Power up your Fabric Development with DAX Studio and Tabular Editor

SQL Saturday Albany

August 2024

Jason Romans

Senior BI Engineer

Builder of Models

redgate COMMUNITY AMBASSADOR 2024

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

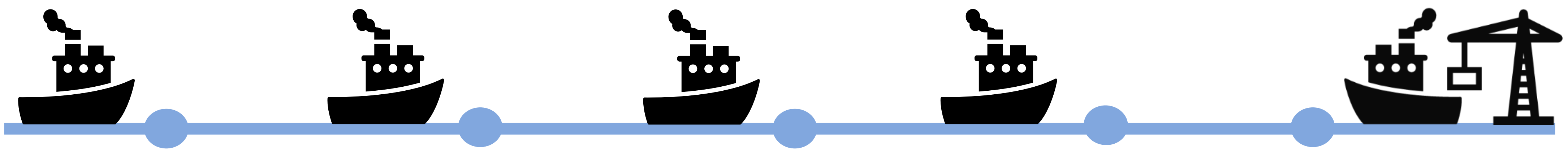
Favorite Data Model

Session Evaluations

- Located in your attendee bag
- Feedback does help

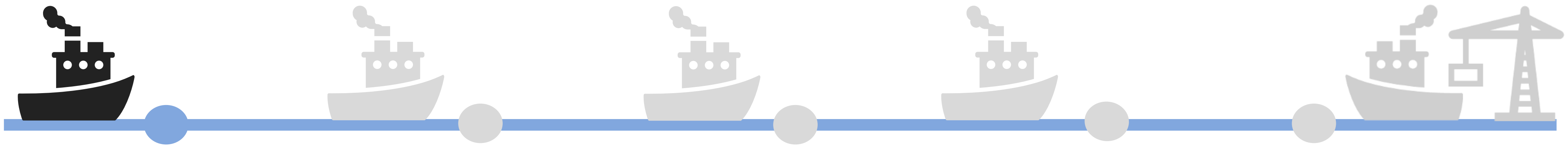
Shoulders of Giants

Our Journey



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

Our Journey



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
    - Default Semantic Model
- 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**

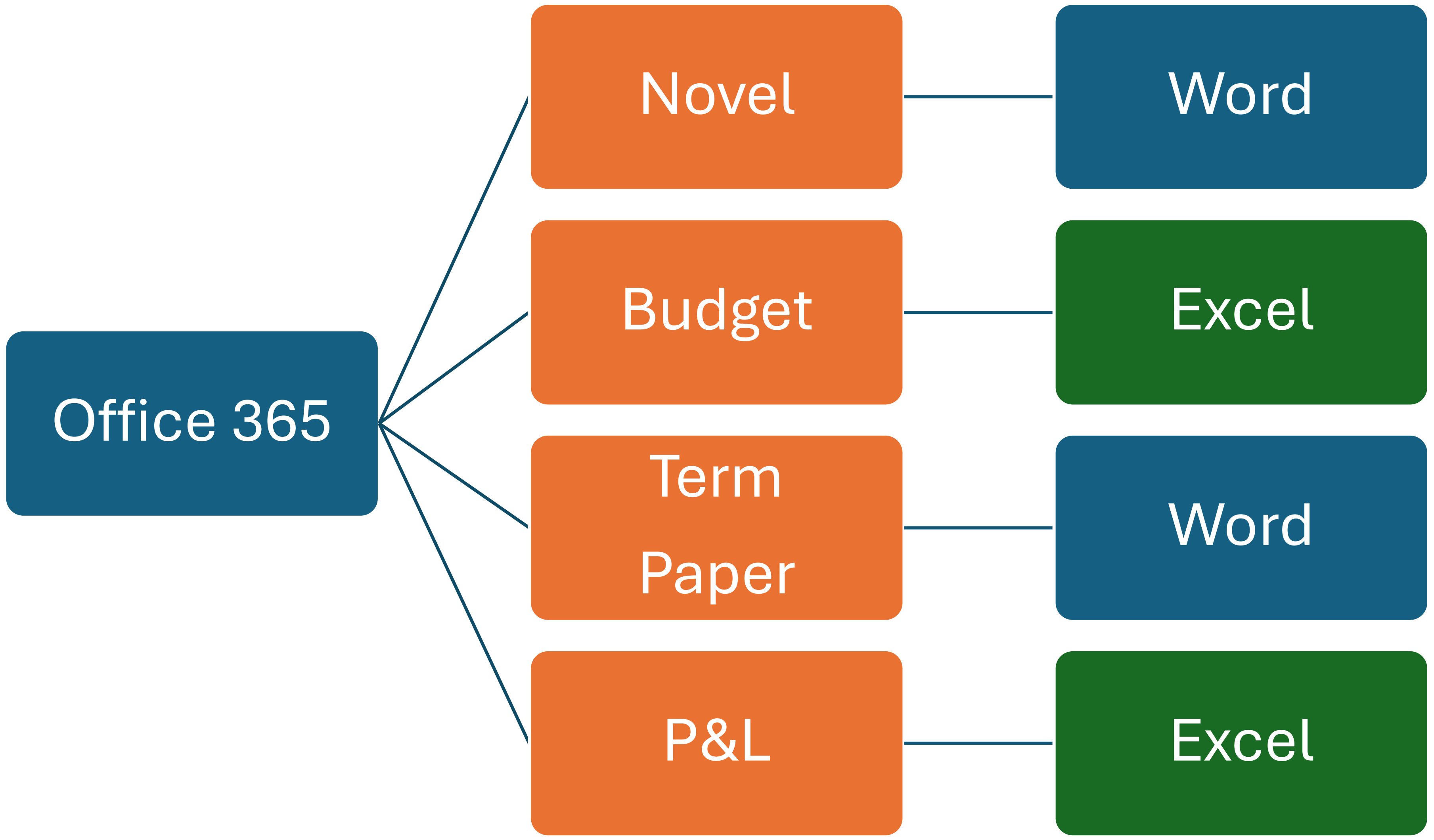
Optimize enterprise-scale semantic models

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

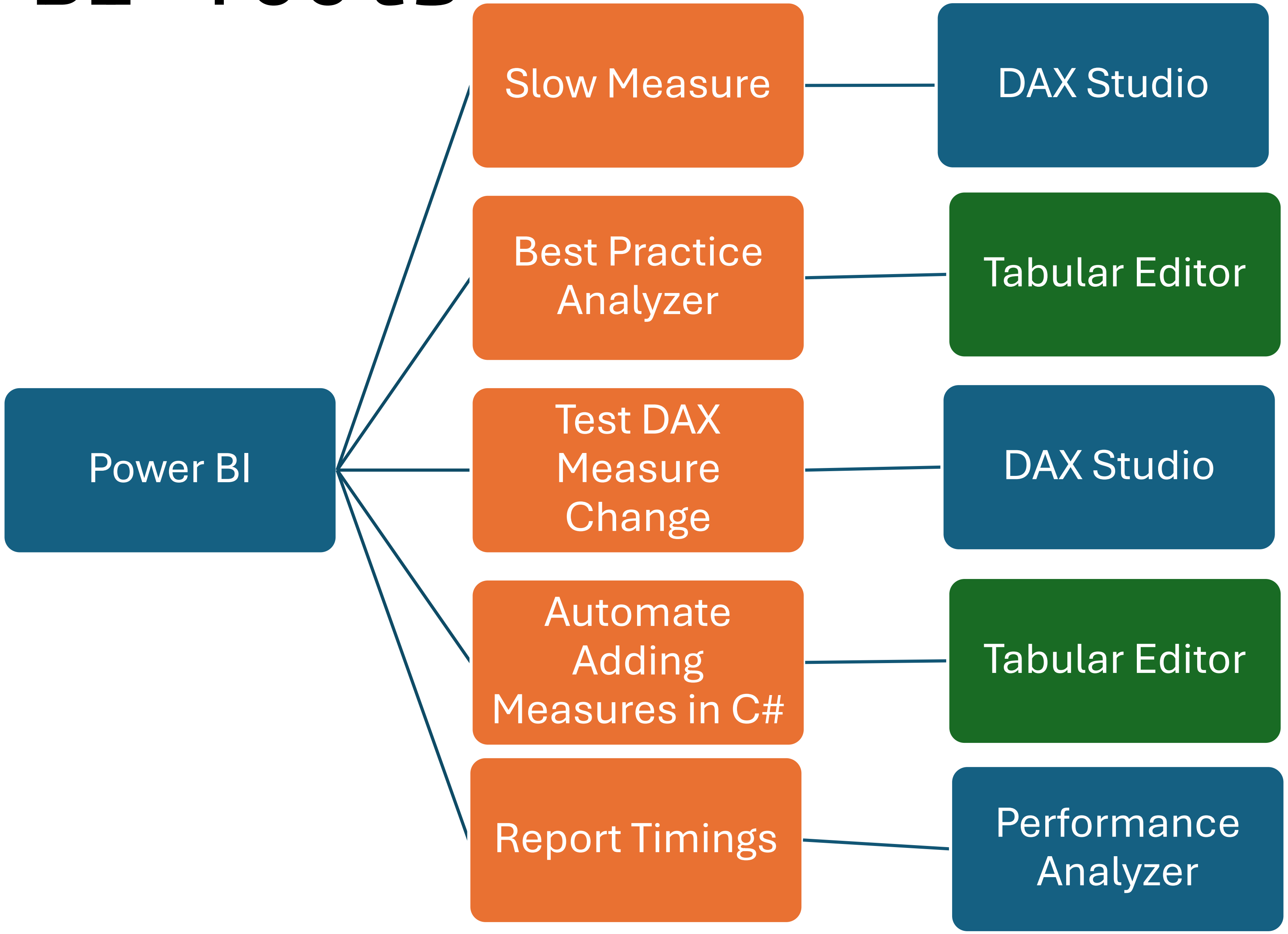
Why is my report slow?

- This should be our motivation
- This can have a lasting impact
  - Power BI Developer Skills
  - Career Development

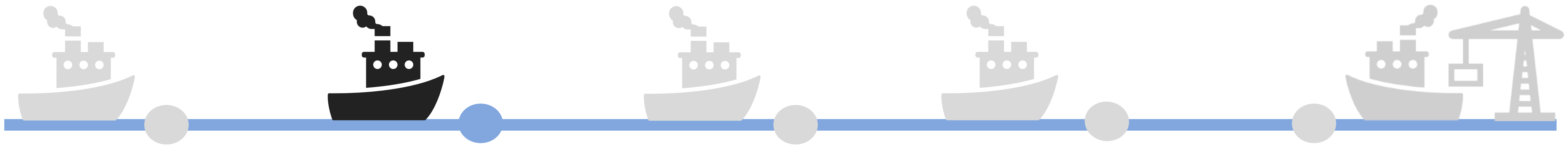
Office 365



Power BI Tools



Our Journey

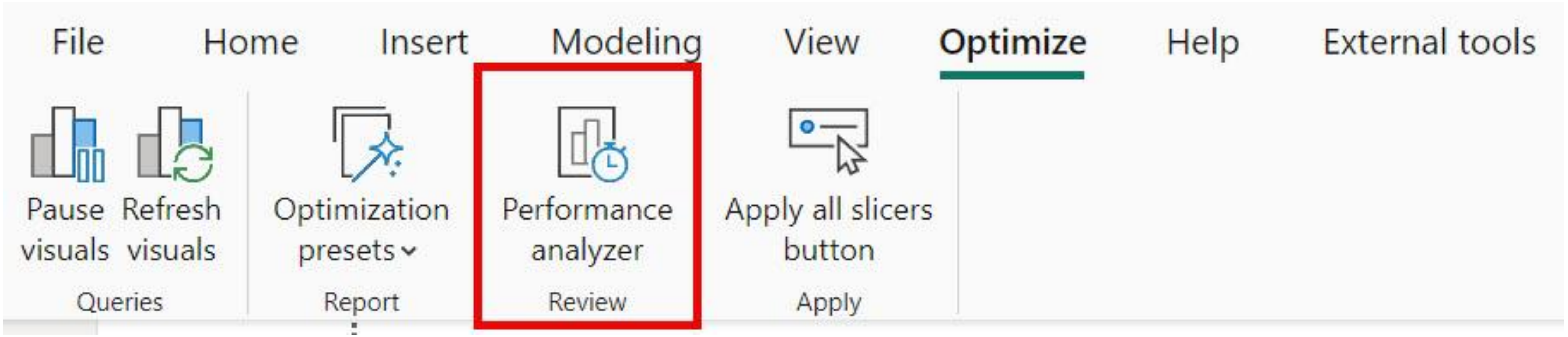


1. Intro
2. **Performance Analyzer**
3. DAX Studio
4. Tabular Editor
5. Conclusion

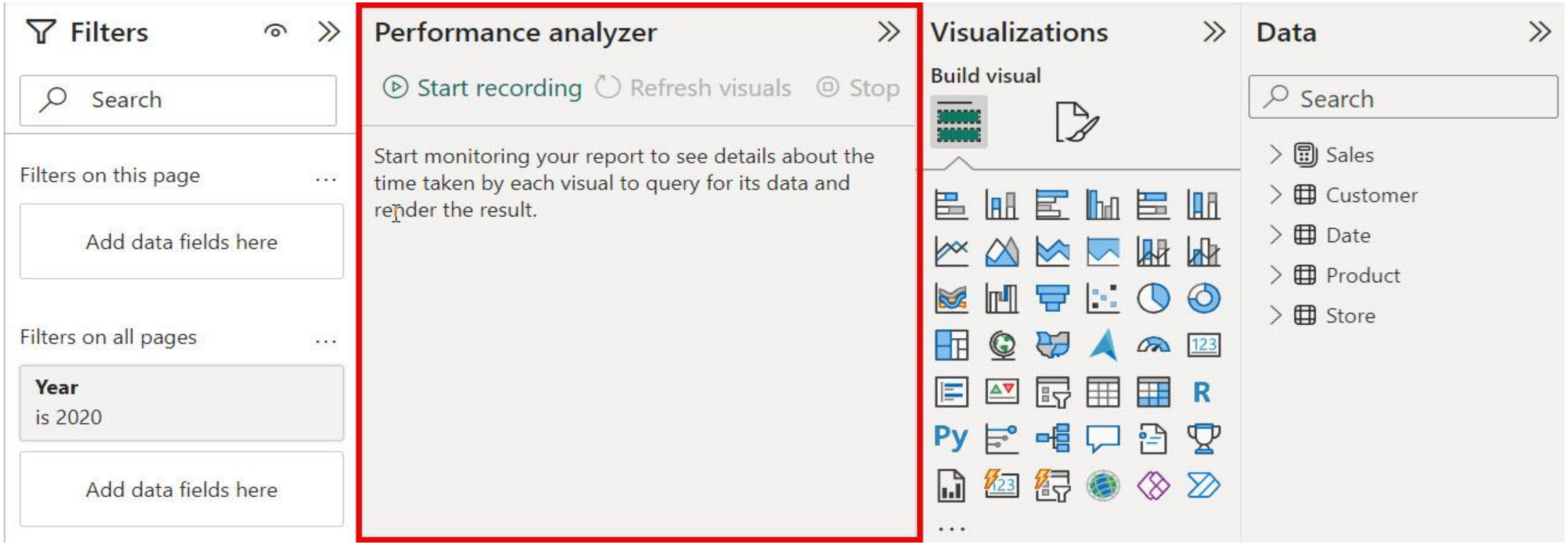
Performance Analyzer

- 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

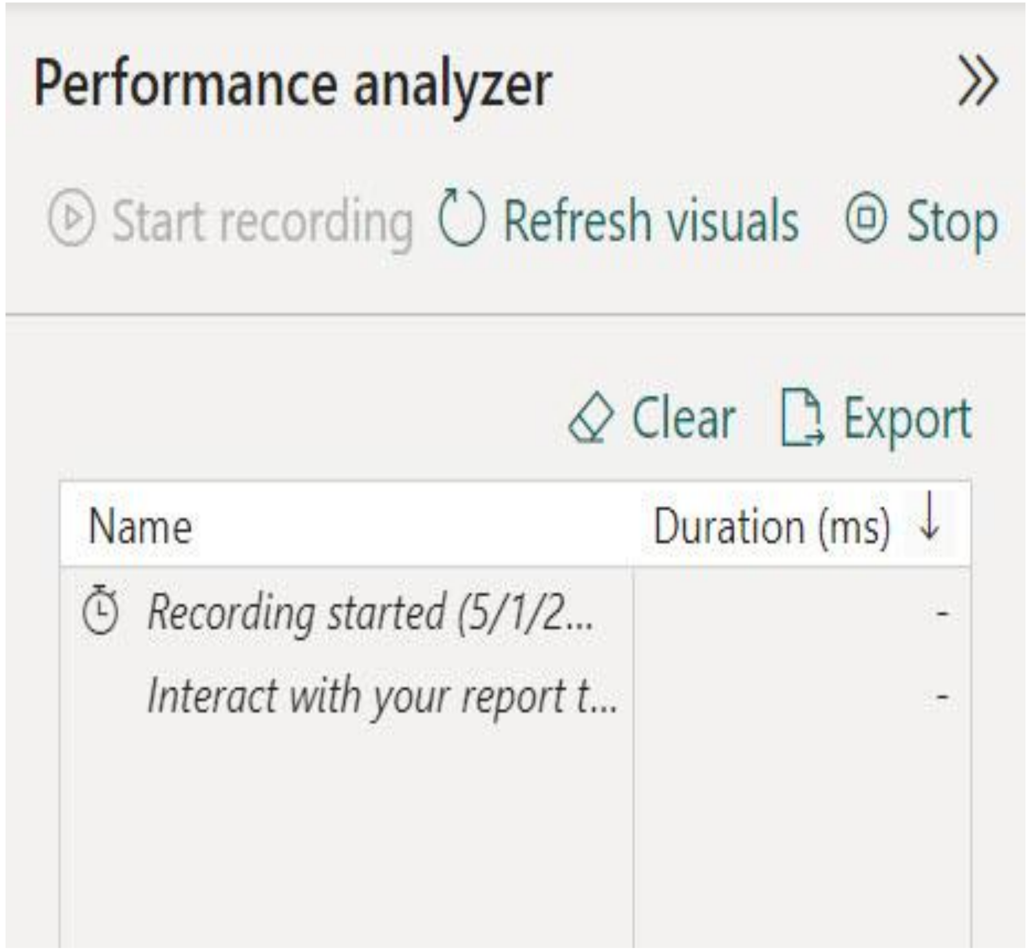
Build visual

Data

Search

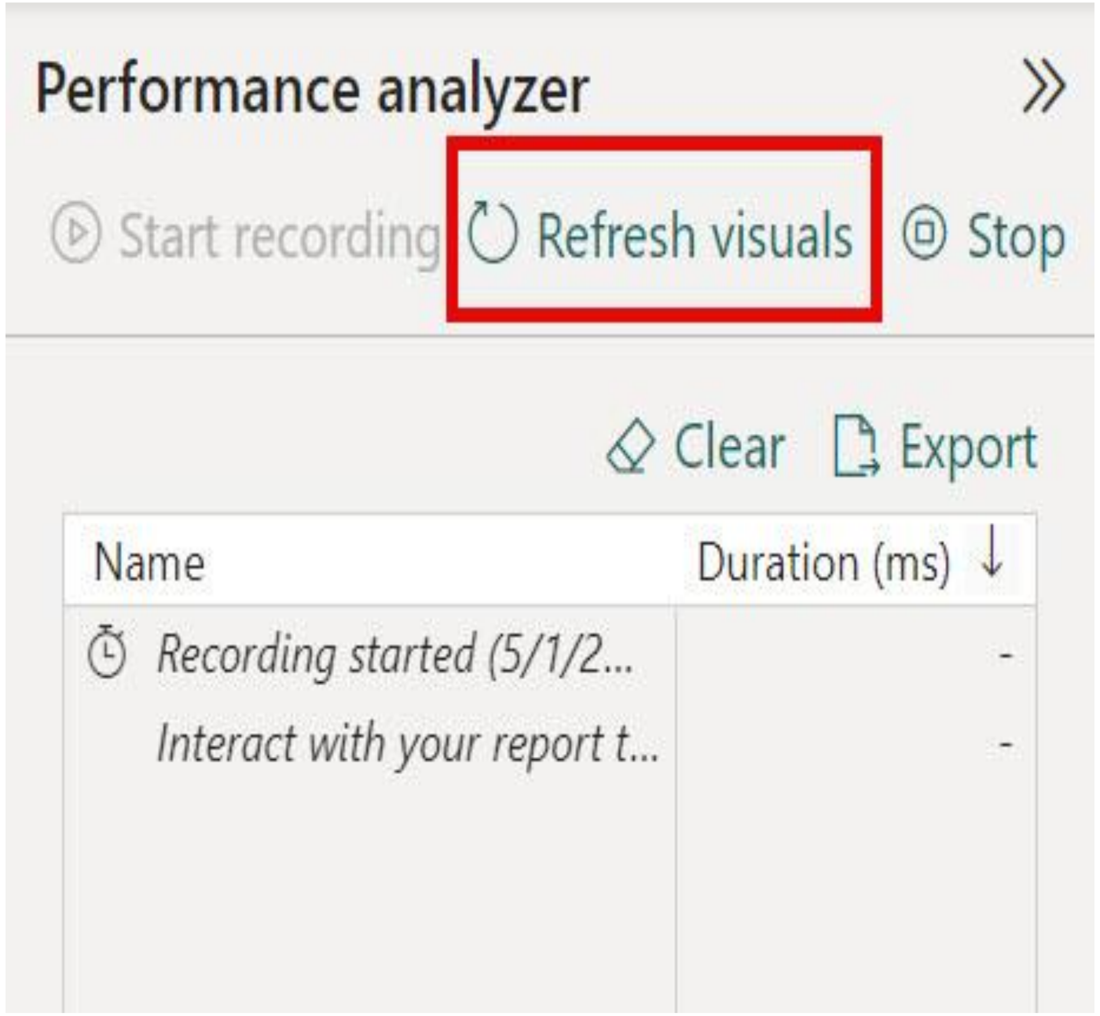
- Sales
- Customer
- Date
- Product
- Store

