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) tool by Microsoft** that allows you to:

- **Connect** to different data sources (Excel, SQL, Cloud, APIs, etc.).
- **Transform & Clean** the data (using Power Query).
- **Visualize** the data with interactive dashboards and reports.
- **Share & Collaborate** on dashboards across teams and organizations.

❖ Difference between Power BI and Excel

Feature	Excel	Power BI
Primary Use	Spreadsheet tool for calculations, data entry, and small-scale analysis.	Business Intelligence tool for data analysis, visualization, and reporting.
Data Handling	Good for small to medium datasets (up to ~1M rows).	Can handle very large datasets (millions of rows) efficiently.
Data Sources	Mostly files (Excel, CSV, Text) and limited databases.	Wide range: Excel, SQL, Azure, APIs, cloud services, SharePoint, Salesforce, etc.
Data Transformation	Requires formulas, Power Query, or VBA.	Built-in **Power Query Editor** with advanced transformation capabilities.
Visualization	Charts are static (though slicers/pivots help).	Interactive, dynamic dashboards (drill-down, filters, slicers).
Automation	Needs macros/VBA for automation.	Automatic refresh from connected data sources (schedule refresh).

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

Data modeling in Power BI is the process of organizing, connecting, and structuring data tables so that they can be used effectively for analysis and visualization.

Key Elements of Data Modeling

1. **Tables**
    - Raw data imported from different sources (Excel, SQL, etc.).
    - Can be **fact tables** (transactions like sales, orders) or **dimension tables** (descriptive info like products, customers).
2. **Relationships**
    - Connections between tables (like "CustomerID" in Sales and Customer tables).
    - Types:
        - **One-to-Many (1:*):** Most common (e.g., one customer → many sales).
        - **Many-to-Many (*:*):** Used when both sides can have multiple matches.
        - **One-to-One (1:1):** Rare, usually for reference or lookup tables.
3. **Keys**
    - **Primary Key:** Unique identifier in a dimension table (e.g., ProductID in Products table).
    - **Foreign Key:** Matching column in a fact table (e.g., ProductID in Sales table).
4. **Star Schema vs Snowflake Schema**
    - **Star Schema:** Central fact table connected to multiple dimension tables. (Preferred in Power BI for performance.)
    - **Snowflake Schema:** Dimensions are normalized into multiple related tables (can make the model more complex).
5. **Measures & Calculated Columns**
    - **Calculated Columns:** New columns added at row level.

- **Measures:** Calculations performed on the fly (like SUM of Sales, Average Profit).

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

When you connect Power BI to a data source, there are mainly four connection types:-

1. Import Mode (Default)

- Data is **imported into Power BI's in-memory engine.**
- Best for small to medium-sized datasets.
- **Fast performance** (because everything is loaded into memory).
- Data is **static** → you need to **refresh** it to get updates.
- Supports full **Power Query transformations** and all **DAX functions**.

☑ Example: Importing an Excel file with sales data.

2. DirectQuery Mode

- Data **stays in the source system**, Power BI only queries it when needed.
- Good for **very large datasets** that can't fit in memory.
- Always shows **real-time or near real-time** data.
- Limited transformations and DAX functions compared to Import.
- Performance depends on the **data source speed**.

☑ Example: Connecting to a large SQL Server database directly.

3. Live Connection (special type of DirectQuery)

- Similar to DirectQuery, but only for **Analysis Services (SSAS)**, **Azure Analysis Services**, or **Power BI Datasets**.
- Power BI does not allow modeling (no new tables/columns).

☑ Example: Connecting to an existing corporate data model in SSAS.

4. Composite Model (Hybrid)

- A mix of **Import** + **DirectQuery** in the same model.

- Lets you keep frequently used tables in **Import** (fast) and large/real-time tables in **DirectQuery**.
- Useful when you need both **performance** and **real-time data**.

☑ Example: Importing small dimension tables (like Products, Customers) but using DirectQuery for a huge Sales fact table.

