Azure SQL Data Warehouse LAB

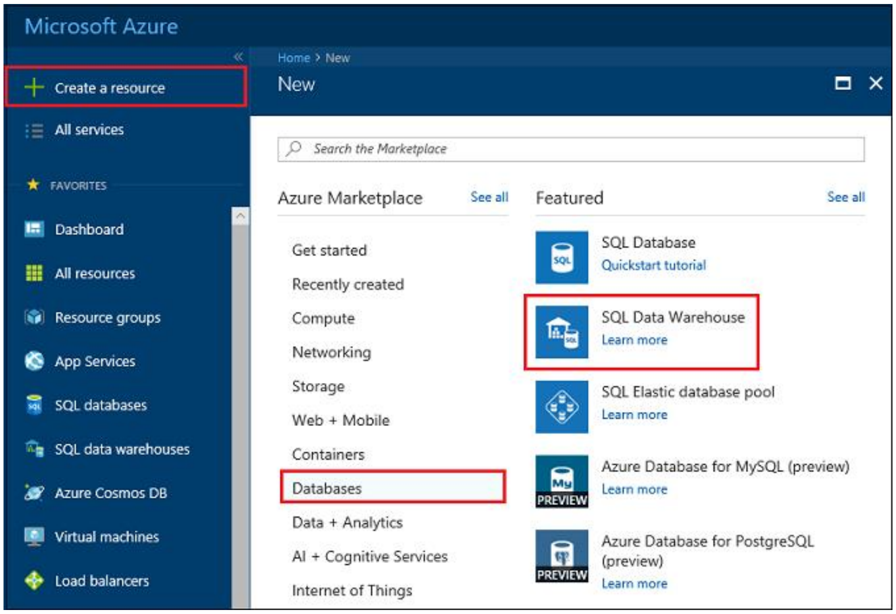
- Create a data warehouse in the Azure portal
- Set up a server-level firewall rule in the Azure portal
- Connect to the data warehouse with SSMS
- Create a user designated for loading data
- Create external tables for data in Azure blob storage
- Use the CTAS T-SQL statement to load data into your data warehouse
- View the progress of data as it is loading
- Create statistics on the newly loaded data
- Create Power BI report based on Azure SQL Data Warehouse
- Test and monitor your Azure SQL Data Warehouse
- Scale Up your SQL Data Warehouse

Create a blank SQL data warehouse

An Azure SQL data warehouse is created with a defined set of compute resources. The database is created within an Azure resource group and in an Azure SQL logical server.

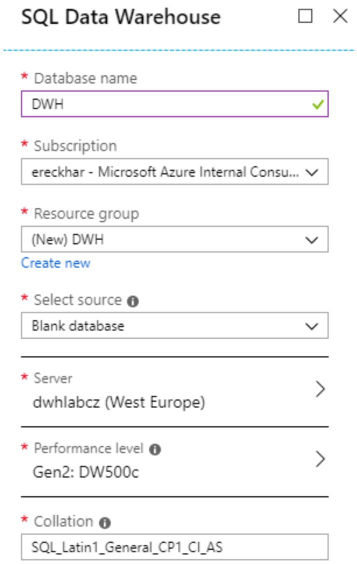
Follow these steps to create a blank SQL data warehouse:

1. Click Create a resource in the upper left-hand corner of the Azure portal.
2. Select Databases from the New page, and select SQL Data Warehouse under Featured on the New page.



3. Fill out the SQL Data Warehouse form with the following information:
   - Any DB name, e.g. DWH
   - Your resource group, e.g. DWH
   - Blank database
   - *Server (next step)*
   - Gen2: DW500c

4. Click Server to create and configure a new server for your new database. Fill out the New server form with the following information:
   - Any unique server name
   - Your admin name, e.g. ServerAdmin
   - Your password
   - Location: West EU



SQL Data Warehouse

* Database name: DWH

* Subscription: ereckhar - Microsoft Azure Internal Consu...

* Resource group: (New) DWH

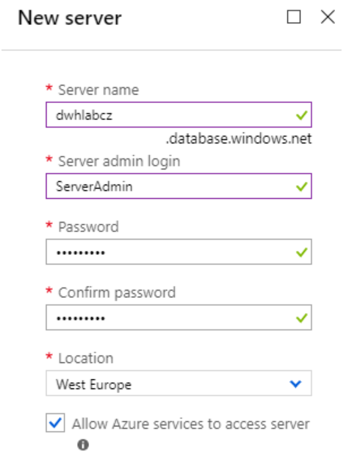
Create new

* Select source: Blank database

* Server: dwhlabcz (West Europe)

* Performance level: Gen2: DW500c

* Collation: SQL_Latin1_General_CP1_CI_AS



New server

* Server name: dwhlabcz .database.windows.net

* Server admin login: ServerAdmin

* Password: •••••••••

* Confirm password: •••••••••

* Location: West Europe

☑ Allow Azure services to access server

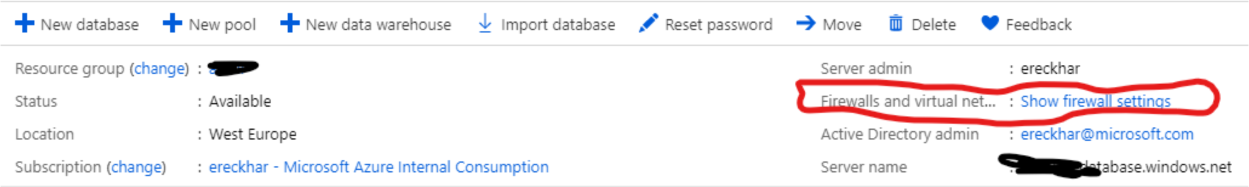
5. Click Apply.
6. Now that you have completed the SQL Database form, click Create to provision the database. Provisioning takes a few minutes.

Create a server-level firewall rule

The SQL Data Warehouse service creates a firewall at the server-level that prevents external applications and tools from connecting to the server or any databases on the server. To enable connectivity, you can add firewall rules that enable connectivity for specific IP addresses. Follow these steps to create a server-level firewall rule for your client's IP address.

Note: SQL Data Warehouse communicates over port 1433. If you are trying to connect from within a corporate network, outbound traffic over port 1433 might not be allowed by your network's firewall. If so, you cannot connect to your Azure SQL Database server unless your IT department opens port 1433.

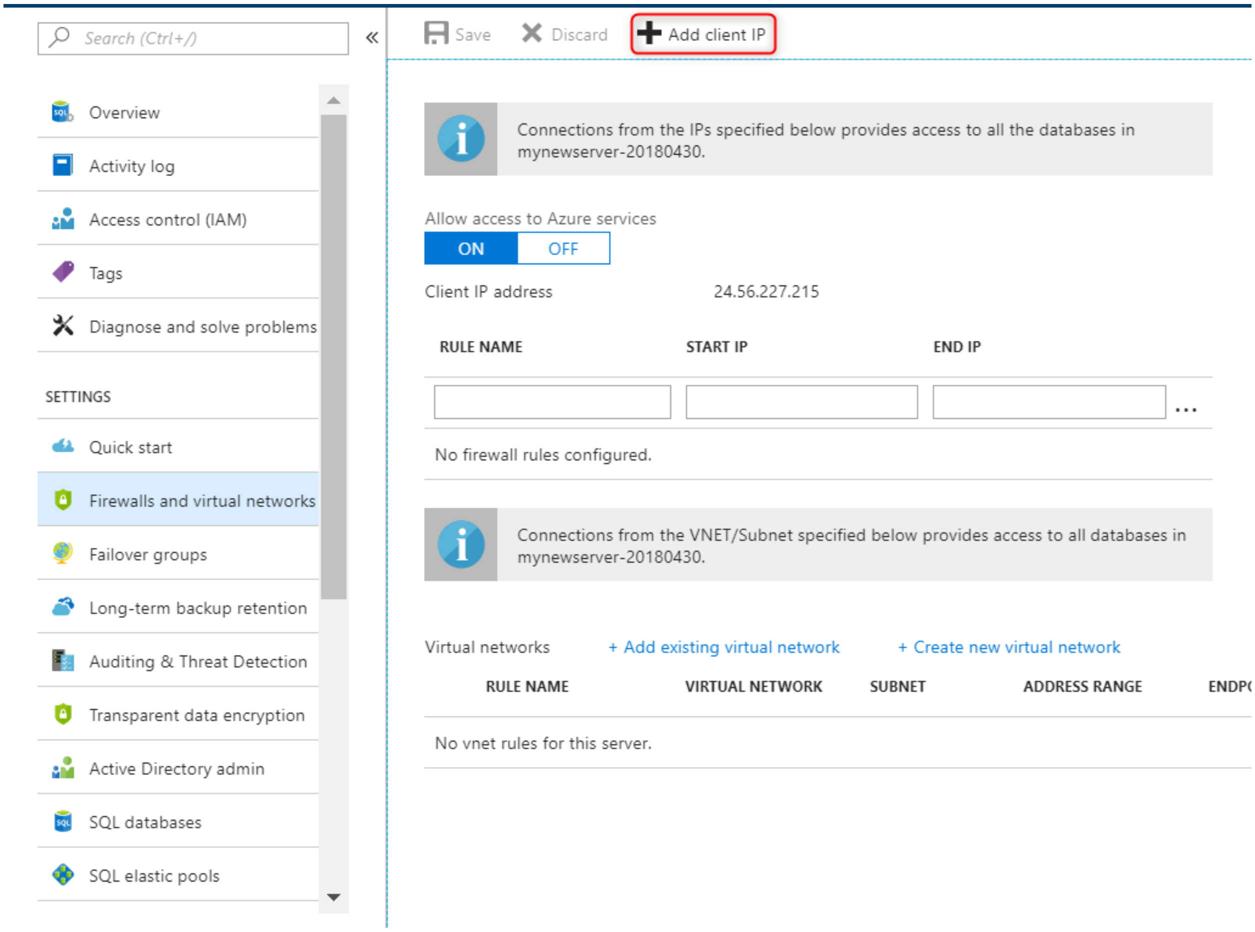
1. After the deployment completes, click SQL databases from the left-hand menu and then click DWH on the SQL databases page. The overview page for your database opens, showing you the fully qualified server name and provides options for further configuration.
2. Copy this fully qualified server name for use to connect to your server and its databases in subsequent quick starts. Then click on the server name to open server settings.
3. Click the server name to open server settings.



