DAX Cheat sheet with examples— Part 1

Overview

In this article I have presented some of the frequent DAX queries I was encountering in my day to day work. The objective of this article is to help users with DAX through an example based approach. I found that it is easier to comprehend the nuances of the DAX language if it supported by simple examples. Note — it is not strictly neccessary that the results have to be obtained via DAX only. If the data model is good then Power BI visuals can often meet the requirements. In this article I have covered the following DAX expressions:

1. EVALUATE
2. DEFINE
3. TABLE
4. COLUMN
5. MEASURE
6. MIN
7. MAX
8. TOPN
9. SELECTCOLUMNS
10. SUM
11. UPPER
12. DISTINCT
13. ORDERBY
14. UNION
15. ROW

16. COUNTBLANK

17. COUNTROWS

18. FILTER

19. IF

20. ISBLANK

21. SUMMARIZE

22. SUMMARIZECOLUMNS

23. GROUPBY

24. CURRENTGROUP

25. COUNTX

26. SUMX

27. MINX

28. MAXX

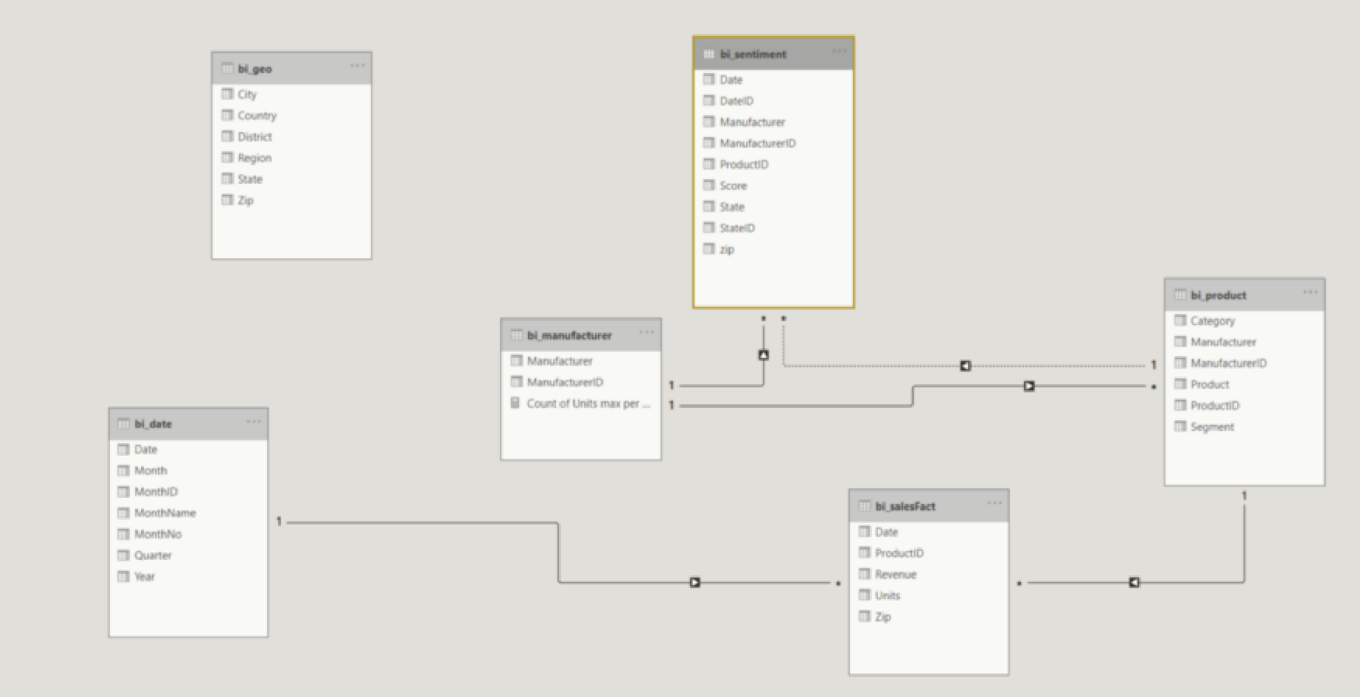
29. SUMX

30. CURRENTGROUP

Sample data

The DAX expressions in this article are written around the MS Access sample database downloadable from Microsoft Learning. A copy of the same can also be downloaded from my Github repo here. A copy of the Power BI report which references this MS Access database can be downloaded from my Github repo here.

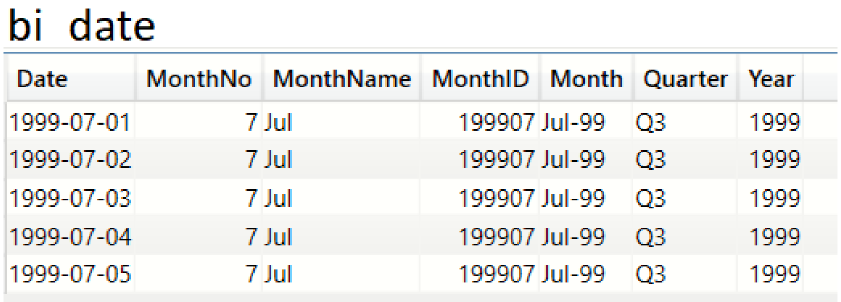
Database schema



Power BI model

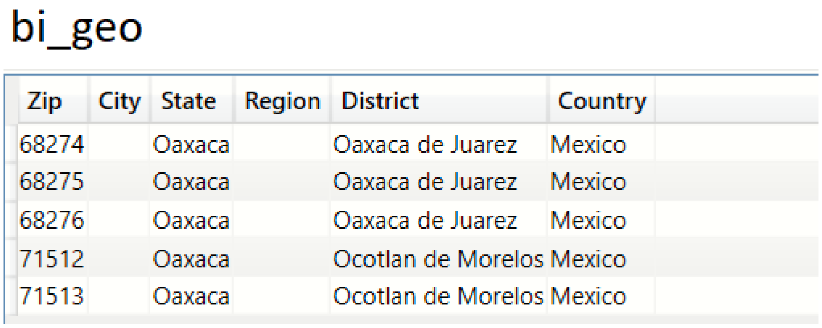
Data

To get a feel of what the data looks like I have presented the top 5 rows from each of the tables in this database



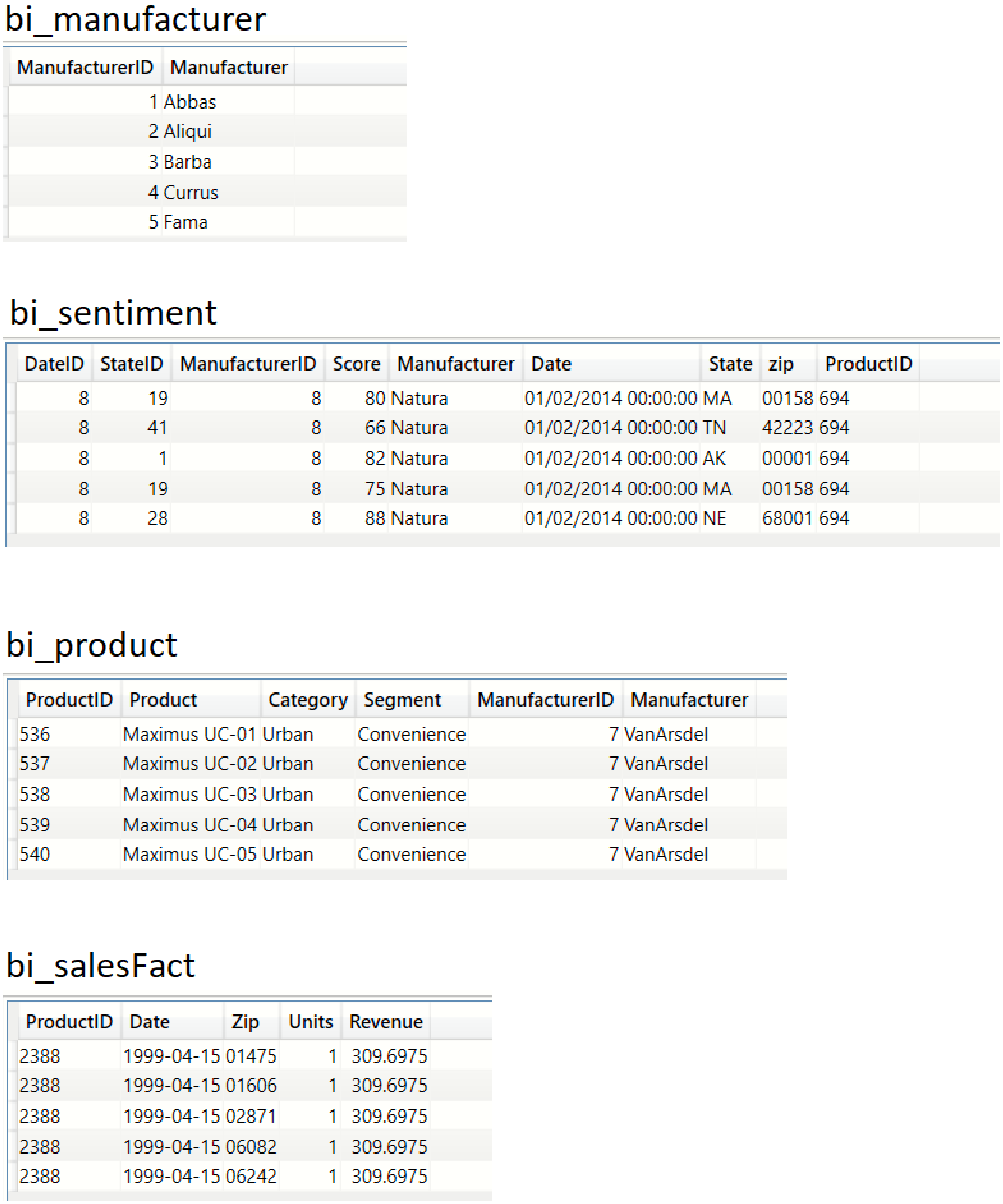
bi date

Date	MonthNo	MonthName	MonthID	Month	Quarter	Year
1999-07-01	7	Jul	199907	Jul-99	Q3	1999
1999-07-02	7	Jul	199907	Jul-99	Q3	1999
1999-07-03	7	Jul	199907	Jul-99	Q3	1999
1999-07-04	7	Jul	199907	Jul-99	Q3	1999
1999-07-05	7	Jul	199907	Jul-99	Q3	1999



bi_geo

Zip	City	State	Region	District	Country
68274		Oaxaca		Oaxaca de Juarez	Mexico
68275		Oaxaca		Oaxaca de Juarez	Mexico
68276		Oaxaca		Oaxaca de Juarez	Mexico
71512		Oaxaca		Ocotlan de Morelos	Mexico
71513		Oaxaca		Ocotlan de Morelos	Mexico



bi_manufacturer

ManufacturerID	Manufacturer
1	Abbas
2	Aliqui
3	Barba
4	Currus
5	Fama

bi_sentiment

DateID	StateID	ManufacturerID	Score	Manufacturer	Date	State	zip	ProductID
8	19	8	80	Natura	01/02/2014 00:00:00	MA	00158	694
8	41	8	66	Natura	01/02/2014 00:00:00	TN	42223	694
8	1	8	82	Natura	01/02/2014 00:00:00	AK	00001	694
8	19	8	75	Natura	01/02/2014 00:00:00	MA	00158	694
8	28	8	88	Natura	01/02/2014 00:00:00	NE	68001	694

bi_product

ProductID	Product	Category	Segment	ManufacturerID	Manufacturer
536	Maximus UC-01	Urban	Convenience	7	VanArsdel
537	Maximus UC-02	Urban	Convenience	7	VanArsdel
538	Maximus UC-03	Urban	Convenience	7	VanArsdel
539	Maximus UC-04	Urban	Convenience	7	VanArsdel
540	Maximus UC-05	Urban	Convenience	7	VanArsdel

bi_salesFact

ProductID	Date	Zip	Units	Revenue
2388	1999-04-15	01475	1	309.6975
2388	1999-04-15	01606	1	309.6975
2388	1999-04-15	02871	1	309.6975
2388	1999-04-15	06082	1	309.6975
2388	1999-04-15	06242	1	309.6975

