Free Trial Getting Started Lab

For any questions or assistance, feel free to contact us at SQLDWFreeTrial@microsoft.com

Introduction

Hello and welcome to Azure SQL Data Warehouse. This tutorial will walk you through the beginning steps involved in getting your Azure SQL Data Warehouse up and running with some interesting data. Visual Studio or SSMS is needed to follow along in the data loading and querying sections.

Guide to Learning

This tutorial uses the styles listed below to guide you.

1.) Mandatory step to get through tutorial

Extra Information about topic will appear in this format

A footnote references more learning in online documentation[1]

! Important step or information

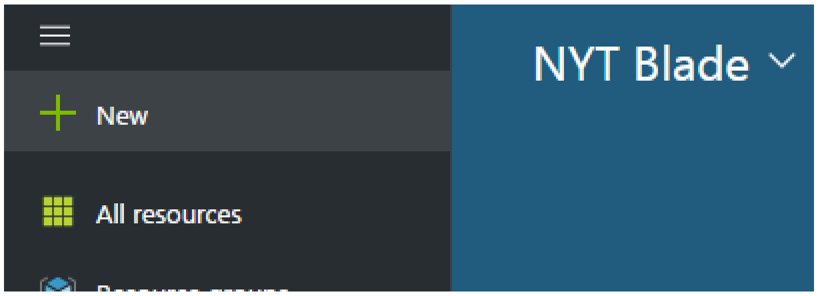
```
CODE SECTIONS
```

Contents

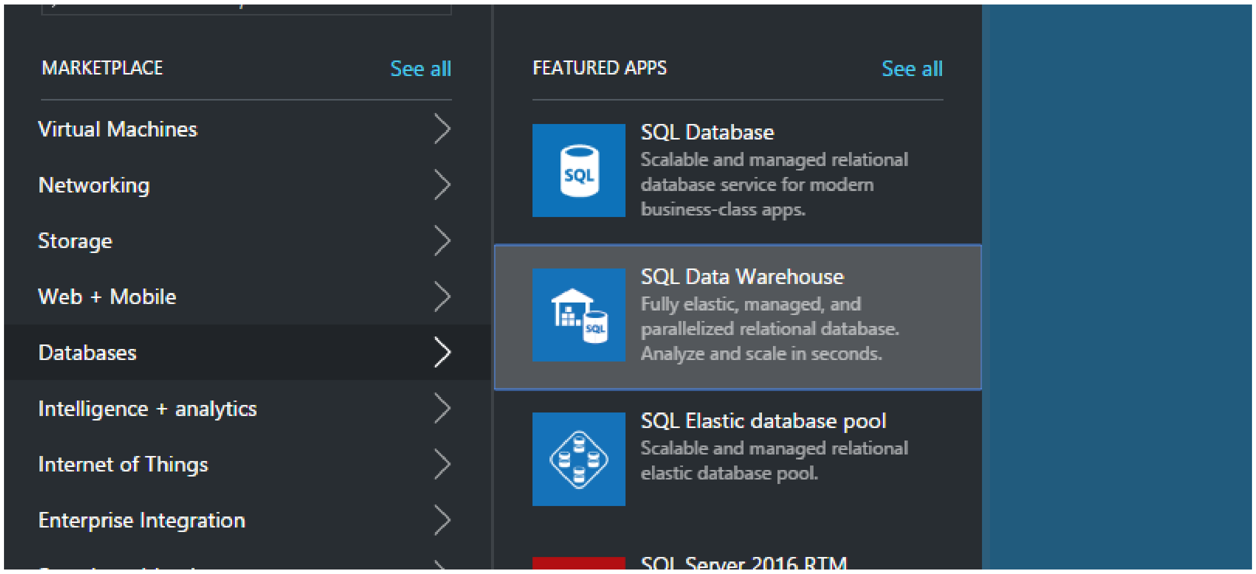
Deploying Azure SQL Data Warehouse

To begin with, let's deploy an Azure SQL Data Warehouse. Once you've logged into your Azure portal, you'll see your Azure services dashboard.

1.) To provision a DW instance, first click on add new (+ New) button from the resource navigation pane. This should bring up the marketplace navigation pane.



2.) Select Databases and then SQL Data Warehouse


MARKETPLACE
See all
Virtual Machines
Networking
Storage
Web + Mobile
Databases
Intelligence + analytics
Internet of Things
Enterprise Integration
FEATURED APPS
See all
SQL Database
Scalable and managed relational database service for modern business-class apps.
SQL Data Warehouse
Fully elastic, managed, and parallelized relational database. Analyze and scale in seconds.
SQL Elastic database pool
Scalable and managed relational elastic database pool.

3.) Fill out deployment details

Database Name: Pick anything you'd like! It is recommended your database includes information about your instance such as name, region, and type if you have multiple such as *mydw-westus1-test*

Subscription: Your Azure subscription

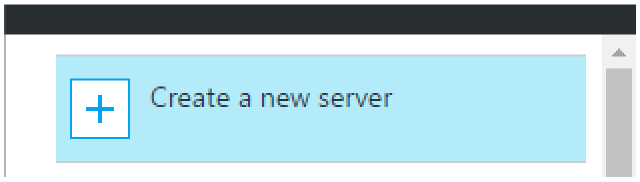
Resource Group: Create new (or Use existing if you have plan on using your Azure SQL Data Warehouse with other services).

Services within a resource group should have the same lifecycle. They are particularly useful for resource administration such as scoping access control as well as templated deployment.

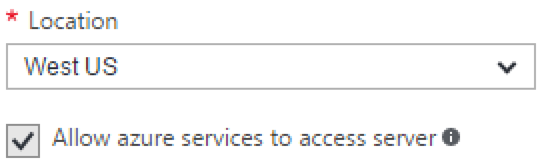
Source: Blank Database

Server: If you have a pre-existing server that you're using for this tutorial, select that. If not, you can always create a new one by clicking on server and filling out some administrative details.

! Free trial members should select the server they created for the free trial

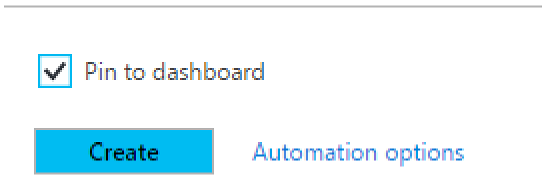


! Be sure to select ***"Allow azure services to access server".***


Location: West US

Allow azure services to access server ✓

4.) Leave the default collation SQL_Latin1_General_CP1_CI_AS
5.) Select performance (We'd recommend 200 DWU for this lab)
6.) Select ***Pin to Dashboard*** and then ***Create***

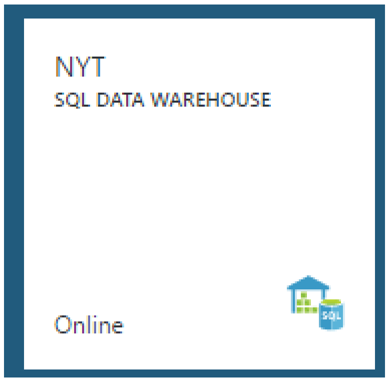


7.) Sit back and wait for your Azure SQL Data Warehouse to deploy! It's normal for this process to take several minutes. The portal will notify you when it your instance is done deploying.

