# Microsoft Power Tools for Data Analysis #19
# CALCULATE DAX Function, Filter Context, ALLSELECTED & KEEPFILTERS in Excel and Power BI
# Notes from Video:

## Table of Contents

1) **DAX Expression** can be a Measure, a Calculated Column or a DAX Query :

2) **Filter Context, First Look**

   i. How does a DAX Expression calculate or evaluate to get the correct answer?

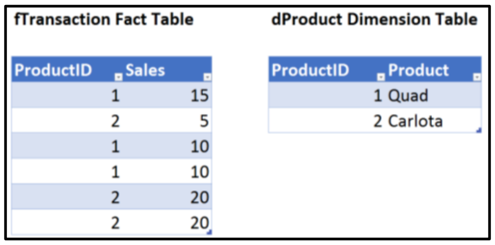
   ii. Simple Answer: **Filter Context**.

   iii. **Filter Context** =

      1. General Definition of Filter Context = Filter the Data Model Tables to get the specific values we need so a DAX Expression can calculate the correct conditional calculation formula answer.
      2. Less General Definition of Filter Context = Conditions / Criteria / Filters (Synonyms) that filter the tables in the Data Model, so the Expression can use the filtered, smaller tables and columns as the source data to make the calculation.

   iv. Simple Example of how DAX Measure evaluates using Filter Context (file used for example::

      1. File name for example: "019-MSPTDA-CALCULATE-FilterContext-01-Start.xlsx"
      2. Tables:



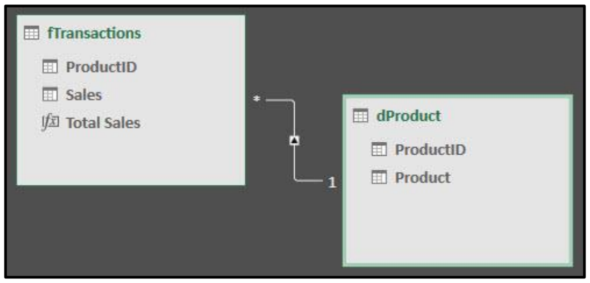
**fTransaction Fact Table**

| ProductID | Sales |
|---|---|
| 1 | 15 |
| 2 | 5 |
| 1 | 10 |
| 1 | 10 |
| 2 | 20 |
| 2 | 20 |

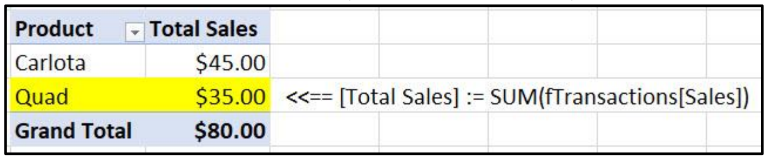
**dProduct Dimension Table**

| ProductID | Product |
|---|---|
| 1 | Quad |
| 2 | Carlota |

      3. Data Model:

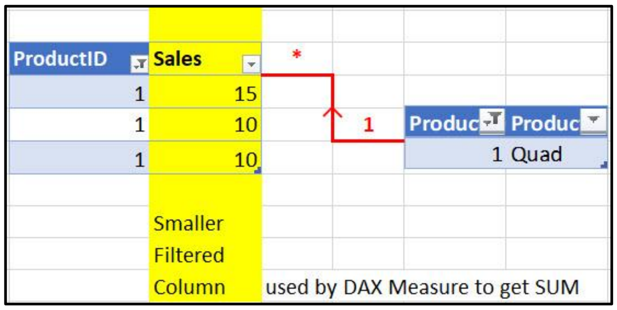


      4. Measure in Excel PivotTable (or Power BI Visual):



| Product | Total Sales | |
|---|---|---|
| Carlota | $45.00 | |
| Quad | $35.00 | <<== [Total Sales] := SUM(fTransactions[Sales]) |
| Grand Total | $80.00 | |

      5. "Quad" Condition in Row Area of PivotTable filters the dProduct Table, and then the filter flows across the Relationship to filter the Fact Table down to just the rows it needs to calculate the Total Sales:



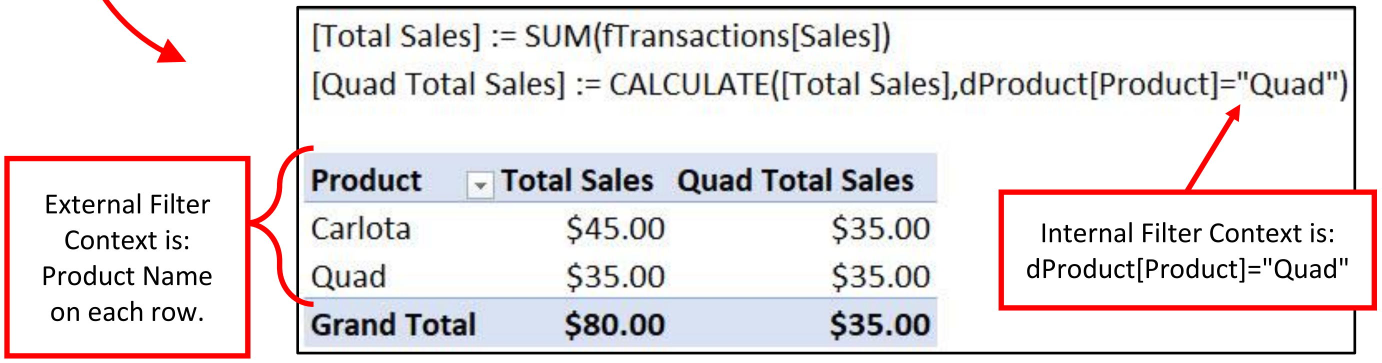
| ProductID | Sales |
|---|---|
| 1 | 15 |
| 1 | 10 |
| 1 | 10 |

| Produc | Produc |
|---|---|
| 1 | Quad |

      6. The Final Filter Context under which the DAX Measure calculates is the filtered Sales column that contains the numbers: 15, 10, 10.

3) **Internal Filter Context, External Filter Context and Final Filter Context**. When DAX Measures are calculated in an Excel PivotTable or Power BI Visual, there are potentially two inputs into Final Filter Context that filters tables in the Data Model:
   i. Internal Filter Context: Conditions/Criteria/Filters from inside the DAX Measure.
   ii. External Filter Context: Row and Column and Filter and Slicer Conditions/Criteria/Filters from the Excel PivotTable or the Power BI Visual.

4) **A simple Example of Internal Filter Context and External Filter Context** working together to construct the **Final Filter Context** under which the DAX Measure can make its calculation is seen below.
   i. Using the same Data Model as in the previous example, the PivotTable with the [Total Sales] and [Quad Total Sales] Measures.
   ii. For the [Quad Total Sales], although we have a PivotTable Row Area condition or criteria trying to filter the underlying Fact Table for each row in the PivotTable, the internal filter of "Quad" overwrites the external filter and so each cell in the PivotTable calculate the Total Sales for the Quad Product.
   iii. We will learn much more about the process of merging the internal filters and external filters, but in this first simple example we can see that filters for the tables in the Data Model can come from the external report or from internally, inside the DAX Measure.



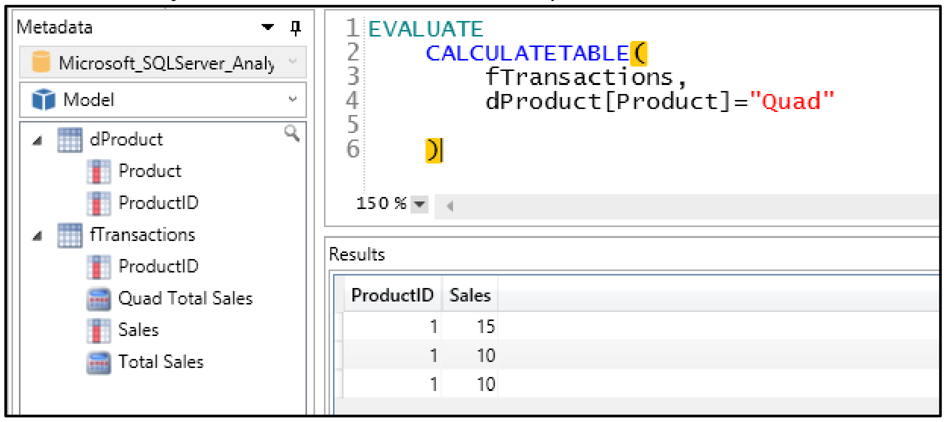
```
[Total Sales] := SUM(fTransactions[Sales])
[Quad Total Sales] := CALCULATE([Total Sales],dProduct[Product]="Quad")
```

| Product | Total Sales | Quad Total Sales |
|---|---|---|
| Carlota | $45.00 | $35.00 |
| Quad | $35.00 | $35.00 |
| **Grand Total** | **$80.00** | **$35.00** |

External Filter Context is: Product Name on each row.
Internal Filter Context is: dProduct[Product]="Quad"

5) **A simple example of just Internal Filter Context** in a DAX Table Function in Query can be seen below:

   i. Using DAX Studio (or Power BI Desktop like in the video) and connecting to the same Data Model as in the previous example, as seen in the below picture for a Query we can use the Internal Filter in the DAX Table Function CALCULATETABLE to filter the Fact Table to show just records for the product "Quad". In this example the CALCULATETABLE function starts with "No External Filter Context" and then uses an Internal Filter for "Quad" on the dProduct[Product] column and then propagates it to the fTransactions table to show just the records for the "Quad" product.



```
EVALUATE
    CALCULATETABLE(
        fTransactions,
        dProduct[Product]="Quad"

    )
```

| ProductID | Sales |
|---|---|
| 1 | 15 |
| 1 | 10 |
| 1 | 10 |

6) **How can we change the Filter Context?**

   i. The CALCULATE and CALCULATETABLE functions are the DAX functions that we use to change the Filter Context (there are a few others also, like TOTALYTD).

This section has all details about CALCULATE & although you might want to read through it now, it will make more sense after you watch the whole video & read the rest of these pdf notes : )

7) **Details about the CALCULATE and CALCULATETABLE DAX Functions**:
  - i. The CALCULATE and CALCULATETABLE DAX Functions can change the Filter Context.
  - ii. Some of the ways that these functions can change the Filter Context:
    1. For a Measure in a report or visualization, it can change the Filter Context by merging the External & Internal Filter Contexts.
    2. For a DAX Table formula or a DAX Query, it can change no Filter Context into a Filter Context by using an internal filter from inside the CALCULATE function.
    3. In a Calculated Column or Iterator Function, CALCULATE can change no Filter Context into a Filter Context by converting all available Row Contexts into an Equivalent Filter Context.
  - iii. Arguments for CALCULATE:
    **CALCULATE(Scalar Expressions, Filter1, Filter2…)**
  - iv. Arguments for CALCULATETABLE:
    **CALCULATETABLE(Table Expressions, Filter1, Filter2…)**
  - v. CALCULATE and CALCULATETABLE both work the same and so when I use the function name "CALCULATE" throughout the rest of this pdf handout, I will be referring to both the CALCULATE and CALCULATETABLE functions.
  - vi. Internal Filters in CALCULATE:
    1. In the CALCULATE filter arguments, we can use any column from our Star Schema Data Model. You can think of the CALCULATE function as being able to see the entire Data Model and filter columns which can then flow across relationships.
    2. The Filter argument will consist of valid lists of values which will serve as the conditions or criteria for filtering (either from a Table Function or Boolean Logical Test Filter).
    3. The Filter1, Filter2… arguments in CALCULATE work together in an AND Logical Test.
    4. Two types of filters:
      - i. Boolean Logical Test Filter, like dProduct[Product]="Quad" (this is a valid list of a single product, "Quad").
      - ii. A Table expression that delivers a valid list of values. Like these DAX functions: FILTER, SAMEPERIODLASTYEAR & DATESADD. Also, Table Filters (as we will see later).
    5. In a single Filter argument, we can run logical tests like:
      - i. AND Logical Test (&& or AND function)
        - 1) Examples:
          - i. AND(fTransactions[Sales]>=10, fTransactions[Sales]<20)
          - ii. fTransactions[Sales]>=10 && fTransactions[Sales]<20
      - ii. OR Logical Test (|| or OR function)
        - 1) Examples:
          - i. OR(dProduct[Product]="Quad", dProduct[Product]="Carlota")
          - ii. dProduct[Product]="Quad" || dProduct[Product]="Carlota"
      - iii. NOT Logical Test(! Or NOT function)
        - 1) Examples:
          - i. NOT(dProduct[Product]="Quad")
          - ii. ! dProduct[Product]="Quad"
    6. A Boolean Logical Test Filter can contain only a single column such as:
      - i. Examples:
        - 1) dProduct[Product]="Quad"
        - 2) fTransactions[Sales]>=10 && fTransactions[Sales]<20
        - 3) dProduct[Product]="Quad" || dProduct[Product]="Carlota"

7. A Boolean expression cannot work on two different columns
    i. Not Allowed because we are using a Boolean Filter:
        1) dProduct[Price]>=dProduct[Cost]*2
    ii. Allowed because we are using a Table Function:
        1) FILTER(ALL(dProduct[Price],dProduct[Cost]),dProduct[Price]>=dProduct[Cost]*2))
    iii. For a Boolean we can not use two different columns because internally the engine does not know which of the two columns it should iterate over, and/or which of any possible external columns it should replace in the Overwrite Operation.
    iv. More explanation a few pages ahead in the "Boolean Filter Restrictions section".

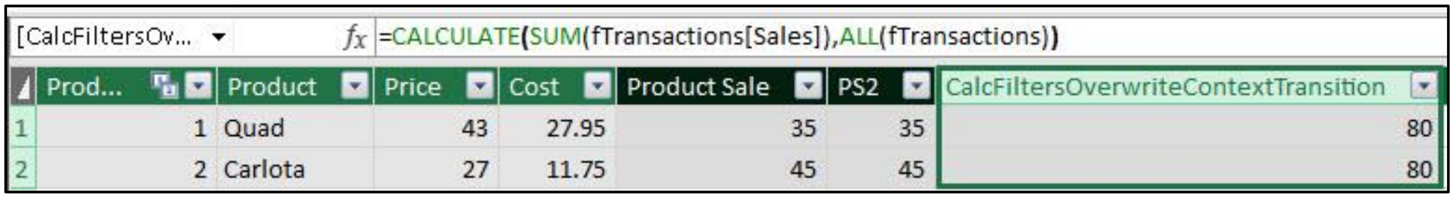
vii. How Internal Filter Context and External Filter Context are Merged to get the Final Filter Context:

1. CALCULATE takes the internal filters from the Filter arguments in CALCULATE and the external filters from the Excel PivotTable or Power BI Visual, and if the same column is used in the External Filter Context and Internal Filter Context, the internal column/s replaces the external column/s. Specifically, the external column/s are removed, the internal column/s remain, then an AND Logical Test is used to merge the External Filter Context and Internal Filter Context, to get the Final Filter Context that the Measure calculates under. Multiple examples are presented later in the pdf notes.
2. Operators used in CALCULATE:
    i. AND Logical Test (Intersect):
        1) AND Logical Test works between the filters in the Internal Filter Context
        2) AND Logical Test works between the filters in the External Filter Context
        3) AND Logical Test will work when it merges the External Filter Context and Internal Filter Context to get the Final Filter Context.
    ii. Overwrite:
        1) If a column used in the filtering exists in both the External Filter Context and Internal Filter Context, the external column is removed from the External Filter Context leaving only the internal column in the Internal Filter Context.
            i. For example, if the external filter was dProduct[Product]="Carlota" and the internal filter was dProduct[Product]="Quad", the external filter of dProduct[Product]="Carlota" is removed leaving an empty filter and the internal filter of dProduct[Product]="Quad" remains.
        2) The External Filter Context and Internal Filter Context are Merged using an AND Logical Test to create the Final Filter Context under which the Measure makes its calculation.
    iii. ALL Remove Operator:
        1) The "ALL Remove" Operator is an operator that is created when we use the ALL Function inside a Filter argument of the CALCULATE function. If we use the ALL function on a column, columns or on a table, all filters on the column, columns or table will be removed and an empty filter will be left in place of the column, columns or table. For example, the formula CALCULATE([Total Sales],ALL(fTransactions)) would remove all filters from all columns in the fTransaction table, leaving all rows for the [Total Sales] calculation and yielding the Grand Total for all transactions.

viii. The CALCULATE Function always determines the Final Filter Context BEFORE it sends the filter to the tables in the Data Model and then calculates the final answer for the first argument. This means that the first argument in CALCULATE is always evaluated last, after the filters are used to determine the Final Filter Context.

ix. CALCULATE performs Context Transition, which means:

   1. All available Row Contexts will be converted to an Equivalent Filter Context ("all available" means that if there are multiple Row Contexts, one after the other, all of those Row Contexts will be converted to an Equivalent Filter Context in an AND Logical Test.
      i. As an example, if you have an aggregate formula like, SUM(fTransactions), in a Calculated Column and you want the table inside the SUM function to be filtered based on a condition from each row in the table, you can wrap the CALCULATE function around the SUM to perform Context Transition thereby converting the Row Context into Filter Context and filtering the table inside the SUM function.
   2. Note: Columns used in a filter inside CALCULATE will override any columns coming from Context Transition. Because the columns are inside CALCULATE, and Row Context filters are coming from outside, or from an External Filter, the Overwrite processes will replace the columns from the External Filter Context created by Context Transition.
      i. Examples:
         1) When you use a formula like CALCULATE(SUM(fTransactions[Sales])) in a Calculated Column in a Product table that has a One-To-Many Relationship with the Fact table you will get the sum of all sales for each product. In the below picture this is the formula we used in the Product Sale Column.
         2) When you use a formula like CALCULATE(SUM(fTransactions[Sales]),ALL(fTransactions)) in a Calculated Column in a Product table that has a One-To-Many Relationship with the Fact table you will get the sum of all sales from all records in the Fact table. This is because the ALL function removes all filters from the fTransactions table and this filter will overwrite the Filter Context for each product in each row of the Calculated Column created by Context Transition.
      ii. Picture from (not seen in video but is in the downloadable finished file named "019-MSPTDA-CALCULATE-FilterContext-01-Finished.xlsx":



[CalcFiltersOv... fx =CALCULATE(SUM(fTransactions[Sales]),ALL(fTransactions))

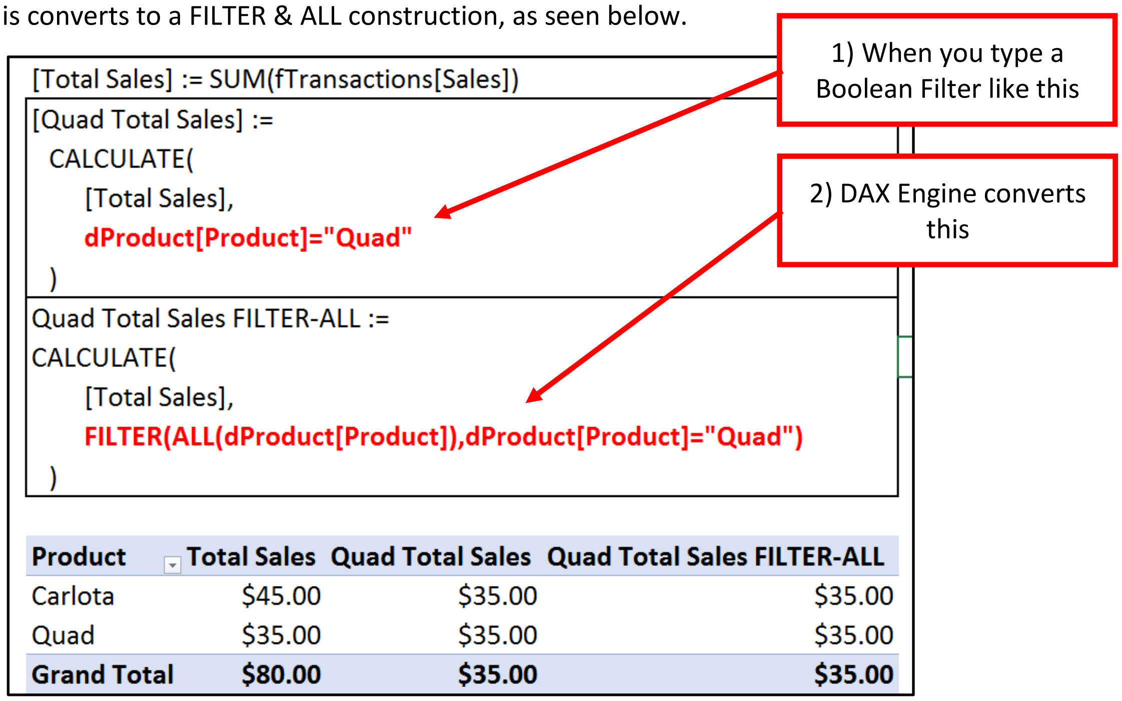
| | Prod... | Product | Price | Cost | Product Sale | PS2 | CalcFiltersOverwriteContextTransition |
|---|---|---|---|---|---|---|---|
| 1 | 1 | Quad | 43 | 27.95 | 35 | 35 | 80 |
| 2 | 2 | Carlota | 27 | 11.75 | 45 | 45 | 80 |

x. All Measures have a hidden CALCULATE Function that performs Context Transition.

xi. In a Star Schema Data Model, when you use the ALL Function on a Fact Table and place it in the Filter Argument of the CALCULATE, you will remove all filters placed on all tables in the Data Model.

8) **What is a Boolean Logical Test Filter?**
   i. "Boolean Logical Test Filter", or just "Boolean Filter" means that you use a single column, a comparative operator and a condition, like: dProduct[Product]="Quad".

9) **Boolean Filter is always converted to a FILTER & ALL DAX Function construction** :
   i. When you type a Boolean Filter in the CALCULATE function, internally in the DAX Engine, the Boolean Filter is converts to a FILTER & ALL construction, as seen below.



```
[Total Sales] := SUM(fTransactions[Sales])
[Quad Total Sales] :=
  CALCULATE(
      [Total Sales],
      dProduct[Product]="Quad"
  )
Quad Total Sales FILTER-ALL :=
CALCULATE(
      [Total Sales],
      FILTER(ALL(dProduct[Product]),dProduct[Product]="Quad")
  )
```

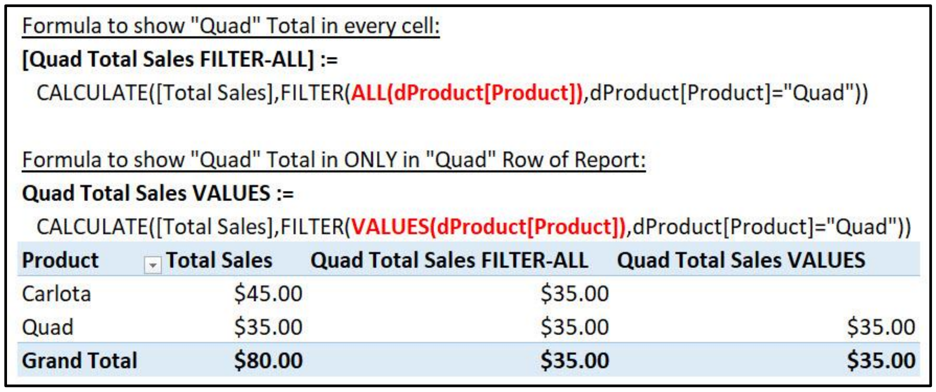
| Product | Total Sales | Quad Total Sales | Quad Total Sales FILTER-ALL |
|---|---|---|---|
| Carlota | $45.00 | $35.00 | $35.00 |
| Quad | $35.00 | $35.00 | $35.00 |
| **Grand Total** | **$80.00** | **$35.00** | **$35.00** |

1) When you type a Boolean Filter like this
2) DAX Engine converts this

   ii. How the FILTER-ALL Combination works in the above picture:
      1. In the 1st argument of the FILTER function, the ALL Function returns a unique list of the all the Products as a table of values.
         i. The ALL Function does not see the External Filter Context from the Product column in the Row Area of the PivotTable and therefore will return a complete unique list in each cell of the PivotTable (this is why the formula returns the same "Quad" Total in each cell).
         ii. When the ALL function is used in the first argument of FILTER, it delivers a table of values (this is an important distinction because later we will see that when you use the ALL function in a Filter argument of CALCULATE it works as a remove operator).
      2. The Condition "Quad" is compared to each row in the ALL Table, iterating over each row.
      3. Only the "Quad" Row gets a TRUE.
      4. The FILTER Table returns a single row table that contains the condition "Quad".
   iii. How the CALCULATE Functions uses the "Quad" Condition to filter rows in the Fact Table in the above picture:
      1. The external filter is the particular product in the Row Area of the PivotTable from the dProduct[Product] column.
      2. The Internal Filter Context is the "Quad" product from the dProduct[Product] column.
      3. In the Overwrite process:
         i. The external filter on the dProduct[Product] column is removed resulting in an empty filter
         ii. The internal filter on the dProduct[Product] column remains as dProduct[Product]="Quad".
         iii. An AND Test in run on the External Filter Context and Internal Filter Context to create the Final Filter Context is: dProduct[Product]="Quad".
      4. The Quad Filter flows across the relationship from the Dimension Table, dProduct, to the Fact Table, fTransactions, and filters the Fact table down to just the rows with the "Quad" Product.
      5. The **[Total Sales] Measure** calculates or evaluates using the filtered fTransactions Sales Column.

