# Summary: Advanced Power Query and Data Modelling

## SESSION OVERVIEW:

By the end of this session, students will be able to:

- Handle missing values and errors using Power Query Editor
- Merge and append tables in Power Query Editor
- Understand major data models in Power BI and the need of data models
- Define relationships in data models and the cardinality

## KEY TOPICS AND EXAMPLES:

### 1. Power Query - 3: Handling missing values/errors and merging/appending tables

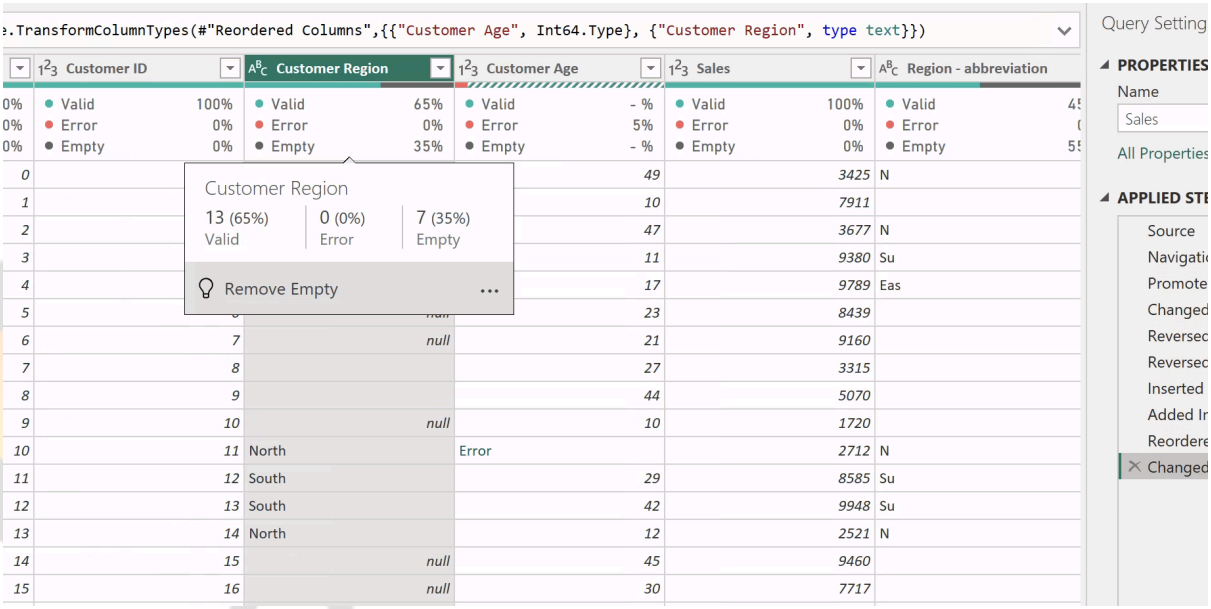
Many times situations arise that require us to handle missing values, errors and outliers. Power Query editor has numerous tools that enable us to handle such situations.

#### a. Handling Missing Values

Power BI automatically assigns a 'null' value to all columns wherever the data is missing. We can also see the number of null/empty values through **column profiling**. This is the green bar below each column that shows how many values are **valid**, have **errors** or are **empty**.

**Note**: By default, Power Query performs data profiling over the first 1,000 rows of our dataset. If we want it to operate over the entire data set, select the **Column Profiling based on the top 1000 rows** button in the lower part of your editor to change the column profiling to **Column Profiling based on entire data set.**

**Example:** This can be seen in the sales dataset provided. The Customer Region column has 35% empty values. On hovering over the column profile, it shows the number (and %) of empty values and provides an option to remove empty values, as shown below.



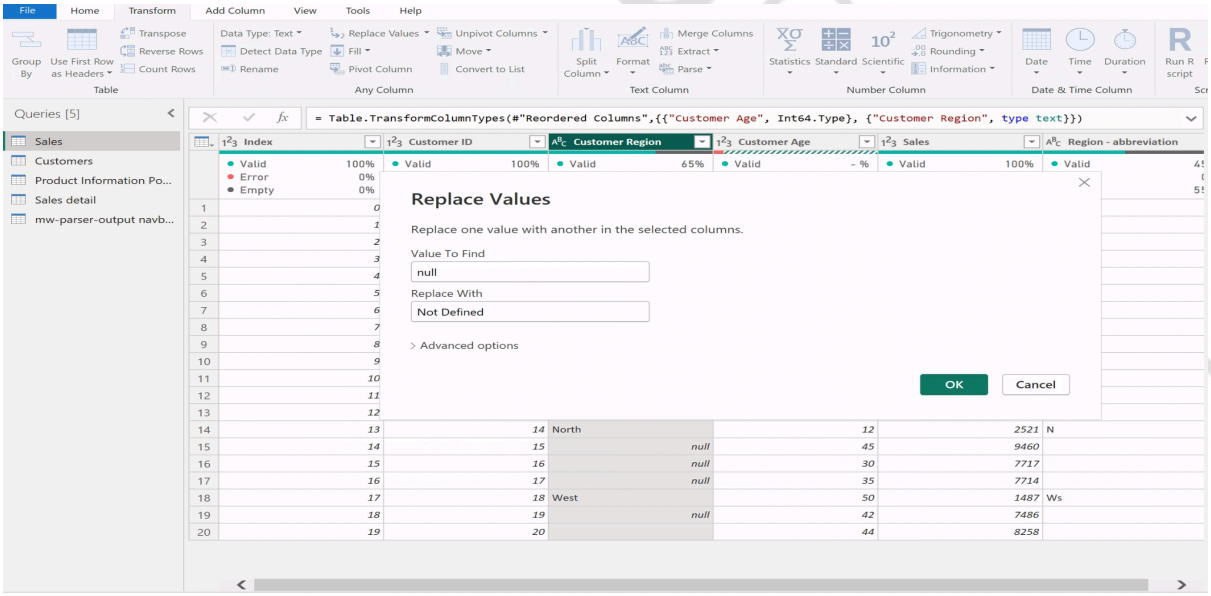

**How do we handle such missing values?**

Here are some ways to do them:

i. **Remove Empty**: This option that is provided by the column profiling bar will simply filter out all rows which have empty values in that particular column. This is usually not desirable as in this case we lose the information stored in the other columns for that filtered out row.

ii. **Replace Values**: Another way to handle missing values is using the Replace Values option. We can replace missing values with a dummy value (e.g. 0) or NA.

**Example:** This is shown for the Sales Dataset below, for the Customer Region column. We can replace the 'null' values by "Not Defined".

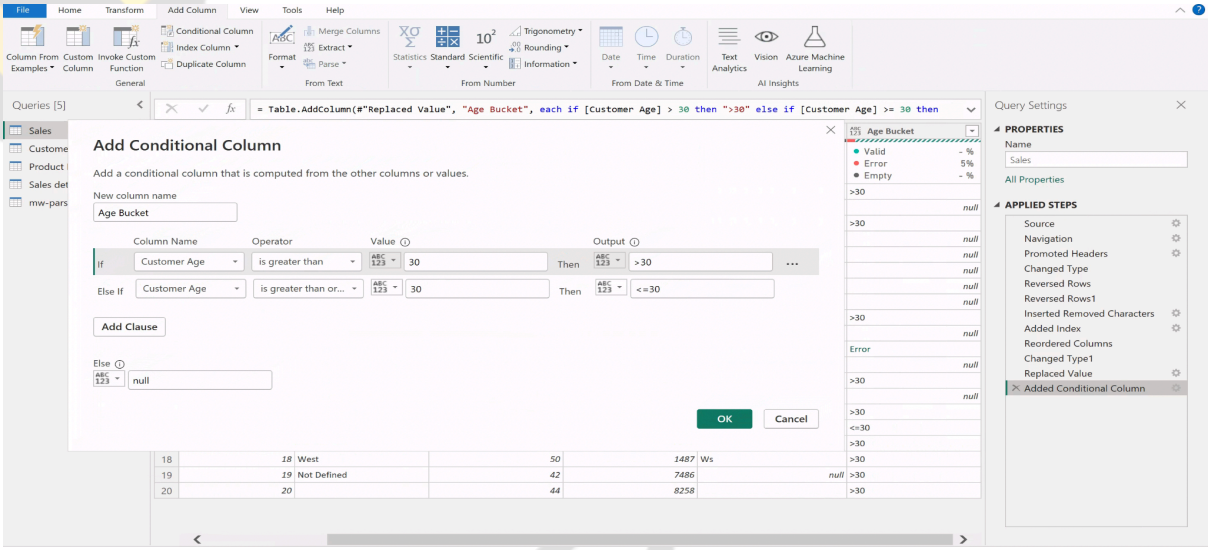



**Note:** The value with which we are replacing must be of the same data type as the value we intend to replace. For example if we are replacing null values in a number data type column with the String value "NA" then it will throw an error. To handle

such errors it is important to first convert the column data type to the required data type (in this case String) and then use the “Replace Values” operation.

iii. **Conditional Column:** We can also create a conditional column that returns a standard value or a value from another column in case of null value. In all other cases, it returns the non-null value from the column.

