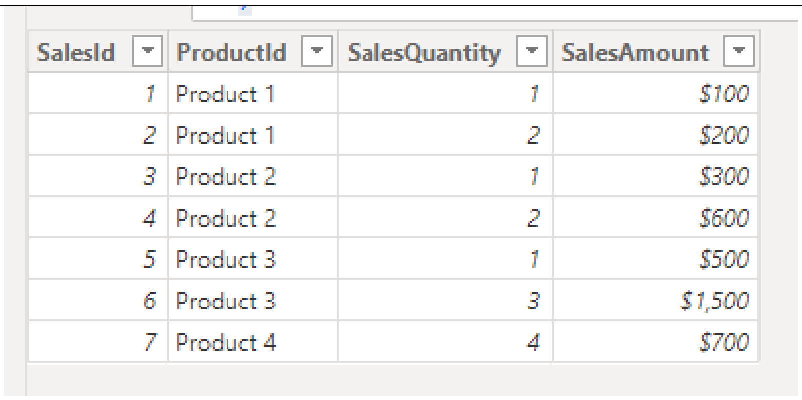
Advanced DAX – Table Manipulation Functions

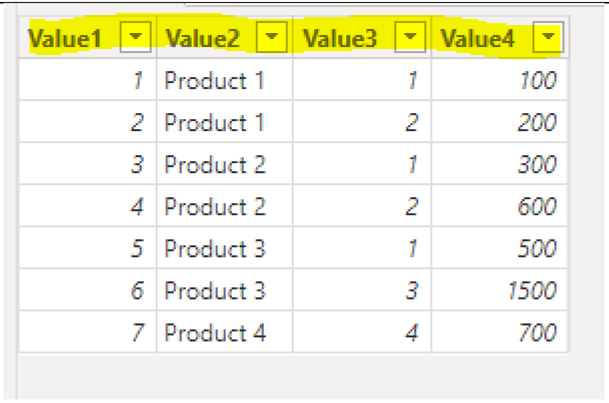
DATATABLE vs Table Constructor

```
Demo Fact (DATATABLE) =
DATATABLE (
 "SalesId", INTEGER,
 "ProductId", STRING,
 "SalesQuantity", INTEGER,
 "SalesAmount", CURRENCY,
{
    {1,"Product 1", 1,100},
    {2,"Product 1", 2,200},
    {3,"Product 2", 1,300},
    {4,"Product 2", 2,600},
    {5,"Product 3", 1,500},
    {6,"Product 3", 3,1500},
    {7,"Product 4", 4,700}
}
)
```



SalesId	ProductId	SalesQuantity	SalesAmount
1	Product 1	1	$100
2	Product 1	2	$200
3	Product 2	1	$300
4	Product 2	2	$600
5	Product 3	1	$500
6	Product 3	3	$1,500
7	Product 4	4	$700

```
Demo Fact (Table Constructor)
=
{
    (1,"Product 1", 1,100),
    (2,"Product 1", 2,200),
    (3,"Product 2", 1,300),
    (4,"Product 2", 2,600),
    (5,"Product 3", 1,500),
    (6,"Product 3", 3,1500),
    (7,"Product 4", 4,700)
}
```



Value1	Value2	Value3	Value4
1	Product 1	1	100
2	Product 1	2	200
3	Product 2	1	300
4	Product 2	2	600
5	Product 3	1	500
6	Product 3	3	1500
7	Product 4	4	700

DISTINCT vs VALUES

Although the DISTINCT and VALUES functions operate similarly in principle, VALUES will return NULLs if a value is missing in one of the joining tables. Here is an example to illustrate this concept.

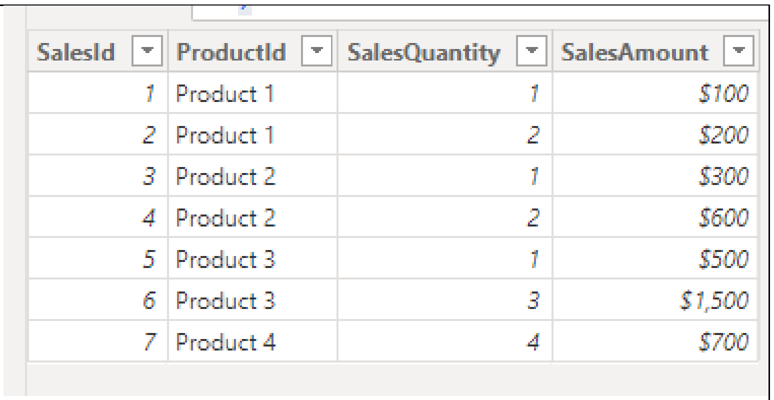
We have created Fact and Dimension (Dim) tables using the DATATABLE function. In the Fact table, there are four types of products, while in the Dim table, only three types of products are present. We then established a relationship between these two tables based on ProductID.