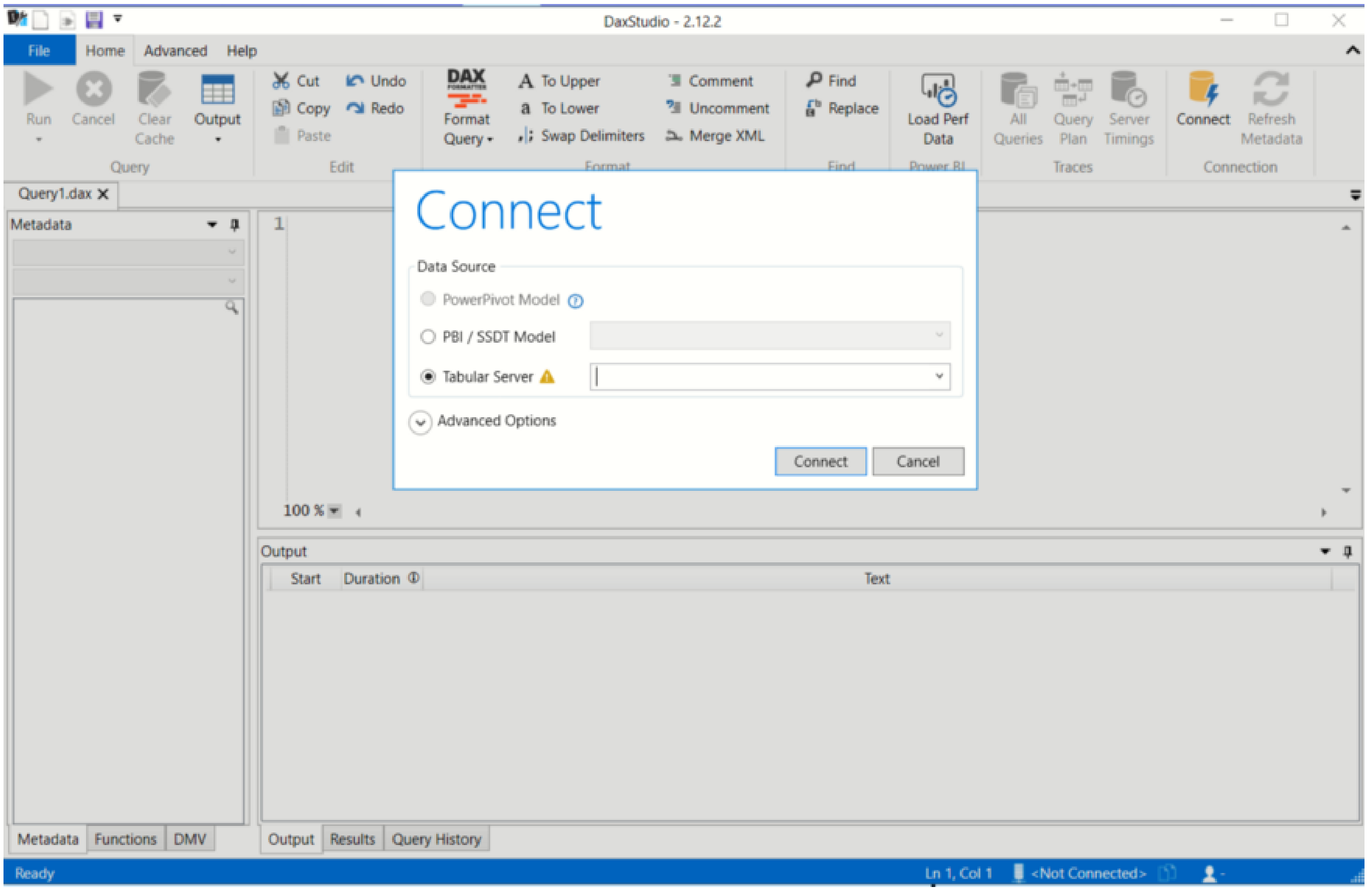
Top N rows from all tables

DAX studio primer

How to use DAX studio?

DAX studio from Microsoft is a very handy tool if you want to experiment with DAX queries outside of Power BI. I have listed some informative videos below. DAX Studio runs independently of Power BI, however it expects a running instance of Power BI to establish a connection.

- DAX studio tutorial: What should I use it for, tool overview
- Why you should use DAX Studio with Power BI
- Computing a measure in DAX Studio



Connect

Data Source

○ PowerPivot Model

○ PBI / SSDT Model

● Tabular Server

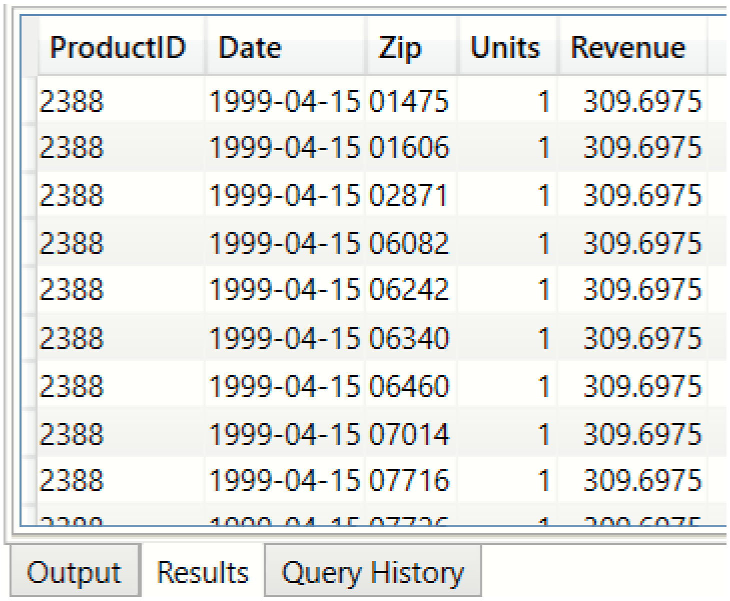
Advanced Options

Connect Cancel

How to execute Table expressions?

DAX studio expects any table expression to be encapsulated inside a `EVALUATE()` block. In the following example, we are inspecting the first 20 rows of the **bi_salesFact** table.

```
EVALUATE
(
        TOPN(20, bi_salesFact)
)
```

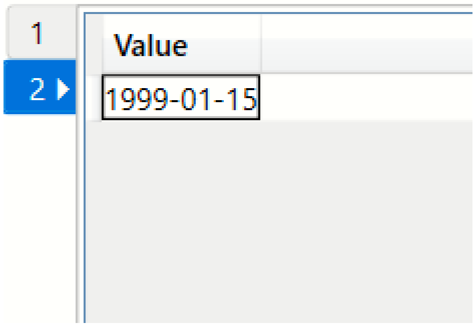


ProductID	Date	Zip	Units	Revenue
2388	1999-04-15	01475	1	309.6975
2388	1999-04-15	01606	1	309.6975
2388	1999-04-15	02871	1	309.6975
2388	1999-04-15	06082	1	309.6975
2388	1999-04-15	06242	1	309.6975
2388	1999-04-15	06340	1	309.6975
2388	1999-04-15	06460	1	309.6975
2388	1999-04-15	07014	1	309.6975
2388	1999-04-15	07716	1	309.6975
2388	1999-04-15	07726	1	309.6975

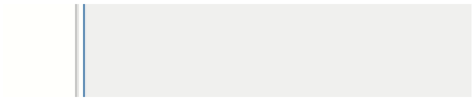
How to execute Scalar expressions?

To execute any expression that returns a scalar value (i.e. not a table) encapsulate the expression in a `EVALUATE {}` block

```
EVALUATE
{
MAX(bi_salesFact[Date])
}

EVALUATE
{
MIN(bi_salesFact[Date])
}
```



	Value
1	
2	1999-01-15



How to create a measure (MEASURE,SUM)?

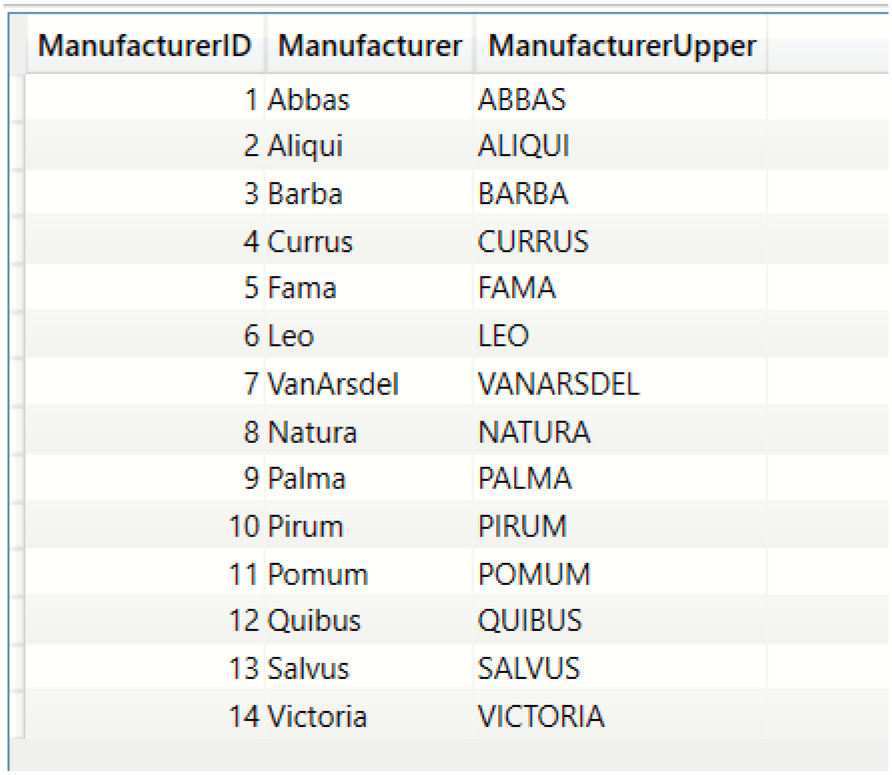
In this example we are calculating the total sales per manufacturer. When using DAX studio, the `DEFINE` keyword should be used to create a new `MEASURE` and this declaration should precede the `EVALUATE()` keyword.

```
DEFINE
MEASURE bi_manufacturer[TotalUnits]= SUM(bi_salesFact[Units])
EVALUATE
(
SELECTCOLUMNS
        (
        bi_manufacturer,
        "id",bi_manufacturer[ManufacturerID],
        "name",bi_manufacturer[Manufacturer],
        "TotalUnits",bi_manufacturer[TotalUnits]
        )

)
```

How to create a calculated column(UPPER,COLUMN)?

In the following example we are creating a new column which converts the manufacturer name to upper case

```
DEFINE
COLUMN bi_manufacturer[ManufacturerUpper]=
UPPER(bi_manufacturer[Manufacturer])
EVALUATE
(
bi_manufacturer
)
```

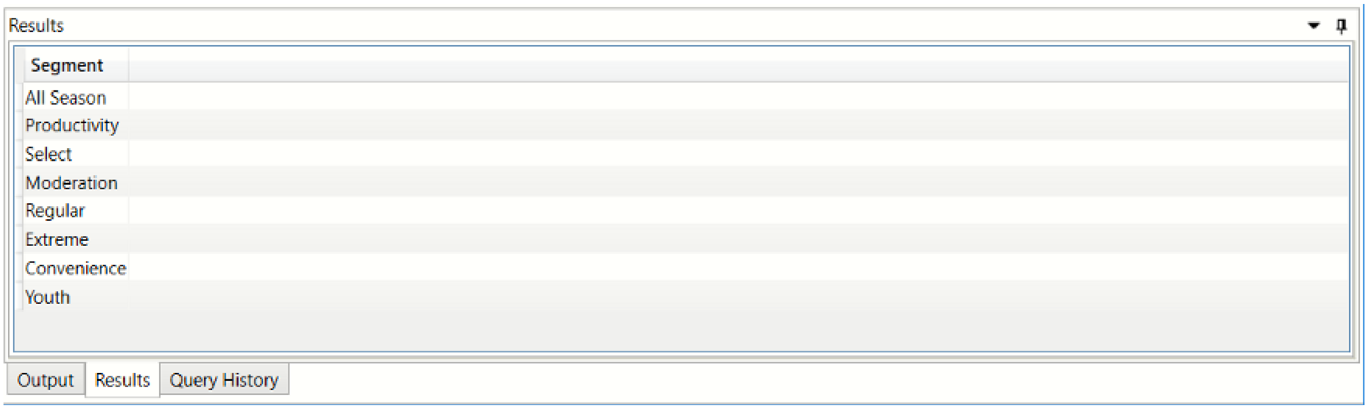


ManufacturerID	Manufacturer	ManufacturerUpper
1	Abbas	ABBAS
2	Aliqui	ALIQUI
3	Barba	BARBA
4	Currus	CURRUS
5	Fama	FAMA
6	Leo	LEO
7	VanArsdel	VANARSDEL
8	Natura	NATURA
9	Palma	PALMA
10	Pirum	PIRUM
11	Pomum	POMUM
12	Quibus	QUIBUS
13	Salvus	SALVUS
14	Victoria	VICTORIA

List of unique Product Segments (DISTINCT, ORDER BY)

In this example we are displaying an unique list of product segments.

