Creating Dashboard with Visualization Tool.

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

A:- Power BI is a **business analytics tool** by Microsoft that helps you:

* Connect to different data sources (Excel, databases, cloud apps, APIs, etc.).
* Transform and clean data.
* Create **interactive dashboards and reports** with visualizations (charts, maps, KPIs).
* Share insights across teams via the Power BI Service or embed in apps/websites.

It’s mainly used for **data visualization, business intelligence (BI), and decision-making**.

|  |  |  |
| --- | --- | --- |
| **Feature** | **Excel** | **Power BI** |
| **Purpose** | Spreadsheet for calculations, tables, pivot tables, and light charts. | Business Intelligence & analytics tool focused on **data visualization** and dashboards. |
| **Data Handling** | Best for small to medium datasets. Performance drops with millions of rows. | Handles **very large datasets** efficiently using the Power BI engine. |
| **Visualization** | Charts and pivot charts (basic, static). | Advanced, **interactive, dynamic** dashboards (filters, slicers, drill-through). |
| **Data Sources** | Mainly files (Excel, CSV) with some connections. | Connects to **hundreds of sources** (SQL, APIs, SharePoint, cloud apps, etc.). |
| **Collaboration** | Share via files/email or OneDrive. | Publish dashboards to the **Power BI Service** for real-time sharing and collaboration. |
| **Automation** | Requires VBA/Macros or Power Query for automation. | Built-in **scheduled refresh** and automatic updates from live data sources. |
| **Ease of Use** | Familiar to most users, good for manual work and quick analysis. | Designed for analysts/decision-makers; steeper learning curve but more powerful for **BI**. |

2) Explain the concept of data modeling in Power BI.

A:- **Data modeling** in Power BI is the process of **structuring and organizing data** from different sources so that it can be used effectively for reporting and analysis.

Think of it as building a **logical framework** that connects multiple tables and defines how they “talk” to each other.

## **Key Elements of Data Modeling in Power BI**

1. **Tables**
   1. Imported from Excel, databases, cloud apps, etc.
   2. Can be *fact tables* (transactional data, e.g., sales records) or *dimension tables* (descriptive info, e.g., product details, customer info).
2. **Relationships**
   1. Links between tables using **primary keys** and **foreign keys**.
   2. Example: Sales[CustomerID] → Customers[CustomerID].
3. **Star Schema** (Best Practice)
   1. Central **Fact Table** surrounded by **Dimension Tables**.
   2. Makes analysis fast, clean, and easy.

3) What are the different types of connections available in Power BI?

A:- **1. Import Mode**

* Data is **imported and stored** inside the Power BI file (.pbix).
* The dataset is compressed and stored in the **VertiPaq engine** (in-memory).
* Fastest query performance, since data is loaded into memory.
* Requires periodic **refreshes** to keep data up to date.
* Best for small to medium datasets that fit into memory.

### **2DirectQuery Mode**

* Data is **not imported** into Power BI.
* Queries are sent **directly to the data source** in real time.
* Always up to date with source data.
* Performance depends on the data source speed and network.
* Useful when dealing with **very large datasets** or when real-time data is needed.
* Some DAX and transformation features are limited.

### **3. Live Connection**

* Similar to DirectQuery, but specifically used for **SQL Server Analysis Services (SSAS)**, **Azure Analysis Services**, or **Power BI Datasets**.
* Power BI only acts as a **visualization layer**; all modeling and measures are done in the source model.
* No data storage inside the PBIX file.

### **4. Composite Model (Import + DirectQuery)**

* A **hybrid approach** where you can use both **imported tables** and **DirectQuery tables** in the same model.
* Gives flexibility: you can keep frequently used small tables in memory (import) and keep large tables in DirectQuery.
* Supports relationships and DAX across both modes.

4) How do you handle data transformation in Power BI?

A:- **Steps to Handle Data Transformation in Power BI**

#### **1. Extract (Connecting to Data Sources)**

* Connect to data sources (Excel, SQL Server, APIs, etc.).
* Choose whether to **Import** or use **DirectQuery**.

