Power Pivot and Power BI:
How the DAX Engine Calculates Measures

powerpivot(pro)
LEADING THE DATA REVOLUTION
T: +1 440.719.9000 | E: simple@powerpivotpro.com | W: www.powerpivotpro.com

1

Total Sales	Year	
Model	2014	2015
Mountain-200		$807,309
Road-150	$2,601,402	$2,948,494
Road-250		$1,571,598

IMPORTANT: Every single measure cell is calculated independently, as an island! (That's right, even the Grand Total cells!) So when a measure returns an unexpected result, we should pick ONE cell and step through it, starting with Step 1 here...

Determine Coordinates of Current Measure Cell:
Calendar[Year]=2015, Products[Model]="Road-150")
Those are the initial *filter context*.

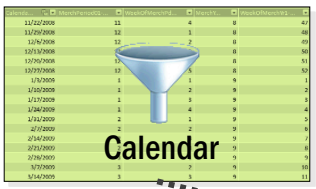
2

If applicable, apply <filters> from CALCULATE(), adding/removing /modifying coordinates and producing a new filter context.

3

Apply the coordinates in the filter context to each of the respective tables (Calendar and Products in this example). This results in a set of "active" rows in each of those tables.

4


Calendar


Products

If the filtered tables (Calendar and Products) are Lookup tables, follow relationships to their related Data tables and filter those tables too. Only Data rows related to active Lookup rows will remain active.

OrderQ...	OrderDate	UnitPr...	ProductKey	SalesAmt
1	1/1/2015	3578.27	313	3578.27
1	1/2/2015	3578.27	312	3578.27
1	1/3/2015	3374.99	350	3374.99
1	1/3/2015	3399.99	345	3399.99
1	1/3/2015	3578.27	310	3578.27
1	1/1/2015	699.0982	338	699.0982
1	1/2/2015	3578.27		

Data Table (Ex: Sales)

5

Once all filters are applied and all relationships have been followed, evaluate the arithmetic– SUM(), COUNTROWS(), etc. in the formula against the remaining active rows.

6

The result of the arithmetic is returned to the current measure cell in the pivot (or dashboard, etc.), then the process starts over at step 1 for the next measure cell.

2 Exercises for Step 1 (Filter Context) of DAX Measure Evaluation Steps

In each of the 9 pivots below, identify the filter context (the set of coordinates coming from the pivot) for the circled cell. (We find that coordinate identification often trips people up, hence this exercise).

In 1-4, the Territories[Country] column is on Rows, & Products[Category] on Columns. [Total Sales] is on Values.

1

Total Sales	Products[Category]			
Territories[Country]	Accessories	Bikes	Clothing	Grand Total
Australia	$138,691	$8,852,050	$70,260	$9,061,001
Canada	$103,378	$1,821,302	$53,165	$1,977,845
France	$63,407	$2,553,576	$27,035	$2,644,018
Germany	$62,233	$2,808,514	$23,565	$2,894,312
United Kingdom	$76,630	$3,282,843	$32,240	$3,391,712
United States	$256,422	$8,999,860	$133,508	$9,389,790
Grand Total	**$700,760**	**$28,318,145**	**$339,773**	**$29,358,677**

2

Total Sales	Products[Category]			
Territories[Country]	Accessories	Bikes	Clothing	Grand Total
Australia	$138,691	$8,852,050	$70,260	$9,061,001
Canada	$103,378	$1,821,302	$53,165	$1,977,845
France	$63,407	$2,553,576	$27,035	$2,644,018
Germany	$62,233	$2,808,514	$23,565	$2,894,312
United Kingdom	$76,630	$3,282,843	$32,240	$3,391,712
United States	$256,422	$8,999,860	$133,508	$9,389,790
Grand Total	**$700,760**	**$28,318,145**	**$339,773**	**$29,358,677**

3

Total Sales	Products[Category]			
Territories[Country]	Accessories	Bikes	Clothing	Grand Total
Australia	$138,691	$8,852,050	$70,260	$9,061,001
Canada	$103,378	$1,821,302	$53,165	$1,977,845
France	$63,407	$2,553,576	$27,035	$2,644,018
Germany	$62,233	$2,808,514	$23,565	$2,894,312
United Kingdom	$76,630	$3,282,843	$32,240	$3,391,712
United States	$256,422	$8,999,860	$133,508	$9,389,790
Grand Total	**$700,760**	**$28,318,145**	**$339,773**	**$29,358,677**

4

Total Sales	Products[Category]			
Territories[Country]	Accessories	Bikes	Clothing	Grand Total
Australia	$138,691	$8,852,050	$70,260	$9,061,001
Canada	$103,378	$1,821,302	$53,165	$1,977,845
France	$63,407	$2,553,576	$27,035	$2,644,018
Germany	$62,233	$2,808,514	$23,565	$2,894,312
United Kingdom	$76,630	$3,282,843	$32,240	$3,391,712
United States	$256,422	$8,999,860	$133,508	$9,389,790
Grand Total	**$700,760**	**$28,318,145**	**$339,773**	**$29,358,677**

In #5, we've swapped Territories[Country] from Rows to Columns, and Products[Category] from Columns to Rows. We've also turned off display of grand totals.

