Power BI Consolidated Questions

1. What is Power BI?

Microsoft developed Power BI, a **business analytics tool** that transforms data from multiple data sources into valuable and interactive insights. It allows user to connect to various data sources, transform and manipulate data, creates interactive reports and dashboards and also shares insights with others.

2. Why should we use Power BI?

Power BI makes it simple for everyone including non technical people to connect, change and visualize raw business data from various sources by transforming it into valuable data that enables them to make proper business decisions.

3. What are the differences between Power BI and Tableau?

Power BI	Tableau
Uses DAX for calculating measures.	Uses MDX for measures and dimensions.
Capable only to handle a limited amount of data.	Capable of handling large amount of data.
Suitable for beginners and experts as well.	Best suited for experts.
UI is simple.	UI is complex.

4. Power BI can connect to which data sources?

Datasource is the point from where the data has been retrieved.

.xlsx(Excel file), .csv(Comma Separated Value), .pbix(Power BI), .XML, database (SQL, Azure HD Insight)

5. What is Power BI desktop?

It allows users to easily connect, transform and visualize data. It allows users to create visuals and reports within their organization. It gets update in almost every month.

Why Use Power BI Desktop?

- **Free to use** – No cost for report creation.
- **Easy to learn** – Drag-and-drop interface, no coding needed.
- **Powerful Data Analysis** – Supports complex DAX calculations.

- **Interactive Reports** – Filters, drill-throughs, and dynamic dashboards.
- **Seamless Integration** – Works with **Excel, SQL Server, Azure, and more**

6. What are available views in Power BI?

- **Report View** – It displays tables interactively, enabling easy analysis and visualization across reports.
- **Table View** – The table view in Power BI displays data in rows and columns, allowing sorting, filtering, conditional formatting and detailed analysis.
- **Model View** – Users can manage relationships between the tables. Supports **creating, editing, and managing relationships** (one-to-many, many-to-many).
- **DAX** – DAX in Power BI enables custom calculations, measures, calculated columns and time based functions for dynamic data analysis.

7. What is Power Query?

Power Query is a **data transformation and ETL (Extract, Transform, Load)** tool in Power BI. It allows you to import data from various data sources and will enable you to clean, transform and reshape your data as per requirement. Power query allows you to write your query once and then run it with a simple refresh. It uses **M Query** Language for calculations.

Example Use Case:

You import sales data from an Excel file but need to remove null values and split full names into first and last names before loading it into Power BI.

8. What is Power Pivot?

Power Pivot is a **data modeling** tool that helps create relationships between tables and build calculated measures using DAX (Data Analysis Expressions). It creates relationships between tables (Fact & Dimension tables).

Example Use Case:

After cleaning data **with Power Query**, you use Power Pivot to create a calculated column for profit (Sales_Amount - Cost) and define relationships between Sales and Customers tables.

9. What are the differences between Power Pivot and Power Query?

Power query is recommended experience for importing data. Power pivot is great for modeling the data you have imported.

With power query, you can locate data sources, make connections, and then shape that data (for example, remove a column, change a datatype or merge table) in such a way that meet your needs. Then you can load your query into excel to create charts and reports. There are 4 phases to using power query

- **Connect** – Import and make connections to data in the cloud on a service or locally.
- **Transform** – Shape the data to meet your needs, while the original source remains unchanged.
- **Combine** – Further shape data by integrating it from multiple sources to get a unique view into the data.
- **Load** – Complete your query and save it into a worksheet or data model.

When you put your data in an excel data model, you can continue enhancing it by performing analytics in Power Pivot.

What you can do in Power Pivot is:

- Create relationship between tables
- Add calculated column and measures with DAX in Power Pivot
- Create KPI in Power Pivot.
- Create perspective
- Organize fields in hierarchies.

Power Query	Power Pivot
To clean data	To do data modeling
Uses M function	Uses DAX function
Requires more memory for M function	Requires less memory for DAX function

When to Use Which?

- **Use Power Query** when you need to **clean, reshape, or merge data** from different sources before analysis.
- **Use Power Pivot** when you need to **analyze data, create relationships, and perform calculations** using DAX.

10. What are the advantages of Power BI?

- It facilitates the creation of interactive data visualization in data centers.
- It enables users to convert data into visuals and share them with others.
- It connects excel queries and dashboards for faster analysis.

- It offers quick and accurate solutions.

11. What are the difference between Power BI and Excel?

Parameter	Power BI	Excel
Primary Use	Data visualization, interactive dashboards, BI reporting	Data analysis, calculations, tabular reporting
Data Refresh	Supports **automated refresh** (on Power BI Service)	Requires manual refresh unless using VBA or Power Query
Data Modeling	Powerful with **DAX**, relationships, and star schemas	Basic modeling with formulas and tables
Visualization	Advanced, interactive visuals and dashboards with real-time updates	Basic charts and static graphs
Cross-Filtering	Offers advanced features in cross-filtering between charts	Doesn't offer advanced features in cross-filtering between charts

Which Should You Choose?

- Choose **Power BI** if you need **dynamic reporting**, **real-time dashboards**, and **data from multiple sources**.
- Stick with **Excel** for **flexible, quick analyses**, especially when working **individually** or for **financial reports**.

12. What is dashboard?

The dashboard is like a single page canvas on which you have various elements to create and visualize reports created by analyzing data. It provides a quick overview of relevant information allowing users to monitor performance, track progress and make data driven decisions at a glance.

13. What are the building blocks of Power BI?

- **Datasets** – It is a collection of data gathered from various sources like SQL Server, Azure, XML,JSON and many more. We can easily fetch any data from any datasource by **getdata**, which is a feature of Power BI.
- **Visualization** – It is the visual aesthetic representation of data in the form of maps, charts and tables. Power BI provides various types of visualizations, such as - Bar charts, Pie charts, Line graphs, Maps, KPI cards etc
- **Reports** – Reports are structured representation of datasets that consist of multiple pages. Reports help to extract important information and insights from dataset to make major business decisions.

- **Dashboard** – A **dashboard** is a high-level, single-page view that consolidates key visualizations from multiple reports.

14. What are the differences between report and dashboard?

Feature	Report	Dashboard
Pages	Can have multiple pages	Single page summery only
Data Source	Uses one dataset per report	Can pull data from **multiple datasets & reports**
Where Created?	Power BI Desktop & Service	Only in Power BI Service
Purpose	Detailed analysis	High-level summary
Customization	Visuals are highly customizable	fixed: limited design option
Best For	Best for detailed analysis	Best for high level, quick insight
Shared as	Shared as full report	Shared as single page view

When to Use Which?

- **Use a Report** when you need **detailed analysis** with multiple pages and filters.
- **Use a Dashboard** when you need a **high-level summary** from multiple reports on a **single page**.

15. What are the various Power BI versions?

- **Power BI Desktop (Free)** – The free tool that connects data sources, transforms data and creates interactive visual reports. It is the best for individual users who create reports. But it cannot share reports without Power BI Pro.
- **Power BI Pro ($10 per user/month)** – with pro version, you get full access to the Power BI Dashboard. Allows sharing and collaboration in Power BI Service. But it cannot handle **large-scale datasets** like Premium.
- **Power BI Premium (Starts at $20 per user/month or $4,995 per capacity/month)** – The premium version is designed for larger organizations, offering dedicated storage for each user (Storage can be 50GB to 100TB). It allows free users to view shared reports.
- **Power BI Service (Cloud-Based)** – A **web-based platform** for hosting and sharing reports, which allows **real-time collaboration** and automatic data refresh. It requires a **Power BI Pro** or **Premium license** for sharing.

16. What do you mean by content pack in Power BI?

A content pack in Power BI is a collection of pre-built ready to use data connection reports or dashboards created by Microsoft or 3rd party providers.

This allows users to quickly connect to specific data sources and gain insights without building reports from scratch. Content packs are commonly used for popular services like google analytics, Salesforce, MS dynamics.

Benefits of Content Packs

- **Saves Time** – No need to build dashboards from scratch.
- **Easy to Share** – Quickly distribute insights across teams.
- **Customizable** – Users can modify imported reports.
- **Standardized Reporting** – Ensures consistency across the organization.

17. Name some commonly used tasks in the power query editor

- Connect to data
- Group rows
- Create custom columns
- Shape and combine data
- Pivoting and unpivoting columns
- Query formulas
- Filling Down or Up

18. What are filters in Power BI?

Filters are mathematical and logical conditions applied to data to filter out unnecessary information and displaying only the relevant information user need.

There can be 3 level of filters

- **Visual level** – applies only to a specific visualization without affecting other visuals on the report page.
- **Page Level** – applies to all visuals on a single report page but does not affect visuals on other pages. This is useful when you want to filter an entire page's data without changing the rest of the report.
- **Report Level** – applies to all pages in a report. This is useful when you want to maintain consistency across different pages without applying filters manually on each one.

19. What are the slicers in Power BI?

Slicers in Power BI are visual filters that allow users to dynamically filter data in reports and dashboards. They provide an interactive way to segment data and improve the user experience. With the help of slicers, you are allowing the end user to filter out anything that is present on the report.

Types of Slicers in Power BI

- **List Slicer** – Displays items as a list with checkboxes.
- **Dropdown Slicer** – Saves space by displaying a collapsible dropdown list.
- **Date Slicer** – Filters data using a date range (slider, before/after, or fixed period).
- **Numeric Range Slicer** – Allows filtering within a range of numbers.
- **Hierarchy Slicer** – Filters based on multi-level fields (e.g., Region → Country → City).
- **Relative Date Slicer** – Filters dynamically (e.g., last 30 days, next quarter).

Example Use Case

Scenario: You have a sales report and want users to filter data by region.
Solution: Add a **Hierarchy Slicer** with **Country → State → City** to allow users to drill down into specific locations.

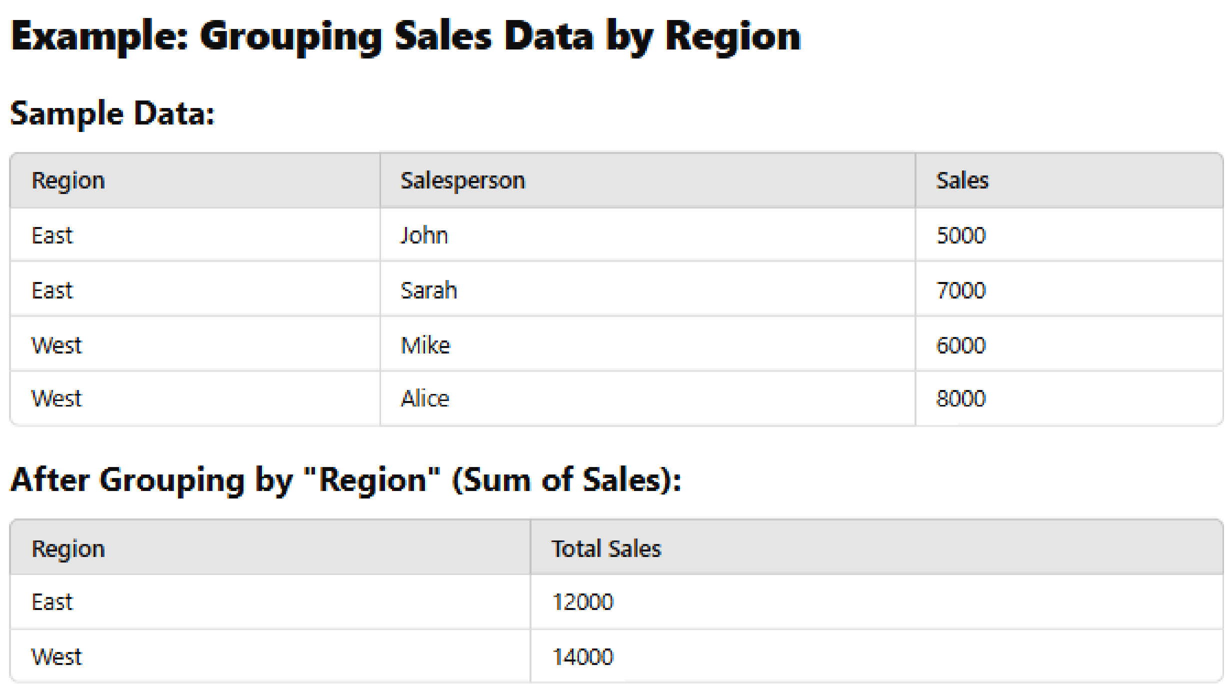
20. Where is data stored in Power BI?

There are two sources to store the data

- **Azure Blob Storage** – when users upload the data, it gets stored here.
- **Azure SQL Database** – All metadata and system artifacts are stored here. They are stored here as either fact table or dimensional table.

21. What do you mean by Group By?

Group By is an operation that allows you to summarize data by grouping rules based on one or more columns. This is often used to calculate aggregate values, like SUM, AVG, COUNT or MAX, MIN value for each group.



Example: Grouping Sales Data by Region

Sample Data:

Region	Salesperson	Sales
East	John	5000
East	Sarah	7000
West	Mike	6000
West	Alice	8000

After Grouping by "Region" (Sum of Sales):

Region	Total Sales
East	12000
West	14000

22. What are some common mistakes that lead to slow down report load times and how to avoid them?

Common mistakes:

- Overloading with unnecessary data, importing large data sets with unnecessary columns or rows slows down the report performance
- Complex or unoptimized DAX formulas, such as nested if statements or excessive measures can slow down performance.
- Excessive visuals, specially complex ones slow down load times

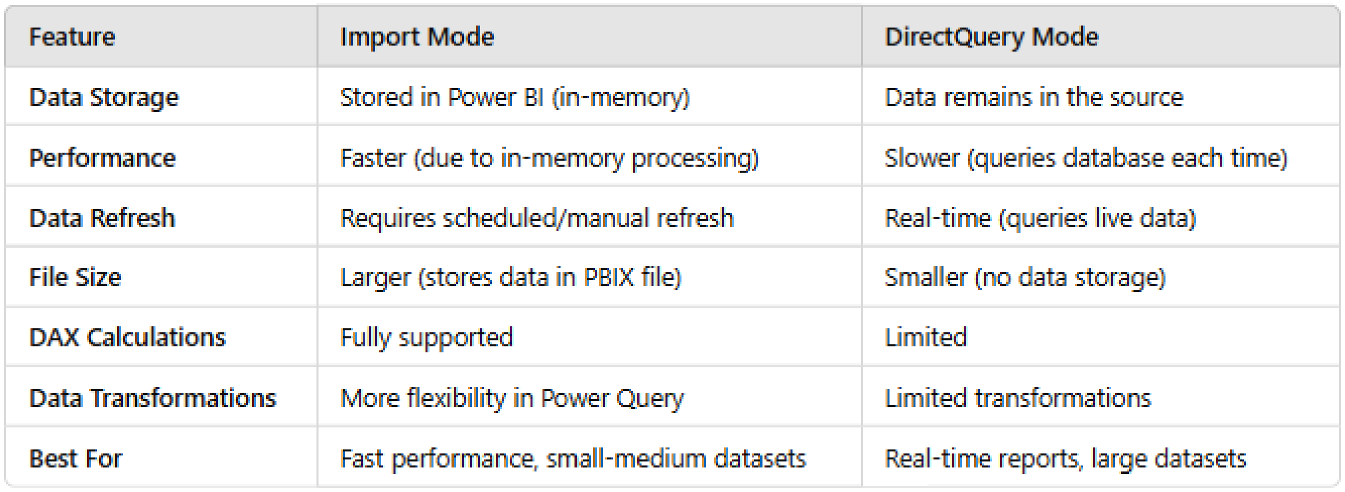
How to avoid them:

- Use **DAX measures**, not **calculated columns**.
- Reduce **columns & rows** in **Power Query**.
- Use **Import Mode** instead of Direct Query (when possible).
- Optimize **DAX functions** for performance.
- Limit **visuals per page** for fast rendering.
- Follow **Star Schema** for better data modeling.

