The 'state' table has 35 distinct values, while the 'city' table has 100 distinct values. A context menu is open over the 'city' table, specifically over the first row ('Hot Springs National Park'). The menu is titled 'Transform' and includes options like 'Copy', 'Remove', 'Remove Other Columns', 'Duplicate Column', 'Add Column From Examples...', 'Remove Duplicates', 'Remove Errors', 'Change Type', 'Trim', 'Replace Values...', 'Replace Errors...', 'Split Column', 'Group By...', 'Fill', 'Unpivot Columns', 'Unpivot Other Columns', 'Unpivot Only Selected Columns', 'Rename...', 'Move', 'Drill Down', and 'Add as New Query'. The 'Trim' option is highlighted with a red box. To the right of the menu, there are sections for 'PROPEI' (Name: store_c, All Prop) and 'APPLIE' (Source, Pro, Ref). At the bottom, there are status bars for '33.3440' and '-80.9292'.

3. Applying Trim to Other Columns:

- Apply the `Trim` function to any other columns that may have similar issues, ensuring all data is consistent.
 - Deleted invisible white spaces on the right side of the State abr column.
 - Delete white spaces on the left and right sides of the State column.
 - Deleted whitespaces on the left side of the City column.

▪ Before trim:

<code>A^bC state abr</code>		<code>A^bC state</code>		<code>A^bC city</code>	
● Valid	100%	● Valid	100%	● Valid	100%
● Error	0%	● Error	0%	● Error	0%
● Empty	0%	● Empty	0%	● Empty	0%
					
35 distinct, 10 unique		35 distinct, 10 unique		100 distinct, 72 unique	
AR	Arkansas			Hot Springs National Park	
TX	Texas			Huntsville	
NC	North Carolina			Asheville	
CA	California			Los Angeles	
TX	Texas			Texarkana	
OK	Oklahoma			Tulsa	
OH	Ohio			Cincinnati	
FL	Florida			Gainesville	
DE	Delaware			Wilmington	
GA	Georgia			Alpharetta	
CO	Colorado			Arvada	
GA	Georgia			Savannah	
CA	California			Sacramento	
FL	Florida			Miami	
AL	Alabama			Birmingham	
TX	Texas			San Antonio	

▪ After trim:

A^B_C state abr		A^B_C state		A^B_C city	
● Valid	100%	● Valid	100%	● Valid	100%
● Error	0%	● Error	0%	● Error	0%
● Empty	0%	● Empty	0%	● Empty	0%
AR	Arkansas			Hot Springs National Park	
TX	Texas			Huntsville	
NC	North Carolina			Asheville	
CA	California			Los Angeles	
TX	Texas			Texarkana	
OK	Oklahoma			Tulsa	
OH	Ohio			Cincinnati	
FL	Florida			Gainesville	
DE	Delaware			Wilmington	
GA	Georgia			Alpharetta	
CO	Colorado			Arvada	
GA	Georgia			Savannah	
CA	California			Sacramento	
FL	Florida			Miami	
AL	Alabama			Birmingham	
TX	Texas			San Antonio	

Notes

We see that the data in the column is separated from each other in the form of a "whitespace-dash-whitespace" pattern. If we use only "-" in the process, there will be spaces in front and behind the data. Separating like " - " is the correct usage. As a result, if we use this method, we do not need to use the trim function.

Cleaning Control Characters with Clean

Let's add Product hierarchy data. The producthierarchy and sales2017 table have a common column, product_id. Therefore, these two tables are related.

We will examine the clean function on this data. So now to clean up this data:

- The first step would be, of course, just use the first row as headers.

1. Understanding Control Characters:

- Control characters like tabs or line breaks are non-printable characters that can be embedded in text, often causing issues in data processing.

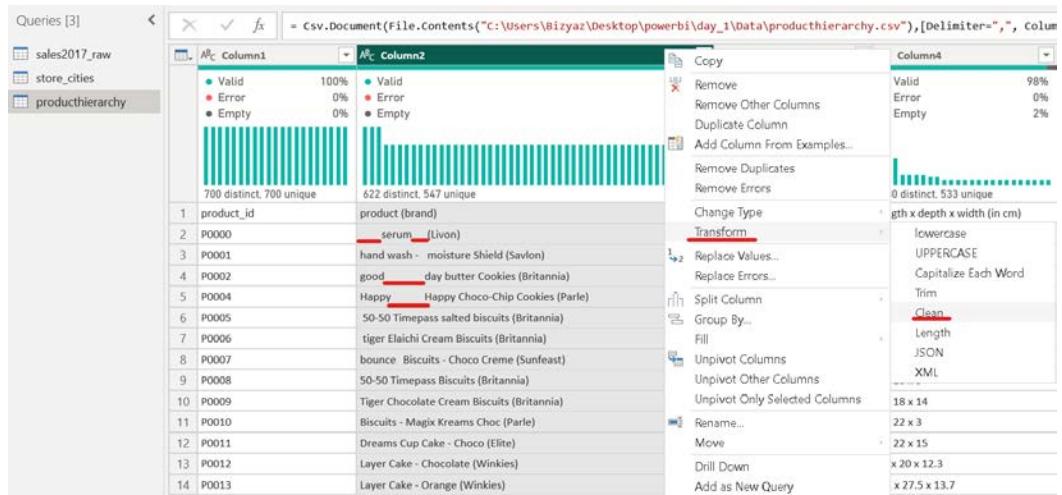
Queries [3]

= Table.PromoteHeaders(Source, [PromoteAllScalars=true])

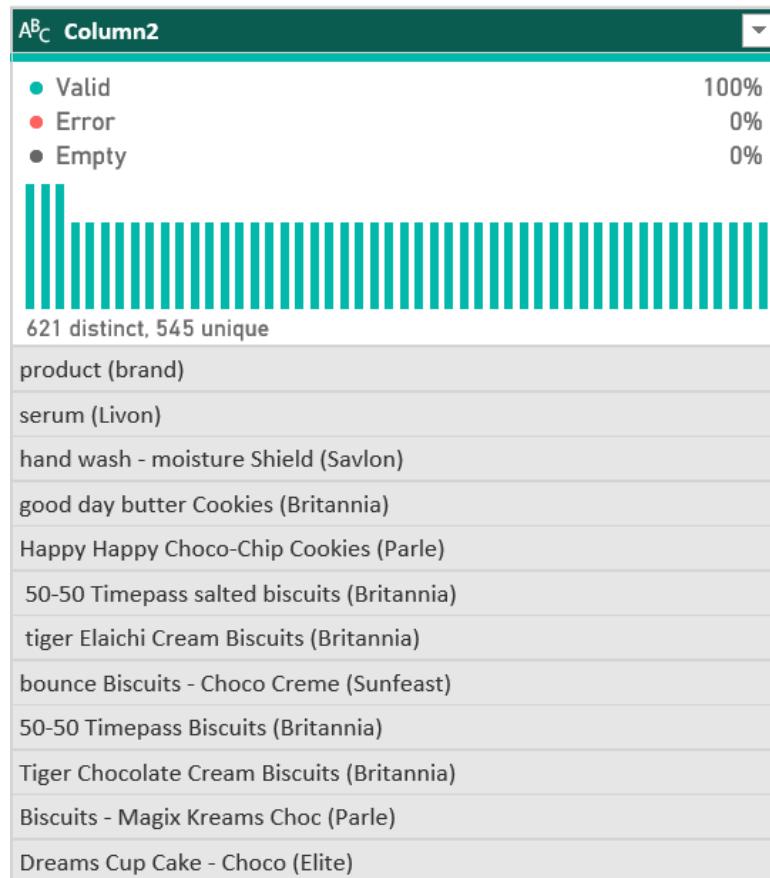
	A _B C product_id	A _B C product (brand)	A _B C
	● Valid 100%	● Valid 100%	100%
	● Error 0%	● Error 0%	0%
	● Empty 0%	● Empty 0%	0%
	699 distinct, 699 unique	621 distinct, 546 unique	11.
1	P0000	serum (Livon)	Ind
2	P0001	hand wash - moisture Shield (Savlon)	Hai
3	P0002	good day butter Cookies (Britannia)	Har
4	P0004	Happy Happy Choco-Chip Cookies (Parle)	Coc
5	P0005	50-50 Timepass salted biscuits (Britannia)	Glu
6	P0006	tiger Elaichi Cream Biscuits (Britannia)	Salt
7	P0007	bounce Biscuits - Choco Creme (Sunfeast)	Glu
8	P0008	50-50 Timepass Biscuits (Britannia)	Cre
9	P0009	Tiger Chocolate Cream Biscuits (Britannia)	Salt
10	P0010	Biscuits - Magix Kreams Choc (Parle)	Cre
11	P0011	Dreams Cup Cake - Choco (Elite)	Cre
12	P0012	Layer Cake - Chocolate (Winkies)	Mu
13	P0013	Layer Cake - Orange (Winkies)	Tea
14	P0014	Sugar Coated Chocolate (Cadbury Gems)	Tea
15	P0015	Cadbury Perk - Chocolate Bar (Cadbury)	Cho
16	P0016	Polo - The Mint With The Hole (Nestle)	Cho
17	P0017	Orbit Sugar-Free Chewing Gum - Lemon & Lime (Wrigleys)	Tof
18	P0018	Sugar Free Chewing Gum - Mixed Fruit (Orbit)	Mir
19	P0019	Chewing Gum - Peppermint (Doublemint)	Mir
20	P0020	Tomato - Local, Organically Grown (Fresho)	Org
21	P0021	Tomato - Local, Organically Grown (Fresho)	Org
22	P0022	Marie Light Biscuits - Active (Sunfeast)	Org
23	P0023	Fulltoss Thai Sriracha (Parle)	Bre
24	P0024	Fulltoss Tangy Tomato (Parle)	Nac
25	P0025	Exam Standard Scale (Camilin)	Nac
26	P0026	Dish Shine Bar (Exo)	Exa
27	P0027	Material Flower - Orange (Fresho)	Nic
	serum (Livon)		

2. Using the Clean Function:

- To remove control characters, select the problematic column.
- Right-click on the column, go to **Transform**, and select **Clean**.



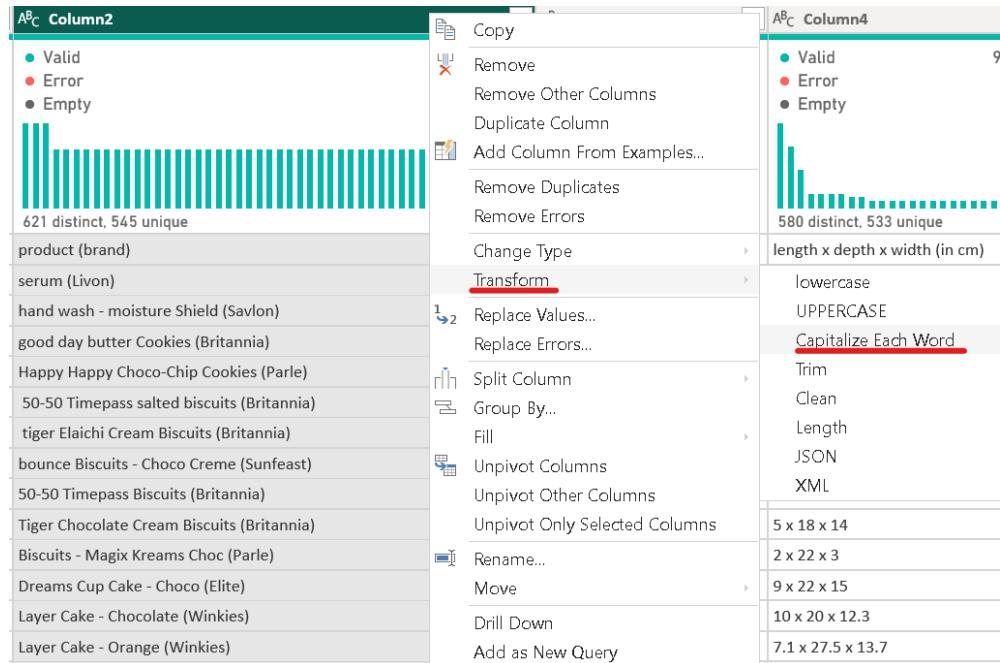
- This function removes all control characters, such as tabs or line breaks, from the text, leaving it clean and ready for analysis.



Additional Text Formatting

1. Capitalizing Each Word:

- You can further format your text by capitalizing the first letter of each word.
- Right-click on the column, go to **Transform**, and select **Capitalize Each Word**.
- This improves readability, especially in fields like product names or titles.



Splitting Columns Based on Text

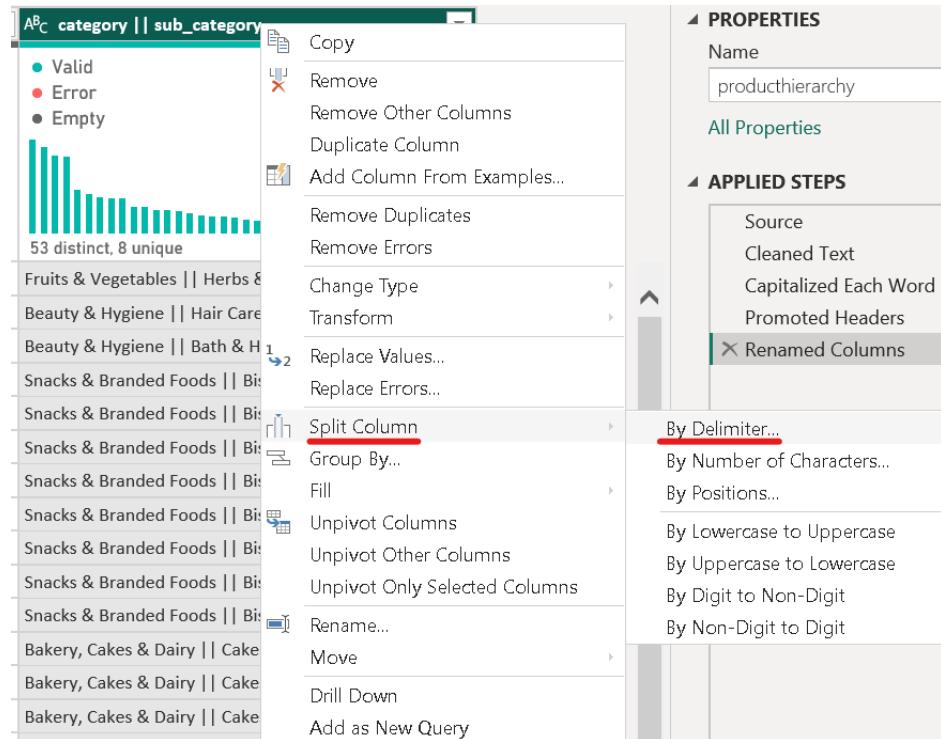
1. Identifying Columns for Splitting:

- After cleaning, you may still have columns containing multiple pieces of information, such as **Category** and **Subcategory**.
- If we examine the **Category** and **Subcategory** columns, we can see that the category and subcategory are separated by " || ".

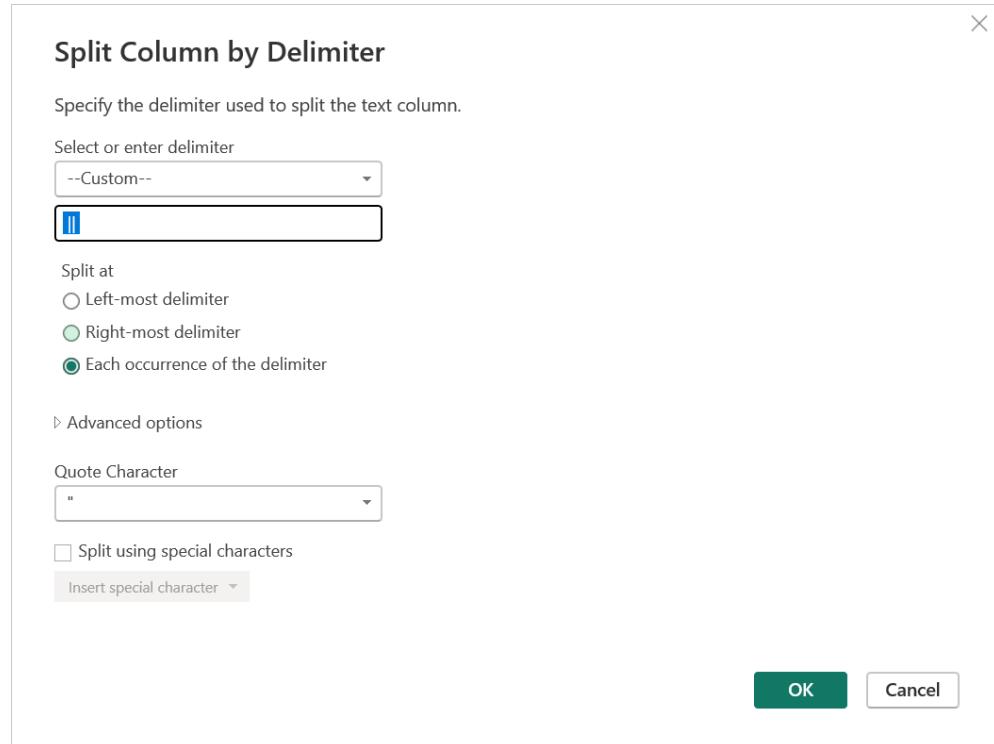
2. Splitting by Delimiter:

- To split these columns:
 - Right-click on the column and choose **Split Column**.

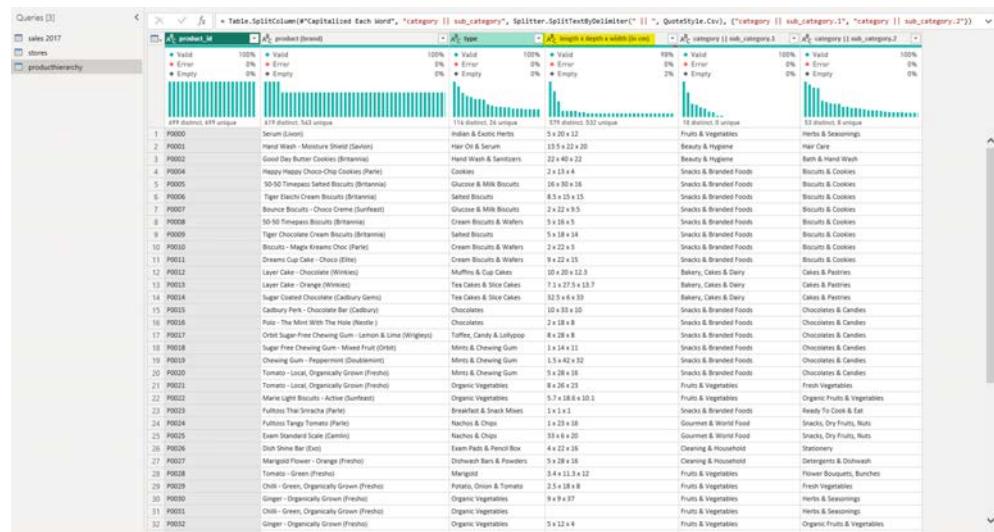
- Select **By Delimiter**, and choose the appropriate delimiter (e.g., a hyphen, space).



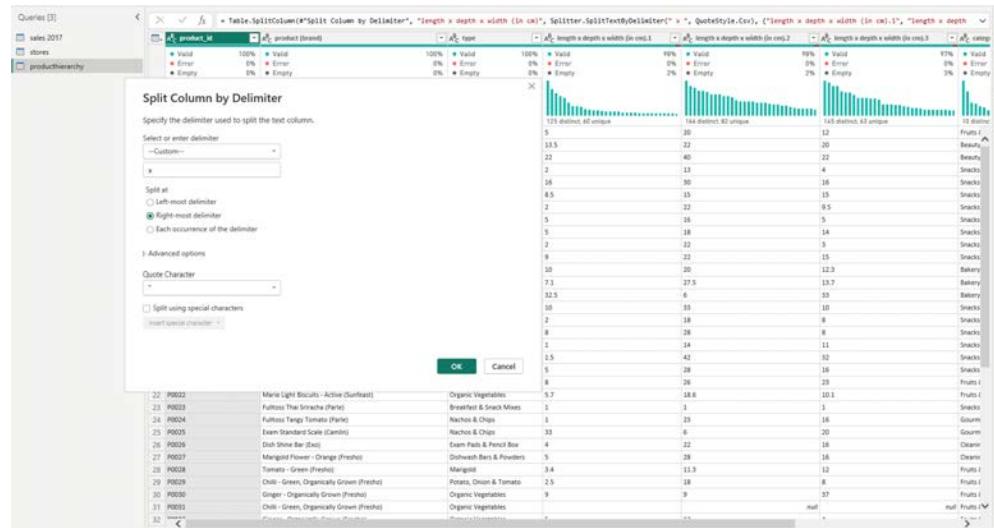
- This process creates separate columns for each piece of information.
- Remember: "whitespace - pipe - pipe - whitespace"



- We apply "delimiter" operation in the "length x depth x width" column.



- Similarly, we divide into columns using "whitespace - x - whitespace".



3. Renaming and Formatting:

- After splitting, rename the new columns to reflect their content accurately (e.g., `Length`, `Width`, `Category`, `Subcategory`).
- Ensure the data types are correctly set, especially for numerical columns.
- The result we get after all the operations is meaningful columns and meaningful data. (The length, depth, width columns were numbers in centimeters, but perhaps we want to display them in inches.)

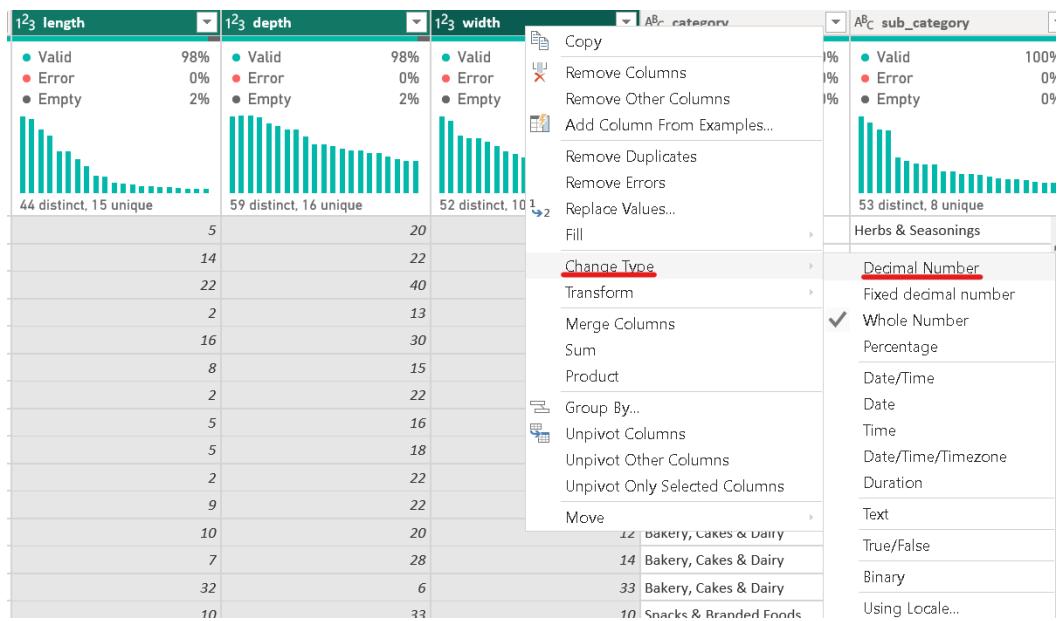
<code>length</code>	<code>depth</code>	<code>width</code>	<code>category</code>	<code>sub_category</code>
98% 0% 2%	98% 0% 2%	98% 0% 2%	97% 0% 3%	100% 0% 0%
44 distinct, 15 unique	59 distinct, 16 unique	52 distinct, 10 unique	10 distinct, 0 unique	53 distinct, 8 unique
5 14 22 2 16 8	20 22 40 13 28 15	12 20 22 4 18 5 18 14 3 22 15 20 27.5 6 13 22 8 14 11 42 28 16 26 28 10.1 1 16 20 16 12 8 37 null null	Fruits & Vegetables Beauty & Hygiene Beauty & Hygiene Bath & Hand Wash Biscuits & Cookies Biscuits & Cookies	Herbs & Seasonings Hair Care Bath & Hand Wash Biscuits & Cookies Biscuits & Cookies

▼ Numerical Operations

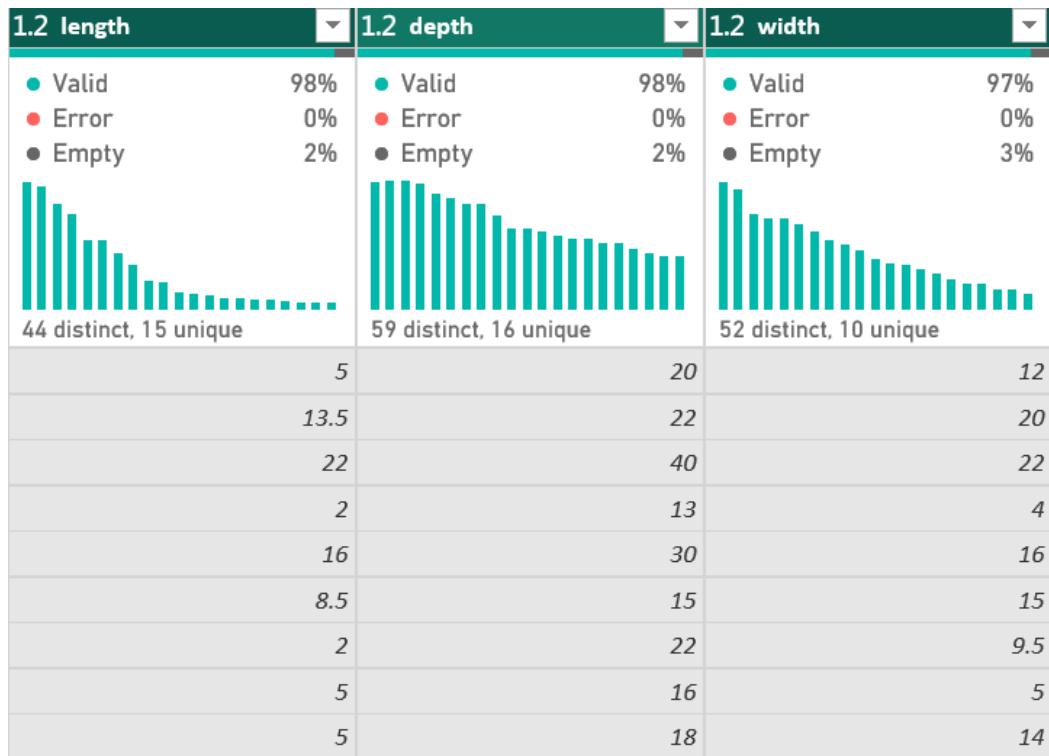
In this section, we'll explore the various transformations that can be applied to numerical columns in Power BI. Before proceeding, it's important to ensure that our numerical data is correctly typed.

Data Type Considerations:

- Initially, our columns were set as **Whole Numbers**. However, this data type rounds values (e.g., 13.5 becomes 14). If precision is required, particularly for values with decimal places, it's better to use the **Decimal Number** data type.

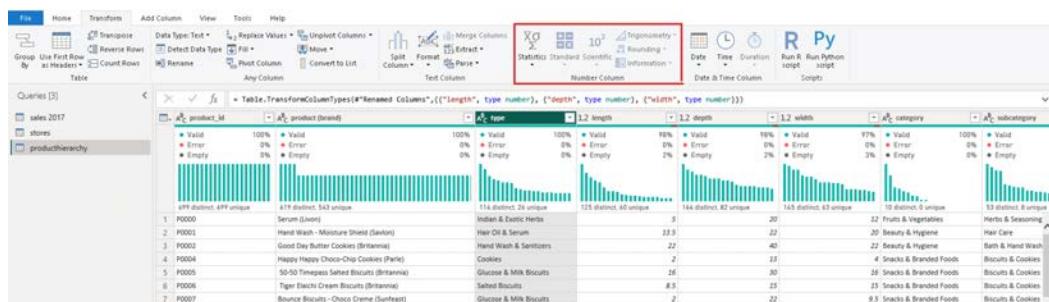


- To adjust this, change the data type from Whole Number to Decimal Number. This ensures that all decimal places are retained.



Applying Numerical Transformations:

- Transforming Data:** To apply numerical operations, select the relevant column and go to the **Transform** tab. Note that the options under "**Number Column**" will only be active if a numerical column is selected.

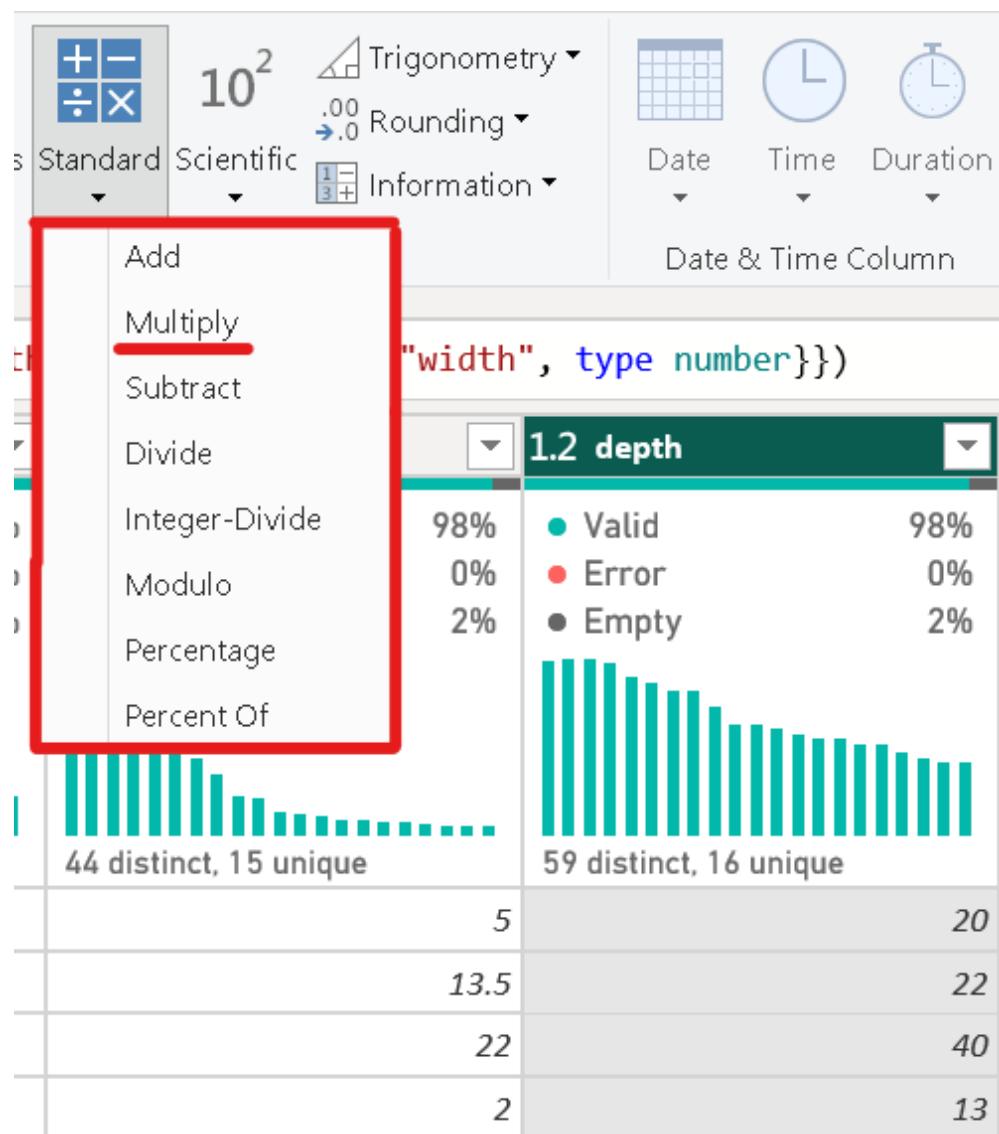


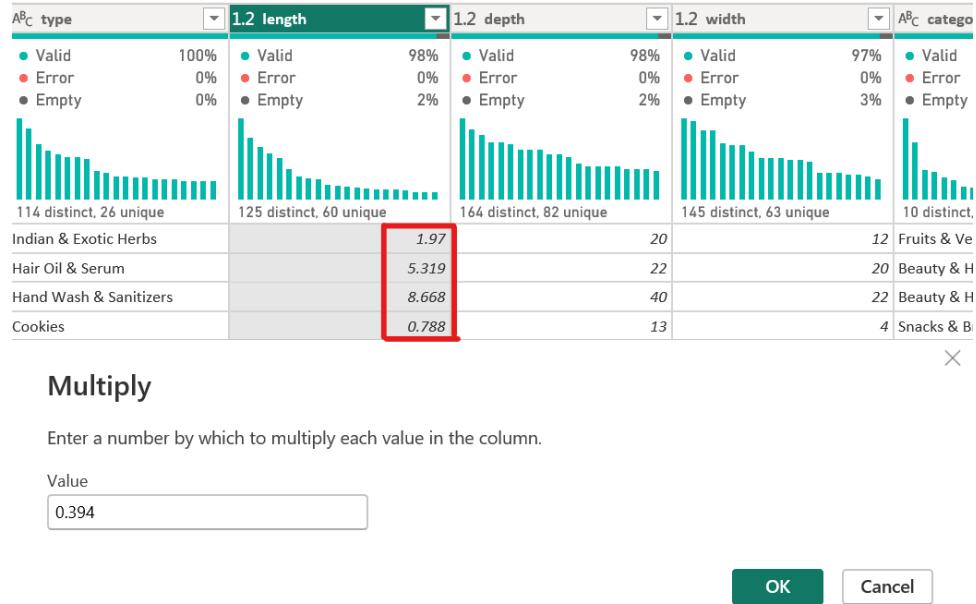
- Basic Operations:**

- Standard Operations:** This includes basic arithmetic operations like addition, subtraction, multiplication, and division. For

example, to convert values from centimeters to inches, you would multiply the column by 0.394.

- **Scientific Operations:** These include more advanced calculations such as logarithms and square roots.
- **Example:**
 - Suppose you want to convert a column of measurements from centimeters to inches. Select the column, choose the **Multiply** function under Standard Operations, and input the conversion factor (0.394). The column values will then be updated accordingly.





▼ Data Model & Relationships

In this section, we'll explore how to create and utilize relationships between tables in Power BI to gain insights from your data. Specifically, we'll look at how to determine which product category generated the highest revenue by linking data from the sales table with the product hierarchy table.

Understanding the Relationship:

- **Data Sources:**
 - **Sales Table:** Contains revenue data, including a **Product ID** for each order.
 - **Product Hierarchy Table:** Contains product categories, also identified by **Product ID**.

To analyze the revenue by category, we need to establish a connection between these two tables using the **Product ID** as a common key.

Steps to Create and Use Relationships:

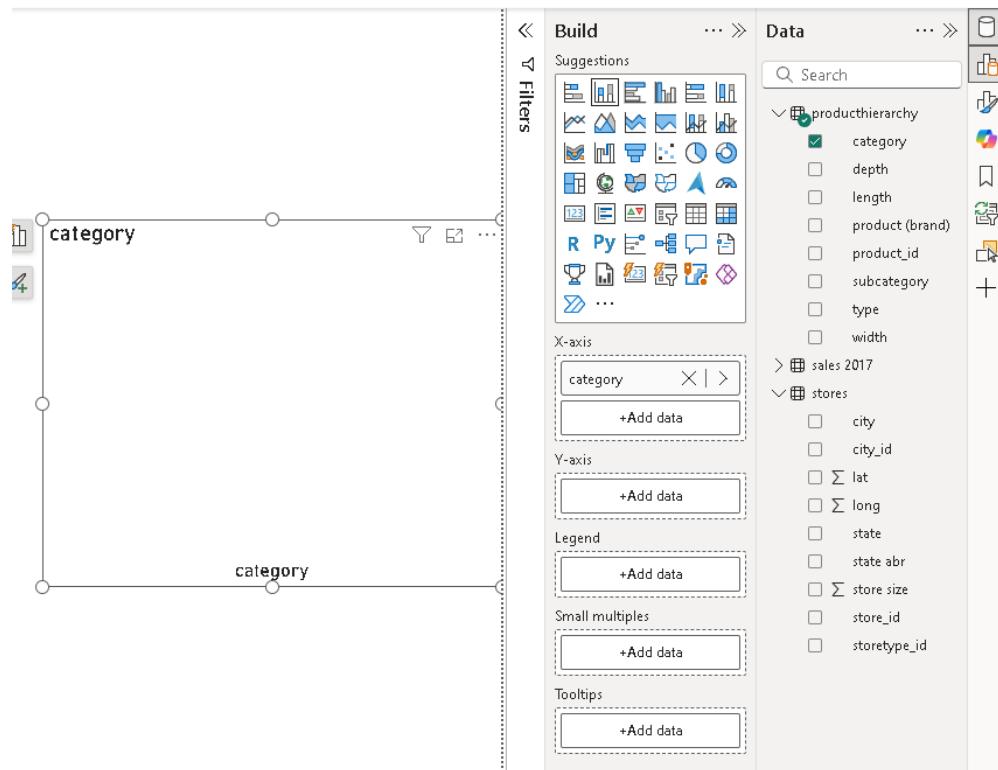
1. Prepare the Report:

- Close and apply any changes to the data, and clear any existing visuals.

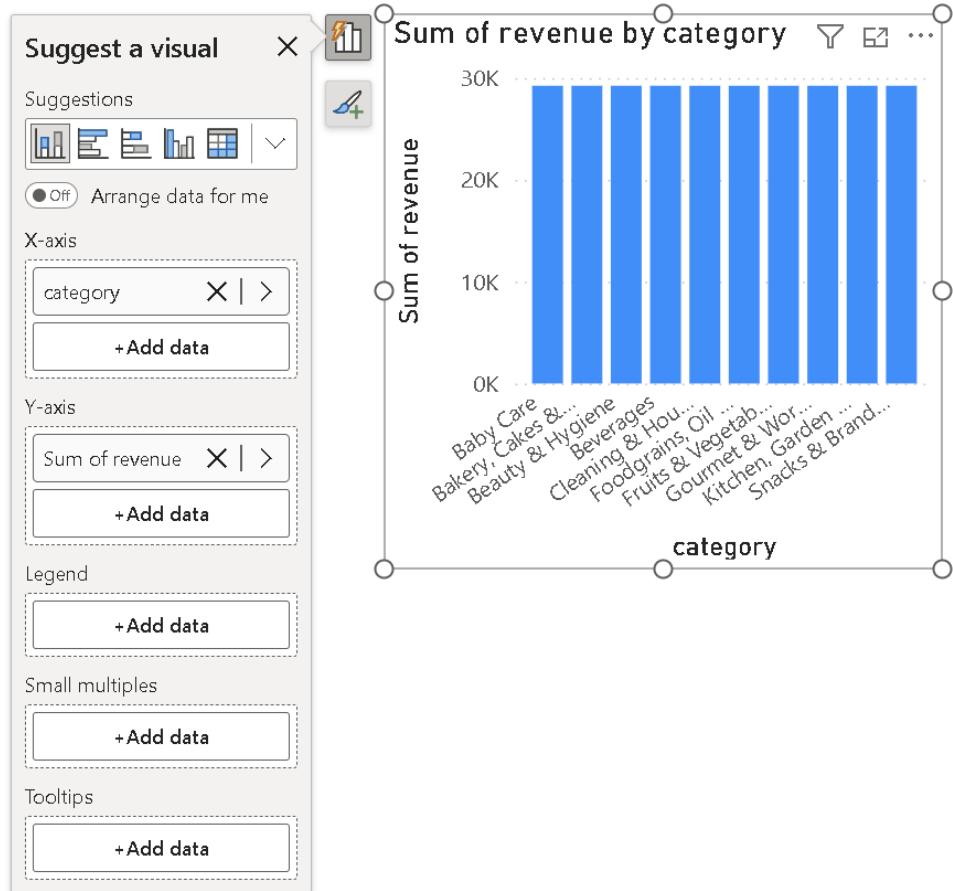
- Select a new visual, such as a **Stacked Column Chart** from the **Home** ribbon.

2. Adding Data to the Visual:

- Expand the **Product Hierarchy** table, drag the **Category** field into the visual's X-axis.



- Next, add the **Revenue** field from the **Sales** table to the Y-axis by either dragging it from the data pane or using the search function.

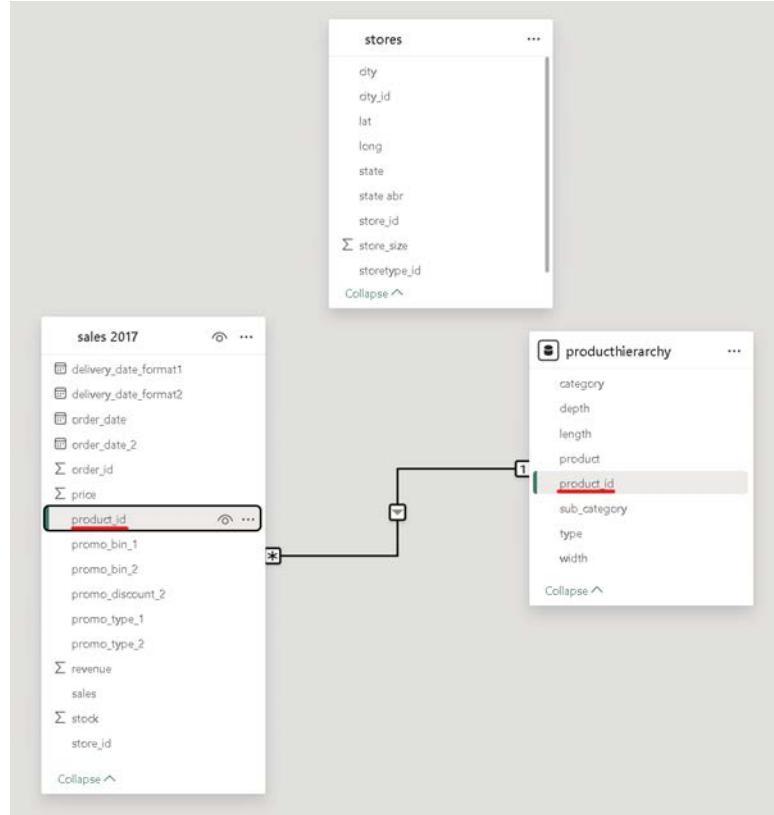


3. Identifying Relationship Issues:

- If all values appear identical, it's likely that the relationship between the tables is either incorrect or not established. This issue needs to be resolved in the **Model View**.

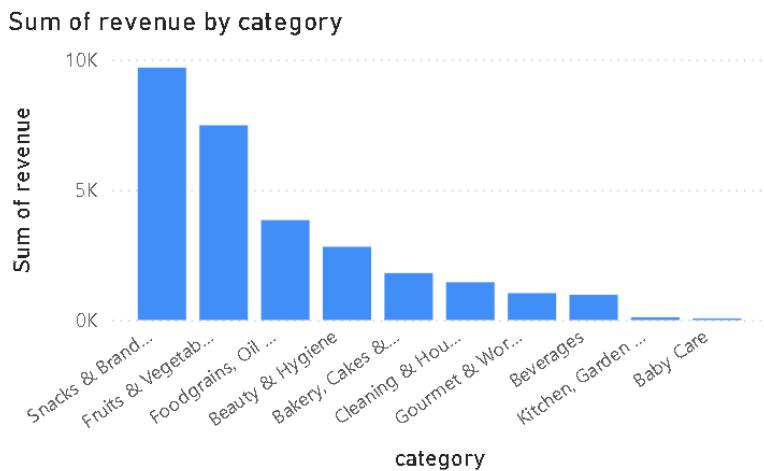
4. Creating the Relationship:

- Go to **Model View** to see all tables.
- Identify the common column, **Product ID**, in both the **Sales** and **Product Hierarchy** tables.
- Drag the **Product ID** from one table and drop it onto the **Product ID** of the other table to establish a relationship. It doesn't matter which direction you drag—both methods achieve the same result.



5. Verifying the Relationship:

- Once the relationship is established, return to the **Report View**.
- Now, the visual should correctly display the revenue by category. For instance, you may see that "Snacks and Brand Foods" is the top category in terms of revenue.



▼ Practice: Data Cleaning & Relationship

Practice.rar

This is a quick exercise to practice what we've learned so far.

Do what is needed to find out the answers to the following questions:

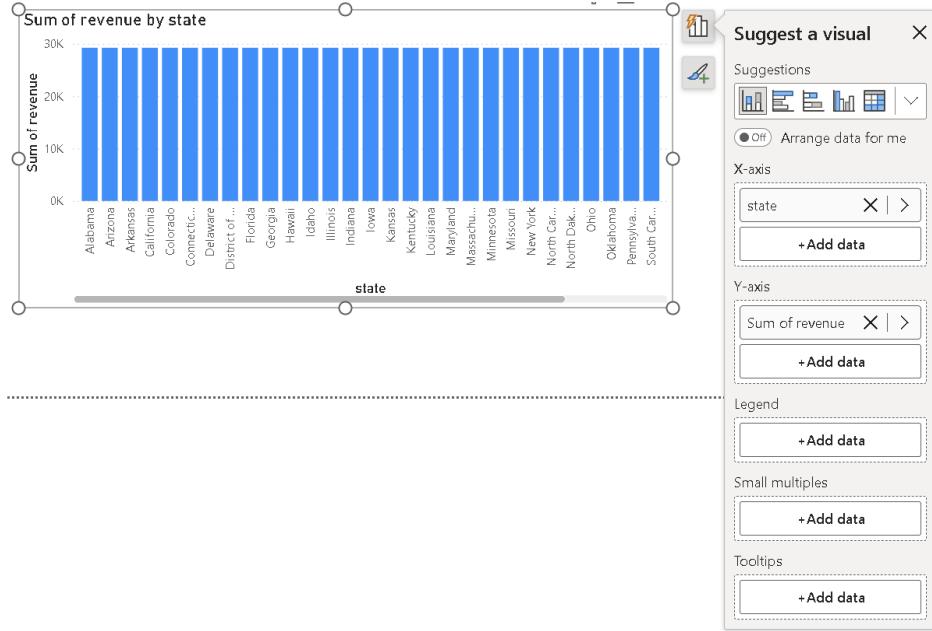
1. ***What is the total revenue in California?***
2. ***How many sales has the product "Namkeen - Rice Kodubale (Sln J)"?***
3. ***What is the average price in Dallas?***

▼ Solution

Exercise 1: What is the Total Revenue in California?

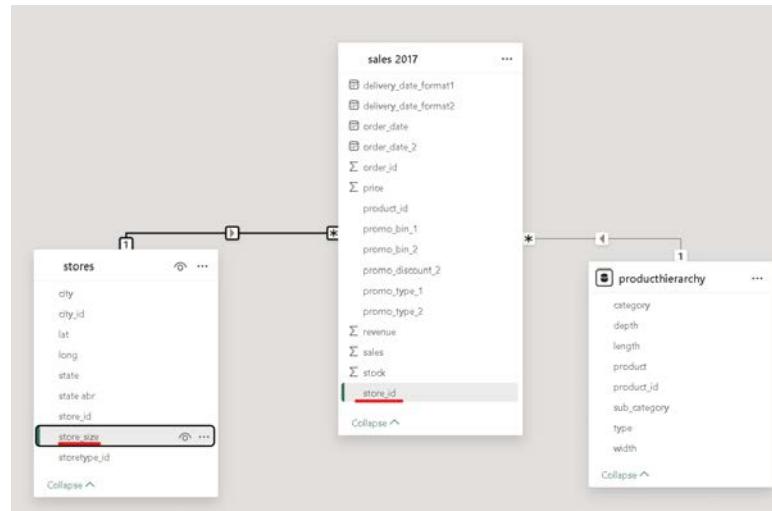
1. Set Up the Visual:

- Start by creating a new visual in Power BI. To see all available fields, clear any filters in the search bar on the right side.
- Drag and drop the **State** field from the **Stores** table onto the canvas. The default visual will be a table, but you can change this to a column or bar chart as needed.
- Next, drag the **Revenue** field from the **Sales** table into the Y-axis of the visual.

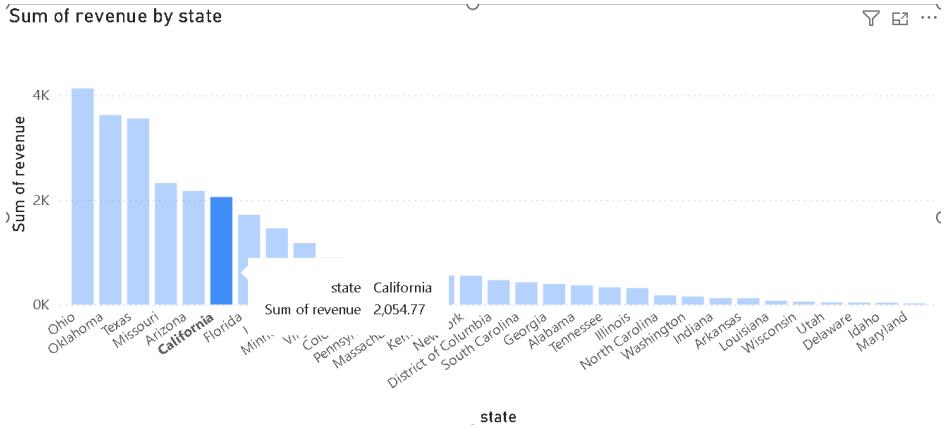


2. Addressing the Relationship Issue:

- If the visual does not display correctly, it's likely due to a missing or incorrect relationship between the tables. To fix this, go to the **Model View**.
- In **Model View**, establish a relationship between the **Store ID** in the **Sales** table and the **Store ID** in the **Stores** table.



- Once the relationship is established, return to the **Report View** to check the visual. You should now see the correct total revenue for California, which is \$2,054.



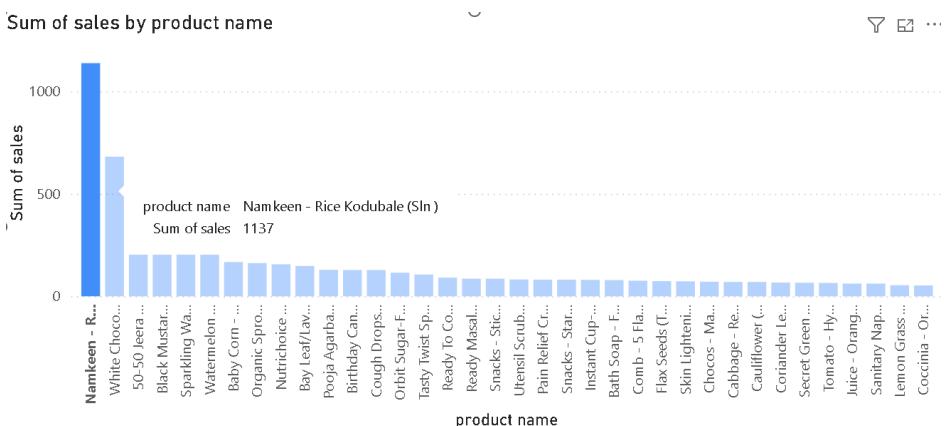
Exercise 2: How many sales has the product "Namkeen - Rice Kodubale (Sln)"?

1. Adjust the Visual:

- Use the same visual by clearing the existing data. Remove the previous fields by clicking the X next to them.
- Drag the **Product Name** field (which may be labeled differently, such as **Product(Brand)**) from the **Product Hierarchy** table to the X-axis. If necessary, rename the column to something more descriptive directly in the Report View by double-clicking the field name.
- Drag the **Sales** field from the **Sales** table to the Y-axis.

2. Result:

- The visual should now display the total sales for the product '**Namkeen**', which is 1,137 units.



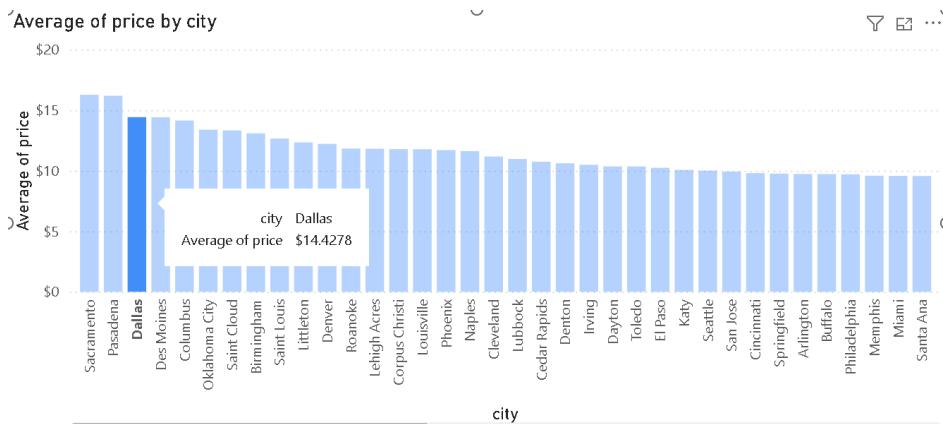
Exercise 3: What is the Average Price in Dallas?

1. Create the Visual:

- Clear the previous data from the visual.
- Drag the **City** field from the **Stores** table to the X-axis and the **Price** field from the **Sales** table to the Y-axis.

2. Change the Aggregation Method:

- By default, the visual will show the sum of prices. To find the average price, change the aggregation method.
- Click on the field options, select **Average**, and the visual will update to show the average price.
- For Dallas, the average price should be \$14.42.



This practice exercise helps reinforce the importance of data cleaning and establishing correct relationships in Power BI to derive meaningful insights. In the next section, we'll take a closer look at stacked column charts within the Report View.

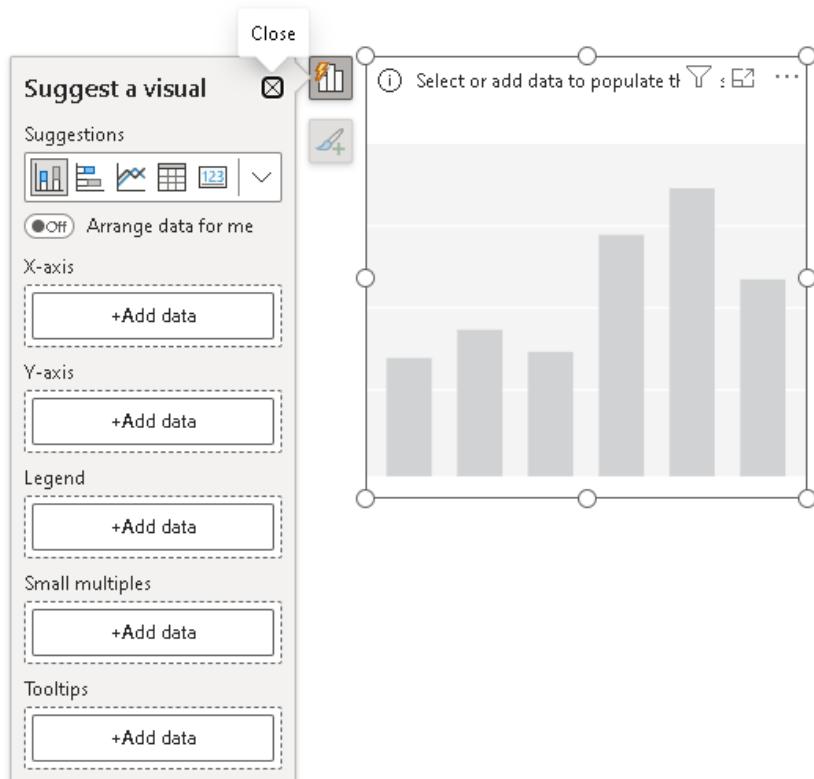
▼ Stacked Column Chart

The stacked column chart is one of the most commonly used visuals in Power BI, offering a clear way to compare data across multiple categories. Let's explore how to effectively use and customize this visual.

Understanding the Stacked Column Chart:

- **Visual Components:**

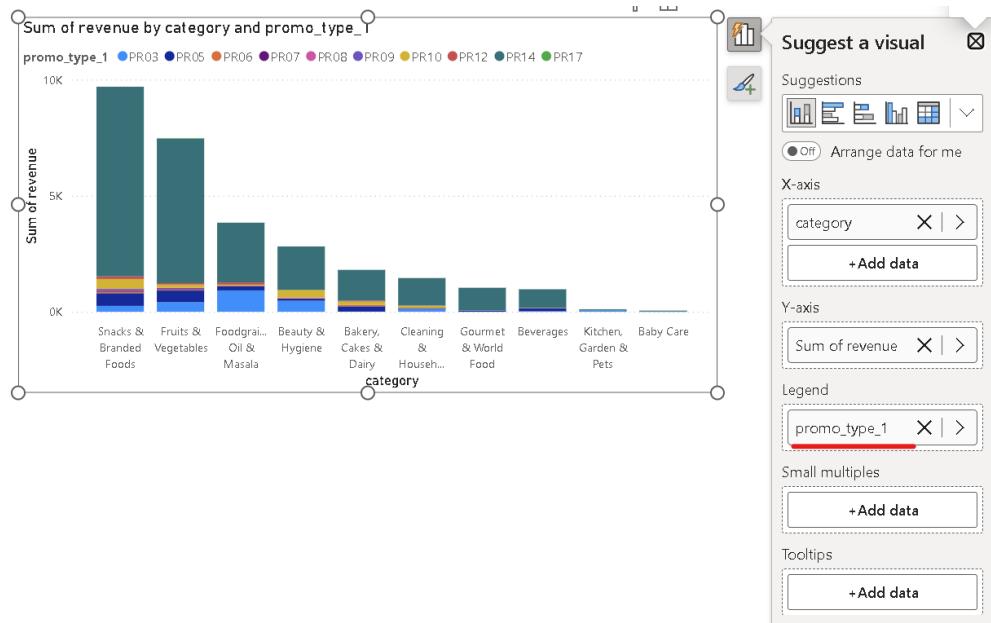
- The chart consists of an **X-axis**, a **Y-axis**, and a **Legend**. The X-axis typically represents categories (e.g., product categories), while the Y-axis represents numerical values (e.g., revenue).
- The Legend allows you to compare additional dimensions within the same visual, such as promo types within product categories.



- **Adding Data:**

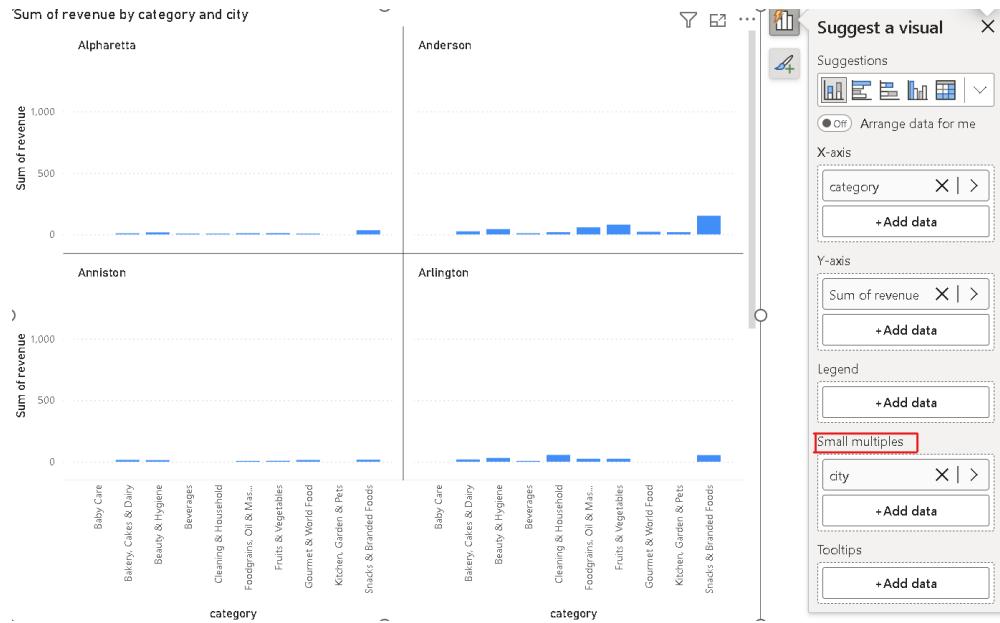
- To populate the chart, you can drag and drop fields into the X-axis, Y-axis, and Legend areas. For example, adding "Promo Type" to the Legend splits the data within each category, allowing for a detailed comparison across promotional efforts.

- The stacked column chart is particularly useful when you want to visualize the total values and their breakdowns by different subcategories.



- Using Small Multiples:**

- Small multiples allow you to create separate charts within the same visual for different categories, such as different cities. However, be cautious as this can make the chart cluttered and less user-friendly. Adjust the visual's size to ensure clarity.

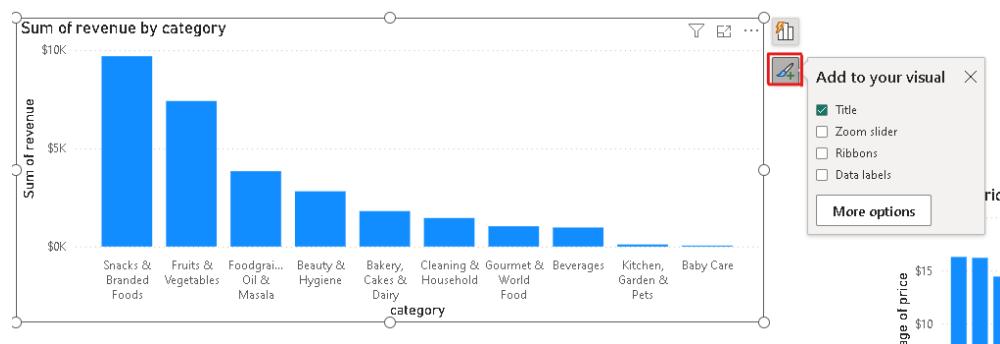


▼ The Format Pane

When building reports in Power BI, it is crucial to format visuals to meet specific needs rather than relying solely on default settings. Proper formatting ensures that your visuals are both visually appealing and informative. In this section, we'll explore how to use the Format Pane to customize visuals in Power BI.

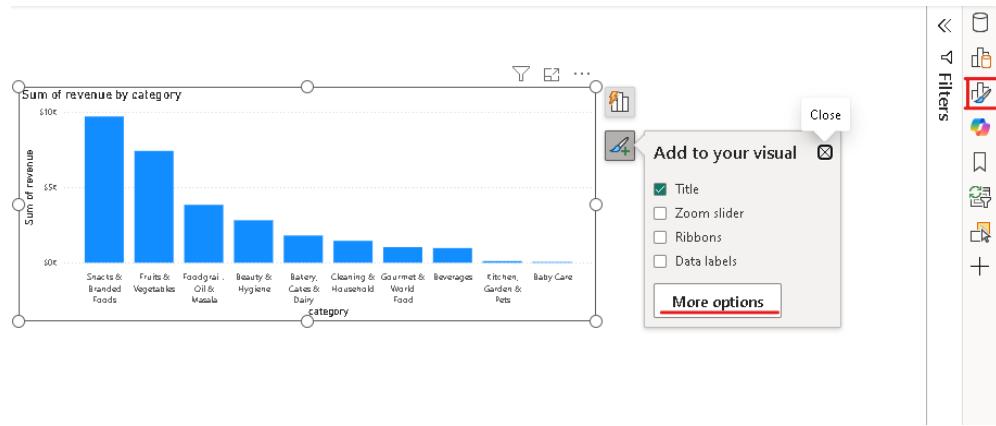
Accessing the Format Pane:

- **Basic Adjustments:**
 - After selecting a visual, you can quickly add or remove elements such as titles, axes, and data labels. These basic options allow for quick modifications, but for more detailed customization, you'll need to use the Format Pane.

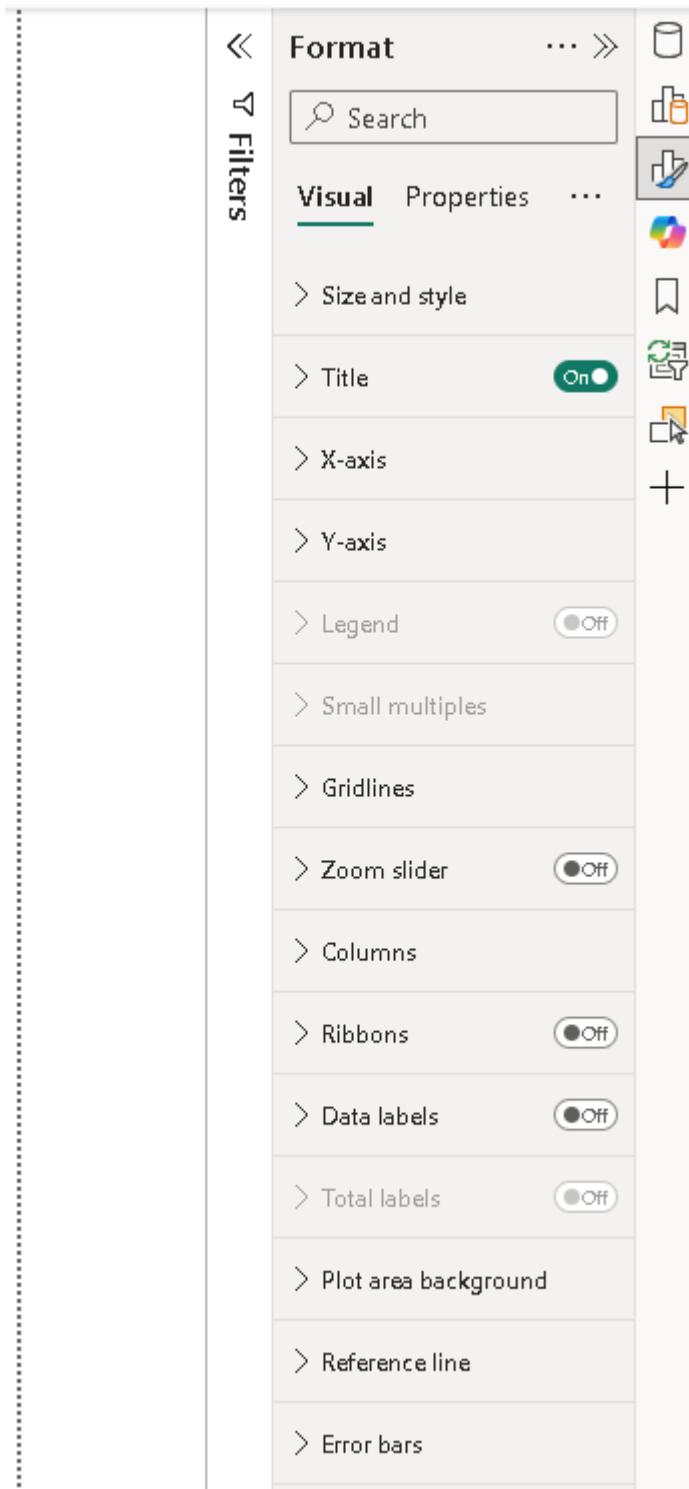


- **Opening the Format Pane:**

- To access the Format Pane, select the image, click the formatting icon and select "**More Options**" or add the pane using the pane switcher on the right side of the screen.



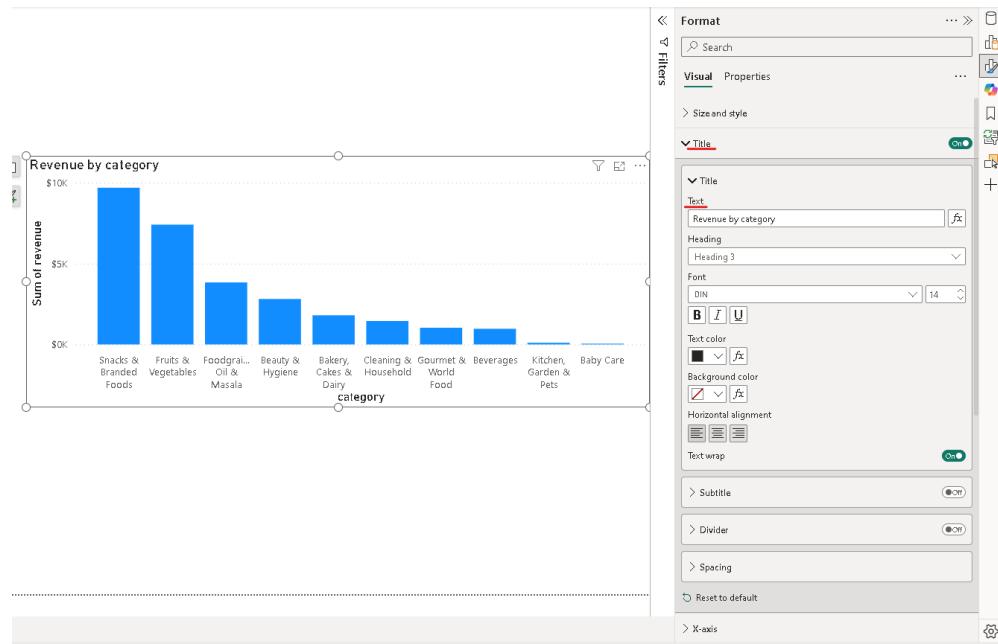
- Once the Format Pane is open, ensure the visual is selected so you can customize it according to your preferences.



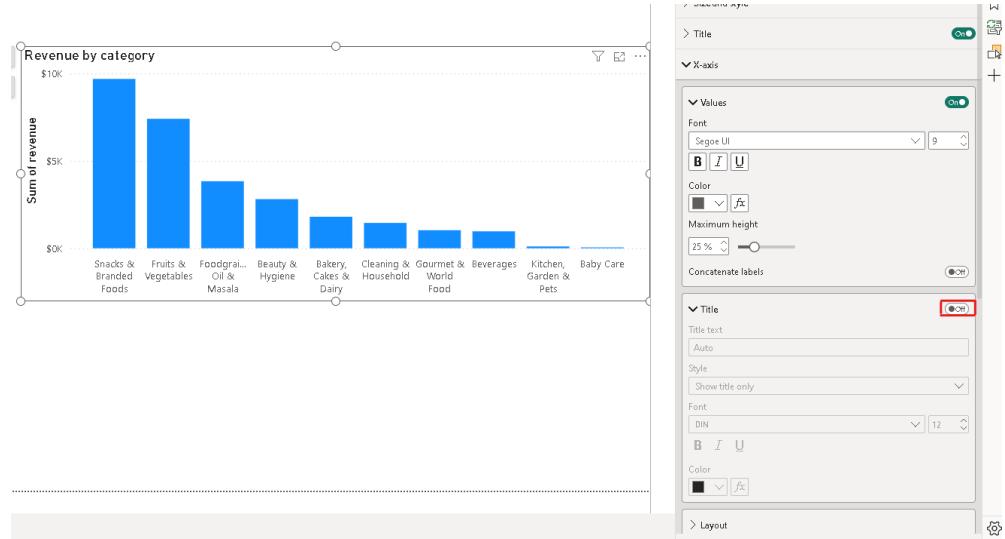
Customizing Visuals:

- **Titles and Labels:**

- You can modify the title text, style, size, and alignment. For example, instead of the default "**Sum of Revenue by Category**" you might shorten it to "**Revenue by Category**" and apply a different font or size to match your report's style. If you want to customize it further, you can change the **heading, font, text color**, etc. that you see below.



- Axes labels can also be customized or removed entirely to create a cleaner look. For instance, if the X-axis labels are self-explanatory, you might choose to hide the axis title to reduce clutter.

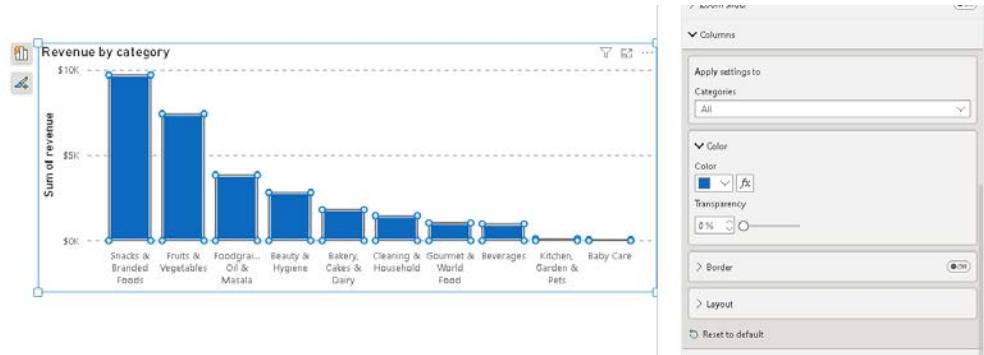


- **Gridlines and Colors:**

- Gridlines can be adjusted to different styles, such as dashed or solid, and their visibility can be enhanced by increasing line thickness.



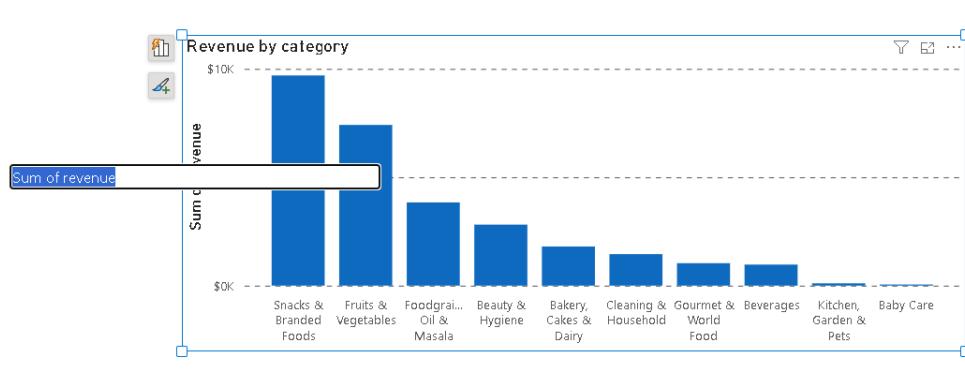
- Changing colors within the visual is also straightforward. Although the Format Pane allows for precise adjustments, Power BI's on-object interaction feature lets you double-click directly on elements within the visual (like bars or text) to quickly change their color or style.



Advanced Formatting:

- **On-Object Interaction:**

- This feature makes it easier to find and modify specific elements within a visual. For example, double-clicking on a data label or axis value will bring up relevant formatting options, allowing you to adjust display units, font size, or color directly.



- **Additional Customization:**

- The Format Pane offers more advanced options, such as adding reference lines or enabling data labels. While on-object interaction provides quick edits, the Format Pane gives access to options that may not be immediately visible.

The Format Pane in Power BI is a powerful tool for customizing visuals, allowing you to tailor your reports to specific requirements. While on-

object interaction offers a more intuitive and direct way to make changes, the Format Pane provides comprehensive control over the appearance and functionality of your visuals.

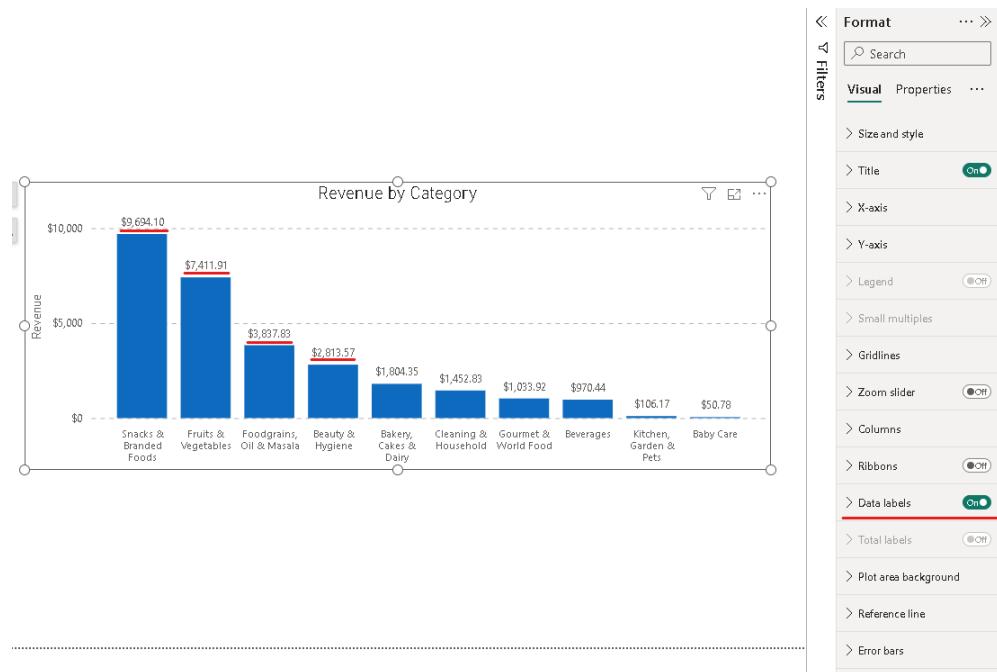
▼ Data Labels

Data labels are a crucial element in many Power BI visualizations, as they provide the exact values directly on the chart, eliminating the need for estimation. This feature enhances clarity and precision in data interpretation.

Enabling and Customizing Data Labels:

- **Activation:**

- Data labels can be turned on in the **Format Pane** for various visuals, such as stacked column charts, clustered bar charts, and pie charts (where they are referred to as "detail labels").

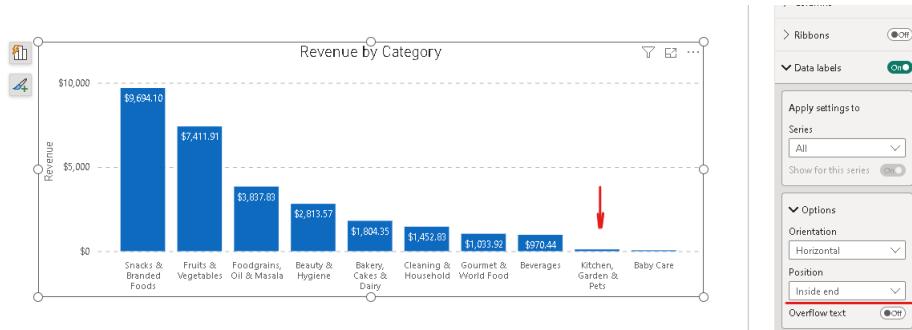


- **Positioning:**

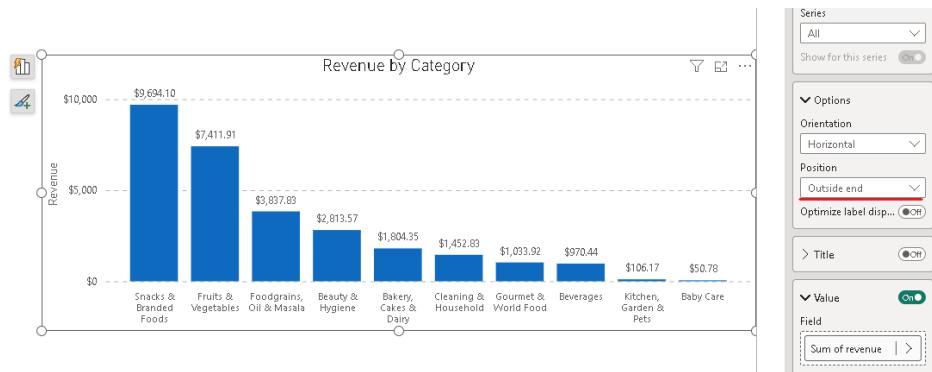
- The default setting for data label positioning is "Auto," which generally works well by automatically placing labels inside or

outside the chart elements based on available space. However, this can be manually adjusted:

- **Inside End:** Labels are placed inside the chart element, but may become unreadable if the element is too small.

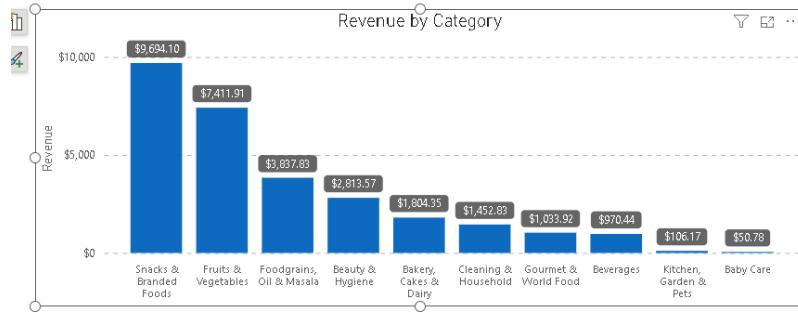
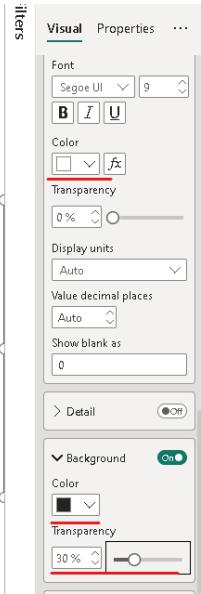


- **Outside End:** Labels are placed outside the chart element, which can be more readable, especially when combined with a background color.



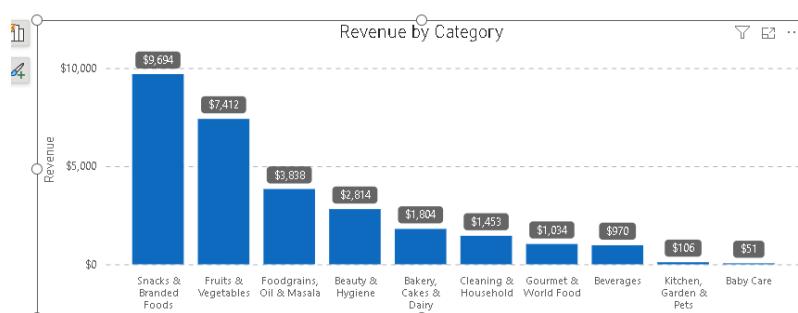
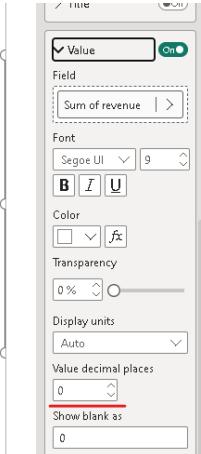
- **Improving Readability:**

- If the labels are hard to read, you can enhance visibility by turning on the background color and adjusting its transparency.
- The label text color can also be customized, such as changing it to white for better contrast.



- **Decimal Places:**

- You can control the number of decimal places displayed in the labels. Reducing decimal points to zero can create a cleaner, more professional look.



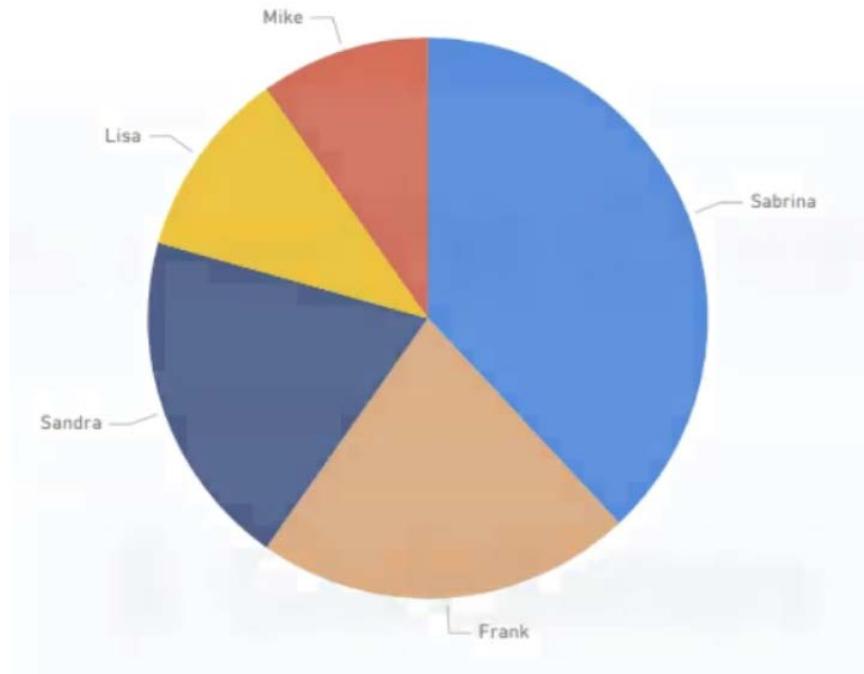
Data labels are a powerful tool in enhancing the readability and precision of your Power BI visuals. While they are available across various visual types, careful customization ensures they are effectively integrated into your reports.

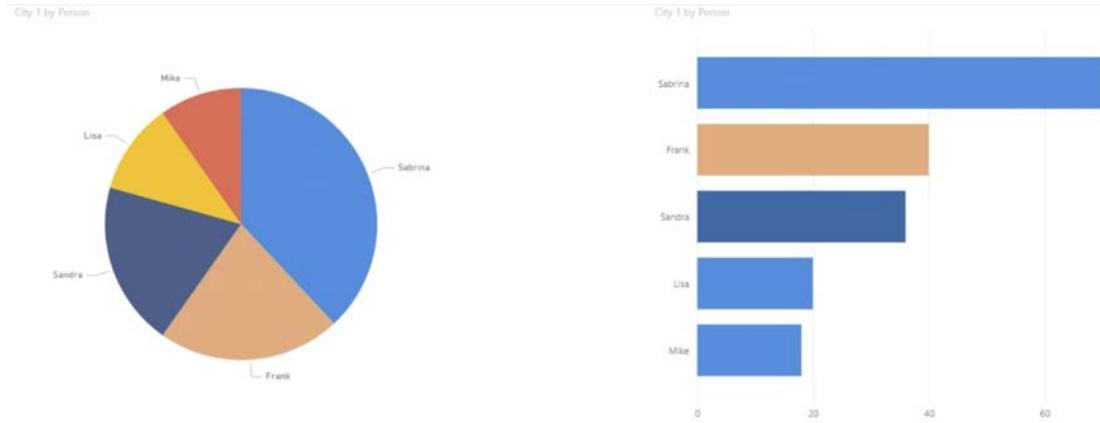
▼ Pie Chart - Guidelines

Should you use pie charts or not? This is a highly debated topic in data visualization. While pie charts may seem straightforward, they can often lead to misinterpretation and are generally less effective than other visualizations. Some experts even recommend avoiding pie charts entirely. However, when used correctly and in the right context, pie charts can be a useful tool. In this section, we'll explore why pie charts are controversial and provide guidelines on how to use them effectively.

Why Are Pie Charts Controversial?

The main issue with pie charts is that they are difficult for the human brain to interpret accurately. For instance, comparing the sizes of slices in a pie chart can be challenging. Imagine trying to determine who has a larger share in a pie chart: Sandra or Frank, Lisa or Mike? It's much harder to gauge than if the same data were displayed in a bar or column chart. This difficulty is amplified when comparing multiple pie charts side by side.

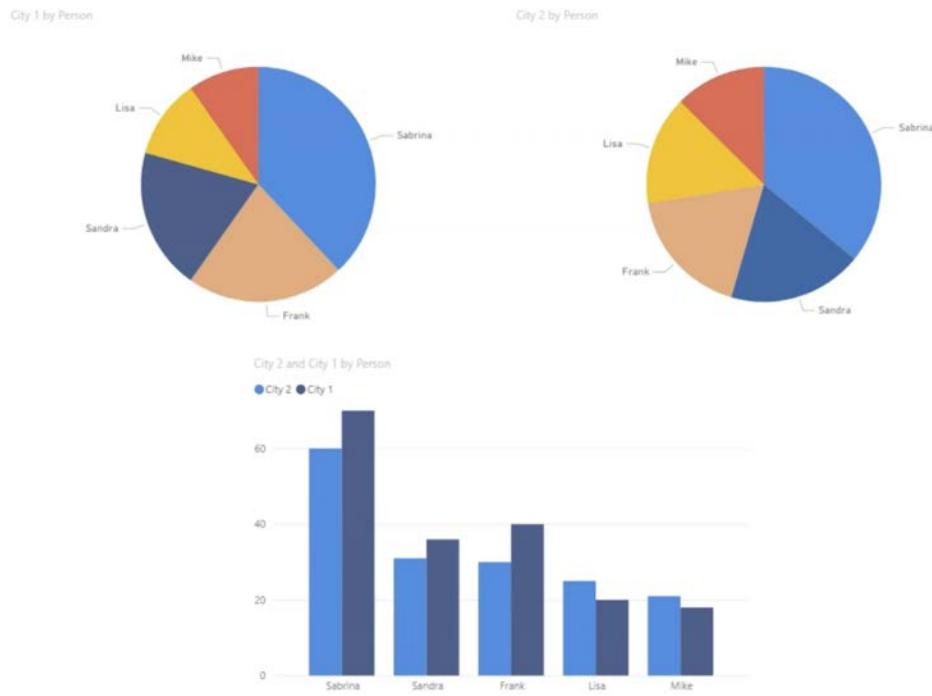




Guidelines for Using Pie Charts:

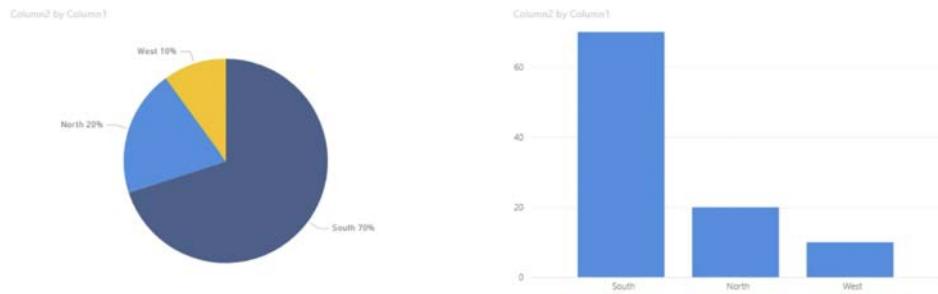
1. Avoid Comparing Similar Values:

- Pie charts are not suitable for comparing values, especially when those values are close in size. For such comparisons, bar charts or column charts are far more effective. Comparing values across multiple pie charts is even more problematic and should be avoided altogether.



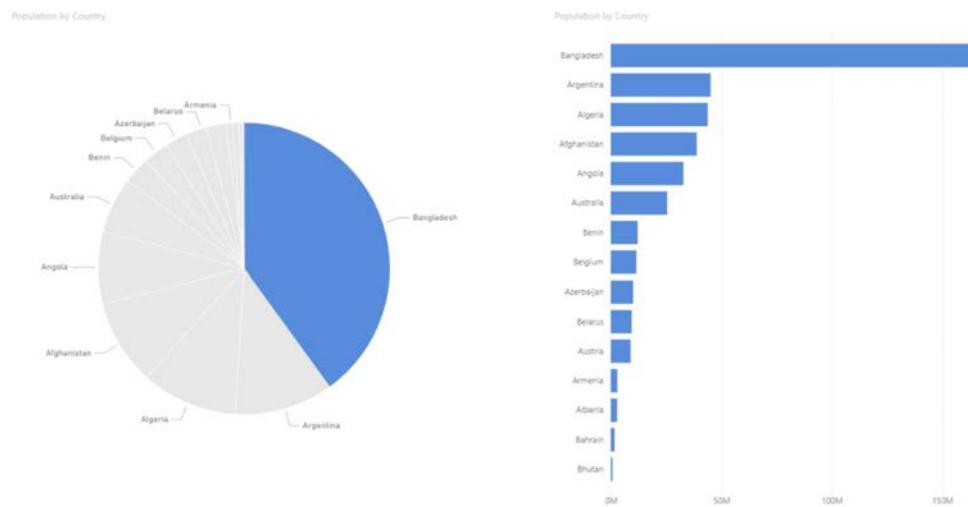
2. Use Pie Charts to Show Percentages of a Whole:

- The only situation where a pie chart is appropriate is when you want to display a percentage of a whole. Pie charts should only be used when the values add up to 100%, as the visual implicitly conveys this information. If the percentages don't total 100%, or if the chart is used to compare absolute values, a pie chart is not the right choice.



3. Limit the Number of Categories:

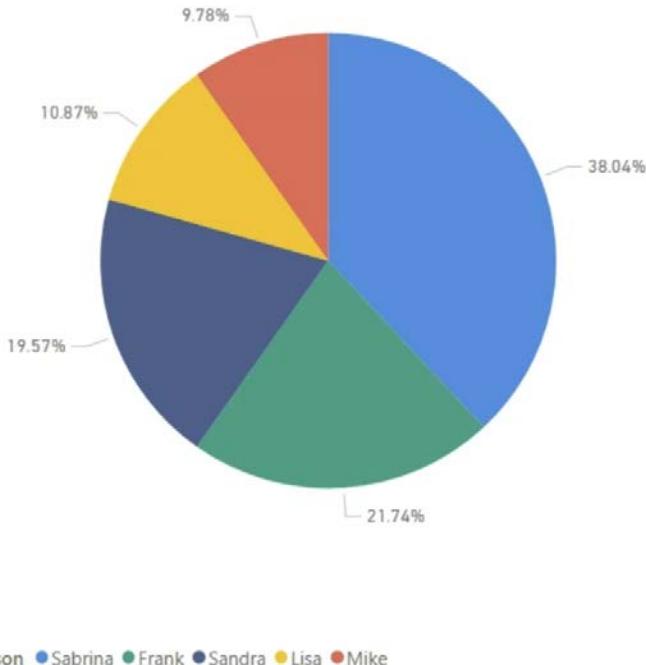
- Pie charts should be limited to two or three categories, with a maximum of five. More than five categories make the chart cluttered and difficult to interpret. In some cases, this rule can be bent if you are highlighting a single, dominant category, but this should be done with caution.



4. Avoid Using Legends:

- Legends in pie charts require the viewer to constantly look back and forth between the chart and the legend, which is inefficient and confusing. Instead, label the slices directly on the chart, and always include data labels to indicate exact values or percentages.

City 1 by Person



Pie charts should be used sparingly and with careful consideration. They are most effective when used to display simple percentages of a whole, with a limited number of categories and clear labels. Always consider whether a bar or column chart might be a better alternative before choosing to use a pie chart.

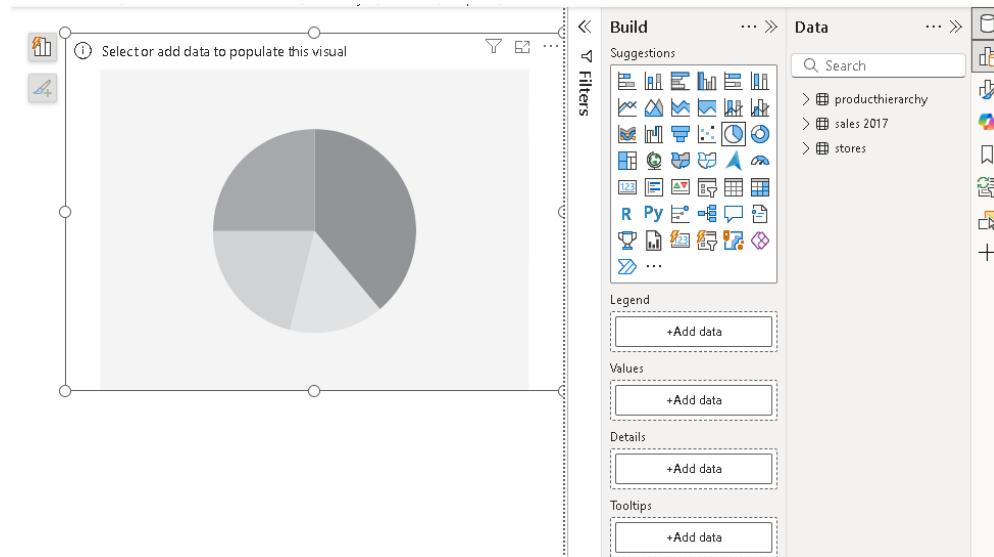
▼ Pie Chart

Let's explore how to create and customize a pie chart in Power BI.

Creating a Pie Chart:

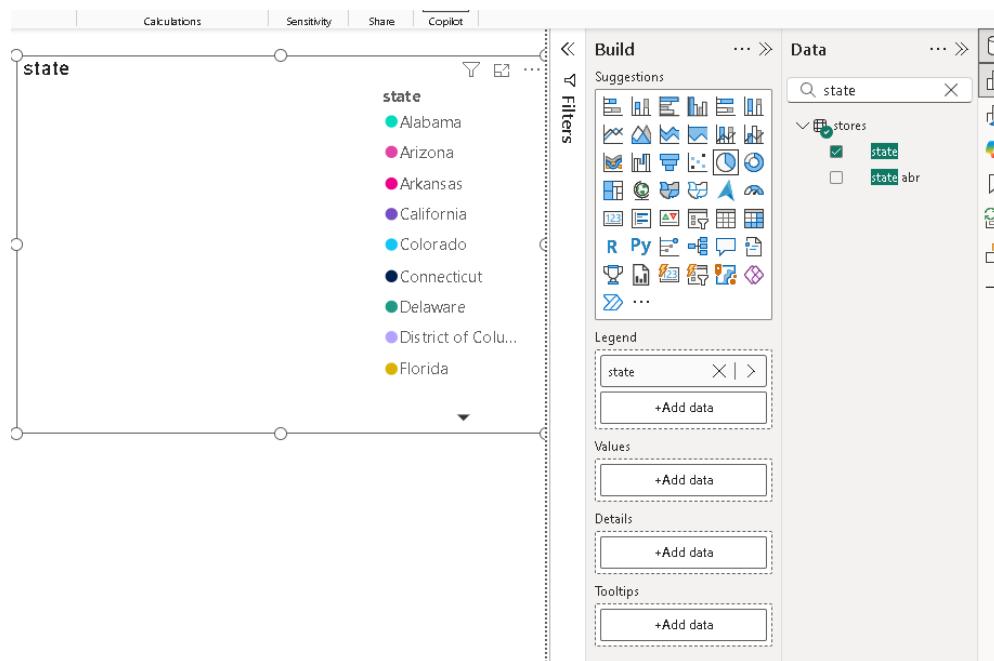
1. Inserting the Pie Chart:

- Start by clicking in an empty space on the canvas, then select the pie chart icon from the Home ribbon. The chart will appear on the canvas, ready to be populated with data.

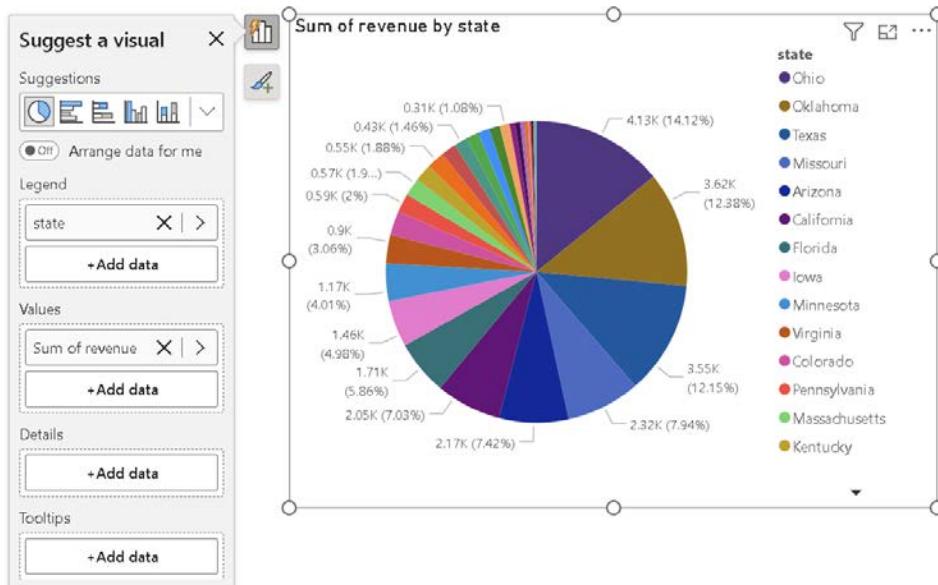


2. Adding Data:

- To populate the pie chart, drag and drop the **State** field from the **Stores** table into the visual. The State field will automatically populate the **Legend** section.



- Next, add **Revenue** from the **Sales** table. You can drag it into the **Values** section or directly into the visual, and it will be placed correctly.



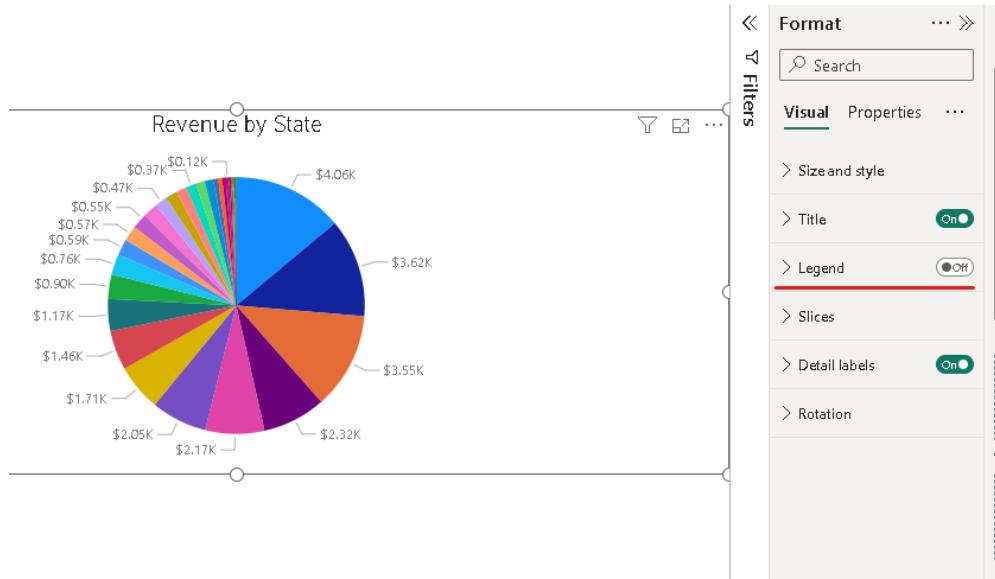
Customizing the Pie Chart:

1. Format Pane:

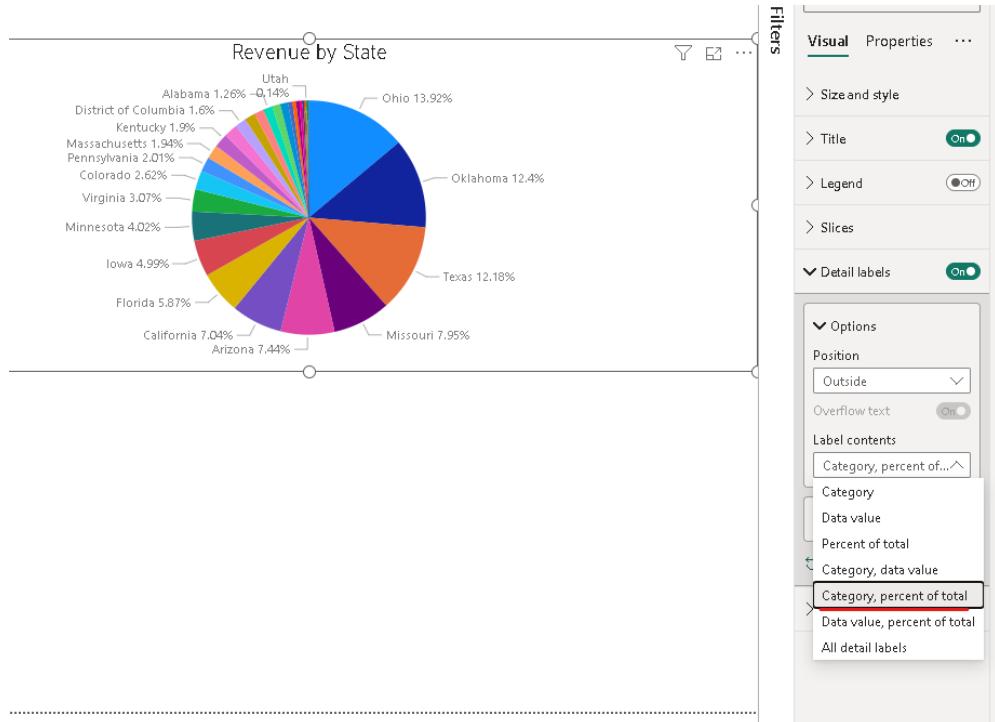
- Open the **Format Pane** to begin customizing the pie chart. Here, you can modify the title, adjust the legend, and tweak the data labels (referred to as "detail labels" in pie charts).

2. Improving Readability:

- **Legend:** While the legend is useful, it can make the chart harder to read. Users have to constantly move their eyes between the legend and the chart, which is inefficient. To improve readability and maximize space, consider turning off the legend.

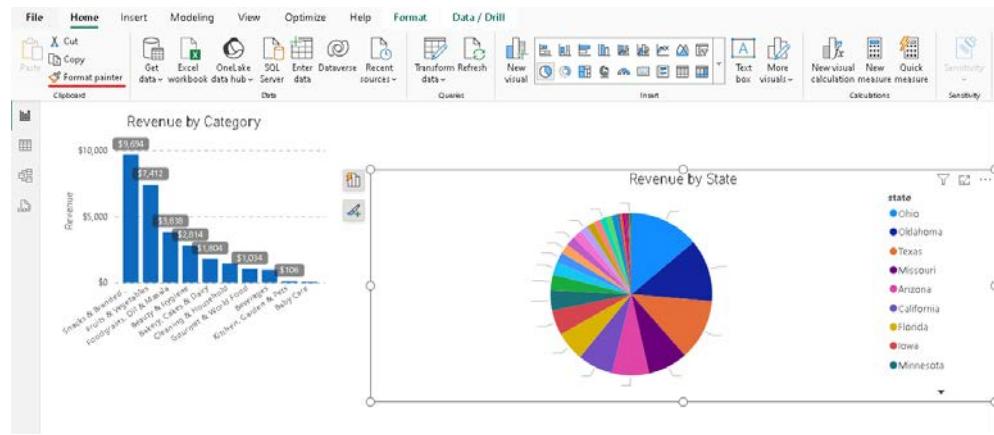


- Detail Labels:** Since the legend is turned off, use detail labels to display the necessary information directly on the chart. Expand the **Label Contents** section and select "Category and Percentage of Total" to show both the state name and its percentage of the total revenue.

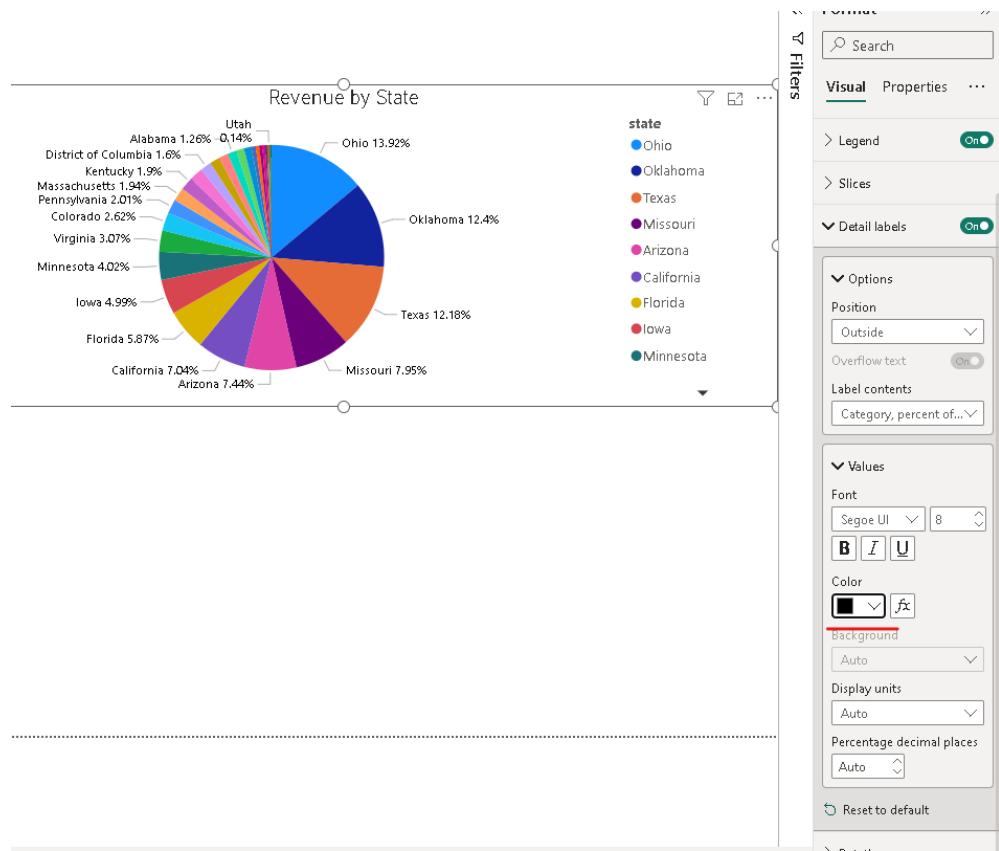


3. Formatting:

- You can use the **Format Painter** to quickly copy the formatting from another visual. Simply select the visual you want to copy from, click on the Format Painter, and then select the pie chart. This will apply the same formatting, including fonts and colors.

