Power up your Fabric Development with DAX Studio and Tabular Editor

Data Saturday Dallas

September 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

Shoulders of Giants

DATA SATURDAY DALLAS 2024 SPONSOR

GOLD

TIMEXTENDER

vmware by Broadcom

SILVER

SAPCG

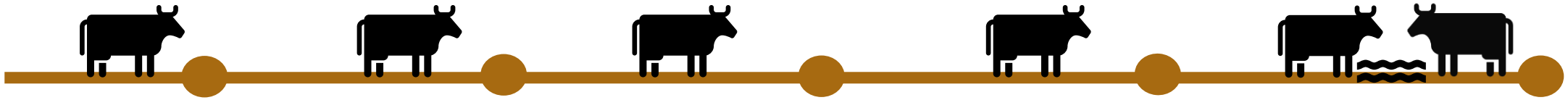
DCAC

silk

Quest

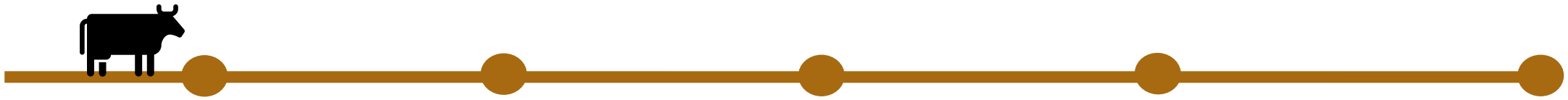
COZYROC

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
    - 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**

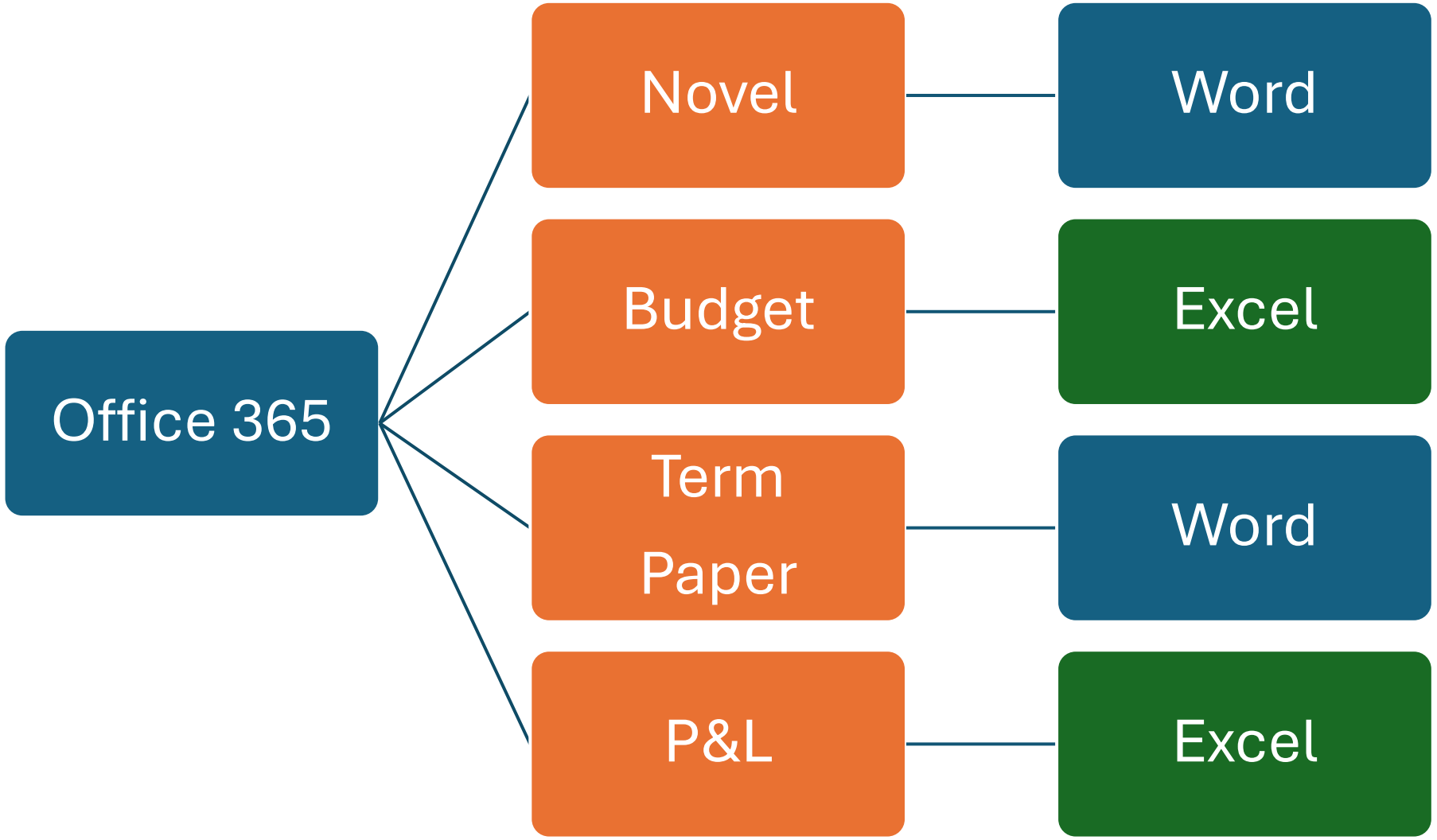
Optimize enterprise-scale semantic models

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

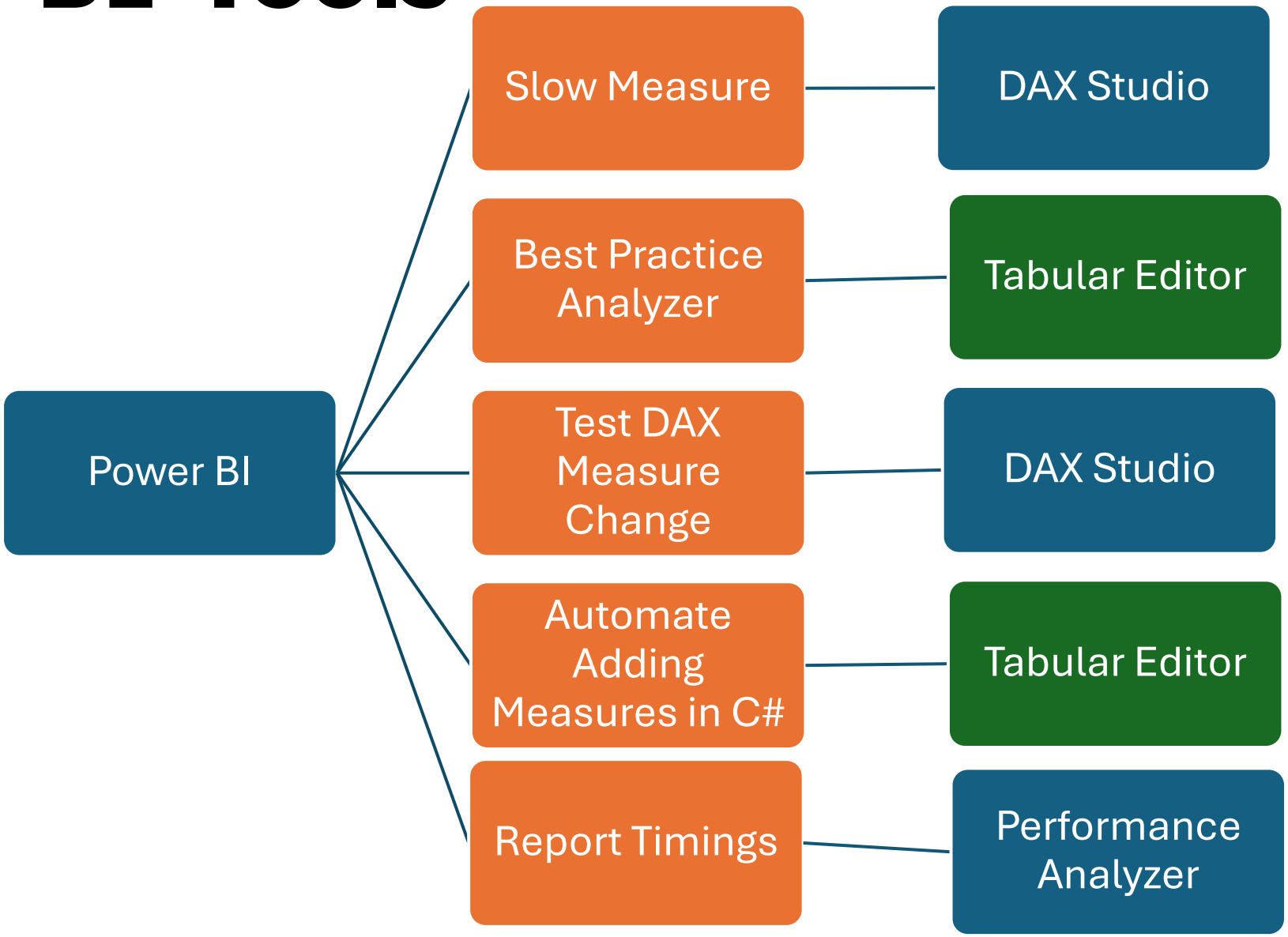
Why is my report slow?

