AZURE SYNAPSE ANALYTICS SERVERLESS SQL POOLS CHEATSHEET

Cheat Sheet Overview

This cheat sheet covers using Synapse Serverless SQL Pools with Delimited files (e.g., CSV), Parquet, and Delta.
When creating an Azure Synapse Analytics workspace, Serverless SQL Pools is enabled by default using the **in-built** engine.

SQL queries below can be run connecting to default database where user has Storage Blob Data Read access to underlying storage account

```
SELECT *
FROM
  OPENROWSET(
     BULK 'https://storage.dfs.core.windows.net/container/folder/**',
     FORMAT = 'CSV',
     FIELDTERMINATOR = ',',
     PARSER_VERSION = '2.0',
     HEADER_ROW = TRUE) AS r;

SELECT *
FROM
  OPENROWSET(
     BULK 'https://storage.dfs.core.windows.net/container/folder/**',
     FORMAT = 'PARQUET') AS r;

SELECT *
FROM
  OPENROWSET(
     BULK  'https://storage.dfs.core.windows.net/container/folder',
     FORMAT = 'DELTA') AS r;
```

Select Statements with File Metadata

As above, with metadata function to return source file names

```
SELECT *,
    r.filename() AS ParquetFileName
FROM
  OPENROWSET(
    BULK 'https://storage.dfs.core.windows.net/container/folder/**',
    FORMAT = 'PARQUET') AS r
```

Select Statements with Partition Metadata

As above, with metadata function to return partition folders. Filepath(x) indicates the folder depth in the data lake

```
SELECT *,
    r.filepath(1) AS PartitionLevel1,
    r.filepath(2) AS PartitionLevel2
FROM
  OPENROWSET(
    BULK 'https://storage.dfs.core.windows.net/container/folder/*/*/**',
    FORMAT = 'PARQUET') AS r
```

Create Database

Create database, security objects, schema, and external data source to be used with external tables and views. This cheat sheet will use Managed Identity to authorize to storage

```
CREATE DATABASE <database_name>;

CREATE MASTER KEY ENCRYPTION
BY PASSWORD = '<strong_password>';

CREATE DATABASE SCOPED CREDENTIAL <credential_name>
WITH IDENTITY='Managed Identity';

CREATE EXTERNAL DATA SOURCE <external_data_source_name>
   WITH (
      LOCATION   = 'https://storage.dfs.core.windows.net/container',
      CREDENTIAL = <credential_name>
      );

CREATE SCHEMA <schema_name AUTHORIZATION dbo;
```

Working with Delimited Files

This section includes creating file formats, external tables and views for working with delimited files in Azure storage

```
CREATE EXTERNAL FILE FORMAT tester
WITH (FORMAT_TYPE = DELIMITEDTEXT,
     FORMAT_OPTIONS(
        FIELD_TERMINATOR = ',',
        STRING_DELIMITER = '"',
        FIRST_ROW=1,
        <other_options>));
```

Create External Table using data source and file format

```
CREATE EXTERNAL TABLE <schema_name>.<table_name>
   (
      [<column_name> <data_type>,
   ...
   )
      WITH (
         LOCATION = 'rootfolder/**',
         DATA_SOURCE = <external_data_source_name>,
         FILE_FORMAT = <csv_format_name>
   );
```

Create View over data lake using CSV

```
CREATE VIEW <schema_name>.<view_name>
AS
SELECT *
FROM
  OPENROWSET(
     BULK 'https://storage.dfs.core.windows.net/container/folder/**',
     FORMAT = 'CSV',
     FIELDTERMINATOR = ',',
     PARSER_VERSION = '2.0')
AS r;
```

Create View over data lake using CSV with header column and using external data source. Views do not support file format

```
CREATE VIEW <schema_name>.<view_name>
SELECT
  <source_column_name_1>, <source_column_name_2>
FROM
  OPENROWSET(
     BULK 'rootfolder/**',
     DATA_SOURCE = '<external_data_source>',
     FORMAT = 'CSV',
     FIELDTERMINATOR = ',',
     PARSER_VERSION = '2.0',
     HEADER_ROW = TRUE)
AS r;
```

