DATA TOBOGGAN

Harnessing Azure Synapse Analytics Serverless SQL Pools in Power BI Dataflows

Azure Synapse Analytics
Data Toboggan
Saturday 30th January 2021

Andy Cutler
Lightning Session

Andy Cutler

Independent Consultant & Contractor

Azure Data Platform & Power BI

www.datahai.co.uk

https://twitter.com/MrAndyCutler

https://www.linkedin.com/in/andycutler/

SCAN ME

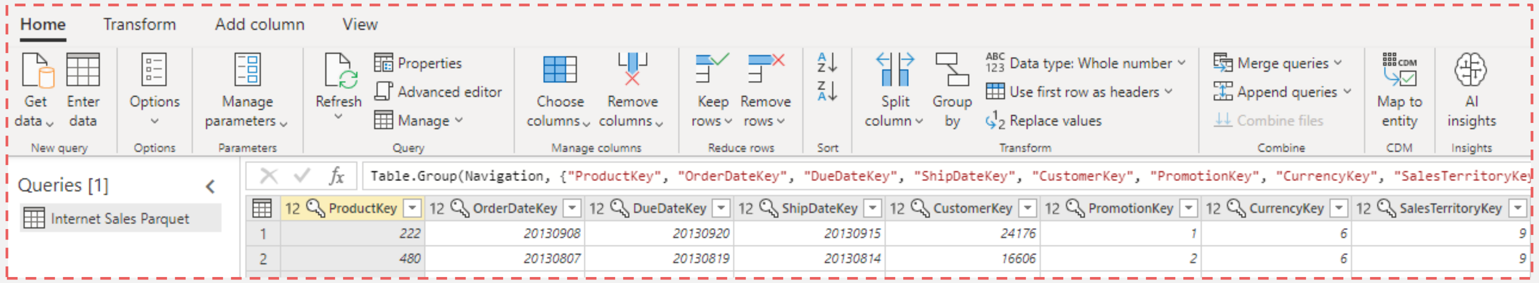
Session Overview

- What are Power BI Dataflows?
- What is Azure Synapse Analytics Serverless SQL Pools?
- Why use Serverless SQL Pools with Dataflows?
- Serverless SQL Pools and Power BI Dataflows Together
- Considerations

Power BI Dataflows

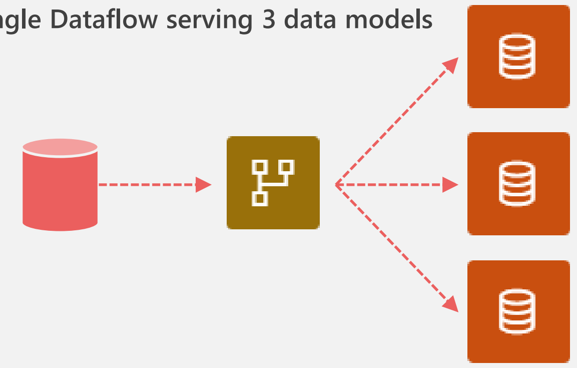
A Power Query based feature in the Power BI Service which enables:

- Connecting to a variety of data sources including SQL Databases and Data Lake Storage
- Cleansing and Transforming the data to suit requirements
- Mapping to common business entities using the Common Data Model
- Creating a centralised repository of data ready for using in Power BI Datasets



```
Table.Group(Navigation, {"ProductKey", "OrderDateKey", "DueDateKey", "ShipDateKey", "CustomerKey", "PromotionKey", "CurrencyKey", "SalesTerritoryKey
```

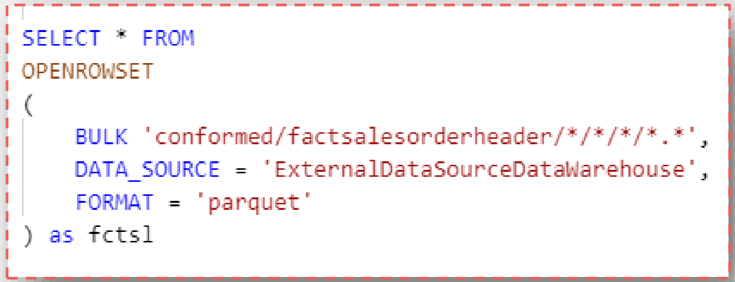
	ProductKey	OrderDateKey	DueDateKey	ShipDateKey	CustomerKey	PromotionKey	CurrencyKey	SalesTerritoryKey
1	222	20130908	20130920	20130915	24176	1	6	9
2	480	20130807	20130819	20130814	16606	2	6	9


