Blinkit

Power BI DeveloperInterview Questions

0-3YOE

16-19LPA

Data Modeling and Relationships

---

1. Explain the difference between Star Schema and Snowflake Schema.

StarSchema:

- Structure: Central Fact Table directly connected to multiple Dimension Tables. Dimension tables are denormalized (flat structure).
- Example:
  - o Fact Table: Sales (with keys like CustomerID, ProductID, etc.)
  - o Dimension Tables: Customer, Product, Region (each having full details directly)
- Benefits:
  - o Simpler design.
  - o Faster query performance due to fewer joins.
  - o Ideal for Power BI and OLAP tools.

Snowflake Schema:

- Structure: Fact Table is connected to normalized dimension tables. That means dimension tables are broken down into sub-dimensions.

swipe

- Example:
    - Dimension "Product" is split into Product → Subcategory → Category.
- Benefits:
    - Reduces data redundancy.
    - Better for data integrity and storage.
- Drawbacks:
    - More complex joins, can slow down performance in Power BI.

In Power BI: Star Schema is highly recommended for data modeling as it simplifies DAX calculations and improves performance.

---

2. How do you handle Many-to-Many (M:M) Relationships in Power BI?

Handling M:M relationships is a critical modeling task to ensure accurate aggregations and visualizations. Here's how you manage them:

Scenario Example:

- Table A: Products (ProductID, ProductName)
- Table B: Campaigns (CampaignID, CampaignName)
-
    - A product can belong to multiple campaigns.
-
    - A campaign can have multiple products.

Solution 1: Using a Bridge Table (Recommended Approach)

➤ Step-by-Step:

1. Create a Bridge Table (e.g., ProductCampaignMapping) containing unique combinations of ProductID and CampaignID.

2. Create One-to-Many (1:) relationships* from:

o Products → ProductCampaignMapping (on ProductID)

o Campaigns → ProductCampaignMapping (on CampaignID)

3. Use DAX measures with TREATAS() or USERELATIONSHIP() if needed for explicit filtering.

Benefits:

- Fully controllable filtering.
- Avoids ambiguity in model.
- Efficient and accurate aggregations.

Solution 2: Use Composite or Bidirectional Relationships (Less Preferred)

- Power BI allows you to directly relate two tables with a many-to-many cardinality using a bidirectional relationship.
- Introduced in later versions of Power BI.
- Works well for small or simple models, but can cause performance issues or filter context ambiguity in complex reports.

Risks:

- DAX becomes harder to debug.
- Unexpected totals and filters due to circular references.

Best Practices:

- Prefer star schema with bridge tables.
- Avoid using many-to-many cardinality unless absolutely necessary.
- Use CALCULATE, TREATAS(), CROSSFILTER(), and USERELATIONSHIP() to manage filters properly.

DAX (Data Analysis Expressions

3. What is the difference between Calculated Columns and Measures?

Feature	Calculated Column	Measure
Definition	Adds a new column to a table using DAX	Calculates values dynamically based on filter context
Storage	Stored in the data model and takes up memory	Not stored; calculated on-the-fly
Context	Operates in row context	Operates in filter context
Usage	Useful for slicers, row-level visuals, relationships	Best for aggregations like SUM, AVERAGE, etc.
Performance	Slower in large models due to memory usage	More optimized and efficient

Example:

- Calculated Column:
  FullName = Customers[FirstName] & " " & Customers[LastName]
- Measure:
  Total Sales = SUM(Sales[SalesAmount])

Best Practice: Use measures instead of calculated columns whenever possible to reduce memory consumption and improve performance.

4. Write a DAX formula to calculate Year-to-Date (YTD) Sales

Prerequisites:

- You should have a proper Date Table marked as a Date Table in your model.

DAX Formula:

YTD Sales =

```
CALCULATE(
  SUM(Sales[SalesAmount]),
  DATESYTD('Date'[Date])
)
```

Explanation:

- SUM(Sales[SalesAmount]): Aggregates sales.
- DATESYTD('Date'[Date]): Returns a table of dates from the start of the year up to the current context.

This measure will dynamically update as the user slices by month, quarter, or any other time period.

