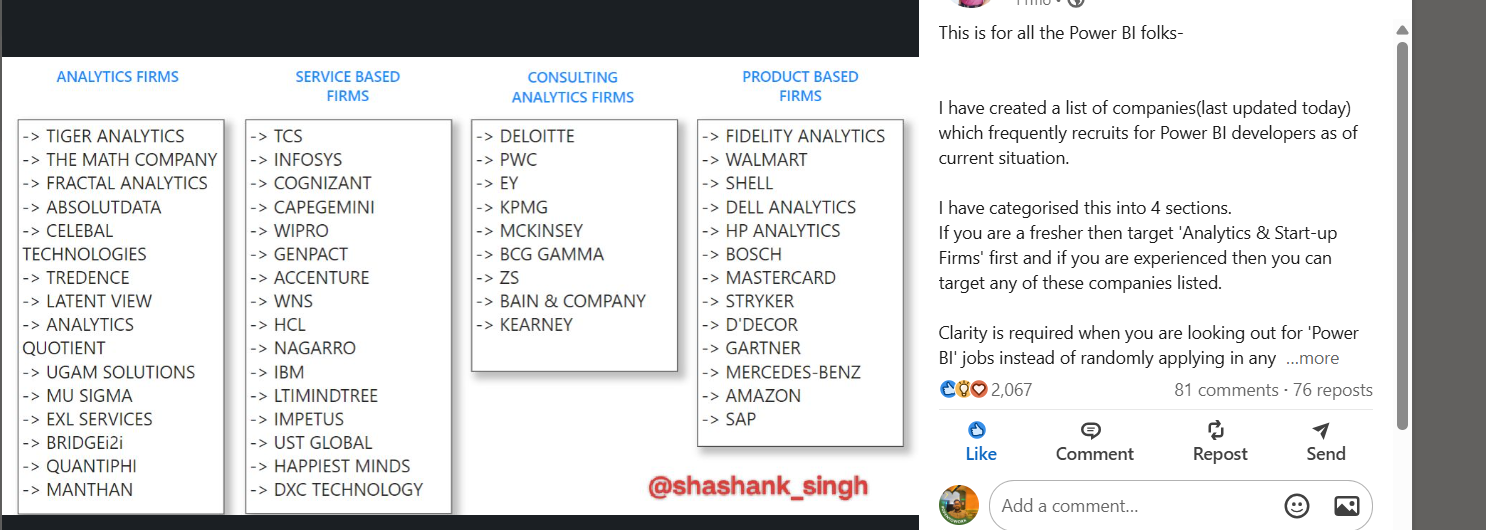
<https://chatgpt.com/share/685316ac-f2c8-8008-a1e2-34046de0a0f3>

Study for All Time Intelligence dax function



**INTERVIEW QUESTIONS:**

* **STAR VS SNOWFLAKE Allexpect, allselected, all dax function**
* **IMPORT VS DIRECT QUERY MODE**
* **ROLES**
* **RLS can be implemented on contributer side**

**Power BI offers a diverse array of visualizations tailored to various data types and analytical needs. Below is a comprehensive list of commonly used Power BI visuals, their typical use cases, and illustrative examples:(**[**Analytics Vidhya**](https://www.analyticsvidhya.com/blog/2024/08/top-power-bi-charts/?utm_source=chatgpt.com)**)**

**📊 Standard Power BI Visuals and Their Use Cases**

**1. Bar and Column Charts**

* **Use Case: Compare values across categories.**
* **Example: Sales by region or product category.**
* **Variants: Clustered, Stacked, and 100% Stacked.(**[**Power UI**](https://www.powerui.com/visuals-reference?utm_source=chatgpt.com)**,** [**Analytics Vidhya**](https://www.analyticsvidhya.com/blog/2024/08/top-power-bi-charts/?utm_source=chatgpt.com)**)**

**2. Line and Area Charts**

* **Use Case: Display trends over time.**
* **Example: Monthly revenue or website traffic.**
* **Variants: Line, Area, Stacked Area, and 100% Stacked Area.(**[**Power UI**](https://www.powerui.com/visuals-reference?utm_source=chatgpt.com)**)**

**3. Pie and Donut Charts**

* **Use Case: Show proportions of a whole.**
* **Example: Market share by company.**
* **Note: Donut charts can display additional information in the center.(**[**Medium**](https://medium.com/%40powerbicourses/basic-visualizations-an-overview-of-standard-visuals-in-power-bi-d23e11323d4c?utm_source=chatgpt.com)**)**

**4. Treemap**

* **Use Case: Visualize hierarchical data with part-to-whole relationships.**
* **Example: Sales by region and sub-region.(**[**Vision Analytics**](https://www.visionanalytics.io/blog/top-10-power-bi-visuals-you-need-to-use/?utm_source=chatgpt.com)**)**

**5. Waterfall Chart**

* **Use Case: Illustrate cumulative effects of sequential positive and negative values.**
* **Example: Profit and loss statements.(**[**Analytics Vidhya**](https://www.analyticsvidhya.com/blog/2024/08/top-power-bi-charts/?utm_source=chatgpt.com)**,** [**Vision Analytics**](https://www.visionanalytics.io/blog/top-10-power-bi-visuals-you-need-to-use/?utm_source=chatgpt.com)**)**

**6. Funnel Chart**

* **Use Case: Represent stages in a process.**
* **Example: Sales pipeline from leads to closed deals.(**[**Power UI**](https://www.powerui.com/visuals-reference?utm_source=chatgpt.com)**)**

**7. Gauge and KPI**

* **Use Case: Display progress toward a goal.**
* **Example: Monthly sales target achievement.(**[**Microsoft Learn**](https://learn.microsoft.com/en-us/power-bi/visuals/power-bi-visualization-types-for-reports-and-q-and-a?utm_source=chatgpt.com)**,** [**Medium**](https://medium.com/%40powerbicourses/basic-visualizations-an-overview-of-standard-visuals-in-power-bi-d23e11323d4c?utm_source=chatgpt.com)**)**

**8. Cards (Single and Multi-row)**

* **Use Case: Highlight key metrics.**
* **Example: Total revenue or number of active users.(**[**Power UI**](https://www.powerui.com/visuals-reference?utm_source=chatgpt.com)**)**

**9. Table and Matrix**

* **Use Case: Present detailed data in rows and columns.**
* **Example: Transaction details or sales by product and region.(**[**Microsoft Learn**](https://learn.microsoft.com/en-us/power-bi/visuals/power-bi-visualization-types-for-reports-and-q-and-a?utm_source=chatgpt.com)**,** [**Vision Analytics**](https://www.visionanalytics.io/blog/top-10-power-bi-visuals-you-need-to-use/?utm_source=chatgpt.com)**)**

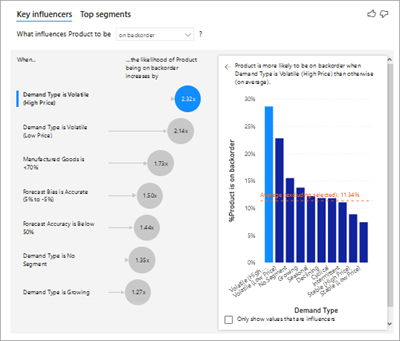
**10. Scatter and Bubble Charts**

* **Use Case: Identify relationships between variables and detect outliers.**
* **Example: Advertising spend vs. sales revenue.(**[**Vision Analytics**](https://www.visionanalytics.io/blog/top-10-power-bi-visuals-you-need-to-use/?utm_source=chatgpt.com)**,** [**Medium**](https://medium.com/%40powerbicourses/basic-visualizations-an-overview-of-standard-visuals-in-power-bi-d23e11323d4c?utm_source=chatgpt.com)**)**

**11. Decomposition Tree**

* **Use Case: Break down a measure into its contributing factors.**
* **Example: Analyzing revenue by region, product, and salesperson.(**[**Power UI**](https://www.powerui.com/visuals-reference?utm_source=chatgpt.com)**)**

**12. Key Influencers**

* **Use Case: Identify factors that influence a particular metric.**
* **Example: Factors affecting customer churn.(**[**Microsoft Learn**](https://learn.microsoft.com/en-us/power-bi/visuals/power-bi-visualization-types-for-reports-and-q-and-a?utm_source=chatgpt.com)**)**
* 
* A key influencer chart displays the major contributors to a selected result or value.
* Key influencers are a great choice to help you understand the factors that influence a key metric. For example, *what influences customers to place a second order* or *why were sales so high last June*.

**13. Maps (Basic, Filled, Shape, ArcGIS, Azure)**

* **Use Case: Display geographical data.**
* **Example: Sales by country or city.(**[**Microsoft Learn**](https://learn.microsoft.com/en-us/power-bi/visuals/power-bi-visualization-types-for-reports-and-q-and-a?utm_source=chatgpt.com)**)**

**14. Ribbon Chart**

* **Use Case: Show rank changes over time.**
* **Example: Top-selling products by month.(**[**Power UI**](https://www.powerui.com/visuals-reference?utm_source=chatgpt.com)**)**

**15. Slicers**

* **Use Case: Filter data interactively.**
* **Example: Filter reports by date range or product category.(**[**Fabriconlake**](https://fabriconelake.com/demystifying-data-visualization-types-in-power-bi-unleash-the-power-of-visual-insights/?utm_source=chatgpt.com)**)**

**16. Q&A Visual**

* **Use Case: Allow users to ask natural language questions.**
* **Example: "What were the sales last quarter?"(**[**Power UI**](https://www.powerui.com/visuals-reference?utm_source=chatgpt.com)**,** [**Microsoft Learn**](https://learn.microsoft.com/en-us/power-bi/visuals/power-bi-visualization-types-for-reports-and-q-and-a?utm_source=chatgpt.com)**)**

**17. Smart Narrative**

* **Use Case: Provide automated textual summaries of visuals.**
* **Example: Narrative explanation of sales trends.(**[**SSW - Enterprise Software Development**](https://www.ssw.com.au/rules/power-bi-visualizations/?utm_source=chatgpt.com)**)**

**18. R and Python Visuals**

* **Use Case: Create custom visuals using R or Python scripts.**
* **Example: Advanced statistical analyses or machine learning models.(**[**Power UI**](https://www.powerui.com/visuals-reference?utm_source=chatgpt.com)**)**

**🧩 Custom Visuals**

**19. Word Cloud**

* **Use Case: Visualize the frequency of words in text data.**
* **Example: Customer feedback analysis.(**[**Analytics Vidhya**](https://www.analyticsvidhya.com/blog/2022/09/top-5-custom-visuals-in-power-bi/?utm_source=chatgpt.com)**)**

**20. Sunburst Chart**

* **Use Case: Display hierarchical data in concentric circles.**
* **Example: Organization structure visualization.(**[**SSW - Enterprise Software Development**](https://www.ssw.com.au/rules/power-bi-visualizations/?utm_source=chatgpt.com)**)**

**21. Timeline Slicer**

* **Use Case: Filter data over time using a visual timeline.**
* **Example: Select data for specific months or years.**

**For a visual reference of these charts, you can explore the** [**Power BI Visuals Reference**](https://www.powerui.com/visuals-reference) **or the** [**Power BI Visualization Types**](https://learn.microsoft.com/en-us/power-bi/visuals/power-bi-visualization-types-for-reports-and-q-and-a) **provided by Microsoft.(**[**Microsoft Learn**](https://learn.microsoft.com/en-us/power-bi/visuals/power-bi-visualization-types-for-reports-and-q-and-a?utm_source=chatgpt.com)**)**

**If you need further assistance in selecting the appropriate visual for your specific dataset or analysis goal, feel free to ask!**

**30**

**Main Motivation Behind Job Change :**

**Money 🡪 now my salary is 45k, I am not able to fulfill my dream with this salary. So I need good salary to fulfill my dream (bike 🡪 Hunter 350. As soon As I will Hunter 350 , I will start making content, also buy go pro then I shot videos , then upload those videos on social media and this social media will grow and that brings me some passive income opportunity.**

**growth🡪 Right now my job is very boring and there is no learning it makes me frustrated also. So new job role will bring a opportunity to learn new skill and exposure which help me to grow my knowledge. In this way I will be able to crack new job..**

**Intro:**

Hello my name is Anjan Dey, Completed BTech in Computer Science in 2022 from JIS College of Engineering. After That I joined persistent systems as an intern on January 2022 and converted to full time on July 2022. For last two and half years I am working as Power BI developer.

I am specialized in data modelling, DAX, Power Query (M), and designing interactive dashboards that turn raw data into actionable insights.

I’ve worked extensively with SQL, Power BI Service features like Row Level Security, Incremental Refresh, and automated email alerts.

I have done the development of a Logistics Analysis Dashboard that improved reporting efficiency by 35% and significantly reduced dashboard load time.

=============================

Project : **Logistics Analysis Dashboard Project (Aug 2023 – Present)**

* In my current role, I worked on a Logistics Analysis Dashboard.
* I designed a robust data model by integrating data from over different sources, including Excel files, SQL databases,
* *I implemented 20+ complex DAX measures to track critical KPIs such as delivery time, shipment accuracy, delayed shipment, and route efficiency."*
* **I have enabled drill-down and drill-through capabilities, which allowed users to analyze data at multiple levels—from high-level KPIs to individual shipment details.**
* "To ensure faster loading and real-time updates, I enabled Incremental Refresh(import), which significantly reduced dashboard load time by 50%."
* *I also set* Email SubscriptionSend copy of report to  *key stakeholders*

**Financial Analysis Dashboard**

* In this project, I developed and deployed a financial dashboard for an enterprise-level client,
* I integrated data from both SQL Server and Excel,
* "I used DAX and time intelligence functions to track metrics like **profit margins, cost trends, across different fiscal quarters,** enabling better forecasting and financial planning.
* I implemented **Row Level Security (RLS)** to ensure department-specific access and handled **scheduled refreshes**
* collaborated closely with finance teams, helped troubleshoot issues, conducted user training sessions, and created technical documentation to support ongoing maintenance and adoption.

**Comparing Power BI License Types**

| **Feature** | **Free** | **Pro ($14/user)** | **Premium Per User (PPU) ($24/user)** | **Premium Capacity (P1 SKU\*)** | **Embedded (A1 SKU\*\*)** | **Microsoft Fabric (F SKU\*\*\*)** |
| --- | --- | --- | --- | --- | --- | --- |
| **Licensing Model** | Per individual | Per individual | Per individual | Shared capacity (org/team) | Shared capacity (for apps) | Shared capacity (Azure-based) |
| **Monthly Cost (USD)** | $0 | $14/user | $24/user | ~$4,995/month (P1) | ~$735/month (A1) | From $156/mo (F2), $5,003 (F64) |
| **Storage Limit** | 1 GB (personal) | 10 GB/user | 100 TB (org) | 100 TB (org) | Varies by SKU | 100 TB + OneLake storage |
| **Max Model Size** | 1 GB | 1 GB | 100 GB | Up to 400 GB (P5) | Varies | Varies (F64 = 25 GB, F512 = 200 GB) |
| **Data Refresh Per Day** | Manual / Limited | 8 per dataset | 48 per dataset | 48 per dataset | 48 per dataset | 48 per dataset |
| **Sharing & Collaboration** | ❌ No | ✅ With Pro users | ✅ With PPU/Premium users | ✅ Org-wide | ❌ (only via API) | ✅ (F64+ share with free users) |
| **Publish to Shared Workspaces** | ❌ No | ✅ Yes | ✅ Yes | ✅ Yes | ❌ No | ✅ Yes |
| **View Reports Made by Others** | ❌ Own only | ✅ (Premium/Fabric) | ✅ (Premium/Fabric) | ✅ Yes | ✅ In embedded apps | ✅ Viewer role needed (F64+) |
| **Premium Features (AI, Paginated)** | ❌ No | ❌ No | ✅ Yes | ✅ Yes | ⚠️ Some (varies) | ✅ Yes (varies by SKU) |
| **Microsoft Fabric Workloads** | ❌ No | ❌ No | ❌ No | ❌ No | ❌ No | ✅ Fully supported |
| **Copilot in Fabric** | ❌ Not available | ❌ Not available | ❌ Not available | ❌ Not available | ❌ Not available | ✅ Available (with supported SKU) |
| **On-Prem Report Server** | ❌ No | ❌ No | ❌ No | ✅ Yes (dual use) | ✅ (SQL Server Ent+SA) | ✅ (F64+ Reserved or S |

**what is power bi workspace**

A **Power BI Workspace** is like a **container or folder** in the **Power BI Service (app.powerbi.com)** where you:

* Publish and store your reports and dashboards
* Manage datasets, dataflows, and workbooks
* Share content with other users or teams
* Set permissions and access levels
* Schedule data refresh, create apps, and manage subscriptions

