

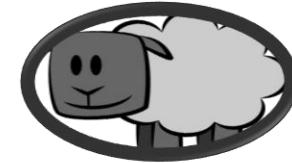
Power up your Fabric Development with DAX Studio and Tabular Editor



SQL Saturday San Diego
September 2024

Jason Romans

Senior BI Engineer
Builder of Models



The Dax Shepherd



Lives in Nashville, Tennessee, United States



Started as SQL Server DBA



Transitioned to the Microsoft BI Stack



**Work on everything from SQL Server Integration Services,
SQL Server Database, Analysis Services, and Power BI**



Simple Talk Author at Redgate



Favorite Data Model

Shoulders of Giants



This Photo by Unknown Author is licensed under [CC BY-SA-NC](#)



This Photo by Unknown Author is licensed under [CC BY-ND](#)

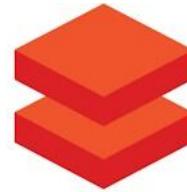
Our Sponsors

UC San Diego

EXTENDED STUDIES

 Microsoft

vmware[®]
by Broadcom



databricks[®]

 COZYROC

datAvail

O'REILLY[®]

 JETBRAINS

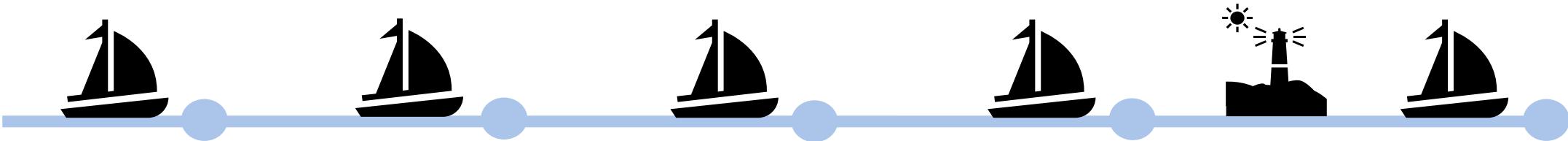
SQL
skills
immerse yourself in sql server

Community Support

Name	Type	Frequency	Join
San Diego SQL Server User Group	Hybrid	Every 3 rd Thursday monthly @ 6.30PM PDT	
LA Data Platform User Group	Virtual	Every 3 rd Wednesday monthly @ 7PM PDT	
San Francisco Microsoft Fabric Meetup	Virtual	Every 2 nd Wednesday monthly @ 7PM PDT	
Sacramento SQL Server User Group	In-Person	Every 1 st Wednesday monthly @ 6PM	

[Find more Azure Data Tech Groups on Meetup](#)

Our Voyage



1. Intro
2. Performance Analyzer
3. DAX Studio
4. Tabular Editor
5. Conclusion

Our Voyage



- 1. Intro**
- 2. Performance Analyzer
- 3. DAX Studio
- 4. Tabular Editor
- 5. Conclusion

Power BI is Part of Fabric

- If you have a Fabric capacity
 - Enhances what you can do with Power BI
 - OneLake
 - Warehouse
 - Lakehouse
 - What we cover applies to Power BI Pro, Power BI Premium, and Fabric

Exam DP-600: Implementing Analytics Solutions Using Microsoft Fabric

Design and build semantic models

- Identify use cases for DAX Studio and Tabular Editor 2

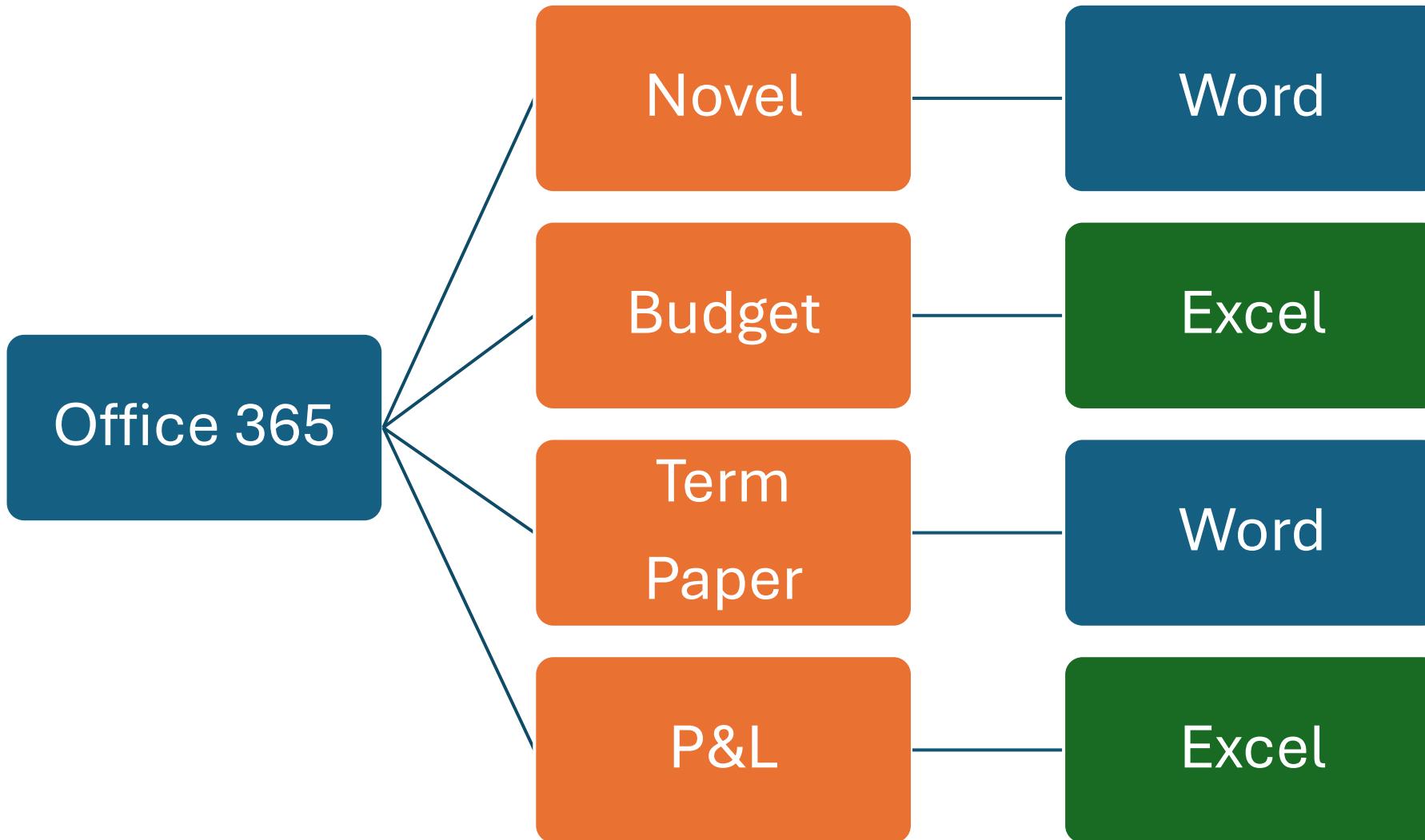
Optimize enterprise-scale semantic models

- Improve DAX performance by using DAX Studio
- Optimize a semantic model by using Tabular Editor 2

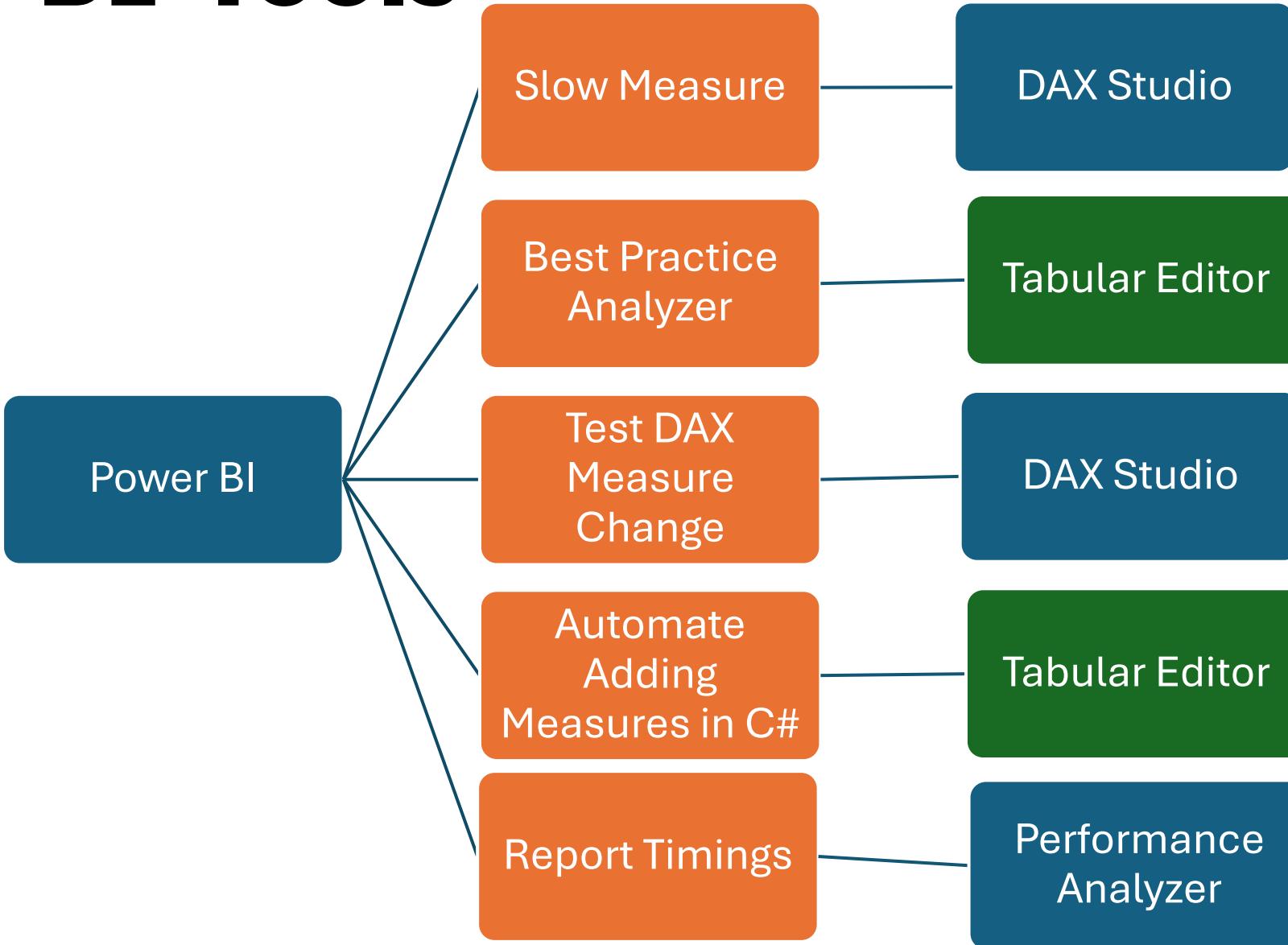
Why is my report slow?

