Q1) - Ranking Scenario

Simple ranking on the basis of Region

```
1  rank = RANKX(Store_DIM,[total sales],,DESC,Dense)
```

Region	total sales	rank
Western Europe	17,29,984.40	1
Central America	12,28,321.11	1
Oceania	11,01,009.71	1
Southeastern Asia	8,82,082.82	1
Southern Asia	8,69,165.10	1
Eastern Asia	8,55,059.39	1
Western US	7,49,559.98	1
Total	**1,26,42,501.91**	**1**

Use All with table's column - remove filter

```
1  rank = RANKX(ALL(Store_DIM[Region]),[total sales],,DESC,Dense)
```

Region	total sales	rank
Western Europe	17,29,984.40	1
Central America	12,28,321.11	2
Oceania	11,01,009.71	3
Southeastern Asia	8,82,082.82	4
Southern Asia	8,69,165.10	5
Eastern Asia	8,55,059.39	6
Western US	7,49,559.98	7
Total	**1,26,42,501.91**	**1**

```
1  rank = IF(HASONEVALUE(Store_DIM[Region]),RANKX(ALL(Store_DIM[Region]),[total sales],,DESC,Dense))
```

Region	total sales	rank
Western Europe	17,29,984.40	1
Central America	12,28,321.11	2
Oceania	11,01,009.71	3
Southeastern Asia	8,82,082.82	4
Southern Asia	8,69,165.10	5
Eastern Asia	8,55,059.39	6
Western US	7,49,559.98	7
Total	**1,26,42,501.91**	

ranking according to region then country

```
Hierarchy Ranking =
var RegionFiltered = ISFILTERED(Store_DIM[Region])
var CountryFiltered = ISFILTERED(Store_DIM[Country])
--case 1 - Region filtering (True)
--case 2 - Region & Country filtering (True)
var ranking = SWITCH(TRUE(),
                RegionFiltered && NOT CountryFiltered, RANKX(ALL(Store_DIM[Region]),[total sales],,DESC,Dense),
                RegionFiltered && CountryFiltered, RANKX(ALL(Store_DIM[Country]),[total sales],,DESC,Dense),
                BLANK())
return ranking

```

Region	total sales	rank	Hierarchy Ranking
Western Europe	**17,29,984.40**	**1**	**1**
France	8,56,985.81	1	1
Germany	6,28,136.19	1	2
Austria	92,539.05	1	3
Netherlands	77,514.95	1	4
Belgium	49,226.70	1	5
Switzerland	24,877.86	1	6
Luxembourg	703.85	1	7
Central America	**12,28,321.11**	**2**	**2**
Mexico	6,35,742.80	1	1
El Salvador	1,80,921.21	1	2
Nicaragua	1,37,351.07	1	3
Guatemala	1,31,602.47	1	4
Total	**1,26,42,501.91**		

IsFilter and CrossFilter

https://www.youtube.com/watch?v=fx6q5zL-InU

Filter Types

There are three types of filters that we need to consider while working wit **Isfiltered** & **Iscrossfiltered** in DAX.

1. **Direct Filter**
   - Direct filters are those filters that effect directly on the column that is being used.
2. **Indirect Filter**
   - The indirect filter is that filter, which is not the direct filter.
3. **Cross Filter**
   - CrossFilter function is neither tabular nor scalar function, It is a specific type of function that changes the direction of a relationship.

Filter Types

- **ISFILTERED DAX Function (Information)**

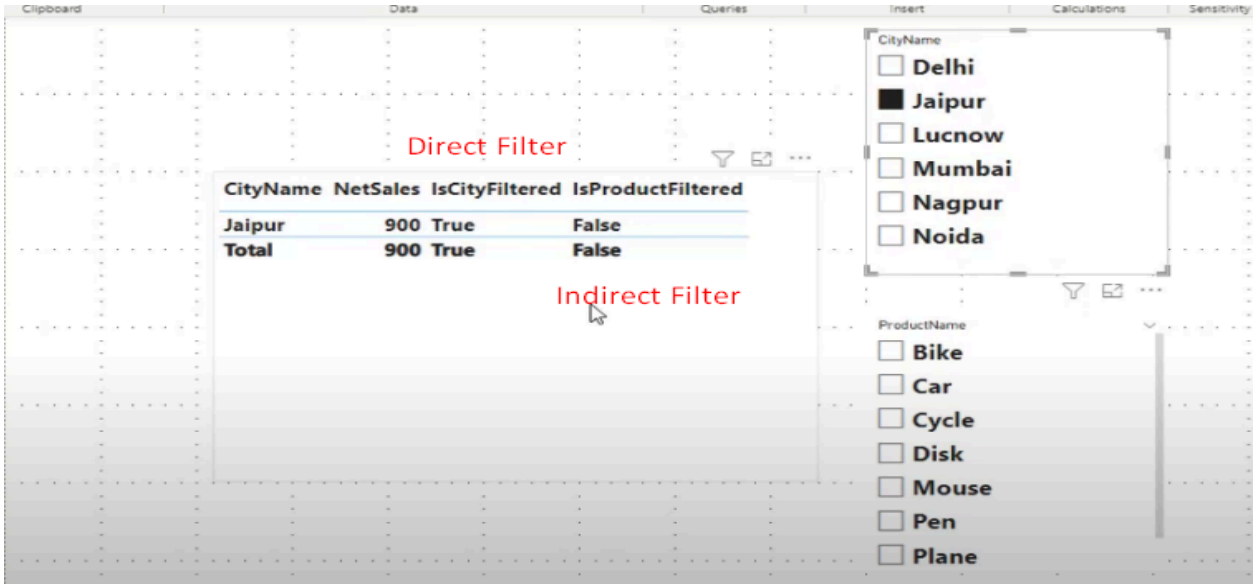
  Returns true when there are direct filters on the specified column.

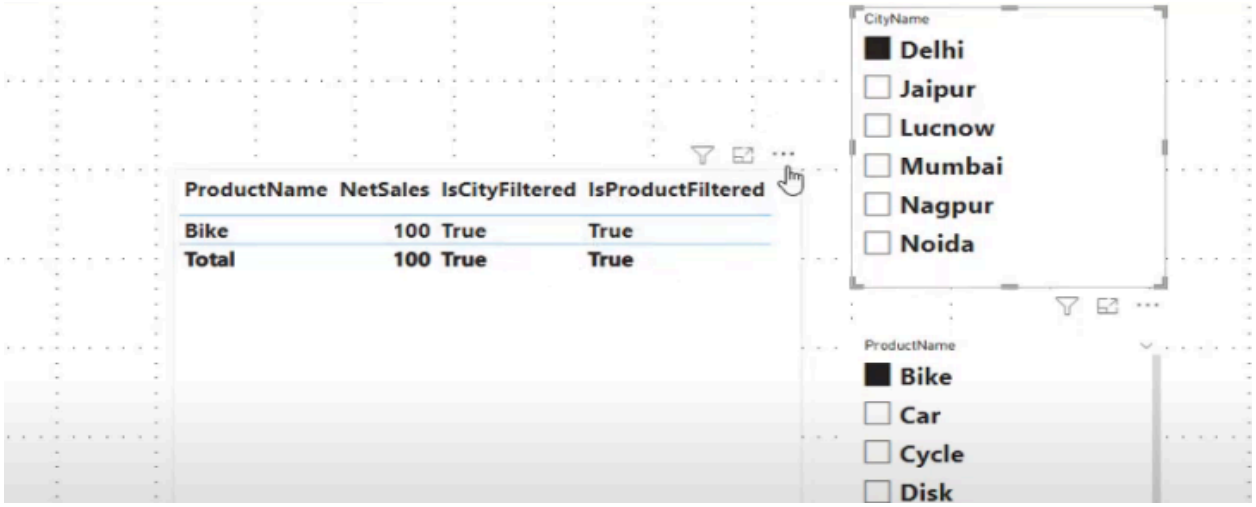
Syntax:

ISFILTERED(<TableNameOrColumnName>)

1. **Create Two measures**
   1. **IsCityFiltered = ISFILTERED(DimCity[CityName])**
   2. **IsProductFiltered = ISFILTERED(DimProduct[ProductName])**

CityName	NetSales	IsCityFiltered	IsProductFiltered
Delhi	700	True	False
Jaipur	900	True	False
Lucnow	600	True	False
Mumbai	2000	True	False
Nagpur	600	True	False
Noida	1500	True	False
Total	6300	False	False





ISCROSSFILTERED

- Returns TRUE when the specified table or column is cross-filtered.
- A column or table is said to be cross-filtered when a filter is applied to ColumnName, any column of TableName, or to any column of a related table.