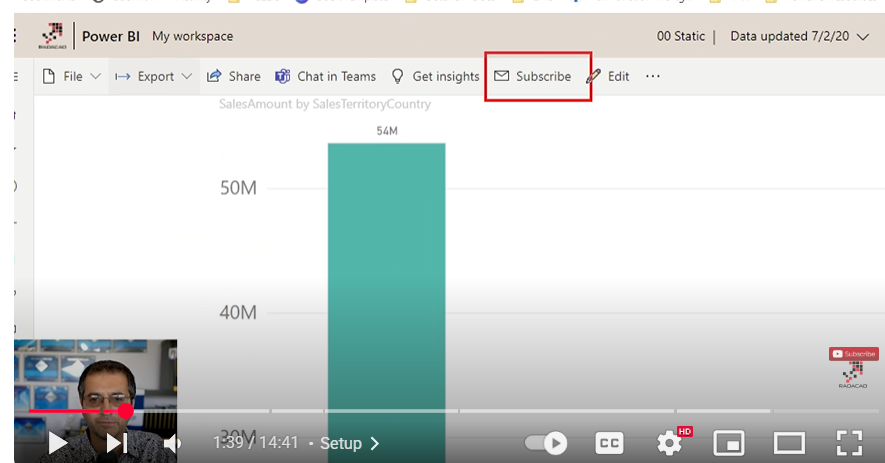
**Power bi Email Subscription:**

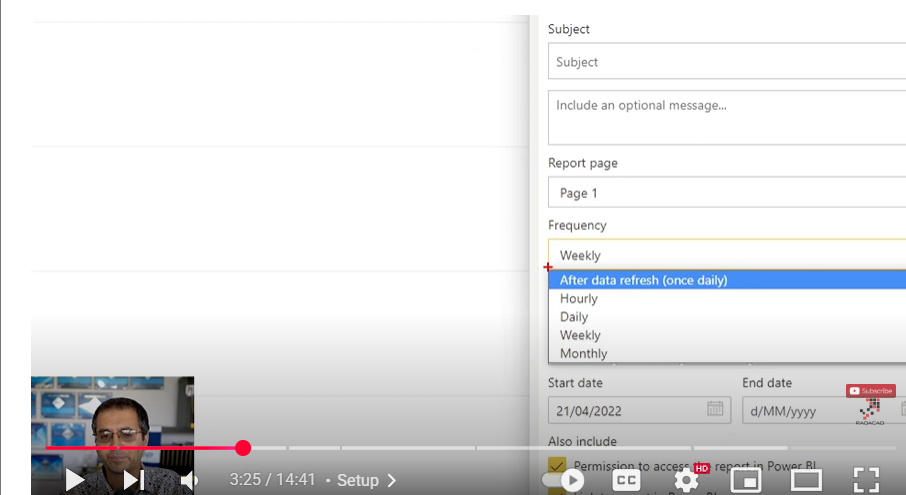
[**https://radacad.com/power-bi-subscription-get-report-updates-in-email**](https://radacad.com/power-bi-subscription-get-report-updates-in-email)

[**https://www.youtube.com/watch?v=f\_Y471pKWh0**](https://www.youtube.com/watch?v=f_Y471pKWh0)

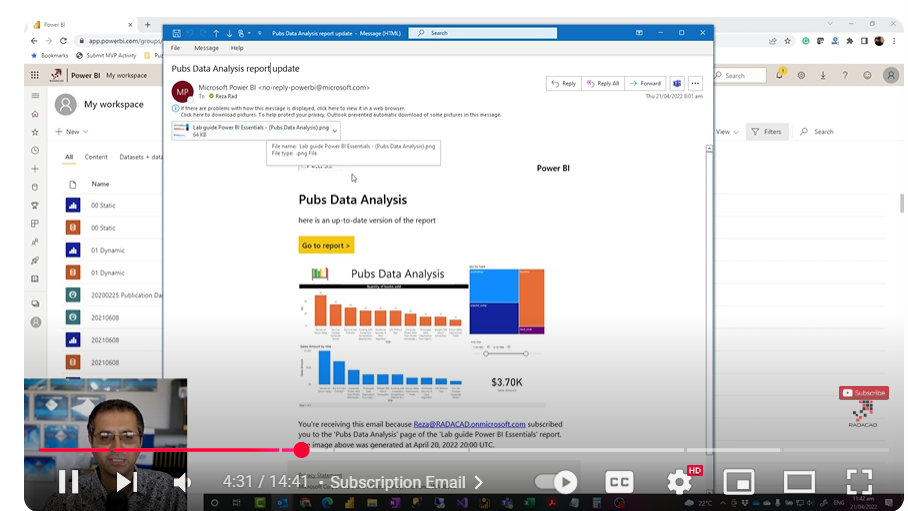
Send copy of report to users.

Open the report from workspace -🡪 click on subscribe 🡪 add new subscription🡪 fill the details like name of subscription, add other people email id , subject , email body , choose frequency (daily , weekly, monthly, hourly, after data refresh(once daily )





**Let see how subscription email look like:**

****

**🌐 What is Power BI Service?**

**Power BI Service** is the **cloud-based platform** (available at https://app.powerbi.com) where you:

* **Publish**, **view**, and **share** Power BI reports and dashboards
* Schedule **data refreshes**
* Create **workspaces** and **apps**
* Set up **Row Level Security**, **email alerts**, and **subscriptions**
* Collaborate with others in your organization

**📌 Think of it as:**

**“The online home of your Power BI content.”**  
You build reports in **Power BI Desktop**, and then publish them to **Power BI Service** for others to access, interact with, and use in real time.

**💼 What Can You Do in Power BI Service?**

| **Feature** | **Description** |
| --- | --- |
| ✅ View & Interact | Use filters, slicers, drilldowns, etc. |
| 🔄 Data Refresh | Schedule auto-refresh for imported datasets |
| 👥 Share & Collaborate | Share reports with teams or create apps |
| 📬 Subscriptions | Set up email delivery of reports |
| 🔐 Security | Apply Row Level Security (RLS) |
| ⚙️ Manage Datasets | Edit credentials, g |

**what is Row Level Security :** [**https://www.youtube.com/watch?v=gT5-S3raIDo**](https://www.youtube.com/watch?v=gT5-S3raIDo)

[**https://www.youtube.com/watch?v=r5XCpeQxXl4**](https://www.youtube.com/watch?v=r5XCpeQxXl4)

[**https://www.youtube.com/watch?v=r5XCpeQxXl4**](https://www.youtube.com/watch?v=r5XCpeQxXl4)

Row-Level Security (RLS) in Power BI is a feature that allows you to restrict data access for users based on their roles. This ensures that each user sees only the data they are authorized to view.

Types of RLS in Power BI

1. Static RLS – Hardcoded rules, where users are manually assigned roles.
2. Dynamic RLS – Uses a lookup table (security table) to assign roles dynamically.
3. Hybrid RLS – A mix of both static and dynamic RLS.

🔹 How to Implement RLS in Power BI?

📌 Step 1: Create Roles in Power BI Desktop

1. Open Power BI Desktop.
2. Go to Modeling > Manage Roles.
3. Click Create and define the role using DAX expressions.
   * Example: If you have a Sales table with a Region column, you can filter it like this:

[Region] = "West"

1. Save the role.

📌 Step 2: Assign Users to Roles in Power BI Service

1. Publish the report to Power BI Service.
2. Navigate to Dataset > Click on "Security".
3. Assign Azure Active Directory (AAD) users or groups to the roles.

**Example Scenarios**

|  |  |
| --- | --- |
| **Scenario** | **Implementation** |
| Only sales managers should see their own region's data | Apply RLS with a filter on the Region column |
| Users should only see their own department's data dynamically | Use a Security Table with USERPRINCIPALNAME() function |
| CEO should see all data, but employees see only their own sales | Assign CEO to a "No Filter" role, while others have filtered views |

**Testing RLS in Power BI**

1. In Power BI Desktop, go to Modeling > View as Roles.
2. Select a role and click OK to see how data is filtered.
3. In Power BI Service, ask users to test their access**.**

**Time Intelligence Function:**

**TotalYTD = Sum from Jan 1st to current date of the year. If you are in 2025 it gives you sum of 1st jan 2025 to today date means last date of data column in your model.**

**TOTALMTD() = Sum from 1st of the month to current date**

**TOTALQTD() = Sum from start of quarter to current date**

**TOTALYTD([total\_Sales]<dates\_column>)**

**TOTALMTD(<expression>, <dates\_column>)**

**TOTALQTD(<expression>, <dates\_column>)**

**DATESMTD() in DAX (Power BI): DATESMTD() is a time intelligence function in DAX that returns a column of dates from the beginning of the month to the current (latest) date in the filter context.**

**It’s mostly used as a date filter in combination with functions like CALCULATE() to get Month-To-Date (MTD) values**

**DATESMTD(<dates>)**

**Imagine your date table has all dates of the year, and today is March 10th**

**Sales MTD =**

**CALCULATE(**

**SUM(Sales[Amount]),**

**DATESMTD(Date[Date])**

**)**

**What happens here:**

* **DATESMTD(Date[Date]) returns March 1 to March 10 (i.e., MTD range).**
* **CALCULATE() sums the sales that happened between those dates**

| * **Function** | * **Description** |
| --- | --- |
| * **DATESMTD()** | * **Returns the list of dates (to use inside CALCULATE)** |
| * **TOTALMTD()** | * **Automatically calculates the MTD sum** |

**Sales MTD =**

* **TOTALMTD(SUM(Sales[Amount]), Date[Date])**
* Is **functionally similar** to:
* Sales MTD =
* CALCULATE(
* SUM(Sales[Amount]),
* DATESMTD(Date[Date])
* ) **Summary:**
* Use DATESMTD() inside CALCULATE() to filter for dates from the start of the month to the current date.
* It’s great when you want **more flexibility** than TOTALMTD() allows.

**Calculate Previous month sales:**

**16)Difference between TotalYTD Vs DatesYtd**

**TotalYTD 🡪 Return single scaler date value. We can add only one filter**

**DatesYTD-> Return Tables of Dates value, as we have to write in Calculate function so we can add more function.**

**18) check the this dax code : calculate(countrows(sales),[Total\_sales\_Dax]>500)**

**This dax code give you error because we can not write measure in the filter part of calculate function.**

**If we want to write a measure in the calculate function then we have to write it inside the FILTER() expression**

**The right dax code is calculate (countrows(sales), filter(sale, [total\_sales\_dax] >500))**

**Implement one scenario : you choose one value from slicer, that selected value will be printed on a card.**

**29) What is different type of tooltip**

[**https://learn.microsoft.com/en-us/power-bi/create-reports/desktop-tooltips**](https://learn.microsoft.com/en-us/power-bi/create-reports/desktop-tooltips)

[**https://www.youtube.com/watch?v=9R59NB3hVQQ&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=29**](https://www.youtube.com/watch?v=9R59NB3hVQQ&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=29)

**In Power BI, Tooltips provide additional context when users hover over a visual. They enhance the interactivity of reports by displaying extra details without cluttering the main visuals.**

**Types of Tooltips in Power BI**

1. **Default Tooltips – Automatically generated based on fields in the visual.**
2. **Report Page Tooltips – Custom-designed tooltips using separate report pages.**
3. **Modern Visual Tooltips – Enhanced tooltips with extra interactivity, allowing drill-through actions.**

**1. Default Tooltips**

By default, when you hover over a visual, Power BI shows a small tooltip with values based on the data fields used in that visual.

How to Customize Default Tooltips:

* Click on the visual.
* Go to the Format Pane > Expand Tooltip.
* You can choose:
  + Auto (default behavior).
  + Fields (to manually select which fields appear in the tooltip).

2. Report Page Tooltips (Custom Tooltips)

Instead of the default tooltip, you can create a custom tooltip page with any visuals you want.

How to Create a Report Page Tooltip:

1. Create a new in page Power BI.
2. Go to Page Settings (Format Pane) and set:
   * Page Size: Change to *Tooltip* (320x240 by default).
   * Tooltip: Set to *On*.
3. Add visuals to the tooltip page.
4. Go back to your original visual, select it, and in the Format Pane > Tooltip, choose:
   * Type: Report Page.
   * Page: Select the tooltip page you created.
5. Hover over the visual, and the custom tooltip will appear.

Benefits:

✅ More detailed insights without taking up space.  
✅ Can include charts, KPIs, or other visuals.  
✅ Provides a better user experience.

3. Modern Visual Tooltips

Modern tooltips provide richer interactivity and support drill-through actions.

How to Enable Modern Tooltips:

* Go to File > Options and Settings > Options > Report settings.
* Enable Modern Visual Tooltips.
* Now, when hovering over a visual, users can see more details and even drill through to another report page.

Best Practices for Tooltips

✅ Keep them concise – Show only necessary information.  
✅ Use visuals wisely – Small bar charts or KPIs work well in tooltips.  
✅ Maintain consistency – Use a uniform tooltip style across the report.  
✅ Optimize performance – Too many complex visuals in tooltips can slow down reports.

**Would you like a hands-on example with a sample dataset? 🚀**

**Interview Question Series:**

1. **Deloitte: select row that only appear in first table 🡪**

**Approach 1: merge the table and select left anti join**

**Approach 2: using Except Dax. Create new Table using Except function.**

**Except -> Returns the rows of the first table in the expression which do not appear in the second table.**

[**https://learn.microsoft.com/en-us/dax/except-function-dax**](https://learn.microsoft.com/en-us/dax/except-function-dax)

[**https://www.youtube.com/watch?v=MoaDV9b5NAc&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza**](https://www.youtube.com/watch?v=MoaDV9b5NAc&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza)

1. Find the revenue by red color product using calculatetable dax fnction

In **DAX (Data Analysis Expressions)**, CALCULATE() and CALCULATETABLE() are used to modify and apply filters to calculations, but they have key differences in their outputs and usage.

**1️⃣ CALCULATE()**

🔹 Returns a **single scalar value** (number, date, or text).  
🔹 Used in **measures or calculated columns** to modify the context of an aggregation.

**Syntax:**

DAX

CopyEdit

CALCULATE(<expression>, <filter1>, <filter2>, ...)

* <expression> → Any aggregation (SUM, AVERAGE, COUNT, etc.).
* <filter1>, <filter2> → Conditions to modify the calculation.

**Example:**

👉 **Total Sales for 2023**

DAX

CopyEdit

TotalSales2023 = CALCULATE(SUM(Sales[Amount]), Sales[Year] = 2023)

✔ SUM(Sales[Amount]) calculates total sales.  
✔ Sales[Year] = 2023 filters the data for the year 2023.

**2️ CALCULATETABLE()**

🔹 Returns a **table** instead of a single value.  
🔹 Used in **table functions, virtual tables, and for advanced filtering.**

**Syntax:**

DAX

CopyEdit

CALCULATETABLE(<table>, <filter1>, <filter2>, ...)

* <table> → A table or filtered subset of a table.
* <filter1>, <filter2> → Conditions to modify the table.

**🔑 Key Differences**

| **Feature** | **CALCULATE()** | **CALCULATETABLE()** |
| --- | --- | --- |
| **Returns** | A single value (scalar) | A table |
| **Use Case** | Aggregations (SUM, COUNT, AVERAGE) | Filtering tables |
| **Works in** | Measures, calculated columns | Table calculations |
| **Output Type** | Number, text, or date | A new table |

**When to Use?**

* ✅ Use **CALCULATE()** when you need a **single value** with modified filters (for KPIs, measures, etc.).
* ✅ Use **CALCULATETABLE()** when you need a **filtered table** for further calculations.

**6)**

**Walmart:**

**You have two table. All the row of 2nd table will be appeared for the all the record of 1st table**

**Approach 1: we can do cross join** [**https://learn.microsoft.com/en-us/dax/crossjoin-function-dax**](https://learn.microsoft.com/en-us/dax/crossjoin-function-dax)

7)

Deloitte part 2

Find the number of specific text appear in how many rows.

**Approach** count of anjan = CALCULATE(COUNT(Sheet1[Column1]), SEARCH("Anjan", Sheet1[Column1],1,0))

<https://learn.microsoft.com/en-us/dax/search-function-dax>

<https://www.youtube.com/watch?v=9cekOOJMcog&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=7>

8) <https://www.youtube.com/watch?v=7SMTiPhXVFg&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=8>

**9) TCS**

**If we don’t select any country from slicer , show sales of india.**

[**https://www.youtube.com/watch?v=pqbvnHj57Es&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=9**](https://www.youtube.com/watch?v=pqbvnHj57Es&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=9)

[**https://learn.microsoft.com/en-us/dax/selectedvalue-function-dax**](https://learn.microsoft.com/en-us/dax/selectedvalue-function-dax)

**10) count networking days between order date and ship date**

[**https://www.youtube.com/watch?v=ZmGuBnndopA&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=10**](https://www.youtube.com/watch?v=ZmGuBnndopA&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=10)

13) sort data on the basis of column:

<https://www.youtube.com/watch?v=IJYCzFc79QE&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=13>

14) **Scenario to calculate DYNAMICALLY TOP 'N' values**

<https://www.youtube.com/watch?v=7scXymOC88o&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=14>

46)Adobe <https://www.youtube.com/watch?v=tVxIzo_sfbE&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=46>

concate 3 column

49) hexaware

Format the contains of the column.

<https://www.youtube.com/watch?v=jvEwoC6bjLY&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=49>

<https://learn.microsoft.com/en-us/dax/format-function-dax>

FORMAT(<value>, <format\_string>[, <locale\_name>])

| **Term** | **Definition** |
| --- | --- |
| value | A value or expression that evaluates to a single value. |
| format\_string | A string with the formatting template. |
| locale\_name | (Optional) Name of the locale to be used by the function. Possible values are strings accepted by the Windows API function [LocaleNameToLCID()](https://learn.microsoft.com/en-us/windows/win32/api/winnls/nf-winnls-localenametolcid). |

**56) Accenture interview Question: (due)**

**1 ) how do you optimize this dax in power bi: Result = a\*b/c\*d**

**Sol :**

**Create measure**

**Result =  
 var a=10, var b=20, var c= 30, var d = 40,**

**Var e = a\*b, var f = c\*d, var g = divide(e,f)**

**Return g;**

1. **Difference between star schema and snowflake schema :**

[**https://www.geeksforgeeks.org/difference-between-star-schema-and-snowflake-schema/**](https://www.geeksforgeeks.org/difference-between-star-schema-and-snowflake-schema/)

1. **What is dataflow in power bi (Due)**

**4) different refresh:**

5) Challenges facing while developing power bi projects

6) **Field Parameter**

[**https://www.youtube.com/watch?v=SpxWzlH-l6Y&t=6s**](https://www.youtube.com/watch?v=SpxWzlH-l6Y&t=6s)

**: A Field Parameter is a dynamic selector that lets users choose which columns or measures to display or analyze at runtime, without editing the report.**

 Dynamically switch **dimensions** (like Category, Region, Year)

 Dynamically switch **measures** (like Sales, Profit, Quantity)

 Build **custom slicers** to control visuals

Let’s say you have this table:

| **Date** | **Product** | **Region** | **Sales** | **Profit** |
| --- | --- | --- | --- | --- |

You want users to **choose** between viewing:

* Sales by Product
* Profit by Region
* Quantity by Date

With Field Parameters, you don’t need to make separate visuals—you can create **a slicer** to let users switch fields **on the fly**!