#### **2. Transform (Shaping Data in Power Query)**

Inside **Power Query Editor**, you can apply various transformations such as:

🔹 **Cleaning Data**

* Remove duplicates
* Handle null or missing values
* Trim/clean whitespace
* Replace values

🔹 **Structuring Data**

* Rename columns
* Change data types (text, number, date, etc.)
* Split or merge columns
* Pivot/unpivot tables

🔹 **Filtering & Sorting**

* Filter rows based on conditions
* Sort data ascending/descending

🔹 **Creating New Columns**

* Add custom columns using formulas (M language)
* Conditional columns (IF-THEN logic)

🔹 **Aggregating & Grouping**

* Group by columns to calculate sums, averages, counts, etc.

#### **3. Load (Bringing Data into Power BI Model)**

* Once transformations are done, you **load** the data into Power BI Desktop.
* Data can then be modeled, related to other tables, and used for DAX calculations and visualizations.

### **Best Practices**

* Do most heavy **data cleaning in Power Query** before loading.
* Keep transformations **step-based and documented** (each step is recorded automatically).
* If working with **large datasets**, filter and reduce data at the source for better performance.

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

A:- **DAX (Data Analysis Expressions)** is a **formula and query language** used in Power BI, Power Pivot, and Analysis Services to create custom **calculations** and **expressions** on your data model.

It’s similar to Excel formulas but much more powerful, designed to work with **relational data** and **filter contexts** in BI models.

#### **🔹 1. Create Calculated Columns & Measures**

* **Calculated Columns** → Stored at the row level in your tables.  
   Example: Profit = Sales[Revenue] - Sales[Cost]
* **Measures** → Calculated on the fly, based on filters/context.  
   Example: Total Sales = SUM(Sales[Revenue])

#### **🔹 2. Enables Advanced Analytics**

* Perform aggregations like **SUM, AVERAGE, MIN, MAX**.
* Use time intelligence (YTD, MTD, QTD, YoY growth).
* Complex business logic (ratios, KPIs, rolling averages).**🔹 3. Works with Context**
* DAX understands **row context** (row-by-row calculations).
* It also uses **filter context** (based on slicers, filters, and visuals).
* This makes dashboards highly **interactive** and **dynamic**.

#### **🔹 4. Extends Beyond Simple Reporting**

* With DAX, you can create:
  + **KPIs** (Key Performance Indicators)
  + **Custom aggregations** (e.g., top N customers, rank, percent of total)
  + **What-if scenarios**
  + **Dynamic calcula**
  + **tions** that respond to user selections

6) Can you explain the difference between calculated columns and measures in Power BI?

A:- **🔹 Calculated Columns vs Measures in Power BI**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Calculated Column** | **Measure** |
| **Definition** | A new column **added to a table**, calculated row by row. | A formula that performs **dynamic calculations** on aggregated data. |
| **Storage** | Values are **stored** in the data model (increase file size). | Values are **not stored**, calculated only when needed (efficient). |
| **Calculation Context** | Works in **row context** (evaluates per row). | Works in **filter context** (depends on visuals, slicers, filters). |
| **Use Case** | When you need a field for each row (e.g., profit per transaction, categorization, flags). | When you need aggregations (e.g., total sales, average revenue, YoY growth). |
| **Performance** | Slower for large datasets (uses more memory). | Faster & more efficient (on-demand calculation). |

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

A:- **🔹 Handling Relationships Between Tables in Power BI**

When you bring multiple tables into Power BI (e.g., Sales, Customers, Products), you often need to connect them so they work together in reports. This is done through **relationships**.

### **1. Types of Relationships**

* **One-to-Many (1:\*)** → Most common.  
   Example: One customer can have many sales.
* **Many-to-One (\*:1)** → Same as above but reversed direction.
* **Many-to-Many (*:*)** → When both tables can have multiple matches.  
   Example: Students and Courses (many students can take many courses).
* **One-to-One (1:1)** → Rare, used when each record matches exactly one other record.

