Microsoft Azure - Starter Kits for Partners

Hands on Lab

Power BI Scenario

Create an Azure SQL Data Warehouse and visualize with Power BI

Last Update: Aug 2016





**MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.**

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The descriptions of other companies’ products in this document, if any, are provided only as a convenience to you. Any such references should not be considered an endorsement or support by Microsoft. Microsoft cannot guarantee their accuracy, and the products may change over time. Also, the descriptions are intended as brief highlights to aid understanding, rather than as thorough coverage. For authoritative descriptions of these products, please consult their respective manufacturers.

© 2016 Microsoft Corporation. All rights reserved. Any use or distribution of these materials without express authorization of Microsoft Corp. is strictly prohibited.

Microsoft and Windows are either registered trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Contents

[Overview 5](#_Toc458451905)

[Objectives 5](#_Toc458451906)

[Prerequisites 6](#_Toc458451907)

[Exercises 6](#_Toc458451908)

[Infrastructure Provisioning 6](#_Toc458451909)

[Exercise 1: Creating an Azure SQL Data Warehouse (New Portal/ARM) 6](#_Toc458451910)

[Exercise 2: Loading data into Azure SQL Data Warehouse using Polybase 16](#_Toc458451911)

[Exercise 3: Azure SQL Data Warehouse integrated with PowerBI 34](#_Toc458451912)

## Overview

In this lab, we walk you through building and deploying a machine learning model using SQL Data Warehouse (SQL DW) for a publicly available dataset -- the NYC Taxi Trips dataset. The binary classification model constructed predicts whether or not a tip is paid for a trip, and models for multiclass classification and regression are also discussed that predict the distribution for the tip amounts paid. The procedure follows the Cortana Analytics Process (CAP) workflow. We show how to setup a data science environment, how to load the data into SQL DW, and how use either SQL DW or an IPython Notebook to explore the data and engineer features to model. We then show how to build and deploy a model with Azure Machine Learning

**Estimated time** to complete this lab: **180 minutes**.

**Audience**: IT Pro, Architect, Application Owners and Developers

### Objectives

In this hands-on lab, you will learn how to:

1. Create an Azure SQL Data Warehouse
2. Load data from a cloud data store using Polybase
3. Visualize the data in Power BI

Each task is separated into ‘Parts’ in this demo script.

The following messaging points will be delivered as part of the tasks above:

1. Ease of deployment
2. Cloud-based MPP data warehouse service
3. Region availability
4. Elastic-scale capability including pause
5. Cost friendly
6. Control over compute separate to storage
7. Data encryption
8. Compliance and auditing
9. Integrated experience
10. Common tooling
11. Rapid and rich visualizations

### Prerequisites

The following is required to complete this hands-on lab:

* A Microsoft Azure subscription - [sign up for a free trial](http://aka.ms/WATK-FreeTrial)
* [Azure PowerShell 0.7.4  or higher](https://azure.microsoft.com/en-us/documentation/articles/powershell-install-configure/)
* Visual Studio 2015

## Exercises

## Infrastructure Provisioning

### Exercise 1: Creating an Azure SQL Data Warehouse (New Portal/ARM)

1. ***Pre-requisites***
2. 1. Azure Subscription with $50 of credit
3. ***Scenario***

Today businesses are collecting more information than ever before to drive insight based on data. To succeed we require three things – scaled data stores, tools to deliver insight and an agile approach to get the business answers quickly.

Let’s start by addressing agility and scale. The Azure SQL Data Warehouse is the industry’s first enterprise class cloud-based data warehouse with dynamic scale. What used to take days or weeks to achieve, we can literally do in minutes.

1. ***Steps***

Creating a data warehouse couldn’t be simpler.

We simply specify the data warehouse details (server, resource group, database, region, etc.).

If we look at the process, we don’t need to select the size of your datawarehouse on disk – Azure grows it dynamically as you need it. This is what is called ‘elastic scale’ and Microsoft are the first to bring this capability to the cloud data warehouse.

