

<Process Title>

<Apex/Group/Operational> Layer

Document Version / Details: Ver. 2.0/ 14-Nov-2022

WORD DOCUMENT

DAX Expression Power Bi

1. Now () dax expression in pbi desktop will give the timing of your computer’s timing

But the same DAX expression when published into pbi service, in pbi service it will show timing as the UTC timing bcoz in pbi service the timing is set to UTC time.

Link : <https://youtu.be/2kmFfbOeFJg>

1. If you write date as dt”2022-09-09” then the data type of this will be considered as date and not text
2. Any parameter inside DAX expression written inside square brackets is considered as optional parameter.
3. The result of the UTCNOW and NOW function changes only when the formula is refreshed. It is not continuously updated.
4. Usually the return type of most of the TIME related DAX functions is datetime
5. EVALUATE TableName means it is going to get me all the that is there in that Table.
6. ALL DAX Expression :-
7. Always use ALL dax expression inside CALCULATE dax expression only
8. If I just want to clear/remove the filters from every table then i can just use ALL( ) and inside All no Table name or Column name : So this will clear filter on all tables.

Eg :

1 All\_Dax12 = CALCULATE([Sum\_Total\_Calls],ALL())

1. ALL dax expression can be applied to whole table or even to a single column
2. For eg : if I want to find total calls based on bucket :

I have created one measure which will give me total calls : 1 Sum\_Total\_Calls = SUM(Sheet1[Total\_calls])

See in the table the value of Sum\_Total\_Calls keeps on changing based on the slicer value.

A screenshot of a computer

Description automatically generated

1. All\_Total\_Calls = CALCULATE([Sum\_Total\_Calls],ALL(Sheet1)) – Check the All\_Total\_Calls column in the table the value remains the same even though the value is selected from the slicer
2. If there are multiple slicers and the columns used in all those slicer are from different tables then you can apply ALL dax function to multiple table at once.
3. All\_Total\_Calls = CALCULATE([Sum\_Total\_Calls],ALL(Sheet1),ALL(Sheet2)) = ALL dax function applied to multiple table.
4. Make it a point that ALL function always comes in the FILTER or the TABLE parameter of DAX expression, it never comes in the EXPRESSION parameter of dax expression.
5. Difference between ALL and ALLcrossfiltered is :

Return type of All is a table or a Column whereas return type of Allcrossfiltered is None. Allcrossfiltered only clears the filters on the table it returns nothing .

Eg:-

1Matrix\_Measure= DIVIDE(SUM(Sheet1[Total\_calls]),CALCULATE(SUM(Sheet1[Total\_calls]),ALL(Sheet1)))

So here in this example ALL works same as allcrossfiltered it is not returning anything it is just clearing all the filters in that table.

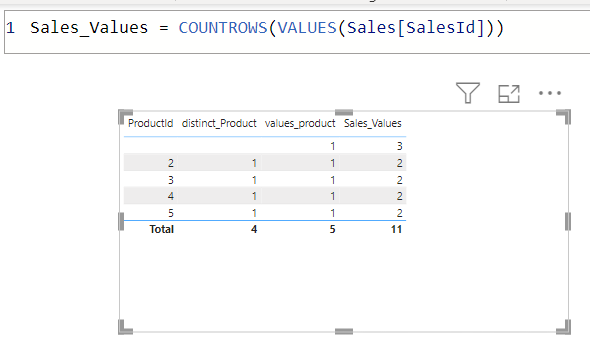
1. In ALLEXCEPT() dax expression : the columns that are specified after the table name must be from the same table that is mentioned as first argument.
2. Blank Row in DAX - Blank Row appears when there is invalid relationship between two tables. Invalid relationship means that the value is present in child table but it is absent in the parent table. The difference between distinct and values DAX expression is that the DISTINCT dax expression does not consider the blank rows in parent table but distinct DAX expression considers blank rows in child table. whereas the values dax expression considers the blank rows in parent table as well as child table.

A screenshot of a spreadsheet

Description automatically generated

Consider LHS as a Product (parent) table and RHS as Sales (child) table :

1. If the value is present in the child table but not present in the parent table then it will be considered as a blank row while using VALUES dax expression



1. If the value is present in parent table but not present in child table then If we add columns which has done calculations on the child table , it will only give the values which are present(common) in both the tables and ignore all other values of parent table. (Ideal Case Scenario)
2. ALLNOBLANKROW does not count blank rows only when the first column and the calculated column both are from parent table. If there are more than 1 blank rows from the child table then also it will be counted as 1 only. Therefore the difference between the value of ALL and ALLNOBLANKROW is only 1.

A screenshot of a spreadsheet

Description automatically generated

A screenshot of a computer

Description automatically generated

Productid col is from parent table

1 allnoblankrow = COUNTROWS(ALLNOBLANKROW('product'[ProductId]))

1. All\_DAX = COUNTROWS(ALL('product'[ProductId]))
2. Difference between ALL and RemoveFilter DAX expression is that RemoveFilters cannot be used as TableExpression it cannot return rows whereas ALL can be used as a table expression and it can return rows. (REMOVEFILTERS just removes the filters from the dax expression )
3. CALCULATETABLE returns a table based on filter conditions. It returns all columns of the base table, but if you only want to see certain columns, you can use the SUMMARIZE Function within CALCULATETABLE.
4. If you are just using CALCULATETABLE or CalculateTable dax with summarize dax expression then it needs to be written in a (Click on Modelling tab and Click on Table)

Because its return type is table

A screenshot of a computer

Description automatically generated

1. Calculate Table used with SUMX – 1 1\_Cal= SUMX(CALCULATETABLE(Sheet1,AND(Sheet1[CC Name]="Genpact",Sheet1[Zone]="North")),Sheet1[Total\_calls])

This is a measure since we have written CalculateTable inside SUMX expression

Note : Wherever there is a Table parameter in DAX Expression there we can write CalculateTable DAX Expression as well as Filter DAX Expression.

That is why in sumx first parameter I have used CalculateTable DAX Expression

Eg of Filter used as first parameter of sumx dax expression :

SUMX(FILTER('InternetSales\_USD', RELATED('SalesTerritory'[SalesTerritoryCountry])<>"United States")

,'InternetSales\_USD'[SalesAmount\_USD])

1. DATEADDDAX = Returns a table that contains a column of dates, shifted either forward or backward in time by the specified number of intervals from the dates in the current context.

1 DateAddDAX = DATEADD(ClosingBalanceData[Date],-1,YEAR)

The following formula calculates dates that are one year before the dates in the current context (ClosingBalanceData Dates Column).

Here in my ClosingBalanceData[Date] column I have dates of 2023 and 2024 year,

So if I write this dax expression I should get the dates with 2022 and 2023 year

But Since my ClosingBalanceDate[Date] Column involves dates only of 2023 year I will get the values of 2023 as op but for 2022 I will be getting blank values, since those dates are not there in the ClosingBalanceDate Date Column.