**How to Create a Field Parameter (Step-by-Step)**

**✅ Step 1: Go to Modeling > New Parameter > Fields**

1. In Power BI Desktop, go to **Modeling > New parameter > Fields**
2. Select fields/measures to include, e.g.:
   * Category
   * Region
   * Year
3. Click **Create**

Power BI will create a new **calculated table** with your fields and an accompanying slicer.

**Step 2: Use It in a Visual**

1. Drag the new parameter (like Field Parameter) to a visual (e.g., X-axis or Columns)
2. Add a **slicer** for the field parameter
3. Let users choose the field they want to analyze!

💡 If it's a **measure parameter**, you can drag it to **Values** in charts.

7) How to keep default total sales value even if the external user apply filter.

Basically we are to asked to stop working of slicer.

Follow this approach: go to format tab 🡪 Edit Interaction🡪 Block Interaction

8)

9) I have category column a, b, c, d. I want to show visual X axis c,d,a,b . how to handle this situation. (Done)

63) find the total sale done by female belongs to "Maharastra","Karnataka", "Gujarat".

<https://www.youtube.com/watch?v=h_g4mPAU890&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=63>

yhaa done by me..

41) **difference between calculated column and measure**

**Difference Between Calculated Column and Measure in Power BI**

| Feature | Calculated Column 🏛️ | Measure 📊 |
| --- | --- | --- |
| Evaluation Context | Evaluated at row level (per row of the table) | Evaluated at query level (aggregated) |
| Storage | Stored in the dataset, increasing file size | Computed on-the-fly, does not increase file size |
| Recalculation | Computed once when data is refreshed | Computed dynamically based on filter context |
| Usage | Used in row-level calculations (e.g., new columns) | Used for aggregations (SUM, AVERAGE, COUNT, etc.) |
| Performance | Can slow down reports if overused | More efficient as it computes only when needed |
| Example Formula | NewColumn = Sales[Quantity] \* Sales[Price] | Total Sales = SUM(Sales[Quantity] \* Sales[Price]) |
| Best For | Creating new fields in tables | Dynamic calculations in visuals |

When to Use What?

✅ Use a Calculated Column if:

* You need row-by-row calculations (e.g., "Profit = Sales - Cost").
* You need to use it in relationships, slicers, or row-level filters.

✅ Use a Measure if:

* You need aggregated values (SUM, AVERAGE, COUNT, etc.).
* You want the result to change dynamically based on filters.

Example

🏛️ Calculated Column (Stored in Table)

Profit = Sales[Revenue] - Sales[Cost]

✔ Creates a new column at the row level.  
✔ Can be used in slicers and relationships.

📊 Measure (Calculated in Real-Time)

Total Profit = SUM(Sales[Revenue]) - SUM(Sales[Cost])

✔ Aggregates data dynamically based on filters in the report.  
✔ More efficient and does not increase storage size.

🚀 Best Practice: Prefer Measures over Calculated Columns whenever possible to improve performance!

What is Filter Context in Power BI?

What is Filter Context in Power BI?

🔹 Filter Context refers to the set of active filters applied to a calculation in Power BI. These filters can come from:

* Slicers & Filters
* Rows & Columns in Tables or Matrices
* Relationships Between Tables

It determines which subset of data is used in a DAX formula.

Example: Filter Context in Calculated Columns vs. Measures

Consider a Sales Table:

| OrderID | Product | Quantity | Price | Revenue |
| --- | --- | --- | --- | --- |
| 1 | Apple | 10 | 5 | 50 |
| 2 | Apple | 8 | 5 | 40 |
| 3 | Orange | 6 | 7 | 42 |

🏛️ Calculated Column (Row Context, No Filter Context)

Revenue = Sales[Quantity] \* Sales[Price]

✅ This runs per row without any filters from visuals.

📊 Measure (Respects Filter Context)

Total Revenue = SUM(Sales[Revenue])

✅ If a user selects "Apple" in a slicer, the measure recalculates based on that filter context.  
✅ If a report table has Product as a row, each row gets a different total:

| Product | Total Revenue |
| --- | --- |
| Apple | 90 |
| Orange | 42 |

Key Difference

* Calculated Columns 🚫 Do not respect filter context; values are precomputed per row.
* Measures ✅ Respect filter context and change dynamically based on user selections.

🚀 Best Practice: Use measures for aggregations because they dynamically respond to filters!

**42) Filter Vs Slicer |**

Difference Between Filter and Slicer in Power BI

| Feature | Filter 🎛️ | Slicer 🔘 |
| --- | --- | --- |
| Purpose | Filters data at different levels (report, page, visual) | Provides an interactive way for users to filter data |
| Location | Found in the Filters Pane | Placed directly in the report canvas |
| User Interaction | Users cannot interact directly while viewing the report | Users can interact and change selections dynamically |
| Filter Scope | Can apply to entire report, page, or a single visual | Affects only visuals connected to the slicer |
| Types | Basic, Advanced, Relative Date, Top N, etc. | List, Dropdown, Hierarchy, Date, Numeric Ranges |
| Use Case | Apply permanent or hidden filters (e.g., show only last 12 months) | Give users control over filtering (e.g., choose a region) |

Example

🎛️ Filter Example

* A filter on Page Level to show only sales from the year 2023.
* A Visual Filter on a sales table to show only products with revenue > $1000.

🔘 Slicer Example

* A slicer with Product Category to allow users to pick categories dynamically.
* A Date Range Slicer to let users choose a custom date range.

When to Use What?

✅ Use Filters when you want predefined filtering (e.g., restrict data to managers).  
✅ Use Slicers when you want users to interact and filter data dynamically.

🚀 Best Practice: Use Filters for performance and Slicers for user-friendly reports!

**40) Append vs Merge in Power BI**

| Feature | Append Queries 🔄 | Merge Queries 🔗 |
| --- | --- | --- |
| Purpose | Combines data by stacking rows (Union) | Combines data by adding columns from another table (Join) |
| How it Works | Adds rows from one table to another | Matches rows from two tables based on a common column |
| Output | More rows but same number of columns | More columns but same number (or fewer) rows |
| Use Case | Combining similar datasets (e.g., Monthly sales data) | Fetching related data from another table (e.g., Customer info for each order) |
| Relationship Required? | No | Yes (Common key column needed) |
| Example | Append Sales\_Jan and Sales\_Feb to form a full dataset | Merge Orders with Customers to get Customer Names |

Example Use Cases

🔄 Append Example (Adding More Rows)

* You have two tables:  
  Sales\_Jan

| OrderID | Product | Revenue |
| --- | --- | --- |
| 1 | Apple | 100 |
| 2 | Orange | 150 |

* Sales\_Feb

| OrderID | Product | Revenue |
| --- | --- | --- |
| 3 | Banana | 200 |
| 4 | Apple | 120 |

After Append → Combined Table:

| OrderID | Product | Revenue |
| --- | --- | --- |
| 1 | Apple | 100 |
| 2 | Orange | 150 |
| 3 | Banana | 200 |
| 4 | Apple | 120 |

🔗 Merge Example (Adding More Columns)

* You have two tables:  
  Orders Table

| OrderID | CustomerID | Product |
| --- | --- | --- |
| 1 | C001 | Apple |
| 2 | C002 | Orange |

* Customers Table

| CustomerID | Customer Name |
| --- | --- |
| C001 | John |
| C002 | Emma |

After Merge (Joining on CustomerID):

| OrderID | CustomerID | Product | Customer Name |
| --- | --- | --- | --- |
| 1 | C001 | Apple | John |
| 2 | C002 | Orange | Emma |

When to Use What?

✅ Use Append when working with the same structure (e.g., combining multiple months of sales).  
✅ Use Merge when working with related data from different tables (e.g., adding customer details to sales data).

🚀 Best Practice: Prefer Append for historical data and Merge for enriching data with additional details!

**73)**

**- 2 : Explain the concept of bi directional cross filtering in power Bi. When we use it and tell about its potential drawbacks**

[**https://www.youtube.com/watch?v=c2dI5dWBoqc&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=73**](https://www.youtube.com/watch?v=c2dI5dWBoqc&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=73)

Bi-Directional Cross Filtering in Power BI

Bi-directional cross-filtering is a feature in Power BI that allows filters to flow in both directions between related tables. Unlike the default single-direction filtering (where filters propagate from one table to another in one direction), bi-directional filtering enables data selection to influence both tables in a relationship.

When to Use Bi-Directional Cross Filtering

1. Complex Data Models: When there are multiple fact tables and dimension tables, bi-directional filtering helps maintain consistent filtering across tables.
2. Many-to-Many Relationships: Used in scenarios where a dimension table is related to multiple fact tables, ensuring proper aggregation.
3. Multiple Filter Contexts: When reports require interaction across multiple tables, bi-directional filtering ensures that selections impact all relevant data.
4. Aggregations Across Multiple Tables: Useful when aggregating data that spans multiple related tables.

Potential Drawbacks

1. Performance Issues: Bi-directional filtering increases query complexity, leading to slower performance, especially with large datasets.
2. Circular Dependency: It can create circular relationships, causing errors or incorrect calculations.
3. Unintended Filter Propagation: Filters may propagate in unexpected ways, leading to incorrect or misleading results.
4. Harder Debugging: Troubleshooting filter logic can become challenging when multiple tables are affected.

Best Practices

* Use bi-directional filtering only when necessary.
* Avoid applying it on large datasets to prevent performance degradation.
* Prefer single-direction filters unless cross-filtering is explicitly required.
* Test the impact of bi-directional filtering on report calculations before deployment.

What is cross filtering :

**CROSSFILTER DAX Function in Power BI**

The CROSSFILTER function in DAX is used to control the filter direction in a relationship between two tables within a DAX calculation. It temporarily overrides the existing relationship direction for the duration of the formula where it is applied.

Syntax:

CROSSFILTER( columnName1, columnName2, direction )

* columnName1, columnName2: These are the columns that define the relationship.
* direction: This determines how filters should be applied. It accepts three values:
  + None – Disables filtering between tables.
  + OneWay – Enables a single-direction filter.
  + Both – Enables bidirectional filtering.

When to Use CROSSFILTER

1. Changing Filter Direction in Measures:
   * If your data model has a single-direction relationship but you need a bidirectional filter in a specific calculation.
2. Handling Many-to-Many Relationships:
   * When working with many-to-many relationships, using CROSSFILTER inside a measure can help propagate filters correctly.
3. Overriding Default Relationship Behavior:
   * When you need different filter behavior inside a DAX formula without permanently modifying the model relationship.

Example 1: Using CROSSFILTER to Apply Bi-Directional Filtering in a Measure

Scenario

You have the following tables:

* Sales (Fact table)
  + Columns: SaleID, ProductID, SaleAmount
* Products (Dimension table)
  + Columns: ProductID, ProductName

By default, Power BI relationships are single-direction (Products → Sales). However, you want a measure that allows filtering in both directions without modifying the model.

DAX Measure:

Total Sales Bi-Dir =

CALCULATE(

SUM(Sales[SaleAmount]),

CROSSFILTER(Products[ProductID], Sales[ProductID], Both)

)

What happens here?

* CROSSFILTER changes the relationship between Products and Sales to bidirectional inside this measure.
* Even if Sales does not normally filter Products, in this measure, filtering on Sales will also filter Products.

Example 2: Disabling Filtering Between Tables

Scenario

If you want to ignore the relationship between Sales and Products, use CROSSFILTER with None.

DAX Measure:

Total Sales No Filter =

CALCULATE(

SUM(Sales[SaleAmount]),

CROSSFILTER(Products[ProductID], Sales[ProductID], None)

)

Effect:

* This measure calculates total sales while ignoring any filters applied from Products.

Example 3: Forcing One-Way Filtering

Scenario

You want to make sure that filtering only works from Products to Sales and not the other way around.

DAX Measure:

Total Sales One-Way =

CALCULATE(

SUM(Sales[SaleAmount]),

CROSSFILTER(Products[ProductID], Sales[ProductID], OneWay)

)

Effect:

* Even if Sales contains filters, they won’t propagate to Products. Only Products can filter Sales.

Key Takeaways

✅ CROSSFILTER is temporary – It applies only within the DAX formula and does not change the model.  
✅ It helps with dynamic filtering – Useful when you need different filter behavior in different measures.  
✅ Use with caution – Overusing bidirectional filtering can lead to performance issues and unintended results.

**How could you optimize the performance of large power bi report with multiple data sources and complex dax calculations.**

**Optimizing Performance in Power BI with Large Datasets & Complex DAX 🚀**

**When working with large Power BI reports containing multiple data sources and complex DAX calculations, performance optimization is crucial. Below are best practices categorized into Data Modeling, DAX Optimization, Query Performance, and Report Rendering.**

**1️Data Modeling Optimization**

**✅ Use Star Schema Instead of Snowflake Schema**

* **A Star Schema (Fact and Dimension tables) improves performance over a Snowflake Schema.**
* **Flatten hierarchies when possible to reduce the number of table joins.**

**✅ Reduce Data Volume**

* **Remove unnecessary columns: Only import the columns that are required for reporting.**
* **Use aggregations: Instead of loading raw transactional data, pre-aggregate data in SQL or Power Query (e.g., daily sales instead of individual transactions).**
* **Reduce cardinality: Avoid high-cardinality columns like timestamps with milliseconds if they aren’t needed.**

**✅ Use Proper Data Types**

* **Use integer or fixed decimal types instead of floating point numbers.**
* **Convert high-cardinality text columns into numeric keys (e.g., replace "Customer Name" with "Customer ID").**

**✅ Disable Bi-Directional Relationships (Where Possible)**

* **Bi-directional filtering slows down performance in large models.++specific cases.**

**2️⃣ DAX Optimization Techniques**

**✅ Avoid Using Calculated Columns**

* **Prefer Power Query transformations over DAX calculated columns because Power Query applies transformations during data load, improving model efficiency.**

**✅ Optimize Measures Instead of Using Calculated Tables**

* **Calculated tables are processed every time the model refreshes, which can slow performance.**
* **Instead, use DAX measures, which are evaluated dynamically.**

**✅ Avoid Using Iterator Functions (SUMX, FILTER, ALL, RANKX)**

* **Iterators scan row by row, which is slow for large datasets.**
* **Prefer using aggregations like SUM, AVERAGE, COUNT instead of SUMX when possible.**

**Example: BAD vs GOOD DAX**

**❌ Slow (Using Iterators):**

**Total Sales = SUMX(Sales, Sales[Quantity] \* Sales[UnitPrice])**

**✅ Optimized Version:**

**Total Sales = SUM(Sales[Quantity]) \* AVERAGE(Sales[UnitPrice])**

* **The optimized version avoids row-by-row iteration.**

**✅ Reduce Cardinality in DAX Measures**

* **High-cardinality calculations (e.g., distinct counts) are expensive.**
* **Use SUMMARIZE or GROUPBY to pre-aggregate data before applying DISTINCTCOUNT.**

**Example: Reducing Cardinality in DISTINCTCOUNT**

**❌ Slow:**

**Unique Customers = DISTINCTCOUNT(Sales[CustomerID])**

**✅ Faster Version:**

**Unique Customers = COUNTROWS(SUMMARIZE(Sales, Sales[CustomerID]))**

* **This version pre-aggregates unique customer IDs, improving performance.**

**✅ Use Variables in DAX (VAR)**

* **Variables reduce redundant calculations by storing intermediate values.**
* **Helps with performance and readability.**

**Example: Using Variables in DAX**

**❌ Slow:**

**Total Sales = SUMX(Sales, Sales[Quantity] \* Sales[UnitPrice])**

**✅ Optimized Version:**

**Total Sales =**

**VAR Qty = SUM(Sales[Quantity])**

**VAR Price = AVERAGE(Sales[UnitPrice])**

**RETURN Qty \* Price**

* **The optimized version calculates SUM and AVERAGE once instead of per row.**

**3️⃣ Query Performance Optimization**

**✅ Use Power Query Instead of DAX for Data Transformations**

* **Move heavy transformations (joins, filters, calculated columns) to Power Query or the source database.**
* **Merge & Append Queries in Power Query instead of using relationships to combine tables dynamically.**

**✅ Reduce the Number of Queries**

* **Merge similar queries and remove unnecessary transformations.**
* **Use parameterized queries instead of hardcoded filters.**

**✅ Optimize Data Refresh Performance**

* **Enable Incremental Refresh (Only refresh new/changed data, not the entire dataset).**
* **Use DirectQuery for real-time data if performance is acceptable.**

**4️⃣ Report Rendering Optimization**

**✅ Limit Visuals Per Page**

* **Reduce the number of visuals per report page (10-20 max).**
* **Too many visuals lead to slow rendering and refresh times.**

**✅ Optimize Slicers & Filters**

* **Use Hierarchical Slicers instead of multiple slicers for improved filtering.**
* **Avoid too many slicers – Instead, use filter panes or drop-down slicers.**

**✅ Avoid High-Cardinality Fields in Visuals**

* **Large categorical fields (e.g., Customer Names with millions of rows) slow down visuals.**
* **Use aggregated values or group data where possible.**

**✅ Turn Off Auto Date/Time**

* **Go to Power BI Settings → Data Load → Uncheck "Auto Date/Time".**
* **This setting creates hidden date tables for every date field, which increases memory usage.**

**Final Thoughts**

**By optimizing data modeling, DAX calculations, query performance, and report rendering, you can significantly improve the speed and responsiveness of your Power BI reports.**

**72)**

**:** Let suppose we have sales table with 4 column given below:

Date 02-05-2024

Product id 1

Product price 100

Discount(%) 30%

How to calculate net sales with discount <https://www.youtube.com/watch?v=kiABuIawkzw&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=72>

Q2: How to give power Bl report access to user with different domain?