We are building an MPP data warehouse and so query speed is super important. By simply moving the slider to the right we add more CPUs into the mix and get the query speed I need. That’s a whole lot simpler than building a network of machines.

The default is 400 DWU (Data Warehousing Unit) and is a good starting point for creating a data warehouse. More information on DWU and concurrency can be found at the following URL:

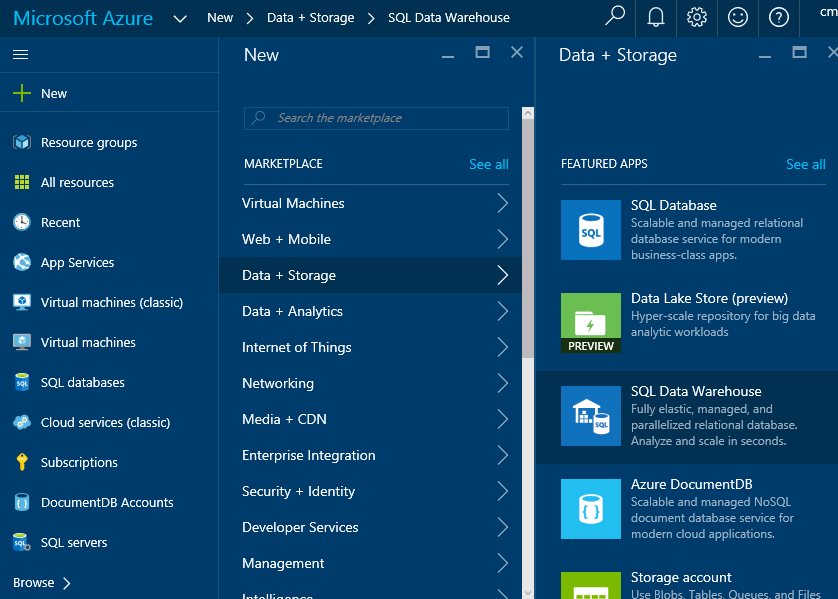
<https://azure.microsoft.com/en-gb/documentation/articles/sql-data-warehouse-develop-concurrency/>

We can also programmatically alter the amount of compute using PowerShell, T-SQL or REST. More information can be found at the following URL:

<https://azure.microsoft.com/en-gb/documentation/articles/sql-data-warehouse-manage-compute-overview/>

When creating our data warehouse we must provide the details for the server, database, ‘sa’ login, etc.

1. Navigate to **https://portal.azure.com** and **sign in** with your Microsoft account.
2. Click **‘+NEW’** then ‘**Data + Storage’** and **‘SQL Data Warehouse’**



1. Fill out the fields in order:
2. Name – **myclouddw**
3. Subscription – If you have multiple subscriptions select the chosen one from the drop-down.
4. Resource Group – Create New - **rgmyclouddw**
5. Source – You can choose to deploy a blank database or use the Sample database (AdventureWorks). We will load new data (unrelated to the sample database) during this lab.
6. Click ‘**Server’** -> Configure required settings (!) -> Create new server
7. Enter the server details as follows:

**Server Name** – enter your choice of server name

**Server Admin login** – your chosen ‘SA’ account name

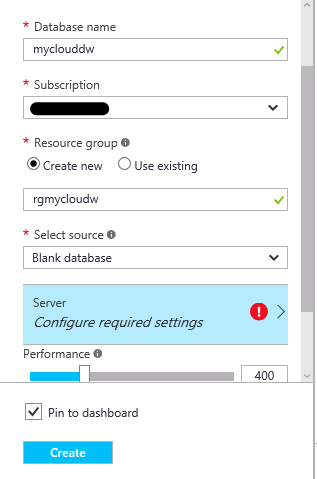
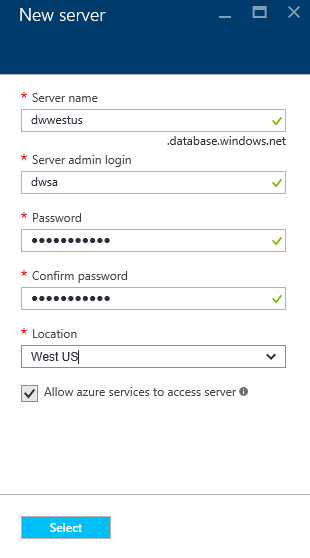
**Password** – The password you would like to set for the server