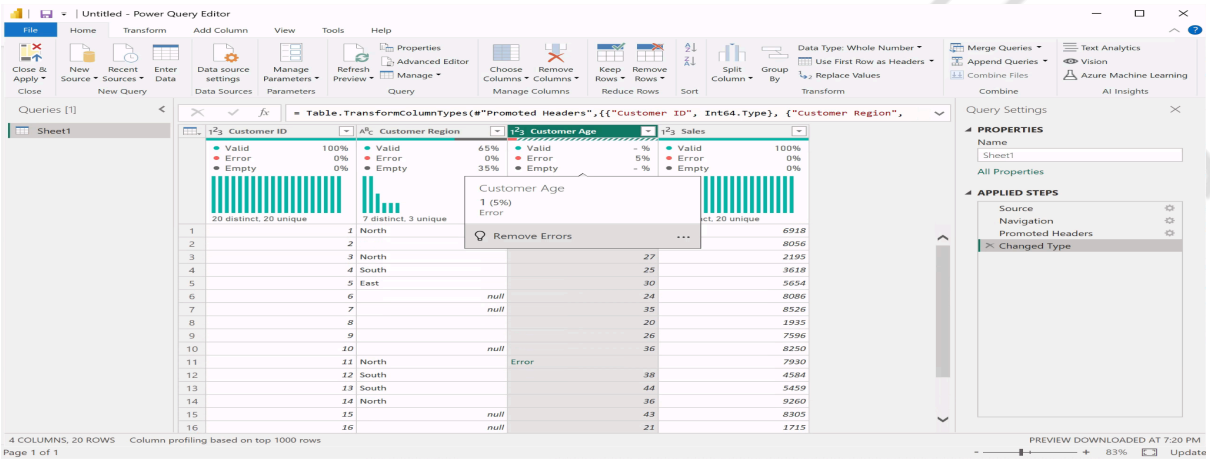
**Example:** This can be seen below for the dataset. If customer age is greater than 30 then it returns “>30”, if it is less than or equal to 30, it returns “<=30” else it returns null.




## b. Handling Errors

In certain situations we get errors in our columns when Power BI is not able to recognise the format of a value. For example, in a number column, if one row has text then Power Query will show an error in that cell.

Whenever we see an error in a column, the column profile bar will change colour and on hovering over the bar, we can see the exact metrics: **number of errors**.
**Example:** In the Sales Dataset, we can see the column profile for the Customer Age column.



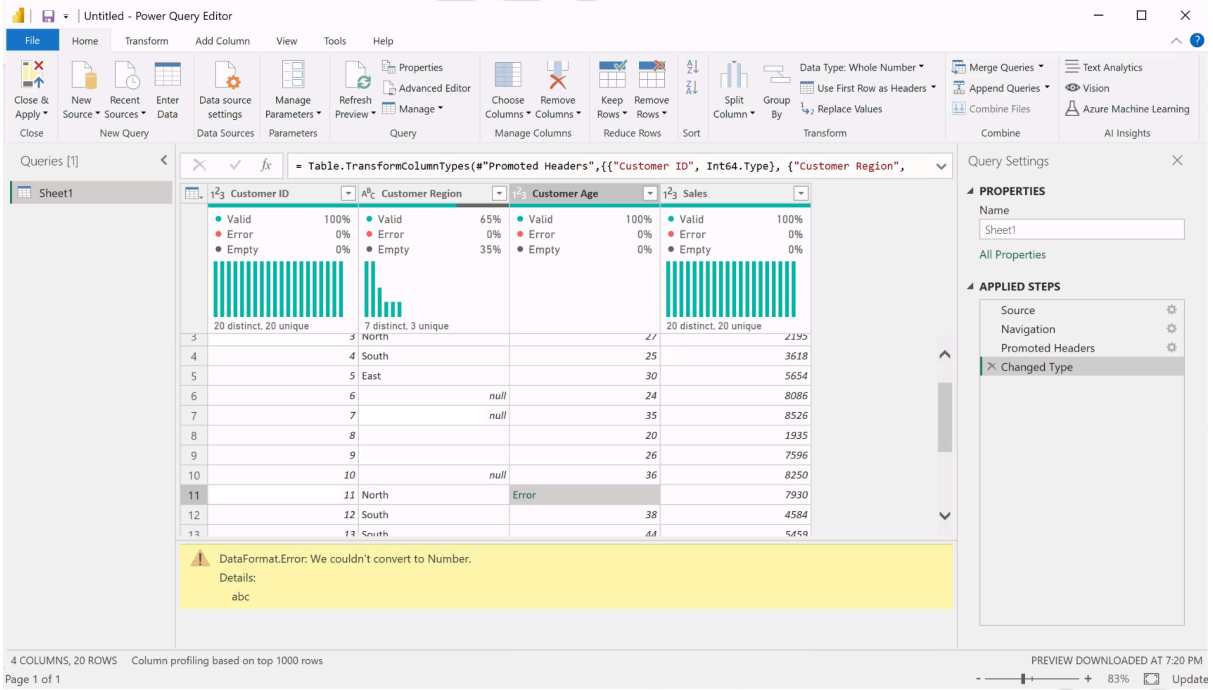
**There are multiple ways to handle errors:**

i. **Remove Errors:** Power Query also provides an option to Remove Errors. It depends on stakeholders if this approach can be used, but in general this approach is not recommended as it removes the entire row and hence there is a loss of data from other columns for that row.

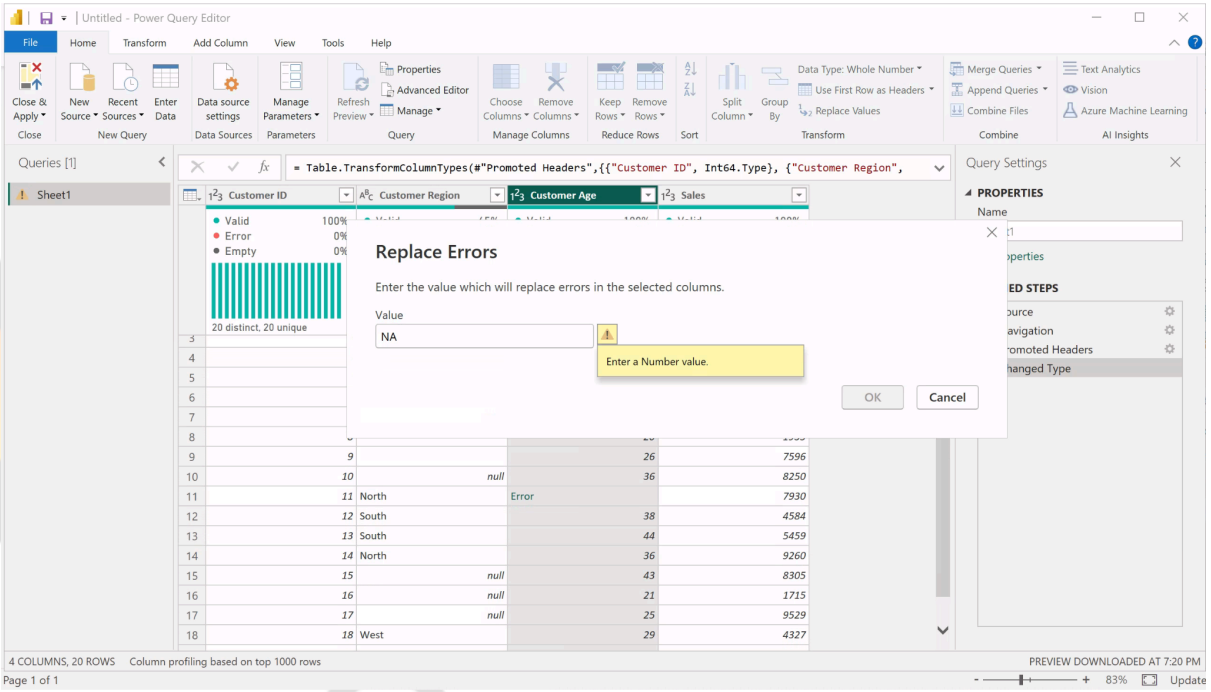
**There are 2 ways to remove errors.**

1. Column Profile bar shows an option to Remove Errors (as shown in image above).
2. Right click on column → Remove Errors

ii. **Replace Errors**: We can replace all the errors in a column by a fixed value (0/1/NA).
**Example:** In this Sales dataset, we can see that one of the values in the customer age column is a data entry error. 'Abc' is mentioned as customer age instead of a numerical value. One of the ways in which we can handle this value is by right clicking on the column → Replace Errors → Enter 34.36 as it is the average age (we can also replace by median or any other suitable value).




