Power BI Interview Questions

1. **What is Power BI ? And Why Power BI ? And all Power BI features ?**

Power BI is a **business intelligence tool** developed by Microsoft that enables organizations to visualize and analyze data, making it easier to make informed decisions. It allows users to connect to various data sources, transform and model data, create interactive visualizations, and share insights through dashboards and reports. Power BI is a powerful, user-friendly, and scalable business intelligence tool that transforms raw data into actionable insights through interactive dashboards and reports.

Key Features of Power BI:

- **Data Connectivity:** Connects to multiple data sources like Excel, SQL Server, Azure, Google Analytics, and APIs.
- **Data Transformation:** Provides tools like Power Query to clean, shape, and transform data.
- **Data Modeling:** Allows creating relationships between tables and using calculated measures with DAX (Data Analysis Expressions).
- **Interactive Visualizations:** Offers a variety of charts, graphs, and custom visuals.
- **AI Features:** Includes AI-powered tools for automated insights and natural language queries.

It comes as a package of three major components:

- **Power BI Service (Cloud Service) -** A cloud-based platform to share, collaborate, and manage reports and dashboards. Sharing a sales performance dashboard with team members in real-time.
  - Publishing reports created in Power BI Desktop.
  - Creating and managing dashboards.
  - Real-time data updates and notifications.
  - Role-based access control for secure sharing
- **Power BI Desktop -** A Windows-based application used to design and create data models, reports, and dashboards. Creating a sales analysis report using multiple data sources.
  - Publishing reports created in Power BI Desktop.
  - Creating and managing dashboards.
  - Real-time data updates and notifications.
  - Role-based access control for secure sharing.
- **Power BI Mobile -** Mobile apps for iOS, Android, and Windows devices to access Power BI dashboards and reports. A regional sales manager checks the latest sales metrics while traveling.
  - View interactive dashboards on the go.
  - Receive real-time notifications for updates.
  - Share insights directly from mobile devices.

Why Use Power BI:

- **Easy to Use: Drag-and-Drop Interface:** No extensive coding knowledge is required; users can create reports using simple drag-and-drop features. **User-Friendly Visualizations:** Built-in templates and visual libraries make it easy to create professional dashboards. **Quick Learning Curve:** Even non-technical users can quickly adapt to its features.
- **Data Connectivity: Diverse Data Sources:** Connects to over 100 data sources, including Excel, SQL Server, Azure, Salesforce, Google Analytics, and APIs. **Real-Time Data:** Supports real-time data integration for live dashboards. **Cloud and On-Premises Support:** Works seamlessly with both cloud-based and on-premises systems.
- **Advanced Analytics with DAX: Allows**: **Powerful Calculations:** The DAX (Data Analysis Expressions) formula language allows users to create custom measures and calculations. **Data Modeling:** Build relationships between tables and perform advanced analytics. **Dynamic Reporting:** Create reports that update dynamically based on user inputs

- **More -** Interactive Visualizations, Scalability and Collaboration, Cost-Effectiveness, AI-Powered Insights, Integration with Microsoft Ecosystem.

Data Sources In Power BI:

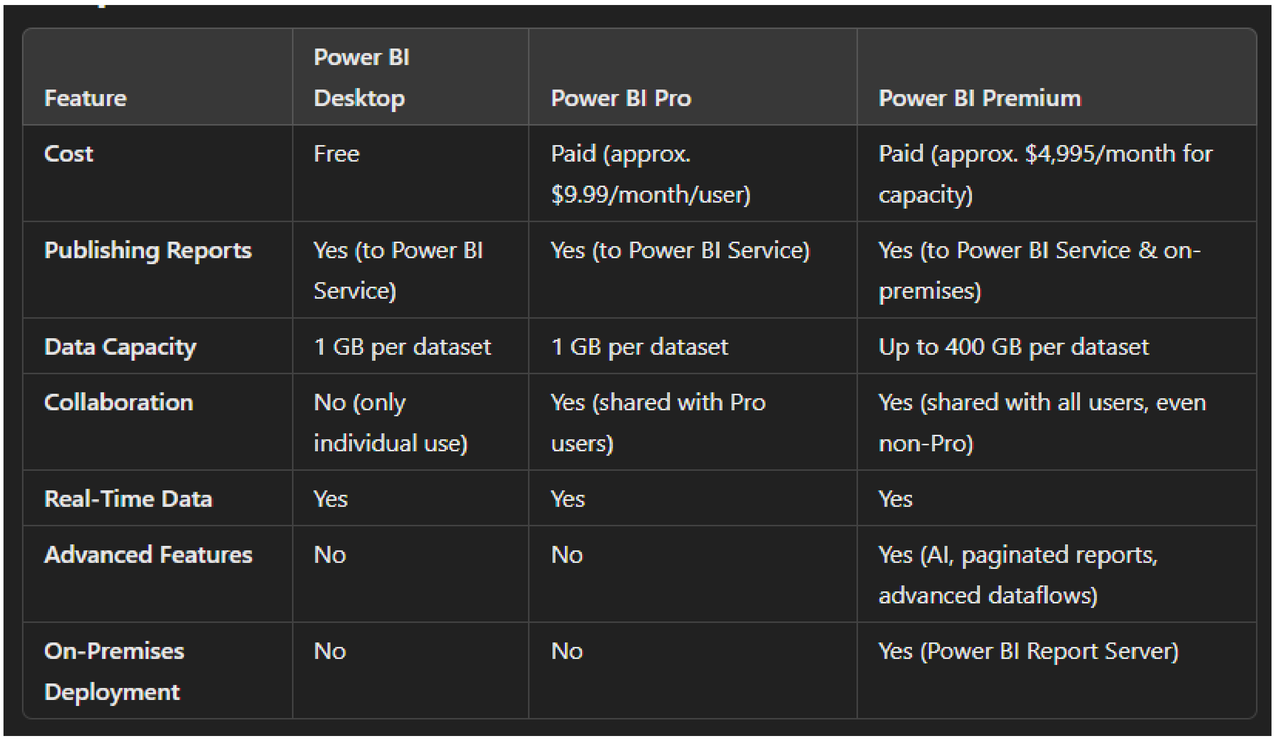
- **File-Based Data Sources:** CSV, Json, Text, XML, etc.
- **Database Sources:** SQL Server, MySQL, PostgreSQL, Oracle Database, Teradata, SAP HANA, Amazon Redshift, etc.
- **Cloud-Based Data Sources:** Azure Services, Google Services, Amazon Services, Snowflake, etc.
- **Online Services:** Microsoft Services, Other Platforms, etc.
- **More:** Business Applications, Web Sources, Power Platform Services, Live and Streaming Data Sources, etc.

Available View in Power BI:

- **Report View:** Used to create, design, and format reports and dashboards.
- **Data View:** Focuses on viewing and exploring the underlying data loaded into the Power BI model.
- **Model View:** Used to manage relationships between tables and design the data model.

Power BI Versions:

- **Power BI Desktop:** Power BI Desktop is a free, self-service application designed for individuals to create reports and dashboards. It's primarily used by data analysts, developers, and business intelligence professionals to connect, analyze, and visualize data.
- **Power BI Pro:** Power BI Pro is a paid version of the Power BI Service that enables collaboration, sharing, and enhanced functionality for small to medium-sized businesses. It is designed for users who need to share and collaborate on reports and dashboards in a team environment. Power BI Pro is a **subscription-based** service.
- **Power BI Premium:** Power BI Premium is a higher-tier, enterprise-level offering that provides additional performance, scalability, and advanced features. It's geared towards larger organizations or businesses that need enhanced capabilities, particularly in terms of data capacity and sharing across large teams. Power BI Premium is licensed on a **per-capacity basis** and can be very expensive (starting from around $4,995 per month per dedicated cloud capacity).



Feature	Power BI Desktop	Power BI Pro	Power BI Premium
Cost	Free	Paid (approx. $9.99/month/user)	Paid (approx. $4,995/month for capacity)
Publishing Reports	Yes (to Power BI Service)	Yes (to Power BI Service)	Yes (to Power BI Service & on-premises)
Data Capacity	1 GB per dataset	1 GB per dataset	Up to 400 GB per dataset
Collaboration	No (only individual use)	Yes (shared with Pro users)	Yes (shared with all users, even non-Pro)
Real-Time Data	Yes	Yes	Yes
Advanced Features	No	No	Yes (AI, paginated reports, advanced dataflows)
On-Premises Deployment	No	No	Yes (Power BI Report Server)

Power Query in Power BI: Power Query is a data transformation and data preparation tool available in Power BI, Excel, and other Microsoft products. It allows users to connect, import, transform, and clean data from a variety of sources before loading it into a data model or a report. Power Query uses a language called M (Power Query M formula language) behind the scenes to apply transformations, but users interact with it through a user-friendly interface.

- **Common Transformations in Power Query:** Filtering, Sorting, Merging/Joining, Pivoting/Unpivoting, Changing Data Types, Grouping, Removing Duplicates
- **Power Query Editor UI Components:** Query Pane, Applied Steps Pane, Data Preview Pane, Ribbon

Filters in Power BI: Filters in Power BI are tools that allow you to control which data is displayed in your reports and visualizations. By applying filters, you can focus on specific subsets of data based on your analysis needs. Filters help refine data at various levels, such as report, page, or visualization, and are a key feature for creating interactive, insightful dashboards.

- **Types of Filters in Power BI:**
    - **Report-Level Filters:** Applied to the entire report, affecting all pages and visualizations within the report. Useful for global filtering, such as showing data for a specific time period or region.
    - **Page-Level Filters:** Applied to a specific page in the report, affecting all visualizations on that page. Useful when different pages require different data contexts.
    - **Visual-Level Filters:** Applied to individual visualizations, affecting only the specific chart or table. Useful when you want to highlight a specific subset of data in a visualization.
    - **Drill-Through Filters:** Enable users to "drill through" from one report page to another to see detailed data for a specific value. Adds interactivity and allows for detailed analysis of specific data points.
    - **Slicer Filters:** Visual filter elements added to the report canvas that allow users to dynamically filter data. Slicers can be based on fields like categories, dates, or numeric ranges.

Filters in Power BI are powerful tools that enhance the flexibility, interactivity, and usability of reports. By using filters effectively, you can tailor your reports to meet specific analytical needs and empower users to explore data dynamically.

2. What is Data Modelling in Power BI ?

Data modeling in Power BI refers to the process of creating relationships between data tables and defining how the data is structured for reporting and analysis. A well-designed data model ensures efficient querying, accurate calculations, and intuitive report building. It is a foundational step in creating meaningful Power BI reports.

Key Components of Data Modeling in Power BI: Tables, Relationships (One-to-Many (1:M), Many-to-Many (M:M), One-to-One (1:1)), Primary and Foreign Keys, Relationships View, Cardinality and Cross-Filter Direction, Measures and Calculated Columns, Hierarchies, etc.

Steps to Build a Data Model in Power BI:

Import Data → Clean and Transform Data → Define Relationships → Optimize the Model → Validate the Model → Create Visualizations

Best Practices for Data Modeling in Power BI:

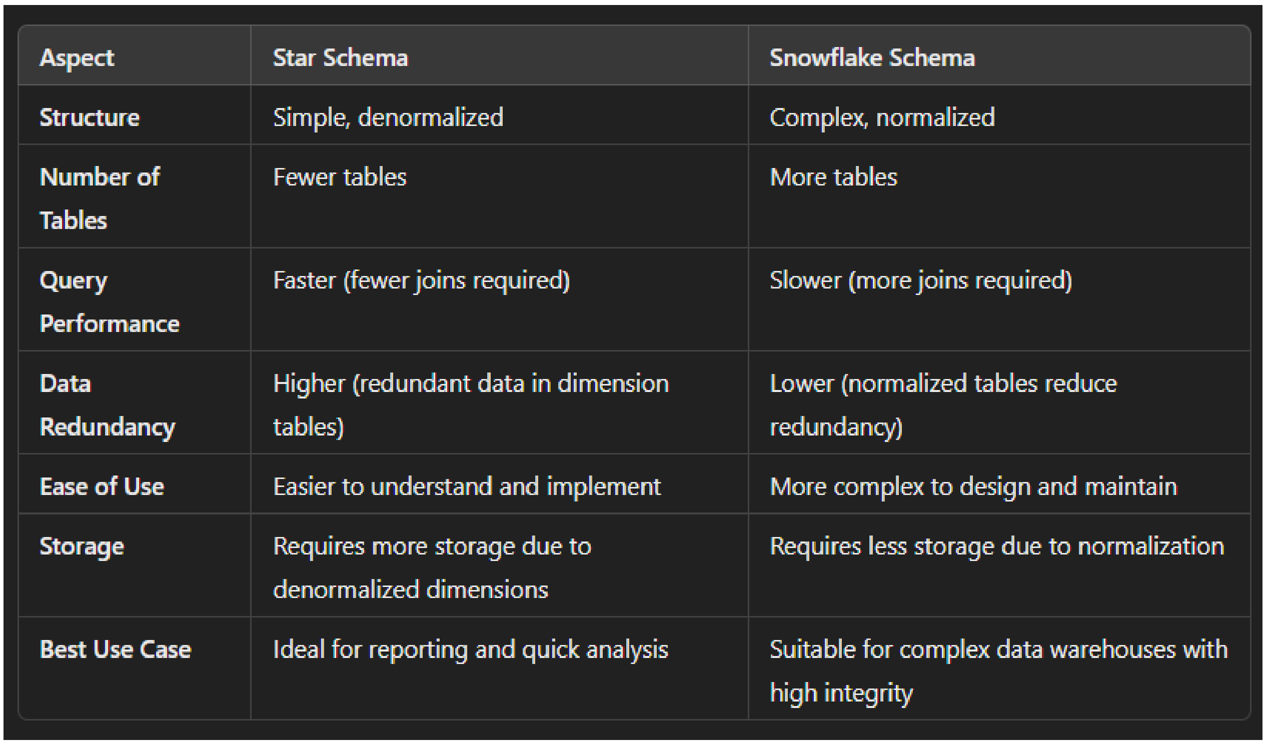
- Use **Star Schema:** Organize the model into fact tables (e.g., Sales) and dimension tables (e.g., Customers, Products).
- Avoid Many-to-Many Relationships: Simplify with bridge tables or calculated columns.
- Optimize Columns and Tables: Remove unused columns and rename fields for clarity.

- Use a Date Table: Create a dedicated date table for time-based analysis.
- Validate Relationships: Regularly test relationships and measures for accuracy.

Star Schema and Snowflake Schema in Data Modeling: The star schema and snowflake schema are two commonly used data modeling techniques in data warehouses and BI tools like Power BI. Both aim to organize data for efficient querying and reporting, but they differ in structure, complexity, and performance.

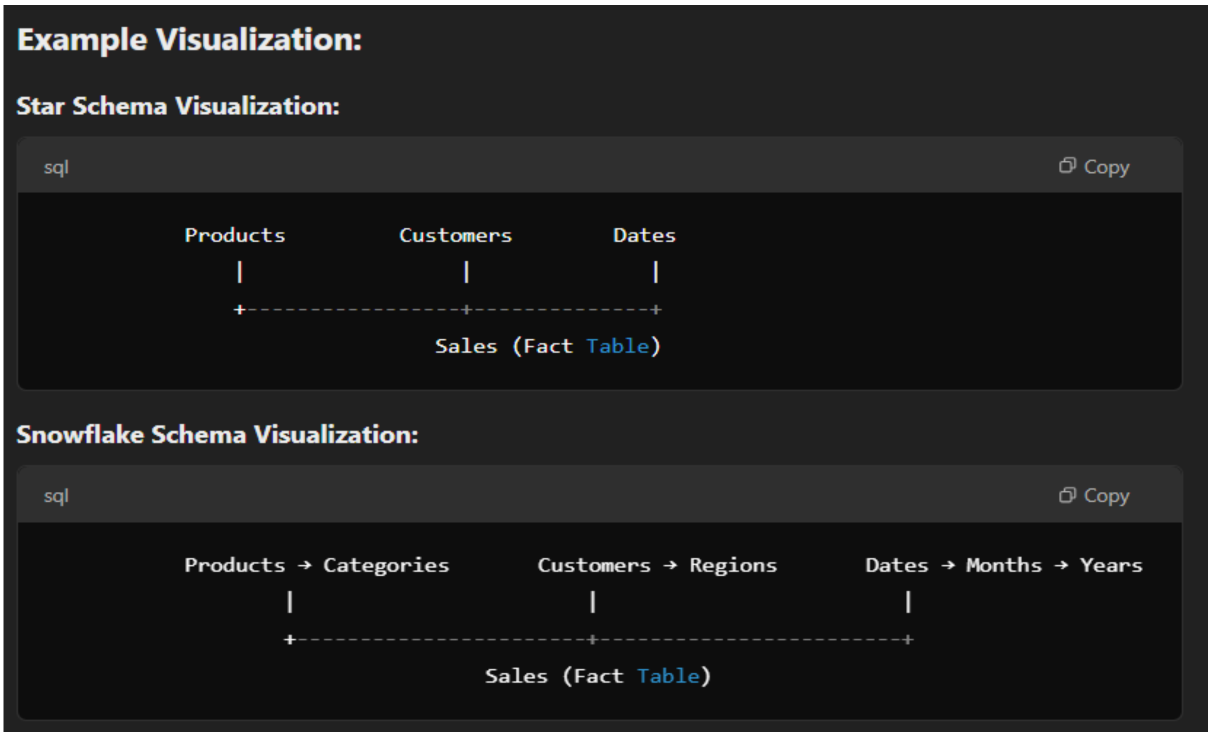
- **Star Schema:** The star schema is the simpler of the two and is widely used in analytics and reporting. It is named for its star-like structure, where a central **fact table** is connected to multiple **dimension tables**.
    - **Fact Table:** Contains the primary measures or metrics of the business, such as sales, revenue, or profit. Includes foreign keys to connect with dimension tables.
    - **Dimension Tables:** Contain descriptive attributes related to the measures, such as product details, customer information, or time dimensions. Are directly connected to the fact table.
    - **Denormalized Structure:** Dimension tables are not normalized, meaning they contain redundant data for simplicity and faster querying.

