SQLSATURDAY

#1068 - Jacksonville, FL - May 4th, 2024

Power up your Fabric Development with DAX Studio and Tabular Editor

JACKSONVILLE

SQL SERVER USERS GROUP

#SQLSatJax

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

A Padawan Learning from the DAX Jedis

I wouldn’t be where I am without the DAX Jedis

DAX or DAX Not, There is No Try

Warning:

MDX

Leads to the Dark Side

SQL SATURDAY

SQLSATURDAY
#1068 - Jacksonville, FL - May 4th, 2024

JACKSONVILLE SQL SERVER USERS GROUP

Thank you to ALL of our sponsors! - Be sure to stop by all tables!

AI Platinum

NLP LOGIX
AI REALIZED

Gold

VyStar Credit Union

vmware by Broadcom

Platinum

UNF School of Computing
College of Computing, Engineering and Construction

SIOS

CENTRIC

IDERA

silk

DBHawk

Silver

COZYROC

DBeaver

onecall

SQLGrease

Quest

Bronze

Simple Talk

Octopus Deploy

Microsoft

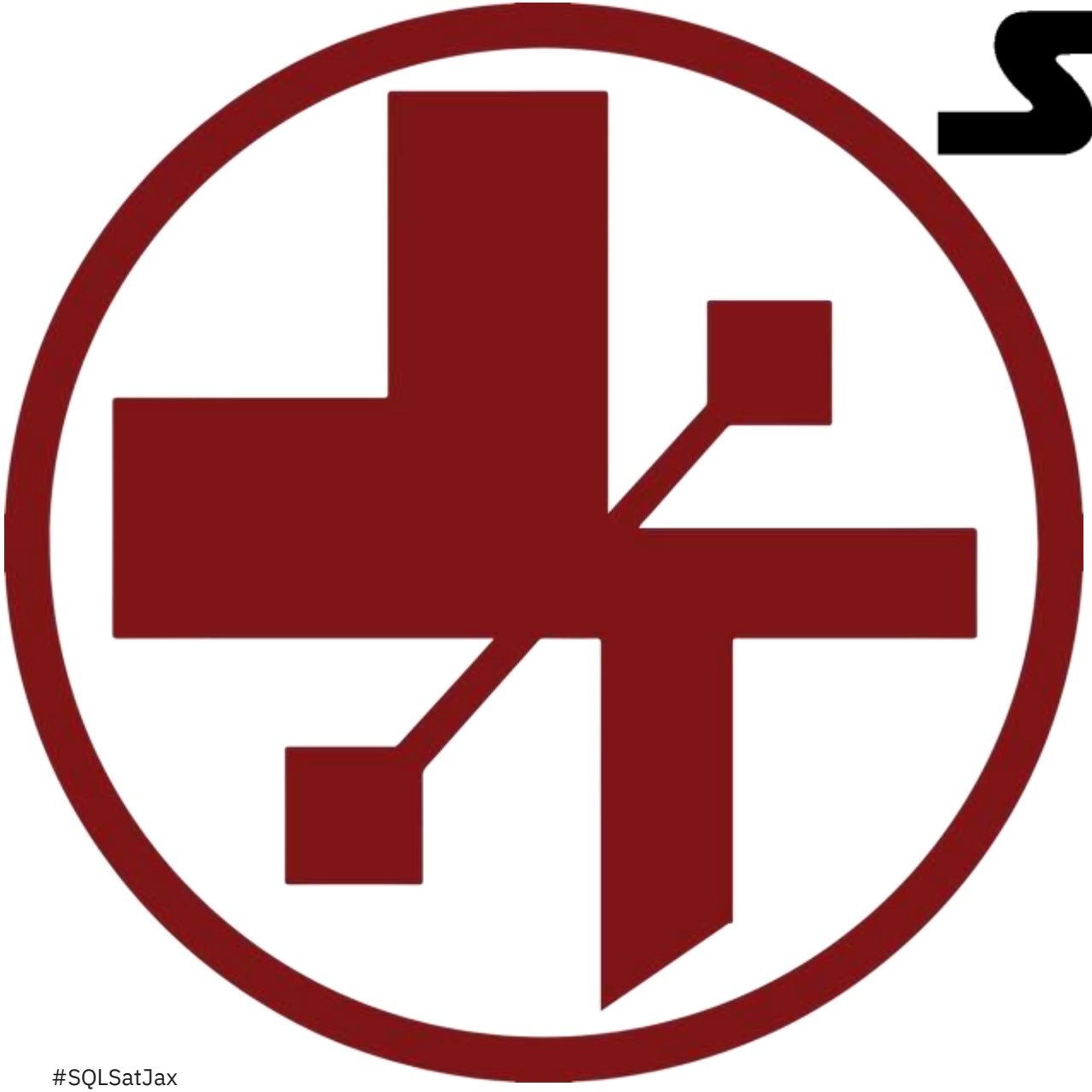
In-Kind

O'REILLY

Monthly Meetings

3rd Wednesday of each month

jssug.org



SQL CLINIC

Get your SQL Questions answered here by our SQL Experts

(Across from Registration)

#SQLSatJax

SQLSATURDAY

#1068 - Jacksonville, FL - May 4th, 2024

501 Legion Charitable Donation & LEGO Drive

Thank the 501 Legion for Supporting Our Event!

JSSUG Will Match Donations up to $1000

Donation Bucket on Registration Table

LEGO donation on V for Victory Table

V FOR VICTORY

Costume Contest Rules

1. Take a picture with the SQL Saturday Backdrop during the event

2. Post the picture to Twitter/X and include the hashtag #SQLSatJax24CC

3. The tweet using the hashtag that has the most "likes" wins a prize!

Session Evaluations

Your feedback is important to us!

Please fill out and hand to speaker after the session!

Event Evaluation

Fill out event evaluation card in your bag and visit all sponsors to be entered to win an Xbox Series X – (Must be present to win)

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

What this session is not

- Learning DAX
- Optimizing DAX
- Different Storage Options – Direct Query
- Complete guide to External Tools

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

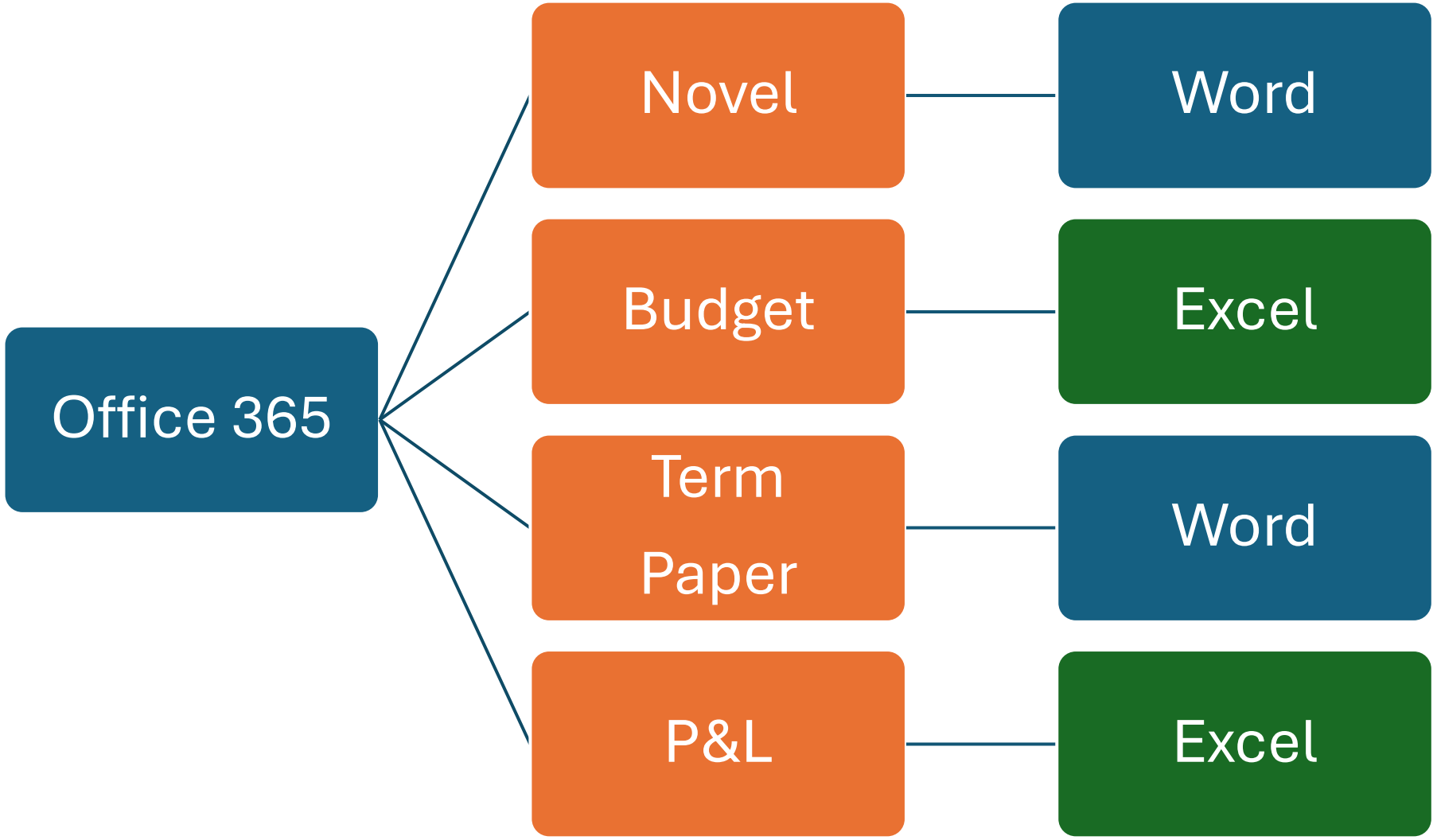
Why is my report slow?

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

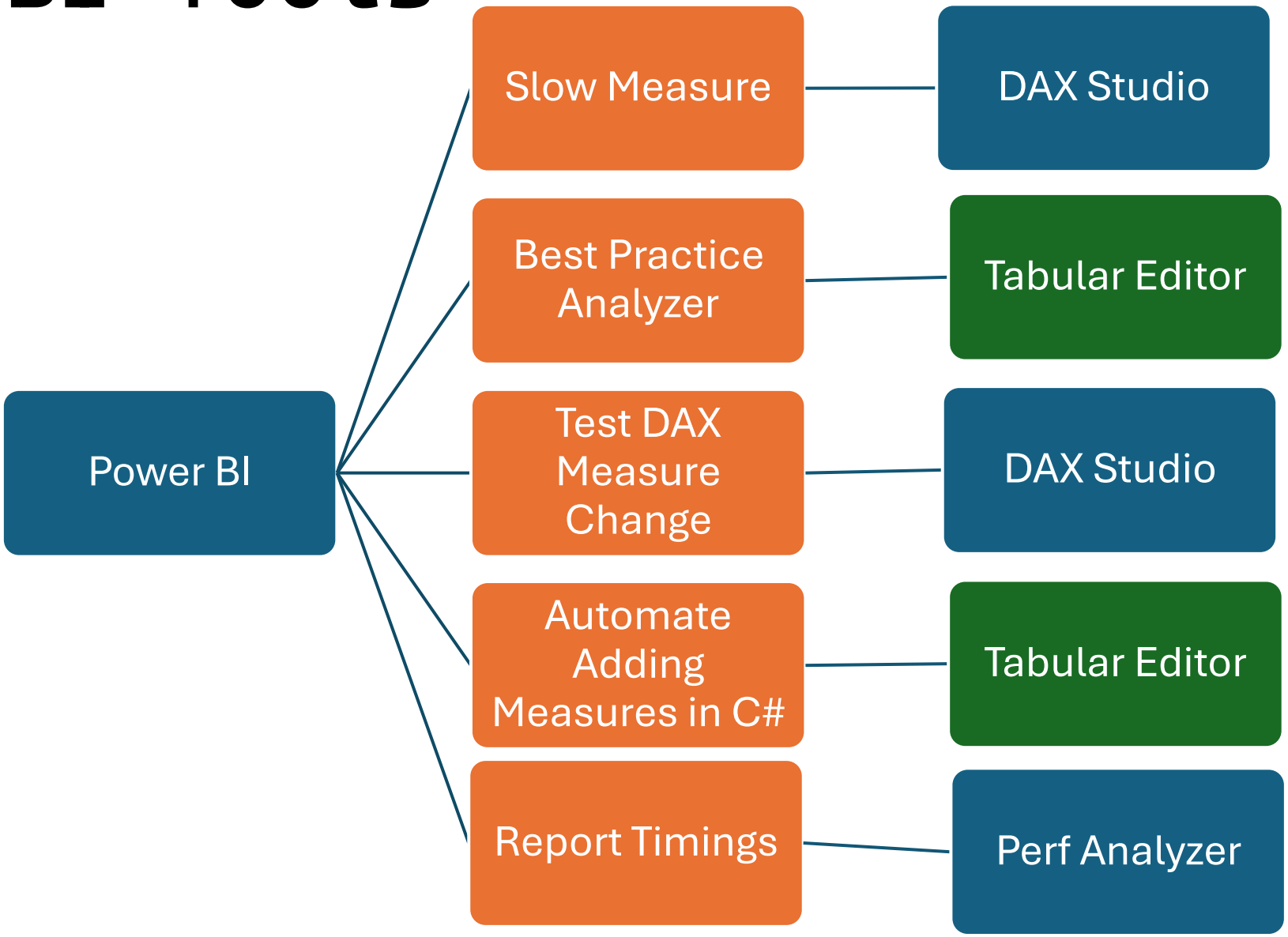
Power BI is Part of Fabric

- If you have a Fabric capacity it enhances what you can do with Power BI
  - OneLake
  - Default Semantic Model
- What we cover today applies to Power BI Pro, Premium, Fabric
  - Exceptions will be noted

