

PwC

Power BI Developer Interview Questions (0-5 Years)

Power BI Fundamentals

1. What Are the Key Components of Power BI?

Power BI consists of several key components that help in data visualization, reporting, and sharing.

✓ Key Components:

- ◆ **Power BI Desktop** – A Windows-based application used for data modeling, transformations, and visualization creation.
- ◆ **Power BI Service (Cloud-based SaaS)** – A web-based platform for sharing and collaborating on reports and dashboards.
- ◆ **Power BI Mobile** – Mobile application for accessing reports on smartphones and tablets.
- ◆ **Power Query** – Data transformation tool used to clean and prepare data.
- ◆ **Power Pivot** – Data modeling engine that helps in creating relationships and calculations.
- ◆ **Power View** – Visualization layer where users create charts, graphs, and reports.
- ◆ **Power BI Gateway** – Bridges on-premises data sources with Power BI Service for real-time data refresh.

◆ **Power BI Report Server** – On-premises reporting solution that allows businesses to keep data within their infrastructure.

🚀 **Result:** These components together enable **end-to-end data analysis and business intelligence solutions**.

2. How Does Power BI Differ from Tableau and Excel?

Power BI, Tableau, and Excel are popular tools used for data analysis, but they differ in features, use cases, and capabilities.

✔ Comparison Table

Feature	Power BI	Tableau	Excel
Ease of Use	User-friendly, drag-and-drop	Steep learning curve	Familiar, formula-based
Data Handling	Handles large datasets efficiently	Great for big data visualization	Struggles with large datasets
Integration	Strong Microsoft integration (Azure, SQL, Excel)	Connects well with various databases	Limited connectivity
Visualization	Interactive dashboards	Advanced visual capabilities	Basic charts & pivot tables
Cost	Affordable with free version	Expensive for enterprises	Included in Microsoft 365

🚀 Result:

- **Power BI** is best for Microsoft users needing an affordable, **end-to-end** BI solution.
- **Tableau** is more powerful for **high-end visualizations** and larger datasets.
- **Excel** is good for traditional **spreadsheet-based** analysis.

3. What Are the Different Views Available in Power BI Desktop?

Power BI Desktop provides different views for building, analyzing, and transforming data.

✔ Power BI Views:

◆ Report View 📊

- Default view where you create visualizations and dashboards.
- Allows adding charts, tables, and slicers.

◆ Data View 📄

- Displays **imported and transformed** data in tabular format.
- Helps in **data cleaning and validation**.

◆ Model View 🔗

- Shows **relationships** between tables using a diagram.
- Helps in defining **primary keys and foreign keys**.

📌 **Example:** If you have Sales and Customer tables, **Model View** helps link them using CustomerID.

🚀 **Result:** These views together allow users to **clean data, create relationships, and build reports efficiently**.

4. What Is the Difference Between Power BI Desktop and Power BI Service?

Power BI Desktop and Power BI Service are two essential parts of Power BI, but they serve different purposes.

✔ Key Differences

Feature	Power BI Desktop	Power BI Service
Function	Used for report creation and data modeling	Used for sharing and collaboration
Storage	Stores files locally (.pbix)	Cloud-based storage
Data Transformation	Uses Power Query for data preparation	Limited transformation options

Feature	Power BI Desktop	Power BI Service
Data Refresh	Manual or scheduled	Auto-refresh available
User Access	Single-user environment	Multi-user collaboration

 **Result:**

- Use **Power BI Desktop** for **developing reports and models**.
- Use **Power BI Service** for **publishing and sharing reports with teams**.

Data Connectivity and Transformation

5. What Are the Different Data Sources Power BI Supports?

Power BI supports a wide range of data sources, allowing users to connect, transform, and visualize data from various platforms.

Common Data Sources in Power BI:

- ◆ **Databases** – SQL Server, MySQL, PostgreSQL, Oracle, MongoDB, Snowflake, etc.
- ◆ **Cloud Services** – Azure SQL Database, Google BigQuery, AWS Redshift, Databricks, etc.
- ◆ **Files** – Excel, CSV, XML, JSON, PDF, Parquet.
- ◆ **Online Services** – SharePoint, Google Analytics, Salesforce, Microsoft Dynamics, Facebook, etc.
- ◆ **APIs and Web Data** – REST APIs, OData Feeds.
- ◆ **Big Data** – Hadoop, Spark, Azure Data Lake.

 **Example:** If you need to analyze customer data from an SQL Server database, Power BI allows direct **database connections**, query execution, and data transformation.