- **Snowflake Schema:** The snowflake schema is a more complex version of the star schema. It gets its name from its snowflake-like structure, where dimension tables are normalized into multiple related tables.
    - **Fact Table:** Same as in the star schema, containing measures and foreign keys.
    - **Normalized Dimension Tables:** Dimension tables are split into multiple related tables to remove redundancy. Each table represents a level of hierarchy.
    - **Complex Structure:** The schema includes more tables and relationships than the star schema.



Aspect	Star Schema	Snowflake Schema
Structure	Simple, denormalized	Complex, normalized
Number of Tables	Fewer tables	More tables
Query Performance	Faster (fewer joins required)	Slower (more joins required)
Data Redundancy	Higher (redundant data in dimension tables)	Lower (normalized tables reduce redundancy)
Ease of Use	Easier to understand and implement	More complex to design and maintain
Storage	Requires more storage due to denormalized dimensions	Requires less storage due to normalization
Best Use Case	Ideal for reporting and quick analysis	Suitable for complex data warehouses with high integrity

Data modeling in Power BI is a powerful tool for structuring data and enabling insightful, efficient analysis. By designing a robust data model, you ensure high performance, data accuracy, and ease of use for creating interactive reports and dashboards.



Example Visualization:

Star Schema Visualization:

```sql
        Products          Customers          Dates
           |                  |                |
           +------------------+----------------+
                           Sales (Fact Table)
```

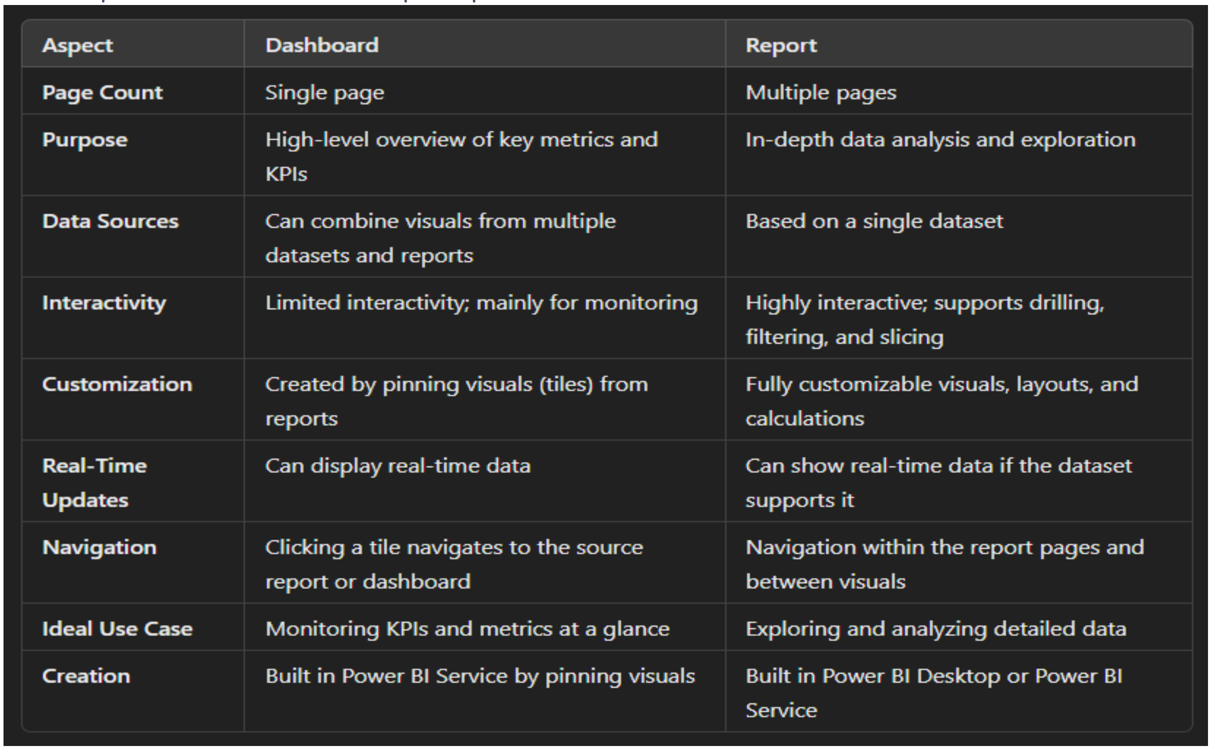
Snowflake Schema Visualization:

```sql
        Products → Categories      Customers → Regions      Dates → Months → Years
               |                          |                         |
               +--------------------------+-------------------------+
                                  Sales (Fact Table)
```

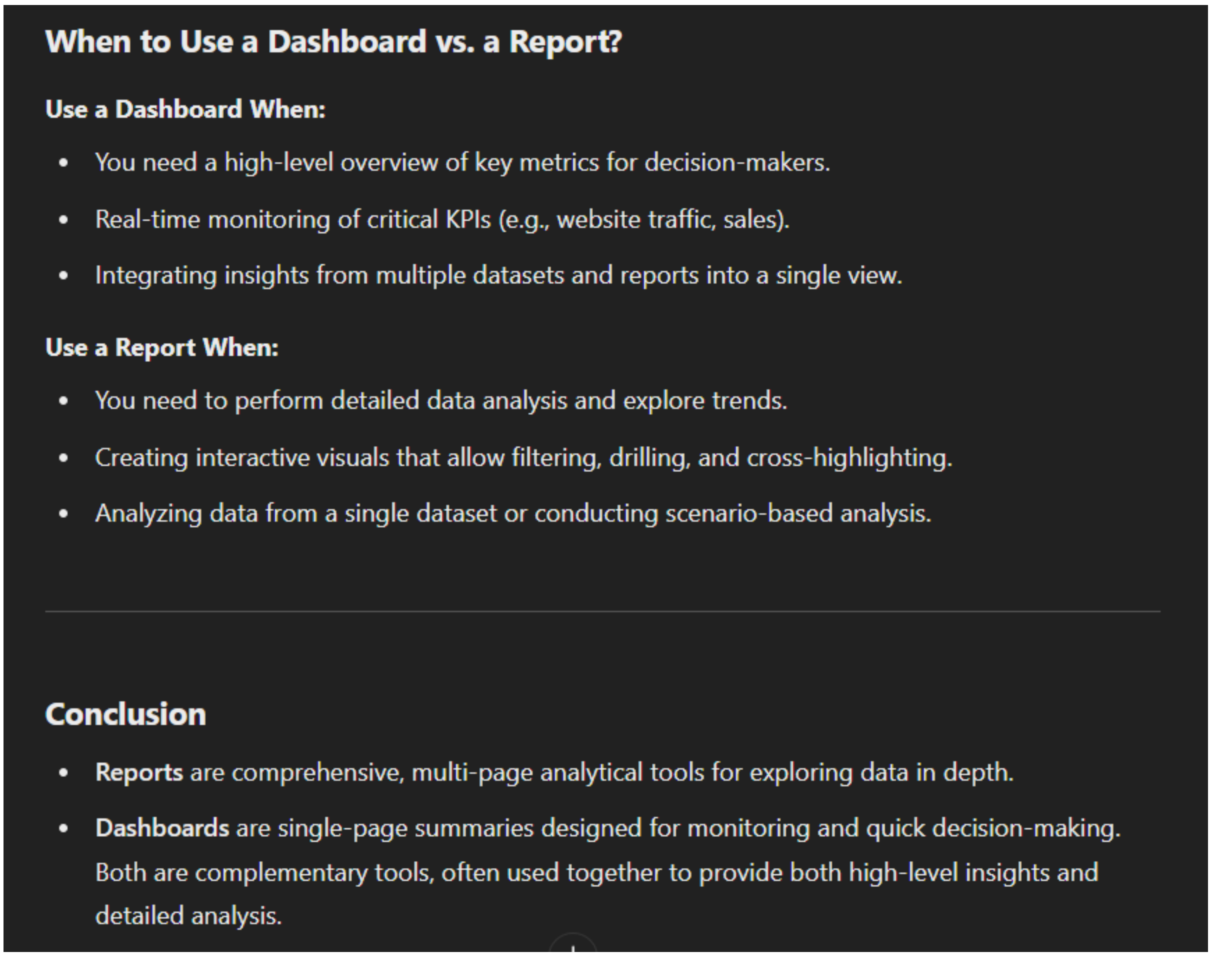
3. **Dashboard and Report in Power BI ?**

Both dashboards and reports are critical components of Power BI, but they serve different purposes and have distinct characteristics.

- **Report:** A **Power BI Report** is a multi-page document containing various visualizations, charts, tables, and metrics based on one or more datasets. It is used to explore and analyze data in detail, offering multiple perspectives through different visuals.
- **Dashboard:** A **Power BI Dashboard** is a single-page, interactive canvas that provides an at-a-glance view of key metrics and insights. It is designed for monitoring purposes and often includes pinned visuals from multiple reports.



Aspect	Dashboard	Report
Page Count	Single page	Multiple pages
Purpose	High-level overview of key metrics and KPIs	In-depth data analysis and exploration
Data Sources	Can combine visuals from multiple datasets and reports	Based on a single dataset
Interactivity	Limited interactivity; mainly for monitoring	Highly interactive; supports drilling, filtering, and slicing
Customization	Created by pinning visuals (tiles) from reports	Fully customizable visuals, layouts, and calculations
Real-Time Updates	Can display real-time data	Can show real-time data if the dataset supports it
Navigation	Clicking a tile navigates to the source report or dashboard	Navigation within the report pages and between visuals
Ideal Use Case	Monitoring KPIs and metrics at a glance	Exploring and analyzing detailed data
Creation	Built in Power BI Service by pinning visuals	Built in Power BI Desktop or Power BI Service



When to Use a Dashboard vs. a Report?

Use a Dashboard When:

- You need a high-level overview of key metrics for decision-makers.
- Real-time monitoring of critical KPIs (e.g., website traffic, sales).
- Integrating insights from multiple datasets and reports into a single view.

Use a Report When:

- You need to perform detailed data analysis and explore trends.
- Creating interactive visuals that allow filtering, drilling, and cross-highlighting.
- Analyzing data from a single dataset or conducting scenario-based analysis.

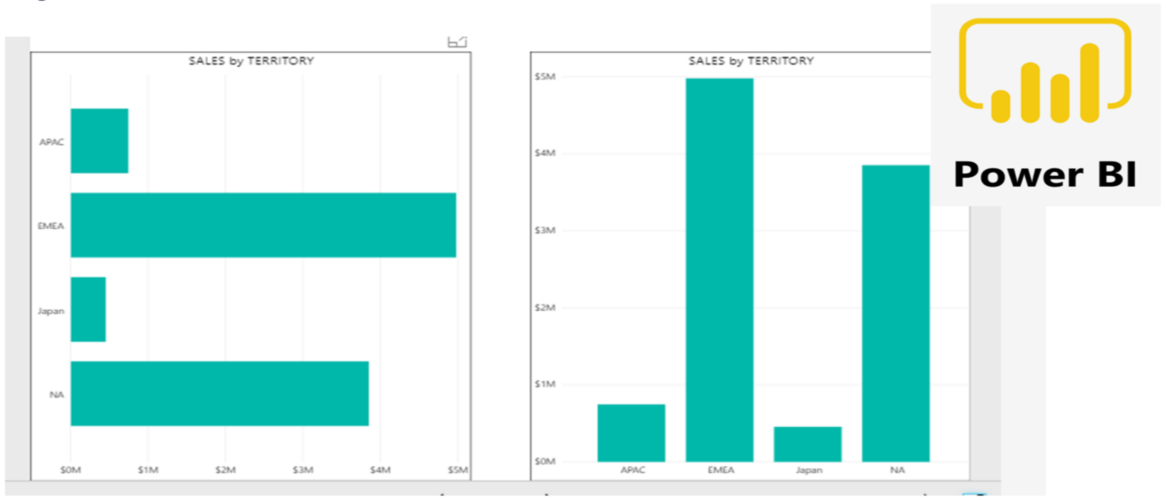
Conclusion

- **Reports** are comprehensive, multi-page analytical tools for exploring data in depth.
- **Dashboards** are single-page summaries designed for monitoring and quick decision-making. Both are complementary tools, often used together to provide both high-level insights and detailed analysis.

4. Types of Visualizations in Power BI?

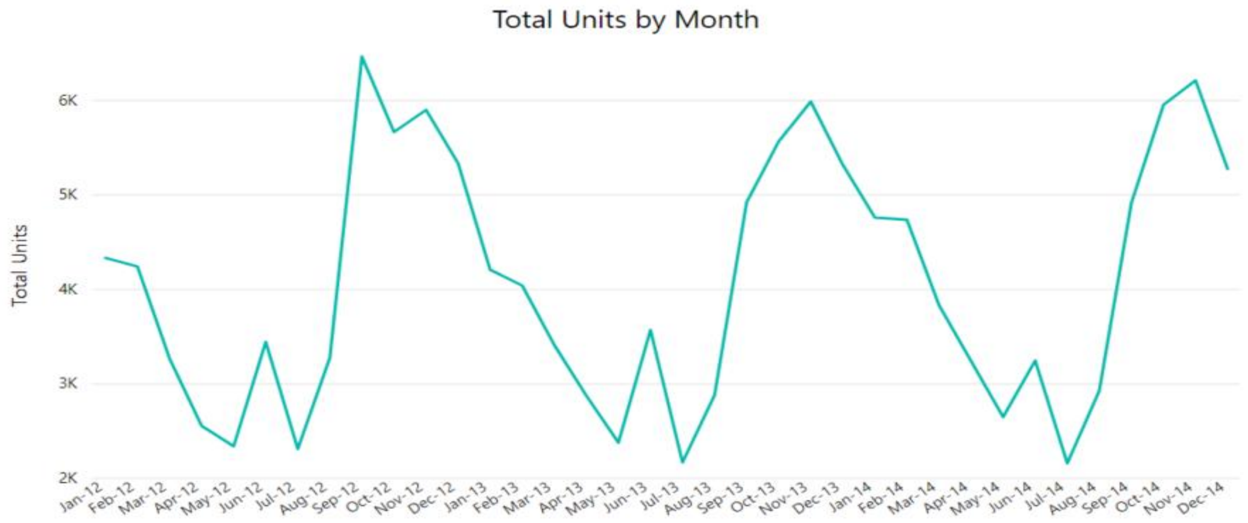
Power BI offers a wide range of visualizations to represent data interactively and effectively. These visuals help users analyze trends, compare values, and derive insights. Below is a detailed explanation of the commonly used types of visualizations in Power BI.

- **Bar and Column Charts: Bar Chart:** Displays data using horizontal bars. **Column Chart:** Displays data using vertical bars. Best for comparing categories or groups. Compare sales performance across regions.

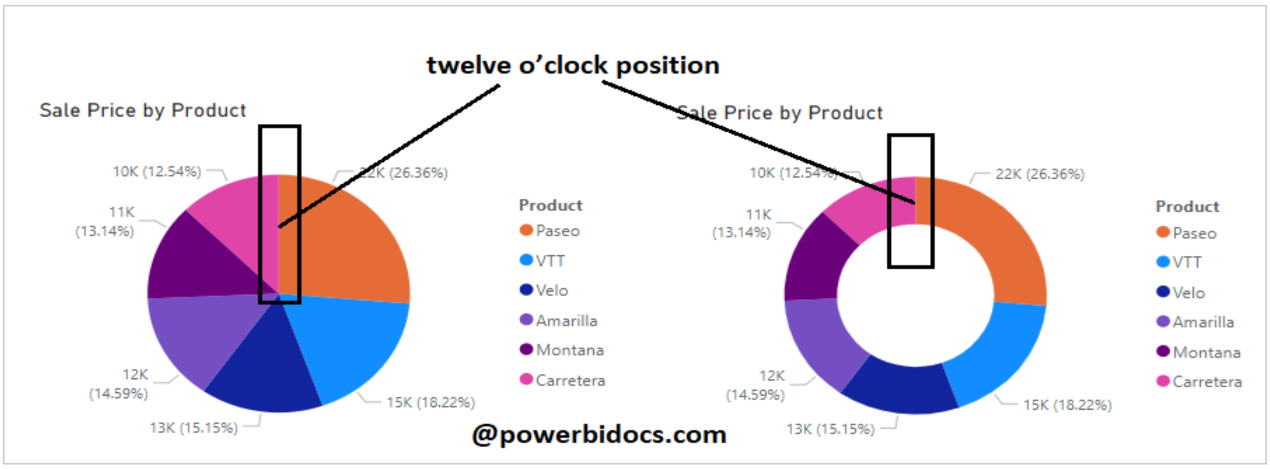


Bars & Column Charts in Power BI

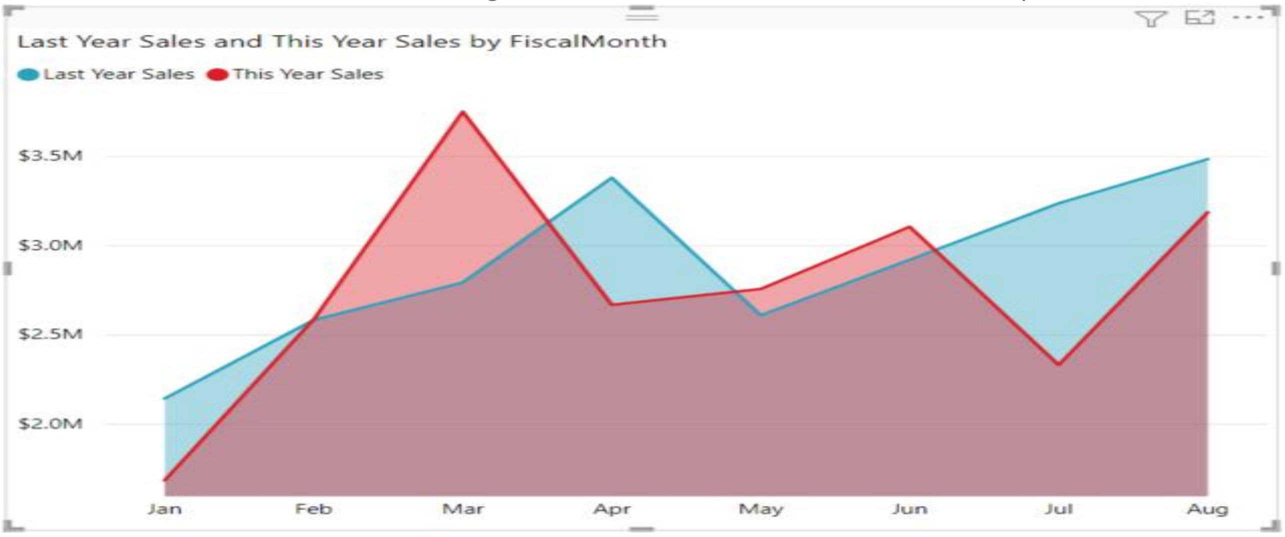
- **Line Chart:** Displays data points connected by a continuous line. Best for analyzing trends over time. Monthly revenue trends for the last year. Revenue growth from January to December.