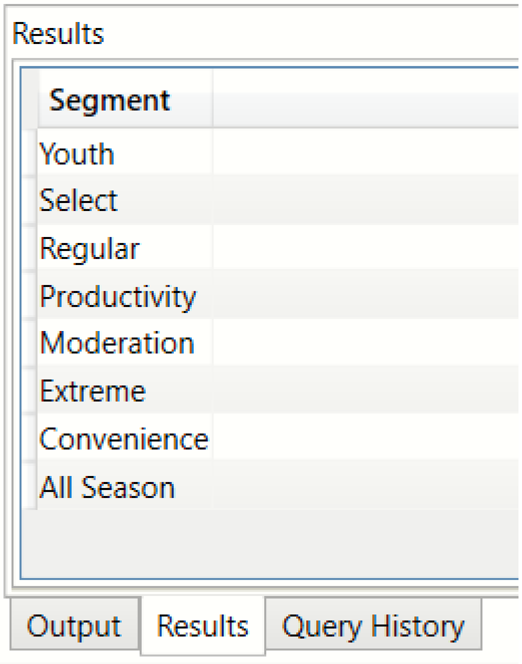
```
EVALUATE
(
DISTINCT( bi_product[Segment])
)
```



Segment
All Season
Productivity
Select
Moderation
Regular
Extreme
Convenience
Youth

Use the `ORDER BY` tag if neccessary

```
EVALUATE
(
DISTINCT( bi_product[Segment])
) ORDER BY bi_product[Segment] DESC
```

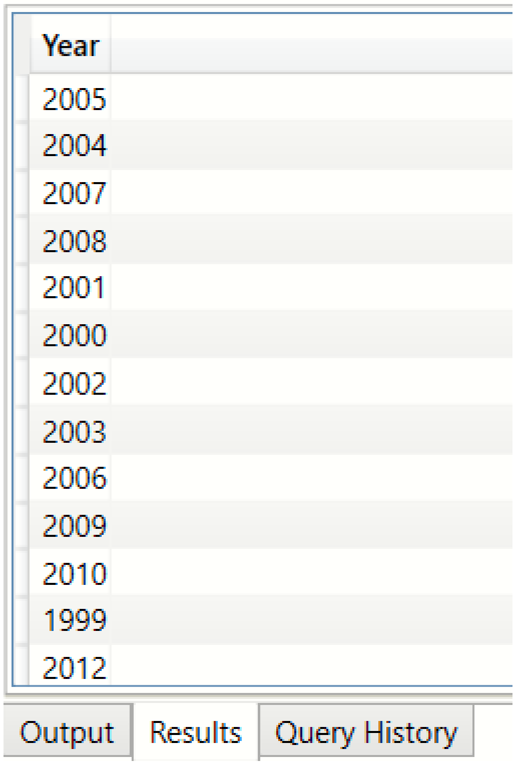


Segment
Youth
Select
Regular
Productivity
Moderation
Extreme
Convenience
All Season

Distinct list of financial years from the Sales table (DISTINCT)

In this query we are creating a calculated column to get the year component from the sales transaction date and then using the `DISTINCT` on the `year` column

```
DEFINE
COLUMN bi_salesFact[Year] = year(bi_salesFact[Date])

EVALUATE
(
DISTINCT( bi_salesFact[Year] )
)
```

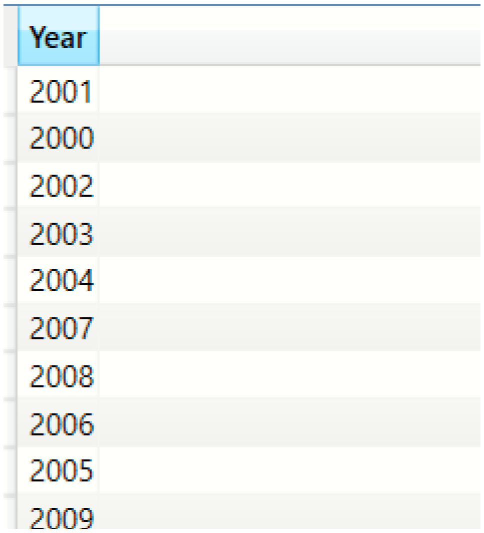


Year
2005
2004
2007
2008
2001
2000
2002
2003
2006
2009
2010
1999
2012

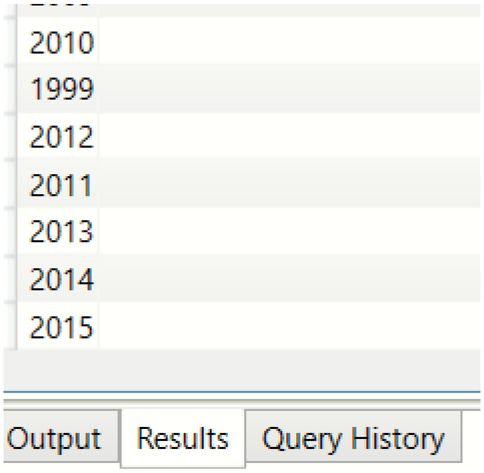
Distinct list of financial years from the Sales table(VALUES)

The `VALUES` expression has a similar behaviour to `DISTINCT`

```
DEFINE
COLUMN bi_salesFact[Year] = year(bi_salesFact[Date])

EVALUATE
(
VALUES( bi_salesFact[Year] )
)
```



Year
2001
2000
2002
2003
2004
2007
2008
2006
2005
2009

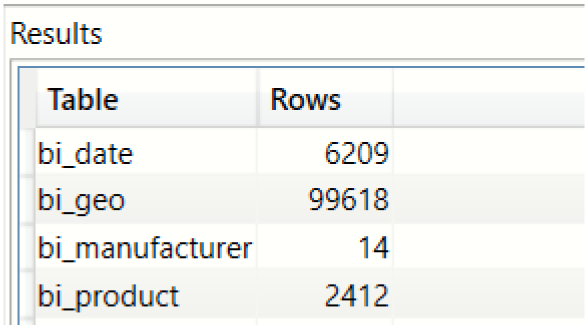


2010
1999
2012
2011
2013
2014
2015

Count of rows from all the tables (ROW,UNION)

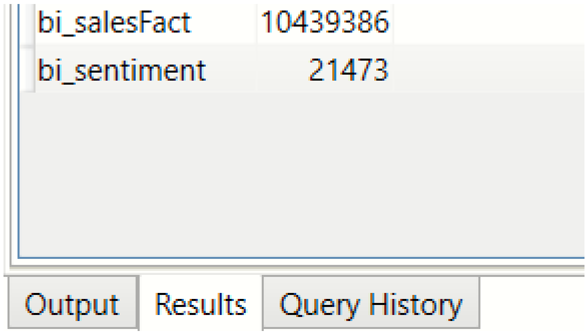
This helps towards the answering the question — “*How much data does my dataset hold?*”

```
EVALUATE
(
        UNION
        (
        ROW("Table","bi_date","Rows",{COUNTROWS(bi_date)}),
        ROW("Table","bi_geo","Rows",{COUNTROWS(bi_geo)}),
        ROW("Table","bi_manufacturer","Rows",
{COUNTROWS(bi_manufacturer)}),
        ROW("Table","bi_product","Rows",{COUNTROWS(bi_product)}),
        ROW("Table","bi_salesFact","Rows",
{COUNTROWS(bi_salesFact)}),
        ROW("Table","bi_sentiment","Rows",{COUNTROWS(bi_sentiment)})
        )

)
```



Results

Table	Rows
bi_date	6209
bi_geo	99618
bi_manufacturer	14
bi_product	2412



bi_salesFact	10439386
bi_sentiment	21473

In the following example we have added an `ORDER BY` clause

```
EVALUATE
(
        UNION
        (
        ROW("Table","bi_date","Rows",{COUNTROWS(bi_date)}),
        ROW("Table","bi_geo","Rows",{COUNTROWS(bi_geo)}),
        ROW("Table","bi_manufacturer","Rows",
{COUNTROWS(bi_manufacturer)}),
        ROW("Table","bi_product","Rows",{COUNTROWS(bi_product)}),
        ROW("Table","bi_salesFact","Rows",
{COUNTROWS(bi_salesFact)}),
        ROW("Table","bi_sentiment","Rows",{COUNTROWS(bi_sentiment)})
        )

) ORDER BY [Rows] DESC
```

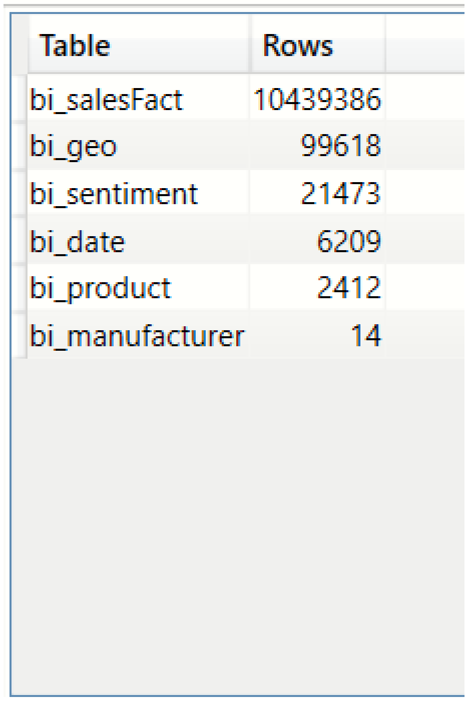
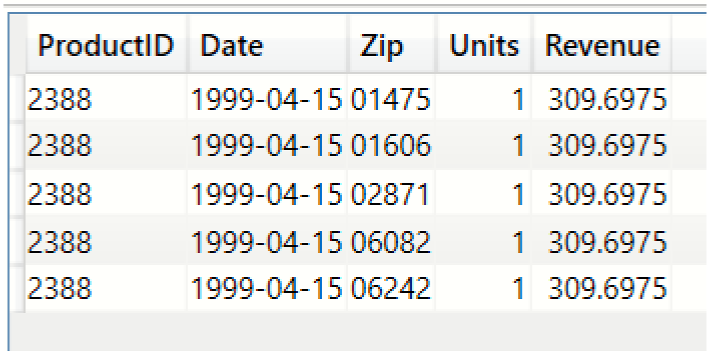


Table	Rows
bi_salesFact	10439386
bi_geo	99618
bi_sentiment	21473
bi_date	6209
bi_product	2412
bi_manufacturer	14

Display N rows from a table (TOPN)

Use this when you want to do a quick visual inspection of a table.

```
EVALUATE
(
TOPN (5,bi_salesFact)
)
```



ProductID	Date	Zip	Units	Revenue
2388	1999-04-15	01475	1	309.6975
2388	1999-04-15	01606	1	309.6975
2388	1999-04-15	02871	1	309.6975
2388	1999-04-15	06082	1	309.6975
2388	1999-04-15	06242	1	309.6975

The `TOPN` expression can also order the results

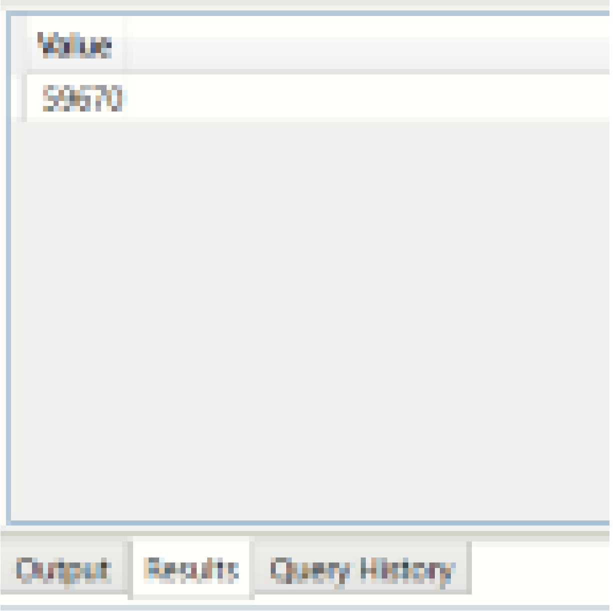
```
EVALUATE
(
TOPN ( 5, bi_salesFact, bi_salesFact[Units], DESC )
)
```

Find rows with blank column values (COUNTBLANK, FILTER,COUNTROWS)

This answers the question. *How many rows in the bi_geo table do not have a Region value*?

```
EVALUATE
{
```

```
COUNTBLANK(bi_geo[Region])
}
```