**Giving Power BI Report Access to Users with a Different Domain**

If you need to **share a Power BI report** with users **outside your organization (different domain)**, there are multiple methods depending on whether they are **internal guests**, **external clients**, or **public users**.

**1️⃣ Share Power BI Report via External Guest Access (Recommended for Specific Users)**

If the external user has a **Microsoft account (Azure AD or Office 365)**, you can invite them as a **guest** in your Power BI tenant.

**Steps to Share Power BI Report with External Users:**

1. **Go to Power BI Service** ([https://app.powerbi.com](https://app.powerbi.com/))
2. **Open the Report** you want to share.
3. Click **Share** (Top-right corner).
4. Enter the external user’s **email address (e.g.,** [**user@differentdomain.com**](mailto:user@differentdomain.com)**)**.
5. Power BI will recognize it as an external domain and prompt you to add the user as a **guest**.
6. Click **Send Invitation**.
7. The external user will receive an email to accept the invitation and access the report.

✅ **Requirements & Considerations**:

* External users **must have a Power BI Pro license** (or the report should be in a Power BI Premium workspace).
* They **must sign in with a Microsoft account** (Gmail or other personal emails won’t work unless linked to Microsoft).
* The **admin must enable external sharing** in **Azure AD settings** (Admin > External collaboration settings).

**2️⃣ Publish to Web (Public Access - No Authentication Required)**

If you want to share the report **publicly**, you can use the **Publish to Web** option.

**Steps:**

1. Open **Power BI Service**.
2. Select the **report**.
3. Click **File > Embed Report > Publish to Web (Public)**.
4. Confirm the warning that the report will be **publicly available**.
5. Copy the **embed link** and share it.

❌ **Limitations & Risks**:

* Anyone with the link **can view the report** (no security or access control).
* Data can be indexed by **search engines** (Google, Bing, etc.).
* **Not suitable for sensitive or confidential data**.

✅ **Best for:**

* Public dashboards (e.g., COVID-19 stats, financial reports, etc.).
* Non-sensitive business performance indicators.

**3️⃣ Use Power BI Embedded (For Clients & Web Applications)**

If you want to embed Power BI reports into an **external website, web app, or customer portal**, use **Power BI Embedded**.

**Steps:**

1. **Go to Power BI Service** → Select Report.
2. **File > Embed Report** → Choose **Website or Portal**.
3. Copy the **secure embed code** and paste it into your web application.
4. Users **must log in** to view the report (no anonymous access).

✅ **Best for:**

* Clients accessing reports inside a **secure web portal**.
* Web apps requiring **user authentication** before viewing reports.

📌 **Note:** Requires **Power BI Pro or Premium Capacity (P1 or higher)**.

**4️⃣ Add External Users via Azure B2B (For Enterprise Access)**

If sharing reports frequently with **external partners, vendors, or customers**, consider **Azure B2B (Business-to-Business)** integration.

**Steps:**

1. **Go to Azure AD Admin Center** → **Users**.
2. Click **"New Guest User"**.
3. Enter the external user's **email address**.
4. Assign them to a **Power BI Security Group**.
5. In **Power BI Service**, grant access to the shared workspace for the guest user.

✅ **Best for:**

* Large organizations collaborating with multiple external vendors or partners.
* Securely managing **long-term access to multiple Power BI reports**.

**🚀 Conclusion: Which Method Should You Choose?**

| **Scenario** | **Best Sharing Method** |
| --- | --- |
| Sharing with **one or two external users** | **Guest User Access** (Azure AD) ✅ |
| Sharing publicly **without authentication** | **Publish to Web** (⚠️ No security) |
| Sharing securely with **external clients via a web app** | **Power BI Embedded** ✅ |
| Sharing with **many external users from a different company** | **Azure B2B (Enterprise)** ✅ |

Would you like a **step-by-step guide** on enabling guest access or configuring Azure B2B? 🚀

**71 )Startup**

1. Using the Sample-Superstore data, find what percent of Profit was generated by the top 10 products with the highest Sa les?-was and creating a dax to calculate % of profit

+

Create a bar chart by month and sales Give Alternate Red and Green Colour to the Bars in the view For Eg Jan, Mar, etc will be displayed as green colour and Feb, Apr, etc as red colour Write the logic to impitment this

<https://www.youtube.com/watch?v=406gH8VrslM&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=71>

61) Capgemini <https://www.youtube.com/watch?v=ppI1TMCGwLQ&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=61>

* + 1. can we use Username and userprinciple name in static RLS

Ans : **Can We Use USERNAME() and USERPRINCIPALNAME() in Static RLS?**

No, **USERNAME()** and **USERPRINCIPALNAME()** cannot be used in **static RLS (Row-Level Security)**.

**Why?**

* **Static RLS** is based on **fixed filtering conditions** applied directly to tables (e.g., filtering data for a specific region or department).
* **USERNAME()** and **USERPRINCIPALNAME()** are **dynamic functions**, meaning they return the logged-in user's credentials and are typically used for **dynamic RLS** instead.

**When Can You Use Them?**

You can use **USERNAME()** or **USERPRINCIPALNAME()** in **dynamic RLS**, where a mapping table (User Access Table) is used to filter data based on the logged-in user.

**Example of Dynamic RLS using USERPRINCIPALNAME():**

1. Create a **User Access Table** with:
   * **User Email** (matches USERPRINCIPALNAME())
   * **Department/Region** (filtering criteria)
2. Define an RLS Role:
3. [User Email] = USERPRINCIPALNAME()

This ensures users see only the data related to their email in the access table.

**Conclusion:**

* **Static RLS → No (Fixed filters, not user-based)**
* **Dynamic RLS → Yes (Filters based on logged-in user using USERNAME() or USERPRINCIPALNAME())**

Would you like a step-by-step guide on implementing dynamic RLS? 😊

* + 1. Does Power bi maintain/store historical data and how

**Does Power BI Maintain/Store Historical Data?**

Power BI **does not** store historical data **by default**, but you can configure it to maintain historical records using various methods.

**How to Maintain Historical Data in Power BI?**

1️⃣ **Using a Data Warehouse (Best Practice)**

* Store historical data in a **SQL Server, Azure Synapse, or a Data Lake**.
* Power BI connects to these sources and visualizes the historical data.

2️⃣ **Using Incremental Refresh (For Large Datasets)**

* Power BI **stores and refreshes only new data** instead of reloading the entire dataset.
* Requires **Power BI Premium** or **Power BI Pro with dataflows**.
* Implemented using **RangeStart** and **RangeEnd** parameters in Power Query.

3️⃣ **Appending Data with Power Query (Manual Approach)**

* Use **Power Query (M Language)** to **append new data** to an existing table instead of replacing it.
* Example: Keep daily/weekly snapshots of sales data.

4️⃣ **Using Dataflows for Historical Data**

* Store transformed data in **Power BI Dataflows**.
* Acts as an intermediate storage for maintaining previous records.

5️⃣ **Exporting to External Storage (CSV, SharePoint, or Databases)**

* Use **Power Automate** or **Power BI APIs** to store snapshots in a **database, SharePoint, or Azure Blob Storage**.

**Conclusion:**

Power BI itself **does not maintain historical data automatically**, but you can **store and track changes** using:  
✅ **Data Warehouses** (Best for long-term storage)  
✅ **Incremental Refresh** (Efficient for large datasets)  
✅ **Appending Data in Power Query** (Good for manual tracking)  
✅ **Dataflows** (Keeps intermediate storage for history)

Would you like a step-by-step guide on implementing any of these? 😊

C) is it possible to create drill down and drill up on 3 fact fables

4)if there are constraint applied on table, does Primary key and Foreign key will affect in power bi7 5)if we are taking data from DB, table has 6k rows and table got truncated has only 1 k rows. Does Power bi will have all the 6 K rows from the table?

**Will Power BI Retain All 6K Rows If the Source Table is Truncated?**

It **depends on the data refresh mode** you are using in Power BI:

**1️⃣ Import Mode (Keeps Old Data Until Refresh)**

✅ If the data was imported into Power BI earlier (when the table had **6K rows**), then Power BI **will still show 6K rows** **until** the next refresh.  
✅ When you refresh Power BI after the table is truncated to **1K rows**, it will **update the dataset** and keep only **1K rows**.

📌 **Key Takeaway:**

* Before refresh → **Power BI has 6K rows**.
* After refresh → **Power BI updates to 1K rows (matching the source)**.

**2️⃣ DirectQuery Mode (Always Shows Current Data)**

❌ Power BI does **not store data locally** in **DirectQuery mode**.  
✅ It **fetches data live** from the database each time a query runs.  
✅ If the table is truncated in the database (now has **1K rows**), Power BI will **immediately reflect this change**.

📌 **Key Takeaway:**

* No refresh is needed; the report will **always show the latest data** (1K rows).

**3️⃣ Incremental Refresh (Can Keep Old Data)**

✅ If **Incremental Refresh** is enabled, Power BI can **retain historical data** (e.g., **keep 6K rows** even if the source table is truncated).  
✅ You need to **define a retention policy** (e.g., keep the last 1 year of data).

📌 **Key Takeaway:**

* If properly configured, Power BI **can keep older data even after the source table is truncated**.

**Final Answer: Will Power BI Still Have 6K Rows?**

🔹 **Import Mode:** Yes, **until you refresh** (then it updates to 1K).  
🔹 **DirectQuery Mode:** No, it will always show the current data (1K rows).  
🔹 **Incremental Refresh:** **Possibly**, if set up to retain historical data.

Would you like a step-by-step guide on setting up **Incremental Refresh**? 😊

6)how will u identify type of license locking at workspace? what is the size of workspace? What is the unit testing in power b? do u do the report validation, how dou do that? how will you cross verify the data which is coming from data source to report? did u done any automation? what is dual mode ?

**62) TCS (<https://www.youtube.com/watch?v=EjnraYcjumg&list=PLk-s1KYuT-PmgxTJKTcXwvxBgMb7rUnza&index=62>**

**)**

* **A) Difference between calculated column and measure and how do they are helpful while creating a report ?**
* **B) how do you improve your power bi report**
* **C) how many types of relation are there in we can perform while creating data model in power Bi ?**
* **D) what are the types of filter you are using in your project**
* **E) if you are calculating total sales and you want to apply filter function inside the calculate column how it will wok ?**

**Ans🡪 filter function can not apply directly in dax code. It is used inside the calculate function. Filter function generate the subset of the table. So filter function can used inside the calculate function and optimized the dax code.**

**Total Sales (Last 30 Days) =**

**CALCULATE(**

**SUM(Sales[SalesAmount]),**

**FILTER(**

**Sales,**

**Sales[OrderDate] >= TODAY() - 30**

**)**

**)**

**F) how many dashboard you have created till date and object of that dashboard. ?**

**Customized answer**

**g) how will you decide that which are matrics will be in the dashboard to show these things to the stake holder without getting any requirement from stakeholder ?**

**I) what is bidirection relationship , do you use it ?**

**J) diff between star and snowflake**

**K) how many types of security in power bi (RLS)**

**l) how many types of role are available in power bi services and what is your role (member role)**

**Types of Roles in Power BI Service**

**Power BI Service has different roles based on workspace permissions, admin roles, and row-level security (RLS) roles.**

**1️⃣ Workspace Roles (Collaboration & Access Control)**

**These roles control access and permissions in Power BI Workspaces:**

| **Role** | **Permissions** |
| --- | --- |
| **Admin** | **Full control (manage members, edit, delete content, publish reports, etc.)** |
| **Member** | **Can edit, share, and publish reports but cannot manage workspace settings.** |
| **Contributor** | **Can create, edit, and publish content but cannot share or manage permissions.** |
| **Viewer** | **Read-only access; can view reports/dashboards but cannot edit or publish.** |

**📌 Where to Assign? → In Power BI Workspaces under "Manage Permissions".**

**2️⃣ Admin Roles (Power BI Tenant Level)**

**These roles manage Power BI settings at the organization level (via Microsoft 365 Admin Center).**

| **Role** | **Permissions** |
| --- | --- |
| **Power BI Admin** | **Full control over Power BI settings, security, capacity, and usage.** |
| **Global Admin** | **Manages all Microsoft 365 services, including Power BI.** |
| **Fabric Admin** | **Manages settings related to Fabric (formerly Azure Synapse in Power BI).** |
| **Capacity Admin** | **Manages Power BI Premium/Embedded capacities.** |

**📌 Where to Assign? → In Microsoft 365 Admin Center under Roles & Permissions.**

**3️⃣ Row-Level Security (RLS) Roles (Data-Level Access)**

**These roles control what data a user can see within a report.**

| **Type** | **Description** |
| --- | --- |
| **Static RLS** | **Predefined filters (e.g., "Managers see all data, Employees see only their department").** |
| **Dynamic RLS** | **Uses USERPRINCIPALNAME() or USERNAME() to filter data dynamically.** |

**📌 Where to Assign? → In Power BI Desktop under Modeling → Manage Roles.**

**🔹 Summary: Types of Roles in Power BI**

| **Role Type** | **Purpose** | **Example** |
| --- | --- | --- |
| **Workspace Roles** | **Manage report access & editing** | **Admin, Member, Contributor, Viewer** |
| **Admin Roles** | **Control Power BI settings at the org level** | **Power BI Admin, Global Admin, Fabric Admin** |
| **RLS Roles** | **Restrict access to data within reports** | **Region-based access for sales teams** |

**Would you like a step-by-step guide on assigning any of these roles? 😊**

**m) connecting different data sources with power bi especially MSSQL, Share point**

**n) if excel file contain multiple sheet how will you import all of them in power bi ?**

**70)**

* 1. **Calculate sales by inactive relationship?**

**In Power BI, relationships between tables can be Active or Inactive:**

**1. Active Relationship**

* **This is the default relationship used by Power BI in calculations and visualizations.**
* **Each table can have only one active relationship between two tables at a time.**
* **It is represented by a solid line in the Power BI model view.**
* **Power BI automatically uses the active relationship in measures and calculated columns.**

**2. Inactive Relationship**

* **An inactive relationship is not used by default in calculations.**
* **It is represented by a dotted line in the Power BI model view.**
* **You can have multiple inactive relationships between two tables, but only one can be active.**
* **To use an inactive relationship in calculations, you must use the USERELATIONSHIP() function in DAX.**

**Example Scenario**

**You have a Sales table and a Date table.**

* **An active relationship is created between Sales[Order Date] and Date[Date] (used in default calculations).**
* **An inactive relationship exists between Sales[Ship Date] and Date[Date] (not used unless explicitly activated).**

**Using Inactive Relationship in DAX**

**DAX**

**Total Sales by Ship Date =**

**CALCULATE(**

**SUM(Sales[Sales Amount]),**

**USERELATIONSHIP(Sales[Ship Date], Date[Date])**

**)**

**This formula forces Power BI to use the inactive relationship.**

**26) you have sales table contains multiple column, now you have to create a table that has only 3 column and following condition should be followed**

**Product[id]>=100, and product[price[<=500 and product name start with “s”**

**Summerizecolumn() :**

**The SUMMARIZECOLUMNS() function is a powerful DAX function used to create grouped summaries of data with aggregations. It is commonly used in table visualizations and DAX queries to return a table with selected columns and aggregated values.**

**Syntax**

**DAX**

**CopyEdit**

**SUMMARIZECOLUMNS(**

**<GroupBy\_Column1>, <GroupBy\_Column2>, ...,**

**<Filter1>, <Filter2>, ...,**

**<Name1>, <Expression1>,**

**<Name2>, <Expression2>, ...**

**)**

**Parameters**

* **GroupBy\_Column – Columns to group the data by (similar to GROUP BY in SQL).**
* **Filters (Optional) – Conditions to filter the data.**
* **Name & Expression – A column name and a corresponding aggregation expression (like SUM, AVERAGE, COUNT, etc.).**

**Example 1: Simple Aggregation**

**Get Total Sales by Product Category**

**DAX**

**CopyEdit**

**SalesByCategory =**

**SUMMARIZECOLUMNS(**

**Products[Category],**

**"Total Sales", SUM(Sales[Amount])**

**)**

**✅ Returns a table like this:**

| **Category** | **Total Sales** |
| --- | --- |
| **Electronics** | **500,000** |
| **Clothing** | **300,000** |
| **Furniture** | **200,000** |

**Example 2: Using Filters**

**Get Total Sales for 2024 by Region**

**DAX**

**CopyEdit**

**Sales2024ByRegion =**

**SUMMARIZECOLUMNS(**

**Sales[Region],**

**Sales[Year] = 2024, -- Filter for the year 2024**

**"Total Sales", SUM(Sales[Amount])**

**)**

**Example 3: Using Multiple Measures**

**Get Total and Average Sales by Customer**

**DAX**

**CopyEdit**

**SalesByCustomer =**

**SUMMARIZECOLUMNS(**

**Customers[CustomerName],**

**"Total Sales", SUM(Sales[Amount]),**

**"Average Sales", AVERAGE(Sales[Amount])**

**)**

**✅ Returns:**

| **Customer** | **Total Sales** | **Average Sales** |
| --- | --- | --- |
| **Alice** | **50,000** | **5,000** |
| **Bob** | **30,000** | **3,000** |

**SUMMARIZE() vs SUMMARIZECOLUMNS()**

| **Feature** | **SUMMARIZE()** | **SUMMARIZECOLUMNS()** |
| --- | --- | --- |
| **Filters** | **Uses CALCULATE()** | **Supports direct filters** |
| **Performance** | **Slower with large data** | **Faster** |
| **Context** | **Works inside measures** | **Used for tables & queries** |
| **Recommended** | **In calculated tables** | **In table visuals & reports** |

**🔹 Use SUMMARIZECOLUMNS() for better performance in reports.  
🔹 Use SUMMARIZE() if you need CALCULATE() for complex filters.**

**LEFT() DAX Function in Power BI**

**The LEFT() function in DAX is used to extract a specified number of characters from the beginning (left side) of a text string.**