### **2. Cardinality**

Defines how rows in one table relate to rows in another:

* 1:\* (most common)
* \*:1
* *:*
* 1:1

### **3. Cross Filter Direction**

* **Single** → Filters flow only one way (from lookup/dimension table to fact table).
* **Both** → Filters flow both ways (useful for many-to-many or complex models, but can cause ambiguity).

### **4. How to Manage Relationships**

In **Model View** of Power BI:

* Drag and drop fields to create a relationship.
* Or go to **Manage Relationships** → Add/Edit relationships.
* Define cardinality (1:\* / *:*) and cross filter direction (Single/Both).

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

A:- A **Power BI Gateway** is a **bridge** that connects **on-premises data sources** (like SQL Server, Oracle, Excel files, etc. stored in your company network) with the **Power BI Service (cloud)**.

Since the Power BI Service is in the cloud and can’t directly access your local data, the **Gateway acts as a secure middle layer**.

## **🔹 Purpose of a Power BI Gateway**

1. **Enable Data Refresh**
   1. Automatically refresh on-premises data in Power BI Service.
   2. Keeps reports/dashboards **up to date** without manual uploads.
2. **Live/Direct Query Connections**
   1. Lets you use **DirectQuery** or **Live Connection** against on-premises databases.
   2. Queries are securely sent from Power BI Service → Gateway → On-premises source.
3. **Secure Data Transfer**
   1. Uses **encrypted connections** to ensure sensitive data doesn’t leave your network unprotected.
4. **Centralized Data Access**
   1. One gateway can be shared across multiple users and services (Power BI, Power Automate, Power Apps, Azure Logic Apps).

## **🔹 Types of Power BI Gateways**

1. **Personal Mode Gateway**
   1. Runs as an app in your user profile.
   2. Used for individual use cases (not shareable).
   3. Best for small-scale or personal reports.
2. **Standard (Enterprise) Gateway**
   1. Installed on a server.
   2. Can be shared across multiple users and data sources.
   3. Supports **scheduled refresh** and **DirectQuery/Live connections**.
   4. Best for organizations.

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

A:- **1. Publish Your Report**

* First, create your report in **Power BI Desktop** and **publish** it to the **Power BI Service**.
* The dataset gets uploaded to a workspace in the Service.

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

* If your dataset uses **on-premises data sources** (SQL Server, Oracle, Excel on a local server), you need a **Power BI Gateway** installed and configured.
* If the data source is **cloud-based** (e.g., Azure SQL, SharePoint Online, Salesforce), no gateway is needed.

### **3. Go to Dataset Settings**

* In the Power BI Service, navigate to your **workspace** → **Datasets**.
* Find your dataset → click **More Options (…) → Settings**.

### **4. Set Up Scheduled Refresh**

Under **Dataset Settings → Scheduled Refresh**:

* **Toggle ON** “Keep your data up to date.”
* Choose **refresh frequency**:
  + Daily / Weekly
  + Up to **8 times/day** for **Pro**
  + Up to **48 times/day** (every 30 min) for **Premium** capacity.
* Set the **time zone** and **refresh times**.
* Enter credentials for the data source (if required).

### **5. Test and Monitor Refresh**

* Use the **“Refresh Now”** button to manually trigger a refresh and check if it works.
* Monitor refresh history under dataset settings → shows **success/failure logs**.
* If refresh fails, Power BI will notify you (email alert).

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

A:- **Row-Level Security (RLS)** in Power BI is a way to **restrict access to data at the row level** for specific users.

Instead of creating separate reports for different users or departments, you define **roles & filters** that ensure each user only sees the data they are allowed to see.

1. **Define Roles & Rules in Power BI Desktop**
   1. You create roles with **DAX filters** applied to specific tables.
   2. Example:

[Region] = "East"

→ This role allows users to only see rows where Region = East.