Performance Analyzer Recording



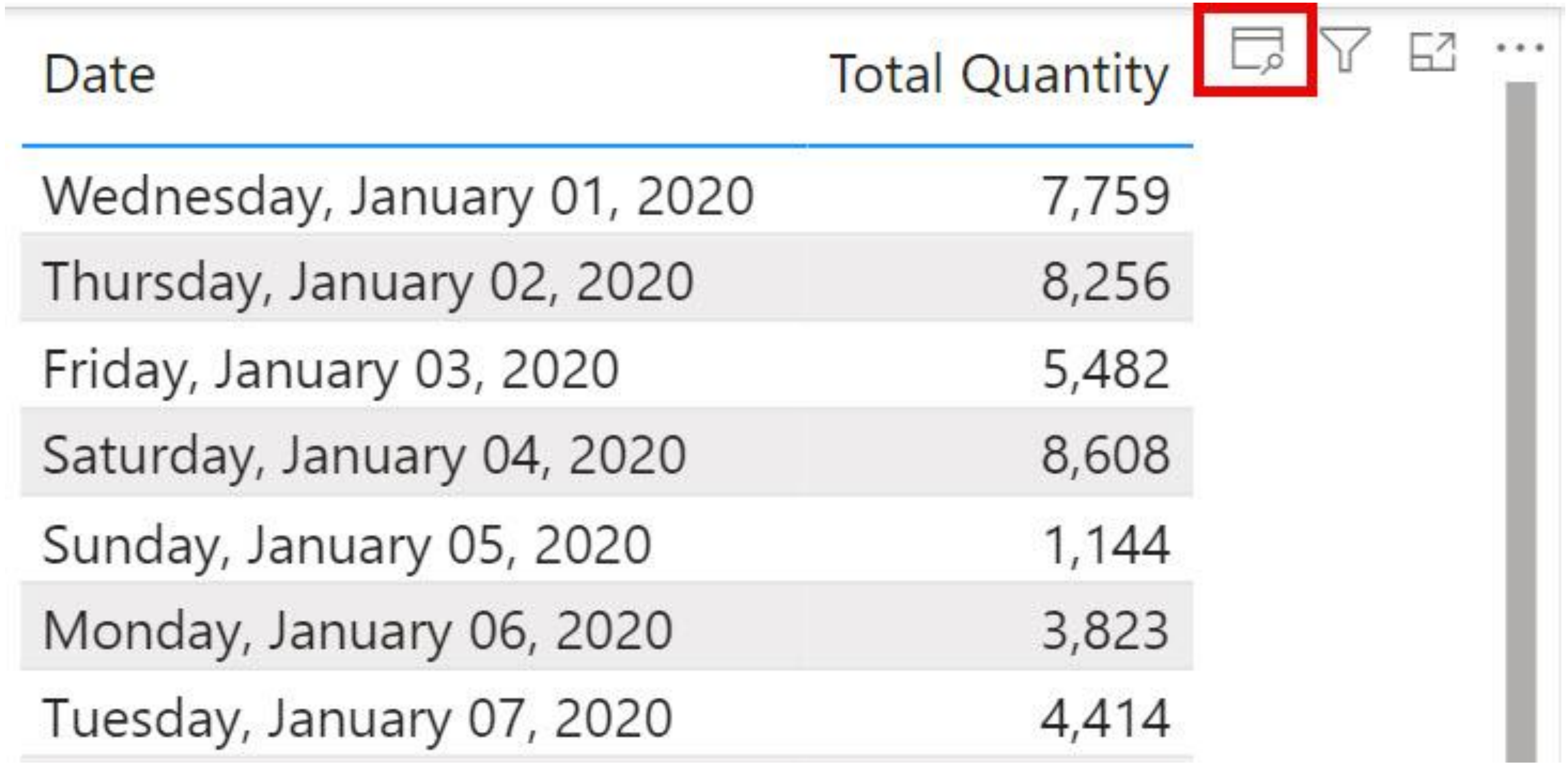
Name	Duration (ms)
Recording started (5/1/2...	-
Interact with your report t...	-

Performance Analyzer
Refresh Visuals



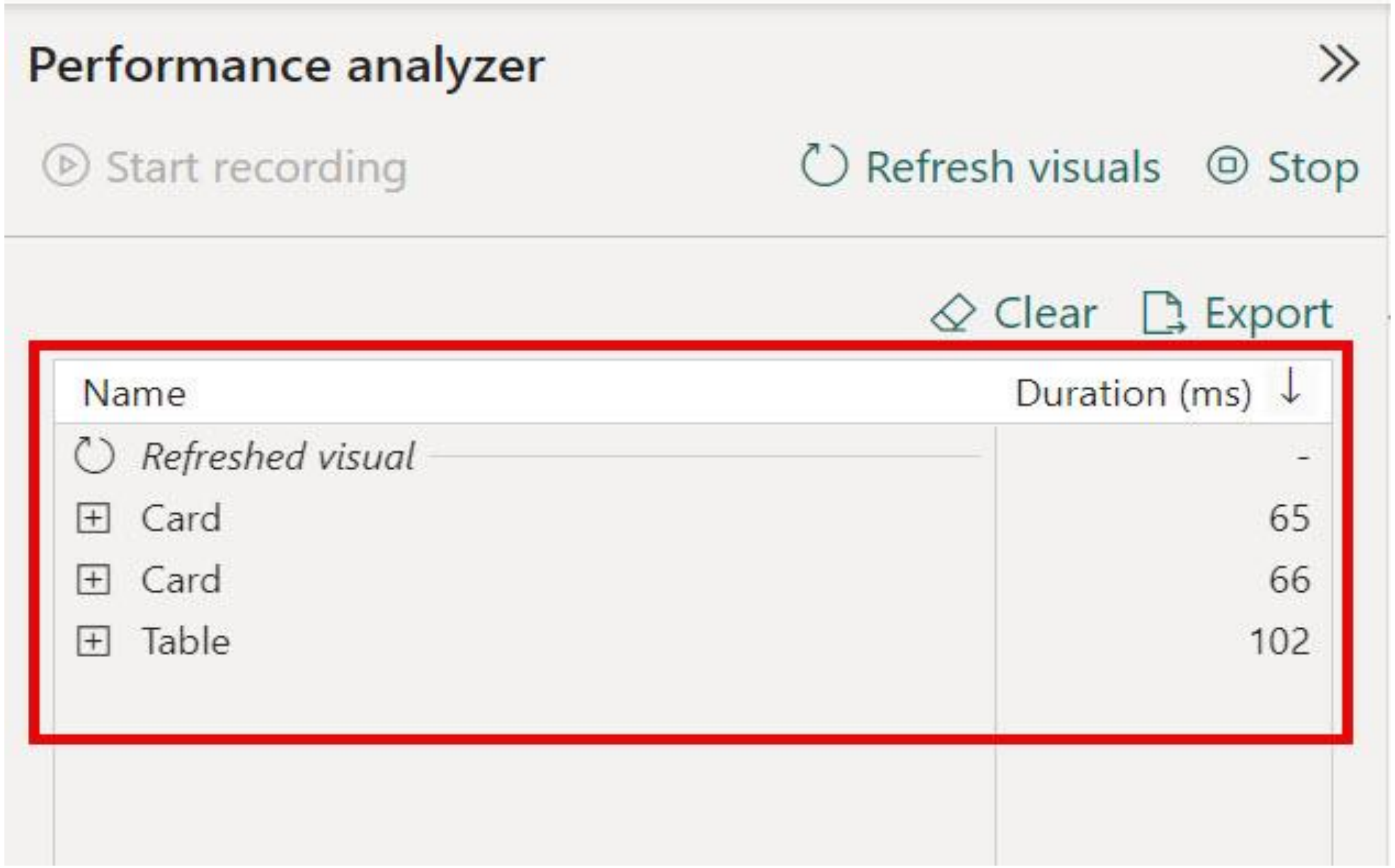
Name	Duration (ms)
Recording started (5/1/2...	-
Interact with your report t...	-

Individual Refresh Button



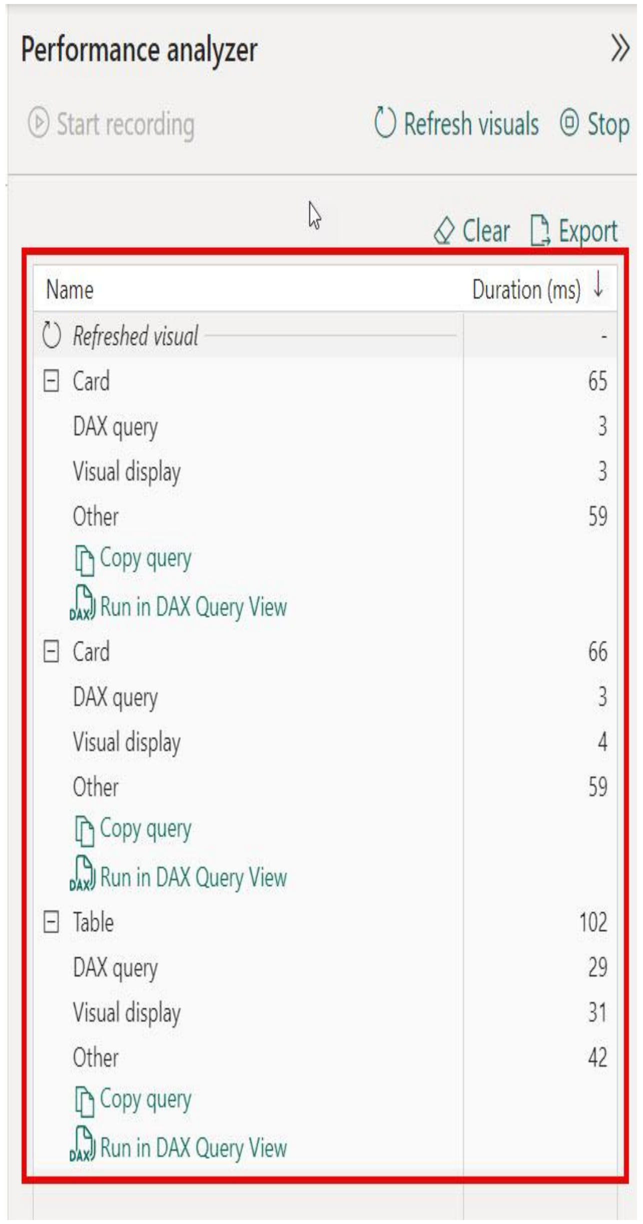
Date	Total Quantity
Wednesday, January 01, 2020	7,759
Thursday, January 02, 2020	8,256
Friday, January 03, 2020	5,482
Saturday, January 04, 2020	8,608
Sunday, January 05, 2020	1,144
Monday, January 06, 2020	3,823
Tuesday, January 07, 2020	4,414

List of Visuals



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

Expanded View



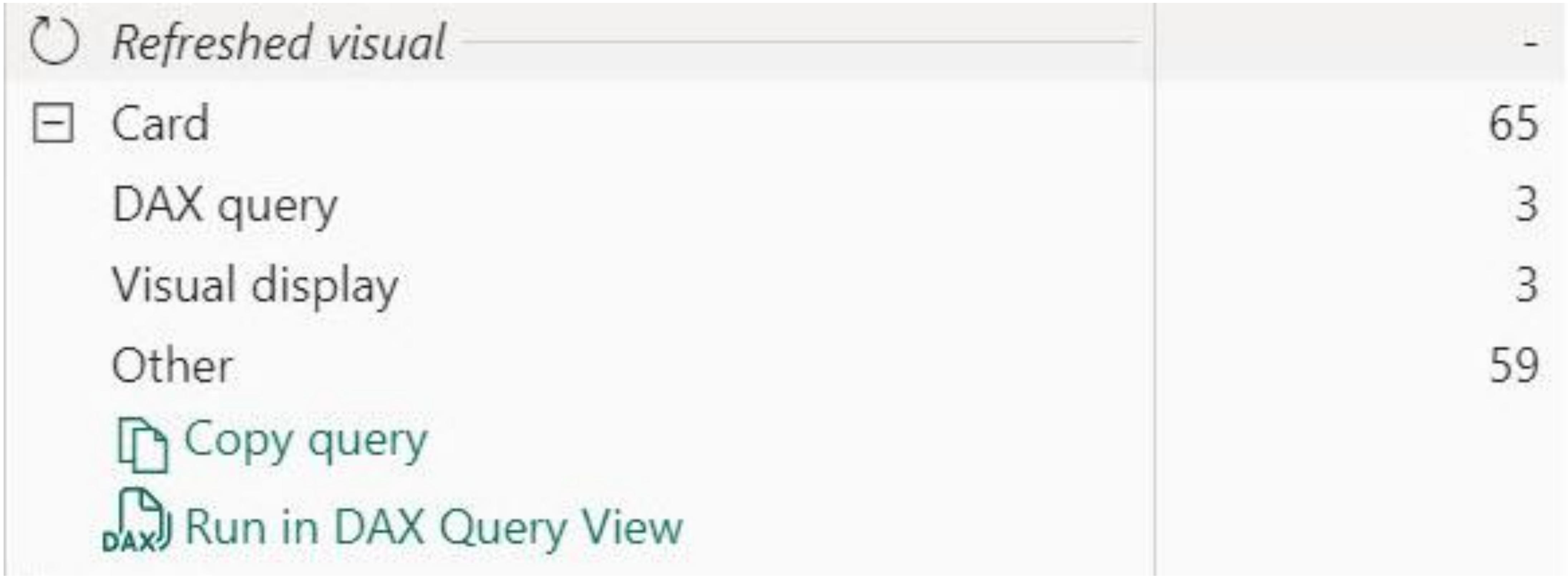
Performance analyzer

Start recording | Refresh visuals | Stop

Clear | Export

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

Detail on Card



Refreshed visual	-
Card	65
DAX query	3
Visual display	3
Other	59

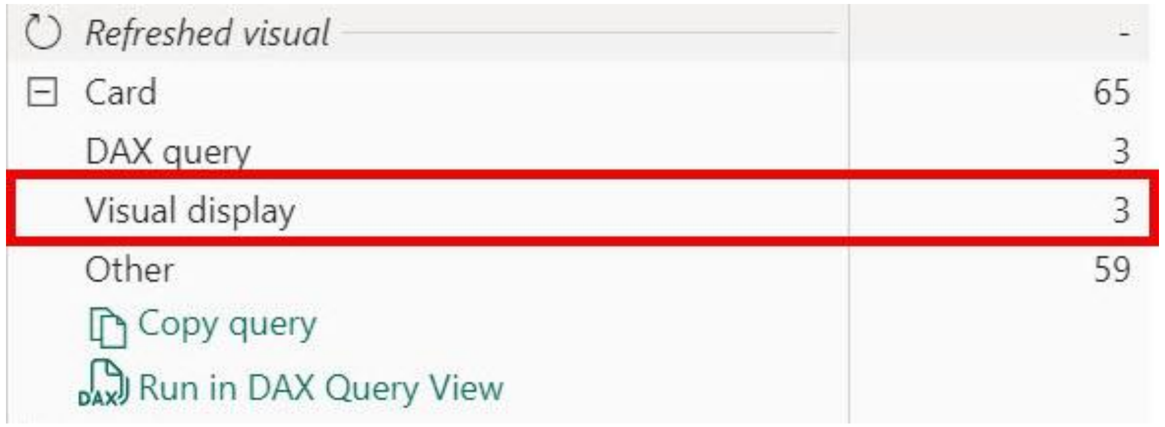
3 Numbers

Visual Display

Other

DAX Query

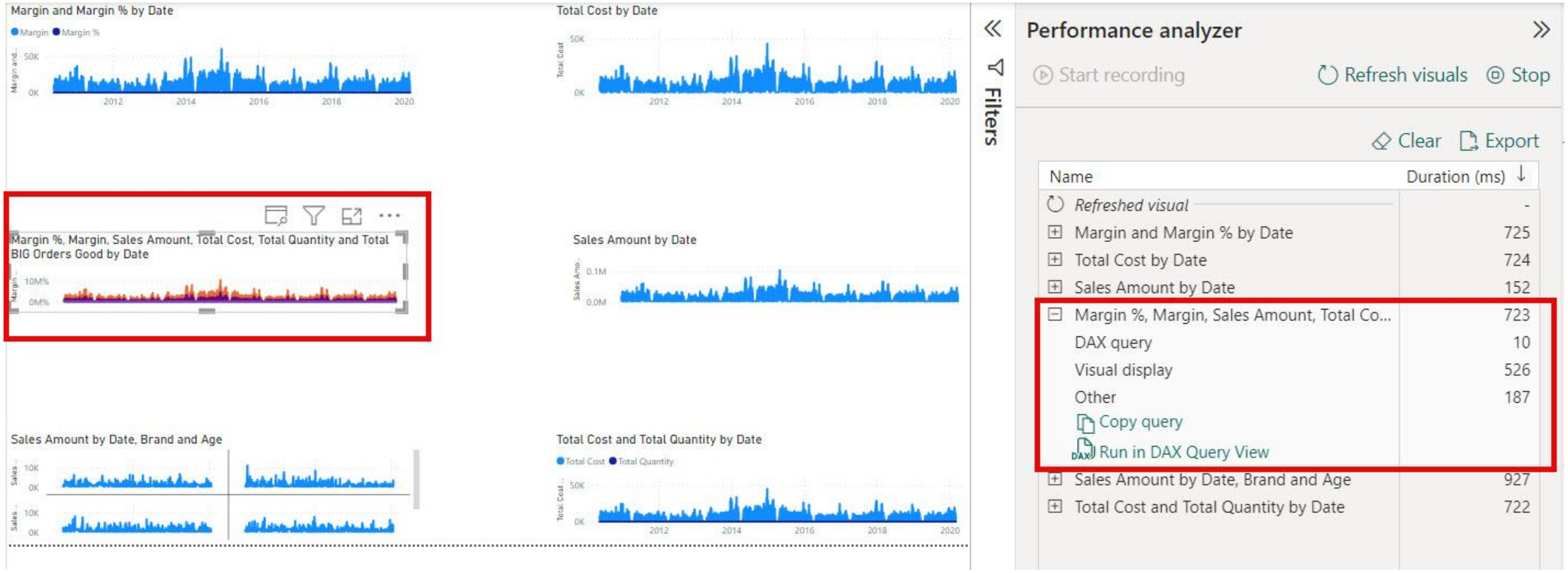
Visual Display



Refreshed visual	-
Card	65
DAX query	3
Visual display	3
Other	59

- Visual Display
  - Time spent on producing the visual

Slow Visual

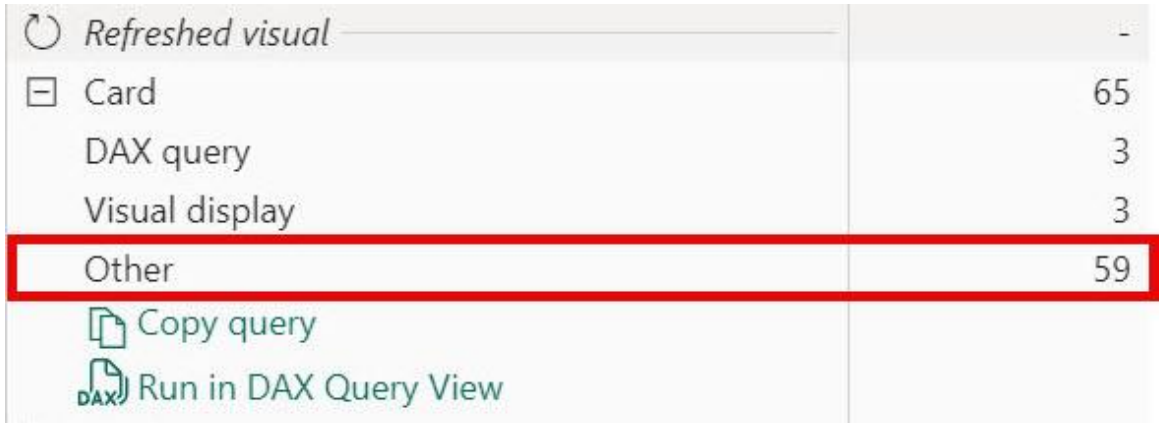

Margin and Margin % by Date
● Margin ● Margin %
Total Cost by Date
Margin %, Margin, Sales Amount, Total Cost, Total Quantity and Total BIG Orders Good by Date
Sales Amount by Date
Sales Amount by Date, Brand and Age
Total Cost and Total Quantity by Date
● Total Cost ● Total Quantity
Filters
Performance analyzer
Start recording
Refresh visuals
Stop
Clear
Export
Copy query
Run in DAX Query View