23. What are the difference between Import mode and Direct query?

When ever you are bringing data in Power BI, you can do it in 2 ways.

- **Import Mode**
    - **(i)** Data is fully imported into Power BI's in-memory engine
    - **(ii)** Provides better performance due to in-memory processing
    - **(iii)** Supports DAX calculations, relationships, and transformations
    - **(iv)** Does NOT refresh in real-time (requires manual/scheduled refresh)
    - **(v)** **Best for**: Small to medium datasets, fast performance, offline analysis
    - **(vi)** **Example**: Importing an Excel file into Power BI for sales analysis.
- **Direct Query Mode**
    - **(i)** Data remains in the source and is queried in real-time
    - **(ii)** Provides up-to-date data without refresh
    - **(iii)** No need to store data in Power BI, reducing file size
    - **(iv)** Slower performance compared to Import Mode, as each interaction sends a query to the database
    - **(v)** **Best for:** Large datasets, real-time analytics, live dashboards
    - **(vi)** **Example:** Connecting Power BI to an SQL database to track live inventory levels.



Feature	Import Mode	DirectQuery Mode
Data Storage	Stored in Power BI (in-memory)	Data remains in the source
Performance	Faster (due to in-memory processing)	Slower (queries database each time)
Data Refresh	Requires scheduled/manual refresh	Real-time (queries live data)
File Size	Larger (stores data in PBIX file)	Smaller (no data storage)
DAX Calculations	Fully supported	Limited
Data Transformations	More flexibility in Power Query	Limited transformations
Best For	Fast performance, small-medium datasets	Real-time reports, large datasets

When to Use Which Mode?

Use **Import Mode** When:

- You need fast performance and interactive dashboards.
- Your dataset is small or medium-sized.
- You need complex DAX calculations and transformations.

Use **Direct Query Mode** When:

- You need real-time data updates.
- Your dataset is too large to fit in Power BI's memory.
- Your organization has strict data security and governance policies (data stays in the source).

24. What is the comprehensive working system in Power BI?

Power BI working system mainly comprises 3 steps

- **Data Integration** – First step is to extract and integrate the data from heterogeneous data sources.
- **Data Processing** – Once the data is assembled and integrated it requires some cleaning up, raw data is not useful therefore. A few transformation and cleaning operations are performed on the data to remove unnecessary data. After the data is transformed, it is stored on data warehouse.
- **Data Presentation** – The data is transformed and cleaned, it is visually represented on the Power BI desktop or reports or dashboards or scorecards. These reports can be shared to various business users.

25. What are the types of visualizations in Power BI?

Visualization is a graphical representation of data. We can use visuals to create reports and dashboards.

Bar & Column Charts

- **Stacked Bar Chart** – Shows total sales while dividing them by regions or product category.
- **Stacked Column Chart** – Shows total revenue while dividing it into product categories.
- **Clustered Bar Chart** – Compare sales data across multiple years for different products.
- **Clustered Column Chart** - Compare sales performance across different products over multiple years.
- **100% Stacked Bar Chart** – Shows the percentage contribution of different brands to total sales.
- **100% Stacked Column Chart** – Shows the percentage contribution of different products to total sales.

Line & Area Charts

- **Line Chart** – Shows revenue changes over months or years.

- **Area Chart** – Tracks sales, stock prices, or weather patterns.
- **Stacked Area Chart** – Shows how different product categories contribute to overall revenue.

Pie and Donut Charts

- **Pie Chart** – Shows the percentage share of different companies.
- **Donut Chart** – Similar to pie chart, but with a hole in the middle.

Table & Matrix

- Table – Displays raw data in tabular format.
- Matrix – Similar to a pivot table, allowing hierarchy-based analysis.

Card & KPI Visuals

- **Card** – Displays a single important metric.
- **Multi-row c**ard – Displays multiple key values.
- **KPI (Key Performance Indicator)** – Tracks progress toward a goal.

  Key Components of a KPI in Power BI

  - **Indicator** – The actual value (e.g., current sales, revenue, profit).
  - **Target (Goal)** – The expected or benchmark value to compare against.
  - **Trend (Direction)** – A visual indicator (arrow, color, etc.) showing performance improvement or decline over time.

Scatter & Bubble Charts

- **Scatter Chart** – Shows relationships between two numerical variables.
- **Bubble Chart** – Similar to scatter, with a third variable represented by size.

Best Practices for Power BI Visuals

- Use **bar/column charts** for comparisons.
- Use **line charts** for trend analysis.
- Use **pie charts sparingly** (not more than 5-6 categories).
- Use **tables only when necessary** (focus on visuals instead).
- Use **filters & slicers** for better interactivity.

26. What are custom visuals in Power BI?

Power BI allows customized visualizations like charts and KPIs. It refrains the developers from creating it from scratch using JQuery. Once a custom visual is ready it is tested thoroughly and post testing they are packed in .pbi file format and shared.

Types of Custom Visuals in Power BI

- **Certified Custom Visuals** – Approved by Microsoft and secure for use in Power BI Service. Example – **Bullet Chart** (for progress visualization), **Gantt Chart** (for project timelines) etc.
- **Non-Certified Custom Visuals** – Developed by third-party vendors but not certified by Microsoft. May have additional features but cannot be exported or used in subscriptions. Example – **Network Graph** (Shows relationships between nodes)**,** **Advanced Heat Map** (Customizable data density visualization) etc
- **Custom Developed Visuals** – Built by developers using **Power BI Visuals SDK**. Can be used for internal business needs or shared on AppSource. Example – **Custom Gauge Chart** (for unique KPI tracking), **Real-time IoT Dashboards** (for sensor monitoring), **Advanced Org Chart** (for company hierarchy)

27. What are the major components of Power BI?

- **Power Query** – **ETL (Extract, Transform, Load) tool** for **cleaning and shaping data. It** uses **M Query language** for advanced transformations.
  Example: Removing duplicate rows from an Excel dataset before loading it into Power BI.

- **Power Pivot** – It is a data modeling engine that uses DAX for calculation and creates relationships between tables.

  Example: Creating a **calculated column** for "Profit Margin" in a financial report

- **Power View (Data Visualization Layer)** – It creates interactive charts, tables, and graphs.

  Example: A marketing team builds a **heatmap to visualize customer demographics**.

- **Power BI Desktop** – It integrates power query, view and pivot for advances queries, modeling and reporting. It allows importing data from multiple sources (Excel, SQL, APIs, etc.)**.**

  Example: Analysts use Power BI Desktop to build an interactive sales report.

- **Power BI Service (Cloud-Based BI)** – Used for **publishing, sharing, and collaborating on reports. It** allows **role-based access control (RLS) and supports scheduled data refresh.**

  Example: A manager views real-time business dashboards via the Power BI Service.
- **Power BI Mobile App** – Displays report on IOS or Android.
  Example: A sales executive checks updated revenue charts on their phone.
- **Power Map** – 3D geo-spatial data visualization.
- **Power Q&A (Natural Language Query)** – Allows users to ask questions in natural language. Uses AI to **interpret questions and generate insights automatically** and works in **Power BI reports and dashboards.**
  Example: Typing **"What were last month's total sales?"** and getting an instant chart.

Section wise questions

Power Query Editor (ETL)

1. What are the transformations used in your project?

Text transformation, number transformation, date transformation and along with that we had conditional formatting, conditional changes and so on.

Joining the table or appending the data or we are applying some conditions, we are replacing some values all are transformation.

2. What are the difference between append and merge?

When we combine 2 datasets, we basically apply join.

Join in SQL is equivalent to **merge** in Power BI.

When we want to append the data one after another, we should go with **append**.

3. What is query folding in Power BI?

When you try to fetch some data from database, Microsoft Power BI sends a query to the database to read the data and then it gets back to the Power BI, Microsoft Power BI uses **M Query**.

When you try to read the data from database, then it uses **SQL** language.

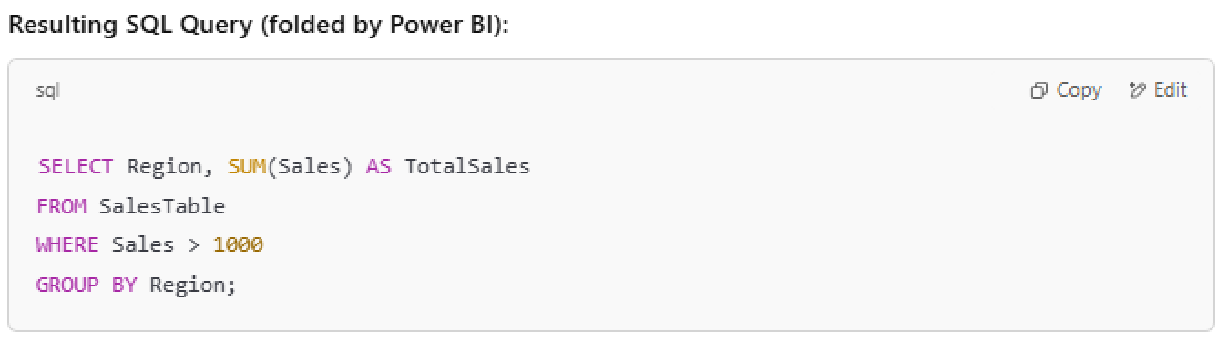
Converting the process from **MQquery to SQL** and from **SQL to MQuer**y is known **as query folding.**

Example Query Folding Process:

Scenario: Connecting to a **SQL Server** database and applying transformations.

Transformation steps:

1. **Filter rows** where Sales > 1000.
2. **Remove unnecessary columns**.
3. **Group data** by Region and calculate **total sales**.



Resulting SQL Query (folded by Power BI):

```sql
SELECT Region, SUM(Sales) AS TotalSales
FROM SalesTable
WHERE Sales > 1000
GROUP BY Region;
```

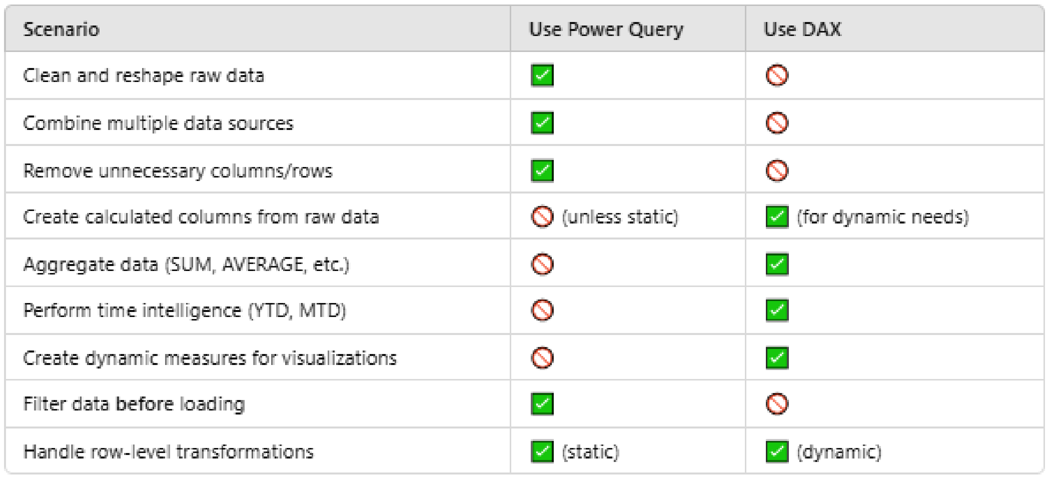
When Does Query Folding Occur?

- Filtering rows
- Removing columns
- Renaming columns
- Grouping data
- Merging queries (joins)
- Aggregations (sum, average)

4. What are the differences between Power Query and DAX?

Parameter	Power Query	DAX
Purpose	Used for **data preparation & transformation** (ETL).	Used for **data modeling, calculations**, and **aggregations**.
Language Used	M language (functional language)	DAX language (formula language, similar to Excel)
When it runs	Runs during **data load** and **refresh**.	Runs **after data load** during **reporting** and **visualization**.
Key functions	Filtering, merging, unpivoting, shaping, data cleaning.	Measures, calculated columns, row context, aggregations, time intelligence.

🎯 When to Use Power Query vs DAX



Scenario	Use Power Query	Use DAX
Clean and reshape raw data	✅	🚫
Combine multiple data sources	✅	🚫
Remove unnecessary columns/rows	✅	🚫
Create calculated columns from raw data	🚫 (unless static)	✅ (for dynamic needs)
Aggregate data (SUM, AVERAGE, etc.)	🚫	✅
Perform time intelligence (YTD, MTD)	🚫	✅
Create dynamic measures for visualizations	🚫	✅
Filter data before loading	✅	🚫
Handle row-level transformations	✅ (static)	✅ (dynamic)

5. What is fact table and dimensional table? Explain star schema and snowflake schema also.

Suppose we are checking a grocery shops database, where they keep details of customer, and which product is sold and who bought those.

