Power BI - Advanced DAX - p2 - Lecture 9

```
Revenue = 'Sales Data'[OrderQuantity] * 'Sales Data'[Retail Price]
```

```
Revenue = 'Sales Data'[OrderQuantity] * RELATED('Product Lookup'[ProductPrice])
```

Instead of adding new column of Retail price from Product Lookup table, Use Related function and directly use it in the formulae to optimise the space.

```
Total Revenue = SUM('Sales Data'[Revenue])
```

CALCULATE

CALCULATE()
Evaluates an expression in a context that is modified by filters

```
USA Revenue = CALCULATE([Revenue] , 'Territory Lookup'[Country] = "United State")
```

=CALCULATE(Expression, [Filter1], [Filter2],...)

Name of an existing measure or a DAX formula for a valid measure
Examples:
- [Total Orders]
- SUM('Returns Data'[Return Quantity])

A Boolean (True/False) expression or a table expression that defines a filter.
Note: these require fixed values or aggregation functions that return a scalar value (you cannot create filters based on measures)
Examples:
- 'Territory Lookup'[Country] = "USA"
- Calendar[Year] <> MAX(Calendar[Year])

PRO TIP:
Think of CALCULATE as a filter modifier; it allows you to overrule existing report filters and "force" new filter context

EXAMPLE: CALCULATE

```
1 Red Sales = CALCULATE( [Quantity Sold], 'Product Lookup'[Product Color] = "Red" )
```

- Here we've defined a new measure named Red Sales, which evaluates the Quantity Sold measure under a filter context where the product color is "Red"

Product Color	Quantity Sold	Red Sales
Black	10,590	4,011
Multi	5,756	4,011
Red	4,011	4,011
Silver	3,257	4,011
Total	**23,614**	**4,011**

Note how we see the the same repeated values for each product color, and even the total!

HEY THIS IS IMPORTANT!

- The CALCULATE function modifies and overrules any competing filter context!

- In this matrix, the "Black" row has competing filter context: Product Color = Black (from the row label) and Product Color= "Red" (from the CALCULATE function)

- Both can't be true at the same time, so the "Red" filter from CALCULATE takes priority

STEP 1

Filter context is detected & applied

Product Color	Quantity Sold	Red Sales
Black	10,590	4,011
Red	4,011	4,011
Silver	3,257	4,011

'Product Lookup'[Product Color] = "Black"

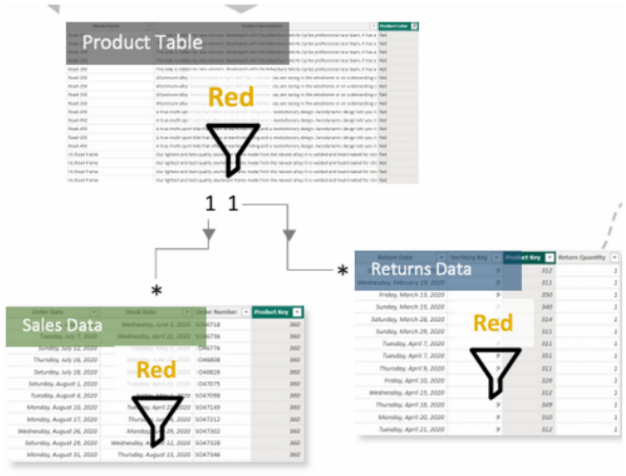
CALCULATE

Filters are modified by CALCULATE
[Product Color] = "Red"

If the measure being evaluated contains a CALCULATE function, filter context is overwritten between Step 1 & Step 2

STEP 2

Filters flow "downstream" to related tables



STEP 3

Measure evaluates against the filtered table

```
1 Quantity Sold =
2 SUM( 'Sales Data'[Order Quantity] )
```

- Sum of the Order Quantity column in the Sales Data table, filtered to rows where the product color is "Red"

= 4,011

```
1 Total Orders = DISTINCTCOUNT('Sales Data'[OrderNumber])
```

25164	84174
Total Orders	Quantity Sold

```
1 Bulk Orders =
2     CALCULATE(
3         [Total Orders],
4         'Sales Data'[OrderQuantity] > 1
5 )
```

