Module 5) Creating Dashboard with Visualization Tool Assignment

1. What is Power BI and how does it differ from Excel?

Power BI is a **Business Intelligence (BI) and data visualization tool** developed by Microsoft.  
It allows you to **connect to multiple data sources, transform data, model relationships, and create interactive dashboards and reports** that can be shared across organizations via the Power BI Service (cloud).

* **Difference between Power BI and Excel**

|  |  |  |
| --- | --- | --- |
| Feature | Power BI | Excel |
| Purpose | Built for **data visualization, reporting, and BI** | Built for **data analysis, calculations, and tabular reporting** |
| Data Handling Capacity | Handles **large datasets (millions of rows)** efficiently using **Power Query + Vertipaq engine** | Slows down with **large datasets (over ~1M rows)** |
| Data Sources | Connects to **cloud, databases, APIs, big data sources** | Limited connectors, mostly flat files and some databases |
| Visualization | Interactive **dashboards, slicers, filters, drill-throughs** | Charts and pivot tables, but less interactive |
| Automation & Refresh | Can **schedule automatic data refresh** in Power BI Service | Manual refresh (unless using VBA/Power Query with some automation) |
| Collaboration | Share dashboards in **Power BI Service / Teams / Mobile** | Sharing through Excel files (email, SharePoint, OneDrive) |
| Security | Supports **Row-Level Security (RLS)** for restricting data access | No native RLS, only password protection or file-level security |
| Data Modeling | Strong **data modeling with relationships, DAX measures** | Limited relational modeling (mostly VLOOKUP, Power Pivot add-in) |
| Ease of Use | Easier for **end-users to explore data visually** | Requires more **manual setup and formulas** |

1. Explain the concept of data modeling in Power BI.

Ans) Data modeling in Power BI is the process of **organizing and structuring data from multiple sources** so that it can be easily analyzed and used for creating reports and dashboards. It acts as the **foundation of Power BI** because without a proper data model, visualizations will not provide accurate insights.

**Key Points about Data Modeling**

1. **Relationships Between Tables**
   * Data modeling connects different tables using **relationships** (similar to primary key–foreign key in databases).
   * Example: *Sales Table* (OrderID, CustomerID, Amount) can be linked with *Customer Table* (CustomerID, Name, City).
2. **Star and Snowflake Schema**
   * Power BI commonly uses a **Star Schema** where a central **Fact Table** (transactions, numeric data) is connected to multiple **Dimension Tables** (categories like Date, Product, Customer).
   * This makes reports faster, more accurate, and easier to maintain.
3. **Data Modeling Components**
   * **Fact Tables** → Contain numerical, quantitative data (e.g., sales amount, profit).
   * **Dimension Tables** → Contain descriptive attributes (e.g., product name, customer name).
   * **Measures (DAX)** → Calculated values like *Total Sales = SUM(Sales[Amount])* created using **Data Analysis Expressions (DAX)**.
   * **Calculated Columns** → Custom columns created for additional fields (e.g., Profit = Sales – Cost).
4. **Cardinality and Direction**
   * Relationships can be **One-to-Many, One-to-One, Many-to-Many**.
   * You can set **cross-filter direction** (single or both) to control how data flows between tables.
5. **Importance of Data Modeling**
   * Ensures **data accuracy** and avoids duplication.
   * Makes reports **faster and scalable**.
   * Helps in creating **complex KPIs** using DAX.
   * Allows **drill-down analysis** across multiple tables.

**Example**

Imagine you are analyzing **Shark Tank Investments** data:

* **Fact Table:** Deals (DealID, Amount Invested, Valuation)
* **Dimension Tables:**
  + Entrepreneurs (EntrepreneurID, Name, Industry)
  + Sharks (SharkID, Name, Gender)
  + Date (DateID, Month, Year)

By building relationships between these tables, you can easily create reports like:

* Total Investment by Industry
* Deals per Shark per Year
* Average Valuation by Sector

1. What are the different types of connections available in Power BI?

Ans) **Types of Connections in Power BI**

When you connect Power BI to a data source, there are mainly **three connection modes**:

**1. Import Mode**

* **Definition:** Data is imported (copied) from the source into Power BI’s internal memory (VertiPaq engine).
* **Characteristics:**
  + Data is stored in the **.pbix file**.
  + Reports are very **fast** because queries run on in-memory data.
  + Requires **manual or scheduled refresh** to update data.
* **Best For:** Small to medium datasets where performance and offline access are important.
* **Example:** Importing an Excel sheet or SQL database table into Power BI.