Consider the outputs of the DISTINCT and VALUES functions when applied to the Dim table:

- The DISTINCT function will return the unique ProductIDs present in the Dim table.
- The VALUES function will return the unique ProductIDs from the Dim table but will include NULLs for any ProductID that is present in the Fact table but not in the Dim table.

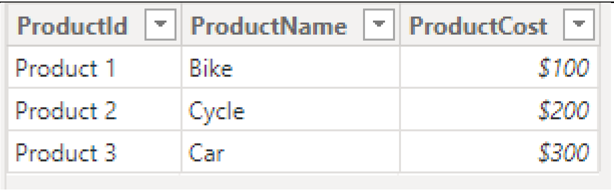
*Creating Fact and Dimension tables

```
Demo Fact (DATATABLE) =
DATATABLE (
 "SalesId", INTEGER,
 "ProductId", STRING,
 "SalesQuantity", INTEGER,
 "SalesAmount", CURRENCY,
{
    {1,"Product 1", 1,100},
    {2,"Product 1", 2,200},
    {3,"Product 2", 1,300},
    {4,"Product 2", 2,600},
    {5,"Product 3", 1,500},
    {6,"Product 3", 3,1500},
    {7,"Product 4", 4,700}
}
)
```



SalesId	ProductId	SalesQuantity	SalesAmount
1	Product 1	1	$100
2	Product 1	2	$200
3	Product 2	1	$300
4	Product 2	2	$600
5	Product 3	1	$500
6	Product 3	3	$1,500
7	Product 4	4	$700

```
Demo Dim (DATATABLE) =
DATATABLE (
 "ProductId", STRING,
 "ProductName", STRING,
 "ProductCost", CURRENCY,
{
    {"Product 1", "Bike", 100},
    {"Product 2", "Cycle", 200},
    {"Product 3", "Car", 300}
}
)
```



ProductId	ProductName	ProductCost
Product 1	Bike	$100
Product 2	Cycle	$200
Product 3	Car	$300

*Creating Relationship

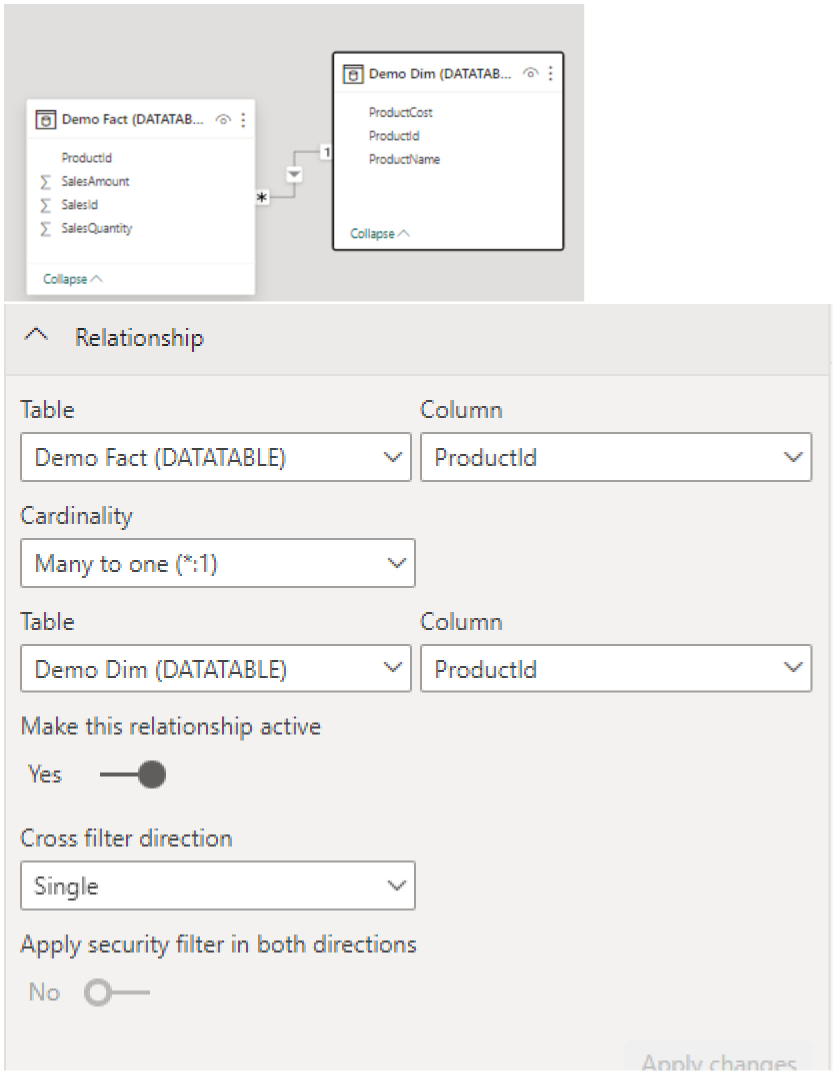


Table: Demo Fact (DATATABLE)
Column: ProductId

Cardinality: Many to one (*:1)