Resource group (change) : [redacted]
Status : Available
Location : West Europe
Subscription (change) : ereckhar - Microsoft Azure Internal Consumption

Server admin : ereckhar
Firewalls and virtual net... : Show firewall settings
Active Directory admin : ereckhar@microsoft.com
Server name [redacted]database.windows.net

4. Click Show firewall settings. The Firewall settings page for the SQL Database server opens.



Search (Ctrl+/)
«
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
SETTINGS
Quick start
Firewalls and virtual networks
Failover groups
Long-term backup retention
Auditing & Threat Detection
Transparent data encryption
Active Directory admin
SQL databases
SQL elastic pools
Save
Discard
Add client IP
+ Add existing virtual network
+ Create new virtual network

Connections from the IPs specified below provides access to all the databases in mynewserver-20180430.

Allow access to Azure services

ON | OFF

Client IP address 24.56.227.215

RULE NAME	START IP	END IP

No firewall rules configured.

Connections from the VNET/Subnet specified below provides access to all databases in mynewserver-20180430.

Virtual networks

RULE NAME	VIRTUAL NETWORK	SUBNET	ADDRESS RANGE	ENDP(

No vnet rules for this server.

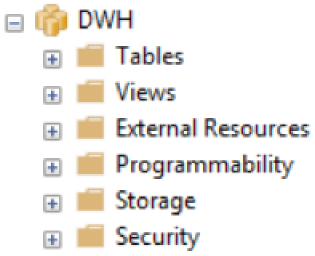
5. Click Add client IP on the toolbar to add your current IP address to a new firewall rule. A firewall rule can open port 1433 for a single IP address or a range of IP addresses.
6. Click Save. A server-level firewall rule is created for your current IP address opening port 1433 on the logical server.
7. Click OK and then close the Firewall settings page.

You can now connect to the SQL server and its data warehouses using this IP address. The connection works from SQL Server Management Studio or another tool of your choice. When you connect, use the ServerAdmin account you created previously.

Connect to the server as server admin

This section uses SQL Server Management Studio (SSMS) to establish a connection to your Azure SQL server.

1. Open SQL Server Management Studio and connect to Server
2. Click Connect. The Object Explorer window opens in SSMS.
3. In Object Explorer, expand Databases. Then expand System databases and master to view the objects in the master database. Expand DWH to view the objects in your new database.



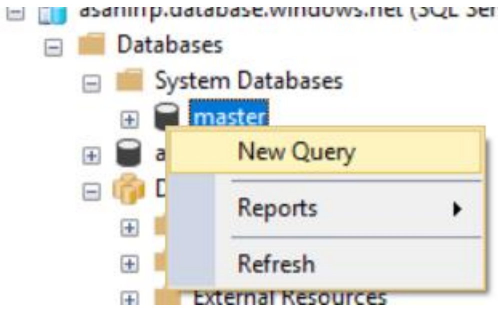
Create a user for loading data

The server admin account is meant to perform management operations, and is not suited for running queries on user data. Loading data is a memory-intensive operation. Memory maximums are defined according to which Generation of SQL Data Warehouse you've provisioned, data warehouse units, and resource class.

It's best to create a login and user that is dedicated for loading data. Then add the loading user to a resource class that enables an appropriate maximum memory allocation.

Since you are currently connected as the server admin, you can create logins and users. Use these steps to create a login and user called LargeRCuser. Then assign the user to the “largerc” dynamic resource class. And Small10RCuser with “staticrc10” static resource class.

1. In SSMS, right-click master to show a drop-down menu, and choose New Query. A new query window opens.

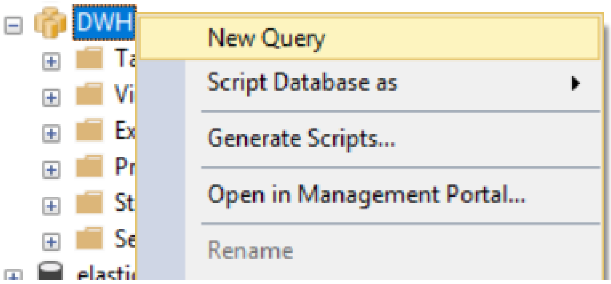


2. In the query window, enter these T-SQL commands to create a login and user (substituting your own password for 'a123STRONGpassword!').

```
CREATE LOGIN LargeRCuser WITH PASSWORD = 'a123STRONGpassword!';
CREATE USER LargeRCuser FOR LOGIN LargeRCuser;

CREATE LOGIN Small10RCuser WITH PASSWORD = 'a123STRONGpassword!';
CREATE USER Small10RCuser FOR LOGIN Small10RCuser;
```

3. Click Execute.

4. Right-click **DWH** database, and choose New Query. A new query Window opens.



5. Enter the following T-SQL commands to create a database user named for the login. The second line grants the new user CONTROL permissions on the new data warehouse. These permissions are similar to making the user the owner of the database. The third line adds the new user as a member of the DWH resource class.

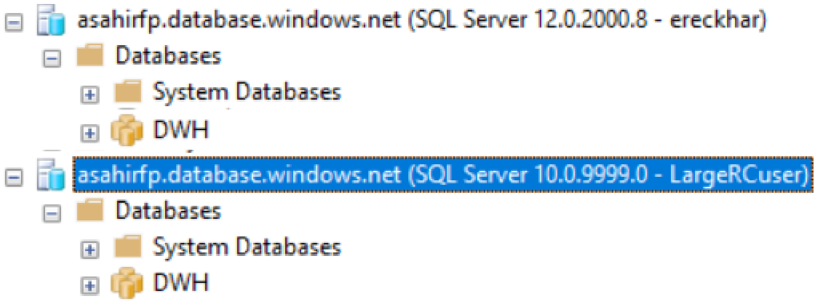
```
CREATE USER LargeRCuser FOR LOGIN LargeRCuser;
CREATE USER Small10rcUser FOR LOGIN Small10rcUser;

GRANT CONTROL ON DATABASE::[DWH] to LargeRCuser, Small10rcUser;
EXEC sp_addrolemember 'staticrc10', Small10rcUser;
EXEC sp_addrolemember 'largerc', LargeRCuser;
```

6. Click Execute.

Connect to the server as the LargeRCuser user

The first step toward loading data is to login as LargeRCuser.

1. In Object Explorer, click the Connect drop down menu and select Database Engine. The Connect to Server dialog box appears.
2. Enter the fully qualified server name, and enter LargeRCuser as the Login. Enter your password for LargeRCuser.
3. Click Connect.
4. When your connection is ready, you will see two server connections in Object Explorer. One connection as ServerAdmin and one connection as LargeRCuser.



asahirfp.database.windows.net (SQL Server 12.0.2000.8 - ereckhar)
Databases
System Databases
DWH
asahirfp.database.windows.net (SQL Server 10.0.9999.0 - LargeRCuser)
Databases
System Databases
DWH

Create external tables for the TAXI data

You are ready to begin the process of loading data into your new data warehouse. This tutorial shows you how to use external tables to load New York City taxi cab data from an Azure storage blob. For future reference, to learn how to get your data to Azure blob storage or to load it directly from your source into SQL Data Warehouse, see the loading overview.