5

Total Sales	Territories[Country]					
Products[Category]	Australia	Canada	France	Germany	United Kingdom	United States
Accessories	$138,691	$103,378	$63,407	$62,233	$76,630	$256,422
Bikes	$8,852,050	$1,821,302	$2,553,576	$2,808,514	$3,282,843	$8,999,860
Clothing	$70,260	$53,165	$27,035	$23,565	$32,240	$133,508

In 6-8, Territories[Continent] and Territories[Region] are on Rows. Customers[Gender] is on Report Filters. In 6 and 7, Customers[Gender] Is not filtered, but in 8, it is filtered to "F". In 6-8, [Total Sales] and [Orders] are on Values.

6

Gender	All		
Continent and Region	Total Sales	Orders	
Europe	**$8,930,042**	**7,999**	
France	$2,644,018	2,484	
Germany	$2,894,312	2,484	
United Kingdom	$3,391,712	3,031	
North America	**$11,367,634**	**12,942**	
Canada	$1,977,845	3,375	
Central	$3,001	9	
Northeast	$6,532	10	
Northwest	$3,649,867	4,058	
Southeast	$12,239	17	
Southwest	$5,718,151	5,473	
Pacific	**$9,061,001**	**6,718**	
Australia	$9,061,001	6,718	
Grand Total	**$29,358,677**	**27,659**	

7

Gender	All		
Continent and Region	Total Sales	Orders	
Europe	**$8,930,042**	**7,999**	
France	$2,644,018	2,484	
Germany	$2,894,312	2,484	
United Kingdom	$3,391,712	3,031	
North America	**$11,367,634**	**12,942**	
Canada	$1,977,845	3,375	
Central	$3,001	9	
Northeast	$6,532	10	
Northwest	$3,649,867	4,058	
Southeast	$12,239	17	
Southwest	$5,718,151	5,473	
Pacific	**$9,061,001**	**6,718**	
Australia	$9,061,001	6,718	
Grand Total	**$29,358,677**	**27,659**	

8

Gender	F		
Continent and Region	Total Sales	Orders	
Europe	**$4,426,724**	**3,957**	
France	$1,271,964	1,232	
Germany	$1,539,713	1,245	
United Kingdom	$1,615,046	1,480	
North America	**$5,751,902**	**6,412**	
Canada	$1,011,320	1,621	
Central	$124	3	
Northeast	$3,836	3	
Northwest	$1,843,586	2,043	
Southeast	$11,938	12	
Southwest	$2,881,098	2,730	
Pacific	**$4,634,993**	**3,373**	
Australia	$4,634,993	3,373	
Grand Total	**$14,813,619**	**13,742**	

Note!

In 9, Territories[Continent] is a Slicer. Customers[Gender] is on Rows. [Orders] is on Values.

9

Continent	
Europe	North America
Pacific	NA

Customers[Gender]	Orders
F	6,412
M	6,530
Grand Total	**12,942**

Answers

1) Territories[Country]="France", Products[Category]="Bikes"
2) Territories[Country]="Germany"
3) Products[Category]="Accessories"
4) No Filters
5) Same as #1!
6) Territories[Continent]="North America", Territories[Region]="Northwest"
7) Same as #6!
8) Territories[Continent]="North America", Customers[Gender]="F"
9) Same as #8!

Power Pivot and Power BI:
Commonly-Used DAX Functions and Techniques

powerpivot(pro)
LEADING THE DATA REVOLUTION
T: +1 440.719.9000 | E: simple@powerpivotpro.com | W: www.powerpivotpro.com

CALCULATE() Function

CALCULATE(<measure expression>, <filter1>, <filter2>, ... <filterN>)

<measure expression>:	[MeasureName] SUM(Table[Column]) Any measure name or valid formula for a measure
"Simple" <filter>:	Sales[TransactionType]=1 Products[Color]="Blue" Calendar[Year]>=2009 Sales[TransType]=1 || Sales[TransType]=3
Advanced <filter>:	ALL(...) FILTER(...) DATESBETWEEN(...) Any other function that modifies filter context
Notes:	Raw <filter>'s override (replace) filter context from pivot Raw <filter>'s must be Table[Column] <operator> <fixed value> Multiple <filter>'s arguments get AND'd together

ALL() Function

ALL(<table>) or ALL(Table[Col1], Table[Col2], ...Table[ColN])

Basic usage:	As a <filter> argument to CALCULATE() Removes filters from specified table or column(s) Strips those tables/columns from the pivot's filter context
Advanced Usage:	Technically, ALL() returns a table So it is also useable wherever a <table expr> is required ...such as the first argument to FILTER()

Common Date Calculations

Year to Date: CALCULATE(<measure>, DATESYTD(Calendar[Date])
Qtr or Month to date: *Substitute DATESQTD or DATESMTD for Quarter or Month to date*