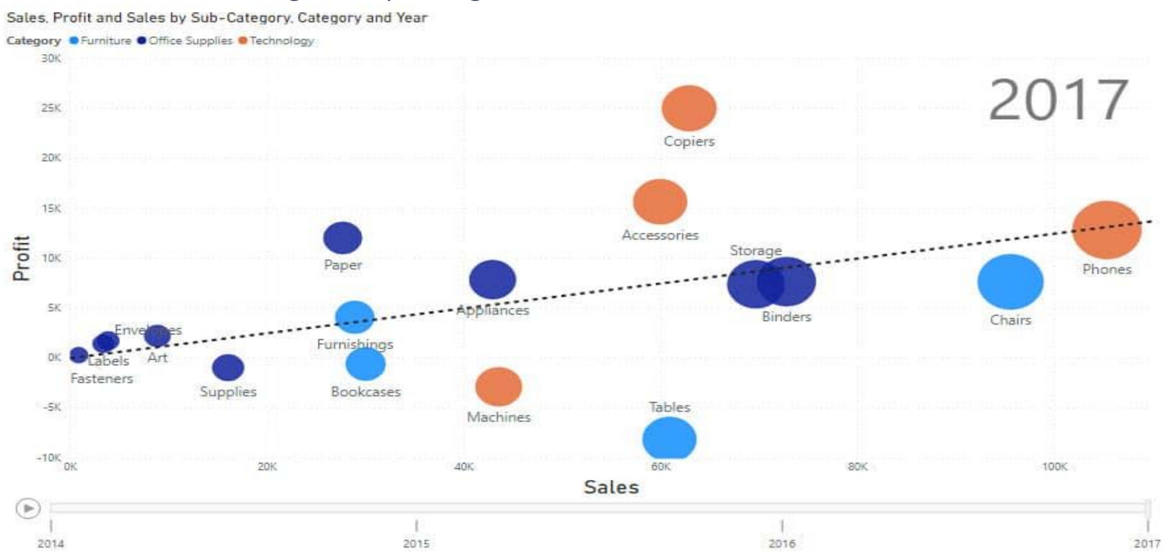
- **Pie and Donut Charts: Pie Chart:** Represents data as slices of a circle. **Donut Chart:** Similar to a pie chart but with a hole in the middle. Best for showing proportions. Market share distribution among competitors. Share of sales across different product categories.



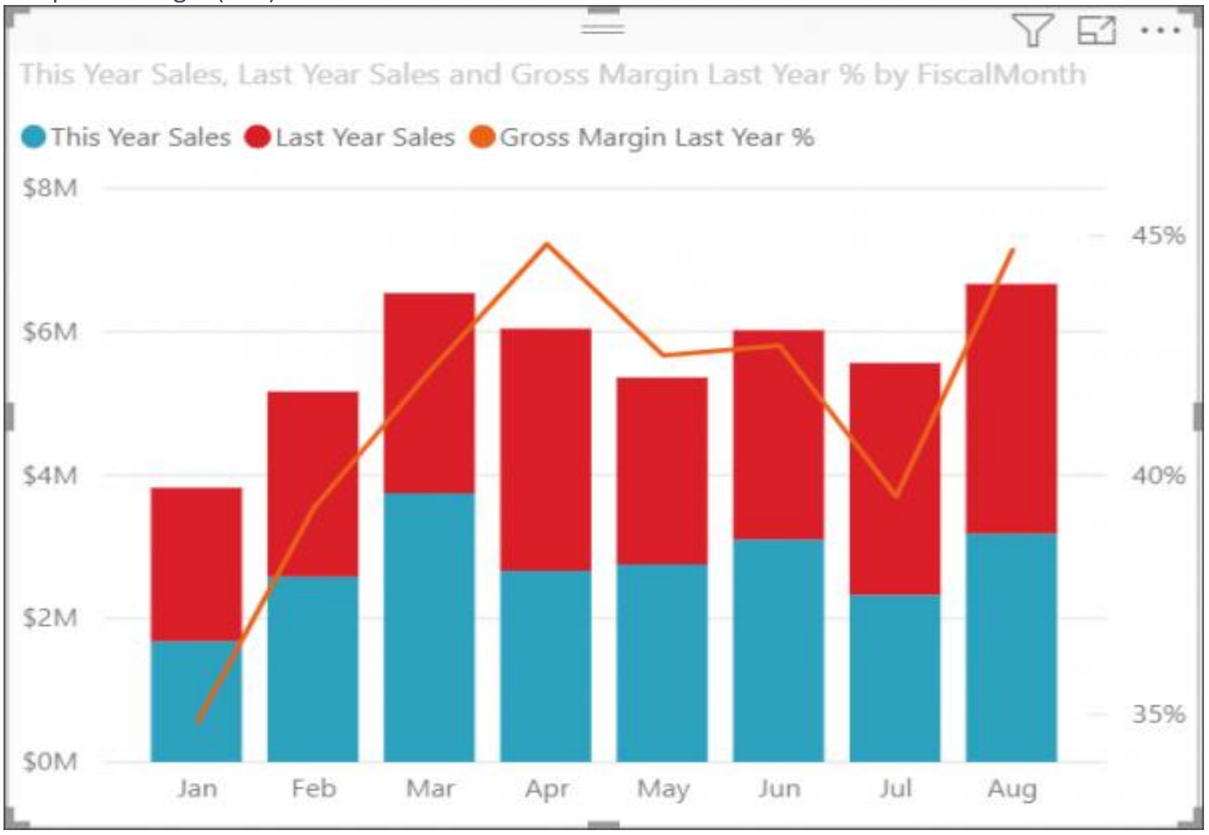
- **Area Chart:** Similar to a line chart but with the area under the line filled. Best for showing cumulative trends. Cumulative sales growth over time. Year-to-date revenue comparison.



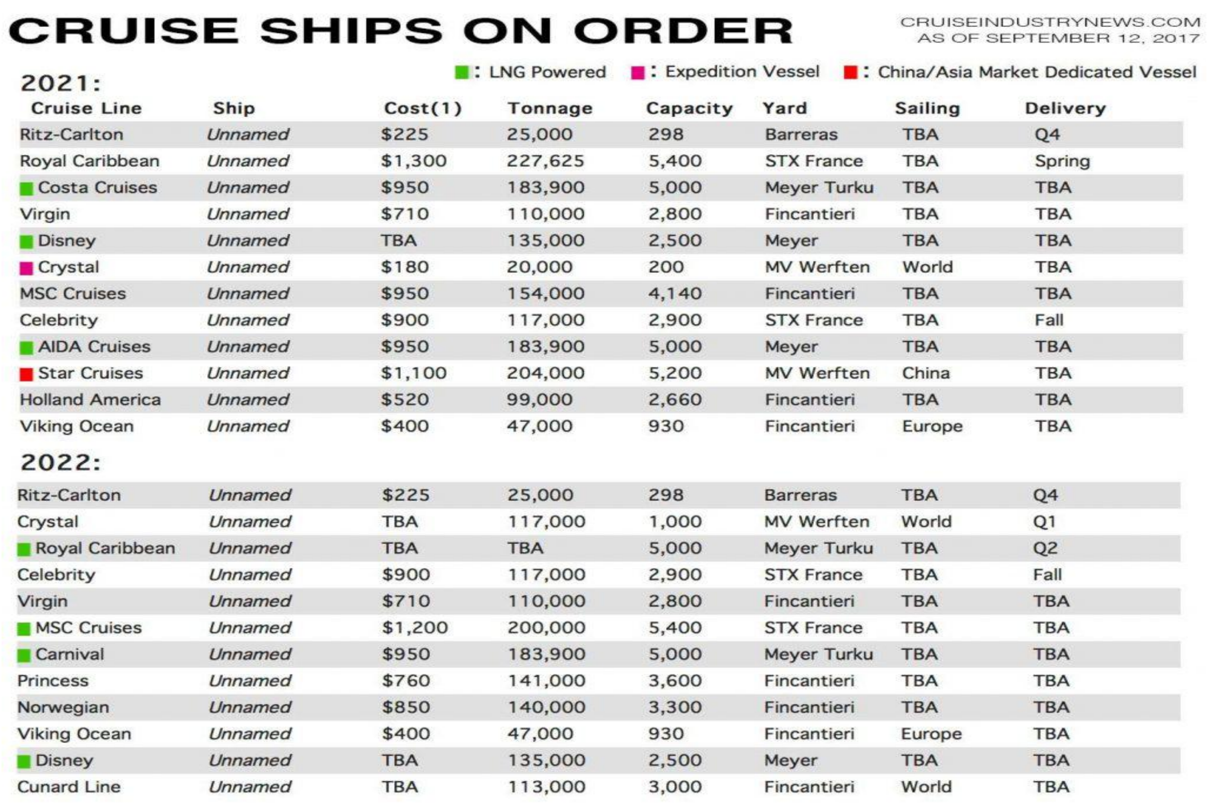
- **Scatter Chart:** Plots data points based on two numerical variables. Best for showing relationships or correlations. Analyze the relationship between advertising spend and sales. Correlation between customer age and spending.



- **Combo Chart:** Combines two chart types, such as line and bar, in one visualization. Best for comparing two different metrics. Sales and profit margins over time. Monthly revenue (bar) and profit margin (line).



- **Table and Matrix: Table:** Displays data in rows and columns. **Matrix:** Similar to a table but allows grouping and drill-down functionality. Best for detailed data views. Tabular sales data by product and region. **Table:** Customer details with total sales. **Matrix:** Sales breakdown by category and subcategory.



CRUISE SHIPS ON ORDER

CRUISEINDUSTRYNEWS.COM
AS OF SEPTEMBER 12, 2017

🟩: LNG Powered 🟪: Expedition Vessel 🟥: China/Asia Market Dedicated Vessel

2021:

Cruise Line	Ship	Cost(1)	Tonnage	Capacity	Yard	Sailing	Delivery
Ritz-Carlton	*Unnamed*	$225	25,000	298	Barreras	TBA	Q4
Royal Caribbean	*Unnamed*	$1,300	227,625	5,400	STX France	TBA	Spring
🟩 Costa Cruises	*Unnamed*	$950	183,900	5,000	Meyer Turku	TBA	TBA
Virgin	*Unnamed*	$710	110,000	2,800	Fincantieri	TBA	TBA
🟩 Disney	*Unnamed*	TBA	135,000	2,500	Meyer	TBA	TBA
🟪 Crystal	*Unnamed*	$180	20,000	200	MV Werften	World	TBA
MSC Cruises	*Unnamed*	$950	154,000	4,140	Fincantieri	TBA	TBA
Celebrity	*Unnamed*	$900	117,000	2,900	STX France	TBA	Fall
🟩 AIDA Cruises	*Unnamed*	$950	183,900	5,000	Meyer	TBA	TBA
🟥 Star Cruises	*Unnamed*	$1,100	204,000	5,200	MV Werften	China	TBA
Holland America	*Unnamed*	$520	99,000	2,660	Fincantieri	TBA	TBA
Viking Ocean	*Unnamed*	$400	47,000	930	Fincantieri	Europe	TBA

2022:

Cruise Line	Ship	Cost(1)	Tonnage	Capacity	Yard	Sailing	Delivery
Ritz-Carlton	*Unnamed*	$225	25,000	298	Barreras	TBA	Q4
Crystal	*Unnamed*	TBA	117,000	1,000	MV Werften	World	Q1
🟩 Royal Caribbean	*Unnamed*	TBA	TBA	5,000	Meyer Turku	TBA	Q2
Celebrity	*Unnamed*	$900	117,000	2,900	STX France	TBA	Fall
Virgin	*Unnamed*	$710	110,000	2,800	Fincantieri	TBA	TBA
🟩 MSC Cruises	*Unnamed*	$1,200	200,000	5,400	STX France	TBA	TBA
🟩 Carnival	*Unnamed*	$950	183,900	5,000	Meyer Turku	TBA	TBA
Princess	*Unnamed*	$760	141,000	3,600	Fincantieri	TBA	TBA
Norwegian	*Unnamed*	$850	140,000	3,300	Fincantieri	TBA	TBA
Viking Ocean	*Unnamed*	$400	47,000	930	Fincantieri	Europe	TBA
🟩 Disney	*Unnamed*	TBA	135,000	2,500	Meyer	TBA	TBA
Cunard Line	*Unnamed*	TBA	113,000	3,000	Fincantieri	World	TBA

- **More visuals -** Card Visualization, Funnel Chart, Gauge Chart, Tree Map, Waterfall Chart, Maps (Basic, Filled, and ArcGIS Maps), KPI Visual, Slicer, Q&A Visual

5. Power BI Workflow?

The **Power BI Workflow** refers to the step-by-step process used to transform raw data into meaningful insights and interactive reports/dashboards. This workflow involves connecting to data sources, cleaning and shaping data, creating relationships, and designing visualizations. Finally, the insights are shared with stakeholders or published for ongoing use.

Components of Power BI Workflow:

- **Data Collection:** Connect Power BI to various data sources like databases, Excel files, cloud services, APIs, etc.
- **Data Transformation (Power Query):** Clean, reshape, and prepare the data using **Power Query Editor**. Removing duplicates, Handling missing values, Creating calculated columns, Unpivoting data for better analysis
- **Data Modeling:** Create relationships between tables and define measures and calculated columns using **DAX (Data Analysis Expressions)**.
- **Data Visualization:** Build interactive reports and dashboards using a variety of visualizations like bar charts, line charts, maps, and slicers. Add filters, slicers, and drill-through capabilities for interactivity.
- **Publishing and Sharing:** Publish reports to the **Power BI Service** for sharing with others. Set up roles and permissions to control access to reports. Schedule data refresh for real-time or periodic updates.
- **Collaboration:** Allowing comments or annotations on reports. Embedding reports into Microsoft Teams or SharePoint. Enabling subscriptions for email notifications.
- Data Refresh and Maintenance: Schedule data refreshes to keep the data up-to-date. Monitor report performance and optimize queries.

6. **Power BI Service?**

Power BI Service is a cloud-based platform provided by Microsoft that enables users to share, collaborate, and access Power BI reports and dashboards. It acts as the online counterpart to Power BI Desktop, allowing organizations to centralize their data insights, manage access, and perform advanced analytics.
It serves as a hub where users can:

- Publish reports created in Power BI Desktop.
- Create dashboards.
- Set up data refresh schedules.
- Collaborate with team members.

Key Features and Components of Power BI Service:

- **Workspaces:** Workspaces are collaborative environments where teams can create, share, and manage reports, dashboards, and datasets. **My Workspace:** Personal space for individual work. **Shared Workspaces:** Collaborative spaces for teams.
- **Dashboards**: Dashboards are single-page, interactive views of data, often combining visuals from multiple reports. Users can pin tiles from different reports to create customized dashboards.
- **Reports:** Multi-page visualizations created in Power BI Desktop and published to the service. Reports allow drill-throughs, interactivity, and detailed exploration.
- **Datasets:** The collection of data used for creating reports and dashboards. Users can manage data refresh, connect new data sources, and modify settings directly in Power BI Service.
- **Sharing and Collaboration:** Share dashboards, reports, and apps with specific users or groups. Features like commenting and annotating allow collaboration directly on visualizations.
- **Apps:** Packaged collections of dashboards and reports shared with a broader audience. Apps are ideal for distributing insights across an organization in a controlled and structured way.
- **Data Refresh:** Schedule automated refreshes for datasets to ensure reports and dashboards show up-to-date information. Supports on-premises and cloud data sources.
- **Row-Level Security (RLS):** Control access to data within a report by defining security roles. Ensures users only see data relevant to them.
- **Mobile Access:** Power BI Service is accessible via the Power BI mobile app, allowing users to view and interact with dashboards and reports on the go.
- **Integration with Other Tools:** Power BI Service is accessible via the Power BI mobile app, allowing users to view and interact with dashboards and reports on the go.