Single Dataflow serving 3 data models

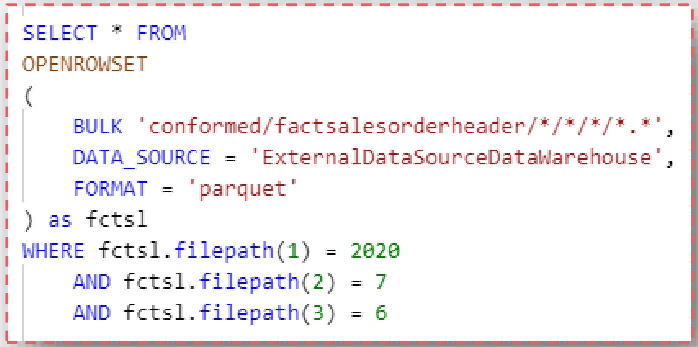
Azure Synapse Analytics Serverless SQL Pools

Cloud Analytics platform to process data in data lake storage

- Ability to query file data “in place” using T-SQL without copying data to internal storage
- Ability to write data to external data lake storage
- Connect Business Intelligence tools using SQL endpoint



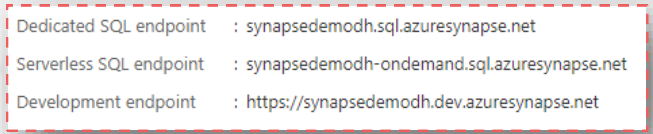
```
SELECT * FROM
OPENROWSET
(
    BULK 'conformed/factsalesorderheader/*/*/*/*.*',
    DATA_SOURCE = 'ExternalDataSourceDataWarehouse',
    FORMAT = 'parquet'
) as fctsl
```



```
SELECT * FROM
OPENROWSET
(
    BULK 'conformed/factsalesorderheader/*/*/*/*.*',
    DATA_SOURCE = 'ExternalDataSourceDataWarehouse',
    FORMAT = 'parquet'
) as fctsl
WHERE fctsl.filepath(1) = 2020
    AND fctsl.filepath(2) = 7
    AND fctsl.filepath(3) = 6
```

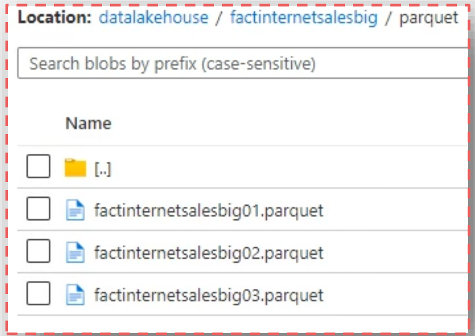
Serverless SQL Pools and Dataflows Together

Power BI can connect to the Serverless SQL endpoint just like any SQL database



Dedicated SQL endpoint	: synapsedemodh.sql.azuresynapse.net
Serverless SQL endpoint	: synapsedemodh-ondemand.sql.azuresynapse.net
Development endpoint	: https://synapsedemodh.dev.azuresynapse.net

Source data in Data Lake

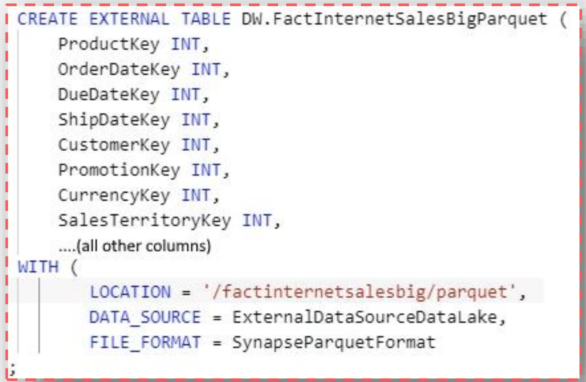


Location: datalakehouse / factinternetsalesbig / parquet

Search blobs by prefix (case-sensitive)

