DAX Deep Dive

Venkata Reddy Konasani

Contents

- Advanced DAX functions
- Calculate function
- Time intelligence functions
- Cumulative values
- Using filters
- Using VAR
- Previous period calculations

Product Sales Case study

Step5: Creating Measures and calculated fields

Excel Formula vs. DAX formula

- What is the difference between Excel formula vs DAX measure ?
- Excel formulas work on every cell in the data. DAX measures work on the full column or full data

What is the quantity sold by each ear ?

Year

2018
2019
2020
2021

New measure – Quantity sold

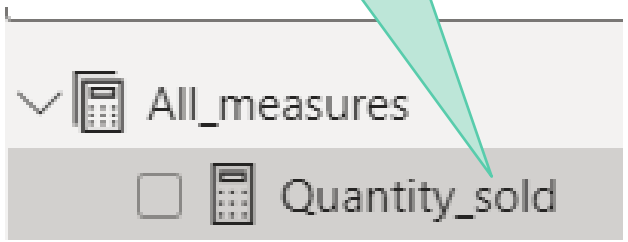
```
Quantity_sold = sum(Sales[Quantity])
```

Year	Quantity_sold
2018	9650
2019	9528
2020	1900
Total	**21078**

New measure – Quantity sold

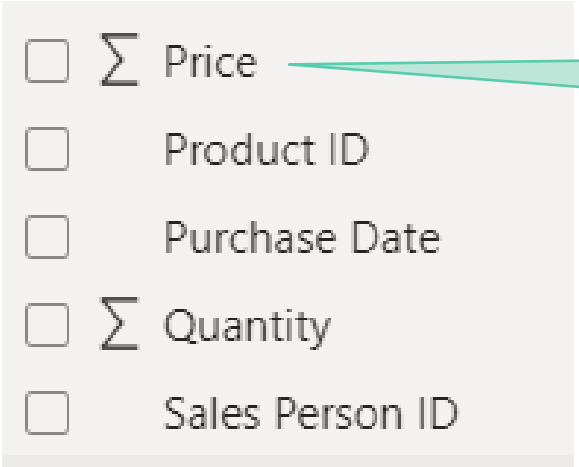
```
1  Quantity_sold = sum(Sales[Quantity])
```

Create a new measure and keep them in all measures table
Note - Create all measures table



What is the revenue(total sales amount) per year

- Create a new measure Total_Sales = Quantity * Price



Multiply price and quantity

Create a new measure Total_Sales = Quantity * Price

```
1  Total_Sales = SUM(Sales[Quantity]*Sales[Price])
```

```
1  Total_Sales = SUM(Sales[Quantity]*Sales[Price])
```

! The SUM function only accepts a column reference as an argument.

- This formula doesn't work. It throws an error.
- Measures work on the whole table at an aggerated level.
- We can either have sum of quantity or sum of price.

Create a new measure Total_Sales = Quantity * Price

```
1  Total_Sales = SUM(Sales[Quantity]*Sales[Price])
```

```
1  Total_Sales = SUM(Sales[Quantity]*Sales[Price])
```

! The SUM function only accepts a column reference as an argument.

- Instead of overall aggregation, If you want to have a calculation at every row level or any category level, then we nee to use iterator functions

Iterator functions

```
Total_Sales = SUMX(Sales, Sales[Quantity]*Sales[Price])
```

We need to mention the table name and the expression

- SUMX is an iterator function.
- It works at the row level

Iterator functions

Order ID	Sum of Price	Sum of Quantity
AX10001	136	1
AX10002	596	1
AX10004	1848	4
AX10005	1899	1
AX10007	951	4
AX10008	1522	2
AX10009	1103	1
AX10011	1278	3
AX10013	1022	1
AX10014	1725	1
AX10015	880	2
AX10016	1052	1
AX10017	603	1
AX10018	684	1
AX10020	1582	2
AX10022	530	2
AX10023	684	4
AX10024	1494	3
AX10025	290	1
AX10027	2464	1
AX10030	2283	1
AX10032	2141	1
AX10034	1367	1

Order ID	Total_Sales
AX10001	136
AX10002	596
AX10004	7392
AX10005	1899
AX10007	3804
AX10008	3044
AX10009	1103
AX10011	3834
AX10013	1022
AX10014	1725
AX10015	1760
AX10016	1052
AX10017	603
AX10018	684
AX10020	3164
AX10022	1060
AX10023	2736
AX10024	4482
AX10025	290
AX10027	2464
AX10030	2283
AX10032	2141
AX10034	1367