Run the following SQL scripts specify information about the data you wish to load. This information includes where the data is located, the format of the contents of the data, and the table definition for the data.

1. In the previous section, you logged into your data warehouse as LargeRCuser. In SSMS, right-click your LargeRCuser connection and select New Query. A new query window appears.
2. Compare your query window to the previous image. Verify your new query window is running as LargeRCuser and performing queries on your DWH database. Use this query window to perform all of the loading steps.
3. Create a master key for the DWH database. You only need to create a master key once per database.

```
CREATE MASTER KEY;
```

4. Run the following CREATE EXTERNAL DATA SOURCE statement to define the location of the Azure blob. This is the location of the external taxi cab data. To run a command that you have appended to the query window, highlight the commands you wish to run and click Execute.

```
CREATE EXTERNAL DATA SOURCE NYTPublic
WITH
(
    TYPE = Hadoop,
    LOCATION = 'wasbs://2013@nytaxiblob.blob.core.windows.net/'
);
```

5. Run the following CREATE EXTERNAL FILE FORMAT T-SQL statement to specify formatting characteristics and options for the external data file. This statement specifies the external data is stored as text and the values are separated by the pipe ('|') character. The external file is compressed with Gzip.

```
CREATE EXTERNAL FILE FORMAT uncompressedcsv
WITH (
    FORMAT_TYPE = DELIMITEDTEXT,
    FORMAT_OPTIONS (
        FIELD_TERMINATOR = ',',
        STRING_DELIMITER = '',
        DATE_FORMAT = '',
        USE_TYPE_DEFAULT = False
    )
);
CREATE EXTERNAL FILE FORMAT compressedcsv
WITH (
    FORMAT_TYPE = DELIMITEDTEXT,
    FORMAT_OPTIONS ( FIELD_TERMINATOR = '|',
        STRING_DELIMITER = '',
    DATE_FORMAT = '',
        USE_TYPE_DEFAULT = False
    ),
    DATA_COMPRESSION = 'org.apache.hadoop.io.compress.GzipCodec'
);
```

6. Run the following CREATE SCHEMA statement to create a schema for your external file format. The schema provides a way to organize the external tables you are about to create.

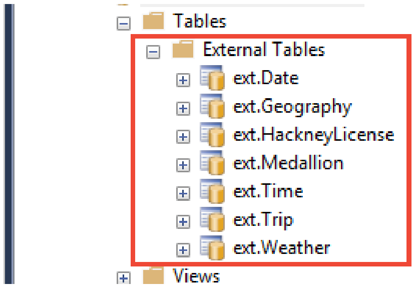
```
CREATE SCHEMA ext;
```

7. Create the external tables. The table definitions are stored in SQL Data Warehouse, but the tables reference data that is stored in Azure blob storage. Run the following T-SQL commands to create several external tables that all point to the Azure blob we defined previously in our external data source. (Elapsed time around 50sec.)

```
CREATE EXTERNAL TABLE [ext].[Date]
(
    [DateID] int NOT NULL,
    [Date] datetime NULL,
    [DateBKey] char(10) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [DayOfMonth] varchar(2) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [DaySuffix] varchar(4) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [DayName] varchar(9) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [DayOfWeek] char(1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [DayOfWeekInMonth] varchar(2) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [DayOfWeekInYear] varchar(2) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [DayOfQuarter] varchar(3) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [DayOfYear] varchar(3) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [WeekOfMonth] varchar(1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [WeekOfQuarter] varchar(2) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [WeekOfYear] varchar(2) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [Month] varchar(2) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [MonthName] varchar(9) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [MonthOfQuarter] varchar(2) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [Quarter] char(1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [QuarterName] varchar(9) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [Year] char(4) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [YearName] char(7) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [MonthYear] char(10) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [MMYYYY] char(6) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [FirstDayOfMonth] date NULL,
    [LastDayOfMonth] date NULL,
    [FirstDayOfQuarter] date NULL,
    [LastDayOfQuarter] date NULL,
    [FirstDayOfYear] date NULL,
    [LastDayOfYear] date NULL,
    [IsHolidayUSA] bit NULL,
    [IsWeekday] bit NULL,
    [HolidayUSA] varchar(50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
)
WITH
(
    LOCATION = 'Date',
    DATA_SOURCE = NYTPublic,
    FILE_FORMAT = uncompressedcsv,
    REJECT_TYPE = value,
    REJECT_VALUE = 0
);
CREATE EXTERNAL TABLE [ext].[Geography]
(
    [GeographyID] int NOT NULL,
    [ZipCodeBKey] varchar(10) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
    [County] varchar(50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [City] varchar(50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [State] varchar(50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [Country] varchar(50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [ZipCode] varchar(50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
)
WITH
(
    LOCATION = 'Geography',
    DATA_SOURCE = NYTPublic,
    FILE_FORMAT = uncompressedcsv,
    REJECT_TYPE = value,
    REJECT_VALUE = 0
);
CREATE EXTERNAL TABLE [ext].[HackneyLicense]
(
    [HackneyLicenseID] int NOT NULL,
    [HackneyLicenseBKey] varchar(50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
    [HackneyLicenseCode] varchar(50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
)
WITH
(
    LOCATION = 'HackneyLicense',
    DATA_SOURCE = NYTPublic,
    FILE_FORMAT = uncompressedcsv,
    REJECT_TYPE = value,
    REJECT_VALUE = 0
);
CREATE EXTERNAL TABLE [ext].[Medallion]
(
    [MedallionID] int NOT NULL,
    [MedallionBKey] varchar(50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
    [MedallionCode] varchar(50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
)
WITH
```

```
(
    LOCATION = 'Medallion',
    DATA_SOURCE = NYTPublic,
    FILE_FORMAT = uncompressedcsv,
    REJECT_TYPE = value,
    REJECT_VALUE = 0
)
;
CREATE EXTERNAL TABLE [ext].[Time]
(
    [TimeID] int NOT NULL,
    [TimeBKey] varchar(8) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
    [HourNumber] tinyint NOT NULL,
    [MinuteNumber] tinyint NOT NULL,
    [SecondNumber] tinyint NOT NULL,
    [TimeInSecond] int NOT NULL,
    [HourlyBucket] varchar(15) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
    [DayTimeBucketGroupKey] int NOT NULL,
    [DayTimeBucket] varchar(100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
)
WITH
(
    LOCATION = 'Time',
    DATA_SOURCE = NYTPublic,
    FILE_FORMAT = uncompressedcsv,
    REJECT_TYPE = value,
    REJECT_VALUE = 0
);
CREATE EXTERNAL TABLE [ext].[Trip]
(
    [DateID] int NOT NULL,
    [MedallionID] int NOT NULL,
    [HackneyLicenseID] int NOT NULL,
    [PickupTimeID] int NOT NULL,
    [DropoffTimeID] int NOT NULL,
    [PickupGeographyID] int NULL,
    [DropoffGeographyID] int NULL,
    [PickupLatitude] float NULL,
    [PickupLongitude] float NULL,
    [PickupLatLong] varchar(50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [DropoffLatitude] float NULL,
    [DropoffLongitude] float NULL,
    [DropoffLatLong] varchar(50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [PassengerCount] int NULL,
    [TripDurationSeconds] int NULL,
    [TripDistanceMiles] float NULL,
    [PaymentType] varchar(50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
    [FareAmount] money NULL,
    [SurchargeAmount] money NULL,
    [TaxAmount] money NULL,
    [TipAmount] money NULL,
    [TollsAmount] money NULL,
    [TotalAmount] money NULL
)
WITH
(
    LOCATION = 'Trip2013',
    DATA_SOURCE = NYTPublic,
    FILE_FORMAT = compressedcsv,
    REJECT_TYPE = value,
    REJECT_VALUE = 0
);
CREATE EXTERNAL TABLE [ext].[Weather]
(
    [DateID] int NOT NULL,
    [GeographyID] int NOT NULL,
    [PrecipitationInches] float NOT NULL,
    [AvgTemperatureFahrenheit] float NOT NULL
)
WITH
(
    LOCATION = 'Weather',
    DATA_SOURCE = NYTPublic,
    FILE_FORMAT = uncompressedcsv,
    REJECT_TYPE = value,
    REJECT_VALUE = 0
)
;
```

8. In Object Explorer, expand DWH to see the list of external tables you just created.



Tables
External Tables
ext.Date
ext.Geography
ext.HackneyLicense
ext.Medallion
ext.Time
ext.Trip
ext.Weather
Views

Load the data into your Azure SQL Data Warehouse

This section uses the external tables you just defined to load the sample data from Azure Storage Blob to SQL Data Warehouse.