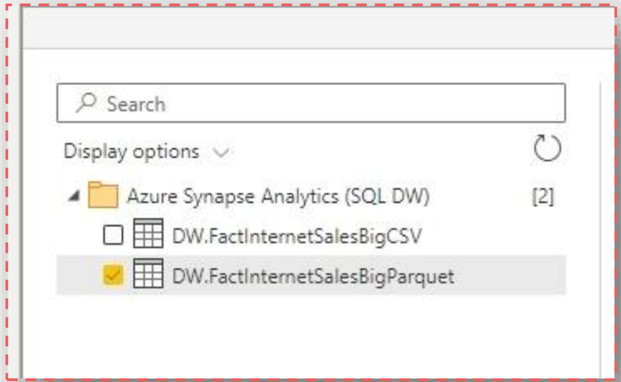
	Name
☐	[..]
☐	factinternetsalesbig01.parquet
☐	factinternetsalesbig02.parquet
☐	factinternetsalesbig03.parquet

Create table to read data from Data Lake



```
CREATE EXTERNAL TABLE DW.FactInternetSalesBigParquet (
    ProductKey INT,
    OrderDateKey INT,
    DueDateKey INT,
    ShipDateKey INT,
    CustomerKey INT,
    PromotionKey INT,
    CurrencyKey INT,
    SalesTerritoryKey INT,
    ....(all other columns)
WITH (
        LOCATION = '/factinternetsalesbig/parquet',
        DATA_SOURCE = ExternalDataSourceDataLake,
        FILE_FORMAT = SynapseParquetFormat
;
```

Connect to table within Power BI Dataflow



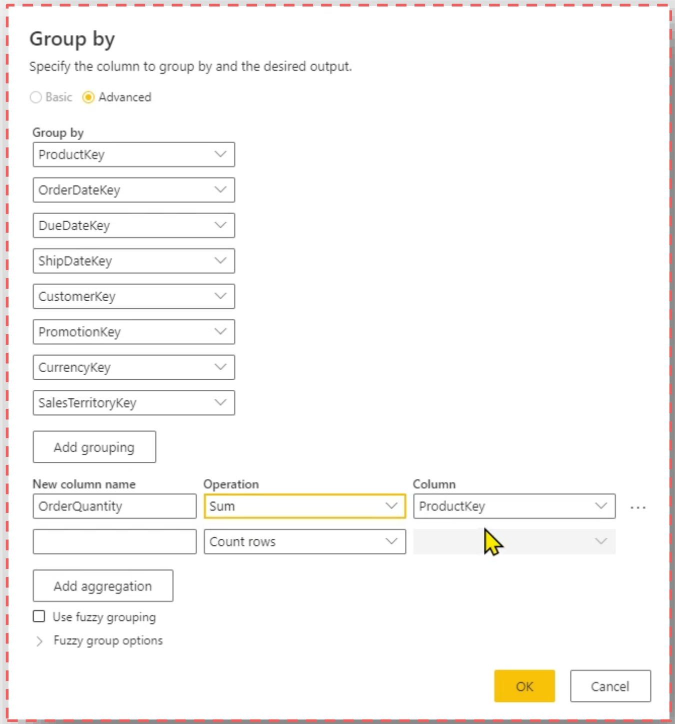
Search
Display options
Azure Synapse Analytics (SQL DW) [2]
DW.FactInternetSalesBigCSV
DW.FactInternetSalesBigParquet

- If you have CSV, Parquet or JSON files in Storage, let SQL Serverless do the data processing
- Can create a Dataflow and use Serverless SQL Pool Database and Tables as a data source
- SQL is pushed down to SQL Serverless due to Power Query's Query Folding feature

Power BI Query Folding

When transformations are applied to the dataset and can be "folded", the logic to perform the transformations is passed to the data source. In this scenarios, Serverless SQL Pool will receive a SQL statement.