Office 365



Power BI Tools



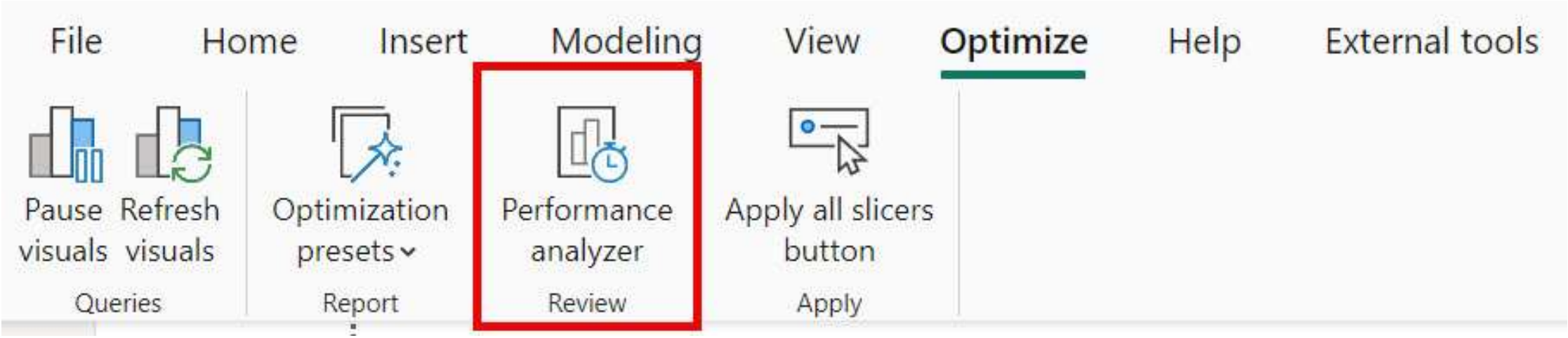
Our Journey

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

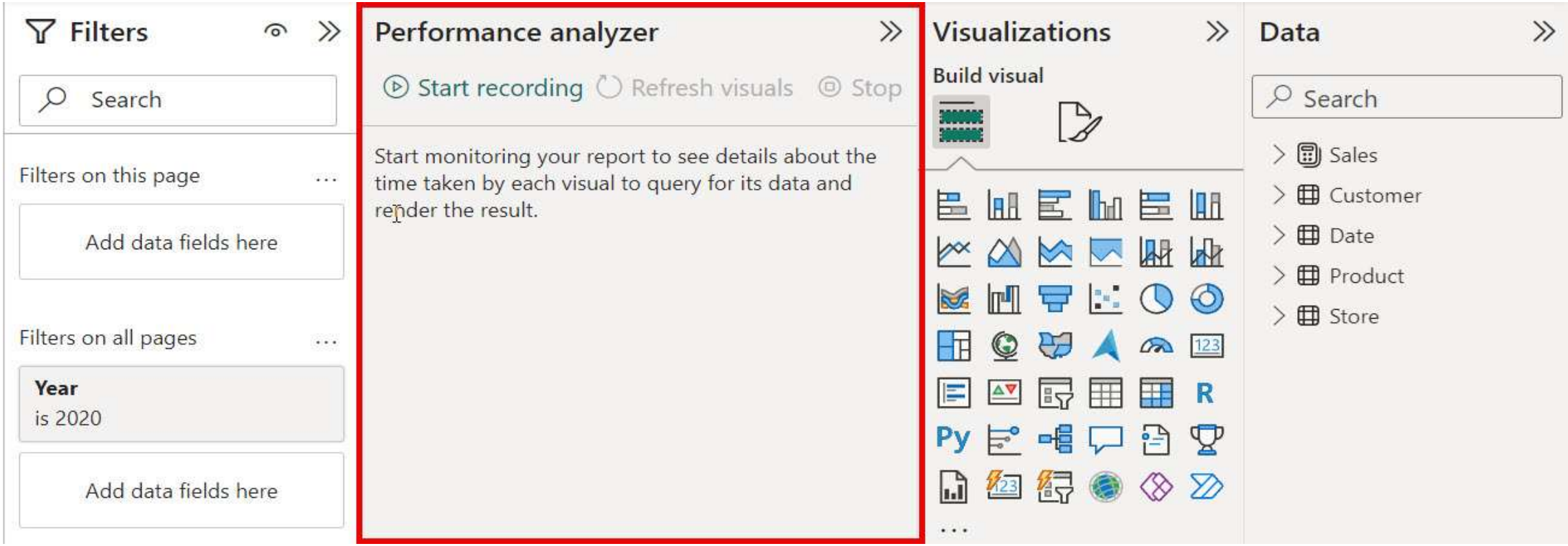
Performance Analyzer

- Not an External Tool
- Built into Power BI Desktop

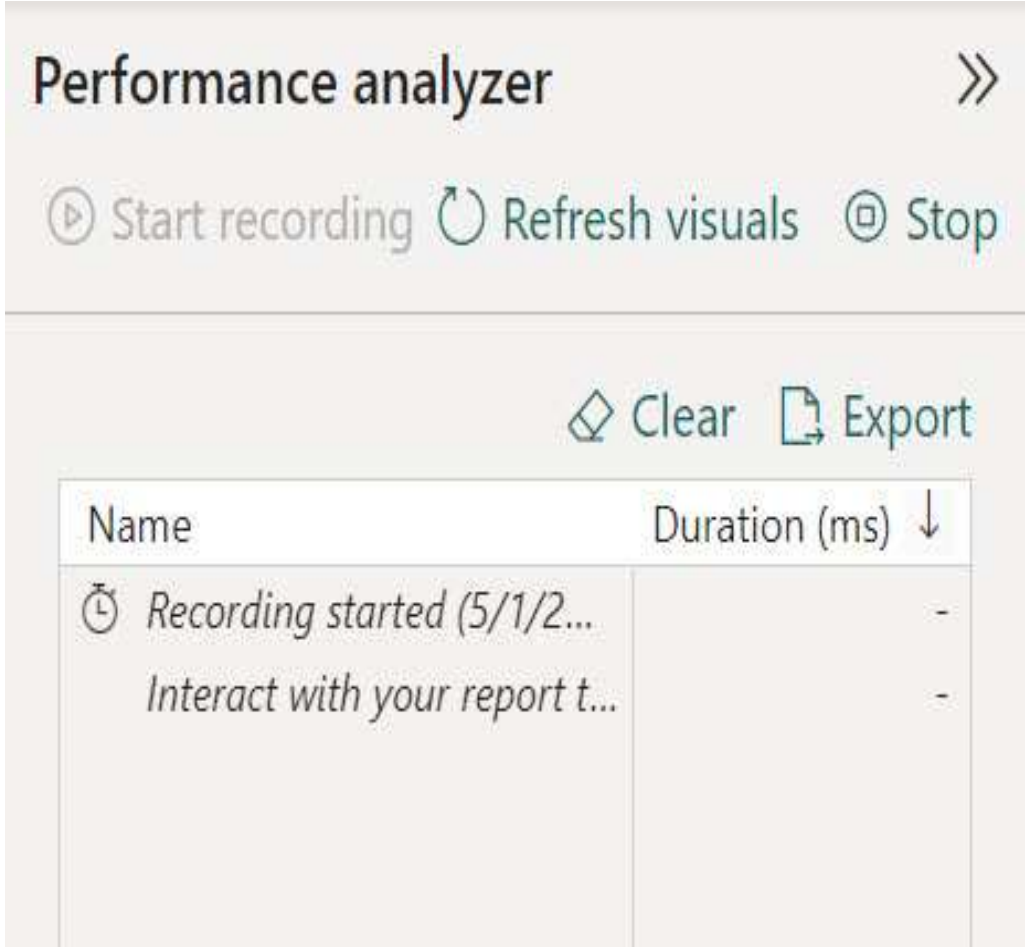
Performance Analyzer



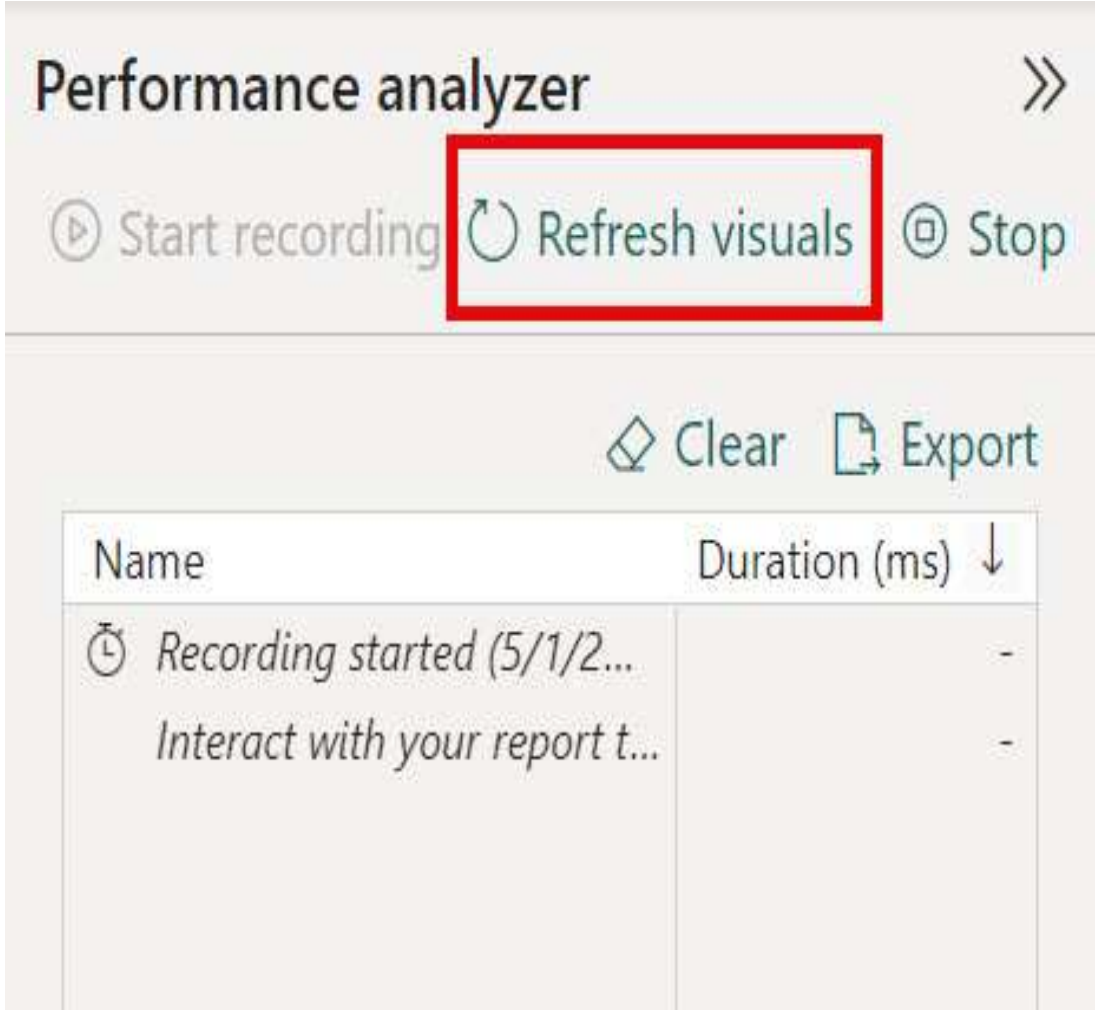
Performance Analyzer
Start Recording



Performance Analyzer Recording



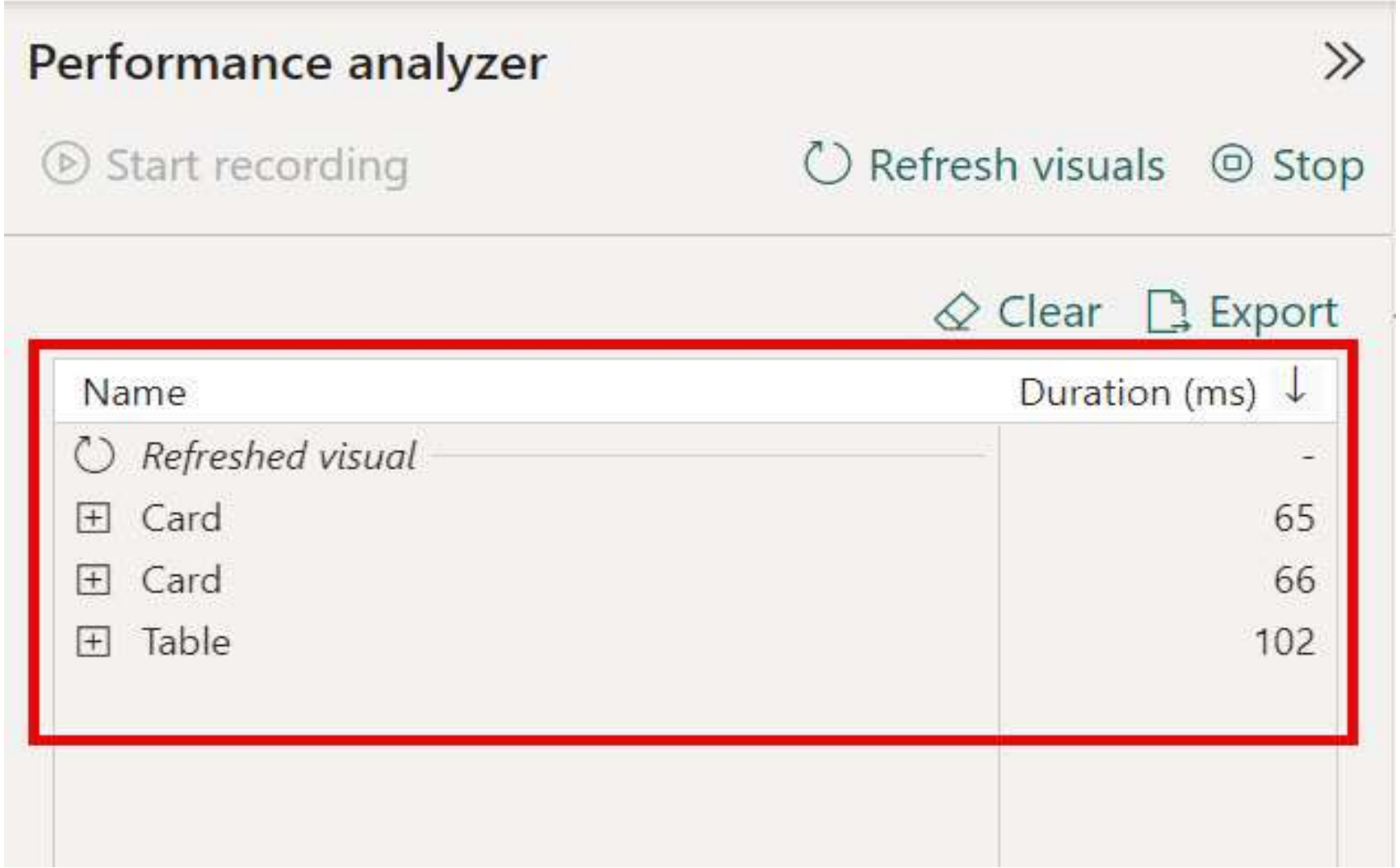
Performance Analyzer Refresh Visuals



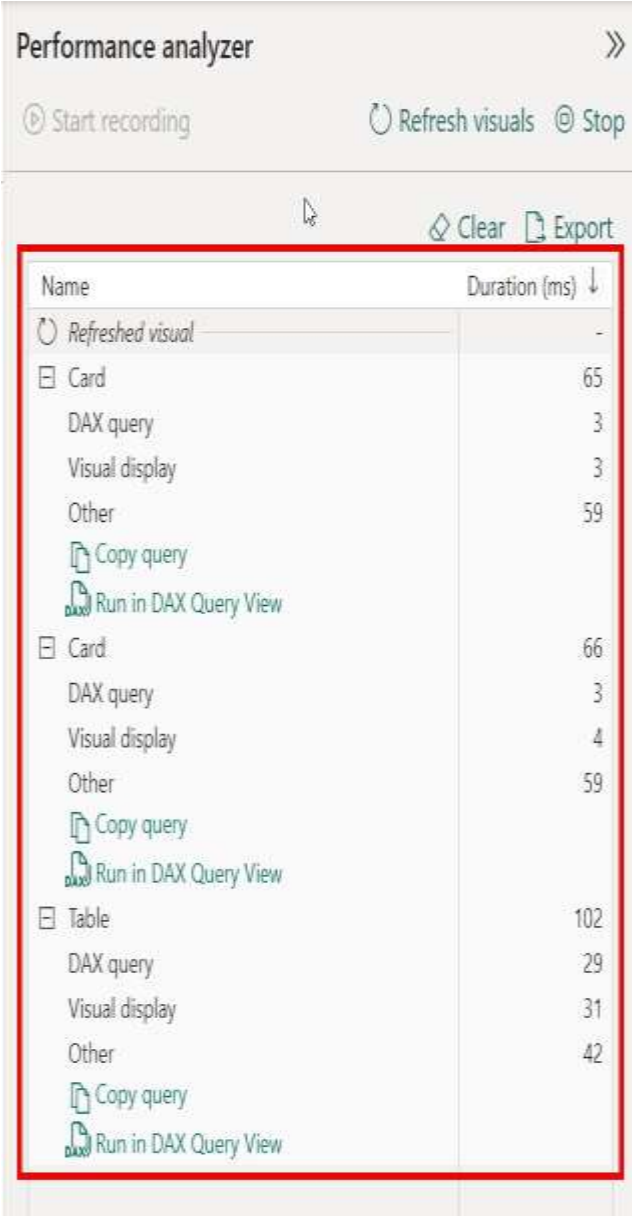
Individual Refresh Button

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



Expanded View



Detail on Card

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

DAX Query

