HCL

Power BI developer Interview questions
(2+ yeo)

1️⃣ A dashboard is published in Power BI Service. On exporting data, the file name is always xyz.xlsx. How do you add a timestamp to the file name?

Problem Context:
When you export data from a Power BI table/visual in the service (as .xlsx or .csv), the downloaded file uses the *visual name* as its default filename (e.g., xyz.xlsx). There's no native Power BI feature to automatically append a timestamp in the exported file name.

Solution Approaches:

Option 1: Use Power Automate Integration

- Power BI can trigger a **Power Automate flow** (via "Power Automate for Power BI" button) when exporting or refreshing data.
- The flow can:
    1. Retrieve the dataset or table data.
    2. Export it to an Excel or CSV file.
    3. Rename the file dynamically using a timestamp expression.

Example file naming expression in Power Automate:

concat('xyz_', utcNow(), '.xlsx')

or for better readability:

concat('xyz_', formatDateTime(utcNow(), 'yyyyMMdd_HHmmss'), '.xlsx')

✅ This ensures every exported file is uniquely timestamped.

Option 2: Schedule Export via Power BI API

- Use Power BI REST API to export the report or dataset.
- In your script or automation (Python/PowerShell), you can rename the downloaded file with the current timestamp:
- import datetime
- timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
- filename = f"xyz_{timestamp}.xlsx"

Option 3: Manual workaround

- Rename manually after download (not recommended for enterprise setups).

2️⃣ How do you create relationships between tables in Power BI?

Purpose:
Relationships define how tables are connected, enabling **cross-table filtering** and **DAX calculations**.

Steps to Create Relationships:

Step 1: Go to **Model View** (bottom-left side of Power BI Desktop).
Step 2: Drag and drop the common column (e.g., CustomerID from *Sales* to CustomerID in *Customer* table).
Step 3: Power BI automatically detects relationship properties. You can also manually open **Manage Relationships → New → Define**.

Relationship Settings:

- **Cardinality:**
    - *One-to-Many (1:*) →* Most common (e.g., Customer → Sales).*
    - *Many-to-One (*:1)* or *Many-to-Many (:)* — for advanced cases.

- **Cross-filter direction:**
    - *Single* → Filters flow one way.
    - *Both* → Enables bi-directional filtering (use carefully to avoid circular dependencies).
- **Active/Inactive:** Only one relationship between two tables can be active at a time.

Example Scenario:
Let's say you have:

- **Sales Table:** OrderID, CustomerID, ProductID, SalesAmount
- **Customer Table:** CustomerID, CustomerName

You'll create a **1-to-Many** relationship between:
Customer[CustomerID] (1) → Sales[CustomerID] (Many)

Best Practices:

- Always ensure key columns have **unique values** in the "One" side.
- Prefer using **numeric surrogate keys**.
- Avoid bi-directional filtering unless necessary.
- For many-to-many cases, use a **bridge table**.

---

3 DAX formula to calculate Year-over-Year (YoY) Growth

Objective: Compare current year's sales with the previous year's sales.

Let's assume:

- Sales table has a column SalesAmount
- Date table is connected to Sales[OrderDate]
- Date table is marked as a **Date Table** in Power BI

Step 1: Calculate Total Sales

Total Sales = SUM(Sales[SalesAmount])

Step 2: Calculate Previous Year Sales

Previous Year Sales = CALCULATE([Total Sales], SAMEPERIODLASTYEAR('Date'[Date]))

✅ SAMEPERIODLASTYEAR() shifts the current filter context by one year.

Step 3: Calculate YoY Growth

There are two common ways to represent YoY Growth:

(a) Absolute Difference:

YoY Growth = [Total Sales] - [Previous Year Sales]

(b) Percentage Growth:

YoY Growth % =

DIVIDE([Total Sales] - [Previous Year Sales], [Previous Year Sales])

This gives the percentage change from the previous year to the current year.