A screenshot of a computer

Description automatically generated

Getting blank values for 2023 year

A screenshot of a computer

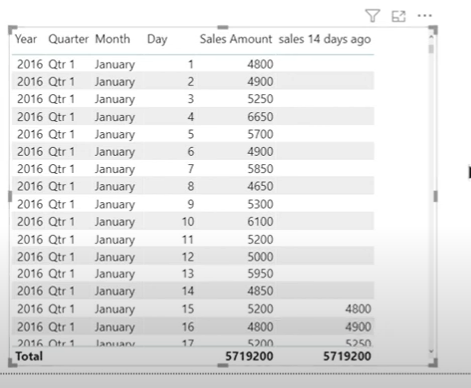
Description automatically generated

Getting values for 2024 year

Note: The result table includes only dates that exist in the **dates** column.

Hint : DateADD dax is basically used for scenarios where suppose if I want to calculate Sales Amount of 14 days ago, sales amount of 1 year ago.





1. DATESBETWEEN dax expression can also be used in the filter parameter of CALCULATE function. For DATESBETWEEN the first parameter is the date column that you need to pass and then the second parameter is STARTDATE and 3rd parameter is ENDDATE .

But the start date and end date both values should be present in the dates column that u passed in first parameter. Otherwise you will get the startdate as which is the first date of dates col and enddate as the last date of dates col.

Note : For whichever Dates DAX Expressions if you pass the DATE Col as a parameter, then whichever expression you are writing that date value must be present in Date Column

1. DatesInPeriod DAX Expression - If you're working with standard date intervals such as days, months, quarters, or years, it's recommended you use the better suited [DATESINPERIOD](https://learn.microsoft.com/en-us/dax/datesinperiod-function-dax) function.
2. DatesinPeriodDAX = DATESINPERIOD(ClosingBalanceData[Date],"09/01/2024",-1,MONTH)

So Here I have made a new table by writing this expression, So I will be getting the values only if BOTH “09/01/2024” and the month prior to that date is available in ClosingBalanceData[Date] Col.

Note : Difference between DateAdd and DatesInPeriod dax expression is that :

DateAdd gives us a single point of Date: eg:-

1 Date\_add = CALCULATE(SUM(ClosingBalanceData[ClosingBalance]),DATEADD(ClosingBalanceData[Date],-3,DAY))

So basically this dax expression will give me what was the sales Amt 3 days ago

If 19th July is today’s date it will give me what was the sales amt on 16th July

DatesInPeriod takes all the dates which fall in that Period, This dax expression is used for calculating Cumulative/Moving Average of Sales

1 Dates\_in\_Period = CALCULATE(SUM(ClosingBalanceData[ClosingBalance]),DATESINPERIOD(ClosingBalanceData[Date],

LASTDATE(ClosingBalanceData[Date]),-3,DAY))

So now this dax expression will give me **Summation** of Sales Amt of 3 days

If 19th July is today’s date then will it give me Summation of SalesAmt of 17th,18th,19th July ,

If 20th July is today’s date then will it give me Summation of SalesAmt of 18th,19th,20th July

1. Difference between DatesYTD and TotalYTD is that we can use TotalYTD single-handedly but DatesYTD should be used inside CALCULATE dax expression. 2nd difference is that we can only apply one/single filter on TOTALYTD whereas we can apply multiple filters on DATESYTD
2. FIRSTNONBLANK and LASTNONBLANK returns the value of the column where expression is not blank,

FIRSTNONBLANKVALUE and LASTNONBLANKVALUE returns the expression value that is non blank .

1. Difference between OPENINGBALANCEMONTH and STARTOFMONTH(Same for CLOSINGBALANCEMONTH and ENDOFMONTH) – OpeningBalanceMonth can used independently since expression parameter is inside OPENINGBALANCEMONTH dax only, whereas STARTOFMONTH dax expression needs to be used inside CALCULATE dax expression.

A screenshot of a computer

Description automatically generated

But Since Opening Balance is Previous Month’s Closing Balance so for the LHS Table (Opening Month DAX) we got the Date from feb Month. All the dates from the Jan Month are missing because the dates in ClosingBalance table start from Jan 1 and we don’t have Closing Balance for December month so all the dates of jan month are missing in the lhs table shown in screenshot.

For ClosingBalance Month we will not face this issue as Openingbalance month. For ClosingBalanceMonth we will also be getting January month’s closing balance.

1. ParallelPeriod Dax – This function takes the current set of dates in the column specified by **dates**, shifts the first date and the last date the specified number of intervals, and then returns all contiguous dates between the two shifted dates. If the interval is a partial range of month, quarter, or year then any partial months in the result are also filled out to complete the entire interval.

The PARALLELPERIOD function is similar to the DATEADD function except that PARALLELPERIOD always returns full periods at the given granularity level instead of the partial periods that DATEADD returns. For example, if you have a selection of dates that starts at June 10 and finishes at June 21 of the same year, and you want to shift that selection forward by one month then the PARALLELPERIOD function will return all dates from the next month (July 1 to July 31); however, if DATEADD is used instead, then the result will include only dates from July 10 to July 21.

A screenshot of a computer

Description automatically generated

Here in the slicer I have selected date as 14th March, so For DateAdd table I will get Sum(Closing Balance) only of March1 – March 14 (Date selected in slicer) as the lastValue(last row) but for parallelPeriod if I have all the entries of March month then it is going to give me Sum(ClosingBalance) till March 31st so that is the difference.

(Check the last row of both the tables there is difference in values). One More difference between them is that DateAdd dax also includes **DATE** Interval whereas ParallelPeriod dax only has Month, Quarter, Year interval.

1. SAMEPERIODLASTYEAR – It is a powerful DAX expression. So in this dax expression Same**Period**Lastyear (Period means year, quarter, month, day) So it helps in comparing current month, current quarter , current year, current day with last year’s same period(day,month, quarter,year)
2. Multi Column Relationships between 2 tables : If you want to establish a relationship between Table1 col1 and Table2 col1 and also if you want to establish a relationship between Table1 col2 and table2 col2 ,Power BI has certain limitations to that As you are establishing **More than one** relationships between 2 tables.

So a trick for that is: Concatenate table 1 col1, col2 and Concatenate table 2 col1, col2 and then make a relationship between these 2 new columns.

Video for ref - <https://www.youtube.com/watch?v=sVACTQh5bOg&t=4s>

Difference between CombineValues and Concatenate DAX is that in In CombineValues dax between each text string there is a separator/delimiter whereas in Concatenate dax function there is no delimiter parameter.

Combine Values dax has delimiter parameter, Concatenate dax does not have delimiter parameter.

Since CONCATENATE dax expression only takes 2 text strings as input, if there are more than 2 strings then we can join them using & (ampersand) operator.

1 Without\_Concatenate\_DAX = "shikha"& "-" & "shah" & "/"

Op – shikha-shah/

1. FIND – Difference between FIND and SEARCH dax is that FIND is case-sensitive, SEARCH is case-insensitive
2. Short Date Format – mm/dd/yyyy eg- 07/25/2023

Long Date Format – dddd mm/d/yyyy eg – Tuesday 07/25/2023

w - Display the day of the week as a number (1 for Sunday through 7 for Saturday)

ww - Display the week of the year as a number (1-54).

1. LEFT DAX – Syntax - LEFT(<text>, <num\_chars>)

If the **num\_chars** argument is a number that is larger than the number of characters available, the function returns the maximum characters available and does not raise an error.