Previous Month: CALCULATE(<measure>, DATEADD(Calendar[Date], -1, Month)
Prev Qtr/Year/Day: *Substitute "Quarter" or "Year" or "Day" for "Month" as last argument*

30-day Moving Avg:

```
CALCULATE(<measure>,
        DATESINPERIOD(Calendar[Date],
                MAX(Calendar[Date]), -30, Day
                )
        ) / 30
```

Time Intelligence with Custom Calendar

When Your Biz Calendar is Too Complex for the Built-In Functions

```
=CALCULATE( <measure expr>,
        FILTER(ALL(<Custom Cal Table>), <custom filter>),
        <optional VALUES() to restore filters on some Cal fields>
        )

=CALCULATE( [Sales],
        FILTER(ALL(Cal445), Cal445[Year]=MAX(Cal445[Year])-1)
        )

=CALCULATE( [Sales],
        FILTER(ALL(Cal445), Cal445[Year]=MAX(Cal445[Year])-1),
        VALUES(Cal445[MonthOfYear])
        )
```

More info at http://ppvt.pro/GFITW

SWITCH() Function

Alternative to Nested IF's!

```
=SWITCH(<value to test>,
        <if it matches this value>, <return this value>,
        <if it matches this value>, <return this value>,
        ...more match/return pairs...,
        <if no matches found, return this optional "else" value>
        )
```

Need Training? Advice? Or Help with a Project?
Contact Us:
Simple@PowerPivotPro.com

FILTER() Function

FILTER(<table expression>, <single rich filter>)

<table expression>:	The Name of a Table, or any of the below... VALUES(Table[Column]) - unique values of Table[Column] for current pivot cell ALL(Table) or ALL(Table[Column]) Any expression that returns a table, such as DATESYTD() Even another FILTER() can be used here for instance
<rich filter>:	Table[Column1] >= Table[Column2] Table[Column] <= [Measure] [Measure1] <> [Measure2] <true/false expr1> && <true/false expr2> Any expression that evaluates to true/false
Notes:	Commonly used as a <filter> argument to CALCULATE() Useful when a richer filter test is required than "simple" filters can do Never use FILTER when a "simple" CALCULATE() <filter> will work Slow and eats memory when used on large tables Use against small (Lookup) tables for better performance Advanced usage: use anywhere a <table expr> is required

VALUES() Function

VALUES(Table[Column])

1-column table, unique: Produces a temporary, single-column table during formula evaluation
(Most common usage) That table contains ONLY the UNIQUE values of Table[Column].

EX: CALCULATE(<measure>, FILTER(VALUES(Customers[PostalCode]), ...))

That allows us to iterate as if we had a PostalCode table, even though we don't! And then the formula above calculates <measure> only for those Postal Codes that "survive" the <filter expr> test inside the FILTER function. And therefore only includes the customers IN those postal codes!

Restoring a filter: CALCULATE([M], ALL(Table), VALUES(Table[Col1]))
(2nd most common usage) ...is roughly equiv to CALCULATE([M], ALLEXCEPT(Table, Table[Col1]))

Note: VALUES(Table[Column]) returns filtered list even if Table[Column] isn't on pivot!

Forcing Grand/Sub Totals to Be the Sum of Their "Parts"

=SUMX(VALUES(Table[Column], <original measure>)

(Where the values of Table[Column] are the "small pieces" that need to be calculated individually and then added up.)

Calc Columns That Reference "Previous" Row(s)

```
=CALCULATE( [Measure],
        FILTER(<table>, Table[Col]=EARLIER(Table[Col])-1)
        )

=CALCULATE( AVERAGE(Tests[Score]),
        FILTER(Tests, Tests[ID]=EARLIER(Tests[ID])-1)
        )
```

Suppressing Subtotals/Grand Totals

=IF(HASONEVALUE(Table[Column]), <measure expr for non-totals>, BLANK())

RANKX() Function

RANKX(<table expr>, <arithmetic expression>, <optional alternate arithmetic expression>, <optional sort order flag>, <optional tie-handling flag>)

Simplest Usage:	RANKX(ALL(Table[Column]), <numerical expr>) EX: RANKX(ALL(Products[Name]), [TotalSales])
Ascending Rank Order:	EX: RANKX(ALL(Products[Name]), [TotalSales],,1)
"Dense" Tie Handling:	EX: RANKX(ALL(Products[Name]), [TotalSales],,Dense)

DIVIDE Function

Returns BLANK() Cells on "Div by Zero", No IF() or IFERROR() required!