**Syntax**

**DAX**

**CopyEdit**

**LEFT(<Text>, <NumberOfCharacters>)**

**Parameters**

* **Text – The text string or column from which you want to extract characters.**
* **NumberOfCharacters – The number of characters to extract from the left side.**

**Example 1: Extract First 3 Characters from a Column**

**If you have a Customers table with a CustomerName column:**

**DAX**

**CopyEdit**

**First3Letters = LEFT(Customers[CustomerName], 3)**

**Result:**

| **CustomerName** | **First3Letters** |
| --- | --- |
| **John Doe** | **Joh** |
| **Alice Smith** | **Ali** |

**Example 2: Extract First 2 Digits from an Order Number**

**DAX**

**CopyEdit**

**OrderPrefix = LEFT(Orders[OrderID], 2)**

**🔹 If OrderID = "A12345", the result is "A1".**

**Example 3: Extract First Word from a Sentence**

**To extract the first word from a Description column:**

**DAX**

**CopyEdit**

**FirstWord = LEFT(Description, SEARCH(" ", Description) - 1)**

**🔹 If Description = "Power BI is great", the result is "Power".**

**Example 4: Using LEFT() in a New Column**

**Create a "Country Code" column from a PhoneNumber column:**

**DAX**

**CopyEdit**

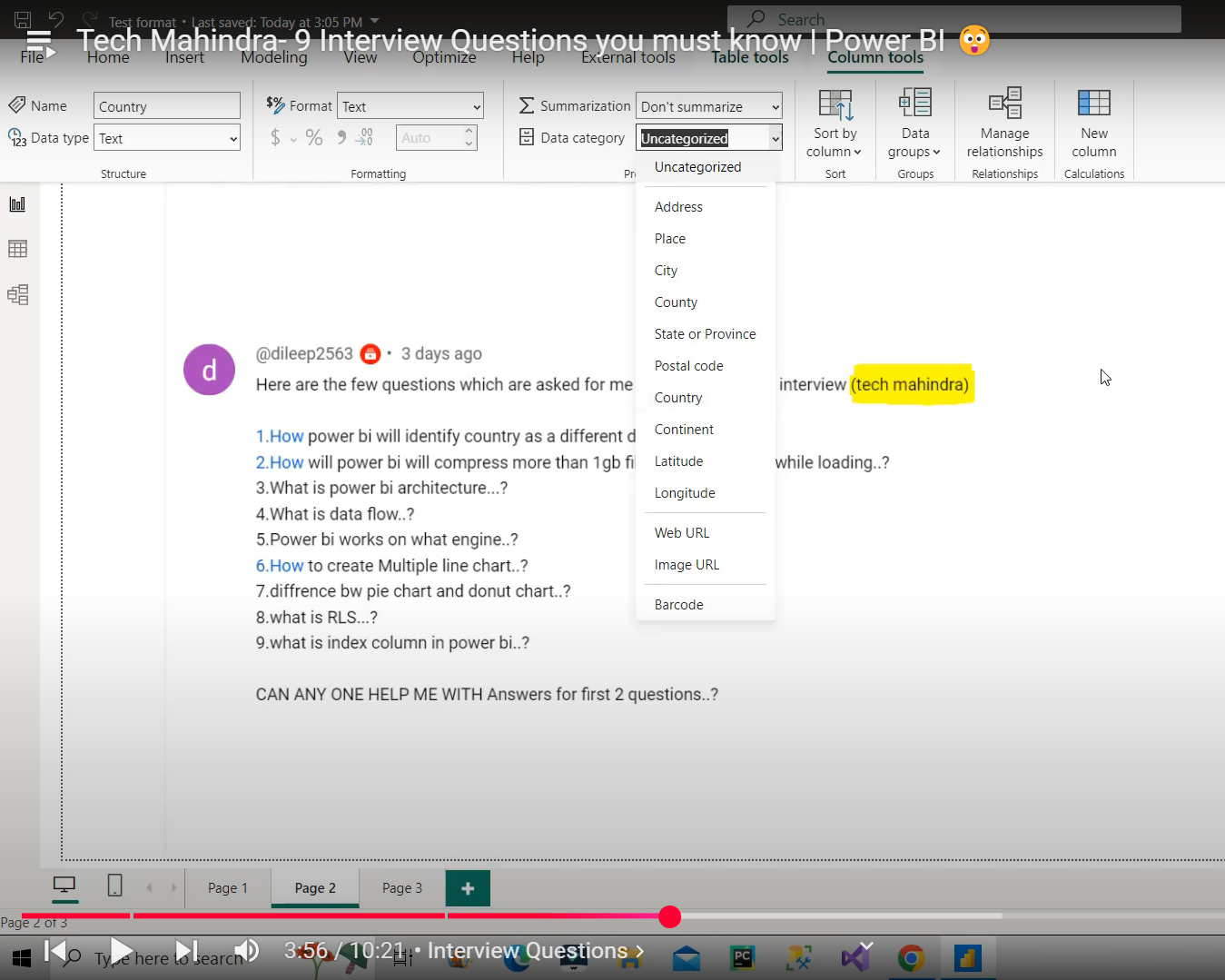
**CountryCode = LEFT(Contacts[PhoneNumber], 3)**

**🔹 If PhoneNumber = "+91-9876543210", the result is "+91".**

**Key Points**

**✔ LEFT() works on text data.  
✔ Use SEARCH() with LEFT() to extract words dynamically.  
✔ Use MID() if you need characters from the middle of a string.  
✔ Use RIGHT() if you need characters from the end of a string.**

**55) How power BI will identify country as different datatype ?**

****

**Ans🡪 Go to data category set country**

**How power bi will compress more than 1 gb file to less than 1 gb file while loading?**

**Ans🡪**

Power BI utilizes the VertiPaq storage engine to compress and optimize data models, allowing for significant file size reduction, potentially compressing a file larger than 1 GB to under 1 GB during loading.

Here's a breakdown of how this compression works:

* Data Compression and Optimization:

When data is imported into Power BI Desktop, the VertiPaq engine compresses and optimizes the data model.

* VertiPaq Storage Engine:

This engine is designed for efficient storage and retrieval of data, allowing for significant compression ratios.

* 10x Compression:

It's possible to achieve 10x compression, meaning that 10 GB of source data could potentially compress to around 1 GB.

* Further Reduction:

When the model is persisted to disk, an additional 20% reduction can be achieved.

* Data Reduction Techniques:
  + Remove Unnecessary Columns: Eliminate columns that aren't essential for your analysis.
  + Aggregate Data: Aggregate data to a higher level of granularity if possible.
  + Use the Right Data Types: Ensure that columns have the appropriate data types to minimize storage space.
  + Disable Auto Date/Time Tables: If you don't need the auto-generated date/time tables, disable them.
  + Reduce Cardinality: Minimize the number of unique values in high-cardinality columns.
  + Optimize DAX Measures: Use DAX measures instead of calculated columns where appropriate.
  + Use Star Schema: Design your data model using a star schema for better performance and storage efficiency.
  + Use Composite Model: Use composite models to combine data from multiple sources and store data in different modes (Import and DirectQuery).

**What is power bi architecture**

**Date extraction 🡪 data cleaning 🡪 visualization 🡪 publishing**

**What is Dataflow?**

Skip this question

Difference between pie chart and donut chart !

**Pie Chart**

A pie chart is a **circular chart** divided into slices to show **proportions of a whole**.

**📌 Characteristics:**

* Shows **100% total** split among categories.
* Each slice represents a **category’s share**.
* **No hole** in the middle.
* Useful for 3–5 categories max — gets messy beyond that.

**✅ Best for:**

* Simple data comparisons.
* When you want to highlight which category is the biggest or smallest.

**🍩 Donut Chart**

A donut chart is basically a **pie chart with a hole** in the middle.

**📌 Characteristics:**

* Also shows proportions of a whole.
* Has a **center hole** that gives it a "donut" appearance.
* You can **display the total value in the center** (a nice trick in Power BI).
* Slightly better for adding **labels or KPIs in the center**.

**✅ Best for:**

* Same use cases as pie chart, but more modern look.
* When you want to display a number (like total sales) in the center.

What is index column !

Nice one, Anjan! The **Index Column** is a simple but super useful feature in Power BI's Power Query. Let’s break it down:

**🔢 What is an Index Column in Power BI?**

An **Index Column** is a column you add in **Power Query Editor** that assigns a unique **numeric value** (like a row number) to each row in your data table.

Think of it like this:

| **Name** | **Sales** | **Index** |
| --- | --- | --- |
| Rahul | 1000 | 0 |
| Anjali | 1200 | 1 |
| Aman | 900 | 2 |

It's just a running number starting from 0 (by default), 1, or even custom steps.

**🛠️ How to add an Index Column:**

1. Go to **Power Query Editor**.
2. On the **"Add Column"** tab → choose **Index Column**.
3. You’ll see options like:
   * **From 0**
   * **From 1**
   * **Custom** (start and step values)

**🎯 Why use an Index Column?**

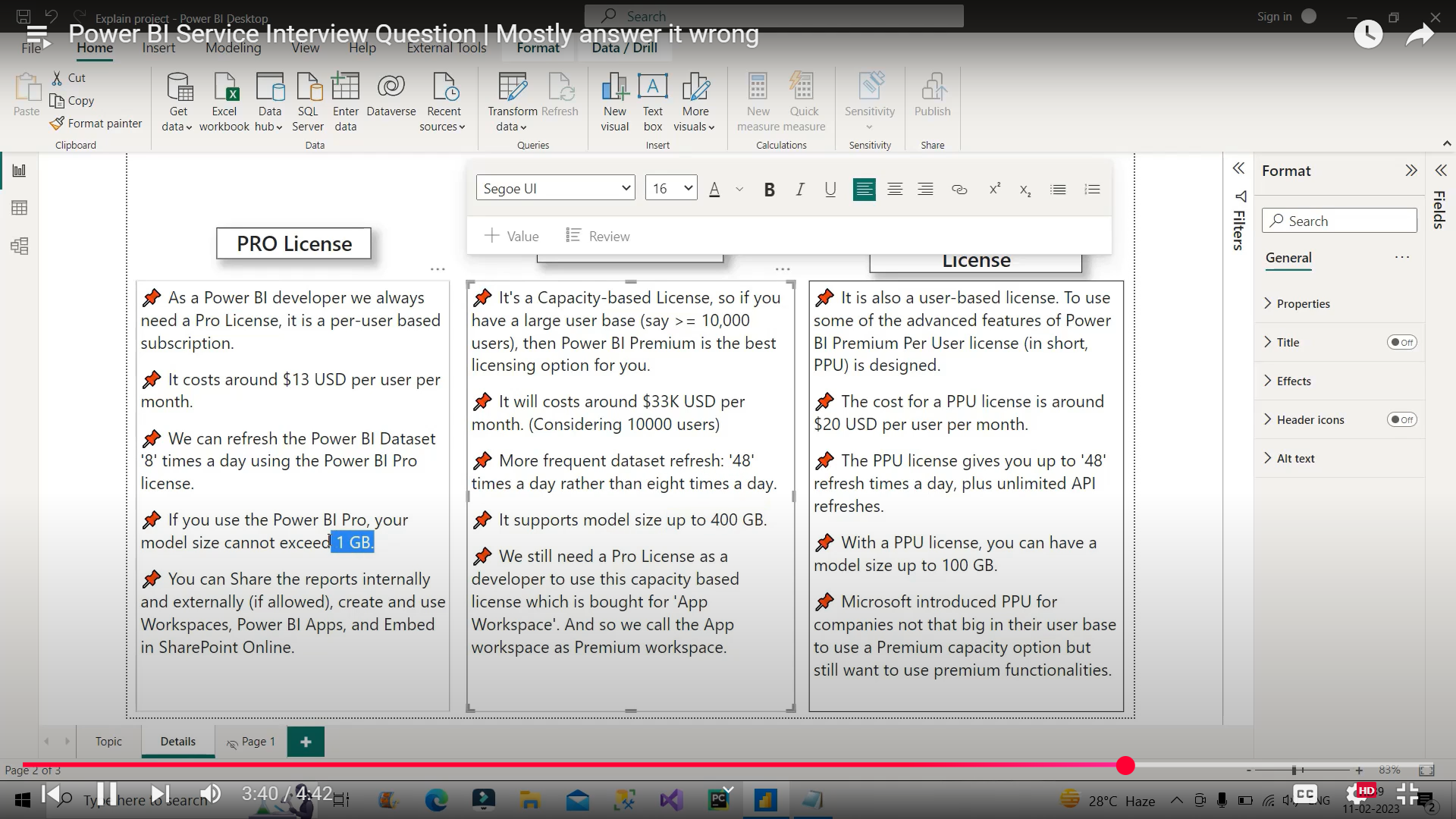
| **Use Case** | **Benefit** |
| --- | --- |
| Sort operations | Helps in **custom sorting** logic. |
| Row reference | Acts like a **unique row ID**. |
| Ranking | Can simulate **rank** if data is sorted first. |
| De-duplication | Helps identify **duplicates** or track **first/last rows** in a group. |
| Merge logic | Used when joining datasets where position matters. |

**🧠 Example Scenario:**

You have a table of sales transactions sorted by date, and you want to find the **first sale** made by each salesperson. You can:

1. Group by salesperson.
2. Sort by date inside each group.
3. Add an **index inside the group**.
4. Filter where index = 0 (first transaction).

Boom — first sale per person 🎯



**------------------------------------------------------------------**

**Another Playlist**

**Question 9)  Calculate Last N Days Sales |**

[**https://www.youtube.com/watch?v=RoClVB8-ql4&list=PLk-s1KYuT-Pk2a1mbFe-nH7BNVdQjsWVU&index=9**](https://www.youtube.com/watch?v=RoClVB8-ql4&list=PLk-s1KYuT-Pk2a1mbFe-nH7BNVdQjsWVU&index=9)

**The DATESINPERIOD function in DAX is u sed to return a table of dates within a specified period, based on a given date column. It is particularly useful when creating time-based calculations such as moving averages, running totals, or comparisons over time.**

**Syntax**

**DATESINPERIOD(<dates>, <start\_date>, <number\_of\_intervals>, <interval>)**

**Parameters**

* **<dates> – A column containing date values.**
* **<start\_date> – A single date that serves as the reference point.**
* **<number\_of\_intervals> – The number of intervals to count forwards (positive) or backwards (negative).**
* **<interval> – The unit of time for the period. It can be one of the following:** 
  + **DAY**
  + **MONTH**
  + **QUARTER**
  + **YEAR**

**How It Works**

**The function returns a table of dates that fall within the specified time period, starting from the given start\_date and extending forward or backward based on number\_of\_intervals.**

**Examples**

**Example 1: Last 3 Months of Sales**

**Last3MonthsSales =**

**CALCULATE(**

**SUM(Sales[TotalSales]),**

**DATESINPERIOD(Sales[OrderDate], MAX(Sales[OrderDate]), -3, MONTH)**

**)**

**🔹 This calculates total sales for the last 3 months from the most recent order date.**

**Example 2: Year-to-Date Sales Using DATESINPERIOD**

**YTD\_Sales =**

**CALCULATE(**

**SUM(Sales[TotalSales]),**

**DATESINPERIOD(Sales[OrderDate], MAX(Sales[OrderDate]), -1, YEAR)**

**)**

**🔹 This retrieves sales data for the last 1 year, starting from the latest available date.**

**Example 3: Last 7 Days Sales**

**Last7DaysSales =**

**CALCULATE(**

**SUM(Sales[TotalSales]),**

**DATESINPERIOD(Sales[OrderDate], MAX(Sales[OrderDate]), -7, DAY)**

**)**

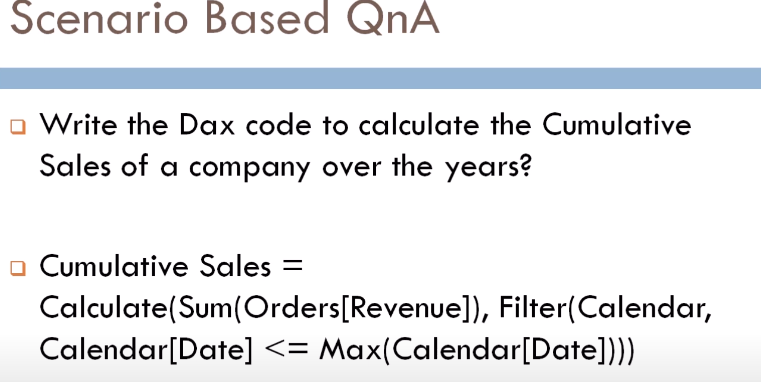
**🔹 This computes sales for the last 7 days.**

**Key Differences from Similar Functions**

| **Function** | **Returns** | **Best Used For** |
| --- | --- | --- |
| **DATESINPERIOD** | **A continuous range of dates based on a start date** | **Moving averages, running totals** |
| **DATESBETWEEN** | **A date range between specific start and end dates** | **Custom date range filtering** |
| **SAMEPERIODLASTYEAR** | **A matching period from the previous year** | **Year-over-Year comparisons** |
| **PREVIOUSMONTH** | **The entire previous month** | **Previous period analysis** |

**Final Notes**

* **DATESINPERIOD always returns a contiguous range of dates.**
* **It works well with CALCULATE to apply time-based filters.**
* **Use negative intervals to go backward in time and positive intervals to go forward.**

****

**Question 8)**

**How will you create a dynamic date table yourself if not present in the model.**

DateTable =

ADDCOLUMNS(

CALENDAR(DATE(2020,1,1), DATE(2030,12,31)), -- Set Start and End Date

"Year", YEAR([Date]),

"Month", FORMAT([Date], "MMMM"),

"Month Number", MONTH([Date]),

"Quarter", "Q" & FORMAT([Date], "Q"),

"Weekday", FORMAT([Date], "dddd"),

"Weekday Number", WEEKDAY([Date], 2) -- Monday = 1

)

**TOTALYTD() DAX Function in Power BI**

The TOTALYTD function in DAX is a **time intelligence** function used to calculate the **Year-To-Date (YTD) total** for a given measure. It accumulates values from the **beginning of the year** up to the selected date.