Advantages of Power BI Service:

- **Centralized Data Management:** Provides a single location to manage and access data insights.
- **Real-Time Updates:** Offers live dashboards that update as new data becomes available.
- **Collaboration:** Enables team members to work together and share insights efficiently.
- **Scalability:** Supports large-scale data handling with Power BI Premium for enterprise needs.

7. **What is RLS and its types in Power BI?**

RLS (Row-Level Security) in Power BI is a feature that allows you to restrict access to data for certain users based on their roles or attributes. RLS is often used when you want different users to view only the data that is relevant to them, without the need to create separate reports or datasets. For example, a sales manager in one region should only see sales data for their region and not for others.

Types of RLS in Power BI:

- **Static RLS:** In Static RLS, users are assigned a predefined filter on the data. The filter does not change and is fixed for each user or role. Static RLS is useful when you have predefined groups of

users with fixed data access needs, for example, regional managers only seeing data for their region.
Example: You have a sales dataset with columns: Region, SalesPerson, and SalesAmount. A role North Region Manager is created and assigned a DAX filter Region = "North". Any user assigned this role will only see rows where the Region column is "North."

- **Dynamic RLS:** Dynamic RLS uses DAX expressions to dynamically filter data based on user attributes like username or email. This allows for flexible and scalable security based on the user's identity or a related attribute in the data. Dynamic RLS is useful when user data is dynamic and changes frequently, and you don't want to manually manage the security roles for each user.
Example: Suppose you have a table UserRoles with columns Username and Region. In your sales data, you want each user to see only the data related to their region. You can create a DAX filter for the role such as: [Region] = LOOKUPVALUE(UserRoles[Region], UserRoles[Username], USERNAME()). This filter uses the USERNAME() function to match the current user's username with their region in the UserRoles table, so each user will only see their specific region's data.
- Set up data refresh schedules.
- Collaborate with team members.

Setting up RLS in Power BI:

- **Define Roles and Filters:** In Power BI Desktop, go to the "Modeling" tab, select "Manage Roles," and click "Create." Define a role and a filter using DAX expressions to restrict data access for users in that role.
- **Testing RLS**: You can test the role you created by selecting "View As Roles" from the "Modeling" tab and checking the report as if you were a user in a specific role.
- **Assigning Users to Roles:** After publishing the report to Power BI Service, you can assign specific users or groups to the roles you've created under the "Security" settings of the dataset.

Advantages of RLS:

- **Security:** It ensures that users only see data that they are authorized to view.
- **Scalability:** You don't need to create multiple reports; RLS can handle large user bases with different data access needs.
- **Flexibility:** Dynamic RLS can adapt to changes in user roles, reducing the need for manual updates.

8. What is Refresh and its types in Power BI?

Refresh in Power BI is the process of updating the data in your Power BI reports and dashboards to reflect the latest changes in the underlying data sources. This ensures that users have access to up-to-date information.

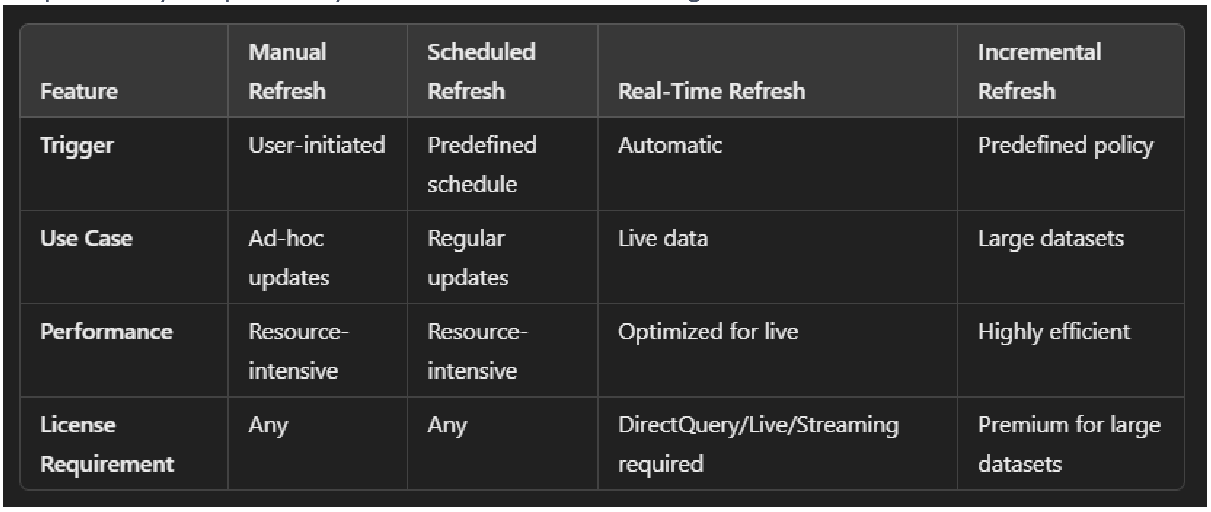
Types of Refreshes in Power BI:

- **Manual Refresh:** Users manually trigger the refresh process to pull the latest data from the source. Suitable for ad-hoc updates when immediate data changes need to be reflected. Eg. You have an Excel file on OneDrive as the data source. When the data in the Excel file is updated, you manually refresh the dataset in Power BI Desktop or Power BI Service to include the latest changes.

- **Scheduled Refresh:** Power BI automatically refreshes the dataset at predefined intervals based on a schedule you configure. Best for data sources that are updated regularly, such as daily or weekly. You can set multiple refresh times. Limited to 8 refreshes per day for Power BI Pro users and 48 refreshes for Power BI Premium users. Eg. Your report is based on a SQL Server database that updates daily at midnight. You configure a scheduled refresh in Power BI Service to refresh the dataset every day at 1:00 AM.

- **Automatic/Real-Time Refresh:** Data is automatically updated in real-time as changes occur in the data source, without the need for manual or scheduled refreshes. For scenarios where near real-time data is critical, such as monitoring live events or operational dashboards. Requires data sources that support DirectQuery, Live Connection, or streaming datasets. Eg. A dashboard monitoring IoT sensors or stock market prices updates in real time using a streaming dataset connected via an API.

- **Incremental Refresh:** Only the new or changed data is refreshed instead of reloading the entire dataset. This significantly reduces refresh time and resource consumption. Ideal for large datasets where only a small portion of data changes over time. Requires a Premium license for datasets larger than 1 GB. Involves configuring parameters for RangeStart and RangeEnd to define the refresh window. Eg. A sales dataset contains historical data from 2015 to the present. You configure an incremental refresh policy to update only the past 7 days of data instead of refreshing the entire dataset.



Feature	Manual Refresh	Scheduled Refresh	Real-Time Refresh	Incremental Refresh
Trigger	User-initiated	Predefined schedule	Automatic	Predefined policy
Use Case	Ad-hoc updates	Regular updates	Live data	Large datasets
Performance	Resource-intensive	Resource-intensive	Optimized for live	Highly efficient
License Requirement	Any	Any	DirectQuery/Live/Streaming required	Premium for large datasets

Refresh in Power BI is a crucial feature for keeping your reports and dashboards up-to-date. By understanding the types of refresh, you can choose the most appropriate method for your business needs, ensuring both efficiency and accuracy in your data reporting.

9. What are different Storage & Connection Modes in Power BI ?

Power BI provides multiple **storage and connection modes** to manage how data is accessed and processed for reports and dashboards. These modes determine where the data resides (in-memory, live connection, or on-demand) and how Power BI interacts with the underlying data sources.

Types of Storage & Connection Modes in Power BI:

- **Import Mode:** Data is imported and stored in the in-memory engine of Power BI (VertiPaq). This mode provides the fastest query performance since data is preloaded into memory. Ideal for small to medium-sized datasets. Scenarios where high-speed performance and flexibility are needed. Full access to DAX functions. Data refreshes are required to keep the data up-to-date.

- **DirectQuery Mode:** Data remains in the source system, and queries are executed directly on the source each time a visual is interacted with in the report. When real-time or near real-time data is required. For large datasets that cannot fit into memory. No data is stored in Power BI. Supports row-level security (RLS). May experience slower performance compared to Import Mode.

- **Live Connection:** A variation of DirectQuery but specifically used for live connections to Analysis Services (SSAS, Azure AS, or Power BI datasets). The entire model is managed in the source system. No local model in Power BI. Limited transformation capabilities in Power BI.

- **Composite Model:** Combines Import and DirectQuery modes in the same model, allowing you to choose the best storage mode for each table in the dataset. When some data needs high performance (Import) and other data requires up-to-date information (DirectQuery). Offers flexibility for handling different data needs. Requires careful design to balance performance and real-time requirements.

Feature	Import Mode	DirectQuery Mode	Live Connection	Composite Model	Push Dataset
Data Storage	Stored in Power BI's in-memory engine (VertiPaq).	Data remains in the source system.	Data remains in the source system.	Mixed: In-memory and source system.	Streamed data, no persistent storage by default.
Performance	High (due to in-memory storage).	Medium (dependent on source query performance).	Medium (depends on Analysis Services or Power BI dataset performance).	Varies (depends on the mix of Import and DirectQuery tables).	High for streaming visuals, but limited for historical data.
Data Freshness	Requires manual or scheduled refresh.	Real-time or near real-time (queries source on demand).	Real-time or near real-time.	Real-time for DirectQuery tables; refreshed for imported tables.	Real-time data only.
Size Limitation	Limited to 1 GB per dataset in Power BI Service (Pro license).	No size limitation; relies on source system capacity.	No size limitation; relies on source system capacity.	Varies based on the mix of Import and DirectQuery tables.	Data streamed in real-time; no historical data stored.
DAX Functionality	Full DAX capabilities supported.	Limited DAX functionality for on-demand queries.	Limited DAX functionality (depends on source model).	Full for imported data; limited for DirectQuery data.	Very limited DAX support.
Transformations	Extensive transformations supported in Power Query.	Limited transformations in Power BI; done at the source.	Limited; transformations must be in the source model.	Import tables support full transformations; DirectQuery tables are limited.	Minimal to none.
Refresh Options	Manual or scheduled refresh.	No refresh needed; queries are on-demand.	No refresh needed; live connection handles updates.	Scheduled refresh for imported data; real-time for DirectQuery.	Real-time updates.
Use Case	Small to medium-sized datasets needing fast performance and flexibility.	Large datasets requiring up-to-date data or on-demand queries.	Enterprise-grade models managed in Analysis Services or Power BI datasets.	Mixed scenarios where some data is real-time and others are historical.	Real-time dashboards and streaming data use cases.
Examples	Sales data imported from Excel or SQL Server for analysis.	Financial data queried directly from a SQL database.	Connection to Azure Analysis Services or SSAS for enterprise reporting.	Historical sales data (Import) + real-time transactions (DirectQuery).	IoT sensor data streaming into a live dashboard.
Licensing Requirements	Power BI Pro or Premium.	Power BI Pro or Premium.	Power BI Pro or Premium.	Power BI Premium recommended for large datasets.	Power BI Pro or Premium.

10. What DAX in Power BI? And It’s All Features.

DAX, short for **Data Analysis Expressions**, is a formula language used in Microsoft Power BI, Excel Power Pivot, and SQL Server Analysis Services (SSAS) for creating custom calculations and aggregations. DAX is designed for working with data models and provides functions and operators to perform advanced data manipulation, create calculated columns, measures, and tables, and enhance data analysis capabilities.

Key Features of DAX:

- **Expression Language:** DAX formulas return a single value (scalar or table) and are used to define measures, calculated columns, or calculated tables.
- **Powerful Aggregation:** DAX provides functions for summarizing data, such as SUM, AVERAGE, and COUNT, as well as advanced calculations like running totals and moving averages.
- **Context-Aware**: DAX operates based on **row context** and **filter context**, allowing calculations to adapt dynamically based on the user’s data selection.
- **Optimized for Relational Data**: DAX is designed to work with tables, relationships, and hierarchies in data models.
- **Reusable Logic**: Measures and calculated columns defined with DAX can be reused throughout the Power BI model and reports.

DAX Engine: The DAX Engine in Power BI is a critical component responsible for processing and executing DAX queries and formulas. It works in conjunction with the VertiPaq Engine (the in-memory data storage engine in Power BI) to deliver fast and efficient data analysis. The DAX Engine is designed to:
Interpret and optimize DAX expressions, Retrieve and manipulate data stored in memory, Interact with the VertiPaq Engine or DirectQuery for data access, It translates high-level DAX formulas into queries that the data engine can execute, ensuring optimized performance

Key Components of the DAX Engine:

- **Query Engine:** Responsible for interpreting the DAX query and determining the most efficient way to retrieve the required data. Breaks down DAX formulas into logical steps, creating an execution plan.
- **Formula Engine (FE):** Handles logical operations, computations, and query plans. Processes DAX formulas and calculates results. Determines the execution strategy for a query. Communicates with the Storage Engine to retrieve raw data.
- **Storage Engine (SE):** Retrieves data from the underlying data source, such as the VertiPaq in-memory engine, DirectQuery, or Hybrid models. Works with compressed and optimized columnar data storage.
- **VertiPaq Engine:** The primary storage engine for most Power BI data models. Stores data in a highly compressed, columnar format to enable fast reads. Works seamlessly with the DAX Engine to perform operations like filtering, sorting, and aggregation.

Workflow of the DAX Engine:

- **User Interaction:** A user interacts with a report (e.g., applying a slicer or clicking a visual).
- **DAX Query Generation**: The DAX Engine translates user actions into DAX queries.
- **Query Execution:** The Formula Engine processes logical operations and communicates with the Storage Engine for data retrieval.
- **Data Retrieval:** The Storage Engine retrieves the relevant data and performs aggregations (if required).
- **Result Return**: The DAX Engine combines the results and returns them to the visualization layer.

Optimization Techniques for the DAX Engine:

- **Reduce Complex Logic in Calculated Columns:** Move calculations to the source or pre-aggregate data.
- **Leverage Measures Over Calculated Columns:** Measures are dynamic and computed only when needed, reducing memory usage.
- **Minimize Row Context Operations:** Use aggregation functions like SUMX sparingly.

- **Avoid Complex Filters**: Simplify filtering logic using efficient DAX functions like KEEPFILTERS or ALL.
- **Optimize Data Model:** Reduce cardinality and relationships in the model.

Context in Power BI: In Power BI, **context** is a fundamental concept that determines how DAX formulas are evaluated. It defines the **scope** of the calculation or operation based on the data being referenced. Context influences what data is considered by a formula and how the results are computed. Understanding **context** in Power BI is essential for building accurate and optimized reports. It ensures that DAX formulas behave as expected, enabling dynamic and powerful calculations. Let me know if you'd like more examples or further clarification!

Types of Contexts in Power BI:

- **Row Context:** Row context applies when a formula is calculated for a specific row in a table. It occurs in calculated columns or when using iterator functions like SUMX, AVERAGEX, or FILTER. DAX formulas in row context operate as if they have access to the current row and its related data.

- **Filter Context:** Filter context applies when filters or slicers are used in a report. It narrows the data considered by a DAX calculation based on applied filters, slicers, or report page interactions. Slicers, filters, or visuals in the report. Explicit DAX formulas, such as CALCULATE or FILTER, which manipulate filters.

- **Query Context:** Query context is determined by the data that Power BI visualizations or queries request. It defines what subset of data is retrieved and displayed.

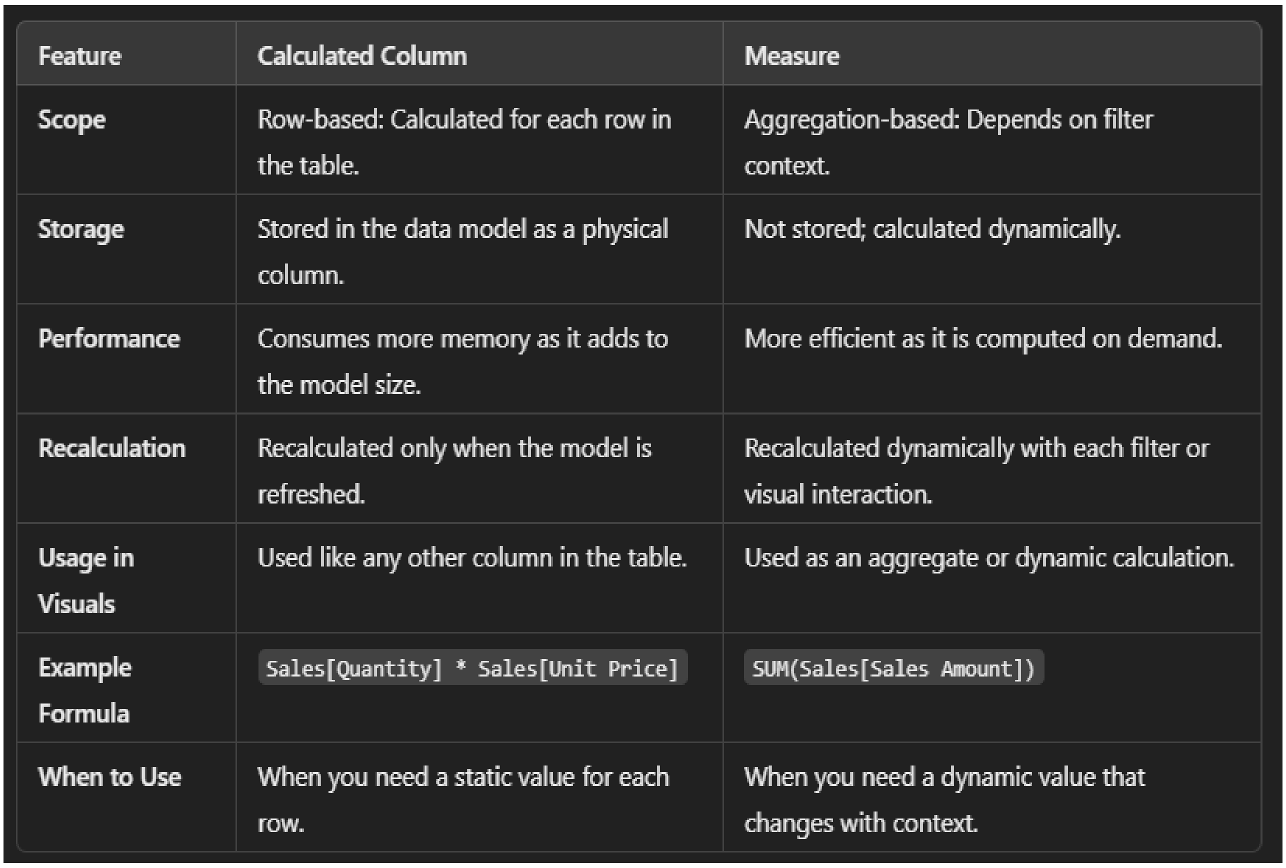
Calculated Column & Measure: In Power BI, **calculated columns** and **measures** are two essential ways to perform calculations on data, but they are used for different purposes and behave differently.