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

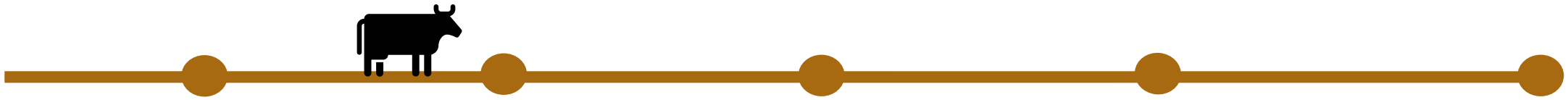
Office 365



Power BI Tools



Our Journey

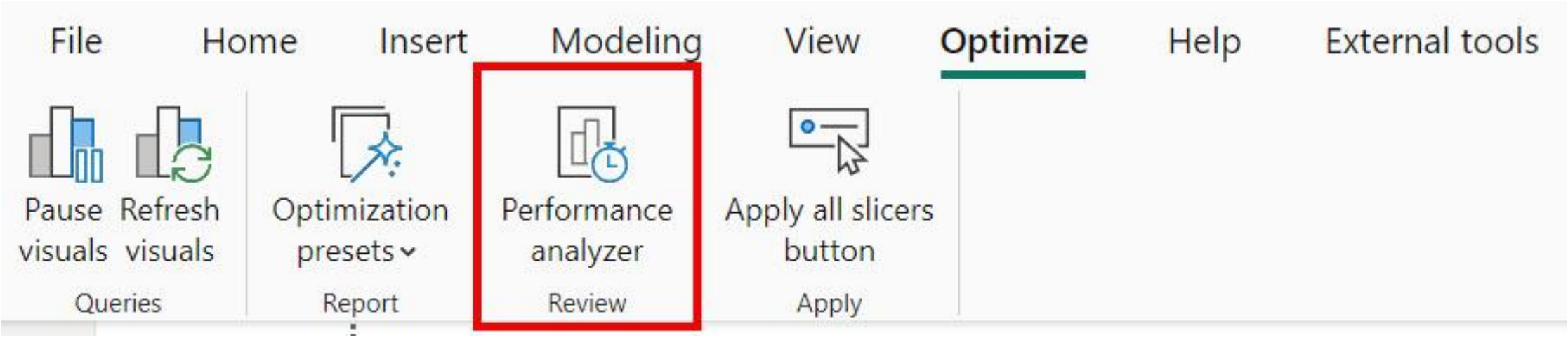


1. Intro
2. **Performance Analyzer**
3. DAX Studio
4. Tabular Editor
5. Conclusion

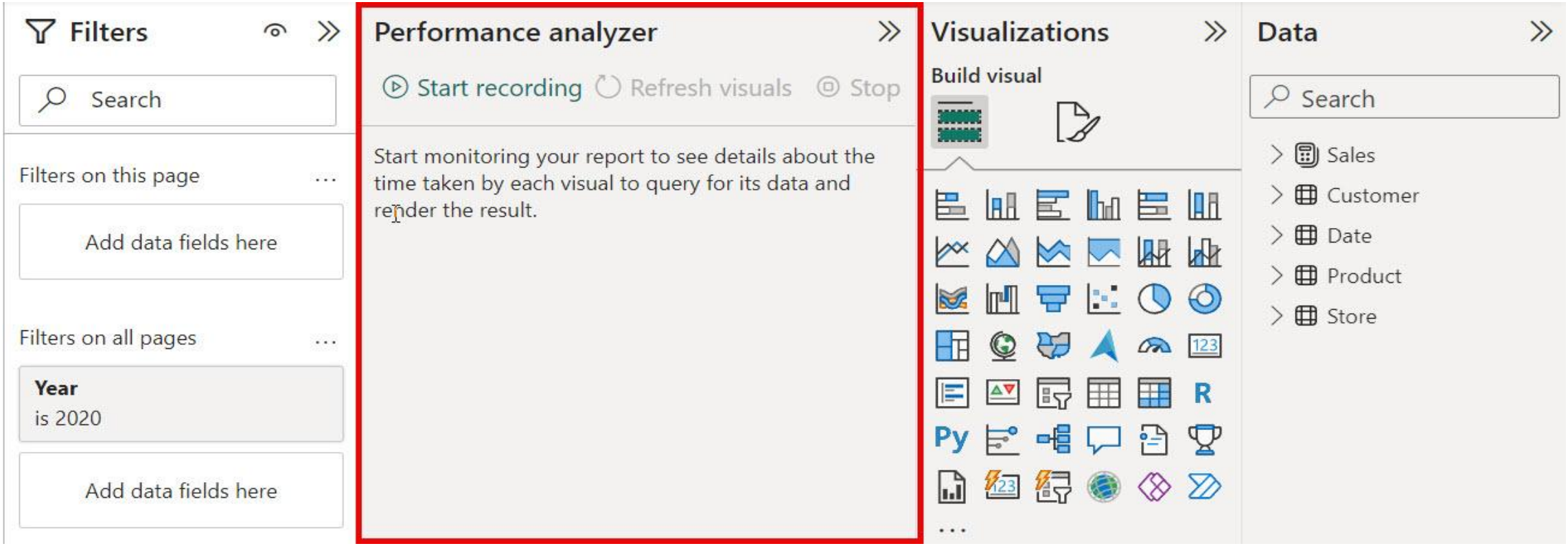
Performance Analyzer

- Not an External Tool
- Built into Power BI Desktop

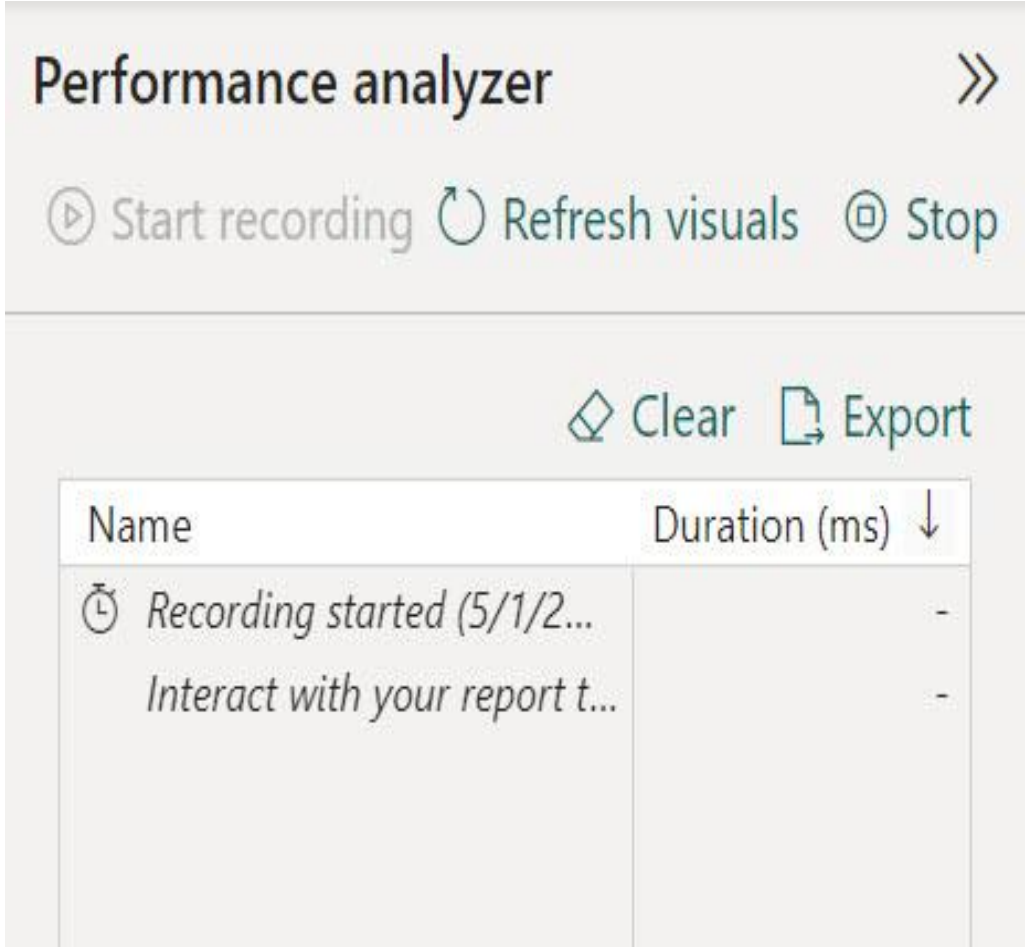
Performance Analyzer



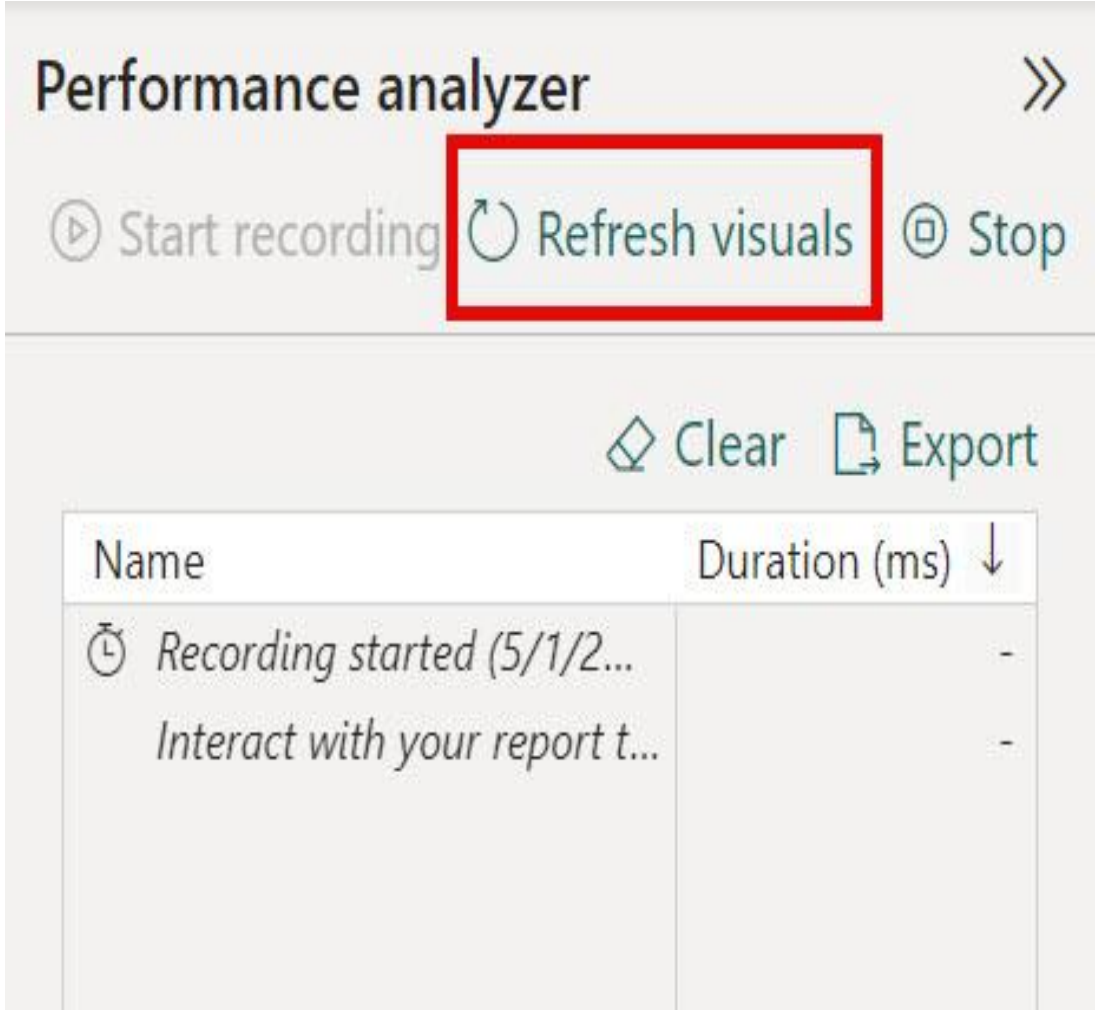
Performance Analyzer
Start Recording



Performance Analyzer Recording



Performance Analyzer Refresh Visuals

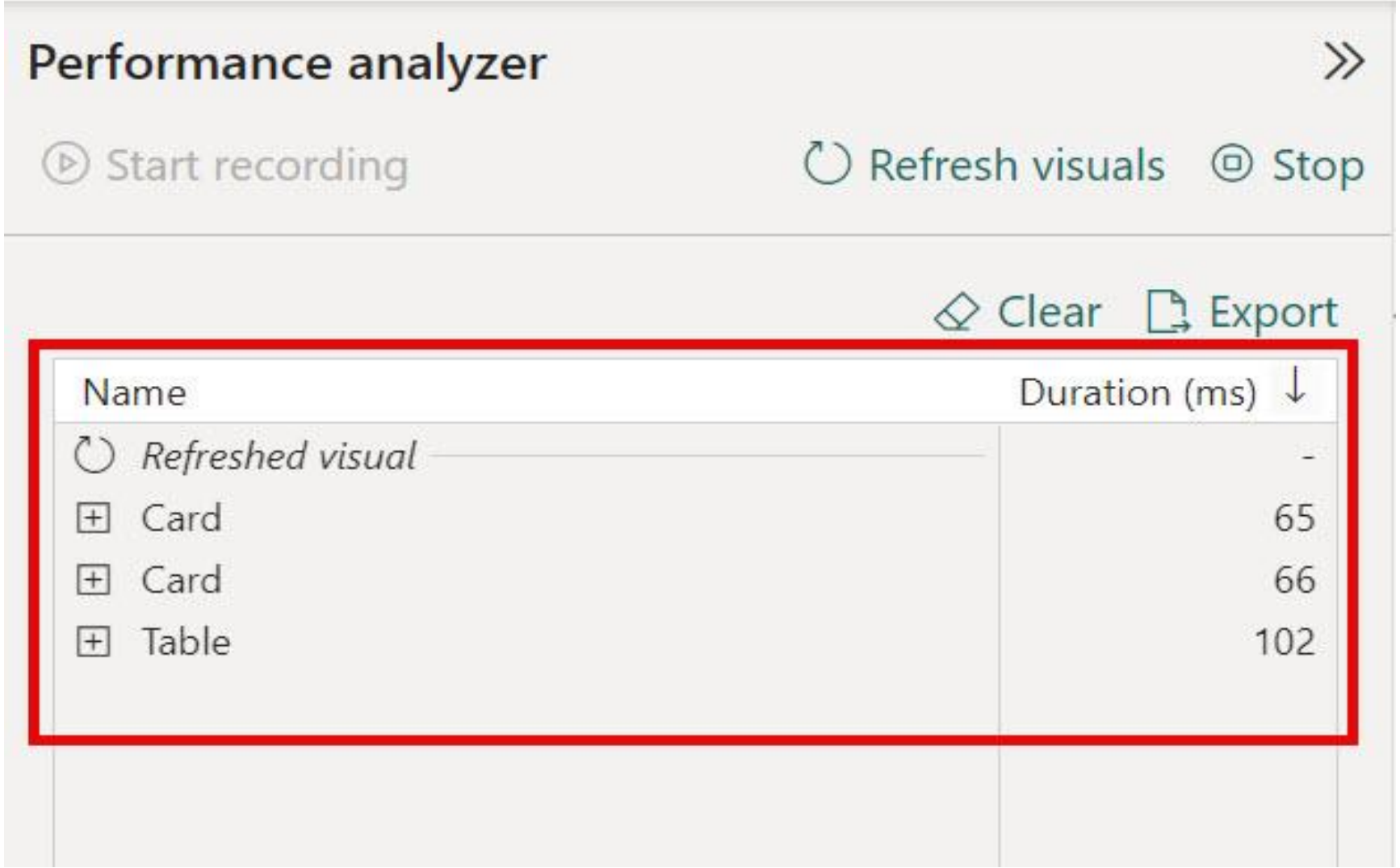


Individual Refresh Button

