Data Modeling in Power BI

Learn Fundamentals of Effective Data Modeling and DAX Language

Said Fawzy

Feb. 2024

Table of Content:

Contents

Chapter 1: Introduction to Data Models 6
What is a Data Model? 6
Model view in Power BI 7
Introduction to schemas 17
Example of Company Schemas 20
Exercise 1 Configuring a Flat schema 23
Exercise 1 Questions 27
Table properties 27
Column properties 29
Exercise 2: Configure a Flat schema with multiple sources 32
Knowledge Check 37
Chapter 2: Tables Relationships 38
Fact and Dimension Tables 38
Normalization and denormalization 39
Difference between normalization and denormalization 42
Normalizing and Denormalizing Data in Power Query 42
Cardinality and Granularity 43
Managing model relationships 45
Model relationships 48
Cross-filter direction 51
Knowledge Check 53
Chapter 3: Star and Snowflake Schema 54
Setting up a Star schema in Power BI 54
Exercise 3: Configuring a Star schema 54
Exercise 3 Questions 58
Setting up a Snowflake schema 58
Exercise 4: Setting up Snowflake schema 59
Exercise 5: Changing Star schema into a Snowflake schema 60
Knowledge Check 65

Chapter 4: Introduction to DAX 66
What is DAX? 66
Dax Fundamentals 66
Row context and filter context 69
Creating calculated columns 71
Exercise 6: Creating Calculated Columns 71
Exercise 7 : Filtering Context 73
Introduction to calculated tables 75
Real-World applications of calculated tables 76
Exercise 8: Adding a calculated table and column 77
Knowledge Check 82
Chapter 5: Measures 83
What are Measures? 83
Benefits of Measures 84
Measure Syntax 84
Exercise 9: Creating Measures 84
Exercise 10: IF Function 89
Exercise 11 DISTINCTCOUNT Function 91
Exercise 12 Creating Aggregation Function 93
Exercise 13: Using CALCULATE FUNCTION 94
Exercise 14: Variables in DAX 98
Exercise 15: OR Condition 99
Exercise 16: Organize your Measures in one table 100
Knowledge Check 101
Chapter 6: More About DAX 102
About DAX 102
Row Context 102
Exercise 17: Row Context 102
Iterators 104
Exercise 18 Iterators 104
Filter Context 107
Exercise 19: Filter Context 107

Chapter 7 : Time Intelligence Functions....112
What is Time Intelligence Functions?....112
Using Time Intelligence in Power BI....113
Setting up a common date table....114
Exercise 20: Set up a common date table Using DAX....115
Exercise 21: Set up a common date table Using M Language....122
Exercise 22: Calculating MTD , QTD , and YTD....123
Exercise 23: Comparing same period last year....127
Exercise 24: Compare to previous period....129
Exercise 25: DATEADD Function....129
Knowledge Check....130
Chapter 8: Modeling Data Project....131
Step 1: Create Tables Relationships....131
Step 2: Configure Tables....135
Step 3: Create Quick Measures....139
Step 4: Create a Date Dimension Table....141
Step 5: Create DAX Calculation....147

Introduction:

This is my second Training material in the series of learning Power BI. The first one was "**Data Preparation for Analysis in Power BI**". It was a training material about fundamentals of ETL (Extract, Transfer, Load) processes in Power BI. This one is to continue the journey of learning Power BI for beginners.

In this stop of the journey, you will learn about data modeling and types of schemas with advantages and disadvantages of each.

I will start walk you through DAX language and how to create tables, columns and measures using this powerful language.

I tried to make the book easy to use with many examples. And this material is a companion to my videos of training available in play lists of my YouTube channel.

I Advise you to follow my Videos on YouTube and use this material with the companied Power Point presentation and files of exercise and solution and try everything yourself as you won't learn unless you get your hands dirty.

You can visit my channel to watch the videos and download all materials from the link in the description of each video.

www.youtube.com/saidfawzy

The evolving success of Power BI made it crucially important for all people in the field of Data Analysis to start using and make use of its power and features that are added to the software day after day. I hope I complete this series with the visualization in Power BI Book soon.

Feel free to contact me through my Linked in: www.linkedin.com/in/saidfawzy.

This Book is free and feel free to share with anyone with the accompanied material. And never hesitate to contact me if you need any help.

Said Fawzy
Manager of Information Center
Arab Contractors
26 Feb. 2024

Chapter 1: Introduction to Data Models

- As a data analyst, you'll often manage thousands, hundreds of thousands, or even millions of records. But how can you generate insights from all this raw data?
- You can convert it into data models.
- Suppose your company needs to generate insights and increase sales from different data sources. These data sources include **customer sales** and **marketing data**. But these data sources are all in separate locations, and the only way to generate insights is to combine them.
- That's where the **data model** comes in. Your company can integrate its data sources as a data model in Microsoft Power BI, then generate insights in the form of visualizations.

What is a Data Model?

- At its core, data modeling is **creating a structured representation of data**. Representation can then be used to support different business aims.
- In other words, a data model shows **how different data elements interact**, and it also outlines the **rules** that **influence** these interactions.

How can Power BI Helps

Microsoft Power BI is software that provides data analysts with a user friendly interface for building data models.

- Other benefits of a Power BI data model are that it can be used to define **relationships** between tables and assign **data types**.
- You can also create **calculated columns** and **measures** and update your model as your business requirements change.
- In Power BI, the foundation of creating reports and dashboards lies within the data model. It's important to understand how to design a data model that effectively aligns with the visual elements within your reports and dashboards.
- There are several steps involved in building a data model in Power BI:
    - **Connect** to your data sources,
    - prepare and **transform** your data, and
    - **configure** table and **column properties**, then
    - create model **relationships**. And finally,
    - create **measures** and **calculated columns** using **DAX** or data analysis expressions.
- Once your data model is in place, you can analyze the data to generate insights to help you achieve your business objectives.

Optimizing Data Model

- By optimizing the data model, you can significantly improve the **performance** of your Power BI reports and dashboards.

- It's also easier to aggregate structured data in a data model, thanks to the clear relationships and hierarchies.
- With an effective data model, you can perform more advanced analytical capabilities like **complex measures** and **predictive analysis**. When your underlying data is structured, organized, and aligned, your insights and reports are more likely to be accurate and reliable.

How to build Data Model with Power BI

- **First, you need to connect to data sources**:
  - by executing a query in Power Query Editor.
  - The result is then loaded into the Power BI data model as a table.
  - Using Power Query in Power BI, you can finish importing and cleaning their data sources. This creates a data model that contains cleaned for example customer, date, employee, and marketing data as separate tables.
  - **Each table in the model** represents a specific **business entity**, and each table also has its own related attributes.
- **The next step is to define the relationships between the tables** in Power BI's model view:
  - The company can link for example its customers and sales tables using the Customer ID column, which is common to both tables.
- **Finally, the company needs to create measures and calculated columns using DAX**:
  - DAX is a syntax used in Power BI to analyze data.
  - You can use DAX to create **aggregations** and **custom calculations** to generate insights on important aspects of their data, like sales totals.

- A strong understanding of data models will help you:
  - maximize your data's full potential.
  - Building sophisticated data models
  - creates a robust foundation for data analysis and generating insights.

Question

Your Company has tasked you with creating a data model to analyze sales data and improve the store's performance. What should your first step in the data modeling process be?

A. Configure column and table properties.
B. Prepare and transform your data.
C. Connect to your data sources.

Model view in Power BI

- The **Model view** visually represents all data tables, relationships, and columns. You can use these visuals to shape and structure your data. The **Model view** is especially crucial when a data model contains complex relationships between its tables.

Model view elements

- The **Model view** can be accessed by selecting the **model** icon on the left sidebar of Power BI desktop. The **Model view** contains the following UI elements:

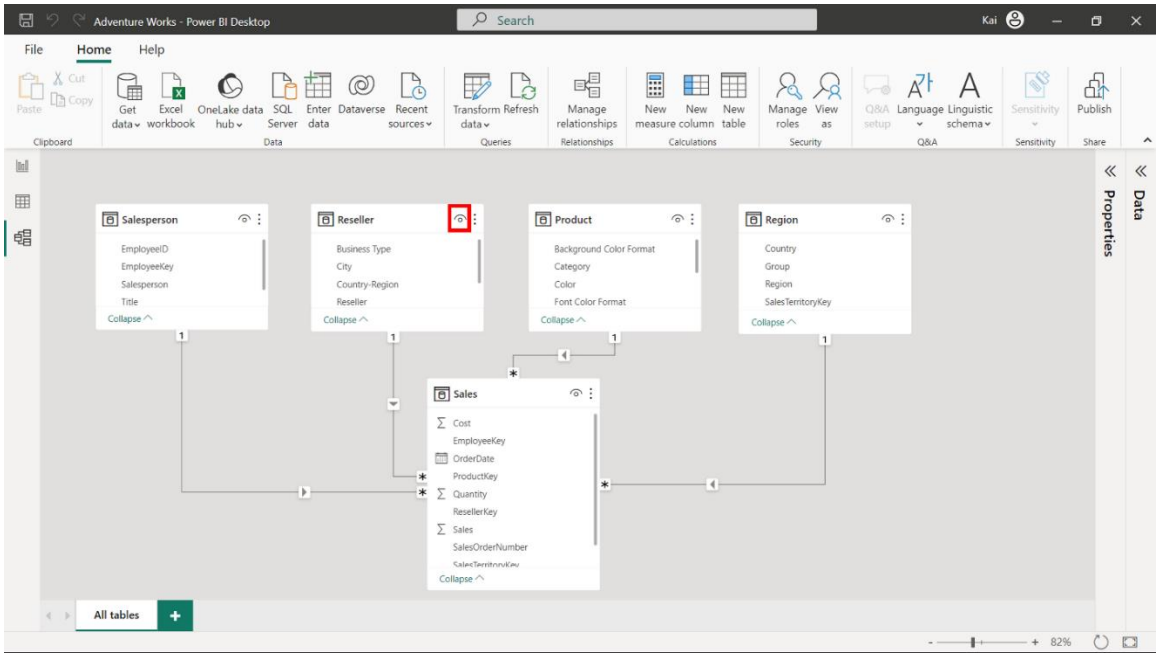
    - **Diagram view (canvas)**
    - **Data pane**
    - **Properties pane**
    - **Home ribbon**

Diagram view (Canvas)

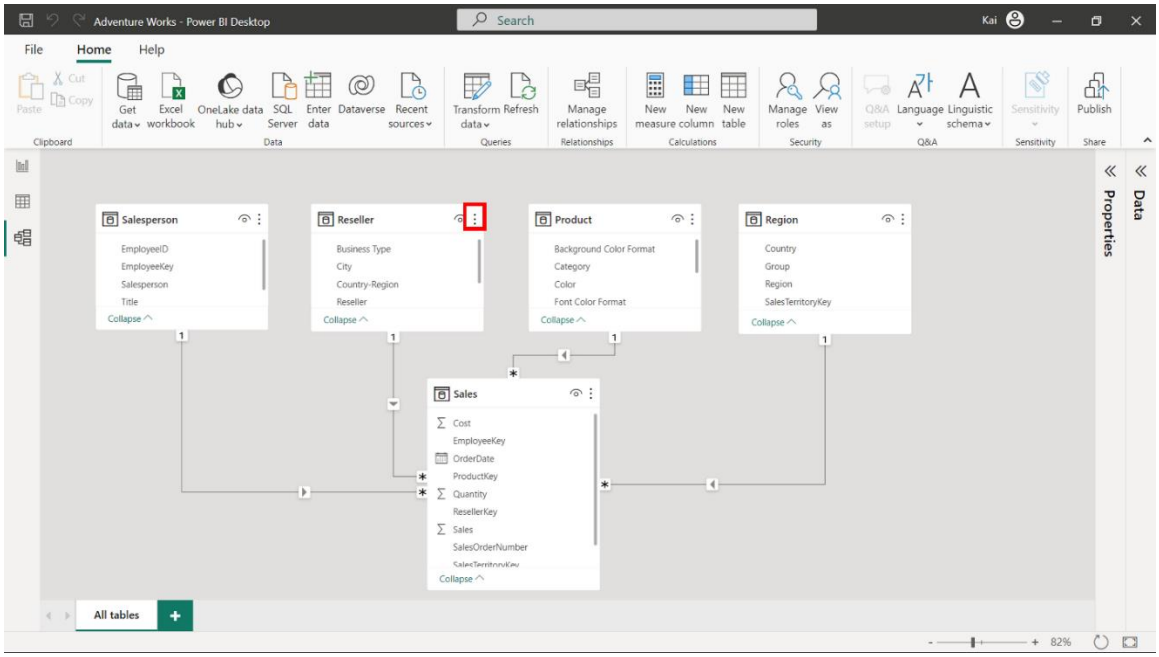
- The **Diagram view** presents a visual overview of your data model. You can use it to visualize important elements of your data model, like the data tables, fields, and relationships. These elements are explored in more detail below.

Data tables

- Data tables represent the raw data you import or connect to Power BI. The tables contain fields (columns) and rows of data.
- In the **Model view**, you can view a list of all your data tables (both imported and calculated).
- You can expand (collapse) the data table to view (hide) the fields within it.
- The **small eye icon** on the top right of the table diagram allows you to hide the table from the Power BI desktop report view.

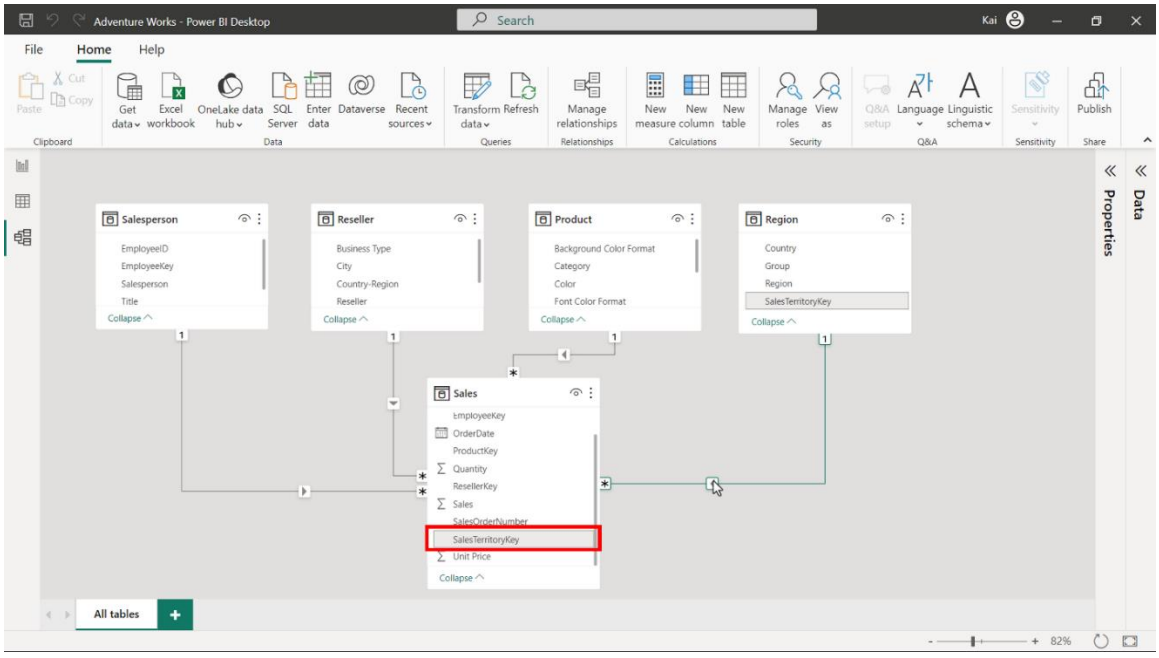


- You can access more options by selecting the **ellipses** beside the eye icon of the table diagram.



Fields

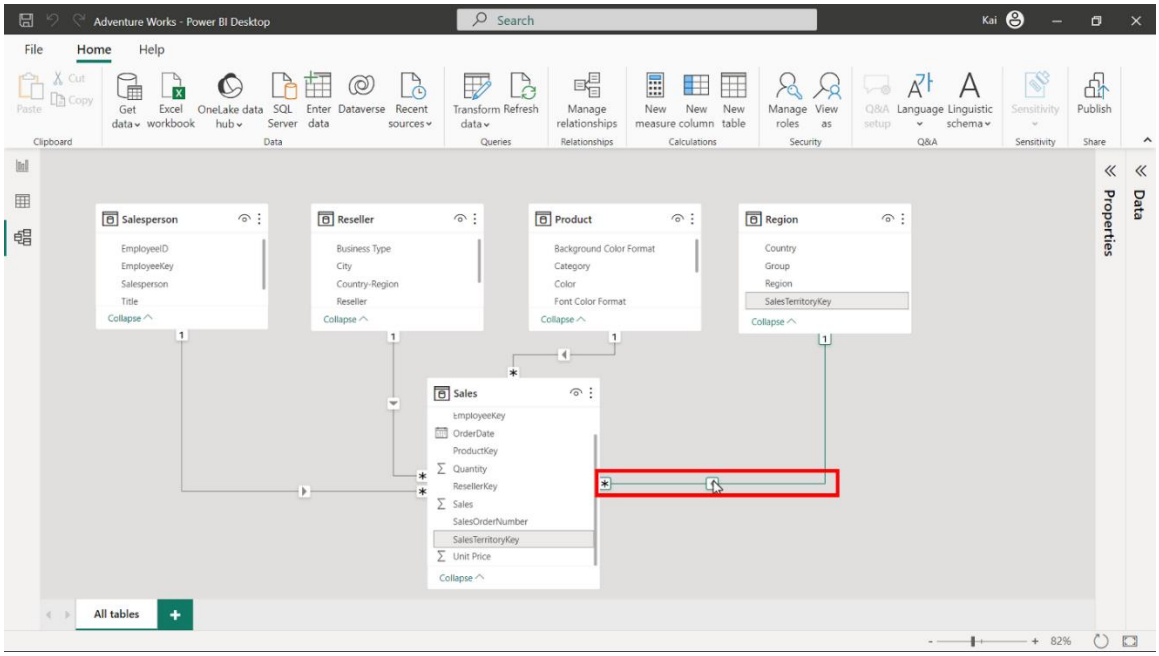
- Fields are columns within your data tables.
- In the **Model view**, these fields are listed under their respective tables. All calculations (calculated columns and measures) also appear as fields in the data tables.



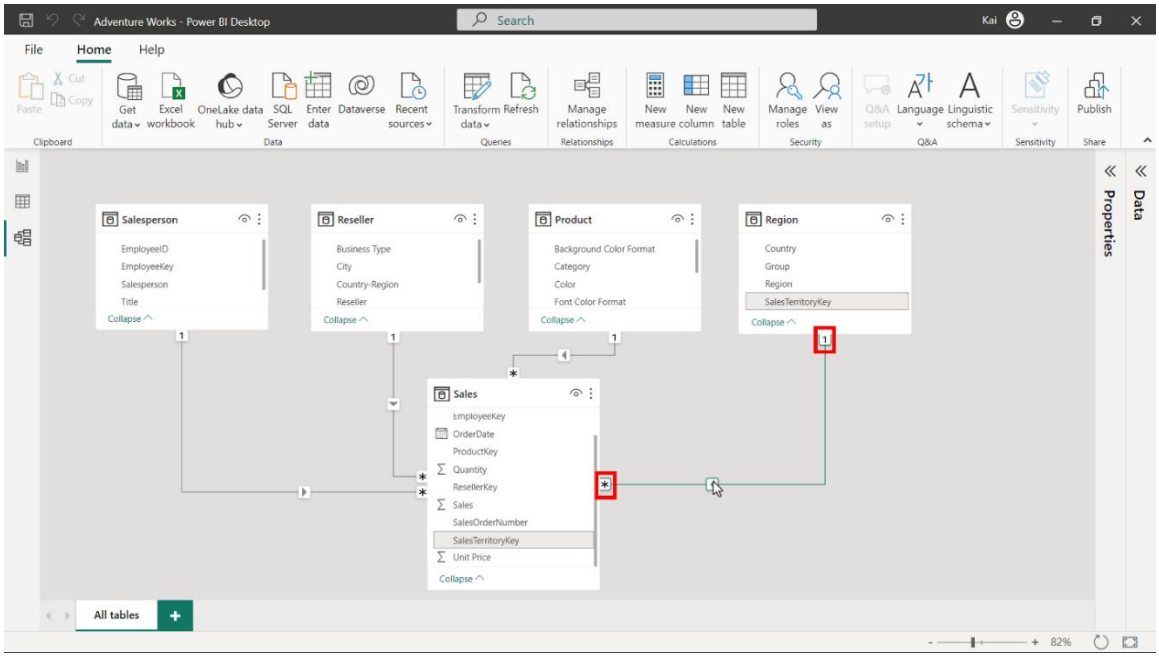
Relationships

- One of the key features of the **Model view** is establishing and managing relationships between tables.

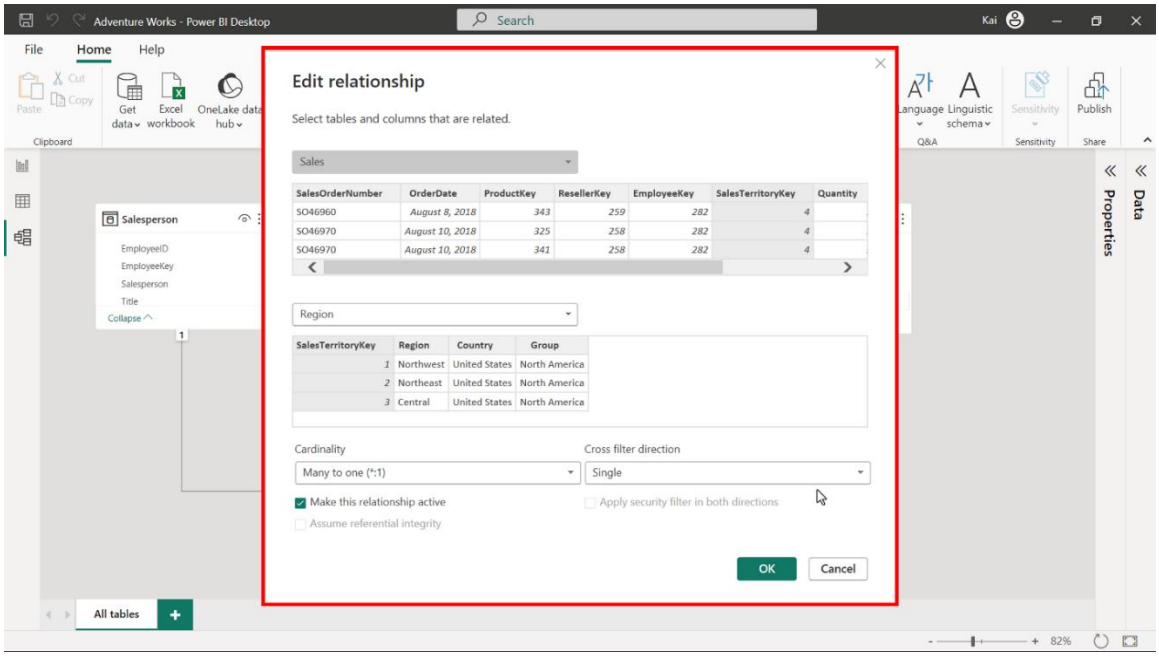
- You can create and configure relationships between tables based on common key fields.
- Tables are related through a relationship line (connector line).
- When you hover over the connector line, it highlights the fields used to establish the relationship.
- The **arrow** on the connector line indicates the direction of the relationship (cross-filter direction).



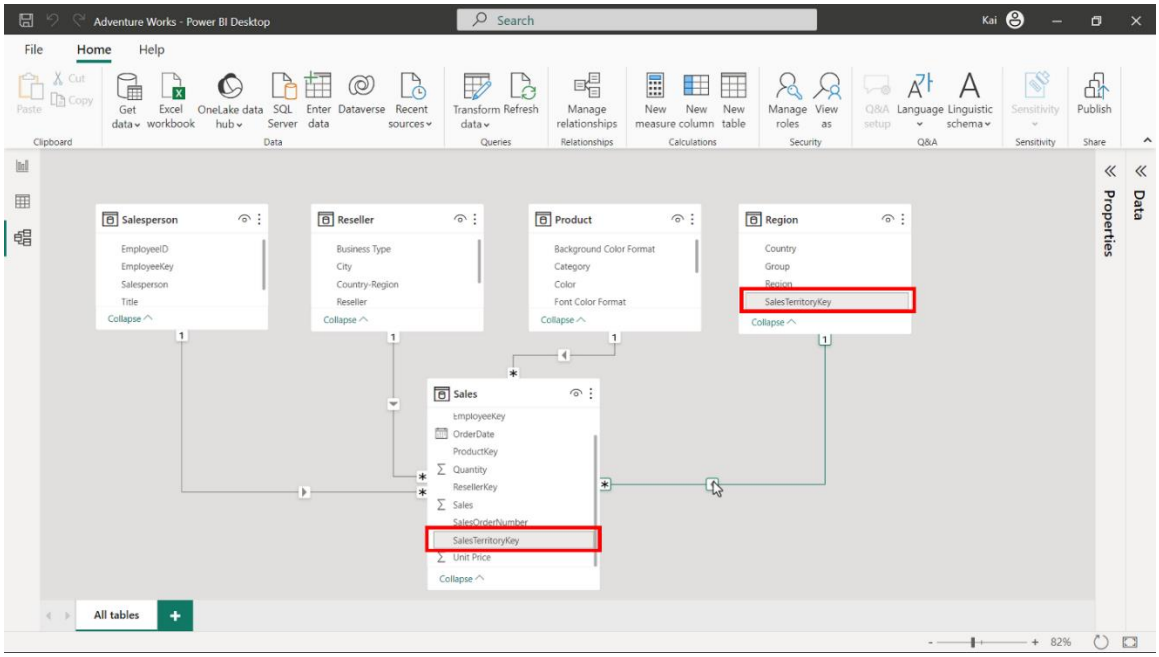
- The **connector line** connects two tables.
- The **1** icon represents the **one** side of the relationship, while the ***** icon indicates the **many** side of the relationship.



- You can select and double-click the **connector line** to open the **Edit relationship** window.
- You can use this window to edit or configure the relationships between your data tables.

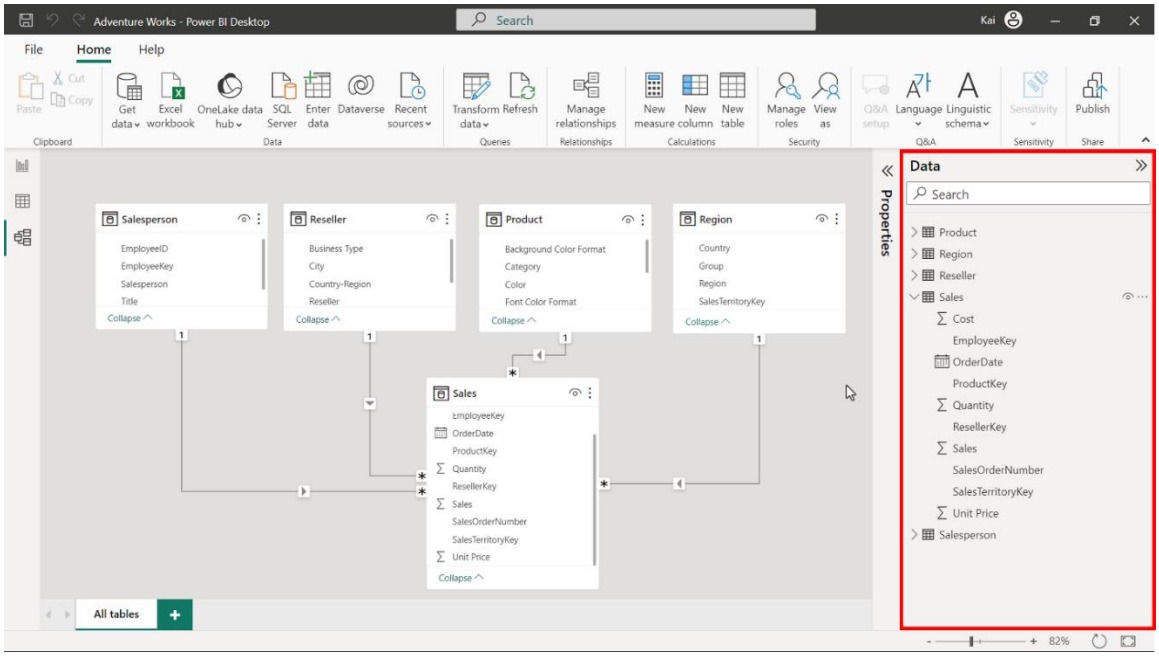


- When you load multiple tables into your data model, Power BI automatically detects their relationship based on the common key field.

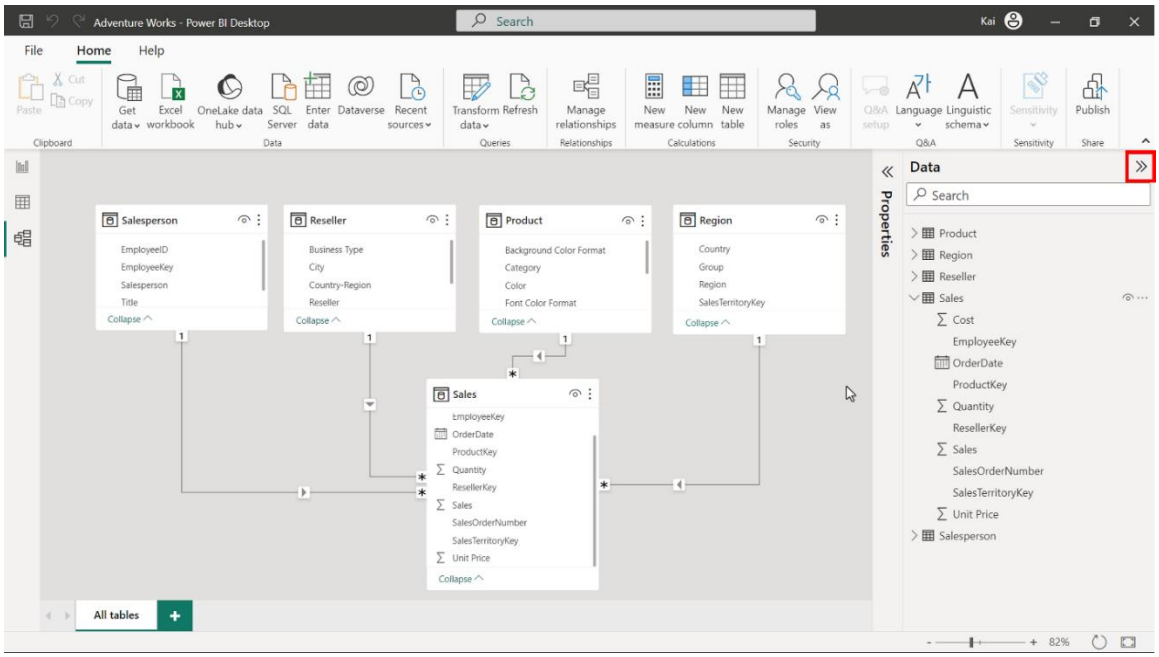


Data Pane

- The **Data pane** is the common element in all three views of Power BI desktop (**Report view**, **Data view**, and **Model view**).
- The **Data pane** lists all data tables and fields within the tables. All types of calculated tables, columns, and measures you will create are listed under the **Data pane**. If your data model contains many tables, you can use the search bar to locate a specific data table.

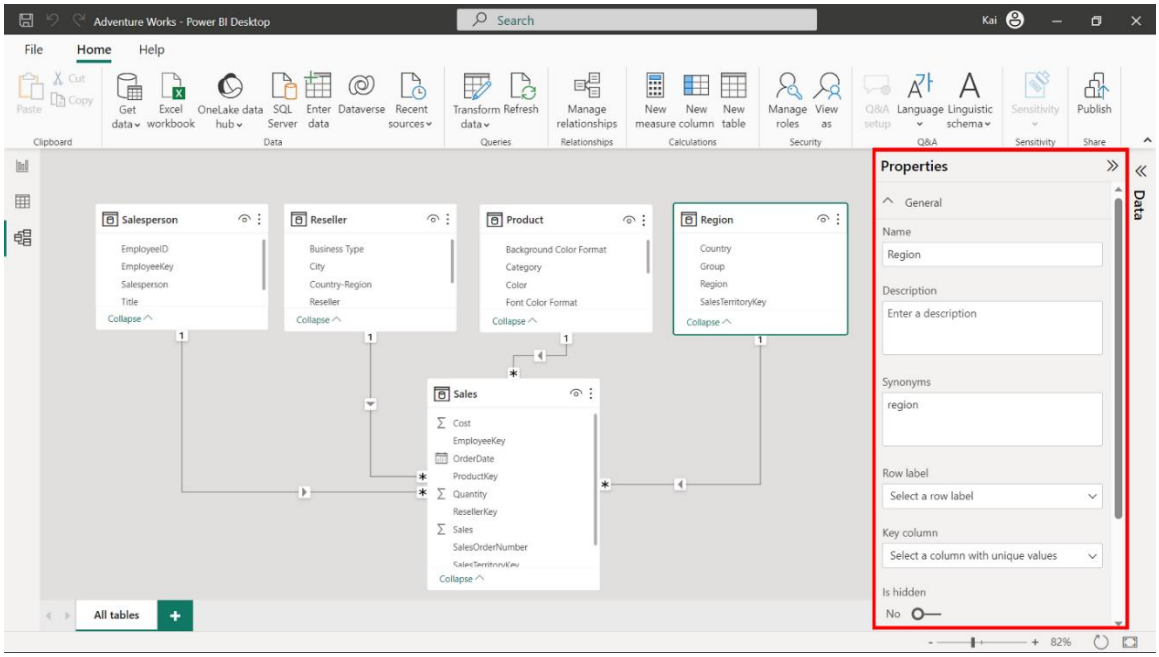


- You can also expand or collapse the data pane by selecting the double arrow.



Properties pane

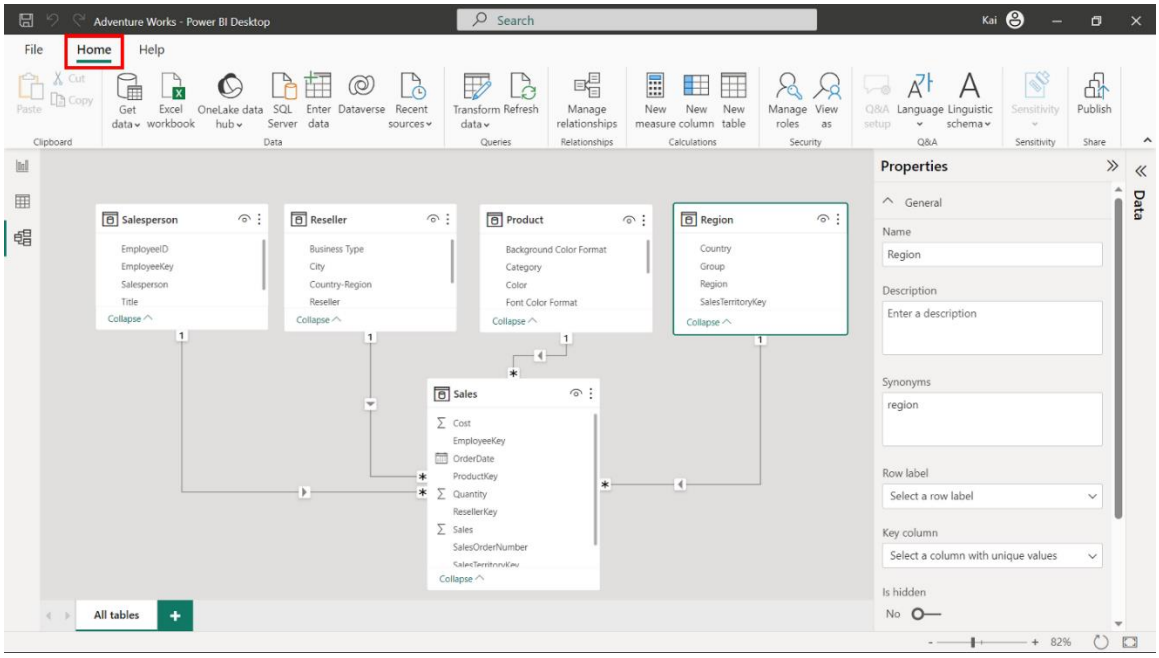
- You can use the **Properties** pane to edit and configure table and column properties within your data model.
- The pane has two sets of properties: **General** and **Advanced**.
- You can use these properties to rename your table and columns, add descriptions, and configure other table and column properties.



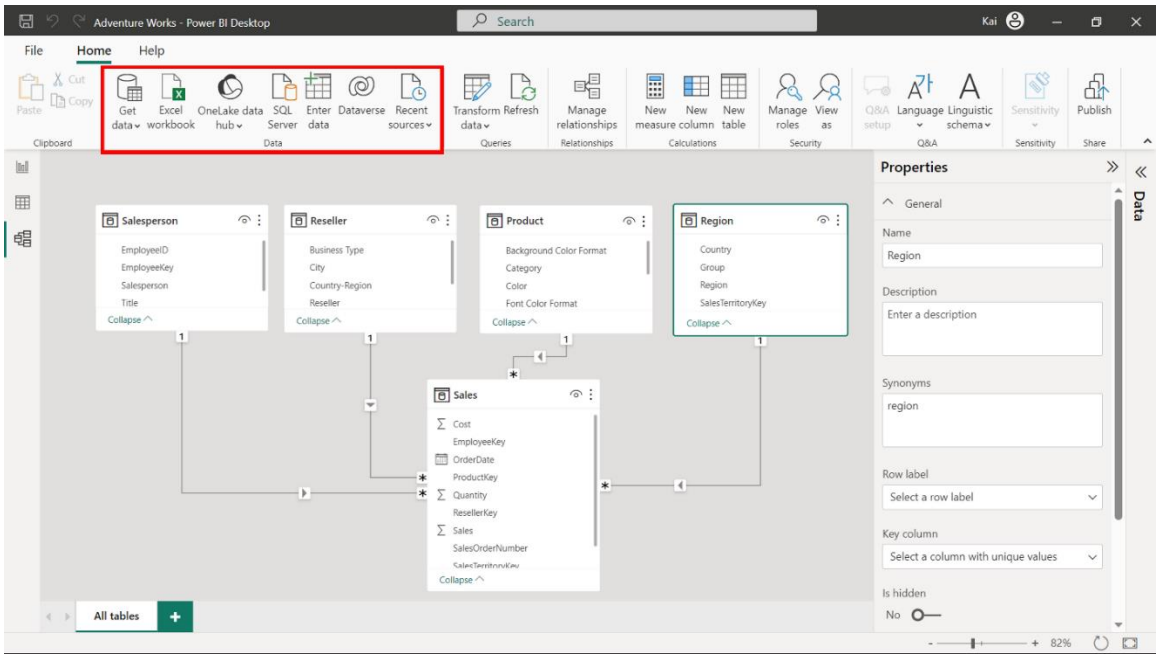
Home ribbon

- The **Home ribbon** of the **Model view** offers a range of functions to shape and structure your data before preparing it for reports and visualizations.

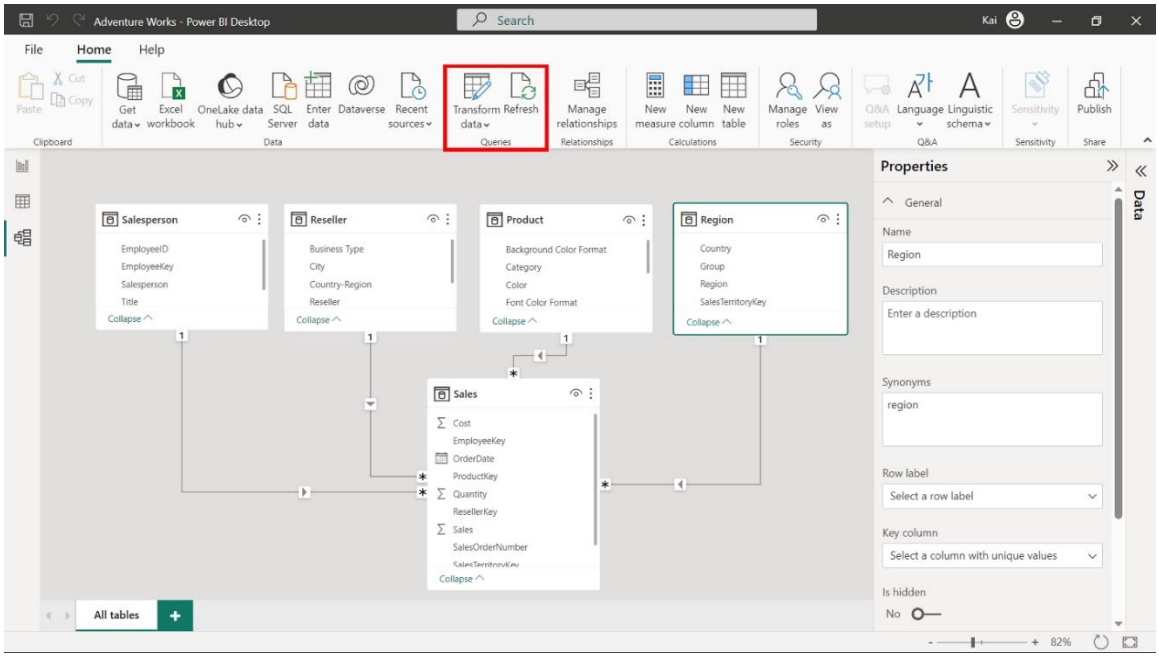
- Below is a list of the various groups of functions available in the **Home ribbon**.



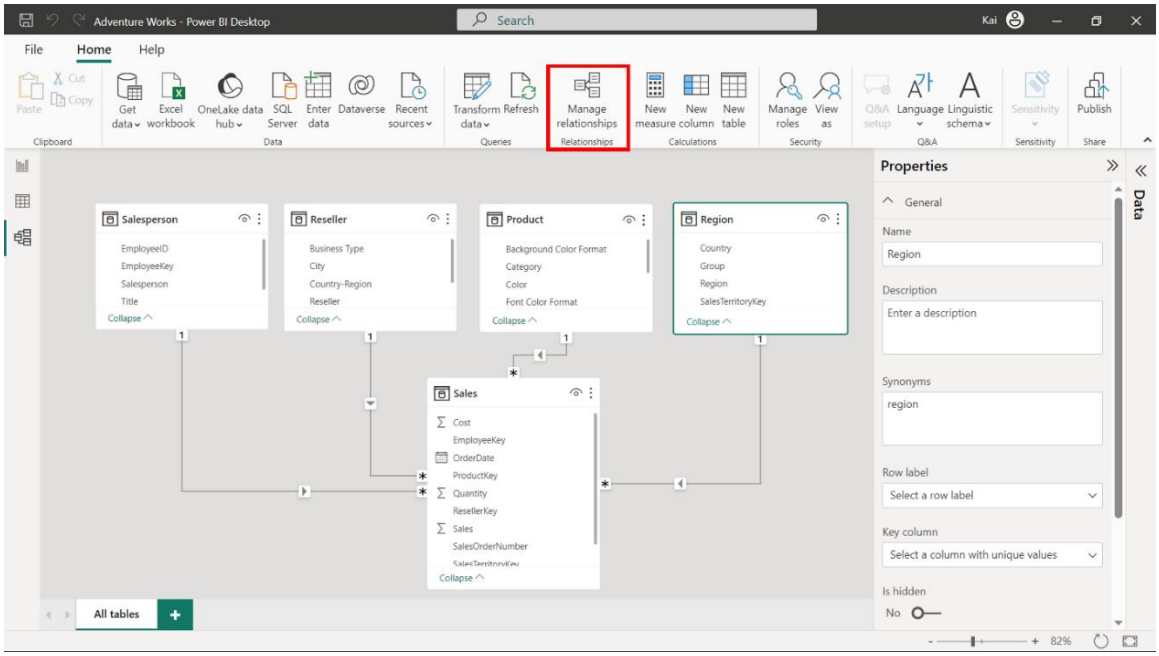
- **Feature:** Data
- **Explanation:** The **Home** tab's **Data group** is like the **Report view** and **Data view**. It offers data connectivity options for Power BI.



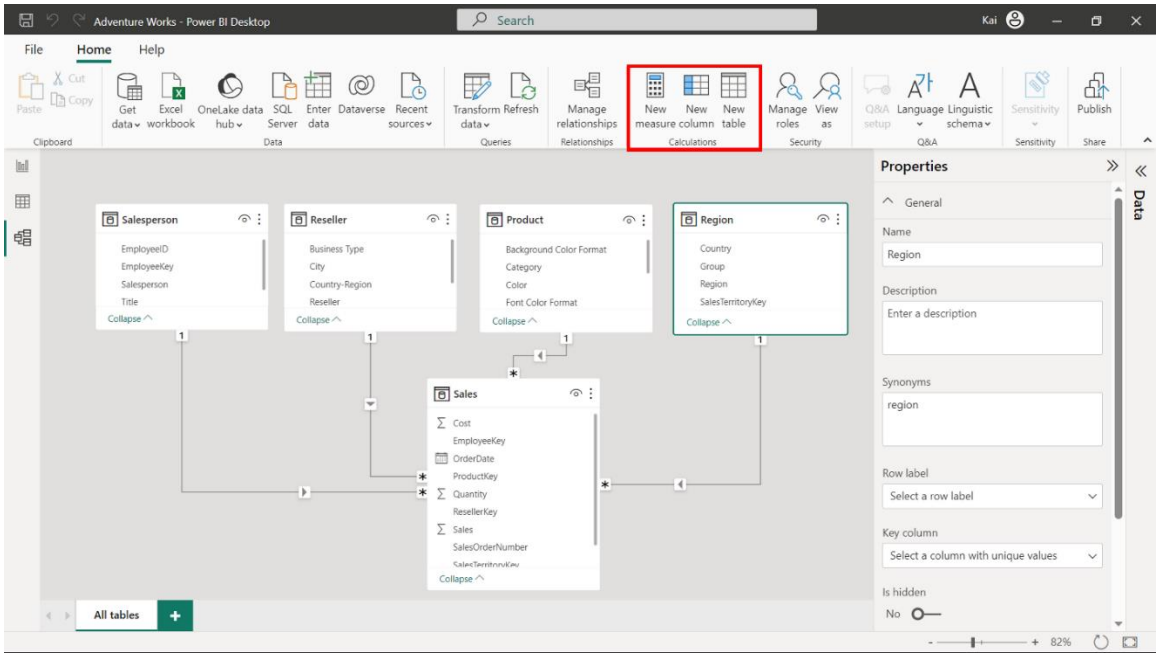
- **Feature:** Queries
- **Explanation:** This section contains two elements:
- **Transform data** redirects you to the Power Query editor, where you can perform necessary transformations on your data.
- **Refresh** is used for data refresh. This is useful for when you change a dataset.



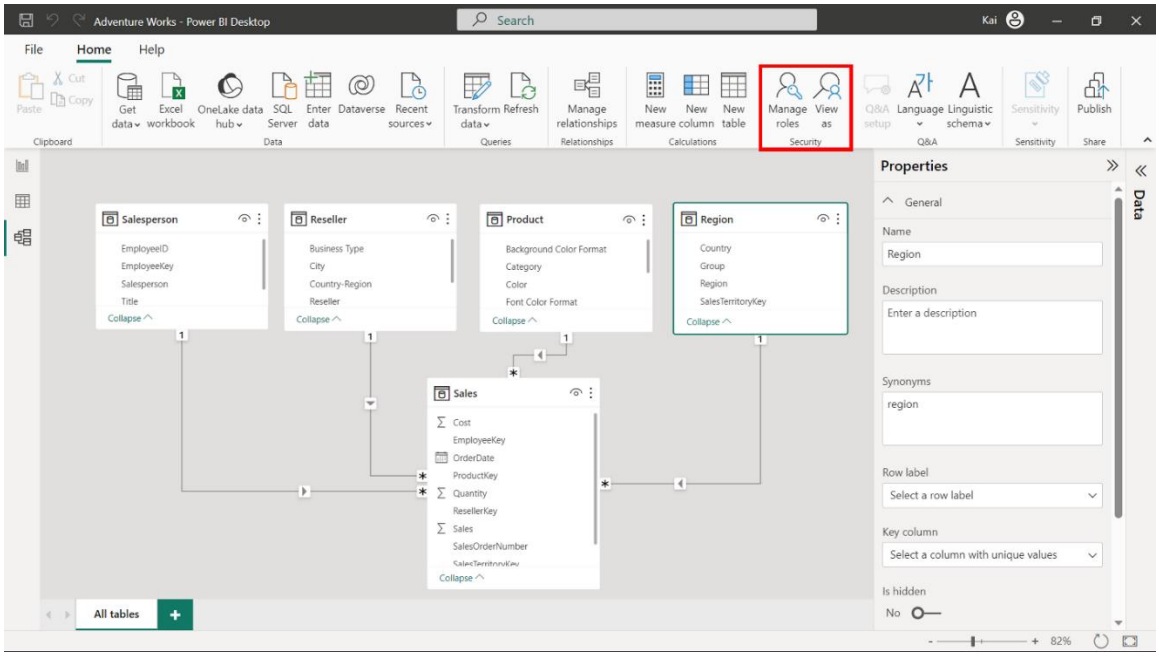
- **Feature:** Manage Relationships
- **Explanation:** This is the only element in the relationships group that you can use to establish new relationships between data tables and edit existing relationships.



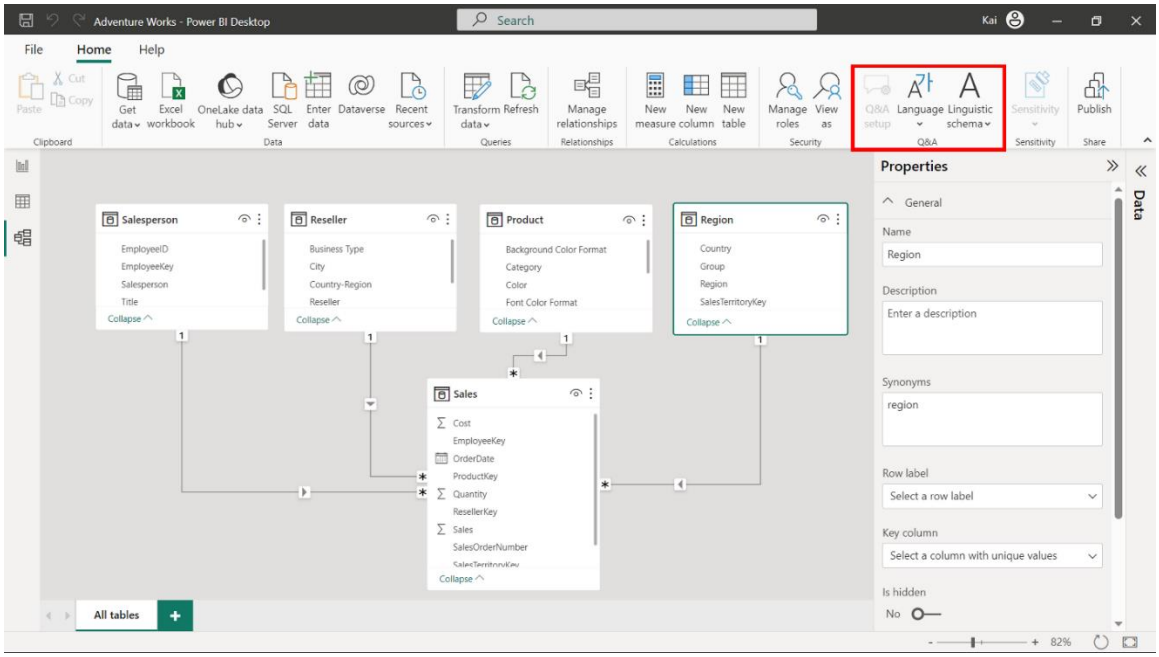
- **Feature:** Calculations
- **Explanation:** You can use this group to add new custom calculations to your data model, like creating new tables, columns, and measures.
- These calculations are created using a Power BI language called Data Analysis Expressions (**DAX**).
- These calculations enable you to enhance your data analytic capabilities and uncover the information hidden in your data.



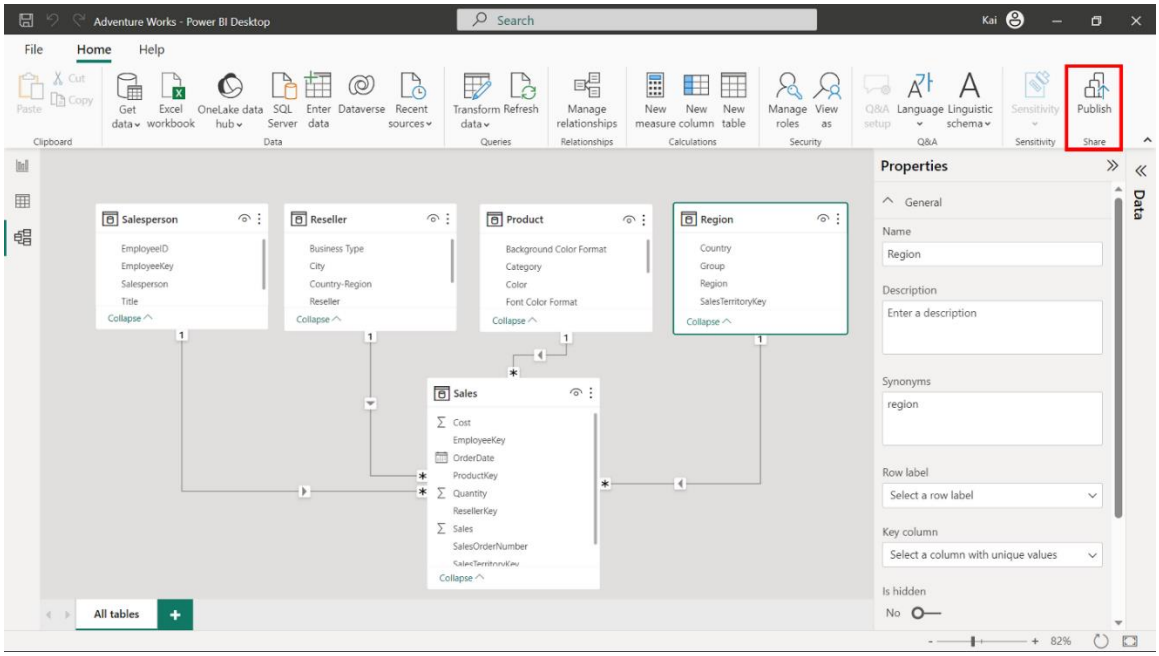
- **Feature:** Security
- **Explanation:** You can use this feature to define roles and rules within Power BI desktop to restrict data access for certain users.
- This provides you with better security for your data.



