**Load New York Taxicab data to Azure SQL Data Warehouse**

PolyBase is a technology that accesses data outside of the database via the T-SQL language. It is the best way to load data into SQL Data Warehouse. With PolyBase, the data loads in parallel from the data source directly to the compute nodes. This tutorial uses PolyBase to load New York Taxicab data from a public Azure blob to Azure SQL Data Warehouse. The tutorial uses the Azure portal and SQL Server Management Studio.

**Creating Azure SQL Data Warehouse:**

**Step 1:** Sign into your Azure Portal using your credentials and click on **Create a resource** in the upper left-hand corner of the Azure portal and select **SQL Data Warehouse** from the **Databases** category. Fill the necessary details as below to create your SQL Data Warehouse

**Database name**: Any name as an identifier

**Subscription:** Select your subscription

**Resource group**: Create a new resource group with a name

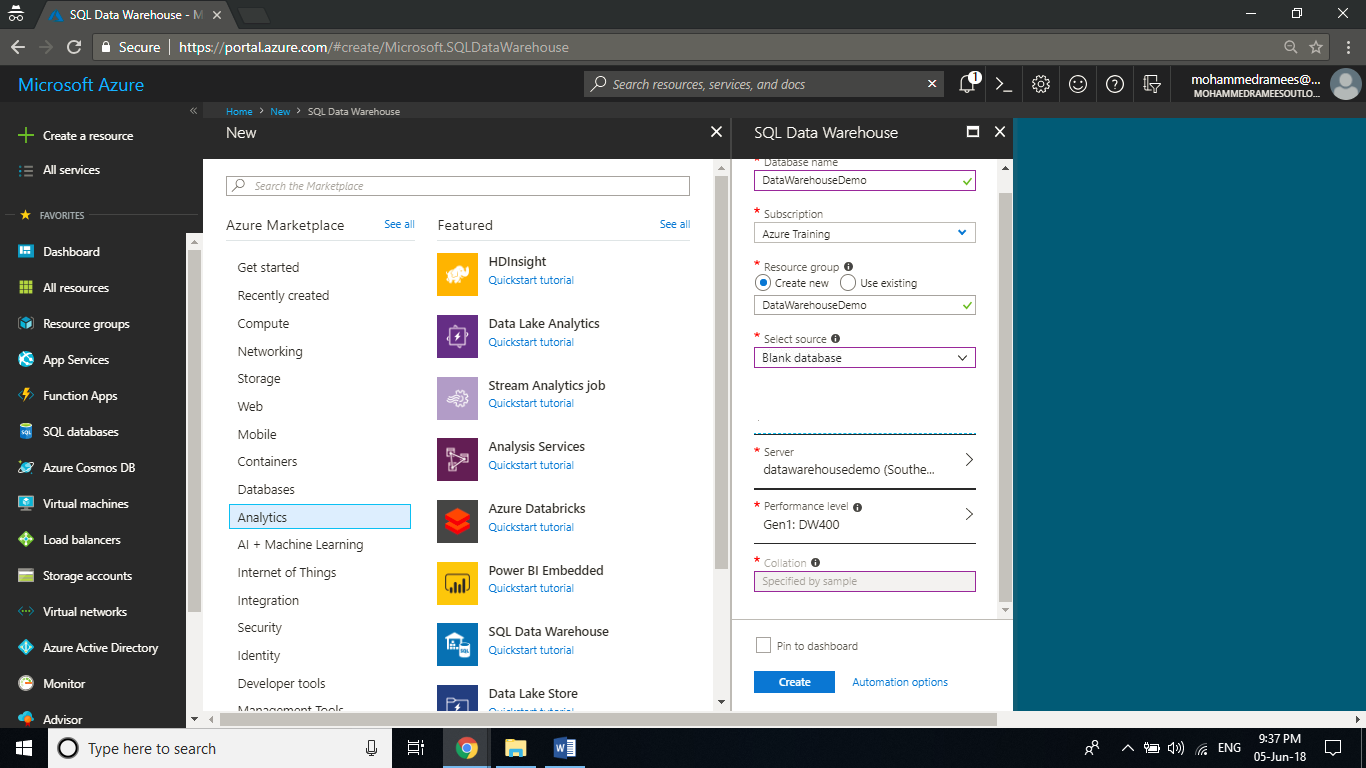
**Select source**: Select sample

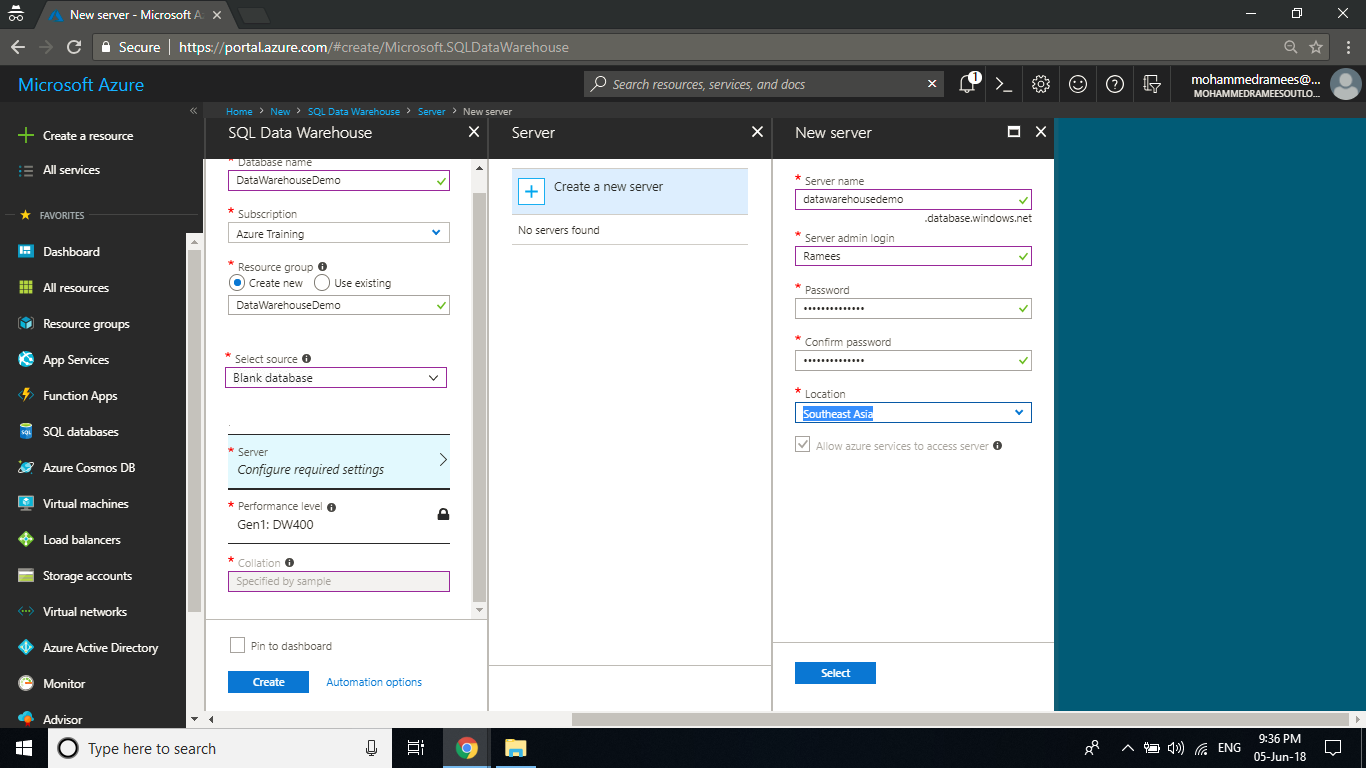
**Select sample:** Blank database

**Server:** Create a new server with unique name and login creadentails. Select a location nearest to you.

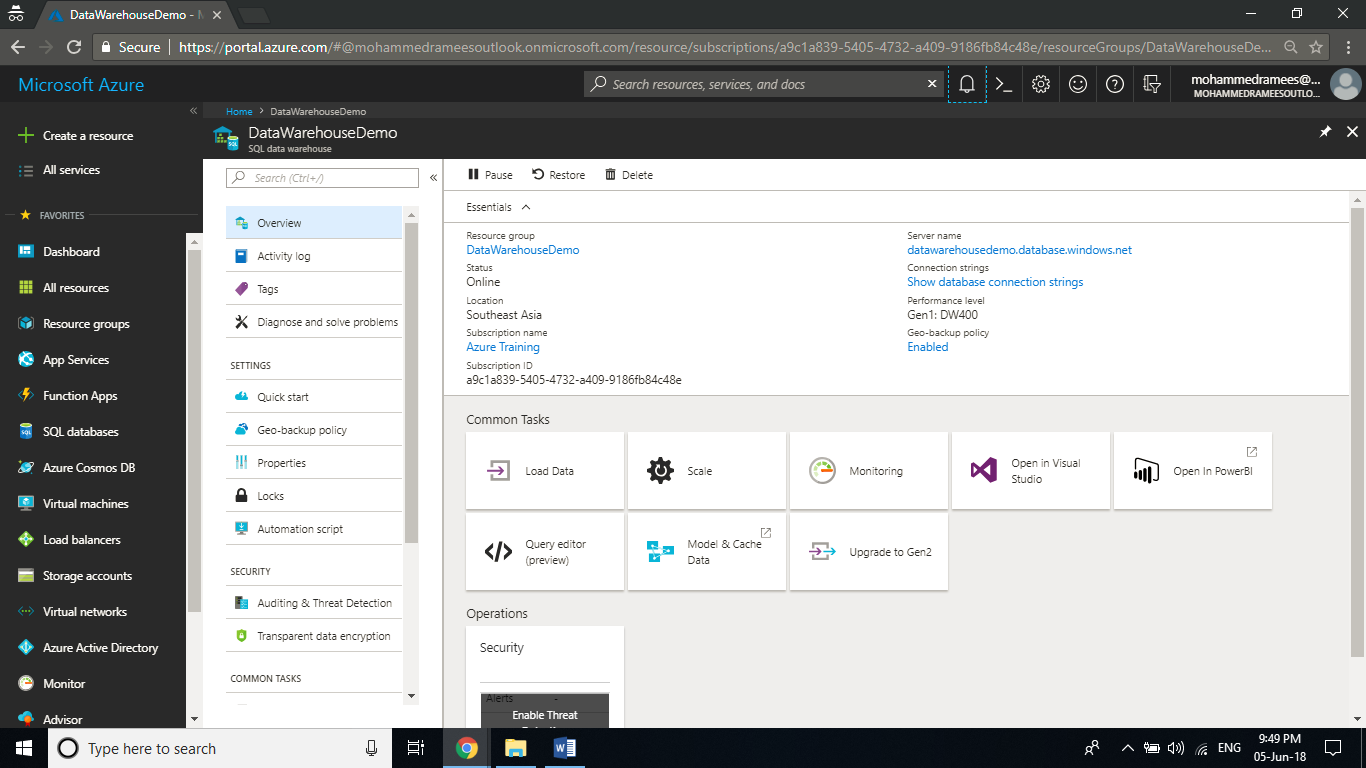
**Perfomance level:** Select DW400 (default)

Click on **Create** to provision the database and wait for few mintues for the provision to be succeded.