1. **Publish to Power BI Service**
   1. After publishing, assign **users or groups** to these roles.
2. **Security Applied at Query Time**
   1. When a user views the report, Power BI automatically applies the filter to the data model in real-time.

## **🔹 Types of RLS**

1. **Static RLS**
   1. Rules are predefined.
   2. Example: East Region manager always sees only East Region data.
2. **Dynamic RLS**
   1. Rules depend on the **logged-in user**.
   2. Example: A formula using USERNAME() or USERPRINCIPALNAME() function so each sales rep only sees their own sales data.
   3. Example DAX:

[SalesRepEmail] = USERPRINCIPALNAME()

## **🔹 Benefits of RLS**

* Ensures **data security & privacy**.
* Avoids creating multiple versions of the same report.
* Simplifies report management and sharing.
* Enforces compliance (sensitive data is hidden from unauthorized users).

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

* A:-**Power BI Desktop** is a **free Windows application** used for **data preparation, modeling, and report creation**.
* It is installed on your local computer.
* Key activities:
  + Connect to data sources
  + Transform and clean data (Power Query)
  + Build data models & relationships
  + Create DAX measures & calculated columns
  + Design interactive reports & dashboards
* File format: **.pbix**

## **🔹 What is Power BI Service?**

* **Power BI Service** is the **cloud-based platform (SaaS)** where you **publish, share, and collaborate** on reports.
* Hosted on **app.powerbi.com** (web).
* Key activities:
  + Publish reports from Power BI Desktop
  + Create dashboards (pin visuals from multiple reports)
  + Share reports with users or groups
  + Schedule data refresh (using gateways if needed)
  + Manage security (Row-Level Security, access permissions)
  + Use **Power BI Apps & Workspaces** for collaboration

## **🔹 Key Differences: Power BI Desktop vs Power BI Service**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Power BI Desktop** | **Power BI Service** |
| **Platform** | Installed app (Windows) | Cloud (Web-based SaaS) |
| **Purpose** | Build reports (development) | Share & collaborate (consumption) |
| **Data Modeling** | Yes (relationships, DAX, calculated columns) | No (only consume models, cannot create relationships) |
| **Data Sources** | Connects to many sources (files, databases, APIs) | Uses published datasets or dataflows |
| **Data Refresh** | Manual refresh | Scheduled refresh via Gateway |
| **Sharing** | Cannot share directly | Share reports, dashboards, apps with others |
| **Cost** | Free | Requires Pro / Premium for sharing & collaboration |

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

A:- **DirectQuery** is a connection mode in Power BI where the data is **not imported into the Power BI model**. Instead, Power BI queries the **data source directly in real time** whenever a user interacts with the report.

This is different from **Import Mode**, where data is loaded and stored inside the PBIX file.

## **🔹 How DirectQuery Works**

* When you create a visual in Power BI (e.g., chart or table), Power BI generates a query.
* That query is sent to the underlying database (SQL Server, Oracle, SAP HANA, etc.).
* The result is returned and displayed in the report.
* Data is always up-to-date because it is **fetched live** from the source.

## **🔹 Advantages of DirectQuery**

1. **Real-Time Data** → Always up to date with the source.
2. **Handles Large Datasets** → No need to import millions of rows into Power BI.
3. **Security** → Uses the security of the source system.
4. **Lightweight PBIX Files** → Since data isn’t stored locally.

## **🔹 Limitations of DirectQuery**

1. **Performance Issues** → Slower, depends on database speed and network.
2. **Limited Transformations** → Some Power Query transformations are not supported.
3. **Restricted DAX Functions** → Not all DAX functions work in DirectQuery mode.
4. **Refresh Limits** → In Power BI Service:
   1. Pro license → up to **8 refreshes/day**.
   2. Premium capacity → up to **48 refreshes/day** (every 30 min).
5. **Dependency on Source** → If the source is down, your report won’t work.

## **🔹 When to Use DirectQuery**

* Data is **too large** to import into memory.
* **Real-time insights** are required (e.g., stock prices, IoT data, sales dashboards).
* Data security policies don’t allow storing data in the Power BI Service.