For eg – left(“shikha”,8)

Since shikha has only 6 characters which is less than number specified in num\_chars argument it won’t raise an error instead it will return full string

1. SEARCH dax expression - You can use the SEARCH function to determine the location of a character or text string within another text string, and then use the MID function to return the text, or use the REPLACE function to change the text.
2. SUBSTITUTE dax expression - Use the SUBSTITUTE function when you want to replace specific text in a text string; use the REPLACE function when you want to replace any text of variable length that occurs in a specific location in a text string.

Eg when you want to replace @outlook.com to @gmail.com Use Substitute

Eg when you want to replace only till 2-3(variable length) characters with new string use Replace dax expression.

1. CROSSFILTER dax - The function returns no value; the function only sets the cross-filtering direction for the indicated relationship, for the duration of the query. CROSSFILTER can only be used in functions that take a filter as an argument, for example: CALCULATE, CALCULATETABLE, CLOSINGBALANCEMONTH, CLOSINGBALANCEQUARTER, CLOSINGBALANCEYEAR, OPENINGBALANCEMONTH, OPENINGBALANCEQUARTER, OPENINGBALANCEYEAR, TOTALMTD, TOTALQTD and TOTALYTD functions.

Tip – Specify the column name in the crossfilter dax expression which do not complete the **cycle** (which is in the opposite/against the direction of the cycle)**.**

Refer to this link - <https://youtu.be/8--Yv18rhyA>

Crossfilter just applies the relationship for that measure

1. Related dax expression –

Syntax – related(column\_name) = Here column name should always be from **ONE** side of relationship table.

Scenario where this is used : Suppose there is Sales table which has Sale\_id col, Sales Amount col, Territory\_key Col(Foreign Key). There is Territory table which has Territory key Col ,Country name Col. Now If I want to Find Total Sales Amount where I don’t know the territory key value but I know that Country Name Should be “United States”. And There is many-to-One relationship between Sales table and Territory table. Under that scenarios this dax expression can be used .

FILTER('InternetSales\_USD',RELATED('SalesTerritory'[SalesTerritoryCountry])="United States")

Note ; Related dax expression should always be used inside **Filter dax**

Eg – 1 Related\_DAX=SUMX(FILTER(Sales,RELATED(Country[Country])="India"),Sales[SalesAmount])

Difference between Related and RelatedTable dax expression is that:

a)Related dax only works from one side to many side of relationship whereas RelatedTable works on both the sides of the relationship.

b) Related dax returns a single value whereas RelatedTable returns a table

RelatedTable dax – You can’t use RelatedTable expression alone itself. It needs to be used inside Table argument inside calculate,filter dax expression etc.

Related goes from many to one side to grab one value from the target table

RelatedTable works from one-to-many side in order to retrieve all the rows that have a relationship with the current row context.

**Hint – If you are using Related dax expression, then Calculated Column is created under ManySide of relationship table, whereas if you are using RelatedTable dax expression Calculated Column is created under One side of relationship table**

Row Context – In Calculated Column there is Row Context by default. SUMX is an iterator which means it introduces row context.

Suppose if there is a measure profit = sumx(Sales, Sales(S.P) – Sales(C.P)) and if there is a slicer for product Name with Products(TV, Speakers,Mobile) etc and if we select a one value from the slicer then filter context is applied first and then the row context. Meaning first it will filter the table with the value selected from the slicer and then the (row context is applied) measure will be calculated.

**When the row context exists, the active relationship gets disabled automatically.**

1. USERELATIONSHIP – There cannot be 2 active relationships between 2 tables at the same time. So Suppose we want the inactive or the other relationship that is not the default one to be used in the Measure for some calculation then this USERELATIONSHIP dax can be used to make that relationship active only till that Calculation is involved. After that Calculation is finished, the default relationship will become active.

Hint : So basically USERELATIONSHIP dax is used to make the Inactive relationship Active during that part of calculation. You cannot use USERELATIONSHIP to create a relationship on the fly, that relationship must exist in the model.

A screenshot of a computer

Description automatically generated

We get this error since No relationship exists between between Delivery Date col and Date col.

A screenshot of a computer

Description automatically generated

Here we are getting the output because the inactive relationship between Delivery Date Column and Date Col already exists in the model and I have tried to activate the relationship using USERELATIONSHIP dax.

1. ADDCOLUMNS = Since the return type of AddColumn dax is a table so you have to create a new table and then write ADDCOLUMN dax expression. It does not modifies the existing table meaning it will not add a new column to the existing table itself, Instead it creates a new table with all the existing columns along with a new column.

This dax expression is also used inside Measures.

A computer screen shot of a computer

Description automatically generated

AddColumns dax is used inside Measure so here it will create a virtual table here in the Measure.

1. CROSSJOIN - Returns a table that contains the Cartesian product of all rows from all tables in the arguments. The columns in the new table are all the columns in all the argument tables.
2. Column names from *table* arguments must all be different in all tables or an error is returned.
3. The total number of rows returned by CROSSJOIN() is equal to the product of the number of rows from all tables in the arguments; also, the total number of columns in the result table is the sum of the number of columns in all tables. For example, if **TableA** has **rA** rows and **cA** columns, and **TableB** has **rB** rows and **cB** columns, and **TableC** has **rC** rows and **cC** column; then, the resulting table has **rA × rb × rC** rows and **cA + cB + cC** columns.
4. ADDMISSINGITEMS - Return type is Table. AddMissingitems is **always** used along with SummarizeColumns dax. So SummarizeColumns will not display rows for which the values of the Measure is blank. So to also include/display the rows where the values of the measure is blank AddMissingitems dax is used.

So the Column names that needs to be written in AddMissingitems dax parameter should be all the column names that are written in Summarize dax

Except the Measure which is mentioned in SummarizeColumn.

1. DATATABLE – This dax expression is used to create a new table by specifying the col name, datatype of colname and Inserting values inside that table like how we create a table in sql.

Eg:-

= DataTable("Name", STRING,

"Region", STRING

,{

{" User1","East"},

{" User2","East"},

{" User3","West"},

{" User4","West"},

{" User4","East"}

}

)

* Unlike DATATABLE, [Table Constructor](https://learn.microsoft.com/en-us/dax/table-constructor) allows any scalar expressions as input values.

So In Table\_Constructor (it is not a dax expression) without specifying the columnname, data type of col you can just specify directly the values (scalar) inside that.

Eg ; Table Constructor eg

A screenshot of a computer

Description automatically generated

1. DISTINCT(<colname>) - Returns a one-column table that contains the distinct values from the specified column. In other words, duplicate values are removed and only unique values are returned.
2. The results of DISTINCT are affected by the current filter context. For example, if you use the formula in the following example to create a measure, the results would change whenever the table was filtered to show only a particular region or a time period.
3. There is another version of the DISTINCT function, [DISTINCT (table)](https://learn.microsoft.com/en-us/dax/distinct-table-function-dax), that returns a table by removing duplicate rows from another table or expression.
4. Example

A screenshot of a computer

Description automatically generated

**Also, note that the results are not additive. That is to say, the total number of unique customers in 2007 is not the sum of unique customers of Accessories, Bikes and Clothing for that year. The reason is that a customer can be counted in multiple groups.**

**Because That CustomerKey can be unique on Accessories and the same CustomerKey can be unique on Bike as well since the Customer has purchased both the product but if I did year wise then that Customerkey will be only counted once. Hence the results are not additive.**

1. Union, intersect, EXCEPT(Except works same as Minus in sql) –
2. Union – If there are duplicates(or the rows that are present in both the tables) it will display that rows twice.