- **Feature:** Q&A
- **Explanation:** Q&A (Question and Answer) is a powerful natural language query and visualization tool that you can use to interact with data more intuitively.



- **Feature:** Share
- **Explanation:** You can use this element to share your report with other users within your team for effective collaboration.



Introduction to schemas

- Generating business insights often means working through large amounts of data. And it's important that this data is stored and structured meaningfully.

- As a data analyst, you'll frequently use schemas when designing your data models. So, it’s important that you understand the different types of schemas available, along with the advantages and disadvantages of each.
- With this knowledge, you can choose the schema that best suits the needs of your database and business objectives.
- With Power BI, you can structure your data using a schema.
- You can use a schema in Power BI to organize and build relationships between these different data sources.

What is a Schema?

- A schema refers to **a structure that defines the organization and relationships of tables within a dataset**.
- It represents **the logical framework** of how the data is organized and connected.
- A schema plays a crucial role in defining the data structure. It also enables efficient data analysis, helps with the creation of visualizations, and assists with generating meaningful insights from your data.
- In the context of Power BI, a schema is a logical blueprint that defines the structure, organization, and relationships of tables.

Types of Schemas

- There are three different types of schemas that can be used to organize and structure data:
    - a flat schema,
    - a star schema, and
    - a snowflake schema.

Flat Schema

- A flat schema is the simplest form of a data model.
- All attributes and fields related to the entity are stored in **a single table**.
- As you discovered in earlier courses, a table is a set of rows containing data, with each row divided into columns. Each column represents a piece of information with a specified data type. The required attributes and entities are stored in the rows and can be extracted as required from the columns.

Advantages of a flat schema.

- It's easy to retrieve data from.
- It's less complex to analyze flat schema data, and it's a simpler way to visualize data.

Disadvantages of Flat Schema

- It requires large datasets, which are **difficult** to maintain and **slow to query**.
- It leads to **data redundancy** and **inconsistency**, so is more suited to smaller datasets.
- And it doesn't allow for complex datasets, which require more flexibility and detail.

Star schema data model

- A star schema is a more advanced approach to structuring and organizing quantitative, or measurable data in Power BI.

- It allows for multiple tables to be connected through one central table.
- In a star schema, a central fact table connects to multiple dimension tables.
- These connections look like a star shape, so it's called a star schema.
- For example your company can build a star schema using a central fact table that contains sales transactions. The company can then link the fact table to dimension tables that contain records for customers, employees, dates, and marketing campaigns.

Components of the star schema

- First, there's the **fact** and **dimension** tables.
- And there are the table **relationships**.
- There are many different types of relationships.

Star schema advantages

- By storing data in separate tables, star schemas help to **reduce data redundancy** and boost **query performance**.
- It also provides a clear, logical data model, which makes it **easier** to **understand** the data structure.

Star Schema Disadvantages

- less flexible than other schema types.
- Adding or modifying tables can require extensive changes to the schema, and
- the star schema can struggle to manage complex relationships.

Snowflake schema.

- A snowflake schema is an extension of the star schema.
- It breaks down the dimension tables into multiple related tables.
- Existing tables in a star schema can be further denormalized into other tables, which creates a hierarchy.
- Yet these tables maintain a relationship with the dimension and central facts tables.
- For example, you can further normalize its **product** data into **supplier** and **category** data tables.

Advantage of Snowflakes Schema

- It provides more efficient data storage and retrieval.
- It improves data integrity and consistency, and
- it reduces data redundancy.
- It also offers scalability and flexibility by integrating new data tables as required.

Disadvantages of a snowflake schema

- It's more **difficult to perform** data analysis because of the extra relationships.
- These new relationships also make the schema **more challenging to understand** and manage, and
- They result in **slower queries**.

Validate Your Schema

- It's important to validate your schemas to make sure they're accurate.
- When validating a schema, you need to check for the following:
    - make sure **each table column** has been assigned the correct **data type**, like text and numeric.
    - Check that **each column** has the **correct formatting applied**.
    - Confirm that **all columns have clear descriptions** with relevant context, and
    - make sure all table and column properties are correctly configured.

Question

Which of the following are benefits of a schema? Select all that apply.

A. A schema helps with generating meaningful insights into your data.
B. A schema helps with the creation of visualizations.
C. A schema helps to enable efficient data analysis.
D. A schema helps to define the structure of your data.

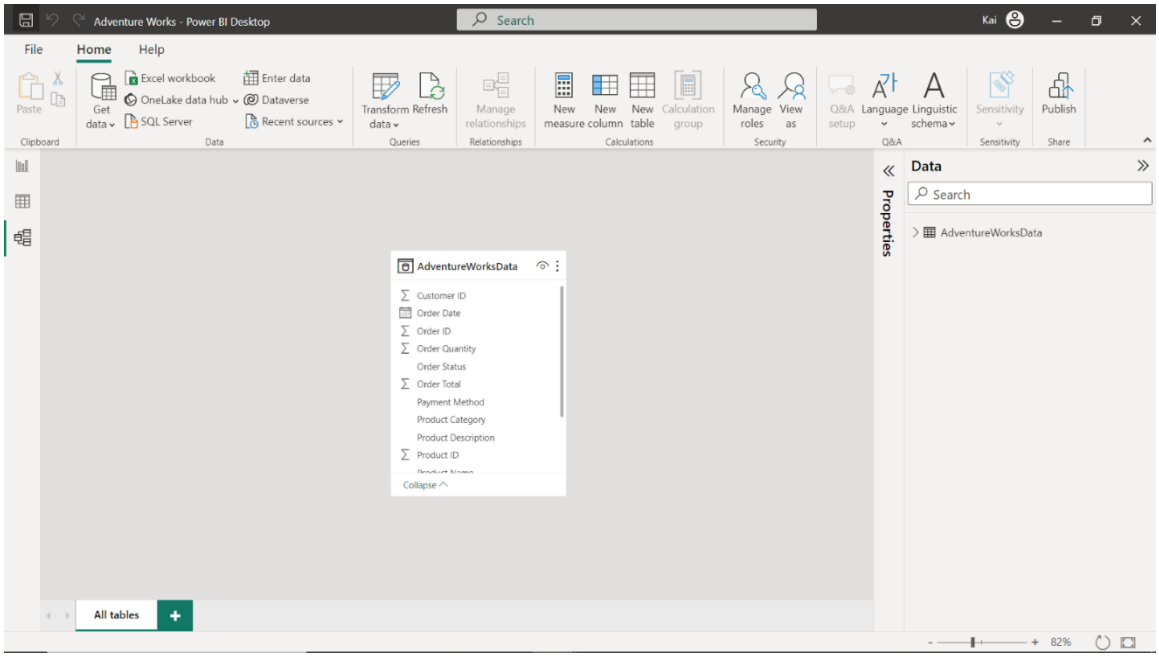
Example of Company Schemas

- A Company has decided to use Power BI for its data analytics. The company must design a schema with several interconnected tables to analyze and visualize its data effectively. Some of the tables in the company schema are as follows:

- **Reseller**: A table with reseller ID, contact information, and demographic data.
- **Regions**: Stores geographical location about the customers like region, country and city.
- **Sales**: Captures transaction data like date, sale amount, transaction ID, and quantities sold.
- **Products**: Stores data about various products, categories, and subcategories with Product ID.
- **Salesperson**: Records data about salespersons, employee ID, hiring data, and designation.

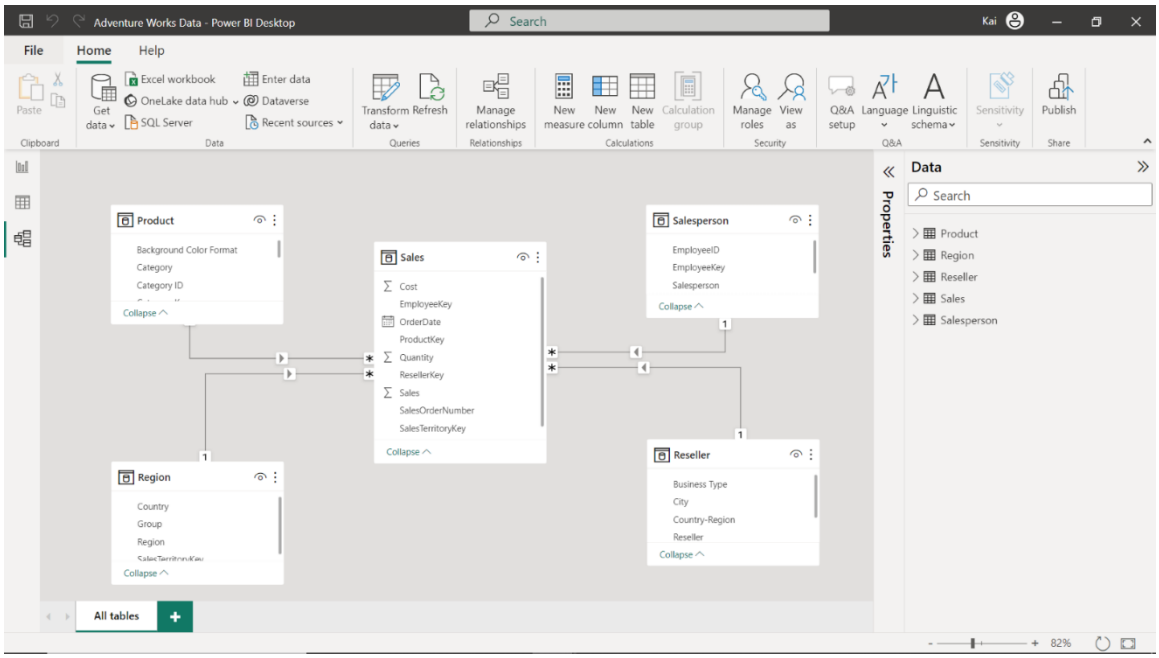
Let’s review the components of various schema using the company’s schema as our example.

Flat schema



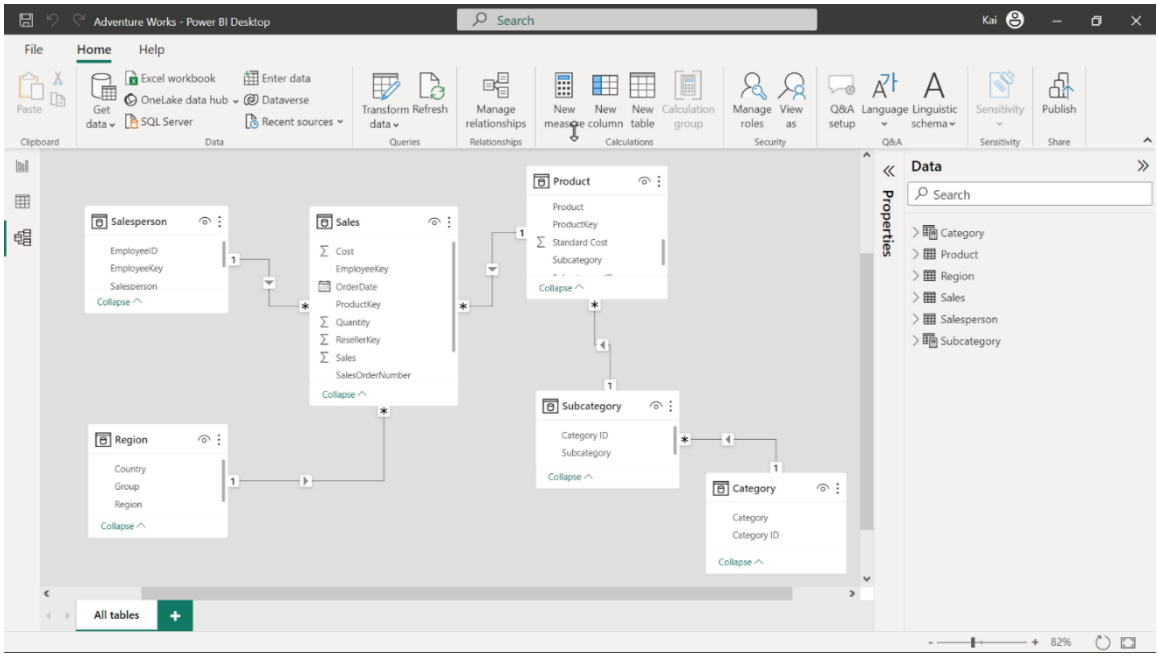
- A Flat schema is a simple database design where all data is stored in a single table. In this schema, each row represents a unique record, and each column represents the record's attributes.
- For **example**, in the **Sales** table, each row represents a sales transaction, while the various columns specify who bought the product, the date on which it was sold, and so on. Since there is only one table in the schema, you don't need to manage relationships between different tables.

Star schema



- A Star schema is a type of schema used in data warehousing and dimensional modeling. In this schema, a central fact table is connected to one or more-dimension tables based on the common field or column in both fact and dimension tables.
- In the Company schema, the **fact table** contains **quantitative data**, such as sales amounts or product quantities. In contrast, the **dimension tables** store **descriptive** data, such as customer information, product details, or dates.

Snowflake schema



- A Snowflake schema is an extension of the Star schema. In this schema, dimension tables are split into multiple related tables to reduce data redundancy and improve data integrity. This process is referred to as normalization.
- In the Company schema, each dimension table is connected to one or more related tables, forming a hierarchical structure that resembles a Snowflake.

The importance of choosing the right schema

- Understanding schemas and the available types is crucial for effective database design and management.
- The Flat schema is simple and easy to work with but may not be suitable for complex data relationships or large datasets.
- The Star schema is a popular choice for dimensional modeling, offering reduced data redundancy and intuitive design. Although it may be less flexible when handling complex relationships between dimensions.
- Lastly, the Snowflake schema provides greater flexibility and improved data integrity through normalizing dimension tables. However, its complexity can make it more challenging to understand and query.

Exercise 1 Configuring a Flat schema

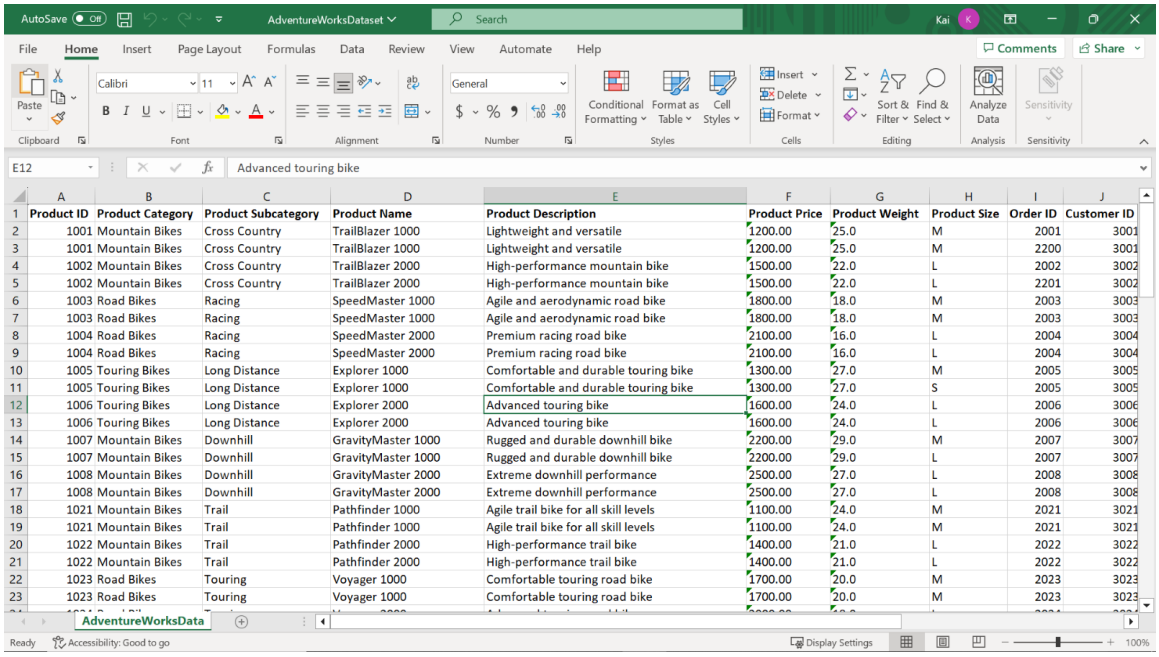
- User file **AdvetureWorksDataset1.xlsx**.

Scenario

- Adventure Works has seen a rise in customer complaints following incorrect and delayed deliveries. The company suspects inconsistencies in its data have caused the issues.
- To fix these issues, Adventure Works needs to create a data model in Power BI that accurately and consistently organizes and integrates its data. You can help the company to develop this data model as a Flat schema.
- The company provides you with an Excel file called ***AdventureWorksDataSet1.xlsx*** The file consolidates all required data into a table containing all relevant fields related to the company's products and orders.
- You must load this dataset into Power BI and develop it as a Flat schema. Be sure to evaluate the data quality and configure the model to ensure that Adventure Works can use it to make informed decisions.

Step 1: Explore the Excel File:

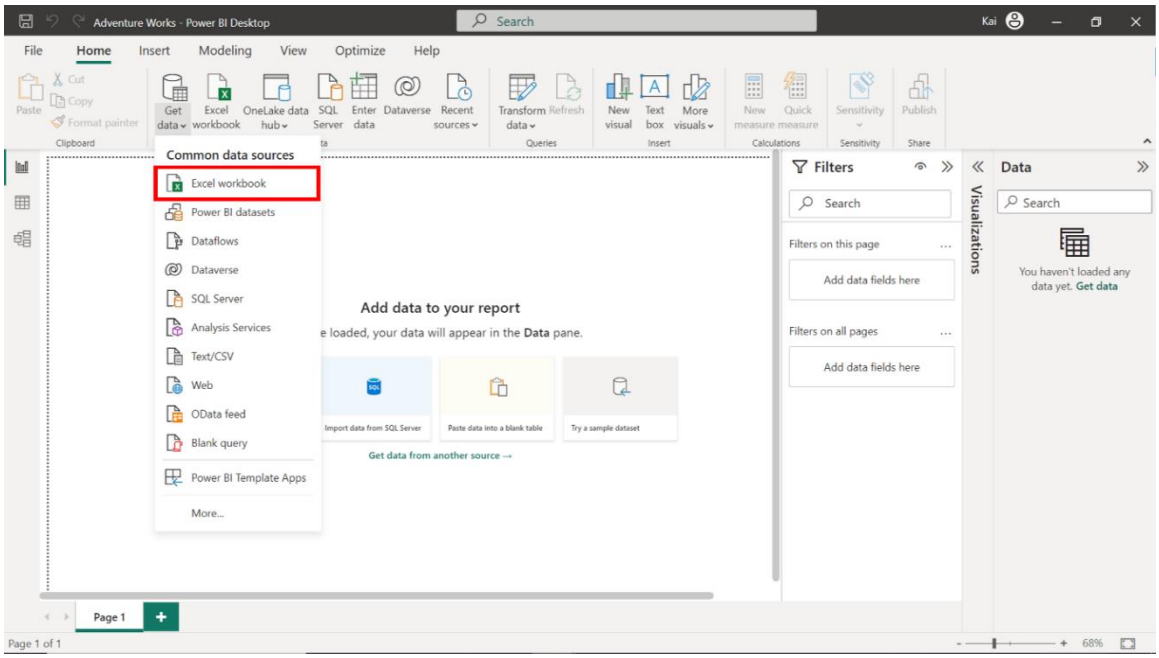
1. Open the Microsoft Excel workbook **AdventureWorksDataset.xlsx**. from your **Files** folder. The workbook contains one worksheet called **AdventureWorksData**, which includes information on the company's sales.



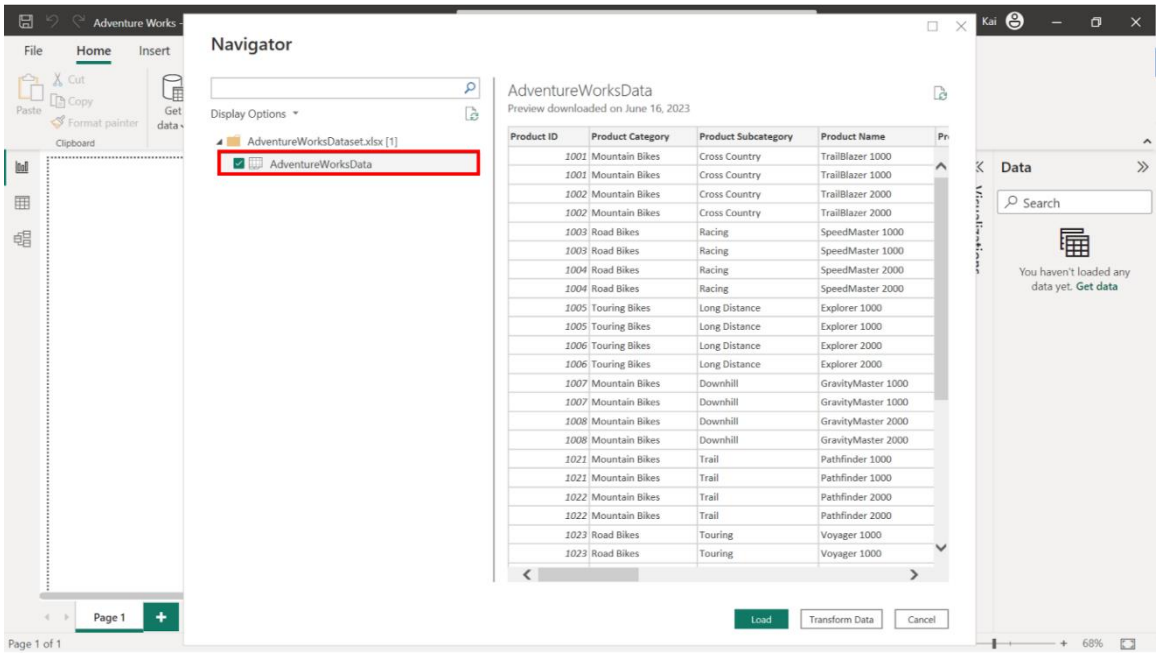
Product ID	Product Category	Product Subcategory	Product Name	Product Description	Product Price	Product Weight	Product Size	Order ID	Customer ID
1001	Mountain Bikes	Cross Country	TrailBlazer 1000	Lightweight and versatile	1200.00	25.0	M	2001	3001
1001	Mountain Bikes	Cross Country	TrailBlazer 1000	Lightweight and versatile	1200.00	25.0	M	2200	3001
1002	Mountain Bikes	Cross Country	TrailBlazer 2000	High-performance mountain bike	1500.00	22.0	L	2002	3002
1002	Mountain Bikes	Cross Country	TrailBlazer 2000	High-performance mountain bike	1500.00	22.0	L	2201	3002
1003	Road Bikes	Racing	SpeedMaster 1000	Agile and aerodynamic road bike	1800.00	18.0	M	2003	3003
1003	Road Bikes	Racing	SpeedMaster 1000	Agile and aerodynamic road bike	1800.00	18.0	M	2003	3003
1004	Road Bikes	Racing	SpeedMaster 2000	Premium racing road bike	2100.00	16.0	L	2004	3004
1004	Road Bikes	Racing	SpeedMaster 2000	Premium racing road bike	2100.00	16.0	L	2004	3004
1005	Touring Bikes	Long Distance	Explorer 1000	Comfortable and durable touring bike	1300.00	27.0	M	2005	3005
1005	Touring Bikes	Long Distance	Explorer 1000	Comfortable and durable touring bike	1300.00	27.0	S	2005	3005
1006	Touring Bikes	Long Distance	Explorer 2000	Advanced touring bike	1600.00	24.0	L	2006	3006
1006	Touring Bikes	Long Distance	Explorer 2000	Advanced touring bike	1600.00	24.0	L	2006	3006
1007	Mountain Bikes	Downhill	GravityMaster 1000	Rugged and durable downhill bike	2200.00	29.0	M	2007	3007
1007	Mountain Bikes	Downhill	GravityMaster 1000	Rugged and durable downhill bike	2200.00	29.0	L	2007	3007
1008	Mountain Bikes	Downhill	GravityMaster 2000	Extreme downhill performance	2500.00	27.0	L	2008	3008
1008	Mountain Bikes	Downhill	GravityMaster 2000	Extreme downhill performance	2500.00	27.0	L	2008	3008
1021	Mountain Bikes	Trail	Pathfinder 1000	Agile trail bike for all skill levels	1100.00	24.0	M	2021	3021
1021	Mountain Bikes	Trail	Pathfinder 1000	Agile trail bike for all skill levels	1100.00	24.0	M	2021	3021
1022	Mountain Bikes	Trail	Pathfinder 2000	High-performance trail bike	1400.00	21.0	L	2022	3022
1022	Mountain Bikes	Trail	Pathfinder 2000	High-performance trail bike	1400.00	21.0	L	2022	3022
1023	Road Bikes	Touring	Voyager 1000	Comfortable touring road bike	1700.00	20.0	M	2023	3023
1023	Road Bikes	Touring	Voyager 1000	Comfortable touring road bike	1700.00	20.0	M	2023	3023

Step 2: Connect to the Excel workbook and load the data into Power BI:

1. Launch the Power BI desktop and select **Get Data** to connect to the source.
2. Navigate to the folder containing the Adventure Works spreadsheet.



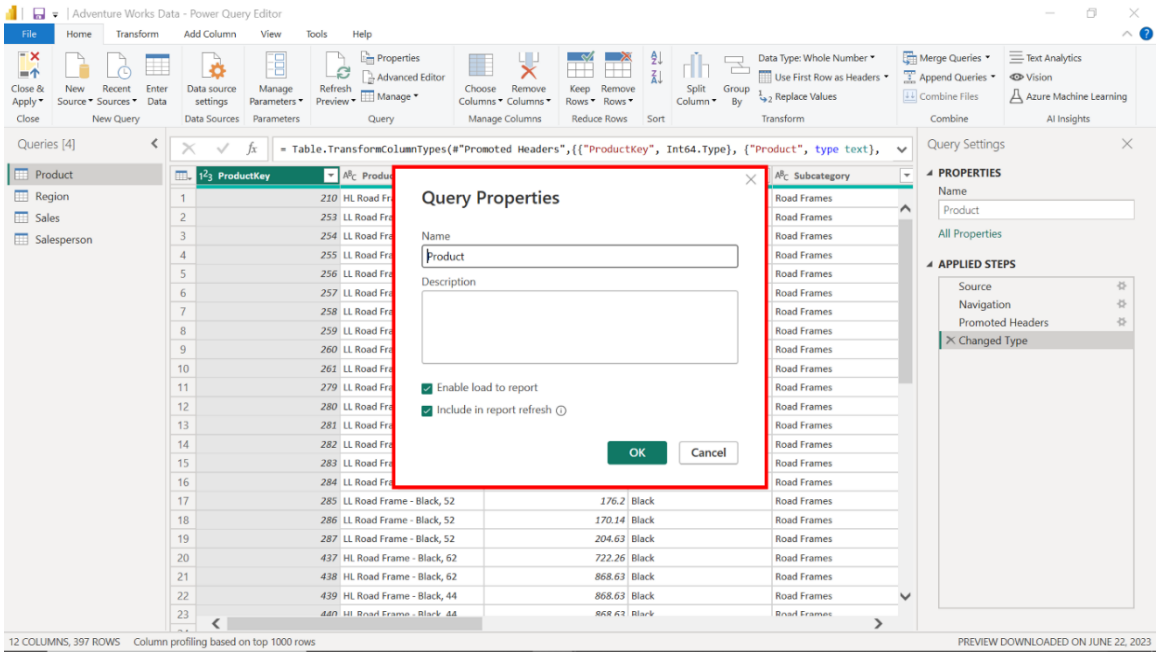
1. A navigator menu appears on the screen displaying a list of tables available within your dataset. In the navigator dropdown menu, select the **Adventure Works** dataset. Then select **Load**.



1. A preview of the dataset is visible in the preview pane.

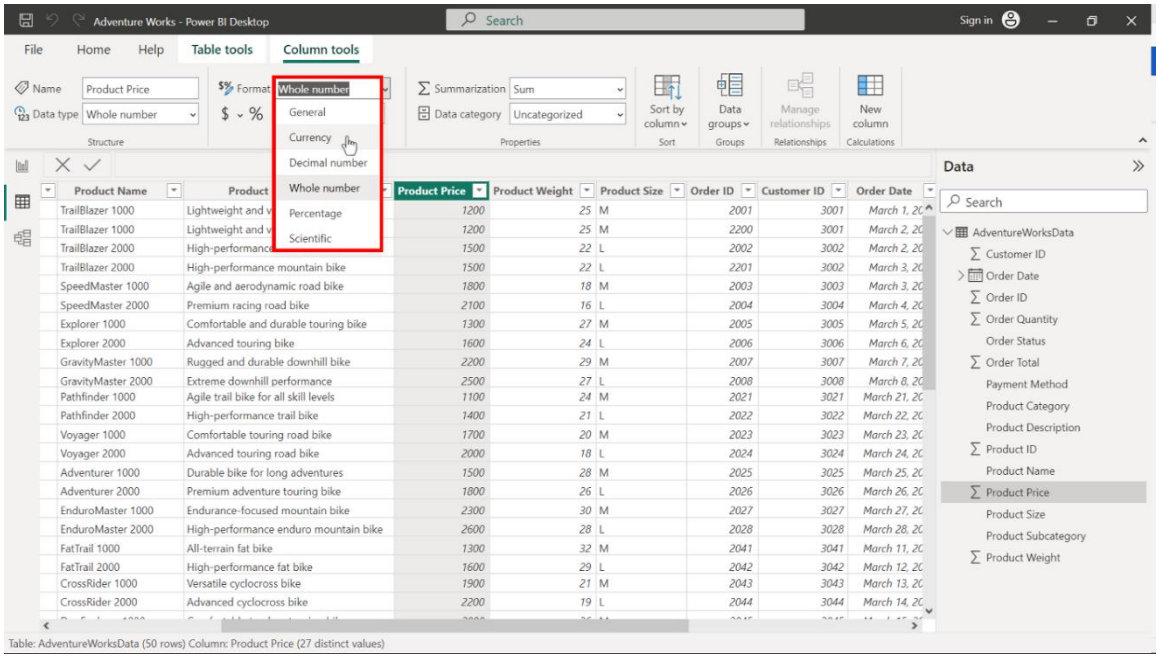
Step 3: Configure the table properties.

1. Open the **Home** tab and select **Transform data** to open the **Power Query** editor.
2. Once in the **Power Query** editor, select **Properties** to open the **Query Properties** dialog box.
3. Change the table's name to **Product** and add a dataset description.



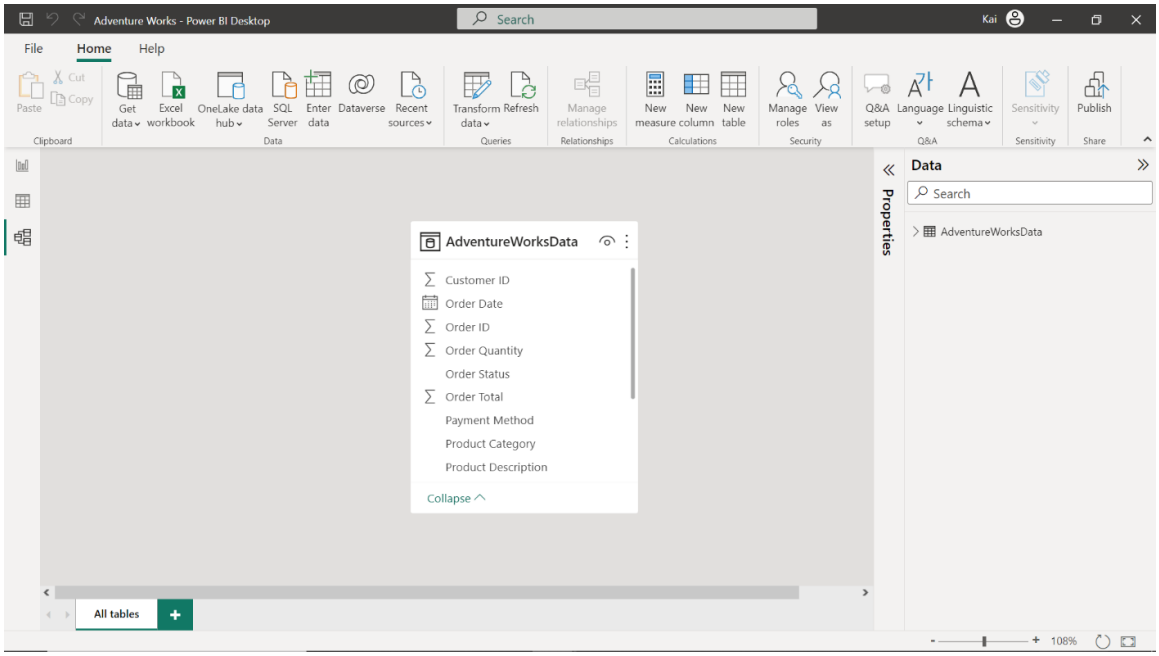
Step 4: Configure Column Properties.

1. In the **Home** tab of the query editor, select **Remove Rows**.
2. Select the **Remove duplicates** option from the dropdown menu.
3. Close and Apply Changes in Power query to get back to Power BI.
4. Select **Table View**.
5. Expand **Product** table in the **Data** Pane in the right.
6. Select the **Product Price** column.
7. Open the **Column Tools** tab in the Power BI desktop interface.
8. Navigate to the **formatting** group and select **Currency** from the dropdown menu. This action displays a **$** sign before the amount in the entire column.



Step 5: Evaluate the data model and save the Power BI project

1. To view the dataset loaded to Power BI, select **Model View** from the left sidebar. The **Model view** should display only one table with all columns as data attributes. This is the typical structure of a **Flat schema**.
2. Adventure Works can now use this Flat schema to build visualizations, reports, and dashboards to draw insights into its business operations.



1. To save the project, access the **File** menu and select **Save As**. Provide an appropriate name for the project and a suitable path to the folder on your local machine.

Exercise 1 Questions

Question 1

Which of the following steps are essential to ensure a well-structured and accurate flat table schema? Select all that apply.

A. Connecting to data sources.
B. Validating the schema.
C. Configuring column properties.
D. Merging all tables into a single table.

Question 2

How many tables were displayed in the **Model View** once the dataset was loaded?

A. A data structure that included multiple related tables.
B. A single table with multiple columns that included different data.
C. A single table with one-to-many relationships.

Question 3

How many rows were present in the dataset after all duplicate rows were removed from the **OrderID** column in the Adventure Works dataset?

A. 96
B. 48
C. 37

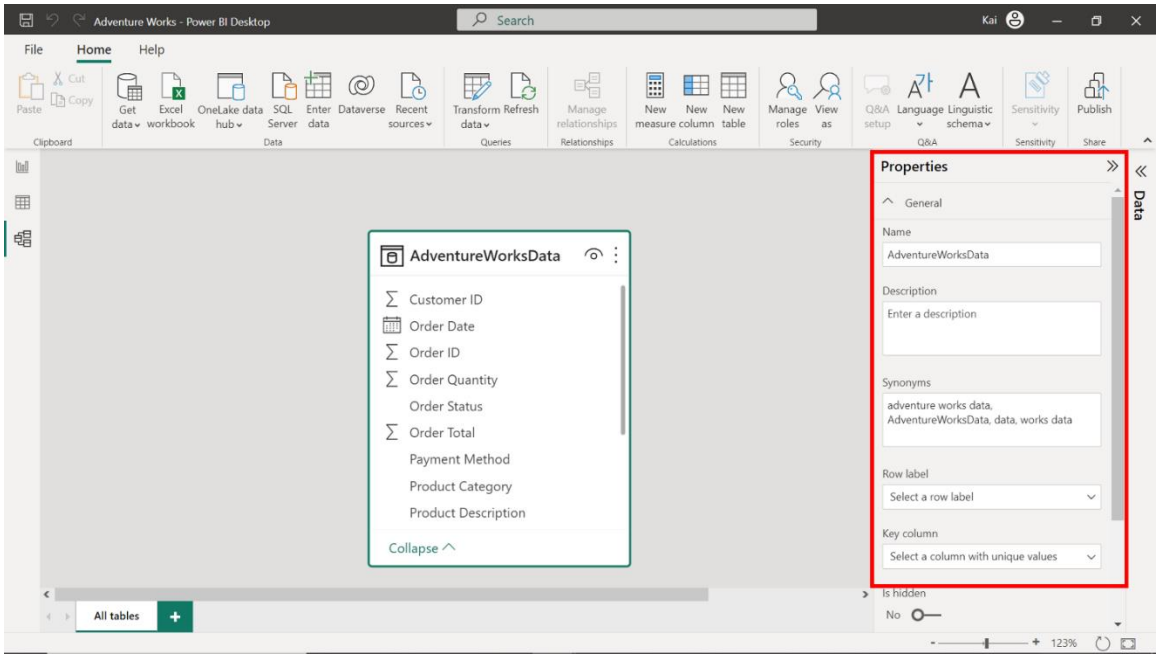
Question 3

What was the data type of the **Product Price** column after you loaded the data to Power BI before applying a transformation to the dataset?

A. Date
B. Decimal Number
C. Whole Number
D. Text

Table properties

- In Power BI, tables are the foundational data modeling, analysis, and visualization elements. Understanding and managing the different properties associated with tables is essential for maximizing the potential of Power BI.
- Some of the key properties of tables in Power BI are as follows:



Name

- **Definition:** The name of a table is a unique identifier that should be descriptive and easy to understand. Tables consist of rows and columns. It shows what the data is with a description in its columns. The data of the whole table is shown in rows.
- **Example:** In the Adventure Works scenario, tables can be named **Sales**, **Customer**, and **Product**. And columns within the **Sales** table could be named **OrderID**, **CustomerID**, **ProductID**, **Quantity**, and **Price**.

Table description

- **Definition:** The table **Description** property provides additional information or context about the table. It can be helpful to improve understanding and collaboration among team members working on the same data model.
- **Example:** In the Adventure Works **Order** table, a table description for the data model "Orders of sales of the product in the dataset" ensures clarity among team members regarding the purpose and content of the table.

Data category

- **Definition:** The **data category** property helps Power BI understand the context of your data by categorizing columns of the tables. It specifies data types, such as geographical locations, URLs, or email addresses.
- **Example:** In the Adventure Works **Customer** table, the column containing customer email addresses can be assigned the data category **email**, enabling Power BI to recognize and treat the data accordingly.

Summarization

- **Definition:** The summarization property determines how values in a numeric column are aggregated by default, such as **SUM**, **AVERAGE**, **MINIMUM**, **MAXIMUM**, or **COUNT**.
- **Example:** Adventure Works can set the summarization property of the **Quantity** column in the **Sales** table to **SUM**, enabling Power BI to automatically calculate the total quantity of products sold when creating visualizations.

Sort by column

- **Definition:** The **Sort by column** property specifies a column within the table that should be the default sort order for the table.
- **Example:** In the Adventure Works **Product** table, the **ProductName** column can be sorted in ascending order, making it easier to browse through the list of products.

Hide or show tables

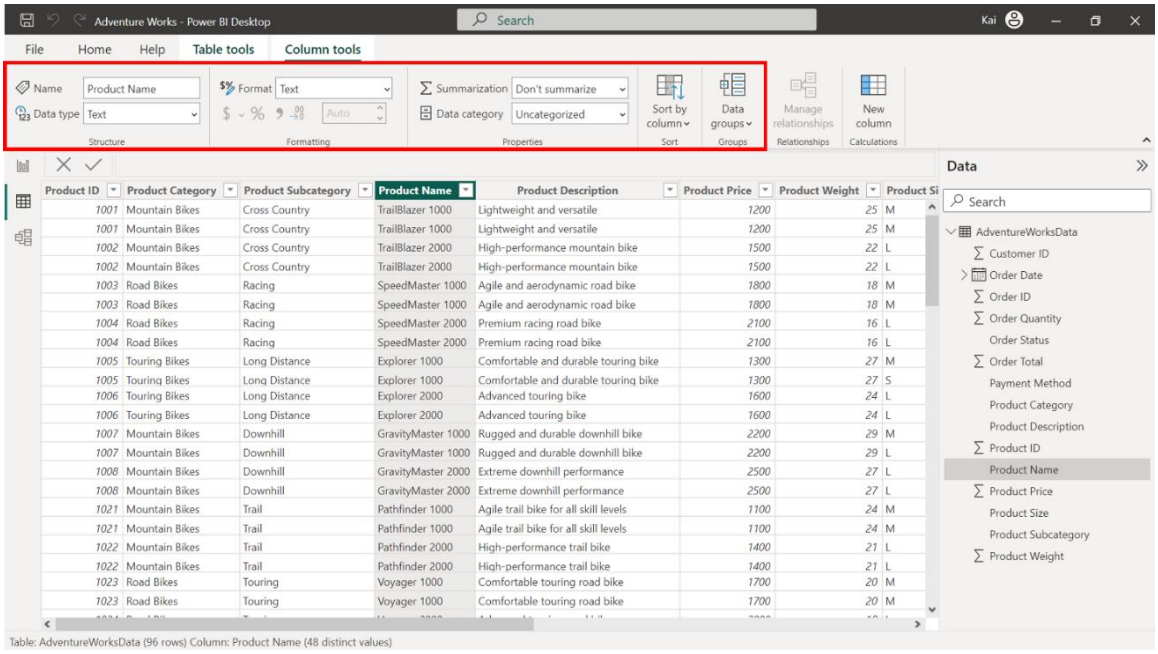
- **Definition:** This property allows you to hide or show tables in your data model. This can be useful when working with calculations or intermediate steps because it does not need to be visible to end users.
- **Example:** Adventure Works might hide tables containing sensitive customer calculations (like profit and annual income) from the **Customer** table to focus on relevant data for their analysis.

Relationship

- **Definition:** Relationships between tables are created to connect related data in a model, enabling you to analyze data across multiple tables.
- **Example:** Adventure Works can create relationships between the **Sales** table's **CustomerID** column and the **Customer** table's **CustomerID** column, as well as between the **Sales** table's **ProductID** column and the **Product** table's **ProductID** column. This allows for seamless data analysis across tables, such as calculating total sales by customer or analyzing the popularity of specific products.

Column properties

- In Power BI table view, column properties refer to the specific settings and characteristics associated with individual columns within a table.
- These properties influence how data in columns is displayed, treated, and used for analysis and visualization. Correctly configuring these properties enhances data accuracy, visualization quality, and overall user experience.
- Some common column properties are as follows:



Name

- **Definition:** The column **Name** property is a unique identifier that should be descriptive and easy to understand. This is used in formulas or when creating relationships with other tables.
- **Example:** The selected column in the Adventure Works dataset is **Product Name**.

Display name

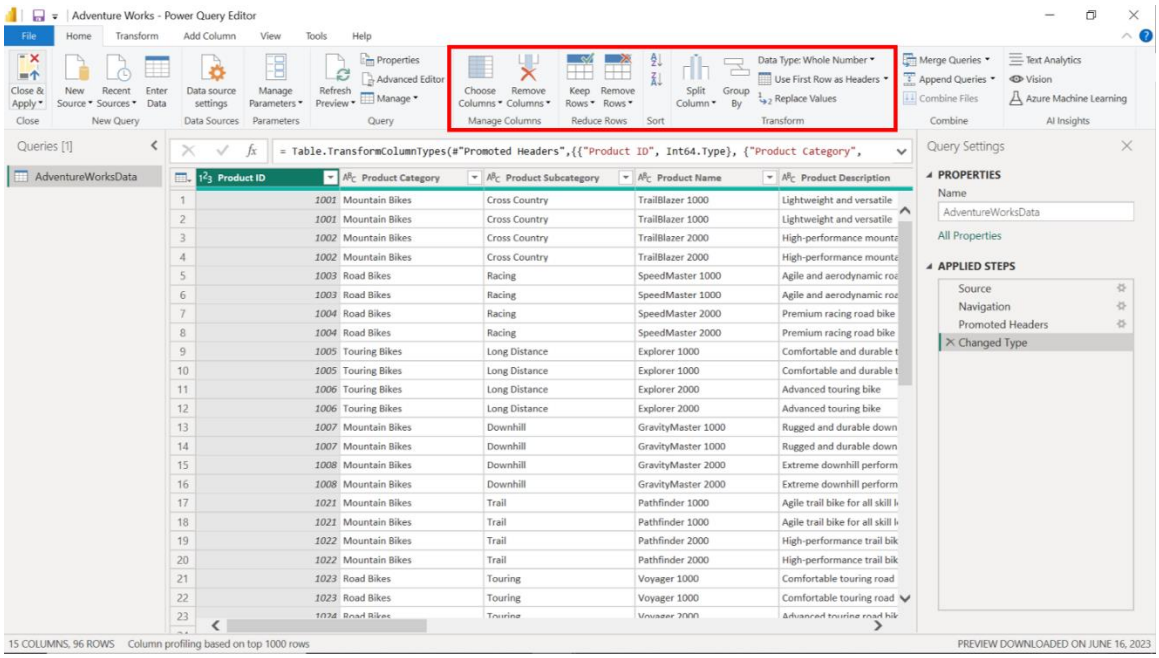
- **Definition:** The column **Display name** property is used to specify the user-friendly name of the column that appears in Power BI.
- **Example:** In the Adventure Works dataset, the display name of the selected column is **Product Name**.

Column description

- **Definition:** Column descriptions provide metadata about columns to improve understanding and collaboration among team members working on the same data model.
- **Example:** In the Adventure Works **Product** table, a column description for the **ProductWeight** column could be "Weight of the product in pounds," ensuring clarity among team members regarding the purpose and content of the column.

Data type

- **Definition:** The data type property determines the values a column can hold (such as text, whole number, decimal number, date/time).
- **Example:** In the Adventure Works **Sales** table, the **OrderID** and **CustomerID** columns can have the data type **Whole Number**, whereas the **OrderDate** column should have the **Date/Time** data type.



Format

- **Definition:** The format property allows you to customize the display of values in a column, such as currency symbols, decimal places, or date and time formats.
- **Example:** For Adventure Works, the **Price** column in the **Sales** table can be formatted to display currency symbols (such as **$**) and have two decimal places.

Sort order

- **Definition:** The **Sort order** property specifies the default sorting of a column's values (such as **ascending**, **descending**, or **custom**).
- **Example:** The Adventure Works **Product** table can have the **ProductName** column sorted in ascending order, making it easier to browse through the list of products.

Column Type

- **Definition:** Column types in Power BI include calculated columns and measures created using DAX (Data Analysis Expressions) formulas to perform calculations or aggregations on the data.
- **Example:** Adventure Works can create a calculated column in the **Sales** table named **TotalPrice**, which multiplies **Quantity** by **Price** to calculate the total price for each transaction.

Data category

- **Definition:** The data category property helps Power BI understand the context of your data by categorizing columns with specific data types, such as geographical locations, URLs, or email addresses.

- **Example:** In the Adventure Works **Customer** table, the column containing customer email addresses can be assigned the data category **email**, enabling Power BI to recognize and treat the data accordingly.

Summarization

- **Definition:** The summarization property determines how values in a numeric column are aggregated by default, such as **SUM**, **AVERAGE**, **MINIMUM**, **MAXIMUM**, or **COUNT**.
- **Example:** Adventure Works can set the summarization property of the **Quantity** column in the **Sales** table to **SUM**, enabling Power BI to automatically calculate the total quantity of products sold when creating visualizations.

Hide or show columns

- **Definition:** This property allows you to hide or show columns in your data model, which can be useful when working with large datasets or when specific columns are unnecessary in your analysis.
- **Example:** Adventure Works might hide columns containing sensitive customer information (like phone numbers or addresses) from the **Customer** table to focus on relevant data for their analysis.

Exercise 2: Configure a Flat schema with multiple sources

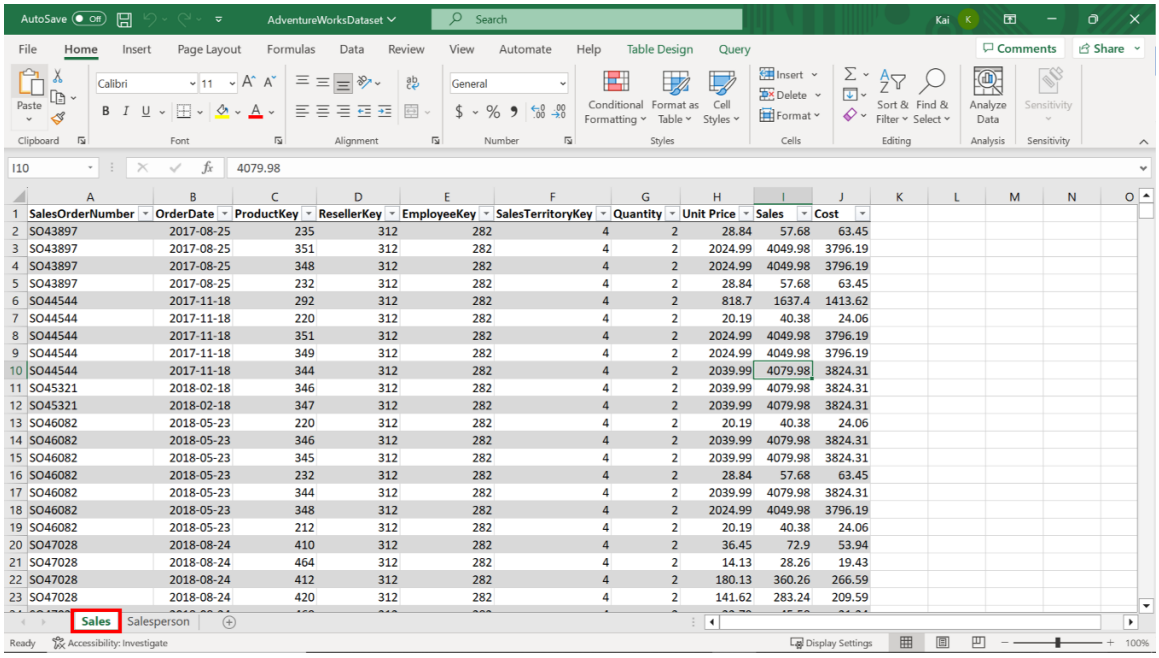
- Use file **AdvetureWorksDataset2.xlsx.**

Scenario

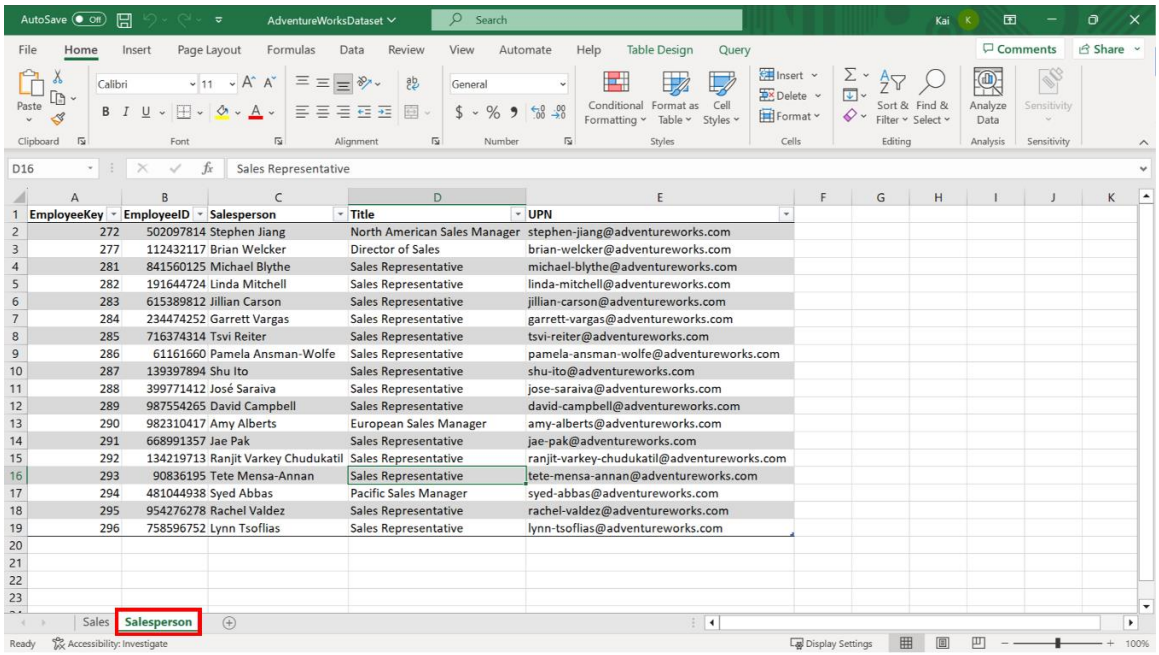
- Adventure Works' system saves sales records into a dataset with attributes like **product**, **category**, and **unit price**. On the other hand, the records of Adventure Works sales team employees are stored in a separate table called **Salespersons**.
- Adventure Works wants to analyze the performance of its sales team. So, it needs to create a data model in Power BI that accurately and consistently organizes and integrates sales data alongside the records of its sales team. You can help the company to develop this data model as a Flat schema.
- The company provides you with an Excel file called ***AdventureWorksDataSet***. The file contains two tables called **Sales** and **Salespersons**.
- You must combine both tables into a consolidated dataset containing all relevant fields related to the company's sales data and sales team.
- You must load this dataset into Power BI and develop it as a **Flat schema**.
- Be sure to combine the data tables and configure the model to ensure that Adventure Works can use it to make informed decisions.

