🧠 Power BI in 15 Days – The Ultimate Guide for Data Analysts

Designed for interview readiness and practical project building!

🔹 Day 1: Introduction to Power BI – Components, Workflow, and Use Cases

✅ 1. What is Power BI?

Power BI is a **Business Intelligence and Data Visualization** tool by Microsoft used to transform raw data into meaningful insights through interactive dashboards and reports.

✅ 2. Power BI Components:

- **Power BI Desktop**: Design reports
- **Power BI Service**: Share via cloud
- **Power BI Mobile**: On-the-go reports
- **Power BI Gateway**: Refresh on-premise data
- **Power BI Report Builder**: For paginated reports

✅ 3. Workflow Overview:

1. Get Data (from Excel, SQL, APIs, etc.)
2. Clean & Transform with Power Query
3. Build Relationships (Data Modeling)
4. Add Calculations using DAX
5. Visualize using charts & slicers
6. Publish to Power BI Service

✅ 4. Use Cases:

- **Sales & Marketing Dashboards**
- **Customer Segmentation**
- **Financial Reporting**
- **Real-Time Monitoring**
- **Executive Summary Dashboards**

🔧 Hands-On:

- Download and install **Power BI Desktop**
- Load a simple Excel file (e.g., Sales_Data.xlsx)
- Navigate through **Home**, **Data**, **Model**, and **Report** tabs

---

💬 INTERVIEW QUESTIONS (Medium to High):

🔹 Q1. What is the difference between Power BI Desktop and Power BI Service?

Ans: Desktop is for designing reports; Service is for publishing, sharing, and collaboration. Desktop is local, Service is cloud-based.

🔹 Q2. How is Power BI different from Excel?

Ans: Power BI handles larger datasets, has real-time data refresh, advanced modeling, and interactive visuals.

🔹 Q3. What are common business scenarios where Power BI is used?

Ans: Customer churn analysis, sales trend analysis, operations dashboards, KPI tracking, marketing campaign analysis.

🔹 **Q4. What is the role of Power Query in Power BI?**

Ans: It is the ETL tool in Power BI – used to import, clean, transform, and load data.

🔹 **Q5. Can you automate data refresh in Power BI? How?**

Ans: Yes, using **Power BI Gateway** for on-premise sources, or **scheduled refresh** for online sources through Power BI Service.

🔹 **Day 2: Power Query – Data Cleaning & Transformation**

Power Query is the **data preparation engine** in Power BI. It allows you to connect to data sources, clean the data, and load it into your data model.

✅ **1. Power Query Editor Overview**

When you click **"Transform Data"** in Power BI, you enter Power Query Editor.
Main panels:

- **Queries pane** – all your tables
- **Applied Steps** – logs your transformations
- **Formula Bar** – shows M code
- **Preview Grid** – view and clean data

✅ **2. Key Data Cleaning Operations**

🧹 **Remove Columns / Rows**

Home → Remove Columns → Choose Unwanted Columns
Home → Remove Rows → Remove Top Rows / Blank Rows / Errors

🔄 Change Data Types
Transform → Data Type → Choose (e.g., Whole Number, Date, Text)

🔀 Split Column
Transform → Split Column → By Delimiter (e.g., comma, space)

🩹 Replace Values
Transform → Replace Values → Enter Old & New Values

🗓️ Extract Date / Time
Add Column → Date → Year / Month / Day

📝 Rename Columns
Double-click column header or use Transform → Rename

🪄 Trim & Clean
Transform → Format → Trim / Clean

✅ 3. Merge and Append Queries

📌 **Merge**: Combine tables **side-by-side** (like SQL JOIN)
Home → Merge Queries → Select Matching Columns

📌 **Append**: Stack rows **vertically** (like SQL UNION)
Home → Append Queries → Select Tables

✅ 4. Applied Steps = Transformation Pipeline

Every change in Power Query is logged as a step and can be reordered or deleted. This is what makes Power BI **reproducible and**

modular.

💬 INTERVIEW QUESTIONS (Medium to High)

🔹 Q1. What is the difference between Merge and Append in Power BI?

Ans:

- *Merge* joins tables based on keys (like SQL JOIN)
- *Append* stacks them vertically (like UNION)

🔹 Q2. What are some common data cleaning steps you do in Power Query?

Ans: Removing nulls, changing data types, trimming spaces, splitting columns, replacing values.

🔹 Q3. Is Power Query case-sensitive?

Ans: Yes, string operations like replace and compare are case-sensitive unless handled using Text.Lower() or Text.Upper() functions in M code.

🔹 Q4. How would you remove duplicate rows in Power BI?

Ans:

Home → Remove Rows → Remove Duplicates

🔹 Q5. What is the advantage of using Power Query over Excel formulas for data cleaning?

Ans: More scalable, structured with repeatable steps, less prone to errors, and ideal for automation.

🔹 Day 3: Power BI Data Modeling – Relationships, Cardinality, Star vs Snowflake Schema

Once your data is cleaned using Power Query, the next step is **building a proper data model**—this is the **heart of Power BI**.

✅ 1. What is a Data Model?

A **data model** connects your tables logically using relationships, enabling calculations and visualizations across multiple tables.
Think of it like the *backbone* of your dashboard.