 **Result:** Power BI provides extensive connectivity options, making it a **versatile BI tool for any industry**.

6. How Does Power Query Work in Power BI?

Power Query is a built-in tool in Power BI used for **data extraction, transformation, and loading (ETL)**.

✓ How Power Query Works?

1. **Connect to Data Sources** – Import data from databases, files, APIs, or cloud services.
2. **Apply Transformations** – Clean, filter, merge, split, pivot, and format data.
3. **Create Data Models** – Establish relationships between datasets.
4. **Load Data into Power BI** – After transformation, data is loaded into Power BI for reporting.

✦ **Example:** If a dataset contains missing values, Power Query allows you to **replace nulls with default values** before loading into Power BI.

🚀 **Result:** Power Query makes **data preparation simple, repeatable, and automated**, saving **manual effort** in cleaning and transforming data.

7. What Are Common Data Transformation Steps in Power BI?

In **Power Query**, data transformation is an essential step before visualization.

✓ Common Data Transformations:

- ◆ **Removing Duplicates** – Ensures data integrity.
- ◆ **Replacing Nulls** – Fills missing values with meaningful data.
- ◆ **Filtering Data** – Eliminates unnecessary rows.
- ◆ **Splitting Columns** – Separates combined data (e.g., splitting "Full Name" into "First Name" and "Last Name").
- ◆ **Merging Queries** – Combines data from multiple sources.
- ◆ **Pivoting & Unpivoting Data** – Reshapes data for better analysis.
- ◆ **Adding Calculated Columns** – Creates new data fields using formulas.

✦ **Example:** If a dataset has date columns in MM/DD/YYYY format but requires YYYY-MM-DD, Power Query can **change the format automatically**.

🚀 **Result:** These transformation steps **ensure clean, structured, and analysis-ready data** for Power BI reports.

8. How Do You Handle Incremental Data Refresh in Power BI?

Incremental Refresh allows Power BI to update only new or modified data instead of reloading the entire dataset.

✅ Steps to Implement Incremental Refresh:

1. **Enable Incremental Refresh** – Configure settings in Power BI Desktop.
2. **Define Date Ranges** – Set rules for loading historical data and refreshing new data.
3. **Use Parameters** – Create RangeStart and RangeEnd parameters to filter data dynamically.
4. **Publish to Power BI Service** – Ensure incremental refresh applies in the cloud.
5. **Schedule Refresh** – Automate data updates at defined intervals.

📌 **Example:** If you have sales data from **2015 to 2025**, you can configure Power BI to **store data from the last 5 years and refresh only the last 1 month**.

🚀 **Result:** Incremental refresh **improves performance, reduces load time, and optimizes storage usage** in Power BI.

Data Modelling

9. What is a Star Schema, and Why is It Important in Power BI?

A **Star Schema** is a data modeling approach used in Power BI and data warehousing where a **central fact table** is linked to multiple **dimension tables**.

✅ Components of Star Schema:

- ◆ **Fact Table** – Contains numerical data (e.g., sales, revenue, profit).
- ◆ **Dimension Tables** – Store descriptive attributes (e.g., customers, products, time).

✅ Why is Star Schema Important?

- ✅ **Optimized Performance** – Reduces query complexity, making reports faster.
- ✅ **Easy to Understand** – Simple structure improves data interpretation.
- ✅ **Efficient Data Aggregation** – Enhances summarization and analytics.

📌 **Example:** A **Sales Dashboard** can have a fact table (Sales) storing **Order Amount** and linked to dimensions like Customers, Products, and Time.

🚀 **Result:** Star schema is the preferred design in Power BI because it **improves report speed, performance, and data organization**.

10. Explain the Role of Relationships in Power BI Data Models.

In Power BI, **relationships** define how tables connect using common fields. This helps in **combining and analyzing data across multiple tables**.

✅ Types of Relationships in Power BI:

- ◆ **One-to-Many (1:M)** – The most common relationship (e.g., one customer has multiple orders).
- ◆ **Many-to-Many (M:M)** – Used when both tables contain non-unique values.
- ◆ **One-to-One (1:1)** – Each record in one table has only one corresponding record in another.

✅ Importance of Relationships:

- ✅ Enables **seamless data connections** across multiple tables.
- ✅ Allows **aggregations, lookups, and filtering** across tables.
- ✅ Supports **DAX calculations** that depend on data relationships.

📌 **Example:** A Sales table can be related to Customers (via CustomerID) and Products (via ProductID) to get meaningful insights.

🚀 **Result:** Relationships **eliminate data duplication**, enabling **efficient modeling and reporting** in Power BI.

11. What Is the Difference Between Calculated Columns and Measures in DAX?

Both **calculated columns** and **measures** are DAX expressions used in Power BI, but they serve different purposes.