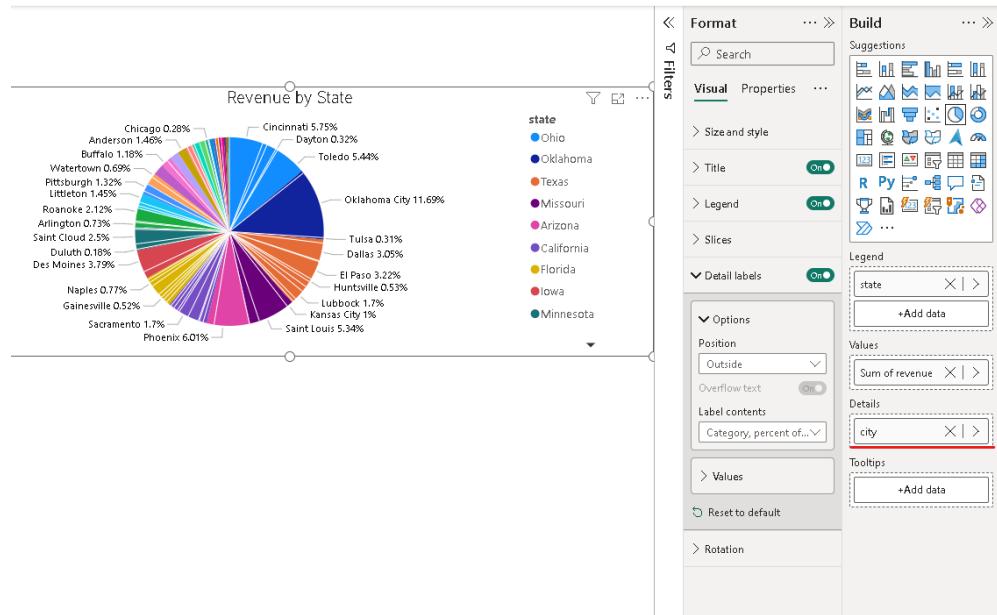


- If the labels disappear after applying the Format Painter, check the label color settings, as they may have defaulted to white or another hard-to-see color. Adjust the color to something more visible, like a darker shade.



4. Adding Detail Levels:

- If you want to break down the data further (e.g., by city), you can add an additional field to the **Details** section. However, be cautious, as adding too much detail can make the chart cluttered and difficult to interpret. For simplicity, it's often better to stick with a single level of detail.



Pie charts can be effective for showing the percentage of a total when used correctly. However, they should be used cautiously, especially when dealing with more than a few categories or multiple levels of detail. In many cases, a bar or column chart may be a better choice.

▼ Project - 2

[project2.rar](#)

▼ Chapter 3

▼ Append Queries

Today, we'll focus on an essential task: appending queries. Often, we have data across multiple files that need to be combined into a single table. For instance, we might have sales data for 2017 in one file and sales data for 2018 and 2019 in separate files. Our goal is to combine these into one table.

Here's how we can append queries:

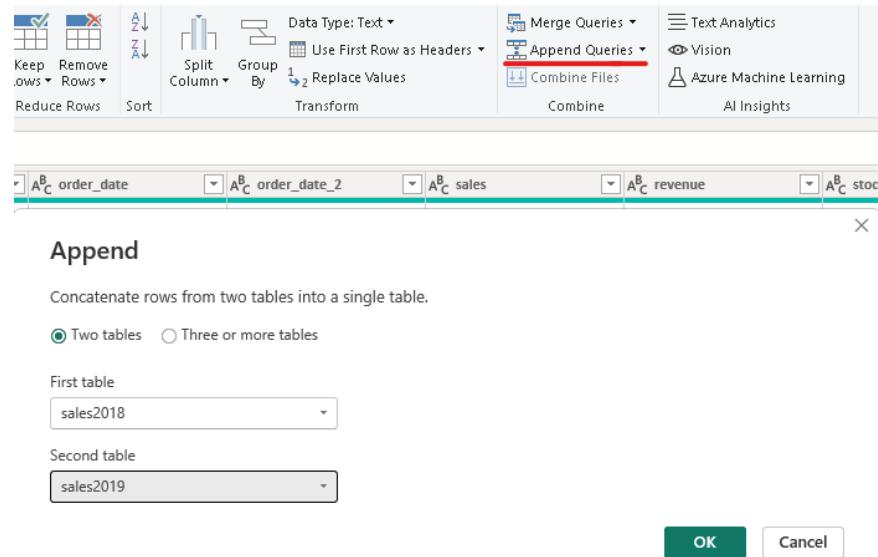
1. **Importing the Data:** First, we need to bring in the data. Go to **New Source** and select your file (e.g., a CSV file). As an example, start

with the 2018 file. Open the file, preview it, and ensure everything looks correct. Then, load and transform the data. First, we'll use the first row as headers. Repeat this process for the 2019 data. Again, load the data, transform it, and use the first row as headers.

2. **Appending Queries:** To append the data, select the first table (e.g., Sales 2017). On the **Home** ribbon, click **Append Queries**. There are two options:

- **Append Queries:** This will append the data to the currently selected query.
- **Append Queries as New:** This will append the data to a new query, which is useful if you want to keep the original queries separate.

Let's use **Append Queries as New**. First, select the 2018 table, then the 2019 table.



It's crucial that both tables have the same column names and structure, as this ensures the appending process works correctly. If the column names don't match (e.g., "Order ID" vs. "Order ID Two"), Power BI will create separate columns for each.

For example:

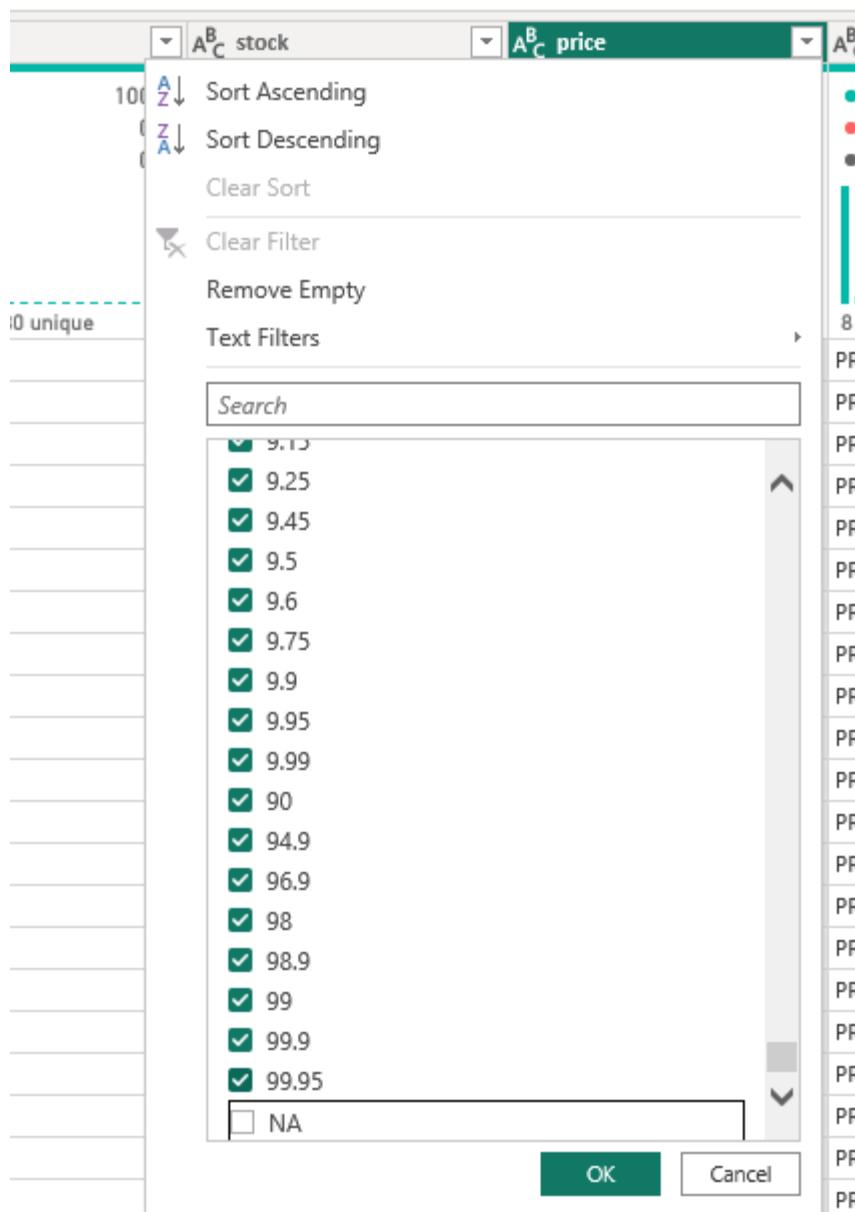
- Let's say we selected sales, sales2017 and sales2019 to use the append queries function. Let's assume that order_id for "sales" is order_id2 instead of order_id.

The screenshot shows the Power BI desktop interface with the 'Append' dialog open. The 'Tables to append' section contains 'sales', 'sales2017', and 'sales2019'. The 'OK' button is highlighted.

- If we examine the table we obtained, we see that there are 2 columns named order_id and order_id2, and this is a situation that we do not want.

The screenshot shows the Power BI desktop interface with the 'Reorder Columns' dialog open. The 'Source' table has columns 'order_id', 'order_id2', 'product_id', and 'store_id'. The 'Reordered Columns' section lists 'order_id', 'order_id2', 'product_id', and 'store_id'. The 'OK' button is highlighted.

- 3. Adjusting Data Types and Column Names:** After appending, you may need to adjust the data types or column names. You can select multiple columns by holding down **Shift** and then change their data types using the **Transform** tab.
- 4. Handling Errors and Inconsistencies:** If there are errors (e.g., missing or incorrect values like “NA” in the price column), you can filter or replace these values. For instance, you might want to filter out rows with “NA” in the price column.



5. **Finalizing the Append:** Once all tables are appended and transformed, ensure the column names and data types are correct. If you find any mismatched column names during the append process, like "Order ID" being split into two columns, you can fix this by renaming or adjusting the original tables before appending them.
6. **Cleaning Up Unused Tables:** After appending the queries, you may no longer need the original individual tables. You can organize your queries for a cleaner and more efficient workspace by grouping or removing unnecessary tables.

▼ Merging and Grouping Queries

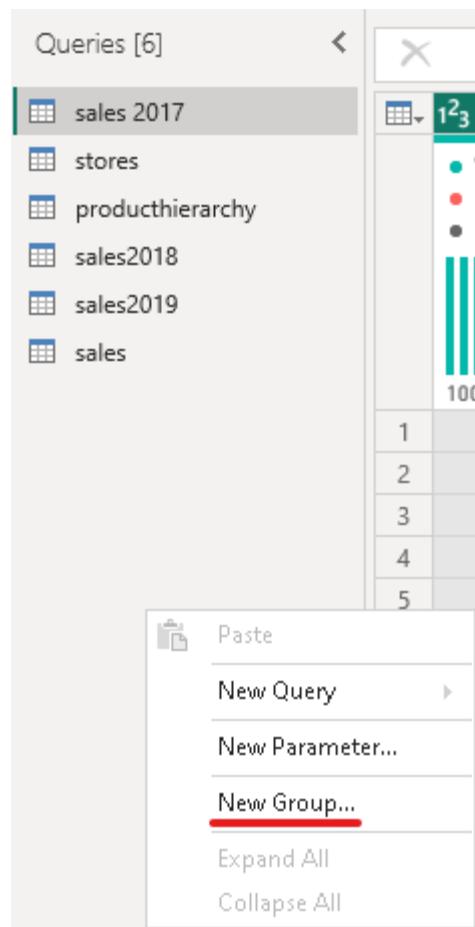
In Power BI, managing multiple queries efficiently is crucial, especially as your project grows. Two key features that help with this are **grouping** queries and **merging** columns.

Grouping Queries

As you work on a project, you may accumulate many queries, some of which may no longer be needed. To keep your workspace organized, you can group these queries:

1. Creating a Group:

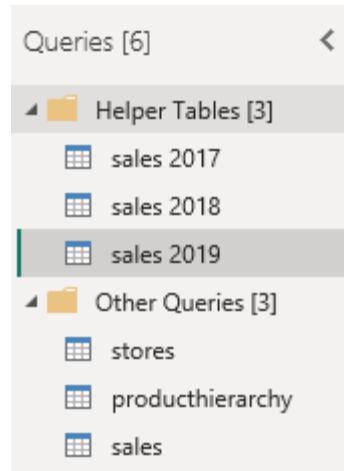
- Right-click in the Queries pane and select **New Group**.



- Assign a name to the group, such as "Helper Tables" for queries you don't need to see regularly.
- Optionally, add a description, then click **OK**.

2. Organizing Queries:

- Drag and drop the queries you want to group into the newly created folder. This keeps your workspace clean and makes it easier to focus on the queries that matter.



Merging Columns

Merging columns is another useful feature, especially when you need to combine split data back into a single column:

1. Selecting Columns to Merge:

- Hold the **Control** key and select the columns you want to merge.
- Right-click and choose **Merge Columns**.

The screenshot shows the Power BI Data View with two columns: 'state abr' and 'state'. The 'state abr' column contains a legend for 'Valid' (green), 'Error' (red), and 'Empty' (grey). Below the legend, it says '35 distinct, 10 unique'. The 'state' column lists state abbreviations and their corresponding names. A context menu is open over the 'state abr' column, with the 'Merge Columns' option highlighted. Other options in the menu include Copy, Remove Columns, Remove Other Columns, Add Column From Examples..., Remove Duplicates, Remove Errors, Replace Values..., Fill, Change Type, Transform, Group By..., Unpivot Columns, Unpivot Other Columns, Unpivot Only Selected Columns, and Move.

A ^B C state abr	A ^B C state
● Valid ● Error ● Empty	%100 %0 %0
	● Valid ● Error ● Empty
35 distinct, 10 unique	35 distinct, 10 un
AR	Arkansas
TX	Texas
NC	North Carolina
CA	California
TX	Texas
OK	Oklahoma
OH	Ohio
FL	Florida
DE	Delaware
GA	Georgia
CO	Colorado

2. Configuring the Merge:

- Choose a separator, such as a colon, hyphen, or custom symbol.
- Rename the new column, for example, "State / State ABR."

The screenshot shows a Power BI data view with four columns: city_id, state / state ABR, city, and lat. Each column has a histogram and summary statistics below it. The histograms show the distribution of values across three categories: Valid (green), Error (red), and Empty (grey).

Column	Distinct Values	Unique Values
city_id	37	15
state / state ABR	35	10
city	100	72
lat	125	110

Merge Columns

Choose how to merge the selected columns.

Separator: --Custom--

New column name (optional): state / state ABR

OK Cancel

- Click **OK** to merge the columns into a single one.

3. Adjusting as Needed:

- If the merge doesn't produce the desired result, you can easily undo the step or modify the settings.

▼ Hiding Tables

To enhance the user experience in Power BI, it's often beneficial to hide certain tables from the report view, especially if they are not directly relevant to the end user or the report developer. This reduces confusion and helps users focus on the essential data.

Why Hide Tables?

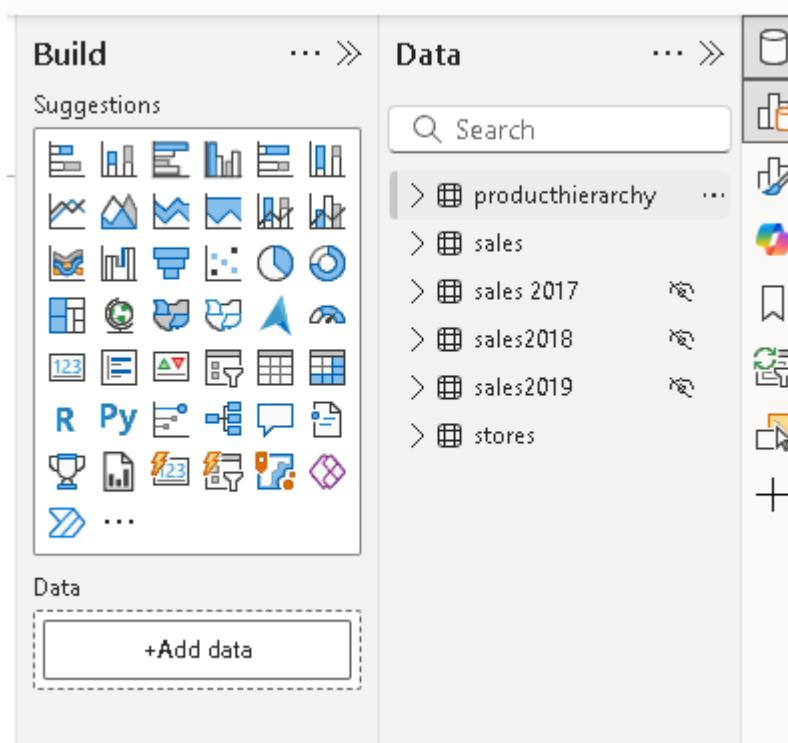
When working with multiple tables, some may serve as intermediate or helper tables that aren't needed in the final report. Leaving these visible can clutter the workspace, leading to confusion about which tables are important. By hiding these tables, you streamline the interface, making it easier for users to navigate and work with the necessary data.

Grouping Tables (Optional):

- As discussed in previous sections, you may group non-essential tables under a specific group like "Helper Tables" to organize them. However, these tables will still appear in the report view and data pane.

Hiding the Tables:

- To hide a table, simply right-click on the table in the Fields pane and select **Hide**. The table will be hidden from the report view.
- Repeat this for any other tables you want to hide.



Improving User Experience:

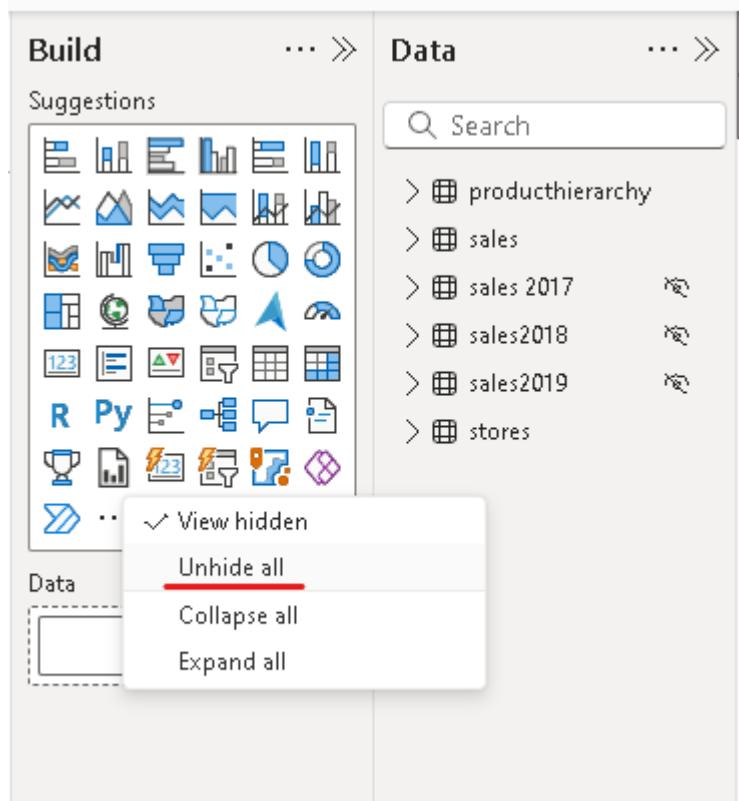
- After hiding irrelevant tables, the report view becomes cleaner, and users can focus on the tables that are truly necessary. This simple action greatly improves the user experience by reducing potential confusion.

Unhiding Tables

If you need to unhide a table:

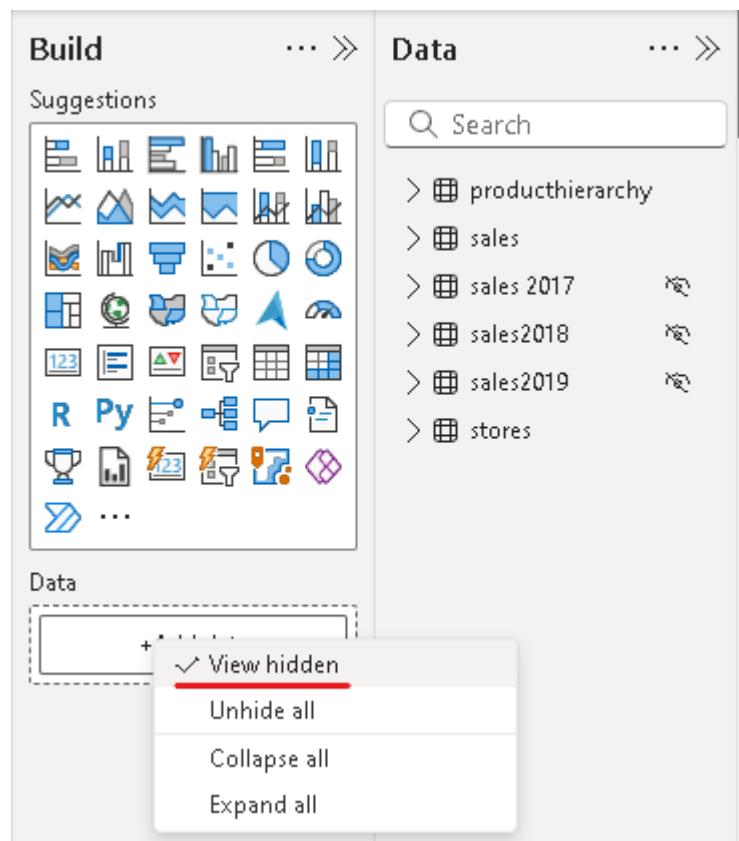
1. Unhide All Tables:

- Right-click in the Fields pane and select **Unhide All**. This will make all hidden tables visible again.



2. View Hidden Tables:

- Alternatively, you can enable the **View Hidden** option. Hidden tables will reappear, marked with a special icon indicating their hidden status. You can then selectively unhide tables as needed.



Publishing Considerations

When you publish the report, any tables you've hidden will remain hidden in the published version. This ensures that only the necessary data is visible to the end users.

▼ Date Hierarchies

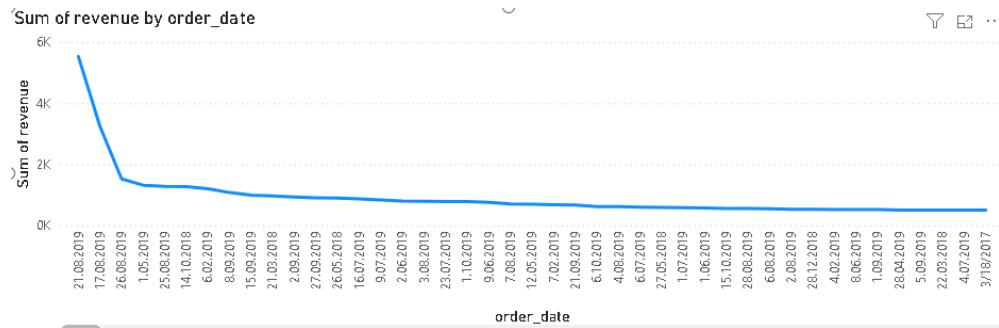
In Power BI, data hierarchies, particularly with date fields, provide a powerful way to visualize and navigate your data. Hierarchies allow you to drill down into different levels of detail, making it easier to analyze trends over time or across different categories.

Understanding Date Hierarchies

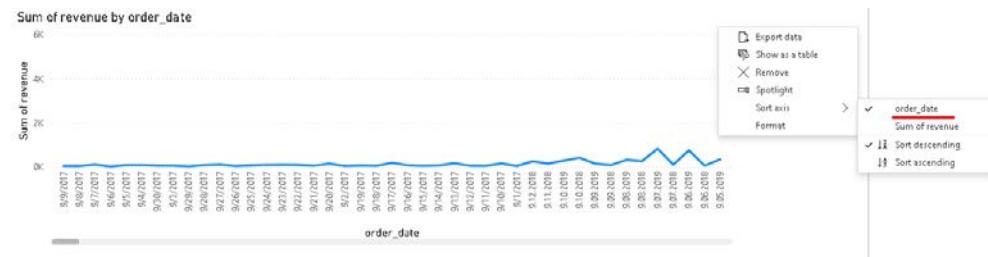
Dates are a unique data type in Power BI, automatically generating a hierarchy that includes levels such as Year, Quarter, Month, and Day. This hierarchy is extremely useful when analyzing how metrics like revenue develop over time.

1. Visualizing Data with Date Hierarchies:

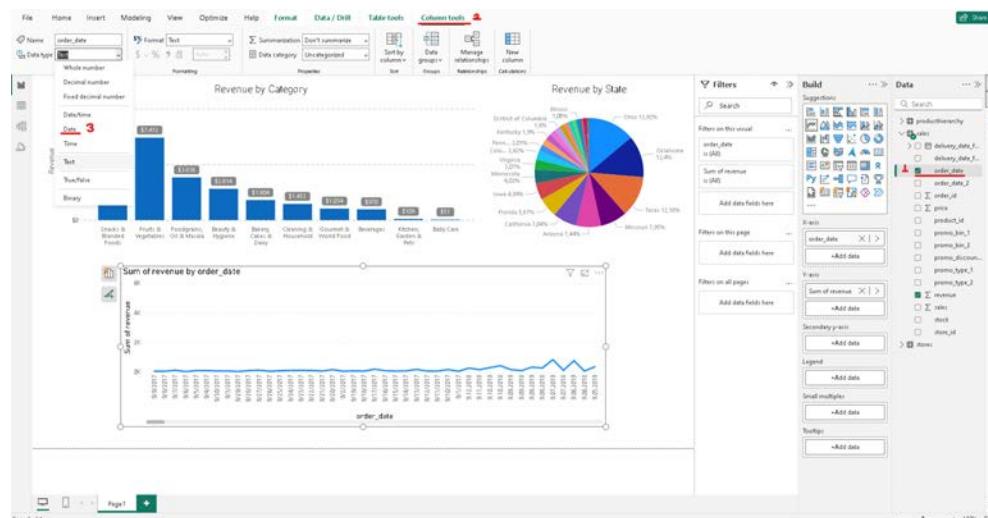
- To visualize data over time, start by dragging the relevant metric, like **revenue**, into your visual. Next, drag the **order date** field into the axis.

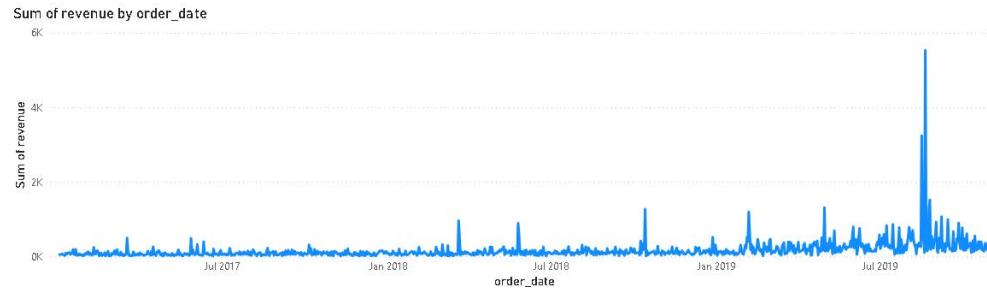


- By default, Power BI sorts this data alphabetically if the date column is set as text.



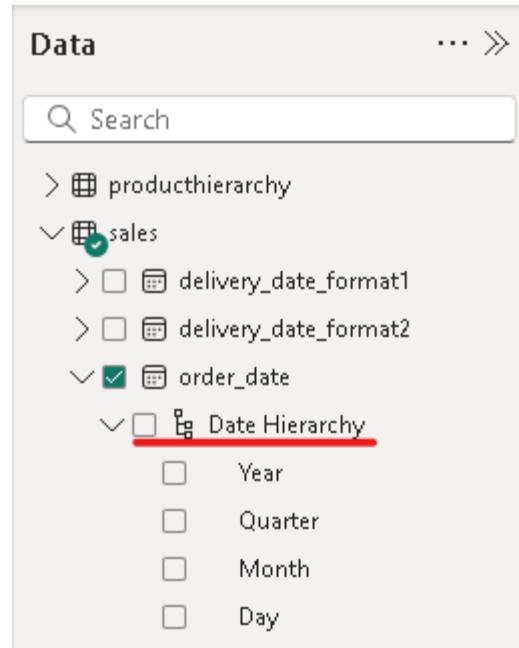
- To correct this, ensure the **order date** column's data type is set to **Date**. You can do this directly in the report view under **Column Tools**.



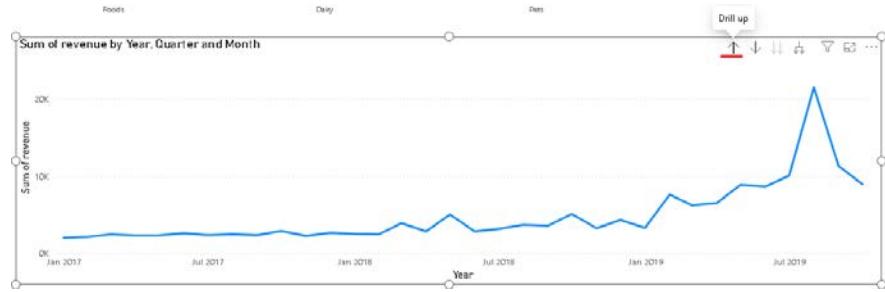


2. Exploring the Date Hierarchy:

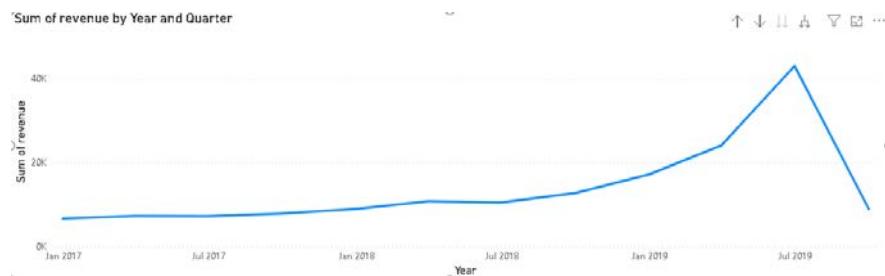
- Once the data type is correctly set, Power BI automatically creates a hierarchy. This hierarchy allows you to drill down from Year to Quarter, Month, and Day within your visual.



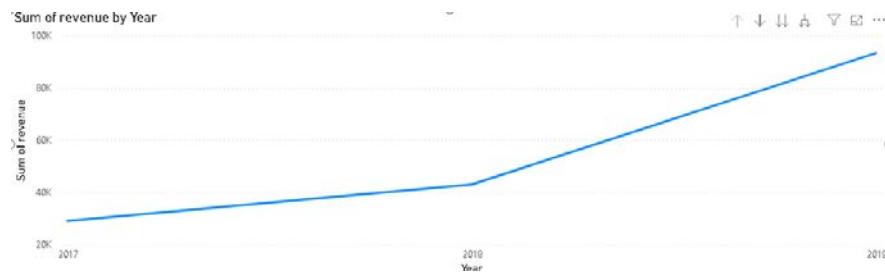
- The hierarchy provides several controls:
 - Drill Up:**
 - Performing a single drill up moves the view from the Day level to the Month level.



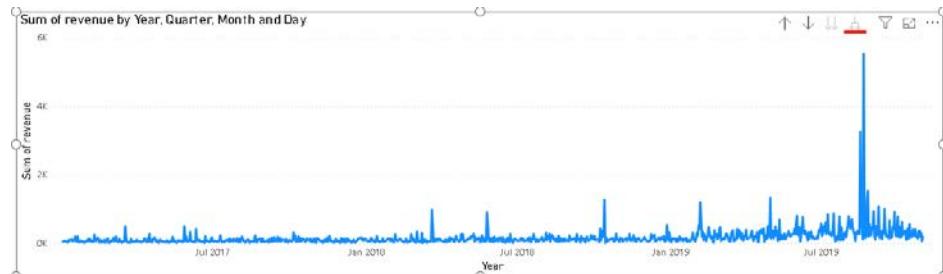
- A second drill up transitions from the Month level to the Quarter level.



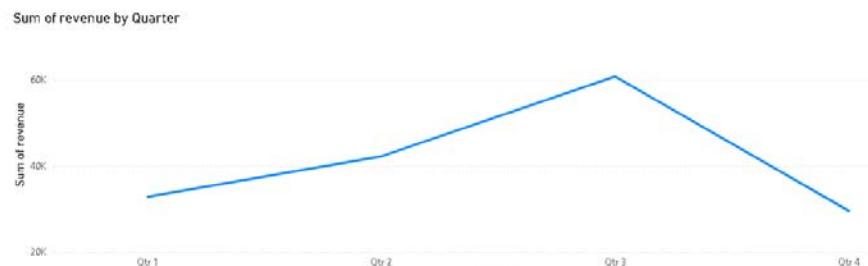
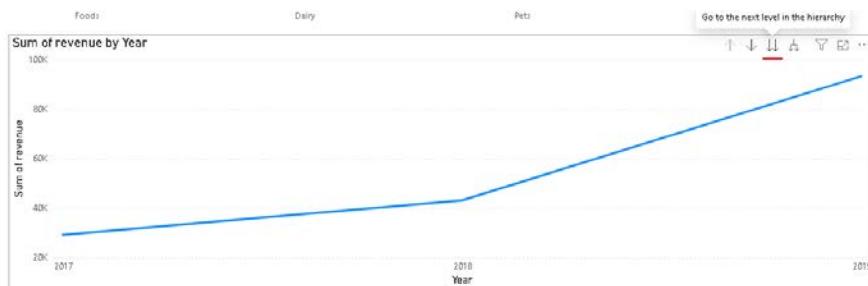
- A third drill up advances from the Quarter level to the Year level.



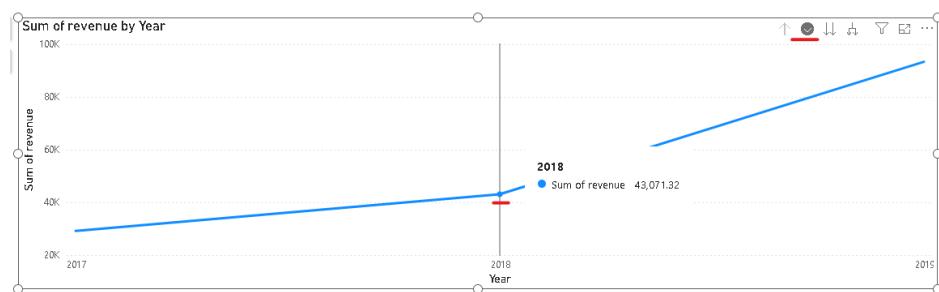
- **Expand All:** **Expand All** allows you to expand the hierarchy back through each level, sequentially displaying **Year → Quarter → Month → Day**.

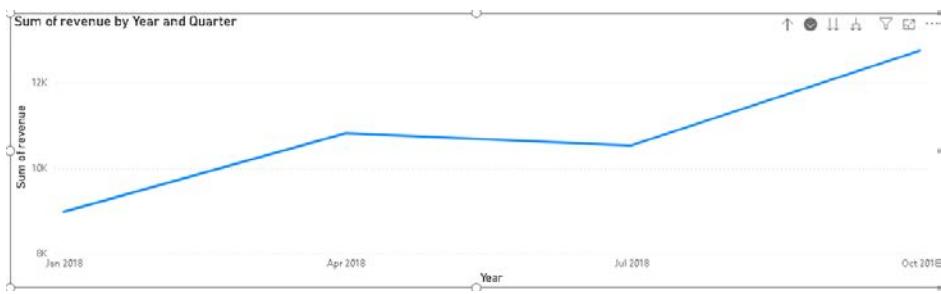


- “**Go to the next level hierarchy**” Mode: Focus on a specific element within the hierarchy, like expanding into a particular year to explore its quarters and months.
 - We move from years to quarters and now we see the quarters of all those years.



- **Drill mode on:** If we activate this, we can now select one element, for example, 2018, and then we will expand into this specific element, so into this specific year.





3. Using Hierarchies in Visuals:

- You can use the entire hierarchy in a visual or select specific levels. For example, you might choose to visualize revenue by Year, or drill down to see it by Month.
- The hierarchy controls in Power BI visuals allow you to easily navigate between different levels of data, providing a more detailed or high-level view as needed.

4. Advantages of Data Hierarchies:

- **Flexible Analysis:** Quickly move between different levels of data, such as seeing annual trends or zooming in on specific months.
- **Enhanced Visuals:** Hierarchies help create dynamic visuals that respond to user interaction, making reports more insightful and engaging.

▼ Drill Down Hierarchy

In our previous discussions, we explored date hierarchies in Power BI. However, the concept of hierarchies and the drill-down functionality extends beyond just dates; it can be applied to various other data fields and visuals. In this section, we will delve into how to effectively create and utilize drill-down hierarchies in Power BI, using examples to illustrate key points.

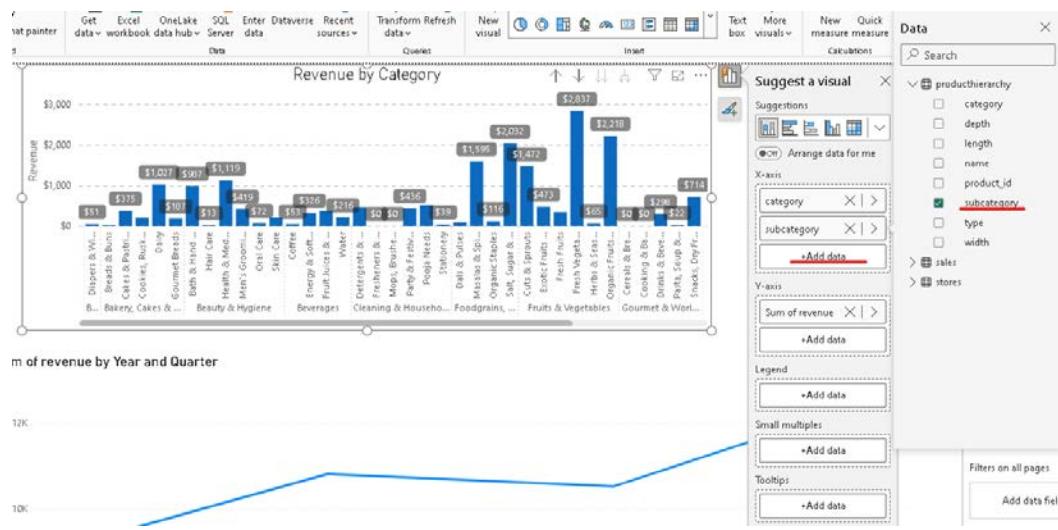
Creating a Drill-Down Hierarchy

To better understand how to create a drill-down hierarchy, let's consider a scenario where you're interested in analyzing revenue, not just by category but also by subcategory. This requires setting up a hierarchy

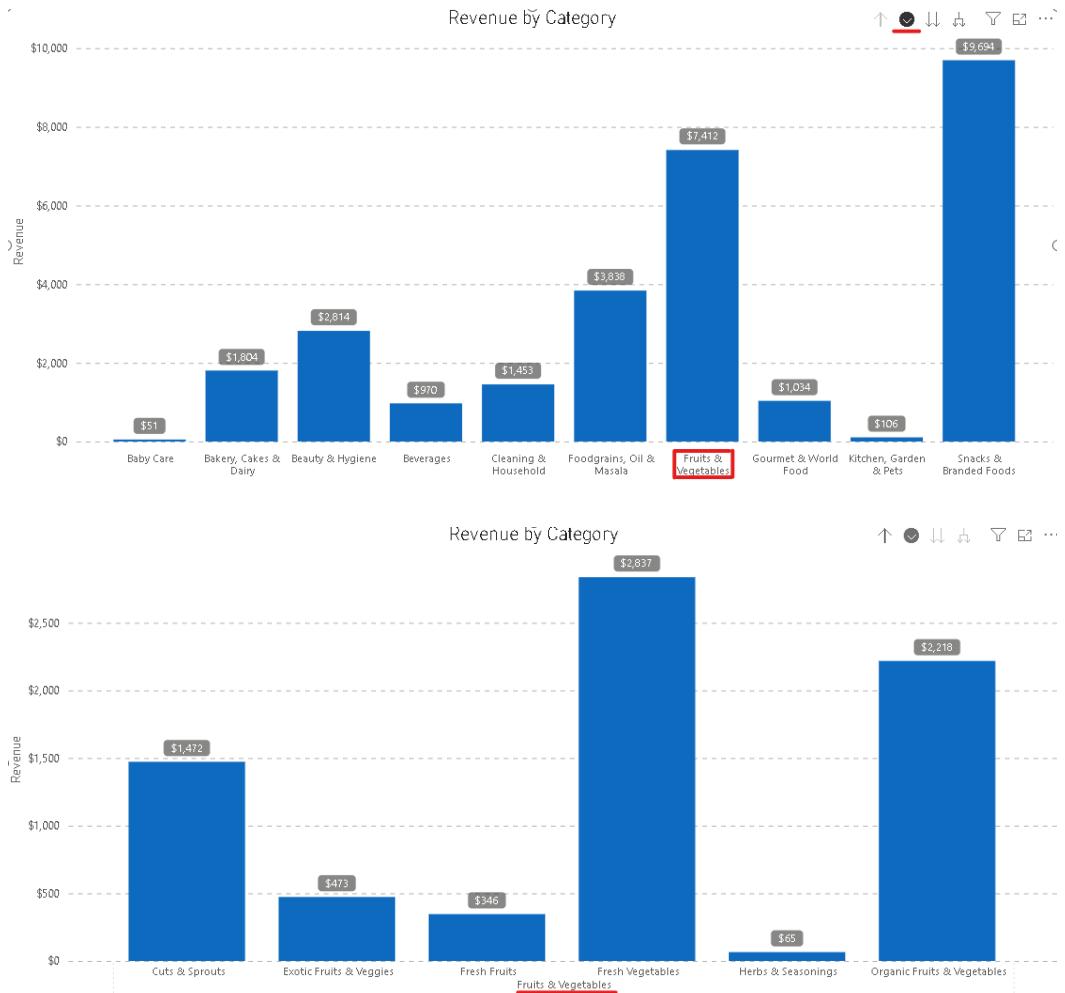
that allows you to drill down from a broad category view to more detailed subcategories within the same visual.

Steps to Create a Drill-Down Hierarchy:

- 1. Select the Visual:** Begin by selecting the visual where you want to apply the hierarchy. For instance, you might choose a bar chart that displays revenue by category.
- 2. Open the Visual Pane:** To add the hierarchy, open the "Build Visual" pane. This allows you to easily modify the visual's settings and fields.
- 3. Drag and Drop Fields:** Drag the subcategory field into the Axis section below the category. This action creates a hierarchy, enabling you to drill down from category to subcategory.



- 4. Activate Drill-Down Mode:** With the hierarchy in place, activate the drill-down mode. This feature allows you to explore data at different levels. For example, by selecting a category like "Fruits and Vegetables," you can view revenue distribution across various subcategories within that category.



5. Adjust Visual Settings: Depending on your data, you may need to adjust the visual settings for clarity. For example, you might want to resize the chart for better visibility.

Troubleshooting Common Issues

Sometimes, after setting up the drill-down hierarchy, you may notice discrepancies in the data or the visual might not display as expected. One common issue is when all values appear identical, which often indicates a problem with the data model.

Steps to Resolve Data Model Issues:

- 1. Check the Model View:** Navigate to the Model View in Power BI to inspect relationships between tables. If you notice that the relationships are incorrect—such as an old sales table still being linked to your hierarchy—you'll need to update them.

2. **Fix Relationships:** Right-click on the incorrect relationships and delete them. Then, establish the correct relationships by linking fields like Product ID from the product hierarchy table to the Product ID in the sales table, and similarly for Store ID.
3. **Update the Visual:** Once the relationships are fixed, return to your visual. You may need to remove and re-add certain fields (like revenue) to ensure the data is accurately displayed.

Expanding Hierarchies in Different Visuals

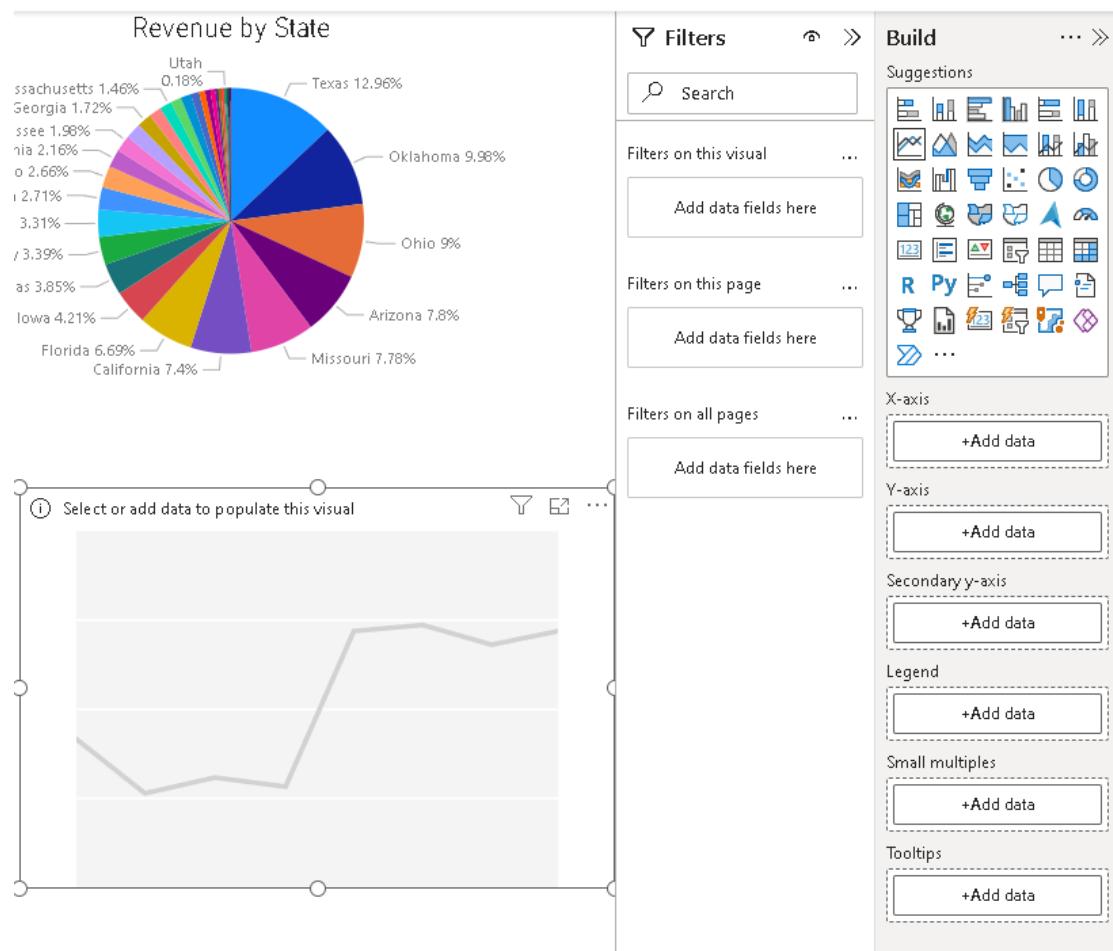
- The drill-down functionality is versatile and can be applied to various visuals beyond simple charts. For example, in a pie chart, you can add a hierarchy to the legend, allowing you to drill down into different segments.
- This powerful feature enables you to explore your data in more depth, gaining insights at both a macro and micro level, depending on your analysis needs.

▼ Line Charts

In this section, we'll explore the essentials of working with line charts in Power BI. We'll focus on key features, customization options, and practical tips to effectively visualize your data.

Overview of Line Chart Components

When you create a line chart in Power BI, several fields become available for customization. The primary components include:



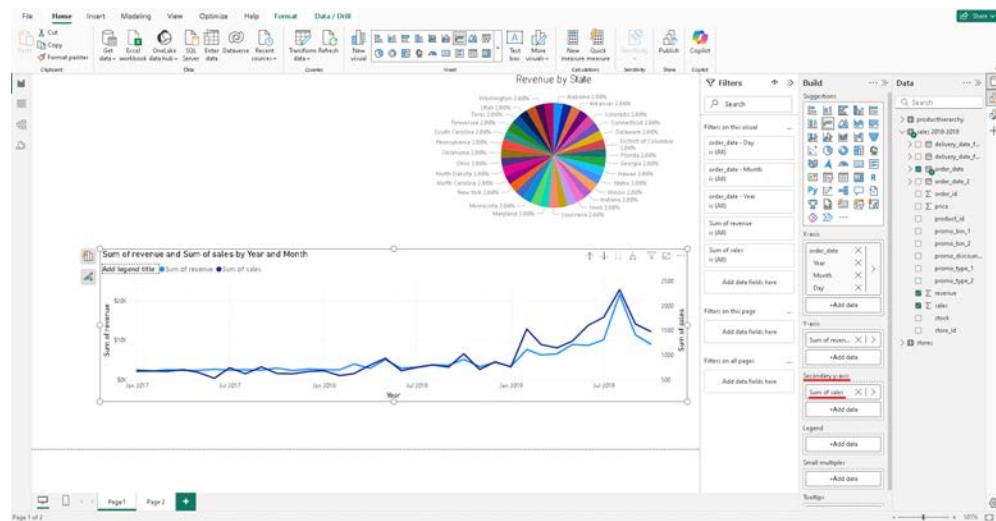
- **X-Axis:** This typically holds your hierarchy, such as dates. You can customize it by including or excluding specific fields. For example, you may choose to remove the quarter field, leaving only the year, month, and date, depending on the level of detail you need.
- **Y-Axis:** This is where you plot your primary data values, such as revenue or sales. You can add multiple data fields to the Y-axis, though this may sometimes require additional customization for clarity.
- **Secondary Y-Axis:** If your data series have significantly different scales, using a secondary Y-axis can help visualize them more effectively. This option allows each data series to have its own independent scale, making comparisons clearer.

Customizing Line Charts

Let's walk through some key customizations you can apply to a line chart:

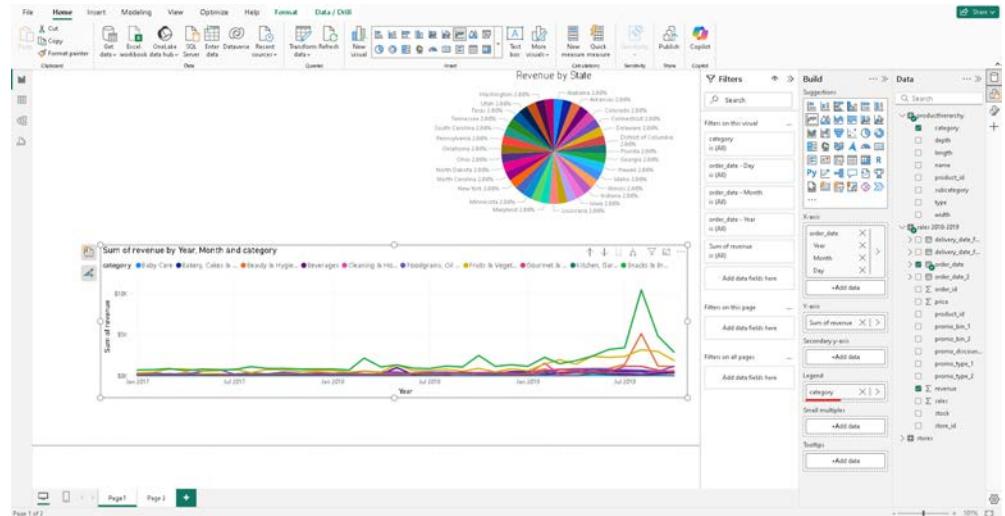
1. Adding a Secondary Y-Axis:

- If you have two data series with different magnitudes, such as revenue and sales, add one of them to the secondary Y-axis. This ensures both series are visible and can be compared more effectively.
- To do this, drag the desired field (e.g., sales) into the secondary Y-axis instead of the primary one.



2. Using Legends:

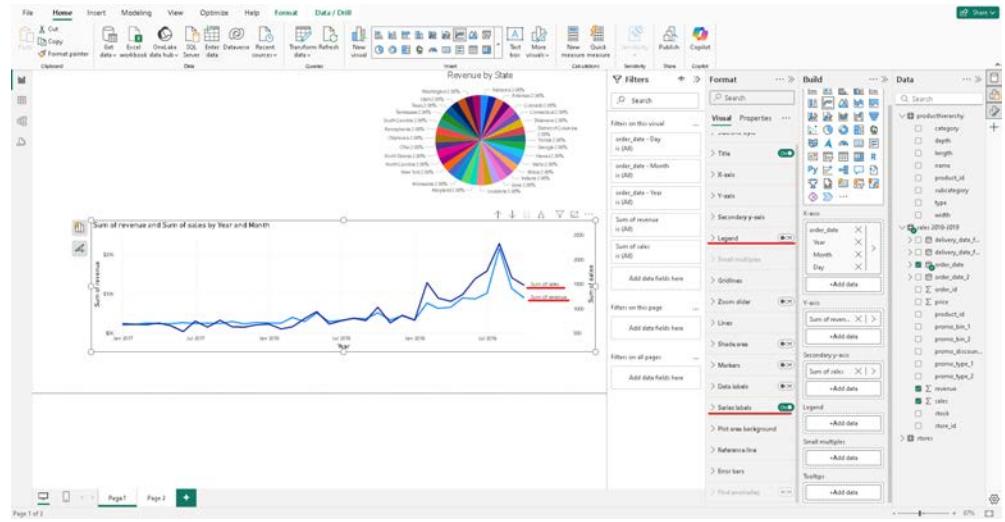
- Legends help differentiate between data series. However, note that legends cannot be used simultaneously with a secondary Y-axis or multiple Y-axis fields. Ensure only one field is used on the Y-axis if you want to include a legend.



- You can turn legends on or off, or customize their appearance, by navigating to the Format pane.

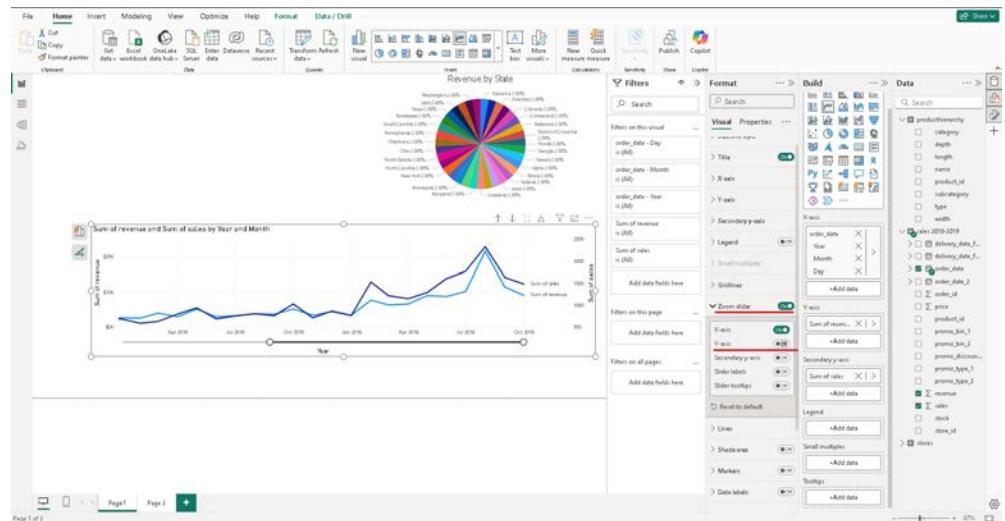
3. Formatting and Labels:

- Customize the visual's appearance by adding series labels directly on the lines. This makes it easier to identify data without referring to a separate legend.
- To rename data series for clarity, double-click the series name in the "Build Visual" pane and enter the desired name (e.g., changing "Sum of Sales" to "Sales").
- Axis titles, labels, and other formatting options are available under the Format pane, allowing further customization of your chart's appearance.



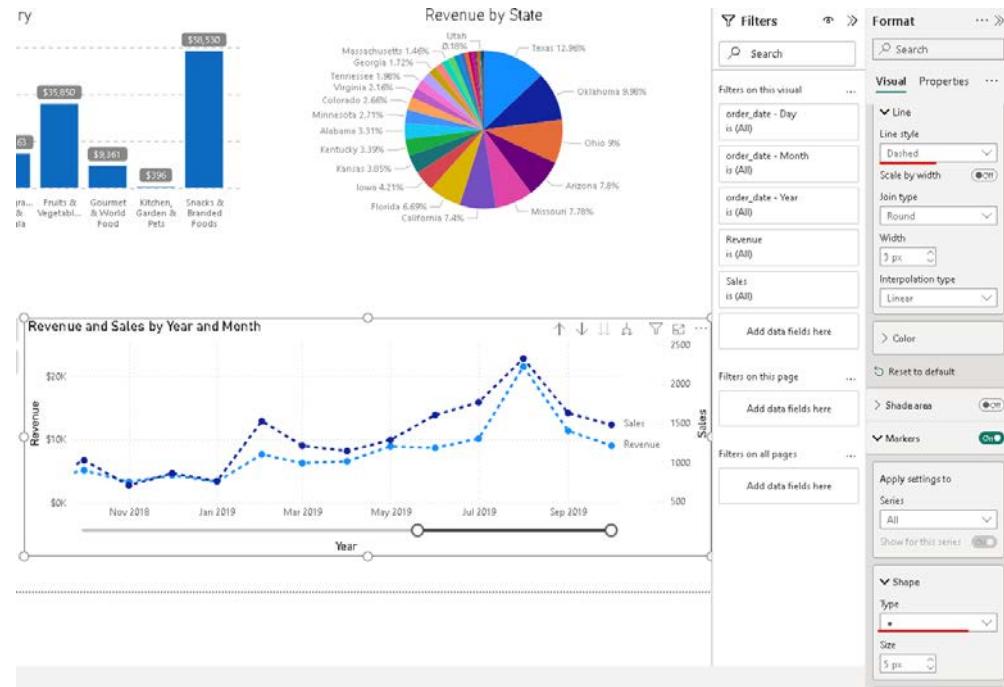
4. Zoom Slider:

- The zoom slider is a useful feature that allows users to zoom in on specific areas of the chart, especially on the X-axis. You can enable or disable the zoom slider in the Format pane, depending on whether it enhances the readability of your chart.



5. Markers and Line Styles:

- Power BI allows you to add markers to data points on your line chart, or customize the line style (e.g., solid or dashed lines). These options can help distinguish between different data series.



- However, use these features sparingly to avoid cluttering the visual. A minimalist approach often helps maintain focus on the data itself.

Practical Tips for Effective Line Charts

- **Clarity Over Complexity:** When adding multiple data series or axes, prioritize clarity. Avoid adding too many elements that could overwhelm the viewer.
- **Consistent Scales:** If possible, use consistent scales across all axes to make comparisons easier. When using a secondary Y-axis, ensure it's necessary and improves the visualization.
- **Customization for Impact:** Tailor your line chart to highlight key insights. Use labels, legends, and formatting options to make your data stand out and tell a clear story.

Line charts in Power BI are powerful tools for data visualization. By mastering their customization options, you can create clear, informative visuals that enhance your data analysis. As you continue working with

line charts, remember to practice and experiment with different settings to find what best suits your data and audience.

▼ Project 3

Project3.rar

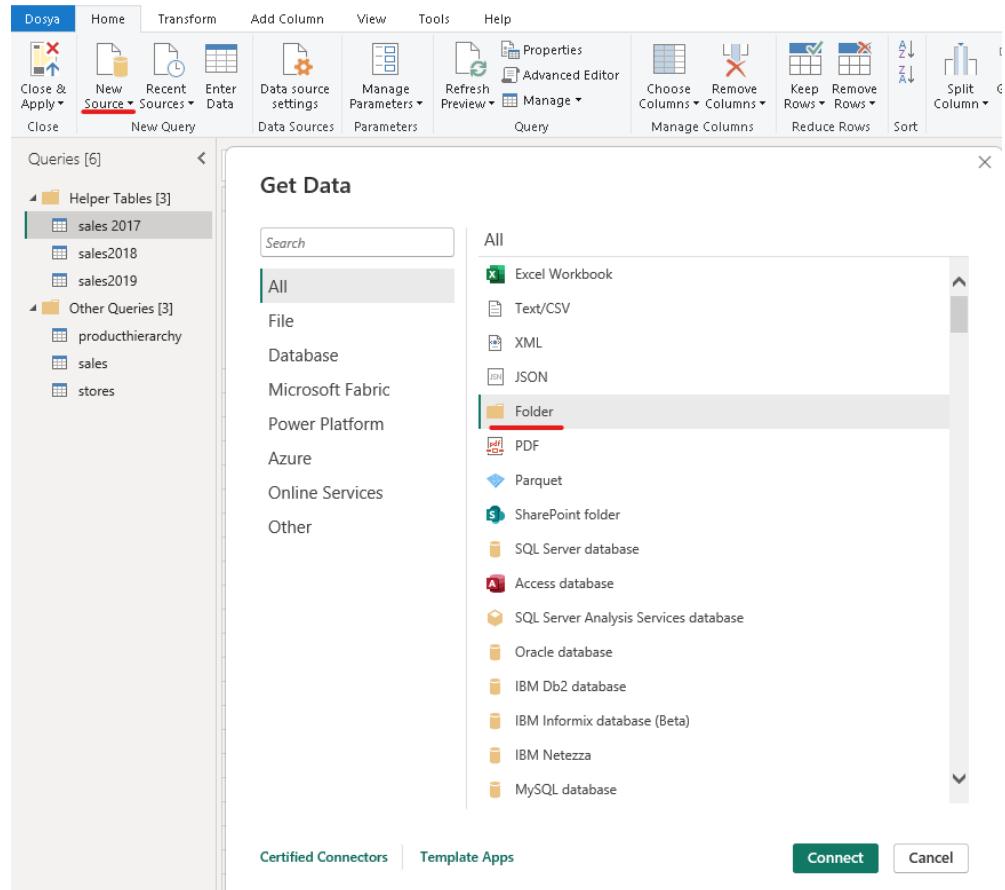
▼ Loading and Combining Data from a Folder

In previous sections, we discussed how to combine data from different sources, such as loading data for 2018 and 2019 separately and then merging them into a single sales table. However, when dealing with a large number of files such as 50 or 100 files that share the same structure manually loading and combining them can be time-consuming. Fortunately, Power BI offers a feature that allows you to load and combine files from a folder automatically, streamlining this process significantly.

Step-by-Step Guide to Loading Data from a Folder

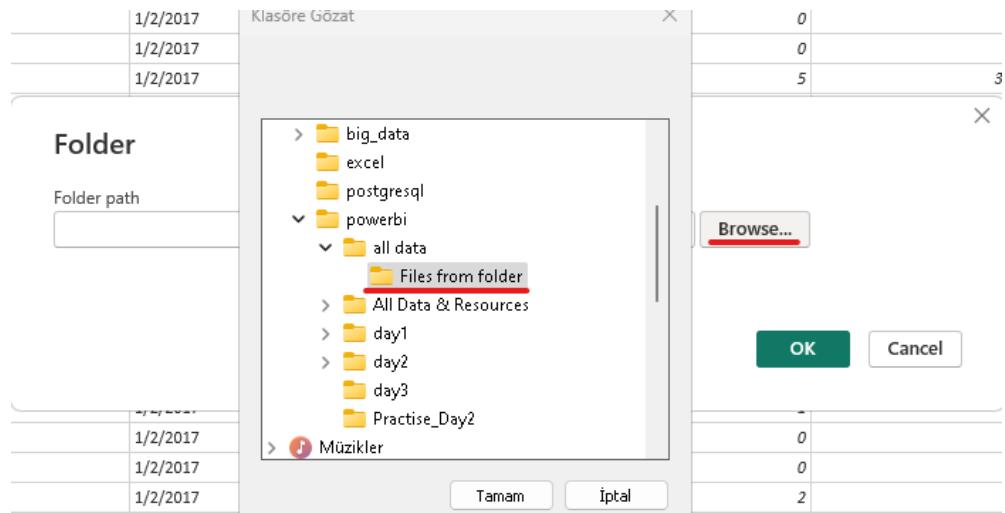
1. Accessing the Data Source:

- Begin by selecting the option to load data from a new source. This can be done from the query editor or directly from the report view.
- Click on **New Source**, then select **More...** to access the full list of data source options.
- In the search bar, type "Folder" to quickly locate the folder data source option. Click **Connect**.



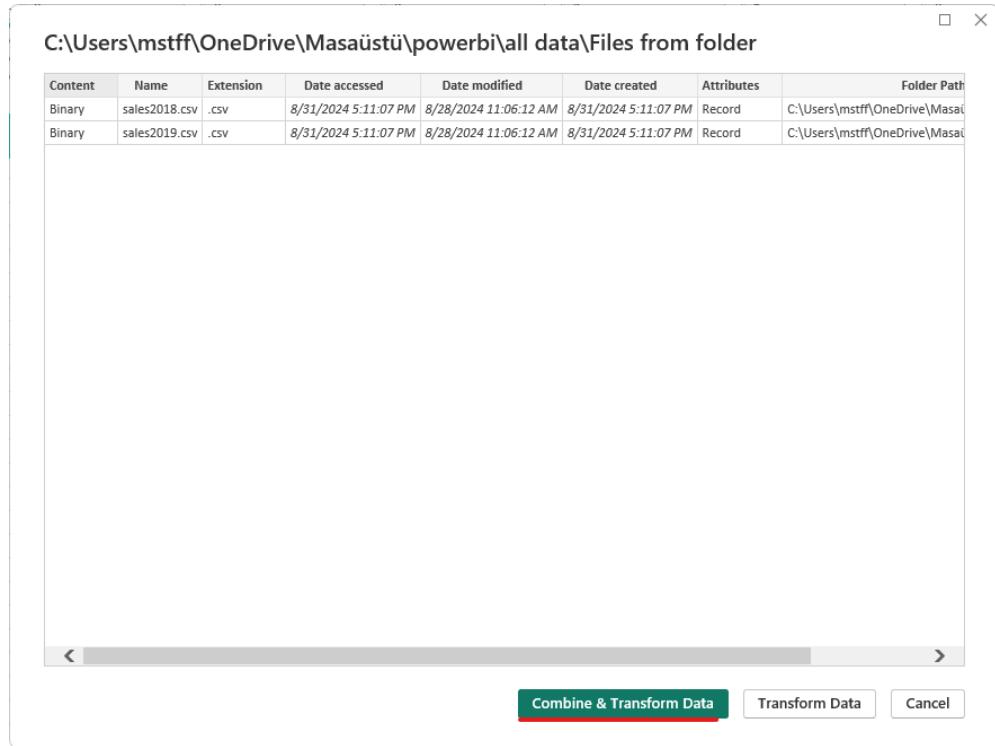
2. Selecting the Folder Path:

- You'll be prompted to enter the path of the folder containing your files. You can either paste the path directly if you have it handy or use the **Browse** button to navigate to the folder manually.
- Once the folder path is entered, click **OK**.



3. Previewing and Combining Files:

- After connecting to the folder, Power BI will display a list of the files within that folder. At this point, you have two options:
 - Combine and Transform Data:** This option allows you to combine all files immediately and make any necessary transformations during the process.
 - Transform Data:** This option lets you preview and adjust the data before combining the files.
- For simplicity and efficiency, choose **Combine and Transform Data** if you are confident the files are ready to be merged.



4. Transforming and Cleaning Data:

- Once you choose to combine the files, Power BI will display a preview of the first file's content. This preview allows you to confirm that the files are structured correctly before proceeding.
- If all files contain headers, you may need to filter these out in the subsequent steps. You can do this by selecting the option to promote the first row to headers and filtering out any additional header rows that may appear from subsequent files.
- After confirming that the data looks correct, click **OK** to proceed with loading and combining the files.

5. Managing the Combined Data:

- Once the files are combined, Power BI may create a helper table to manage the data. If this table is unnecessary for your analysis, you can collapse or remove it to focus on the main data set.
- You can also choose to remove columns, such as the source file path, if they are not needed in your analysis. Additionally, ensure

that data types are correctly detected and adjusted as needed to maintain data integrity.

Efficiency and Practicality

- By using the "Files from Folder" feature, you can significantly reduce the time and effort required to load and combine multiple files. This method is particularly useful when dealing with large datasets that are regularly updated, as you can simply refresh the connection to the folder to load new data. Once your data is combined and cleaned, you can proceed with more advanced data modeling techniques, such as the star schema or fact-dimension model, which will be covered in the next section.

▼ Understanding the Fact-Dimension Model (Star Schema)

[Fact-Dimension+Model.pdf](#)

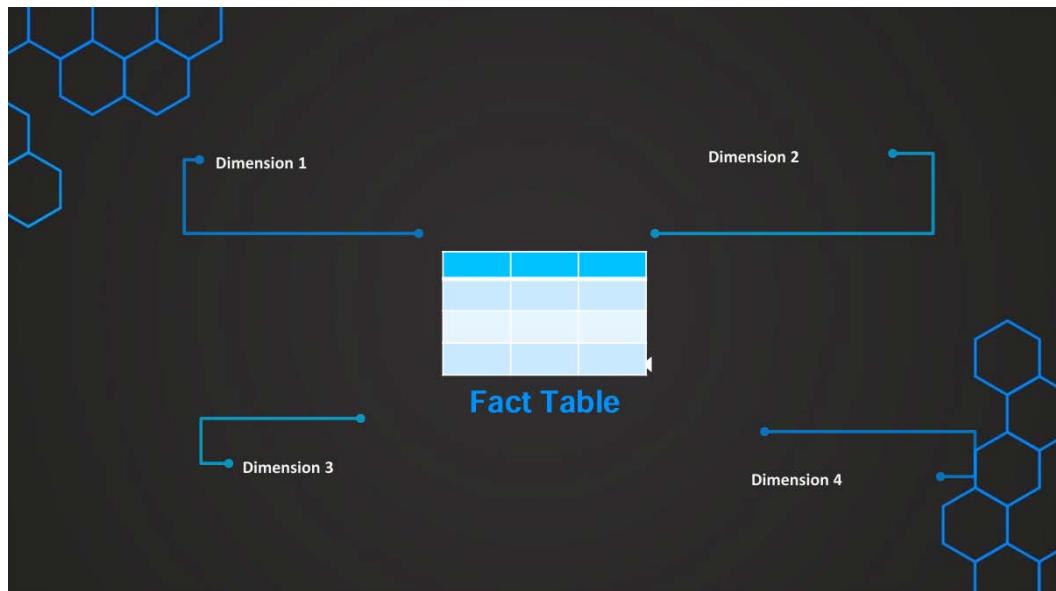
In this section, we'll delve into the Fact-Dimension Model, also known as the Star Schema. This is a fundamental concept in data modeling, essential for every data analyst to understand. While we'll cover some theory, don't worry we'll also apply these concepts practically.

What is the Fact-Dimension Model?

The Fact-Dimension Model is a type of database schema commonly used in data warehousing and analytics. It's called a "star schema" because of its structure: a central fact table is connected to multiple dimension tables, forming a shape that resembles a star.

- **Fact Table:** This is the central table in the model, containing transactional or quantitative data. It usually includes metrics such as sales, revenue, or quantities, along with foreign keys that link to dimension tables.

- **Dimension Tables:** Surrounding the fact table are dimension tables, which contain descriptive attributes related to the fact data. For example, a dimension table might hold details about products, dates, or locations.



How the Star Schema Works

Let's consider a practical example involving a supermarket's sales data:

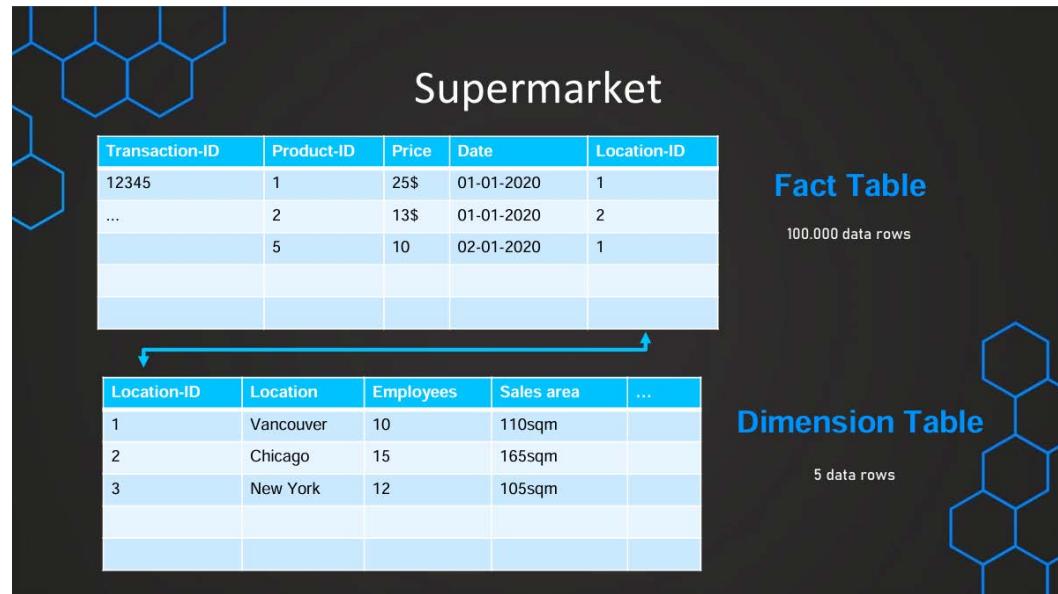
1. Fact Table:

- The fact table could be a transactional table where each row represents a sale. This table would include metrics like `Transaction ID`, `Product ID`, `Date`, `Price`, `Location ID`, `Location` and `Employees`.
- The fact table is usually large, containing millions of rows, as it records every transaction.

The table is titled "Supermarket" and contains data for five transactions. The columns are: Transaction-ID, Product-ID, Price, Date, Location-ID, Location, and Employees.

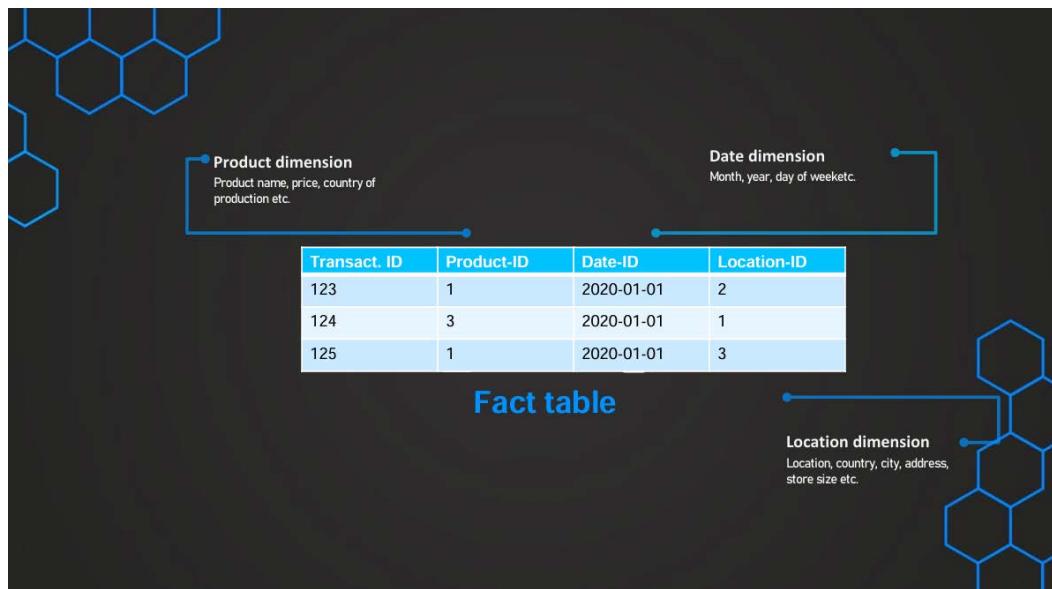
Transaction-ID	Product-ID	Price	Date	Location-ID	Location	Employees
12345	1	25\$	01-01-2020	1	Vancouver	10
...	2	13\$	01-01-2020	2	Chicago	15
	5	10	02-01-2020	1	Vancouver	10

2. Dimension Tables:



Why did we split it into 2 tables like this?

Initially, storing transaction data in a single table becomes inefficient due to duplication of information such as Location ID, Location. To increase this efficiency, a "dimension" table is created that contains fixed information.



- **Product Dimension:** This table includes attributes like `Product`, `Product Name`, `Country`, and `Price`. It provides detailed information about each product.
- **Location Dimension:** This table includes `Location`, `City`, `Address`, `Country` and `Store Size`. It provides contextual information about where the sales occurred.
- **Date Dimension:** This table might include `Day of Week`, `Month` and `Year`. It allows you to analyze sales over time.

By linking these dimension tables to the fact table using foreign keys, you can perform detailed analysis. For example, you can aggregate sales data by product category, location, or time period.

Key Concepts: Primary Key and Foreign Key

Supermarket

Foreign Key

Transaction-ID	Product-ID	Price	Date	Location-ID
12345	1	25\$	01-01-2020	1
...	2	13\$	01-01-2020	2
	5	10	02-01-2020	1

Primary Key

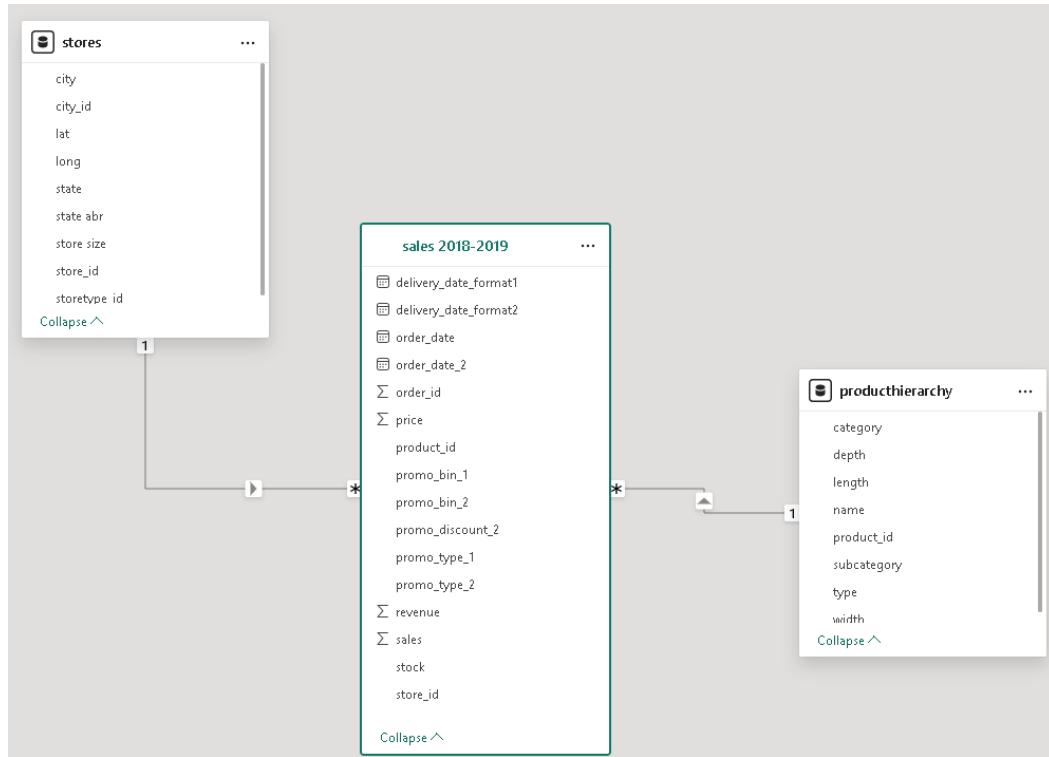
Location-ID	Location	Employees	Sales area	...
1	Vancouver	10	110sqm	
2	Chicago	15	165sqm	
3	New York	12	105sqm	

- **Primary Key:** This is a unique identifier for each row in a table. In the fact table, the primary key might be `Transaction ID`. In the product dimension table, it would be `Product ID`. Each primary key must be unique within its table.
- **Foreign Key:** This is a column in the fact table that links to the primary key in a dimension table. For example, the `Transaction ID` in the fact table is a foreign key that references the `Product ID` in the product dimension table. Foreign keys do not need to be unique and can repeat across multiple rows.

Our Table Relationships

If we look at the table relationships in our examples:

- Sales 2018-2019 table is a Fact Table
- Stores dimension
- Product Hierarchy dimension



The Fact-Dimension Model, or Star Schema, is a powerful tool in data analytics, enabling efficient and scalable data modeling. While not every dataset requires a star schema, understanding this concept allows you to design better models when the need arises. As you continue your work in Power BI, consider how you can apply these principles to optimize your data structures and improve performance.

Benefits of the Star Schema

- **Efficiency:** By organizing data into a star schema, you reduce redundancy and improve query performance. Instead of repeating the same descriptive information in every row of the fact table, you store it once in a dimension table and reference it as needed.
- **Scalability:** The star schema is designed to handle large volumes of data efficiently. As your dataset grows, the model remains performant because the fact table stays relatively slim.
- **Simplicity:** The star schema is straightforward and easy to understand, making it easier to build and maintain complex data models.

▼ Editing Relationships

In this section, we'll explore how to edit existing relationships in Power BI, focusing on key concepts like cardinality and cross-filter direction. Understanding how to manage relationships in your data model is essential for accurate analysis and reporting.

Navigating to the Model View

To begin editing relationships, first, close the Query Editor by clicking on "Close & Apply." Next, navigate to the Model View where you can see all your tables and their relationships visually represented.

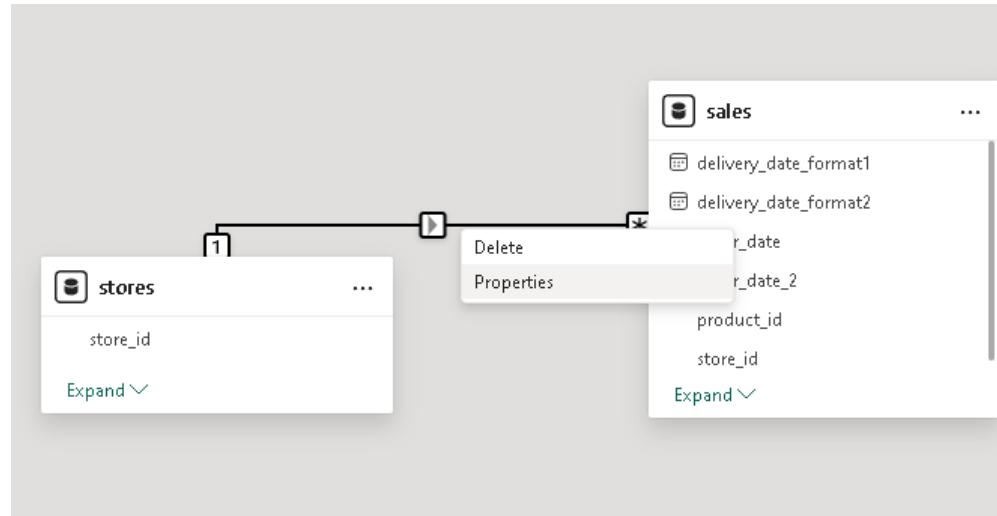
- In Model View, you can drag the tables around to organize them as needed.
- When you hover over a relationship line between tables, the columns used to establish that relationship will be highlighted. This helps you quickly identify which fields are connected.

Editing a Relationship

To edit an existing relationship, follow these steps:

1. Select the Relationship:

- Right-click on the relationship line between two tables.
- Choose "Properties" from the context menu. This opens the relationship settings window.

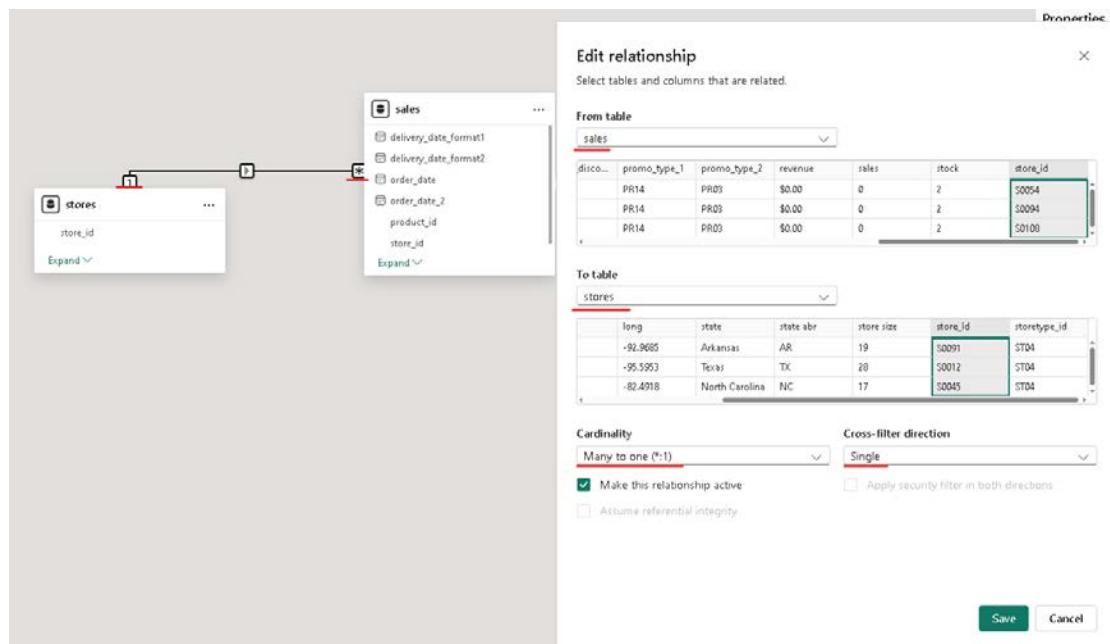


2. Modify the Relationship:

- In the properties window, you'll see the columns currently used to establish the relationship. If a wrong column was selected, you can easily correct it here by choosing the correct column from the dropdown list.
- After making any changes, ensure that the relationship is correctly defined.

Understanding Cardinality

Cardinality defines the nature of the relationship between two tables:



Cardinality

- Many to one (*:1)
- Many to one (*:1)
- One to one (1:1)
- One to many (1:*)
- Many to many (*:*)

- **One-to-Many (1:*) Relationship:**

- This is the most common type of relationship in data models. It means that one record in the first table (e.g., `Stores`) can be associated with multiple records in the second table (e.g., `Sales`).
- In the relationship diagram, this is represented by a "1" on one side and a "*" (many) on the other.

- **Many-to-One (*:1) Relationship:**

- This is essentially the reverse of a one-to-many relationship, where multiple records in the first table are linked to a single record in the second table.

- Power BI automatically detects and assigns the appropriate cardinality based on the data.
- **One-to-One (1:1) Relationship:**
 - In a one-to-one relationship, each record in the first table corresponds to exactly one record in the second table. This is less common and usually only necessary in specific scenarios.
- **Changing Cardinality:**
 - You can change the cardinality in the properties window, but this is rarely necessary. Power BI usually determines the correct cardinality automatically. If you attempt to set an invalid cardinality, Power BI will notify you.

Additional Options in Relationship Settings

- **Active/Inactive Relationship:**
 - Relationships can be marked as active or inactive. An active relationship is used by default in calculations, while inactive relationships can be used in specific measures or calculations when needed.

Edit relationship

Select tables and columns that are related.

From table

sales

disco...	promo_type_1	promo_type_2	revenue	sales	stock	store_id
PR14	PR03	\$0.00	0	2	S0054	
PR14	PR03	\$0.00	0	2	S0094	
PR14	PR03	\$0.00	0	2	S0108	

To table

stores

long	state	state abr	store size	store_id	storetype_id
-92.9685	Arkansas	AR	19	S0091	ST04
-95.5953	Texas	TX	28	S0012	ST04
-82.4918	North Carolina	NC	17	S0045	ST04

Cardinality

Many to one (*:1)

Cross-filter direction

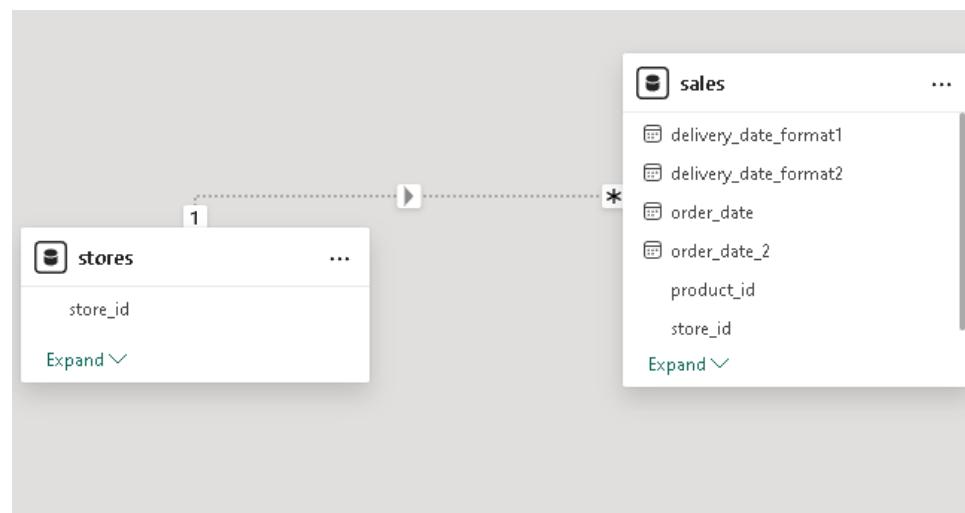
Single

Make this relationship active

Apply security filter in both directions

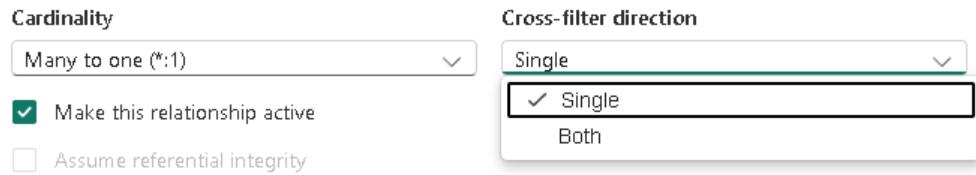
Assume referential integrity

Save **Cancel**



- **Cross-Filter Direction:**

- This setting controls how filters flow between tables. The options include:



- Single:** Filters flow in one direction (from one table to another).
- Both:** Filters can flow in both directions, which is useful in certain complex models but can sometimes lead to circular references if not managed carefully.

Editing relationships in Power BI is a crucial step in ensuring your data model accurately represents the relationships within your data.

Understanding cardinality and how to modify relationships allows you to correct and optimize your data model, leading to more accurate reports and insights. In the next section, we'll dive deeper into the cross-filter direction and how it affects your data model.

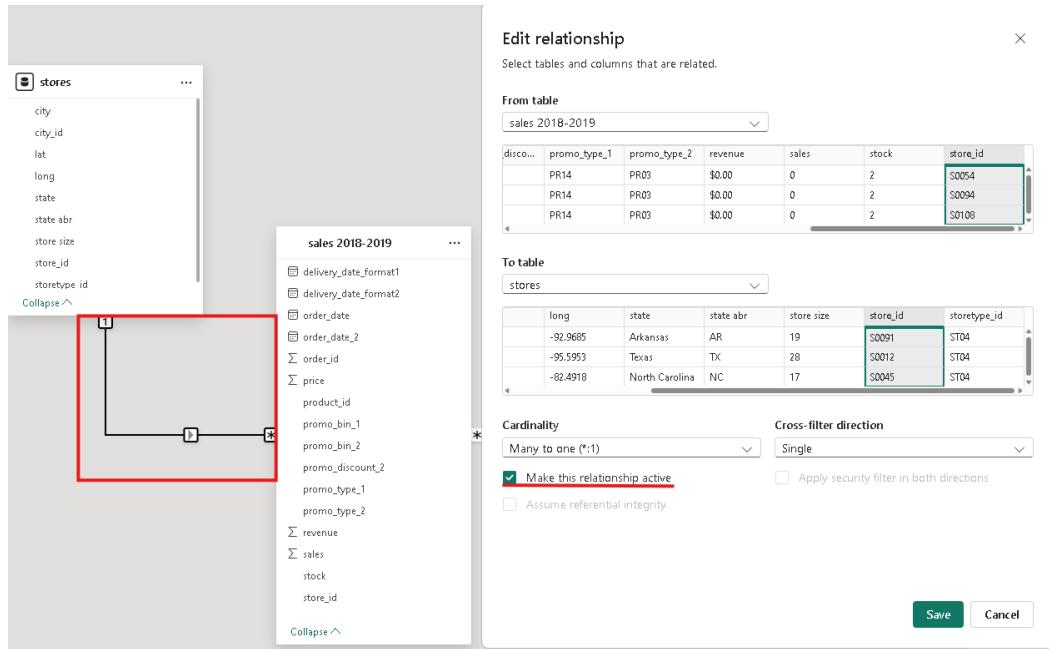
▼ Activating and Deactivating Relationships

In this section, we'll explore the concept of active and inactive relationships in Power BI. Understanding how to manage these relationships is crucial for building accurate and functional data models.

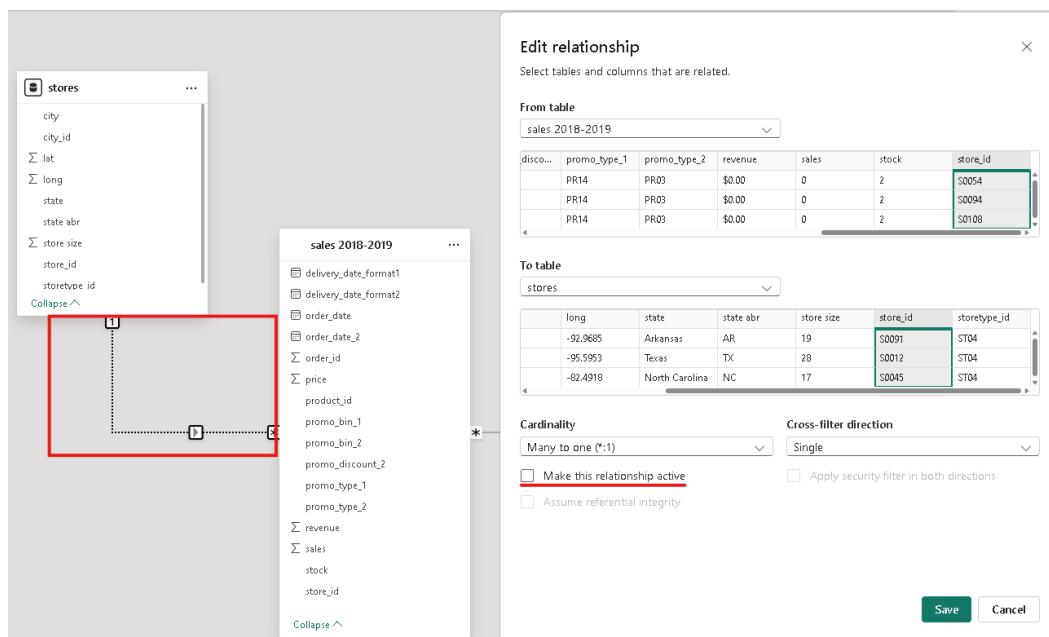
What Are Active and Inactive Relationships?

In Power BI, relationships between tables can be either active or inactive:

- Active Relationship:** This is the default state of a relationship. An active relationship is used in your reports and visuals. When a relationship is active, Power BI automatically applies filters and relationships during calculations and data visualizations.



- Inactive Relationship:** An inactive relationship exists in the data model but does not influence the report unless explicitly activated in a DAX function. Inactive relationships are represented by dotted lines in the Model View, indicating they are not currently in use. In other words, the relationship exists but is not being used.



How to Activate or Deactivate a Relationship

1. Opening the Relationship Settings:

- In the Model View, locate the relationship line between two tables. Right-click on the line and select "Properties" to open the relationship settings window.

2. Activating/Deactivating the Relationship:

- Inside the properties window, you'll find a checkbox labeled "Make this relationship active."
- **To deactivate a relationship**, uncheck this box. The line representing the relationship will turn into a dotted line, indicating that the relationship is inactive.
- **To reactivate the relationship**, check the box again, and the line will return to a solid state, indicating the relationship is active.

Understanding the Impact of Inactive Relationships

When a relationship is inactive:

- It will not impact your reports or visuals. The relationship is essentially ignored in any automatic data filtering.
- Inactive relationships can still be used in DAX calculations through specific functions like `USERELATIONSHIP`, allowing you to temporarily activate them within a formula.

Inactive relationships are often used when:

- **Circular References**: If your data model creates a circular reference (where tables are connected in a loop), Power BI will automatically set one of the relationships to inactive to prevent logical conflicts. For example, if you try to create a new relationship between two tables that would complete a loop with existing relationships, Power BI may automatically deactivate it.
- **Multiple Relationships**: If you need different relationships for different calculations, you can have one active relationship and others inactive, switching between them as needed in DAX.

Practical Example: Circular References

Consider a scenario where you have three tables `Sales`, `Stores`, and `Products` and you try to create relationships that would connect these tables in a circular manner. Power BI will automatically deactivate one of these relationships to avoid inconsistencies.

For instance:

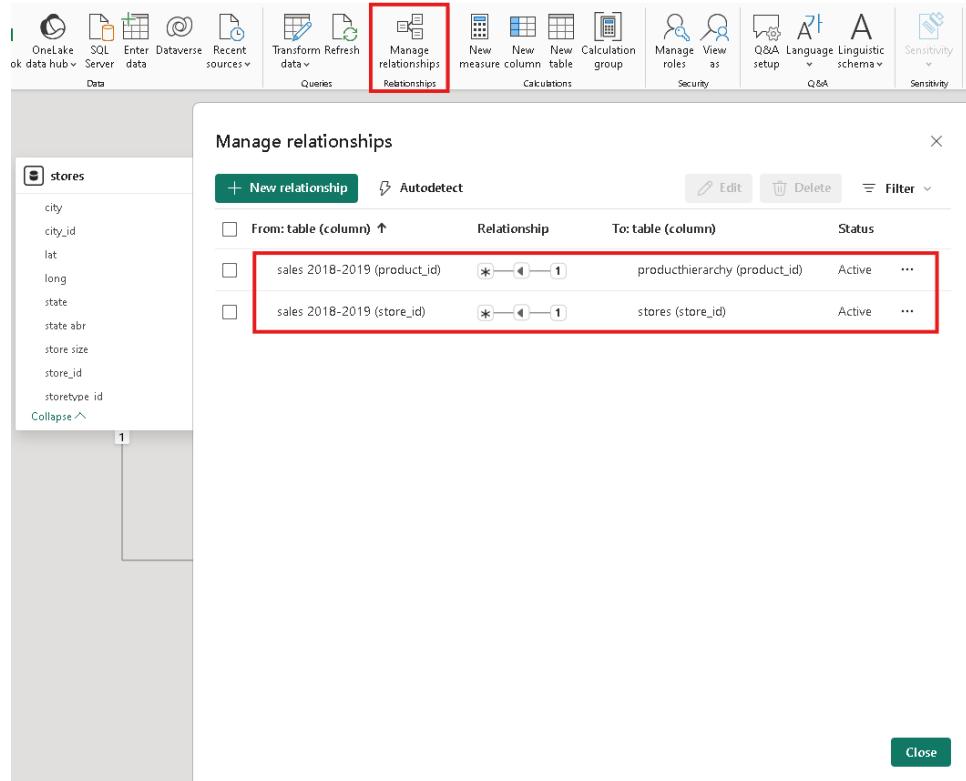
- If you already have an active relationship between `Sales` and `Stores` and another between `Sales` and `Products`, creating a relationship between `Stores` and `Products` may result in an inactive relationship due to the potential for circular logic.
- If you attempt to activate this new relationship without adjusting the others, Power BI will prevent you from doing so, as it would create logical conflicts.

To resolve this:

- You could deactivate an existing relationship to make the new one active, but this should be done carefully to avoid disrupting your data model.
- Alternatively, you may decide that the new relationship isn't necessary and delete it to maintain the integrity of your model.

▼ Auto-Detected Relationships and Manage Relationships

Power BI has an auto-detect feature that automatically creates relationships between tables based on matching column names. While convenient, this feature can sometimes lead to inactive or incorrect relationships if not carefully reviewed:



- **Inactive Relationships:** If Power BI detects relationships that could cause circular references or other issues, it may set them as inactive. If you rely on auto-detection, you might end up with unexpected inactive relationships that can affect your reports.
- **Performance Issues:** Automatically detected relationships may not always be the most efficient or logical for your specific data model, leading to potential performance problems.

It's essential to review and manually adjust relationships as needed, rather than relying solely on auto-detection.

Managing active and inactive relationships is a critical aspect of building robust data models in Power BI. Understanding when and how to use these relationships, especially in complex models, can prevent logical errors and ensure accurate reporting.

▼ 3- Data Transformations

▼ Tables

Working with Table Visuals in Power BI

In this section, we'll explore the use of the table visual in Power BI. While tables may not provide immediate visual cues like graphs or charts, they are invaluable when you need to display detailed data. Let's go through how to create and configure a table visual effectively.

Why Use a Table Visual?

Table visuals are particularly useful when you need to:

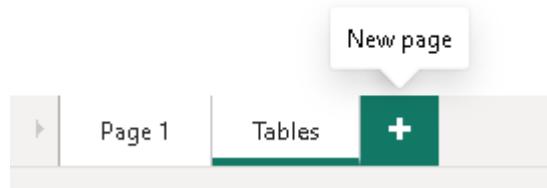
- Display detailed data points, such as specific values for each day or transaction.
- Provide a comprehensive view that includes multiple data fields and their relationships.
- Show raw data in a way that is easy to understand and analyze.

Unlike graphs that give quick insights through visual patterns, tables are ideal for in-depth analysis where exact numbers and specific details are important.

Creating a Table Visual

1. Adding a New Page:

- Start by creating a new page in your report. You can do this by clicking the "+" icon at the bottom of the screen. Rename the page to something relevant, such as "Tables," by double-clicking on the page name.



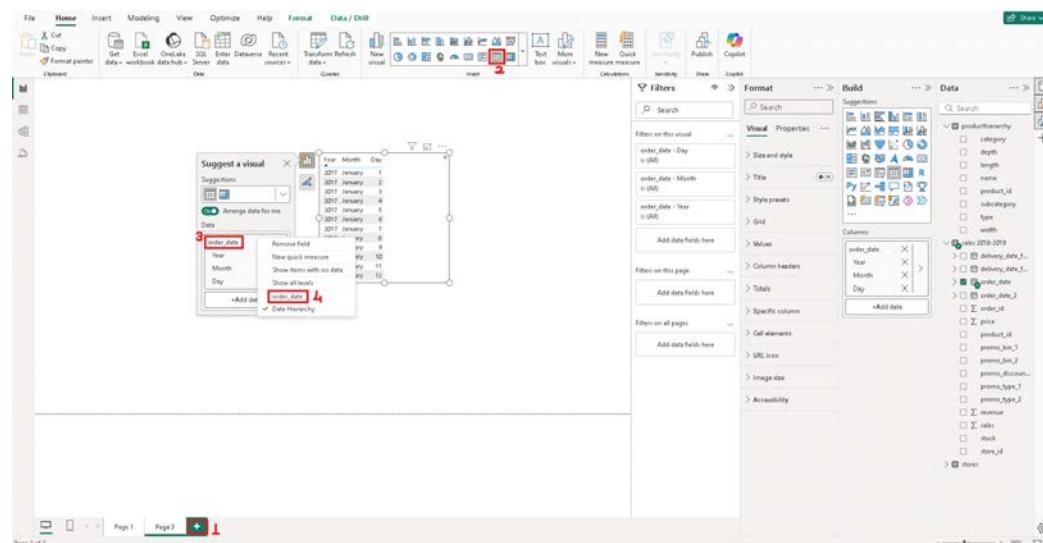
2. Inserting the Table Visual:

- On the Home ribbon, select the "Table" option in the Insert area. This will add a table visual to your page.

- Next, open the "Build a visual" pane to start adding data to your table.

3. Adding Data to the Table:

- Begin by adding the **Order Date**. You can do this by dragging the **Order Date** field into the table visual or into the columns area in the "Build a visual" pane.
- By default, Power BI might add a date hierarchy (Year, Quarter, Month, Day). If you want to display the exact date instead of the hierarchy, right-click on the **Order Date** field and select "Order Date" instead of "Date Hierarchy."



4. Formatting the Date:

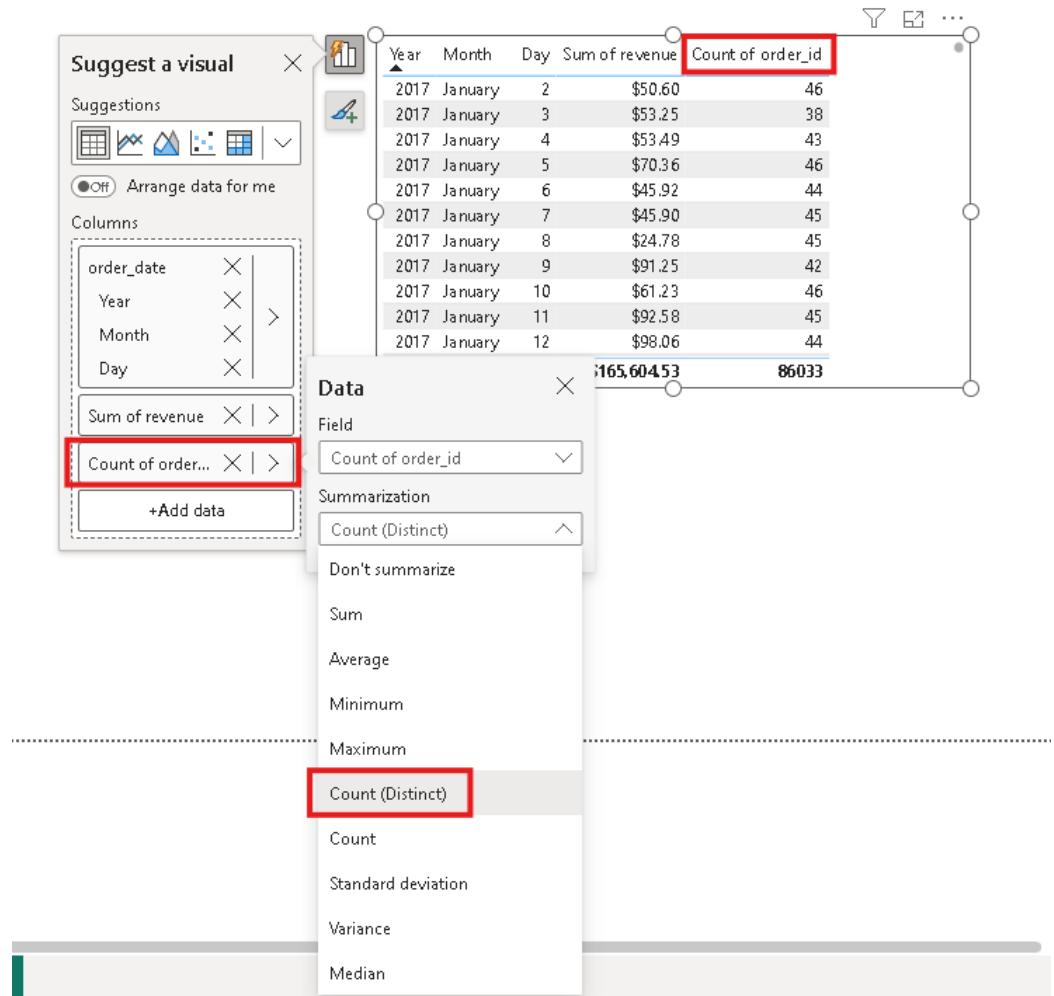
- To change the date format, select the **Order Date** column. This will open the "Column Tools" ribbon. Here, under "Format," you can choose a different date format, such as "Short Date," which might be more suitable for your needs.

5. Adding Revenue and Aggregating Data:

- Next, drag the **Revenue** field into the table. By default, Power BI will sum the revenue values. You can change the aggregation method (e.g., from Sum to Average) by selecting the column and modifying the aggregation option in the "Data options" pane.

6. Counting Orders:

- To add the number of orders, drag the `order_ID` field into the table. By default, Power BI will sum the `Order ID`, but this isn't meaningful. Instead, change the aggregation to "Count Distinct" to count unique orders. This is useful if some orders might appear multiple times in your data.



7. Renaming Columns:

- To make your table more user-friendly, rename the columns. Double-click on the column names in the table and rename them to something more intuitive. For example, change "Order Date" to "Date," "Sum of Revenue" to "Revenue," and "Count of Order ID" to "Orders."

8. Using the "Don't Summarize" Option:

- In the table visual, you have the unique option to choose "Don't Summarize" for numerical fields like revenue. This means instead of aggregating the data (like summing), Power BI will show each individual value as it appears in the dataset. This is useful when you need to see each transaction or entry separately.

The screenshot shows a Power BI interface with a table visual. The table has columns: Year, Month, Day, revenue, and Count of order_id. The data shows various dates in January 2017 with their corresponding revenue and order counts. A red box highlights the entire table area. To the right, a context menu titled "Suggest a visual" is open. It includes a "Suggestions" section with icons for different visual types. Below it is a "Columns" section listing "order_date", "Year", "Month", and "Day". The "Data" section shows "Field" set to "revenue" and "Summarization" set to "Don't summarize". A red box highlights the "Don't summarize" option under "Summarization". Other summarization options listed include Sum, Average, Minimum, Maximum, Count (Distinct), Count, Standard deviation, Variance, and Median.

The table visual in Power BI is a powerful tool for displaying detailed data that requires a close examination of individual records. While it lacks the immediate visual impact of charts and graphs, it excels in providing clarity and specificity. In the next section, we'll delve into customizing and formatting the table visual to enhance its readability and usability.

▼ Customizing Tables

In Power BI, customizing tables allows for enhanced presentation and clarity. Here's how you can format and customize tables to suit your design needs:

1. Accessing the Format Pane:

- Select the table you want to customize.
- Open the **Format Pane** from the right-hand side. If it's not visible, click the plus icon to add the pane.

2. Using Style Presets:

- Power BI provides several style presets such as **Minimal**, **Bold header**, and **Alternating rows**. These can be used as starting points.

- After applying a preset, further customization is possible.
 - **Minimal:** A clean, minimalist style that emphasizes data clarity.