**Question 1)**

**What is M Query in Power BI?**

**M Query is the formula language used in Power Query Editor in Power BI, Excel, and other Microsoft tools. It is used to clean, transform, and shape data before loading it into the Power BI model.**

* **M stands for "Mashup" because it is designed to combine data from multiple sources.**
* **It is a functional, case-sensitive language similar to F# and is optimized for data transformation tasks.**
* **Every step in Power Query generates an M Query in the background.**

**Where to Write M Query in Power BI?**

**You can write and modify M Queries in Power Query Editor in Power BI.**

**Steps to Write M Query in Power BI**

1. **Open Power Query Editor:**
   * **In Power BI Desktop, go to Home → Click Transform Data.**
2. **View Existing M Query:**
   * **In Power Query Editor, click Advanced Editor from the View tab.**
   * **Here, you will see the auto-generated M Query for the selected table.**
3. **Manually Write or Edit M Query:**
   * **In Advanced Editor, you can modify or write new M queries.**
   * **Example of a simple M Query:**
   * **let**
   * **Source = Excel.Workbook(File.Contents("C:\Data\sales.xlsx"), null, true),**
   * **Sheet1 = Source{[Name="Sheet1"]}[Content],**
   * **ChangedType = Table.TransformColumnTypes(Sheet1, {{"Date", type date}, {"Sales", type number}})**
   * **in**
   * **ChangedType**
   * **Click Done to apply changes.**

**When to Use M Query?**

* **Custom Data Transformations: If the standard Power Query UI does not provide the required transformation.**
* **Performance Optimization: You can optimize queries for faster data processing.**
* **Reusability: You can create custom functions using M Query.**

**Would you like help writing a specific M Query for your use case? 😊**

**How will you change the datatype of date column(20210503) having whole number as its datatype.**

**Answer - > select the column , change data type to text then date.**

What is **Query Folding** in Power BI?

**Query Folding** is the process by which **Power Query** translates your data transformation steps into a **single query** that is **executed by the data source** (like SQL Server, Oracle, etc.), instead of processing the data locally inside Power BI.

**Why is Query Folding Important?**

* **Performance**: Large data transformations are pushed to the source (like SQL), which is optimized for such tasks.
* **Efficiency**: Less data is pulled into Power BI.
* **Incremental Refresh**: Requires query folding to filter rows at the source level.

**Example**

Suppose you import a SQL table and apply these steps in Power Query:

1. Filter rows where Country = 'India'
2. Remove columns Email, Phone
3. Sort by Sales

If Query Folding is active, Power BI will **send a single SQL query** like:

SELECT Name, Country, Sales

FROM Customers

WHERE Country = 'India'

ORDER BY Sales

All transformations happen **at the SQL Server**, not in Power BI.

**What Breaks Query Folding?**

Some operations stop query folding:

* Adding custom columns with complex M logic
* Merging with non-foldable sources (like Excel or Web)
* Using Table.Buffer()
* Reordering columns (in some cases)

| **Feature** | **Description** |
| --- | --- |
| What it does | Pushes transformations to data source |
| When it happens | During Power Query steps (Extract/Transform) |
| Benefits | Improves performance, supports incremental load |
| View Native Query | Check for folding on right-click step |
| Stops Folding | Complex M, buffers, merges with local sources |

**Does Query Folding change the structure of raw data at the source (e.g., SQL Server)?**

**❌ No — Query Folding does not change the raw data at the source.**

**✅ What Actually Happens:**

* Power BI **generates a SQL query** (or equivalent command) based on your transformations.
* That query is **sent to the data source** (like SQL Server) **at runtime** to get only the necessary data.
* The **raw table in SQL remains untouched** — only a filtered/transformed result is retrieved temporarily.

SELECT \* FROM Sales;

But in Power BI, you applied:

* Filter: Country = 'India'
* Remove columns: Email, Phone
* Rename: CustomerName → Name

Power BI will generate a SQL query like:

SELECT CustomerName AS Name, Country, Sales

FROM Sales

WHERE Country = 'India';

This query:

* Runs **only when Power BI needs the data** (during refresh).
* Returns a **custom result** to Power BI.
* Does **not alter** or rewrite the Sales table in SQL Server

Power Query, Merge, refresh, parameters

**Power BI Refresh**:

Three types

* Manual Refresh (Import Mode)
* Schedule Refresh
* Incremental Refresh**(Premium Per Capacity**)

**Incremental Refresh**

**Definition:**

**Incremental refresh** means **only refreshing the new or changed data** instead of reloading the entire dataset every time. This makes it **faster and more efficient** for large datasets.

**Where It's Used:**

* When you have **millions of rows**
* Mainly used with **dates (like transaction date, order date, etc.)**

**Requirements:**

* Use **Power BI Pro/Premium**
* Must have a **date column** in your table
* Setup is done in **Power BI Desktop**, but runs in **Power BI Servic**

**How It Works:**

You define:

* **Range of data to load initially** (e.g., last 5 years)
* **How much data to refresh regularly** (e.g., last 1 month)

Power BI uses **partitions** to load only that month during each refresh, saving time.

**How to Set It:**

1. Add a **RangeStart** and **RangeEnd** parameter (Date/Time type)
2. Apply a filter to your date column using those parameters
3. Go to **Modeling > Incremental Refresh**
4. Define:
   * How much historical data to keep (e.g., 5 years)
   * How much to refresh (e.g., 1 month)
5. Publish to Power BI Service – It will apply automatically.

**🔄 Example:**

You have a transaction table with 10 million rows. You set Incremental Refresh to keep 5 years of data, but only refresh the last 1 month daily. So, instead of refreshing all 10 million rows every time, only the new or changed rows from last month are refreshed.

| **Scenario** | **Use This** |
| --- | --- |
| Small data | Manual or Scheduled Refresh |
| Medium data | Scheduled Refresh |
| Large data (millions of rows) | Incremental Refresh |
| Real-time requirement | Live Connection / DirectQuery |

let consider I have million of transaction record from 2016 to 2019. I want to do incremental refresh month wise. So in parameter start date is 1st december 2019 and end date 30th December.. then I refresh the data....but my doubt is power will take responsibility for sepetember month as well or I have to change start date and end date in power bi

Excellent and very practical question, Anjan! Let's clarify your doubt step by step with **your scenario**:

**🧩 Your Scenario Recap:**

* You have **millions of records from 2016 to 2019**
* You want to use **Incremental Refresh month-wise**
* You set:
  + RangeStart = **1st December 2019**
  + RangeEnd = **31st December 2019**
* Now you're asking:

“Will Power BI handle past months (like September), or do I need to keep changing the parameters?”

**🧠 Key Concept: The Parameters Are Just for Development Preview**

✅ You **do not need to keep changing** the parameters manually.

**👉 RangeStart and RangeEnd are used only in Power BI Desktop to filter data preview and define the logic.**

Once you **publish the report to Power BI Service**, Power BI will **automatically take care of managing date ranges**, partitions, and refreshing logic **based on what you define in the Incremental Refresh policy.**

**🔍 Example Clarified**

Let’s say:

* In **Incremental Refresh settings**, you configure:
  + **Store data for: 4 years**
  + **Refresh last: 1 month**
* Even if RangeStart = 1-Dec-2019 and RangeEnd = 31-Dec-2019, after publishing:
  + Power BI will load **all data from 2016 to 2019**
  + It will **create partitions per month** or as needed
  + It will **only refresh December 2019** every time (since it’s the latest month, per policy)

You **do not need to change parameters manually** to refresh for September, October, etc. Power BI handles it internally.

**✅ Summary Answer for Interview:**

"In Power BI Desktop, I set RangeStart and RangeEnd just to define and test the filter logic. But once the report is published, Power BI ignores those hardcoded values. Instead, it uses the incremental refresh settings (like 'last 4 years' and 'refresh last 1 month') to manage partitions and refresh logic. So I don't have to change dates manually—Power BI handles it intelligently."

**🔁 1. Scheduled Refresh**

**Scheduled Refresh allows automatic refreshing of your data at defined intervals (like daily or hourly) from your data source in the Power BI Service (cloud).**

**Where It's Used:**

* When using Import Mode (data is loaded into Power BI)
* Mostly in Power BI Service (not Power BI Desktop)

**Requirements:**

* Your data source must be reachable from the cloud, or
* Use a Data Gateway if the data source is on-premises (e.g., Excel/SQL Server on your computer**)**

**How to Set It:**

1. Go to your dataset in Power BI Service
2. Click on Settings > Scheduled Refresh
3. Set refresh frequency (daily, hourly, etc.)
4. Enter credentials (to access the data source)
5. Save

**Example:**

**You upload a report that connects to an Excel file on a local server. You set a gateway and schedule a refresh at 7 AM daily to ensure the latest sales data is updated.**

**Power BI Mode**

**Import Mode:**

**DirectQuery Mode**

**Live Connection**

**Composite Models (Hybrid Mode)**

**Import Mode:** Loads data into Power BI memory (VertiPaq) for fastest performance.  This mode delivers fast performance thanks to in-memory querying. Data Stored in PBIX. It's important to understand that imported data is always stored to disk. When queried or refreshed, the data must be fully loaded into memory of the Power BI capacity. Once in memory, Import models can then achieve very fast query results. However, while there are compelling advantages associated with Import models, there are disadvantages, too:

* The entire model must be loaded to memory before Power BI can query the model, which can place pressure on available capacity resources, especially as the number and size of Import models grow.
* Model data is only as current as the latest refresh, and so Import models need to be refreshed, usually on a scheduled basis.
* A full refresh removes all data from all tables and reloads it from the data source. This operation can be expensive in terms of time and resources for the Power BI service, and the data sources.

Power BI can achieve incremental refresh to avoid truncating and reloading entire tables.

From a Power BI service resource perspective, Import models require:

* Sufficient memory to load the model when it's queried or refreshed.
* Processing resources and extra memory resources to refresh data.

**DirectQuery Mode:**

*DirectQuery* mode is an alternative to Import mode. Models developed in DirectQuery mode don't import data. Instead, they consist only of metadata defining the model structure. When the model is queried, native queries are used to retrieve data from the underlying data source.

There are two main reasons to consider developing a DirectQuery model:

* When data volumes are too large, even when [data reduction methods](https://learn.microsoft.com/en-us/power-bi/guidance/import-modeling-data-reduction) are applied, to load into a model, or practically refresh.
* When reports and dashboards need to deliver *near real-time* data, beyond what can be achieved within scheduled refresh limits. Scheduled refresh limits are eight times a day for shared capacity, and 48 times a day for a Premium capacity.

There are several advantages associated with DirectQuery models:

* Import model size limits don't apply.
* Models don't require scheduled data refresh.
* Report users see the latest data when interacting with report filters and slicers. Also, report users can refresh the entire report to retrieve current data.
* Real-time reports can be developed by using the [Automatic page refresh](https://learn.microsoft.com/en-us/power-bi/create-reports/desktop-automatic-page-refresh) feature.
* Dashboard tiles, when based on DirectQuery models, can update automatically as frequently as every 15 minutes.

Disadvantages

* DAX formulas are limited to use only functions that can be transposed to native queries understood by the data source. Calculated tables aren't supported.

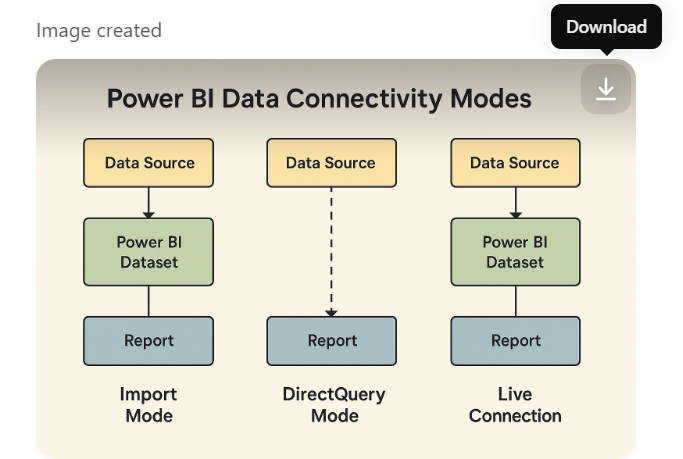
From a Power BI service resource perspective, DirectQuery models require:

* Minimal memory to load the model (metadata only) when it's queried.
* Sometimes the Power BI service must use significant processor resources to generate and process queries sent to the data source. When this situation arises, it can affect throughput, especially when concurrent users are querying the model.
* **Composite mode**

Mix of **Import + DirectQuery** in one model

✅ You can:

* Import static tables (e.g., Product List)
* Use DirectQuery for real-time tables (e.g., Sales, Orders)
* Combine them into **one report** — best of both worlds!



**Why Do Data Modelers Use Import Mode for Dimension Tables in Composite Models?**

**🔍 Composite Model Recap:**

In Power BI, **Composite Models** allow combining:

* **Import Mode** tables (cached in memory)
* **DirectQuery Mode** tables (queried live from source)

**✅ Dimension Tables → Usually in Import Mode**

**🔑 Reasons:**

**1. Dimensions Are Typically Small and Stable**

* Tables like Date, Product, Region, Customer don’t change frequently
* So, importing them once is efficient
* No need to query them repeatedly from the source

✅ Result: **Faster performance**

**2. Speeds Up Query Performance**

* Imported dimension tables live in **VertiPaq memory** (Power BI’s in-memory engine)
* They allow **fast joins** with large fact tables (even if fact is DirectQuery)

🚀 Example:

You slice a huge sales table (DirectQuery) by Product (Import)  
→ The filter on Product can be evaluated locally = **much faster**

**3. Avoids Chained Queries**

* If both fact and dimension tables are in DirectQuery from **different sources**, Power BI can't natively join them
* That leads to **chained or inefficient queries**

✅ Keeping dimensions in Import breaks this chain and **simplifies relationships**

**4. Supports Relationships + Filtering Logic**

* Relationships work **best and most flexibly** when the lookup (dimension) table is in Import mode
* Directional filtering (like many-to-one) works faster and more reliably

**5. Enables Custom Columns, DAX, and Transformations**

* You can add calculated columns, format data, or enrich dimensions easily
* DirectQuery restricts many of these features

**🧠 Summary Table**

| **Benefit** | **Why Import for Dimensions?** |
| --- | --- |
| 🔗 Stable and smaller | Doesn’t need frequent refresh |
| ⚡ High performance | Joins and slicers work faster |
| 🚫 Avoids DirectQuery limits | Bypass query folding and chained queries |
| ✅ Full DAX & modeling flexibility | Calculated columns, hierarchies, etc. |

**DAX CHEAT SHEET**

**🔢 Aggregate Functions**

1. **SUM**
   * **Syntax: SUM(column)**
   * **Description: Returns the sum of all the numbers in a column.**
2. **AVERAGE**
   * **Syntax: AVERAGE(column)**
   * **Description: Returns the average of the values in a column.**
3. **MIN**
   * **Syntax: MIN(column)**
   * **Description: Returns the smallest value in a column.**
4. **MAX**
   * **Syntax: MAX(column)**
   * **Description: Returns the largest value in a column.**
5. **COUNT**
   * **Syntax: COUNT(column)**
   * **Description: Counts the number of rows in the specified column that contain non-blank values.**
6. **COUNTROWS**
   * **Syntax: COUNTROWS(table)**
   * **Description: Returns the number of rows in a table.**
7. **COUNTBLANK**
   * **Syntax: COUNTBLANK(column)**
   * **Description: Counts the number of blank values in a column.**
8. **DISTINCTCOUNT**
   * **Syntax: DISTINCTCOUNT(column)**
   * **Description: Counts the number of distinct values in a column.**
   * DISTINCTCOUNT function counts the BLANK value that means if your column contains 3 blank value, it will be counted 1 . To skip the BLANK value, use the [DISTINCTCOUNTNOBLANK](https://learn.microsoft.com/en-us/dax/distinctcountnoblank-function-dax) function.
   * This function is not supported for use in DirectQuery mode when used in calculated columns or row-level security (RLS) rules.
9. **SUMX**
   * **Syntax: SUMX(table, expression)**
   * **Description: Returns the sum of an expression evaluated over a table.**
   * **Table The table containing the rows for which the expression will be evaluated.**
   * **Expression The expression to be evaluated for each row of the table.**
   * The SUMX function takes as its first argument a table, or an expression that returns a table. The second argument is a column that contains the numbers you want to sum, or an expression that evaluates to a column.
   * The SUMX is an [iterator function](https://learn.microsoft.com/en-us/dax/dax-glossary#iterator-function).
   * Only the numbers in the column are counted. Blanks, logical values, and text are ignored.
   * This function is not supported for use in DirectQuery mode when used in calculated columns or row-level security (RLS) rules.

In one line if you want to the diff between sum and sumx ..in sum function we only pass one column ,,

But in sumxwe can pass expression like sheet1[col1] \* sheet1[col2] ..  
and one more difference is we cant pass table in sum, but in sumx we can pass table ,so we can put filter function on table if we need specific rows.

1. **AVERAGEX**

* **Syntax: AVERAGEX(table, expression)**
* **Description: Returns the average of an expression evaluated over a table.**
* **Table Name of a table, or an expression that specifies the table over which the aggregation can be performed.**
* **Expression An expression with a scalar result, which will be evaluated for each row of the table in the first argument.**

**Remarks**

* The AVERAGEX function enables you to evaluate expressions for each row of a table, and then take the resulting set of values and calculate its arithmetic mean. Therefore, the function takes a table as its first argument, and an expression as the second argument.
* In all other respects, AVERAGEX follows the same rules as AVERAGE. You cannot include non-numeric or null cells. Both the table and expression arguments are required.
* When there are no rows to aggregate, the function returns a blank. When there are rows, but none of them meet the specified criteria, then the function returns 0.
* This function is not supported for use in DirectQuery mode when used in calculated columns or row-level security (RLS) rules.