Note: This tutorial loads the data directly into the final table. In a production environment, you will usually use CREATE TABLE AS SELECT to load into a staging table. While data is in the staging table you can perform any necessary transformations. To append the data in the staging table to a production table, you can use the INSERT...SELECT statement. For more information, see Best practices for loading data into Azure SQL Data Warehouse.

The script uses the CREATE TABLE AS SELECT (CTAS) T-SQL statement to load the data from Azure Storage Blob into new tables in your data warehouse. CTAS creates a new table based on the results of a select statement. The new table has the same columns and data types as the results of the select statement. When the select statement selects from an external table, SQL Data Warehouse imports the data into a relational table in the data warehouse.

1. Run the following script to load the data into new tables in your data warehouse. Elapsed time around 10-15 min.

```
CREATE TABLE [dbo].[Date]
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    CLUSTERED COLUMNSTORE INDEX
)
AS SELECT * FROM [ext].[Date]
OPTION (LABEL = 'CTAS : Load [dbo].[Date]')
;
CREATE TABLE [dbo].[Geography]
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    CLUSTERED COLUMNSTORE INDEX
)
AS
SELECT * FROM [ext].[Geography]
OPTION (LABEL = 'CTAS : Load [dbo].[Geography]')
;
CREATE TABLE [dbo].[HackneyLicense]
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    CLUSTERED COLUMNSTORE INDEX
)
AS SELECT * FROM [ext].[HackneyLicense]
OPTION (LABEL = 'CTAS : Load [dbo].[HackneyLicense]')
;
CREATE TABLE [dbo].[Medallion]
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    CLUSTERED COLUMNSTORE INDEX
)
AS SELECT * FROM [ext].[Medallion]
OPTION (LABEL = 'CTAS : Load [dbo].[Medallion]')
;
CREATE TABLE [dbo].[Time]
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    CLUSTERED COLUMNSTORE INDEX
)
AS SELECT * FROM [ext].[Time]
OPTION (LABEL = 'CTAS : Load [dbo].[Time]')
;
CREATE TABLE [dbo].[Weather]
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    CLUSTERED COLUMNSTORE INDEX
)
AS SELECT * FROM [ext].[Weather]
OPTION (LABEL = 'CTAS : Load [dbo].[Weather]')
;
CREATE TABLE [dbo].[Trip]
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    CLUSTERED COLUMNSTORE INDEX
)
AS SELECT * FROM [ext].[Trip]
OPTION (LABEL = 'CTAS : Load [dbo].[Trip]')
;
```

2. View your data as it loads. You’re loading several GBs of data and compressing it into highly performant clustered columnstore indexes. Run the following query that uses a dynamic management views (DMVs) to show the status of the load.

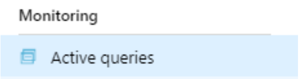
```
SELECT
    r.command,
    s.request_id,
    r.status,
    count(distinct input_name) as nbr_files,
    sum(s.bytes_processed)/1024/1024/1024.0 as gb_processed
FROM
    sys.dm_pdw_exec_requests r
    INNER JOIN sys.dm_pdw_dms_external_work s
    ON r.request_id = s.request_id
WHERE
    r.[label] = 'CTAS : Load [dbo].[Date]' OR
    r.[label] = 'CTAS : Load [dbo].[Geography]' OR
    r.[label] = 'CTAS : Load [dbo].[HackneyLicense]' OR
    r.[label] = 'CTAS : Load [dbo].[Medallion]' OR
    r.[label] = 'CTAS : Load [dbo].[Time]' OR
    r.[label] = 'CTAS : Load [dbo].[Weather]' OR
    r.[label] = 'CTAS : Load [dbo].[Trip]'
GROUP BY
    r.command,
    s.request_id,
    r.status
ORDER BY
    status desc,
    gb_processed desc;
```

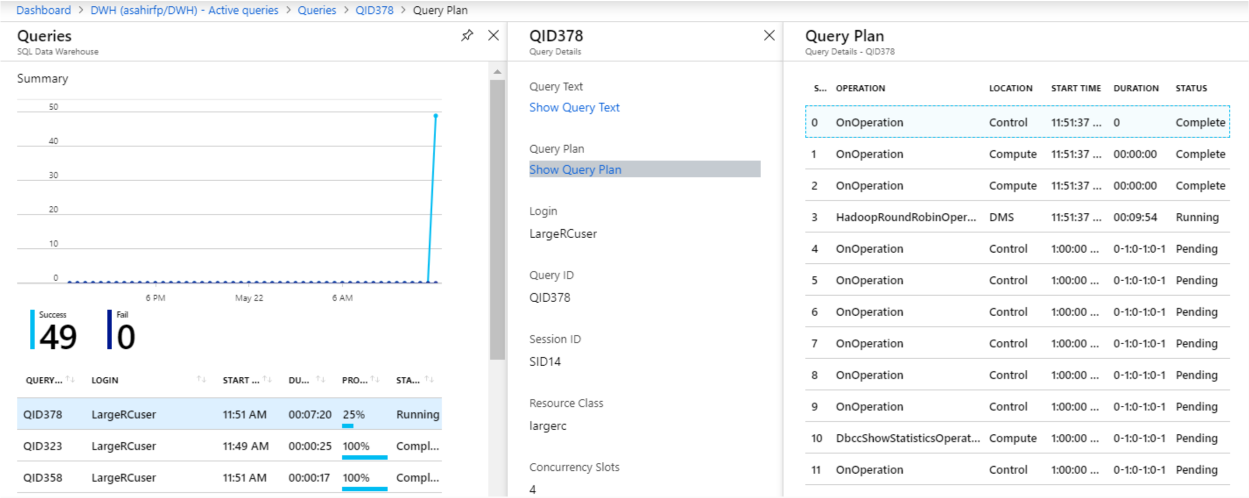
3. View all system queries.

```
SELECT * FROM sys.dm_pdw_exec_requests;
```

Note: here you can find various scripts for DWH monitoring and here you can find system views for Azure SQL DWH.

4. Use Azure Portal > your DWH > Active Queries and check Query Plan




Dashboard > DWH (asahirfp/DWH) - Active queries > Queries > QID378 > Query Plan
Queries
SQL Data Warehouse
Summary
50
40
30
20
10
0
6 PM
May 22
6 AM
Success
49
Fail
0

QUERY...	LOGIN	START ...	DU...	PRO...	STA...
QID378	LargeRCuser	11:51 AM	00:07:20	25%	Running
QID323	LargeRCuser	11:49 AM	00:00:25	100%	Compl...
QID358	LargeRCuser	11:51 AM	00:00:17	100%	Compl...

QID378
Query Details

Query Text
Show Query Text

Query Plan
Show Query Plan

Login
LargeRCuser

Query ID
QID378

Session ID
SID14

Resource Class
largerc

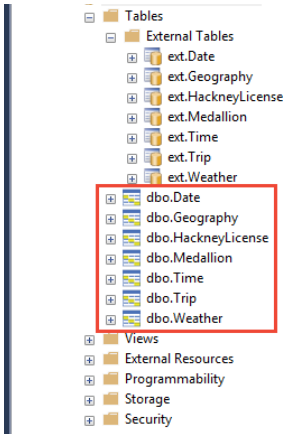
Concurrency Slots
4

Query Plan
Query Details - QID378

S...	OPERATION	LOCATION	START TIME	DURATION	STATUS
0	OnOperation	Control	11:51:37 ...	0	Complete
1	OnOperation	Compute	11:51:37 ...	00:00:00	Complete
2	OnOperation	Compute	11:51:37 ...	00:00:00	Complete
3	HadoopRoundRobinOper...	DMS	11:51:37 ...	00:09:54	Running
4	OnOperation	Control	1:00:00 ...	0-1:0-1:0-1	Pending
5	OnOperation	Control	1:00:00 ...	0-1:0-1:0-1	Pending
6	OnOperation	Control	1:00:00 ...	0-1:0-1:0-1	Pending
7	OnOperation	Control	1:00:00 ...	0-1:0-1:0-1	Pending
8	OnOperation	Control	1:00:00 ...	0-1:0-1:0-1	Pending
9	OnOperation	Control	1:00:00 ...	0-1:0-1:0-1	Pending
10	DbccShowStatisticsOperat...	Compute	1:00:00 ...	0-1:0-1:0-1	Pending
11	OnOperation	Control	1:00:00 ...	0-1:0-1:0-1	Pending

5. Check your tables and data

```
EXEC sp_spaceused N'dbo.date';
EXEC sp_spaceused N'dbo.geography';
EXEC sp_spaceused N'dbo.hackneylicense';
EXEC sp_spaceused N'dbo.medallion';
EXEC sp_spaceused N'dbo.time';
EXEC sp_spaceused N'dbo.trip';
EXEC sp_spaceused N'dbo.weather';
```



- Tables
  - External Tables
    - ext.Date
    - ext.Geography
    - ext.HackneyLicense
    - ext.Medallion
    - ext.Time
    - ext.Trip
    - ext.Weather
  - dbo.Date
  - dbo.Geography
  - dbo.HackneyLicense
  - dbo.Medallion
  - dbo.Time
  - dbo.Trip
  - dbo.Weather
- Views
- External Resources
- Programmability
- Storage
- Security

Create statistics on newly loaded data

To achieve high query performance, it's important to create statistics on each column of each table after the first load. It's also important to update statistics after substantial changes in the data, see more information here. New version of Azure SQL DWH supports auto creation and auto update.