Order Date	Revenue	Orders
1/2/2017	\$0.00	37
1/2/2017	\$0.69	1
1/2/2017	\$0.83	1
1/2/2017	\$1.16	2
1/2/2017	\$1.30	1
1/2/2017	\$2.78	1
1/2/2017	\$4.54	1
1/2/2017	\$4.58	1
1/2/2017	\$33.56	1
Total	86033	

- **Bold Header:** Highlights the table headers with bold formatting.

Order Date	Revenue	Orders
1/2/2017	\$0.00	37
1/2/2017	\$0.69	1
1/2/2017	\$0.83	1
1/2/2017	\$1.16	2
1/2/2017	\$1.30	1
1/2/2017	\$2.78	1
1/2/2017	\$4.54	1
1/2/2017	\$4.58	1
1/2/2017	\$33.56	1
Total	86033	

- **Alternating Rows:** Adds alternating row colors for better readability.

Order Date	Revenue	Orders
1/2/2017	\$0.00	37
1/2/2017	\$0.69	1
1/2/2017	\$0.83	1
1/2/2017	\$1.16	2
1/2/2017	\$1.30	1
1/2/2017	\$2.78	1
1/2/2017	\$4.54	1
1/2/2017	\$4.58	1
1/2/2017	\$33.56	1
Total	86033	

3. Grid Customization:

- You can adjust the gridlines, including adding **borders** to the top, bottom, left, or right of the table.

The screenshot shows a table visualization with columns: Order Date, Revenue, and Orders. The data includes several rows and a total row at the bottom. To the right of the visualization is a settings pane. In the 'Grid' section of the pane, there is a 'Color' dropdown menu. A yellow circle highlights this dropdown.

- The color of gridlines and borders can also be changed for better visual contrast.

4. Row and Text Colors:

- Customize the **text color**, **background color**, and **alternate row color** for a more appealing look.

The screenshot shows the same table visualization as the previous image. To the right is a settings pane. The 'Values' section is highlighted with a yellow circle. This section contains settings for 'Font' (set to Segoe UI), 'Text color' (set to black), and 'Background color' (set to light blue). Other sections like 'Font style' and 'Size' are also visible but not highlighted.

- For example, you can apply light blue for text and a darker blue for alternating rows.

5. Column Headers and Totals:

- You can change the background color and text color of the column headers and totals. This allows for a distinction between regular data rows and important metrics like totals or averages.
- You can also modify the **font style** and **size** for values, column headers, and totals.

A screenshot of a Power BI report interface. On the left is a table visual with columns: Order Date, Revenue, and Orders. The data shows various dates from January 2017 with corresponding revenue and order counts. A yellow circle highlights the ellipsis icon in the top right corner of the table. To the right is the Power BI context pane, which includes sections for Filters on this page, Filters on all pages, and a Totals section. The Totals section is expanded, showing a Values group with a 'Total' label, font settings (DIN Light, size 10), and text color options.

6. Modifying Labels:

- You can rename the **Total label** to something more appropriate, such as **Average**. This can be done under the totals section in the format pane.

7. Handling Aggregations:

- If certain values like revenue aren't displaying, it might be due to the aggregation method. Ensure that the correct aggregation (e.g., Sum, Average) is applied.

A screenshot of the Power BI Data pane. On the left, there's a table visual with columns: Order Date, Revenue, and Orders. The Data pane shows the 'revenue' field selected. In the Aggregation dropdown, the 'Sum' option is highlighted with a yellow circle. Other aggregation options listed include Average, Minimum, Maximum, Count (Distinct), Count, and Standard deviation.

8. Column-Specific Formatting:

- Customize individual columns by changing their background or text color. For example, the **Order Date** column can be highlighted with a different background for easy distinction.

▼ Conditional Formatting (Rules)

Now, we'll explore another option: **rules-based conditional formatting**.

Here's how it works:

1. Accessing Conditional Formatting Rules:

- Turn on the **Icons** option to apply rules-based formatting with icons. By default, Power BI applies icons to different values based on a descending order. The highest values receive specific icons, and lower values receive others.

The screenshot shows a Power BI report with a table visual. On the right, the 'Icons' settings dialog is open, with the 'Icons' section highlighted by a yellow circle. Below it, the 'Icons - Icons' dialog is also open, showing three rules defined based on revenue percentages. The first rule uses a red diamond icon for values >= 0% and < 33%. The second rule uses a yellow triangle icon for values between 33% and 67%. The third rule uses a green circle icon for values >= 67%. The table visual displays these icons next to the revenue values.

Order Date	Revenue	Orders
1/2/2017	\$50.60	46
1/3/2017	\$53.25	38
1/4/2017	\$53.49	43
1/5/2017	\$70.36	46
1/6/2017	\$45.92	44
1/7/2017	\$45.90	45
1/8/2017	\$24.78	45
1/9/2017	\$91.75	42
1/10/2017	\$61.23	46
1/11/2017	\$92.58	45
1/12/2017	\$99.09	44
Total	\$165,604.53	86033

2. Understanding Rule-Based Decisions:

- The assignment of icons is based on rules you define. To customize these rules, click on the conditional formatting icon (the formula icon). You will see preset rules, such as stars or other symbols.
- You can select from various styles, including colorful options, but the real focus is understanding how these rules operate.

3. Defining Rules by Percentages:

- Power BI uses a percentage scale where **0%** represents the lowest value and **100%** represents the highest value in the table.
- You can define specific rules by adjusting percentages. For instance, if you want a star icon to appear for values from **99%** to

100%, you can set this range. The star will be applied to all values except the highest one.

4. Creating Additional Rules:

- You can add more rules, such as using a **half-star** for values between **50%** and **80%**.
- When rules overlap, Power BI applies the rule with the **stronger priority**. This priority is determined by the rule's position in the list, with lower rules overriding higher ones if both apply.

The screenshot shows the 'Icons - Icons' dialog box. The 'Format style' dropdown is set to 'Rules' and 'Apply to' is 'Values only'. The 'What field should we base this on?' dropdown is set to 'Sum of revenue' and 'Summarization' is 'Sum'. The 'Icon layout' dropdown is 'Left of data' and 'Icon alignment' is 'Top'. The 'Style' dropdown contains three yellow stars. Below it are three conditional rules:

- If value ≥ 0 and < 50 then ★½
- If value ≥ 50 and < 80 then ★½
- If value ≥ 80 and ≤ 100 then ★

The 'Ok' button at the bottom left is circled in red.

Order Date	Revenue	Orders
8/21/2019	★	144
8/17/2019	★	178
8/26/2019	★★	142
5/1/2019	★★	158
8/25/2019	★★	132
10/14/2018	★★	70
2/6/2019	★★	126
9/8/2019	★★	132
9/15/2019	★★	122
3/21/2018	★★	60
9/2/2019	★★	172
Total	\$165,604.53	86033

5. Adjusting Values and Icons:

- Be mindful of whether you are using **percentages** or **absolute numbers** when setting rules. Ensure consistency to avoid confusion in the visual output.
- As you add more rules, you can select icons from the preset list or customize them based on your specific needs.

In summary, rules-based conditional formatting allows you to visually differentiate data by applying icons or symbols according to customized conditions. This is a powerful way to give clear visual cues in your tables, enhancing readability and interpretation.

▼ Project 4

[Project4.rar](#)

▼ Merge Queries

In this section, we'll cover how to merge queries in Power BI, a crucial process when you need to combine data from different tables into a single, comprehensive view.

When to Merge Queries

Merging queries is essential when you want to combine data from two or more tables that share a common field. For example, if you have a table containing store information and another table with cost data (such as building or development costs), you might want to merge these tables to have all related information in one place, making it easier to analyze and report.

Loading the Data

Before merging, you need to load the tables into Power BI:

1. **Open Query Editor:** Start by opening the Power BI Query Editor.
2. **Load Data from Excel:** If your data is in an Excel file, go to **New Source > Excel Workbook**. Navigate to the file location and click **Open**.
3. **Select Sheets:** In the Navigator, select the relevant sheets that contain the data you want to load. You'll see a preview on the right side. After selecting the necessary sheets, click **OK** to load them into the Query Editor.

Preparing the Data

Before merging, it's crucial to ensure that the data types are correctly set:

1. **Detect Data Types:** In the Query Editor, select the columns and click on **Transform > Detect Data Type** to ensure all columns have the appropriate data types.
2. **Use First Row as Headers:** If needed, transform the first row into headers by selecting **Use First Row as Headers** to correctly label the columns.

Merging the Queries

Now that your data is prepared, you can proceed to merge the queries:

1. **Select the Base Table:** Start by selecting the table you want to merge data into, such as the **Stores** table.

2. Merge Queries: In the Home tab, click on **Merge Queries**. You can choose to merge into the existing query or create a new one.

The screenshot shows the Power BI desktop interface with the 'Merge Queries' dialog open. The 'Merge Type' is set to 'Left Outer Join' and the 'Merge Into' option is 'New Query'. The 'Transform' tab is active. The 'Applied Steps' pane on the right shows a single step: 'Truncated Text'. The main area displays two tables, 'Table1' and 'Table2', with their columns and data visible.

3. Select the Second Table: Choose the second table you want to merge with, such as the table containing cost data.

4. Select Matching Columns: Identify the common column between the two tables (e.g., **Store ID**). Select this column in both tables to establish the relationship.

5. Choose Join Type: Power BI offers several types of joins:

- **Left Outer Join:** Keeps all rows from the first (left) table and matches them with the corresponding rows from the second table. Unmatched rows from the second table are not included.
- **Right Outer Join:** Keeps all rows from the second (right) table and matches them with the corresponding rows from the first table. Unmatched rows from the first table are not included.

- **Full Outer Join:** Includes all rows from both tables, filling in with nulls where there is no match.
- **Inner Join:** Only includes rows where there is a match in both tables.
- **Left Anti Join:** Keeps only the rows from the first table that do not have a corresponding match in the second table.
- **Right Anti Join:** Keeps only the rows from the second table that do not have a corresponding match in the first table.

For most cases, **Left Outer Join** is the most commonly used option.

Merge

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

stores

store_id	storetype_id	store size	city_id	state abr	state	city	lat	lon
S0091	ST04	19	C013	AR	Arkansas	Hot Springs National Park	34.5137	-91.0000
S0012	ST04	28	C005	TX	Texas	Huntsville	30.7813	-90.3000
S0045	ST04	17	C008	NC	North Carolina	Asheville	35.6004	-82.5000
S0032	ST03	14	C019	CA	California	Los Angeles	33.7866	-118.2000
S0027	ST01	21	C002	NY	-	-	-	-

Stores - construction costs

store_id	building-costs	land and development costs	Other related costs
S0091	3467959	1180964	421335
S0012	2205562	744172	316866
S0045	498766	170288	56093
S0032	7750009	2585892	1216210
S0027	1338071	444113	191515

Join Kind

Left Outer (all from first, matching from second)

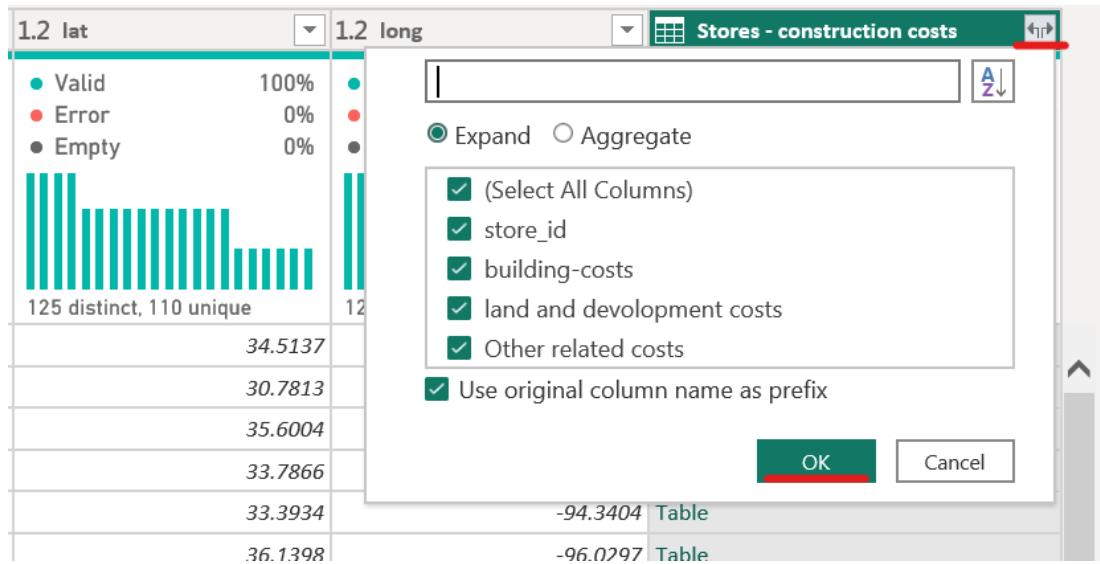
Use fuzzy matching to perform the merge

Fuzzy matching options

The selection matches 144 of 144 rows from the first table.

OK **Cancel**

6. **Expand the Merged Table:** After merging, a new column is added, typically containing a table. Click the expand icon in the header of this column to select which columns from the second table you want to include. You can also decide whether to keep the original column names as prefixes.



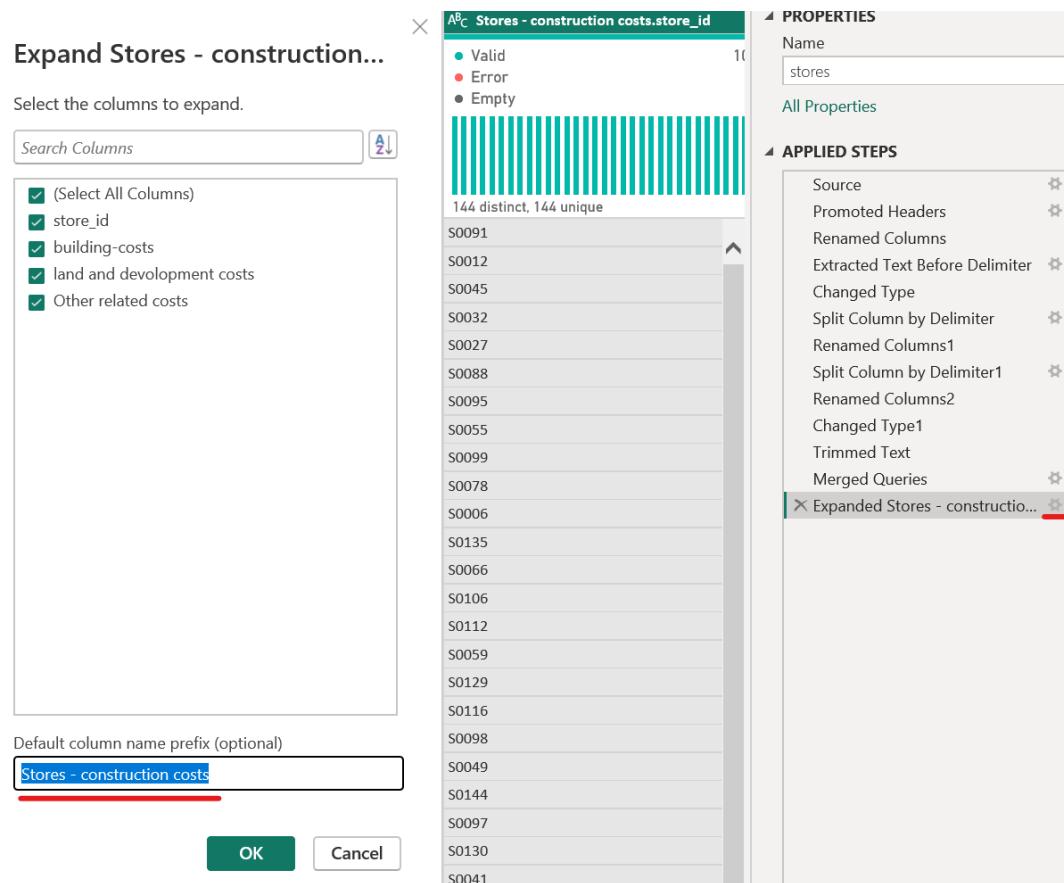
Managing Merged Data

Once merged, you may need to clean up and manage the resulting data:

- **Remove Duplicates:** If the `Store ID` appears in both tables after merging, you can remove one of the duplicate columns.
- **Modify Step Settings:** If you need to adjust any steps taken during the merge, you can click on the gear icon next to each step in the Applied Steps pane to modify the settings. Be cautious with changes, as altering a previous step can cause errors in subsequent steps.

For example, if you change or remove a column name, it could break references in later steps, leading to errors like "Column not found."

- **Fixing Errors:** If an error occurs after modifying steps, you can often resolve it by revisiting the affected steps or by reapplying the merge.



Merging queries in Power BI allows you to combine data from multiple sources into a single table, making it easier to perform comprehensive analyses. By carefully selecting join types and managing merged data, you can ensure that your tables contain the most relevant and accurate information.

▼ Pivot & Unpivot

In this section, we'll discuss the concepts of pivoting and unpivoting data in Power BI, essential techniques for restructuring data to fit your analysis needs.

Understanding Pivoting and Unpivoting

- **Pivoting** transforms rows into columns, making it easier to see categories side by side.

- **Unpivoting** converts columns into rows, which is useful for summarizing and aggregating data more effectively.

Let's explore these concepts with an example using the `Stores` table.

Unpivoting Data

Imagine you have a `Stores` table with columns for different types of costs (e.g., `Land and Development Costs`, `Building Costs`, and `Other Costs`). This layout might be challenging to work with, especially if you want to sum these costs or perform comparative analysis. Unpivoting transforms these cost columns into rows, making the data more flexible for analysis.

Steps to Unpivot Data:

1. **Open Query Editor:** Start by opening the Query Editor in Power BI.
2. **Select Columns to Unpivot:** Navigate to the `Stores` table and select the columns you want to unpivot (e.g., `Land and Development Costs`, `Building Costs`, `Other Costs`). Hold the `ctrl` key to select multiple columns.
3. **Unpivot Columns:** Go to the **Transform** tab and choose **Unpivot Columns**. This action will convert the selected columns into two new columns:

1.2 building-costs

123 land and development costs

123 Other related costs

Copy
Remove Columns
Remove Other Columns
Add Column From Examples...
Remove Duplicates
Remove Errors
Replace Values...
Fill
Change Type
Transform
Merge Columns
Sum
Product
Group By...
Unpivot Columns
Unpivot Other Columns
Unpivot Only Selected Columns
Move

que

144 distinct, 144 unique

	1180964	421335
744172	316866	
170288	56093	
2585892	1216210	
444113	191515	
1809581	721278	
1464663	474972	
1358411	379994	
2074759	689460	
1684553	373194	
1789534	518395	

- **Attribute:** Contains the original column names (e.g., `Building Costs`).
- **Value:** Contains the corresponding data values.

Queries [7]

Helper Tables [2]

Sales 2017

Sales 2018

Sales 2019

Other Queries [4]

product hierarchy

sales

Stores - countries

Attribute

Value

Attribute	Value
store_id	100% Valid, 0% Error, 0% Empty
store_name	100% Valid, 0% Error, 0% Empty
state_id	100% Valid, 0% Error, 0% Empty
city_id	100% Valid, 0% Error, 0% Empty
state	100% Valid, 0% Error, 0% Empty
city	100% Valid, 0% Error, 0% Empty
district	100% Valid, 0% Error, 0% Empty
store_sqft	100% Valid, 0% Error, 0% Empty
state_sqft	100% Valid, 0% Error, 0% Empty
city_sqft	100% Valid, 0% Error, 0% Empty
district_sqft	100% Valid, 0% Error, 0% Empty
lat	100% Valid, 0% Error, 0% Empty
long	100% Valid, 0% Error, 0% Empty
attribute	100% Valid, 0% Error, 0% Empty
value	100% Valid, 0% Error, 0% Empty

4. **Rename Columns:** It's a good practice to rename these columns for clarity. For example, rename `Attribute` to `Cost Type` and `Value` to `Cost`.

This unpivoted structure allows you to sum costs across different types and visualize them more effectively.

Potential Issues with Unpivoting:

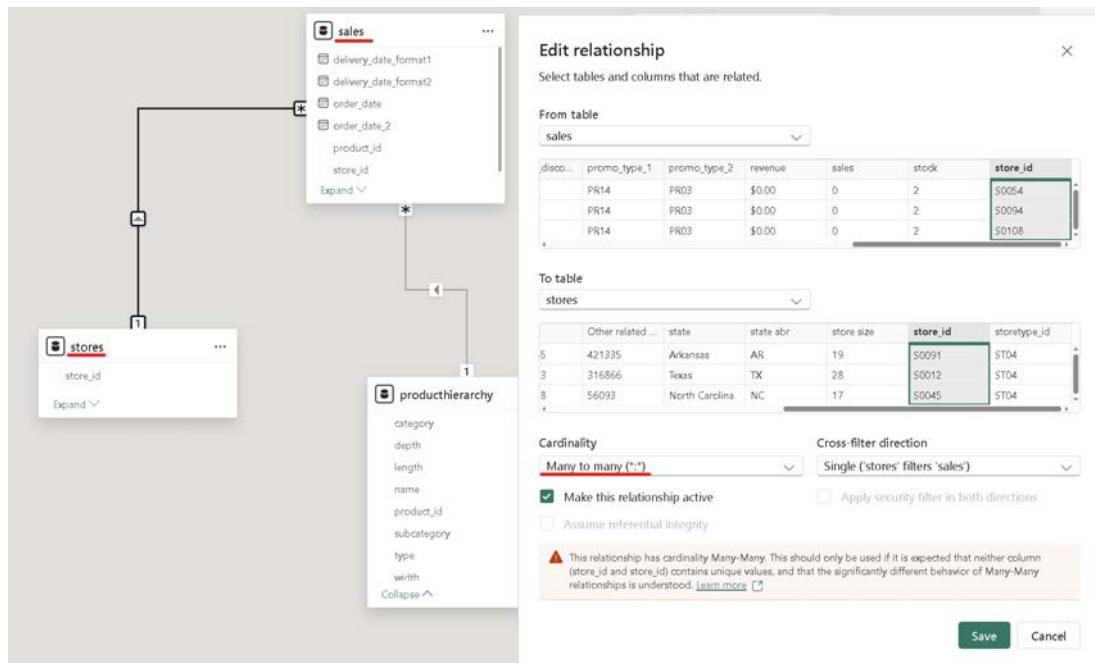
Unpivoting may affect your data model, particularly relationships. For example, if `Store ID` is part of a one-to-many relationship, unpivoting could

introduce duplicate **Store ID** values, leading to errors. Power BI will alert you with an error message like:

"Column Store ID contains a duplicate value. This is not allowed for columns on the one side of a many-to-one relationship."

To resolve this:

- **Change Relationship Type:** Modify the relationship from one-to-many to many-to-many. Be cautious, as many-to-many relationships can complicate your model and are less common.



After unpivoting, you can visualize your data more effectively, such as by using a stacked bar chart to break down costs by type.

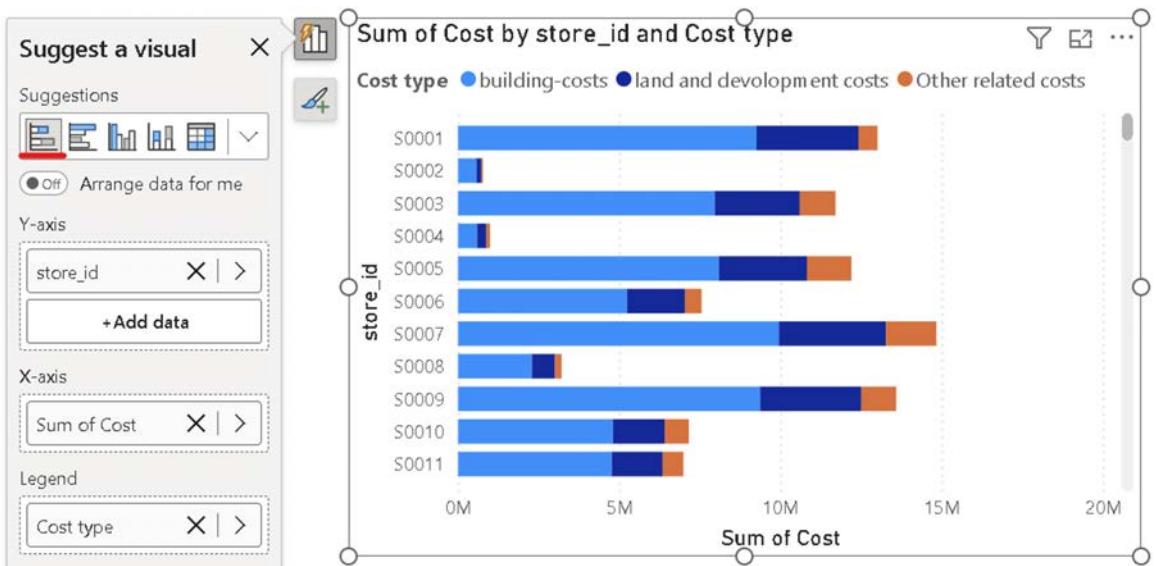
Suggest a visual

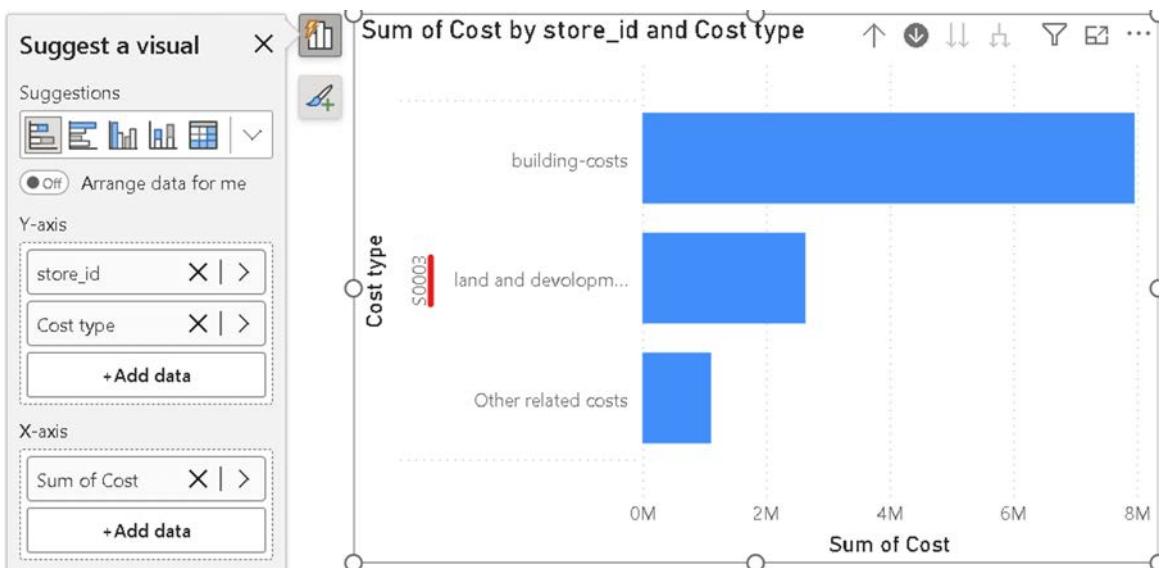
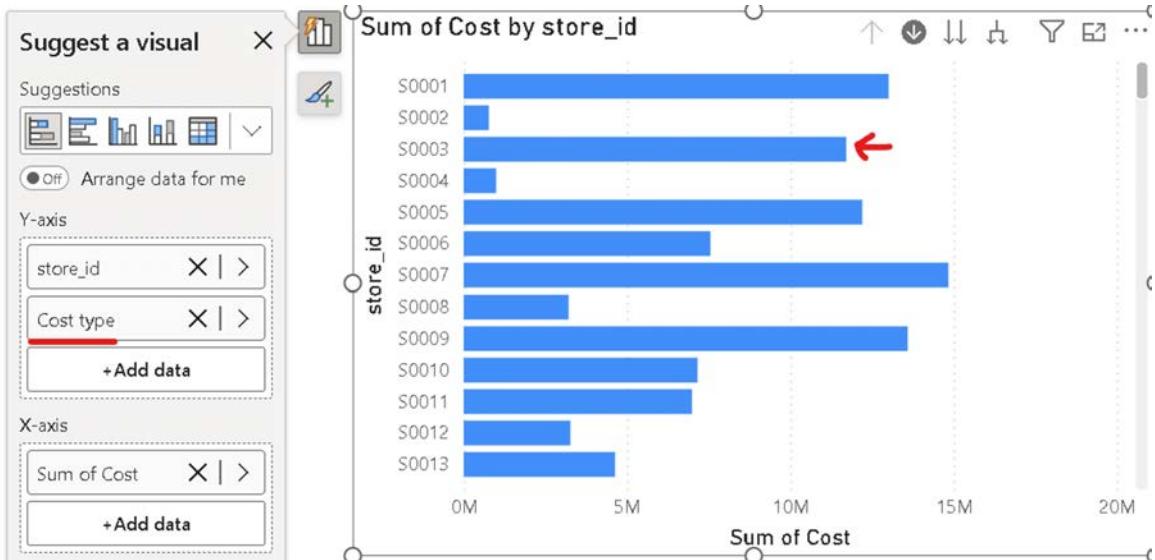
Columns

- store_id
- Cost type
- Sum of Cost

+ Add data

store_id	Cost type	Sum of Cost
S0001	building-costs	9,258,470.00
S0001	land and development costs	3,163,188.00
S0001	Other related costs	587,734.00
S0002	building-costs	569,314.00
S0002	land and development costs	159,262.00
S0002	Other related costs	42,444.00
S0003	building-costs	7,962,313.00
S0003	land and development costs	2,638,015.00
S0003	Other related costs	1,110,819.00
S0004	building-costs	590,842.00
S0004	land and development costs	291,449.00
S0004	Other related costs	111,664.00
S0005	building-costs	8,098,244.00
Total		1,000,479,871.00





Pivoting Data

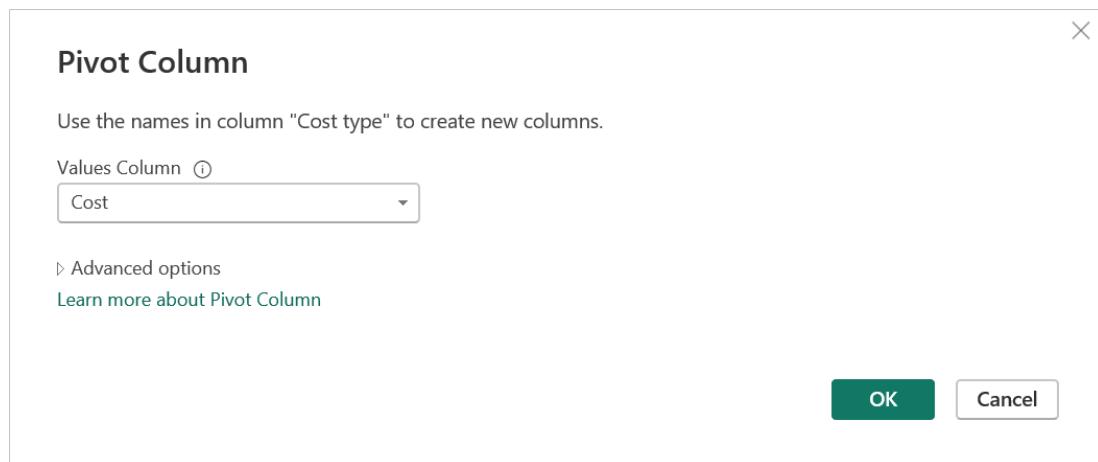
Pivoting is the reverse of unpivoting. It transforms rows back into columns, useful when you need to aggregate or distribute data across multiple columns.

Steps to Pivot Data:

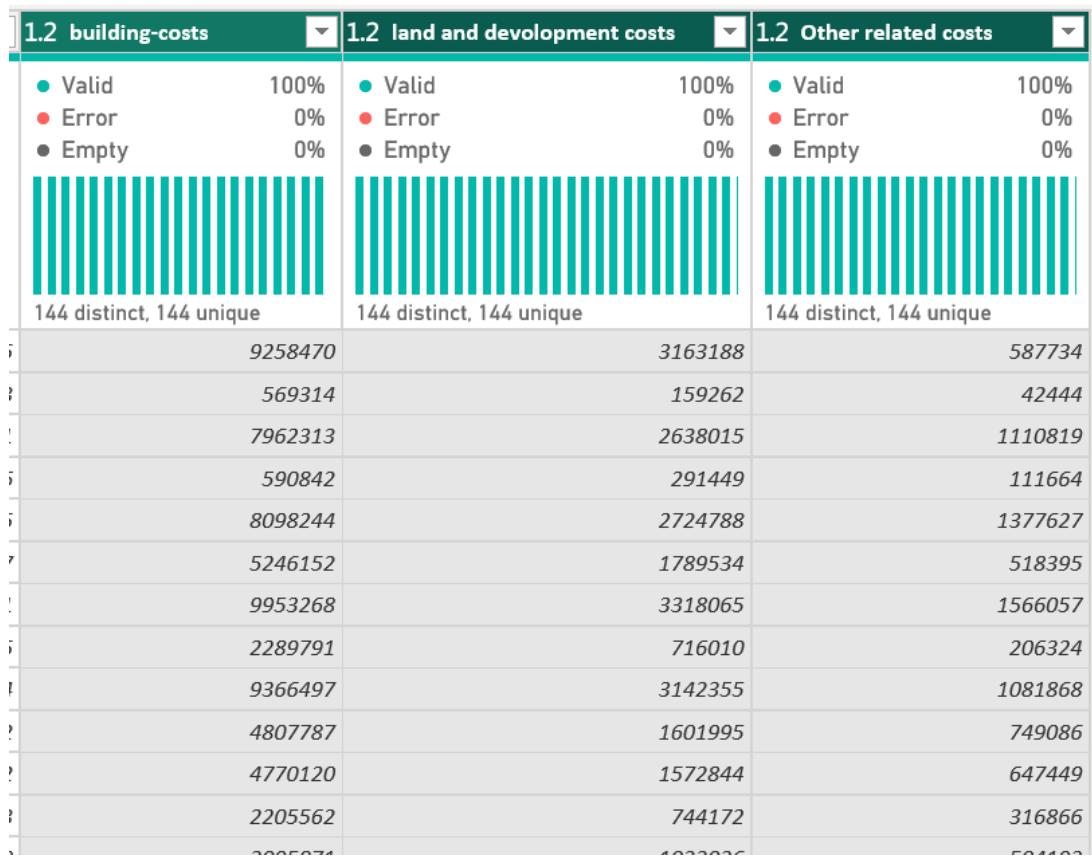
- Select the Column to Pivot:** In the Query Editor, select the column containing the categories you want to pivot (e.g., **Cost Type**).
- Pivot Column:** Under the **Transform** tab, select **Pivot Column**.

The screenshot shows the Power BI ribbon with the 'Transform' tab selected. Under the 'Pivot Column' section, the 'Pivot Column' button is highlighted. The main area displays a table with columns: '1.2 lat', '1.2 long', 'Cost type', and '1.2 Cost'. Each column has a summary bar chart above it showing counts for 'Valid', 'Error', and 'Empty' values.

3. **Choose the Value Column:** Select the column that contains the values to be spread across the new columns (e.g., `Cost`).



4. **Complete the Pivot:** Click **OK** to pivot the data. The values from the `Cost` column will be distributed across new columns named after the unique values in the `Cost Type` column.



Pivoting is useful when you need to return to a wide format after performing analysis on the unpivoted data.

Pivoting and unpivoting are powerful tools in Power BI that allow you to restructure your data according to your analysis needs. By understanding when and how to use these techniques, you can enhance the flexibility and effectiveness of your data models and visualizations.

▼ Many-to-Many relationships & Crossfilter Direction

In this section, we'll explore the concept of many-to-many relationships in Power BI, as well as how crossfilter direction impacts the behavior of these relationships in your data model.

Understanding Many-to-Many Relationships

A **many-to-many relationship** occurs when multiple records in one table relate to multiple records in another table. This type of relationship is more complex than the standard one-to-many or many-to-one relationships and requires careful handling in Power BI.

Scenario Example:

Suppose you have a

`Stores` table and a `Sales` table:

- The `Stores` table contains store details, including `Store ID`.
- The `Sales` table logs sales transactions, also using `Store ID`.

If each store can appear multiple times in the `Sales` table (which it likely will) and there's a possibility that a store ID could be repeated in another table (like a `Costs` table), you're dealing with a many-to-many relationship.

Creating and Understanding the Relationship

To create or view a relationship in Power BI:

1. **Close and Apply:** After making changes in the Query Editor, click on **Close & Apply**.
2. **Access the Data Model:** Go to the **Model** view to visualize and manage relationships between tables.
3. **View Relationship:** Double-click on the relationship line between the tables to open the relationship settings. Here, you'll see options for cardinality (e.g., many-to-many) and crossfilter direction.

Crossfilter Direction

Crossfilter direction controls how filters are applied across related tables in Power BI.

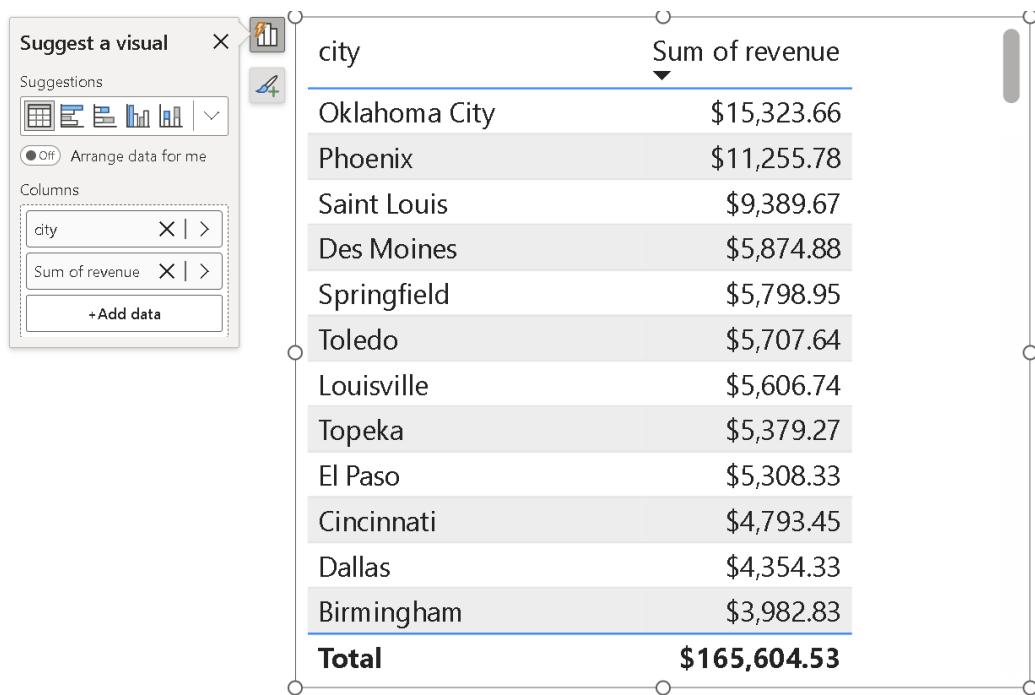
- **Single (One-way) Filtering:** By default, the filter direction is set to single, meaning that filtering occurs in one direction only—from one table to another. For example, filtering by `Store ID` in the `Stores` table affects the `Sales` table but not vice versa.
- **Both (Bidirectional) Filtering:** This option allows filters to flow both ways between tables. It's useful in certain scenarios, particularly in

many-to-many relationships, but it should be used sparingly due to performance concerns.

Demonstrating Crossfilter Direction:

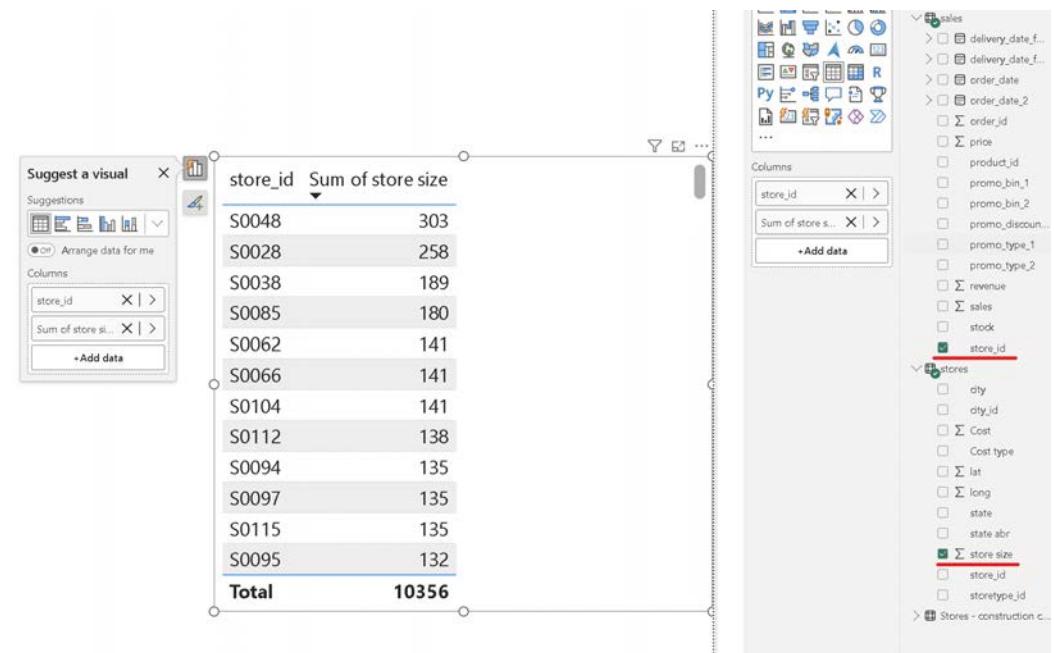
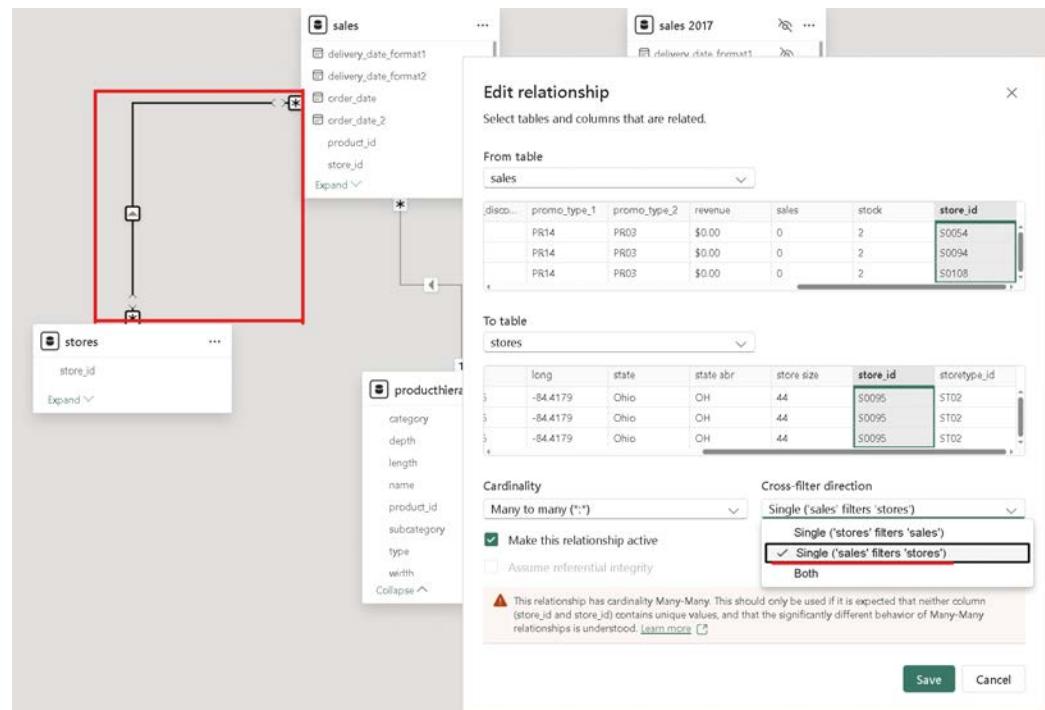
1. Single Direction Filtering:

- Let's say you have a report where you want to filter sales data by `City` from the `Stores` table and aggregate `Revenue` from the `Sales` table. With a single direction filter from `Stores` to `Sales`, this works correctly because the city filter applies to the sales data.



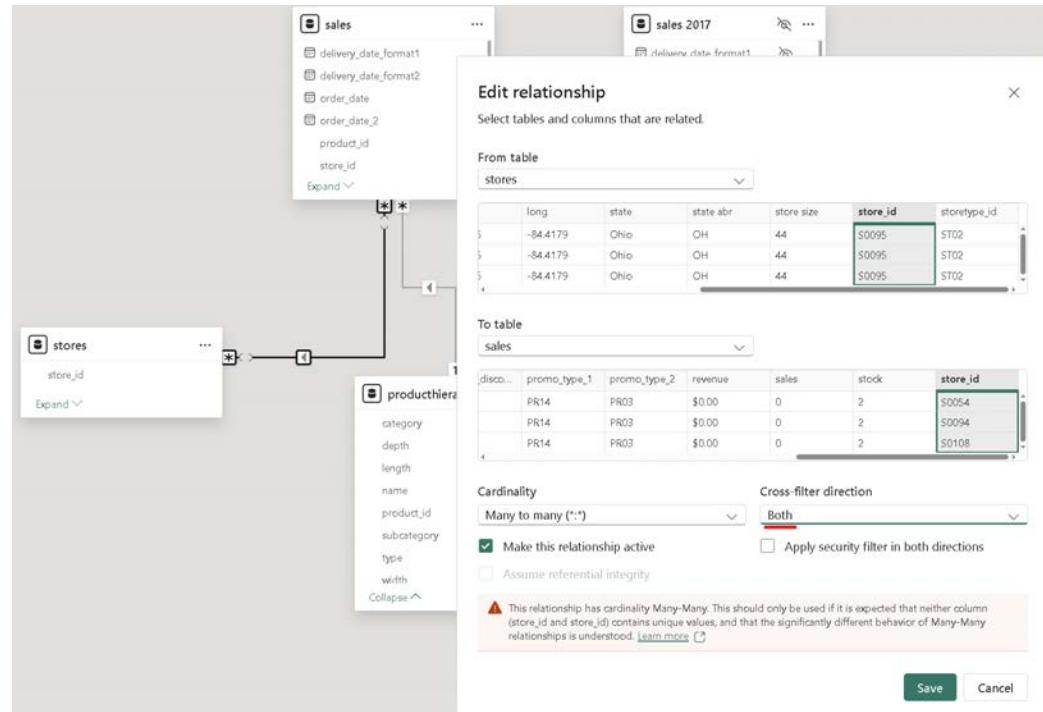
2. Reversing the Filter Direction:

- If you try to filter data in the opposite direction (e.g., using a field from the `Sales` table to filter data in the `Stores` table), it might not work as expected if the filter direction is not set correctly. For instance, using `Store ID` from `Sales` to aggregate `Store Size` from `Stores` would produce identical values for all records if the filter direction doesn't allow this.



3. Setting Bidirectional Filtering:

- To enable filtering in both directions, you can change the crossfilter direction to **Both**. This allows filters to flow both from **Sales** to **Sales** and vice versa, but it should be used only when necessary, as it can slow down the data model due to increased computational demands.



Best Practices for Using Crossfilter Direction

1. Filter from Dimension to Fact Tables:

- Typically, you should filter from dimension tables (e.g., `Stores`) to fact tables (e.g., `Sales`). This approach aligns with how data is usually aggregated and simplifies your data model.

2. Avoid Bidirectional Filtering:

- Unless absolutely necessary, avoid using the **Both** option for crossfilter direction. It's more compute-intensive and can degrade performance. Instead, structure your queries and reports to utilize single-direction filtering whenever possible.

3. Handle Many-to-Many Relationships with Care:

- Many-to-many relationships add complexity to your model. Always consider if there's a way to simplify the relationship, such as by restructuring the data or using bridge tables.

Understanding and managing many-to-many relationships and crossfilter direction in Power BI is crucial for building effective data models. By

following best practices, you can ensure your reports are both accurate and performant, making your data analysis more reliable and efficient.

▼ 4- Interactive Visualizations

▼ Slicer Visual

In this section, we'll explore the Slicer visual in Power BI, a crucial tool for interactive filtering in reports. Slicers allow users to dynamically filter data, enhancing the report's usability and customization.

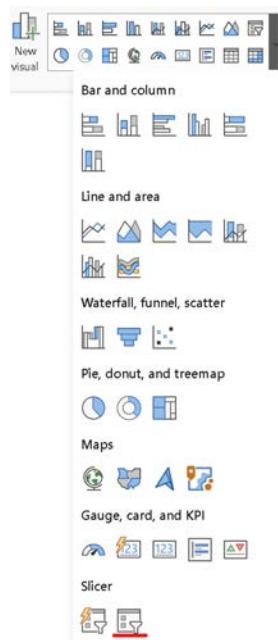
Introduction to Slicer Visual

The Slicer visual is a powerful filtering tool that lets you control which data is displayed in your report. Whether you're focusing on specific stores, cities, or other dimensions, slicers make it easy to narrow down the data set based on your needs.

Adding and Configuring a Slicer

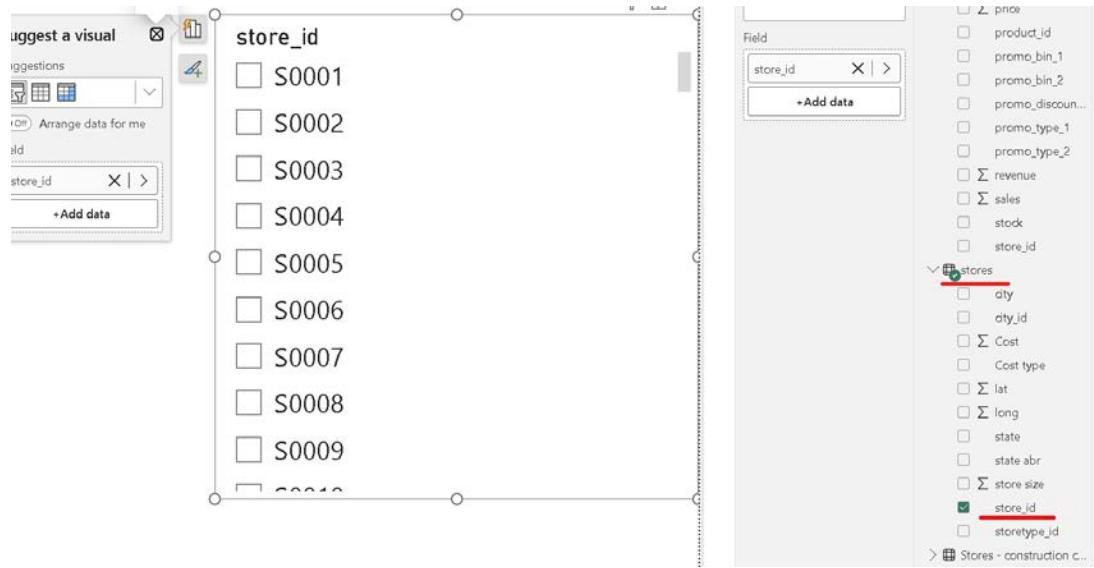
1. Adding a Slicer:

- To add a slicer to your report, go to the **Visualizations** pane and select the Slicer icon. This adds a new slicer to your report canvas.



2. Configuring the Slicer:

- Drag a field, such as `Store ID`, from the `Fields` pane into the slicer. This allows you to filter the report based on specific store IDs.
- You can select multiple stores by holding the `Ctrl` key or by configuring the slicer settings to allow easier multi-selection.

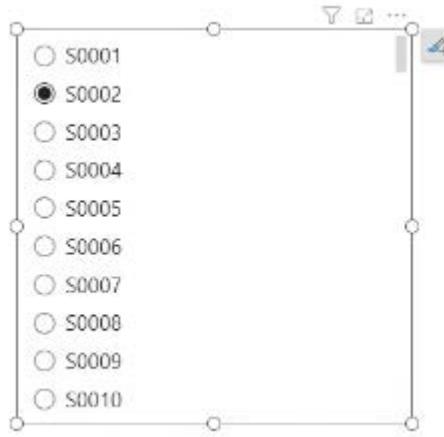


Slicer Settings and Customization

Power BI provides several options to customize slicers:

1. Multi-Select with Ctrl:

- By default, Power BI requires you to hold the `Ctrl` key to select multiple items. This behavior can be changed:
 - **Single Select:** Allows only one selection at a time.



- **Multi-Select:** You can select multiple items without holding the **ctrl** key. This is more intuitive for users who frequently need to select multiple values.

store_id

- S0001
- S0002
- S0003
- S0004
- S0005
- S0006
- S0007
- S0008
- S0009
- S0010

Slicer Options

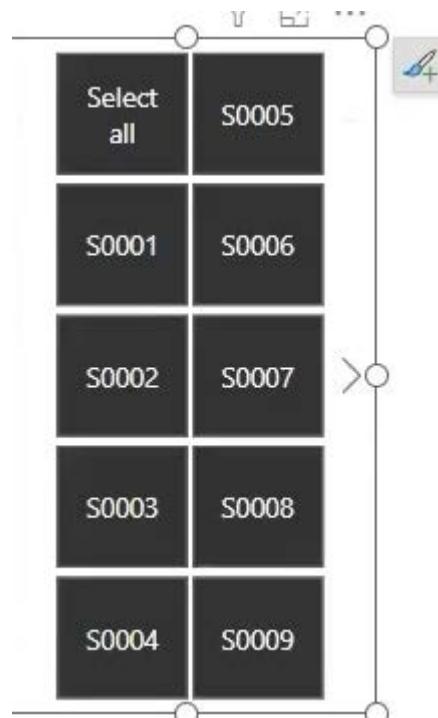
- Style: Vertical list
- Selection:
 - Single select (Off)
 - Multi-select with C... (On) (Underlined)
 - Show "Select all" o... (Off)
- Reset to default
- Slicer header: On
- Values

2. Slicer Styles:

- **Vertical List:** Displays items in a list format. This is the default and is easy to use for most scenarios.
- **Dropdown:** Converts the slicer into a dropdown menu, saving space on the report canvas. This option is ideal when screen real estate is limited.



- **Tiles:** Displays items as selectable tiles. While visually appealing, tiles can take up more space and might not be as intuitive as lists or dropdowns.



3. Slicer Header and Title:

- You can customize or remove the slicer header. This is useful if you want to reduce clutter or if the slicer's purpose is already clear from context.

4. Select All Option:

- Enabling the "Select All" option can be convenient when users frequently need to include all items in their selection. This option adds a checkbox for selecting all values at once.

5. Hierarchical Slicers:

- You can create hierarchical slicers by adding multiple fields to the slicer. For example, first select `city` and then `store_id`. This allows users to filter by city first and then drill down to specific stores within that city.



Advanced Slicer Use Cases

Slicers can also be configured for more complex scenarios, such as filtering by numerical values:

1. Using Numerical Values in Slicers:

- You can add numerical fields (e.g., `Price`, `Cost`) to a slicer to filter based on ranges. Power BI provides options like "Between," "Less Than," and "Greater Than" for filtering numerical values.



- **Crossfiltering Considerations:** Be mindful of the data model's crossfilter direction. If you're filtering from `Sales` to `Stores`, ensure the relationship supports the desired filtering direction, or adjust the crossfilter settings accordingly.

Best Practices for Slicers

- **Keep It Simple:** Use slicers that are easy for users to understand and interact with. Avoid overly complex configurations unless necessary.
- **Optimize Space:** Use dropdowns or hierarchical slicers to save space on the report canvas.
- **Test Filtering Behavior:** Always check that slicers filter data as expected, particularly in complex models where crossfilter direction can impact the results.

The Slicer visual in Power BI is an essential tool for making reports interactive and user-friendly. By understanding and utilizing the various settings and customization options, you can create slicers that meet your specific data filtering needs, ensuring that your reports are both effective and efficient.

▼ Edit Page

In this section, we'll discuss how to enhance the visual appeal of your Power BI report by formatting the page itself. Proper page design is crucial for creating professional, readable, and visually appealing reports.

Accessing Page Format Options

To start editing your page design:

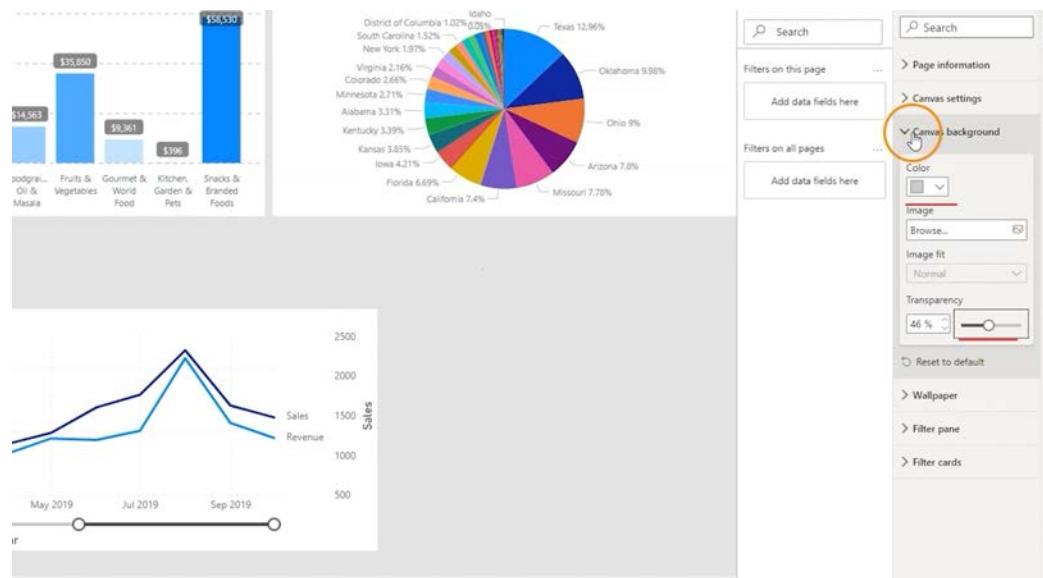
1. **Select the Page:** Click anywhere on the report canvas where no visual is selected to access the page format options.
2. **Open the Format Pane:** If the Format Pane isn't visible, you can open it by clicking on the paint roller icon in the Visualizations pane.

Customizing the Canvas Background and Wallpaper

Power BI allows you to customize both the canvas background (the area where your visuals reside) and the wallpaper (the area surrounding the canvas):

1. Canvas Background:

- **Set a Background Color:** Choose a background color for the canvas. For example, you might opt for a darker shade to contrast with lighter visuals.
- **Adjust Transparency:** Use the transparency slider to control the intensity of the background color. Reducing transparency to 0% makes the background fully opaque, while increasing it makes the color more subtle.



2. Wallpaper:

- **Set a Wallpaper Color:** The wallpaper extends beyond the canvas, covering the entire page. You can choose a color that complements your canvas background.
- **Add an Image:** You can upload an image to use as wallpaper. For best results, adjust the image fit:
 - **Fit:** Resizes the image to fit within the entire wallpaper area.

- **Fill:** Stretches the image to cover the entire area, which might distort the image.



- **Adjust Transparency:** Similar to the canvas, you can adjust the wallpaper's transparency to make the image less distracting. This is particularly useful if you want to create a subtle background effect.

3. Balancing Design Elements:

- **Reduce Distractions:** While background images can add visual interest, they can also be distracting. It's often best to keep them subtle, ensuring that the focus remains on the data and insights.

Customizing Visuals and Visual Backgrounds

After setting up the canvas and wallpaper, you might want to adjust the visuals themselves to ensure they stand out:



1. Visual Backgrounds:

- **Set Background Color and Transparency:** Select a visual and navigate to the **Background** option under the **Format** pane. You can set a background color and adjust its transparency to help the visual pop against the page background.

2. Consistent Styling:

- **Apply Uniform Styles:** For a cohesive look, apply similar background settings to all visuals on the page. For example, if you set one visual to 30% transparency, consider applying the same setting to others.

Formatting the Filter Pane and Other Elements



1. Filter Pane Customization:

- Adjust Filter Pane Background:** You can customize the filter pane by selecting it and setting a background color and transparency, similar to the canvas and visuals. This helps integrate the filter pane with the overall design of the report.

2. Input Boxes:

- Customize Input Box Appearance:** If your report includes input boxes, you can adjust their background color to match the overall design. For example, setting the input box to white with a slight transparency can help it blend seamlessly into the report while remaining functional.

Best Practices for Page Design

- Focus on Data:** While it's important for the report to look good, the primary focus should always be on the data. Avoid overly complex or distracting backgrounds that might take attention away from the information being presented.
- Play with Settings:** Experiment with different settings to find the right balance that enhances readability and aesthetics without overwhelming the user.

- **Consistency is Key:** Maintain a consistent design language across all pages of the report to create a professional and polished look.

Editing the page design in Power BI involves a combination of customizing the canvas, wallpaper, and visual backgrounds to create a visually appealing and user-friendly report. By balancing aesthetics with functionality, you can ensure that your report not only looks good but also effectively communicates your data insights.

▼ Project 5

[Projects5.rar](#)

▼ Filter Pane

In this section, we'll explore the Filter Pane in Power BI, a powerful tool that allows report developers to control how data is filtered across visuals, pages, and the entire report.

Understanding the Filter Pane vs. Slicer Visual

Before diving into the Filter Pane, it's important to distinguish between the **Filter Pane** and the **Slicer Visual**:

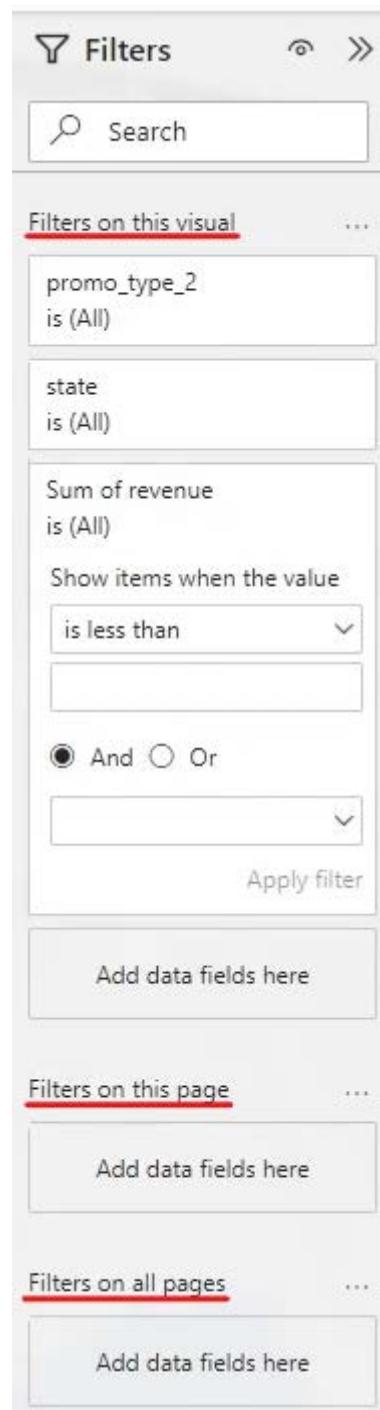
- **Slicer Visual:** This is an interactive tool that end users can manipulate directly on the report canvas. It allows users to dynamically filter the report content as they explore the data.
- **Filter Pane:** Primarily used by report developers, the Filter Pane enables more controlled filtering of data. Filters applied here can be set to apply to individual visuals, specific pages, or the entire report. Once configured, the Filter Pane can be hidden from end users if needed.

Navigating the Filter Pane

To begin working with the Filter Pane:

1. Access the Filter Pane: Ensure the Filter Pane is visible by clicking on the arrow to expand it if it's collapsed.

2. Sections of the Filter Pane:



- **Filters on this Visual:** Appears when a visual is selected, allowing you to apply filters specific to that visual.
- **Filters on this Page:** Filters data across all visuals on the current page.
- **Filters on all Pages:** Applies filters across every page in the report.

Applying Filters

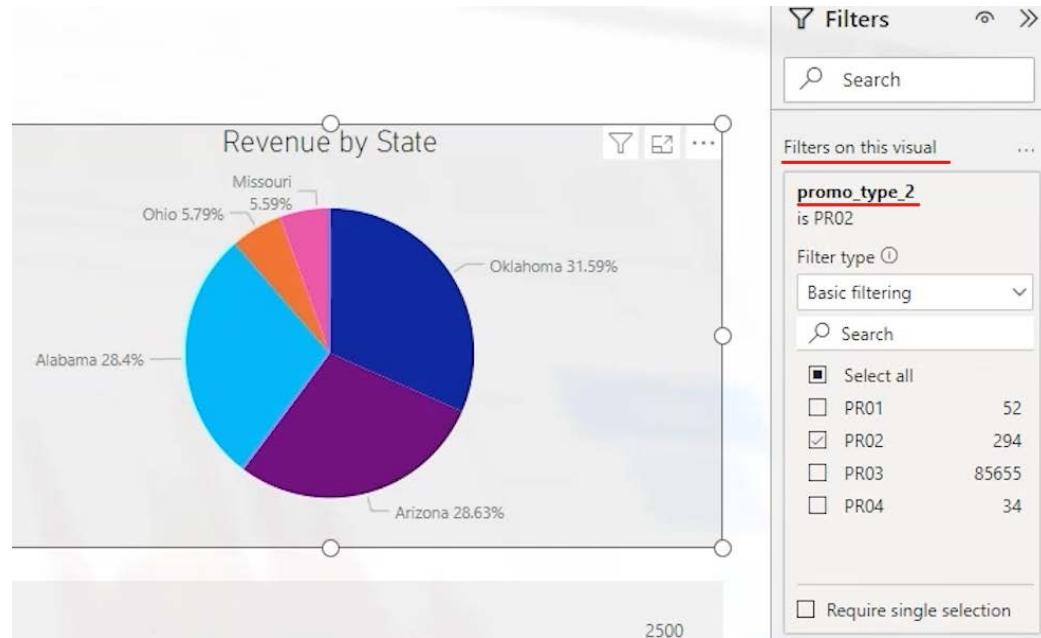
Filters can be applied by dragging and dropping fields into the relevant section of the Filter Pane.

1. Adding Filters:

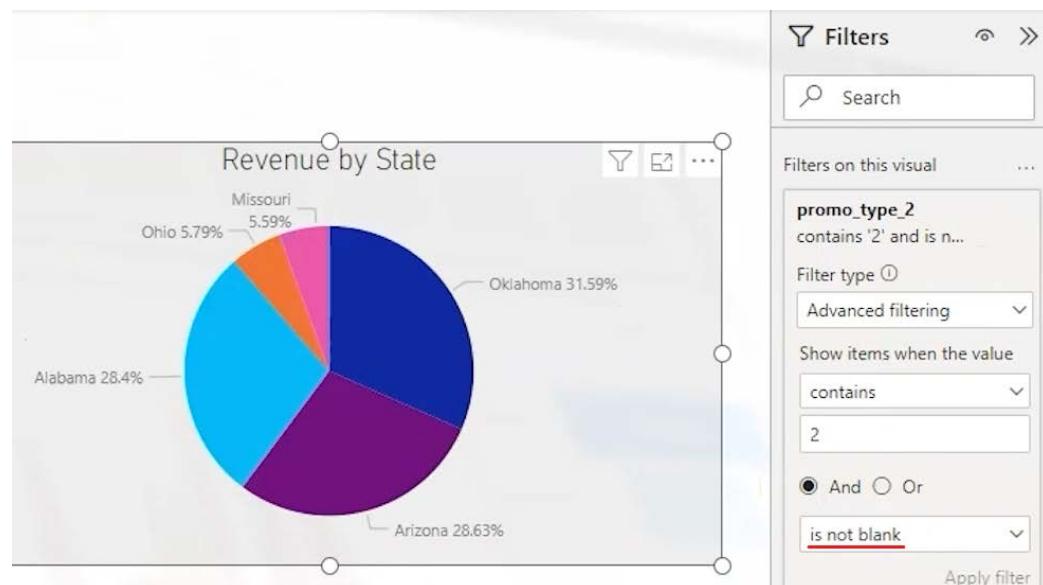
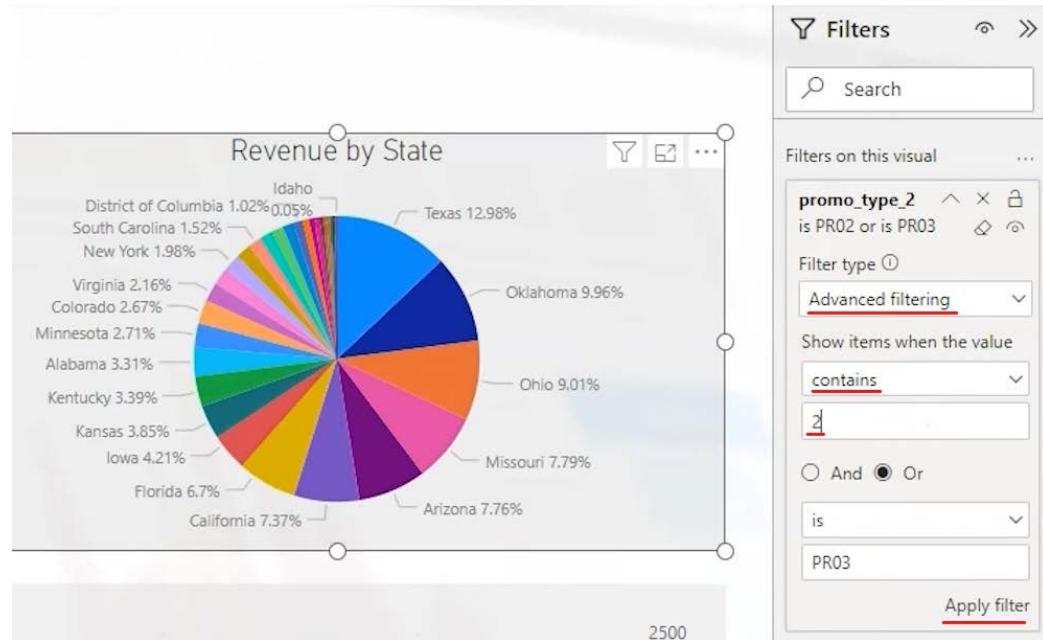
- Drag a field, such as `State` or `Revenue`, from the Fields Pane into one of the filter sections.
- You can filter visuals based on fields that aren't directly displayed in the visual itself. For instance, you might filter by `promo_type2` even if it isn't a visible field in the visual.

2. Filter Types:

- **Basic Filtering:** For categorical fields like `State` or `Promo Type`, you can select specific values to include or exclude.



- **Advanced Filtering:** Allows for more complex conditions, such as filtering text fields that contain, start with, or are not blank.



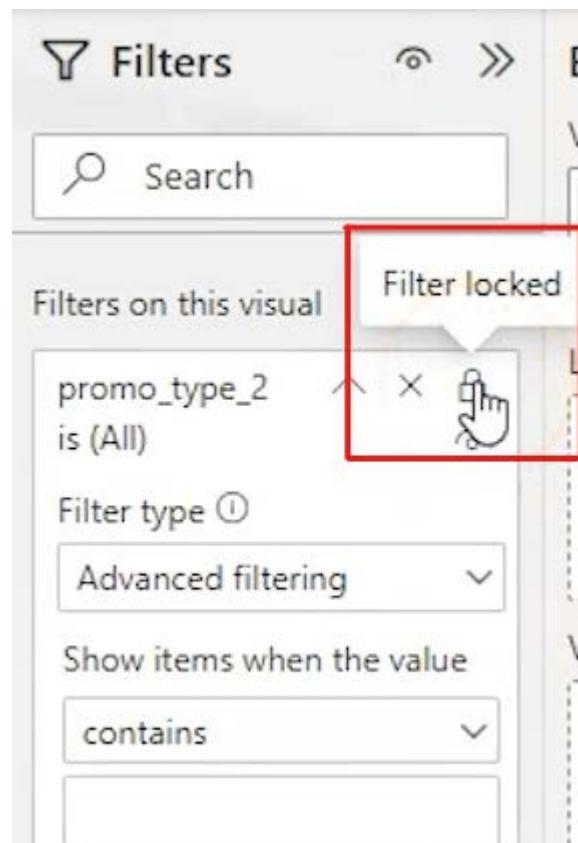
- **Top N:** "Top N" filter type in the Filter Pane allows you to filter data to display only the top (or bottom) N values based on a specific metric.

Locking and Hiding Filters

Filters applied via the Filter Pane can be locked or hidden to control how end users interact with them:

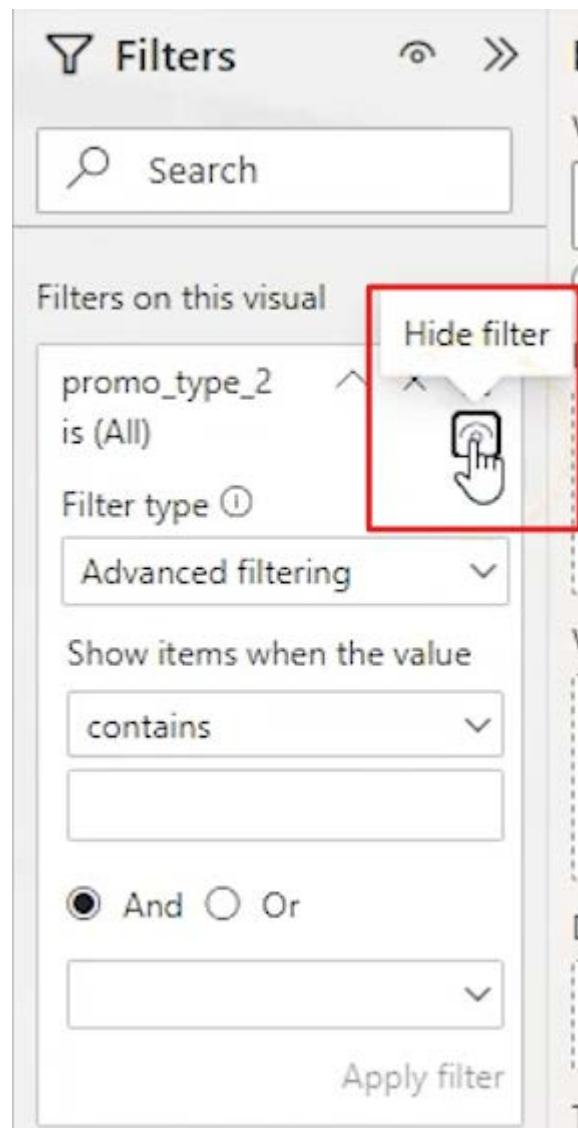
1. Locking Filters:

- Locking a filter prevents end users from altering it once the report is published. This ensures that critical filters remain consistent.



2. Hiding Filters:

- You can hide individual filters or the entire Filter Pane from end users. This is useful when you want to apply background filtering without exposing these options to users.

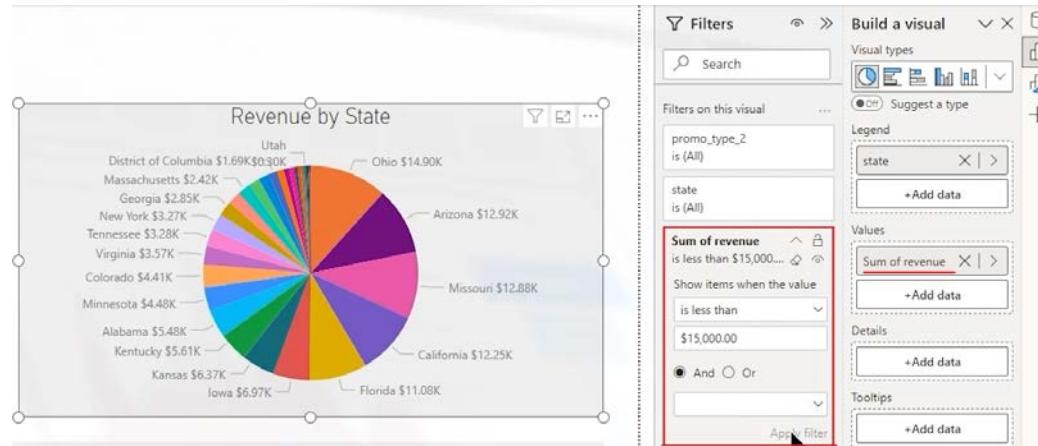


Working with Aggregated Data

Power BI allows you to apply filters to aggregated data directly within the Filter Pane:

1. Filtering Aggregated Values:

- When you apply a filter to an aggregated field like `Sum of Revenue`, Power BI filters based on the total for that category. For instance, filtering "Sum of Revenue is less than 15,000" will hide any categories where the total revenue exceeds that threshold.



2. Top N Filtering:

- Power BI also provides the option to filter by the top N values, which allows you to show only the top or bottom categories based on a specific measure. This advanced filtering technique is particularly useful for highlighting key data points.(It will be examined in detail in the next section.)

Best Practices

- Use Filters Strategically:** Apply filters in the Filter Pane to control which data is visible in your reports. This is especially useful for excluding irrelevant or sensitive data.
- Lock or Hide Filters When Needed:** Lock filters to prevent changes and hide them to maintain a clean user interface.
- Leverage Advanced Filtering:** Use advanced and top N filtering to create more targeted and insightful reports.

The Filter Pane in Power BI is a powerful tool that allows for precise control over how data is filtered and displayed in your reports. By understanding and effectively using this feature, you can ensure that your reports are not only visually appealing but also tailored to show exactly the data that matters most.

▼ Using Top N & Relative Date Filters

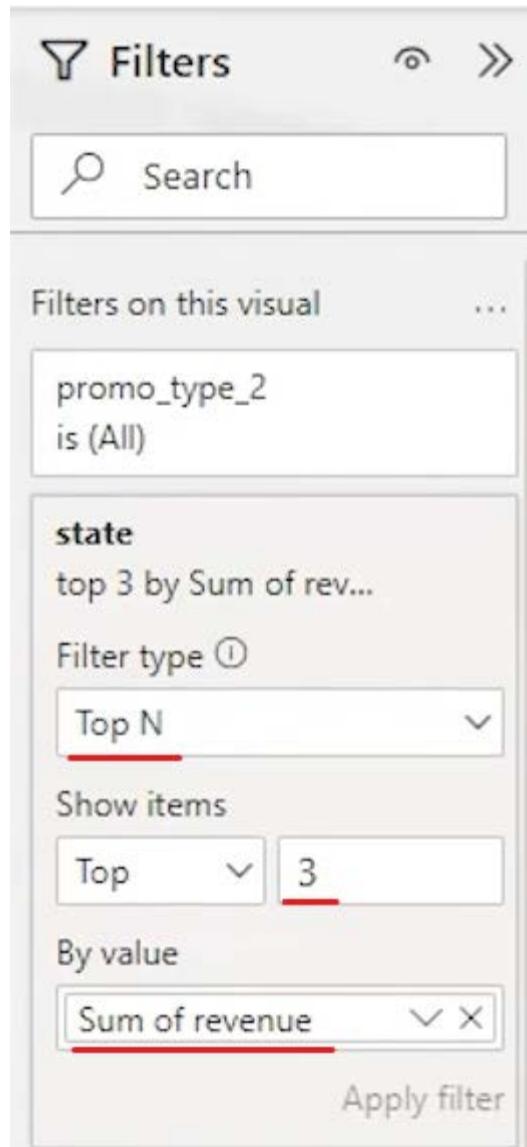
In this section, we will explore two powerful filtering options in Power BI: **Top N Filter** and **Relative Date Filter**. These filters are essential tools for dynamically controlling data visibility in your reports, enabling you to highlight key insights effectively.

Top N Filter

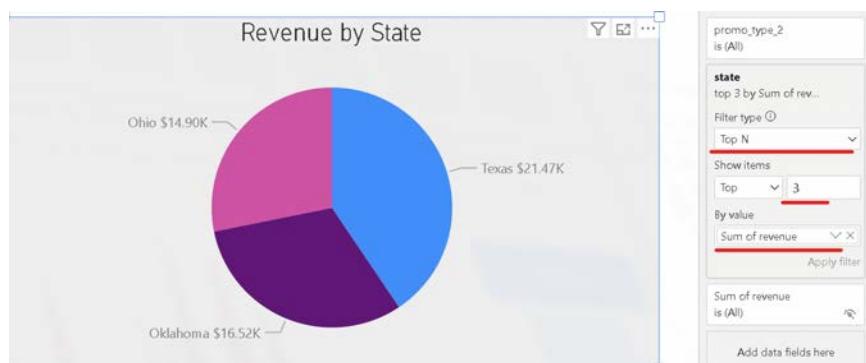
The Top N filter allows you to display a specific number of top or bottom items based on a selected measure, such as revenue or cost. This filter is particularly useful when you want to focus on the highest or lowest performing entities, like states, products, or salespeople.

Steps to Apply Top N Filter:

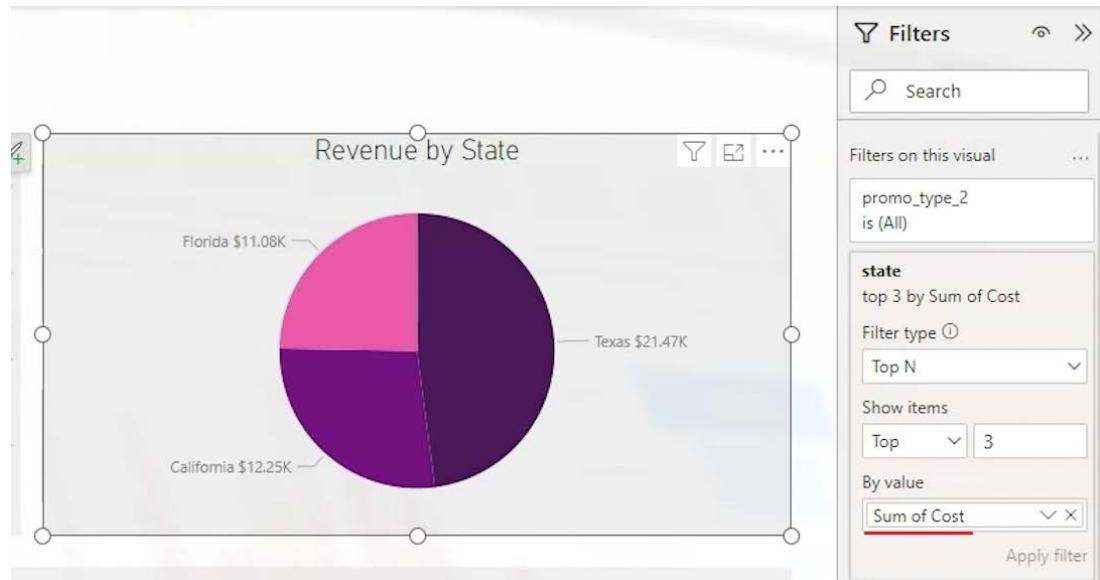
1. **Select a Visual:** When you select a visual on your report, the Filters pane will show options to filter that specific visual.
2. **Add a Field to the Filter:** Drag the field you want to filter by, such as "State," into the Filters pane.
3. **Choose Top N Filter Type:** In the filter options, select "Top N."
4. **Specify the Number of Items:** Enter the number of top or bottom items you want to display. For example, to see the top three states by revenue, enter "3" in the input box.
5. **Define the Measure:** Drag a measure, such as "Revenue," into the "By value" box to filter the top three states based on the total revenue. Ensure that the aggregation method is set appropriately, usually to "Sum."



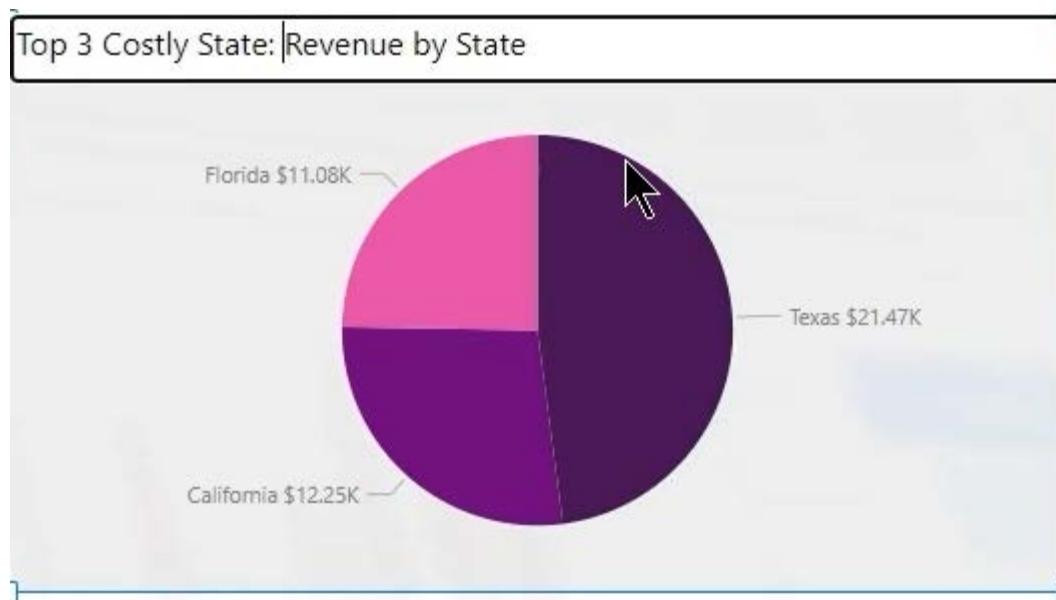
6. **Apply the Filter:** Click "Apply filter" to see the results. The visual will now display the top three states by revenue.



7. **Dynamic Filtering:** You can also switch from showing the top items to showing the bottom items, or you can change the measure to something else, like "Cost," to see the most costly states.



8. **Customize the Visual Title:** It's important to update the visual title to reflect the filter applied, e.g., "Top 3 States by Revenue" or "Top 3 Most Costly States." This ensures clarity for report users.



Important Notes:

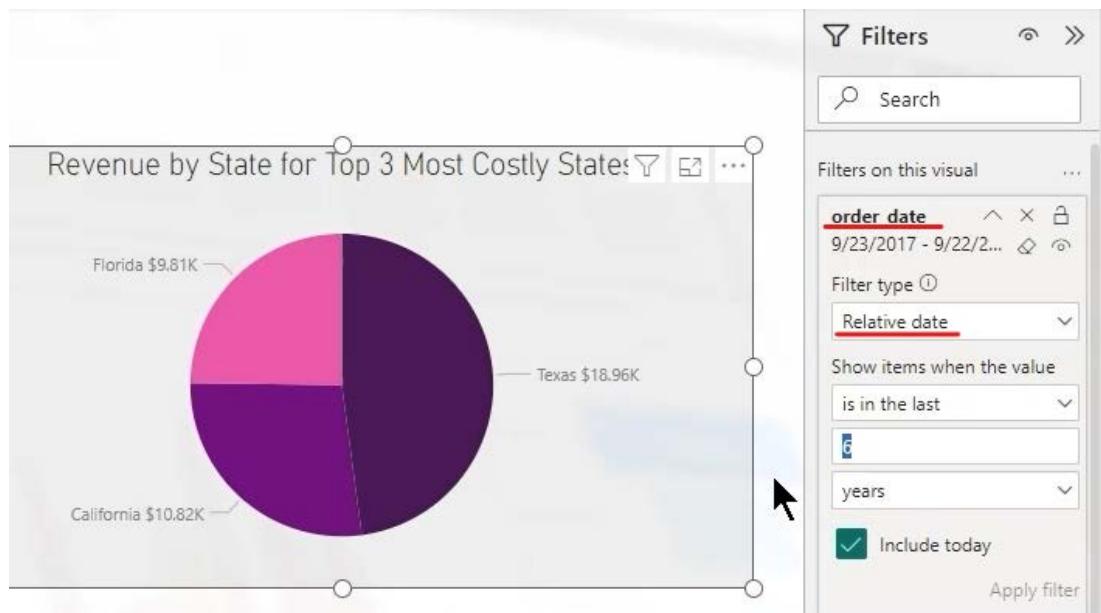
- Be cautious when using pie charts with Top N filters, as pie charts suggest a percentage of a whole, which may not be accurate in this context.
- Always ensure that the visual title clearly reflects the data being filtered.

Relative Date Filter

The Relative Date filter allows you to dynamically filter data based on date ranges relative to the current date, such as the last 7 days, this month, or the previous year. This is particularly useful for tracking trends over recent periods without needing to update the date range manually.

Steps to Apply Relative Date Filter:

1. **Select a Date Field:** Drag a date field into the Filters pane.
2. **Choose Relative Date Filtering:** In the filter options, select "Relative Date" from the available filter types.
3. **Set the Time Frame:** Define the relative time frame you want to filter by, such as "In the last 30 days" or "In the last 6 years."
4. **Include the Current Date (Optional):** You can choose to include today in the range, which can be useful for real-time data analysis.
5. **Apply the Filter:** Once set, the visual will display data based on the selected relative date range.



Both the Top N and Relative Date filters in Power BI provide powerful ways to refine your data view dynamically. By using these filters, you can create more focused and insightful reports that automatically adjust to reflect the most relevant data, saving you time and enhancing report usability.

▼ Syncing Slicers

In Power BI, slicers are an essential tool for filtering data across your reports. However, there are cases where you may want the same slicer to filter data consistently across multiple report pages. This is where **Sync Slicers** come into play. Syncing slicers ensures that when a selection is made on one page, the same selection is automatically applied on other pages, maintaining a consistent filter throughout your report.

Why Sync Slicers?

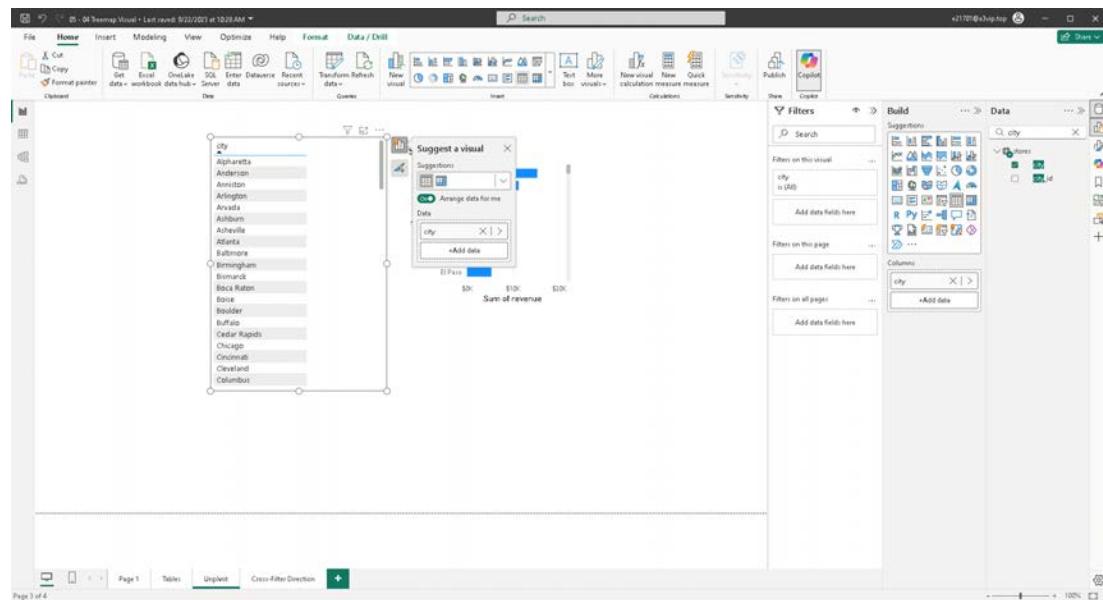
Often, you may need to filter data based on a specific criterion, such as a city, and you want this filter to apply consistently across all pages of your report. Without syncing, you'd have to manually select the same filter on each page, which is inefficient and prone to errors. Sync Slicers solve this problem by linking slicers across pages, ensuring that your filters are consistent throughout your analysis.

Scenario:

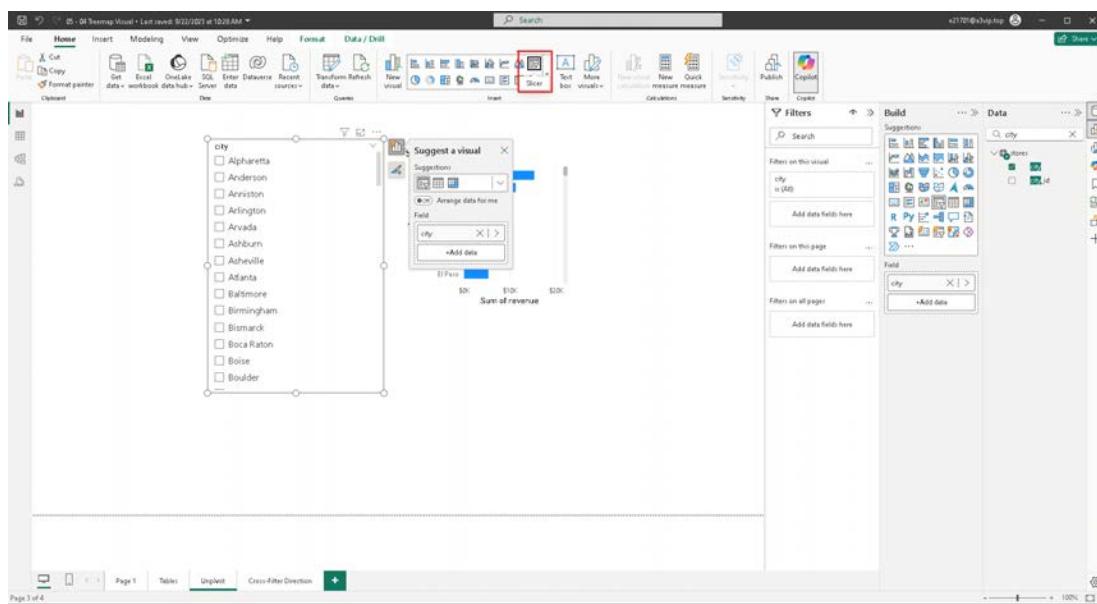
For example, let's consider a scenario where I add a slicer to filter by cities. After dragging and dropping the "City" field into the canvas, I convert it into a slicer visual. I then select a city, say Anderson, and remove any other visuals that could create conflicting filters. Now, I want this same slicer to apply on another page. Although I can manually add the same slicer to the second page by repeating the steps (dragging and dropping the "City" field), without syncing, the selections on one page won't affect the other.

Step 1: Add a Slicer to the First Page

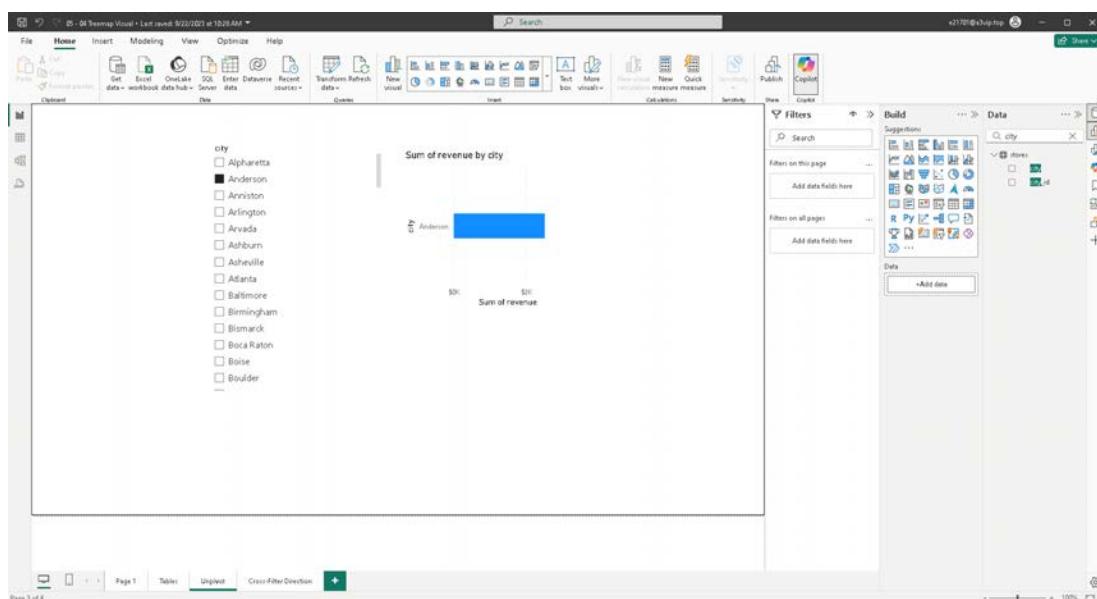
- 1. Drag and Drop a Field:** Drag and drop the field you want to filter by (e.g., "City") onto the canvas.



- 2. Convert to Slicer Visual:** Change the visual to a slicer by selecting the slicer option.



3. Make a Selection: Choose a value from the slicer, such as "Anderson."

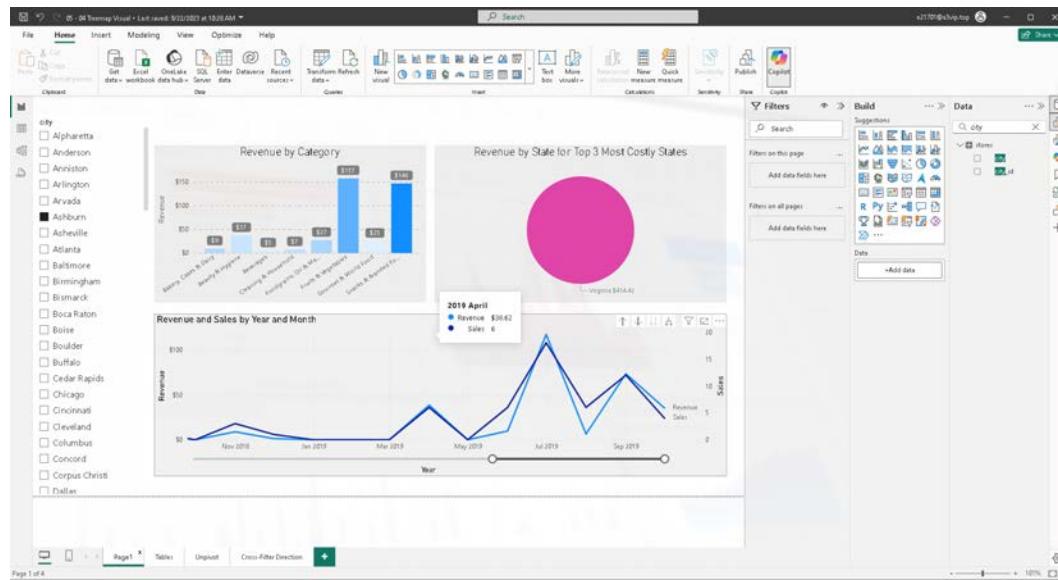


4. Remove Unnecessary Visuals: If needed, remove other visuals that may cause conflicting filters.

Step 2: Add the Same Slicer to Another Page

1. Go to Another Page: Navigate to the next page where you want the slicer to be applied.

- 2. Repeat Slicer Creation:** Drag the same field (e.g., "City") and convert it to a slicer visual, just like you did on the first page.
- 3. Test Selection:** Try selecting a value (e.g., "Ashburn") on this new page. Notice that it won't reflect the change on the first page yet because they aren't synced.



Step 3: Sync the Slicers (Two Methods)

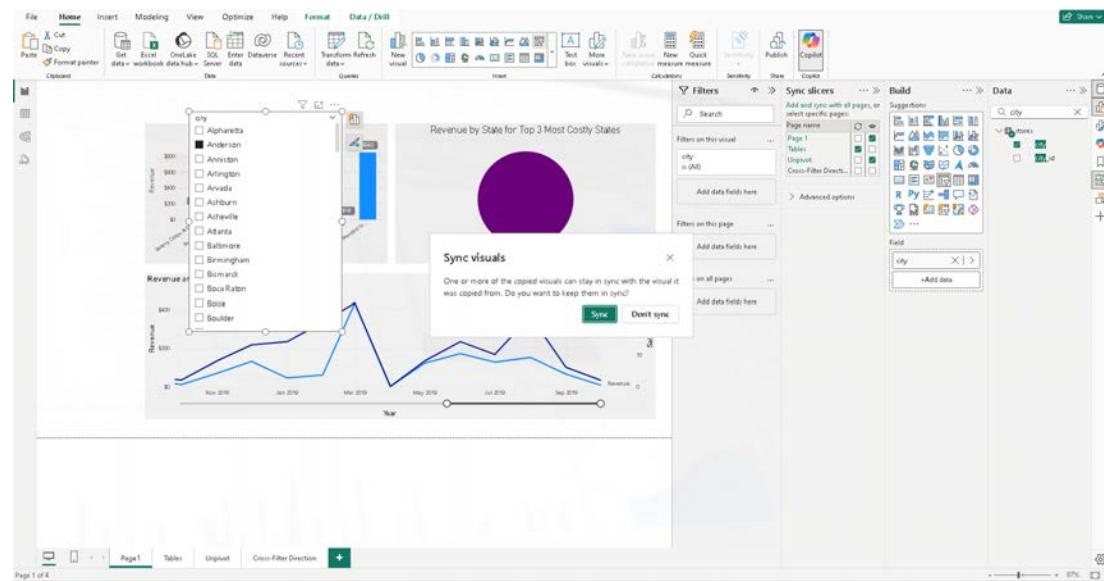
Method 1: Copy and Sync Automatically

- 1. Copy the Slicer:**
 - Go back to the first page.
 - Select the slicer visual, press **Ctrl + C** (or right-click and choose "Copy Visual").

- 2. Paste the Slicer:**

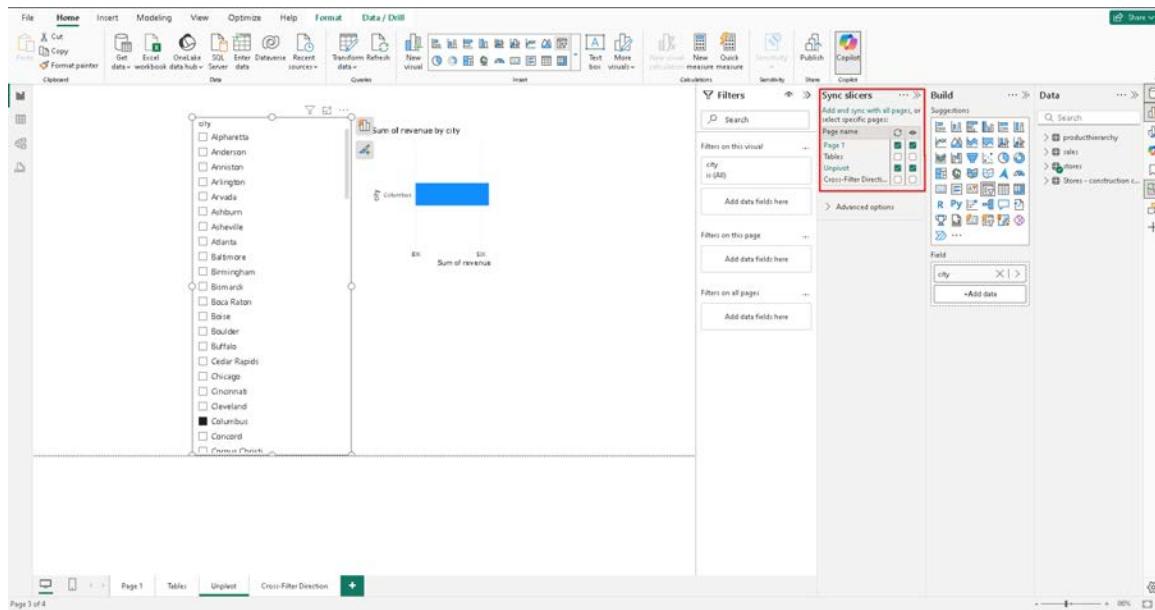
- Go to the second page.
- Press **Ctrl + V** to paste the slicer.

- 3. Sync Prompt:** Power BI will prompt you with the option to sync slicers across pages.



4. **Confirm Sync:** Choose "Yes" to sync. Now, any selection made in the slicer on one page will be reflected on the other.
5. Check: Let's choose Birmingham.

Method 2: Use the Sync Slicers Pane



1. Open the Sync Slicers Pane:

- On any page, go to the "View" tab in the ribbon and enable the Sync Slicers Pane.

2. Select the Slicer:

- Click on the slicer you want to sync on the first page.

3. Review Sync Settings:

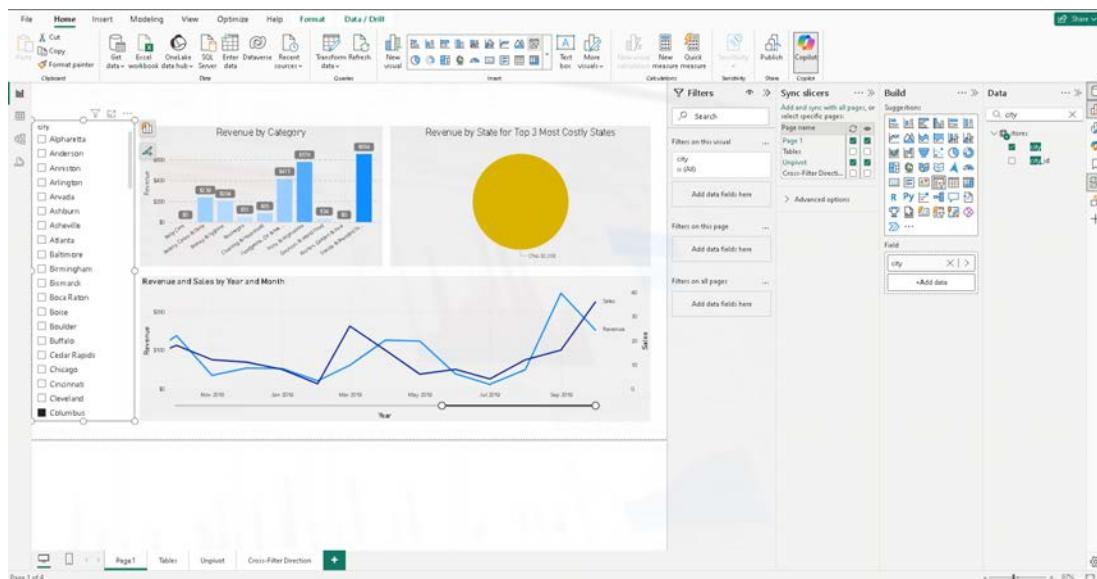
- In the Sync Slicers Pane, you'll see a table with all the report pages listed.
- Two columns appear:
 - Visibility (Eye Icon):** This controls whether the slicer is visible on a page.
 - Sync (Checkbox):** This ensures the slicers are synced across pages.

4. Sync the Pages:

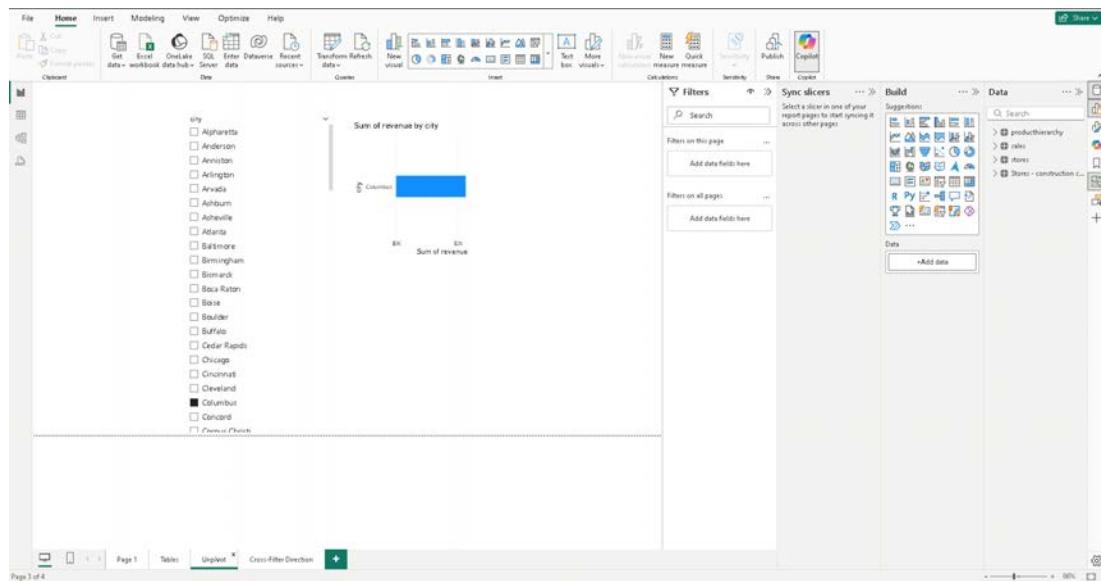
- Check the boxes in the sync column for the pages where you want the slicers to sync.
- You can also control whether the slicer should be visible or hidden on each page.

Step 4: Test the Synced Slicers

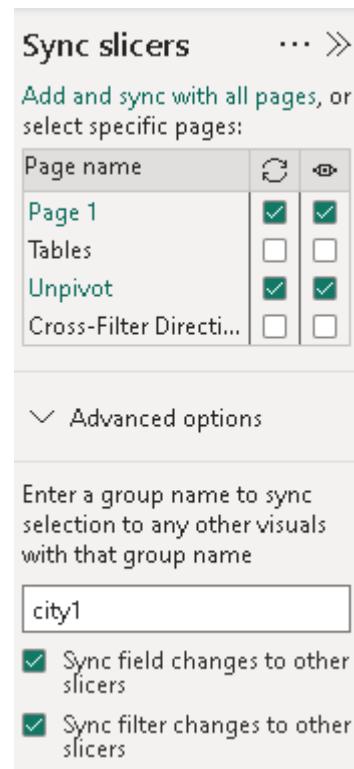
1. Go to the First Page: Select a value (e.g., "Columbus") from the slicer.



