Optimising Power BI with Azure Synapse Analytics Serverless SQL Pools

Power BI Fest
Saturday 20th November 2021

Microsoft Partner
Silver Data Analytics
Silver Data Platform

MVP Microsoft® Most Valuable Professional

Andy Cutler

Independent BI/DW Consultant

Azure Data Platform & Power BI

datahai.co.uk/blog

serverlesssql.com

twitter.com/MrAndyCutler

linkedin.com/in/andycutler/

Session Overview

Synapse Analytics Setup

Connecting Power BI

DirectQuery or Import

Filtering & Data Processed

Aggregates & Incremental

Synapse Analytics

Serverless SQL Pools

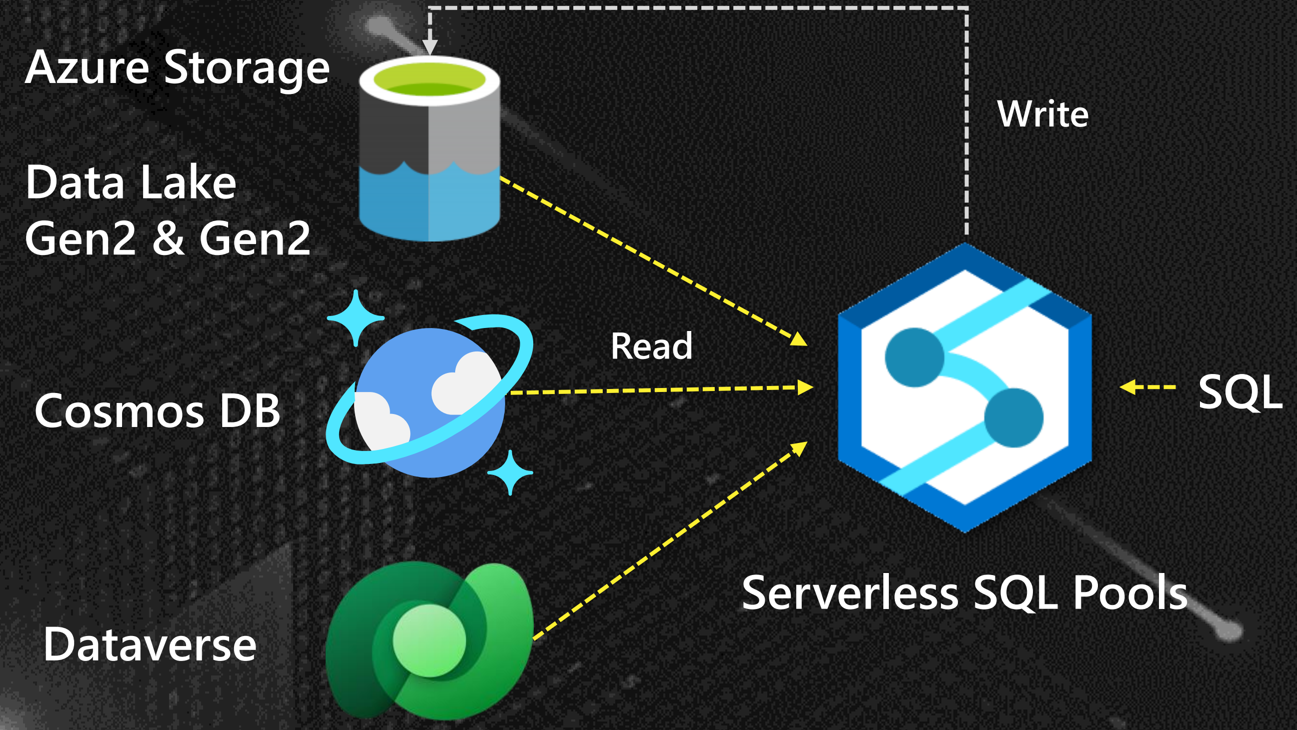
Serverless SQL Pools

Query external data from Azure Storage, Cosmos DB and Dataverse

Familiar SQL objects

- Databases
- Stored Procedures
- DMVs
- Views
- External Tables

What is Serverless?



Serverless SQL Pools cost is based on the amount of data processed and not compute/time to execute

~$5 per 1TB Data Processed (Write/Read)

No data is stored within Serverless SQL Pools

Serverless SQL Scenarios

Microsoft state 3 scenarios that Serverless SQL Pools can be useful for

Scenarios

Data Exploration

Analyse CSV, Parquet & JSON data stored in Azure Storage using common T-SQL commands. Query Cosmos DB in real-time.

Logical Data Warehouse

Create a relational structure over raw data stored in Azure Storage and Cosmos DB without transforming and moving data.

Data Transformation

Data stored in Azure Storage can be transformed using T-SQL and datasets returned to BI tools such as Power BI

Lightweight SQL Engine

Think of Serverless SQL Pools as a lightweight SQL engine

Lightweight

We can create a Synapse Analytics workspace and only ever use the Serverless SQL Pools service for data processing

We can create Views and External Tables over disparate Data Lake data to bring this data together

Use Serverless SQL Pools to do the “heavy lifting” in terms of data processing when data is stored in a Data Lake

Creating a Synapse Analytics Workspace

A Synapse Analytics Workspace can be provisioned using:

- Azure Portal
- PowerShell
- CLI
- ARM
- Bicep

Creation

We can create a Synapse Analytics workspace in just a few steps:

Specify the Azure Subscription

Select or Create a Resource Group

Enter a Workspace name

Select or Create a Storage Account (Data Lake Gen2)

Enter a file system name

Specify SQL admin credentials

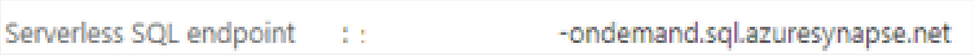
Specify if workspace is created in a Managed Virtual Network