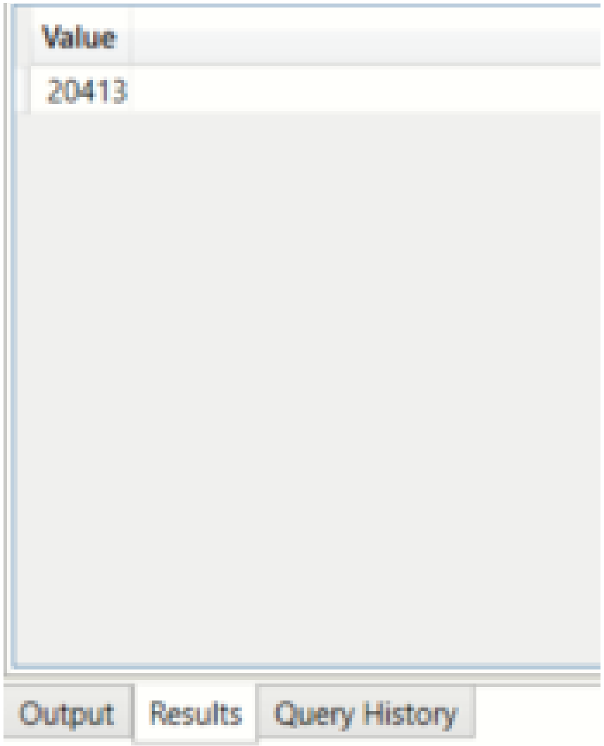
Value
59670

The same result can also be achieved by using `COUNTROWS` on a `FILTER` expression

```
EVALUATE
{
          COUNTROWS
                    (
                    FILTER(bi_geo, ISBLANK(bi_geo[Region]))
                    )
}
```

In the following example we are counting blank regions for a specific country

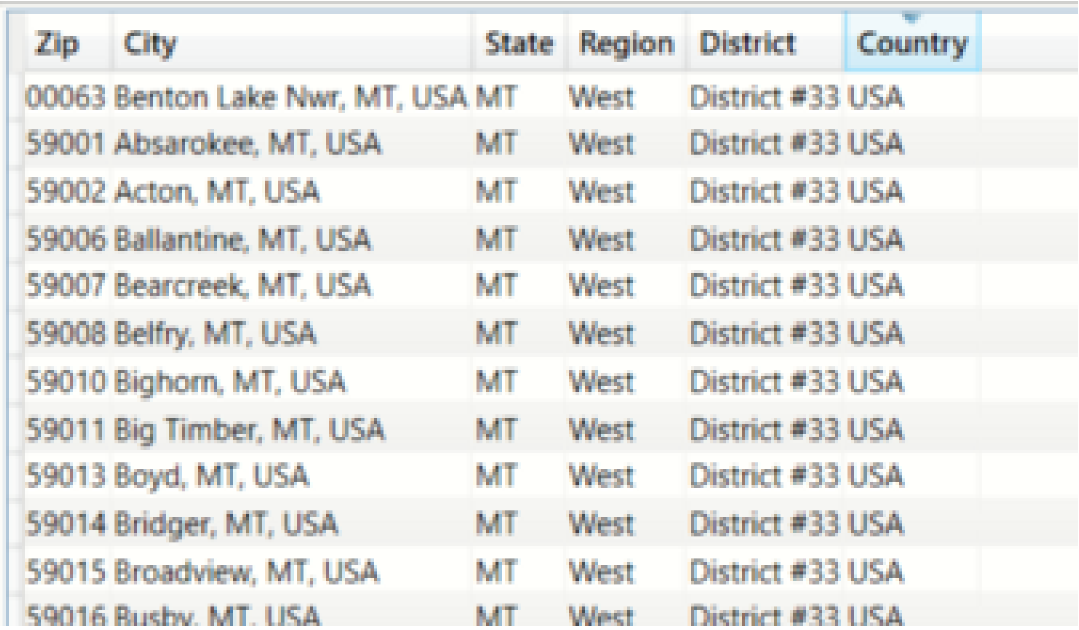
```
EVALUATE
{
          COUNTROWS
                    (
                    FILTER(bi_geo, ISBLANK(bi_geo[Region]) &&
bi_geo[Country]="France")
                    )
}
```



Value
20413

In the following example we are displaying all rows where **Region** is non-blank

```
EVALUATE
 (
         FILTER (
            bi_geo,
            ISBLANK ( bi_geo[Region] )=FALSE
        )
 )
```



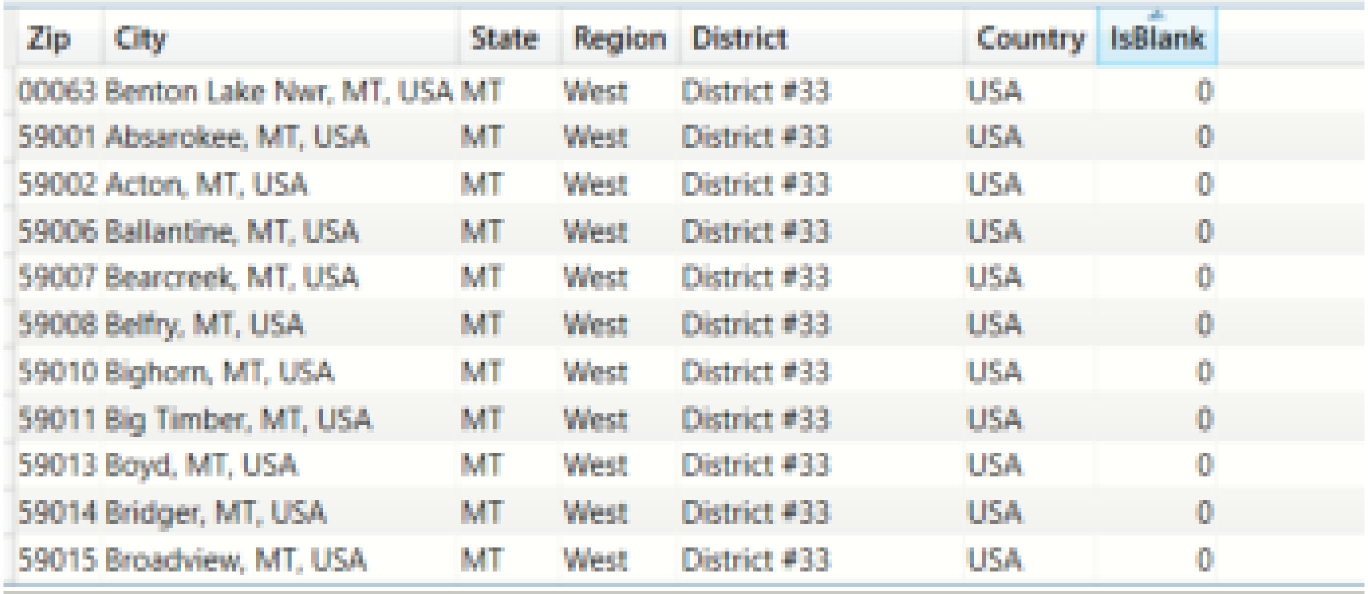
Zip	City	State	Region	District	Country
00063	Benton Lake Nwr, MT, USA	MT	West	District #33	USA
59001	Absarokee, MT, USA	MT	West	District #33	USA
59002	Acton, MT, USA	MT	West	District #33	USA
59006	Ballantine, MT, USA	MT	West	District #33	USA
59007	Bearcreek, MT, USA	MT	West	District #33	USA
59008	Belfry, MT, USA	MT	West	District #33	USA
59010	Bighorn, MT, USA	MT	West	District #33	USA
59011	Big Timber, MT, USA	MT	West	District #33	USA
59013	Boyd, MT, USA	MT	West	District #33	USA
59014	Bridger, MT, USA	MT	West	District #33	USA
59015	Broadview, MT, USA	MT	West	District #33	USA
59016	Busby, MT, USA	MT	West	District #33	USA

59018	Clyde Park, MT, USA	MT	West	District #33 USA

Add a calculated column to return 1 if region is blank otherwise 0 (ISBLANK, IF)

In this example we are creating a new calculated column on the table `bi_region` and using the `IF` expression to return either 1 or 0

```
EVALUATE
{
        COUNTROWS
                (
                FILTER(bi_geo, ISBLANK(bi_geo[Region]))
                )
}

DEFINE
COLUMN bi_geo[IsBlank] = IF( ISBLANK(bi_geo[Region]) ,1,0)
EVALUATE
(
        bi_geo
)
```

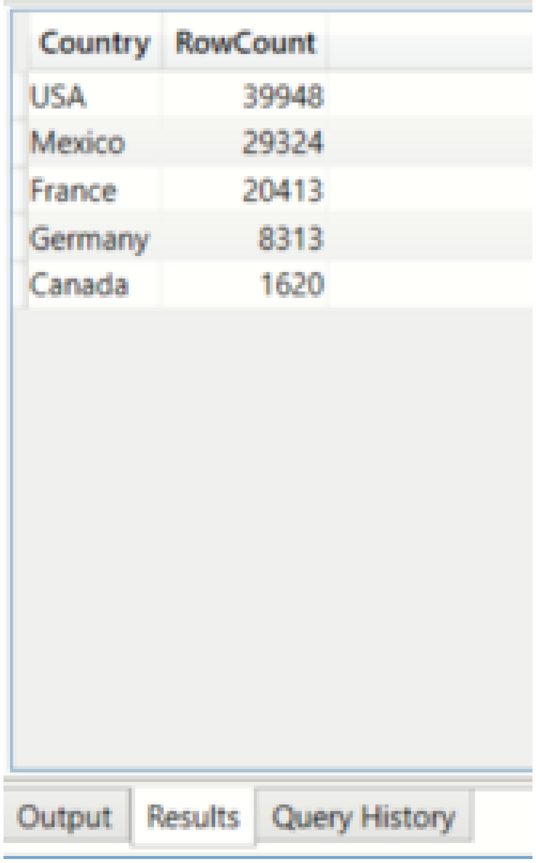


Zip	City	State	Region	District	Country	IsBlank
00063	Benton Lake Nwr, MT, USA	MT	West	District #33	USA	0
59001	Absarokee, MT, USA	MT	West	District #33	USA	0
59002	Acton, MT, USA	MT	West	District #33	USA	0
59006	Ballantine, MT, USA	MT	West	District #33	USA	0
59007	Bearcreek, MT, USA	MT	West	District #33	USA	0
59008	Belfry, MT, USA	MT	West	District #33	USA	0
59010	Bighorn, MT, USA	MT	West	District #33	USA	0
59011	Big Timber, MT, USA	MT	West	District #33	USA	0
59013	Boyd, MT, USA	MT	West	District #33	USA	0
59014	Bridger, MT, USA	MT	West	District #33	USA	0
59015	Broadview, MT, USA	MT	West	District #33	USA	0

What is the distribution of values in the Country column of the bi_geo table? (SUMMARIZE)

In this example we want to know the distinct list of countries and the total number of rows per country

```
EVALUATE
(
        SUMMARIZE(
                bi_geo,
                bi_geo[Country],
                "RowCount",COUNT(bi_geo[Country])
                )
) order by [RowCount] desc
```



Country	RowCount
USA	39948
Mexico	29324
France	20413
Germany	8313
Canada	1620

What is the distribution of values in the Region columm of the bi_geo table? (SUMMARIZE,SUMMARIZECOLUMNS,GROUPBY)

This verifies that total rows(99618)=total non blanks(18929+14512+6507) + total blanks(59670). Note the presence of the blank row and the value of the `Count` is blank too. This is because by default the `SUMMARIZE` , `SUMMARIZECOLUMNS` and `GROUPBY` functions ignore blanks.