- SUMX is an iterator function.
- It works at the row level

Iterator functions

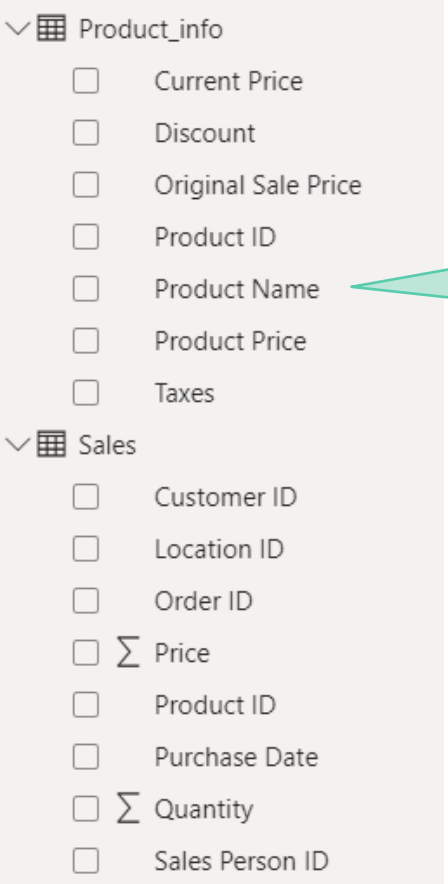
```
Total_Sales = SUMX(Sales, Sales[Quantity]*Sales[Price])
```

Year	Quantity_sold	Total_Sales
2018	9650	11690763
2019	9528	11565545
2020	1900	2404901
Total	**21078**	**25661209**

- In every row Quantity*price is calculated and the overall sum is printed here

What is the profit per year?

- Profit in an order = (Product Sale Price - Product price)* Quantity



- Product Price and Product sale price are in different tables
- We have to use the function RELATED in this scenario

Related function

- `Profit = SUMX(Sales, (Sales[Price]-RELATED(Product_info[Product Price]))*Sales[Quantity])`

Year	Quantity_sold	Total_Sales	Profit
2018	9650	11690763	3805480
2019	9528	11565545	3756605
2020	1900	2404901	781808
Total	**21078**	**25661209**	**8343893**

- Related is a frequently used useful measure

Using the measures inside measures

- Profit in an order = (Product Sale Price - Product price)* Quantity
- We can try a different formula which will give the same result

- Profit in an order = Product Sale Price*Quantity - Product price* Quantity
- Profit in an order = Total Sales - Product price* Quantity

Using the measures inside measures

```
Profit1 = SUMX(Sales, [Total_Sales]-(RELATED(Product_info[Product Price])*Sales[Quantity]))
```

- Total sales is an existing measure

Year	Quantity_sold	Total_Sales	Profit	Profit1
2018	9650	11690763	3805480	3805480
2019	9528	11565545	3756605	3756605
2020	1900	2404901	781808	781808
Total	**21078**	**25661209**	**8343893**	**8343893**