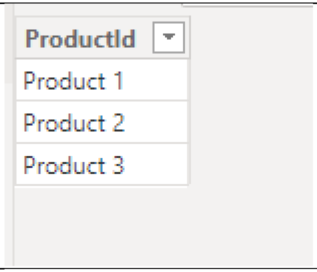
Table: Demo Dim (DATATABLE)
Column: ProductId

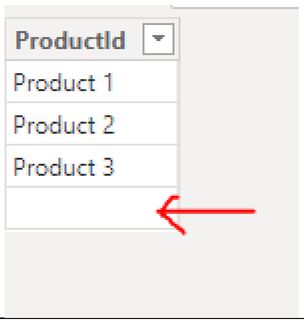
Make this relationship active: Yes

Cross filter direction: Single

Apply security filter in both directions: No

*Difference between DISTINCT and VALUES

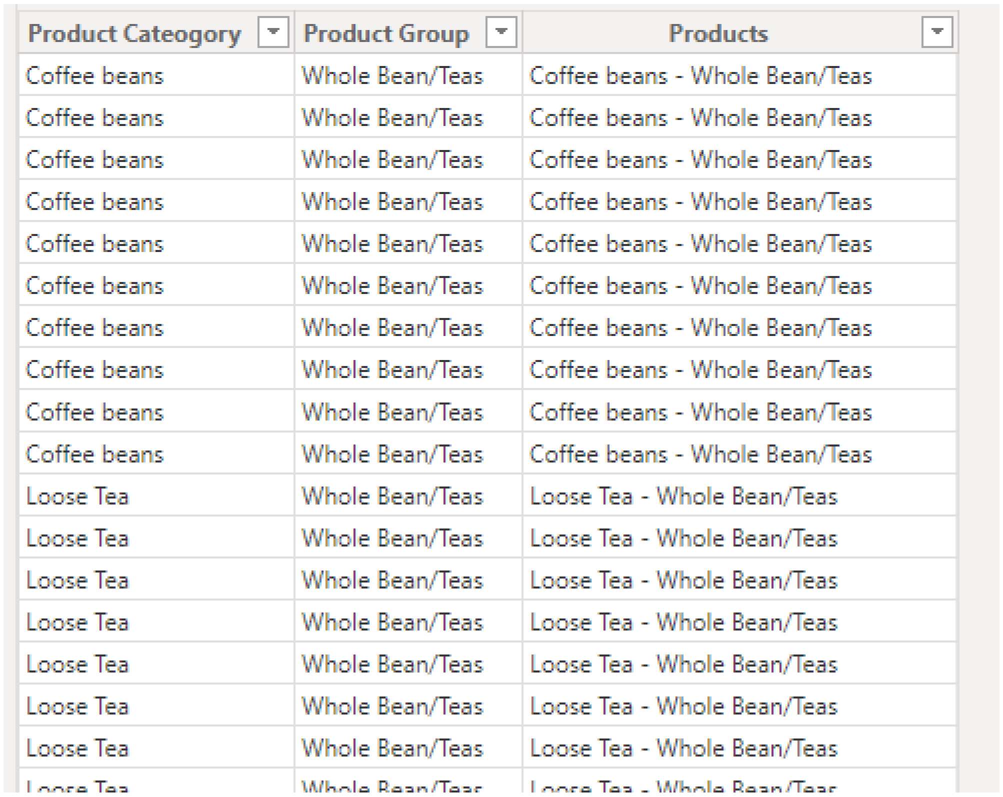
`Demo DISTINCT =`
`DISTINCT(`
`        'Demo Dim (DATATABLE)'[ProductId]`
`)`
	