Step 1: Explore the Excel File

- Open the Microsoft Excel workbook **AdventureWorksDataset.xlsx**. The workbook contains two worksheets named **Sales** and **Salespersons**, as depicted in the screenshots below.



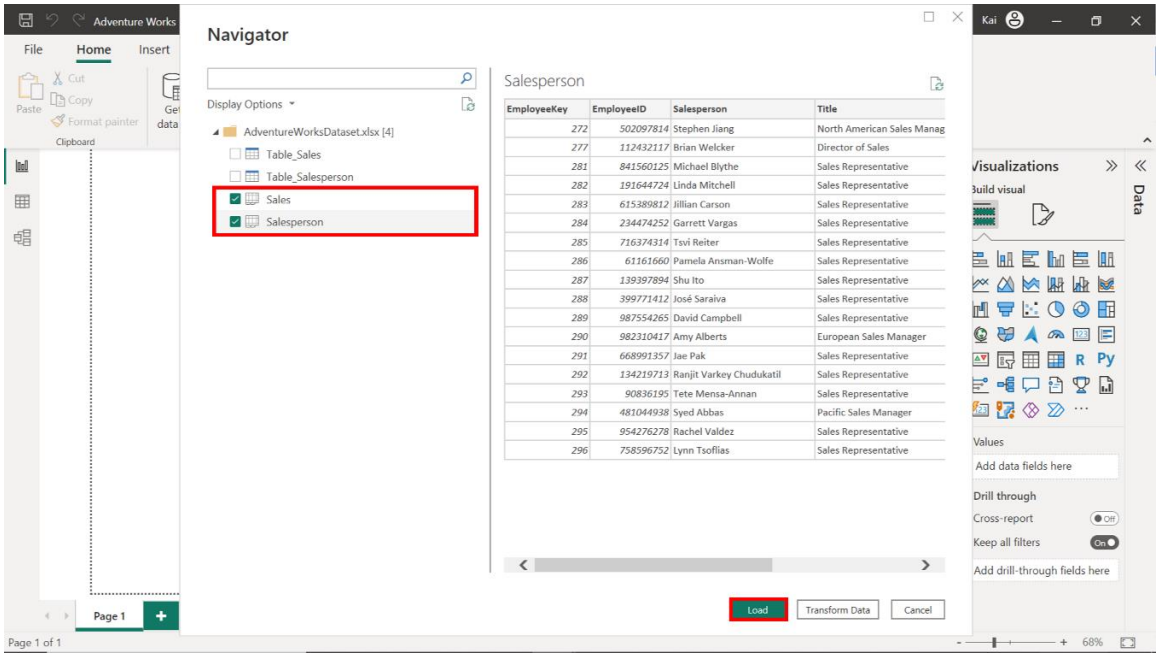
SalesOrderNumber	OrderDate	ProductKey	ResellerKey	EmployeeKey	SalesTerritoryKey	Quantity	Unit Price	Sales	Cost
SO43897	2017-08-25	235	312	282	4	2	28.84	57.68	63.45
SO43897	2017-08-25	351	312	282	4	2	2024.99	4049.98	3796.19
SO43897	2017-08-25	348	312	282	4	2	2024.99	4049.98	3796.19
SO43897	2017-08-25	232	312	282	4	2	28.84	57.68	63.45
SO44544	2017-11-18	292	312	282	4	2	818.7	1637.4	1413.62
SO44544	2017-11-18	220	312	282	4	2	20.19	40.38	24.06
SO44544	2017-11-18	351	312	282	4	2	2024.99	4049.98	3796.19
SO44544	2017-11-18	349	312	282	4	2	2024.99	4049.98	3796.19
SO44544	2017-11-18	344	312	282	4	2	2039.99	4079.98	3824.31
SO45321	2018-02-18	346	312	282	4	2	2039.99	4079.98	3824.31
SO45321	2018-02-18	347	312	282	4	2	2039.99	4079.98	3824.31
SO46082	2018-05-23	220	312	282	4	2	20.19	40.38	24.06
SO46082	2018-05-23	346	312	282	4	2	2039.99	4079.98	3824.31
SO46082	2018-05-23	345	312	282	4	2	2039.99	4079.98	3824.31
SO46082	2018-05-23	232	312	282	4	2	28.84	57.68	63.45
SO46082	2018-05-23	344	312	282	4	2	2039.99	4079.98	3824.31
SO46082	2018-05-23	348	312	282	4	2	2024.99	4049.98	3796.19
SO46082	2018-05-23	212	312	282	4	2	20.19	40.38	24.06
SO47028	2018-08-24	410	312	282	4	2	36.45	72.9	53.94
SO47028	2018-08-24	464	312	282	4	2	14.13	28.26	19.43
SO47028	2018-08-24	412	312	282	4	2	180.13	360.26	266.59
SO47028	2018-08-24	420	312	282	4	2	141.62	283.24	209.59



EmployeeKey	EmployeeID	Salesperson	Title	UPN
272	502097814	Stephen Jiang	North American Sales Manager	stephen-jiang@adventureworks.com
277	112432117	Brian Welcker	Director of Sales	brian-welcker@adventureworks.com
281	841560125	Michael Blythe	Sales Representative	michael-blythe@adventureworks.com
282	191644724	Linda Mitchell	Sales Representative	linda-mitchell@adventureworks.com
283	615389812	Jillian Carson	Sales Representative	jillian-carson@adventureworks.com
284	234474252	Garrett Vargas	Sales Representative	garrett-vargas@adventureworks.com
285	716374314	Tsvi Reiter	Sales Representative	tsvi-reiter@adventureworks.com
286	61161660	Pamela Ansman-Wolfe	Sales Representative	pamela-ansman-wolfe@adventureworks.com
287	139397894	Shu Ito	Sales Representative	shu-ito@adventureworks.com
288	399771412	José Saraiva	Sales Representative	jose-saraiva@adventureworks.com
289	987554265	David Campbell	Sales Representative	david-campbell@adventureworks.com
290	982310417	Amy Alberts	European Sales Manager	amy-alberts@adventureworks.com
291	668991357	Jae Pak	Sales Representative	jae-pak@adventureworks.com
292	134219713	Ranjit Varkey Chudukatil	Sales Representative	ranjit-varkey-chudukatil@adventureworks.com
293	90836195	Tete Mensa-Annan	Sales Representative	tete-mensa-annan@adventureworks.com
294	481044938	Syed Abbas	Pacific Sales Manager	syed-abbas@adventureworks.com
295	954276278	Rachel Valdez	Sales Representative	rachel-valdez@adventureworks.com
296	758596752	Lynn Tsoflias	Sales Representative	lynn-tsoflias@adventureworks.com

Step 2: Get data from the Excel workbook

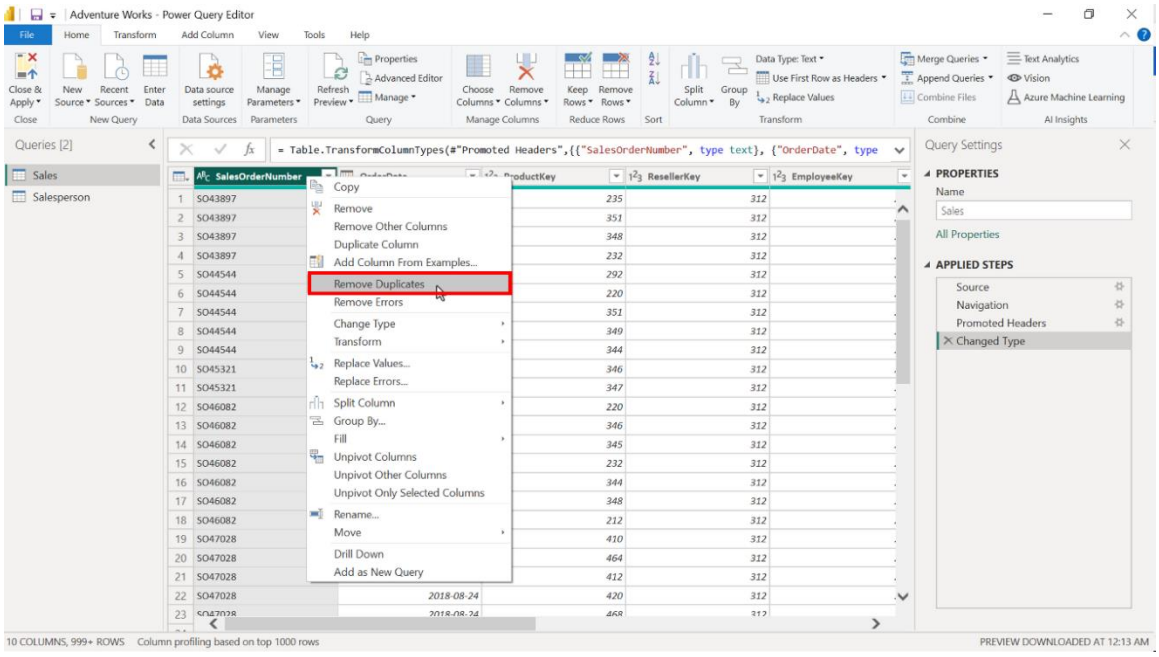
1. Import the data from the Excel sheet into Power BI.
2. Open a preview of the table in the **Preview pane**.



Step 3: merge the tables into one dataset.

1. Identify and remove all duplicate values in the **SalesOrderNumber** column in the **Sales** dataset.

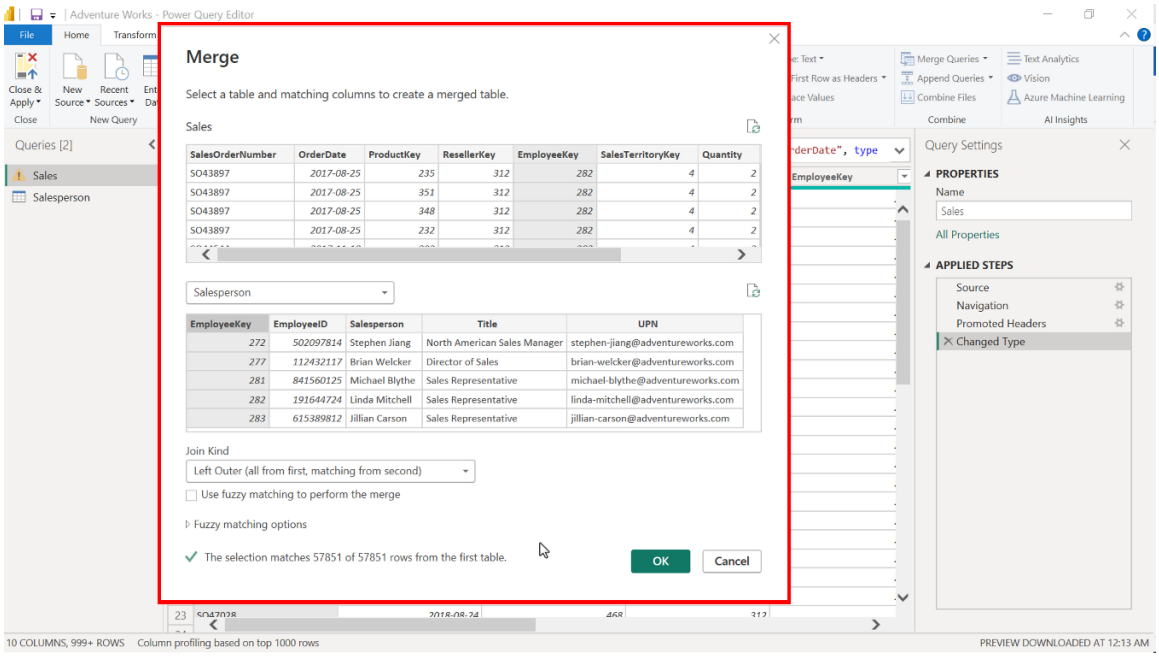
Tip: You can remove duplicate values using the **Remove Duplicates** feature.



1. Identify common columns with matching or similar values in both tables. These values can be used to merge both tables.

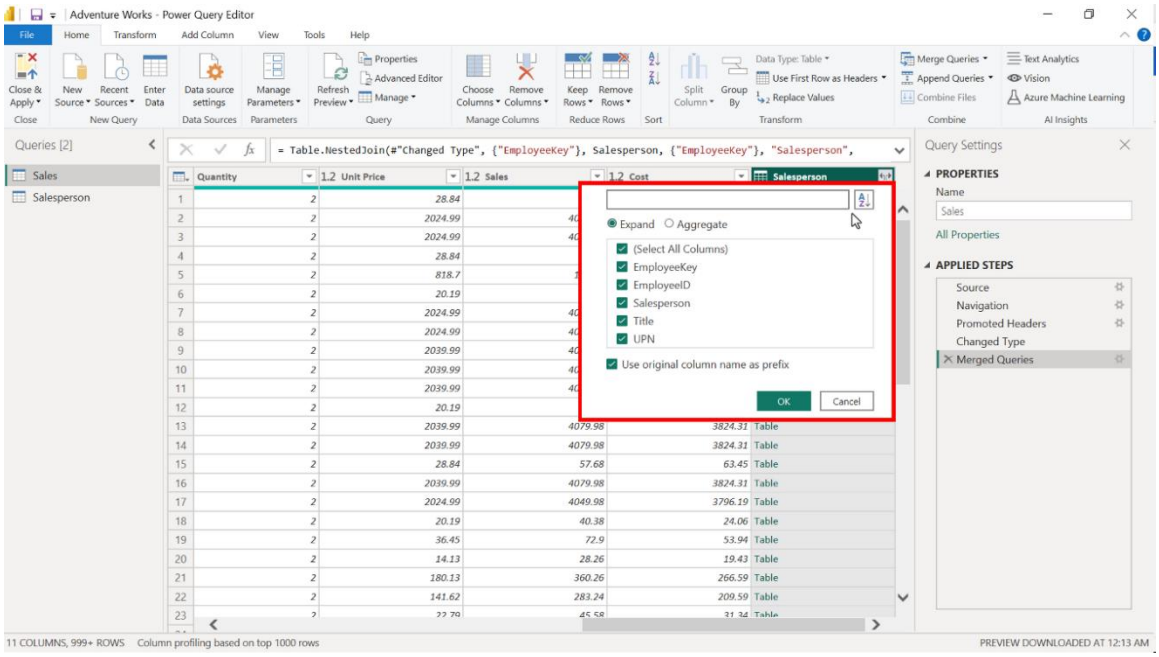
Tip: You can check the columns related to **employees** at both tables.

1. In the **Query Editor**, select the **Sales** table and click on **Merge queries**. This action opens the **Merge** dialog box in which you can configure the merge options. You must select the second table to merge with, ensuring you match the column and **Join Kind**.



1. The merged table is displayed as a column at the end of the table. Select the **column** to expand the merged table. Select **Employee ID**, **Salesperson**, and **Title** columns to expand and select **OK**.
2. After merging the tables, you can delete the **Salesperson** table from the model, as this table has been integrated with the **Sales** table as a Flat schema.
3. Try to delete **Salesperson** table ➔ you get error message➔ why?
4. Instead of deleting the **Salesperson** table just right click and chose enable load option.
5. Close and **Apply** Changes.
6. Go to **Report View**, **Model View** and **Table View** Only Sales table were loaded to model and appears in Data Pane.

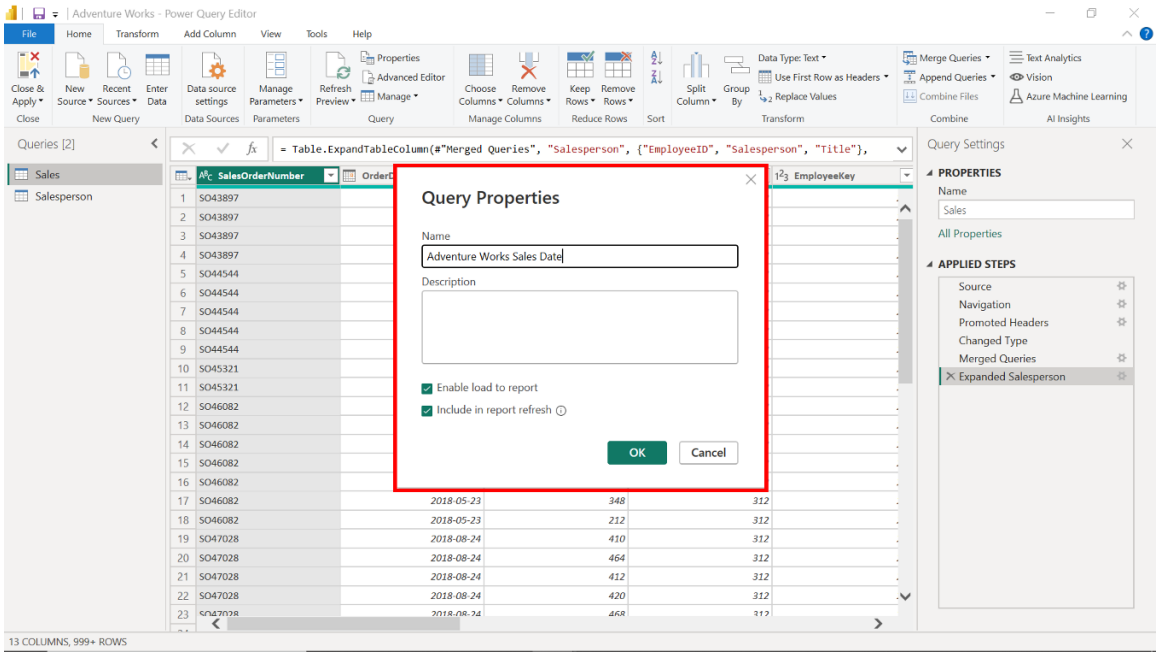
Tip: You can combine tables using **Merge queries** in the **Power Query** editor and the **Left outer** join type.



Step 4: Configure table and column properties.

1. Configure the table properties by renaming the table **Adventure Works Sales Data** and adding a brief description of the table in Power BI desktop.

Tip: You can configure table properties in the **Model view** of the Power BI desktop.

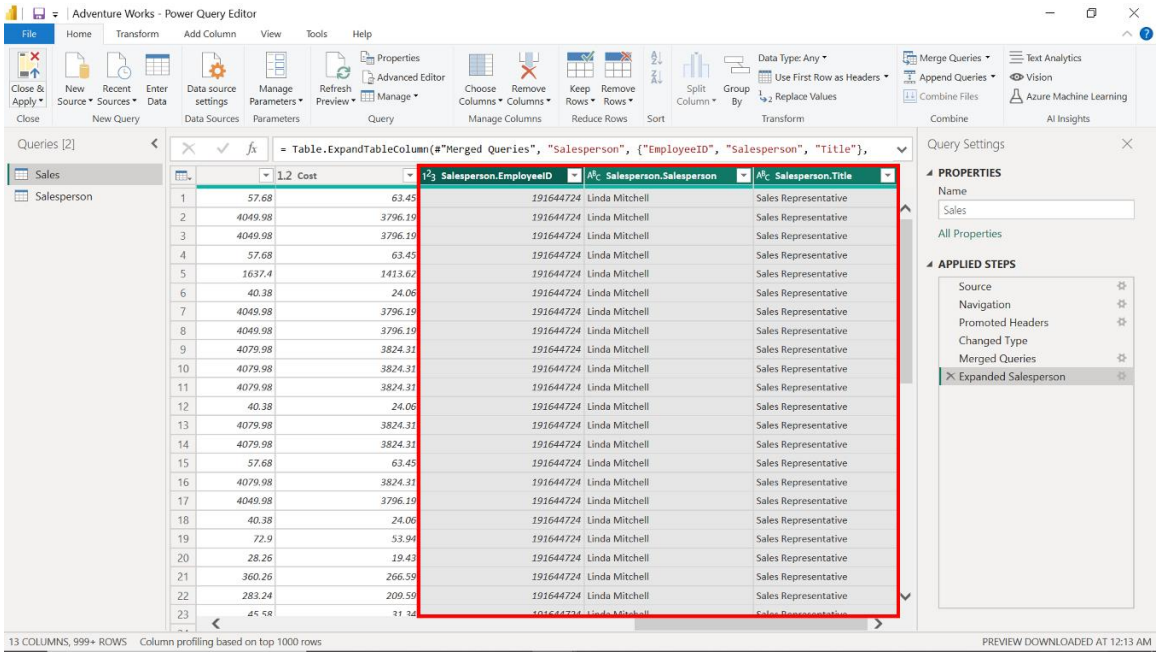


1. Configure column properties as follows:

- Merge queries autogenerate column names. Rename the columns as follows:
  - **Saleperson.EmployeeID** to **Employee ID**,

    - **Salesperson.Salesperson** to **Salesperson**
    - and **Salesperson.Title** to **Title**.
- Change and format the data types of the columns.
- Add additional information to the columns.

Tip: You can configure column properties both in the **Power Query editor** and in the **Model view** of the Power BI desktop.



Step 5: Save the Power BI project.

- Save your Flat schema Power BI project to your local machine.

Knowledge Check

Question 1

In Power BI, relationships are established between the tables based on _____ that match between the tables.

A. Column fields
B. Table properties
C. Rows

Question 2

What is the primary characteristic differentiating a Snowflake schema from a Star schema?

A. Denormalized dimension tables.
B. A hierarchical structure.
C. A central Fact table.
D. Normalized dimension tables.

Question 3

What are the limitations of using a Flat schema in Power BI? Select all that apply.

A. A Flat schema cannot be used to perform aggregations.
B. A Flat schema offers a lack of flexibility for organizing data from multiple sources.
C. A Flat schema offers limited capacity for storing large volumes of data.

Question 4

True or False: In Power BI, a schema is automatically created when you import data from various sources and establish relationships between tables.

A. True
B. False

Question 5

Which property cannot be adjusted for a table or column in Power BI?

A. Table relationship
B. Data type
C. Sort order

Chapter 2: Tables Relationships

Fact and Dimension Tables

- As you discovered You can use schemas for data organization. And two central components of all schemas are **fact** and **dimension** tables.
- As you learned earlier, a schema is a **logical and visual representation of how your fact and dimension tables relate**.
- They're the backbone of schemas in Power BI.

Fact Tables

- Fact tables are called fact tables because they consist of the **measurements**, **metrics**, or **facts** of a business process. In other words, they hold **quantifiable measurable** data.
- For example, In a star schema **Sales Orders** table it includes **transaction details** like **Order ID**, **Product ID**, **Customer ID**, **Quantity**, and **Total Price**. These are **core facts** about transactions, like the **customer** who made the purchase, the **price** of the product they purchased, and so on.
- And this fact table is related to dimension tables.

Dimension Tables

- Dimension tables are typically **textual fields** and provide **descriptive attributes** related to fact data.
- They offer the context surrounding a business process event.
- In the Star schema, the dimension tables are linked to the fact table and include **date**, **customer**, **sales**, and **product** data.
- These are **descriptive details** that can be used to identify individual customers.

Tables in Star Schema

- In the Star Schema model, the fact table sits at the center. The dimension tables radiate out like the points of a star. Each dimension table is directly connected to the fact table.

- For example, the Sales Order table is the central fact table in the Adventure Works Star schema. The dimension tables like date, customer, and product are connected directly to it.
- This structure simplifies queries because you only need to navigate through two tables to answer questions like, what were the total sales on a particular date?
- And these fact and dimension tables can also be used to extend a Star schema into a Snowflake schema.

Tables in Snowflake Schema

- A Snowflake schema makes use of dimension tables by **normalizing** them.

Normalization

- Normalization means that existing tables within a schema are divided into additional related tables.
- This technique creates a structure that resembles a snowflake. This is where we get the name Snowflake schema from.
- For instance, in addition to a central **fact table**, **product** dimension table could be split into a product table connected to **subcategory** and **category** tables.
- This schema reduces data redundancy, but adds complexity to queries.

Question

Adventure Works is building a star schema in Power BI. Which of the following tables in the schema can be used to store measurable business data like order and product IDs, quantities, and total cost?

A. A **Customer** table that holds data on customers.
B. A **Sales** table that contains data on sales transactions.
C. A **Product** table that contains information on products.

Normalization and denormalization

- One of the most important jobs of a data analyst in Microsoft Power BI is creating and managing data models.
- By executing this task effectively, you can make it easier for your team to understand the data. However, one of the questions you'll often face is "to normalize or denormalize?"
- An appropriately designed data model provides the following advantages:
    - Faster data exploration

- Easier aggregate creation
- Precise reporting
- Quicker report creation
- Simpler report maintenance

However, further advantages can also be provided by using normalization or denormalization.

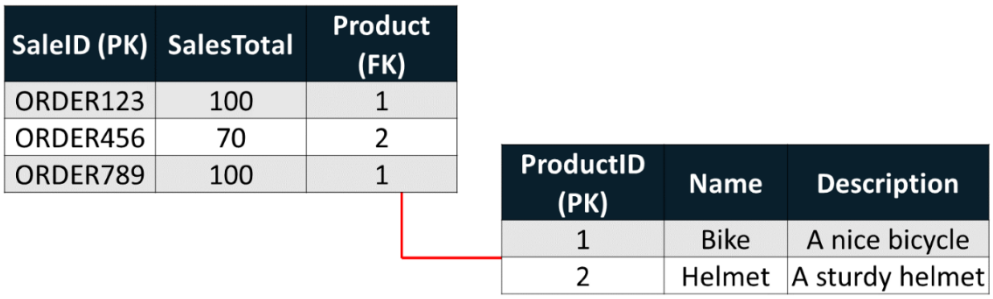
Normalization

- Normalization is a data model design technique that involves structuring **data to minimize redundancy and ensure data integrity**.
- It divides data into multiple related tables, each with a specific purpose.
- This approach **reduces data duplication**.
- However, it often requires creating complex relationships between the tables.

Normalization offers the following advantages:

- Removal of redundant data.
- Improved data integrity.
- Easier maintenance of your data model.

With normalization, relationships are established between tables by using primary and foreign keys. Primary keys are columns that uniquely identify each row of data.



SaleID (PK)	SalesTotal	Product (FK)
ORDER123	100	1
ORDER456	70	2
ORDER789	100	1

ProductID (PK)	Name	Description
1	Bike	A nice bicycle
2	Helmet	A sturdy helmet

- For example, the **Product** table has a unique key column named **productid**. Additional columns represent product attributes such as **name** and **description**.

- The second table, **Sales**, has a unique key column named **SaleId**, and a **SalesTotal** column. It also has a foreign key column named **Product**. Each field in this column represents an **ID** in the **Product** table.
- The relationship between the **Sales** table and the **Product** table is an example of a **many-to-one relationship**. In other words, there can be many rows in the **Sales** table related to one row in the **Product** table.
- This schema is normalized because the **Sales** table uses a foreign key to associate each row with a specific product.

Denormalization

- Denormalization is the reverse process of normalization.
- It involves converting the normalized schema into a schema that has redundant information.
- Implementing denormalization helps to avoid expensive queries between the tables but at the cost of creating redundant or duplicated data.

To demonstrate, let's return to the previous example. To denormalize the Adventure Works schema, the **Sales** table must contain a **SaleId**, **SalesTotal, ProductName** and **ProductDescription** columns.

SaleID	SalesTotal	ProductName	Product Description
ORDER123	100	Bike	A nice bicycle
ORDER456	70	Helmet	A sturdy helmet
ORDER789	100	Bike	A nice bicycle

Note how the product information is now duplicated in multiple rows.

On the other hand, denormalization offers some benefits:

- Instead of loading multiple data tables in your data model, you have only one large table, which could be more efficient in performance. With no relationship between tables, queries to join the data are reduced.
- Filter propagation between the tables may sometimes be less efficient than filters applied to a single table.
- Establishing a hierarchy from fields within a table is simpler.

However, despite these benefits, you don’t have to create a denormalized data model for every dataset. During the design stage, you must consider other factors that might also affect your model, like analytical and business requirements, your data sources, and the size of your model.

Difference between normalization and denormalization

- There are several differences between normalized and denormalized models, which it’s important to be aware of:

Data integrity

- Data integrity is maintained during the normalization process.
- On the other hand, data integrity is harder to maintain with a denormalized model. This is because multiple rows may need to be updated when data changes.

Redundant data

- Redundant data is eliminated when normalization is performed.
- However denormalization increases redundant data.

Model size

- Normalization increases the number of tables and join queries (due to relationships between tables).
- In contrast, denormalization reduces the number of tables and relationships.

Memory

- Disk space is overused in a denormalized schema because the same data is stored elsewhere.
- On the contrary, disk space is optimized in a normalized schema.

Normalizing and Denormalizing Data in Power Query

Now that you’re familiar with normalization and denormalization let’s review how these techniques are implemented.

Load your data

- When data is loaded into Power BI from a single data source, it likely represents a denormalized data structure.
- In this case, you can use the Power Query editor to transform and shape your source data into multiple normalized tables.
- Once you’ve performed normalization in Power Query, you can build relationships between your data tables in the modeling view of Power BI desktop.

Design your model

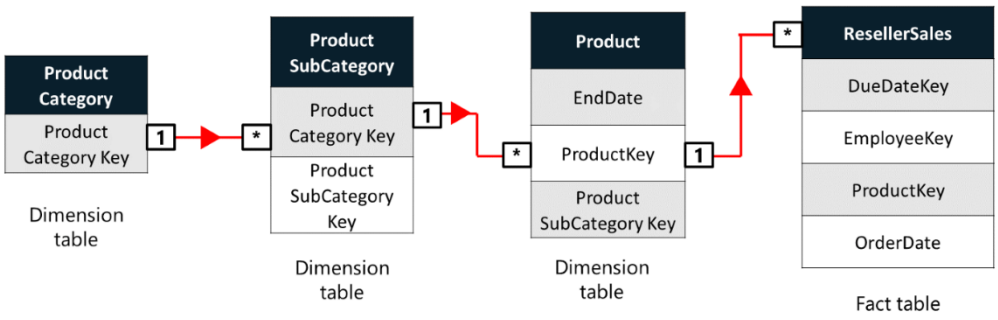
- Your responsibility is to design an optimized data model with tables that represent the business requirements.

- Any decision around normalization or denormalization must be aligned with these requirements.

Example:

- In the following example, It classifies products by **category** and **subcategory**.
- Products are assigned to **subcategories**. While **subcategories** are assigned to **categories**.
- In this model, the **Product** dimension table is normalized and stored in three separate tables:

- **Category**,
- **Subcategory**,
- and **Product**.



Denormalize your model

It's in the business interest of Adventure Works to denormalize this model. Its data analysts can use Power Query to transform these tables into a single product table using the **Merge queries** feature.

Cardinality and Granularity

- The data required to answer the questions is stored across several tables, posing a complex data analytics challenge.
- You can solve this challenge using cardinality, and by identifying the table relationships.

Cardinality

- In the context of data analytics, cardinality refers to the nature of relationships between two datasets. In other words, how tables in your database relate to each other.
- It's important that your cardinality settings are correct. Incorrect settings can lead to inaccurate data analysis and flawed business decisions.
- There are three types of cardinalities, or relationships between tables in Power BI.

The first is a one-to-one relationship.

- In this instance, a record in one column of Table A corresponds to a unique record in one column of Table B.
- One-to-one relationships are less common in data modeling, but they are useful when dealing with specific scenarios.
- For example, a single business entity can be loaded as two or more model tables because the data might come from different sources.
- This scenario is common for dimension tables. For example, in a dataset, each bicycle model has a unique model ID listed in the product ID column and a separate table lists specific features for each model ID, and a product features column. Together, these columns form a one-to-one relationship between the two tables.

The one-to-many relationship.

- Each record in a column of Table A, corresponds to multiple records in a column of Table B. But not the other way around.
- For example a company lists its stores in Table A, and it lists the employees of each store in Table B.
- The relationships between the stores and their employees establish a one-to-many relationship. This is because each employee works for one store, but each store has many employees.
- This is the most common type of relationship in data modeling, where one table acts as the primary table and the other tables act as related tables.

The many-to-many relationship.

- This is where multiple records in a column of Table A are related to multiple records in a column of Table B, in both directions.
- Many-to-many relationships are often used to establish a relationship between two fact tables or two dimension tables.
- In a company tables, a customer can purchase many different bicycle models logged in Table B. Each bicycle model can be purchased by multiple customers, recorded in Table A.
- This creates a many-to-many relationship.

Granularity

- Granularity refers to the level of detail or depth of a dataset.
- The granularity of your data should align with the business questions you need to answer.

- For example, if your company wants to view customer purchase histories over the past year. With the **days** of granularity, you can explore individual transactions to analyze individual customer behavior and identify purchase patterns.
- However, if you want to understand which specific bicycle models are performing well in a **region**, you need **sales data** with **high granularity**.
- **High granularity data is the dataset that captures detailed information about the transactions**.
- For example, geographical sales of products can be captured as a **continent**, **country**, **state**, **city**, and all the way down to individual **stores**.
- But for a more general analysis, like **total sales per store**, **a lower level of granularity suffices**.
- **Low granularity data refers to the dataset that captures a high level summary, or an aggregated level overbroad or categories**.
- An example of this is **monthly sales of a product category**. The sales data is summarized at the category level, but only on a monthly basis.
- Understanding the granularity of your data is crucial for establishing correct cardinality.
- It also influences how you set up your cross-filter direction in Power BI, which you will learn more about in a future lesson.
- But be careful when judging the required level of granularity. Misjudging the level of granularity can lead to misrepresented data and incorrect business insights, and excessive granularity can lead to too much data and slow down your queries.

Question

You are working on two tables where each record in a column of **Table A** corresponds to multiple records in a column of **Table B**, but not vice versa. What kind of relationship, or cardinality, is this an example of?

A. A one-to-one relationship.
B. A many-to-many relationship.
C. A one-to-many relationship.

Managing model relationships

- At your company, huge volumes of data are generated from various sources, including **customer management**, **inventory management**, **product**, and **sales** transaction records, individual **store** sales, and more.
- The data seems overwhelming and disjointed. Fortunately, Power BI's data modeling tool can create a coherent structure from these disparate data sources.

What are model relationships?

- In data analytics, a data model conceptualizes a data structure, the relationships between the different data tables, and the rules that govern these relationships.
- Data models are the basis of any data exploration, particularly in Microsoft Power BI. They transform diverse data into coherent, visually immersive, and interactive insights. This crucial process transforms the raw data into valuable, actionable insights.

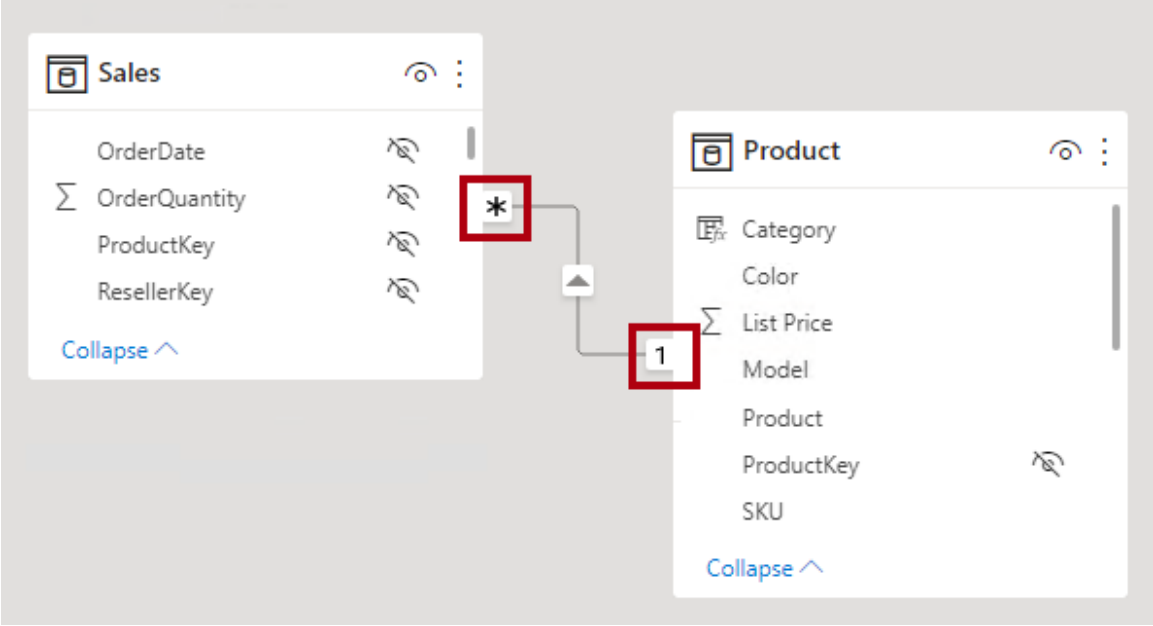
The purpose of relationships

- In Power BI, data tables are connected via model relationships. These relationships can be filtered in one or two directions based on the selected cardinality and cross-filter direction.
- The filter typically propagates from **one side** of the relationship to the '**many'** side. The flow (propagation) of the filter occurs if there is a clear relationship path connecting the tables involved.
- The relationship paths are deterministic in Power BI. This means that filters are propagated similarly without any random variation.
- However, relationships can be disabled or altered in the filter context to meet the specific analysis needs using DAX calculations. Types of cardinality in Power BI

Cardinality refers to the nature of relationships between two data tables. In other words, how tables in a data model relate. Three different types of cardinality, or relationships, can exist between data tables.

- One-to-one
- One-to-many
- Many-to-many

You can identify the relationship between tables in the **Model view**. For example, the diagram below depicts a one-to-many relationship. For each row in the **Product** table, representing an individual product, there are many rows in the **Sales** table, each representing a sale of that product.



Creating relationships in Power BI

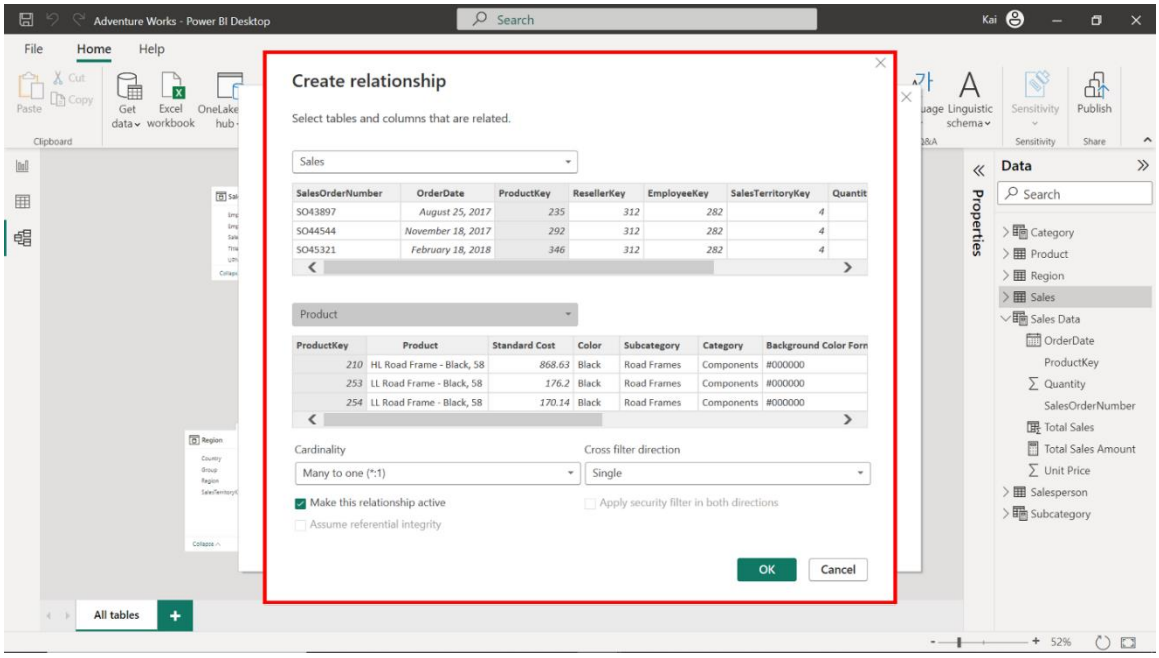
There are two ways relationships can be created in Microsoft Power BI: automatically, or manually. Let's explore these methods further.

Autodetect relationships

- When you load two to more tables using Power Query, Power BI desktop autodetects the relationships between the loaded tables based on a common column. Cardinality and cross-filter direction are automatically set.
- Sometimes, autodetected relationships are not accurate. In these instances, you need to create the relationship manually. You can also disable the **autodetect relationships** function in Power BI desktop.

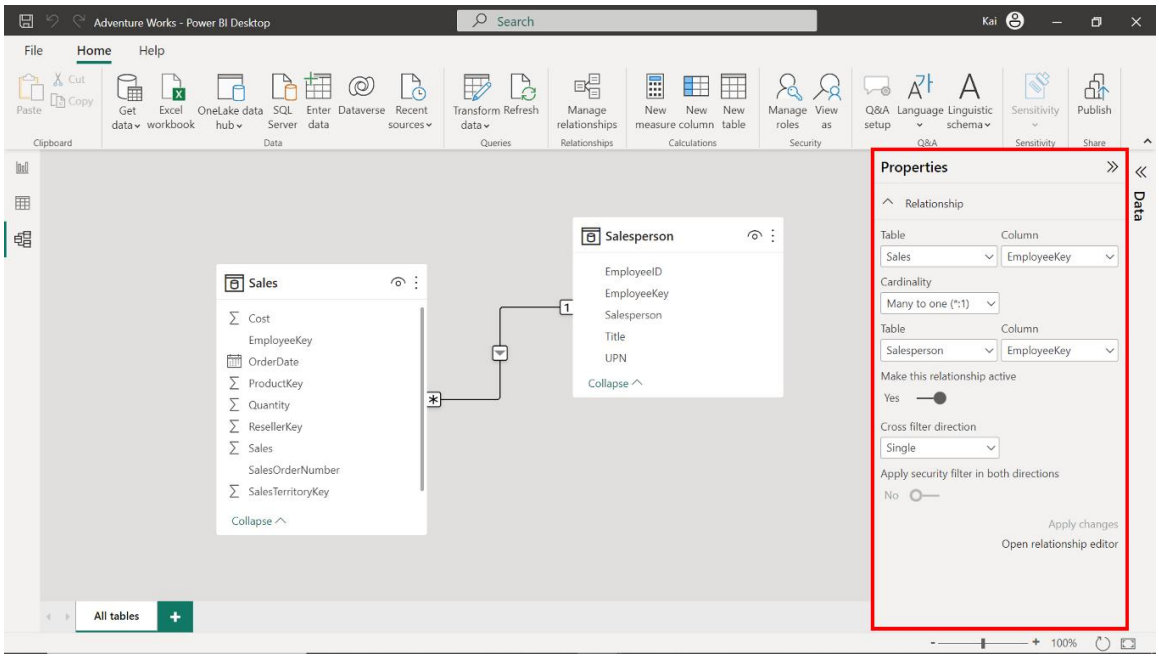
Manually create relationships

- You can build relationships manually using the **Model view** in Power BI desktop. Access the **Home** tab, then select **Manage Relationships**. This action brings up the **Create relationship** dialog box. You can use this dialog box to create and configure relationships between the loaded tables of a dataset. An example of this is shown in the following diagram:



Editing relationships

- You can edit relationships in Power BI using the **Properties** pane under the **Model view** (as shown in the diagram below) or the **Relationship Editor** dialog box.
- This dialog box can be opened in multiple ways. You can open it by selecting the **relationship line** between the two tables. or you can select **Manage Relationships** from the **Model view** of Power BI desktop.

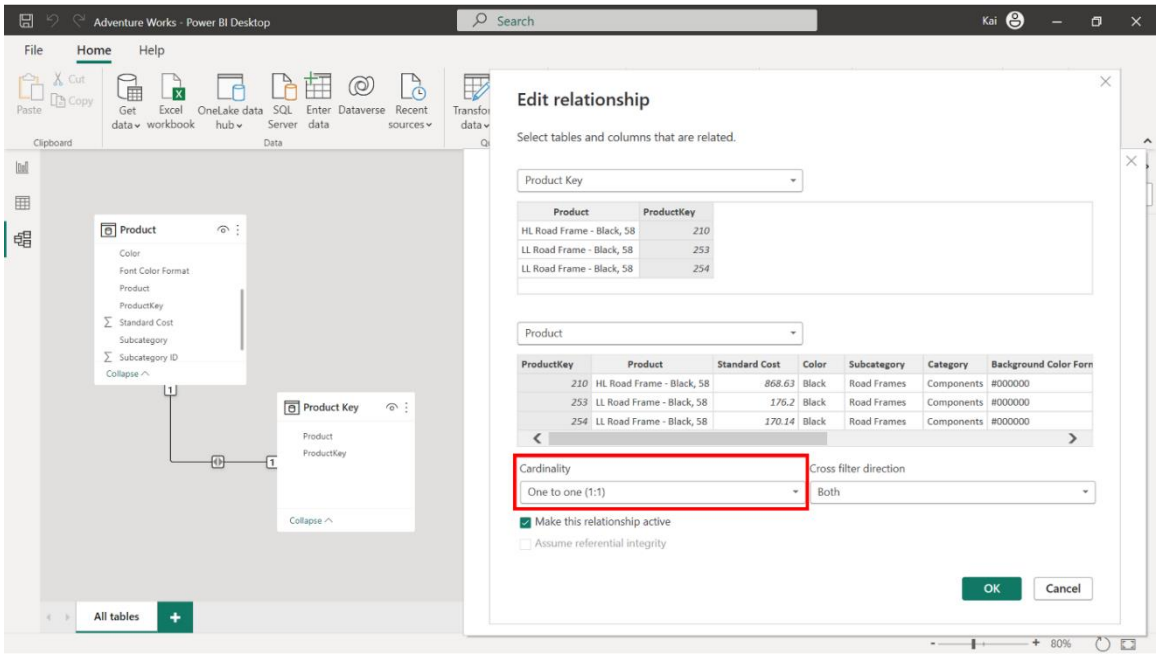


Model relationships

One-to-one Relationship

For example, Adventure Works has two dimension tables: **Product** and **Product Category**. Each table has a **SKU (Stock keeping unit)** column. All fields in these columns contain unique values.

A one-to-one relationship exists between these two tables based on the **SKU** column, because it is common to both. This means that when **SKU** filters the **Product Category** table, the **Products** table is filtered for products associated with the SKU.

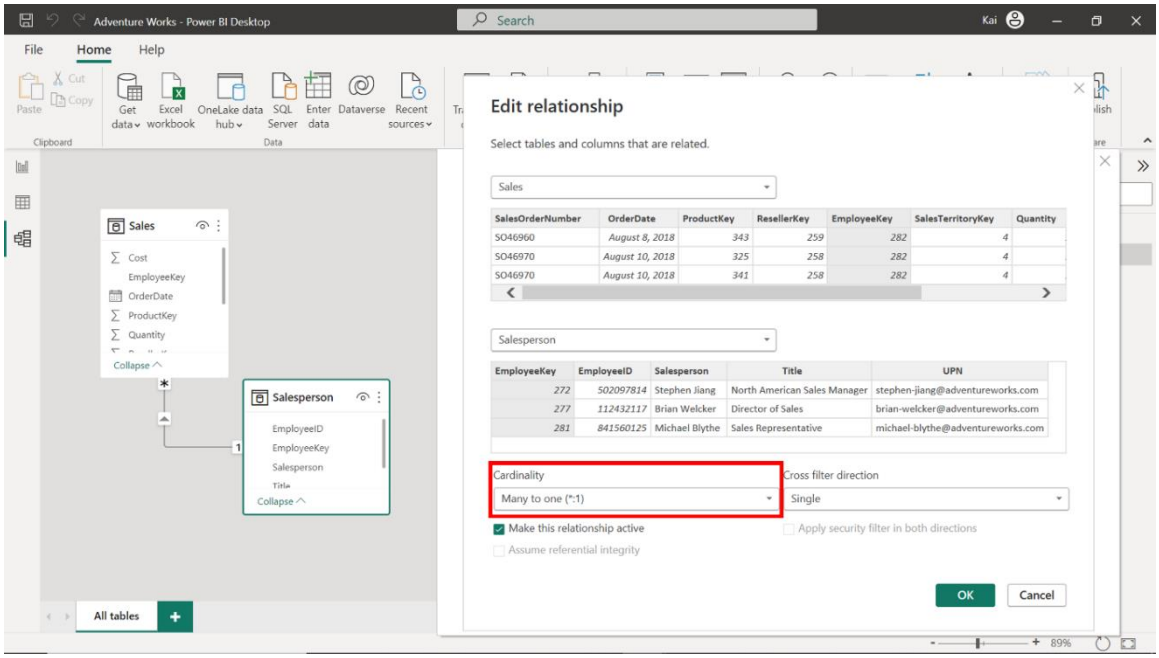


One-to-many relationship

- A one-to-many relationship exists where each value in the column of one table (**Table A**) can be associated with multiple values in the column of another table (**Table B**).
- This is the most common cardinality type and the default relationship in Power BI.
- In most data models, a one-to-many relationship describes the directionality between the Fact and the dimension table.

For example, in Adventure Works, the **Sales** table (the Fact table) is associated with the **Salesperson** table (the dimension table). Both tables have an **EmployeeKey** column, which establishes the relationship between the tables. In the **Salesperson** table, the **EmployeeKey** column contains a unique value in each row, as each salesperson only exists once. Each salesperson can have multiple sales, so their **EmployeeKey** may be repeated in multiple rows of the **Sales** table.

This is illustrated in the diagram below, where each salesperson can be associated with multiple sales.

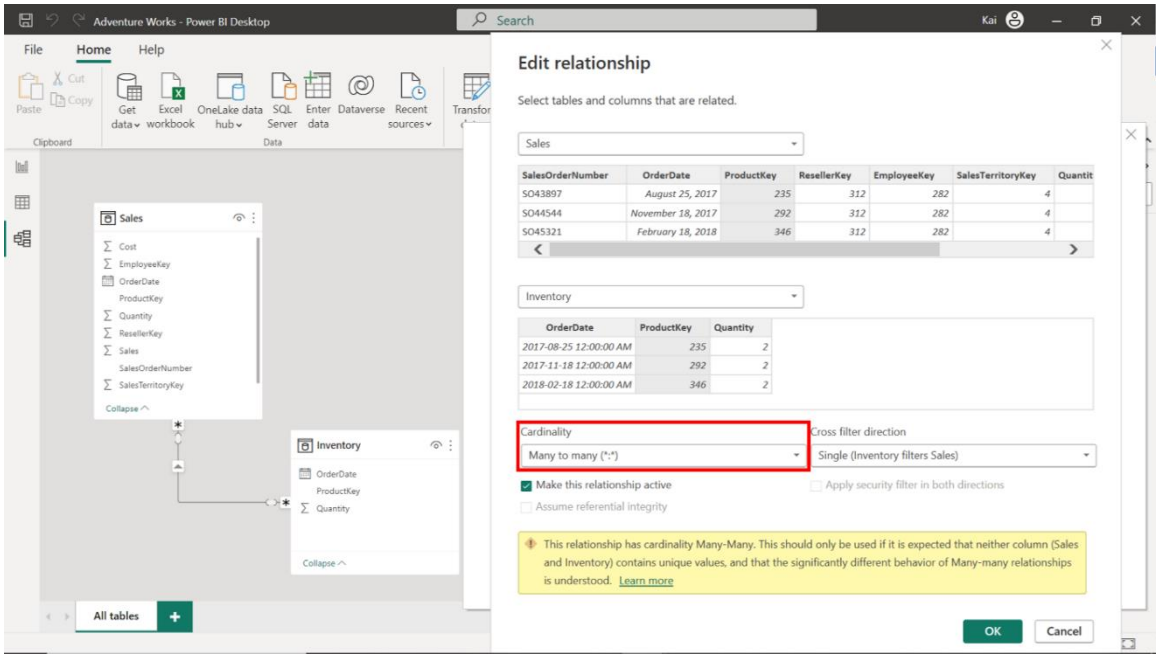


Many-to-many relationship

- A many-to-many relationship exists when multiple values in the column of one table can be associated with multiple values in the column of a related table. This relationship **does not require unique values** for either table.
- The disadvantage of a many-to-many relationship is that it introduces **ambiguity** in data analysis. So, it's only recommended to use it in certain specific scenarios.
- Typically, this relationship relates to two Fact tables or two dimension tables.

Let's take the example of Adventure Works' **Sales** and **Inventory** Fact tables. Both tables have a **ProductKey** column connected through a many-to-many relationship. This indicates that multiple sales can exist for a specific product key, and the product key can exist in multiple inventories in the warehouse.

In the data model, this means that for a given value of **ProductKey**, there can be multiple rows in the **Sales** table and multiple rows in the **Inventory** table. This must be a many-to-many relationship because no **ProductKey** field has unique values.



Cross-filter direction

- If your company needs to calculate which members of its sales team have sold the most product types and should be awarded a bonus. However, the data required to generate this insight is spread across multiple tables with **fixed cross-filter directions**.
- You can help your company analyze this data by **changing the cross-filter directions** of its tables.