✅ 2. Create Relationships in Power BI

📍 Go to **Model View**
📍 Drag fields to create relationships (usually *primary key → foreign key*)
You can also use:
Model → Manage Relationships → New

✅ 3. Relationship Cardinality

Power BI supports four types:

Cardinality	Description
One-to-One (1:1)	Unique value on both sides
One-to-Many (1:*)	Unique on one side, duplicates on the other
Many-to-One (*:1)	Same as above, reversed
Many-to-Many (*:*)	Non-unique on both sides (used sparingly)

Tip: Most models use * *1:* (One-to-Many)**.

✅ 4. Cross Filter Direction

- **Single**: Filters flow from one side to the other
- **Both**: Enables bi-directional filtering (*can cause ambiguity*)

Use **Single** unless absolutely required.

✅ 5. Star vs Snowflake Schema

⭐ **Star Schema** (Recommended):

- Central Fact Table surrounded by Dimension Tables
- Simple, fast, and preferred by Power BI

❄️ **Snowflake Schema**:

- Dimensions are normalized (split into multiple related tables)
- More complex; can slow down performance

✅ 6. Active vs Inactive Relationships

Power BI allows only **one active relationship** between two tables at a time.

To use inactive ones in DAX:
CALCULATE(SUM(Sales[Amount]), USERELATIONSHIP(Date[FullDate], Sales[OrderDate]))

💬 INTERVIEW QUESTIONS (Medium to High)

🔹 Q1. What is the difference between Star and Snowflake Schema in Power BI?

Ans:

- Star Schema has de-normalized dimension tables → faster, simpler
- Snowflake is normalized → more joins, can affect performance

🔹 **Q2. What is the role of cardinality in Power BI relationships?**

Ans: Determines how data flows between tables, affects filter behavior, and calculation context.

🔹 **Q3. Can you have more than one relationship between two tables?**

Ans: Yes, but only one can be active. Others are used via USERELATIONSHIP() in DAX.

🔹 **Q4. What happens if there are circular dependencies in relationships?**

Ans: Power BI will throw an error and prevent model from loading – must redesign the model to eliminate loops.

🔹 **Q5. When would you use bi-directional filtering?**

Ans: For complex models where filters must flow in both directions (e.g., role-playing dimensions), but use sparingly due to ambiguity risks.

🔹 **Day 4: Power BI DAX Basics – Calculated Columns, Measures, Row Context vs Filter Context**

DAX (**Data Analysis Expressions**) is the **brain** of Power BI. It lets you create custom calculations for better insights.

✅ **1. Calculated Columns vs Measures**

Feature	Calculated Column	Measure
Stored in Data Model	Yes (takes up memory)	No (calculated on the fly)
Evaluation Context	Row Context	Filter Context
Use Case	Row-wise logic (e.g., profit = sales - cost)	Aggregations (e.g., total sales)
Syntax Example	Sales[Profit] = Sales[Revenue] - Sales[Cost]	Total Sales = SUM(Sales[Revenue])

✅ 2. Create Them In Power BI

- **Calculated Column**:
  Modeling → New Column
- **Measure**:
  Modeling → New Measure

✅ 3. Row Context vs Filter Context

Row Context:

- Used in **Calculated Columns**
- Operates on each row independently
  👉 Think: "What is true **for this row**?"

Filter Context:

- Used in **Measures**
- Depends on visuals/filters in reports
  👉 Think: "What is true **given this filter**?"

✅ 4. Implicit vs Explicit Measures

- **Implicit**: Auto-created by dragging a field (like SUM, AVG)
- **Explicit**: Written with DAX
  🔥 Explicit measures are **better for reusability** and **control**

✅ 5. Common DAX Functions

- SUM(), AVERAGE(), COUNT(), CALCULATE(), FILTER()
- IF(), SWITCH(), RELATED(), ALL(), REMOVEFILTERS()

💬 INTERVIEW QUESTIONS (Medium to High)

🔹 Q1. What's the difference between Calculated Columns and Measures?

Ans: Columns are evaluated row-by-row and stored, Measures are evaluated dynamically during report rendering.

🔹 Q2. What is Row Context and when is it used?

Ans: It refers to the current row being evaluated (in Calculated Columns or iterators like SUMX()).

🔹 Q3. Why prefer explicit measures over implicit?

Ans: Explicit measures are reusable, optimized, and easier to maintain in complex reports.

🔹 Q4. What's the use of CALCULATE in DAX?

Ans: It changes the filter context to perform advanced calculations. It's the **most powerful** DAX function.

🔹 Q5. Can you use a measure in a Calculated Column?

Ans: No. Measures depend on filter context, while calculated columns are row context-based.

🔹 Day 5: Power BI DAX Time Intelligence – YTD, MTD, LY, DATESYTD, SAMEPERIODLASTYEAR

Time Intelligence helps you analyze data **over time** – essential for trends, growth, seasonality, and forecasting.