Confirm password – Confirm the password as entered in the previous step

**Location** – Set this to West US (or you may set it to a region of your preference)

Allow Azure services to access server – Leave checked\enabled (default)

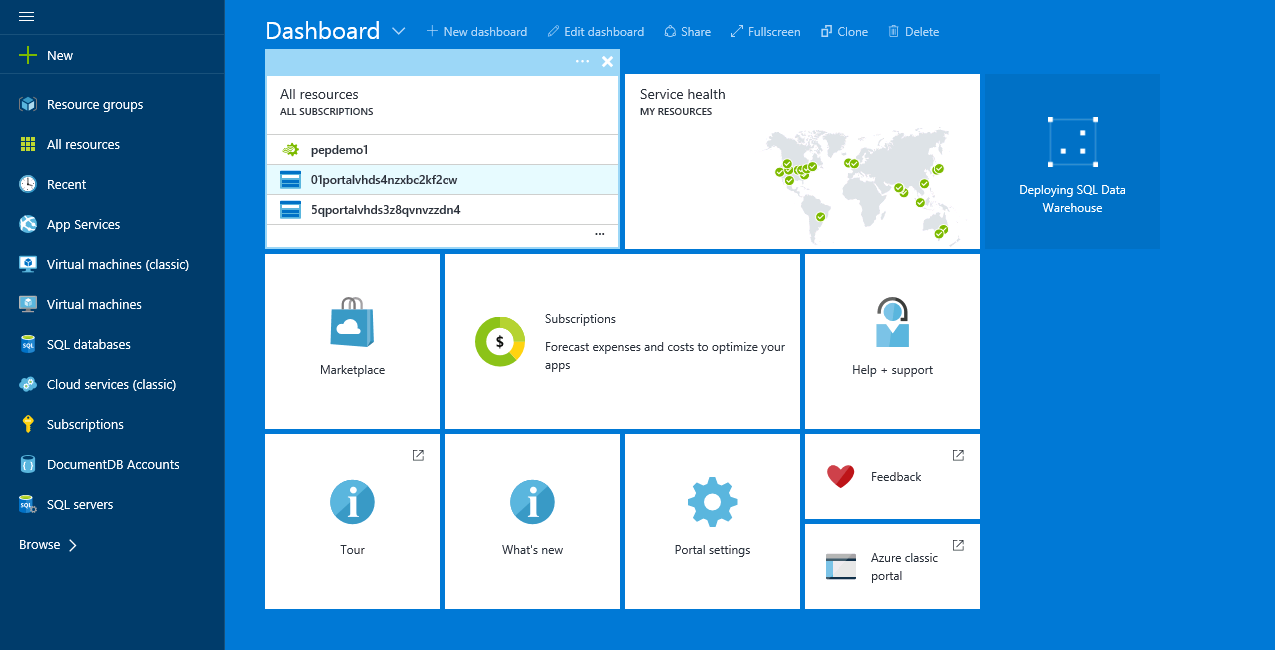
1. Once complete, click Select at the bottom of the blade to save the configuration and return to the previous blade screen.