For union, there should be same number of columns in both the tables with same data type otherwise it will return error.

1. Intersect – It will display the common rows from both the table
2. Except – It will only display the records of the first/Left Table which are not present in second table. If a row appears at all in both tables, it and its duplicates are not present in the result set. If a row appears in only table\_expression1, it and its duplicates will appear in the result set.
3. SUMMARIZE – Its return type is a table, What makes a summarize dax expression unique is that the expressions of the newly created column are computed in both row context and filter context. Summarize creates a filter context with the values of the column you are grouping by but it also creates a row context that lets you evaluate the values of the current row of this dataset.

In Summarize group\_by\_Column\_name(2nd argument of Summarize dax) we can write column name from the table that was mentioned in the first argument of summarize dax or **we can write column\_name from its Related table also** .

A screenshot of a computer

Description automatically generated

1. Rollup is always used inside Summarize – it also displays the subtotals for the groupBy column. (Check the last rows – it give the subtotal by Name with all products from 2nd col)

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Here Since I have written Customers name inside Rollup dax function it will Roll up by Customers name and will display the subtotals by Products (2nd col).

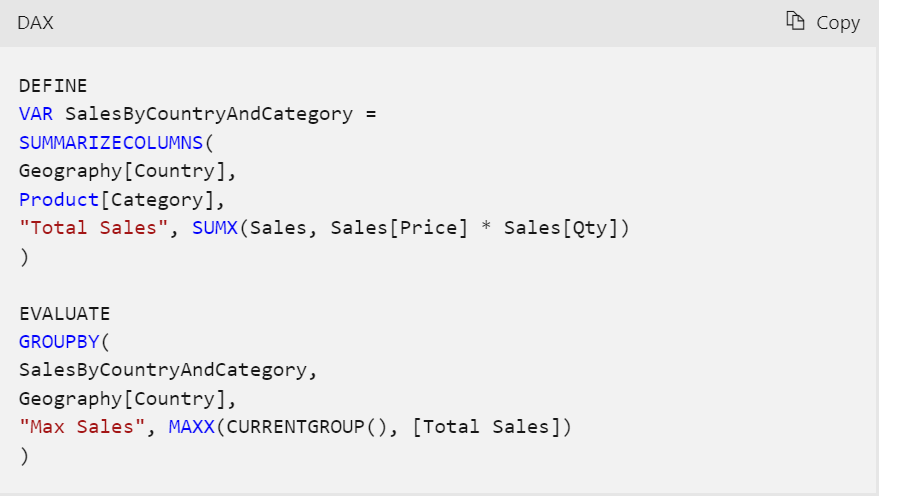
(Subtotal of each product with all Customers from 1st col)

1. GROUP BY DAX - The GROUPBY function is similar to the [SUMMARIZE](https://learn.microsoft.com/en-us/dax/summarize-function-dax) function. However, GROUPBY does not do an implicit [CALCULATE](https://learn.microsoft.com/en-us/dax/calculate-function-dax) for any extension columns that it adds. GROUPBY permits a new function, [CURRENTGROUP](https://learn.microsoft.com/en-us/dax/currentgroup-function-dax), to be used inside aggregation functions in the extension columns that it adds.

CURRENTGROUP dax is **always** used inside new extension column that is created in GroupBy dax expression.

Difference between GroupBy and Summarize is that Summarize dax already knows that the col names that are specified in GroupByColname argument of Summarize it should do calculations based on that group by columns. But For GROUPBY dax even though we have specified col name in the groupby argument of that dax, in the new (extension)column that we create inside dax expression we need to put CURRENTGROUP dax expression in the expression of the newly created column saying that we want to do calculation based on the groupby cols specified in that argument.

Eg:



* GROUPBY is primarily used to perform aggregations over intermediate results from DAX table expressions. For efficient aggregations over physical tables in the model, consider using [SUMMARIZECOLUMNS](https://learn.microsoft.com/en-us/dax/summarizecolumns-function-dax) or [SUMMARIZE](https://learn.microsoft.com/en-us/dax/summarize-function-dax) function.

1. SUMMARIZECOLUMN dax – Difference between Summarize and SummarizeColumn dax is that you can **add a filter condition** to the group by columns in **SummarizeColumn dax** but you can’t add a filter condition to the group by column of Summarize dax.

Another difference is that the op(Table) of the dax is not in the order in which col names are specified in the SummarizeColumn dax expression . But for Summarize dax it is in the same order in which it is specified in dax expression.

A screen shot of a computer

Description automatically generated

Summarize col dax with Filter

1. IGNORE dax – Ignore dax is always used inside SUMMARIZECOLUMNS dax expression. Syntax – Ignore(<Expression>)

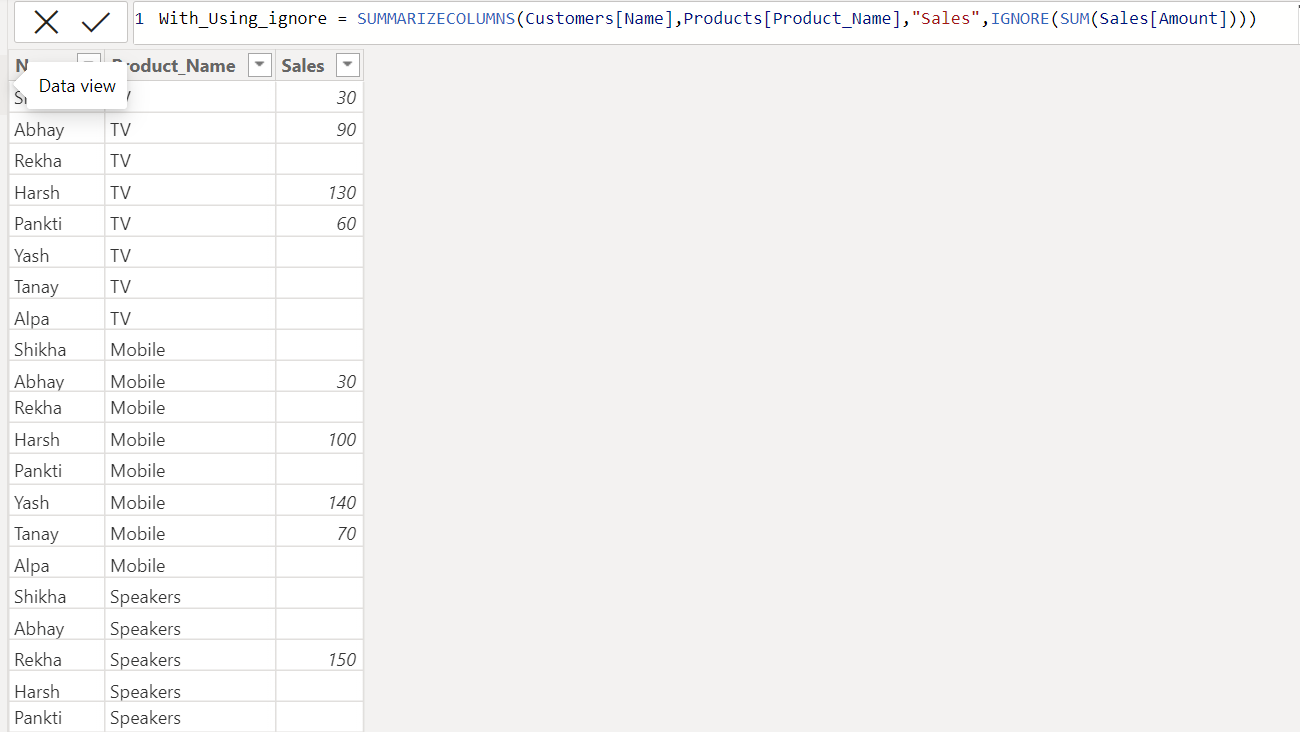
SummarizeColumns does not display the rows where the expression value is blank.