Create View over data lake using CSV with no header column using ordinal positioning and using external data source. WITH specifies data types and can be used in any View or Select

```
CREATE VIEW <schema_name>.<view_name>
SELECT
  <user_defined_column_name_1>,  <user_defined_column_name_2>
FROM
  OPENROWSET(
     BULK 'rootfolder/**',
     DATA_SOURCE = '<external_data_source>',
     FORMAT = 'CSV',
     FIELDTERMINATOR = ',',
     PARSER_VERSION = '2.0',
     HEADER_ROW = FALSE
)
WITH
(
    <user_defined_column_name_1> VARCHAR(20) 0,
    <user_defined_column_name_2> INT 1)
AS r;
```

Create Views over partitioned folders in data lake using CSV, we use the filepath() function to expose the partitioned folder and this can be used in WHERE clauses to filter

```
CREATE VIEW <schema_name>.<view_name>
SELECT
  <source_column_name_1>,  <source_column_name_2>,
  r.filepath(1) AS PartitionLevel1,  r.filepath(2) AS PartitionLevel2
FROM
  OPENROWSET(
     BULK 'rootfolder/*/*/**',
     DATA_SOURCE = '<external_data_source>',
     FORMAT = 'CSV',
     FIELDTERMINATOR = ',',
     PARSER_VERSION = '2.0',
     HEADER_ROW = TRUE)
AS r;
```

Working with Parquet Files

This section includes creating file formats, external tables and views for working with Parquet files in Azure storage

```
CREATE EXTERNAL FILE FORMAT <parquet_format_name>
WITH (FORMAT_TYPE = PARQUET);


CREATE EXTERNAL TABLE <schema_name>.<table_name>
   (
      [<column_name> <data_type>,
   ...
   )
      WITH (
         LOCATION = 'rootfolder/**',
         DATA_SOURCE = <external_data_source_name>,
         FILE_FORMAT = <parquet_format_name>
   );
```

Create Views over data lake using Parquet

```
CREATE VIEW <schema_name>.<view_name>
AS
SELECT *
FROM
  OPENROWSET(
     BULK 'https://storage.dfs.core.windows.net/container/folder/**',
     FORMAT = 'PARQUET')
AS r;
```

Create Views over data lake using Parquet and use WITH to specify data types and using external data source

```
CREATE VIEW <schema_name>.<view_name>
AS
SELECT
  <column_1_from_file>,
  <column_2_from_file>
FROM
  OPENROWSET(
     BULK 'rootfolder/**',
     DATA_SOURCE = '<external_data_source>',
     FORMAT = 'PARQUET')
WITH
(
    <column_1_from_file> VARCHAR(20),
    <column_2_from_file> INT)
AS r;
```

AZURE SYNAPSE ANALYTICS SERVERLESS SQL POOLS CHEATSHEET

Create Views over partitioned folders in data lake using Parquet and use WITH to specify data types and using external data source

```
CREATE VIEW <schema_name>.<view_name>
AS
SELECT
  <column_1_from_file>,
  <column_2_from_file>,
  r.filepath(1) AS PartitionLevel1,
  r.filepath(2) AS PartitionLevel2
FROM
  OPENROWSET(
     BULK 'rootfolder/*/*/**',
     DATA_SOURCE = '<external_data_source>',
     FORMAT = 'PARQUET')
WITH
(
    <column_1_from_file> VARCHAR(20),
    <column_2_from_file> INT)
AS r;
```

Working with Delta Files

This section includes creating file formats, external tables and views for working with the Delta format in Azure storage. Note that partitioned external tables are not supported with Delta. Views do support partitioning.

```
CREATE EXTERNAL FILE FORMAT <delta_format_name>
WITH (FORMAT_TYPE = DELTA);

CREATE EXTERNAL TABLE <schema_name>.<table_name>
   (
      [<column_name> <data_type>,
   ...
   )
      WITH (
         LOCATION = 'deltafolder/**',
         DATA_SOURCE = <external_data_source_name>,
         FILE_FORMAT = <delta_format_name>
   );
```

Create Views over data lake using Delta

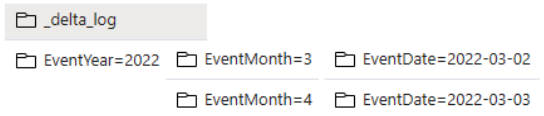
```
CREATE VIEW <schema_name>.<view_name>
AS
SELECT *
FROM
  OPENROWSET(
     BULK 'https://storage.dfs.core.windows.net/container/deltafolder',
     FORMAT = 'DELTA')
AS r;
```

Create Views over data lake using Delta and use WITH to override data types and using external data source

```
CREATE VIEW <schema_name>.<view_name>
AS
SELECT
  <column_1_from_file>,
  <column_2_from_file>
FROM
  OPENROWSET(
     BULK 'deltafolder',
     DATA_SOURCE = '<external_data_source>',
     FORMAT = 'DELTA')
WITH
(
    <column_1_from_file> VARCHAR(20),
    <column_2_from_file> INT)
AS r;
```

Create Views over partitioned data lake using Delta and use WITH to override data types and using external data source. We do not use the filepath function, we specify the folder partition names in the SELECT statement.