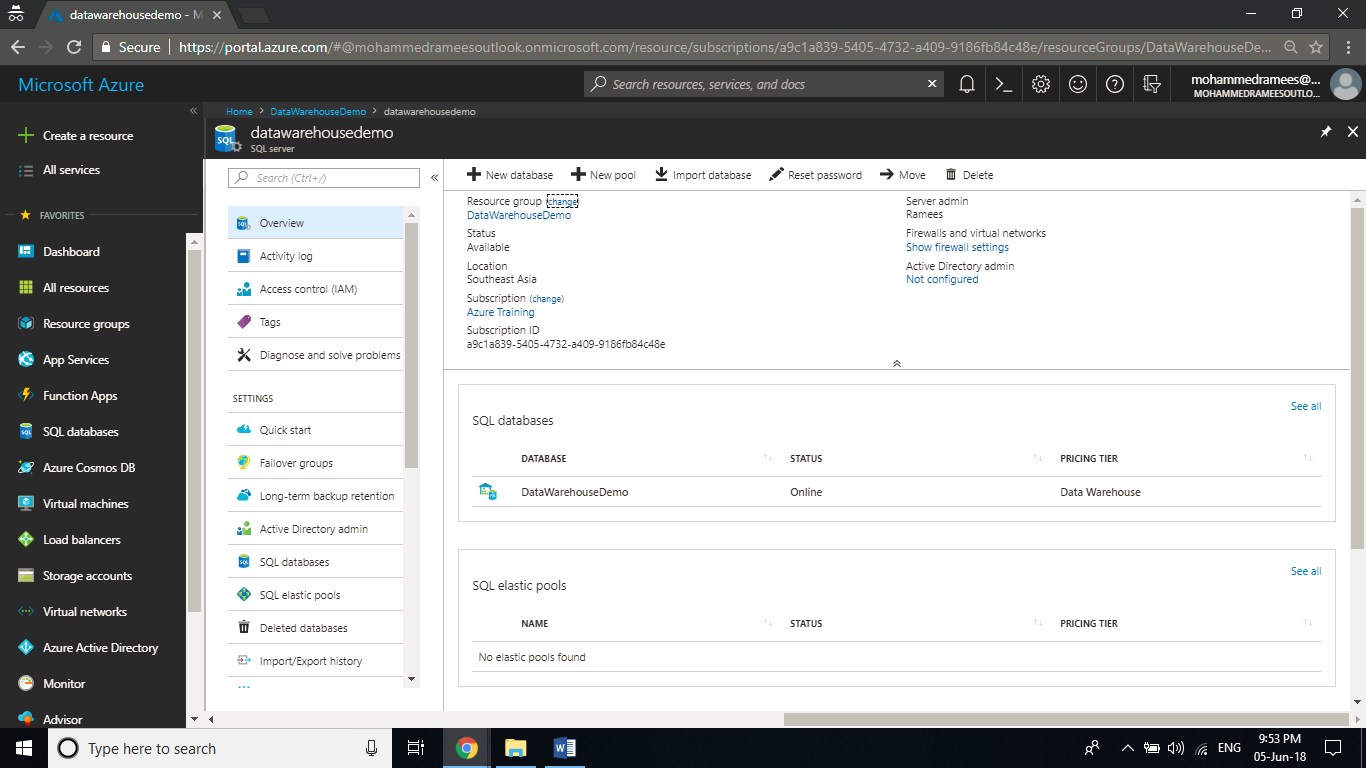




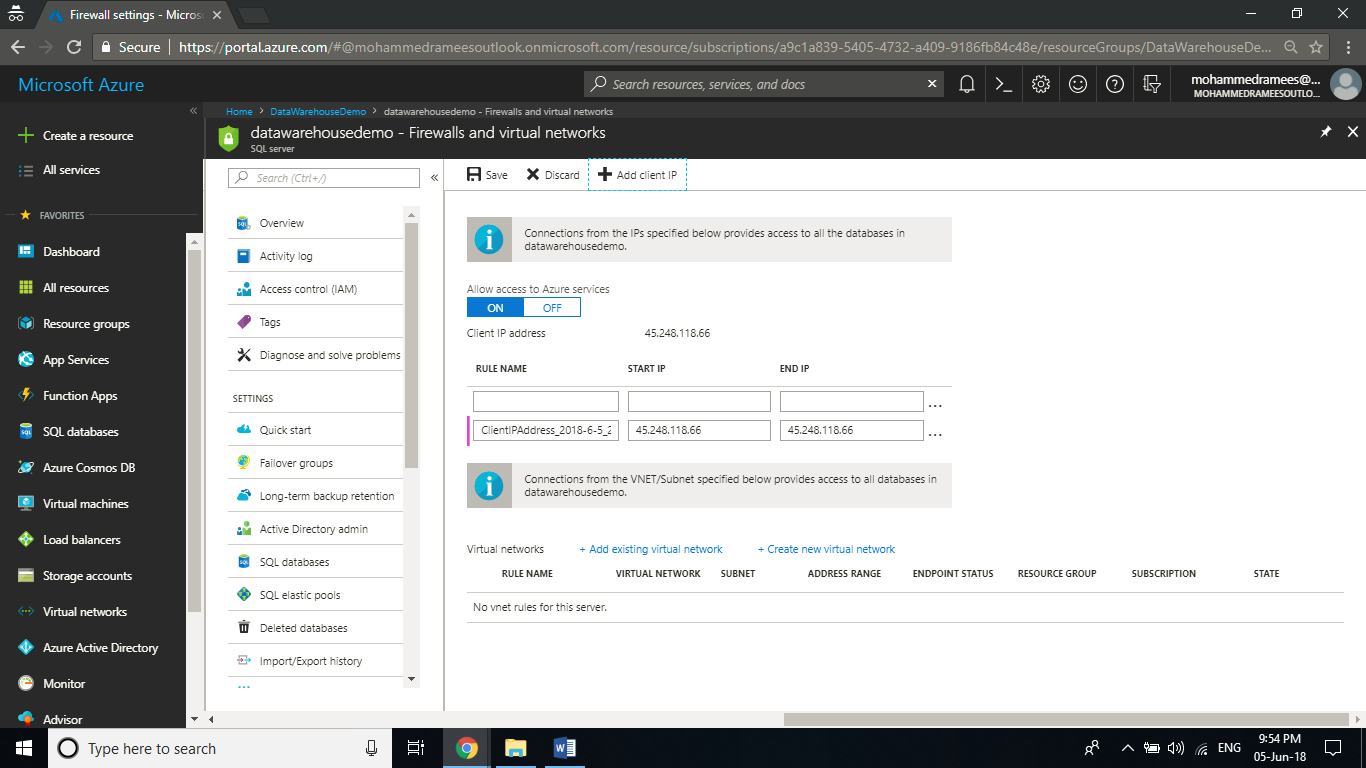
**Step 2:** Once the provision is completed open the resource and you can see the settings blade for the newly created Azure Data Warehouse. Click on the **Server name** to open the settings for the server.

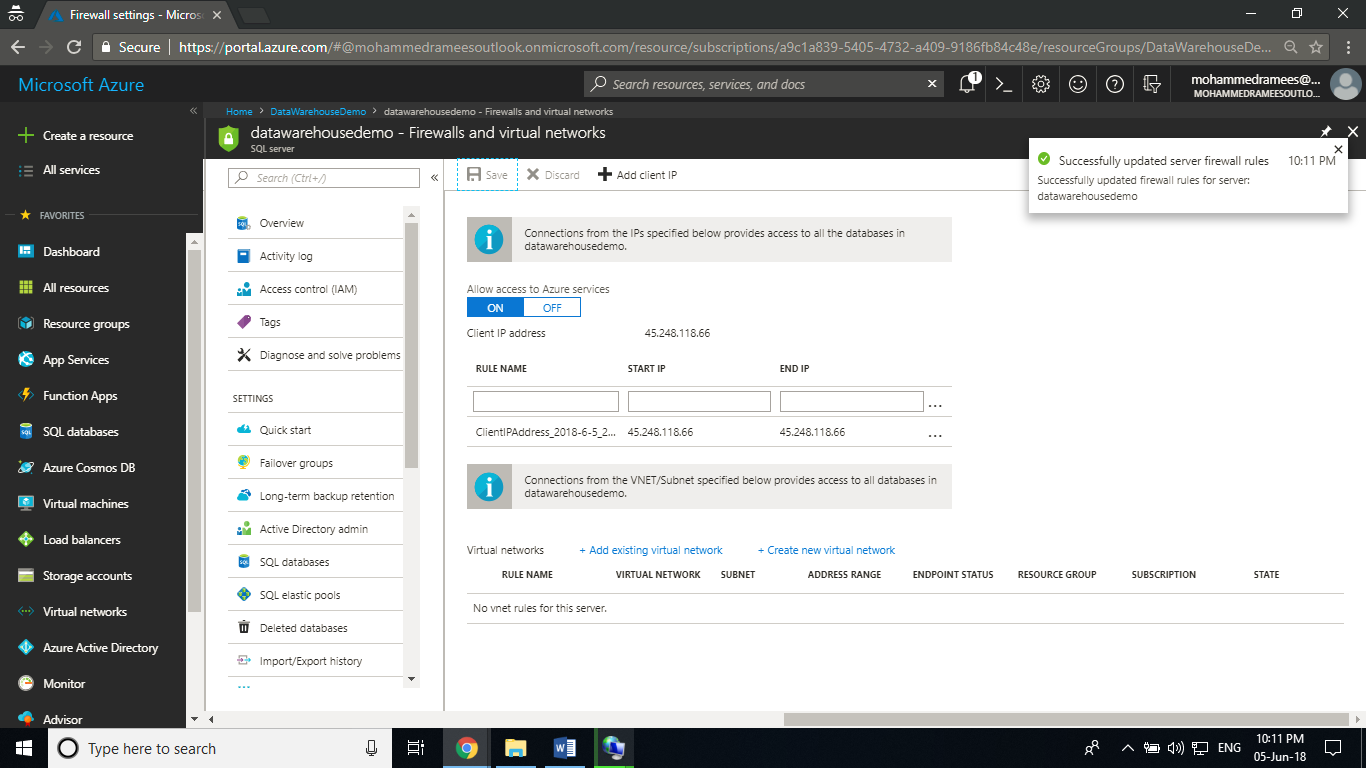


**Step 3:** Now you have got the settings for the server. The SQL Data Warehouse service creates a firewall at the server-level that prevents external applications and tools from connecting to the server or any databases on the server. To enable connectivity, you can add firewall rules that enable connectivity for specific IP addresses. Click on **Show Firewall** **settings** to create a server-level firewall rule for your client's IP address.