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...	723
DAX query	10
Visual display	526
Other	187
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

Other



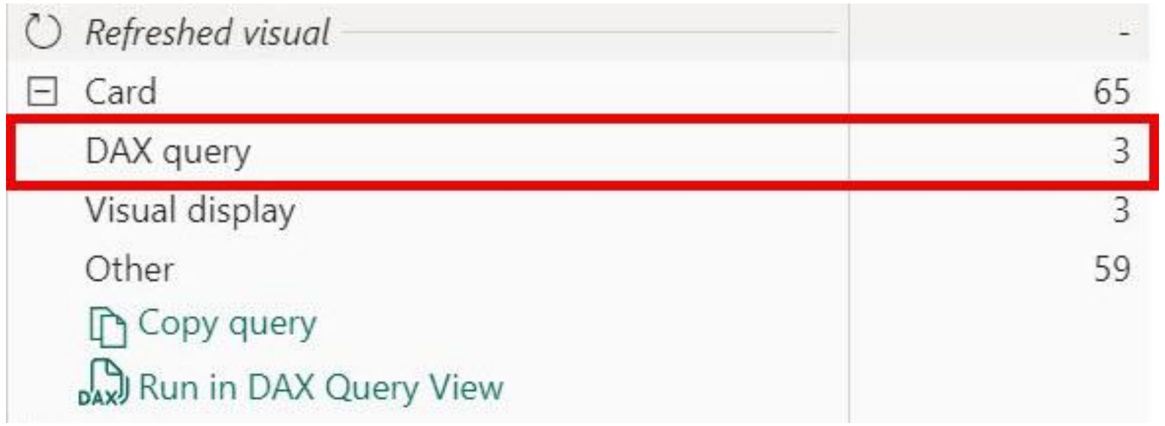
Refreshed visual	-
⊟ Card	65
DAX query	3
Visual display	3
Other	59

- Other
  - Time waiting until DAX Query can be executed

Other – Solutions

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

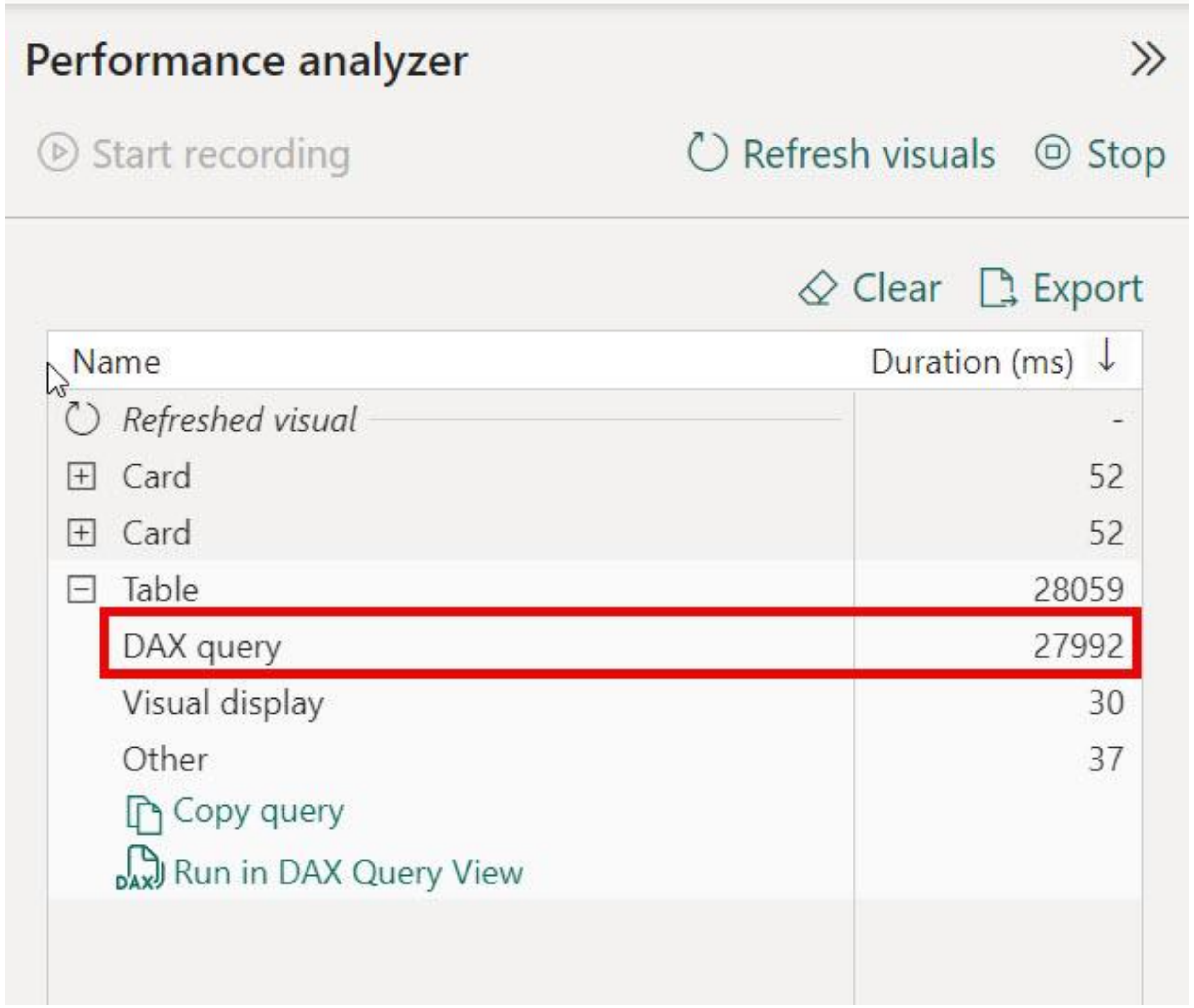
DAX Query



Refreshed visual	-
Card	65
DAX query	3
Visual display	3
Other	59

- DAX Query
  - Time it takes to execute the DAX query.

Slow DAX Query



Performance analyzer

Start recording | Refresh visuals | Stop

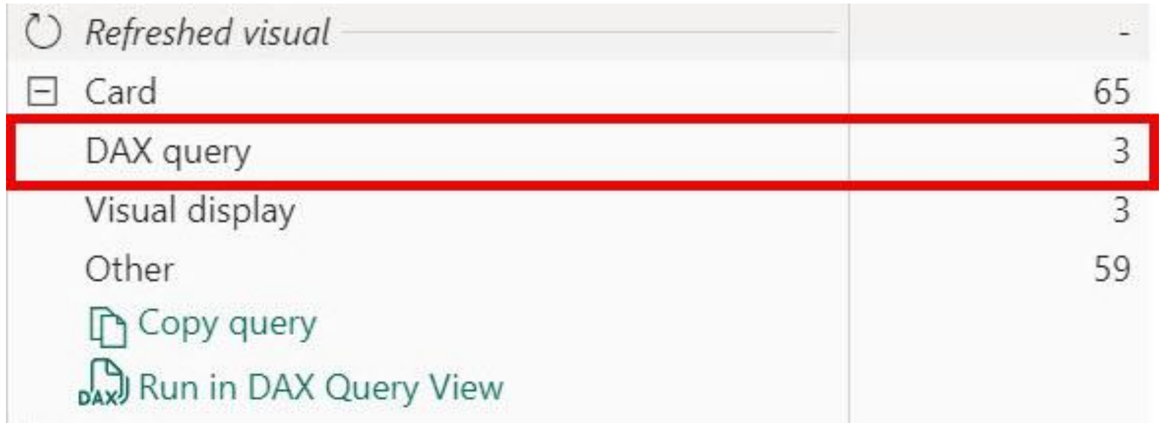
Clear | Export

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

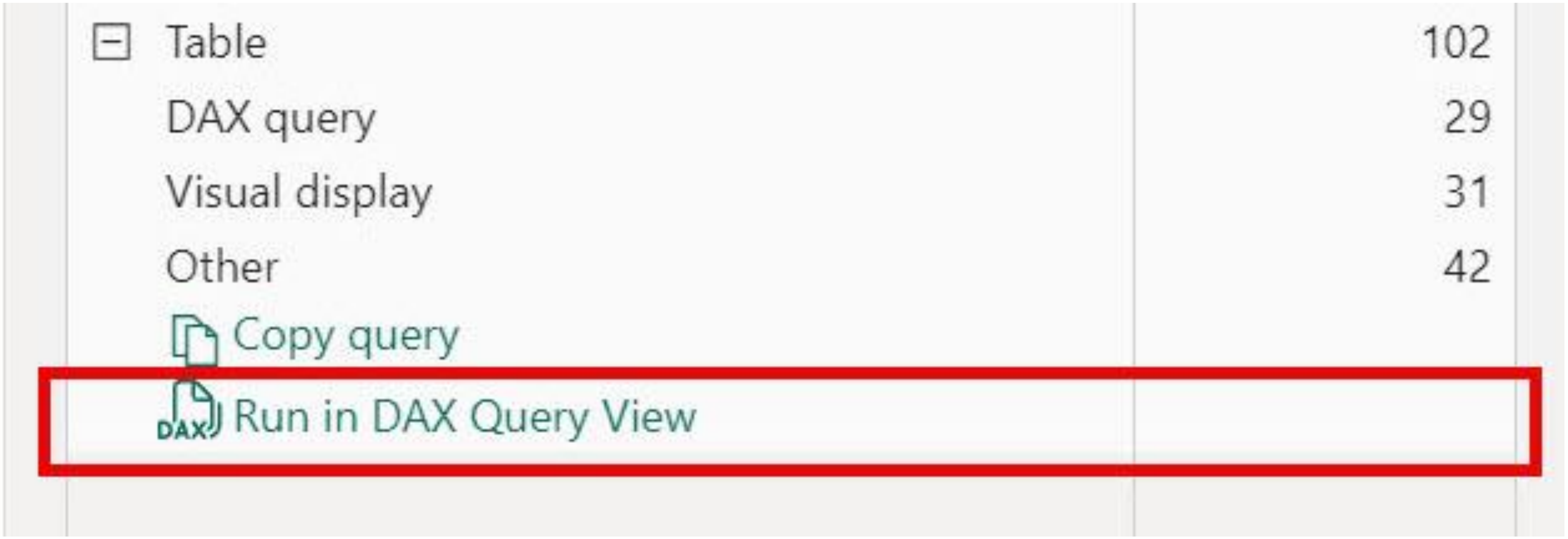
DAX Query – Solutions



Refreshed visual	-
Card	65
DAX query	3
Visual display	3
Other	59

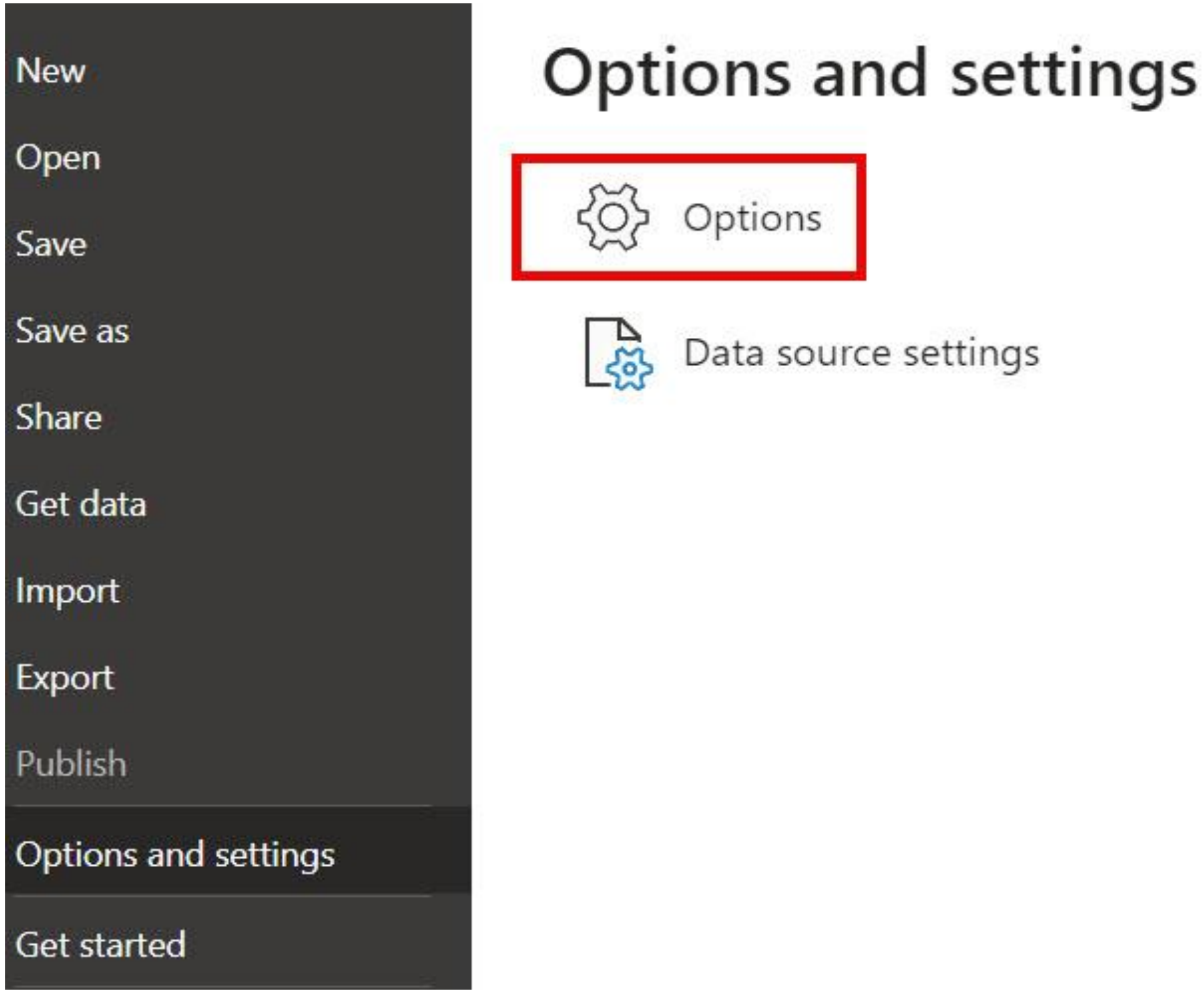
- Investigate query
  - Where is most of the query time spent
- Can the DAX code be optimized?
  - Rewrite the DAX code

Run in DAX Query View (GA)

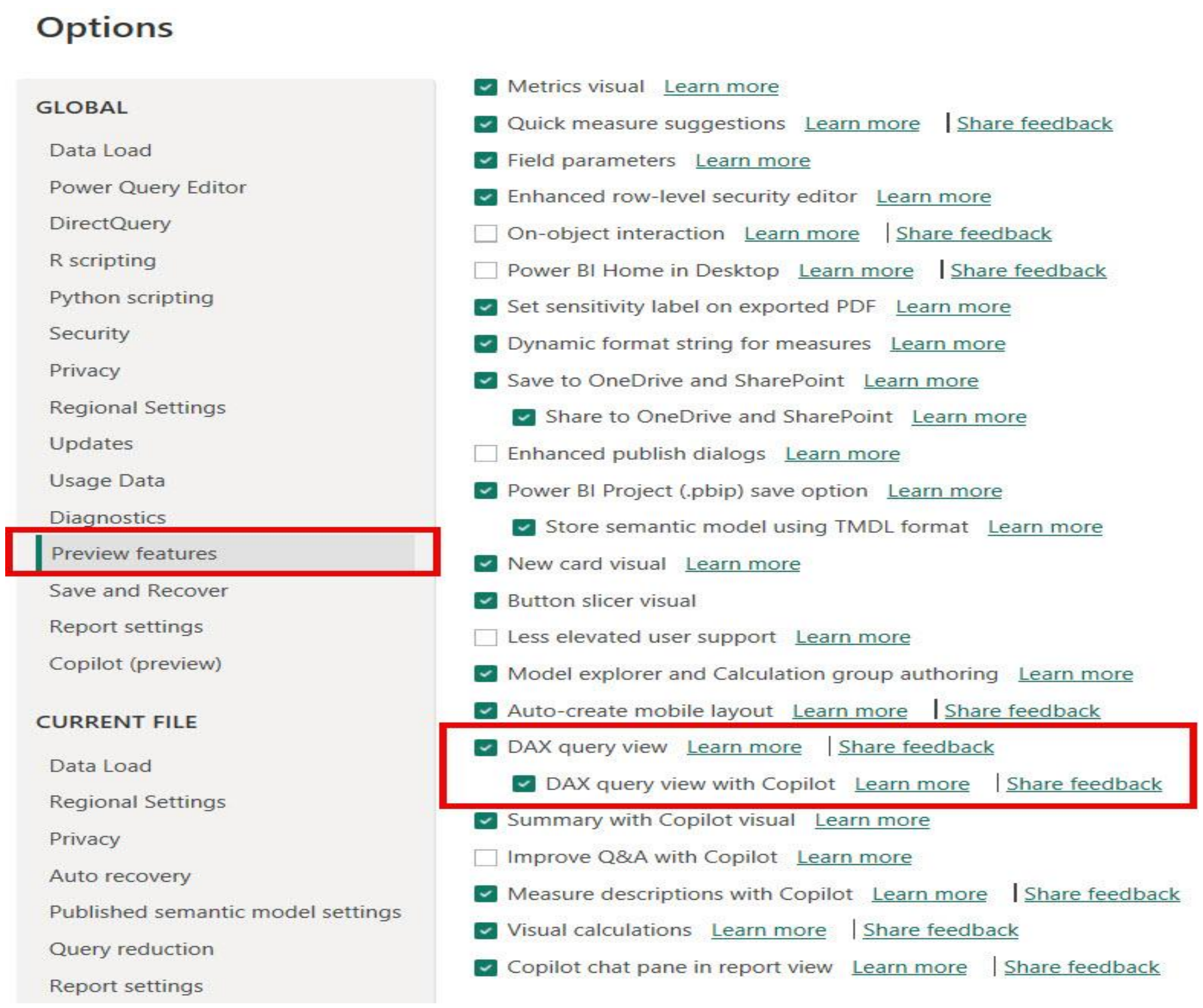


⊟ Table	102
DAX query	29
Visual display	31
Other	42

Enable DAX Query View (Before May 2024 Release)