✅ 1. Prerequisites

- Your Date Table must be:
  - **Marked as a Date Table** in Power BI
  - Have a continuous **Date column** (no gaps)

Modeling → Mark as Date Table → Select Date column

✅ 2. YTD (Year-To-Date)

```
Sales_YTD =
CALCULATE(
    SUM(Sales[Revenue]),
    DATESYTD('Date'[Date])
)
```

- Cumulative sum from Jan 1st to current date

✅ 3. MTD (Month-To-Date)

```
Sales_MTD =
CALCULATE(
    SUM(Sales[Revenue]),
    DATESMTD('Date'[Date])
)
```

- From 1st of current month to the current date

✅ 4. QTD (Quarter-To-Date)

```
Sales_QTD =
CALCULATE(
    SUM(Sales[Revenue]),
    DATESQTD('Date'[Date])
)
```

✅ 5. SAMEPERIODLASTYEAR

```
Sales_LY =
CALCULATE(
    SUM(Sales[Revenue]),
    SAMEPERIODLASTYEAR('Date'[Date])
)
```

- Returns values from the same period last year (e.g., Feb 2023 ↔️ Feb 2022)

✅ 6. PARALLELPERIOD

```
Sales_LY_Parallel =
CALCULATE(
    SUM(Sales[Revenue]),
    PARALLELPERIOD('Date'[Date], -1, YEAR)
)
```

- Flexible version to shift time forward/backward

✅ 7. TOTALYTD / TOTALMTD (Shortcut functions)

```
Total_Sales_YTD =
TOTALYTD(SUM(Sales[Revenue]), 'Date'[Date])
```

💬 INTERVIEW QUESTIONS (Medium to High)

🔹 Q1. What's the difference between SAMEPERIODLASTYEAR and DATEADD?

Ans:

- SAMEPERIODLASTYEAR = same period, previous year
- DATEADD = more flexible; you can shift by day, month, quarter, year in any direction

🔹 Q2. Why must we mark a Date Table?

Ans: So Power BI recognizes and correctly uses it in Time Intelligence functions. Otherwise, calculations may fail or return incorrect results.

🔹 Q3. How does CALCULATE work with DATESYTD()?

Ans: CALCULATE modifies the filter context to include all dates from

Jan 1st to the current row's date, then sums over that context.

- **Q4. What's the impact of missing dates in your Date Table?**

Ans: Time intelligence functions may return partial or wrong values because they assume continuity.

- **Q5. Can you do YTD without a Date Table?**

Ans: Technically yes, but it's **not reliable**. Proper time intelligence requires a marked Date table.

- **Day 6: DAX Iterators – SUMX, AVERAGEX, MINX, MAXX, COUNTX**

Iterators are **row-by-row evaluators**. Unlike standard functions like SUM() which work on a column directly, **iterators** allow **custom expressions for each row** before aggregating.

✅ 1. SUMX – Sum with a custom expression

```
Total_Profit =
SUMX(
    Sales,
    Sales[Quantity] * Sales[UnitPrice]
)
```

- Evaluates Quantity * UnitPrice **for each row**, then adds them up.

🧠 Use case: When column isn't pre-calculated in your table but needs computation.

✅ 2. AVERAGEX – Average of a calculated expression

```
Avg_Sales_Per_Unit =
```

```
AVERAGEX(
    Sales,
    Sales[Revenue] / Sales[Quantity]
)
```

- Useful when computing **dynamic averages**.

✅ 3. MINX, MAXX – Minimum/Maximum with logic

```
Lowest_Cost_Product =
MINX(
    Products,
    Products[CostPrice] * 1.05
)
```

- You can apply transformations to values before comparison.

✅ 4. COUNTX – Count non-blank results from an expression

```
Count_Effective_Discounts =
COUNTX(
    Sales,
    IF(Sales[Discount] > 0, 1, BLANK())
)
```

- Powerful for conditional counts.

✅ 5. RANKX – Ranking rows with context

```
Product_Rank =
RANKX(
    ALL(Products),
    [Total_Sales],
    ,
    DESC,
```

Dense
)

- Rank each product based on total sales.

⚠️ **Iterator Performance Tip:**

- Iterators **can slow down** if used on large tables without filters.
- Always prefer **pre-calculated columns** if logic is constant.

💬 **INTERVIEW QUESTIONS (Medium to High)**

🔹 **Q1. When do you use SUMX over SUM?**

Ans: When you need to sum the result of a row-level calculation (e.g., price × quantity) instead of just summing a column.

🔹 **Q2. Why does SUMX give different results than SUM in some cases?**

Ans: Because it evaluates a **custom expression per row**, whereas SUM() directly totals the column values.

🔹 **Q3. What happens if an expression in AVERAGEX returns BLANK?**

Ans: Those blanks are **excluded from the average**.

🔹 **Q4. What's the difference between RANKX and using RANK in Power Query?**

Ans: RANKX is dynamic and changes with slicers; Power Query's rank is static (computed at data load time).

🔹 **Q5. What's the impact of using ALL() inside RANKX()?**

Ans: ALL() removes filters, so you rank globally rather than within a group.

🔹 Day 7: DAX FILTER Function Deep Dive + ALL, ALLEXCEPT, and Context Transition

FILTER is **one of the most powerful and misunderstood functions in DAX**. It's used to build **custom filter contexts** and works great with CALCULATE.

✅ 1. FILTER() – Filter a table with a condition

```
High_Value_Orders =
CALCULATE(
    COUNTROWS(Sales),
    FILTER(Sales, Sales[Revenue] > 1000)
)
```

- Think of FILTER as: **row-by-row table scan that returns only matching rows**.
- **Used within CALCULATE** to change the context.