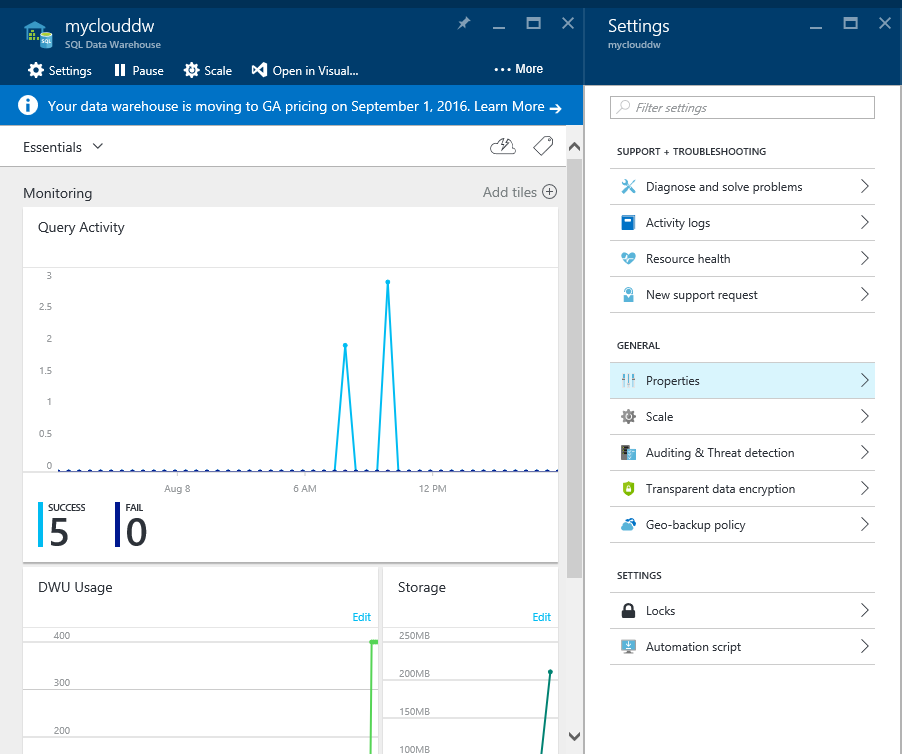
1. Check ‘**Pin to dashboard**’
2. Click ‘**Create’** to begin the provisioning process, this will take you back to the Portal homepage.



The data warehouse will take a few minutes to create. Once created, click on the tile located on the dashboard to open the data warehouse blade.

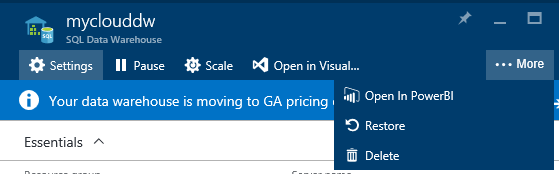


1. Click on the data warehouse tile located on the dashboard. We will now explore the data warehouse blade. We have a toolbar across the top of the blade and the Settings blade located to the right hand side.



Looking at the toolbar we see:

Settings, Pause, Scale, Open in Visual Studio, Open in PowerBI, Restore, Delete.



**Settings** opens the Settings blade as already shown.

**Pause** allows us to Pause (or Resume) the data warehouse service.

**Scale** enables us to leverage the elastic scale capabilities and increase or decrease the amount of compute resources assigned to the data warehouse.

**Open in Visual Studio** enables direct toolset integration with Visual Studio enabling a rapid and integrated experience for administrators and developers alike.

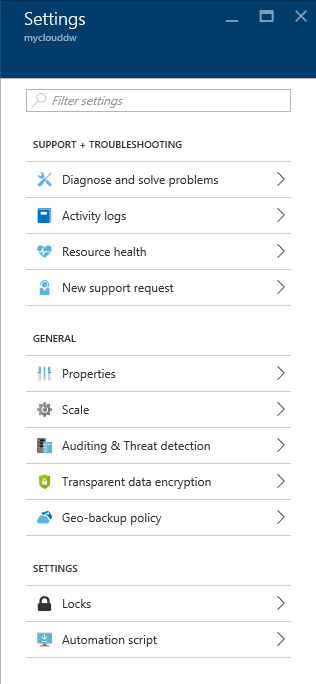
**Open in PowerBI** is a unique integration feature with Azure SQL Data Warehouse which enables insight against data directly from within the data warehouse blade.

**Restore** enables restores of databases. Regular backups are taken by Azure automatically and are available for restoration should it be required.