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

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

DAX Query – Solutions

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

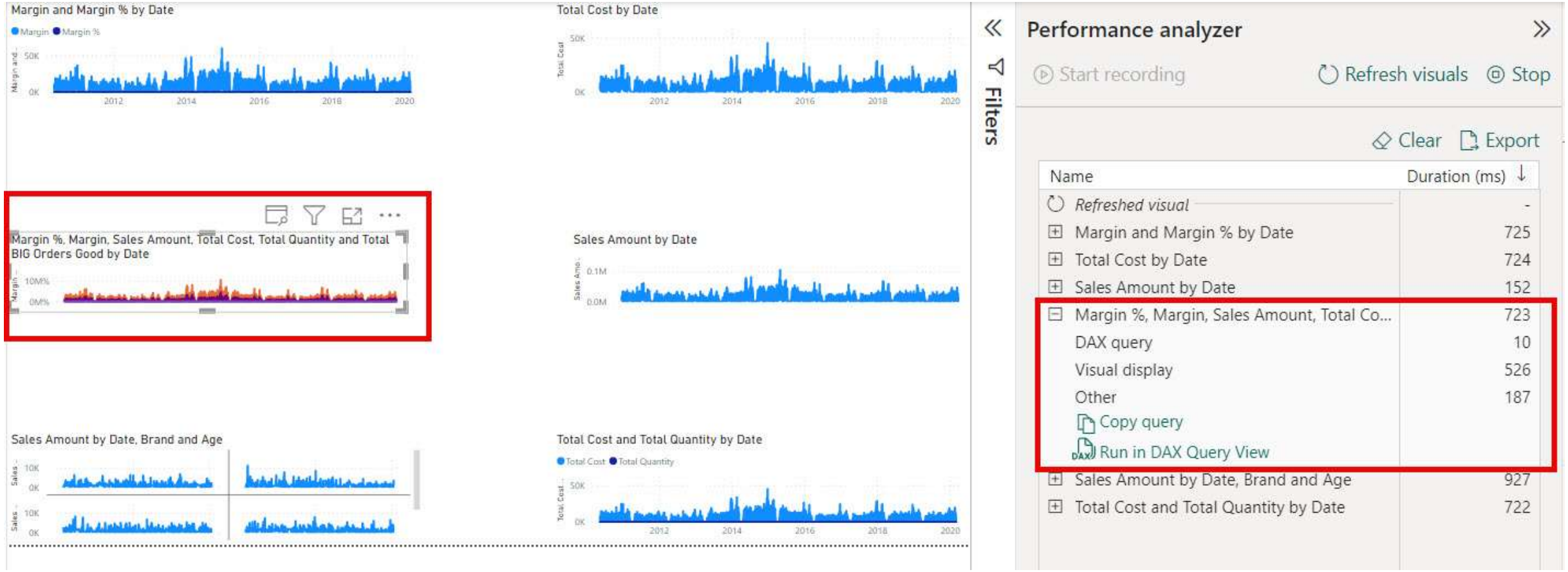
- Make DAX go faster
  - We will see how to start on this journey

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

Slow Visual


Margin and Margin % by Date
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
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... 723
DAX query 10
Visual display 526
Other 187
Copy query
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

Other

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

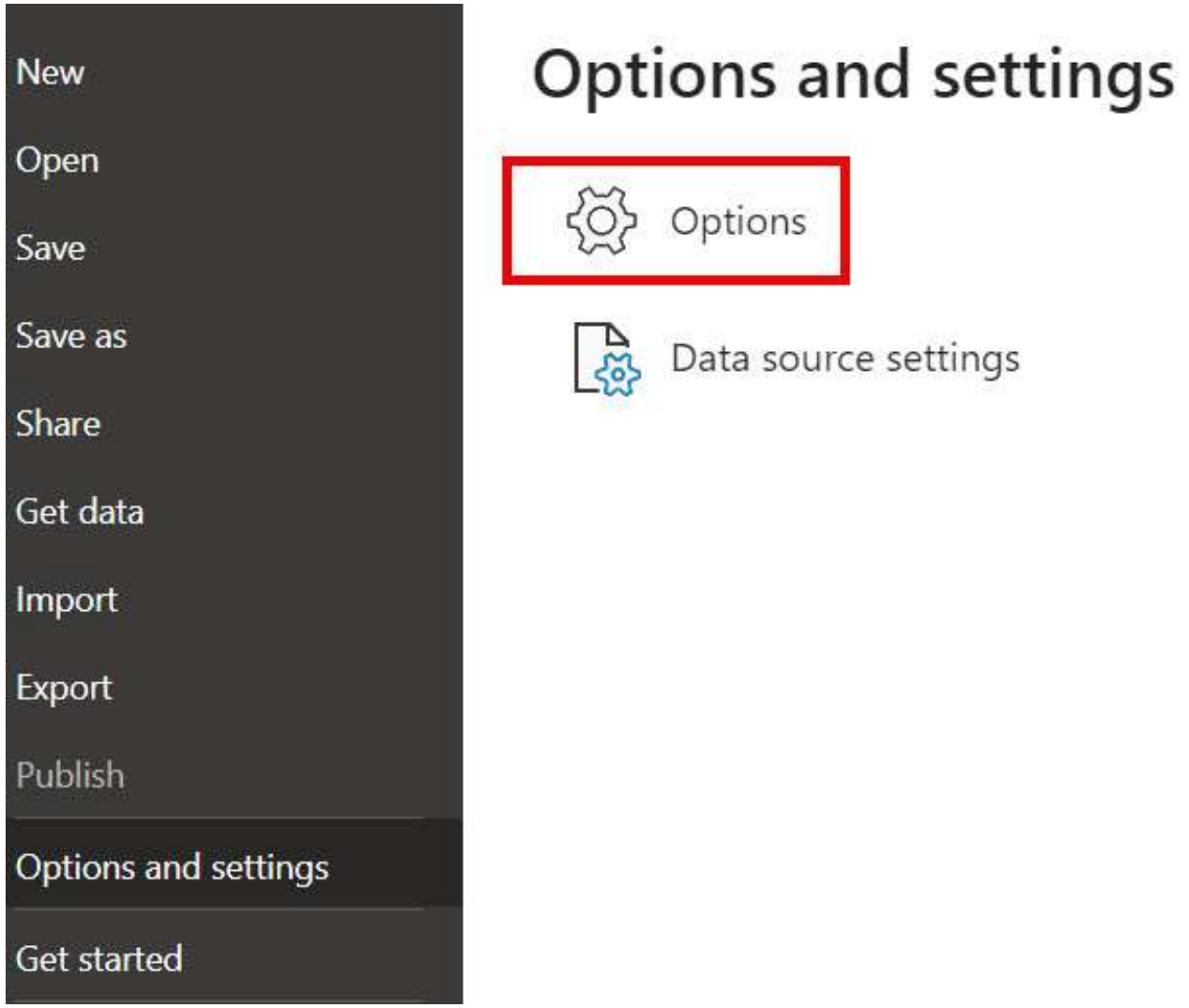
- Other
  - Time waiting until DAX Query could be executed