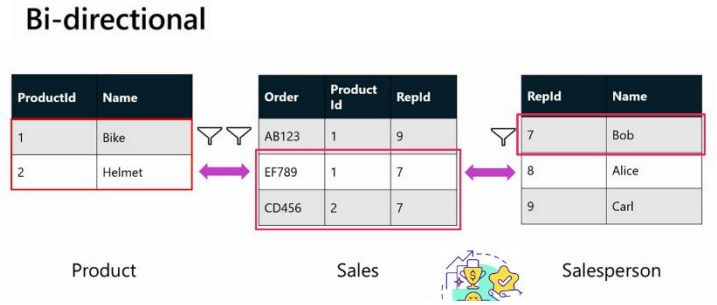
what data analysts mean by cross-filter direction?

- In Power Bi, cross-filter direction refers to the **pathway** or the **direction** through which **filtering** happens between two tables in a data model.
- It dictates how data from one table **influences** the data in another table.
- This enables relational analysis without resorting to complex queries or manual data consolidation.
- Power BI relationships are **directional in nature**. Unlike other database management systems, the direction significantly impacts how filtering operates.
- Having a clear understanding of relationship direction is a crucial aspect of data modeling in Power BI.
- Let's look at how direction plays an important role.
- The Company dataset contains three tables **product**, **sales**, and **salesperson**.
- The product dimension table is connected to the sales fact table using a one-to-many relationship based on the product ID column common to both tables.
- And a one-to-many relationship also connects the salesperson dimension table to the sales fact table based on their common rep ID columns.
- There are two types of cross-filter direction:

1 Sigle cross-filter direction

- The first is single cross-filter direction. This is the default setting in Power BI.
- The filter propagates from one table to another, but not vice versa. A good example of single cross-filter direction is the scenario you just explored.
- product and salesperson dimension tables are connected to the company's sales fact table via a one-to-many relationship.
- Each arrow points in a single direction, indicating that the relationship's direction is single.
- This means that sales data can be filtered by both product and salesperson. So when the product table is filtered for product one, the sales table is automatically filtered for all sales of product one.

2 bi-directional Cross-filtering



- Bi-directional filtering is filtering **against** the direction of a relationship.
- Sometimes you'll need to do this to answer a particular question.
- For example, If your company requires a report on employee performance. The report must show the number of products sold by each salesperson. You can generate this report using **bi-directional cross-filtering**.
- To generate the required results, you must filter from the **sales** fact table to the **salesperson** and **products** dimension tables.
- So, you need to change the direction of the filter to both.
-
- Let's look at the process steps for this action. You can apply a filter in the **salesperson** table for a specific sales team member. This filters the sales table for all sales by that person. The filter **propagates** to the **product** table as the direction is bi-directional. We have now determined how many unique products the salesperson has sold.
- However, there are a few important points to note when using bi-directional filtering:
    - Bi-directional cross-filter relationships can **negatively impact performance**.

    - And configuring a bi-directional relationship can also result in **ambiguous filter propagation** paths.
- You can **disable** filter propagation within a relationship in Power BI using the **cross-filter DAX function**. This setting can be particularly useful in certain advanced scenarios where you must isolate data for independent analysis.

Question

Cross-filter direction refers to the direction in which filtering occurs between two tables in a data model.

A. True
B. False

Knowledge Check

Question 1

In the context of Power BI, which of the following descriptions best outlines the main purpose of a Fact table?

A. A Fact table is primarily used for storing detailed, transactional business data.
B. A Fact table is primarily used for storing descriptive attributes of business dimensions.
C. A Fact table is primarily used for storing measured, quantitative data about a business process.

Question 2

Which of the following statements are true regarding cardinality and cross-filter direction in Power BI? Select all that apply:

A. Cardinality and cross-filter direction are two key elements of model relationships in Power BI.
B. Setting a cross-filter direction to **Both** allows filters to be applied from either direction in a relationship.
C. Cardinality defines the number of unique values in one column compared to another.

Question 3

True or False: In Power BI, you can create a many-to-many relationship between tables.

A. True
B. False

Question 4

In data analysis, _____ refers to the level of detail or summarization of your data.

A. Data granularity
B. Data cardinality
C. Cross-filter direction

Question 5

What is the role of dimension tables in Power BI?

A. They store measured, quantitative data about a business process.
B. They store the descriptive attributes of a business process.
C. They store transactional data related to a business process.

Chapter 3: Star and Snowflake Schema

Setting up a Star schema in Power BI

let's review the steps for setting opera star schema in Power BI.

1. The first step is to disable auto detect. Power BI also detects relationships when you load multiple tables. You need to disable the function so you can set your own relationships.
2. The next step is to load your fact and dimension tables into Power BI. Select the required tables from your Excel spreadsheet or other relevant location and load them into the application.
3. Once you've loaded the tables, you must create relationships between them. You can join tables by dragging relationships between key columns or from the manage relationships section of Power BI Desktop.
4. Finally, you need to set cardinality and cross filter direction. You must set cardinality to determine how your database tables relate, and you need to set the cross filter direction to determine the pathway through which filtering occurs between your tables.

Question

True or False: The autodetect function must be disabled before loading multiple tables. This is to prevent Power BI from automatically creating relationships between tables.

A. True
B. False

Exercise 3: Configuring a Star schema

- Use file **AdventureWorksDataset3.xlsx**.

Scenario

- Adventure Works wants to analyze its Sales data to generate insights into its business.
- It needs to create a data model using a Star schema.
- You can help the company to build this schema using the datasets within the workbook **AdventureWorksData**.

Steps:

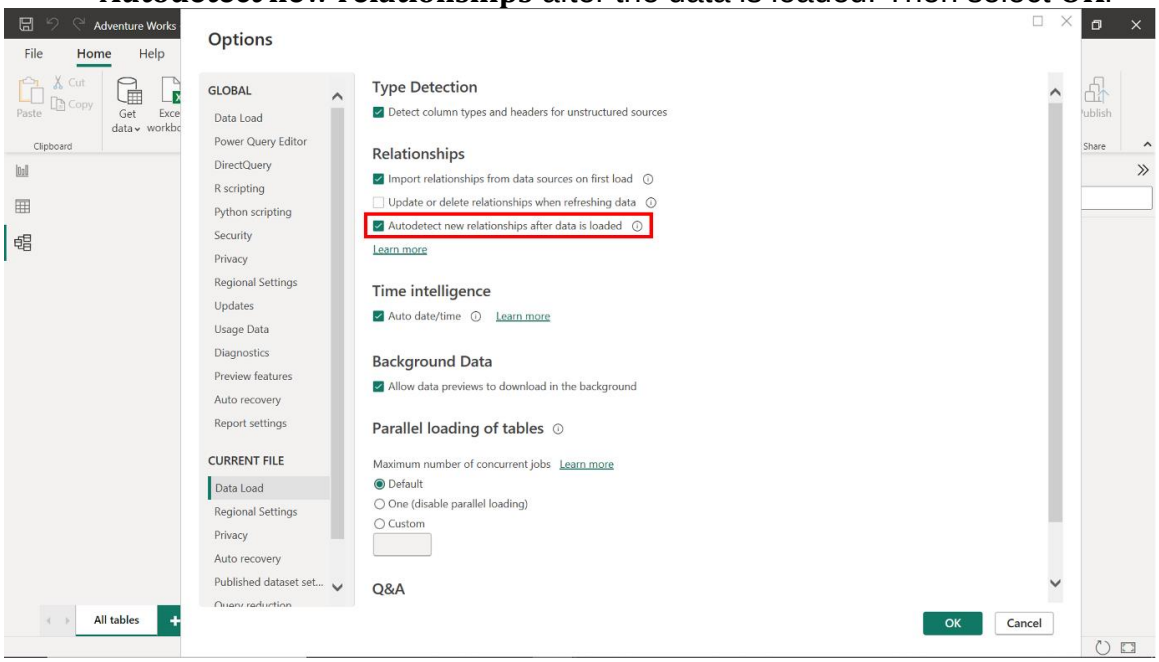
Step 1: Explore the Excel File.
Step 2: Disable autodetect relationships in Power BI.
Step 3: Load the data from the Excel workbook.
Step 4: Configure a Star Schema.

Step 1: Explore the Excel File

1. Explore the file in Exercise 3 Folder.
2. the Excel workbook **AdventureWorksData.xlsx**. The workbook contains four tables of data: **Sales**, **Products**, **Region**, and **Salesperson**.

Step 2: Disable autodetect relationships in Power BI

1. To disable autodetect functionality, select **File**, then **Options** and **Settings**, and select **Options**. This opens the **Options** dialog box.
2. On the left bar of the dialog box, select **Data Load** and then deselect **Autodetect new relationships** after the data is loaded. Then select **OK**.



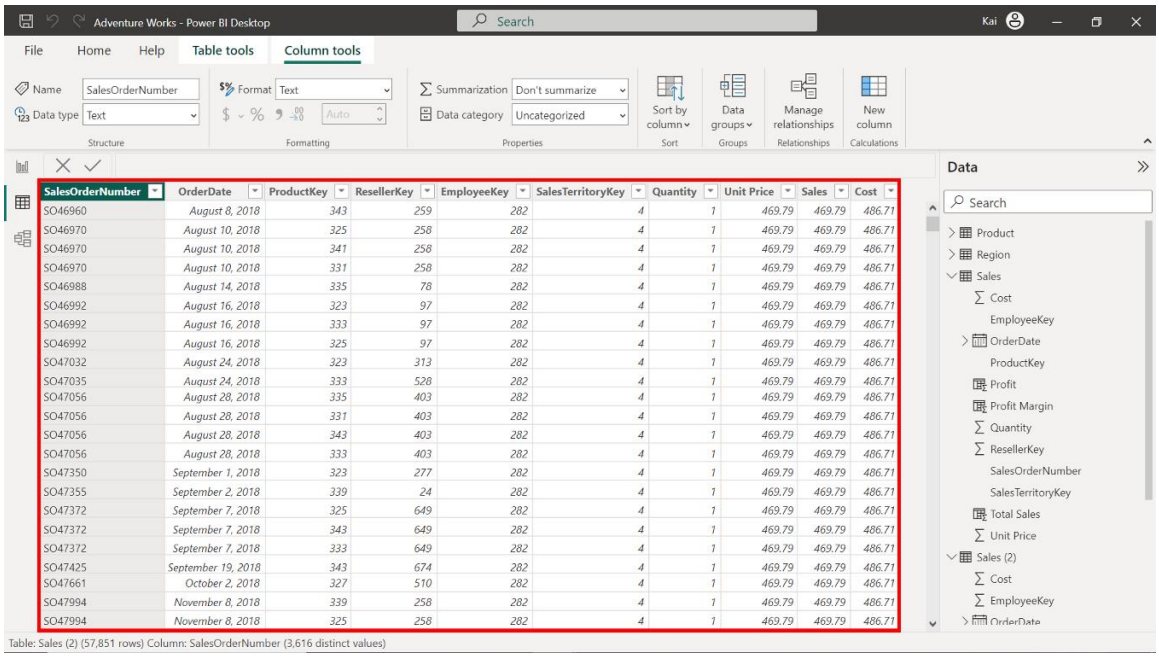
Step 3: Load the data from the Excel workbook

1. Load the data from the Excel sheet into Power BI. Ensure you load all tables in the workbook.
2. Open a preview of the table in the **Preview** pane.

Tip: You can import data using the **Get Data** drop-down menu.

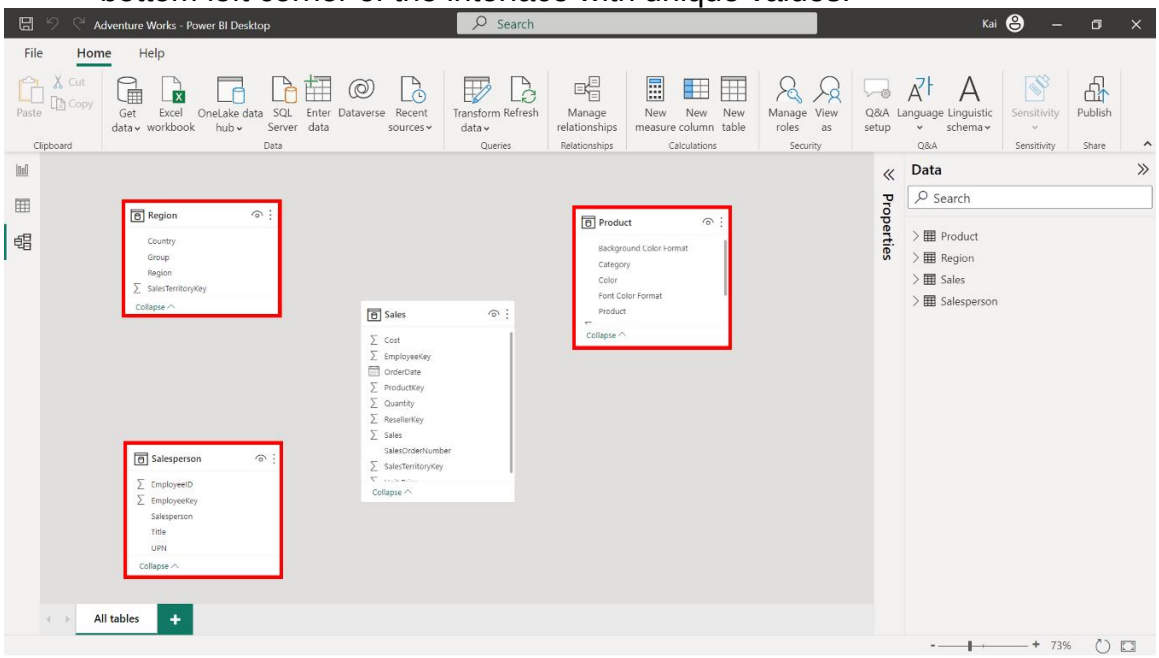
Step 4: Configure a Star Schema

1. In Table View explore the tables and identify the Key Column.
2. A unique identifier is usually an ID column or key column within the data table. Once you select a **column**, Power BI displays **the total number of rows** at the bottom left corner of the interface with **unique values**. For the **ID** column, the number of rows and **unique** values should be **the same**.
3. In the Adventure Works dataset, the **Sales** table is the fact table that records transactional details.



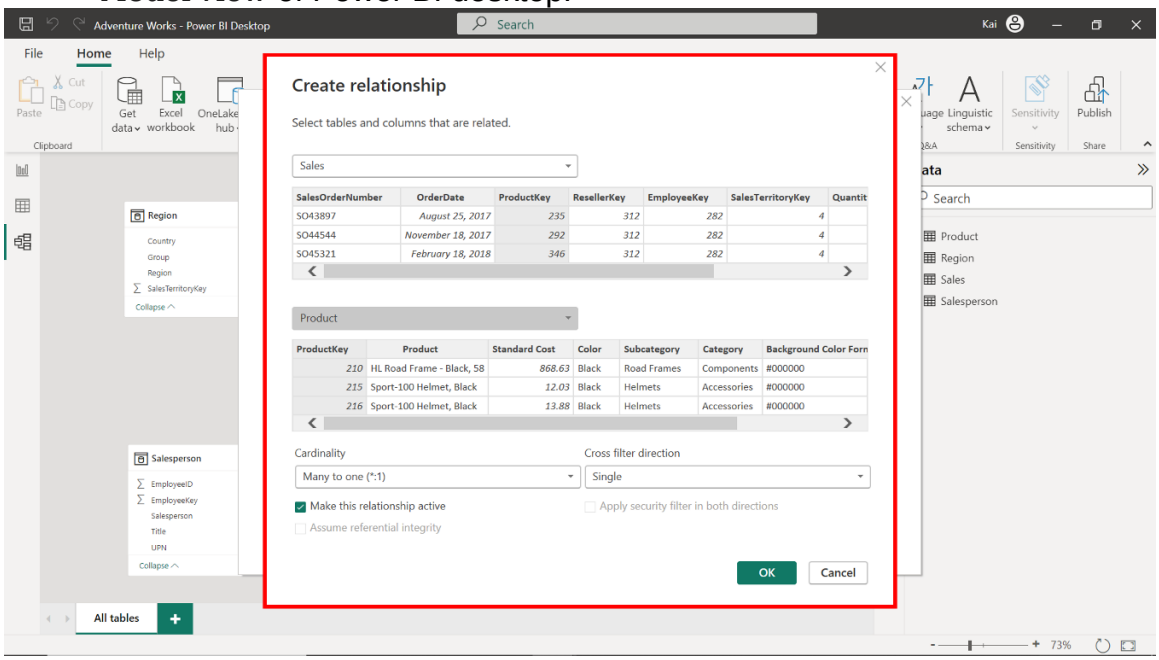
SalesOrderNumber	OrderDate	ProductKey	ResellerKey	EmployeeKey	SalesTerritoryKey	Quantity	Unit Price	Sales	Cost
SO46960	August 8, 2018	343	259	282	4	1	469.79	469.79	486.71
SO46970	August 10, 2018	325	258	282	4	1	469.79	469.79	486.71
SO46970	August 10, 2018	341	258	282	4	1	469.79	469.79	486.71
SO46970	August 10, 2018	331	258	282	4	1	469.79	469.79	486.71
SO46988	August 14, 2018	335	78	282	4	1	469.79	469.79	486.71
SO46992	August 16, 2018	323	97	282	4	1	469.79	469.79	486.71
SO46992	August 16, 2018	333	97	282	4	1	469.79	469.79	486.71
SO46992	August 16, 2018	325	97	282	4	1	469.79	469.79	486.71
SO47032	August 24, 2018	323	313	282	4	1	469.79	469.79	486.71
SO47035	August 24, 2018	333	528	282	4	1	469.79	469.79	486.71
SO47056	August 28, 2018	335	403	282	4	1	469.79	469.79	486.71
SO47056	August 28, 2018	331	403	282	4	1	469.79	469.79	486.71
SO47056	August 28, 2018	343	403	282	4	1	469.79	469.79	486.71
SO47056	August 28, 2018	333	403	282	4	1	469.79	469.79	486.71
SO47350	September 1, 2018	323	277	282	4	1	469.79	469.79	486.71
SO47355	September 2, 2018	339	24	282	4	1	469.79	469.79	486.71
SO47372	September 7, 2018	325	649	282	4	1	469.79	469.79	486.71
SO47372	September 7, 2018	343	649	282	4	1	469.79	469.79	486.71
SO47372	September 7, 2018	333	649	282	4	1	469.79	469.79	486.71
SO47425	September 19, 2018	343	674	282	4	1	469.79	469.79	486.71
SO47661	October 2, 2018	327	510	282	4	1	469.79	469.79	486.71
SO47994	November 8, 2018	339	258	282	4	1	469.79	469.79	486.71
SO47994	November 8, 2018	325	258	282	4	1	469.79	469.79	486.71

4. The **Products**, **Region**, and **Salesperson** tables are the dimension tables. To determine the unique identifier, check the total number of rows at the bottom left corner of the interface with unique values.

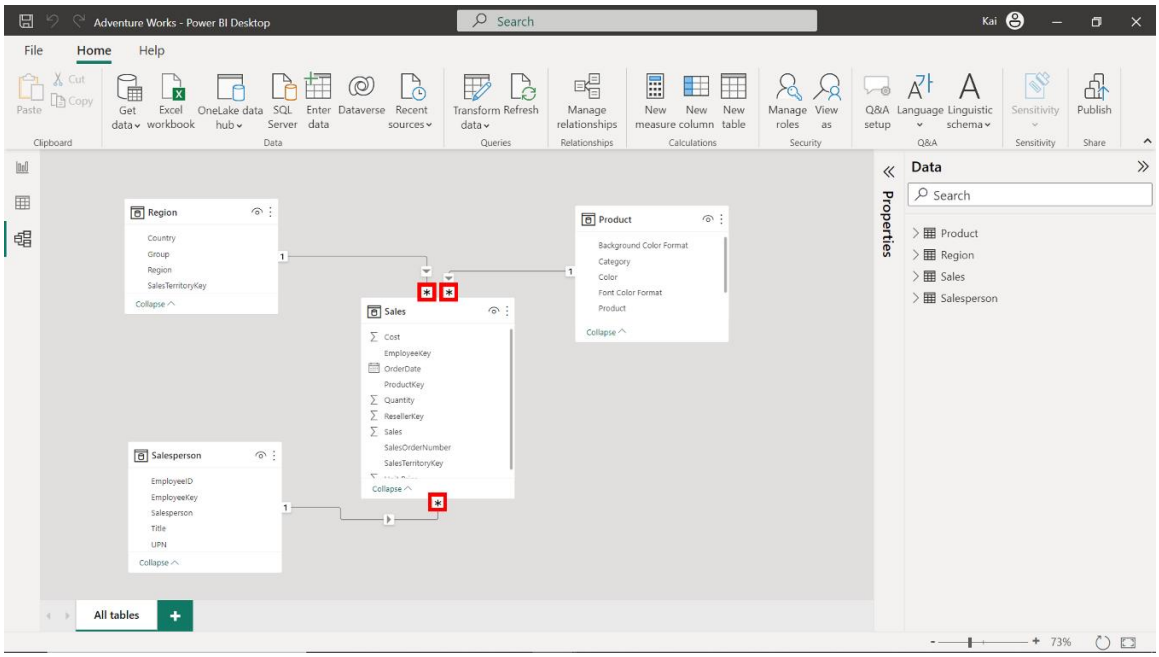


5. To create the schema and the model relationships between the fact table and the dimension tables, select **Model view** on the left sidebar of Power BI desktop.
6. You must establish relationships between the fact and dimension tables in the **Model view**. Select and drag the foreign key fields from the fact table to their corresponding primary key fields in the dimension tables.

7. Connect **Sales.ProductKey** to **Products.ProductKey**, then **Sales.SalesTerritoryKey** to **Region.SalesTerritoryKey**, and finally, **Sales.EmployeeKey** to **Salesperson.EmployeeKey**.
8. Alternatively, you can build relationships using the **Create Relationship** dialog box. Access the dialog box by selecting **Manage Relationship** in the **Model view** of Power BI desktop.



9. Once the relationships are established, double-check to ensure each type is a many-to-one relationship. Select the **connector line** between the tables to open the **Edit Relationship** dialog box. Here you can verify that the cardinality is set to **Many-to-One** and that the cross-filter direction is set to **Single**.



Save the Power BI project.

Exercise 3 Questions

Question 1

True or False: The **Sales** table was identified as a dimension table in the exercise.

A. True
B. False

Question 2

Which relationship type was configured between the Fact table and dimension tables in the exercise?

A. Many-to-many
B. Many-to-one
C. One-to-one

Question 3

True or False: The default cross-filter direction is set to **Single**, meaning that filters applied to the **Products** table will also apply to the **Sales** table, but not vice versa.

A. True
B. False

Setting up a Snowflake schema

- Data is not always structured in a way that provides quick insights. But by leveraging the Snowflake design schema, you can unlock your data's full potential.
- The Snowflake schema is a type of database schema design that optimizes data storage and retrieval by **normalizing** the data into multiple related tables.

- Unlike the star schema, which uses **denormalized** data with fewer tables, the Snowflake schema consists of a central fact table connected to one or more-dimension tables. The dimension tables are further connected to other related tables to create a hierarchy.
- For example, a **product** dimension table has a **product category** and a **product subcategory**. In a star schema, all three fields exist in one dimension table.
- However, in a Snowflake schema, you can split this single table into three different tables. And all these tables are related to one another via one to many relationships.
- Now, when you filter a specific product category, the filter is propagated through the tables from product category to subcategory, product and then sales.
-
- **The Snowflake schema offers many benefits:**
    - It simplifies dimension tables by splitting them into separate tables.
    - Simplifying dimension tables also improves data integrity because hierarchical relationships more accurately represent the data.
    - And splitting datasets into separate tables also helps to reduce data redundancy, because each attribute is only stored once.
    - It also enhances data analysis because a more efficient structure means more accurate insights.
    - And finally, a Snowflake schema leads to better management of data using hierarchies.

Exercise 4: Setting up Snowflake schema

- Use file **AdvetureWorksDataset4.xlsx**.

Scenario

- Adventure Works has created a Star schema to store its sales data.
- However, the Star schema poses several issues with data analysis and visualization.
- A solution to these issues is to change the schema to a Snowflake schema. A Snowflake schema will create a more complex structure, leading to better performance and easier maintenance. Adventure Works provides you with an Excel file called *Adventure Works Data*. The Excel file contains six tables. These tables are called **Sales**, **Product**, **Region**,**Salesperson,Category,** and **Subcategory**.

Steps

Step 1: Explore the Excel File.
Step 2: Disable autodetect relationships in Power BI.
Step 3: Load the data from the Excel workbook.
Step 4: Configure a Snowflake Schema.

Detailed Step

1. Before uploading the dataset, you first need to turn off Power BI's auto detect feature. This feature automatically creates relationships between the tables, but you need to do this manually.
2. To disable this feature, open Power Bi desktop, select **File ➔options and Settings**, and then **Options within settings**.

3. This opens the options dialog box. Select the **Data load** option to the left of the dialog box, then deselect Auto detect new relationships after data is loaded. Then select OK.
4. Now you can load the Adventure Works dataset.
5. From the home tab, select Get data, then select Excel workbook from the options in the drop down menu, navigate to the dataset and select Open. The navigator menu presents a list of available tables from the dataset, select the following tables, **Category**, **Product**, **Region**, **Sales**, **Salesperson**, and **Subcategory**. Then select Load.
6. The tables are loaded into Power Bi and presented in the model view.
7. You can now establish the relationships between the fact and dimension tables. You can do this by dragging the primary key from the dimension table to the foreign key in the fact table.
8. For example, drag the **product key** column from the products dimension table to the product key column in the select fact table.
9. You can then repeat this process for all related tables in the Snowflake schema.
10. Next, you must create **hierarchies** in the dimension tables to enable greater data analysis.
11. Create relationships between the **product** table and the **category** and **subcategory** tables based on the **category ID** and **subcategory ID** respectively via a one-to-many relationship. This creates a hierarchy of product dimensions.

Question

What is the primary advantage of using a Snowflake schema in Power BI over a Star schema?

A. The Snowflake schema reduces query complexity.
B. The Snowflake schema is more suitable for complex data structures.
C. The Snowflake schema requires less storage space.

Exercise 5: Changing Star schema into a Snowflake schema

Use file **AdventureWorksDataset5.xlsx**.

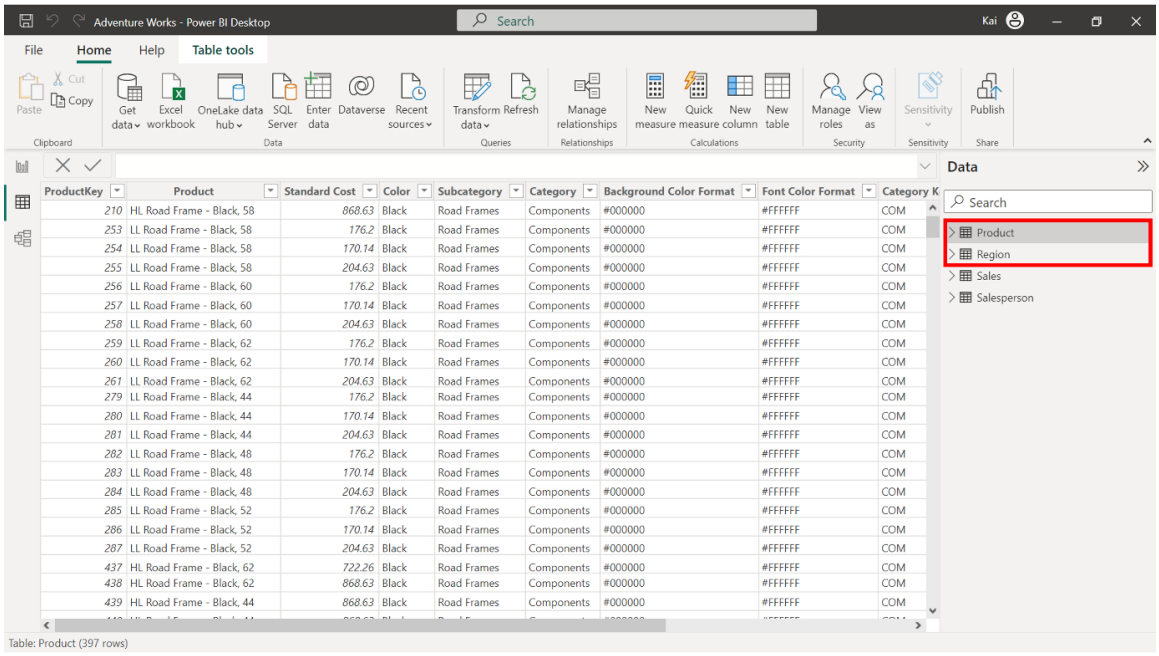
Scenario

- Adventure Works has created a Star schema to store its sales data.
- However, the Star schema poses several issues with data analysis and visualization.
- A solution to these issues is to change the schema to a Snowflake schema.
- A Snowflake schema will create a more complex structure, leading to better performance and easier maintenance.
- Adventure Works provides you with an Excel file called *Adventure Works Data*. The Excel file contains four tables. These tables are called **Sales**, **Product**, **Region**, and **Salesperson**.

Step 1: Create a Strar Scema like the one you have created in the previous exercise 3, *Configuring a Star schema*

Step 2: Identify the dimension tables in the star schema that can be normalized further into related tables.

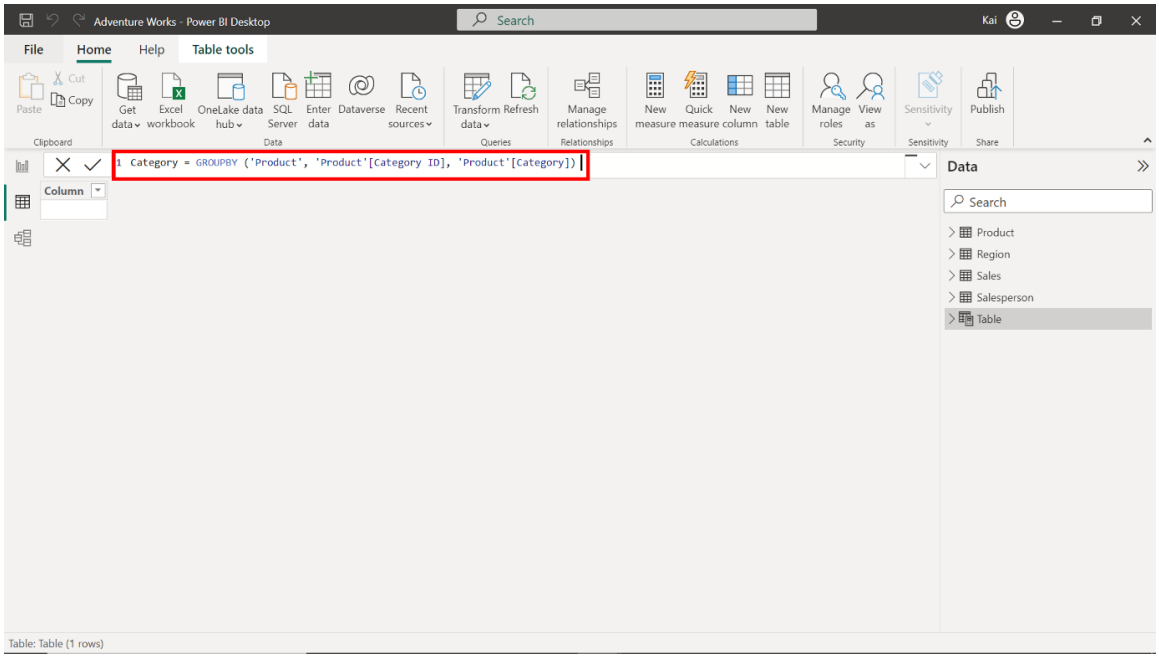
1. In the case of the Adventure Works star schema, two separate dimension tables can be normalized into look-up tables.
2. These are **Product** and **Region**.
3. You must normalize the **Product** table into **Category** and **Subcategory** tables to generate a Product hierarchy.



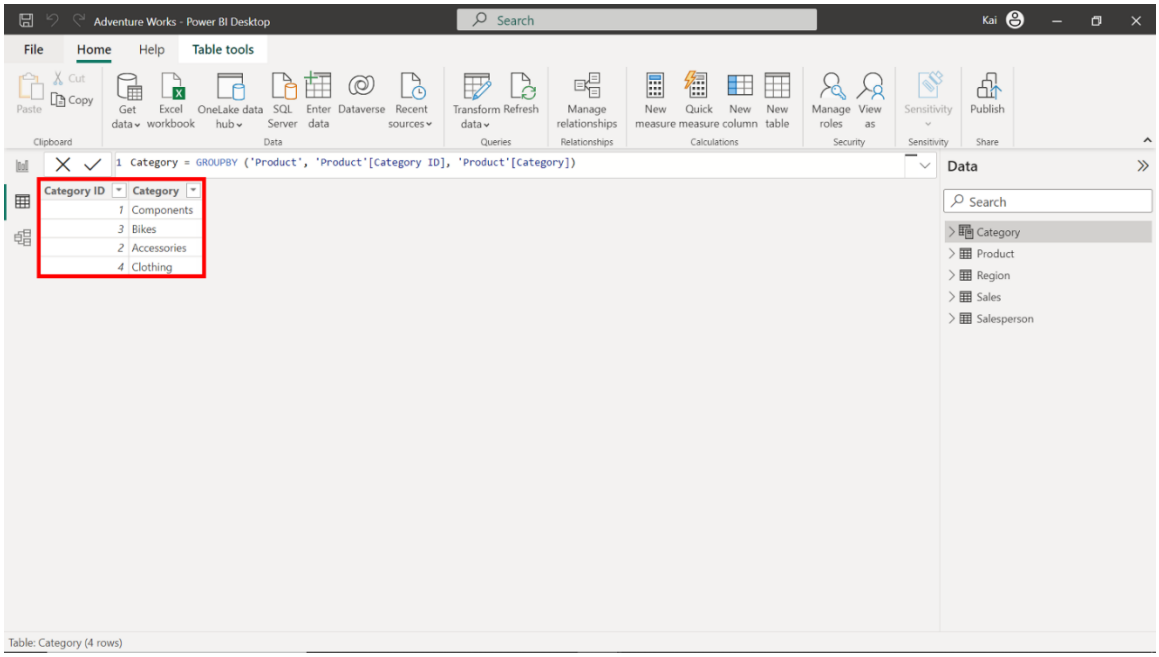
4. In the Power BI Data view, within the **Calculations** group, select **New Table**.
5. Copy and paste the following DAX codes in the formula bar to create a new **Category** table.

Tip: If you encounter an error with copy/paste, manually type the query. You'll explore DAX in more detail in a later module.

```
Category = GROUPBY ('Product', 'Product'[Category ID], 'Product'[
Category])
```



6. Once input, the DAX code generates a new table, as shown in the following image.

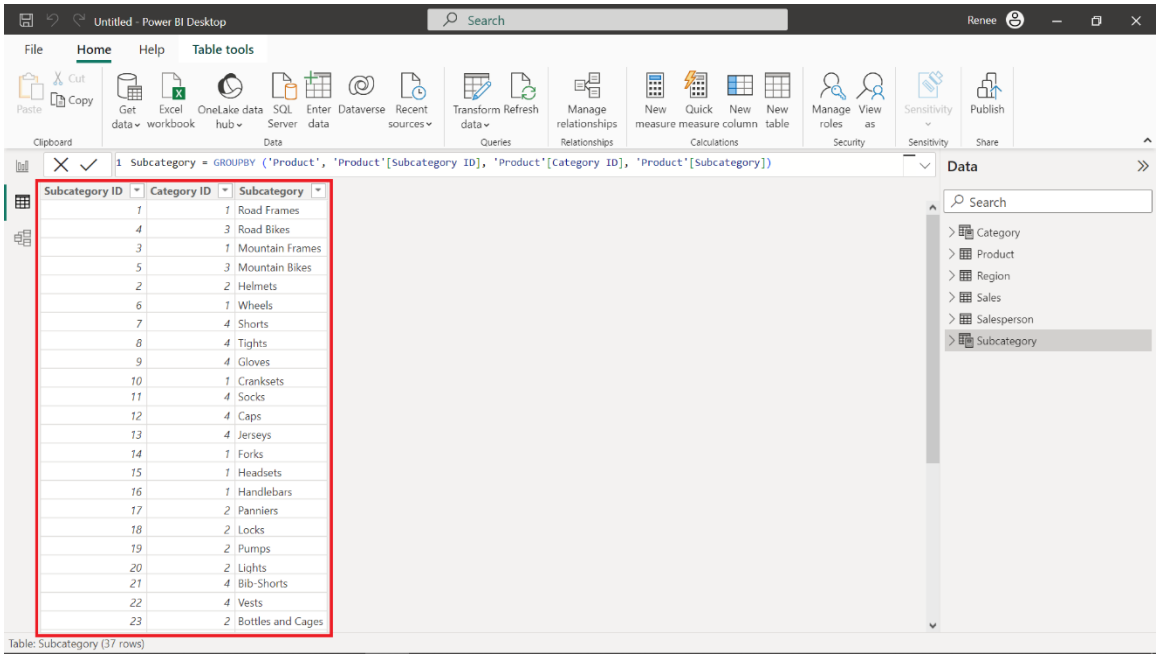


7. Repeat the same process to create a **Subcategory** table using the following DAX query.

Tip: If you encounter an error with copy/paste, manually type the query.

```
Subcategory = GROUPBY ('Product', 'Product'[Subcategory ID], 'Product'[Category ID], 'Product'[Subcategory])
```

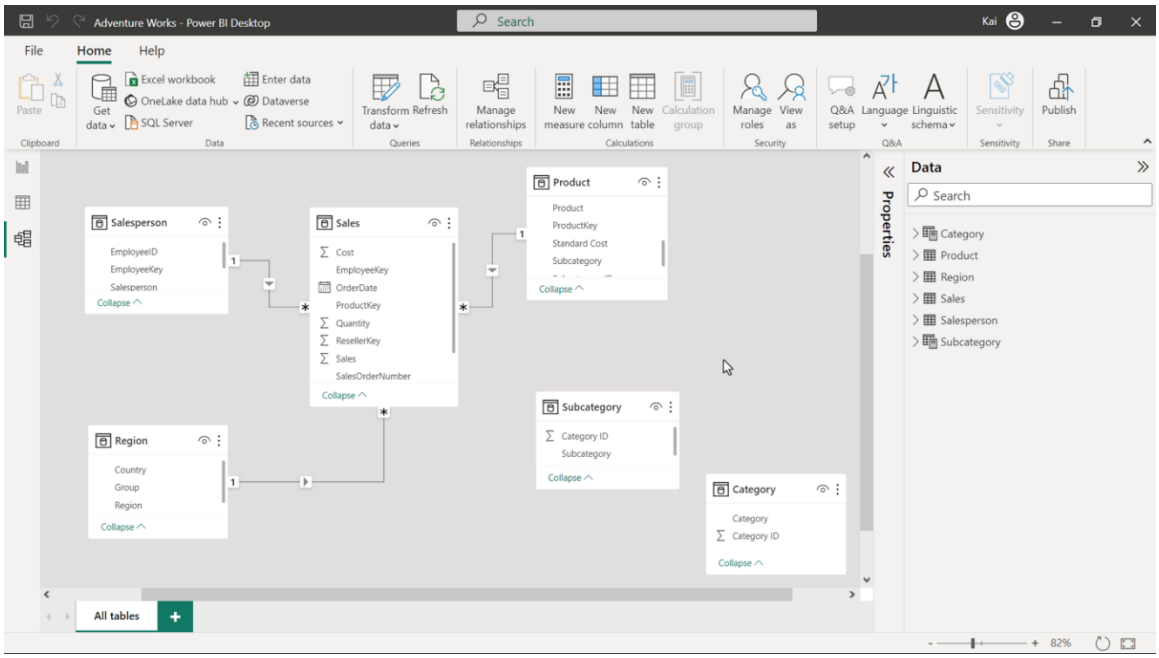
8. Once input, the DAX code generates a new table, as shown in the following image.



Subcategory ID	Category ID	Subcategory
1	1	Road Frames
4	3	Road Bikes
3	1	Mountain Frames
5	3	Mountain Bikes
2	2	Helmets
6	1	Wheels
7	4	Shorts
8	4	Tights
9	4	Gloves
10	1	Cranksets
11	4	Socks
12	4	Caps
13	4	Jerseys
14	1	Forks
15	1	Headsets
16	1	Handlebars
17	2	Panniers
18	2	Locks
19	2	Pumps
20	2	Lights
21	4	Bib-Shorts
22	4	Vests
23	2	Bottles and Cages

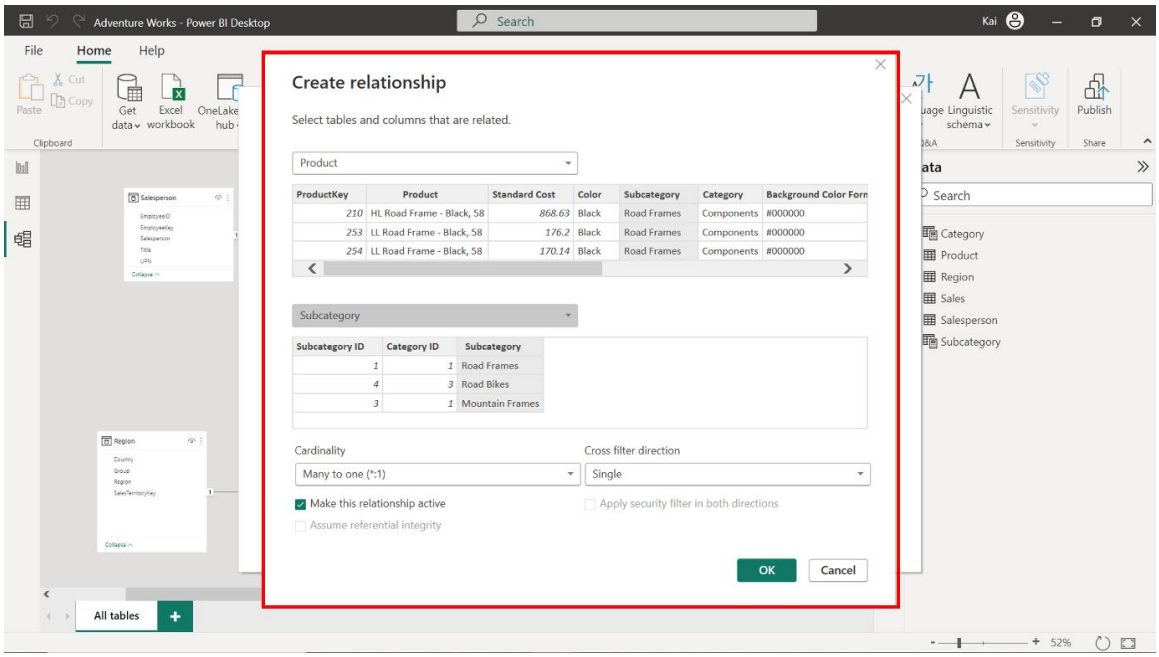
Step 3: Configure the Snowflake schema.

1. Once you finish creating new tables, Power BI attempts to autodetect and establish the relationships between these newly created tables and the already-existing tables. If relationships were automatically created, you need to remove these relationships. **Select** and **right-click** the **relationship connector**, then select **delete**.

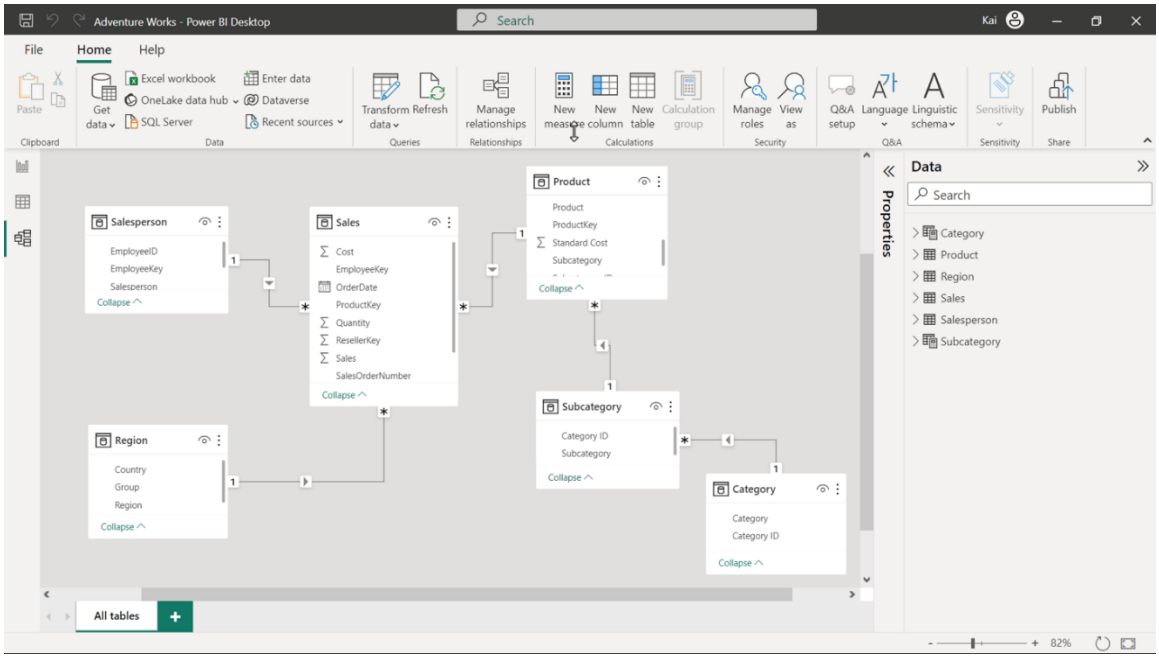


1. Create relationships between the **Product** table and **Subcategory** tables based on the **Subcategory ID**.
2. Create further relationships between the **Category** and **Subcategory** tables based on the **Category ID**.

3. You can create new relationships from the model view of Power BI desktop by selecting **Manage relationships** in the Home tab of the top Ribbon menu.



1. To configure these relationships, access the **Manage relationships** dialog box from the **Model view** of Power BI desktop.
2. Make sure the cardinality for each relationship is set to **Many-to-one** and that all cross-filter directions are set to **Single**.
3. These new relationships create a **Product** hierarchy.
4. Any filter applied to the **Category** table now propagates to the **Sales** fact table to compute the **Sales** figures based on each **Product** category.
5. This helps to analyze top-performing product categories and make strategic decisions.



Save the project.

Knowledge Check

Question 1

Which of the following statements is correct regarding a Star schema Fact table?

A. A Fact table stores an accumulation of business entities.
B. A Fact table must have a unique column
C. A Fact table stores an accumulation of business events.

Question 2

How are dimension tables structured in a Snowflake schema?

A. They are connected in a hierarchical structure with multiple levels.
B. They are normalized with a separate table for each attribute.
C. They are fully denormalized, with all attributes in a single table.

Question 3

What is the primary benefit of normalizing dimension tables in Power BI?

A. It simplifies data querying and reporting.
B. It reduces storage requirements.
C. It improves data quality and accuracy.

Question 4

Which of the following statements is true about relationships in Power BI?

A. Relationships can only be created between columns that contain the same data type.
B. Relationships can only be created between tables with the same number of rows.
C. Relationships can be created between tables that contain different types of data.

Question 5

True or False: A Star schema is more suitable for complex hierarchies and relationships.

A. True
B. False

Chapter 4: Introduction to DAX

- What if you're analyzing a data model and the data you need isn't in the original model?
- If it's not possible to derive the data from the original model, you can use DAX (data analysis expressions) to create custom calculations to generate the data.
- For example, if your company needs to identify its top-selling products and calculate its revenue but these insights are beyond the scope of the original data model. They can only be generated by calculating the existing data. So you must use DAX or data analysis expressions to complete this task.

What is DAX?

- DAX is a programming language used in:
  - Microsoft SQL Server Analysis Services,
  - Power Pivot in Excel, and
  - Power BI.
- It is a library of:
  - functions,
  - operators, and
  - constants used in formulas or expressions to create additional information about the data are not present in the original data model.
- With DAX expressions, you can create **custom calculations** on data models to extract maximum information from your data to solve real-world problems.

Dax Fundamentals

- To master DAX, you need to understand its
  - syntax,
  - different data types,
  - the operators, and
  - how to refer to columns and tables using functions.

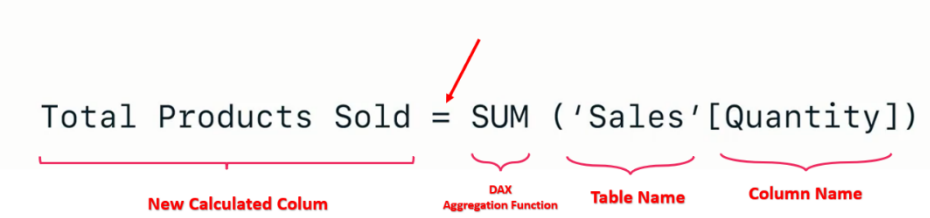
DAX Syntax

- DAX usually computes values over columns in a table, so you need to know how to reference a column in a table:
  - First, write the **name** of your new calculation.
  - Then add the **equals sign** operator.
  - Next, write the name of your **DAX function**,
  - then **parenthesis** that contain the logic of your formula.

- Write a **table name** enclosed in **single quotes**, followed by the **column name** enclosed in **square brackets**.
- You can omit the table name if the reference column is on the **same table**. Let's demonstrate this using an example from Adventure Works.

Example of DAX Syntax

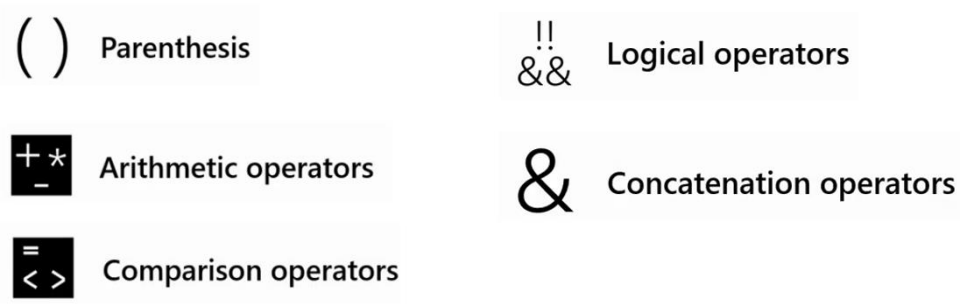
- If sales table doesn't include any data that denotes the total number of products sold. The company could generate this data using DAX.



- In the DAX expression:
  - **sales** is the table name followed by the column name **quantity** to be referenced and
  - **SUM** is the **DAX aggregation function**.
  - **Total product sold** is the name of the new **calculated column** that holds the results of the calculation.
- When executed, this DAX formula adds a **new column** to the existing table that contains the required data.

Operators

- DAX formulas rely on operators.
- There are many different types of operators.
- They can be used to perform arithmetic calculations, compare values, work with strings or test conditions.
- Some commonly used operators in DAX include:



-
  - ( ) parenthesis for grouping arguments,
  - **Arithmetic Operators** for performing basic functions like:
    - (+ , - , *, /) addition and subtraction multiplication subdivision, and
    - comparison operators for comparing values.
  - DAX also uses **logical operators** to return **true false** values and
  - **concatenation operators** to combine two or more values into a single string.

Example of Using Operators

- You can use operators in a DAX formula to calculate its **total revenue**.

```
Total Revenue =
SUMX (Sales[Quantity] * Sales[Unit Price])
```

- In this example:
  - the multiplication operator multiplies the **unit price** by the **quantity** to compute the **total revenue**.
  - The **parenthesis ( )** group the arguments of the expression and
  - the **SUMX** DAX function adds the argument values to calculate the total revenue.

DAX functions

- DAX functions perform various calculations, manipulate data, and create custom expressions.

DAX functions Example

- As you discovered in an earlier example, to calculate their total revenue, you can perform this calculation using the **SUMX** DAX function.
- For now, you just need to be familiar with the concept of functions. You'll explore functions in more detail later in this lesson.

- It's also important to understand that DAX is not just about formulas and functions. It involves understanding:
  - **the data model**,
  - **the relationships between tables**, and
  - **the contexts in which calculations are made**.
- For instance, understanding how the tables relate to one another in your data model is crucial for creating meaningful calculations.
- There are several important aspects of a relationship that will help you to understand DAX:
  - Tables connected via a relationship are not the same. They are either on one or many sides of the relationship.
  - Columns used to build the relationship are the keys of the relationship. The column on one side of the relationship needs to have unique values.
  - Tables relationships can be either single or bidirectional, the direction of the relationship determines the direction of automatic filtering.

Recommendation to Learn DAX

- Remember, mastering DAX requires practice:
  - Start with simple formulas and
  - gradually incorporate more complex functions and operators and
  - ensure you understand your data model and the relationships within it.
  - As your comfort with DAX grows, so will your ability to turn data into meaningful insights.

- Eventually, you'll be able to unleash the full potential of your data using DAX and gain valuable insights for decision-making.

Question

Identify the components of a data analysis expression. Select all that apply.