- **Calculated Column:** A calculated column adds a new column to a table based on a DAX formula, with each row having a value computed based on the formula. The column is **row-based**, meaning it evaluates the formula for every row in the table. It is stored in the data model, increasing the size of the model. Calculated columns are static after they are created, meaning they are recalculated only when the table is refreshed. When you need a value that will be used as a field in visuals (e.g., creating categories or flags). When the calculation depends on a row-by-row context.
  Example: Total Sales = Sales[Quantity] * Sales[Unit Price]

- **Measure:** A measure is a dynamic calculation that performs aggregations or computations based on the current filter context. Measures are **aggregation-based**, meaning they summarize or calculate data dynamically based on slicers, filters, or visuals. Measures do not store results; they are computed on demand, making them memory-efficient. Measures are recalculated every time the filter context changes. When performing dynamic calculations or aggregations that depend on filters, slicers, or visualizations. For measures like total sales, averages, percentages, or ratios.
  Example: Total Sales Measure = SUM(Sales[Total Sales])

- **When to Use Calculated Columns or Measures:**
  - **Use Calculated Columns**: When creating row-level fields like categories, flags, or conditions. If you need to use the new column in relationships or hierarchies.
    Example: Revenue = Sales[Quantity] * Sales[Unit Price]

  - **Use Measures**: For dynamic aggregations like totals, averages, or percentages. When performance and memory optimization are priorities.
    Example: Total Revenue = SUM(Sales[Revenue])

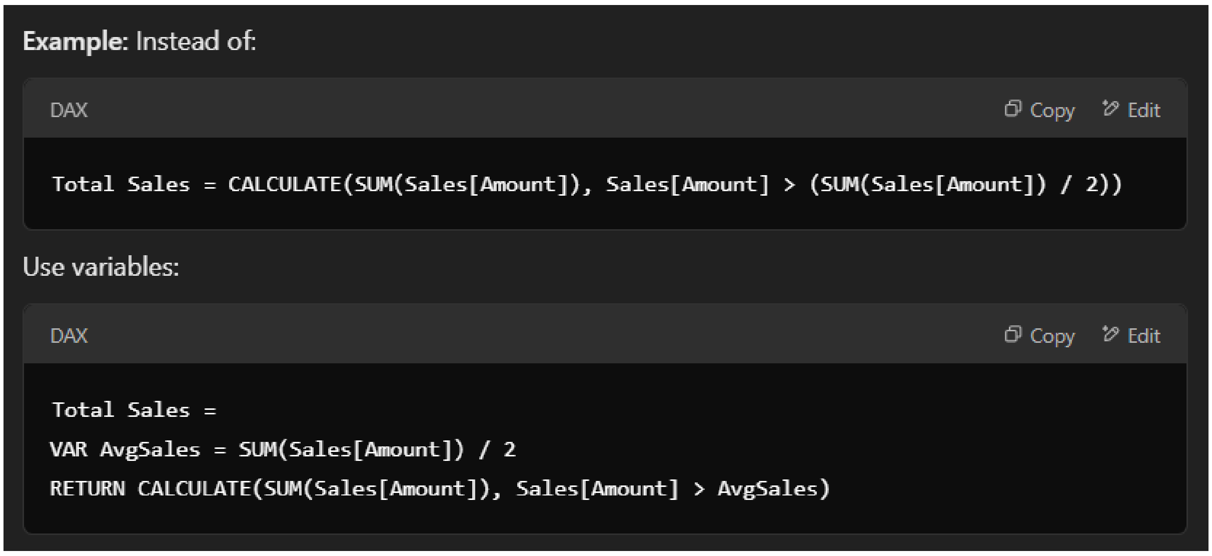


Feature	Calculated Column	Measure
Scope	Row-based: Calculated for each row in the table.	Aggregation-based: Depends on filter context.
Storage	Stored in the data model as a physical column.	Not stored; calculated dynamically.
Performance	Consumes more memory as it adds to the model size.	More efficient as it is computed on demand.
Recalculation	Recalculated only when the model is refreshed.	Recalculated dynamically with each filter or visual interaction.
Usage in Visuals	Used like any other column in the table.	Used as an aggregate or dynamic calculation.
Example Formula	`Sales[Quantity] * Sales[Unit Price]`	`SUM(Sales[Sales Amount])`
When to Use	When you need a static value for each row.	When you need a dynamic value that changes with context.

11. How to Optimization Dax queries in Power BI ?

Optimizing DAX queries in Power BI is crucial for improving performance, especially when dealing with large datasets. Below are detailed strategies and techniques for optimizing DAX queries:

- **Avoid Complex Calculations in Filter Expressions:** Complex expressions inside CALCULATE, FILTER, or SUMX can slow down query performance because Power BI needs to process these calculations for each row.



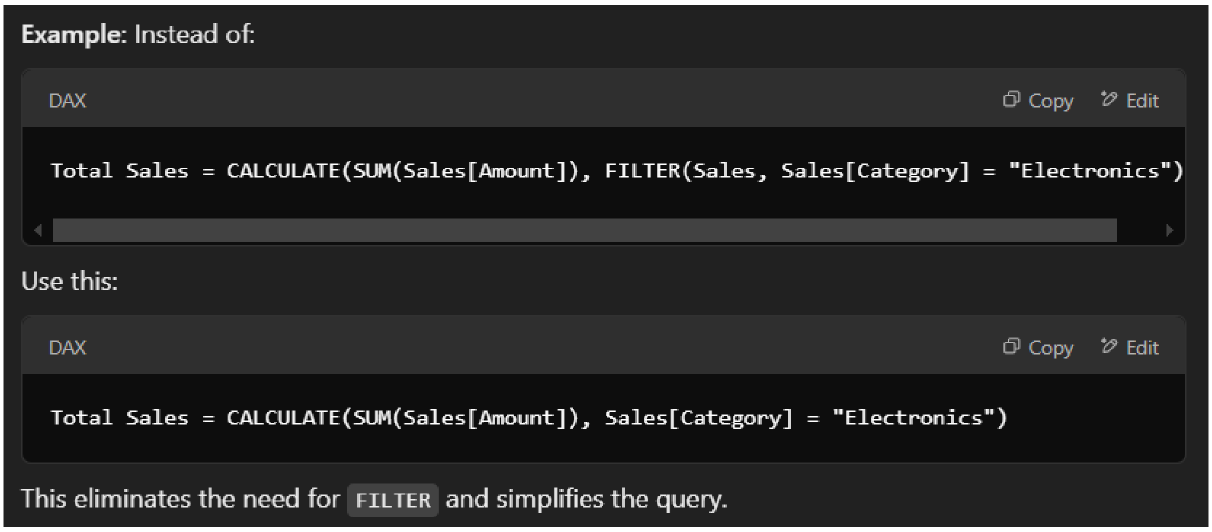
Example: Instead of:

```
Total Sales = CALCULATE(SUM(Sales[Amount]), Sales[Amount] > (SUM(Sales[Amount]) / 2))
```

Use variables:

```
Total Sales =
VAR AvgSales = SUM(Sales[Amount]) / 2
RETURN CALCULATE(SUM(Sales[Amount]), Sales[Amount] > AvgSales)
```

- **Minimize the Use of FILTER:** FILTER can be slow because it evaluates each row in the table and returns a table, which can lead to inefficiency. Where possible, use simpler alternatives like direct column references inside CALCULATE or use ALL, ALLEXCEPT, or REMOVEFILTERS to modify filter context more efficiently.



Example: Instead of:

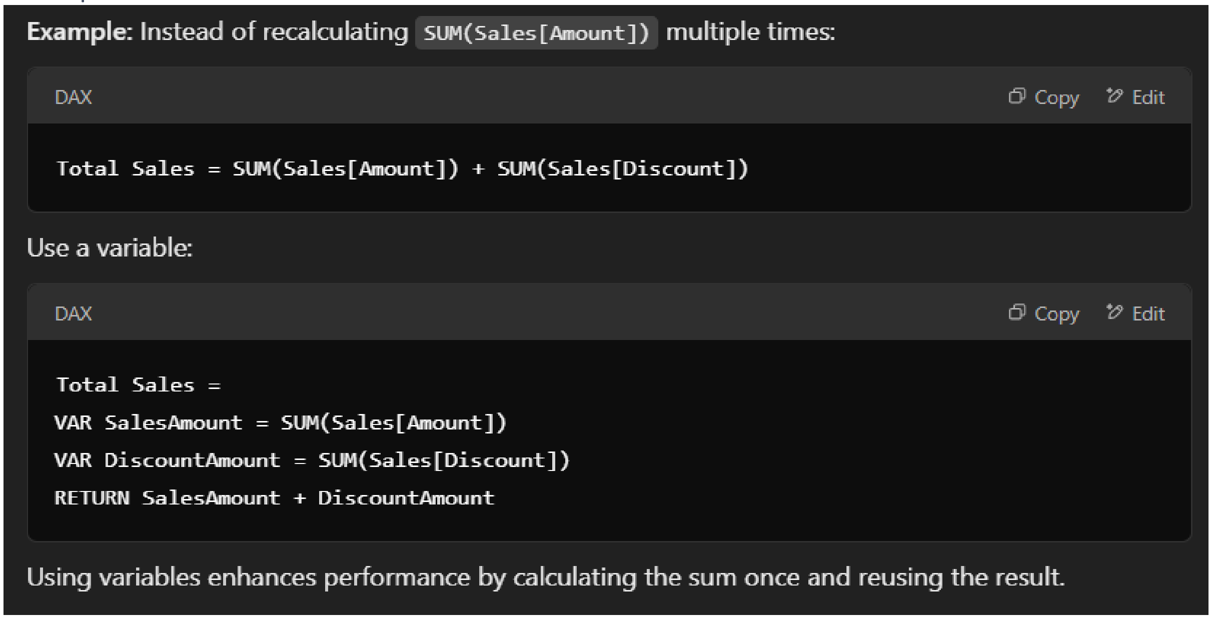
```
Total Sales = CALCULATE(SUM(Sales[Amount]), FILTER(Sales, Sales[Category] = "Electronics")
```

Use this:

```
Total Sales = CALCULATE(SUM(Sales[Amount]), Sales[Category] = "Electronics")
```

This eliminates the need for `FILTER` and simplifies the query.

- **Use Variables (VAR):** Using VAR helps avoid recalculating the same expression multiple times, making the query more efficient. Store intermediate results in variables and use them within the expression.



Example: Instead of recalculating `SUM(Sales[Amount])` multiple times:

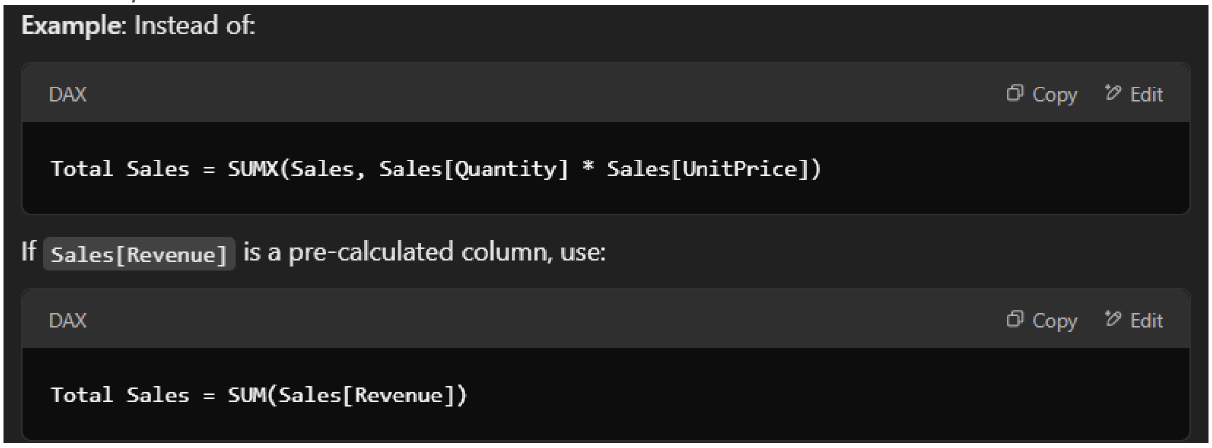
```
Total Sales = SUM(Sales[Amount]) + SUM(Sales[Discount])
```

Use a variable:

```
Total Sales =
VAR SalesAmount = SUM(Sales[Amount])
VAR DiscountAmount = SUM(Sales[Discount])
RETURN SalesAmount + DiscountAmount
```

Using variables enhances performance by calculating the sum once and reusing the result.

- **Avoid Nested Iterators (e.g., SUMX, AVERAGEX, COUNTX):** Iterators like SUMX, AVERAGEX, and COUNTX are useful but can slow down performance when used in deeply nested calculations. Minimize nested iterators, or try to use aggregate functions like SUM, AVERAGE, or COUNT directly instead.



Example: Instead of:

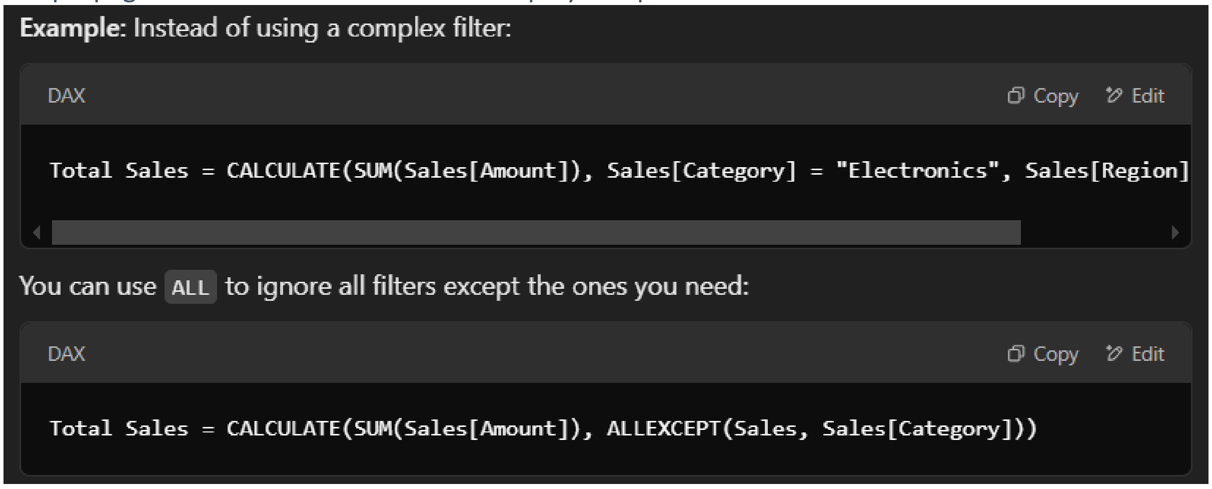
```
Total Sales = SUMX(Sales, Sales[Quantity] * Sales[UnitPrice])
```

If `Sales[Revenue]` is a pre-calculated column, use:

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

- **Optimize Relationships:** Inefficient relationships between tables can slow down performance, especially with large datasets. Ensure that relationships are properly indexed, use **star schemas**, and avoid complex many-to-many relationships where possible.
    - **Best Practices**:
        - Use **single-directional** relationships unless there's a strong reason for **bidirectional**.
        - Ensure that relationships are between **key columns** with unique values in the lookup table.
        - If possible, **filter by primary keys** instead of directly using fact tables in relationships.

- **Use ALL, ALLSELECTED, ALLEXCEPT to Control Filter Context:** These functions help you modify or remove filters from the calculation, which can improve performance by limiting unnecessary filter propagation. Use these functions to simplify complex filter scenarios and avoid excessive recalculation.