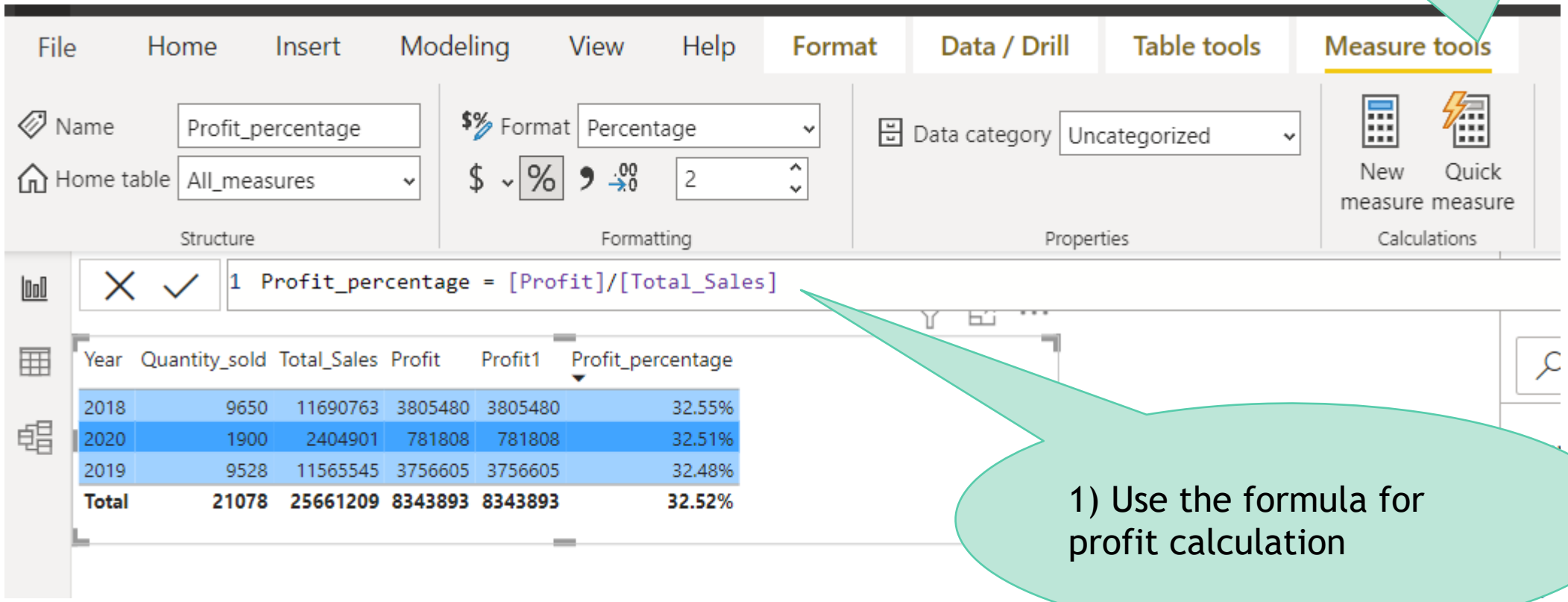
- Both are resulting the same values

Find the percentage of profit

- Find the percentage of profit
- Profit/overall sales

Find the percentage of profit

2) Use measure tools to format it as percentage



Year	Quantity_sold	Total_Sales	Profit	Profit1	Profit_percentage
2018	9650	11690763	3805480	3805480	32.55%
2020	1900	2404901	781808	781808	32.51%
2019	9528	11565545	3756605	3756605	32.48%
Total	**21078**	**25661209**	**8343893**	**8343893**	**32.52%**

1) Use the formula for profit calculation

Show the last year sales

- Display last year sales

Calculate() function

- We can use calculate function to remove the context for total sales.
- From the current context, if we want to refer to a different context.
- When the row context is this year, how do you show the previous year sales?

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

Year	Quantity_sold	Total_Sales	Last_year_sales
2018	9650	11690763	
2019	9528	11565545	11690763
2020	1900	2404901	11565545
2021			2404901
Total	**21078**	**25661209**	**25661209**

Show the last year sales

```
Last_year_sales = CALCULATE([Total_Sales], SAMEPERIODLASTYEAR(Dates[Date]))
```

Year	Quantity_sold	Total_Sales	Last_year_sales
2018	9650	11690763	
2019	9528	11565545	11690763
2020	1900	2404901	11565545
2021			2404901
Total	**21078**	**25661209**	**25661209**

- Use SAMEPERIODLASTYEAR() function to fetch the last year values.

Calculate the sales till this date

- Calculate Cumulative sales till date

Calculate the sales till this date

Structure | Formatting | Properties

```
Cumulative_Sales = CALCULATE([Total_Sales], Dates[Date]<=MAX(Dates[Date]))
```

Year	Quantity_sold	Total_Sales	Last_year_sales	Cumulative_Sales
2018	9650	11690763		11690763
2019	9528	11565545	11690763	23256308
2020	1900	2404901	11565545	25661209
2021			2404901	25661209
Total	**21078**	**25661209**	**25661209**	**25661209**

- Take date until now and compare it with the MAX date

Cumulative Profit till date

- Calculate cumulative profit till date

Cumulative Profit till date

```
Profit_tilldate = CALCULATE([Profit], Dates[Date]<=MAX(Dates[Date]))
```

Year	Quantity_sold	Total_Sales	Last_year_sales	Cumulative_Sales	Profit	Profit_tilldate
2018	9650	11690763		11663813	3805480	3805480
2019	9528	11565545	11690763	23245667	3756605	7562085
2020	1900	2404901	11565545	25661209	781808	8343893
2021			2404901	25661209		8343893
Total	**21078**	**25661209**	**25661209**	**25661209**	**8343893**	**8343893**

- Less than or equal to

Focus on a particular category

- Display sales from “Los Angeles County” without using a slicer

Focus on a particular category

- Display sales from “Los Angeles County” without using a slicer

```
LA_County_Sales = CALCULATE([Total_Sales], Geographical_Data[County]="Los Angeles County")
```

Year	Total_Sales	LA_County_Sales
2018	11690763	2589676
2019	11565545	2510357
2020	2404901	530170
Total	**25661209**	**5630203**