Syntax

ISCROSSFILTERED(<TableNameOrColumnName>)

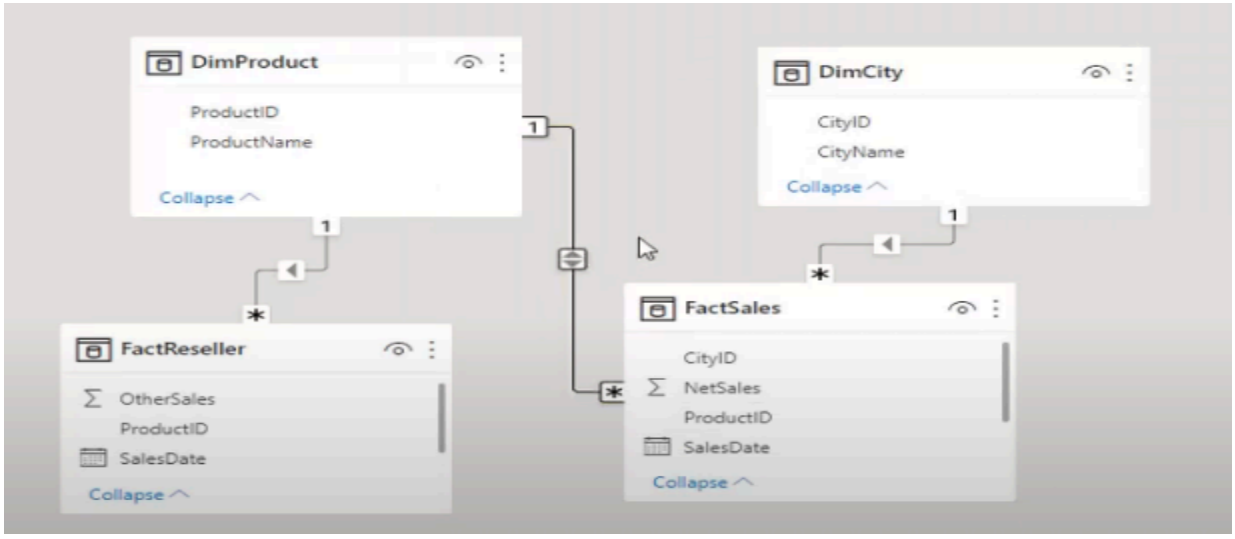
TableNameOrColumnName : The name of an existing table or column. It cannot be an expression.

Return value

TRUE when ***ColumnName*** or a column of ***TableName*** is being cross-filtered. Otherwise returns FALSE.

```
1 IsProductCrossFiltered = ISCROSSFILTERED(DimProduct[ProductName])
```

CityName	NetSales	OtherSales	IsProductCrossFilter
Delhi	700	549	False
Jaipur	900	549	False
Lucnow	600	549	False
Mumbai	2000	549	False
Nagpur	600	549	False
Noida	1500	549	False
Total	6300	549	False




```
1 IsProductCrossFiltered = ISCROSSFILTERED(DimProduct[ProductName])
```

CityName	NetSales	OtherSales	IsProductCrossFilter
Delhi	700	482	True
Jaipur	900	305	True
Lucnow	600	328	True
Mumbai	2000	372	True
Nagpur	600	221	True
Noida	1500	328	True
Total	6300	549	False

country and sales in the same table

Without All

```
1 Countries_Ranking =
2
3 RANKX(Countrywise_Sales,[Total_Sales],,DESC,Dense)
```

Country	Total_Sales	Countries_Ranking
Australia	450000	1
Germany	300000	1
India	350000	1
UK	250000	1
USA	700000	1
Total	**2050000**	**1**

With All - ignore all filter

```
1 Countries_Ranking_ALL =
2
3 RANKX(ALL(Countrywise_Sales),[Total_Sales],,DESC,Dense)
```

Country	Total_Sales	Countries_Ranking_ALL
USA	700000	1
Australia	450000	2
India	350000	3
Germany	300000	4
UK	250000	5
Total	**2050000**	**1**

With Allselected - ignore inner filter but respect outer filter

```
1 Countries_Ranking_ALLSELECTED =
2
3
4 RANKX(ALLSELECTED(Countrywise_Sales),[Total_Sales],,DESC,Dense)
```

Country	Total_Sales	Countries_Ranking_ALL	Countries_Ranking_ALLSELECTED
USA	700000	1	1
Australia	450000	2	2
India	350000	3	3
Germany	300000	4	4
UK	250000	5	5
Total	**2050000**	**1**	**1**

Allselected with slicer (outer filter)

Country	Total_Sales	Countries_Ranking_ALL	Countries_Ranking_ALLSELECTED
Australia	450000	2	1
India	350000	3	2
Germany	300000	4	3
UK	250000	5	4
Total	**1350000**		**1**

```
1 rank = RANKX(ALL(sales),SUM(sales[Sales]),,DESC,Dense)
```

Ranking Scenario - Why we need to add All, What will happen if we replace All with Allselected.

Country	Sum of Sales
Australia	450000
Germany	300000
India	350000
UK	250000
USA	700000
Total	**2050000**

Country	Sum of Sales	rank
Australia	450000	1
Germany	300000	1
India	350000	1
UK	250000	1
USA	700000	1
Total	**2050000**	**1**

SUM(sales[Sales]) calculates the sum of sales in the current context. In some scenarios, this might always return the same value if the context isn't changing correctly within the RANKX function, leading to all categories getting the same rank.

Using a measure like [total_sales] ensures that the calculation of total sales is done correctly across the desired context. Measures are context-aware and recalculated in each row of the visual, which ensures that the ranking is done based on the correct sales values.

---

2. Rank- All

```
1 rank_all = RANKX(ALL(sales),[total_sales],,DESC,Dense)
```

Ranking Scenario - Why we need to add All, What will happen if we replace All with Allselected.

Country	Sum of Sales
Australia	450000
Germany	300000
India	350000
UK	250000
USA	700000
Total	**2050000**

Country	Sum of Sales	rank
Australia	450000	1
Germany	300000	1
India	350000	1
UK	250000	1
USA	700000	1
Total	**2050000**	**1**

SUM(sales[Sales]) calculates the sum of sales in the current context. In some scenarios, this might always return the same value if the context isn't changing correctly within the RANKX function, leading to all categories getting the same rank.

Using a measure like [total_sales] ensures that the calculation of total sales is done correctly across the desired context. Measures are context-aware and recalculated in each row of the visual, which ensures that the ranking is done based on the correct sales values.

Country	Sum of Sales	rank_all
USA	700000	1
Australia	450000	2
India	350000	3
Germany	300000	4
UK	250000	5
Total	**2050000**	**1**

All removes any active filters from the column. (inside the visual and outside the visual)

Country	Sum of Sales	rank_allselected
USA	700000	1
Australia	450000	2
India	350000	3
Germany	300000	4
UK	250000	5
Total	**2050000**	**1**

AllSelected removes the filter inside the visual but not from outside the visual)

Country

- ☐ Australia
- ☐ Germany
- ☐ India
- ☐ UK
- ☐ USA

3. Rank- Allselected

```
1 rank_allselected = RANKX(ALLSELECTED(sales),[total_sales],,DESC,Dense)
```

Ranking Scenario - Why we need to add All, What will happen if we replace All with Allselected.

Country	Sum of Sales
Australia	450000
Germany	300000
India	350000
UK	250000
Total	**1350000**

Country	Sum of Sales	rank
Australia	450000	1
Germany	300000	1
India	350000	1
UK	250000	1
Total	**1350000**	**1**

SUM(sales[Sales]) calculates the sum of sales in the current context. In some scenarios, this might always return the same value if the context isn't changing correctly within the RANKX function, leading to all categories getting the same rank.

Using a measure like [total_sales] ensures that the calculation of total sales is done correctly across the desired context. Measures are context-aware and recalculated in each row of the visual, which ensures that the ranking is done based on the correct sales values.

Country	Sum of Sales	rank_all
Australia	450000	2
India	350000	3
Germany	300000	4
UK	250000	5
Total	**1350000**	**1**

All removes any active filters from the column. (inside the visual and outside the visual)

Country	Sum of Sales	rank_allselected
Australia	450000	1
India	350000	2
Germany	300000	3
UK	250000	4
Total	**1350000**	**1**

AllSelected removes the filter inside the visual but not from outside the visual)

4. Hasonevalue

```
1 rank_allselected =
2 if(HASONEVALUE(sales[Country]),
3 (RANKX(ALLSELECTED(sales),[total_sales],,DESC,Dense)),BLANK())
```

Ranking Scenario - Why we need to add All, What will happen if we replace All with Allselected.