DAX Query View – Preview Features



Options

GLOBAL

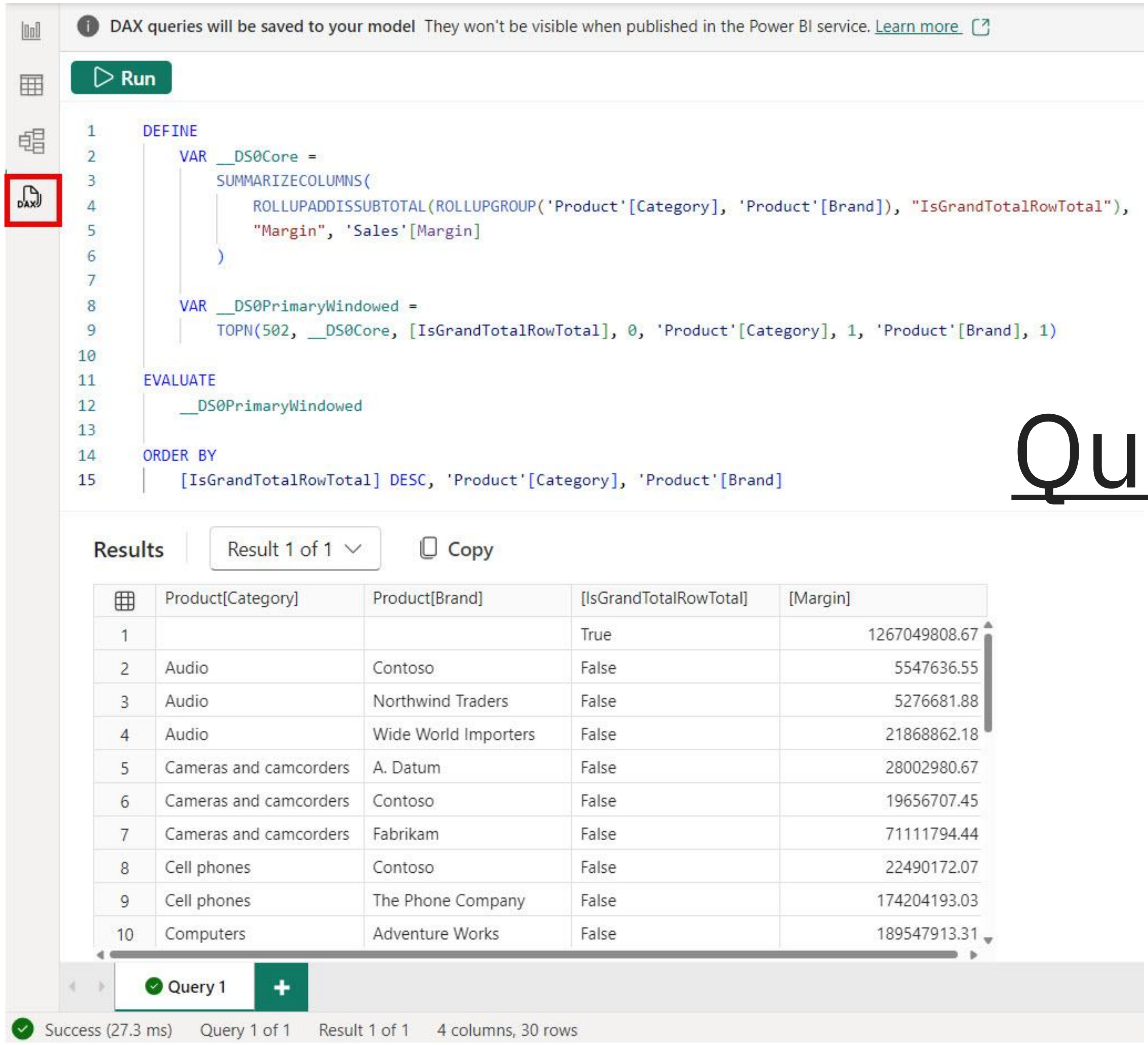
- Data Load
- Power Query Editor
- DirectQuery
- R scripting
- Python scripting
- Security
- Privacy
- Regional Settings
- Updates
- Usage Data
- Diagnostics
- Preview features
- Save and Recover
- Report settings
- Copilot (preview)

CURRENT FILE

- Data Load
- Regional Settings
- Privacy
- Auto recovery
- Published semantic model settings
- Query reduction
- Report settings

- [✓] Metrics visual Learn more
- [✓] Quick measure suggestions Learn more | Share feedback
- [✓] Field parameters Learn more
- [✓] Enhanced row-level security editor Learn more
- [ ] On-object interaction Learn more | Share feedback
- [ ] Power BI Home in Desktop Learn more | Share feedback
- [✓] Set sensitivity label on exported PDF Learn more
- [✓] Dynamic format string for measures Learn more
- [✓] Save to OneDrive and SharePoint Learn more
  - [✓] Share to OneDrive and SharePoint Learn more
- [ ] Enhanced publish dialogs Learn more
- [✓] Power BI Project (.pbip) save option Learn more
  - [✓] Store semantic model using TMDL format Learn more
- [✓] New card visual Learn more
- [✓] Button slicer visual
- [ ] Less elevated user support Learn more
- [✓] Model explorer and Calculation group authoring Learn more
- [✓] Auto-create mobile layout Learn more | Share feedback
- [✓] DAX query view Learn more | Share feedback
  - [✓] DAX query view with Copilot Learn more | Share feedback
- [✓] Summary with Copilot visual Learn more
- [ ] Improve Q&A with Copilot Learn more
- [✓] Measure descriptions with Copilot Learn more | Share feedback
- [✓] Visual calculations Learn more | Share feedback
- [✓] Copilot chat pane in report view Learn more | Share feedback

Query from Visual


DAX queries will be saved to your model They won't be visible when published in the Power BI service. Learn more
Run

```
DEFINE
    VAR __DS0Core =
        SUMMARIZECOLUMNS(
            ROLLUPADDISSUBTOTAL(ROLLUPGROUP('Product'[Category], 'Product'[Brand]), "IsGrandTotalRowTotal"),
            "Margin", 'Sales'[Margin]
        )

    VAR __DS0PrimaryWindowed =
        TOPN(502, __DS0Core, [IsGrandTotalRowTotal], 0, 'Product'[Category], 1, 'Product'[Brand], 1)

EVALUATE
    __DS0PrimaryWindowed

ORDER BY
    [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

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

Query View

- Queries are saved with the model
- Share or preserve slow query
- First exposure to DAX Queries
- Query View 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

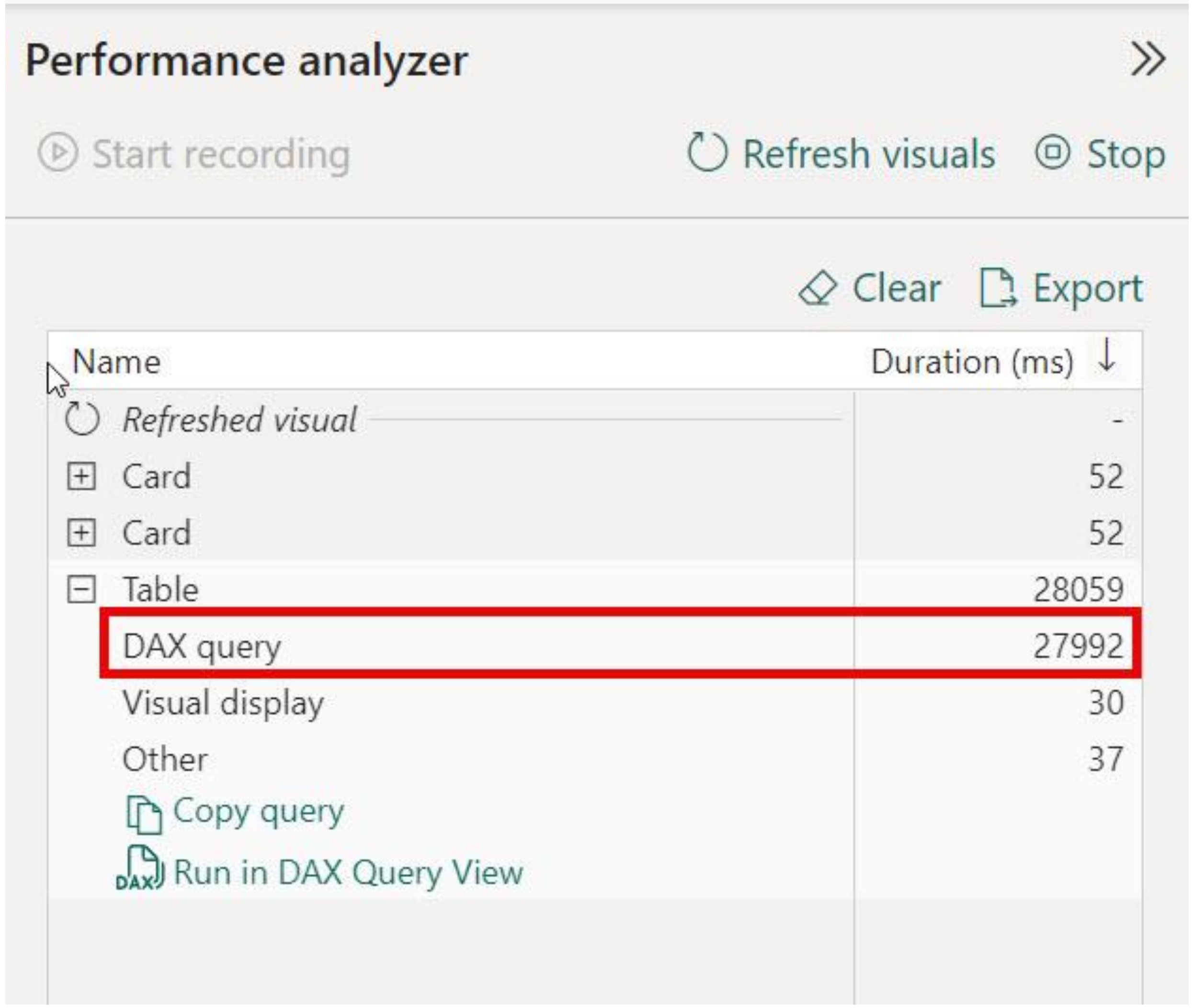
Power BI Demo
Performance Analyzer

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?



Name	Duration (ms) ↓
Refreshed visual	-
Card	52
Card	52
Table	28059
DAX query	27992
Visual display	30
Other	37

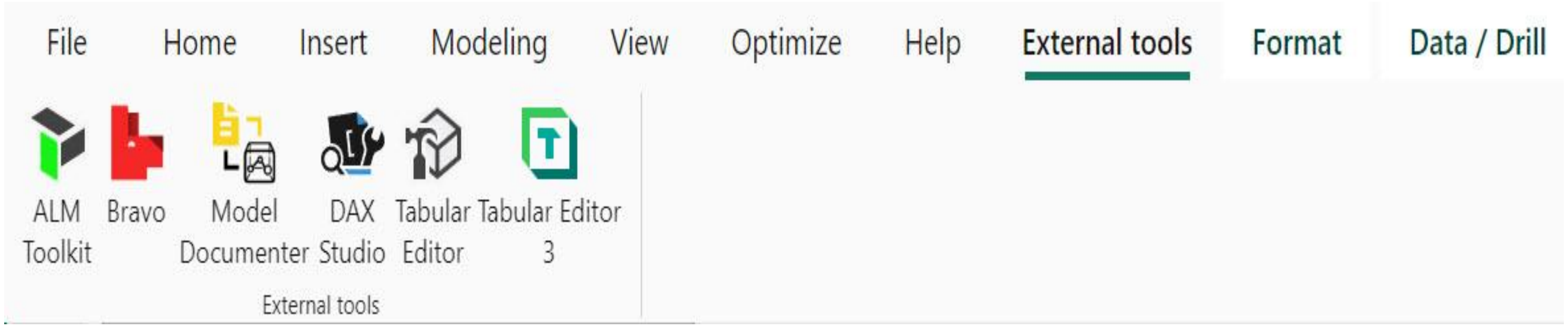
DAX

External Tools

Installation

- Full Install
- Power BI Desktop
  - External Tools Tab

External Tools Tab



Our Journey

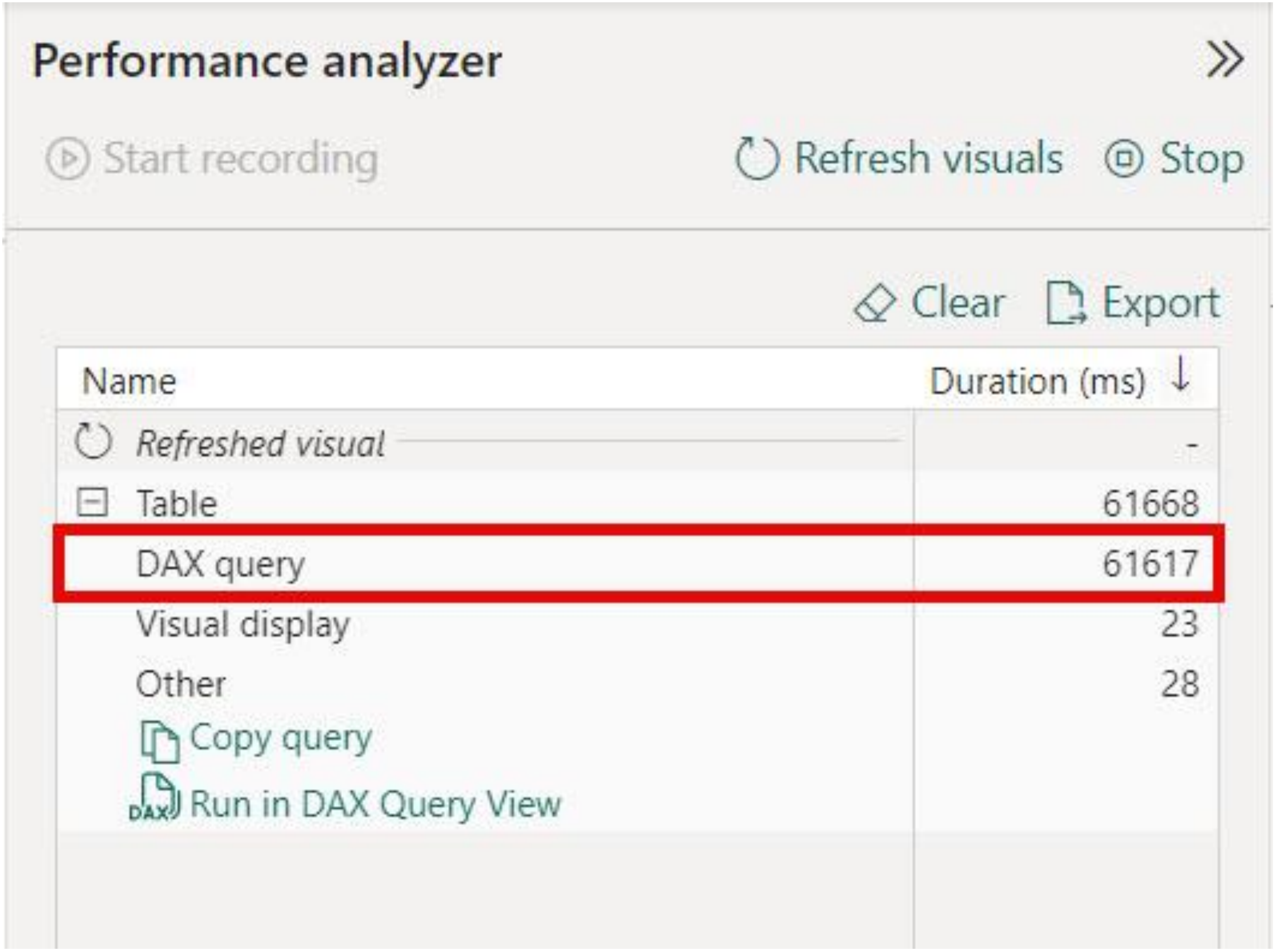
1. Intro
2. Performance Analyzer
3. **DAX Studio**
4. Tabular Editor
5. Conclusion

What if the DAX Query is Slow?

DAX Studio

- Performance Tuning
  - Where is the query spending the most time
    - Formula Engine
    - Storage Engine

Slow DAX Query



Performance analyzer

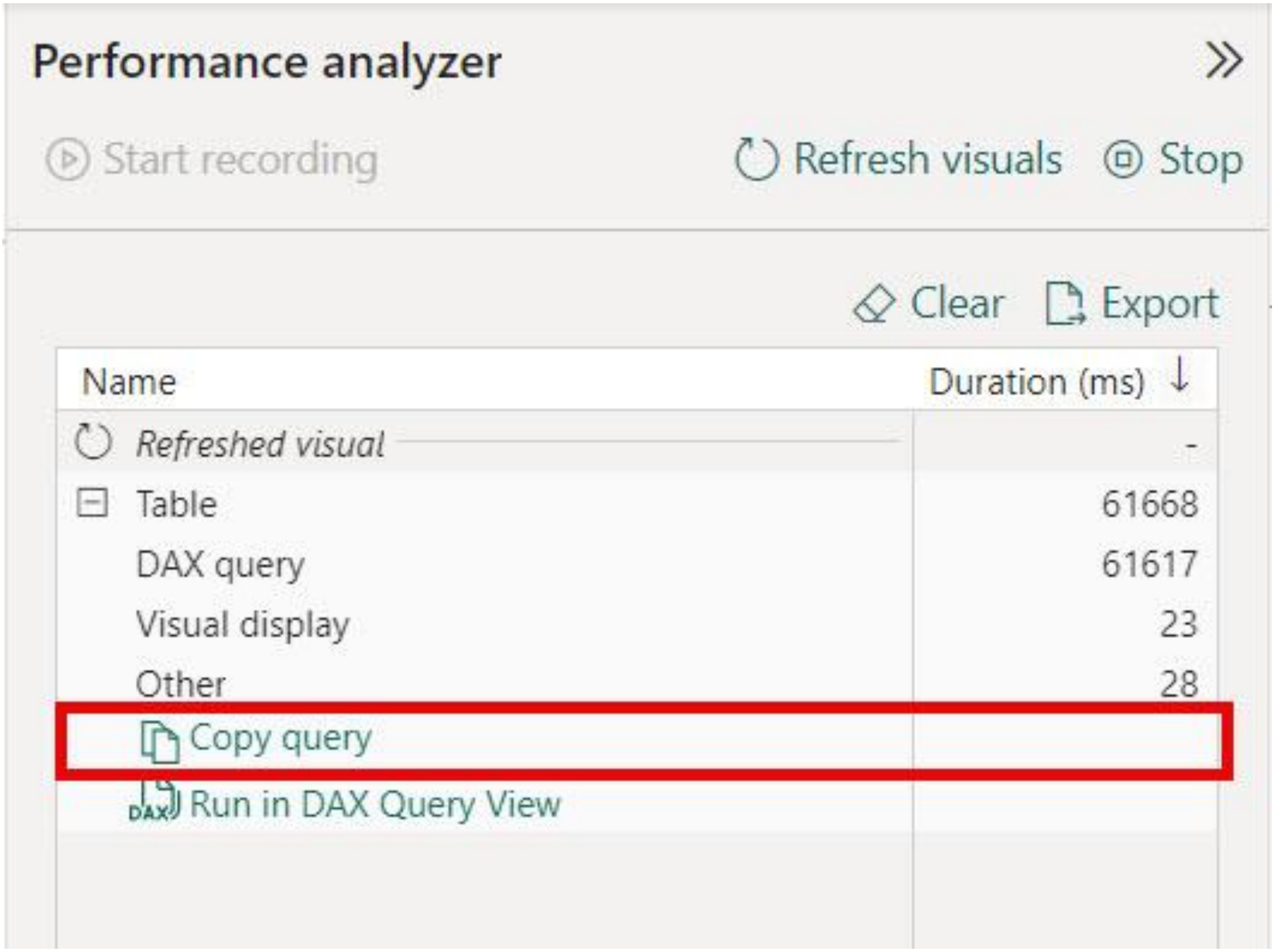
Start recording   Refresh visuals   Stop

Clear   Export

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

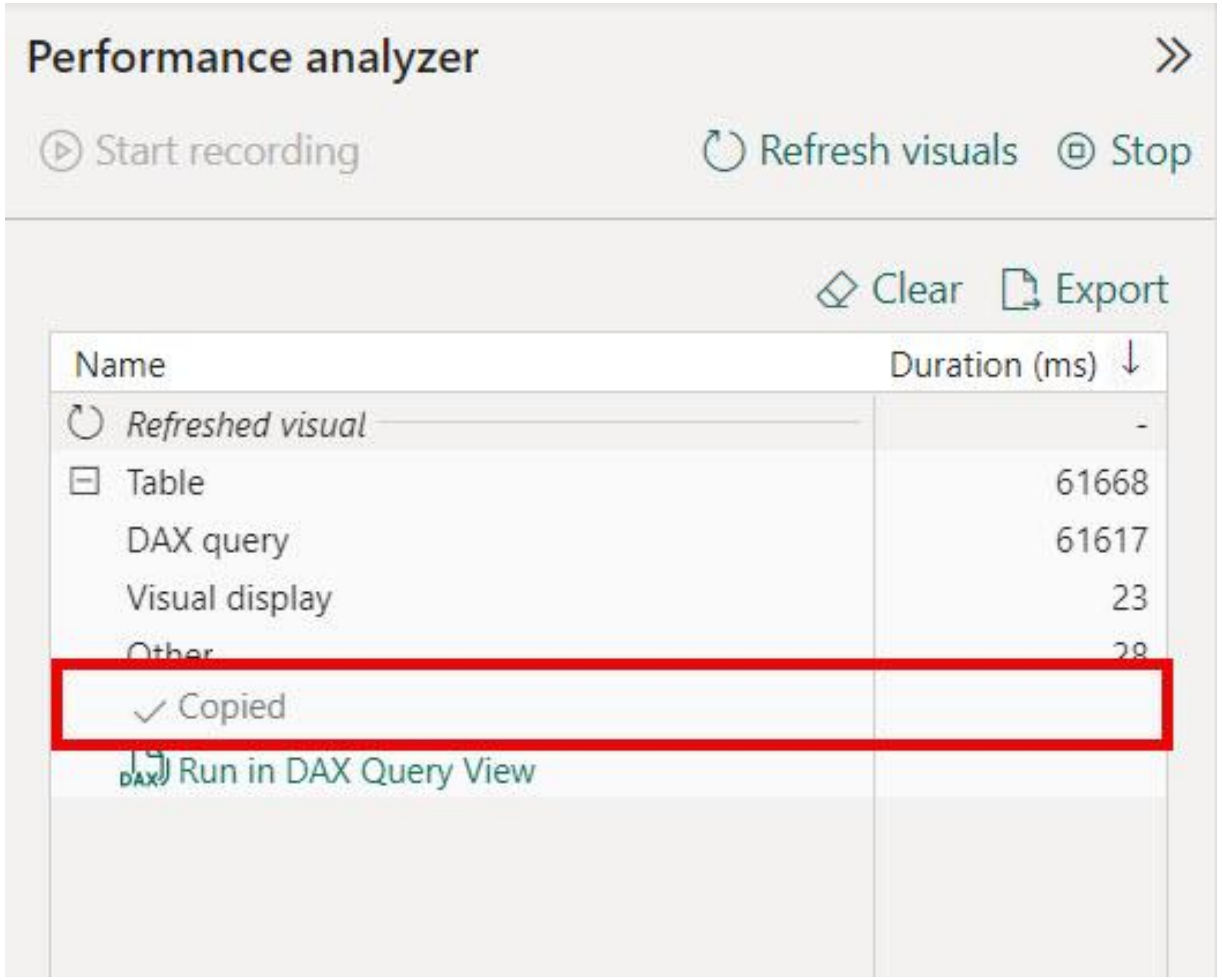
Copy query
Run in DAX Query View

Copy query



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

Copied



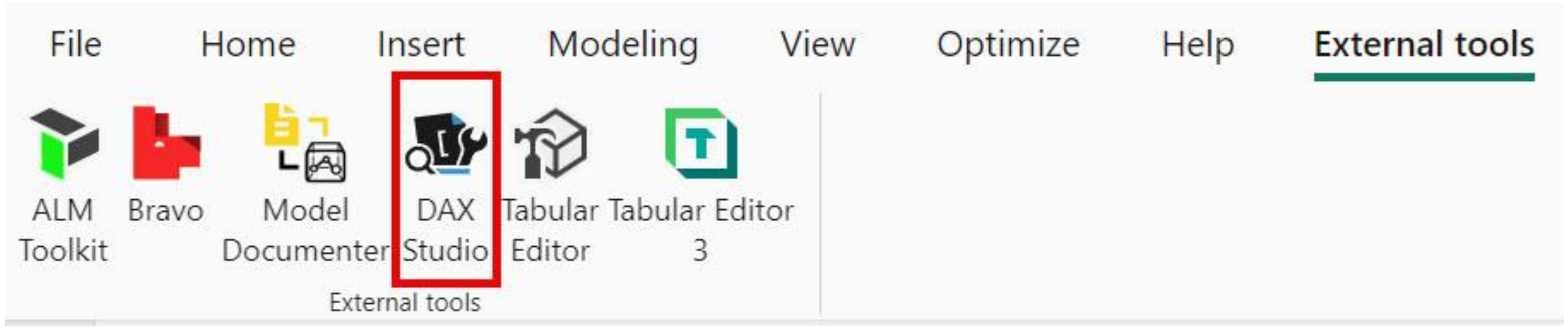
Performance analyzer

Start recording　Refresh visuals　Stop

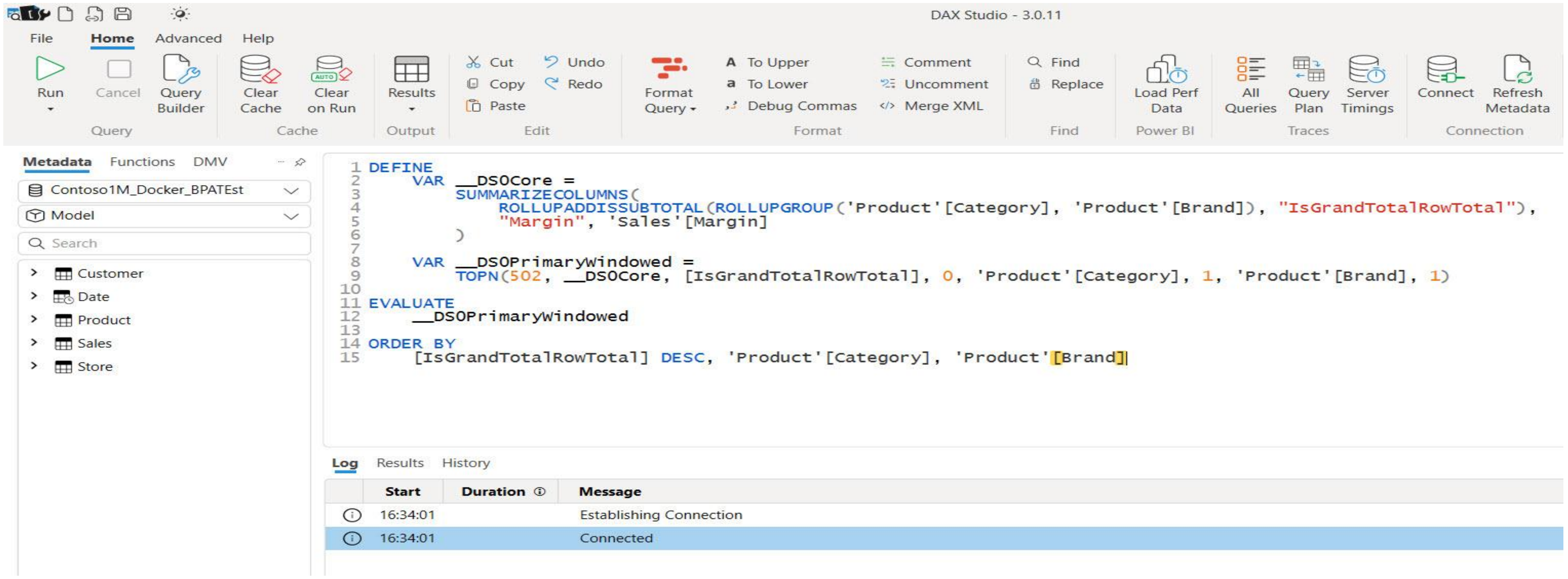
Clear　Export

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

Power BI – Open DAX Studio



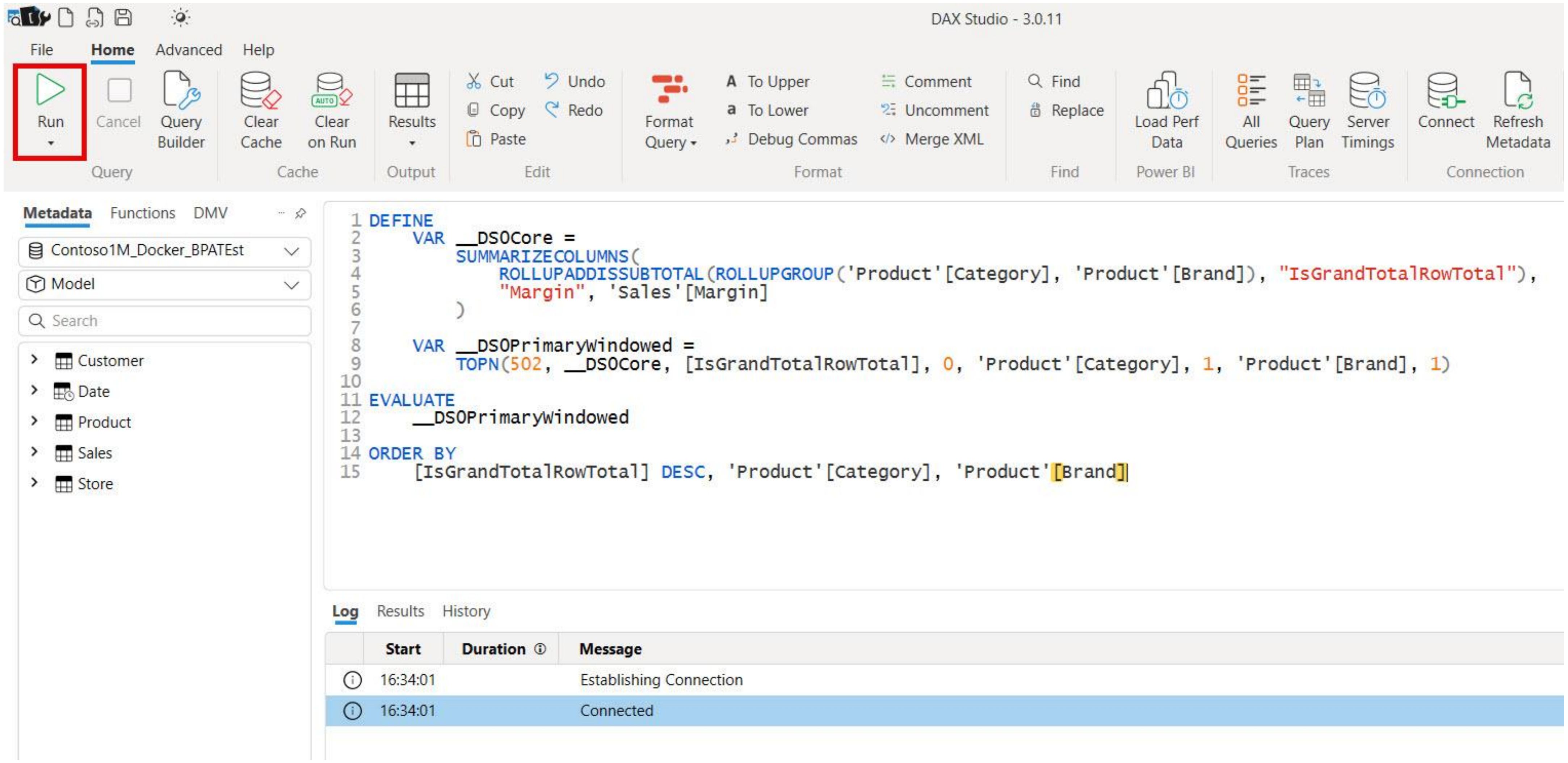
Paste in DAX Studio