10) **VALUES rather than ALL in first argument of FILTER** :

i. As seen in video, when we use VALUES rather than ALL in first argument of FILTER, we get an amount only in the row that contains the condition because the VALUES function can see the External Fiulter Context, but the ALL Function does not. Here is a picture from the video:



Formula to show "Quad" Total in every cell:

**[Quad Total Sales FILTER-ALL] :=**

CALCULATE([Total Sales],FILTER(**ALL(dProduct[Product])**,dProduct[Product]="Quad"))

Formula to show "Quad" Total in ONLY in "Quad" Row of Report:

**Quad Total Sales VALUES :=**

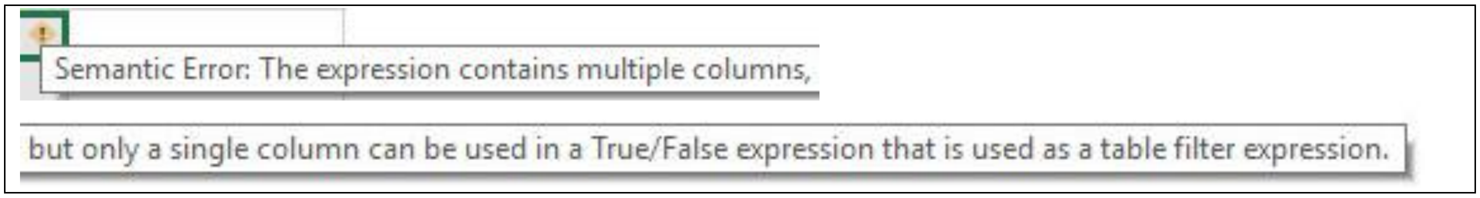
CALCULATE([Total Sales],FILTER(**VALUES(dProduct[Product])**,dProduct[Product]="Quad"))

| Product | Total Sales | Quad Total Sales FILTER-ALL | Quad Total Sales VALUES |
|---|---|---|---|
| Carlota | $45.00 | $35.00 | |
| Quad | $35.00 | $35.00 | $35.00 |
| **Grand Total** | **$80.00** | **$35.00** | **$35.00** |

11) **Boolean Condition Restrictions** :

i. Only a Single Column can be used in a Boolean Filter in the Filter argument of CALCULATE.

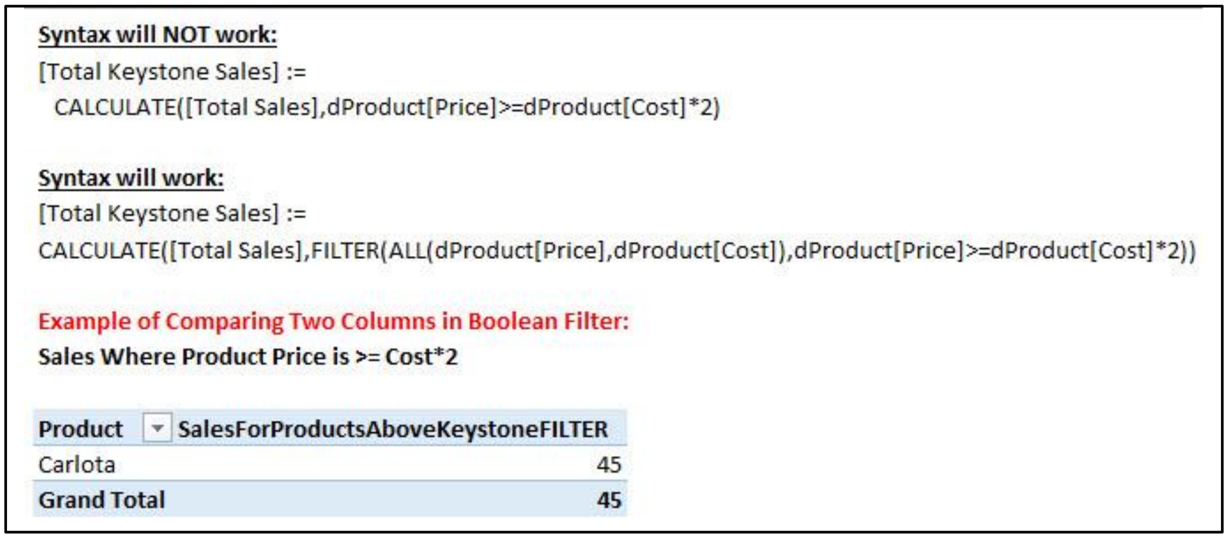
1. The syntax of directly comparing two columns in a Boolean Statement is not allowed by the DAX Engine. This is the error message that you will get if you try:



Semantic Error: The expression contains multiple columns,

but only a single column can be used in a True/False expression that is used as a table filter expression.

1. Reasons we can only use a single column in a Boolean Logical Test Filter:
   i. All Boolean Logical Test Filter are converted to a single column FILTER and ALL construction and the DAX engine must detect a single column that the FILTER function can iterate.
   ii. If you try to use two different columns in a single CALCULATE Filter argument and the same column/s that are used in the Boolean Filter are also in the External Filter Context, the engine does not know which ones to replace: one, the other, or both?
2. If you have two or more columns used in your Boolean Statement, instead of authoring the filter as a Boolean Statement, you must use Table functions (like FILTER) to construct the filter.
3. Example as seen in video is below:



**Syntax will NOT work:**

[Total Keystone Sales] :=

CALCULATE([Total Sales],dProduct[Price]>=dProduct[Cost]*2)

**Syntax will work:**

[Total Keystone Sales] :=

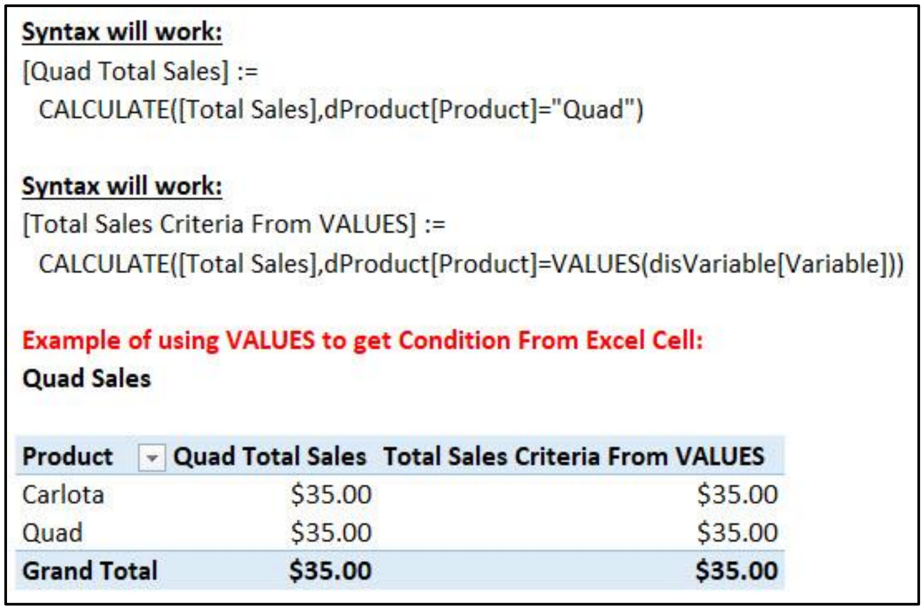
CALCULATE([Total Sales],FILTER(ALL(dProduct[Price],dProduct[Cost]),dProduct[Price]>=dProduct[Cost]*2))

**Example of Comparing Two Columns in Boolean Filter:**

**Sales Where Product Price is >= Cost*2**

| Product | SalesForProductsAboveKeystoneFILTER |
|---|---|
| Carlota | 45 |
| **Grand Total** | **45** |

ii. The Condition for a Boolean Filter can be a Literal or the VALUES function :
   1. The VALUES can be used to get a variable from a disconnected table (no relationship with any other table) and convert it to a scalar value so it can be used as a condition in a Boolean Filter.
   2. Example as seen in video:



**Syntax will work:**

```
[Quad Total Sales] :=
  CALCULATE([Total Sales],dProduct[Product]="Quad")
```

**Syntax will work:**

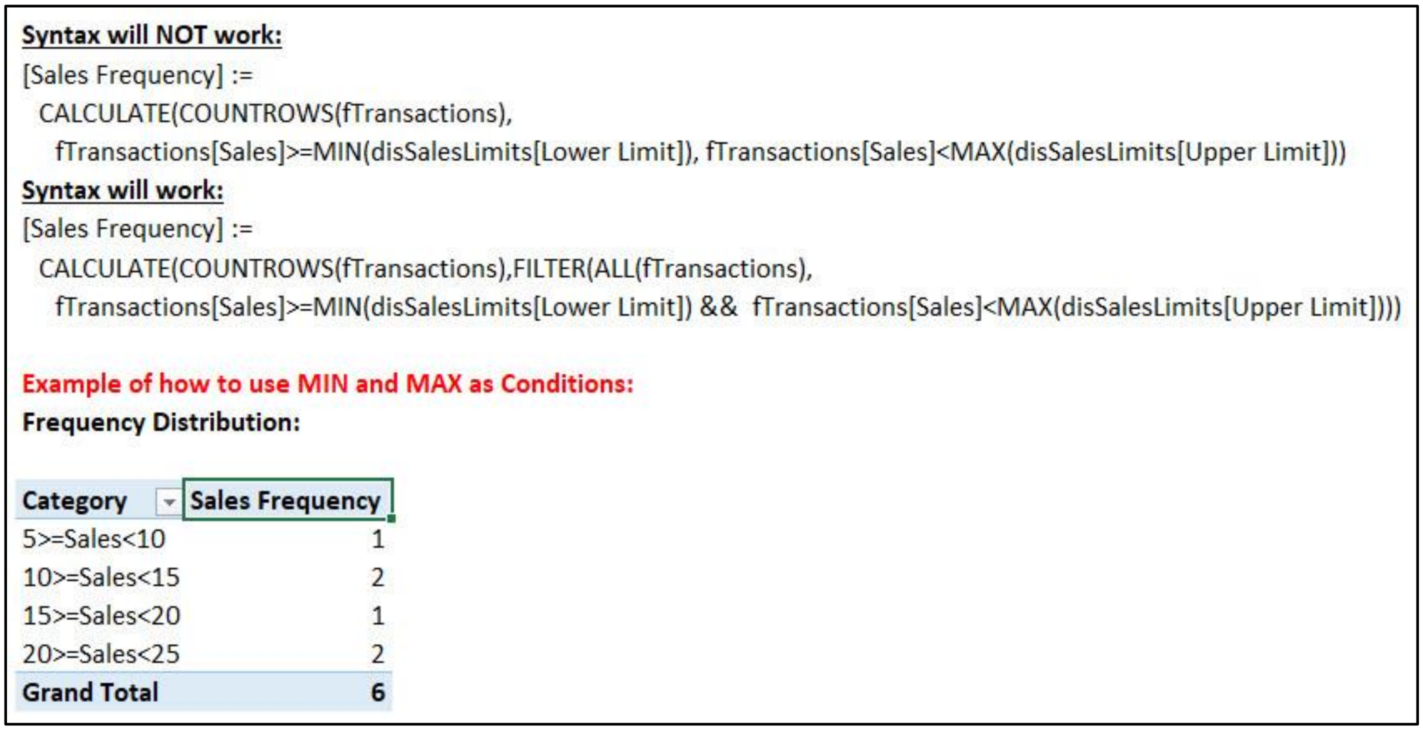
```
[Total Sales Criteria From VALUES] :=
  CALCULATE([Total Sales],dProduct[Product]=VALUES(disVariable[Variable]))
```

**Example of using VALUES to get Condition From Excel Cell:**

**Quad Sales**

| Product | Quad Total Sales | Total Sales Criteria From VALUES |
|---|---|---|
| Carlota | $35.00 | $35.00 |
| Quad | $35.00 | $35.00 |
| **Grand Total** | **$35.00** | **$35.00** |

iii. Results from aggregate functions like MIN and MAX functions are not allowed as conditions for Boolean Filters. However, functions like MIN and MAX can be used in the FILTER Function.
   1. In the video we create a Frequency Distribution to count how many sales there were between an upper and lower limit. We use a Disconnected Category Table names “disSalesLimits”. Here is a picture of the formulas we created in the video:



**Syntax will NOT work:**

```
[Sales Frequency] :=
  CALCULATE(COUNTROWS(fTransactions),
    fTransactions[Sales]>=MIN(disSalesLimits[Lower Limit]), fTransactions[Sales]<MAX(disSalesLimits[Upper Limit]))
```

**Syntax will work:**

```
[Sales Frequency] :=
  CALCULATE(COUNTROWS(fTransactions),FILTER(ALL(fTransactions),
    fTransactions[Sales]>=MIN(disSalesLimits[Lower Limit]) &&  fTransactions[Sales]<MAX(disSalesLimits[Upper Limit])))
```

**Example of how to use MIN and MAX as Conditions:**

**Frequency Distribution:**

| Category | Sales Frequency |
|---|---|
| 5>=Sales<10 | 1 |
| 10>=Sales<15 | 2 |
| 15>=Sales<20 | 1 |
| 20>=Sales<25 | 2 |
| **Grand Total** | **6** |

11) **AND Logical Test and OR Logical Test Boolean Formulas we saw in the Video** :

i. AND Logical Tests:

```
[Sales Between 10 and 20 One Argument] :=
CALCULATE([Total Sales],fTransactions[Sales]>=10 && fTransactions[Sales]<25)

[Sales Between 10 and 20 Two Arguments] :=
CALCULATE([Total Sales],fTransactions[Sales]>=10,fTransactions[Sales]<25)

[Sales Between 10 and 20 AND] :=
CALCULATE([Total Sales],AND(fTransactions[Sales]>=10, fTransactions[Sales]<25))

[Carlota Sales >15] :=
CALCULATE([Total Sales],dProduct[Product]="Carlota", fTransactions[Sales]>15)
```

ii. OR Logical Tests:

```
[Freestyle Boom Sales] :=
CALCULATE([Total Sales],dProduct[Product]="Quad" || dProduct[Product]="Carlota")

[Freestyle Boom Sales OR] :=
CALCULATE([Total Sales],OR(dProduct[Product]="Quad", dProduct[Product]="Carlota"))
```

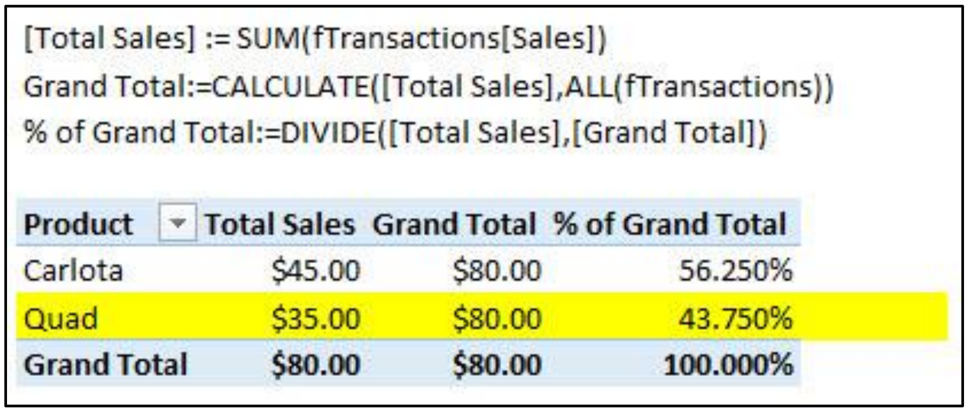
12) **ALL DAX Function & Grand Totals**. When you use the ALL DAX function in the Filter argument of CALCULATE, ALL DAX function will remove filters.

i. Review of what you are allowed to use in the ALL DAX Table function:
   1. ALL(column)
   2. ALL(column1,column2…)
   3. ALL(table)

ii. When you use the ALL DAX function in the Filter argument of CALCULATE, the ALL DAX function will remove filters from the column, columns or tables that sit in the ALL function. You can think of the ALL function in the Filter argument of CALCULATE as the "Remove Operator".

iii. Using the ALL DAX Function on the Fact Table can be useful when you need to calculate the Grand Total Amount for every cell in the Excel PivotTable or Power BI Visual. This Grand Total can then be used as the denominator to calculate a "% of Grand Total" formula.

iv. Example of ALL as the "ALL Remove Operator":

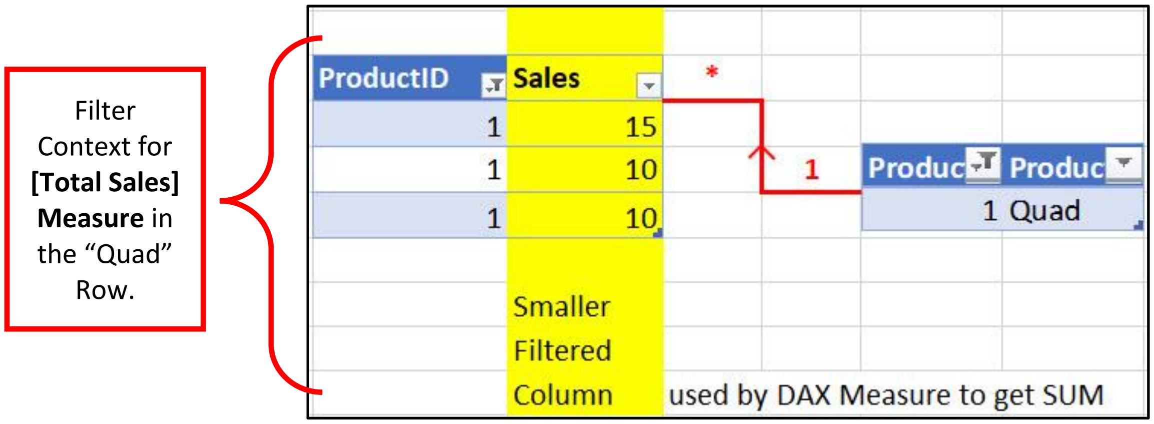


```
[Total Sales] := SUM(fTransactions[Sales])
Grand Total:=CALCULATE([Total Sales],ALL(fTransactions))
% of Grand Total:=DIVIDE([Total Sales],[Grand Total])
```

| Product | Total Sales | Grand Total | % of Grand Total |
|---|---|---|---|
| Carlota | $45.00 | $80.00 | 56.250% |
| Quad | $35.00 | $80.00 | 43.750% |
| **Grand Total** | **$80.00** | **$80.00** | **100.000%** |

v. The steps for how the ALL DAX function works as a remove operator in CALCULATE for the second row in the PivotTable shown above (the row with "Quad" in the Row Area of the PivotTable) are listed here:
   1. First, consider how the **[Total Sales] Measure** is calculated for the "Quad" row. When the **[Total Sales] Measure** sits in the "Quad" row, the Final Filter Context on the Fact Table looks like this:

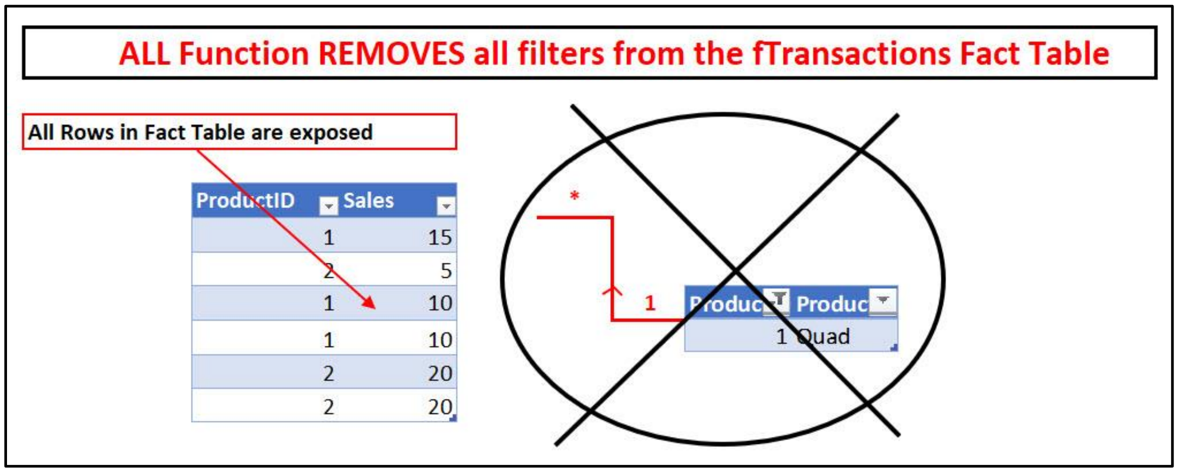
Filter Context for **[Total Sales] Measure** in the "Quad" Row.



| ProductID | Sales |
|---|---|
| 1 | 15 |
| 1 | 10 |
| 1 | 10 |

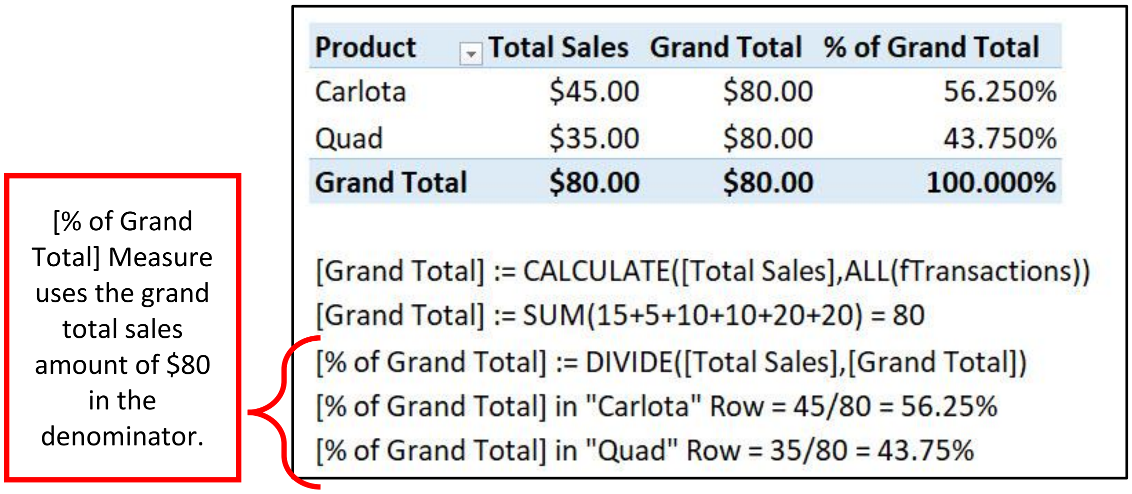
| Produc | Produc |
|---|---|
| 1 | Quad |

   2. But when the **[Grand Total] Measure** sits in the "Quad" row of the PivotTable, the "Quad" Condition tries to filter the Fact Table as seen in the previous picture, but the ALL(fTransaction) formula element "REMOVES" all filters from the Fact Table so that when the **[Grand Total Measure]** is calculated, the Final Filter Context on the Fact Table looks like this (picture on next page):


ALL Function REMOVES all filters from the fTransactions Fact Table
All Rows in Fact Table are exposed
ProductID
Sales
*
1
Produc
Produc
1 Quad

| ProductID | Sales |
|---|---|
| 1 | 15 |
| 2 | 5 |
| 1 | 10 |
| 1 | 10 |
| 2 | 20 |
| 2 | 20 |

3. In the Formula, The ALL Function Removes all the filters on the Fact Table and the CALCULATE allows that change in filtering to take effect on the Measure. The ALL function "Removes" filters and then the CALCULATE Function changes the Filter Context so that all rows in the Fact Table Sales Column can be used to calculate the Grand Total sales amount. In the below picture, we can see that we can use the Grand Total Measure as the denominator to calculate the **[% of Grand Total] Measure**:



| Product | Total Sales | Grand Total | % of Grand Total |
|---|---|---|---|
| Carlota | $45.00 | $80.00 | 56.250% |
| Quad | $35.00 | $80.00 | 43.750% |
| **Grand Total** | **$80.00** | **$80.00** | **100.000%** |

[Grand Total] := CALCULATE([Total Sales],ALL(fTransactions))
[Grand Total] := SUM(15+5+10+10+20+20) = 80
[% of Grand Total] := DIVIDE([Total Sales],[Grand Total])
[% of Grand Total] in "Carlota" Row = 45/80 = 56.25%
[% of Grand Total] in "Quad" Row = 35/80 = 43.75%

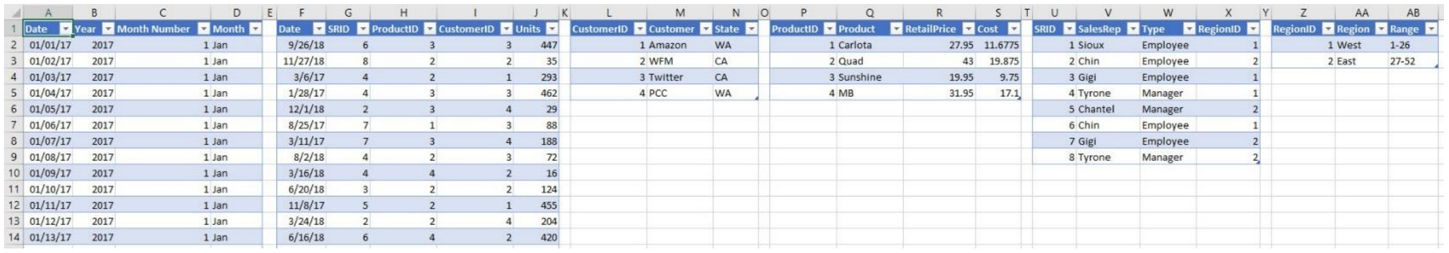
[% of Grand Total] Measure uses the grand total sales amount of $80 in the denominator.