1. **MINX**

* **Syntax: MINX(table, expression)**
* **Description: Returns the minimum value of an expression over a table.**

1. **MAXX**

* **Syntax: MAXX(table, expression)**
* **Description: Returns the maximum value of an expression over a table.**
* The table argument to the MAXX function can be a table name or an expression that evaluates to a table. The second argument indicates the expression to be evaluated for each row of the table.
* Of the values to evaluate, only the following are counted:
  + Numbers
  + Texts
  + Dates
* If the expression has variant or mixed value types such as text and number, then by default MAXX considers only numbers. If <variant> = TRUE``, the maximum value is returned.
* This function is not supported for use in DirectQuery mode when used in calculated columns or row-level security (RLS) rules.
* The following formula uses an expression as the second argument to calculate the total amount of taxes and shipping for each order in the InternetSales table. The expected result is 375.7184.
* = MAXX(InternetSales, InternetSales[TaxAmt]+ InternetSales[Freight])

The following formula first filters the InternetSales table by using a FILTER expression to return a subset of orders for a specific sales region defined as, [SalesTerritory] = 5. The MAXX function then evaluates the expression used as the second argument for each row of the filtered table and returns the highest amount for taxes and shipping for just those orders. The expected result is 250.3724

= MAXX(FILTER(InternetSales,[SalesTerritoryCode]="5"), InternetSales[TaxAmt]+ InternetSales[Freight])

In one line if you want to the diff between max and maxx ..in max function we only pass one column ,,

But in maxx we can pass expression like sheet1[col1] \* sheet1[col2] ..  
and one more difference is we cant pass table in max , but in maxx we can pass table ,so we can put filter function on table if we need specific rows.

**📅 Date & Time Functions**

1. **TODAY**
   * **Syntax: TODAY()**
   * **Description: Returns the current date.**
2. **NOW**
   * **Syntax: NOW()**
   * **Description: Returns the current date and time.**
3. **YEAR**
   * **Syntax: YEAR(date)**
   * **Description: Extracts the year from a date.**
4. **MONTH**
   * **Syntax: MONTH(date)**
   * **Description: Extracts the month from a date.**
5. **DAY**
   * **Syntax: DAY(date)**
   * **Description: Extracts the day of the month from a date.**
6. **WEEKDAY**
   * **Syntax: WEEKDAY(date, [return\_type])**
   * **Description: Returns the day of the week for a date.**
7. **WEEKNUM**
   * **Syntax: WEEKNUM(date, [return\_type])**
   * **Description: Returns the week number for a date.**
8. **HOUR**
   * **Syntax: HOUR(datetime)**
   * **Description: Extracts the hour from a time.**
9. **MINUTE**
   * **Syntax: MINUTE(datetime)**
   * **Description: Extracts the minute from a time.**
10. **SECOND**
    * **Syntax: SECOND(datetime)**
    * **Description: Extracts the second from a time.**
11. **DATE**
    * **Syntax: DATE(year, month, day)**
    * **Description: Returns a date from individual year, month, and day values.**
12. **DATEDIFF**
    * **Syntax: DATEDIFF(start\_date, end\_date, interval)**
    * **Description: Returns the difference between two dates.**
    * [**https://learn.microsoft.com/en-us/dax/datediff-function-dax**](https://learn.microsoft.com/en-us/dax/datediff-function-dax)
13. **EDATE**
    * **Syntax: EDATE(start\_date, months)**
    * **Description: Returns the date that is the specified number of months before or after the start date.**
14. **EOMONTH**
    * **Syntax: EOMONTH(start\_date, months)**
    * **Description: Returns the last day of the month before or after the start date.**
15. **FORMAT**
    * **Syntax: FORMAT(value, format\_string)**
    * **Description: Converts a value to text in the specified format.**
    * **= FORMAT( 12345.67, "General Number")**
    * **= FORMAT( 12345.67, "Currency")**
    * **= FORMAT( 12345.67, "Fixed")**
    * **= FORMAT( 12345.67, "Standard")**
    * **= FORMAT( 12345.67, "Percent")**
    * **= FORMAT( 12345.67, "Scientific")**
16. **CALENDAR**
    * **Syntax: CALENDAR(start\_date, end\_date)**
    * **Description: Returns a table with a single column of dates.**
    * **= CALENDAR (DATE (2015, 1, 1), DATE (2021, 12, 31))**
17. **CALENDARAUTO**
    * **Syntax: CALENDARAUTO([fiscal\_year\_end\_month])**
    * **Description: Returns a table with one column of dates automatically based on data.**

This function is not supported for use in DirectQuery mode when used in calculated columns or row-level security (RLS) rules.

1. **DATEVALUE**
   * **Syntax: DATEVALUE(date\_text)**
   * **Description: Converts a date in text format to a date value.**

**🧠 Logical Functions**

1. **IF**
   * **Syntax: IF(condition, true\_result, false\_result)**
   * **Description: Returns one value if a condition is true and another if it's false.**
2. **IFERROR**
   * **Syntax: IFERROR(value, value\_if\_error)**
   * **Description: Returns a value if there's no error; otherwise, returns an alternate value.**
3. **SWITCH**
   * **Syntax: SWITCH(expression, value1, result1, ..., else\_result)**
   * **Description: Evaluates an expression against a list of values and returns results accordingly.**
4. **AND**
   * **Syntax: AND(condition1, condition2)**
   * **Description: Returns TRUE if both conditions are true.**
5. **OR**
   * **Syntax: OR(condition1, condition2)**
   * **Description: Returns TRUE if any condition is true.**
6. **NOT**
   * **Syntax: NOT(condition)**
   * **Description: Returns the opposite of a logical value.**
7. **ISBLANK**
   * **Syntax: ISBLANK(value)**
   * **Description: Checks whether a value is blank.**
8. **ISNUMBER**
   * **Syntax: ISNUMBER(value)**
   * **Description: Checks if a value is a number.**
9. **ISTEXT**
   * **Syntax: ISTEXT(value)**
   * **Description: Checks if a value is text.**
10. **TRUE**
    * **Syntax: TRUE()**
    * **Description: Returns the logical value TRUE.**
11. **FALSE**
    * **Syntax: FALSE()**
    * **Description: Returns the logical value FALSE.**

**🧮 Mathematical Functions**

1. **ABS**
   * **Syntax: ABS(number)**
   * **Description: Returns the absolute value of a number.**
2. **ROUND**
   * **Syntax: ROUND(number, num\_digits)**
   * **Description: Rounds a number to a specified number of digits.**
3. **ROUNDUP**
   * **Syntax: ROUNDUP(number, num\_digits)**
   * **Description: Rounds a number up, away from zero.**
4. **ROUNDDOWN**
   * **Syntax: ROUNDDOWN(number, num\_digits)**
   * **Description: Rounds a number down, toward zero.**
5. **CEILING**
   * **Syntax: CEILING(number, significance)**
   * **Description: Rounds a number up to the nearest multiple of significance.**
6. **FLOOR**
   * **Syntax: FLOOR(number, significance)**
   * **Description: Rounds a number down to the nearest multiple of significance.**
7. **MOD**
   * **Syntax: MOD(number, divisor)**
   * **Description: Returns the remainder after a number is divided.**
8. **DIVIDE**
   * **Syntax: DIVIDE(numerator, denominator, [alternateResult])**
   * **Description: Performs division and handles divide-by-zero errors.**
9. **POWER**
   * **Syntax: POWER(number, power)**
   * **Description: Returns the result of a number raised to a power.**
10. **INT**
    * **Syntax: INT(number)**
    * **Description: Rounds a number down to the nearest integer.**
11. **EXP**
    * **Syntax: EXP(number)**
    * **Description: Returns e raised to the power of the number.**
12. **LOG**
    * **Syntax: LOG(number, base)**
    * **Description: Returns the logarithm of a number to the specified base.**
13. **LN**
    * **Syntax: LN(number)**
    * **Description: Returns the natural logarithm of a number.**
14. **SQRT**
    * **Syntax: SQRT(number)**
    * **Description: Returns the square root of a number.**

**✏️ Text Functions**

1. **CONCATENATE**
   * **Syntax: CONCATENATE(text1, text2)**
   * **Description: Joins two text strings into one text string.**
2. **CONCATENATEX**
   * **Syntax: CONCATENATEX(table, expression, [delimiter], [order\_by\_expression], [order])**
   * **Description: Concatenates the result of an expression evaluated over a table.**
3. **LEFT**
   * **Syntax: LEFT(text, num\_chars)**
   * **Description: Returns the leftmost characters from a text value.**
4. **RIGHT**
   * **Syntax: RIGHT(text, num\_chars)**
   * **Description: Returns the rightmost characters from a text value.**
5. **MID**
   * **Syntax: MID(text, start\_num, num\_chars)**
   * **Description: Returns a substring from the middle of a text string.**
6. **LEN**
   * **Syntax: LEN(text)**
   * **Description: Returns the number of characters in a text string.**
7. **UPPER**
   * **Syntax: UPPER(text)**
   * **Description: Converts text to uppercase.**
8. **LOWER**
   * **Syntax: LOWER(text)**
   * **Description: Converts text to lowercase.**
9. **TRIM**
   * **Syntax: TRIM(text)**
   * **Description: Removes all spaces from text except for single spaces between words.**
10. **REPLACE**
    * **Syntax: REPLACE(old\_text, start\_num, num\_chars, new\_text)**
    * **Description: Replaces part of a text string with a different text string.**
11. **SUBSTITUTE**
    * **Syntax: SUBSTITUTE(text, old\_text, new\_text, [instance\_num])**
    * **Description: Substitutes new text for old text in a string.**
12. **SEARCH**
    * **Syntax: SEARCH(find\_text, within\_text, [start\_num])**
    * **Description: Returns the position of one text string within another.**
13. **FIND**
    * **Syntax: FIND(find\_text, within\_text, [start\_num])**
    * **Description: Returns the starting position of one text string within another. Case-sensitive.**
14. **TEXT**
    * **Syntax: TEXT(value, format\_text)**
    * **Description: Formats a number and converts it to text.**
15. **VALUE**
    * **Syntax: VALUE(text)**
    * **Description: Converts a text string that represents a number to a number.**

**🔍 Filter Functions**

1. **FILTER**

* **Syntax: FILTER(table, filter\_expression)**
* **Description: Returns a table that represents a subset of another table.**
* **Table The table to be filtered. The table can also be an expression that results in a table.**
* **filter** **A Boolean expression that is to be evaluated for each row of the table. For example, [Amount] > 0 or [Region] = "France"**
* You can use FILTER to reduce the number of rows in the table that you are working with, and use only specific data in calculations. FILTER is not used independently, but as a function that is embedded in other functions that require a table as an argument.
* This function is not supported for use in DirectQuery mode when used in calculated columns or row-level security (RLS) rules.

1. **ALL**

* **Syntax: ALL(tableOrColumn, ...)**
* **Description: Removes filters from one or more columns or tables.**
* **Returns all the rows in a table, or all the values in a column, ignoring any filters that might have been applied. This function is useful for clearing filters and creating calculations on all the rows in a table**
* This function is not used by itself, but serves as an intermediate function that can be used to change the set of results over which some other calculation is performed.
* **ALLEXCEPT**
* **Syntax: ALLEXCEPT(table, column1, column2, …)**
* **Description: Removes all context filters except those that have been applied to the specified columns.**
* [**https://learn.microsoft.com/en-us/dax/all-function-dax**](https://learn.microsoft.com/en-us/dax/all-function-dax)

1. **ALLSELECTED (https://www.youtube.com/watch?v=XXwpzxqya3o)**

* **Syntax: ALLSELECTED([table\_or\_column])**
* **Description: Returns all values in a column or table under the current filters.**

1. **VALUES**

* **Syntax: VALUES(column)**
* **Description: Returns a one-column table that contains the distinct values from the specified column.**
* **Duplicate values are removed and only unique values are returned. A BLANK value can be added. When the input parameter is a table name, returns the rows from the specified table. Duplicate rows are preserved. A BLANK row can be added.**
* **When You Use VALUES(Table)**

This returns **the entire table** Products, but:

* It **does not remove** duplicate rows.
* It **preserves** all rows, even if some are repeated.

| * **ID** | * **Category** |
| --- | --- |
| * **1** | * **Electronics** |
| * **2** | * **Furniture** |
| * **3** | * **Electronics** |
| * **3** | * **Electronics** |

* **VALUES(Products)**

| * **ID** | * **Category** |
| --- | --- |
| * **1** | * **Electronics** |
| * **2** | * **Furniture** |
| * **3** | * **Electronics** |
| * **3** | * **Electronics** |

* All rows are **preserved** — including duplicates.

| **Syntax** | **Output** | **Duplicates** | **Use Case Example** |
| --- | --- | --- | --- |
| **VALUES(Column)** | **One-column table with unique values** | **❌ Removed** | **For slicers, counting unique categories** |
| **VALUES(Table)** | **Full table with all rows** | **✅ Preserved** | **Used in advanced filtering** |

1. **DISTINCT**

* **Syntax: DISTINCT(column)**
* **Description: Returns a one-column table that contains the distinct values in a column.**
* **It consider null value one time**

1. **KEEPFILTERS**

* **Syntax: KEEPFILTERS(expression)**
* **Description: Modifies the filter context applied by CALCULATE.**

1. **REMOVEFILTERS**

* **Syntax: REMOVEFILTERS([table\_or\_column])**
* **Description: Removes all filters from the specified columns or tables.**

1. **CROSSFILTER**

* **Syntax: CROSSFILTER(column1, column2, direction)**
* **Description: Specifies the cross-filtering direction to be used in a calculation.**

1. **USERELATIONSHIP**

* **Syntax: USERELATIONSHIP(column1, column2)**
* **Description: Specifies an existing relationship to be used in a calculation.**

**🔁 Iterator Functions**

1. **ADDCOLUMNS**

* **Syntax: ADDCOLUMNS(table, name, expression, ...)**
* **Description: Returns a table with new columns specified by expressions.**

1. **SELECTCOLUMNS**

* **Syntax: SELECTCOLUMNS(table, name, expression, ...)**

1. **Description: Returns a table with specific columns defined by expressions.**

| **Function** | **Grouping** | **Custom Columns** | **Measures** | **Best For** |
| --- | --- | --- | --- | --- |
| **SUMMARIZE** | **✅ Yes** | **✅ Yes** | **⚠️ Complex ones may need CALCULATE** | **General grouping** |
| **GROUPBY** | **✅ Yes** | **✅ Yes (with addcolumns-style expressions)** | **❌** | **Custom row-wise logic** |
| **SUMMARIZECOLUMNS** | **✅ Yes** | **✅ Yes** | **✅ Direct measure support** | **Better performance, used in measures** |

**GENERATE**

* **Syntax: GENERATE(table1, table2)**
* **Description: Returns a table by evaluating a table expression for each row of another table.**

1. **CROSSJOIN**

* **Syntax: CROSSJOIN(table1, table2)**
* **Description: Returns a table that is the Cartesian product of all rows from the specified tables.**

1. **UNION**

* **Syntax: UNION(table1, table2)**
* **Description: Returns a table that contains all rows from both tables.**

1. **INTERSECT**

* **Syntax: INTERSECT(table1, table2)**
* **Description: Returns a table with rows common to both tables.**

1. **EXCEPT**

* **Syntax: EXCEPT(table1, table2)**
* **Description: Returns a table with rows in the first table that do not appear in the second table.**

1. **NATURALINNERJOIN**

* **Syntax: NATURALINNERJOIN(table1, table2)**
* **Description: Returns a table with an inner join based on common columns.**

1. **NATURALLEFTOUTERJOIN**

* **Syntax: NATURALLEFTOUTERJOIN(table1, table2)**
* **Description: Returns a table with a left outer join based on common columns.**

1. **GROUPBY**

* **Syntax: GROUPBY(table, groupBy\_columnName, [name, expression]...)**
* **Description: Groups a table by one or more columns.**

1. **SUMMARIZE**

* **Syntax: SUMMARIZE(table, groupBy\_columnName, name, expression, ...)**
* **Description: Returns a summary table for the requested totals over a set of groups.**

1. **SUMMARIZECOLUMNS**

* **Syntax: SUMMARIZECOLUMNS(columnName, ...)**
* **Description: Returns a summary table over a set of groups.**

1. **ADDMISSINGITEMS**

* **Syntax: ADDMISSINGITEMS(table1, table2)**
* **Description: Ens**
* **ISFILTERED**
  + **Syntax:** ISFILTERED(columnName)
  + **Description:** Returns TRUE if the specified column is being filtered directly.
* **HASONEVALUE**
  + **Syntax:** HASONEVALUE(columnName)
  + **Description:** Returns TRUE when the context for the column has been filtered down to one distinct value.
* **PATH**
  + **Syntax:** PATH(parent, child)
  + **Description:** Returns a delimited text string with the identifiers of all the parents to the current identifier.
* **PATHITEM**
  + **Syntax:** PATHITEM(path, position, [type])
  + **Description:** Returns the item at the specified position from a PATH string.
* **PATHCONTAINS**
  + **Syntax:** PATHCONTAINS(path, item)
  + **Description:** Indicates whether a PATH contains a specific item.
* **PATHLENGTH**
  + **Syntax:** PATHLENGTH(path)
  + **Description:** Returns the length of the PATH string.
* **USERELATIONSHIP**  
  - **Syntax:** USERELATIONSHIP(column1, column2)  
  - **Description:** Specifies an existing relationship to be used in a calculation that otherwise isn't active.