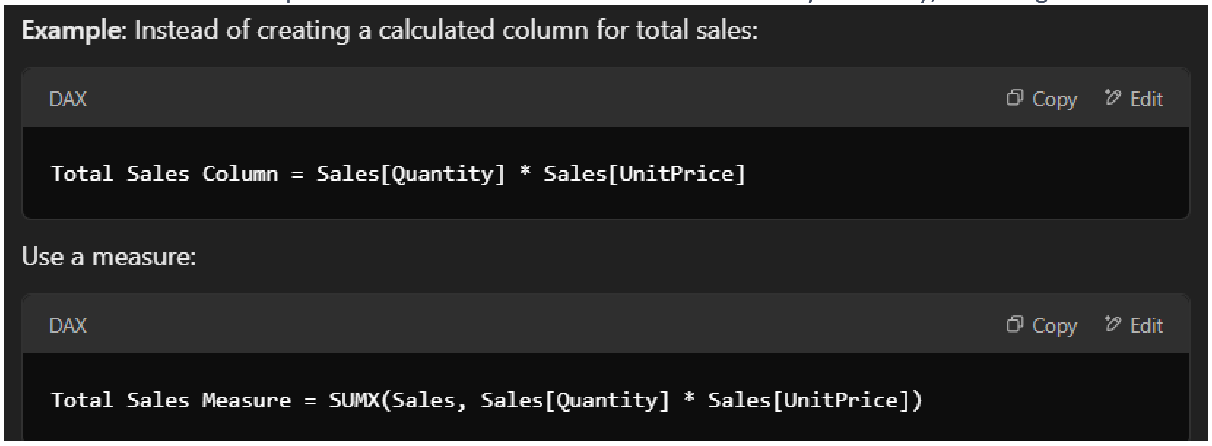
Example: Instead of using a complex filter:

```DAX
Total Sales = CALCULATE(SUM(Sales[Amount]), Sales[Category] = "Electronics", Sales[Region]
```

You can use `ALL` to ignore all filters except the ones you need:

```DAX
Total Sales = CALCULATE(SUM(Sales[Amount]), ALLEXCEPT(Sales, Sales[Category]))
```

- **Limit the Use of Calculated Columns (Use Measures Where Possible):** Calculated columns add data to the model, which increases memory usage and processing time. Use measures instead of calculated columns whenever possible because measures are calculated dynamically, reducing the load on memory.



Example: Instead of creating a calculated column for total sales:

```DAX
Total Sales Column = Sales[Quantity] * Sales[UnitPrice]
```

Use a measure:

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

- **Monitor and Debug with DAX Studio:** Analyzing the performance of DAX queries is essential for identifying bottlenecks and optimizing performance. Use **DAX Studio** to analyze your DAX queries and check the **query plan** and **server timings** to identify performance issues.

Optimizing DAX queries in Power BI involves a combination of reducing unnecessary calculations, simplifying filter expressions, minimizing iterators, and using the right data structures and types. By adhering to these best practices, you can ensure that your reports perform efficiently even with large datasets.

12. How to Optimization Power BI Performance ?

Optimizing Power BI performance is crucial for ensuring that your reports and dashboards run smoothly, especially with large datasets or complex calculations. Here are several strategies to improve Power BI performance:

- **Reduce Data Load:**
  - **Limit the columns and rows:** Only load the necessary data into your model. Remove unnecessary columns and rows that won't be used in reports.
  - **Use DirectQuery or Live Connection:** For large datasets, use DirectQuery or live connection to avoid importing data into Power BI, relying on the database to perform the calculations.
- **Optimize Data Model:**
  - **Star Schema:** Organize your data model into a star schema (fact tables and dimension tables) for better performance. This reduces complexity and improves query performance.
  - **Avoid Bi-Directional Relationships:** Use single-directional relationships wherever possible. Bi-directional relationships can add overhead and slow down performance.
  - **Data Types**: Choose the appropriate data types for your columns (e.g., integers instead of decimals, and avoid using text for numerical values).
  - **Disable Auto Date/Time**: Power BI automatically creates hidden date tables for date fields, which can increase file size. Disable this in "Options" to avoid unnecessary tables.
- **Optimize DAX Measures and Calculations:**
  - **Use Variables in DAX**: Store intermediate results using variables to avoid recalculating the same values multiple times in a measure.
  - **Avoid Iterative Calculations**: Minimize the use of DAX functions like SUMX, FILTER, or EARLIER, as they can slow down performance with large datasets.
  - **Pre-calculate Complex Measures**: If a measure is complex, consider calculating it at the source (SQL or data warehouse) instead of in Power BI.
- **Improve Query Performance:**
  - **Use Query Folding**: When using Power Query, ensure that query folding is happening, meaning transformations are pushed down to the data source for execution rather than being done in Power BI.
  - **Filter Early**: Apply filters in Power Query or during data import to reduce the amount of data brought into the model.
  - **Avoid Complex Joins**: Limit complex joins in Power Query; try to keep the transformations simple or push them to the data source.
- **Indexing in Data Sources:**
  - **Optimize Indexing**: Ensure the data source has proper indexing to speed up query performance. This is particularly important when using DirectQuery.
- **Use Incremental Data Refresh:**
  - **Incremental Refresh**: For large datasets, use incremental refresh to only refresh a portion of the data rather than refreshing the entire dataset each time.
- **Optimize Power BI Service Performance:**
  - **Dataflow Optimization**: If using Power BI Dataflows, optimize transformations and schedules to minimize unnecessary data refreshes.
  - **Scheduled Refresh Settings:** Set refresh schedules to run during off-peak hours and avoid excessive concurrent refreshes.
- **Optimize Power BI Service:**
  - **Use Premium Capacity:** Power BI Premium provides higher performance with dedicated resources. If your reports require high performance, consider using Premium capacity.
- **Reduce Visual Complexity:**

- **Limit Visuals on a Page:** Avoid putting too many visuals on a single report page. Each visual adds a query, and loading too many visuals simultaneously can slow down performance.
- **Use Simpler Visuals:** Complex visuals with many data points (e.g., large maps or scatter plots) can slow down performance. Opt for simpler visuals when possible.

13. Power BI Best Practices ?

- **Data Modeling:**
  - **Use Star Schema**: Organize your data model in a star schema with fact tables and dimension tables for better performance and easier understanding.
  - **Use Relationships Wisely**: Always create relationships based on unique keys and avoid many-to-many relationships unless necessary.
  - **Avoid Bi-directional Filters**: Limit the use of bi-directional filters, as they can create ambiguity in calculations and affect performance.
- **Data Transformation:**
  - **Use Power Query Efficiently**: Clean and transform data using Power Query before loading it into the model. Avoid unnecessary steps to keep the query load time minimal.
  - **Optimize Queries**: Remove unnecessary columns, filter data early in the query process, and aggregate data where possible before loading it into the model.
  - **Use Variables in DAX**: For complex DAX calculations, use variables to store intermediate results for better readability and performance.
- **Performance Optimization:**
  - **Reduce Model Size**: Limit the size of your dataset by excluding unnecessary columns or tables and by summarizing data where possible.
  - **Use Aggregations**: Aggregate data at a higher level to reduce the volume of data processed in reports.
  - **Enable Query Folding**: Let Power Query push transformations to the data source to improve performance (especially with large datasets).
  - **Monitor Performance**: Use Performance Analyzer to monitor your report's performance and identify slow visuals or queries.
- **Visualization Best Practices:**
  - **Use Appropriate Visuals**: Choose visuals that effectively convey the story and avoid overcomplicating the report with unnecessary charts.
  - **Limit the Use of Highly Detailed Tables**: Instead of showing raw data in tables, summarize it using aggregates or metrics to enhance readability.
  - **Focus on User Experience**: Make the report interactive, with slicers and filters that allow users to explore the data, while keeping the visuals clean and simple.
  - **Consistency in Design**: Use consistent colors, fonts, and layouts for a professional and cohesive look.
- **DAX Optimization:**
  - **Avoid Complex Row-Level Calculations**: Where possible, avoid using row-level DAX calculations in large datasets, as they can negatively impact performance.
  - **Use Efficient DAX Functions**: Functions like SUMX, FILTER, and CALCULATE are powerful but can be expensive. Use them wisely to avoid performance bottlenecks.
  - **Understand Context**: Ensure you are clear about row, filter, and query context when writing DAX formulas to avoid unexpected results.
- **Collaboration and Sharing:**
  - **Use Workspaces for Collaboration**: Organize reports and datasets into workspaces for better team collaboration and management.
  - **Set Up Row-Level Security (RLS)**: Implement RLS for sensitive data to ensure that users see only the data relevant to them.

    - **Regular Data Refreshes**: Set up scheduled data refreshes to ensure the report always reflects the latest data.
- **Documentation and Version Control:**
    - **Document the Model and DAX Calculations**: Provide clear documentation on data models, relationships, and custom calculations, making it easier for others to understand.
    - **Version Control**: Keep track of changes in your reports and datasets to avoid overwriting critical work.

14. What is Power BI Dataflow in Power BI ?

A **Power BI Dataflow** is a collection of ETL (Extract, Transform, Load) processes created and managed within the Power BI service. It is designed to centralize data preparation by allowing users to ingest, clean, transform, and store data in a reusable manner. Dataflows are built on **Azure Data Lake** and can store data in the Common Data Model (CDM) format.

Key Features of Power BI Dataflow:

- **Centralized Data Preparation:** Allows multiple users to reuse the same data transformations across different reports and dashboards.
- **No-Code Data Transformation:** Users can create dataflows using Power Query, a visual, no-code interface.
- **Reusable Data:** Dataflows enable storing transformed data for reuse across multiple datasets and reports.
- **Integration with Azure Data Lake:** Power BI Dataflows can store data in an Azure Data Lake, making it accessible for other services.
- **Scheduled Refresh:** Dataflows can refresh automatically at scheduled intervals to keep data up-to-date.
- **Scalability:** Designed to handle large datasets and complex transformations, suitable for enterprise-level reporting.

How to Create a Power BI Dataflow:

- **Access Power BI Service:** Navigate to the Power BI service (https://app.powerbi.com).
- **Create a Dataflow:** Go to a **workspace**, click on New, and select **Dataflow**.
- **Define Entities:** Choose to add new tables or link to external data sources like SQL Server, Azure Blob Storage, Excel files, etc.
- **Transform Data:** Use the Power Query editor to clean, reshape, and transform the data.
- **Save and Load:** Save the dataflow, and Power BI will store the data in its storage or Azure Data Lake.
- **Schedule Refresh:** Set up a refresh schedule to keep the dataflow updated.

15. How do you handle many-to-many relationships in Power BI ?

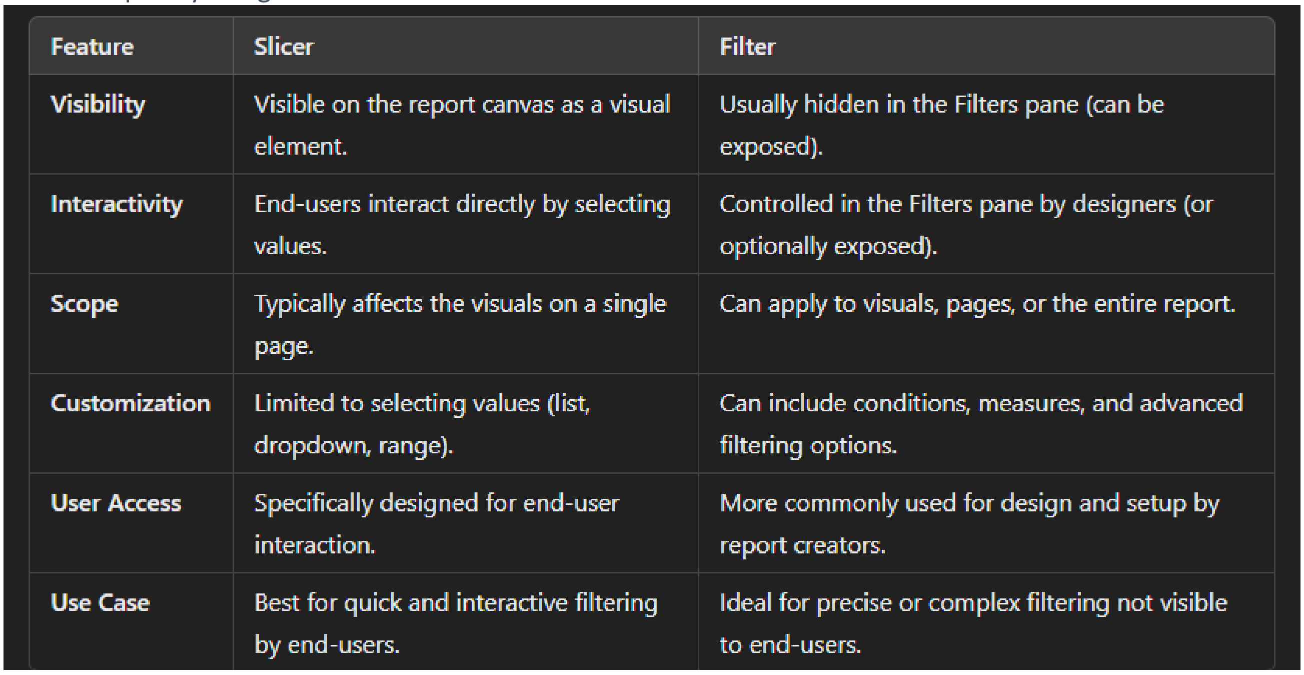
- **Use a Bridge Table:**
    - **When to Use:** The most common way to handle many-to-many relationships, especially if there is no direct key that uniquely identifies the relationship between two tables.
    - **Steps:** Create a bridge table that contains unique combinations of the keys from the related tables. Establish one-to-many relationships from the bridge table to each of the original tables. Use the bridge table to filter data across the related tables.

16. What is the difference between a slicer and a filter in Power BI ?

In Power BI, **slicers** and **filters** are used to refine and control the data displayed in reports, but they serve slightly different purposes and offer different functionalities.

- **Slicer:** A slicer is a visual element added to a Power BI report that allows users to interactively filter data by selecting specific values from a list, dropdown, or range.

- **Filter:** A filter is a feature in Power BI used to restrict the data that is shown in visuals. Filters can be set at various levels (visual, page, or report) and are not typically visible to end-users unless explicitly designed to be.

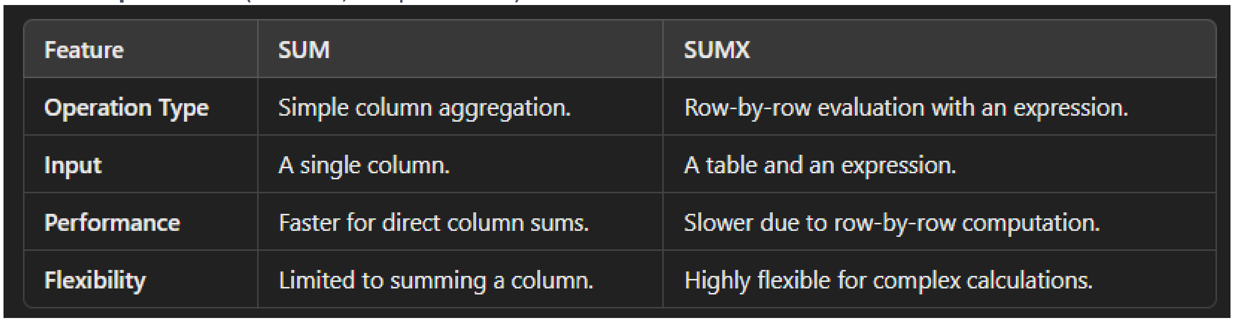


Feature	Slicer	Filter
Visibility	Visible on the report canvas as a visual element.	Usually hidden in the Filters pane (can be exposed).
Interactivity	End-users interact directly by selecting values.	Controlled in the Filters pane by designers (or optionally exposed).
Scope	Typically affects the visuals on a single page.	Can apply to visuals, pages, or the entire report.
Customization	Limited to selecting values (list, dropdown, range).	Can include conditions, measures, and advanced filtering options.
User Access	Specifically designed for end-user interaction.	More commonly used for design and setup by report creators.
Use Case	Best for quick and interactive filtering by end-users.	Ideal for precise or complex filtering not visible to end-users.

17. How Explain the difference between SUM and SUMX in DAX ?

- **SUM Function:** SUM is a simple aggregation function that calculates the total of a column containing numeric values.
  Example: SUM(<column>)

- **SUMX Function:** SUMX is an iterator function that evaluates an expression for each row of a table and then sums the results of those expressions.
  Example: SUMX(<table>, <expression>)



Feature	SUM	SUMX
Operation Type	Simple column aggregation.	Row-by-row evaluation with an expression.
Input	A single column.	A table and an expression.
Performance	Faster for direct column sums.	Slower due to row-by-row computation.
Flexibility	Limited to summing a column.	Highly flexible for complex calculations.

18. What is a Semantic Model ?

A **Semantic Model** in Power BI is a structured layer of metadata that provides business-friendly definitions for complex data. It acts as an abstraction layer between raw data and end-users, enabling them to interact with data in a simplified, intuitive, and meaningful way. The semantic model includes tables, columns, measures, hierarchies, relationships, and other metadata that allow users to explore and analyze data without needing technical expertise in database querying.

- **Components of a Semantic Model in Power BI:**
  - **Tables and Columns:** Represent the raw data, often imported or connected from various data sources. Includes calculated columns derived using DAX formulas.

    - **Relationships:** Define how tables are connected (e.g., one-to-many, many-to-one relationships). Enable navigation and aggregation of data across related tables.
    - **Measures:** Aggregations or calculations defined using DAX (e.g., Total Sales, Profit Margin). Provide reusable business metrics for analysis.
    - **Hierarchies:** Logical arrangements of data to allow drill-down capabilities (e.g., Year → Quarter → Month → Day).
    - **Calculated Tables:** New tables derived from existing ones using DAX to support specific analysis needs.
    - **Metadata Layer:** Includes descriptive information like table names, column names, and formatting to make the data model user-friendly.
    - **Row-Level Security (RLS):** Restricts access to data based on user roles.