But By using Ignore inside SummarizeColumns dax it will also display the rows where the expression value is BLANK.

A screenshot of a computer

Description automatically generated

Without using ignore



With using ignore

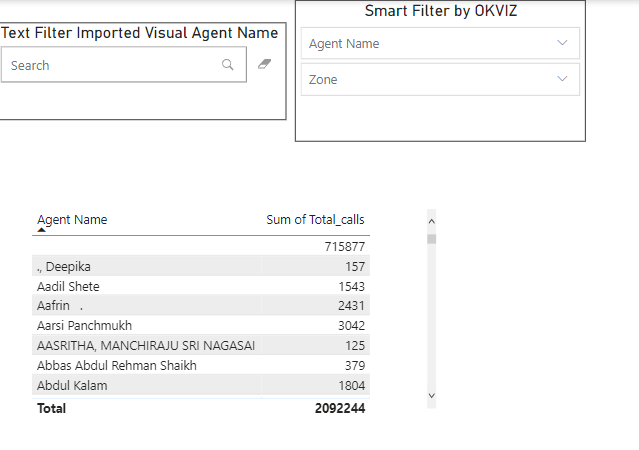
1. ROW dax - Returns a table with a single row containing values that result from the expressions given to each column.
2. **Imported Visuals in PowerBi** 
   1. Text Filter

A screenshot of a computer

Description automatically generated

It acts as a wildcard – and whatever text we put inside a text filter it searches as a wildcard to particular page of Power Bi report

* 1. Smart Filter by OKVIZ – It helps in doing multiple select of a value. You can add more than 1 column names in Smart Filter.



Here I have added Agent Name and Zone column in Smart Filter.

A screenshot of a computer

Description automatically generated

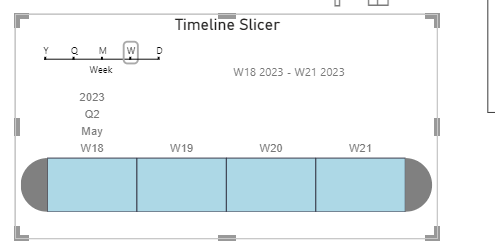
Here I have added multiple names in Agent Name filter , and the table below got filtered accordingly.

* 1. Play Axis – It also acts as a filter. The difference between Play Axis inside Scatter plot and this separate Play Axis visual is that Play Axis inside scatter plot will only do the animation/changes inside that scatter plot only,it will not changes the remaining graphs, where as with this imported PLAY axis visual it will change also the graphs in the page based on field.

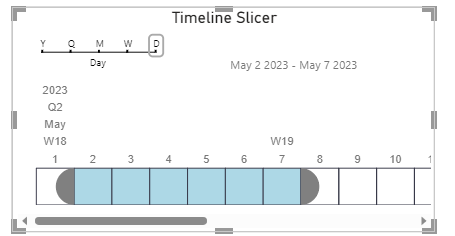
A screenshot of a computer

Description automatically generated

* 1. Timeline Slicer-



It is very useful as by using this we can convert dates to quarter,month,week,day by just clicking on the week,day,month above in top LHS



Here I got the data from 2nd May to 7th may

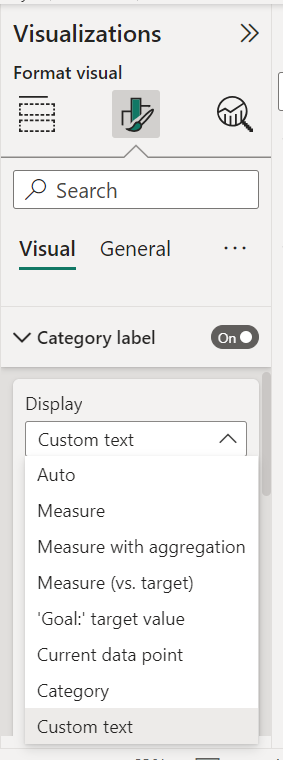
* 1. Card with States by Okviz –
     1. Hardcode the target values directly into the visual

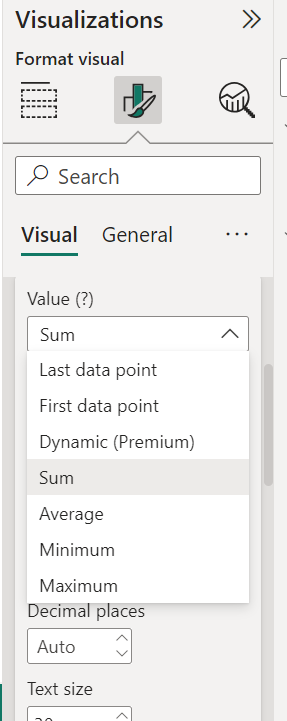
A graph on a screen

Description automatically generated

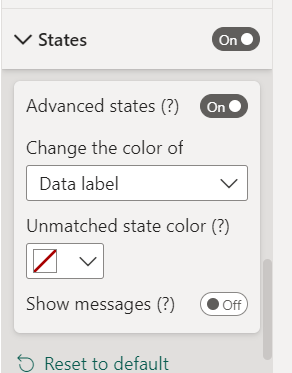
Target >436k is the Custom Text I have added