Other – Solutions

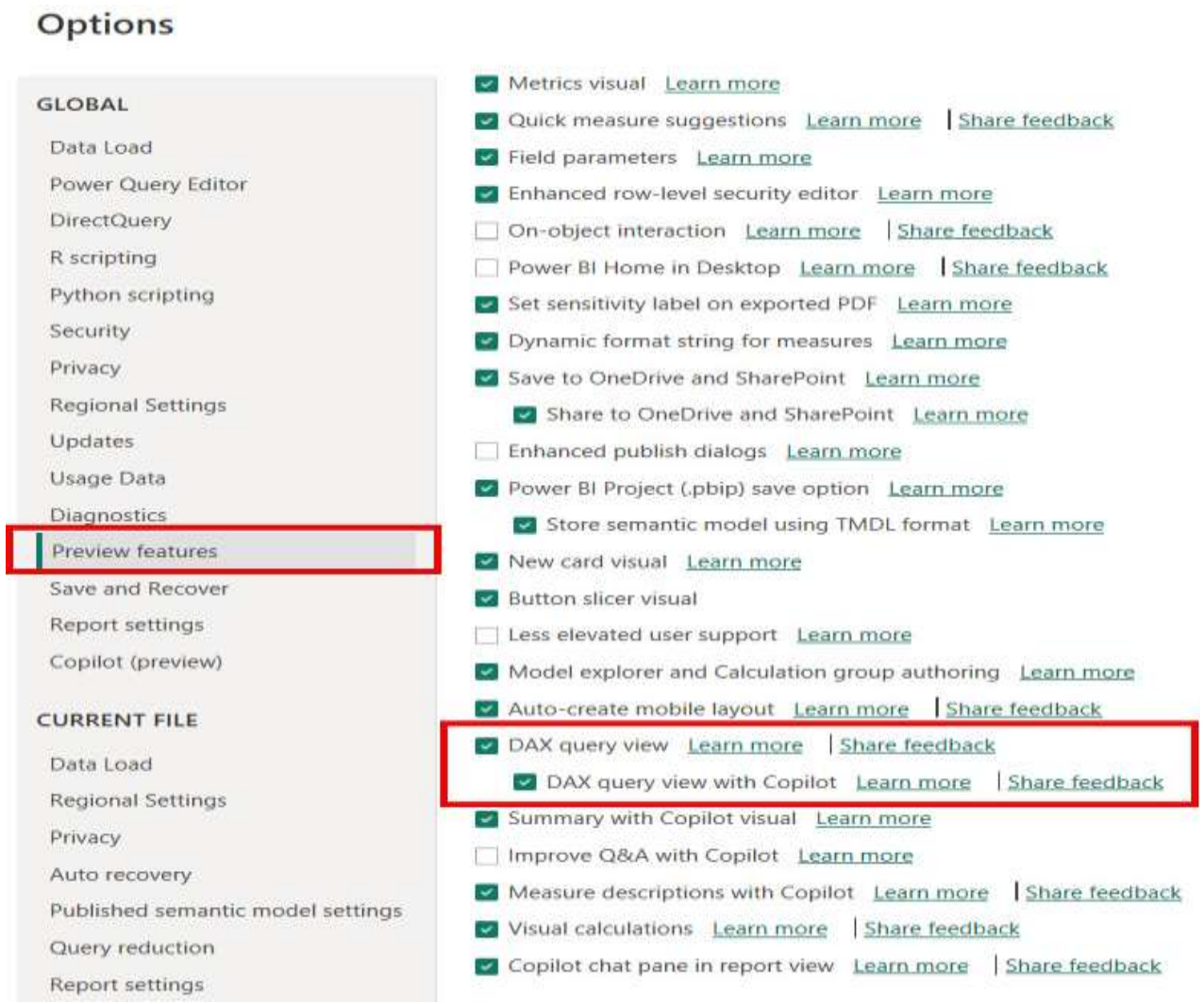
- Reduce Visualizations on the Page
- Check to see if bottleneck elsewhere

Run in DAX Query View (Preview)

Enable DAX Query View – Options



DAX Query View – Preview Features



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

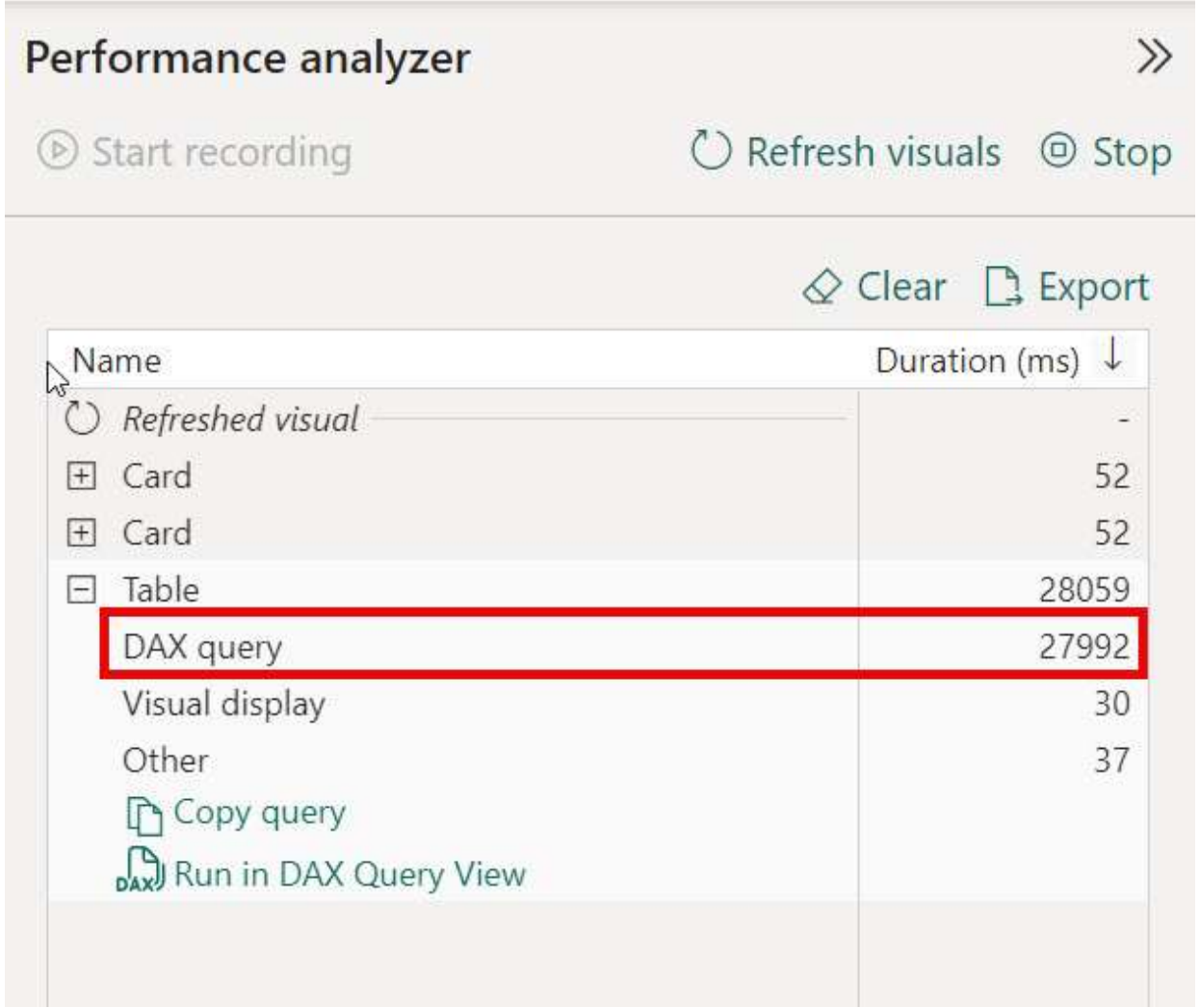
Query 1

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

Query View

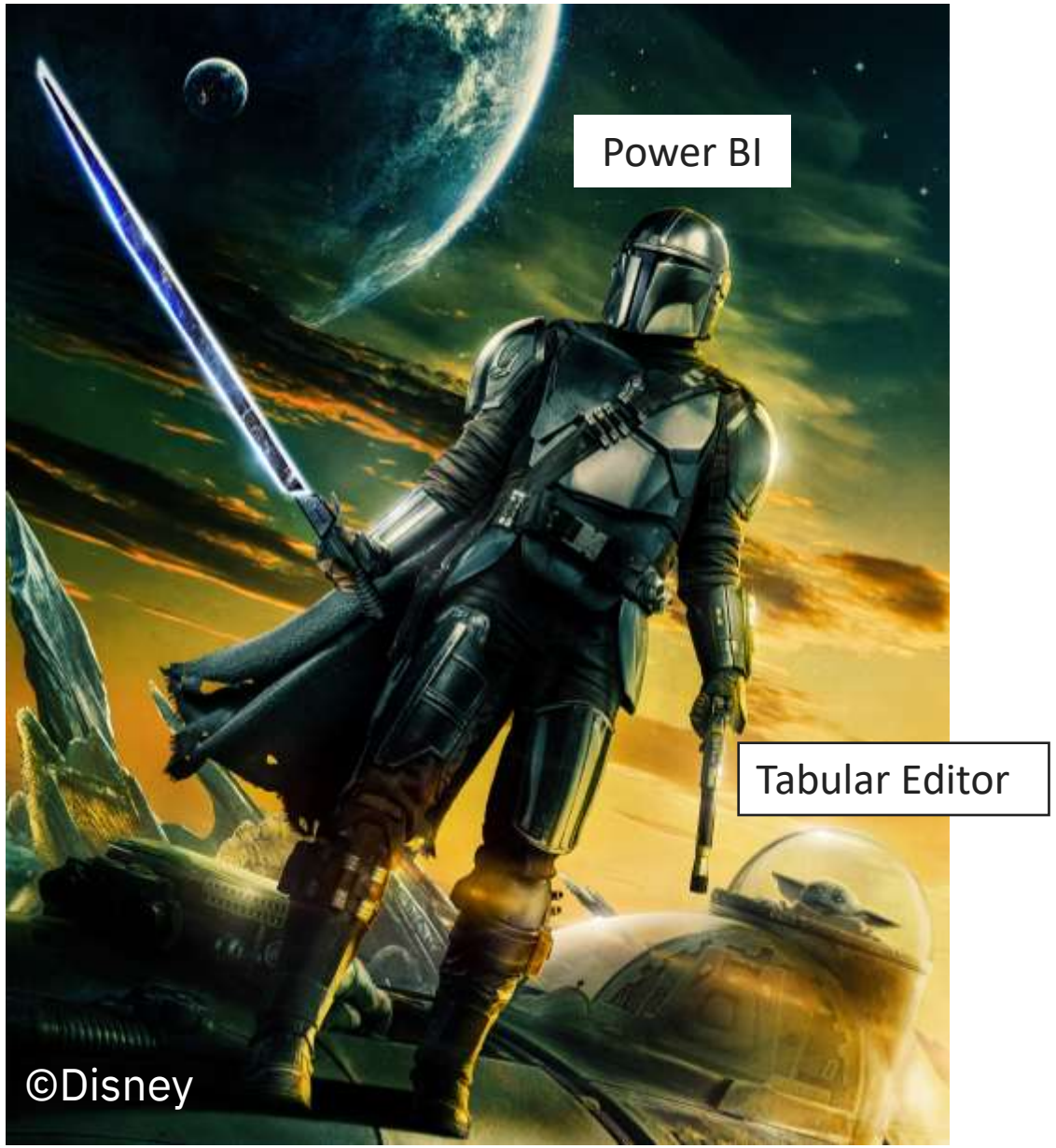
- Queries are saved with the model
- Share or preserve slow query
- First exposure to DAX Queries
- Query View has many uses – i.e. validations

What about that slow DAX Query?