Step 4: Example Output

Year	Total Sales	Prev Year Sales	YoY Growth %
2022	1,200,000	1,000,000	20%
2023	1,500,000	1,200,000	25%

Alternative Function:

You can also use:

YoY Growth % =

VAR PY_Sales = CALCULATE([Total Sales], DATEADD('Date'[Date], -1, YEAR))

RETURN DIVIDE([Total Sales] - PY_Sales, PY_Sales)

Both SAMEPERIODLASTYEAR() and DATEADD() can achieve the same result depending on your calendar structure.

✅ **Final Summary:**

Question	Key Point
Export with timestamp	Use Power Automate or REST API naming with utcNow()
Create Relationships	Use Model View, define key columns, choose cardinality
YoY Growth DAX	Use SAMEPERIODLASTYEAR() or DATEADD() for previous year comparison

---

🔛 Creating Dynamic Charts Based on User Selection

Scenario:
You want users to switch between different measures (e.g., Sales, Profit, Quantity) or dimensions (e.g., Region, Category, Year) **using a slicer or button** — without creating multiple visuals.

✅ **Approach 1: Using a *Disconnected Table* + *Measure Switching***

Step 1: Create a Disconnected Table

- Go to *Home* → *Enter Data* → create a table manually:

  Metric

  Sales

  Profit

  Quantity

- Name it **Metric Selector**.

Step 2: Create a Slicer

- Add the Metric column from the *Metric Selector* table to a slicer on the report.

Step 3: Create a Dynamic Measure

```
Selected Metric Value =
VAR SelectedMetric = SELECTEDVALUE('Metric Selector'[Metric])
RETURN
SWITCH(
  TRUE(),
  SelectedMetric = "Sales", SUM(Sales[SalesAmount]),
  SelectedMetric = "Profit", SUM(Sales[Profit]),
  SelectedMetric = "Quantity", SUM(Sales[Quantity]),
  BLANK()
)
```

Step 4: Use in Charts

- Use Selected Metric Value as the **Y-axis** in charts.
- Users can now pick the metric dynamically via the slicer.

✅ **Approach 2: Dynamic Axis Selection**

If users want to switch between *Region / Product / Category*, create a disconnected "Dimension Selector" table and use **Field Parameters** (new Power BI feature).

Steps:

1. Go to **Modeling → New Parameter → Fields.**
2. Select fields you want users to toggle between (e.g., Region, Category, Product Name).
3. Power BI automatically creates a parameter table and slicer.
4. When users pick an option, the chart axis changes dynamically.

Real-World Use Case:

- Creating dashboards with limited visuals but maximum interactivity.

- Allows business users to customize the report without editing visuals.

5️⃣ Drill-Down and Drill-Through Reports

These are both interactivity features but used for different purposes.

✅ Drill-Down

Purpose:
To explore data hierarchically within the **same visual** (e.g., Year → Quarter → Month → Day).

Steps:

1. Create a hierarchy in your **Date Table** (Year > Quarter > Month > Day).
2. Add the hierarchy to the visual's axis.
3. Enable **Drill Mode** (down arrow icon in the top-right corner of the visual).
4. Users can:
    - **Click once** → Drill down to the next level.
    - **Right-click → Drill down / Drill up**.
    - **Expand all levels** to view multiple hierarchies simultaneously.

Example:

- Chart initially shows *Sales by Year*.
- Clicking 2023 drills down to *Quarter-wise sales*, then *Month-wise*, etc.

✅ Drill-Through

Purpose:
To move from one page to another report page containing *detailed information* about the selected item.

Steps:

1. Create a new page → name it "Customer Details".

2. On that page, add a *Drill-through filter field* (e.g., Customer Name).
3. Add visuals that show details about that customer (orders, returns, etc.).
4. On the main report page, right-click any *Customer Name* → select *Drill-through* → *Customer Details*.