Example using SUMMARIZE

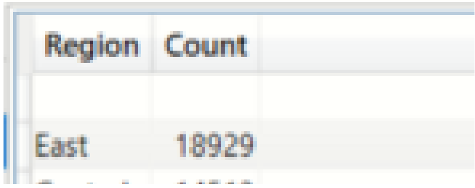
```
EVALUATE
(
SUMMARIZE(bi_geo,bi_geo[Region], "Count", COUNT(bi_geo[Region]) )
) ORDER BY [Count] DESC
```

Example using SUMMARIZECOLUMNS

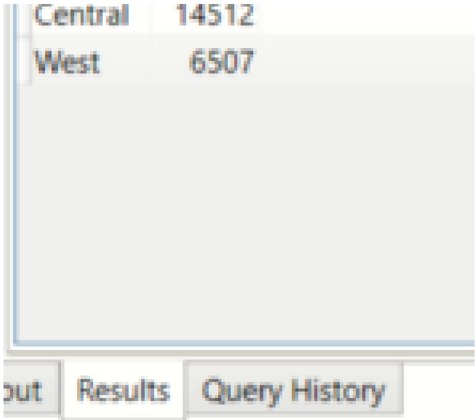
```
EVALUATE
(
        SUMMARIZECOLUMNS(bi_geo[Region], "Count",
COUNT(bi_geo[Region]) )
)
```

Example using GROUPBY

```
EVALUATE
(
        GROUPBY(
                bi_geo,bi_geo[Region],
                "Count", COUNTX(CURRENTGROUP() ,bi_geo[Region])
                )
)
```



Region	Count
East	18929



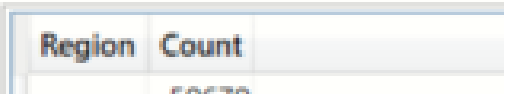
Central	14512
West	6507

What is the distribution of values in the Region columm of the bi_geo table taking into account the blank values? (GROUPBY, SELECTEDGROUP(),IF, ISBLANK)

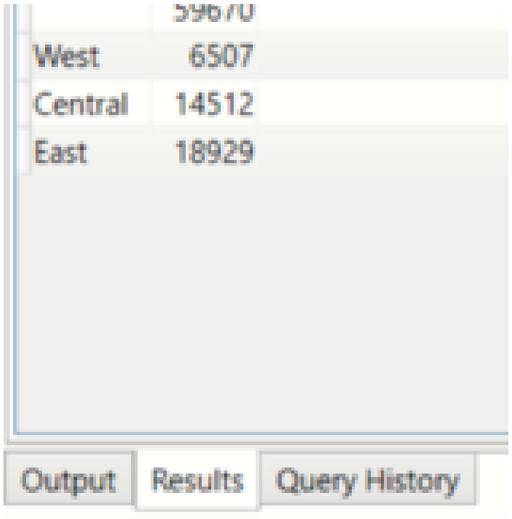
Approach 1

In this approach we are using `GROUPBY` and using `ISBLANK` and `IF` to convert the blank values into a non-blank value. Take note that the specified replacement value in the `IF` only helps in `GROUPBY` counting correctly

```
EVALUATE
(
        GROUPBY
        (
                bi_geo, bi_geo[Region], "Count",
                COUNTX
                        (
                        CURRENTGROUP(),
                        IF
                                (
                                        ISBLANK([Region]),
                                                "some non blank
value",
                                                [Region]
                                )
                        )

        )
)
```



Region	Count

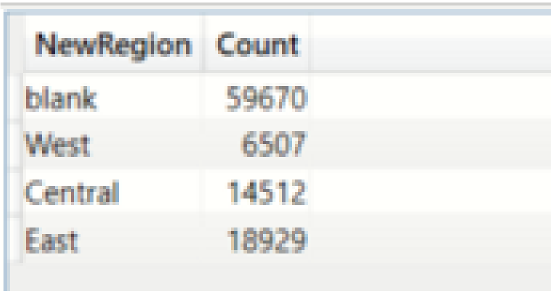


	59670
West	6507
Central	14512
East	18929

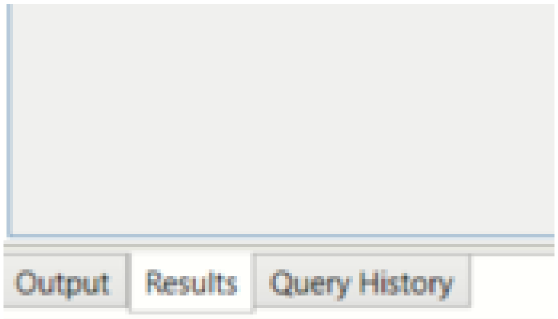
Approach 2

In this approach we are first creating a calculated table with a new column `NewRegion` where the blank value has been replaced by the string 'blank' and then using `SUMMARIZECOLUMNS` to do the grouping

```
DEFINE
TABLE allRegions=CALCULATETABLE(
        SELECTCOLUMNS
                (
                bi_geo,
                "NewRegion",
                IF(ISBLANK(bi_geo[Region]),"blank", bi_geo[Region])
                )
                )

EVALUATE
(

SUMMARIZECOLUMNS(allRegions[NewRegion],"Count",COUNT(allRegions[NewRe
gion]))
)
```



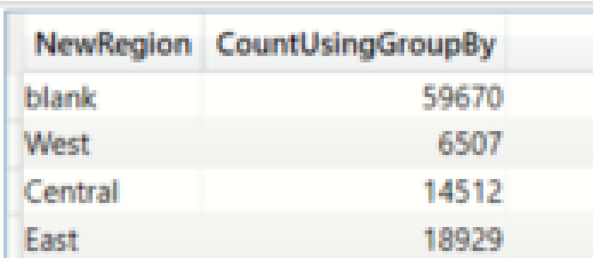
NewRegion	Count
blank	59670
West	6507
Central	14512
East	18929



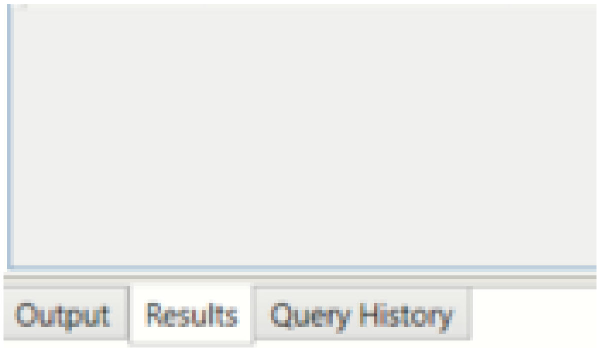
Approach 3

This is similar to the previous approach where we first created a calculated table using `CALCULATETABLE` and replaced the blank values with the string 'blank'. We are now using `GROUPBY` to do the grouping on the calculated table

```
DEFINE
TABLE allRegions=CALCULATETABLE(
        SELECTCOLUMNS
                (
                bi_geo,
                "NewRegion",
                IF(ISBLANK(bi_geo[Region]),"blank", bi_geo[Region])
                )
                )

EVALUATE
(
        GROUPBY
                (
                allRegions,
                allRegions[NewRegion],
                "CountUsingGroupBy",
                COUNTX(
                        CURRENTGROUP(),
                        allRegions[NewRegion])
                )
)
```



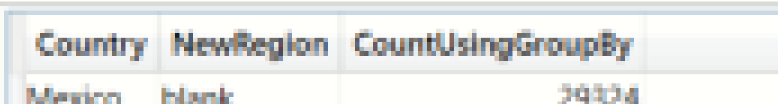
NewRegion	CountUsingGroupBy
blank	59670
West	6507
Central	14512
East	18929



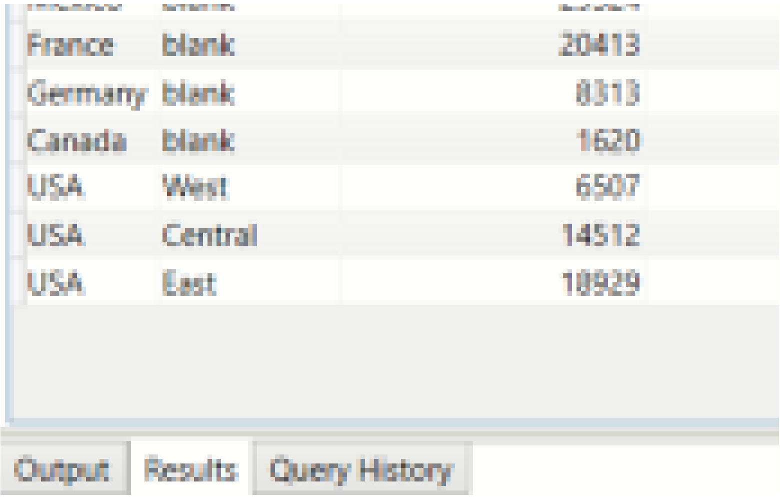
Approach 4

We are expanding on the previous approach of using `GROUPBY` and `CALCULATETABLE` and grouping by Country and Region

```
DEFINE
    TABLE allRegions =
        CALCULATETABLE (
            SELECTCOLUMNS (
                bi_geo,
                "Country", bi_geo[Country],
                "NewRegion",
                    IF (
                        ISBLANK ( bi_geo[Region] ),
                        "blank",
                        bi_geo[Region]
                    )
            )
        )
EVALUATE
(
    GROUPBY (
        allRegions,
        allRegions[Country],
        allRegions[NewRegion],
        "CountUsingGroupBy",
            COUNTX (
                CURRENTGROUP (),
                allRegions[NewRegion]
            )
    )
)
```



Country	NewRegion	CountUsingGroupBy
Mexico	blank	29324

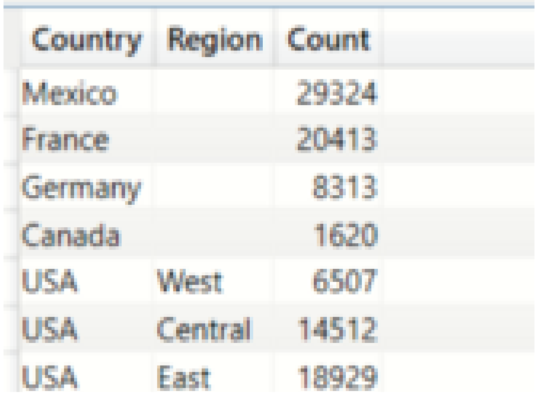


France	blank	20413
Germany	blank	8313
Canada	blank	1620
USA	West	6507
USA	Central	14512
USA	East	18929

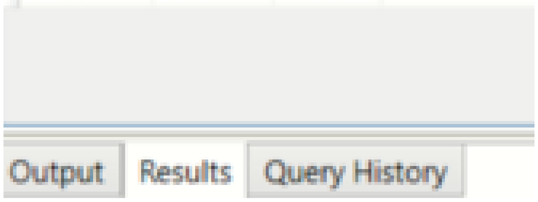
Approach 5

We could simply use `GROUPBY` and `IF`, `ISBLANK` to replace blank values with some string. Attention! `COUNTX` will refuse to count rows with blank values and therefore the `IF` clause is very important

```
EVALUATE
(
        GROUPBY(
                bi_geo,
                bi_geo[Country],bi_geo[Region],
                "Count",
                        COUNTX
                                (
                                        CURRENTGROUP(),
                                        IF(ISBLANK ( bi_geo[Region]
),"blank",bi_geo[Region])
                                )
                )
)
```



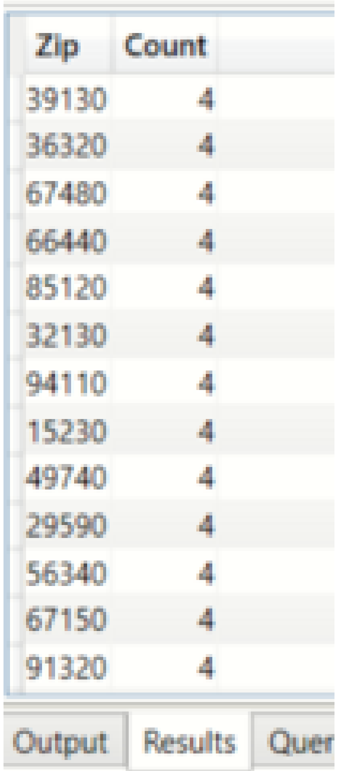
Country	Region	Count
Mexico		29324
France		20413
Germany		8313
Canada		1620
USA	West	6507
USA	Central	14512
USA	East	18929



Are there any duplicates in the 'zip' column of bi_Geo table? (SUMMARIZE,COUNT)

This will help us establish the cardinality of a foreign key relationship with the `zip` column. Looking at the results we can conclude that there are indeed duplicates and hence 1-many relationship between `bi_Geo` and `bi_SalesFact` is ruled out.