Power BI

Connecting to Serverless SQL Pools

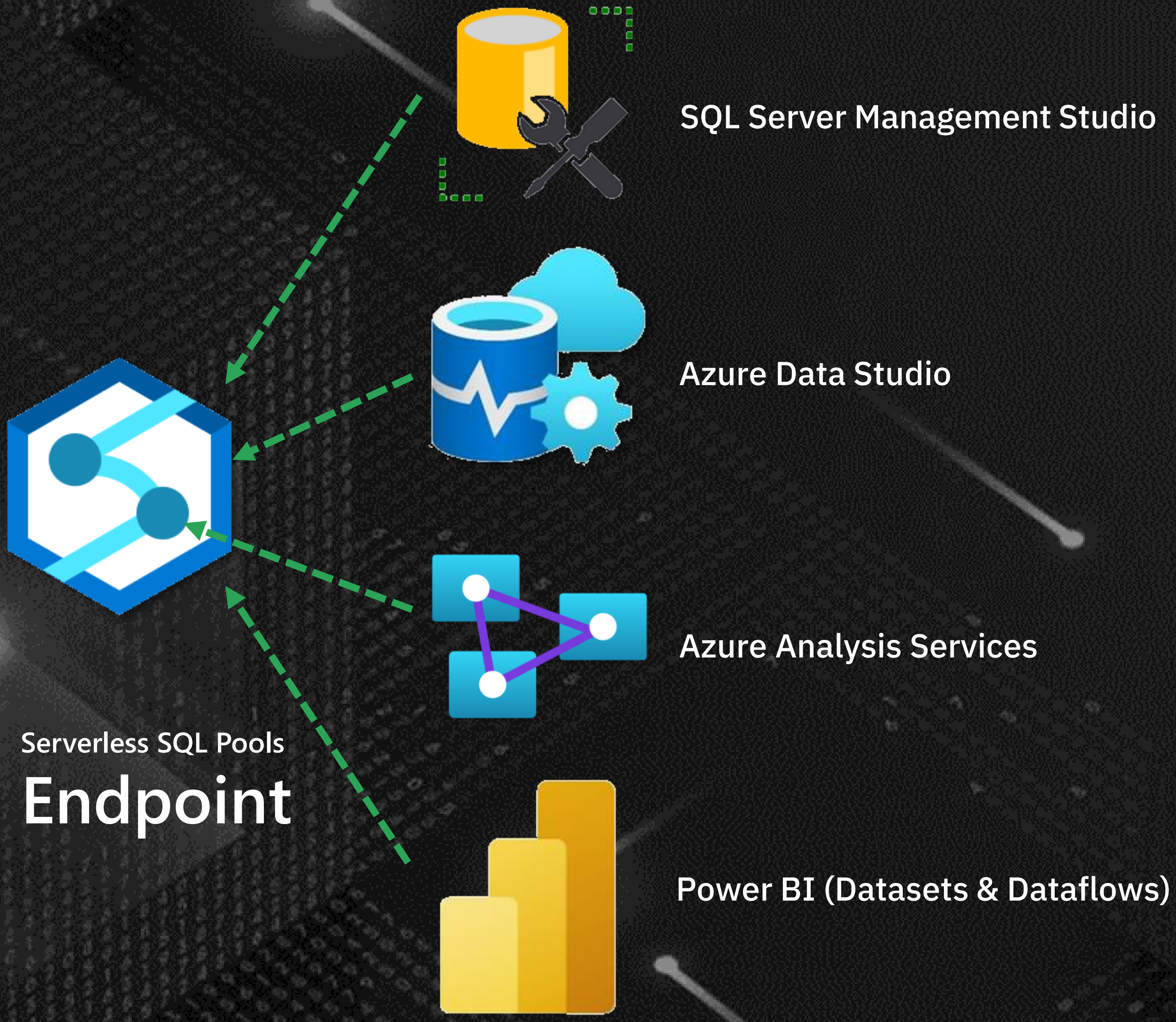
Connection

Serverless SQL Pools has a separate endpoint which other data services can connect to and issue SQL statements



Serverless SQL endpoint : : -ondemand.sql.azuresynapse.net

Endpoint



SQL Server Management Studio

Azure Data Studio

Azure Analysis Services

Power BI (Datasets & Dataflows)

Serverless SQL Pools

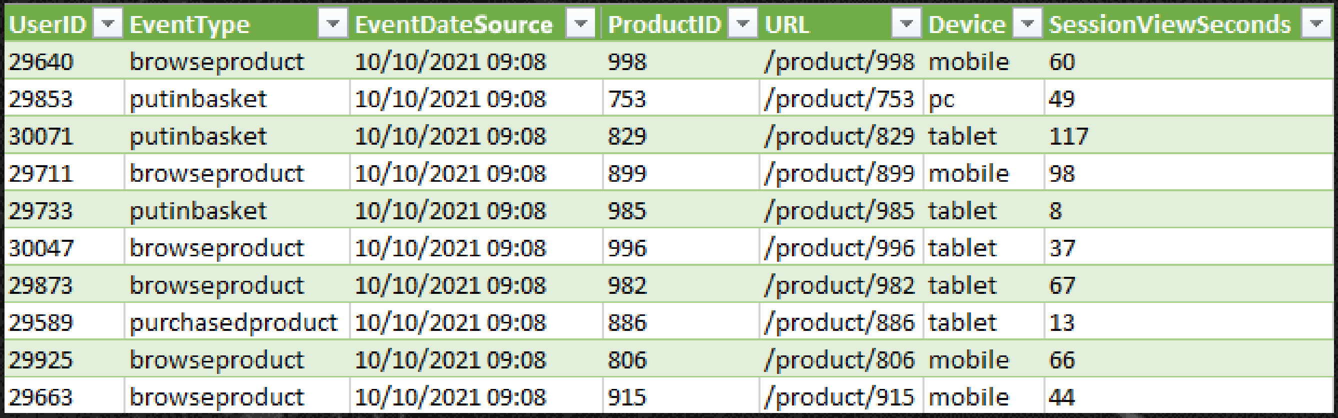
Endpoint

Data Example

We have Web Telemetry data being streamed into Azure Data Lake Gen2 into a folder structure

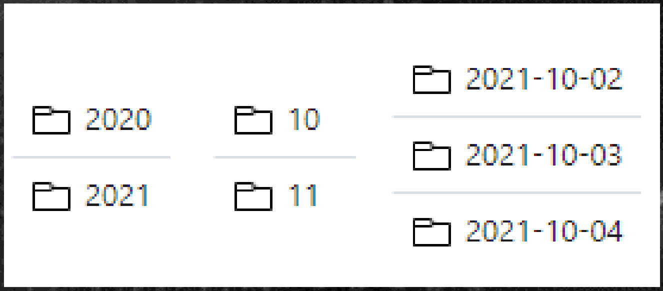
Source Data

In the Web Telemetry data we have 7 columns



UserID	EventType	EventDateSource	ProductID	URL	Device	SessionViewSeconds
29640	browseproduct	10/10/2021 09:08	998	/product/998	mobile	60
29853	putinbasket	10/10/2021 09:08	753	/product/753	pc	49
30071	putinbasket	10/10/2021 09:08	829	/product/829	tablet	117
29711	browseproduct	10/10/2021 09:08	899	/product/899	mobile	98
29733	putinbasket	10/10/2021 09:08	985	/product/985	tablet	8
30047	browseproduct	10/10/2021 09:08	996	/product/996	tablet	37
29873	browseproduct	10/10/2021 09:08	982	/product/982	tablet	67
29589	purchasedproduct	10/10/2021 09:08	886	/product/886	tablet	13
29925	browseproduct	10/10/2021 09:08	806	/product/806	mobile	66
29663	browseproduct	10/10/2021 09:08	915	/product/915	mobile	44

The file format is Parquet



There is a 3 level folder structure with the Parquet data being stored in the YYYY-MM-DD folder