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

In Power BI, data transformation is mainly done using the **Power Query Editor**.
It allows you to clean, reshape, and prepare raw data before loading it into your data model.

Steps to Handle Data Transformation

1. **Load Data into Power Query**

2. **Common Data Cleaning & Transformation Tasks**

- **Remove unnecessary columns** → keep only useful data for analysis.
- **Rename columns** → make names meaningful (e.g., "Cust_ID" → "Customer ID").
- **Change data types** → ensure correct types (Date, Number, Text, etc.).
- **Remove duplicates** → clean repeated rows.
- **Filter rows** → remove blanks, errors, or unwanted records.
- **Replace values** → fix incorrect/missing values.

3. **Data Shaping (Restructuring the Table)**

- **Split columns** → e.g., split "Full Name" into "First Name" & "Last Name".
- **Merge columns** → combine fields like "City + State".
- **Unpivot columns** → convert wide-format data into a long format for analysis.
- **Group data** → aggregate by categories (e.g., total sales by region).

4. **Combine Data from Multiple Sources**

- **Merge Queries (Joins):** Combine two tables (like SQL joins). Example: merge Sales with Customers.
- **Append Queries (Union):** Stack tables with the same structure (e.g., 2023 sales + 2024 sales).

5. **Handle Missing/Invalid Data**

- Replace missing values with default (0, Unknown, or average).
- Remove rows with null values if they are not useful.
- Use conditional logic to correct errors.

6. **Create Calculated Columns** (in Power Query or DAX later)

- Add new columns based on logic, e.g.,
    - If SalesAmount > 1000 → "High Value"
    - Else → "Low Value"

7. **Apply & Load**

- After transformations, click **Close & Apply**

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

DAX (**Data Analysis Expressions**) is a **formula language** used in Power BI, Excel Power Pivot, and Analysis Services to create **custom calculations** and **aggregations** on data.

Why DAX is Important in Power BI?

1. **Advanced Calculations**
    - Built-in aggregations like SUM, AVERAGE, COUNT.
    - Advanced metrics like Year-to-Date (YTD), Moving Average, Growth %, etc.
2. **Row Context & Filter Context**
    - DAX understands relationships between tables (via the data model).
    - It applies filters dynamically based on slicers, visuals, or user selections.
3. **Custom Measures & KPIs**
    - You can define your own **business-specific metrics**.
    - Example: Profit Margin = (Sales – Cost) / Sales.
4. **Time Intelligence**
    - DAX makes it easy to calculate trends over time.
5. **Flexibility in Reporting**
    - With DAX, you are not limited to raw columns.
    - You can create dynamic insights (like Top 10 Customers, % Contribution).

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

Feature	Calculated Column	Measure

Storage	Stored in the model (increases size).	Not stored, calculated on demand.
Row vs Aggregate	Works at **row level**.	Works at **aggregate level**.
Performance	Slower for large datasets (uses memory).	Faster, as it calculates only when needed.
Use Case	When you need a column value for filtering, sorting, or joining.	When you need KPIs, totals, averages, ratios, trends.
Example	Profit per transaction (row-wise).	Total Profit across all transactions (aggregate).

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

Steps to Handle Relationships

1. **Check Data Types**
    - The related columns must have the **same data type** (e.g., both as Text or both as Integer).
2. **Go to Model View**
    - Switch to **Model View** in Power BI.
    - Drag & drop the related fields (e.g., Sales[CustomerID] → Customers[CustomerID]).
3. **Manage Relationships**
    - Use **Home → Manage Relationships** to create, edit, or delete relationships manually.
4. **Set Relationship Properties**
    - **Cardinality** (defines type of relationship):
        - **One-to-Many (1:*)** → Most common (e.g., one Customer → many Sales).

        - **One-to-One (1:1)** → Rare (e.g., Employee → Employee Details).
        - **Many-to-Many (:)** → Both tables can have duplicates (requires caution).
    - **Cross Filter Direction**:
        - **Single** → Filters flow in one direction (better performance, fewer errors).
        - **Both** → Filters flow both ways (used in complex models, but can cause ambiguity).