The GROUP BY transformation in Dataflows



Group by

Specify the column to group by and the desired output.

( ) Basic [✓ Advanced]

Group by

- ProductKey
- OrderDateKey
- DueDateKey
- ShipDateKey
- CustomerKey
- PromotionKey
- CurrencyKey
- SalesTerritoryKey

Add grouping

New column name	Operation	Column
OrderQuantity	Sum	ProductKey
	Count rows	

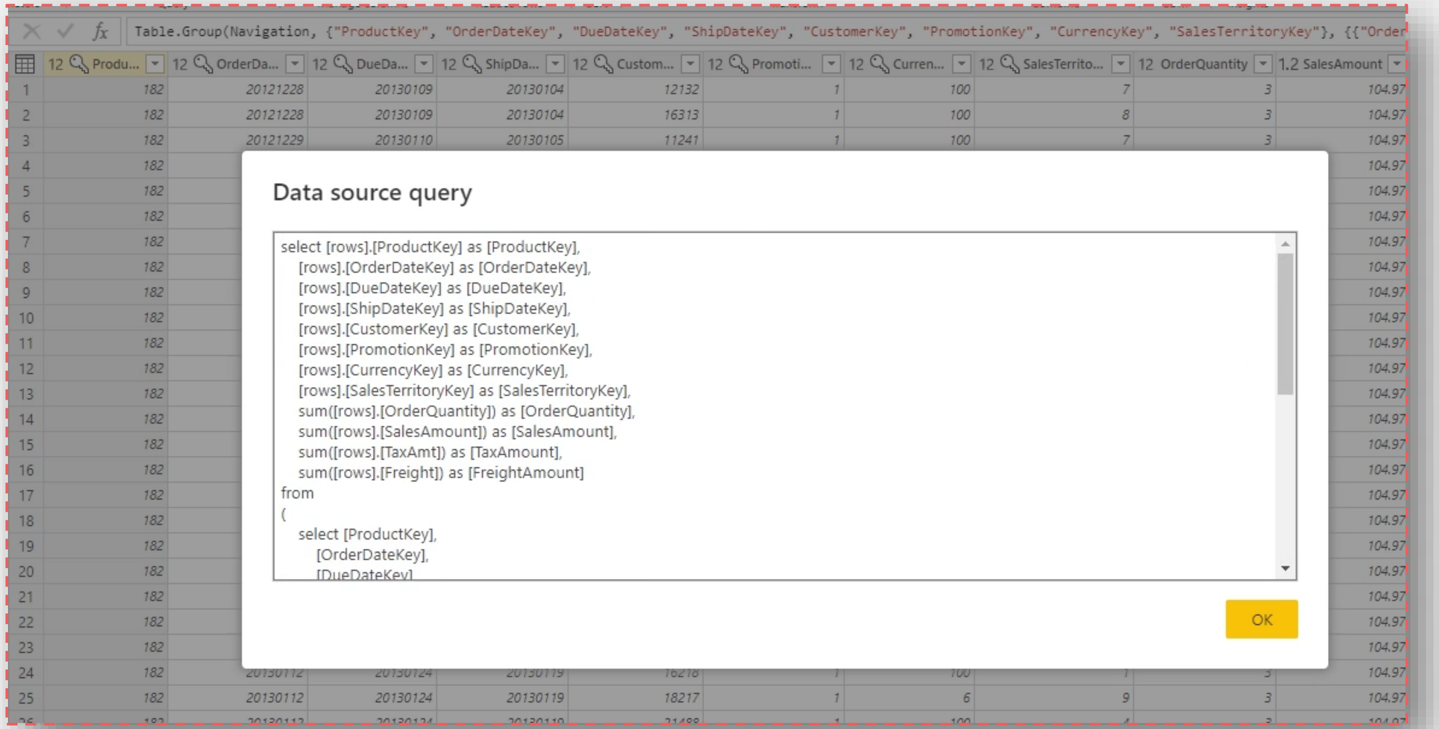
Add aggregation

☐ Use fuzzy grouping

Fuzzy group options

OK Cancel

The SQL generated by the transformation which will be sent to the Serverless SQL Pool (Query Folding)



```
Table.Group(Navigation, {"ProductKey", "OrderDateKey", "DueDateKey", "ShipDateKey", "CustomerKey", "PromotionKey", "CurrencyKey", "SalesTerritoryKey"}, {{"Order
```