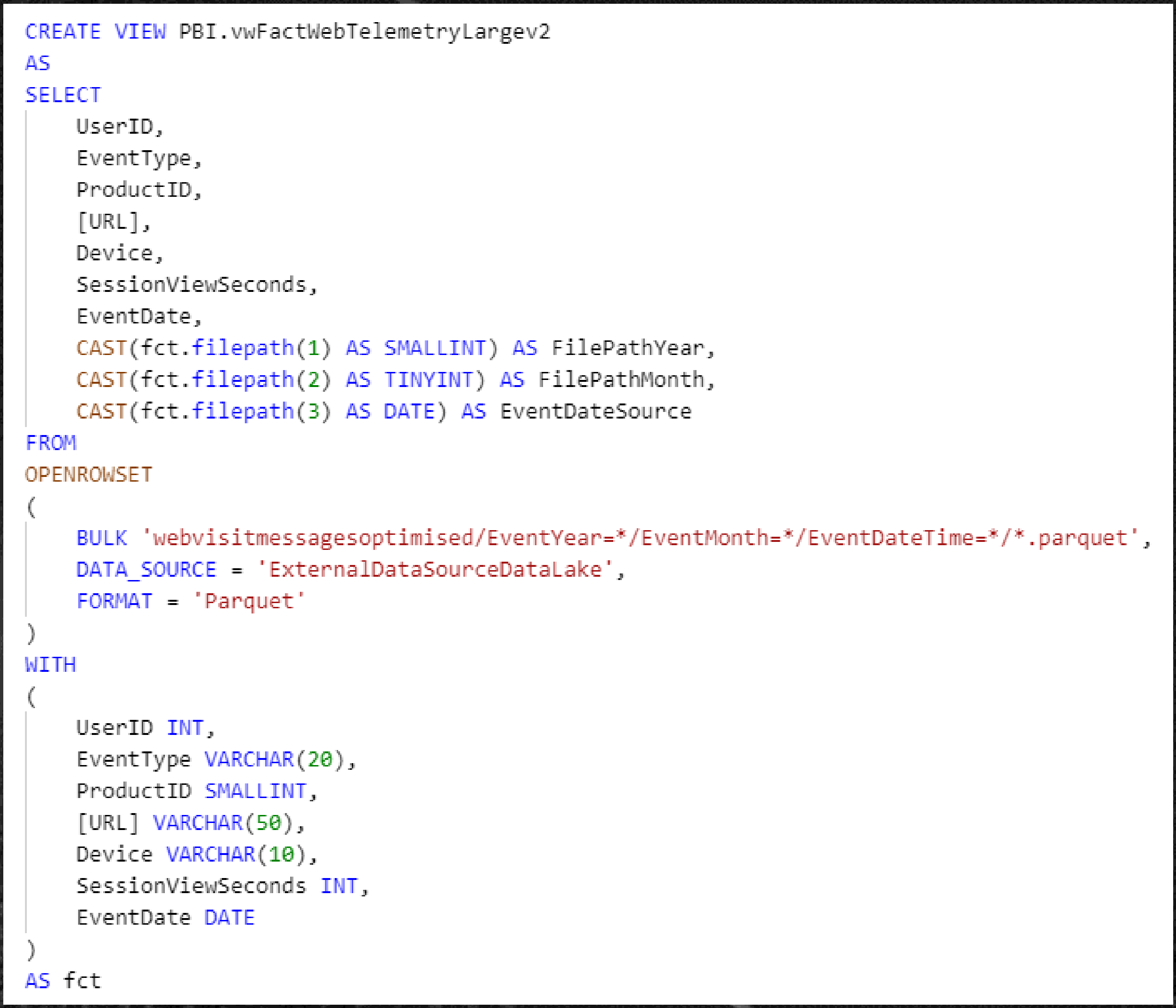
The Date column is surfaced in a View in Serverless as a Date column

Creating SQL View

We can create a SQL View in Serverless SQL Pools to cast structure over this data stored in the Data Lake

SQL

Create View syntax for the Web Telemetry data



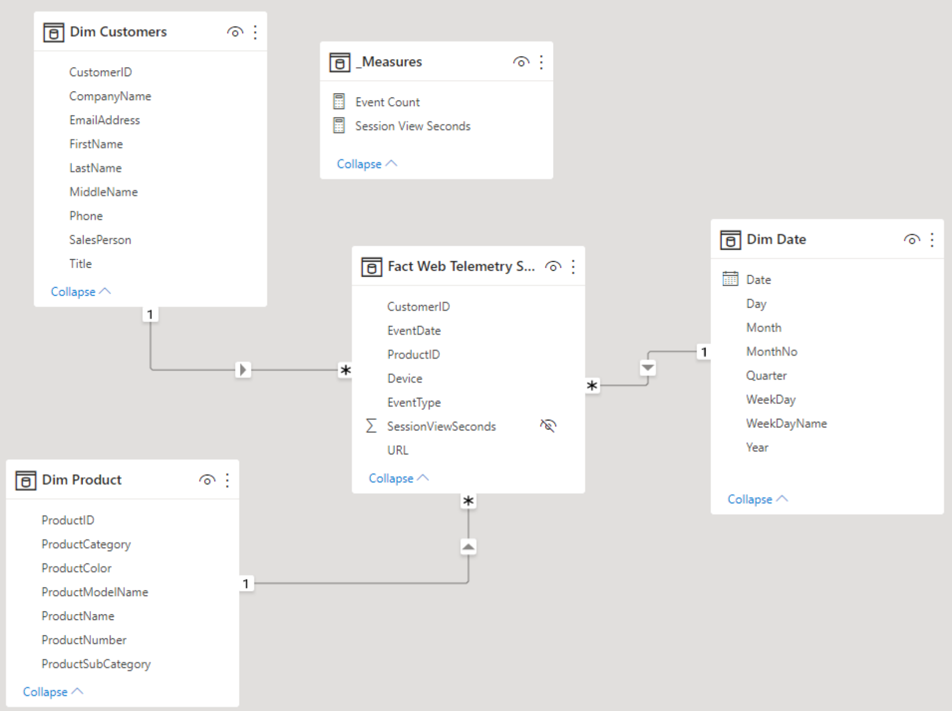
```
CREATE VIEW PBI.vwFactWebTelemetryLargev2
AS
SELECT
    UserID,
    EventType,
    ProductID,
    [URL],
    Device,
    SessionViewSeconds,
    EventDate,
    CAST(fct.filepath(1) AS SMALLINT) AS FilePathYear,
    CAST(fct.filepath(2) AS TINYINT) AS FilePathMonth,
    CAST(fct.filepath(3) AS DATE) AS EventDateSource
FROM
OPENROWSET
(
    BULK 'webvisitmessagesoptimised/EventYear=*/EventMonth=*/EventDateTime=*/*.parquet',
    DATA_SOURCE = 'ExternalDataSourceDataLake',
    FORMAT = 'Parquet'
)
WITH
(
    UserID INT,
    EventType VARCHAR(20),
    ProductID SMALLINT,
    [URL] VARCHAR(50),
    Device VARCHAR(10),
    SessionViewSeconds INT,
    EventDate DATE
)
AS fct
```

Import

We can load data into Power BI from Serverless SQL Pools

We can keep the granularity the same as the source

Loading Data


Dim Customers

CustomerID
CompanyName
EmailAddress
FirstName
LastName
MiddleName
Phone
SalesPerson
Title

Collapse

_Measures

Event Count
Session View Seconds

Collapse

Fact Web Telemetry S...