Feature	Calculated Column	Measure
Storage	Stored in the table (precomputed)	Computed on demand
Evaluation	Row-by-row calculation	Aggregation-level calculation
Performance	Takes up memory	Optimized for performance
Use Case	Used when each row needs a computed value	Used for dynamic calculations

📌 **Example:**

- ◆ **Calculated Column:** Total Price = Sales[Quantity] * Sales[Unit Price] (Row-wise).
- ◆ **Measure:** Total Sales = SUM(Sales[Total Price]) (Aggregates on demand).

🚀 **Result:** Measures are **preferred** over calculated columns because they are **faster and memory-efficient**.

12. How Do You Implement Row-Level Security (RLS) in Power BI?

Row-Level Security (RLS) controls **data access based on user roles**, ensuring each user sees only relevant data.

✅ **Steps to Implement RLS in Power BI:**

1. **Define Security Roles** – Create roles in Power BI Desktop.
2. **Apply DAX Filters** – Use FILTER or LOOKUPVALUE to restrict data.
3. **Assign Users** – Assign roles to specific users in Power BI Service.
4. **Test Security Rules** – Validate access using "View as Role" in Power BI Desktop.

📌 **Example:** If a Sales Manager should only see data for their region:

[Region] = USERPRINCIPALNAME()

This ensures users only view data related to their assigned region.

🚀 **Result:** RLS **enhances security, maintains data privacy, and prevents unauthorized access** in Power BI.

DAX

13. What is DAX, and How Does It Work in Power BI?

DAX (Data Analysis Expressions) is a formula language in Power BI, used to perform calculations and aggregations on data models. It is similar to Excel formulas but optimized for **data modeling, relationships, and performance**.

✓ Key Features of DAX:

- ✓ **Calculated Columns** – Row-wise calculations stored in tables.
- ✓ **Measures** – Dynamic aggregations computed on demand.
- ✓ **Filtering & Context Manipulation** – Controls how data is aggregated.
- ✓ **Powerful Functions** – Includes time intelligence, statistical, and logical functions.

✦ Example:

Total Sales = SUM(Sales[Amount])

This measure dynamically calculates the total sales amount in reports.

✈ **Result:** DAX enables **customized calculations** and enhances **data analysis** in Power BI.

14. Explain the Difference Between SUM(), SUMX(), and CALCULATE().

Function	Purpose	Use Case
SUM()	Aggregates a column's total	SUM(Sales[Amount])
SUMX()	Iterates over a table, performing row-wise calculations	SUMX(Sales, Sales[Quantity] * Sales[Price])
CALCULATE()	Modifies the filter context of a measure	CALCULATE(SUM(Sales[Amount]), Sales[Region] = "West")

✦ Example:

Total Revenue = SUMX(Sales, Sales[Quantity] * Sales[Unit Price])

This computes total revenue dynamically by **multiplying each row's quantity with the unit price**.

 **Result:**

✓ **SUM()** works on a column, **SUMX()** iterates row-wise, and **CALCULATE()** modifies filtering behavior.

15. How Do You Create a Dynamic Measure in Power BI?

A **dynamic measure** changes based on slicers, filters, or user input.

✓ **Steps to Create a Dynamic Measure:**

- 1 .Create a table with different metric names (e.g., "Sales", "Profit", "Orders").
- 2 .Use a slicer to let users select a metric.
- 3 .Write a **DAX SWITCH() function** to return the selected metric.

✦ **Example: Dynamic Sales Measure**

Selected Measure =

```
SWITCH(
    SELECTEDVALUE(MeasureTable[Metric]),
    "Sales", SUM(Sales[Amount]),
    "Profit", SUM(Sales[Profit]),
    "Orders", COUNT(Sales[OrderID])
)
```

Now, selecting "Sales" will show total sales, while selecting "Profit" will display profit.

 **Result:** This enables **interactive dashboards** where users can **choose different KPIs dynamically**.

16. What is the Difference Between ALL(), ALLEXCEPT(), and REMOVEFILTERS()?

Function	Purpose	Example
ALL()	Ignores all filters applied to a table or column	ALL(Sales) removes all filters from Sales table
ALLEXCEPT()	Removes all filters except specified columns	ALLEXCEPT(Sales, Sales[Region]) keeps Region filter
REMOVEFILTERS()	Removes filters like ALL(), but applies selectively	REMOVEFILTERS(Sales[Category]) removes Category filter only

🚀 Example: Using ALL() in DAX Measure

Total Sales (Ignoring Filters) = CALCULATE(SUM(Sales[Amount]), ALL(Sales))

This returns **total sales for all data, ignoring filters** applied in reports.