**2. DirectQuery Mode**

* **Definition:** Data is **not stored in Power BI**. Instead, Power BI sends queries to the source system in real time.
* **Characteristics:**
  + Always **up to date** since queries run directly on the source.
  + Performance depends on the source system and network speed.
  + Some **DAX functions and transformations** may not be supported.
* **Best For:** Very large datasets that cannot fit in memory, or when **real-time reporting** is required.
* **Example:** Connecting directly to a SQL Server or Azure SQL Database without importing.

**3. Live Connection**

* **Definition:** Similar to DirectQuery but mainly used for **Analysis Services (SSAS), Azure Analysis Services, and Power BI Datasets**.
* **Characteristics:**
  + No data is imported; queries are passed to the SSAS/Power BI dataset.
  + All data modeling is done in the source (not inside Power BI).
  + Supports centralized **enterprise-level data models**.
* **Best For:** Large organizations with **pre-built models** in SSAS or when using a shared Power BI dataset.
* **Example:** Connecting Power BI to a corporate SSAS Tabular Model or to a published dataset in Power BI Service.

**4. Composite Mode (Hybrid)**

* **Definition:** A combination of **Import + DirectQuery** in the same model.
* **Characteristics:**
  + Some tables can be imported for speed.
  + Some tables can remain in DirectQuery for **real-time data**.
  + Offers flexibility and balance between **performance and freshness**.
* **Best For:** Scenarios where you want real-time reporting for some data (e.g., sales transactions) while keeping historical data imported.
* **Example:** Importing historical sales data but using DirectQuery for today’s transactions.

1. How do you handle data transformation in Power BI?

**Handling Data Transformation in Power BI**

Data transformation in Power BI is the process of **cleaning, shaping, and preparing raw data** so it can be used for analysis and reporting.  
This is mainly done using **Power Query Editor** (based on M language).

**Steps to Handle Data Transformation**

**1. Connect to Data Source**

* Import or connect to data from **Excel, SQL, Web, APIs, etc.**
* Data first loads into the **Power Query Editor**.

**2. Clean the Data**

* **Remove errors or null values**  
  (e.g., delete blank rows/columns).
* **Replace values** (e.g., “N/A” → null).
* **Remove duplicates** to ensure accuracy.

**3. Shape the Data**

* **Rename columns** for better readability.
* **Split or merge columns** (e.g., Full Name → First Name, Last Name).
* **Change data types** (text, number, date).
* **Filter rows** to include only relevant data.

**4. Transform Data**

* **Unpivot columns** to normalize tables.
* **Pivot data** to create summary tables.
* **Group data** (e.g., sales by region).
* **Add calculated columns** using M or later with **DAX**.

**5. Combine Data**

* **Append Queries** → Stack tables with same structure (e.g., sales from different regions).
* **Merge Queries** → Join tables on a key (e.g., Sales table + Customer table).

**6. Load into Data Model**

* After transformations, load the clean dataset into **Power BI Data Model**.
* Use relationships, DAX, and measures for further analysis.

**Example**

Suppose you have **Sales Data** with missing customer names and wrong date formats:

1. Remove null values from Customer column.
2. Change Date column type to *Date*.
3. Split “Full Address” into *City* and *State*.
4. Merge Sales Table with Customer Table using CustomerID.
5. Load into Power BI model for reporting.

5) What is DAX (Data Analysis Expressions) and why is it important in Power BI?

**DAX (Data Analysis Expressions)** is a **formula and expression language** used in Power BI, Power Pivot, and Analysis Services.  
It is designed specifically for **data modeling, calculations, and analysis** on tabular data.

DAX is similar to Excel formulas but more powerful, as it works with **relational data models and aggregations**.

**Why is DAX Important in Power BI?**

1. **Performs Advanced Calculations**
   * DAX allows creation of **Measures** (like Total Sales, Profit Margin) and **Calculated Columns** (like Age = TODAY() – DOB).
   * Example:
   * Total Sales = SUM(Sales[Amount])
2. **Works Across Multiple Tables**
   * DAX can use **relationships** between tables to perform calculations without needing complex joins.
   * Example: *Total Sales by Customer Region* even if data is split into Sales and Customer tables.
3. **Enables Time Intelligence**
   * DAX has built-in functions for analyzing data over time.
   * Example: Year-to-Date (YTD), Month-to-Date (MTD), Previous Year comparisons.
   * Example:
   * Sales YTD = TOTALYTD(SUM(Sales[Amount]), 'Date'[Date])
4. **Dynamic and Interactive Reports**
   * DAX makes visuals **respond to slicers, filters, and drill-downs** dynamically.
   * Example: A “Profit %” card updates automatically when a user selects a different year or region.
5. **Supports Business KPIs**
   * Helps build key performance indicators like **Revenue Growth %, Average Deal Size, Churn Rate**.