CustomerID
EventDate
ProductID
Device
EventType
SessionViewSeconds
URL

Collapse

Dim Date

Date
Day
Month
MonthNo
Quarter
WeekDay
WeekDayName
Year

Collapse

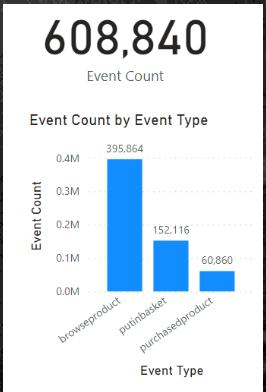
Dim Product

ProductID
ProductCategory
ProductColor
ProductModelName
ProductName
ProductNumber
ProductSubCategory

Collapse

In this example we're importing 600K rows into a Power BI data model

We are performing the same data modelling operations as with any imported data source



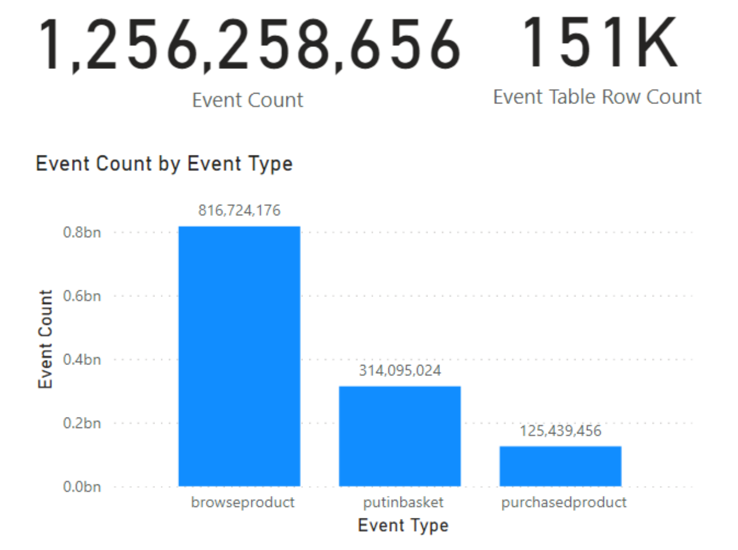
Bear in mind the volume of source data as if data is being loaded to a Data Lake, the volume could grow very quickly

Import with Grouping

Larger datasets may require aggregating

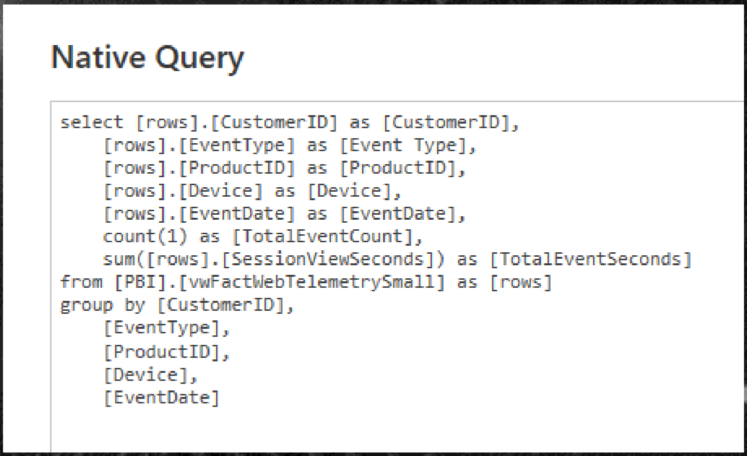
We must ensure as much processing is pushed to Serverless SQL Pools (Query Folding)

Aggregate Data



In this example we're aggregating over 1.2B rows into 150K rows using Power Query Grouping

We are performing the same data modelling operations as with any imported data source



Native Query

```
select [rows].[CustomerID] as [CustomerID],
    [rows].[EventType] as [Event Type],
    [rows].[ProductID] as [ProductID],
    [rows].[Device] as [Device],
    [rows].[EventDate] as [EventDate],
    count(1) as [TotalEventCount],
    sum([rows].[SessionViewSeconds]) as [TotalEventSeconds]
from [PBI].[vwFactWebTelemetrySmall] as [rows]
group by [CustomerID],
    [EventType],
    [ProductID],
    [Device],
    [EventDate]
```

Serverless SQL Pools is running the aggregate query due to Query Folding

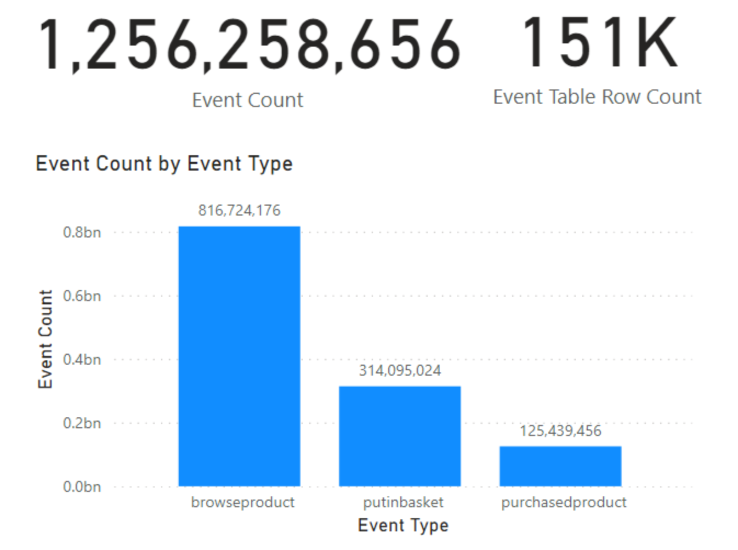
We have lost the granularity of the source data

Import with Grouping

Larger datasets may require aggregating

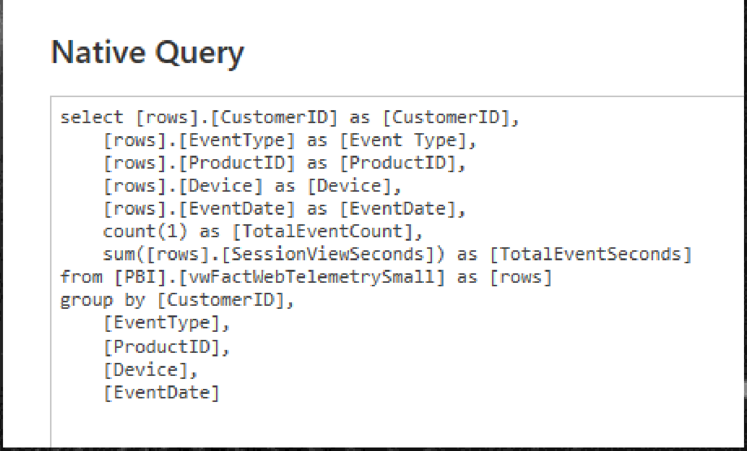
We must ensure as much processing is pushed to Serverless SQL Pools (Query Folding)

Aggregate Data



In this example we're aggregating over 1.2B rows into 150K rows using Power Query Grouping

We are performing the same data modelling operations as with any imported data source