Configuring Firewall

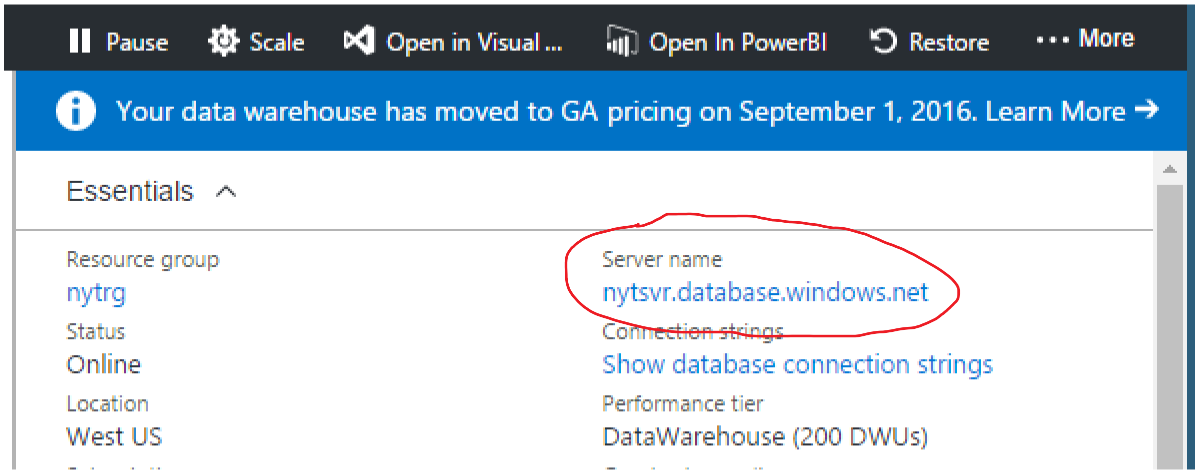
Before you begin to use your Azure SQL Data Warehouse, you must set up firewall rules to access your server.

1.) Select your DW instance from your dashboard. (Your resource pane may already be open post deployment)



NYT
SQL DATA WAREHOUSE
Online

2.) Select the server that your Data Warehouse instance is on from the Essentials Tile area



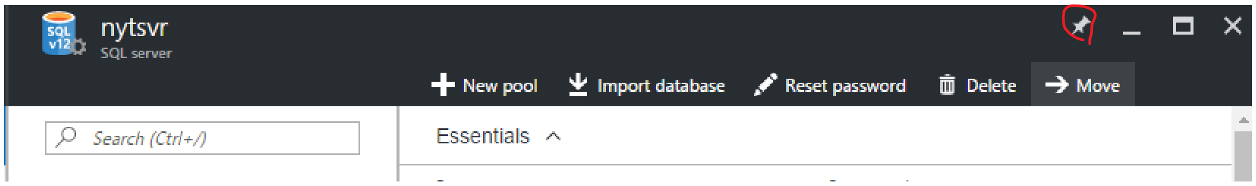
Your data warehouse has moved to GA pricing on September 1, 2016. Learn More →

Essentials

Resource group
nytrg
Status
Online
Location
West US

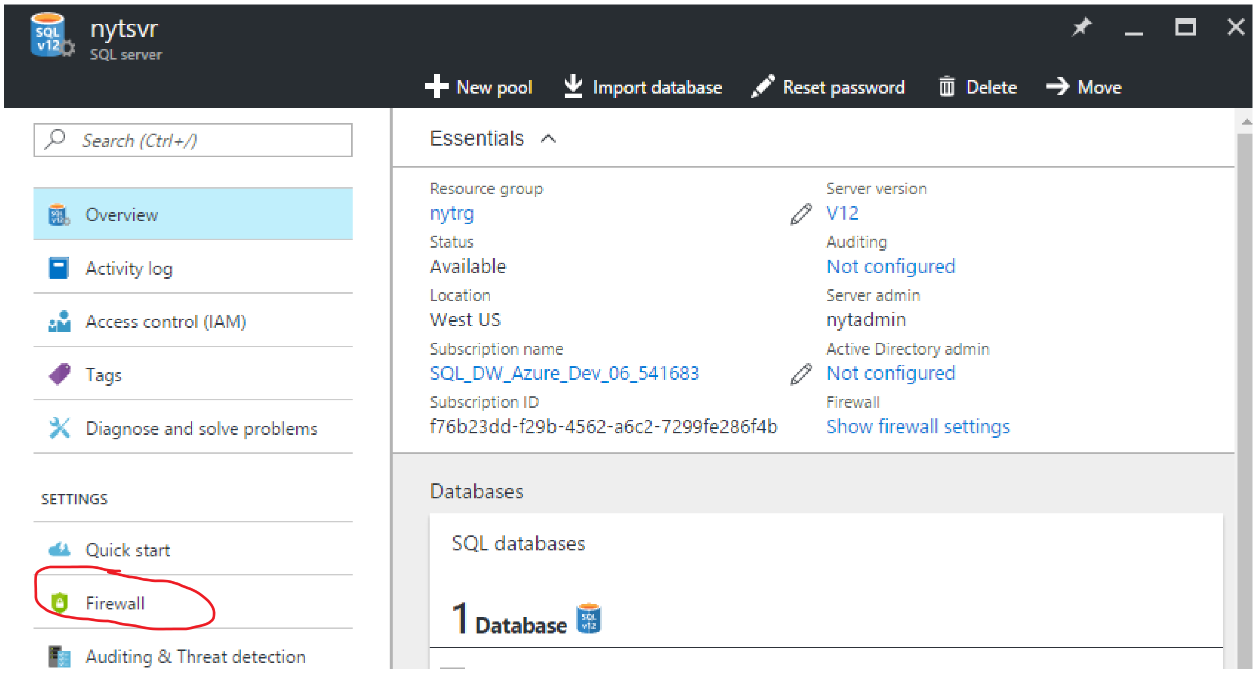
Server name
nytsvr.database.windows.net
Connection strings
Show database connection strings
Performance tier
DataWarehouse (200 DWUs)

3.) If your server isn't already pinned to the dashboard, go ahead and do that by selecting the pin icon in the top right of the server pane.