🚀 Result:

- ✅ **ALL()** removes all filters.
- ✅ **ALLEXCEPT()** removes all filters **except specific columns**.
- ✅ **REMOVEFILTERS()** selectively removes filters from a column or table.

Visualization and Reporting

17. How Do You Create an Interactive Dashboard in Power BI?

An **interactive dashboard** in Power BI allows users to explore data dynamically using filters, slicers, and visual interactions.

✅ Steps to Create an Interactive Dashboard:

- 1 .**Import Data** – Connect to a data source using Power Query and clean the data.
- 2 .**Create Relationships** – Ensure correct relationships between tables in **Model View**.
- 3 .**Build Visuals** – Use **bar charts, pie charts, KPIs, cards, and tables** for insights.
- 4 .**Add Interactivity** – Use **slicers, filters, and cross-highlighting** for dynamic exploration.
- 5 .**Enable Drill-through and Drill-down** – Allow users to go deeper into data.

- 6 .**Use Bookmarks and Buttons** – Save states for storytelling and easy navigation.
- 7 .**Optimize Performance** – Reduce unnecessary visuals, enable **query reduction** in slicers, and use **aggregations** for faster loading.
- 8 .**Publish and Share** – Upload the report to **Power BI Service** and share it with stakeholders.

📌 **Example:**

- **Sales Dashboard** with **slicers** for Year, Region, and Product Category.
- **Drill-through** for deep analysis of specific regions.
- **Bookmarks** for toggling between different views.

🚀 **Result:** Users can **click, filter, and drill-down** into the data for deeper insights.

18. What Are Custom Visuals in Power BI, and How Do You Use Them?

Custom visuals are additional Power BI visuals beyond the built-in options, used for advanced reporting needs.

✅ **Types of Custom Visuals:**

- 1 .**Certified Custom Visuals** – Microsoft-approved, available in the Power BI store.
- 2 .**Third-party Custom Visuals** – Created by developers for unique business needs.
- 3 .**Custom Built Visuals** – Developed using **Power BI Developer Tools (TypeScript & D3.js)**.

✅ **How to Use Custom Visuals?**

- 1 .Go to **Visualizations Pane** → Click "**Get more visuals**".
- 2 .Search for the required visual (e.g., **Sunburst Chart, Word Cloud**).
- 3 .Click **Add** and use it in the report.

📌 **Example:**

- **Sankey Chart** for visualizing flow between categories.
- **Gantt Chart** for project timelines.
- **Radar Chart** for performance comparison.

 **Result:** Custom visuals enhance **data storytelling** and make reports more **engaging and insightful**.

19. How Do You Create Drill-through and Drill-down Reports?

- ✓ **Drill-down:** Allows users to explore data hierarchically (e.g., Year → Month → Day).
- ✓ **Drill-through:** Allows users to click a data point and open a detailed report on that category.

✂ Steps to Create Drill-down:

1. Create a hierarchy in **Model View** (e.g., Year > Quarter > Month > Day).
2. Add the hierarchy to a **bar chart or line chart**.
3. Enable the **drill-down button** (downward arrow in the chart).
4. Click on data points to navigate **deeper into details**.

✂ Example:

- A **Sales Trend Chart** where clicking on "2024" breaks down into quarters → months.

✂ Steps to Create Drill-through:

1. Create a **detailed report page** (e.g., Regional Sales Detail).
2. Add a **Drill-through filter** (drag a field like Region to Drill-through pane).
3. In the main report, right-click a visual → **Drill-through** → Select the detailed report.

✂ Example:

- Right-clicking on a **North America sales bar** can navigate to a detailed **"Regional Sales Analysis"** page.

 **Result:** Users can **interactively explore** data at different levels, improving insights and usability.

20. What Is the Use of Bookmarks in Power BI?

✂ **Bookmarks** save the current state of a report, including:

- ✓ Applied **filters and slicers**

- ✓ Selected **visuals and page layouts**
- ✓ Drill-through and expanded visuals

✂ How to Use Bookmarks?

- 1 .Navigate to the **View tab** → Click **Bookmarks Pane**.
- 2 .Set the report view as desired (apply filters, slicers, etc.).
- 3 .Click **“Add”** in the Bookmark pane and rename it.
- 4 .Use **Buttons** to navigate between bookmarks.

📌 Example:

- A **Sales Report** with two bookmarks:
 - **Bookmark 1:** Revenue by Region
 - **Bookmark 2:** Product Performance

🚀 **Result:** Bookmarks create **interactive storytelling** and enhance **user experience** in dashboards.