2. **Go to the Second Page:** Check that the same selection ("Columbus") is automatically applied here as well.



Advanced Options



- You can choose to keep slicers synced but not visible, allowing the slicer to filter the data without being shown on certain pages.
- Conversely, you can have visible slicers that aren't synced, depending on your needs.

Practical Tips

- **Visibility vs. Sync:** Remember, a slicer can be synced without being visible. This allows for a clean report design while still ensuring consistent filtering.
- **Multiple Slicers:** You can sync multiple slicers across different pages, and each can have different sync and visibility settings.

Syncing slicers in Power BI enhances the interactivity and consistency of your reports. Whether you use the simple copy-paste method or the more advanced Sync Slicers Pane, this feature helps you maintain consistent filtering across multiple report pages, leading to a more seamless and efficient data analysis experience.

▼ Understanding and Utilizing the Treemap Visual

The Treemap visual in Power BI is a powerful alternative to traditional charts like column charts, bar charts, or even pie charts. It is particularly useful for comparing values visually, as it provides an immediate sense of the size of each value within the context of the whole.

When to Use a Treemap Visual

Treemaps are ideal when you need to represent hierarchical data with multiple categories, especially when space is limited. They are also effective when you have many categories to display and want to avoid scrolling, as Treemaps allow you to present all categories at once in a compact and visually intuitive format.

Adding a Treemap Visual

Let's walk through the process of adding a Treemap to your report:

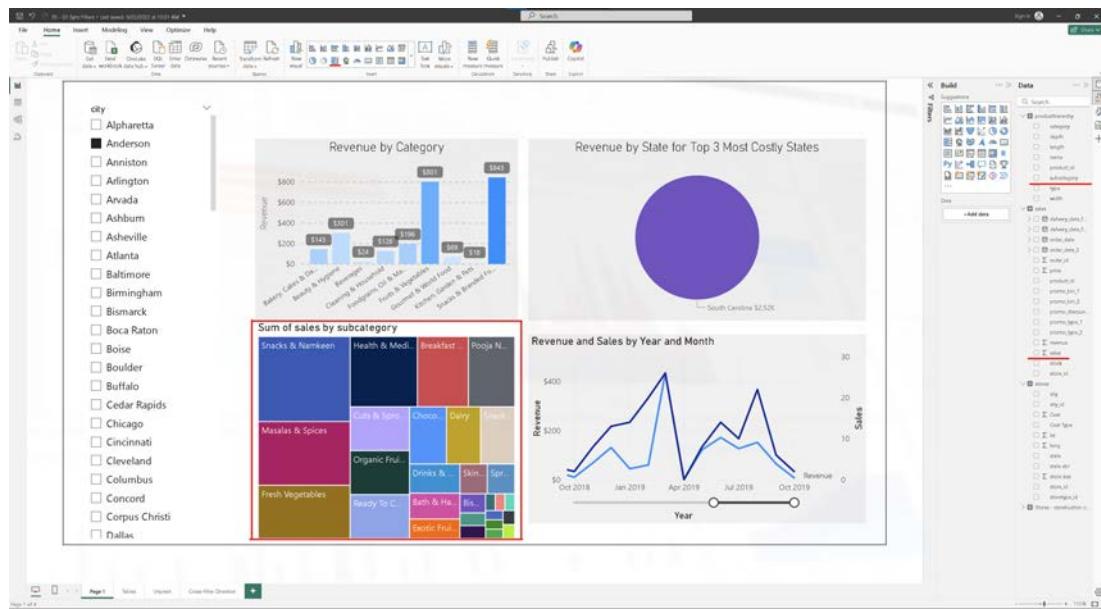
1. **Insert the Treemap Visual:** Start by selecting the Treemap visual from the Visualizations pane in Power BI.

2. **Add Data Fields:**

- **Category Field:** Drag the field representing your categories, such as "Subcategory," into the "Category" well.
- **Values Field:** Drag a measure, such as "Sales," into the "Values" well.



Once you've added these fields, the Treemap will display, with each rectangle representing a category. The size of each rectangle corresponds to the value of the category, allowing for easy comparison.

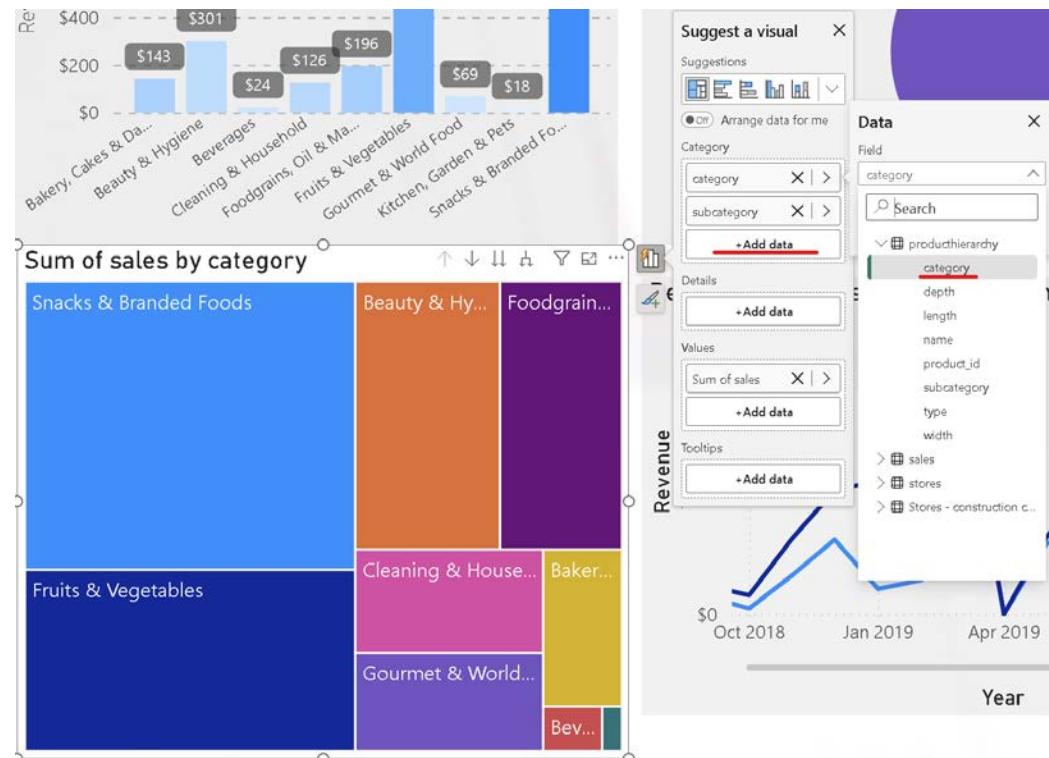


Hierarchies in Treemaps

Treemaps also support hierarchies, allowing you to display multiple levels of data:

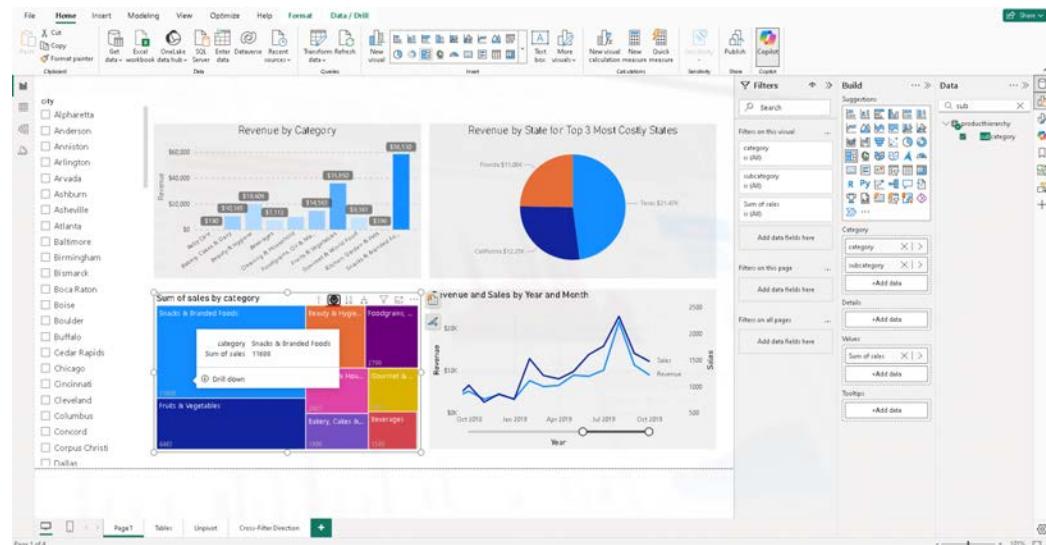
1. Adding a Hierarchy:

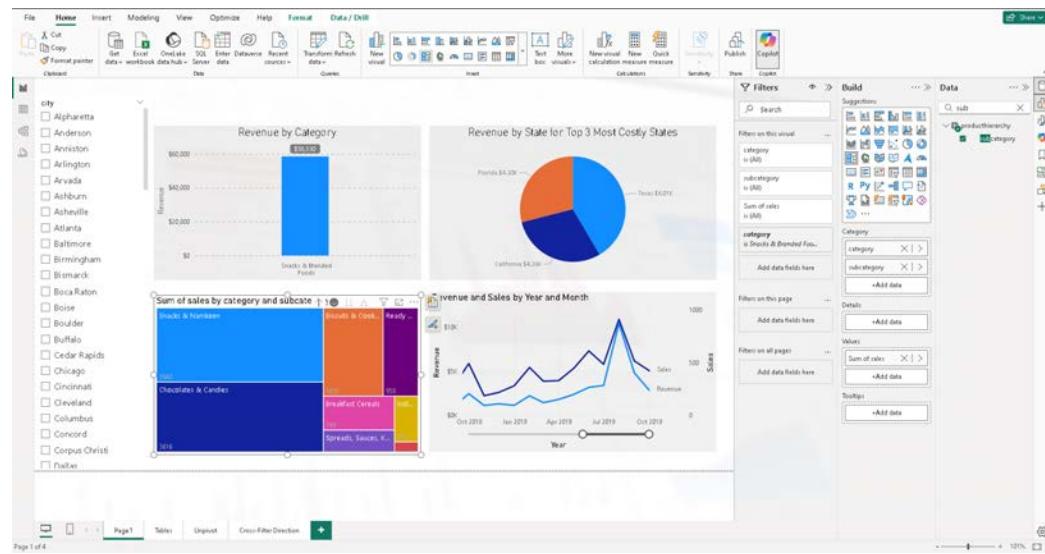
- You can create a hierarchy by adding another field, such as "Category," above or below your "Subcategory" field in the "Category" well.
- This enables you to drill down into different levels of your data, from broader categories to more specific subcategories.



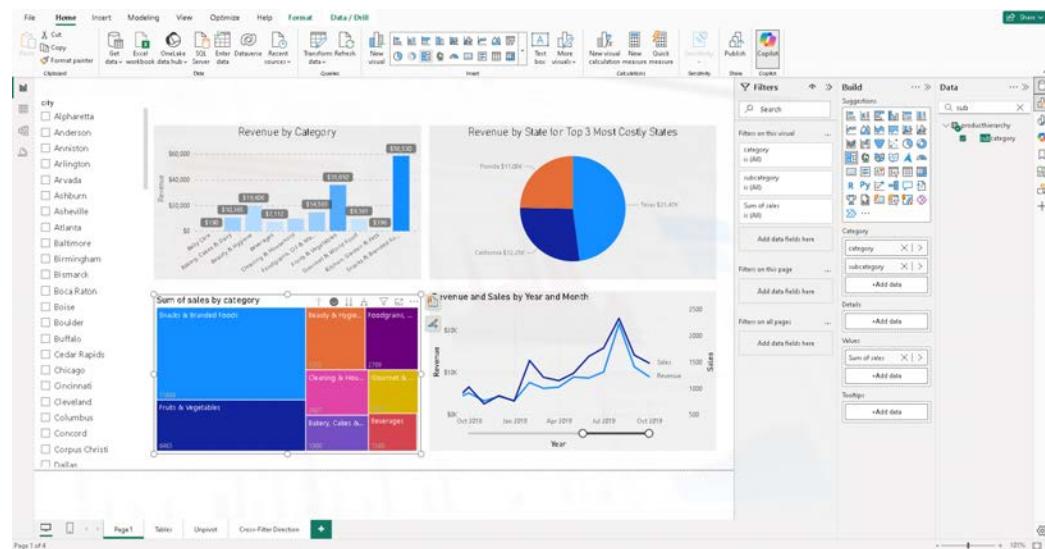
2. Using Drill-Down Features:

- **Drill Down:** Enable drill-down mode to explore different levels of the hierarchy within the Treemap.



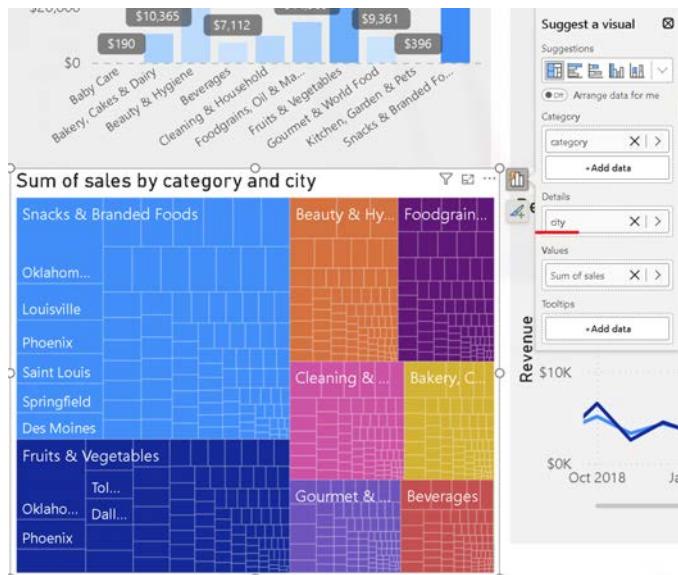


- **Drill Up:** You can also drill back up to view higher-level categories.



3. Adding Details:

- You can add more detailed information to the Treemap by dragging fields such as "State" or "City" into the "Details" well.
- This allows you to see how categories break down by additional dimensions, such as geographic location.



Formatting and Customization

Treemaps offer various formatting options to enhance their readability and alignment with your report's design:

- **Data Labels:** By default, values are only visible when you hover over a category. However, you can enable data labels to display values directly on the Treemap. You can also customize the units, font, color, and overall design of these labels.
- **Category Labels:** You can turn on or off the category labels to either simplify the visual or provide more detailed context.
- **Color Customization:** You can adjust the color scheme to match your report's theme or to highlight specific categories.

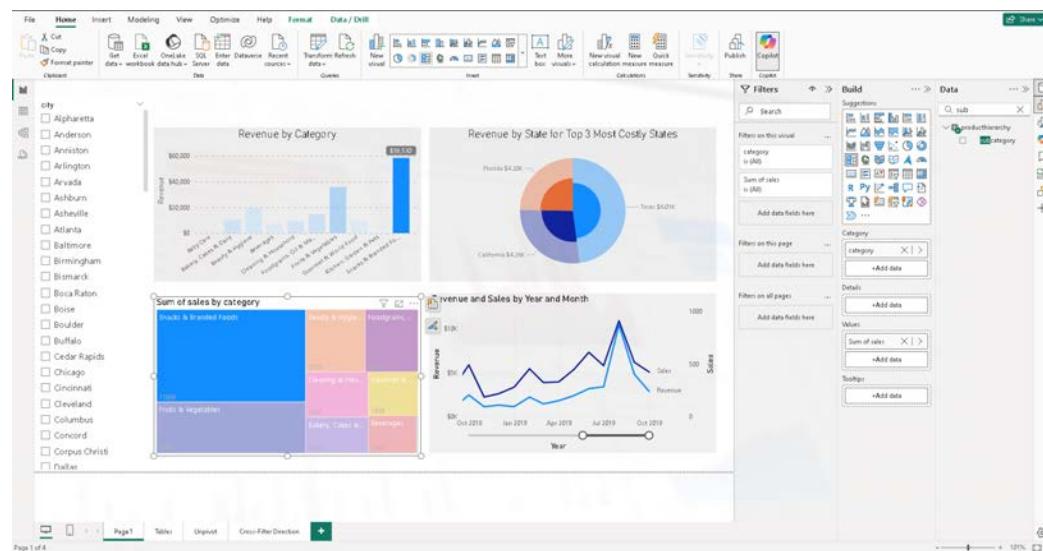
Advantages of Treemap Visuals

- **Space Efficiency:** Unlike other visuals that may require scrolling or take up significant space, Treemaps use space very efficiently, making them ideal for dashboards or reports with limited real estate.
- **Hierarchical Data Representation:** Treemaps can represent hierarchical data effectively by breaking down categories into subcategories, providing a visual overview of the data structure.

Interactivity and Cross-Filtering

One of the powerful features of Treemap visuals in Power BI is their ability to interact with other visuals:

- **Cross-Filtering:** Clicking on a category within the Treemap will automatically filter other visuals on the page based on that selection, enabling a dynamic and interactive reporting experience.
 - **Example:** If you select a specific subcategory in the Treemap, other charts or tables on the same page will update to reflect data related to that subcategory only.



The Treemap visual in Power BI is an excellent tool for visually representing and comparing hierarchical data. It is particularly useful in scenarios where space efficiency is critical, or when you need to display many categories at once. With its ability to support hierarchies, drill-down features, and cross-filtering, the Treemap visual adds significant value to any Power BI report, making it a versatile and powerful option for data visualization.

▼ Editing Interactions

One of the most powerful features of Power BI is the ability to create interactive reports where visuals can cross-filter and highlight data in other visuals. This interactivity allows users to explore data more dynamically and gain deeper insights. However, the default interaction settings might not always fit your specific needs. Fortunately, Power BI allows you to

customize how visuals interact with each other, offering greater control over the user experience.

Understanding Visual Interactions

When you select an item in one visual, such as a bar in a bar chart, Power BI can automatically filter or highlight data in other visuals on the same page.

There are three primary interaction modes:

1. **Highlight:** Selected data in one visual is highlighted in others, while non-selected data is grayed out but still visible.
2. **Filter:** Selected data in one visual filters out non-relevant data in others, showing only the selected data.
3. **None:** The selected data does not affect other visuals.

Editing Visual Interactions

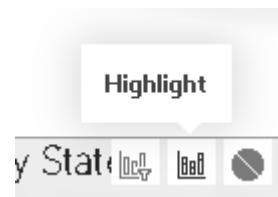
To edit interactions between visuals in Power BI, follow these steps:

1. **Select a Visual:** Start by clicking on the visual that you want to use as the starting point for filtering or highlighting. For example, select a column chart or a pie chart.
2. **Enable Edit Interactions:**
 - Go to the "Format" tab in the ribbon.
 - Click on the "Edit Interactions" button. This will activate the interaction editing mode and display interaction controls on all other visuals on the page.



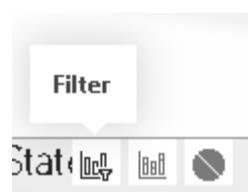
3. Customize Interactions:

- **Highlight:** This option is represented by a small circle icon. When selected, the visual will highlight relevant data in the other visuals, keeping all data visible but dimming non-selected items.





- **Filter**: This is represented by a funnel icon. Selecting this option will filter the other visuals to show only the selected data.



- **None:** This is represented by a null or crossed-out symbol. Selecting this option will disable interaction, so the selected visual won't affect the others.



4. Apply Changes:

- If you want the selected visual to have a different interaction effect on another visual, simply click on the corresponding interaction control for that visual.
- For example, if you want a pie chart to fully filter a bar chart rather than just highlighting it, switch from the highlight mode to the filter mode.

5. Adjusting Slicer Interactions:

- You can also customize how slicers interact with other visuals. For instance, you can set a slicer to filter only specific visuals while

leaving others unaffected. This level of control allows for more tailored data analysis experiences.

Finalizing and Formatting

After configuring the interactions:

- **Format the Visuals:** Ensure consistency in the appearance of your visuals by using the Format Painter tool. This can help maintain a professional look across your report.
- **Update Titles and Labels:** Adjust the titles and labels to reflect the interaction changes. This helps users understand what data they are viewing and how the visuals are connected.

Customizing visual interactions in Power BI enhances the interactivity and usability of your reports. By using the "Edit Interactions" feature, you can control how data flows between visuals, providing a more tailored experience that aligns with the specific needs of your analysis. This flexibility allows you to create reports that are not only visually appealing but also highly functional and user-friendly.

▼ Drillthrough

The Drillthrough feature in Power BI is a powerful tool that allows users to navigate from a summary-level report to a more detailed view. This is particularly useful when you want to provide an in-depth analysis of a specific category, product, or data point, enabling users to "drill through" to another page dedicated to that detailed view.

Setting Up a Drillthrough Page

To create a drillthrough experience, you need to set up a dedicated page in your report, often referred to as the "details page." Here's how to do it:

1. Create the Drillthrough Page:

- Start by creating a new page in your report where you want the detailed data to be displayed.

- This page will serve as the landing page when a user drills through from a summary page.

2. Enable Drillthrough:

- On the drillthrough page, click on an empty space on the canvas to ensure no visuals are selected.
- Go to the "Format" tab and expand the "Page Information" section.
- Set the **Page Type** to "Drillthrough."

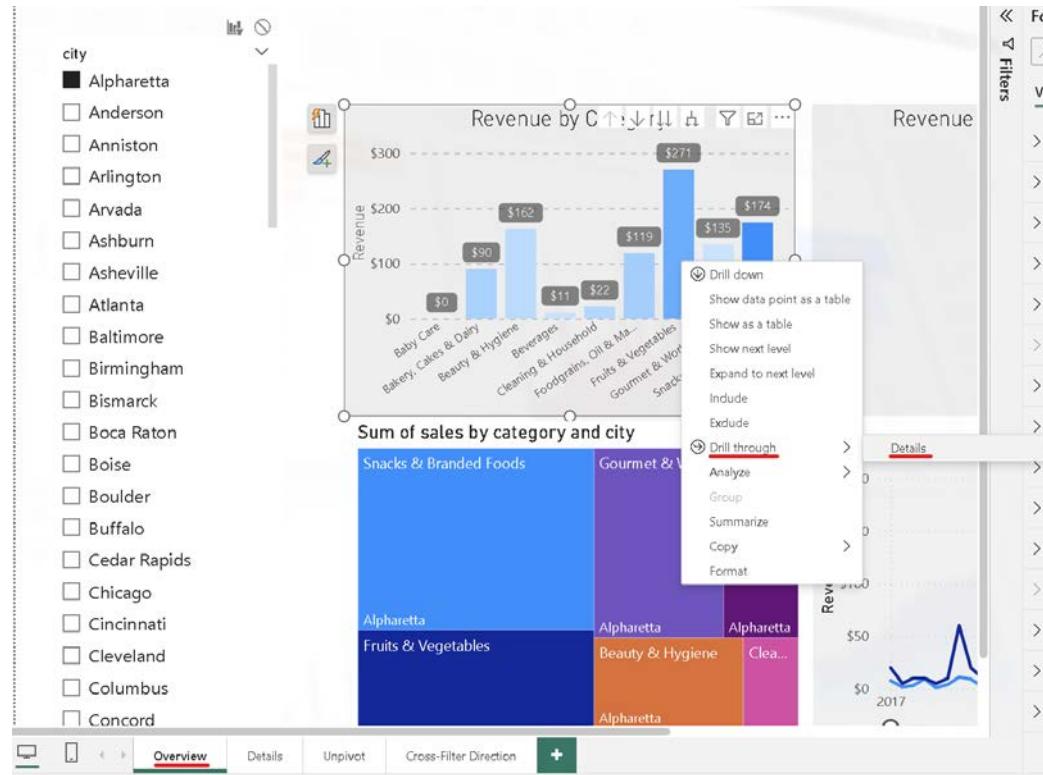
3. Add Fields for Drillthrough:

- Drag the field you want to use for the drillthrough, such as "Category" or "Subcategory," into the "Add drill-through fields here" section on the right side of the canvas.
- This field will be used to filter the data on the drillthrough page based on the user's selection on the summary page.

The screenshot shows the Power BI desktop interface with the 'Format' tab selected in the ribbon. In the 'Page information' section of the 'Format' pane, the 'Page type' dropdown is set to 'Drillthrough'. The 'Data' pane on the right displays a hierarchical structure under the 'category' node, including depth, length, name, product_id, subcategory, type, and width. A small data grid is visible in the center-left of the screen, showing a table with columns Order Date, Revenue, and Orders, containing data from August 2019 to March 2018.

4. Demonstrate Drillthrough:

- Navigate back to your summary page where you have a visual, such as a bar chart, displaying the categories or other fields.
- Right-click on a category (e.g., "Fruits and Vegetables") and select the drillthrough option. You should see the drillthrough page name in the menu.



- Clicking on it will take you to the drillthrough page, where the data is automatically filtered based on the category you selected.

The screenshot shows a drillthrough page for the 'Fruits & Vegetables' category. On the left is a table with columns 'Order Date', 'Revenue', and 'Orders'. The data is as follows:

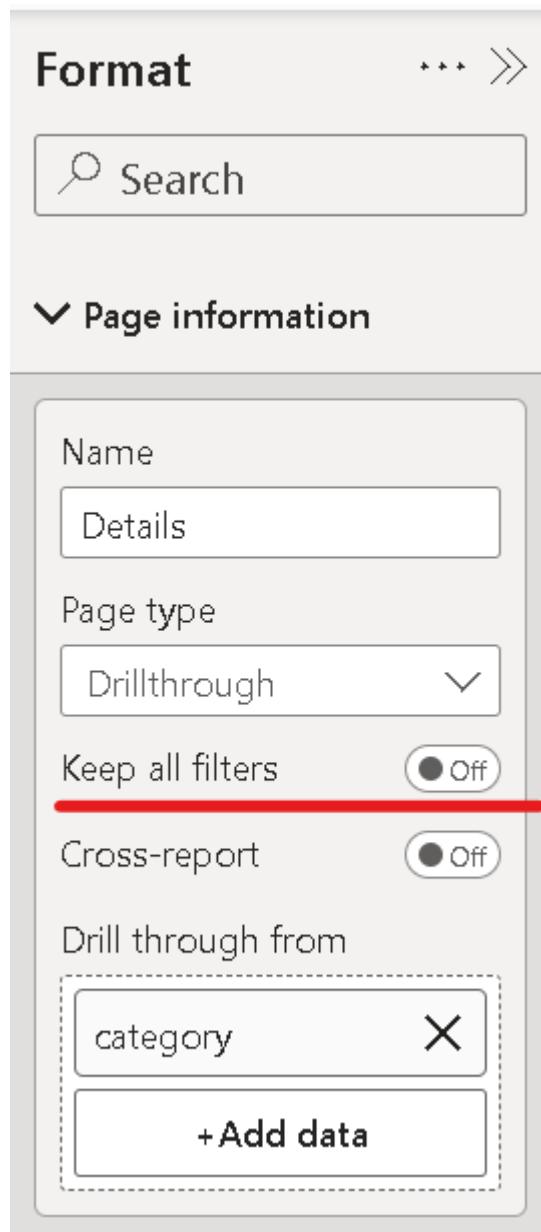
Order Date	Revenue	Orders
8/3/2019	\$121.20	2
10/1/2019	\$43.98	4
3/7/2018	\$25.38	2
1/28/2019	\$15.06	2
9/8/2019	\$12.96	2
9/12/2019	\$11.02	2
6/16/2019	\$9.72	2
11/13/2017	\$6.71	1
1/17/2018	\$4.63	1
9/15/2019	\$3.70	2
Total	\$270.84	236

On the right, there are two filter panes: 'category' (set to 'Fruits & Vegetables') and 'city' (set to 'Alpharetta'). A sidebar at the bottom right says 'Add data fields here'.

Configuring Additional Options

- **Keep All Filters:**

- By default, any filters applied on the summary page (such as a slicer for "City") are carried over to the drillthrough page. If you do not want these filters to be transferred, you can disable this option by turning off "Keep all filters" in the drillthrough setup.
- For example, if you drill through on "Snacks and Branded Foods" while "City: Alpharetta" is selected, both filters will apply unless you turn off "Keep all filters." If turned off, only the category filter will be applied on the drillthrough page.



- **Using Buttons for Navigation:**

- When a page is set as a drillthrough page, Power BI automatically adds a back button to the page. This button allows users to easily return to the previous page they came from. You can customize this button or remove it if not needed.
- When published to the Power BI service, users do not need to hold the "Ctrl" key to click this button they can simply click it to navigate back.

- **Hiding the Drillthrough Page:**

- Often, you might want to hide the drillthrough page from the report's navigation to prevent users from accessing it directly. You can do this by right-clicking the page tab and selecting "Hide Page." The page will still function as a drillthrough target but will not be visible in the report's tab menu.

Drillthrough is a vital feature in Power BI that enhances the interactivity and depth of your reports. By setting up a drillthrough page and configuring it properly, you allow users to navigate from a broad overview to specific, detailed insights with ease. Whether using categorical data, numerical summaries, or a combination of both, drillthrough pages make your reports more dynamic and user-friendly.

▼ Project 6

[Projects6.rar](#)

▼ 5- Advance Transformations & Visualizations

▼ Creating Custom Columns

Custom columns in Power BI are a powerful feature that allows you to create new data fields based on existing ones using formulas and transformations. This feature is particularly useful when you need to derive new metrics or perform calculations that aren't directly available in your dataset. In this lecture, we'll explore how to create custom columns using the Power Query Editor and the M language, which underpins the transformations in Power BI.

What is a Custom Column?

A custom column is a new column that you create in the Power Query Editor by writing a formula that performs calculations or transformations on

existing columns. These calculations can range from simple arithmetic operations to more complex functions.

Setting Up a Custom Column

Let's walk through the process of creating a custom column, using an example where we want to calculate the volume of a product based on its dimensions: width, length, and depth.

The screenshot shows the Power BI desktop application. The ribbon at the top includes File, Home, Insert, Modeling, View, Optimize, Help, and various data import options. The main workspace displays two tables. The first table has columns Order Date, Revenue, and Orders, with data from January 1, 2017, to January 31, 2017, totaling \$264.69. The second table lists various products with their dimensions: width, depth, and length. A tooltip for the word 'volume' is displayed over the 'Build' pane, which contains a 'Data' section with a search bar and a list of tables and measures. The 'Filters' pane on the left shows filters applied to the 'Revenue' and 'Subcategory' columns. The bottom of the screen shows navigation tabs for Overview, Data, Unpivot, and Cross-Filter Direction, along with a page number indicator.

1. Open Power Query Editor:

- In your Power BI report, go to the "Home" tab and click on "Transform data" to open the Power Query Editor.

2. Navigate to the Desired Table:

- In the Power Query Editor, select the table where you want to add the custom column. For our example, we'll use the "producthierarchy" table.

	product_id	name	type	length	depth	width
1	P0000	Serum (Lyon)	Indian & Exotic Herbs	5	20	12
2	P0001	Hand Wash + Moisture Shield (Savlon)	Hair Oil & Serum	18.5	22	20
3	P0002	Good Day Butter Cookies (Britannia)	Hand Wash & Sanitizers	22	40	22
4	P0004	Happy Happy Choco-Chip Cookies (Parle)	Cookies	2	35	4
5	P0005	50-50 Timetess Salted Biscuits (Britannia)	Glucose & Milk Biscuits	16	30	16
6	P0006	Tiger Elachi Cream Biscuits (Britannia)	Salted Biscuits	8.5	15	15
7	P0007	Bounce Biscuits - Choco Creme (Sunfeast)	Glucose & Milk Biscuits	2	22	9.5
8	P0008	50-50 Timetess Biscuits (Britannia)	Cream Biscuits & Wafers	5	16	5
9	P0009	Tiger Chocolate Cream Biscuits (Britannia)	Salted Biscuits	5	18	14
10	P0010	Biscuits - Magix Kreams Choc (Parle)	Cream Biscuits & Wafers	2	22	3
11	P0011	Dreams Cup Cake - Choco (Elite)	Cream Biscuits & Wafers	8	22	15

3. Add a Custom Column:

- Go to the "Add Column" tab in the ribbon and select "Custom Column." This will open the "Add Custom Column" dialog box where you can define your new column.

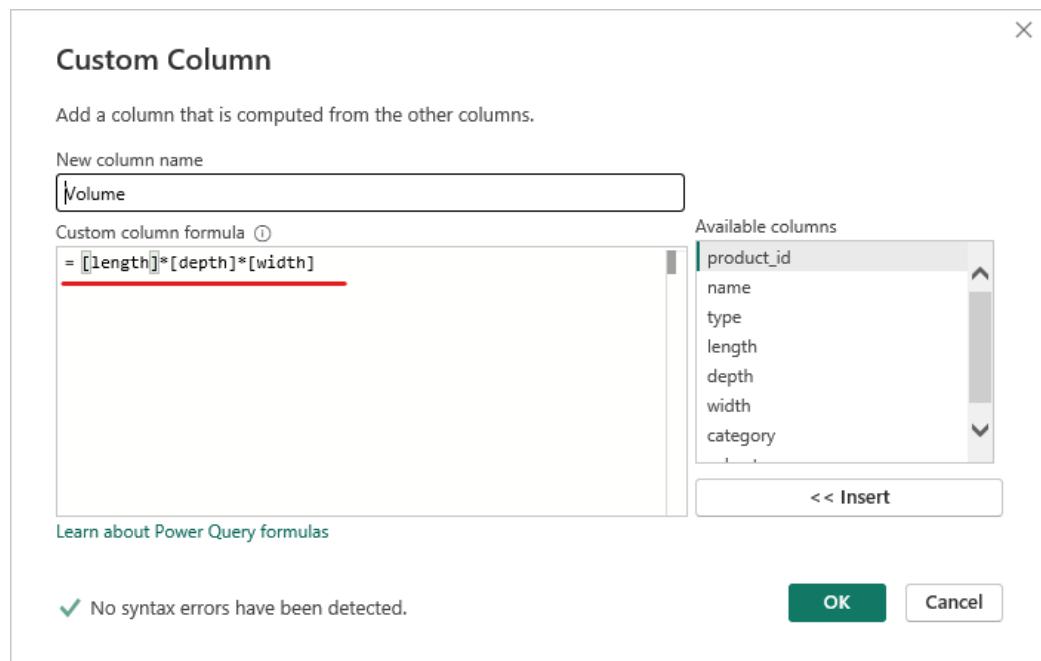
The screenshot shows the Power BI Data Editor interface. The top navigation bar includes 'File', 'Home', 'Transform', 'Add Column', 'View', 'Tools', and 'Help'. Below the 'Transform' tab, there are several icons: 'Column From Examples', 'Custom Column' (which is highlighted with a red box), 'Invoke Custom Function', 'Conditional Column', 'Index Column', 'Duplicate Column', 'Format' (with dropdown options 'ABC', '123 Extract', 'abc Parse'), 'Merge Columns', 'Parse', and 'Statistics'. The main area displays a list of 'Queries [7]' under 'Helper Tables [3]' and 'Other Queries [4]'. A preview pane on the right shows two columns: 'product_id' and 'name'. The 'product_id' column has a summary table with three rows: 'Valid' (100%), 'Error' (0%), and 'Empty' (0%). The 'name' column also has a summary table with three rows: 'Valid' (100%), 'Error' (0%), and 'Empty' (0%). Below these summaries are three rows of data: 1. P0000 Serum (Livon); 2. P0001 Hand Wash - Moisture Shield (Sa...); 3. P0002 Good Day Butter Cookies (Britan).

4. Name the Column:

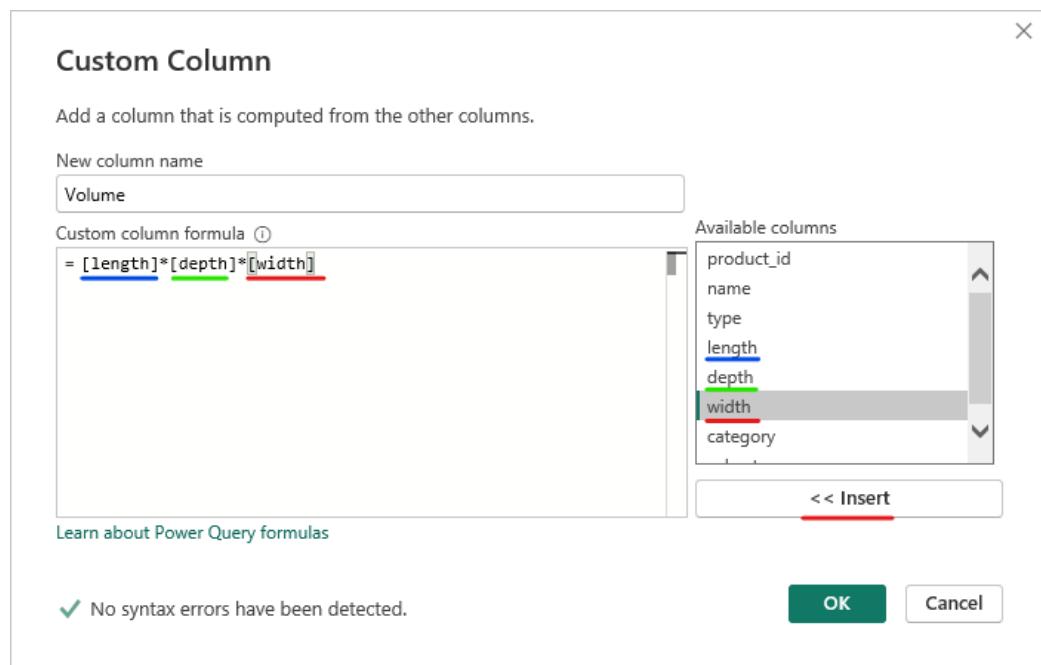
- Give your new column a meaningful name, such as "Volume."

5. Write the Formula:

- In the "Custom Column Formula" box, you can write your formula using the M language. For example, to calculate the volume of a product (width × length × depth), you would write:



- You can select the columns directly from the "Available columns" list to ensure accuracy.

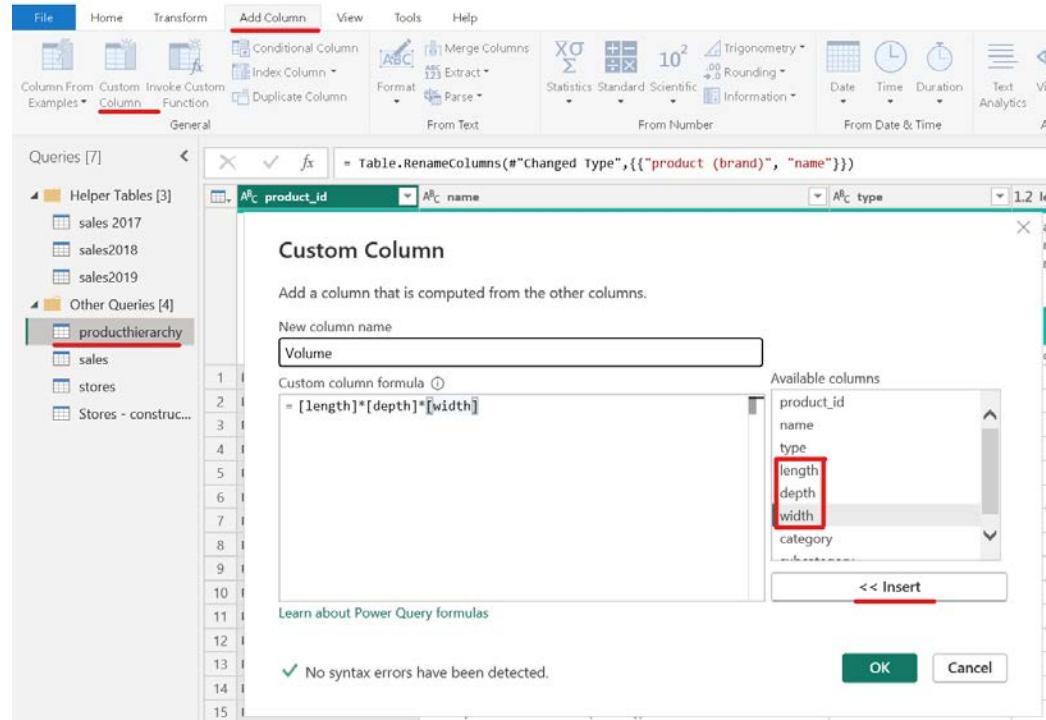


6. Syntax Check:

- Power BI automatically checks the syntax of your formula. If there are any errors, they will be highlighted so you can correct them before proceeding.

7. Click OK:

- Once you've written the formula and named the column, click "OK." Your new custom column will be added to the table.



name	width	depth	length	Volume
0	15.30	4.80	3.80	279.07
50-50 Timepass Salted Biscuits (Britannia)	16.00	30.00	16.00	7,680.00
Tiger Elaichi Cream Biscuits (Britannia)	15.00	15.00	8.50	1,912.50
1-2-3 Noodles - Chicken Flavour (Wai Wai)	7.00	24.00	1.60	268.80
1-2-3 Noodles - Chicken Flavour (Wai Wai)	9.00	13.50	4.00	486.00
1-2-3 Noodles - Veg Masala Flavour (Wai Wai)	4.90	14.50	3.00	213.15
1-2-3 Noodles - Veg Masala Flavour (Wai Wai)	10.00	17.50	4.00	700.00
3 Layer Gum - Sugarfreee Peppermint Flavour (Center Fresh)	7.00	18.00	3.00	378.00
3 Layer Gum - Sugarfreee Strawberry Flavour (Center Fresh)	1.00	1.00	1.00	1.00
50-50 Jeera Masti Biscuits (Britannia)	4.50	17.50	1.70	133.88
50-50 Maska Chaska Salted Biscuits (Britannia)				

Modifying Custom Columns

If you need to change the formula or adjust the custom column later, you can easily do so:

- **Click on the Gear Icon:** In the "Applied Steps" pane in the Power Query Editor, find the step where you added the custom column. Click the gear icon next to it to reopen the "Add Custom Column" dialog.
- **Edit the Formula:** Make the necessary changes to your formula, and click "OK" to apply the changes.

Custom columns are a versatile and powerful tool in Power BI, enabling you to perform custom calculations and create new data fields based on your existing data. Whether you're calculating product volumes, creating new metrics, or performing other transformations, custom columns can help you tailor your data model to meet your specific needs.

▼ Enable & Disable Load

We'll discuss the concept of enabling and disabling data loads in Power BI, a practice that can significantly improve performance by controlling which data is loaded into the data model. Understanding when and how to disable certain loads can optimize both the efficiency and usability of your reports.

What Does It Mean to Enable or Disable a Load?

Enabling or disabling a load refers to the decision of whether a particular table should be loaded into the Power BI data model. Loading a table into the model allows its data to be used in visuals, calculations, and other report elements. However, every table that is loaded consumes memory and processing power, which can slow down your report's performance, especially if the data isn't needed for final analysis or reporting.

Why Disable a Load?

In scenarios where performance is an issue, reducing the amount of data loaded into the model can lead to faster report refreshes and better overall responsiveness. This is particularly useful if your report contains large or numerous tables that are only used for intermediate transformations or calculations but are not needed for final reporting.

Steps to Disable a Load

Let's go through the steps to disable a load in Power BI:

1. Open the Query Editor:

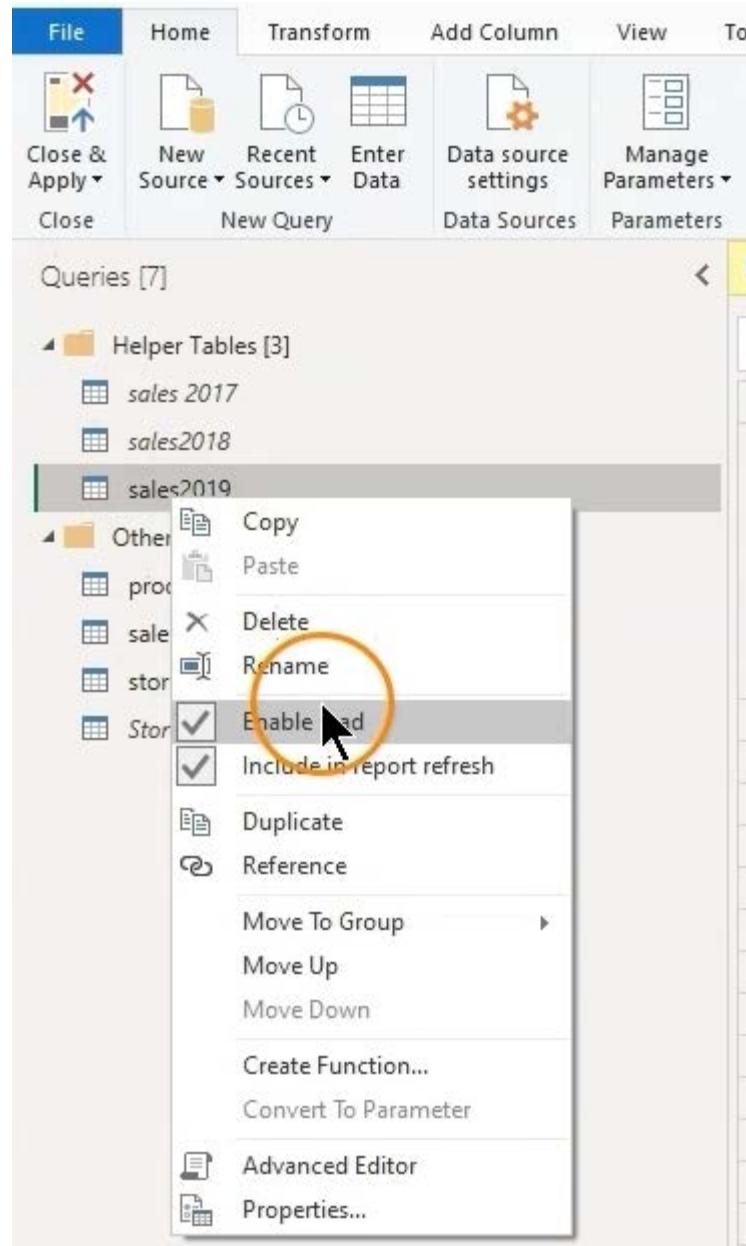
- From your Power BI report, go to the "Home" tab and select "Transform data" to open the Power Query Editor.

2. Identify Tables to Disable:

- Review the tables in your model. Identify any tables that are used only for data transformations and are not required in the final report (e.g., helper tables).

3. Disable the Load:

- Right-click on the table you wish to exclude from the data model.
- Uncheck the "Enable load" option. This action tells Power BI not to load the table into the model but still allows it to be used in intermediate steps of your data transformation process.



4. Warning Message:

- Upon disabling the load, Power BI will display a warning indicating that the table will be removed from the report and any visuals using its data will be broken. It's important to ensure that the table isn't needed for any visuals before disabling the load.



Possible Data Loss Warning

Disabling load will remove the table from the report, and any visuals that use its columns will be broken.

Continue

Cancel

5. Apply Changes:

- After disabling the load, click "Close & Apply" to save your changes. The table will no longer be part of the data model, reducing the memory usage and potentially improving performance.

Example: Disabling a Helper Table

Imagine you have a "Store Construction Costs" table that you've used to merge data with another "Stores" table. Once the merge is complete, the "Store Construction Costs" table is no longer needed for any visuals. You can disable the load for this table:

- Right-click on "Store Construction Costs" in the Query Editor.
- Uncheck "Enable load."

The screenshot shows the Power BI Data Editor interface. The ribbon at the top includes File, Home, Transform, Add Column, View, Tools, and Help. The Home tab is selected. Below the ribbon are various icons for file operations like Close & Apply, New Source, Recent Sources, Enter Data, Data source settings, Manage Parameters, Refresh Preview, Properties, Advanced Editor, Choose Columns, and Manage. The main area displays a list of queries under 'Queries [7]'. The 'Stores - construction' query is selected and has a context menu open. The menu options are: Copy, Paste, Delete, Rename, Enable load (which is checked), Include in report refresh (which is checked), Duplicate, Reference, Move To Group, Move Up, Move Down, Create Function..., Convert To Parameter, Advanced Editor, and Properties... . To the right of the menu, there is a preview pane showing a table with columns 'store_id' and '1.2 bu'. The preview indicates 144 distinct and unique values. A status bar at the bottom of the preview pane says 'This preview may be up to 4 days old'.

- Apply your changes.

Now, the "Construction Costs" table is still used in the transformation steps, but it won't consume memory in the final data model, helping to optimize performance.

Recognizing Disabled Tables

After disabling the load for a table, it will appear in italic font in the Query Editor. This indicates that the table is still part of the transformation process but is not loaded into the final data model.

Performance Benefits

Disabling unnecessary tables from being loaded into the data model can have several benefits:

- **Reduced Memory Usage:** Tables that are not loaded into the data model do not consume memory, which can reduce the overall size of your Power BI file and improve performance, especially on larger datasets.
- **Improved Report Speed:** With fewer tables loaded, Power BI can refresh and interact with the data more quickly, leading to a smoother user experience.
- **Cleaner Data Model:** By only loading necessary tables, the data model becomes easier to manage and understand, reducing potential confusion for users.

Additional Option: Include in Report Refresh

Along with the "Enable load" option, there's also an "Include in report refresh" option:

The screenshot shows the Power BI desktop interface. In the center, there's a data grid titled "Stores - constructed" with columns "store_id" and "bu". The "store_id" column has 144 distinct values. On the left, the "Queries [7]" pane is visible, listing "Helper Tables [3]" (sales 2017, sales2018, sales2019) and "Other Queries [4]" (producthierarchy, sales, stores). A context menu is open over the "Stores - constructed" table, with the "Include in report refresh" option selected (indicated by a red arrow).

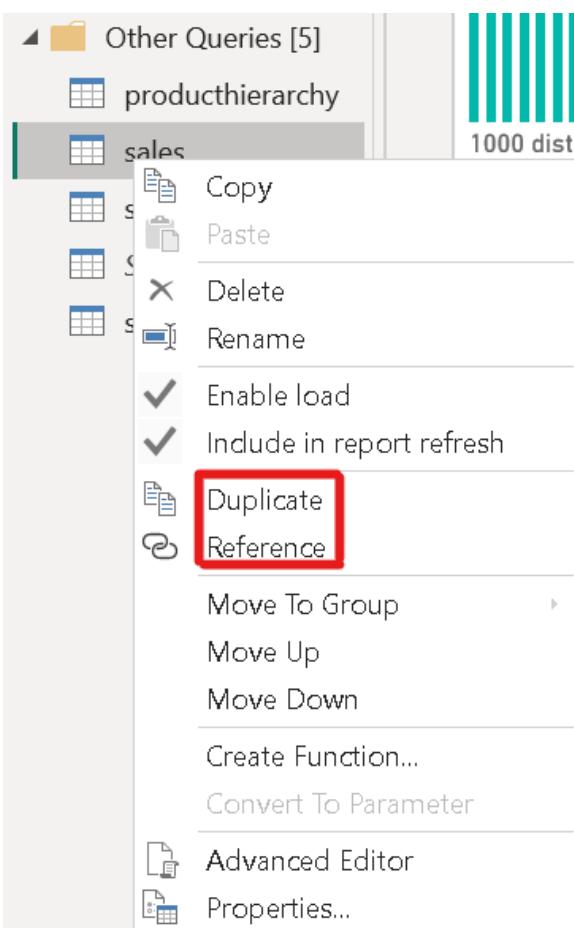
- **Include in Report Refresh:** If checked, the table will be included in the data refresh process every time you refresh the report. If unchecked, the table will not be refreshed, meaning it will retain the data from the last time it was loaded.
- This can be useful if a table does not require frequent updates or if you want to reduce the time taken during the refresh process.

Enabling and disabling loads is a crucial technique in Power BI for optimizing report performance. By carefully managing which tables are loaded into your data model, you can significantly improve the efficiency of your reports. This practice not only enhances performance but also

streamlines the user experience by ensuring that only necessary data is included in the final report.

▼ References & Duplicates

In Power BI, creating copies of queries is a common task, whether you need to modify data for a specific scenario or to optimize your data model. There are two main ways to duplicate a query: creating a **Duplicate** or creating a **Reference**. Each method has its specific use cases, advantages, and drawbacks. In this lecture, we'll explore both methods, understand their differences, and learn when to use each one.



Duplicating a Query

A **Duplicate** in Power BI is essentially a complete copy of an existing query, including all the transformation steps applied to it.

- **How to Create a Duplicate:**

1. In the Power Query Editor, right-click on the query you want to duplicate.
2. Select "Duplicate."

The screenshot shows the Power Query Editor interface. On the left, the 'Queries [9]' pane lists several queries under 'Helper Tables' and 'Other Queries'. The 'sales' query is currently selected and highlighted with a red box. A context menu is open over this query, with the 'Duplicate' option highlighted. The main workspace displays the schema and preview of the 'sales' query, which contains columns 'order_id' and 'product_id'.

order_id	product_id
18117	P0129
18118	P0513
18119	P0229
18120	P0275
18122	P0448
18123	P0602
18124	P0510
18125	P0275
18126	P0655
18127	P0283
18129	P0579
18130	P0382
18131	P0608
18132	P0296
18133	P0234
18134	P0226
18135	P0389
18136	P0370

3. A new query will be created, typically named with the original query name followed by " (2)" (e.g., "Sales (2)").

- **Characteristics of a Duplicate:**
 - The duplicated query is entirely independent of the original query. This means that any changes you make to the duplicate do not affect the original query and vice versa.
 - All the transformation steps from the original query are copied over, which can be beneficial if you want to apply slight modifications to an existing query while keeping the original intact.
- **Drawbacks of Using a Duplicate:**
 - **Performance Impact:** Since all transformations are duplicated, Power BI must reprocess each step, leading to potential performance issues, especially with large datasets.
 - **Maintenance Complexity:** If the original query changes, the duplicated query will not automatically reflect those changes, requiring manual updates to maintain consistency.

Creating a Reference

A **Reference** is another way to create a copy of a query, but with significant differences from a duplicate.

- **How to Create a Reference:**
 1. In the Power Query Editor, right-click on the query you want to reference.
 2. Select "Reference."

The screenshot shows the Power BI Data Editor interface. On the left, the 'Queries [9]' pane is open, displaying a tree structure of queries. Under the 'sales' node, there is a query named 'sales (Reference)' which is highlighted with a red underline. A context menu is open over this query, listing various actions such as Copy, Paste, Delete, Rename, and several options related to loading and reference management. In the background, a preview window shows a table with columns 'order_id', 'Val', 'Err', and 'Em'. The preview indicates there are 1000 distinct values and 1000 unique values.

3. A new query is created, which references the original query's result.

- **Characteristics of a Reference:**

- The reference query does not duplicate the transformation steps but instead points to the final result of the original query. This makes it much more efficient because it doesn't require reprocessing the steps already performed by the original query.
- Any changes made to the original query are automatically reflected in the reference. This ensures consistency and reduces maintenance efforts.

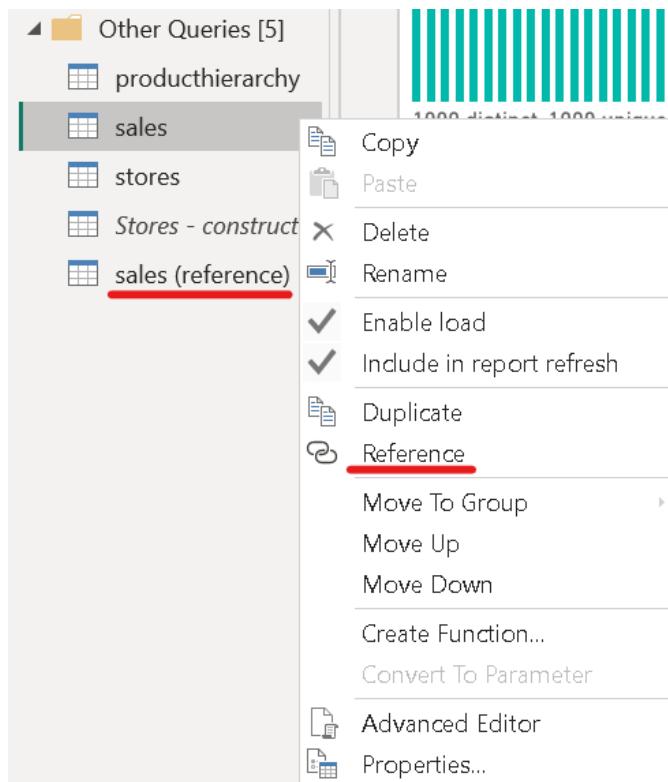
- **Advantages of Using a Reference:**

- **Performance Efficiency:** By reusing the results of the original query, references are more efficient and less resource-intensive.
- **Easier Maintenance:** Since the reference query dynamically reflects any changes made to the original query, it simplifies maintenance and ensures consistency across your data model.

Example Scenario: Creating a Date Dimension

Suppose you have a "Sales" table and you need to create a date dimension for your report. Instead of duplicating the "Sales" table, which would involve unnecessary duplication of data and transformations, you can create a reference.

- **Step 1:** Create a reference of the "Sales" table.



- **Step 2:** In the reference query, remove all columns except the "Order Date" column.

The screenshot shows the Power BI Data Editor interface with the 'sales' table selected. The left sidebar lists various queries and helper tables. The main area displays the 'sales' table with columns: 'order_id', 'product_id', 'store_id', and 'order_date'. The 'order_date' column has a context menu open, with the 'Remove Duplicates' option highlighted.

- **Step 3:** Remove duplicate dates to ensure each date appears only once.

This screenshot shows the 'order_date' column from the previous table with its context menu open again. The 'Remove Duplicates' option is clearly highlighted.

- **Step 4:** Add date-related features, such as the day of the week, month, or quarter.

This approach reduces the data loaded into your model, improving performance and making your data model more efficient.

Key Differences Between Duplicate and Reference

- **Independence:** A duplicate is independent of the original query, while a reference is dependent on the original query.
- **Performance:** A reference is generally more efficient since it leverages the processed data of the original query, whereas a duplicate requires reprocessing.
- **Use Case:** Use a duplicate when you need to create a variation of the original query with different transformations. Use a reference when you want to reuse the results of the original query with minimal additional processing.

In most cases, using a **Reference** is the better option due to its efficiency and ease of maintenance. However, there are scenarios where a **Duplicate** is appropriate, particularly when you need a completely independent query to apply different transformations. Understanding these differences will help you make informed decisions when building and optimizing your Power BI data model.

▼ Column From Example

In Power BI, the **Columns From Example** feature allows you to create new columns based on sample data that you provide. This feature is particularly useful when you want to add new data points or perform transformations without writing complex M language formulas. In this lecture, we'll explore how to efficiently create additional columns using examples, focusing on date-related features like weekdays.

Why Use Columns From Example?

The **Columns From Example** feature is beneficial because:

- **Ease of Use:** You don't need to know or write M language code; instead, you can input sample values, and Power BI will automatically generate the appropriate formula.
- **Speed:** It's faster than manually writing custom columns, especially when you're unsure of the exact syntax or function names.
- **Flexibility:** It allows you to create a wide range of columns by simply providing examples, making it easy to extract specific data from existing

columns.

Creating Date-Related Features with Columns From Example

Let's assume we have an "Order Date" column in our dataset, and we want to create new columns that extract the weekday, month, or other date-related information. Here's how to do it using Columns From Example:

Example Scenario: Creating a Weekday Column

Suppose you want to create a column that shows the day of the week for each order date:

1. Navigate to the Power Query Editor:

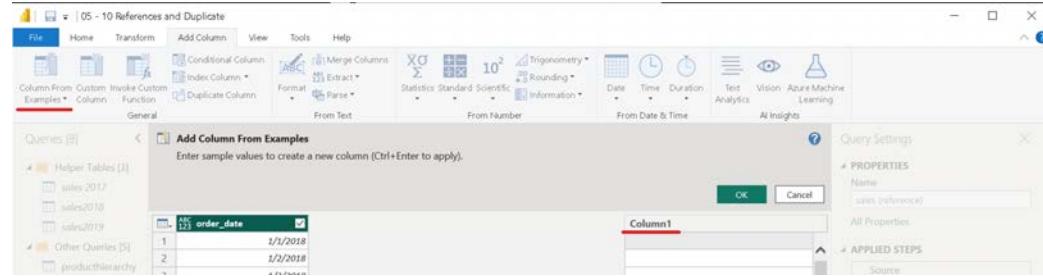
- In Power BI, go to the "Home" tab and click "Transform data" to open the Power Query Editor.

2. Select the Table and Column:

- In the Power Query Editor, select the table that contains your "Order Date" column. This will be the table from which you'll create the new columns.
- **Step 1:** Select the "Order Date" column.

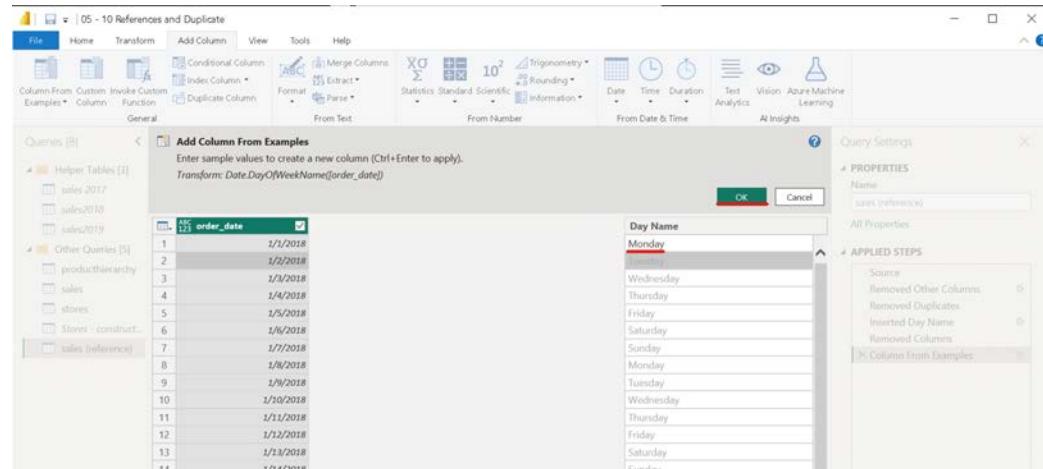
3. Use Columns From Example:

- Go to the "Add Column" tab in the ribbon and click on "Columns From Example." You have the option to apply this to "From All Columns" or "From Selection."
 - **From All Columns:** Power BI will consider all columns in the table when generating suggestions.
 - **From Selection:** Power BI will only consider the selected column(s) for suggestions.
- **Step 2:** Choose "Columns From Example" under the "Add Column" tab.



4. Input Sample Values:

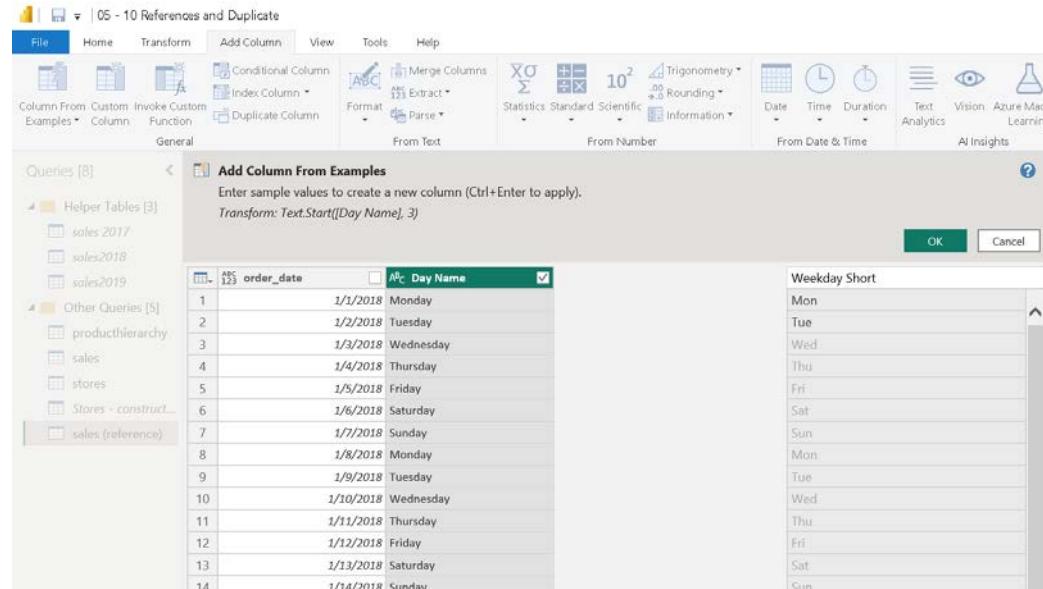
- In the newly created column, double-click a cell to start typing a sample value. For instance, if you want to extract the weekday from an "Order Date" of January 1, 2018, you could type "Monday."
- Power BI will automatically suggest and apply a formula to populate the entire column based on your input.
- Step 3:** In the first cell, type "Monday" (if the date corresponds to a Monday). Power BI will fill in the weekday for all rows based on this example.



- Step 4:** Select the "Day Name" column and **Step 2:** Choose "Columns From Example" under the "From Selection" tab.

5. Refine the Output:

- If Power BI doesn't generate the desired result on the first try, you can refine the column by providing more examples. For instance, if you input "Mon" instead of "Monday," Power BI might adjust the formula to extract the first three characters of the weekday, creating a column of short weekday names.
- **Step 5:** In the first cell, type "Mon" and second cell type "Tue".



6. Finalize the Column:

- Once the correct data is populated, you can rename the column to something meaningful, such as "Weekday" or "Weekday Short."
- Click "OK" to apply the transformation.

7. Add Additional Columns:

- You can repeat the process to add other date-related columns, such as "Month Name," "Year," or "Week Number," by providing appropriate examples.

Now, Power BI will have generated a new column listing the weekday for each order date, without the need for manual formula writing.

Connecting the New Column to the Data Model

After creating your new columns, you may want to integrate them into your data model for analysis:

1. Close and Apply:

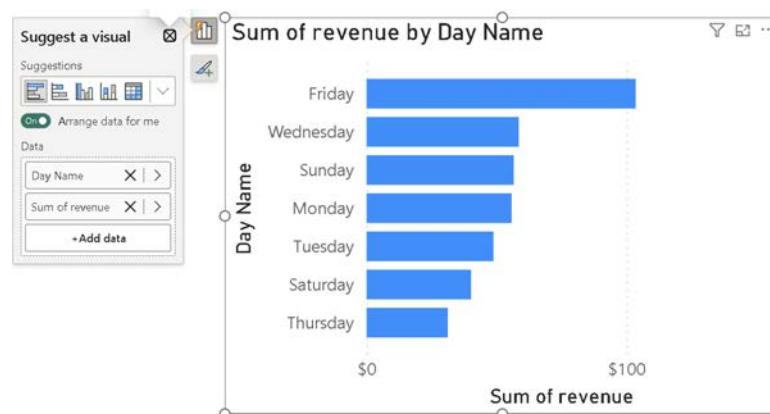
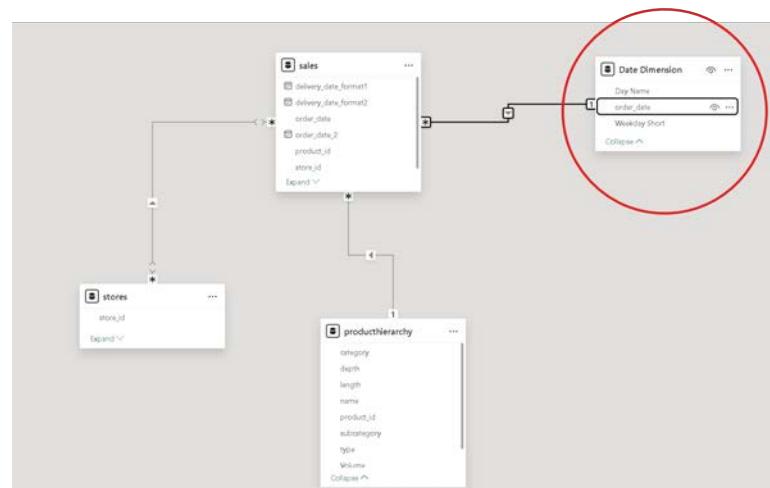
- Click "Close & Apply" in the Power Query Editor to save your changes and return to the main Power BI interface.

2. Establish Relationships:

- Go to the "Model" view to connect your new date dimension (if created) to your fact table. For example, connect the "Order Date" in your fact table (e.g., "Sales") to the corresponding date field in your new date dimension.

3. Use in Reports:

- You can now use the new columns, such as "Weekday," in your reports. For instance, you can create a visual that shows sales revenue by weekday to analyze patterns.



The **Columns From Example** feature in Power BI is a powerful and user-friendly tool for creating new columns without needing to write complex code. By simply providing examples, you can quickly generate useful data points like weekdays, months, and more. This feature not only speeds up

the process of data transformation but also makes it accessible to users who may not be familiar with the M language.

▼ Project 7

Project7.rar

▼ Conditional Columns

When working with categorical data in Power BI, such as days of the week, you may encounter challenges in sorting your data in a logical sequence. By default, Power BI sorts data based on the values in descending order, which might not align with the natural order you need like starting with Monday and ending with Sunday. To address this, we can create a custom sorting order using the "Conditional Column" feature.

Creating a Conditional Column for Custom Sorting

The first step to achieving the desired sorting order is to create a custom column that assigns a specific numerical value to each day of the week:

1. **Navigate to the Date Dimension:** Start by selecting your date dimension table, where you'll add the new conditional column.
2. **Add a Conditional Column:**
 - Go to the **Add Column** tab and select **Conditional Column**.
 - This opens a window where you can define the conditions for your new column.

The screenshot shows the Power BI desktop interface. In the top ribbon, the 'Add Column' tab is selected, and the 'Conditional Column' option is highlighted. Below the ribbon, the 'Conditional Column' dialog box is open, prompting the user to 'Create a new column that conditionally adds the values in the currently selected column.' The 'order_date' table is visible in the preview pane, showing columns for 'Day Name' and 'Weekday Short'. The preview pane also displays some statistics and the first few rows of the data.

3. Set Conditions for Each Day:

- Define a rule for each day of the week. For example:

The screenshot shows the 'Add Conditional Column' dialog box. It contains a table with six rows, each defining a condition based on the 'Day Name' column and its value. The conditions map the days of the week to numerical values from 1 to 7. The dialog also includes an 'Else' clause at the bottom. The 'OK' button is visible at the bottom right of the dialog.

Condition	Value	Output
If Day Name equals Monday	Monday	1
Else If Day Name equals Tuesday	Tuesday	2
Else If Day Name equals Wednesday	Wednesday	3
Else If Day Name equals Thursday	Thursday	4
Else If Day Name equals Friday	Friday	5
Else If Day Name equals Saturday	Saturday	6
Else		7

- If the **Day Name** equals "Monday", assign the value **1**.
- If the **Day Name** equals "Tuesday", assign the value **2**.
- Continue this pattern for all days of the week, up to **7** for Sunday.

- You can also set an **Else** condition, which applies if none of the specified conditions are met typically used for any other unspecified cases.

4. Use Operators Appropriately:

- Ensure that the appropriate operators are used based on the data type. For text fields like day names, use text-based operators. If dealing with numerical or date fields, use the corresponding operators.

5. Finalizing the Conditional Column:

- After setting all conditions, click **OK** to create the column.
- Optionally, convert this column to a **whole number** format for efficient data storage.

order_date	A ^B _C Day Name	A ^B _C Weekday Short	1 ² ₃ Custom
● Valid 100%	● Valid 100%	● Valid 100%	● Valid 100%
● Error 0%	● Error 0%	● Error 0%	● Error 0%
● Empty 0%	● Empty 0%	● Empty 0%	● Empty 0%
1000 distinct, 1000 unique	7 distinct, 0 unique	7 distinct, 0 unique	7 distinct, 0 unique
1/1/2018 Monday	Mon		1
1/2/2018 Tuesday	Tue		2
1/3/2018 Wednesday	Wed		3
1/4/2018 Thursday	Thu		4
1/5/2018 Friday	Fri		5
1/6/2018 Saturday	Sat		6
1/7/2018 Sunday	Sun		7
1/8/2018 Monday	Mon		1
1/9/2018 Tuesday	Tue		2
1/10/2018 Wednesday	Wed		3
1/11/2018 Thursday	Thu		4
1/12/2018 Friday	Fri		5
1/13/2018 Saturday	Sat		6
1/14/2018 Sunday	Sun		7
1/15/2018 Monday	Mon		1
1/16/2018 Tuesday	Tue		2
1/17/2018 Wednesday	Wed		3
1/18/2018 Thursday	Thu		4
1/19/2018 Friday	Fri		5
1/20/2018 Saturday	Sat		6
1/21/2018 Sunday	Sun		7
1/22/2018 Monday	Mon		1
1/23/2018 Tuesday	Tue		2
1/24/2018 Wednesday	Wed		3
1/25/2018 Thursday	Thu		4
1/26/2018 Friday	Fri		5
1/27/2018 Saturday	Sat		6

Sorting Using the Conditional Column

With the conditional column in place, you can now use it to sort your weekday data in the correct chronological order:

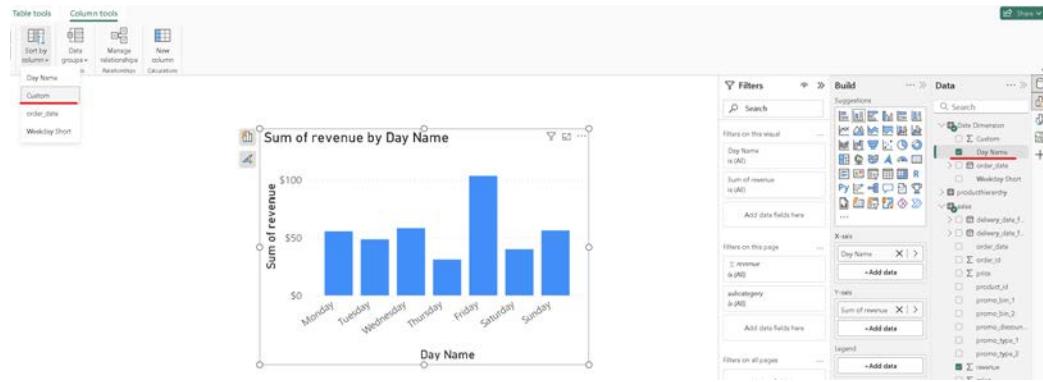
1. Select the Column to Sort:

- Go back to your dataset and select the original column you want to sort (e.g., Day Name).

2. Sort by the Conditional Column:

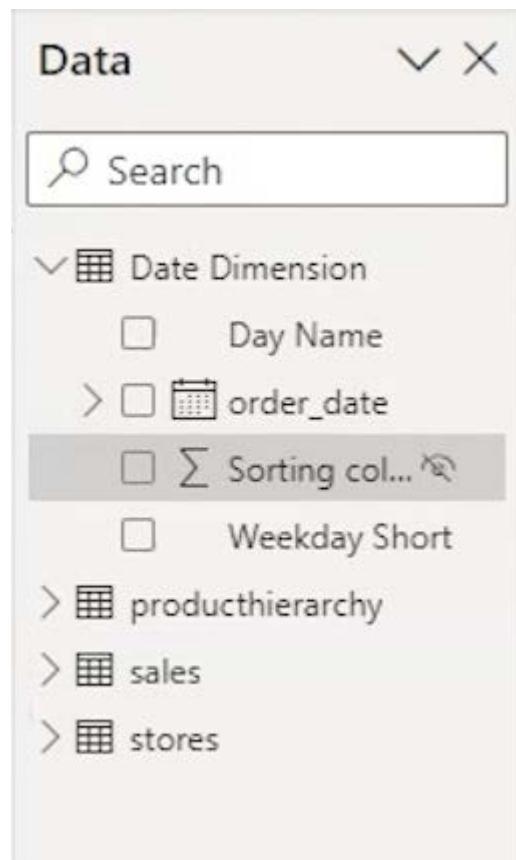
- Go to the **Column tools** tab and select **Sort by Column**.

- Choose the custom conditional column you just created. This will ensure the days of the week are sorted according to the numerical values you assigned.



3. Make the Sorting Column Invisible:

- To keep your report clean, you can hide the sorting column from view.
- Click on the three dots next to the column name and select **Hide**. The column will still function as the sorting key but won't appear in the report view.



By using conditional columns in Power BI, you can effectively manage the sorting and categorization of your data to ensure that it aligns with the logical sequence required for your analysis. This approach is particularly useful when working with categorical data, such as days of the week, where a natural order is essential for accurate reporting. With your custom sorting column in place, your Power BI visuals will be both accurate and intuitive, making your reports more effective and easier to interpret.

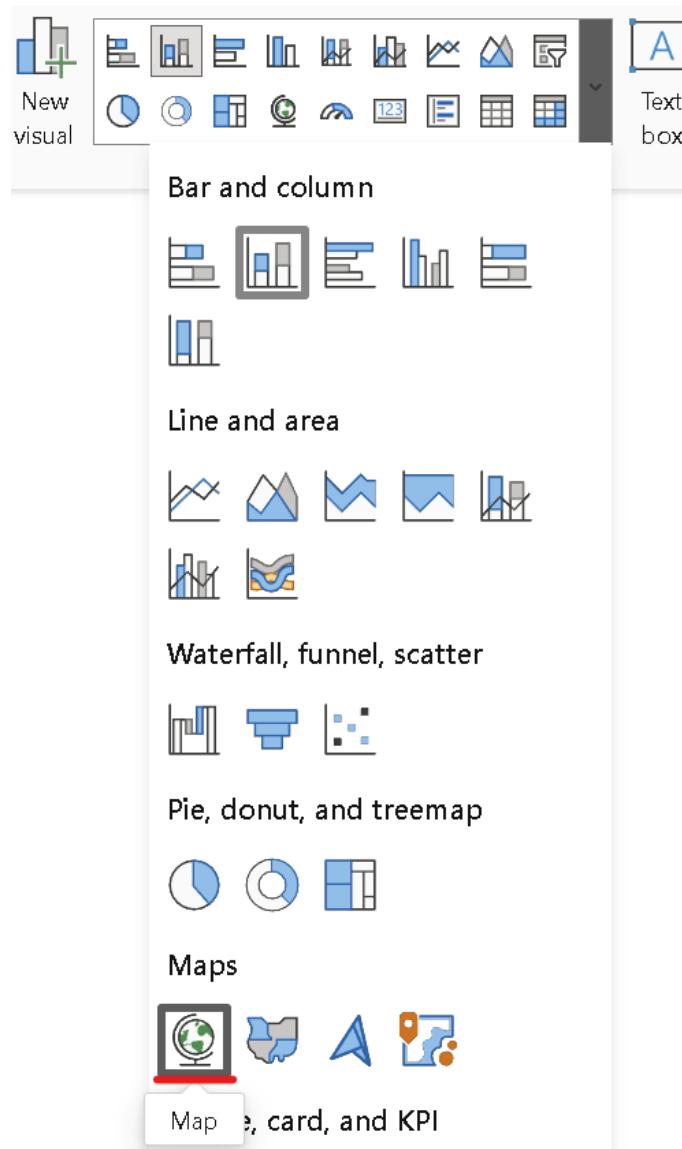
▼ Maps

In this lecture, we will explore how to effectively use map visualizations in Power BI to represent geographical data. Power BI offers powerful tools to display data such as cities, states, and coordinates on maps, allowing you to visualize patterns and trends across different locations.

To begin, let's create a map to visualize the cities in our dataset:

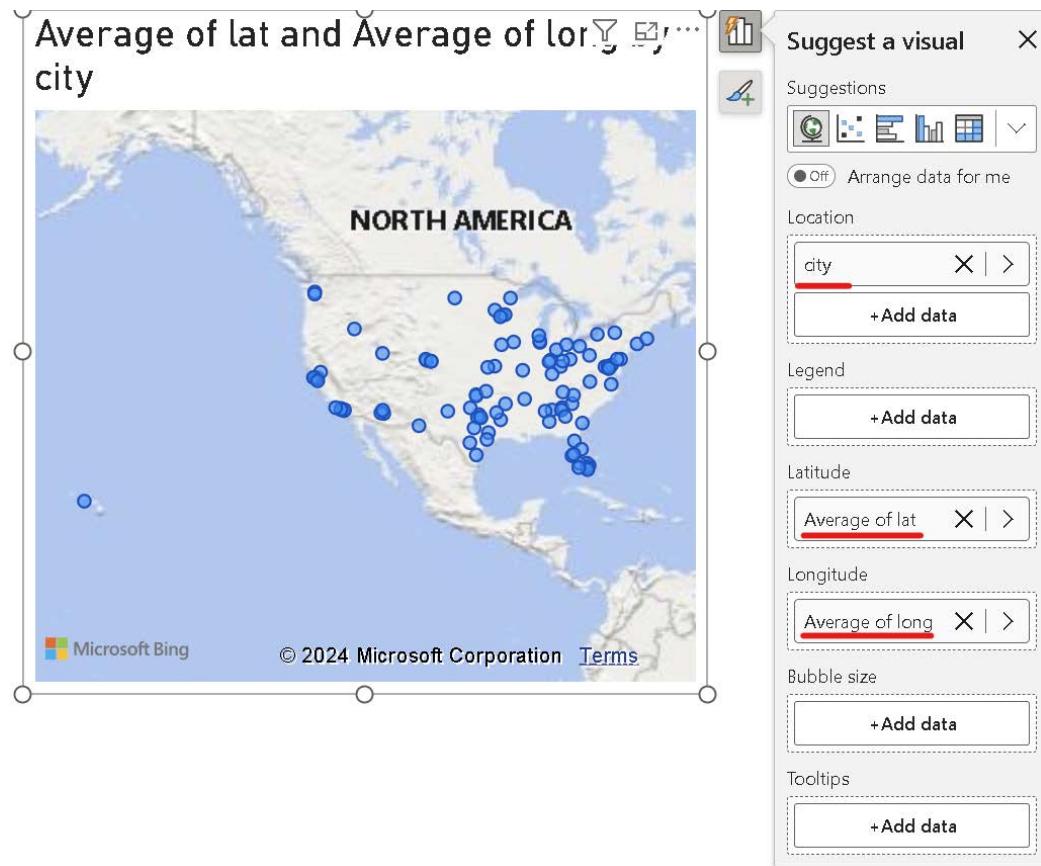
1. Add a Map Visual:

- Navigate to a new page in your Power BI report.
- In the **Home** tab, click on the dropdown and select the "Map" visual.



2. Drag Data into the Map:

- Drag the **City** field into the location field of the map. The map will automatically plot the cities.
- If your data includes cities with the same name in different countries (e.g., Birmingham in the UK and USA), Power BI might incorrectly plot them. This is where additional geographical fields, such as **State** or **Latitude** and **Longitude**, become essential.



3. Correcting Location Issues:

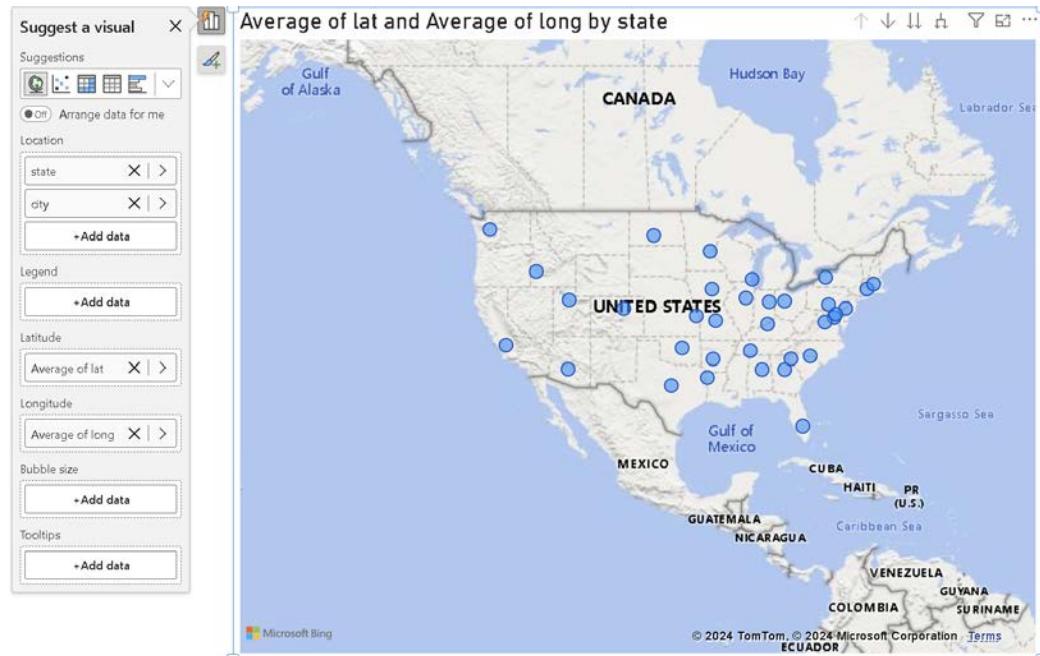
- To ensure accuracy, drag the **Latitude** and **Longitude** fields into their respective sections in the map visual. This will correct any misplacement of cities by precisely locating them based on coordinates.
- By default, Power BI may set the latitude and longitude fields to aggregate as "Average." This setting works well, but if you prefer, you can change it to "Do Not Summarize." Note, however, that doing so will require removing the location field, which might result in losing some city labels.

Enhancing Your Map with Hierarchies and Data

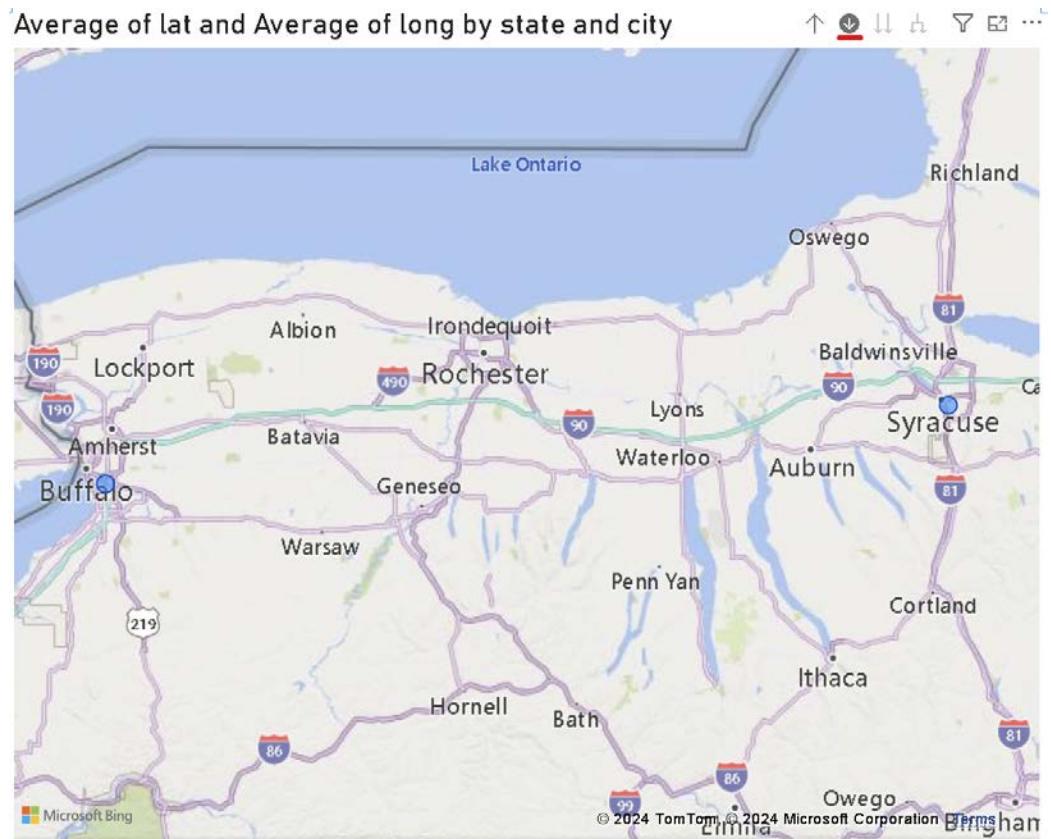
You can add more layers of information to your map to make it more informative:

1. Add Hierarchical Data:

- For example, you can drag the **State** field into the map to create a hierarchy. This allows you to drill down from the state level to the city level.

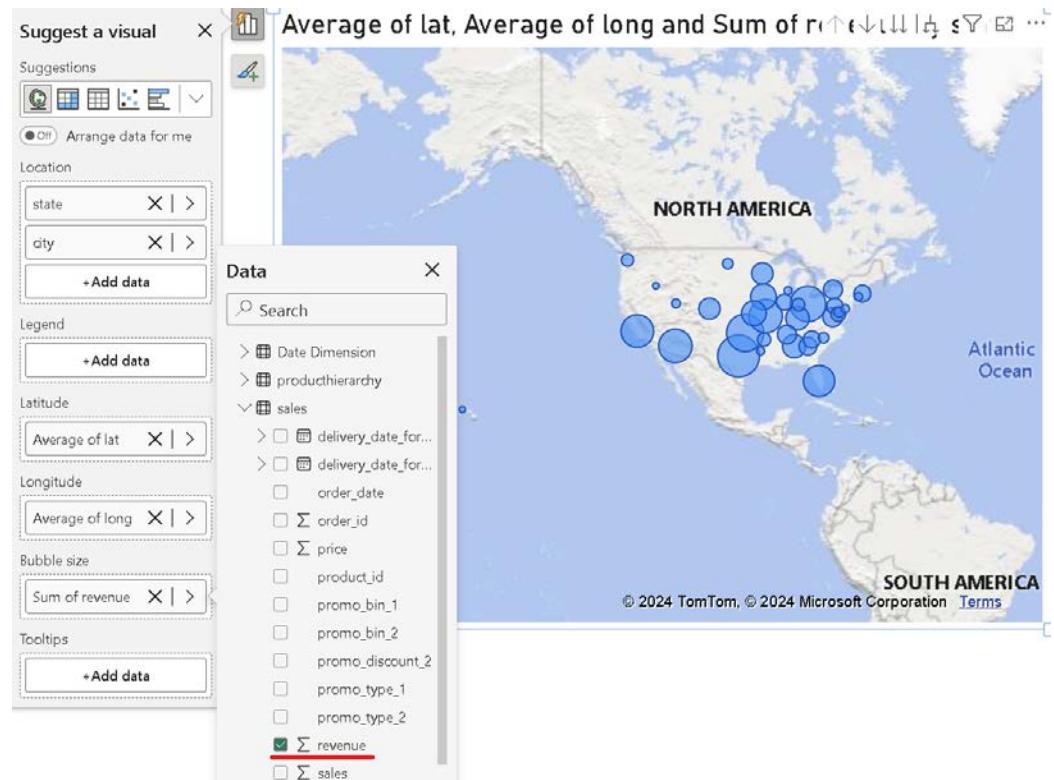


- Use the drill-down feature to explore data at different levels of granularity, such as viewing all cities within a specific state.



2. Visualizing Data with Bubble Size:

- To visualize data like revenue across different locations, drag the **Revenue** field into the **Bubble Size** section. The size of the bubbles on the map will now represent the revenue in each location, providing a quick visual comparison.



Customizing Map Appearance

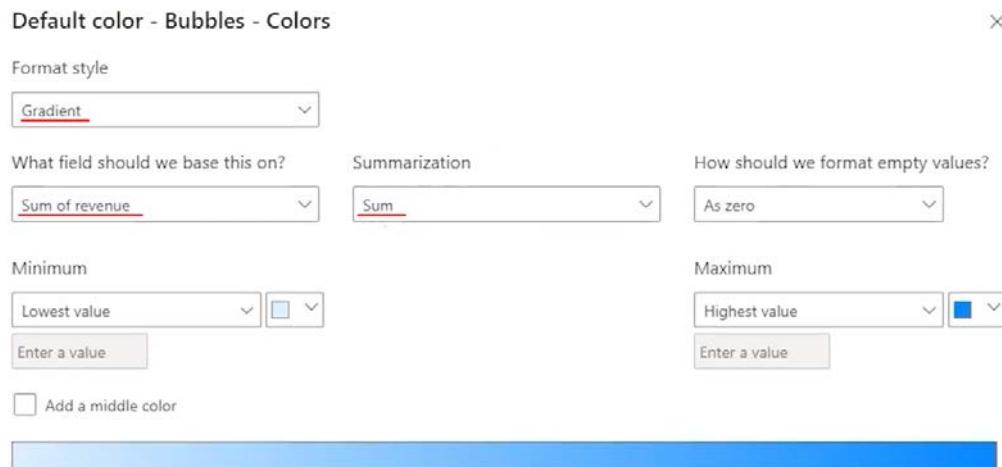
Power BI offers several customization options to enhance the visual appeal and functionality of your map:

1. **Change Bubble Size and Color:**

- In the **Format** pane, you can adjust the bubble size or change the colors.



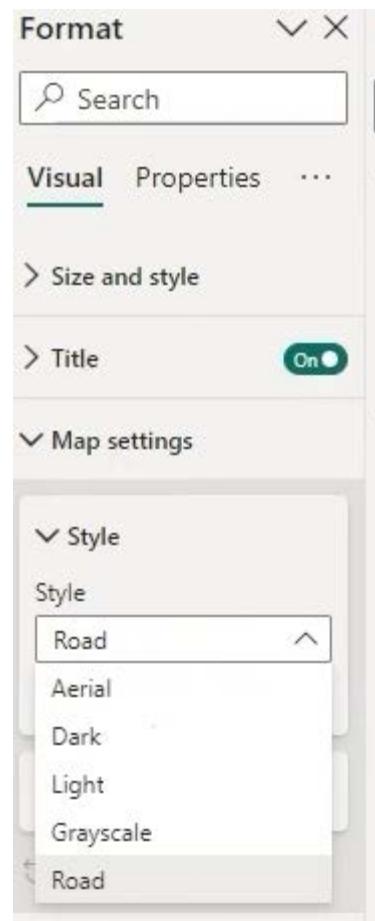
- For instance, you can use a gradient color scheme where darker colors represent higher revenues.



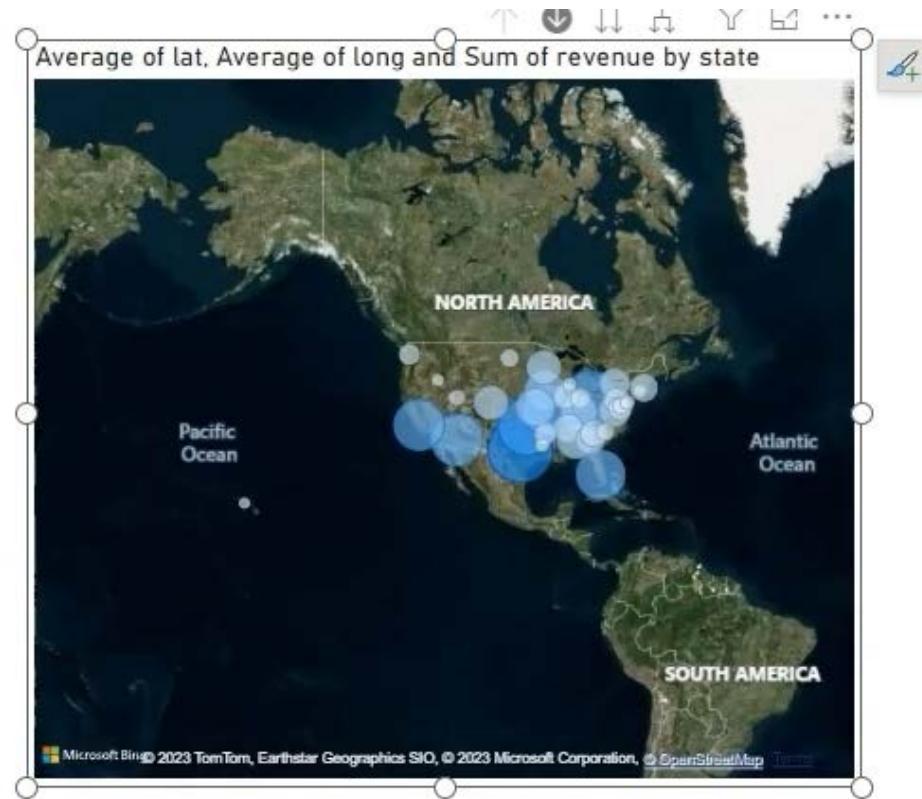


2. Map Style and Settings:

- Modify the map's appearance by choosing different styles such as **Road, Aerial, Light, or Dark** modes.



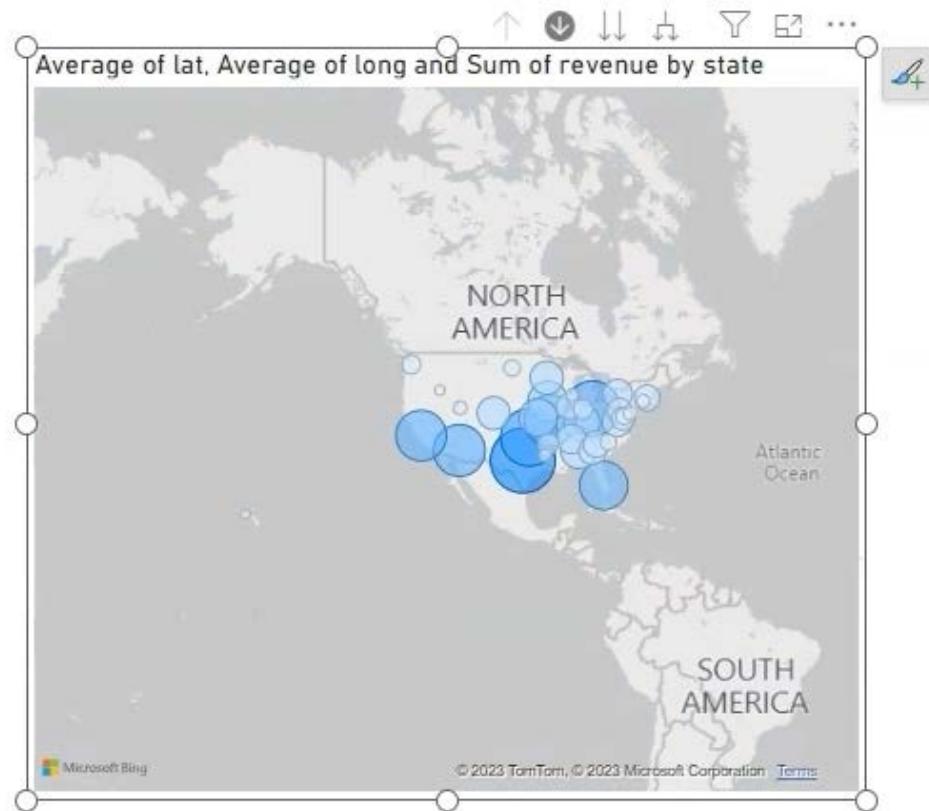
- o Aerial:



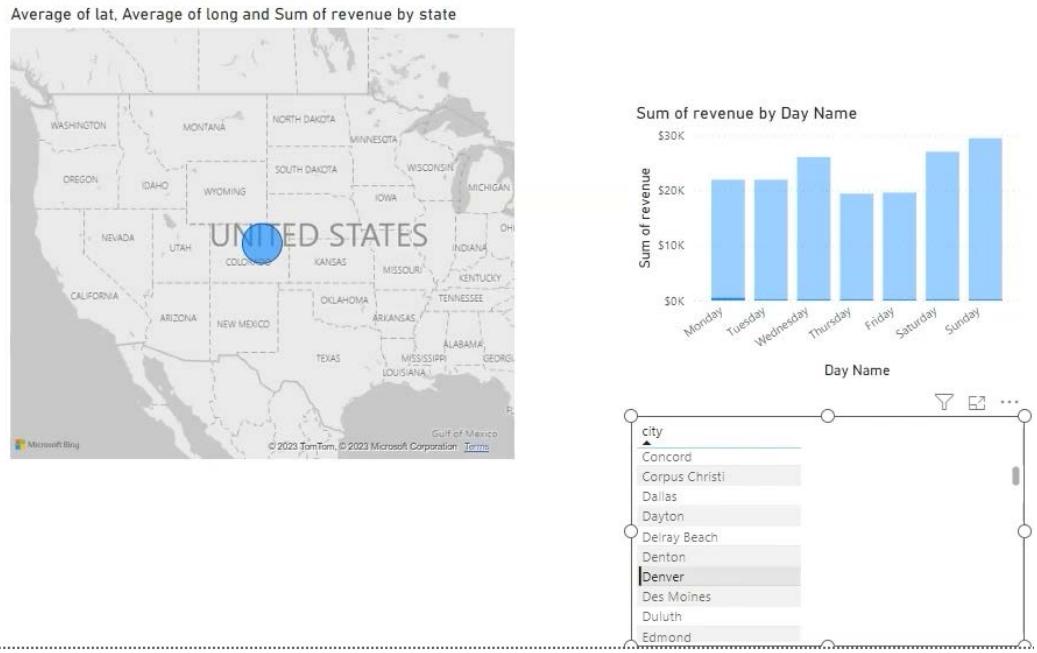
- o Light:



- Grayscale:

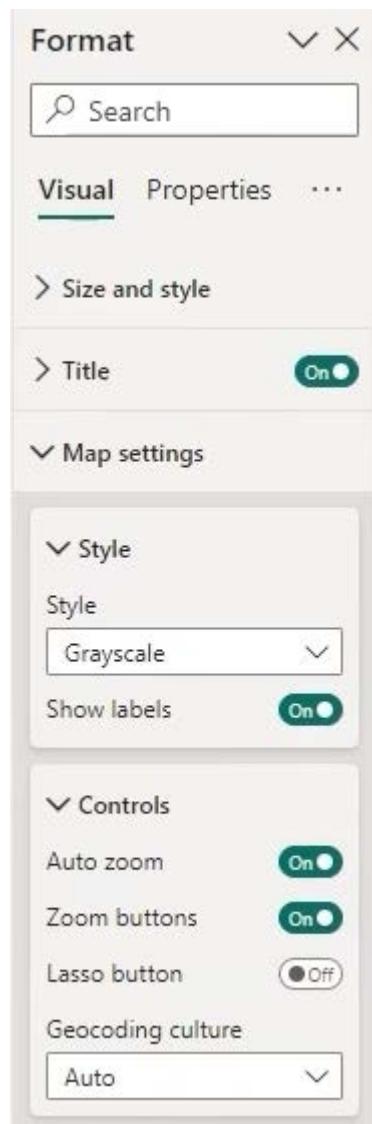


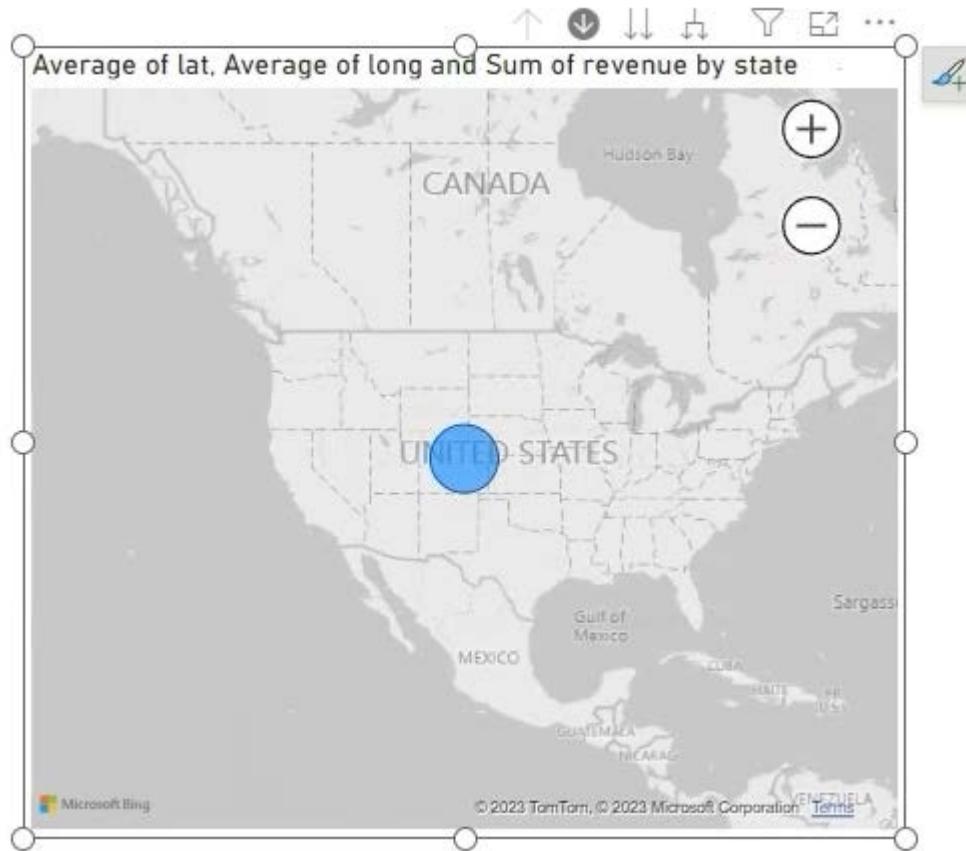
- Enable **Auto Zoom** to automatically zoom into selected areas when interacting with the map.



3. Add Interactive Features:

- You can enable zoom buttons to make navigation easier.

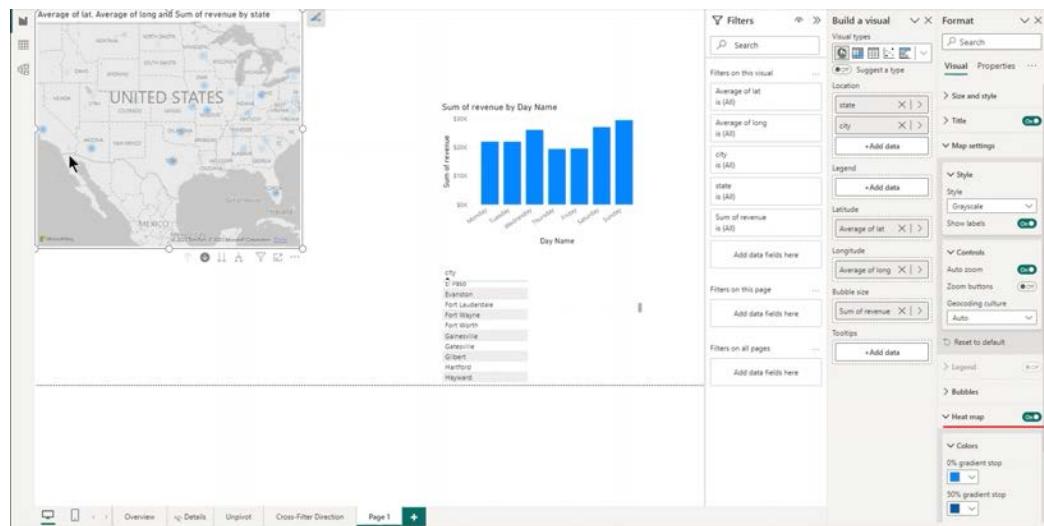




- Use the **Lasso** tool to select multiple cities or areas on the map, which is useful for highlighting and comparing different regions simultaneously.

4. Utilizing Heat Maps:

- Power BI allows you to switch to a **Heat Map** mode, which highlights areas with high concentrations of data points. This feature is particularly useful when dealing with large datasets spread across many locations.



By mastering map visualizations in Power BI, you can effectively convey complex geographical data, making your reports more insightful and easier to understand.

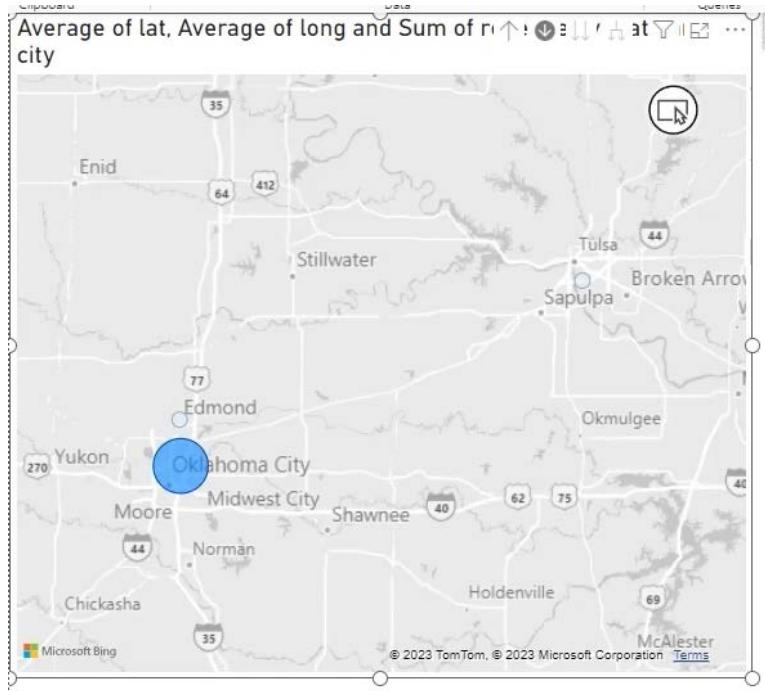
▼ Filled Maps

In the previous lecture, we discussed how map visuals can be used not only to display data but also to interact with other data points in your report. For example, selecting a location on the map can filter the data in other visuals, and vice versa. While standard maps work well for data visualization, when the primary goal is to filter data, a Filled Map is often more effective.