**Step 4:** It will open the List of Client IPs allowed to access the server. Click on **Add client IP** to add your current IP address, and click on **Save** to save the changes. A firewall rule can open port 1433 for a single IP address or a range of IP addresses.

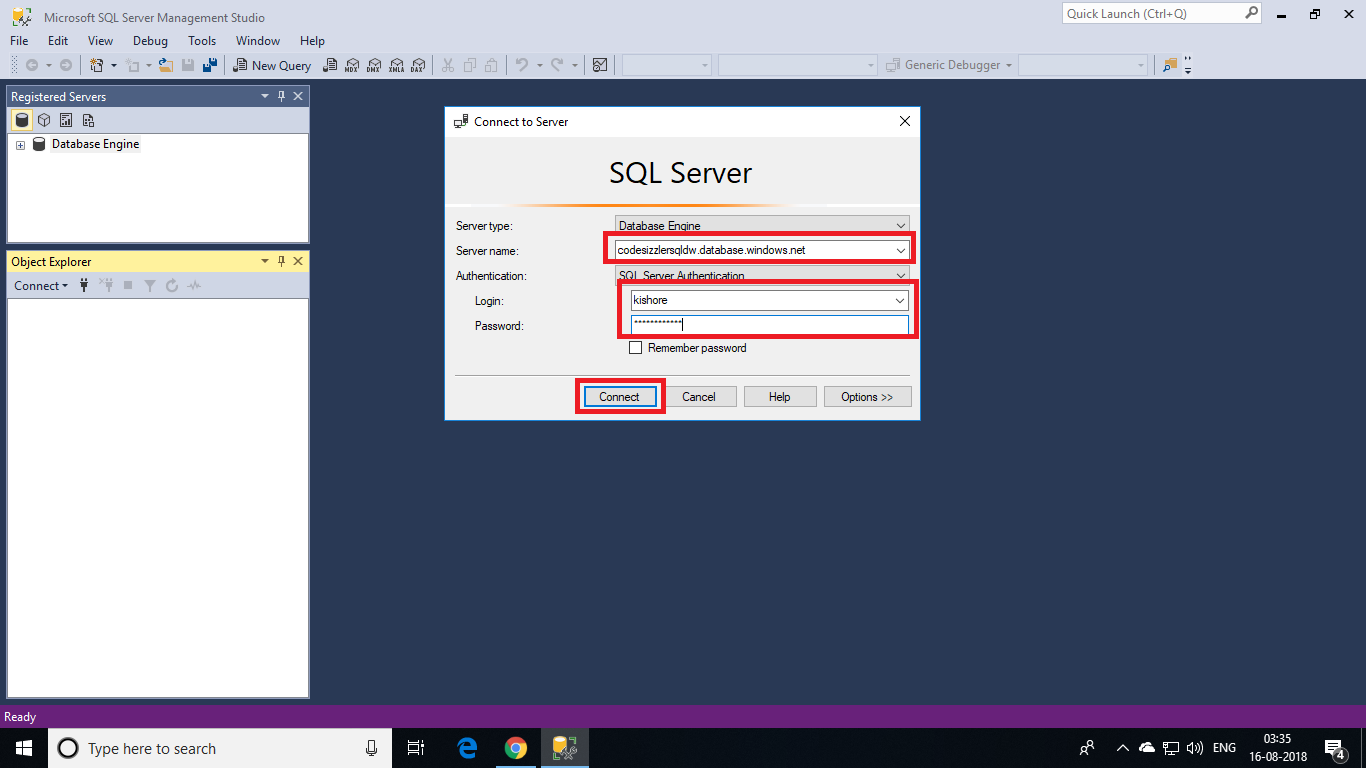




Note down the fully qualified DNS of the SQL Data Warehouse so that you can connect to it from the SQL Server Management Studio.

## **Connecting to the server as server admin:**

This section uses SQL Server Management Studio (SSMS) to establish a connection to your Azure SQL server. Open SQL Server Management Studio and in the **Connect Server** dialog box, enter **DNS** of your **SQL Data Warehouse** and authenticate to it by giving down the user name and password.

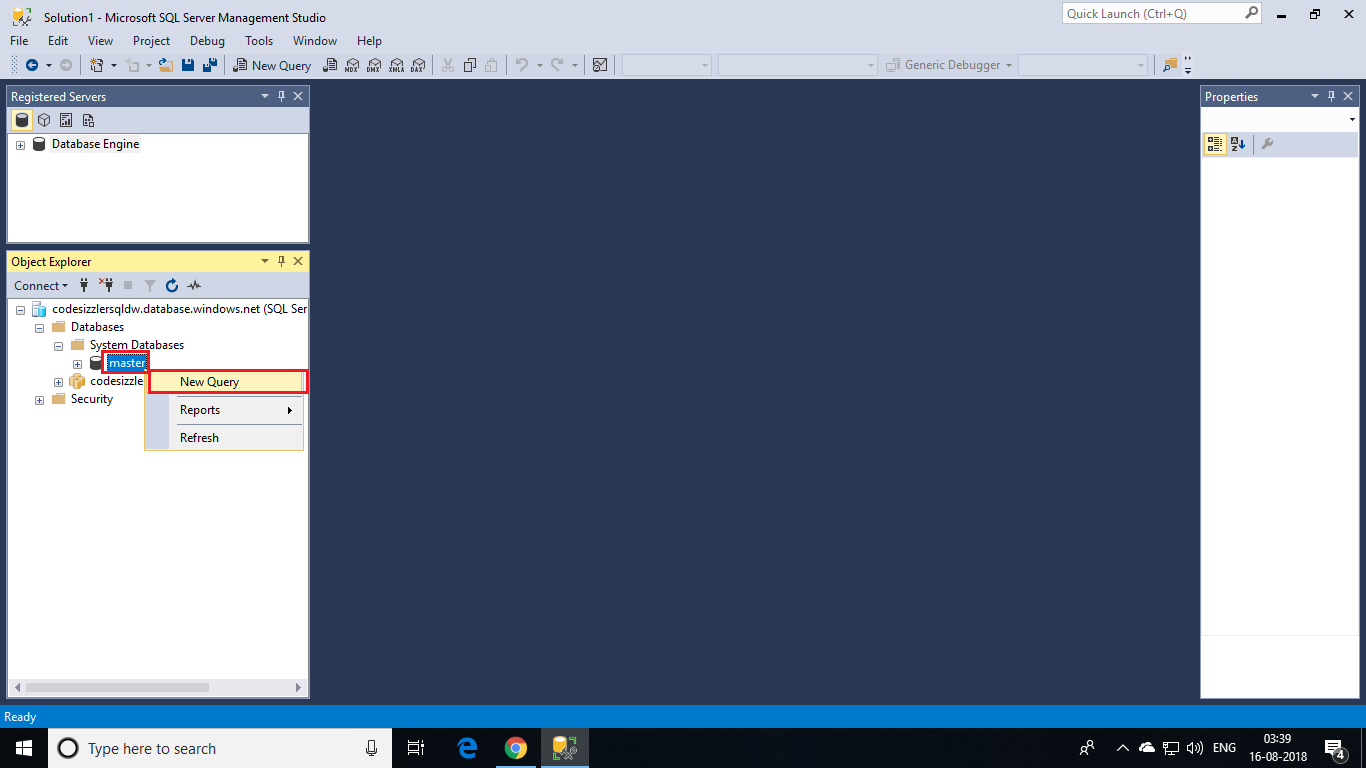


**Create a user for loading data:**

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 **LoadingUser**. Then assign the user to the **staticrc20** resource class.

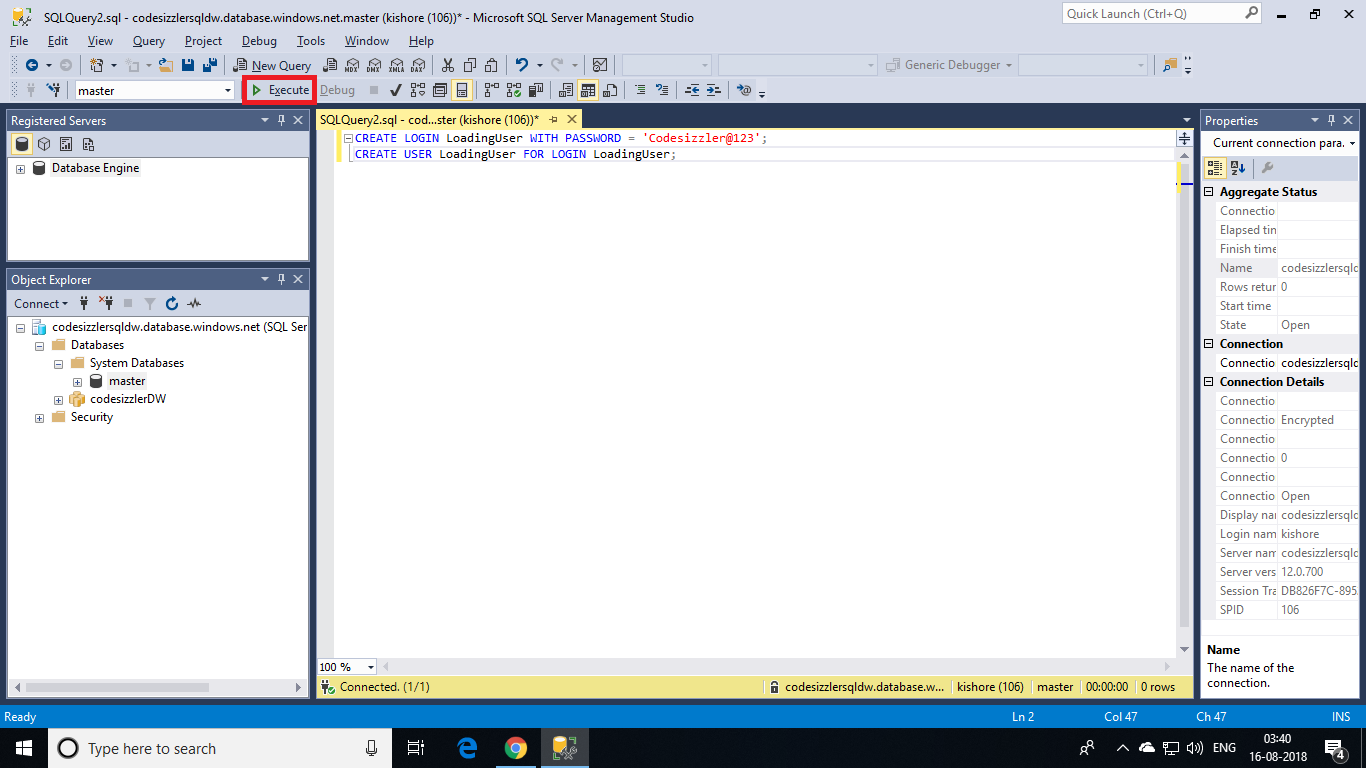
In SSMS, right-click **master** to show a drop-down menu, and choose **New Query**. A new query window opens.



