Advanced DAX - Iterative Functions

SUM vs SUMX

```
Sales Amount (SUM) =
 SUM('Sales by Store'[quantity_sold]) * SUM('Sales by Store'[unit_price])
```

```
Sales Amount (SUMX) =
SUMX(
        'Sales by Store',
        'Sales by Store'[quantity_sold]*'Sales by Store'[unit_price]
)
```

Product Category	Sum of quantity_sold	Sum of unit_price	Sales Amount (SUM)	Sales Amount (SUMX)
Bakery	141433	492,566.67	$69,665,181,838.11	$501,291.32
Branded	4697	81,118.00	$381,011,246.00	$83,784.00
Coffee	545936	1,081,306.45	$590,324,118,087.20	$1,651,861.15
Coffee beans	11095	225,231.95	$2,498,948,485.25	$240,414.95
Drinking Chocolate	106067	288,381.75	$30,587,787,077.25	$439,942.50
Flavours	63825	33,035.20	$2,108,471,640.00	$51,060.00
Loose Tea	7381	68,451.90	$505,243,473.90	$68,451.90
Packaged Chocolate	2815	25,584.16	$72,019,410.40	$25,584.16
Tea	422388	776,435.60	$327,957,080,212.79	$1,190,314.90
Total	**1305637**	**3,072,111.68**	**$4,011,062,677,540.15**	**$4,252,704.88**

Note: As observed, the sales amount in the third column (Sales Amount (SUM)) appears incorrect. This issue arises because the RANK function is directly computing the product of the total sum of quantity and unit price. Instead, we need to calculate the product for each individual line item and then sum these products. In this scenario, the SUMX function is the appropriate solution.

AVERAGEX

Average Daily Sales

```
Average Daily Sales (AVERAGEX) =
AVERAGEX(
            'Calendar',
            [Customer Sales]
)
```

Year	Sum of quantity_sold	Sum of unit_price	Sales Amount (SUMX)	Average Daily Sales (AVERAGEX)
⊟ 2017	514961	1,211,922.26	$1,678,074.11	$4,597.46
⊟ Qtr 1	78977	186,473.61	$257,273.51	$2,858.59
⊞ January	24907	59,159.89	$81,845.09	$2,640.16
⊞ February	23586	55,236.79	$76,273.99	$2,724.07
⊞ March	30484	72,076.93	$99,154.43	$3,198.53
⊟ Qtr 2	135978	319,290.13	$443,417.78	$4,872.72
⊞ April	36559	85,983.16	$119,309.01	$3,976.97
⊞ May	48360	113,439.64	$157,208.99	$5,071.26
⊞ June	51059	119,867.33	$166,899.78	$5,563.33
⊟ Qtr 3	140726	330,923.51	$458,275.06	$4,981.25
⊞ July	48404	113,633.65	$157,968.55	$5,095.76
⊞ August	47416	111,712.82	$154,485.32	$4,983.40
⊞ September	44906	105,577.04	$145,821.19	$4,860.71
⊟ Qtr 4	159280	375,235.01	$519,107.76	$5,642.48
⊞ October	51880	122,063.19	$169,223.54	$5,458.82
⊞ November	55086	129,875.65	$179,999.30	$5,999.98
⊞ December	52314	123,296.17	$169,884.92	$5,480.16
⊟ 2018	588694	1,384,343.75	$1,916,544.75	$5,250.81
Total	1305637	3,072,111.68	$4,252,704.88	$5,009.08

RANK vs RANKX

```
Sales Rank - RANK =

--Default Order is ASC
RANK(
        DENSE,
        ALLSELECTED('Product Lookup'[product_category]),
        ORDERBY([Customer Sales],DESC)
)
```

```
Sales Rank - RANKX =
--Default Order is DESC
RANKX(
                ALLSELECTED('Product Lookup'[product_category]),
                [Customer Sales]
)
```

Product Category	Customer Sales	Sales Rank - RANKX	Sales Rank - RANK
Coffee	$1,651,861.15	1	1
Tea	$1,190,314.90	2	2
Bakery	$501,291.32	3	3
Drinking Chocolate	$439,942.50	4	4
Coffee beans	$240,414.95	5	5
Branded	$83,784.00	6	6
Loose Tea	$68,451.90	7	7
Flavours	$51,060.00	8	8
Packaged Chocolate	$25,584.16	9	9

Note: The RANK function in DAX is relatively new. By default, it sorts in ascending order, whereas the RANKX function sorts in descending order. Additionally, the RANK function provides the capability to partition the data while ranking the results, offering more flexibility in analysis

Here is an interesting use case:

Adding ranking at each level separately in a matrix. This approach allows for more granular insights, as it provides a rank for each hierarchical level, enhancing the depth of analysis and comparison within the dataset.

```
Sales Rank Example 2 - RANK =
--Default Order is ASC
RANK(
        DENSE,
        ALLSELECTED('Product Lookup'[product_category],'Product
Lookup'[product_type]),
        ORDERBY([Customer Sales],DESC),,PARTITIONBY('Product
Lookup'[product_category])
)
```

```
Sales Rank Example 2 - RANKX =
--Default Order is DESC
RANKX(
        ALLSELECTED('Product Lookup'[product_category],'Product
Lookup'[product_type]),
        [Customer Sales]
)
```

```
Sales Rank At Each Level of Group =
IF(
    ISINSCOPE('Product Lookup'[product_type]),
    RANK(
        DENSE,
        ALLSELECTED('Product Lookup'[product_category],'Product
Lookup'[product_type]),
        ORDERBY([Customer Sales],DESC),,
        PARTITIONBY('Product Lookup'[product_category])
    ),
    RANKX(
            ALLSELECTED('Product Lookup'[product_category]),
            [Customer Sales]
    )
)
```

```
ISINSCOPE =
IF(
    ISINSCOPE('Product Lookup'[product_type]),
    "True",
    "False"
    )
```

Product Category	Customer Sales	Sales Rank Example 2 - RANKX	Sales Rank Example 2 - RANK	Sales Rank At Each Level of Group	ISINSCOPE
Coffee	**$1,651,861.15**	**1**		**1**	**False**
Barista Espresso	$559,635.60	1	1	1	True
Gourmet brewed coffee	$430,073.20	4	2	2	True
Premium brewed coffee	$236,639.15	7	3	3	True
Organic brewed coffee	$230,235.70	8	4	4	True
Drip coffee	$195,277.50	10	5	5	True
Seasonal drink		30	6	6	True
Specialty coffee		30	6	6	True
Tea	**$1,190,314.90**	**1**		**2**	**False**
Brewed Chai tea	$468,702.90	2	1	1	True
Brewed Black tea	$289,859.50	5	2	2	True
Brewed herbal tea	$287,066.00	6	3	3	True
Brewed Green tea	$144,686.50	12	4	4	True
Bakery	**$501,291.32**	**2**		**3**	**False**
Scone	$224,272.91	9	1	1	True
Total	**$4,252,704.88**	**1**		**1**	**False**

CONCATENATEX

Showing Employees involved in top Sales

```
Employees Involved In Sales =

 CONCATENATEX(
   TOPN(
        3,
        'Employee Lookup',
        [Customer Sales],
        DESC
   ),
   'Employee Lookup'[first_name] & " " & 'Employee Lookup'[last_name], --Column to
concatenate
    ", ", --Delimiter
   'Employee Lookup'[staff_id], --Order By column
    ASC --Order
 )
```