**Delete** will delete the data warehouse from the server.

Looking at the Settings blade we see the sections:

Support & Troubleshooting, Resource Management and General



**Support & Troubleshooting**

**Diagnose and solve problems >Troubleshoot** enables an online troubleshooter tool.

**Activity Logs** displays management activity logs around the data warehouse service.

**Resource Health** displays the health of the data warehouse.

**New support request** enables a support ticket to be raised directly from within the data warehouse blade.

**General**

**Properties** displays the properties of the data warehouse (collation, server name, resources, location, subscription, etc.)

**Scale** enables changing the amount of compute resource and cost model using the intuitive power of a slider which shows compute usage and hourly cost.

**Auditing & threat detection** is where we configure the auditing (login tracking, query execution, etc.) and enable output of the logs to Azure Blob Storage for loading into the data warehouse for operational data integration and analysis.

**Transparent data encryption** enables or disables encryption of data whilst at rest whilst stored in the data warehouse.

**Geo-backup policy-** Geo-backup copies one of the automatic snapshots each day to RA-GRS storage. This can be used in an event of a disaster to recover your data warehouse to a new region.

**Settings**

**Locks** displays any resource locks impacting the service or workload.

**Automation script** deploys resources with Azure Resource Manager templates in a single, coordinated operation. Define resources and configurable input parameters and deploy with script or code.

We have now completed the creation of our data warehouse and are familiar with the security, auditing and compliance capabilities as well as the differentiators of elastic scale, pause and resume along with a full suite of cloud services to plug in to and do ‘anything analytical’ with your data.

Now we have a data warehouse, we need some data….

**Step by step:** <https://azure.microsoft.com/en-us/documentation/articles/sql-data-warehouse-get-started-provision/>

### Exercise 2: Loading data into Azure SQL Data Warehouse using Polybase

1. ***Pre-requisites***
2. Visual Studio 2015 Community Edition has been provided for this part of the lab.
3. We have provided a template script to perform all of the stages of this lab. The script can be found within the virtual machine at the following location:
4. “C:\xxx\Sample Data”
5. ***Scenario***

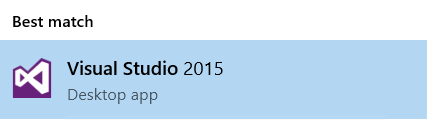
So we have created our data warehouse and now we want to load data into it. We can do this through the traditional ways of ETL and tooling such as SQL Server Integration Services or third-party tooling. However today we are going to use Polybase; a Big Data technology created by Microsoft in partnership with Gray Systems Laboratories in Cambridge which allows querying of data located in Hadoop, Azure HDInsight or Azure Blob storage.

Our source data is located in Azure Blob Storage in the West US region and we will query and load this data using Polybase. The data is based on the NYC Taxi public dataset which was originally obtained from <http://chriswhong.com/open-data/foil_nyc_taxi/>.

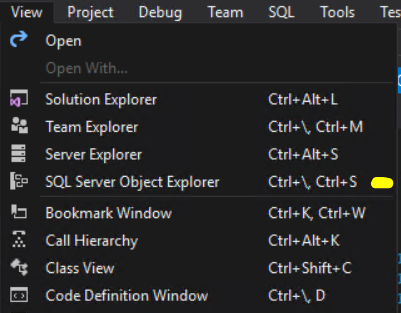
1. ***Steps***

We need to open our data management toolset in order to prepare the data warehouse with a security, schema and data. SQL Server Data Tools (available within all versions of Visual Studio 2015) is the toolset to use within Visual Studio 2015 for Azure SQL Data Warehouse.

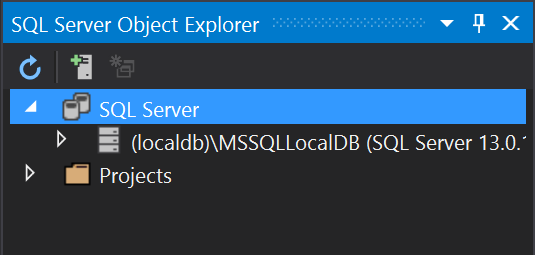
1. Open Visual Studio 2015 by clicking on the Start button and typing **Visual Studio**. Select the Visual Studio 2015 desktop app icon.