- Motivating Factor
- This can have a lasting impact
 - Power BI Developer Skills
 - Career Development

Office 365



Power BI Tools



Our Voyage

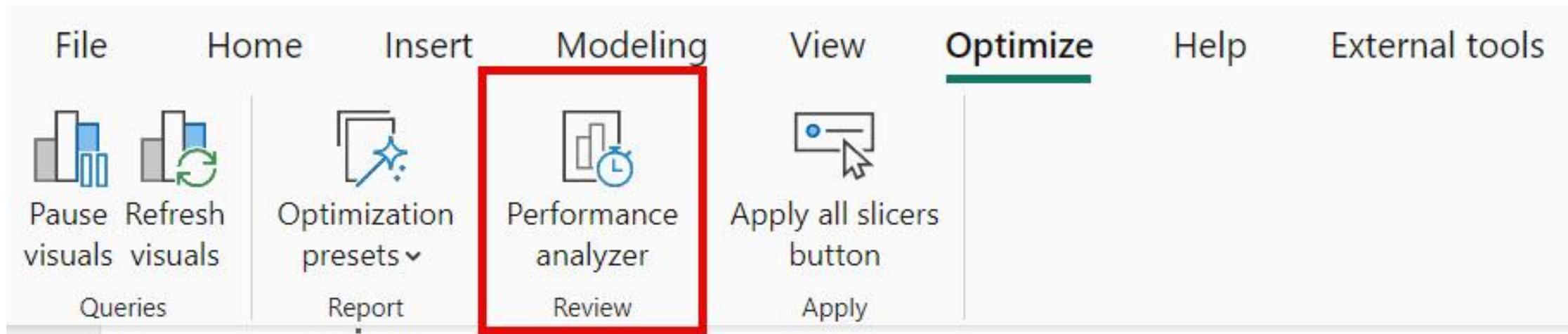


- 1. Intro
- 2. Performance Analyzer
- 3. DAX Studio
- 4. Tabular Editor
- 5. Conclusion

Performance Analyzer

- Not an External Tool
- Built into Power BI Desktop

Performance Analyzer



Performance Analyzer

Start Recording

Filters »

Search

Filters on this page ...

Add data fields here

Filters on all pages ...

Year
is 2020

Add data fields here

Performance analyzer »

Start recording Refresh visuals Stop

Start monitoring your report to see details about the time taken by each visual to query for its data and render the result.

Visualizations »

Build visual



...

Data »

Search

>  Sales

>  Customer

>  Date

>  Product

>  Store

Performance Analyzer Recording

Performance analyzer [»](#)

[Start recording](#) [Refresh visuals](#) [Stop](#)

[Clear](#) [Export](#)

Name	Duration (ms) ↓
⌚ Recording started (5/1/2...	-
<i>Interact with your report t...</i>	-

Performance Analyzer

Refresh Visuals

Performance analyzer >

(Start recording (Refresh visuals) (Stop)

(Clear) (Export)

Name	Duration (ms)
Recording started (5/1/2... Interact with your report t...	-

Individual Refresh Button

Note: Values for Other will be different

Date	Total Quantity				...
Wednesday, January 01, 2020	7,759				
Thursday, January 02, 2020	8,256				
Friday, January 03, 2020	5,482				
Saturday, January 04, 2020	8,608				
Sunday, January 05, 2020	1,144				
Monday, January 06, 2020	3,823				
Tuesday, January 07, 2020	4,414				

List of Visuals

Performance analyzer »

▶ Start recording ⟳ Refresh visuals □ Stop

⟲ Clear ⤒ Export

Name	Duration (ms)
⟳ Refreshed visual	-
+ Card	65
+ Card	66
+ Table	102

Expanded View

Performance analyzer >>

(Start recording (Refresh visuals @ Stop

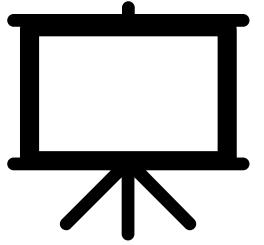
Clear Export

Name	Duration (ms)
(Refreshed visual	-
Card	65
DAX query	3
Visual display	3
Other	59
Copy query	
DAX Run in DAX Query View	
Card	66
DAX query	3
Visual display	4
Other	59
Copy query	
DAX Run in DAX Query View	
Table	102
DAX query	29
Visual display	31
Other	42
Copy query	
DAX Run in DAX Query View	

Detail on Card

 Refreshed visual	-
 Card	65
DAX query	3
Visual display	3
Other	59
 Copy query	
 Run in DAX Query View	

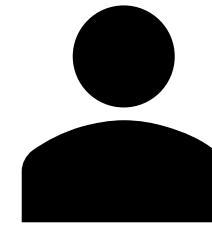
3 Numbers



Visual Display



Other



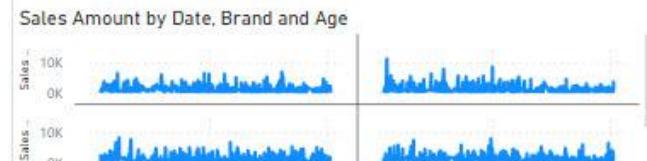
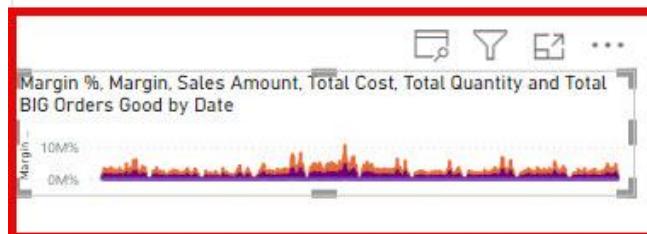
DAX Query

Visual Display

⟳ Refreshed visual	-
⊖ Card	65
DAX query	3
Visual display	3
Other	59
 Copy query	
 Run in DAX Query View	

- Visual Display
 - Time spent on producing the visual
- Restaurant
 - Plating

Slow Visual



Filters

Performance analyzer

Start recording

Refresh visuals Stop

Clear Export

Name	Duration (ms)
Refreshed visual	-
Margin and Margin % by Date	725
Total Cost by Date	724
Sales Amount by Date	152
Margin %, Margin, Sales Amount, Total Co... DAX query	723
Visual display	10
Other	526
Copy query	187
Run in DAX Query View	
Sales Amount by Date, Brand and Age	927
Total Cost and Total Quantity by Date	722

Visual Display - Solutions

- Reduce the complexity of visual
 - Granularity of visual
- Use a Background Image
- More in Resources

Other

⌚ Refreshed visual	-
✉ Card	65
DAX query	3
Visual display	3
Other	59
Copy query	
Run in DAX Query View	

- Other
 - Time waiting until DAX Query can be executed
- Restaurant
 - Waiting to order

Other - Solutions

- Reduce Visualizations on the Page
 - Scrollbar?
- Evaluate where the bottleneck is
 - May be a combination of other factors

DAX Query



- DAX Query
 - Time it takes to execute the DAX query
- Restaurant
 - Time to make the food

Slow DAX Query

Performance analyzer [»](#)

[Start recording](#) [Refresh visuals](#) [Stop](#)

[Clear](#) [Export](#)

Name	Duration (ms) ↓
⟳ Refreshed visual	-
+ Card	52
+ Card	52
⊖ Table	28059
DAX query	27992
Visual display	30
Other	37
Copy query	
Run in DAX Query View	

DAX Query - Solutions



- Investigate query
 - Where is most of the query time spent
 - Can the DAX code be optimized?
 - Rewrite the DAX code

Run in DAX Query View

Table	102
DAX query	29
Visual display	31
Other	42
 Copy query	
 Run in DAX Query View	

DAX queries will be saved to your model. They won't be visible when published in the Power BI service. [Learn more](#)

Run

DAX

```
1 DEFINE
2     VAR __DS0Core =
3         SUMMARIZECOLUMNS(
4             ROLLUPADDISSUBTOTAL(ROLLUPGROUP('Product'[Category], 'Product'[Brand]), "IsGrandTotalRowTotal"),
5             "Margin", 'Sales'[Margin]
6         )
7
8     VAR __DS0PrimaryWindowed =
9         TOPN(502, __DS0Core, [IsGrandTotalRowTotal], 0, 'Product'[Category], 1, 'Product'[Brand], 1)
10
11    EVALUATE
12        __DS0PrimaryWindowed
13
14    ORDER BY
15        [IsGrandTotalRowTotal] DESC, 'Product'[Category], 'Product'[Brand]
```

Results | Result 1 of 1 ▾ | Copy

	Product[Category]	Product[Brand]	[IsGrandTotalRowTotal]	[Margin]
1			True	1267049808.67
2	Audio	Contoso	False	5547636.55
3	Audio	Northwind Traders	False	5276681.88
4	Audio	Wide World Importers	False	21868862.18
5	Cameras and camcorders	A. Datum	False	28002980.67
6	Cameras and camcorders	Contoso	False	19656707.45
7	Cameras and camcorders	Fabrikam	False	71111794.44
8	Cell phones	Contoso	False	22490172.07
9	Cell phones	The Phone Company	False	174204193.03
10	Computers	Adventure Works	False	189547913.31

Query 1 +

Success (27.3 ms) | Query 1 of 1 | Result 1 of 1 | 4 columns, 30 rows

Query from Visual

Query View

- Queries are saved with the model
- Share or preserve slow query
- First exposure to DAX Queries
- Has many uses - i.e. validations
- Not good for diagnosing slow queries

DAX Query View for Web

- Just Announced
<https://powerbi.microsoft.com/en-us/blog/deep-dive-into-dax-query-view-for-web/>
- Use against models in Workspace

Power BI Demo

Performance Analyzer

Performance Analyzer

- Informed Decisions
- Don't underestimate this tool
- You can isolate where the issue is
- Why spend all day optimizing DAX if it isn't the issue

What about that slow DAX Query?

Performance analyzer

»

Start recording Refresh visuals Stop

Clear Export

Name	Duration (ms)
Refreshed visual	-
Table	61668
DAX query	61617
Visual display	23
Other	28

Copy query Run in DAX Query View



[This Photo](#) by Unknown Author is licensed under [CC BY](#)

DAX

External Tools



Installation

- Full Install
- Power BI Desktop
 - External Tools Tab

External Tools Tab

A screenshot of a software application's ribbon interface, specifically focusing on the "External tools" tab. The ribbon tabs include File, Home, Insert, Modeling, View, Optimize, Help, External tools (which is underlined in green), Format, and Data / Drill. Below the tabs, there are six icons with corresponding labels: ALM Toolkit (green cube), Bravo (red cube), Model Documenter (yellow document with 'B' icon), DAX Studio (blue cube with gear and magnifying glass), Tabular Editor (black cube with wrench), and Tabular Editor (green cube with white 'T'). At the bottom center of the ribbon, the text "External tools" is displayed.

ALM Toolkit	Bravo	Model Documenter	DAX Studio	Tabular Editor	Tabular Editor

External tools

Our Voyage



1. Intro
2. Performance Analyzer
- 3. DAX Studio**
4. Tabular Editor
5. Conclusion

What if the DAX Query is Slow?