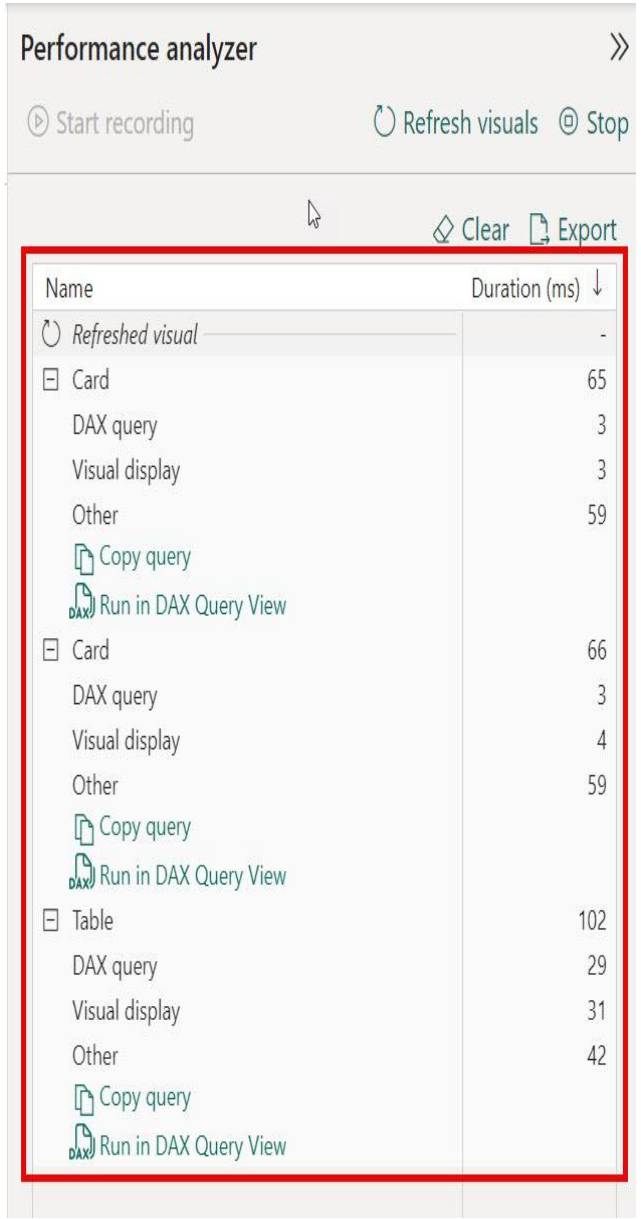
Note: Values for Other will be different

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



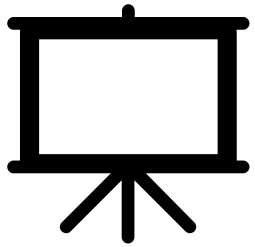
Expanded View



Detail on Card

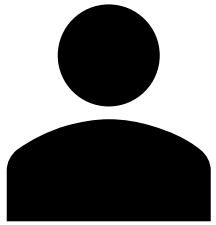
Refreshed visual	-
Card	65
DAX query	3
Visual display	3
Other	59
Copy query	
Run in DAX Query View	

3 Numbers



Visual Display

Other



DAX Query

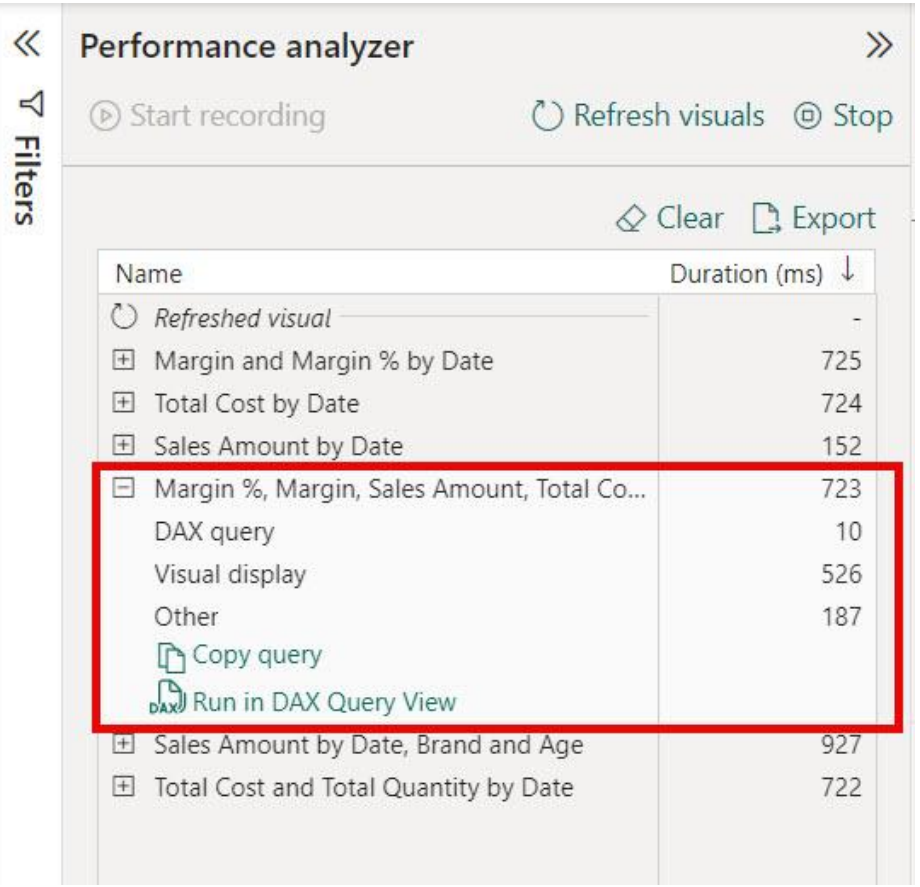
Visual Display

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


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
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
- 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