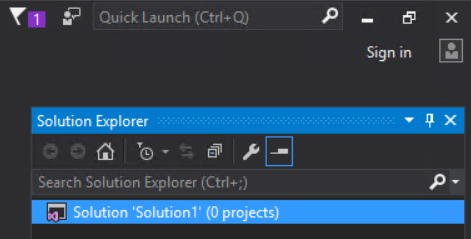
1. Sign in to Visual Studio by using a Microsoft account.
2. On the toolbar click View – SQL Server Object Explorer



1. You should now see the new pane appear docked on the left-hand side within the main window.



1. Locate the Solution Explorer pane to the right hand side.

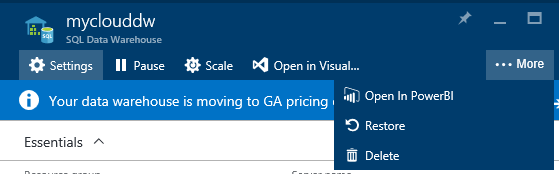


1. Click the Pin icon



We will now open the template script within Visual Studio and review it. If we wanted to we can manually specify the connection details within the SQL Server Object Explorer pane, however Azure SQL Data Warehouse is integrated with other Azure services directly from the Azure portal. One of those integration points is the ability to launch Vis-ual Studio and provide the connection details (minus the password, this requires filling in) to create a connection to the data warehouse.

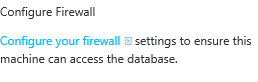
1. Navigate to the Azure portal within the web browser which should be open from the last exercise. If not, open the browser and navigate to <https://portal.azure.com>
2. Open the Azure SQL Data Ware-house blade from the tile on the portal dashboard (you pinned it in the earlier exercise)
3. Looking at the top of the blade you will see a row of icons as shown in the screenshot.



1. Click the ‘**Open in Visual Studio**’ button.



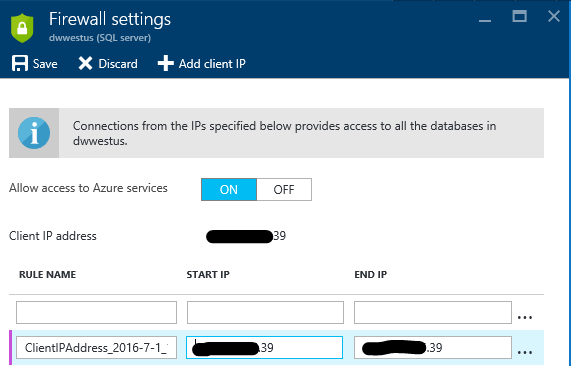
1. Click ‘**Configure your firewall**’ button to allow your client connection.



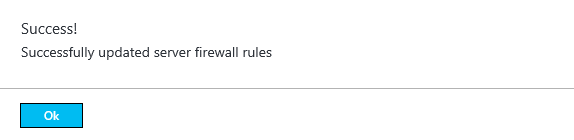
1. Click ‘Add client IP’ button.



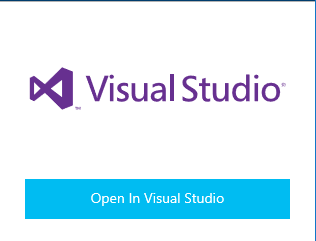
1. Add your client IP address and click **Save** to close.