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

A **Power BI Gateway** is a **bridge** that connects on-premises data sources to Power BI cloud services.

Key Purposes of Power BI Gateway

Secure Connection → Provides a safe way to connect cloud Power BI to on-premises data without moving everything to the cloud.

Data Refresh → Enables **scheduled refresh** of reports so dashboards always show updated data.

Live/Direct Query → Allows real-time queries on on-premises data sources.

Centralized Management → With Standard Gateway, admins can manage multiple data sources and users from one place.

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

When you publish a Power BI report to Power BI Service (cloud), the data may need to be updated regularly. Scheduling a data refresh ensures your dashboards always show the latest information.

- **Prerequisites:**

1. Power BI Service: You need a Power BI Service account.

2. Power BI Gateway: If your data source is on-premises, you need a Power BI Gateway.

➢ **Scheduling Data Refresh:**

1. Go to the Dataset Settings: Navigate to the dataset you want to schedule a refresh for.

2. Click on the Three Dots: Click on the three dots next to the dataset name.

3. Select Schedule Refresh: Select "Schedule Refresh" from the dropdown menu.

4. Configure the Refresh Schedule: Set the refresh frequency, time, and timezone.

5. Choose the Data Sources: Select the data sources you want to refresh.

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

Row-Level Security (RLS) is a feature in Power BI that restricts data access for specific users, so each user sees only the data they are allowed to see.

How Row-Level Security Works

1. **Create Roles in Power BI Desktop**
    - Go to **Modeling → Manage Roles**.
    - Define a role with **DAX filter expressions** to limit rows.
    - Users assigned this role will **only see East region data**.

2. **Test Roles in Power BI Desktop**
    - Use **Modeling → View As** to simulate what a user in a role sees.
3. **Publish to Power BI Service**
    - Publish the report to Power BI Service.
4. **Assign Users to Roles**
    - In Power BI Service → Workspace → Dataset → **Security** → add users to roles.
    - Users see only the rows defined in their assigned role.

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

Power BI Desktop is a free application that allows users to create reports and dashboards on their local machine. It's a powerful tool for data analysis, visualization, and business intelligence.

Power BI Service:

Power BI Service is a cloud-based platform that allows users to share, collaborate, and interact with reports and dashboards.

Feature	Power BI Desktop	Power BI Service
Platform	Windows application	Cloud-based (web)
Primary Use	Data prep, modeling, report creation	Sharing, collaboration, dashboarding
Data Transformation	Full support via Power Query	Limited (mostly viewing, minor edits)
Data Modeling	Full modeling and DAX support	Mostly consumes published dataset
Sharing Reports	Cannot share directly	Can share dashboards/reports with users
Data Refresh	Manual refresh	Scheduled refresh possible

12) Explain the concept of Direct Query in Power BI

DirectQuery is a connection mode in Power BI where data remains in the source system (like SQL Server, Oracle, SAP, etc.) and is queried live whenever you interact with a report.

How DirectQuery Works

- When you create a visual in Power BI, a **query is sent directly to the underlying data source**.
- The result is fetched **on demand** and displayed in the visual.
- Any changes in the source data are **immediately reflected** in the report.

Key Features of DirectQuery

1. **Real-Time or Near Real-Time Data**
    - Reports always show the latest data from the source.
2. **No Data Storage in Power BI**
    - Useful for very **large datasets** that cannot fit in memory.
3. **Limited Transformations**
    - Complex transformations are better handled in the source system.
    - Power Query transformations are supported but limited compared to Import mode.
4. **Supports DAX Measures**
    - You can still create **aggregations, calculated measures, and KPIs**.

When to Use DirectQuery

Very **large datasets** (millions of rows)
Need **real-time reporting**
Data cannot be imported for **security or compliance reasons**

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

A Power BI Template (.pbit file) is a reusable Power BI file that contains:

- Data model (tables, relationships)
- Reports and visuals (charts, dashboards)
- Queries and transformations (Power Query steps)
- Measures, calculated columns, and DAX formulas

How Power BI Templates Are Useful

1. **Reusability**
    - Create a report once and **reuse the structure for multiple datasets**.
    - Example: Monthly Sales Report template can be reused for different regions.