Country	Sum of Sales
Australia	450000
Germany	300000
India	350000
UK	250000
Total	**1350000**

Country	Sum of Sales	rank
Australia	450000	1
Germany	300000	1
India	350000	1
UK	250000	1
Total	**1350000**	**1**

SUM(sales[Sales]) calculates the sum of sales in the current context. In some scenarios, this might alwa return the same value if the context isn't changing correctly within the RANKX function, leading to all categories getting the same rank.

Using a measure like [total_sales] ensures that the calculation of total sales is done correctly across the desired context. Measures are context-aware and recalculated in each row of the visual, which ensures the ranking is done based on the correct sales values.

Country	Sum of Sales	rank_all
Australia	450000	2
India	350000	3
Germany	300000	4
UK	250000	5
Total	**1350000**	**1**

All removes any active filters from the column. (inside the visual and outside the visual)

Country	Sum of Sales	rank_allselected
Australia	450000	1
India	350000	2
Germany	300000	3
UK	250000	4
Total	**1350000**	

AllSelected removes the filter inside the visual but not from outside the visual)

All and Allexpect
https://www.sqlbi.com/articles/managing-all-functions-in-dax-all-allselected-allnoblankrow-allexcept/

Find % of subcategory sales

```
1  totalsubcatSales =
2  DIVIDE([total sales],
3  CALCULATE([total sales],
4          ALL(Product_DIM[Sub-Category])))
```

Sub-Category	total sales	totalsubcatSales
Phones	17,06,824.14	0.14
Copiers	15,09,436.27	0.12
Chairs	15,01,681.76	0.12
Bookcases	14,66,572.24	0.12
Storage	11,26,812.97	0.09
Appliances	10,10,535.53	0.08
Machines	7,79,060.07	0.06
Tables	7,57,041.92	0.06
Accessories	7,49,237.02	0.06
Binders	4,61,869.39	0.04
Furnishings	3,85,155.97	0.03
Art	3,71,613.15	0.03
Supplies	2,42,811.13	0.02
Total	**1,26,42,501.91**	**1.00**

Q2) -

Identify Individual Month Sales as Percentage of Total Sales

Step 1- create DimDateTable

```
ADDCOLUMNS(
  CALENDAR(MIN(Sales_OrgA[Date]),Max(Sales_OrgA[Date])),
    "Year",Year([Date]),
    "Quarter","Q"&QUARTER([Date]),
    "MonthNumber",MONTH([Date]),
    "MonthName",FORMAT([Date],"MMM"),
    "Week",WEEKNUM([Date]),
    "Day",FORMAT([Date],"DDDD")
)
```

Step 2- create relation btw “DimDateTable” and “Sales table” (Fact table) with date column
Step 3 -

```
1 Perc_Of_TotalSales =
2
3 Var Dr = CALCULATE([Total_Sales],REMOVEFILTERS(Sales_OrgA))
4
5
6 Var res = DIVIDE([Total_Sales],Dr)
7
8 return res
```

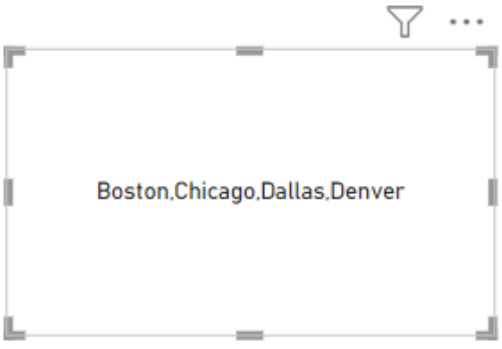
```
1 Remov_Fil =
2
3 CALCULATE([Total_Sales],REMOVEFILTERS(Sales_OrgA))
```

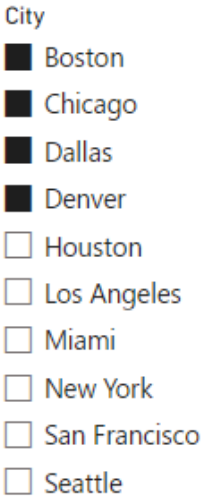
MonthName	Perc_Of_TotalSales	Total_Sales	Remov_Fil
Apr	8.47%	15,500.89	182,975.34
Aug	7.15%	13,087.27	182,975.34
Dec	8.18%	14,962.95	182,975.34
Feb	8.38%	15,336.13	182,975.34
Jan	7.94%	14,525.69	182,975.34
Jul	8.05%	14,728.13	182,975.34
Jun	8.90%	16,293.78	182,975.34
Mar	9.53%	17,440.67	182,975.34
May	6.03%	11,038.05	182,975.34
Nov	7.95%	14,549.80	182,975.34
Oct	10.66%	19,496.89	182,975.34
Sep	8.75%	16,015.08	182,975.34
Total	**100.00%**	**182,975.34**	**182,975.34**

Step 4 - sort month name with month number

Q 3) Concatenate values based on slicer selection

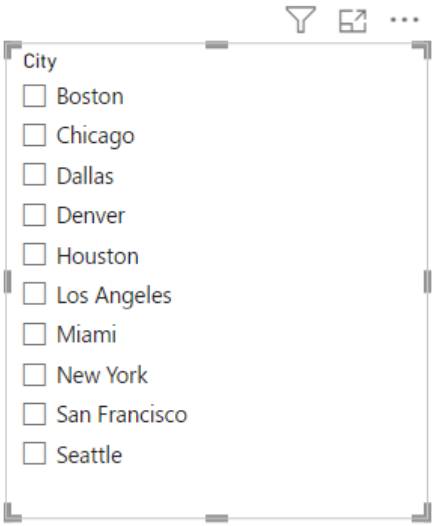
```
1  concatenatex = CONCATENATEX(location_dimension,location_dimension[City],",")
```





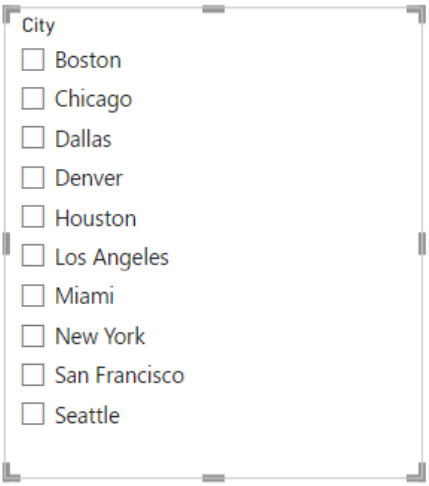
```
concatenatex = CONCATENATEX(location_dimension,location_dimension[City],",")
```

Boston,Chicago,Dallas,Denver,Houston,Los Angeles,Miami,New York,San Francisco,Seattle




```
concatenatex =
var concat = CONCATENATEX(location_dimension,location_dimension[City],",")
RETURN IF(ISFILTERED(location_dimension[City]),concat,"Please select some values")
```