**Examples of DAX Functions**

* **Aggregate Functions** → SUM(), AVERAGE(), COUNTROWS()
* **Filter Functions** → CALCULATE(), FILTER()
* **Time Intelligence** → SAMEPERIODLASTYEAR(), DATESYTD()
* **Logical Functions** → IF(), SWITCH()

6)difference between calculated columns and measures in Power BI?

|  |  |  |
| --- | --- | --- |
| Aspect | Calculated Column | Measure |
| Definition | A new column created in a table, with values calculated **row by row** using DAX. | A calculation created using DAX that is **evaluated on the fly** when used in visuals. |
| Storage | Values are **stored** in the data model (increases file size). | Values are **not stored**, only calculated when needed (no extra storage). |
| Context | Works in **row context** (each row has a value). | Works in **filter context** (depends on filters, slicers, and visuals). |
| Performance | Can slow down model if dataset is large (because it stores results). | More efficient, as calculations are dynamic and lightweight. |
| Use Cases | Needed when:  – Creating new fields for filtering, grouping, or sorting  – Building relationships | Needed when:  – Creating KPIs (e.g., Total Sales, Profit %)  – Aggregations that respond to filters |
| Example | Profit = Sales[Revenue] – Sales[Cost] → Creates a new column “Profit” for each row. | Total Profit = SUM(Sales[Revenue]) – SUM(Sales[Cost]) → Gives total profit dynamically in a visual. |

7) How do you handle relationships between tables in Power BI?

Ans)

**Handling Relationships Between Tables in Power BI**

In Power BI, relationships define how **two or more tables are connected** so that data can be analyzed together.  
They are similar to **primary key – foreign key** relationships in databases.

**Steps to Handle Relationships**

**1. Identify Keys**

* Each relationship is created using a **common field** (column) between two tables.
* Example: *Sales[CustomerID]* connects to *Customer[CustomerID]*.

**2. Create Relationships**

* Go to **Model View** in Power BI.
* Drag and drop the common column between tables, or use **Manage Relationships** → *New*.

**3. Choose Cardinality (Type of Relationship)**

* **One-to-Many (1:\*)** → Most common (e.g., one Customer → many Sales).
* **One-to-One (1:1)** → Rare, when both tables have unique matching records.
* **Many-to-Many (*:*)** → Used when both tables have duplicates; often avoided for performance unless necessary.

**4. Set Cross-Filter Direction**

* **Single** → Filters flow from one table to another (common choice).
* **Both** → Filters flow both ways (use carefully, can affect performance).

**5. Ensure Relationship Integrity**

* Avoid duplicates in key columns.
* Check for null values.
* Use **Star Schema** (Fact + Dimension tables) for best performance.

**Example**

Imagine a Shark Tank dataset:

* **Deals Table (Fact):** DealID, Amount, SharkID, EntrepreneurID
* **Sharks Table (Dimension):** SharkID, SharkName, Gender
* **Entrepreneurs Table (Dimension):** EntrepreneurID, Name, Industry

**Relationships:**

* Deals[SharkID] → Sharks[SharkID] (One-to-Many)
* Deals[EntrepreneurID] → Entrepreneurs[EntrepreneurID] (One-to-Many)

This allows analysis like:

* Total Investment per Shark
* Deals Count by Industry

8) What is the purpose of a Power BI Gateway?

Ans)

**Purpose of a Power BI Gateway**

A **Power BI Gateway** is a **bridge (connector)** that enables secure data transfer between **on-premises data sources** (like SQL Server, Oracle, SAP, local Excel files) and **Power BI Service (cloud)**.

It ensures that reports and dashboards in the Power BI Service can access and refresh data stored in local (on-premises) systems.

**Key Purposes / Functions**

1. **Secure Data Transfer**
   * Provides a safe and encrypted way to move data from on-premises systems to the cloud without uploading raw databases.
2. **Scheduled Refresh**
   * Allows you to **schedule automatic refresh** of datasets so that dashboards always show the latest data.
3. **Live / Direct Query Connection**
   * Supports **real-time or near real-time queries** against on-premises databases without importing the data.
4. **Centralized Management**
   * Can be managed by IT teams to control access, security, and performance.
5. **Supports Multiple Microsoft Services**
   * Not only for Power BI — also works with **Power Apps, Power Automate, Azure Logic Apps** for connecting to on-premises data.