Native Query

```
select [rows].[CustomerID] as [CustomerID],
    [rows].[EventType] as [Event Type],
    [rows].[ProductID] as [ProductID],
    [rows].[Device] as [Device],
    [rows].[EventDate] as [EventDate],
    count(1) as [TotalEventCount],
    sum([rows].[SessionViewSeconds]) as [TotalEventSeconds]
from [PBI].[vwFactWebTelemetrySmall] as [rows]
group by [CustomerID],
    [EventType],
    [ProductID],
    [Device],
    [EventDate]
```

Serverless SQL Pools is running the aggregate query due to Query Folding

We have lost the granularity of the source data

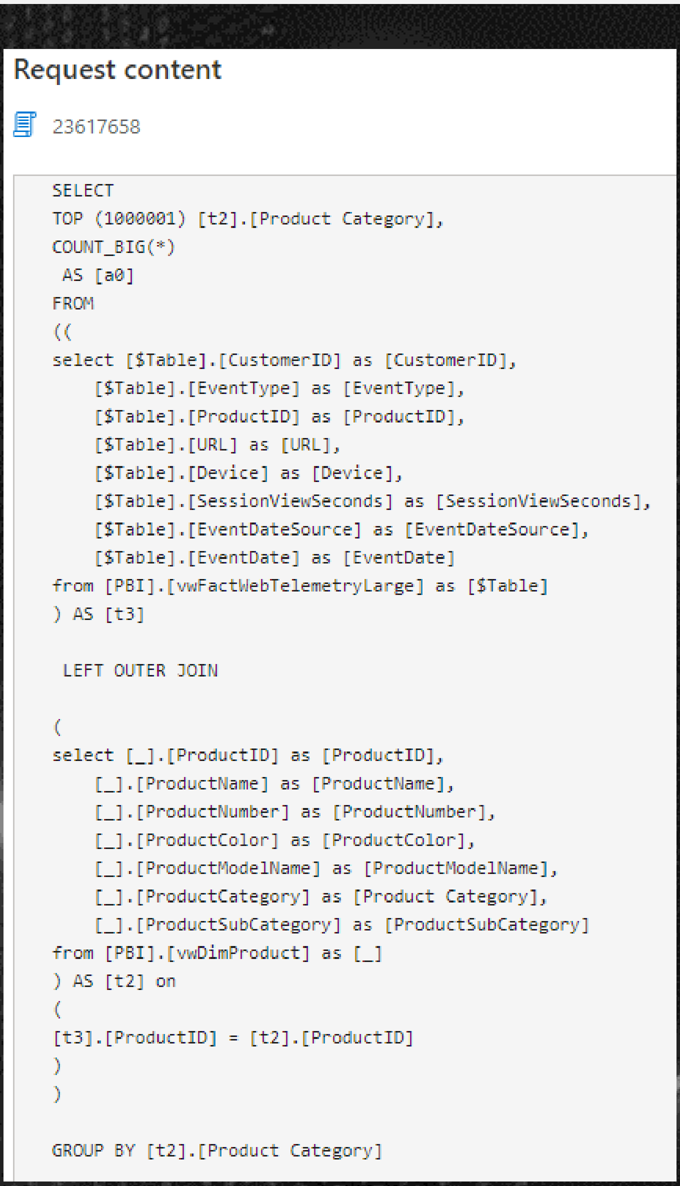
DirectQuery

We can connect without needing to import data

We have access to the same granularity as the source

Data is accessible as soon as received in the source

Connecting live



Request content

23617658

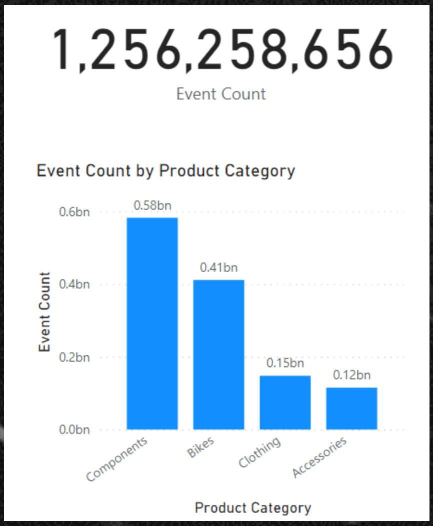
```
SELECT
TOP (1000001) [t2].[Product Category],
COUNT_BIG(*)
 AS [a0]
FROM
((
select [$Table].[CustomerID] as [CustomerID],
    [$Table].[EventType] as [EventType],
    [$Table].[ProductID] as [ProductID],
    [$Table].[URL] as [URL],
    [$Table].[Device] as [Device],
    [$Table].[SessionViewSeconds] as [SessionViewSeconds],
    [$Table].[EventDateSource] as [EventDateSource],
    [$Table].[EventDate] as [EventDate]
from [PBI].[vwFactWebTelemetryLarge] as [$Table]
) AS [t3]

 LEFT OUTER JOIN

(
select [_].[ProductID] as [ProductID],
    [_].[ProductName] as [ProductName],
    [_].[ProductNumber] as [ProductNumber],
    [_].[ProductColor] as [ProductColor],
    [_].[ProductModelName] as [ProductModelName],
    [_].[ProductCategory] as [Product Category],
    [_].[ProductSubCategory] as [ProductSubCategory]
from [PBI].[vwDimProduct] as [_]
) AS [t2] on
(
[t3].[ProductID] = [t2].[ProductID]
)
)

GROUP BY [t2].[Product Category]
```

Keep accessing source rows with no loss of granularity

No need to import as we're connecting live

Queries are run by Serverless SQL Pools

Performance will not be as fast as import



Filtering in DirectQuery

We can use the filepath() columns to filter and partition prune to reduce the data processed

Partition Pruning

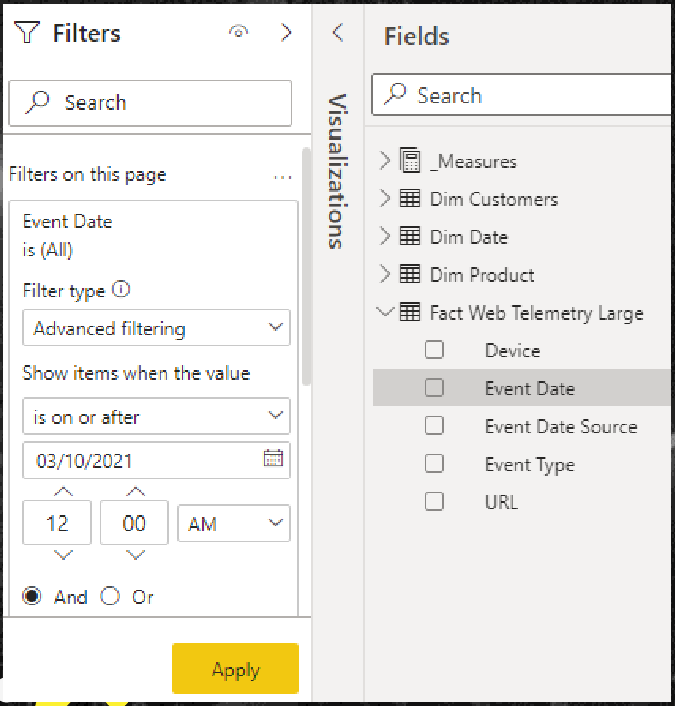
We have 2 Date columns in the Fact View: EventDateSource & EventDate