13) **Hidden Context Transition for Measures in an Excel PivotTable or Power BI Visual**:.
   i. When a Measure is dropped into a PivotTable or Power BI Visual, in order for the conditions from the Row, Column and Filter Areas to flow into the Measure, the "Row Context" Condition must be converted to "Filter Context" so that the tables can be filtered and the Measure can calculate the final answer. You can think of the rows in a PivotTable or Power BI Visual as rows that the Measure needs to iterate in order to pick out the correct conditions for the Measure to calculate the final answer. The mechanism for accomplishing this is "Context Transition".
   ii. Any Measure in a PivotTable or Power BI Visual performs Context Transition (Row Context into Equivalent Filter Context) so that the conditions from the Row, Column, Filter, and Slicer from each cell (Row Context) can flow into the Measure, then filter the Data Model tables, and the Measure can calculate the final answer.
   iii. When you drop a Measure into a PivotTable or Power BI Visuals that has a Grand Total Row, two different processes occur:
      1. The Row or Column Area of Pivot / Power BI Report calculates as an Iterator with Context Transition.
      2. The Grand Total Cell Calculates without a hidden Context Transition and Iteration.
   iv. The importance of knowing about the Context Transition in an Excel PivotTable or Power BI Visual will become obvious when we study the ALLSELECTED DAX Function.

14) **Step-By-Step Example of the CALCULATE Overwrite Operator**:

1] File name for example: “019-MSPTDA-CALCULATE-FilterContext-02-Finished.xlsx”

2] Tables shown here:



| Date | Year | Month Number | Month |
|---|---|---|---|
| 01/01/17 | 2017 | 1 | Jan |
| 01/02/17 | 2017 | 1 | Jan |
| 01/03/17 | 2017 | 1 | Jan |
| 01/04/17 | 2017 | 1 | Jan |
| 01/05/17 | 2017 | 1 | Jan |
| 01/06/17 | 2017 | 1 | Jan |
| 01/07/17 | 2017 | 1 | Jan |
| 01/08/17 | 2017 | 1 | Jan |
| 01/09/17 | 2017 | 1 | Jan |
| 01/10/17 | 2017 | 1 | Jan |
| 01/11/17 | 2017 | 1 | Jan |
| 01/12/17 | 2017 | 1 | Jan |
| 01/13/17 | 2017 | 1 | Jan |

| Date | SRID | ProductID | CustomerID | Units |
|---|---|---|---|---|
| 9/26/18 | 6 | 3 | 3 | 447 |
| 11/27/18 | 8 | 2 | 2 | 35 |
| 3/6/17 | 4 | 2 | 1 | 293 |
| 1/28/17 | 4 | 3 | 3 | 462 |
| 12/1/18 | 2 | 3 | 4 | 29 |
| 8/25/17 | 7 | 1 | 3 | 88 |
| 3/11/17 | 7 | 3 | 4 | 188 |
| 8/2/18 | 4 | 2 | 3 | 72 |
| 3/16/18 | 4 | 4 | 2 | 16 |
| 6/20/18 | 3 | 2 | 2 | 124 |
| 11/8/17 | 5 | 2 | 1 | 455 |
| 3/24/18 | 2 | 2 | 4 | 204 |
| 6/16/18 | 6 | 4 | 2 | 420 |

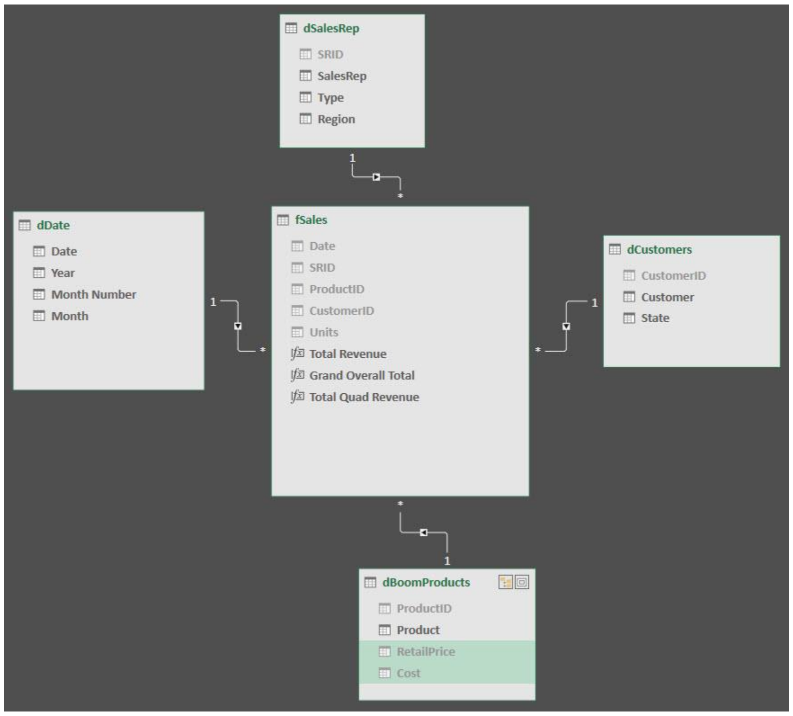
| CustomerID | Customer | State |
|---|---|---|
| 1 | Amazon | WA |
| 2 | WFM | CA |
| 3 | Twitter | CA |
| 4 | PCC | WA |

| ProductID | Product | RetailPrice | Cost |
|---|---|---|---|
| 1 | Carlota | 27.95 | 11.6775 |
| 2 | Quad | 43 | 19.875 |
| 3 | Sunshine | 19.95 | 9.75 |
| 4 | MB | 31.95 | 17.1 |

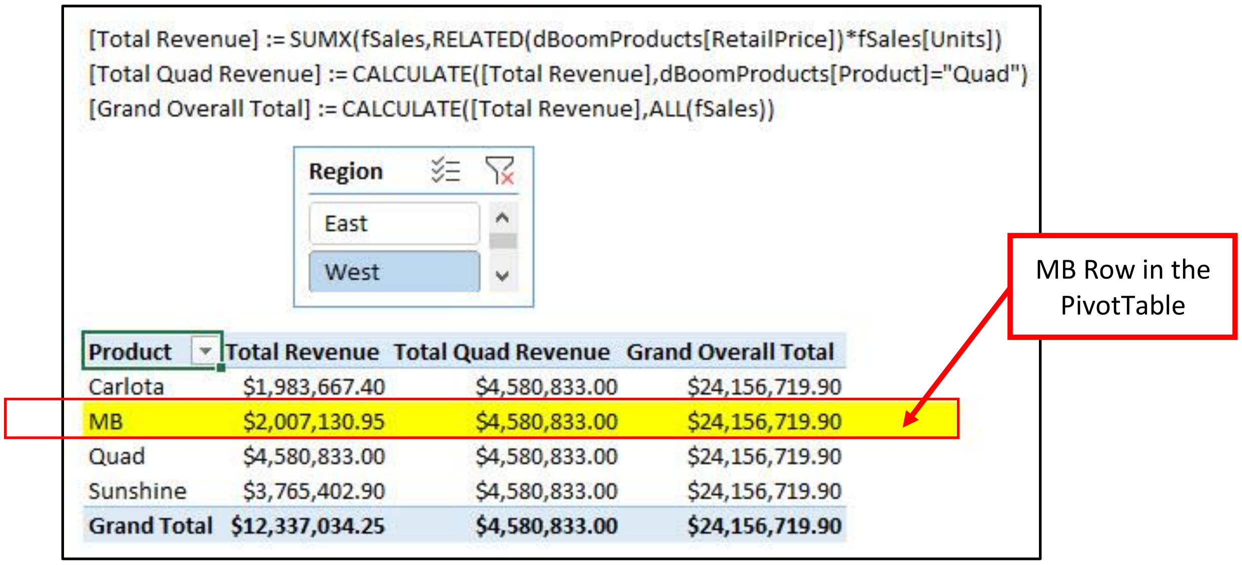
| SRID | SalesRep | Type | RegionID |
|---|---|---|---|
| 1 | Sioux | Employee | 1 |
| 2 | Chin | Employee | 2 |
| 3 | Gigi | Employee | 1 |
| 4 | Tyrone | Manager | 1 |
| 5 | Chantel | Manager | 2 |
| 6 | Chin | Employee | 1 |
| 7 | Gigi | Employee | 2 |
| 8 | Tyrone | Manager | 2 |

| RegionID | Region | Range |
|---|---|---|
| 1 | West | 1-26 |
| 2 | East | 27-52 |

3] New Data Model shown here:



4] Measures in Excel PivotTable Example with Product on the Row Area of the PivotTable and a Slicer with the Region “West” selected:



```
[Total Revenue] := SUMX(fSales,RELATED(dBoomProducts[RetailPrice])*fSales[Units])
[Total Quad Revenue] := CALCULATE([Total Revenue],dBoomProducts[Product]="Quad")
[Grand Overall Total] := CALCULATE([Total Revenue],ALL(fSales))
```

| Product | Total Revenue | Total Quad Revenue | Grand Overall Total |
|---|---|---|---|
| Carlota | $1,983,667.40 | $4,580,833.00 | $24,156,719.90 |
| MB | $2,007,130.95 | $4,580,833.00 | $24,156,719.90 |
| Quad | $4,580,833.00 | $4,580,833.00 | $24,156,719.90 |
| Sunshine | $3,765,402.90 | $4,580,833.00 | $24,156,719.90 |
| **Grand Total** | **$12,337,034.25** | **$4,580,833.00** | **$24,156,719.90** |

5] For the MB Row in the PivotTable in the above picture, here is how the Final Filter Context is determined for the **[Total Quad Revenue] Measure**:

1) The external and internal filters for the MB Row in the PivotTable are:

    1. Internal filter = dBoomProducts[Product]="Quad"
    2. External filter = dBoomProducts[Product]="MB" AND dSalesRep[Region]="West"

2) OVERWRITE Process to get Final Filter Context:

    Because the **dBoomProducts[Product]** column is used in both the internal and external filters, internal wins! This happens:

    dBoomProducts[Product]="Quad"   **OVERWRITES**   dBoomProducts[Product]="MB".

    1. First, the filter dBoomProducts[Product]="MB" is removed from the External Filter Context.
    2. This means that after the filter is removed from the Product Column in the external filter, the new listing of external and internal filters for the MB Row in the PivotTable would look like this:

        1. Internal filter = dProduct[Product]="Quad"
        2. External filter = dSalesRep[Region]="West"

    3. Then an AND Logical Test is run on the internal and external filters:

        1. Internal filter = dProduct[Product]="Quad"
           AND
        2. External filter = dSalesRep[Region]="West"

    4. Which Results in Final Filter Context for Dimension Tables and Fact Table that Measure uses to calculate the final answer:

        **Final Filter Context** = dProduct[Product]="Quad" AND dSalesRep[Region]="West"

6] For the MB Product Row in the above picture, here is how the Final Filter Context is determined for the **[Grand Overall Total] Measure**:

1) The external and internal filters for the MB Row in the PivotTable are:

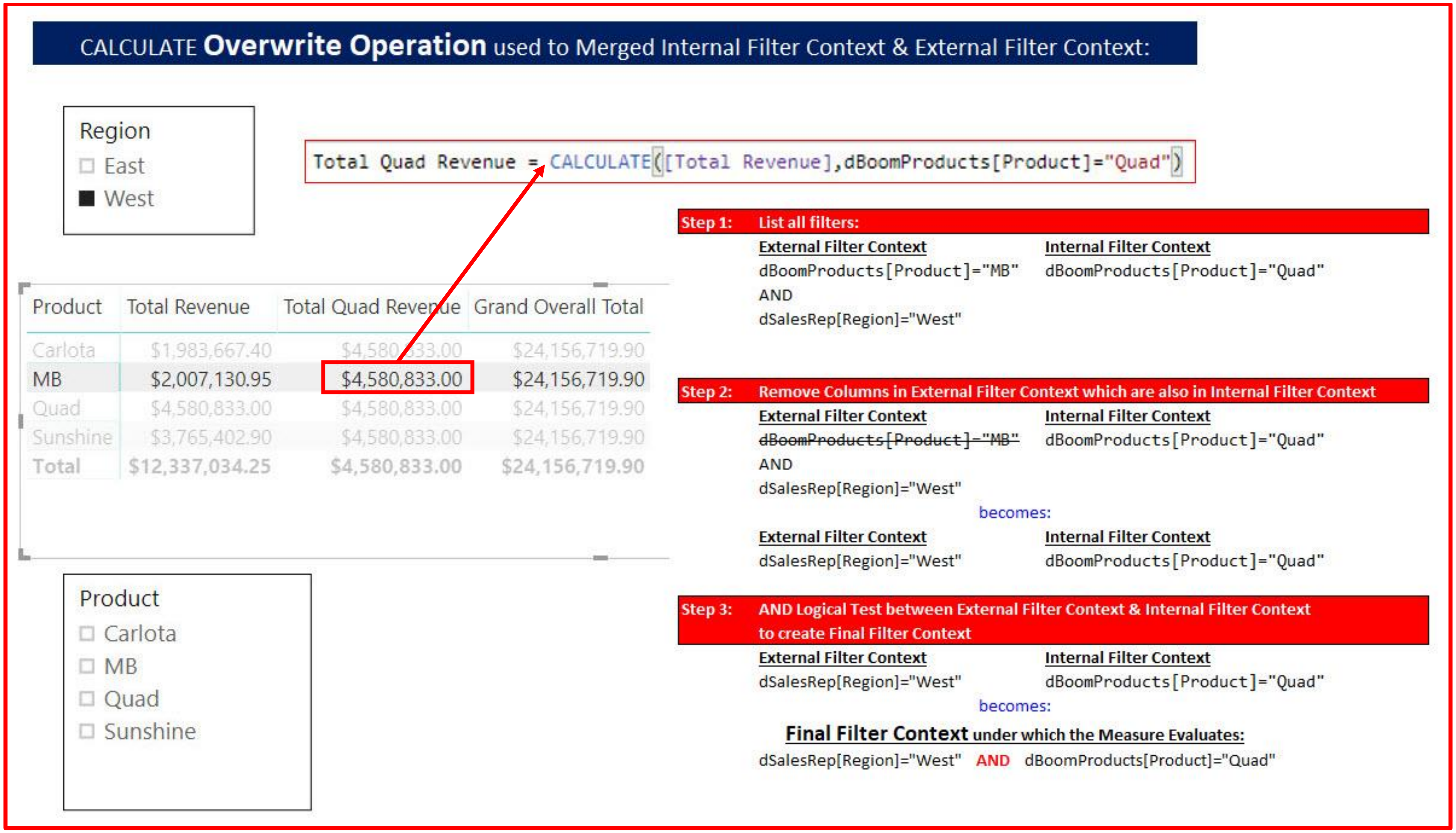
    i. Internal Filter = ALL(fTransactions) means all filters, everywhere, are removed.
    ii. External Filter = dProduct[Product]="MB" AND dSalesRep[Region]="West"

2) OVERWRITE Process to get Final Filter Context:

    i. Because the only internal filter is the ALL function wrapped around the Fact Table, all filters in the Data Model are removed.

    **Final Filter Context** = ALL(fTransactions) means all filters, everywhere, are removed.

**[7] In the video, we saw the same example as on the previous two pages in Power BI Desktop. Here is a picture of the report we used and the Description of the Overwrite operation:**



CALCULATE **Overwrite Operation** used to Merged Internal Filter Context & External Filter Context:

Region
- ☐ East
- ■ West

```
Total Quad Revenue = CALCULATE([Total Revenue],dBoomProducts[Product]="Quad")
```

| Product | Total Revenue | Total Quad Revenue | Grand Overall Total |
|---|---|---|---|
| Carlota | $1,983,667.40 | $4,580,833.00 | $24,156,719.90 |
| MB | $2,007,130.95 | $4,580,833.00 | $24,156,719.90 |
| Quad | $4,580,833.00 | $4,580,833.00 | $24,156,719.90 |
| Sunshine | $3,765,402.90 | $4,580,833.00 | $24,156,719.90 |
| Total | $12,337,034.25 | $4,580,833.00 | $24,156,719.90 |

Product
- ☐ Carlota
- ☐ MB
- ☐ Quad
- ☐ Sunshine

**Step 1:** List all filters:

| External Filter Context | Internal Filter Context |
|---|---|
| dBoomProducts[Product]="MB" | dBoomProducts[Product]="Quad" |
| AND | |
| dSalesRep[Region]="West" | |

**Step 2:** Remove Columns in External Filter Context which are also in Internal Filter Context

| External Filter Context | Internal Filter Context |
|---|---|
| ~~dBoomProducts[Product]="MB"~~ | dBoomProducts[Product]="Quad" |
| AND | |
| dSalesRep[Region]="West" | |

becomes:

| External Filter Context | Internal Filter Context |
|---|---|
| dSalesRep[Region]="West" | dBoomProducts[Product]="Quad" |

**Step 3:** AND Logical Test between External Filter Context & Internal Filter Context to create Final Filter Context

| External Filter Context | Internal Filter Context |
|---|---|
| dSalesRep[Region]="West" | dBoomProducts[Product]="Quad" |

becomes:

**Final Filter Context under which the Measure Evaluates:**

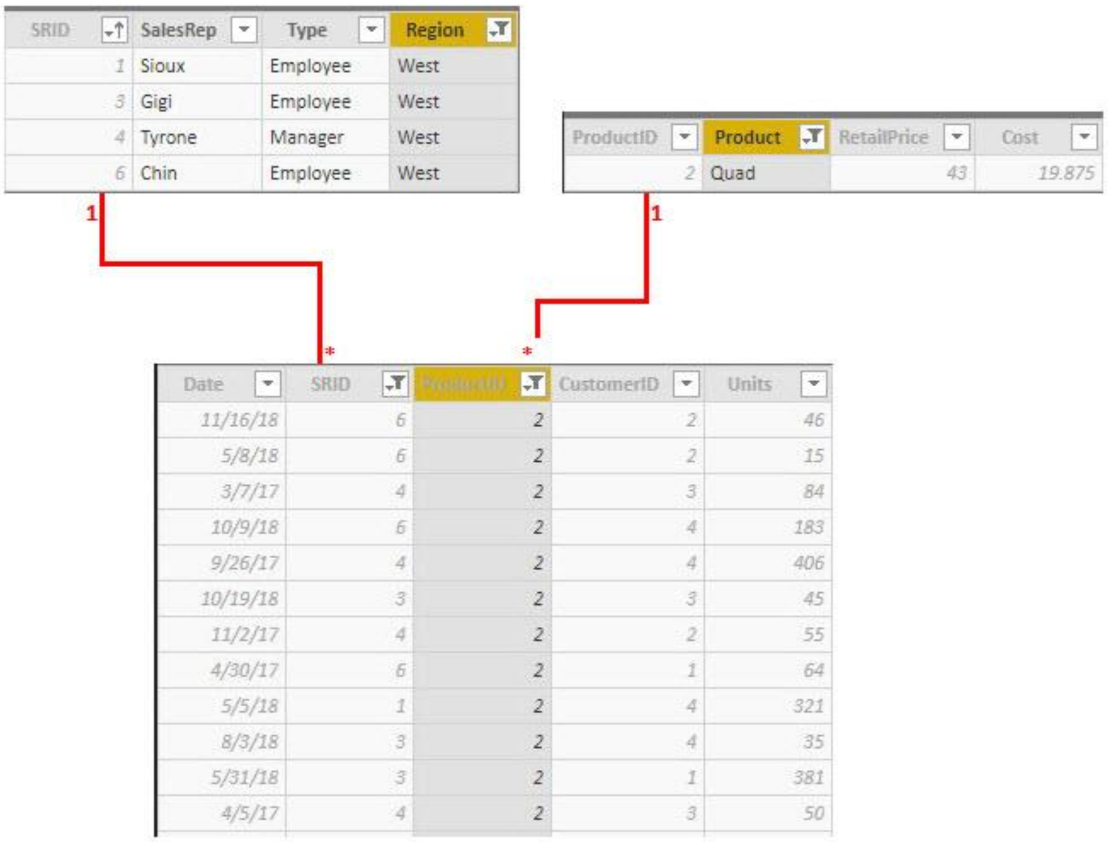
dSalesRep[Region]="West" AND dBoomProducts[Product]="Quad"

[8] The Final Filter Context of:

**Final Filter Context under which the Measure Evaluates:**

dSalesRep[Region]="West" **AND** dBoomProducts[Product]="Quad"

[9] Will filter the underlying Fact table like this:

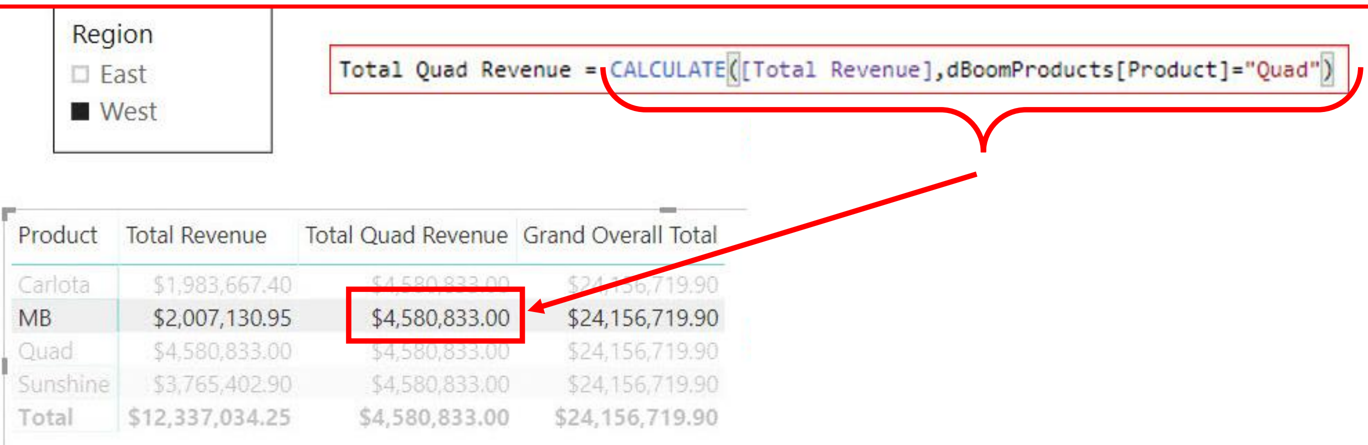


| SRID | SalesRep | Type | Region |
|---|---|---|---|
| 1 | Sioux | Employee | West |
| 3 | Gigi | Employee | West |
| 4 | Tyrone | Manager | West |
| 6 | Chin | Employee | West |

| ProductID | Product | RetailPrice | Cost |
|---|---|---|---|
| 2 | Quad | 43 | 19.875 |

| Date | SRID | ProductID | CustomerID | Units |
|---|---|---|---|---|
| 11/16/18 | 6 | 2 | 2 | 46 |
| 5/8/18 | 6 | 2 | 2 | 15 |
| 3/7/17 | 4 | 2 | 3 | 84 |
| 10/9/18 | 6 | 2 | 4 | 183 |
| 9/26/17 | 4 | 2 | 4 | 406 |
| 10/19/18 | 3 | 2 | 3 | 45 |
| 11/2/17 | 4 | 2 | 2 | 55 |
| 4/30/17 | 6 | 2 | 1 | 64 |
| 5/5/18 | 1 | 2 | 4 | 321 |
| 8/3/18 | 3 | 2 | 4 | 35 |
| 5/31/18 | 3 | 2 | 1 | 381 |
| 4/5/17 | 4 | 2 | 3 | 50 |

[10] So the Measure can calculate the final answer of $4,580,833, as seen here:



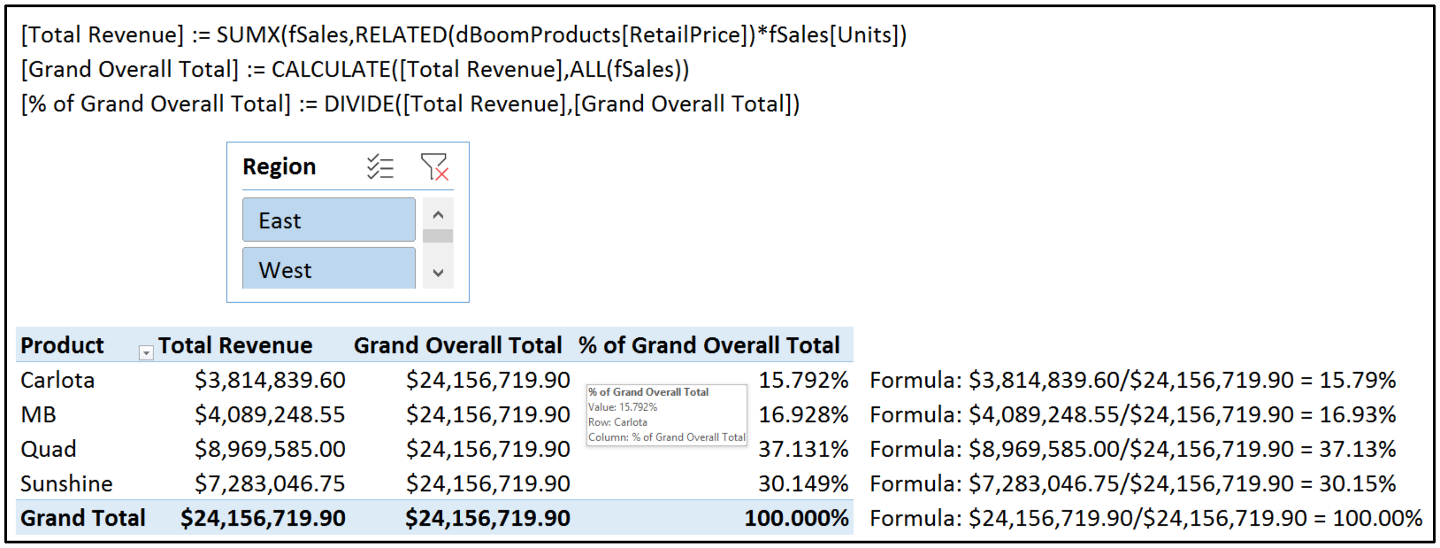
```
Total Quad Revenue = CALCULATE([Total Revenue],dBoomProducts[Product]="Quad")
```

| Product | Total Revenue | Total Quad Revenue | Grand Overall Total |
|---|---|---|---|
| Carlota | $1,983,667.40 | $4,580,833.00 | $24,156,719.90 |
| MB | $2,007,130.95 | $4,580,833.00 | $24,156,719.90 |
| Quad | $4,580,833.00 | $4,580,833.00 | $24,156,719.90 |
| Sunshine | $3,765,402.90 | $4,580,833.00 | $24,156,719.90 |
| Total | $12,337,034.25 | $4,580,833.00 | $24,156,719.90 |

15) **ALLSELECTED to create Grand Totals that respect the filtering in the PivotTable** :

i. In the below picture, the **[% of Grand Overall Total] Measure**, calculates the correct percentages for each product by dividing revenue for the product by the Grand Total amount, and the Grand Total Row in the PivotTable shows the correct 100%:

ii. This formula uses the Total Product Revenue in the Numerator and the Grand Overall Total in the Denominator.