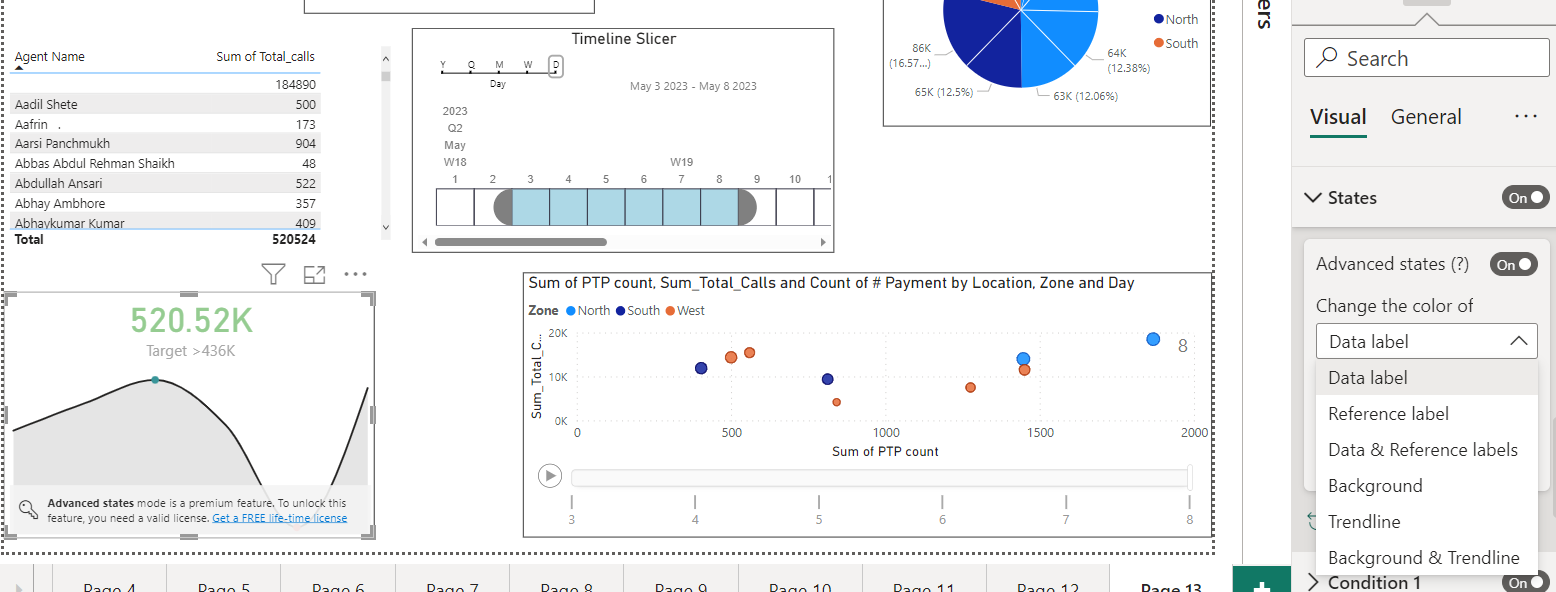
So for that go to Category Label and select Custom Text option.





Since my trendline axis is enables its asking me in the Data label attribute what should be displayed (First point means the at the first point in trend what is the calculation)Since I wanted the sum of the whole line I have selected it as Sum

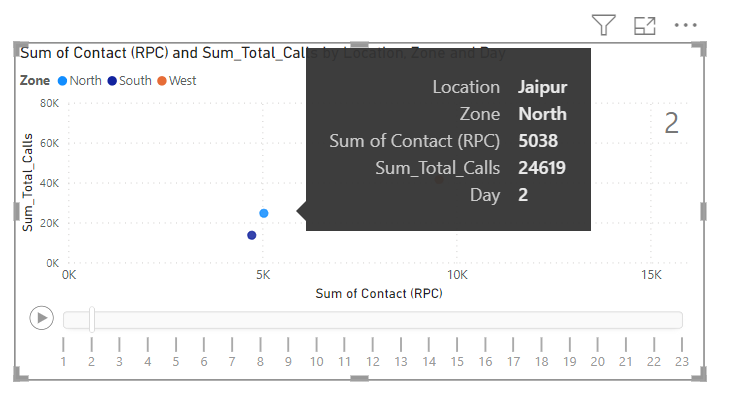




Since I want to change the color of the Data Label based on the value of that ,Enable the States and select Data Label

<https://www.youtube.com/watch?v=64-eK-tdTPc> – **Scatter Plot In Power Bi**

Usually it is better to put numeric values on X-axis as well as Y-axis on Scatter plot. And Put the subset of legend Attribute in Values Attribute.



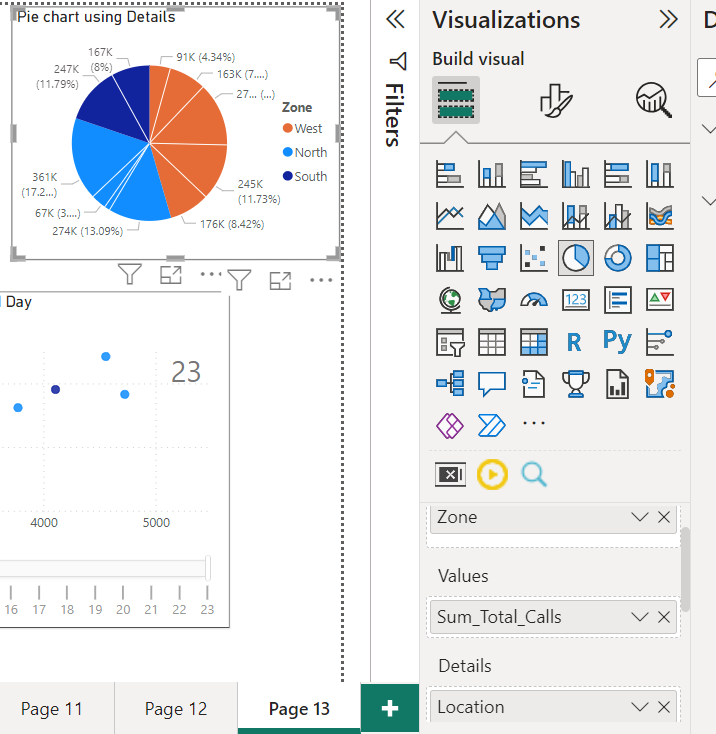
Now currently this tooltip points at Light-Blue dot and gives me information only of 2nd May Jaipur Location but if I want to see the trend for Jaipur Location throughout the month and I have to Ctrl+Click to that dot and it will give me full trend of that selected dot.

A graph with blue lines

Description automatically generated

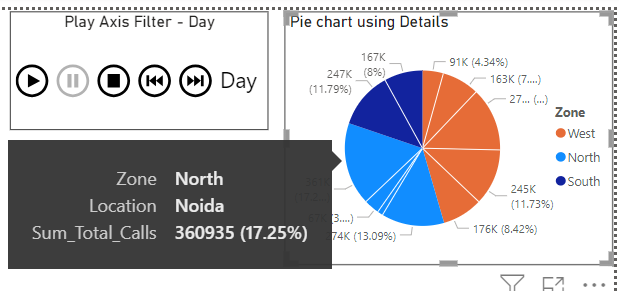
This is the trend of Jaipur Location

* Details Attribute in Power bi Visual – (Hint : - Details is always a subset of Legend) which means The column inside Detail should be a subset of the Column inside Legend Attribute.