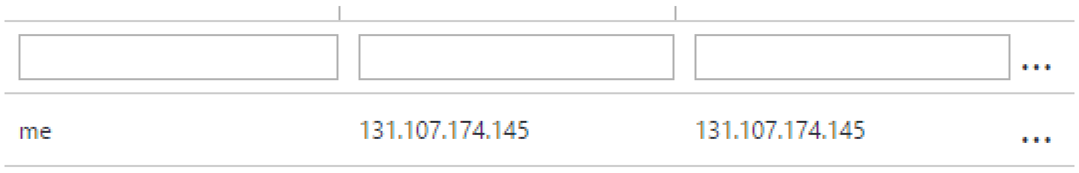
nytsvr
SQL server

4.) Select Firewall rules from the settings on the left

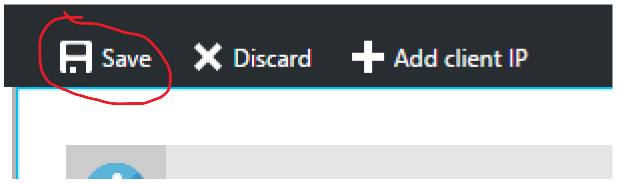


Resource group: nytrg
Status: Available
Location: West US
Subscription name: SQL_DW_Azure_Dev_06_541683
Subscription ID: f76b23dd-f29b-4562-a6c2-7299fe286f4b
Server version: V12
Auditing: Not configured
Server admin: nytadmin
Active Directory admin: Not configured
Firewall: Show firewall settings

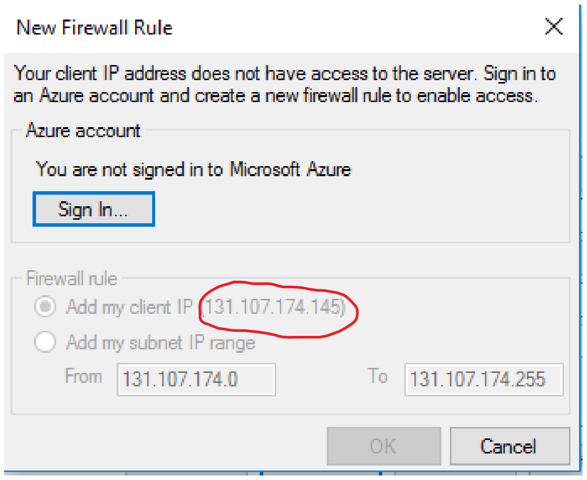
5.) Add an IP by giving it a rule name, start IP, and end IP. Be sure to click **save** after you're done.



me	131.107.174.145	131.107.174.145



! If you are in a network, the client IP the portal suggests may not be correct. If you have trouble connecting with VSTS or SSMS, add the IP listed in the error pane.



New Firewall Rule

Your client IP address does not have access to the server. Sign in to an Azure account and create a new firewall rule to enable access.

Azure account

You are not signed in to Microsoft Azure

Sign In...

Firewall rule

Add my client IP (131.107.174.145)

Add my subnet IP range

From 131.107.174.0 To 131.107.174.255

OK Cancel

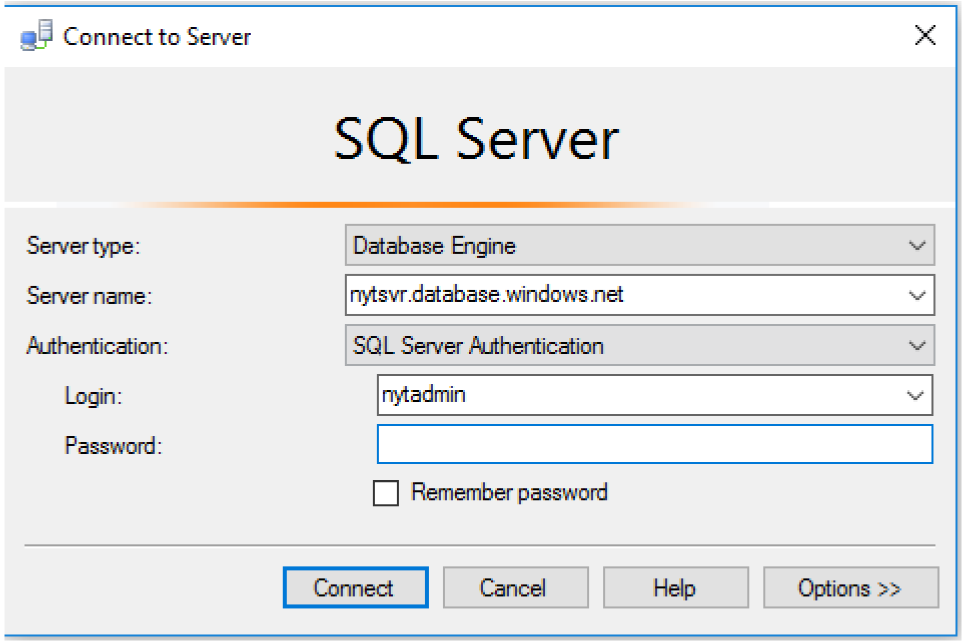
Creating a New User

This demo will demonstrate how to connect to your Azure SQL Data Warehouse using SQL Server management Studio. If you prefer, you can also access your DW Instance from PowerShell or Visual Studio with SQL Server Developer Tools installed. [1]

Before we load data, you'll want to create a new user for your DW. By default, the admin class that comes with your server is given a **smaller resource class**.

Azure SQL Data Warehouse uses resource classes to allocate memory to queries. Certain queries, like large joins or loads to clustered column store tables, will benefit from larger memory allocations. Some queries, like pure scans, will see no benefit. On the flip side, utilizing larger resource classes impacts concurrency, so you will want to take this into consideration before moving all your users to a large resource class.

1.) Connect to your server that your Data Warehouse is on.



Connect to Server

SQL Server

Server type: Database Engine

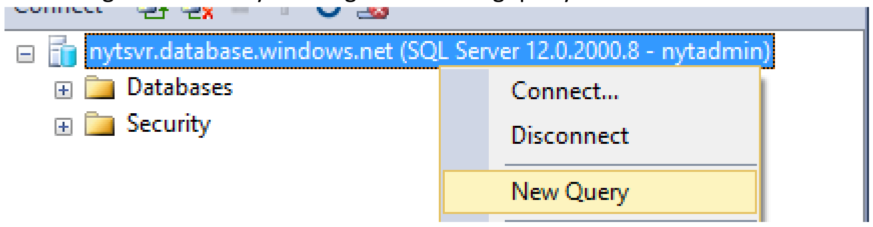
Server name: nytsvr.database.windows.net

Authentication: SQL Server Authentication

Login: nytadmin

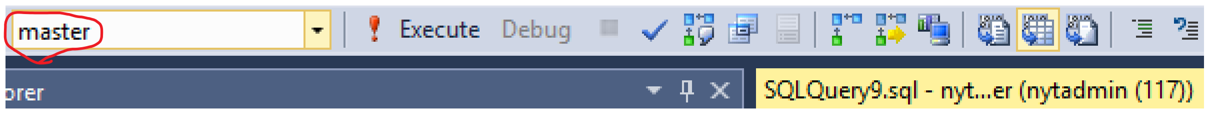
Password:

2.) Create login and user by running the following query on the **master** database of your server



nytsvr.database.windows.net (SQL Server 12.0.2000.8 - nytadmin)

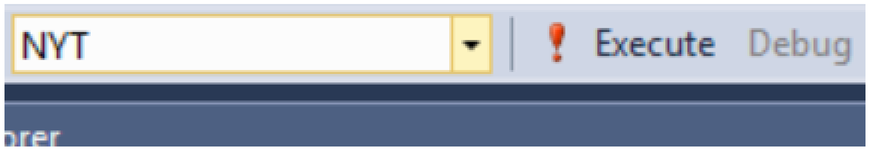
[1] https://azure.microsoft.com/en-us/documentation/articles/sql-data-warehouse-query-visual-studio/



```
-- Connect to master database and create a login
CREATE LOGIN XLRCLogin WITH PASSWORD = 'a123reallySTRONGpassword!';
CREATE USER XLRCUser FOR LOGIN XLRCLogin;
```

! From this point forward, all queries are run on the Data Warehouse database instance itself and **not on the master database**.