EventDateSource: Original Event date which is stored in the Parquet data

Serverless SQL Pools needs to scan all folders and files

EventDate: Result of the filepath() function to return the folder name

No support to join to another table and have that table filter, E.G Date dimension



Event Date
is (All)

Filter type: Advanced filtering

Show items when the value

is on or after

03/10/2021

12 : 00 AM

● And ○ Or

- _Measures
- Dim Customers
- Dim Date
- Dim Product
- Fact Web Telemetry Large
  - □ Device
  - □ Event Date
  - □ Event Date Source
  - □ Event Type
  - □ URL

We can use the Date dimension as context rather than filtering

We can use the option to add a single apply button for filters and slicers

Engine

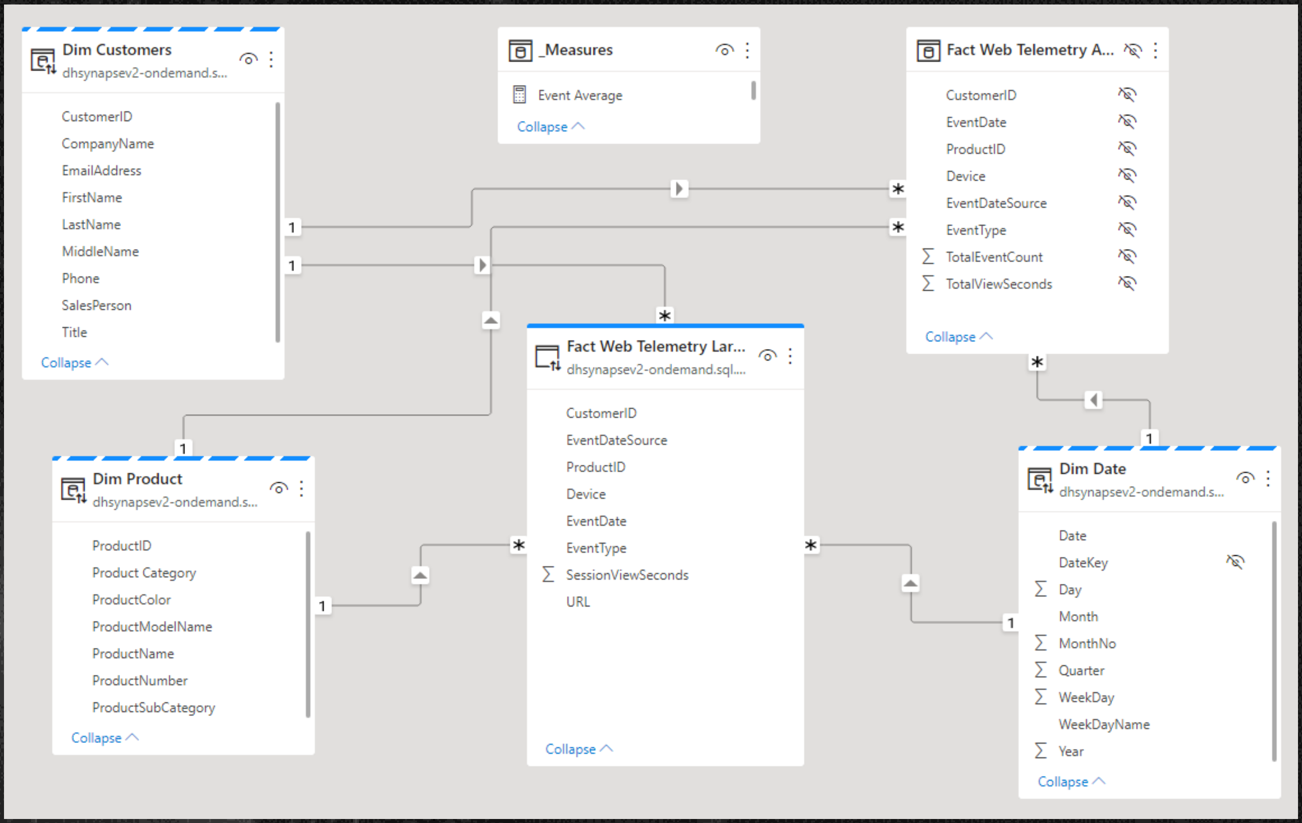
Pushing Processing to Serverless SQL Pools

Aggregates

We can import an aggregate table into Power BI and keep the source granularity accessible using DirectQuery

Improve Speed

We can use Aggregations to reduce the time to answer specific aggregate queries



We must be mindful of query coverage to ensure aggregations are being hit as much as possible

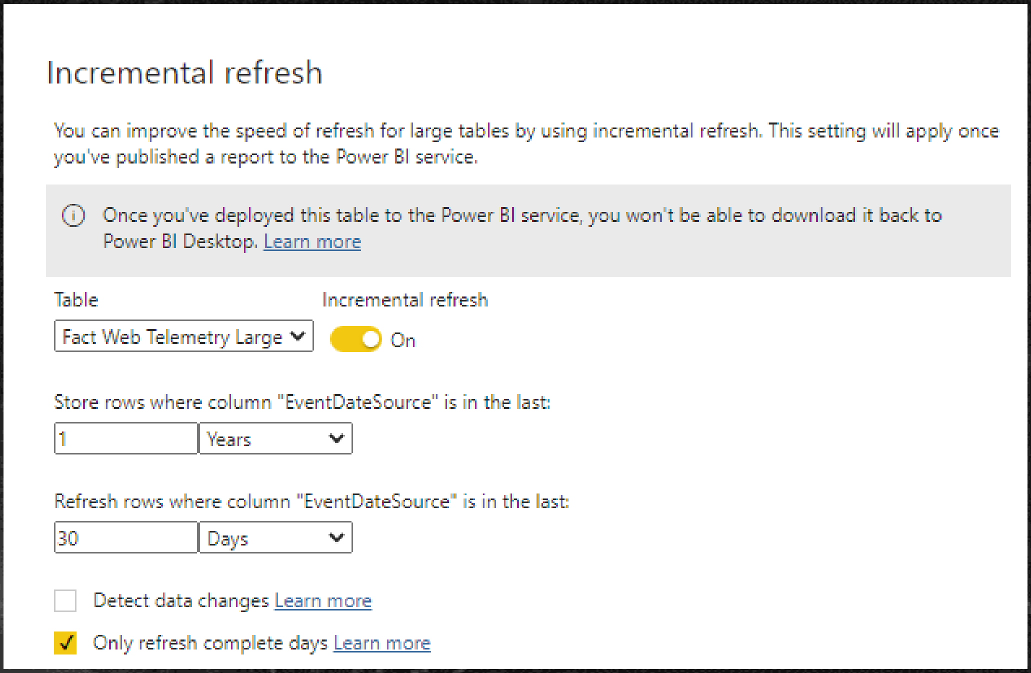
Incremental Loading

We can setup incremental loading if there is a date/time column

Use the result of a filepath function to return a date/time value from the source folder to enable partition pruning

Efficiency

If we are able to import data (either row by row, or by aggregating/grouping) then we can take advantage of incremental refresh in Power BI and partition pruning in Serverless SQL Pools.