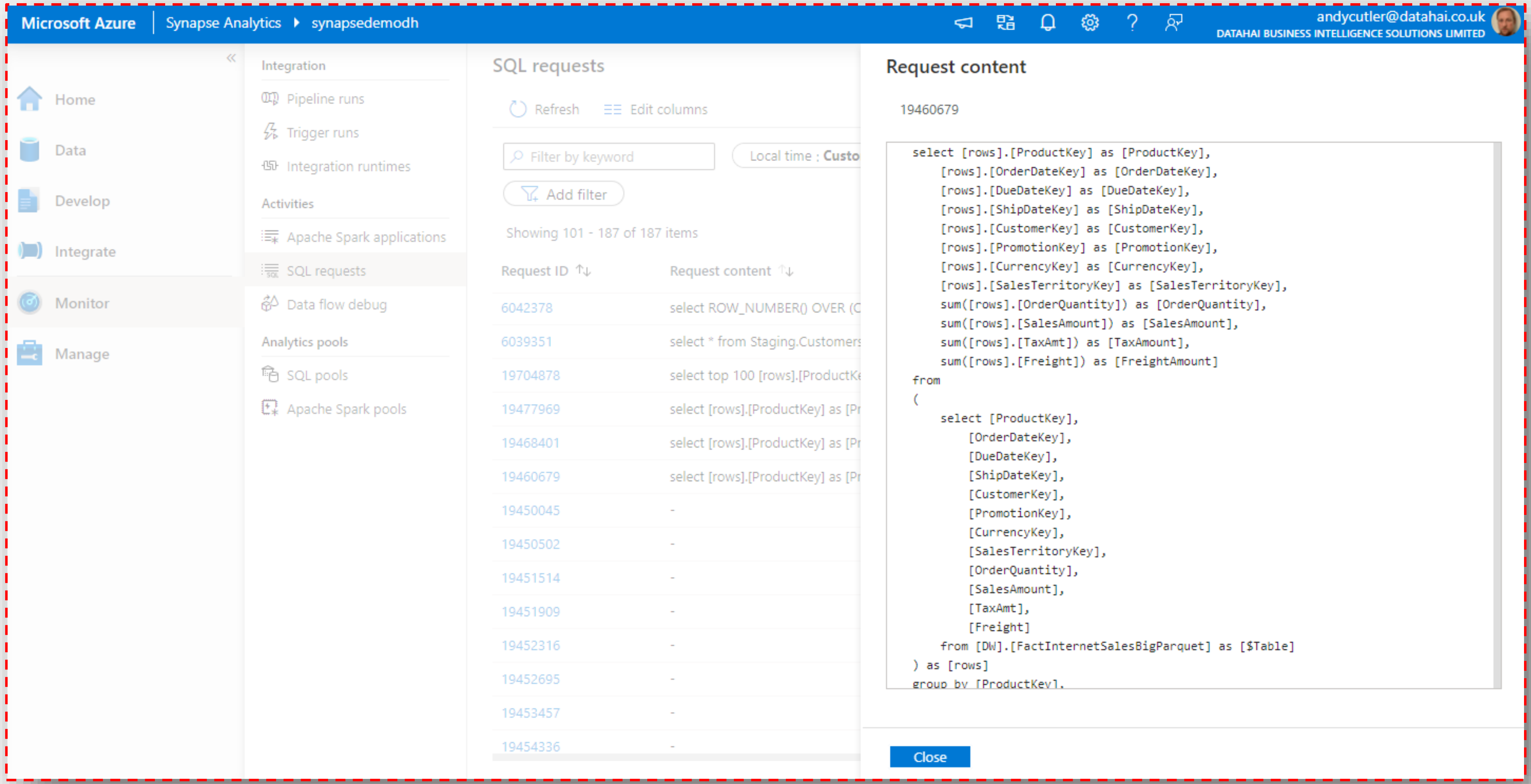
	Produ...	OrderDa...	DueDa...	ShipDa...	Custom...	Promoti...	Curren...	SalesTerrito...	OrderQuantity	SalesAmount
1	182	20121228	20130109	20130104	12132	1	100	7	3	104.97
2	182	20121228	20130109	20130104	16313	1	100	8	3	104.97
3	182	20121229	20130110	20130105	11241	1	100	7	3	104.97
4	182									104.97
5	182									104.97
6	182									104.97
7	182									104.97
8	182									104.97
9	182									104.97
10	182									104.97
11	182									104.97
12	182									104.97
13	182									104.97
14	182									104.97
15	182									104.97
16	182									104.97
17	182									104.97
18	182									104.97
19	182									104.97
20	182									104.97
21	182									104.97
22	182									104.97
23	182									104.97
24	182	20130112	20130124	20130119	16218	1	100	1	3	104.97
25	182	20130112	20130124	20130119	18217	1	6	9	3	104.97