Transaction Table or Main Table

Transaction_Date	Item_Name	Item_Catagory	Price/KG	Order_Weight(KG)	Customer_Name	Customer_Email	Customer_Address
12-Feb-25	white potato	vegetable	60	2.2	Ram Kumar	ram@gmail.com	New Town
13-Feb-25	broccoli	vegetable	120	1.5	Ram Kumar	ram@gmail.com	New Town
14-Feb-25	banana	fruit	200	3	Shyam Kumar	shyam@gmail.com	Barrackpore
15-Feb-25	grapes	fruit	240	0.5	Shyam Kumar	shyam@gmail.com	Barrackpore
16-Feb-25	spinach	vegetable	110	1.2	Jodu Mondal	jodu@gmail.com	Tollygunj
17-Feb-25	apple	fruit	180	2.5	Ram Kumar	ram@gmail.com	New Town
18-Feb-25	broccoli	vegetable	120	2.3	Modhu Kumar	modhu@gmail.com	Shyambazar
19-Feb-25	banana	fruit	200	8	Bimal Mondal	bimal@gmail.com	Dumdum
20-Feb-25	grapes	fruit	240	4	Ram Kumar	ram@gmail.com	New Town

This is the total table, or you can say Fact table, but this is highly **denormalized** database because everything is stored in a single row. Here the problem is, suppose the address or email id of "Shyam Kumar" is changed, so we have to change it in multiple places which becomes hectic. So people usually creates separate table for different attributes, like customer details, Sales and so on, those are called dimension table.

Fact Table – Complete business data.

Dimension Table – Lookup data.

Customer Table

Customer_ID	Customer_Name	Customer_Email	Customer_Address
401	Ram Kumar	ram@gmail.com	New Town
402	Shyam Kumar	shyam@gmail.com	Barrackpore
403	Jodu Mondal	jodu@gmail.com	Tollygunj
404	Modhu Kumar	modhu@gmail.com	Shyambazar
405	Bimal Mondal	bimal@gmail.com	Dumdum

Item Table

Item_ID	Item_Name	Item_Catagory	Price/KG
21	white potato	vegetable	60
22	broccolli	vegetable	120
23	spinach	vegetable	110
24	banana	fruit	200
25	grapes	fruit	240
26	apple	fruit	180

Sales Table

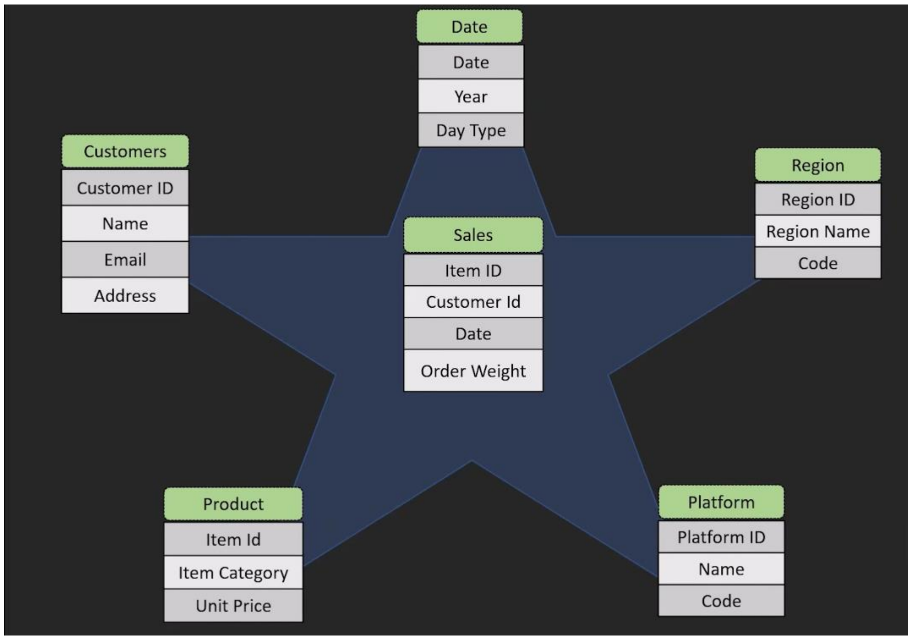
Transaction_Date	Item_ID	Customer_ID	Order_Weight
12-Feb-25	21	401	2.2
13-Feb-25	22	401	1.5
14-Feb-25	24	402	3
15-Feb-25	25	402	0.5
16-Feb-25	23	403	1.2
17-Feb-25	26	401	2.5
18-Feb-25	22	404	2.3
19-Feb-25	24	405	0.8
20-Feb-25	25	401	0.4

Date table

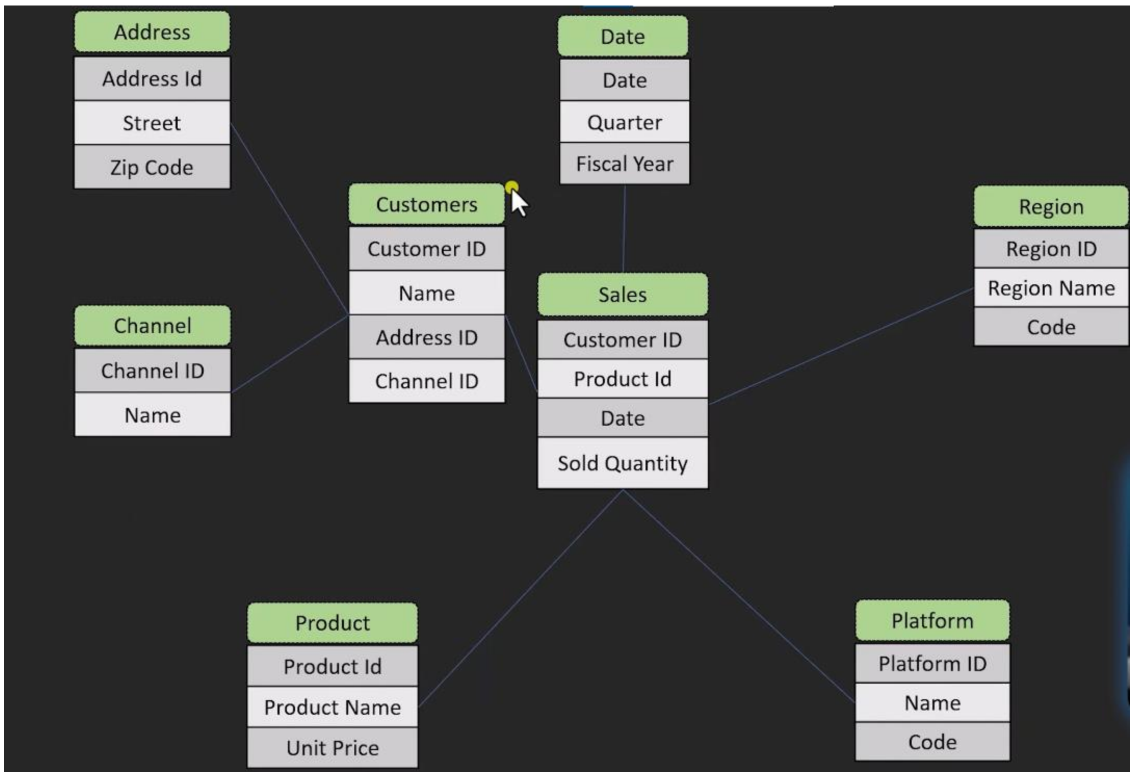
Date	Year	Month	Day Type
12-Feb-25	2025	February	Weekday
13-Feb-25	2025	February	Weekday
14-Feb-25	2025	February	Weekday
15-Feb-25	2025	February	Weekend
16-Feb-25	2025	February	Weekend
17-Feb-25	2025	February	Weekday
18-Feb-25	2025	February	Weekday
19-Feb-25	2025	February	Weekday
20-Feb-25	2025	February	Weekday

So basically the **fact table** (Transactional table or main table) stays in the middle and in the side all we have is **dimensional table** (Customer, Item, Sales and Date table).

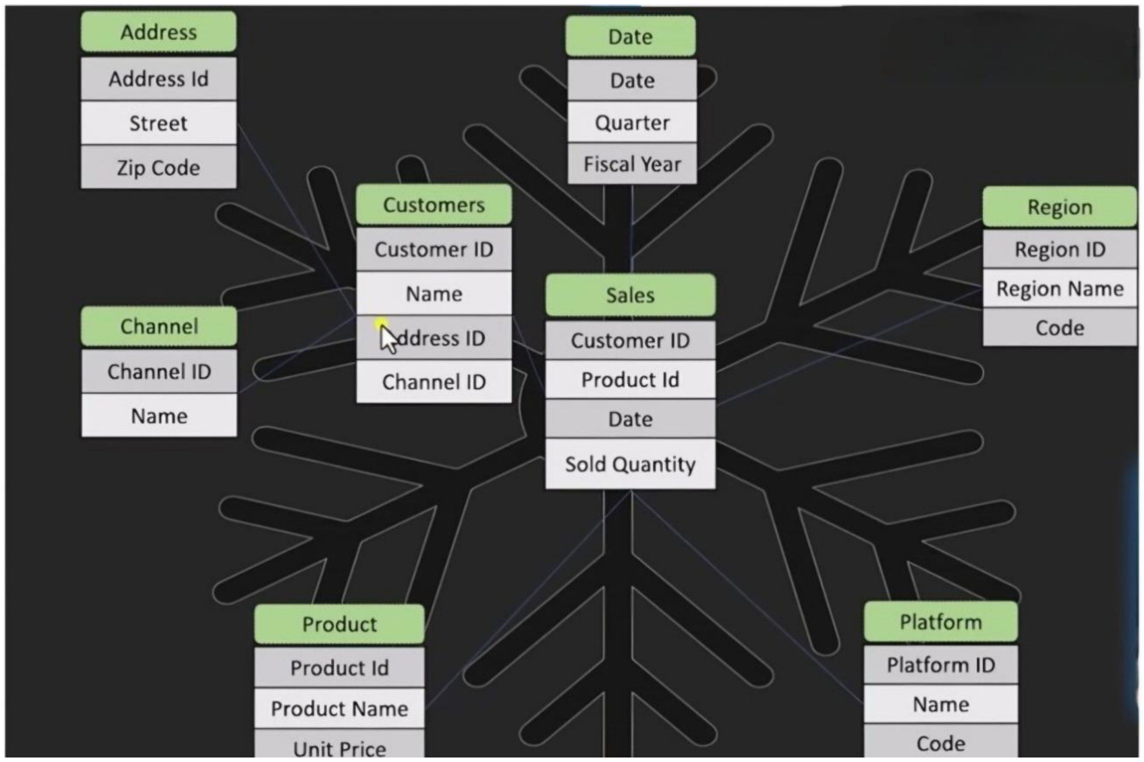
When all the dimension table is connected to a fact table, which can be resembled in shape of a star, it becomes **star schema**. It performs better because **joins are simple** here.



When some of the dimension table can have another dimension table connected to it or a fact table can be connected to a dimension table indirectly it becomes snowflake. **Joins are complex** here, takes more time for optimization.



Looks like snowflake, that is why it is called **snowflake schema**.



Power View / Power BI Desktop

1. What are the charts you have used in your project?

We commonly used column chart, pie chart, donut chart, cards, multi-cards, KPI maps, waterfall charts etc.

Example: I had a situation where I wanted to compare month and month – waterfall chart

There was some key numbers which I wanted to point out – cards.

2. What is Bookmark in Power BI?

A bookmark captures the state of a report page. It includes the changes you made to filters, slicers, and visuals on that page. It enables users to quickly navigate between different report views without manually adjusting filters or visuals each time.

3. How can you make your reports dynamic?

By making use of bookmarks, by selecting certain visuals and by also making use of buttons and actions you can make your report dynamic.

Power BI Service

1. Which Power BI license you have used in your project?

There are some features which you will get in pro and some in premium. Premium access will not be given to everyone. Lead used **premium** version and we used normal **pro** version.

2. What is the difference between my workspace and workspace?

This is basically the difference between rough notebook and class book. Rough notebook is something you keep it for a rough work and class book is something you write in neat format and that will be shown to many people.

My Workspace is for my personal use, I will be able to see all the reports has been put into the workspace. I will be only eligible person who can view or edit that.

Workspace is generically for entire collaboration of the team. Multiple people can go and work and publish the report into one. It is common for everyone for one project. You can have one or more workspaces where all the team members can go and work.

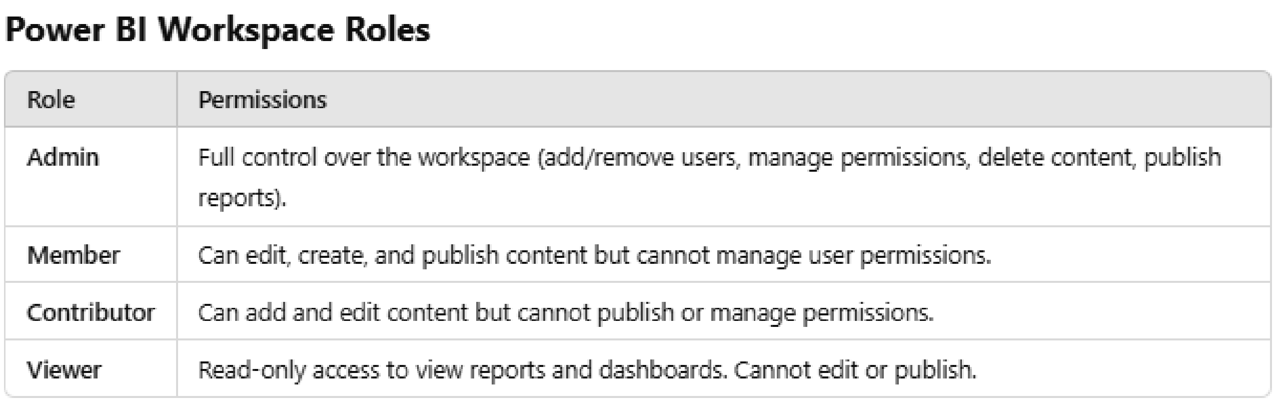
3. What are the roles available in workspace?

- Admin
- Contributor
- Member
- Viewer

Admin is a person who can access anything, who can add as many people as he wants and he will be having highest access whereas the lowest access will be given to **Viewer**, he will only be able to view and he will not be in the position to add anything.