Example:

- Right-click on *Customer = John Doe* in the summary dashboard → drill through to a detailed page showing John's purchase history.

Difference Between Drill-Down and Drill-Through

Feature	Drill-Down	Drill-Through
Purpose	Navigate hierarchical levels within same visual	Navigate to a different detailed report page
Example	Year → Quarter → Month	Customer Summary → Customer Detail Page
Interaction Type	Within same chart	Across pages
Filters Applied	Automatically from hierarchy	Context-based (specific selection filters next page)

6️⃣ Handling Data Refresh Failures

Problem Context:
Data refresh failures are common in Power BI Service — especially when scheduled refreshes depend on gateways, credentials, or data source changes.

✅ Common Reasons for Refresh Failures

Cause	Description
Gateway not configured	On-premise data sources need an enterprise or personal gateway.
Invalid credentials	Passwords expired, token expired, or credentials changed.
Schema changes	Column renamed or removed from the source.
Query timeout / large dataset	Dataset exceeds memory limits or long query execution.
Missing dependencies	Linked queries in Power Query referencing invalid steps.

✅ **Steps to Troubleshoot**

1️⃣ **Check Refresh History**

- In Power BI Service → Dataset → **Refresh history** → check detailed error messages.

2️⃣ **Verify Gateway Connection**

- Go to **Manage Gateways** → ensure your data source is connected.
- Update credentials if expired.

3️⃣ **Test Queries in Power BI Desktop**

- Refresh data manually in Desktop → check if any transformation or column name has changed.
- Fix Power Query errors.

4️⃣ **Incremental Refresh (for large data)**

- Enable Incremental Refresh for large fact tables to reduce refresh load.
- Configure partitions by Date column.

5️⃣ **Optimize Data Model**

- Remove unnecessary columns, reduce cardinality, disable auto-date hierarchy.
- Use query folding wherever possible.

6️⃣ **Check Scheduled Refresh Limits**

- Power BI Pro: 8 refreshes/day
- Power BI Premium: 48 refreshes/day

7️⃣ Use Power Automate for Notifications

- Set up a Power Automate flow → send an email or Teams alert when refresh fails.

✅ Example Real-World Scenario

You have a Sales dashboard connected to an on-prem SQL Server via Gateway.
If the Gateway service is stopped or credentials expire, the scheduled refresh will fail with an error:
"Cannot connect to the data source".

Fix: Restart the gateway service, re-enter credentials under *Manage Connections* → *Edit credentials*, and test the connection again.

✅ Final Summary

Question	Key Concept	Solution Summary
Dynamic Charts	User-driven visuals	Disconnected table + measure switching / field parameters
Drill-Down & Drill-Through	Data exploration	Hierarchical vs page-level navigation
Refresh Failures	Data reliability	Check gateway, credentials, schema, optimize refresh

---

7️⃣ Handling Data Transformation in Power Query

Context:
Power Query (a.k.a. Query Editor) is the **data preparation layer** in Power BI — used to **clean, transform, and shape data** before loading it into the model.

Handling data transformation efficiently ensures your model is optimized, consistent, and refreshes smoothly.

✅ Common Data Transformations in Power Query

Transformation	Description	Example / Function
Remove Columns	Keep only necessary columns to reduce model size	*Home → Remove Columns*
Rename Columns	Rename for clarity	*Transform → Rename Column*
Change Data Type	Convert text, date, number, etc.	*Transform → Data Type → Whole Number / Date / Text*
Remove Duplicates	Eliminate repeated rows	*Remove Rows → Remove Duplicates*
Filter Rows	Filter specific records	*Keep Rows / Remove Rows*
Replace Values	Replace missing or incorrect values	*Transform → Replace Values*
Merge Queries (JOIN)	Combine two tables based on a key	*Home → Merge Queries → Join Type*
Append Queries (UNION)	Stack data from multiple sources	*Home → Append Queries*
Add Conditional Columns	Logic-based derived columns	*Add Column → Conditional Column*
Group By	Aggregate data	*Transform → Group By (Sum, Count, Avg, etc.)*
Split Column	Split text by delimiter	*Transform → Split Column → By Delimiter*
Pivot / Unpivot Columns	Convert rows → columns or vice versa	*Transform → Pivot / Unpivot Columns*