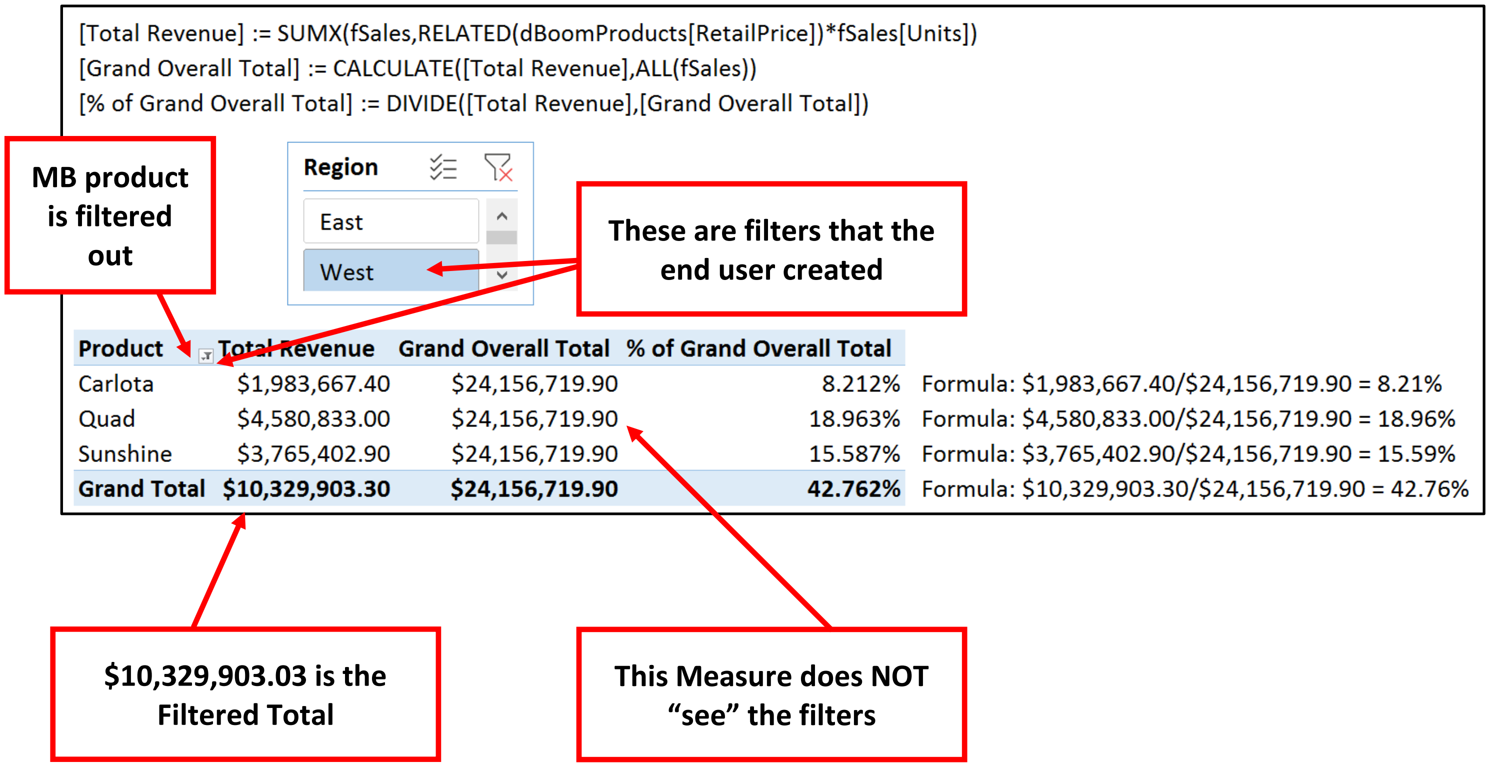


```
[Total Revenue] := SUMX(fSales,RELATED(dBoomProducts[RetailPrice])*fSales[Units])
[Grand Overall Total] := CALCULATE([Total Revenue],ALL(fSales))
[% of Grand Overall Total] := DIVIDE([Total Revenue],[Grand Overall Total])
```

| Product | Total Revenue | Grand Overall Total | % of Grand Overall Total | |
|---|---|---|---|---|
| Carlota | $3,814,839.60 | $24,156,719.90 | 15.792% | Formula: $3,814,839.60/$24,156,719.90 = 15.79% |
| MB | $4,089,248.55 | $24,156,719.90 | 16.928% | Formula: $4,089,248.55/$24,156,719.90 = 16.93% |
| Quad | $8,969,585.00 | $24,156,719.90 | 37.131% | Formula: $8,969,585.00/$24,156,719.90 = 37.13% |
| Sunshine | $7,283,046.75 | $24,156,719.90 | 30.149% | Formula: $7,283,046.75/$24,156,719.90 = 30.15% |
| **Grand Total** | **$24,156,719.90** | **$24,156,719.90** | **100.000%** | Formula: $24,156,719.90/$24,156,719.90 = 100.00% |

iii. But as seen in the picture below, if we filter the PivotTable on the Row Area or with the Slicer, notice:

1. The **[Grand Overall Total] Measure** does not change. The **[Grand Overall Total] Measure** does not see the filters for the Region or the Product.
2. The **[% of Grand Overall Total] Measure** uses the filtered product revenue in the numerator and the NON-Filtered Grand Overall Total in the denominator. This might be the calculation that you want. But if you wanted both the numerator and denominator to reflect the filters being applied by the report user, then we need to switch from using the ALL function as a filter inside the CALCULATE function to the ALLSELECTED function.

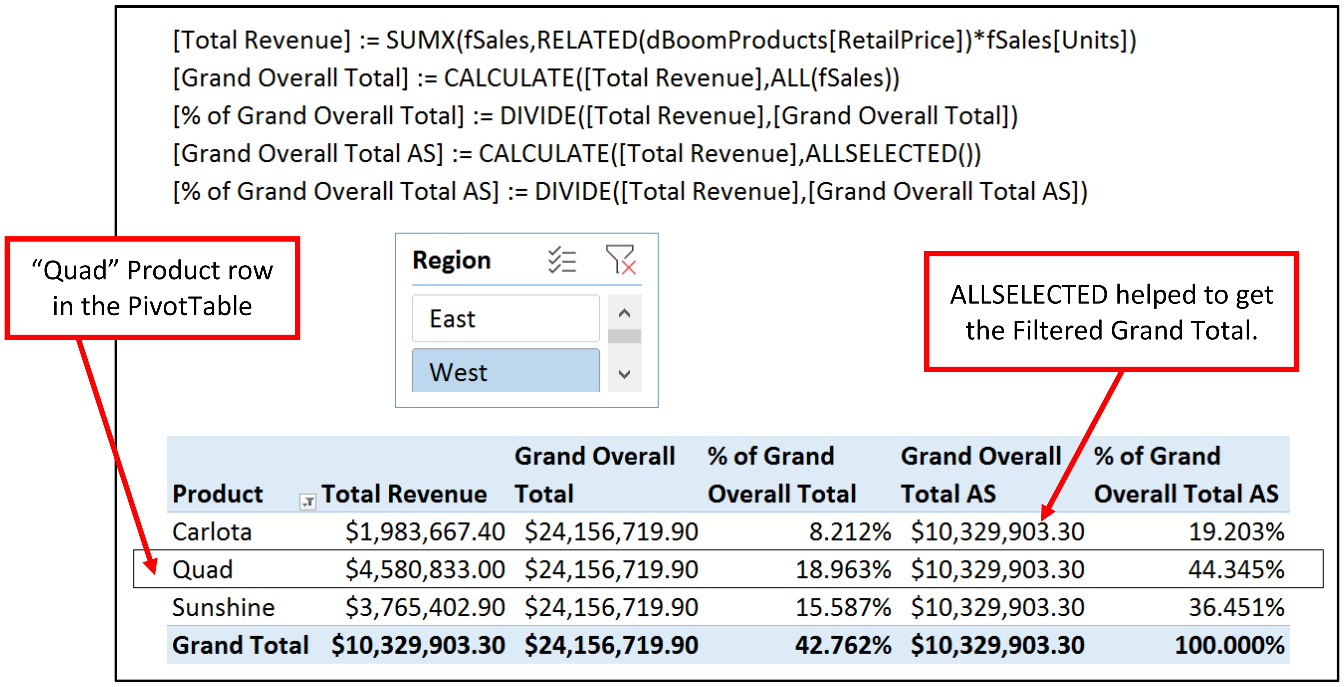

Region
East
West
MB product is filtered out
These are filters that the end user created
$10,329,903.03 is the Filtered Total
This Measure does NOT "see" the filters

```
[Total Revenue] := SUMX(fSales,RELATED(dBoomProducts[RetailPrice])*fSales[Units])
[Grand Overall Total] := CALCULATE([Total Revenue],ALL(fSales))
[% of Grand Overall Total] := DIVIDE([Total Revenue],[Grand Overall Total])
```

| Product | Total Revenue | Grand Overall Total | % of Grand Overall Total | |
|---|---|---|---|---|
| Carlota | $1,983,667.40 | $24,156,719.90 | 8.212% | Formula: $1,983,667.40/$24,156,719.90 = 8.21% |
| Quad | $4,580,833.00 | $24,156,719.90 | 18.963% | Formula: $4,580,833.00/$24,156,719.90 = 18.96% |
| Sunshine | $3,765,402.90 | $24,156,719.90 | 15.587% | Formula: $3,765,402.90/$24,156,719.90 = 15.59% |
| **Grand Total** | **$10,329,903.30** | **$24,156,719.90** | **42.762%** | Formula: $10,329,903.30/$24,156,719.90 = 42.76% |

iv. By switching from ALL(fTransactions) to ALLSELECTED() as the only filter in the Filter argument of the CALCULATE function, then the final answer for the **Measure [Grand Overall Total AS]** will be the total revenue for the filters shown in the PivotTable. And the final answers for the **Measure [% Grand Overall Total AS]** will have the correctly filtered amounts in the numerator and denominator.

v. To understand what the ALLSELECTED() function is doing to get the correct filtered grand total for the [Grand Overall Total AS] Measure, look at the **"Quad" Product row in the PivotTable** in the picture below. For the Measure in this row, the ALLSELECTED removes the "Quad" Product Row Context and jumps back to the previous Grand Total Filter that contains the products "Carlota", "Quad", and Sunshine" in the "West" Region. Said a different way: ALLSELECTED will remove the "Row Context" generated by the automatic Context Transition for each cell in the PivotTable and jump back to expose the filtered Grand Total Filter Context, which consists of the filters "Carlota", "Quad", and Sunshine" products and the "West" Region.



```
[Total Revenue] := SUMX(fSales,RELATED(dBoomProducts[RetailPrice])*fSales[Units])
[Grand Overall Total] := CALCULATE([Total Revenue],ALL(fSales))
[% of Grand Overall Total] := DIVIDE([Total Revenue],[Grand Overall Total])
[Grand Overall Total AS] := CALCULATE([Total Revenue],ALLSELECTED())
[% of Grand Overall Total AS] := DIVIDE([Total Revenue],[Grand Overall Total AS])
```

"Quad" Product row in the PivotTable
Region
East
West
ALLSELECTED helped to get the Filtered Grand Total.

| Product | Total Revenue | Grand Overall Total | % of Grand Overall Total | Grand Overall Total AS | % of Grand Overall Total AS |
|---|---|---|---|---|---|
| Carlota | $1,983,667.40 | $24,156,719.90 | 8.212% | $10,329,903.30 | 19.203% |
| Quad | $4,580,833.00 | $24,156,719.90 | 18.963% | $10,329,903.30 | 44.345% |
| Sunshine | $3,765,402.90 | $24,156,719.90 | 15.587% | $10,329,903.30 | 36.451% |
| **Grand Total** | **$10,329,903.30** | **$24,156,719.90** | **42.762%** | **$10,329,903.30** | **100.000%** |

vi. The full definition of the ALLSELECTED DAX function is:

**Removes the last filter generated from Context Transition and jumps back to the Previous Filter**

vii. To fully understand ALLSELECTED and Context Transition, we need to re-visit the topic of Context Transition on the next page.

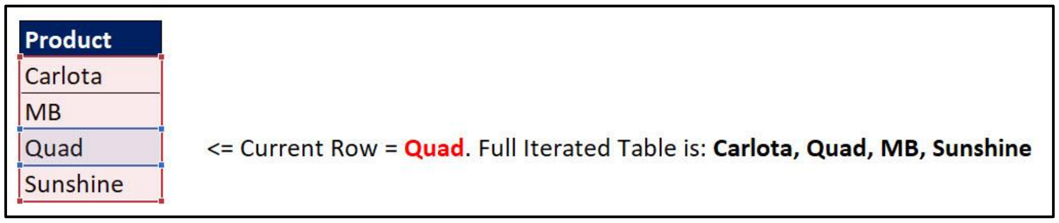
16) **Context Transition Full Story** :

i. Context Transition occurs when you use the CALCULATE Function or the hidden CALCULATE Function wrapped around every Measure in these places:
   1. Calculated Column,
   2. Iterator Function
   3. An Excel PivotTable or Power BI Visual.

ii. Context Transition will convert all available Row Contexts into an Equivalent Filter Context. If there are stacked up Row Contexts, one after the other, they will be merged in an AND Logical Test. Further, if there are columns from the Context Transition (External Filter Context) that are the same as columns inside the CALCULATE (Internal Filter Context), then the Overwrite process will remove the external columns and the internal columns will remain.

iii. Context Transition has a hidden process that only comes into play when we use the ALLSELECTED DAX Function.

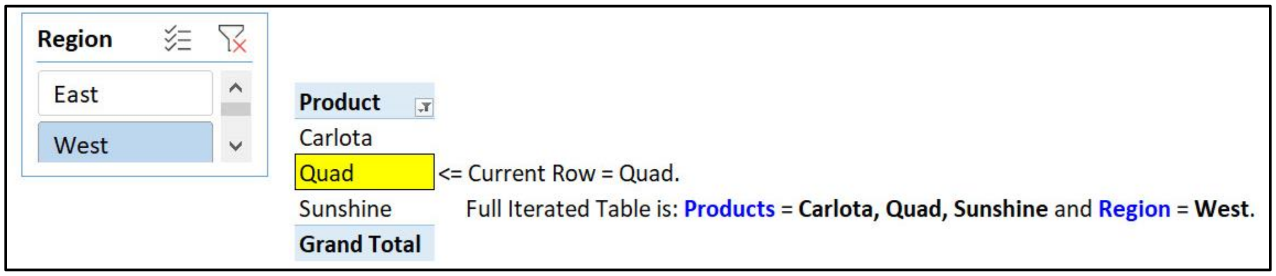
iv. The hidden process in Context Transition is this: when the CALCULTAE function and the DAX Engine initiate the transition from Row Context to Filter Context, two layers of filters are created where one layer is the current row and the other is the full iterated table behind it.
   1. Example 1. In the below table of Product names, the "Current Row" would be "Quad" and the "Full Iterated Table Behind It" would be the full list of product names.



| Product |
|---|
| Carlota |
| MB |
| Quad |
| Sunshine |

<= Current Row = **Quad**. Full Iterated Table is: **Carlota, Quad, MB, Sunshine**

   2. Example 2: In the Below PivotTable, the "Current Row" would be "Quad" and the "Full Iterated Table Behind It", or more specifically, the Filter Context for the Grand Total Cell would be Products = Carlota, Quad, Sunshine and the Region = West.



| Product |
|---|
| Carlota |
| Quad |
| Sunshine |
| **Grand Total** |

<= Current Row = Quad.
Full Iterated Table is: **Products** = **Carlota, Quad, Sunshine** and **Region** = **West.**

v. These two layers or filters or stacks of filters can be thought of with different synonyms:
   1. Current Row = Second Filter = Last Filter= Inner Filter
   2. Full Iterated Table = First Filter = Previous Filter = Outer Filter

vi. The Two Filters Get applied one after the other by the engine:
   1. First = Full Iterated Table
   2. Second = Current Row

vii. Most of the time we don't have to think of the two layers or filters. Most of the time we just think of the "Current Row" in a Calculated Column, or in an Iterator Function. But, when we use the ALLSELECTED DAX Function, we have to think about these two filters because ALLSELECTED will only remove the last filter created by Context Transition. If we have two iterations, one after the other, then things might seem difficult if we are not paying attention. Our next section will illustrate this.

17) **ALLSELECTED DAX Function** :

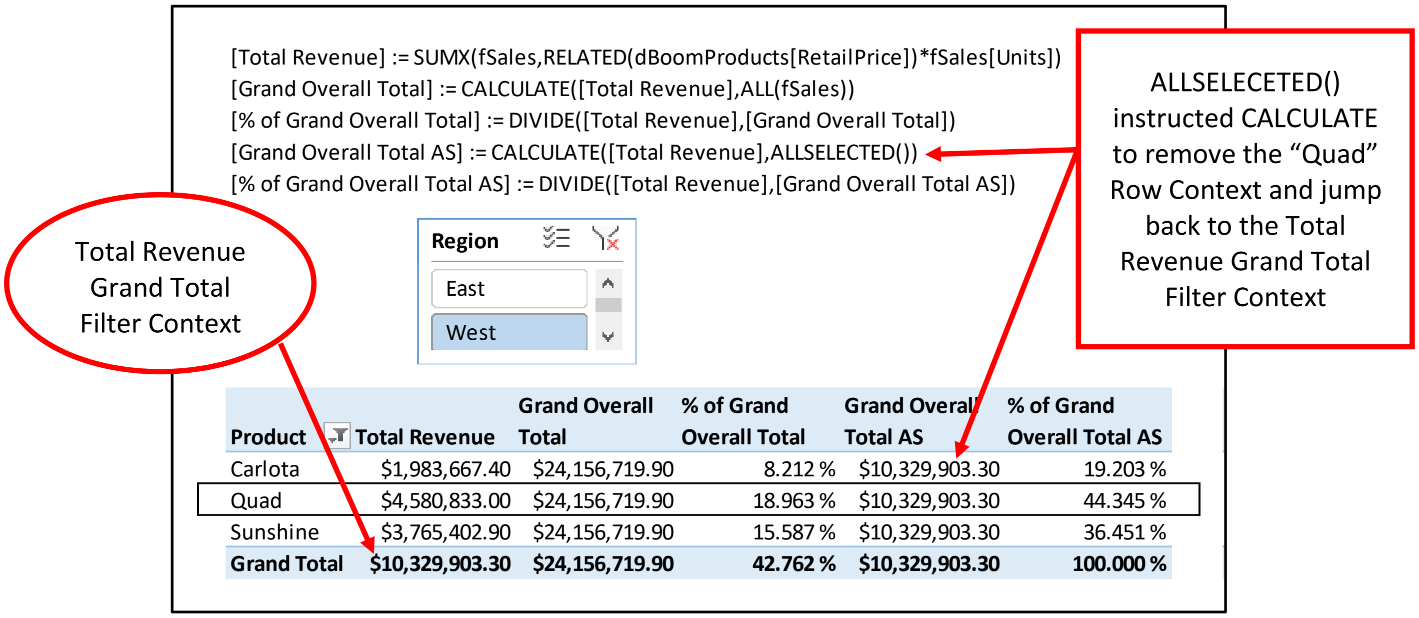
i. What ALLSELECTED does:

   **Removes the last filter generated from Context Transition and jumps back to the Previous Filter**

ii. Most of the Time, ALLSELECTED() is used to get a Denominator Number for % of Grand Totals and allow Filter or Slicer selections to influence the Grand Total.

iii. A formula like this: **CALCULATE([Total Sales],ALLSELECTED())**

   ALLSELECTED would remove the last filter (Row Filter) and jump back to the Row/Column/Filter/Slicer filtering for the Grand Total cell in the External Filter Context in the Excel PivotTable or the Power BI Visual. This is helpful if you would like your Measure to calculate the Grand Overall Total based on any applied Filters by the client (end report consumer). As seen in the picture on the next page, ALLSELECETED() instructed CALCULATE to remove the "Quad" Row Context and jump back to the Total Revenue Grand Total Filter Context which are the filters "Carlota", "Quad", and Sunshine" from the dBoomProduct table and the "West" Region from the dSalesRep table as seen here.:



[Total Revenue] := SUMX(fSales,RELATED(dBoomProducts[RetailPrice])*fSales[Units])
[Grand Overall Total] := CALCULATE([Total Revenue],ALL(fSales))
[% of Grand Overall Total] := DIVIDE([Total Revenue],[Grand Overall Total])
[Grand Overall Total AS] := CALCULATE([Total Revenue],ALLSELECTED())
[% of Grand Overall Total AS] := DIVIDE([Total Revenue],[Grand Overall Total AS])

ALLSELECECTED() instructed CALCULATE to remove the "Quad" Row Context and jump back to the Total Revenue Grand Total Filter Context