DAX Studio

- Download
 - <https://daxstudio.org/>
- Author Measures and Queries
- Connect to Semantic Models
- Performance Tuning
 - Details on how the query is processed

Slow DAX Query

Performance analyzer [»](#)

[Start recording](#) [Refresh visuals](#) [Stop](#)

[Clear](#) [Export](#)

Name	Duration (ms)
Refreshed visual	-
Table	61668
DAX query	61617
Visual display	23
Other	28
Copy query	
Run in DAX Query View	

Copy query

Performance analyzer >>

(Start recording (Refresh visuals) (@ Stop)

Clear Export

Name	Duration (ms)
(Refreshed visual	-
Table	61668
DAX query	61617
Visual display	23
Other	28
Copy query	
Run in DAX Query View	

Copied

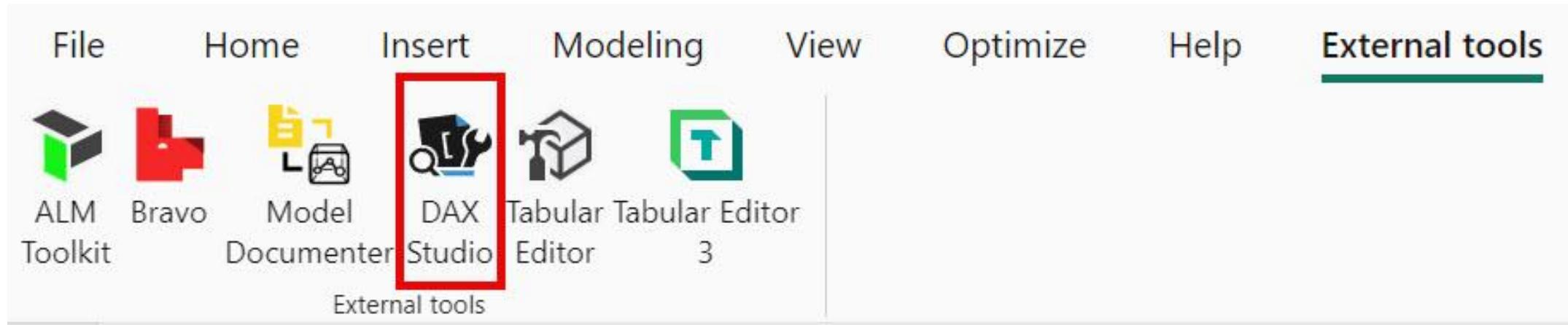
Performance analyzer »

(+) Start recording ↻ Refresh visuals @ Stop

◊ Clear Export

Name	Duration (ms)
⟳ Refreshed visual	-
⊖ Table	61668
DAX query	61617
Visual display	23
Other	29
✓ Copied	
DAX Run in DAX Query View	

Power BI - Open DAX Studio



Paste in DAX Studio

DAX Studio - 3.0.11

File Home Advanced Help

Run Cancel Query Builder Clear Cache Clear on Run Results Output Edit Format Query A To Upper Comment Find Power BI

Cache

Copy Undo Redo To Lower Uncomment Replace

Format Comma Debug Commas Merge XML

Find All Queries Power BI

Load Perf Data Traces

Query Plan Server Timings Connection

Metadata Functions DMV

Contoso1M_Docker_BPATEst Model

Search

Customer Date Product Sales Store

```
1 DEFINE VAR __DSOCore =
2     SUMMARIZECOLUMNS(
3         ROLLUPADDISSUBTOTAL(ROLLUPGROUP('Product'[Category], 'Product'[Brand]), "IsGrandTotalRowTotal"),
4         "Margin", 'Sales'[Margin]
5     )
6
7     VAR __DSOPrimarywindowed =
8         TOPN(502, __DSOCore, [IsGrandTotalRowTotal], 0, 'Product'[Category], 1, 'Product'[Brand], 1)
9
10    EVALUATE
11        __DSOPrimarywindowed
12
13    ORDER BY
14        [IsGrandTotalRowTotal] DESC, 'Product'[Category], 'Product'[Brand]
```

Log Results History

	Start	Duration ⓘ	Message
ⓘ	16:34:01		Establishing Connection
ⓘ	16:34:01		Connected

Let's Run It

DAX Studio - 3.0.11

The screenshot shows the DAX Studio interface version 3.0.11. The toolbar at the top includes icons for File, Home, Advanced, Help, Run (highlighted with a red box), Cancel, Query Builder, Clear Cache, Clear on Run, Results, Cut, Undo, Copy, Redo, Paste, Format Query, To Upper, To Lower, Comment, Uncomment, Debug Commas, Merge XML, Find, Replace, Load Perf Data, All Queries, Query Plan, Server Timings, Connect, and Refresh Metadata. Below the toolbar is a sidebar with tabs for Metadata, Functions, and DMV. The Metadata tab is selected and shows a dropdown menu with 'Contoso1M_Docker_BPATEst' and 'Model' options, along with a search bar. The main area contains a DAX query:

```
1 DEFINE  
2 VAR __DSOCore =  
3     SUMMARIZECOLUMN(  
4         ROLLUPADDISSUBTOTAL(ROLLUPGROUP('Product'[Category], 'Product'[Brand]), "IsGrandTotalRowTotal"),  
5         "Margin", 'Sales'[Margin]  
6     )  
7  
8     VAR __DSOPrimarywindowed =  
9         TOPN(502, __DSOCore, [IsGrandTotalRowTotal], 0, 'Product'[Category], 1, 'Product'[Brand], 1)  
10  
11 EVALUATE  
12     __DSOPrimarywindowed  
13  
14 ORDER BY  
15     [IsGrandTotalRowTotal] DESC, 'Product'[Category], 'Product'[Brand]
```

At the bottom, there are tabs for Log, Results, and History. The Log tab displays a table with columns Start, Duration, and Message, showing two entries: 'Establishing Connection' at 16:34:01 and 'Connected' at 16:34:01.

	Start	Duration ⓘ	Message
ⓘ	16:34:01		Establishing Connection
ⓘ	16:34:01		Connected

Results - Like Query View

The screenshot shows the Power BI Query Editor interface. The top navigation bar includes File, Home, Advanced, Help, and various toolbar icons for running queries, canceling, clearing cache, and managing results.

The left sidebar displays the 'Metadata' tab, showing the current model and a search bar. Below this, a tree view lists the data sources: Customer, Date, Product, Sales, and Store.

The main area contains a DAX query:

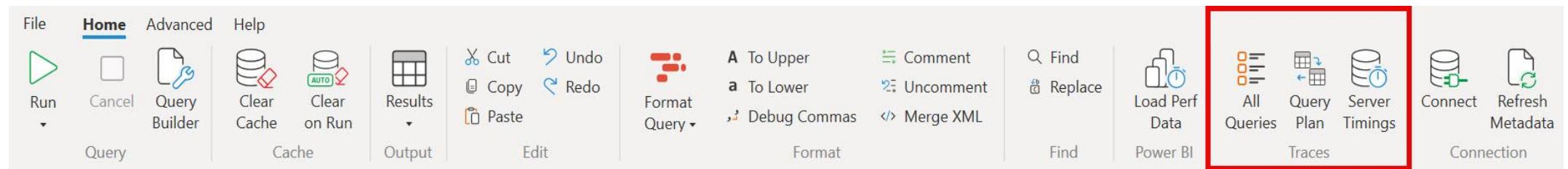
```
1 DEFINE  
2     VAR __DS0Core =  
3         SUMMARIZECOLUMNS(  
4             ROLLUPADDISSUBTOTAL(ROLLUPGROUP('Product'[Category], 'Product'[Brand]), "IsGrandTotalRowTotal"),  
5             "Margin", 'sales'[Margin]  
6         )  
7  
8     VAR __DS0Primarywindowed =  
9         TOPN(502, __DS0Core, [IsGrandTotalRowTotal], 0, 'Product'[Category], 1, 'Product'[Brand], 1)  
10  
11 EVALUATE  
12     __DS0Primarywindowed  
13  
14 ORDER BY  
15     [IsGrandTotalRowTotal] DESC, 'Product'[Category], 'Product'[Brand]
```

The results pane, highlighted with a red border, displays the output of the query as a table:

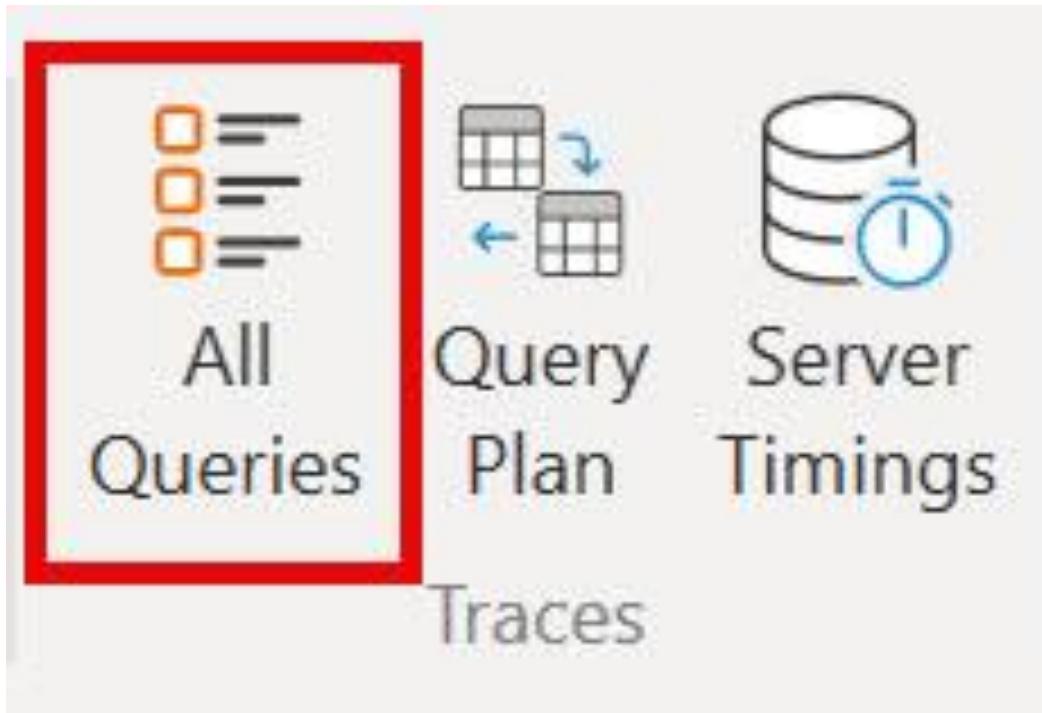
Category	Brand	IsGrandTotalRowTotal	Margin
		True	1267049808.6663
Audio	Contoso	False	5547636.5545
Audio	Northwind Traders	False	5276681.882
Audio	Wide World Importers	False	21868862.1847
Cameras and camcorders	A. Datum	False	28002980.6738
Cameras and camcorders	Contoso	False	19656707.4536
Cameras and camcorders	Fabrikam	False	71111794.4404
Cell phones	Contoso	False	22490172.075
Cell phones	The Phone Company	False	174204193.028
Computers	Adventure Works	False	189547913.3088

The bottom status bar indicates the query is ready, with details like line 15, column 50, and 30 rows processed.