=DIVIDE(<numerator>, <denominator>, <optional val to return when div by zero>)

4

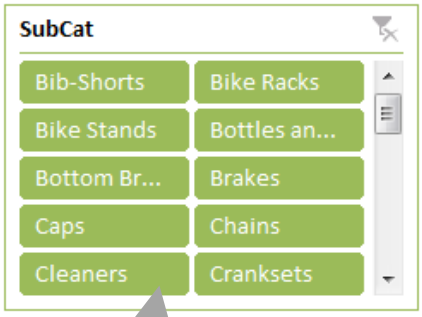
Data Tables

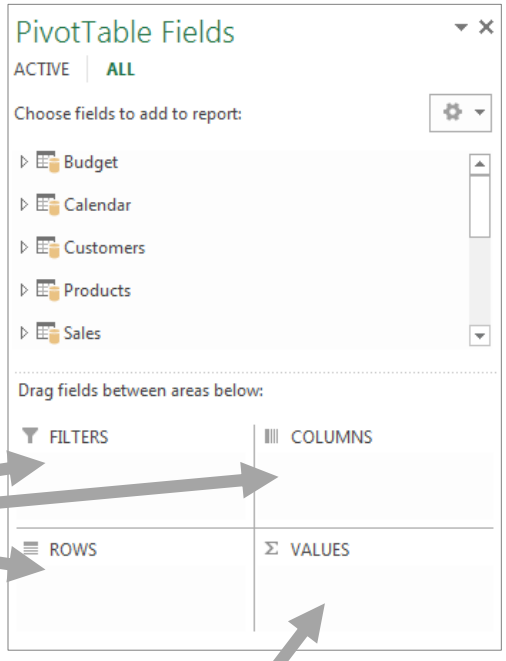
- Contain the numbers
- EX: Sales, Budget, Inventory, etc.
- Sometimes called “fact” tables
- Measures/calc fields tend to come from data tables
- In diagram view, the “dot” or “*” end of a relationship.

Lookup Tables

- Tend to have fewer rows than data tables
- EX: Calendar, Customers, Stores, Products, etc.
- Sometimes called “dimension,” “reference,” or “master” tables
- Row, Column, Report Filter, and Slicer fields
- In diagram view, the “arrow” or “1” end of a relationship.

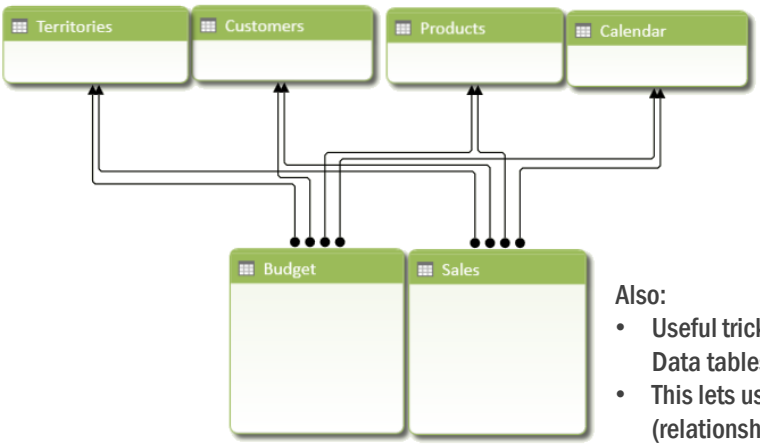
Under “Ideal” Conditions, Data and Lookup Tables are Used Like THIS in Pivots:





Every field used in these places comes from Lookup tables.
(Note that these are the places that contribute to filter context during measure calculation!)

And every field in the Values Area Comes from Data tables.
(Although we DO occasionally write measures against Lookup tables, such as days elapsed, products offered, etc.

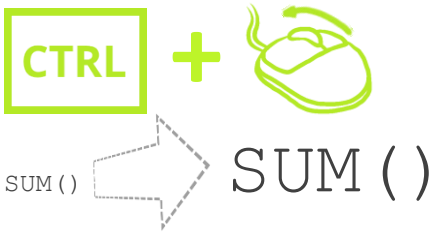


Note:
- Data tables are “spliced together” ONLY by sharing one or more Lookup tables
- Now follow the field list guidelines above and you can compare Budget v Actuals (for instance) in a single pivot!
- Data tables are never related directly to each other!

Also:
- Useful trick: Arrange Lookup tables “up high” on the diagram and Data tables “down low.”
- This lets us envision filters flowing “downhill” across relationships (relationships are “1-way”)

Make the formula font bigger!

(Hold CTRL key down and roll mouse wheel forward)



Insert New Lines in Formulas:

```
=CALCULATE ( [Total],
             Table[Column]=6
           )
```

When writing measures/calc fields:

1) Always INCLUDE table names on column references.

`Table[Column]` **YES** `[Column]` **NO**

2) Always EXCLUDE table names when referencing other measures.

`[Measure]` **YES** `Table[Measure]` **NO**

By following this convention, you will ALWAYS immediately know the difference between a measure and a column reference, on sight, and that's a BIG win for readability and debugging.

(But when writing a calc column, it is acceptable to omit the table name from a column reference, since you rarely reference measures in calc columns.)

NEVER write the same formula twice!