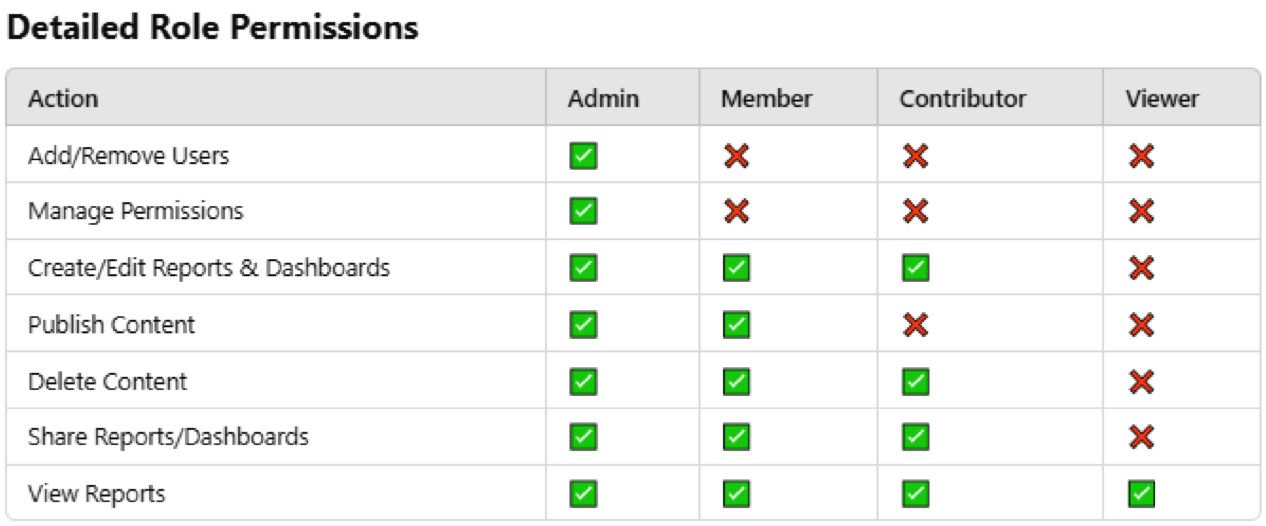
A **Contributor** is responsible for reports, he can add features to the report.

A **Member** is having almost equal roles, but he will be one position higher than the **Contributor**, he can add people.



Power BI Workspace Roles

Role	Permissions
Admin	Full control over the workspace (add/remove users, manage permissions, delete content, publish reports).
Member	Can edit, create, and publish content but cannot manage user permissions.
Contributor	Can add and edit content but cannot publish or manage permissions.
Viewer	Read-only access to view reports and dashboards. Cannot edit or publish.



Detailed Role Permissions

Action	Admin	Member	Contributor	Viewer
Add/Remove Users	✅	❌	❌	❌
Manage Permissions	✅	❌	❌	❌
Create/Edit Reports & Dashboards	✅	✅	✅	❌
Publish Content	✅	✅	❌	❌
Delete Content	✅	✅	✅	❌
Share Reports/Dashboards	✅	✅	✅	❌
View Reports	✅	✅	✅	✅

Choosing the Right Role

- **Admin** → Best for IT Admins or Managers who need full control.
- **Member** → Best for Report Developers who manage content but don't control permissions.
- **Contributor** → Best for Analysts who need to create reports but don't manage the workspace.
- **Viewer** → Best for End Users who only need to view reports.

4. What is RLS ?

RLS stands for Row Level Security.

Row-Level Security (RLS) is a feature in **Power BI** that restricts data access for different users based on their roles. It ensures that users **only see the data they are authorized to view**.

Ex: If you are in India, you are not authorized to view china data, if you are in US you are not authorized to view India data. So likewise you are not supposed to see other data which is irrelevant to you.

Types of RLS:

- **Static** – You can go to the manage rules and you can set up some filters and you can add the certain people to the certain region and only those people will be able to access it. Disadvantage is you need to do it manually, like add people manually and also every RLS has to go from desktop to Power BI service. In Power BI service directly you can't do the operation.
- **Dynamic** – Dynamic RLS is a more scalable way to restrict data access based on the logged-in user's email or role**.** Instead of manually assigning users to static roles, dynamic RLS automatically filters data based on user information stored in a table. We can use **USERPRINCIPALNAME()** as DAX function which is responsible for capture the system name or username. Based on that, you will be given access to the reports. In this case once you give this, you can also have an entitlement table what type of access they need in a separate table, you can map it and in Power BI service you can give access.

Benefits of RLS

- **Data Security** – Users only see data relevant to them.
- **Better Performance** – No need for separate datasets for different users.
- **Scalability** – Dynamic RLS scales with growing teams.

5. What do you mean by Gateway ? How many types of gateway you can have in Power BI?

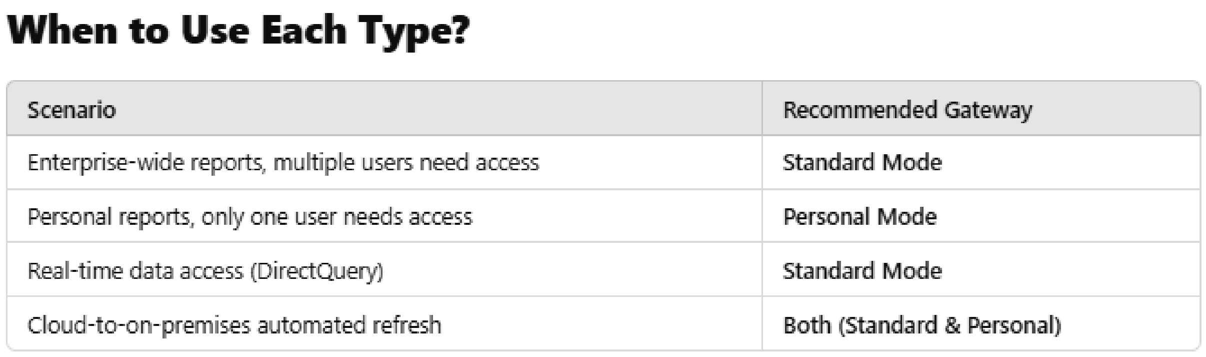
A **Gateway** is a **bridge** that enables **secure data transfer** between **on-premises data sources** (e.g., SQL Server, Excel, SharePoint, ERP systems) and **Power BI Service (cloud)**.

2 types of gateway we can have

- Standard
- Normal (Personal gateway)

Normal is for one person. One can use it for personal purpose and **standard** is preferred in organization where multiple people can work on it.

Personal	Standard
Designed for **single-user access** (only the person who installs it can use it).	Supports multiple users
Supports **scheduled refresh only** (No DirectQuery.	Allows **DirectQuery and Scheduled Refresh**.
Recommended for personal practice level.	Recommended for **enterprise level solutions** where multiple reports need live data.
Works with **Power BI only** (not for PowerApps or other services).	Can be used for **Power BI, PowerApps, Power Automate, Azure Logic Apps**.



When to Use Each Type?

Scenario	Recommended Gateway
Enterprise-wide reports, multiple users need access	Standard Mode
Personal reports, only one user needs access	Personal Mode
Real-time data access (DirectQuery)	Standard Mode
Cloud-to-on-premises automated refresh	Both (Standard & Personal)

Key Benefits of Using a Gateway

- **Secure Data Access** – No need to move on-prem data to the cloud.
- **Automatic Data Refresh** – Keep reports up to date.
- **Supports Multiple Data Sources** – Connect SQL, Oracle, Excel, etc.
- **Enterprise-Ready** – Standard Mode allows multiple users to connect securely.

6. What is dataflow in Power BI service?

We know about power query editor, which is used for transformation, but the question is can we do transformation in Power BI service? The answer is yes, we can do it through dataflow. By using dataflow in Power BI service, we can create dataset, reuse a dataset which already been used in different workspace and this will also help us to transform the data.

In other words we can say is dataflow is power query editor in Power BI service.

7. What is self service BI?

SSBI has enabled many business professionals with no technical or coding background to use Power BI and generate reports and draw predictions successfully.

8. What are the difference in data modeling between Power BI Desktop and Power Pivot for Excel?

Power Pivot for excel supports only **single directional** relationships (one to many), calculated columns and one import mode.

Power BI Desktop supports **bi-directional** cross filtering connections, security, calculated tables and multiple import options.

9. Define bidirectional cross filtering.

Bi-directional cross filtering makes the job for data modelers easier, means when you have to filter data from one table to another table, then you have to use bi-directional cross filtering.

When to Use Bidirectional Cross-Filtering?

- **Many-to-Many Relationships** – When you need both tables to filter each other.
- **Complex Data Models** – When relationships involve multiple intermediate tables.
- **Aggregated Views** – If you want summary tables to be affected by fact tables.

When NOT to Use Bidirectional Cross-Filtering?

- **Large Datasets** – It can **slow down performance** due to additional filtering.
- **Unnecessary Loops** – It can create **circular dependencies**, leading to incorrect results.
- **Simple Hierarchies** – When a single-direction filter is enough, avoid bidirectional filters.

Alternative to Bidirectional Filtering: Using **crossfilter()** function in DAX

10. What are the various types of refresh options provided in Power BI?

1. Scheduled Refresh – It is where you define the frequency and time slots to refresh the dataset. Some data sources do not require a gateway to be configurable for refresh, other sources might require gateway.

Configuration:

- Power BI Service → Dataset → Settings → Scheduled Refresh.
- Can set up to 8 refreshes/day for Pro users and 48 refreshes/day for Premium capacity.

Best For:

- Reports requiring regular updates (e.g., daily sales reports).

Requires:

- Gateway for on-premises data sources.

2. Incremental Refresh – This service dynamically separates data that needs to be refreshed frequently from data that can be refreshed less frequently. In other words it refreshes **only new or updated data**, rather than the entire dataset.

Configuration:

- Enable in Power BI Desktop (under Table Properties).
- Define RangeStart and RangeEnd parameters for the incremental policy.

Best For:

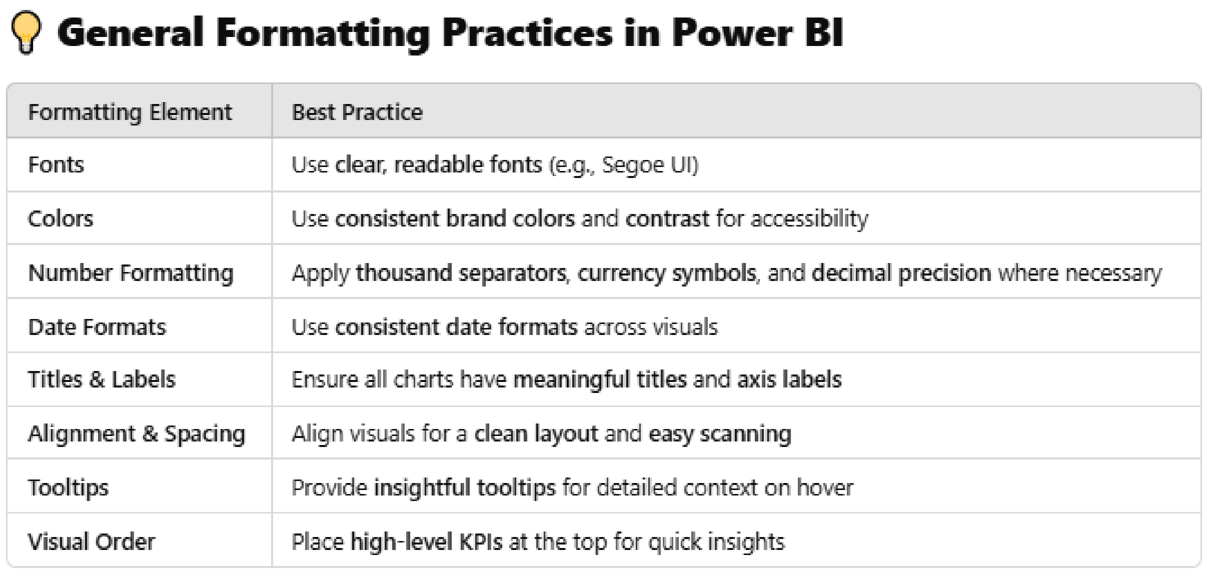
- Large datasets where only recent data changes (e.g., sales transactions).

Requires:

- Power BI Pro with Premium capacity for full support.

11. Why Should We Apply General Formatting in Power BI?

General formatting in Power BI is crucial for making reports and dashboards not only visually appealing but also understandable, consistent, and impactful. Proper formatting enhances the user experience, improves data storytelling, and ensures clear communication of insights.



💡 General Formatting Practices in Power BI

Formatting Element	Best Practice
Fonts	Use **clear, readable fonts** (e.g., Segoe UI)
Colors	Use **consistent brand colors** and **contrast** for accessibility
Number Formatting	Apply **thousand separators, currency symbols**, and **decimal precision** where necessary
Date Formats	Use **consistent date formats** across visuals
Titles & Labels	Ensure all charts have **meaningful titles** and **axis labels**
Alignment & Spacing	Align visuals for a **clean layout** and **easy scanning**
Tooltips	Provide **insightful tooltips** for detailed context on hover
Visual Order	Place **high-level KPIs** at the top for quick insights

12. Why and how would you use a custom visual file?

You will use a custom visual file if the pre-packaged files doesn't fit the needs of your business. Developers create custom visual files, and you can import and use them in the same way as you would do for pre-packaged files.
Interviewer might ask what kind of custom visuals you should use in your report?
Then in order to improve the performance of your report please always use MS certified custom visuals.

Benefits of Using Custom Visual Files

- **Greater flexibility** and **customization**.
- Ability to **address specific business needs**.
- **Improves storytelling** through **advanced visuals**.
- **Reusable** across multiple reports and projects.

Considerations before Using Custom Visuals

- **Performance Impact:** Some custom visuals may slow down report performance.
- **Security Concerns:** Use trusted sources like **AppSource** for third-party visuals.
- **Compatibility:** Ensure visuals are updated and **compatible** with your version of Power BI.

User Adoption: Ensure end-users are trained to **interpret new visuals** properly.

DAX (Data Analytic Expression)

DAX

1. What is DAX?

DAX (Data Analytics Expression) is a collection of functions, operators and constants used in formulas to calculate and return values. In other words it helps you to create new information from data you already have.

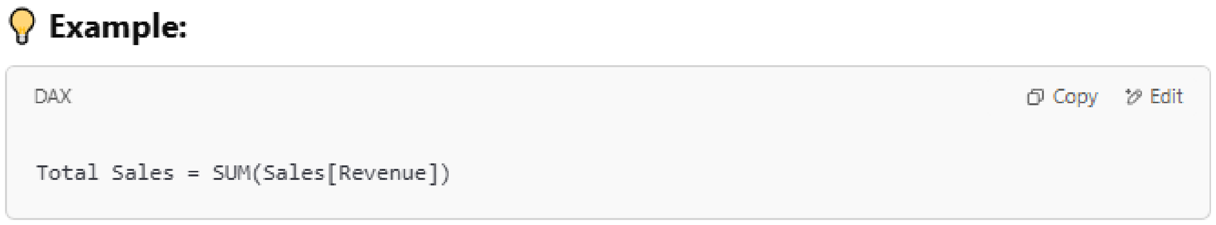
When to Use DAX in Power BI?

- When you need **custom aggregations** (e.g., Total Sales by Region).
- When using **relationships between tables** (e.g., linking Customers and Sales).
- When performing **advanced filtering** (e.g., Sales only for VIP customers).
- When analyzing **time-based data** (e.g., Monthly Growth, Year-over-Year comparisons).

2. What are 3 fundamental concepts of DAX?

Predominantly there are 3 concepts.

