



Unit-3

Power BI Models

Data Modeling

- Data modeling is the process of organizing and structuring data in a way that allows for efficient retrieval and analysis. In Power BI, this involves creating a relationship between data tables that will allow us to join and aggregate information from multiple sources.
- One of the key benefits of data modeling in Power BI is that it allows us to create a single source of truth for our data. By organizing and structuring our data in a consistent and logical manner, we can ensure that everyone in our organization is working with the same information. This can help to eliminate confusion and errors, and ensure that decisions are based on accurate and reliable data.
- Another important aspect of data modeling in Power BI is the ability to create calculated columns and measures. These are calculations that are performed on our data, and can be used to create new insights and metrics. For example, we might create a calculated column that calculates the profit margin for each product, or a measure that calculates the total sales for a particular region. These calculations can be used in our visualizations and reports, allowing us to gain deeper insights into our data.

The Importance of Data Modeling in Power BI

- Proper data modeling is essential for creating accurate and meaningful data visualizations. When data is poorly structured or organized, it can lead to inaccurate results and flawed conclusions. A well-designed data model not only ensures that data is accurate but also makes it easier to manage, update, and maintain over time.
- One of the key benefits of data modeling in Power BI is the ability to create relationships between different data sources. This allows for more complex analyses and the ability to combine data from multiple sources into a single report. Additionally, data modeling can help identify data quality issues and inconsistencies, allowing for more effective data cleaning and preparation.
- Another important aspect of data modeling is the ability to optimize performance. By properly structuring data and creating efficient relationships, queries and calculations can be executed more quickly, improving the overall speed and responsiveness of the report. This is especially important when dealing with large datasets or complex calculations.

Steps to Create a Data Model in Power BI

- Creating a data model in Power BI involves several steps:
- Identify the data sources that you want to include in your data model.
- Import the data into Power BI.
- Create a relationship between the data tables.
- Define calculated columns and measures to summarize and aggregate data.
- Optimize the performance of your data model.
- It is important to note that creating a data model in Power BI requires a good understanding of the data and its relationships. Before importing the data, it is recommended to clean and transform it to ensure accuracy and consistency. Additionally, Power BI offers various tools and features to enhance the data model, such as creating hierarchies, adding perspectives, and using advanced DAX functions. By utilizing these tools, you can create a robust and efficient data model that provides valuable insights for your business.

How to Create Relationships between Tables in Power BI

- One of the most important steps in creating a data model in Power BI is defining relationships between tables. To create a relationship, you will need to identify the fields that link the two tables together. This is typically done by defining a primary key in one table and a foreign key in the other. Once the relationship is defined, Power BI will automatically join the tables together when you create visualizations.
- It is important to note that relationships can be one-to-one, one-to-many, or many-to-many. A one-to-one relationship means that each record in one table is related to only one record in the other table. A one-to-many relationship means that each record in one table can be related to multiple records in the other table. A many-to-many relationship means that multiple records in one table can be related to multiple records in the other table.

Filtering and Sorting Data Models in Power BI

- Power BI allows you to filter and sort data in your data model in a variety of ways. To filter data, you can use slicers, which allow you to select specific values or ranges to view. You can also use filters to hide or exclude specific data based on certain criteria. Sorting data can be done manually using sorting options in the visualization or by defining a sort order in the data model.

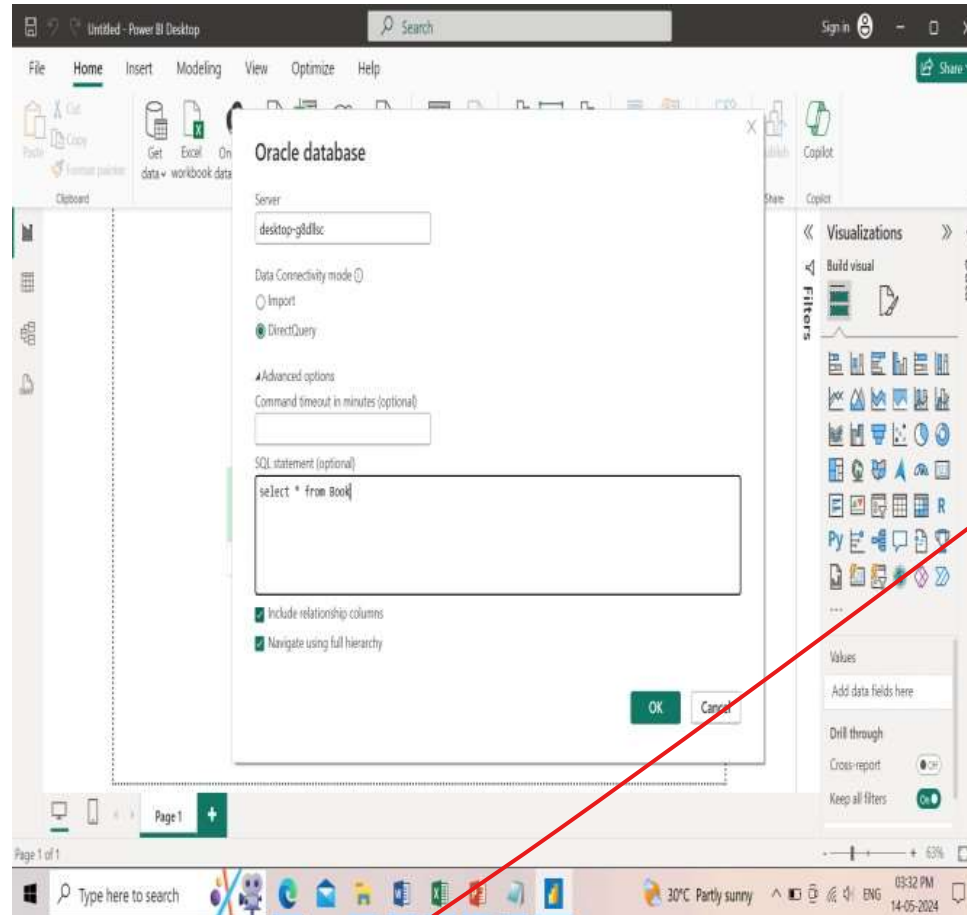
Composite Model

- With composite models, you can connect to different kinds of data sources when you use Power BI Desktop or the Power BI service. You can make those data connections in a couple of ways:
- By importing data to Power BI, which is the most common way to get data.
- By connecting directly to data in its original source repository by using DirectQuery. To learn more about DirectQuery, see [DirectQuery in Power BI](#).
- When you use DirectQuery, composite models make it possible to create a Power BI model, such as a single *.pbix* Power BI Desktop file that does either or both of the following actions:

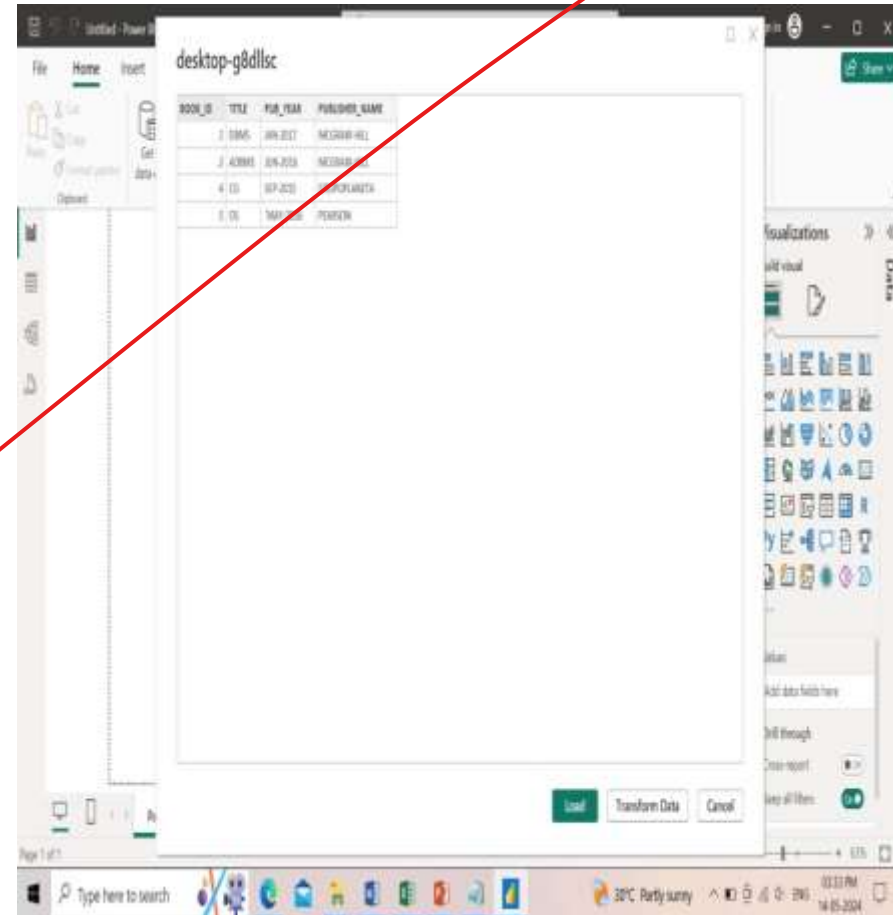
Steps to create Direct Query Method

- Step1: Open powerBI Desktop
- Step2: Goto GetData
- Step3: Select Oracle Database
- Step4: Give ServerName
- Step5: Select Direct Query
- Step6: Select Advance option
- Step7: Give Direct Query.
- Ex: Select * from Book
- Step8: Click on Load
- Step9: Data will be imported into powerBI Desktop.
- Step10: Do Data visualization and Analysis by using Charts.

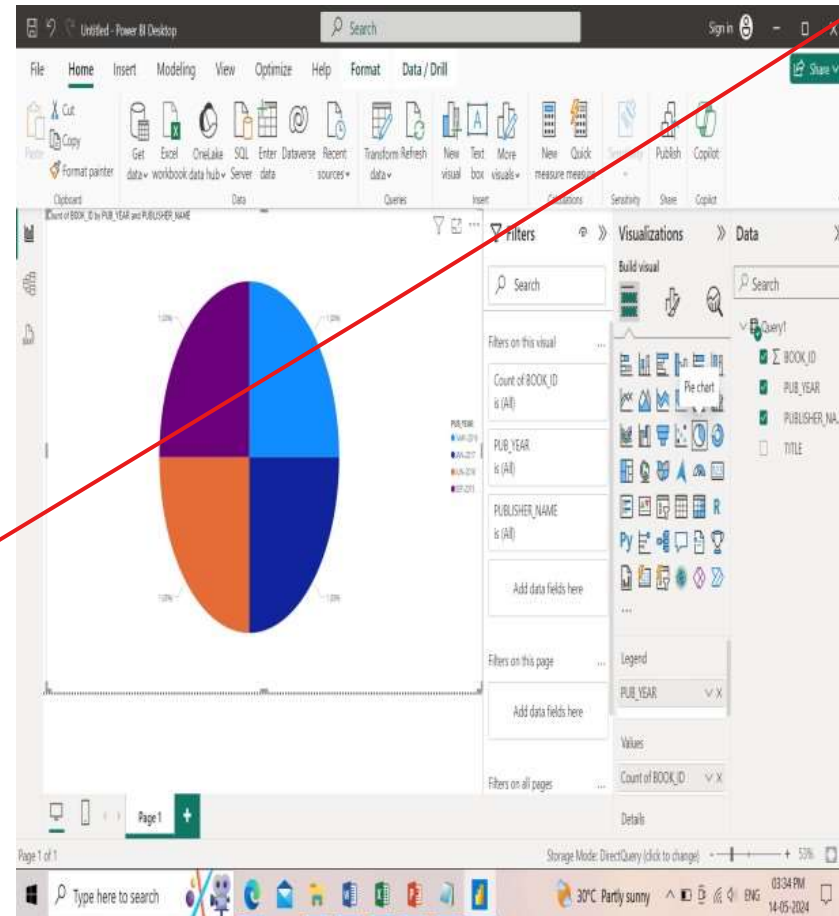
Output



Contd..



Output



Steps to Combine Tables in a Composite Model

- Steps to Combine Tables in a Composite Model

Initial Import:

- Step 1: Open Power BI Desktop.
- Step 2: Click on Get Data and select your data source (e.g., SQL Server).
- Step 3: Enter connection details and select Import mode.
- Step 4: Choose the SalesHistory table and load it into Power BI.

Adding Direct Query Table:

- Step 1: In Power BI Desktop, go to Home -> Get Data.
- Step 2: Select the same data source (e.g., SQL Server).
- Step 3: Enter connection details and select Direct Query mode.
- Step 4: Choose the CurrentSales table and load it into Power BI.

Contd..

Creating Relationships:

- Step 1: In the Model view, create relationships between the imported SalesHistory table and the Direct Query CurrentSales table if needed. For instance, you might relate them using a common key like ProductID or Date.

Combining Data:

- Step 1: Use DAX to create calculated tables or measures that combine data from both SalesHistory and CurrentSales.
- Step 2: For example, you might create a measure to calculate total sales by combining historical and current sales data:

DAX

- Copy code
- $\text{TotalSales} = \text{SUM}(\text{SalesHistory}[\text{SalesAmount}]) + \text{SUM}(\text{CurrentSales}[\text{SalesAmount}])$
- Step 3: Use these measures in your reports to present a combined view of historical and current sales data.