3.) Add a user for your Data Warehouse instance with the new login by running this query on your **Data Warehouse**



```
-- Connect to SQL DW database and create a database user
CREATE USER XLRCUser FOR LOGIN XLRCLogin;
```

4.) Give your user DB control

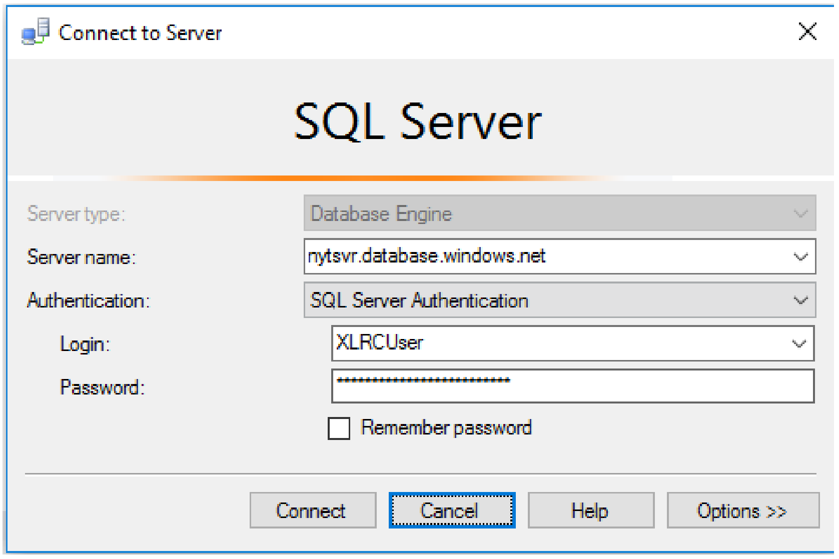
```
GRANT CONTROL ON DATABASE::[NYT] to XLRCUser;
```

! If your DB has hyphens in it, be sure to wrap it in brackets!

5.) Add your new user XLRCUser to a larger resource class role

```
-- Adds user XLRCUser to the xlargerc resource class role
EXEC sp_addrolemember 'xlargerc', 'XLRCUser';
```

6.) Login with your new user



Server type: Database Engine
Server name: nytsvr.database.windows.net
Authentication: SQL Server Authentication
Login: XLRCUser
Password: **************************
Remember password

Loading data

For this lab, we're going to use a smaller dataset that only includes the year 2013 but you can also access a decade's worth of data.

Defining External Data

1.) Define an external data source

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

! In the future when you use your own private Azure Storage Blobs, you must implement a master key and credentials. Read more about this in the documentation.[2]

2.) Define the external file formats. The `CREATE EXTERNAL FILE FORMAT` command is used to specify the format of the external data that you'll be loading from. For the New York public taxi data, we've used the two formats for storing the data in Azure Blob Storage

```
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'
);
```

3.) Create a schema for your external file format

```
CREATE SCHEMA ext;
GO
```

4.) Create the external tables. These are the tables reference data stored in Hadoop or Azure blob storage.

```
CREATE EXTERNAL TABLE [ext].[Date]
```

[2] https://azure.microsoft.com/en-us/documentation/articles/sql-data-warehouse-load-from-azure-blob-storage-with-polybase/#1-configure-the-data-source

```sql
(
    [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 = 'Weather2013'
,   DATA_SOURCE = NYTPublic
,   FILE_FORMAT = uncompressedcsv
,   REJECT_TYPE = value
,   REJECT_VALUE = 0
)
;
```

CTAS (Create Table As Select)

As a SQL DW user, you will be using the CTAS statement often. Create table as select or CTAS is one of the most important T-SQL features available. It is a fully parallelized operation that creates a new table based on the output of a SELECT statement. CTAS is the simplest and fastest way to create a copy of a table.[3]

5.) Load your data from external tables into your Data Warehouse instance!

```
CREATE TABLE [dbo].[Date]
WITH
(   DISTRIBUTION = ROUND_ROBIN
,   CLUSTERED COLUMNSTORE INDEX
```

[3] https://azure.microsoft.com/en-us/documentation/articles/sql-data-warehouse-develop-ctas/

```
)
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]')
;
```

! You're loading several GBs of data and compressing it into highly performant Cluster Columnstore Indexes. Run the DMV query below and then grab a coffee or snack while Azure SQL Data Warehouse does some heavy lifting. ☺

Dynamic Management View

6.) Create a **new query window** and watch as your data come in with this Dynamic Management View (DMV)[4].

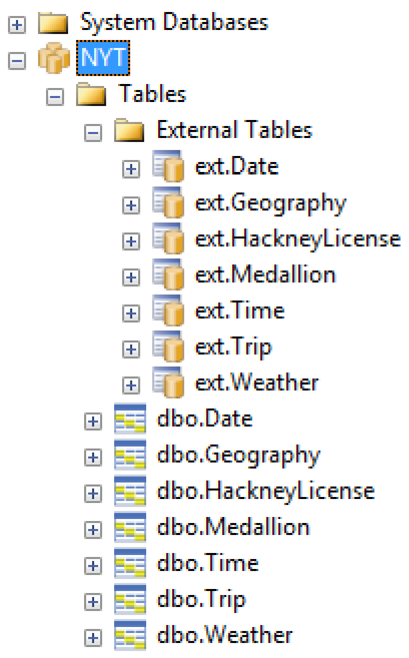
```
SELECT
    r.command,
    s.request_id,
    r.status,
    count(distinct input_name) as nbr_files,
    sum(s.bytes_processed)/1024/1024 as gb_processed
FROM
    sys.dm_pdw_exec_requests r
    inner join sys.dm_pdw_dms_external_work s
        on r.request_id = s.request_id
WHERE
    r.[label] = 'CTAS : 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
    nbr_files desc, gb_processed desc;
```

7.) View all queries

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

8.) Enjoy seeing your data nicely loaded into your Azure SQL Data Warehouse.