Incremental refresh

You can improve the speed of refresh for large tables by using incremental refresh. This setting will apply once you've published a report to the Power BI service.

ⓘ Once you've deployed this table to the Power BI service, you won't be able to download it back to Power BI Desktop. Learn more

Table: Fact Web Telemetry Large

Incremental refresh: On

Store rows where column "EventDateSource" is in the last: 1 Years

Refresh rows where column "EventDateSource" is in the last: 30 Days

□ Detect data changes Learn more

☑ Only refresh complete days Learn more

We can use filepath column EventDate to enable incremental refresh.

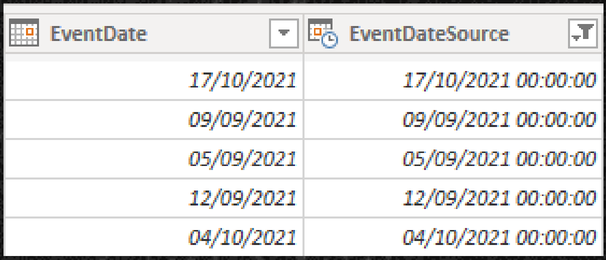
This will then enable “partition pruning” in Serverless SQL Pools to reduce data processed and increase read performance

Filtering with Incremental

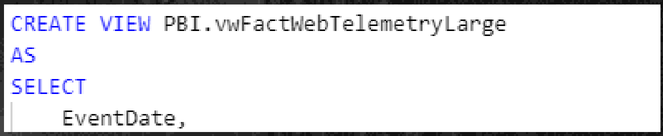
We can optimise the incremental refresh by using an existing source folder partition scheme

Efficiency

EventDateSource column is a Date column within the Parquet data

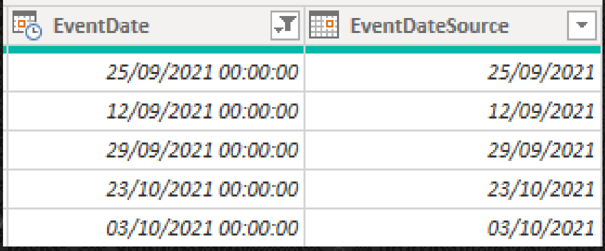


EventDate	EventDateSource
17/10/2021	17/10/2021 00:00:00
09/09/2021	09/09/2021 00:00:00
05/09/2021	05/09/2021 00:00:00
12/09/2021	12/09/2021 00:00:00
04/10/2021	04/10/2021 00:00:00

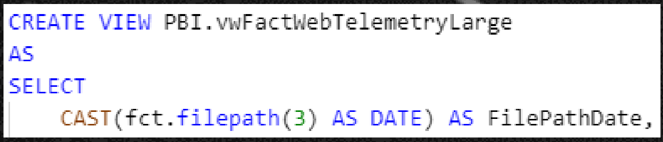


```
CREATE VIEW PBI.vwFactWebTelemetryLarge
AS
SELECT
    EventDate,
```

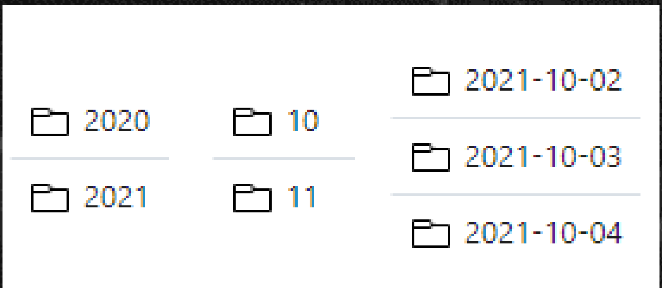
EventDate column is a Date column returned by the filepath() function



EventDate	EventDateSource
25/09/2021 00:00:00	25/09/2021
12/09/2021 00:00:00	12/09/2021
29/09/2021 00:00:00	29/09/2021
23/10/2021 00:00:00	23/10/2021
03/10/2021 00:00:00	03/10/2021



```
CREATE VIEW PBI.vwFactWebTelemetryLarge
AS
SELECT
    CAST(fct.filepath(3) AS DATE) AS FilePathDate,
```



2020
2021

10
11

2021-10-02
2021-10-03
2021-10-04

If we use the Date column from the data within the Parquet file(s) then Serverless SQL Pools needs to scan all folders and files to find the relevant data

Filtering with Incremental

We can optimise the incremental refresh by using an existing source folder partition scheme

Efficiency

Difference in Data Processed:

Incremental set for last 30 days

None-Partitioned Date Column:

Initial Refresh:

- History: 26.3GB
- Incremental: ~120GB (30 x 4)

Incremental Refresh:

- Incremental: ~120GB (30 x 4)

Partitioned Date Column:

Initial Refresh:

- History: 26.3GB
- Incremental: 17GB

Incremental Refresh:

- Incremental: 17GB

Dataflows

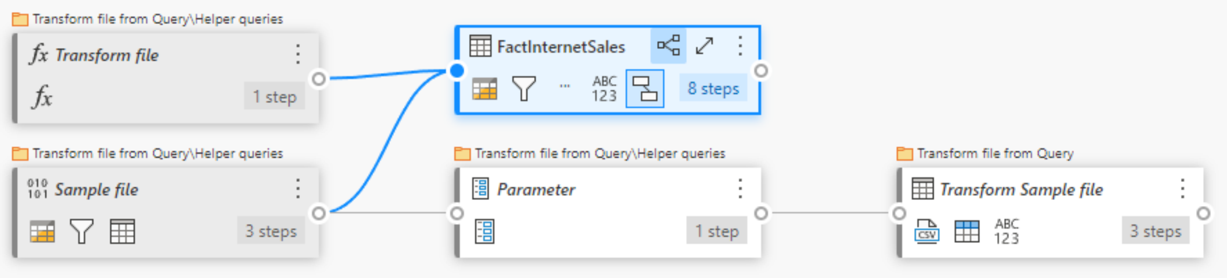
Using Serverless SQL Pools to do the "heavy lifting" for Power BI

We can use Query Fold transformations such as Grouping down to Serverless SQL Pools

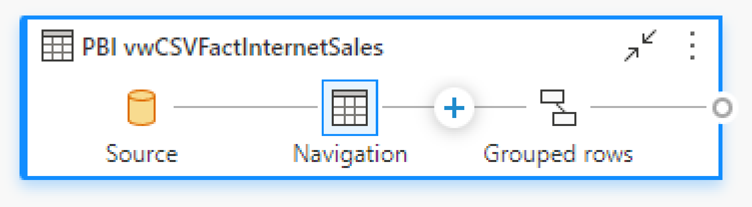
Efficiency

3 x 1.5GB CSV files (4.5GB total, 22M Rows)

Connecting to Data Lake Gen2 and using Grouping: 120K Rows


Transform file from Query\Helper queries
fx Transform file
1 step
Transform file from Query\Helper queries
Sample file
3 steps
FactInternetSales
ABC 123
8 steps
Transform file from Query\Helper queries
Parameter
1 step
Transform file from Query
Transform Sample file
ABC 123
3 steps