(i) Syntax – The syntax of DAX defines how formulas are written. Syntax error will result to an error message



Example:

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

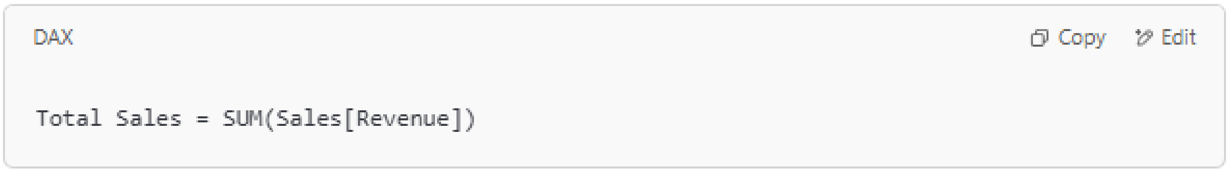
- Total Sales: **New measure** name.
- Sum: **DAX function** that adds values.
- Sales[Revenue]: **Table and column reference**.

(ii) Functions – Functions in DAX are predefined formulas that perform calculations. It refers to instructions performed in a specific sequence to achieve a specific result.

a. **Aggregation Functions**

- Perform **summaries** of data.
- Examples: SUM(), AVERAGE(), MIN(), MAX(), COUNT().

Example:

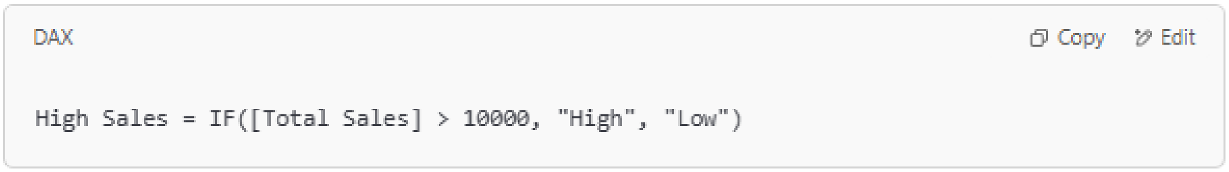


```
Total Sales = SUM(Sales[Revenue])
```

b. Logical Functions

- **Test conditions** and return **TRUE** or **FALSE**.
- Examples: IF(), AND(), OR(), SWITCH().

Example:

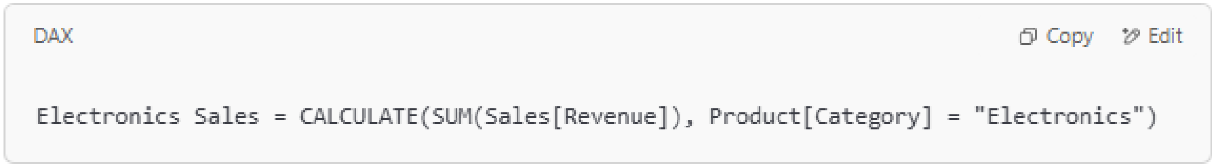


```
High Sales = IF([Total Sales] > 10000, "High", "Low")
```

c. Filter Functions

- **Control the filter context** in calculations.
- Examples: FILTER(), CALCULATE(), ALL(), RELATED().

Example:

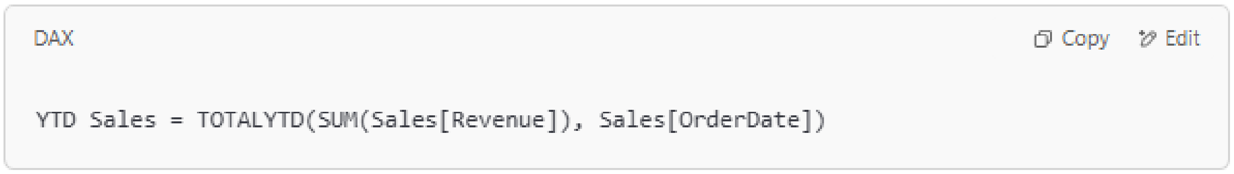


```
Electronics Sales = CALCULATE(SUM(Sales[Revenue]), Product[Category] = "Electronics")
```

d. Time Intelligence Functions

- Handle **date-based calculations**, like **year-to-date**, **month-over-month**, or **same period last year**.
- Examples: TOTALYTD(), SAMEPERIODLASTYEAR(), DATEADD().

Example:



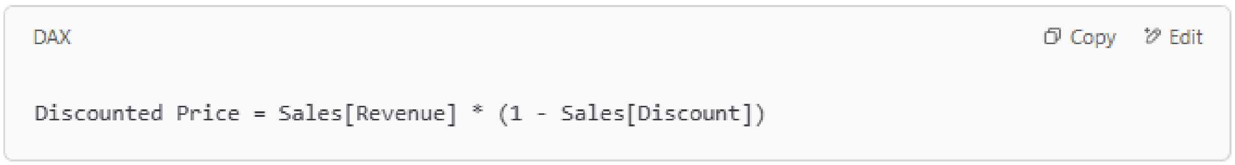
```
YTD Sales = TOTALYTD(SUM(Sales[Revenue]), Sales[OrderDate])
```

(iii) Context – Context determines how DAX calculations are evaluated**.** There are two main types**:**

a) Row Context

- The **current row** in a table where a calculation is performed.
- Usually created by:
  - **Calculated columns**
  - **Iterators** (e.g., SUMX(), FILTER())

Example:



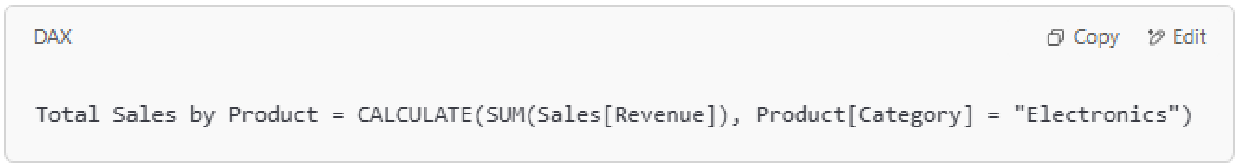
```
Discounted Price = Sales[Revenue] * (1 - Sales[Discount])
```

  - Here, **each row** uses its **Revenue** and **Discount** values for the calculation.

b) Filter Context

- The **subset of data** visible to a calculation based on **filters applied** (e.g., slicers, rows in a visual, or explicit filters in DAX).
- Typically used in **measures**.

Example:



```
Total Sales by Product = CALCULATE(SUM(Sales[Revenue]), Product[Category] = "Electronics")
```

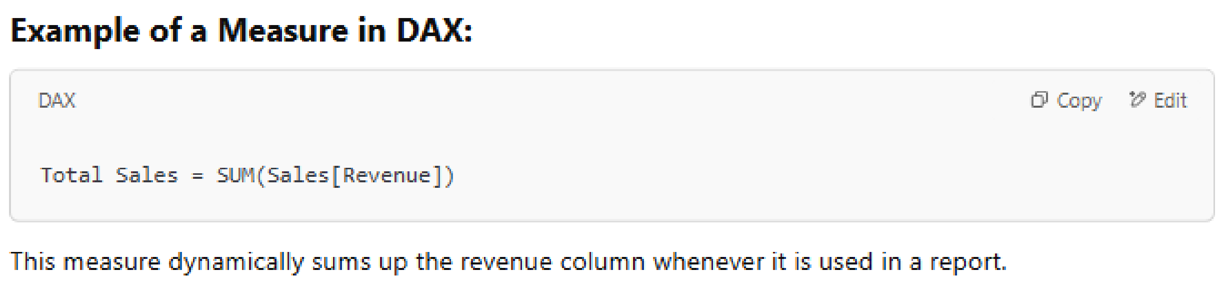
- The **CALCULATE()** function modifies the **filter context** to show sales **only for Electronics**.

3. Explain the difference between measure and calculated column?

In Power BI and Power Pivot, both **measures** and **calculated columns** are used for performing calculations, but they serve different purpose

Measures - Dynamic, Aggregated Calculations

- Uses **DAX (Data Analysis Expressions)** for aggregation (Sum, Average etc)
- More efficient because it calculates only when needed.
- Works well with **filters, slicers, and context changes**.



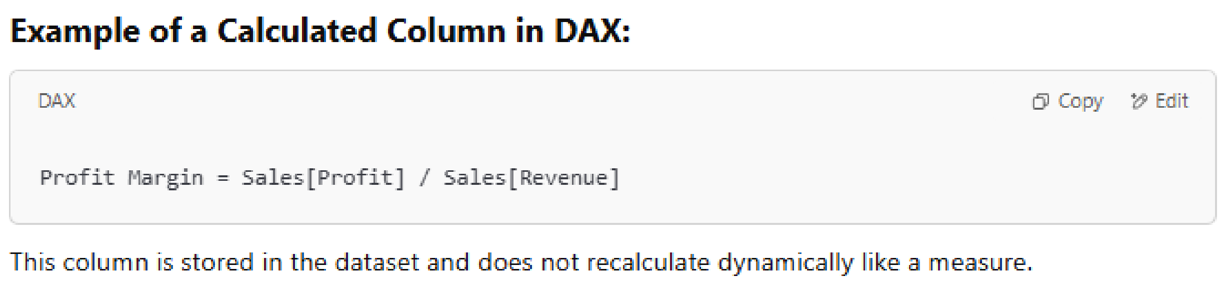
Example of a Measure in DAX:

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

This measure dynamically sums up the revenue column whenever it is used in a report.

Calculated Column - Static, Row-by-Row Calculation

- Uses **DAX** for row-wise calculations.
- Calculated **during data refresh** (values are stored in the model).
- Increases **data model size** since results are stored in memory.
- Best for **row-level calculations** (e.g., concatenations, derived columns).



Example of a Calculated Column in DAX:

```
Profit Margin = Sales[Profit] / Sales[Revenue]
```

This column is stored in the dataset and does not recalculate dynamically like a measure.

	Measure	**Calculated Column**
Calculation Type	Aggregated (dynamic)	Row-by-row (static)
Storage	Not stored, calculated on demand	Stored in the data model
Performance	More efficient	Can slow performance (increases model size)
Context	Changes based on report filters	Fixed at data load time
Example	SUM(Sales[Revenue])	Sales[Profit] / Sales[Revenue]

When to Use Which?

- **Measures** are typically used for aggregation, like sum, average or ratios.
- **Use a Calculated Column** when you need **row-level calculations** that don’t change dynamically (e.g., concatenations, categorization).

4. What is Row Context and Filter Context in DAX?

Row Context: Refers to the current row being evaluated. It is present in calculated columns and iterators like SUMX, FILTER, etc.

Example:
Imagine a table Sales with columns: Product, Quantity, and UnitPrice.

If we create a **calculated column**:

```dax
TotalPrice = Sales[Quantity] * Sales[UnitPrice]
```

- For each row, DAX multiplies Quantity and UnitPrice for that *specific row*.
- The **row context** here is implicit because the calculation is performed on each row independently.

Filter Context: Comes from report filters, slicers, or CALCULATE(). It defines **which data** is visible when a measure is calculated.

Example:
Suppose you have a measure:

```dax
TotalSales = SUM(Sales[TotalPrice])
```

- If you add this measure to a report and use a **slicer** for Product, Power BI applies a **filter context** to show TotalSales *only* for the selected product.
- The filter context changes dynamically based on user interaction.

💡 Row Context vs Filter Context

Aspect	Row Context	Filter Context
Scope	One row at a time	Entire dataset, based on applied filters
Occurs In	Calculated columns, iterator functions (`X` functions)	Measures, visual filters, `CALCULATE()`
Example Function	`SUMX()` , `FILTER()`	`CALCULATE()` , `ALL()` , `FILTER()`
Implicit/Explicit	Implicit in calculated columns, explicit in iterators	Explicit in DAX with `CALCULATE()` , visuals
Navigation	Needs `RELATED()` for related tables	Automatically respects relationships in model

5. Name some DAX functions you have used in your project.

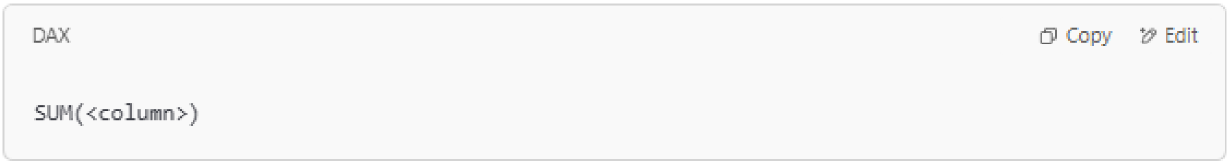
- Filter function - `FILTER()`, `CALCULATE()`, `ALL()`, `RELATED()`.
- Time intelligence function - `TOTALYTD()`, `SAMEPERIODLASTYEAR()`, `DATEADD()`.
- Date function – DATE()
- Logical function - `IF()`, `AND()`, `OR()`, `SWITCH()`.
- Mathematical function – ABS(), POWER(),MOD(), SUM(), AVG(), MIN()/MAX()

6. What are the differences between SUM() and SUMX() ?

`SUM()` Function - The `SUM()` function adds up all the values in a single column.

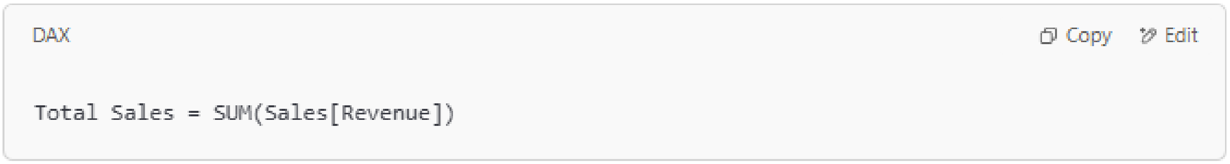
What it does: Calculates the total revenue by summing all the values in the `Revenue` column of the Sales table.