[4] https://azure.microsoft.com/en-us/documentation/articles/sql-data-warehouse-manage-monitor/

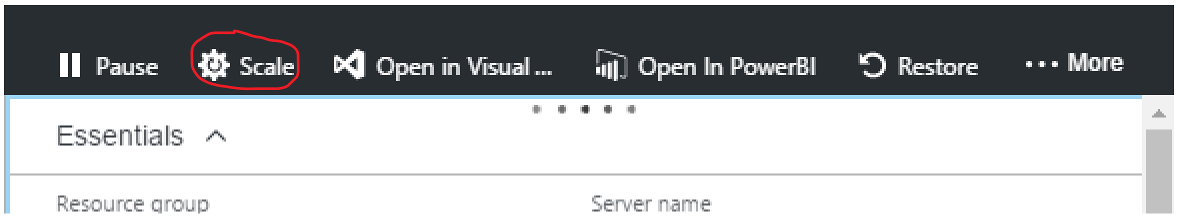

System Databases
NYT
Tables
External Tables
ext.Date
ext.Geography
ext.HackneyLicense
ext.Medallion
ext.Time
ext.Trip
ext.Weather
dbo.Date
dbo.Geography
dbo.HackneyLicense
dbo.Medallion
dbo.Time
dbo.Trip
dbo.Weather

Scan Query with Scaling

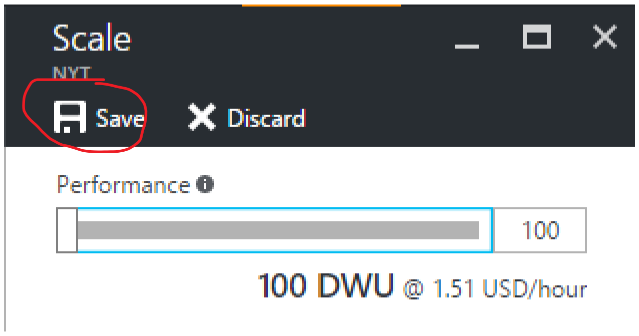
Let's now get a good idea of how scaling affects the speed of your queries.

Before we begin. Let's scale our operation down to 100 DWUs so we can get an idea of how one compute node might perform on its own.

1.) Go to your portal and select your DW instance
2.) Click scale



3.) Scale your Data Warehouse down to 100 DWU and hit save



100 DWU @ 1.51 USD/hour

4.) Wait for your scale operation to finish. Please note, scaling operations will kill your currently running queries and prevent new ones from being run during the scaling operation.
5.) First let's do a scan operation on the trip data selecting the top million entries for all the columns. If you're eager to move on quickly, feel free to select fewer rows.

```sql
SELECT TOP(1000000) * FROM dbo.[Trip]
```

Take note of the time it took to run this initial operation.

____

6.) Scale up your instance to a new DWU. Remember each 100 DWU is adding another compute node to your Azure SQL Data Warehouse and run the query again. You should notice a significant difference.

Azure SQL Data Warehouse is a Massively Parallel Processing (MPP) platform. Queries and operations that can parallelize work among various nodes will experience the true power of Azure SQL Data Warehouse.

Join Query with Statistics

1.) Run a query that joins the Date table with the Trip table

```
SELECT TOP (1000000) dt.[DayOfWeek]
      ,tr.[MedallionID]
      ,tr.[HackneyLicenseID]
      ,tr.[PickupTimeID]
      ,tr.[DropoffTimeID]
      ,tr.[PickupGeographyID]
      ,tr.[DropoffGeographyID]
      ,tr.[PickupLatitude]
      ,tr.[PickupLongitude]
      ,tr.[PickupLatLong]
      ,tr.[DropoffLatitude]
      ,tr.[DropoffLongitude]
      ,tr.[DropoffLatLong]
      ,tr.[PassengerCount]
      ,tr.[TripDurationSeconds]
      ,tr.[TripDistanceMiles]
      ,tr.[PaymentType]
      ,tr.[FareAmount]
      ,tr.[SurchargeAmount]
      ,tr.[TaxAmount]
      ,tr.[TipAmount]
      ,tr.[TollsAmount]
      ,tr.[TotalAmount]
  FROM [dbo].[Trip] as tr
  join
  dbo.[Date] as dt
  on tr.DateID = dt.DateID
```

As you might expect, the query takes much longer when you shuffle data among the nodes, especially in a join scenario like this.

2.) Let's see how this differs when we create statistics on the column we're joining by running the following:

```
CREATE STATISTICS [dbo.Date DateID stats] ON dbo.Date (DateID);
CREATE STATISTICS [dbo.Trip DateID stats] ON dbo.Trip (DateID);
```

SQL DW does not automatically manage statistics for you. Statistics are important for query performance and it is highly recommended you create and update statistics.

You will gain the most benefit by having statistics on columns involved in joins, columns used in the WHERE clause and columns found in GROUP BY.

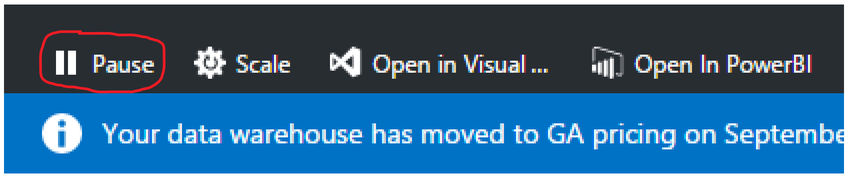
3.) Run the query from Step 1 again and observe any performance differences. While the differences in query performance will not be as drastic as scaling up, you should notice a discernable speed up.

Next Steps

You’re now ready to query and explore. Everything after this is best practices or tips. Enjoy ☺

- Ask a question on our forums[5] or StackOverflow[6]
- Watch videos on Azure SQL Data Warehouse at Channel9[7]
- Check out of the rest of the documentation on Azure[8] or MSDN[9]
- Check out customer case studies[10] to explore how SQL Data Warehouse might fit your needs

If you’re done exploring for the day, make sure to pause your instance! In production, you can experience enormous savings by pausing and scaling to meet your business needs.



Your data warehouse has moved to GA pricing on Septembe

Useful Readings

- Concurrency and workload management in Azure SQL Data Warehouse
- Best Practices for Azure SQL Data Warehouse
- Query Monitoring
- Top 10 Best Practices for Building a Large Scale Relational Data Warehouse
- Migrating Data to Azure SQL Data Warehouse

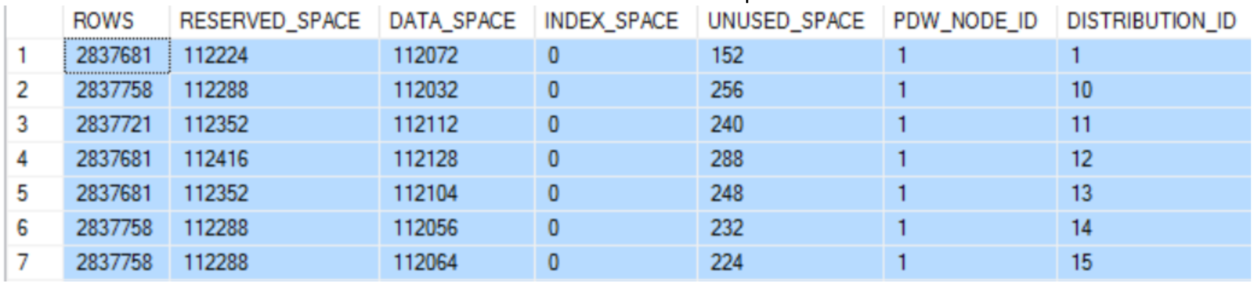
[5] https://social.msdn.microsoft.com/Forums/azure/en-US/home?forum=AzureSQLDataWarehouse
[6] https://stackoverflow.com/questions/tagged/azure-sqldw
[7] https://channel9.msdn.com/Tags/sql-data-warehouse
[8] https://azure.microsoft.com/en-us/documentation/services/sql-data-warehouse/
[9] https://msdn.microsoft.com/en-us/library/bb510741.aspx
[10] https://customers.microsoft.com/en-US/search?sq=%22Azure%20SQL%20Data%20Warehouse%22&ff=story_product_categories%26%3EAzure&p=0