Data source query

```
select [rows].[ProductKey] as [ProductKey],
    [rows].[OrderDateKey] as [OrderDateKey],
    [rows].[DueDateKey] as [DueDateKey],
    [rows].[ShipDateKey] as [ShipDateKey],
    [rows].[CustomerKey] as [CustomerKey],
    [rows].[PromotionKey] as [PromotionKey],
    [rows].[CurrencyKey] as [CurrencyKey],
    [rows].[SalesTerritoryKey] as [SalesTerritoryKey],
    sum([rows].[OrderQuantity]) as [OrderQuantity],
    sum([rows].[SalesAmount]) as [SalesAmount],
    sum([rows].[TaxAmt]) as [TaxAmount],
    sum([rows].[Freight]) as [FreightAmount]
from
(
    select [ProductKey],
        [OrderDateKey],
        [DueDateKey]
```

OK

Serverless SQL Pools Monitoring



SQL requests

Showing 101 - 187 of 187 items

Request ID	Request content
6042378	select ROW_NUMBER() OVER (C
6039351	select * from Staging.Customers
19704878	select top 100 [rows].[ProductKe
19477969	select [rows].[ProductKey] as [Pr
19468401	select [rows].[ProductKey] as [Pr
19460679	select [rows].[ProductKey] as [Pr
19450045	-
19450502	-
19451514	-
19451909	-
19452316	-
19452695	-
19453457	-
19454336	-