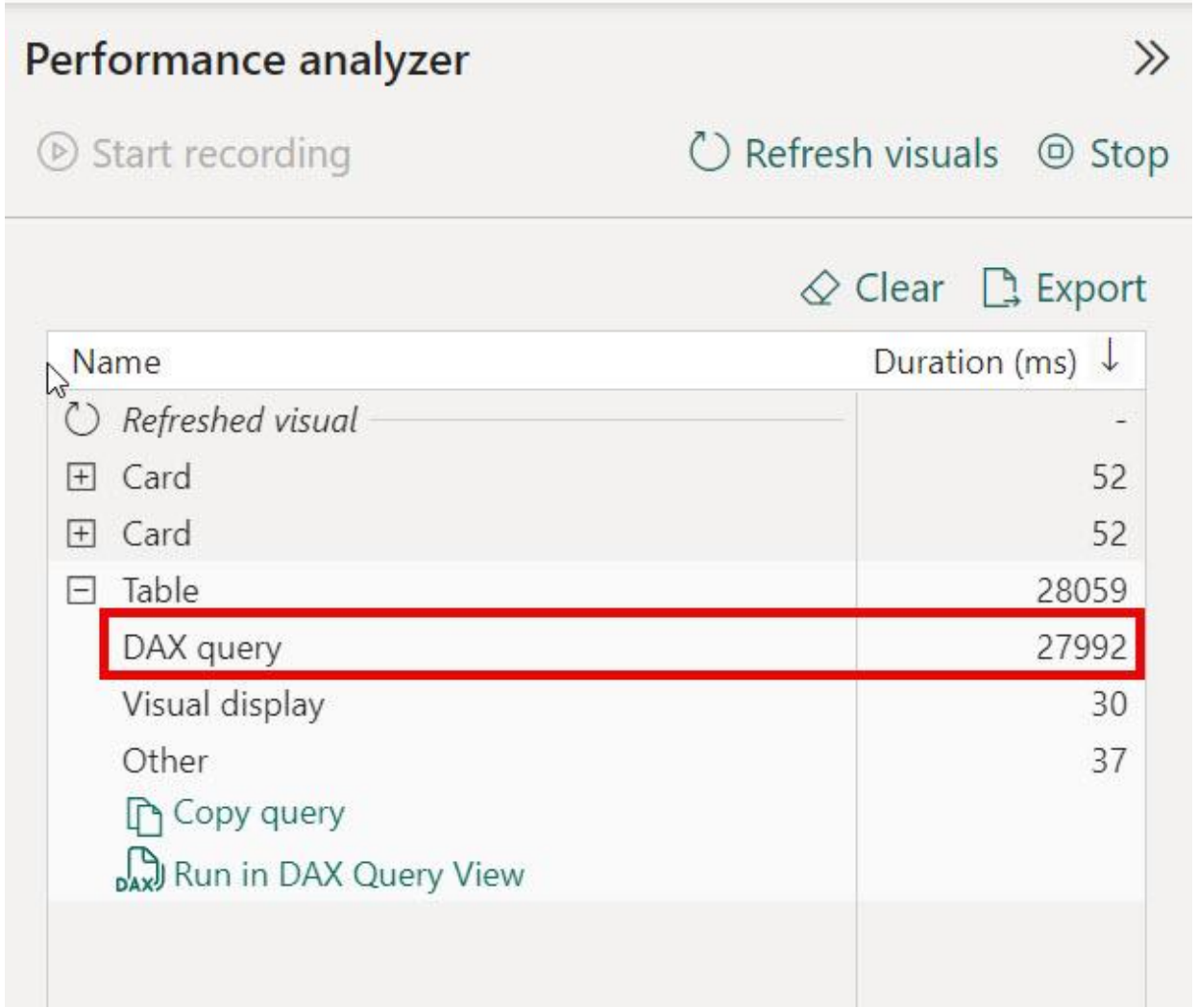
Other - Solutions

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

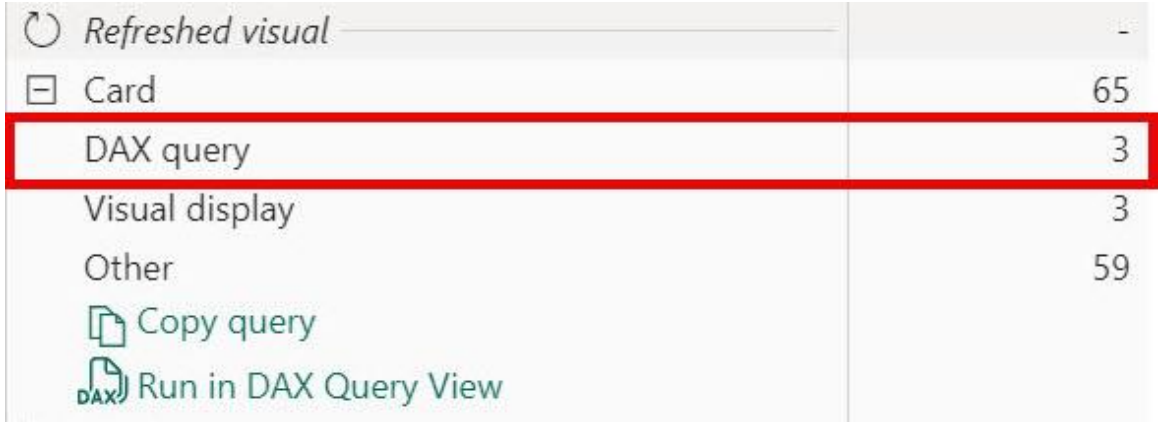
DAX Query

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

Slow DAX Query



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

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
- 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/

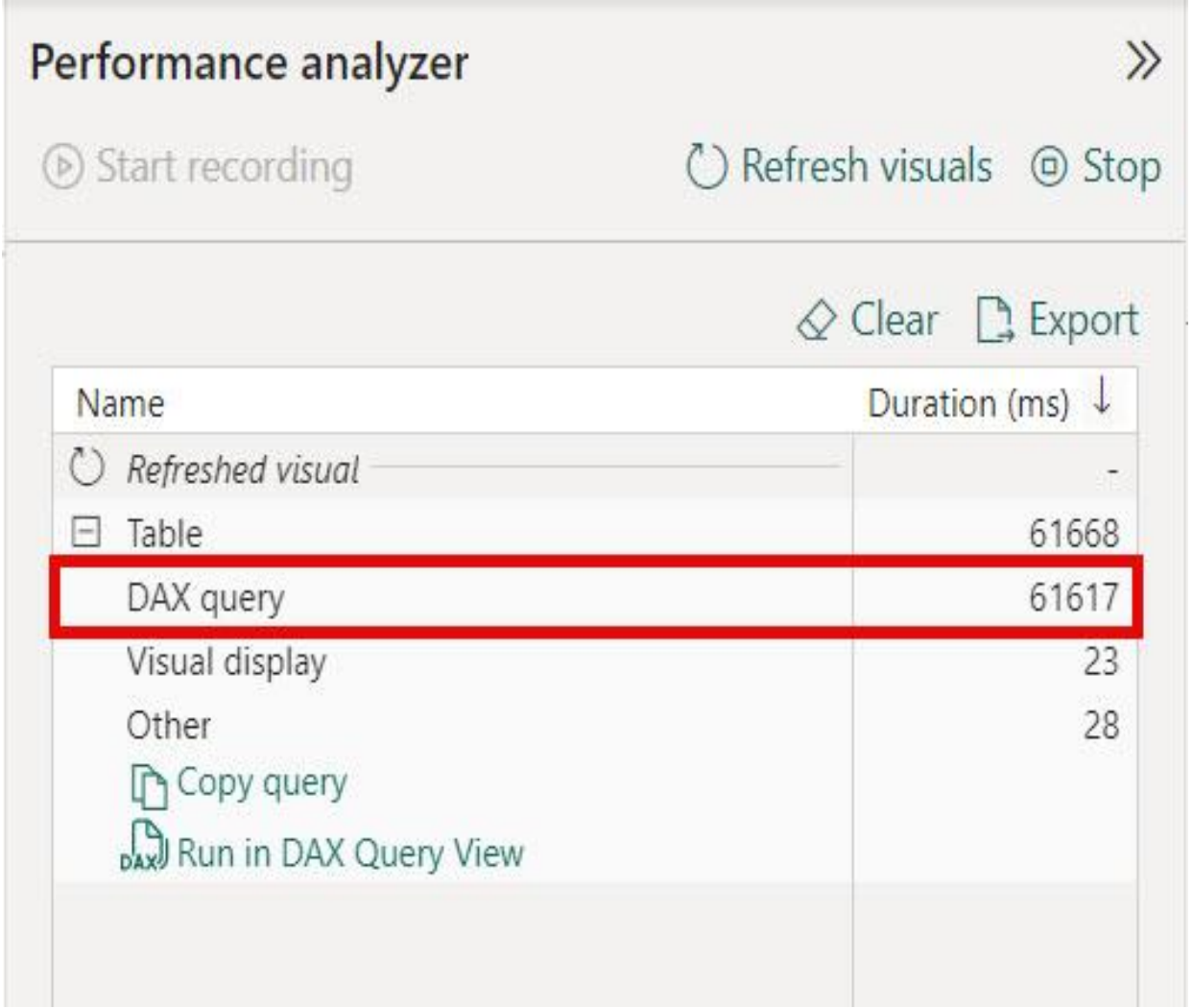
- Use against models in Workspace

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?



DAX

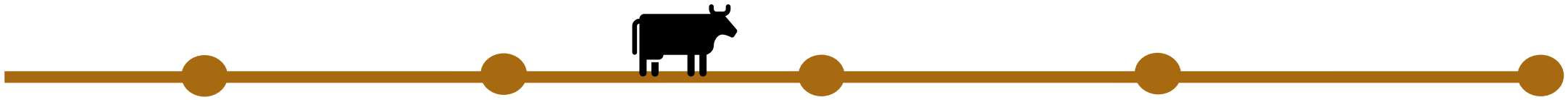
External Tools

Installation

- Full Install
- Power BI Desktop
  - External Tools Tab

External Tools Tab

Our Journey



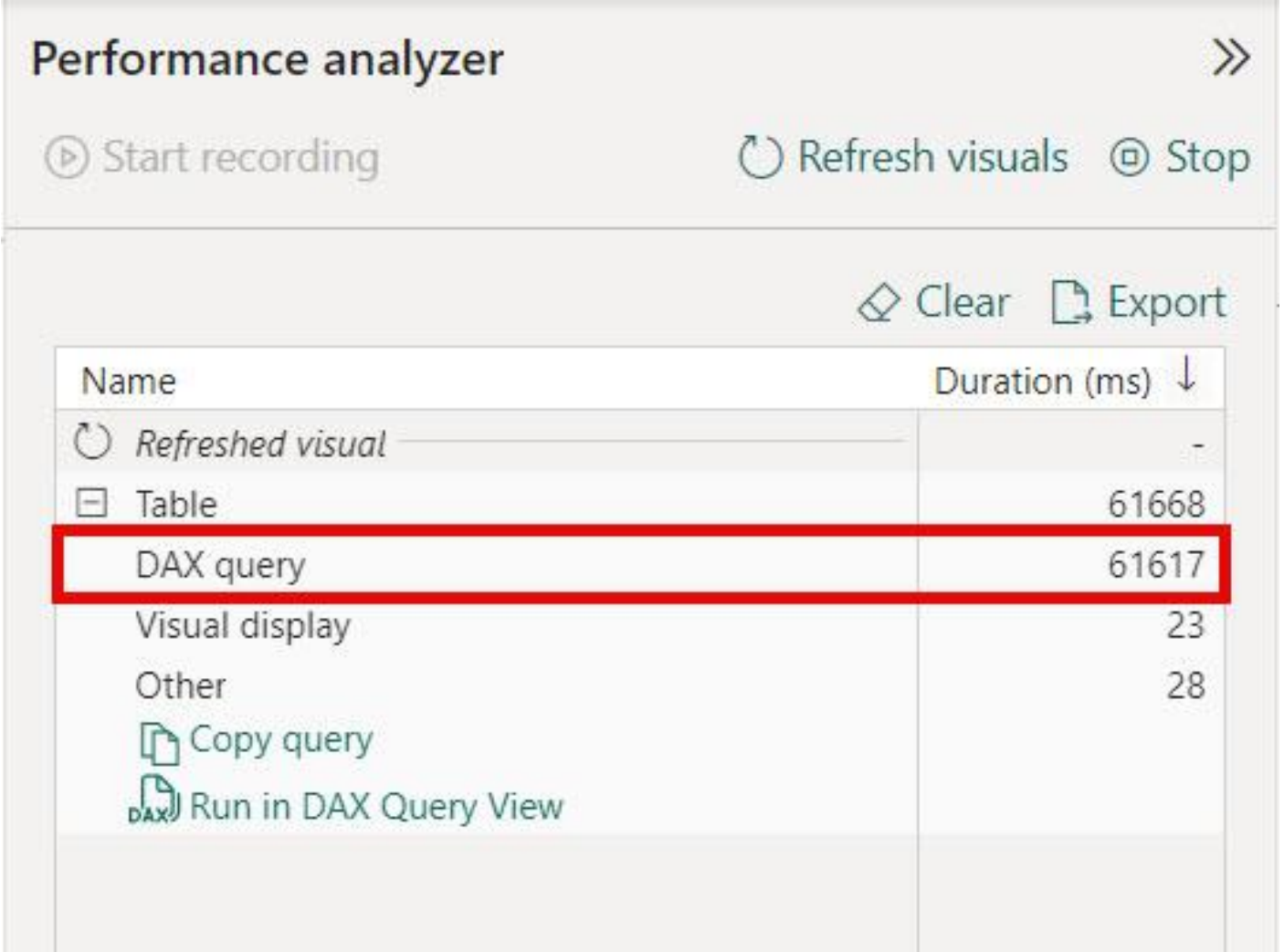
1. Intro
2. Performance Analyzer
3. **DAX Studio**
4. Tabular Editor
5. Conclusion

What if the DAX Query is Slow?