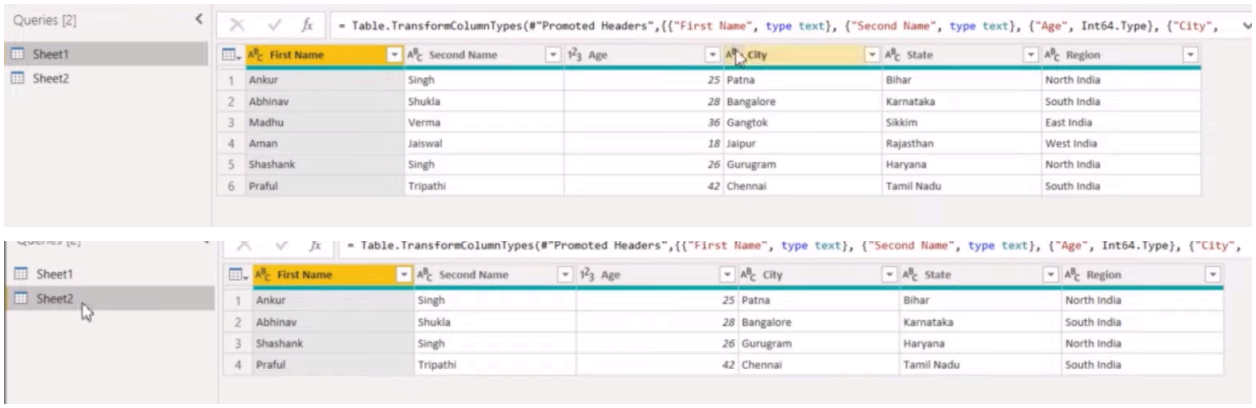
Please select some values



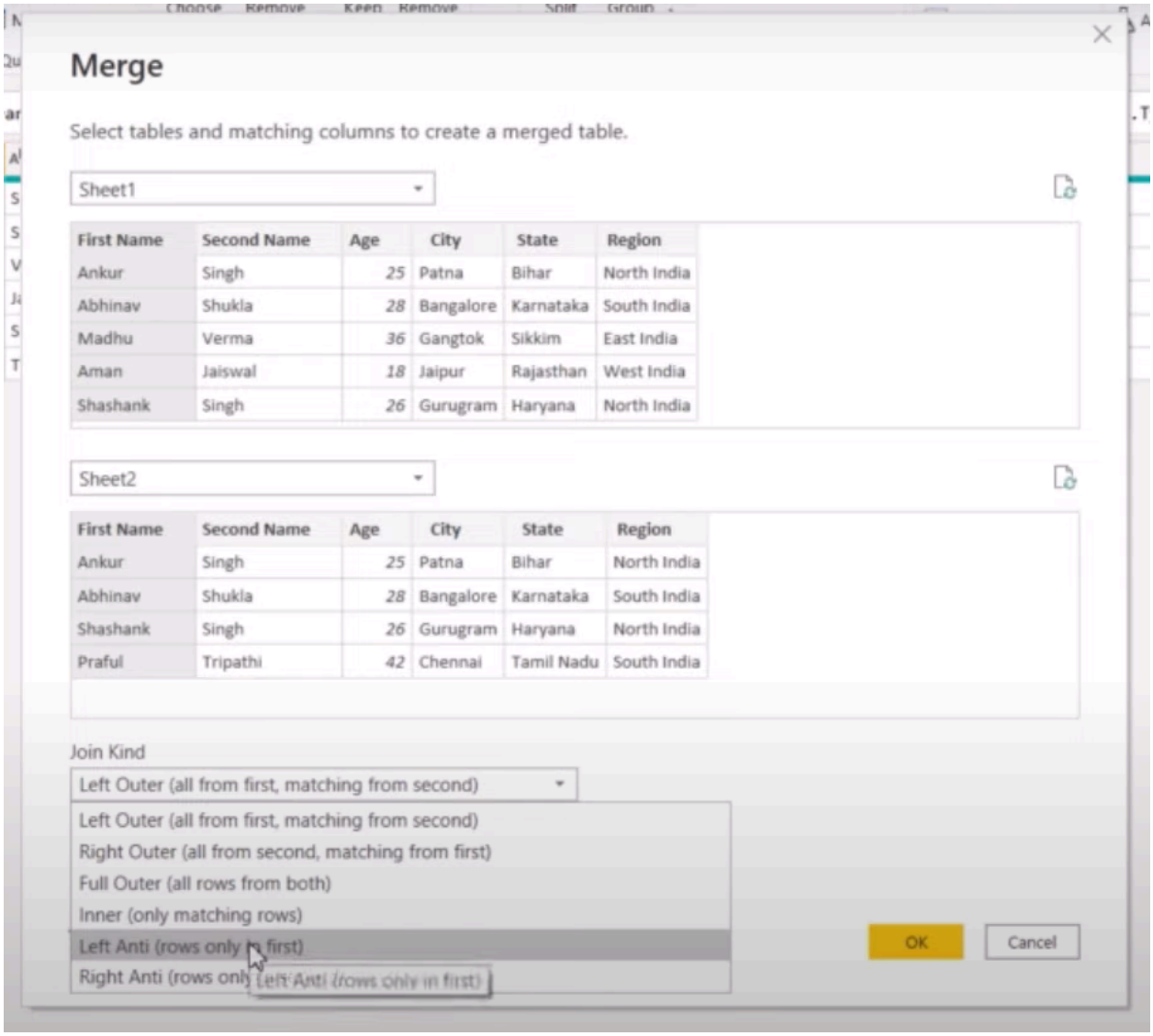
Concatenate:
https://radacad.com/concatenatex-in-power-bi-and-dax-concatenate-values-of-a-column

Related and Relatedtable:
https://www.sqlbi.com/articles/using-related-and-relatedtable-in-dax/

Q 3) you have two tables, you have to find those records in Table 1 which are not present in Table 2.



Power query approach



Dax approach

```
1 Table = EXCEPT(Sheet1, Sheet2)
```

First Name	Second Name	Age	City	State	Region
Madhu	Verma	36	Gangtok	Sikkim	East India
Aman	Jaiswal	18	Jaipur	Rajasthan	West India

Q4) Calculate sales via inactive relationship

```
1 Total Revenue with inactive stats = CALCULATE([Total Revenue],
  USERELATIONSHIP(AW_Calendar[Date],AW_Sales[StockDate]))
```

Q5) calculate() vs Calculatetable()

```
1 Total Profit from Red Color Prodct = CALCULATE([Total profit], AW_Products
  [ProductColor]="Red")
```

```
1 Profit Using CT = SUMX(CALCULATETABLE(AW_Products,AW_Products[ProductColor]="Red"),
  [Total profit])
```

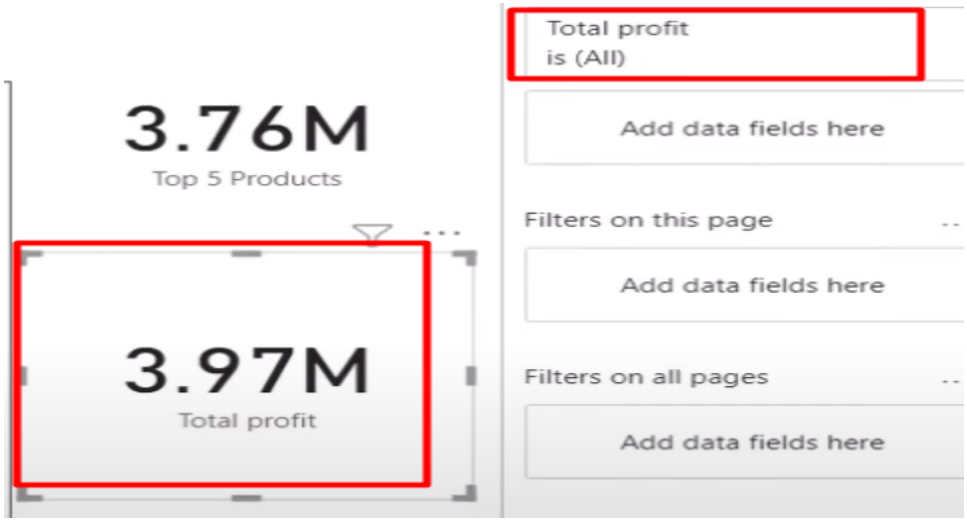
Use CALCULATE when:

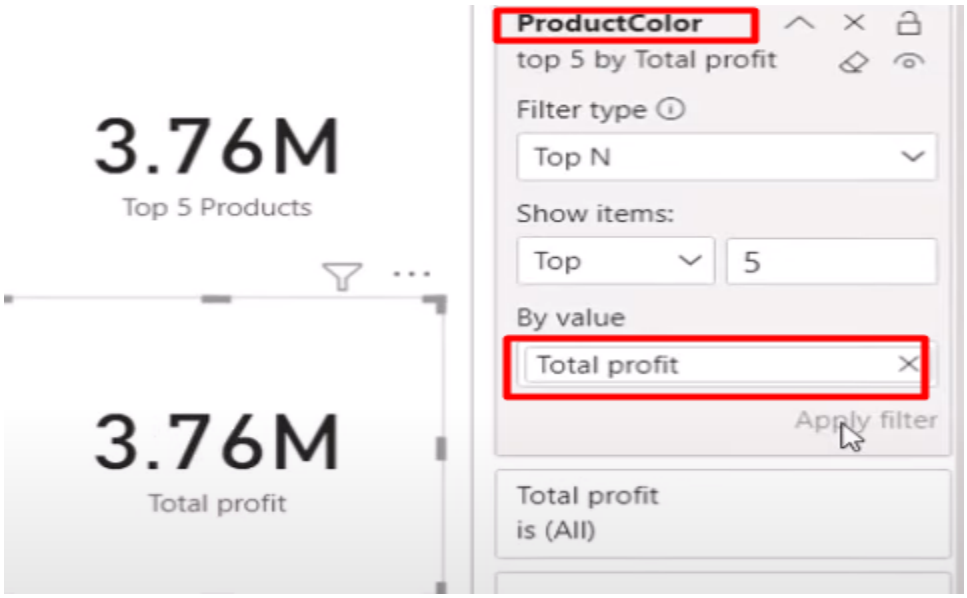
- You need to modify the filter context for specific columns or measures.
- Fine-tuning calculations based on conditions or criteria is required.