Converting to a Filled Map

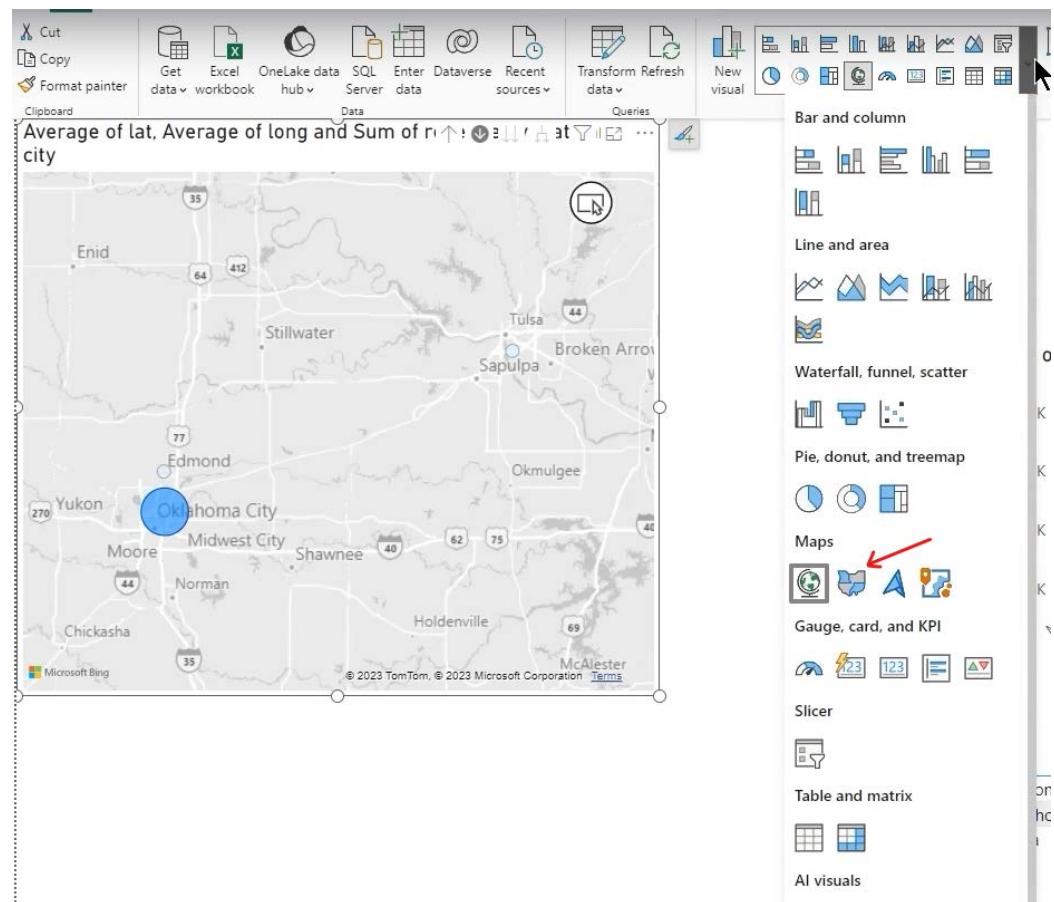
Let's demonstrate how to convert a standard map into a Filled Map:

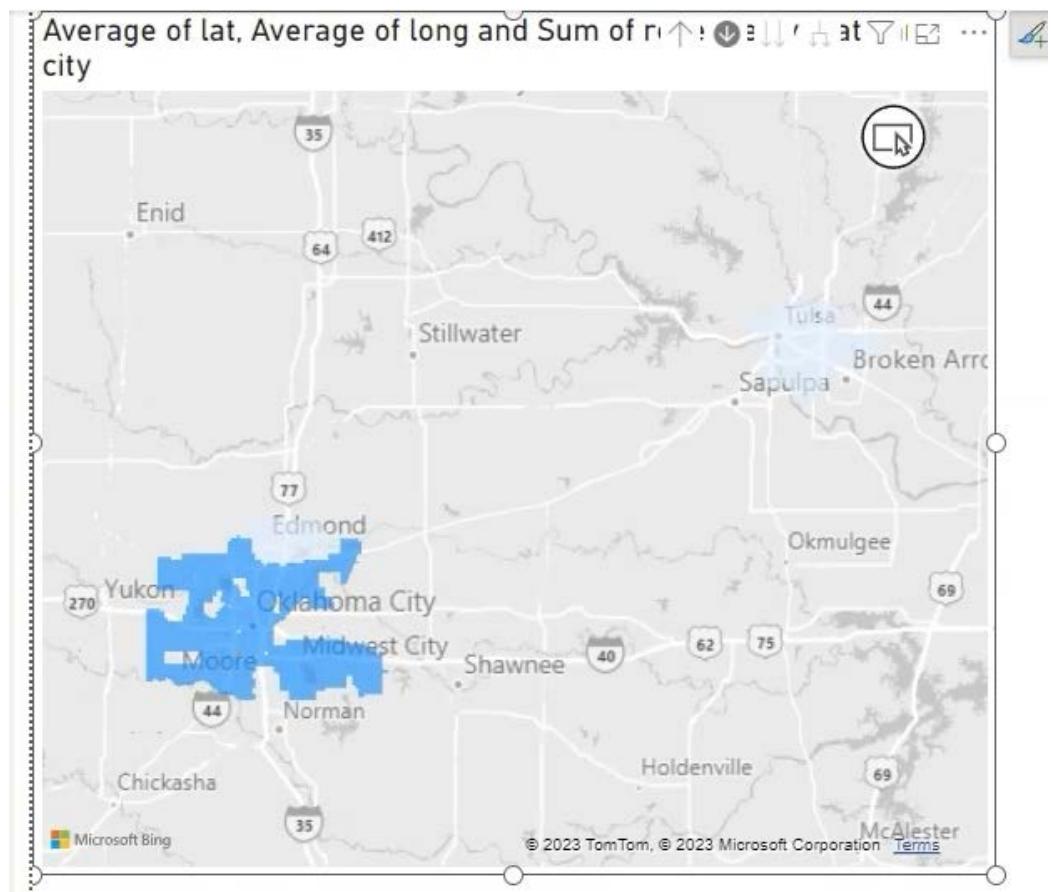
- 1. Select the Map Visual:** Start by selecting the map visual you created earlier.



2. Change to Filled Map:

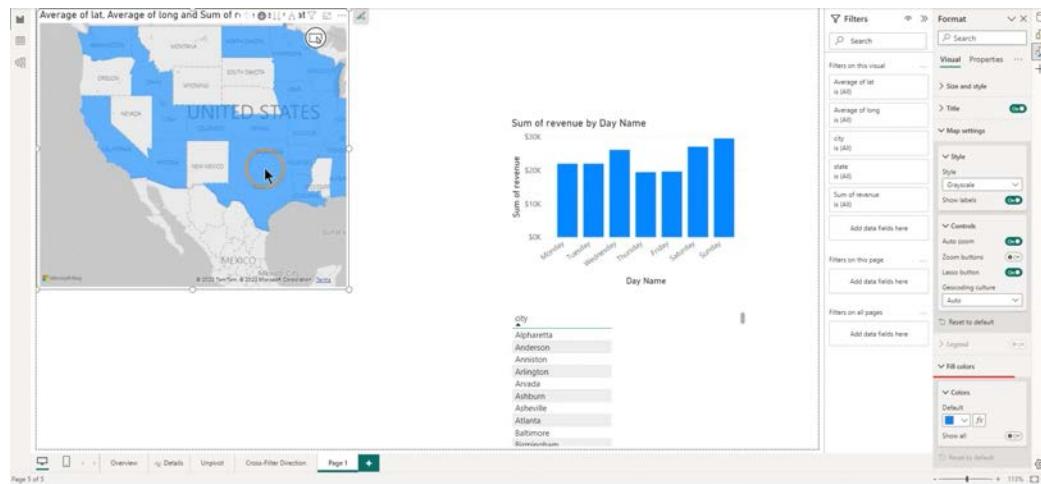
- With the map visual selected, navigate to the **Visualizations** pane.
- Change the visual type from a standard map to a **Filled Map**. This will automatically adjust the map to fill the geographical areas with colors based on your data.





3. Configure the Filled Map:

- If no specific area is selected, the map will display all regions (e.g., all states) filled with colors according to the data.
- Go to the **Format** tab to adjust the appearance. You can apply conditional formatting to the fill colors or use a legend to differentiate between regions.



Using Filled Maps for Filtering

Filled Maps are particularly useful when the goal is to filter other data based on geographical areas:

- **Filtering with Filled Maps:** Selecting a region on the Filled Map will filter the other visuals in the report according to the selected area. This makes it easier for users to quickly identify and interact with data from specific locations.
- **Clear Indication of Filtering Purpose:** Because Filled Maps emphasize geographical boundaries and fill areas with color, they intuitively indicate that their primary function is filtering rather than detailed data visualization.

Customizing the Filled Map

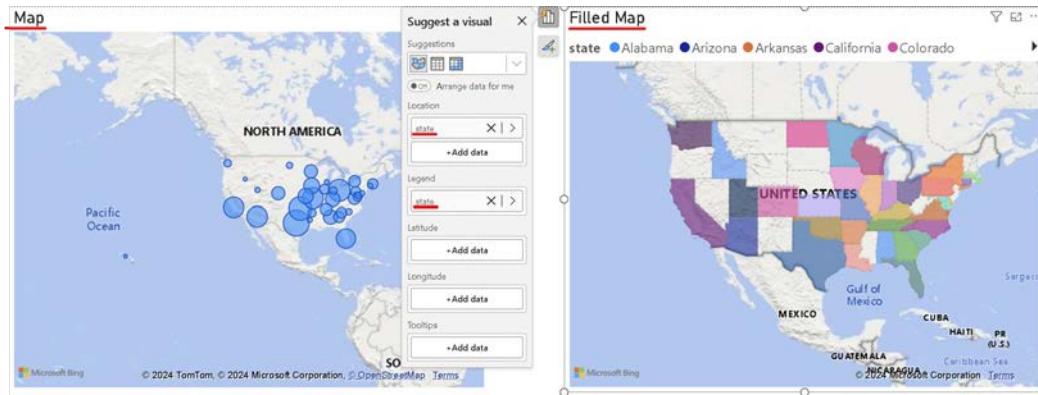
To enhance the clarity and effectiveness of your Filled Map, you can make several customizations:

1. **Remove Unnecessary Fields:** Simplify the map by removing unnecessary fields such as latitude and longitude, focusing instead on regions like states or countries.
2. **Add or Modify Legend:**
 - You can add a legend to show what each color represents, helping users understand the data distribution across regions.

- Alternatively, if the legend isn't needed, it can be turned off in the **Format** pane.

3. Adjust Borders and Colors:

- Modify the map's appearance by changing the fill colors or adding borders to regions, making it clear that the map is intended for filtering.



Choosing Between Standard Maps and Filled Maps

- Use Filled Maps:** When your primary goal is to use the map as a filter for other data, Filled Maps are more effective. They provide a clear visual indication of regions and are intuitive for users who need to filter data based on geographical areas.
- Use Standard Maps:** If the map is intended to provide detailed insights, such as visualizing specific data points or comparing values across locations, the standard map is more appropriate.

Filled Maps in Power BI offer a powerful way to filter and interact with data based on geographical locations. They are particularly useful when the primary purpose of the map is to act as a filter rather than a detailed data visualization tool. By understanding when and how to use Filled Maps, you can create more intuitive and effective reports that meet your users' needs.

▼ Forecasts

In this section, we'll explore how to leverage Power BI's forecasting feature to predict future values, such as revenue, without needing extensive

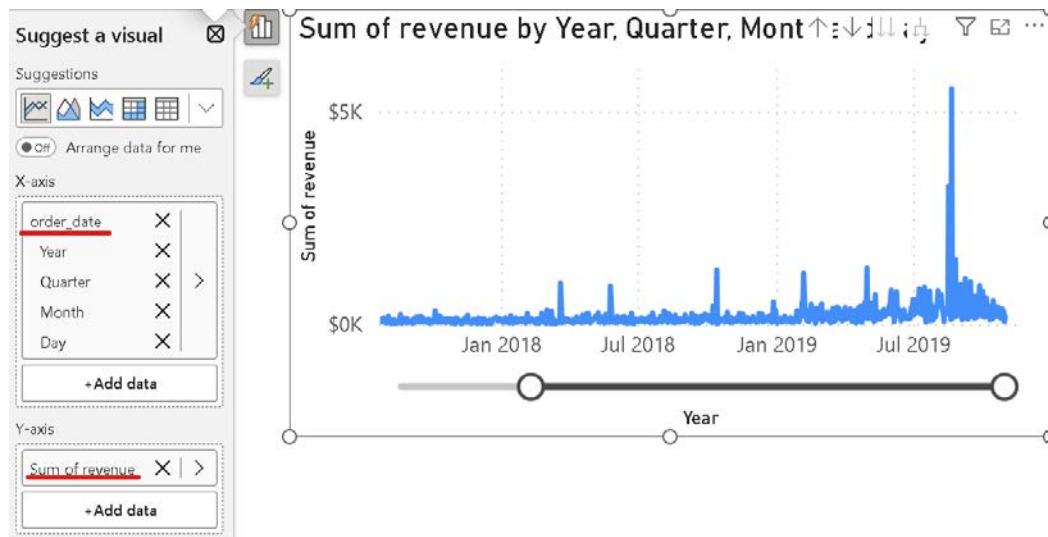
knowledge of machine learning. This capability allows you to provide valuable insights to your clients by projecting future trends based on historical data.

Adding a Forecast to a Line Chart

Let's start by creating a forecast for revenue over the next few days:

1. Create a Line Chart:

- First, create a line chart by selecting the appropriate visual.
- Add the **Revenue** field to the values section and the **Order Date** from your date dimension to the axis. This will display the historical revenue data over time.

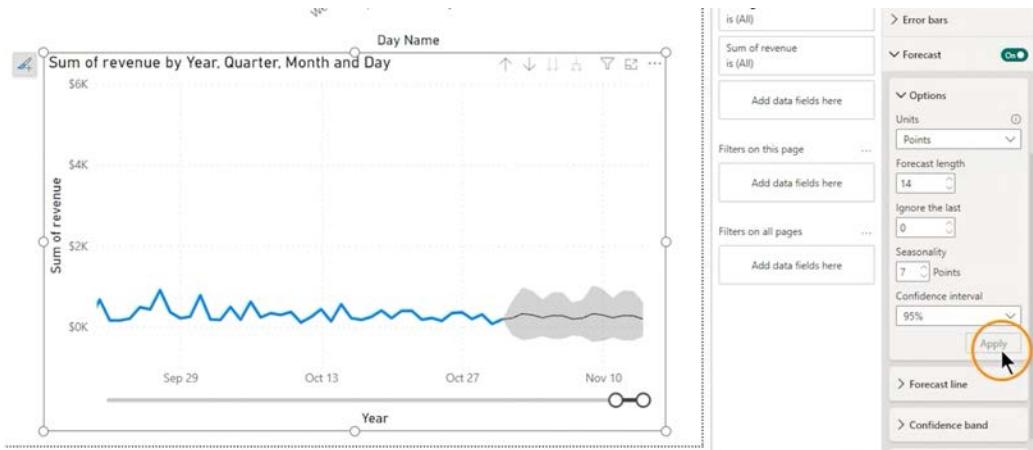


2. Enable Forecasting:

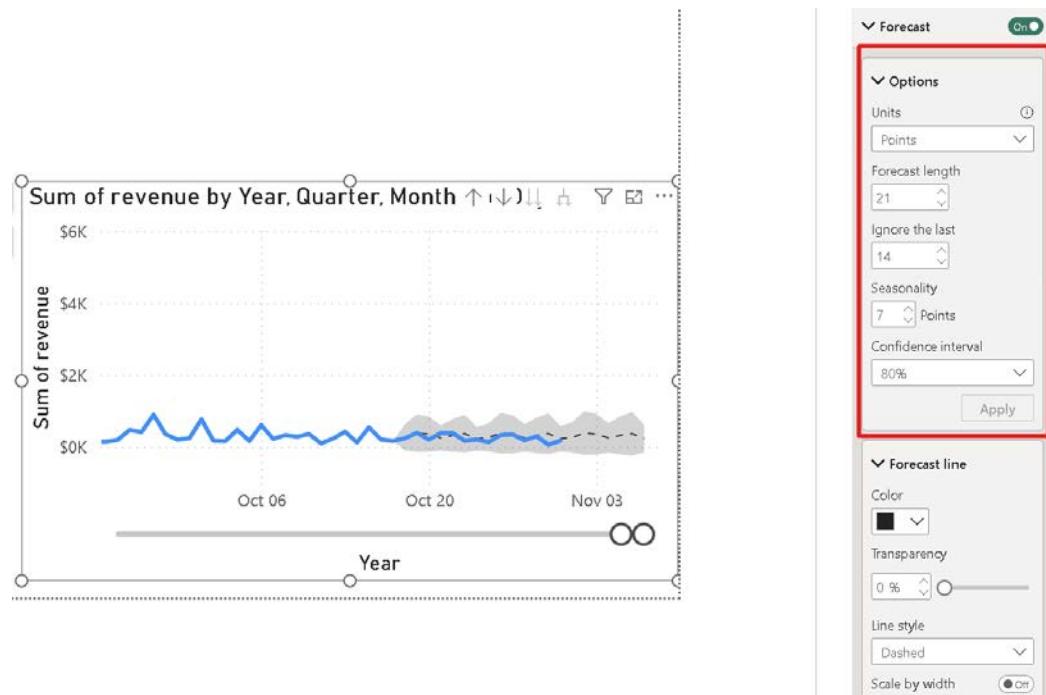
- To add a forecast, expand the hierarchy levels in the date field if you want a more granular forecast (e.g., daily predictions).
- In the **Analytics** pane, you'll find the option to add a **Forecast**. Turn this option on to generate a basic forecast based on your historical data.

3. Customizing the Forecast:

- **Adjust the Forecast Length:** Specify the number of future periods you want to forecast. For example, you might want to forecast revenue for the next 7 or 14 days.



- **Set Seasonality:** If you know your data has a seasonal pattern (e.g., higher sales on weekends), you can set a seasonality value. For example, a seasonality of 7 points would account for weekly patterns.
- **Modify Confidence Interval:** The confidence interval represents the range within which the forecast is likely to fall. By default, it's set at 95%, meaning the model is 95% confident that the future values will fall within the shaded area on the chart. You can adjust this interval to be more or less strict, which will change the size of the shaded area.



4. Visual and Design Adjustments:

- You can customize the appearance of the forecast line, such as changing its color, making it dashed, or adjusting its transparency.
- The confidence interval area can also be styled differently, either as a filled region or as a line.
- Add tooltips to provide additional context when users hover over the forecasted values.

Evaluating Forecast Accuracy

It's important to assess how accurate the forecast might be:

- **Backtesting:** One way to evaluate the reliability of the forecast is by excluding the most recent data (e.g., the last 7 days) and comparing the forecasted values with the actual historical data. This gives users a sense of how well the forecast model is performing.
- **Confidence Intervals:** The gray area around the forecast line is the confidence interval, indicating where the actual values are expected to fall. Adjusting this interval can help you understand the potential variability in the forecast.

Power BI's forecasting feature is a powerful tool that allows you to predict future trends with ease. By customizing forecast length, seasonality, and confidence intervals, you can tailor the forecast to fit the specific needs of your analysis. This can be particularly valuable when clients or stakeholders want to see projections based on the data you've provided.

▼ Find Anomalies

In this section, we'll explore the "Find Anomalies" feature in Power BI, which allows you to identify and analyze unexpected data points, or anomalies, in your datasets. This feature is particularly useful for spotting irregularities, such as unusually high or low values, and understanding the factors that may have contributed to these anomalies.

Enabling Anomalies Detection in a Line Chart

To use the anomaly detection feature, follow these steps:

1. Create a Line Chart:

- Start by creating a line chart with your data. For example, you might plot revenue over time using the **Order Date** as the axis and **Revenue** as the value.

2. Disable Forecasting:

- If you've already enabled the forecast feature on this chart, you'll need to disable it first, as anomaly detection and forecasting cannot be used simultaneously.

3. Enable Find Anomalies:

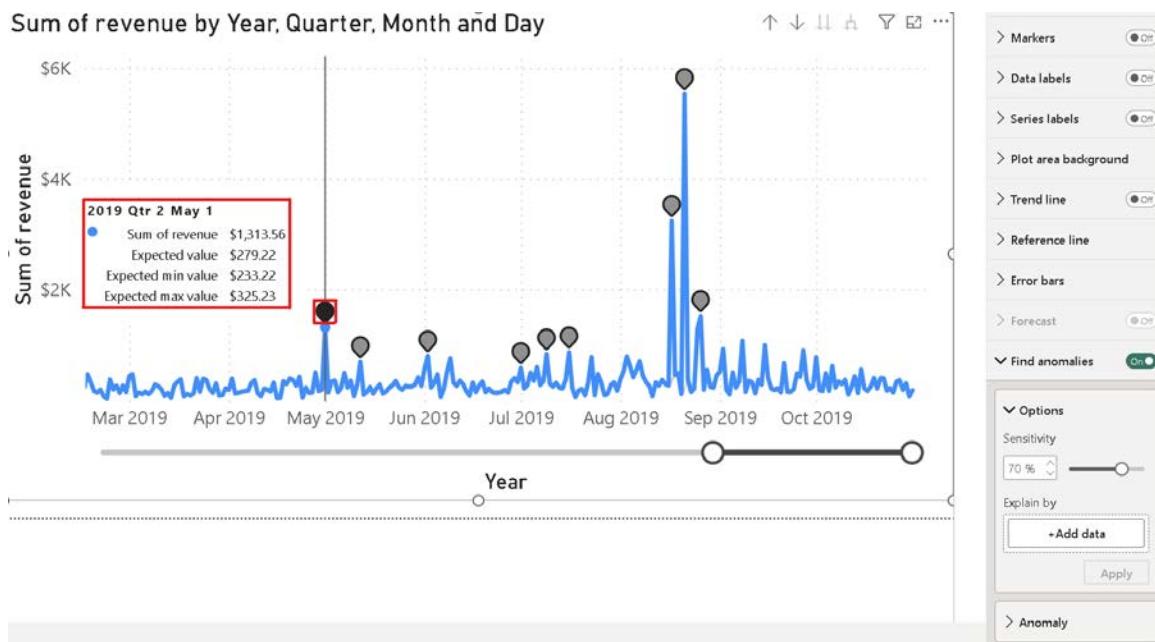
- Go to the **Analytics** pane (available under the **Format** tab) and turn on the **Find Anomalies** option.



- The chart will automatically scan for any data points that fall outside the expected range, identifying them as anomalies.

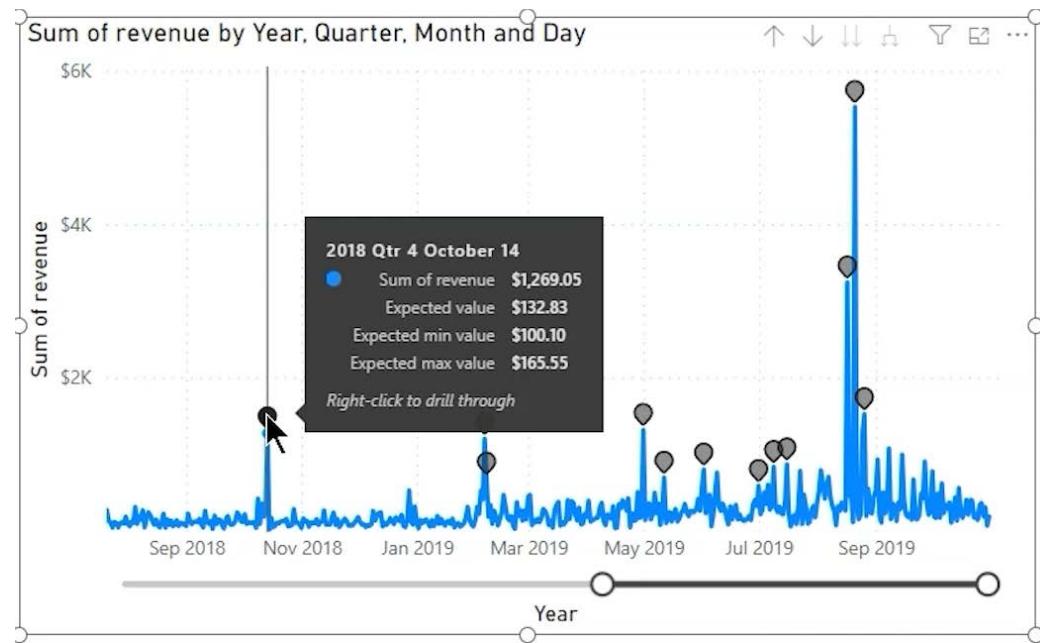
Understanding and Analyzing Anomalies

Once anomalies are detected, Power BI provides additional insights to help you understand why these anomalies occurred:



1. Hover Over Anomalies:

- As you hover over an identified anomaly on the chart, a tooltip will display the actual value compared to the expected range. For example, if the actual revenue was \$1,269 but the expected value was around \$130, this significant deviation would be flagged as an anomaly.



2. View Detailed Anomaly Information:

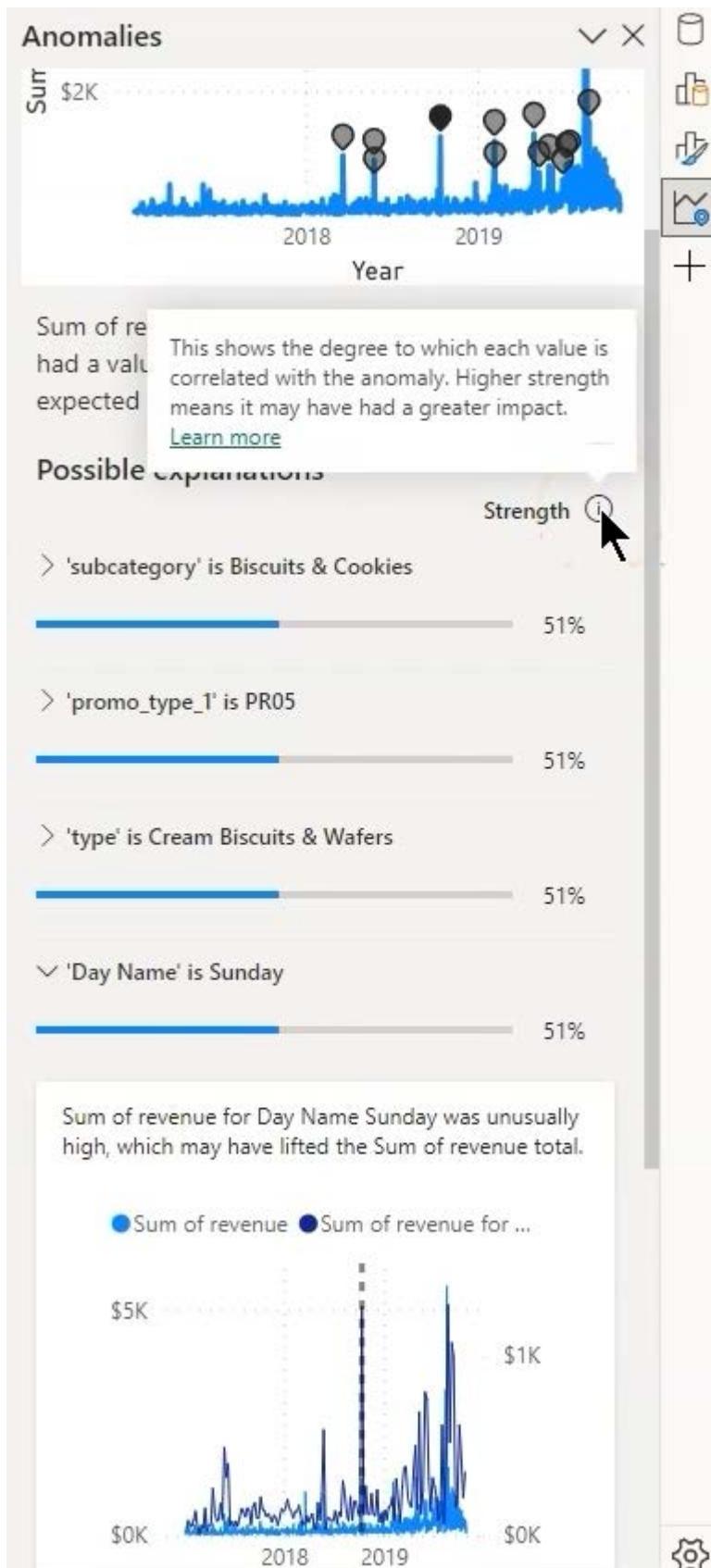
- Clicking on an anomaly opens the **Anomalies Pane**, which provides detailed explanations for the irregularity. Power BI attempts to explain the anomaly by analyzing related data points.



- For instance, the pane might reveal that the anomaly was influenced by factors such as a specific subcategory having unusually high sales, or that the anomaly occurred on a Sunday, when sales are generally higher.

3. Strength of Contributing Factors:

- Power BI assigns a strength percentage to each factor that contributed to the anomaly, indicating how strongly that factor is correlated with the deviation.



- For example, if "Weekday: Sunday" has a strength of 51%, this suggests that the fact that the anomaly occurred on a Sunday is likely a significant reason for the unexpected high revenue.

4. Correlation Indicators:

- Power BI also provides visual cues, such as icons, to indicate the degree of correlation between the identified factor and the anomaly. This helps you quickly assess how likely it is that a particular factor caused the anomaly.

When to Use Anomalies Detection

Anomalies detection is a valuable tool when you need to:

- Identify outliers in your data that may require further investigation.
- Understand underlying factors that contribute to unexpected data points.
- Improve data quality by spotting and analyzing irregularities that may indicate errors or unusual events.

However, if your primary goal is to forecast future trends rather than analyze past irregularities, it's better to use the forecast feature instead of anomaly detection. Each feature serves a different purpose, and the choice depends on your specific analysis needs.

The "Find Anomalies" feature in Power BI enhances your ability to detect and understand outliers in your data. By providing insights into what might have caused these anomalies, it allows you to make more informed decisions and gain deeper insights into your data patterns.

▼ 6- Advance Dashboard Enhancements

▼ Drillthrough with Buttons

In this section, we'll explore how to enhance your Power BI dashboards by enabling drillthrough functionality using buttons. This approach is often more intuitive and user-friendly compared to the default drillthrough method, which requires right-clicking and navigating through context menus —actions that some users might overlook or find cumbersome.

Why Use Buttons for Drillthrough?

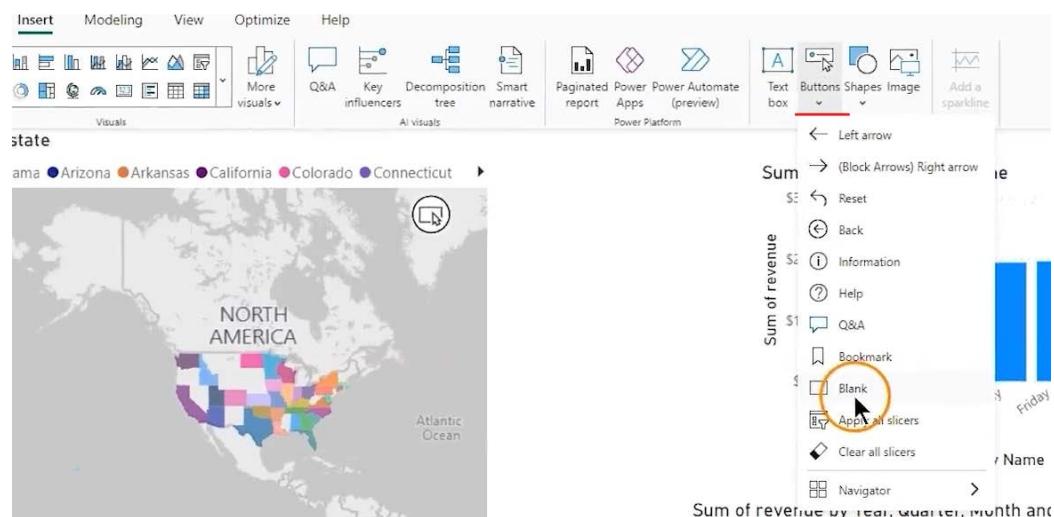
The standard drillthrough feature in Power BI involves right-clicking on a data point, selecting "Drillthrough," and then choosing a target page. While effective, this method might not be immediately obvious to all users, especially those who are less familiar with Power BI. By using a button for drillthrough, you can make this functionality more accessible and visually appealing, ensuring that users can easily interact with your reports.

Setting Up Drillthrough with a Button

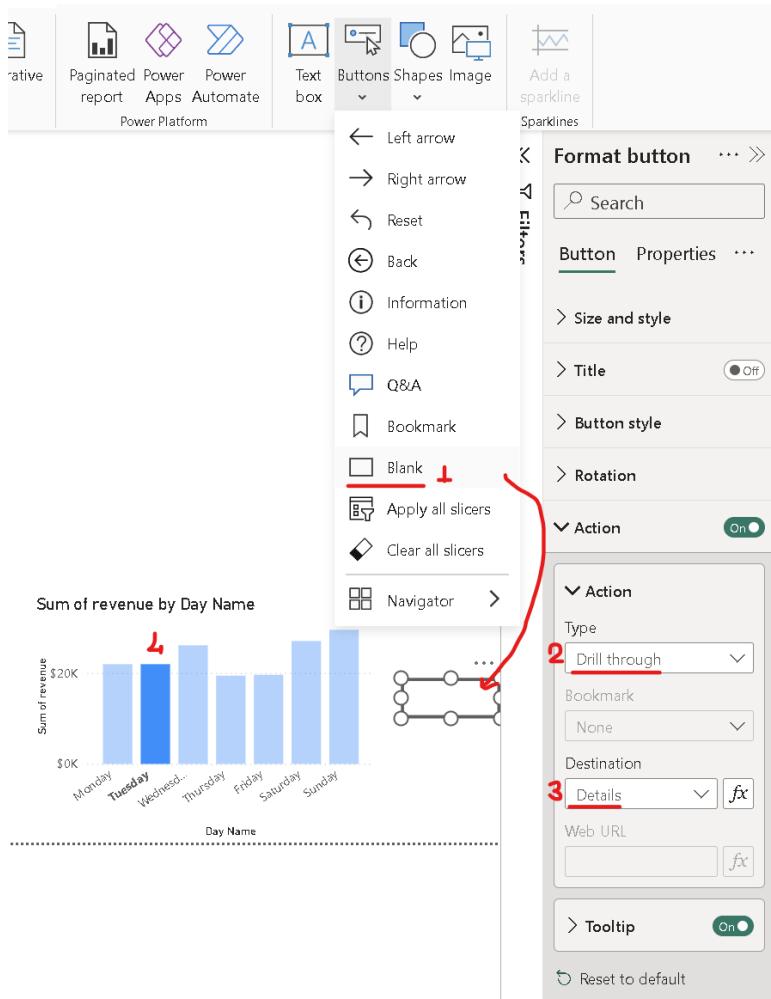
Here's how you can set up a drillthrough using a button in Power BI:

1. Add a Button to Your Page:

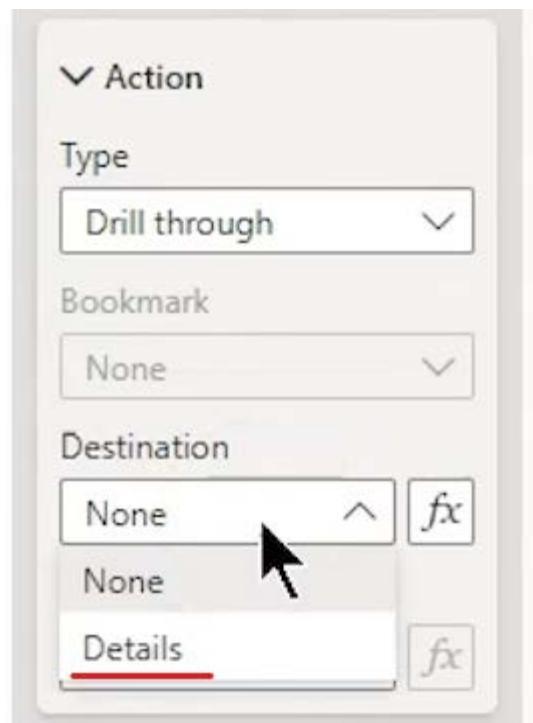
- Go to the **Insert** tab in Power BI Desktop.
- Select **Button** from the available options. You can choose from various predefined buttons, but for maximum flexibility, it's often best to start with a **Blank** button.



2. Configure the Button Action:

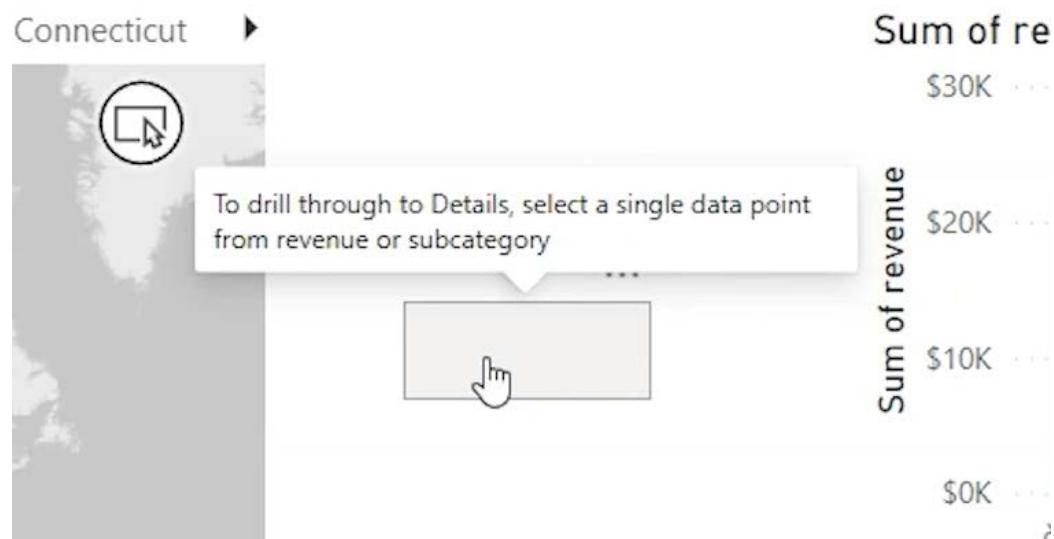


- With the button selected, navigate to the **Format** pane.
- Under **Action**, toggle the action switch to **On**.
- Expand the **Action** section, and set the **Type** to **Drillthrough**.
- Choose the destinationpage (e.g., "Details" or any other relevant page) where you want the drillthrough to navigate.



3. Handling Disabled State:

- Initially, the button may appear grayed out and will display a message when hovered over, such as "To drillthrough, select a single data point." This occurs because the button requires a selection from specific fields (e.g., Revenue or Subcategory) on the current page to activate.



- To address this, ensure that your data visualizations are set up correctly, allowing users to select data points that correspond to the drillthrough target.(We could now select this day because in here we have the revenue aggregated)



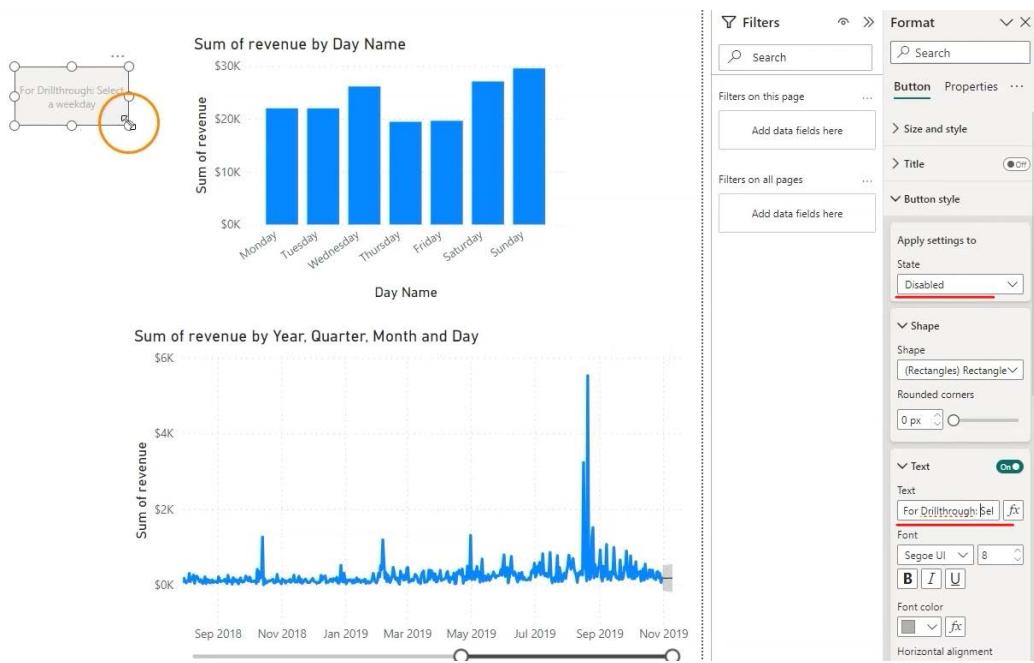
4. Customize Button Appearance:

- You can further enhance user experience by customizing the button's appearance based on its state (e.g., default, disabled, hover, and clicked states).
- For instance, when the button is disabled, you might display the text "Select a data point to drillthrough." When it's enabled, change the text to something like "Click to drillthrough."



To Customize the Button:

- In the **Format** pane, adjust the text, size, and color for each button state.
- For example, you can change the text under **Default** to "Select a data point to drillthrough" and under **Enabled** to "Click to drillthrough."
- Modify the button's shape, such as rounding the corners, or adjust the colors to better fit the design of your dashboard.



5. Test the Drillthrough Functionality:

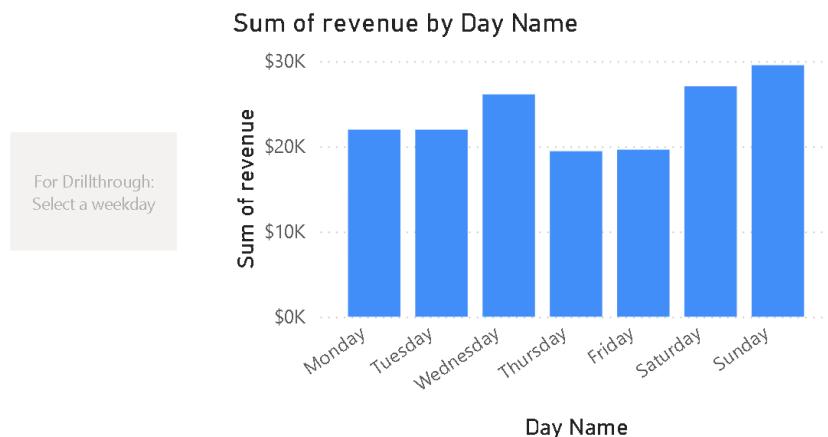
- After setting up the button, test the drillthrough by selecting a data point on your report page. The button should now be active.
- Click the button (remember to hold the **Ctrl** key while in Power BI Desktop) to navigate to the drillthrough page.

	Order Date	Revenue	Orders
1	1/21/2017	\$0.00	1
2	2/21/2017	\$0.00	1
3	4/4/2017	\$0.00	1
4	6/13/2017	\$0.00	1
5	7/18/2017	\$0.00	1
6	8/23/2017	\$0.00	1
7	9/19/2017	\$0.00	1
8	10/17/2017	\$0.00	1
9	10/24/2017	\$0.00	1
10	12/1/2017	\$0.00	2
Total		\$46.00	81

- Once on the drillthrough page, the data should be filtered according to the selection you made on the previous page.

6. Enhancing User Experience:

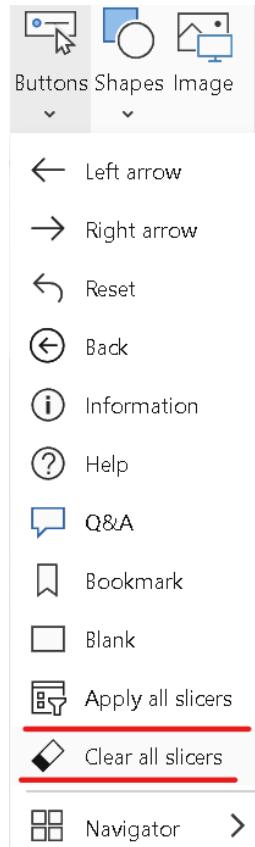
- You can also add additional interactive elements, such as hover effects. For example, you might want the button to change text or color when the user hovers over it, making it clear that the button is actionable.



Using buttons for drillthrough in Power BI not only enhances the interactivity of your dashboards but also makes the drillthrough feature more accessible and intuitive for users. By providing clear visual cues and customizing the button's appearance based on its state, you ensure that users can easily understand and utilize this powerful feature. In the next lecture, we'll look at additional ways to make your Power BI dashboards even more user-friendly and visually appealing.

▼ Clear & Apply Slicers

In this section, we'll explore how to enhance your Power BI dashboards by using buttons to clear and apply slicer selections. These buttons can significantly improve the user experience by making it easier to manage slicer settings across your reports.



Why Use Clear and Apply Slicer Buttons?

Slicers are an essential feature in Power BI, allowing users to filter data interactively. However, managing multiple slicers across a report can sometimes become cumbersome, especially if you need to clear or apply selections frequently. The "Clear All Slicers" and "Apply All Slicers" buttons streamline this process, providing a more intuitive and efficient way to interact with slicers.

Setting Up the Clear All Slicers Button

The "Clear All Slicers" button allows users to reset all slicer selections on a page with a single click.

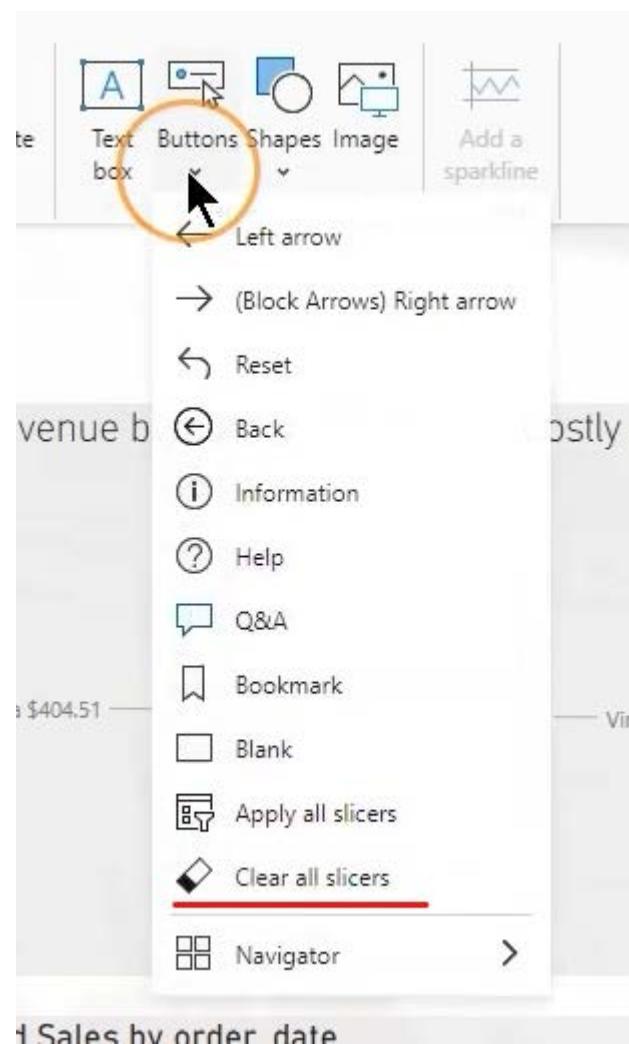
1. Add Slicers to Your Page:

- Begin by adding the slicers you need. For example, you might have slicers for **City** and **State**.

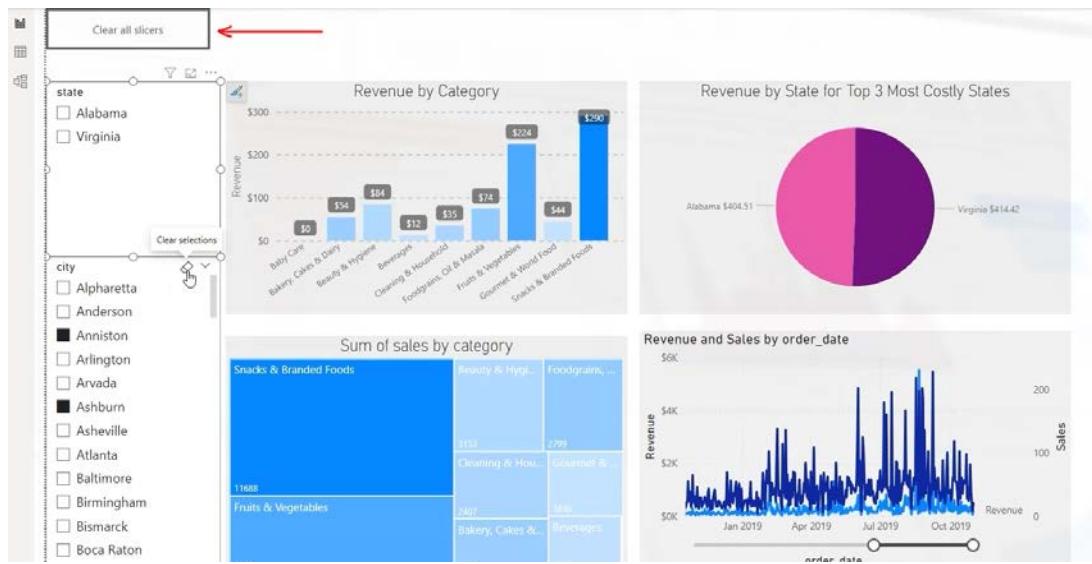


2. Insert the Clear All Slicers Button:

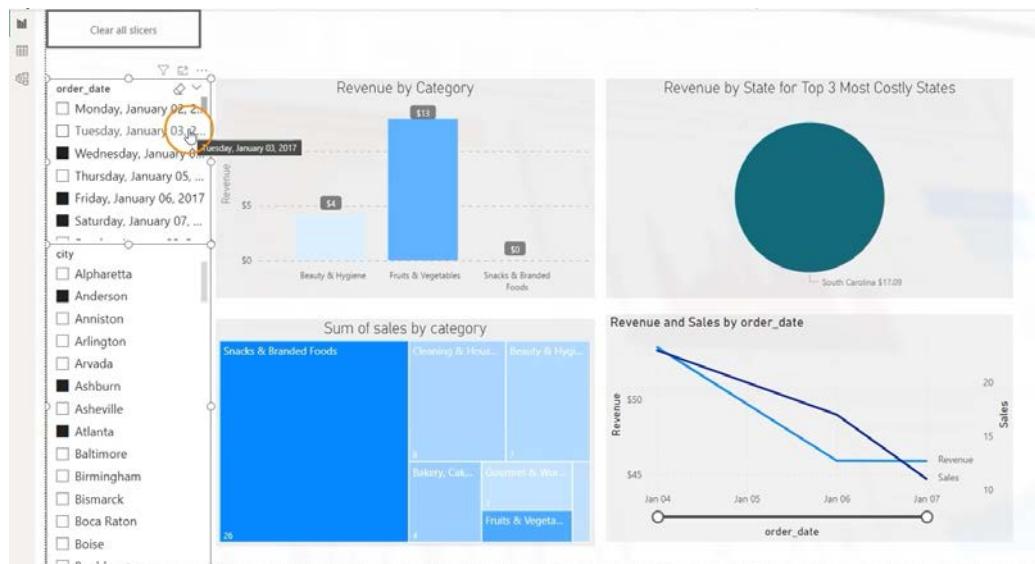
- Go to the **Insert** tab and select **Buttons**.
- Select **Clear all slicers**.



3. Using the Clear All Slicers Button:



- Once the button is set up, users can click it to clear all slicer selections on the page. In Power BI Desktop, you'll need to hold the **Ctrl** key while clicking the button. In the Power BI service, a regular click will suffice.
 - Before:



- After:



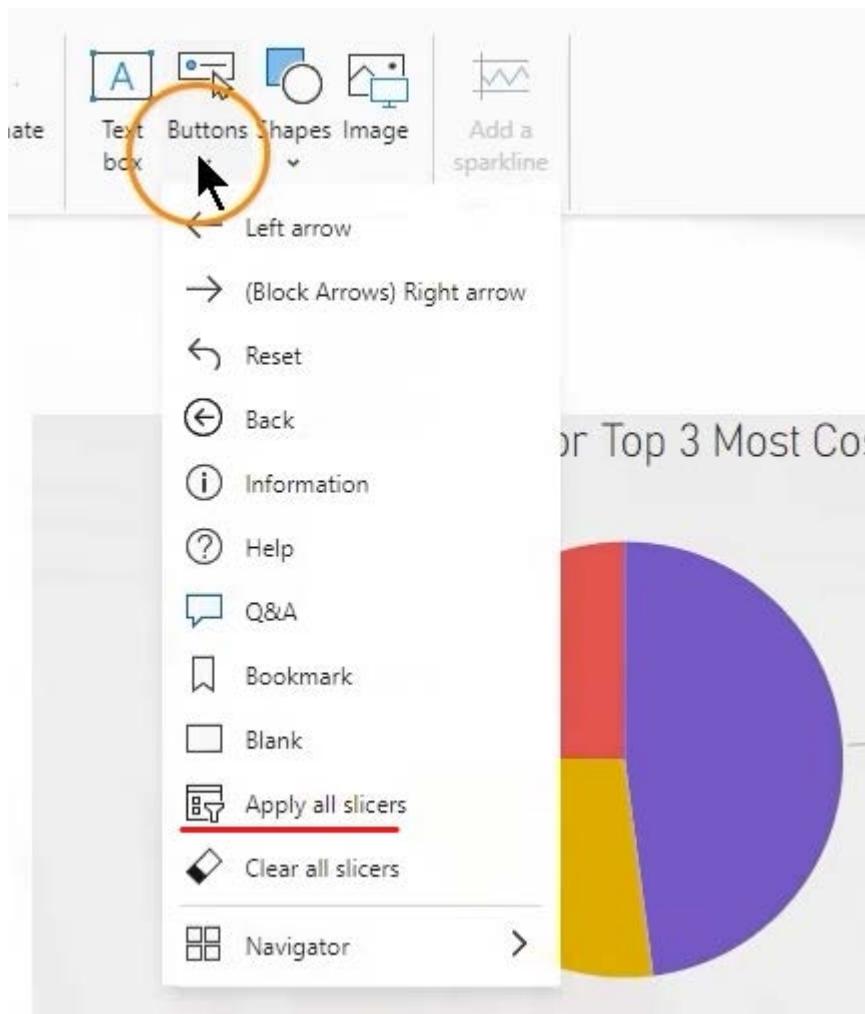
This feature is particularly useful when multiple slicers are in use, and you want to quickly reset all selections without manually deselecting each slicer.

Setting Up the Apply All Slicers Button

The "Apply All Slicers" button allows users to finalize slicer selections and apply them across the report.

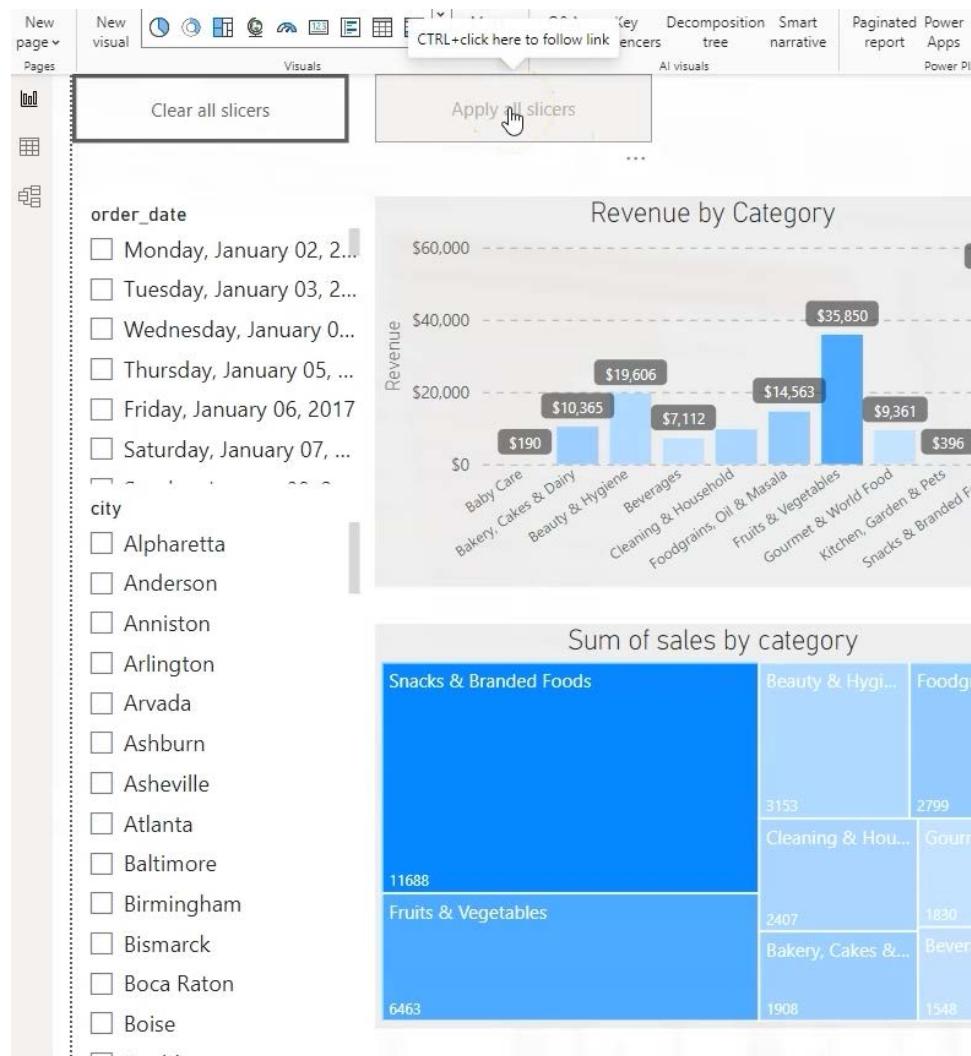
1. Insert the Apply All Slicers Button:

- Similar to the Clear All Slicers button, go to the **Insert** tab and choose **Buttons**.
- Select the **Apply all slicers**.

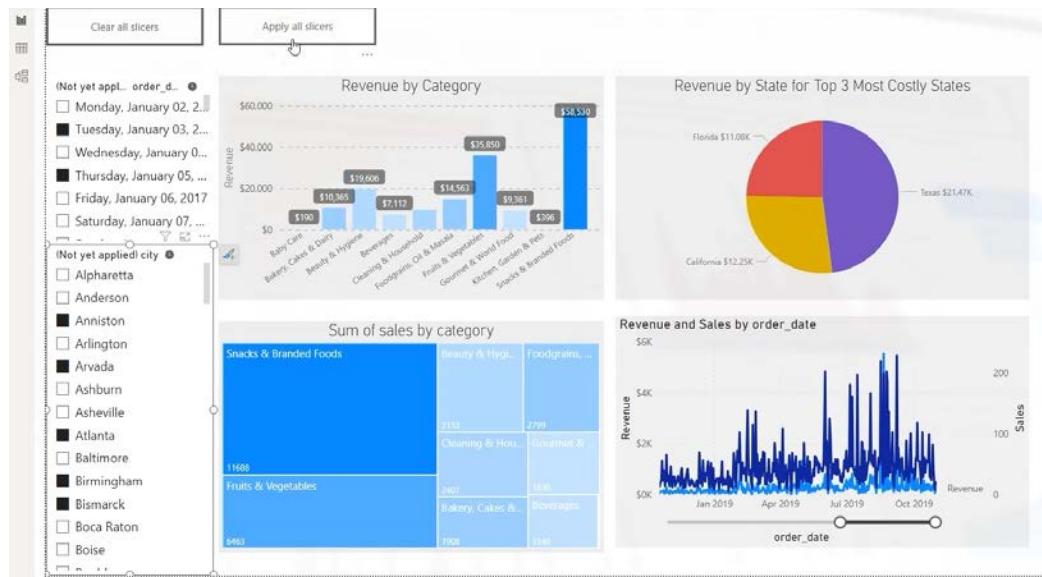


2. Understanding the Apply All Slicers Button:

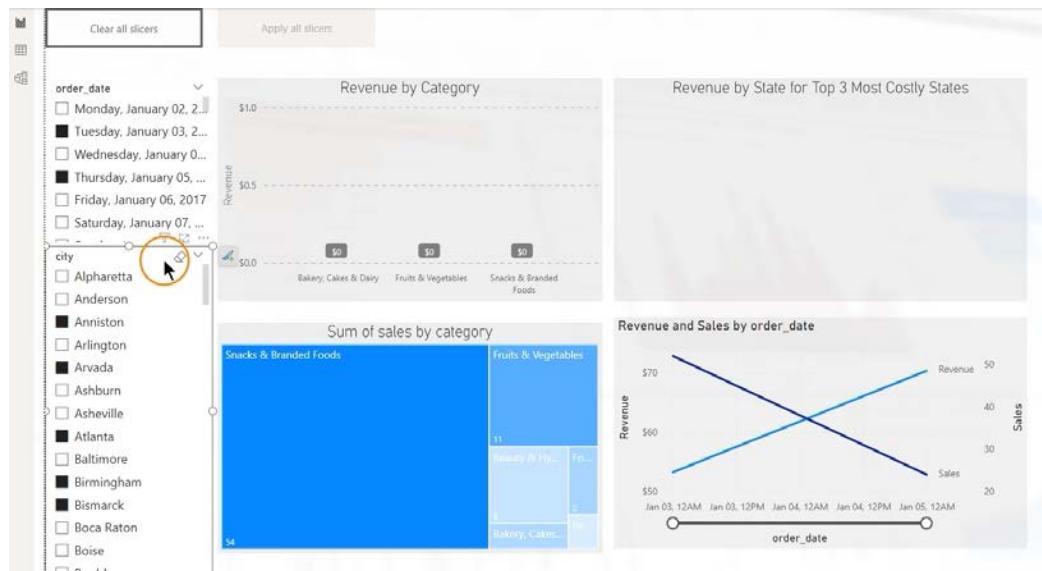
- After adding the button, you'll notice it is initially grayed out, indicating that no slicer selections have been made.



- Once a selection is made in any slicer, the button becomes active, but the changes are not applied until the button is clicked. This allows users to make multiple selections before committing the changes.



- After select “Apply all slicers”



3. Syncing Across Multiple Pages:

- The Apply All Slicers button is particularly useful when slicers are synced across multiple pages. It adds an extra layer of control, ensuring that selections are only applied when the user is ready.

Customizing Button Appearance and Behavior

To make the buttons more intuitive, you can customize their appearance and behavior based on their state (e.g., default, disabled, or hover states).

1. Modify Text and Appearance:

- In the **Format** pane, you can change the button's text, color, and style for each state. For example, when the Apply All Slicers button is disabled, you might display the text "Make Selection" to prompt the user.

2. State-Specific Customizations:

- Customize the button's appearance for different states, such as adding a shadow or changing the fill color when the button is disabled. This provides visual cues to users, enhancing the interactivity and clarity of your report.

3. Testing and Applying:

- Test the buttons by making and clearing selections in your slicers. Ensure that the visual feedback provided by the buttons (e.g., text changes or color changes) aligns with the user actions.

Using the Clear All Slicers and Apply All Slicers buttons in Power BI greatly enhances the usability of your dashboards. These buttons provide a more user-friendly way to manage slicer selections, especially in complex reports with multiple slicers. By customizing these buttons, you can create a more intuitive and visually appealing experience for your report users.

▼ Back & Reset Button

In this section, we'll explore the implementation and customization of the Back and Reset buttons within Power BI dashboards. These buttons are crucial for enhancing user navigation and interaction, ensuring a seamless and intuitive user experience.

Adding and Configuring Buttons

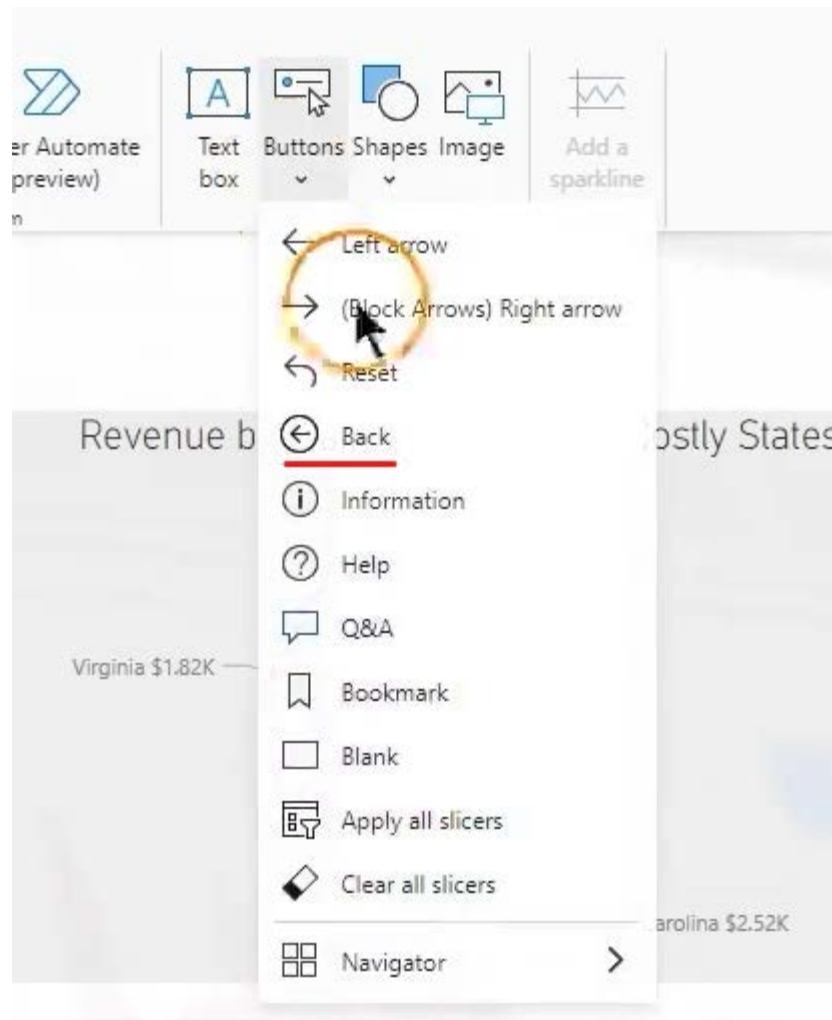
To add a new button, navigate to the **Insert** tab and select **Buttons**. Here, you can choose from various predefined buttons, including the **Back** and **Reset** buttons.

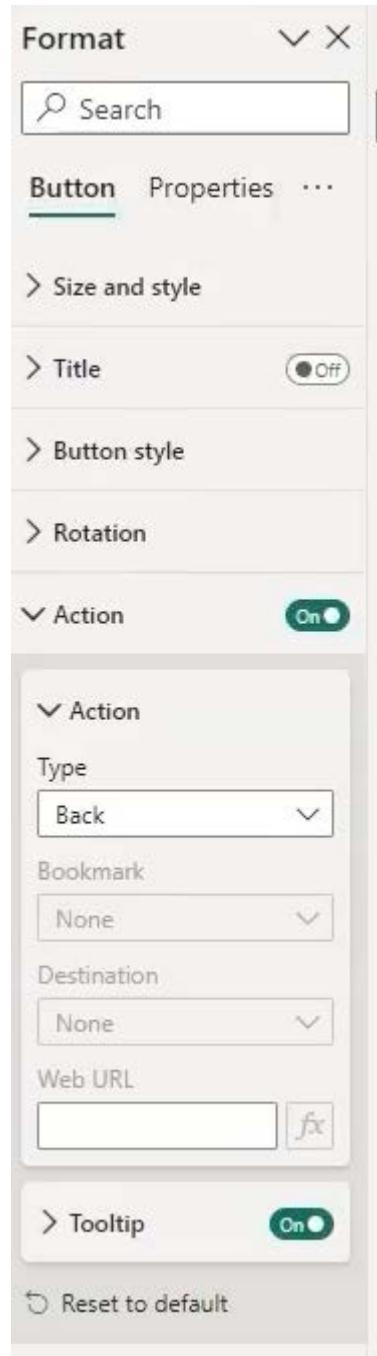
1. Back Button:

- **Functionality:** The Back button is pre-configured to return the user to the previously viewed page. This is particularly useful in multi-

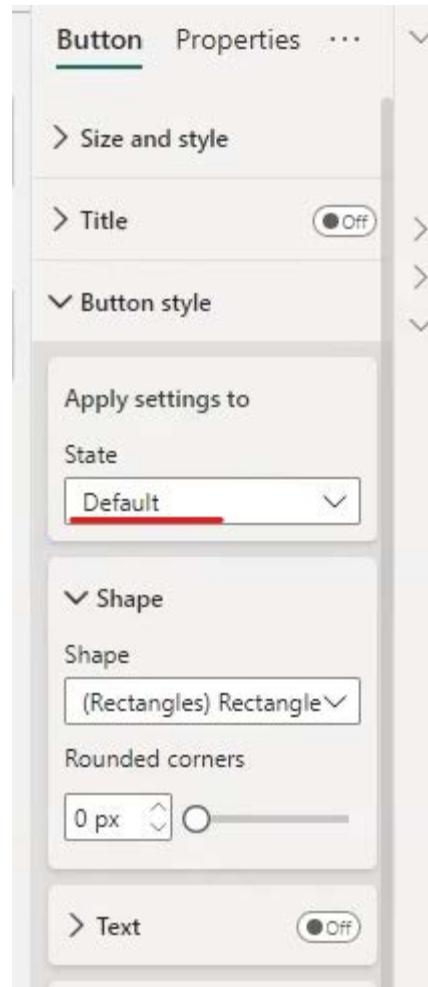
page reports, where users need a quick way to navigate back to the prior context.

- **Customization:**

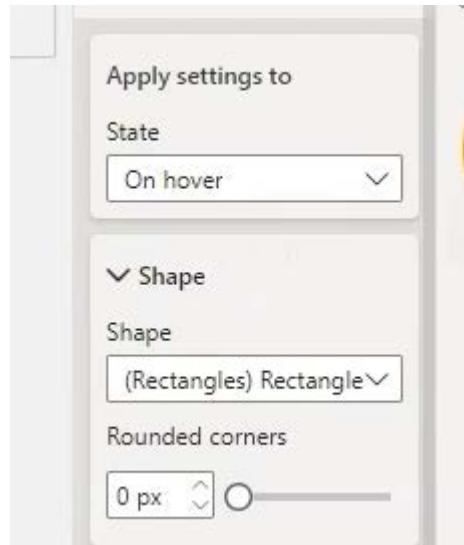




- You can modify the appearance of the Back button through the **Button Style** options. Customize different states such as **Default**, **Hover**, and **Pressed** to enhance visual feedback.

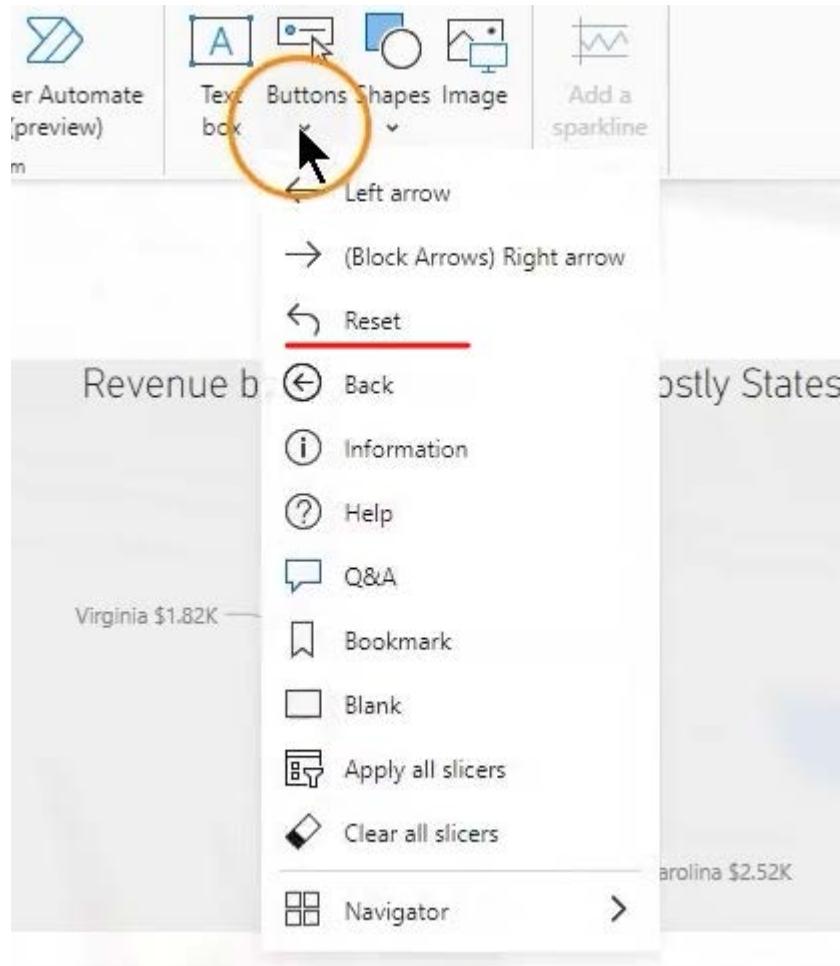


- For instance, in the **Default** state, you might want the button to appear slightly transparent to avoid distraction. Upon hovering, the button could become more opaque, indicating it is active and ready for interaction.



- Additionally, you can add a tooltip to the button to provide more context. For example, a tooltip that says "Click here to go to the previous page" can be added under the **Tooltip** section.

2. Reset Button:



- **Functionality:** The Reset button is used to clear or reset the report to its default state. Unlike the Back button, this button does not come pre-configured with an action, so you'll need to assign an appropriate action.
- **Customization:**
 - You can assign the Reset button a specific action using the **Action** settings. For instance, you might link it to a bookmark that resets all filters and selections in the report.



- Like the Back button, the Reset button can also be styled in different states to match the overall design of your report.

Enhancing User Experience

By effectively customizing these buttons, you can significantly improve the user experience in your Power BI reports:

- Visual Feedback:** Adjusting the button states ensures that users receive clear visual cues during interaction, making the report more intuitive.
- Tooltips:** Adding informative tooltips can guide users, especially when the button's function isn't immediately obvious, enhancing the report's usability.

In summary, the Back and Reset buttons are powerful tools for improving navigation and user interaction in Power BI dashboards. By customizing these buttons' appearance and behavior, you can create a more polished and user-friendly report. In the next section, we will delve into the use of bookmarks to further enhance interactivity and user control.

▼ Bookmarks

Bookmarks in Power BI are a powerful feature that allow you to capture and store specific states of your report page, including filter settings, visual configurations, and even the visibility of specific visuals. This feature is invaluable for creating interactive reports that can adapt to user needs with a single click.

What Are Bookmarks?

A bookmark in Power BI captures the state of a report page at a given moment. This includes:

- **Filter Settings:** The current selections in slicers and other filter controls.
- **Visual States:** The visibility, formatting, and configuration of visuals. For instance, a bookmark can store whether a pie chart is visible or hidden, or whether a table is sorted in a particular way.
- **Page Navigation:** The ability to save the user's current page and return to it later.

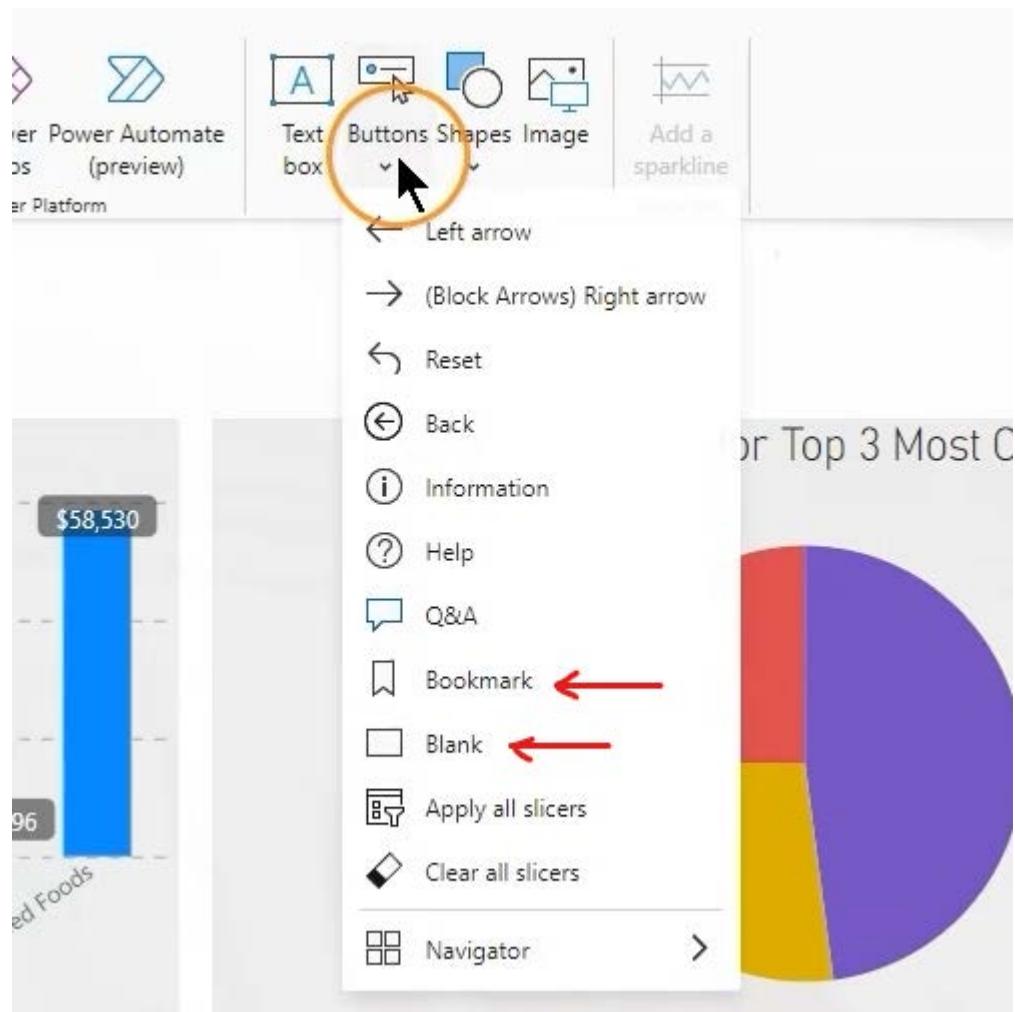
Creating and Using Bookmarks

In Power BI, a bookmark captures the current state of your report page, including how data is filtered. For example, slicer selections or a specific drill-down mode could be active. You can also adjust how a visual is formatted or whether it is displayed at all. This allows you to show different visuals or filtering options based on the bookmark selected.

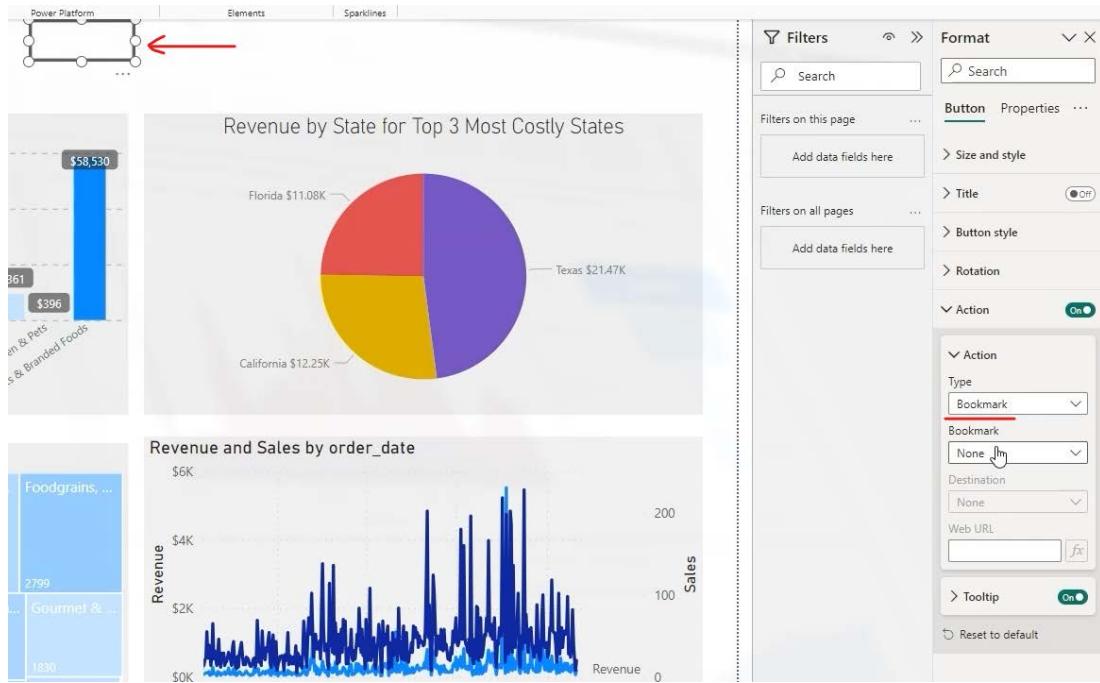
Creating a Bookmark

To create a bookmark, follow these steps:

1. Add a **button** to your report. You can use a pre-configured button like a "bookmark button" or a **blank button**, to which you can manually assign actions.

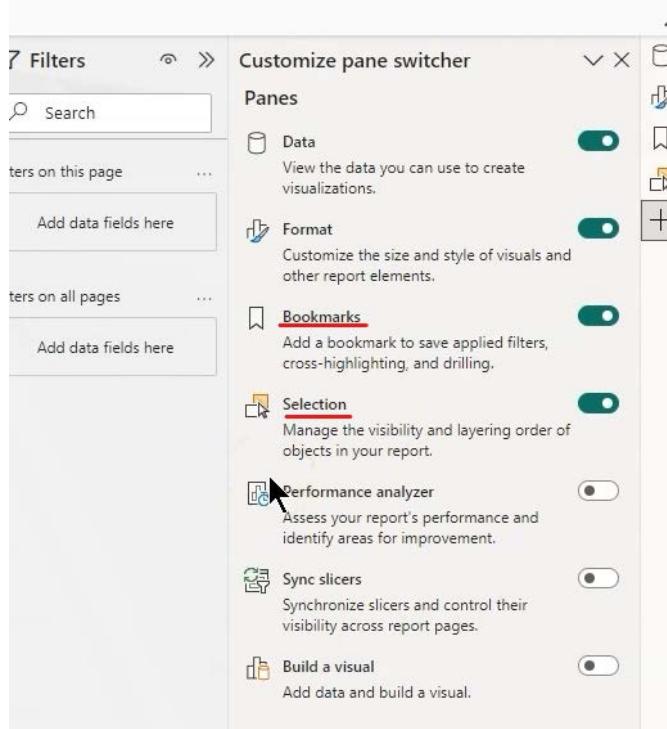


2. Once the button is added, go to **Actions** and set the action type to **Bookmark**.

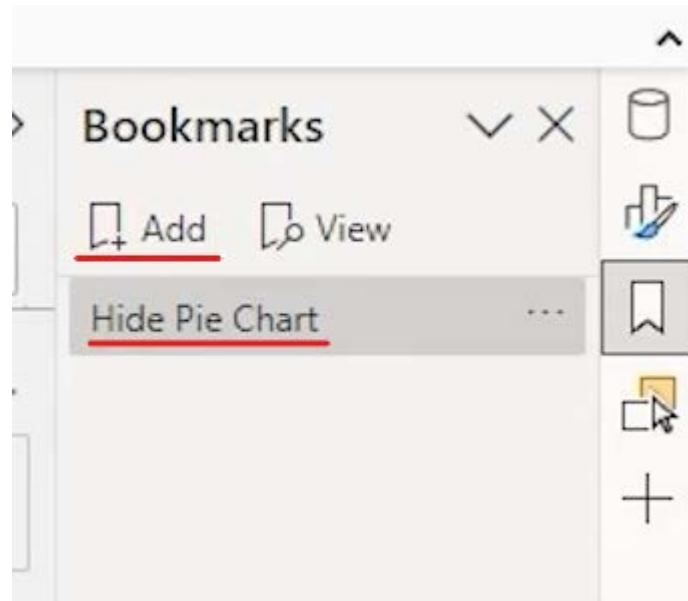


3. If you haven't created any bookmarks yet, do this by adding the **Bookmarks Pane** to your view. This can be done by going to the **View tab** and selecting **Bookmarks Pane**.

- Activate Bookmark and Selection.



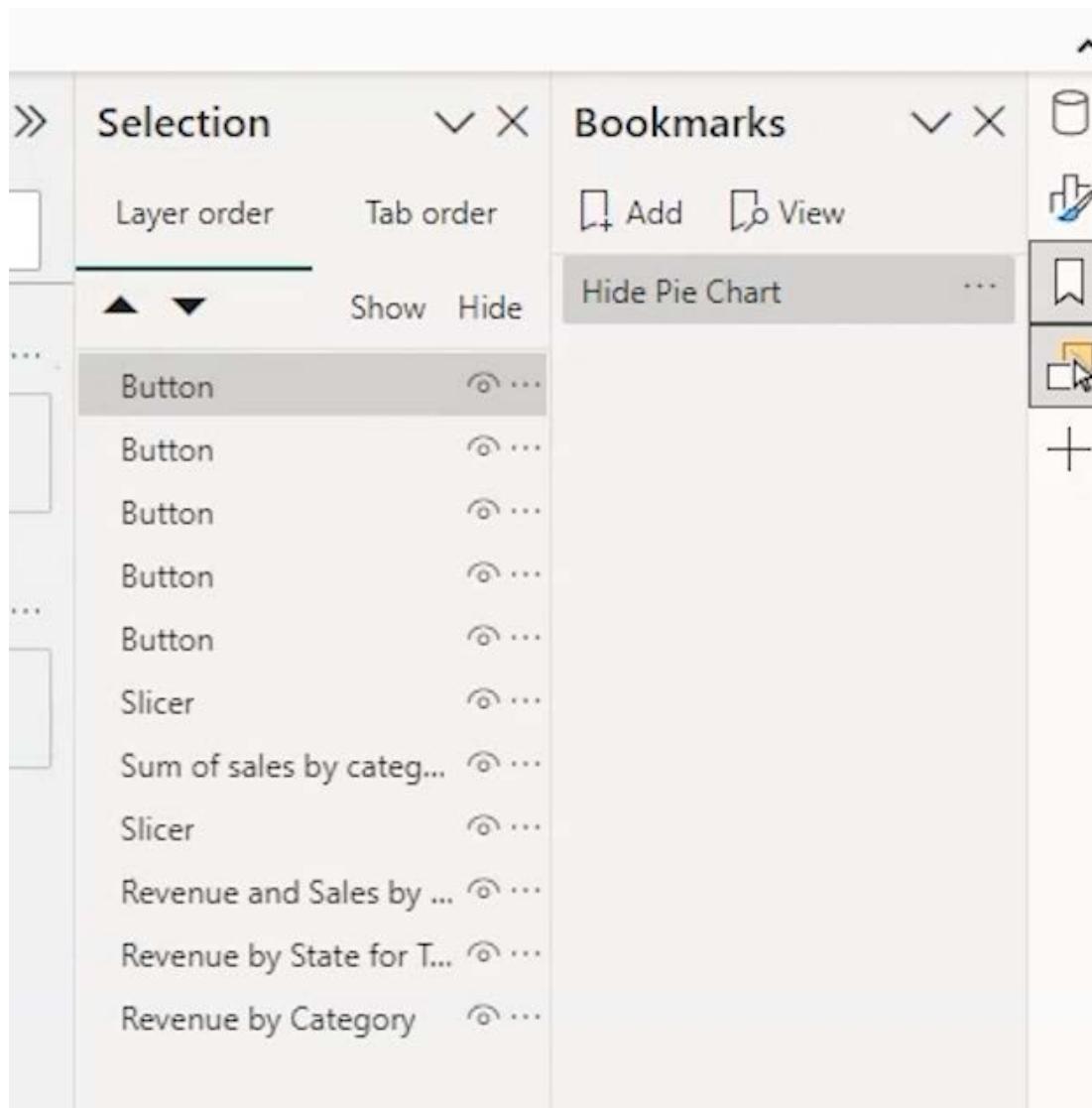
- Create bookmark.



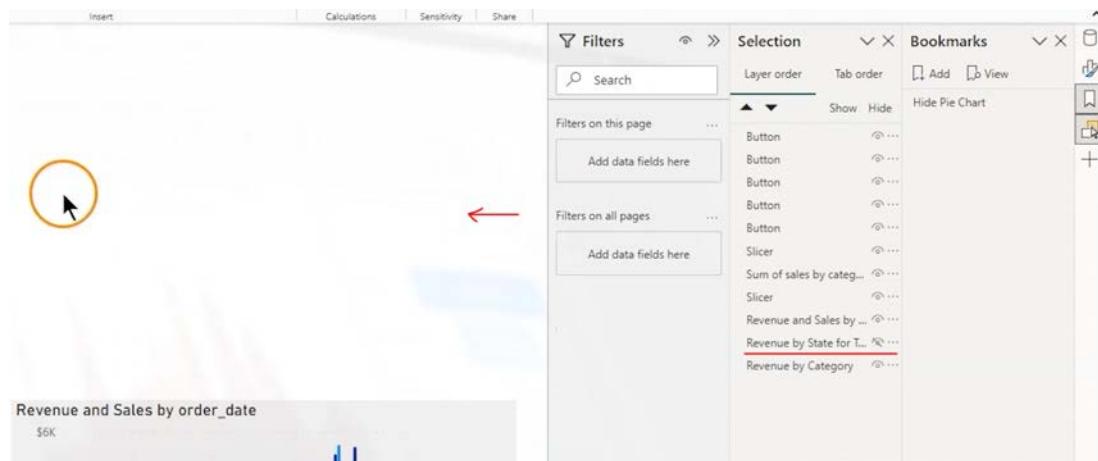
Managing Visuals and States

To manage which visuals are displayed:

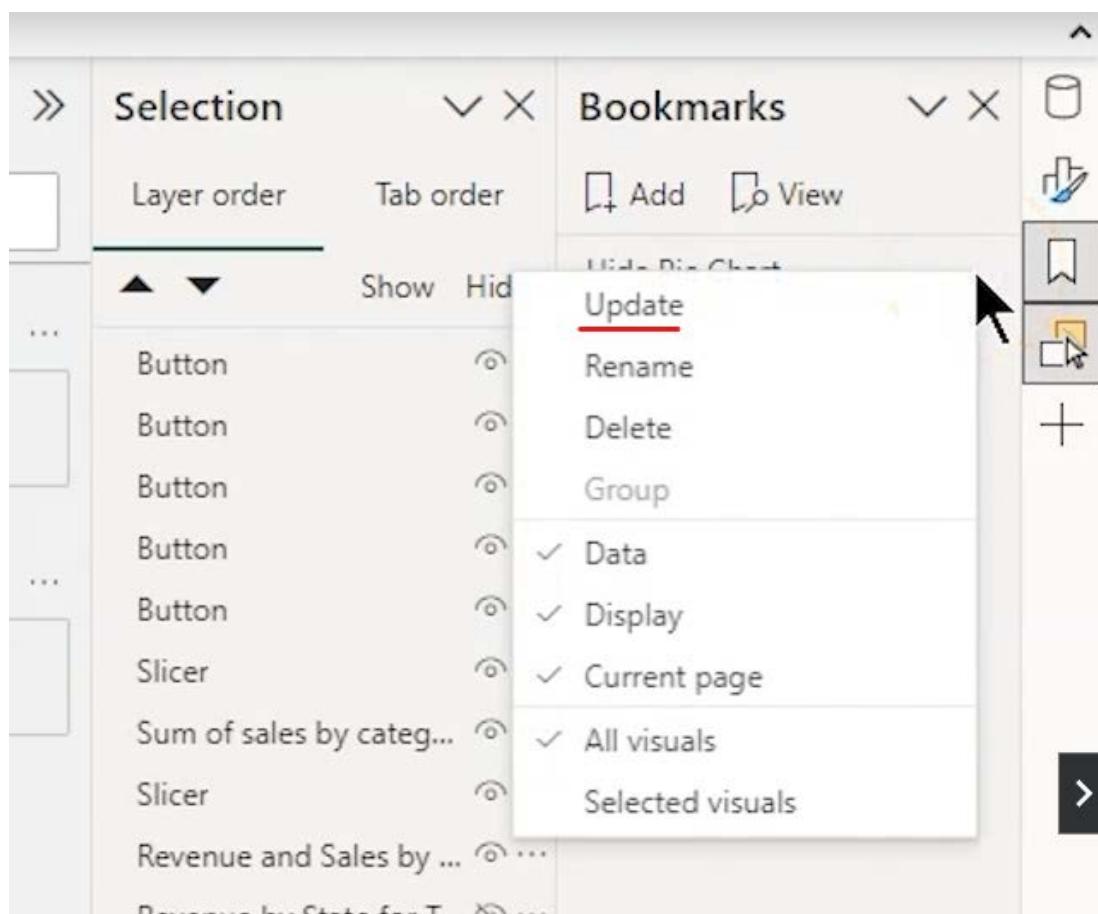
1. Open the **Selection Pane** to control the visibility of visuals.



2. Click on a visual (e.g., a pie chart) and use the eye icon to hide or show it.



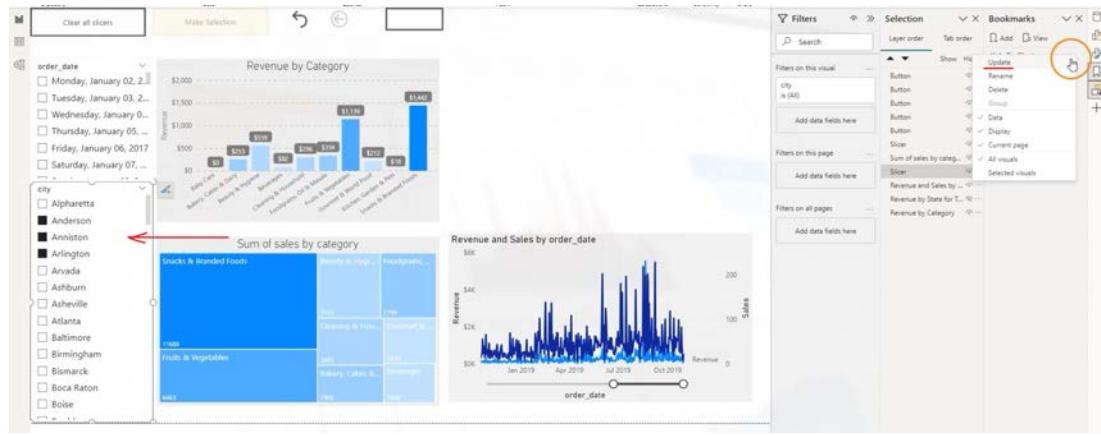
3. Save the current state (whether visuals are visible or hidden) by selecting the bookmark and clicking **Update**.



Filtering and Storing Selections

You can also store filtering options with bookmarks:

1. Apply slicers and filters, and then update the bookmark.



1. Data, Display, and Page Options:

- **Data:** By default, bookmarks capture the data state, including filters and slicer selections. If you want to exclude data settings from the bookmark, you can deselect the **Data** option in the bookmark settings.
- **Display:** This option controls whether the visibility of visuals is saved in the bookmark. If you want to toggle visibility of certain visuals, make sure this option is selected.
- **Current Page:** This option ensures that when a bookmark is applied, the report automatically navigates to the page where the bookmark was created.

Switching Between States

To switch between different report states, click the button associated with a bookmark. The page will adjust to the saved state, including which visuals are visible, the current page, and any filters or selections.

Controlling Visuals and Filters

There are options to control whether all visuals or only selected visuals are affected by the bookmark. You can choose to:

- Apply changes to **all visuals** (default setting).

- Apply changes to **selected visuals only** (e.g., if you want one visual to change without affecting others).

Example of Updating Visual States

Let's say you have several slicers applied. You can update the bookmark to store these slicer selections. If you later want only a specific visual to reflect these changes, use the **Selected Visuals Only** option when updating the bookmark.

Summary of Bookmark Usage

Bookmarks are useful for:

- Hiding and showing visuals.
- Changing filters and data views.
- Customizing reports for different scenarios.

▼ Card Visual

In this section, we'll explore the Card Visual in Power BI, a simple yet powerful tool for highlighting a single, prominent value on your report. This visual is ideal for displaying key metrics such as total revenue, profit, or any other critical figure that you want to emphasize.

What is the Card Visual?

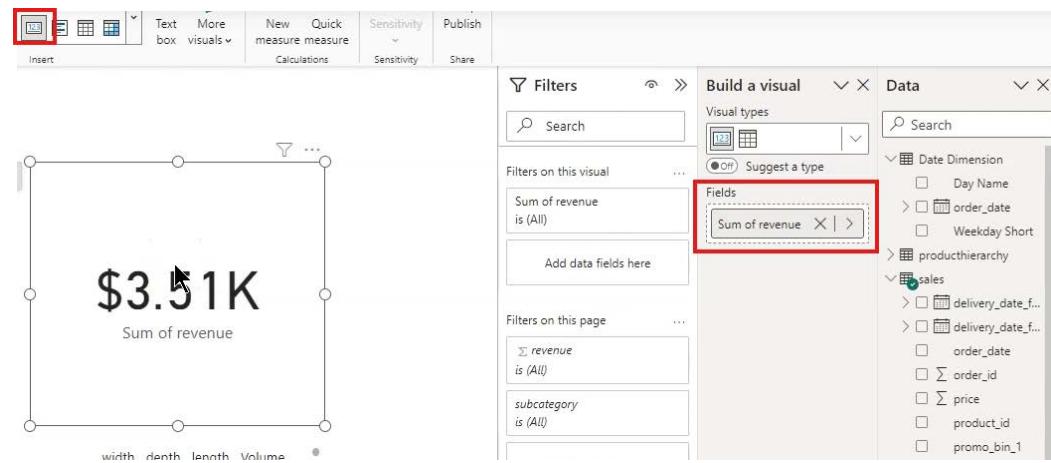
The Card Visual is designed to showcase a single data point clearly and prominently, making it an excellent choice for displaying summary metrics. It's particularly useful when you need to draw attention to a specific value in your report.

Creating and Customizing a Card Visual

1. Inserting a Card Visual:

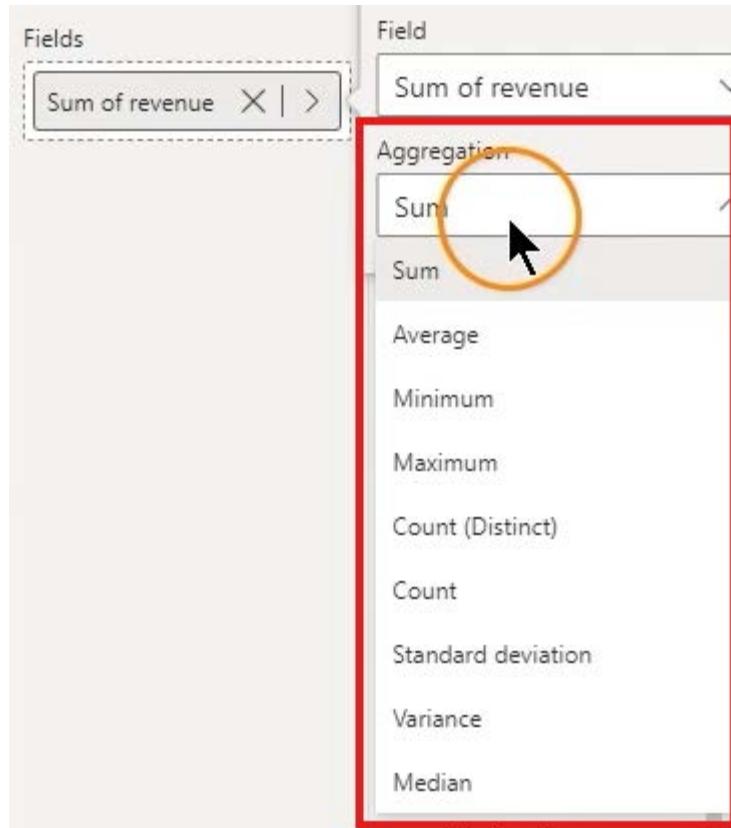
- To add a Card Visual, navigate to the **Insert** tab and select **Card** from the list of visuals. Once selected, you can place it anywhere on your report page by dragging it to the desired location.
- For example, if you want to display the total revenue on a detail page, simply drag and drop the **Revenue** field from the **Data Pane**

into the Card Visual. Remember, the Card Visual can only display one field at a time.

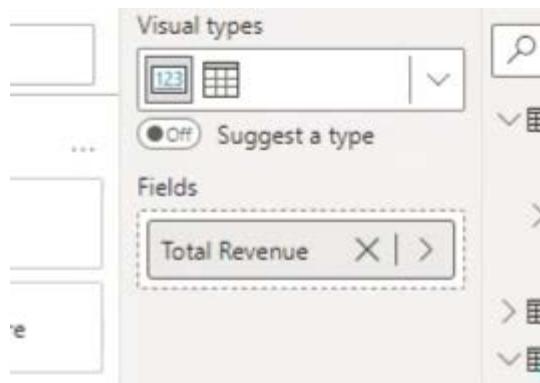


2. Field Aggregation:

- The value displayed in the Card Visual is aggregated based on the default aggregation method (usually **Sum**). If needed, you can change this aggregation method by selecting the field and adjusting the **Aggregation** settings.

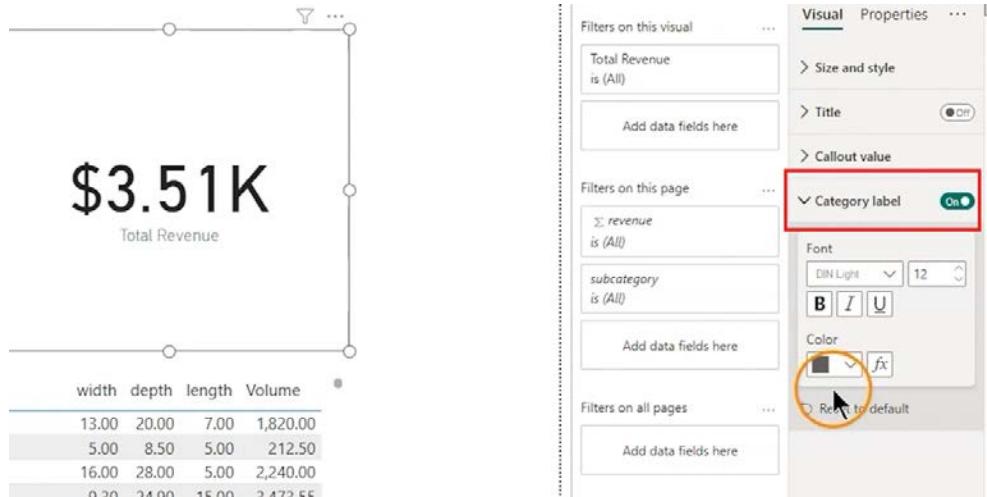


- If you prefer to display the value as "Total Revenue" instead of the field name, you can rename it by double-clicking on the field name in the **Build a Visual** pane.



3. Formatting the Card Visual:

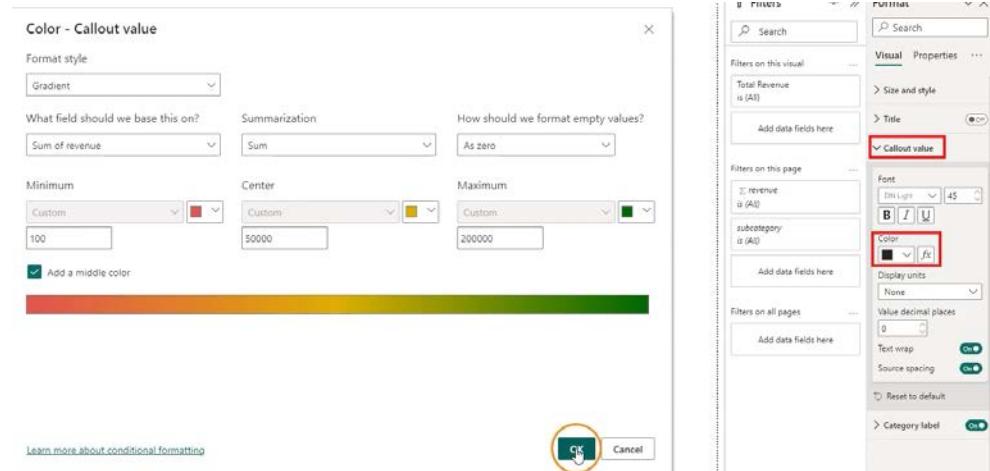
- The **Format Pane** provides extensive options to customize the appearance of the Card Visual. Here's what you can do:
 - **Category Label:** Modify the label, including font type, size, and color.



- **Callout Value:** Change the display settings of the actual value, such as font size, color, and decimal places. For instance, you can adjust the **Display Units** to "None" if you want to see the full number instead of abbreviated units (e.g., 3,500 instead of 3.5K).



- **Conditional Formatting:** Apply color-based visual cues to the value. For example, you can set the card to display green for values above a certain threshold and red for lower values.



\$3,507

Total Revenue

The Card Visual is a straightforward but effective way to highlight important metrics in your Power BI reports. By carefully formatting and configuring the Card Visual, you can ensure that key figures stand out, providing immediate insights to your audience.

▼ Multi-Row Card

In this section, we'll explore the **Multi-Row Card** visual in Power BI, which is ideal for displaying multiple values in a compact, space-efficient manner. Unlike the standard Card Visual, which displays only one value, the Multi-Row Card allows you to showcase several metrics simultaneously, making it perfect for summarizing key data points side by side.

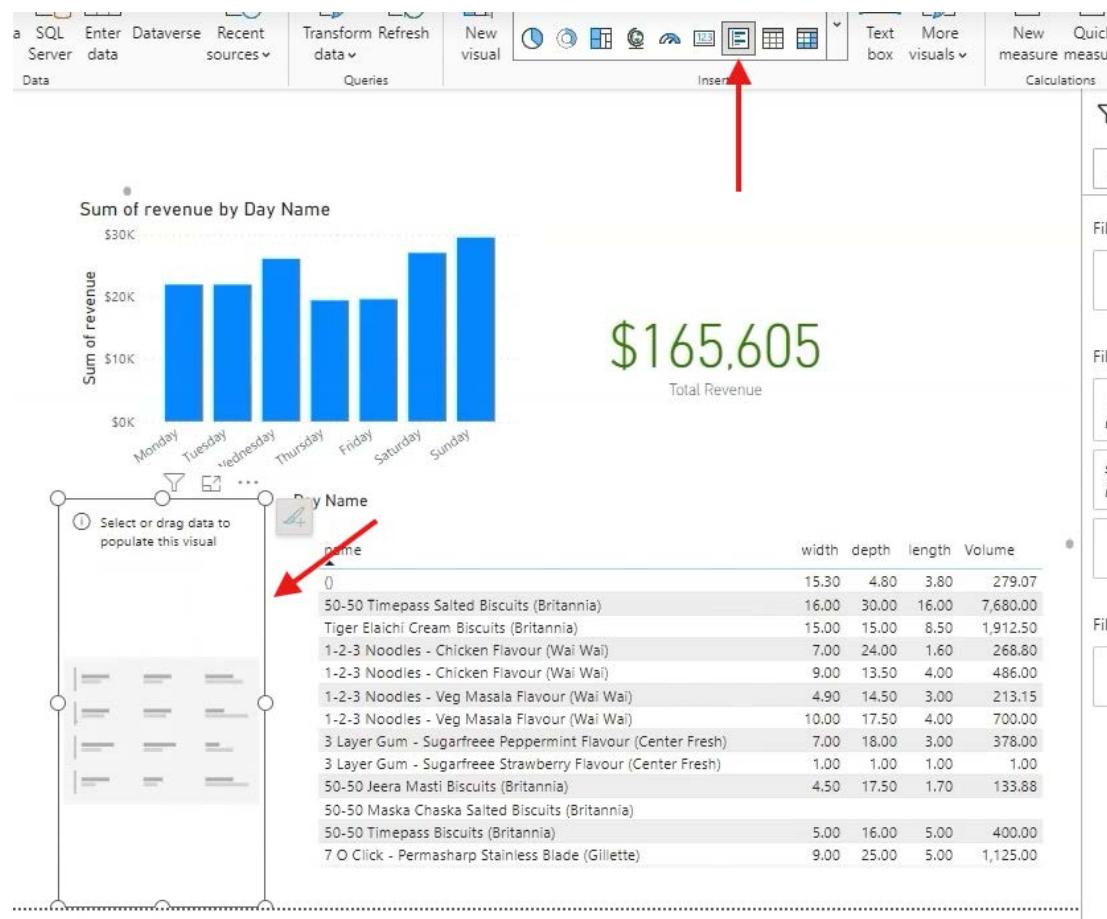
What is the Multi-Row Card Visual?

The Multi-Row Card Visual is designed to display multiple fields or metrics in a grid-like format. It is particularly useful when you need to present

several related values, such as revenue, sales, and profit, in a clear and organized way within a limited space.

Creating and Customizing a Multi-Row Card Visual

1. Inserting a Multi-Row Card Visual:



The screenshot shows the Power BI interface with the Insert ribbon tab selected. A red arrow points to the Multi-Row Card icon in the ribbon bar. Below the ribbon, there is a chart titled "Sum of revenue by Day Name" and a large green card displaying "\$165,605 Total Revenue". On the left, there is a placeholder for a visual with the text "Select or drag data to populate this visual". A red arrow also points to the placeholder area. To the right, there is a table with columns: Day Name, width, depth, length, and Volume. The table lists various items like biscuits and noodles with their dimensions and volume.