Traces



DAX Studio - All Queries



Like SQL Server Profiler

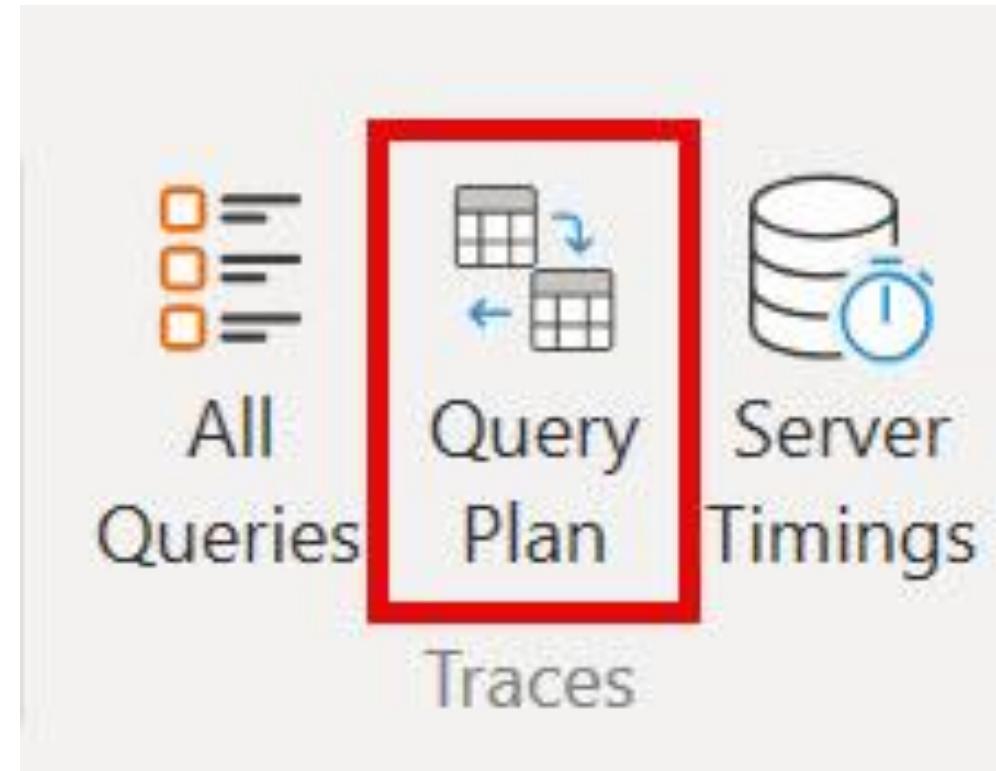
DAX Studio - All Queries

Log Results History **All Queries** ... X

Record Pause Stop Clear Copy Export Info Filters

StartTime	Type	Duration	User	Database	Query
10:12:54	DAX	10ms	MSI\roman	Contoso1M_Docker_BPATest	DEFINE VAR __DSOCore = SUMMARIZECOLUMNS(ROLLUPADDISSTOTAL(ROLLUPGROUP('Product'[Category], 'Product'[Brand]), "IsGrandTotalRowTotal"), "Margin", 'Sa...
10:12:51	DAX	0ms	MSI\roman	Contoso1M_Docker_BPATest	DEFINE VAR __DSOCore = SUMMARIZE('Product', 'Product'[Category], 'Product'[Brand]) VAR __DSOPrimaryWindowed = TOPN(501, __DSOCore, 'Product'[Category], 1, 'Pro...
10:12:50	DAX	2ms	MSI\roman	Contoso1M_Docker_BPATest	DEFINE VAR __DSOCore = DISTINCT('Product'[Category]) VAR __DSOPrimaryWindowed = TOPN(501, __DSOCore, 'Product'[Category], 1) EVALUATE __DSOPrimaryWindowed
10:11:45	DAX	187ms	MSI\roman	Contoso1M_Docker_BPATest	DEFINE VAR __DSOCore = SUMMARIZECOLUMNS(ROLLUPADDISSTOTAL(ROLLUPGROUP('Date'[Year Month], 'Product'[Brand]), "IsGrandTotalRowTotal"), "Total_BIG_Or...
10:11:39	DAX	9ms	MSI\roman	Contoso1M_Docker_BPATest	DEFINE VAR __DSOCore = SELECTCOLUMNS(KEEPFILTERS(FILTER(KEEPFILTERS(SUMMARIZECOLUMNS('Date'[Year Month], 'Product'[Brand], "CountRowsSales", COUNT...)
10:11:24	DAX	6ms	MSI\roman	Contoso1M_Docker_BPATest	EVALUATE ROW("MinBrand", CALCULATE(MIN('Product'[Brand])))
10:10:33	DAX	31,984ms	MSI\roman	Contoso1M_Docker_BPATest	DEFINE VAR __DSOCore = SUMMARIZECOLUMNS(ROLLUPADDISSTOTAL(ROLLUPGROUP('Date'[Year Month], 'Product'[Brand]), "IsGrandTotalRowTotal"), "Total_BIG_Or...

Query Plan



Query Plans

Two Types

- Logical
- Physical

DAX Studio – Query Plan

Log Results History Server Timings **Query Plan**

Record

Line	Records	Physical Query Plan
------	---------	---------------------

1 PartitionIntoGroups: IterPhyOp LogOp=Order IterCols(0, 1, 2, 3)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good], "[])) #Groups=1 #Rows=120
2 1 AggregationSpool<Order>: SpoolPhyOp #Records=1
3 Proxy: IterPhyOp LogOp=TableVarProxy IterCols(0, 1, 2, 3)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good], "[]))
4 Proxy: IterPhyOp LogOp=TableVarProxy IterCols(0, 1, 2, 3)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good], "[]))
5 Union: IterPhyOp LogOp=Union IterCols(0, 1, 2, 3)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good], "[]))
6 GroupSemijoin: IterPhyOp LogOp=GroupSemiJoin IterCols(0, 1, 2)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good])
7 119 Spool_Iterator<SpoolIterator>: IterPhyOp LogOp=DistinctCount_Vertipaq IterCols(0)('Date'[Year Month]) #Records=119 #KeyCols=70 #ValueCols=1
8 119 ProjectionSpool<ProjectFusion<Copy>>: SpoolPhyOp #Records=119
9 **Cache: IterPhyOp #FieldCols=1 #ValueCols=1**
10 GroupSemijoin: IterPhyOp LogOp=GroupSemiJoin IterCols(0, 1, 2)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good])
11 1 Spool_Iterator<SpoolIterator>: IterPhvOn IterOn=DistinctCount_Vertipaq #Records=1 #KeyCols=70 #ValueCols=1

Line	Logical Query Plan
------	--------------------

1 __DS0Core: Union: RelLogOp VarName=__DS0Core DependOnCols() 0-3 RequiredCols(0, 1, 2, 3)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good], "[]))
2 GroupSemiJoin: RelLogOp DependOnCols() 0-2 RequiredCols(0, 1, 2)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good])
3 Scan_Vertipaq: RelLogOp DependOnCols() 0-0 RequiredCols(0)('Date'[Year Month])
4 Constant: ScaLogOp DependOnCols() Boolean DominantValue=false
5 Calculate: ScaLogOp MeasureRef=[Total BIG Orders Good] DependOnCols(0)('Date'[Year Month]) Integer DominantValue=BLANK
6 DistinctCount_Vertipaq: ScaLogOp DependOnCols(0)('Date'[Year Month]) Integer DominantValue=BLANK
7 Scan_Vertipaq: RelLogOp DependOnCols(0)('Date'[Year Month]) 2-2 RequiredCols(0)('Date'[Year Month])
8 Filter_Vertipaq: RelLogOp DependOnCols() 1-1 RequiredCols(1)('Sales'[Net Price])
9 Scan_Vertipaq: RelLogOp DependOnCols() 1-1 RequiredCols(1)('Sales'[Net Price])
10 GreaterThan: ScaLogOp DependOnCols(1)('Sales'[Net Price]) Boolean DominantValue=NONE
11 'Sales'[Net Price]: ScaLogOp DependOnCols(1)('Sales'[Net Price]) Currency DominantValue=NONE

What the French Toast

Query Plan – One Useful Thing

Line	Records	Physical Query Plan
1		PartitionIntoGroups: IterPhyOp LogOp=Order IterCols(0, 1, 2, 3)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good], "[])) #Groups=1 #Rows=120
2	1	AggregationSpool<Order>: SpoolPhyOp #Records=1
3		Proxy: IterPhyOp LogOp=TableVarProxy IterCols(0, 1, 2, 3)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good], "[]))
4		Proxy: IterPhyOp LogOp=TableVarProxy IterCols(0, 1, 2, 3)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good], "[]))
5		Union: IterPhyOp LogOp=Union IterCols(0, 1, 2, 3)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good], "[]))
6		GroupSemijoin: IterPhyOp LogOp=GroupSemiJoin IterCols(0, 1, 2)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good])
7	119	Spool_Iterator<SpoolIterator>: IterPhyOp LogOp=DistinctCount_Vertipaq IterCols(0)('Date'[Year Month]) #Records=119 #KeyCols=70 #ValueCols=1
8	119	ProjectionSpool<ProjectFusion<Copy>>: SpoolPhyOp #Records=119
9		Cache: IterPhyOp #FieldCols=1 #ValueCols=1
10		GroupSemijoin: IterPhyOp LogOp=GroupSemiJoin IterCols(0, 1, 2)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good])
11	1	Spool_Iterator<SpoolIterator>: IterPhyOp LogOp=DistinctCount_Vertipaq IterCols(0)('Date'[Year Month]) #Records=1 #KeyCols=70 #ValueCols=1

Line	Logical Query Plan
1	_DS0Core: Union: RelLogOp VarName=_DS0Core DependOnCols() 0-3 RequiredCols(0, 1, 2, 3)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good], "[]))
2	GroupSemiJoin: RelLogOp DependOnCols() 0-2 RequiredCols(0, 1, 2)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good])
3	Scan_Vertipaq: RelLogOp DependOnCols() 0-0 RequiredCols(0)('Date'[Year Month])
4	Constant: ScaLogOp DependOnCols() Boolean DominantValue=false
5	Calculate: ScaLogOp MeasureRef=[Total BIG Orders Good] DependOnCols(0)('Date'[Year Month]) Integer DominantValue=BLANK
6	DistinctCount_Vertipaq: ScaLogOp DependOnCols(0)('Date'[Year Month]) Integer DominantValue=BLANK
7	Scan_Vertipaq: RelLogOp DependOnCols(0)('Date'[Year Month]) 2-2 RequiredCols(0)('Date'[Year Month])
8	Filter_Vertipaq: RelLogOp DependOnCols() 1-1 RequiredCols(1)('Sales'[Net Price])
9	Scan_Vertipaq: RelLogOp DependOnCols() 1-1 RequiredCols(1)('Sales'[Net Price])
10	GreaterThan: ScaLogOp DependOnCols(1)('Sales'[Net Price]) Boolean DominantValue=NONE
11	'Sales'[Net Price]: ScaLogOp DependOnCols(1)('Sales'[Net Price]) Currency DominantValue=NONE

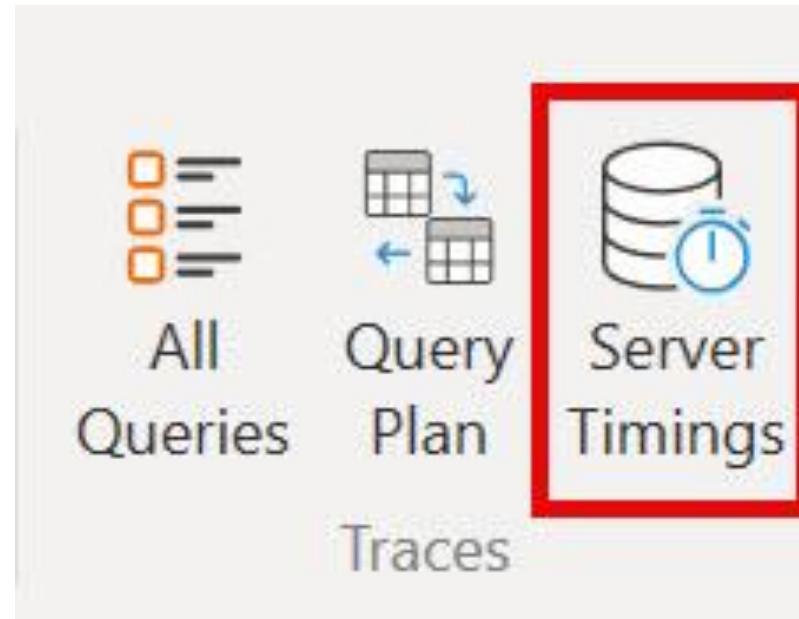
DAX Studio – Records

[Log](#)[Results](#)[History](#)[Server Timings](#)[Query Plan](#)[Record](#)[Pause](#)[Stop](#)[Clear](#)[Export](#)[Info](#)