**Types of Gateways in Power BI**

1. **Personal Gateway**
   * Installed on a personal machine.
   * Meant for individuals, supports **import and scheduled refresh** only.
   * Cannot be shared with others.
2. **On-premises Data Gateway (Standard Mode)**
   * Installed on a server.
   * Supports **DirectQuery, Live Connection, and scheduled refresh**.
   * Can be used by multiple users and multiple data sources.
   * Best for **enterprise use**.

**Example**

Imagine your company stores **sales data in SQL Server on-premises**.

* Without a gateway → You would have to manually export and upload the data to Power BI.
* With a gateway → Power BI Service can connect securely to SQL Server and refresh dashboards automatically (daily/hourly).

9) How can you schedule data refresh in Power BI Service?

Ans)

**How to Schedule Data Refresh in Power BI Service**

Scheduling data refresh ensures that your **Power BI reports and dashboards always show the latest data** without manually re-uploading datasets.

**Steps to Schedule a Data Refresh**

**1. Publish Your Report to Power BI Service**

* First, create and save your Power BI report (.pbix) in **Power BI Desktop**.
* Publish it to your **Power BI Workspace (cloud)**.

**2. Configure Gateway (if needed)**

* If your data source is **on-premises (e.g., SQL Server, local Excel)**, you must install and configure a **Power BI Gateway**.
* Cloud data sources (e.g., Azure, SharePoint Online) usually don’t need a gateway.

**3. Open Dataset Settings**

* In **Power BI Service**, go to your **workspace**.
* Find the **dataset** connected to your report.
* Click on **More Options (…) → Settings**.

**4. Set Up Scheduled Refresh**

* In the **Settings** pane:
  + Go to **Dataset → Scheduled refresh**.
  + Turn **Keep data updated = ON**.
  + Set **Refresh frequency**: Daily or Weekly.
  + Set **Time**: Choose one or multiple times per day.
  + Enter credentials (if required) to authenticate the data source.

**5. Save and Monitor**

* Click **Apply** to save the schedule.
* You can monitor refresh history under **Refresh history** in dataset settings.
* If refresh fails, Power BI sends an **email notification**.

**Example**

Suppose you have a **Sales Dashboard** connected to SQL Server (on-premises):

* Install and configure a **Gateway**.
* Schedule refresh every morning at **7:00 AM**.
* Now, managers always see **updated sales figures** when they open the dashboard.

10) Explain the concept of row-level security in Power BI.

Ans)

**Row-Level Security (RLS) in Power BI**

**Row-Level Security (RLS)** is a feature in Power BI that restricts **what data a user can see** based on filters you define.  
Instead of giving everyone access to the full dataset, RLS ensures users only see the data that is **relevant to them**.

**How RLS Works**

* You create **roles** and define **DAX filter rules** on tables.
* When a user is assigned to a role, the filters are automatically applied every time they view the report.
* This happens **without changing the original dataset**.

**Types of RLS**

1. **Static RLS**
   * Fixed filters are applied to roles.
   * Example: A role "India\_Sales" is restricted with
   * [Country] = "India"
   * All users in this role will only see sales data for India.
2. **Dynamic RLS**
   * Filters are applied based on **logged-in user identity** (USERNAME() or USERPRINCIPALNAME()).
   * More flexible, as one role can serve multiple users.
   * Example:
   * [Region] = LOOKUPVALUE(UserTable[Region], UserTable[Email], USERPRINCIPALNAME())
   * Here, each user only sees their assigned region based on the mapping in *UserTable*.

**Steps to Implement RLS in Power BI**

1. **Define Roles in Power BI Desktop**
   * Go to **Modeling → Manage Roles → Create**.
   * Add DAX filter expressions (e.g., [Region] = "East").
2. **Test Roles in Power BI Desktop**
   * Use **View as Roles** to check what data different roles can see.
3. **Publish to Power BI Service**
   * After publishing, go to **Dataset → Security**.
   * Assign users or groups to roles.

**Example**

Imagine a sales dataset with regions: North, South, East, West.

* Create roles: North\_Manager, South\_Manager, etc.
* Apply filters like [Region] = "North".
* Assign each manager to their role in Power BI Service.  
  👉 Now, when a **North Manager** logs in, they see only **North region sales**, not the full dataset.

11) What is the Power BI Desktop and how does it differ from Power BI Service?