Filter

```
Weekend Orders =
    CALCULATE(
        [Total Orders],
        'Calendar Lookup'[Weekend] = "Weekend")
```

```
Weekday Orders = CALCULATE([Total Orders] , 'Calendar Lookup'[Weekend] = "Weekday")
```

DAX MEASURE TOTALS

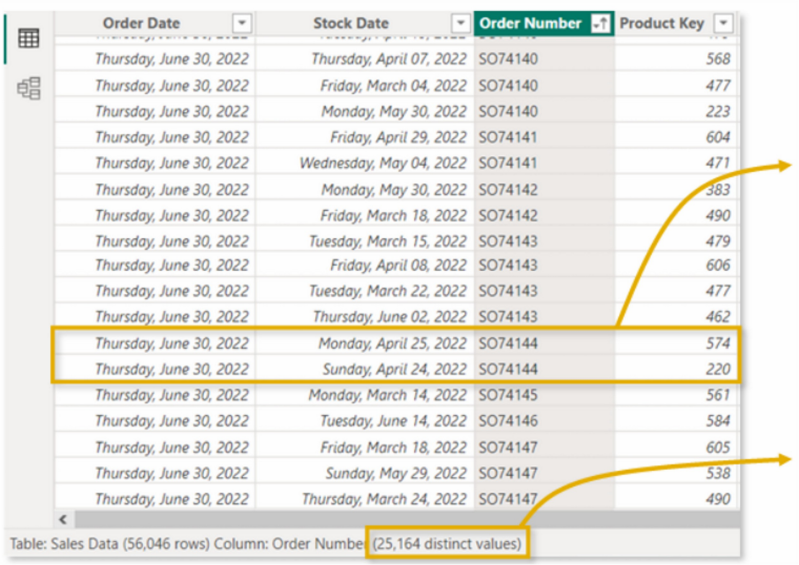
Measure totals may seem incorrect or inconsistent depending on how they are calculated, because they don't simply add up the visible values in the report

```
1 Total Orders =
2 DISTINCTCOUNT(
3     'Sales Data'[Order Number]
4 )
```

[Total Orders] counts distinct orders in the Sales Data table

Total Returns look right, but shouldn't Total Orders be 37,888??

Category Name	Total Returns	Total Orders
Accessories	1,115	16,983
Bikes	427	13,929
Clothing	267	6,976
Total	**1,809**	**25,164**



Order Date	Stock Date	Order Number	Product Key
Thursday, June 30, 2022	Thursday, April 07, 2022	SO74140	568
Thursday, June 30, 2022	Friday, March 04, 2022	SO74140	477
Thursday, June 30, 2022	Monday, May 30, 2022	SO74140	223
Thursday, June 30, 2022	Friday, April 29, 2022	SO74141	604
Thursday, June 30, 2022	Wednesday, May 04, 2022	SO74141	471
Thursday, June 30, 2022	Monday, May 30, 2022	SO74142	383
Thursday, June 30, 2022	Friday, March 18, 2022	SO74142	490
Thursday, June 30, 2022	Tuesday, March 15, 2022	SO74143	479
Thursday, June 30, 2022	Friday, April 08, 2022	SO74143	606
Thursday, June 30, 2022	Tuesday, March 22, 2022	SO74143	477
Thursday, June 30, 2022	Thursday, June 02, 2022	SO74143	462
Thursday, June 30, 2022	Monday, April 25, 2022	SO74144	574
Thursday, June 30, 2022	Sunday, April 24, 2022	SO74144	220
Thursday, June 30, 2022	Monday, March 14, 2022	SO74145	561
Thursday, June 30, 2022	Tuesday, June 14, 2022	SO74146	584
Thursday, June 30, 2022	Friday, March 18, 2022	SO74147	605
Thursday, June 30, 2022	Sunday, May 29, 2022	SO74147	538
Thursday, June 30, 2022	Thursday, March 24, 2022	SO74147	490

Table: Sales Data (56,046 rows) Column: Order Numbe (25,164 distinct values)

Order SO74144 included two products: a bike and a helmet. That counts as 1 distinct order for the Total and 1 distinct order for BOTH Accessories & Bikes

With no filter context, there are 25,164 total distinct orders

PRO TIP:
Understand EXACTLY how your measures calculate and what they are designed to measure

ASSIGNMENT: CALCULATE

1. Create a new measure named Bike Returns to calculate the total quantity of bikes returned.

2. Create a matrix to show Bike Returns (values) by Start of Month (rows). What do you notice about the volume of bike returns over time?