⚡ Syntax:



```
SUM(<column>)
```

💡 Example:

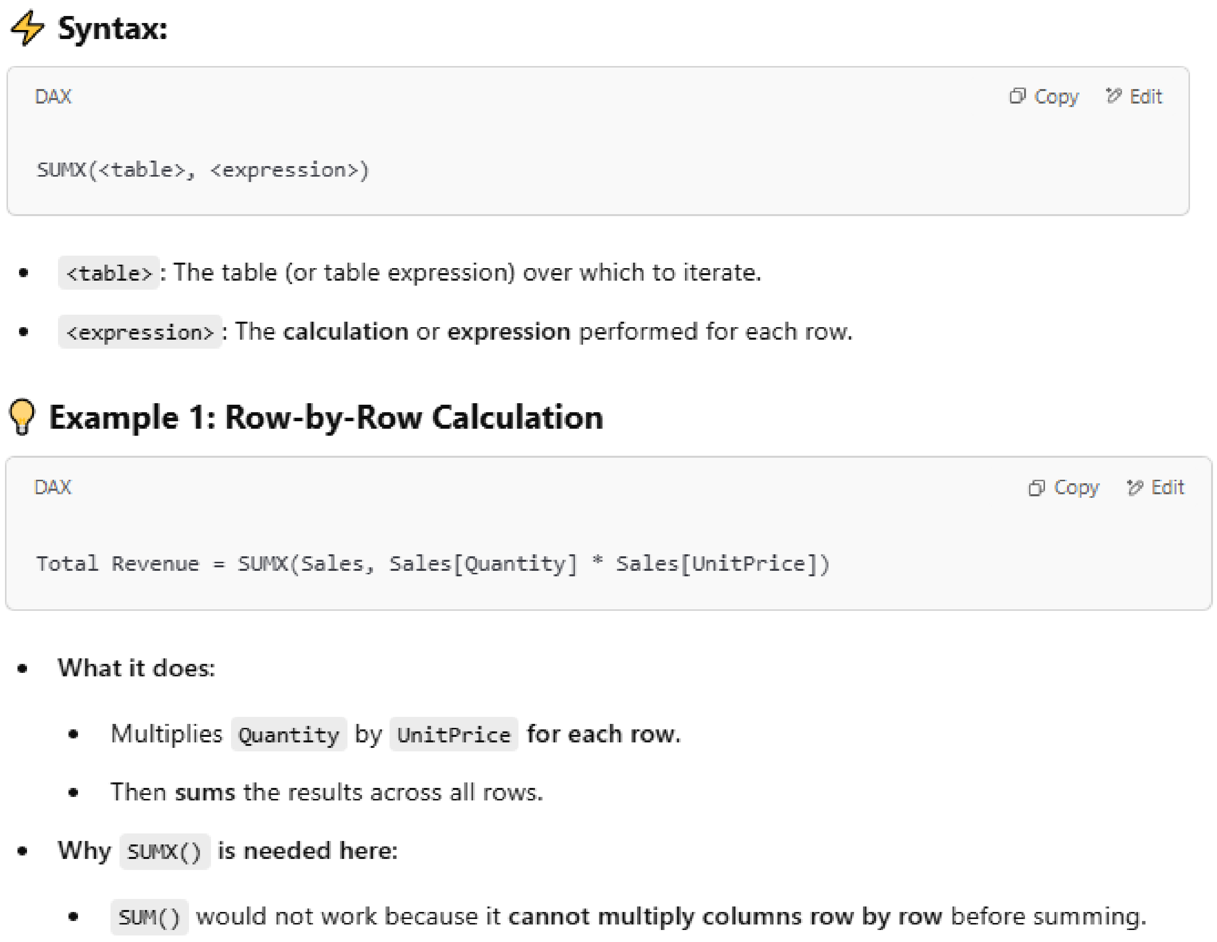


```
Total Sales = SUM(Sales[Revenue])
```

Key Points:

- Only works with numeric columns.
- Does not perform row-by-row calculations.
- Best for simple summations when no row context manipulation is needed.
- Faster and more efficient for straightforward aggregations.

SUMX() Function - The SUMX() function iterates over a table, evaluates an expression for each row, and then sums the results.



⚡ Syntax:

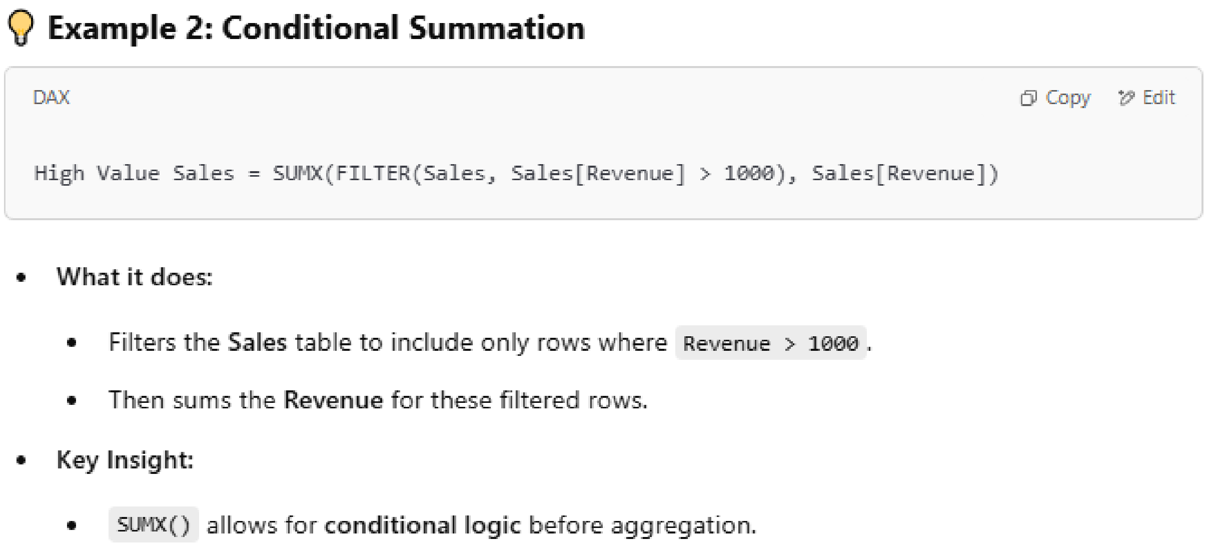
```DAX
SUMX(<table>, <expression>)
```

- `<table>`: The table (or table expression) over which to iterate.
- `<expression>`: The **calculation** or **expression** performed for each row.

💡 Example 1: Row-by-Row Calculation

```DAX
Total Revenue = SUMX(Sales, Sales[Quantity] * Sales[UnitPrice])
```

- **What it does:**
  - Multiplies `Quantity` by `UnitPrice` **for each row.**
  - Then **sums** the results across all rows.
- **Why `SUMX()` is needed here:**
  - `SUM()` would not work because it **cannot multiply columns row by row** before summing.



💡 Example 2: Conditional Summation

```DAX
High Value Sales = SUMX(FILTER(Sales, Sales[Revenue] > 1000), Sales[Revenue])
```

- **What it does:**
  - Filters the **Sales** table to include only rows where `Revenue > 1000`.
  - Then sums the **Revenue** for these filtered rows.
- **Key Insight:**
  - `SUMX()` allows for **conditional logic** before aggregation.

Key Points:

- Performs row-by-row evaluation (iterative).
- Supports complex expressions, including multiplication, division, and custom logic per row.
- Can handle filters and dynamic calculations that SUM() cannot.
- Slightly slower than SUM() due to the row iteration, so use only when necessary.

Use SUM() when:

- You need a simple total from a single numeric column.
- No row-by-row logic or calculations are required.

Use SUMX() when:

- You need to calculate values per row before summing.
- Your calculation involves multiple columns or custom expressions.
- Conditional calculations or filters need to be applied first.

7. What is SUMMERIZE() function?

It is a table level function which will give us new table as per requirement. It is used to create a summery table based on specific columns or aggregations from an existing table. It is used to group data by one or more columns and calculate aggregations, similar to a SQL group by statement.

Syntax:

SUMMARIZE(<table>, <groupBy_Column Name>,<groupBy_Column Name>,……

[name_of_new_column1], expression1, [name_of_new_column2], expression2, ...)

Parameters:

- **table**: The table to summarize.
- **groupby_Column Name(s):** The columns by which the data will be grouped.
- **name_of_new_column1:** The alias or name for the aggregated column.
- **expression**: The DAX expression to compute the value for the new column (aggregation).

Example 1: Basic Grouping with Aggregation

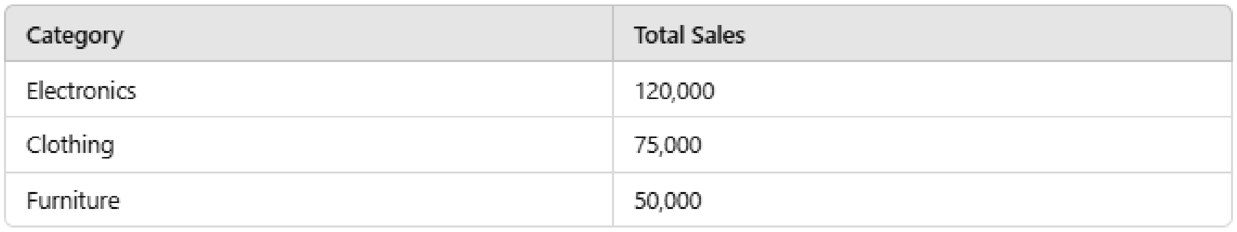
Scenario: Summarize total sales by product category.

Sales_Summary = SUMMARIZE(Sales, Product[Category], "Total Sales", SUM(Sales[Revenue]))

Explanation:

- **Sales**: The source table.
- **Product[Category]**: The grouping column (**Category**).
- **"Total Sales"**: The **name** for the new column.
- **SUM(Sales[Revenue])**: The **aggregation expression** (total revenue per category).

Resulting Table:



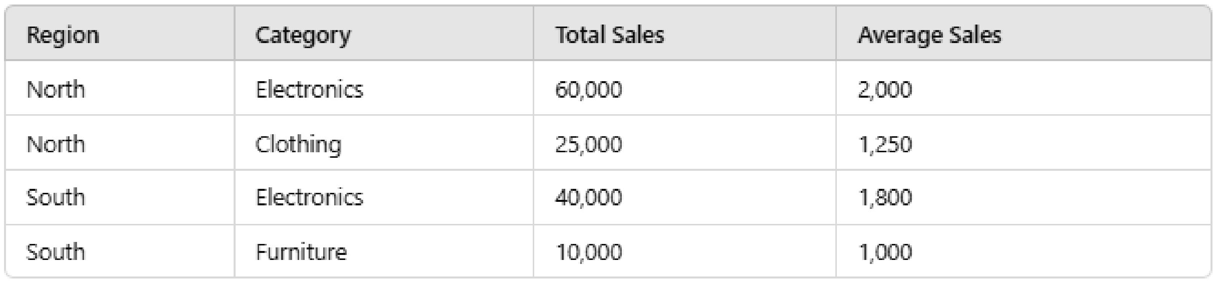
Category	Total Sales
Electronics	120,000
Clothing	75,000
Furniture	50,000

Example 2: Multiple Groupings and Calculations

Scenario: Summarize total and average sales by region and category.

Regional_Sales_Summary = SUMMARIZE(Sales, Region[Name], Product[Category], "Total Sales", SUM(Sales[Revenue]), "Average Sales", AVERAGE(Sales[Revenue]))

Resulting Table:



Region	Category	Total Sales	Average Sales
North	Electronics	60,000	2,000
North	Clothing	25,000	1,250
South	Electronics	40,000	1,800
South	Furniture	10,000	1,000

8. What is the need of date master table?

For most of the time intelligence functions, your calculation has to be taken with respect to today's date, Ex: with respect to today you can take last one month, last quarter date, so all these are something which we keep as a reference by creating the date master. You are telling the system that this is what is the correct date and based on this please calculate. The purpose of date master table is if you have a lot of time intelligence functions please use this date master as table.

9. What is SAMEPERIODLASTYEAR() ?

Always you can not compare the current month with last month.

Example: You can not compare January (2025) to December (2024), due to Christmas, New Year, vacation the sales might be up. In January the scenario might be different and sale might be down.

In this case what we can do?

We can compare current year January (2025) to last year January (2024) or current year December (2024) to last year December (2023).

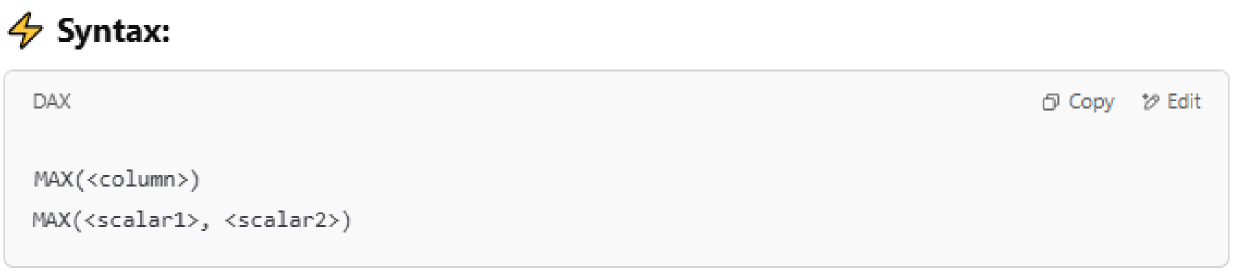
10. What is ALL and ALLEXCEPT function?

The `ALL()` function removes all filters from a specified table or column(s). It’s frequently used in calculations where you need to ignore slicers, visual filters**,** or row context to get a global total or reset context**.**

The `ALLEXCEPT()` function removes all filters except for the specified columns**.** It’s perfect for scenarios where you want to keep certain grouping filters while removing others**.**

11. What are the differences between MAX(), MAXA() and MAXX() DAX functions?

MAX() Function – The MAX() function returns the largest value in a column or between two scalar expressions. It works only with numeric, date/time, and text data types but ignores logical values (TRUE, FALSE) and treats blanks as empty.



⚡ **Syntax:**

```
MAX(<column>)
MAX(<scalar1>, <scalar2>)
```

- **<column>:** The column from which the maximum value is returned.
- **<scalar1>, <scalar2>:** Two values to compare, returning the larger one.