Ans)

**Power BI Desktop vs Power BI Service**

**1. Power BI Desktop**

* A **free Windows application** installed on your computer.
* Used mainly for:
  + **Connecting to data sources** (Excel, SQL, Web, APIs, etc.)
  + **Data transformation** (using Power Query)
  + **Data modeling** (relationships, DAX, measures)
  + **Creating reports and visualizations**
* Output is saved as a **.pbix file**.
* Mainly used by **data analysts/developers** to design reports before publishing.

**2. Power BI Service**

* A **cloud-based platform (app.powerbi.com)** hosted by Microsoft.
* Used mainly for:
  + **Publishing and sharing reports/dashboards** created in Power BI Desktop.
  + **Scheduled data refresh** (automatic updates).
  + **Collaboration** (multiple users, comments, apps, workspaces).
  + **Row-Level Security (RLS)** management.
  + **Access via web or mobile app**.
* Requires a **Power BI Pro or Premium license** for full sharing and collaboration.
* Mainly used by **end users, managers, business teams**.

**Key Differences Table**

|  |  |  |
| --- | --- | --- |
| Feature | Power BI Desktop | Power BI Service |
| Platform | Windows app (installed locally) | Cloud (web-based) |
| Purpose | Build & design reports | Share, collaborate & consume reports |
| Users | Data analysts, report developers | Business users, managers, teams |
| Data Modeling | Yes (relationships, DAX, transformations) | No (model already created in Desktop) |
| Data Refresh | Manual (reload in Desktop) | Automatic (scheduled refresh via gateway) |
| Sharing | Not possible directly (only via .pbix file) | Possible through dashboards, apps, and workspaces |
| Cost | Free | Requires Pro/Premium license for sharing & collaboration |

12) Explain the concept of Direct Query in Power BI.

Ans) **DirectQuery** is a connection mode in Power BI where data is **not imported or stored** in the Power BI file.  
Instead, every time you interact with a report (filter, slicer, visual), Power BI **sends a query directly to the underlying data source** and retrieves the results in real time.

**Key Characteristics of DirectQuery**

1. **No Data Storage in PBIX**
   * The .pbix file only stores metadata (model + visuals), not the data itself.
2. **Real-Time / Near Real-Time Data**
   * Since queries are executed directly on the source system, reports always show the **latest data**.
3. **Supported Sources**
   * Works with databases like **SQL Server, Oracle, Azure SQL, Snowflake, SAP HANA**, etc.
4. **Performance Depends on Source**
   * Query execution speed depends on the **database performance** and **network speed**.
5. **Limited Transformations**
   * Some Power Query transformations and DAX functions are **not supported** in DirectQuery mode.

**Advantages of DirectQuery**

* Real-time reporting (no need for scheduled refresh).
* Handles **very large datasets** without importing them into Power BI.
* Centralized data governance (data stays in the source system).

**Disadvantages of DirectQuery**

* Performance may be slow if the source database is not optimized.
* Limited DAX functions and modeling capabilities compared to Import mode.
* Maximum **1 million rows per query** can be returned (visual limit).
* Requires continuous network connection to the source.

**Example**

Suppose you connect Power BI to a **SQL Server with 100 million rows of sales data**:

* If you use **Import Mode**, Power BI tries to load all 100M rows (very heavy).
* If you use **DirectQuery**, Power BI sends queries like:
* SELECT Region, SUM(SalesAmount)
* FROM Sales
* GROUP BY Region;

The query runs directly on SQL Server and only brings back aggregated results into Power BI.

13) What are Power BI templates and how are they useful?

Ans) A **Power BI Template** (.pbit file) is a special type of Power BI file that contains the **structure of a report without the actual data**.  
It saves the **report design, queries, data model, DAX measures, visuals, and relationships**, but does not store the dataset.

When someone opens a template, they only need to provide the **data source connection details** to load their own data into the pre-built report.

**Key Features of Power BI Templates**

* File extension: **.pbit**
* Contains:
  + Data source queries
  + Data model (relationships, measures, calculated columns)
  + Report design (visuals, formatting, themes)
* Does **not** contain:
  + Actual data (making the file size very small)

**Usefulness of Power BI Templates**

1. **Reusability**
   * Same report structure can be reused across multiple datasets or projects.
   * Example: A “Sales Dashboard Template” can be shared with different regional teams who connect it to their own sales database.
2. **Consistency**
   * Ensures a **standardized report format** across an organization (branding, colors, KPIs).
3. **Efficiency**
   * Saves development time since analysts don’t need to build reports from scratch.
4. **Collaboration**
   * Developers can design the model once, and business users only provide their own data source to use the report.
5. **File Size**
   * Much smaller than .pbix files since it does not store data.