A. Column and table references
B. Data types
C. Syntax
D. Operators

Row context and filter context

- DAX as a useful language for generating business insights using formulas. However, data analysts need to understand that DAX generates insights from data based on the **context of that data** , **row** and **filter** context impact data evaluation.
- For example if you need to answer business specific questions like what the total sales for each product are, and what are the top selling items by category. It can generate these insights using DAX. DAX formulas answer these questions by evaluating the relevant data according to its row and filter context.
- DAX computes formulas within a context. The evaluation context of a DAX formula is the surrounding area of the cell in which DAX evaluates and computes the formula.
- The surrounding area is determined by the set of rules and filters to be evaluated in a DAX expression. It determines which subset of data is used to perform calculations. DAX expressions adapt or refer to the context for evaluating dynamic and contexts aware results

Row Context

- Row context refers to the tables current row being evaluated within a calculation.
- When a DAX expression is evaluated for a specific row, it considers the values of the columns in that row as the context of the calculation.
- This allows for calculations to be performed at row level and it's especially useful for iterating to rows within a table.
- For instance, if you create a formula for a calculated column, the row context for your formula includes the values from all the columns in the current row.
- For Example if The table contains sales data for multiple products over one month stored within the following columns: **days**, **product**, **category**, **quantity**, and **price**.
- If you want to to create a **total sales** calculated column that shows the total sales data for each product in the table, you can use a DAX formula to multiply the **quantity** data in the quantity column by the **price** data in the price column for each item.
- The formula iterates through the relevant **quantity** and **price** column values at the **row level** and returns the results and the total sales calculated column.
- In other words, the formula calculates the new values via **row context**.

Filter context.

- As the name suggests, filter context refers to the filter constraints applied to the data before it's evaluated by the DAX expression.
- In the previous example, a different result was produced in each cell because the same DAX expression was evaluated against different subsets of data.
- However, with filter context, you can determine which rows or **subsets** should be included or excluded from the calculation.
- For Example if your company must calculate the total sales for all items in **Category X**. The company can create a DAX formula containing filters that targets all sales recorded against Category X.
- Once the formula is executed, it iterates through each row and retrieves only the data with the value of X. Row and filter context also interact with each other to produce results.
- When a DAX expression is evaluated, **it first considers the filter context**, then the **row context takes effect**.
- For example, The company can use the **filter** context to narrow it sales data to the selected **region**.
- The row context then iterates each row in the filtered results and calculates the sales totals.
- As you've just discovered, a filter applied on a table column affects all table rows, filtering rows that satisfy that filter.
- If you apply two or more filters to columns in the same table, they are executed under a logical end condition. This means only the rows satisfying all the filters are processed by the DAX expression in that filter context.

- Be careful when applying a filter in a large data model with multiple tables, a filter context automatically propagates through the relationships between the tables in the data model based on the selected cross filter direction of the relationships.
- In this example, this means that when data is filtered in the sales order table, then data in the related tables is also filtered.
- You can disconnect the tables to prevent propagation.
- A rule context, on the other hand, doesn't automatically propagate through a data model relationships. If you have a row context in a table, you can iterate the roles of a table on the many side of a one-to-many or many-to-many relationship using the related table function. You can also access the rows of the parent table using the related function of DAX.
- Understanding the context of DAX expressions at the row unfiltered level is important as you continue to build data models for reporting and visualization.
- Contexts defects how DAX interprets and analyze your data. So always consider the context when creating and executing your DAX formulas.

Question

True or False: Filter context can only be applied to a calculation via writing a DAX expression.

A. True
B. False

Creating calculated columns

- You might often encounter tables that don't have the data you need.
- You can generate this data by combining existing columns to create a new calculated column.
- If your company is analyzing the data in its **Sales** table and realizes there's no data for the profit margins on its product categories in the original data source. **Calculated columns** are the perfect solution to this problem.
- You can add data on its profit margins using DAX expressions to create new calculated columns within the original data source.

What is a Calculated Column?

- A calculated column is **a new column added to an existing data table in Power Bi**.
- Data analysts can use calculated columns to **derive new data from existing columns and add it to the data model**.
- Once added, these columns can be used in any part of a report or visual just like any other column.

Difference between Traditional and Calculated Columns

- Traditional columns are filled with data **imported** from a data source.
- A calculated column is created by defining a **DAX expression**.

Creating Calculated Column

- You can create a DAX expression that calculates the data from two or more columns.
- The result of this calculation is then added to the table as the newly calculated column.
- Write the name of your calculated column and an equals operator. Then write the names of the tables to be referenced in single quotation marks, and their respective column names in square brackets. Include a relevant arithmetic operator depending on the operation required.
- For example, you can create a Total **Sales** calculated column by multiplying the quantity and unit price columns in its Sales table.

Exercise 6: Creating Calculated Columns

Scenario

- Your company wants to calculate its **profit margin** from its **Sales** data in its Sales table by creating calculated columns.
- However, the table is missing this data
- You need to add it using DAX and Calculated Columns

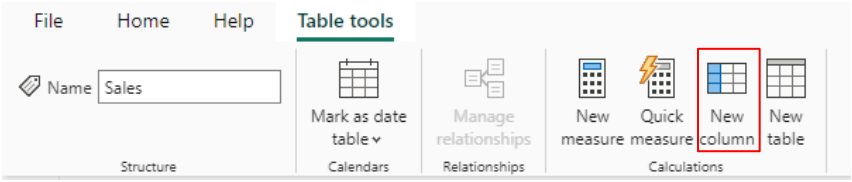
Steps

1. Use file **Sales6.xlsx**

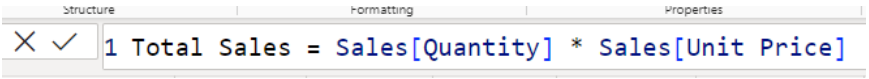
2. Create a new Power BI file and import the Sales table.
3. The workbook contains one table called **Sales**. The table tracks Company recent sales data.
4. Load the table Sales into your model.
5. Your company wants to calculate its **profit margin** from its **Sales** data in its Sales table by creating calculated columns.

Total Sales Column

6. Access power BI's **table view** to view the **Sales** table, your company needs to calculate its profit margin. But to do this, it must first calculate its **Total Sales** for the quantity of each item sold.
7. However, the table is missing this data. You can add this data to the table by creating a new Total Sales column. You just need to **multiply** the **quantity** and **unit price** columns.
8. Select the Sales table from the data pane on the right-hand side of Power Bi desktop.



9. In the **Table Tools** tab, select the **new column** from the **Calculations** group.
10. This opens the **DAX formula bar**.
11. Write **DAX code** in the Formula bar that multiplies the quantity column by

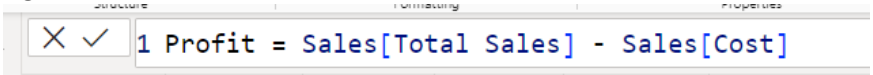


the unit price column and adds the result as a new **Total Sales** column.

12. Press **Enter** to execute the code.
13. A new Total Sales calculated column appears under the Sales table in the Data view on the right-hand side of the Power Bi interface.
14. You can use this new column in any report or visualization like any other table column.

Profit Column

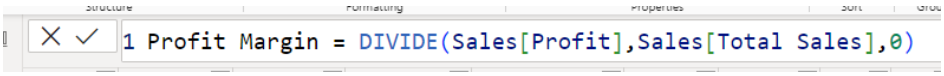
15. Now that you've identified the Total Sales data, you can create a **Profit** column to determine how much profit has been made on each item.
16. Write another DAX formula that subtracts the **cost** from the **Total sales** and generates the data as a new **Profit** column.



17. Press **Enter** to execute the formula. The new Profit Calculated column is added to the sales table.

Profit Margin

18. Now that you've identified the profits, you can create the **profit margin** column.
19. Select new column again. Then write another DAX formula in the formula bar that **divides** the **Profit** and **Total Sales** columns and generates the



result in a **Profit Margin** Calculated column.

20. Press **Enter** to execute the formula. The profit margin column is added to the data.

Format Columns

21. Finally, you need to format the calculated columns, select the **profit** column and format it as **currency**.
22. Then format the **profit margin** column as a **percentage**.

Question

In Microsoft Power BI, what is the function of a calculated column?

A. It visualizes data in various formats such as charts and graphs.
B. It performs calculations on data using a DAX formula.
C. It sorts and filters data based on user-defined criteria.

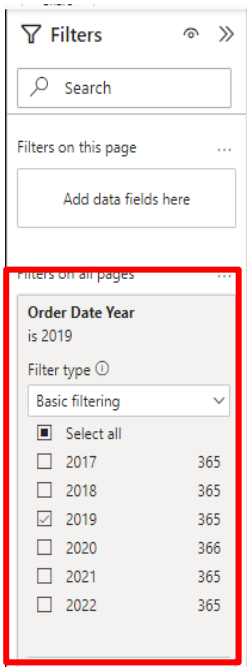
Exercise 7 : Filtering Context

Scenario

- We want to review some of the filtering context like:
  - Filters on All Pages.
  - Filters on this Page.
  - Filters on a Visual.

Steps:

1. Use file **7 Filtering Context Start.pbix**



Step 1: Review the Data Model

2. Go to **Model View**.
3. Review the two tables **Sales** and **Products** and their **relationship**.

Step 2: Review the Calculated field

4. Go to **table view**.
5. Select **Sales** Table.
6. Select the calculated column **Total Sales**.
7. Let us explore deeper the DAX expression of the column.
8. Imagin a **Pointer** goes row by row and calculate each value in the column in the **Row Context**.
9. This happens more the 60,000 times for all rows one by one.

Create 3 tables in Report View

10. Go to **Report View**.
11. Create table with 3 columns: **Product**, **Year**, **Total Sales**

    (Expand Order Date Hierarchy to find the Year field)

12. Select the Table you have created and Copy (Ctl+C)
13. Paste the table again in the **same Page**.
14. Create **another page** and paste the table again.
15. We now have 3 copies of the same visual.

Step 4: Filters on All Pages

16. Go bake to Page 1
17. From table **Order Date hierarchy** Drag **Year** filed to the **Filters on All Pages** in the **Filters** Pane.
18. In **Filter Type** select **Basic Filtering** and Chose **2019.**
19. Notice that the two visuals in the page has accepted 2019 and the table were filtered to one row showing totals of **2019**.
20. Also, if you go to the other page it will see that filter were applied.
21. That is because we have applied the filter to **All Pages**.
22. Click the (**X**) sign in the filter to clear the filter from All Pages.

Step 5: Filters on This Page

23. Repeat steps 16,17,18 but this time in **Filters on this Page**.
24. This time it will be applied to Page 1 only and Page 2 is not affected.
25. Click the (**X**) sign in the filter to clear the filter.

Step 6: Filters on a Visual

26. Go to Page 1 and select the 2nd Visual.

27. You will find now that the Filter Pan has a new Option **Filters on this Viual.**
28. Repeat steps 13,14, 15 but this time in **Filters on this Visual.**
29. Only that visual is filtered.
30. Click the (**X**) sign in the filter to clear the filter.

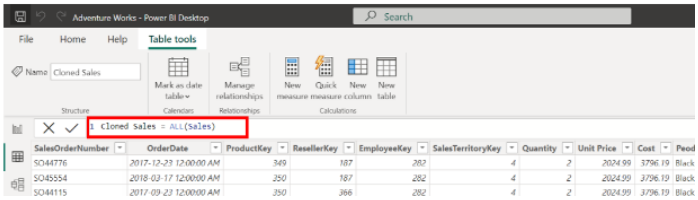
Notice also that Power BI filter the table in each row **in context of each year** (**2017,2018,2019,2020**) each row is calculated filtering only record form each.

Introduction to calculated tables

- You might not always be able to answer business questions using an existing data model. It could lack the required data or be too complex.
- In these instances, you can use calculated and cloned tables to enhance your datasets and improve your analysis.
- If your company needs answers to business-specific questions about its sales and marketing, but its current data model isn't up to the task. However, by creating calculated tables, the company can compare and analyze its data to generate the required insights.

cloning a table

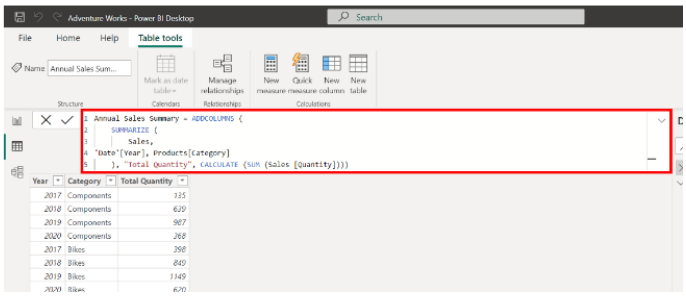
- Cloning a table can be extremely useful for manipulating or augmenting data without affecting the original table.
- This is especially true when working with tables that are refreshed periodically, and any changes you made to the original table might be overwritten.
- For example, if your company wants argument its sales table to generate insights, but it doesn't want to alter the original data.
- So, the company can create and work from a **cloned version** of the table while leaving the original intact.
- A table can be cloned using a simple DAX formula. Type the new table's name, an equals operator, and the original table name in parentheses. Add the word, all, to instruct Power BI to clone all data from the target table.
- This formula states that the clone table is equal to the original table.



- You can use this syntax to create a clone of sales table called **Sales Data**.
-

Create Calculated Table form other tables

- You can also use DAX to create a calculated table based on data from various sources.
- For example, your company might want to combine customer data from a database with sales data from an Excel spreadsheet to analyze the relationship between its sales and customers.
- The company can use DAX to merge these sources and enable its analysis.
- Calculated tables can also be used to normalize dimension tables.
- Your company can use DAX to split their **product** dimension table into **category** and **subcategory** tables. This creates a hierarchy that enables more efficient data exploration and reporting.
- Calculated tables are not simple duplicates. Instead, they are created based on **calculations**, **transformations**, or **aggregations** performed on existing data.
- For instance, you can create a calculated table showing each bicycle



model's total sales.

Real-World applications of calculated tables

There are many real-world applications of calculated tables. Here are some examples of how you can make use of calculated tables.

Profitability Analysis

- By creating a calculated table for profitability analysis, you can draw insights into gross margins, net profits, and profit margins. This helps organizations identify their most profitable products, categories, services, and customer segments.

Customer Segmentation

- Understanding customer behavior is crucial for marketing efforts. You can create a calculated table with DAX to facilitate customer segmentation based on transaction history. This helps businesses to tailor their marketing strategies for each customer segment.

Time Intelligence Analysis

- A date table is one of the most common tables that can be created using various methods in Power BI. One method of creating a date table is using the DAX function **CALENDER**. For example, a company can record several date columns

in their dataset, like order date or shipping date. The calculated table can then analyze the data using time intelligence built-in DAX functions like **TOTALYTD**, **year-to-date**, **month-to-date**, and so on.

Budgeting and Forecasting

- By defining a calculated table in DAX for budget allocation and integrating it with historical data, you can perform variance analysis and make data-driven forecasts for future periods.

Question

Which of the following statements regarding calculated tables in Power BI and DAX are true? Select all that apply.

A. Calculated tables can only be created from a single data source.
B. Calculated tables are created within Power BI using DAX expressions.
C. Calculated tables combine data from different sources to facilitate deeper data analysis.
D. Changes to the original data source will always be reflected in the cloned calculated tables.

Exercise 8: Adding a calculated table and column

Use file **AdventureWorksDataset8.xlsx**.

Scenario

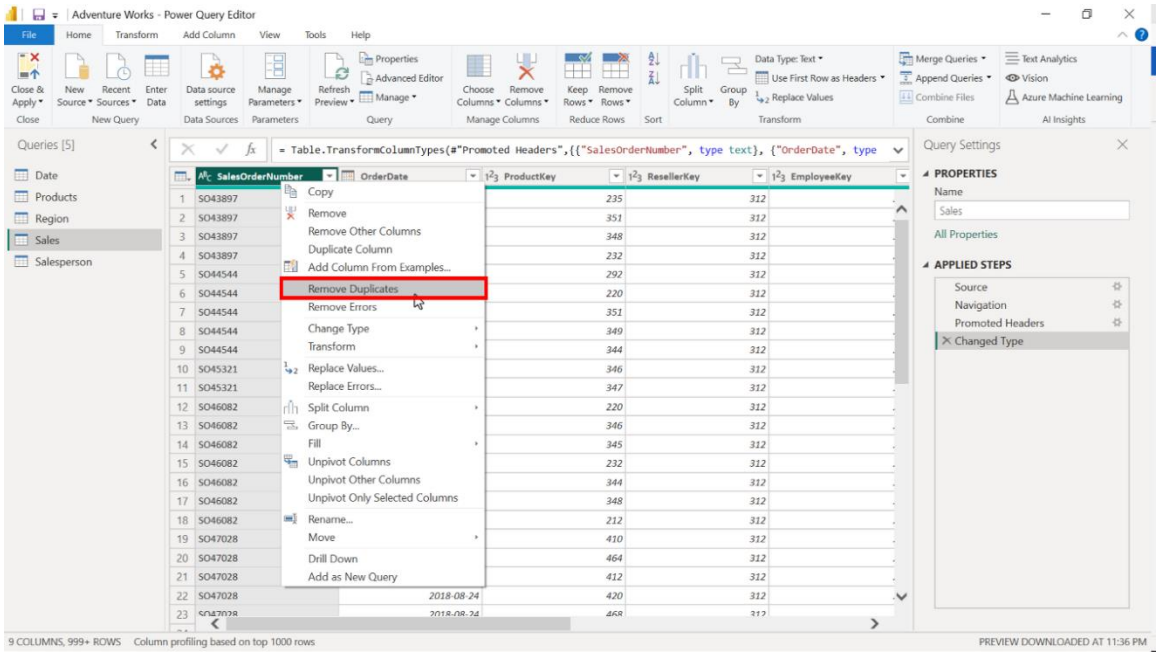
- Your company needs your help to analyze its sales data to generate insights into its sales performance. However, you must analyze the company's data without altering the original dataset. You must also create summary tables and normalize dimension tables for analysis.
- The company provides you with an Excel workbook called ***AdventureWorksData8.xlsx***. You must load it to Power BI to complete your assigned task.
- Be sure to evaluate the data quality and configure the model to ensure that Adventure Works can use it to make informed decisions.

Step 1: Connect to the Adventure Works Dataset.

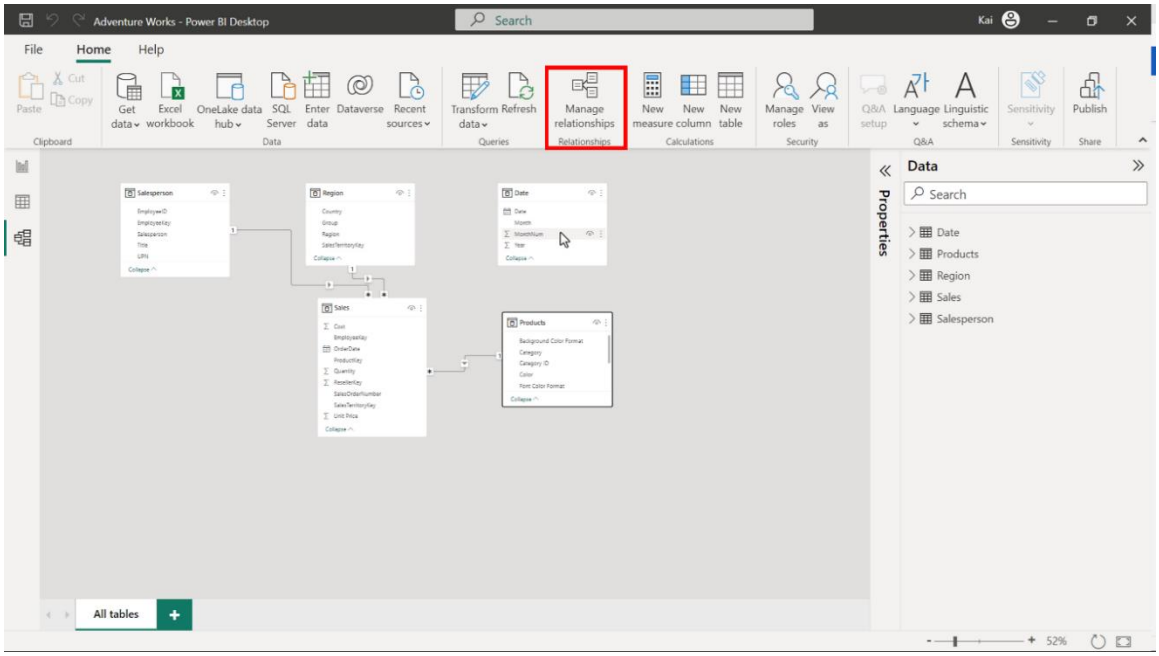
1. Launch Power BI desktop. To create a new project, select the **File** menu, then select **New**. Import the Adventure Works dataset that you have. In the **Home** tab, select the **Get Data** drop-down menu. Then select an appropriate data source. For the current exercise, select **Excel Workbook** and navigate to the Adventure Works dataset folder.
2. Once you select and load the data, Power BI opens a **Navigator** dialog box that lists all the tables available to load in the Excel file, along with the data preview on the right side of the Navigator. Select the **Sales**, **Product**, **Region**, **Date,** and **Salesperson** tables, then select **Load**.

Step 2: Remove all duplicate values and set the relationships

1. To eliminate all duplicate data, access the **Power Query editor**, right-click on the **SalesOrderNumber** columns, and select **Remove duplicates** from the drop-down menu.



1. To configure the model relationships, access the **Model view** of Power BI desktop and select **Manage relationships**. From here, you can edit cardinality and cross-filter direction between the tables.

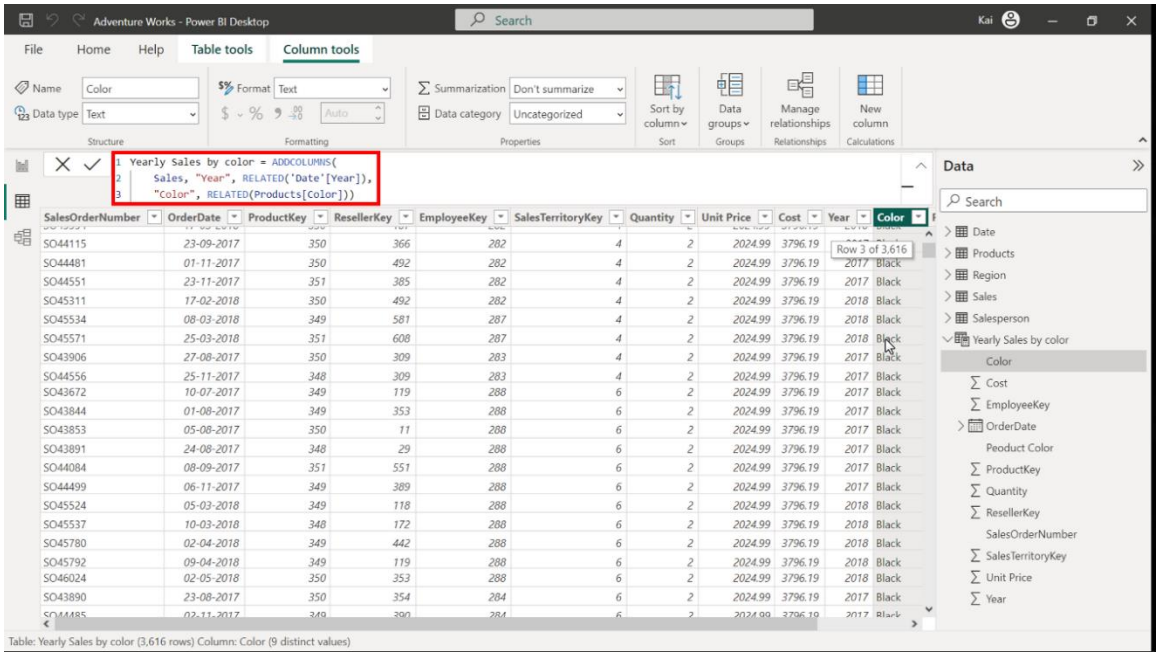


Step 3: Create a calculated table.

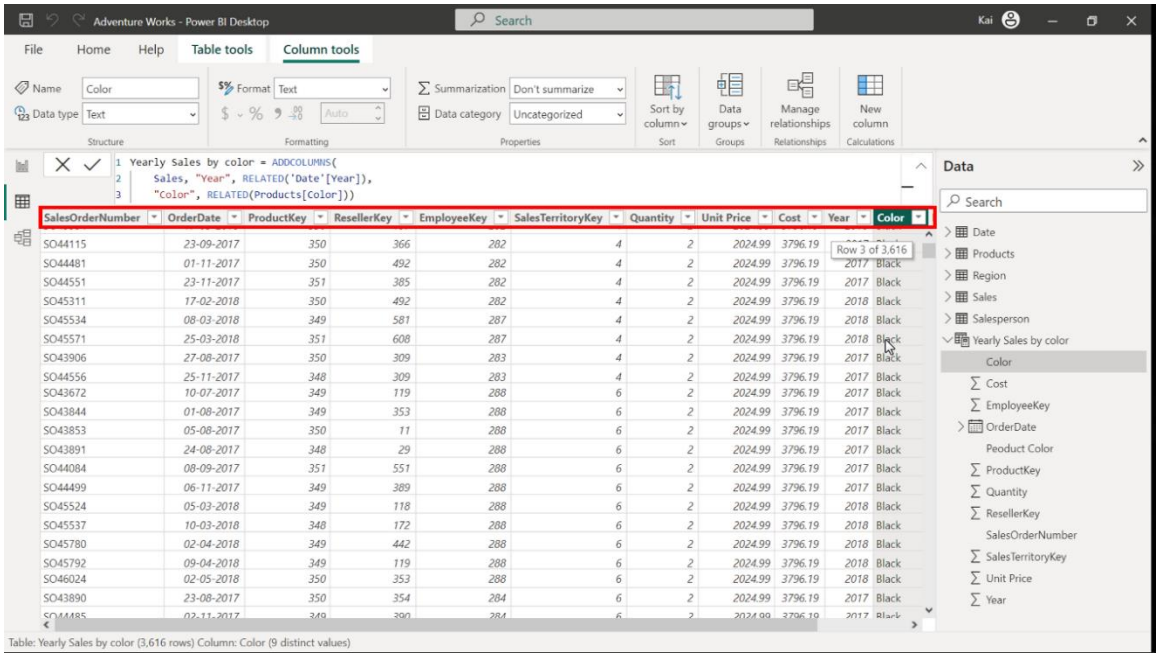
1. Access the **Model view** in the calculations group to create a new table. Select **New table**. Copy and paste the following DAX code into the formula bar:

```
Yearly Sales by color =

ADDCOLUMNS (

Sales,

"Year", RELATED ( 'Date'[Year]),

"Color", RELATED ( Products[Color]))
```

- **ADDCOLUMNS**: Adds calculated columns to the given table or table expression. In this instance, the **Sales** table is the main table to which you need to add two more columns, one from the **Date** table and one from the **Product** table.
- **Year** and **Color** in double quotes are the names of the new columns to be added in the new calculated table.
- **RELATED**: Returns a related value from another table. In this case, **Product color values** from the **Product table** and **Year** information from the **Date** table.



1. Note that the resulting table has 11 columns.

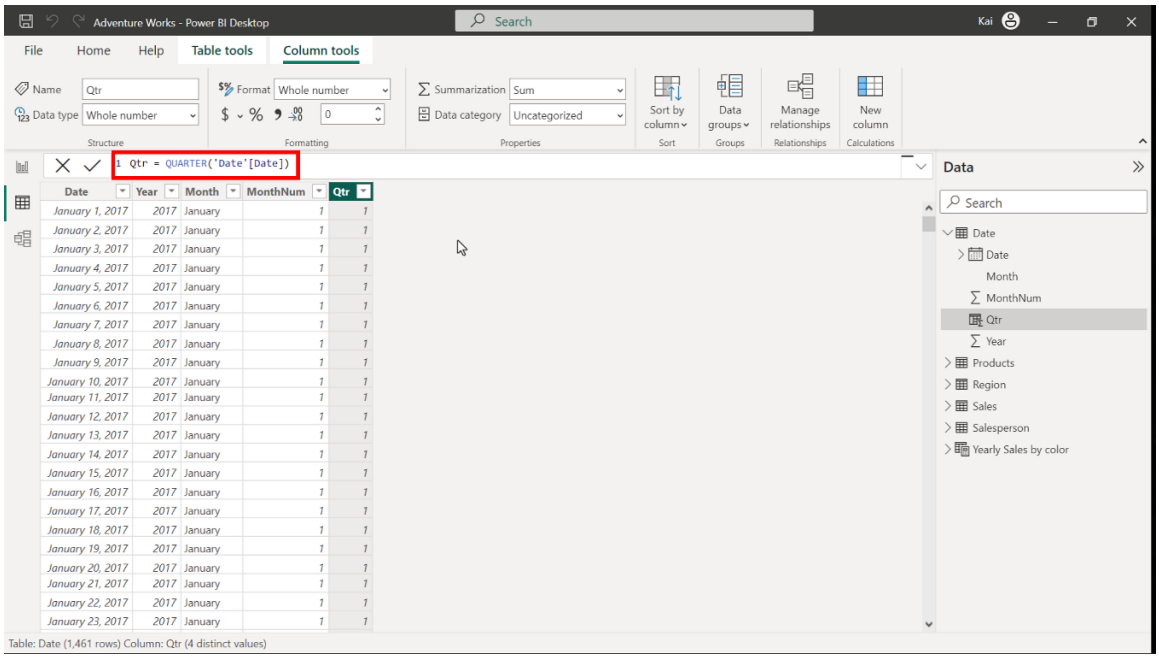


Step 4: Create calculated columns.

1. To create a new column, select the **Date** table from the **Data pane** on the right side of Power BI interface. Access **Model view** in the **Calculations group** and select **New column**. Copy and paste the following DAX code into the formula bar:

```
Qtr = QUARTER('Date'[Date])
```

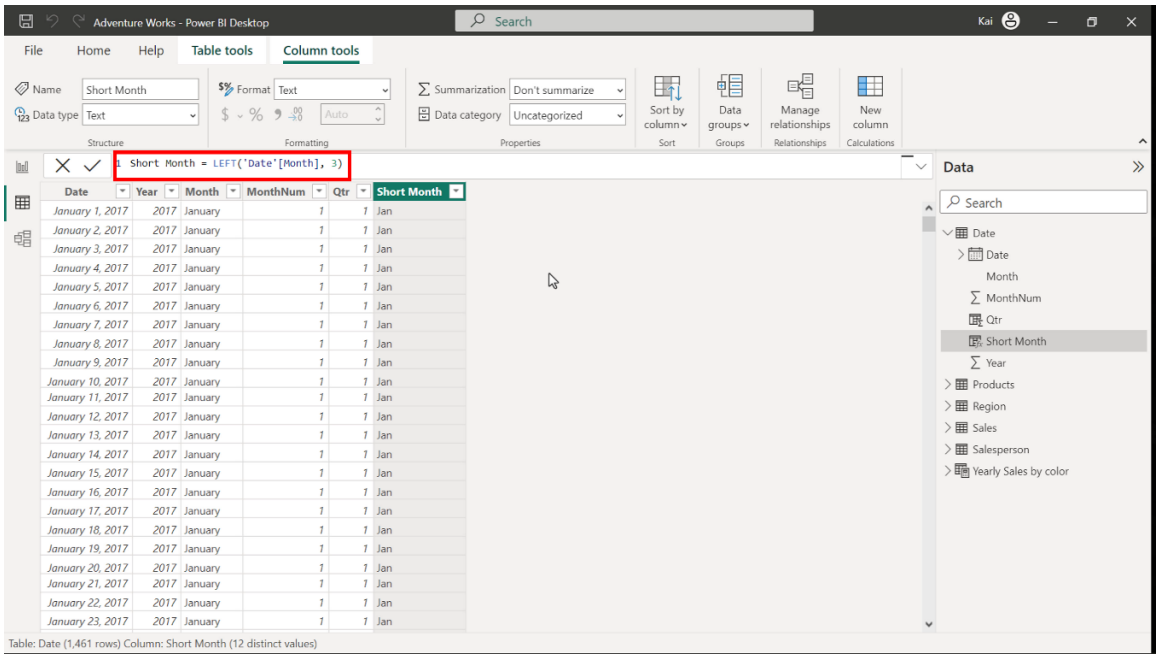
- **QUARTER**: Returns each quarter as a number from the **Date** column.
- **Date** in single quotes is the table, and **Date** in square brackets is the column within the table.



1. Select the **Date** table from the **Data pane** on the right side of Power BI interface. Access **Model view** in the **Calculations group** and select **New column**. Copy and paste the following DAX code into the formula bar:

```
Month =LEFT ( 'Date'[Month], 3 )
```

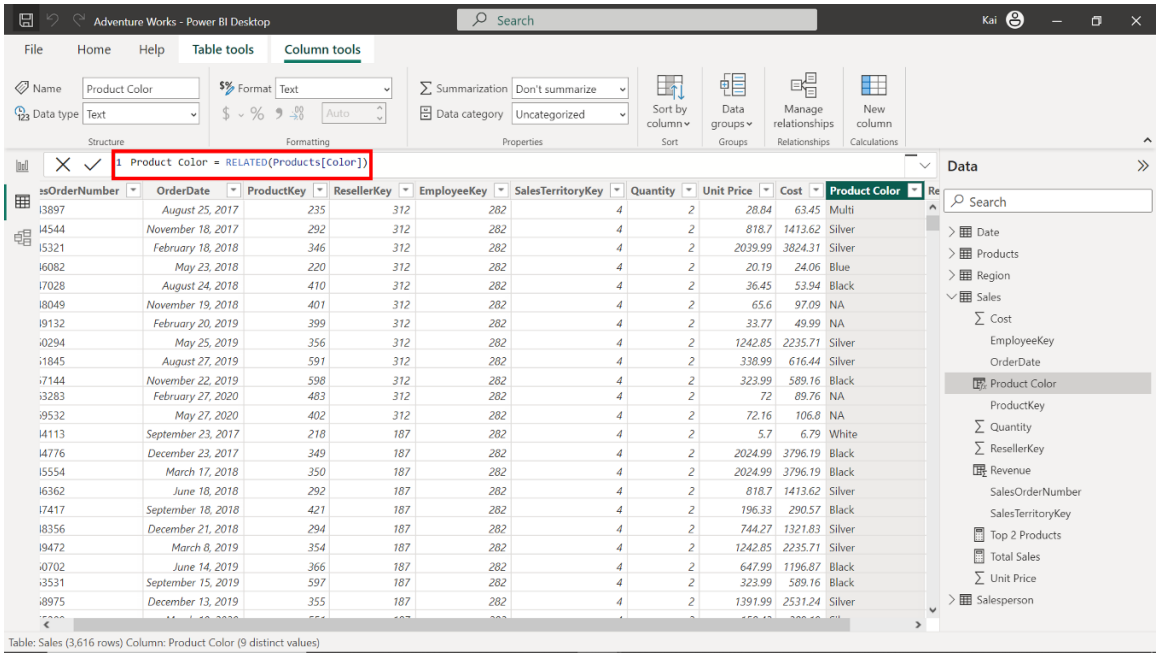
- **LEFT**: Returns the specified number of characters from the start of a text string.
- **Date** in single quotes is the table to be referenced, and **Month** in square brackets is the column name. The number **3** specifies the number of characters in the short month column.



1. To create a new column, select the **Product** table from the **Data pane** on the right side of Power BI interface. Access **Model view** in the **calculations group**. Then select **New column** to expand the formula bar. Copy and paste the following DAX code into the formula bar:

```
Product Color = RELATED ( Products[Color] )
```

- **RELATED** here is the same as referencing a column from another table.



Step 5: Clone Sales Table.

1. Create New Table and Use the following Code:

```
Cloned Sales = ALL(Sales)
```

Step 6:Create Annual Sales Summary Table

1. Create a new calculated table to summarize the Annual Sales using the following code:

```
Annual Sales Summary =
        ADDCOLUMNS(SUMMARIZE(
        Sales,
        'Date'[Year],
         Products[Category]),
        "Total Quantity",
         CALCULATE(SUM(Sales[Quantity])))
```

Knowledge Check

Question 1

You are a data analyst working with a large sales dataset for an e-commerce company. You are employing Power BI as an analytical tool. In which of the following scenarios would it be beneficial to use DAX functions? Select all that apply.

A. Calculating the average sales per region.
B. Filtering the data to display only this year's sales.
C. Creating a column that flags high-value customers.
D. Importing a new sales dataset.

Question 2

True or False: You can create a calculated table in DAX to generate a table based on the defined expression.

A. True
B. False

Question 3

What is the purpose of the **RELATED** function in DAX?

A. To retrieve data from a related table.
B. To filter data based on specific criteria.
C. To establish relationships between tables.

Question 4

Why should you clone a table in Power BI for use as a calculated table?

A. To keep a backup of your original dataset.
B. To reduce the size of your data model.
C. To perform calculations that are not possible on the original dataset.

Question 5

Regarding DAX, what does the term "evaluation context" refer to?

A. The set of filters and context that influence DAX calculations.
B. The visualization and charts created by DAX formulas.
C. The connection between Power BI and external data sources.

Chapter 5: Measures

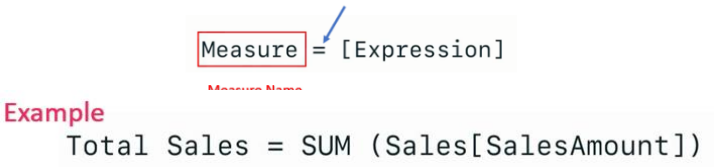
What are Measures?

- Measures in Power BI are used to perform calculations on data model fields.
- Measures play a pivotal role in data analysis and interpretation.
- Measures are used in Power BI to perform aggregations, calculations or evaluations on data that provide meaningful insights.
- Measures are typically used in data visualization elements. Examples of these elements include charts, tables, and cards.
- By using measures, you can compute aggregated values such as **sums**, **averages**, **minima**, **maxima**, **counts** or more complex statistical calculations.

Benefits of Measures

- **Measures are calculated in the context of the visualization** of a report they are used in:
    - This means they are dynamically updated based on filtering and other interactions within the report.
    - In other words, if the context changes, then so does the measure.
    - This dynamic calculation allows you to dive deeper into data and gain insights from different angles and perspectives.
- **Measures are reusable**:
    - Once created, you can continue to recall them in your code.
    - This reduces the repetitive work of creating the same calculations and ensures data consistency across all reports.
- **Track Performance**:
    - Measures can be used to track the performance of different aspects of a business. Measures are commonly used to create key performance indicators, or **KPI**s, essential to monitor business performance.
    - KPIs provide a quick snapshot of performance against predefined targets or benchmarks.
- **Maintain consistency**:
    - Measures help maintain consistency in metrics across different visualizations and reports.
    - Consistency ensures the same results show, regardless of filtering or grouping.
    - In your measures, your calculations must be standardized and uniformly applied throughout the analysis. This ensures accurate and reliable reporting across various visualizations and dashboards.

Measure Syntax



- reusable and complex calculation capabilities, enabling businesses to gain insights from their data and make data driven decisions effectively and efficiently.

Question

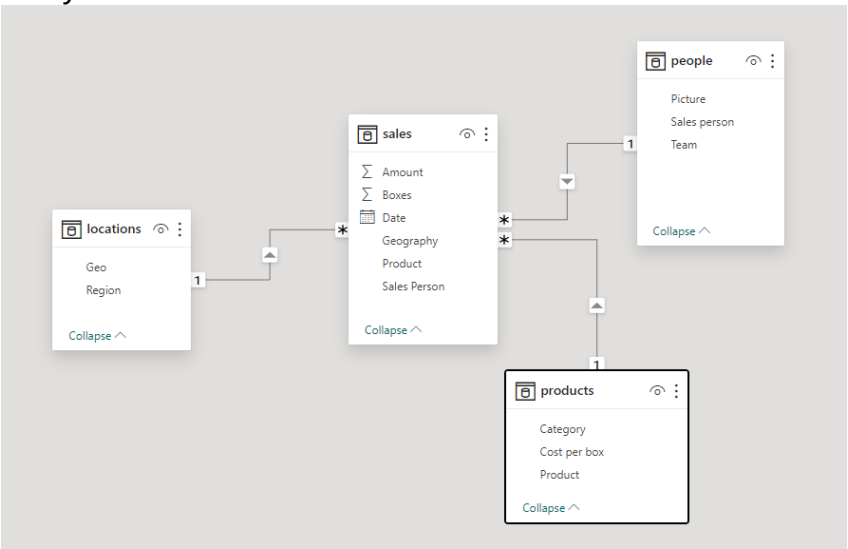
Why are measures used in Power BI?

A. To filter and categorize data.
B. To perform calculations and aggregations.
C. To store descriptive attributes.

Exercise 9: Creating Measures

1. Open **Create Measures START.pbix.**

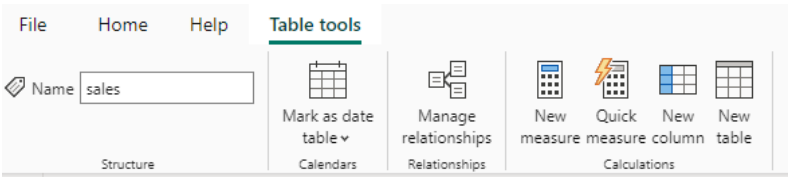
2. Explore your Data Model.

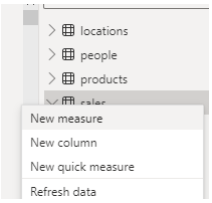


3. This is a chocolate company; Sales are recorded in **Sales** table.
4. The Salespersons recorded in **People** table.
5. The location of the Sales is stored in **Location** table.
6. What chocolate you have sold in the Product table.

Create Total Sales Measure

7. We want to answer the question: What is the total money we have generated?
8. Go to **Table view** and see the data in the tables.
9. In the Sales table we have an **Amount** Column, that is what we want to get its total.
10. Select **Sales** table.
11. **Table tools** ribbon appears on top.
12. Create new measure either:
    a. Right Click **Sales** table and select **New Measure**, or
    b. Select **New Measuer** from the Ribbon.

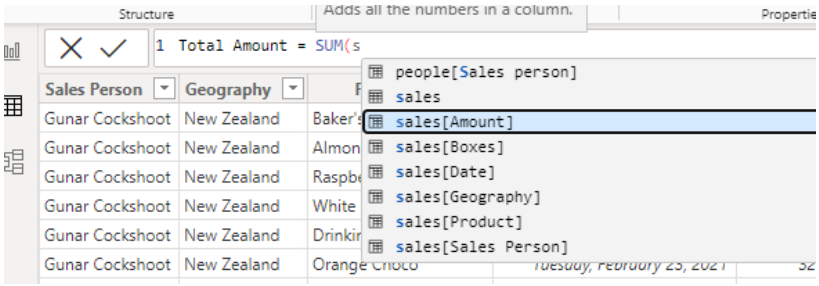




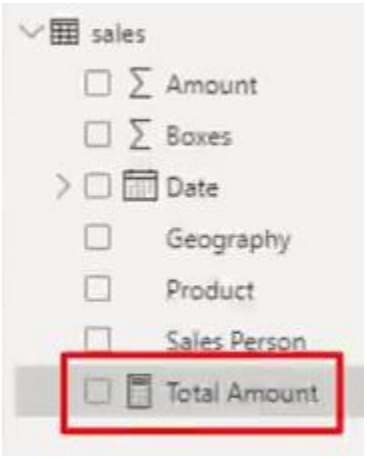
13. Write the DAX Code to create the measure:

```
Total Amount = SUM(sales[Amount])
```

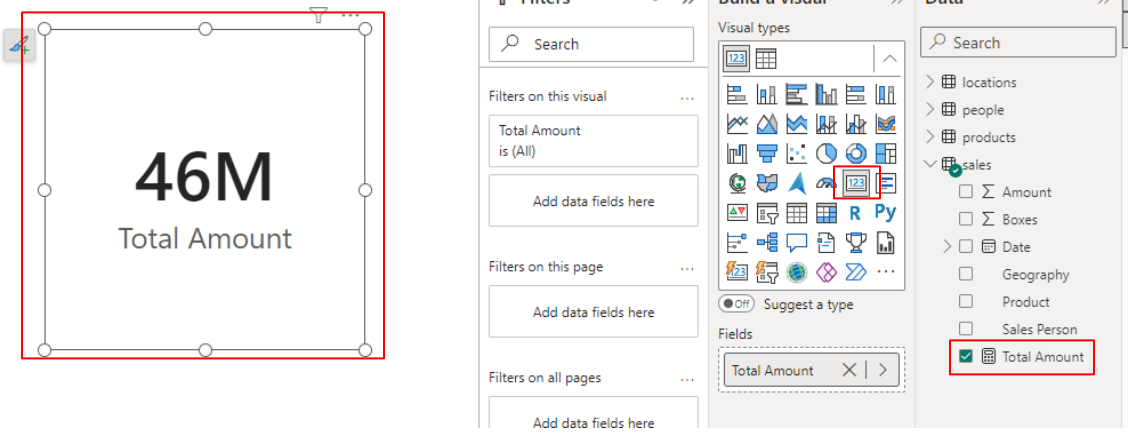
14.



15. Just write **SUM** and choose sales[**Amount**] then press Enter.
16. The Measure appears in the table.



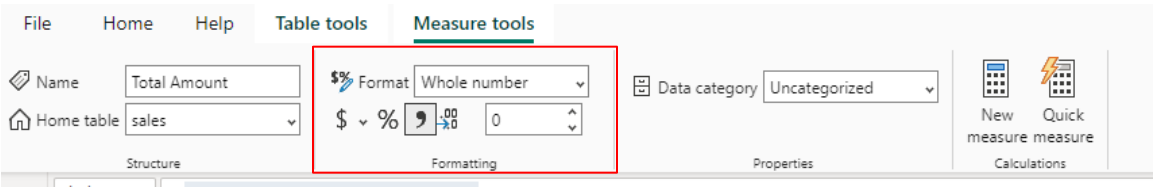
17. Think of a measure as something Acts on **Top of your data.**
18. That is why we could not consider it as a column in the table.
19. You can only see its value when you use in reports.
20. Go to **Report View** and Create a Card with the measure you have created.



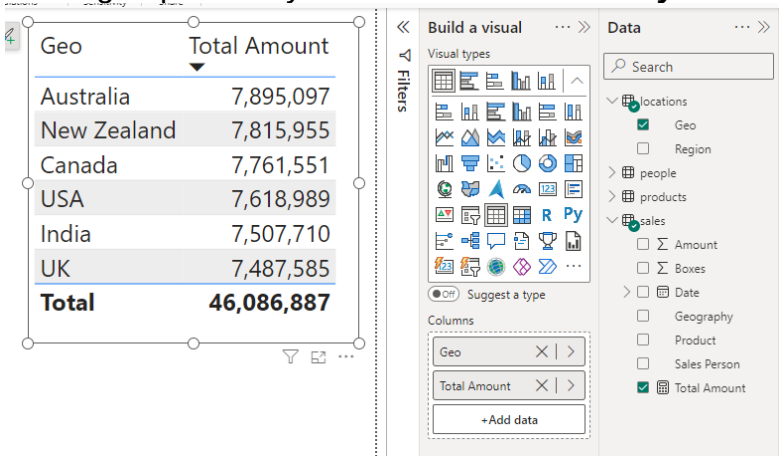
21. To make it total clearer create a table with **Geo** and the **Total Amount** measure.

Format Measure

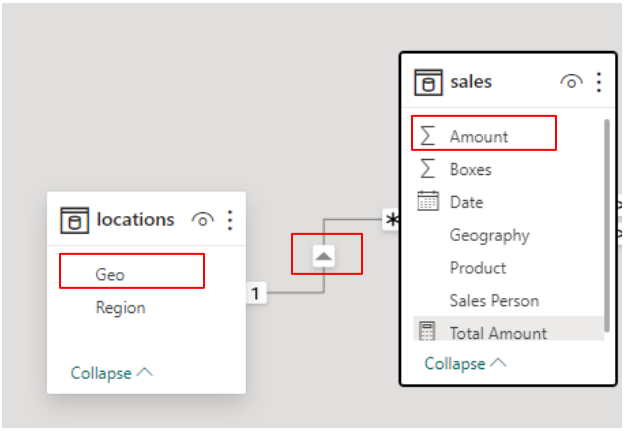
22. Go to **table view** and select your measure.
23. **Measure Tools** appear on the ribbon.



24. In **Format** group format your measure as **Currency** with **no decimals**.



25. That is the power of Power BI it follows the filter direction to calculate each region amount as you specified in the measure.
26. For example it takes **Australia** form the **Geo** and filter the **Amount** according to and then calculate your measure to get the total. `



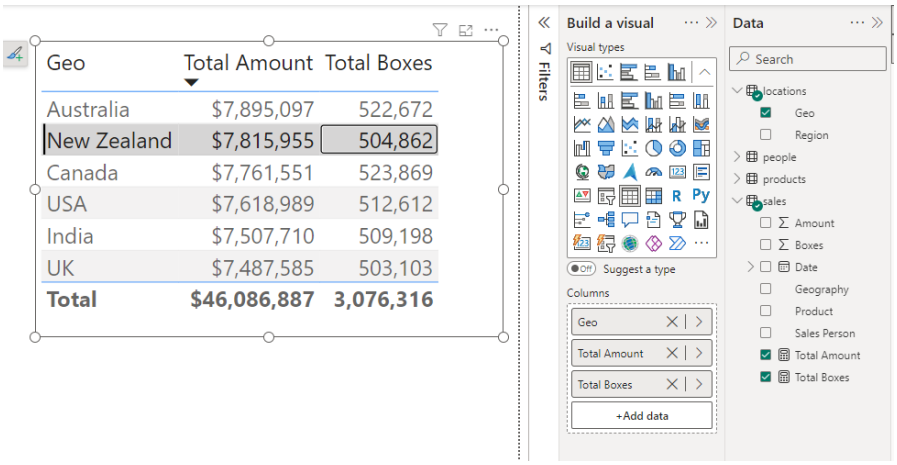
27. It takes into account the filter direction in the model.
28. Remember that Power BI first **Filter** Data then Apply **calculation**.

Create Total Boxes Measure

29. In the same table Sales create another measure **Total Boxes**.

```
Total Boxes = SUM(sales[Boxes])
```

30. Add the new measure to your table you have created in report view and format it as a whole number with separator.



Create Total Shipment Count

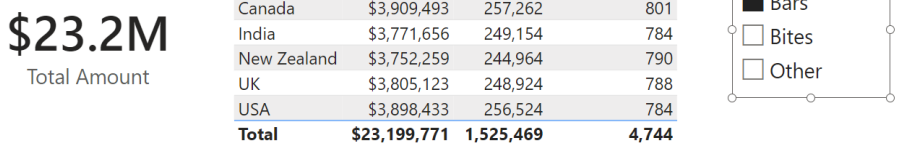
31. We want to know the total shipments we have made.
32. That is the count of all our sales rows.
33. Create **Shipment Count** measure and add it to your report table.

```
Shipment Count = COUNTROWS(sales)
```

34. Format it as **Whole Number** with separator.
35. You can use any filed name to count rows.

Adding Category Filter to your Report

36. Add a **Filter** to your Report and Add **Category** column to it.
37. As you can see all values were calculated again using the new filter.
38. Select Bars Category and see how the other two Visuals are filtered.
39. Remember Power Bi first **filter** the data then **calculate**.



Reuse Measure in Calculation

40. You can reuse the two measures you have defined before and create new measure:

```
Amount per Shipment = [Total Amount] /[Shipment Count]
```

41. You can start to write “[“ so you see all measures you have created.
42. Add the new measure to your report table.
43. Create another measure **Amount per Box**:

```
Amount per Box = DIVIDE([Total Amount],[Total Boxes],0)
```

44. Notice that we have used **DIVIDE** function which saves me the error when dividing by **Zero**. It allows you to all **alternative value**.
45. Notice we do not write the table name before the measure name.
46. You can write the table name before measure.
47. But it is better not to do it.
48. Add your new measure to your table report.

Geo	Total Amount	Total Boxes	Total Shipments	Amount per Shipment	Amount per Box
Australia	$7,895,097	522,672	1,578	5,003.23	15.11
USA	$7,618,989	512,612	1,556	4,896.52	14.86
New Zealand	$7,815,955	504,862	1,601	4,881.92	15.48
Canada	$7,761,551	523,869	1,590	4,881.48	14.82
India	$7,507,710	509,198	1,556	4,825.01	14.74
UK	$7,487,585	503,103	1,568	4,775.25	14.88
Total	**$46,086,887**	**3,076,316**	**9,449**	**4,877.44**	**14.98**

Reusing your Measures in many places

49. Create a new Page in your report and create a new table with **Salesperson**, **Amount per Box** , and **Amount per Shipment**.
50. Click on the columns to sort to analyze which salesperson has the top Amount per box and amount per shipment.