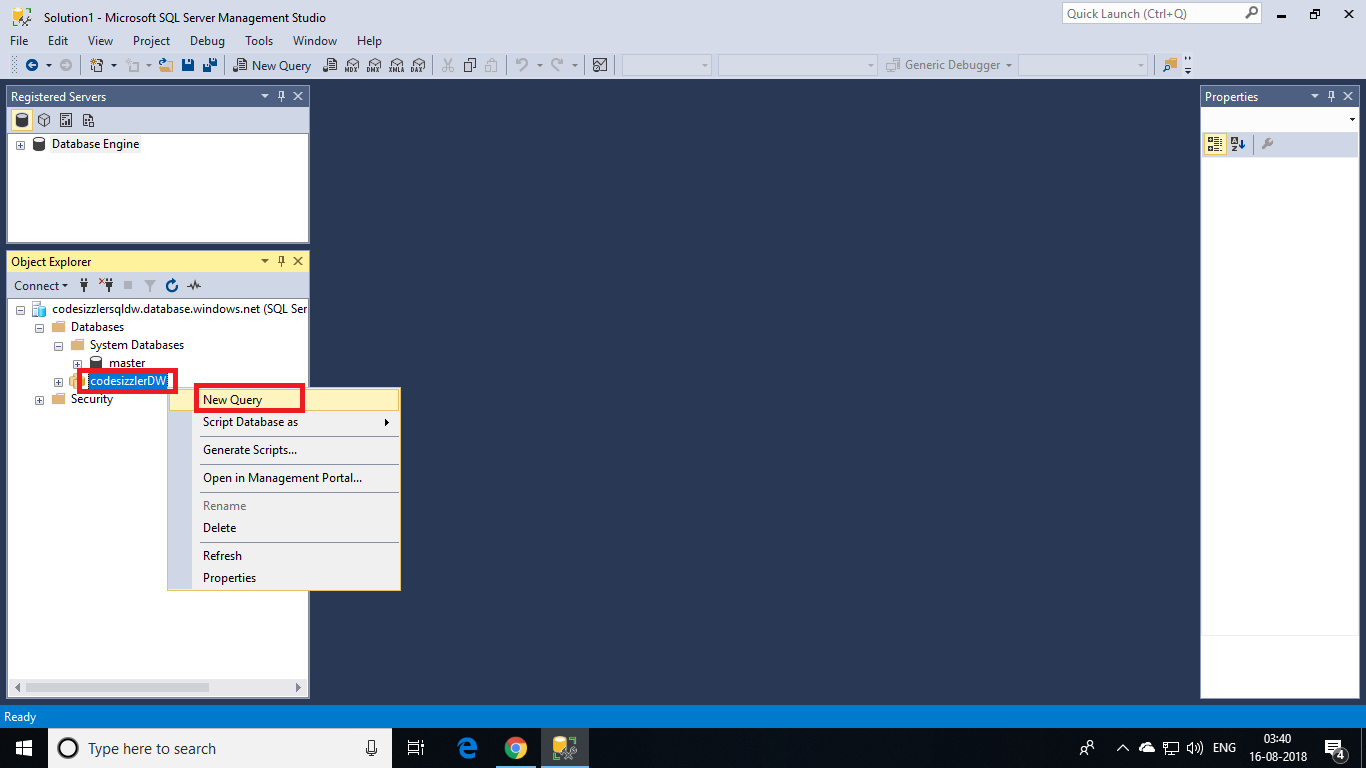
In the query window, enter these T-SQL commands to create a login and user named **LoadingUser**, substituting your own password for ‘Codesizzler@123'. This will create a new user for your Database.

CREATE LOGIN LoadingUser WITH PASSWORD = 'Codesizzler@123';

CREATE USER LoadingUser FOR LOGIN LoadingUser;



Now, right click your Database that you created while deploying the SQL Data Warehouse and click on **New Query**.

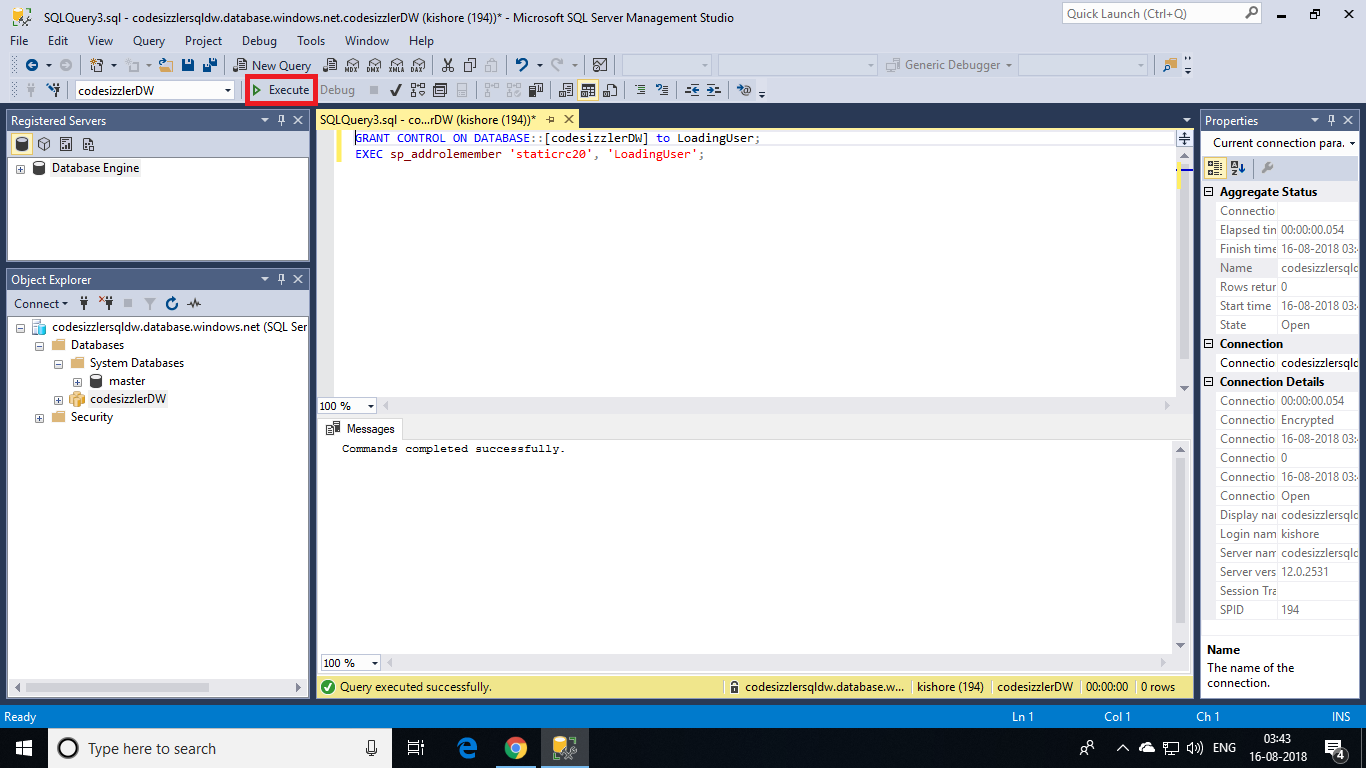


Enter the following commands and execute to grant control for the **LoadingUser** on your database in **Data Warehouse.** Make sure to replace the database name in the command with your database name. It is denoted with an yellow mark