	ProductId
	Product 1
	Product 2
	Product 3

`Demo VALUES =`
`VALUES(`
`        'Demo Dim (DATATABLE)'[ProductId]`
`)`
	
	ProductId
	Product 1
	Product 2
	Product 3

SELECTCOLUMNS vs ADDCOLUMNS

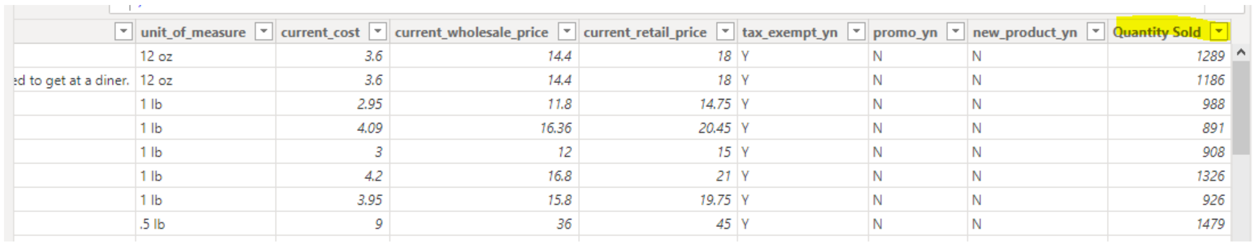
The SELECTCOLUMNS function has the same signature as the ADDCOLUMNS function and behaves similarly, with a key difference. While ADDCOLUMNS starts with the specified table and adds new columns to it, SELECTCOLUMNS begins with an empty table and only includes the columns defined in the formula.

```
Demo SELECTCOLUMNS =
SELECTCOLUMNS(
                'Product Lookup',
                "Product Cateogory", 'Product Lookup'[product_category],
                "Product Group", 'Product Lookup'[product_group],
                "Products",'Product Lookup'[product_category]&" - "& 'Product
Lookup'[product_group]
 )
```



Product Cateogory	Product Group	Products
Coffee beans	Whole Bean/Teas	Coffee beans - Whole Bean/Teas
Coffee beans	Whole Bean/Teas	Coffee beans - Whole Bean/Teas
Coffee beans	Whole Bean/Teas	Coffee beans - Whole Bean/Teas
Coffee beans	Whole Bean/Teas	Coffee beans - Whole Bean/Teas
Coffee beans	Whole Bean/Teas	Coffee beans - Whole Bean/Teas
Coffee beans	Whole Bean/Teas	Coffee beans - Whole Bean/Teas
Coffee beans	Whole Bean/Teas	Coffee beans - Whole Bean/Teas
Coffee beans	Whole Bean/Teas	Coffee beans - Whole Bean/Teas
Coffee beans	Whole Bean/Teas	Coffee beans - Whole Bean/Teas
Coffee beans	Whole Bean/Teas	Coffee beans - Whole Bean/Teas
Loose Tea	Whole Bean/Teas	Loose Tea - Whole Bean/Teas
Loose Tea	Whole Bean/Teas	Loose Tea - Whole Bean/Teas
Loose Tea	Whole Bean/Teas	Loose Tea - Whole Bean/Teas
Loose Tea	Whole Bean/Teas	Loose Tea - Whole Bean/Teas
Loose Tea	Whole Bean/Teas	Loose Tea - Whole Bean/Teas
Loose Tea	Whole Bean/Teas	Loose Tea - Whole Bean/Teas
Loose Tea	Whole Bean/Teas	Loose Tea - Whole Bean/Teas
Loose Tea	Whole Bean/Teas	Loose Tea - Whole Bean/Teas

```
Demo ADDCOLUMNS =
 ADDCOLUMNS(
            'Product Lookup',
            "Quantity Sold", SUMX(RELATEDTABLE('Sales by Store'), 'Sales by Store'[quantity_sold])
 )
```