Sales person	Amount per Box	Amount per Shipment
Jehu Rudeforth	14.76	6,558.86
Mallorie Waber	15.01	5,657.39
Roddy Speechley	15.19	5,550.66
Camilla Castle	15.04	5,528.40
Van Tuxwell	14.68	5,169.23
Madelene Upcott	15.48	5,027.48
Jan Morforth	14.84	4,996.56
Marney O'Breen	14.25	4,959.98
Dotty Strutley	15.10	4,945.42
Beverie Moffet	15.30	4,940.56
Total	**14.98**	**4,877.44**

Exercise 10: IF Function

1. User file If **10 IF Function Start.pbix.**
2. If you want to check salespeople who have met their targets or not.
3. Create a new Page for your report page3.
4. Create a Measure **Amount per Shipment Target Acieved?:**

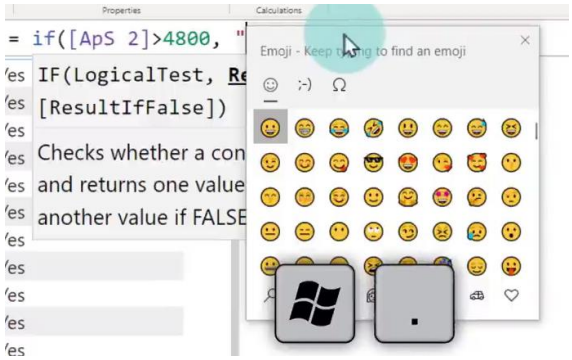
```
5. Amount Per Shipment Target  Achived? = IF([Amount per
                Shipment] > 4800 ,"Yes", "No")
```

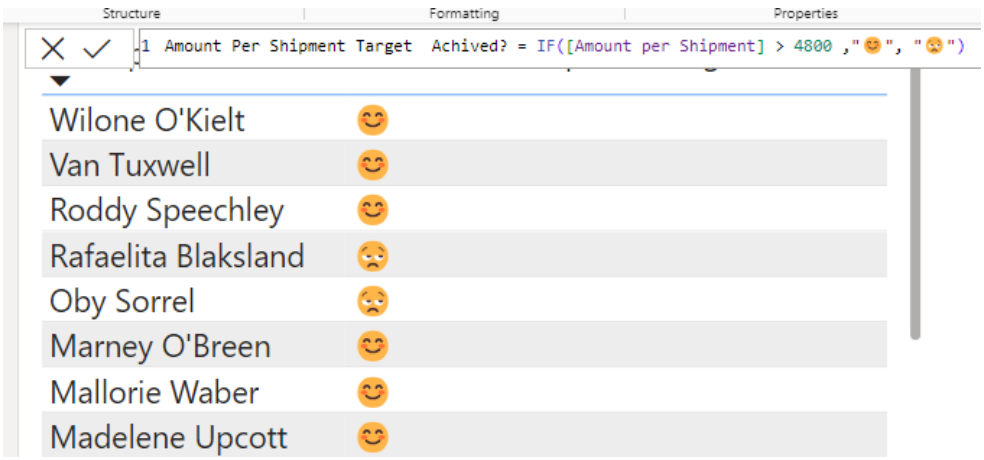
6. Create a new table with **sales person** and **Amount per shipment target Achieved**

Sales person	Amount Per Shipment Target Achived?
Wilone O'Kielt	Yes
Van Tuxwell	Yes
Roddy Speechley	Yes
Rafaelita Blaksland	No
Oby Sorrel	No
Marney O'Breen	Yes
Mallorie Waber	Yes
Madelene Upcott	Yes

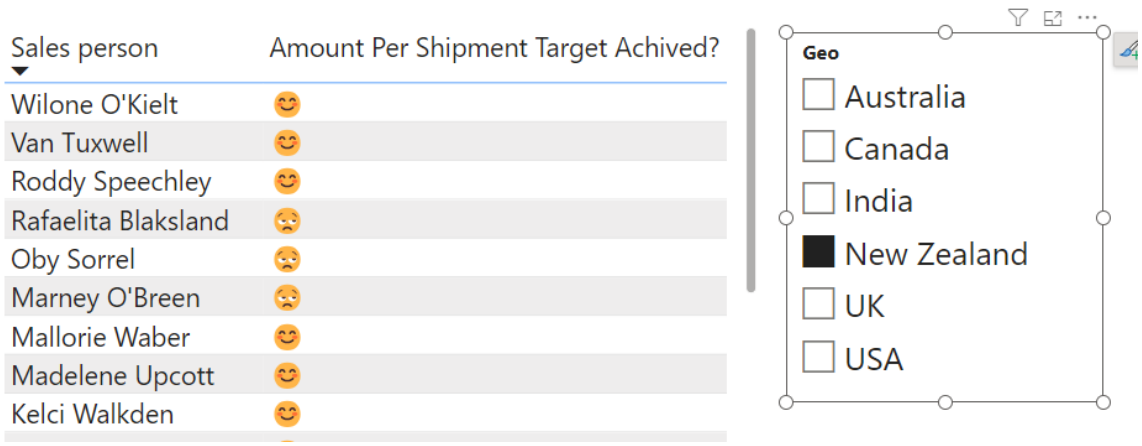
Using Emoji Trick

7. Select your measure again and start to edit.
8. You can use Emoji instead of Yes and No words.
9. Delete "**Yes**" and "**No**" in the quotations and start writing.
10. Press (**Windows key + dot Key**)
11. This will show the emoji list.
12. Select **happy face** and **sad face** instead of yes and no.





13. Add a filter with Geo to your report and see how people in each country do.

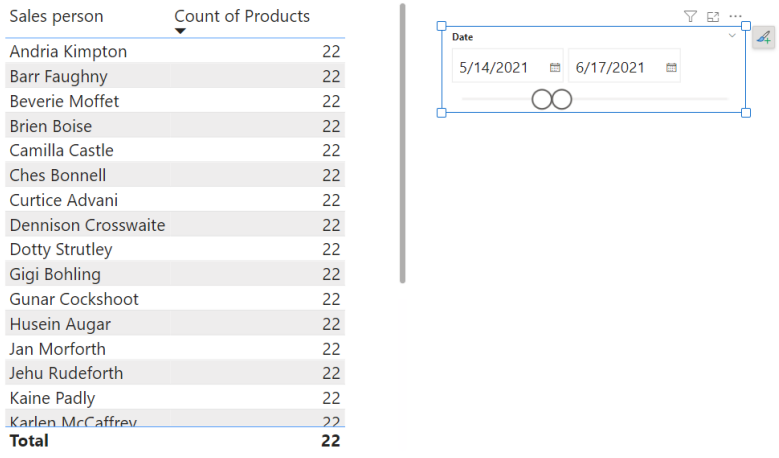


Exercise 11 DISTINCTCOUNT Function

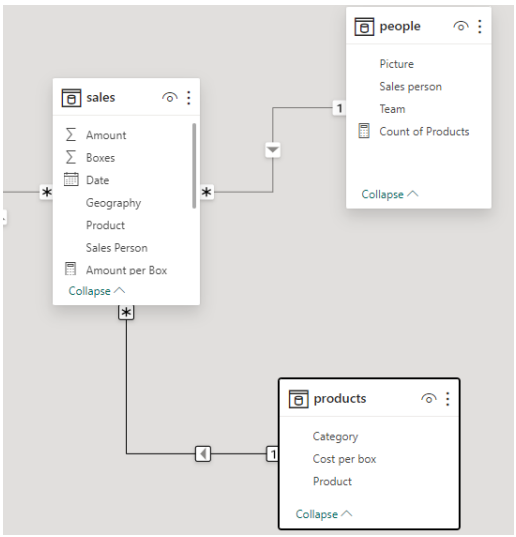
1. Use file **11 DISTINCTCOUNT Function Start.pbix**
2. I want to know if each salesperson sells all the products or only some of them.
3. Create a new page name it “DISTINCTCOUNT Function”.
4. Create new table and add **salesperson** filed to it.
5. We want to filter the count of Product for each salesperson.
6. Try to add this DAX measure:

```
Count of Products = COUNTROWS(products)
```

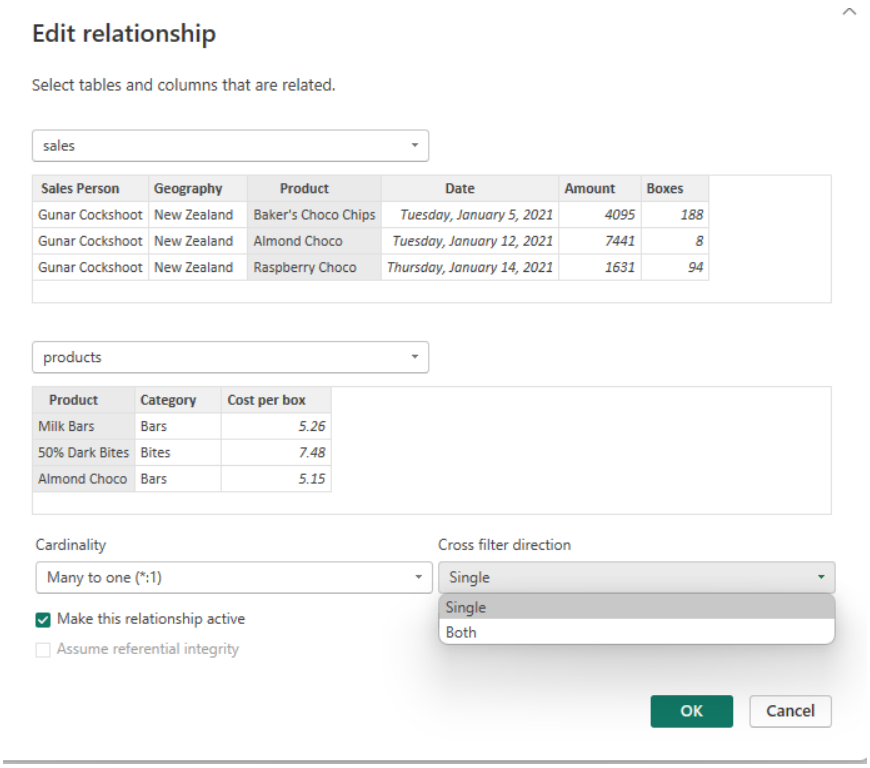
7. Add the measure to your table notice that all have **22**.
8. Check your **product table** it has all **22** products.
9. Add filter to your report and add **Date** to it.



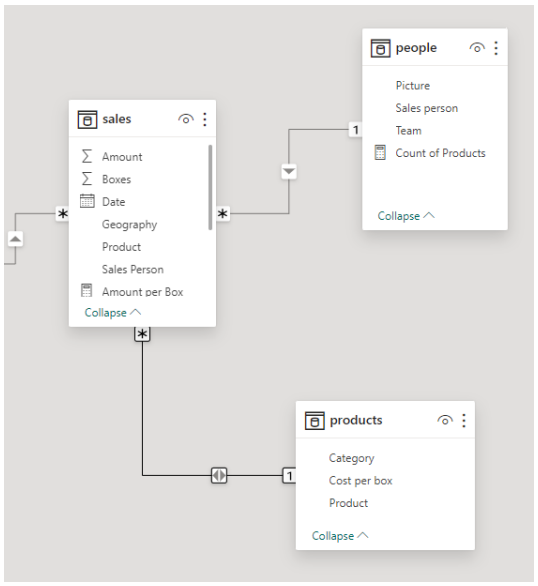
10. Make the period only one week, notice also 22 still there for everyone.
11. What does it mean?
12. It means that there is no filter happen.
13. Can you till why?
14. Go to your model can you see the filter direction?



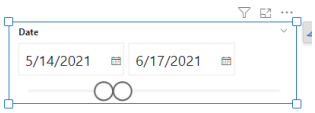
15. Notice that:
    a. **People** table filters➔ **Sales** table and
    b. **Product**s table filter **Sales** table
16. But People cannot filter products.
17. To solve this problem, you can change the connection between **Sales** and **Products** to **both**.
18. Double click on the connection and choose **Both** in the “cross filter direction” option.



19. Now your connection allows People to filter Product.
20. Go to your report and see the result.



Sales person	Count of Products
Gunar Cockshoot	20
Beverie Moffet	18
Husein Augar	17
Oby Sorrel	17
Rafaelita Blaksland	17
Gigi Bohling	16
Barr Faughny	15
Ches Bonnell	15
Wilone O'Kielt	15
Kelci Walkden	14
Marney O'Breen	14
Curtice Advani	13
Dotty Strutley	13
Karlen McCaffrey	13
Madelene Upcott	13
Jan Morforth	12
Total	**22**

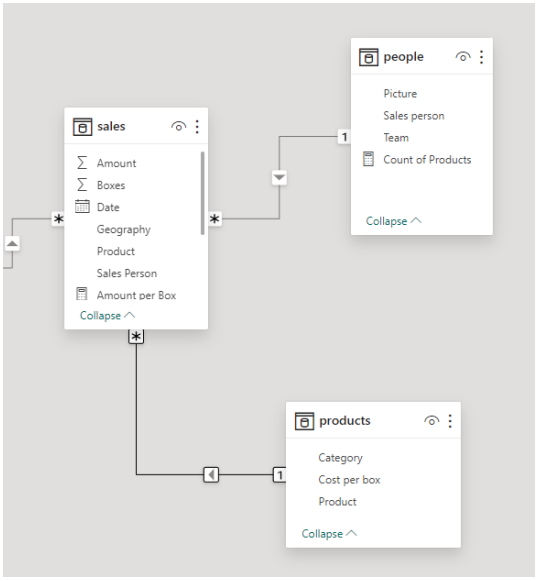


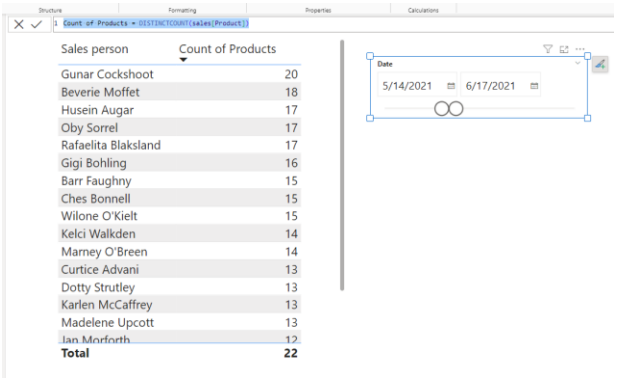
Solve the problem using COUNTDISTINCT

21. You can solve the problem also by Filtering Product in the Sales table, so you do not need to change the cross-filtering direction.
22. Go to your connection and make it Single again.
23. Change the code of DAX for the Product Count measure to:

```
Count of Products = DISTINCTCOUNT(sales[Product])
```

24. You will get the same result.

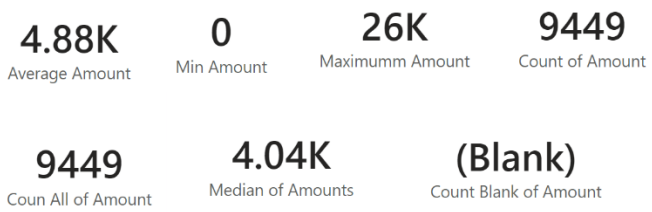




Exercise 12 Creating Aggregation Function

1. Use file **12 Aggregation Functions.pbix**
2. Create Some measures for Amount Column like (Average, Min , Max , Count , CountA , Median , COUNTBLANK)

3. Create a New Page.
4. Use cards for each measure to show in the page.



Exercise 13: Using CALCULATE FUNCTION

1. Use file **13 CALACULATE Function.pbix**
2. As you have seen the Power BI first **filter** then **calculate**.
3. In DAX you create the calculation so far, and Power BI filter the apply your calculations.
4. What if you want to **interfere** with the process of Filtering too?
5. Create a new page and add a table to your report.
6. Add **Product** and **Shipment Count** to your table.

Product	Total Shipments
50% Dark Bites	400
70% Dark Bites	413
85% Dark Bars	453
99% Dark & Pure	427
After Nines	447
Almond Choco	434
Baker's Choco Chips	418
Caramel Stuffed Bars	431
Total	**9,449**

7. This gives you how many times each product was shipped.

Low level Shipment Count

8. What if I want to know How many Shipments were happened while the Number of Chocolate boxes were Low **(<50 Box**)?
9. What do we want to do?
10. Go to **table View** and select the arrow in the **Boxes** field in Sales table.
11. You can see that number of Boxes is from 0 and up to 3360.
12. We want to count Number of Shipment for each product, **ONLY** when the Number of Boxes in this shipment were under **50**.
13. So, we want to add that **Extra Layer of filtering** before calculation happens.
14. You can use the special function in DAX **CALCULATE** for that.

15. Create new measure:

```
Low Low Shipment Count = CALCULATE([Shipment Count], sales[Boxes] < 50)
```

16. Add the new measure to your table.

Product	Total Shipments	Low Level Shipment Count
50% Dark Bites	400	51
70% Dark Bites	413	43
85% Dark Bars	453	63
99% Dark & Pure	427	57
After Nines	447	47
Almond Choco	434	57
Baker's Choco Chips	418	50
Caramel Stuffed Bars	431	53
Total	**9,449**	**1086**

17. Now Create **Low Box Shipment %** measure:

```
Low Box Sipment % = DIVIDE([Low Shipment Count], [Shipment Count],0)
```

18. Format the measure as % and added it to your table

Product	Total Shipments	Low Level Shipment Count	Low Box Sipment %
50% Dark Bites	400	51	12.75%
70% Dark Bites	413	43	10.41%
85% Dark Bars	453	63	13.91%
99% Dark & Pure	427	57	13.35%
After Nines	447	47	10.51%
Almond Choco	434	57	13.13%
Baker's Choco Chips	418	50	11.96%
Caramel Stuffed Bars	431	53	12.30%
Total	**9,449**	**1086**	**11.49%**

Calculate Bar Chocolate Shipments

19. I want to filter only for Bar Chocolate.
20. It is so easy just calculate the **Total Shipment** measure with filtering the Category to be Bars:

```
Bar Shipment = CALCULATE([Shipment Count],products[Category] = "Bars")
```

21. Add the new measure to your table

Product	Total Shipments	Low Level Shipment Count	Low Box Sipment %	Bar Shipment
50% Dark Bites	400	51	12.75%	
70% Dark Bites	413	43	10.41%	
85% Dark Bars	453	63	13.91%	453
99% Dark & Pure	427	57	13.35%	427
After Nines	447	47	10.51%	
Almond Choco	434	57	13.13%	434
Baker's Choco Chips	418	50	11.96%	418
Caramel Stuffed Bars	431	53	12.30%	431
Total	**9,449**	**1086**	**11.49%**	**4744**

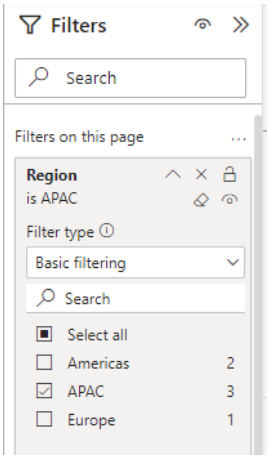
22. Create another measure **Bar Shipment %**

```
Bar Shipment % = DIVIDE([Bar Shipment],[Shipment Count])
```

23. Format it as %
24. Create another table for **salesperson** and Bar Shipments, **Low Box Shipments Shipments, Bar Shipment , Bar Shipment % ,Total Shipments.**

Sales person	Low Level Shipment Count	Bar Shipment	Bar Shipment %	Total Shipments
Gunar Cockshoot	51	242	53.42%	453
Ches Bonnell	55	223	52.72%	423
Marney O'Breen	52	218	52.53%	415
Curtice Advani	44	215	52.31%	411
Beverie Moffet	50	223	52.22%	427
Roddy Speechley	24	107	51.69%	207

25. Sort **Bar Shipment %** to know which Salesperson have high Bar shipment %
26. Expand **Filter Pan**
27. In this **Page Filter** Add:
    a. Region
    b. Type of Filter: Basic Type.
    c. Chose Region: **APAC**



28. See how salesmen do in this Region for **Bars Shipment %.**

Overwriting the Normal Filter Using CALCULATE

25. You can overwrite the normal filter.
26. For Example, we are now filtering the region for **APAC**.
27. Let us make a measure that calculate the shipment of American region only.
28. Create a measure:

```
Americas Shipment = CALCULATE([Shipment Cont] ,
        locations[Region]="Americas")
```

29. Add the new measure to your table.
30. Notice that Total Shipments of Americas are calculated although we have filtered for APAC.

Create Measure for Friday Amount

31. Suppose I want to know Sales in some day of the week, **Friday** for example.
32. So, I want calculate Total Amount when Day = Friday.
33. Go to Table View.
34. Select Sales table and notice you have a **Date** column.
35. Click **New Column** from the ribbon.
36. Write the DAX expression to create **Weekday** column:

```
Weekday = WEEKDAY(sales[Date].[Date])
```

37. This will return a column with numbers represent the day.
38. Sunday is 1 and Saturday is 7
39. Notice that Power BI calculates each value in **Row Context**.
40. So, every value corresponds to the value in value in Date column in the same row.
41. Now we can create a report to answer what is our total sales on Fridays?
42. Create New Page in your report view.
43. Create new table with **Product** and **Total Amount**.
44. Add new measure to calculate the Friday Amount:

```
Friday Amount = CALCULATE([Total
  Amount],sales[Weekday] = 6)
```

45. Format as Currency with decimal = 0
46. Add the new measure to your table.

Product	Total Amount	Friday Amount
50% Dark Bites	$1,937,565	$404,208
70% Dark Bites	$2,012,682	$389,270
85% Dark Bars	$2,128,140	$385,007
99% Dark & Pure	$2,145,192	$334,201
After Nines	$2,174,172	$480,683
Almond Choco	$2,021,208	$464,037
Total	**$46,086,887**	**$9,327,556**

Using Short Format of Calculate

47. Power BI allows you to write CALCULATE Function in short format.
48. The syntax is:

Measuer Name = [Calculation] (The Condition)

49. Try to change your Friday Amount to the New format and see:

```
Friday Amount = [Total Amount](sales[Weekday] = 6)
```

50. You get the same result.

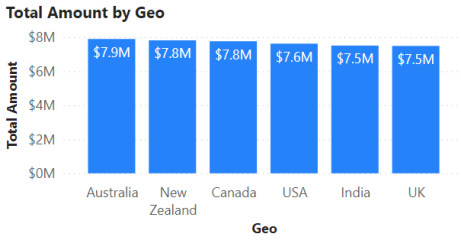
Exercise 14: Variables in DAX

1. Use file **14 Variables in DAX.pbix**
2. Suppose we want to have a measure calculating Total Amount in Both Newland and Australia.
3. Create new measure NZAU Amount:
4. Use (**Shift + Enter**) to go to new line.

```
NZAU Amount =
    var NZAmount = CALCULATE([Total Amount],locations[Geo] =
"New Zealand")
    var AUAmount = CALCULATE([Total Amount],locations[Geo] =
"Australia")
RETURN
    NZAmount +AUAmount
```

5. Use **Tab key** to have spaces in the line.
6. Format the measure as currency.
7. Create A new Page.
8. Create A column visual showing total amount for each country or create a table showing that

Geo	Total Amount
Australia	$7,895,097
Canada	$7,761,551
India	$7,507,710
New Zealand	$7,815,955
UK	$7,487,585
USA	$7,618,989
Total	**$46,086,887**



9. Create a table with product and NZAU Amount

Exercise 15: OR Condition

1. Use file **15 OR condition.pbix**.
2. You can reach what we have done in the previous exercise using the OR condition in calculation instead of using variables.

Using in operator

1. Create a new Measure NZAUSales

Product	NZAU Amount
50% Dark Bites	$699,734
70% Dark Bites	$701,316
85% Dark Bars	$686,028
99% Dark & Pure	$714,175
After Nines	$752,304
Almond Choco	$622,223
Baker's Choco Chips	$635,887
Total	**$15,711,052**

```
NZAUSales = CALCULATE( [Total Amount],locations[Geo] in { "New
Zealnd" , "Australia"})
```

2. Format as currency.
3. Create a table in a new page and add Product and the **NZAUSales** measure.

Product	NZAUSales
50% Dark Bites	$334,299
70% Dark Bites	$410,347
85% Dark Bars	$403,067
99% Dark & Pure	$306,831
After Nines	$326,354
Almond Choco	$323,085
Baker's Choco Chips	$326,606
Caramel Stuffed Bars	$380,184
Choco Coated Almonds	$396,802
Drinking Coco	$338,737
Total	**$7,895,097**

Using OR

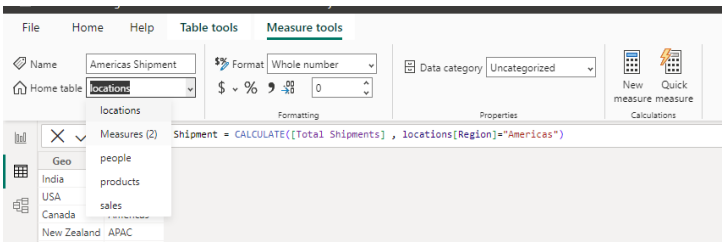
4. Change the NZAU measure code using OR Condition
5.

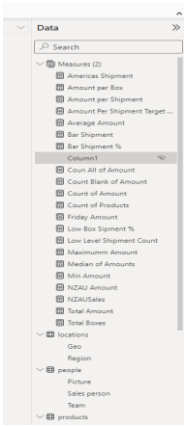
```
NZAUSales = CALCULATE( [Total Amount],locations[Geo] = "New
                    Zealnd"  || locations[Geo] = "Australia")
```

6. You get the same result.

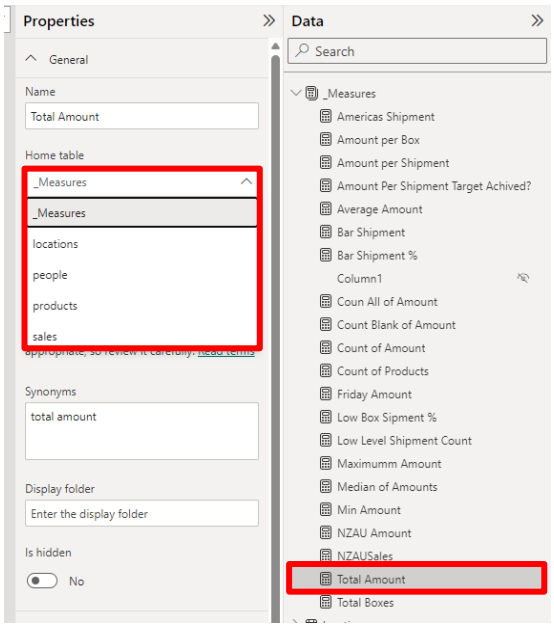
Exercise 16: Organize your Measures in one table

1. Use file **16 Organizing Measures in a table .pbix**
2. Go to **Table View**
3. Select **Enter Data**
4. Create a one column empty table with name **_Measurs (** _ is to make it on Top).
5. You can move all your measures to the new table either:
    a. Go to your **table view**
    b. Select each measure and move to the **Measures** table using





    Location option in **measure tools** ribbon
    c. Click on the Eye Icon next to **Column 1** to hide.
    d. Or just right click and delete it.
6. Or:
    a. Go to **Model View**.
    b. Select all measure you want.
    c. In Properties Pane Change the **Home Table** to the _Measure Table.
    d.



Knowledge Check

Question 1

Which of the following statements about the measures is correct?

A. Measures can reference columns directly.
B. Measures store values in the model.
C. Measures can reference other measures directly.

Question 2

A. Which DAX function can summarize the number of entries in a table?
B. **COUNTROWS**
C. **SUM**
D. **AVERAGE**

Question 3

How can measures help with data analysis in Power BI?

A. By creating new tables and relationships between data tables in your models.
B. By combining data from various external data sources.
C. By performing calculations and aggregations for data visualization and analysis

Question 4

True or False: Measures in Power BI are calculated columns that store results based on a specific DAX expression.

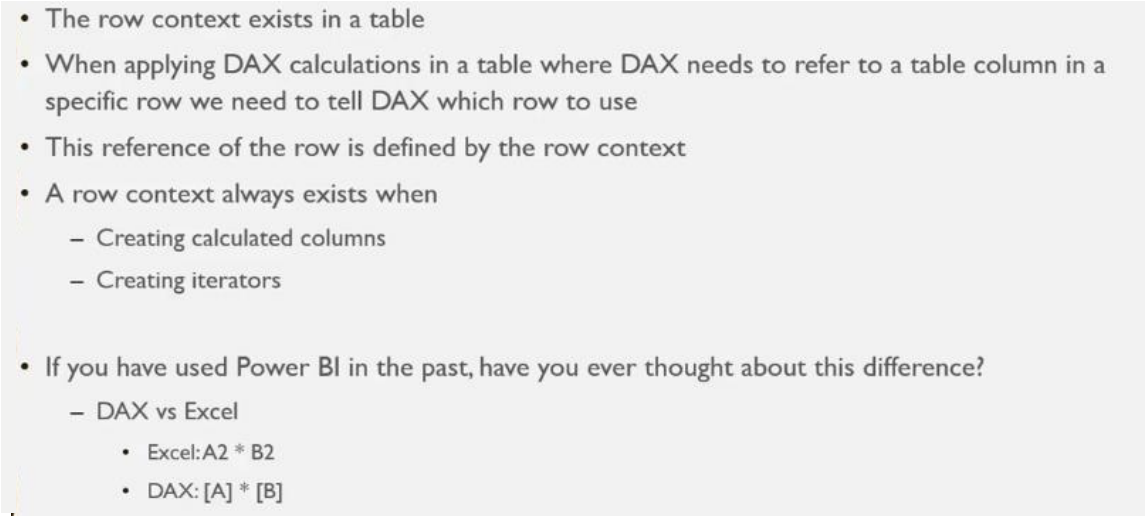
A. True
B. False

Chapter 6: More About DAX

About DAX

- DAX seems to be easy to understand but takes whole life to master.
- Repetition is the key (How to become a great photographer? Take thousands of Pictures).
- Always keep fundamentals of DAX in your mind while working with code.

Row Context



Exercise 17: Row Context

1. Use file: **17 Row Context Start.pbix**.
2. Remember that when you calculate a column in Excel you use it row by row in a table, the calculation **ITERATES** through the table row by row.
3. In the **table view** select **Orders** table.
4. Home ➔ New Column.

```
SalesColumn = Orders[SellingPrice] * Orders[Amount]
```

5. If you only type sell you will have the option to select a column?
6. Why?

7. Because you can here reference a **Cell** from that column, so you are in the **Row Context**.
8. Compare you **SalesColumn** to **Sales** Column you have in the original table, they are the same.
9. Row context means iterating over the table row by row.
10. Right Click the Orders table and Create a new measure:

TotalSales = SUM(Orders[SalesColumn])

11. Go to **Report View**.
12. Create a new table with **Category, TotalSales.**

Category	TotalSales
Cheese	251,147.60
Condiments	113,992.56
Cookies	101,646.94
Deserts	179,863.65
Drinks	288,694.65
Fish	137,989.31
Fruits	105,745.70
Meat	174,622.55
Total	**1,353,702.96**

13. Now try to add the **SalesColumn** to the table.

Category	TotalSales	Sum of SalesColumn
Cheese	251,147.60	251,147.60
Condiments	113,992.56	113,992.56
Cookies	101,646.94	101,646.94
Deserts	179,863.65	179,863.65
Drinks	288,694.65	288,694.65
Fish	137,989.31	137,989.31
Fruits	105,745.70	105,745.70
Meat	174,622.55	174,622.55
Total	**1,353,702.96**	**1,353,702.96**

14. You will get the exact Numbers.
15. It is best practice not to use the **default** aggregation of a column.

Creating Total Sales Directly

1. It is better not to use **Calculated Column** as much as you can, try to use measure instead because they do not take size in your model.
2. Try to Create new measure **TotalSales2** (SellingPrie * Amount) :
3. Try to write the name of the column **Selling Price**, Power BI won't allow you.
4. Why?
5. That is because measure itself **does not have a Row context**.
6. You can only aggregate a column using aggregation function like **SUM**.
7. If you want to do that you will need Iterators.

Iterators

- Iterators „iterate“ over an object
- The object can be a „standard“ table or a virtual/temp table
- The value iterators create can be a scaler (number, text, date) or a virtual/temp table
- An iterator always has (at least) the 2 arguments:
  - The object (aka table) to iterate over
  - The expression which should be evaluated for each row in the object
- The expression can be considered a temporary column which only exists until the final value is created
- Most DAX Iterator functions can be identified by the „X“ at the end of the function name

- Iterators has row context (They iterate row by row)
- **Temp/Virtual** tables are the one generated in a measure for temporary use. They only exist when the measure needs them to calculate, then removed and never exist in model.
- The iterator can return a scaler means one value.
- Also, Iterator can return a Virtual/Temp table.
- Iterator needs two arguments at least:
  - The table to iterate through.
  - The expression that will be evaluated row by row.
- As the iterator use **table** and table has **Row Context** also Iterator has Row Context.
- You can consider the **expression** as a **virtual column**, it exists until the calculation finished then removed.
- Most iterator functions end with the X letter (SUMX, MINX,MAXX……).

- Some common iterators:
  - SUMX
  - MINX / MAXX
  - AVERAGEX
  - RANKX
  - FILTER
- Syntax: SUMX(Table, Expression)
  = Aggregation – Object(Table) – Expression to be evaluated for each row in the table
- Example: SUMX(Orders, Orders[Amount] * Orders[Price])

Exercise 18 Iterators

1. Use file **18 Iterators Start.pbix**.

2. Now let us solve the problem we had in the previous exercise.
3. Create a measure but use iterator this time:

```
TotalSales2 =
SUMX(Orders,Orders[SellingPrice]*Orders[Amount])
```

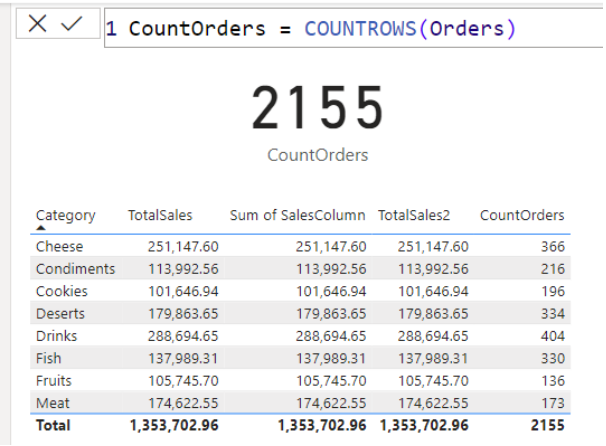
4. Notice that this time Power BI allows you to select the sellingPrice Column.
5. That is because iterators have row context.
6. Add the new measure to your table and notice you have the same result as before but this time you did not add sizes to your model using

Category	TotalSales	Sum of SalesColumn	TotalSales2
Cheese	251,147.60	251,147.60	251,147.60
Condiments	113,992.56	113,992.56	113,992.56
Cookies	101,646.94	101,646.94	101,646.94
Deserts	179,863.65	179,863.65	179,863.65
Drinks	288,694.65	288,694.65	288,694.65
Fish	137,989.31	137,989.31	137,989.31
Fruits	105,745.70	105,745.70	105,745.70
Meat	174,622.55	174,622.55	174,622.55
Total	**1,353,702.96**	**1,353,702.96**	**1,353,702.96**

calculated column.

Using Filter Function

7. SUMX function return a single value (Scaler).
8. But Filter Function is an iterator, and it returns a table.
9. I want to count orders in my table Orders.
10. Let us create a measure **CountOrders** and show it in a Card visual:
11. Add it to your table, and see it works.

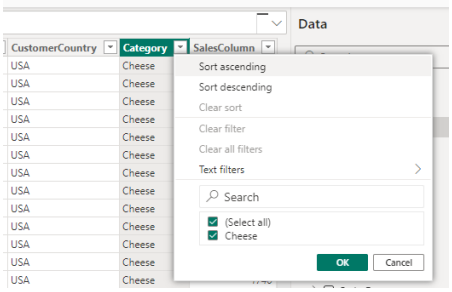


Category	TotalSales	Sum of SalesColumn	TotalSales2	CountOrders
Cheese	251,147.60	251,147.60	251,147.60	366
Condiments	113,992.56	113,992.56	113,992.56	216
Cookies	101,646.94	101,646.94	101,646.94	196
Deserts	179,863.65	179,863.65	179,863.65	334
Drinks	288,694.65	288,694.65	288,694.65	404
Fish	137,989.31	137,989.31	137,989.31	330
Fruits	105,745.70	105,745.70	105,745.70	136
Meat	174,622.55	174,622.55	174,622.55	173
Total	**1,353,702.96**	**1,353,702.96**	**1,353,702.96**	**2155**

12. You can not use Filter directly in a measure, because it returns a table.
13. But you can use it inside other functions in a measure.
14. Let us create a table to see how Filter works.
15. Create a new table **CheesTable** that only have orders of **Cheese Category.**

```
CheeseTable = FILTER(Orders,Orders[Category] = "Cheese")
```

16. A table is added to your model, and you can see it only have the orders of Cheese.



17. Why Filter is Row context function?
18. Because to create a table it goes row by row in the table to check if the category is Cheese or not to result the file filtered table.
19. Now I want to create a measure that count the Cheese row and tell me how many cheeses order I have sold.
20. You can use COUNTROW again, but this time give Filter function as an argument.

```
CheeseMeasure = COUNTROWS(FILTER(Orders,Orders[Category] = "Cheese"))
```

21. Show the measure in another Card Visual.

22. You can remove the **CheeseTable** it has no value to our model.

Combine Iterators

23. We want to evaluate the sales of the customers from USA.
24. We want to:
    a. Filter (Keep only rows) of customers only from USA.
    b. For those remaining rows, I want go row by row.
    c. Calculate **SellingPrice** * **Amount** for each row.
    d. Aggregate all result and have the SUM of all Sales.

```
TotalUSASales =

    SUMX(FILTER(Orders,Orders[CustomerCountry] = "USA"),

    Orders[SellingPrice] * Orders[Amount]
```

25. Remember that:

a. **SUMX** and **FILTER** functions are **row context** functions.
b. **FILTER** returns a table (Virtual/Temp).
c. **SUMX** returns a Scalar (One Value).
d. The Temp table created by **Filter** and temp column created by **Expression** are removed automatically after calculation.

26. Add the new measure to your table.
27. Format all your measures in table as currency with 0 decimal.

Category	TotalSales	Sum of SalesColumn	TotalSales2	CountOrders	TotalUSASales
Cheese	$251,148	$251,148	$251,148	366	$40,208
Condiments	$113,993	$113,993	$113,993	216	$18,789
Cookies	$101,647	$101,647	$101,647	196	$20,379
Deserts	$179,864	$179,864	$179,864	334	$38,820
Drinks	$288,695	$288,695	$288,695	404	$64,246
Fish	$137,989	$137,989	$137,989	330	$24,211
Fruits	$105,746	$105,746	$105,746	136	$10,309
Meat	$174,623	$174,623	$174,623	173	$44,554
Total	**$1,353,703**	**$1,353,703**	**$1,353,703**	**2155**	**$261,513**

28. You can format your DAX code using the site: DAX Formatter.

https://www.daxformatter.com

Filter Context

- Filter Context relates to a subset of filters applied to the model before a DAX expression is evaluated
- „Filters applied by the user interface“ (Visualization, Slicers,…)
- 2 different Filter Contexts
  - Implicit (see above)
  - Explicit (CALCULATE/CALCULATETABLE)
- Filter Context filters the table to the specific rows the expression can be evaluated on
- Filter Context can consist of one or more filters (Intersection of filters)
- Filter Context does not iterate!
- When the Filter Context is empty all data can be evaluated
- Filter Context propagates through the relationship (one-to-many, both) to related tables in the model

Exercise 19: Filter Context

1. Use file **19 Filtesr ContextStart.pbix**.
2. Create a new Page.
3. Create a new **_Measure** Table and move all your measures to for easy to find.
4. Create new table with **Category**, **TotalSales2**.

Category	TotalSales2
Drinks	$288,695
Cheese	$251,148
Deserts	$179,864
Meat	$174,623
Fish	$137,989
Condiments	$113,993
Fruits	$105,746
Cookies	$101,647
Total	**$1,353,703**

5. Notice that we have filter here for each category and every Number is different.
6. So, the table **Orders** were filtered to **Cheese** for example then the measure were evaluated.
7. This is a filter context, and you have only one filter (**Category**).
8. Now add **Region** filed to the table.

Category	Region	TotalSales2
Cheese	America	$92,445
Cheese	Europe	$158,702
Condiments	America	$39,949
Condiments	Europe	$74,043
Cookies	America	$37,035
Cookies	Europe	$64,612
Deserts	America	$74,810
Deserts	Europe	$105,054
Drinks	America	$134,332
Drinks	Europe	$154,363
Total		**$1,353,703**

9. Now you have two filters, one for **Category** and one for **Region**.
10. For example, the first row has filter **Cheese** and **America** applied to get Number **92,455**.
11. Remove the Region field from the table.
12. Another way to apply Filters is Slicers.
13. Add Slicer to report with **ShipMode** filed.
14. Add a Card Visual with **TotalSales2** measure.
15. Select **DHL Express** from the ShipMode filter, both table and card are

filtered.

16. So, for TotaSales2 Measure:

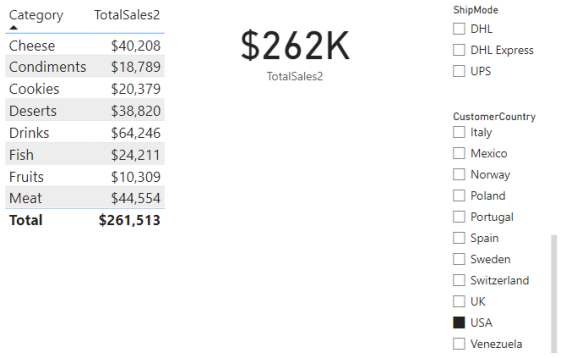
```
TotalSales2 = SUMX(Orders,Orders[SellingPrice]*Orders[Amount])
```

a. The orders table is not evaluated all row by row now.
b. Orders table is filtered first for **DHL Express**.
c. Then the Expression is evaluated row by row in the expression (`Orders[SellingPrice]*Orders[Amount]`).
d. `Then SUMX is evaluated.`

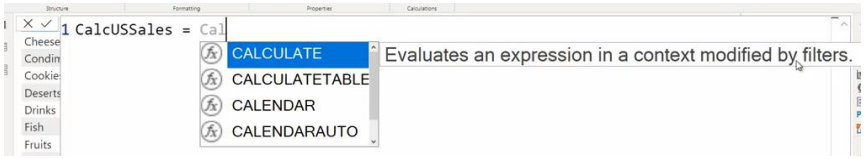
17. Also, in the table two filtered were applied.
18. So, filters can come form one visual or more.

Using CALCULATE Function

19. You can also apply filter using the CALCULATE Function.
20. If I want to filter my table for customers USA.
21. Add filter visual with CustomerCountry filed.
22. Filter to USA.



23. I can also force the filter in the measure using the CALCULATE Function.
24. Right Click your **_Mesures** table and create new measures.



25. Notice what tooltip say about CALCULATE function
26. CALCULATE take two arguments or more:
    a. Expression to be evaluated.
    b. Filter to apply before calculation.
27. I want to apply the filter to my previous measure **TotalSales2**.
28. So, I will give it as a first argument.
29. In Filter argument make customerCountry = “USA”.

```
CalcUSSales = CALCULATE([TotalSales2],Orders[CustomerCountry] = "USA")
```

30. Add the new measure to your table.

Category	TotalSales2	TotalUSASales
Cheese	$251,148	$40,208
Condiments	$113,993	$18,789
Cookies	$101,647	$20,379
Deserts	$179,864	$38,820
Drinks	$288,695	$64,246
Fish	$137,989	$24,211
Fruits	$105,746	$10,309
Meat	$174,623	$44,554
Total	**$1,353,703**	**$261,513**

31. Question : CALCULATE creates filter context, why it is allowed to use row context in the expression (`Orders[CustomerCountry] = "USA")?`
32. `Why am I allowed to refer to specific column if I do not have row context?`
33. The answer is it is a short expression for a filter function.
34. Try the complete expression and it will work

```
CalcUSSales2 = CALCULATE([TotalSales2] ,
FILTER(ALL(Orders[CustomerCountry]) , Orders[CustomerCountry]
="USA")
```

Category	TotalSales2	TotalUSASales	CalcUSSales2
Cheese	$251,148	$40,208	$40,208
Condiments	$113,993	$18,789	$18,789
Cookies	$101,647	$20,379	$20,379
Deserts	$179,864	$38,820	$38,820
Drinks	$288,695	$64,246	$64,246
Fish	$137,989	$24,211	$24,211
Fruits	$105,746	$10,309	$10,309
Meat	$174,623	$44,554	$44,554
Total	**$1,353,703**	**$261,513**	**$261,513**

35. **ALL** Function return all column with no filter

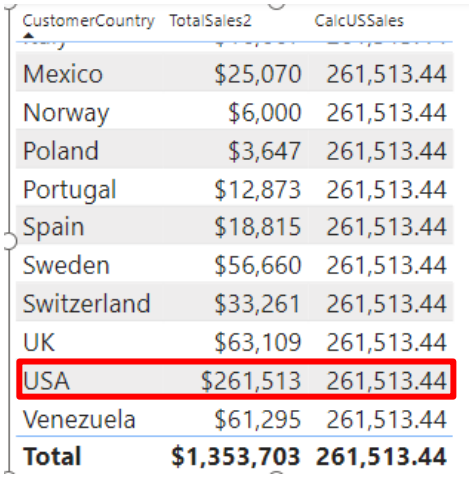
CALCULATE overrides visual filters

36. Create a new page.
37. Create a new table with **CustomerCountry** m [**TotalSales2**].
38. The total sales is filtered for each country.
39. Add to table your measuer[CalcUSSales].
40. Notice that all values are the same.
41. It is **USA** sales and correct in the line of USA for sure only.
42. That is because **CALCULATE overrides external filters**.
43. That is why the filter only see USA everywhere.
44. For example for row of **Mexico** filter will not apply.

Keeping Filters Using CALCULATE

45. If we want to apply the filter while using Calculate.
46. You can do this Using the **KEEPFILTERS** Function.

47. Go back to your **CalcUSSales** Measure and modify to keep filters



CustomerCountry	TotalSales2	CalcUSSales
Mexico	$25,070	261,513.44
Norway	$6,000	261,513.44
Poland	$3,647	261,513.44
Portugal	$12,873	261,513.44
Spain	$18,815	261,513.44
Sweden	$56,660	261,513.44
Switzerland	$33,261	261,513.44
UK	$63,109	261,513.44
USA	$261,513	261,513.44
Venezuela	$61,295	261,513.44
Total	**$1,353,703**	**261,513.44**

48. your code must be like that:

```
CalcUSSales = CALCULATE([TotalSales2],
KEEPFILTERS(Orders[CustomerCountry] = "USA"))
```

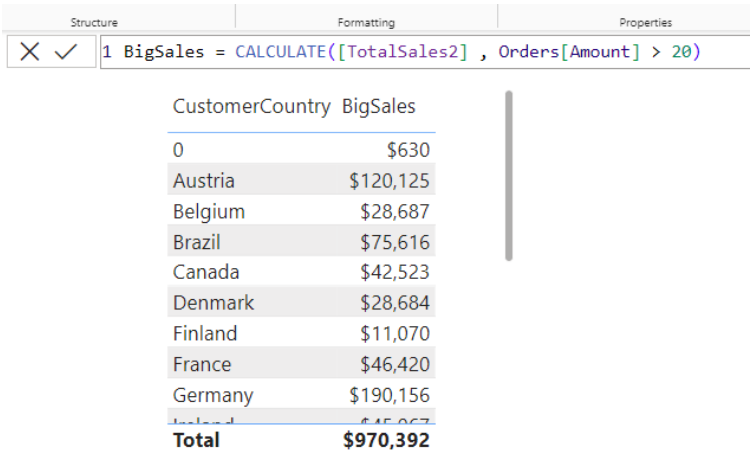
49. See the result now on the table.

CustomerCountry	TotalSales2	CalcUSSales
Finland	$20,627	
France	$89,242	
Germany	$242,495	
Ireland	$52,624	
Italy	$16,667	
Mexico	$25,070	
Norway	$6,000	
Poland	$3,647	
Portugal	$12,873	
Spain	$18,815	
Sweden	$56,660	
Switzerland	$33,261	
UK	$63,109	
USA	$261,513	261,513.44
Venezuela	$61,295	
Total	**$1,353,703**	**261,513.44**

50. Only USA sales appear because you have applied the two filters.
51. For example, UK filter and USA filter gives blank.

Calculate Gig sales!

52. We would like to know the sales of orders that have amount > 20.
53. Create a measure and table showing the countries and the Big Sales measure.
54. Try it yourself, you should have something like that.
55.



Chapter 7 : Time Intelligence Functions

What is Time Intelligence Functions?

- Time intelligence functions refers to **methods and processes that aggregation compared data over time**.

Time Related Dimensions:

- Data analysts can deploy time intelligence functions to analyze data, based on time-related dimensions. Time-related dimensions include:
    - Dates,
    - weeks,
    - months,
    - quarters and
    - years.
- You can also generate comparisons of time-related data over annual periods and year-to-date or YTD.

Why do data analysts view time intelligence as important?

- Time intelligence provides the ability to analyze data within the context of time. This enables a more in-depth understanding of trends and patterns.

Benefits of time Intelligence:

- **Trend Analysis**
    - Time intelligence is useful for **trend analysis**. Identifying trends and past business performance is crucial for future decisions.
    - For example, You can use time intelligence data to examine historical sales trends and recognize if certain products sell better at specific times of the year.
    - Identifying trends in past business performance is crucial for future decisions.
- **Forecasting and predictive analysis**

    - Insights derived from time intelligence also help with forecasting and predictive analysis.
    - You can forecast future trends and plan activity based on historical trends.
    - It can make informed predictions about sales and demands, which helps with:
        - resource planning,
        - budgeting, and
        - risk management.
    - For instance, if the data shows a consistent increase in mountain bike sales every spring, the company can ensure adequate inventory before the season starts.
- **Real-time performance monitoring**
    - Time intelligence also enables real-time performance monitoring.
    - This is possible by creating dynamic measures like **year-to-date** or **YTD** and month-to-date or **MTD**.
    - You can use these measures to monitor real-time performance against key performance indicators. The company can then use these insights to respond quickly to changing conditions.
- **Comparative analysis**
    - Time intelligence calculations facilitate comparative analysis.
    - An example of this is year-over-year, or **YOY functions**.
    - Company can compare its current growth rate, sales performance, and other metrics against data from previous years to analyze its progress.
- **Optimizing Sales and Marketing Strategies**
    - Time intelligence also facilitates the optimization of sales and marketing strategies.
    - Company can analyze its sales trends and the impact of its marketing efforts over time.
    - It can then use the results of these analyses to fine-tune its marketing strategies and sales tactics to improve its results.

Using Time Intelligence in Power BI

- Implementing time intelligence involves creating calculated fields and measures to analyze data over time.
- You can use Power BI automatic time intelligence features or deployed DAX formulas to create quick measures.
- Power BI offers date-time feature that allows easy data analysis by year, quarter, month, and days. This is useful for smaller data models.
- Power BI automatically creates **a one date table** for each **date** column in the dat model to analyze data, but different data attributes. This table is hidden from the user because Power BI handles it automatically.
- You can also use custom DAX calculations to shape your data model and implement time intelligence calculations with more complex and non-standard requirements.

Question

What is the advantage of time intelligence in Power BI for businesses?

A. It enables businesses to analyze data based on time-related dimensions.

B. It enables businesses to analyze data through real-time data processing.
C. It enables businesses to create hierarchies based on date data for visualization.

Setting up a common date table

The role of a common date table

- A **common date table** or **date dimension** is a prerequisite for time intelligence calculations.
- You can't execute them without a date dimension.
- The date dimension must meet the following requirements:
    - There must be one record per day,
    - there must be no missing or blank dates, and
    - it must start from the minimum date and end at the maximum date, corresponding to the fields in your parameters.

But what if your data model is missing a date dimension?

- In this instance, you can use Power BI's, **auto date/time intelligence**.
- You can also create a date dimension in Power Bi using either **Power Query** or **DAX**. This is useful when working on large datasets with complex calculations.

CALENDER and CALENDERAUTO

- You can create a date dimension with DAX using the **CALENDAR** and **CALENDARAUTO** functions.
- Both functions return a calculated table with a **single date column** and a list of date values when executed.

```
Date =

  CALENDAR

    (

       DATE (2017, 1, 1),

       DATE (2021, 12, 31)

    )
```

Question

True or False: The **CALENDARAUTO** function takes both the start and the end date from a dataset.

A. True
B. False

Exercise 20: Set up a common date table Using DAX

- Use file **20 Set up a common date table.pbix**

Scenario

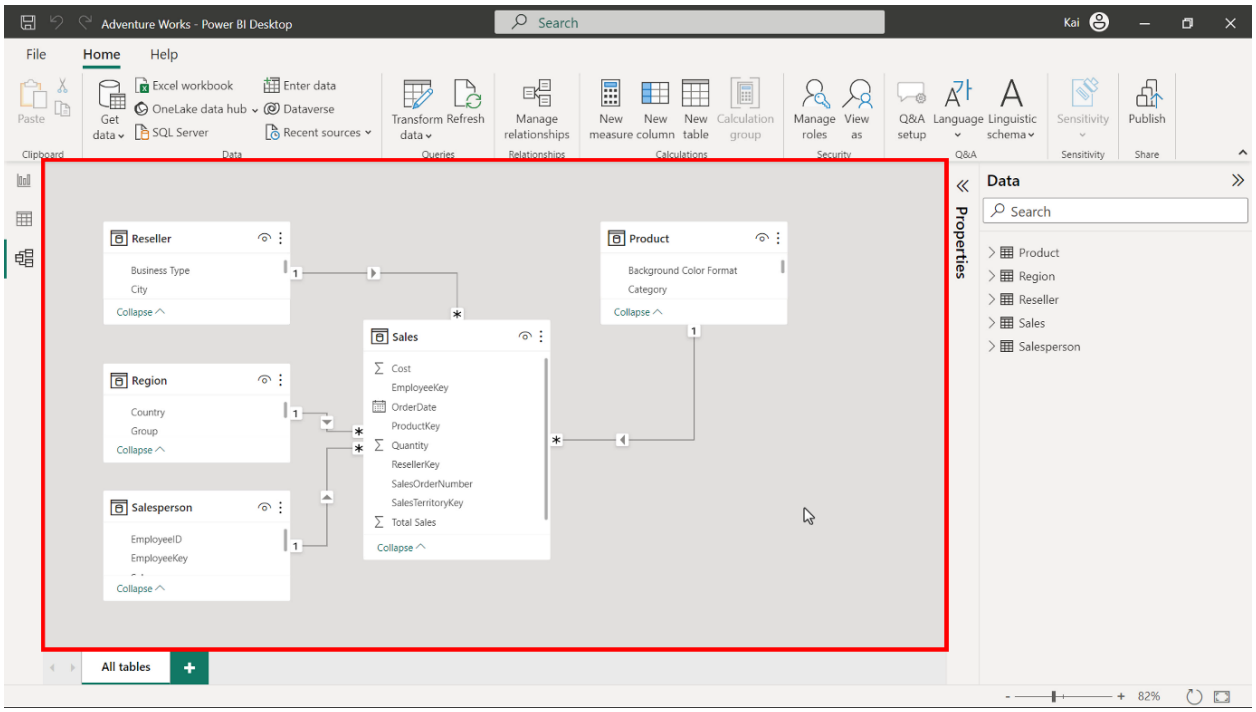
Your company collects data from a range of different sources and collates this data in a data model that contains the following four tables:

- **Sales**
- **Salesperson**
- **Products**
- **Reseller**
- And **Region**

However, there's no date dimension table in these datasets. This makes it difficult to perform time intelligence analysis. Help Adventure Works to create a common-date table in its data m

Step 1: Observe the data model and create a date dimension table using DAX.