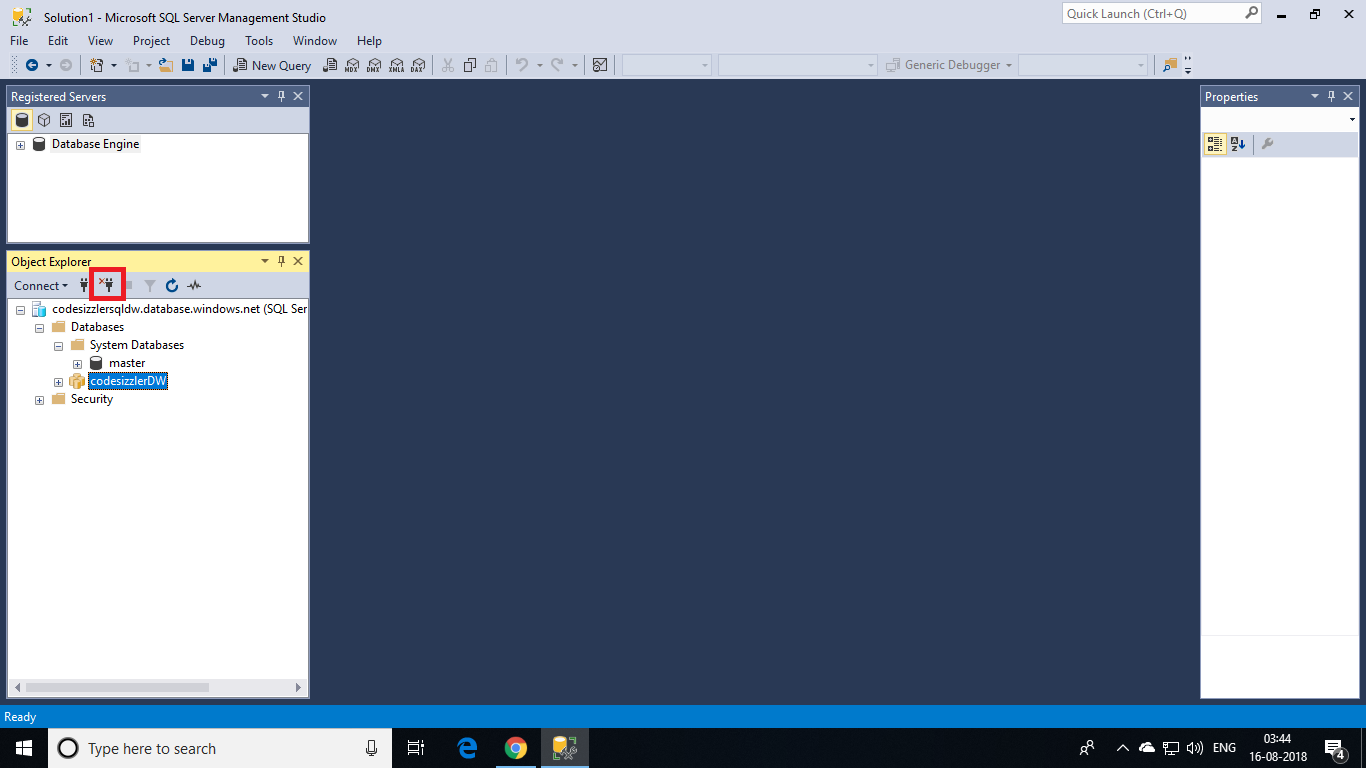
CREATE USER LoadingUser FOR LOGIN LoadingUser;

GRANT CONTROL ON DATABASE::[mySampleDataWarehouse] to LoadingUser;

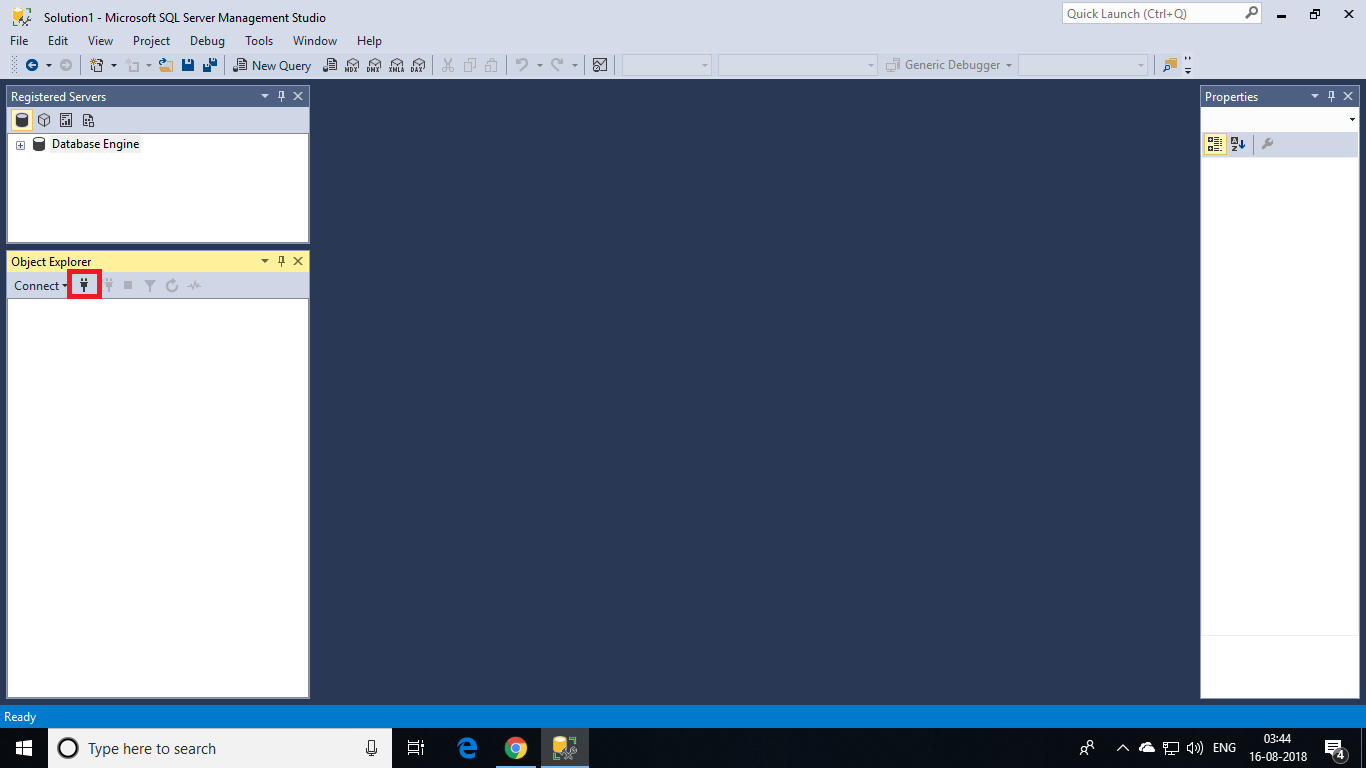
EXEC sp\_addrolemember 'staticrc20', 'LoadingUser';



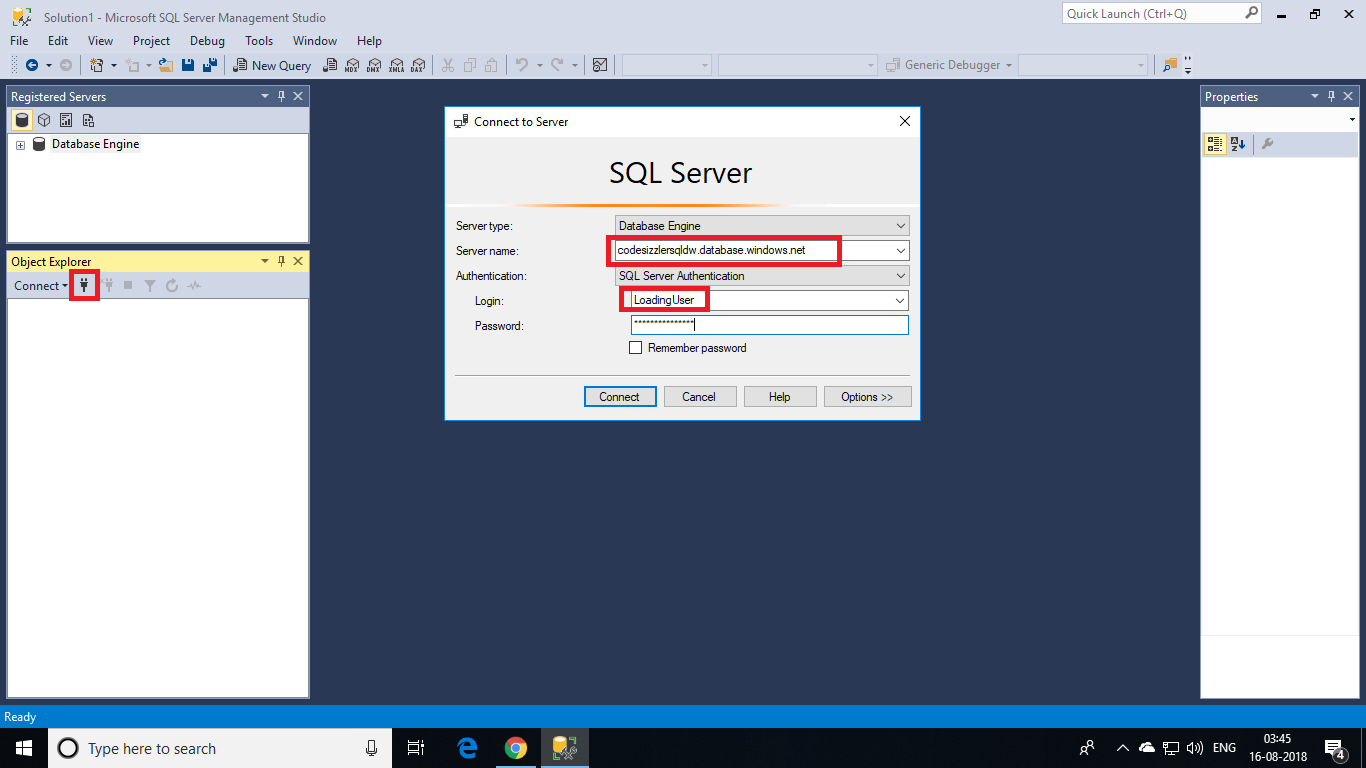
Click the **Disconnect** option and drop the connection of your **SQL Data Warehouse**.