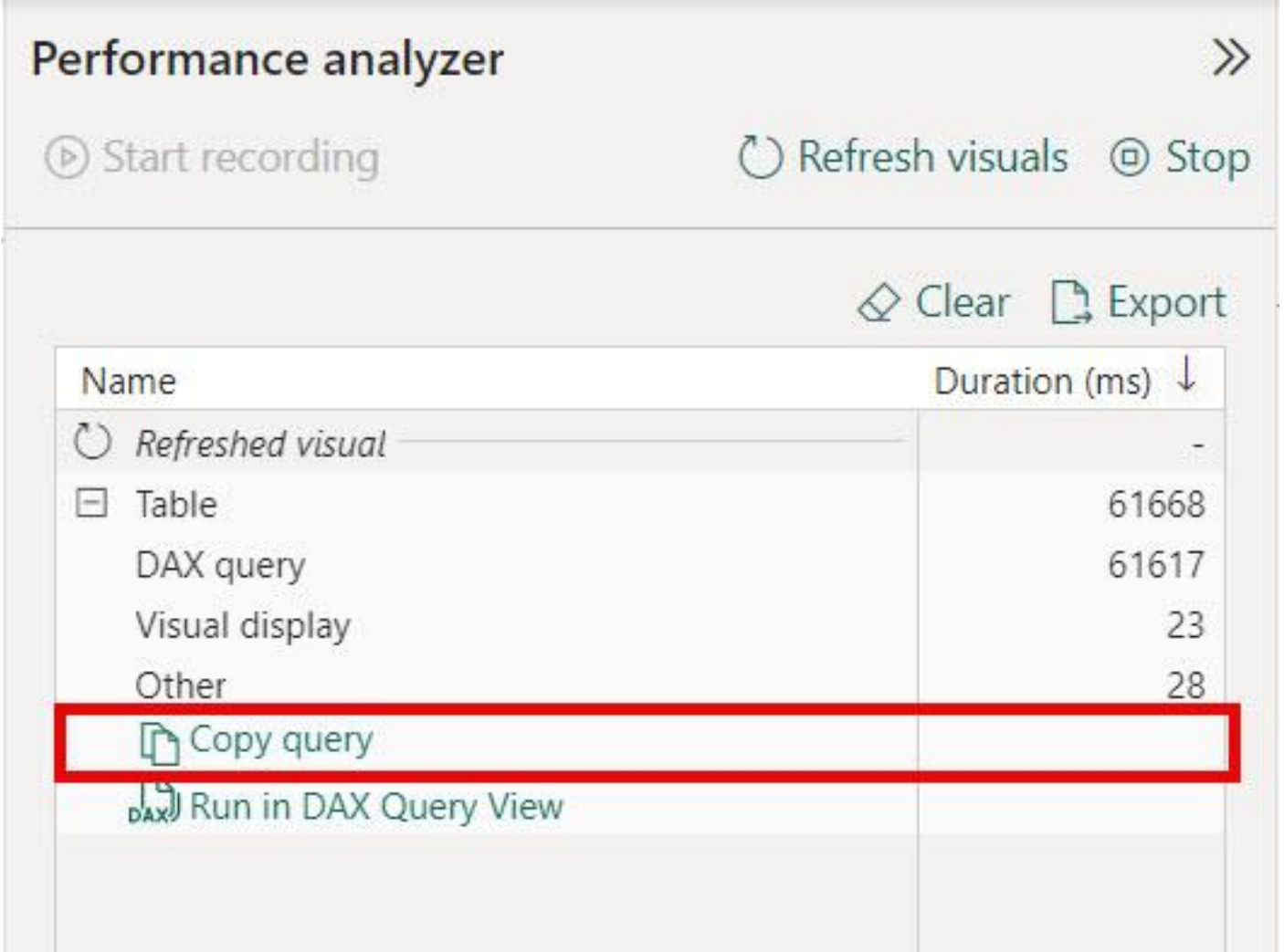
DAX Studio

- Author Measures and Queries
- Connect to Semantic Models
- Performance Tuning
  - Details on how the query is processed

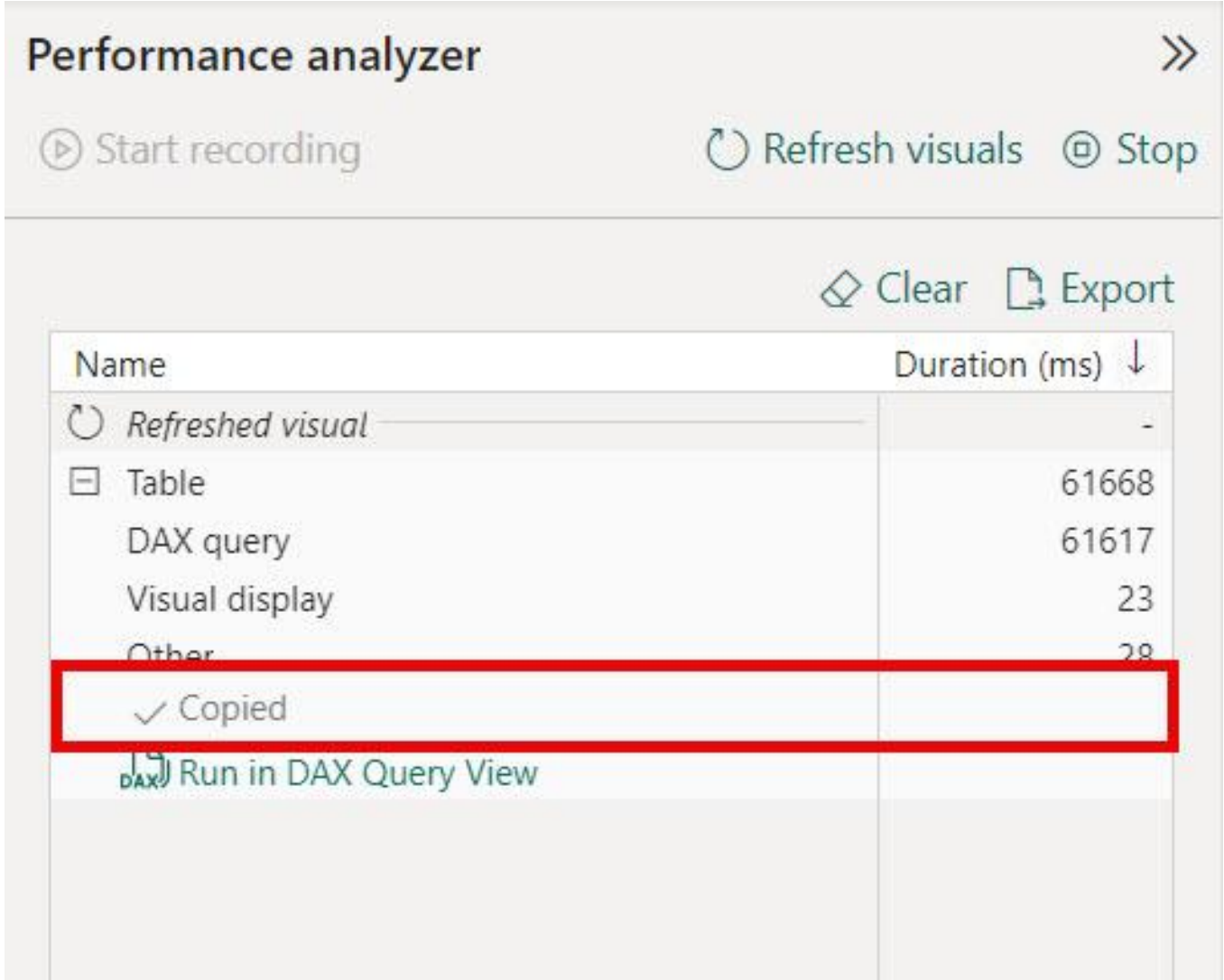
Slow DAX Query



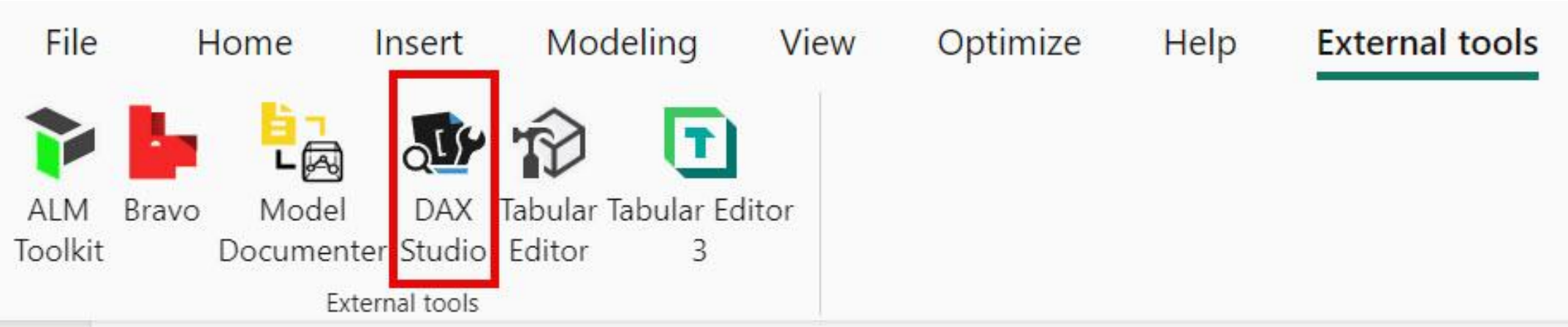
Copy query


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

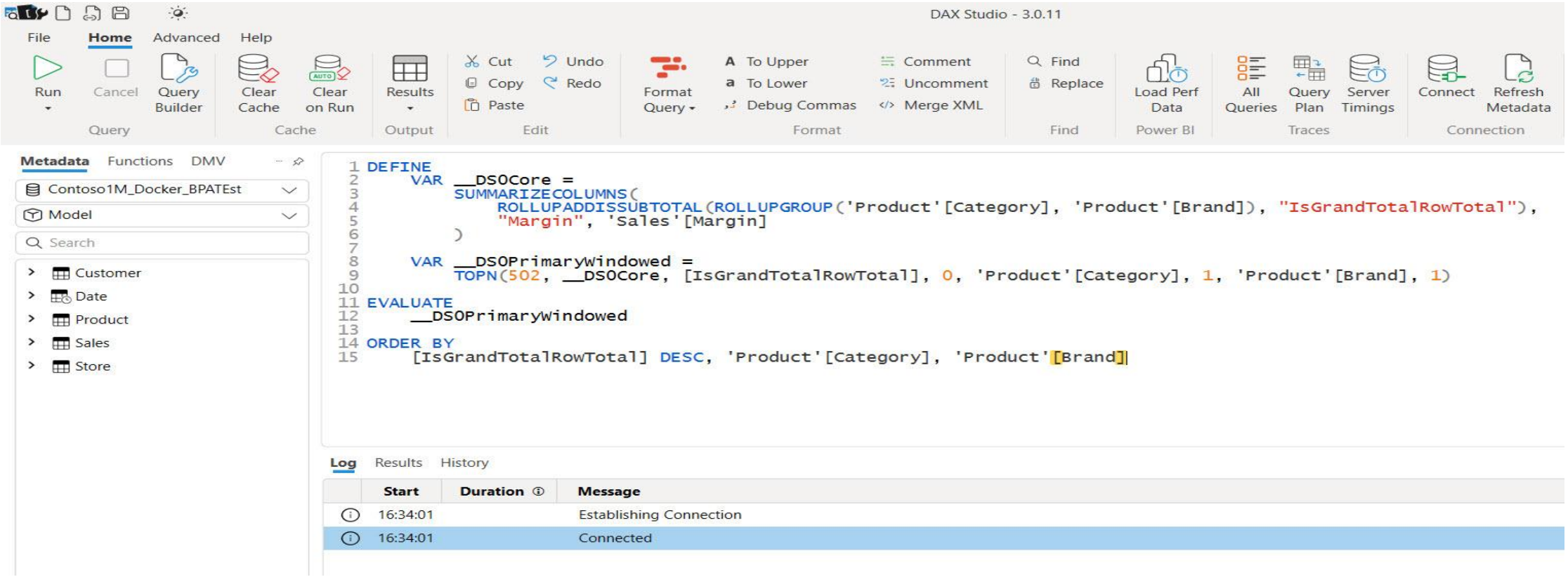
Copied


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

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

```
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]
```