For example, you should define basic measures like these, even for "simple" calculations like SUM:

```
[Total Sales]:= SUM(Table[Amount])
[Total Cost]:= SUM(Table[Cost])
```

And then references those measures whenever you are tempted to rewrite the SUM in another measure:

YES

```
[Total Margin]:=
[Total Sales] - [Total Cost]
```

YES

```
[Year to Date Sales]:=
CALCULATE([Total Sales], DATESYTD(Dates[Date])
```

NO

```
[Total Margin]:=
SUM(…) - SUM(…)
```

NO

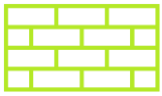
```
[Year to Date Sales]:=
CALCULATE(SUM(…), DATESYTD(Dates[Date])
```

Measures (Calculated Fields) Are:

1. Used in cases when a single row can't give you the answer (typically aggregates like sum, etc.)
2. Only "legal" to be used in the Values area of a pivot
3. Never pre-calculated
4. ALWAYS re-calculated in response to pivot changes – slicer or filter change, drill down, etc.
5. Return different answers in different pivots
6. Not a source of file size increase
7. "Portable Formulas!!"

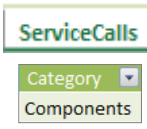
Calculated Columns Are:

1. Used to "stamp" numbers or properties on each row of a table
2. "Legal" on row/column/filter/slicer of pivots
3. Useful for grouping and filtering, for instance
4. Also usable as inputs to measures
5. Pre-calculated and stored – making the file bigger
6. NEVER re-calculated in response to pivot changes
7. Only re-calculated on data source refresh or on change to "precedent" (upstream) columns

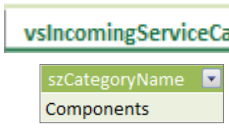


Rename after import!

Overly-long and/or cryptically-named tables and columns make your formulas harder to read AND write, and since Power Pivot 2010 and 2013 don't fix up formulas on rename, it pays to rename immediately after import.



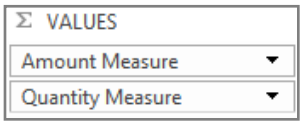
YES



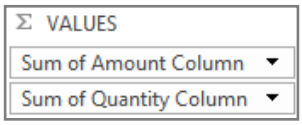
NO

NEVER Use Columns in Pivot Values Area (Write the Measure/Calc Field Instead)

YES:



NO:



(See re-use & maintenance benefits in DAX Formulas for Power Pivot, Ch6)

6

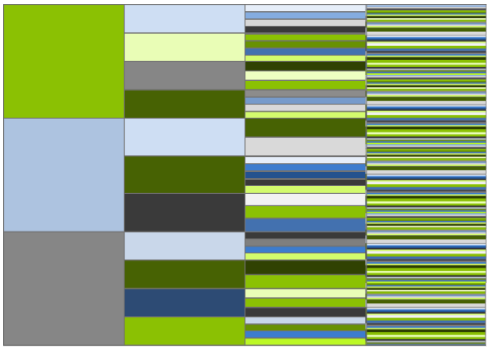
Reducing File Size

PowerPivot, PowerBI Designer, and SSAS Tabular all store and compresses data in a "column stripe" format, as pictured here.

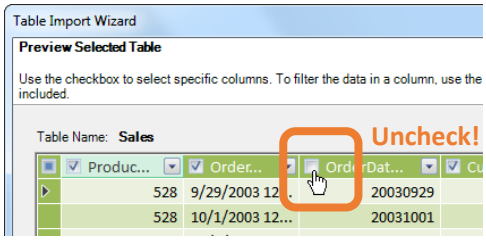
Each column is less compressed than the one before * it.
(* The compression order of the columns is auto-decided by the engine at import time, and not something we can see or control.)

This column-oriented storage is VERY unlike traditional files, databases, and compression engines.

Sometimes, a single column is "responsible" for a large fraction of the file's size (like the 125 MB pictured here.)

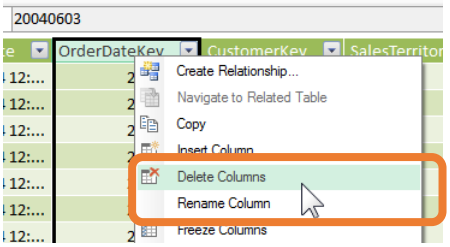


What does that MEAN to us? We want fewer columns!

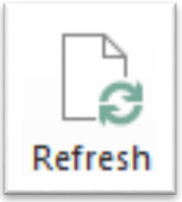


Uncheck unneeded columns during import (or by going to Table Properties later).

OR



1) Delete unneeded columns, and then...



2) Refresh the table.