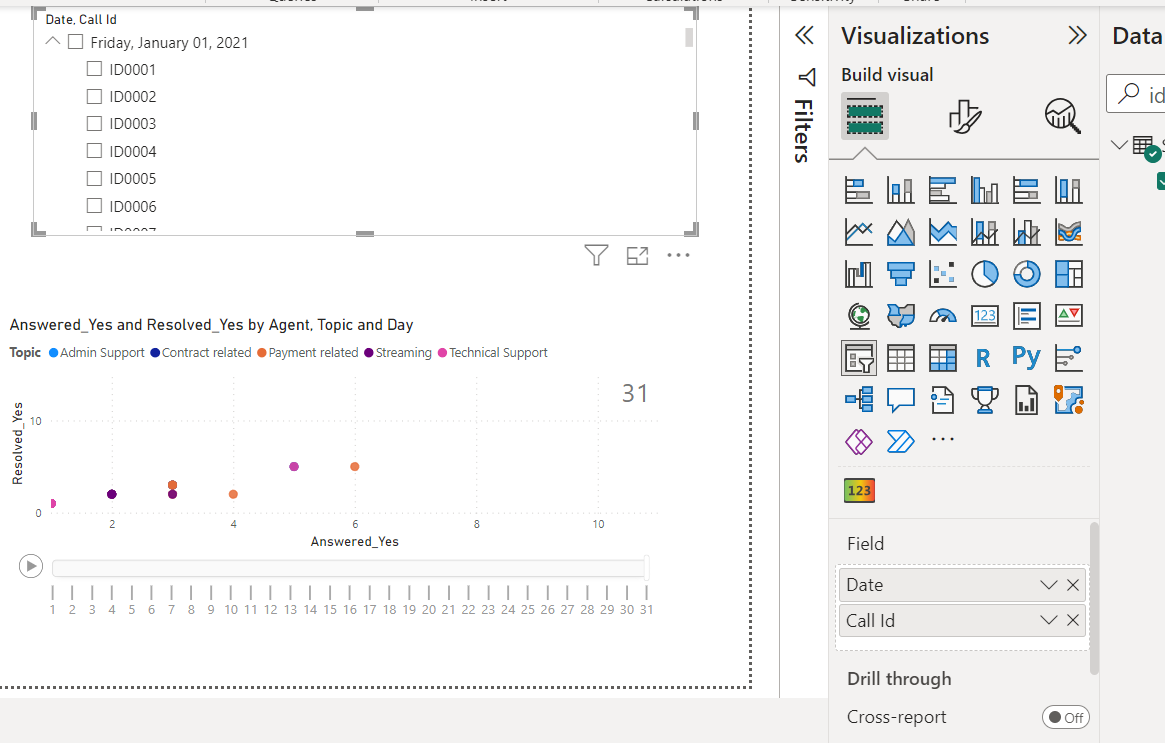


In this pie-chart we have used Details Attribute. So in Legend we have used Zone column and in Details we have used Location column (check at the bottom rhs).

So inside each Zone(Legend Column) we can deep-dive and also check the contributions of each Location(Details Column)



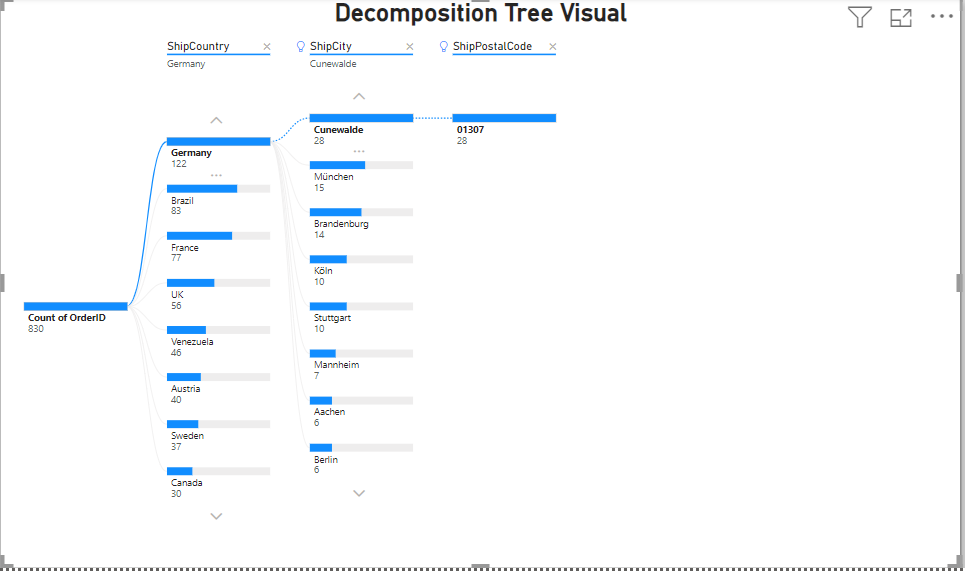
* We can create a hierarchy in a slicer by putting multiple fiels inside sliver visual



Here I have put date and below that I have put Call\_id column so it created a hierarchy in one single slicer itself

So basically, under that date which all entries of the Call id are present I only get those CallId under that date