125

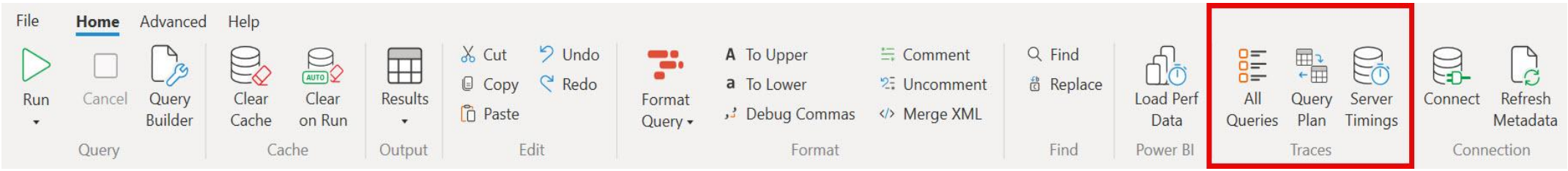
Log | Results | History | Server Timings

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

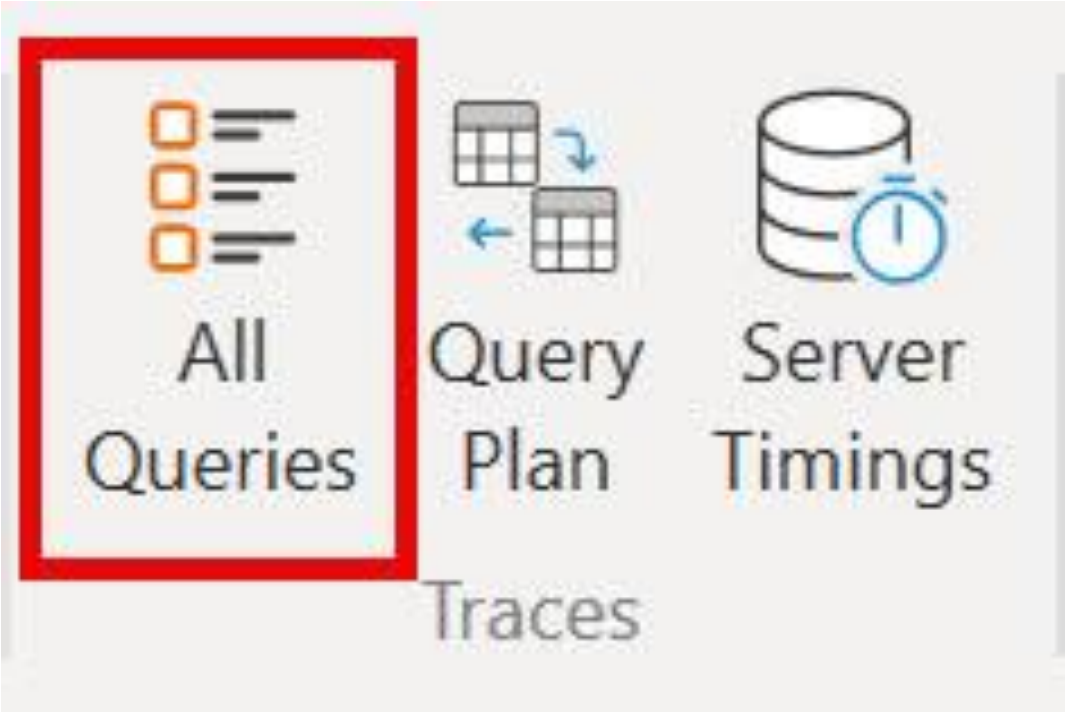
Query1.dax*

Ready | Ln 15, Col 50 | localhost:53193 | 16.0.126.27 | 102 | 30 rows | 00:00.0

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

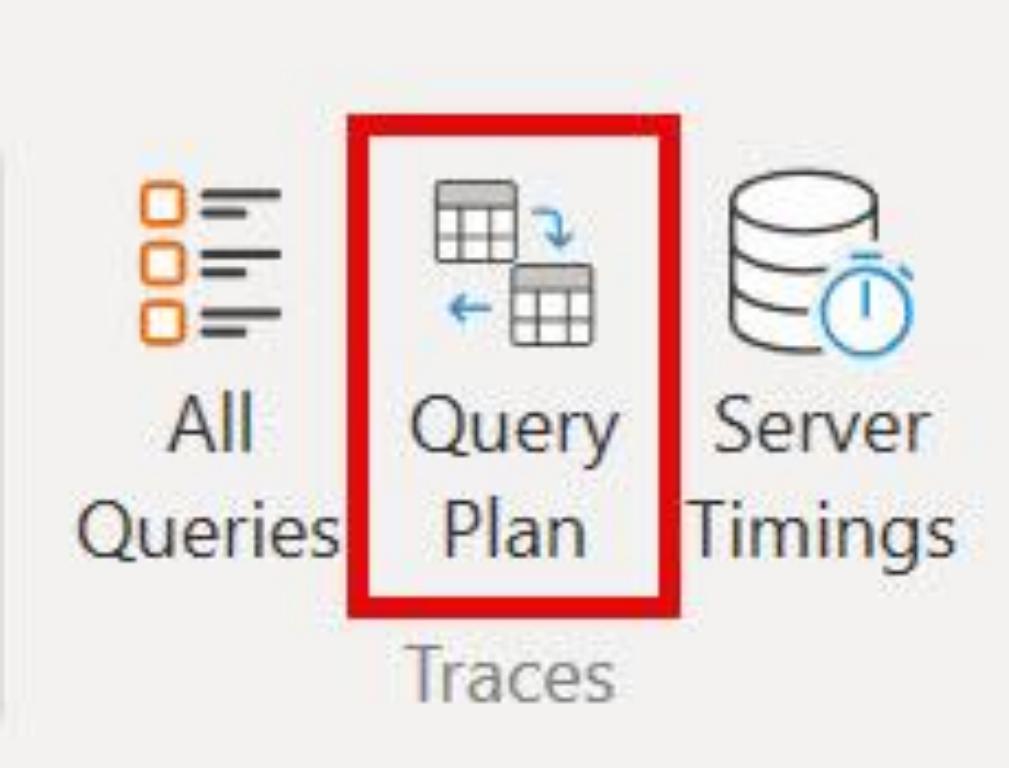
DAX Studio – All Queries

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...

Query Plan



Query Plans

Two Types

- Logical
- Physical

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

Query Plan – One Useful Thing

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 – Records

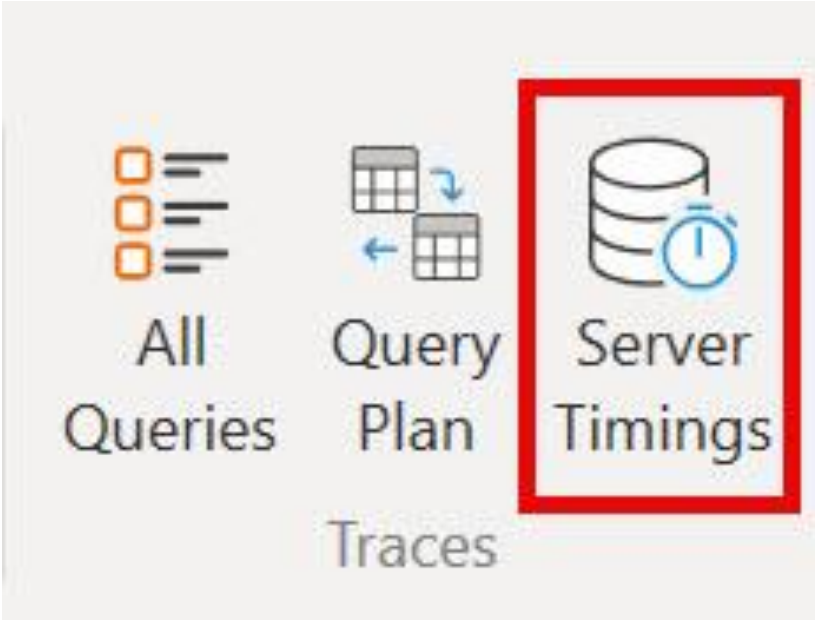
Log Results History Server Timings **Query Plan**

Record Pause Stop Clear Export Info

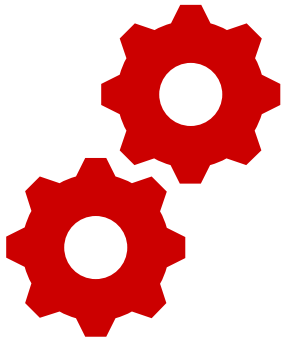
Line	Records	Physical Query Plan
1		PartitionIntoGroups: IterPhyOp LogOp=Order IterCols(0, 1, 2, 3)('Date'[Year Month], ''[IsGrandTotalRowTotal], ''[Total_BIG_Orders_Good], ''[]) #Groups=1 #Rows=120
2	1	AggregationSpool<Order>: SpoolPhyOp #Records=1
3		Proxy: IterPhyOp LogOp=TableVarProxy IterCols(0,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