1. Create this stored procedure that updates statistics on all columns of all tables.

```
CREATE PROCEDURE    [dbo].[prc_sqldw_create_stats]
(   @create_type    tinyint -- 1 default 2 Fullscan 3 Sample
,   @sample_pct     tinyint
)
AS

IF @create_type IS NULL
BEGIN
    SET @create_type = 1;
END;

IF @create_type NOT IN (1,2,3)
BEGIN
    THROW 151000,'Invalid value for @stats_type parameter. Valid range 1 (default), 2
(fullscan) or 3 (sample).',1;
END;

IF @sample_pct IS NULL
BEGIN;
    SET @sample_pct = 20;
END;

IF OBJECT_ID('tempdb..#stats_ddl') IS NOT NULL
BEGIN;
    DROP TABLE #stats_ddl;
END;

CREATE TABLE #stats_ddl
WITH    (   DISTRIBUTION    = HASH([seq_nmbr])
        ,   LOCATION        = USER_DB
        )
AS
WITH T
AS
(
SELECT      t.[name]                        AS [table_name]
,           s.[name]                        AS [table_schema_name]
,           c.[name]                        AS [column_name]
,           c.[column_id]                   AS [column_id]
,           t.[object_id]                   AS [object_id]
,           ROW_NUMBER()
            OVER(ORDER BY (SELECT NULL))    AS [seq_nmbr]
FROM        sys.[tables] t
JOIN        sys.[schemas] s         ON  t.[schema_id]       = s.[schema_id]
JOIN        sys.[columns] c         ON  t.[object_id]       = c.[object_id]
LEFT JOIN   sys.[stats_columns] l   ON  l.[object_id]       = c.[object_id]
                                    AND l.[column_id]       = c.[column_id]
                                    AND l.[stats_column_id] = 1
LEFT JOIN    sys.[external_tables] e    ON    e.[object_id]        = t.[object_id]
WHERE       l.[object_id] IS NULL
AND            e.[object_id] IS NULL -- not an external table
)
SELECT  [table_schema_name]
,       [table_name]
,       [column_name]
,       [column_id]
,       [object_id]
,       [seq_nmbr]
,       CASE @create_type
        WHEN 1
        THEN    CAST('CREATE STATISTICS '+QUOTENAME('stat_'+table_schema_name+ '_' +
table_name + '_'+column_name)+' ON
'+QUOTENAME(table_schema_name)+'.'+QUOTENAME(table_name)+'('+QUOTENAME(column_name)+')' AS
VARCHAR(8000))
        WHEN 2
        THEN    CAST('CREATE STATISTICS '+QUOTENAME('stat_'+table_schema_name+ '_' +
table_name + '_'+column_name)+' ON
'+QUOTENAME(table_schema_name)+'.'+QUOTENAME(table_name)+'('+QUOTENAME(column_name)+') WITH
FULLSCAN' AS VARCHAR(8000))
        WHEN 3
        THEN    CAST('CREATE STATISTICS '+QUOTENAME('stat_'+table_schema_name+ '_' +
table_name + '_'+column_name)+' ON
'+QUOTENAME(table_schema_name)+'.'+QUOTENAME(table_name)+'('+QUOTENAME(column_name)+') WITH
SAMPLE '+CONVERT(varchar(4),@sample_pct)+' PERCENT' AS VARCHAR(8000))
        END AS create_stat_ddl
FROM T
;

DECLARE @i INT              = 1
,       @t INT              = (SELECT COUNT(*) FROM #stats_ddl)
,       @s NVARCHAR(4000)   = N''
;

WHILE @i <= @t
BEGIN
    SET @s=(SELECT create_stat_ddl FROM #stats_ddl WHERE seq_nmbr = @i);
    PRINT @s
    EXEC sp_executesql @s
    SET @i+=1;
END

DROP TABLE #stats_ddl;
```

2. Run this command to create statistics on all columns of all tables in the data warehouse. Elapsed time around 2 min.

```
EXEC [dbo].[prc_sqldw_create_stats] 1, NULL;
```

Test and monitor your Azure SQL Data Warehouse

1. Run below query from SSMS as LargeRCuser

```
SELECT
[t2].[State],
[t1].[MonthYear],
SUM([t0].[TripDistanceMiles]) sum_Distance,
AVG([t0].[TripDistanceMiles]) avg_Distance,
COUNT_BIG([t0].[DateID]) count_Trips,
SUM([t0].[FareAmount]) sum_Fare,
SUM([t0].[TaxAmount]) sum_Tax,
SUM([t0].[TipAmount]) sum_Tip,
SUM([t0].[TollsAmount]) sum_Tolls,
SUM([t0].[TotalAmount]) sum_Total
FROM
(
((select [$Table].[DateID] as [DateID],
    [$Table].[PickupGeographyID] as [PickupGeographyID],
    [$Table].[TripDistanceMiles] as [TripDistanceMiles],
    [$Table].[FareAmount] as [FareAmount],
    [$Table].[TaxAmount] as [TaxAmount],
    [$Table].[TipAmount] as [TipAmount],
    [$Table].[TollsAmount] as [TollsAmount],
    [$Table].[TotalAmount] as [TotalAmount]
from [dbo].[Trip] as [$Table]) AS [t0]

 left outer join

(select [$Table].[DateID] as [DateID],
    [$Table].[MonthName] as [MonthName],
    [$Table].[MonthYear] as [MonthYear]
from [dbo].[Date] as [$Table]) AS [t1] on
( [t0].[DateID] = [t1].[DateID] )
)

 left outer join

(select [$Table].[GeographyID] as [GeographyID],
    [$Table].[State] as [State]
from [dbo].[Geography] as [$Table]) AS [t2] on
( [t0].[PickupGeographyID] = [t2].[GeographyID] )
)
WHERE
```

```
(( [t2].[State] IN ('NJ','NY') )
 AND
(( [t1].[MonthName] IN ('July','October','August','December','February')))
)
GROUP BY [t1].[MonthYear], [t2].[State]
ORDER BY 4 desc
OPTION (LABEL = 'My Report')
```

2. Find performance stats and exec. plan for “My Report”

```
-- Find a query with the Label
SELECT  *
FROM    sys.dm_pdw_exec_requests
WHERE   [label] = 'My Report';
```

```
-- Find the distributed query plan steps for a specific query.
-- Replace request_id with value from Step 1.
SELECT * FROM sys.dm_pdw_request_steps
WHERE request_id = 'QIDxxxx'
ORDER BY step_index;
```

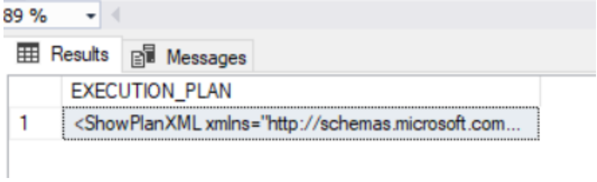
When a DSQL plan is taking longer than expected, the cause can be a complex plan with many DSQL steps or just one step taking a long time. If the plan is many steps with several move operations, consider optimizing your table distributions to reduce data movement. The Table distribution article explains why data must be moved to solve a query and explains some distribution strategies to minimize data movement.

```
-- Find the distribution run times for a SQL step: for SQL operations: OnOperation,
RemoteOperation, ReturnOperation.
SELECT * FROM sys.dm_pdw_request_steps
WHERE request_id = 'QIDxxxx'
and operation_type in ('OnOperation','RemoteOperation','ReturnOperation')
ORDER BY step_index;
```

```
-- Replace request_id and step_index with values from Step 1 and 3.
SELECT * FROM sys.dm_pdw_sql_requests
WHERE request_id = 'QIDxxxx' AND step_index = x ;
```

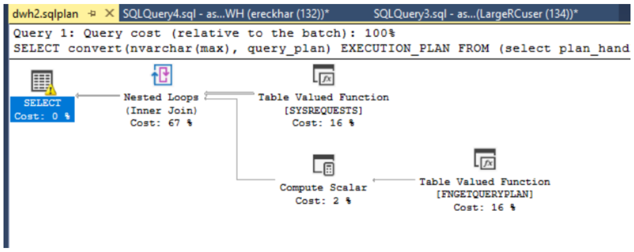
```
-- Find the SQL Server execution plan for a query running on a specific SQL Data
Warehouse Compute or Control node.
-- Replace distribution_id and spid with values from previous query.
-- DBCC PDW_SHOWEXECUTIONPLAN (distribution_id, spid)
DBCC PDW_SHOWEXECUTIONPLAN(xx, xxxx);
```

Copy Execution plan into Notepad and save as dwh.sqlplan.