Request content

19460679

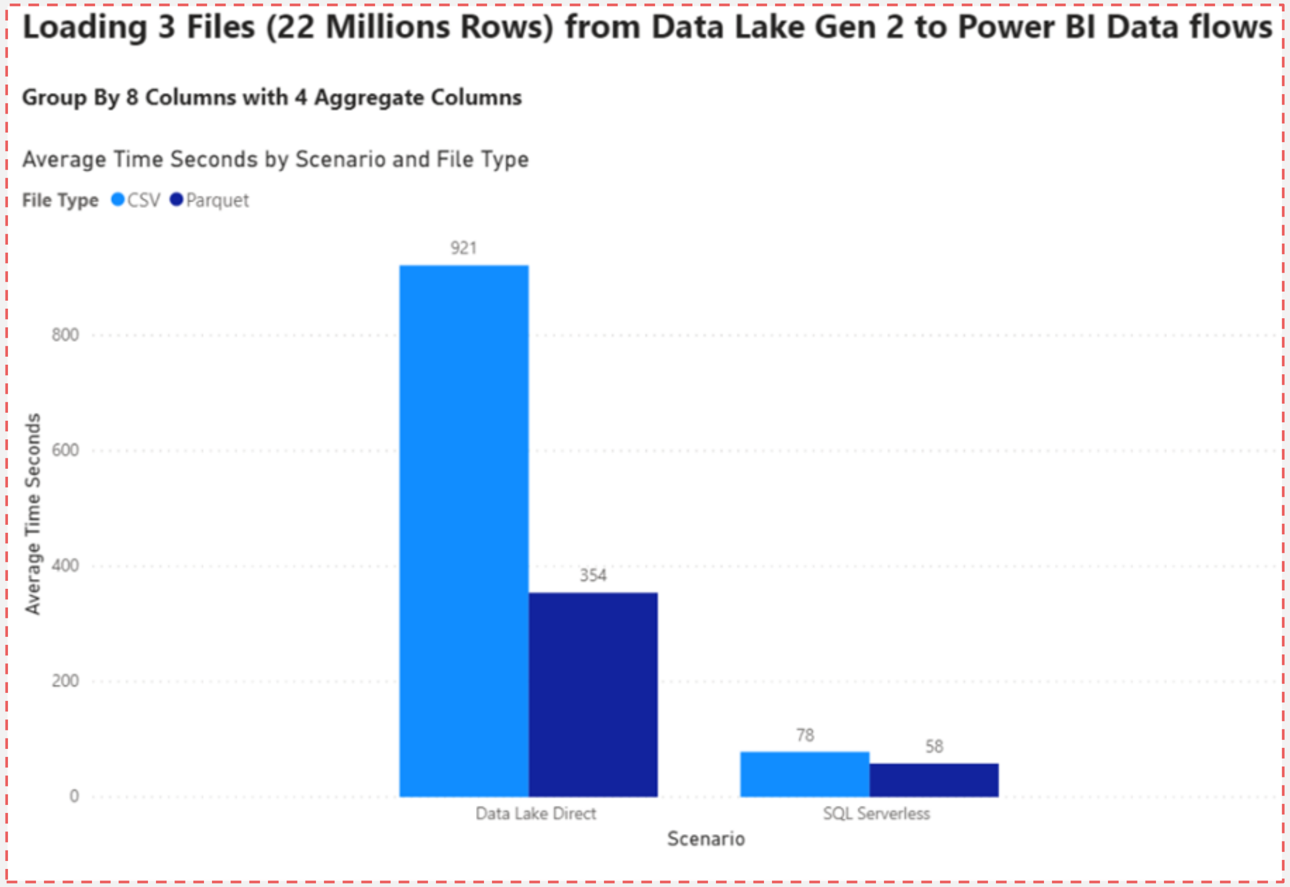
```
select [rows].[ProductKey] as [ProductKey],
    [rows].[OrderDateKey] as [OrderDateKey],
    [rows].[DueDateKey] as [DueDateKey],
    [rows].[ShipDateKey] as [ShipDateKey],
    [rows].[CustomerKey] as [CustomerKey],
    [rows].[PromotionKey] as [PromotionKey],
    [rows].[CurrencyKey] as [CurrencyKey],
    [rows].[SalesTerritoryKey] as [SalesTerritoryKey],
    sum([rows].[OrderQuantity]) as [OrderQuantity],
    sum([rows].[SalesAmount]) as [SalesAmount],
    sum([rows].[TaxAmt]) as [TaxAmount],
    sum([rows].[Freight]) as [FreightAmount]
from
(
    select [ProductKey],
        [OrderDateKey],
        [DueDateKey],
        [ShipDateKey],
        [CustomerKey],
        [PromotionKey],
        [CurrencyKey],
        [SalesTerritoryKey],
        [OrderQuantity],
        [SalesAmount],
        [TaxAmt],
        [Freight]
    from [DW].[FactInternetSalesBigParquet] as [$Table]
) as [rows]
group by [ProductKey],
```

The SQL generated by Power BI can be seen in the Monitor area of Synapse Analytics Studio.

The statistics include the query time and also the Data Processed amount, which is a vital statistic.

Data Loading Performance Analysis

Comparing the performance of Serverless SQL Pools and the Power BI native Data Lake connector



The data loading tests were carried out on CSV and Parquet data.

Total files size together: 4.5 GB

Pushing the data transformations down to the Serverless SQL Pool has reduced the time taken to load the Power BI Dataflow.

Your results may vary!