- **Optimize Data Model:**
    - **Star Schema:** Organize your data model into a star schema (fact tables and dimension tables) for better performance. This reduces complexity and improves query performance.
    - **Avoid Bi-Directional Relationships:** Use single-directional relationships wherever possible. Bi-directional relationships can add overhead and slow down performance.
    - **Data Types**: Choose the appropriate data types for your columns (e.g., integers instead of decimals, and avoid using text for numerical values).
    - **Disable Auto Date/Time**: Power BI automatically creates hidden date tables for date fields, which can increase file size. Disable this in "Options" to avoid unnecessary tables.
- **Optimize DAX Measures and Calculations:**
    - **Use Variables in DAX**: Store intermediate results using variables to avoid recalculating the same values multiple times in a measure.
    - **Avoid Iterative Calculations**: Minimize the use of DAX functions like SUMX, FILTER, or EARLIER, as they can slow down performance with large datasets.
    - **Pre-calculate Complex Measures**: If a measure is complex, consider calculating it at the source (SQL or data warehouse) instead of in Power BI.

A marketing manager uses a Power BI dashboard to analyze sales trends across regions. The manager interacts with slicers for regions, time hierarchies for trends, and measures like "Total Sales" and "Profit Margin" to make decisions, all without worrying about the complexity of the underlying data sources. By leveraging the **semantic model**, the manager can focus on insights, not data preparation or technical queries.

19. What is the difference between VALUES and DISTINCT in DAX ?

- **VALUES Function:** The VALUES function returns a single column table of unique values from a column or a table.
  Example: VALUES(<column>) | VALUES(<table>)
  Unique Regions (Values) = COUNTROWS(VALUES(Sales[Region]))

- **DISTINCT Function:** The DISTINCT function returns a single-column table of unique values, ignoring any filter context.
  Example: DISTINCT(<column>)
  Unique Regions (Distinct) = COUNTROWS(DISTINCT(Sales[Region]))

Feature	VALUES	DISTINCT
Filter Context	Respects the current filter context.	Ignores filter context and returns all distinct values.
Blank Row Handling	Includes a blank row for missing or unrelated values.	Does not include a blank row for unmatched relationships.

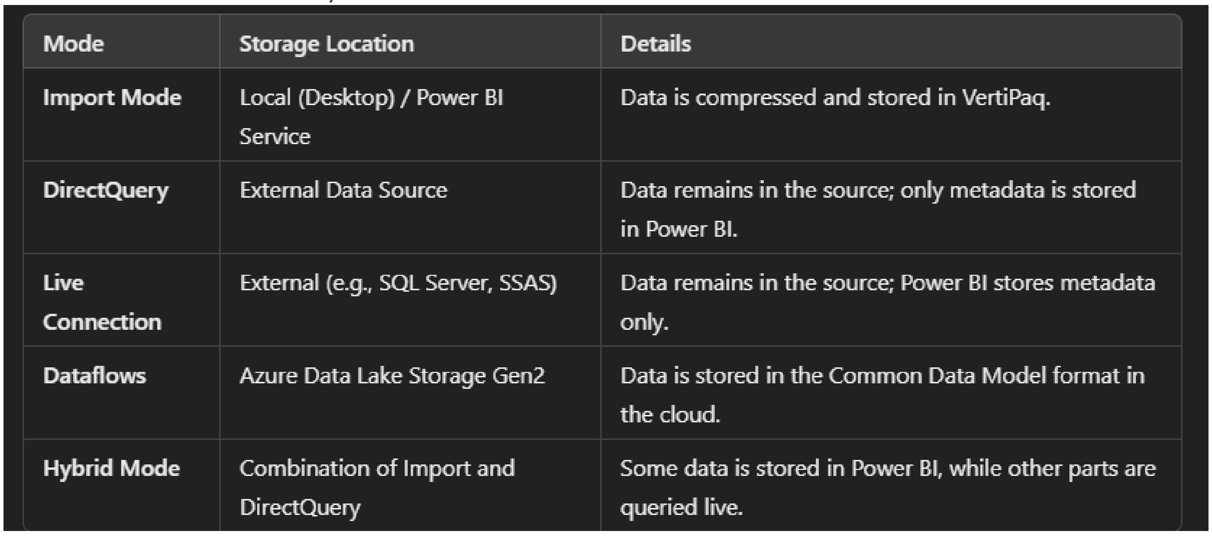
Performance	Slightly slower due to filter context dependency.	Faster as it doesn't consider filters.
Primary Use Case	For context-aware unique values.	For absolute unique values in a column.

20. Where is data stored in Power BI ?

- **In-Memory Storage (Power BI Desktop):** When you import data into Power BI Desktop, it is stored in a highly compressed, in-memory columnar database powered by the VertiPaq engine. This approach allows for fast querying and efficient storage. Data is stored in the .pbix file on your local machine. The VertiPaq engine compresses the data, making it highly optimized for performance. Data is fully loaded into memory for analysis, enabling quick calculations and visuals.

- **Power BI Service (Cloud Storage):** When you publish a Power BI report or dataset to the Power BI Service, the data can be stored in the cloud within Microsoft's Azure infrastructure.

  Storage Options in Power BI Service:
  - **Import Mode:** Data is imported and stored in the Power BI Service, using the same VertiPaq compression. Refreshing the dataset will pull updated data from the original data source.
  - **DirectQuery/Live Connection:** Data is not stored in Power BI. Queries are executed directly on the data source in real time. Only metadata (model structure, measures, relationships) is stored in Power BI.
  - **Hybrid Storage Mode:** Combines Import and DirectQuery, storing some data in Power BI while querying other parts in real time.
- **SQL-based Storage (Analysis Services):** If you're using a live connection to an **Azure Analysis Services** model or an **on-premises SQL Server Analysis Services** (SSAS) model, the data remains in the external database. Power BI only stores metadata.

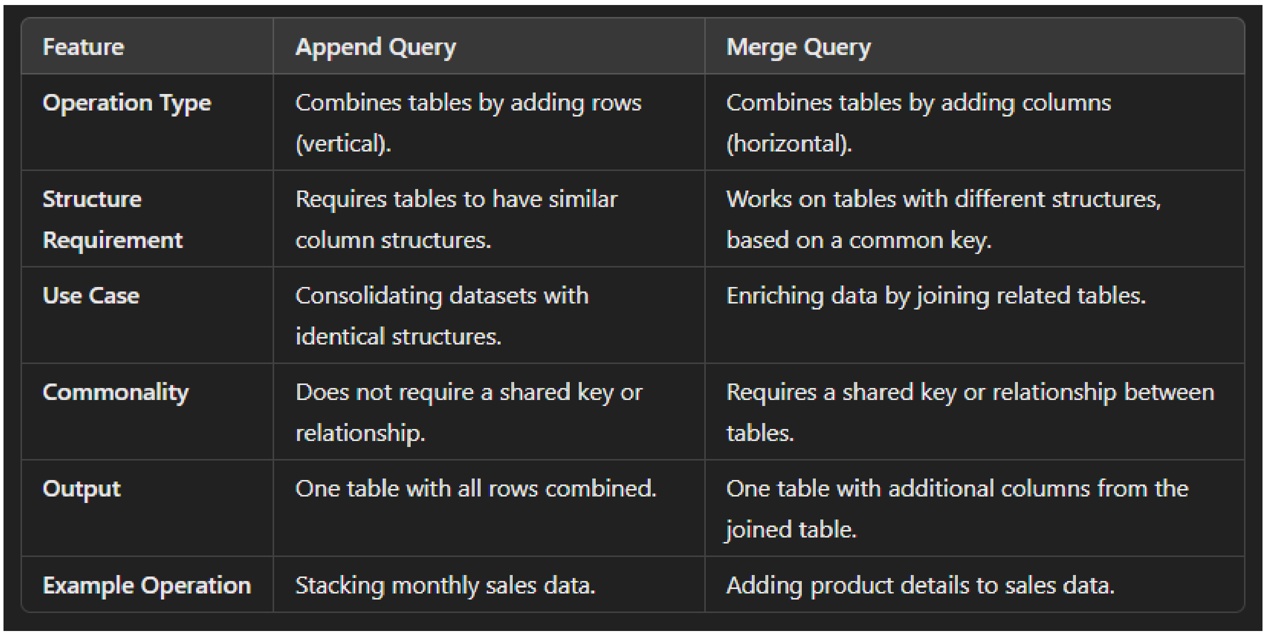


Mode	Storage Location	Details
Import Mode	Local (Desktop) / Power BI Service	Data is compressed and stored in VertiPaq.
DirectQuery	External Data Source	Data remains in the source; only metadata is stored in Power BI.
Live Connection	External (e.g., SQL Server, SSAS)	Data remains in the source; Power BI stores metadata only.
Dataflows	Azure Data Lake Storage Gen2	Data is stored in the Common Data Model format in the cloud.
Hybrid Mode	Combination of Import and DirectQuery	Some data is stored in Power BI, while other parts are queried live.

21. What is the difference between 'Append Query' and 'Merge Query' in Power BI ?

In Power BI, **Append Query** and **Merge Query** are two powerful operations used to manipulate and combine datasets.

- **Append Query:** The Append Query operation is used to combine two or more tables by stacking their rows on top of each other.
- **Merge Query:** The Merge Query operation is used to combine two tables by joining their rows based on a common column or key. It works horizontally by adding new columns to the table.

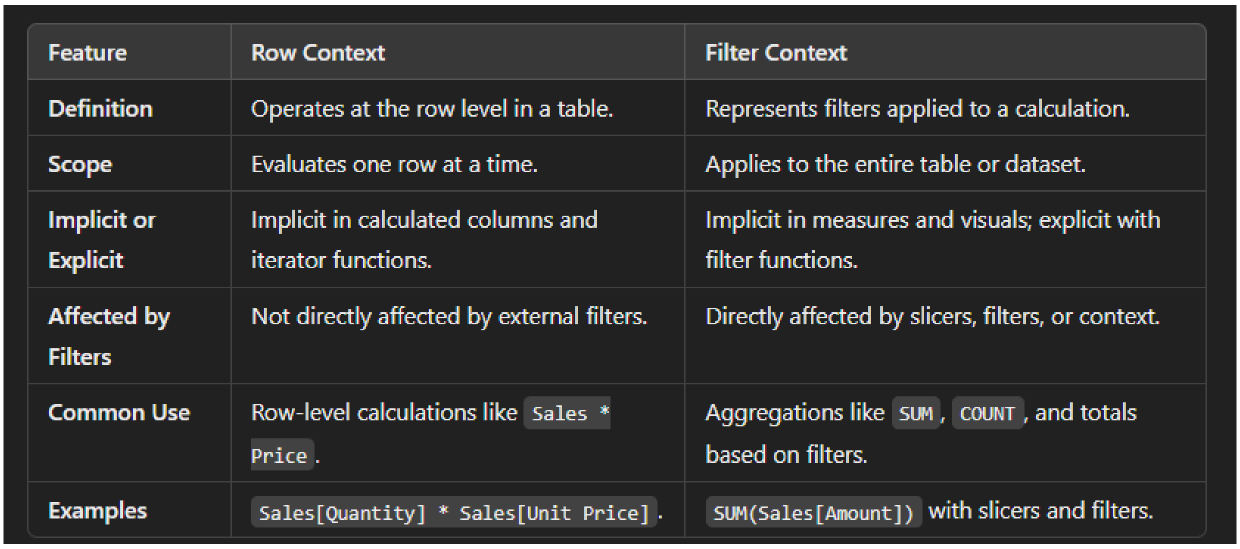


Feature	Append Query	Merge Query
Operation Type	Combines tables by adding rows (vertical).	Combines tables by adding columns (horizontal).
Structure Requirement	Requires tables to have similar column structures.	Works on tables with different structures, based on a common key.
Use Case	Consolidating datasets with identical structures.	Enriching data by joining related tables.
Commonality	Does not require a shared key or relationship.	Requires a shared key or relationship between tables.
Output	One table with all rows combined.	One table with additional columns from the joined table.
Example Operation	Stacking monthly sales data.	Adding product details to sales data.

22. What is the difference between 'Row Context' and 'Filter Context' in Power BI ?

In Power BI's **Data Analysis Expressions (DAX)**, **Row Context** and **Filter Context** are two critical concepts that define how calculations are evaluated.

- **Row Context:** Row Context refers to the context of a single row within a table. It is used when calculations or expressions are evaluated for each row individually. Row-level calculations, such as deriving a value for each record in a dataset.
- **Filter Context:** Filter Context refers to the filters applied to a calculation, either explicitly (via slicers, filters, or page filters) or implicitly (via relationships or context propagation). Aggregations like sums, averages, or counts under specific filters or conditions. Dynamic calculations affected by user interactions in visuals.



Feature	Row Context	Filter Context
Definition	Operates at the row level in a table.	Represents filters applied to a calculation.
Scope	Evaluates one row at a time.	Applies to the entire table or dataset.
Implicit or Explicit	Implicit in calculated columns and iterator functions.	Implicit in measures and visuals; explicit with filter functions.
Affected by Filters	Not directly affected by external filters.	Directly affected by slicers, filters, or context.
Common Use	Row-level calculations like `Sales * Price`.	Aggregations like `SUM`, `COUNT`, and totals based on filters.
Examples	`Sales[Quantity] * Sales[Unit Price]`.	`SUM(Sales[Amount])` with slicers and filters.

23. What is the KPIs in Power BI ?

A **Key Performance Indicator (KPI)** is a measurable value that demonstrates how effectively an individual, team, or organization is achieving a business objective. In Power BI, KPIs are used to visually track and measure the performance of a specific metric against a defined target. KPIs in Power BI enable businesses to measure and improve performance effectively, providing actionable insights that drive decision-making and strategy execution.

Components of a KPI in Power BI:

- **Indicator (Value):** The actual metric or value that you want to measure.
  Example: Sales revenue, profit margin, or customer satisfaction score.
- **Target (Goal):** The benchmark or target value that the metric is compared against.
  Example: A sales target of $1,000,000 or a profit margin goal of 20%
- **Status (Indicator Symbol/Color):** A visual representation (e.g., color-coded icons) that shows whether the metric is on track, above, or below the target.

Benefits of Using KPIs in Power BI:

- **Quick Insights:** Easily track performance at a glance without complex reports.
- **Customizable:** Define and format KPIs based on your specific business needs.
- **Trend Analysis:** Use the trend axis to analyze how metrics change over time.
- **Actionable Insights:** Visual feedback allows teams to act quickly to improve performance.
- **Integration:** Combine with other visuals and data models to create comprehensive dashboards.

Best Practices for KPIs in Power BI:

- **Focus on Critical Metrics:** Select KPIs that align with your business objectives.
- **Set Realistic Targets:** Ensure the target values are achievable and meaningful.
- **Use Effective Visuals:** Ensure the color coding and icons are intuitive and easy to understand.
- **Test Filters and Contexts:** Verify that KPIs respond correctly to slicers and filters in the report.

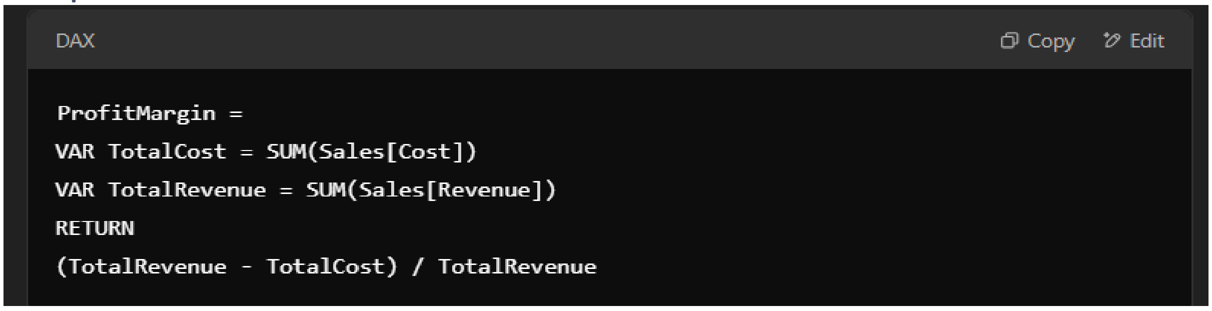
24. What are the fundamental concepts of DAX ?

Data Analysis Expressions (DAX) is a powerful formula language used in Microsoft Power BI, Excel, and Analysis Services to perform data calculations and analysis. Here are the **fundamental concepts of DAX**:

- **Calculated Columns:** A calculated column is an additional column created in a table using DAX formulas. The values are computed row by row and stored in the model.
  Example: TotalCost = Sales[UnitPrice] * Sales[Quantity]
- **Measures:** Measures are calculations used in reports and visuals. They are dynamic and calculated at the time of query execution, based on filter context.
  Example: TotalSales = SUM(Sales[Amount])
- **Row Context:** Refers to the current row being evaluated in a calculated column or an iteration function like SUMX or FILTER.
  Example: In a calculated column, Sales[TotalCost] = Sales[UnitPrice] * Sales[Quantity] operates in row context.
- **Filter Context:** The set of filters applied to data before the DAX formula is calculated. Filters come from slicers, filters, or rows/columns in visuals.
  Example: If a slicer filters a report to show only "2023" data, measures respect that filter.
- **Evaluation Context:** Combination of row context and filter context. Determines how DAX expressions are evaluated based on the applied context.
- **Aggregation Functions:** Functions like SUM, AVERAGE, MIN, MAX, and COUNT are used to perform aggregations.
  Example: AverageSales = AVERAGE(Sales[Amount])
- **Iterators:** Iterators are functions that perform row-by-row operations, like SUMX, AVERAGEX, and FILTER.
  Example: TotalProfit = SUMX(Sales, Sales[Profit] * Sales[Quantity])
- **Time Intelligence:** Specialized DAX functions for analyzing time-related data, such as year-to-date (YTD), month-to-date (MTD), and previous period calculations.