Focus on a Set of categories1

- Display sales from “Los Angeles County” only for Product 1

Focus on a Set of categories1

- Display sales from “Los Angeles County” only for Product 1

```
LA_and_Prod1 = CALCULATE([Total_Sales],
                         Geographical_Data[County]="Los Angeles County" ,
                         Product_info[Product Name] = "Product 1")
```

Year	Total_Sales	LA_County_Sales	LA_and_Prod1
2018	11690763	2589676	20169
2019	11565545	2510357	24651
2020	2404901	530170	13446
Total	**25661209**	**5630203**	**58266**

- Use Shift+enter for the next line

Focus on a Set of categories2

- Display sales from “Los Angeles County” only for Product 1 , Product 2, Product 3

Focus on a Set of categories2

- Display sales from “Los Angeles County” only for Product 1 , Product 2, Product 3

```
LA_and_Prod1_2_3 = CALCULATE([Total_Sales],
                             Geographical_Data[County]="Los Angeles County" ,
                             Product_info[Product Name] IN {"Product 1", "Product 2", "Product 3" } )
```

Year	Total_Sales	LA_County_Sales	LA_and_Prod1	LA_and_Prod1_2_3
2018	11690763	2589676	20169	54643
2019	11565545	2510357	24651	48456
2020	2404901	530170	13446	26463
Total	**25661209**	**5630203**	**58266**	**129562**

- IN operator syntax

Using VAR in the DAX formula

- Defining VARs
- Using FILTER function

```
LA_Prod_new =
VAR Filter1 =FILTER(Geographical_Data, Geographical_Data[County]="Los Angeles County")
VAR Filter2 = FILTER(Product_info, Product_info[Product Name] IN {"Product 1", "Product 2", "Product 3" } )
RETURN
CALCULATE([Total_Sales], Filter1, Filter2)

// Measure to calcuate the Prod1, 2, 3 sales in LA County
```

- Using the vars in the calculation
- We can also include comments by keeping //

Year	Total_Sales	LA_County_Sales	LA_and_Prod1	LA_and_Prod1_2_3	LA_Prod_new
2018	11690763	2589676	20169	54643	54643
2019	11565545	2510357	24651	48456	48456
2020	2404901	530170	13446	26463	26463
Total	**25661209**	**5630203**	**58266**	**129562**	**129562**

- This is the industry standard of wring the complex DAX formulas.

Using VAR in the DAX formula

- Display sales for the customers with Age <40 in cities with population more than 200,000 and product current price more than 2000

Using VAR in the DAX formula

- Display sales for the customers with Age <40 in cities with population more than 200,000 and product current price more than 2000.

```
Sales_3Conditions =
VAR Filter1 =FILTER(Customer_info, Customer_info[Age]<40)
VAR Filter2 = FILTER(Geographical_Data, Geographical_Data[Population]>200000)
VAR Filter3 = FILTER(Product_info, Product_info[Current Price]>2000)
RETURN

CALCULATE([Total_Sales], Filter1, Filter2, Filter3)

// Measure to calcuate Sales under three conditions
```

Year	Total_Sales	Sales_3Conditions
2018	11690763	504397
2019	11565545	554334
2020	2404901	125354
Total	**25661209**	**1184085**

Running totals

- Calculate Last 2 days sales
- Calculate Last 7 days sales
- Calculate Last 30 days sales
- Calculate Last 90 days sales

Calculate Last 2 days sales

```
Last2Days_Sales = CALCULATE([Total_Sales],
                            Dates[Date]>=MAX(Dates[Date])-2,
                            Dates[Date]<MAX(Dates[Date]))
```

Date	Total_Sales	Last2Days_Sales
01 January 2018	60777	
02 January 2018	39297	60777
03 January 2018	7861	100074
04 January 2018	28193	47158
05 January 2018	25968	36054
06 January 2018	27311	54161
07 January 2018	37426	53279
08 January 2018	36614	64737
09 January 2018	28954	74040
10 January 2018	6947	65568
11 January 2018	19572	35901
12 January 2018	54444	26519
13 January 2018	6894	74016
14 January 2018	22537	61338
15 January 2018	43734	29431
16 January 2018	40652	66271
Total	**25661209**	

- Date must be greater than two days back and less than today.