```
DEFINE
    VAR __DSOCore =
        SUMMARIZECOLUMNS(
            ROLLUPADDISSUBTOTAL(ROLLUPGROUP('Product'[Category], 'Product'[Brand]), "IsGrandTotalRowTotal"),
            "Margin", 'Sales'[Margin]
        )

    VAR __DSOPrimaryWindowed =
        TOPN(502, __DSOCore, [IsGrandTotalRowTotal], 0, 'Product'[Category], 1, 'Product'[Brand], 1)

EVALUATE
    __DSOPrimaryWindowed

ORDER BY
    [IsGrandTotalRowTotal] DESC, 'Product'[Category], 'Product'[Brand]
```

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

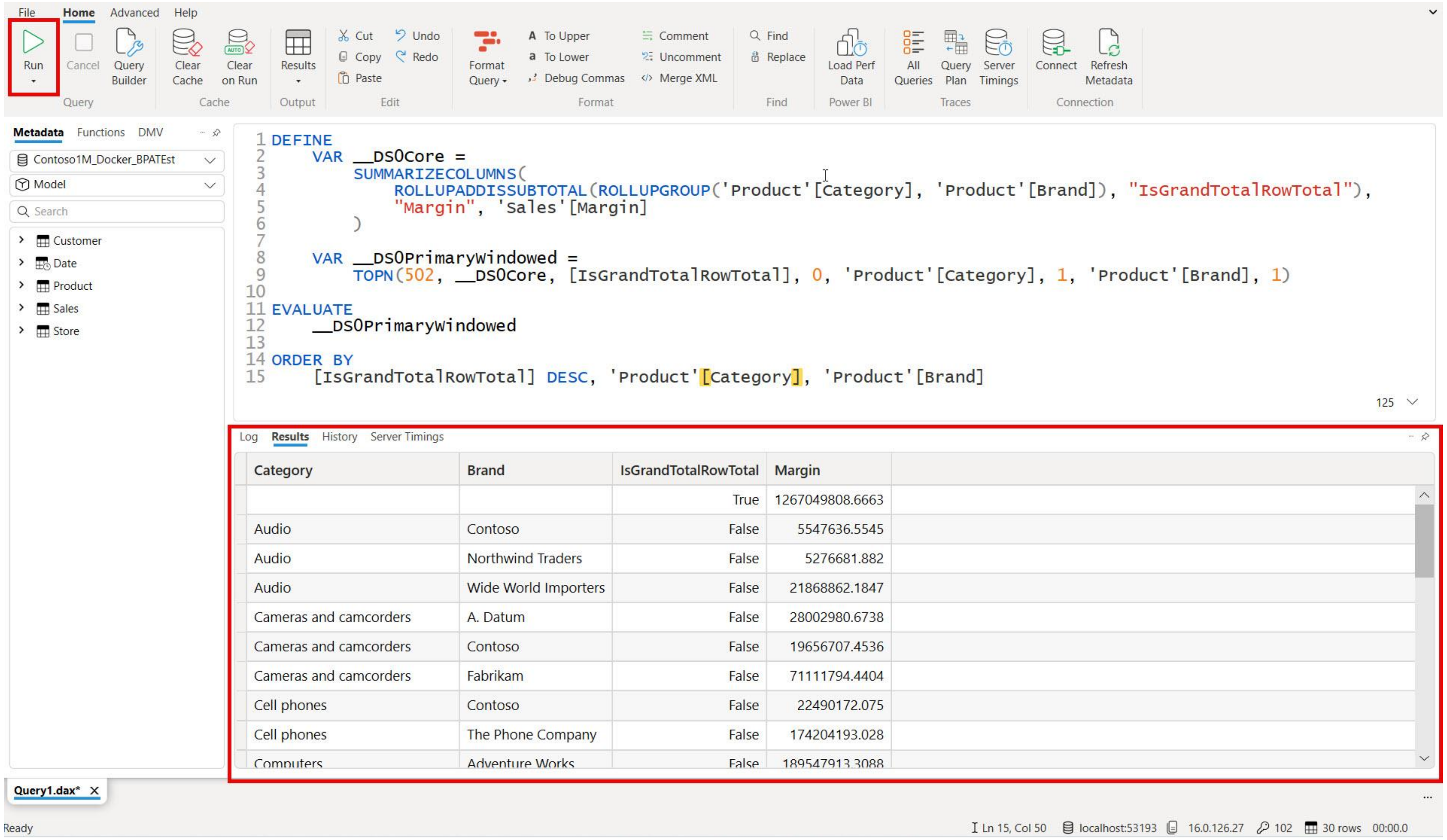
Let's Run It



```
DEFINE
    VAR __DS0Core =
        SUMMARIZECOLUMNS(
            ROLLUPADDISSUBTOTAL(ROLLUPGROUP('Product'[Category], 'Product'[Brand]), "IsGrandTotalRowTotal"),
            "Margin", 'Sales'[Margin]
        )

    VAR __DS0PrimaryWindowed =
        TOPN(502, __DS0Core, [IsGrandTotalRowTotal], 0, 'Product'[Category], 1, 'Product'[Brand], 1)

EVALUATE
    __DS0PrimaryWindowed

ORDER BY
    [IsGrandTotalRowTotal] DESC, 'Product'[Category], 'Product'[Brand]
```

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

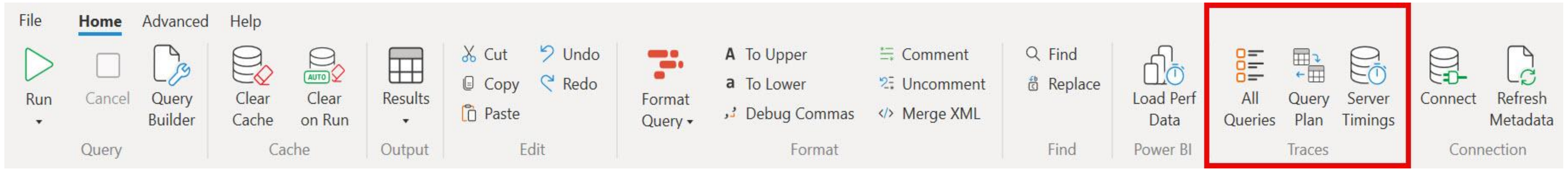
Results – Like Query View



```
DEFINE
    VAR __DS0Core =
        SUMMARIZECOLUMNS(
            ROLLUPADDISSUBTOTAL(ROLLUPGROUP('Product'[Category], 'Product'[Brand]), "IsGrandTotalRowTotal"),
            "Margin", 'Sales'[Margin]
        )

    VAR __DS0PrimaryWindowed =
        TOPN(502, __DS0Core, [IsGrandTotalRowTotal], 0, 'Product'[Category], 1, 'Product'[Brand], 1)