**Note**: Here, if we try to replace this value by **NA** or any other **string** value, it will throw an error as shown below. This is because Customer Age is a numerical column and it accepts only numbers. A replacement of any numerical value cannot be a **string/text**.



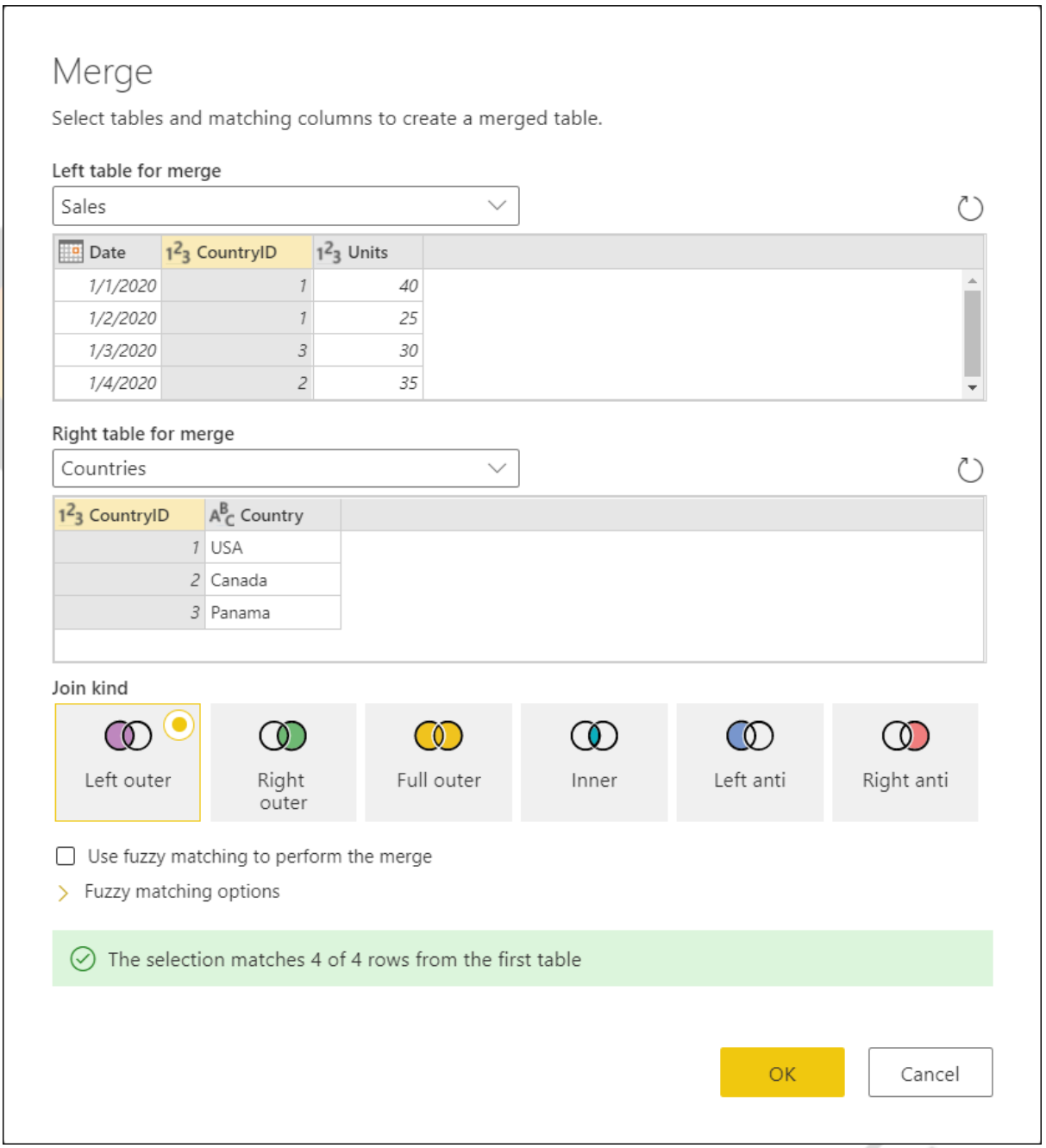

## c. Merging Tables

When we have data in two tables and we want to bring columns from one table to the other, we may need to merge tables. Merging tables functions like Excel VLOOKUP or SQL joins. This can be done using the **Merge Queries** option in the Combine group in the Home tab.

**Example:** In the example below, there are two tables:

- **Sales:** The CountryID field is an identifier from the Countries table.
- **Countries:** This table contains the CountryID and the country name.

We can join these tables using the column column, in this case Country ID.

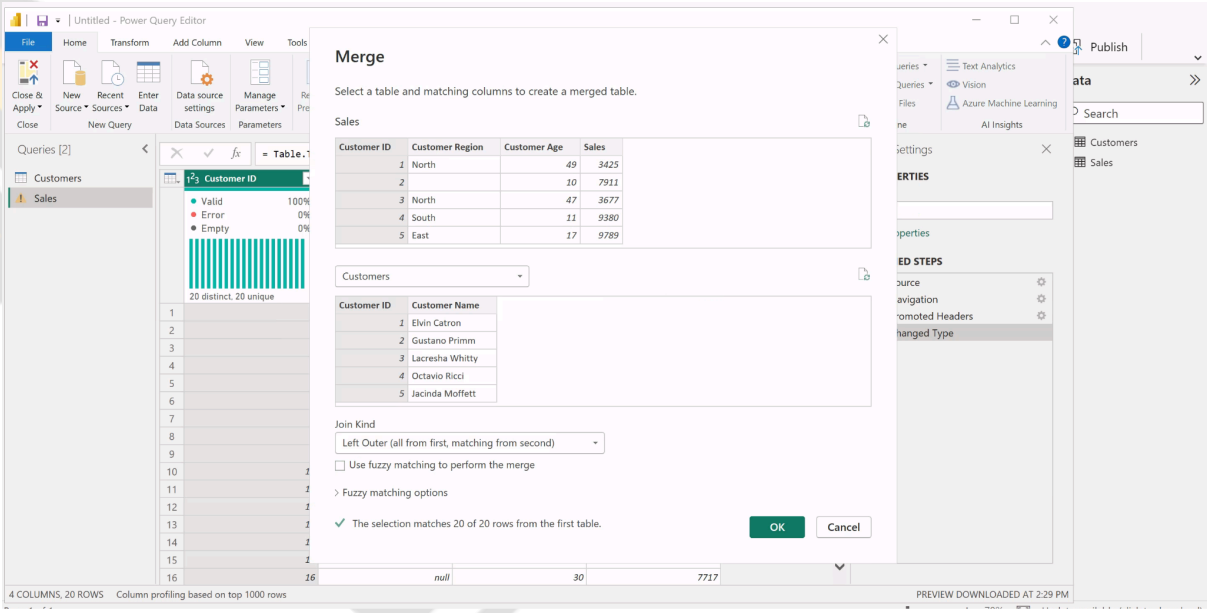



There are multiple types of joins that Power BI provides:

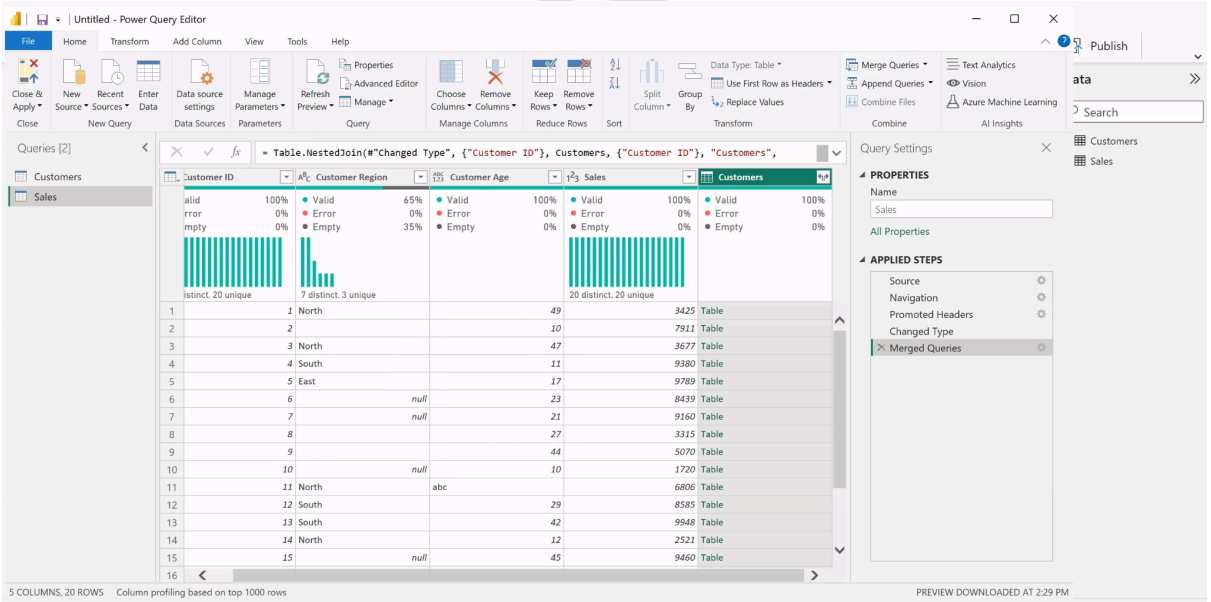
- **Left outer**: Returns all rows from the left table and matching rows from the right table.
- **Right outer**: Returns all rows from the right table and matching rows from the left table.
- **Full outer**: Returns all rows from both the tables.
- **Inner**: Returns only matching rows from both the left and the right tables.
- **Left anti**: Returns only the rows from the left table that do not match with rows from the right table.
- **Right anti**: Returns only the rows from the right table that do not match with rows from the left table.

**Example:** To practice merging tables, this dataset can be used. It has two sheets: **Sales** and **Customers**. These steps need to be followed to merge these tables:

1. Go to Merge Queries
2. Select the Customers table from the dropdown menu in the Merge window
3. Select the customer ID column in both tables
4. click OK

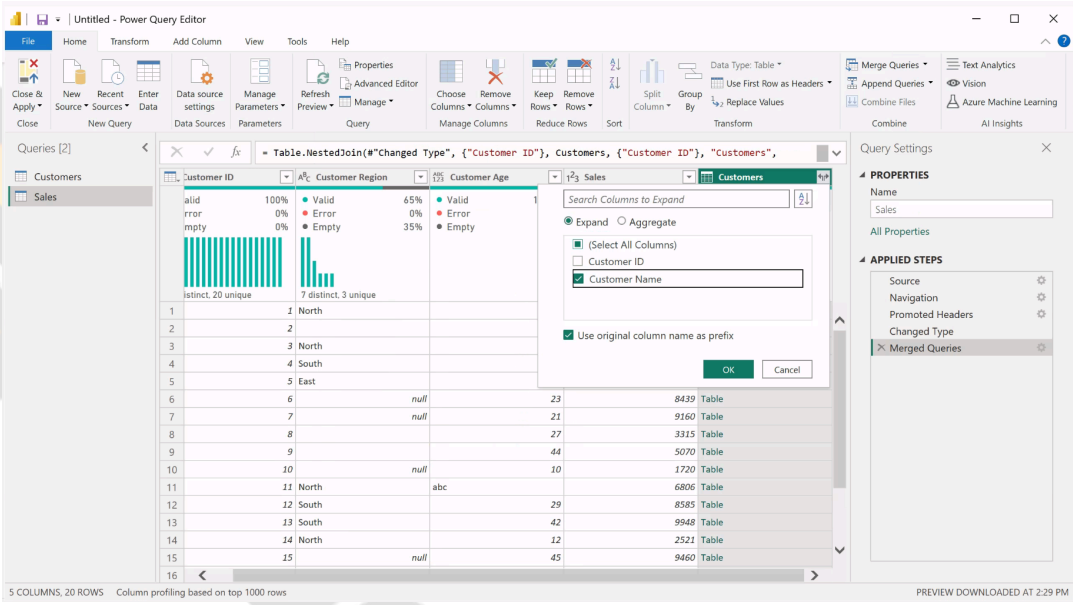



5. It shows an additional column: Customers (shown below). Click on the icon on the top right of the Customers column.

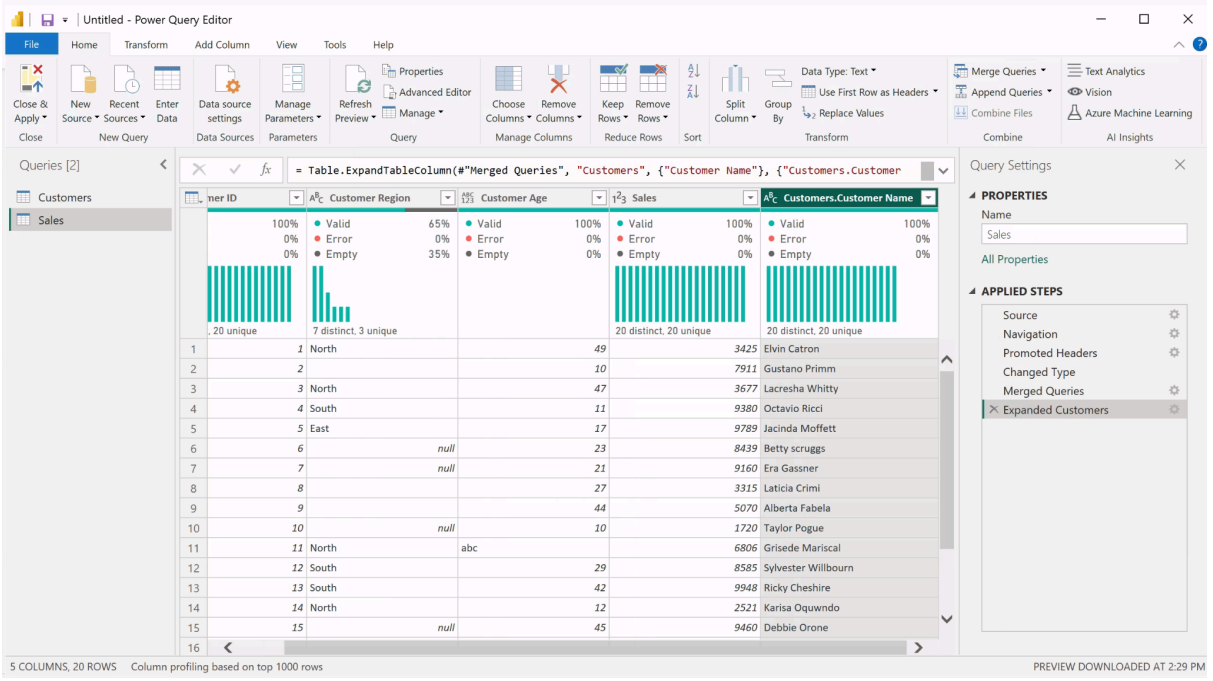


6. Select only the Customer Name column from the dropdown list. Make sure the Expand option is selected instead of aggregate.
    a. **Expand:** When you choose "Expand," Power BI brings in all individual rows from the merged table and creates a new column within your existing table.

**b. Aggregate:** Choosing "Aggregate" allows you to summarize the data from the merged table before incorporating it into your main table. You can choose various aggregation functions like SUM, and COUNT




7. On clicking OK, the Customers.Customer Name column appears in your existing data. In this manner we have merged both tables.