Connecting to Serverless SQL Pools View and using Grouping: 120K Rows

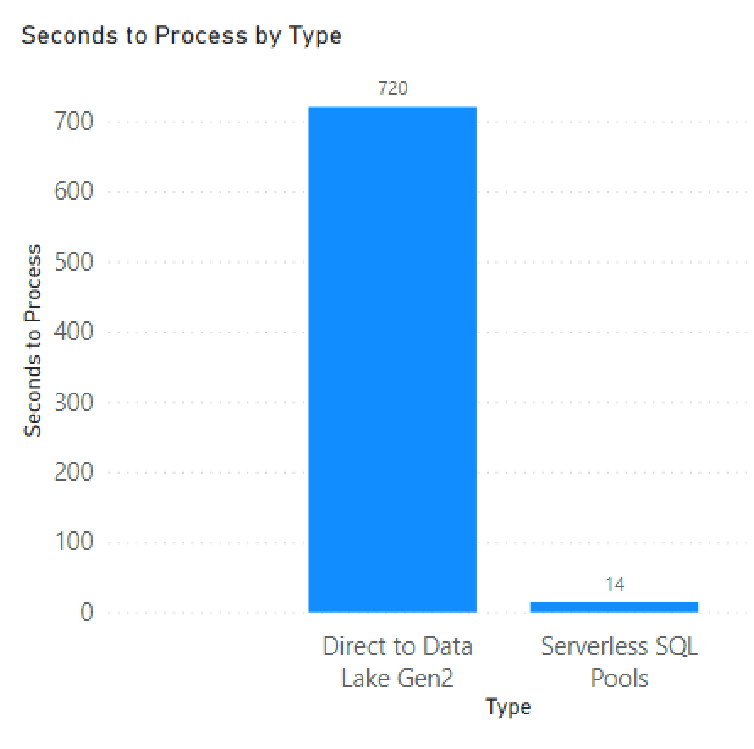


Dataflows

Using Serverless SQL Pools to do the "heavy lifting" for Power BI

We can use Query Fold transformations such as Grouping down to Serverless SQL Pools

Efficiency



Performance using Serverless SQL Pools reduced the time to process from 12 minutes to under 1 minute

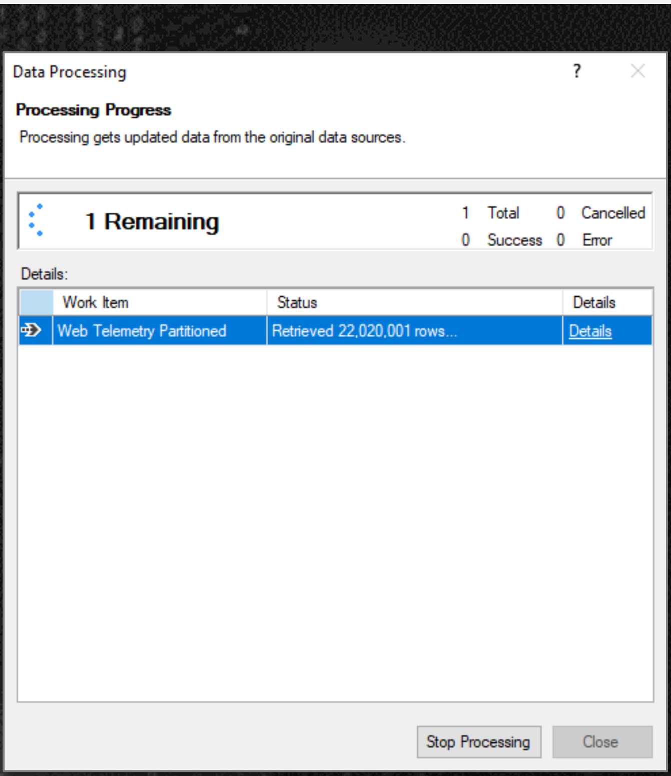
Workspace Settings:

- Premium-Per-User
- Enhanced Compute Engine Settings: On

Azure Analysis Services

We can also connect Azure Analysis Service and import data

Import to Tabular



Data Processing

Processing Progress

Processing gets updated data from the original data sources.

1 Remaining

1	Total	0	Cancelled
0	Success	0	Error

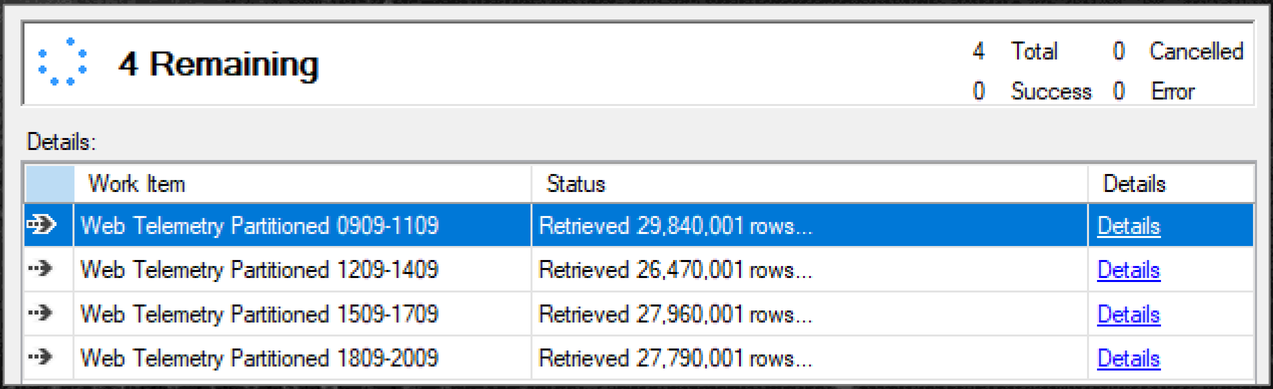
Details:

Work Item	Status	Details
Web Telemetry Partitioned	Retrieved 22,020,001 rows...	Details

We can connect to Serverless SQL Pools from Azure Analysis Services and import and model data

Azure Analysis Services can scale to 400GB RAM

In this example, the Fact table has been partitioned



4 Remaining

4	Total	0	Cancelled
0	Success	0	Error

Details:

Work Item	Status	Details
Web Telemetry Partitioned 0909-1109	Retrieved 29,840,001 rows...	Details
Web Telemetry Partitioned 1209-1409	Retrieved 26,470,001 rows...	Details
Web Telemetry Partitioned 1509-1709	Retrieved 27,960,001 rows...	Details
Web Telemetry Partitioned 1809-2009	Retrieved 27,790,001 rows...	Details

References

Icons

https://www.flaticon.com/packs/design-thinking-154
https://www.flaticon.com/packs/cloud-computing-network-7
https://www.flaticon.com/packs/business-797
https://www.flaticon.com/packs/startups-45
https://www.flaticon.com/packs/ninja-53
https://www.flaticon.com/packs/biochemistry-51
https://www.flaticon.com/packs/social-marketing-6
https://www.flaticon.com/packs/organization-10