**Example**

Imagine your company wants the **same Sales Dashboard** for India, USA, and UK:

* Instead of building three dashboards, you create **one Power BI Template** (Sales\_Template.pbit).
* Each region connects it to their own sales database.
* Result → Consistent dashboards across regions with minimal effort.

14) How do you handle incremental data refresh in Power BI?

Ans) **Incremental Data Refresh in Power BI**

**Concept**

**Incremental Data Refresh** is a feature in Power BI that allows you to refresh **only new or changed data** instead of reloading the entire dataset.  
This is especially useful for **large datasets** (millions of rows), where reloading all historical data every time is inefficient.

**How It Works**

1. **Define a Date/Time Column**
   * Your fact table (e.g., Sales table) must have a column like *OrderDate* or *TransactionDate*.
2. **Parameters Setup in Power BI Desktop**
   * Create two parameters:
     + **RangeStart** → Starting date of refresh
     + **RangeEnd** → Ending date of refresh
3. **Filter Data in Power Query**
   * Apply a filter on the Date column using **RangeStart and RangeEnd**.
   * Example: Load only data between RangeStart and RangeEnd.
4. **Configure Incremental Refresh Policy** (in Desktop → Model view)
   * Define how much data to **store** and how much to **refresh**.
   * Example:
     + Store last **5 years** of data
     + Refresh only the **last 1 month**
5. **Publish to Power BI Service**
   * Incremental refresh works only in the **Power BI Service** (not in Desktop).
   * When scheduled, only the defined range (e.g., last 1 month) will refresh.

**Benefits of Incremental Refresh**

* ✅ Faster refresh times (only new/changed rows are processed).
* ✅ Reduces load on source systems (no full refresh every time).
* ✅ Efficient handling of **large datasets** (billions of rows).
* ✅ Saves cost in Premium/Pro environments by optimizing resources.

**Example**

* A company has **10 years of sales data (100M rows)**.
* They only need to refresh the **last 30 days** because historical data never changes.
* Using Incremental Refresh:
  + Power BI loads all **10 years once** (initial load).
  + Future refreshes → Only **last 30 days** are refreshed.
  + Result = Huge time savings and faster performance.

**15)** **What is the role of Power Query in Power BI?**

**Ans) Power Query is a data transformation and preparation tool in Power BI.  
It allows users to connect to multiple data sources, clean, reshape, and transform data before loading it into the Power BI data model.**

**It follows the ETL (Extract, Transform, Load) process:**

* **Extract → Connect to different data sources (Excel, SQL, Web, APIs, etc.)**
* **Transform → Clean and shape the data (remove nulls, split columns, merge tables, etc.)**
* **Load → Push the prepared data into the Power BI model for analysis and reporting**

**Key Roles of Power Query in Power BI**

1. **Data Connectivity**
   * **Connects to hundreds of data sources (Excel, SQL Server, Oracle, Web, APIs, Azure, etc.).**
2. **Data Cleaning & Transformation**
   * **Remove duplicates, filter rows, change data types.**
   * **Merge & append queries (combine tables).**
   * **Pivot & unpivot data.**
   * **Add calculated columns.**
3. **Automation**
   * **Once a query is created, it can be refreshed automatically whenever new data is available.**
4. **User-Friendly Interface**
   * **No need for coding — uses a GUI.**
   * **However, advanced users can write scripts in M language (the formula language behind Power Query).**
5. **Consistency**
   * **Ensures the data is clean, consistent, and ready before being used in reports.**

**Example**

**Suppose you import a Sales CSV file every month:**

* **Using Power Query, you can:**
  + **Remove null rows.**
  + **Split “Customer Name” into First and Last name.**
  + **Merge with a “Products” table for category info.**
  + **Load the final clean table into Power BI.**

**16)** **Explain the difference between calculated columns and calculated tables in Power BI.**

**Ans)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | |  |  |  |  |  |   Aspect | Calculated Column | Calculated Table |
| Definition | **A new column created within an existing table using DAX.** | **A new standalone table created using DAX.** |
| Data Level | **Works at the row level of a table.** | **Works at the table level (summary, filtered, or derived table).** |
| Storage | **Increases the size of the existing table.** | **Creates and stores a completely new table.** |
| Use Case | **Add extra information to rows (e.g., Profit = Revenue – Cost).** | **Create new datasets for analysis (e.g., Summary tables, Top N tables).** |
| Example | **Adding “Profit” column to Sales table.** | **Creating a “Top 10 Customers” table.** |

17) How do you create custom visuals in Power BI?