✅ 2. ALL() – Remove filters

```
Overall_Sales =
CALCULATE(
    SUM(Sales[Revenue]),
    ALL(Sales)
)
```

- Removes all filters from the Sales table.
- Useful when you want **totals regardless of slicers or visuals**.

✅ 3. ALLEXCEPT() – Remove all filters except specified columns

```
Sales_Per_Region =
CALCULATE(
    SUM(Sales[Revenue]),
    ALLEXCEPT(Sales, Sales[Region])
)
```

- Removes all filters **except Region**, so values reflect total sales **per region**, ignoring other slicers.

✅ 4. Context Transition

```
Total_Sales =
CALCULATE(
    SUM(Sales[Revenue])
)
```

- In a row context (like inside a calculated column), **CALCULATE turns it into a filter context**.
- This is called **context transition**.

✅ 5. FILTER vs Boolean Filters in CALCULATE

```
-- Boolean filter (faster)
CALCULATE(SUM(Sales[Revenue]), Sales[Region] = "East")

-- FILTER function (more flexible)
CALCULATE(SUM(Sales[Revenue]), FILTER(Sales, Sales[Region] =
"East"))
```

- Use FILTER() when you need **complex logic**.
- Use boolean filters for **simple conditions** (they're faster).

⚠️ Performance Tips:

- Avoid nesting FILTER unless necessary.
- Combine FILTER with ALL when computing **% of total**.
- FILTER always returns a **table**, not a value.

💬 INTERVIEW QUESTIONS (Medium to High)

🔹 Q1. What's the difference between FILTER and a Boolean filter in CALCULATE?

Ans: Boolean filters are faster and simpler; FILTER is more flexible and can handle complex logic.

🔹 Q2. What does ALL() do inside a measure?

Ans: It removes filters from the specified table or column, allowing for total aggregation.

🔹 Q3. Why does CALCULATE require a table expression like FILTER()?

Ans: CALCULATE modifies context using filter tables, and FILTER returns a table expression.

🔹 Q4. What is context transition in DAX?

Ans: When CALCULATE changes a row context (like in a calculated column) into a filter context so that aggregation functions work correctly.

🔹 Q5. When would you use ALLEXCEPT()?

Ans: When building measures like "% of Region Total" while **keeping one filter and removing others**.

🔹 Day 8: Power BI Visuals – Slicers, Drillthrough, Tooltips, and Custom Visual Enhancements

Visuals are not just for presentation—they can **transform how data tells a story**. Mastering slicers, drillthrough, and tooltips gives you the superpower to create **dynamic, interactive dashboards**.

✅ 1. Slicers – Basic to Advanced Filtering

- Drag a **dimension** (e.g., Region, Date, Category) into a slicer.
- Use dropdown, list, or **between-date** types.

📌 **Tip**: Slicers apply **visual-level filters** across multiple visuals at once.
🔁 **Sync Slicers**: Sync across pages from the View tab → Sync Slicers.

✅ 2. Drillthrough – Contextual Deep-Dive Pages

Let users **right-click → Drillthrough** into another report page.
Steps:

1. Create a new page.
2. Drag Customer ID or Product Name to **Drillthrough Filters pane**.
3. Add visuals.
4. Right-click from any visual to drill through.

📌 Great for building **customer profiles**, **sales breakdowns**, etc.

✅ 3. Report Page Tooltips – Add Mini Reports

- Turn any page into a **tooltip** by:
  - Enabling Tooltip under page settings.
  - Design it with summary cards, charts, etc.
- In visuals, go to Format → Tooltip → Page → Choose your

custom tooltip.

📌 Used for **hover insights**, like showing details when hovering over a bar/point.

✅ 4. Bookmarking + Button Navigation

Create **interactive dashboards** with:

- **Buttons** (Back, Next, Info)
- **Bookmarks** (to store visual states)
- **Action settings** (on click → go to page/bookmark)

📌 Example: Toggle between chart types or views (e.g., Monthly vs Yearly).

✅ 5. Hierarchy Navigation with Drill Down / Drill Up

- Use **+** icon to drill down into date hierarchy (Year → Quarter → Month).
- Turn on "Drill Mode" for seamless navigation.

📌 Important for **temporal analysis** or any multi-level dimension (e.g., Country → State → City).

✅ 6. Dynamic Titles & Cards

Use DAX to **dynamically show selected filters**:

```
Selected_City =
SELECTEDVALUE(Customer[City], "All Cities")
```

Use it in a **Card** to show which city is selected.

💬 INTERVIEW QUESTIONS (Medium to High)

🔹 Q1. How does drillthrough differ from drill down in Power BI?

Ans: Drillthrough navigates to a separate page with detailed insights; drill down happens within a visual's hierarchy.

🔹 Q2. How do report page tooltips improve UX?

Ans: They provide contextual insights on hover without cluttering the main view.

🔹 Q3. Can slicers affect tooltip pages?

Ans: Yes, if the tooltip uses the same fields affected by the slicer.

🔹 Q4. What are some use cases for dynamic titles?

Ans: Show selected filters, make reports more readable and customizable per user selection.

🔹 Q5. Why use bookmarks in Power BI?

Ans: For creating interactive storytelling dashboards, toggles, what-if simulations, or page navigation.