Example 3 level partition structure in Azure storage:




```
CREATE VIEW <schema_name>.<view_name>
AS
SELECT
  <column_1_from_file>,
  <column_2_from_file>,
  EventYear,
  EventMonth,
  EventDate
FROM
  OPENROWSET(
     BULK 'deltafolder ',
     DATA_SOURCE = '<external_data_source>',
     FORMAT  = 'DELTA')
WITH
(
    <column_1_from_file> VARCHAR(20),
    <column_2_from_file> INT,
    EventYear INT,
    EventMonth TINYINT,
    EventDate DATE)
AS r;
```

Create Stored Procedure

We can create and execute stored procedures in a Serverless SQL Pools database

```
CREATE OR ALTER PROC LDW.spGetResults
AS
BEGIN
   SELECT EventType, EventDateTime FROM LDW.vwWebTelemertyCSV
END;

EXEC LDW.spGetResults
```

Create Temp Tables

Creating temp tables is possible but with limitations. Temp tables only support inserting using VALUES or the results of a stored procedure

```
CREATE TABLE #tmpResults
(   EventType VARCHAR(20),
   EventDate DATETIME2(0));

INSERT INTO #tmpResults (EventType, EventDate)
VALUES ('Event Type','2023-03-07 15:30:54');

INSERT INTO #tmpResults EXEC LDW.spGetResults;
```

Select using External Tables

Query the external tables and views using standard T-SQL

Select from an external table, in this example we'll select all the data and use standard SQL aggregation syntax

```
SELECT
   Column1,
   Column2,
   COUNT(*) AS TotalCount
FROM <schema>.<external_table_over_csv_or_parquet_or_delta>
GROUP BY
   Column1,
   Column2;
```

Query data using a view, this works similar to querying an external table

```
SELECT
   Column1,
   Column2,
   COUNT(*) AS TotalCount
FROM <schema>.<view_over_csv_or_parquet_or_delta>
GROUP BY
   Column1,
   Column2;
```

Query partitioned Parquet / CSV files using a View that exposes the partition scheme as columns using the filepath function

```
SELECT
   Column1,
   Column2,
      COUNT(*) AS TotalCount
FROM <schema>.<view_over_parquet>
WHERE <partition_column_from_filepath> = ''value'
GROUP BY
   Column1,
   Column2;
```

Query Delta Lake using a View that exposes the partition scheme as columns and use this as a filter in the WHERE clause

```
SELECT Column1,Column2,
      COUNT(*) AS TotalCount
FROM <schema>.<view_over_delta>
WHERE <partition_column> = ''value'
GROUP BY
   Column1,
   Column2;
```

Select Objects Metadata

Show metadata from External tables including file format and external data source

```
SELECT
   et.[name] AS TableName,
   et.[location] AS TableLocation,
   ef.[name] AS FileFormatName,
   ef.[format_type] AS FileFormatType,
   es.[name] AS DataSourceName,
   es.[location] AS DataSourceLocation
FROM sys.external_tables et
INNER JOIN sys.external_file_formats ef
   ON ef.file_format_id = et.file_format_id
INNER JOIN sys.external_data_sources es
   ON es.data_source_id = et.data_source_id;
```

Get Data Processed and Usage Limits

Use system views to select usage stats and user-configured limits

```
SELECT
   [type] AS DataUsageWindow,
   data_processed_mb,
   CAST(data_processed_mb AS DECIMAL(10,3)) / 1000 AS GB,
   (CAST(data_processed_mb AS DECIMAL(10,3)) / 1000) / 1000 AS TB
FROM sys.dm_external_data_processed

SELECT
   [name] AS DataLimitWindow,
   value AS TBValue,
   CAST(value_in_use AS INT) AS TBValueInUse
FROM sys.configurations WHERE [name] LIKE 'Data processed %'
```