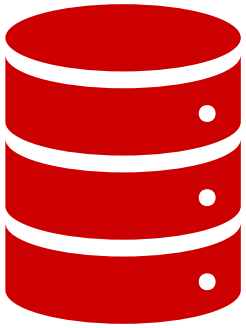
DAX Studio – Server Timings



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

Beyond the Scope

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

Vertipaq

- Column Based
- Encoded
- Compressed

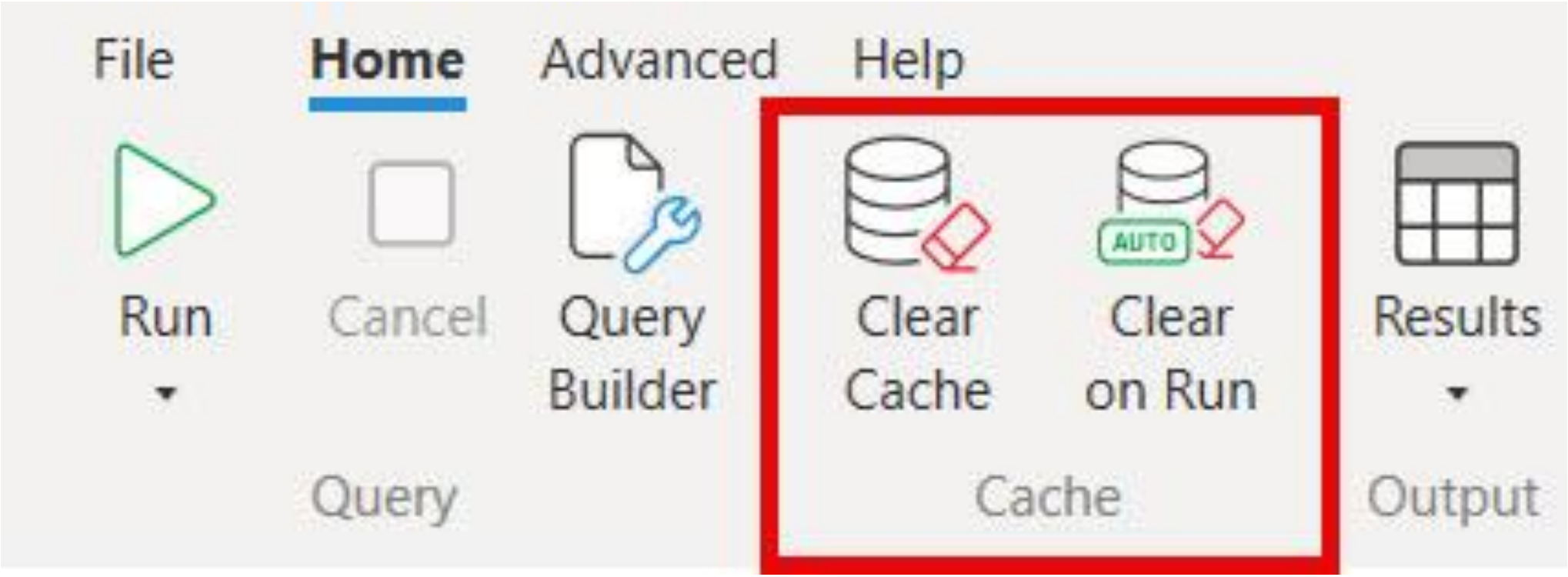
Vertipaq

- Restaurant Analogy
  - Local Storeroom that has been optimized for speed

Direct Query

- Restaurant Analogy
- Supplier brings what I need
- Able to combine more but at what cost

Cache


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

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 – Return Product Table

```
EVALUATE
'Product'
```

Storage Engine Query

```
1 EVALUATE
2 '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	**SE**
36 ms 97.3%	1 ms 2.7%
SE Queries	**SE Cache**
1	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

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(

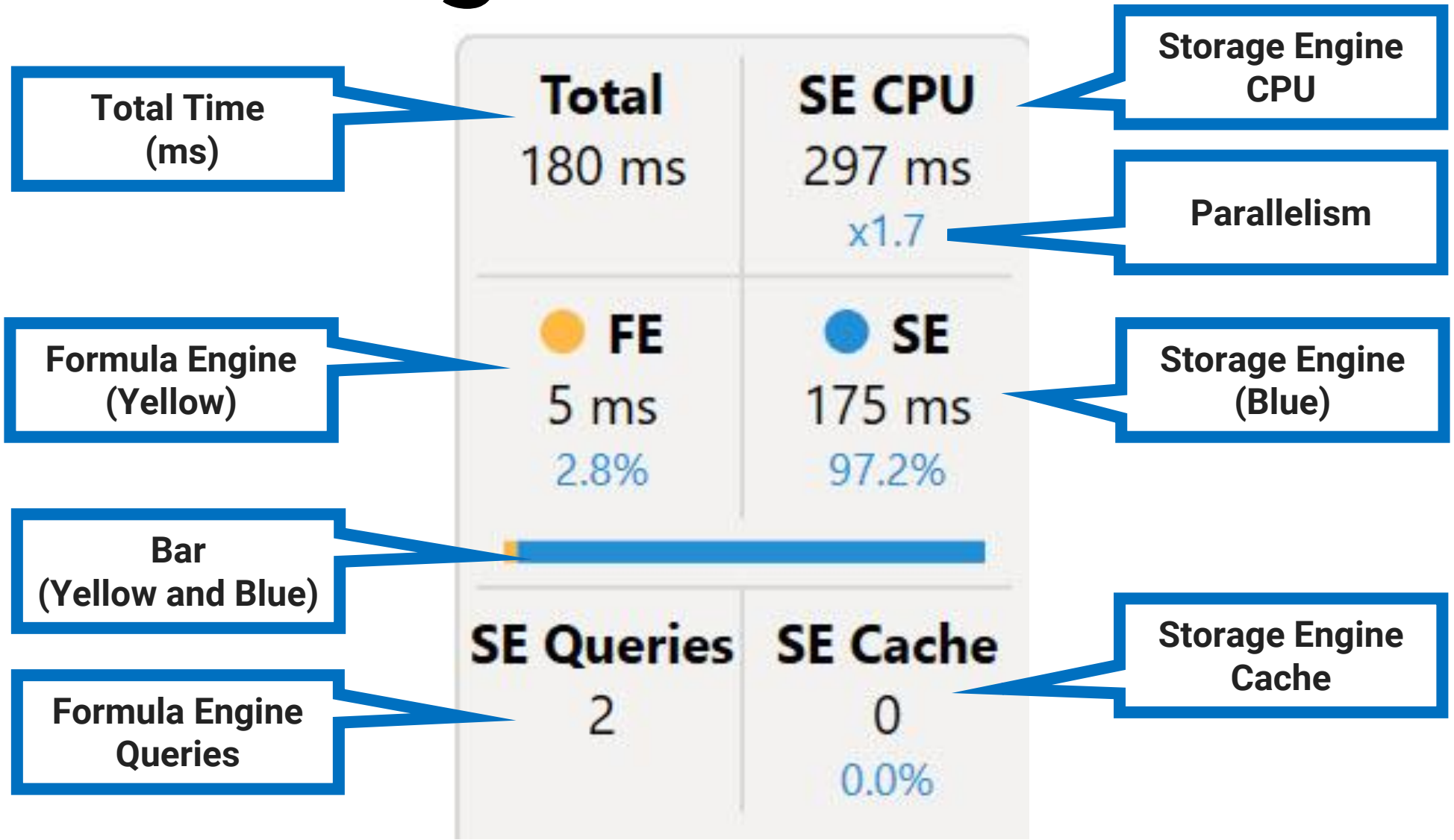
Storage Engine Query - Timeline

This one 33% in FE.
Blue shows where SE comes in the timeline.

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

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

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%

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 – Heavy 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 (Vertipaq)

- IF Statements
- CONCATENATE

CallbackDataID

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

The more you know…

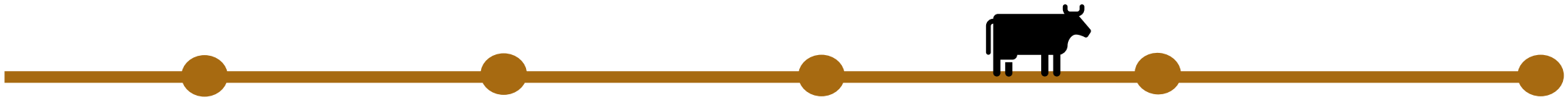
- If you can think like the engines
  - You can anticipate performance issues

Direct Query Model

- Direct Query
  - How many queries are being sent to Source
  - What about the Date table in Import and other tables Direct Query

Power BI Demo
DAX Studio

Our Journey



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

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

Tabular Editor 3.x

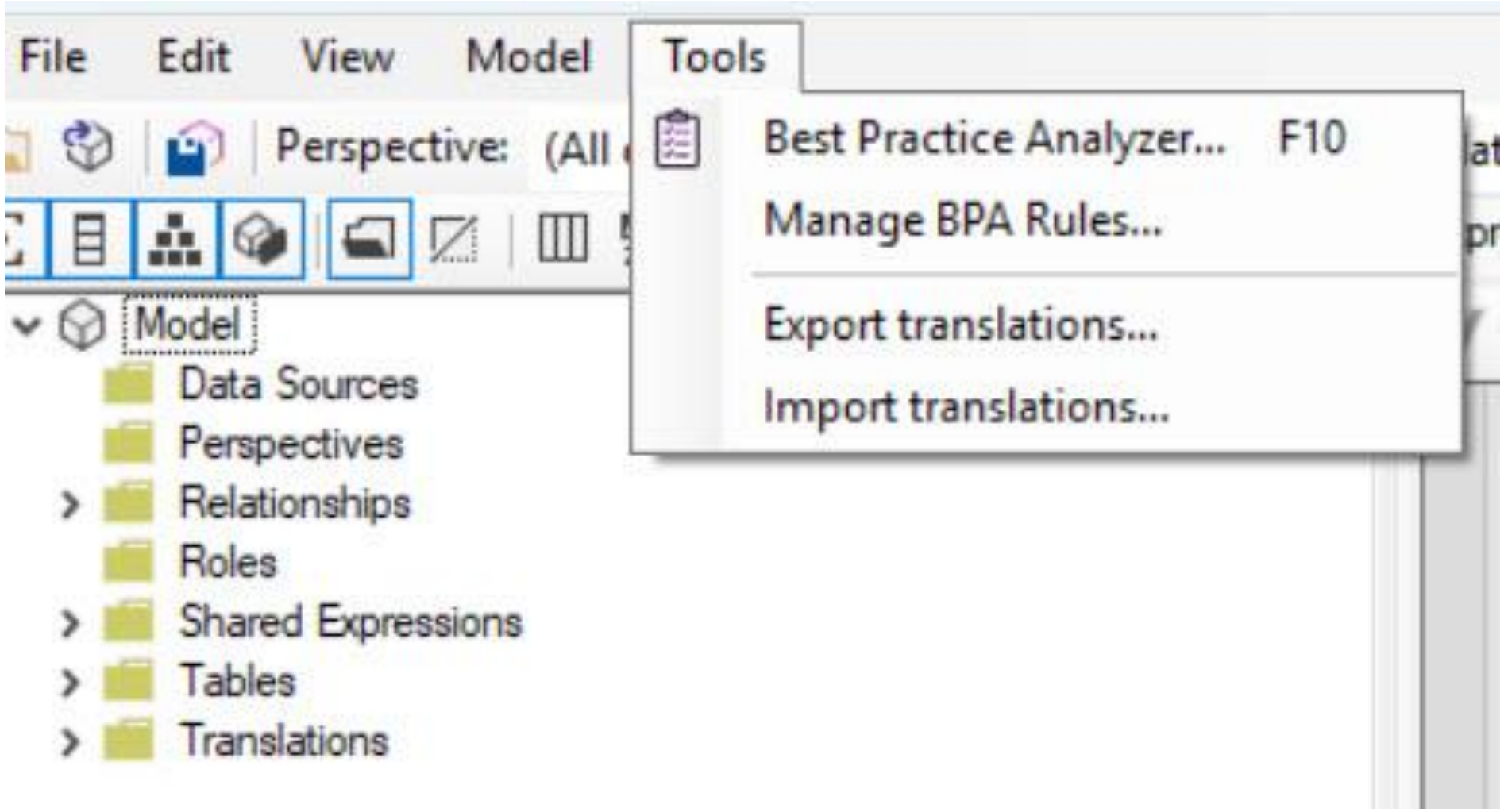
- Paid Version
- Extra Features
  - DAX Debugger
  - Script DAX

Tabular Editor 2

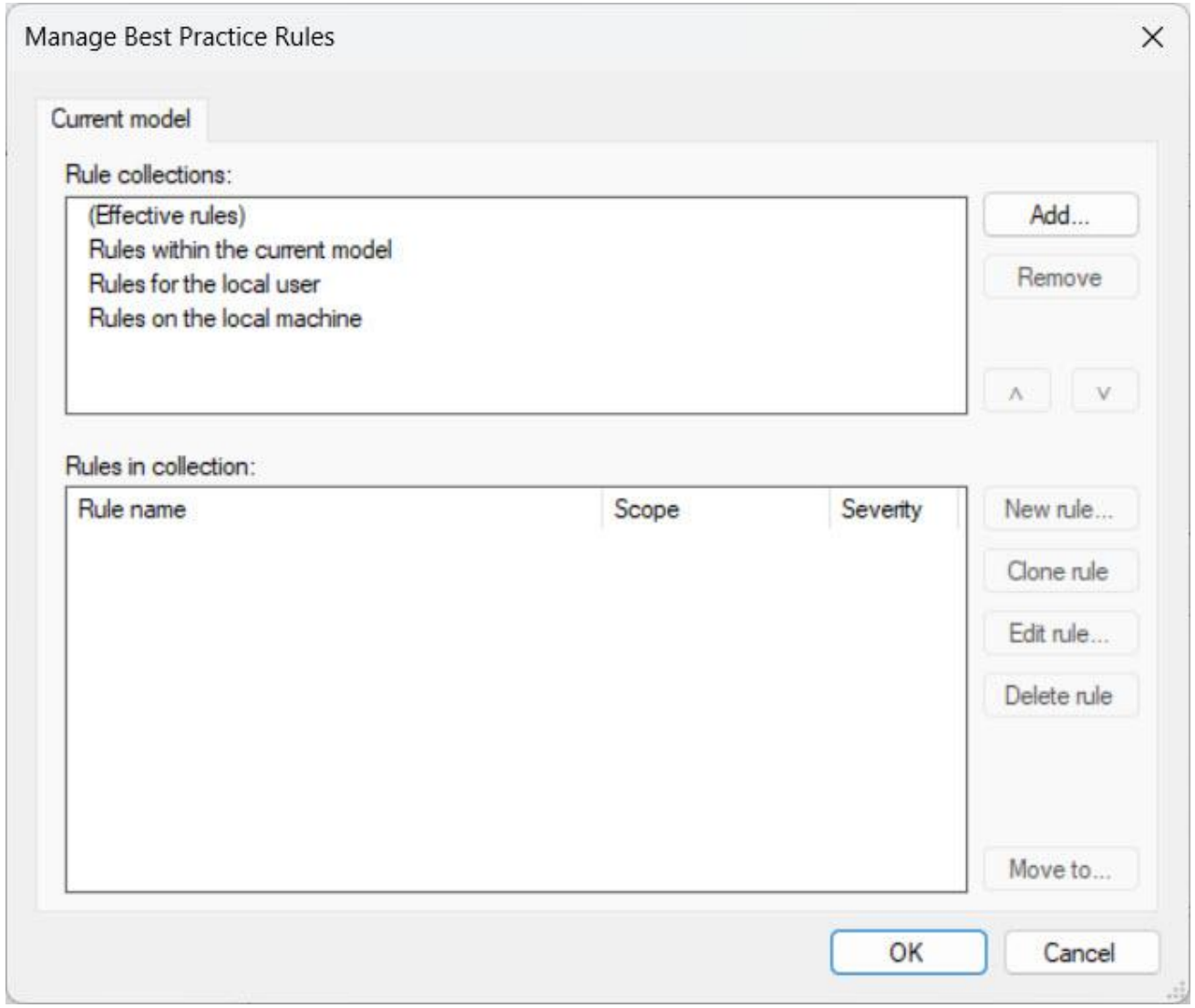
Optimize a semantic model by using Tabular Editor 2

Best Practice Analyzer

Best Practice Analyzer Location



Add and Manage Rules



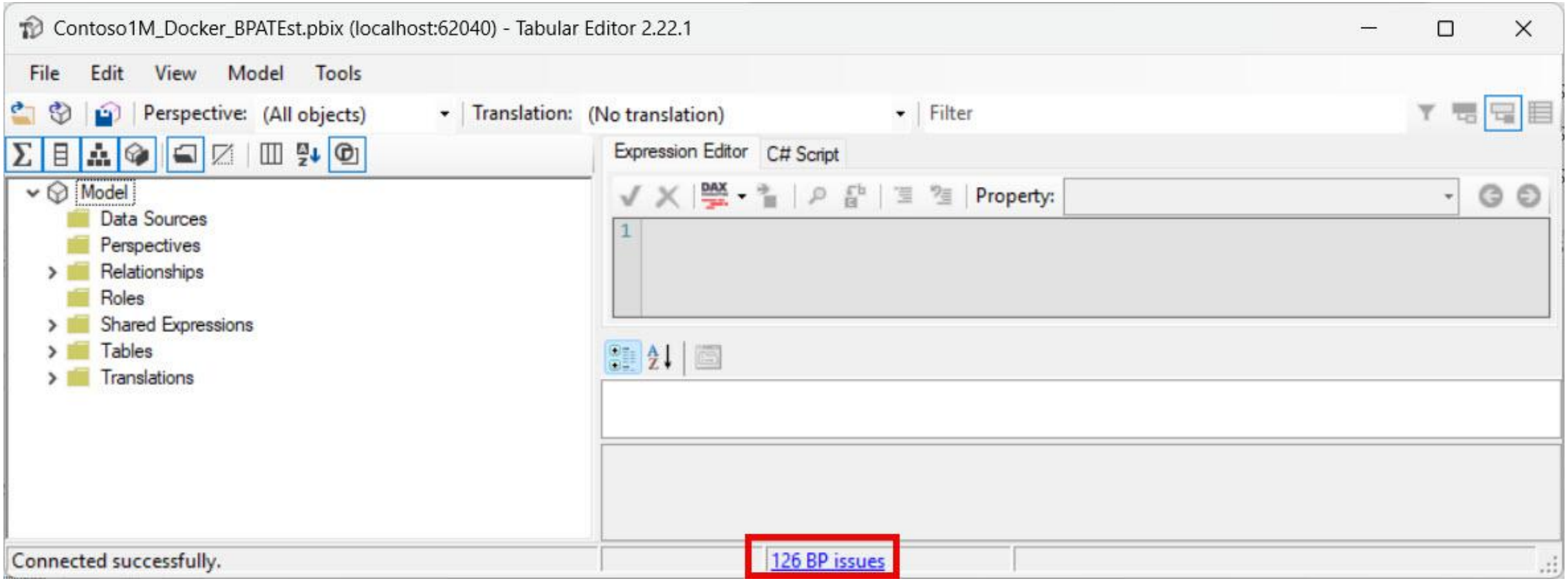
Best Practice Analyzer – C# Script Download