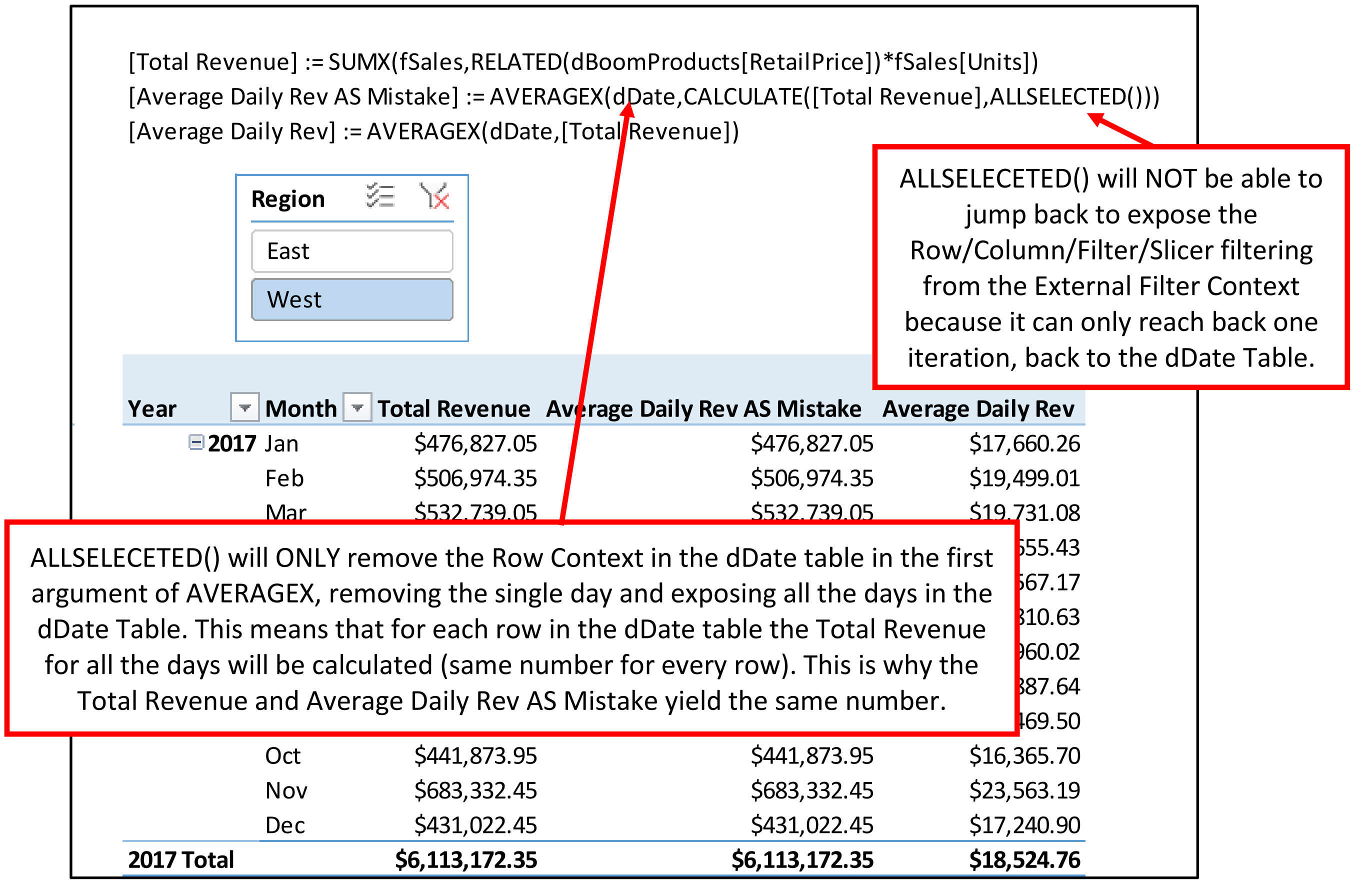
Total Revenue Grand Total Filter Context

| Product | Total Revenue | Grand Overall Total | % of Grand Overall Total | Grand Overall Total AS | % of Grand Overall Total AS |
|---|---|---|---|---|---|
| Carlota | $1,983,667.40 | $24,156,719.90 | 8.212 % | $10,329,903.30 | 19.203 % |
| Quad | $4,580,833.00 | $24,156,719.90 | 18.963 % | $10,329,903.30 | 44.345 % |
| Sunshine | $3,765,402.90 | $24,156,719.90 | 15.587 % | $10,329,903.30 | 36.451 % |
| **Grand Total** | **$10,329,903.30** | **$24,156,719.90** | **42.762 %** | **$10,329,903.30** | **100.000 %** |

iv. But a formula like this **AVERAGEX(dDate,CALCULATE([Total Sales],ALLSELECTED())**

ALLSELECTED would remove the last filter (Row Filter in row in dDate Table) and jump back to the dDate Table. Because ALLSELECETED removes only the last filter created during Context Transition, it will NOT be able to jump back to expose the filtering from the External Filter Context in the Excel PivotTable or the Power BI Visual Grand Total Cell.

v. Example of the incorrect answers if we use ALLSELECTED in an iterator function like AVERAGEX:



[Total Revenue] := SUMX(fSales,RELATED(dBoomProducts[RetailPrice])*fSales[Units])
[Average Daily Rev AS Mistake] := AVERAGEX(dDate,CALCULATE([Total Revenue],ALLSELECTED()))
[Average Daily Rev] := AVERAGEX(dDate,[Total Revenue])

ALLSELECETED() will NOT be able to jump back to expose the Row/Column/Filter/Slicer filtering from the External Filter Context because it can only reach back one iteration, back to the dDate Table.

| Year | Month | Total Revenue | Average Daily Rev AS Mistake | Average Daily Rev |
|---|---|---|---|---|
| ⊟ **2017** | Jan | $476,827.05 | $476,827.05 | $17,660.26 |
| | Feb | $506,974.35 | $506,974.35 | $19,499.01 |
| | Mar | $532,739.05 | $532,739.05 | $19,731.08 |
| | | | | 55.43 |
| | | | | 67.17 |
| | | | | 10.63 |
| | | | | 60.02 |
| | | | | 87.64 |
| | | | | 69.50 |
| | Oct | $441,873.95 | $441,873.95 | $16,365.70 |
| | Nov | $683,332.45 | $683,332.45 | $23,563.19 |
| | Dec | $431,022.45 | $431,022.45 | $17,240.90 |
| **2017 Total** | | **$6,113,172.35** | **$6,113,172.35** | **$18,524.76** |

ALLSELECETED() will ONLY remove the Row Context in the dDate table in the first argument of AVERAGEX, removing the single day and exposing all the days in the dDate Table. This means that for each row in the dDate table the Total Revenue for all the days will be calculated (same number for every row). This is why the Total Revenue and Average Daily Rev AS Mistake yield the same number.

18) **KEEPFILTERS DAX Function** :

i. KEEPFILTERS, in brief:
   1. Changes the Overwrite Operation in CALCULATE to an AND Logical Test.

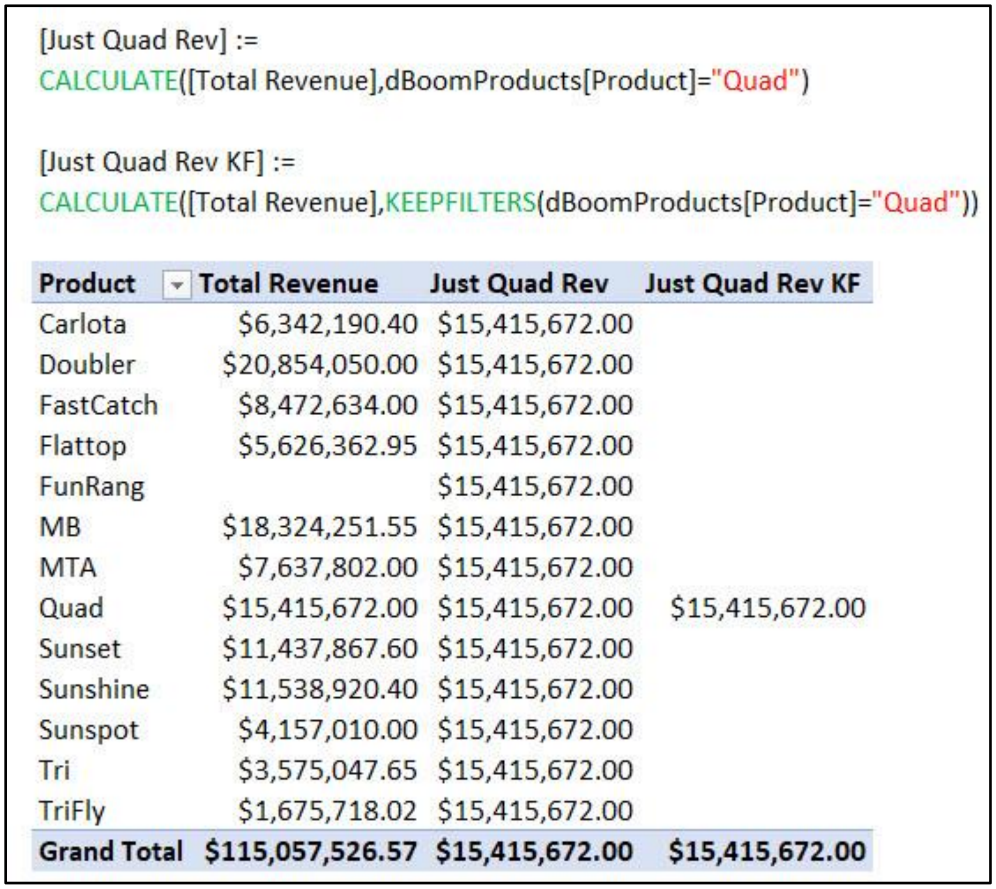
ii. KEEPFILTERS, in more detail:
   1. Merges the internal filter/s inside of KEEPFILTERS with the full External (Previous) Filter Context as an AND Logical Test, rather than as an Overwrite Operation.

iii. KEEPFILTERS is not a Table Function.

iv. You can use the KEEPFILTERS function in:
   1. Filter argument of CALCULATE
      or
   2. Wrapped around a table in first argument of an Iterator function.

v. For the examples below we will use the Excel file named "019-MSPTDA-CALCULATE-FilterContext-03-Start.xlsx".

vi. Example of KEEPFILTER being used in the filter argument of CALCULATE, is shown below :
   1. In the **[Just Quad Rev] Measure** with a Boolean Filter, the internal filter of "Quad" Overwrites the external filter for each row in the PivotTable with the result being the "Quad" Total Revenue in each cell.
   2. For the **[Just Quad Rev KF] Measure** we use the KEEPFILTERS function around the Boolean "Quad" filter. This means that for each row in the PivotTable an AND Logical Test will be run between the internal filter and the external filter. In the below picture we can see that the only row in the PivotTable that gets an answer is the "Quad" row because that is the only row that gets a TRUE (Product = "Quad") for the internal and external filters.

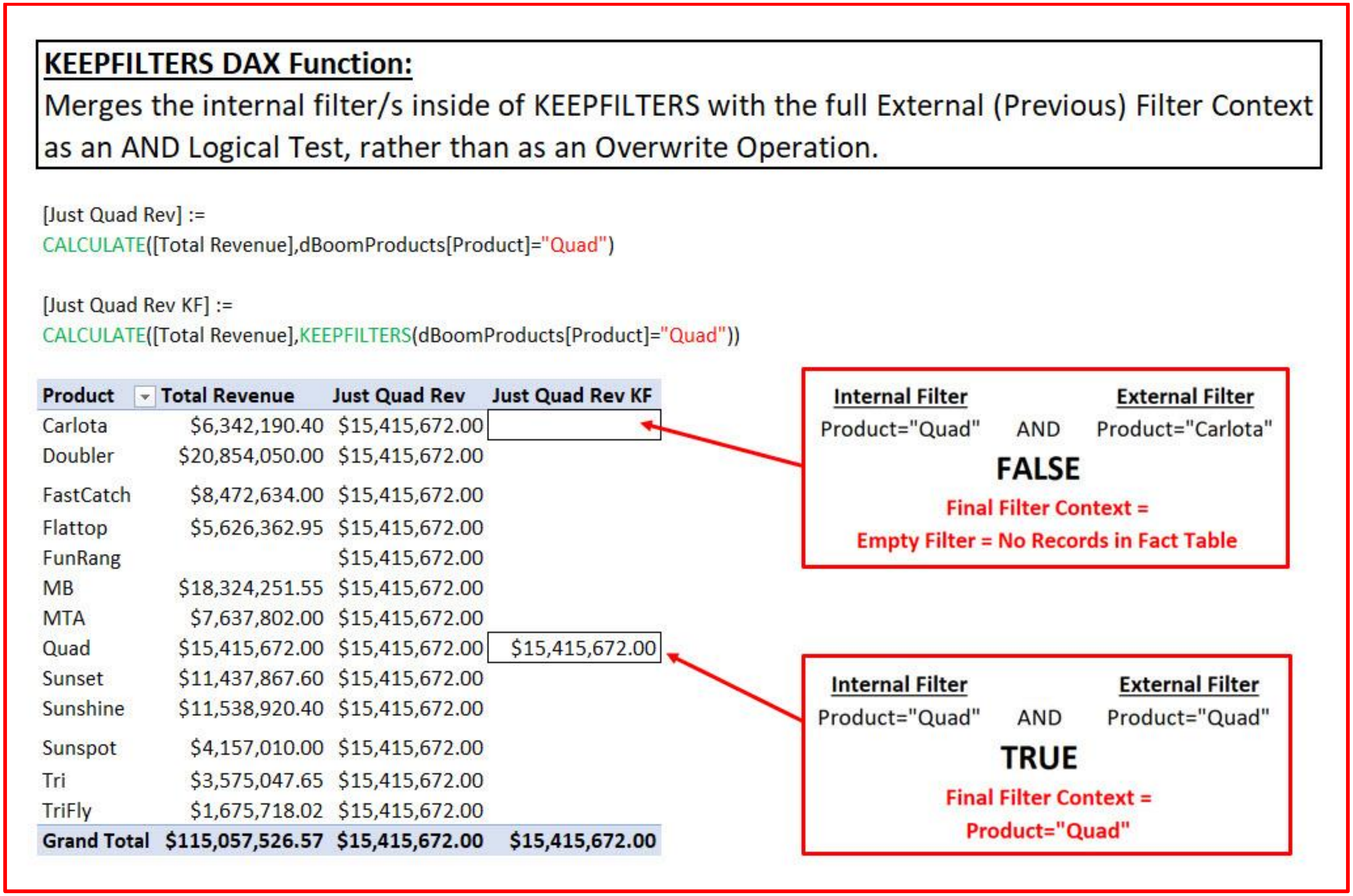


```
[Just Quad Rev] :=
CALCULATE([Total Revenue],dBoomProducts[Product]="Quad")

[Just Quad Rev KF] :=
CALCULATE([Total Revenue],KEEPFILTERS(dBoomProducts[Product]="Quad"))
```

| Product | Total Revenue | Just Quad Rev | Just Quad Rev KF |
|---|---|---|---|
| Carlota | $6,342,190.40 | $15,415,672.00 | |
| Doubler | $20,854,050.00 | $15,415,672.00 | |
| FastCatch | $8,472,634.00 | $15,415,672.00 | |
| Flattop | $5,626,362.95 | $15,415,672.00 | |
| FunRang | | $15,415,672.00 | |
| MB | $18,324,251.55 | $15,415,672.00 | |
| MTA | $7,637,802.00 | $15,415,672.00 | |
| Quad | $15,415,672.00 | $15,415,672.00 | $15,415,672.00 |
| Sunset | $11,437,867.60 | $15,415,672.00 | |
| Sunshine | $11,538,920.40 | $15,415,672.00 | |
| Sunspot | $4,157,010.00 | $15,415,672.00 | |
| Tri | $3,575,047.65 | $15,415,672.00 | |
| TriFly | $1,675,718.02 | $15,415,672.00 | |
| **Grand Total** | **$115,057,526.57** | **$15,415,672.00** | **$15,415,672.00** |

vii. To understand the logical of the AND Logical Test for the KEEPFILTERS function, here are two pictorial examples on the next page:



**KEEPFILTERS DAX Function:**

Merges the internal filter/s inside of KEEPFILTERS with the full External (Previous) Filter Context as an AND Logical Test, rather than as an Overwrite Operation.

```
[Just Quad Rev] :=
CALCULATE([Total Revenue],dBoomProducts[Product]="Quad")

[Just Quad Rev KF] :=
CALCULATE([Total Revenue],KEEPFILTERS(dBoomProducts[Product]="Quad"))
```

| Product | Total Revenue | Just Quad Rev | Just Quad Rev KF |
|---|---|---|---|
| Carlota | $6,342,190.40 | $15,415,672.00 | |
| Doubler | $20,854,050.00 | $15,415,672.00 | |
| FastCatch | $8,472,634.00 | $15,415,672.00 | |
| Flattop | $5,626,362.95 | $15,415,672.00 | |
| FunRang | | $15,415,672.00 | |
| MB | $18,324,251.55 | $15,415,672.00 | |
| MTA | $7,637,802.00 | $15,415,672.00 | |
| Quad | $15,415,672.00 | $15,415,672.00 | $15,415,672.00 |
| Sunset | $11,437,867.60 | $15,415,672.00 | |
| Sunshine | $11,538,920.40 | $15,415,672.00 | |
| Sunspot | $4,157,010.00 | $15,415,672.00 | |
| Tri | $3,575,047.65 | $15,415,672.00 | |
| TriFly | $1,675,718.02 | $15,415,672.00 | |
| **Grand Total** | **$115,057,526.57** | **$15,415,672.00** | **$15,415,672.00** |

| **Internal Filter** | | **External Filter** |
|---|---|---|
| Product="Quad" | AND | Product="Carlota" |

**FALSE**

**Final Filter Context =**

**Empty Filter = No Records in Fact Table**

| **Internal Filter** | | **External Filter** |
|---|---|---|
| Product="Quad" | AND | Product="Quad" |

**TRUE**

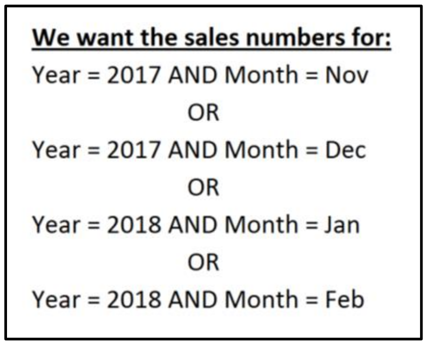
**Final Filter Context =**

**Product="Quad"**

viii. KEEPFILTERS can also be used to solve a Complex Filter Reduction Error. To understand how KEEPFILTERS can help with this error, we have to define a “Complex Filter”.

ix. First attempt at defining a Complex Filter :

1. Complex Filter = a filter that combines an OR Logical Test with an AND Logical Test, where multiple items are selected from the columns involved in the filter.
    i. Example:



**We want the sales numbers for:**

Year = 2017 AND Month = Nov

OR

Year = 2017 AND Month = Dec

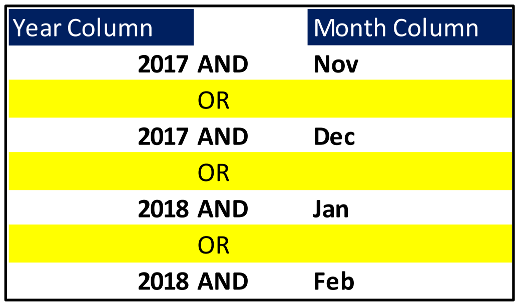
OR

Year = 2018 AND Month = Jan

OR

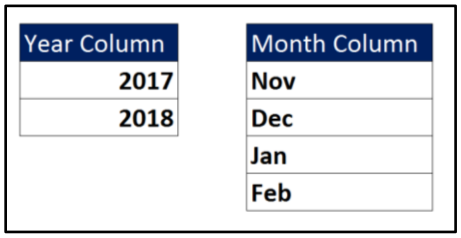
Year = 2018 AND Month = Feb

    ii. As seen in the below picture, we have four AND Logical Tests that are being used as individual elements in an OR Logical Test. In addition, there are multiple items selected for the Year and Month Columns, where the years 2017 and 2018 are selected on the Year Column and the months Nov, Dec, Jan and Feb are selected on the Month Column.



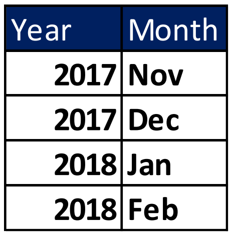
| Year Column | Month Column |
|---|---|
| **2017 AND** | **Nov** |
| OR | |
| **2017 AND** | **Dec** |
| OR | |
| **2018 AND** | **Jan** |
| OR | |
| **2018 AND** | **Feb** |

2. If we looked at this Complex Filter individually, as separated columns, we would see this:



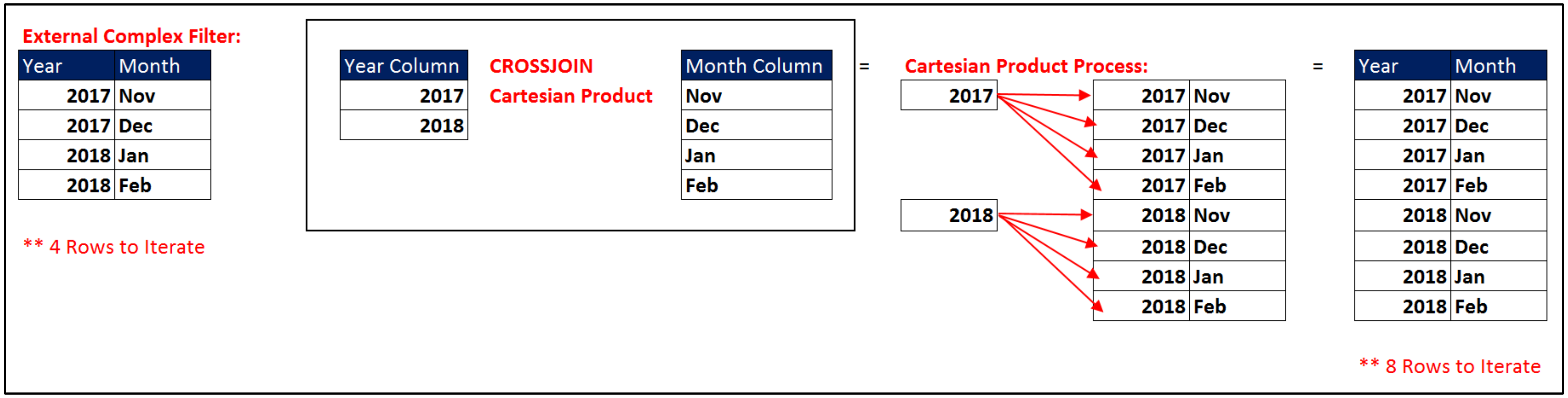
| Year Column | Month Column |
|---|---|
| **2017** | **Nov** |
| **2018** | **Dec** |
| | **Jan** |
| | **Feb** |

3. The problem with listing the columns individually, as in the above picture, is that we lose the definition of the AND and OR Logical tests, we lose the logic of the Complex Filter. We can no longer tell which year goes with which month.
4. Now we want to remember our Excel skills from the pre-requisite class Busn 218. If we were to use the Advanced Filter feature or Excel Database Functions, the combination of the AND and OR Logical Tests for our example would look like the below picture, where the AND Logical Test is listed on a single row and the OR Logical Test is listed on separate rows.



| Year | Month |
|---|---|
| **2017** | **Nov** |
| **2017** | **Dec** |
| **2018** | **Jan** |
| **2018** | **Feb** |

5. As seen on the previous page, Complex Filters will always look like tables (relations from Relational Algebra terminology) with two or more columns and two or more rows, and there are multiple items select for the columns (2017 and 2018 selected on Year; Nov, Dec, Jan and Feb selected on Month). This means that the two columns are locked in a relationship and cannot be separated. For example, for the first row in the above filter, the Year 2017 and the Month Nov must be used together in an AND Logical test. In the last row, the Year 2018 and the Month Feb must be used together in an AND Logical test. Because we can NOT separated the two columns that exist in the External Complex Filter Context, if we had an internal filter for the Year Column and it removed the Year column from the External Complex Filter Context, it would destroy the Complex Filter and cause a "Complex Filter Reduction Error".
6. Before we define what a "Complex Filter Reduction Error" is, we want to look at a definitive way to determine if your filter is a Complex Filter.
7. The rule for determining if you have a Complex Filter is :
    i. If you execute a CROSSJOIN on the columns involved in the Complex Filter and the result has more rows to iterate than the original Complex Filter, then you have a Complex Filter.
8. As seen below, if we execute a CROSSJOIN on the two columns involved in our External Complex Filter, we would end up with eight rows to iterate over rather than the original four rows. In this way, we know that we have a Complex Filter. If this Complex Filter is in the External Filter Context of our Excel PivotTable or Power BI Visual, then we may run the risk of a Complex Filter Reduction Error, which we will define next.