3. Create a new measure named Bike Sales to calculate the total quantity of bikes sold, and add it to the matrix. What do you notice?

4. Create a new measure named Bike Return Rate using either CALCULATE or DIVIDE, and add it to the matrix

5. How would you respond to the Product VP's concerns about rising bike returns?

```
Bikes Returns = CALCULATE([Return Quantity] , 'Product Categories Lookup'
[CategoryName] = "Bikes")
```

```
Bikes Sales = CALCULATE([Quantity Sold] , 'Product Categories Lookup'[CategoryName]
= "Bikes")
```

```
Bikes Return Rate = CALCULATE([Return Rate],'Product Categories Lookup'
[CategoryName] = "Bikes")
```

Start of Year	Bikes Returns	Bikes Sales	Bikes Retur
01-01-2020	**86**	**2630**	**3.27%**
01-01-2020	4	184	2.17%
01-02-2020	4	165	2.42%
01-03-2020	9	198	4.55%
01-04-2020	14	204	6.86%
01-05-2020	11	206	5.34%
01-06-2020	4	212	1.89%
01-07-2020	3	247	1.21%
01-08-2020	6	278	2.16%
01-09-2020	2	196	1.02%
01-10-2020	11	223	4.93%
01-11-2020	5	191	2.62%
01-12-2020	13	326	3.99%
01-01-2021	**172**	**5610**	**3.07%**
01-01-2021	8	242	3.31%
01-02-2021	8	267	3.00%
01-03-2021	8	266	3.01%
01-04-2021	5	290	1.72%
01-05-2021	10	329	3.04%
01-06-2021	8	312	2.56%
01-07-2021	12	506	2.37%
01-08-2021	14	485	2.89%
01-09-2021	22	575	3.83%
01-10-2021	26	612	4.25%
01-11-2021	25	688	3.63%
01-12-2021	26	1038	2.50%
01-01-2022	**171**	**5689**	**3.01%**
01-01-2022	14	766	1.83%
01-02-2022	22	806	2.73%
01-03-2022	27	888	3.04%
01-04-2022	38	956	3.97%
01-05-2022	36	1116	3.23%
01-06-2022	34	1157	2.94%
Total	**429**	**13929**	**3.08%**

Product A	Product B
10000	90000

product A = 10K/100K = 10%

product B = 90K/100K = 90%

Total Orders	All Orders	% of All Orders
16983	25164	67.49%
13929	25164	55.35%
6976	25164	27.72%
	25164	
25164	**25164**	**100.00%**

ALL

ALL() :-
Returns all rows in a table, or all values in a column, ignoring any filters that have been applied.

=ALL(Table or Column, [Column2], [Column3],...)

The table or column that you want to clear filters on
Examples:
- Transactions
- Products[Category]

Additional columns that you want to clear filters on (optional)
- Cannot specify columns if your first parameter is a table
- All columns must include the table name and come from the same table

Examples:
- 'Customer Lookup'[City], 'Customer Lookup'[Country]
- Products[Product Name]

PRO TIP:
Instead of adding filter context, the *ALL* function removes it. This is often used in "% of Total" calculations, when the denominator needs to remain fixed regardless of filter context.

```
All Orders =
    CALCULATE(
        [Total Orders],
        ALL(
            'Sales Data'))
```

```
% of All Orders =
    DIVIDE(
        [Total Orders],
        [All Orders])
```

```
Overall Average Price =
    CALCULATE(
        [Average Retail Price],
        ALL(
            'Product Lookup'))
```

CategoryName	Weekend Orders	Total Orders	All Orders	% of All Orders	Average Retail Price	Overall Average Price
		1	25,165	0.00%		$714.44
Accessories	4,913	16,983	25,165	67.49%	$34.26	$714.44
Bikes	3,995	13,929	25,165	55.35%	$1,541.38	$714.44
Clothing	1,962	6,976	25,165	27.72%	$50.68	$714.44
Components			25,165		$432.19	$714.44
Total	**7,214**	**25,165**	**25,165**	**100.00%**	**$714.44**	**$714.44**

ASSIGNMENT: CALCULATE & ALL

1.Create a new measure named *All Returns* to calculate the total number of returns, regardless of filter context

2. Create a new measure named *% of All Returns* that divides *Total Returns* by *All Returns*

3.Create a matrix to show *% of All Returns* (values) by product *Category Name* (rows). Which category accounts for the largest percentage of returns? The smallest?