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

A:- A **Power BI Template** is a special file with the extension **.pbit** (Power BI Template file).

It contains:

* **Report structure** (visuals, layouts, themes, interactions)
* **Data model** (relationships, DAX measures, calculated columns)
* **Queries & transformations** (Power Query steps)

But it does **not** contain the actual **data** itself — only the metadata, schema, and design.

## **🔹 How are Power BI Templates Useful?**

1. **Reusability**
   1. Create a standard report once and reuse it with different datasets.
   2. Example: A sales dashboard template can be used by multiple regional teams with their own data.
2. **Consistency**
   1. Ensures all reports follow the same design standards, KPIs, and formatting.
   2. Useful for large organizations with strict reporting guidelines.
3. **Faster Development**
   1. Speeds up report creation since users don’t have to start from scratch.
   2. Saves time in repetitive reporting scenarios.
4. **Sharing Best Practices**
   1. Experienced Power BI developers can build optimized templates and share them with business users.
   2. Non-technical users can just plug in their data sources.
5. **Parameterization**
   1. Templates can include **parameters** (like date ranges, region names, or connection strings).
   2. Makes them flexible for different users or scenarios.

## **🔹 Example**

Imagine you build a **Financial Report Template (.pbit)**:

* Contains KPIs like Revenue, Profit Margin, YoY Growth.
* Contains standard visuals like Trend Charts, Profit by Region, etc.
* Finance team members in different countries can open the same template and connect to their local databases → instantly get the same styled report with their own data.

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

A:- **🔹 What are Power BI Templates?**

A **Power BI Template** is a special file with the extension **.pbit** (Power BI Template file).

It contains:

* **Report structure** (visuals, layouts, themes, interactions)
* **Data model** (relationships, DAX measures, calculated columns)
* **Queries & transformations** (Power Query steps)

But it does **not** contain the actual **data** itself — only the metadata, schema, and design.

## **🔹 How are Power BI Templates Useful?**

1. **Reusability**
   1. Create a standard report once and reuse it with different datasets.
   2. Example: A sales dashboard template can be used by multiple regional teams with their own data.
2. **Consistency**
   1. Ensures all reports follow the same design standards, KPIs, and formatting.
   2. Useful for large organizations with strict reporting guidelines.
3. **Faster Development**
   1. Speeds up report creation since users don’t have to start from scratch.
   2. Saves time in repetitive reporting scenarios.
4. **Sharing Best Practices**
   1. Experienced Power BI developers can build optimized templates and share them with business users.
   2. Non-technical users can just plug in their data sources.
5. **Parameterization**
   1. Templates can include **parameters** (like date ranges, region names, or connection strings).
   2. Makes them flexible for different users or scenarios.

## **🔹 Example**

Imagine you build a **Financial Report Template (.pbit)**:

* Contains KPIs like Revenue, Profit Margin, YoY Growth.
* Contains standard visuals like Trend Charts, Profit by Region, etc.
* Finance team members in different countries can open the same template and connect to their local databases → instantly get the same styled report with their own data.

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

A:- **Power Query** is the **data connection and transformation engine** in Power BI (also used in Excel, Power Apps, etc.).  
 It lets you **connect, clean, transform, and shape data** before loading it into the Power BI data model.

It uses a functional language called **M (Power Query Formula Language)** behind the scenes.

## **🔹 Role of Power Query in Power BI**

1. **Data Connectivity (Extract)**
   1. Connects to multiple sources: Excel, SQL Server, Oracle, APIs, SharePoint, Web, etc.
   2. Supports both **structured** (databases, files) and **unstructured** (web pages, logs) data.
2. **Data Transformation (Transform)**
   1. Clean messy data (remove nulls, duplicates, errors).
   2. Change data types (e.g., text → date).
   3. Split/merge columns, unpivot data, reshape tables.
   4. Apply filters, sorting, grouping, and aggregation.
   5. Add calculated/custom columns.
3. **Data Shaping (Load)**
   1. Shape data into a star schema-friendly format (fact and dimension tables).
   2. Reduce dataset size by removing unnecessary columns/rows.
   3. Create relationships-ready tables.
4. **Automation & Reusability**
   1. Each transformation is recorded as a **step**.
   2. Steps are **repeatable** — if new data comes in, the same transformations are applied automatically.
5. **Performance Optimization**
   1. Pushes transformations back to the source (query folding) where possible, reducing load time.