Day Name	width	depth	length	Volume
0	15.30	4.80	3.80	279.07
50-50 Timepass Salted Biscuits (Britannia)	16.00	30.00	16.00	7,680.00
Tiger Elaichi Cream Biscuits (Britannia)	15.00	15.00	8.50	1,912.50
1-2-3 Noodles - Chicken Flavour (Wai Wai)	7.00	24.00	1.60	268.80
1-2-3 Noodles - Chicken Flavour (Wai Wai)	9.00	13.50	4.00	486.00
1-2-3 Noodles - Veg Masala Flavour (Wai Wai)	4.90	14.50	3.00	213.15
1-2-3 Noodles - Veg Masala Flavour (Wai Wai)	10.00	17.50	4.00	700.00
3 Layer Gum - Sugarfree Peppermint Flavour (Center Fresh)	7.00	18.00	3.00	378.00
3 Layer Gum - Sugarfree Strawberry Flavour (Center Fresh)	1.00	1.00	1.00	1.00
50-50 Jeera Masti Biscuits (Britannia)	4.50	17.50	1.70	133.88
50-50 Maska Chaska Salted Biscuits (Britannia)				
50-50 Timepass Biscuits (Britannia)	5.00	16.00	5.00	400.00
7 O Click - PermaSharp Stainless Blade (Gillette)	9.00	25.00	5.00	1,125.00

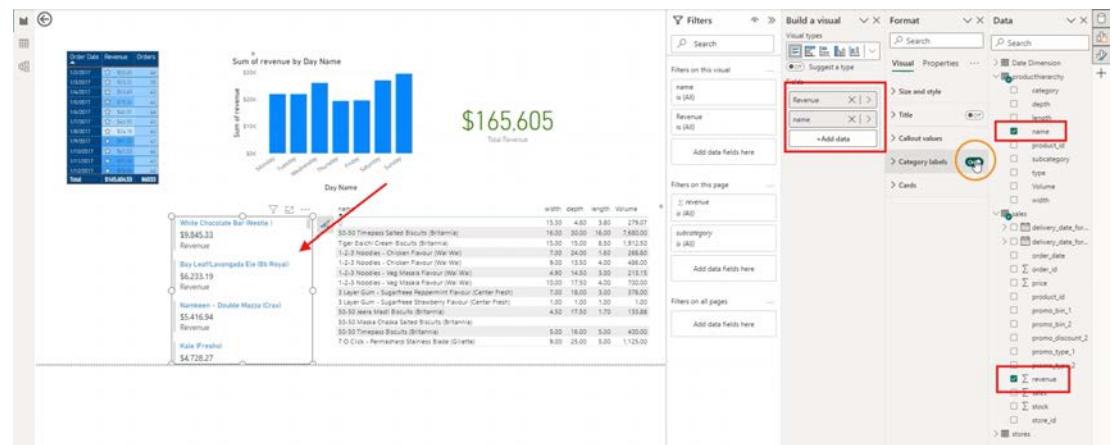
- To add a Multi-Row Card Visual, click on an empty space in your report and navigate to the **Visualizations Pane**. Select the **Multi-Row Card** icon.
- Place the visual in your desired location by dragging it around the report canvas.

2. Adding Fields to the Multi-Row Card:



- In the **Build a Visual** pane, you can add multiple fields to the Multi-Row Card. For instance, you can include **Revenue**, **Sales**, and **Profit** by dragging these fields from the **Data Pane** into the visual.
- Each field added will be displayed as a separate row within the Multi-Row Card. You can adjust the aggregation type for each field, such as **Sum**, **Average**, or **Count Distinct**.

3. Working with Categorical Fields:

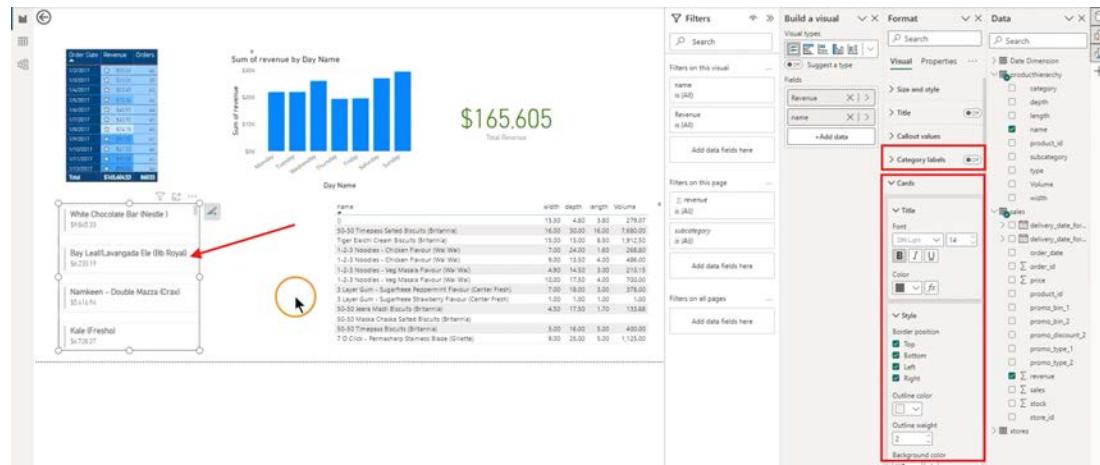


- If you add a categorical field, like **Product Name**, the Multi-Row Card will behave differently. Instead of summarizing the values, it

will display the metrics for each individual category (e.g., Revenue, Sales, and Profit for each product).

- This is useful for breaking down key metrics by categories, but it's important to ensure that the visual isn't overcrowded, which can happen if there are too many categories.

4. Formatting the Multi-Row Card Visual:



- Category Labels:** You can choose to display or hide the category labels (e.g., "Revenue" or "Sales"). If you find the labels repetitive, you can turn them off in the **Format Pane** under **Category Labels**.
- Callout Value:** Customize the appearance of the values, including font type, size, and color. For instance, you might choose a specific font or apply conditional formatting to highlight certain values.
- Borders and Layout:** You can add borders around each value or the entire visual to improve readability. Adjust the padding, margin, and other layout settings to ensure that the data is presented clearly.

5. Applying Filters to the Multi-Row Card:



- To make the visual more focused, you can apply filters to display only the top N categories based on a certain metric (e.g., top 3 products by revenue).
- To do this, use the **Filters Pane** to add a **Top N** filter. Set the filter to display the top 3 categories by revenue, for example, by selecting **Revenue** as the measure and **Top 3** as the condition.

6. Adjusting Visual Layout:

- The layout and format of the Multi-Row Card can be customized further by resizing the visual. The number of columns and rows displayed adjusts dynamically based on the visual's size.
- This flexibility allows you to fit the visual neatly into your report layout without sacrificing readability.

The Multi-Row Card Visual in Power BI is a versatile tool for displaying multiple related values in a compact format. Whether summarizing key metrics or breaking down data by categories, this visual helps you present data efficiently without overwhelming the viewer. By customizing the layout and applying filters, you can ensure that your Multi-Row Card Visual delivers the insights you need in a clear, concise manner.

▼ project 8

[Project8.rar](#)

▼ Report Page Tooltip

In this section, we'll explore one of the most powerful features in Power BI: displaying an entire report page as a tooltip. This allows you to provide rich, visual insights directly within a tooltip, enhancing the interactivity and depth of your reports.

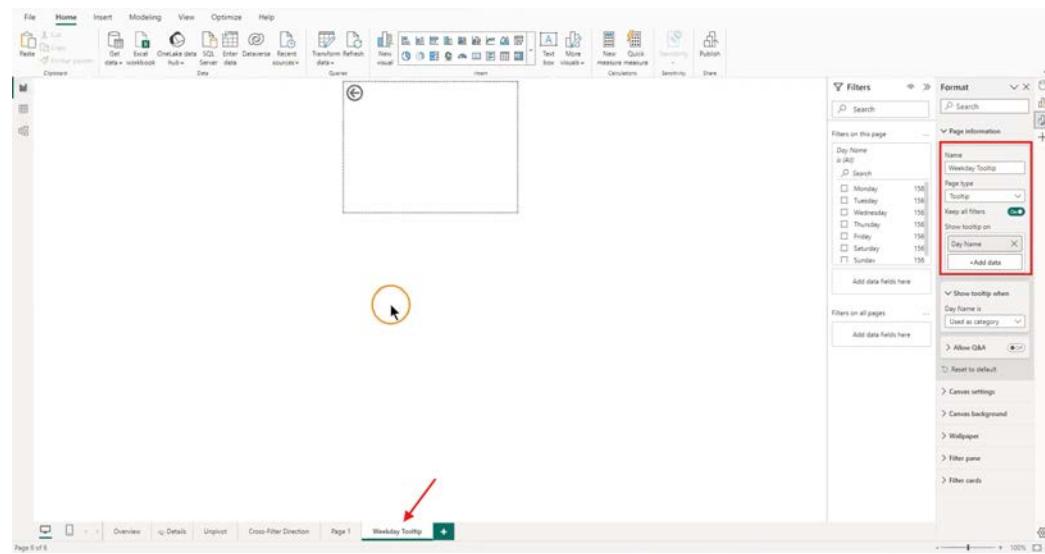
What is a Report Page Tooltip?

A tooltip in Power BI typically displays basic information when you hover over a data point, such as a simple text summary of a value. However, with the Report Page Tooltip feature, you can replace the standard text-based tooltip with a fully interactive report page. This allows you to display additional visual elements, such as charts or other visuals, directly within the tooltip, providing more detailed insights without cluttering the main report.

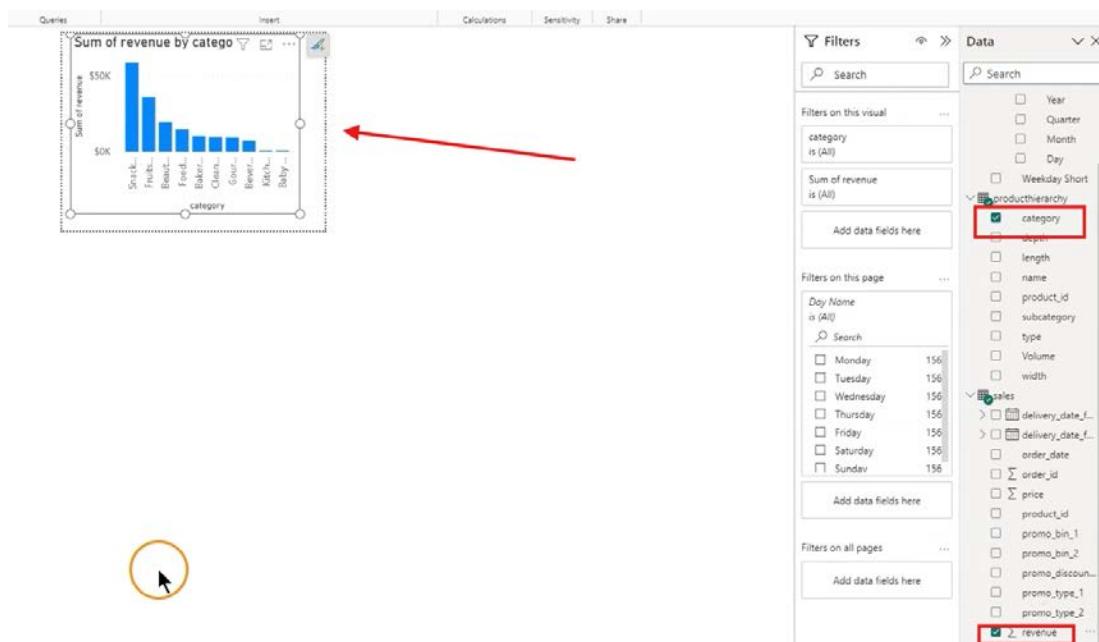
Setting Up a Report Page Tooltip

1. Creating a Tooltip Page:

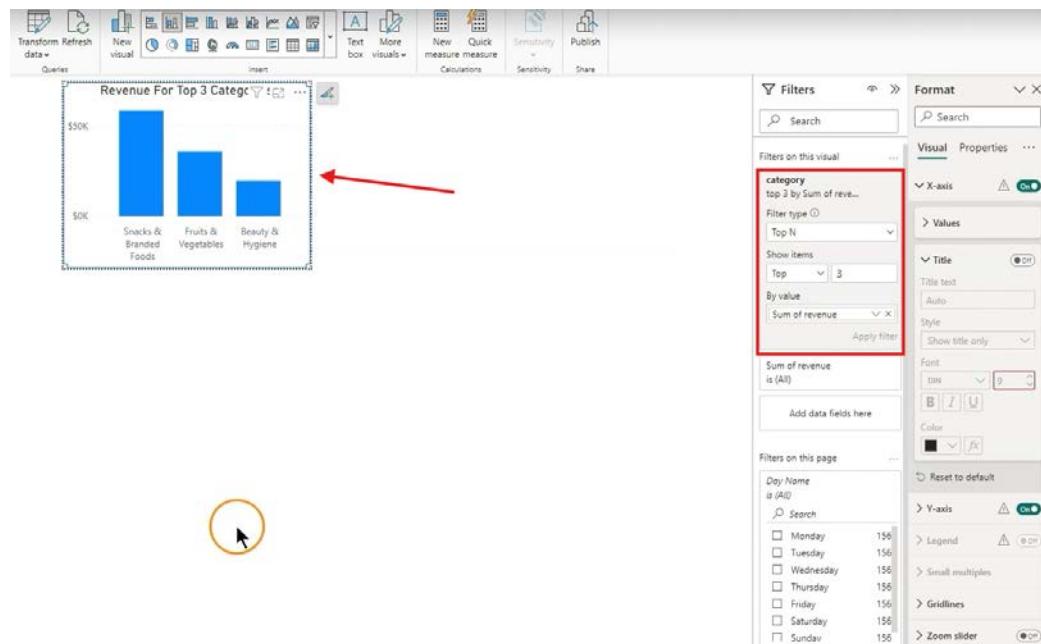
- Start by creating a new page in your report. Click the + icon at the bottom of the Power BI interface to add a new page.
- Rename the page to something descriptive, like "Weekday Tooltip," by double-clicking on the page name.
- With the page selected, go to the **Format Pane** and find the **Page Information** section. Change the **Page Type** to **Tooltip**. This will resize the canvas to a smaller tooltip size, making it easier to design for this specific use case.



2. Designing the Tooltip:



- On this new tooltip page, you can add any visualizations that you want to display as part of the tooltip. For example, you might create a stacked column chart that shows the top three product categories by revenue for each weekday.
- Drag and drop the relevant fields (e.g., **Category** and **Revenue**) into your visual. Format the chart as needed, such as by filtering it to display only the top three categories using the **Top N** filter.



- Adjust the visual's formatting to ensure it looks clean and concise. You might want to hide unnecessary titles or axis labels to make the most of the limited space.

3. Enable Tooltip on Visuals

- Navigate back to the **details page** where your visuals are.
- **Hover over the weekdays** to check if the tooltip is working.



- If not, ensure that the tooltip feature is enabled:

- Open the **Format Pane** for the visual and ensure that tooltips are enabled under properties.
- Select the tooltip page you created (e.g., "Weekday Tooltip").

4. Hide the Tooltip Page

- Once the tooltip is working correctly, **hide the page** by right-clicking the page tab and selecting "Hide Page."
 - This prevents the user from manually accessing the page, as it is intended only to be viewed as a tooltip.

The Report Page Tooltip feature in Power BI allows you to add rich, interactive visualizations directly into tooltips, greatly enhancing the user experience by providing deeper insights without overwhelming the main report page. By following these steps, you can create tooltips that not only convey more information but also maintain a clean and user-friendly report design.

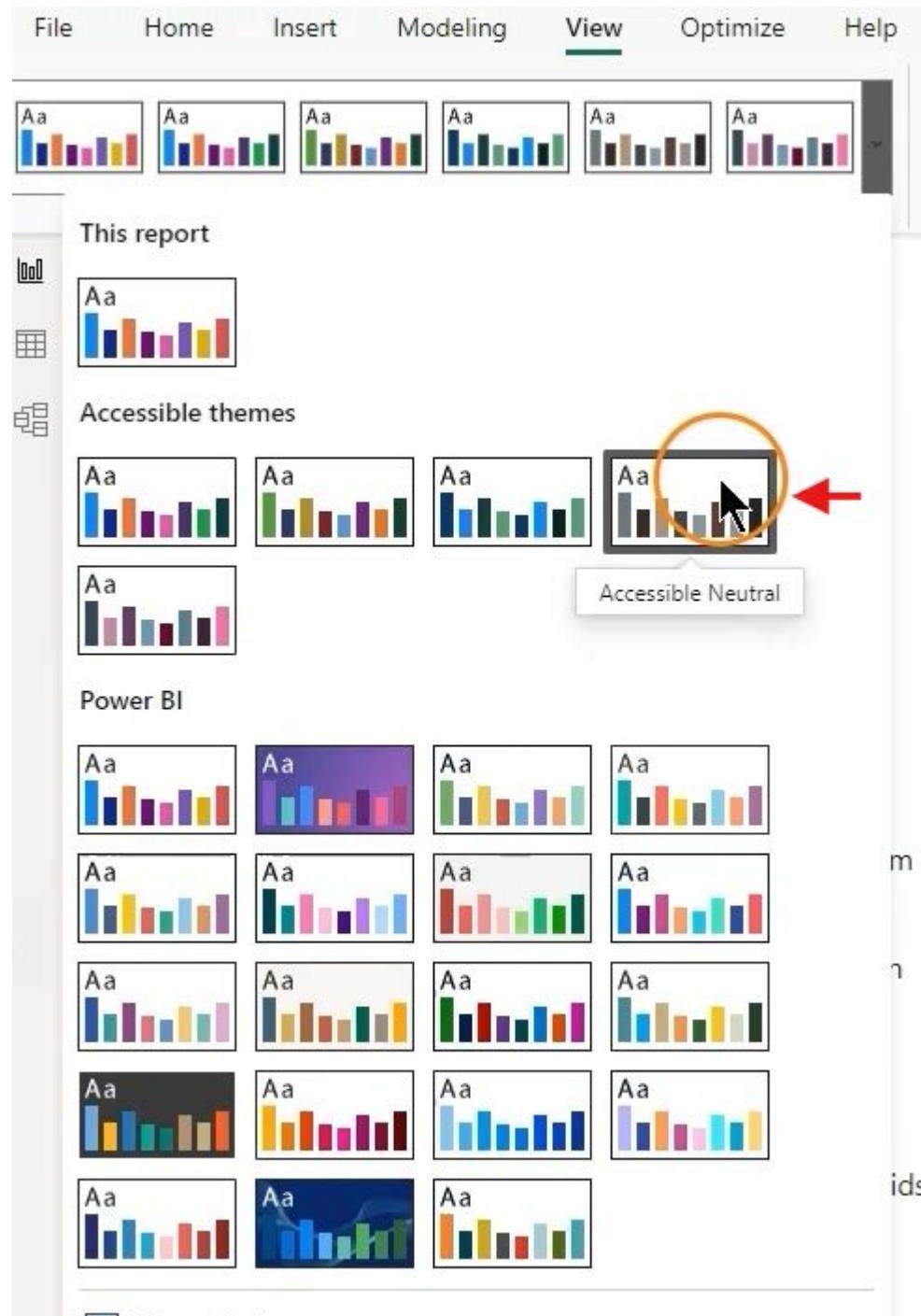
▼ Working with Themes

Step 1: Apply Out-of-the-Box Themes

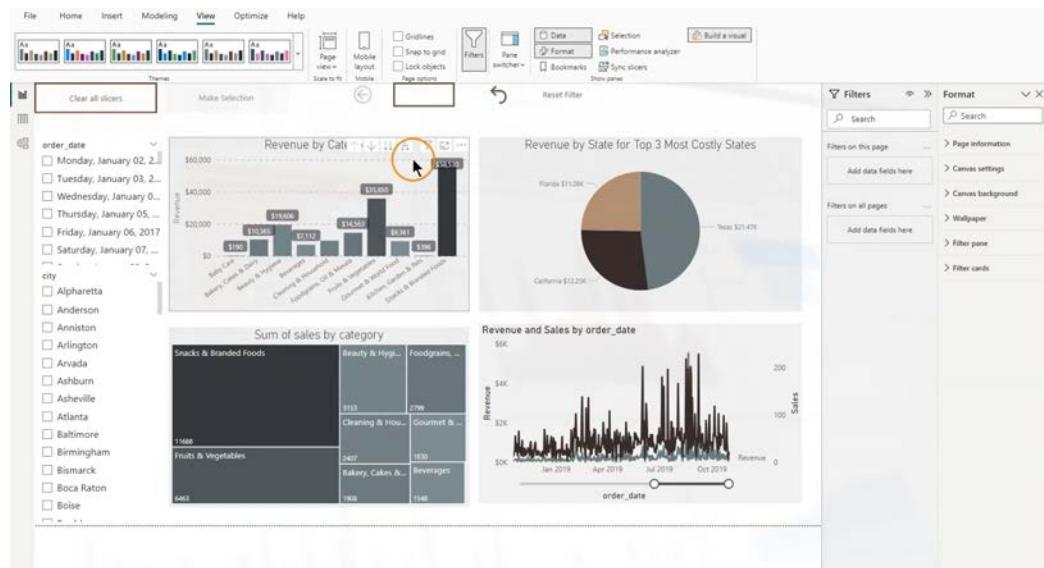
1. **Navigate to the View Ribbon** in Power BI.



2. Click on the **theme dropdown** and select from the available **out-of-the-box themes**, such as "Accessible" or "Neutral."



- Applying these themes will change the default values (e.g., colors, fonts) across all pages of your report.



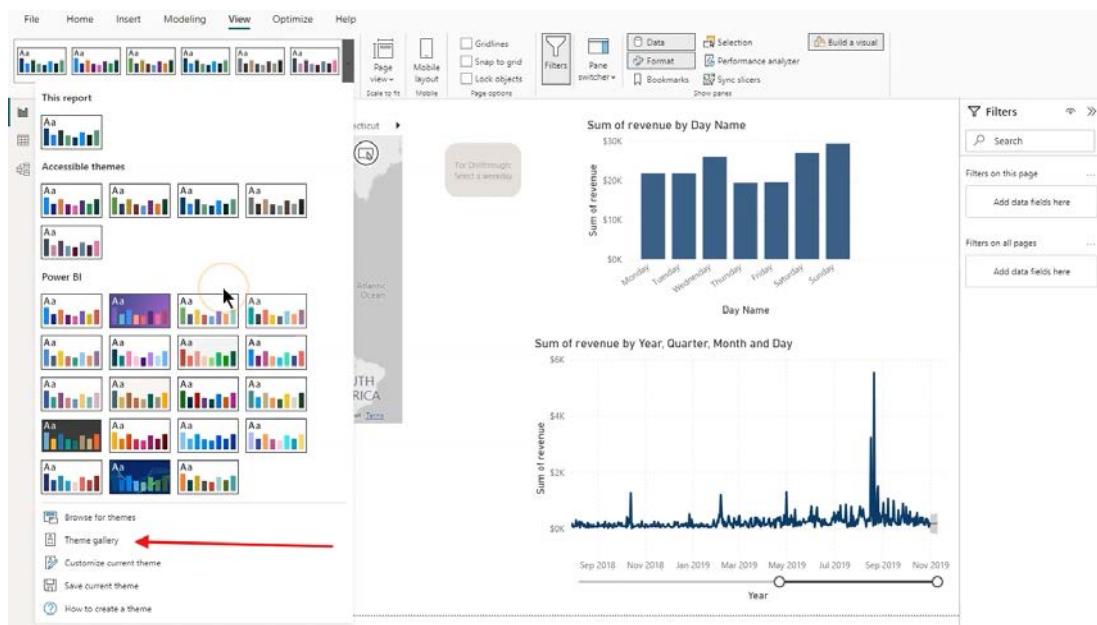
3. **Observe the changes:** If no specific customizations have been made (e.g., for titles or colors), the default settings from the selected theme will be applied.

Step 2: Customize Themes

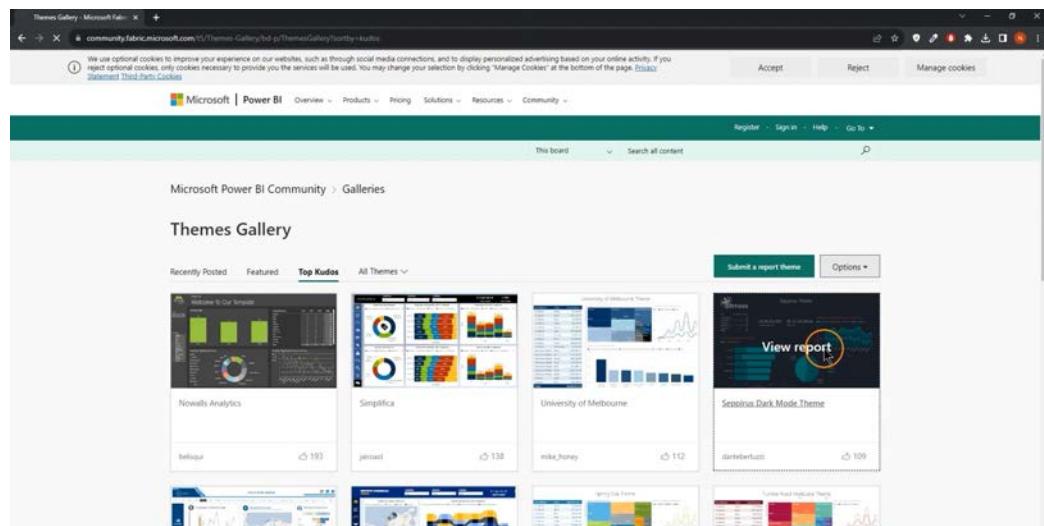
1. If the default theme does not fully meet your needs, you can **customize it**:
 - **Navigate to the theme dropdown** and select **Customize Current Theme**.
 - Modify various elements like:
 - **Theme Colors**: Adjust the default color palette to better match your corporate branding.
 - **Text Settings**: Customize text styles, such as title fonts and colors, to suit your report's design.
 - Apply your changes and see how they reflect on the report's visuals.
2. Example customizations:
 - Change the **title color** to a lighter red.
 - Modify the **first color in the color palette** to match your desired visual identity.

Step 3: Browse for More Themes

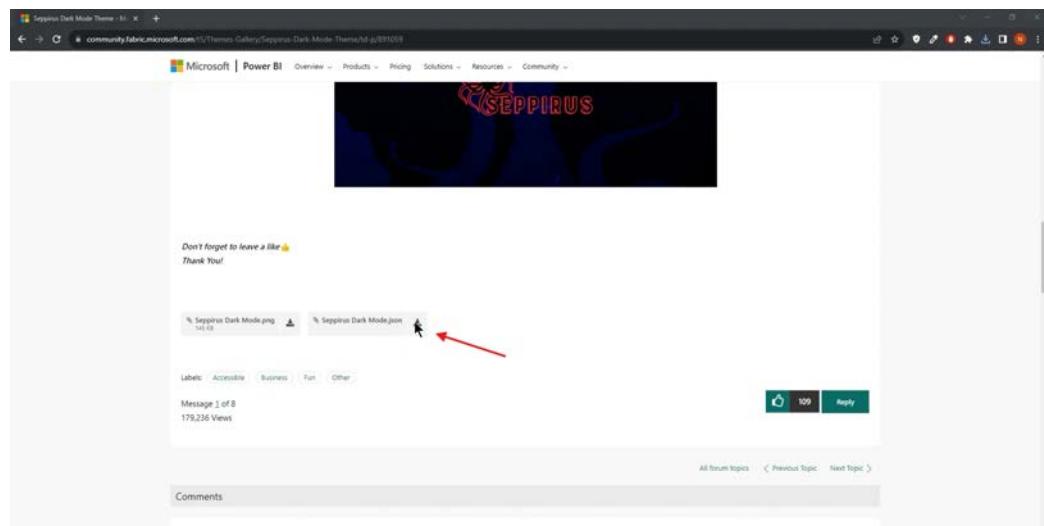
1. Power BI also allows you to **import themes from the Theme Gallery**:



- Click on the **Browse for Themes** option in the theme dropdown.
- Explore various themes in the **Power BI Theme Gallery**, where you can find themes based on user preferences or trending designs.



- Select a theme, preview its pages, and download the **JSON file** containing the theme settings.



2. Import the Theme:

- Go back to Power BI and select **Browse for Themes**.

The screenshot shows the Microsoft Power BI ribbon with the 'View' tab selected. A context menu is open over a bar chart, listing theme options:

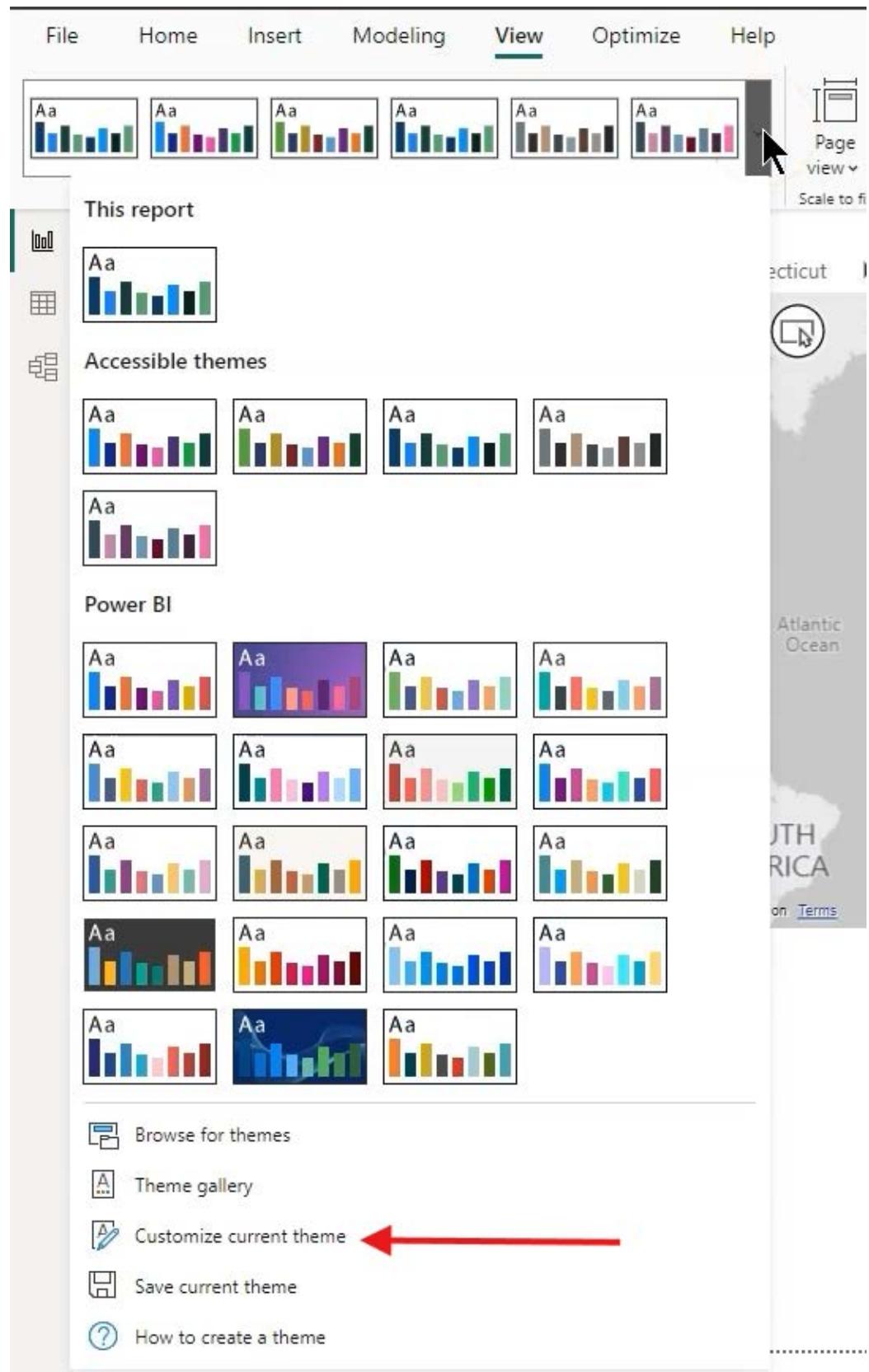
- Browse for themes
- Theme gallery
- Customize current theme
- Save current theme
- How to create a theme

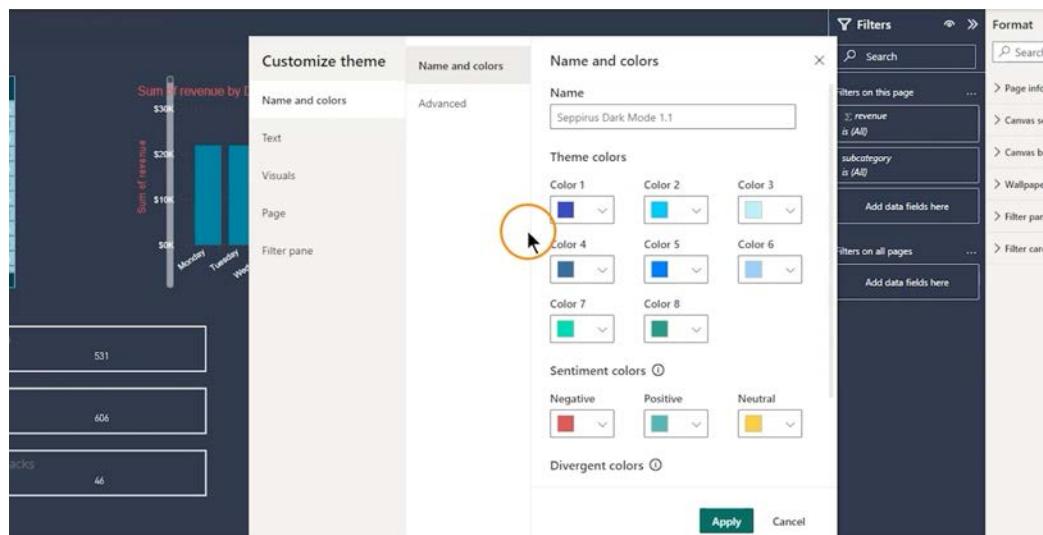
- Choose the downloaded theme file and apply it to your report. The theme will automatically adjust the design of your report.

Step 4: Create and Save Custom Themes

1. Modify a Theme:

- After customizing your theme (e.g., changing colors, text styles), you can save your changes as a new theme.





- Go to the theme dropdown and select **Save Current Theme**.
- Power BI will save the theme as a **JSON file** containing all the customizations (colors, fonts, backgrounds, etc.).

2. Reuse the Theme:

- To apply the saved theme to other reports, simply import the JSON file using the **Browse for Themes** option.
- This helps ensure consistent design across all reports developed by different users.

▼ 7- DAX Essentials

▼ Calculated Columns

In this section, we'll dive into the concept of **Calculated Columns** in Power BI, an essential aspect of DAX (Data Analysis Expressions). DAX is a powerful language used to perform advanced calculations and data manipulations within your data model. By understanding how to create and use calculated columns, you can enhance your data model and unlock deeper analytical insights.

What Are Calculated Columns?

A calculated column is a new column that you create within a table in your data model, based on a formula written in DAX. Unlike measures, which perform calculations dynamically based on user interactions with the report,

calculated columns are computed row by row at the time they are created and stored in the table. This makes them useful for adding static, row-level calculations that you can use like any other column in your data model.

Creating a Calculated Column

Let's walk through the process of creating a calculated column:

1. Navigating to the Table View:

The screenshot shows the Power BI Table View interface. At the top, there are tabs for File, Home, Help, Table tools, and Column tools. The Column tools tab is selected, showing settings for 'Name' (set to 'promo_discount_2'), 'Data type' (set to 'Text'), and 'Format' (set to 'Text'). Below these are options for 'Summarization' (set to 'Don't summarize') and 'Data category' (set to 'Uncategorized'). To the right, there are buttons for 'Sort by', 'Group by', 'Manage relationships', and 'New column'. The main area displays several tables: 'order_line' (with columns like 'order_line_id', 'product_id', 'store_id', 'order_date', 'sales', 'revenue', 'stock', 'price', 'promo_type_3', 'promo_line_id', 'promo_type_2', 'promo_line_p_1', 'promo_discount_2', 'delivery_date', 'order_id', 'date', 'product_id', 'product_name', 'product_type', 'price', 'revenue', 'sales', 'stock', 'store_id', 'stores', and 'date'). Other tables visible include 'product', 'store', and 'date'. Relationships are shown between 'order_line' and 'product', 'order_line' and 'store', and 'order_line' and 'date'. The 'promo_discount_2' column is highlighted in the table preview.

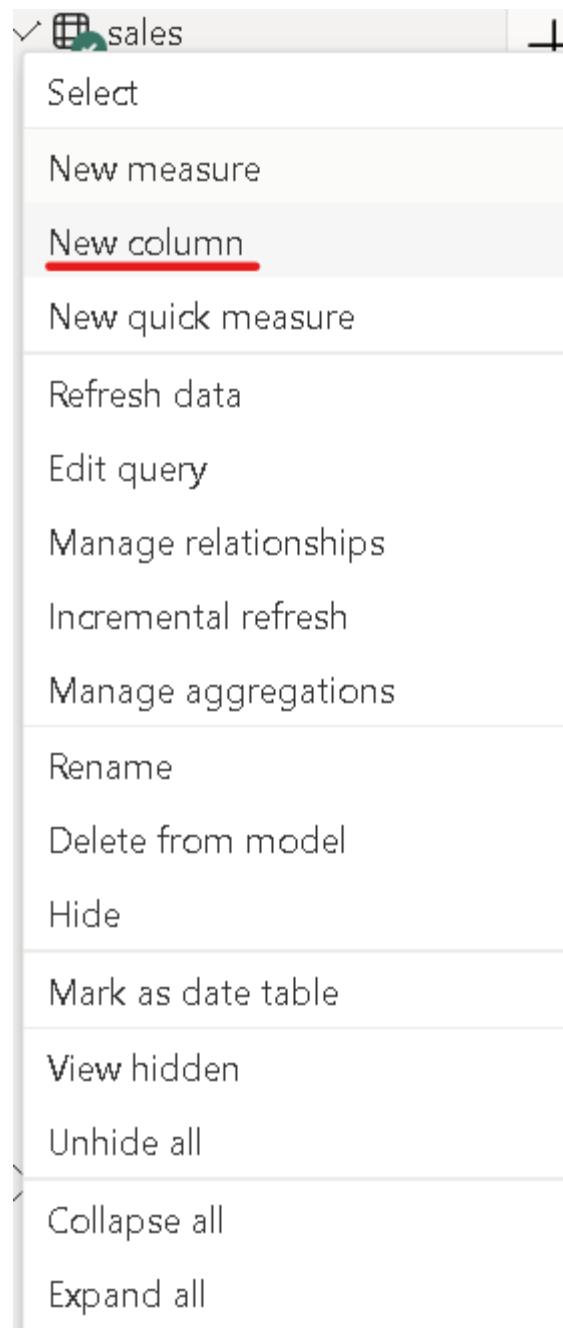
- To create a calculated column, it's often easiest to start in the **Table View** in Power BI. This view allows you to see all the existing columns and data within your tables, which can help you plan your calculations.

2. Identifying the Need for a Calculated Column:

- Suppose you have a table that includes a column for **Price** and another for **Discount**. The **Discount** column contains some numeric values representing percentage discounts (e.g., 50 for 50%) and some "NA" values where no discount is applied.
- You want to create a new column called **Discounted Price** that calculates the price after applying the discount. This requires a DAX formula to handle both the numeric discount values and the "NA" cases.

3. Creating the Column:

- Right-click on the table where you want to create the calculated column, and select **New Column** from the context menu. This opens the DAX formula bar at the top of the canvas.



- In the formula bar, you can enter the DAX expression to calculate the **Discounted Price**. The formula might look like this:

The screenshot shows the Power BI formula bar with the following details:

- Logical Test:** IF(LogicalTest, ResultIfTrue, [ResultIfFalse])
- Description:** Checks whether a condition is met, and returns one value if TRUE, and another value if FALSE.
- Formula:** Discounted Price = IF(sales[promo_discount_2] == "NA", sales[price], sales[price] * sales[promo_discount_2] / 100)
- UI Elements:** Number dropdown, Sort by column dropdown, Data groups dropdown, Mana relations dropdown, Sort button, Groups button, Relation button.

- If we explain the code:
 - There are "NA" values in the "promo_discount_2" column. Since these values are "text", they cannot be processed with numbers. Therefore, the "price * promo_discount_2 / 100" operation will be applied for values that are not "NA" as a result of this if.

4. Handling Errors:

- Initially, you might encounter an error if the discount column contains "NA" as a text value. Power BI cannot directly multiply a text value by a number. The formula above addresses this by using an **IF** statement to handle the "NA" cases separately.
- After writing the formula, press **Enter** to create the column. If everything is correct, the new **Discounted Price** column will appear in your table with the calculated values.

5. Adjusting for Percentage Discounts:

- Remember that discount values like "50" represent 50%. Therefore, to correctly apply the discount, the formula divides the discount by 100. This ensures that the calculation reflects the intended percentage.

6. Renaming the Column:

The screenshot shows the Power BI Data Editor interface. A calculated column named 'Discounted Prices' has been created. The formula is: `=IF(sales.promo_discount_2>=0,"",sales.price*(1-sales.promo_discount_2)/100)`. The column is highlighted with a red border. The Data pane on the right lists columns such as order_id, product_id, store_id, order_date, sales, revenue, price, promo_type_1, promo_type_2, promo_desc_1, promo_desc_2, delivery_date, and delivery_date_format2.

- You can give your calculated column a meaningful name, like **Discounted Price**, to make it easy to identify in your data model.

Understanding the Role of Calculated Columns

Calculated columns are different from measures in a few key ways:

- Row-Level Calculations:** Calculated columns perform row-level calculations. Each row in the table is processed independently, and the results are stored in the table.
- Static Values:** Once calculated, the values in a calculated column are static and do not change unless the underlying data or the formula is updated.
- Storage:** The results of calculated columns are stored in the data model, which can increase the size of your model. This is different from measures, which calculate values on the fly.

▼ Measures

In this section, we'll explore **Measures**, a fundamental concept in DAX (Data Analysis Expressions) within Power BI. While calculated columns create physical columns in your data model, measures are dynamic calculations that aggregate data on the fly, offering powerful capabilities for real-time analysis without consuming additional storage space.

What Are Measures?

Measures are dynamic calculations that you create in Power BI using DAX. Unlike calculated columns, which store results in a table, measures perform calculations on demand based on the context of your report. This means that measures do not occupy physical space in your data model—they are computed in real-time as users interact with the report.

Key Differences Between Measures and Calculated Columns

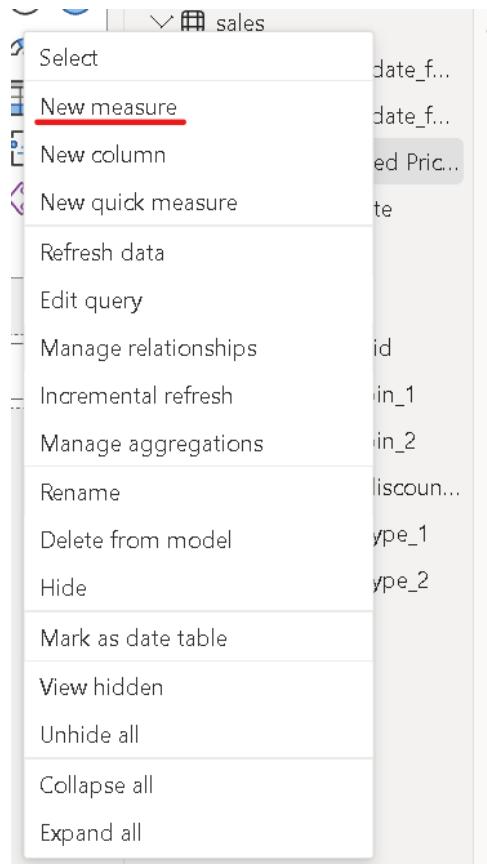
- **Storage:** Calculated columns are stored in the data model, taking up memory and storage space. Measures, however, do not take up storage space because they are calculated on the fly.
- **Scope:** Calculated columns operate at a row level, meaning each row in the table has a unique value for the calculated column. Measures operate at the aggregate level, calculating results based on the context provided by filters, slicers, or report visuals.
- **Performance:** Because measures are not stored in the data model, they are often more efficient for calculations that do not need to be stored for each row but rather calculated as needed.

Creating a Measure

Let's walk through the process of creating a measure in Power BI:

1. Navigating to Create a Measure:

- Measures can be created in either the **Table View** or the **Report View**. However, creating them in the Report View is often preferred for convenience.
- To create a measure, right-click on the table where you want to store the measure and select **New Measure**. While measures are stored in a specific table for organizational purposes, they are not tied to that table they can be used across your entire data model.



2. Writing a Measure Formula:

- Suppose you want to calculate a 5% revenue split for a customer. You can create a measure called **Revenue Split (5%)** that calculates 5% of the total revenue.
- Enter the following DAX formula in the formula bar:

The screenshot shows the Power BI interface with the following components:

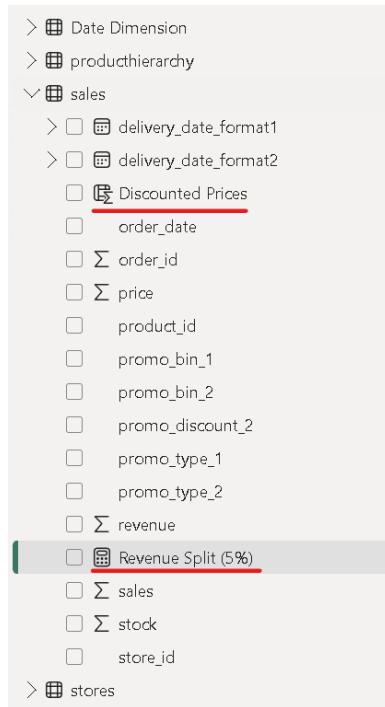
- Formula Bar:** Displays the formula `1 Revenue Split (5%) = SUM(sales[revenue])*0.05`. The entire formula is highlighted with a red box.
- Table View:** Shows a table with columns: Order Date, Revenue, and Orders. The data for January 2017 is as follows:

Order Date	Revenue	Orders
1/2/2017	\$50.60	46
1/3/2017	\$53.25	38
1/4/2017	\$53.49	43
1/5/2017	\$70.36	46
1/6/2017	\$45.92	44
1/7/2017	\$45.90	45
1/8/2017	\$2478	45
1/9/2017	\$0.00	0
- Visual:** A bar chart titled "Sum of revenue by Day Name" showing daily revenue totals. The Y-axis ranges from \$10K to \$30K.

- This formula sums up the **Revenue** column from the **Sales** table and then multiplies it by 5% (0.05). The result is dynamically calculated each time the measure is used in a report.

3. Naming and Formatting:

- You can name the measure anything that makes sense for your analysis, such as **Revenue Split (5%)**. DAX allows you to include spaces and special characters in measure names for readability.
- Once created, the measure will be indicated by a calculator icon, distinguishing it from other columns.



4. Using the Measure in a Report:

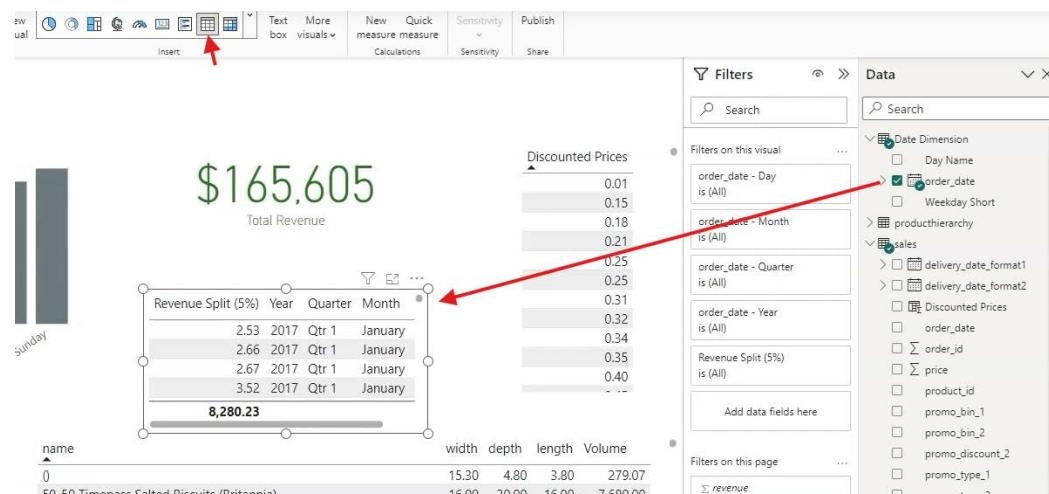
- After creating the measure, you can drag and drop it into your report visuals just like any other field. For example, you might add it to a table or chart to display the 5% revenue split.
- Measures are context-sensitive, meaning their values change based on the filters and slicers applied in the report. For example, if you filter the report by a specific year, the measure will recalculate to show the 5% split of revenue only for that year.

5. Understanding Aggregation in Measures:

- Measures are inherently aggregative. This means that the calculation will always consider the entire dataset or the filtered context in the report.



- If you use a measure in a table visual alongside other fields like dates or product categories, the measure will display aggregated results (e.g., the sum of revenue split for each date or category).



Advantages of Using Measures

- Performance Efficiency:** Since measures do not consume storage in the data model, they are generally more efficient, especially when dealing with large datasets.

- **Flexibility:** Measures offer flexibility as they can be reused across multiple visuals and adjusted automatically based on the report context.
- **Scalability:** Measures are ideal for calculations that need to adapt to different levels of data aggregation, such as totals, averages, or other summary statistics.

Understanding the distinction between calculated columns and measures is crucial for effective Power BI modeling. Measures allow you to perform powerful, on-the-fly calculations that enhance the flexibility and efficiency of your reports. While calculated columns are best suited for row-level calculations that need to be stored in your data model, measures excel at providing dynamic, aggregated insights that adjust automatically based on the report context.

▼ AVERAGE

In this section, we will explore the `AVERAGE` function, one of the commonly used functions in DAX (Data Analysis Expressions) within Power BI. The `AVERAGE` function is utilized to calculate the mean of a set of numbers, providing valuable insights for data analysis in your reports.

Using the AVERAGE Function

In previous lessons, we used the `SUM` function to calculate the total revenue from our sales data. However, there may be scenarios where we need to calculate the average revenue per order instead of the total revenue. In such cases, we must use the `AVERAGE` function to create a new measure.

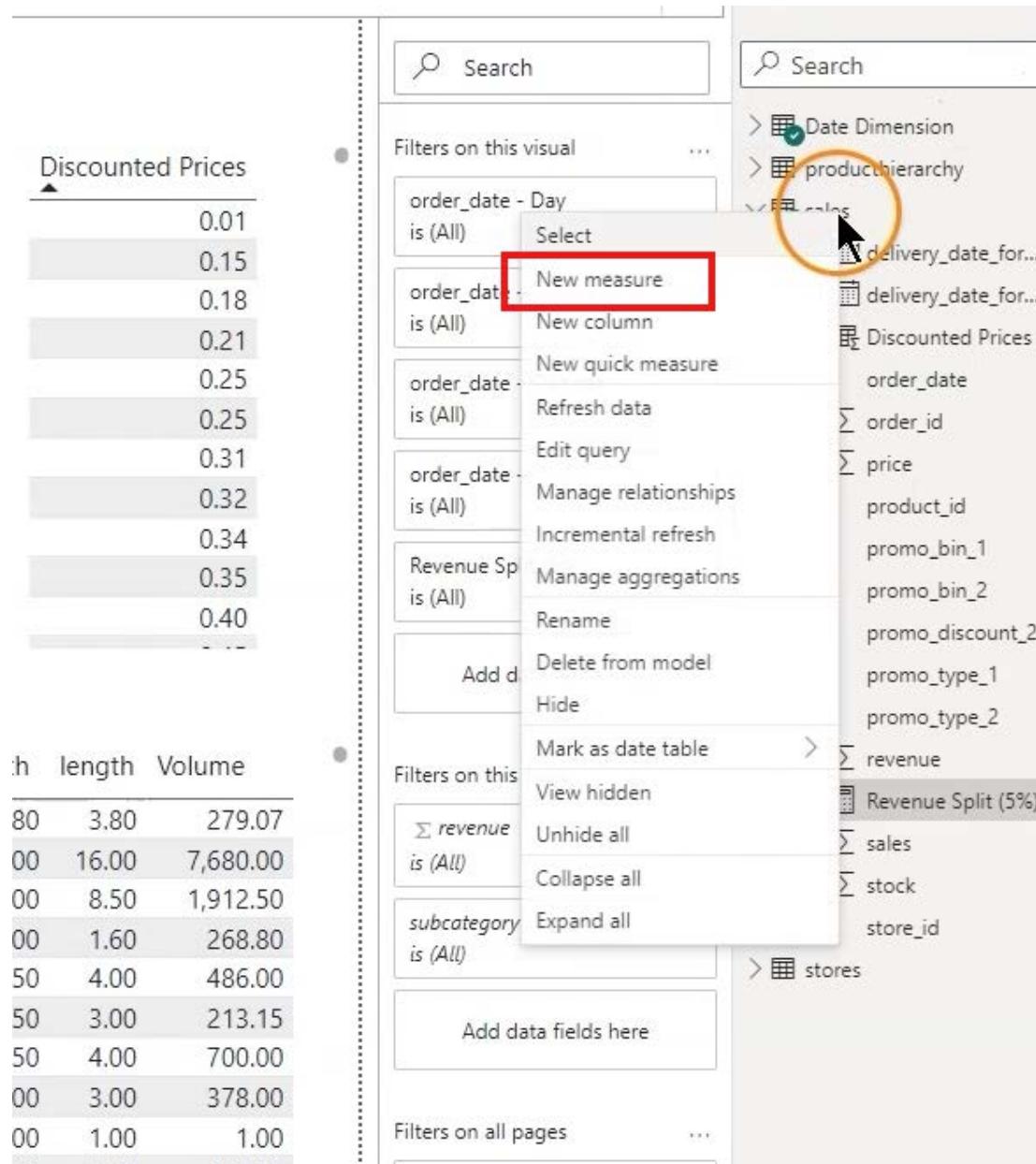
Example Scenario:

Consider a customer who wants to know the average revenue per order in addition to the total revenue. To achieve this, we need to create a new measure using the AVERAGE function.

Step-by-Step Guide to Creating a Measure with the AVERAGE Function

1. **Switch to Data View:** In Power BI, go to the Data view by selecting the appropriate icon on the left pane.

2. **Create a New Measure:** Right-click on the desired table (e.g., Sales table) and select "New Measure."



3. **Define the Measure Name:** Enter a name for the new measure, such as

Average Revenue .

4. **Write the DAX Formula:** In the formula bar, enter the following DAX formula:

1 AVG Revenue = AVERAGE(sales[revenue])*0.05

This formula calculates the average of the values in the `Revenue` column of the `Sales` table.

Filter Context and Measures

In DAX, the calculation of measures is influenced by the **filter context**. Measures are dynamically calculated based on the filters applied in the report. For instance, if you apply filters for specific years or months in your report, the `Average Revenue` measure will calculate the average revenue only for the selected period.

The filter context includes all selections and slicers applied in the report, which dynamically affects the measure calculations. When a measure is evaluated, all active filters are considered, and the calculation is performed accordingly.

Year	Quarter	Month	Day	Revenue Split (5%)	AVG Revenue
2019	Qtr 3	September	27	45.01	0.30
2019	Qtr 3	August	2	26.17	0.16
2018	Qtr 4	December	28	26.09	0.39
2019	Qtr 4	October	4	24.19	0.17
2019	Qtr 1	March	1	21.77	0.16
2019	Qtr 1	March	22	19.70	0.16
2019	Qtr 3	September	13	18.23	0.12
Total				980.41	0.09

Comparing Measures and Calculated Columns

It is important to understand the difference between measures and calculated columns. A measure is computed on-the-fly based on the current filter context, while a calculated column is evaluated row by row during data refresh and remains static unless the data model is refreshed.

For example, the `AVERAGE` function can be used in both measures and calculated columns. However, in this lesson, we focus on using the `AVERAGE`

function in a measure, as it requires dynamic calculation based on the filter context.

In this section, we have covered the basics of using the `AVERAGE` function in DAX to calculate average values. This function allows for dynamic and flexible analysis in your reports.

▼ Measure Table

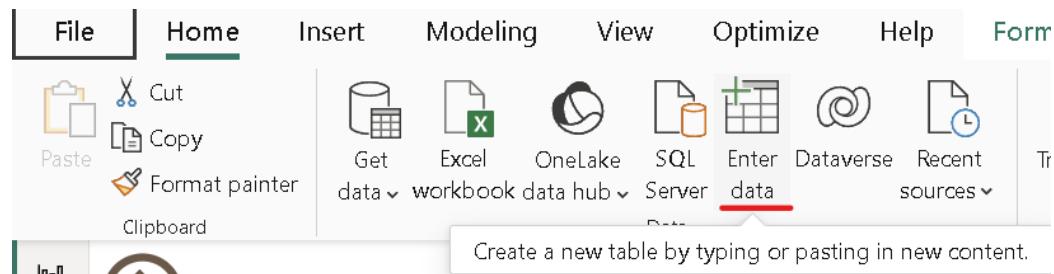
To efficiently manage all our measures in Power BI, it is advisable to create a dedicated "Measures Table." While measures in Power BI appear to be associated with specific tables, they are not technically part of those tables. This can lead to organizational confusion, especially as the number of measures increases. To maintain clarity and streamline the management of measures, we can create a separate table specifically for this purpose.

Creating a Measures Table

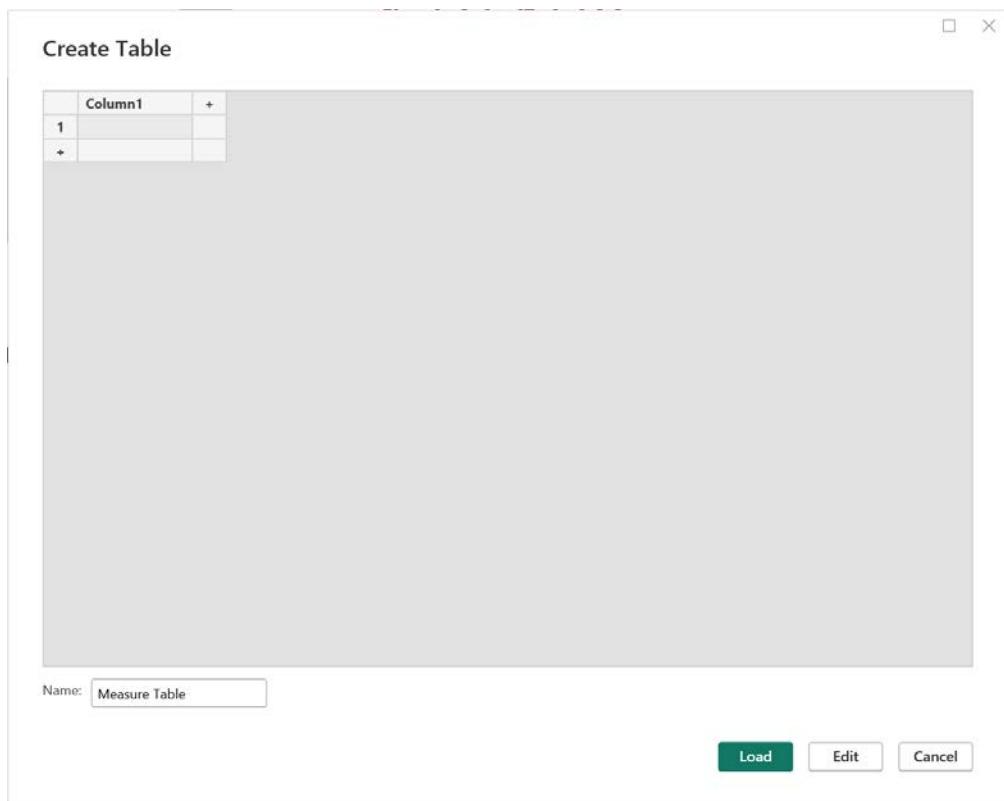
A Measures Table is essentially a regular table used solely for storing measures. Here's how to create one:

1. Create a New Table:

- Navigate to the "Home" tab in Power BI Desktop.
- Click on "Enter Data" to create a new table. This method allows you to manually input data. However, for a Measures Table, the data input step is just a formality.



- Name the new table "Measures Table" or any name that clearly indicates its purpose. You can add a dummy column (e.g., "Column1") with no data, as this will be deleted later.

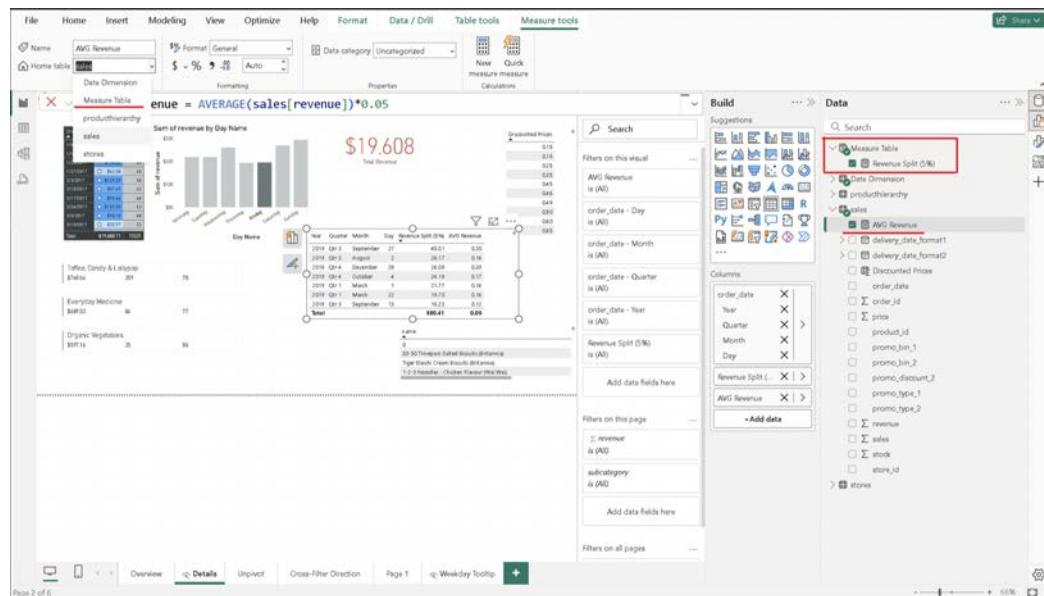


2. Load the Table into the Model:

- Once you've named the table and input the placeholder column, click "Load" to add the table to your model.

3. Moving Measures to the Measures Table:

- After the Measures Table is created, you can start moving your existing measures into it. To do this:
 - Select the measure you want to move.
 - In the "Measure Tools" tab, locate the "Home Table" field. This field allows you to change the table where the measure is stored.
 - Select "Measures Table" from the dropdown list. The measure will now appear under the Measures Table.



4. Finalizing the Measures Table:

- Once you've moved at least one measure to the Measures Table, you can delete the placeholder column by right-clicking on it and selecting "Delete from Model." Confirm the deletion, and the table will now appear as a dedicated Measures Table, indicated by a unique icon.

By following these steps, you can ensure that all your measures are neatly organized in a single table, improving the readability and maintainability of your data model.

▼ COUNT, DISTINCTCOUNT & More

In this section, we'll explore various DAX functions used for counting in Power BI, including `COUNT`, `DISTINCTCOUNT`, `COUNTROWS`, and `COUNTBLANK`.

Understanding these functions is crucial for performing accurate data analysis, especially when you need to count different types of data or handle datasets with missing values.

1. Understanding COUNT and DISTINCTCOUNT

Suppose our customer wants to analyze the number of different promotion types they have. To do this, we can use DAX functions that count values in a column. Let's start by creating a measure to count all promotions.

Creating a COUNT Measure

To create a measure that counts the number of entries in the "Promo" column:

1. Create a New Measure:

- Right-click on the "Measures Table" or in any existing table where you want to store this measure.
- Select "New Measure."

2. Define the Measure:

- Name it appropriately, such as `Count of Promos`.
- Use the `COUNT` function to count the total number of entries in the "Promo" column:

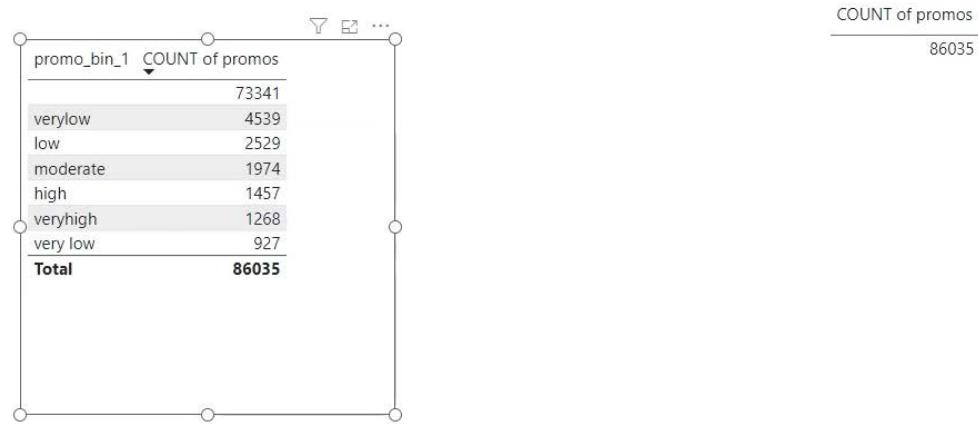
`COUNT of Promos = COUNT(sales[promo_bin_1])`

- This measure counts all non-blank values in the "Promo" column.
- Example: COUNT of Stock

`COUNT of Stock = COUNT(sales[stock])`

Visualizing the COUNT Measure

- Drag this measure into a table visualization to see the total count. For example, if it shows 86,000, this represents the total number of entries, including duplicates.



Using DISTINCTCOUNT for Unique Values

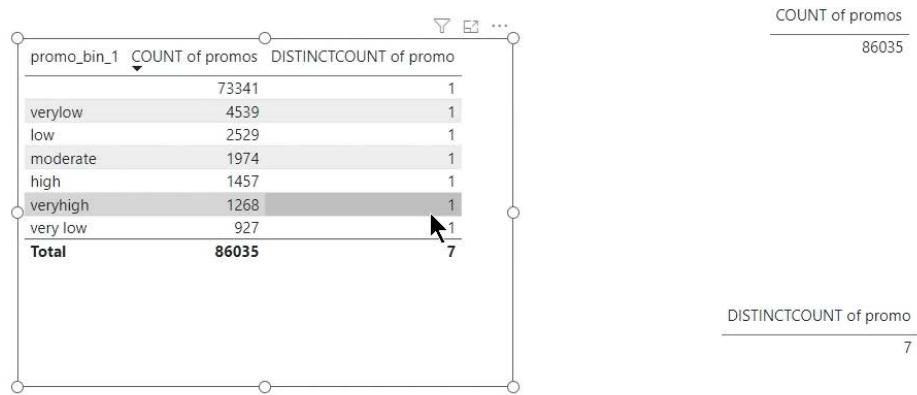
Now, if we want to count only the unique promotion types:

1. Create a DISTINCTCOUNT Measure:

- Right-click again and select "New Measure."
- Name it `Distinct Count of Promos` and define it using the `DISTINCTCOUNT` function:

```
1 DISTINCT COUNT of Promos = DISTINCTCOUNT(sales[promo_bin_1])
```

- This measure counts only the distinct values in the "Promo" column, ignoring duplicates.
- When you add this measure to a table visualization, it might show a smaller number, such as 7, indicating there are 7 unique promotion types.
- The first row is blank.



2. Using COUNTROWS for Row Counting

While `COUNT` and `DISTINCTCOUNT` are used for counting values in a column, `COUNTROWS` counts the total number of rows in a table.

Creating a COUNTROWS Measure

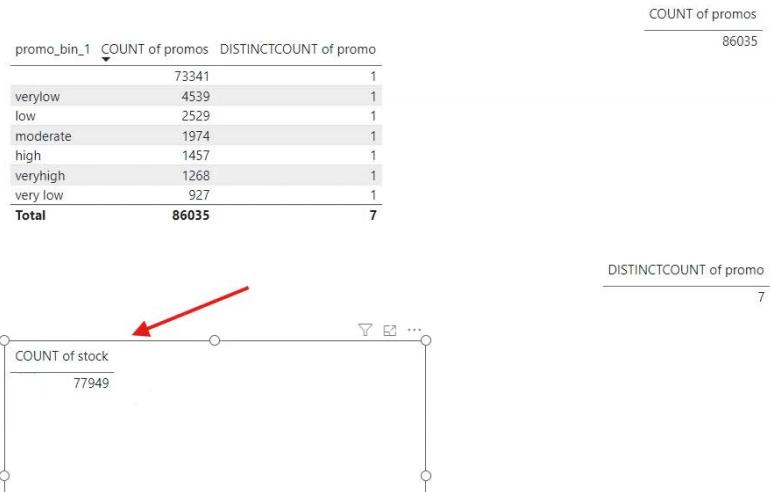
To find out the total number of rows, including blanks:

1. Create a COUNTROWS Measure:

- Right-click and select "New Measure."
- Name it `Count of Rows` and define it using `COUNTROWS`:

COUNTROWS of Stock = `COUNTROWS(sales)`

- This measure counts all rows in the "Sales" table, including those with blank values in any column.



3. Handling Blanks with COUNTBLANK

Sometimes, it's necessary to identify how many entries are missing data. This is where `COUNTBLANK` comes in handy.

Creating a COUNTBLANK Measure

To count blank values in a specific column:

1. Create a COUNTBLANK Measure:

- Right-click and select "New Measure."
- Name it `Count of Blank Promos` and use the `COUNTBLANK` function:

`COUNT Blank of Stock = COUNTBLANK(sales[stock])`

`Blanks (calculated) of stock = COUNTROWS(sales)-COUNT(sales[stock])`

COUNTROWS of Stock	COUNT of Stock	Blanks (calculated) of stock	COUNT Blank of Stock
86035	77949	8086	8086

Summary of Count Functions

- `COUNT`: Counts non-blank values in a column.

- **DISTINCTCOUNT** : Counts distinct non-blank values in a column.
- **COUNTROWS** : Counts the total number of rows in a table.
- **COUNTBLANK** : Counts blank values in a column.

Each of these functions has its use case, depending on whether you need to count total entries, unique values, total rows, or blanks. Understanding when and how to use each function is key to mastering data analysis in Power BI.

▼ SUMX & Iterative Functions

In this session, we'll explore **row-based or iterative functions** in Power BI, focusing on how they enable calculations at the row level within measures. This approach is essential when performing dynamic, row-level computations that do not require storing intermediate results in the data model. By leveraging iterative functions like **SUMX**, we can calculate totals in a flexible and efficient manner without physically adding new columns.

Why Use Row-Based Functions?

When dealing with row-level calculations, we often encounter a need to evaluate expressions for each row and then aggregate the results. Let's consider an example from a **Sales** table, where we want to calculate revenue based on the **discounted price**:

1. If we were to multiply sales by the discounted price for each row, we could store this in a new calculated column. However, this adds unnecessary data to the model, affecting performance and increasing storage.
 2. A more efficient method is to compute this directly within a measure, which is more dynamic and doesn't require storing intermediate results.
-

Attempting a Basic SUM

A common mistake when performing these calculations is to try and sum multiple values directly, like multiplying **sales** and **discounted prices** in a basic SUM. Here's why this doesn't work:

promo_bin_1	COUNT of promos	DISTINCTCOUNT of promo
verylow	4539	1
low	2529	1
moderate	107	1

- If we simply sum all sales and then sum all discounted prices and multiply them, we get an incorrect result.
- This is because Power BI first aggregates the entire sales column and the entire discounted price column separately, then multiplies the two sums together. We need to calculate the product of these two values for each row **before** summing them.

Introducing SUMX for Row-Level Calculations

To solve this, we use the **SUMX** function, which evaluates an expression for each row and then aggregates the results.

Syntax of SUMX:

```
SUMX ( Table, Expression )
```

- **Table**: The table where the row-level expression is evaluated.
- **Expression**: The calculation to perform on each row.

Example Walkthrough

Let's walk through the steps:

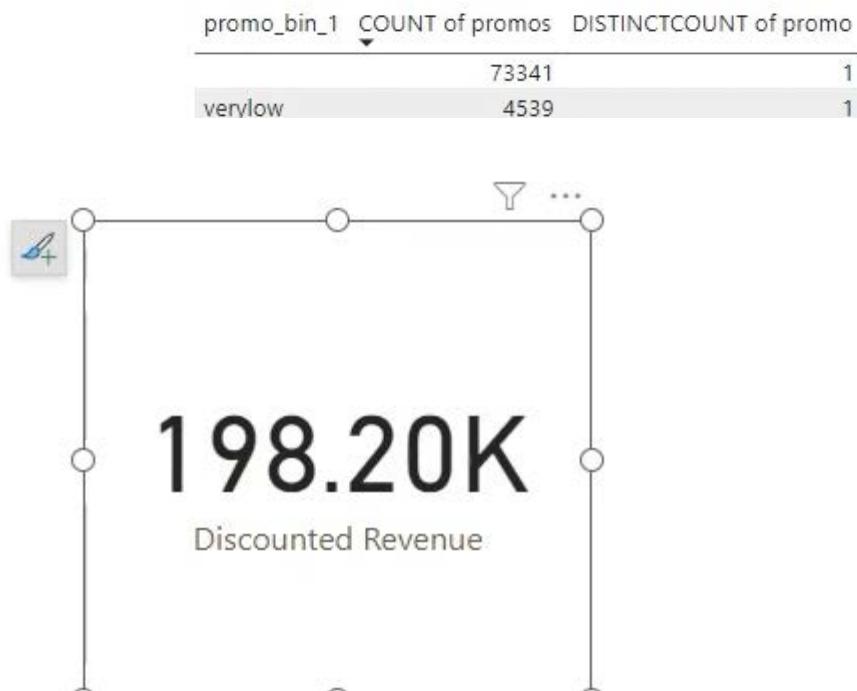
1. Create a New Measure:

- Navigate to the **Measures** table and create a new measure called "Discounted Revenue."

2. Write the SUMX Function:

- Use the **SUMX** function to calculate the revenue on a row-by-row basis:

Structure	Formatting	Properties
X ✓	1 Discounted Revenue = SUMX(sales,sales[sales]*sales[Discounted Prices])	



3. Evaluate the Result:

- This will iterate through each row in the **Sales** table, multiply **Quantity** by **Discounted Price**, and sum the results, giving the correct revenue total.

Advantages of Using SUMX and Iterative Functions

- **Flexibility:** Measures using **SUMX** are dynamic, allowing for complex calculations without physically adding columns to the data model.

- **Performance:** Since iterative functions don't store intermediate calculations, they are often more efficient for large datasets.
- **Scalability:** As new filters or calculations are added, the measure can dynamically adjust without having to recompute or store new values.

Iterative functions like **SUMX** is powerful tools in Power BI, enabling complex, row-level calculations without adding physical columns to the data model. They enhance flexibility, improve performance, and allow for more scalable solutions when working with large datasets or complex reporting needs.

▼ AVERAGEX

In this lecture, we'll explore the **AVERAGEX** function, another iterative function in Power BI. Like **SUMX**, it calculates an expression at the row level and then aggregates the results. However, instead of summing the results, **AVERAGEX** calculates the average of the evaluated row-level expressions.

What is AVERAGEX?

The **AVERAGEX** function is used to compute the average of an expression that is evaluated on each row of a table. It's especially useful when you need to calculate the average of derived or computed values rather than just column values.

Syntax of AVERAGEX:

```
AVERAGEX ( Table, Expression )
```

- **Table:** The table over which the expression is evaluated row by row.
- **Expression:** The calculation or expression evaluated for each row.

Example: Calculating the Base Dimension

Let's say we have a **Products** table containing **Length** and **Width** columns, and we want to calculate the average **Base Dimension** (calculated as the

product of Length and Width) for each product.

Here's how we would do this with **AVERAGEX**:

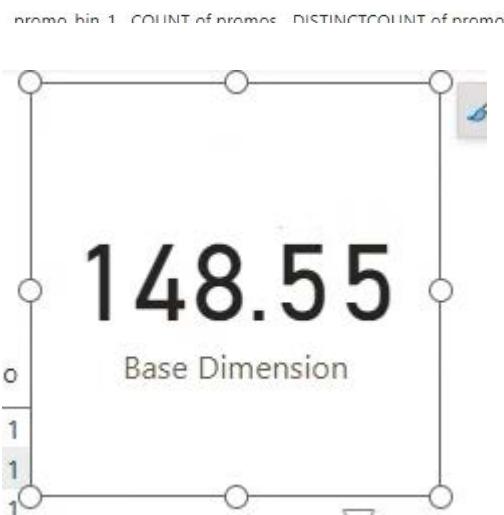
1. Create a New Measure:

- Navigate to the **Measures** table in Power BI and create a new measure named "Base Dimension."

2. Write the AVERAGEX Function:

- Use the **AVERAGEX** function to compute the base dimension:

Structure	Formatting	Properties	measure measure	Calculations
X ✓	1 Base Dimension = AVERAGEX(producthierarchy, producthierarchy[length]*producthierarchy[width])			



3. Explanation:

- The function iterates over the **Products** table, calculating the product of **Length** and **Width** for each row, and then takes the average of those results.

Example Walkthrough

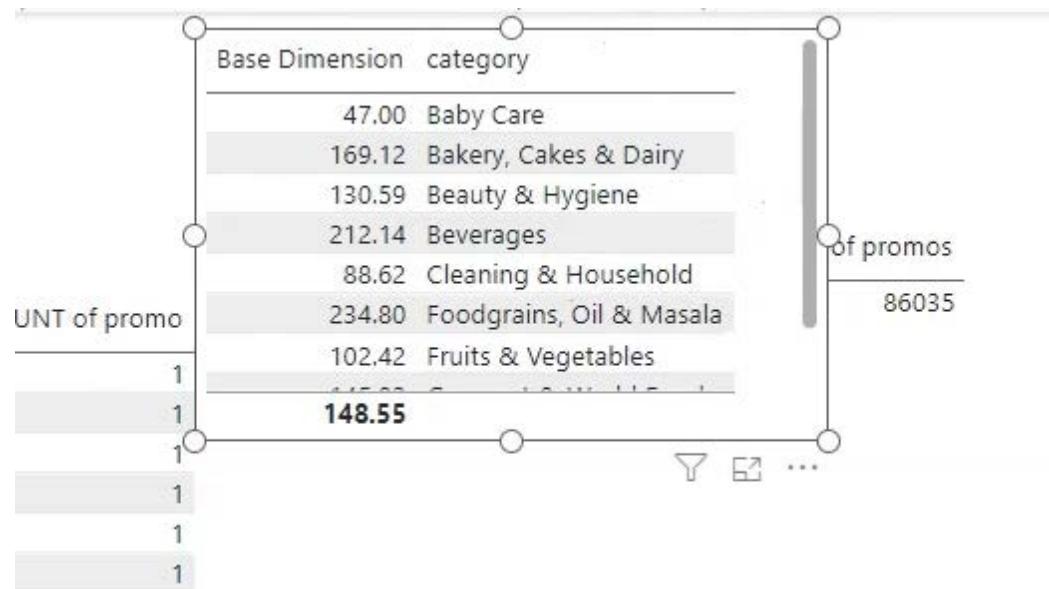
After creating the measure, you can visualize it in different contexts:

1. Overall Average:

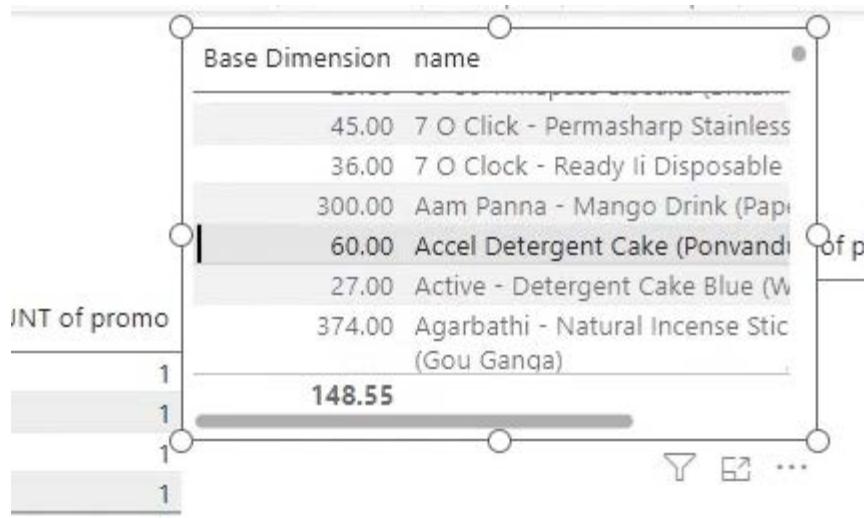
- Drag the **Base Dimension** measure onto the Canvas. This will show the average base dimension across all products. For example, it may show an overall base dimension of **148.55**.

2. Contextual Averages:

- You can add different categories to the visualization. For instance, you could calculate the base dimension average for each **Product Category** or for individual **Products**:
- When grouped by **Category**, Power BI will calculate the average base dimension within each category.



- Similarly, grouping by **Product Name** will show the average base dimensions for individual products.



This dynamic capability to evaluate averages at different levels such as product or category makes **AVERAGEX** a highly flexible function.

Rounding the Results

After calculating the average, you may notice that the values have many decimal places. If you need to round these values, you can use the **ROUND** function in combination with **AVERAGEX**.

Example of Rounding:

```
Base Dimension Rounded = ROUND(AVERAGEX ( Products, Products[Length] * Products[Width] ), 2)
```

- In this example, the result is rounded to two decimal places.

Conclusion

The **AVERAGEX** function is a powerful tool for performing row-level calculations and averaging the results dynamically. It allows you to create flexible and context-aware calculations without adding unnecessary columns to the data model. You can easily apply it in different contexts, such as by product category or individual product, giving you control over how data is aggregated and displayed.

In the next lecture, we'll dive deeper into advanced use cases and explore how to manage decimal places and rounding more effectively.

▼ ROUND

In this lecture, we'll explore the **ROUND** function in Power BI, which allows us to control the precision of values in both **measures** and **calculated columns**. We can round numbers to a specific number of decimal places or to the nearest whole number, and even round values to the nearest tens, hundreds, or other multiples by using negative values.

What is the ROUND Function?

The **ROUND** function is used to round a number to a specified number of digits.

Syntax of ROUND:

```
ROUND ( number, num_digits )
```

- **number**: The value or expression that you want to round.
- **num_digits**: The number of decimal places to which you want to round the value.
 - **0**: Rounds to the nearest whole number.
 - **Positive numbers**: Rounds to that many decimal places.
 - **Negative numbers**: Rounds to the nearest 10s, 100s, etc.

Example 1: Rounding Length and Width Separately

Let's start by rounding the **Length** and **Width** values in our base dimension calculation.

1. Rounding Length:

```
ROUND(Products[Length], 0)
```

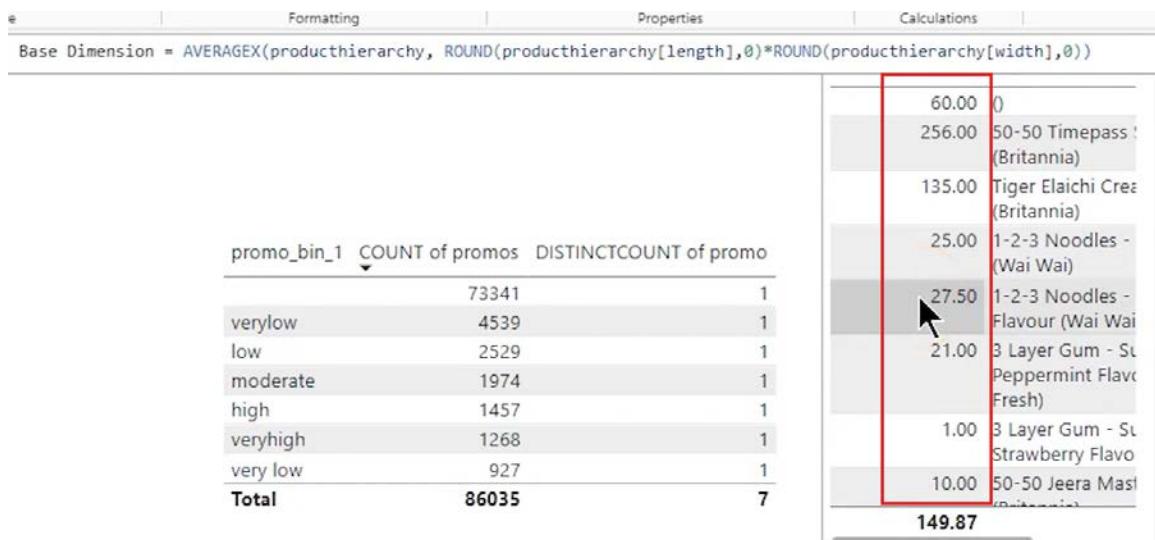
- This rounds the length to the nearest whole number. We use `0` for whole number rounding.

2. Rounding Width:

```
ROUND(Products[Width], 0)
```

- Similarly, we can round the width to the nearest whole number.

Now, if we calculate the **Base Dimension** as the product of rounded length and width:

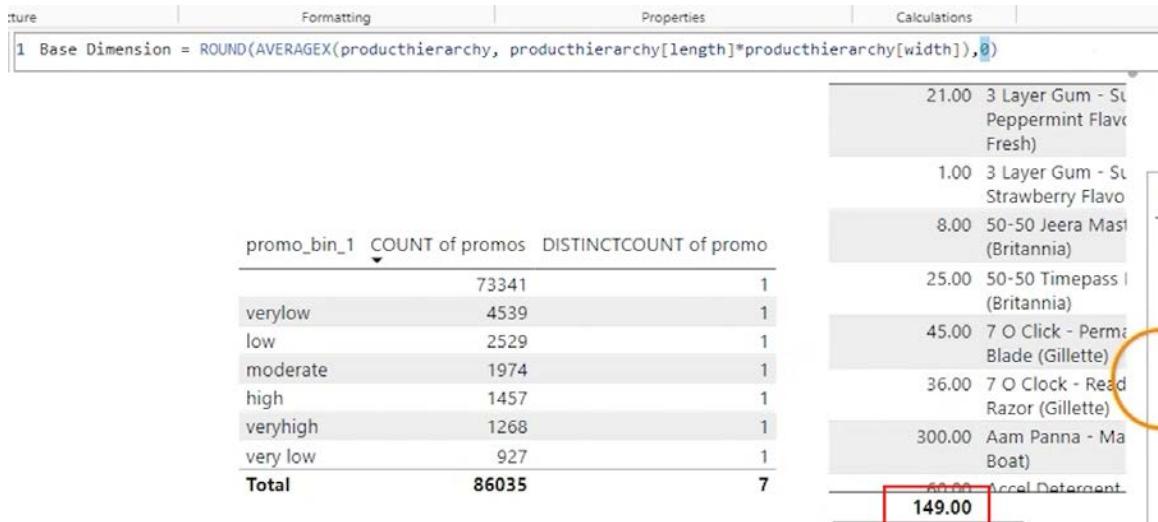


In this case, each length and width value is rounded first, and then their product is calculated before averaging the results.

Example 2: Rounding the Final Result

Instead of rounding the individual components, we might prefer to round the **final result** of the base dimension calculation. This can give a cleaner, overall average rounded to a specified precision.

Let's apply the **ROUND** function around the result of the **AVERAGEX** function:



Here, we are rounding the final average result to a whole number by using `0`. If you wanted to round to one decimal place, you could replace `0` with `1`.

Example 3: Rounding to Tens, Hundreds, etc.

The **ROUND** function can also be used to round to larger units, such as the nearest tens, hundreds, or thousands. This is done by using **negative values** for the `num_digits` argument.

Example of Rounding to the Nearest 10:

```
Rounded to Tens = ROUND(Products[SalesAmount], -1)
```

- If the value of **SalesAmount** is 36, it becomes 40.
- If the value is 49, it becomes 50.

Example of Rounding to the Nearest 100:

```
Rounded to Hundreds = ROUND(Products[SalesAmount], -2)
```

- If the value is 152, it becomes 200.
- If the value is 249, it becomes 200.

This flexibility allows us to round values as needed based on the context of the report.

The **ROUND** function in Power BI is a simple yet powerful tool that helps control the precision of your calculations. Whether you need to round to decimal places, whole numbers, or larger units (like tens or hundreds), the **ROUND** function provides the flexibility to do so. You can round individual components within expressions or the final result of a calculation, depending on the requirements of your report.

▼ RELATED

In this lecture, we will explore the **RELATED** function in Power BI, which allows us to leverage existing relationships within our data model to perform calculations that involve multiple tables. This function is particularly useful when you need to access data from related tables to enhance your measures or calculated columns.

What is the RELATED Function?

The **RELATED** function allows you to retrieve values from a related table based on existing relationships in your data model. It's commonly used in calculated columns or measures when data from one table is needed to perform calculations in another table.

Syntax of RELATED:

```
RELATED ( column_name )
```

- **column_name**: The column from the related table that you want to reference.

Use Case: Calculating Fulfillment Costs

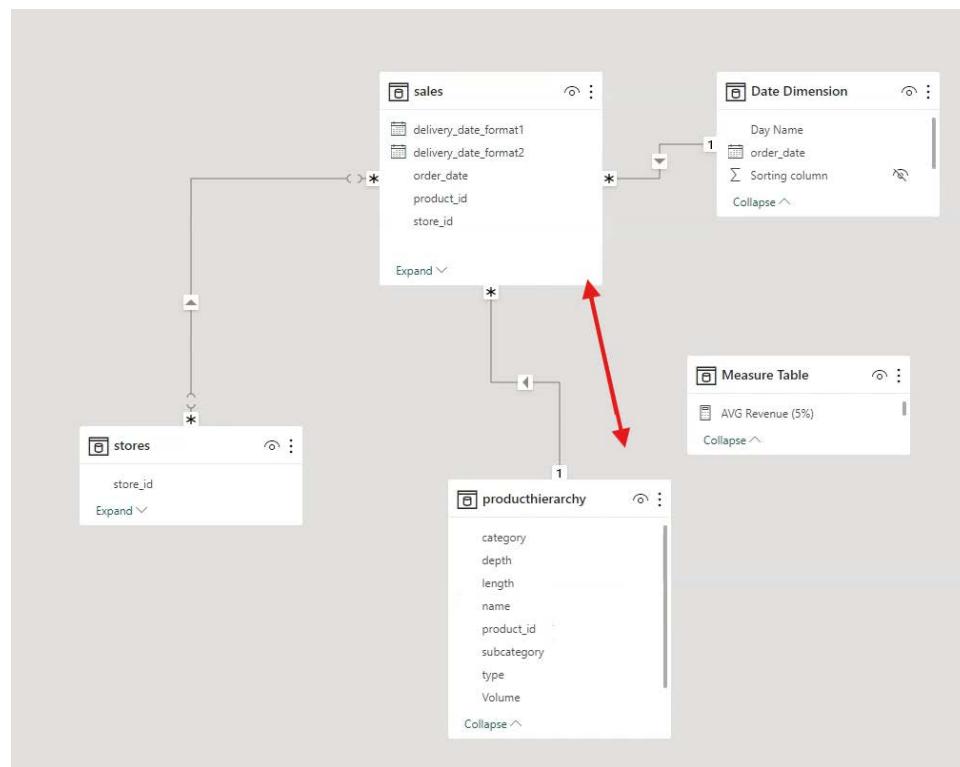
Let's consider a use case where we have a logistics company fulfilling orders, and we need to calculate the cost based on the volume of products sold. The volume information is stored in the **Producthierarchy** table, while the

sales data is in the `Sales` table. By using the `RELATED` function, we can bring the volume information into the `Sales` table to perform the necessary calculations.

Step-by-Step Calculation

1. Setting Up the Data Model:

- Ensure that there is a relationship between the `Sales` table and the `Producthierarchy` table. In this case, the relationship is based on the product ID.



2. Creating the Measure:

We'll create a measure that multiplies sales by volume for each row in the sales table and then divides the result by 1,000 to calculate the final cost.

- **Sales Table:** Contains the number of products sold.
- **Product Table:** Contains the volume of each product.

The relationship between these tables allows us to use the **RELATED** function to retrieve product volume for each sale.

Here's how to write the measure:

```
Fulfillment Cost =  
SUMX(  
    Sales,  
    Sales[Quantity] * RELATED(Products[Volume]) / 100  
    0  
)
```

3. Using the **RELATED** Function:

- Since the volume data is stored in a different table (**Producthierarchy** table), we use the **RELATED** function to bring this data into the **Sales** table for our calculation.

```
1 Cost of Fulfillment = SUMX(sales, sales[sales]*RELATED(producthierarchy[Volume]))/1000
```

order_id	Sum of sales	Volume	Cost of Fulfillment
1	0	1,482.00	0.00
2	0	1,460.54	0.00
3	5	946.13	4.73
4	0	1,462.50	0.00
5	1	633.75	0.63
6	0	400.00	0.00
7	0	400.00	0.00
8	0	147.00	0.00
9	0	564.87	0.00
10	2	135.00	0.27
11	0	345.00	0.00
12	1	156.00	0.16
Total		31832	127,704.24

- Explanation:**

- `SUMX`: An iterator function that evaluates an expression row by row.
- `Sales[sales]`: Refers to the number of units sold in each order.
- `RELATED(Producthierarchy [Volume])`: Retrieves the volume from the `Producthierarchy` table for each product sold.
- `/ 1000`: Converts the volume to a suitable unit of measure for calculating the cost.

4. Displaying the Results:

- After creating the measure, add it to a table visual in your report. This will display the fulfillment cost for each order. You can also aggregate these values to see the total fulfillment cost across all orders.

Why Use the RELATED Function?

- **Cross-table calculations:** You can bring in columns from related tables without needing to physically merge or create new columns in the data model.
- **Dynamic calculations:** RELATED allows you to perform dynamic calculations on the fly based on relationships in the model.
- **Clean data model:** By using RELATED, there's no need to create duplicate columns or redundant calculations across tables.

The `RELATED` function in Power BI is a vital tool for data modeling and advanced calculations. By leveraging relationships in your data model, you can perform complex analyses that span multiple tables, making your dashboards more insightful and informative.

▼ 8- DAX - CALCULATE & Filtering

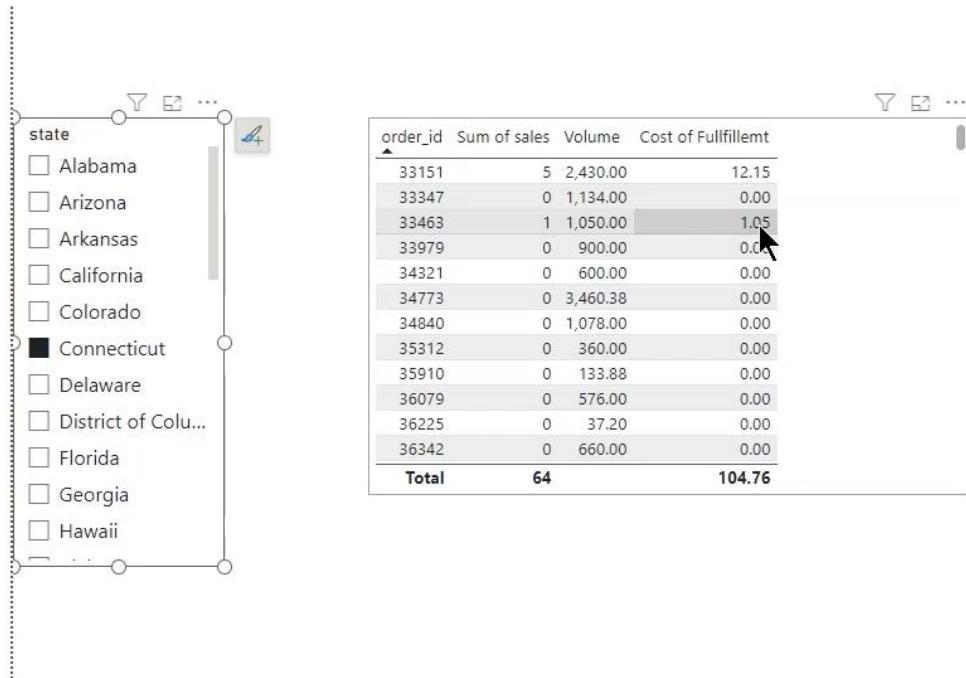
▼ CALCULATE

The **CALCULATE** function is one of the most important and frequently used functions in Power BI's DAX (Data Analysis Expressions). It allows you to modify the **filter context** of a calculation, meaning it can adjust or override

the filters applied to a particular calculation. This gives you greater control over how specific measures or expressions are evaluated, even in the presence of other filters from visuals or slicers.

What is the Filter Context?

The **filter context** in Power BI determines what data is being used for a specific calculation. It can come from multiple sources, such as:



```
1 Cost of Fullfillemt (California) = CALCULATE(SUMX(sales, sales[sales]*RELATED(producthierarchy[Volume]))/1000,stores[state]:"California")
```

The screenshot shows a Power BI interface. On the left, there is a slicer titled 'state' with a dropdown arrow. The list includes Alabama, Arizona, Arkansas, California, Colorado, Connecticut, Delaware, District of Colu..., Florida, Georgia, Hawaii, and a 'more' option. To the right of the slicer is a table with four columns: 'category', 'Cost of Fullfillemt', 'Sum of revenue', and 'Cost of Fullfillemt (California)'. The table contains 11 rows of data, each representing a product category with its corresponding values. At the bottom of the table, there are two summary rows: 'Total' and 'Cost of Fullfillemt (California)'.

category	Cost of Fullfillemt	Sum of revenue	Cost of Fullfillemt (California)
Baby Care	\$5.06	\$189.96	0.72
Bakery, Cakes & Dairy	\$1,673.54	\$10,364.81	147.35
Beauty & Hygiene	\$6,593.95	\$19,606.23	203.04
Beverages	\$1,541.68	\$7,112.19	84.33
Cleaning & Household	\$2,074.87	\$9,631.30	212.01
Foodgrains, Oil & Masala	\$87,276.02	\$14,562.93	232.55
Fruits & Vegetables	\$8,181.41	\$35,850.16	571.89
Gourmet & World Food	\$3,814.09	\$9,360.90	290.72
Kitchen, Garden & Pets	\$90.06	\$395.69	0.00
Snacks & Branded Foods	\$16,453.55	\$58,530.36	1,135.77
Total	\$127,704.24	\$165,604.53	2,878.40

- **Filters from visuals** (e.g., selecting a specific category or state)
- **Slicers or filter panes** applied to the report
- **Rows or columns** in tables that contribute to the filter context based on selections

With the **CALCULATE** function, we can override this context to achieve more specific results.

Syntax of CALCULATE

```
CALCULATE(  
    <expression>,  
    <filter1>,  
    <filter2>,  
    ...  
)
```

- **expression**: The calculation or measure that we want to evaluate.
- **filters**: One or more conditions that modify the existing filter context.

Example Use Case: Calculating Fulfillment Cost for California

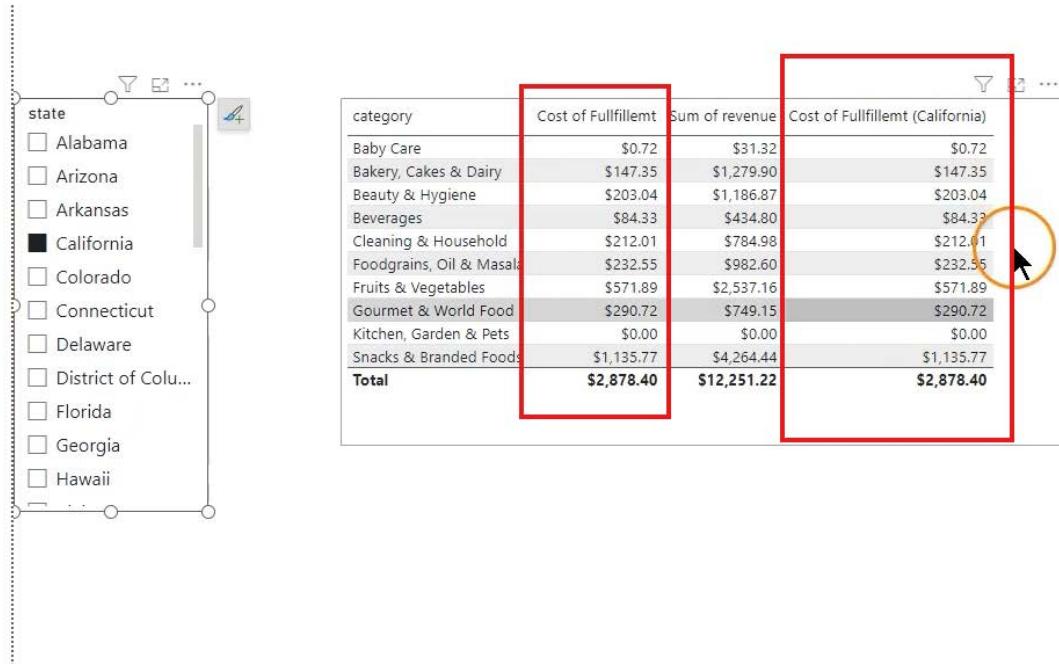
Let's imagine we've already created a measure called **Fulfillment Cost** that calculates the cost of fulfillment for products across all states. Now, suppose we want to calculate the **cost of fulfillment** specifically for orders shipped to **California**.

To achieve this, we can use the **CALCULATE** function to filter the data so that it only includes sales from California. Here's how we would write this measure:

```
1 Cost of Fullfillemt (California) = CALCULATE(SUMX(sales, sales[sales]*RELATED(producthierarchy[Volume]))/1000,stores[state] = "California")
```

The screenshot shows a Power BI interface with a calculated column being defined. On the left, there is a dropdown menu for 'state' with options for various US states. On the right, a table is displayed with columns: 'category', 'Cost of Fullfillemt', 'Sum of revenue', and 'Cost of Fullfillemt (California)'. The 'Cost of Fullfillemt (California)' column contains values such as 0.72, 147.35, 203.04, etc. A formula bar at the top shows the DAX code for the calculated column.

category	Cost of Fullfillemt	Sum of revenue	Cost of Fullfillemt (California)
Baby Care	\$5.06	\$189.96	0.72
Bakery, Cakes & Dairy	\$1,673.54	\$10,364.81	147.35
Beauty & Hygiene	\$6,593.95	\$19,606.23	203.04
Beverages	\$1,541.68	\$7,112.19	84.33
Cleaning & Household	\$2,074.87	\$9,631.30	212.01
Foodgrains, Oil & Masala	\$87,276.02	\$14,562.93	232.55
Fruits & Vegetables	\$8,181.41	\$35,850.16	571.89
Gourmet & World Food	\$3,814.09	\$9,360.90	290.72
Kitchen, Garden & Pets	\$90.06	\$395.69	0.00
Snacks & Branded Foods	\$16,453.55	\$58,530.36	1,135.77
Total	\$127,704.24	\$165,604.53	2,878.40



- **Fulfillment Cost:** This is the measure we previously created, calculating the fulfillment cost.
- **Sales[State] = "California":** This filter limits the data to only orders from California.

This measure will now always display the fulfillment cost for **California**, regardless of any other filters that might be applied.

How CALCULATE Overrides Filter Context

The **CALCULATE** function can **override** any filters applied to a measure. For example, if your visual is showing data for all states, but you've used the **CALCULATE** function to filter specifically for California, the measure will still return values only for California, ignoring any other state filters.

Example:

The screenshot shows a Power BI report interface. On the left, there is a vertical list of states under a 'state' header, with California selected (indicated by a black square). To the right is a table with four columns: 'category', 'Cost of Fulfillment', 'Sum of revenue', and 'Cost of Fulfillment (California)'. The table lists various product categories with their respective costs and revenues. The 'Total' row at the bottom is highlighted with a yellow circle, and a mouse cursor is positioned over it.

category	Cost of Fulfillment	Sum of revenue	Cost of Fulfillment (California)
Baby Care	\$0.54	\$17.85	\$0.72
Bakery, Cakes & Dairy	\$97.19	\$715.70	\$147.35
Beauty & Hygiene	\$331.61	\$1,297.79	\$203.04
Beverages	\$89.71	\$570.77	\$84.33
Cleaning & Household	\$93.82	\$466.43	\$212.01
Foodgrains, Oil & Masala	\$152.28	\$1,050.97	\$232.55
Fruits & Vegetables	\$437.66	\$2,109.25	\$571.89
Gourmet & World Food	\$282.43	\$650.21	\$290.72
Kitchen, Garden & Pets	\$0.00	\$0.00	\$0.00
Snacks & Branded Foods	\$1,030.85	\$4,200.12	\$1,135.77
Total	\$2,516.09	\$11,079.09	\$2,878.40

- If you have a report showing fulfillment costs by state, the column calculated using the **CALCULATE** function for California will always show the value for California, even if other states are selected.

Customizing the Filter Context with CALCULATE

- Override filters:** Use CALCULATE to apply specific filters (like focusing on a single state or category) while keeping other parts of the data model unaffected.
- Apply multiple filters:** You can apply multiple filters within CALCULATE to refine your calculation.
- Static vs. dynamic filtering:** CALCULATE allows you to enforce static filters (like always showing data for California) even when the report or slicers apply different filters.

▼ FILTER

In our previous discussion, we explored the **CALCULATE** function, which is used to modify the filter context of a given measure. Now, we will focus on the **FILTER** function, a powerful tool that works within the **CALCULATE** function to enable more complex filtering scenarios.

When to Use the **FILTER** Function

The `FILTER` function is particularly useful when simple filter conditions aren't sufficient for your calculations. While the `CALCULATE` function can directly apply basic filter conditions, more intricate scenarios require the `FILTER` function. This function allows us to create complex filters by generating a temporary table based on specific criteria, which can then be used in our calculations.

How the `FILTER` Function Works

The `FILTER` function takes two primary arguments:

1. **Table:** The table you want to filter.
2. **Filter Expression:** The condition that determines which rows to include in the temporary table.

Here's how it works: the `FILTER` function creates an in-memory table that includes only the rows meeting the specified condition. This temporary table is then used to calculate the measure.

Example: Filtering the Cost of Fulfillment by California

Let's explore an example where we want to filter the cost of fulfillment specifically for California, but with the flexibility to modify the value depending on the state.

1. Creating a New Measure with `FILTER`:

- Start by creating a new measure. For clarity, we'll name it "Cost of Fulfillment - Filtered."

```
Cost of Fullfilment (California,Filter) = CALCULATE(SUMX(sales,
    sales[sales]*RELATED(producthierarchy[Volume]))/1000,
    FILTER(stores, stores[state]=="California"))
```

state	Cost of Fulfillment	Sum of revenue	Cost of Fulfillment (California)	Cost of Fulfillment (California, FILTER)
Alabama	\$31,371.18	\$5,481.68	\$2,878.40	
Arizona	\$4,299.31	\$12,924.08	\$2,878.40	
Arkansas	\$211.81	\$1,240.67	\$2,878.40	
California	\$2,878.40	\$12,251.22	\$2,878.40	2,878.40
Colorado	\$1,188.35	\$4,412.98	\$2,878.40	
Connecticut	\$104.76	\$257.42	\$2,878.40	
Delaware	\$73.66	\$275.56	\$2,878.40	
District of Columbia	\$487.02	\$1,685.58	\$2,878.40	
Florida	\$2,516.09	\$11,079.09	\$2,878.40	
Georgia	\$656.57	\$2,847.75	\$2,878.40	
Hawaii	\$44.35	\$54.38	\$2,878.40	
Idaho	\$26.43	\$84.42	\$2,878.40	
Total	\$127,704.24	\$165,604.53	\$2,878.40	2,878.40

- **Explanation:**

- `CALCULATE` : Modifies the filter context.
- `FILTER` : Creates a temporary table with rows where the state is California.
- `Stores[State] = "California"` : The filter condition applied within the `FILTER` function.

2. Adjusting for Different States:

- Suppose you want to assign different fulfillment costs depending on whether the state is California or not. You can modify the measure like this:

```
Cost of Fulfillment (California,Filter) =
CALCULATE(SUMX(sales,
    sales[sales]*RELATED(producthierarchy[Volume]))/750,
    FILTER(stores, stores[state]=="California")) +
CALCULATE(SUMX(sales,
    sales[sales]*RELATED(producthierarchy[Volume]))/1000,
    FILTER(stores, stores[state]<>"California"))
```

state	Sum of revenue	Cost of Fullfillemnt	Cost of Fullfillemnt (California, FILTER)
Alabama	\$5,481.68	\$31,371.18	31,371.18
Arizona	\$12,924.08	\$4,299.31	4,299.31
Arkansas	\$1,240.67	\$211.81	211.81
California	\$12,251.22	\$2,878.40	3,837.87
Colorado	\$4,412.98	\$1,188.35	1,188.35
Connecticut	\$257.42	\$104.76	104.76
Delaware	\$275.56	\$73.66	73.66
District of Columbia	\$1,685.58	\$487.02	487.02
Florida	\$11,079.09	\$2,516.09	2,516.09
Georgia	\$2,847.75	\$656.57	656.57
Hawaii	\$54.38	\$44.35	44.35
Idaho	\$84.42	\$26.43	26.43
Total	\$165,604.53	\$127,704.24	128,663.71

- **Explanation:**

- `IF` : This function checks if the state is California, assigning a value of 750 if true, and 1000 otherwise.
- `FILTER` : Filters the `Stores` table to include either California or other states.