Transformation	Description	Example / Function
Remove Errors	Eliminate rows with transformation errors	*Remove Rows → Remove Errors*

✅ **Example: Cleaning a Sales Table**

Raw Table:

Date	Product	Amount	Discount%	Region	Status
2025-09-10	Laptop	50000	10	North	Delivered
NULL	Phone	15000	NULL	South	Cancelled

Steps in Power Query:

1. **Remove null Date rows** → Filter out blanks.
2. **Replace null Discount% with 0** → Transform → Replace Values.
3. **Add Net Sales column** → = [Amount] * (1 - [Discount%]/100).
4. **Change Date to Date Type**.
5. **Trim text fields** (Product, Region).
6. **Load transformed data to model**.

M-code (auto-generated by Power Query):

```
= Table.AddColumn(#"Replaced Value", "Net Sales", each [Amount] * (1 - [Discount%]/100))
```

✅ **Best Practices**

- Always **remove unnecessary columns early** in the query to reduce load time.
- Use **Query Folding** — perform transformations that push back to the source (SQL, etc.).
- **Avoid too many nested steps** — they can slow refreshes.
- Rename steps meaningfully (e.g., “Filtered_Null_Dates” instead of “Step3”).
- Use **staging queries**: extract → transform → load in stages for clarity.

8️⃣ How Would You Get Sales for the Last 45 Days?

Objective:
To calculate total sales for the most recent 45 days dynamically from today's date.

Let's assume:

- Table: Sales
- Columns: OrderDate, SalesAmount
- Related Date table: 'Date'[Date]

✅ DAX Solution 1: Using DATESINPERIOD()

```
Sales (Last 45 Days) =
CALCULATE(
  SUM(Sales[SalesAmount]),
  DATESINPERIOD(
    'Date'[Date],
    MAX('Date'[Date]),
    -45,
    DAY
  )
)
```

Explanation:

- DATESINPERIOD() creates a **date range** from the **current max date** and looks **back 45 days**.
- CALCULATE() applies this filter to the SalesAmount measure.

Result:
Returns total sales for the last 45 days (rolling window) relative to the most recent date in your data context.

✅ **DAX Solution 2: Using TODAY() (when not using a Date table)**

If you don't have a Date table, use TODAY() directly:

```
Sales (Last 45 Days) =
CALCULATE(
  SUM(Sales[SalesAmount]),
  FILTER(
    Sales,
    Sales[OrderDate] >= TODAY() - 45 &&
    Sales[OrderDate] <= TODAY()
  )
)
```

Explanation:

- Filters Sales table where the order date is within the last 45 days from today.

✅ **DAX Solution 3: Rolling 45-Day Sales Trend**

If you want to show **trend by day**, use a measure that calculates daily sales in a rolling 45-day window:

```
Rolling 45-Day Sales =
CALCULATE(
  [Total Sales],
  DATESINPERIOD(
    'Date'[Date],
    MAX('Date'[Date]),
    -45,
    DAY
```

)

)

Then plot 'Date'[Date] on the X-axis and this measure on Y-axis → gives a **moving window trend chart**.

✅ **Example Output**

Date Range	Total Sales (₹)
Last 7 days	₹3,25,000
Last 30 days	₹8,10,000
Last 45 days	**₹10,75,000**

✅ **Best Practices**

- Always use a **proper Date table** for consistent time calculations.
- Mark the table as a *Date Table* in Power BI.
- When visualizing rolling periods, ensure the **x-axis is continuous** (not categorical).