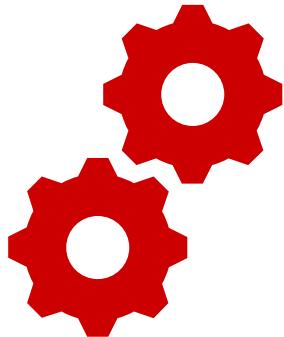
Line	Records	Physical Query Plan
1		PartitionIntoGroups: IterPhyOp LogOp=Order IterCols(0, 1, 2, 3)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good], "[])) #Groups=1 #Rows=120
2	1	AggregationSpool<Order>: SpoolPhyOp #Records=1
3		Proxy: IterPhyOp LogOp=TableVarProxy IterCols(0, 1, 2, 3)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good], "[]))
4		Proxy: IterPhyOp LogOp=TableVarProxy IterCols(0, 1, 2, 3)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good], "[]))
5		Union: IterPhyOp LogOp=Union IterCols(0, 1, 2, 3)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good], "[]))
6		GroupSemijoin: IterPhyOp LogOp=GroupSemiJoin IterCols(0, 1, 2)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good])
7	119	Spool_Iterator<SpoolIterator>: IterPhyOp LogOp=DistinctCount_Vertipaq IterCols(0)('Date'[Year Month]) #Records=119 #KeyCols=70 #ValueCols=1
8	119	ProjectionSpool<ProjectFusion<Copy>>: SpoolPhyOp #Records=119
9		Cache: IterPhyOp #FieldCols=1 #ValueCols=1
10		GroupSemijoin: IterPhyOp LogOp=GroupSemiJoin IterCols(0, 1, 2)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good])
11	1	Spool_Iterator<SpoolIterator>: IterPhyOp LogOp=DistinctCount_Vertipaq IterCols(0)('Date'[Year Month]) #Records=1 #KeyCols=70 #ValueCols=1

Line	Logical Query Plan
1	_DS0Core: Union: RelLogOp VarName=_DS0Core DependOnCols() 0-3 RequiredCols(0, 1, 2, 3)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good], "[]))
2	GroupSemiJoin: RelLogOp DependOnCols() 0-2 RequiredCols(0, 1, 2)('Date'[Year Month], "[IsGrandTotalRowTotal], "[Total_BIG_Orders_Good])
3	Scan_Vertipaq: RelLogOp DependOnCols() 0-0 RequiredCols(0)('Date'[Year Month])
4	Constant: ScaLogOp DependOnCols() Boolean DominantValue=false
5	Calculate: ScaLogOp MeasureRef=[Total BIG Orders Good] DependOnCols(0)('Date'[Year Month]) Integer DominantValue=BLANK
6	DistinctCount_Vertipaq: ScaLogOp DependOnCols(0)('Date'[Year Month]) Integer DominantValue=BLANK
7	Scan_Vertipaq: RelLogOp DependOnCols(0)('Date'[Year Month]) 2-2 RequiredCols(0)('Date'[Year Month])
8	Filter_Vertipaq: RelLogOp DependOnCols() 1-1 RequiredCols(1)('Sales'[Net Price])
9	Scan_Vertipaq: RelLogOp DependOnCols() 1-1 RequiredCols(1)('Sales'[Net Price])
10	GreaterThan: ScaLogOp DependOnCols(1)('Sales'[Net Price]) Boolean DominantValue=NONE
11	'Sales'[Net Price]: ScaLogOp DependOnCols(1)('Sales'[Net Price]) Currency DominantValue=NONE

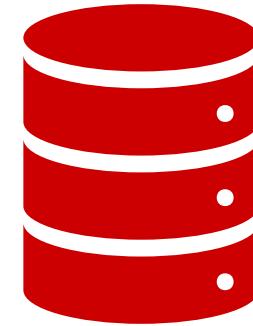
DAX Studio – Server Timings



A Tale of Two Engines



Formula Engine



Storage Engine

Formula Engine

- Conductor
- Does not cache
- Single threaded
- Complex Operations

Storage Engine

- Ability to cache
- Ability to Multithread
- Operations depend on the storage engine
 - Vertipaq is very limited
 - But optimized

Querying the Storage Engine

- Not Referring to refreshing the Storage Engine
 - Breaking Query Folding mainly refers to when you refresh the data in the Storage Engine (Vertipaq)

Beyond the Scope



Vertipaq (Import)

- Column Based
- Encoded
- Compressed

Storage Engine(s)

- Different Storage Engines
 - Vertipaq (Import)
 - Direct Query SQL (SQL Server, Snowflake)
- Combination of these
 - Import with Direct Query SQL
 - Direct Query over Tabular

Cache

File **Home** Advanced Help

Run ▾ Cancel Query Builder

Query

 Clear Cache

 Clear on Run

 Cache

Results ▾

Output

Location of Query and Server Timings

Metadata Functions DMV

Contoso1M_Docker_BPATEst ▾

Model ▾

Search

Customer

Date

Product

Sales

Store

```
1 // DAX Query
2 DEFINE
3     VAR __DSOFilterTable =
4         TREATAS({2020}, 'Date'[Year])
5
6     VAR __DSOCore =
7         SUMMARIZECOLUMNS(
8             ROLLUPADDISSTOTAL('Date'[Date], "IsGrandTotalRowTotal"),
9             __DSOFilterTable,
10            "Total_Quantity", 'Sales'[Total_Quantity],
11            "Total_BIG_Orders_Good", 'Sales'[Total_BIG_Orders_Good],
12            "Total_BIG_Orders_Bad", 'Sales'[Total_BIG_Orders_Bad]
13        )
14
15     VAR __DSOPrimarywindowed =
16         TOPN(502, __DSOCore, [IsGrandTotalRowTotal], 0, 'Date'[Date], 1)
17
18 EVALUATE
19     __DSOPrimarywindowed
20
21 ORDER BY
22     [IsGrandTotalRowTotal] DESC, 'Date'[Date]
```

Log Results History Server Timings Query Plan

	Start	Duration ⓘ	Message
ⓘ	14:06:42		Establishing Connection
ⓘ	14:06:43		Connected
ⓘ	14:31:47		Query Trace Started
ⓘ	14:31:48		Query Started
ⓘ	14:31:49		Query 1 Completed (64 rows returned)
✓	14:31:49 66		Query Batch Completed
ⓘ	14:32:00		Query Trace Started

DAX – Return Product Table

EVALUATE

'Product'

Storage Engine Query

The screenshot shows the Storage Engine Query interface with the following details:

- Top Bar:** Shows the command `1 EVALUATE` and the object `2 'Product'`.
- Toolbar:** Includes Log, Results, History, and Server Timings tabs. The Server Timings tab is active.
- Server Timings Panel:** Displays performance metrics:
 - Total: 37 ms
 - SE CPU: 0 ms (x0.0)
 - FE: 36 ms (97.3%)
 - SE: 1 ms (2.7%)It also shows the number of SE Queries (1) and SE Cache (0).
- Timeline View:** A table showing query details. The second row is highlighted with a red border and shows:
 - Line: 2
 - Subclass: Scan
 - Duration: 1 ms
 - CPU: 0 ms
 - Par.: 0
 - Rows: 2,517
 - KB: 148
 - Timeline: A horizontal bar indicating the execution time.
 - Query: `SELECT 'Product'[RowNumber], 'Product'[ProductKey],`
- SQL Panel:** Displays the full SQL query:

```
SET DC_KIND="AUTO";
SELECT
    'Product'[RowNumber],
    'Product'[ProductKey],
    'Product'[Product Code],
    'Product'[Product Name],
    'Product'[Manufacturer],
    'Product'[Brand],
    'Product'[Color],
    'Product'[Weight Unit Measure],
    'Product'[Weight],
    'Product'[Unit Cost],
    'Product'[Unit Price],
    'Product'[Subcategory Code],
    'Product'[Subcategory],
    'Product'[Category Code],
    'Product'[Category]
FROM 'Product';
```

Estimated size: rows = 2,517 bytes = 151,020
- Bottom Right:** A large red arrow points downwards from the SQL panel towards the bottom of the interface.

Vertipaq - xmSQL

SELECT

```
'Product'[RowNumber],  
'Product'[ProductKey],  
'Product'[Product Code],  
'Product'[Product Name],  
'Product'[Manufacturer],  
'Product'[Brand],  
'Product'[Color],  
'Product'[Weight Unit Measure],  
'Product'[Weight],  
'Product'[Unit Cost],  
'Product'[Unit Price],  
'Product'[Subcategory Code],  
'Product'[Subcategory],  
'Product'[Category Code],  
'Product'[Category]  
FROM 'Product';
```

Storage Engine Query - Timeline

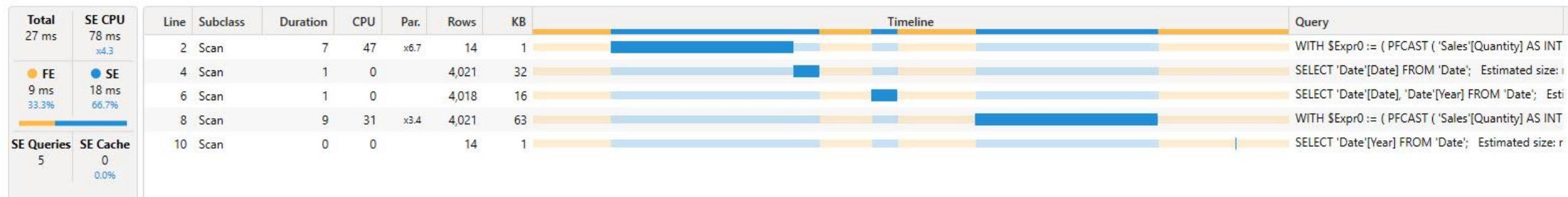
This one is mainly in FE.
Blue shows where SE comes into the timeline

Line	Subclass	Duration	CPU	Par.	Rows	KB	Timeline	Query
2	Scan	0	0		4,021	32		SELECT 'Date'[Date] FROM 'Date'; Estimated size: rows = 4,021 bytes = 32,168
4	Scan	0	0		4,021	32		SELECT 'Date'[Date] FROM 'Date'; Estimated size: rows = 4,021 bytes = 32,168
6	Scan	0	0		4,021	63		SELECT 'Date'[Date], MAX ('Date'[Date]) FROM 'Date'; Estimated size: rows = 4,021 bytes = 64,336
8	Scan	7	31	x4.4	4,021	63		WITH \$Expr0 := (PFCAST ('Sales'[Quantity] AS INT) * PFCAST ('Sales'[Net Price] AS INT)) SELECT 'Date'[Date], SUM (@Expr0) FR