Use CALCULATETABLE when:

- You want to modify the filter context for entire tables.
- Filtering and shaping entire tables based on conditions are necessary.

Q5) Calculate top n products






```
1 Top 5 Products = CALCULATE([Total profit],FILTER(VALUES(AW_Products
  [ProductColor]),RANKX(AW_Products,[Total profit],,DESC,Dense) <=5))
```

can we use like this -- calculate([Total_Sales], TopN (5,values (Products [Product]), [Total_Sales], desc)

Dynamic Top 5-10 Filter

```
1 top_ranking =
2 var top_rank = IF(HASONEVALUE(Store_DIM[Region]),RANKX(ALLSELECTED(Store_DIM[Region]),[total sales],,DESC,Dense))
3 return IF(top_rank<=dynamic_ranking[dynamic_ranking Value],[total sales])
```

Region	total sales	rank	top_ranking
Western Europe	17,29,984.40	1	17,29,984.40
Central America	12,28,321.11	2	12,28,321.11
Oceania	11,01,009.71	3	11,01,009.71
Southeastern Asia	8,82,082.82	4	8,82,082.82
Southern Asia	8,69,165.10	5	8,69,165.10
Eastern Asia	8,55,059.39	6	8,55,059.39
Western US	7,49,559.98	7	7,49,559.98
Eastern US	7,07,915.93	8	7,07,915.93
Northern Europe	6,51,626.43	9	6,51,626.43
South America	6,28,335.66	10	6,28,335.66
Southern Europe	6,09,940.11	11	
Central US	4,85,135.30	12	
Total	**1,26,42,501.91**		**1,26,42,501.91**

dynamic_ranking

10

Region	top_ranking
Central America	12,28,321.11
Eastern Asia	8,55,059.39
Eastern US	7,07,915.93
Northern Europe	6,51,626.43
Oceania	11,01,009.71
South America	6,28,335.66
Southeastern Asia	8,82,082.82
Southern Asia	8,69,165.10
Western Europe	17,29,984.40
Western US	7,49,559.98
Total	**1,26,42,501.91**

Parameters

Add parameters to visuals and DAX expressions so people can use slicers to adjust the inputs and see different outcomes. Learn more

What will your variable adjust?

Numeric range

Name

dynamic_ranking

Data type

Whole number

Minimum

1

Maximum

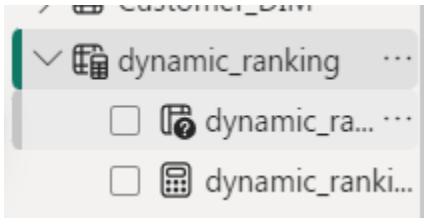
20

Increment

1

Default

```
Ranking =
VAR _topCategory = RANKX( ALL(Categories), [Total Sales], , DESC)
VAR _bottomCategory = RANKX( ALL(Categories), [Total Sales], , ASC)

VAR _topProducts = RANKX( ALL(Products), [Total Sales], , DESC)
VAR _bottomProducts = RANKX( ALL(Products), [Total Sales], , ASC)

VAR _ranking = IF(
    CONTAINSSTRING(SELECTEDVALUE(Breakdown[Breakdown Fields]), "Category")
    , IF(SELECTEDVALUE(TopBottom[Value]) = "Top",_topCategory, _bottomCategory)
    , IF(SELECTEDVALUE(TopBottom[Value]) = "Top",_topProducts, _bottomProducts)
)
RETURN
IF( ranking <= 'Ranking Option'[Ranking Option Value], [Total Sales])
```

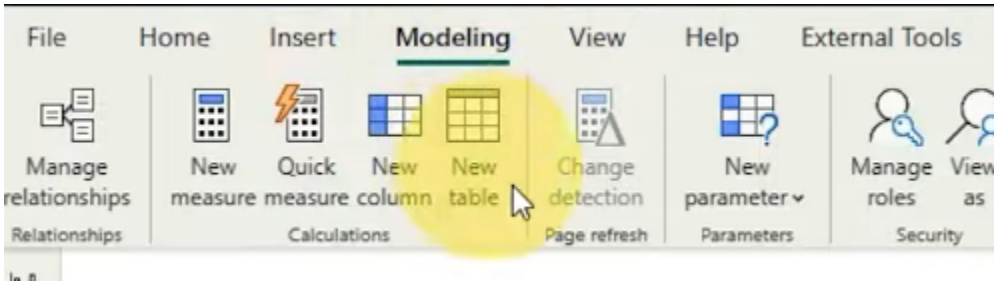



```
dynamic_ranking = GENERATESERIES(1, 20, 1)
```

```
dynamic_ranking Value = SELECTEDVALUE('dynamic_ranking'[dynamic_ranking])
```

 # dynamic Top and Bottom

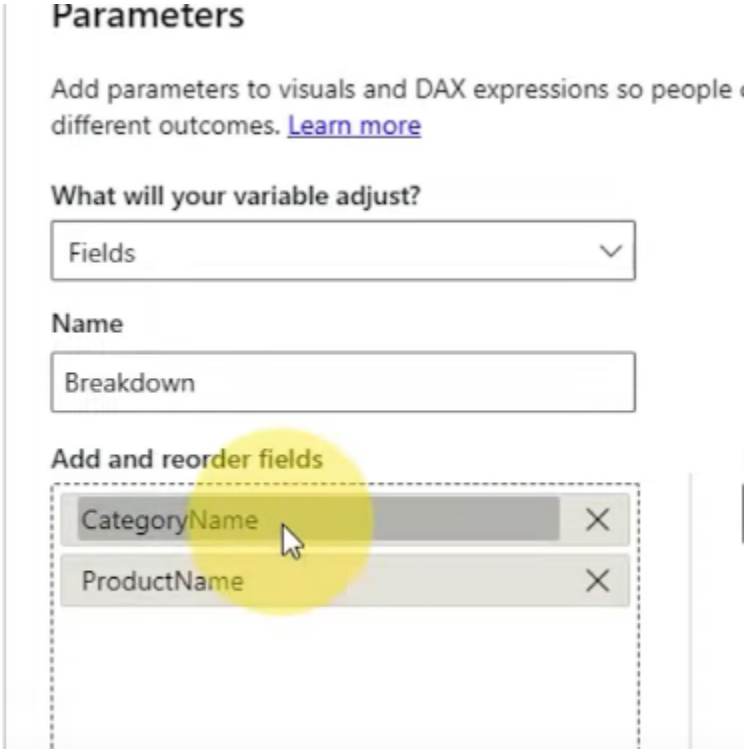
Create table

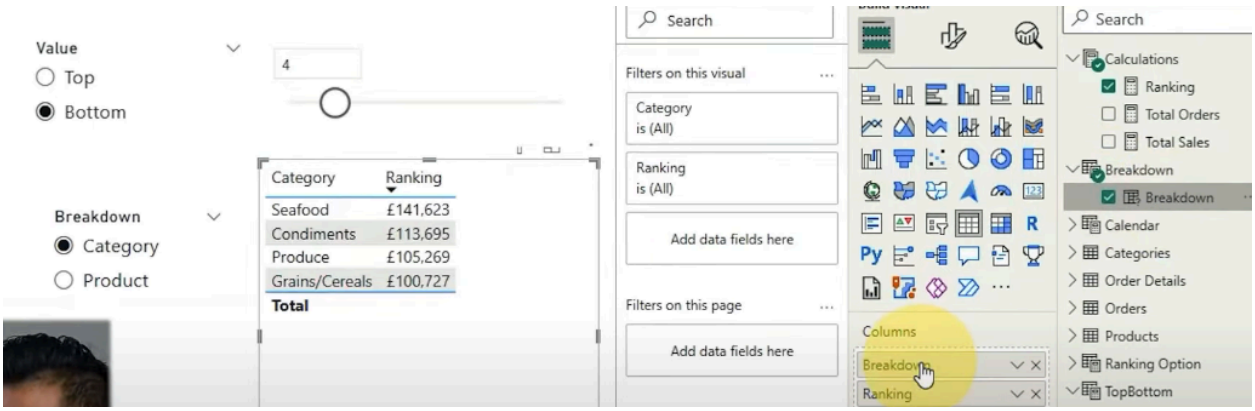



```
TopBottom = { "Top", "Bottom" }
```

```
Ranking =
VAR _topCategory = RANKX( ALL(Categories), [Total Sales], , DESC)
VAR _bottomCategory = RANKX( ALL(Categories), [Total Sales], , ASC)

VAR _ranking = IF(SELECTEDVALUE(TopBottom[Value]) = "Top"
    ,_topCategory
    , _bottomCategory
)
RETURN
IF(_ranking <= 'Ranking Option'[Ranking Option Value], [Total Sales])
```