## d. <u>Appending Queries</u>

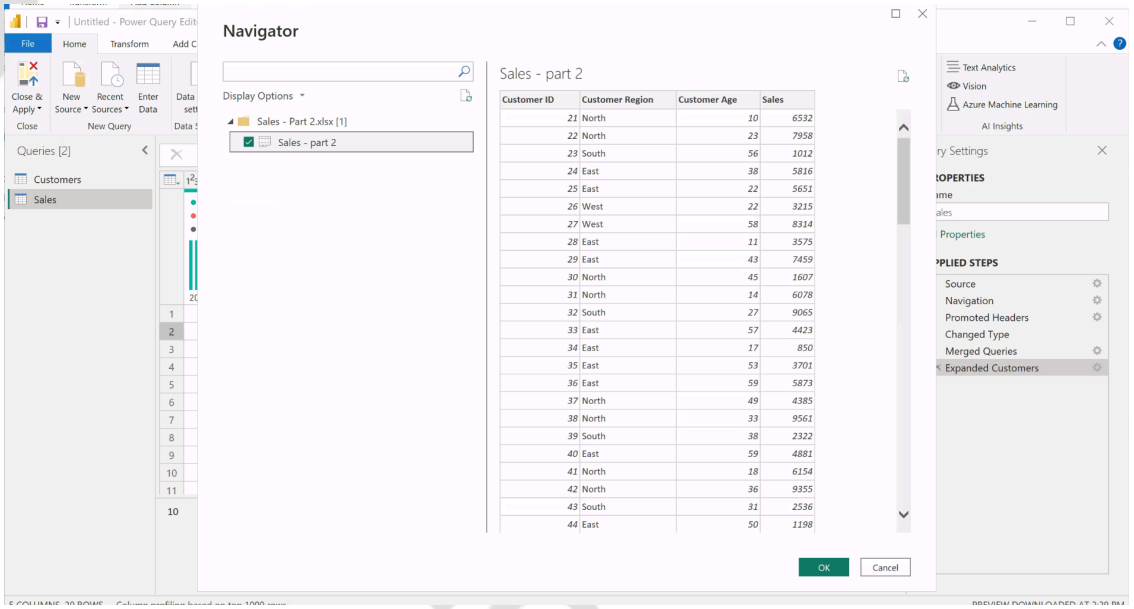
If we have multiple sheets or datasets in the same format and they need to be placed on top of each other, we use the Append functionality. Key salient features include:

- The data type of each column and the order of columns should be the same in the tables that are to be appended.

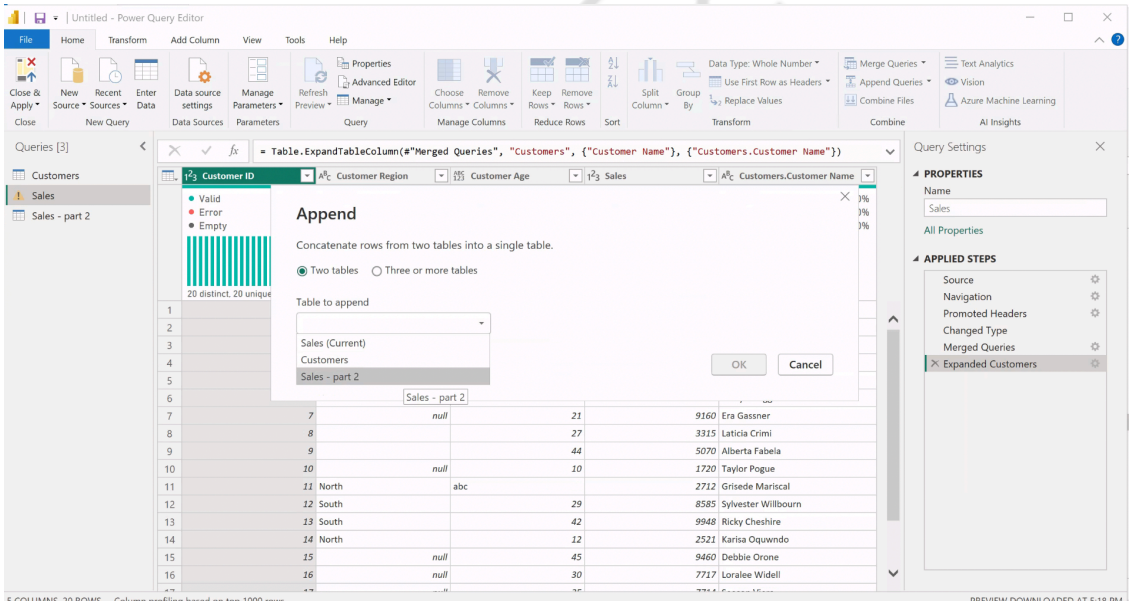
- It is similar to the UNION functionality in SQL.
- The append dialogue box presents 2 options:
    - **Append Queries**: This is the default option, allowing you to select multiple queries (tables) you want to combine.
    - **Append Queries as New**: Choosing this option creates a new query with the appended data, keeping the original tables intact.

**For example:** Get data from a new data source: Sales - Part 2 Excel file and use the append operation on the Sales table in the Sales dataset. This is demonstrated below.

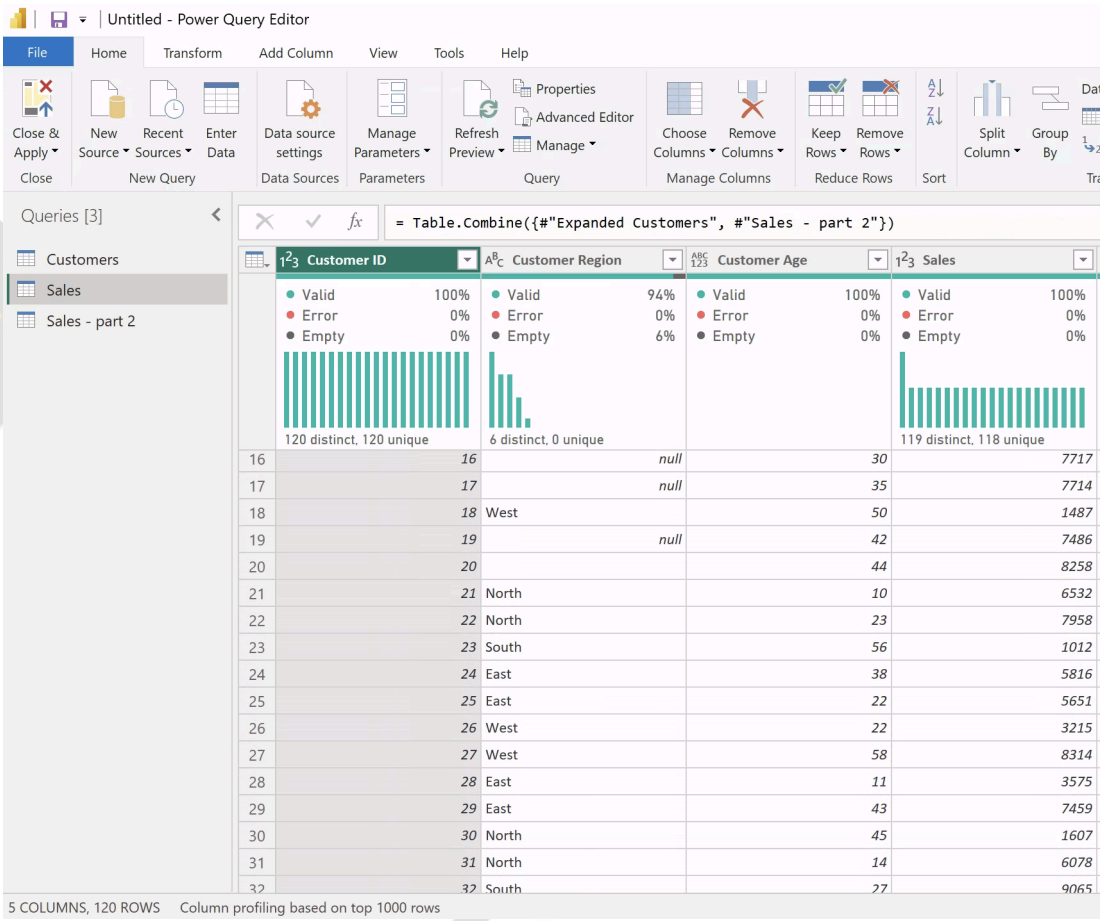
**Step 1:** In the Power Query Editor, Get data from Sales - Part 2 Excel file.




**Step 2:** In the Sales dataset, click on Append queries. In the Append window, select the Sales - Part 2 dataset as shown below.