```
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";
*/
```

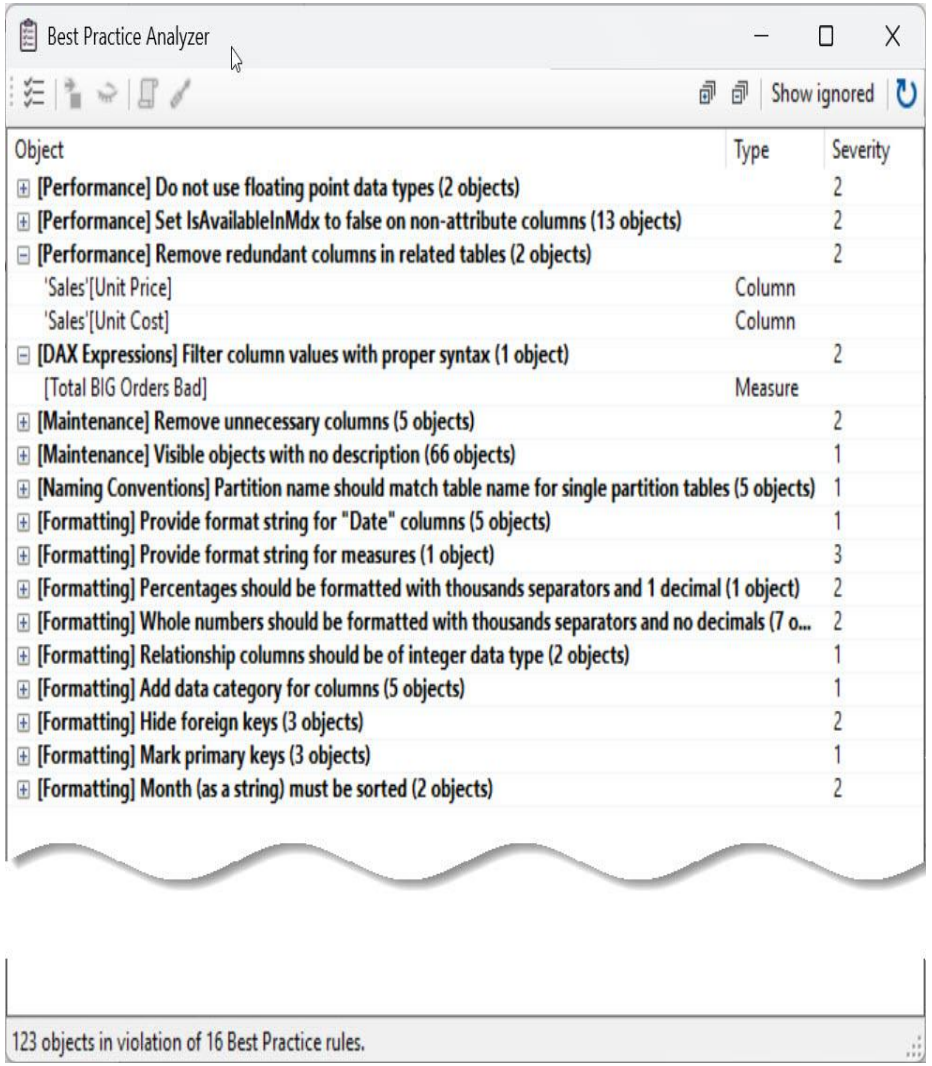
Nope – Not Running a Script

- 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



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

What about that slow pattern?

Filter by the Whole Table

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

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','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"))

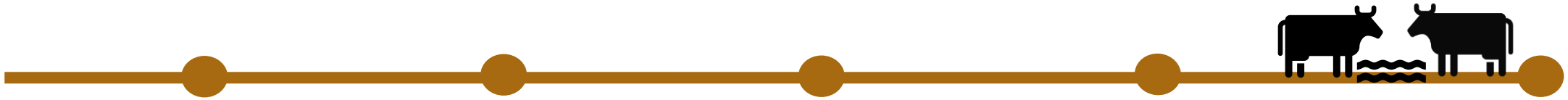
Top 10 Power BI mistakes and their best practice solutions (https://www.elegantbi.com/post/top10bestpractices)

Change Description

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

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 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]

DON’T FORGET TO SUPPORT NTSSUG

Follow us on LinkedIn

Join us monthly for more learning.

Third Thursday of every Month right here at this same location.

Meetup

Google

Thank you

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