Distribution of Data

After loading your data, you may want to see how the row distributions for each table. One way of doing this is by executing the following command to see the number of rows for each distribution for the Trip table.

```
DBCC PDW_SHOWSPACEUSED ("dbo.Trip")
```

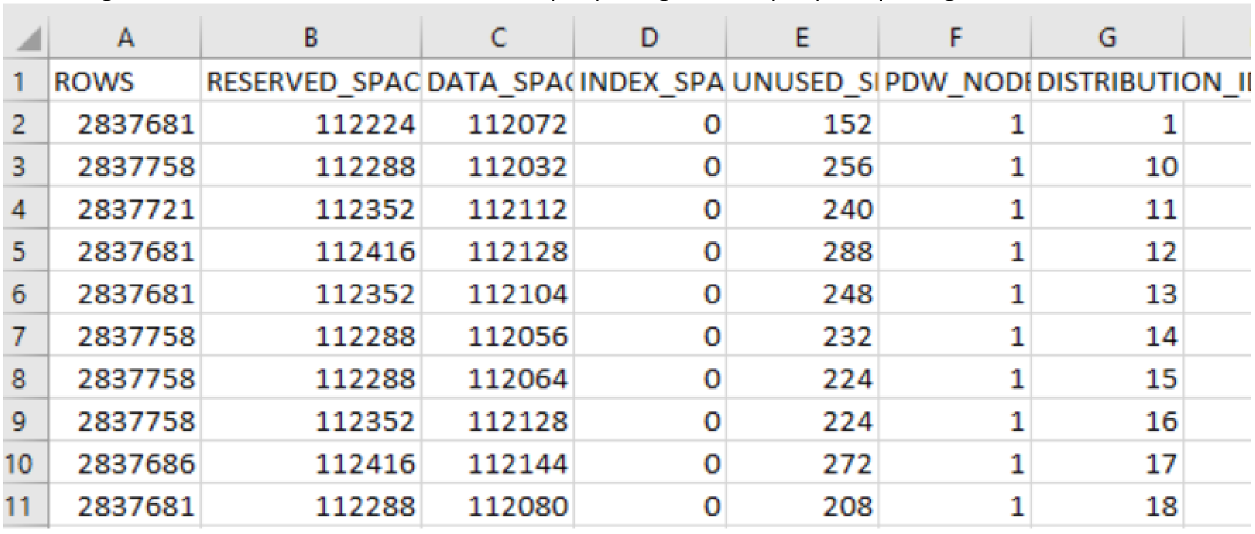
which will return the number of rows on each distribution for the Trip table like here:



	ROWS	RESERVED_SPACE	DATA_SPACE	INDEX_SPACE	UNUSED_SPACE	PDW_NODE_ID	DISTRIBUTION_ID
1	2837681	112224	112072	0	152	1	1
2	2837758	112288	112032	0	256	1	10
3	2837721	112352	112112	0	240	1	11
4	2837681	112416	112128	0	288	1	12
5	2837681	112352	112104	0	248	1	13
6	2837758	112288	112056	0	232	1	14
7	2837758	112288	112064	0	224	1	15

In this tutorial, we have used ROUND_ROBIN distribution which, as the name states, distributes rows one by one across the distributions.

With large loads, you'll want to use hash distribution get to get a more even distribution.[11] One nice way of visualizing the data is to take the results from the query alongside the query and putting it into an excel sheet.



	A	B	C	D	E	F	G
1	ROWS	RESERVED_SPAC	DATA_SPAC	INDEX_SPA	UNUSED_SI	PDW_NODI	DISTRIBUTION_II
2	2837681	112224	112072	0	152	1	1
3	2837758	112288	112032	0	256	1	10
4	2837721	112352	112112	0	240	1	11
5	2837681	112416	112128	0	288	1	12
6	2837681	112352	112104	0	248	1	13
7	2837758	112288	112056	0	232	1	14
8	2837758	112288	112064	0	224	1	15
9	2837758	112352	112128	0	224	1	16
10	2837686	112416	112144	0	272	1	17
11	2837681	112288	112080	0	208	1	18

Select all of the data, go to the insert tab, and then create a bar chart with the DISTRIBUTION_ID column for your horizontal axis and your ROWS column as your data.

[11] https://azure.microsoft.com/en-us/documentation/articles/sql-data-warehouse-best-practices/#hash-distribute-large-tables

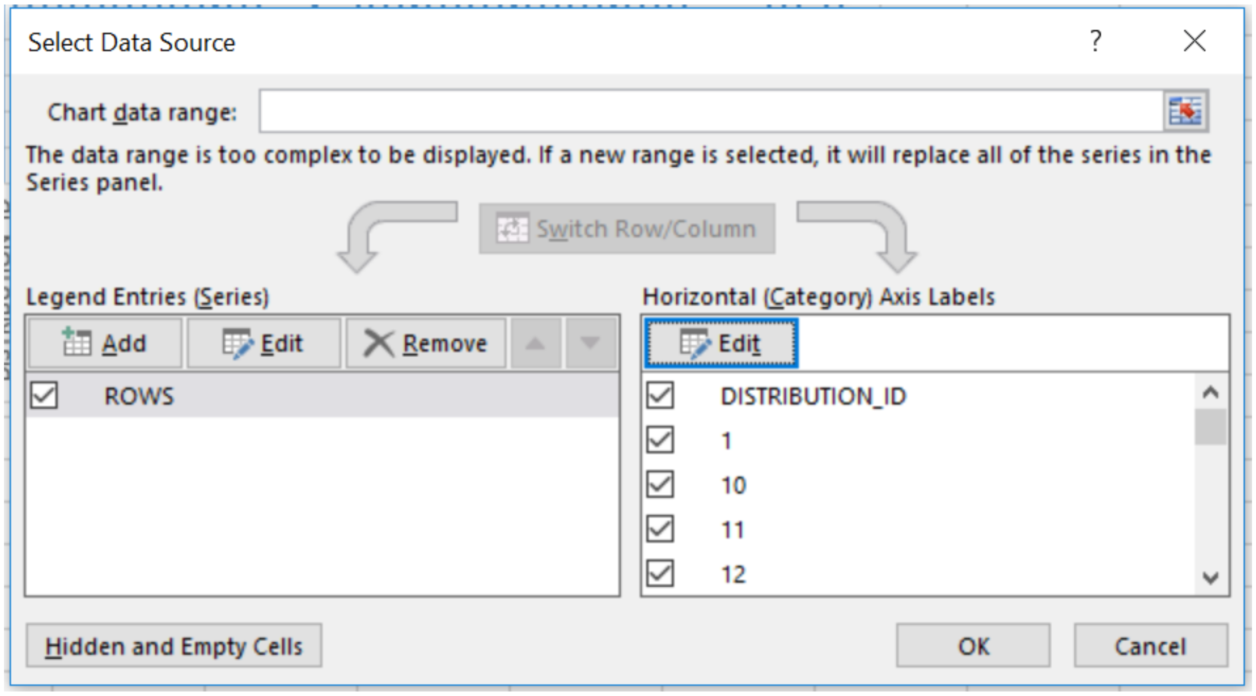

Select Data Source

Chart data range:

The data range is too complex to be displayed. If a new range is selected, it will replace all of the series in the Series panel.