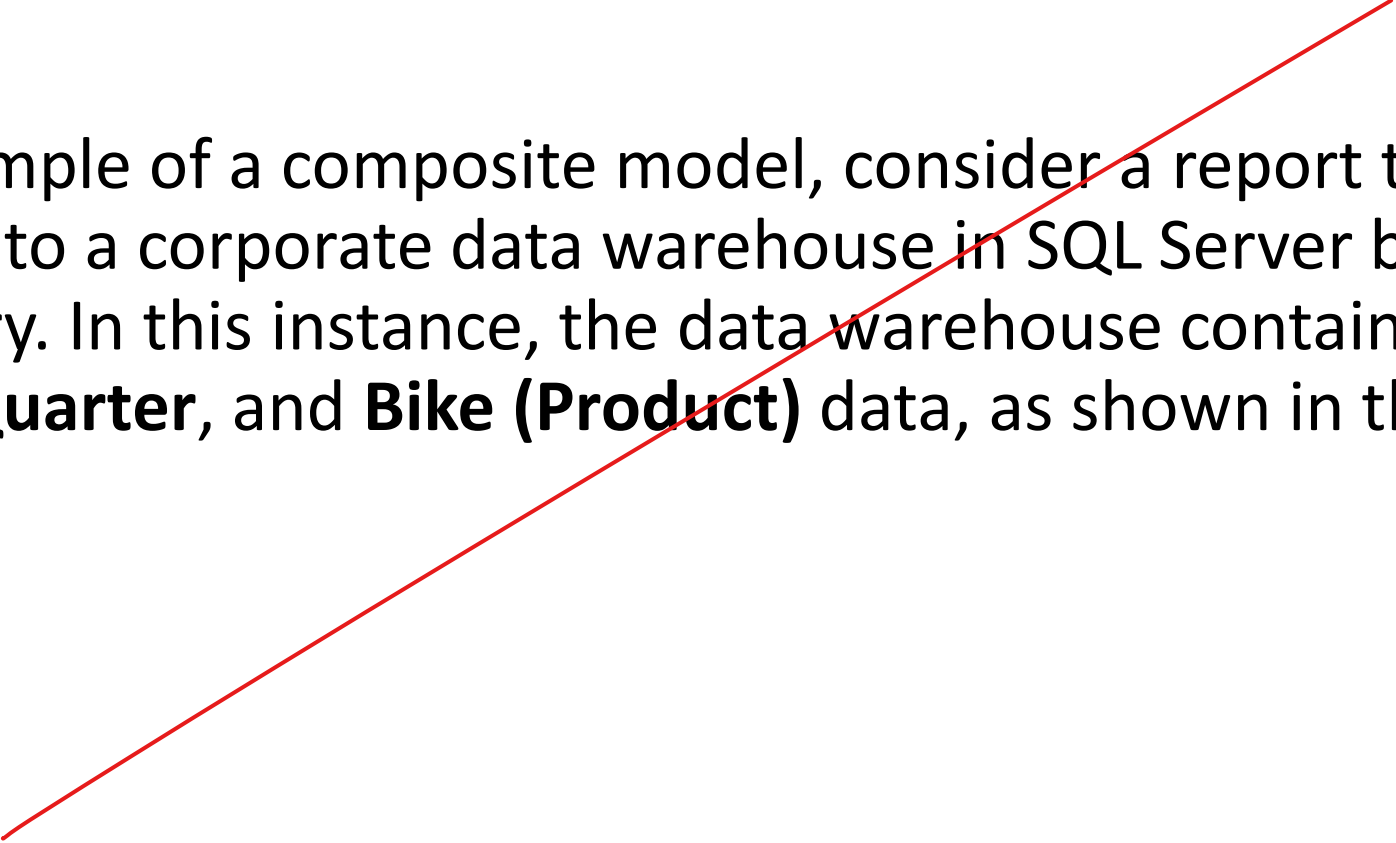
Contd..

- Combines data from one or more DirectQuery sources.
- Combines data from DirectQuery sources and import data.

For example, by using composite models, you can build a model that combines the following types of data:

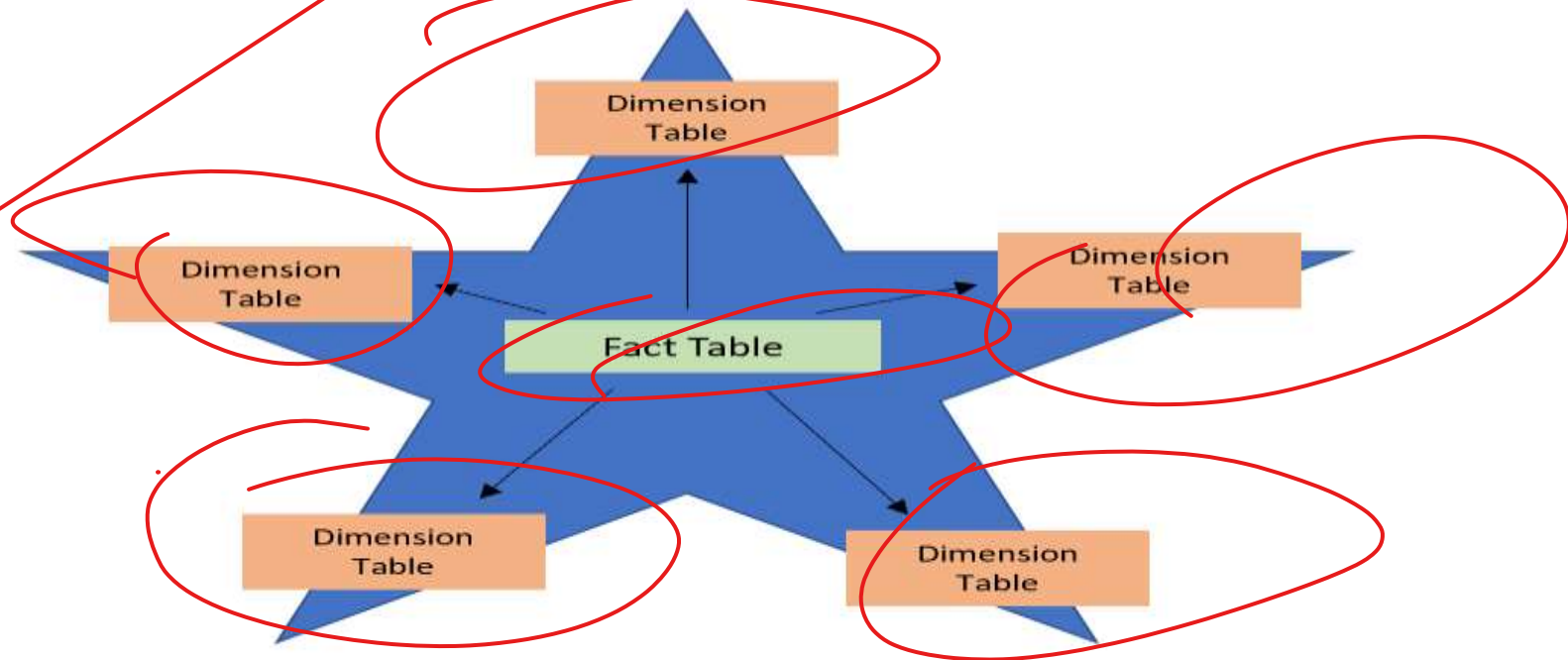
- Sales data from an enterprise data warehouse.
- Sales-target data from a departmental SQL Server database.
- Data that's imported from a spreadsheet.
- A model that combines data from more than one DirectQuery source or that combines DirectQuery with import data is called a composite model.
- You can create relationships between tables as you always have, even when those tables come from different sources. Any relationships that are cross-source are created with a cardinality of many-to-many, regardless of their actual cardinality. You can change them to one-to-many, many-to-one, or one-to-one. Whichever cardinality you set, cross-source relationships have different behavior. You can't use Data Analysis Expressions (DAX) functions to retrieve values on the one side from the many side. You might also see a performance impact versus many-to-many relationships within the same source.