🔹 Day 9: Power BI Modeling – One-to-Many, Many-to-Many, USERELATIONSHIP(), and Relationship Logic

Data modeling is the **backbone of Power BI**—a solid model ensures accurate calculations, meaningful visuals, and better performance.

✅ 1. One-to-Many Relationships (Most Common)

- One side: Dimension table (Date, Product, Customer)
- Many side: Fact table (Sales, Orders, Transactions)

📌 Example: Customer[CustomerID] → Sales[CustomerID]
Helps with filtering and aggregations like:

```
TotalSales = SUM(Sales[Amount])
```

✅ 2. Many-to-Many Relationships (M:N)

When both tables have **duplicates** of the key column.
📌 Example:

- Students and Courses linked via an intermediate table Enrollment.

Power BI handles this with **composite models**, but **be cautious**—can lead to ambiguous relationships or performance issues.

✅ 3. Inactive Relationships & USERELATIONSHIP()

Used when **two tables have multiple logical connections**, but only one can be **active** at a time.
📌 Example:

- Sales[OrderDate] → Date[Date] (Active)
- Sales[ShipDate] → Date[Date] (Inactive)

Use DAX to activate the inactive one:

```
SalesByShipDate =
CALCULATE(
    SUM(Sales[Amount]),
    USERELATIONSHIP(Sales[ShipDate], Date[Date])
)
```

✅ 4. Cross Filter Direction

- **Single**: Filters flow from dimension → fact.
- **Both**: Filters in both directions (used for M:M or slicers on both ends).

📌 Only use **bi-directional filters** when absolutely necessary—may lead to performance or circular reference issues.

✅ 5. Creating Relationships in Model View

- Go to **Model View**.
- Drag fields to link them or use **Manage Relationships**.
- Review relationship cardinality and filter direction.

✅ 6. Role-Playing Dimensions (Date Example)

You can use the **same Date table** for multiple date fields by duplicating it:

Date_Ordered → Sales[OrderDate]
Date_Shipped → Sales[ShipDate]

Manage using **USERELATIONSHIP()** in measures.

💬 INTERVIEW QUESTIONS (Medium to High)

🔹 Q1. What is the difference between one-to-many and many-to-many in Power BI?

Ans: One-to-many has unique values on one side; many-to-many

allows duplicates on both, often using a bridging table.

🔹 **Q2. When would you use USERELATIONSHIP()?**

Ans: To activate an inactive relationship in a specific measure, such as choosing between OrderDate and ShipDate.

🔹 **Q3. What are the risks of bidirectional filtering?**

Ans: Can create circular dependencies and slow performance; should be used sparingly and carefully.

🔹 **Q4. How would you handle role-playing dimensions in Power BI?**

Ans: Duplicate the dimension table and use USERELATIONSHIP() in DAX to control which one is used in a measure.

🔹 **Q5. Why is proper cardinality important in relationships?**

Ans: Ensures correct aggregation, filtering, and model behavior.

🔹 **Day 10: Power BI Performance Tips – Query Reduction, Aggregations, and Optimizing DAX**

As your Power BI reports grow in complexity, **performance tuning** becomes essential. A fast dashboard isn't just user-friendly—it's trusted and used more often.

✅ **1. Query Reduction in Visuals**

Power BI visuals often trigger **multiple queries**. To reduce this:

- Use **fewer visuals** on a page.
- Avoid excessive slicers and filters.
- Use **"Edit Interactions"** to reduce unnecessary cross-filtering.

📌 In Options → Query Reduction:

- Turn on **"Add a single Apply button"** to slicers to avoid triggering queries every time the user clicks.

✅ 2. Avoid SELECTEDVALUE() in Calculated Columns

Why? It evaluates at model load time, which is bad for performance. Use it in **measures only**, and ensure you handle ambiguity with SELECTEDVALUE(column, "default").

✅ 3. Pre-Aggregation with Summary Tables

Instead of calculating everything at runtime, **pre-aggregate** where possible:

From: 10 million rows of transaction data
To: Aggregated table with daily or monthly summaries
This reduces the size of the dataset and the load on DAX engines.

✅ 4. Use Variables in DAX

Helps **avoid repeated calculations**, improving both **readability** and **performance**.

```
SalesMargin =
VAR TotalSales = SUM(Sales[Amount])
VAR Cost = SUM(Sales[Cost])
RETURN
TotalSales - Cost
```

✅ 5. Avoid Complex Nested IFs or SWITCH

These slow down calculations significantly. Prefer logical short-circuits or measures split into simpler components.

📌 Use SWITCH(TRUE(), ...) instead of many nested IFs.

✅ 6. Disable Auto Date/Time (unless needed)

Go to: **File ➔ Options ➔ Data Load ➔ Time Intelligence** ➡️ Uncheck "Auto date/time for new files"

This setting generates hidden tables behind the scenes and slows down large models.

✅ 7. Use Performance Analyzer

💡 Found in **View tab ➔ Performance Analyzer**
This tool helps you:

- See how long each visual takes
- Identify slow queries or measures
- Export performance logs for optimization

💬 INTERVIEW QUESTIONS (Medium to High)

🔹 Q1. What are some best practices for optimizing Power BI performance?

Ans: Use variables in DAX, minimize visuals, pre-aggregate data, avoid SELECTEDVALUE in calculated columns, and disable auto date/time.