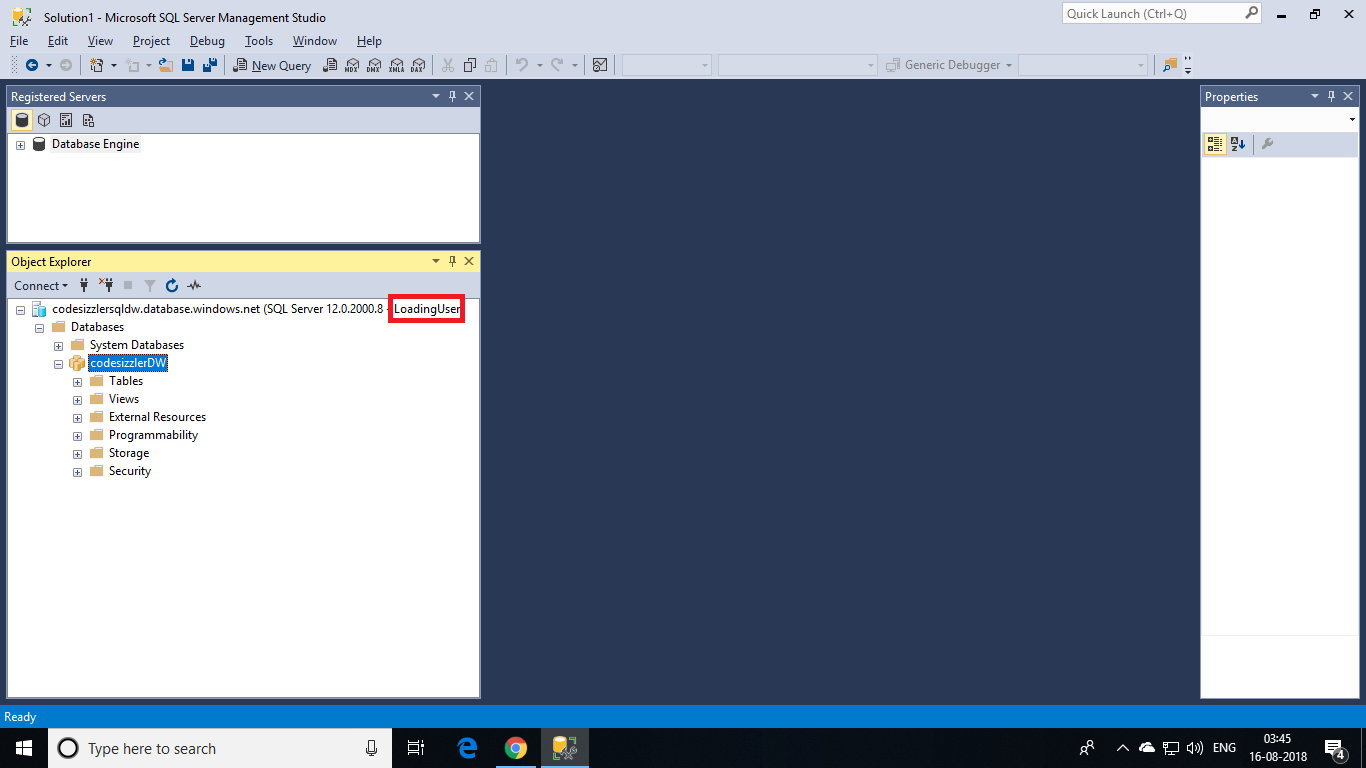
Again, click the **Connect** button.



Again, click the **Connect** option in the object explorer and connect to your **SQL Data Warehouse** using the new user you created in previous step. This will make you to login into the server



When your connection is ready, you will be able to see that you have logged in to your database using the new user.



## **Create external tables for the sample data:**

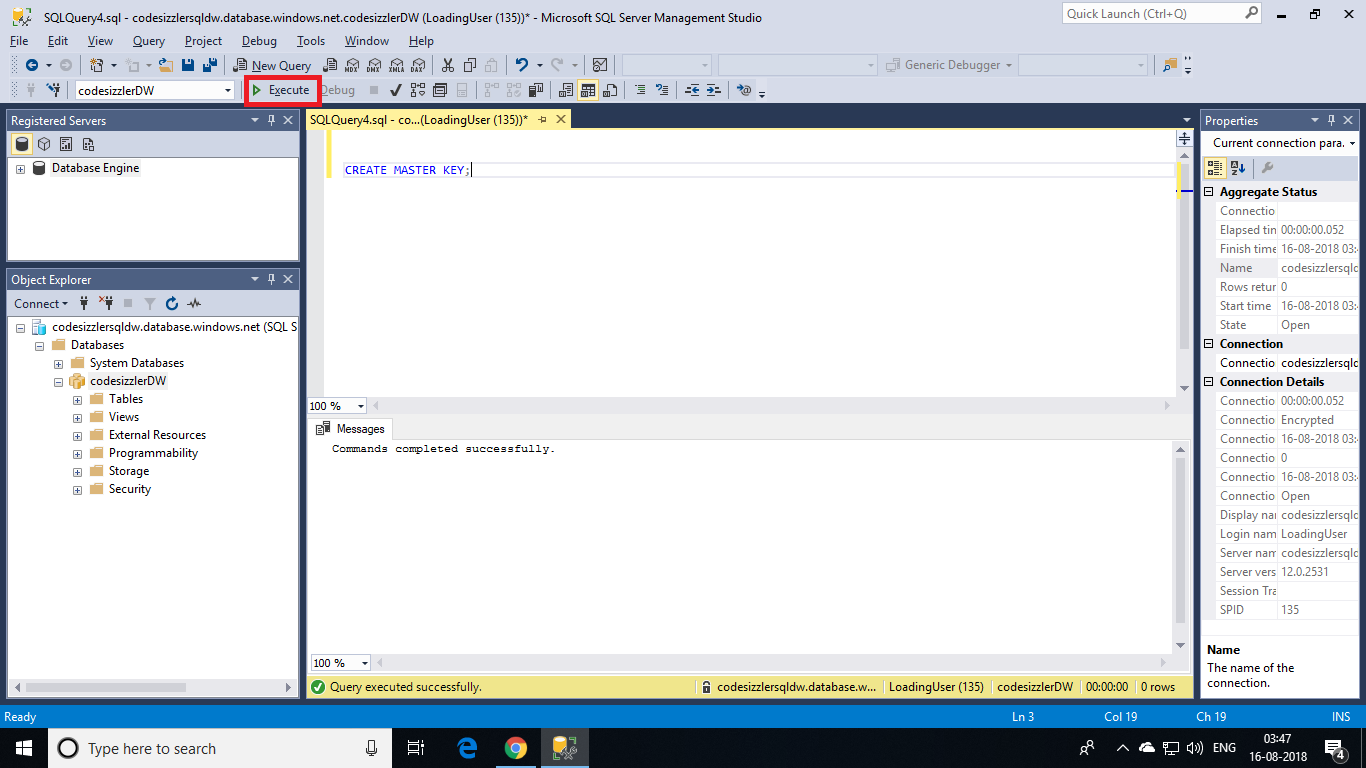
You are ready to begin the process of loading data into your new data warehouse. This tutorial shows you how to use external tables to load New York City taxi cab data from an Azure storage blob.

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.

You logged into your data warehouse as **LoadingUser**. In SSMS, right-click your **Database** connection and select **New Query**. A new query window appears.

Now, Create a master key for the **codesizzlerDW** database. You only need to create a master key once per database.

CREATE MASTER KEY;



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

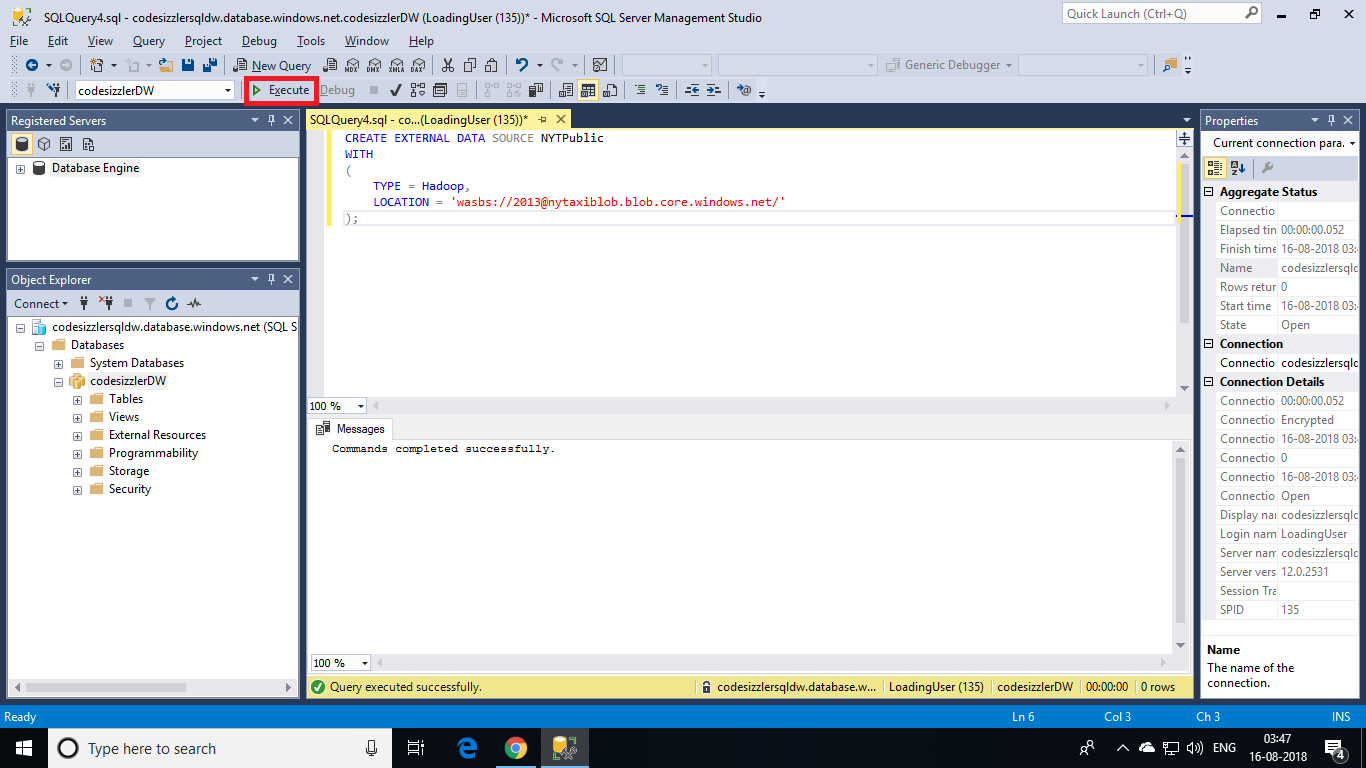
WITH

(

TYPE = Hadoop,

LOCATION = 'wasbs://2013@nytaxiblob.blob.core.windows.net/'

);



Run the following [CREATE EXTERNAL FILE FORMAT](https://docs.microsoft.com/en-us/sql/t-sql/statements/create-external-file-format-transact-sql) 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 = '',