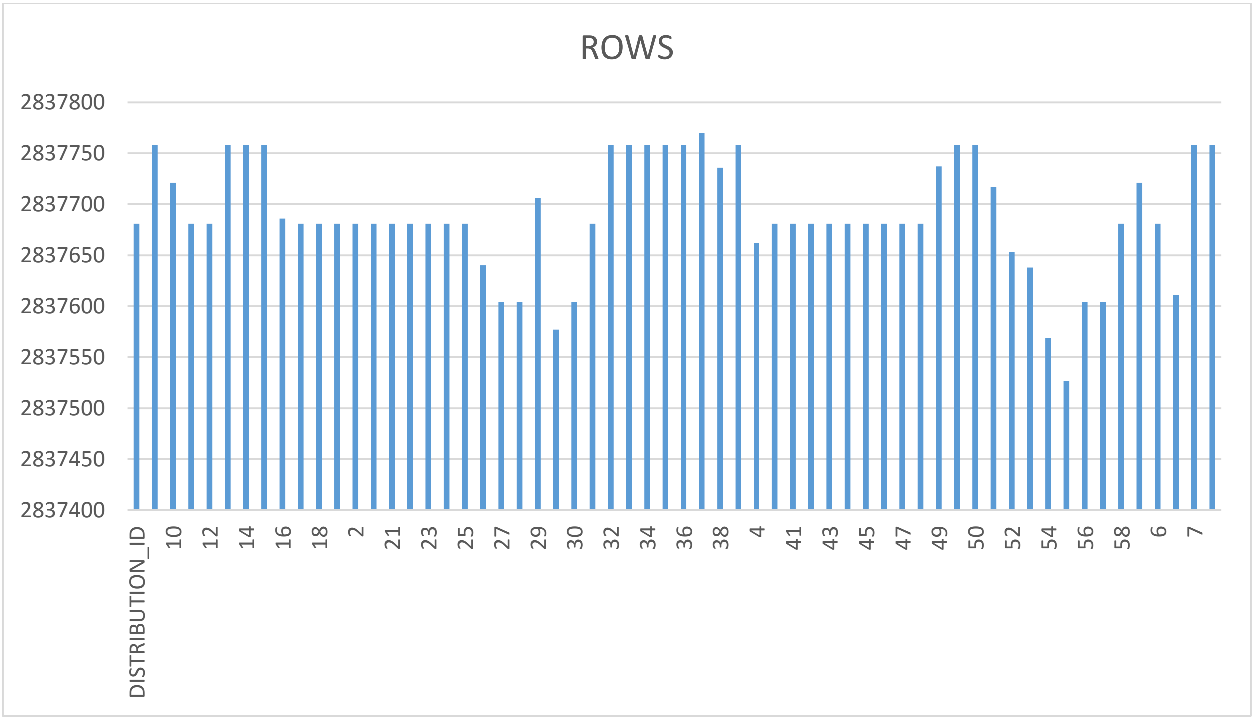
Legend Entries (Series)

☑ ROWS

Horizontal (Category) Axis Labels

☑ DISTRIBUTION_ID
☑ 1
☑ 10
☑ 11
☑ 12

You should then be able to see the data in a nice bar chart



As you can see, the distribution for the dbo.Trip table is not bad.

Generally for hash distributions, you'll want to select a single column which will:

1. Not be updated
2. Distribute data evenly, avoiding data skew
3. Minimize data movement

Earlier we used a CTAS statement to create tables in our Azure SQL Data Warehouse from external files. Try creating another Trip table but with a hash distribution.

```
CREATE TABLE dbo.TripHashed
WITH
(
DISTRIBUTION = Hash(DateID),
CLUSTERED COLUMNSTORE INDEX
)
AS SELECT * FROM dbo.Trip;
```

Visualize the distribution with the previous technique and then compare its performance to the previous join query. How does it differ? Sometimes ROUND_ROBIN distributions are not a bad choice. Read more about distributions best practices in the documentation.[12]

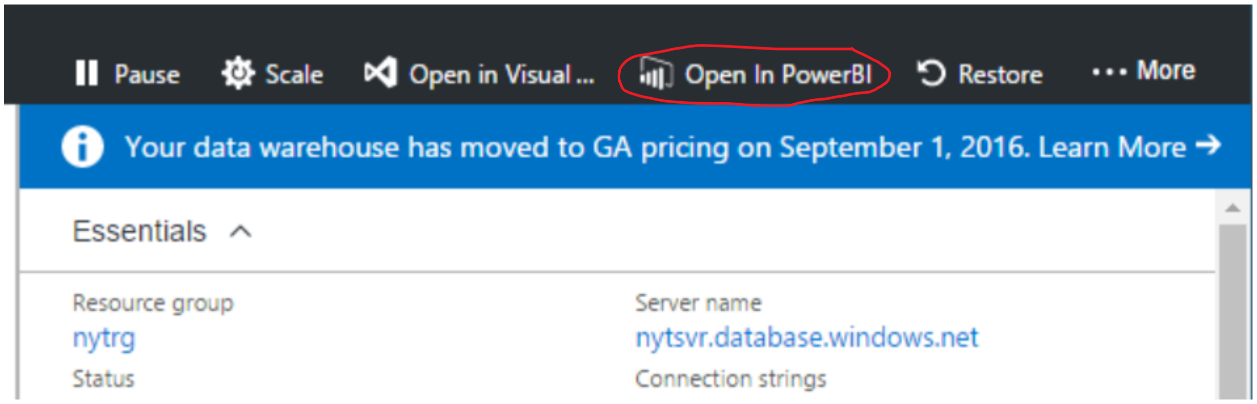
[12] https://azure.microsoft.com/en-us/documentation/articles/sql-data-warehouse-tables-distribute

PowerBI (Online)

As with Azure SQL Database, Azure SQL Data Warehouse Direct Connect allows user to leverage powerful logical pushdown alongside the analytical capabilities of Power BI. With Direct Connect, queries are sent back to your Azure SQL Data Warehouse in real time as you explore the data.

! For production use, it is not advised to setup Azure SQL Data Warehouse and PowerBI with direct querying for reporting or day-to-day analysis. For best results, SQL DW should be used in combination with other analytics services such as Azure Analysis Services or SQL Server Analysis Services which can cache recent data, leaving more room for powerful select ad-hoc queries on the Azure SQL Data Warehouse instance directly.