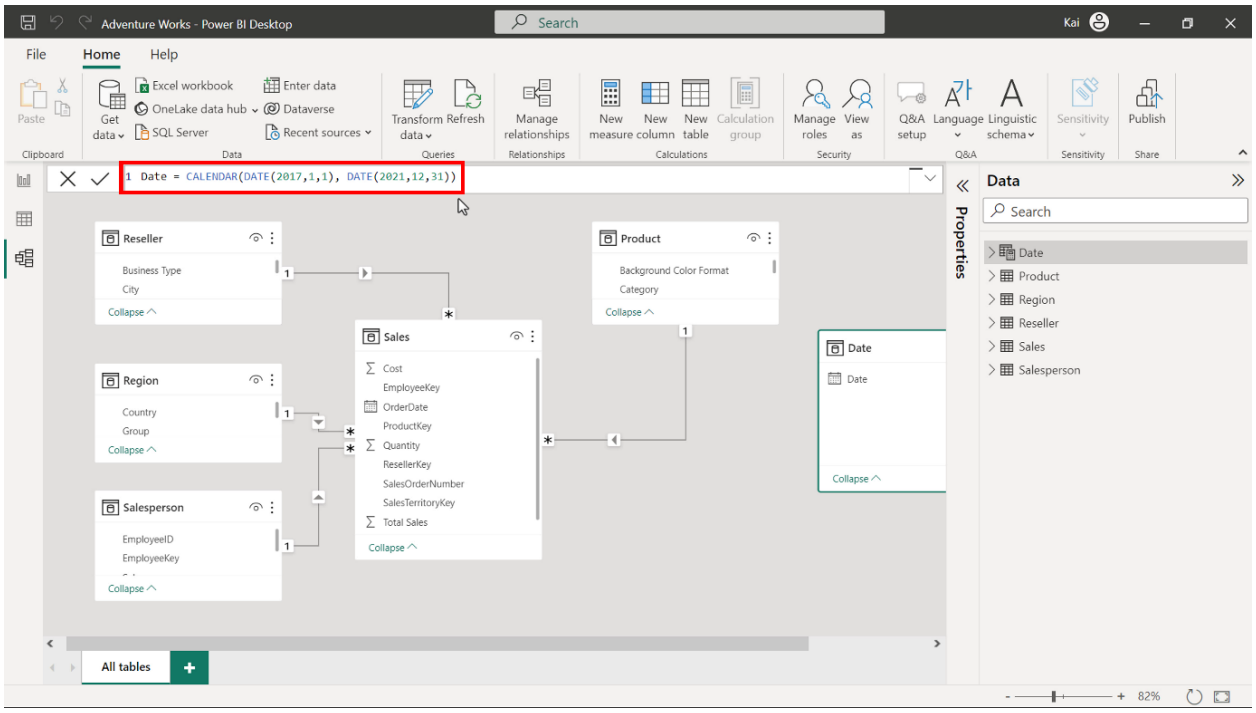
1. Access the **Model** tab in Power BI to view the data model's tables. Note that there is no date dimension table present.



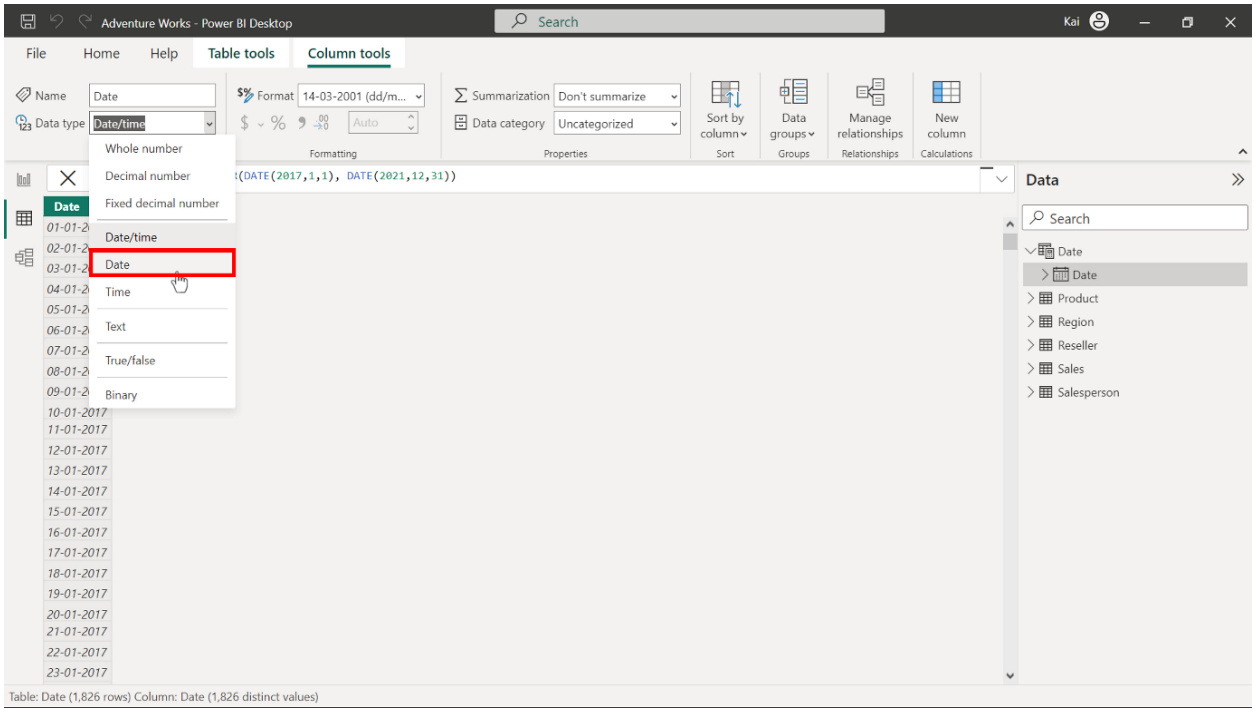
1. Navigate to the **Home** tab and select **New** table from the calculations group. In the formula bar, input the following DAX code using the **CALENDAR** function to create a table with a single column containing dates.

```
1
Date = CALENDAR ( DATE( 2017, 1, 1 ), DATE ( 2021, 12, 31 ))
```

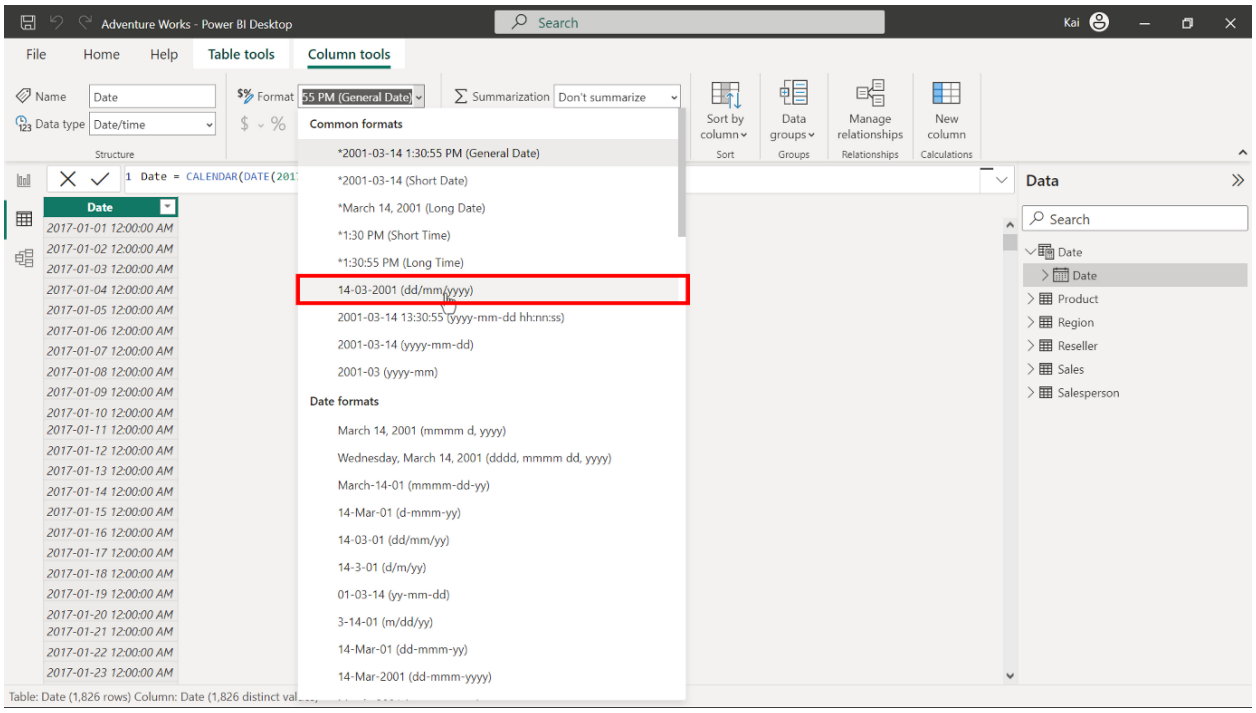
In the **CALENDAR** function, you need to specify the start and end dates. The Adventure Works sales data starts in 2017 and ends in 2020. The start date must occur on or before the date column of the dataset. The end date must be on or after the end date of the dataset. Execute the code to generate a table with a single date column containing a list of dates with time.



1. Now you must format and configure the table. Rename the column as **Date** and format the column as **Date** data type. Select the **Date** column and navigate to the **Column tool** tab.



Select the appropriate date format from the format drop-down list of options.

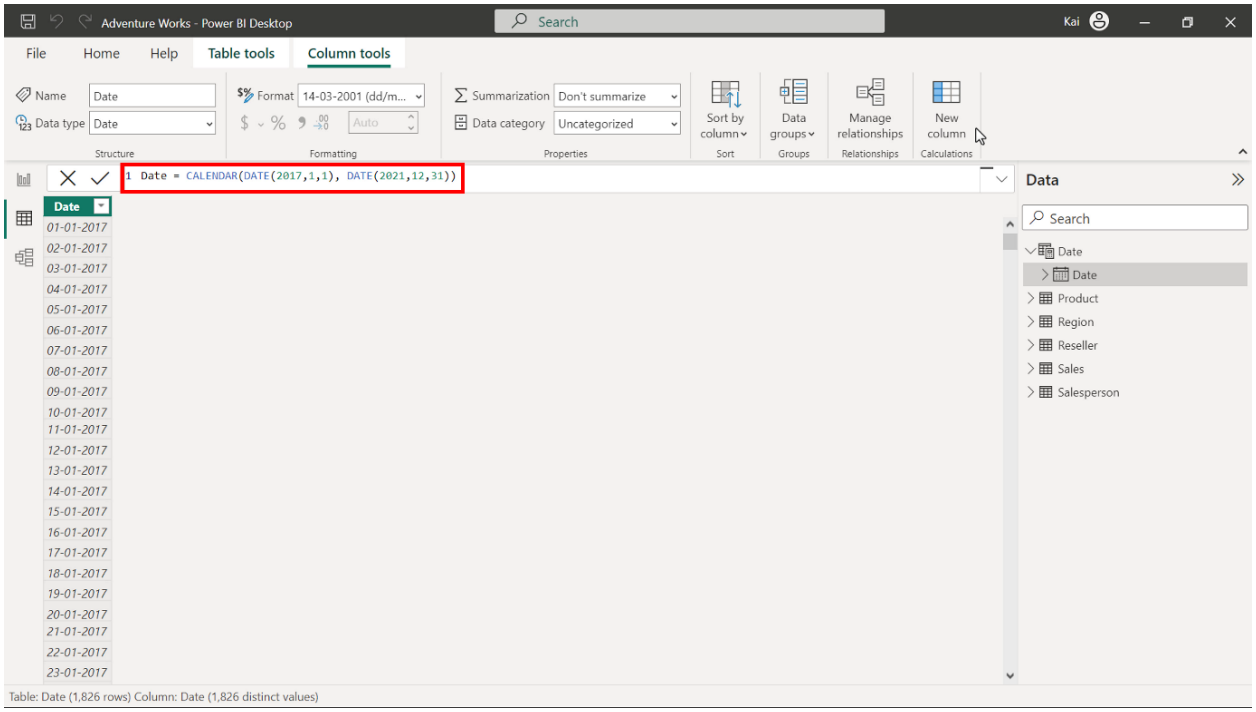


1. Next, you must populate the date dimension table with related columns like **year**, **month number**, **month**, **day of the week**, and **week number**. Select **New column** from the **Calculations** group of the **Column tools** tab to expand the DAX formula bar. Then enter the following DAX codes, one in each step.

```
Year = YEAR ( 'Date'[Date] )
Month = FORMAT ( 'Date'[Date], "MMMM" )
```

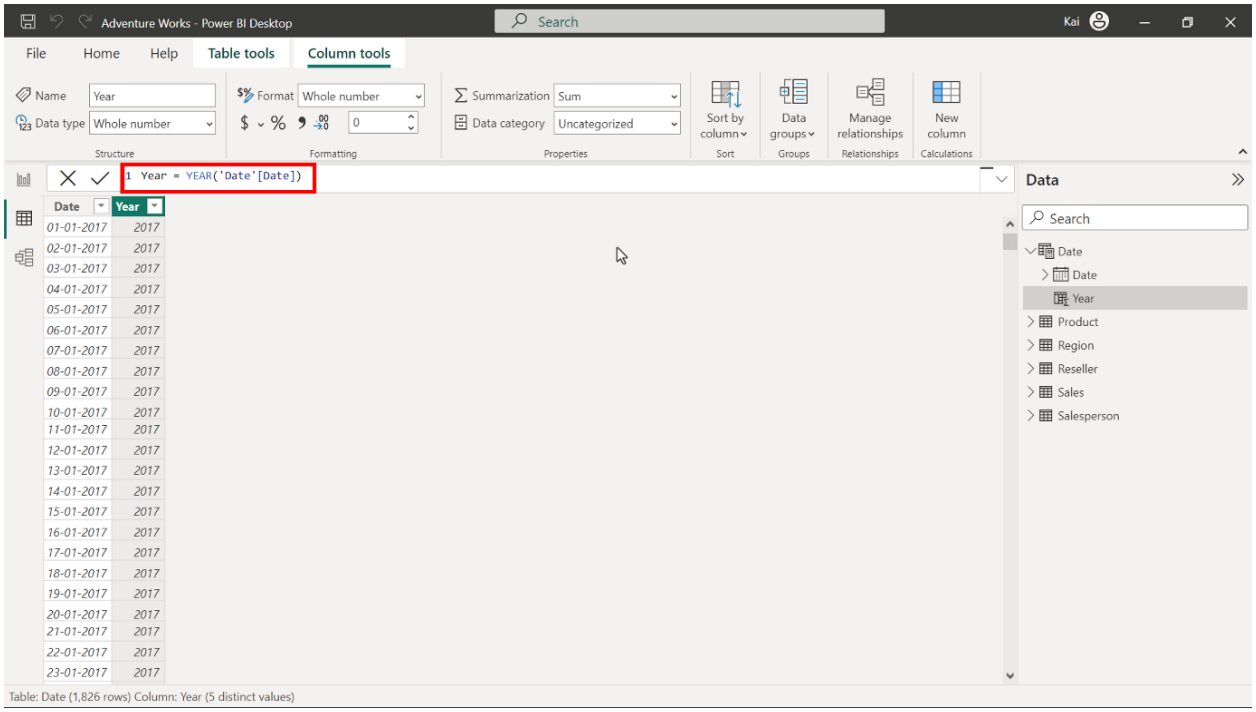
```
Month Number = MONTH ( 'Date'[Date] )
Day of the Week = FORMAT ( WEEKDAY( 'Date'[Date] ), "dddd" )
Week Number = WEEKNUM ( 'Date'[Date])
```

- The date-related functions in the above DAX formulas, like **YEAR**, **MONTH**, **WEEKNUM**, **WEEKDAY** extract the relevant information from the date column of the table.



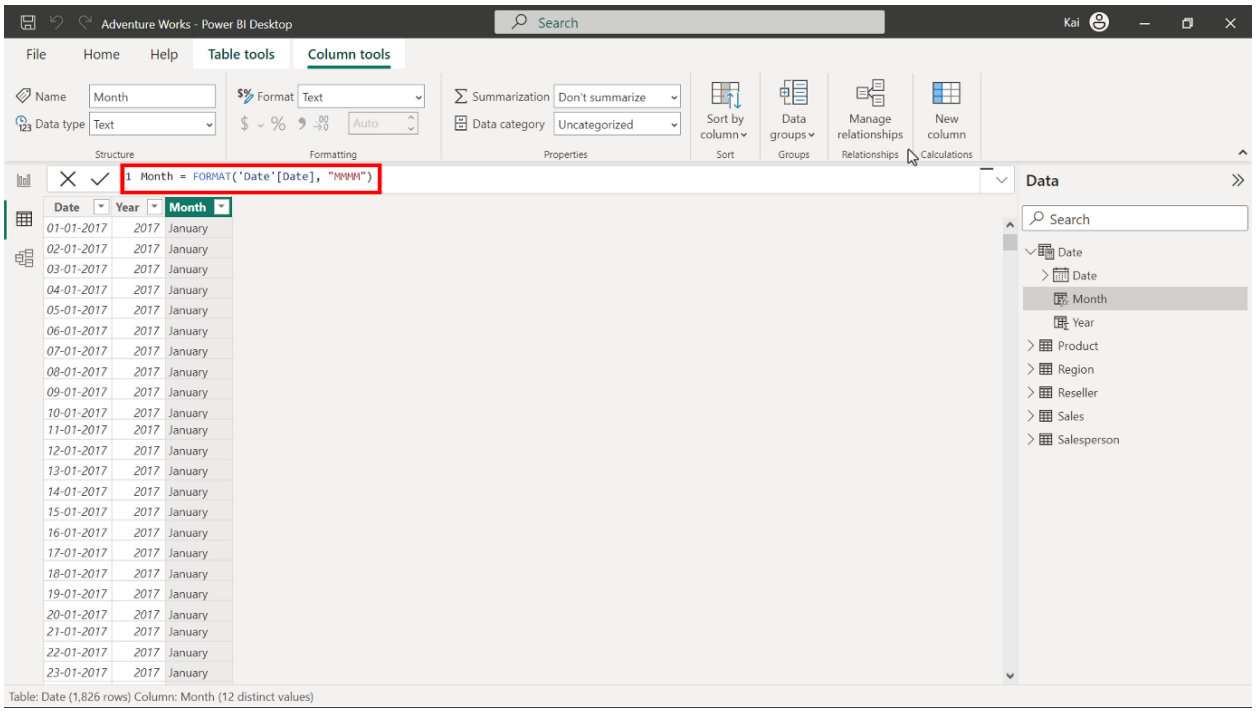
Copy and paste the following code into the formula bar to add the **YEAR** data.

```
1
Year =  YEAR ( 'Date'[Date] )
```



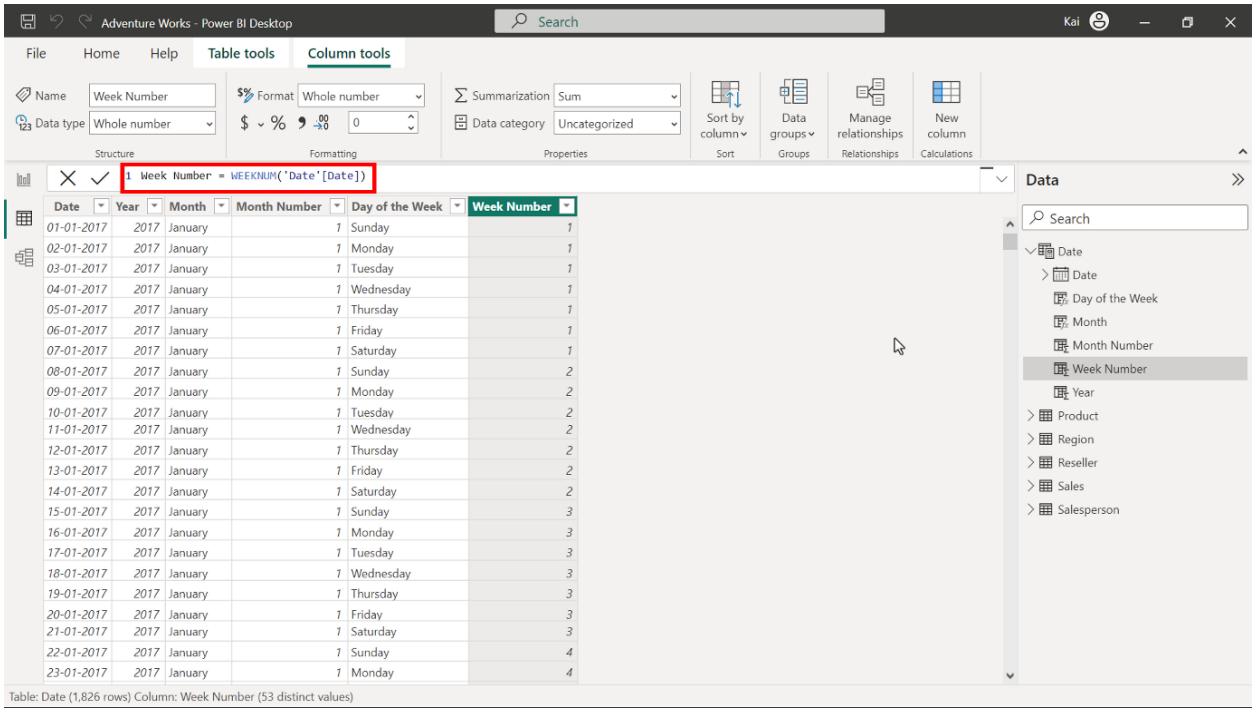
Copy and paste the following code into the formula bar to add the **MONTH** data.

```
1
Month = FORMAT ( 'Date'[Date], "MMMM" )
```

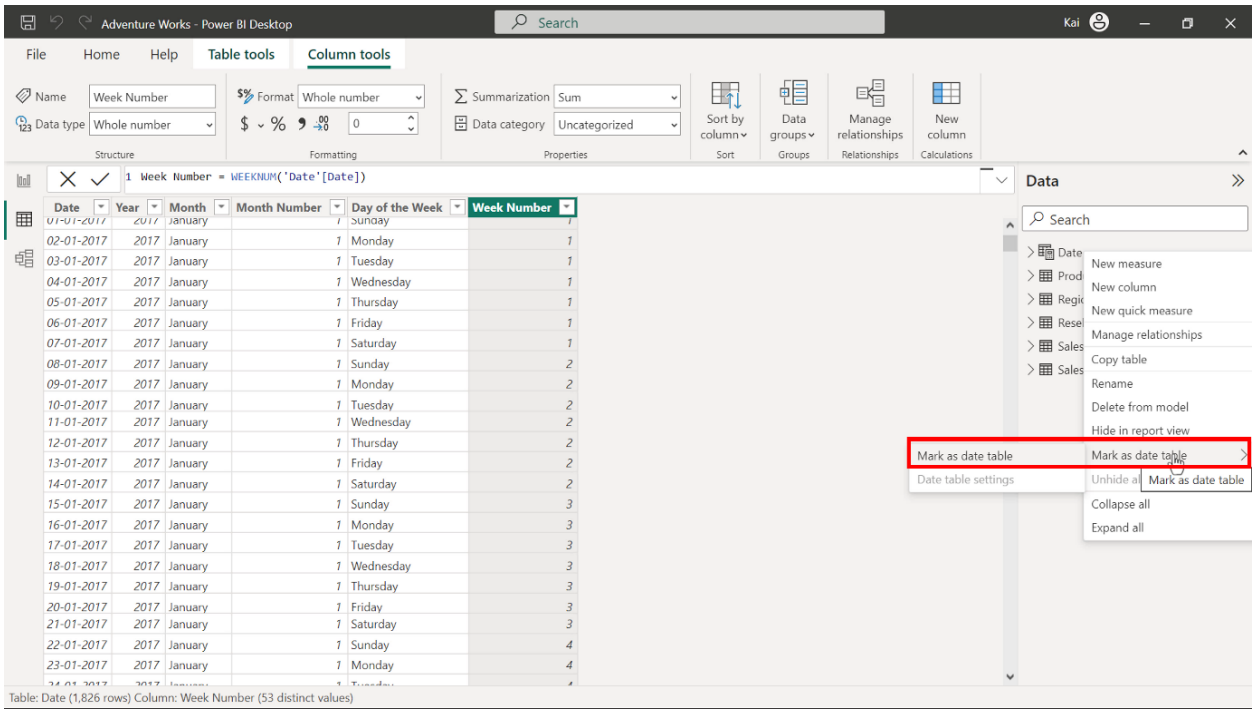


Copy and paste the following code into the formula bar to add the **DAY OF THE WEEK** data.

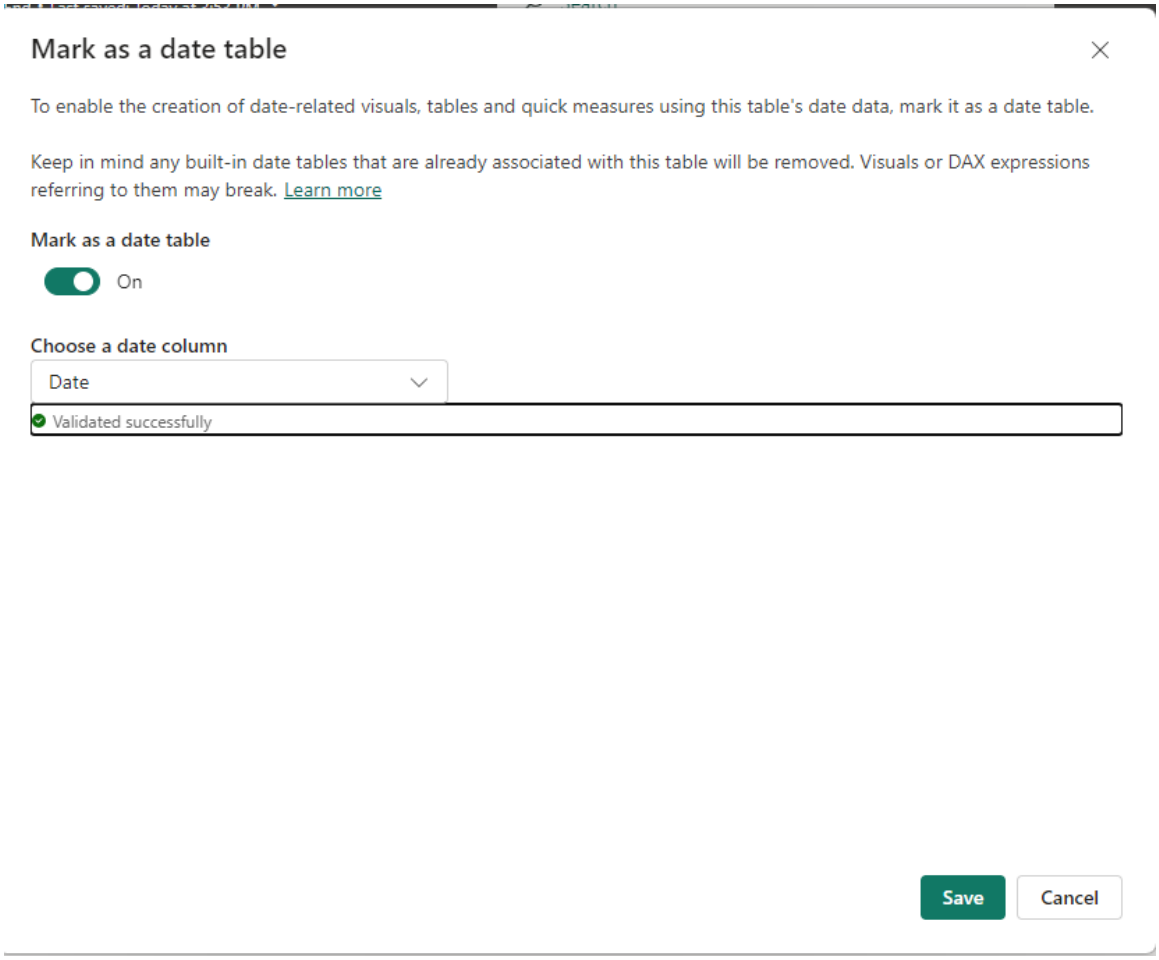
```
1
Day of the Week = FORMAT ( WEEKDAY( 'Date'[Date] ), "dddd" )
```



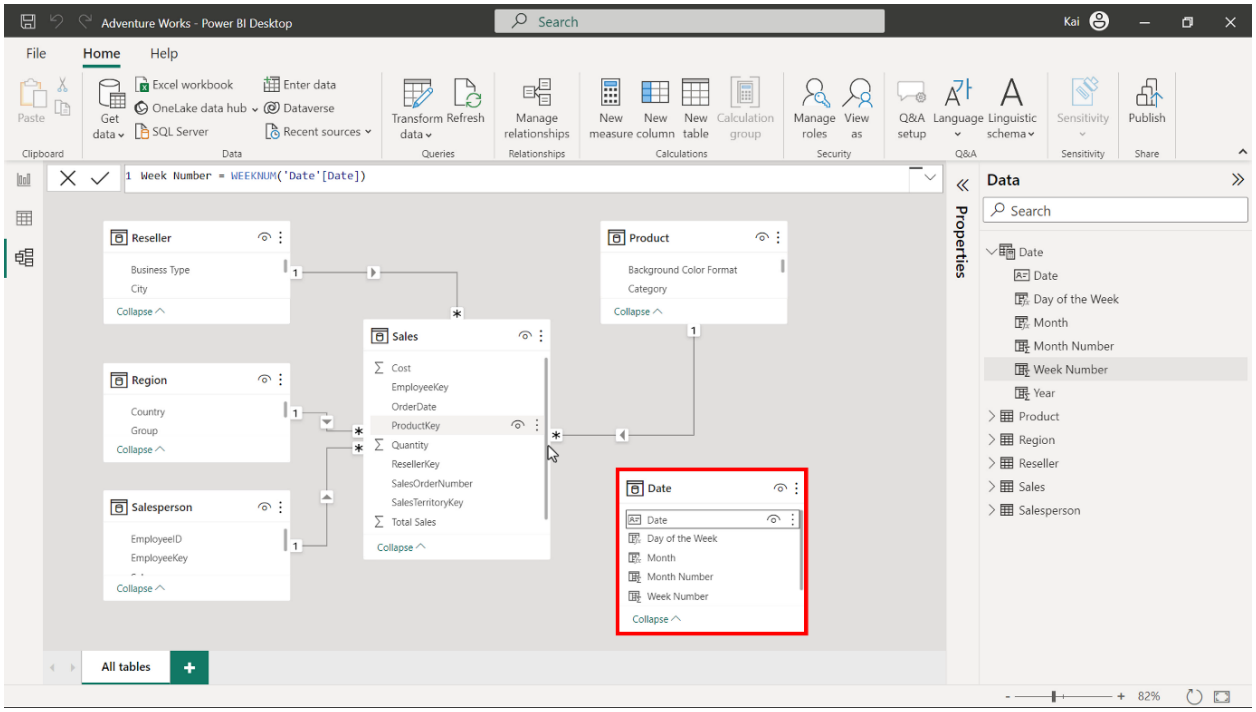
1. Next, mark the newly created **date** dimension table as a date table. Select the ellipses on the right side of the date table and select **Mark as date table** from the drop-down list of options. This opens a dialog box that states **Mark as date table**.



Select the **Date** option from the **Date** column drop-down menu. Once you select the date column, it displays a message **Validate successfully**. Select **OK**.



1. Next, you must establish a relationship between the **Date** and the **Fact** table in the data model. A **Date** dimension table is ready for analysis and reporting in your data model.



Save your project

Exercise 21: Set up a common date table Using M Language

1. Use file **21 Set up a common date table Using M Language.pbix**
2. Let us do the same thing of the previous exercise but this time using M Language in Power Query.
3. Select Home ➔ Transform Data to Open Power Quey.
4. In Power Query Home ➔ New Source.
5. Select Blank Query
6. Enter the following M language code :

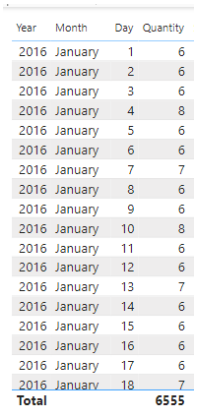
```
= List.Dates(#date(2017,01,01),

365*5,

#duration(1,0,0,0))
```

7. Power BI generate a list of Dates.
8. This date must be converted to a common date table.
9. Select Transform tab ➔ To table.
10. The list is converted to table
11. Rename the column to Date.
12. Change the column data type to date.
13. Now you need to populate the table with the related columns.

14. From the Add column tab
15. Click the Date button and add the following columns
16. Year , Month , Name of Month , Name of Day,Week of Year
17. Rename the query as Date
18. Select Close and Apply
19. Go to data model
20. Mark the Date table as Date table as you did in Exercise 20
21. Create the relationship between Date and Sales table like you did in Exercise 20.

Exercise 22: Calculating MTD , QTD , and YTD

1. User file 22 YTD MTD QTD Functions START.pbix
2. This is a Company Saling Clothes.



Year	Month	Day	Quantity
2016	January	1	6
2016	January	2	6
2016	January	3	6
2016	January	4	8
2016	January	5	6
2016	January	6	6
2016	January	7	7
2016	January	8	6
2016	January	9	6
2016	January	10	8
2016	January	11	6
2016	January	12	6
2016	January	13	7
2016	January	14	6
2016	January	15	6
2016	January	16	6
2016	January	17	6
2016	January	18	7
Total			**6555**

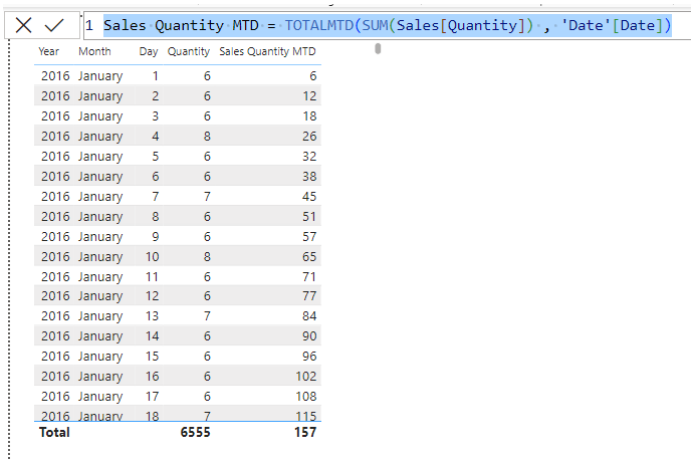
3. You have created a table showing your daily sales
4.
5. It wants to calculate the Quantity they have sold:
    a. from the Start of Year till Date.
    b. from the Start of Month till Date.
    c. form the start of Quarter till Date.

Month to Date (MTD)

6. Create a measure **Sales Quantity MTD**

```
Sales Quantity MTD = TOTALMTD(SUM(Sales[Quantity]) , 'Date'[Date])
```

7. Add the new measure to your table and see the result.



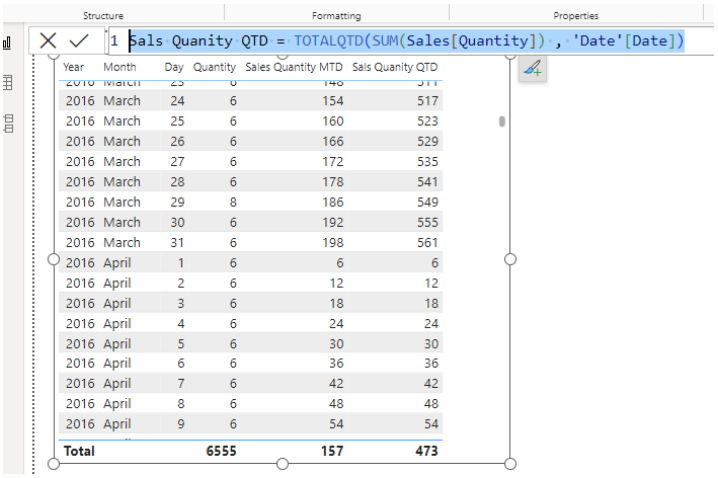
Year	Month	Day	Quantity	Sales Quantity MTD
2016	January	1	6	6
2016	January	2	6	12
2016	January	3	6	18
2016	January	4	8	26
2016	January	5	6	32
2016	January	6	6	38
2016	January	7	7	45
2016	January	8	6	51
2016	January	9	6	57
2016	January	10	8	65
2016	January	11	6	71
2016	January	12	6	77
2016	January	13	7	84
2016	January	14	6	90
2016	January	15	6	96
2016	January	16	6	102
2016	January	17	6	108
2016	January	18	7	115
Total			**6555**	**157**

8. Notice the quantity total is cumulative from the start of month to every day.
9. When the month ends, it starts to count from the beginning of month.

Quarter to Date (QTD)

10. Create a new measure **Sales Quantity QTD**

```
Sals Quanity QTD = TOTALQTD(SUM(Sales[Quantity]) , 'Date'[Date])
```



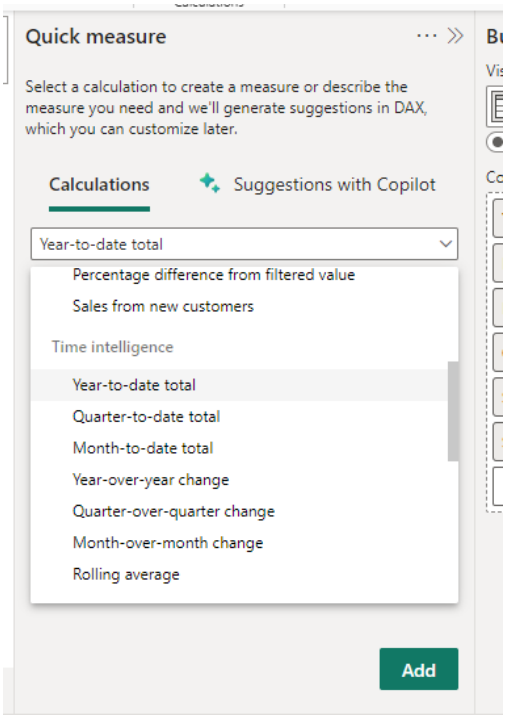
Year	Month	Day	Quantity	Sales Quantity MTD	Sals Quanity QTD
2016	March	24	6	154	517
2016	March	25	6	160	523
2016	March	26	6	166	529
2016	March	27	6	172	535
2016	March	28	6	178	541
2016	March	29	8	186	549
2016	March	30	6	192	555
2016	March	31	6	198	561
2016	April	1	6	6	6
2016	April	2	6	12	12
2016	April	3	6	18	18
2016	April	4	6	24	24
2016	April	5	6	30	30
2016	April	6	6	36	36
2016	April	7	6	42	42
2016	April	8	6	48	48
2016	April	9	6	54	54
Total			**6555**	**157**	**473**

11. Add the measure to your table.
12. Notice it continues to accumulate quantity till the end of March.
13. It starts to count again when April Starts.

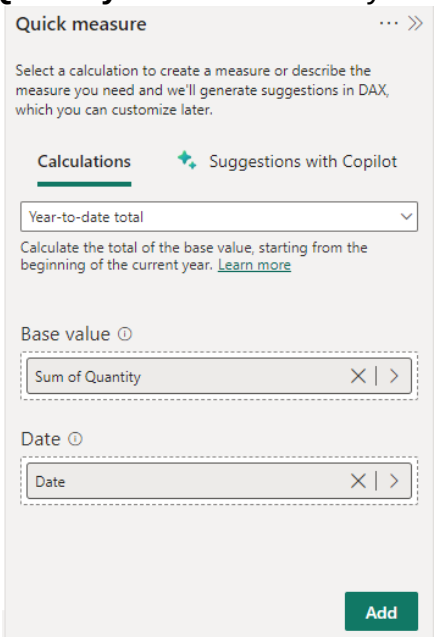
Year to Date (YTD) Using Quick Measure

14. Right click Sales Table and chose **Quick Measure** Option.

15. In Calculation Drop down List select **Year-to-Date Total** under Group **Time intelligence.**



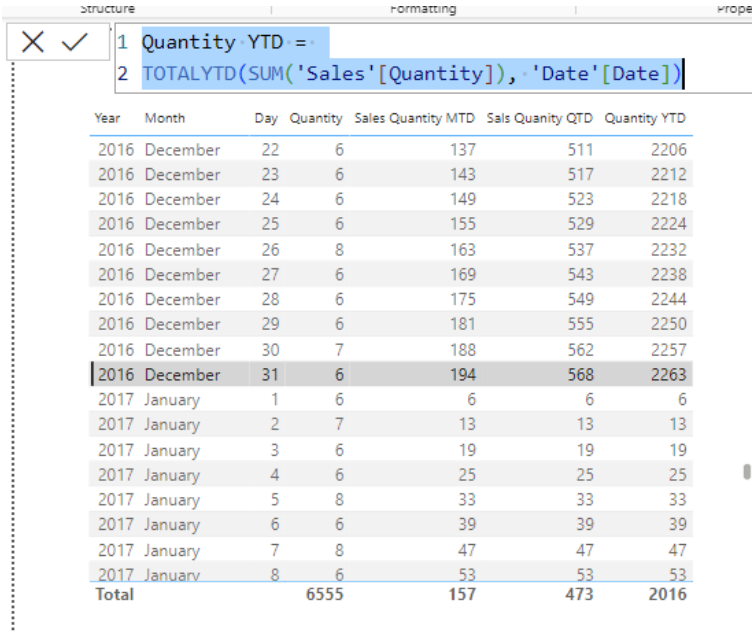
16. Add Quantity to Base Value form Sales table.
17. Add Date from Date table to Date as in the figure below.
18. Click Add.
19. A new Measure **Quantity YTD** is added to your Sales Table



20. You can rename it if you want to.
21. Review its Code:

```
Quantity YTD =
TOTALYTD(SUM('Sales'[Quantity]), 'Date'[Date])
```

22. Add to your table.
23. Notice that it cumulate till end of year this time



Year	Month	Day	Quantity	Sales Quantity MTD	Sals Quanity QTD	Quantity YTD
2016	December	22	6	137	511	2206
2016	December	23	6	143	517	2212
2016	December	24	6	149	523	2218
2016	December	25	6	155	529	2224
2016	December	26	8	163	537	2232
2016	December	27	6	169	543	2238
2016	December	28	6	175	549	2244
2016	December	29	6	181	555	2250
2016	December	30	7	188	562	2257
2016	December	31	6	194	568	2263
2017	January	1	6	6	6	6
2017	January	2	7	13	13	13
2017	January	3	6	19	19	19
2017	January	4	6	25	25	25
2017	January	5	8	33	33	33
2017	January	6	6	39	39	39
2017	January	7	8	47	47	47
2017	January	8	6	53	53	53
Total			6555	157	473	2016

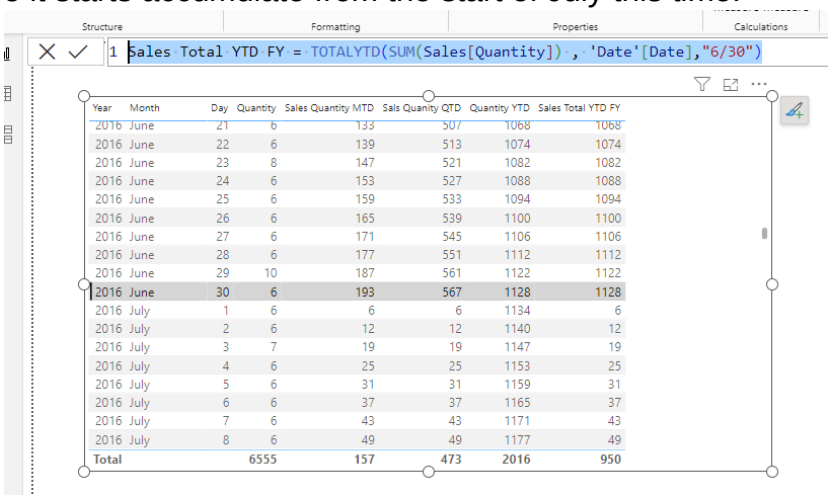
YTD Using Financial Year

24. You have an option to specify the end date of year you want.
25. Let us say our year ends by the end of June.
26. Let us create a new measure **Sales Quantity YTD FY**

```
Sales Total YTD FY = TOTALYTD(SUM(Sales[Quantity]) ,
                     'Date'[Date],"6/30")
```

27. Add Your new measure to the table.

28. Notice it starts accumulate from the start of July this time.



Year	Month	Day	Quantity	Sales Quantity MTD	Sals Quanity QTD	Quantity YTD	Sales Total YTD FY
2016	June	21	6	133	507	1068	1068
2016	June	22	6	139	513	1074	1074
2016	June	23	8	147	521	1082	1082
2016	June	24	6	153	527	1088	1088
2016	June	25	6	159	533	1094	1094
2016	June	26	6	165	539	1100	1100
2016	June	27	6	171	545	1106	1106
2016	June	28	6	177	551	1112	1112
2016	June	29	10	187	561	1122	1122
2016	June	30	6	193	567	1128	1128
2016	July	1	6	6	6	1134	6
2016	July	2	6	12	12	1140	12
2016	July	3	7	19	19	1147	19
2016	July	4	6	25	25	1153	25
2016	July	5	6	31	31	1159	31
2016	July	6	6	37	37	1165	37
2016	July	7	6	43	43	1171	43
2016	July	8	6	49	49	1177	49
Total			6555	157	473	2016	950

Exercise 23: Comparing same period last year

1. Use file **23 SAMEPERIODLASTYEAR Start.pbix**
2. Our Clothes company want to compare sales with the same period last year.
3. They want to compare year to year , Quarter to Quarter and Month to Month and even Day to day.
4. Create new Page.
5. Create a measure Total Quantity

```
Total Quantity = SUM(Sales[Quantity])
```

6. Create a table to show total quantity for every year.

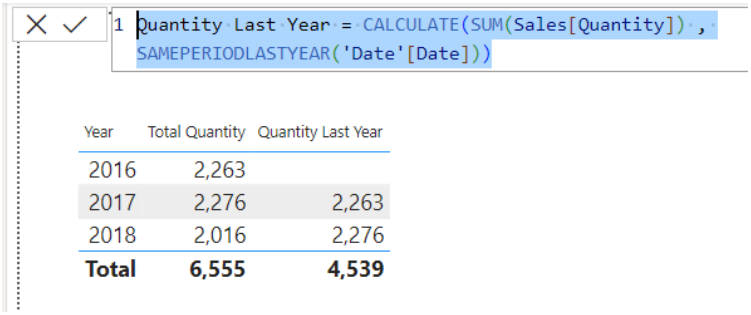
Year	Total Quantity
2016	2263
2017	2276
2018	2016
Total	**6555**

Compare Year to Year

7. Now I want to compare Quantity with last year.
8. Create a new measure **Quantity Last Year**

```
Quantity Last Year = CALCULATE(SUM(Sales[Quantity]) ,
        SAMEPERIODLASTYEAR('Date'[Date]))
```

9. Add the new measure to your table



```
1 Quantity Last Year = CALCULATE(SUM(Sales[Quantity]),
  SAMEPERIODLASTYEAR('Date'[Date]))
```

Year	Total Quantity	Quantity Last Year
2016	2,263	
2017	2,276	2,263
2018	2,016	2,276
Total	**6,555**	**4,539**

Compare Quarters

10. Now add column Quarter to your table after year.
11. You can now compare quarters of the year with last one.

Year	Quarter	Total Quantity	Quantity Last Year
2016	Qtr1	561	
2016	Qtr2	567	
2016	Qtr3	567	
2016	Qtr4	568	
2017	Qtr1	560	561
2017	Qtr2	566	567
2017	Qtr3	575	567
2017	Qtr4	575	568
2018	Qtr1	563	560
2018	Qtr2	503	566
2018	Qtr3	477	575
Total		**6,555**	**4,539**

Compare Months

12. Add Month column after Quarter to compare monhs.

Year	Quarter	Month	Total Quantity	Quantity Last Year
2016	Qtr1	January	193	
2016	Qtr1	February	170	
2016	Qtr1	March	198	
2016	Qtr2	April	183	
2016	Qtr2	May	191	
2016	Qtr2	June	193	
2016	Qtr3	July	189	
2016	Qtr3	August	190	
2016	Qtr3	September	188	
2016	Qtr4	October	189	
2016	Qtr4	November	185	
2016	Qtr4	December	194	
2017	Qtr1	January	194	193
2017	Qtr1	February	171	170
2017	Qtr1	March	195	198
Total			**6,555**	**4,539**

13. Try yourself weeks and days.

Exercise 24: Compare to previous period

1. Use file **24 Copare to Previous Period Start.pbix**
2. Now we want to compare to previous period

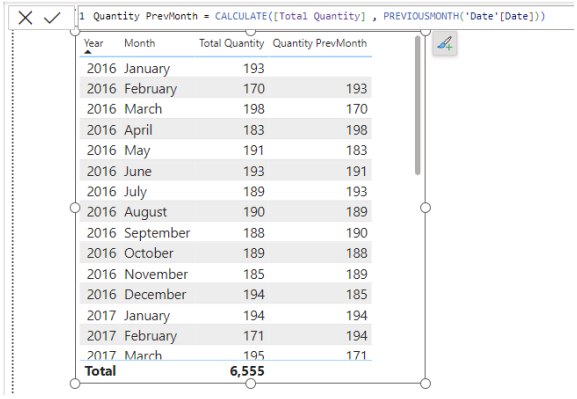
Compare to previous month

3. Create new page

Year	Month	Total Quantity
2016	January	193
2016	February	170
2016	March	198
2016	April	183
2016	May	191
2016	June	193
2016	July	189
2016	August	190
2016	September	188
2016	October	189
2016	November	185
2016	December	194
2017	January	194
2017	February	171
2017	March	195
Total		**6,555**

4. Create new table with **years** and **month** and [Total Quantity] measure.
5. Create **Quantity PrevMoth** measure

```
Quantity PrevMonth = CALCULATE([Total Quantity] , PREVIOUSMONTH('Date'[Date]))
```



6. Add your measure to your table.
7. Try yourself **PREVIOUSYEAR** and **PREVIOUSQUARTER** functions.

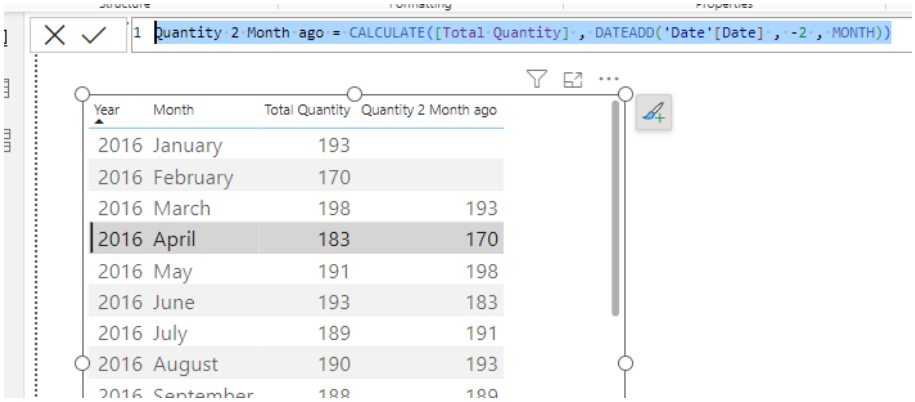
Exercise 25: DATEADD Function

1. Use file **25 DATEADD Function Start.pbix**
2. My company wants to compare Quantity sold by the one they sold 2 months before.
3. Create a table with **year** , **month** and [**Total Quantity**]

Compare to 2 Months Ago

4. Create a measure:

```
Quantity 2 Month ago = CALCULATE([Total Quantity] ,
        DATEADD('Date'[Date] , -2 , MONTH))
```



Year	Month	Total Quantity	Quantity 2 Month ago
2016	January	193	
2016	February	170	
2016	March	198	193
2016	April	183	170
2016	May	191	198
2016	June	193	183
2016	July	189	191
2016	August	190	193
2016	September	188	189

5. Add the new measure to your table
6. Notice it compared to two months ago.
7. You can use it with day, month, quarter , and year intervals.

Knowledge Check

Question 1

Why is it important to have a properly defined **date** column in the data model to execute time intelligence functions?

A. To enable the use of advanced data visualization techniques.
B. To create calculated columns and measures in the data model.
C. To ensure data accuracy and consistency in time-based calculations.

Question 2

Which of the following features of Power BI can you use to add a **date** or common-dimension table to the data model? Select all that apply.