Example of Composite Model

- For an example of a composite model, consider a report that has connected to a corporate data warehouse in SQL Server by using DirectQuery. In this instance, the data warehouse contains **Sales by Country, Quarter, and Bike (Product)** data, as shown in the following image:
- 

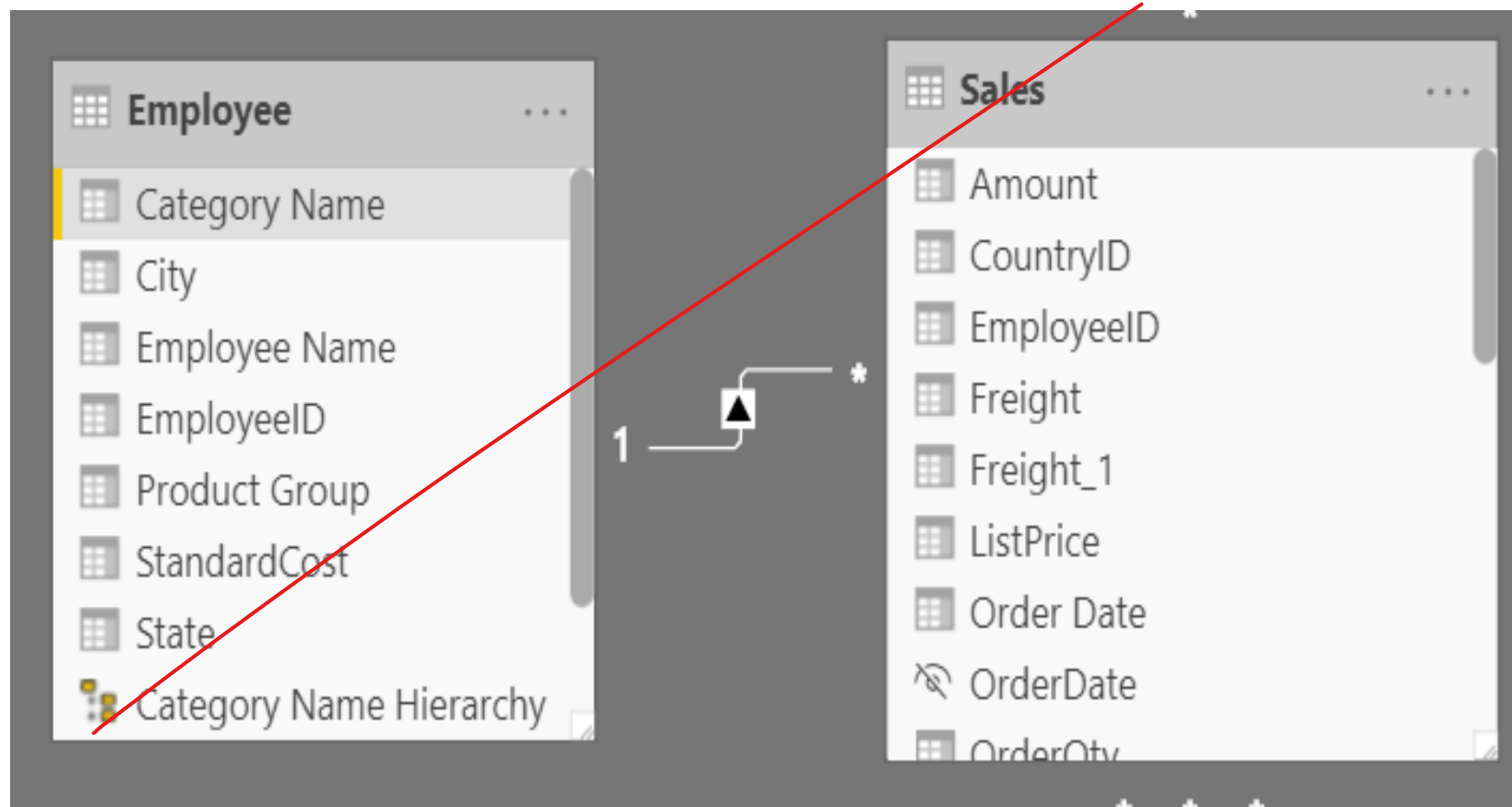
Star Schema

- You can design a star schema to simplify your data. It's not the only way to simplify your data, but it is a popular method; therefore, every Power BI data analyst should understand it. In a star schema, each table within your semantic model is defined as a dimension or a fact table, as shown in
- the following visual