**Step 3:** As can be seen in this image, the queries (tables) have been appended.




| | Customer ID | Customer Region | Customer Age | Sales |
|---|---|---|---|---|
| 16 | 16 | null | 30 | 7717 |
| 17 | 17 | null | 35 | 7714 |
| 18 | 18 | West | 50 | 1487 |
| 19 | 19 | null | 42 | 7486 |
| 20 | 20 | | 44 | 8258 |
| 21 | 21 | North | 10 | 6532 |
| 22 | 22 | North | 23 | 7958 |
| 23 | 23 | South | 56 | 1012 |
| 24 | 24 | East | 38 | 5816 |
| 25 | 25 | East | 22 | 5651 |
| 26 | 26 | West | 22 | 3215 |
| 27 | 27 | West | 58 | 8314 |
| 28 | 28 | East | 11 | 3575 |
| 29 | 29 | East | 43 | 7459 |
| 30 | 30 | North | 45 | 1607 |
| 31 | 31 | North | 14 | 6078 |
| 32 | 32 | South | 27 | 9065 |

## 2. Models in Power BI

Data Modelling in Power BI enables us to connect multiple data sources using relationships. Relationships define how data sources are connected to each other. **Note**: In Excel, Data Modelling is called **Power Pivot**.

**Types of models in Power BI:**

### A. Star Schema

**a.** It is a design approach that is widely used by data engineers as it presents a user-friendly structure and high performance analytics queries.

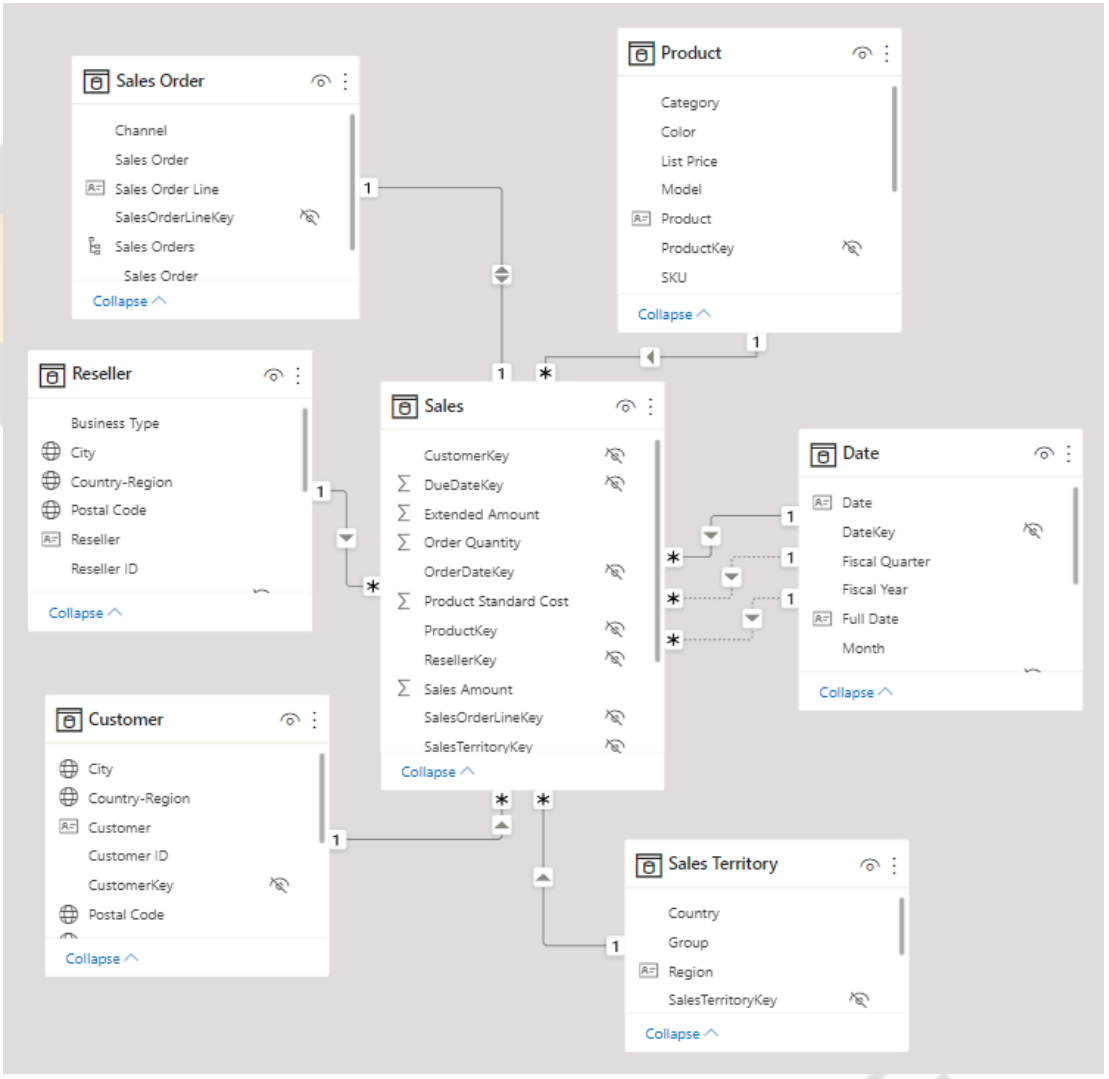
**b.** It classifies model tables as fact or dimension. A fact table forms the center of the star while dimension tables form the points of the star.

**c.** **Fact Table**: It stores events or observations. It includes dimension key columns and **numeric measure columns** that can be summarised. The main goal of a fact table is to summarise.

**d.** **Dimension Table**: It stores business entities. It includes a key column and **descriptive columns** for filtering and grouping.

e. Dimension tables are usually related to fact tables by using one-to-many relationships. This allows filters applied to dimension table columns to propagate to the fact table. We can hence directly apply the filters on the fact table to get the desired output.

**Note**: Star schema is an example of a type of dimensional data modelling. Another type of dimensional modelling type is the Snowflake schema.

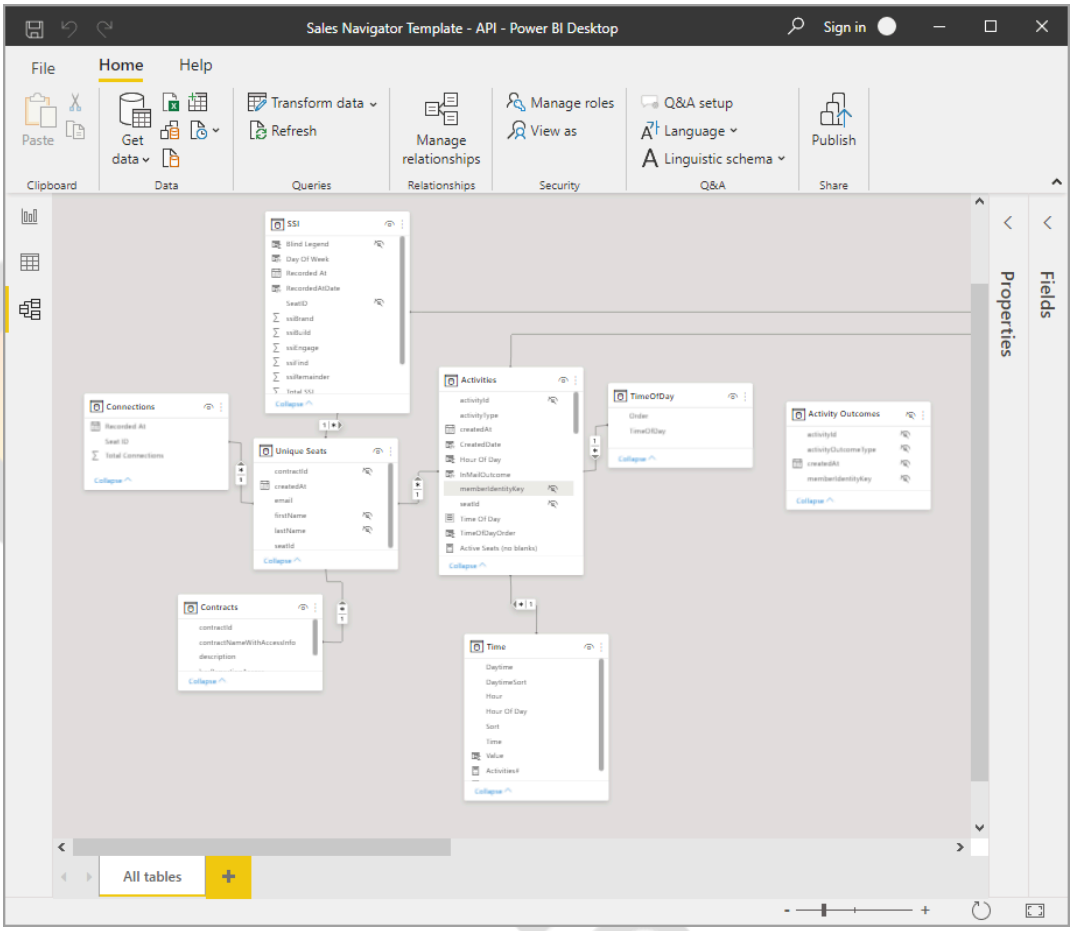



The above model comprises 7 tables, the central one of which is the fact table (**Sales**) and the rest 6 are dimension tables:

- Sales Order
- Reseller
- Customer
- Sales Territory
- Date
- Product

## B. ER Model

It represents **entities** and **relationships** in a diagrammatic form. This model is widely used in relational databases. Entities are represented by tables, attributes by columns and relationships by lines connecting entities.