When to Avoid Using the `FILTER` Function

While the `FILTER` function is powerful, it's not always necessary. For simpler scenarios where a basic filter condition is sufficient, directly applying filters within the `CALCULATE` function without `FILTER` is more efficient. This approach improves performance and reduces complexity.

The `FILTER` function is a valuable tool for handling complex filtering requirements in Power BI. While it may not be necessary for simple conditions, it shines in more intricate scenarios, enabling you to perform advanced calculations by creating temporary tables on the fly. In our next session, we'll build on this concept by incorporating additional logical operators to further refine our filtering techniques.

▼ Operators & Comments

In this session, we'll explore how to use logical operators in DAX formulas and the importance of proper formatting and commenting for clarity and maintainability.

Logical Operators in DAX

Logical operators are essential when you need to apply multiple conditions in your DAX calculations. Let's revisit our earlier example where we filtered data for California and then calculated values for all states except California. Now, we want to extend this by including Florida in our calculations.

1. Including Florida in the Calculation:

- To include Florida along with California, we use the logical **OR** operator (`||`). This operator allows us to specify that the condition can be true for either state.
- **Explanation:**
 - `Stores[State] = "California" || Stores[State] = "Florida"` : This condition checks if the state is either California or Florida. If either condition is true, the row is included in the calculation.

2. Excluding Both California and Florida:

- To exclude both California and Florida, we use the logical **AND** operator (`&&`). This operator ensures that both conditions must be met for a row to be excluded.
- **Explanation:**
 - `Stores[State] <> "California" && Stores[State] <> "Florida"` : This condition excludes rows where the state is either California or Florida.

```
Cost of Fulfilment (California,Filter) =
CALCULATE(SUMX(sales,
    sales[sales]*RELATED(producthierarchy[Volume])/750,
    FILTER(stores, stores[state]== "California" || stores[state]== "Florida")) +
CALCULATE(SUMX(sales,
    sales[sales]*RELATED(producthierarchy[Volume])/1000,
    FILTER(stores, stores[state]<> "California" && stores[state]<> "Florida"))
```

- Let's take a look at the results:

state	Sum of revenue	Cost of Fulfilment	Cost of Fulfilment (California,Filter)
Alabama	\$5,481.68	\$31,371.18	\$31,371.18
Arizona	\$12,924.08	\$4,299.31	\$4,299.31
Arkansas	\$1,240.67	\$211.81	\$211.81
California	\$12,251.22	\$2,878.40	\$3,837.87
Colorado	\$4,412.98	\$1,188.35	\$1,188.35
Connecticut	\$257.42	\$104.76	\$104.76
Delaware	\$275.56	\$73.66	\$73.66
District of Columbia	\$1,685.58	\$487.02	\$487.02
Florida	\$11,079.09	\$2,516.09	\$3,354.78
Georgia	\$2,847.75	\$656.57	\$656.57
Hawaii	\$54.38	\$44.35	\$44.35
Total	\$165,604.53	\$127,704.24	\$129,502.41

Importance of Formatting and Commenting

As your DAX expressions become more complex, it's crucial to format your code for readability and add comments to explain the logic. Here are some best practices:

1. Code Formatting:

- **Line Breaks:** Use line breaks after key functions like `CALCULATE` to separate the expression and filter conditions.
- **Indentation:** Indent your code to make it easier to understand the structure of the formula.
- **Spacing:** Add spaces around operators and after commas for better readability.

This approach makes the formula easier to read and maintain, especially as the logic becomes more complex.

2. Adding Comments:

- Comments are added using `//` in DAX. They provide explanations or notes about what a particular part of the code is doing.

Example:

```

1 Cost of Fullfilment (California,Filter) =
2
3 // States California and Florida
4   CALCULATE(
5     | SUMX(sales, sales[sales]*RELATED(producthierarchy[Volume])/750,
6     | FILTER(stores, stores[state]=="California" || stores[state]=="Florida"))
7
8 + CALCULATE(
9   | SUMX(sales, sales[sales]*RELATED(producthierarchy[Volume])/1000,
10  | FILTER(stores, stores[state]<>"California" && stores[state]<>"Florida"))

```

By commenting on your code, you make it easier for others (or even yourself at a later time) to understand the purpose and function of your DAX expressions.

Logical operators like `||` (OR) and `&&` (AND) are fundamental in DAX for creating complex filter conditions. Proper formatting and commenting of your DAX code are equally important for ensuring that your work is understandable and maintainable. As your formulas grow in complexity, these practices will save time and reduce errors.

▼ Project 9 : DAX Challenge

[Projects9 .rar](#)

▼ ALL

In this section, we'll explore the `ALL` function, a versatile table function in DAX (Data Analysis Expressions). While the `FILTER` function creates virtual tables by applying filter conditions, the `ALL` function works by removing filters. Its primary use is to ignore existing filters within a specified context, which is helpful in various calculations, such as determining totals or percentages unaffected by current filter selections.

Overview of the `ALL` Function

The `ALL` function "clears" filters from the context of the provided table or column. This makes it an essential tool when creating calculations that require ignoring filters applied by slicers, visuals, or other data selections in Power BI.

Example Scenario: Using `ALL` in a Measure

Let's consider a practical example where we calculate the total cost across all states, regardless of applied filters. We'll also create a percentage calculation based on this total cost.

Step 1: Creating a Measure Using `ALL`

- 1. Create a New Measure:** Right-click on the data model and select "New Measure." Name it `All Costs`.
- 2. Use the `CALCULATE` Function:** We'll use the `CALCULATE` function to modify the filter context. Inside `CALCULATE`, the expression can be another measure or calculation. In this case, we could use an existing measure like `Cost of Fulfillment` for California or other states.

```
All Costs = CALCULATE([Cost of Fullfillemnt (California, FILTER)], ALL(stores))
```

- 3. Explanation:** The `ALL(Stores)` part removes any filters applied to the `Stores` table, so the total cost for all states is returned, even if we filter the data in visuals.

state	Cost of Fulfillment (California, FILTER)	All Costs
Alabama	\$31,371.18	129,502.41
Arizona	\$4,299.31	129,502.41
Arkansas	\$211.81	129,502.41
California	\$3,837.87	129,502.41
Colorado	\$1,188.35	129,502.41
Connecticut	\$104.76	129,502.41
Delaware	\$73.66	129,502.41
District of Columbia	\$487.02	129,502.41
Florida	\$3,354.78	129,502.41
Georgia	\$656.57	129,502.41
Hawaii	\$44.35	129,502.41
Idaho	\$26.43	129,502.41
Total	\$129,502.41	129,502.41

129.50K

All Costs

Step 2: Use Case – Calculating Percentages

Now, we can use this `All Costs` measure to calculate the percentage share of the cost for a specific state compared to the total cost across all states.

- Create a Percentage Measure:** Create another measure to calculate the percentage of the cost from a specific state (e.g., Alabama) relative to the total cost.

Percentage Cost = `[Cost of Fulfillment (California, FILTER)] / [All Costs]`

- Explanation:** This formula divides the cost for the selected state (e.g., Alabama) by the total cost across all states, calculated using the `ALL` function. The result is the percentage contribution of the state's cost to the overall cost.

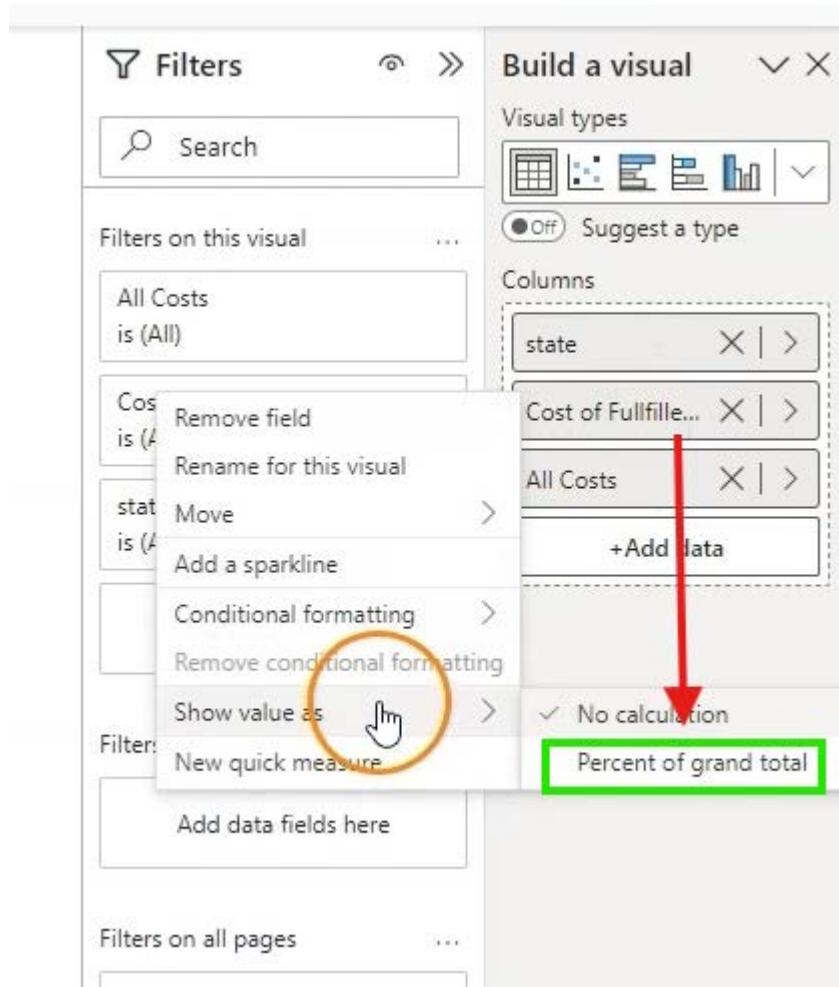
Step 3: Handling Filter Context in Visuals

When we apply filters in Power BI (e.g., selecting a few states from a slicer), most visuals update based on the selected filters. However, the `ALL` function ensures that our `All Costs` measure remains unaffected by these filters.

- This allows us to display consistent percentage values regardless of filter selections. For example, if we filter the data to show only a subset of states, the percentage calculated using `ALL` will still show the contribution relative to the total cost across all states, not just the filtered selection.

Step 4: Displaying Results in a Visual

1. **Add the Measures to a Visual:** Drag both the `Cost Percentage` and `All Costs` measures into a visual, such as a table or chart.
2. **Format the Measure:** Format the `Cost Percentage` measure to display values as percentages for better readability.
 - Right-click the measure in the visual and select "Show value as" → "Percentage of grand total."



3. **Result:** You'll see that the `Cost Percentage` remains constant regardless of any filters applied to the visual, thanks to the `ALL` function clearing the filter context.

state	Cost %	Costs	All Costs	Percentage Cost
Alabama	24.22%	\$31,371.18	129,502.41	24.22%
Arizona	3.32%	\$4,299.31	129,502.41	3.32%
Arkansas	0.16%	\$211.81	129,502.41	0.16%
California	2.96%	\$3,837.87	129,502.41	2.96%
Colorado	0.92%	\$1,188.35	129,502.41	0.92%
Connecticut	0.08%	\$104.76	129,502.41	0.08%
Delaware	0.06%	\$73.66	129,502.41	0.06%
District of Columbia	0.38%	\$487.02	129,502.41	0.38%
Florida	2.59%	\$3,354.78	129,502.41	2.59%
Georgia	0.51%	\$656.57	129,502.41	0.51%
Hawaii	0.03%	\$44.35	129,502.41	0.03%
Total	100.00%	\$129,502.41	129,502.41	100.00%

Step 5: Adding Cities

- Add Cities:** Add cities to see cost breakdown by city.

state	Percentage Cost	Costs	city
Alabama	0.10%	\$126.62	Anniston
Alabama	24.07%	\$31,176.83	Birmingham
Alabama	0.05%	\$67.74	Montgomery
Arizona	0.04%	\$46.14	Gilbert
Arizona	2.97%	\$3,852.63	Phoenix
Arizona	0.31%	\$400.54	Scottsdale
Arkansas	0.16%	\$211.81	Hot Springs National Park
California	0.01%	\$15.60	Concord
California	0.15%	\$189.17	Hayward
California	0.20%	\$255.41	Irvine
California	0.40%	\$523.65	Los Angeles
Total	100.00%	\$129,502.41	

- Observe:** Observe that the Percentage Cost has changed. Now, only the percentage of cities is calculated.

3. **Create Percentage Cost by State:** This measure calculates the ratio of the cost of a particular state to the total cost across all states. The **ALL** function removes city-based filters so that an accurate percentage can be calculated considering the cost across all cities.

```
Percentage Cost by State = CALCULATE([Cost of Fullfillemnt (California, FILTER)], ALL(stores[city])) / [All Costs]
```

4. **Result:** You'll see the cost percentage for all of Alabama. You can also see individual costs for cities.

state	Percentage Cost by State	Percentage Cost	Costs	city
Alabama	24.22%	0.10%	\$126.62	Anniston
Alabama	24.22%	24.07%	\$31,176.83	Birmingham
Alabama	24.22%	0.05%	\$67.74	Montgomery
Arizona	3.32%	0.04%	\$46.14	Gilbert
Arizona	3.32%	2.97%	\$3,852.63	Phoenix
Arizona	3.32%	0.31%	\$400.54	Scottsdale
Arkansas	0.16%	0.16%	\$211.81	Hot Springs National Park
California	2.96%	0.01%	\$15.60	Concord
California	2.96%	0.15%	\$189.17	Hayward
California	2.96%	0.20%	\$255.41	Irvine
California	2.96%	0.40%	\$523.65	Los Angeles
California	2.96%	0.08%	\$109.45	Oakland
Total	100.00%	100.00%	\$129,502.41	

Conclusion

The **ALL** function is a powerful tool for controlling filter contexts in DAX. In this example, it allows us to calculate values like total costs across all data and percentages that remain consistent even when filters are applied. This functionality is particularly useful when you need to perform calculations that should not be influenced by user selections or slicers, such as calculating percentages of a grand total.

By combining **ALL** with the **CALCULATE** function, you can build robust measures that deliver consistent and accurate results in Power BI, regardless of the filters applied in the report.

▼ ALLEXCEPT

The **ALLEXCEPT** function in DAX is a variation of the **ALL** function, with a more focused application. While the **ALL** function removes all filters from a

table or column, **ALLEXCEPT** removes all filters except for those applied to specific columns that you choose to retain. This makes the function particularly useful when you need to clear the filter context from multiple columns but keep the context for one or more specific columns.

Purpose of Using ALLEXCEPT

- Let's say we added store_id to our table. We see that Percentage Cost by State is broken again. You will see the use of ALLEXCEPT to prevent this

state	Percentage Cost by State	Percentage Cost	Costs	city	store_id
Alabama	0.10%	0.10%	\$126.62	Anniston	S0093
Alabama	23.76%	23.76%	\$30,767.52	Birmingham	S0043
Alabama	0.05%	0.05%	\$64.56	Birmingham	S0111
Alabama	0.27%	0.27%	\$344.74	Birmingham	S0112
Alabama	0.05%	0.05%	\$67.74	Montgomery	S0068
Arizona	0.04%	0.04%	\$46.14	Gilbert	S0143
Arizona	0.35%	0.35%	\$456.32	Phoenix	S0022
Arizona	1.57%	1.57%	\$2,036.26	Phoenix	S0031
Arizona	0.74%	0.74%	\$962.92	Phoenix	S0051
Arizona	0.24%	0.24%	\$305.33	Phoenix	S0108
Arizona	0.07%	0.07%	\$91.81	Phoenix	S0128
Arizona	0.31%	0.31%	\$400.54	Scottsdale	S0010
Total	100.00%	100.00%	\$129.502.41		

Understanding the ALLEXCEPT Function

The **ALLEXCEPT** function works by removing filters from all columns in a table, except for those that are explicitly specified. This allows for more control over the filter context, especially when you are dealing with tables containing many columns and you want to retain only a few of them for filtering.

Syntax

The syntax for the **ALLEXCEPT** function is straightforward:

```
ALLEXCEPT(Table, Column1, Column2, ...)
```

- Table:** The table from which the filters are removed.

- **Column1, Column2, ...:** The columns that should *not* have their filters removed.

Use Case: Calculating Percentage Cost by State

Let's walk through a practical use case to demonstrate the ALLEXCEPT function. Suppose you want to calculate the percentage of total costs for each state, but you want to ignore any filters applied to other columns, such as `City` or `Store_ID`. In this case, the ALLEXCEPT function can be used to clear all filters except for those on the `State` column.

1. Creating the Measure:

- Right-click on your table in Power BI and select **New Measure**.
 - Name the measure, for example, **Percentage Cost by State**.
 - Use the **CALCULATE** function along with **ALLEXCEPT** to modify the filter context.

```
1 Percentage Cost by State = CALCULATE([Cost of Fullfillemnt (California, FILTER)],  
2 | ALLEXCEPT(stores, stores[state])) / [All Costs]
```

In this measure, the **ALLEXCEPT** function removes all filters from the `'Stores'` table, except for the filters applied to the `State` column. This ensures that the calculation only respects the `State` filter, while ignoring any filters on `City`, `Store_ID`, or other columns.

2. Benefits of Using ALLEXCEPT:

- **Efficiency:** Instead of manually specifying each column you want to ignore in the filter context, the ALLEXCEPT function makes it easier by allowing you to specify only the columns you want to retain.
 - **Flexibility:** Even if more columns are added to your table later, the ALLEXCEPT function will continue to work without needing manual adjustments, as long as the specified columns (e.g., `State`) remain unchanged.

Practical Example: Avoiding Filter Breaks

Consider a scenario where you manually add filters to individual columns like `City` and `Store_ID`. Initially, you might try removing these filters one by one using the **ALL** function. However, if you later add more columns (e.g., `City_ID`), you would have to update your measure to account for each new column. This can quickly become cumbersome and prone to errors. By using **ALLEXCEPT**, you simplify the process by only retaining the filter on the `State` column, and removing everything else in one step.

state	Percentage Cost by State	Percentage Cost	Costs	city	store_id
Alabama	24.22%	0.10%	\$126.62	Anniston	S0093
Alabama	24.22%	23.76%	\$30,767.52	Birmingham	S0043
Alabama	24.22%	0.05%	\$64.56	Birmingham	S0111
Alabama	24.22%	0.27%	\$344.74	Birmingham	S0112
Alabama	24.22%	0.05%	\$67.74	Montgomery	S0068
Arizona	3.32%	0.04%	\$46.14	Gilbert	S0143
Arizona	3.32%	0.35%	\$456.32	Phoenix	S0022
Arizona	3.32%	1.57%	\$2,036.26	Phoenix	S0031
Arizona	3.32%	0.74%	\$962.92	Phoenix	S0051
Arizona	3.32%	0.24%	\$305.33	Phoenix	S0108
Arizona	3.32%	0.07%	\$91.81	Phoenix	S0128
Arizona	3.32%	0.31%	\$400.54	Scottsdale	S0010
Total	100.00%	100.00%	\$129,502.41		

Common Mistakes to Avoid

- **Unclosed Parentheses:** As with many DAX functions, forgetting to close parentheses can cause syntax errors. Ensure that every function and statement is properly closed.
- **Selecting the Correct Columns:** Ensure you are selecting the right table and columns when using the ALLEXCEPT function, as it could produce unintended results if incorrect columns are excluded from the filter context.

Conclusion

The **ALLEXCEPT** function is a powerful alternative to **ALL** when you need more granular control over your filter context. By removing all filters except for those applied to specific columns, you can achieve cleaner, more efficient calculations without manually specifying each column to ignore. This function is especially useful in complex models where multiple columns

may have filters applied, but only a few should be considered in your calculations.

Utilizing **ALLEXCEPT** can help you build more robust and flexible measures, particularly when calculating percentage-based metrics or other analyses that rely on selective filtering.

▼ ALLSELECTED

DAX - ALLSELECTED Function: Overview and Use Case

In this section, we'll explore the **ALLSELECTED** function in DAX, which is part of the "ALL" family of functions. The **ALLSELECTED** function is particularly useful when you want to remove filters within a specific visual in Power BI while still respecting filters applied outside of the visual.

Understanding the ALLSELECTED Function

The **ALLSELECTED** function works by removing filters only from the current visual, allowing external filters (those applied outside the visual) to remain in effect. This makes it ideal for scenarios where you want to manipulate data within a visual, such as showing percentages that sum up to 100%, but still account for external filtering from slicers or report-level filters.

Key Difference: ALL vs. ALLSELECTED

While the **ALL** function removes all filters regardless of their origin (both inside and outside the visual), **ALLSELECTED** selectively removes filters only within the current visual. This means filters applied via slicers or external filters remain in effect, while row-level filters within the visual are ignored.

Use Case: Calculating Percentage Cost by Selected States

Let's consider a scenario where we want to calculate the percentage of total costs for selected states (e.g., California, Colorado, and Alabama) and ensure the total sums to 100%, but without removing any external filters applied outside the visual (such as slicers).

1. Creating the Measure:

- Right-click on your table in Power BI and select **New Measure**.
- Name the measure, for example, **Percentage Cost by Selected States**.
- Use the **CALCULATE** function along with **ALLSELECTED** to manipulate the filter context.

All Costs = `CALCULATE([Cost of Fulfillment (California, FILTER)], ALLSELECTED(stores))`

In this measure, the **ALLSELECTED** function removes any filters applied within the visual itself, but still allows external filters (like slicers) to affect the results. This allows the measure to always return the total cost of fulfillment for the selected states within the visual, ensuring that the sum of percentages reaches 100%.

2. Visualizing the Results:

- Drag this measure into a visual that shows the cost for each state.
- When you apply slicers or report-level filters (for instance, filtering by a specific region), the total cost within the visual will still respect these external filters.
- Each state within the visual will show its contribution to the total selected cost, and the sum will always be 100%.

state	Percentage Cost by State	Percentage Cost	Costs	city	store_id
Alabama	88.57%	0.36%	\$126.62	Anniston	S0093
Alabama	88.57%	86.86%	\$30,767.52	Birmingham	S0043
Alabama	88.57%	0.18%	\$64.56	Birmingham	S0111
Alabama	88.57%	0.97%	\$344.74	Birmingham	S0112
Alabama	88.57%	0.19%	\$67.74	Montgomery	S0068
Arkansas	0.60%	0.60%	\$211.81	Hot Springs National Park	S0091
California	10.84%	0.04%	\$15.60	Concord	S0047
California	10.84%	0.53%	\$189.17	Hayward	S0072
California	10.84%	0.72%	\$255.41	Irvine	S0058
California	10.84%	0.53%	\$187.60	Los Angeles	S0032
California	10.84%	0.95%	\$336.05	Los Angeles	S0117
California	10.84%	0.31%	\$109.45	Oakland	S0140
Total	100.00%	100.00%	\$35,420.86		

Key Benefits of ALLSELECTED

- **Control Over Filter Context:** Allows for precise control over which filters to ignore, maintaining external filter contexts while ignoring internal ones.
- **Ideal for Dynamic Totals:** Helps create dynamic totals that adjust based on slicer selections, while still providing consistent calculations inside visuals.
- **Flexibility in Percentages:** Particularly useful in percentage calculations where the sum must always equal 100%, even when external filters are applied.

Conclusion

The **ALLSELECTED** function provides a fine level of control over the filter context within Power BI visuals. By using **ALLSELECTED**, you can ensure that calculations within a visual ignore internal filters but respect any external filters applied through slicers or other report-level filters. This makes it a powerful tool for creating dynamic, context-aware metrics such as percentages, where the total must always sum to 100%, even when external filters are in place.

Understanding how and when to use **ALLSELECTED** will enhance your ability to build accurate, flexible measures in Power BI that can dynamically respond to user selections.

▼ 9- DAX - Time Intelligence

▼ DATEADD

In this section, we will explore the **DATEADD** function, which is part of DAX's **Time Intelligence** functions. These functions allow for advanced analysis of data over time, making it possible to perform comparisons across different time periods, such as year-over-year or month-over-month comparisons. The **DATEADD** function is particularly useful for shifting dates by a specified interval (e.g., moving forward or backward by days, months, quarters, or years).

What Does the DATEADD Function Do?

The **DATEADD** function shifts the dates in a given column by a specified interval. This is particularly useful when you want to compare data from different time periods, such as comparing the performance of the current year to the previous year. The function modifies the filter context to move the date range and retrieve data for the new date range.

Syntax of DATEADD

The basic syntax of **DATEADD** is as follows:

```
DATEADD(<dates>, <number_of_intervals>, <interval>)
```

- **dates**: A column containing dates (e.g., 'Sales'[OrderDate]).
- **number_of_intervals**: The number of intervals to shift the date by. Use negative values to go backwards and positive values to go forwards.
- **interval**: The unit of time to shift by, which can be **DAY**, **MONTH**, **QUARTER**, or **YEAR**.

Use Case: Comparing Year-Over-Year Logistics Costs

Let's walk through an example where we want to compare **logistics costs** for the current year with the costs from the **previous year**.

1. Create a New Measure:

- Right-click on your table in Power BI and select **New Measure**.
- Name the measure, for example, **Previous Year Cost of Logistics**.
- Use the **CALCULATE** function, which allows us to modify the filter context. The **DATEADD** function will shift the date context back by one year.
-

Previous Year Cost of Logistics =

```
| CALCULATE([Cost of Logistics],  
| | | | DATEADD('Date Dimension'[order_date], -1, YEAR))
```

In this measure, the **DATEADD** function shifts the '**Date Dimension**'[**order_date**] column back by one year (**-1, YEAR**). The **CALCULATE** function then computes the **Cost of Logistics** for the previous year based on this shifted context.

2. Visualizing the Results:

- Add this new measure to a visual alongside your current **Cost of Logistics** measure.
- Now you will see a side-by-side comparison of the logistics costs for the current year and the previous year, allowing you to analyze trends and changes year-over-year.

Year	Quarter	Month	Day	Cost of Logistics	Previous Year Cost of Logistics
2017	Qtr 1	January	2	\$10.72	
2017	Qtr 1	January	3	\$20.78	
2017	Qtr 1	January	4	\$33.70	
2017	Qtr 1	January	5	\$16.49	
2017	Qtr 1	January	6	\$8.76	
2017	Qtr 1	January	7	\$13.13	
2017	Qtr 1	January	8	\$3.12	
2017	Qtr 1	January	9	\$21.09	
2017	Qtr 1	January	10	\$10.08	
2017	Qtr 1	January	11	\$72.24	
2017	Qtr 1	January	12	\$57.46	
Total				\$129,502.41	102,891.28

Year	Quarter	Month	Day	Cost of Logistics	Previous Year Cost of Logistics
2017	Qtr 4	December	31	\$41.22	
2018	Qtr 1	January	1	\$11.69	
2018	Qtr 1	January	2	\$32.89	10.72
2018	Qtr 1	January	3	\$15.49	20.78
2018	Qtr 1	January	4	\$12.59	33.70
2018	Qtr 1	January	5	\$44.61	16.49
2018	Qtr 1	January	6	\$12.73	8.76
2018	Qtr 1	January	7	\$125.54	13.13
2018	Qtr 1	January	8	\$33.96	3.12
2018	Qtr 1	January	9	\$46.33	21.09
2018	Qtr 1	January	10	\$6.11	10.08
Total				\$129,502.41	102,891.28

Flexibility of the DATEADD Function

The **DATEADD** function offers great flexibility. For example, you can easily change the interval from **YEAR** to **MONTH** or **QUARTER** to adjust your comparison period. Let's assume the customer now wants to see a comparison of **monthly** logistics costs instead of yearly:

1. Modify the Measure for Monthly Comparison:

Previous Year Cost of Logistics =

```
CALCULATE([Cost of Logistics],
    DATEADD('Date Dimension'[order_date], -1, MONTH))
```

Year	Quarter	Month	Cost of Logistics	Previous Year Cost of Logistics
2017	Qtr 1	January	\$741.67	
2017	Qtr 1	February	\$696.26	741.67
2017	Qtr 1	March	\$3,923.95	696.26
2017	Qtr 2	April	\$685.67	3,923.95
2017	Qtr 2	May	\$7,043.41	685.67
2017	Qtr 2	June	\$18,213.66	7,043.41
2017	Qtr 3	July	\$838.73	18,213.66
2017	Qtr 3	August	\$12,096.88	838.73
2017	Qtr 3	September	\$935.13	12,096.88
2017	Qtr 4	October	\$748.44	935.13
2017	Qtr 4	November	\$618.82	748.44
Total			\$129,502.41	129,502.41

Here, the interval is set to **MONTH**, shifting the date context back by one month. This allows for a month-over-month comparison of logistics costs.

Advanced Use Case: Calculating Percentage Change Month-Over-Month

Now, let's extend the example by calculating the **percentage change** in logistics costs between the current month and the previous month. This is a common metric used in business reporting to track trends.

1. Create a Measure for Monthly Percentage Change:

- First, we need to calculate the difference between the current month's cost and the previous month's cost.
- Then, divide the difference by the previous month's cost to get the percentage change.

```
. Monthly Change of Cost =  
([Cost of Logistics] - [Previous Monthly Cost of Logistics])  
/[Previous Monthly Cost of Logistics]
```

Year	Quarter	Month	Cost of Logistics	Previous Monthly Cost of Logistics	Monthly Change of Cost
2017	Qtr 1	January	\$741.67		Infinity
2017	Qtr 1	February	\$696.26	741.67	-6.12%
2017	Qtr 1	March	\$3,923.95	696.26	463.57%
2017	Qtr 2	April	\$685.67	3,923.95	-82.53%
2017	Qtr 2	May	\$7,043.41	685.67	927.23%
2017	Qtr 2	June	\$18,213.66	7,043.41	158.59%
2017	Qtr 3	July	\$838.73	18,213.66	-95.40%
2017	Qtr 3	August	\$12,096.88	838.73	1342.29%
2017	Qtr 3	September	\$935.13	12,096.88	-92.27%
2017	Qtr 4	October	\$748.44	935.13	-19.96%
2017	Qtr 4	November	\$618.82	748.44	-17.32%
Total			\$129,502.41	129,502.41	0.00%

2. Visualize the Results:

- Drag this new measure into your table or chart visual.
- Format the measure as a percentage for easier interpretation.

This measure will now display the month-over-month percentage change in logistics costs. You can track if costs are increasing or decreasing from month to month and by what percentage.

Practical Insights

- **Day-to-Day Comparison:** You can also adjust the interval to **DAY** to track daily changes in costs, sales, or other metrics.
- **Quarterly Analysis:** By setting the interval to **QUARTER**, you can compare quarterly performance (e.g., comparing Q1 to Q4 of the previous year).
- **Year-to-Date and Month-to-Date:** With the **DATEADD** function, it's possible to calculate year-to-date (YTD) and month-to-date (MTD) values by using additional DAX functions like **TOTALYTD** or **TOTALMTD**.

Conclusion

The **DATEADD** function is a versatile and powerful tool for performing time-based comparisons in Power BI. It allows you to shift the date context easily, making it possible to compare current data with data from past periods, whether it be year-over-year, month-over-month, or even day-to-day. By integrating this with other DAX functions like **CALCULATE**, you can create dynamic and insightful reports that provide a clear view of how your data changes over time.

▼ Year-to-Date & Month-to-Date

In Power BI, **Time Intelligence** is a crucial component for analyzing data over specific time periods, such as calculating **Year-to-Date (YTD)** or **Month-to-Date (MTD)** values. These calculations help track cumulative metrics, such as revenue or costs, over a period of time. This section will focus on creating running totals using DAX's built-in **YEAR-TO-DATE** and **MONTH-TO-DATE** functions.

What is Year-to-Date (YTD)?

A **Year-to-Date (YTD)** calculation shows the running total of a metric (such as revenue or costs) from the beginning of the year to a specific date. For example, if you're tracking costs for the current year, YTD will sum all costs up to the given day in the year.

Creating a Year-to-Date (YTD) Measure

Let's go through the process of creating a YTD measure to track the cumulative **Cost of Logistics** for a given year.

1. Set Up the Visual:

- First, modify your table visual to display only the **Cost of Logistics** and the **Order Date**.
- Remove any unnecessary columns, and ensure the **Order Date** comes from the **Date Dimension**.

2. Create the YTD Measure:

- Right-click on the **Measures Table** and select **New Measure**.
- Name the new measure something like **YTD Cost of Logistics**.

Here's the formula using DAX:

```
YTD Cost =
```

```
CALCULATE( [Cost of Logistics], DATESYTD('Date Dimension'[order_date]))
```

order_date	Cost of Logistics	YTD Cost
Monday, January 02, 2017	\$0.85	\$0.85
Tuesday, January 03, 2017	\$4.53	\$5.38
Wednesday, January 04, 2017	\$0.30	\$5.67
Thursday, January 05, 2017	\$0.00	\$5.67
Friday, January 06, 2017	\$1.31	\$6.98
Saturday, January 07, 2017	\$1.47	\$8.46
Sunday, January 08, 2017	\$0.00	\$8.46
Monday, January 09, 2017	\$7.26	\$15.72
Tuesday, January 10, 2017	\$2.95	\$18.66
Total	\$39,824.93	\$5,412.39



In this formula:

- The **CALCULATE** function modifies the filter context to apply a time-based calculation.
- The **DATESYTD** function calculates the year-to-date values by summing up the **Cost of Logistics** from the beginning of the year up to the specified date.

3. Optional Argument: Custom Year-End Date

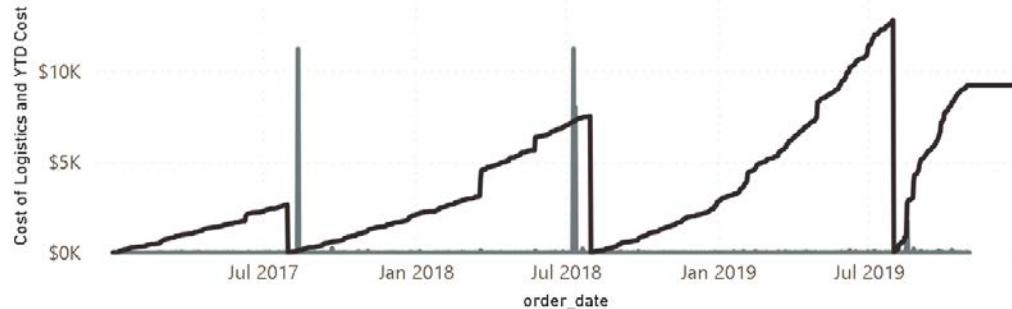
- By default, the year ends on **December 31st**. However, you can specify a custom year-end date (e.g., July 31st). To do this, use the optional argument in the **DATESYTD** function:

YTD Cost =

`CALCULATE(SUM(sales[revenue]), DATESYTD('Date Dimension'[order_date],"31-07"))`

Cost of Logistics and YTD Cost by order_date

●Cost of Logistics ●YTD Cost



This formula treats **August 1st** as the beginning of a new year, which can be useful for businesses that have a fiscal year starting in a month other than January.

4. Visualizing the YTD Measure:

- Drag the **YTD Cost of Logistics** measure into your visual.
- The result will show the cumulative costs up to each day within the year.

You can also switch to a **Line Chart** to visualize how costs accumulate over time. For example, the chart might show spikes in certain months, indicating high activity, while the running total continues to build.

What is Month-to-Date (MTD)?

Similarly, **Month-to-Date (MTD)** shows the running total of a metric from the beginning of the month to a specific date. It's useful for tracking how metrics like revenue or costs are trending within the current month.

Creating a Month-to-Date (MTD) Measure

To calculate **MTD** values for the same **Cost of Logistics**, follow the same process but use the **DATESMTD** function.

1. Create the MTD Measure:

YTD Cost =

`CALCULATE(SUM(sales[revenue]), DATESMTD('Date Dimension'[order_date]))`

Cost of Logistics and YTD Cost by order_date

●Cost of Logistics ●YTD Cost



This formula tracks the total **Cost of Logistics** from the start of the current month to the specified date.

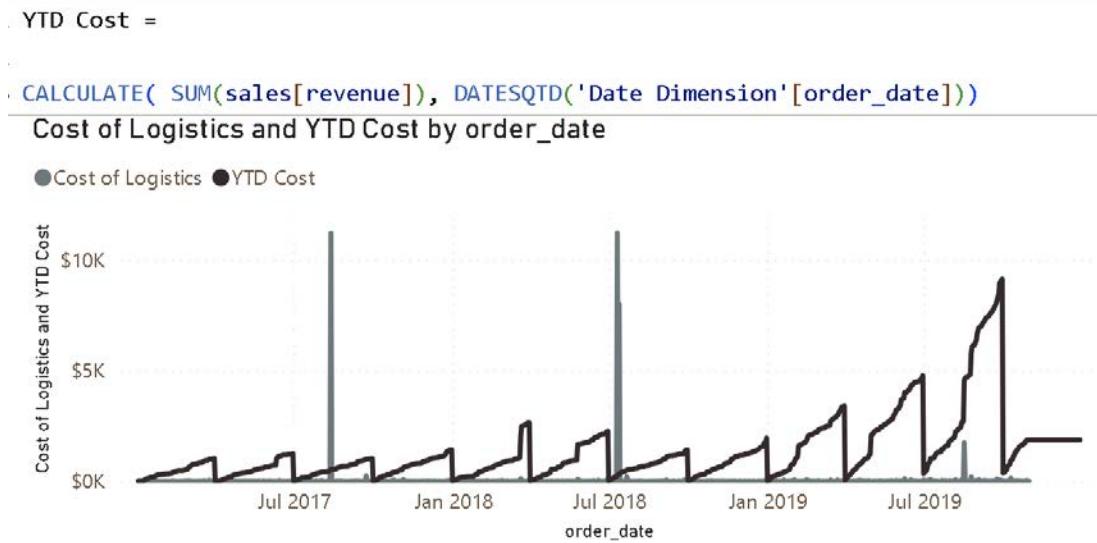
2. Visualizing the MTD Measure:

- Drag the **MTD Cost of Logistics** measure into your visual.
- The table or chart will now display the running total of costs within the current month, resetting at the start of each month.

Customizing with Quarter-to-Date (QTD)

If you want to track cumulative totals for a quarter, use the **DATESQTD** function. The process is similar to YTD and MTD, but it calculates from the beginning of the quarter.

1. Create the QTD Measure:



This formula provides the cumulative total for the current quarter.

Practical Example: Comparing YTD Revenue

You can apply the same approach to **Revenue** instead of costs. Simply adjust the measure to calculate the **YTD Revenue**.

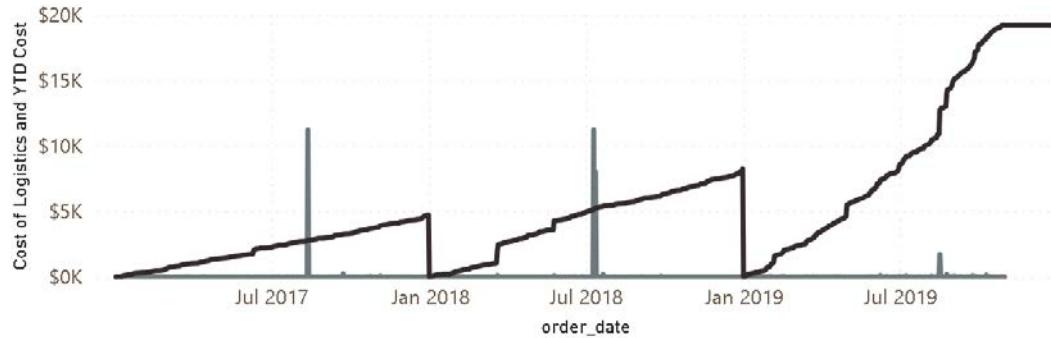
1. Create a YTD Revenue Measure:

YTD Cost =

`CALCULATE(SUM(sales[revenue]), DATESYTD('Date Dimension'[order_date]))`

Cost of Logistics and YTD Cost by order_date

●Cost of Logistics ●YTD Cost



This will sum up the **Revenue** from the start of the year until the selected date.

Additional Insights

- **Interactive Filtering:** Power BI's visuals allow you to dynamically filter by month, quarter, or year, and the YTD/MTD measures will update automatically. For example, switching between a monthly or yearly view will change how the running total is displayed.
- **Running Totals with Custom Date Hierarchies:** If you remove or modify the date hierarchy in the visual, the YTD, QTD, and MTD calculations will still work correctly. This provides flexibility in how the running totals are displayed.

Conclusion

The **Year-to-Date (YTD)** and **Month-to-Date (MTD)** functions in DAX allow you to calculate cumulative totals for important business metrics, such as revenue or costs. These functions provide powerful insights into how your metrics are accumulating over specific time periods, helping you track progress, performance, and trends throughout the year or month.

▼ FORMAT

In this section, we'll explore the **FORMAT** function in DAX, which allows you to customize the display of date, time, and numerical values in Power BI.

The **FORMAT** function is particularly useful when standard formats do not meet the specific requirements of your business or users. This function gives you control over how values are presented, offering flexibility in formatting dates, numbers, currencies, and percentages.

Understanding the FORMAT Function

The **FORMAT** function in DAX can be used to change the appearance of values in your reports without altering the underlying data. It works with both dates and numerical values and can also accommodate different locales, providing localized formats based on your audience's language or region.

Syntax of the FORMAT Function

The basic syntax of **FORMAT** is:

```
FORMAT(<value>, <format_string>, [<locale_name>])
```

- **value**: The column or measure you want to format.
- **format_string**: A predefined or custom string that dictates how the value should be displayed.
- **locale_name (optional)**: Specifies the locale used for formatting. If not specified, the default locale is used.

Example 1: Basic Date Formatting

Let's say your customer wants to display a date without the year, only showing the weekday, day, and month. While Power BI offers standard formats like **short date** or **long date**, the **FORMAT** function allows you to define a custom format.

- Short Date

```
Dates Custom = FORMAT('Date Dimension'[order_date], "Short Date")
```

order_date	Dates Custom
Monday, January 02, 2017	1/2/2017
Tuesday, January 03, 2017	1/3/2017
Wednesday, January 04, 2017	1/4/2017
Thursday, January 05, 2017	1/5/2017
Friday, January 06, 2017	1/6/2017
Saturday, January 07, 2017	1/7/2017
Sunday, January 08, 2017	1/8/2017
Monday, January 09, 2017	1/9/2017
Tuesday, January 10, 2017	1/10/2017
Wednesday, January 11, 2017	1/11/2017

- Long Date

```
Dates Custom = FORMAT('Date Dimension'[order_date], "Long Date")
```

order_date	Dates Custom
Monday, January 02, 2017	Monday, January 2, 2017
Tuesday, January 03, 2017	Tuesday, January 3, 2017
Wednesday, January 04, 2017	Wednesday, January 4, 2017
Thursday, January 05, 2017	Thursday, January 5, 2017
Friday, January 06, 2017	Friday, January 6, 2017
Saturday, January 07, 2017	Saturday, January 7, 2017
Sunday, January 08, 2017	Sunday, January 8, 2017
Monday, January 09, 2017	Monday, January 9, 2017
Tuesday, January 10, 2017	Tuesday, January 10, 2017
Wednesday, January 11, 2017	Wednesday, January 11, 2017

1. Creating a Custom Date Column:

- Start by creating a new calculated column. Right-click on the date dimension and select **New Column**.
- Name the column **Custom Date**.

Here's how to write the DAX formula using the **FORMAT** function:

```
Dates Custom = FORMAT('Date Dimension'[order_date], "dddd")
```

order_date	Dates Custom
Monday, January 02, 2017	Monday
Tuesday, January 03, 2017	Tuesday
Wednesday, January 04, 2017	Wednesday
Thursday, January 05, 2017	Thursday
Friday, January 06, 2017	Friday
Saturday, January 07, 2017	Saturday
Sunday, January 08, 2017	Sunday
Monday, January 09, 2017	Monday
Tuesday, January 10, 2017	Tuesday
Wednesday, January 11, 2017	Wednesday

```
Dates Custom = FORMAT('Date Dimension'[order_date], "dddd, mmmm d")
```

order_date	Dates Custom
Monday, January 02, 2017	Monday, January 2
Tuesday, January 03, 2017	Tuesday, January 3
Wednesday, January 04, 2017	Wednesday, January 4
Thursday, January 05, 2017	Thursday, January 5
Friday, January 06, 2017	Friday, January 6
Saturday, January 07, 2017	Saturday, January 7
Sunday, January 08, 2017	Sunday, January 8
Monday, January 09, 2017	Monday, January 9
Tuesday, January 10, 2017	Tuesday, January 10
Wednesday, January 11, 2017	Wednesday, January 11

- In this example:
 - `dddd` returns the full name of the weekday (e.g., "Monday").
 - `MMMM` returns the full name of the month (e.g., "January").
 - `dd` returns the day of the month with a leading zero if necessary (e.g., "01" or "25").

2. Result:

After you hit

Enter, the new **Custom Date** column will display dates in the format:
Monday, January 01.

Example 2: Custom Locale for Date Formatting

If your customer requires the date to be displayed in a specific language, you can add a locale to the **FORMAT** function. For instance, displaying the date in French:

```
Custom Date (FR) = FORMAT('Date'[OrderDate], "dddd, MMMM dd", "fr-FR")
```

This will display the date in French, where "Monday, January 01" becomes **Lundi, Janvier 01.**

▼ Example 3: Formatting Numbers

The **FORMAT** function can also be used to format numerical values, adding thousands separators, decimal places, or even displaying values as percentages.

1. Formatting Numbers with Thousands Separator:

```
DAXCopy code  
Formatted Sales = FORMAT(Sales[Amount], "#,##0")
```

- The `#,##0` format string adds commas to separate thousands and ensures no decimal places are displayed.
- For example, 12345 will be formatted as **12,345.**

2. Formatting Numbers as Currency:

```
DAXCopy code  
Formatted Sales as Currency = FORMAT(Sales[Amount], "$#,##0.00")
```

- This format string displays values as currency with two decimal places (e.g., **\$12,345.00**).

3. Formatting Percentages:

DAXCopy code

```
Formatted Percentage = FORMAT([Growth], "0.00%")
```

- If the growth rate is 0.0456, it will be displayed as **4.56%**.

Custom Formatting with Special Characters

You can also include special characters or symbols in your custom format. For example, if you want to display the day of the week followed by the word "Day" and a colon:

DAXCopy code

```
Formatted Date with Text = FORMAT('Date'[OrderDate],  
"dddd 'Day:' MMMM dd")
```

In this case, anything within single quotes (`'...'`) is treated as a literal and displayed as-is. This would result in **Monday Day: January 01**.

Handling Locale-Sensitive Formats

The **FORMAT** function automatically adapts to the locale you specify, changing the way numbers and dates are displayed based on local conventions. For example, in Germany, the format `#,##0.00` would display a number as **12.345,00**, using a comma for the decimal separator and periods for the thousands separator.

Important Considerations

- **FORMAT Returns Text:** It's important to note that the **FORMAT** function converts the value to a text string. Once a value is formatted using this function, it is no longer recognized as a number or date for further

calculations. This means you cannot apply aggregations like sums or averages to a **formatted** column.

- **Performance Impact:** While **FORMAT** offers powerful customization, using it on large datasets or in complex calculations can affect performance. Use it judiciously and only when necessary.

Conclusion

The **FORMAT** function in DAX is a versatile tool that allows you to customize the presentation of dates, numbers, and other values in Power BI. By leveraging **predefined formats**, **custom formats**, and **locale support**, you can meet the specific display needs of your audience, whether it's adapting to different languages or following specific business conventions.

This function is crucial when you need full control over how data is presented, but remember that **FORMAT** outputs text, so it should be used primarily for display purposes rather than further calculations.

▼ Complete Date Table

In this lecture, we will walk through creating a **complete date table** in Power BI, which includes a variety of **date-related features** that are essential for time-based analysis. Date tables are crucial for understanding patterns in data, such as sales trends across weeks, months, quarters, and years. A well-structured date table enhances reporting and analytics by allowing you to slice and analyze data efficiently.

Why Do You Need a Complete Date Table?

A date table contains all possible dates over a given period and includes additional fields such as day of the week, month, quarter, year, and other useful date-related attributes. This is essential because many business metrics, like revenue or costs, depend on time-related features for comparison. For example:

- **Year-to-date** comparisons.
- **Quarterly** analysis of sales.
- Identifying trends across **specific days** or **weeks**.

Without a complete date table, missing dates (for example, when no sales are made on certain days) might result in incomplete analysis.

Step 1: Create the Date Table Using the CALENDAR Function

We'll start by creating a basic date table that includes all dates between the minimum and maximum dates in your dataset. This can be done using the **CALENDAR** function in DAX, which generates a table of dates between two specified boundaries.

1. Creating the Table:

- In Power BI, go to the **Table Tools** tab and select **New Table**.
- Name this table something like **DateTable**.

Here's the DAX code to create the date range:

```
DataTable = CALENDAR(MIN('Date Dimension'[order_date]), MAX('Date Dimension'[order_date]))
```

- The **CALENDAR** function generates a date column that includes every date between the **first order date** (`MIN('Sales'[OrderDate])`) and the **last order date** (`MAX('Sales'[OrderDate])`).
- This ensures that there are no gaps in the dates, even on days where no orders were placed.

Date	
1/2/2017 12:00:00 AM	
1/3/2017 12:00:00 AM	
1/4/2017 12:00:00 AM	
1/5/2017 12:00:00 AM	
1/6/2017 12:00:00 AM	
1/7/2017 12:00:00 AM	
1/8/2017 12:00:00 AM	
1/9/2017 12:00:00 AM	
1/10/2017 12:00:00 AM	
1/11/2017 12:00:00 AM	
1/12/2017 12:00:00 AM	

Step 2: Add Date-Related Features Using ADDCOLUMNS

Once the basic date table is created, we can enhance it by adding more columns that provide useful date attributes. For this, we'll use the **ADDCOLUMNS** function to extend the table with additional columns like **year**, **month**, **week number**, **quarter**, and others.

1. Using ADDCOLUMNS:

- To extend your date table, replace the basic **CALENDAR** function with **ADDCOLUMNS**, which allows us to add multiple columns dynamically.

Here's an example of how to extend the date table:

The screenshot shows the Power BI Data View interface. At the top, there is a DAX code editor window containing the following DAX script:

```

1 DateTable =
2 ADDCOLUMNS(
3 CALENDAR(MIN('Date Dimension'[order_date]), MAX('Date Dimension'[order_date])),
4 "DateAsInteger", FORMAT([Date],"YYYYMMDD"),
5 "Year", YEAR( [date] ), "MonthNo", FORMAT( [date], "MM" ),
6 "YearMonthNo", FORMAT( [date], "YYYY/MM" ),
7 "YearMonth", FORMAT( [date], "YYYY/mmm" ),
8 "MonthShort", FORMAT( [date], "mmm" ),
9 "MonthLong", FORMAT( [date], "mmmm" ),
10 "WeekNo", WEEKDAY( [date] ),
11 "WeekDay", FORMAT( [date], "dddd" ),
12 "WeekDayShort", FORMAT( [date], "ddd" ),
13 "Quarter", "Q" & FORMAT( [date], "Q" ),
14 "YearQuarter", FORMAT( [date], "YYYY" ) & "/" & FORMAT( [date], "Q" ))

```

Below the code editor is a preview table with the following data:

Date	DateAsInteger	Year	MonthNo	YearMonthNo	YearMonth	MonthShort	MonthLong	WeekNo	WeekDay	WeekDayShort	Quarter	YearQuarter
7/1/2018 12:00:00 AM	20180701	2018	07	2018/07	2018/Jul	Jul	July	1	Sunday	Sun	Q3	2018/Q3
7/2/2018 12:00:00 AM	20180702	2018	07	2018/07	2018/Jul	Jul	July	2	Monday	Mon	Q3	2018/Q3
7/3/2018 12:00:00 AM	20180703	2018	07	2018/07	2018/Jul	Jul	July	3	Tuesday	Tue	Q3	2018/Q3
7/4/2018 12:00:00 AM	20180704	2018	07	2018/07	2018/Jul	Jul	July	4	Wednesday	Wed	Q3	2018/Q3
7/5/2018 12:00:00 AM	20180705	2018	07	2018/07	2018/Jul	Jul	July	5	Thursday	Thu	Q3	2018/Q3

- **YEAR([Date]):** Extracts the year from the date.
- **MONTH([Date]):** Extracts the month as a number.
- **FORMAT([Date], "MMMM"):** Returns the full month name (e.g., "January").
- **QUARTER([Date]):** Extracts the quarter number (1-4).
- **WEEKNUM([Date]):** Returns the week number of the year.
- **WEEKDAY([Date], 2):** Returns the day of the week (with Monday as day 1).
- **FORMAT([Date], "dddd"):** Returns the full day name (e.g., "Monday").

Step 3: Include Additional Custom Columns

You can keep adding more custom columns to your date table depending on your needs. For example, you might want to create an **integer representation** of the date or additional flags for fiscal year calculations.

Here are some more advanced columns you could add:

```

DAXCopy code
DateTable = ADDCOLUMNS(
    CALENDAR(MIN('Sales'[OrderDate]), MAX('Sales'[OrderDa
    te])),
```

```

    "Year", YEAR([Date]),
    "Month Number", MONTH([Date]),
    "Month Name", FORMAT([Date], "MMMM"),
    "Quarter", QUARTER([Date]),
    "Week Number", WEEKNUM([Date]),
    "Day of Week", WEEKDAY([Date], 2),
    "Day Name", FORMAT([Date], "dddd"),
    "DateAsInteger", FORMAT([Date], "YYYYMMDD"), -- Date
as an integer (YYYYMMDD)
    "IsWeekend", IF(WEEKDAY([Date], 2) > 5, TRUE(), FALSE
())
-- Flag for weekends
)

```

- **DateAsInteger**: This formats the date as an integer in **YYYYMMDD** format.
- **IsWeekend**: This flag checks if the day is a weekend (Saturday or Sunday) and returns **TRUE** or **FALSE**.

Step 4: Using the Date Table in Analysis

Once the complete date table is created, you can start using it in your analysis. Link this table to other fact tables (such as sales data) based on the **OrderDate** or similar fields. With the added features, you can now easily:

- Filter data by year, month, quarter, or week.
- Perform **time intelligence calculations** (e.g., Year-to-Date, Quarter-to-Date).
- Identify trends based on specific date features, such as the most active weekday for sales.

Conclusion

Creating a **complete date table** using DAX in Power BI is a powerful step in improving your ability to perform detailed time-based analysis. By leveraging functions like **CALENDAR**, **ADDCOLUMNS**, and **FORMAT**, you

can build a date table that includes essential attributes such as year, month, quarter, day of the week, and even custom flags like **IsWeekend**.

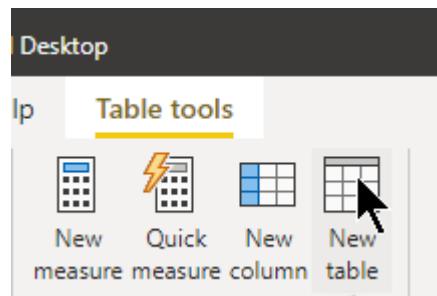
This table enables comprehensive **time intelligence analysis** and ensures your reports are accurate and flexible, making it easy to derive insights from your data. You can continue to extend this date table by adding more columns based on your specific analysis needs, giving you the flexibility to analyze your data from various time perspectives.

▼ Complete Date Table with features

Create a date table for your future projects!

This is the script you can use in your future projects and reports in order to create a date table in the beginning of your projects.

To do that just select any table in your fields pane, so the *Table tools* appear. Then just click on "New table" and paste in that code.



Important: Of course you need to replace the table 'Date Dimension'[order_date] with your own columns / table.

So, just find the date field in your data and quickly create a date table.

```
DateTable =  
ADDCOLUMNS(  
CALENDAR(MIN('Date Dimension'[order_date]), MAX('Date Dimension'[order_date]), "DateAsInteger", FORMAT([Date], "YYYYMMDD"),  
"Year", YEAR([date]), "MonthNo", FORMAT([date], "MM"), "YearMonthNo", FORMAT([date], "YYYY/MM"),  
"YearMonth", FORMAT([date], "YYYY/mmm"), "MonthShort", FORMAT([date], "mmm"))
```

```
"MonthLong", FORMAT ( [date], "mmmm" ),  
"WeekNo", WEEKDAY ( [date] ),  
"WeekDay", FORMAT ( [date], "dddd" ),  
"WeekDayShort", FORMAT ( [date], "ddd" ),  
"Quarter", "Q" & FORMAT ( [date], "Q" ),  
"YearQuarter", FORMAT ( [date], "YYYY" ) & "/Q" & FORMAT (
```

Hope it helps :)

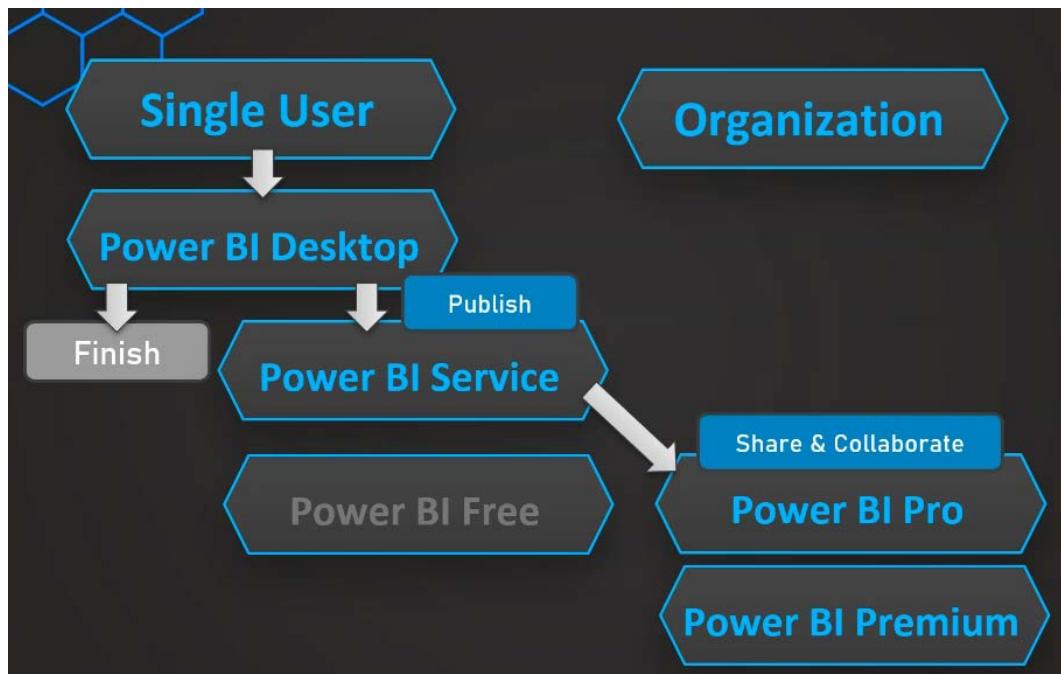
▼ Project 10 : DAX Challenge

[Project10.rar](#)

▼ 10- Publishing Reports & Power BI Services

▼ Publishing Reports to Power BI Service

Once you've developed your report in Power BI Desktop, you may ask, "What's the next step?" This section discusses the options available for publishing your reports depending on your specific needs and organizational structure.



1. Local Consumption of Reports

If you're an individual user and don't need to share your report with others, the simplest approach is to consume your report locally on your machine through Power BI Desktop. In this scenario, no further steps are required. You're already done, and you can view your report directly within the application.

2. Publishing to Power BI Service

However, if you need to access your reports remotely—whether from another device or location—you can publish your report to the Power BI Service. This cloud-based service allows you to access your reports from anywhere via a browser or mobile device.

- **Power BI Free License:** If you're an individual user and don't need to share your report with others, the **Power BI Free** plan will suffice. With this plan, you can upload and access your reports online without incurring any additional costs. It provides basic functionalities, including report viewing and limited interaction.

3. Collaborating with Others: Power BI Pro

When working in a larger organization where collaboration is key, you'll likely need to share and collaborate on reports with your colleagues. In this case, the **Power BI Free** plan will not meet your requirements, as it does not support sharing or collaboration features.

To enable sharing, collaboration, and advanced reporting capabilities, you'll need the **Power BI Pro** license. The Power BI Pro plan includes:

- Full access to share reports with others.
- Collaborate with team members in real time.
- Perform more advanced data analysis.
- Access additional Power BI features not available in the Free version, such as content packaging and distribution.

This option is ideal for organizations that need team-wide access to reports, with the ability to create, share, and interact with reports across the organization.

4. Scaling with Power BI Premium

For larger organizations with more complex requirements or those needing to scale their Power BI environment, the **Power BI Premium** plan offers even greater capabilities:

- **Enterprise-level performance:** With dedicated cloud resources, Premium allows organizations to handle larger datasets and more intensive workloads.
- **Enhanced scalability:** This plan supports the needs of organizations that require extensive usage of Power BI reports across multiple teams and departments.
- **Paginated Reports:** This feature enables you to create fixed-layout reports optimized for printing or PDF generation.
- **AI-powered insights:** Power BI Premium includes AI and machine learning capabilities to unlock advanced analytical scenarios.

Premium also offers the flexibility of **per-user (Premium Per User)** or **capacity-based** pricing models, allowing organizations to choose the most

cost-effective solution based on their usage.

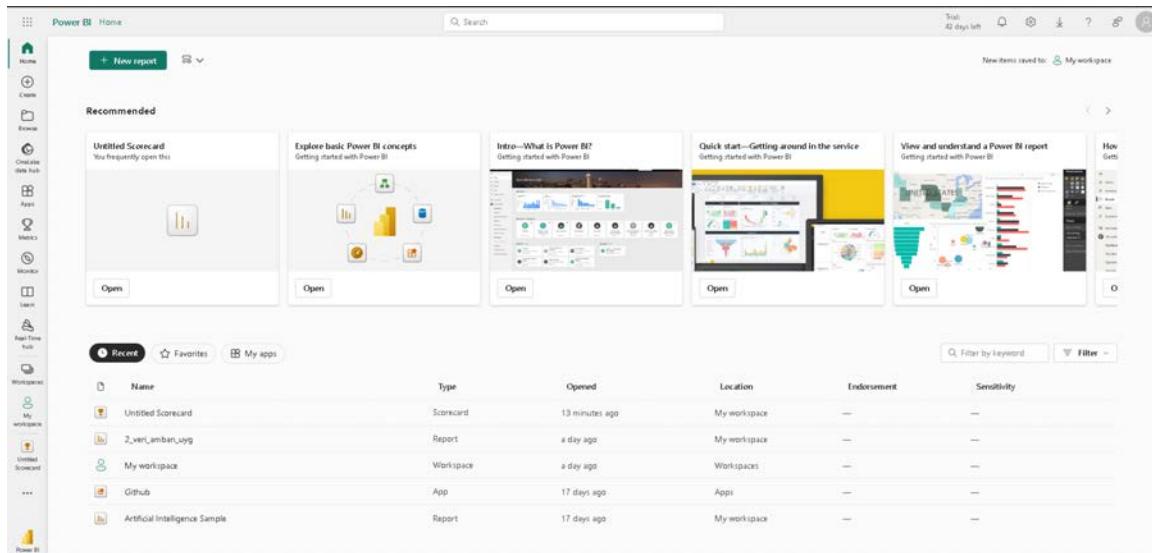
5. Summary of Licensing Models

To summarize, Power BI offers three main licensing options depending on your needs:

- **Power BI Free:** Suitable for individual users who do not need to share reports or collaborate.
- **Power BI Pro:** Ideal for small to medium-sized teams or organizations that need to collaborate and share reports across users.
- **Power BI Premium:** Best for large organizations with advanced performance, scalability, and collaboration requirements.

▼ Quick Interface Tour of Power BI Service

Before we dive into the more detailed functionalities of Power BI in the cloud, let's start by getting familiar with the main interface. Once you've logged into your Power BI account, you will see several key elements that help you navigate and manage your reports effectively.



1. Navigation Pane (Left Sidebar)

The most important feature on the main screen is the **Navigation Pane** on the left side. This pane allows you to move between different sections of

Power BI and is something you'll use frequently. Here's a quick breakdown of the key options:

- **Home**: The default section where you can access all your Power BI assets.
- **Favorites**: Contains reports and dashboards that you've marked as favorites for quick access.
- **Recent**: Displays the reports and dashboards you've recently opened.

While these sections will be discussed in detail throughout the course, it's important to note that the navigation pane is essential for accessing different parts of Power BI Service. Although you have the option to hide this pane, it's recommended to keep it visible for easy access.

2. Top Right Navigation Menu

In the top-right corner of the screen, you will find additional controls and settings, including:

- **Notifications**: Alerts related to your reports or shared content.
- **Settings**: This is where you can manage various account configurations, such as:
 - **Gateways**: Manage your on-premises data gateways (to be discussed later in the course).
 - **Account Settings**: Customize personal settings, including language preferences.

You can also access the **Download** button here, which allows you to download important components such as:

- **Power BI Desktop**: The desktop version of Power BI for report creation.
- **Data Gateway**: A tool for connecting on-premises data sources to Power BI Service.

Additionally, this menu contains your **Profile** section, where you can sign out of your account when necessary.

3. Workspaces Overview

Another critical part of Power BI Service is **Workspaces**. Workspaces are collaborative environments where users can share, edit, and publish reports within a team or organization. These will play a key role in the next steps of your Power BI journey. Workspaces allow you to:

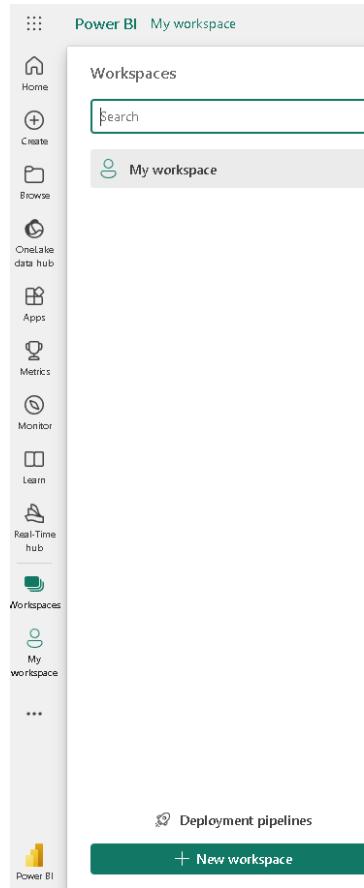
- Organize your reports and dashboards.
- Collaborate with team members.
- Manage permissions and access levels.

▼ My Workspace in Power BI

When publishing reports in Power BI, you need to publish them into a **Workspace**. A workspace is essentially a container where you manage your reports, datasets, and other assets. For individual use, Power BI provides a default workspace called **My Workspace**.

1. My Workspace Overview

- **My Workspace** is your personal workspace in Power BI Service. It is designed for individual use, and reports stored here are not shared with others unless explicitly published or shared.
- If you're working on personal projects or don't need to collaborate with others, you can safely publish your reports here without the need to set up additional workspaces.



2. Components of My Workspace

Within **My Workspace**, you will find several key elements:

- **Dashboards:** Dashboards are collections of key visuals from one or more reports, often used to highlight important insights. You can curate different elements from multiple reports to create a unified view.
- **Reports:** These are the actual reports you've created in Power BI Desktop. Reports contain multiple pages and visuals based on your datasets.
- **Workbooks:** Typically, these refer to Excel files, but are not the primary focus in Power BI unless you're working with Excel-based data.
- **Datasets:** Datasets are crucial, as they represent the underlying data model used by your reports. Every time you publish a report, a corresponding dataset is uploaded to the cloud. This dataset allows

Power BI to refresh the data, ensuring that your report stays up to date with the latest information.

3. Using My Workspace vs. Shared Workspaces

While **My Workspace** is a great option for personal projects, it is limited to your own use. If you're working in a team or need to collaborate with others, you will likely need to create or join a **Shared Workspace**. In shared workspaces, multiple users can access and collaborate on reports, datasets, and dashboards.

- **My Workspace:** Best for individual use, personal projects, and reports that don't require collaboration.
- **Shared Workspaces:** Ideal for team projects, collaboration, and sharing reports with others.

▼ Publishing Report

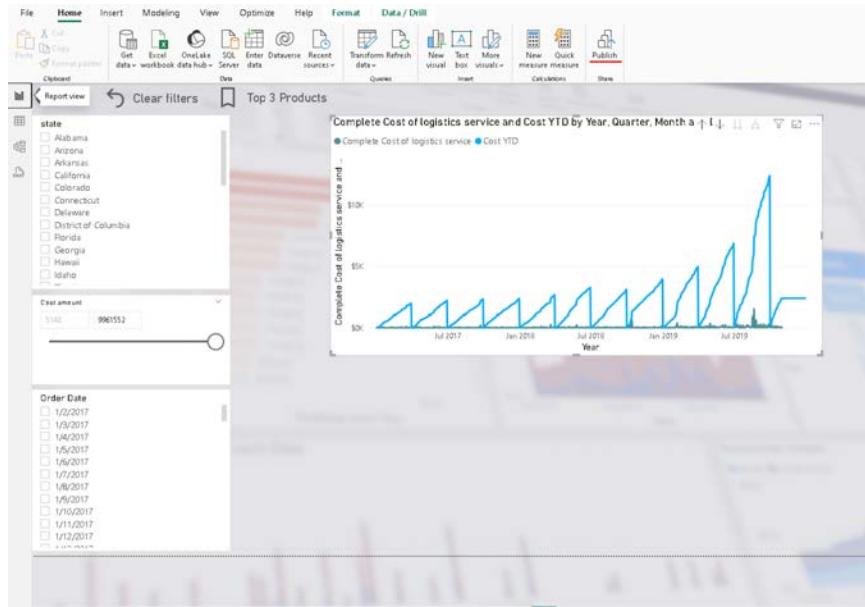
Now that the report is ready and the customer has requested to move it to the cloud, it's time to publish it to Power BI Service. Here's a step-by-step guide to quickly and easily publish your report.

1. Using the Publish Button

The easiest way to publish a report is by clicking the **Publish** button in Power BI Desktop. This will upload your report to Power BI Service, allowing you to access it online through your account.

Before publishing, ensure that:

- **You are signed in:** Power BI needs to know which account the report should be published to. You can sign in by clicking on the profile icon at the top of the window or directly when prompted after clicking the Publish button. Use the email address associated with your Power BI account.
- **Save your report:** If you haven't saved the latest changes, Power BI will prompt you to do so before publishing.

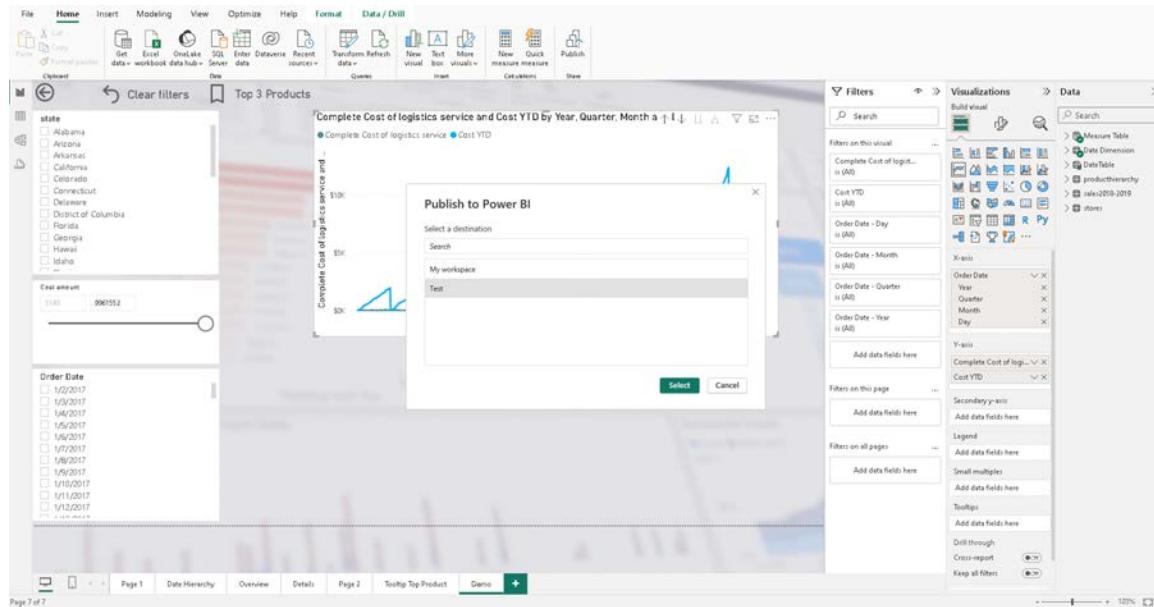


2. Selecting a Destination Workspace

After signing in, Power BI will ask you to select a destination for your report. You can choose either:

- **My Workspace:** This is your personal workspace, suitable for individual projects.
- **Other Workspaces:** If you've created or have access to shared workspaces, they will also appear here.

Since no shared workspaces have been created yet, select **My Workspace** for now.



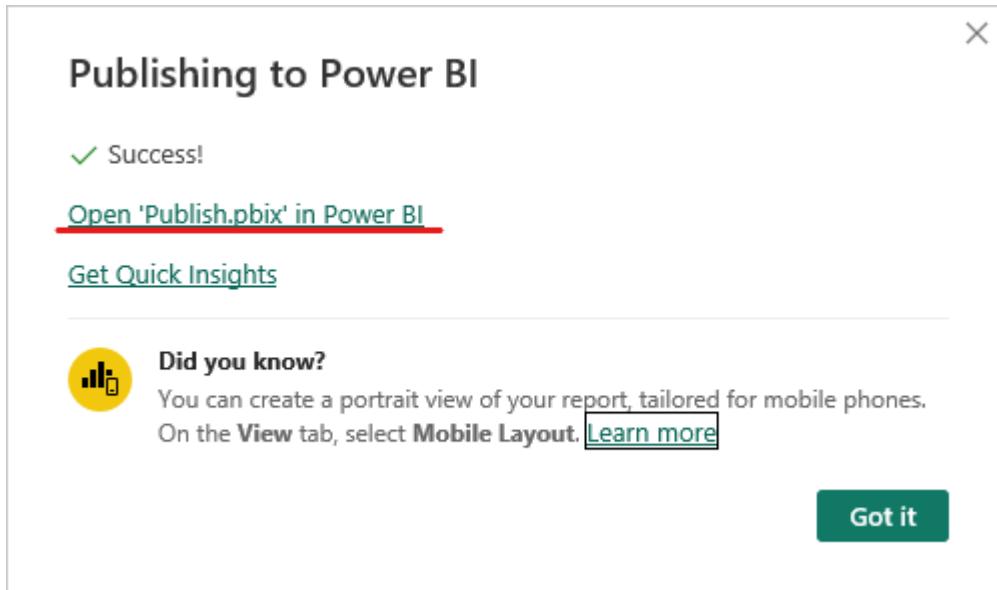
3. Publishing the Report

Once you've selected the destination, click **Select** to begin the publishing process. Power BI will upload both your report and its associated dataset to the selected workspace. The process is quick, and you'll receive a confirmation message once the report is successfully published.

4. Accessing Your Published Report

After publishing, you have two options to view your report in the cloud:

- **Click the provided link:** This will take you directly to the published report in Power BI Service.
- **Navigate through your account:** You can log into Power BI Service in your browser, navigate to **My Workspace**, and find the published report there. You may need to refresh the page if it doesn't appear immediately.



In the **My Workspace** section, you'll find both the **report** and the **dataset** listed under the names you assigned. The report is what you will view and interact with, while the dataset represents the underlying data model.

▼ Creating and Editing Reports in Power BI Service

1. Introduction to Power BI Service

Power BI Service is a cloud-based platform for sharing, collaborating, and managing Power BI reports and dashboards. You can create, edit, and view reports directly within the service after publishing from Power BI Desktop.

2. Creating Reports in Power BI Service

2.1 Data Sources

- **Importing Data:** Connect to various data sources such as Excel files, databases (SQL Server, Azure, etc.), cloud sources (SharePoint, OneDrive), and Power BI datasets.
- **Using Datasets:** Create reports from existing datasets that have been published to the Power BI Service.

2.2 Starting a New Report

- **From a Dataset:** Select a published dataset and click "Create report".
 - **Create a Blank Report:** Choose a dataset and design the report from scratch.
-

3. Editing Reports

3.1 Adding Visuals

- **Visual Types:** Choose from a variety of visuals, including bar charts, line charts, pie charts, tables, matrices, maps, and custom visuals.
- **Field Pane:** Drag and drop fields (columns or measures) into visualizations.
- **Visualization Pane:** Customize visual properties like colors, labels, and axes.

3.2 Filters and Slicers

- **Report-Level Filters:** Apply filters that affect all pages of the report.
- **Page-Level Filters:** Apply filters that affect only the selected page.
- **Visual-Level Filters:** Apply filters to a specific visual.
- **Slicers:** Add slicers to allow interactive filtering by users.

3.3 Editing Interactions

- Control how visuals on a report page interact with each other. For instance, selecting a value in one visual can highlight or filter data in another visual.

3.4 Drillthrough and Drill Down

- **Drillthrough:** Set up a page to focus on a specific entity by using a drillthrough filter.
 - **Drill Down/Drill Up:** Enable users to drill down to see more detailed data or drill up to view summarized data.
-

4. Customizing Report Layout

4.1 Adding Report Pages

- Add multiple pages to your report for organizing data into different sections.

4.2 Page Settings

- Adjust page size, orientation (landscape or portrait), and background settings for a professional layout.
-

5. Saving and Publishing Reports

5.1 Saving Reports

- **Auto-Save:** Reports are auto-saved when working in Power BI Service.
- **Save As:** Create a duplicate of an existing report for modification without affecting the original.

5.2 Publishing Reports

- Once a report is finalized, it can be published to specific workspaces for sharing and collaboration.
-

6. Sharing and Collaborating on Reports

6.1 Sharing Reports

- Share reports with specific users, groups, or within your organization through "Share" functionality.
- Control permissions to view or edit reports when sharing.

6.2 Exporting Reports

- **PDF Export:** Export your report as a PDF for offline sharing or printing.

- **PowerPoint Export:** Export your report as a PowerPoint file to include in presentations.

6.3 Collaborating in Workspaces

- **Workspaces:** Use Power BI workspaces for collaboration where multiple users can create, edit, and share reports.
-

7. Report Settings and Versioning

7.1 Changing Report Settings

- Manage permissions, refresh schedules, and other settings for each report from the report's settings page.

7.2 Version History

- Track changes to reports using version history. Restore previous versions if needed.
-

8. Mobile View

- Design and optimize reports for viewing on mobile devices by configuring the mobile layout in Power BI Service.
-

9. Additional Features

9.1 Bookmarks

- Create bookmarks to capture and save report views. You can switch between views with a click, useful for storytelling.

9.2 Q&A Visuals

- Add a Q&A visual to your report allowing users to ask natural language questions and get visualized answers.

9.3 Performance Analyzer

- Analyze the performance of report visuals to identify bottlenecks or slow-loading visuals.
-

10. Conclusion

Power BI Service offers powerful tools for creating and editing reports directly in the cloud. Whether you're building reports from scratch or editing existing ones, the service provides an interactive and collaborative environment for data analysis and visualization. By mastering the key features—such as creating visuals, managing filters, customizing layouts, and sharing reports you can efficiently turn data into actionable insights.

▼ Editing Reports in Power BI Service

After publishing your report to the cloud, it's important to understand the relationship between the **report** and the **dataset**. These two components work together, and knowing how to manage and edit them in Power BI Service is crucial for ongoing report development and maintenance.

1. Navigating My Workspace

Once you've published a report, both the report and its associated dataset will appear in **My Workspace**. To get a clearer overview:

- Click on **My Workspace** to expand and view all content.
- You'll see both the **report** and the **dataset** that was published. The **dataset** represents the data model, while the **report** is the visual layer built on top of this data.

The screenshot shows the Power BI workspace interface. On the left is a navigation bar with icons for Home, Create, Explore, Data Hub, Apps, Metrics, Monitor, Real-Time, Workspaces, and Power BI. The main area has a title 'Power BI Test' and a sub-section 'Test' with a status 'For testing'. Below this is a search bar and a toolbar with 'Create app', 'Manage access', and 'Workspace settings'. A central circular icon with a document and plus sign is present. Below it is a message: 'Choose from predesigned task flows or add a task to build one (preview) Select from one of Microsoft's predesigned task flows or add a task to start building one yourself.' A button 'Select a predesigned task flow' and a dropdown 'Add a task' are shown. At the bottom is a table with columns: Name, Type, Task, Owner, Refreshed, Next refresh, Endorsement, Sensitivity, and Included in app. Two rows are listed: 'Publish' (Report) and 'Publish' (Semantic model). The 'Publish' (Report) row is highlighted with a red border.

2. Using the Dataset for New Reports

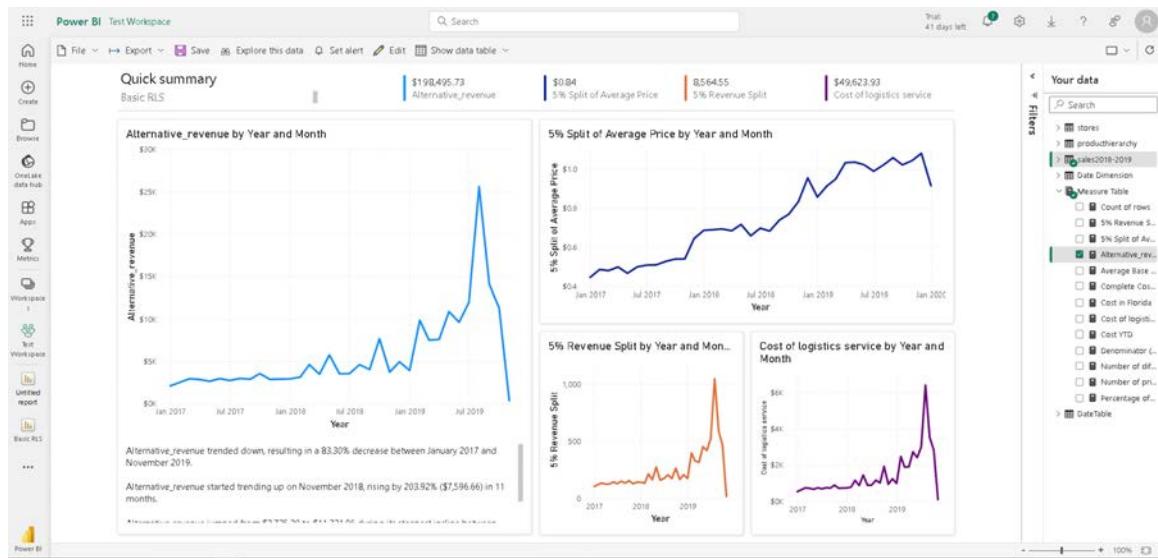
A key advantage of publishing your dataset is the ability to reuse it in multiple reports. You can create new reports based on the same dataset, avoiding the need to rebuild the data model each time. Here's how:

- In **My Workspace**, click the three dots next to the dataset and select **Create Report**.

This screenshot is similar to the previous one but shows a context menu open over the 'Publish' dataset. The menu options include: Explore this data (preview), Analyze in Excel, CREATE REPORT (highlighted in red), Auto-create report, Create paginated report, Delete, Get quick insights, Security, Rename, Open data model, Settings, Refresh history, Download this file, Manage permissions, View workspace lineage, View item lineage, Move to, and Write DAX queries. The rest of the interface is identical to the first screenshot.

- This will open the familiar Power BI interface, similar to what you're used to in Power BI Desktop, where you can use the same data model to create new visualizations.

For example, you can drag in fields like **Weekdays** and **Sales** to create a new chart, such as a stacked bar chart, just like in Power BI Desktop.



3. Saving New Reports

Once you're happy with your new report, you can save it:

- Click the **Save** button and give the report a new name, such as "Second Report."

Name	Type	Task	Owner	Refreshed	Next refresh
Publish	Report	—	Test	25/09/24, 15:39:24	—
Publish	Semantic model	—	Test	25/09/24, 15:39:24	N/A
second report	Report	—	Test	25/09/24, 15:39:24	—

- The new report will now appear alongside the original report in **My Workspace**, both based on the same dataset.

4. Editing Existing Reports

Power BI Service also allows you to edit reports directly in the cloud without needing to switch back to Power BI Desktop. Here's how you can make adjustments to an existing report:

- Open your published report from **My Workspace**.

- If any visuals are broken or outdated, you can fix them directly by clicking the **Edit** button.
- This brings you into the familiar editing interface, where you can adjust or delete visuals as needed.

For example, if there are two broken visuals, you can delete them by selecting and removing them directly in the report.

5. Saving Edits

After making changes, saving the report is simple:

- Click **Save** to apply the updates.
- You can then switch back to **Reading Mode** to view the final report, with all your edits applied.

The page navigation in the reading mode is positioned on the left side, which is considered more user-friendly, making it easier for users to navigate between report pages.

6. Updating an Existing Report from Power BI Desktop

In some cases, you may prefer to update your report in **Power BI Desktop** rather than in the cloud. After making changes locally, you can re-publish the updated report to Power BI Service.

▼ Creating & Exploring Workspaces in Power BI

To share reports and collaborate with others, you'll need to create **Workspaces** in Power BI. **My Workspace** is a personal environment that doesn't support sharing or collaboration. To enable collaboration, we need to create a dedicated workspace. Here's how it works.

1. Why Create a Workspace?

When a customer or team wants to collaborate on reports or share them with others, **My Workspace** isn't sufficient. A **Workspace** in Power BI serves as a collaborative environment where multiple users can access, modify, and share reports, datasets, and dashboards. Think of it as a project-specific space designed for teamwork.

2. Creating a Workspace

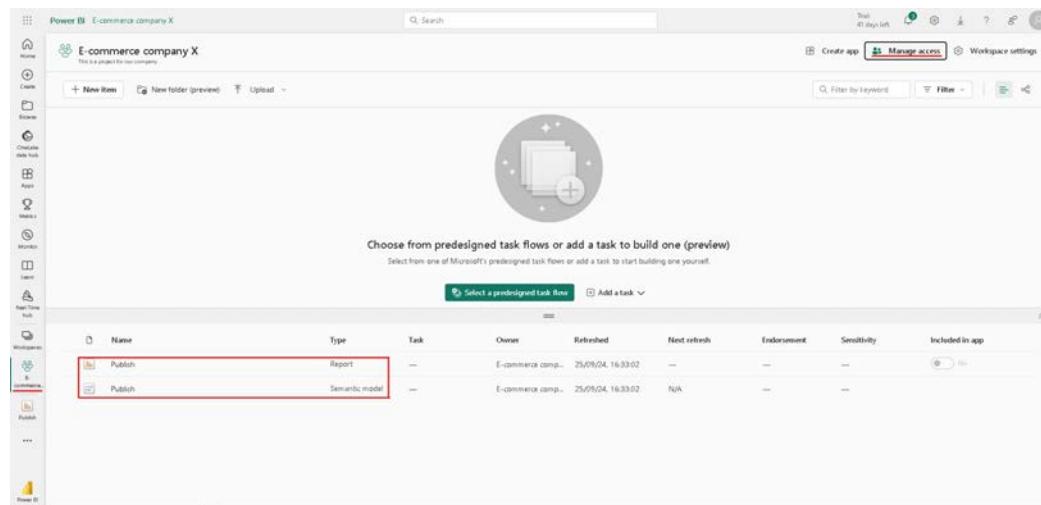
To create a workspace, follow these steps:

1. Navigate to the **Workspaces** section in Power BI.
2. Click on **Create a Workspace**.
 - Note: Workspace creation requires a **Power BI Pro** license, as sharing and collaboration are Pro features.
 - If you don't have a Pro license, you can activate a free 60-day trial by selecting **Try Free**.
3. Once you're using Power BI Pro, proceed to create the workspace by naming it. For example, you can name it **E-commerce Project** and add a description, such as "Project for sharing reports with the team."
4. After entering the necessary details, click **Save** to create the workspace.

3. Workspace Features

After creating the workspace, you'll see a familiar interface that looks like **My Workspace**, but with additional functionality for collaboration:

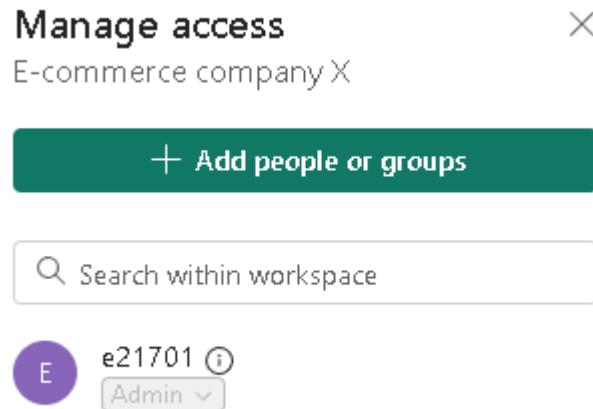
- You can publish reports to this workspace, just like you would in **My Workspace**, but now other users with access can view and work on the reports.
- To publish a report to this new workspace, follow the same process you used for publishing to **My Workspace**:
 - Open your report in Power BI Desktop, select **Publish**, and choose the newly created workspace as the destination.



4. Granting Access to Other Users

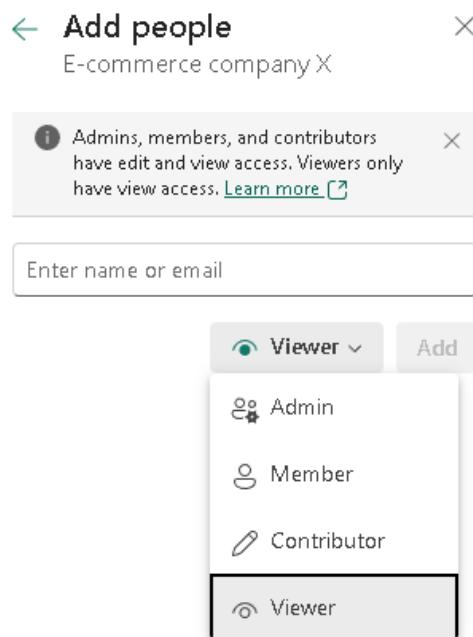
To collaborate with others, you need to give them access to the workspace. Here's how:

1. In the **Workspaces** pane, click on the three dots next to your workspace name.
2. Select **Manage Access**.



3. In the **Access Settings**, you can add users by entering their email addresses and assigning them roles:
 - **Admin:** Full control, including the ability to manage the workspace and invite new members.

- **Member:** Can contribute and work on reports but cannot manage workspace settings or invite other admins.
- **Contributor:** Can edit and update reports but cannot manage overall workspace settings.
- **Viewer:** Can view the reports but cannot edit or make changes.



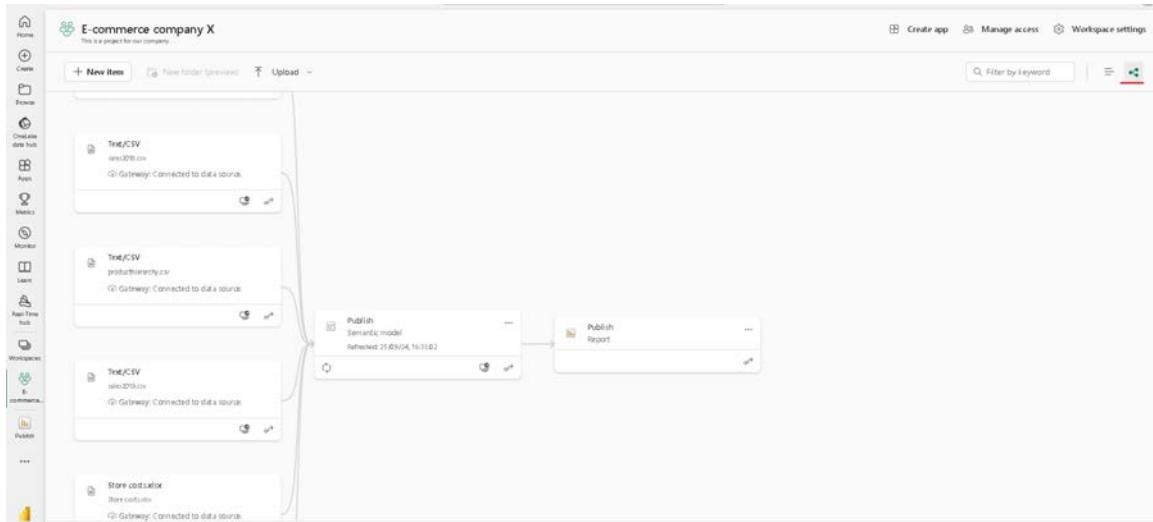
This role-based access ensures that different users have appropriate permissions for working within the workspace.

5. Exploring the Workspace

Once the workspace is set up, you can manage its content similarly to **My Workspace**, but with some added capabilities:

- You can create and publish multiple reports based on a single dataset.
- View and analyze the **lineage** of your reports and datasets, which shows how data sources, datasets, and reports are interconnected.

To switch between different views (List and Lineage), use the view toggle at the top of the workspace. The **Lineage** view provides a graphical representation of how your reports are built from datasets, showing dependencies across multiple sources and reports.



6. Refreshing Data

In the workspace, you can manually refresh your reports and datasets:

- Select **Refresh** next to a dataset or report to update the data. If no updates are available, nothing will change.
- However, if you are working with on-premises data, you'll need to set up a **Data Gateway** to enable refreshing of datasets connected to local data sources.

If a refresh fails, you'll see an error, which typically indicates the need for a data gateway setup. This will be covered in the next session.

Something went wrong

Scheduled refresh has been disabled.

Please try again later or contact support. If you contact support, please provide these details.

[See details](#)

[Get help](#)

[Close](#)

Name	Type	Task	Owner	Refreshed	Next refresh
Publish	Report	—	E-commerce co...	25/09/24, 16:43:49	—
Publish	Semantic model	—	E-commerce co...	25/09/24, 16:43:49	N/A

▼ 11- Scheduling & Data Gateway

▼ Installing and Using a Data Gateway in Power BI

When working with Power BI, you may encounter situations where you need to refresh your data from local files or on-premises servers. This is where a **Data Gateway** comes into play, allowing Power BI Service to securely access and update data stored on local machines or servers. Here's a detailed guide on how to install and use a Data Gateway in Power BI.

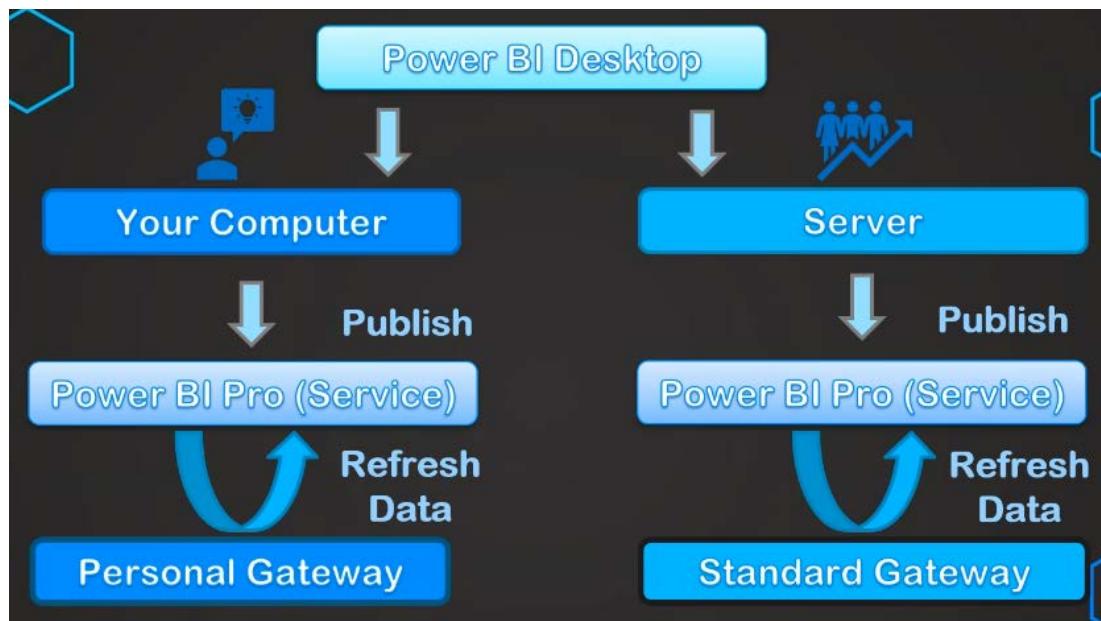
1. Why Do You Need a Data Gateway?

Yesterday, we attempted to refresh data in our published report on Power BI Service, but the refresh failed. This happened because Power BI Service cannot directly access files stored on your local machine or on-premises servers. To bridge this gap, we need to use a **Data Gateway**. A gateway provides a secure connection between Power BI Service and your local data sources, ensuring that data can be updated regularly and seamlessly.

2. Types of Data Gateways

There are two main types of Data Gateways in Power BI:

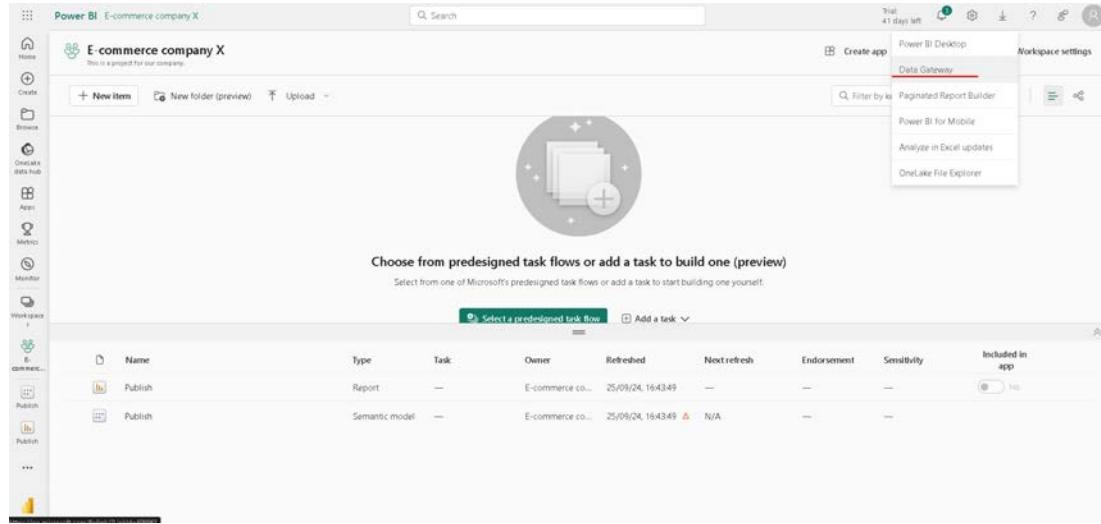
- **Personal Mode Gateway:** This is used when the data source is on your local machine. It is suitable for individual use where data is stored locally and you are the sole user of the gateway.
- **Standard Mode Gateway:** This is used when working with on-premises servers, such as databases or shared data files. It allows multiple users and is typically used in enterprise environments.



3. Installing a Personal Mode Data Gateway

If you're working with local files, follow these steps to install a **Personal Mode Data Gateway**:

1. Go to **Power BI Service** and click on the **Downloads** button in the top-right corner.



2. Select **Data Gateway**. This will take you to the download page, where you can choose between the **Standard Mode** and **Personal Mode** gateways.

Connect to on-premises data sources with a Power BI gateway

Keep your dashboards and reports up to date by connecting to your on-premises data sources without the need to move the data. Query large datasets and take advantage of your existing investments. Get the flexibility you need to meet individual needs, and the needs of your organization.

[Download standard mode](#)

[Download personal mode](#)

- Since we're working with local files, select **Download Personal Mode**.
- 3. Download the installation file for the **On-Premises Data Gateway** in **Personal Mode** and run the installer.
- 4. During installation, specify the installation path (or leave the default settings) and agree to the terms and privacy statement. Click **Install** to begin the installation.
- 5. Once installed, you'll be prompted to sign in to your Power BI account. This allows the gateway to connect to your account and enable data refreshes from your local machine.

4. Using the Data Gateway

After installing the gateway, Power BI will notify you that the gateway is online and ready to use. Now, you can return to Power BI Service to check if your dataset can be refreshed.

1. Navigate to your dataset in **My Workspace** or the relevant workspace.
2. Click **Refresh Now** on your dataset.

However, you might still encounter an issue where the refresh doesn't work, despite the successful gateway installation. This could happen because the **data source credentials** haven't been configured yet.

5. Configuring Data Source Credentials

In order to refresh your dataset through the gateway, you need to sign in to each individual data source:

1. Go to the **Settings** of your dataset in Power BI Service.

The screenshot shows the 'Settings' page in the Power BI Service. At the top, there is a navigation bar with icons for notifications (3), settings, download, help, and user profile. Below the navigation bar, the title 'Settings' is displayed. The main content area is organized into sections: 'Preferences' (with links to General, Notifications, Item settings, and Developer settings), 'Resources and extensions' (with links to Manage group storage, Power BI settings, Manage connections and gateways, Manage embed codes, and Azure Analysis Services migrations), and 'Governance and insights' (with links to Admin portal and Microsoft Purview hub (preview)). The 'Power BI settings' link under 'Resources and extensions' is highlighted with a red underline.

2. Under **Data Source Credentials**, you'll see a warning if credentials are missing or invalid.
3. Click **Edit Credentials** for each data source. Even for simple files like CSVs, Power BI requires confirmation of access.

Settings for Publish

[View semantic model](#)

This semantic model has been configured by [e21701@a3vip.top](#).

Last refresh failed: 25/09/2024, 16:43:49
Scheduled refresh has been disabled. [Show details](#)

Refresh history

Semantic model description

Describe the contents of this semantic model.

500 characters left

Apply Discard

Gateway and cloud connections

Data source credentials

Failed to test the connection to your data source. Please retry your credentials. [Learn more](#)

Store costs.xlsx	Edit credentials	Show in lineage view
producthierarchy.csv	Edit credentials	Show in lineage view
sales2017_raw.csv	Edit credentials	Show in lineage view
sales2018.csv	Edit credentials	Show in lineage view
sales2019.csv	Edit credentials	Show in lineage view

- For each file, select **Sign In**, verify the file path, and complete the process.
- If you're connecting to a database or a server, you'll need to enter the database credentials (username and password) as well.

After successfully updating the credentials for each data source, the X or warning message will disappear. Your dataset should now be ready for scheduled or manual refreshes.

6. Standard Mode Data Gateway (For Server-Based Data)

If you're working in a larger organization with data stored on a server, you'll need the **Standard Mode Data Gateway**. The installation process is similar, but the gateway must be installed on the server itself, and it will require administrator access.

- The **Standard Gateway** supports multiple users and allows data to be refreshed from a variety of on-premises data sources, including SQL databases, shared folders, and enterprise systems.

- Your IT administrator typically manages the installation of the Standard Gateway, but the general installation steps are similar to the **Personal Gateway**.

7. Updating Data Sources in Power BI Desktop

If you're sharing your project files with others or moving the files across machines, you may need to update the file paths in Power BI Desktop. Here's how:

1. Open Power BI Desktop and navigate to **Transform Data > Data Source Settings**.
2. Select the dataset you need to update and click **Change Source** to point it to the correct file on your local machine.
3. Once updated, publish the revised report to Power BI Service.

▫Gateway and cloud connections

To use a data gateway, make sure the computer is online and the data source is added in [Manage Connections and Gateways](#). If you're using an On-premises data gateway (standard mode), please select the corresponding data sources and then click apply.

Gateway connections

Use an On-premises or VNet data gateway



Gateway	Department	Contact information	Status	Actions
<input checked="" type="radio"/> Personal Gateway			Running on DESKTOP-ULTVN24	

Cloud connections

No cloud connections

[Apply](#) [Discard](#)

▫Data source credentials

Store costs.xlsx	Edit credentials	Show in lineage view	
producthierarchy.csv	Edit credentials	Show in lineage view	
sales2017_raw.csv	Edit credentials	Show in lineage view	
sales2018.csv	Edit credentials	Show in lineage view	
sales2019.csv	Edit credentials	Show in lineage view	
store_cities.csv	Edit credentials	Show in lineage view	

8. Final Steps and Troubleshooting

Once you've configured your data sources and the gateway is online, you should be able to refresh your datasets without issues. Remember, if you're

using shared project files, ensure that the file paths are updated to reflect your local machine's structure.

To sum up:

- **Install the Data Gateway** for secure access to local or on-premises data.
- **Configure data source credentials** for each file or database.
- **Check file paths** in Power BI Desktop to ensure data sources are correctly linked.

▼ Scheduled Refresh

Manually refreshing your dataset every time the underlying data updates is inefficient, especially if the data is updated regularly. To automate this process, Power BI offers a **Scheduled Refresh** feature that allows you to set up automatic data refreshes. Here's how you can configure it for your published datasets.

1. Accessing Scheduled Refresh Settings

To set up a scheduled refresh for a dataset:

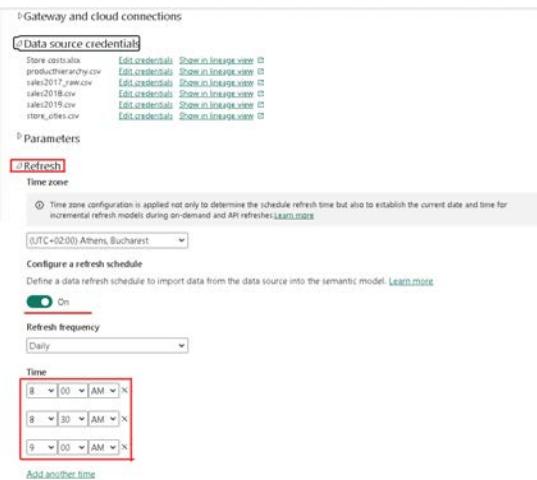
1. Navigate to the Power BI **Settings** in Power BI Service.
2. Under Semantic Models, find the dataset you wish to refresh automatically (e.g., your published dataset).
3. Click on the **Scheduled Refresh** option.

2. Enabling and Configuring Scheduled Refresh

Once you are in the **Scheduled Refresh** section:

- **Turn on Refresh:** Toggle the option to enable automatic refreshes.
- **Frequency:** Choose how often you want the dataset to refresh:
 - **Daily:** You can set specific times for the refresh to occur, for example at **12:30 PM**.

- **Multiple Times a Day:** If needed, you can schedule multiple refreshes throughout the day, such as **8:30 AM** and **6:00 PM**.
- **Weekly:** For less frequent updates, you can schedule the refresh to occur on specific days of the week (e.g., every Sunday).



3. Refresh Frequency Limits: Free vs. Pro

The number of scheduled refreshes you can configure depends on the Power BI license:

- **Power BI Free:** You are allowed up to **8 refreshes** per dataset per day.
- **Power BI Pro:** Allows up to **48 refreshes** per day, which provides more flexibility for larger, frequently updated datasets.

4. Failure Notifications

It's essential to stay informed if something goes wrong with your data refresh:

- **Refresh Failure Notifications:** You can configure Power BI to send notifications if the refresh fails. By default, the dataset owner will receive the alerts.
- **Add Additional Recipients:** You can also include other team members by entering their email addresses to ensure that multiple people are informed in case of an issue.

5. Applying the Scheduled Refresh

Once you've configured the refresh schedule, click **Apply** to save the settings. Power BI will now automatically refresh your dataset based on the schedule you've set, ensuring that your reports and dashboards always have up-to-date data.

▼ Creating Apps

Now that we've set up our report in the cloud, configured a workspace, installed a data gateway for automatic refreshes, and scheduled regular updates, it's time to distribute the report to the right people in a structured and user-friendly way. This is where **Power BI Apps** come in. Apps allow you to distribute your reports and dashboards to a wide audience efficiently and securely.

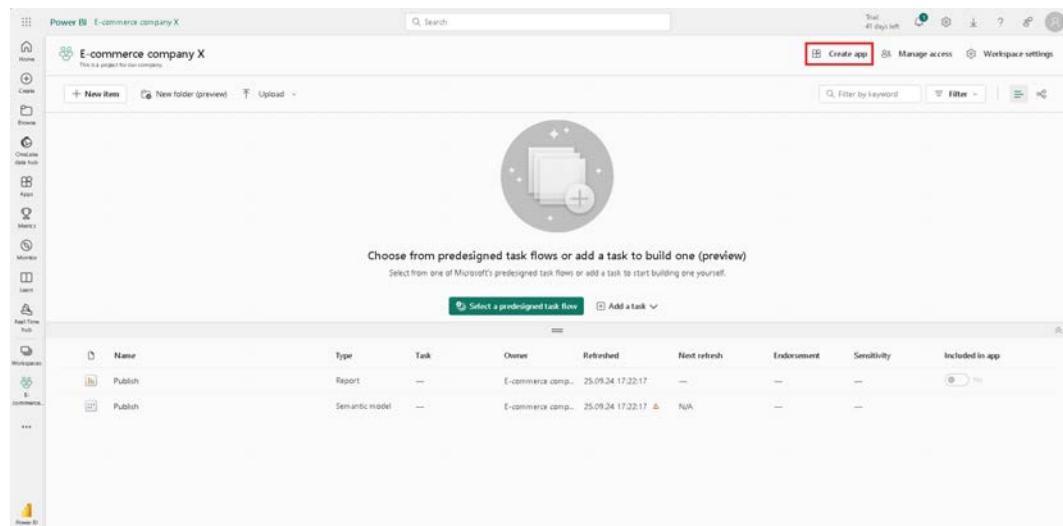
1. Why Use Power BI Apps?

Power BI Apps are designed to share insights with a large group of users in an organized manner. Rather than sharing individual reports or datasets, apps offer a polished end-product that allows users to consume the content without having access to edit or alter it. This is ideal for businesses needing to share data insights with different teams or departments while maintaining control over the content.