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

A:- **🔹 Calculated Columns vs Calculated Tables in Power BI**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Calculated Column** | **Calculated Table** |
| **Definition** | A new column added to an existing table, calculated row by row. | A new table created entirely by a DAX expression. |
| **Where Stored** | Inside an existing table. | As a separate new table in the model. |
| **Context** | Works in **row context** (evaluates per row). | Works in **table context** (creates an entire table at once). |
| **Use Case** | When you need a derived value at the **row level**. | When you need a completely new table for analysis, filtering, or relationships. |
| **Storage** | Increases model size (since values are stored per row). | Increases model size, but stores entire new table. |

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

A:- **🔹 Ways to Create Custom Visuals in Power BI**

### **1. Using Power BI Marketplace (AppSource)**

* Power BI provides a **Visuals Marketplace** with hundreds of prebuilt custom visuals created by Microsoft and third parties.
* Steps:
  + In Power BI Desktop, go to **Visualizations pane → Get more visuals (…) → Get more visuals from AppSource**.
  + Browse (e.g., word cloud, funnel chart, heat maps, bullet charts, etc.).
  + Click **Add** → visual appears in your pane → use it like built-in visuals.

Easiest way, no coding needed.

### **2. Importing a Custom Visual (.pbiviz File)**

* If someone (developer/vendor) has built a custom visual using the Power BI SDK, you can import it.
* Steps:
  + In Power BI Desktop, go to **Visualizations → (…) → Import a visual from file**.
  + Select the .pbiviz file.
  + The visual is added to your toolbox.

### **3. Building Your Own Custom Visuals (for Developers)**

* Power BI provides a **Custom Visuals SDK (Software Development Kit)**.
* Built using:
  + **TypeScript** (logic)
  + **D3.js or other JavaScript libraries** (for rendering)
  + **Node.js tools** for packaging
* Steps (developer workflow):
  + Install Node.js & Power BI Visuals Tools (npm install -g powerbi-visuals-tools).
  + Create a new visual project (pbiviz new MyCustomVisual).
  + Develop your visual in TypeScript + D3.js.
  + Package it into a .pbiviz file (pbiviz package).
  + Import into Power BI Desktop or publish to AppSource.

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

A:- **🔹 1. Data Modeling Best Practices**

* Use a **Star Schema** (Fact + Dimension tables) instead of a Snowflake or flat table.
* Avoid unnecessary columns/rows (remove unused fields in Power Query).
* Reduce cardinality (e.g., don’t keep unnecessary unique IDs or long text columns).
* Prefer **numeric and integer keys** over text-based keys.
* Avoid bi-directional relationships unless absolutely required.
* Use **calculated measures** instead of **calculated columns** where possible (measures are more efficient).

## **🔹 2. Data Source & Storage**

* Use **Import Mode** for best performance (if data size allows).
* Use **DirectQuery only when necessary** (real-time needs or very large datasets).
* For large data, implement **Incremental Refresh** instead of full refresh.
* Leverage **Aggregations** (summary tables for large fact tables).
* Push heavy transformations to the **data source** (SQL, etc.) instead of Power Query.

## **🔹 3. DAX & Calculations**

* Write **optimized DAX**:
  + Use SUMX, FILTER carefully (they scan row by row).
  + Avoid CALCULATE nesting too much.
  + Use variables (VAR) to avoid repeated calculations.
* Avoid creating complex logic in calculated columns — use measures instead.
* Prefer DIVIDE() instead of X / Y to handle divide-by-zero gracefully.