External Tools

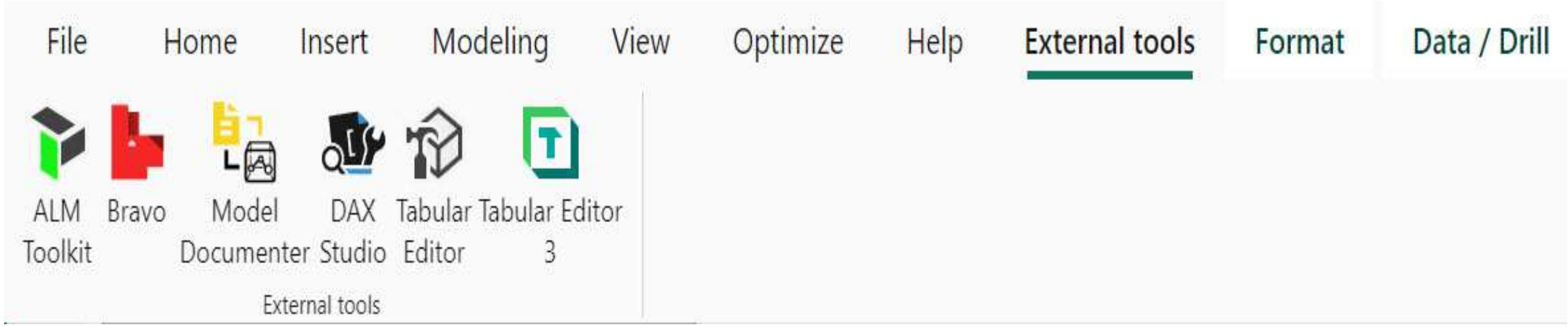
Power BI


Power BI
Tabular Editor
©Disney

Installation

- Full Install
- Power BI Desktop
  - External Tools Tab

External Tools Tab




Tabular Editor
DAX Studio
Power BI
Bravo
©Disney

What if the DAX Query is Slow?