2. Creating an App in Power BI

To create an app, follow these steps:

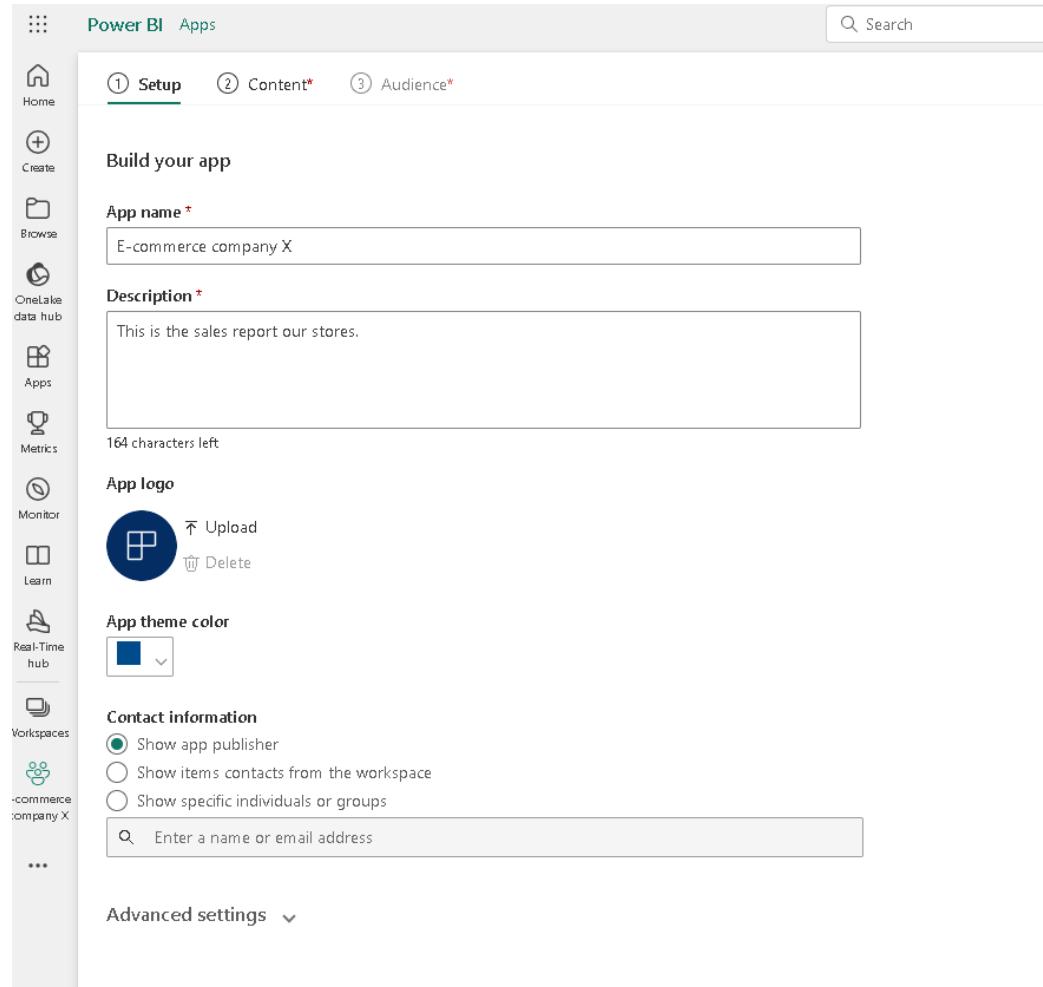
1. **Go to Your Workspace:** Navigate to the workspace where your report is published. Remember, apps can only be created within **workspaces** and not in **My Workspace**. You'll need a **Power BI Pro** license to access this feature.
2. **Create an App:** In the workspace, you'll see an option to **Create App** at the top of the page.



- **Select Reports to Include:** You can choose to include one or multiple reports in the app. In our example, we'll use a single report.

3. Setup the App:

- On the **Setup** tab, give it a name and add a description to help people find the app. You can also set a theme color, add a link to a support site, and specify contact information.
- **Description:** Provide a name and description for the app. For instance, you can name it "E-commerce company X" and add a description like, "This is the sales report for our stores."



- **Logo and Theme:** You can upload a company logo and customize the theme color (e.g., blue) to match your branding.
- **Allow saving a copy of a report (Optional)**

Before you leave the **Setup** tab, you can decide if you want to allow app users who have build permissions to save copies of reports to their workspace. Once they save the reports, app users can customize their report copies to meet their needs.

- Expand **Advanced settings** and select **Allow users to make a copy of the reports in this app**.

Advanced settings ^

Navigation pane

Expand navigation pane by default
 Collapse navigation pane by default

Access to hidden content

Turn on this setting if you want to grant access to content even if it's hidden from navigation.
Learn more about access [↗](#)

Off

Global app settings

Install this app automatically.
 Allow users to make a copy of the reports in this app.

Support site

Share where your users can find help

When you select that, app users who have build permissions can save a copy of a report from the app consumer view. You can also grant build permissions to your app users through the app using **Advanced** option under **Manage audience access** pane.

- **Allow access to hidden content**

Also, before you leave the **Setup** tab, you can decide if you want users to have access to hidden content.

Caution

If users have a direct link to *any* of the content in your app, they can access the hidden content, even if that item is visually hidden in the navigation pane for that audience.

1. Expand **Advanced settings**.
2. Under **Access to hidden content**, slide the toggle to **On**.

Advanced settings ^

Navigation pane

- Expand navigation pane by default
- Collapse navigation pane by default

Access to hidden content

Turn on this setting if you want to grant access to content even if it's hidden from navigation.

Learn more about access [\[link\]](#)

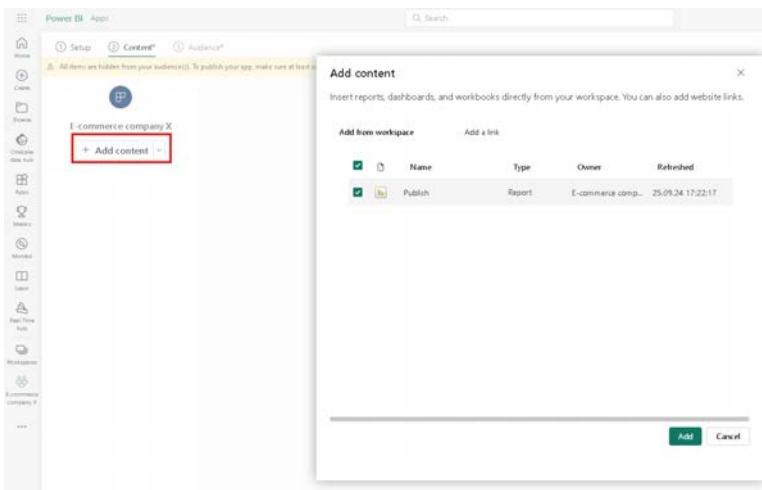


- Select **Next: Add content**.

- **Add content to the app**

On the **Content** tab, you add the content from the workspace to the app.

1. Select **Add content** on the **Content** tab.
2. Select the contents that you want to add from the current workspace.

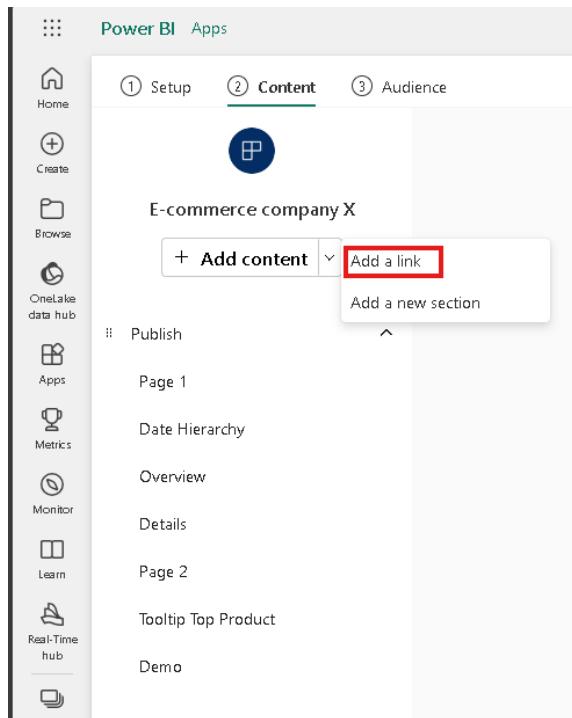


Note

When you publish an app, it includes all app content by default for each audience group. However, when

you update an app, newly added content isn't included by default.

3. You can also add links to other websites. Select **Add a link** from the drop-down menu next to **Add content**.



After you've added the content, you can change the order of the content:

4. Select **Next: Add audience**.

- **Add audience**

Next, you'll need to set the permissions for who can access the app:

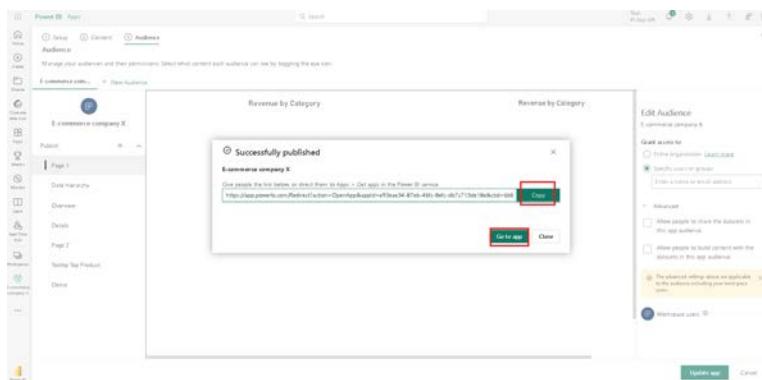
1. **Permissions:** In the **Permissions** section, specify who can access the app. You can either:
 - Share it with the entire organization.
 - Select specific individuals by entering their email addresses.

2. **Access Roles:** You can grant different levels of access, ensuring only the right people can view the app.

4. Publishing the App

Once you've set up the app and permissions:

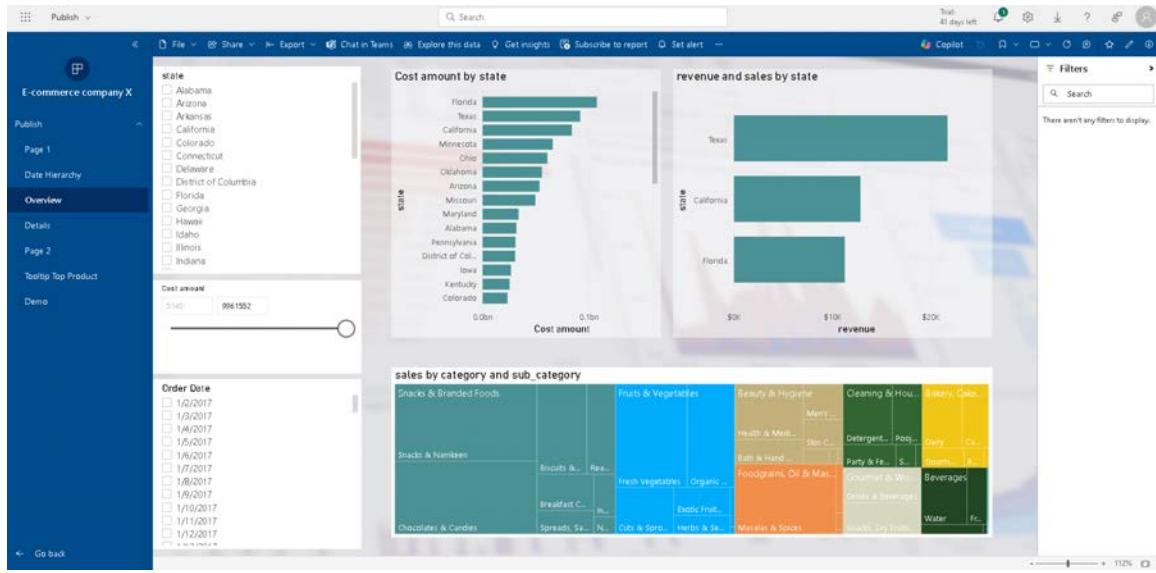
1. Click **Publish App**. The app will be published, and you'll receive a confirmation message.
2. You can either:
 - **Copy the link** and share it with users.



- **Navigate to the App** directly to view it as an end-user would.

5. Exploring the App as an End-User

The app provides a simplified, user-friendly interface designed for consumption:



- **No Editing Options:** Users won't be able to modify or edit the reports, ensuring the content remains consistent.
- **Full-Screen Mode:** Users can switch to full-screen mode for presentations.
- **Page Navigation & Filters:** You can control whether the filter pane or page navigation is visible to viewers.
 - If you want to hide certain elements from the app viewers, you can configure this within the report settings before publishing the app.

This ensures the app offers a tailored experience for the end-user, making it easy to navigate and consume insights without unnecessary complexity.

6. Accessing the App

Users can access the app through several ways:

1. **Via Shared Link:** If you share the app link, users can directly access the app.
2. **In Power BI Service:** Once they've accessed the app, it will appear under their **Apps** section in Power BI Service.
3. **Favorites:** Users can mark the app as a favorite for easy access.

The app is designed for a seamless user experience, allowing team members to view and interact with the report without worrying about data management or editing capabilities.

▼ Creating Dashboards in Power BI

Now that we have managers who only need high-level overviews of specific pages or visuals, we can create custom **dashboards** to provide these insights. Dashboards allow us to gather key visuals from different reports into one place, making it easier for decision-makers to get an overview of the most important data. Let's walk through how to create and customize a dashboard in Power BI.

1. Why Dashboards?

Dashboards are perfect for users like managers who don't need to see detailed reports but want quick access to key metrics. Unlike reports, dashboards are a high-level view of critical visuals or pages from different reports. Managers can use these dashboards to quickly assess the data and, if needed, dive into specific reports for more detail.

2. Limitations in Apps

While **Power BI Apps** are great for distributing reports, they are read-only, meaning users can't pin visuals from an app to a dashboard. Instead, dashboards are created within a **workspace** by pinning visuals or entire pages directly from reports.

3. Pinning Visuals to a Dashboard

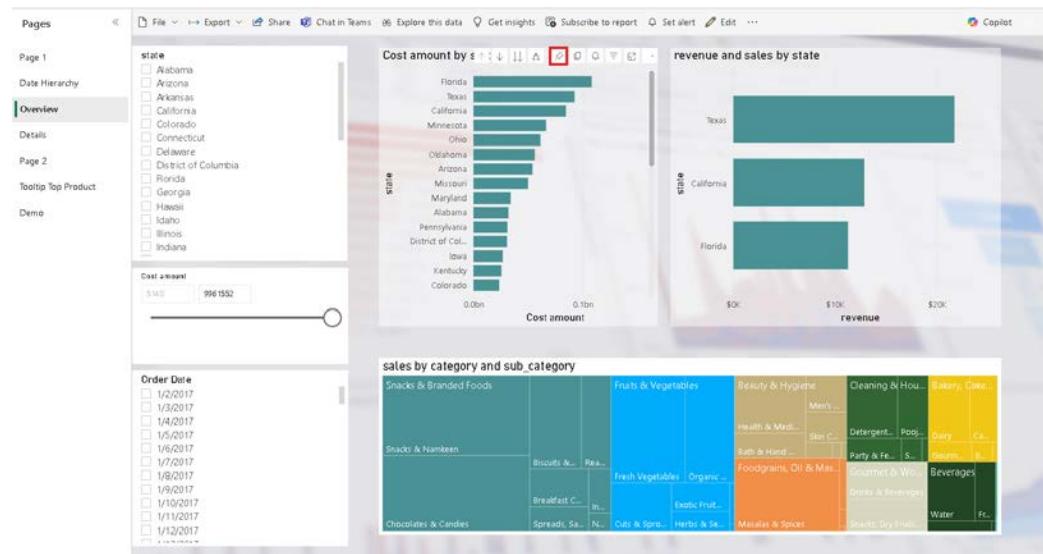
Here's how to create a dashboard by pinning visuals from a report:

1. **Go to Your Report:**

- Navigate to your workspace and open the report you want to use for the dashboard.

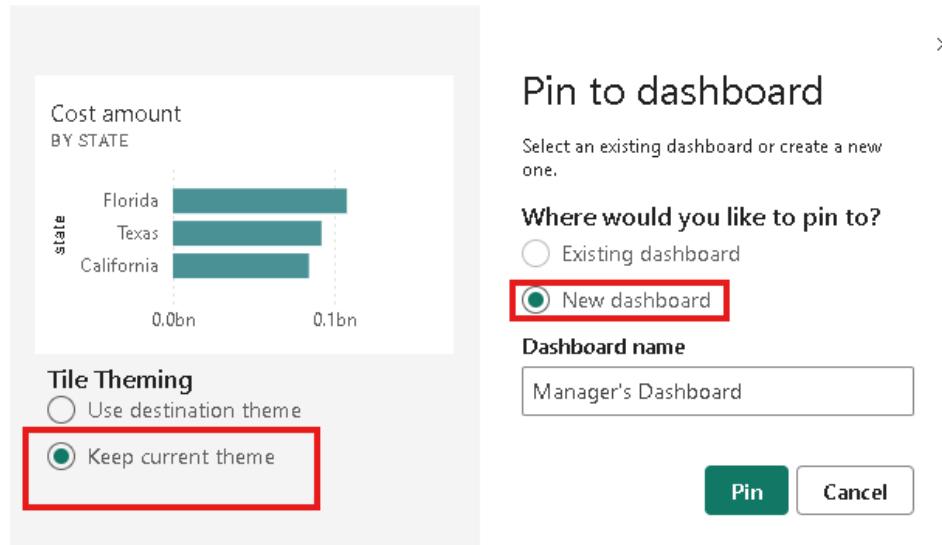
2. Select a Visual:

- In the report, locate the visual (e.g., a chart or graph) that you want to pin.
- Click the **Pin** icon that appears in the visual header.



3. Create a New Dashboard:

- Since you don't have any dashboards yet, choose to create a **New Dashboard**.
- Name the dashboard (e.g., "Manager's Dashboard") and select the theme—either the current report theme or the default dashboard theme.



4. Pin the Visual:

- After creating the dashboard, pin the visual to it. You can now either go directly to the dashboard or continue adding more visuals to the same dashboard.

4. Accessing the Dashboard

Once you've pinned a visual, you can find the dashboard in your workspace:

- **Workspace:** In the workspace, the dashboard will appear with a different icon than reports. You can open it from here.

Name	Type	Task	Owner	Refreshed	Next refresh	Endorsement	Sensitivity
Manager's Dashboard	Dashboard	—	E-commerce comp...	—	—	—	—
Publish	Report	—	E-commerce comp...	25.09.24 17:22:17	—	—	—
Publish	Semantic model	—	E-commerce comp...	25.09.24 17:22:17	N/A	—	—

- **Favorites:** If this dashboard is critical, you can also mark it as a **favorite** for quicker access.

5. Customizing the Dashboard Layout

You can easily customize the layout of your dashboard:

1. Resize and Rearrange Visuals:

- In the dashboard, you can adjust the size of the visuals and move them around to arrange them in the most useful way for your managers.
- For instance, you might want to make the sales visual larger while keeping less critical visuals smaller.

2. Change Themes:

- You can also change the overall dashboard theme to match your organization's branding or preferences. If needed, you can upload a custom **JSON theme** file, similar to the process in Power BI Desktop.

6. Pinning an Entire Page to the Dashboard

In addition to pinning individual visuals, you can also pin entire report pages to a dashboard:

1. Navigate to the Report Page:

- Go back to your report and select the page you want to pin.

2. Pin the Page:

- Click the three dots next to the page title and select **Pin to Dashboard**.
- Choose the existing dashboard (e.g., "Manager's Dashboard") and pin the entire page to it.

This allows the dashboard to show not only individual visuals but also a full-page overview, providing managers with a more comprehensive look at the data.

7. Sharing Dashboards

Once your dashboard is ready, you can share it with others in the organization:

1. Share the Dashboard:

- Click **Share** in the dashboard view.
- Enter the email addresses of the people you want to share the dashboard with. You can choose whether recipients can further share the dashboard.

2. Add a Message (Optional):

- If needed, you can include a message to explain what the dashboard contains or any important instructions for the recipients.

This makes it easy to distribute high-level insights to managers or other key stakeholders.

8. Using the Dashboard

Managers can now use the dashboard to monitor high-level metrics. If they need more detailed information, they can click on any visual in the dashboard to navigate back to the full report from which the visual was pinned. This allows them to seamlessly switch between an overview and detailed data analysis.

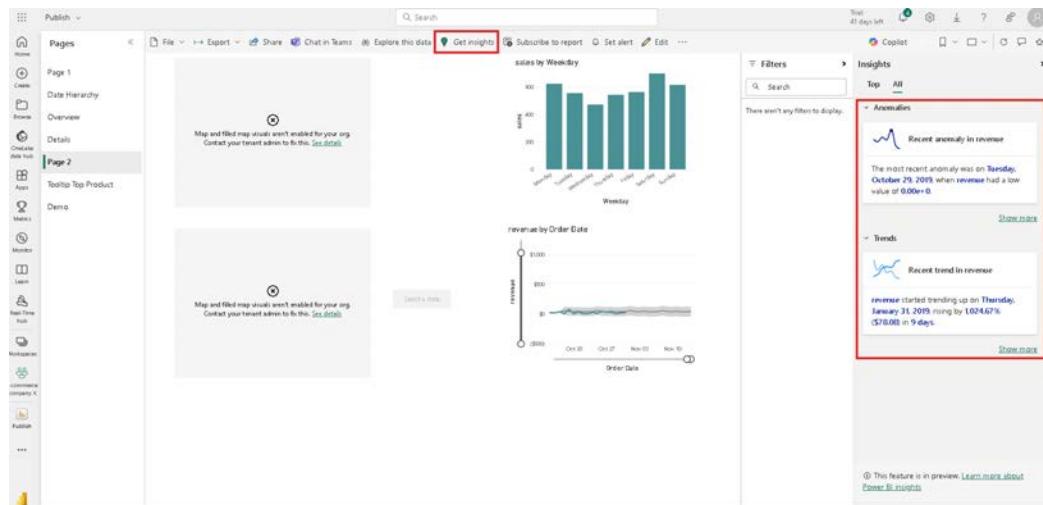
▼ Getting Quick Insights in Power BI

Power BI's **Quick Insights** feature uses artificial intelligence to automatically generate insights from your data, offering a fast way to uncover trends, anomalies, and key metrics without manual analysis.

- 1. How to Use:** Simply click on the **Get Insights** option within a report or dataset to generate AI-driven insights.
- 2. Example Insights:**
 - Revenue Analysis:** AI may highlight segments with significantly higher revenue.

The screenshot shows a Power BI report interface. At the top, there is a navigation bar with options like Publish, File, Export, Share, Chat in Team, Explore this data, Get insights, Subscribe to report, Set alert, and Edit. Below the navigation bar, there are sections for Pages, Details, and Data Hierarchy. A central area displays a table titled 'Top 3 Products' with columns: Order_ID, Product_Short, revenue, %Revenue_Split_By, width, depth, length, and unit. The table lists several products with their respective details. To the right of the table, there is a 'Filters' pane and an 'Insights' pane. The 'Insights' pane is highlighted with a red box and contains a section titled 'KPI analysis' with a 'Revenue analysis' card. The card states: 'Overall Revenue is currently at \$171,291.00. Revenue for product type EXHAIR and other segments are significantly higher than others.' At the bottom of the interface, there is a note: 'This feature is in preview. Learn more about Power BI insights.'

- Trend Analysis:** You might see patterns, such as an upward trend in revenue on specific dates.
- Anomalies:** The system can also detect unusual spikes or drops in your data.



While the insights might not always be groundbreaking, they can provide a helpful starting point for further analysis.

▼ 12- Row Level & Page Level Security

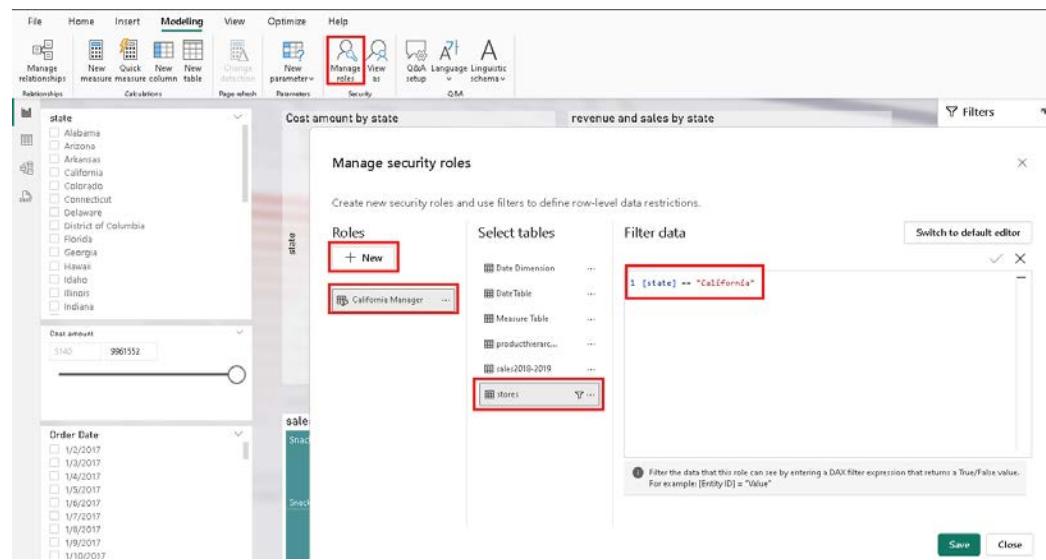
▼ Basic Row-Level Security (RLS) in Power BI

When distributing reports to multiple users, it's often necessary to restrict data visibility based on user roles. For example, managers may only need to see data relevant to their specific region. This is where **Row-Level Security (RLS)** comes in, allowing you to filter data based on user roles.

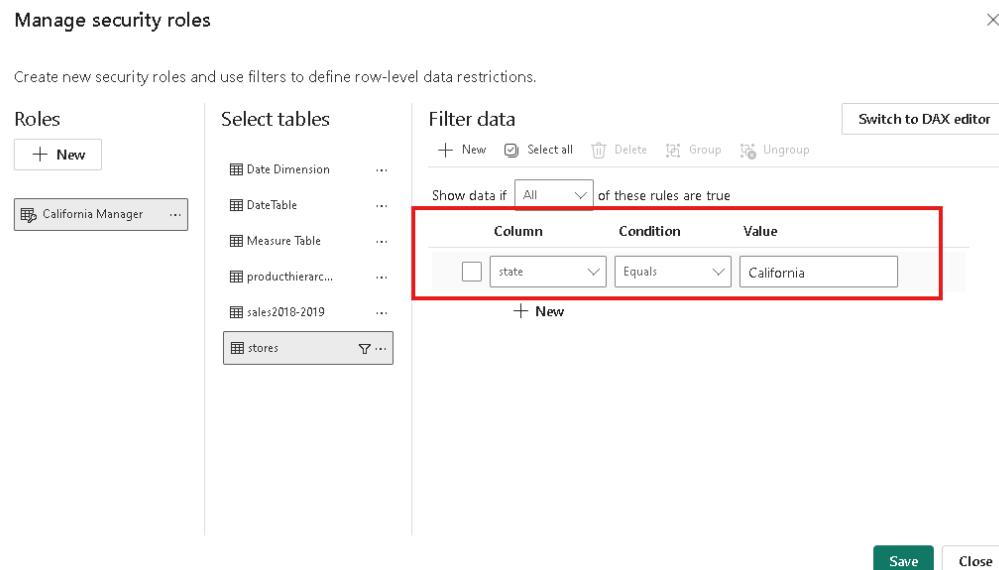
Steps to Implement RLS:

1. Create Roles in Power BI Desktop:

- Navigate to the **Model** tab and select **Manage Roles** under the **Security** section.
- Define roles using DAX expressions to filter data. For instance, to create a "California Manager" role, apply a filter to the **State** field where `State = "California"`.

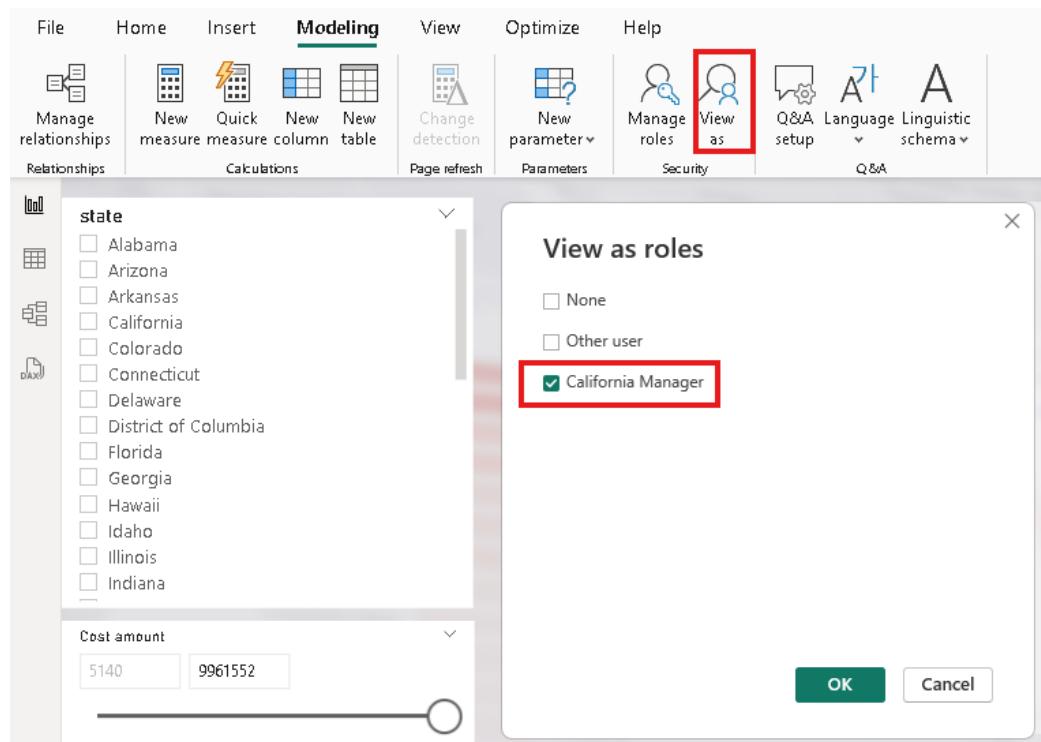


- Another method is to select "Switch the default editor". Data can also be filtered with the If block. You can switch between the two.

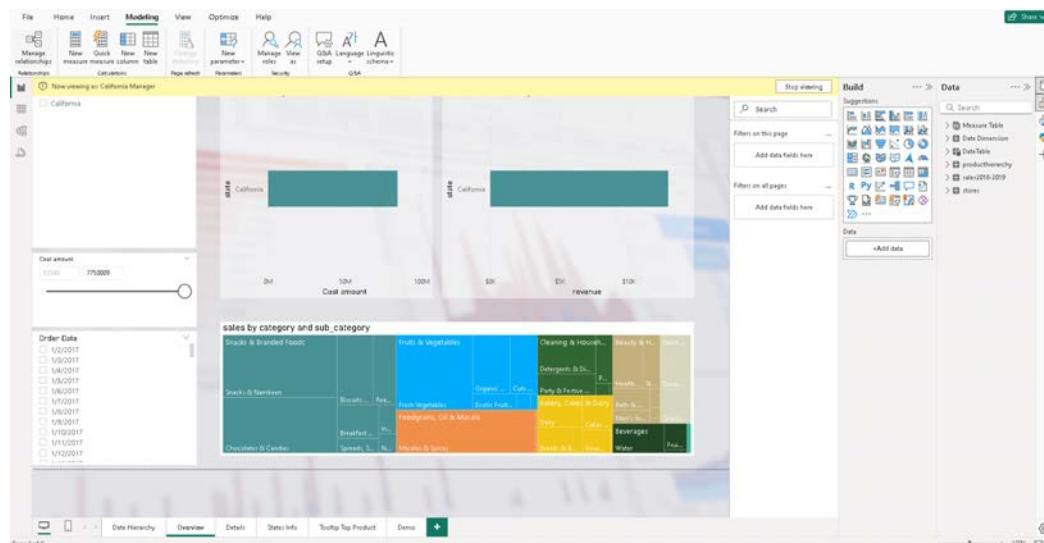


2. Test Roles:

- Use the **View As** feature to simulate the role and see how data will be filtered for that user. This ensures the role behaves as expected.



- The preview for the selected role is as shown in the image.



3. Adjust for Multiple States:

- If a manager oversees multiple regions, modify the DAX expression using the **OR** condition (||) to include multiple states (e.g., California or Texas).

Manage security roles

Create new security roles and use filters to define row-level data restrictions.

Roles

+ New

California Manager ...

Select tables

- Date Dimension ...
- DateTable ...
- Measure Table ...
- producthierarc... ...
- sales2018-2019 ...
- stores** ...

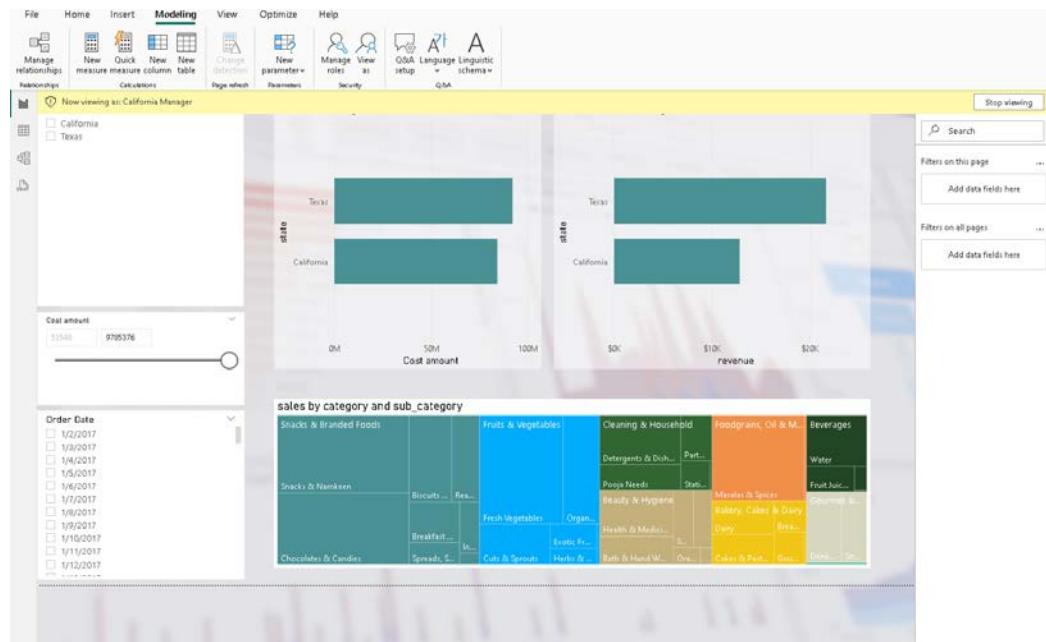
Filter data

```
1 [state] == "California" || [state] == "Texas"
```

Switch to default editor ✓ ✎

Filter the data that this role can see by entering a DAX filter expression that returns a True/False value. For example: [Entity ID] = "Value"

Save Close



4. Publish to Power BI Service:

- Once the RLS roles are set in Power BI Desktop, publish the report to Power BI Service, where you can assign roles to specific users or groups.

Row-Level Security ensures users only see the data they are authorized to view, enhancing privacy and data governance in shared reports.

▼ Row-Level Security (RLS) in the Cloud

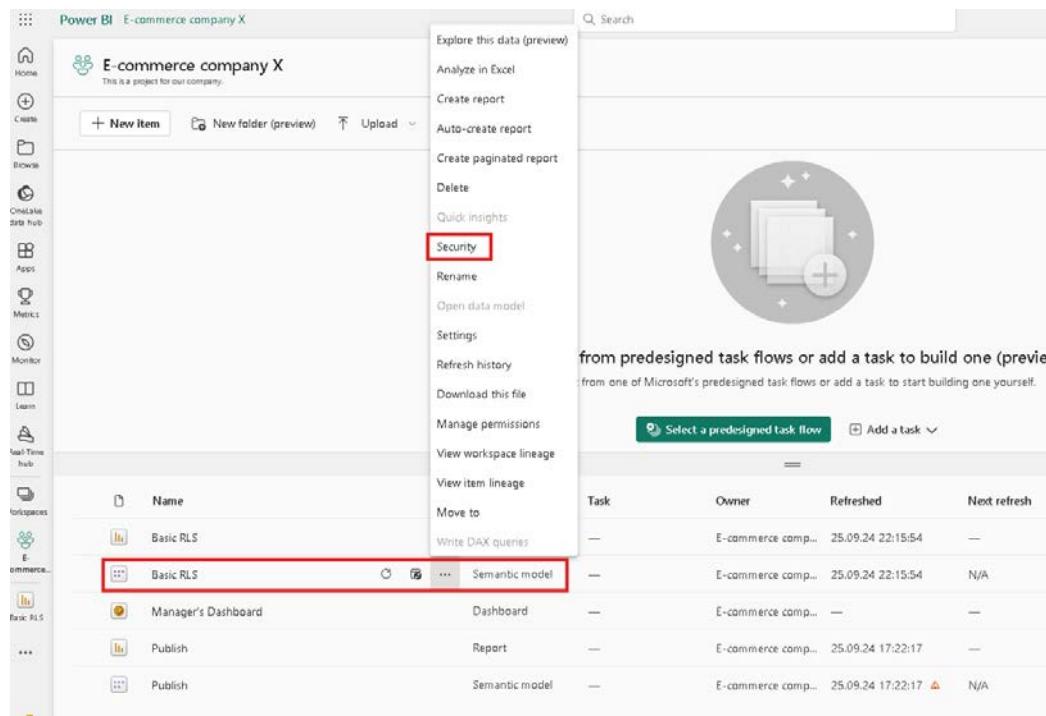
After setting up Row-Level Security (RLS) in Power BI Desktop, you can apply it in the cloud by publishing your report to Power BI Service. Here's how to configure RLS in the cloud:

1. Publish the Report:

- Publish your report with RLS to your workspace in Power BI Service. In this case, we publish it to the **e-commerce company X** workspace.

2. Assign Users to Roles:

- After publishing, go to the dataset associated with your report in the workspace.
- Click the three dots next to the dataset and select **Security**.

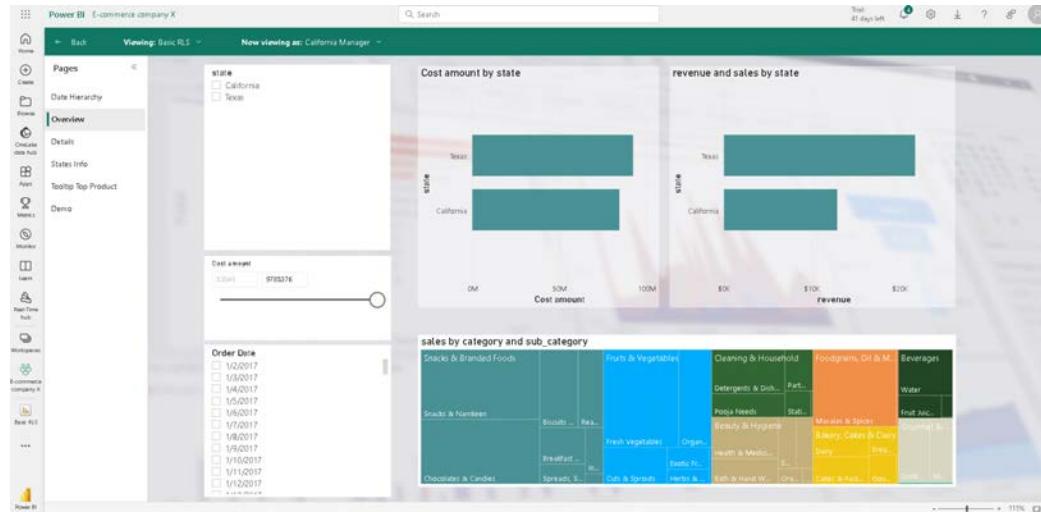


- In the security settings, you'll see the roles you defined (e.g., "California Manager"). Add users by entering their email addresses into the respective role.

The screenshot shows the Power BI service interface for managing Row-Level Security (RLS). On the left, there's a vertical sidebar with various navigation icons and links. The main content area is titled "Row-Level Security". It displays two roles: "California Manager (0)" and "Members (0)". Next to each role is a small red square icon. Below these icons is a button labeled "Test as role", which is highlighted with a red box. To the right of the "Test as role" button is a search bar containing the placeholder text "Enter email addresses" and a grey "Add" button.

3. Test the Role in Power BI Service:

- To ensure RLS is working, select the three dots next to the dataset and choose **Test as role**. This simulates how the report will look for users assigned to that role.



RLS in the cloud ensures that only authorized users can view specific data, making it a crucial tool for data privacy and governance in shared reports.

▼ Data Model & Row-Level Security in Power BI

Understanding the **data model** is crucial when implementing **Row-Level Security (RLS)** because filters in RLS are applied based on the relationships between tables. Here's a quick overview of how RLS interacts with your data model:

1. Filter Direction in Relationships:

- Filters in Power BI flow based on the defined relationships between tables. For example, if you filter by **State** in the **Stores** table, it will also filter the **Sales** table, but not the other way around due to the **one-way filter direction**.
- If you need bidirectional filtering (e.g., filtering by **Price** to affect **Stores**), you would need to set the relationship to **both directions**.

2. Potential Issues:

- If you don't account for filter directions, you may unintentionally restrict data or fail to filter as expected. For example, filtering **Stores** by **Price** wouldn't work unless you enable bidirectional filtering between **Sales** and **Stores**.

3. Dynamic RLS:

- Manually setting up RLS for each role can be time-consuming, especially in large datasets with many users. This is where **Dynamic RLS** comes in, allowing for role-based access that adjusts automatically based on the user's attributes (e.g., email address), reducing the need for static roles.

By keeping the data model in mind, you ensure that your RLS is set up correctly, filtering data as intended across related tables.

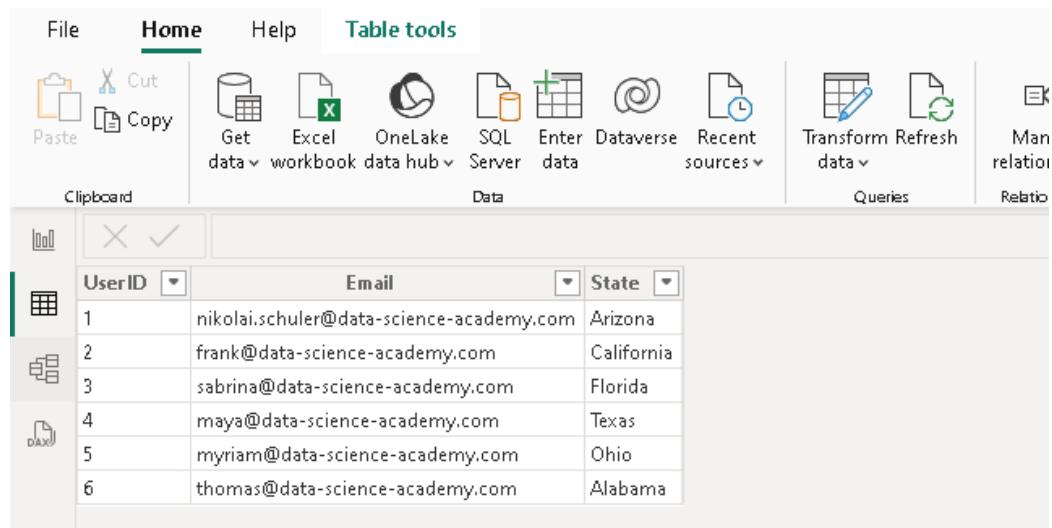
▼ Dynamic Row-Level Security (RLS) in Power BI

Dynamic Row-Level Security (RLS) is a more flexible approach to managing access to data in Power BI, especially useful for large organizations with many users. Instead of manually setting up roles for each user, dynamic RLS leverages a security table, making role assignments easier and more scalable.

Key Steps to Implement Dynamic RLS:

1. Create a Security Table:

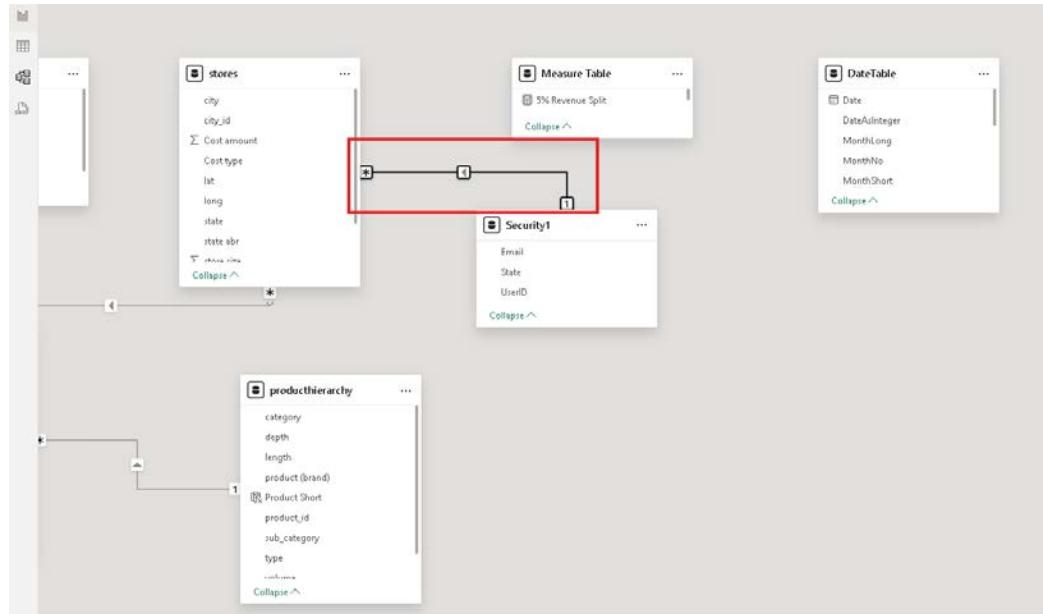
- In this table (stored in Excel or a database), list users (e.g., email addresses) and the data they can access, such as states or regions. For example, "Nikolai Shuler" can access "Arizona".



UserID	Email	State
1	nikolai.schuler@data-science-academy.com	Arizona
2	frank@data-science-academy.com	California
3	sabrina@data-science-academy.com	Florida
4	maya@data-science-academy.com	Texas
5	myriam@data-science-academy.com	Ohio
6	thomas@data-science-academy.com	Alabama

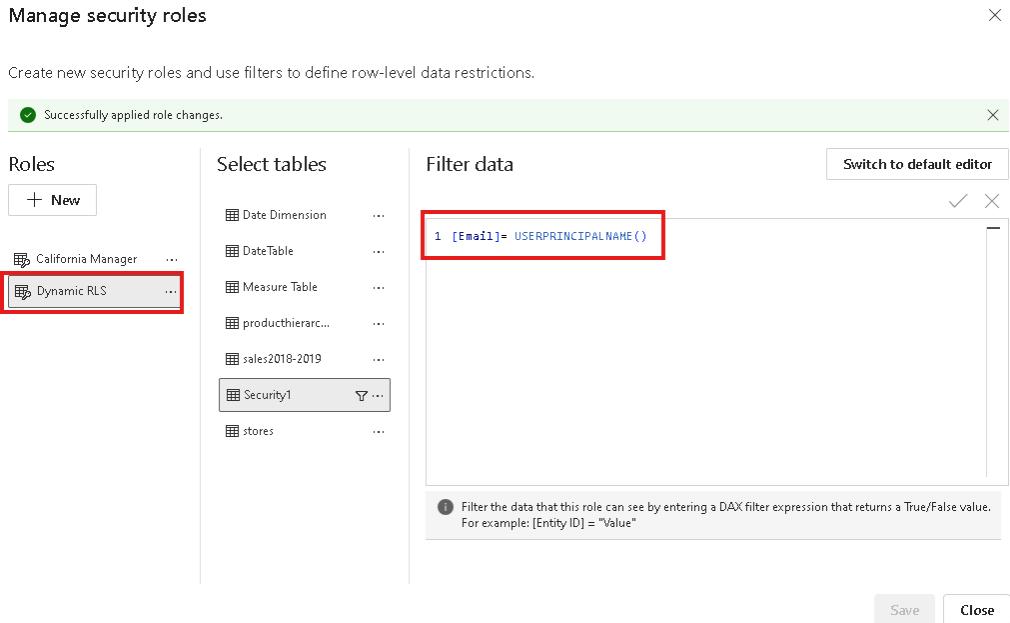
2. Connect the Security Table to the Data Model:

- In Power BI, load the security table and establish a relationship between the **state** in the security table and the **state** in your **Stores** table.



3. Use DAX for Dynamic Filtering:

- In the RLS configuration, use the DAX function `USERPRINCIPALNAME()` to dynamically filter data based on the logged-in user. This ensures each user only sees the data assigned to them in the security table.



4. Publish to the Cloud:

- After setting up dynamic RLS, publish the report to Power BI Service. Assign users to roles based on the security table, and test it by simulating the view as specific users.

Dynamic RLS simplifies data security management, especially for large organizations, by allowing easy scaling and automated access control based on user attributes.

▼ Many-to-Many Row-Level Security (RLS) in Power BI

While **Dynamic RLS** offers flexibility, one potential issue arises when dealing with many-to-many relationships. For instance, if a user like **Frank** needs access to multiple states (e.g., both **California** and **Texas**), this can create conflicts in the data model due to one-to-many relationship limitations.

Issue:

- If a user is added multiple times to the security table with access to different states (e.g., Frank is listed twice for California and Texas), a conflict occurs because the relationship between the **Security** and **Stores** tables is set to **one-to-many**.

Solution:

- Change the relationship **cardinality** to **many-to-many**. This allows multiple instances of the same value (e.g., multiple entries for Frank) on both sides of the relationship.

Steps:

1. Open the relationship between the **Security** and **Stores** tables.
2. Change the **cardinality** to **many-to-many**.
3. Ensure the filter direction is correct, typically from the **Security** table to the **Stores** table.

Once the relationship is updated, the security setup will work correctly, and you can continue adding users with multiple data access permissions.

Testing:

After publishing the updated report to Power BI Service, assign roles and test the RLS by simulating different users, such as Frank, to confirm they have access to all relevant data (e.g., both California and Texas).

By understanding and adjusting the relationship model, you can handle more complex RLS scenarios in Power BI.

▼ Page-Level Security in Power BI (Workaround)

In this scenario, the customer has a requirement to restrict access to certain pages for specific users, creating what we call **Page-Level Security**.

Unfortunately, Power BI does not support this as a built-in feature. As a workaround, we need to design custom page navigation tied to user roles to limit access to particular pages.

Approach Outline:

1. Understanding the Requirement:

Some users, like Frank, should only see the Overview page, while others, like Myriam, should have access to multiple pages. This data will be provided in a table detailing user-page relationships.

	A	B	C
1	Email	Page Access	
2	frank@data-science-academy.com	Overview	
3	myriam@data-science-academy.com	Details	
4	myriam@data-science-academy.com	Overview	
5			

2. Incorporating the User-Page Table:

- Import the table into Power BI by selecting the **Excel Workbook** where the table (e.g., "Page Level Security") is stored.

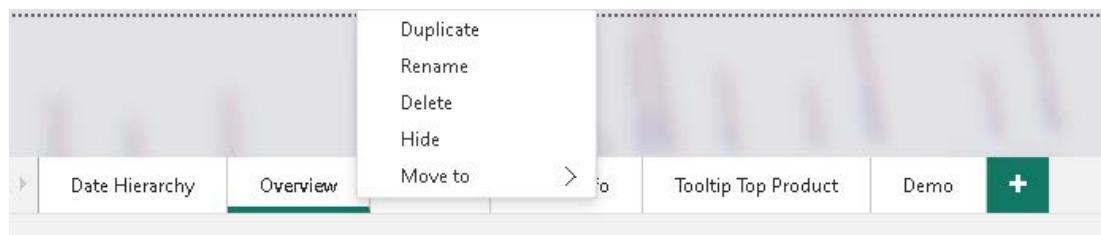
The screenshot shows the Power BI Navigator interface. On the left, there is a search bar and a 'Display Options' dropdown. Below these, a list of files is shown, with 'Page-Level+Security.xlsx [1]' expanded to reveal a table named 'PLS'. This table is selected, indicated by a checked checkbox next to its name. To the right of the table preview, there is a preview of the 'PLS' data, which is identical to the table shown at the top of the page. At the bottom right of the Navigator, there are three buttons: 'Load' (highlighted in green), 'Transform Data', and 'Cancel'.

Column1	Column2
Email	Page Access
frank@data-science-academy.com	Overview
myriam@data-science-academy.com	Details
myriam@data-science-academy.com	Overview

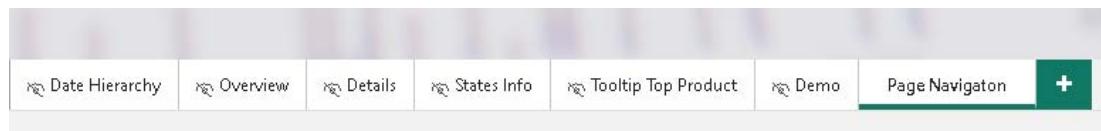
- Perform basic transformations, such as setting the first row as the header, and close and apply the changes.

3. Building Custom Page Navigation:

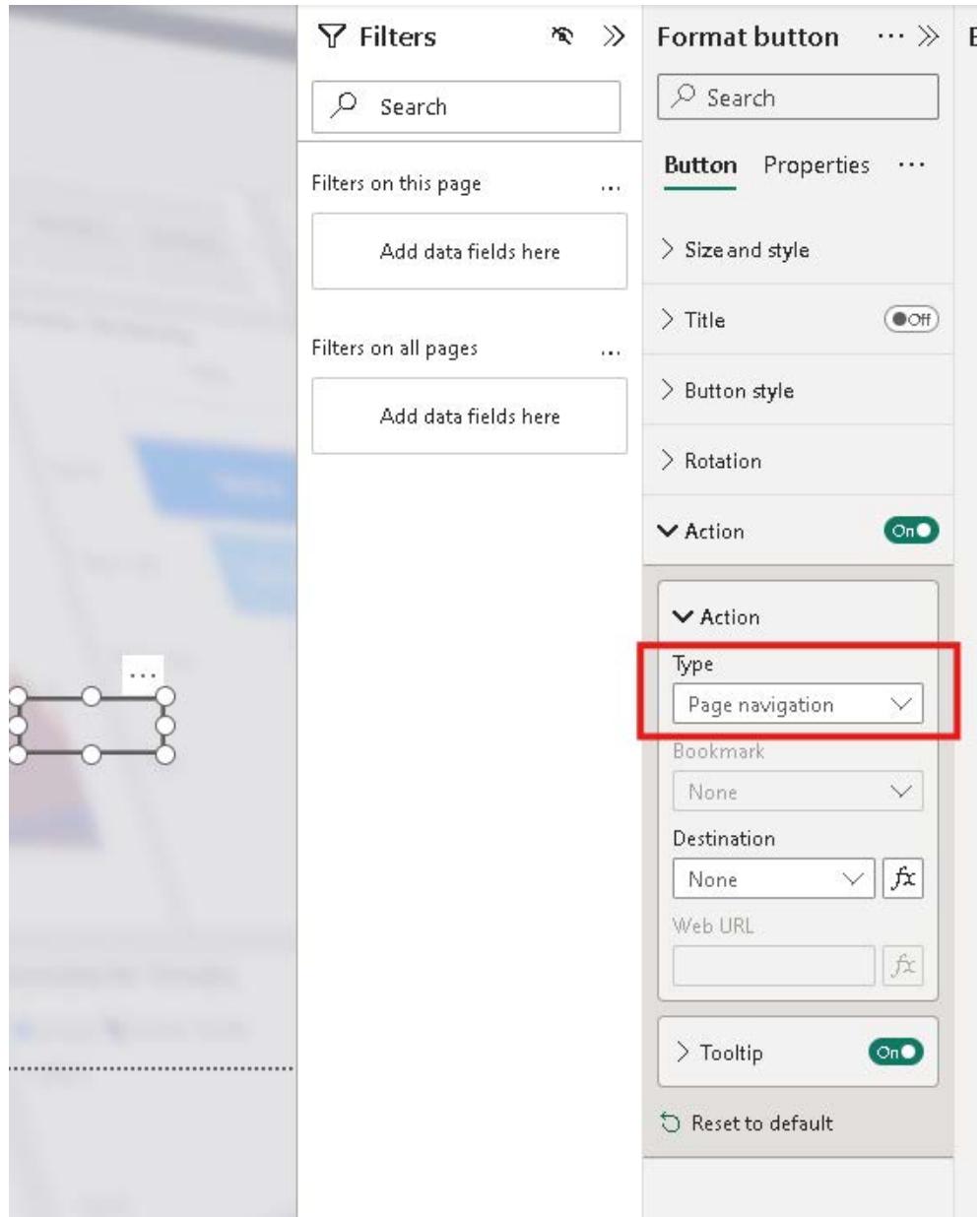
- Duplicate an existing page to create a **Page Navigation** section.



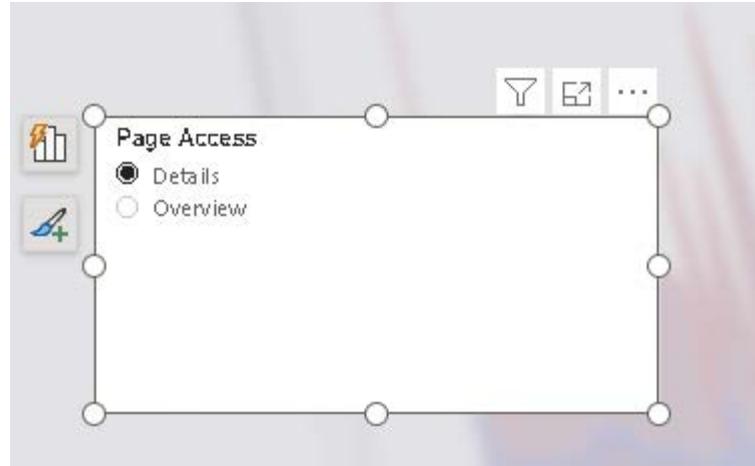
- Hide all pages by right-clicking on the page tabs and selecting **Hide Page**.



- Use buttons to enable navigation:
 - Insert a **Button** and add an **Action** of type **Page Navigation**.



- Since static navigation isn't sufficient, we base the button navigation on a field, like **Page Access**.
- Implement a **Page Access** filter to allow users to select which page they can navigate to, ensuring only single selection is allowed (as it wouldn't make sense to navigate to multiple pages at once).

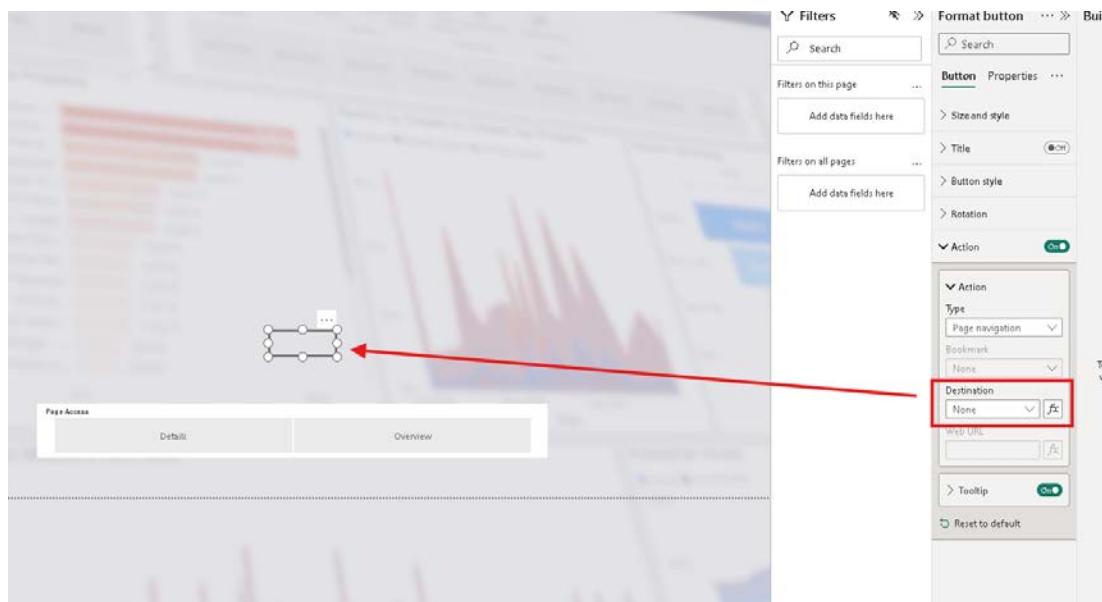


4. Enhancing User Experience:

- Design the **Page Access** filter in a horizontal orientation to improve the UI.

A screenshot of a Power BI page. On the left, there is a horizontal filter panel titled "Page Access" with two buttons: "Details" (selected) and "Overview". To the right of the filter panel, there is a large, blurred chart area. On the far right, there is a "Slicer settings" pane open, showing various options like "Style" (set to "Tile"), "Multi-select with C...", and "Show 'Select all' o...".

- Add a label to the button, such as "Go to selected page," to clarify its function.



Destination - Action - Action

Format style

Field value

What field should we base this on?

First Page Access

Search

> Measure Table

> Date Dimension

> DateTable

> PLS

Email

Page Access

> producthierarchy

> sales2018-2019

> Security1

> stores

Summarization

First

[Learn more about conditional formatting](#)

OK

Cancel

- Enable transparency on the button's background for a sleeker design.

5. Implementing Role-Level Security:

- Integrate role-level security to ensure only authorized users see the appropriate pages. This is done through DAX filters:
 - Go to **Modeling > Manage Roles**.

- o Apply a filter on the email field using the DAX function

USERPRINCIPALNAME()

The screenshot shows the 'Manage roles' screen in Power BI. On the left, there's a list of roles: 'Dynamic RLS' (selected), 'Create', and 'Delete'. In the center, there's a list of tables: 'Date Dimension', 'DataTable', 'Measure Table', 'PLS' (selected), 'producthierarchy', 'sales2018-2019', 'Security1', and 'stores'. On the right, a modal window titled 'Table filter DAX expression' is displayed. It contains the DAX expression '[Email] = USERPRINCIPALNAME()' and a note: 'Filter the data that this role can see by entering a DAX filter expression that returns a True/False value. For example: [Entity ID] = "Value"'.

- Once done, publish the changes to the cloud and test the role-level access.

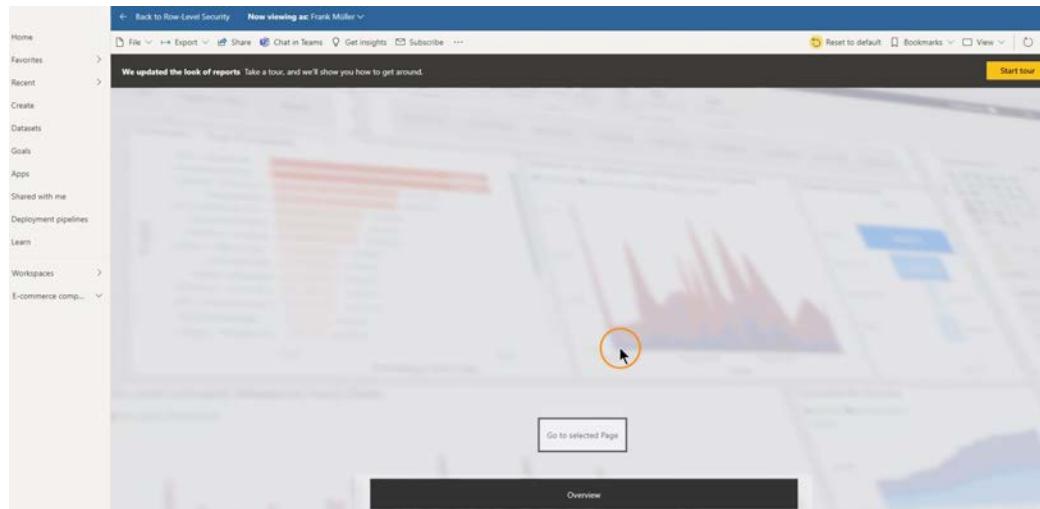
6. Testing and Deployment:

- After publishing, verify by testing with a specific user (e.g., Frank). If implemented correctly, Frank will only have access to the Overview page.

Row-Level Security

The screenshot shows the 'Members (1)' section of the 'Dynamic RLS' configuration. It includes a 'Test' button (circled in orange), an input field 'Enter email addresses' containing 'Frank Müller', and an 'Add' button.

- To include this in an app, ensure only the relevant report is added, and update the app accordingly.



Considerations:

While this workaround provides a solution, it is not entirely secure. Hidden pages are still accessible if someone shares a direct link. If the data is sensitive, it's better to separate the reports for different user groups rather than relying solely on this method.

▼ 13- M Language Fundamentals

▼ Goals

Today, we will be covering an **advanced topic**: the **M language**. While this may not be something essential for everyday use as a Power BI data analyst, understanding M can be a valuable skill, providing you with deeper insights into how Power BI handles data transformations behind the scenes. Think of it as a **bonus** topic that can enhance your Power BI capabilities.

What is M Language?

M language is the **data transformation language** that powers the **Power Query** editor in Power BI. Every step you apply in Power Query (filtering, merging, or transforming data) is ultimately written in M. By learning the fundamentals of M, you'll be able to:

- **Understand the code behind applied steps** in Power Query.
- **Modify and customize** transformations directly in the M code, giving you more control and flexibility.
- **Write your own basic M code** for custom data transformations.

What Will You Learn ?

Specifically, you will:

- Understand the **basics** of the M language and its structure.
- Gain confidence in **reading** and **modifying** M code to adjust existing transformations.
- Write basic **custom transformations** using M, helping you tackle unique scenarios that aren't directly available in the Power Query user interface.

Remember, this is a **highly advanced** topic and not something you need to fully master for general Power BI usage. The goal is to introduce you to the fundamentals so that if you ever encounter M code, you can confidently understand and make basic modifications.

▼ What is M Language?

M Language, also known as Power Query M, is a data transformation language developed by Microsoft specifically for Power Query in Power BI, Excel, and other Microsoft products. It operates behind the scenes, automating the data transformation steps performed in Power BI's Query Editor. Whenever you apply transformations like filtering, renaming, or changing data types, M Language automatically generates the necessary formulas to execute these transformations.

The screenshot shows the Power BI Query Editor interface. On the left is a table with columns: product_id, product_name (brand), type, length, depth, width, and category. The 'category' column contains values like Fruits & Vegetables, Beauty & Hygiene, Snacks & Sweets, and Bakery, Cakes & Dairy. On the right, the 'Query Settings' pane is open, showing the 'APPLIED STEPS' section. One step is highlighted with a yellow background: 'Changed Type1'. This step corresponds to the formula in the formula bar: = Table.AddColumn(#"Changed Type", "volume", each [length]*[depth]*[width]).

Though Power BI users typically interact with the graphical interface in the Query Editor, which simplifies the transformation process, understanding M Language offers significant advantages for advanced users. These advantages include:

- **Enhanced Understanding:** Knowing how M Language works deepens your comprehension of Power BI's Query Editor and its functionality.
- **Efficient Shortcuts:** By working directly with M Language, users can create custom transformations more quickly and efficiently, avoiding the limitations of the graphical interface.

Why Learn M Language?

Although it's possible to use Power BI professionally without directly interacting with M Language, understanding its fundamentals can provide more flexibility and control over data transformation processes. For example:

- **Customizing Data Transformations:** Imagine you need to change a column's data type from an integer to a decimal. Instead of reapplying transformations via the interface, you can modify the M Language formula directly in the formula bar. This allows you to make adjustments more quickly and precisely.

- Before:

```
= Table.TransformColumnTypes(#"Renamed Columns1", {"length", Int64.Type}, {"depth", Decimal.Type}, {"width", Int64.Type})
```

A _C product_id	A _C product (brand)	A _C type	i ₂ ₃ length	i ₂ ₃ depth
1 P0000	Serum (Livon)	Indian & Exotic Herbs	5	20
2 P0001	Hand Wash - Moisture Shield (Savlon)	Hair Oil & Serum	14	22
3 P0002	Good Day Butter Cookies (Britannia)	Hand Wash & Sanitizers	22	40
4 P0004	Happy Happy Choco-Chip Cookies (Parle)	Cookies	2	13
5 P0005	50-50 Timepass Salted Biscuits (Britannia)	Glucose & Milk Biscuits	16	30
6 P0006	Tiger Elaichi Cream Biscuits (Britannia)	Salted Biscuits	8	15
7 P0007	Bounce Biscuits - Choco Creme (Sunfeast)	Glucose & Milk Biscuits	2	22
8 P0008	50-50 Timepass Biscuits (Britannia)	Cream Biscuits & Wafers	5	16
9 P0009	Tiger Chocolate Cream Biscuits (Britannia)	Salted Biscuits	5	18
10 P0010	Biscuits - Magix Kreams Choc (Parle)	Cream Biscuits & Wafers	2	22
11 P0011	Dreams Cup Cake - Choco (Elite)	Cream Biscuits & Wafers	9	22
12 P0012	Layer Cake - Chocolate (Winkies)	Muffins & Cup Cakes	10	20

- After:

```
= Table.TransformColumnTypes(#"Renamed Columns1", {"length", Int64.Type}, {"depth", Decimal.Type}, {"width", Int64.Type})
```

A _C product (brand)	A _C type	i ₂ ₃ length	i ₂ ₃ depth
Serum (Livon)	Indian & Exotic Herbs	5	20
Hand Wash - Moisture Shield (Savlon)	Hair Oil & Serum	14	22

- Error Handling:** M Language also helps when troubleshooting. If an error occurs due to a column name change, such as renaming "Volume" to "VOLUME", the query might break. Understanding the formula allows you to correct such issues quickly by editing the column references directly in the M code.

- Before:

```
= Table.AddColumn(#"Changed Type", "volume", each [length]*[depth]*[width])
```

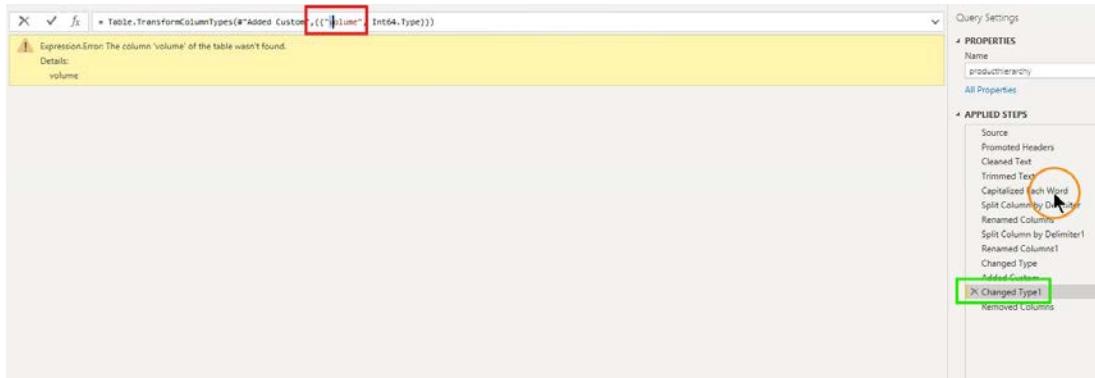
- After:

```
= Table.AddColumn(#"Changed Type", "volume", each [length]*[depth])
```

A _C category	A _C sub_category	A _C Volume
Herbs & Spices	Herbs & Seasonings	100
2 Serum	20 Beauty & Hygiene	297
3 H & Sanitizers	22 Beauty & Hygiene	26
4	4 Snacks & Sweets	880
5 Milk Biscuits	16 Snacks & Sweets	127.5
6 Wafers	15 Snacks & Sweets	44
7 Milk Biscuits	20 Snacks & Sweets	80
8 Wafers & Wafers	5 Snacks & Sweets	90
9 Wafers	14 Snacks & Sweets	44
10 Wafers & Wafers	3 Snacks & Sweets	26
11 Wafers & Wafers	15 Snacks & Sweets	188
12 Cup Cakes	12 Bakery, Cakes & Dairy	200
13 & Slice Cakes	14 Bakery, Cakes & Dairy	108.25
14 & Slice Cakes	32 Bakery, Cakes & Dairy	195
15	10 Snacks & Sweets	330
16	8 Snacks & Sweets	36
17 Indy & Lollypop	8 Snacks & Sweets	224
18 chewing Gum	11 Snacks & Sweets	14
19 chewing Gum	32 Snacks & Sweets	63
20 chewing Gum	16 Snacks & Sweets	140

- Reusable Steps:** In cases where you work with similar data sets, M Language enables you to easily replicate transformation steps across

multiple tables. Instead of manually applying transformations to a new table with a similar structure, you can reuse or modify the M code to save time.



How M Language Works in Power BI

- Automatic Formula Generation:** Each time a transformation is applied through the Query Editor, M Language generates a corresponding formula. These formulas execute the transformation without requiring manual intervention.
- Manual Adjustments in the Formula Bar:** Power BI allows users to edit M Language formulas directly in the formula bar. This is useful for refining data transformations, such as modifying column types or adjusting calculations without redoing steps in the editor interface.
- Error Messages and Fixes:** When changes in the query steps cause errors, such as when a column is renamed or removed, M Language helps diagnose and fix these issues. You can backtrack and update the query to reflect the correct column names or data references, allowing for smoother query execution.

Key Benefits of Learning M Language

- Flexibility:** M Language provides flexibility to perform complex data transformations that are difficult or impossible through the standard Query Editor interface.
- Efficiency:** It allows advanced users to streamline repetitive tasks, such as applying transformations to similar datasets, saving time and

reducing manual effort.

- **Control:** By working with M Language, you gain finer control over the data transformation process, enabling you to optimize queries and handle edge cases more effectively.

▼ Basics About the Advanced Editor

In Power BI, the **Advanced Editor** is a powerful feature that allows users to manipulate and fine-tune data transformation processes directly through the M Language. This editor displays the M Language code generated by Power BI's Query Editor when you perform actions such as loading data, removing rows, or renaming columns. By using the Advanced Editor, you gain more control and flexibility over how data transformations are applied, especially when working with similar datasets or making bulk changes.

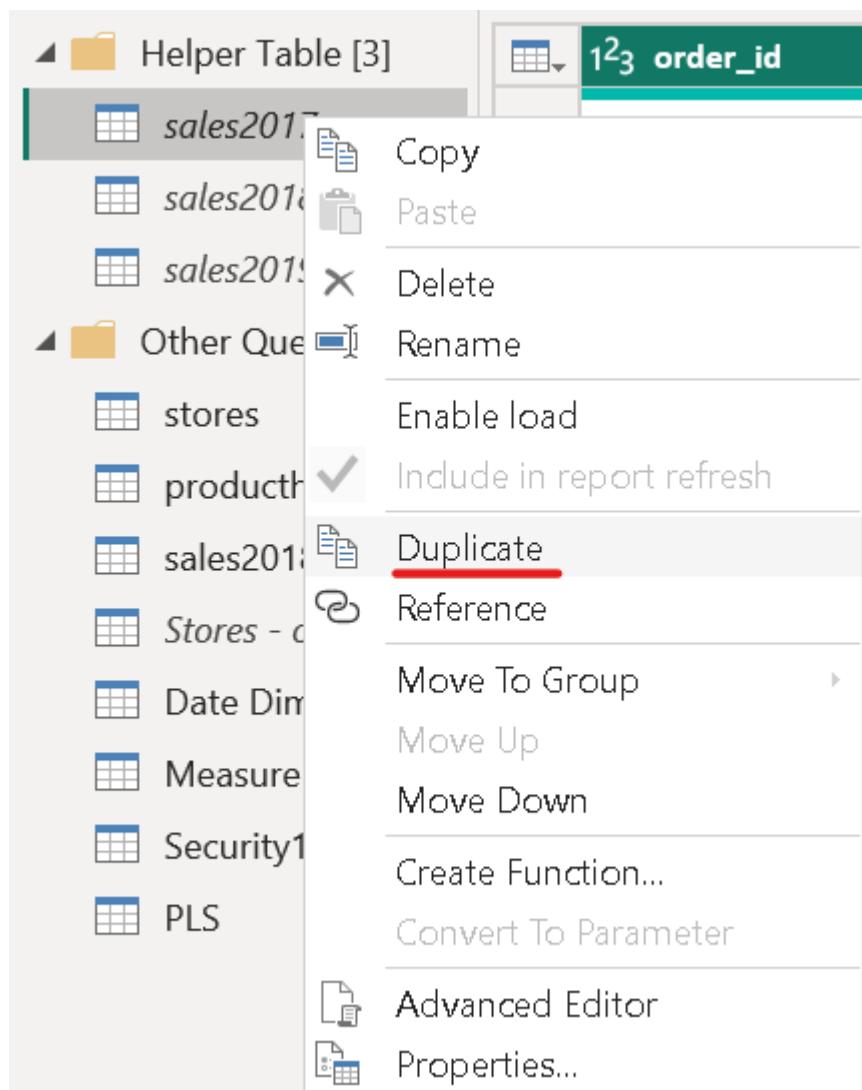
Example Scenario: Importing a New File

Let's consider a practical example where we want to import a new file named *Sales 2020 raw*, which has the same structure as an existing file, *Sales 2017*. Instead of reapplying all the steps from scratch, we can use the Advanced Editor to duplicate and adjust the existing query, making the process faster and more efficient.

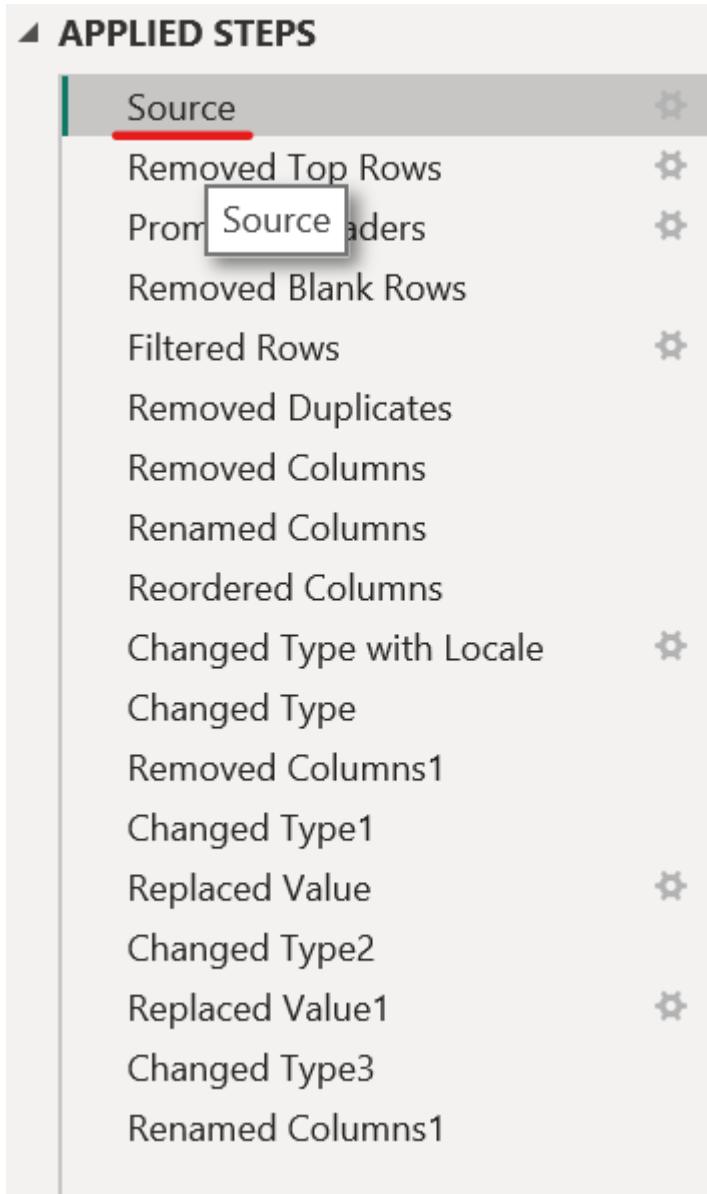
Here's how it works:

1. Duplicating the Query:

- First, you can duplicate the query used for the *Sales 2017* data.



- Select Source.



2. Adjusting the Data Source:

- Open the **Advanced Editor** for the duplicated query.
- In the M code, locate the step that defines the data source (usually labeled **Source**).
- Update the file path and file name to point to the new *Sales 2020 raw* file. For example:

```
 Csv.Document(File.Contents("C:\Users\bizay\Desktop\powerbi\day4\M-language\sales2020_raw.csv"),[Delimiter=",", Columns=17, Encoding=1252, QuoteStyle=QuoteStyle.None])
```

- Ensure that the file path is enclosed in quotation marks, and the file extension is correct.

3. Applying the Same Transformations:

- Since the *Sales 2020* file has the same structure as *Sales 2017*, all subsequent transformation steps (e.g., removing top rows, renaming columns) will be automatically applied to the new data without needing further modification.

4. Appending Data:

- If you need to append the new *Sales 2020* data to an existing table (e.g., *Sales 2017*), navigate to the query where the data is appended. In the M code for that query, you can update the relevant step to include the new table:



- After this, all transformations are applied, and the new data becomes part of the consolidated dataset.

5. Finalizing the Query:

- After making these adjustments in the Advanced Editor, click **Close and Apply**. Power BI will update the dataset, including the new data from *Sales 2020*.

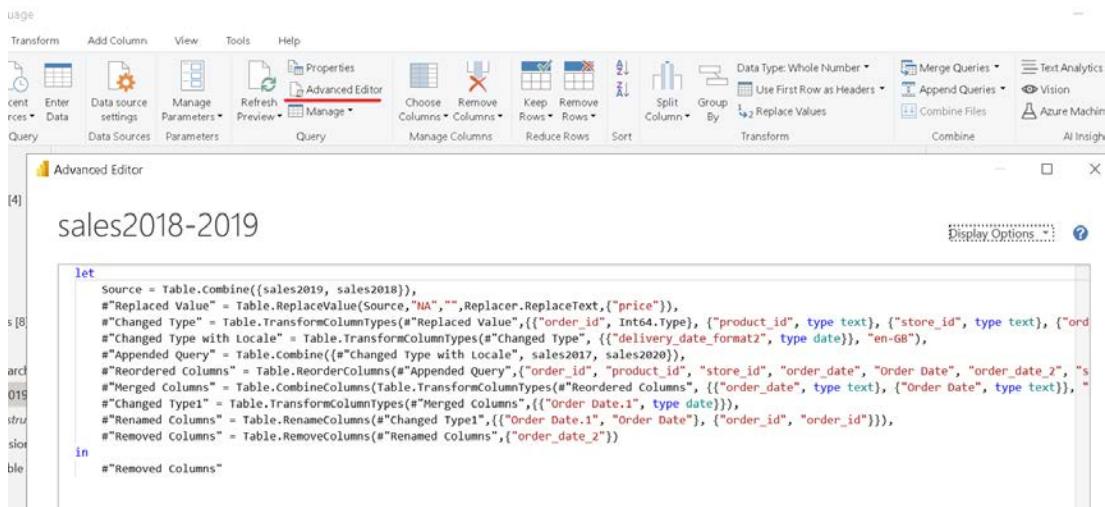
Understanding the M Language Structure in the Advanced Editor

The Advanced Editor uses a structured format to define how data is processed. The key components are:

- **let** and **in** **Blocks:**

Every query begins with a **let** block and ends with an **in** block. The **let** block contains the steps or transformations, and the **in** block specifies the final output, usually the result of the last step.

Example:



```

let
    Source = Table.Combine({sales2019, sales2018}),
    #"Replaced Value" = Table.ReplaceValue(Source, "NA", "", Replacer.ReplaceText,{"price"}),
    #"Changed Type" = Table.TransformColumnTypes("Replaced Value",{{"order_id", Int64.Type}, {"product_id", type text}, {"store_id", type text}, {"ord
    #"Changed Type with Locale" = Table.TransformColumnTypes(#"Changed Type", {"delivery_date_format2", type date}), "en-GB"),
    #"Appended Query" = Table.Combine({#"Changed Type with Locale", sales2017, sales2020}),
    #"Reordered Columns" = Table.ReorderColumns("Appended Query",{"order_id", "product_id", "store_id", "order_date", "Order Date", "order_date_2", "s
    #"Merged Columns" = Table.CombineColumns(Table.TransformColumnTypes(#"Reordered Columns",{{"order_date", type text}, {"Order Date", type text}}), "o
    #"Changed Type1" = Table.TransformColumnTypes(#"Merged Columns",{{"Order Date.1", type date}}),
    #"Renamed Columns" = Table.RenameColumns(#"Changed Type1",{{"Order Date.1", "Order Date"}, {"order_id", "order_id_2"}}),
    #"Removed Columns" = Table.RemoveColumns(#"Renamed Columns", {"order_date_2"})
in
    #"Removed Columns"

```

- **Steps and Variables:**

Each transformation step is stored in a variable (e.g., **Source**, **RemovedTopRows**, **PromotedHeaders**). These variables represent the data at various stages of transformation and are listed sequentially. The result of one step becomes the input for the next.

- **Referencing Variables:**

Steps can refer to previous variables. For example, in the line

`RemovedTopRows = Table.Skip(Source, 1)`, the **Source** variable (which holds the original data) is used as input for the `Table.Skip` function, which removes the top row.

- **Naming Conventions:**

Variable names must not contain spaces. If you wish to use names with spaces, you need to enclose them in quotation marks and precede them with a hash symbol (`#`).

- **Commas:**

Each transformation step ends with a comma, except for the final step.
This helps separate each step in the query process.

Working with Data Types and Variables

In M Language, understanding key data types like **values**, **lists**, **tables**, and **functions** is crucial when working in the Advanced Editor. These elements allow you to manipulate data efficiently and perform advanced operations. Here's a brief overview:

- **Values:** Represent individual data points, such as a single number or string.
- **Lists:** Ordered sequences of values, such as a list of dates or numbers.
- **Tables:** Structured data, much like an Excel table, with rows and columns.
- **Functions:** Reusable blocks of code that perform specific tasks, such as filtering rows or changing data types.

In conclusion, mastering the Advanced Editor in Power BI, and the underlying M Language, provides a robust way to customize, optimize, and automate data transformations, saving time and improving the accuracy of your data analytics processes.

▼ Object Types (Values & Lists in M Language)

In Power BI's M Language, understanding the foundational data types is essential for working with data transformations. The key object types include **Single Values**, **Lists**, and **Tables**. While tables are the most common data type used in Power BI, it's important to first grasp the concepts of **Values** and **Lists** before progressing to more complex structures like tables and functions.

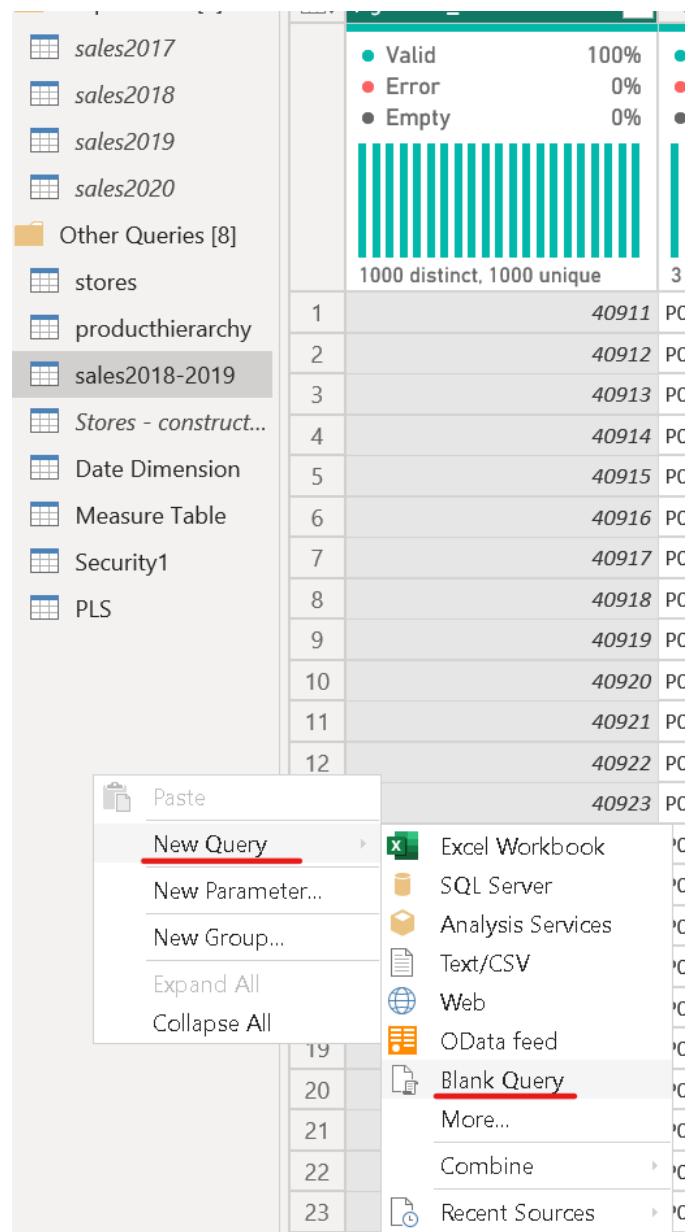
1. Single Values

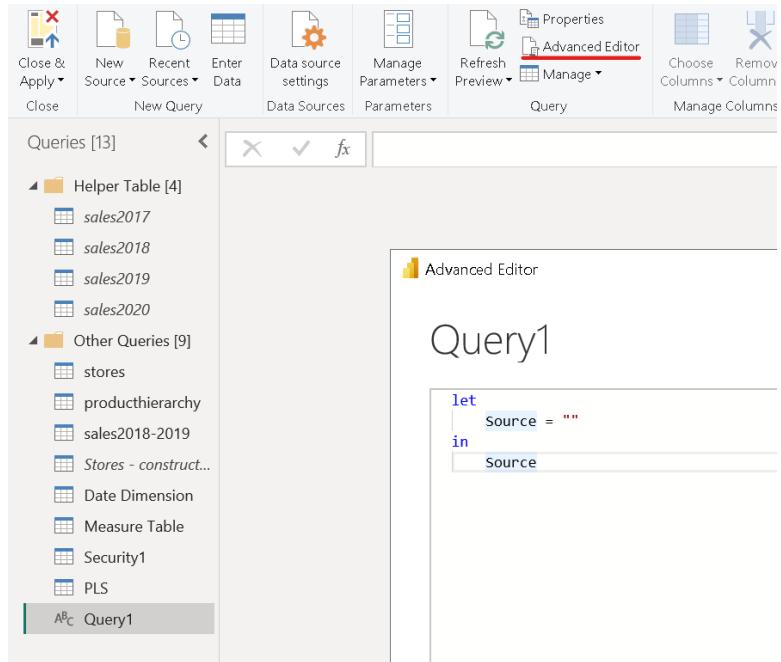
A **Value** in M Language is the simplest form of data, representing a single entity. Values can be numbers, text, or logical types (true/false). They are often used as individual elements in calculations or transformations, though less commonly than lists or tables.

Example of a Value:

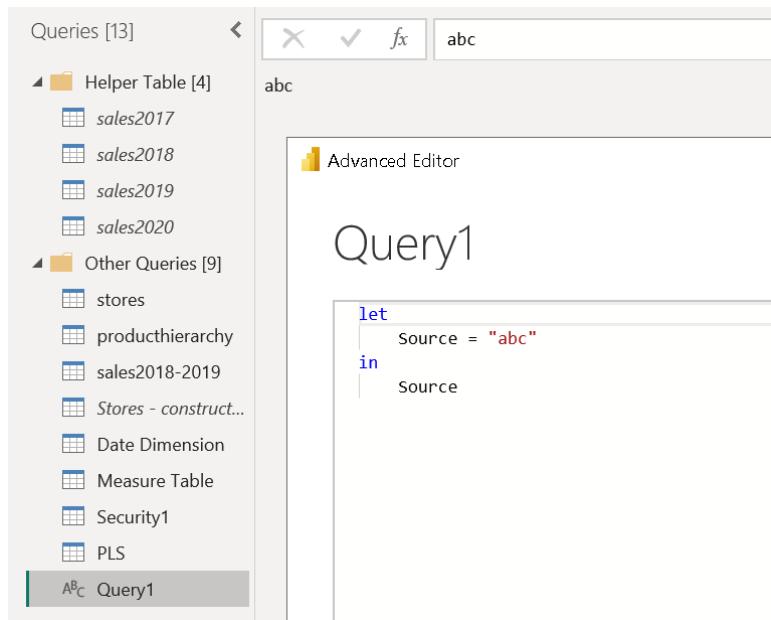
To understand how values work, let's create a blank query in Power BI:

1. Right-click and select **New Query > Blank Query**.





2. Open the **Advanced Editor** to write M code. You can create a value as follows: This is a simple text value `"ABC"`. When you run this query, Power BI will display the value "ABC". Similarly, you can use numbers:



The screenshot shows the Power BI Advanced Editor interface. On the left, there is a tree view of 'Queries [13]' containing two main categories: 'Helper Table [4]' and 'Other Queries [9]'. Under 'Helper Table [4]', there are four items: 'sales2017', 'sales2018', 'sales2019', and 'sales2020'. Under 'Other Queries [9]', there are nine items: 'stores', 'producthierarchy', 'sales2018-2019', 'Stores - construct...', 'Date Dimension', 'Measure Table', 'Security1', and 'PLS'. At the bottom of the tree view, 'Query1' is selected. The main workspace on the right is titled 'Query1' and contains the following M Language code:

```
let
    Source = 23
in
    Source
```

While values may seem basic, they are foundational for certain functions and transformations. For example, numeric operations can only be applied to numeric values.

Applying Functions to Values:

M Language offers many functions that can operate on values. For example, you can use the `Number.Sign` function to determine if a number is positive or negative:

The screenshot shows the Power BI Advanced Editor with a query titled 'Query1'. The code is as follows:

```
let
    Source = Number.Sign(-23)
in
    Source
```

A tooltip for the `Number.Sign` function is displayed, showing its definition: `Number.Sign(number as nullable number) => ...` and the note: "Returns 1 if the number is positive, -1 if it is negative, and 0 if it is zero."

This will return `-1` because the value is negative. Functions like these help you perform operations directly on values.

2. Lists

A **List** is an ordered collection of values. Unlike a single value, a list can hold multiple values of different types, such as numbers, text, or even other lists. Lists are useful for working with sequences of data, and many M Language functions operate on lists.

Creating a List:

Lists in M Language are defined using curly brackets `{}`. Here's an example of a list with mixed data types:

The screenshot shows the Power BI Advanced Editor interface. On the left, the 'Queries [13]' pane is open, displaying a list of queries under 'Helper Table [4]' and 'Other Queries [9]'. The 'Query1' query is selected and highlighted in grey. In the center, there is a preview pane titled 'List' showing three rows of data: 1, -1; 2, 5; and 3, test. Above the preview, the M code for the query is displayed:

```
let
    Source = {Number.Sign(-23), 5, "test"}
in
    Source
```

In this case, the list contains three numbers and a text value, `"test"`. The output will be displayed as a list in Power BI.

Using a Function Inside a Lists:

Using the Text.Upper function in a list.

The screenshot shows the Power BI Query Editor interface. On the left, the 'Queries [13]' pane lists various queries like 'Helper Table [4]', 'sales2017', etc., and 'Query1' which is currently selected. On the right, the main area shows a table titled 'List' with three rows:

	List
1	-1
2	5
3	TEST

Below the table is the 'Advanced Editor' pane containing the following M language code:

```
let
    Source = {Number.Sign(-23), 5, Text.Upper("test")}
in
    Source
```

Applying Functions to Lists:

You can apply various list functions in M Language. For example, if you want to apply the `List.Sum` function to a list of numbers, you can do it as follows:

The screenshot shows the Power BI Advanced Editor interface. On the left, the 'Queries [13]' pane is open, displaying a tree structure of queries. Under 'Helper Table [4]', there are four entries: 'sales2017', 'sales2018', 'sales2019', and 'sales2020'. Under 'Other Queries [9]', there are nine entries: 'stores', 'producthierarchy', 'sales2018-2019', 'Stores - construct...', 'Date Dimension', 'Measure Table', 'Security1', 'PLS', and 'Query1'. The 'Query1' entry is highlighted with a grey background. At the top right of the editor, there is a formula bar with the text '= List.Sum(Source)' and a result value of '15'. Below the formula bar, the title 'Query1' is displayed in large font. The main code area shows the following DAX code:

```
let  
    Source = {1, 2, 3, 4, 5},  
    SumOfList = List.Sum(Source)  
in  
    SumOfList
```

This query will return the sum of the list, which is 15.

Using Values and Lists Together

It's essential to understand how values and lists interact. Lists can contain values, and many list functions are designed to work on individual values within a list. For example, you might want to transform each item in a list to uppercase letters if they are strings:

```
let  
    Source = {"abc", "def", "ghi"},  
    UppercaseList = List.Transform(Source, Text.Upper)  
in  
    UppercaseList
```

This query will return a list with all text values transformed to uppercase:

```
{"ABC", "DEF", "GHI"}.
```

In M Language, **Values** and **Lists** are the building blocks for data manipulation. Values represent single pieces of data, while lists provide ordered collections of values that can be transformed using list functions.

▼ Table Functions & Transforming Columns in M Language

In Power BI, tables are one of the most important object types, as they represent structured data and are the foundation for most transformations. Understanding how to use **table functions** to manipulate and transform tables is crucial for efficient data processing. In this section, we'll focus on the `Table.TransformColumns` function and its application in transforming column values. We'll also introduce the `Table.AddColumn` function, which is essential for adding new columns based on transformations.

Using Lists to Create Tables:

Lists can also be transformed into tables. If you want to transform a list into a table, you would use the `Table.FromList` function:

The screenshot shows the Power BI ribbon with the 'Transform' tab selected. In the 'Convert' section, the 'To Table' button is highlighted with a red underline. Other options like 'Keep Items' and 'Remove Items' are also visible.

Queries [13]

- Helper Table [4]**
 - `sales2017`
 - `sales2018`
 - `sales2019`
 - `sales2020`
- Other Queries [9]**
 - `stores`
 - `producthierarchy`
 - `sales2018-2019`
 - `Stores - construct...`
 - `Date Dimension`
 - `Measure Table`
 - `Security1`
 - `PLS`
 - Query1**

Advanced Editor

```
= {Number.Sign(-23), 5, Text.Upper("test")}
```

	List
1	-1
2	5
3	TEST

The screenshot shows the Power BI Advanced Editor with the following code:

```
let
    Source = {Number.Sign(-23), 5, Text.Upper("test")},
    #"Converted to Table" = Table.FromList(Source, Splitter.SplitByNothing(), null, null, ExtraValues.Error)
in
    #"Converted to Table"
```

Queries [13]

- Helper Table [4]**
 - `sales2017`
 - `sales2018`
 - `sales2019`
 - `sales2020`
- Other Queries [9]**
 - `stores`
 - `producthierarchy`
 - `sales2018-2019`
 - `Stores - construct...`
 - `Date Dimension`
 - `Measure Table`
 - `Security1`
 - `PLS`
 - Query1**

Advanced Editor

Query1

	Column1
1	-1
2	5
3	TEST

This transformation takes a list and converts it into a table with a single column.

Working with Tables

Tables in M Language are more complex than values and lists, as they consist of rows and columns. When we apply transformations in Power BI's Query Editor, most of those transformations are performed on tables, and these actions are translated into M code using table functions.

To better understand how to manipulate tables using M Language, let's look at an example where we want to transform the text in a specific column to lowercase.

Example: Using `Table.TransformColumns` to Transform Text

Let's begin by applying a transformation to a column using the `Table.TransformColumns` function. Here's how it works:

- 1. Start with a Table:** Imagine we already have a table where one of the columns contains text data, such as `"Column1"`. Now, we want to convert all text in this column to lowercase.

- 2. Use the `Table.TransformColumns` Function:**

The

`Table.TransformColumns` function is designed to transform one or more columns in a table.

- The first argument is the table you want to transform.
- The second argument is a list that specifies which column(s) to transform and what transformation to apply.

- 3. Apply the Transformation:**

Let's write the M code to convert the text in

`"Column1"` to lowercase using the `Text.Lower` function.

In this example:

```

let
    Source = {"ABC", "DEF", "GHI", Text.Upper("test")},
    #"Converted to Table" = Table.FromList(Source, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
    Transform = Table.TransformColumns(#"Converted to Table", {"Column1", Text.Lower})
in
    Transform

```

- `Source` represents the original table.
- `Table.TransformColumns` is used to reference `"Column1"` and apply the `Text.Lower` function to its values.

When you run this code, all text values in `"Column1"` will be converted to lowercase. Note that if `"Column1"` contains numbers or other non-text values, you might encounter errors, as `Text.Lower` only works on text data.

Understanding the Structure

- **Referencing the Table:** The first argument, `Source`, refers to the table that contains the data we are working with. This can be a previous step in your query.
- **Specifying the Transformation:** The second argument is a **list**. Each element in the list is a pair containing the column name and the transformation function. In this case, the transformation function is `Text.Lower`.

▼ Table.AddColumn & Each in M Language

In Power BI's M Language, the `each` keyword and the `Table.AddColumn` function are key components for adding and transforming columns in tables. Understanding how they work allows you to perform more advanced data manipulations, customize queries, and make adjustments to your datasets.

▼ Understanding `Table.AddColumn`

The `Table.AddColumn` function is one of the most commonly used functions in Power BI's M Language. It allows you to create new columns

in your table based on existing data. The general syntax for `Table.AddColumn` is:

```
Table.AddColumn(table as table, newColumnName as text, columnGenerator as function) as table
```

- `table`: This is the table from a previous step that you are working with.
- `newColumnName`: This is the name of the new column you want to create.
- `columnGenerator`: This is a function or expression that defines the values for the new column.

Example: Adding a New Column

Let's add a new column that transforms text from `"Column1"` to lowercase but keeps the original column intact.

```
let
    Source = {"ABC", "DEF", "GHI", Text.Upper("test")},
    #"Converted to Table" = Table.FromList(Source, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
    #"Transform" = Table.TransformColumns(#"Converted to Table", {"Column1", Text.Lower}),
    #"AddNumberColumn" = Table.AddColumn(#"Transform", "Test Number Column", each 123, type number)

in
    #"AddNumberColumn"
```

In this example:

- The `Table.TransformColumns` function converts the values in the `"Column1"` to lowercase.
- The `Table.AddColumn` function adds a new column named `"Test Number Column"`, and assigns the value `123` to each row in this column.

▼ Understanding `each`

The `each` keyword is a shorthand in M Language that refers to a row in a table. When you use `each`, Power BI understands that you are applying a function to each row of the table. For example, in the previous example, `each Date.DayOfWeekName([OrderDate])` applies the `Date.DayOfWeekName` function to each value in the `OrderDate` column.

Whenever you refer to a column within `Table.AddColumn` or other similar functions, `each` is needed to tell Power BI to evaluate the function for each row in the column.

Another Example with `each`

Let's add a new column that contains the length of the text in each row of `"Column1"`:

	Column1	Length of Text
1	abc	3
2	def	3
3	ghi	3
4	test	4

Advanced Editor

Query1

```

let
    Source = {"ABC", "DEF", "GHI", Text.Upper("test")},
    #"Converted to Table" = Table.FromList(Source, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
    #"Transform" = Table.TransformColumns(#"Converted to Table", {"Column1", Text.Lower}),
    #"AddLengthColumn" = Table.AddColumn(#"Transform", "Length of Text", each Text.Length([Column1]), type number)
in
    #"AddLengthColumn"
  
```

In this example:

- The `Table.AddColumn` function adds a new column called `"Length of Text"`.
- The `each Text.Length([Column1])` expression calculates the length of each value in `"Column1"` and applies it to each row. `each` is used to evaluate this expression for every row in the table.
- The `type number` ensures the new column contains numeric values (the length of each string).

This example shows how `each` can be used to dynamically calculate values for each row, based on the data in the table.

Common Mistakes with `each`

One common mistake is to reference a column without using `each`, which results in an error. Without `each`, M Language expects a single value rather than a column of values. Always use `each` when you are applying a transformation to a column in a table.

Conclusion

The `each` keyword and `Table.AddColumn` function are crucial for transforming and adding columns in Power BI's M Language. By mastering these tools, you can:

- Apply complex transformations to your data.
- Create new columns based on calculations or expressions.
- Ensure the data types of new columns are correctly defined.

Understanding how `each` works in conjunction with column references is key to avoiding errors and optimizing your data transformation processes. In the next section, we will explore more advanced transformations and learn how to work with **conditional columns** and **grouping data** using M Language.

▼ More Functions & Next Steps in M Language

Now that you've gained a foundational understanding of M Language, you're in a great position to start exploring more advanced functions and concepts. While it's impossible (and unnecessary) to memorize every

function in M Language, the skills you've developed will help you navigate and apply new functions as needed. A key resource that will aid you in this process is the **Power Query M formula language documentation**, which provides a comprehensive reference for all M Language functions.

Leveraging Documentation to Expand Your Knowledge

The **Power Query M Language documentation** is an invaluable resource for discovering new functions and understanding how to use them. You can easily access it online by searching for "Power Query M function reference." Here, you'll find categorized lists of functions, such as date functions, text functions, and number functions.

For instance, if you're looking for a function related to dates, you can navigate to the **Date functions** section in the documentation and explore functions like `Date.DaysInMonth`, which returns the number of days in a given month.

Example: Using `Date.DaysInMonth` to Add a Column

Let's say you want to add a column to your data that shows the number of days in the month for each date in the **OrderDate** column. Here's how you can do it by leveraging the documentation and applying the function:

1. Find the Function in Documentation:

- Search for the function `Date.DaysInMonth` in the documentation to understand its syntax and usage.
- The documentation will show that this function takes a date as input and returns the number of days in that date's month.

2. Apply the Function in M Language:

Now let's use

`Table.AddColumn` to create a new column in your table that calculates the days in the month for each entry in the **OrderDate** column.

```
let  
    Source = YourPreviousTableStep,  
    AddedColumnTable = Table.AddColumn(
```

```
Source,
"DaysInMonth",
each Date.DaysInMonth([OrderDate]),
type number
)
in
AddedColumnTable
```

In this example:

- `Source` refers to the previous step in your query.
- `"DaysInMonth"` is the name of the new column being added.
- `each Date.DaysInMonth([OrderDate])` applies the `Date.DaysInMonth` function to each value in the **OrderDate** column.
- The new column is set to the **number** data type using `type number`.

3. Result:

After running this query, your table will now include a column named **DaysInMonth**, which shows the number of days for each corresponding date in **OrderDate**.

Using Power BI's Interface to Learn M Language Syntax

Another powerful trick to deepen your understanding of M Language is to observe how Power BI automatically generates M code when you perform actions through the graphical interface. This allows you to learn by example and customize the generated code for your own needs.

For instance, if you want to extract the **year** from a date column:

1. Add a New Column:

- Use Power BI's graphical interface to create a new column that extracts the year from the **OrderDate** column. This is done via the **Add Column** tab and selecting the appropriate date transformation.

2. Check the Generated Code:

- Open the **Advanced Editor** to see the M code that Power BI generated. You might see something like this:

```
mCopy code
let
    Source = YourPreviousTableStep,
    AddedColumnTable = Table.AddColumn(Source, "Year", each Date.Year([OrderDate]), type number)
in
    AddedColumnTable
```

3. Modify the Code:

- You can now modify this code to extract other date parts, such as the month, by changing the function from `Date.Year` to `Date.Month`:

```
mCopy code
let
    Source = YourPreviousTableStep,
    AddedColumnTable = Table.AddColumn(Source, "Month", each Date.Month([OrderDate]), type number)
in
    AddedColumnTable
```

Learning by Experimentation and Customization

As you become more comfortable with M Language, you can experiment by applying various transformations through the interface, observing the generated code, and customizing it to suit your needs. This is a great way to practice and understand how different functions work in real-world scenarios.

Final Thoughts and Next Steps

At this stage, you should feel confident about:

- Navigating the M Language documentation to find relevant functions.
- Understanding the structure of M Language queries, including functions like `Table.AddColumn`, `each`, and the syntax for referencing columns.
- Using Power BI's interface to generate M code, which you can modify and adapt.

Tips for Continuing Your M Language Journey:

1. **Explore New Functions:** As you work with more complex datasets, continue to explore functions in the Power Query M function reference, such as:
 - **Text Functions:** For string manipulation (e.g., `Text.Upper`, `Text.Replace`).
 - **List Functions:** For working with lists and sequences (e.g., `List.Transform`, `List.Sum`).
 - **Date Functions:** For advanced date manipulations (e.g., `Date.AddMonths`, `Date.FromText`).
2. **Experiment with Queries:** Keep practicing by creating new queries and making adjustments in the **Advanced Editor**. Observe how different functions work, and learn from the errors that occur to strengthen your understanding.
3. **Join the Community:** The Power BI community is a great place to ask questions and share knowledge. Forums, tutorials, and examples from other users can help you solve specific challenges and discover new techniques.
4. **Stay Updated:** As Power BI evolves, new functions and features may be added to M Language. Make it a habit to check the official documentation and stay current with the latest updates.

By following these steps and applying what you've learned, you'll become more proficient in M Language and be able to unlock the full potential of Power BI's data transformation capabilities. Thank you for your dedication, and I wish you continued success in your data analytics journey!

▼ Last Project

[Last Project Instructions.pdf](#)

[Last Project.rar](#)