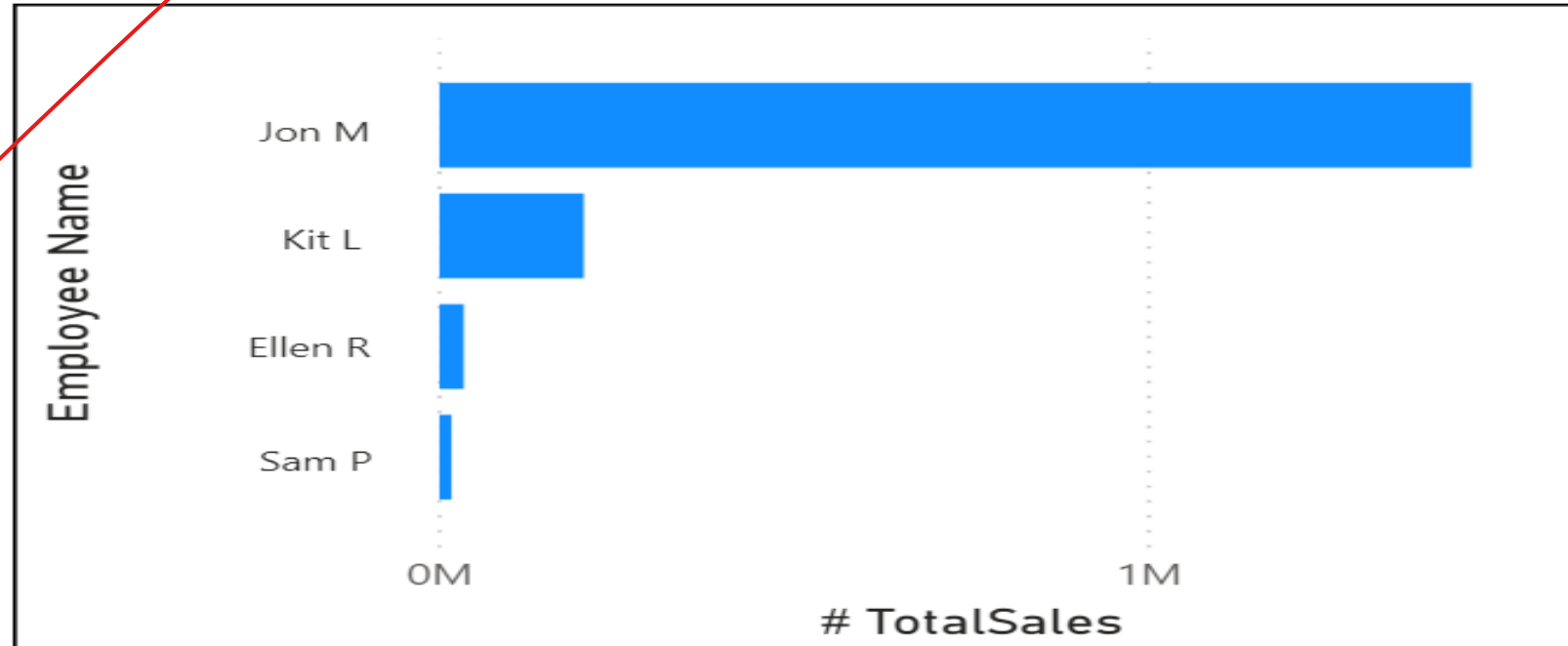
- **Fact tables** contain observational or event data values: sales orders, product counts, prices, transactional dates and times, and quantities. Fact tables can contain several repeated values. For example, one product can appear multiple times in multiple rows, for different customers on different dates. These values can be aggregated to create visuals. For instance, a visual of the total sales orders is an aggregation of all sales orders in the fact table. With fact tables, it is common to see columns that are filled with numbers and dates. The numbers can be units of measurement, such as sale amount, or they can be keys, such as a customer ID. The dates represent time that is being recorded, like order date or shipped date.
- **Dimension tables** contain the details about the data in fact tables: products, locations, employees, and order types. These tables are connected to the fact table through key columns. Dimension tables are used to filter and group the data in fact tables. The fact tables, on the other hand, contain the measurable data, such as sales and revenue, and each row represents a unique combination of values from the dimension tables. For the total sales orders visual, you could group the data so that you see total sales orders by product, in which product is data in the dimension table.
- Fact tables are much larger than dimension tables because numerous events occur in fact tables, such as individual sales. Dimension tables are typically smaller because you are limited to the number of items that you can filter and group on. For instance, a year contains only so many months, and the United States are composed of only a certain number of states.
- Considering this information about fact tables and dimension tables, you might wonder how you can build this visual in Power BI.
- The pertinent data resides in two tables, Employee and Sales, as shown in the following semantic model. Because the Sales table contains the sales order values, which can be aggregated, it is considered a fact table. The Employee table contains the specific employee name, which filters the sales orders, so it would be a dimension table. The common column between the two tables, which is the primary key in the Employee table, is **EmployeeID**, so you can establish a relationship between the two tables based on this column.

Contd.



Contd..

- When creating this relationship, you can build the visual according to the requirements, as shown in the following figure. If you did not establish this relationship, while keeping in mind the commonality between the two tables, you would have had more difficulty building your visual.



Analytic queries

- Analytic queries, also known as analytical queries, are database queries that focus on retrieving and analyzing data to gain insights and support decision-making processes. These queries typically involve aggregating, summarizing, and analyzing large volumes of data to uncover patterns, trends, correlations, and anomalies. Analytic queries are commonly used in business intelligence, data warehousing, and data analytics applications to extract valuable insights from complex datasets. Here are some key characteristics and types of analytic queries:
- **Aggregation:** Analytic queries often involve aggregating data to calculate summary statistics such as sums, averages, counts, min/max values, and other aggregate functions. Aggregation allows users to understand the overall trends and patterns in the data.
- **Grouping:** Analytic queries frequently use grouping operations to organize data into meaningful subsets based on one or more criteria. Grouping enables users to analyze data at different levels of granularity, such as by time periods, geographic regions, product categories, or customer segments.
- **Window Functions:** Window functions are a powerful feature in analytic queries that allow calculations to be performed over a "window" of rows defined by a specified range or partition. Common window functions include ranking, row numbering, cumulative sums, moving averages, and percentiles.

Contd..

- **Joins and Relationships:** Analytic queries often involve joining multiple tables or datasets to combine related information for analysis. Joins establish relationships between tables based on common keys or attributes, allowing users to perform analysis across multiple dimensions.
- **Filtering:** Analytic queries may include filtering operations to narrow down the dataset to specific subsets of data based on specified conditions or criteria. Filtering helps focus the analysis on relevant data and excludes irrelevant or outliers.
- **Advanced Analytics:** Some analytic queries go beyond basic aggregation and include advanced analytical techniques such as predictive modeling, machine learning, clustering, and sentiment analysis. These techniques enable organizations to uncover deeper insights and predictive patterns in their data.
- **Performance Optimization:** Analytic queries often require optimization techniques to improve performance, especially when dealing with large volumes of data. Techniques such as indexing, partitioning, query optimization, and caching are used to enhance query performance and reduce execution times.
- **Visualization and Reporting:** Analytic queries are often used to generate data visualizations and reports that communicate insights effectively to decision-makers. Visualization tools such as charts, graphs, dashboards, and heatmaps are used to represent the analyzed data visually.
- **Ad-Hoc Analysis:** Analytic queries support ad-hoc analysis, allowing users to explore and interact with data dynamically. Users can pose questions, refine queries, and drill down into details to uncover new insights and answer specific business questions on the fly.
- **Data Exploration:** Analytic queries facilitate data exploration by providing interactive tools and capabilities for navigating and exploring large datasets. Users can perform exploratory analysis, discover patterns, and gain a deeper understanding of the underlying data.