**External Complex Filter:**

| Year | Month |
|---|---|
| 2017 | Nov |
| 2017 | Dec |
| 2018 | Jan |
| 2018 | Feb |

** 4 Rows to Iterate

| Year Column |
|---|
| 2017 |
| 2018 |

**CROSSJOIN**
**Cartesian Product**

| Month Column |
|---|
| Nov |
| Dec |
| Jan |
| Feb |

=

**Cartesian Product Process:**

| | | |
|---|---|---|
| 2017 | 2017 | Nov |
| | 2017 | Dec |
| | 2017 | Jan |
| | 2017 | Feb |
| 2018 | 2018 | Nov |
| | 2018 | Dec |
| | 2018 | Jan |
| | 2018 | Feb |

=

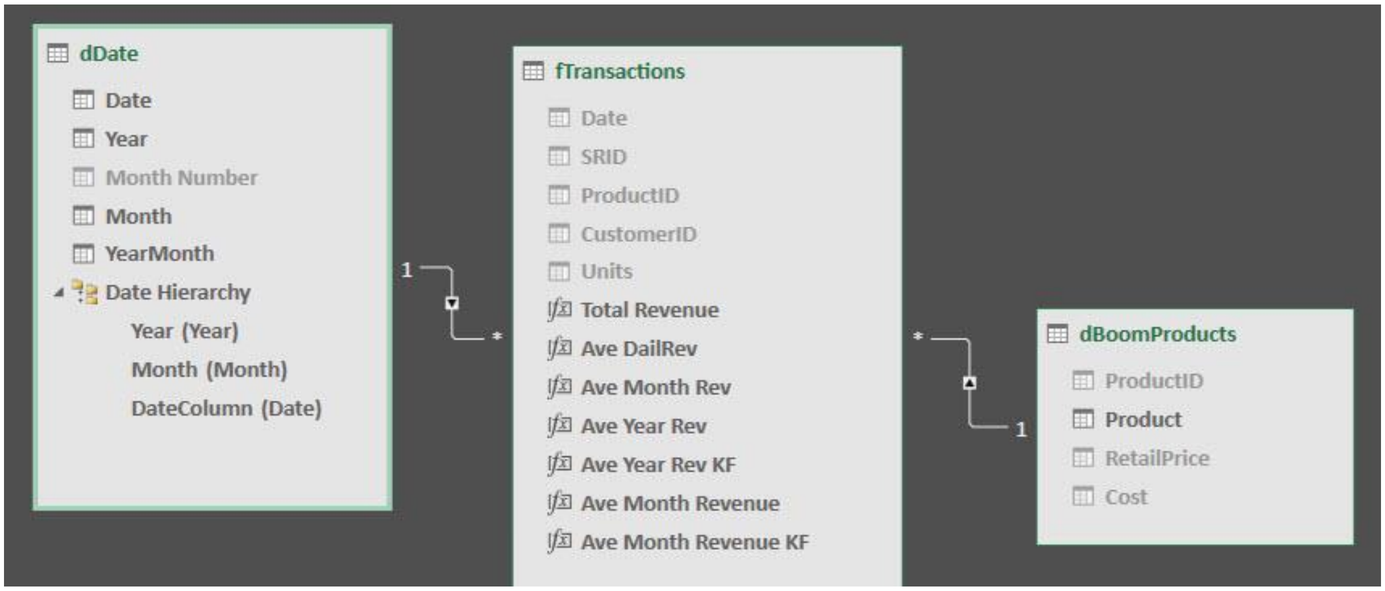
| Year | Month |
|---|---|
| 2017 | Nov |
| 2017 | Dec |
| 2017 | Jan |
| 2017 | Feb |
| 2018 | Nov |
| 2018 | Dec |
| 2018 | Jan |
| 2018 | Feb |

** 8 Rows to Iterate

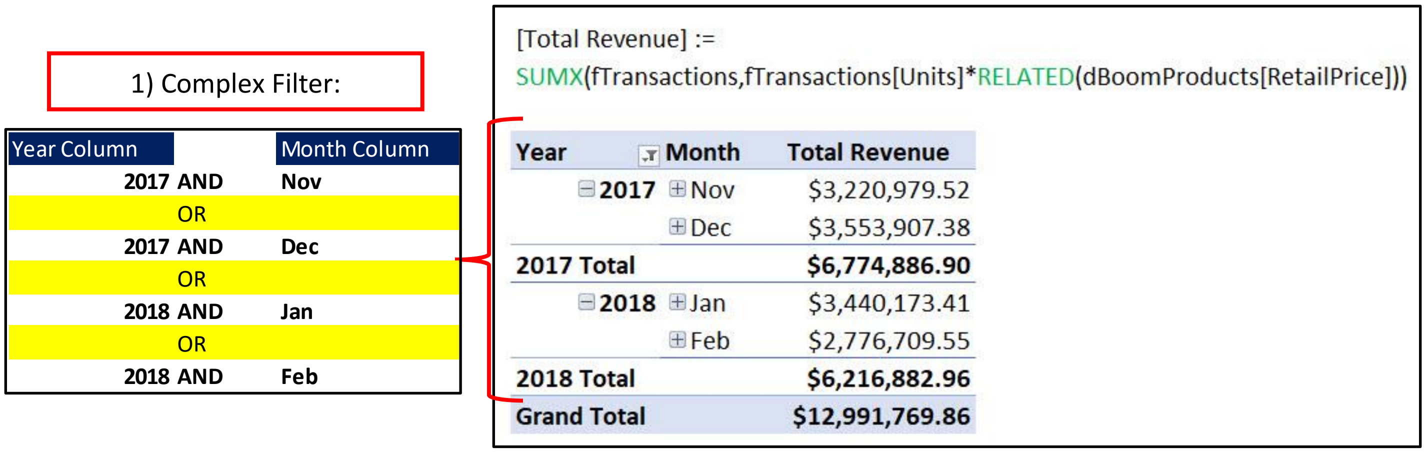
x. Complex Filter Reduction Error can happen when:
   1. We have a complex filter on two or more columns in an Excel PivotTable Report or Power BI Visualization (report).
   2. In a Measure, we have an Iterator function that is iterating over one or more of the columns involved in the external complex filter.
   3. Context Transition (Row into Filter Context) is occurring for the Measure in the report and for the 2nd argument in the iterator.
   4. The Overwrite process in CALCULATE replaces the External Column/s with the Internal Columns/s and leads to the incorrect number of iterating rows in the Iterator function.
   5. This problem only happens when a complex filter is used in report (External Filter Context), the Iterator in the Measure uses the same one or more columns internally, and then uses the internal columns to overwrite the external complex filter columns.
   6. The KEEPFILTERS function can help to solve this error by instructing CALCULATE to use an AND Logical test rather than the Overwrite Operation.

xi. When we should use the KEEPFILTERS function to avoid a Complex Filter Reduction Error:
   1. We have a Measure that contains an Iterator function that is using one or more columns to iterate across.
   2. The same one or more columns in the first argument of the iterator are used in the External Report and may be filtered as a complex filter.
   3. When we have Iteration and Context Transition in our Measure that might create the wrong number of iterations, and so we need to use KEEPFILTERS to force the CALCULATE function Overwrite Operation to be executed as an AND Logical Test.

xii. **Example 01 of KEEPFILTERS in first argument of an Iterator to solve Complex Filter Reduction Error** :
   1. File name for example: "019-MSPTDA-CALCULATE-FilterContext-03-Start.xlsx"
   2. Here is what our Data Model looks like:



3. In the below example we have a complex filter in the external report based on the Year and Month Columns that are part of the Date Hierarchy, as seen here:



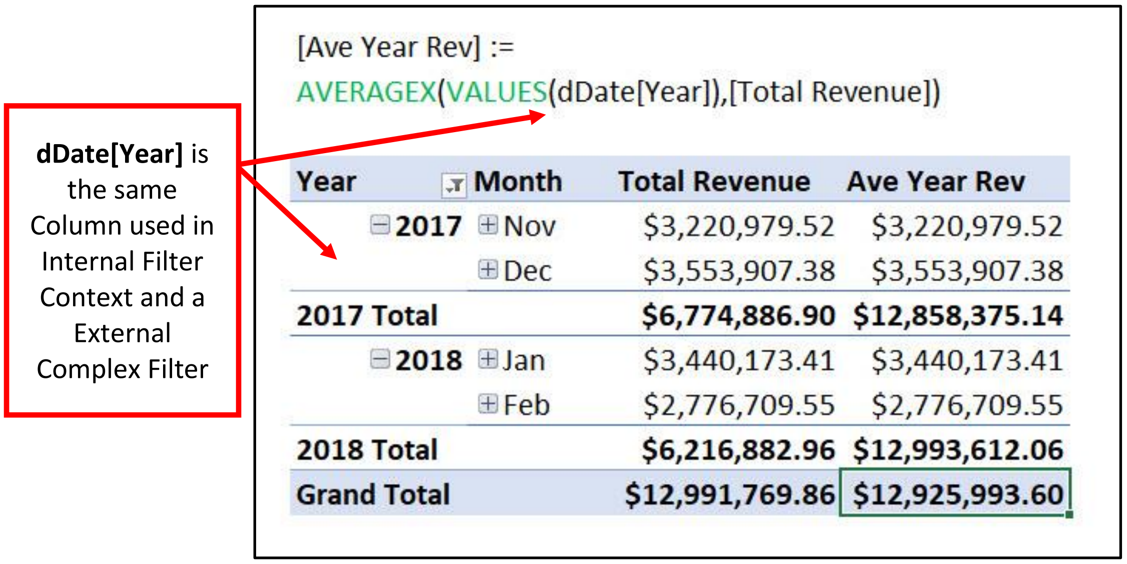
1) Complex Filter:

| Year Column | Month Column |
|---|---|
| 2017 AND | Nov |
| OR | |
| 2017 AND | Dec |
| OR | |
| 2018 AND | Jan |
| OR | |
| 2018 AND | Feb |

[Total Revenue] :=
SUMX(fTransactions,fTransactions[Units]*RELATED(dBoomProducts[RetailPrice]))

| Year | Month | Total Revenue |
|---|---|---|
| ⊟2017 | ⊞Nov | $3,220,979.52 |
| | ⊞Dec | $3,553,907.38 |
| 2017 Total | | $6,774,886.90 |
| ⊟2018 | ⊞Jan | $3,440,173.41 |
| | ⊞Feb | $2,776,709.55 |
| 2018 Total | | $6,216,882.96 |
| Grand Total | | $12,991,769.86 |

4. As seen below, the **[Ave Year Rev] Measure** uses the dDate[Year] column in the first argument of the Iterator AVERAGEX. This is the same column that is being used in the External Complex Filter in the Row Area of the PivotTable:



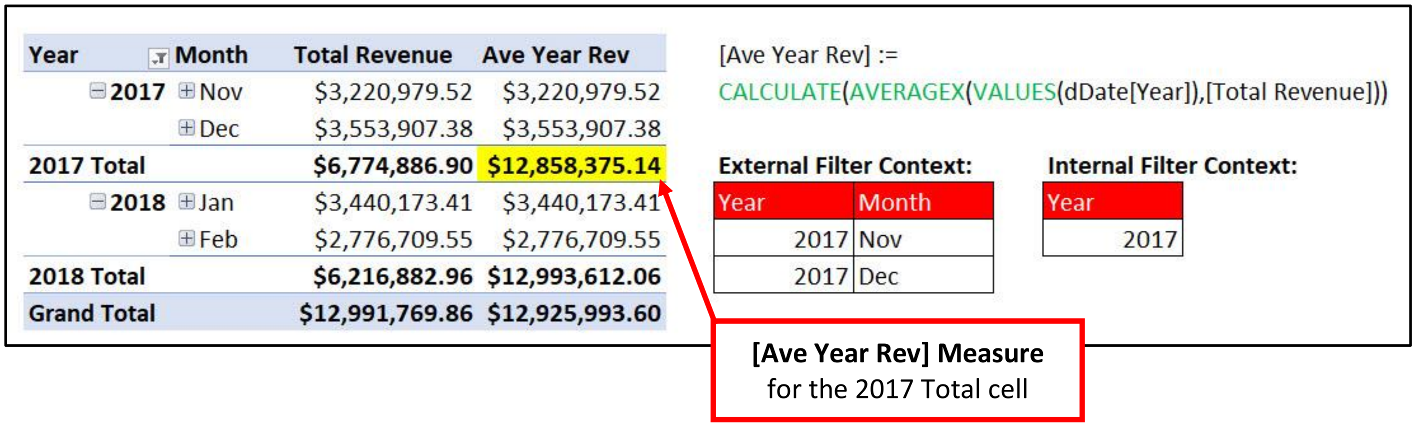
**dDate[Year]** is the same Column used in Internal Filter Context and a External Complex Filter

[Ave Year Rev] :=
AVERAGEX(VALUES(dDate[Year]),[Total Revenue])

| Year | Month | Total Revenue | Ave Year Rev |
|---|---|---|---|
| ⊟2017 | ⊞Nov | $3,220,979.52 | $3,220,979.52 |
| | ⊞Dec | $3,553,907.38 | $3,553,907.38 |
| 2017 Total | | $6,774,886.90 | $12,858,375.14 |
| ⊟2018 | ⊞Jan | $3,440,173.41 | $3,440,173.41 |
| | ⊞Feb | $2,776,709.55 | $2,776,709.55 |
| 2018 Total | | $6,216,882.96 | $12,993,612.06 |
| Grand Total | | $12,991,769.86 | $12,925,993.60 |

5. Once we have these facts, we run the risk of a Complex Filter Reduction Error:
   i. Complex Filter in external report.
   ii. Iterator Measure that uses a column from the External Complex Filter.
   iii. Context Transition for the Measure uses the internal column to overwrite external column, and in the process, we lose the filtering specifications from the External Complex Filter and will iterate over the incorrect number of rows leading to the incorrect result for our Measure.

6. To see how the error occurs, we will first take a look at the External and Internal Filter Context for the **[Ave Year Rev] Measure** in the 2017 Total cell. Remember that a hidden CALCULATE function is wrapped around every Measure and pulls the External Filter Context into the Measure to be merged with the Internal Filter Context. Here is a picture:



| Year | Month | Total Revenue | Ave Year Rev |
|---|---|---|---|
| 2017 | Nov | $3,220,979.52 | $3,220,979.52 |
| | Dec | $3,553,907.38 | $3,553,907.38 |
| **2017 Total** | | **$6,774,886.90** | **$12,858,375.14** |
| 2018 | Jan | $3,440,173.41 | $3,440,173.41 |
| | Feb | $2,776,709.55 | $2,776,709.55 |
| **2018 Total** | | **$6,216,882.96** | **$12,993,612.06** |
| **Grand Total** | | **$12,991,769.86** | **$12,925,993.60** |

```
[Ave Year Rev] :=
CALCULATE(AVERAGEX(VALUES(dDate[Year]),[Total Revenue]))
```

**External Filter Context:**

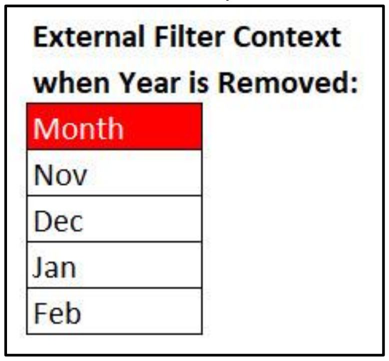
| Year | Month |
|---|---|
| 2017 | Nov |
| 2017 | Dec |

**Internal Filter Context:**

| Year |
|---|
| 2017 |

**[Ave Year Rev] Measure**
for the 2017 Total cell

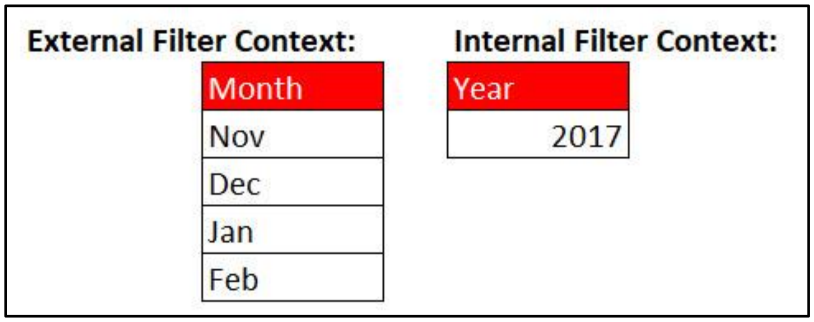
7. In the above picture, realize that the yearly average for a single year (2017) is a single number divided by one. Which seems silly, but we need to learn how this cell is calculated to understand how the error in the Grand Total cell occurs.
8. For the above picture, the External Filter Context contains the Year column and Month column combined into an AND Logical Test. This means that the two columns are related. This relationship between the two columns will be broken when the CALCULATE Overwrite process removes the Year column from the External Complex Filter and replaces it with the internal column.
9. As the first part of the Overwrite process, when the Year is removed from the External Filter Context, because the Month column was related to the Year in a specific way and it no longer has the relationship, all four months are used, as seen here:



**External Filter Context when Year is Removed:**

| Month |
|---|
| Nov |
| Dec |
| Jan |
| Feb |

10. In the second part of the Overwrite process, the Internal Filter Context with the Year column is used in place of the Year column in the external filter, as seen here:



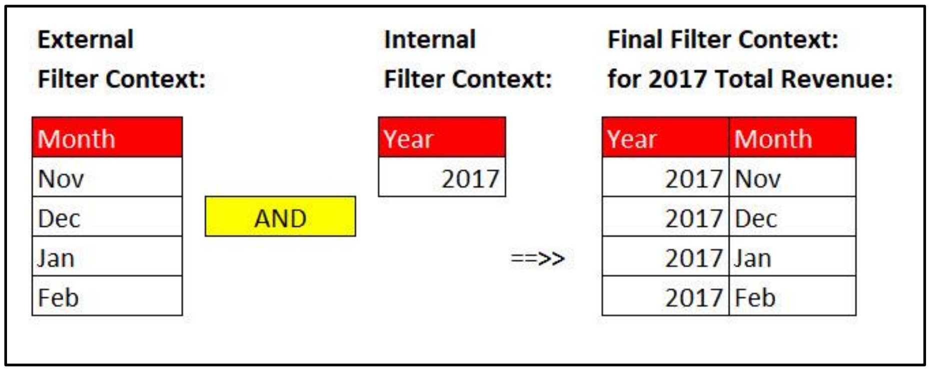
**External Filter Context:**

| Month |
|---|
| Nov |
| Dec |
| Jan |
| Feb |

**Internal Filter Context:**

| Year |
|---|
| 2017 |

11. The last part of the Overwrite process merges the External Filter Context with the Internal Filter Context in an AND Logical Test, as seen here:



**External Filter Context:**

| Month |
|---|
| Nov |
| Dec |
| Jan |
| Feb |

AND

**Internal Filter Context:**

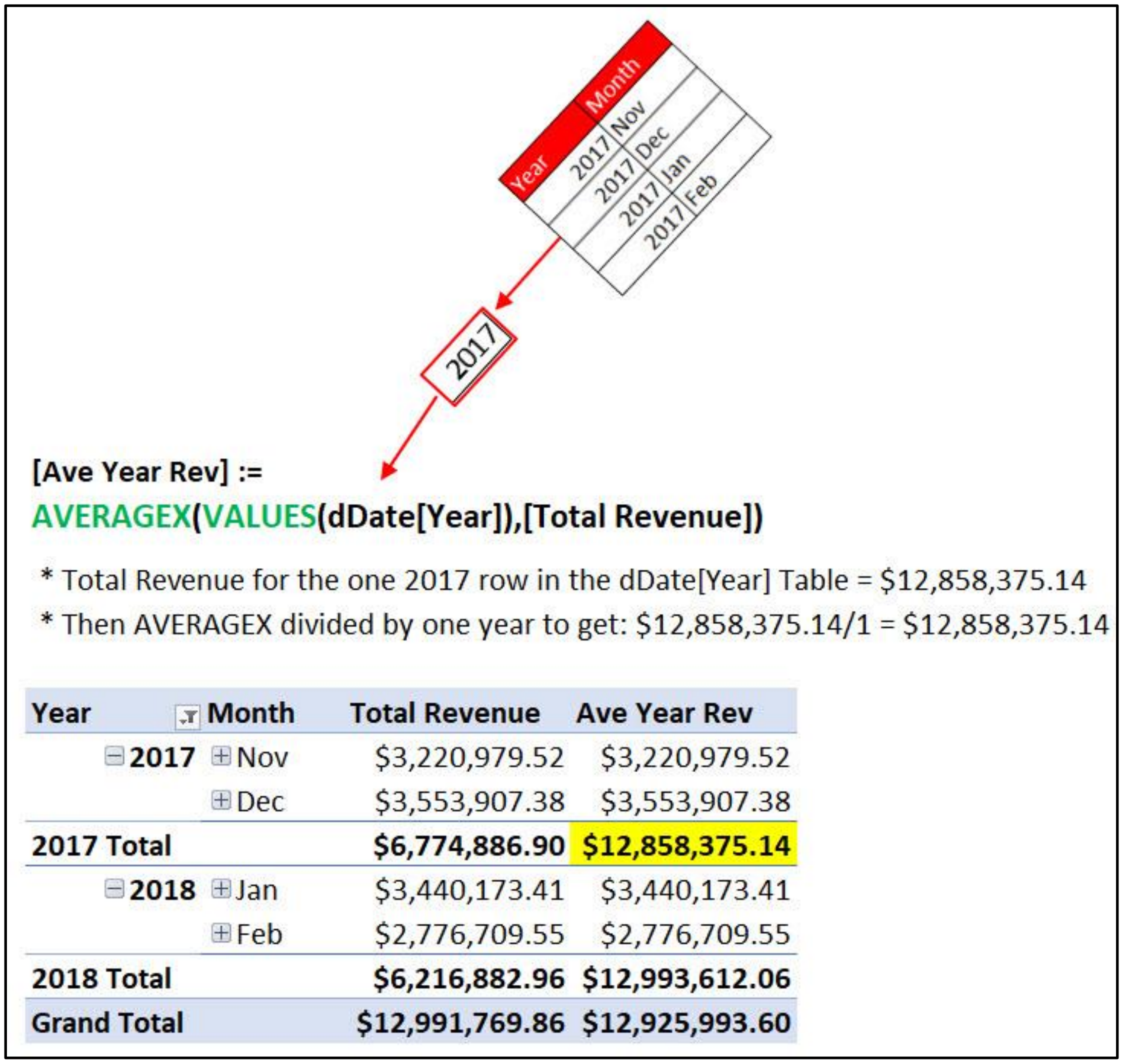
| Year |
|---|
| 2017 |

==>>

**Final Filter Context: for 2017 Total Revenue:**

| Year | Month |
|---|---|
| 2017 | Nov |
| 2017 | Dec |
| 2017 | Jan |
| 2017 | Feb |

12. As seen below in the picture, this process instructs the single year row in the dDate[Year] column (that sits in the VALUES function in the first argument of the AVERAGEX function) to filter the Fact Table down to just the rows for the Year 2017 and the months Jan, Feb, Nov and Dec. The Total Revenue for the one 2017 row in the dDate[Year] Table = $12,858,375.14. Then then the AVERAGEX function divides the single year amount by one to get: $12,858,375.14/1 = $12,858,375.14. This is not the correct answer because it is NOT based on the total revenue for ONLY Nov, 2017 and Dec 2017.