Ans) Custom visuals in Power BI are **user-defined visualizations** created to extend the standard set of charts/graphs.  
They are useful when the default visuals (bar, pie, line, etc.) are not enough for the required analysis.

**Ways to Create Custom Visuals**

**1. Import from AppSource (Ready-Made)**

* Power BI has a marketplace called **AppSource**, where developers publish custom visuals.
* Steps:
  1. In Power BI Desktop → Go to **Visualizations pane**.
  2. Click on **Get more visuals (…) → Get more visuals from AppSource**.
  3. Search and download visuals (e.g., Funnel with % completion, Gantt chart, Sankey diagram).
  4. The visual will appear in the pane and can be used like built-in visuals.

**2. Create Your Own Custom Visuals (Advanced)**

* Developers can create fully custom visuals using **Power BI Developer Tools** with **TypeScript, Node.js, and D3.js**.
* Steps:
  1. Install **Power BI Developer Tools** (npm install -g powerbi-visuals-tools).
  2. Create a new project (pbiviz new MyCustomVisual).
  3. Develop the visual using **TypeScript + D3.js**.
  4. Test it in Power BI Desktop.
  5. Package it as a .pbiviz file and import into Power BI.

**Benefits of Custom Visuals**

* Provide **unique visualizations** not available by default.
* Improve storytelling and user engagement.
* Allow organizations to meet **specific business requirements**.
* Reusable across multiple reports and teams.

**Example**

* If you want to show **customer flow from product views → add to cart → purchases**, you can use or create a **Sankey Chart**.
* Default Power BI visuals cannot do this, so a custom visual is imported or built.

18) What are the best practices for optimizing performance in Power BI?

Ans) Optimizing Power BI reports is important to ensure **fast loading, smooth navigation, and efficient use of resources**.  
Here are the best practices:

**1. Optimize Data Model**

* Remove unnecessary columns and tables (keep only required data).
* Use **star schema** instead of a snowflake schema for better performance.
* Avoid high-cardinality columns (e.g., long text, unique IDs).
* Use proper **data types** (e.g., integers instead of strings).

**2. Reduce Data Volume**

* Apply filters in **Power Query** to load only relevant records.
* Use **aggregation tables** for summarized data.
* Implement **incremental refresh** instead of refreshing full data every time.

**3. Efficient DAX Calculations**

* Use **Measures** instead of Calculated Columns (measures are calculated at query time and don’t increase model size).
* Avoid complex or nested DAX calculations in visuals.
* Pre-calculate values in the data source when possible.

**4. Optimize Relationships**

* Keep relationships simple (prefer **single-direction** relationships where possible).
* Use integer surrogate keys instead of long text keys.

**5. Optimize Visuals**

* Limit the number of visuals on a single page (too many visuals slow rendering).
* Use simple charts instead of heavy visuals like maps or custom visuals (if not necessary).
* Disable unnecessary **interactions** between visuals.

**6. Manage Data Refresh**

* Use **Power BI Gateway** for efficient data refresh.
* Schedule refreshes at off-peak hours.
* Use **incremental refresh** to update only new/changed data.

**7. Use Performance Tools**

* Use the **Performance Analyzer** in Power BI Desktop to identify slow visuals.
* Monitor report performance using **Power BI Service metrics**.

19) How can you integrate Power BI with other Microsoft products like Azure and Office 365?

Ans) **Integrating Power BI with Azure and Office 365**

Power BI integrates seamlessly with other Microsoft products, which helps in **data connectivity, collaboration, and advanced analytics**.

**1. Integration with Azure**

* **Azure SQL Database / Azure Synapse** → Connect directly to cloud databases for real-time analytics.
* **Azure Data Lake** → Store and retrieve big data for use in Power BI.
* **Azure Machine Learning** → Import predictive models and apply AI/ML insights within Power BI reports.
* **Azure Active Directory (AAD)** → Manage user authentication and enable **row-level security**.
* **Azure Analysis Services** → Use advanced data models and connect them with Power BI for reporting.

**2. Integration with Office 365**

* **Excel** → Import Excel data/models into Power BI and export Power BI data back to Excel.
* **SharePoint & OneDrive** → Publish and share Power BI reports directly; keep data synced with cloud files.
* **Teams** → Embed Power BI reports inside Teams channels for collaboration.
* **Outlook & PowerPoint** → Share Power BI dashboards through email or embed them in presentations.
* **Power Automate (part of O365)** → Automate workflows (e.g., send alerts when KPIs cross a threshold).