EVALUATE
    __DS0PrimaryWindowed

ORDER BY
    [IsGrandTotalRowTotal] DESC, 'Product'[Category], 'Product'[Brand]
```

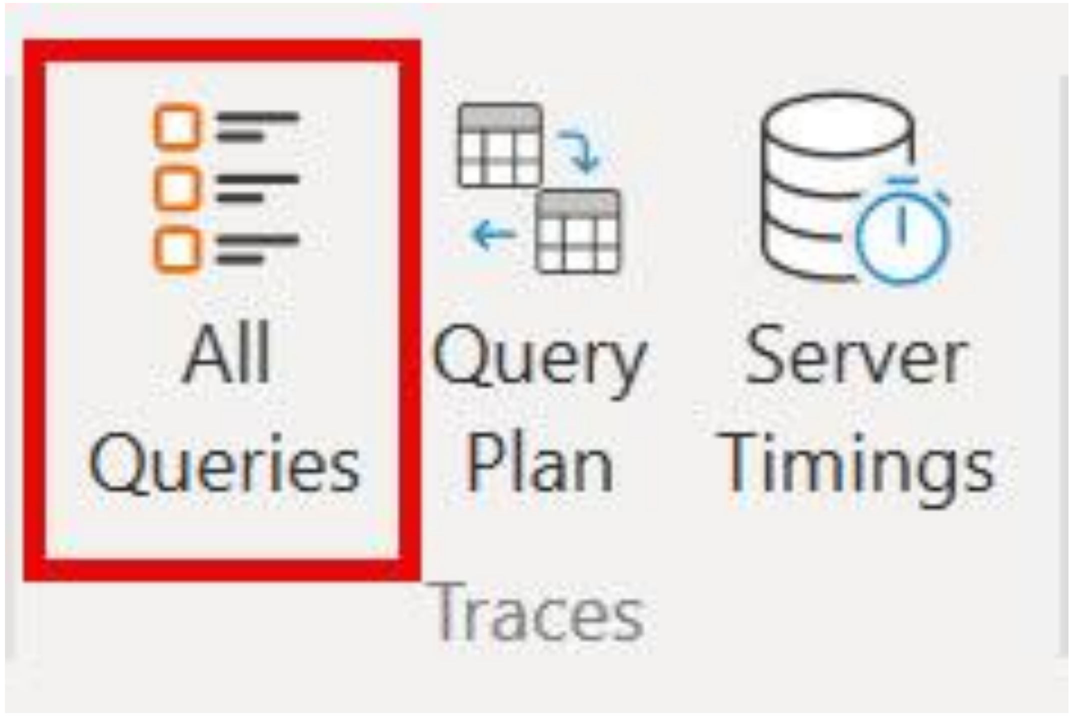
Category	Brand	IsGrandTotalRowTotal	Margin
		True	1267049808.6663
Audio	Contoso	False	5547636.5545
Audio	Northwind Traders	False	5276681.882
Audio	Wide World Importers	False	21868862.1847
Cameras and camcorders	A. Datum	False	28002980.6738
Cameras and camcorders	Contoso	False	19656707.4536
Cameras and 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

Traces



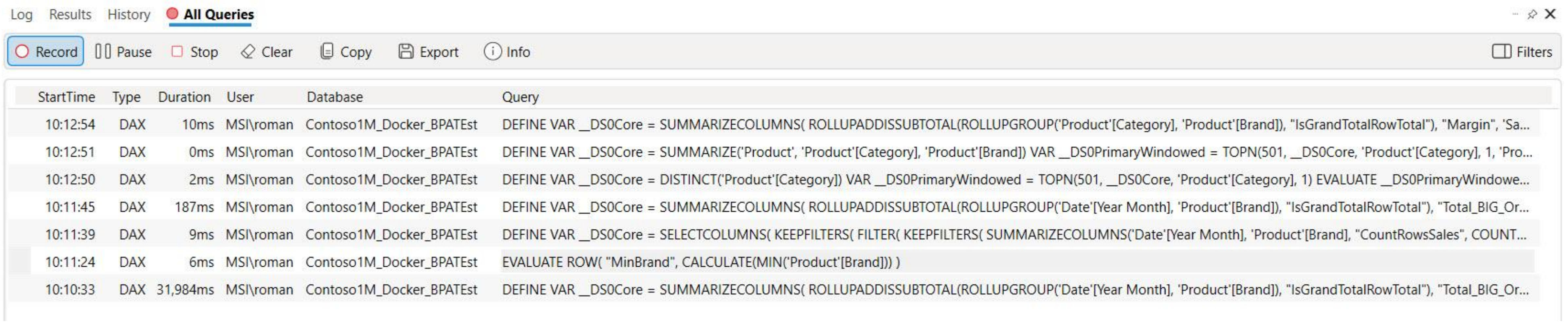
File
Home
Advanced
Help
Run
Cancel
Query Builder
Query
Clear Cache
Clear on Run
Cache
Results
Output
Cut
Copy
Paste
Undo
Redo
Edit
Format Query
To Upper
To Lower
Debug Commas
Comment
Uncomment
Merge XML
Format
Find
Replace
Find
Load Perf Data
Power BI
All Queries
Query Plan
Server Timings
Traces
Connect
Refresh Metadata
Connection

DAX Studio – All Queries



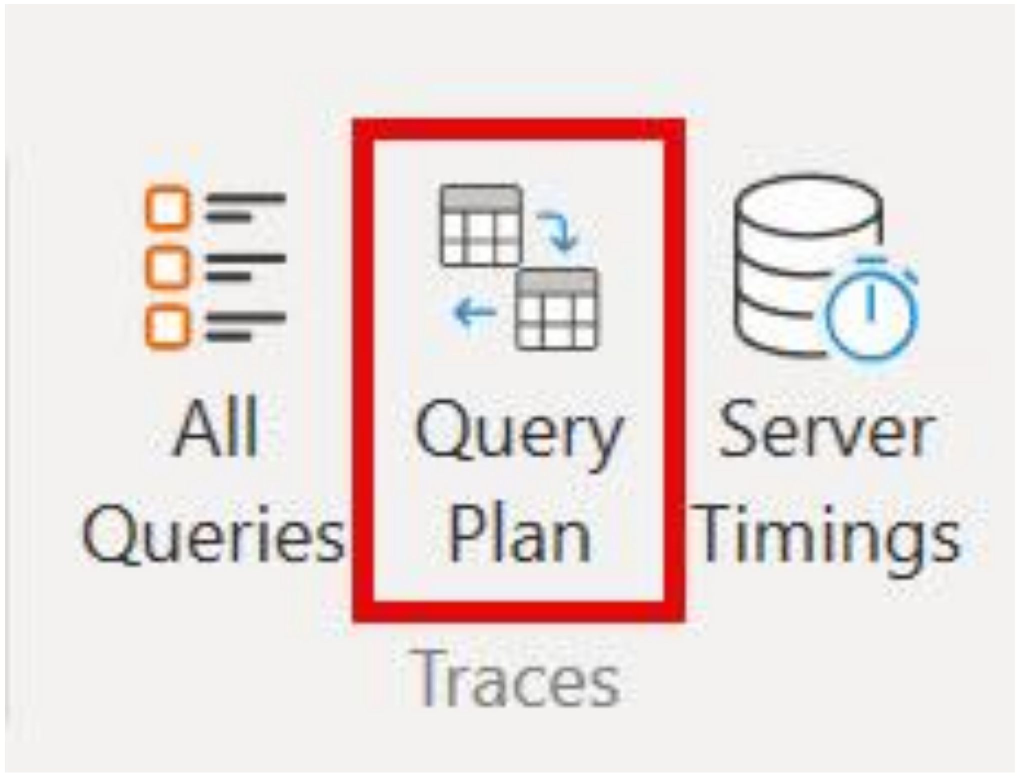
Like SQL Server Profiler

DAX Studio – All Queries



StartTime	Type	Duration	User	Database	Query
10:12:54	DAX	10ms	MSI\roman	Contoso1M_Docker_BPATEst	DEFINE VAR __DS0Core = SUMMARIZECOLUMNS(ROLLUPADDISSUBTOTAL(ROLLUPGROUP('Product'[Category], 'Product'[Brand]), "IsGrandTotalRowTotal"), "Margin", 'Sa...
10:12:51	DAX	0ms	MSI\roman	Contoso1M_Docker_BPATEst	DEFINE VAR __DS0Core = SUMMARIZE('Product', 'Product'[Category], 'Product'[Brand]) VAR __DS0PrimaryWindowed = TOPN(501, __DS0Core, 'Product'[Category], 1, 'Pro...
10:12:50	DAX	2ms	MSI\roman	Contoso1M_Docker_BPATEst	DEFINE VAR __DS0Core = DISTINCT('Product'[Category]) VAR __DS0PrimaryWindowed = TOPN(501, __DS0Core, 'Product'[Category], 1) EVALUATE __DS0PrimaryWindowe...
10:11:45	DAX	187ms	MSI\roman	Contoso1M_Docker_BPATEst	DEFINE VAR __DS0Core = SUMMARIZECOLUMNS(ROLLUPADDISSUBTOTAL(ROLLUPGROUP('Date'[Year Month], 'Product'[Brand]), "IsGrandTotalRowTotal"), "Total_BIG_Or...
10:11:39	DAX	9ms	MSI\roman	Contoso1M_Docker_BPATEst	DEFINE VAR __DS0Core = 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 __DS0Core = SUMMARIZECOLUMNS(ROLLUPADDISSUBTOTAL(ROLLUPGROUP('Date'[Year Month], 'Product'[Brand]), "IsGrandTotalRowTotal"), "Total_BIG_Or...

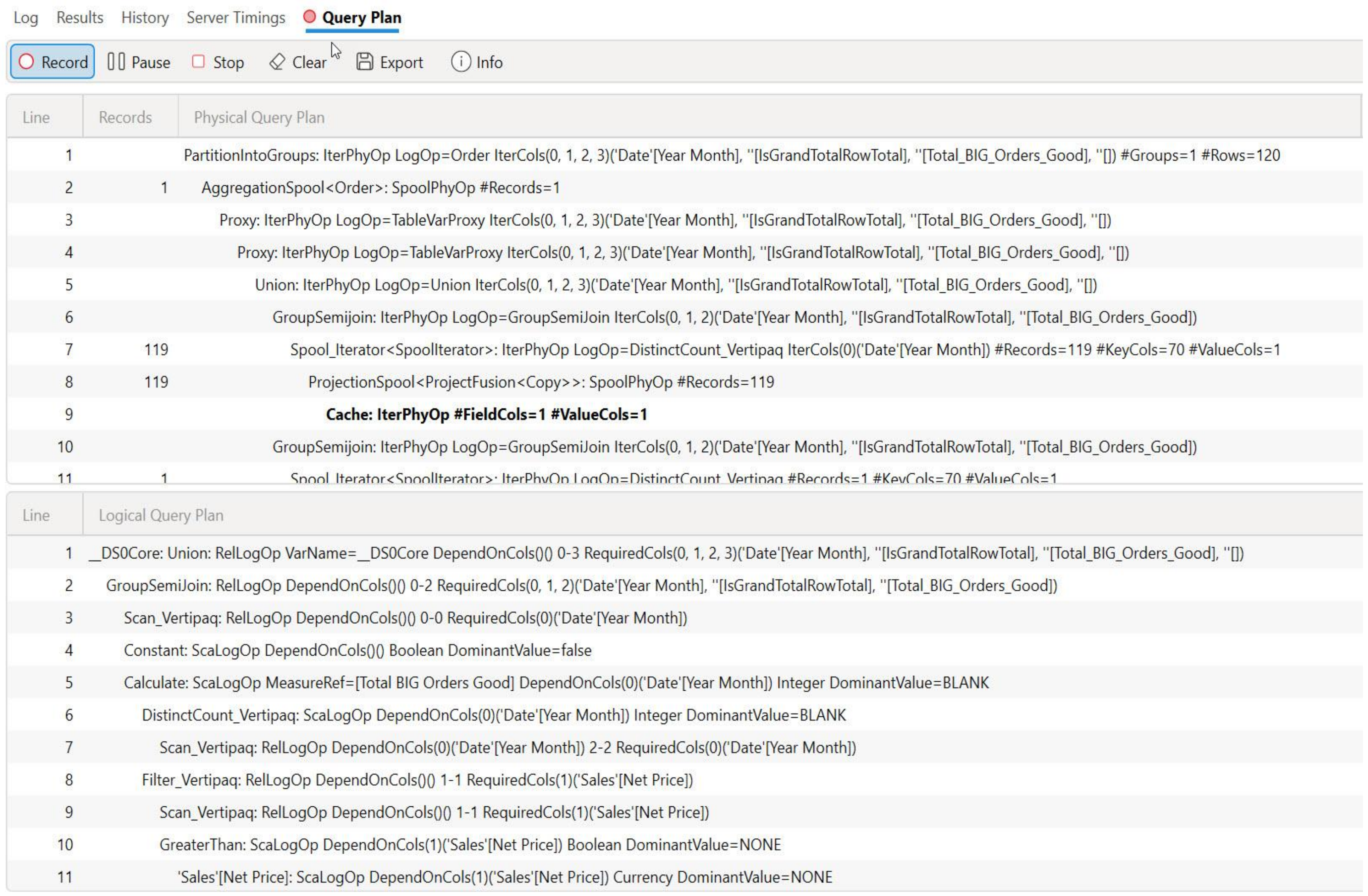
Query Plan



DAX Studio

- Query Plans
  - Logical
  - Physical

DAX Studio – Query Plan

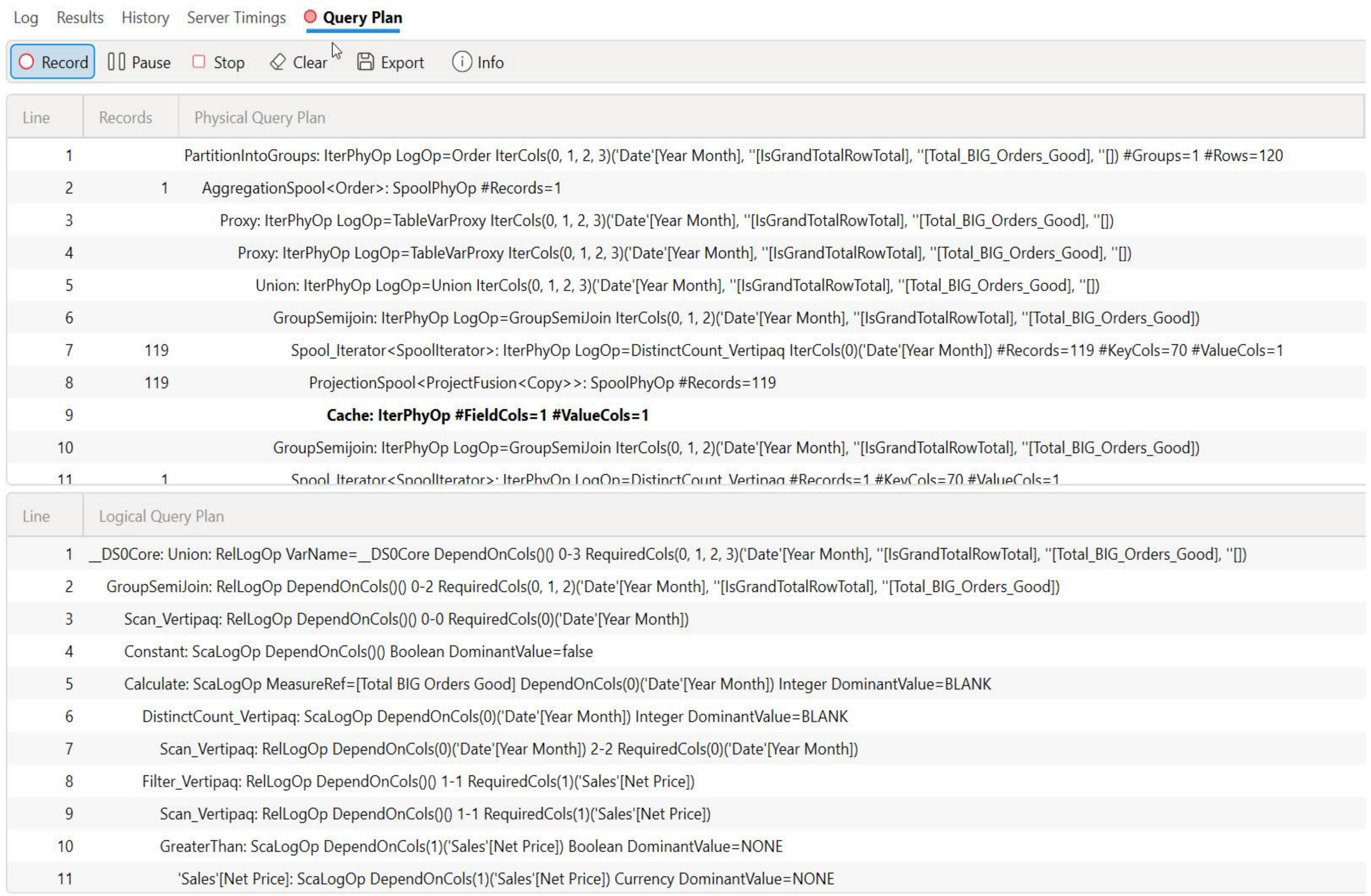


```
Line  Records  Physical Query Plan
1              PartitionIntoGroups: IterPhyOp LogOp=Order IterCols(0, 1, 2, 3)('Date'[Year Month], ''[IsGrandTotalRowTotal], ''[Total_BIG_Orders_Good], ''[]) #Groups=1 #Rows=120
2     1          AggregationSpool<Order>: SpoolPhyOp #Records=1
3                  Proxy: 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 #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

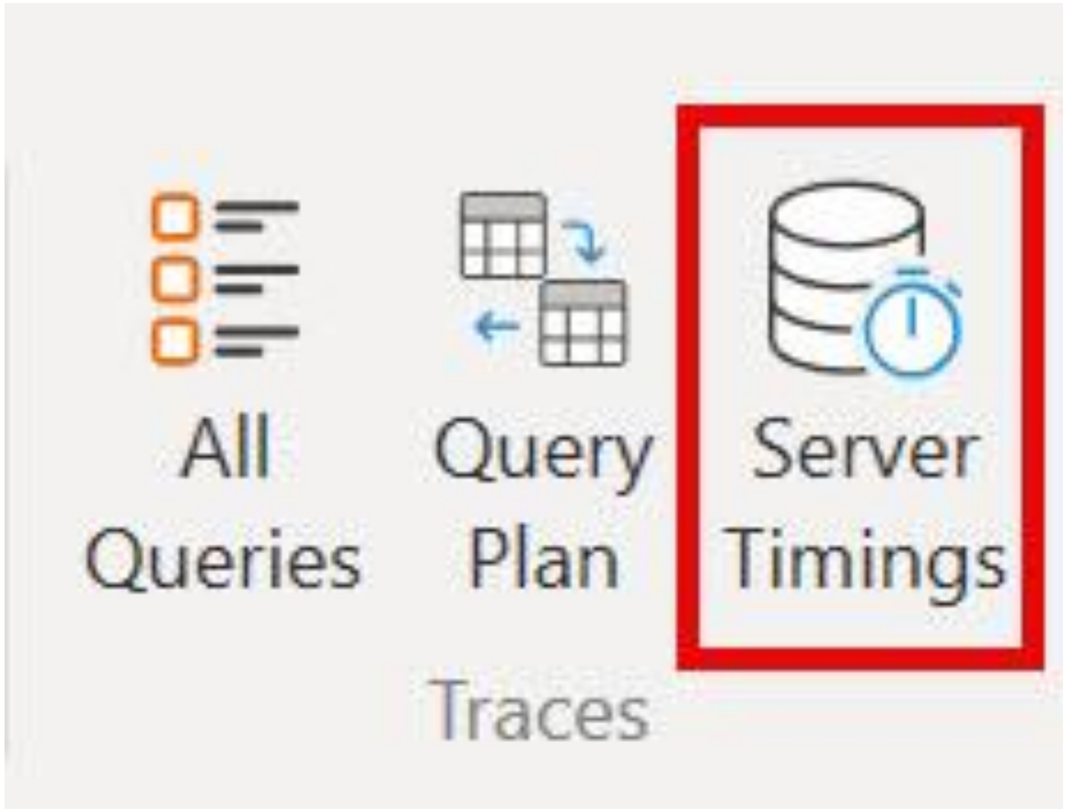
DAX Studio – Query Plan



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 #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

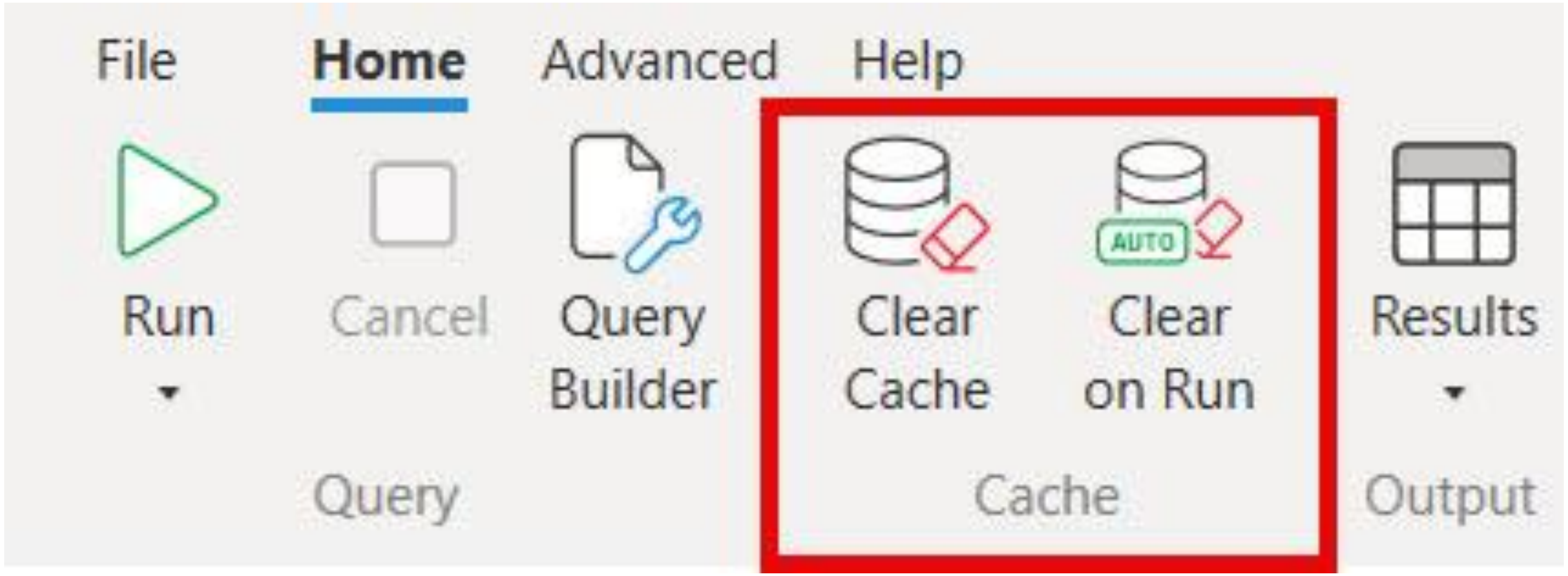
Formula Engine

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

Storage Engine

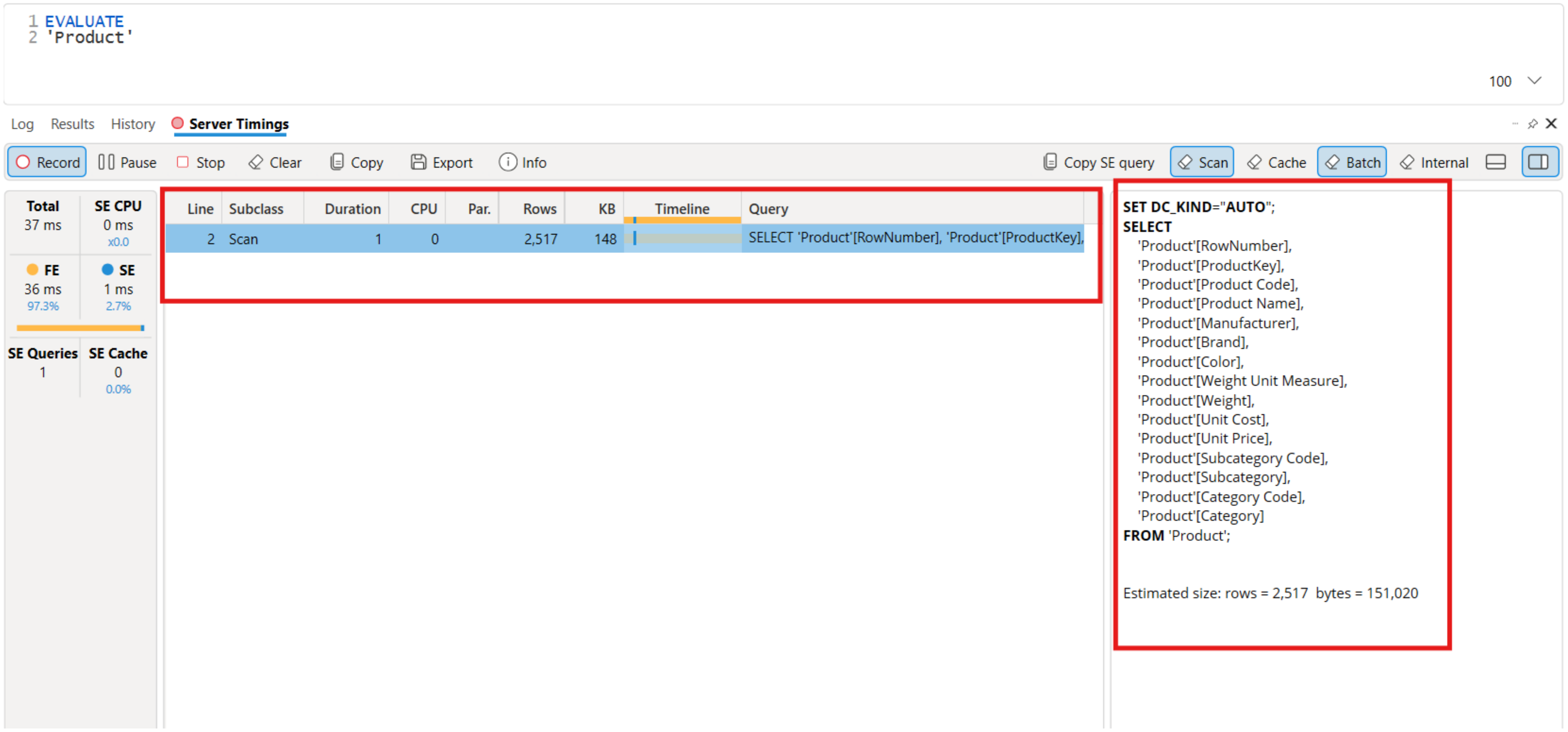
- Can cache
- Can be multithreaded
- Operations depend on the storage engine
  - Vertipaq is very limited