🎯 Example: Basic Usage

Data (Sales Table):

Product	Revenue	Discount	InStock
Laptop	5000	10	TRUE
Smartphone	4000	15	FALSE
Tablet	3000	12	TRUE
Headphones	1500	5	FALSE

DAX Calculation:

DAX Copy Edit

```dax
Max Revenue = MAX(Sales[Revenue])
```

🏃 Result:

java Copy Edit

```java
Max Revenue = 5000
```

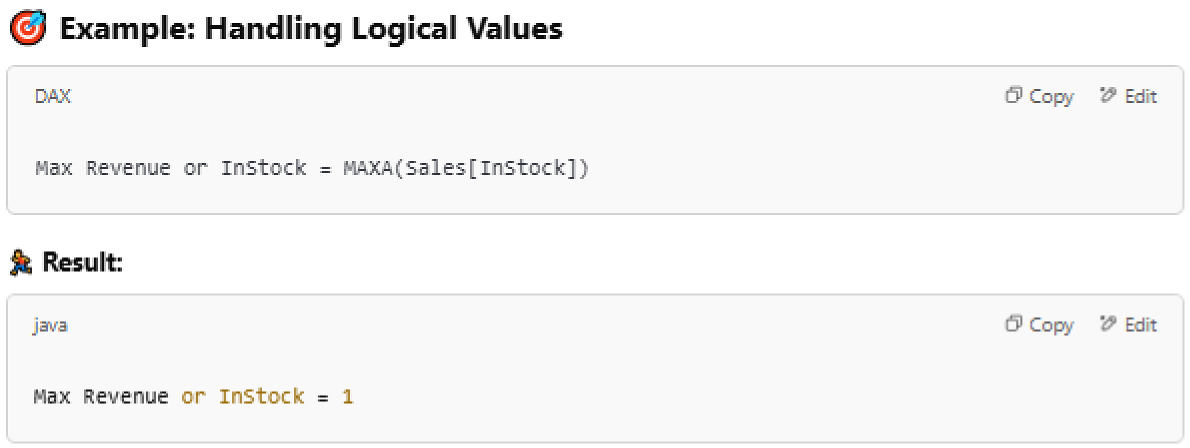
MAXA() Function – The MAXA() function also returns the largest value but includes logical values (TRUE, FALSE) and treats them as numbers:

- `TRUE = 1`
- `FALSE = 0`
- Blanks are treated as 0.
- Works similarly to MAX(), but logical values affect the result.

⚡ Syntax:

DAX Copy Edit

```dax
MAXA(<column>)
```



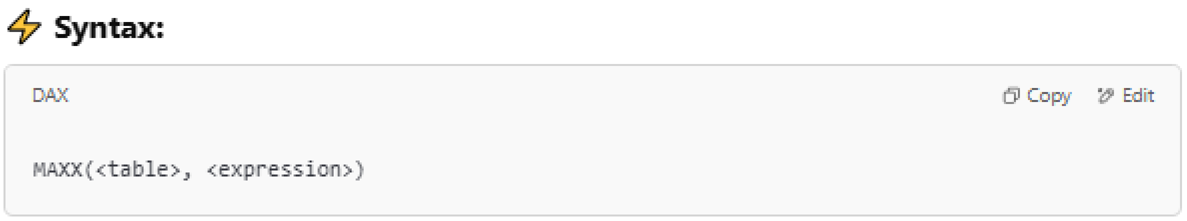
🎯 Example: Handling Logical Values

```DAX
Max Revenue or InStock = MAXA(Sales[InStock])
```

Result:

```java
Max Revenue or InStock = 1
```

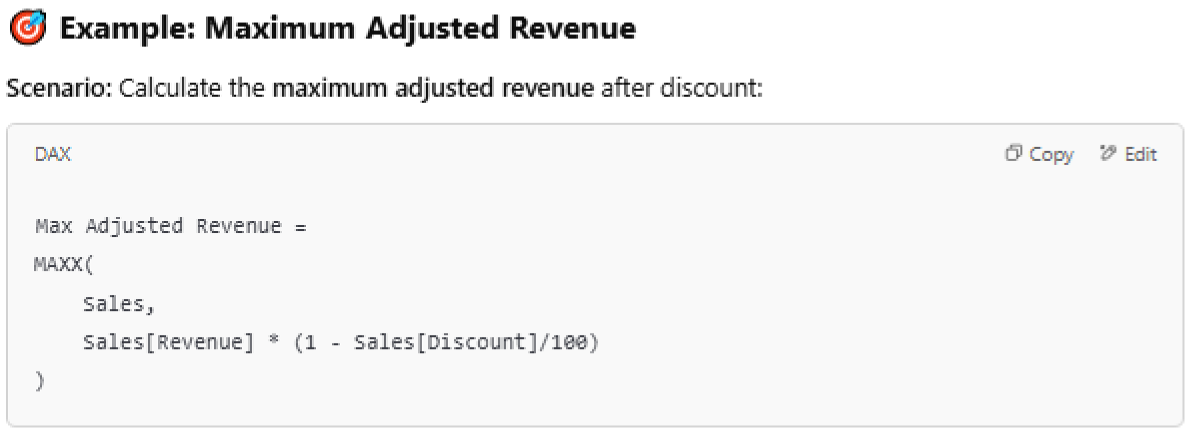
MAXX() Function – The MAXX() function evaluates an expression for each row of a table and returns the largest value.



⚡ Syntax:

```DAX
MAXX(<table>, <expression>)
```

- **<table>:** The table to iterate over.
- **<expression>:** The expression evaluated for each row.



🎯 Example: Maximum Adjusted Revenue

Scenario: Calculate the **maximum adjusted revenue** after discount:

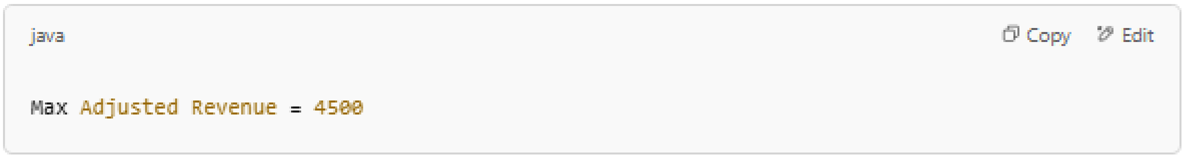
```DAX
Max Adjusted Revenue =
MAXX(
    Sales,
    Sales[Revenue] * (1 - Sales[Discount]/100)
)
```

Calculation Breakdown:

- **Laptop:** 5000 * (1 - 0.10) = 4500
- **Smartphone:** 4000 * (1 - 0.15) = 3400

- **Tablet:** 3000 * (1 - 0.12) = 2640
- **Headphones:** 1500 * (1 - 0.05) = 1425

Result:



```java
Max Adjusted Revenue = 4500
```

12. What are the differences between distinct and values in DAX?

The **DISTINCT()** function returns a one-column table that contains the **unique** values from that specified column. In other words, duplicate values are removed and only unique values are returned.

Syntax - DISTINCT(<column>)

- **<column>:** The column from which you want to return distinct values.

The **VALUES()** function returns unique values visible in the current context. It also returns a **blank row** if it exists in the column, unlike `DISTINCT()`.

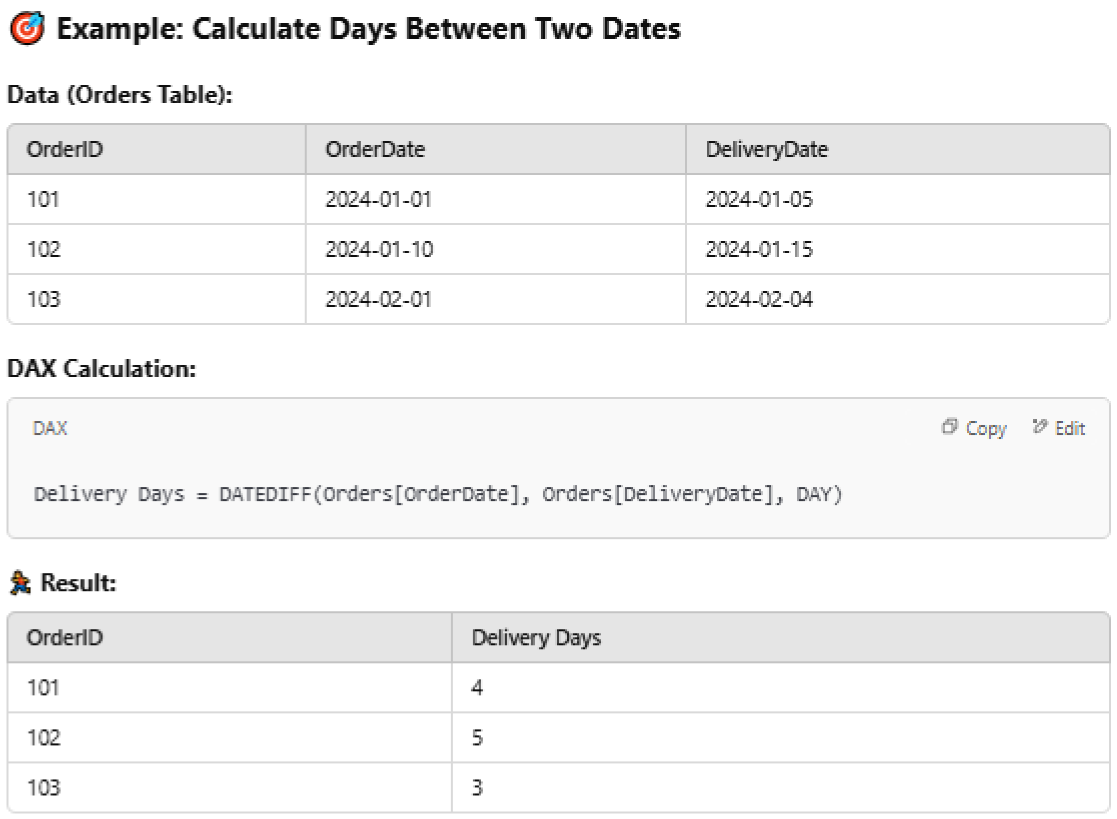
Syntax - VALUES(<table/column>)

- **<column>:** The column for which unique values are returned, respecting filter context.
- **<table>:** Returns a table with the unique rows (all columns) of the specified table.

13. What is the difference between DATEDIFF() and DATESINPERIOD() function?

DATEDIFF() function calculates the difference between 2 dates and returns the result in the specified time unit (day, month, year)
Syntax – DATEDIFF(<start_date>, <end_date>, <interval>)

- Interval – the unit of time (**SECOND**, **MINUTE**, **HOUR**, **DAY**, **WEEK**, **MONTH**, **QUARTER**, **YEAR**).



🎯 Example: Calculate Days Between Two Dates

Data (Orders Table):

OrderID	OrderDate	DeliveryDate
101	2024-01-01	2024-01-05
102	2024-01-10	2024-01-15
103	2024-02-01	2024-02-04

DAX Calculation:

```DAX
Delivery Days = DATEDIFF(Orders[OrderDate], Orders[DeliveryDate], DAY)
```

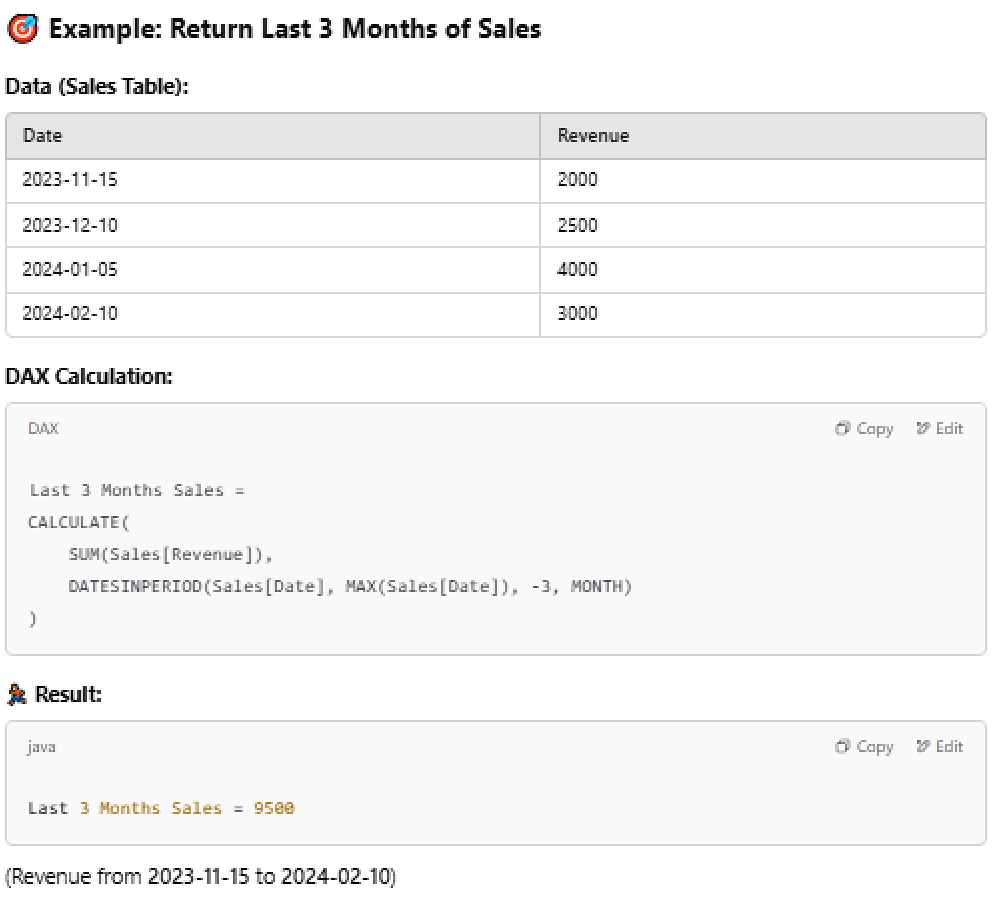
🏃 Result:

OrderID	Delivery Days
101	4
102	5
103	3

The **`DATESINPERIOD()`** function **returns a table** containing a **date range** starting from a given date and extending forward or backward based on the **interval** and **unit**.

Syntax – DATESINPERIOD(<dates>, <start_date>, <number_of_intervals>, <interval>)

- **`<dates>`**: A **date column** (e.g., from a calendar table).
- **`<start_date>`**: The **anchor date** to start the period.
- **`<number_of_intervals>`**: Positive for future periods, negative for past periods.
- **<interval>**: The **time unit** (**DAY**, **MONTH**, **QUARTER**, **YEAR**).