## Phases of a Data Model:

1. **Conceptual data model:** It defines the overall structure of the business and the data. We define at the table level and data level (entities and entity relationships). For instance, you may have customer, employee, and product data.
2. **Logical data model:** Here we look at specific attributes within the entities: the columns/attributes of a table. For instance, Customer A buys Product B from Sales Associate C.
3. **Physical data model:** Here we look at the data type and size of the attributes of the table. It is the actual implementation of the data model.

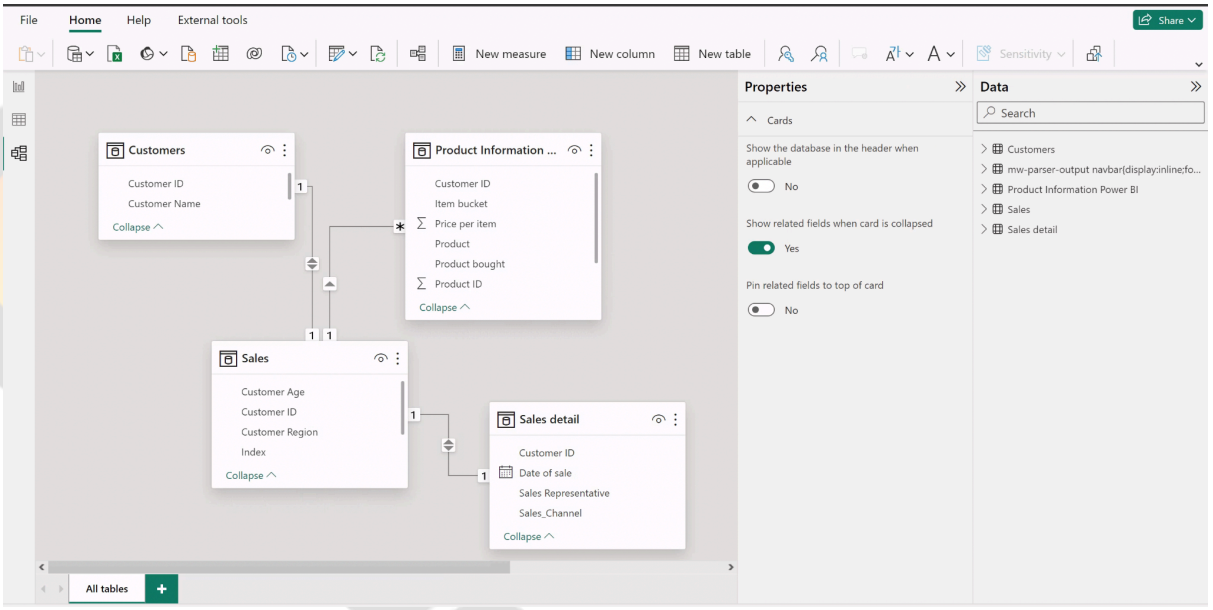
**Need of a Good Data Model in Power BI:**

- Saves time, effort and money
- Increases performance and speed of analytics queries
- Reduces errors
- Maintains quality of data

**Example of Data Model:**

Use the Sales dataset, Product Information dataset and Sales Detail dataset for implementing a data model practically in Power BI.

Once you import the datasets in Power BI, it will automatically detect the relationships and create a **star schema** model, as shown below. The central table is the fact table and the rest are dimension tables.



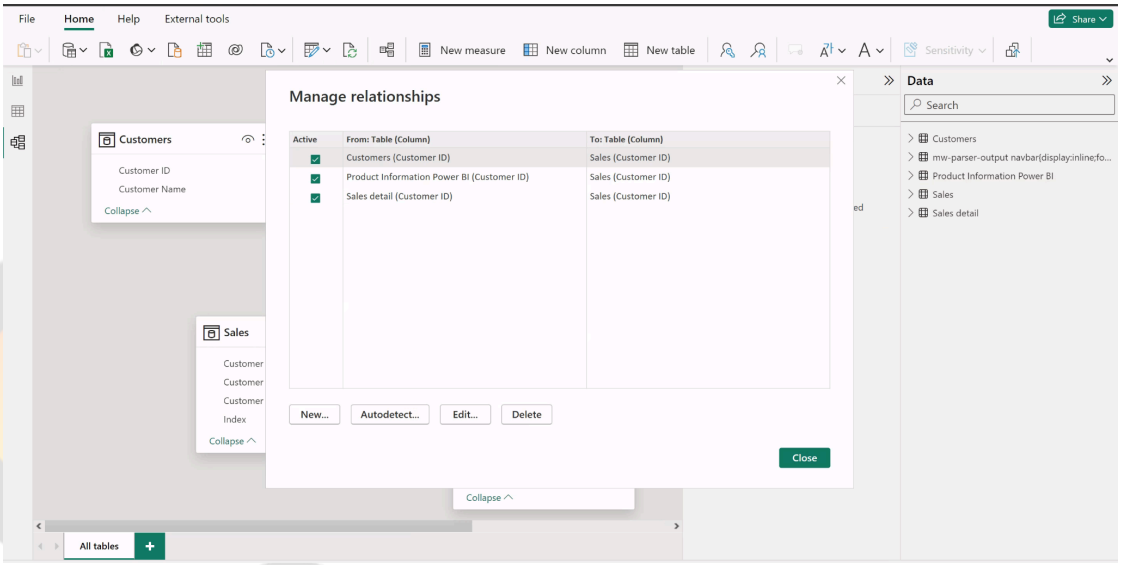

Power BI has automatically identified the common column: Customer ID from the four tables and has created all necessary relationships

- Sales table in the Sales dataset
- Customers table in the Sales dataset
- Product Information table
- Sales Detail table

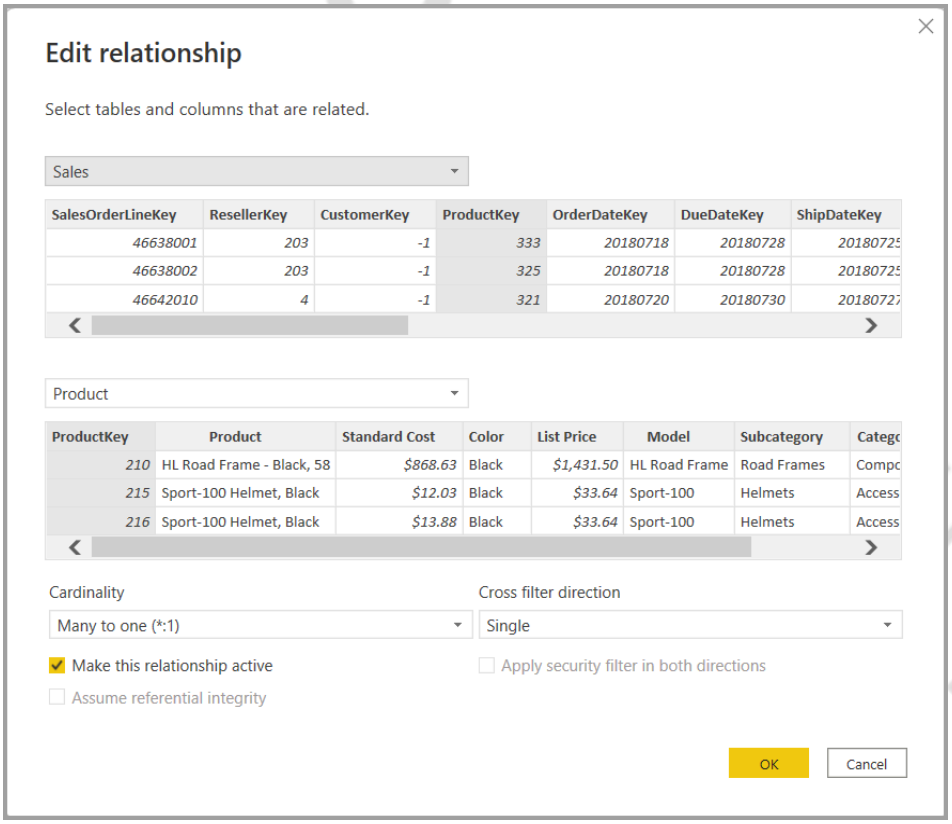
## 3. Relationships and Cardinality

- After we have defined the model, as explained in the previous section, we need to define relationships among our data sources. In the Models section of Power BI Desktop, we can see the relationship between our data sources. Power BI tries to automatically detect the relationship between different data sources. These relationships are defined using columns in the dataset.