Cache


File
Home
Advanced
Help
Run
Cancel
Query Builder
Clear Cache
AUTO
Clear on Run
Results
Query
Cache
Output

```
EVALUATE
'Product'
```

Storage Engine Query



```
EVALUATE
'Product'
```

100

Log Results History Server Timings

Record Pause Stop Clear Copy Export Info Copy SE query Scan Cache Batch Internal

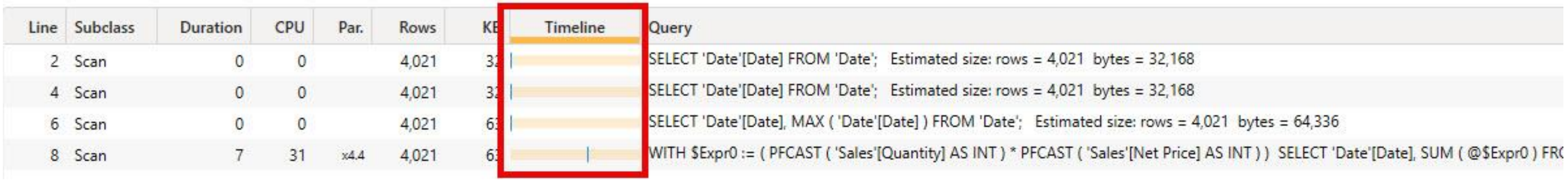
Total	SE CPU
37 ms	0 ms x0.0
FE 36 ms 97.3%	SE 1 ms 2.7%
SE Queries 1	SE Cache 0 0.0%

Line	Subclass	Duration	CPU	Par.	Rows	KB	Timeline	Query
2	Scan	1	0		2,517	148		SELECT 'Product'[RowNumber], 'Product'[ProductKey],

```
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
```

Storage Engine Query - Timeline

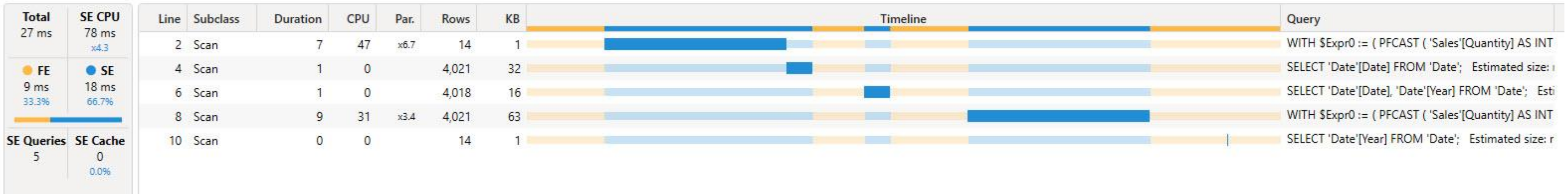
This one is mainly in FE.
Blue shows where SE comes into the timeline



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.



Total	SE CPU
27 ms	78 ms x4.3
● FE 9 ms 33.3%	● SE 18 ms 66.7%
SE Queries 5	**SE Cache** 0 0.0%

Line	Subclass	Duration	CPU	Par.	Rows	KB	Timeline	Query
2	Scan	7	47	x6.7	14	1		WITH $Expr0 := (PFCAST ('Sales'[Quantity] AS INT
4	Scan	1	0		4,021	32		SELECT 'Date'[Date] FROM 'Date'; Estimated size:
6	Scan	1	0		4,018	16		SELECT 'Date'[Date], 'Date'[Year] FROM 'Date'; Esti
8	Scan	9	31	x3.4	4,021	63		WITH $Expr0 := (PFCAST ('Sales'[Quantity] AS INT
10	Scan	0	0		14	1		SELECT 'Date'[Year] FROM 'Date'; Estimated size: r

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';
```

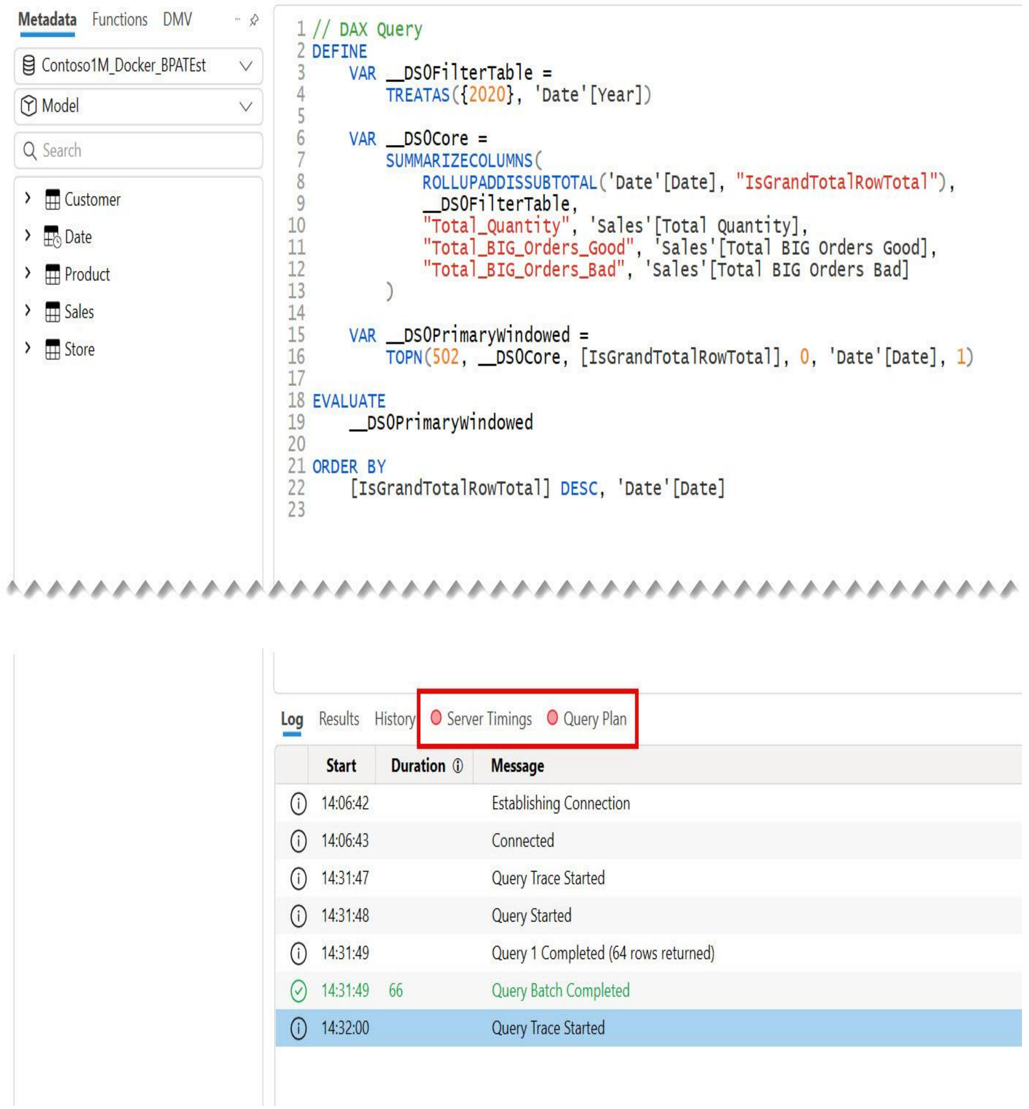
Type of Storage Engine

- Vertipaq
  - Needs Formula Engine for IF Statement
- SQL Server
  - Can push IF equivalent to SQL Server

Vertipaq

- Column Based
- Compressed
- Encoded

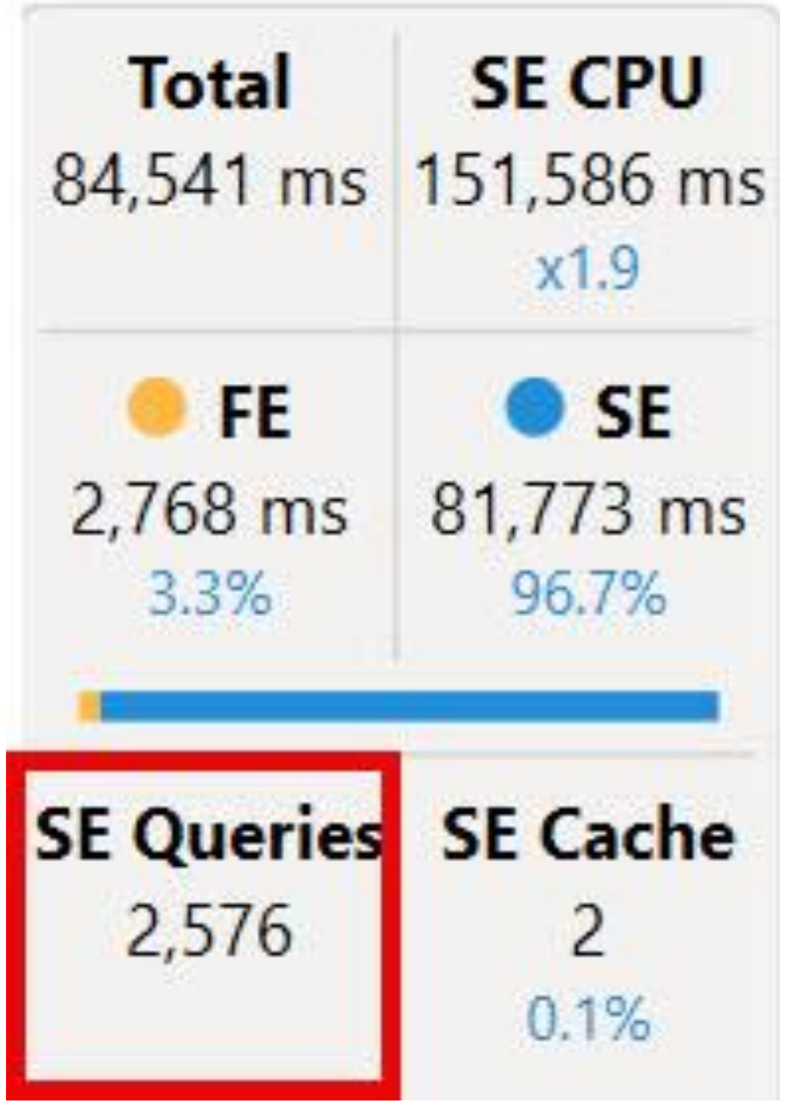
Location of Query and Server Timings



```
// DAX Query
DEFINE
    VAR __DS0FilterTable =
        TREATAS({2020}, 'Date'[Year])

    VAR __DS0Core =
        SUMMARIZECOLUMNS(
            ROLLUPADDISSUBTOTAL('Date'[Date], "IsGrandTotalRowTotal"),
            __DS0FilterTable,
            "Total_Quantity", 'Sales'[Total Quantity],
            "Total_BIG_Orders_Good", 'Sales'[Total BIG Orders Good],
            "Total_BIG_Orders_Bad", 'Sales'[Total BIG Orders Bad]
        )

    VAR __DS0PrimaryWindowed =
        TOPN(502, __DS0Core, [IsGrandTotalRowTotal], 0, 'Date'[Date], 1)