20) Explain the concept of aggregations in Power BI.

Ans) Aggregations in Power BI are a way to **pre-calculate summary data (totals, averages, counts, etc.)** at a higher level of granularity so that reports run faster.  
Instead of scanning billions of rows every time, Power BI can use an **aggregation table** with fewer rows.

**How It Works**

1. A **detailed fact table** stores all transaction-level data (e.g., millions of sales records).
2. An **aggregation table** stores pre-summarized data (e.g., total sales by region, month, or product).
3. When a report query matches the aggregation level, Power BI automatically fetches results from the **smaller aggregation table** instead of scanning the large fact table.
4. If more detail is needed, Power BI still goes to the detailed table.

**Types of Aggregations**

* **Sum** → Total Sales = SUM(SalesAmount)
* **Average** → Average Sales = AVERAGE(SalesAmount)
* **Count / Distinct Count** → Number of Orders
* **Min / Max** → Earliest Order Date, Latest Shipment Date

**Benefits**

* Faster performance for large datasets.
* Reduced memory usage (smaller aggregation tables).
* Ability to handle **big data** scenarios (millions/billions of rows).

**Example**

* **Fact table (SalesTransactions)**: 100 million rows with columns (Date, Region, Product, SalesAmount).
* **Aggregation table (SalesAggregated)**: 1 million rows with data **summarized by Month + Region + Product**.
* When a report asks for “Total Sales by Region per Month,” Power BI uses the **aggregation table**, making the report load much faster.

21) How do you handle error handling and data quality in Power BI?

Ans) **Error Handling and Data Quality in Power BI**

In Power BI, ensuring **data quality** and handling **errors** is crucial for building accurate and reliable reports.

**1. Using Power Query for Data Quality**

* **Remove Duplicates** → Delete repeated rows to keep clean data.
* **Handle Missing/Null Values** → Replace nulls with default values, averages, or remove them if not needed.
* **Change Data Types** → Ensure columns have correct types (e.g., Date, Number, Text).
* **Standardize Data** → Trim spaces, fix text case (Upper/Lower), and remove special characters.
* **Column Profiling** → Use **Column Quality, Column Distribution, and Column Profile** features in Power Query to check errors, empty values, and anomalies.

**2. Error Handling in Power Query**

* **Error Values** → Rows with errors (like division by zero or invalid date) can be:
  + Replaced with default values.
  + Removed from the dataset.
  + Corrected manually or with rules.
* **Conditional Logic** → Use “IF…THEN…ELSE” to replace faulty values with meaningful data.
* **Applied Steps Pane** → Review and backtrack transformation steps to fix mistakes.

**3. Data Validation in DAX**

* Create calculated measures to check for unusual values (e.g., negative sales, invalid dates).
* Use DAX functions like ISBLANK(), IFERROR(), or COALESCE() to handle nulls or calculation errors.

**4. Monitoring & Governance**

* Schedule **data refresh** and monitor failures in Power BI Service.
* Use **data source credentials** and gateways properly to avoid refresh errors.
* Apply **Row-Level Security (RLS)** to prevent invalid data exposure.

**5. Best Practices**

* Always clean data **as close to the source** as possible.
* Document all transformations in Power Query.
* Perform periodic audits of data quality.

22) What is the purpose of Power BI Embedded and when would you use it?

Ans) **Power BI Embedded**

**Definition**

Power BI Embedded is a Microsoft Azure service that allows developers to **embed interactive Power BI reports, dashboards, and visuals directly into custom applications, websites, or portals**.  
It provides **analytics-as-a-service**, so users can see insights **without needing a Power BI license**.

**Purpose of Power BI Embedded**

* To provide **analytics inside apps** without switching to the Power BI portal.
* To let organizations **share insights with external users (customers, partners, vendors)** securely.
* To allow developers to **customize visuals and user experiences**.
* To reduce licensing costs for apps with many end-users.

**When to Use Power BI Embedded**

1. **Customer-Facing Applications**
   * Example: A bank embeds Power BI dashboards inside its customer portal so clients can analyze their expenses.
2. **Internal Business Applications**
   * Example: A company’s HR app displays employee performance dashboards directly inside the app.
3. **SaaS Products**
   * If you’re building a SaaS platform, you can embed Power BI reports so each client sees their own data securely.
4. **Websites & Portals**
   * Embed live dashboards on websites for users (with security control).

**Benefits**

* Seamless **integration with apps**.
* Rich, **interactive visualizations**.
* **Row-Level Security (RLS)** ensures each user only sees their data.
* Cost-effective way to share reports with **external users**.