Our Journey

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

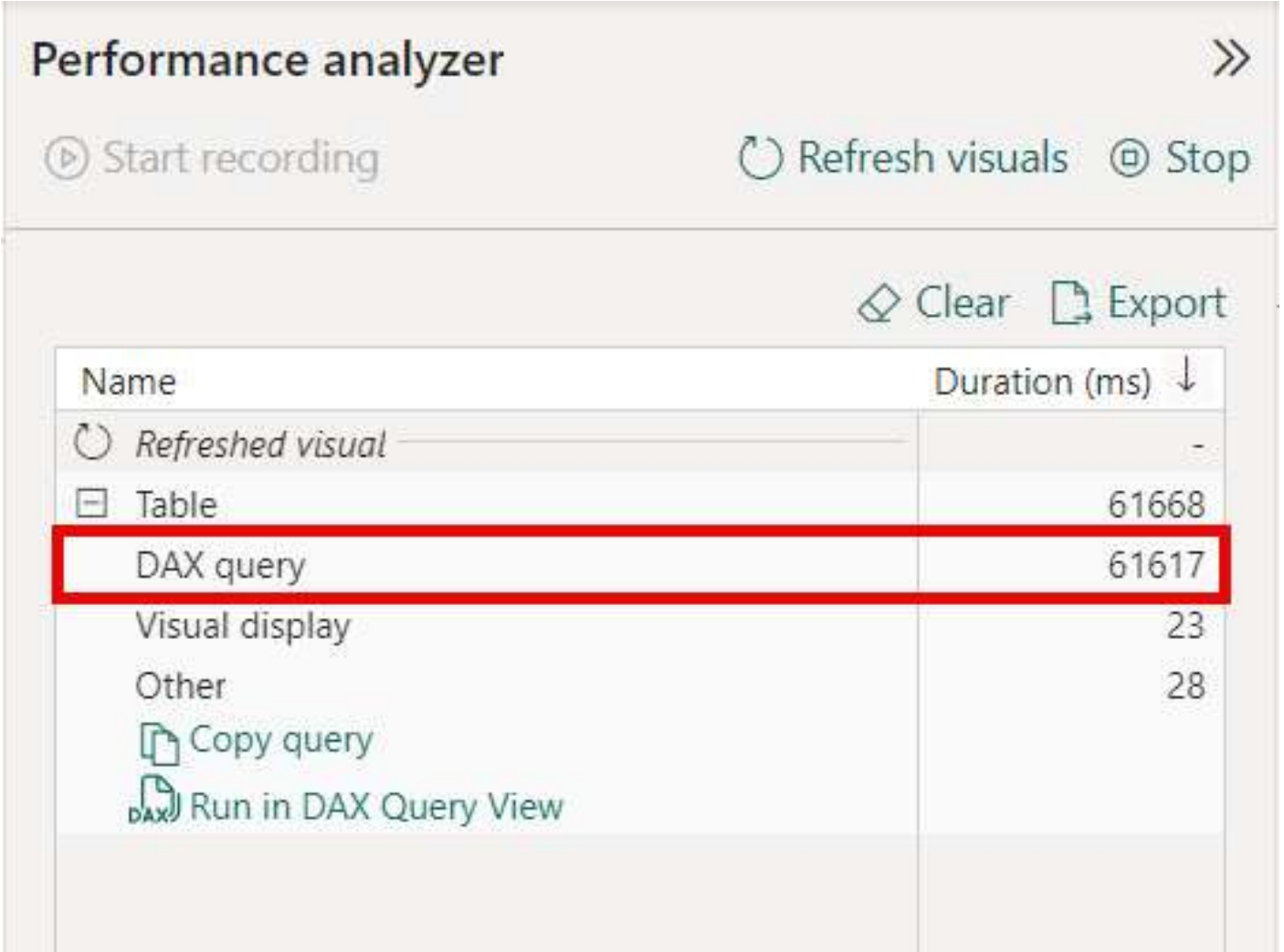
DAX Studio

- Query Plans
  - Logical
  - Physical

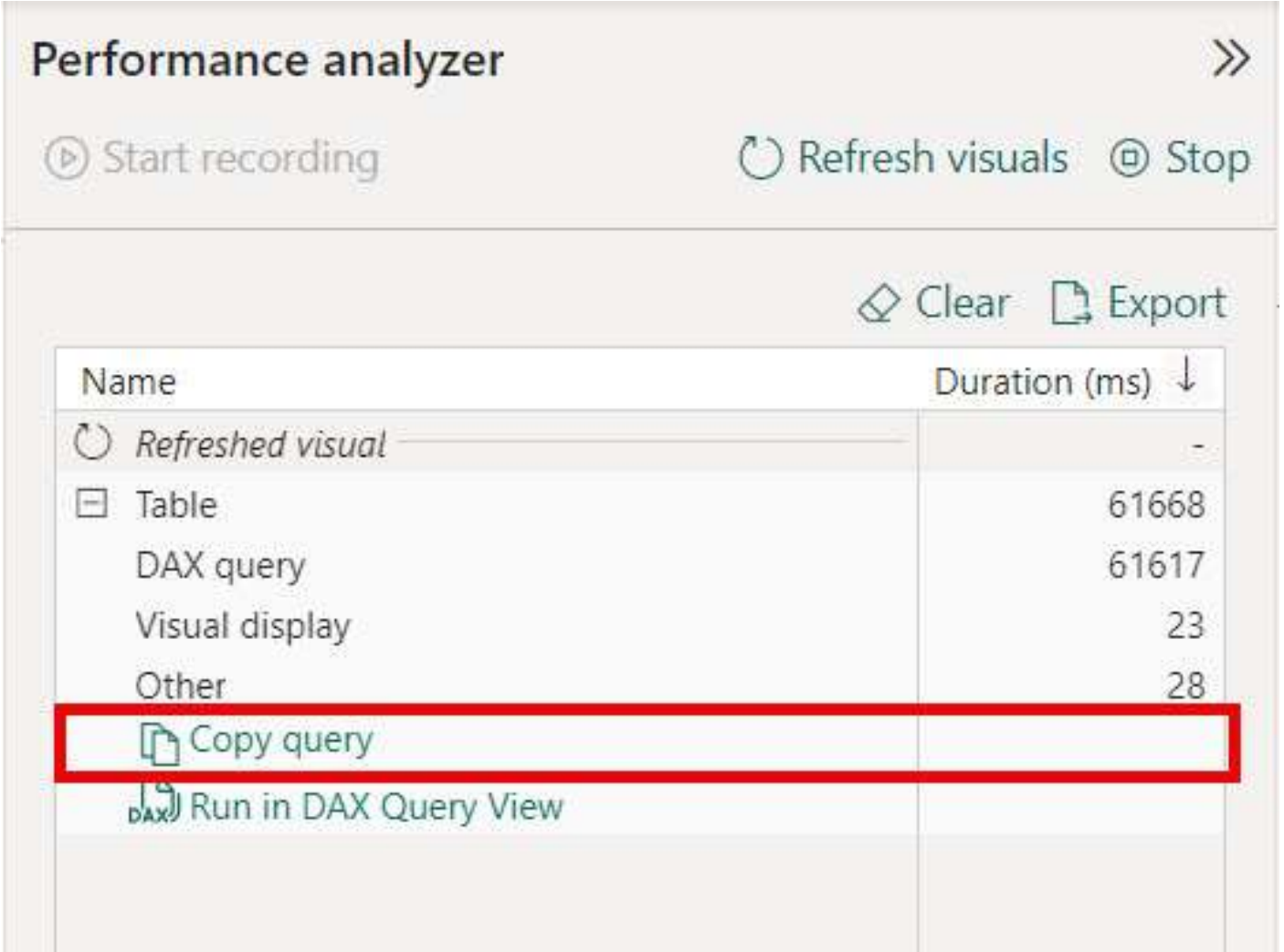
DAX Studio

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

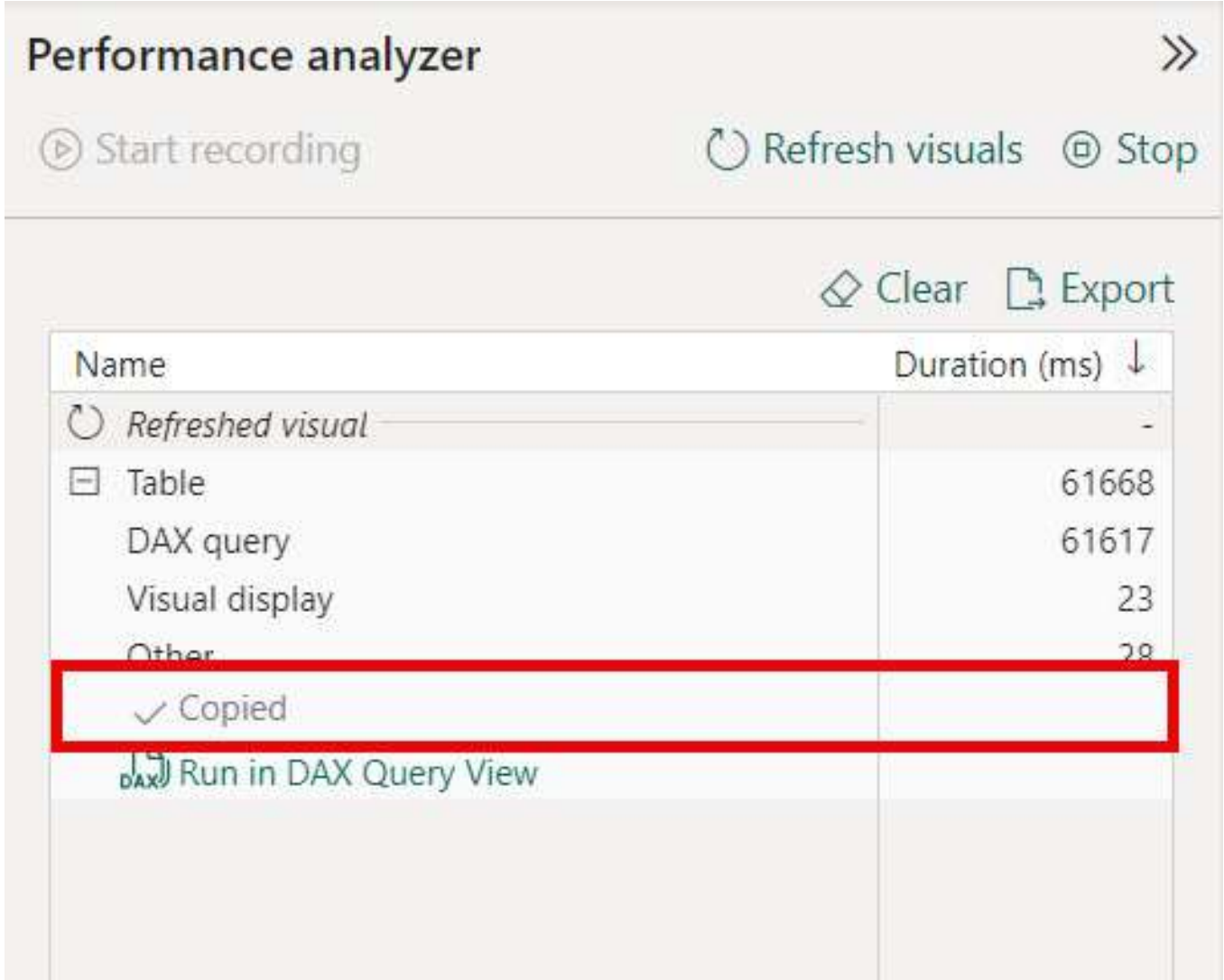
Slow DAX Query



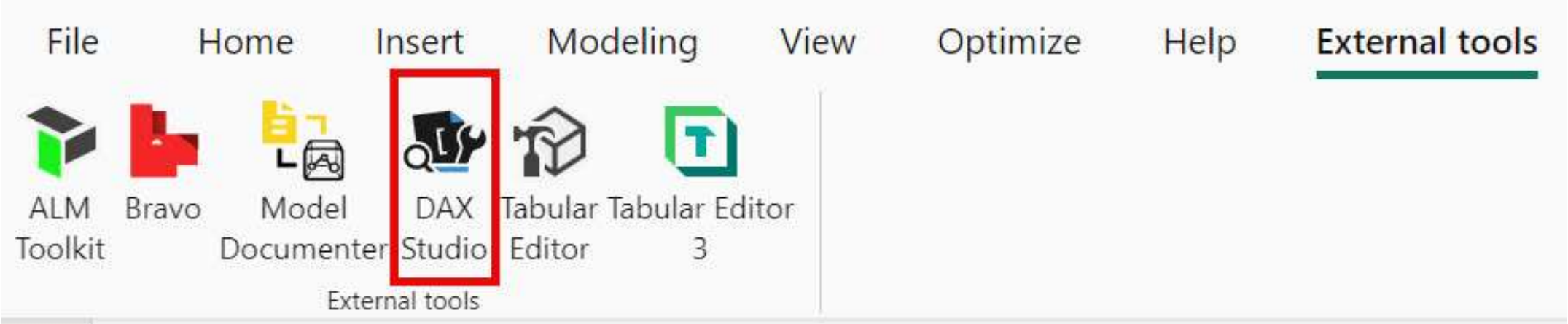
Copy query


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

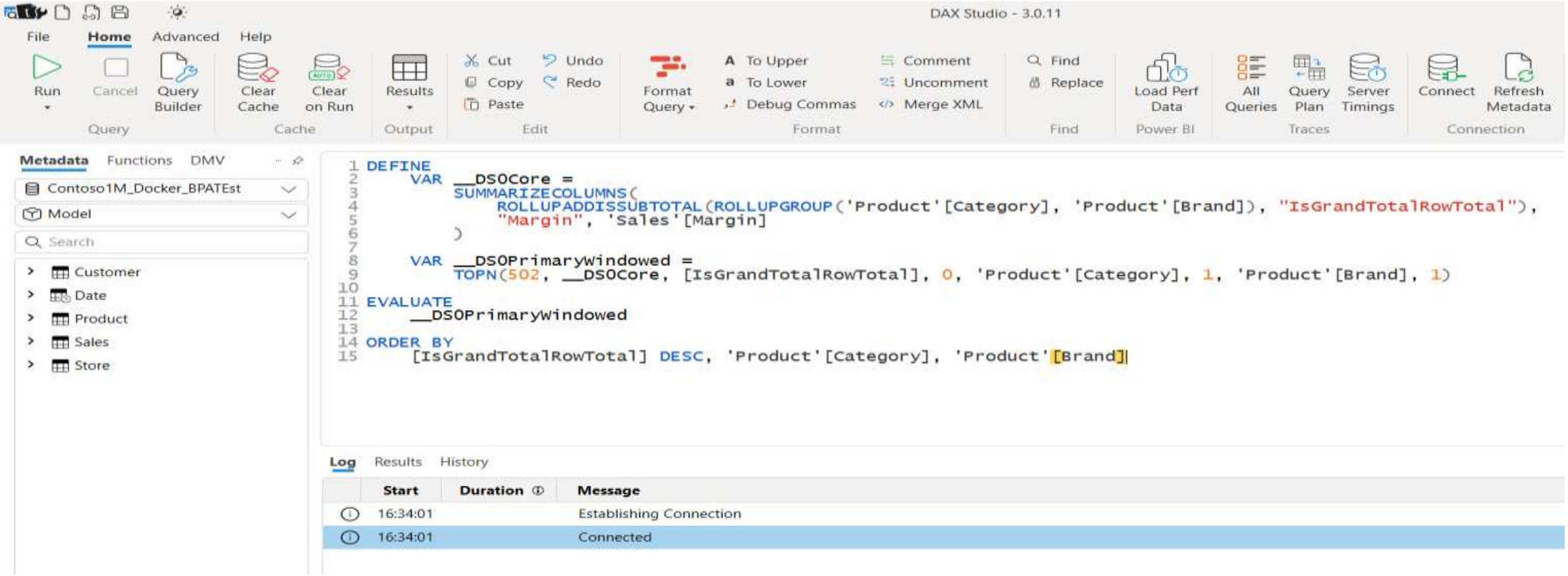
Copied


Performance analyzer
Start recording
Refresh visuals
Stop
Clear
Export
Name
Duration (ms)
Refreshed visual
-
Table
61668
DAX query
61617
Visual display
23
Other
28
Copied
Run in DAX Query View

Power BI – Open DAX Studio



Paste in DAX Studio



Let’s Run It

DAX Studio - 3.0.11

File | Home | Advanced | Help

Run | Cancel | Query Builder | Clear Cache | Clear on Run | Results | Cut | Copy | Paste | Undo | Redo | Format Query | To Upper | To Lower | Debug Commas | Comment | Uncomment | Merge XML | Find | Replace | Load Perf Data | All Queries | Query Plan | Server Timings | Connect | Refresh Metadata

Query | Cache | Output | Edit | Format | Find | Power BI | Traces | Connection

Metadata | Functions | DMV

Contoso1M_Docker_BPATEst

Model

Search

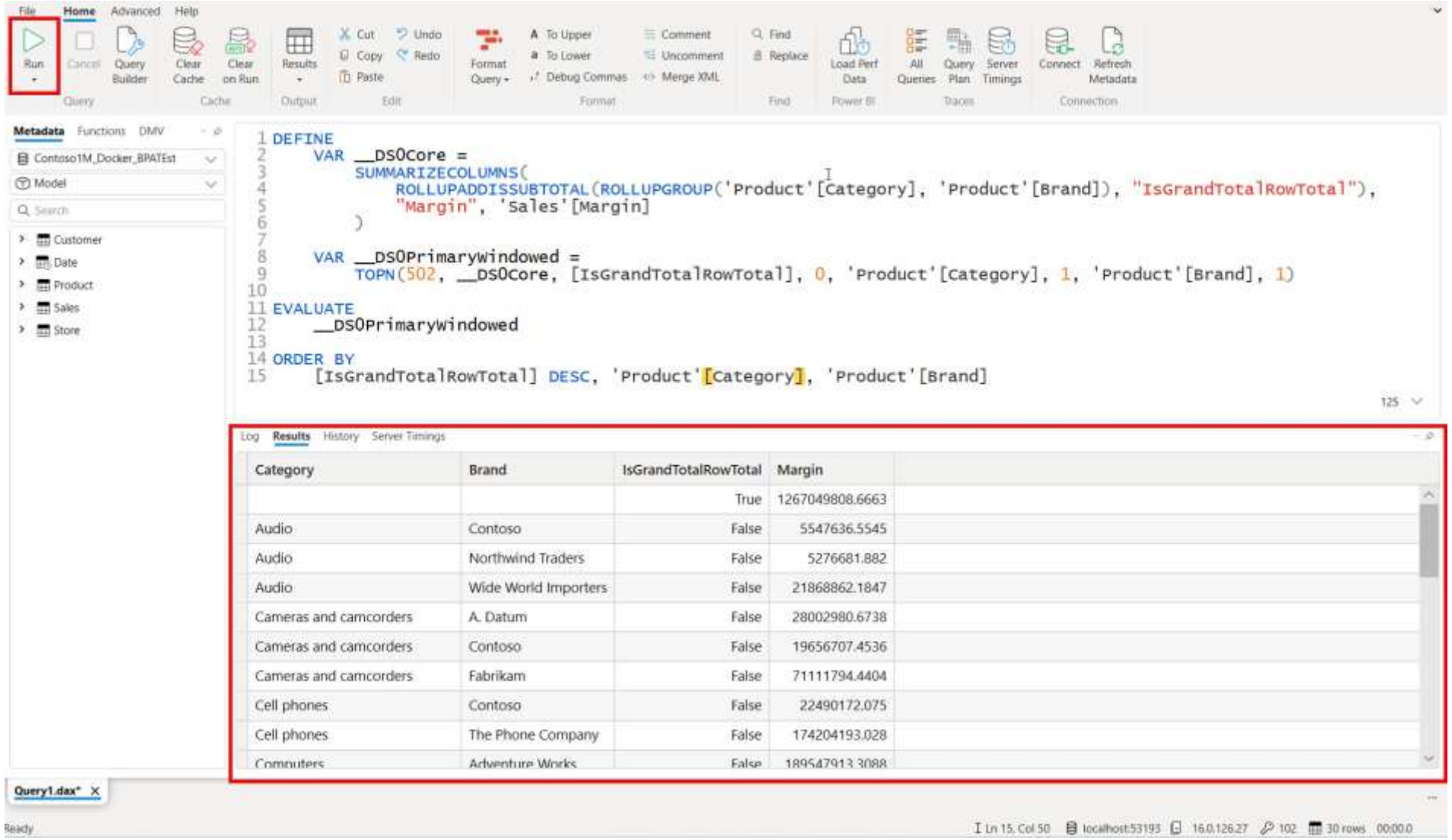
- Customer
- Date
- Product
- Sales
- Store

```
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]
```

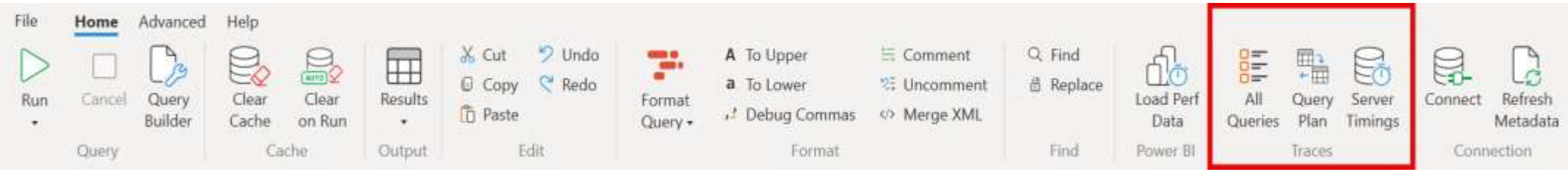
Log | Results | History

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

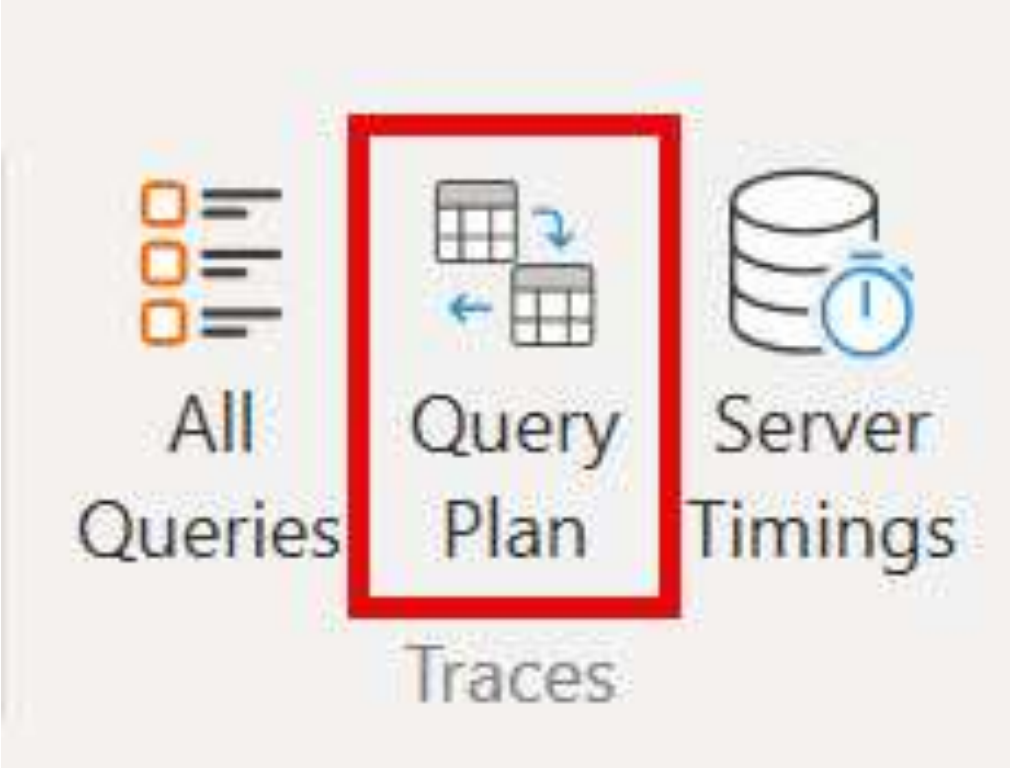
Results – Like Query View



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

Query Plan



DAX Studio – Query Plan

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 #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