2. **Standardization**
    - Ensures **consistent report design, measures, and visuals** across the organization.
    - Helps maintain **brand colors, formats, and layouts**.
3. **Efficiency**
    - Saves **time and effort**; no need to build the same report repeatedly.
4. **Collaboration**
    - Templates can be shared with colleagues without sharing sensitive data.
    - Each user can connect to their own dataset while keeping the same report structure.
5. **Training & Learning**
    - Templates can serve as **learning resources** for beginners to see standard report design and DAX formulas.

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

Incremental Refresh is a feature in Power BI that allows you to refresh only the new or changed data instead of refreshing the entire dataset every time.

How Incremental Refresh Works

1. **Partitioning the Data**
    - Power BI divides the data into **partitions** (typically by date, e.g., months or days).
    - Only **recent partitions** are refreshed during scheduled refresh, older partitions remain unchanged.
2. **Setting a Date/Time Column**
    - You need a **date or datetime column** in your table to define which data is new or historical.
3. **Defining Refresh Policy**
    - In Power BI Desktop → **Modeling** → **Incremental Refresh**:
        - Define **RangeStart** and **RangeEnd** parameters.

- Example: Keep **5 years of historical data**, refresh **last 1 month**.

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

Power Query is a **data connection and transformation tool** in Power BI (also available in Excel). It allows you to **import, clean, reshape, and combine data** before loading it into the Power BI model for analysis.

Key Roles of Power Query

Data Import – Connects to various sources (Excel, SQL, Web, APIs, etc.).

Data Cleaning – Remove duplicates, blanks, and errors.

Data Transformation – Rename, split, merge, pivot/unpivot, and change data types.

Automation – Every transformation step is recorded and can be reused automatically.

Combine Data – Append or merge queries from multiple tables/sources.

Prepare Data Model – Ensures the dataset is clean and structured before loading into Power BI for analysis.

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

Calculated Columns

A calculated column is a new column added to an existing table using a DAX formula.

Calculated Tables

A calculated table is a new table created in the data model using a DAX formula.

Feature	Calculated Column	Calculated Table
Definition	New column added to existing table	New table created in the model

Calculation	Row by row	Evaluates the entire table (can aggregate/filter)
Storage	Stored in the same table	Stored as a separate table
Use Case	Row-level calculations, slicers, relationships	Summary tables, filtered tables, supporting tables

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

Custom visuals are visualizations not included by default in Power BI.

- **Import from AppSource**
  - Go to Visualizations Pane → More Options (…) → Get More Visuals.
  - Download visuals from Microsoft AppSource (e.g., Heatmaps, Gantt charts, KPI cards).
  - Imported visuals appear in the visualization pane.
- **Import from File (.pbiviz)**
  - Developers can create visuals using Power BI Custom Visual SDK (built with TypeScript, D3.js).
  - Exported as a .pbiviz file.
  - Import into Power BI using Import a Visual from File.
- **Develop Your Own Visual (Advanced)**
  - Install Power BI Visuals SDK (Node.js + PowerShell).
  - Use programming (TypeScript, JavaScript, D3.js) to design the visual.
  - Package and test the custom visual in Power BI Desktop.

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

Power BI performance depends on **data volume, data model design, DAX calculations, and visuals**.

Here are some best practices:

Data Model Optimization:

1. Use a Star Schema: Use a star schema data model to reduce data redundancy and improve query performance.

2. Optimize Data Types: Use optimal data types for columns to reduce storage and improve query performance.

3. Remove Unnecessary Columns: Remove unnecessary columns to reduce data size and improve query performance.

Data Loading and Refresh:

1. Use Incremental Refresh: Use incremental refresh to load only new or updated data.

2. Schedule Refresh: Schedule data refresh during off-peak hours to minimize impact on performance.

3. Use Power BI Gateway: Use Power BI Gateway to manage data refresh and loading.

Report and Visualization Optimization:

1. Use Simple Visuals: Use simple visuals, such as tables and cards, instead of complex visuals.

2. Limit Data Points: Limit the number of data points displayed in visuals.

3. Avoid Over-Filtering: Avoid over-filtering data, which can impact performance.

Query Optimization:

1. Use DAX Optimizations: Use DAX optimizations, such as using CALCULATE instead of FILTER.

2. Avoid Complex Calculations: Avoid complex calculations in measures.

3. Use Query Folding: Use query folding to push calculations to the data source.

Monitoring and Troubleshooting:

1. Use Power BI Performance Analyzer: Use Power BI Performance Analyzer to identify performance bottlenecks.

2. Monitor Query Performance: Monitor query performance and optimize slow queries.

3. Troubleshoot Issues: Troubleshoot issues and optimize performance.

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

Power BI works very well with the **Microsoft ecosystem**, including **Azure, Office 365, Excel, Teams, SharePoint**, and more. Integration allows **seamless data access, collaboration, and advanced analytics**.

Integration with Azure:

1. Azure SQL Database / Data Lake / Synapse → Connect Power BI directly to cloud databases for reporting.

2. Azure Analysis Services → Use as a semantic layer and connect Power BI via Live Connection.

3. Azure Machine Learning → Import ML models into Power BI for predictive analytics.

4. Azure Active Directory (AAD) → Manage authentication, user security, and Row-Level Security (RLS).

Integration with Office 365:

1. Excel → Import PivotTables, Power Query, and PowerPivot models into Power BI; export visuals back to Excel.

2. SharePoint Online → Publish and embed Power BI reports directly in SharePoint pages.

3. Teams → Embed Power BI dashboards inside Microsoft Teams for collaboration.

4. Outlook → Share links or snapshots of Power BI reports via email.

5. OneDrive → Keep Excel/CSV files in OneDrive; Power BI auto-refreshes when the file updates.

20) Explain the concept of aggregations in Power BI.

aggregations are summarized representations of detailed data. Instead of loading and querying millions of rows from a fact table, you can create a smaller **aggregation table** that stores pre-aggregated values.

Types of Aggregations

1. **Implicit Aggregations**
    - Created automatically when you drag and drop fields into visuals.
    - Example: Dragging *Sales Amount* into a table shows SUM(Sales Amount) by default.
2. **Explicit Aggregations**
    - Defined using **DAX measures** or pre-aggregated tables.
    - Example: Creating a measure →
    - Total Sales = SUM(Sales[SalesAmount])
3. **User-Defined Aggregation Tables**
    - A smaller table created in the model (using SQL, Power Query, or DAX) that stores pre-aggregated data.
    - Example: A table that contains *Total Sales by Product and Year* instead of row-level transactions.

Why Aggregations Are Important?

- **Performance Optimization:** Queries run faster because the engine doesn't scan huge fact tables.
- **Memory Efficiency:** Aggregated tables take less space than detailed data.
- **Scalability:** Essential for handling big data scenarios.

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

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?

Power BI Embedded is a **Microsoft Azure service (PaaS)** that allows developers and organizations to **embed Power BI reports, dashboards, and visuals directly into custom applications, websites, or portals**.

When to Use Power BI Embedded

You'd use it in scenarios where:

1. **Analytics inside a custom app**
    - Example: A healthcare company builds a patient portal and wants patients to see **interactive Power BI reports inside the portal** instead of logging in separately to Power BI.
2. **For Customers (External Users)**
    - When you want to share reports with customers/vendors/partners **without requiring them to have a Power BI license**.
    - Instead, the application manages access, and you pay for usage via Azure capacity (A SKUs).
3. **White-label Solutions**
    - SaaS companies embed Power BI into their own software and brand it as their **own analytics solution**.
4. **Custom Security Models**
    - You can implement **Row-Level Security (RLS)** so each customer/user only sees their own data within the same report.
5. **Seamless User Experience**
    - Embedding ensures users don't need to switch between multiple platforms; everything happens inside your app.