Configuring visual reports

- Configuring visual reports in Power BI involves designing and customizing visualizations to effectively communicate insights from your data. Here's a step-by-step guide on how to configure visual reports in Power BI:
- **Select Data Source:** Connect to your data source by importing data into Power BI or establishing a direct query/live connection to the source. Ensure that the data is properly structured and contains the necessary fields for analysis.
- **Choose Visualizations:** Drag and drop visualizations from the Visualizations pane onto the report canvas. Power BI offers a wide range of visualizations including bar charts, line charts, pie charts, maps, tables, matrices, and more. Choose the appropriate visualization type based on the nature of your data and the insights you want to convey.
- **Configure Fields:** Assign data fields to the various elements of the visualization (e.g., axes, values, categories) by dragging them from the Fields pane onto the corresponding fields in the visualization. This defines how the data will be represented and analyzed within the visualization.
- **Customize Visualizations:** Customize the appearance and behavior of visualizations using the formatting options available in the Formatting pane. This includes adjusting colors, fonts, labels, axes, data labels, tooltips, and other visual properties to enhance readability and visual appeal.
- **Add Interactivity:** Add interactivity to visualizations by enabling features such as drill-down, drill-through, cross-filtering, and highlighting. These interactive features allow users to explore and analyze data dynamically by clicking on data points or interacting with slicers and filters.
- **Create Calculations and Measures:** Define calculated columns, calculated tables, and measures using Data Analysis Expressions (DAX) to perform advanced calculations and aggregations on your data. These calculations can be used to derive new insights or customize the behavior of visualizations.

Contd..

- **Apply Filters and Slicers:** Use slicers, filters, and interactions to control the data displayed in visualizations dynamically. Users can filter data based on specific criteria or slice data by different dimensions to focus on relevant subsets of the data.
- **Arrange and Layout:** Arrange visualizations on the report canvas to create a logical flow and narrative for your report. Group related visualizations together, use containers and shapes to organize content, and adjust the layout to optimize space and readability.
- **Add Report Pages:** Create multiple report pages to organize related visualizations and present different aspects of your analysis. Each report page can focus on a specific topic, dataset, or perspective, allowing users to navigate through the report seamlessly.
- **Review and Publish:** Review the visual report for accuracy, completeness, and effectiveness in communicating insights. Once satisfied, publish the report to Power BI Service or share it with others via email, web embedding, or collaboration tools.
- **Schedule Refresh:** If your report relies on data that is regularly updated, schedule automatic data refreshes to keep the report up to date with the latest data from the source.

Semantic models

- Semantic models are the backbone of business intelligence (BI) tools like Power BI, providing a structured representation of data that allows users to analyze and interpret information effectively. Here's a deeper dive into semantic models:

Components of Semantic Models:

- **Entities or Tables:** These represent the different data sources or entities within your organization. For example, you might have tables for sales transactions, customers, products, employees, etc.
- **Attributes or Columns:** Attributes define the characteristics or properties of the entities. Each column within a table represents a specific attribute. For instance, in a sales table, attributes might include sales amount, date, customer ID, product ID, etc.
- **Relationships:** Relationships define how entities or tables are related to each other. These relationships establish connections between tables based on common fields or keys. For example, the relationship between a sales table and a product table might be based on the product ID.
- **Measures:** Measures are calculated values derived from the data in your tables. These can include aggregations (such as sums, averages, counts) or custom calculations. Measures are typically used to answer questions or perform analysis.
- **Hierarchies:** Hierarchies represent ordered levels of data within a single attribute. For instance, a date hierarchy might include levels for year, quarter, month, and day.

Contd..



Benefits of Semantic Models:

- **Consistency:** Semantic models provide a standardized way of organizing and representing data, ensuring consistency across reports and analyses.
- **Ease of Use:** By providing a clear structure and relationship between data elements, semantic models make it easier for users to navigate and understand the data.
- **Flexibility:** Semantic models can accommodate complex data structures and relationships, allowing for a wide range of analysis and reporting scenarios.
- **Scalability:** Semantic models can scale to handle large volumes of data and accommodate changes or additions to the data model over time.
- **Performance:** Well-designed semantic models can improve query performance by optimizing data access and retrieval.
- **Interactivity:** Semantic models enable interactive data exploration and analysis, allowing users to drill down into details or slice and dice data dynamically.
- **Implementation:**
 - In Power BI, semantic models are created using Power BI Desktop, where users can import data from various sources, define relationships, create calculated columns and measures, and build hierarchies. Once the semantic model is defined, users can create reports and visualizations based on the model to analyze and communicate insights effectively.

Functions in DAX

- Functions in DAX (Data Analysis Expressions) are the building blocks used to create formulas and expressions for manipulating and analyzing data within Power BI, Excel Power Pivot, and Analysis Services Tabular models. These functions cover a wide range of functionalities, from basic arithmetic operations to advanced analytical calculations. Here's an overview of the main categories of functions in DAX:

1. Mathematical and Trigonometric Functions:

- These functions perform basic mathematical operations and trigonometric calculations.
- SUM: Calculates the sum of values in a column or expression.
- AVERAGE: Computes the arithmetic mean of values in a column or expression.
- MIN/MAX: Finds the minimum or maximum value in a column or expression.
- SQRT: Computes the square root of a number.
- SIN/COS/TAN: Calculates the sine, cosine, and tangent of an angle, respectively.

DAX[Data Analysis Expression] Query in powerBI

- In Power BI, DAX *formulas* are used to define different types of calculations for your data, but can also be used to define role security. DAX *queries*, on the other hand, can be used to return data from the model.
- DAX queries are similar to SQL queries in that they can show you data you already have. DAX queries don't create items in the model or visuals in the report.
- DAX queries have two main parts:
- An **EVALUATE** statement, which is required. It specifies what and how data is returned in the query.
- A **DEFINE** statement, which is optional. It allows you to define DAX formulas, such as a measure, to use in the query. Measures can be added to the model using CodeLens when used in DEFINE.
- EVALUATE is a **keyword automatically included by Power BI in every query**. It's used to answer and create your visuals in Power BI. It's also used to contain table expressions in a DAX query. To use EVALUATE in a query, you can input EVALUATE and then provide a table expression.
- A **DEFINE** statement, which is optional. It allows you to define DAX formulas, such as a measure, to use in the query. Measures can be added to the model using CodeLens when used in DEFINE.