A. Data Analysis Expressions
B. The **Data modeling** tab
C. Power Query and M language
D. Manual data entry

Question 3

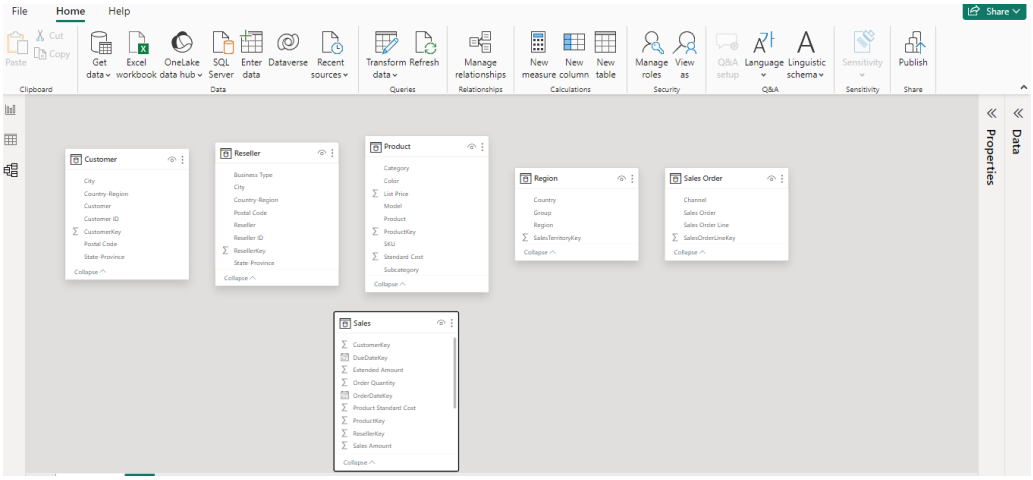
True or False: DAX time intelligence functions can be used with Power BI's autogenerated date/time dimension.

A. True
B. False

Chapter 8: Modeling Data Project

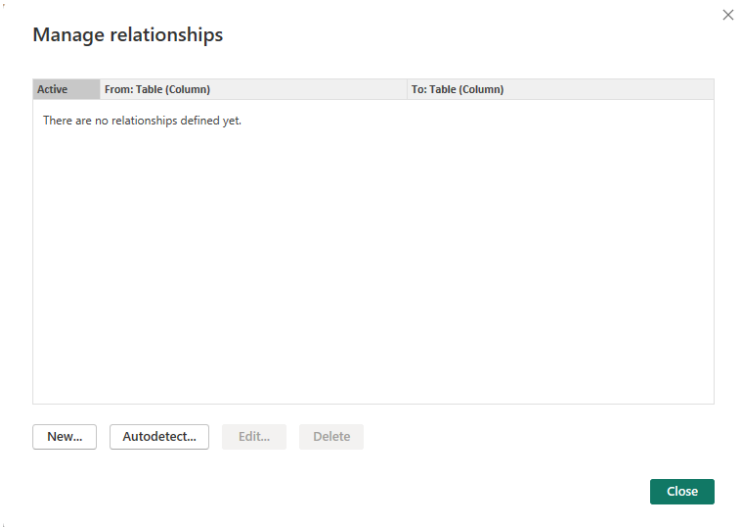
Step 1: Create Tables Relationships

1. The first step in data modeling is defining the initial relationships between the tables. In this task. You will define relationships using different user interface techniques.
2. Select the **model view** icon on the left bar.
3. zoom out to show our tables to make things a little bit more visible using the icon on bottom right corner.
4. Grab the sales table and drag it down. So you can see all the

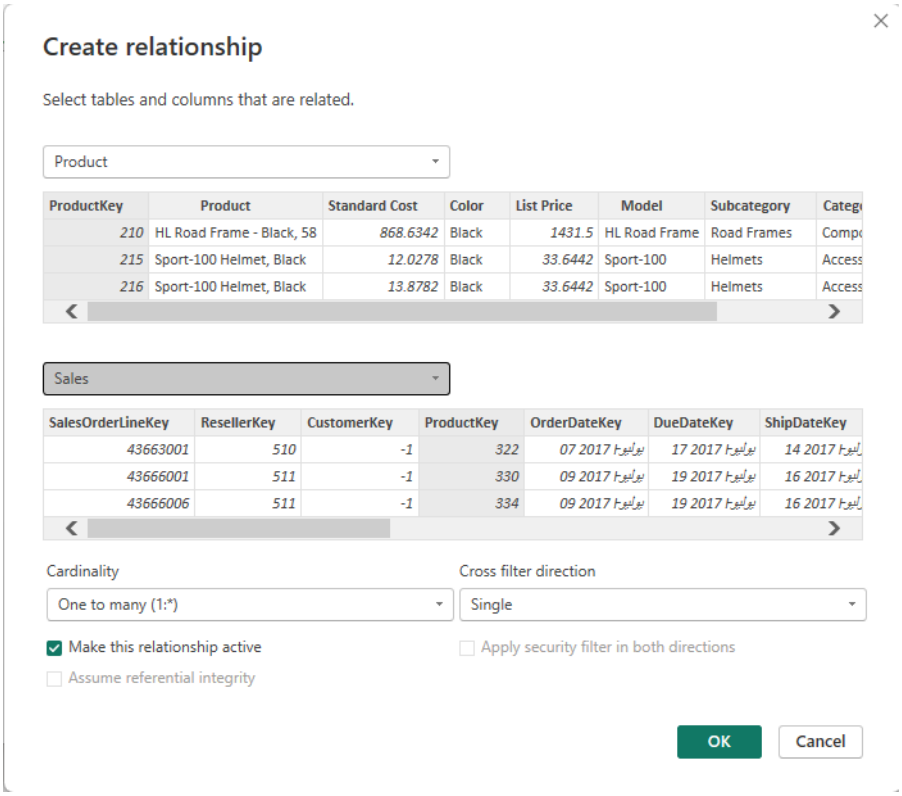


tables that are involved inside the data model.

5. We have **customer reseller**, **product region**, **sales order** and **sales**.
6. There are currently no defined relationships between the tables. This is our task. You are going to manually create the relationships instead of using the auto detect feature.
7. First select the **manage relationship** button on the ribbon.
8. This screen currently shows all of our relationships between the tables. And of course, we don't have any yet because we're going to manually.



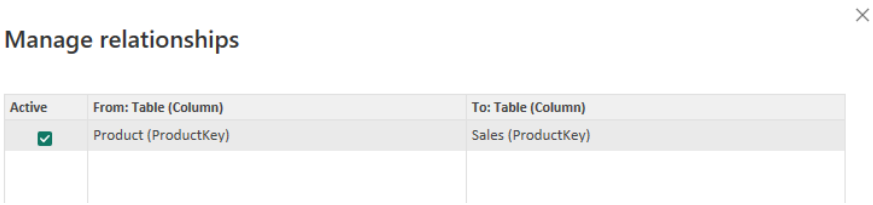
9. **Do not** select the **auto detect button**, select the **new button**.
10. And for the table, we're going to select **product**. And for the second table, we are going to select **sales**
11. next, confirm their **cardinality** and **cross filter direction**.
12. The cardinality has **one to many** selected. This is correct. The product is used in many sales.



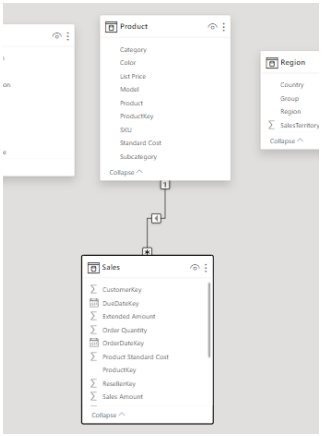
13. The **single filter** direction means that the filter propagates from the one side to the mini side. In this case, it means filters applied to the

product table will **propagate** to the **sales** table but not in the opposite direction.

14. Select the **OK** button, you'll now see in the list our newly created **relationship**.



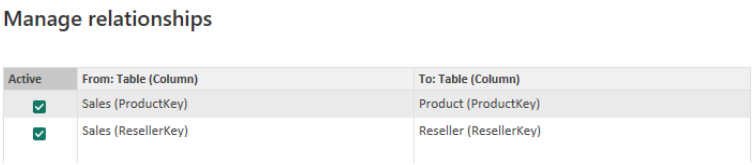
15. If you need to **edit**, you can always select the edit button.
16. Now select **close**.
17. You will now see a **line** from the **product** table to the **sales** table. The product shows a one with the line including an arrow, which is the cross-filter direction to the sales table that shows the star,



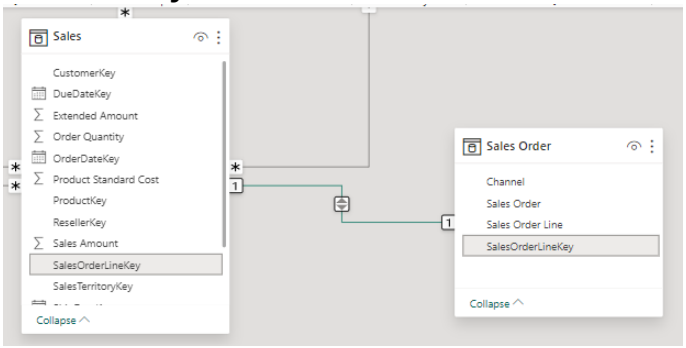
which is the many sideof the cardinality.

18. Next, you will create the same type of relationship using the **drag and drop capability** by dragging a field with the same name from one table to another.
19. The relationship will automatically be defined.
20. At the **reseller** table, drag and drop the **reseller key** from the reseller table onto the **sales reseller key** field.
21. Now the line appears to show the relationship was created.

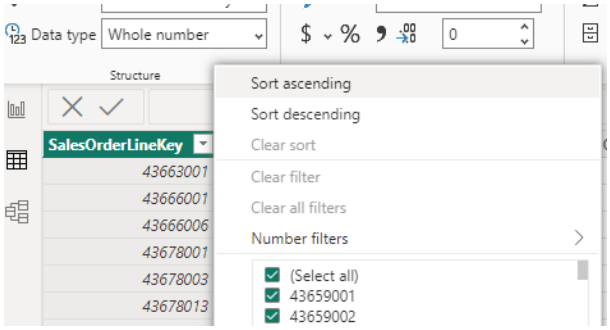
22. You can check with the **managed relationships button** on the ribbon to see that we now have two relationships.



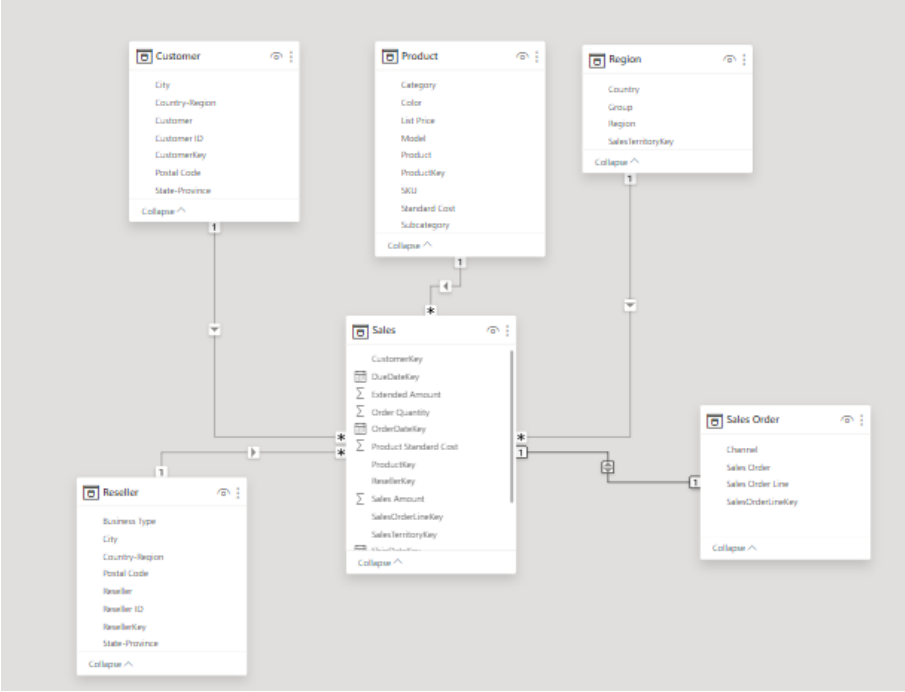
23. Next, create a relationship from **region** to **sales** by using the **sales territory key** and inside sales, we have the **sales territory key** too.
24. Define a relationship from the **customer**, **customer key** to the sales **customer key**.
25. Define the relationship between **sales order**, and **sales**, using field **sales order line key**.



26. The line between the two tables show a **1 to 1 relationship**.
27. when creating the relationship. Power bi I detected a 1 to 1 relationship. This would also set the **cross filter** to **both**.
28. You need to explore the **data** to confirm this is correct.
29. Select the table view button on the left menu. Then select the **sales** table. We can now view the data with inside the **sales** table.
30. Now sort the **sales order line key** ascending by selecting the arrow next to the column name.



31. Do we see any **duplicate numbers**?
32. I don't, this table seems to have **unique numbers** that explains the one side of the relationship.
33. Now on the **table view**, select the **sales order** table, let's go ahead and, and sort the **sales order line key** column.
34. Once again, let's take a look for any duplicate numbers. The table seems to have **unique numbers**.
35. So once again, this explains why in the relationship we had a one on that side.
36. So, everything looks good. Let's select our model view on the left menu and you have now defined the base relationships between all the tables. You are ready for the next task.



Step 2: Configure Tables

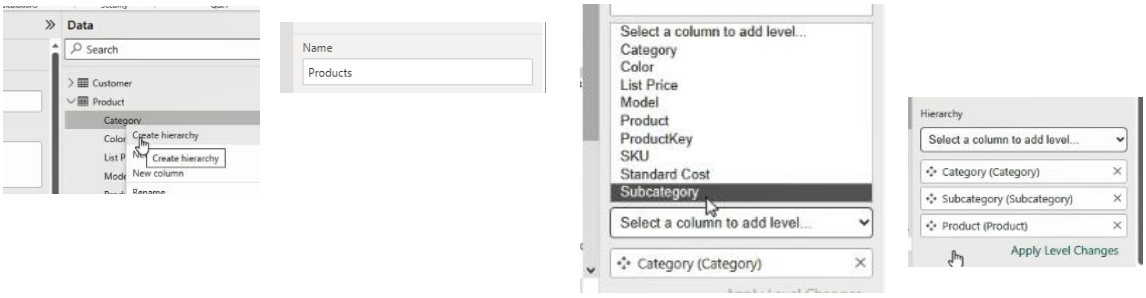
1. Once you have defined the basic table relationships, your next task is to configure table specific properties.
2. This includes **field formatting** and **field groupings.**
3. Select the model view and expand the **properties** and **data** panes.

Configure Product table

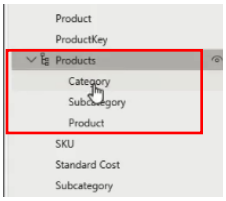
4. Select the **product** table.

Creating Hierarchy

5. You'll be creating a **hierarchy** on **category subcategory** and **product**:
6. Right Click on the **category** column and select **create hierarchy.**
7. In the **properties pane.**
    a. For the **name**, enter **products**,
    b. Scroll to the bottom, you'll see an area to build a **field list**.
    c. select a column to add: select **subcategory**. And then select **product**.
    d. The next most important thing to do, select, **apply level changes**.

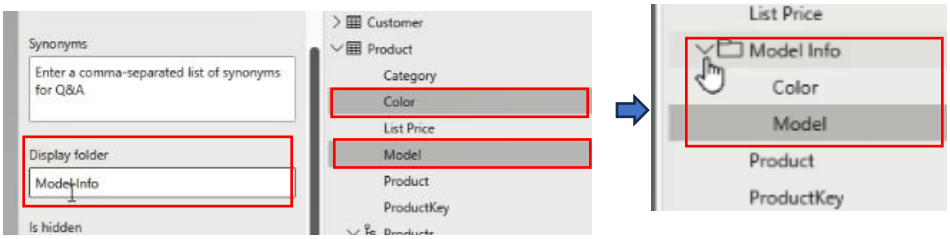


8. You can now look at the hierarchy inside the data pane.
9. You have products categories, subcategory and then the product.



Organize Columns into Display Folder

10. Next, we will organize columns into a display folder.
11. select the **model** field and **products** and while holding the **control key**, select the **color** field.



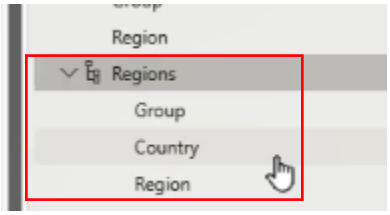
12. In the **properties** pane and the **display folder** field enter "**model info"**.
13. If you leave that field, it will save and under the folder, we now have the model info, we have the color and the model.

Configure Region Table

Create Hierarchy Regions

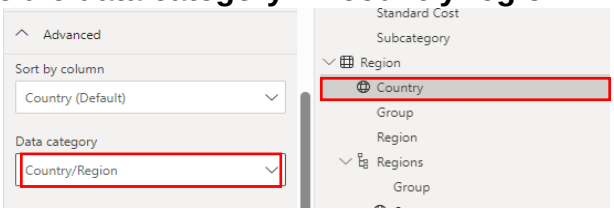
14. Next, you need to configure the region table,

15. Select the **region** table.
16. Create a **hierarchy** on **group country** and **region**.
17. Don't forget to select **apply level changes** and you should see our regions listed in the table.



Data Category

18. Next, select the **country** field **that is not part of our hierarchy**.
19. And then with inside the **properties**, scroll down to the **advanced** options and change the **data category** to "**country region**".



Configure Reseller Table Hierarchies:

20. Next, we will configure the **reseller** table,
21. Select the reseller table.
22. Create a **hierarchy** on **business type** and **reseller** columns.
23. For the name type **resellers**.
24. apply level changes. And once again, you should see the hierarchy in the

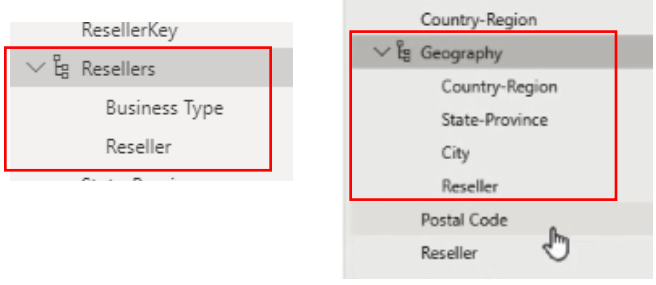


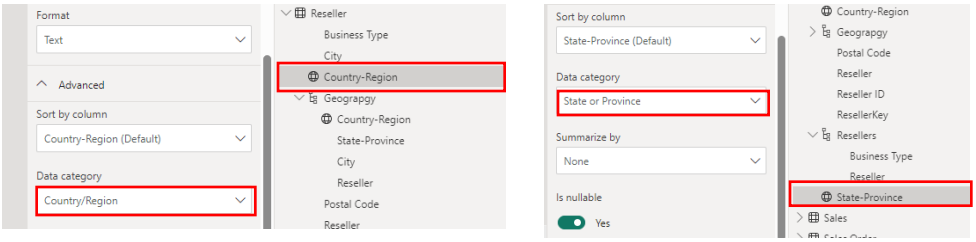
table.

25. Create a second hierarchy in the reseller table. This time on **country region**, **state**, **province**, **city** and **reseller** columns.

26. For name type “**geography”.**
27. **Apply level changes** and you should see **geography** and the reseller table.

Data Category

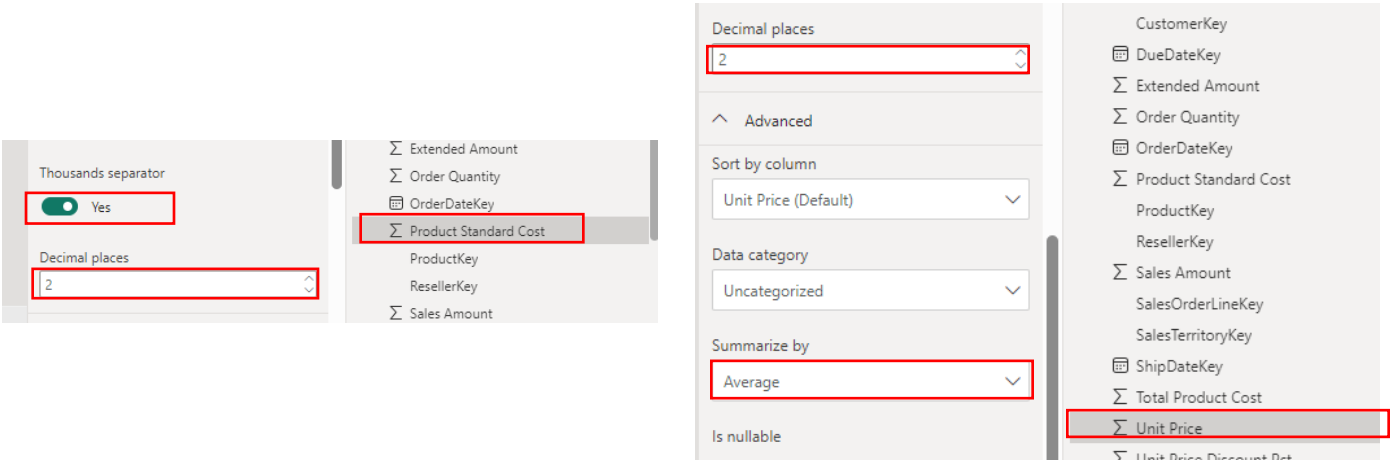
28. Next we'll set the **data category** for **country region, state province** and **city**
29. select the **country** region field that's **not part of** the **geography hierarchy**.
30. And then in our properties select **advanced** and for **data category**, select **country region**.



31. Next select **state province**, which is not part of our geography hierarchy. And for the data category, select **state or province**
32. and then our final field will be **city** once again, not part of our hierarchy for data category. Configure as **city**.

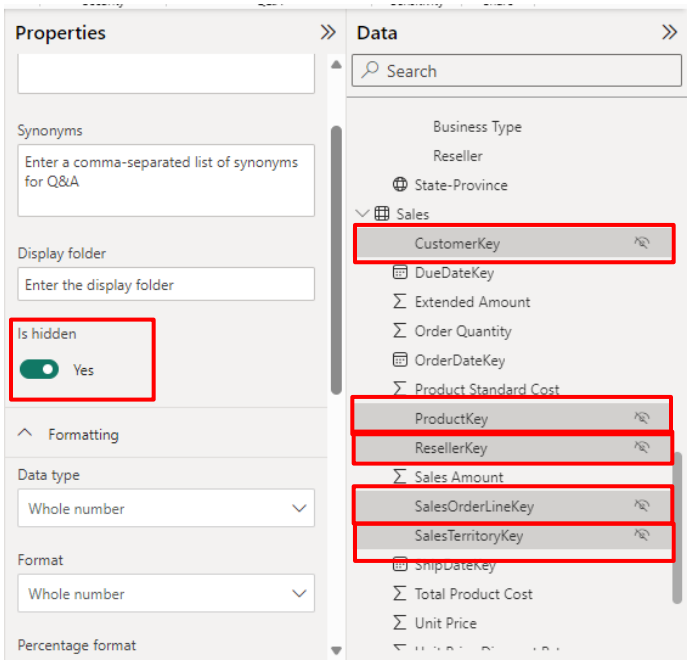
Configure Sales Table

33. Select the sales table.
34. select the **product standard cost** column on the **properties** tab slide the 1000 separator to **yes**.
35. Now select the **unit price** column and inside properties set the **decimal places** to 2.
36. In the **advanced** section set, the **summarize by** to **average**.



Bulck Update Columns

37. as you could see if you had to individually update every column for the same setting. This could take a while.
38. Fortunately, you can **bulk update columns**.
39. For example, you can hide all the **keys** because they provide no other value than linking the tables together.
40. To make this a little bit easier, expand all of the tables just so we can see.
41. select the **customer key**. Then while holding down the **control key**,select the **product key**, **the sales territory key**, continue holding down the control key and the **reseller key**,then the **sales order line key** with inside **sales** table and then also the **sales territory key** with inside sales table, the **product key**, the **reseller key** and the **customer keys**.
42. Finally, down in **sales order** table, we should have our **sales order line key**.
43. In **properties** pane we will select **is hidden to Yes**.



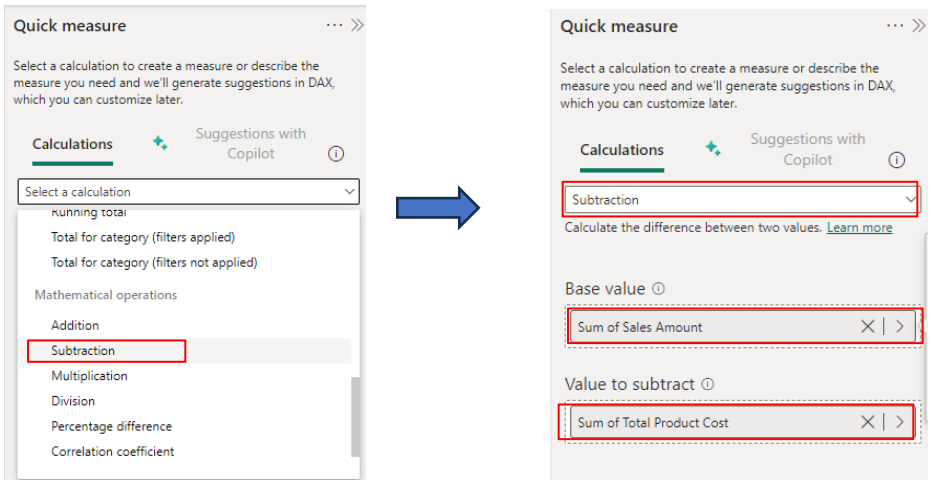
44. Now this has updated all of those columns to be hidden.
45. You are ready for the next task but remember to also save your file.

Step 3: Create Quick Measures

1. To quickly create calculated fields without the need for writing DAX calculations, you can create quick measures.
2. in this task, you will create quick measures to enhance the data.
3. Select the **table view**,
4. select the **sales** table and expand it.
5. Right Click **sales** table, you'll see an option for **new quick measure**.

6. New quick measure is also available to you with inside the **tables tools**, ribbon menu.

Quick Subtraction Measure

7. Select **subtraction** for the **base value**
8. 

9. drag and dropped the **sales amount** field from the sales table
10. for the **value to subtract** drag and drop the **total product cost** field from the sales table.
11. Then select **add.**
12. As you can see the measure has been added to our sales table.
13. Looks like it has a very long name. Let's rename this measure to **profit**.
14. So right mouse on the column, select **rename** and type in **profit**.
15. Notice the DAX code in the function bar for the new measure

```
Profit =
SUM('Sales'[Sales Amount]) - SUM('Sales'[Total Product Cost])
```

Quick Division Measure

16. Our calculation is going to be division.
17. So select **division** for the **mathematic operations**
18. for the **numerator** drag and dropped the **profit** field from the **sales** table and for the **denominator** drag and dropped the **sales of Amount** field.
19. Then select **add.**
20. as you can see our new measure has been created with a very long descriptive name. rename it to **profit margin**.
21. Notice the DAX code of the new quick measure

```
Profit Margin =
DIVIDE([Profit], SUM('Sales'[Sales Amount]))
```

Format measures

22. Now that we have the **profit margin** column, we need to format the field.
23. So, in the ribbon, we have the formatting section set the format to **percentage** and confirm that the decimals are 2.

Test The Measure

24. test our quick measures. Select the report icon on the left menu. This is the report view area.
25. double click on **table** that now drops a table into our canvas
26. from the sales table. Select the check box for **profit**, **profit margin** and **sales amount.**

Sum of Sales Amount	Profit	Profit Margin
109,809,274.20	12,551,366.25	11.43%

27. This shows an overall total for the sales table.
28. I think we need to see a little bit more detail.
29. select the **product** table and then select the check box for **category**.

Category	Sum of Sales Amount	Profit	Profit Margin
Accessories	1,272,057.89	634,467.16	49.88%
Bikes	94,620,526.21	10,515,096.61	11.11%
Clothing	2,117,613.45	368,836.00	17.42%
Components	11,799,076.66	1,032,966.48	8.75%
Total	**109,809,274.20**	**12,551,366.25**	**11.43%**

30. Now we have the profit, profit margin sales amount for each category.
31. You are ready for the next task. Don't forget to press the save button first.

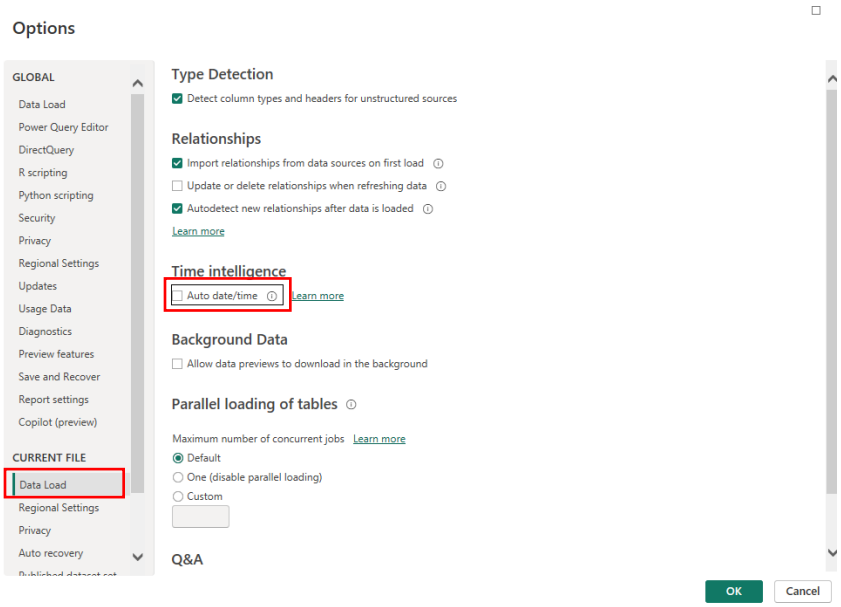
Step 4: Create a Date Dimension Table

1. Common analysis scenarios require date related groupings.
2. An example of this is a custom **Fiscal year** instead of using a **calendar year**.
3. In this task, you will create a date table for use in your data.

Turn off the system auto date functionality

4. From the File select **file➔ options and settings ➔ options**

5. in the section, **current file**, select **data load**, then **uncheck** the checkbox



auto detect time and then select **OK**.

Create Date Table Using DAX

6. Select the **table view** from the left menu.
7. From the ribbon, select **new table**
8. in the formula bar, you will enter a DAX calculation.
9. Enter the following calculation:

```
Date = CALENDARAUTO(6)
```

10. Press Enter the function generated a new table named **Date.**
11. The (**6**) is an extra parameter that is used to define **the last month of the year**. In this case, **June**.
12. by default, it is **12** or **December.**
13. take a minute to review the data that was generated.
14. Format the Date Column as **Date** only

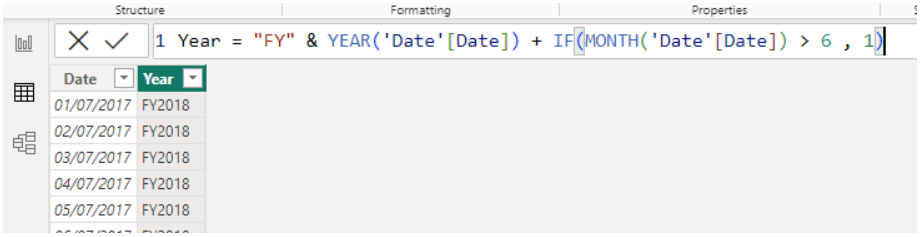
Add More Columns

15. Now you need to add in a few extra columns to enable filtering and grouping for different time periods.

Create Year Column

16. On the **table tools** ribbon select **new column**.
17. You will see that the formula bar has a new formula for us to enter in the calculation for the column. Enter the formula:

```
Year = "FY" & YEAR('Date'[Date]) +
  IF(MONTH('Date'[Date]) > 6 , 1)
```



Create Quarter Column

18. Create the Quarter Column using this formula:

```
Quarter = 'Date'[Year] & " Q" &
    IF(MONTH('Date'[Date]) <=3,3,
    IF(MONTH('Date'[Date]) <=6 ,4,
    IF(MONTH('Date'[Date]) <=9,1,2)))
```

Date	Year	Quarter
01/07/2017	FY2018	FY2018 Q1
02/07/2017	FY2018	FY2018 Q1
03/07/2017	FY2018	FY2018 Q1
04/07/2017	FY2018	FY2018 Q1
05/07/2017	FY2018	FY2018 Q1
06/07/2017	FY2018	FY2018 Q1
07/07/2017	FY2018	FY2018 Q1
08/07/2017	FY2018	FY2018 Q1

Create a Month Column

19. Create a Month Column with the code

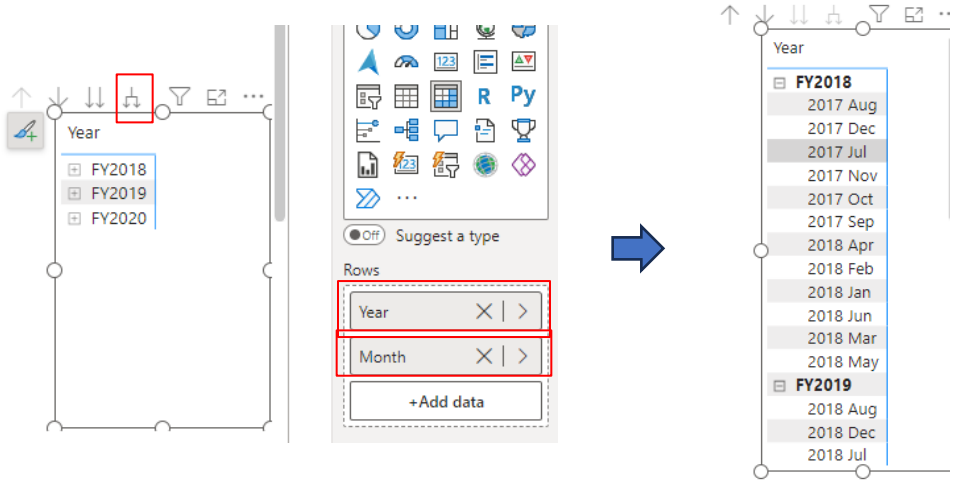
```
Month = FORMAT('Date'[Date] , "yyyy MMM")
```

Date	Year	Quarter	Month
01/07/2017	FY2018	FY2018 Q1	2017 Jul
02/07/2017	FY2018	FY2018 Q1	2017 Jul
03/07/2017	FY2018	FY2018 Q1	2017 Jul
04/07/2017	FY2018	FY2018 Q1	2017 Jul
05/07/2017	FY2018	FY2018 Q1	2017 Jul
06/07/2017	FY2018	FY2018 Q1	2017 Jul
07/07/2017	FY2018	FY2018 Q1	2017 Jul

Test Your Date Table

20. Go to Report view.

21. Create new Page.
22. Add a **Matrix**.
23. Add **Year** and **Month** Field to **Rows**.
24. Click on the Fork Icon to expand the next level (Months)
25.



26. Notice the Month is not sorted well.
27. Let us fix that

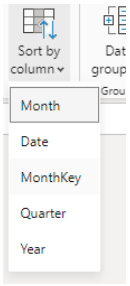
Create MonthKey column

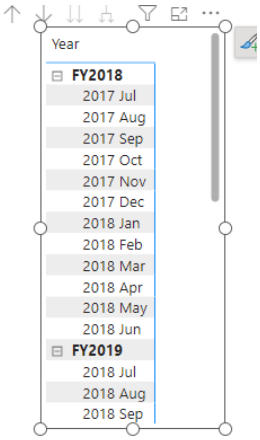
28. Go to **table view**.
29. Select **Date** table.
30. Create new Column **MonthKey** to filter the date on
31. Use the following DAX code:

```
MonthKey = YEAR('Date'[Date]) *100 + MONTH('Date'[Date])
```

Date	Year	Quarter	Month	MonthKey
01/07/2017	FY2018	FY2018 Q1	2017 Jul	201707
02/07/2017	FY2018	FY2018 Q1	2017 Jul	201707
03/07/2017	FY2018	FY2018 Q1	2017 Jul	201707
04/07/2017	FY2018	FY2018 Q1	2017 Jul	201707
05/07/2017	FY2018	FY2018 Q1	2017 Jul	201707
06/07/2017	FY2018	FY2018 Q1	2017 Jul	201707
07/07/2017	FY2018	FY2018 Q1	2017 Jul	201707
08/07/2017	FY2018	FY2018 Q1	2017 Jul	201707
09/07/2017	FY2018	FY2018 Q1	2017 Jul	201707

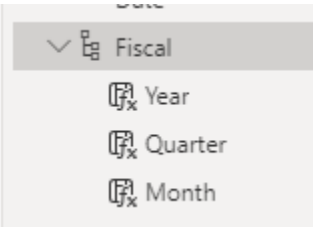
32. Now Select the **Month** column
33. From the ribbon select **Sort by Column**.
34. Select **MonthKey**.
35. Go to Matrix now and it should be updated.
36. Now go to Model View
37. Select **MonthKey** and Hide
38. Select **is Hidden** and make it **Yes** .





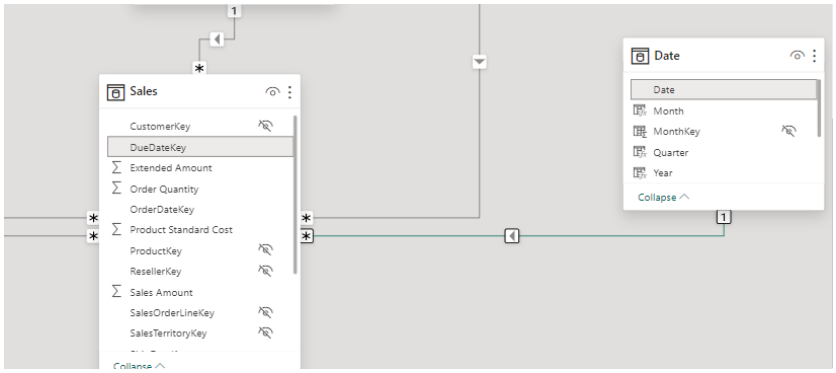
Create Date Hierarchy

39. Select **Year** column and right click and select **Create New Hierarchy.**
40. Make the **Name** : **Fiscal.**
41. Add **Quarter** then **Month**
42. Click Apply Level Changes



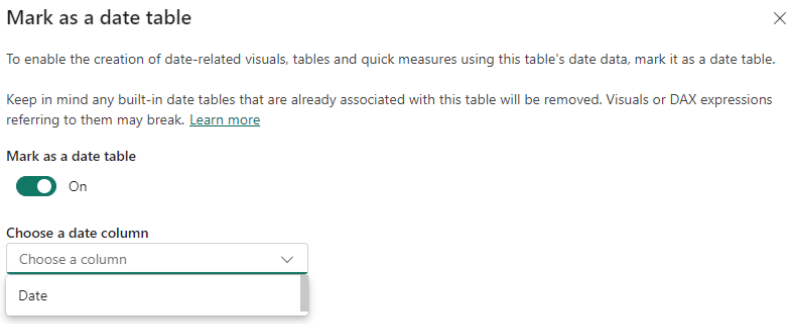
Set Relationship

43. Now set the relationship between Date table and Sales table.
44. Use **Date** field from Date table and **OrderDateKey** from Sales table.



Mark As a Date Table

45. Go to table view.
46. From the ribbon select **Mark as a Date Table**.
47. In the Dialogue box make it **on**.
48. Select **Date** field.
49. Save.



Create Some Measure

50. Select Sales table and create some measures.
51. **Avg Price**:

```
Avg Price = AVERAGE(Sales[Unit Price])
```

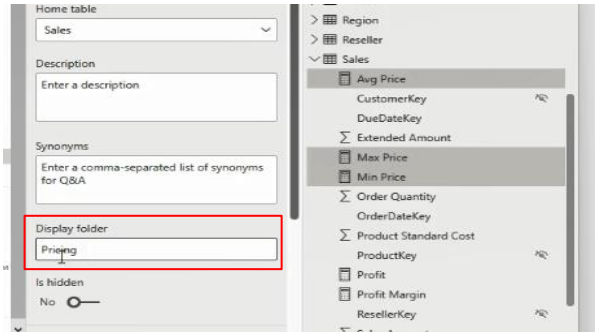
52. **Min Price:**

```
Min Price = MIN(Sales[Unit Price])
```

53. **Max Price:**

```
Max Price = MAX(Sales[Unit Price])
```

54. Go to Model view.
55. Arrange those measures to a Display Folder : **Pricing**



Display Measure in Matrix

56. Go to your **Report View**.

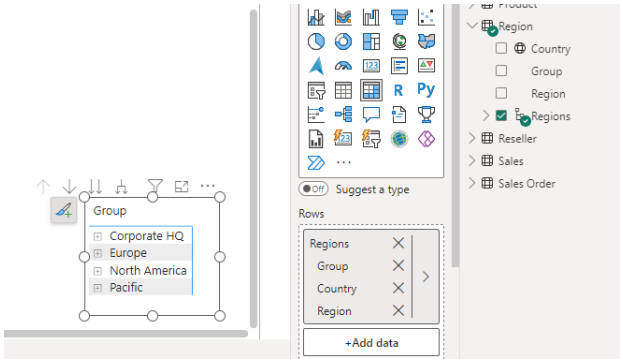
57. Select your **Matrix**.

Year	Avg Price	Max Price	Min Price
FY2018	**1,288.13**	**3,578.27**	**4.75**
2017 Jul	1,785.38	3,578.27	5.19
2017 Aug	1,371.54	3,578.27	5.01
2017 Sep	1,122.39	3,578.27	4.75
2017 Oct	1,381.91	3,578.27	5.19
2017 Nov	1,118.65	3,578.27	5.01
2017 Dec	1,268.60	3,578.27	5.19
2018 Jan	1,735.97	3,578.27	5.19
2018 Feb	1,190.45	3,578.27	5.01
2018 Mar	1,596.34	3,578.27	5.19
2018 Apr	1,672.56	3,578.27	5.19
2018 May	1,198.20	3,578.27	5.01
2018 Jun	1,002.16	3,578.27	4.75
FY2019	**573.74**	**2,443.35**	**2.29**
2018 Jul	689.43	2,443.35	5.01
2018 Aug	474.43	2,443.35	4.75
Total	**465.18**	**3,578.27**	**1.33**

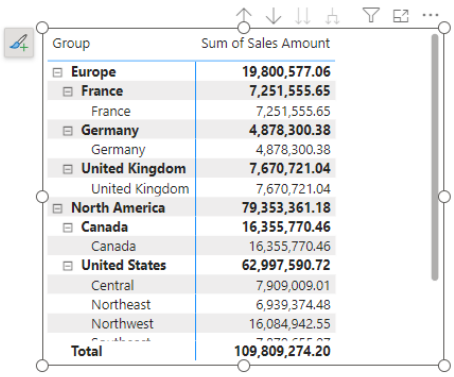
58. Chek the measures **Avg Price**, **Min Price** , and **Max Price** to add to matrix.
59. Save your File.

Step 5: Create DAX Calculation

1. In this task you will create advanced DAX calculations to solve some business problems.
2. Go to **Report View**.
3. Create New Page.
4. Add a **Matrix** to report.
5. From **Region** table drag the **Regions** Hierarchy to Rows.



6. From **Sales** table Check **Sales Amount** to Add in **Values**.



Group	Sum of Sales Amount
Europe	**19,800,577.06**
France	**7,251,555.65**
France	7,251,555.65
Germany	**4,878,300.38**
Germany	4,878,300.38
United Kingdom	**7,670,721.04**
United Kingdom	7,670,721.04
North America	**79,353,361.18**
Canada	**16,355,770.46**
Canada	16,355,770.46
United States	**62,997,590.72**
Central	7,909,009.01
Northeast	6,939,374.48
Northwest	16,084,942.55
Total	**109,809,274.20**

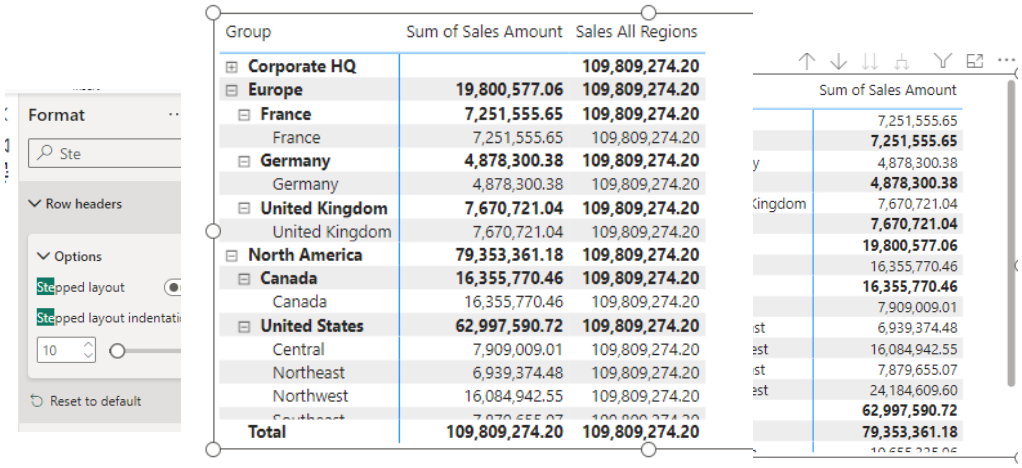
7. Select the **Fork** icon twice to expand the hierarchy.
8. In **format** pane search for **Stepped Layout** and make it **off**

Create Sales % All Region Measure

9. Select Sales table and create new measure:

```
Sales All Regions = CALCULATE(SUM(Sales[Sales Amount]) , REMOVEFILTERS(Region))
```

10. Select **Sales All Region** measure to add to matrix.

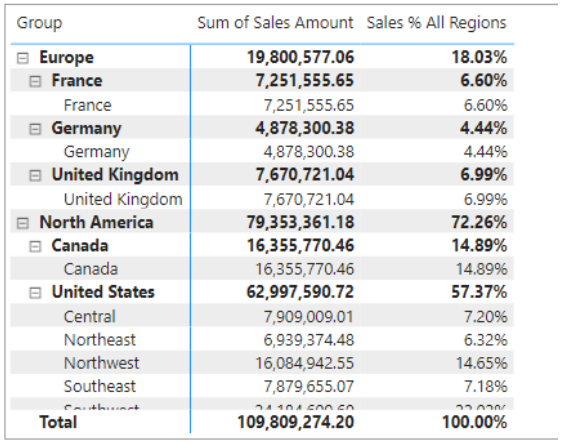


Group	Sum of Sales Amount	Sales All Regions
Corporate HQ		**109,809,274.20**
Europe	**19,800,577.06**	**109,809,274.20**
France	**7,251,555.65**	**109,809,274.20**
France	7,251,555.65	109,809,274.20
Germany	**4,878,300.38**	**109,809,274.20**
Germany	4,878,300.38	109,809,274.20
United Kingdom	**7,670,721.04**	**109,809,274.20**
United Kingdom	7,670,721.04	109,809,274.20
North America	**79,353,361.18**	**109,809,274.20**
Canada	**16,355,770.46**	**109,809,274.20**
Canada	16,355,770.46	109,809,274.20
United States	**62,997,590.72**	**109,809,274.20**
Central	7,909,009.01	109,809,274.20
Northeast	6,939,374.48	109,809,274.20
Northwest	16,084,942.55	109,809,274.20
Total	**109,809,274.20**	**109,809,274.20**

Sum of Sales Amount
7,251,555.65
7,251,555.65
4,878,300.38
4,878,300.38
7,670,721.04
7,670,721.04
19,800,577.06
16,355,770.46
16,355,770.46
7,909,009.01
6,939,374.48
16,084,942.55
7,879,655.07
24,184,609.60
62,997,590.72
79,353,361.18

11. Edit the measure to become **Sales % All Region** measure as follow:
12. Format as **Percentage** with **2** decimals.

13. See your matrix now



Group	Sum of Sales Amount	Sales % All Regions
Europe	**19,800,577.06**	**18.03%**
France	**7,251,555.65**	**6.60%**
France	7,251,555.65	6.60%
Germany	**4,878,300.38**	**4.44%**
Germany	4,878,300.38	4.44%
United Kingdom	**7,670,721.04**	**6.99%**
United Kingdom	7,670,721.04	6.99%
North America	**79,353,361.18**	**72.26%**
Canada	**16,355,770.46**	**14.89%**
Canada	16,355,770.46	14.89%
United States	**62,997,590.72**	**57.37%**
Central	7,909,009.01	7.20%
Northeast	6,939,374.48	6.32%
Northwest	16,084,942.55	14.65%
Southeast	7,879,655.07	7.18%
Total	**109,809,274.20**	**100.00%**

Create Sales YTD Measure

14. Go back to matrix of years in Page 2
15. Create new measure:

```
Sales YTD = TOTALYTD(SUM(Sales[Sales Amount]) ,
'Date'[Date],"30-6")
```

16. Add to your matrix.

Year	Avg Price	Max Price	Min Price	Sales YTD
FY2018	**1,288.13**	**3,578.27**	**4.75**	**23,348,493.12**
2017 Jul	1,785.38	3,578.27	5.19	919,251.52
2017 Aug	1,371.54	3,578.27	5.01	2,823,398.53
2017 Sep	1,122.39	3,578.27	4.75	5,187,522.85
2017 Oct	1,381.91	3,578.27	5.19	6,413,754.83
2017 Nov	1,118.65	3,578.27	5.01	9,528,026.83
2017 Dec	1,268.60	3,578.27	5.19	11,746,917.19
2018 Jan	1,735.97	3,578.27	5.19	12,902,257.51
2018 Feb	1,190.45	3,578.27	5.01	15,863,641.93
2018 Mar	1,596.34	3,578.27	5.19	17,634,108.75
2018 Apr	1,672.56	3,578.27	5.19	19,034,705.14
2018 May	1,198.20	3,578.27	5.01	21,326,934.25
2018 Jun	1,002.16	3,578.27	4.75	23,348,493.12
FY2019	**573.74**	**2,443.35**	**2.29**	**33,636,168.52**
2018 Jul	689.43	2,443.35	5.01	2,304,212.57
2018 Aug	474.43	2,443.35	4.75	5,941,043.44
Total	**465.18**	**3,578.27**	**1.33**	**52,824,612.56**

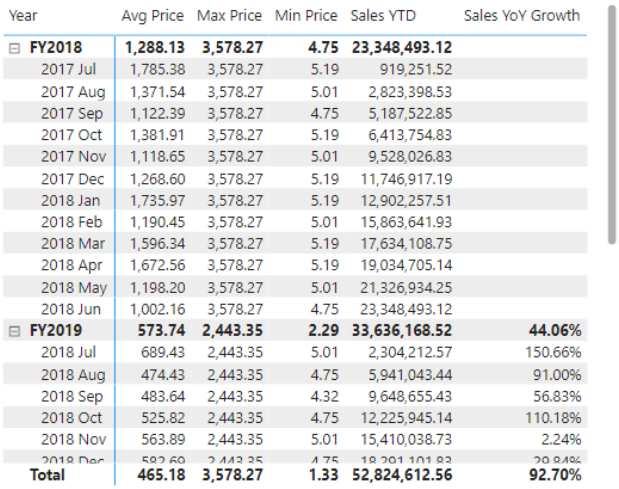
Create Sales YoY Growth

17. Create the measure

```
Sales YoY Growth =
    Var TotalSales =SUM(Sales[Sales Amount])
    var SalesPriorYear =
        CALCULATE(SUM(Sales[Sales Amount]) ,
        PARALLELPERIOD('Date'[Date],-12,MONTH))
    RETURN
```

```
DIVIDE((TotalSales -SalesPriorYear), SalesPriorYear)
```

18. Format as Percentage with 2 decimals
19. Add to your matrix.
20. Add your two measures to a **Display folder** and name it **Time**



Year	Avg Price	Max Price	Min Price	Sales YTD	Sales YoY Growth
FY2018	**1,288.13**	**3,578.27**	**4.75**	**23,348,493.12**	
2017 Jul	1,785.38	3,578.27	5.19	919,251.52	
2017 Aug	1,371.54	3,578.27	5.01	2,823,398.53	
2017 Sep	1,122.39	3,578.27	4.75	5,187,522.85	
2017 Oct	1,381.91	3,578.27	5.19	6,413,754.83	
2017 Nov	1,118.65	3,578.27	5.01	9,528,026.83	
2017 Dec	1,268.60	3,578.27	5.19	11,746,917.19	
2018 Jan	1,735.97	3,578.27	5.19	12,902,257.51	
2018 Feb	1,190.45	3,578.27	5.01	15,863,641.93	
2018 Mar	1,596.34	3,578.27	5.19	17,634,108.75	
2018 Apr	1,672.56	3,578.27	5.19	19,034,705.14	
2018 May	1,198.20	3,578.27	5.01	21,326,934.25	
2018 Jun	1,002.16	3,578.27	4.75	23,348,493.12	
FY2019	**573.74**	**2,443.35**	**2.29**	**33,636,168.52**	**44.06%**
2018 Jul	689.43	2,443.35	5.01	2,304,212.57	150.66%
2018 Aug	474.43	2,443.35	4.75	5,941,043.44	91.00%
2018 Sep	483.64	2,443.35	4.32	9,648,655.43	56.83%
2018 Oct	525.82	2,443.35	4.75	12,225,945.14	110.18%
2018 Nov	563.89	2,443.35	5.01	15,410,038.73	2.24%
2018 Dec	582.60	2,443.35	4.75	18,291,101.83	20.84%
Total	**465.18**	**3,578.27**	**1.33**	**52,824,612.56**	**92.70%**

Intelligence