1. **Only import the columns that you truly need! (you can always go grab more columns later if needed).**
2. **For your Data tables, 5-10 columns is a good goal (Lookup tables can have many more than that).**
3. **If you delete a column after import, refresh that table – the engine re-optimizes the storage during refresh.**

Calculated Column Notes

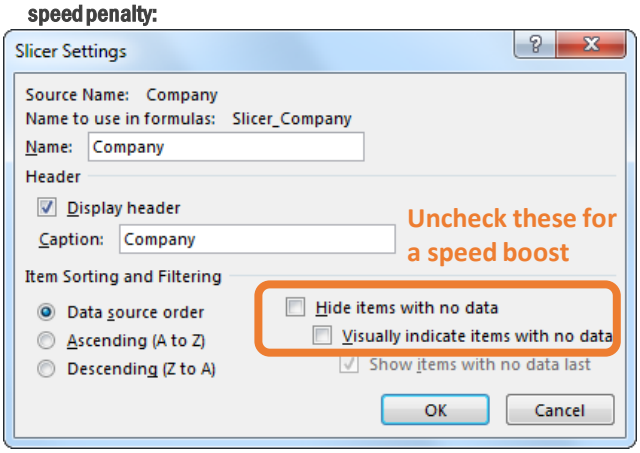
1. **Calc columns bloat the file more than columns imported from a data source.**
2. **So consider implementing the calc column in the database (or use Power Query), then import it.**
3. **Unlike calc columns, measures do NOT add file size!**
4. **So in "simple arithmetic" cases like [Profit Margin], it's best to just subtract one measure from another ([Sales] – [Cost]), and avoid adding a calc column to perform the subtraction (which you'd then SUM to create your measure).**

Words of Wisdom

1. **If your file size is not a problem, don't worry about ANYTHING on this page. These tips are just for when you DO have a problem ☺**
2. **The smaller the table is in terms of row count, the less these tips and tricks matter. A few extra columns in a 10k-row table are no big deal, but ONE extra column in a million-row table sometimes IS.**
3. **So focus on Data tables. Lookup tables = less crucial.**
4. **Large files also eat more RAM. If your server is strained or 32-bit Excel breaks down, reduce file size.**

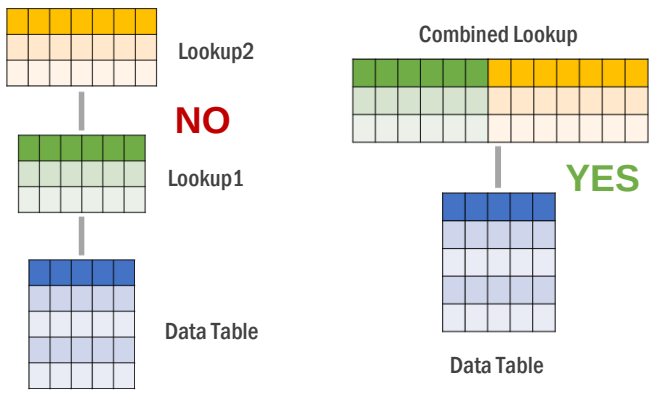
Slicers Can Slow Things Down!

1. **A single slicer can double the update time of a pivot!**
2. **Consider unchecking these checkboxes on some slicers to remove that speed penalty:**



Avoid "Multi-Hop" Lookups (if Possible)

Combine "chained" lookup tables into one table:



powerpivot(pro)

Separate Lookup Tables Offer BIG File Size Savings

OrderDate	CustomerKey	ExtendedAmount	ProductKey	ProductName	StandardCost	Color	ModelName
7/1/2001	14501	699.0982	336	Road-650 Black, 62	413.1463	Black	Road-650
7/1/2001	25863	3399.99	346	Mountain-100 Silver, 44	1912.1544	Silver	Mountain-100
7/1/2001	28389	3399.99	346	Mountain-100 Silver, 44	1912.1544	Silver	Mountain-100
7/1/2001	21768	3578.27	310	Road-150 Red, 62	2171.2942	Red	Road-150
7/1/2001	11003	3399.99	346	Mountain-100 Silver, 44	1912.1544	Silver	Mountain-100
7/2/2001	27645	3578.27	311	Road-150 Red, 44	2171.2942	Red	Road-150
7/2/2001	11011	3399.99	344	Mountain-100 Silver, 38	1912.1544	Silver	Mountain-100
7/2/2001	11005	3374.99	351	Mountain-100 Black, 48	1898.0944	Black	Mountain-100
7/2/2001	16624	3578.27	310	Road-150 Red, 62	2171.2942	Red	Road-150
7/3/2001	27621	3578.27	312	Road-150 Red, 48	2171.2942	Red	Road-150
7/3/2001	27616	3578.27	312	Road-150 Red, 48	2171.2942	Red	Road-150
7/3/2001	20042	699.0982	330	Road-650 Red, 52	413.1463	Red	Road-650
7/3/2001	16517	3578.27	314	Road-150 Red, 56	2171.2942	Red	Road-150
7/3/2001	16351	3578.27	313	Road-150 Red, 52	2171.2942	Red	Road-150

NO

The table pictured above combines Data table columns (OrderDate, CustomerKey, ExtendedAmount, and ProductKey) with columns that should be "outsourced" to a Lookup table (ProductName, StandardCost, Color, and ModelName can all be "looked up" from the ProductKey).

Instead, split the Lookup-specific columns out into a separate Lookup table, and remove duplicate rows (in that Lookup table) so that we have just one row per unique ProductKey.