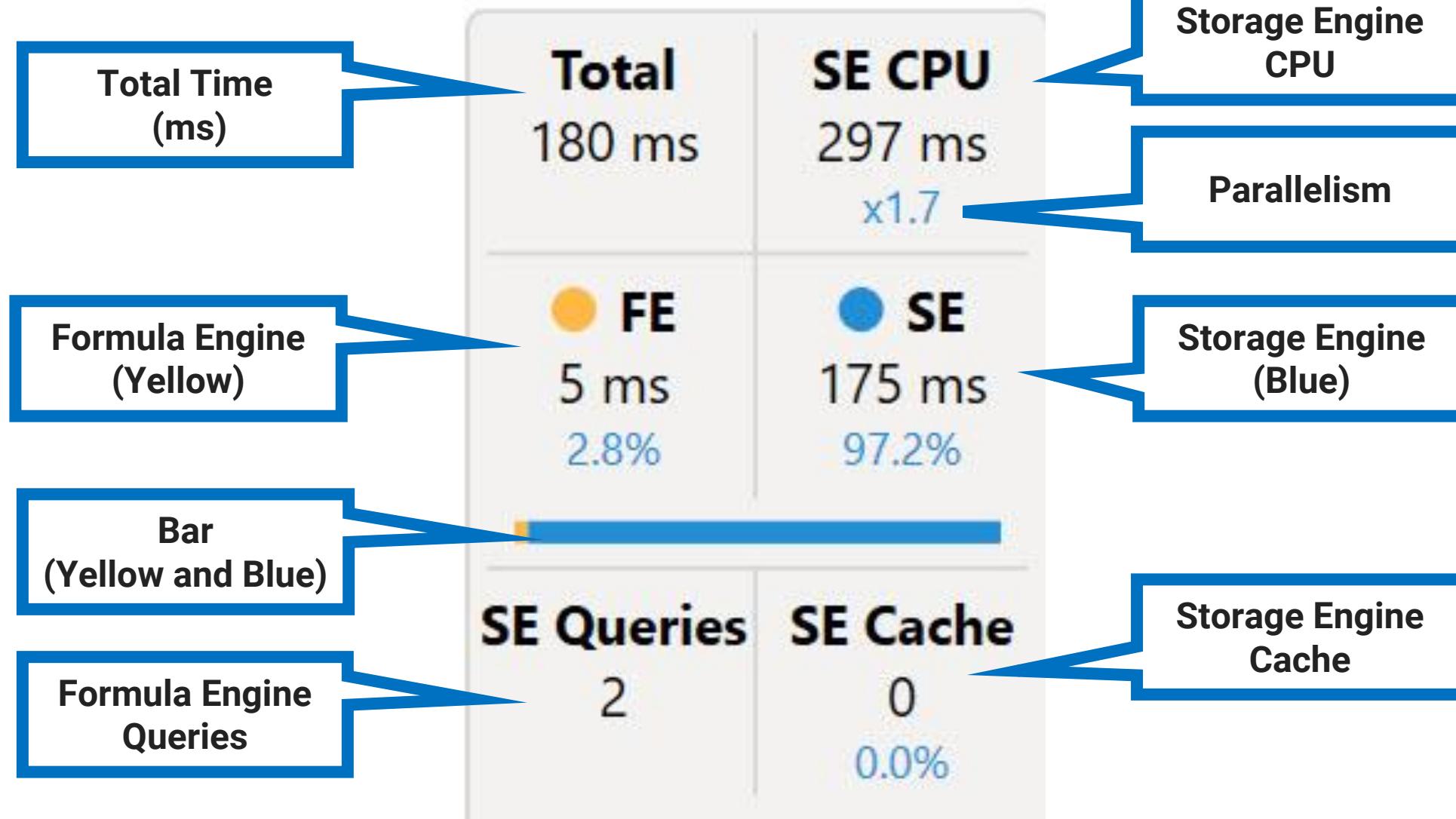
Storage Engine Query - Timeline

This one 33% in FE.

Blue shows where SE comes in the timeline.



Server Timings

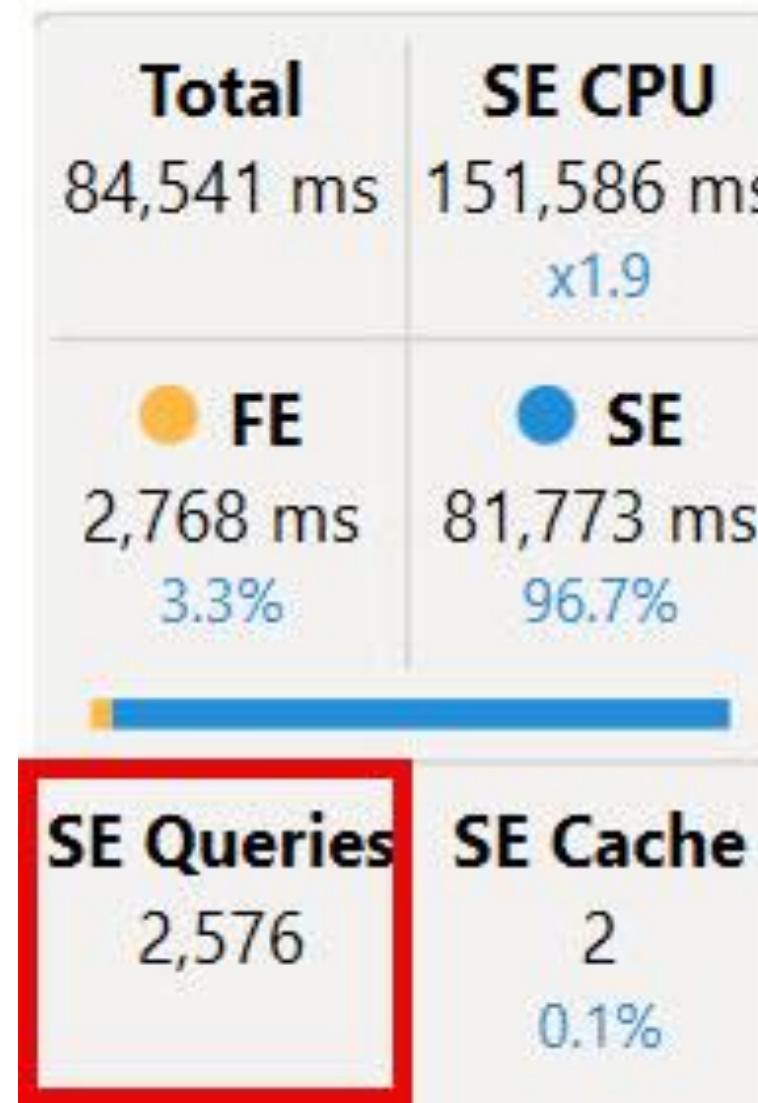


Simple way to increase Query Time

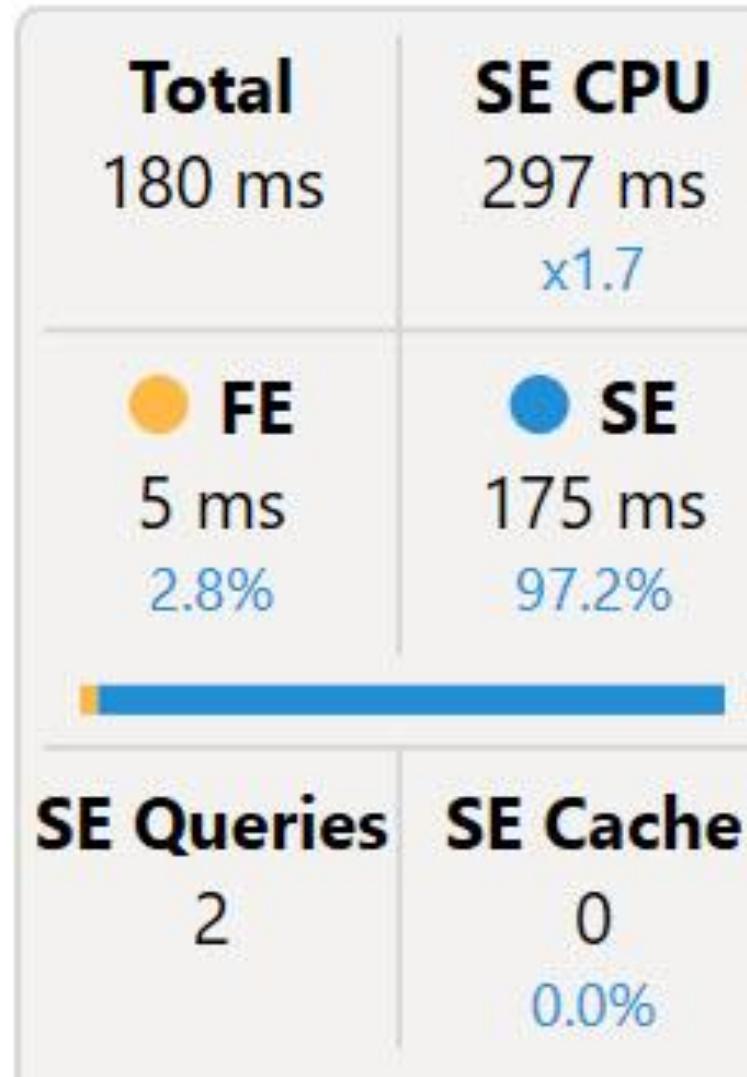
Filter by the full table

```
CALCULATE (  
    [Sales Amount],  
    FILTER (Sales, Sales[Quantity] > 1  
)
```

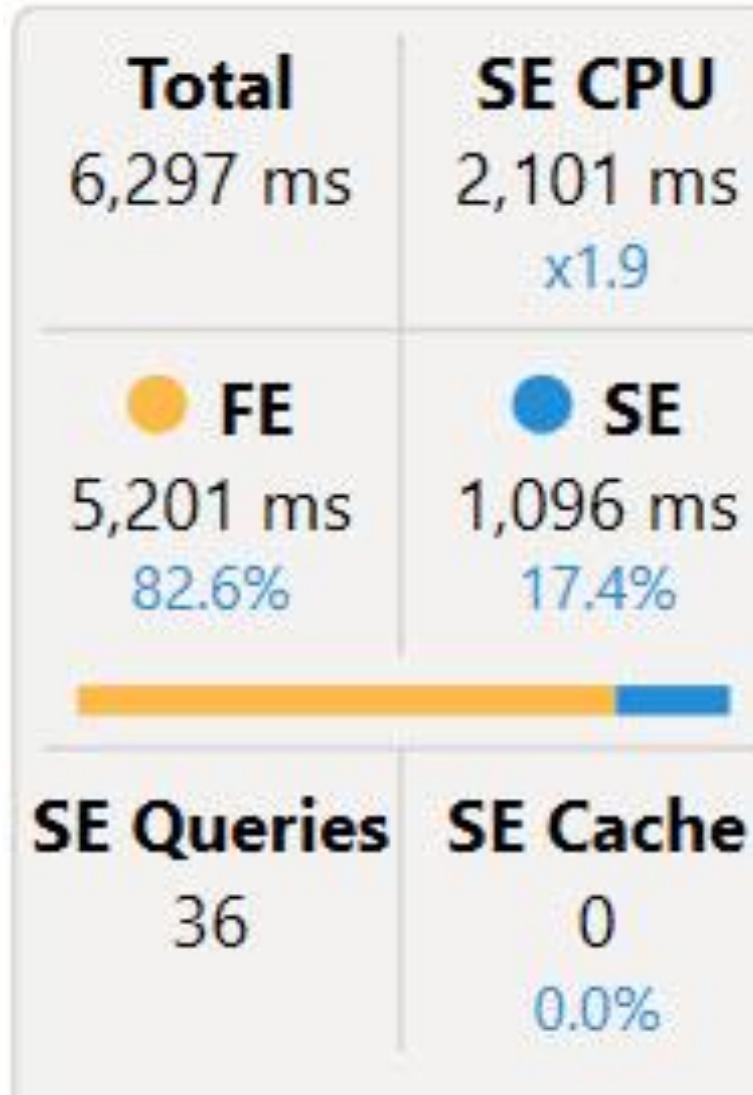
DAX Studio – Server Timings - Bad



DAX Studio – Server Timings - Good



DAX Studio – Heavy Formula Engine



Ways to invoke Formula Engine (Vertipaq)

- **IF Statements**
- **CONCATENATE**

CallbackDataID

WHERE

```
( COALESCE ( [CallbackDataID] ( IF (  
'Product'[Color] = "Blue", 1,0 ) ) ) ]  
( PFDATAID ( 'Product'[Color] ) ) ) <> 0 );
```

The more you know...