Considerations

Cost

SQL Serverless does cost money!
£3.727 per 1 Terabyte (1TB) of Data Processed

When developing/testing, use a smaller file or set of smaller files

Data at rest does not necessarily translate directly into data processed

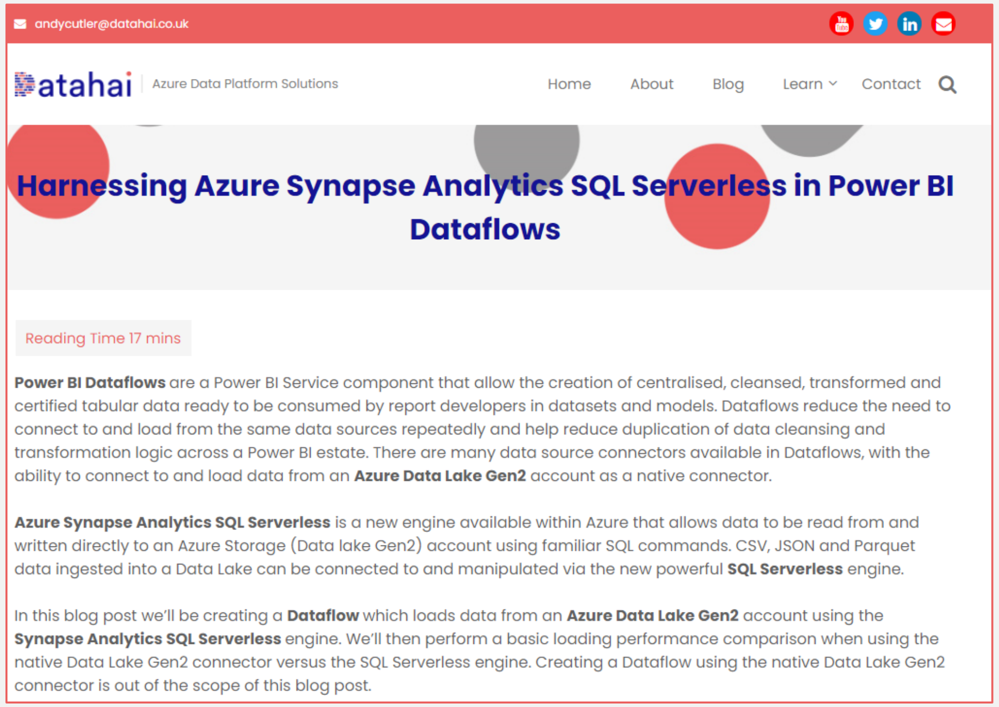
Infrastructure

Adds another service into a data architecture which will need managing

However, you can use Synapse Analytics SQL Serverless as a processing engine without any data warehousing

Reference

https://www.datahai.co.uk/power-bi/harnessing-azure-synapse-analytics-sql-serverless-in-power-bi-dataflows/



Harnessing Azure Synapse Analytics SQL Serverless in Power BI Dataflows

Reading Time 17 mins

Power BI Dataflows are a Power BI Service component that allow the creation of centralised, cleansed, transformed and certified tabular data ready to be consumed by report developers in datasets and models. Dataflows reduce the need to connect to and load from the same data sources repeatedly and help reduce duplication of data cleansing and transformation logic across a Power BI estate. There are many data source connectors available in Dataflows, with the ability to connect to and load data from an **Azure Data Lake Gen2** account as a native connector.

Azure Synapse Analytics SQL Serverless is a new engine available within Azure that allows data to be read from and written directly to an Azure Storage (Data lake Gen2) account using familiar SQL commands. CSV, JSON and Parquet data ingested into a Data Lake can be connected to and manipulated via the new powerful **SQL Serverless** engine.

In this blog post we'll be creating a **Dataflow** which loads data from an **Azure Data Lake Gen2** account using the **Synapse Analytics SQL Serverless** engine. We'll then perform a basic loading performance comparison when using the native Data Lake Gen2 connector versus the SQL Serverless engine. Creating a Dataflow using the native Data Lake Gen2 connector is out of the scope of this blog post.

SCAN ME