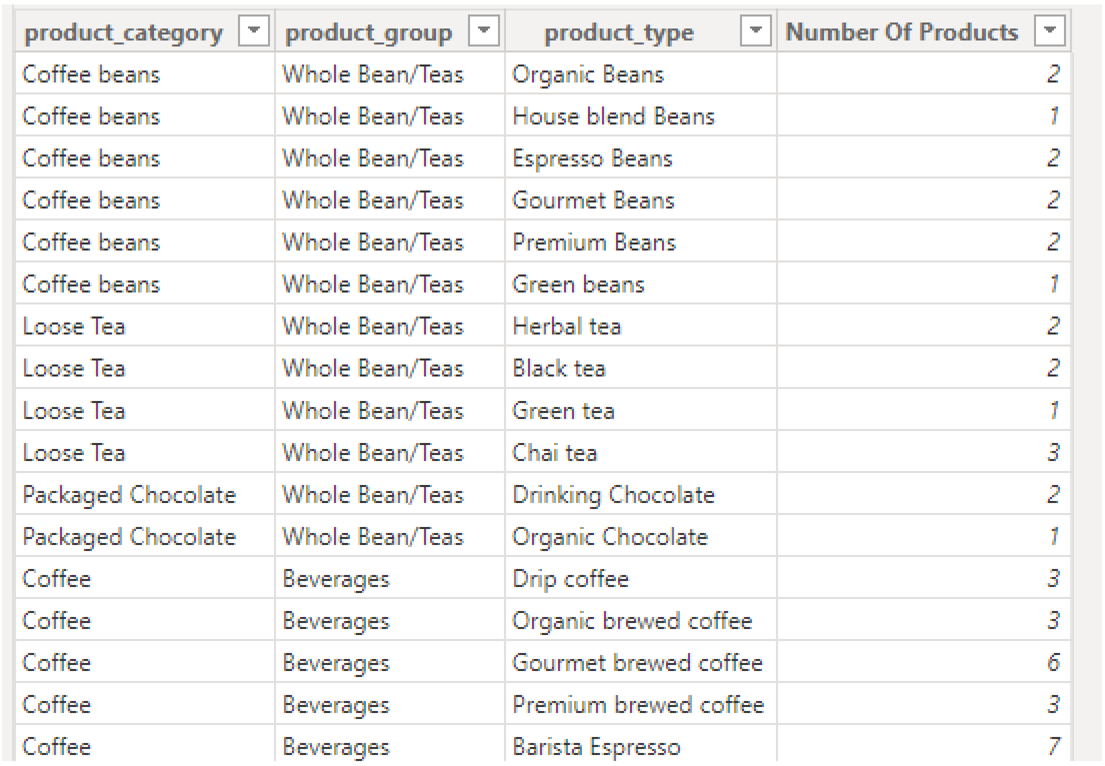


	unit_of_measure	current_cost	current_wholesale_price	current_retail_price	tax_exempt_yn	promo_yn	new_product_yn	Quantity Sold
	12 oz	3.6	14.4	18	Y	N	N	1289
ed to get at a diner.	12 oz	3.6	14.4	18	Y	N	N	1186
	1 lb	2.95	11.8	14.75	Y	N	N	988
	1 lb	4.09	16.36	20.45	Y	N	N	891
	1 lb	3	12	15	Y	N	N	908
	1 lb	4.2	16.8	21	Y	N	N	1326
	1 lb	3.95	15.8	19.75	Y	N	N	926
	.5 lb	9	36	45	Y	N	N	1479

SUMMARIZE vs SUMMARIZECOLUMNS vs GROUPBY

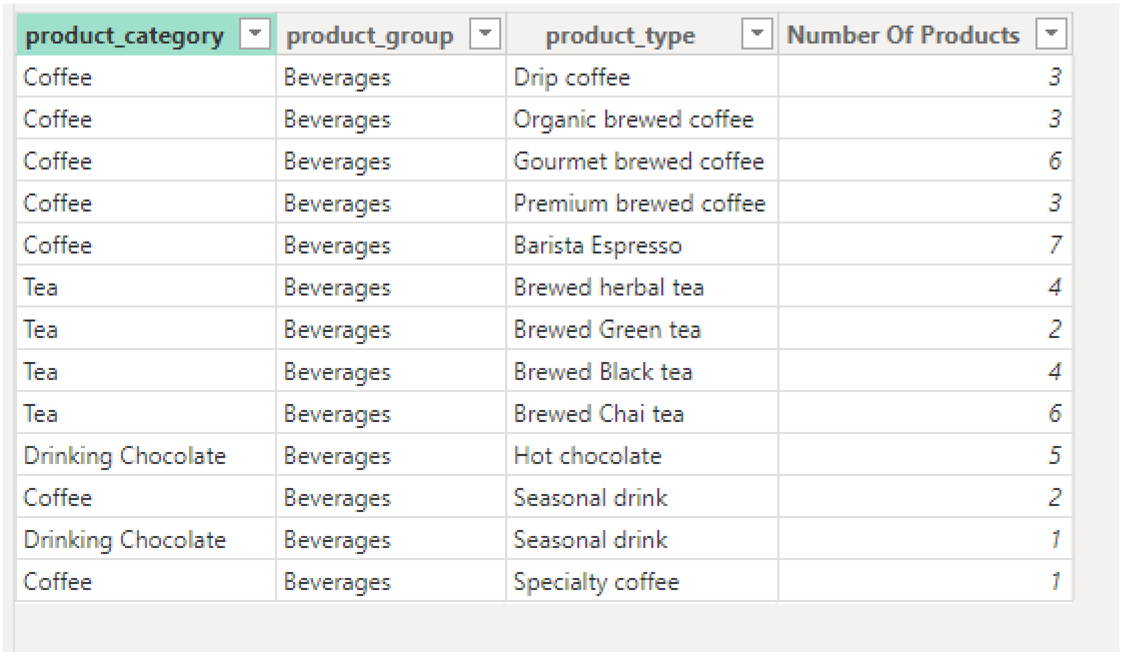
One distinct difference between SUMMARIZE and SUMMARIZECOLUMNS is that SUMMARIZECOLUMNS allows you to apply filters directly on the data that you need to summarize.

```
Demo SUMMARIZE =

SUMMARIZE(
    'Product Lookup',
    'Product Lookup'[product_category],
    'Product Lookup'[product_group],
    'Product Lookup'[product_type],
    "Number Of Products",COUNT('Product Lookup'[product_id])
 )
```