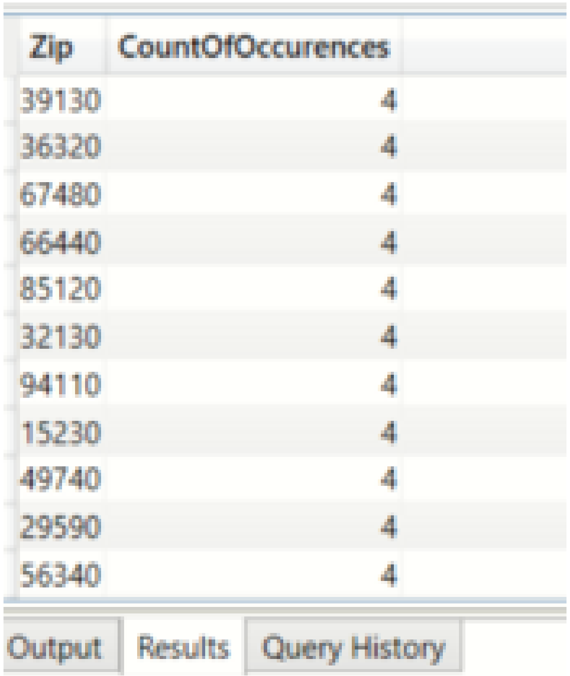
```
EVALUATE
(
SUMMARIZE(bi_geo,bi_geo[Zip], "Count", COUNT(bi_geo[Zip]) )
) ORDER BY [Count] DESC
```



Zip	Count
39130	4
36320	4
67480	4
66440	4
85120	4
32130	4
94110	4
15230	4
49740	4
29590	4
56340	4
67150	4
91320	4

The above can also be achieved by using `SUMMARIZECOLUMNS`

```
EVALUATE
(
                         SUMMARIZECOLUMNS
                         (
                                 bi_geo[Zip],

"CountOfOccurences",COUNT(bi_geo[Zip])
                         )
) ORDER BY [CountOfOccurences] DESC
```

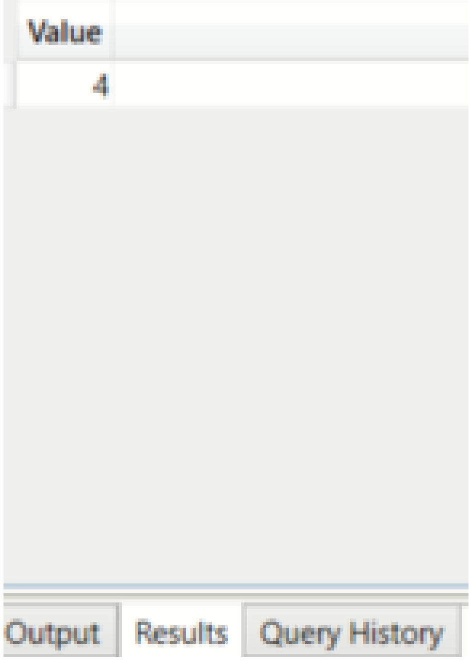


Zip	CountOfOccurences
39130	4
36320	4
67480	4
66440	4
85120	4
32130	4
94110	4
15230	4
49740	4
29590	4
56340	4

What is the highest number of times a single ‘zip’ code has been duplicated? (SUMMARIZECOLUMNS)

In this example `MAXX` and `SUMMARIZECOLUMNS` are used together to get the group with the highest count

```
EVALUATE
{
        MAXX
        (
                                SUMMARIZECOLUMNS
                                (
                                        bi_geo[Zip],

"CountOfOccurences",COUNT(bi_geo[Zip])
                                ),
                                [CountOfOccurences]

        )
}
```



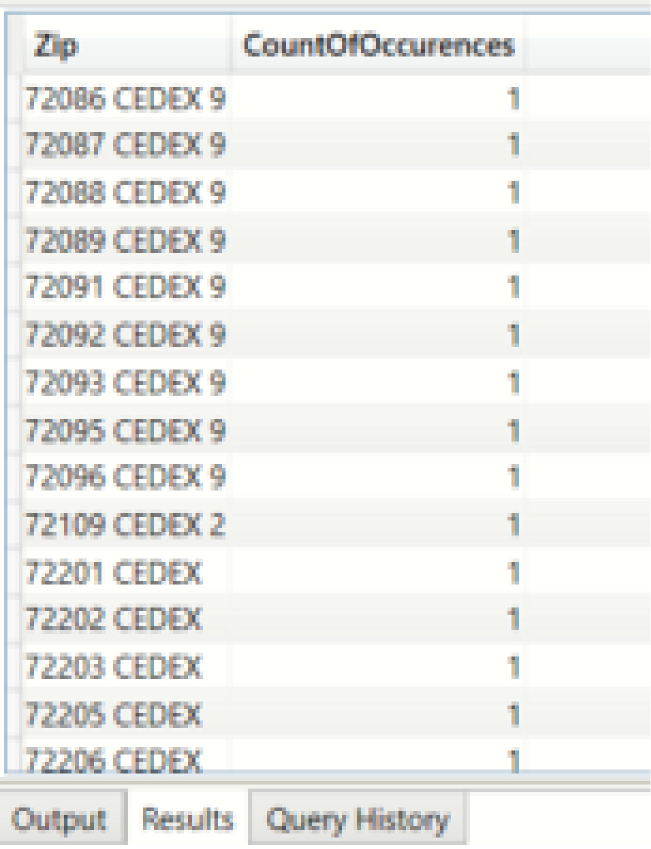
Value
4

How many values in the 'zip' column are not duplicated? (SUMMARIZECOLUMNS, FILTER, COUNT)

We will arrive at this result in 2 steps. We will first `SUMMARIZE` the row counts per group and then `FILTER` on this result to give us ony those rows where the row count is 1

Step 1: Use `FILTER` and `SUMMARIZECOLUMNS` to produce a flat table of all zip codes which are used only once

```
EVALUATE
(
          FILTER
          (
                    SUMMARIZECOLUMNS
                    (
                              bi_geo[Zip],
                              "CountOfOccurences",COUNT(bi_geo[Zip])
                    ),
                    [CountOfOccurences]=1
          )
)
```

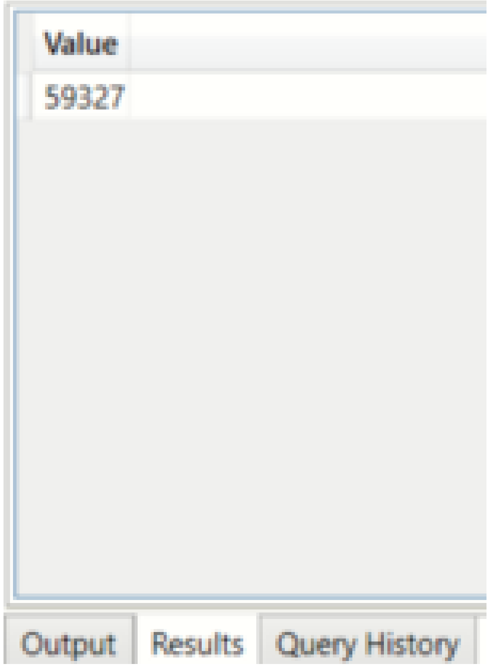


Zip	CountOfOccurences
72086 CEDEX 9	1
72087 CEDEX 9	1
72088 CEDEX 9	1
72089 CEDEX 9	1
72091 CEDEX 9	1
72092 CEDEX 9	1
72093 CEDEX 9	1
72095 CEDEX 9	1
72096 CEDEX 9	1
72109 CEDEX 2	1
72201 CEDEX	1
72202 CEDEX	1
72203 CEDEX	1
72205 CEDEX	1
72206 CEDEX	1

Step 2: Use `COUNTROWS` on the table produced in the previous step to get a scalar value

```
EVALUATE
{
          COUNTROWS
          (
                    FILTER
                    (
                              SUMMARIZECOLUMNS
```

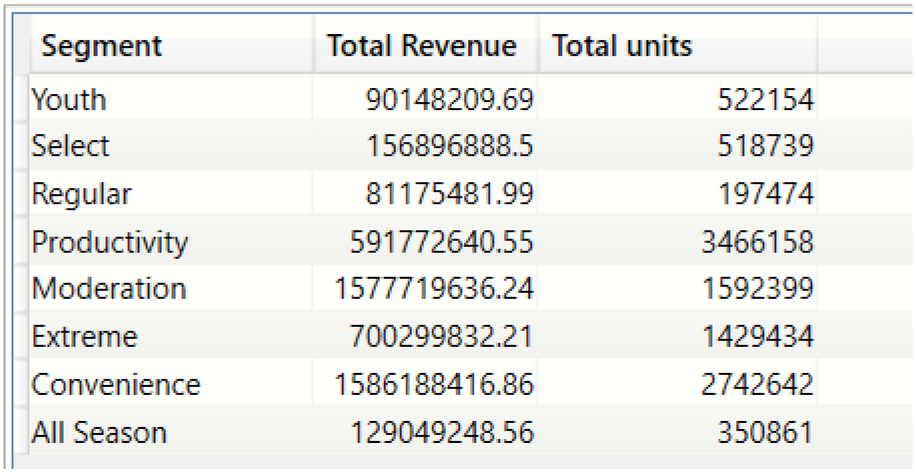
```
                                (
                                        bi_geo[Zip],

"CountOfOccurences",COUNT(bi_geo[Zip])
                                ),
                                [CountOfOccurences]=1
                        )
                )

}
```



```
Value
59327
```

Total units sold and total revenue earned per Product Segment (SUMMARIZE, SUM, ROUND)

```
EVALUATE
(
SUMMARIZE(
        bi_salesFact, bi_product[Segment] ,
        "Total Revenue",ROUND(SUM(bi_salesFact[Revenue]),2) ,
        "Total units", SUM(bi_salesFact[Units]) )

)
ORDER BY bi_product[Segment] DESC
```

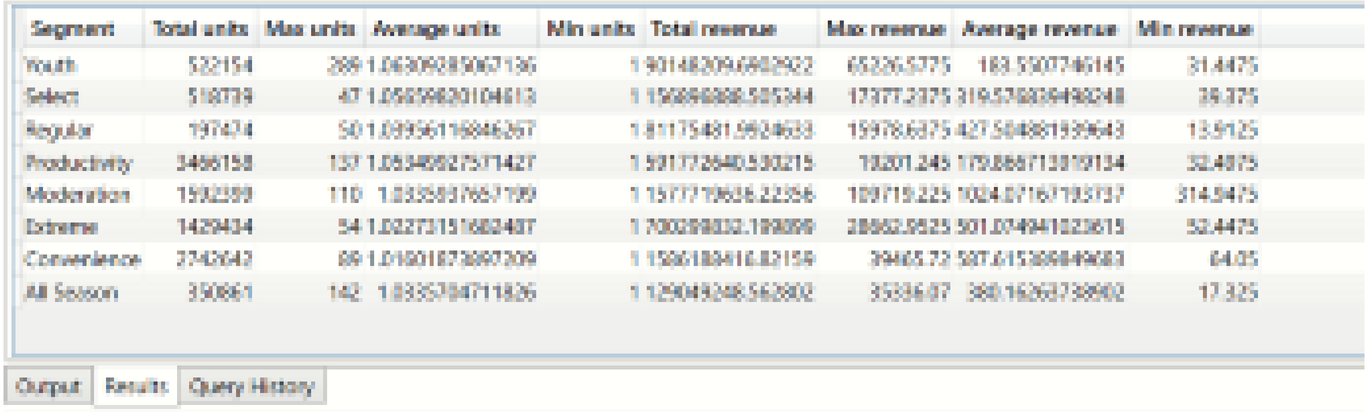


Segment	Total Revenue	Total units
Youth	90148209.69	522154
Select	156896888.5	518739
Regular	81175481.99	197474
Productivity	591772640.55	3466158
Moderation	1577719636.24	1592399
Extreme	700299832.21	1429434
Convenience	1586188416.86	2742642
All Season	129049248.56	350861

Min,Max,Avg sales per Product Segment (GROUPBY, SUMX,MINX, MAXX, AVERAGEX)

In this example we are calculating the statistics of sales in bi_SalesFact table on a per segment basis