	EXECUTION_PLAN
1	<ShowPlanXML xmlns="http://schemas.microsoft.com...

Open dwh.sqlplan in SSMS and check execution plan


dwh2.sqlplan
SQLQuery4.sql - as...WH (ereckhar (132))*
SQLQuery3.sql - as...(LargeRCuser (134))*

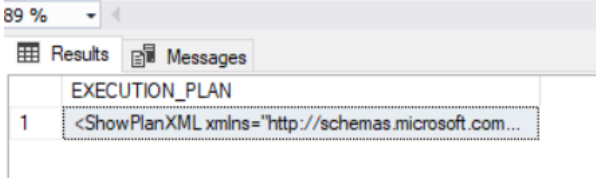
```
Query 1: Query cost (relative to the batch): 100%
SELECT convert(nvarchar(max), query_plan) EXECUTION_PLAN FROM (select plan_hand
```

```
--##################################################################
-- Find the distributed query plan steps for a specific query.
-- Replace request_id with value from Step 1.
SELECT * FROM sys.dm_pdw_request_steps
WHERE request_id = 'QIDxxxx'
ORDER BY step_index;


-- Find the information about all the workers completing a Data Movement Step: for Data
Movement operations: ShuffleMoveOperation, BroadcastMoveOperation, TrimMoveOperation,
PartitionMoveOperation, MoveOperation, CopyOperation
SELECT * FROM sys.dm_pdw_request_steps
WHERE request_id = 'QIDxxxx'
and operation_type in ('ShuffleMoveOperation', 'BroadcastMoveOperation',
'TrimMoveOperation', 'PartitionMoveOperation', 'MoveOperation', 'CopyOperation')
ORDER BY step_index;


-- Replace request_id and step_index with values from Step 1 and 3.
SELECT * FROM sys.dm_pdw_dms_workers
WHERE request_id = 'QIDxxxx' AND step_index = 8;
```

Check the total_elapsed_time column to see if a particular distribution is taking significantly longer than others for data movement. For the long-running distribution, check the rows_processed column to see if the number of rows being moved from that distribution is significantly larger than others. If so, this finding might indicate skew of your underlying data.

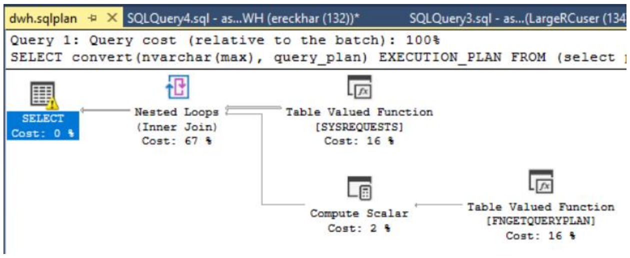
```
-- Find the SQL Server estimated plan for a query running on a specific SQL Data
Warehouse Compute or Control node.
-- Replace distribution_id and sql_spid with values from previous query.
-- DBCC PDW_SHOWEXECUTIONPLAN ( distribution_id, sql_spid )
DBCC PDW_SHOWEXECUTIONPLAN(xx, xxxx);
```

Copy Execution plan into Notepad and save as dwh.sqlplan.



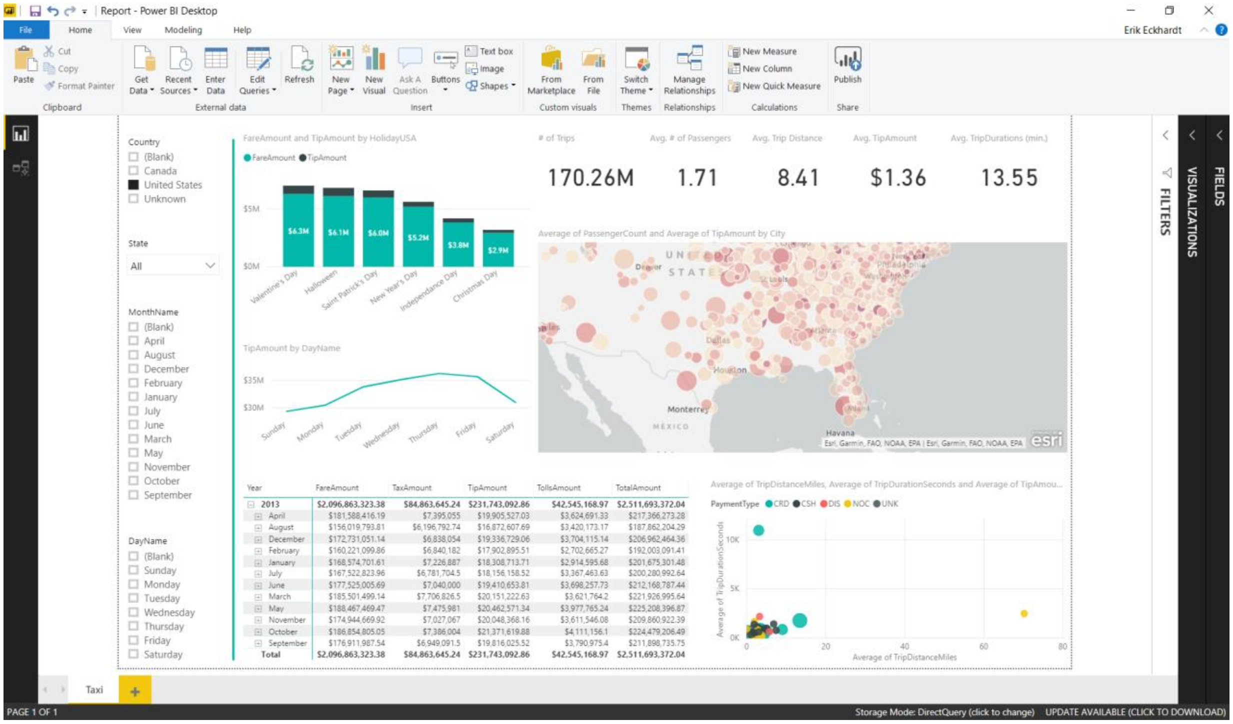
	EXECUTION_PLAN
1	<ShowPlanXML xmlns="http://schemas.microsoft.com...

Open dwh.sqlplan in SSMS and check execution plan



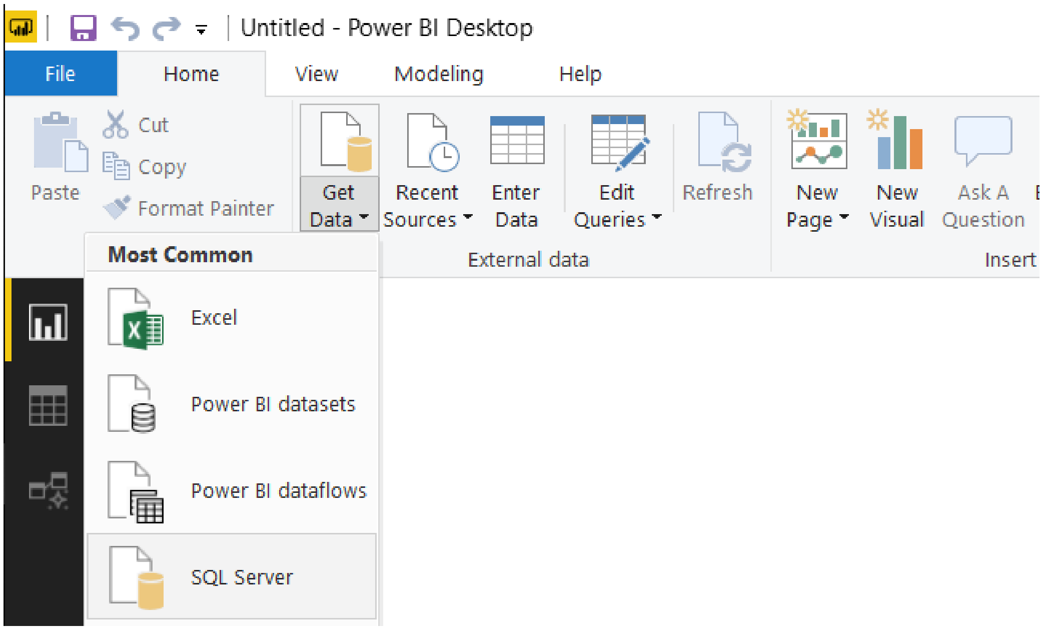
```
Query 1: Query cost (relative to the batch): 100%
SELECT convert(nvarchar(max), query_plan) EXECUTION_PLAN FROM (select
```