---

5. How do you optimize a DAX formula for performance?

Optimizing DAX is crucial in large datasets and enterprise reports.

Techniques:

1. Prefer Measures Over Calculated Columns
    - o Measures are computed on demand, saving memory.
2. Avoid Row Context in Iterators When Possible
    - o Functions like SUMX, FILTER can be slow if used unnecessarily.
3. Use Variables (VAR)
    - o Helps avoid repeated calculations and improves readability.

```
VAR TotalSales = SUM(Sales[SalesAmount])
RETURN TotalSales * 0.1
```

4. Reduce Cardinality
    - o Avoid columns with high cardinality (e.g., long text or timestamps).
    - o Use IDs or categories instead of detailed descriptions.
5. Use Appropriate Relationships

- Use star schema and avoid unnecessary bidirectional relationships.

6. Avoid Complex Nested CALCULATE or FILTER
   - Minimize complexity using KEEPFILTERS(), REMOVEFILTERS(), etc., smartly.
7. Monitor with Performance Analyzer
   - Use Power BI's Performance Analyzer to identify bottlenecks in visuals or measures.

Remember: Efficient models + Clean DAX logic = Fast Power BI reports.

Data Connectivity and Transformation

---

6. What are the different data sources you have connected to using Power BI?

Power BI supports a wide range of data connectors. Here are common examples:

File-based Sources:

- Excel
- CSV
- JSON
- XML
- PDF

Database Sources:

- SQL Server (via DirectQuery or Import)
- MySQL
- PostgreSQL
- Oracle
- Azure SQL Database
- Amazon Redshift

Cloud & Online Services:

- SharePoint Online
- OneDrive
- Azure Blob Storage / Data Lake
- Google BigQuery
- Salesforce

  Dynamics 365

• Web APIs (REST APIs via Power Query with authentication)

Other:

- OData Feed
- ODBC / OLEDB

• SAP HANA, SAP BW

In real-world projects, it's common to use a mix (e.g., Excel for lookup tables + SQL Server for fact data + SharePoint for metadata).

---

7. How do you handle data transformation in Power Query?

Power Query is the ETL (Extract, Transform, Load) engine in Power BI.

Common Transformation Steps:

1. Remove/Filter Rows
   → Remove nulls, duplicates, top/bottom rows.
2. Change Data Types
   → Ensure columns are correctly typed (e.g., date, number, text).
3. Split Columns
   → Split by delimiter or position (e.g., Full Name → First Name & Last Name).
4. Pivot/Unpivot Columns
   → Reformat data for better analysis (e.g., turning month columns into rows).

5. Group By and Aggregate
   → Summarize data during transformation.

6. Merge/Append Queries
   → Combine multiple queries/tables (see Q8 below).

7. Create Conditional Columns
   → Similar to IF logic in Excel.

8. Replace Values
   → Useful for cleaning and standardizing.

All transformations are written in M language behind the scenes, and are step-based and non-destructive.

---

8. Can you explain the difference between Append and Merge queries?

Feature	Append Queries	Merge Queries
Purpose	Stack data from same structured tables	Join two queries based on a common key
Equivalent To	SQL UNION	SQL JOIN
Use Case	Monthly sales files → single table	Customer table + Region table via RegionID
Result	More rows (vertically added)	More columns (horizontally joined)

Examples:

- Append: Combining Jan, Feb, Mar sales Excel files into one dataset.
- Merge: Joining Orders table with Customers on CustomerID.

Tip: Always check data types of join keys in Merge — mismatched types will silently fail.

Data Visualization Best Practices

9. How do you decide which visualization type to use for a dataset?

Choosing the right visualdependsondatatype,audience,andpurposeofthereport.
Below are the most common guidelines:

Based on Data Intent:

Data Intent	Recommended Visualizations
Comparison	Bar chart, Column chart, Line chart
Trend over time	Line chart, Area chart
Proportion	Donut chart, Pie chart, TreeMap
Distribution	Histogram, Box and Whisker chart
Relationship/Correlation	Scatter plot, Bubble chart
KPIs or Summary	Cards, Multi-row cards, KPI visual
Hierarchy	Matrix, Table, Decomposition Tree
Geographical insights	Map, Filled Map, Shape Map

Best Practices:

- Keep visuals simple and minimal – avoid clutter.
- Use color to draw attention or indicate status (e.g., red = risk).
- Use tooltips for detailed drill-down without crowding visuals.
- Limit pie/donut charts to <6 categories for readability.