**Decomposition tree Visual**

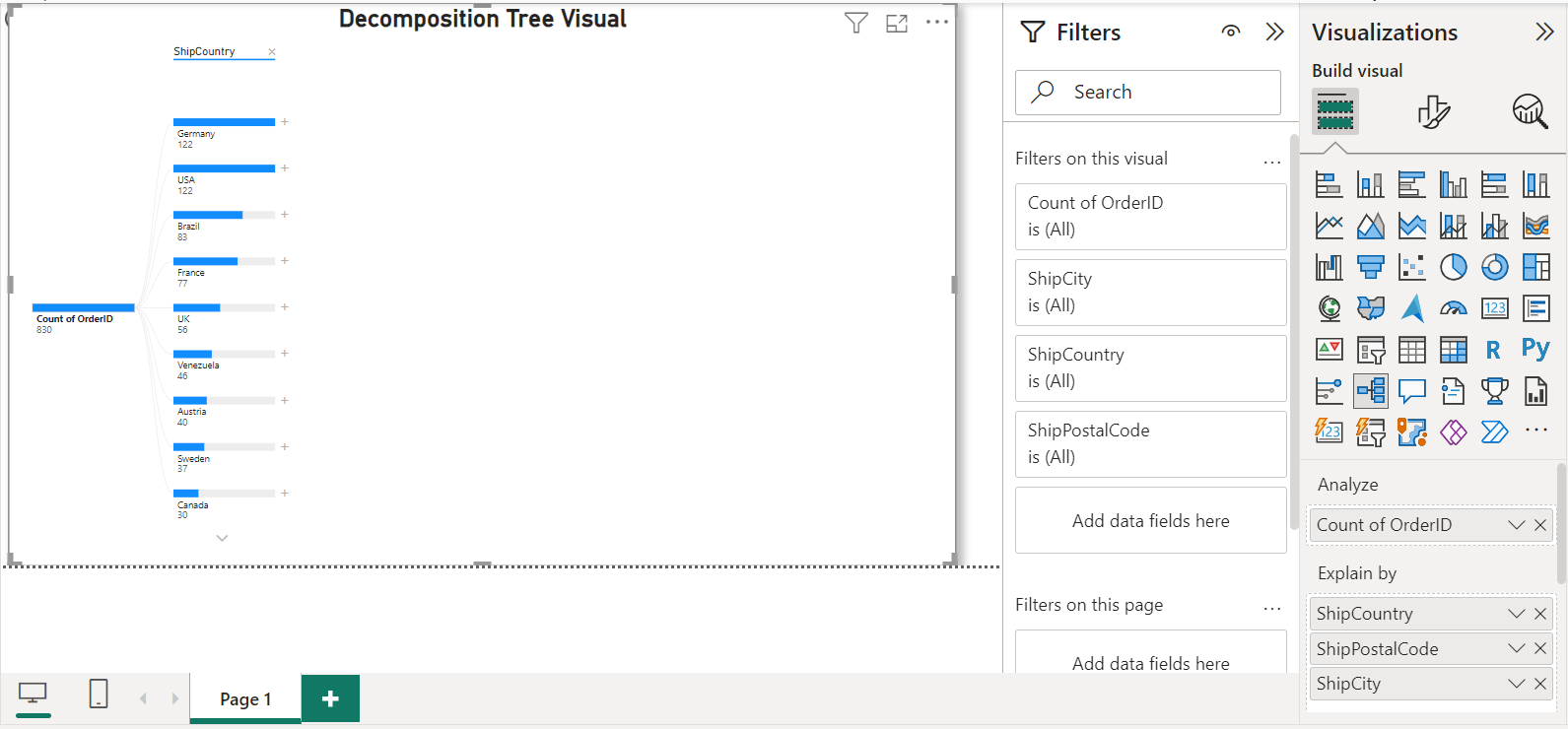


So Decomposition Tree can be used when we want to show the bifurcation Suppose for Example Count of Orders by Country -->City-->Region

A screenshot of a computer

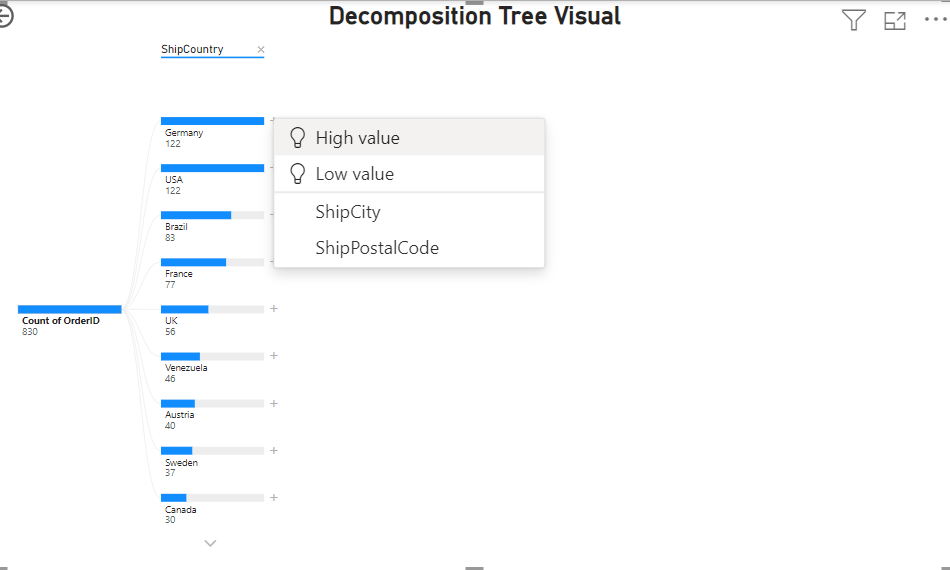
Description automatically generated

If we click on + then it will futher split by the next field(Here it follows the order which is there in the Explain By attribute of that visual )



So here for this visual the order is Country -->City -->Region

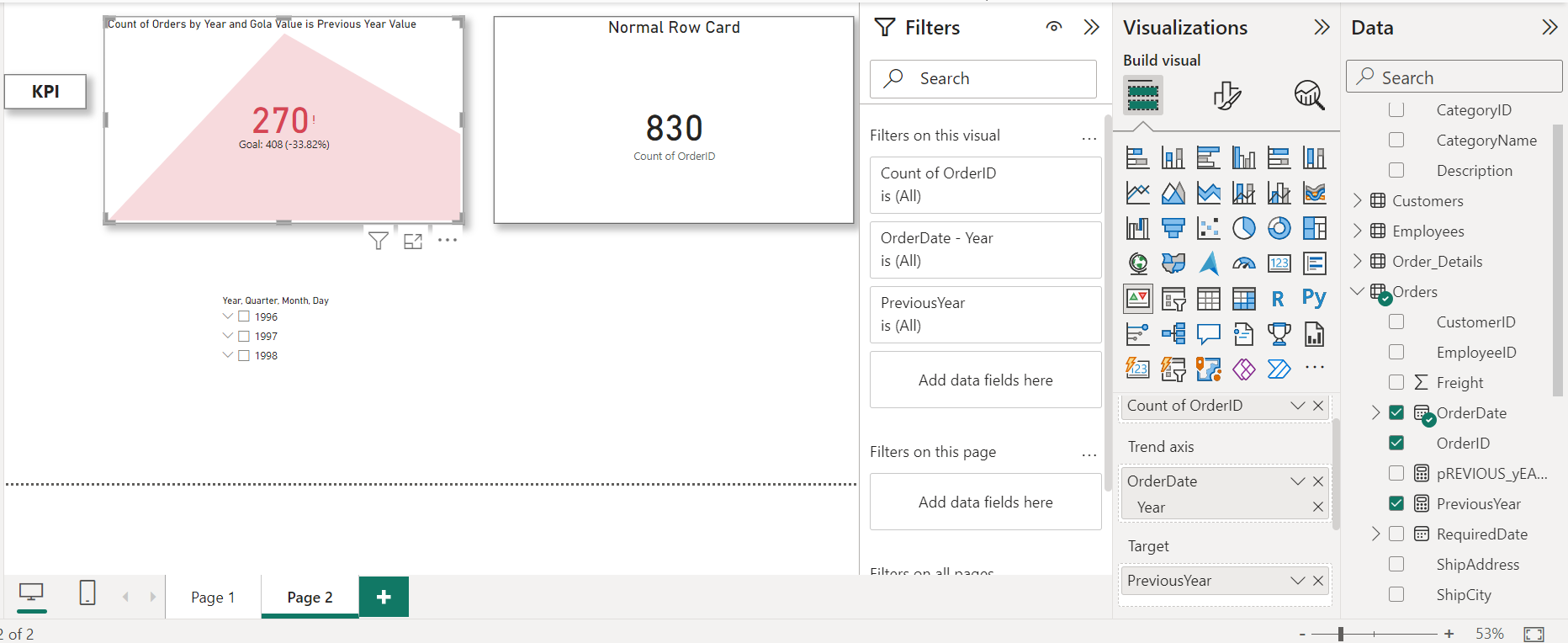
However we can change this order.



So suppose now we want to split Ship City by High Value so what is does is , it shows all the City Values based on the Country Selected and the High Value of that city is highlighted in Bold

**KPI Visual -** This is used when we want to see a metric value Against a Goal/Target Value along with a Trendline Axis.   
Target Value Can be One static Value or a previous Month/Year/Quarter Total Sales for eg.

Whereas in Normal Row Card only a value is Seen (No option of viewing Goal along with Trendline Axis)



The value (270) that you see will always be the Latest Value

For eg in Data the values consist of 3 years (1996,1997,1998) , So here 270 is the Count of orders of 1998 year(Latest Value of year according to data) and in the Target We have added Dax expression of Previous year . So here in Target it is Total Orders of 1997 year(Previous year)

1 PreviousYear = CALCULATE(COUNT(Orders[OrderID]),PREVIOUSYEAR(Orders[OrderDate].[Date]))