## **🔹 4. Visuals & Reports**

* Minimize the number of visuals per page (each visual = separate query).
* Avoid high-cardinality slicers (e.g., don’t use customer names if millions exist).
* Use **Aggregated fields** instead of showing raw detailed rows in visuals.
* Disable unnecessary **interactions** between visuals.
* Avoid too many bookmarks & complex conditional formatting.

## **🔹 5. Power BI Service (Cloud)**

* Use **Power BI Premium/Capacity** for very large models.
* Enable **Query Caching** to speed up repeated queries.
* Monitor performance with **Performance Analyzer** in Desktop and **Power BI Service Metrics**.
* Use **Gateways** on stable servers with enough resources if using on-prem data.

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

A:- **🔹 Ways to Integrate Power BI with Microsoft Products**

### **1. Integration with Azure**

* **Azure SQL Database / Azure Synapse** → Connect directly as a data source (Import or DirectQuery).
* **Azure Data Lake Storage** → Store and connect to big data for Power BI reports.
* **Azure Data Factory** → ETL pipelines can prepare and load data into Power BI datasets.
* **Azure Machine Learning** → Integrate ML models with Power BI for predictive analytics.
* **Azure Analysis Services** → Build advanced tabular models, then connect Power BI as a front-end.
* **Azure Active Directory (AAD)** → Manage authentication, roles, and Row-Level Security (RLS).

### **2. Integration with Office 365 (Microsoft 365)**

* **Excel Integration**
  + Import Excel tables/pivot tables into Power BI.
  + Use **Analyze in Excel** feature to query Power BI datasets from Excel.
* **SharePoint & OneDrive**
  + Store Power BI reports or source files (Excel, CSV) in SharePoint/OneDrive.
  + Automatic refresh when the file is updated.
* **Teams Integration**
  + Embed Power BI reports directly inside **Microsoft Teams** channels or chats.
  + Pin dashboards for team-wide visibility.
* **Outlook & PowerPoint**
  + Export Power BI visuals to **PowerPoint** (live or static).
  + Share links to Power BI reports directly via **Outlook**.

### **3. Power Platform Integration**

* **Power Apps**
  + Embed Power BI dashboards inside Power Apps.
  + Use Power Apps to collect data → feed into Power BI reports.
* **Power Automate**
  + Automate workflows triggered by Power BI alerts (e.g., send an email when sales drop below target).
* **Power Virtual Agents**
  + Build chatbots that fetch data from Power BI datasets.

## **🔹 Real-World Example**

Imagine a sales organization:

* Data stored in **Azure SQL Database**.
* ETL pipeline managed in **Azure Data Factory**.
* Cleaned data loaded into **Power BI** for dashboards.
* Dashboards shared in **Microsoft Teams** for collaboration.
* Managers analyze reports in **Excel** via "Analyze in Excel".
* Security managed via **Azure Active Directory** with RLS.

20) Explain the concept of aggregations in Power BI.

A:- **🔹 What are Aggregations in Power BI?**

**Aggregations** are **summary tables** that store pre-aggregated (rolled-up) data at a higher level, instead of always querying the detailed fact table.

They help Power BI answer queries **faster** by using the smaller aggregated table instead of scanning billions of rows in the main fact table.

Think of it like having a **summary layer** on top of raw detailed data.

## **🔹 How Aggregations Work**

1. You create a **separate table** that summarizes data from the large fact table.  
    Example: Instead of storing 1 billion transaction rows, store **daily sales totals by product and region**.
2. In Power BI, you define an **aggregation mapping** to tell Power BI which columns/measures in the detail table can be answered from the aggregation table.
3. When a user runs a query (e.g., “Total Sales by Region by Month”), Power BI automatically checks if the **aggregation table** can answer it:
   1. If yes → Uses aggregation table (fast).
   2. If not → Falls back to detailed fact table (slower, but accurate).