Example: SalesYTD = TOTALYTD(SUM(Sales[Amount]), Calendar[Date])

- **Relationships:** DAX leverages relationships between tables in the data model to retrieve and calculate values. Functions like RELATED and RELATEDTABLE are used to work across related tables.
  Example: ProductCategory = RELATED(Product[Category])
- **Calculated Tables:** Create new tables in the model using DAX expressions.
  Example: TopProducts = TOPN(10, Sales, Sales[Amount], DESC)
- **Variables:** Variables store intermediate results to improve performance and readability.
  Example:



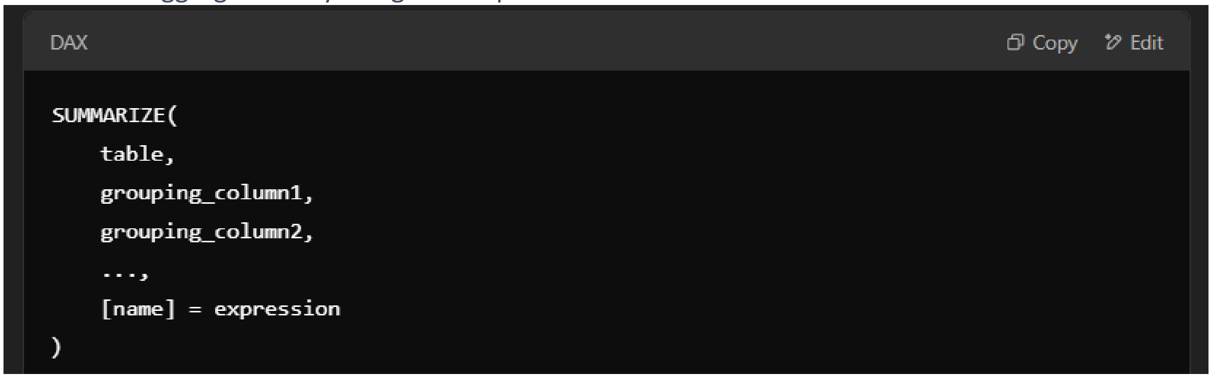
```
ProfitMargin =
VAR TotalCost = SUM(Sales[Cost])
VAR TotalRevenue = SUM(Sales[Revenue])
RETURN
(TotalRevenue - TotalCost) / TotalRevenue
```

- **Error Handling:** Use functions like IFERROR or ISBLANK to handle errors in DAX expressions.
  Example: SafeDivision = DIVIDE(Sales[Revenue], Sales[Quantity], 0)

25. Difference between 'Summerize' vs 'Summerized' in Power BI ?

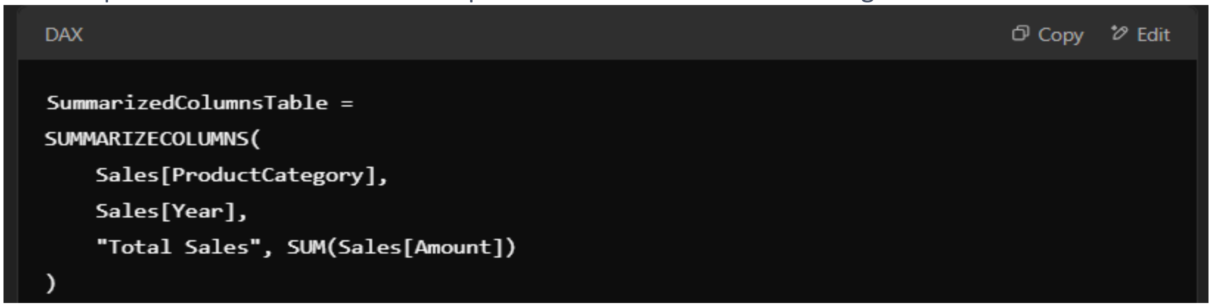
In Power BI, both **SUMMARIZE** and **SUMMARIZECOLUMNS** are DAX functions used for creating summaries of data, but they differ in terms of functionality, use cases, and performance. Here's a detailed comparison:

- **SUMMARIZE Function:** SUMMARIZE is used to group data by specified columns and optionally apply aggregations to those groups. Allows you to create a summary table by grouping data based on specific columns. Can include calculated columns within the summary. Allows you to add custom aggregations by using DAX expressions.

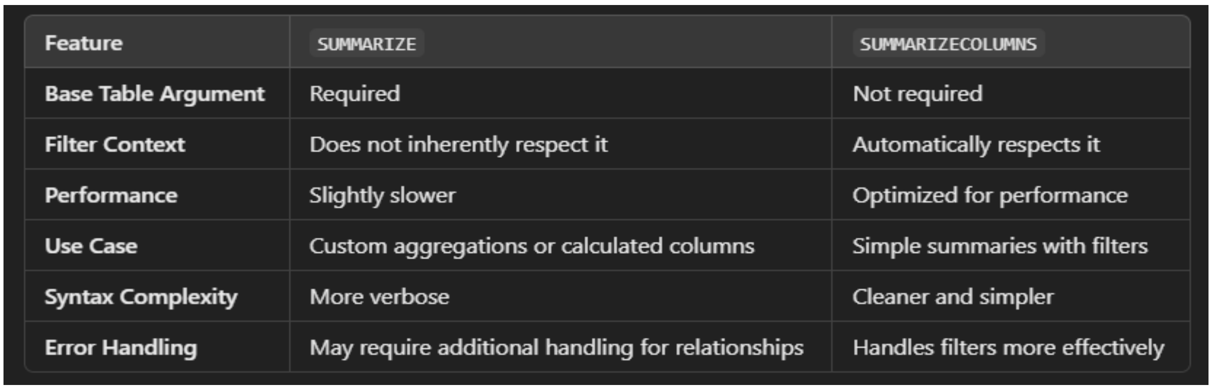


```
SUMMARIZE(
    table,
    grouping_column1,
    grouping_column2,
    ...,
    [name] = expression
)
```

- **SUMMARIZECOLUMNS Function:** SUMMARIZECOLUMNS is an optimized version of SUMMARIZE, designed for creating summaries while leveraging filter context effectively. Automatically applies the existing filter context in the report. More efficient and performant, especially when used in complex data models. Does not require the base table as the first argument.



```
SummarizedColumnsTable =
SUMMARIZECOLUMNS(
    Sales[ProductCategory],
    Sales[Year],
    "Total Sales", SUM(Sales[Amount])
)
```



Feature	SUMMARIZE	SUMMARIZECOLUMNS
Base Table Argument	Required	Not required
Filter Context	Does not inherently respect it	Automatically respects it
Performance	Slightly slower	Optimized for performance
Use Case	Custom aggregations or calculated columns	Simple summaries with filters
Syntax Complexity	More verbose	Cleaner and simpler
Error Handling	May require additional handling for relationships	Handles filters more effectively

Prefer SUMMARIZECOLUMNS whenever possible due to its performance advantages and simpler syntax. Use SUMMARIZE only for specific scenarios where additional flexibility is required.

26. What is gateways in Power BI ?

A **gateway** in Power BI is a bridge that enables secure communication between Power BI Service (cloud) and on-premises data sources. It allows you to connect to data sources located within your private network (on-premises) without exposing them to the internet, enabling **data refresh**, **live queries**, and **secure data transfers**.

27. What is Book marks in Power BI ?

Bookmarks in Power BI are a feature that allows users to save and restore specific states of a report page. They capture the configuration of a report, including filters, slicers, visuals, and even drill-through settings, making it easier to navigate and present insights.

Key Features of Bookmarks:

- **Capture Report States:** Bookmarks save the exact state of a report, including applied filters, slicer selections, and visual positions.
- **Switch Between Views:** Bookmarks enable users to toggle between different views of the same report, making it ideal for storytelling and presentations.
- **Interactivity:** Bookmarks can work with buttons, allowing users to create a guided navigation experience.
- **Dynamic Reporting:** Use bookmarks to create interactive dashboards by allowing users to switch between different metrics, visuals, or themes.

Use Cases of Bookmarks:

- **Storytelling:** Create a sequence of bookmarks to guide stakeholders through a data story during presentations.
- **Scenario Analysis:** Save different scenarios (e.g., "Best Case," "Worst Case") and switch between them with bookmarks.
- **Dynamic Filtering:** Set up bookmarks for commonly used filter combinations to quickly switch views.
- **Toggle Visual:** Use bookmarks to toggle between charts, such as switching between a bar chart and a line chart.
- **Drill-Through Navigation:** Combine bookmarks with drill-through features to enhance navigation and detail exploration.

Limitations of Bookmarks:

- **Static Capture:** Bookmarks capture a static state, so they need to be updated if the report layout changes.
- **No Dynamic Updates:** Changes in data or visuals not explicitly included in the bookmark won’t be updated.
- **Maintenance:** Managing many bookmarks can become challenging in complex reports.

Tips for Effective Bookmark Management:

- **Use Clear Names:** Name bookmarks descriptively, such as "Sales by Region" or "Q4 Analysis."
- **Group Bookmarks:** Organize bookmarks into groups for easier navigation in the Bookmarks Pane.
- **Combine with Buttons:** Use buttons for user-friendly interaction and navigation.
- **Preview Before Saving:** Always test the state before saving bookmark to ensure it captures the desired view.

28. Which types of reports developed in Power BI ?

In Power BI, various types of reports are developed to meet specific business needs and analytical objectives. The types of reports vary based on the audience, purpose, and data being analysed.

- **Operational Reports:** Focus on day-to-day business operations, providing detailed and real-time data. High granularity. Includes tables and detailed transactional data. Refreshes frequently to ensure up-to-date insights.
  Eg. Inventory reports, Daily sales reports, Customer support performance.

- **Analytical Reports:** Provide deeper insights into data through trends, patterns, and predictions. Focus on data exploration and visualization. Use advanced features like DAX measures, forecasting, and clustering. Help decision-makers understand the “why” behind the data.
  Eg. Profitability analysis, Trend analysis for sales or revenue, Customer segmentation.

- **Dashboard Reports:** Provide a high-level overview of key metrics and KPIs on a single page. Summary of critical business metrics. Combines data from multiple reports. Interactive, with drill-through and drill-down capabilities.
  Eg. Executive dashboards with KPIs for revenue, expenses, and profit, Employee performance dashboards, Marketing campaign performance dashboard.

- **Financial Reports:** Focus on financial data, helping businesses monitor and manage financial performance. Includes profit and loss statements, balance sheets, and cash flow reports. Tracks financial KPIs like revenue, expenses, gross margin, and ROI.
  Eg. Budget vs. actuals report, Financial statement analysis, Expense tracking by department.

- **Sales and Marketing Reports:** Monitor sales performance and marketing effectiveness. Tracks metrics like revenue, conversions, pipeline, and campaign ROI. Includes regional or demographic breakdowns.
  Eg. Monthly sales performance report, Lead conversion analysis, Marketing campaign ROI report.

- **Strategic Reports:** Support long-term business planning and strategic decision-making. High-level insights and actionable recommendations. Incorporates forecasting and scenario analysis.
  Eg. Market trend analysis, Growth strategy reports, Competitor analysis.

- **Performance Reports:** Measure and monitor the performance of teams, departments, or projects. Includes KPIs and benchmarks. Provides comparisons with targets or past performance.
  Eg. Employee performance scorecards, Project status and milestone tracking, Departmental efficiency reports.

- **Real-Time Reports:** Provide live updates for time-sensitive decisions. Uses streaming datasets. Ideal for monitoring operational activities in real-time.
  Eg. Real-time stock level tracking, IoT device performance dashboards, Live customer interaction tracking.

- **Customer and Service Reports:** Understand customer behavior and improve service delivery.

- **Custom Reports:** Tailored to meet specific business needs or unique use cases.
- **Compliance and Audit Reports:** Ensure adherence to regulatory requirements and internal policies.

Best Practices for Power BI Reports:

- **Understand the Audience:** Tailor reports based on the end-users' needs (e.g., executives vs. analysts).
- **Optimize Performance:** Use optimized data models and efficient DAX queries.
- **Ensure Interactivity:** Leverage slicers, filters, and drill-through options for better engagement.
- **Focus on Key Metrics:** Highlight essential KPIs and metrics for better decision-making.
- **Maintain Visual Clarity:** Avoid overcrowding reports with too many visuals or excessive data.

29. Recently faced challenges in power Bi? and solution for that in details ?

1. Data Connectivity Issues: Connecting to large or complex data sources sometimes results in errors or slow performance. For instance, connecting to databases with millions of rows often causes timeouts or crashes.
Solution:

- **Optimize Data Loading:** Use query folding to offload the data transformation workload to the database rather than Power BI.
- **Direct Query Mode:** For large datasets, use Direct Query instead of Import mode to fetch only the required data on demand.
- **Incremental Refresh:** Configure incremental data refresh for large datasets to avoid reloading the entire dataset during updates.

2. Slow Report Performance: Reports with multiple visuals and large datasets often experience slow performance, leading to longer load times and poor user experience.
Solution:

- **Optimize DAX Measures:** Simplify complex DAX calculations and use measures instead of calculated columns when possible.
- **Reduce Visuals:** Limit the number of visuals on a single page and reduce the complexity of charts and tables.
- **Aggregate Data:** Aggregate data at the source or in Power BI to minimize the amount of raw data processed.
- **Enable Query Caching:** Use query caching to store frequently accessed data locally for faster retrieval.

3. Data Modeling Issues: Improper data modeling, such as creating unnecessary relationships or not using star schema, leads to inconsistencies and incorrect results.
Solution:

- **Follow Best Practices:** Use a star schema design by separating fact tables and dimension tables.
- **Avoid Many-to-Many Relationships:** Redesign the model to minimize the use of many-to-many relationships.
- **Use Relationship Filtering:** Configure bidirectional filtering only when absolutely necessary.

4. Visualizations Not Updating: Reports do not update properly after data refresh, causing discrepancies between the underlying data and visuals.
Solution:

- **Verify Data Refresh:** Check the dataset refresh history and ensure the refresh process is completing successfully.
- **Clear Cache:** Clear the browser and Power BI Desktop cache to avoid outdated visuals.
- **Refresh Individual Visuals:** Use the "Refresh" button for individual visuals to verify the data.

5. Publishing and Sharing Issues: Sharing Power BI reports with external users or within the organization sometimes fails due to licensing or configuration restrictions.
Solution:

- **Licensing:** Ensure that all users have the appropriate Power BI license (Pro or Premium).
- **Workspaces:** Publish reports in shared or premium workspaces accessible to the intended audience.
- **Row-Level Security (RLS):** Configure RLS to control data access for different user groups.

6. Handling Dynamic Data: Creating reports for dynamic datasets where columns and values change frequently leads to broken reports.
Solution:

- **Dynamic Measures:** Use dynamic DAX measures to handle changing data values.
- **Parameterization:** Implement parameters for columns or values that might change.
- **Data Transformation:** Regularly audit and transform incoming data to ensure schema consistency.

7. Error in DAX Calculations: Complex DAX formulas often result in errors or unexpected results.
Solution:

- **Debugging:** Use the DAX editor's error-checking features to debug issues.
- **Break Down Formulas:** Break down complex formulas into smaller, testable parts.
- **Use Variables:** Define variables in DAX to simplify calculations and improve readability.

8. Export Limitations: Exporting data or reports from Power BI to Excel or PDF is often limited to a certain number of rows or lacks formatting options.
Solution:

- **Paginated Reports:** Use Power BI paginated reports for better export functionality.
- **API Integration:** Leverage Power BI REST APIs to export larger datasets programmatically.
- **Custom Scripts:** Use custom scripts or third-party tools to bypass export limitations.

9. Integration with Other Tools: Integrating Power BI with other tools like SharePoint, Microsoft Teams, or external applications can be challenging due to compatibility or API limitations.
Solution:

- **Native Connectors:** Use Power BI's native connectors for seamless integration with Microsoft tools.
- **Custom API Integration:** For non-standard tools, build custom connectors using APIs.
- **Scheduled Syncs:** Automate data synchronization between tools to maintain consistency.

10. Version Control: Managing different versions of Power BI reports leads to confusion and errors.
Solution:

- **Version Naming:** Use a consistent naming convention for different versions of reports.
- **Source Control Tools:** Use tools like Git or SharePoint for version control and collaborative development.
- **Backups:** Regularly back up Power BI files (.pbix) to avoid data loss.

30. What is query folding in power BI ? How It's work in details ?

Query folding is a critical concept in Power BI that significantly enhances performance and efficiency during data preparation. It refers to the process where Power Query translates applied transformations into a single query statement, which is executed by the data source rather than by Power BI. Query folding occurs when Power Query generates a native query in the source system that performs data transformations as much as possible. This approach reduces the amount of data transferred to Power BI, improving processing speed and optimizing resource usage.

1. How Query Folding Works

1. **Connecting to Data Sources:** When a connection is established, Power Query attempts to push data transformations back to the source system.
   - Example: For SQL databases, Power Query generates SQL statements that include the applied transformations.
2. **Transformation Optimization:** Transformations such as filtering, sorting, joining, grouping, and aggregating are pushed to the data source if it supports them.
   - Power Query creates a single query that includes all transformations and sends it to the source.
3. **Query Execution:** The source system processes the query and returns the results to Power BI. This minimizes the volume of data being transferred and leverages the source system's computational power.

Benefits of Query Folding

1. **Performance Improvement:** Reduces data transfer time by processing transformations at the source.
2. **Efficiency:** Leverages the data source's processing capabilities, reducing the load on Power BI.
3. **Scalability:** Handles large datasets effectively by filtering and aggregating data before transfer.
4. **Simplified Maintenance**: Consolidates transformations into a single query, making it easier to manage and debug.

When Query Folding Occurs

- Query folding is most effective when connecting to relational databases (e.g., SQL Server, Oracle, MySQL).
- It also works with certain cloud-based sources like Azure SQL Database and SharePoint Online.
- Not all data sources support query folding; for flat files like Excel or CSV, transformations occur within Power BI.

For More Power BI Questions -> Link