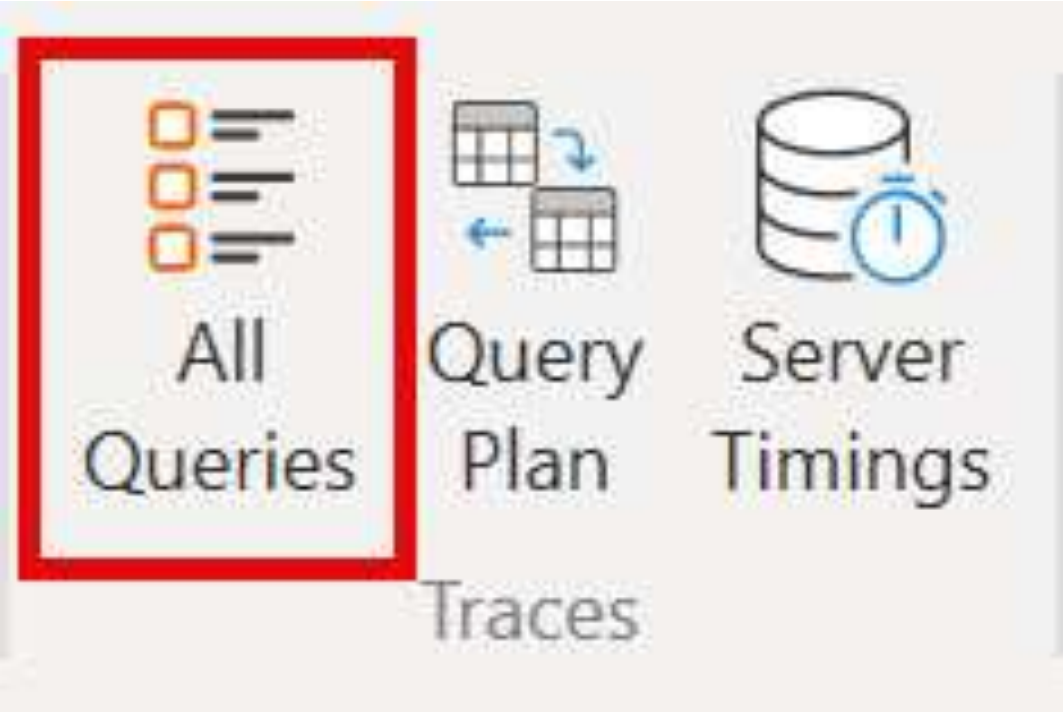
Log Results History Server Timings **Query Plan**

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 #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 – All Queries



Like SQL Profiler

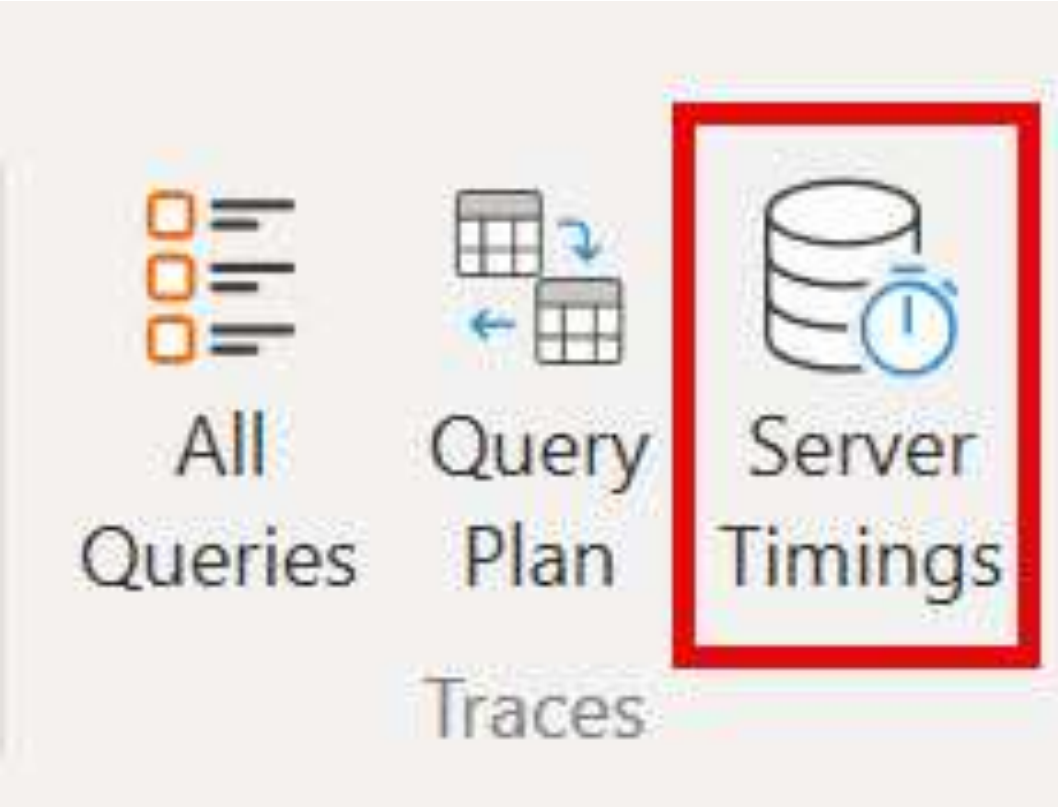
DAX Studio – All Queries

Log Results History **All Queries**

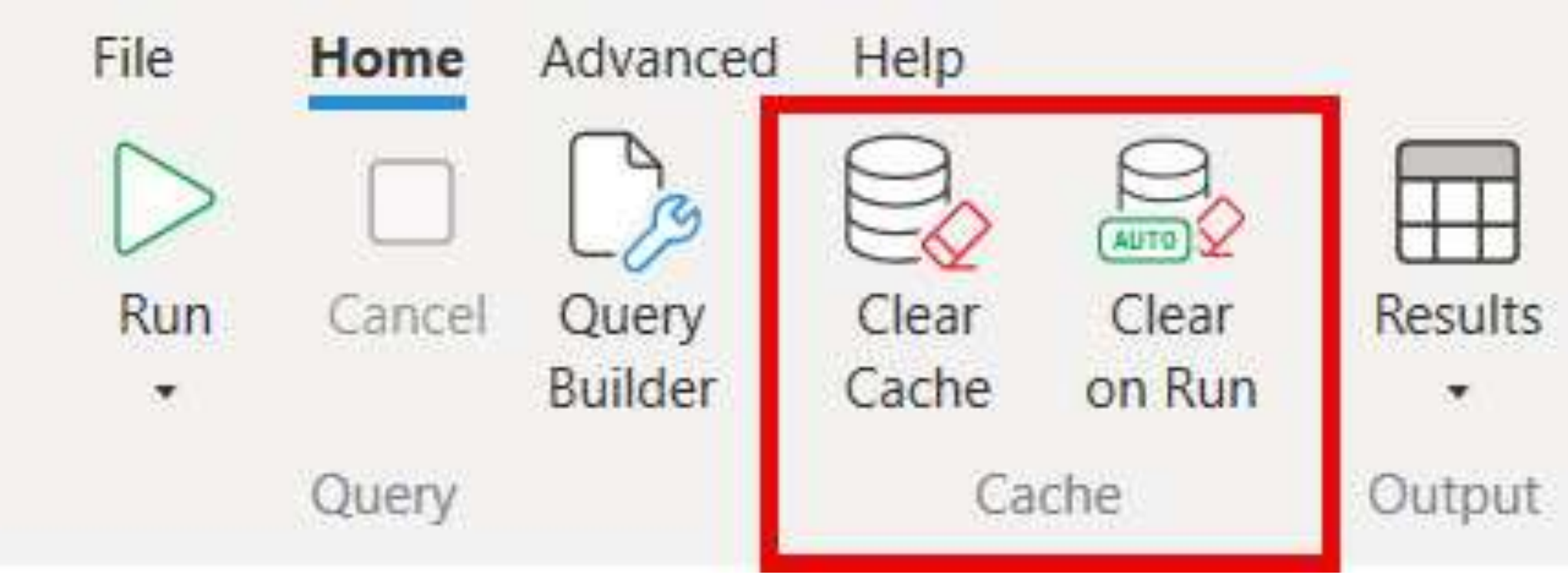
Record Pause Stop Clear Copy Export Info Filters

StartTime	Type	Duration	User	Database	Query
10:12:54	DAX	10ms	MSI\roman	Contoso1M_Docker_BPATEst	DEFINE VAR __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...

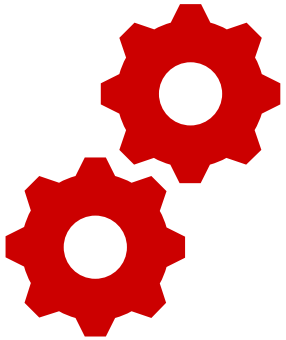
DAX Studio – Server Timings



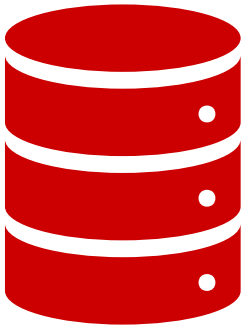
Cache


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

Two Engines



Formula Engine



Storage Engine

Formula Engine

- Does not cache
- Single threaded

Storage Engine

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

DAX

```
EVALUATE
'Product'
```

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

Metadata Functions DMV

Contoso1M_Docker_BPATEst

Model

Search

- Customer
- Date
- Product
- Sales
- Store

```
// 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]
```

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 Studio – Server Timings – Bad

Total	SE CPU
84,541 ms	151,586 ms x1.9
FE	**SE**
2,768 ms 3.3%	81,773 ms 96.7%
SE Queries	**SE Cache**
2,576	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	**SE**
5 ms 2.8%	175 ms 97.2%
SE Queries	**SE Cache**
2	0 0.0%

DAX Studio – Most Formula Engine

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

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

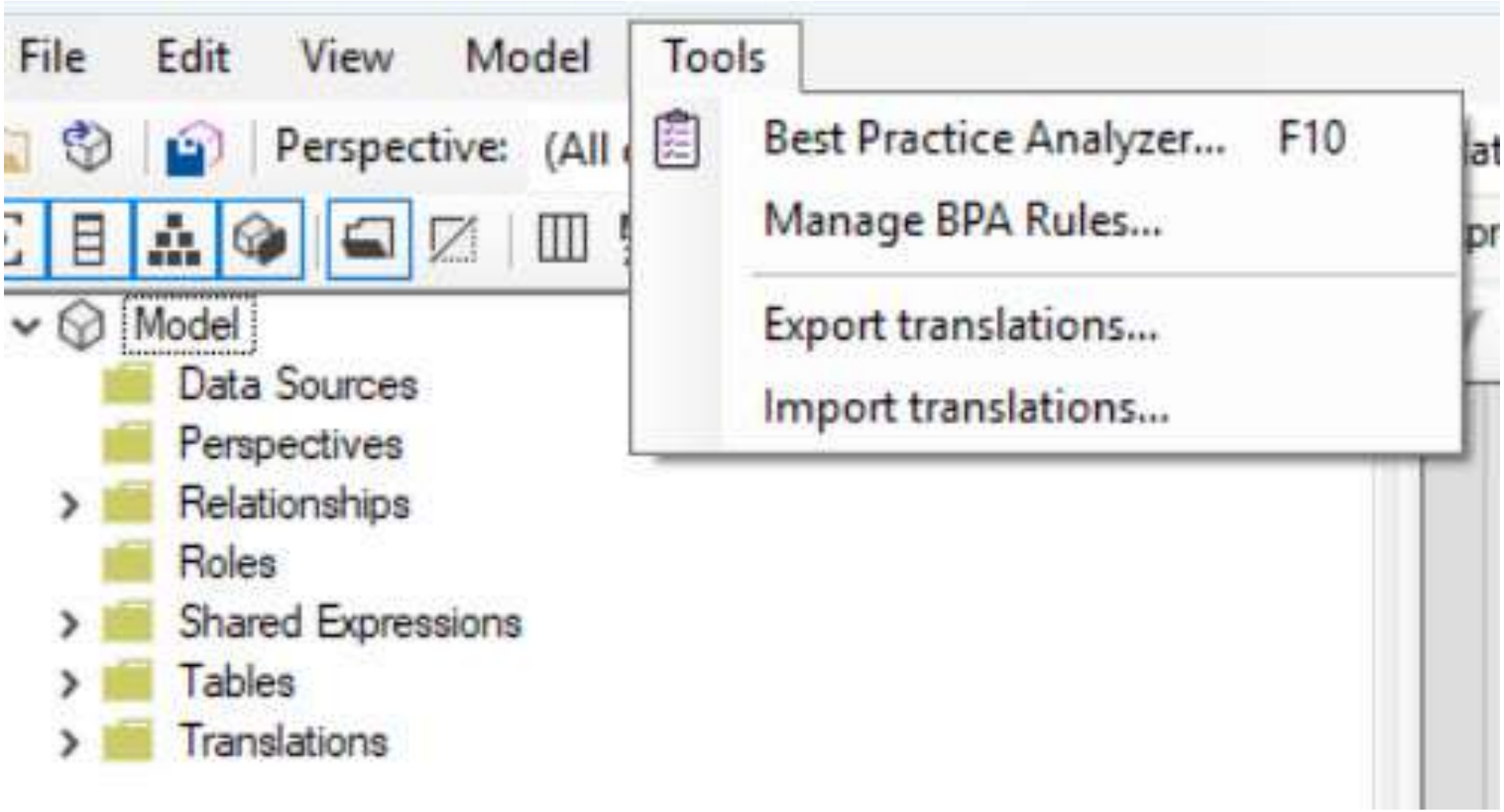
- Paid Version
- Extra Features
  - DAX Debugger