🔹 Q2. When would you use Performance Analyzer?

Ans: To debug slow visuals, measure rendering times, and identify heavy queries.

🔹 Q3. Why is it better to use SWITCH(TRUE(), ...) over nested IFs?

Ans: Easier to read, more efficient, avoids DAX engine doing

unnecessary evaluations.

🔹 **Q4. What is the impact of disabling Auto Date/Time?**

Ans: Reduces model size and speeds up refreshes—especially in large datasets.

🔹 **Q5. How does query reduction improve performance?**

Ans: It prevents Power BI from sending multiple queries unnecessarily, leading to faster load times.

🔹 **Day 11: Power BI Time Intelligence – YTD, MTD, PYTD, and Custom Date Logic**

Time-based calculations are **must-haves** in dashboards. They let you compare **Year-to-Date (YTD)**, **Month-to-Date (MTD)**, and **Previous Period** metrics in just a few lines of DAX.

✅ **1. Year-to-Date (YTD)**

```
TotalSalesYTD =
CALCULATE(
    SUM(Sales[Amount]),
    DATESYTD(DimDate[Date])
)
```

🔍 **DATESYTD()** returns all dates from the beginning of the year to the current context.

✅ 2. Month-to-Date (MTD)

```
TotalSalesMTD =
CALCULATE(
    SUM(Sales[Amount]),
    DATESMTD(DimDate[Date])
)
```

🔍 Great for monthly progress and forecasting needs.

✅ 3. Previous Year (PY) & PYTD

```
TotalSalesPY =
CALCULATE(
    SUM(Sales[Amount]),
    SAMEPERIODLASTYEAR(DimDate[Date])
)

TotalSalesPYTD =
CALCULATE(
    SUM(Sales[Amount]),
    DATESYTD(SAMEPERIODLASTYEAR(DimDate[Date]))
)
```

Use it to analyze **YoY growth trends**.

✅ 4. Custom Date Filters

For example: Sales last 30 days

```
Last30DaysSales =
CALCULATE(
    SUM(Sales[Amount]),
    FILTER(
        ALL(DimDate),
        DimDate[Date] >= TODAY() - 30 && DimDate[Date] <= TODAY()
    )
)
```

📌 FILTER(ALL(...)) ensures the full date table is considered.

✅ 5. Cumulative (Running Total)

```
RunningTotal =
CALCULATE(
    SUM(Sales[Amount]),
    FILTER(
        ALLSELECTED(DimDate),
        DimDate[Date] <= MAX(DimDate[Date])
    )
)
```

Use ALLSELECTED if you want to respect slicers and filters.

✅ 6. Dynamic Measures: This Year vs Last Year Growth

```
YoYGrowth =
DIVIDE(
    [TotalSales] - [TotalSalesPY],
    [TotalSalesPY],
    0
)
```

💬 INTERVIEW QUESTIONS (Medium to High)

- **Q1. What's the difference between DATESYTD() and TOTALYTD()?**

Ans: DATESYTD() returns dates; TOTALYTD() combines DATESYTD + aggregation.

- **Q2. How do you calculate running totals in Power BI?**

Ans: Using CALCULATE + FILTER + ALLSELECTED to include all earlier dates.

- **Q3. Why do we use SAMEPERIODLASTYEAR()?**

Ans: For exact period shift (e.g., comparing Feb 2023 to Feb 2022).

- **Q4. What's the need for a Date Table in Time Intelligence?**

Ans: Time functions like DATESYTD need a **continuous date range** to work properly.

- **Q5. How would you calculate a custom "Last N Days" sales measure?**

Ans: Use FILTER(ALL(DimDate), Date >= TODAY() - N && Date <= TODAY()).

🔹 Day 12: Power BI Data Modeling – Relationships, Star vs Snowflake, Cross Filter Direction

Data modeling is the **foundation of Power BI**. A well-designed model improves performance, enables accurate calculations, and simplifies DAX!

✅ 1. Star vs Snowflake Schema

🔸 Star Schema:

- Fact table in the center.
- Connected to denormalized dimension tables.
- Preferred for Power BI.

🔸 Snowflake Schema:

- Dimension tables are normalized (i.e., split into sub-dimensions).
- Slower performance, more joins.

📌 **Interview Tip:** Always recommend Star schema for Power BI unless data size or structure forces normalization.

✅ 2. One-to-Many (1:*) Relationships

Power BI automatically detects and builds relationships. You can also manage them in the **Model View**.

Example:
CustomerID in Sales Table → CustomerID in Customer Table

✔ Keep the *one* side on the dimension table
✔ Keep the *many* side on the fact table

✅ 3. Cross Filter Direction

- **Single**: Default, filters only from dimension → fact
- **Both**: Filters in both directions, used in complex models or for slicers across multiple fact tables

⚠️ Be cautious with *Both* – it can create ambiguity and circular dependencies.

✅ 4. Active vs Inactive Relationships

You can have **multiple relationships** between tables but only one **active** at a time.