1. **what is measure and calculated Column in power bi**

In Power BI, **Measures** and **Calculated Columns** are both used to perform calculations using **DAX (Data Analysis Expressions)**, but they serve different purposes and are evaluated at different times. Here's a breakdown of their differences:

**What it is:  
A calculated column is a new column added to your data model, created using a DAX expression.**

**When it's evaluated:  
It is calculated row by row at data refresh time (i.e., when the model is loaded or refreshed).**

**Where it's stored:  
The values are stored in memory in the data model, increasing the size of your model.**

**Use case examples:**

* **Creating a new column for Year from a Date column.**
* **Measure**
* **What it is:**  
  A **measure** is a DAX formula used to perform **aggregations or calculations** on your data.
* **When it's evaluated:**  
  It is calculated **on the fly at query time**, based on user interaction with the report (like filters or slicers).
* **Where it's stored:**  
  Only the DAX expression is stored — **no data is physically stored** in the model for a measure.

**Use case examples:**

* **Calculating total sales, average, count, etc.**
* **Dynamic KPIs based on filters or slicers.**
* **Ratios, trends, growth percentages.**

| **Feature** | **Calculated Column** | **Measure** |
| --- | --- | --- |
| **Evaluated at** | **Data load/refresh** | **Query/report interaction** |
| **Stored in model** | **Yes (increases size)** | **No (just the formula)** |
| **Works row by row?** | **Yes** | **No (works on aggregates)** |
| **Use for** | **Filters, relationships, visuals** | **Aggregations, KPIs, performance** |

**When to Use What?**

* **Use Calculated Columns when:**
  + **You need to filter, slice, or group data by that column.**

**Use Measures when:**

* **You need aggregated values (SUM, COUNT, AVG, etc.).**
* **You want to calculate dynamic values that change based on filters or visuals.**

**How to handle missing value in Power BI:**

Handling missing values (like nulls or blanks) in Power BI is a common task in data cleaning and modeling. You can handle them either in the Power Query Editor (data preparation stage) or using DAX (data modeling/reporting stage), depending on your needs.

**In Power Query Editor (Data Preparation Stage)**

Open Power Query:  
Click **Home > Transform Data** to open the **Power Query Editor**.

**🛠️ Options to Handle Missing Values:**

**✅ Remove Rows with Missing Values**

* Go to the column with missing values.
* Click **Remove Rows > Remove Blank Rows**.
* This deletes all rows where that column is null/blank
* **Using DAX (During Reporting/Modeling)**
* When you don't want to change the raw data, but want to handle blanks dynamically in your report visuals or calculated columns/measures.

CleanValue = IF(ISBLANK([Column]), "Unknown", [Column])

COALESCE() in Power BI (DAX

COALESCE() is a DAX function that returns the **first non-blank value** from a list of expressions. It's often used to handle missing values (BLANK()) in calculated columns or measures.

COALESCE(expression1, expression2, ..., expressionN)

* It checks each expression **from left to right** and returns the **first value that is not blank**.
* If all expressions are blank, it returns **blank**.
* **Replace Blank Values with Default**
* Cleaned Sales = COALESCE(Sales[Amount], 0)

What is a KPI in Power BI?

**KPI** stands for **Key Performance Indicator**. It is a measurable value that indicates how effectively a business is achieving its key objectives. In **Power BI**, a **KPI visual** is used to track the **performance of a metric** over time compared to a **target**.

**✅ Purpose of a KPI in Power BI**

* Monitor the **progress** of a business goal (e.g., revenue, customer acquisition).
* Quickly **visualize performance** (actual vs. target).
* Provide a **snapshot** of current status using colors and symbols (e.g., red, green, arrows).

**How to Create a KPI in Power BI**

1. **Prepare your data** with:
   * A **value measure** (e.g., Total Sales).
   * A **target measure** (e.g., Sales Goal).
   * A **trend axis** (e.g., Date).
2. **Insert the KPI visual**:
   * Go to **Visualizations pane** → Click the **KPI icon**.
   * Drag the following:
     + **Indicator** → the main measure (e.g., Total Sales)
     + **Trend axis** → a date/time field
     + **Target goal** → a measure or field representing the goal (e.g., Sales Target)

| **Month** | **Total Sales** | **Sales Target** |
| --- | --- | --- |
| Jan | 80,000 | 100,000 |
| Feb | 110,000 | 100,000 |
| Mar | 95,000 | 100,000 |

In a KPI visual:

* Feb will show **green** (over target)
* Jan and Mar will show **red** (under target)

**Benefits of Using KPI in Power BI**

* Clear and **intuitive visual feedback** (good/bad/neutral).
* Helps management make **quick decisions**.
* Can display **trends** over time to monitor progress

What is SCD in Power BI?

**SCD** stands for **Slowly Changing Dimension**. It’s a data warehousing concept used to **track changes in dimension data over time** — such as changes in customer names, addresses, job titles, etc.

In **Power BI**, handling SCD is essential when analyzing **historical data** that needs to reflect the correct version of dimension values **at the time a transaction occurred**.

**Why is SCD important?**

Imagine a customer changed their location. You want:

* **Historical sales reports** to show the **old location** for older transactions.
* And for new transactions to reflect the **updated location**.

| **SCD Type** | **Description** |
| --- | --- |
| **Type 1** | Overwrite old data with new (no history preserved) |
| **Type 2** | Keep full history (store new record with timestamp/versioning) |
| **Type 3** | Keep limited history (add new columns for previous values) |

**How to Handle SCD in Power BI**

**🔹 SCD Type 1: Overwrite (No History)**

* Simply update the record in the dimension table.
* In Power BI, import the updated dimension — no extra logic needed.
* Suitable when history isn't important.

**SCD Type 2: Maintain History (Most Common)**

This is where things get more complex — and **Power BI doesn’t handle SCD2 automatically**, but you can implement it using:

**1. Create a Historical Dimension Table**

The dimension table should have:

* A **StartDate** and **EndDate**
* A **CurrentFlag** (optional)
* A **SurrogateKey** (unique for each version of a record)

| **CustomerKey** | **CustomerID** | **Name** | **City** | **StartDate** | **EndDate** |
| --- | --- | --- | --- | --- | --- |
| 1 | 101 | John Doe | New York | 2020-01-01 | 2022-06-30 |
| 2 | 101 | John Doe | Boston | 2022-07-01 | 9999-12-31 |

**2. Fact Table Includes TransactionDate and CustomerID**

Join it to the historical dimension using **DAX or Power Query logic**:

RELATED(Customer[City]) -- works only if the correct dimension version is matched

**Advanced join logic:** You may need to use **bi-directional filtering** or **custom filtering** in measures to match TransactionDate between fact and dimension.

**📌 In Power Query:**

* You can merge the fact table with the dimension table using:
  + CustomerID
  + TransactionDate between StartDate and EndDate

Top Techniques to Make Power BI Interactive:

1. 🔘 **Use Slicers**

 **Slicers** act like filters users can click on.

 Examples: Date ranges, categories, regions, product names.

Enable Drill-Down/Drill-Through

**Drill-Down**

* Add hierarchy in visuals (e.g., Year > Quarter > Month).
* Users can click to "drill down" into more detail.

**How:** Add multiple fields to the **Axis** area and use the drill icons on top of the visual.

**🔁 Drill-Through**

* Create a new page with a detailed view.
* Add a field to the **Drill-through filter pane**.
* Users right-click a visual → drill into another page for more context.

**Use Bookmarks + Buttons**

**4. 🎛 Use Tooltips and Report Pages**

* Tooltips display extra details when hovering over a data point.
* You can even create **custom tooltip report pages** with charts and KPIs

Use Conditional Formatting

* Dynamically change color, size, or font based on values.
* Apply to bars, data labels, or background colors to highlight trends or risks.

**5. 🎨 Use Conditional Formatting**

* Dynamically change color, size, or font based on values.
* Apply to bars, data labels, or background colors to highlight trends or risks.

**Example:** Highlight low sales in red, high in green.

**Enable Sync Slicers Across Pages**

* Slicers can control visuals on **multiple pages**.

**How:** Go to **View > Sync Slicers Pane** → check "Sync" and "Visible".

**Create Date or Time Intelligence Filters**

* Use **relative date slicers** (e.g., “last 7 days”) for dynamic analysis.
* Add DAX measures like:

**Use What-If Parameters**

* Allow users to input values and **see how metrics change** (e.g., forecast growth, price changes).

Go to **Modeling > New Parameter** to create one.

**onus: Combine Features**

Imagine a report where:

* Users select a **region** with a slicer,
* Then **drill-through** to customer-level sales,
* With a **tooltip** showing profit margin,
* And **bookmarks** to switch between visuals.

This creates a dynamic, exploratory experience!

| **Goal** | **Feature to Use** |
| --- | --- |
| Let user filter data | Slicers, filters |
| Navigate between views | Bookmarks + buttons |
| See more detail | Drill-down & drill-through |
| Highlight insights | Conditional formatting |
| Compare scenarios | What-if parameters |

**🎨 What is a Custom Visual in Power BI?**

A **Custom Visual** in Power BI is a visual component **developed outside the standard set** of visuals provided by Microsoft. These are created by Microsoft, third-party developers, or the community and can be **imported into your reports** to expand Power BI’s visual capabilities.

**✅ Why Use Custom Visuals?**

While built-in visuals (bar chart, table, pie chart, etc.) are enough for many cases, **custom visuals** are used when:

* Built-in visuals **don’t meet specific business requirements**
* You need **advanced data visualization** or **interactive storytelling**
* You want to create **richer, more intuitive** dashboards

**📦 Where to Get Custom Visuals**

* **Microsoft AppSource**: [https://appsource.microsoft.com](https://appsource.microsoft.com/)
* In Power BI Desktop:  
  👉 Go to **Visualizations Pane** → **“...” > Get more visuals**

**📌 Popular Custom Visual Examples & Use Cases**

| **Visual Type** | **Use Case** |
| --- | --- |
| **Bullet Chart** | Better KPI tracking (vs. simple bar/target chart) |
| **Chiclet Slicer** | Image-based slicers (e.g., country flags, product images) |
| **Hierarchy Slicer** | Multi-level slicer for complex hierarchies (e.g., Country > State > City) |
| **Gantt Chart** | Project management timelines, task tracking |
| **Timeline Slicer** | Advanced date selection with better UI |
| **KPI Indicator by MAQ Software** | Custom formatting and icons for KPI values |
| **Sunburst or Sankey Diagrams** | Relationship flows, process mapping, or multi-level breakdowns |
| **Radar Chart** | Skill matrix, performance comparison across metrics |
| **Word Cloud** | Text analysis from surveys, reviews, or comments |

**🛠️ Custom Visual Use Case Scenarios**

**🔹 1. Project Tracking with Gantt Chart**

* Visualize tasks, timelines, owners, and progress in a visual timeline.
* Helps teams see dependencies and bottlenecks clearly.

**🔹 2. Image-based Filtering with Chiclet Slicer**

* Use logos or icons instead of plain text to make dashboards more user-friendly.
* Example: Select product brands by clicking on their logo.

**🔹 3. Flow Analysis with Sankey or Funnel Visuals**

* Show conversion or transaction flows (e.g., marketing funnel, website click paths).
* Great for journey or process mapping.

**🔹 4. Performance Tracking with Radar Chart**

* Compare performance across multiple KPIs.
* Useful for employee skills, product performance, etc.

**🔐 Are Custom Visuals Safe?**

* Microsoft **certifies** some visuals (they have a blue tick ✔️).
* Certified visuals do **not access external services** and are tested for security.
* Non-certified visuals may require extra caution, especially in enterprise environments.

**🧠 Summary**

| **Feature** | **Custom Visuals in Power BI** |
| --- | --- |
| Purpose | Add new types of visuals beyond built-in options |
| How to use | Import from AppSource or visual marketplace |
| Best for | Advanced visuals, better UX, domain-specific needs |
| Security | Prefer **certified visuals** for enterprise use |

Pivot and Unpivot in power BI:

| **Operation** | **Purpose** | **Analogy** |
| --- | --- | --- |
| **Pivot** | Turns **rows into columns** | Like a matrix transpose (flattened) |
| **Unpivot** | Turns **columns into rows** | Makes wide tables into long format |

**UNPIVOT**

**✅ What It Does:**

* Converts **multiple columns** into **row values** under one column.
* Helps normalize wide data into a long format for better analysis.

**What It Does:**

* Converts values in a **column** into **column headers**.
* Useful when you want to summarize or reshape a dataset.

In **Power BI**, a **Quick Measure** is a pre-built DAX expression that you can use to perform common calculations without having to write complex DAX code manually.

**✅ How to Create a Quick Measure in Power BI:**

1. **Open Power BI Desktop.**
2. In the **Fields** pane, right-click on a table (e.g., Sales).
3. Select **New Quick Measure**.
4. In the **Quick Measures** dialog:
   * Choose a **calculation type** (e.g., Filtered value, Aggregation, Time Intelligence).
   * Select the **fields** and **conditions** as required.
5. Power BI will auto-generate the **DAX formula** and create a measure in your model.

**✅ TOPN Syntax in DAX (Power BI)**

TOPN(<n\_value>, <table>, <orderBy\_expression>[, <order>[, <orderBy\_expression>, <order>]…])

Top 5 customers by Sales Amount

TOPN(

5,

Customers,

Customers[Total Sales]

)

Top 3 products by Quantity sold (ascending)

TOPN(

3,

Products,

Products[Quantity Sold],

ASC

)

3. **Use TOPN with CALCULATE to compute total sales for top 10 products**

CALCULATE(

SUM(Sales[Sales Amount]),

TOPN(

10,

Products,

Products[Sales Amount]

)

)

**Notes:**

* TOPN returns a **table**, not a scalar value.
* You can **nest** it inside other functions like CALCULATE, SUMX, etc.
* Supports **multiple columns** for ordering.

| **Function** | | **Description** | | **Output** |
| --- | --- | --- | --- | --- |
| PATH | | Returns the full path from child to top parent | | String |
| PATHLENGTH | | Counts the number of levels in a path | | Integer |
| PATHITEM | | Returns the item at a given position in the path | | String/Int |
| PATHCONTAINS | | Checks if an ID is part of the path | | TRUE/FALSE |
|  |  | |
|  |  | |
| **EmployeeID** | **ParentEmployeeID** | |
| 1 (CEO) | (blank) | |
| 2 | 1 | |
| 3 | 2 | |

**PATH(child, parent)**

**✅ Purpose:**

Returns a **delimited string** that represents the **lineage** (or path) from a child to its top-most parent.

**🧾 Syntax:**

PATH(<child\_id>, <parent\_id>)

For EmployeeID = 3, this DAX:

PATH(Employee[EmployeeID], Employee[ParentEmployeeID])

Returns:

"3|2|1"

It shows the full **reporting path** from the employee up to the CEO.

**2. PATHLENGTH(path)**

**✅ Purpose:**

Returns the **number of levels** in a hierarchy path returned by PATH.

**🧾 Syntax:**

PATHLENGTH(<path>)

**📘 Example:**

Using the previous result:

PATHLENGTH("3|2|1")

Returns:

3

This tells you there are 3 levels from the current employee to the top (CEO).

✅ **Use case**: Find how "deep" an employee is in the hierarchy.

**3. PATHITEM(path, position, [type])**

**✅ Purpose:**

Returns the **item at a specific position** in a hierarchy path.

**🧾 Syntax:**

dax

CopyEdit

PATHITEM(<path>, <position>, [<type>])

* position: 1-based index (1 = first item)
* type: Optional. Use INTEGER or TEXT. Default is TEXT.

**📘 Example:**

PATHITEM("3|2|1", 2, INTEGER)

Returns:

2

It returns the **2nd value** in the path as an integer.

✅ **Use case**: Determine who an employee reports to at a certain level (e.g., direct manager).

**4. PATHCONTAINS(path, item)**

**✅ Purpose:**

Checks if a specific **item exists in the path**.

**🧾 Syntax:**

PATHCONTAINS(<path>, <item>)

**📘 Example:**

PATHCONTAINS("3|2|1", 2)

Returns:

TRUE

✅ **Use case**: Check if a person is in another person's reporting chain.

**What is a DAX Measure?**

A **measure** is a calculation used in reports, such as **SUM**, **AVERAGE**, **COUNT**, or more advanced expressions using **DAX (Data Analysis Expressions)**.

**🔍 Implicit vs Explicit Measures**

| **Feature** | **Implicit Measure** | **Explicit Measure** |
| --- | --- | --- |
| **Definition** | Automatically created by Power BI | Manually created using DAX |
| **Creation Method** | Dragging a field (like Sales) into a visual | Writing a DAX formula using New Measure |
| **Customizable?** | ❌ Limited — only basic aggregation options | ✅ Fully customizable (logic, filters, conditions) |
| **Reusability** | ❌ Cannot be reused in other visuals/measures | ✅ Reusable across visuals |
| **Naming** | Auto-generated (e.g., “Sum of Sales”) | User-defined (e.g., Total Sales) |
| **Performance Control** | ❌ Less optimized for complex logic | ✅ You control the logic & context |

**✅ Implicit Measure Example**

If you **drag a numeric column** (e.g., Sales[Amount]) into a chart:

* Power BI automatically applies an aggregation like SUM, COUNT, AVERAGE, etc.
* You’ll see **“Sum of Amount”** in the legend or tooltip.

You didn’t write any DAX — Power BI did it **implicitly**.

**✅ Explicit Measure Example**

Go to **Modeling → New Measure** and write:

dax

CopyEdit

Total Sales = SUM(Sales[Amount])

This is now an **explicit measure**. You can:

* Reuse it in multiple visuals
* Apply it in other DAX measures
* Use CALCULATE, FILTER, and other advanced DAX

**🎯 When to Use Each?**

| **Use Case** | **Recommended Measure Type** |
| --- | --- |
| Quick visual with basic sum/count | **Implicit** is OK |
| Complex business logic (YTD, RLS-aware, custom filters) | **Explicit** |
| Want to reuse measure across visuals | **Explicit** |
| Need formatting or naming control | **Explicit** |

Would you like me to recommend a custom visual based on your specific Power BI project or industry?