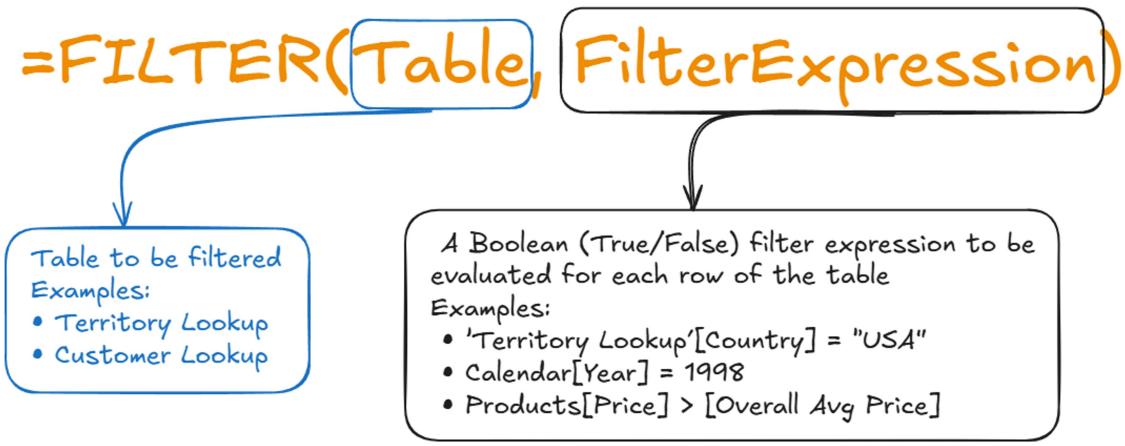
```
All Returns =
    CALCULATE(
        [Total Returns],
        ALL(
            'Returns Data'))
```

```
% of All Returns =
    DIVIDE(
        [Total Returns],
        [All Returns])
```

CategoryName	Total Returns	All Returns	% of All Returns
		1,809	
Accessories	1,115	1,809	61.64%
Bikes	427	1,809	23.60%
Clothing	267	1,809	14.76%
Components		1,809	
Total	**1,809**	**1,809**	**100.00%**

FILTER

FILTER() :-
Returns a table that represents a subset of another table or expression



HEY THIS IS IMPORTANT!

- FILTER is used to add new filter context, and can handle more complex filter expressions than CALCULATE (by referencing measures, for example)

- Since FILTER returns an entire table, it's often nested within other functions, like CALCULATE or SUMX

PRO TIP:
• Since FILTER iterates through each row in a table, it can be slow and computationally expensive; only use FILTER if a simple CALCULATE function won't get the job done!

```
High Ticket Orders =
    CALCULATE(
        [Total Orders],
        'Product Lookup'[ProductPrice] > 714.44 ))
```