Create Power BI report based on Azure SQL Data Warehouse

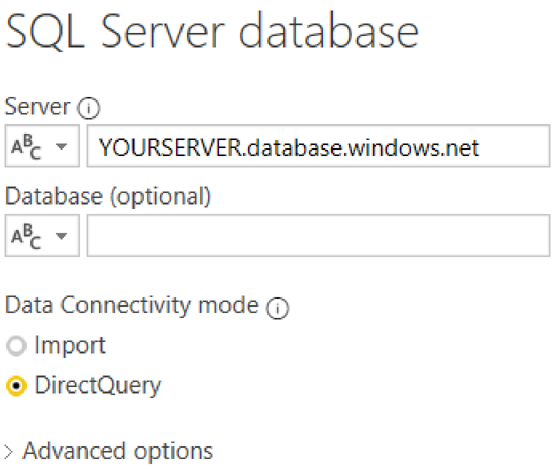


Year	FareAmount	TaxAmount	TipAmount	TollsAmount	TotalAmount
2013	$2,096,863,323.38	$84,863,645.24	$231,743,092.86	$42,545,168.97	$2,511,693,372.04
April	$181,588,416.19	$7,395,055	$19,905,527.03	$3,624,691.33	$217,366,273.28
August	$156,019,793.81	$6,196,792.74	$16,872,607.69	$3,420,173.17	$187,862,204.29
December	$172,731,051.14	$6,838,054	$19,336,729.06	$3,704,115.14	$206,962,464.36
February	$160,221,099.86	$6,840,182	$17,902,895.51	$2,702,665.27	$192,003,091.41
January	$168,574,701.61	$7,226,887	$18,308,713.71	$2,914,595.68	$201,675,301.48
July	$167,522,823.96	$6,781,704.5	$18,156,158.52	$3,367,463.63	$200,280,992.64
June	$177,525,005.69	$7,040,000	$19,410,653.81	$3,698,257.73	$212,168,787.44
March	$185,501,499.14	$7,706,826.5	$20,151,222.63	$3,621,764.2	$221,926,995.64
May	$188,467,469.47	$7,475,981	$20,462,571.34	$3,977,765.24	$225,208,396.87
November	$174,944,669.92	$7,027,067	$20,048,368.16	$3,611,546.08	$209,860,922.39
October	$186,854,805.05	$7,386,004	$21,371,619.88	$4,111,156.1	$224,479,206.49
September	$176,911,987.54	$6,949,091.5	$19,816,025.52	$3,790,975.4	$211,898,735.75
Total	$2,096,863,323.38	$84,863,645.24	$231,743,092.86	$42,545,168.97	$2,511,693,372.04

1. Download and install Power BI Desktop.
2. Connect to SQL Data Warehouse



3. Add your credentials and use DirectQuery mode



SQL Server database

Server ⓘ

YOURSERVER.database.windows.net

Database (optional)

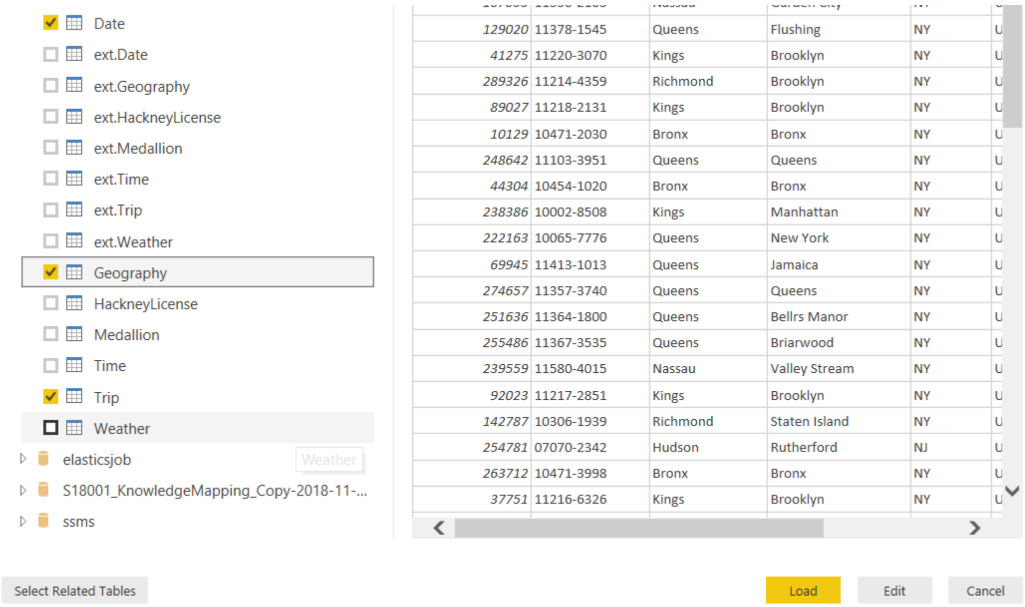
Data Connectivity mode ⓘ

○ Import

● DirectQuery

> Advanced options

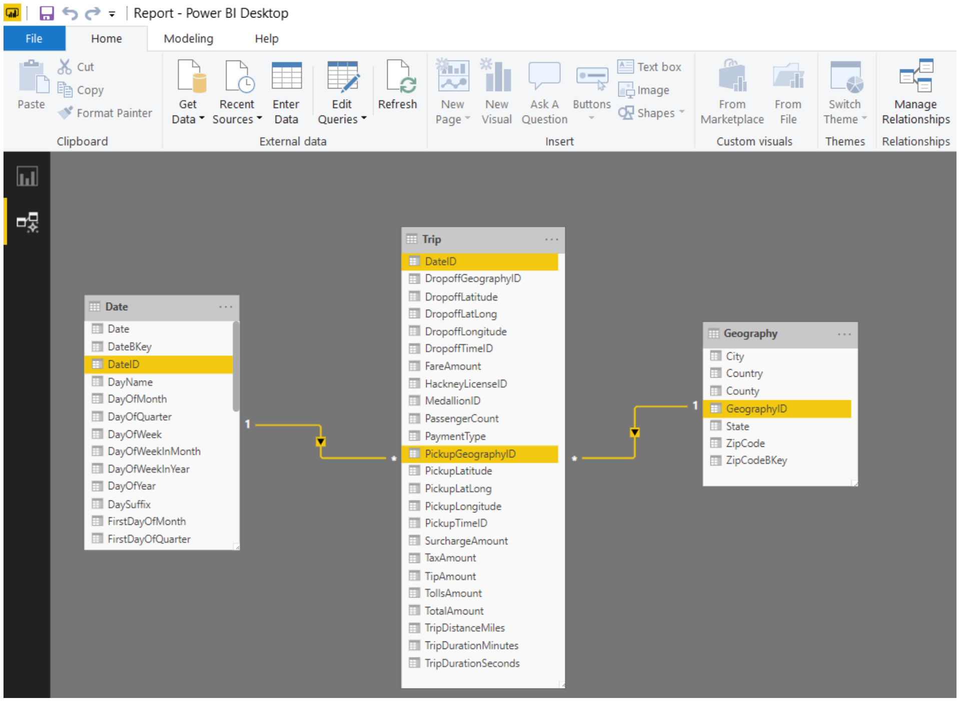
4. Select tables: Date, Geography and Trip