- If you can think like the engines
 - You can anticipate performance issues

Power BI Demo

DAX Studio

Our Voyage



1. Intro
2. Performance Analyzer
3. DAX Studio
4. Tabular Editor
5. Conclusion

Proactive Measures*

(*no pun intended)

- How do we find issues with the Model and DAX ahead of time

Tabular Editor

- Main Uses
 - Develop the model
 - Make changes
 - Audit the model
 - Best Practice Analyzer

Tabular Editor 2.x

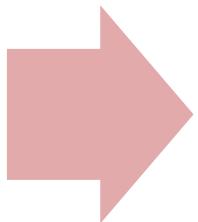
- Download
 - <https://github.com/TabularEditor/TabularEditor>
- Free version
- Version listed in DP-600 Study Guide

Tabular Editor 3.x

- Paid Version
- Extra Features
 - DAX Debugger
 - Script DAX

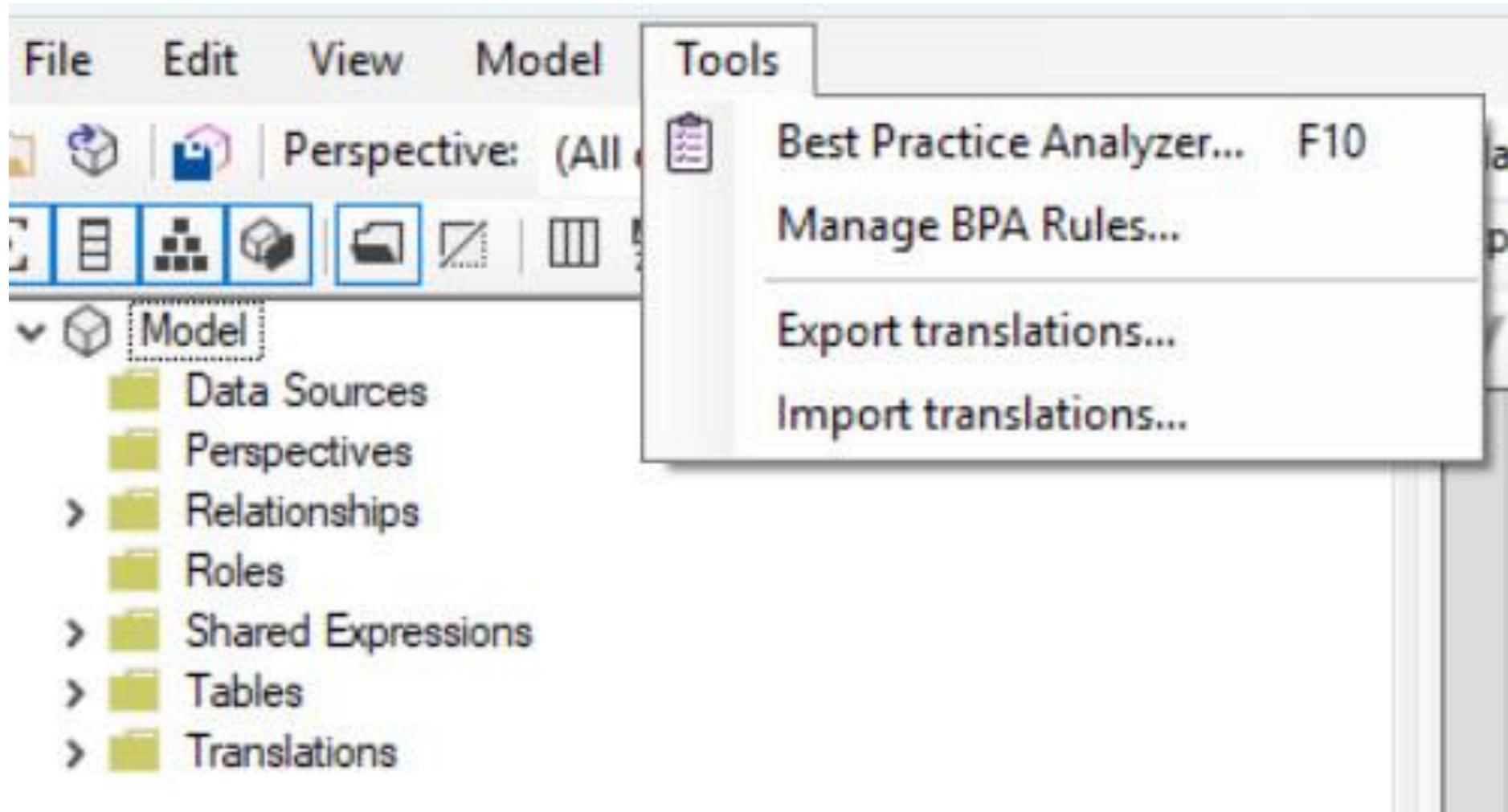
Tabular Editor 2

Optimize a semantic
model by using
Tabular Editor 2

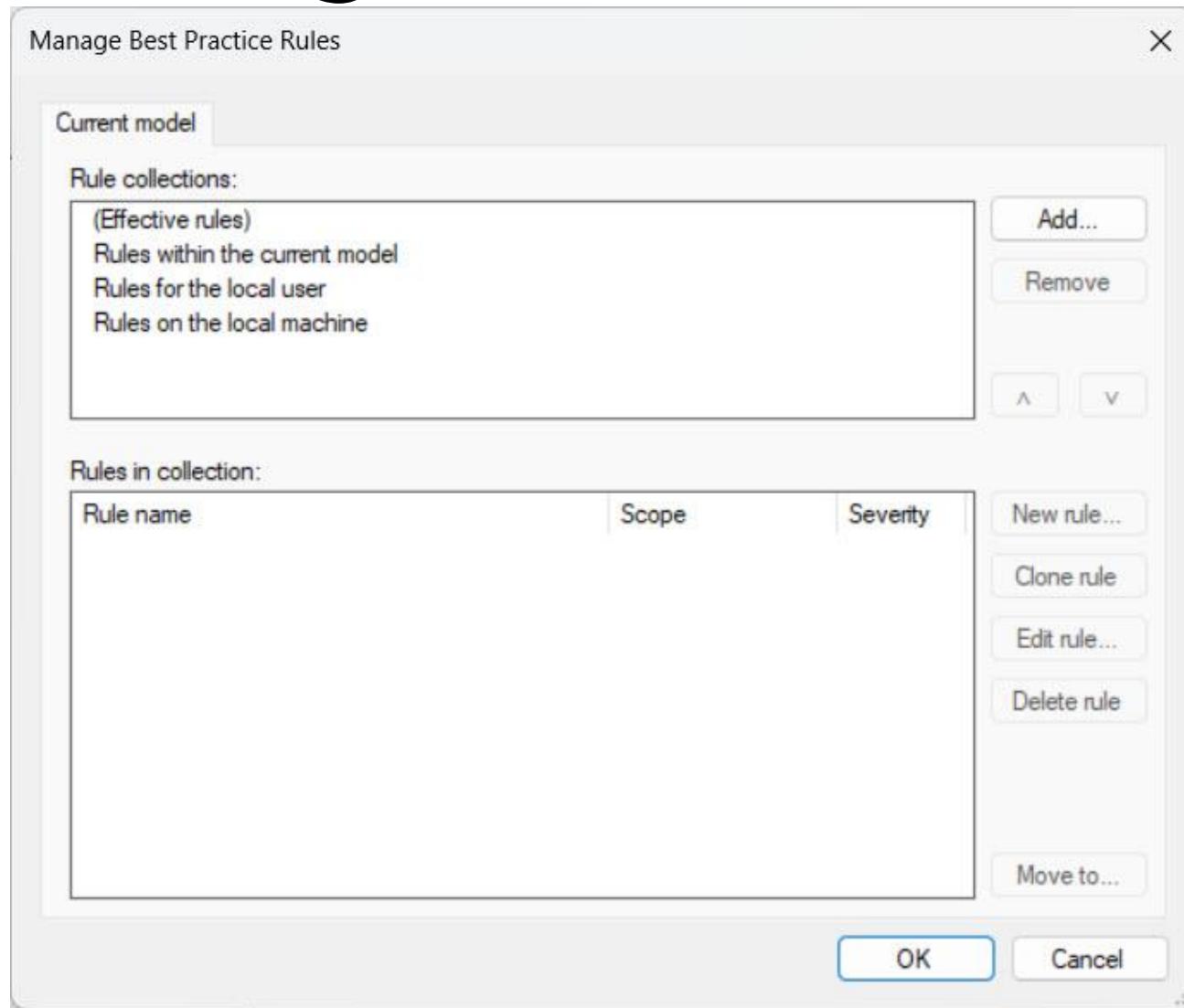


Best Practice Analyzer

Best Practice Analyzer Location



Add and Manage Rules



Best Practice Analyzer – C# Script Download

```
System.Net.WebClient w = new System.Net.WebClient();

string path = System.Environment.GetFolderPath(System.Environment.SpecialFolder.LocalApplicationData);
string url = "https://raw.githubusercontent.com/microsoft/Analysis-Services/master/BestPracticeRules/BPARules.json";
string version = System.Windows.Forms.Application.ProductVersion.Substring(0,1);
string downloadLoc = path + @"\TabularEditor\BPARules.json";

if (version == "3")
{
    downloadLoc = path + @"\TabularEditor3\BPARules.json";
}

w.DownloadFile(url, downloadLoc);

/*
// Italian
string url = "https://raw.githubusercontent.com/microsoft/Analysis-Services/master/BestPracticeRules/Italian/BPARules.json";

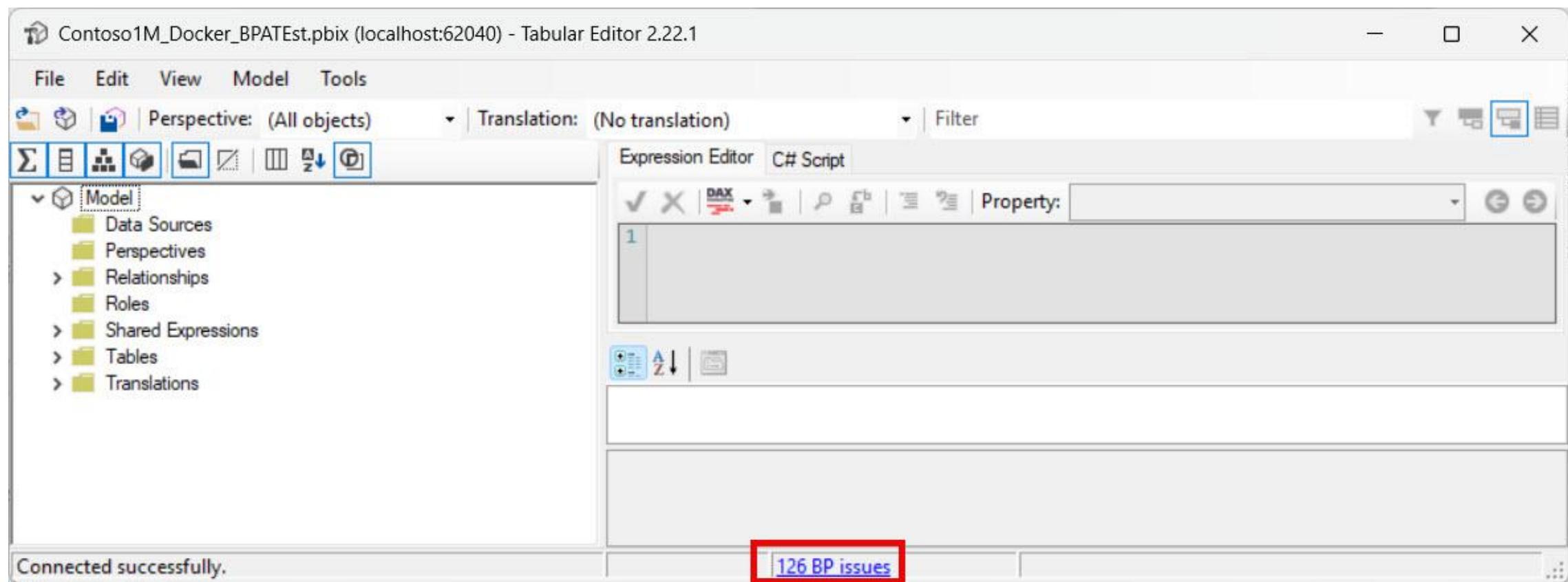
// Japanese
string url = "https://raw.githubusercontent.com/microsoft/Analysis-Services/master/BestPracticeRules/Japanese/BPARules.json";

// Spanish
string url = "https://raw.githubusercontent.com/microsoft/Analysis-Services/master/BestPracticeRules/Spanish/BPARules.json";
*/
```