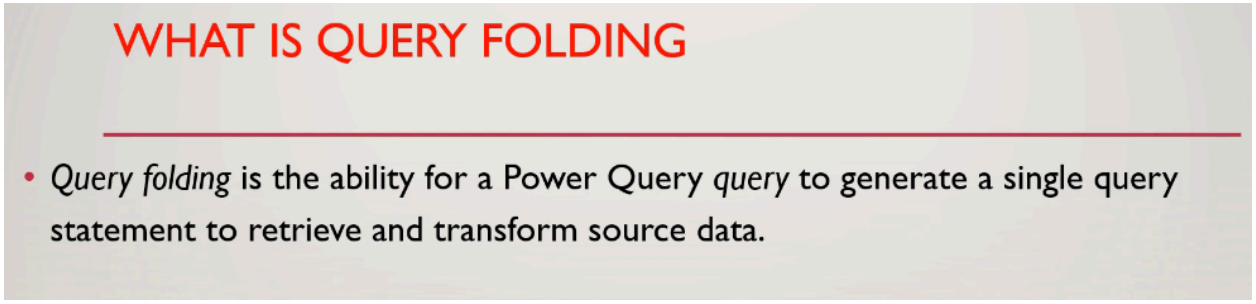
 # dynamic Category change

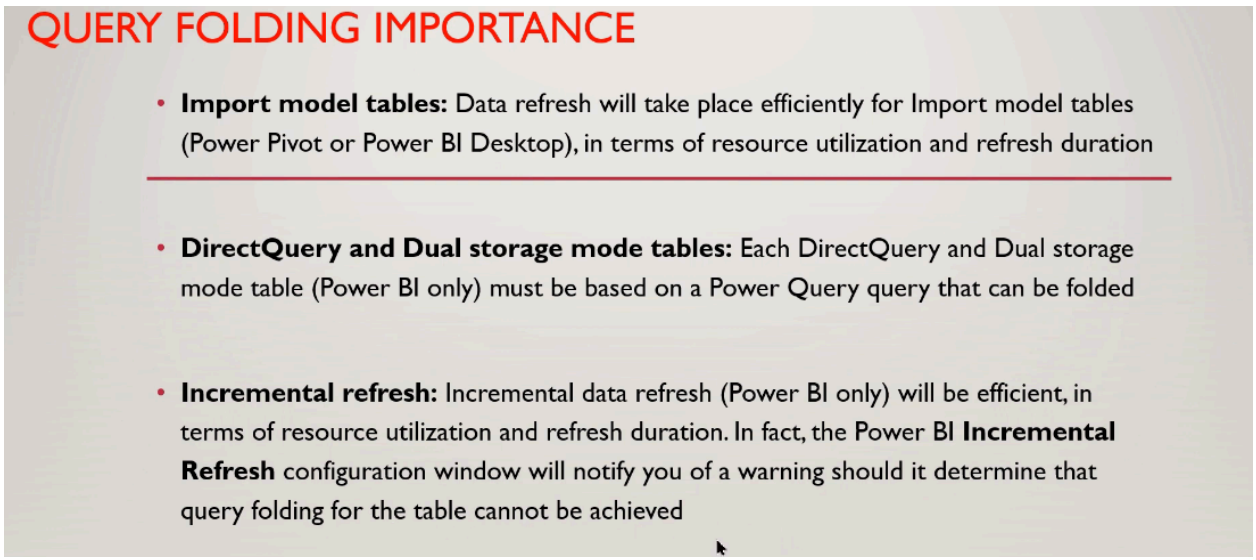




```
1  Ranking =
2  VAR _topCategory = RANKX( ALL(Categories), [Total Sales], , DESC)
3  VAR _bottomCategory = RANKX( ALL(Categories), [Total Sales], , ASC)
4
5  VAR _topProducts = RANKX( ALL(Products), [Total Sales], , DESC)
6  VAR _bottomProducts = RANKX( ALL(Products), [Total Sales], , ASC)
7
8  VAR _ranking = IF(
9      CONTAINSSTRING(SELECTEDVALUE(Breakdown[Breakdown Fields]), "Category")
10     , IF(SELECTEDVALUE(TopBottom[Value]) = "Top",_topCategory, _bottomCategory)
11     , IF(SELECTEDVALUE(TopBottom[Value]) = "Top",_topProducts, _bottomProducts)
12 )
13 RETURN
14 IF( ranking <= 'Ranking Option'[Ranking Option Value], [Total Sales])
```