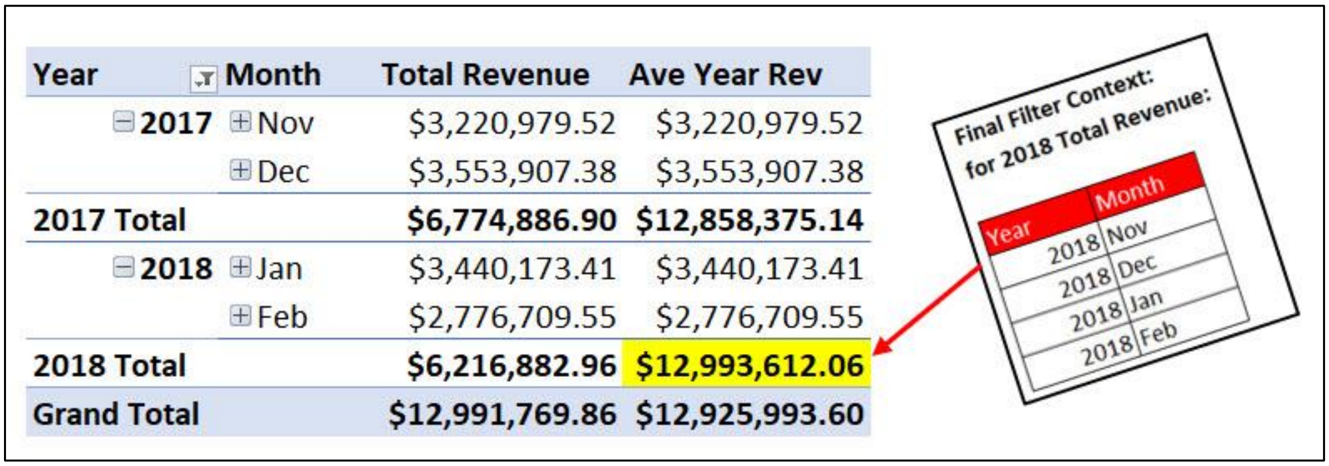


[Ave Year Rev] :=
AVERAGEX(VALUES(dDate[Year]),[Total Revenue])

* Total Revenue for the one 2017 row in the dDate[Year] Table = $12,858,375.14
* Then AVERAGEX divided by one year to get: $12,858,375.14/1 = $12,858,375.14

| Year | Month | Total Revenue | Ave Year Rev |
|---|---|---|---|
| 2017 | Nov | $3,220,979.52 | $3,220,979.52 |
| | Dec | $3,553,907.38 | $3,553,907.38 |
| **2017 Total** | | **$6,774,886.90** | **$12,858,375.14** |
| 2018 | Jan | $3,440,173.41 | $3,440,173.41 |
| | Feb | $2,776,709.55 | $2,776,709.55 |
| **2018 Total** | | **$6,216,882.96** | **$12,993,612.06** |
| **Grand Total** | | **$12,991,769.86** | **$12,925,993.60** |

13. This process is also used to calculate the incorrect amount for the 2018 Total cell, as seen here:

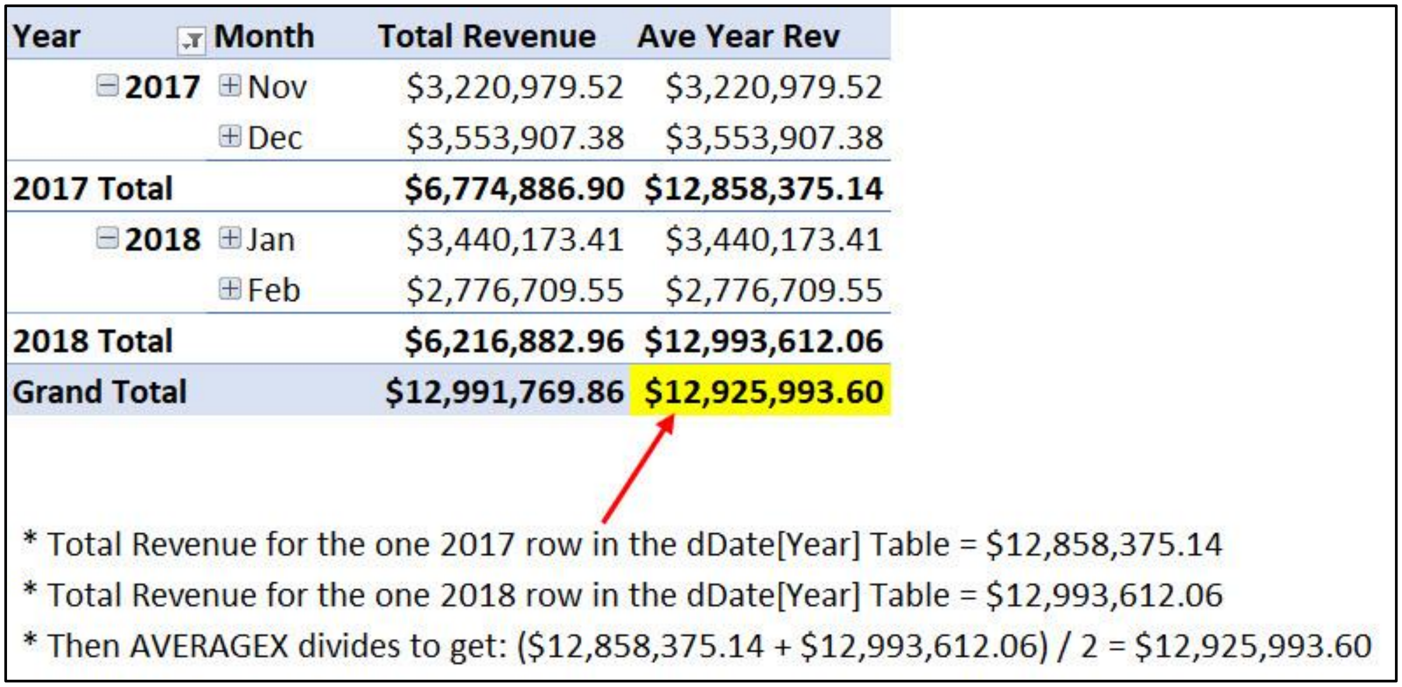


| Year | Month | Total Revenue | Ave Year Rev |
|---|---|---|---|
| 2017 | Nov | $3,220,979.52 | $3,220,979.52 |
| | Dec | $3,553,907.38 | $3,553,907.38 |
| **2017 Total** | | **$6,774,886.90** | **$12,858,375.14** |
| 2018 | Jan | $3,440,173.41 | $3,440,173.41 |
| | Feb | $2,776,709.55 | $2,776,709.55 |
| **2018 Total** | | **$6,216,882.96** | **$12,993,612.06** |
| **Grand Total** | | **$12,991,769.86** | **$12,925,993.60** |

Final Filter Context:
for 2018 Total Revenue:

| Year | Month |
|---|---|
| 2018 | Nov |
| 2018 | Dec |
| 2018 | Jan |
| 2018 | Feb |

14. In the Grand Total, the incorrect average is calculated this way:



| Year | Month | Total Revenue | Ave Year Rev |
|---|---|---|---|
| 2017 | Nov | $3,220,979.52 | $3,220,979.52 |
| | Dec | $3,553,907.38 | $3,553,907.38 |
| **2017 Total** | | **$6,774,886.90** | **$12,858,375.14** |
| 2018 | Jan | $3,440,173.41 | $3,440,173.41 |
| | Feb | $2,776,709.55 | $2,776,709.55 |
| **2018 Total** | | **$6,216,882.96** | **$12,993,612.06** |
| **Grand Total** | | **$12,991,769.86** | **$12,925,993.60** |

* Total Revenue for the one 2017 row in the dDate[Year] Table = $12,858,375.14
* Total Revenue for the one 2018 row in the dDate[Year] Table = $12,993,612.06
* Then AVERAGEX divides to get: ($12,858,375.14 + $12,993,612.06) / 2 = $12,925,993.60

15. In the above picture, the incorrect answer was caused by a Filter Reduction Error that cause the AVERAGEX function to iterate over the incorrect number of rows.

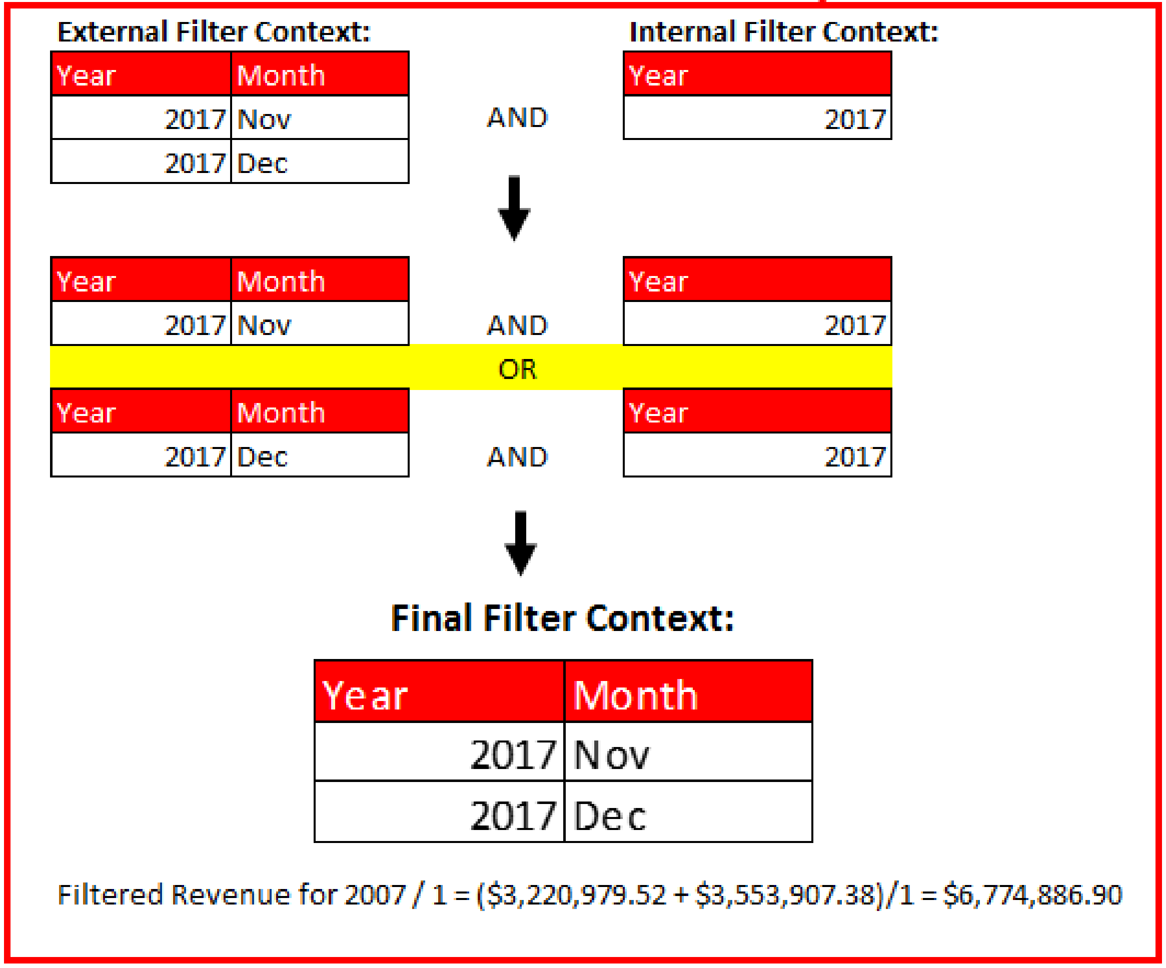
16. The DAX solution to the Filter Reduction Error, is to wrap the KEEPFILTERS function around the Year column filter to force an AND Logical Test to run rather than the Overwrite Operation.
17. As seen on the next page, KEEPFILTERS forces the full External Filter Context and the internal filter on Year to be merged as an AND Logical Text to get the correct average for the filtered yearly amounts of $6,495,884.93.
18. The key to how KEEPFILTERS solves the Complex Filter Reduction Error is that it runs an AND Logical Test rather than a Overwrite Operation, as seen in the 2017 Total cell for the **[Ave Year Rev KF] Measure** in the picture on the next page:

```
[Total Revenue] :=
SUMX(fTransactions,fTransactions[Units]*RELATED(dBoomProducts[RetailPrice]))

[Ave Year Rev] :=
AVERAGEX(VALUES(dDate[Year]),[Total Revenue])

[Ave Year Rev KF] :=
AVERAGEX(KEEPFILTERS(VALUES(dDate[Year])),[Total Revenue])
```

| Year | Month | Total Revenue | Ave Year Rev | Ave Year Rev KF |
|---|---|---|---|---|
| 2017 | Nov | $3,220,979.52 | $3,220,979.52 | $3,220,979.52 |
| | Dec | $3,553,907.38 | $3,553,907.38 | $3,553,907.38 |
| **2017 Total** | | **$6,774,886.90** | **$12,858,375.14** | **$6,774,886.90** |
| 2018 | Jan | $3,440,173.41 | $3,440,173.41 | $3,440,173.41 |
| | Feb | $2,776,709.55 | $2,776,709.55 | $2,776,709.55 |
| **2018 Total** | | **$6,216,882.96** | **$12,993,612.06** | **$6,216,882.96** |
| **Grand Total** | | **$12,991,769.86** | **$12,925,993.60** | **$6,495,884.93** |



**External Filter Context:**

| Year | Month |
|---|---|
| 2017 | Nov |
| 2017 | Dec |

AND

**Internal Filter Context:**

| Year |
|---|
| 2017 |

| Year | Month |
|---|---|
| 2017 | Nov |

AND

| Year |
|---|
| 2017 |

OR

| Year | Month |
|---|---|
| 2017 | Dec |

AND

| Year |
|---|
| 2017 |

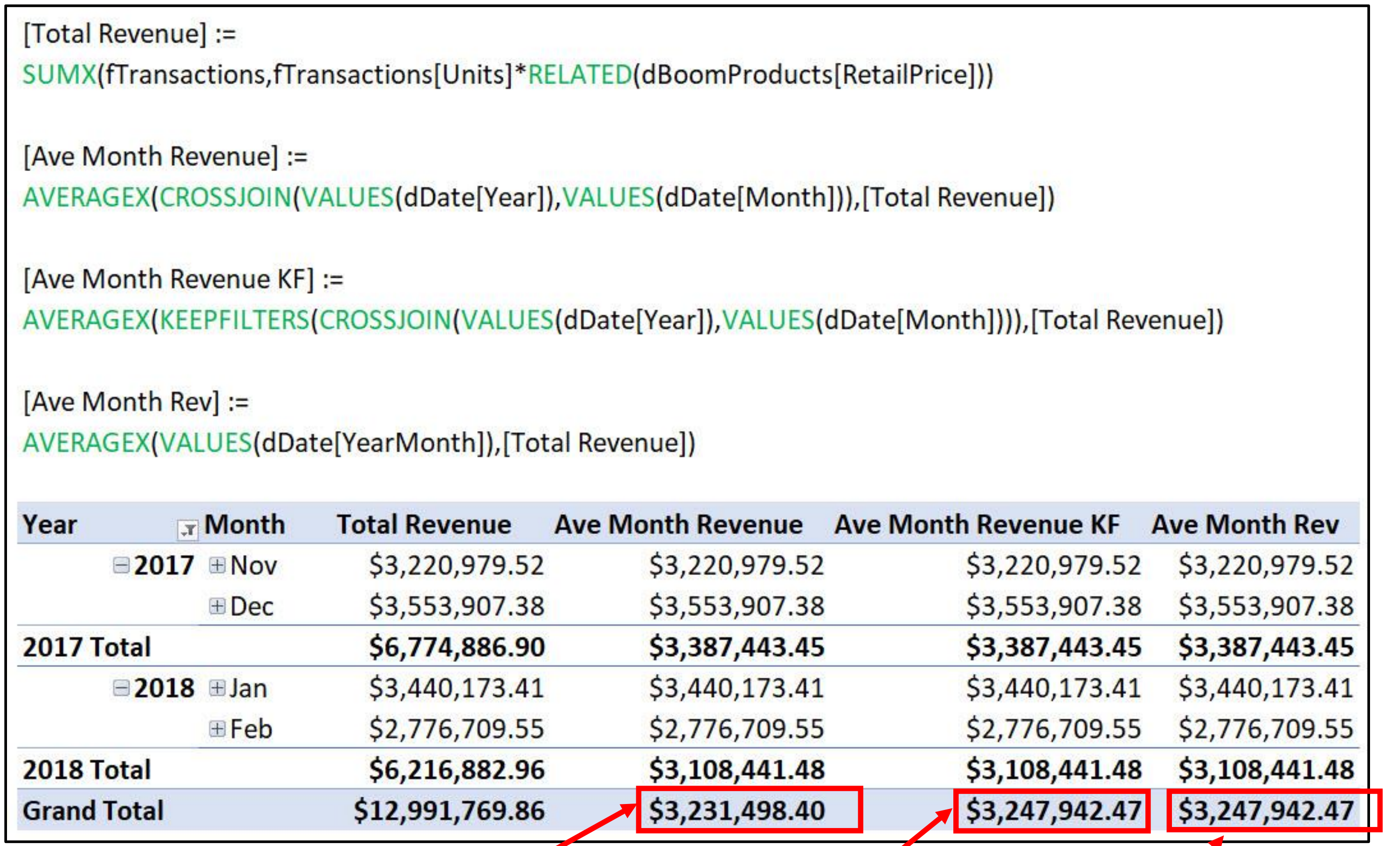
**Final Filter Context:**

| Year | Month |
|---|---|
| 2017 | Nov |
| 2017 | Dec |

Filtered Revenue for 2007 / 1 = ($3,220,979.52 + $3,553,907.38)/1 = $6,774,886.90

xiii. **Example 02 of KEEPFILTERS in first argument of an Iterator to solve Complex Filter Reduction Error** :

1. In the below picture, we see a complex filter on the Year and Month columns in the External Report and we are iterating over the Year & Month columns in the first argument of AVERAGEX to calculate Average Monthly Revenue. The [Ave Month Revenue] Measure makes a Complex Filter Reduction Error. The [Ave Month Revenue KF] Measure does NOT make a Complex Filter Reduction Error because the KEEPFILTERS functions forces an AND Logical test when it merges the filters in the first argument of the AVERAGEX with the External Filter Context. A description for there two Measures and for a Data Modeling Solution for the Complex Filter Reduction Error (rather than a DAX formula solution) are shown in the below picture:



```
[Total Revenue] :=
SUMX(fTransactions,fTransactions[Units]*RELATED(dBoomProducts[RetailPrice]))

[Ave Month Revenue] :=
AVERAGEX(CROSSJOIN(VALUES(dDate[Year]),VALUES(dDate[Month])),[Total Revenue])

[Ave Month Revenue KF] :=
AVERAGEX(KEEPFILTERS(CROSSJOIN(VALUES(dDate[Year]),VALUES(dDate[Month]))),[Total Revenue])

[Ave Month Rev] :=
AVERAGEX(VALUES(dDate[YearMonth]),[Total Revenue])
```

| Year | Month | Total Revenue | Ave Month Revenue | Ave Month Revenue KF | Ave Month Rev |
|---|---|---|---|---|---|
| ⊟2017 | ⊞Nov | $3,220,979.52 | $3,220,979.52 | $3,220,979.52 | $3,220,979.52 |
| | ⊞Dec | $3,553,907.38 | $3,553,907.38 | $3,553,907.38 | $3,553,907.38 |
| **2017 Total** | | **$6,774,886.90** | **$3,387,443.45** | **$3,387,443.45** | **$3,387,443.45** |
| ⊟2018 | ⊞Jan | $3,440,173.41 | $3,440,173.41 | $3,440,173.41 | $3,440,173.41 |
| | ⊞Feb | $2,776,709.55 | $2,776,709.55 | $2,776,709.55 | $2,776,709.55 |
| **2018 Total** | | **$6,216,882.96** | **$3,108,441.48** | **$3,108,441.48** | **$3,108,441.48** |
| **Grand Total** | | **$12,991,769.86** | **$3,231,498.40** | **$3,247,942.47** | **$3,247,942.47** |

For the **[Ave Month Revenue] Measure** in the Grand Total cell, the Internal Columns for Year and Month replace the External Complex Filter Columns of Year and Month. This means that the CROSSJOIN Function produces eight rows for the [Total Revenue] Measure to iterate over: 2017-Nov, 2017-Dec, 2017-Jan, 2017-Feb, 2018-Nov, 2018-Dec, 2018-Jan, 2018-Feb, rather than the correct four rows containing the months 2017-Nov, 2017-Dec, 2018-Jan, 2018-Feb.

$3,231,498.4 is NOT correct.

For the **[Ave Month Revenue KF] Measure** in the Grand Total cell, the Internal Columns for Year and Month are merged in an AND Logical Test with the External Complex Filter Columns of Year and Month. The correct four rows containing 2017-Nov, 2017-Dec, 2018-Jan, 2018-Feb are used to calculate monthly revenue, and then those amounts are used by the AVERAGEX function.

$3,247,942,47 is correct.

The final Measure, **[Ave Month Rev]**, is perhaps the best way to deal with a Complex Filter Reduction Error. The problem was solved in the Data Modeling phase of our project. Rather than risk using the same Year and Month Column from the PivotTable inside the Measure, we create a separated YearMonth column in the Date Table that: 1) creates the correct grain for our iterator, and, 2) will not be involved in an Overwrite Operations because it is a different column than the Year and Month.

15) **Expanded Table Concept & Relationships** :

- i. Assumptions and Notes before covering Expanded Tables :
    1. When we think of Relationships between tables, we know that filters flow from the One-Side to the Many-Side, just as the arrow points.
    2. This discussion about Expanded Tables will not cover Expanded Tables with Bi-Directional Filtering. In this class we will not be use Bi-Directional Filters that are available in Power BI Desktop. There are other ways to accomplish Bi-Directional Filtering that have less risk (risk of ambiguous model). When we need to get a filter to flow from the Many-Side to the One-Side without using Bi-directional Filtering, we will use the DAX CROSSFILTER function or Table Filters.
    3. This discussion about Expanded Tables will mostly avoid talking about Snow Flake Data Models. In this class we are creating Star Schema Data Models that are de-normalized, not Snow Flake Data Models. It is easy enough for us to use Power Query in the Data Modeling phase of a project to convert Snow Flake to Star Schema Data Models. With Star Schema Data Models, DAX Formulas are easier, we don't have PivotTable "Auto-Exist" Problems and Expanded Table diagrams are less complicated.
- ii. Expanded Table Concept & Relationships :
    1. In DAX, every table has a corresponding expanded table, which contain all columns from the table itself plus all columns from the tables that can filter the original through a one-to-many relationship. See diagrams on next two pages.
    2. In a Star Schema Data Model with a Fact Table and Dimension Tables with One-To-Many Relationships, this means that the Fact Table will contain all columns in the Data Model.
    3. The Expanded Table diagram can be helpful because:
        - i. It shows a complete list of all columns, that when filtered, will filter the table.
        - ii. It allows you to see that when you use a Table as a filter in the Filter Argument of CALCULATE, all the columns in that table will be used as filters.
        - iii. The implication of Expanded Tables is that when we use a table as a filter, this allows us to send a filter from the Many-Side to the One-Side.
    4. Marco Russo and Alberto Ferrari use a particular Expanded Table Diagram in their Definitive DAX book that helps to visualize Expanded Tables. On the next two pages you can see a picture of our Star Schema Data Model using an Expanded Tables diagram and of a Snow Flake Data Model using an Expanded Tables diagram.
- ii. Files we will use the Expanded Table Examples are these:
    1. "019-StarSchemaDataModel.xlsx"
    2. "019-SnowFlakeDataModel.xlsx"

**Star Schema Data Model and Expanded Table Diagram:**

Color coding:

| | |
|---|---|
| 🟨 | Native Column = Columns in actual table |
| 🟦 | Expanded Columns (Derived Columns) = columns that are in Expanded Tables |

**Expanded Table Diagram Helps:**

**1) Which Tables will a "Column Filter" affect**

ALL(fTransactions[Units]) works only on that single column

ALL(dProduct[Product]) works on dProdct & fTransactions tables

**2) Which Columns will a "Table Filter" affect**

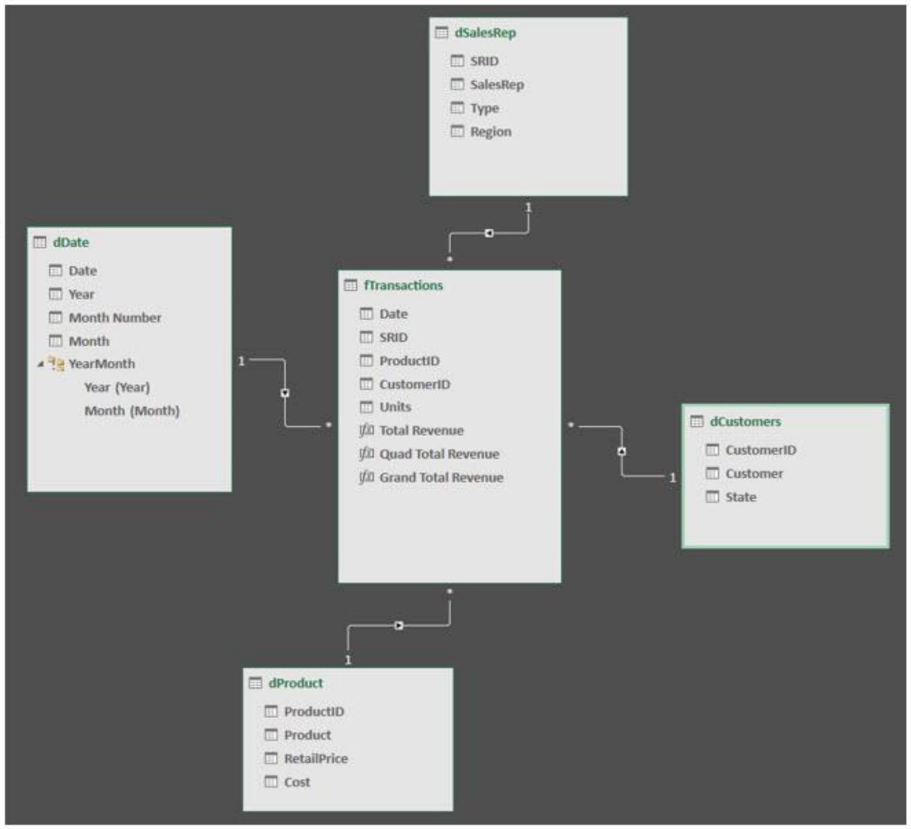
ALL(fTransactions) as filter in CALC will remove all columns as filters from Expanded Table

fTransactions as a filter in CALC can send a filter backwards across Many-To-One Relationship