Tabular Editor 2

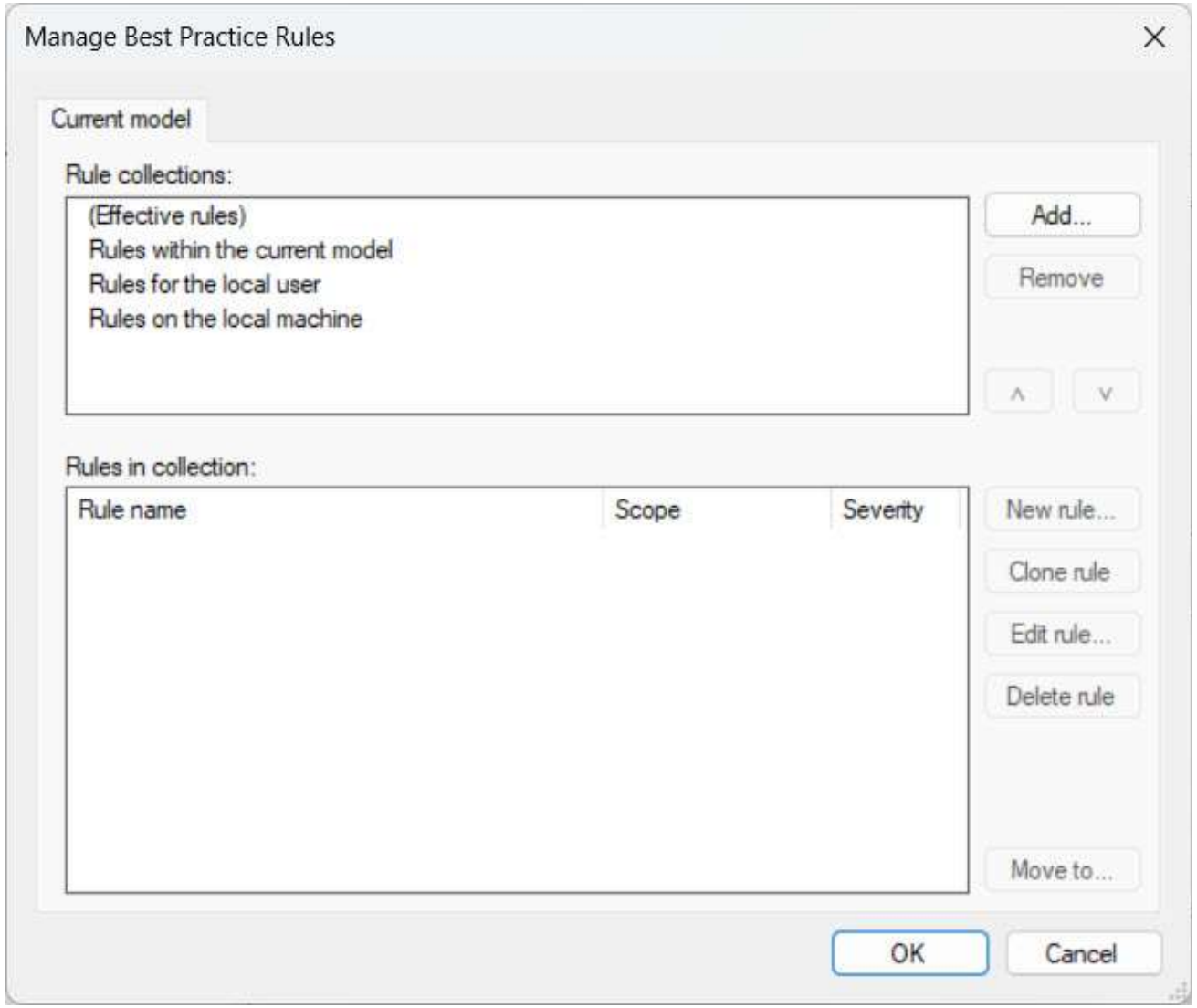
Optimize a semantic model by using Tabular Editor 2

Best Practice Analyzer

Best Practice Analyzer Location



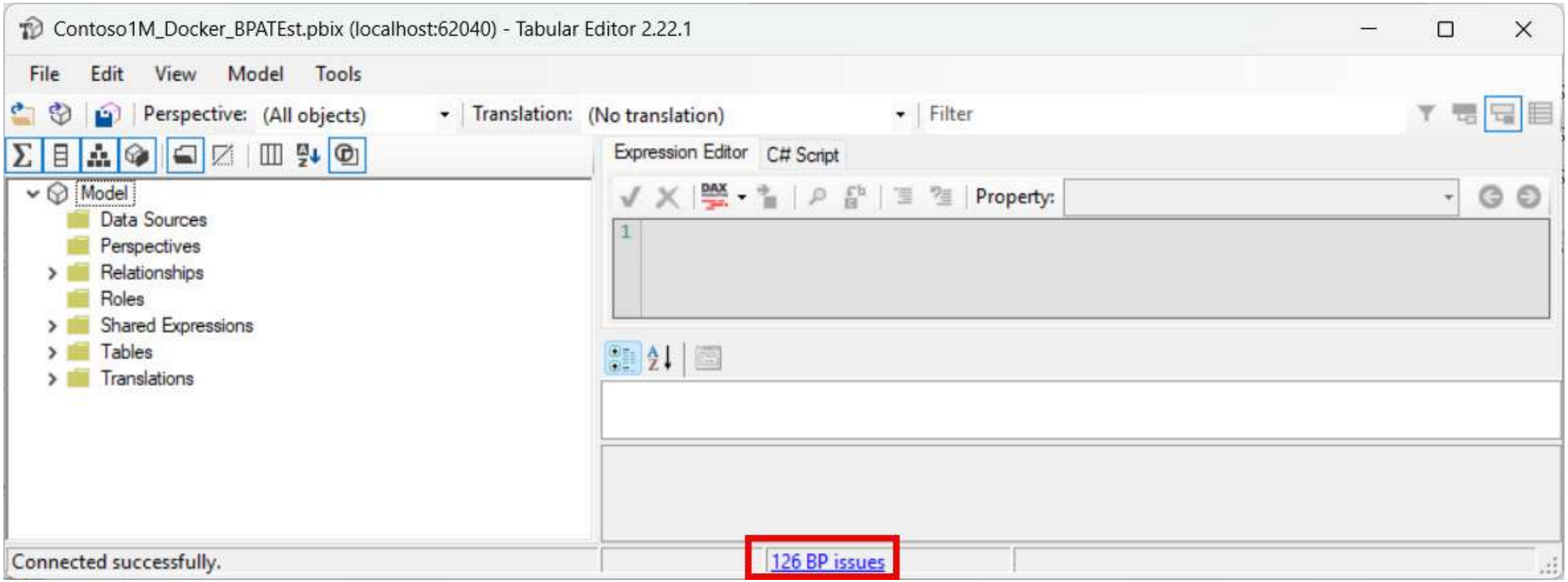
Add and Manage Rules



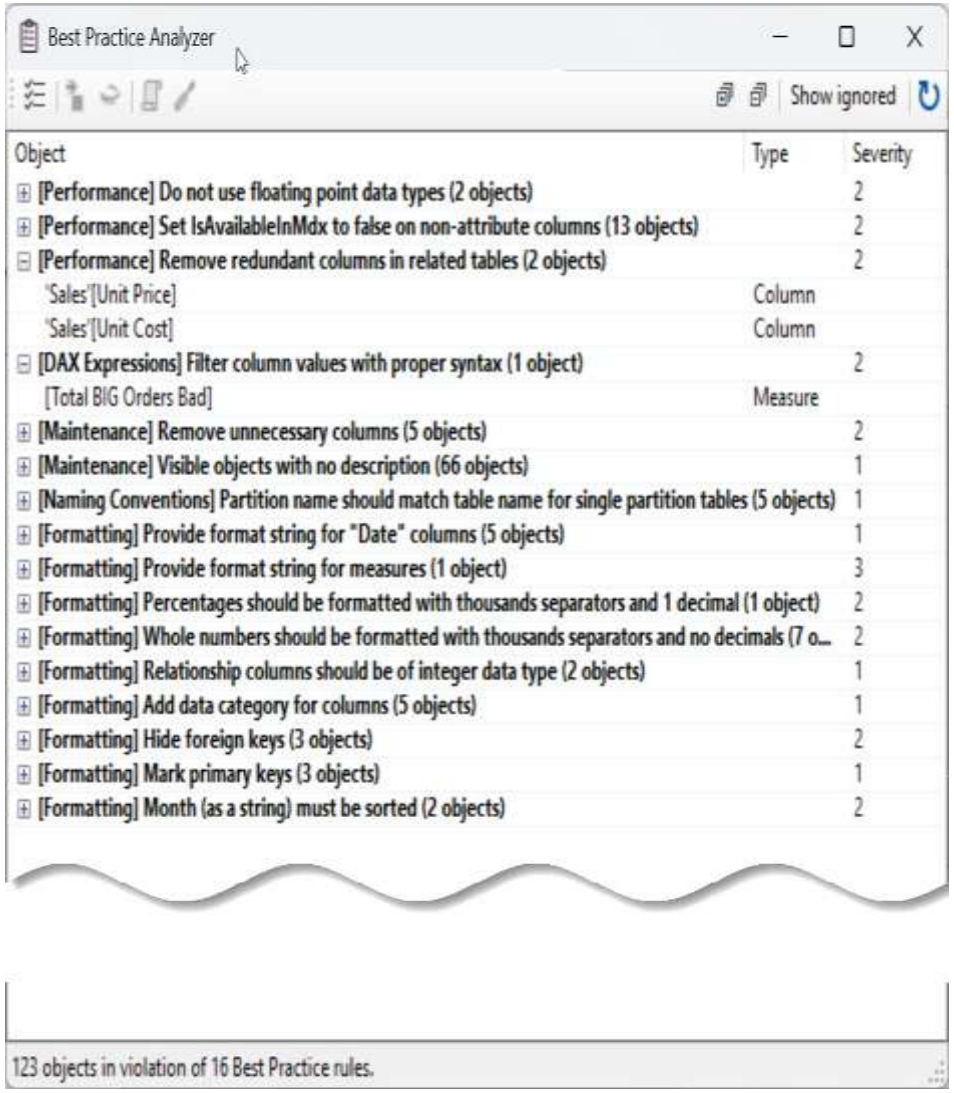
Best Practice Analyzer – C# Script Download

```
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



BPA – Rule Exceptions for model



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)

OK Cancel

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

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 Editor
- Learn DAX

Resources

[Tabular Editor – Wonderful Training Free]

[Tabular Editor – Blog Posts]

[DAX Studio]

[Tabular Editor 2.x]

Resources

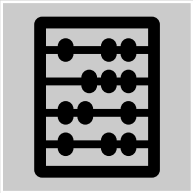
Data Goblins - Sample Datasets

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

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