 # What is Query Folding?

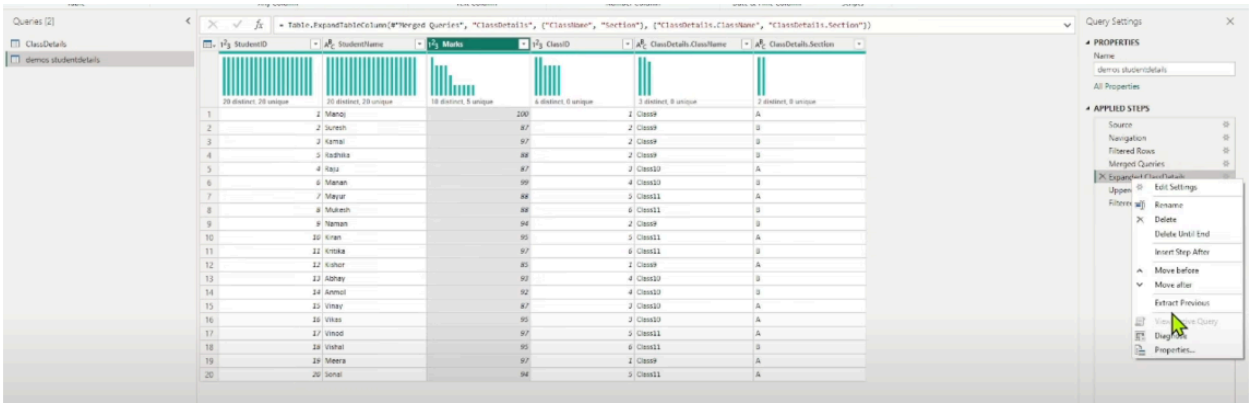




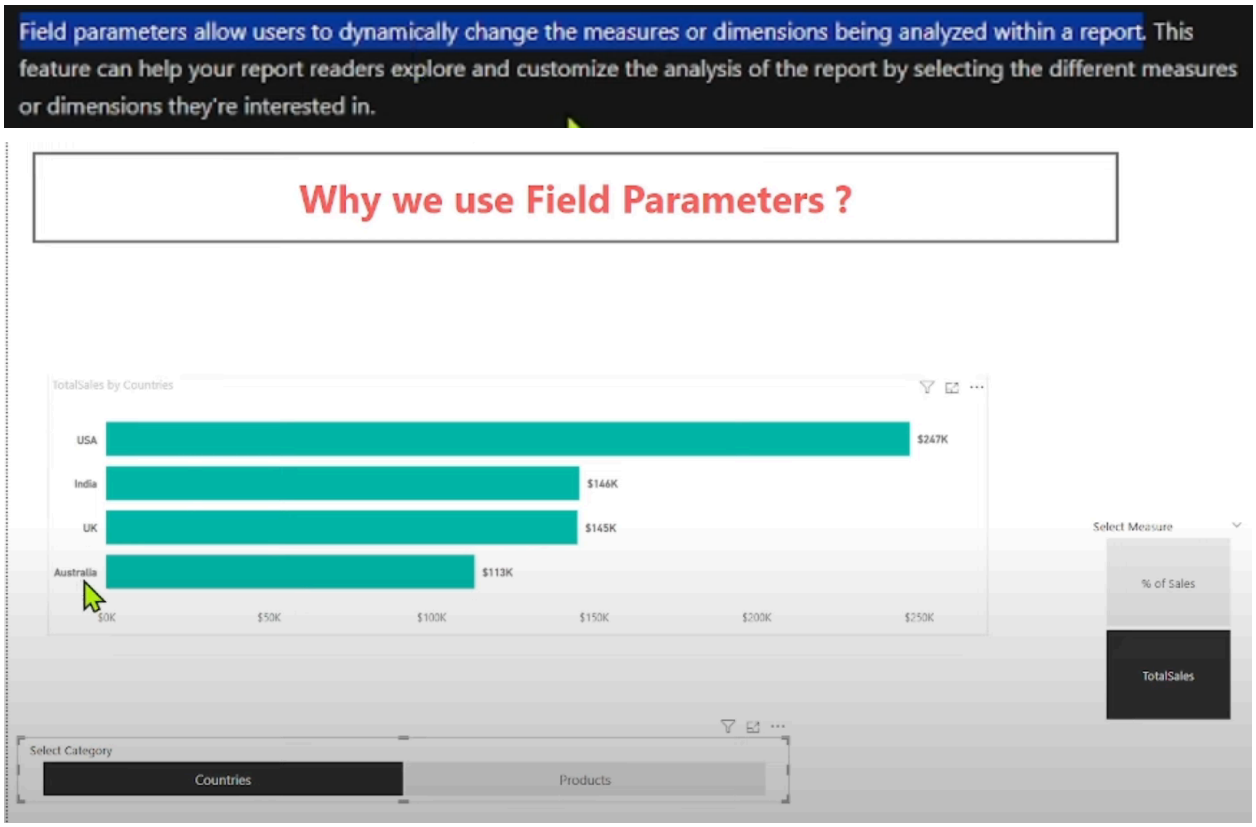
Only available for SQL related table

https://www.youtube.com/watch?v=Akz42d8hgCE

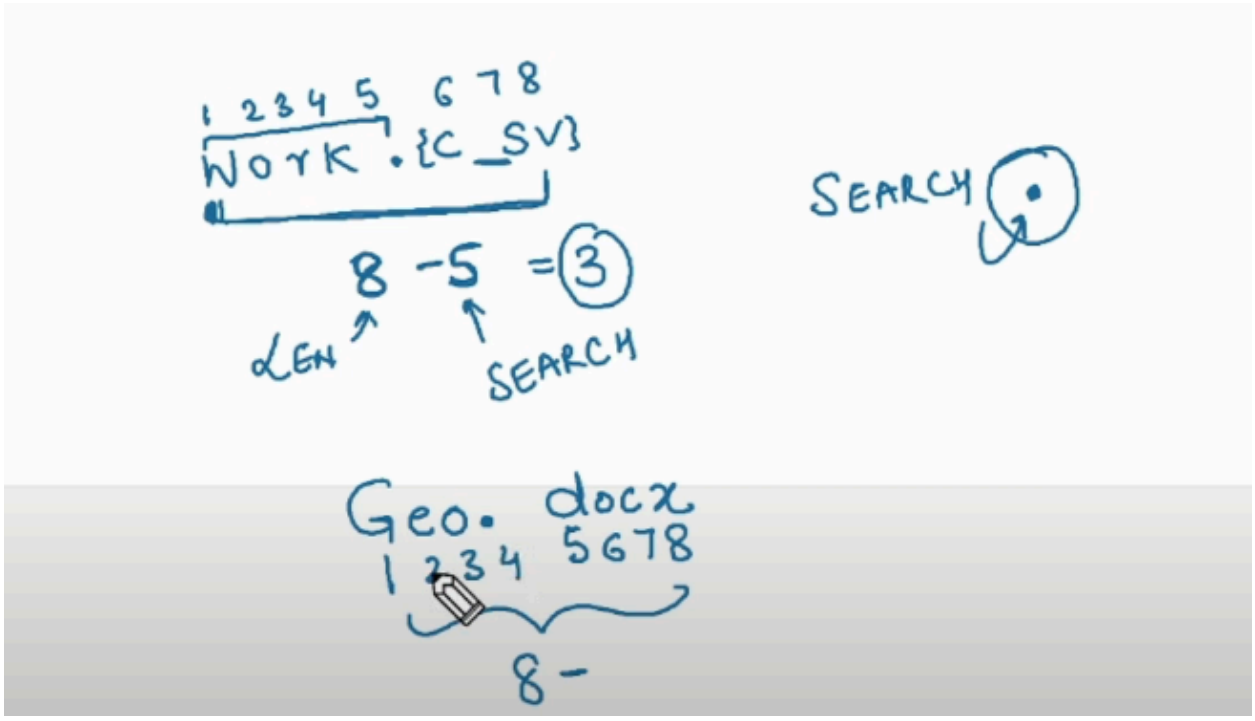
https://www.youtube.com/watch?v=xeBx8NDFXDk&list=PLFbjn94wCCeWIFO2w6LdeHmf7kEROoYkLE&index=21



What are Field Parameters?



 # Search()


1 2 3 4 5 6 7 8
Work.{C_SV}
8 -5 =3
LEN
SEARCH
SEARCH
Geo. docx
1 2 3 4 5 6 7 8
8-


1 Extensions = RIGHT(FileDetails[FileNames],Len(FileDetails[FileNames]) - SEARCH(".",FileDetails[FileNames]))
FileNames
Extensions
Work.csv
csv
Test.py
py
Employee.txt
txt
Department.sql
sql
Geo.docx
docx

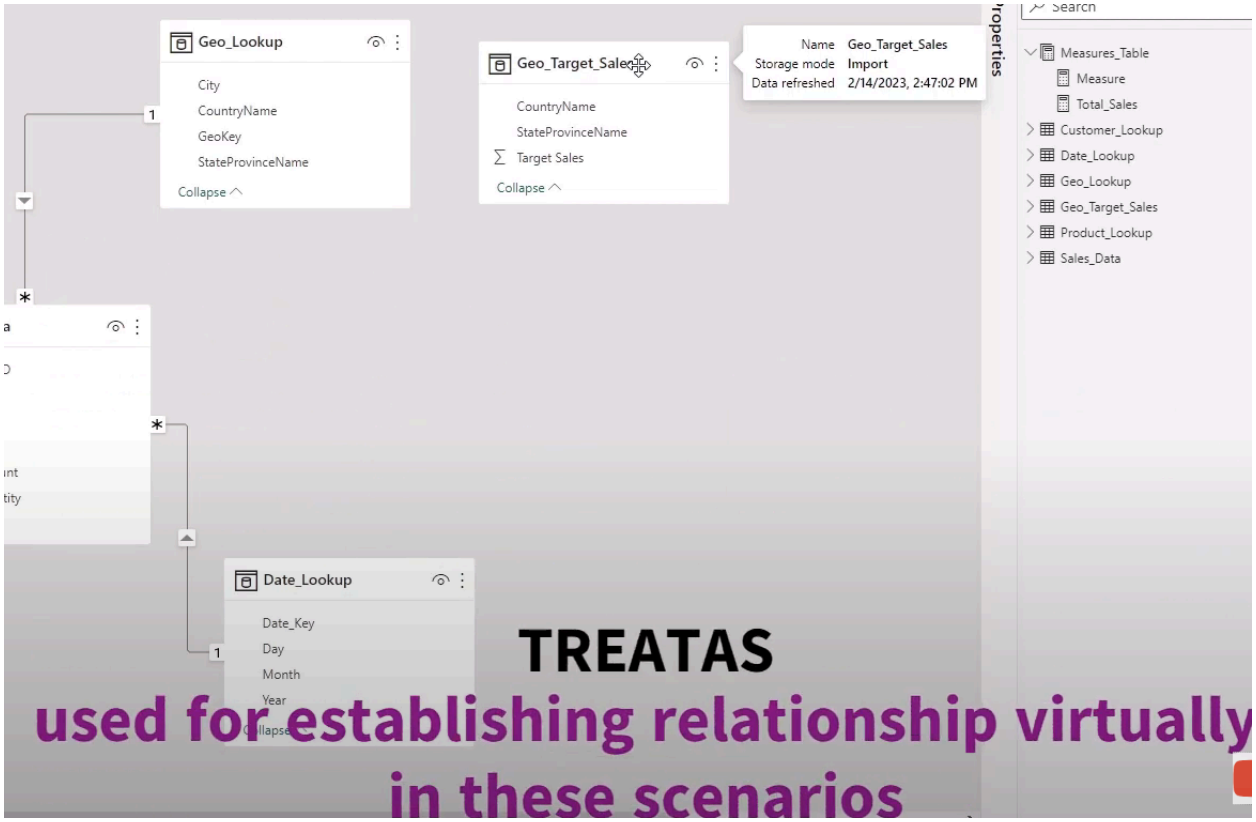
Rankx (Top 2)

```
Top 2 Customers (RANKX) =
/*
Identify Top 2 Customer for each product so as to reward them with good discount in future

*/

Var rankcust =
IF(HASONEVALUE(Customer_Lookup[Customer_Name]),
RANKX(ALL(Customer_Lookup),[Total_Sales_Amount],,DESC,Dense))

Var top2 = IF(rankcust < 3 ,[Total_Sales_Amount])

Return top2