1.) Open your DW instance in the portal and click on "Open in PowerBI"

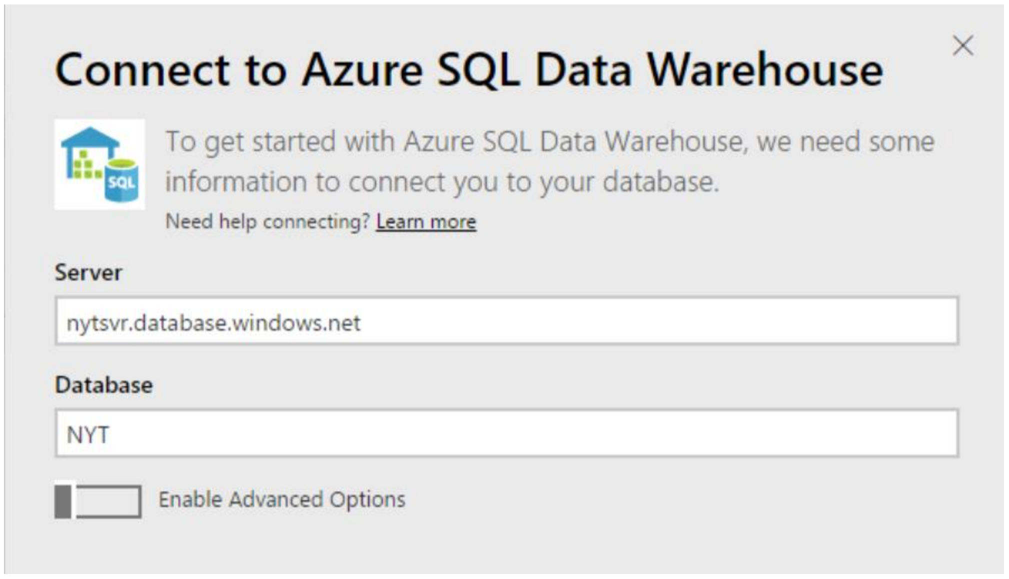


Your data warehouse has moved to GA pricing on September 1, 2016. Learn More →

Essentials

Resource group: nytrg
Server name: nytsvr.database.windows.net
Status
Connection strings

2.) Make sure your server name is correct on the Connect to Azure SQL Data Warehouse page and hit next.



Connect to Azure SQL Data Warehouse

To get started with Azure SQL Data Warehouse, we need some information to connect you to your database.

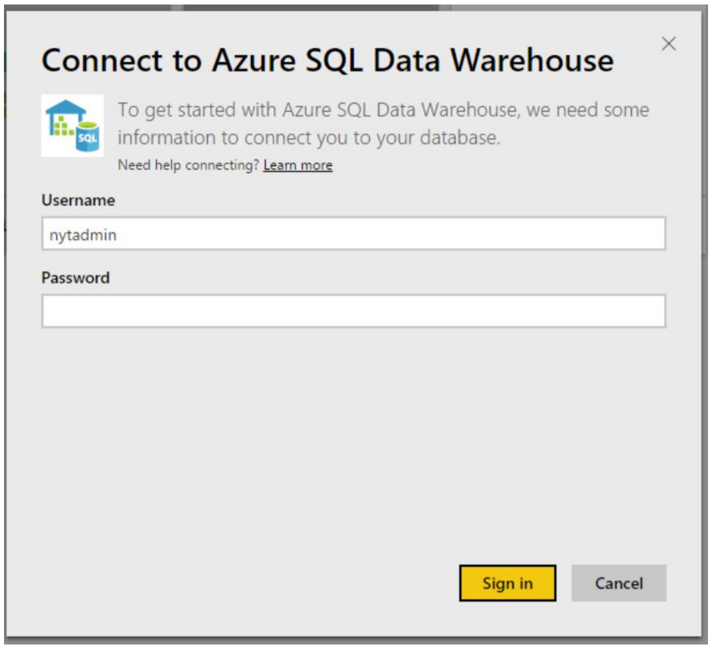
Need help connecting? Learn more

Server: nytsvr.database.windows.net

Database: NYT

Enable Advanced Options

3.) Provide the correct login credentials



Connect to Azure SQL Data Warehouse

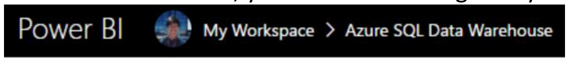
To get started with Azure SQL Data Warehouse, we need some information to connect you to your database.

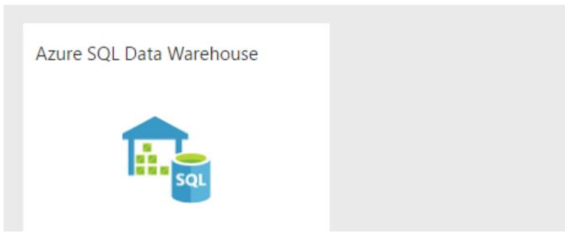
Need help connecting? Learn more

Username: nytadmin

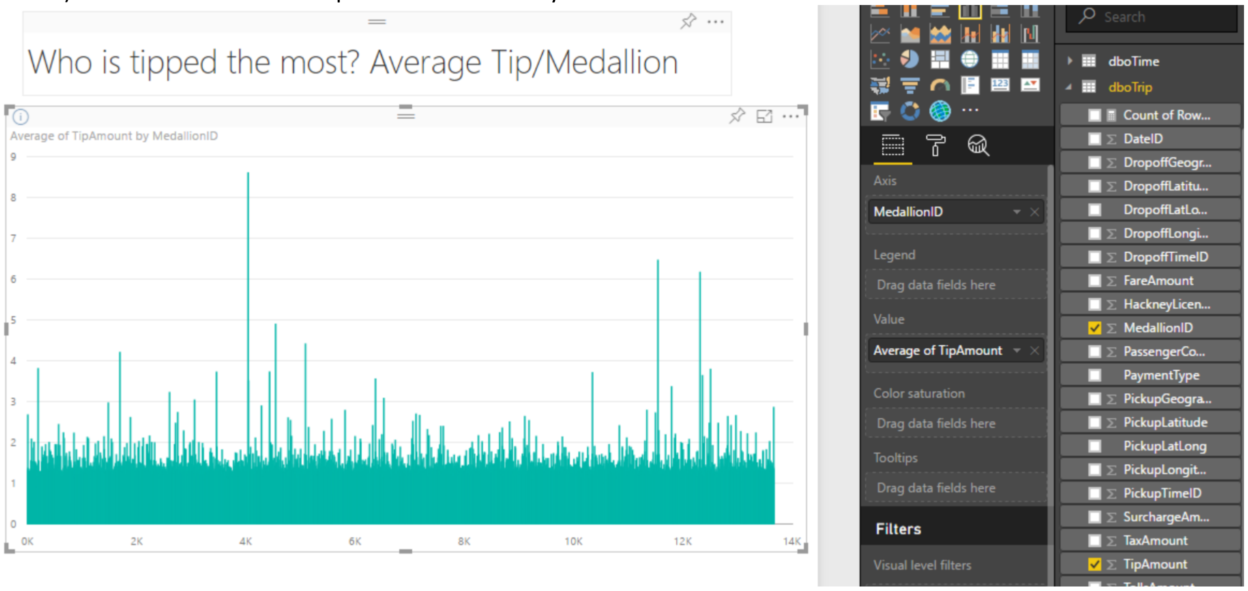
Password:

4.) After a few moments, you should be brought to your dashboard with a new Azure SQL Data Warehouse tile.





5.) Click on the tile and explore PowerBI with your data!



Who is tipped the most? Average Tip/Medallion

Axis: MedallionID

Value: Average of TipAmount