🎯 Example: Return Last 3 Months of Sales

Data (Sales Table):

Date	Revenue
2023-11-15	2000
2023-12-10	2500
2024-01-05	4000
2024-02-10	3000

DAX Calculation:

```DAX
Last 3 Months Sales =
CALCULATE(
    SUM(Sales[Revenue]),
    DATESINPERIOD(Sales[Date], MAX(Sales[Date]), -3, MONTH)
)
```

Result:

```java
Last 3 Months Sales = 9500
```

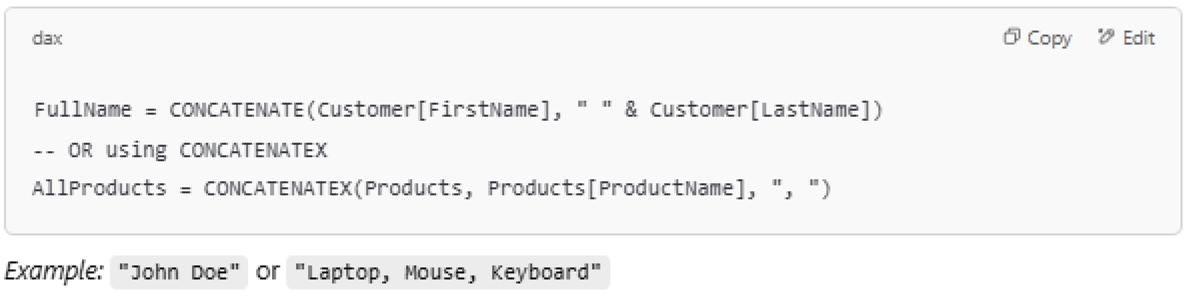
(Revenue from 2023-11-15 to 2024-02-10)

"+" stands for look forward

"-" stands for look backward

14. What are the TEXT() functions in DAX?

- **CONCATENATE()**joins two text strings into one. **CONCATENATEX()**joins text values from a table or column with a specified delimiter.



```dax
FullName = CONCATENATE(Customer[FirstName], " " & Customer[LastName])
-- OR using CONCATENATEX
AllProducts = CONCATENATEX(Products, Products[ProductName], ", ")
```

Example: `"John Doe"` or `"Laptop, Mouse, Keyboard"`

- **`FORMAT()`** – Converts numbers, dates, or values to a text string in a specified format.

```DAX
FORMAT(TODAY(), "MMMM YYYY")   -- Returns "February 2025"
```

- **`LEFT()`, `RIGHT()`** – Extracts a specific number of characters from the start (`LEFT`) or end (`RIGHT`) of a text string.

```DAX
LEFT("PowerBI", 5)    -- Returns "Power"
RIGHT("PowerBI", 2)   -- Returns "BI"
```

- **`MID()`** – Extracts a substring from a text string, given a start position and length.

```DAX
MID("PowerBI", 3, 3)   -- Returns "wer"
```

- **`LEN()`** – Returns the length of a text string (number of characters).

```DAX
LEN("PowerBI")   -- Returns 7
```

- **`REPLACE()` & `SUBSTITUTE()`** – **REPLACE()** replaces part of a string based on position. **SUBSTITUTE()** replaces occurrences of a substring.

```dax
UpdatedCode = REPLACE(Product[ProductCode], 2, 3, "XYZ")
CorrectedText = SUBSTITUTE(Sales[Comment], "old", "new")
```

Example: `AB12345` → `AXYZ45`

- **`SEARCH()` / `FIND()`** – Finds the position of a substring. (`SEARCH` is case-insensitive; `FIND` is case-sensitive.)

```DAX
SEARCH("BI", "PowerBI")   -- Returns 6
```

- **`TRIM()`** – Removes all extra spaces, leaving only single spaces between words.

```
DAX

TRIM("  Power   BI ")   -- Returns "Power BI"
```

- **UPPER()**, **LOWER()**, **PROPER()** – Changes the case of text.

```
DAX

UPPER("power bi")   -- Returns "POWER BI"
LOWER("Power BI")   -- Returns "power bi"
PROPER("power bi")  -- Returns "Power Bi"
```

- **REPT()** – Repeats a text string a specified number of times.

```
DAX

REPT("*", 5)   -- Returns "*****"
```

15. What is SWICTH() and IF() function?

IF() function evaluates a condition and returns one value if the condition is true and another value if the condition is false.

Syntax – IF(<condition>, <result_if_true>, [<result_if_false>])

🎯 Example:

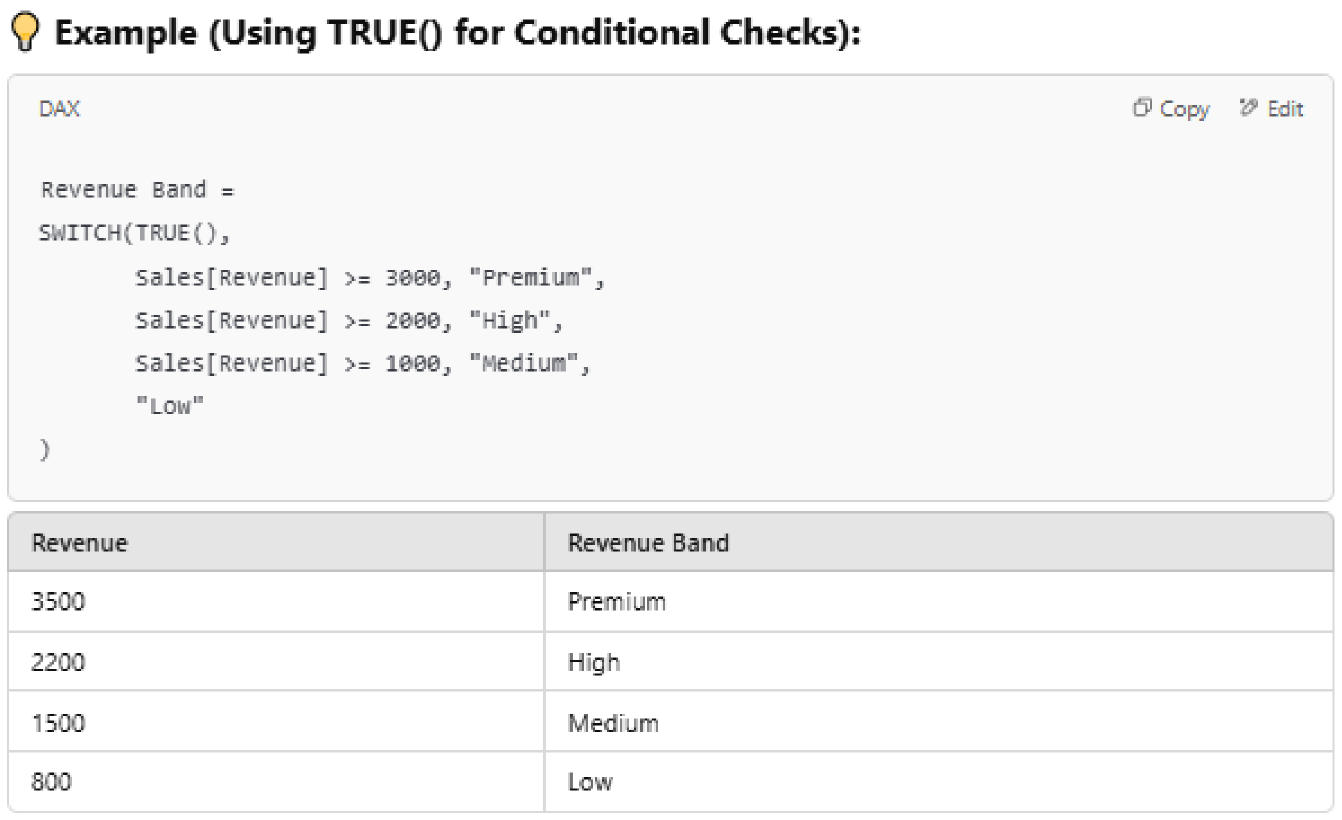
```
DAX

Sales Category = IF(Sales[Revenue] > 1000, "High Revenue", "Low Revenue")
```

🏃 Result:

Revenue	Sales Category
1500	High Revenue
800	Low Revenue

SWITCH() function evaluates an expression against multiple possible values and returns the corresponding result. It's a cleaner alternative to multiple nested `IF()` statements.

Syntax – SWITCH(<expression>, <value1>, <result1>, <value2>, <result2>, ..., [<else_result>])



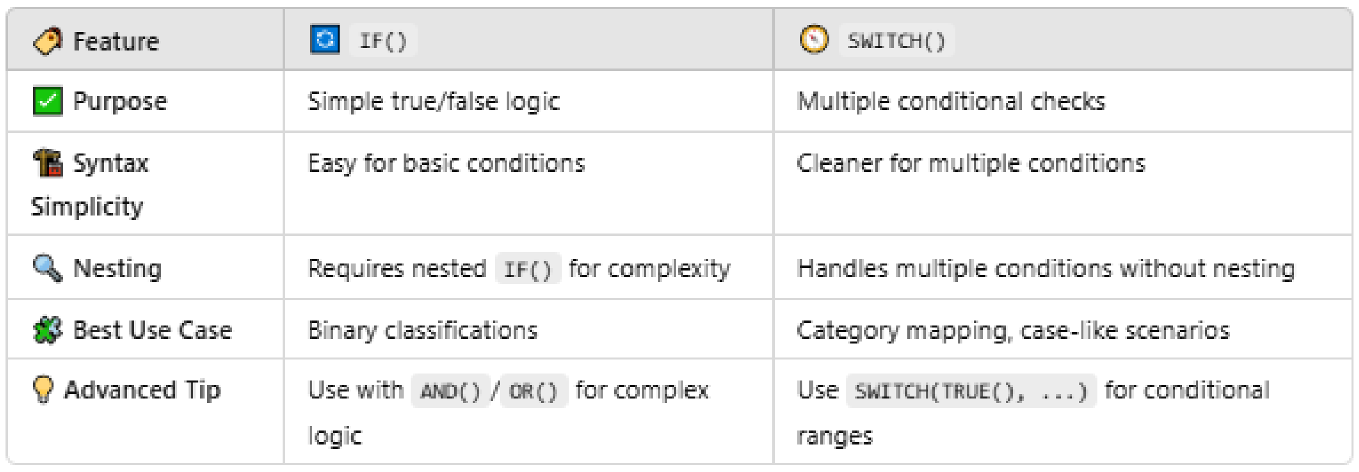
💡 Example (Using TRUE() for Conditional Checks):

DAX Copy Edit

```
Revenue Band =
SWITCH(TRUE(),
       Sales[Revenue] >= 3000, "Premium",
       Sales[Revenue] >= 2000, "High",
       Sales[Revenue] >= 1000, "Medium",
       "Low"
)
```

Revenue	Revenue Band
3500	Premium
2200	High
1500	Medium
800	Low

Key difference between IF() and SWITCH() function.



🏷️ Feature	🔄 `IF()`	🧭 `SWITCH()`
✅ Purpose	Simple true/false logic	Multiple conditional checks
🏗️ Syntax Simplicity	Easy for basic conditions	Cleaner for multiple conditions
🔍 Nesting	Requires nested `IF()` for complexity	Handles multiple conditions without nesting
🧩 Best Use Case	Binary classifications	Category mapping, case-like scenarios
💡 Advanced Tip	Use with `AND()` / `OR()` for complex logic	Use `SWITCH(TRUE(), ...)` for conditional ranges

16. What is the difference between CALCULATE() and CALCULATETABLE() function?

CALCULATE() modifies the context of a calculation and returns a single value.

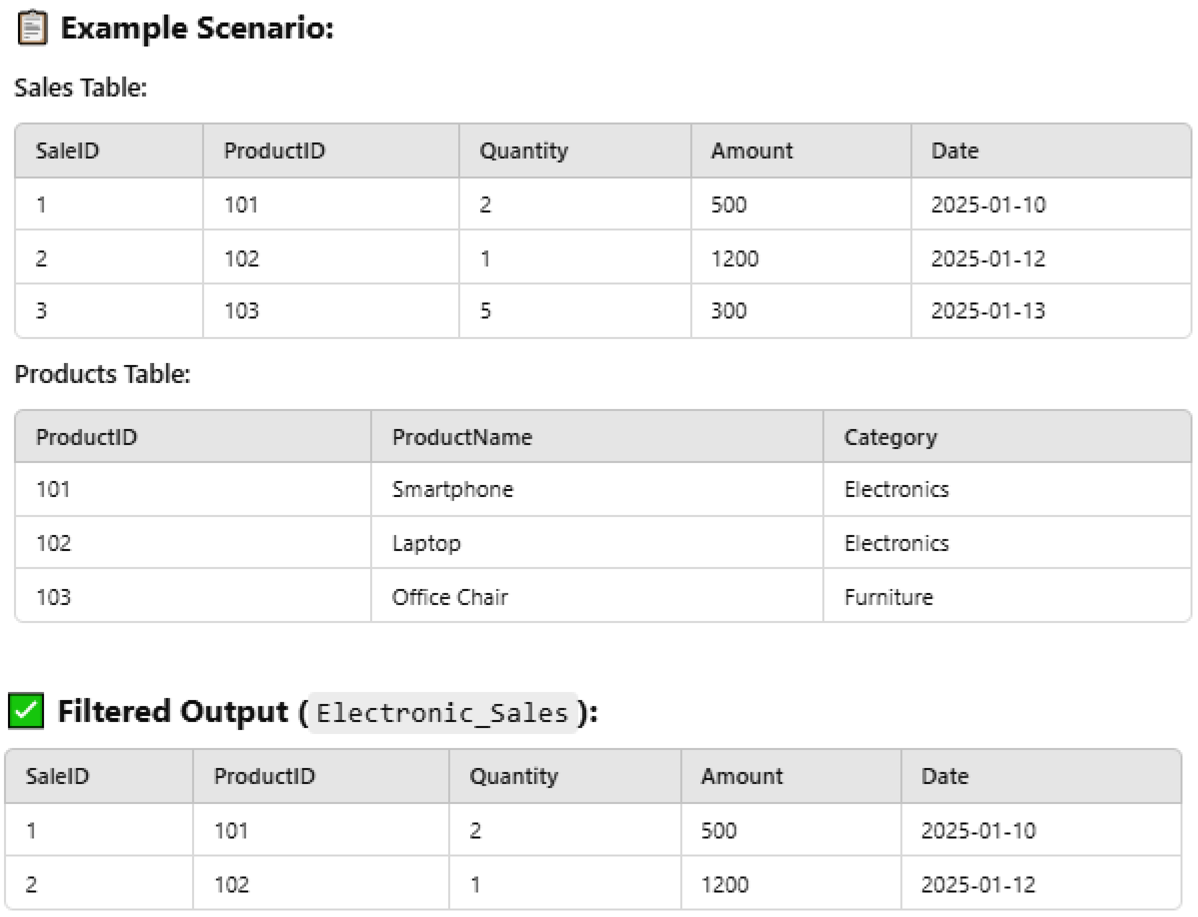
Syntax – CALCULATE(<expression>, <filter1>, <filter2>, ...)

Example: Total_Electronics_Sales = CALCULATE(SUM(Sales[Amount]),Products[Category]=”Electronics”)

Result – A single scaler value representing the total sales of “Electronics”

CALCULATETABLE() modifies the context of a calculation and returns a table.

Electronic_Sales = CALCULATETABLE(Sales,Products[Category]=”Electronics”)



📋 Example Scenario:

Sales Table:

SaleID	ProductID	Quantity	Amount	Date
1	101	2	500	2025-01-10
2	102	1	1200	2025-01-12
3	103	5	300	2025-01-13

Products Table:

ProductID	ProductName	Category
101	Smartphone	Electronics
102	Laptop	Electronics
103	Office Chair	Furniture

✅ Filtered Output (`Electronic_Sales`):

SaleID	ProductID	Quantity	Amount	Date
1	101	2	500	2025-01-10
2	102	1	1200	2025-01-12