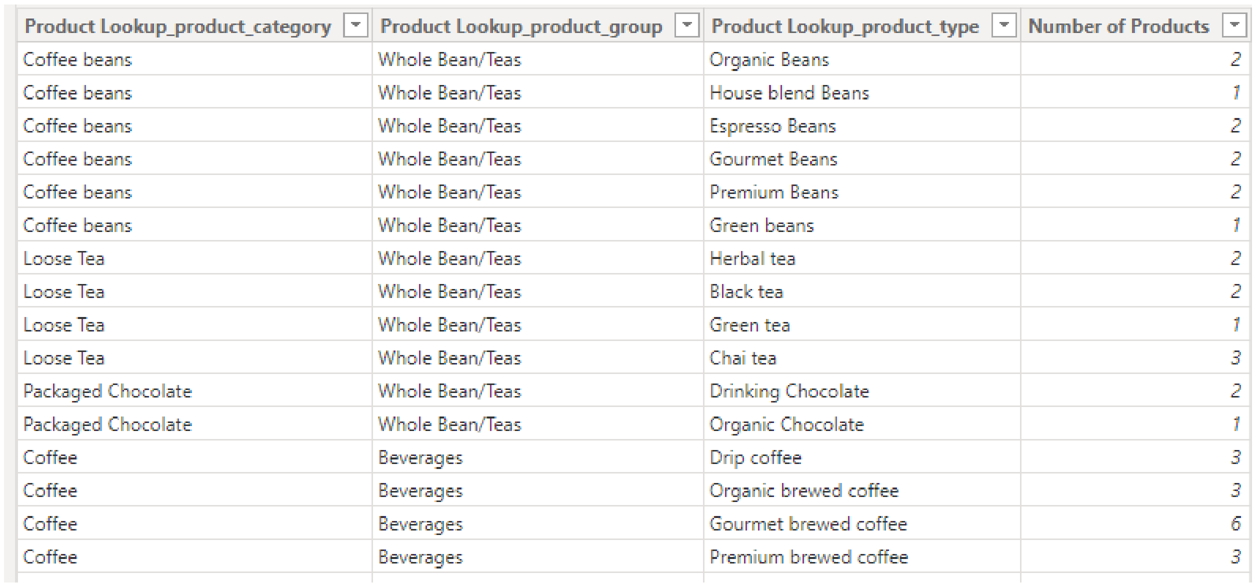
product_category	product_group	product_type	Number Of Products
Coffee beans	Whole Bean/Teas	Organic Beans	2
Coffee beans	Whole Bean/Teas	House blend Beans	1
Coffee beans	Whole Bean/Teas	Espresso Beans	2
Coffee beans	Whole Bean/Teas	Gourmet Beans	2
Coffee beans	Whole Bean/Teas	Premium Beans	2
Coffee beans	Whole Bean/Teas	Green beans	1
Loose Tea	Whole Bean/Teas	Herbal tea	2
Loose Tea	Whole Bean/Teas	Black tea	2
Loose Tea	Whole Bean/Teas	Green tea	1
Loose Tea	Whole Bean/Teas	Chai tea	3
Packaged Chocolate	Whole Bean/Teas	Drinking Chocolate	2
Packaged Chocolate	Whole Bean/Teas	Organic Chocolate	1
Coffee	Beverages	Drip coffee	3
Coffee	Beverages	Organic brewed coffee	3
Coffee	Beverages	Gourmet brewed coffee	6
Coffee	Beverages	Premium brewed coffee	3
Coffee	Beverages	Barista Espresso	7

```
Demo SUMMARIZECOLUMNS =

SUMMARIZECOLUMNS(
    'Product Lookup'[product_category],
    'Product Lookup'[product_group],
    'Product Lookup'[product_type],
    FILTER('Product Lookup','Product Lookup'[product_group] = "Beverages"),
    "Number Of Products",COUNT('Product Lookup'[product_id])
 )
```



product_category	product_group	product_type	Number Of Products
Coffee	Beverages	Drip coffee	3
Coffee	Beverages	Organic brewed coffee	3
Coffee	Beverages	Gourmet brewed coffee	6
Coffee	Beverages	Premium brewed coffee	3
Coffee	Beverages	Barista Espresso	7
Tea	Beverages	Brewed herbal tea	4
Tea	Beverages	Brewed Green tea	2
Tea	Beverages	Brewed Black tea	4
Tea	Beverages	Brewed Chai tea	6
Drinking Chocolate	Beverages	Hot chocolate	5
Coffee	Beverages	Seasonal drink	2
Drinking Chocolate	Beverages	Seasonal drink	1
Coffee	Beverages	Specialty coffee	1

```
Demo GROUPBY =
 GROUPBY(
            'Product Lookup',
            'Product Lookup'[product_category],
            'Product Lookup'[product_group],
            'Product Lookup'[product_type],
            "Number of Products",COUNTX(CURRENTGROUP(),'Product Lookup'[product_id])
 )
```