Ask yourself: What is the one insight the user should take away from this visual?

---

10. How would you optimize a slow Power BI report?

Performance optimization is essential for large datasets and multi-user environments.

Optimization Techniques:

A. Data Model Optimizations

- Use star schema instead of snowflake.
- Remove unused columns and tables.
- Reduce high-cardinality columns (e.g., long texts, timestamps).
- Avoid large calculated columns; prefer measures.

B. Query Optimizations

- Filter at source level in Power Query.
- Disable Auto Date/Time feature (adds hidden tables).
- Use Import mode instead of DirectQuery for speed (unless real-time is required).
- Avoid complex Power Query steps like merging large tables unnecessarily.

C. DAX Optimizations

- Use variables (VAR) to reduce repeated calculations.
- Minimize use of iterators like SUMX, FILTER, etc., unless necessary.
- Use CALCULATE and REMOVEFILTERS efficiently.

D. Visual & Report Layer

- Reduce number of visuals on a single page.
- Avoid heavy visuals like Maps and Tables with too many rows.
- Use Bookmarks and Tooltips instead of many visuals.
- Use Performance Analyzer to identify slow visuals.

Rule of thumb: If a report takes more than 5 seconds to load, find the bottleneck (data model, visuals, or DAX).

---

11. What are the best practices for creating an interactive dashboard?

A good dashboard should be clean, insightful, and engaging.

Design Principles:

1. Define the Purpose First
    - o Identify who the users are and what decisions they need to make.
2. Use Slicers and Filters Wisely
    - o Use slicers for key dimensions like Date, Region, Product.
    - o Use sync slicers across pages to maintain context.
3. Use Drill-Through and Drill-Down
    - o Allow users to click and explore granular insights.
    - o Set up Drill-through pages (e.g., from Region → Store level).
4. Apply Consistent Formatting
    - o Use consistent fonts, colors, and spacing.
    - o Align visuals properly – use gridlines and snap to grid.
5. Limit the Number of Visuals per Page
    - o Avoid overwhelming users. Use navigation buttons or tabs.
6. Leverage Tooltips and Bookmarks
    - o Tooltips add depth without clutter.
    - o Bookmarks create storytelling or guided exploration.
7. Display Key KPIs Clearly
    - o Use cards or KPIs at the top for quick insights (e.g., Total Sales, Growth %).
8. Mobile Responsiveness
    - o Use Mobile Layout View to design mobile-friendly versions.

A dashboard should tell a story without needing a narrator.

Advanced Features

---

12. How do you implement Row-Level Security (RLS) in Power BI?

Row-Level Securityisusedto restrictaccesstodatabasedontheuserviewingthe report.

Steps to Implement RLS:

1. Create a Role in Power BI Desktop:
    - o Go to Modeling → Manage Roles
    - o Click Create, name the role (e.g., RegionManager)
    - o Apply a DAX filter on a table, e.g.:

```
[Region] = "South"
```

2. Use Dynamic RLS for User-Based Filtering:
    - o Create a User table (e.g., email to region mapping).
    - o Add DAX like:

```
[Region] = LOOKUPVALUE(UserTable[Region], UserTable[Email], USERNAME())
```

3. Test the Role in Desktop:
    - o Use View As Roles to simulate the effect.
4. Publish to Power BI Service:
    - o After publishing, go to Dataset → Security.
    - o Assign actual users or groups to the defined role.

Note: RLS works only on import mode or DirectQuery, not with live connection to Analysis Services.

13. Explain the difference between Power BI Desktop and Power BI Service

Feature	Power BIDesktop	PowerBI Service
Purpose	Report creation, modeling, and DAX development	Sharing, collaboration, and publishing
Access Type	Windows-only app (installed locally)	Web-based (cloud platform)
Data Modeling	Full modeling, DAX writing, Power Query	Limited to dataset refresh and connections
RLS Definition	Defined here	Enforced and assigned here
Visual Interaction	Design reports using visuals and custom visuals	View, filter, export, comment, subscribe
Sharing Capability	Not available	Share via workspace, apps, links
Schedule Refresh	Not available	Available (via gateway or cloud data sources)

You develop in Desktop and publish to the Service for collaboration and consumption.