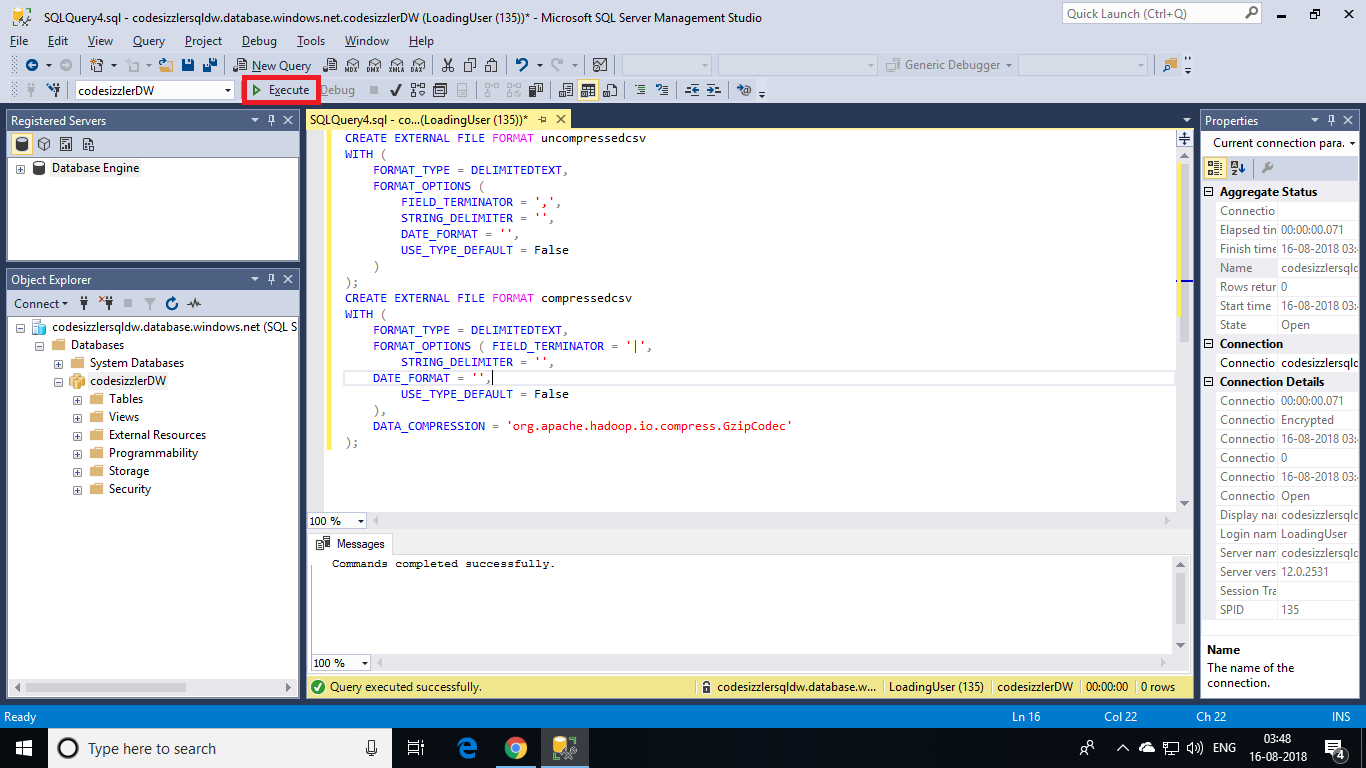
DATE\_FORMAT = '',

USE\_TYPE\_DEFAULT = False

),

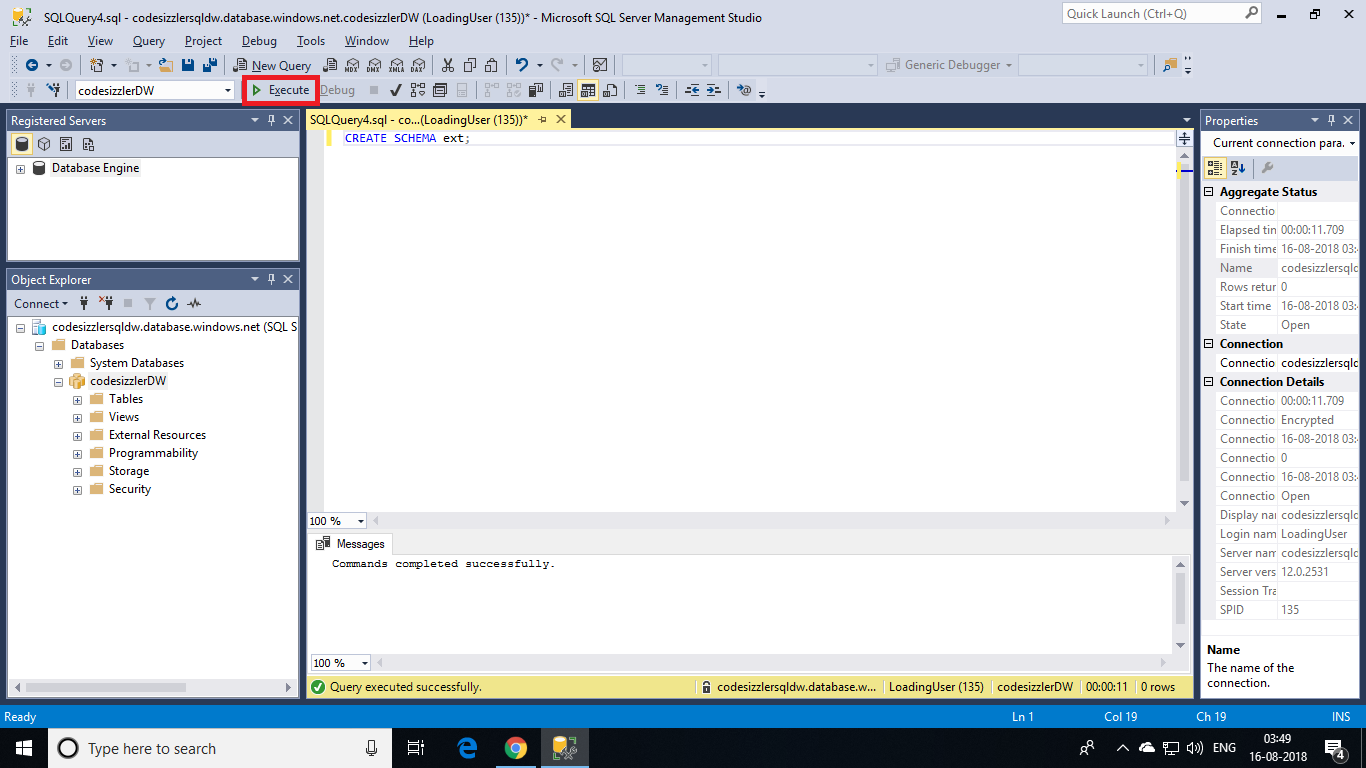
DATA\_COMPRESSION = 'org.apache.hadoop.io.compress.GzipCodec'

);



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;



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.