```

Nathan	2750000	
Simon	3630000	
Tim	3930000	
Tina	3240000	
Victor	3840000	
Jennifer	4500000	4500000
Jasmine	4860000	4860000
Total	**56520000**	**56520000**

Countries that are not able to beat the Sales Target

```


//Interview Question : Identify Countries which are not able to beat the Sales Target









/*
//Concept of TREATAS

TREATAS : Transferring filter context from one table to another virtually

ColumnNames : Return a list of rows that are also present in the columns specified in Table Expression

**Key Condition : Count of columns must equal to the count of columns specified in Table Expression with same order as well

Good Practice : Always try to create physical relationship wherever possible
*/
```

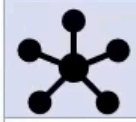



```
19
20 CALCULATE(SUM(Geo_Target_Sales[Target Sales]),
21 TREATAS(SUMMARIZE(Geo_Lookup,Geo_Lookup[StateProvinceName],Geo_Lookup[CountryName]),
22 Geo_Target_Sales[StateProvinceName],Geo_Target_Sales[CountryName]))
23
24
```

CountryName	Sum of Sales_Amount	TargetSales (TREATAS_Demo)	CountriesComparision
Australia	$54,910,000	$51,521,550	106.58%
Germany	$51,115,000	$43,690,514	116.99%
United Kingdom	$32,220,000	$32,417,112	99.39%
United States	$206,635,000	$148,441,743	139.20%
Total	**$344,880,000**	**$276,070,919**	**124.92%**

 # How to tell your Project Experience

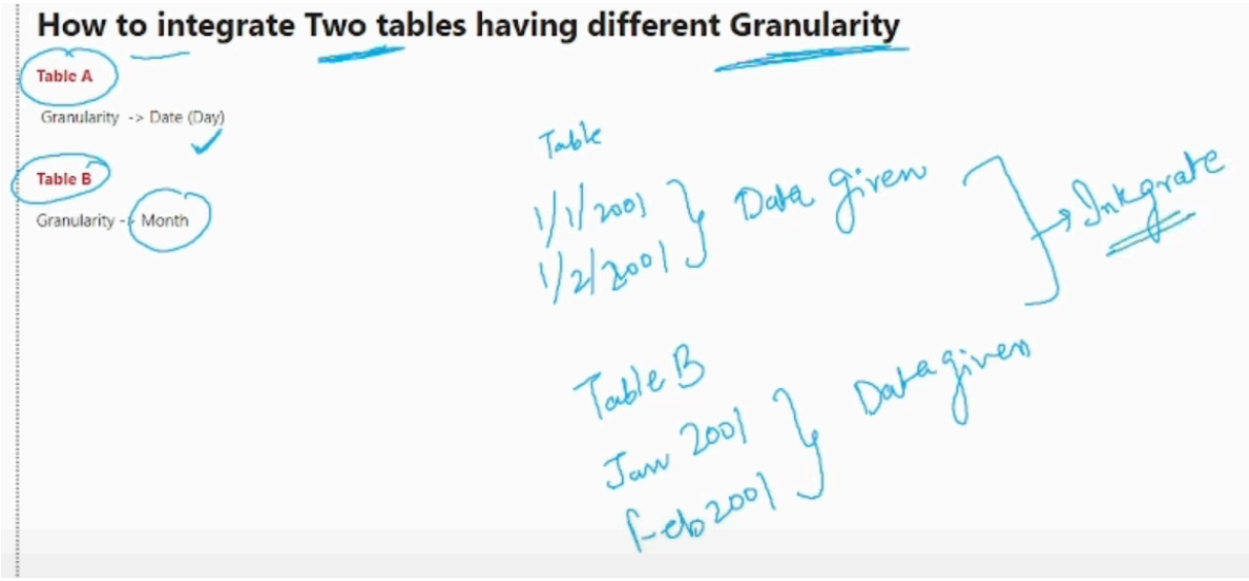
Sections	Key Words	Details
Experience	3 Years ,5 Years , 7 Years	Total Experience and Power BI Experience
Industries	Healthcare	To Analyze Patient Data to monitor health Medical Equipment and Medical Staff Data for effective Utilization Hospital Finances or Health Schemes related Data to identify overall revenue, Profit etc.. Claims Related Data
Industries	Finance	Analyze Balance Sheets Income Statements Cash Flows
Industries	Retail	Analysis of Customer Data to understand overall purchase behavior Identify Sales Growth Inventory Management for efficient supply chain
Roles and Responsibilities	Developer	Data Integration Activities , Data Modeling , Working on design Power BI Reports and Dashboards , Testing
Roles and Responsibilities	Technical Lead	Working on Architecture Design Providing Technical Guidance Approving Code Changes Code Reviews

	Challenges Faced	Solution
	**Data Quality Issues**	1) Design plan for Data Cleansing Activities (*Identification of Issues and then address those issues*) 2) Data Validation **Data should be Consistent and Accurate
	**Data Modeling Issues**	Design Relationship among different entities when dealing with large datasets.
	**Performance Issues**	Loading time of visuals in reports was high hence used different optimization techniques like Query Folding..
	**Data Security**	Setting Row Level Security - User has Access to relevant data

	Client Feedback/Achievement	Details
	**Productivity and Efficiency**	1) Decrease Processing time by 20% or increase sales growth by 5% 2) Can Explain saving in $terms as well
	**Responsive Design**	Appreciated for responsive design of report
	**Feature Rich Report**	Appreciated for implementing drill down, filters etc. into the reports which made them more interactive Added tooltips to enhance user experience

	Lessons Learned	Details
	**Testing Before Deployment**	Need to test each and every aspect of report before any production deployment
	**Skills**	Technical skills, Problem Solving Skills , Communication Skills

 # How to integrate two tables having different granularity



SUMMARIZE Function in DAX:

(Table Manipulation function -> Manipulate and Return Tables)

Usage -> To Aggregate and Summarize Data

Helpful in creating summary tables from Large Dataset which ultimately helps in exploring different perspective

Summarized tables are helpful in visualizing Trends , Patterns etc..

```
1 MONTH NAME = FORMAT(Sales_North[Date],"MMMM")
```

Date	SalesAmount($)	Year	Month Number	MONTH NAME
1/1/2021	41276	2021	1	January
1/2/2021	52956	2021	1	January
1/3/2021	26467	2021	1	January
1/4/2021	32952	2021	1	January
1/5/2021	42836	2021	1	January
1/6/2021	59610	2021	1	January

```
1 Sales_North_Summ =
2 SUMMARIZE(
3     Sales_North,
4     Sales_North[Year],
5     Sales_North[Month Number],
6     "Month Name",MIN(Sales_North[Month Name]),
7     "Total Sales ($)",SUM(Sales_North[SalesAmount($)]))
```

Table

→ Group by

Pending

Year	Month Number	Month Name	Total Sales ($)
2021	1	January	1096849
2021	2	February	1155954
2021	3	March	1182560
2021	4	April	1187762
2021	5	May	1107422
2021	6	June	1187837

```
1 Sales_Integrated =
2 UNION(Sales_West,
3 SUMMARIZE(
4     Sales_North_Summ,
5     Sales_North_Summ[Year],
6     Sales_North_Summ[Month Number],
7     Sales_North_Summ[Month Name],
8     Sales_North_Summ[Total Sales ($)]
9     ))
```

Sales →

Sales_West

Order of columns

Count of columns

Year	Month Number	Month Name	Total Sales($)
2021	1	January	406496
2021	2	February	559199
2021	3	March	245312
2021	4	April	201645
2021	5	May	501036
2021	6	June	299489
2021	1	January	1096849
2021	2	February	1155954
2021	3	March	1182560
2021	4	April	1187762
2021	5	May	1107422
2021	6	June	1187837