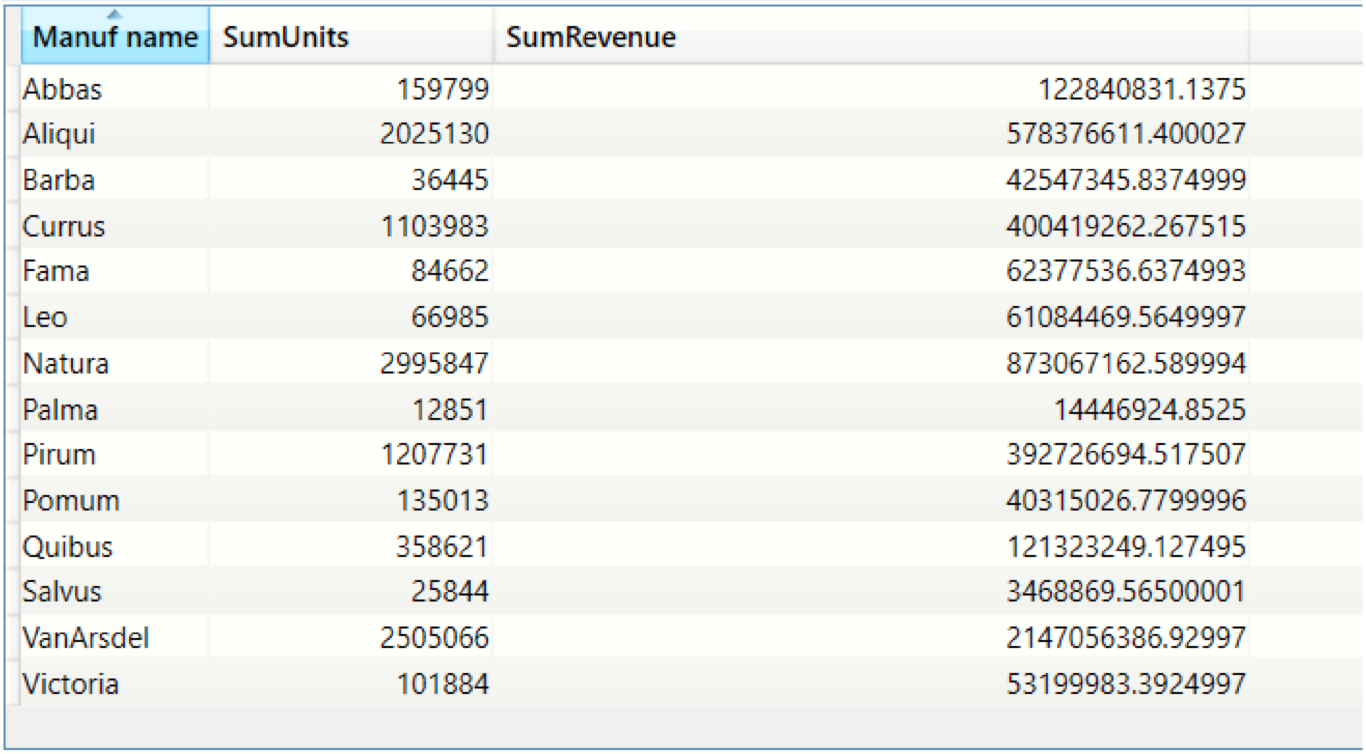
```
EVALUATE
 (
        GROUPBY
        (
                bi_salesFact,
                bi_product[Segment],
                "Total units",
(SUMX(CURRENTGROUP(),bi_salesFact[Units])),
                "Max units",
MAXX(CURRENTGROUP(),bi_salesFact[Units]),
                "Average units",
AVERAGEX(CURRENTGROUP(),bi_salesFact[Units]),
                "Min units",
MINX(CURRENTGROUP(),bi_salesFact[Units]),

                "Total revenue",
(SUMX(CURRENTGROUP(),bi_salesFact[Revenue])),
                "Max revenue",
MAXX(CURRENTGROUP(),bi_salesFact[Revenue]),
                "Average revenue",
AVERAGEX(CURRENTGROUP(),bi_salesFact[Revenue]),
                     "Min revenue",
MINX(CURRENTGROUP(),bi_salesFact[Revenue])

        )
 ) order by [segment] DESC
```



Segment	Total units	Max units	Average units	Min units	Total revenue	Max revenue	Average revenue	Min revenue
Youth	522154	289	1.06309285067136	1	90148209.6902922	65226.5775	183.5507746145	31.4475
Select	518739	47	1.05659820104613	1	158898888.505344	17377.2375	319.578839498248	39.375
Regular	197474	50	1.09956116846267	1	81175481.9924633	15978.6375	427.504881939643	13.9125
Productivity	3486158	137	1.05349827571427	1	591772640.590215	10201.245	179.866713019134	32.4875
Moderation	1592399	110	1.03358376657199	1	1577719636.22356	109719.225	1024.07167193737	314.9475
Extreme	1429434	54	1.02273151682487	1	700299832.199899	28862.9525	501.074941023615	52.4475
Convenience	2742642	89	1.01801873897209	1	1586189416.82159	39465.72	587.615389849683	64.05
All Season	350861	142	1.0335784711826	1	129049248.562802	35336.07	380.16263738902	17.325

Total units sold and revenue earned per Manufacturer (SELECTCOLUMNS)

```
EVALUATE
(
SELECTCOLUMNS
        (
        bi_manufacturer,
        "Manufacturer name", bi_manufacturer[Manufacturer] ,
        "SumUnits" , CALCULATE(SUM( bi_salesFact[Units])) ,
        "SumRevenue",CALCULATE(SUM( bi_salesFact[Revenue]))
        )
)
```

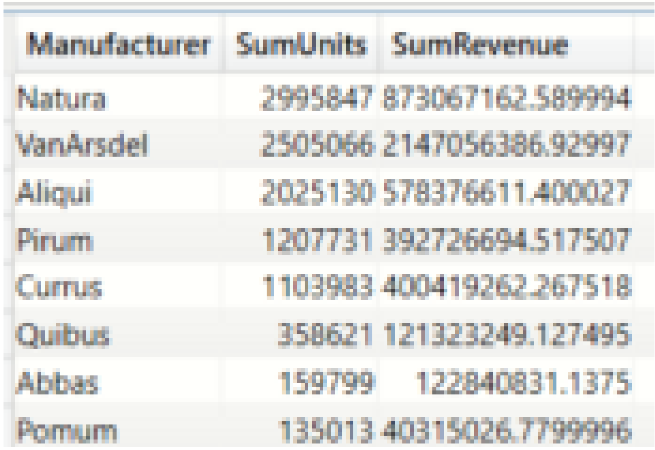


Manuf name	SumUnits	SumRevenue
Abbas	159799	122840831.1375
Aliqui	2025130	578376611.400027
Barba	36445	42547345.8374999
Currus	1103983	400419262.267515
Fama	84662	62377536.6374993
Leo	66985	61084469.5649997
Natura	2995847	873067162.589994
Palma	12851	14446924.8525
Pirum	1207731	392726694.517507
Pomum	135013	40315026.7799996
Quibus	358621	121323249.127495
Salvus	25844	3468869.56500001
VanArsdel	2505066	2147056386.92997
Victoria	101884	53199983.3924997

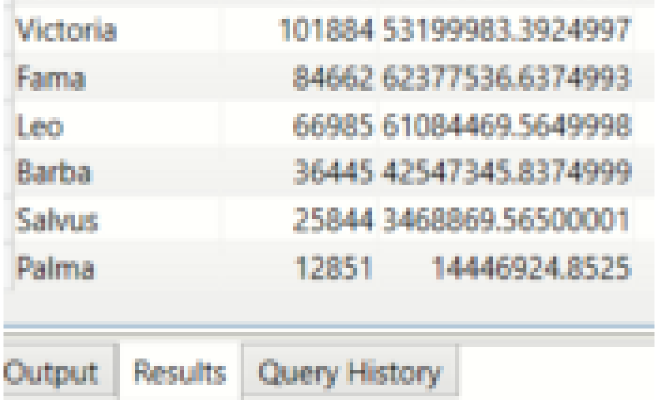
Total units sold and revenue earned per Manufacturer (SUMMARIZE)

We are using `SUMMARIZE` to produce the same result

```
EVALUATE
(
        SUMMARIZE(
                bi_manufacturer, bi_manufacturer[Manufacturer],
                "SumUnits" , CALCULATE(SUM( bi_salesFact[Units])),
                "SumRevenue",CALCULATE(SUM( bi_salesFact[Revenue]))
                )

) ORDER BY [SumUnits] DESC
```



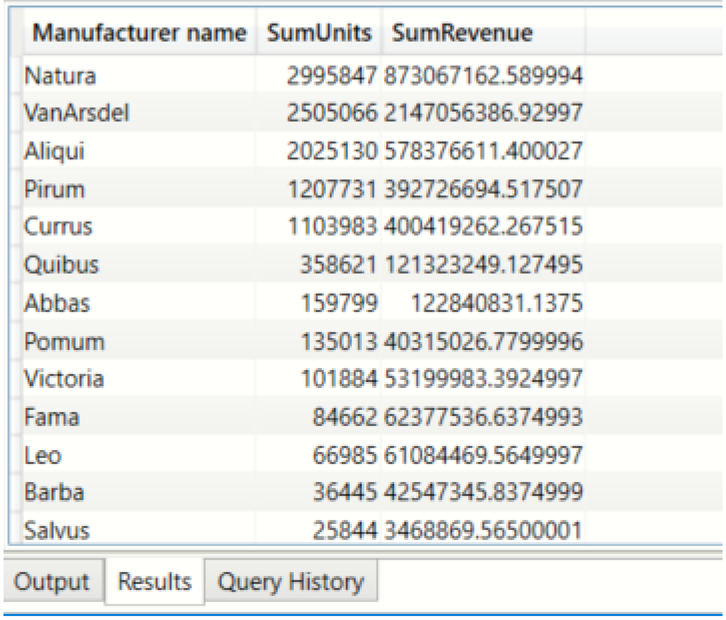
Manufacturer	SumUnits	SumRevenue
Natura	2995847	873067162.589994
VanArsdel	2505066	2147056386.92997
Aliqui	2025130	578376611.400027
Pirum	1207731	392726694.517507
Currus	1103983	400419262.267518
Quibus	358621	121323249.127495
Abbas	159799	122840831.1375
Pomum	135013	40315026.7799996



Victoria	101884	53199983.3924997
Fama	84662	62377536.6374993
Leo	66985	61084469.5649998
Barba	36445	42547345.8374999
Salvus	25844	3468869.56500001
Palma	12851	14446924.8525

Sort the manufacturers on Total units sold (SELECTCOLUMNS, ORDER BY)

```
EVALUATE
(
SELECTCOLUMNS
        (
        bi_manufacturer,
        "Manufacturer name", bi_manufacturer[Manufacturer] ,
        "SumUnits" , CALCULATE(SUM( bi_salesFact[Units])) ,
        "SumRevenue",CALCULATE(SUM( bi_salesFact[Revenue]))
        )
) ORDER BY  [SumUnits] DESC
```



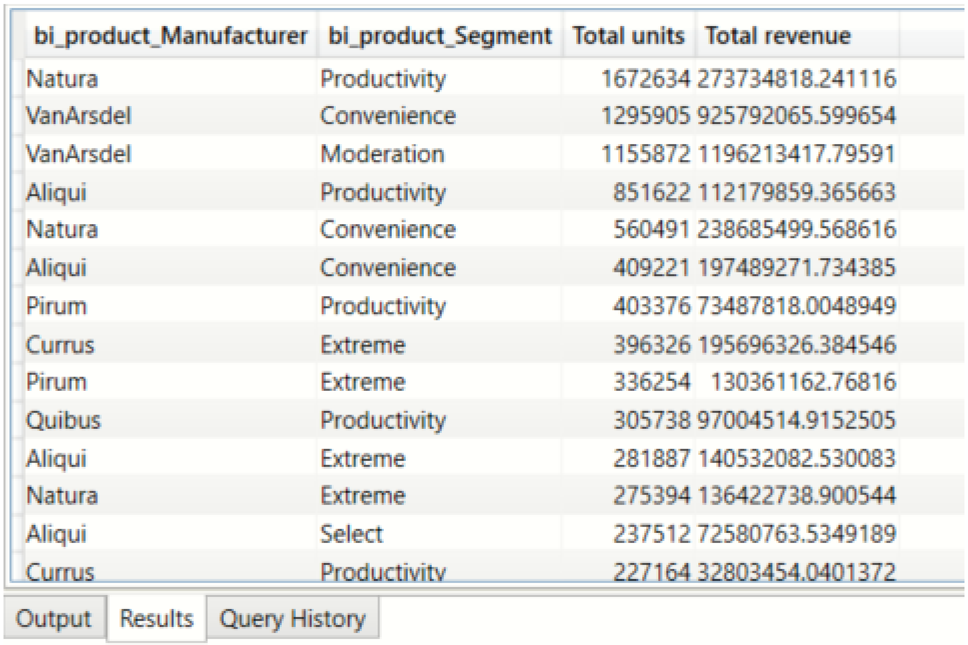
Manufacturer name	SumUnits	SumRevenue
Natura	2995847	873067162.589994
VanArsdel	2505066	2147056386.92997
Aliqui	2025130	578376611.400027
Pirum	1207731	392726694.517507
Currus	1103983	400419262.267515
Quibus	358621	121323249.127495
Abbas	159799	122840831.1375
Pomum	135013	40315026.7799996
Victoria	101884	53199983.3924997
Fama	84662	62377536.6374993
Leo	66985	61084469.5649997
Barba	36445	42547345.8374999
Salvus	25844	3468869.56500001

The above can also be achieved by using SUMMARIZE

```
EVALUATE
(
        SUMMARIZE(
                bi_manufacturer, bi_manufacturer[Manufacturer],
                "SumUnits" , CALCULATE(SUM( bi_salesFact[Units])),
                "SumRevenue",CALCULATE(SUM( bi_salesFact[Revenue]))
                )

) ORDER BY [SumUnits] DESC
```

Total units sold and revenue earned per Manufacturer per Segment

```
DEFINE
TABLE manuf_segment_totalunits = GROUPBY( bi_salesFact,
bi_product[Manufacturer], bi_product[Segment] , "Total units",SUMX(
CURRENTGROUP(), bi_salesFact[Units] ) ,"Total revenue",SUMX(
CURRENTGROUP(), bi_salesFact[Revenue] ) )
EVALUATE
(
manuf_segment_totalunits
)  order by [Total units] DESC
```