- We can also define a new relationship. To create a new relationship, we need to drag a column name from one data source to the respective column in the other data source. Another way to do this is to go to the Manage Relationships option under the Home tab and click on the New button. This is shown in the image below:

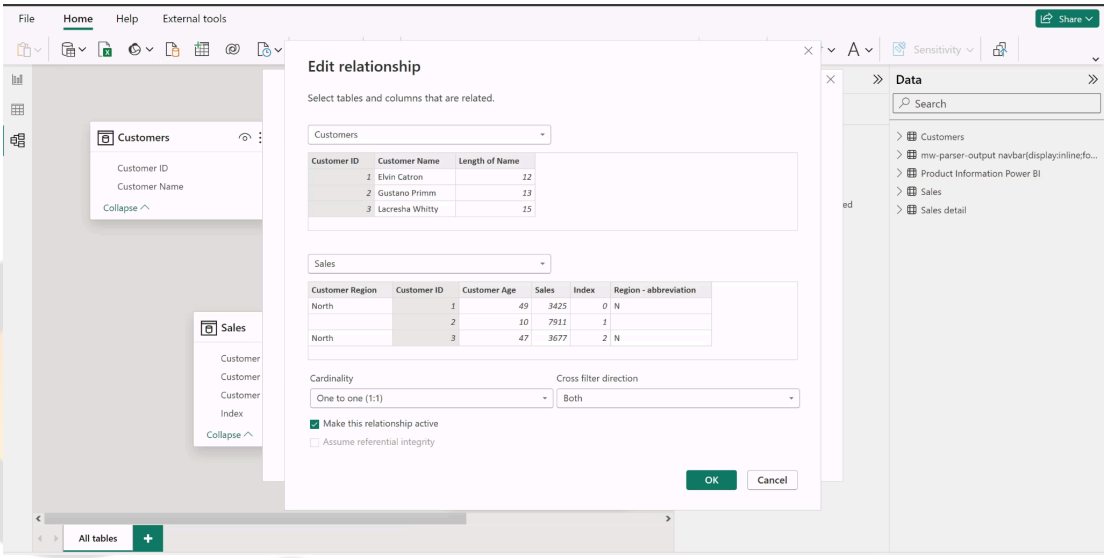



- We can change an existing relationship below selecting a relationship and clicking on **Edit** to edit it. This is shown in the image below:




- The relationship view can also be used to hide a column in the report. Right click the column name in the data source and select “Hide in report view”.

**Example:** In our dataset, we have a relationship between Sales and customers through Customer ID column, as shown below.




## Cardinality:

Each model relationship has a particular cardinality associated with it. Cardinality refers to the relationship or the number of distinct values in the relationship. It determines how rows in one table relate to rows in another table based on specific columns. “One” means the column has unique values while “many” means that the column can contain duplicate values. The four types of cardinality are:

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

Power BI detects the best cardinality for each relationship in your data model. You can also change this in the data model.

**Note: Many to many relationships are not recommended.** This is due to the following reasons:

- The model doesn’t provide flexibility in the ways you use the filter or group functionalities.
- It also makes it difficult to create valid calculations and aggregations, slows down queries, and impacts overall report performance.
- It can lead to unpredictable results, and lack of normalization.

Hence, many to many relationships should be avoided. It should be noted that the cardinality defaults to many-to-many when it determines neither table contains unique values for the relationship columns.

**Example:** In our dataset, Power BI has intelligently identified a **one-to-one cardinality** on the **Customer ID** column in **Sales** sheet as well as **Customers** sheet.

### A. Direction of Data Flow

Relationships are defined using a **cross filter direction**, which determines the direction that filters will propagate. There are 2 types of cross filter directions:

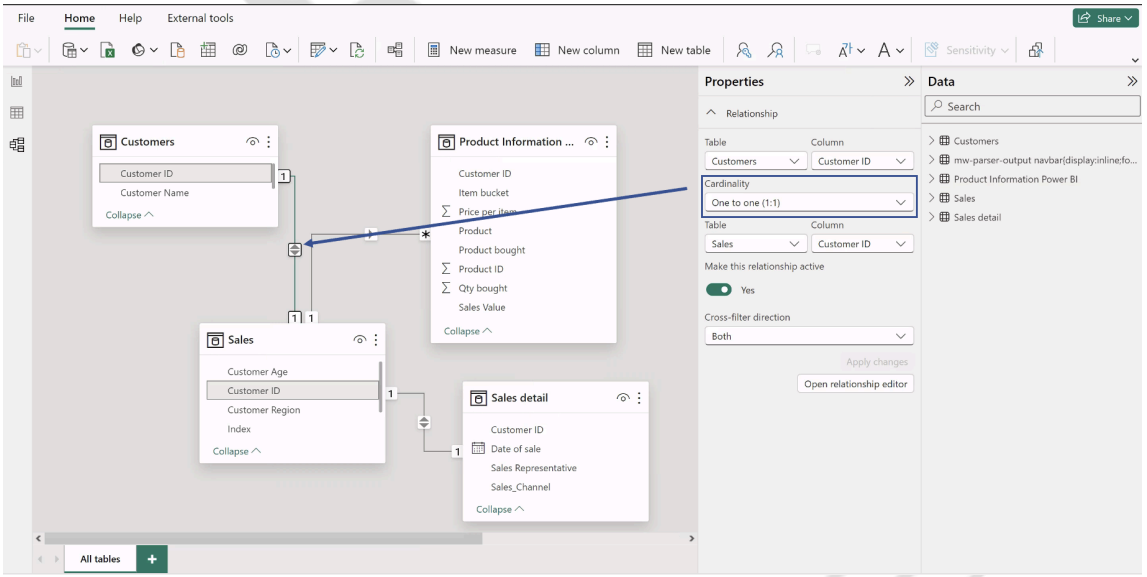
**Single**

Only one table in the relationship can filter data across to the other table. If we apply filters in the other table, they are not applied in the first table.

**Bi-directional**

Data can be filtered in either direction. If we filter data in the first table, the other table also gets filtered, and vice versa. This type is not recommended as it can create ambiguous filter paths.

As can be seen from the image below, Power BI has automatically classified some relationships as single directional and some as bi-directional.




## B. Active/Inactive Relationships

As can be seen from the image above, we have an option (toggle) to make a relationship active/inactive in the Data Modelling window. What if you need more than one connection between tables? This is where active and inactive relationships come in.

### Active Relationships: The Workhorses:

An active relationship is the default connection between two tables. It's symbolized by a **solid line** in the Power BI model view.

- When you filter data in one table, the filter automatically propagates to the connected table through the active relationship. This allows you to explore related data seamlessly.
- Each table can only have one active relationship with another table. This prevents ambiguity and ensures clear filter flow.

**Inactive Relationships: Hidden Paths:**

Inactive relationships are additional connections you create between tables, represented by **dashed lines**.

- They don't participate in automatic filter propagation.
- The power of inactive relationships lies in their controlled activation.
- You can use the USERELATIONSHIP function in DAX expressions to activate a specific inactive relationship for calculations or custom visuals.

**Why Use Inactive Relationships?**

Imagine a "Customers" table with a "Customer Type" column. You can have separate inactive relationships for "Customer Type" connecting to a "Sales" table and a "Support Tickets" table, allowing for role-based analysis.

Active relationships are the backbone for filter propagation in your Power BI model. But don't underestimate the power of inactive relationships. They offer flexibility and control for complex data analysis scenarios, especially when dealing with multiple relationships or role-playing dimensions.