**3) See which Expanded Table Columns are in play with any Table Filter**

ALLEXCEPT(fTransactions,dDate[Date]) will provide full Expanded Table as filter without the dDate[Date] column

| | dSalesRep | dCustomers | dProduct | dDate | fTransactions |
|---|---|---|---|---|---|
| SRID | 🟨 | | | | 🟦 |
| SalesRep | 🟨 | | | | 🟦 |
| Type | 🟨 | | | | 🟦 |
| Region | 🟨 | | | | 🟦 |
| CustomerID | | 🟨 | | | 🟦 |
| Customer | | 🟨 | | | 🟦 |
| State | | 🟨 | | | 🟦 |
| ProductID | | | 🟨 | | 🟦 |
| Product | | | 🟨 | | 🟦 |
| RetailPrice | | | 🟨 | | 🟦 |
| Cost | | | 🟨 | | 🟦 |
| Date | | | | 🟨 | 🟦 |
| Year | | | | 🟨 | 🟦 |
| Month Number | | | | 🟨 | 🟦 |
| Month | | | | 🟨 | 🟦 |
| Date | | | | | 🟨 |
| SRID | | | | | 🟨 |
| ProductID | | | | | 🟨 |
| CustomerID | | | | | 🟨 |
| Units | | | | | 🟨 |



**Snow Flake Data Model and Expanded Table Diagram:**

Color coding:

| | |
|---|---|
| 🟨 | Native Column = Columns in actual table |
| 🟦 | Expanded Columns (Derived Columns) = columns that are in Expanded Tables |

**Expanded Table Diagram Helps:**

**1) Which Tables will a "Column Filter" affect**

ALL(fTransactions[Units]) works only on that single column

ALL(dProduct[Product]) works on dProdct & fTransactions tables

**2) Which Columns will a "Table Filter" affect**

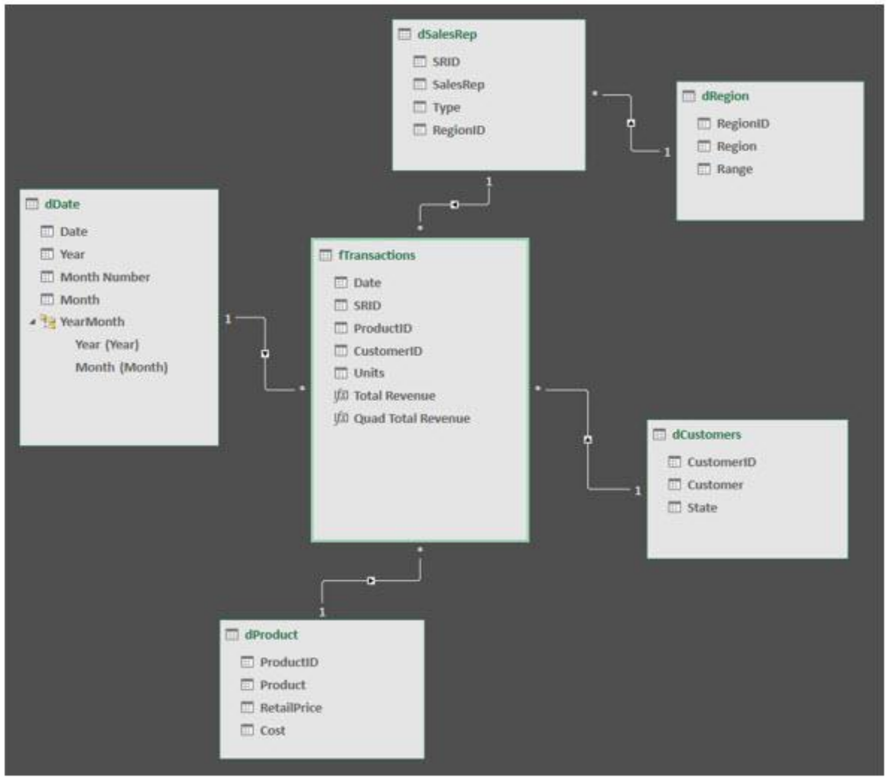
ALL(fTransactions) as filter in CALC will remove all columns as filters from Expanded Table

fTransactions as a filter in CALC can send a filter backwards across Many-To-One Relationship

**3) See which Expanded Table Columns are in play with any Table Filter**

ALLEXCEPT(fTransactions,dDate[Date]) will provide full Expanded Table as filter without the dDate[Date] column

| | dRegion | dSalesRep | dCustomers | dProduct | dDate | fTransactions |
|---|---|---|---|---|---|---|
| RegionID | 🟨 | 🟦 | | | | 🟦 |
| Region | 🟨 | 🟦 | | | | 🟦 |
| Range | 🟨 | 🟦 | | | | 🟦 |
| SRID | | 🟨 | | | | 🟦 |
| SalesRep | | 🟨 | | | | 🟦 |
| Type | | 🟨 | | | | 🟦 |
| RegionID | | 🟨 | | | | 🟦 |
| CustomerID | | | 🟨 | | | 🟦 |
| Customer | | | 🟨 | | | 🟦 |
| State | | | 🟨 | | | 🟦 |
| ProductID | | | | 🟨 | | 🟦 |
| Product | | | | 🟨 | | 🟦 |
| RetailPrice | | | | 🟨 | | 🟦 |
| Cost | | | | 🟨 | | 🟦 |
| Date | | | | | 🟨 | 🟦 |
| Year | | | | | 🟨 | 🟦 |
| Month Number | | | | | 🟨 | 🟦 |
| Month | | | | | 🟨 | 🟦 |
| Date | | | | | | 🟨 |
| SRID | | | | | | 🟨 |
| ProductID | | | | | | 🟨 |
| CustomerID | | | | | | 🟨 |
| Units | | | | | | 🟨 |



17) **Table Filters & Expanded Table as filters in the Filter argument of CALCULATE**:

   i. When we use Table Filters in the Filter argument of the CALCULATE function, all columns in that table will have an affect.

      1. Examples:

         i. ALL Table Filter:

            CALCULATE([Total Sales],ALL(fTransactions)) will remove all columns from the Expanded fTransactions Table and therefore the Data Model.

         ii. ALL Column Filter:

            CALCULATE([Total Sales],ALL(dProduct[Product])) will affect the [Product] column in the dProduct table and the [Product] column in the fTransactions Expanded table.

         iii. Table Filter to go Backwards Across Many-To-One Relationship:

            CALCULATE(DISTINCTCOUNT(dDate[Month]),fTransactions) will allow all columns in the Expanded fTransactions Table to flow to the dDate Table. If you have a PivotTable Row Area filter for a Product flow into the Measure, the formula will count the number of unique months that the product was sold.

         iv. Use Expanded Column from Expanded Table (Column not in original table!):

            ALLEXCEPT(fTransactions,dDate[Month])) is an example of using the full Expanded fTransactions Table to exclude a column from different table. The ALL will remove all filters from the Expanded fTransactions Table, except for the filters on the dDate[Month] column.

   ii. Examples from video on next page:

**dDate Table**

| Date | Month |
|---|---|
| 1/1/2019 | Jan |
| 1/2/2019 | Jan |
| 1/3/2019 | Jan |
| 2/1/2019 | Feb |
| 2/2/2019 | Feb |
| 2/3/2019 | Feb |

**fTransactions Table**

| Date | Product | Sales |
|---|---|---|
| 1/1/2019 | Quad | 100 |
| 2/1/2019 | Quad | 150 |
| 2/1/2019 | Quad | 250 |
| 1/1/2019 | Carlota | 75 |
| 1/2/2019 | Carlota | 145 |
| 1/3/2019 | Carlota | 230 |

**dProduct Table**

| Product | Price |
|---|---|
| Quad | 43 |
| Carlota | 29 |

**Expanded Table Diagram:**

| | | Tables | | |
|---|---|---|---|---|
| | | dDate Table | dProduct Table | fTransactions Table |
| Columns | Date | 🟦 | | 🟦 |
| | Month | 🟦 | | 🟦 |
| | Product | | 🟩 | 🟩 |
| | Price | | 🟩 | 🟩 |
| | Date | | | 🟧 |
| | Product | | | 🟧 |
| | Sales | | | 🟧 |

**Expanded Table Diagram Helps to:**

1) Determine which tables a single column filters will affect (look to see if table contains the column)
   ALL(fTransactions[Sales]) works only on that single column
   ALL(dProduct[Product]) works on dProduct & fTransactions tables
2) See all the columns that will be affected if you use Full Table as a Filter
   ALL(fTransactions) as filter in CALC will remove all columns as filters from Expanded Table
   fTransactions as a filter in CALC can send a filter backwards across Many-To-One Relationship
3) See which Expanded Table Columns are in play with any Table Filter
   ALLEXCEPT(fTransactions,dDate[Month]) will provide full Expanded Table as filter without the dDate[Month] column

**Table Filters in CALCULATE:**

1) Remove all columns in Expanded Table:
   [Grand Overall Total] := CALCULATE([Total Sales],ALL(fTransactions))

2) Use Expanded fTransations Table to pass a filter backward across a Many-To-One Relationship
   [Count Distinct Months Prod. Sold] := CALCULATE(DISTINCTCOUNT(dDate[Month]),fTransactions)

3) Use Expanded Table Column not in original Table
   [Remove Everything Except Month] := CALCULATE([Distinct Months],ALLEXCEPT(fTransactions,dDate[Month]))

**Column Filter in CALCULATE**

[ALL works on Prod. Column Only] :=CALCULATE([Total Sales],ALL(dProduct[Product]))

| Product | Total Sales | Grand Overall Total | ALL works on Prod. Column Only | Distinct Months | Count Distinct Months Prod. Sold |
|---|---|---|---|---|---|
| Carlota | $450.00 | $950.00 | $950.00 | 2 | 1 |
| Quad | $500.00 | $950.00 | $950.00 | 2 | 2 |
| **Grand Total** | **$950.00** | **$950.00** | **$950.00** | **2** | **2** |

| Month | Distinct Months | Remove Everything With ALL | Remove Everything Except Month |
|---|---|---|---|
| Feb | 1 | 2 | 1 |
| Jan | 1 | 2 | 1 |
| **Grand Total** | **2** | **2** | **2** |

| Date |
|---|
| 1/1/19 |
| 1/2/19 |
| 1/3/19 |
| 2/1/19 |
| 2/2/19 |
| 2/3/19 |

**All DAX Formulas in Data Model:**

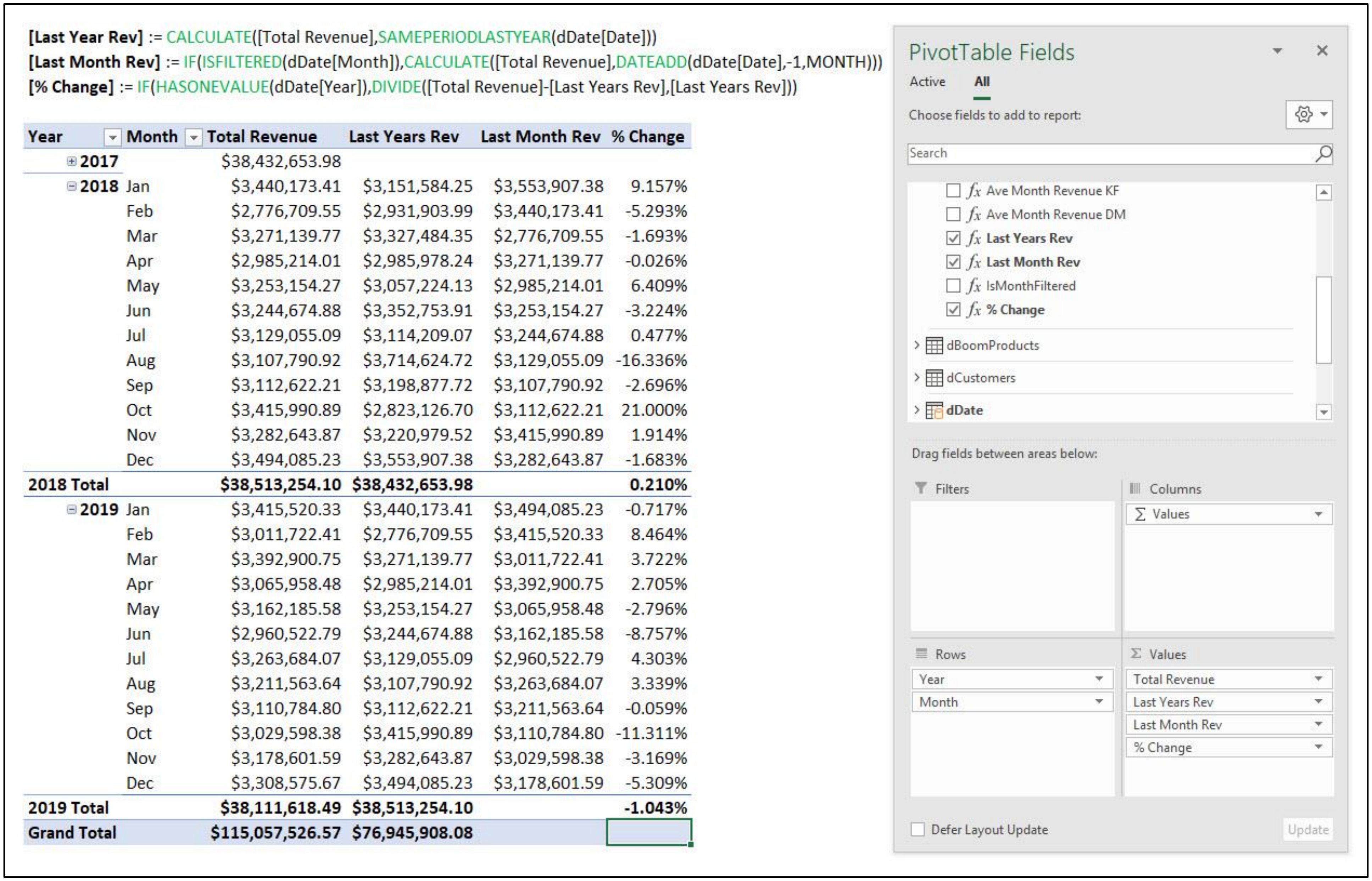
```
[Total Sales] := SUM(fTransactions[Sales])
[Grand Overall Total] := CALCULATE([Total Sales],ALL(fTransactions))
[ALL works on Prod. Column Only] : =CALCULATE([Total Sales],ALL(dProduct[Product]))
[Distinct Months] : =DISTINCTCOUNT(dDate[Month])
[Count Distinct Months Prod. Sold] := CALCULATE(DISTINCTCOUNT(dDate[Month]),fTransactions)
[RemoveDate] := CALCULATE([Distinct Months],ALL(fTransactions))
[Remove Everything Except Month] :=CALCULATE([Distinct Months],ALLEXCEPT(fTransactions,dDate[Month]))
```

17) **DAX Formula Evaluation Context Summary** :

- i. There are Two Evaluation Contexts:
    1. Row Context = allows a formula in a Calculated Column or an Iterator Function or in a PivotTable/Power BI Visualization to see the row and use the values from the row to make a Row-By-Row Calculation.
    2. Filter Context = all the Filters / Conditions / Criteria that filter the underlying tables in the Data Model to provide the final values for the Measure to use to calculate the final answer.
- ii. CALCULATE and CALCULATETABLE DAX functions can do these two things:
    1. Change the Filter Context.
    2. Perform Context Transition, which takes all available Rows Contexts and merges them with an AND Logical Test and converts then to Filter Context.
- iii. All Measures have a hidden CALCULATE function wrapped around it.
- iv. There are two types of Filter Contexts that are used to determine the Final Filter Context under which the Measure makes its final calculation:
    1. External Filter Context = Filters / Conditions / Criteria from Excel PivotTables or Power BI Visualizations.
    2. Internal Filter Context = Filters / Conditions / Criteria from inside the CALCULATE function.
- v. How Final Filter Context is determined:
    1. Filters / Conditions / Criteria from Excel PivotTables or Power BI Visualizations flow into a Measure.
    2. Inside the Measure the internal and external filters are merged into the Final Filter Context using the operators:
        - i. And Logical Test (Intersect)
        - ii. Overwrite
        - iii. Remove
- vi. When the ALL functions is used in a CALCULATE Filter argument, all the filters for the column, columns or table are removed and become an empty filter.
- vii. When Complex Filters exist in the External Filter Context and the same columns are used in the first argument of an Iterator function, then you can use KEEPFILTERS to perform an AND Logical Test rather than Overwrite.
- viii. ALLSELECTED removes the last filter generated from context transition. When the last filter generated by Context Transition is the Row Context in a PivotTable, ALLSELECTED() can help to calculate the correct filtered Grand Total amount.
- ix. Column filters work on just the column.
- x. Table filters work on Expanded Table and can go backwards across One-To-Many Relationship.

25) **Examples of Time Intelligence Functions as filters in CALCULATE** :

- i. Time Intelligence Functions work with Date and Time and require a Date or Time Table in order to work correctly.
- ii. A Few Examples of Time Intelligence Functions:
    1. SAMEPERIODLASTYEAR
        - i. This function Returns a table of dates.
        - ii. When you provide the Primary Key Column from a Date Table, SAMEPERIODLASTYEAR will see the External Filter Context for the date, whether or not it is a day, month, quarter or year, and it will jump back one year and get the correct "One-Year-Back" Dates and provide it as a valid list of dates to the CALCULATE function. CALCULATE will change the Filter Context and the Final Filter Context will be the dates for that "One-Year-Back" period. The underlying Date table will be filtered to those "One-Year-Back" Dates and the Measure will make its calculation based on those "One-Year-Back" Dates.
    2. DATEADD
        - i. This function Returns a table of dates.
        - ii. The DATADD function can move forward or backwards for a certain interval (Day, Month or Year). It will provide a valid list of dates to the CALCULATE function so the Measure can make the calculation based on the Change Filter Context.
    3. TOTALYTD
        - i. This function calculates cumulative totals.
        - ii. This function can see the External Filter Context for Dates and change the internally create a list of dates for a running, cumulative total, then create the correct Final Filter Context to the measure can calculate a cumulative total. We saw this function back in video#15.
    4. DATESINPERIOD
        - i. This function Returns a table of dates.
        - ii. Returns a table that contains a column of dates that begins with a specified start date and continues for the specified number of intervals. We used this DAX function in video #16 to construct a set of dates for a 12-month running average.
    5. LASTDATE DAX
        - i. This function Returns the last date that is listed in the External Filter Context.
        - ii. We used this DAX function in video #16 to construct a set of dates for a 12-month running average.
- iii. We will see more Time Intelligence functions later in the class.
- iv. Example from the video shown on next page:
- v.



**[Last Year Rev]** := CALCULATE([Total Revenue],SAMEPERIODLASTYEAR(dDate[Date]))
**[Last Month Rev]** := IF(ISFILTERED(dDate[Month]),CALCULATE([Total Revenue],DATEADD(dDate[Date],-1,MONTH)))
**[% Change]** := IF(HASONEVALUE(dDate[Year]),DIVIDE([Total Revenue]-[Last Years Rev],[Last Years Rev]))

| Year | Month | Total Revenue | Last Years Rev | Last Month Rev | % Change |
|---|---|---|---|---|---|
| ⊞ **2017** | | $38,432,653.98 | | | |
| ⊟ **2018** | Jan | $3,440,173.41 | $3,151,584.25 | $3,553,907.38 | 9.157% |
| | Feb | $2,776,709.55 | $2,931,903.99 | $3,440,173.41 | -5.293% |
| | Mar | $3,271,139.77 | $3,327,484.35 | $2,776,709.55 | -1.693% |
| | Apr | $2,985,214.01 | $2,985,978.24 | $3,271,139.77 | -0.026% |
| | May | $3,253,154.27 | $3,057,224.13 | $2,985,214.01 | 6.409% |
| | Jun | $3,244,674.88 | $3,352,753.91 | $3,253,154.27 | -3.224% |
| | Jul | $3,129,055.09 | $3,114,209.07 | $3,244,674.88 | 0.477% |
| | Aug | $3,107,790.92 | $3,714,624.72 | $3,129,055.09 | -16.336% |
| | Sep | $3,112,622.21 | $3,198,877.72 | $3,107,790.92 | -2.696% |
| | Oct | $3,415,990.89 | $2,823,126.70 | $3,112,622.21 | 21.000% |
| | Nov | $3,282,643.87 | $3,220,979.52 | $3,415,990.89 | 1.914% |
| | Dec | $3,494,085.23 | $3,553,907.38 | $3,282,643.87 | -1.683% |
| **2018 Total** | | **$38,513,254.10** | **$38,432,653.98** | | **0.210%** |
| ⊟ **2019** | Jan | $3,415,520.33 | $3,440,173.41 | $3,494,085.23 | -0.717% |
| | Feb | $3,011,722.41 | $2,776,709.55 | $3,415,520.33 | 8.464% |
| | Mar | $3,392,900.75 | $3,271,139.77 | $3,011,722.41 | 3.722% |
| | Apr | $3,065,958.48 | $2,985,214.01 | $3,392,900.75 | 2.705% |
| | May | $3,162,185.58 | $3,253,154.27 | $3,065,958.48 | -2.796% |
| | Jun | $2,960,522.79 | $3,244,674.88 | $3,162,185.58 | -8.757% |
| | Jul | $3,263,684.07 | $3,129,055.09 | $2,960,522.79 | 4.303% |
| | Aug | $3,211,563.64 | $3,107,790.92 | $3,263,684.07 | 3.339% |
| | Sep | $3,110,784.80 | $3,112,622.21 | $3,211,563.64 | -0.059% |
| | Oct | $3,029,598.38 | $3,415,990.89 | $3,110,784.80 | -11.311% |
| | Nov | $3,178,601.59 | $3,282,643.87 | $3,029,598.38 | -3.169% |
| | Dec | $3,308,575.67 | $3,494,085.23 | $3,178,601.59 | -5.309% |
| **2019 Total** | | **$38,111,618.49** | **$38,513,254.10** | | **-1.043%** |
| **Grand Total** | | **$115,057,526.57** | **$76,945,908.08** | | |

PivotTable Fields

Active **All**

Choose fields to add to report:

Search

- ☐ *fx* Ave Month Revenue KF
- ☐ *fx* Ave Month Revenue DM
- ☑ *fx* **Last Years Rev**
- ☑ *fx* **Last Month Rev**
- ☐ *fx* IsMonthFiltered
- ☑ *fx* **% Change**
- dBoomProducts
- dCustomers
- **dDate**

Drag fields between areas below:

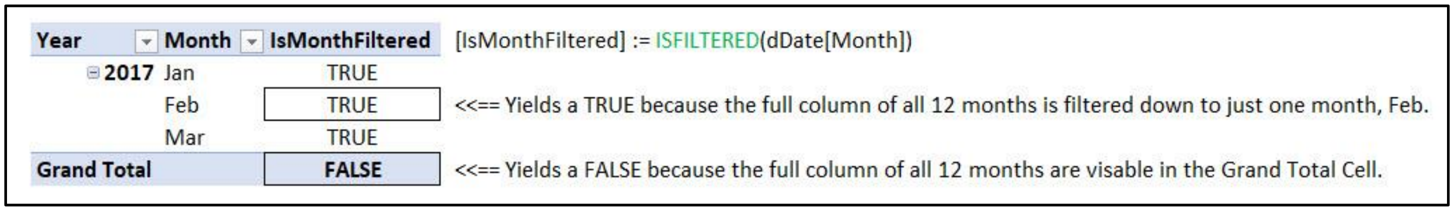
| Filters | Columns |
|---|---|
| | Σ Values |

| Rows | Values |
|---|---|
| Year | Total Revenue |
| Month | Last Years Rev |
| | Last Month Rev |
| | % Change |

☐ Defer Layout Update Update

26) **ISFILTERED, HASONEVALUE, BLANK, COUNTROWS DAX Functions** :

i. ISFILTERED
   1. This function delivers a TRUE if the column is filtered and FALSE if the column is not filtered.
   2. Example:



| Year | Month | IsMonthFiltered | |
|---|---|---|---|
| | | | [IsMonthFiltered] := ISFILTERED(dDate[Month]) |
| 2017 | Jan | TRUE | |
| | Feb | TRUE | <<== Yields a TRUE because the full column of all 12 months is filtered down to just one month, Feb. |
| | Mar | TRUE | |
| Grand Total | | FALSE | <<== Yields a FALSE because the full column of all 12 months are visable in the Grand Total Cell. |

ii. HASONEVALUE
   1. HASONEVALUE is a DAX function that yields a TRUE when a Column shows one value and a FALSE when the column shows more than one value.
   2. The HASONEVALUE function is often used when you need a FALSE value in a Grand Total Cell.

iii. BLANK
   1. BLANK() DAX function is like a Null in a Database or Power Query or like an Empty Cell in an Excel Spreadsheet. It is not a "Zero Length Text String" (Two Double Quotes in an Excel formula like: "" ).
   2. The BLANK() DAX function can be used as a function that is typed into a formula, and it is automatically uses in the third argument of both the IF and DIVIDE DAX Functions.

iv. COUNTROWS
   1. This functions delivers a count of rows in a table. It is a helpful function when you want Filter Context to filter a Table and then determine how many rows are in the table. Seen in this video to calculate the frequency on a Frequency Distribution calculation where we count between a lower and upper limit.