---

14. What are the different types of filters in Power BI?

PowerBI providesfilteringatmultiplelevelstocontroldatavisibility.

Types of Filters:

Filter Type	Scope	Description
Visual Level	Single visual only	Affects only that chart or visual (e.g., show only "2024")

Filter Type	Scope	Description
Page Level	All visuals on a page	Filters everything on the selected report page
Report Level	Entire report (all pages)	Global filter for all visuals and pages
Drillthrough Filter	Specific to drillthrough page	Used to pass filtered context from one page to another
Slicer Filter	User-driven, dynamic	Allows user interaction to filter visuals
	Interactive filtering between visuals	
Cross-Filtering		Click on one visual to affect others
URL Filters	Pass filters in URL	Append ?filter=Table/Field eq 'Value' to a report URL
RLS Filter	Security-based	Automatically applies based on user identity

Best Practice: Minimize unnecessary filters at multiple levels to avoid confusion and performance lag.

Scenario-Based Questions

15. How would you handle missing values and outliers in Power BI?

Power BI does not have built-in imputation tools like Python or R, but you can still handle missing values and outliers using Power Query and DAX.

Handling Missing Values:

In Power Query (M language):

- Use “Replace Values” to fill missing data (nulls).
- Use “Fill Down” or “Fill Up” for forward/backward fill.

• Use Conditional Columns or if statements to replace nulls with default values.

Example:

= Table.ReplaceValue(#"Previous Step", null, 0, Replacer.ReplaceValue, {"Sales"})

Using DAX:

- Use COALESCE() or IF() to replace null values in visuals:

TotalRevenue = COALESCE(SUM(Sales[Revenue]), 0)

Handling Outliers:

Power BI does not automatically detect outliers, but you can:

- Use box plots (via custom visuals) to visualize outliers.
- Use DAX to flag values beyond a statistical threshold.

Example (IQR method):

```
OutlierFlag =
VAR Q1 = PERCENTILEX.INC(Sales, Sales[Amount], 0.25)
VAR Q3 = PERCENTILEX.INC(Sales, Sales[Amount], 0.75)
VAR IQR = Q3 - Q1
RETURN
IF(Sales[Amount] < Q1 - 1.5 * IQR || Sales[Amount] > Q3 + 1.5 * IQR, "Outlier", "Normal")
```

You can then exclude or highlight outliers using this flag in visuals.

Comment PowerBI to get complete pdf

- Average Order Value
- Orders by Category (Fruits, Grocery, Dairy)
- Orders by Location/City
- Cancelled vs Delivered Orders

Step 2: Prepare the Data

- Ensure date hierarchy exists for daily tracking.
- Clean product, order, and customer tables.
- Use relationships in star schema: Fact_Orders, Dim_Date, Dim_Product, Dim_Customer.

Step 3: Create KPIs (DAX)

```
DailyOrders = COUNTROWS(Fact_Orders)

DailyRevenue = SUM(Fact_Orders[Revenue])

AvgOrderValue = DIVIDE(SUM(Fact_Orders[Revenue]), COUNTROWS(Fact_Orders))
```

Step 4: Build Visuals

- KPI cards: Total Revenue, Daily Orders
- Line chart: Daily trends of orders/revenue
- Bar chart: Revenue by product category
- Map: Orders by city/state
- Stacked bar: Delivered vs Cancelled orders

Step 5: Interactivity

- Add slicers for Date, Category, and City.
- Use drill-through pages for city/store-level performance.
- Create bookmarks for switching between Daily / Weekly / Monthly views.

Goal: Make it interactive, fast, and mobile-responsive with consistent design and colors.

---

17. If a report is loading slowly, what steps would you take to improve performance?

Aslowreportusuallypointstoissues inthedatamodel,visuals,orDAX.