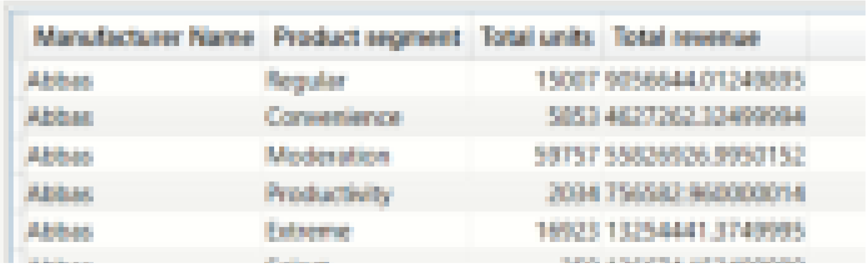


bi_product_Manufacturer	bi_product_Segment	Total units	Total revenue
Natura	Productivity	1672634	273734818.241116
VanArsdel	Convenience	1295905	925792065.599654
VanArsdel	Moderation	1155872	1196213417.79591
Aliqui	Productivity	851622	112179859.365663
Natura	Convenience	560491	238685499.568616
Aliqui	Convenience	409221	197489271.734385
Pirum	Productivity	403376	73487818.0048949
Currus	Extreme	396326	195696326.384546
Pirum	Extreme	336254	130361162.76816
Quibus	Productivity	305738	97004514.9152505
Aliqui	Extreme	281887	140532082.530083
Natura	Extreme	275394	136422738.900544
Aliqui	Select	237512	72580763.5349189
Currus	Productivity	227164	32803454.0401372

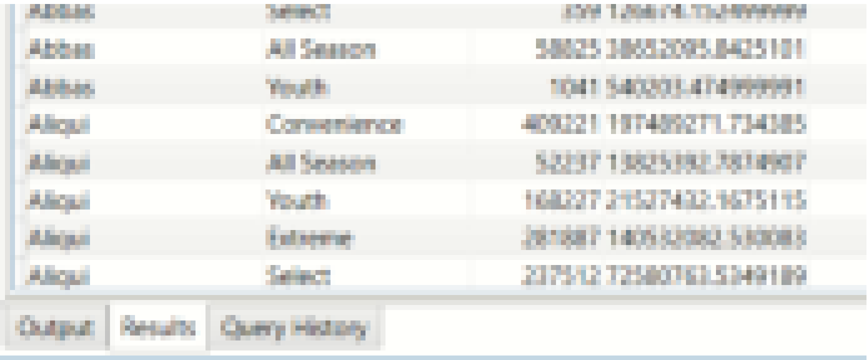
Total units sold and revenue earned per Manufacturer per Segment (renamed columns)

In this example we demonstrate how to rename the columns

```
DEFINE
TABLE manuf_segment_totalunits = GROUPBY( bi_salesFact,
bi_product[Manufacturer], bi_product[Segment] , "total_units",SUMX(
CURRENTGROUP(), bi_salesFact[Units] ) ,"total_revenue",SUMX(
CURRENTGROUP(), bi_salesFact[Revenue] ) )

EVALUATE
(
SELECTCOLUMNS(
        manuf_segment_totalunits ,
        "Manufacturer Name",[bi_product_Manufacturer],
        "Product segment",[bi_product_Segment],
        "Total units",[total_units],
        "Total revenue",[total_revenue]
        )
) ORDER BY [Manufacturer Name]
```



Manufacturer Name	Product segment	Total units	Total revenue
Abbas	Regular	15007	9056644.01249895
Abbas	Convenience	5853	4627262.32499994
Abbas	Moderation	59757	55826026.9950152
Abbas	Productivity	2034	756582.960000014
Abbas	Extreme	16923	13254441.3749995



Abbas	Select	359	126674.152499999
Abbas	All Season	58825	38652095.8425101
Abbas	Youth	1041	540203.474999991
Aliqui	Convenience	409221	197489271.734385
Aliqui	All Season	52237	13825392.7874907
Aliqui	Youth	168227	21527432.1675115
Aliqui	Extreme	281887	140532082.530083
Aliqui	Select	237512	72580763.5349189

Best selling and worst selling Product segment for every Manufacturer (SELECTEDVALUE,SUMMARIZE, MAXX, SUM, MINX)

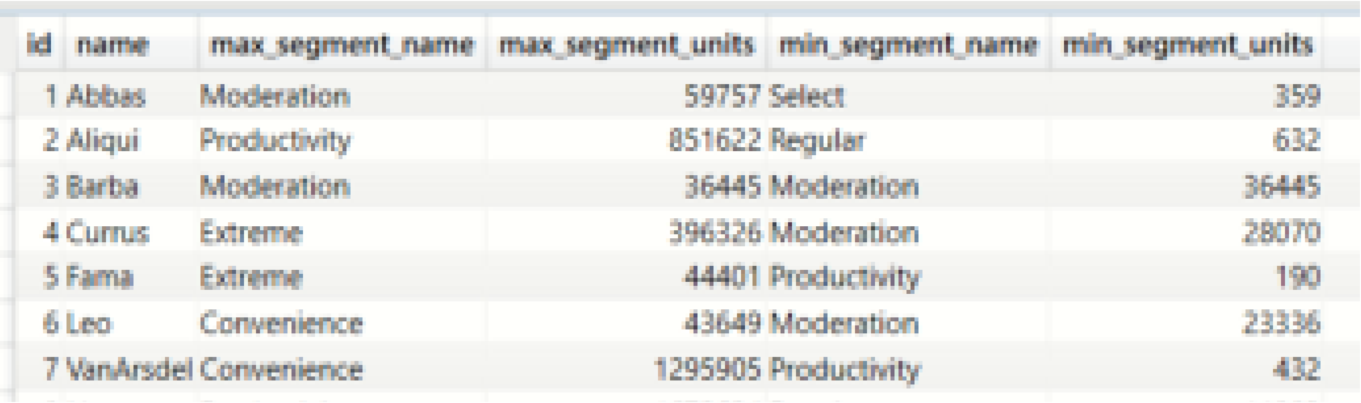
We will attempt to answer the question — "*For every manufacturer what was the best performing and worst performing product segment with regards to units sold*?" To achieve this we will create 4 measures

1. **segment_maxunits_name** Calculates the name of the product segment for a manufacturer which sold the highest number of units
2. **segment_maxunits_value** Calculates the total units sold by a manufacturer for the product segment calculated by the measure `segment_maxunits_name`
3. **segment_minunits_name** Calculates the name of the product segment for a manufacturer which sold the least number of units
4. **segment_minunits_value** Calculates the total units sold by a manufacturer for the product segment calculated by the measure `segment_minunits_name`

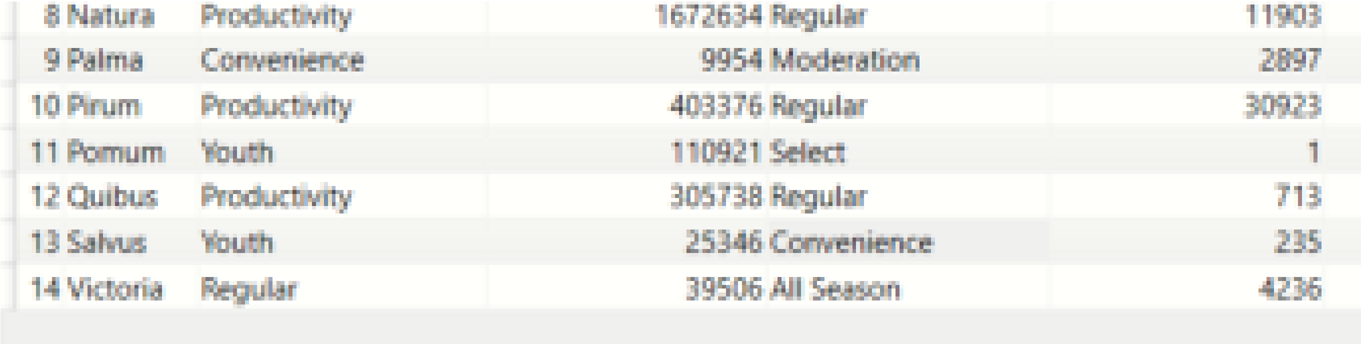
Step 1 Use the `SUMMARIZE` expression on the `bi_salesFact` and group by Segment. Use the `SELECTEDVALUE` to filter the records going into `SUMMARIZE` so that we are dealing with sales related to the current manufacturer only.
Step 2 Create measures on the `bi_manufacturer` which will pick the maximum and minimum from the output of **Step 1**
Step 3 Create measures which use the maximum and minimum values from **Step 2** to

filter the results of the `SUMMARIZE` operation in **Step 1** and we are now left with the rows which have the `segment` name.

```
DEFINE

MEASURE bi_manufacturer[segment_maxunits_value] =
//Get the max units sold by a segment

        VAR summary=
                        SUMMARIZE
                                (
                                FILTER (  bi_salesFact,
RELATED(bi_product[ManufacturerID] ) =
SELECTEDVALUE(bi_manufacturer[ManufacturerID])),
                                bi_product[Segment],
                                "TOTAL UNITS",
CALCULATE(SUM(bi_salesFact[Units]))
                                )
        VAR maxUnit=MAXX(summary,[TOTAL UNITS])
        VAR     x=SELECTEDVALUE(bi_product[Category])
        RETURN maxUnit

MEASURE bi_manufacturer[segment_maxunits_name] =
//Now get the segment name which sold the max units
        VAR summary=
                        SUMMARIZE
                                (
                                FILTER (  bi_salesFact,
RELATED(bi_product[ManufacturerID] ) =
SELECTEDVALUE(bi_manufacturer[ManufacturerID])),
                                bi_product[Segment],
                                "TOTAL UNITS",
CALCULATE(SUM(bi_salesFact[Units]))
                                )
        VAR maxUnitValue=MAXX(summary,[TOTAL UNITS])
        var maxSegmentName = CALCULATE( MAXX(FILTER( summary, [TOTAL
UNITS]=maxUnitValue),[Segment]))

        RETURN maxSegmentName

MEASURE bi_manufacturer[segment_minunits_value] =
//Get the min units sold by a segment

        VAR summary=
                        SUMMARIZE
                                (
                                FILTER (  bi_salesFact,
RELATED(bi_product[ManufacturerID] ) =
SELECTEDVALUE(bi_manufacturer[ManufacturerID])),
                                 bi_product[Segment],
                                 "TOTAL UNITS",
CALCULATE(SUM(bi_salesFact[Units]))
                                 )
        VAR minUnit=MINX(summary,[TOTAL UNITS])
        RETURN minUnit

MEASURE bi_manufacturer[segment_minunits_name] =
//Now get the segment name which sold the min units
        VAR summary=
                        SUMMARIZE
                                 (
                                 FILTER (  bi_salesFact,
RELATED(bi_product[ManufacturerID] ) =
SELECTEDVALUE(bi_manufacturer[ManufacturerID])),
                                 bi_product[Segment],
                                 "TOTAL UNITS",
CALCULATE(SUM(bi_salesFact[Units]))
                                 )
        VAR minUnit=MINX(summary,[TOTAL UNITS])
        var minSegmentName = CALCULATE( MAXX(FILTER( summary, [TOTAL
UNITS]=minUnit),[Segment]))

        RETURN minSegmentName

EVALUATE
(
        SELECTCOLUMNS(
                bi_manufacturer,
                "id",[ManufacturerID],
                "name",[Manufacturer],
                "max_segment_name",      [segment_maxunits_name],
                "max_segment_units",[segment_maxunits_value],
                "min_segment_name",      [segment_minunits_name],
                "min_segment_units",[segment_minunits_value]
                )
)
```



id	name	max_segment_name	max_segment_units	min_segment_name	min_segment_units
1	Abbas	Moderation	59757	Select	359
2	Aliqui	Productivity	851622	Regular	632
3	Barba	Moderation	36445	Moderation	36445
4	Currus	Extreme	396326	Moderation	28070
5	Fama	Extreme	44401	Productivity	190
6	Leo	Convenience	43649	Moderation	23336
7	VanArsdel	Convenience	1295905	Productivity	432



8	Natura	Productivity	1672634	Regular	11903
9	Palma	Convenience	9954	Moderation	2897
10	Pirum	Productivity	403376	Regular	30923
11	Pomum	Youth	110921	Select	1
12	Quibus	Productivity	305738	Regular	713
13	Salvus	Youth	25346	Convenience	235
14	Victoria	Regular	39506	All Season	4236

Conclusion

I would be delighted to hear from you. Did you spot any mistakes? Did I miss anything obvious? Your feedback would be very beneficial for my future work. Thank you.