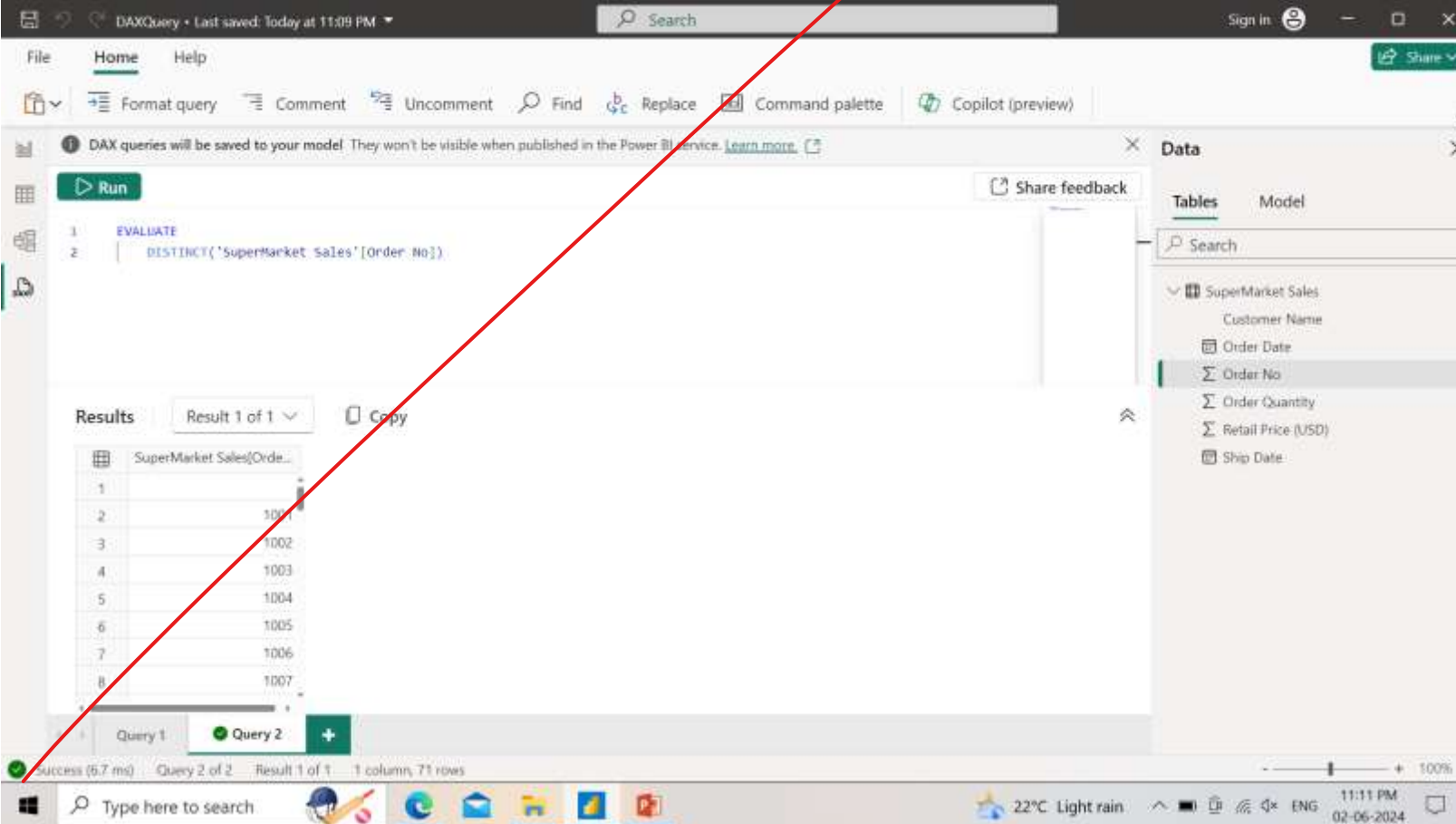
DAX query view layout



Steps to create DAX Query in powerBI

- Step1:open powerBI Desktop
 - Step2:Click on GetData
 - Step3:Click on Excel/CSV File
 - Step4:Click on Load and import data in powerBI
 - Step5:Click on DAX Query View in powerBI Desktop
 - Step6:Click on any of the Fields.Right Click->Quick queries->Show Data preview
 - Step7:DAX Query Evaluate will be Displayed
- ```
EVALUATE DISTINCT('SuperMarket Sales'[Order No])
```
- Step8:Click on Run to get output.

# DAX Query Output



The screenshot displays the DAX Query Editor interface. The query editor shows a DAX query that evaluates the distinct values of the 'Order No' column from the 'SuperMarket Sales' table. The results pane shows a table with 71 rows of distinct order numbers. The status bar at the bottom indicates the query was successful and returned 71 rows.