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,

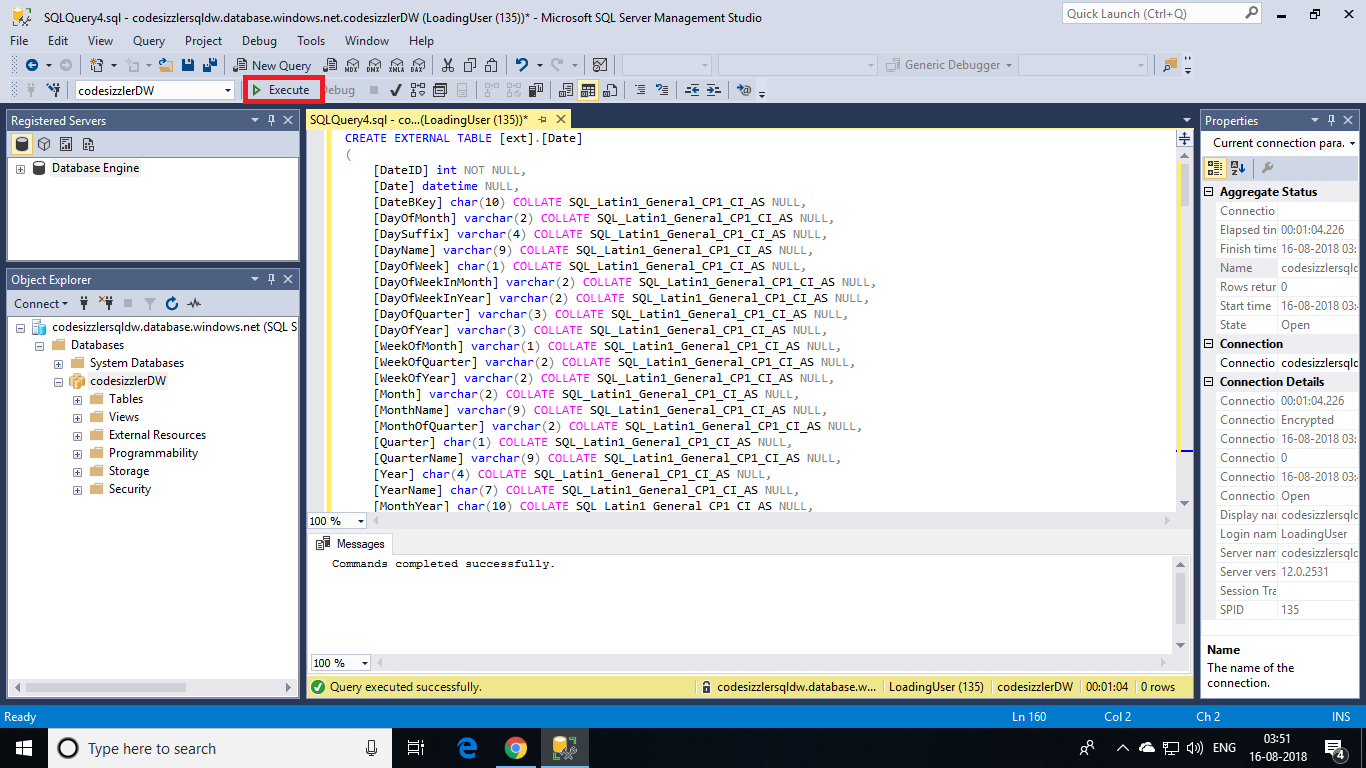
FILE\_FORMAT = uncompressedcsv,

REJECT\_TYPE = value,

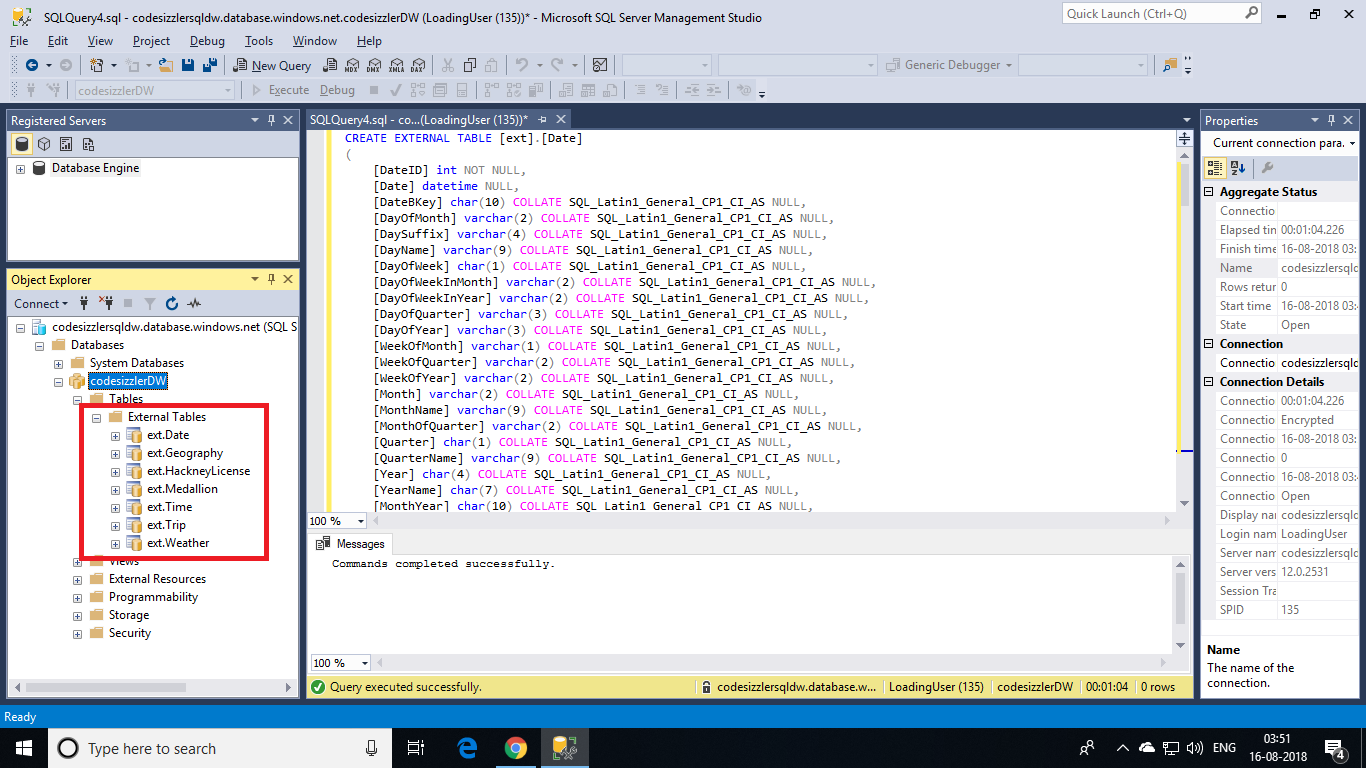
REJECT\_VALUE = 0

)

;



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



## **Loading the data into your data warehouse:**

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

Run the following script to load the data into new tables in your data warehouse.

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,

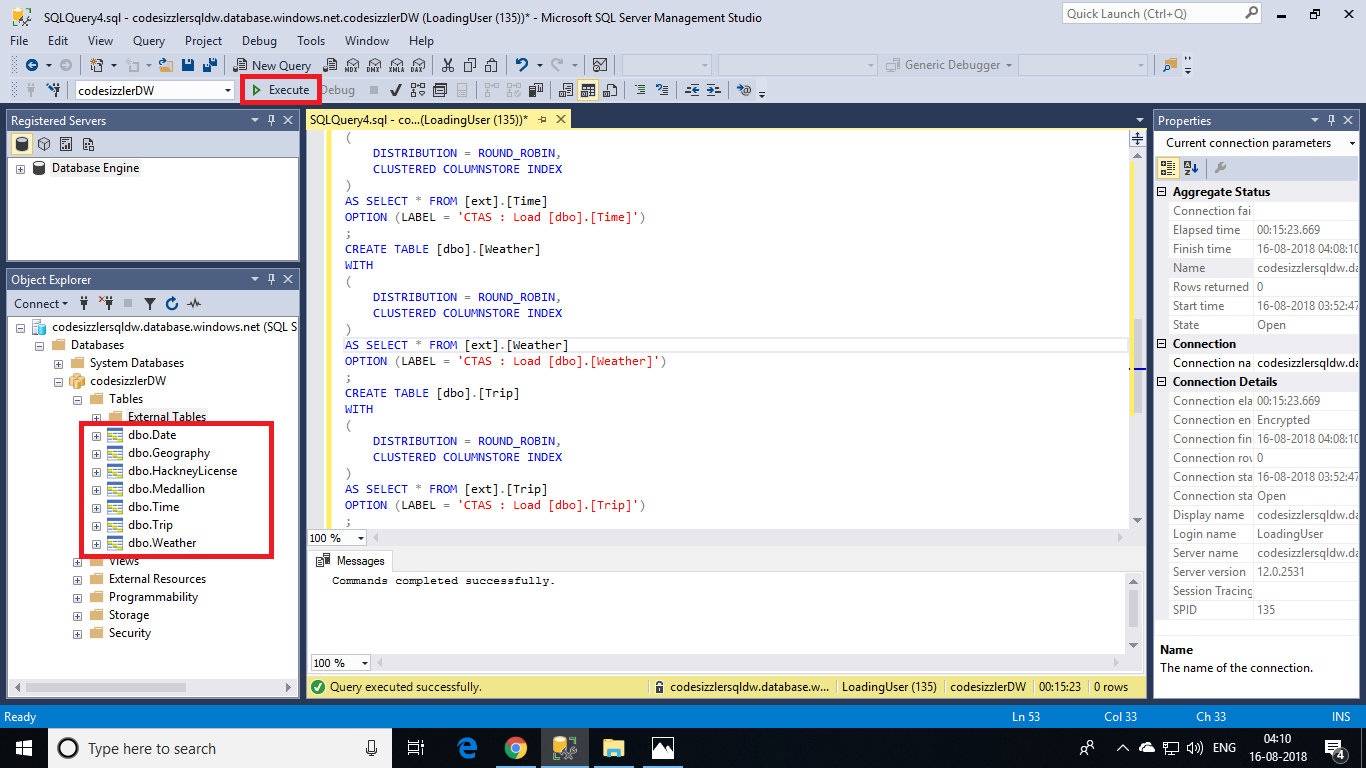
CLUSTERED COLUMNSTORE INDEX

)

AS SELECT \* FROM [ext].[Trip]

OPTION (LABEL = 'CTAS : Load [dbo].[Trip]')

;



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. After starting the query, grab a coffee and a snack while SQL Data Warehouse does some heavy lifting.

SELECT

r.command,

s.request\_id,

r.status,

count(distinct input\_name) as nbr\_files,

sum(s.bytes\_processed)/1024/1024/1024 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,

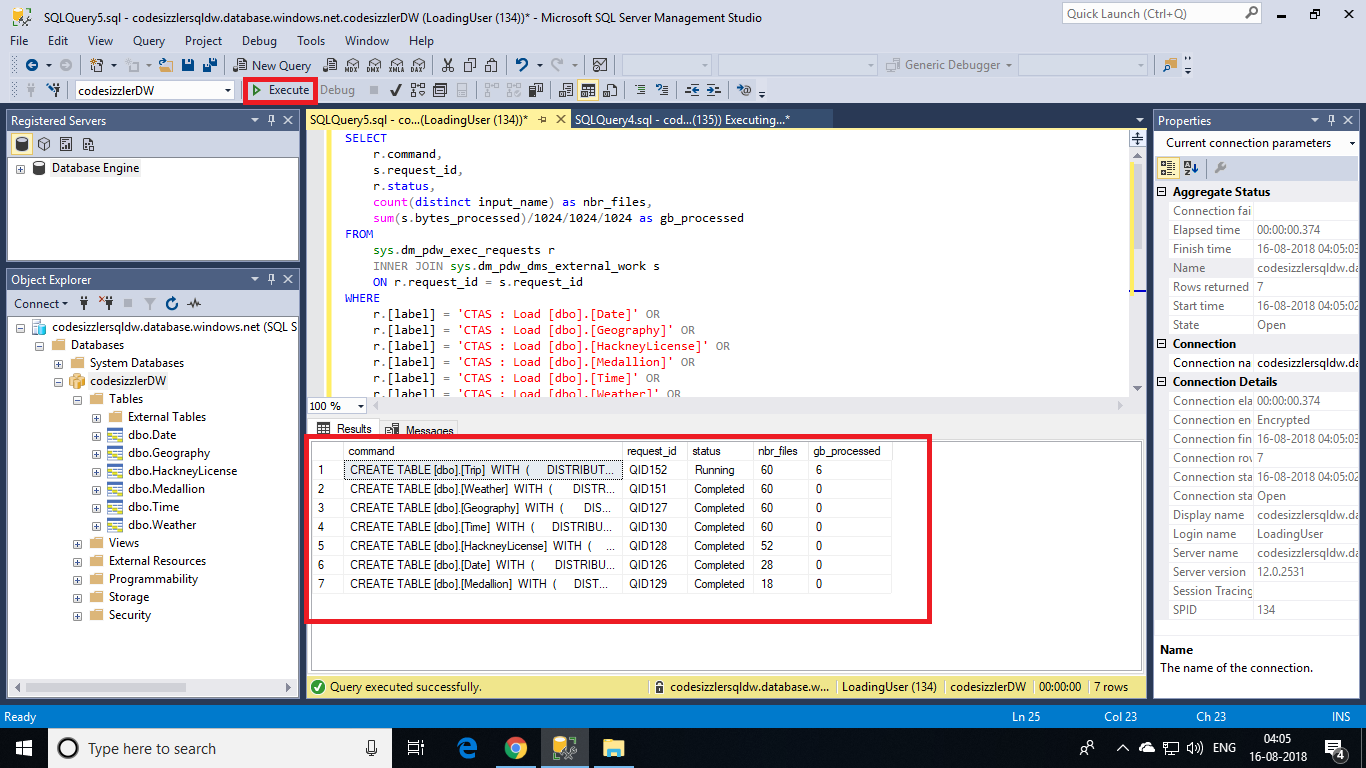
s.request\_id,

r.status

ORDER BY

nbr\_files desc,

gb\_processed desc;



## **Create statistics on newly loaded data:**

SQL Data Warehouse does not auto-create or auto-update statistics. Therefore, 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.

Run these commands to create statistics on columns that are likely to be used in joins.

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

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