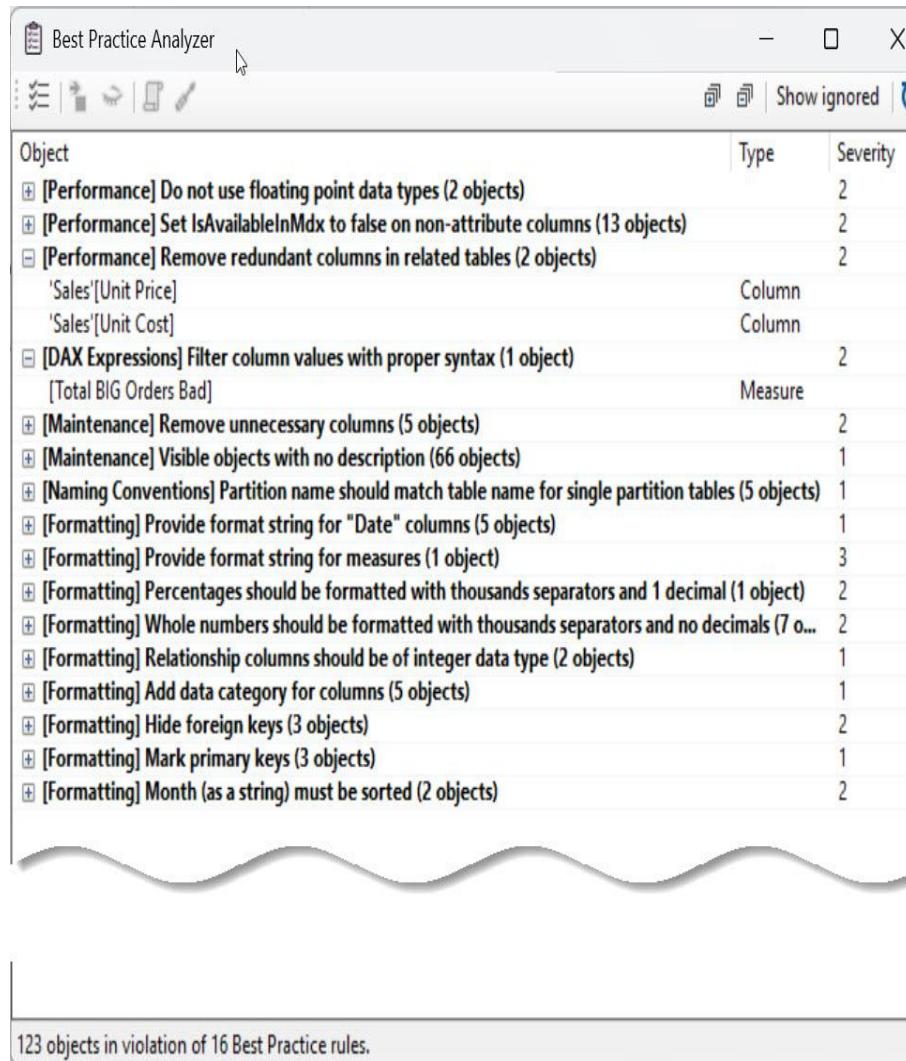
Nope – Not Running a Script

- Alternate Method
 - Microsoft Analysis-Services – Best Practice Rules
 - <https://github.com/microsoft/Analysis-Services/tree/master/BestPracticeRules>
 - Other languages besides English

BPA – Rule Exceptions for model



BPA – Rule Exceptions for model



The screenshot shows the Best Practice Analyzer (BPA) interface. The window title is "Best Practice Analyzer". The toolbar includes icons for refresh, search, and export. A button labeled "Show ignored" with a circular arrow icon is also present. The main area is a table with three columns: "Object", "Type", and "Severity". The table lists 16 rule violations across four categories: Performance, DAX Expressions, Maintenance, and Naming Conventions. The violations include recommendations like "Do not use floating point data types", "Set IsAvailableInMdx to false on non-attribute columns", and "Partition name should match table name for single partition tables". The severity levels range from 1 to 3.

Object	Type	Severity
[Performance] Do not use floating point data types (2 objects)		2
[Performance] Set IsAvailableInMdx to false on non-attribute columns (13 objects)		2
[Performance] Remove redundant columns in related tables (2 objects)		2
'Sales'[Unit Price]	Column	
'Sales'[Unit Cost]	Column	
[DAX Expressions] Filter column values with proper syntax (1 object)		2
[Total BIG Orders Bad]	Measure	
[Maintenance] Remove unnecessary columns (5 objects)		2
[Maintenance] Visible objects with no description (66 objects)		1
[Naming Conventions] Partition name should match table name for single partition tables (5 objects)		1
[Formatting] Provide format string for "Date" columns (5 objects)		1
[Formatting] Provide format string for measures (1 object)		3
[Formatting] Percentages should be formatted with thousands separators and 1 decimal (1 object)		2
[Formatting] Whole numbers should be formatted with thousands separators and no decimals (7 o...		2
[Formatting] Relationship columns should be of integer data type (2 objects)		1
[Formatting] Add data category for columns (5 objects)		1
[Formatting] Hide foreign keys (3 objects)		2
[Formatting] Mark primary keys (3 objects)		1
[Formatting] Month (as a string) must be sorted (2 objects)		2

123 objects in violation of 16 Best Practice rules.

BPA – Manage Rules

DAX Expressions

Rules in collection:	Scope	Severity
Rule name		
<input type="checkbox"/> DAX Expressions		
<input checked="" type="checkbox"/> [DAX Expressions] Avoid using '1-(x/y)' syntax	Measures,Calculated Columns,Calculation Items	2
<input checked="" type="checkbox"/> [DAX Expressions] Avoid using the IFERROR function	Measures,Calculated Columns	2
<input checked="" type="checkbox"/> [DAX Expressions] Column references should be fully qualified	Measures,KPIs,Table Permissions,Calculation Items	3
<input checked="" type="checkbox"/> [DAX Expressions] Filter column values with proper syntax	Measures,Calculated Columns,Calculation Items	2
<input checked="" type="checkbox"/> [DAX Expressions] Filter measure values by columns, not tables	Measures,Calculated Columns,Calculation Items	2
<input checked="" type="checkbox"/> [DAX Expressions] Inactive relationships that are never activated	Relationships	2
<input checked="" type="checkbox"/> [DAX Expressions] Measure references should be unqualified	Measures,Calculated Columns,Calculated Tables,KPIs,Calculation Items	3
<input checked="" type="checkbox"/> [DAX Expressions] Measures should not be direct references of other measures	Measures	2
<input checked="" type="checkbox"/> [DAX Expressions] No two measures should have the same definition	Measures	2
<input checked="" type="checkbox"/> [DAX Expressions] The EVALUATEANDLOG function should not be used in production models	Measures	1
<input checked="" type="checkbox"/> [DAX Expressions] Use the DIVIDE function for division	Measures,Calculated Columns,Calculation Items	2
<input checked="" type="checkbox"/> [DAX Expressions] Use the TREATAS function instead of INTERSECT for virtual relationships	Measures,Calculation Items	2

What about that slow pattern?



Filter by the Whole Table

```
CALCULATE (
    [Sales Amount],
    FILTER (Sales, Sales[Quantity] > 1
)
```

Edit Rule

Edit Best Practice Rule X

Name	[DAX Expressions] Filter measure values by columns, not tables	Severity	2	Category	DAX Expressions
ID	FILTER_MEASURE_VALUES_BY_COLUMNS				
Description	Option 1: FILTER(VALUES(Table'[Column]).[Measure] > Value) Option 2: FILTER(ALL(Table [Column]).[Measure] > Value)				
<p>Reference: https://docs.microsoft.com/power-bi/guidance/dax-avoid-avoid-filter-as-filter-argument</p>					
Applies to	Calculated Columns, Calculation Items, Measures	satisfying the following criteria:			
<p>Rule Expression Editor</p> <pre>RegEx.IsMatch(Expression, "(?i)CALCULATE\s*\((\s*[^\,]+,\s*(?i)FILTER\s*\((\s*\'*[A-Za-z0-9\]+\'*\s*,\s*\[[^\]]\]+\])") or RegEx.IsMatch(Expression, "(?i)CALCULATETABLE\s*\(([^\,]*,\s*(?i)FILTER\s*\((\s*\'*[A-Za-z0-9\]+\'*,\s*\[")</pre>					
Minimum Compatibility Level	CL 1200 (SQL Server 2016 / Azure AS)	OK Cancel			

Description with help link

- Instead of using this pattern FILTER('Table','Table'[Column]="Value") for the filter parameters of a CALCULATE or CALCULATETABLE function, use one of the options below. As far as whether to use the KEEPFILTERS function, see the second reference link below.
- Option 1: KEEPFILTERS('Table'[Column]="Value")
- Option 2: 'Table'[Column]="Value"
- Reference: <https://docs.microsoft.com/power-bi/guidance/dax-avoid-avoid-filter-as-filter-argument>
- Reference: <https://www.sqlbi.com/articles/using-keepfilters-in-dax/>

Another Source

9. Filtering a column

The FILTER function is often overused. Its main purpose is for filtering columns based on measure values. If you're just filtering a column value, there's generally no need to use this function. In fact, using it in that scenario often degrades performance.

Don't use this logic:

US Revenue = **CALCULATE** ([Revenue], **FILTER** ('Geography', 'Geography'[Area] = "United States"))

Option 1:

US Revenue 1 = **CALCULATE** ([Revenue], 'Geography'[Area] = "United States")

Option 2:

US Revenue 2 = **CALCULATE** ([Revenue], **KEEPFILTERS** ('Geography'[Area] = "United States"))

[Top 10 Power BI mistakes and their best practice solutions](https://www.elegantbi.com/post/top10bestpractices)
(<https://www.elegantbi.com/post/top10bestpractices>)

Change Description

- Link to company best practices
- Link to SharePoint that has more information

Power BI Demo

Tabular Editor

Our Voyage



1. Intro
2. Performance Analyzer
3. DAX Studio
4. Tabular Editor
5. Conclusion

Conclusion

- Slow Report?
 - Start with the report
 - Performance Analyzer
 - DAX Studio
 - Tabular Editor

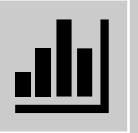
Take Away

- Time Invested in these tools
 - Can only enhance your skillset
 - Can make you more productive
- Power BI Desktop can only do so much

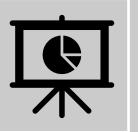
Resources



[Tabular Editor – Wonderful Training Free](#)



[Tabular Editor – Blog Posts](#)



[DAX Studio](#)



[Tabular Editor 2.x](#)

Resources



[Data Goblins - Sample Datasets](#)



[Data Mozart - Lots of DP-600 Resources](#) [Visual Speed](#)

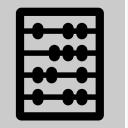


[The Definitive Guide to DAX - 2nd Edition](#)



[Optimizing DAX Book 2nd Edition](#)

Resources



[Elegant BI Blog \(Excellent Source\)](#)



[Microsoft Best Practice Rules Blog](#)



[Microsoft Github Best Practice Rules](#)



[DP-600 Exam Prep Book](#)

Community Support

Name	Type	Frequency	Join
San Diego SQL Server User Group	Hybrid	Every 3 rd Thursday monthly @ 6.30PM PDT	
LA Data Platform User Group	Virtual	Every 3 rd Wednesday monthly @ 7PM PDT	
San Francisco Microsoft Fabric Meetup	Virtual	Every 2 nd Wednesday monthly @ 7PM PDT	
Sacramento SQL Server User Group	In-Person	Every 1 st Wednesday monthly @ 6PM	

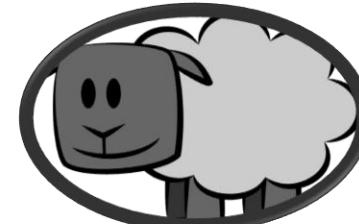
[Find more Azure Data Tech Groups on Meetup](#)

Thank you

Jason Romans

thedaxshepherd@gmail.com

www.thedaxshepherd.com



The Dax Shepherd