Calculate Last 7 days sales

```
Last7Days_Sales = CALCULATE([Total_Sales],
                            Dates[Date]>=MAX(Dates[Date])-7,
                            Dates[Date]<MAX(Dates[Date]))
```

Date	Total_Sales	Last2Days_Sales	Last7Days_Sales
01 January 2018	60777		
02 January 2018	39297	60777	60777
03 January 2018	7861	100074	100074
04 January 2018	28193	47158	107935
05 January 2018	25968	36054	136128
06 January 2018	27311	54161	162096
07 January 2018	37426	53279	189407
08 January 2018	36614	64737	226833
09 January 2018	28954	74040	202670
10 January 2018	6947	65568	192327
11 January 2018	19572	35901	191413
12 January 2018	54444	26519	182792
13 January 2018	6894	74016	211268
14 January 2018	22537	61338	190851
15 January 2018	43734	29431	175962
Total	**25661209**		

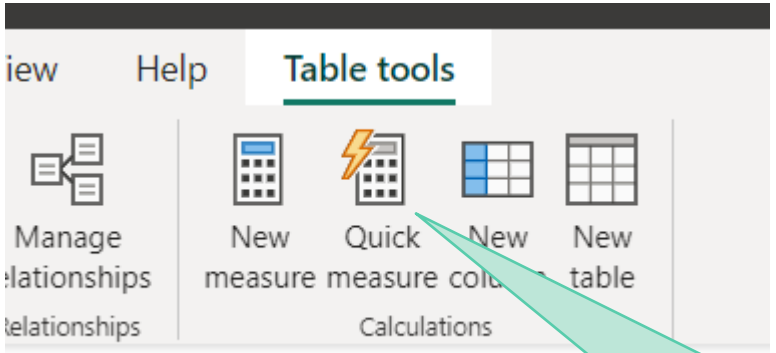
- Similar calculations for last 30 days and 90 days.

Create a new table

- Take the all the rows from the sales table with quantity > 3. Create a new table with this data.

Create a new table

- Take the all the rows from the sales table with quantity > 3. Create a new table with this data.




```
Sales_Subset = FILTER(Sales, Sales[Quantity]>3)
```

Order ID	Product ID	Location ID	Sales Person ID	Customer ID	Purchase Date	Quantity	Price
AX10056	ENX2040	A118	EMP1007	C1160	09-10-2019 00:00:00	4	1889
AX10097	ENX2024	A167	EMP1007	C1396	30-01-2019 00:00:00	4	356
AX10110	ENX2066	A107	EMP1013	C1342	10-11-2019 00:00:00	4	2091
AX10152	ENX2027	A146	EMP1043	C1314	21-01-2019 00:00:00	4	2497
AX10195	ENX2085	A156	EMP1025	C1694	17-10-2019 00:00:00	4	1862
AX10252	ENX2091	A173	EMP1031	C1602	02-02-2019 00:00:00	4	368
AX10258	ENX2043	A168	EMP1009	C1273	13-09-2019 00:00:00	4	545
AX10269	ENX2095	A136	EMP1012	C1506	15-10-2019 00:00:00	4	1743
AX10327	ENX2050	A112	EMP1039	C1773	16-09-2019 00:00:00	4	2391
AX10357	ENX2074	A154	EMP1043	C1798	26-01-2019 00:00:00	4	504
AX10376	ENX2024	A128	EMP1002	C1426	15-01-2019 00:00:00	4	356
	X2077	A158	EMP1009	C1103	26-09-2019 00:00:00	4	530
		A145	EMP1035	C1610	25-01-2019 00:00:00	4	203
		19	EMP1013	C1648	27-04-2019 00:00:00	4	947
			EMP1034	C1562	25-04-2019 00:00:00	4	561
			EMP1002	C1455	09-01-2019 00:00:00	4	1380
			EMP1027	C1625	22-09-2019 00:00:00	4	561
			EMP1001	C1720	30-07-2019 00:00:00	4	2005
			EMP1038	C1092	20-12-2019 00:00:00	4	1338
		A153	EMP1043	C1559	21-02-2019 00:00:00	4	1582

Click on Table >>Table tools >>new table

Previous time period

- Display monthly sales, also add a new column last month sales

Previous time period

- Display monthly sales, also display a column with last month sales

```
Last_Month_Sales = CALCULATE([Total_Sales], DATEADD(Dates[Date],-1,MONTH))
```

Year	Month	Total_Sales	Last_Month_Sales
2018	January	986347	
2018	February	889042	986347
2018	March	1136226	889042
2018	April	893923	1136226
2018	May	969286	893923
2018	June	986392	969286
2018	July	988343	986392
2018	August	1015583	988343
2018	September	907439	1015583

Need a DATEADD() function

Product Sales Case study

Step6: Creating a Dashboard