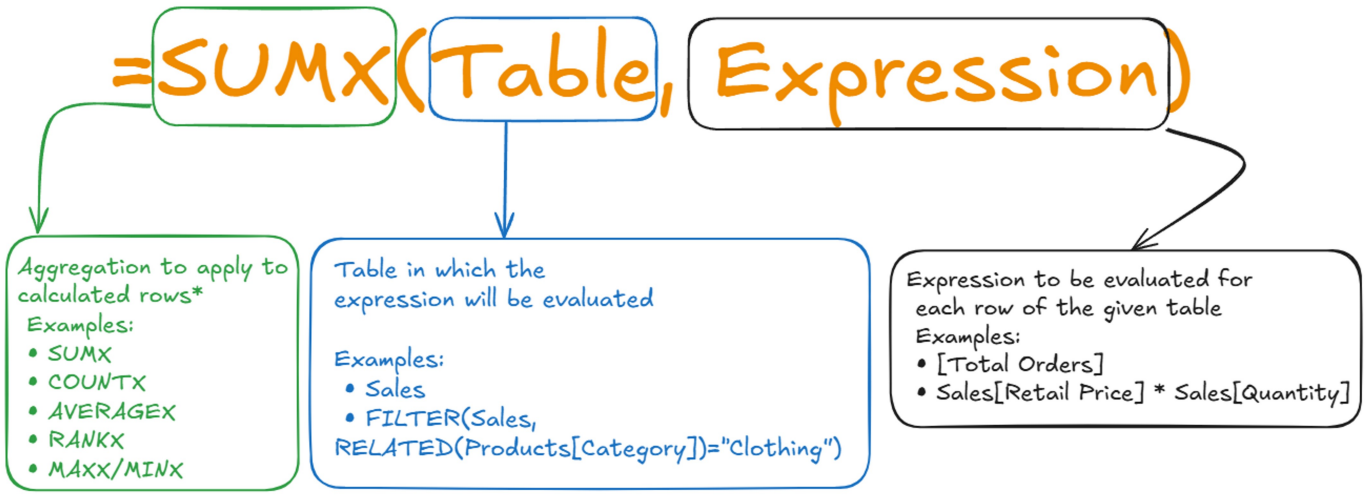
Instead of calling the hard coded value, we must use some filter function to calculate the values row basis.

```
High Ticket Orders =
    CALCULATE(
        [Total Orders],
        FILTER(
            'Product Lookup',
            'Product Lookup'[ProductPrice] > [Overall Average Price]))
```

CategoryName	Weekend Orders	Total Orders	All Orders	% of All Orders	Average Retail Price	Overall Average Price	High Ticket Orders
		1	25,165	0.00%		$714.44	
Accessories	4,913	16,983	25,165	67.49%	$34.26	$714.44	
Bikes	3,995	13,929	25,165	55.35%	$1,541.38	$714.44	11312
Clothing	1,962	6,976	25,165	27.72%	$50.68	$714.44	
Components			25,165		$432.19	$714.44	
Total	**7,214**	**25,165**	**25,165**	**100.00%**	**$714.44**	**$714.44**	**11312**

ITERATOR FUNCTIONS

Iterator (or "X") functions allow you to loop through the same expression on each row of a table, then apply some sort of aggregation to the results (SUM, MAX, etc.)



PRO TIP:
• Imagine that iterator functions add a temporary new column to a table, calculate a value in each row based on the given expression, then aggregate the values within that temporary column (similar to SUMPRODUCT in Excel).

Product Price	Quantity	Overall Price
100	10	1,000
200	5	1,000
400	5	2,000
500	2	1,000
1000	1	1,000

= 6,000

Total Cost = (100 * 10) + (200* 5) + (400 * 5) + (500*2) + (1000 * 1) = 6,000

```
Total Revenue =
    SUMX(
        'Sales Data',
        'Sales Data'[OrderQuantity] *
        RELATED(
            'Product Lookup'[ProductPrice]))
```

ASSIGNMENT: ITERATORS

1. Create a new measure named Total Cost that multiplies the order quantities in the Sales Data table by the product cost in the Product Lookup table, then calculates the sum

```
Total Cost =
    SUMX(
        'Sales Data',
        'Sales Data'[OrderQuantity] *
        RELATED(
            'Product Lookup'[ProductCost]))
```

2. Create a new measure named Total Profit (revenue minus cost)

```
Total Profit =
    [Total Revenue] - [Total Cost]
```

3. Create a matrix to show Total Profit (values) by Year (rows). How much profit has AdventureWorks earned so far in 2022?

Year	Total Revenue	Total Cost	Total Profit
2020	$64,04,933.58	$38,03,328.35	$26,01,605.23
2021	$93,24,203.79	$53,57,171.92	$39,67,031.87
2022	$91,85,449.45	$52,96,486.05	$38,88,963.40
Total	**$2,49,14,586.82**	**$1,44,56,986.32**	**$1,04,57,600.50**

TIME INTELLIGENCE

Time Intelligence patterns are used to calculate common date-based comparisons

Performance To-Date

```
=CALCULATE(Measure, DATESYTD(Calendar[Date]))
```

Use DATESYTD for Years, DATESQTD for Quarters, DATESMTD for Months.