OrderDate	CustomerKey	ExtendedAmount	ProductKey
7/1/2001	14501	699.0982	336
7/1/2001	25863	3399.99	346
7/1/2001	28389	3399.99	346
7/1/2001	21768	3578.27	310
7/1/2001	11003	3399.99	346
7/2/2001	27645	3578.27	311
7/2/2001	11011	3399.99	344
7/2/2001	11005	3374.99	351
7/2/2001	16624	3578.27	310
7/3/2001	27621	3578.27	312
7/3/2001	27616	3578.27	312
7/3/2001	20042	699.0982	330
7/3/2001	16517	3578.27	314
7/3/2001	16351	3578.27	313
7/4/2001	27606	3578.27	314
7/4/2001	13513	3578.27	311
7/5/2001	27601	3578.27	310

RELATIONSHIP

ProductKey	ProductName	StandardCost	Color	ModelName	Category	SubCategory
310	Road-150 Red, 62	2171.2942	Red	Road-150	Road Bikes	Bikes
311	Road-150 Red, 44	2171.2942	Red	Road-150	Road Bikes	Bikes
312	Road-150 Red, 48	2171.2942	Red	Road-150	Road Bikes	Bikes
313	Road-150 Red, 52	2171.2942	Red	Road-150	Road Bikes	Bikes
314	Road-150 Red, 56	2171.2942	Red	Road-150	Road Bikes	Bikes
330	Road-650 Red, 52	413.1463	Red	Road-650	Road Bikes	Bikes
336	Road-650 Black, 62	413.1463	Black	Road-650	Road Bikes	Bikes
344	Mountain-100 Silver, 38	1912.1544	Silver	Mountain-100	Mountain Bikes	Bikes
346	Mountain-100 Silver, 44	1912.1544	Silver	Mountain-100	Mountain Bikes	Bikes

YES

Duplicate removal makes a relationship possible with the Data table, AND makes the Lookup table small in terms of row count.

(Duplicate removal is performed in the database, or using Power Query– see Power Pivot Alchemy, chapter5 for an example).

Our "big" table now has significantly fewer columns. On net, our file is potentially now MUCH smaller– because our largest table (Data table) has shed multiple columns. The small Lookup table is not significant, even if it contains 50+ columns.

YES

"Unpivot" ALSO Offers Big File Size Savings

Region	1/1/2015	1/2/2015	1/3/2015	1/4/2015	1/5/2015	1/6/2015	1/7/2015	1/8/2015
North	$2,106	$4,712	$1,996	$4,147	$5,044	$1,869	$3,004	$8,032
South	$2,470	$1,375	$6,133	$7,040	$1,951	$1,141	$7,871	$1,850
East	$6,283	$3,591	$7,646	$2,417	$8,487	$6,973	$3,520	$3,540
West	$8,383	$2,925	$8,109	$7,996	$6,916	$4,401	$8,315	$5,995

NO

Region	Normal Sales	Promotional Sales	Refunds	Bulk Sales	Cost of Goods
North	$2,106	$4,712	$1,996	$5,044	$4,147
South	$2,470	$1,375	$1,850	$1,951	$7,040
East	$6,283	$3,591	$1,951	$8,487	$2,417
West	$8,383	$2,925	$2,106	$6,916	$7,996

NO

This "unpivot" transformation results in increased rows but fewer columns. Counterintuitively this can yield VERY significant filesize reduction. (See Power Pivot Alchemy, Ch 5, for an example of performing this transformation with Power Query).

In the case of dates or months, this also removes the need for tedious formula repetition, AND enables time intelligence calcs.

Region	Date	Value
North	1/1/2015	$2,106
South	1/1/2015	$2,470
East	1/1/2015	$6,283
West	1/1/2015	$8,383
North	1/2/2015	$4,712
South	1/2/2015	$1,375
East	1/2/2015	$3,591
West	1/2/2015	$2,925
North	1/3/2015	$1,996
South	1/3/2015	$6,133
East	1/3/2015	$7,646
West	1/3/2015	$8,109
North	1/4/2015	$4,147
South	1/4/2015	$7,040
East	1/4/2015	$2,417
West	1/4/2015	$7,996
North	1/5/2015	$5,044
South	1/5/2015	$1,951
East	1/5/2015	$8,487
West	1/5/2015	$6,916
North	1/6/2015	$1,869
South	1/6/2015	$1,141
East	1/6/2015	$6,973

YES

Region	Amount Type	Amount
North	Normal Sales	$2,106
North	Promotional Sales	$4,712
North	Refunds	$1,996
North	Bulk Sales	$5,044
North	Cost of Goods	$4,147
South	Normal Sales	$2,470
South	Promotional Sales	$1,375
South	Refunds	$1,850
South	Bulk Sales	$1,951
South	Cost of Goods	$7,040
East	Normal Sales	$6,283
East	Promotional Sales	$3,591
East	Refunds	$1,951
East	Bulk Sales	$8,487
East	Cost of Goods	$2,417
West	Normal Sales	$8,383
West	Promotional Sales	$2,925
West	Refunds	$2,106
West	Bulk Sales	$6,916
West	Cost of Goods	$7,996