EVALUATE
    __DS0PrimaryWindowed

ORDER BY
    [IsGrandTotalRowTotal] DESC, 'Date'[Date]
```

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 Studio – Server Timings – Bad

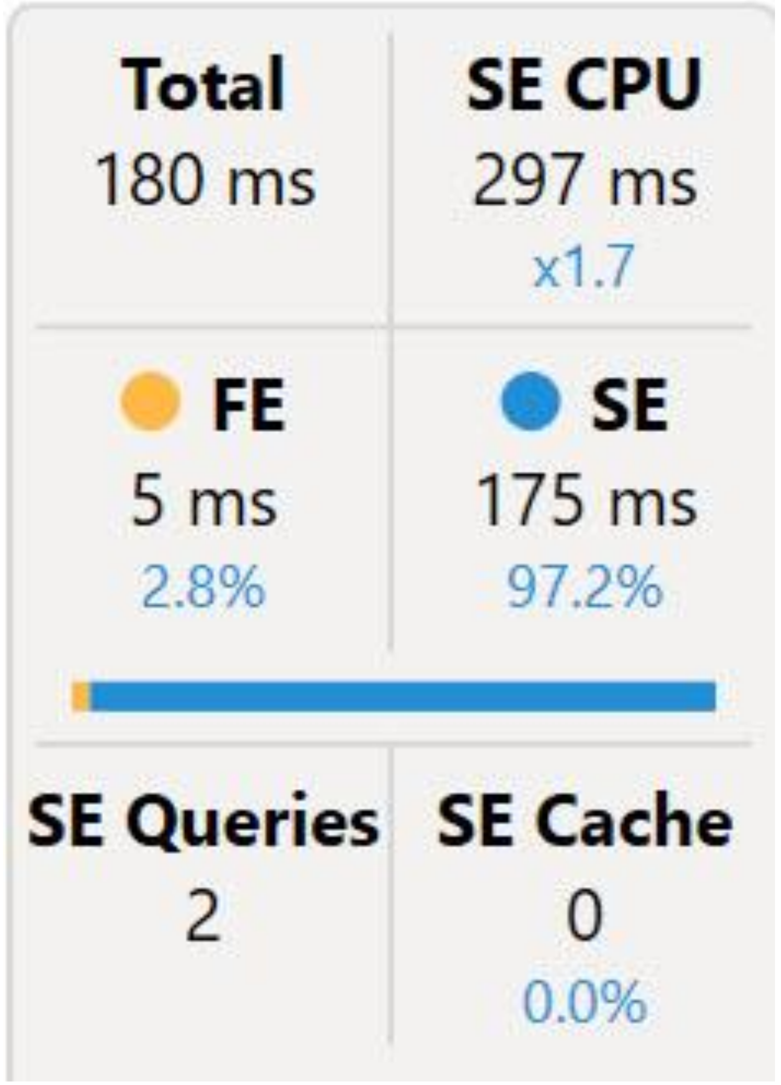


Total 84,541 ms	**SE CPU** 151,586 ms x1.9
● **FE** 2,768 ms 3.3%	● **SE** 81,773 ms 96.7%
SE Queries 2,576	**SE Cache** 2 0.1%

Server Timings – Bad – DAX Code

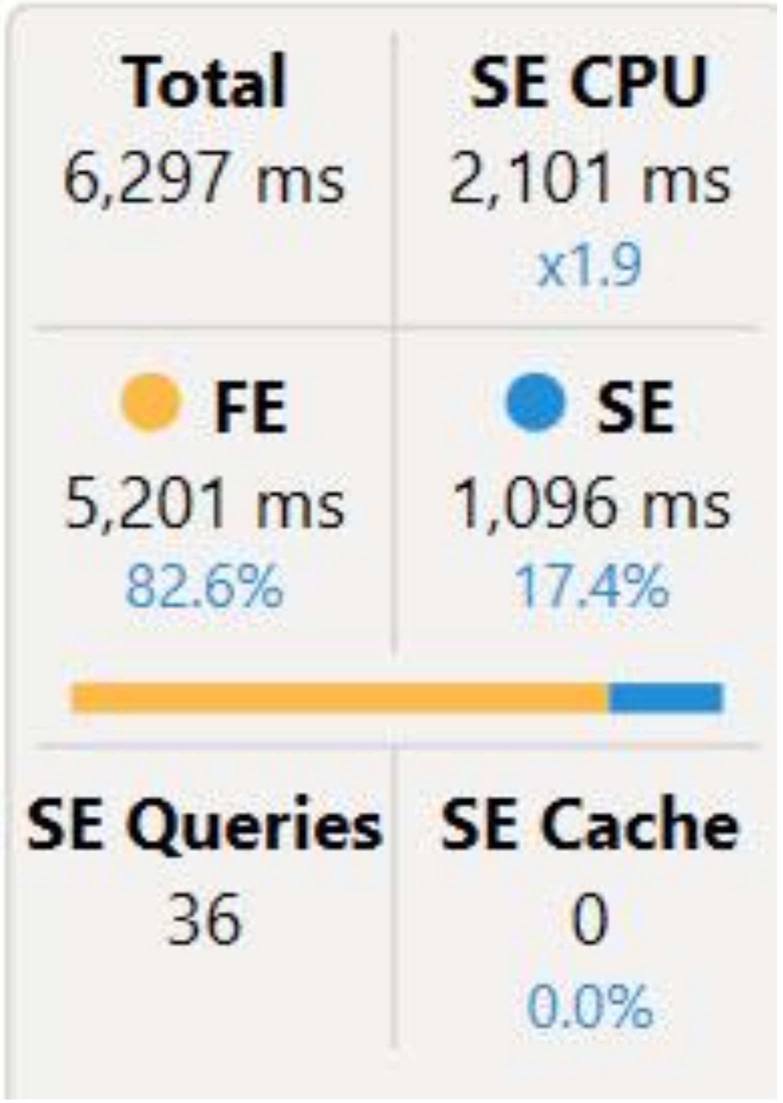
```
CALCULATE (
    DISTINCTCOUNT ( Sales[Order Number] ),
    FILTER ( 'Sales', 'Sales'[Net Price] > 10 )
)
```

DAX Studio – Server Timings – Good



Total	SE CPU
180 ms	297 ms x1.7
● FE 5 ms 2.8%	● SE 175 ms 97.2%
SE Queries 2	SE Cache 0 0.0%

DAX Studio – Most Formula Engine



Total	SE CPU
6,297 ms	2,101 ms x1.9
● FE 5,201 ms 82.6%	● SE 1,096 ms 17.4%
SE Queries 36	SE Cache 0 0.0%

Ways to invoke Formula Engine

- IF Statements
- CONCATENATE

CallbackDataID

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

Simple way to increase Query Time

- Filter by the full table

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

The more you know…

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

Direct Query Model

- Direct Query
  - How many queries are being sent to Source
  - What about Date table in Import and rest DQ

Power BI Demo
DAX Studio

Our Journey

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

Tabular Editor

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

Tabular Editor 2.x

- Free version
- Version listed in DP-600 Study Guide

Tabular Editor 3.x

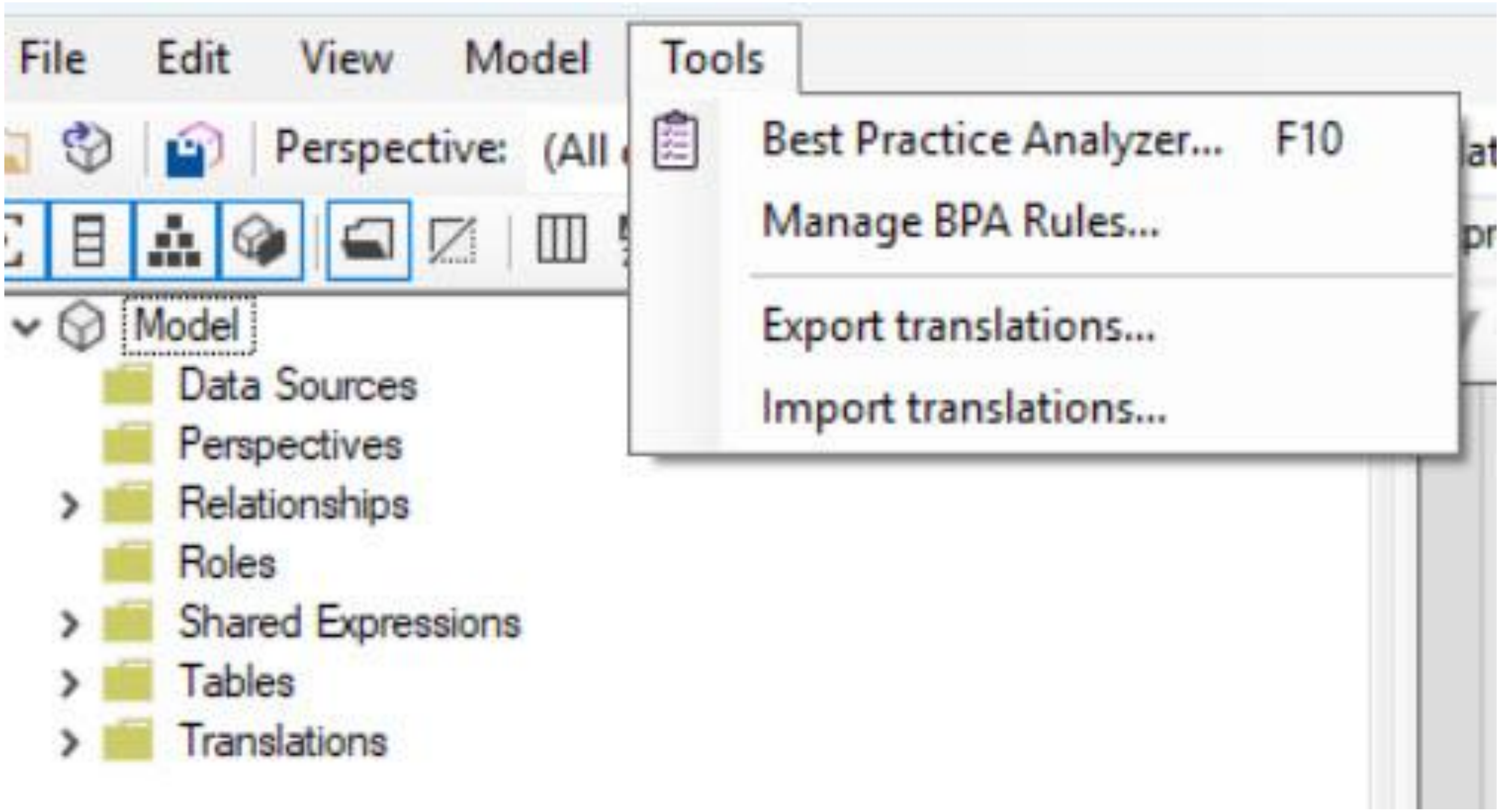
- Paid Version
- Extra Features
  - DAX Debugger

Tabular Editor 2

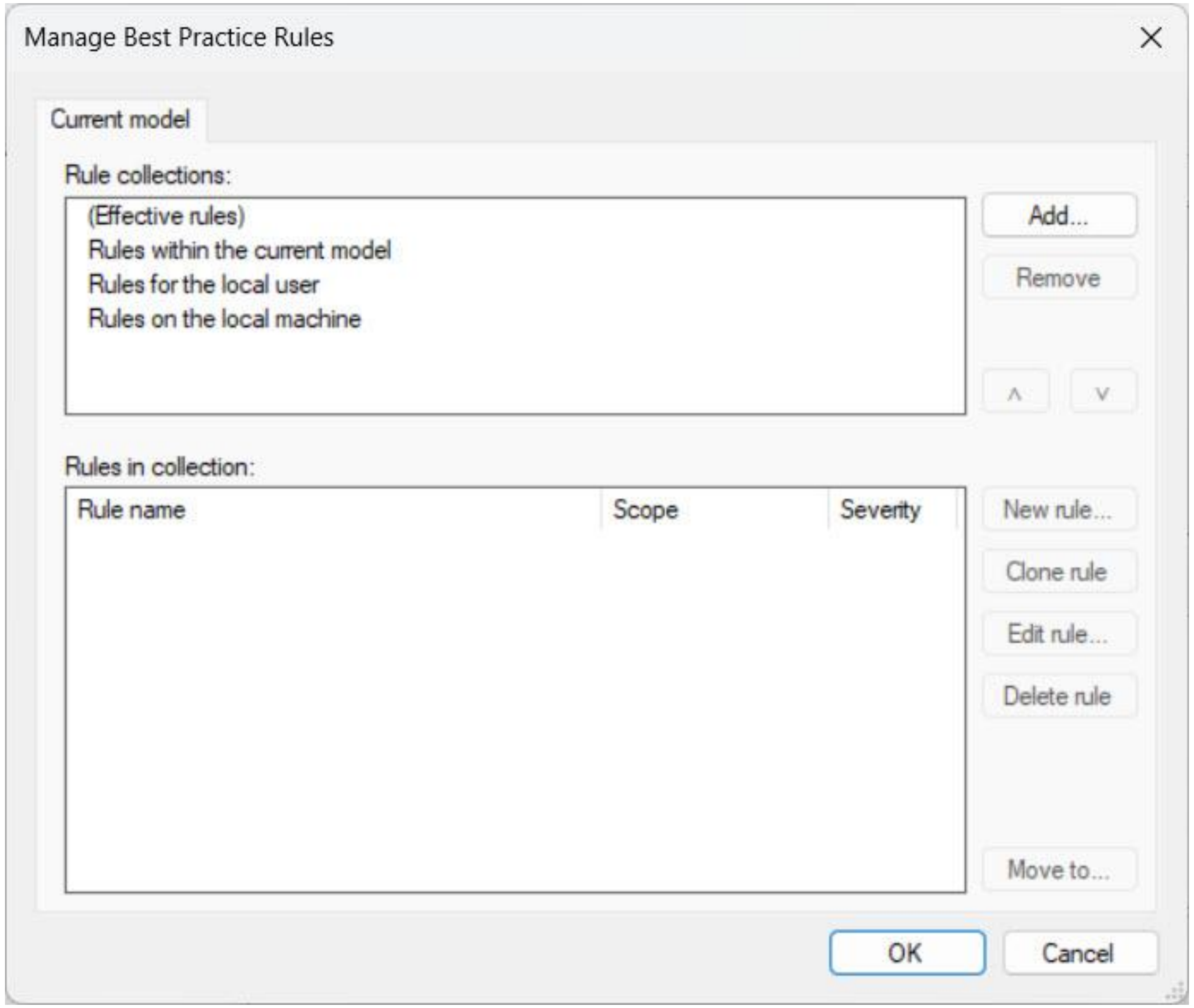
Optimize a semantic model by using Tabular Editor 2

Best Practice Analyzer

Best Practice Analyzer Location



Add and Manage Rules



Manage Best Practice Rules

Current model

Rule collections:

(Effective rules)
Rules within the current model
Rules for the local user
Rules on the local machine

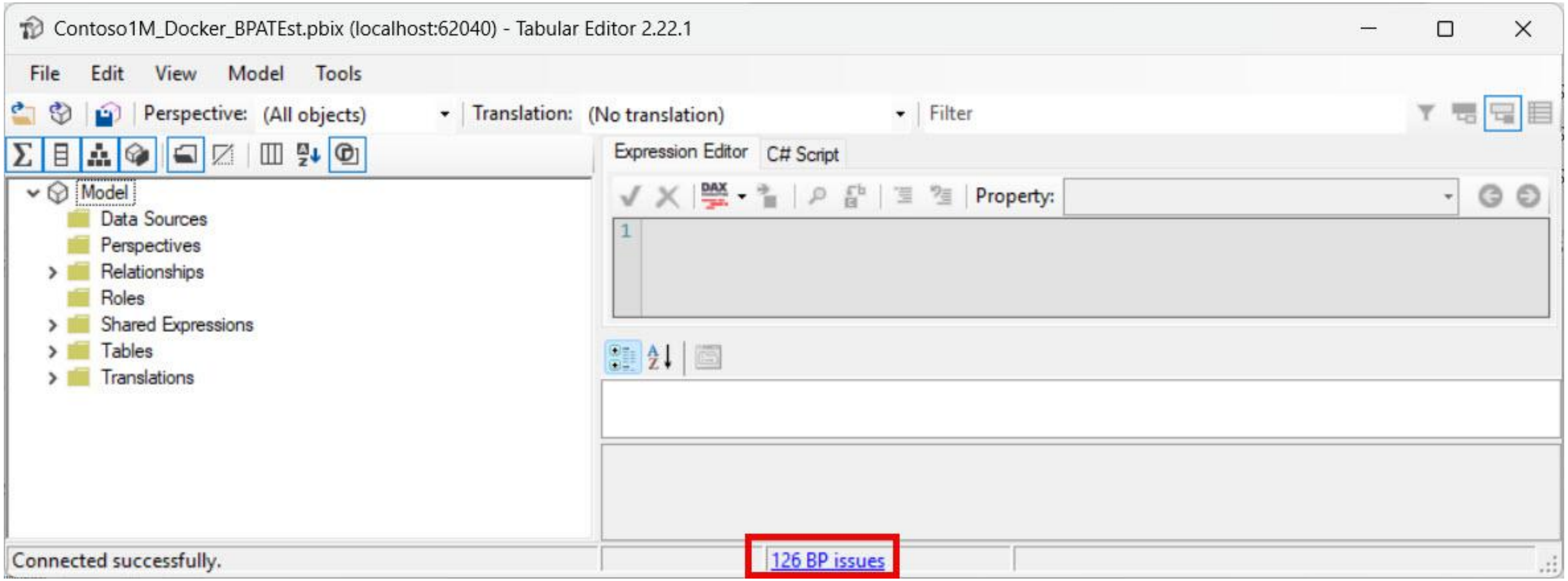
Rules in collection:

Rule name	Scope	Severity

Best Practice Analyzer – C# Script Download

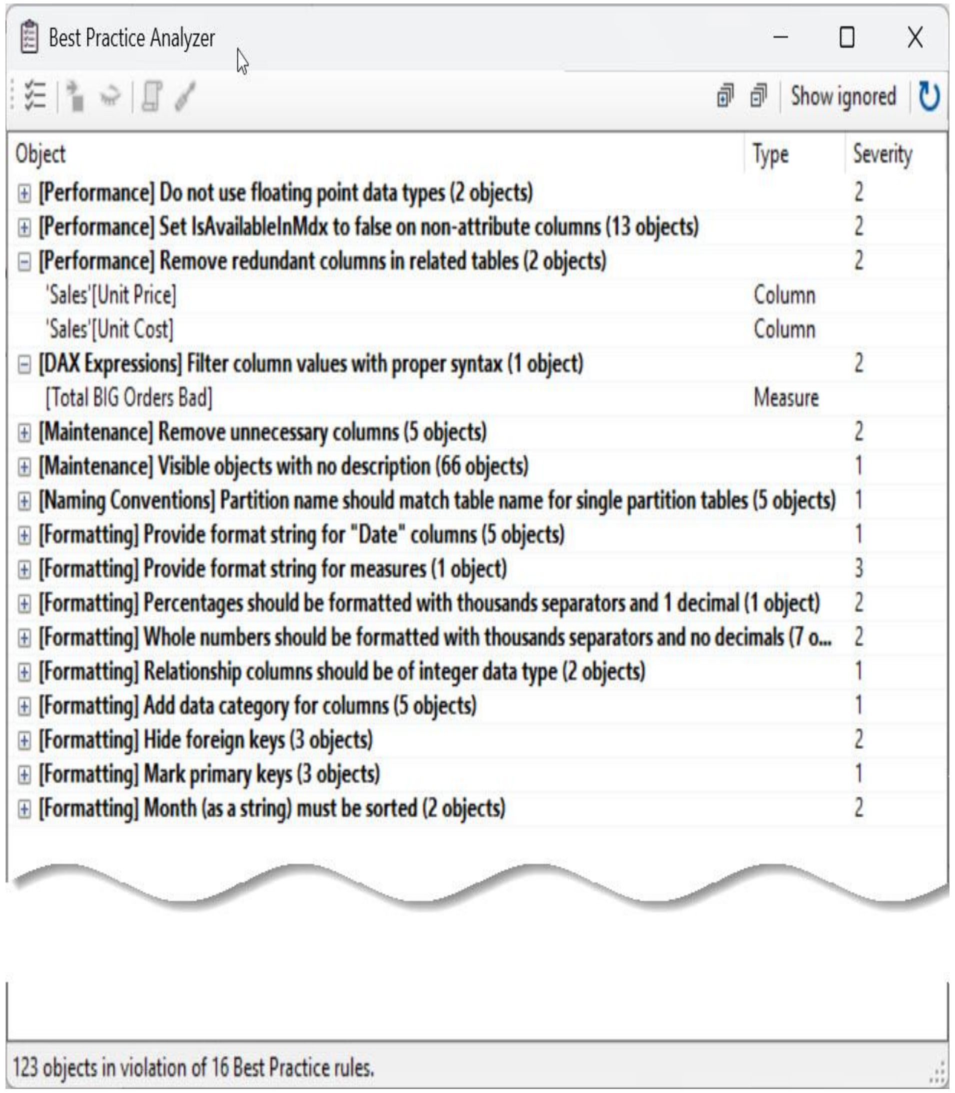
```
System.Net.WebClient w = new System.Net.WebClient();
string path = System.Environment.GetFolderPath
(System.Environment.SpecialFolder.LocalApplicationData);
string url =
"https://raw.githubusercontent.com/microsoft/Analysis-Services/master/
      BestPracticeRules/BPARules.json";
string downloadLoc = path+@"\TabularEditor\BPARules.json";
w.DownloadFile(url, downloadLoc);
```

BPA – Rule Exceptions for model



Connected successfully.

BPA – Rule Exceptions for model



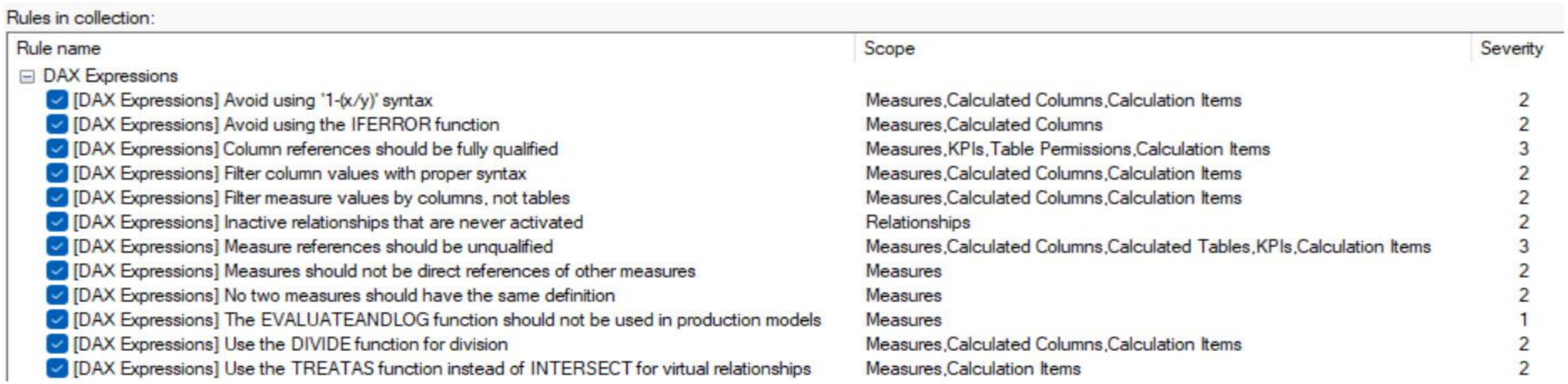
Best Practice Analyzer

Show ignored

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.

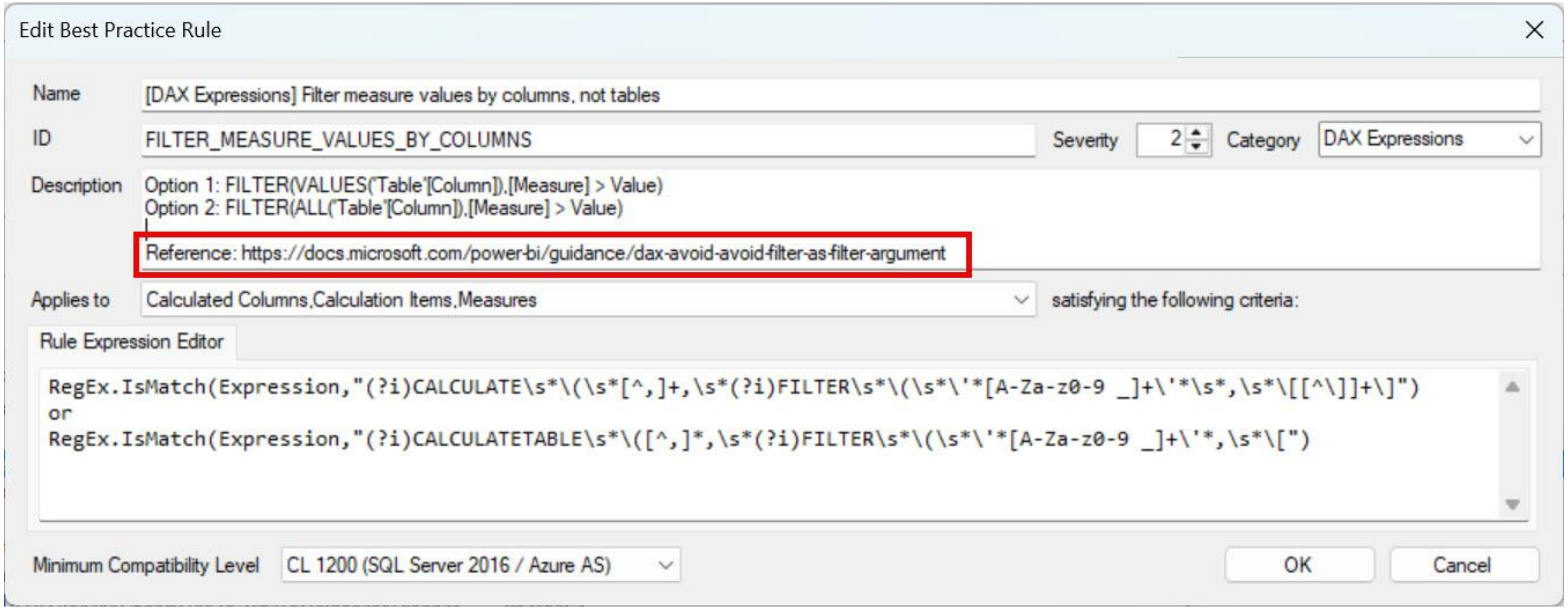
BPA – Manage Rules
DAX Expressions



Rules in collection:

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

Edit Rule



Edit Best Practice Rule

Name: [DAX Expressions] Filter measure values by columns, not tables

ID: FILTER_MEASURE_VALUES_BY_COLUMNS

Severity: 2

Category: DAX Expressions

Description:

Option 1: FILTER(VALUES('Table'[Column]),[Measure] > Value)
Option 2: FILTER(ALL('Table'[Column]),[Measure] > Value)

Reference: https://docs.microsoft.com/power-bi/guidance/dax-avoid-avoid-filter-as-filter-argument

Applies to: Calculated Columns,Calculation Items,Measures satisfying the following criteria:

Rule Expression Editor

```
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*\[")
```

Minimum Compatibility Level: CL 1200 (SQL Server 2016 / Azure AS)

Description with help link

- Instead of using this pattern FILTER('Table',[Measure]>Value) for the filter parameters of a CALCULATE or CALCULATETABLE function, use one of the options below (if possible). Filtering on a specific column will produce a smaller table for the engine to process, thereby enabling faster performance. Using the VALUES function or the ALL function depends on the desired measure result.

- Option 1: FILTER(VALUES('Table'[Column]),[Measure] > Value)
- Option 2: FILTER(ALL('Table'[Column]),[Measure] > Value)

- Reference: https://docs.microsoft.com/power-bi/guidance/dax-avoid-avoid-filter-as-filter-argument

Power BI Demo
Tabular Editor

Our Journey

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 Editior

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

Thank you

The Dax Shepherd

Jason Romans
thedaxshepherd@gmail.com
www.thedaxshepherd.com