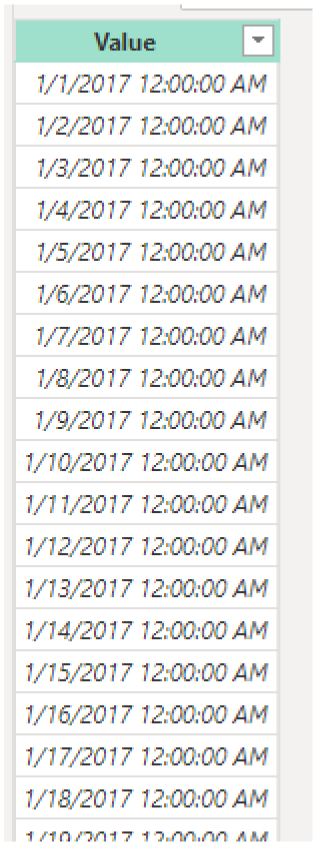
Product Lookup_product_category	Product Lookup_product_group	Product Lookup_product_type	Number of Products
Coffee beans	Whole Bean/Teas	Organic Beans	2
Coffee beans	Whole Bean/Teas	House blend Beans	1
Coffee beans	Whole Bean/Teas	Espresso Beans	2
Coffee beans	Whole Bean/Teas	Gourmet Beans	2
Coffee beans	Whole Bean/Teas	Premium Beans	2
Coffee beans	Whole Bean/Teas	Green beans	1
Loose Tea	Whole Bean/Teas	Herbal tea	2
Loose Tea	Whole Bean/Teas	Black tea	2
Loose Tea	Whole Bean/Teas	Green tea	1
Loose Tea	Whole Bean/Teas	Chai tea	3
Packaged Chocolate	Whole Bean/Teas	Drinking Chocolate	2
Packaged Chocolate	Whole Bean/Teas	Organic Chocolate	1
Coffee	Beverages	Drip coffee	3
Coffee	Beverages	Organic brewed coffee	3
Coffee	Beverages	Gourmet brewed coffee	6
Coffee	Beverages	Premium brewed coffee	3

Note: The CURRENTGROUP function is used within the GROUPBY function to create an extension column. It returns the set of rows from the table specified in the GROUPBY function that belong to the current group being processed. CURRENTGROUP takes no arguments and can only be used as the first argument in certain aggregation functions like SUMX, COUNTX, etc.

GENERATESERIES

```
Demo GENERATESERIES =

GENERATESERIES(
                MIN('Sales by Store'[transaction_date]),
                MAX('Sales by Store'[transaction_date])
)
```

* Generates a series of dates starting and ending between the dates specified in the two arguments.



Value
1/1/2017 12:00:00 AM
1/2/2017 12:00:00 AM
1/3/2017 12:00:00 AM
1/4/2017 12:00:00 AM
1/5/2017 12:00:00 AM
1/6/2017 12:00:00 AM
1/7/2017 12:00:00 AM
1/8/2017 12:00:00 AM
1/9/2017 12:00:00 AM
1/10/2017 12:00:00 AM
1/11/2017 12:00:00 AM
1/12/2017 12:00:00 AM
1/13/2017 12:00:00 AM
1/14/2017 12:00:00 AM
1/15/2017 12:00:00 AM
1/16/2017 12:00:00 AM
1/17/2017 12:00:00 AM
1/18/2017 12:00:00 AM
1/19/2017 12:00:00 AM

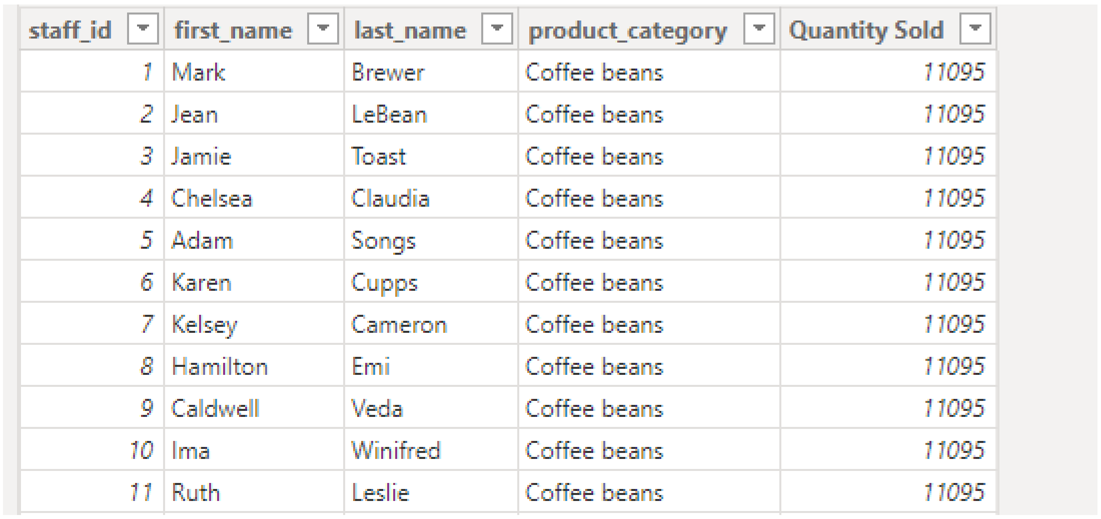
GENERATE vs GENERATEALL

GENERATE: Returns a table with the Cartesian product between each row in *table1* and the table that results from evaluating *table2* in the context of the current row from *table1*.