YES

In this case you will need to use CALCULATE to write your "base" measures. EX:

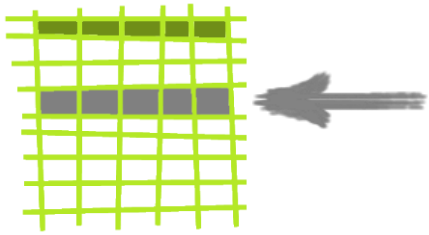
```
CALCULATE(SUM(Table[Amount]),
Table[Amount Type]="Refunds")
```

What Makes a Valid Calendar/Dates Table?

Date	MonthShort	DayOfWeekNum	DayOfWeek	DayOfMonthNum
6/1/2015 12:00:00 AM	Jun	2	Mon	1
6/2/2015 12:00:00 AM	Jun	3	Tue	2
6/3/2015 12:00:00 AM	Jun	4	Wed	3
6/4/2015 12:00:00 AM	Jun	5	Thu	4
6/5/2015 12:00:00 AM	Jun	6	Fri	5
6/6/2015 12:00:00 AM	Jun	7	Sat	6
6/7/2015 12:00:00 AM	Jun	1	Sun	7
6/8/2015 12:00:00 AM	Jun	2	Mon	8

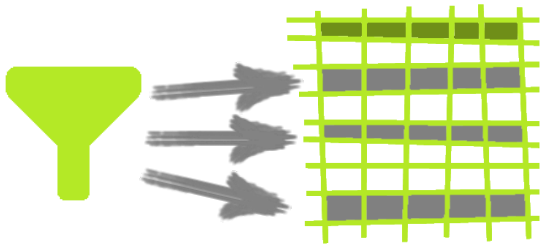
1. Must contain a column of actual Date data type, not just text or a number that looks like a date.
2. That Date column must NOT contain times – 12:00 AM is "zero time" and is EXACTLY what you want to see.
3. There CANNOT be "gaps" in the Date column. No skipped dates, even if your business isn't open on those days.
4. Must be "Marked as Date Table" via button on the Power Pivot window's ribbon (not applicable in Power BI Desktop).
5. May contain as many other columns as desired. Go nuts ☺
6. Should not contain dates that "precede" your actual data – needless rows DO impact performance.
7. You MUST then use this as a proper Lookup table – don't use dates from your Data tables on Rows/Columns/Etc.!

(Slightly) Advanced Concept: Row Context



- You HAVE a Row Context in a Calculated Column.
- But you do NOT have a Row Context in a Measure (Calculated Field).
- A calc column is calculated on a row-by-row basis, so there's one row "in play" for each evaluation of the formula.
- So =[Column] resolves to a single value (the value from "this row"), w/out error.
- "The current row" is called Row Context.
- You may only reference a "naked' column (naked = no aggregation fxn), and have it resolve to a single number, date, or text value when you have a Row Context.

(Slightly) Advanced Concept: Filter Context



- You HAVE a Filter Context in a Measure / Calc Field.
- But you do NOT have a Filter Context in a Calc Column.
- Each cell in a Pivot's values area is calculated based on the filters (coordinates) specified for that cell.
- Those filters resolve to a set of multiple rows in the underlying data tables, rather than a single row.
- =[Column] is therefore illegal as a formula, or as part of a formula where a single value is needed.
- So this is why aggregation functions are required in measures – to "collapse" multiple values into one.

Exception: Filter Context in Calc Columns

- Aggregation functions like SUM *always* reference the Filter Context
- Since there is no Filter Context in a calc column, =SUM([Column]) will return the sum of the ENTIRE column – you get the same answer all the way down.
- But you can tell the DAX engine to use a Row Context as if it were ALSO a Filter Context, by wrapping the aggregation function in a CALCULATE.
- EX: =CALCULATE(SUM[Column])) "respects" the context of each row, AND also relationships
- So in a Lookup table, you can use CALCULATE(SUM(Data[Col])) to get the sum of all "matching" rows from the related Data table.
- Furthermore, the DAX engine always "adds" a CALCULATE "wrapper" whenever you reference a Measure. So =[MySumMeasure] ALSO respects Row Context and Relationships.

Exception: Row Context in Measures

- Certain functions step through tables one row at a time, even when used within a Measure.
- Those "iterator" functions are said to create Row Contexts during their operation.
- Ex: FILTER(*table*, *expr*) and SUMX(*table*, *expr*)
- In both examples, you CAN reference a column, within the *expr* argument, and use that column as a single value, within the *expr* argument.
- Note however that the column MUST "come from" the table specified in the *table* argument.
- Also note that this Row Context only exists within the evaluation of the iterator function itself (FILTER, SUMX, etc.) and does NOT exist elsewhere in the measure formula.