Step-by-Step Troubleshooting:

A. Use Performance Analyzer

- Built-in tool to identify slow visuals or DAX.
- Go to View → Performance Analyzer → Start Recording.

B. Optimize Data Model

- Use a star schema.
- Remove unnecessary columns or tables.
- Avoid calculated columns; prefer measures.
- Limit high-cardinality columns (e.g., avoid detailed timestamps).

C. Simplify Visual Layer

- Reduce the number of visuals per page.
- Avoid Map visuals or large Tables with many rows.
- Use pagination or summarized views.

D. Improve DAX Performance

- Use VAR to store interim values.
- Avoid iterators (SUMX, FILTER) unless necessary.
- Use CALCULATE() carefully – it introduces context transition.

E. Query and Source Optimization

- Push filters to Power Query (filter rows early).
- Use Import Mode instead of DirectQuery when real-time is not needed.
- For DirectQuery:
    - o Optimize SQL views/stored procedures.
    - o Use aggregations or composite models if needed.

F. Other Tips

- Disable Auto Date/Time.
- Use Incremental Refresh for large datasets.
- Separate heavy calculations into pre-aggregated tables if needed.

Rule: Think “Reduce, Simplify, Optimize” — from data source to visuals.

SQL for Power BI

18. Write an SQL query to find the top 5 highest-selling products

Assume you have a table Sales with the following columns:

- ProductID
- ProductName
- QuantitySold

You want to find the top 5 products by total quantity sold.

```
SELECT
  ProductName,
  SUM(QuantitySold) AS TotalQuantitySold
```

```
FROM
  Sales
GROUP BY
  ProductName
ORDER BY
  TotalQuantitySold DESC
LIMIT 5;
```

Explanation:

- SUM(QuantitySold) aggregates the quantity sold for each product.
- GROUP BY ProductName groups rows by product.
- ORDER BY TotalQuantitySold DESC sorts products from highest to lowest.
- LIMIT 5 returns only the top 5 rows.

You can run this query in SQL Server, PostgreSQL, or your data source before importing into Power BI or use it as a view or stored procedure.

---

19. How do you join two tables in SQL before loading them into Power BI?

Joining is essential when working with normalized data. You use JOIN statements in SQL to combine data before loading to Power BI.

Example Scenario:

- Orders(OrderID, CustomerID, OrderDate)
- Customers(CustomerID, CustomerName)

You want to create a combined dataset that includes customer names with order info:

```
SELECT
  o.OrderID,
  o.OrderDate,
```

```
  c.CustomerName
FROM
  Orders o
INNER JOIN
  Customers c ON o.CustomerID = c.CustomerID;
```

Types of Joins:

- INNER JOIN: Returns matching records in both tables.
- LEFT JOIN: Returns all records from the left table and matching ones from the right.
- RIGHT JOIN: Opposite of LEFT JOIN.
- FULL OUTER JOIN: Returns all records when there's a match in either table.

Once joined, the result can be loaded into Power BI as a flat table to reduce complexity in your model.

---

20. What is a Common Table Expression (CTE), and how is it used in Power BI reports?

What is a CTE?

A CTE (Common Table Expression) is a temporary result set defined using the WITH clause, which you can reference within a SQL query.

Syntax:

```
WITH CTE_Name AS (
  SELECT column1, column2
  FROM SomeTable
  WHERE condition
)
SELECT *
FROM CTE_Name
```

```
WHERE another_condition;
```

Example Use Case in Power BI:

You want to find products whose sales are above average:

```
WITH ProductSales AS (
  SELECT
    ProductID,
    SUM(Quantity) AS TotalQuantity
  FROM Sales
  GROUP BY ProductID
),
AboveAvgProducts AS (
  SELECT
    ProductID,
    TotalQuantity
  FROM ProductSales
  WHERE TotalQuantity > (SELECT AVG(TotalQuantity) FROM ProductSales)
)
SELECT * FROM AboveAvgProducts;
```

In Power BI, you can:

- Use a CTE inside a SQL view, which you then import.
- Use a CTE in DirectQuery mode for dynamic querying.
- CTEs help simplify complex logic and improve query readability.

Compared to nested subqueries, CTEs are more readable and maintainable.