**DAX Query:**

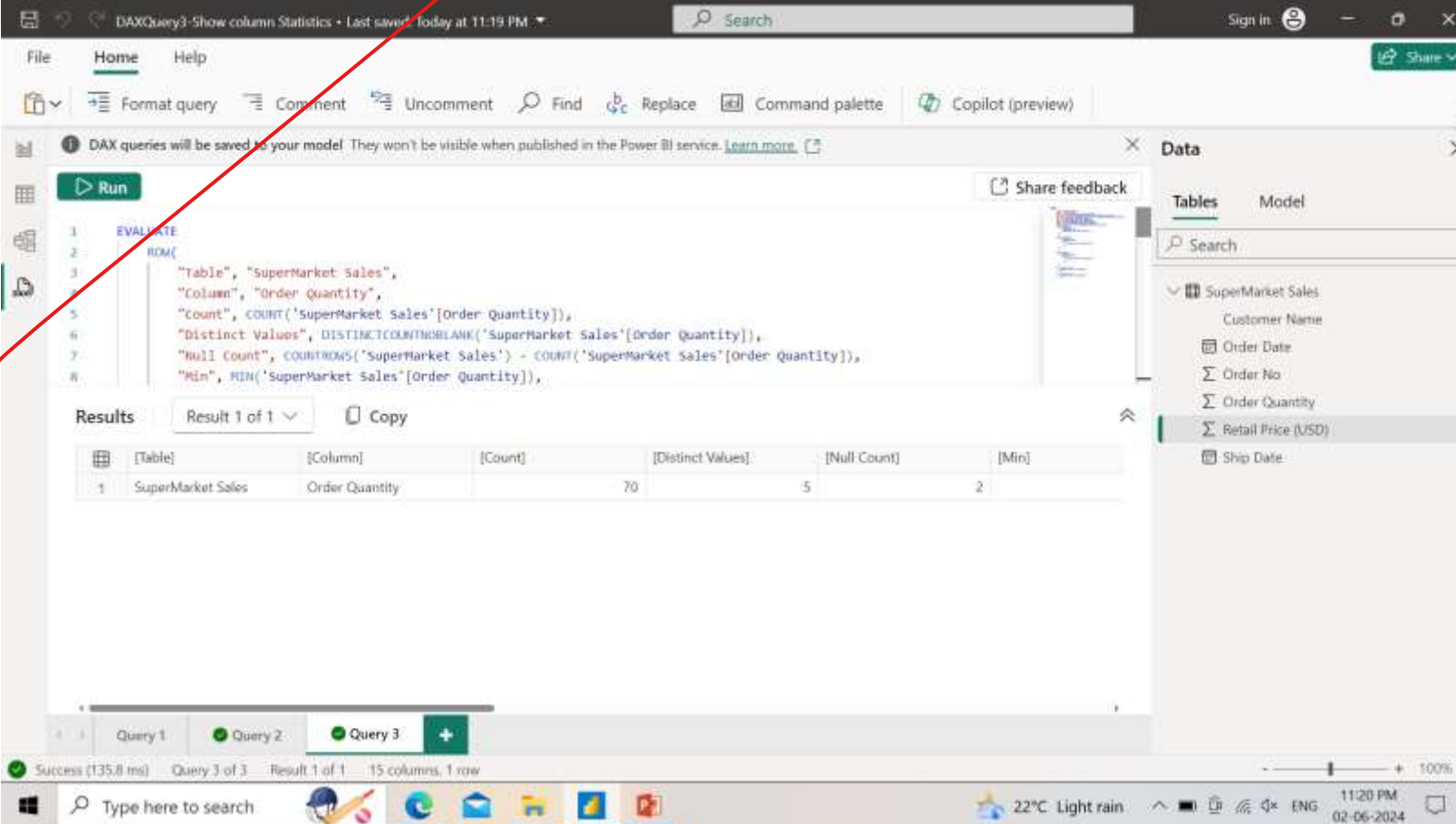
```
1 EVALUATE
2 | DISTINCT('SuperMarket Sales'[Order No])
```

**Results:**

| SuperMarket Sales[Order No] |
|-----------------------------|
| 1                           |
| 2                           |
| 3                           |
| 4                           |
| 5                           |
| 6                           |
| 7                           |
| 8                           |
| 1001                        |
| 1002                        |
| 1003                        |
| 1004                        |
| 1005                        |
| 1006                        |
| 1007                        |

**Query 2 of 2: Result 1 of 1: 1 column, 71 rows**

# Column statistics DAX Query



DAXQuery3-Show column Statistics • Last saved: Today at 11:19 PM

File Home Help

Format query Comment Uncomment Find Replace Command palette Copilot (preview)

DAX queries will be saved to your model. They won't be visible when published in the Power BI service. [Learn more](#)

Run Share feedback

```
1 EVALUATE
2 SUMMARIZE(
3 "Table", "SuperMarket Sales",
4 "Column", "Order Quantity",
5 "Count", COUNT('SuperMarket Sales'[Order Quantity]),
6 "Distinct Values", DISTINCTCOUNT('SuperMarket Sales'[Order Quantity]),
7 "Null Count", COUNTROWS('SuperMarket Sales') - COUNT('SuperMarket Sales'[Order Quantity]),
8 "Min", MIN('SuperMarket Sales'[Order Quantity])
9)
```

Results Result 1 of 1 Copy

|   | [Table]           | [Column]       | [Count] | [Distinct Values] | [Null Count] | [Min] |
|---|-------------------|----------------|---------|-------------------|--------------|-------|
| 1 | SuperMarket Sales | Order Quantity | 70      | 5                 |              | 2     |

Query 1 Query 2 Query 3 +

Success (135.8 ms) Query 3 of 3 Result 1 of 1 15 columns, 1 row

22°C Light rain 11:20 PM 02-06-2024

# Contd..

## 2. Statistical Functions:

- These functions are used for statistical analysis and calculations.
- AVERAGEX: Calculates the average of an expression evaluated for each row in a table.
- MEDIAN: Computes the median of values in a column or expression.
- VAR/STDEV: Computes the variance and standard deviation of values in a column or expression.
- RANKX: Assigns a rank to each row in a table based on the value of an expression.

## 3. Logical Functions:

- These functions evaluate logical conditions and return true or false values.
- IF: Evaluates a condition and returns one value if true and another if false.
- AND/OR: Combines multiple conditions and returns true if all conditions are met (AND) or if at least one condition is met (OR).
- NOT: Negates a logical value (true becomes false and vice versa).



# Contd..

## 4. Text Functions:

- These functions manipulate text strings and perform operations such as concatenation, substring extraction, and case conversion.
- CONCATENATE: Joins two or more text strings together.
- LEFT/RIGHT/MID: Extracts a specified number of characters from the left, right, or middle of a text string.
- UPPER/LOWER: Converts text to uppercase or lowercase, respectively.

## 5. Date and Time Functions:

- These functions work with date and time values, performing operations like date arithmetic, extraction, and formatting.
- DATE: Creates a date value from year, month, and day components.
- YEAR/MONTH/DAY: Extracts the year, month, or day component from a date value.
- DATEDIFF: Calculates the difference between two dates in terms of a specified interval (e.g., days, months, years).
- FORMAT: Formats a date or time value using a specified format string.

# Contd..

## 6. Aggregate Functions:

- These functions aggregate values across rows or columns, taking into account any filters applied to the data.
- SUMX: Calculates the sum of an expression evaluated for each row in a table.
- AVERAGEX: Calculates the average of an expression evaluated for each row in a table.
- COUNT/COUNTX: Counts the number of non-blank values in a column or expression.

## 7. Filter Functions:

- These functions apply filters to data, either by modifying the filter context or by filtering data directly.
- FILTER: Filters a table based on a specified condition.
- ALL/ALLEXCEPT/ALLSELECTED: Modifies the filter context to remove filters from specific columns or tables.
- RELATED/RELATEDTABLE: Retrieves related values from a related table based on the current context.

## 8. Time Intelligence Functions:

- These functions facilitate time-based analysis, allowing for calculations such as year-to-date, month-over-month, and moving averages.
- TOTALYTD: Calculates the year-to-date total of a specified expression.
- DATESBETWEEN: Returns a table of dates that fall within a specified range.
- SAMEPERIODLASTYEAR: Returns values from the same period in the previous year.
- These categories cover the main types of functions in DAX, but there are many more functions available for various purposes. Mastery of DAX functions is essential for building sophisticated data models and deriving valuable insights from your data.

# Unit-3 Experiments

- 1.Power BI DirectQuery Model
  - 2.Composite Query Model
  - 3.DAX
- 