Use USERELATIONSHIP() in DAX to activate an inactive one temporarily.

```
TotalAmountByShipDate =
CALCULATE(
    SUM(Sales[Amount]),
    USERELATIONSHIP(Sales[ShipDate], DimDate[Date])
)
```

✅ 5. Role Playing Dimensions (Date)

When a fact table has multiple date fields (Order Date, Ship Date, Delivery Date), use:

- A **single DimDate** table
- Multiple **inactive relationships**
- USERELATIONSHIP() to switch dynamically

✅ 6. Snowflake Case - Should You Flatten?

When loading a snowflake model:
✅ Flatten using Power Query merge
✅ Reduces joins and speeds up visuals

💬 INTERVIEW QUESTIONS (Medium to High)

🔹 Q1. What is the difference between Star and Snowflake Schema?

Ans: Star has denormalized dimensions; Snowflake has normalized (split) dimensions. Star is better for Power BI.

🔹 Q2. What is Cross Filter Direction? When should you use Both?

Ans: Single filters from dimension to fact; Both is used when slicing across multiple fact tables.

🔹 Q3. What are active and inactive relationships?

Ans: Only one relationship is active between two tables. Inactive ones can be used with USERELATIONSHIP().

🔹 Q4. How do you handle multiple date fields with the same date table?

Ans: Use role-playing dimensions with USERELATIONSHIP() in measures.

🔹 Q5. Why are relationships important in DAX?

Ans: Without relationships, DAX functions like CALCULATE and FILTER won't behave as expected.

🔹 Day 13: Power BI Performance Optimization – DAX vs M, Aggregations, and Best Practices

Power BI can handle large datasets—but only if optimized properly. Today we cover practical techniques that help you deliver fast, efficient reports.

✅ 1. DAX vs Power Query (M) – Where to Transform?

Task	Do it in Power Query (M)	Do it in DAX
Data Cleaning / Shaping	✅ Yes	❌ No
Calculated Columns (Static)	✅ Preferred	Possible, but heavy on model
Calculated Measures	❌ Not supported	✅ DAX only

📌 **Rule of Thumb**: Push as much work as possible to Power Query. DAX is for analysis, not ETL.

✅ 2. Use Aggregations to Minimize Data

Instead of loading all transaction-level data, aggregate it in Power Query using **Group By**.

Example:
Group Sales by ProductID and Date → Sum of SalesAmount
✅ Reduces row count
✅ Improves visual performance

✅ 3. Avoid DAX Calculated Columns (Unless Necessary)

DAX columns increase memory usage. If it can be done in Power Query → **do it there.**

✅ 4. Use Measures Instead of Columns in Visuals

❌ Using columns in visuals = pre-aggregated data → slow
✅ Use measures → faster, dynamic, and optimized
Total Sales = SUM(Sales[SalesAmount])

✅ 5. Remove Unused Columns

Only import what's necessary for reporting and calculations. Each column takes space in the in-memory engine (VertiPaq).

✅ 6. Turn Off Auto Date/Time

By default, Power BI creates a hidden date table for every datetime field.
Go to **File > Options > Data Load** → Disable **Auto Date/Time** for faster models.

✅ 7. Cardinality Matters

Lower cardinality = better performance
Examples:

- Use integers for keys
- Avoid long text columns
- Avoid columns with many unique values in visuals

✅ 8. Limit Visuals on One Page

Each visual triggers a DAX query. Too many = slow report!

✅ Keep visuals minimal and focused
✅ Use bookmarks and drillthrough for detail

💬 INTERVIEW QUESTIONS (Medium to High)

🔹 Q1. What's the difference between DAX and M?

Ans: M (Power Query) is used for data transformation. DAX is for aggregations and calculations on already loaded data.

🔹 Q2. When should you prefer Power Query over DAX?

Ans: Always prefer Power Query for data shaping, cleaning, and static columns.

🔹 Q3. How does column cardinality affect performance?

Ans: Higher cardinality = more memory and slower visuals. Keep columns low in unique values where possible.

🔹 Q4. What's better: calculated columns or measures?

Ans: Measures are better. Calculated columns increase memory and can slow down performance.

🔹 Q5. What is VertiPaq?

Ans: Power BI's in-memory storage engine, optimized for columnar data. Compression depends on column cardinality.

🔹 Day 14: Power BI DAX – Time Intelligence (YTD, MTD, SAMEPERIODLASTYEAR)

Time-based analysis is one of Power BI's strongest features. Today, we cover essential **DAX Time Intelligence functions** that unlock powerful date-driven insights.

✅ 1. CALCULATE() – The Base for All Time Intelligence

```
Total Sales YTD = CALCULATE(
    [Total Sales],
    DATESYTD('Date'[Date])
)
```

✅ CALCULATE() modifies the filter context
✅ Required for all time functions (YTD, MTD, etc.)

✅ 2. YTD (Year-to-Date)

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

Used to track cumulative sales or metrics from the start of the year.

✅ 3. MTD (Month-to-Date)

```
Total Sales MTD = CALCULATE(
    SUM(Sales[SalesAmount]),
    DATESMTD('Date'[Date])
)
```

Monitor performance during a specific month, up to the current date.

✅ 4. QTD (Quarter-to-Date)

```
Total Sales QTD = CALCULATE(
    SUM(Sales[SalesAmount]),
    DATESQTD('Date'[Date])
)
```

Similar to MTD/YTD but for the quarter.

✅ 5. SAMEPERIODLASTYEAR

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

Used for YoY comparisons (sales, growth, etc.)