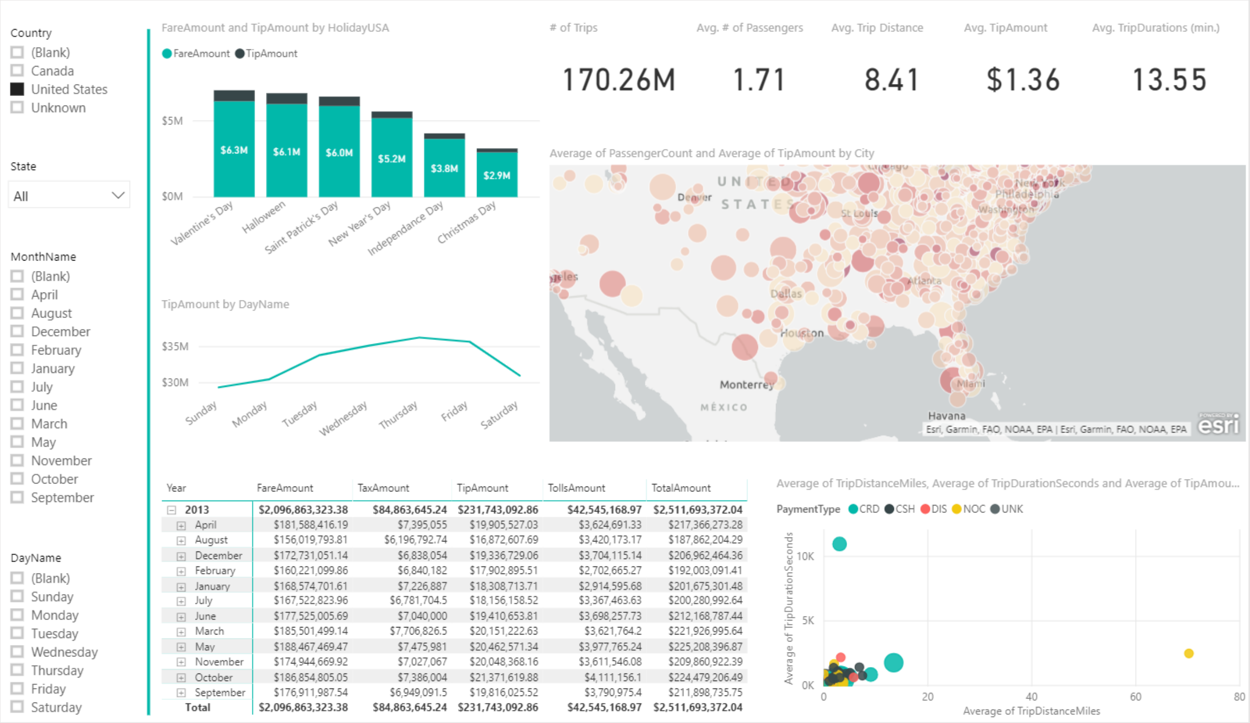
- ☑ Date
- □ ext.Date
- □ ext.Geography
- □ ext.HackneyLicense
- □ ext.Medallion
- □ ext.Time
- □ ext.Trip
- □ ext.Weather
- ☑ Geography
- □ HackneyLicense
- □ Medallion
- □ Time
- ☑ Trip
- □ Weather
- elasticsjob
- S18001_KnowledgeMapping_Copy-2018-11-...
- ssms

129020	11378-1545	Queens	Flushing	NY	U
41275	11220-3070	Kings	Brooklyn	NY	U
289326	11214-4359	Richmond	Brooklyn	NY	U
89027	11218-2131	Kings	Brooklyn	NY	U
10129	10471-2030	Bronx	Bronx	NY	U
248642	11103-3951	Queens	Queens	NY	U
44304	10454-1020	Bronx	Bronx	NY	U
238386	10002-8508	Kings	Manhattan	NY	U
222163	10065-7776	Queens	New York	NY	U
69945	11413-1013	Queens	Jamaica	NY	U
274657	11357-3740	Queens	Queens	NY	U
251636	11364-1800	Queens	Bellrs Manor	NY	U
255486	11367-3535	Queens	Briarwood	NY	U
239559	11580-4015	Nassau	Valley Stream	NY	U
92023	11217-2851	Kings	Brooklyn	NY	U
142787	10306-1939	Richmond	Staten Island	NY	U
254781	07070-2342	Hudson	Rutherford	NJ	U
263712	10471-3998	Bronx	Bronx	NY	U
37751	11216-6326	Kings	Brooklyn	NY	U

5. Create relationships: DateID = DateID & PickupGeographyID = GeographyID



6. Create your report, e.g.:



Country
□ (Blank)
□ Canada
■ United States
□ Unknown

State
All

MonthName
□ (Blank)
□ April
□ August
□ December
□ February
□ January
□ July
□ June
□ March
□ May
□ November
□ October
□ September

DayName
□ (Blank)
□ Sunday
□ Monday
□ Tuesday
□ Wednesday
□ Thursday
□ Friday
□ Saturday

# of Trips	Avg. # of Passengers	Avg. Trip Distance	Avg. TipAmount	Avg. TripDurations (min.)
170.26M	1.71	8.41	$1.36	13.55

Year	FareAmount	TaxAmount	TipAmount	TollsAmount	TotalAmount
⊟ **2013**	**$2,096,863,323.38**	**$84,863,645.24**	**$231,743,092.86**	**$42,545,168.97**	**$2,511,693,372.04**
⊞ April	$181,588,416.19	$7,395,055	$19,905,527.03	$3,624,691.33	$217,366,273.28
⊞ August	$156,019,793.81	$6,196,792.74	$16,872,607.69	$3,420,173.17	$187,862,204.29
⊞ December	$172,731,051.14	$6,838,054	$19,336,729.06	$3,704,115.14	$206,962,464.36
⊞ February	$160,221,099.86	$6,840,182	$17,902,895.51	$2,702,665.27	$192,003,091.41
⊞ January	$168,574,701.61	$7,226,887	$18,308,713.71	$2,914,595.68	$201,675,301.48
⊞ July	$167,522,823.96	$6,781,704.5	$18,156,158.52	$3,367,463.63	$200,280,992.64
⊞ June	$177,525,005.69	$7,040,000	$19,410,653.81	$3,698,257.73	$212,168,787.44
⊞ March	$185,501,499.14	$7,706,826.5	$20,151,222.63	$3,621,764.2	$221,926,995.64
⊞ May	$188,467,469.47	$7,475,981	$20,462,571.34	$3,977,765.24	$225,208,396.87
⊞ November	$174,944,669.92	$7,027,067	$20,048,368.16	$3,611,546.08	$209,860,922.39
⊞ October	$186,854,805.05	$7,386,004	$21,371,619.88	$4,111,156.1	$224,479,206.49
⊞ September	$176,911,987.54	$6,949,091.5	$19,816,025.52	$3,790,975.4	$211,898,735.75
Total	**$2,096,863,323.38**	**$84,863,645.24**	**$231,743,092.86**	**$42,545,168.97**	**$2,511,693,372.04**

Scale Up your SQL Data Warehouse

```
-- Check number of nodes (DWH database)
select * from sys.dm_pdw_nodes
```

Q: How many nodes do you have?

```
-- Check number of distribution per node (DWH database)
select count(*) distributions, pdw_node_id from sys.pdw_distributions group by
pdw_node_id
```

Q: How many distributions per node do you have?

```
-- Find your Data Warehouse SLO (Master database)
SELECT  db.name [Database], ds.edition [Edition], ds.service_objective [Service Objective]
FROM   sys.database_service_objectives ds
JOIN   sys.databases db ON ds.database_id = db.database_id
WHERE  db.name = 'XXX'

-- Scale UP to 2 nodes – 1000c Azure SQL DWH (Master database)
ALTER DATABASE XXX MODIFY (SERVICE_OBJECTIVE = 'DW1000c');
```

```
-- Monitor Scale process (Master database)
SELECT TOP 1 state_desc
FROM sys.dm_operation_status
WHERE resource_type_desc = 'Database'
AND major_resource_id = 'XXX'
AND operation = 'ALTER DATABASE'
ORDER BY start_time DESC
```

```
-- Check number of nodes (DWH database)
select * from sys.dm_pdw_nodes
```

Q: How many nodes do you have?

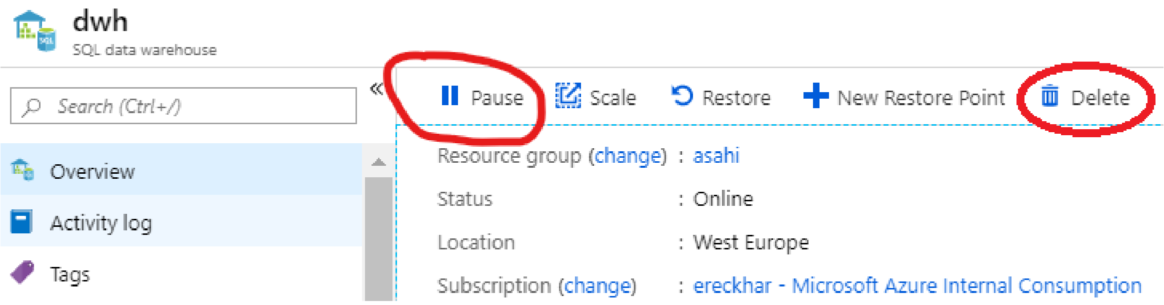
```
-- Check number of distribution per node (DWH database)
select count(*) distributions, pdw_node_id from sys.pdw_distributions group by
pdw_node_id
```

Q: How many distributions per node do you have?

NOTE: Try performance of your PBI report and above query.

Clean your environment

Pause or delete your Azure SQL Data Warehouse



Resource group (change) : asahi

Status : Online

Location : West Europe

Subscription (change) : ereckhar - Microsoft Azure Internal Consumption