Previous Period

```
=CALCULATE(Measure,
DATEADD(Calendar[Date], -1, MONTH))
```

Select an interval (DAY, MONTH, QUARTER, or YEAR) and the # of intervals to compare (e.g. previous month, rolling 10-day)

Running Total

```
=CALCULATE(Measure,
DATESINPERIOD(Calendar[Date],
MAX(Calendar[Date]), -10, DAY))
```

PRO TIP:
To calculate a moving average, use the running total calculation above and divide by the number of intervals

```
YTD Revenue =
    CALCULATE(
        [Total Revenue],
        DATESYTD(
            'Calendar Lookup'[Date]
        )
    )
```

Year	Total Revenue	YTD Revenue
2020	**$64,04,934**	**$64,04,934**
January	$5,85,313	$5,85,313
February	$5,32,226	$11,17,539
March	$6,43,436	$17,60,975
April	$6,53,364	$24,14,339
May	$6,59,326	$30,73,665
June	$6,69,989	$37,43,654
July	$4,86,115	$42,29,769
August	$5,36,453	$47,66,221
September	$3,44,063	$51,10,284
October	$4,04,277	$55,14,561
November	$3,26,611	$58,41,172
December	$5,63,762	$64,04,934
Total	**$2,49,14,587**	**$91,85,449**

```
Previous Month Revenue =
    CALCULATE(
        [Total Revenue],
        DATEADD(
            'Calendar Lookup'[Date],
            -1,
            MONTH))
```

DATEADD(**Dates**, NumberOfIntervals, Interval)

Moves the given set of dates by a specified interval.

Year	Total Revenue	YTD Revenue	Previous Month Revenue
2020	**$64,04,934**	**$64,04,934**	**$58,41,172**
January	$5,85,313	$5,85,313	
February	$5,32,226	$11,17,539	$5,85,313
March	$6,43,436	$17,60,975	$5,32,226
April	$6,53,364	$24,14,339	$6,43,436
May	$6,59,326	$30,73,665	$6,53,364
June	$6,69,989	$37,43,654	$6,59,326
July	$4,86,115	$42,29,769	$6,69,989
August	$5,36,453	$47,66,221	$4,86,115
September	$3,44,063	$51,10,284	$5,36,453
October	$4,04,277	$55,14,561	$3,44,063
November	$3,26,611	$58,41,172	$4,04,277
December	$5,63,762	$64,04,934	$3,26,611
Total	**$2,49,14,587**	**$91,85,449**	**$2,30,87,600**

10% growth month on month

```
Revenue Target =
    [Previous Month Revenue] * 1.1
```

Year	YTD Revenue	Previous Month Revenue	Revenue Target
2020	**$64,04,934**	**$58,41,172**	**$64,25,289**
January	$5,85,313		
February	$11,17,539	$5,85,313	$6,43,844
March	$17,60,975	$5,32,226	$5,85,449
April	$24,14,339	$6,43,436	$7,07,780
May	$30,73,665	$6,53,364	$7,18,700
June	$37,43,654	$6,59,326	$7,25,258
July	$42,29,769	$6,69,989	$7,36,988
August	$47,66,221	$4,86,115	$5,34,727
September	$51,10,284	$5,36,453	$5,90,098
October	$55,14,561	$3,44,063	$3,78,469
November	$58,41,172	$4,04,277	$4,44,704
December	$64,04,934	$3,26,611	$3,59,272
Total	**$91,85,449**	**$2,30,87,600**	**$2,53,96,360**

```
10-Day Rolling Revenue =
    CALCULATE(
        [Total Revenue],
        DATESINPERIOD(
            'Calendar Lookup'[Date],
            MAX('Calendar Lookup'[Date]),
            -10,
            DAY))
```

Year	Total Revenue	10-Day Rolling Revenue
1	$8,351	$8,351
2	$14,313	$22,665
3	$28,041	$50,706
4	$17,713	$68,419
5	$7,856	$76,275
6	$21,266	$97,541
7	$8,555	$1,06,096
8	$25,365	$1,31,461
9	$14,313	$1,45,774
10	$14,110	$1,59,884
11	$31,620	$1,83,152
12	$25,048	$1,93,887
13	$7,856	$1,73,701
Total	**$2,49,14,587**	**$6,16,274**