✅ 6. PREVIOUSMONTH / PREVIOUSYEAR

```
Sales Prev Month = CALCULATE(
    [Total Sales],
    PREVIOUSMONTH('Date'[Date])
)
```

✅ Compare KPIs across time frames
✅ Great for performance dashboards

✅ 7. DATEADD() for Custom Time Offsets

```
Sales 3 Months Ago = CALCULATE(
    [Total Sales],
    DATEADD('Date'[Date], -3, MONTH)
)
```

Flexible function to move forward/backward in time.

💬 INTERVIEW QUESTIONS (Medium to High)

🔹 Q1. Why is CALCULATE() required for time intelligence in DAX?

Ans: CALCULATE changes the filter context so DAX can apply date-based filters like DATESYTD or SAMEPERIODLASTYEAR.

🔹 Q2. Difference between SAMEPERIODLASTYEAR and DATEADD()?

Ans: SAMEPERIODLASTYEAR keeps the same date range from the previous year; DATEADD can move any number of units (days, months, years).

🔹 Q3. What's the use of DATESYTD()?

Ans: Returns a table of dates from the start of the year to the current row's date—used in YTD calculations.

🔹 Q4. Can we use MTD or QTD without a proper Date table?

Ans: No. Time intelligence functions require a complete, continuous date table marked as a Date Table in Power BI.

🔹 Q5. What if your fiscal year starts in April?

Ans: DATESYTD and other functions accept a second parameter for

fiscal year end month (e.g., DATESYTD('Date'[Date], "3") for April).
Let's bring it home with a BANG! 💥

🔹 Day 15: Power BI Interview Questions – Medium to High (Rapid Fire Round 💼)

Here's your **Power BI Final Round** prep: real-world, scenario-based interview questions that recruiters *love* to ask. Let's get you **interview-ready** 🔥

✅ 1. What's the difference between Calculated Column and Measure?

- **Calculated Column**: Evaluated row-by-row in the data model. Stored physically.
- **Measure**: Calculated on aggregation level (e.g., sum, average), computed on the fly.

🧠 Use columns for filters/slicers, measures for visualizations.

✅ 2. How do you handle performance issues in Power BI reports?

- Optimize DAX formulas
- Use star schema
- Reduce cardinality
- Use aggregations
- Avoid unnecessary visuals
- Use **Performance Analyzer**

✅ 3. What is Row-Level Security (RLS)? How do you implement it?

- **RLS** restricts data access based on user roles.
- Implement via:
  - **Manage Roles** in Power BI Desktop
  - Define DAX filters (e.g., [Region] = USERNAME())
  - Publish to Service → Assign roles

✅ 4. What's the difference between FILTER() and CALCULATE()?

- FILTER() returns a table → often used inside CALCULATE()
- CALCULATE() modifies the filter context to evaluate an expression

🧠 CALCULATE(SUM(Sales[Amount]), FILTER(...)) is a classic pattern.

✅ 5. What is a Fact Table vs Dimension Table?

- **Fact**: Transactional data (sales, revenue)
- **Dimension**: Descriptive attributes (product, date, region)
- Use **star schema** for optimized relationships.

✅ 6. How do you handle many-to-many relationships in Power BI?

- Use a **bridge table**
- Use **COMPOSITE models** or TREATAS() in DAX

✅ 7. When would you use ALL() and REMOVEFILTERS()?

- ALL() removes all filters from a column/table
- REMOVEFILTERS() works similarly but is more explicit in behavior

✅ Often used in % of total / grand total scenarios.

✅ 8. Difference: DirectQuery vs Import Mode?

Feature	Import	DirectQuery
Speed	Fast	Slower
Data Storage	In-memory	On-demand
Real-time?	No (needs refresh)	Yes
DAX Limitations	No	Yes

✅ 9. How do you create dynamic titles in Power BI?

- Create a **DAX measure** for title:

```
Title = "Sales Report - " & SELECTEDVALUE('Date'[Year])
```

- Add it to a **card visual** and hide axis titles.

✅ 10. Explain DAX Function: RANKX()

```
Rank_Product = RANKX(
    ALL('Product'),
    [Total Sales],
    ,
    DESC,
    Dense
)
```

Ranks products based on total sales.
Used in leaderboard visuals, KPI tracking.

💼 Bonus High-Level Scenario-Based Questions

🔹 Q: How do you build a Power BI report from scratch?

1. Connect to data
2. Clean/transform in Power Query
3. Model data (relationships, calculated columns)
4. Create visuals
5. Add slicers, filters, KPIs
6. Publish to Power BI Service
7. Set refresh schedules, RLS if needed

🔹 Q: How do you deal with data from multiple sources?

- Use Power Query to merge/append
- Create a **data model** with clear relationships
- Ensure consistency in key fields

- **Q: What KPIs have you implemented in past dashboards?**
  - Sales Growth %, Customer Churn, Revenue per Region
  - Conversion Rate, Avg Handling Time (AHT), NPS

📚 Congratulations!

You've now completed the **15-Day Power BI Interview Guide** 🚀
From **basic visuals** to **advanced DAX and performance tuning**, you've built the confidence to crack interviews at **EY, Accenture, PwC, PhysicsWallah, Zepto, Walmart, Genpact** and more.