## **🔹 Example**

* **Fact Table (Transactions)** → 1 billion rows (Sales by TransactionID).
* **Aggregation Table (AggregatedSales)** → Summarized by ProductID, Date, Region (only a few million rows).

When a user asks:

* **Sales by Region, Month** → Served from Aggregation Table (super fast).
* **Sales by TransactionID** → Served from Fact Table (fallback).

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

A:- **1. Error Handling in Power Query**

Power Query is the first place where errors usually appear (during data extraction & transformation).

* **Use "Remove Errors" option** → Automatically remove rows with errors.
* **Replace Errors** → Replace error values with defaults (e.g., 0 or “Unknown”).
* **Conditional Logic** → Use try … otherwise in M language to gracefully handle failures.

try [Column1] otherwise "N/A"

* **Keep Error Rows** → Create a separate error table to review problematic records.
* **Validate Data Types** → Ensure correct data types (Date, Number, Text) to prevent type mismatch errors.

## **2. Data Quality Handling**

Good data quality = trustworthy insights. In Power Query, you can:

* **Remove Duplicates** → Eliminate redundant rows.
* **Fill Missing Values** → Use “Fill Down” or replace nulls with default values.
* **Trim & Clean** → Remove extra spaces and non-printable characters.
* **Data Profiling Tools** (Power Query → View Tab → Column Quality, Column Profile, Column Distribution):
  + Detect nulls, distinct values, and error percentages.
  + Spot anomalies in your dataset.

## **3. Monitoring Data Quality in Reports**

* Build **Data Quality Dashboards** → Track missing values, error counts, duplicates, etc.
* Use **KPIs** to flag data issues (e.g., % of incomplete records).

## **4. Error Handling in DAX**

* Use DAX functions to prevent runtime errors in calculations:
  + IFERROR(expression, alternateResult) → Handles calculation errors.
  + DIVIDE(x, y, alternateResult) → Avoids divide-by-zero errors.
  + COALESCE(value1, value2) → Returns first non-blank value.

## **5. Governance & Process**

* Use **Dataflows** → Centralize transformations and apply consistent rules.
* Document transformation logic for transparency.
* Automate **data refresh failure alerts** in Power BI Service.
* Work with source system owners to improve data quality upstream.

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

A:- **Power BI Embedded** is an **Azure service (PaaS)** that allows developers to **embed Power BI reports, dashboards, and visuals** into **custom applications or websites**.

It provides **interactive analytics** without requiring users to go to the Power BI Service.

👉 Think of it as: *“Power BI inside your app”*.

# **🔹 Purpose of Power BI Embedded**

1. **Embed Analytics into Applications**
   1. Deliver dashboards/reports directly within your **web or mobile app**.
   2. Users don’t need to log in to Power BI separately.
2. **Customization & Branding**
   1. Fully customize look & feel to match your app’s UI.
   2. Control navigation, filters, and available features.
3. **Scalable & Cost-Effective**
   1. Instead of buying Power BI Pro licenses for every end user, you pay for **Azure capacity (A SKUs)** and allow unlimited external users.
4. **Secure Access**
   1. Row-Level Security (RLS) and Azure Active Directory integration ensure users only see their own data.

# **🔹 When Would You Use Power BI Embedded?**

**Independent Software Vendors (ISVs)** → You build a SaaS app (e.g., CRM, ERP, HR app) and want to provide analytics to customers inside the app.

**Enterprise Applications** → Embedding reports in internal business portals (like intranet apps) for employees.

**Customer-Facing Portals** → Banks, healthcare providers, retailers can embed Power BI dashboards into their **customer portals** to show account details, usage patterns, etc.

**Scenarios with Many External Users** → When giving Power BI access to thousands of customers/partners, Embedded is cheaper than giving everyone a Pro license.

# **🔹 Licensing & Deployment**

* Runs on **Azure capacity nodes (A SKUs)**.
* You manage authentication and report embedding via **APIs + SDKs (JavaScript, .NET, Python, etc.)**.
* Supports **multi-tenant architecture** for SaaS apps.