ASSIGNMENT: TIME INTELLIGENCE

Add the following measures to the model:

1. Previous Month Returns
2. Previous Month Orders
3. Previous Month Profit
4. Order Target (10% increase over previous month)
5. Profit Target (10% increase over previous month)
6. 90-day Rolling Profit

```
Previous Month Returns =
    CALCULATE(
        [Total Returns],
        DATEADD(
            'Calendar Lookup'[Date],
            -1,
            MONTH))
```

```
Previous Month Order =
    CALCULATE(
        [Total Orders],
        DATEADD(
            'Calendar Lookup'[Date],
            -1,
            MONTH))
```

```
Previous Month Profit =
    CALCULATE(
        [Total Profit],
        DATEADD(
            'Calendar Lookup'[Date],
            -1,
            MONTH))
```

```
Order Target =
    [Previous Month Order] * 1.1
```

```
Profit Target =
    [Previous Month Profit] * 1.1
```

Year	Total Returns	Previous Month Returns	Total Orders	Order Target	Previous Month Order	Total Profit	Profit Target	Previous Month Profit
			1					
			1					
2020	**85**	**72**	**2,630**	**2,534.40**	**2304**	**$26,01,605**	**26,01,252.67**	**$23,64,775**
January	4		184			$2,35,815		
February	4	4	165	202.40	184	$2,12,187	2,59,396.09	$2,35,815
March	9	4	198	181.50	165	$2,59,085	2,33,405.96	$2,12,187
April	14	9	204	217.80	198	$2,63,032	2,84,993.72	$2,59,085
May	11	14	206	224.40	204	$2,66,276	2,89,335.22	$2,63,032
June	4	11	212	226.60	206	$2,70,068	2,92,904.08	$2,66,276
July	3	4	247	233.20	212	$1,96,683	2,97,075.01	$2,70,068
August	6	3	278	271.70	247	$2,18,355	2,16,351.00	$1,96,683
September	2	6	196	305.80	278	$1,40,516	2,40,190.92	$2,18,355
October	10	2	223	215.60	196	$1,68,582	1,54,567.59	$1,40,516
November	5	10	191	245.30	223	$1,34,176	1,85,439.74	$1,68,582
December	13	5	326	210.10	191	$2,36,830	1,47,593.34	$1,34,176
2021	**764**	**614**	**10,695**	**9,861.50**	**8965**	**$39,67,032**	**38,65,608.27**	**$35,14,189**
January	8	13	242	358.60	326	$1,82,044	2,60,513.09	$2,36,830
February	8	8	267	266.20	242	$2,00,044	2,00,248.51	$1,82,044
March	8	8	266	293.70	267	$1,99,611	2,20,048.42	$2,00,044
April	5	8	290	292.60	266	$2,09,521	2,19,571.76	$1,99,611
May	10	5	329	319.00	290	$2,33,013	2,30,473.48	$2,09,521
June	8	10	312	361.90	329	$2,27,745	2,56,313.91	$2,33,013
July	45	8	506	343.20	312	$3,42,622	2,50,518.97	$2,27,745
Total	**1,809**	**1643**	**25,165**	**25,319.80**	**23018**	**$1,04,57,600**	**1,06,54,638.35**	**$96,86,035**

```
90 - Days Rolling Profit =
    CALCULATE(
        [Total Profit],
        DATESINPERIOD(
            'Calendar Lookup'[Date],
            MAX('Calendar Lookup'[Date]),
            -90,
            DAY))
```

Year	Total Profit	90 - Days Rolling Profit
19	$4,507	$6,02,761
20	$12,825	$6,15,586
21	$5,628	$6,21,214
22	$8,663	$6,29,876
23	$7,105	$6,36,981
24	$17,104	$6,54,085
25	$7,105	$6,61,190
26	$10,502	$6,71,692
27	$4,221	$6,75,913
28	$11,256	$6,87,169
29	$8,512	$6,95,680
30	$5,628	$7,01,308
31		$3,456
April		**$3,456**
Total	**$7,01,308**	**$7,01,308**