Performance Optimization

21. How Do You Optimize Power BI Reports for Performance?

Optimizing Power BI reports ensures **faster loading, better responsiveness, and efficient data processing**.

✓ Best Practices for Performance Optimization:

1 .Reduce Data Model Size

- Remove **unnecessary columns** and rows.
- Use **integer-based keys** instead of text for relationships.
- Avoid using **GUIDs (Globally Unique Identifiers)** in joins.

2 .Optimize DAX Measures

- Use **SUMX()** and **COUNTX()** **carefully**, as they iterate row by row.
- Avoid unnecessary **CALCULATE()** statements.
- Use **variables** (VAR) to store intermediate calculations.

3 .Use Proper Data Types

- Convert **DateTime** to **Date** if time is not required.
- Convert text columns to **categorical** when applicable.

4 .Enable Query Reduction

- In slicer settings, enable **“Apply changes”** to reduce unnecessary recalculations.
- Use **Aggregations** to summarize data and reduce query complexity.

5 .Reduce Visual Overload

- Limit the number of visuals on a page (target **8-10 visuals**).
- Use **pre-filtered pages** instead of loading all data in a single view.

6 .Use Star Schema Instead of Flat Tables

- Optimize the data model by using **Fact & Dimension tables** instead of a single large table.

 **Result:** Applying these techniques improves **report responsiveness, reduces query execution time, and enhances user experience.**

22. What Is the Use of Aggregations in Power BI?

 **Aggregations** precompute summarized data to **reduce query time** when working with large datasets.

 **Types of Aggregations:**

1 .Implicit Aggregation:

- Power BI automatically sums, counts, or averages data in visuals.

2 .Explicit Aggregation:

- Using **DAX functions** like SUM(), COUNT(), AVERAGE(), etc.

3 .Pre-Aggregated Tables:

- Creating **summary tables** in Power BI or SQL before importing data.

✓ How Aggregations Improve Performance?

- ✓ Queries fetch **pre-aggregated data** instead of scanning millions of rows.
- ✓ Reduces **memory usage** and **speeds up visuals**.
- ✓ Aggregated tables work well with **DirectQuery mode**, improving real-time reporting.

📌 Example:

- Instead of querying **500M sales transactions**, use an **aggregated table** with SUM(SalesAmount) grouped by Region and Month.

🚀 **Result:** Aggregations reduce **query execution time** and improve **dashboard performance**.

23. How Do You Handle Large Datasets in Power BI?

Power BI struggles with **large datasets**, but **efficient modeling** ensures smooth performance.

✓ Best Practices for Handling Large Datasets:

1 .Use DirectQuery Mode Instead of Import

- DirectQuery fetches data **on demand**, reducing memory load.
- Works well when the dataset is **over 10GB**.

2 .Implement Aggregations

- Store **summary tables** (e.g., Monthly Sales) instead of raw transactional data.

3 .Partition Data for Incremental Refresh

- Load **only new data** instead of reloading the entire dataset.
- Configure Incremental Refresh (e.g., load **last 2 years of data**).

4 .Use Star Schema Instead of Snowflake

- Flatten complex joins to **reduce relationships** and query complexity.

5 .Optimize DAX Queries

- Avoid **nested CALCULATE() functions** and use **SUMX()** carefully.

✦ Example:

- Instead of loading **50M sales records**, aggregate data to "**Sales by Region & Month**" for efficient analysis.

🚀 **Result:** Power BI handles **billions of rows efficiently**, ensuring **faster dashboards and reports**.

24. How Do You Reduce Report Load Time in Power BI?

Slow Power BI reports can frustrate users. Optimize reports using these techniques:

✅ Techniques to Improve Load Time:

1 .Reduce Data Volume

- Remove unnecessary columns & rows before loading data.
- Use **date filters** to limit data (e.g., load only last 2 years of sales).

2 .Optimize Queries in Power Query

- Use **native database queries (SQL statements)** instead of Power Query transformations.
- Disable **auto-detect relationships** in Power BI.

3 .Reduce Visual Complexity

- Reduce the number of visuals on a page (**8-10 visuals max**).
- Avoid heavy visuals like **maps** and **tables with too many rows**.

4 .Use Measures Instead of Calculated Columns

- Measures are calculated **on-demand** while calculated columns **increase data model size**.

5 .Enable Performance Analyzer

- Use **Performance Analyzer** to check slow queries and optimize them.

 Example:

- A report showing **global sales** loads slowly → Apply a **Region filter** to show only **one continent at a time**.

 **Result:** Reports load **faster, use fewer resources, and improve user experience**.

Pratik Jugant Mohapatra