GENERATE(<table1>, <table2>)

In the following example, I have added a Quantity Sold column to show the number of products sold by each employee in a particular category. However, you can observe that the numbers seem incorrect or repetitive. This issue arises because the GENERATE function generates two tables separately and produces a Cartesian product without applying filters.

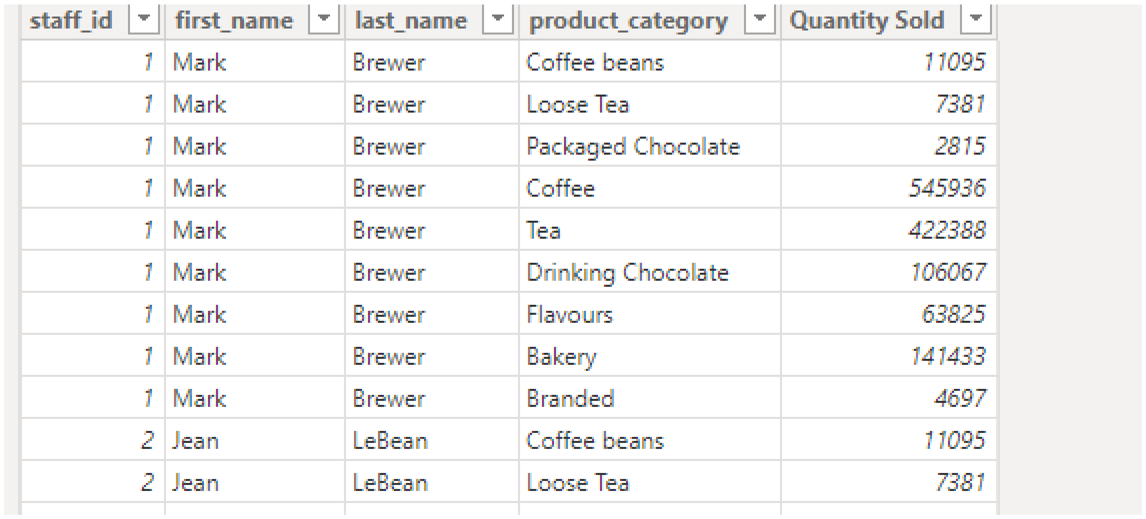
```
Demo GENERATE =

Demo GENERATE =
GENERATE(   //Table 1
            SUMMARIZE(
                        'Employee Lookup',
                        'Employee Lookup'[staff_id],
                        'Employee Lookup'[first_name],
                        'Employee Lookup'[last_name]
            ),
            //Table 2
            SUMMARIZE(
                        'Product Lookup',
                        'Product Lookup'[product_category],
                        "Quantity Sold",SUMX('Sales by Store','Sales by Store'[quantity_sold])
            )
```



staff_id	first_name	last_name	product_category	Quantity Sold
1	Mark	Brewer	Coffee beans	11095
2	Jean	LeBean	Coffee beans	11095
3	Jamie	Toast	Coffee beans	11095
4	Chelsea	Claudia	Coffee beans	11095
5	Adam	Songs	Coffee beans	11095
6	Karen	Cupps	Coffee beans	11095
7	Kelsey	Cameron	Coffee beans	11095
8	Hamilton	Emi	Coffee beans	11095
9	Caldwell	Veda	Coffee beans	11095
10	Ima	Winifred	Coffee beans	11095
11	Ruth	Leslie	Coffee beans	11095

To overcome this issue, GENERATEALL function comes to the rescue.

```
Demo GENERATEALL =

GENERATEALL(   //Table 1
            SUMMARIZE(
                        'Employee Lookup',
                        'Employee Lookup'[staff_id],
                        'Employee Lookup'[first_name],
                        'Employee Lookup'[last_name]
            ),
            //Table 2
            SUMMARIZE(
                        'Product Lookup',
                        'Product Lookup'[product_category]
                        ,"Quantity Sold",SUMX('Sales by Store','Sales by Store'[quantity_sold])
            )
)
```



staff_id	first_name	last_name	product_category	Quantity Sold
1	Mark	Brewer	Coffee beans	11095
1	Mark	Brewer	Loose Tea	7381
1	Mark	Brewer	Packaged Chocolate	2815
1	Mark	Brewer	Coffee	545936
1	Mark	Brewer	Tea	422388
1	Mark	Brewer	Drinking Chocolate	106067
1	Mark	Brewer	Flavours	63825
1	Mark	Brewer	Bakery	141433
1	Mark	Brewer	Branded	4697
2	Jean	LeBean	Coffee beans	11095
2	Jean	LeBean	Loose Tea	7381