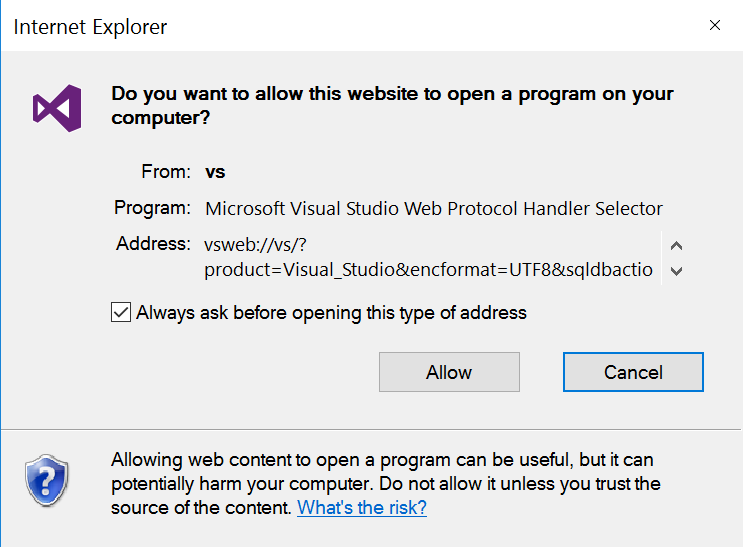
1. You will see the firewall rule is successfully updated.



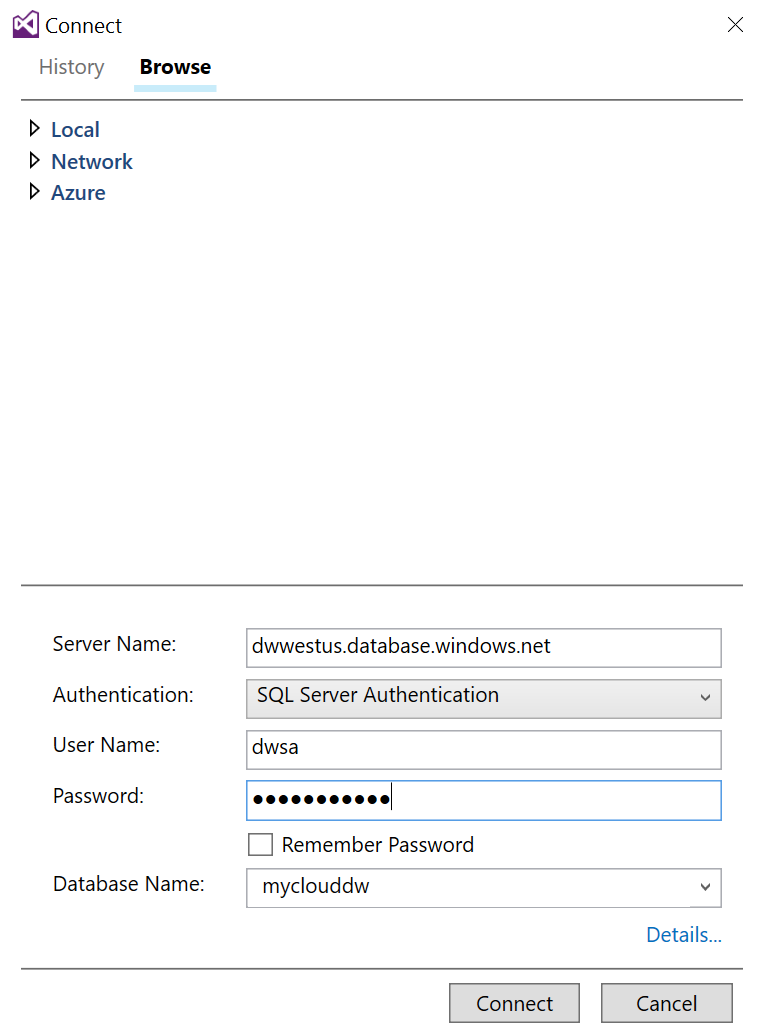
1. Click the new ‘Open in Visual Studio’ button on the new blade that has appeared.



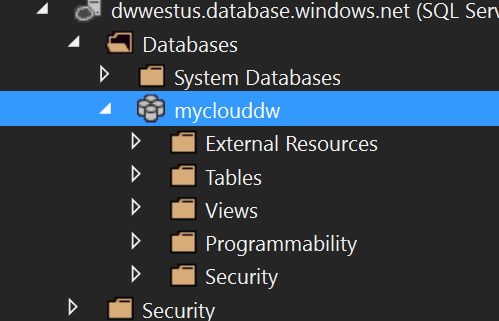
1. Click ‘Allow’ on the dialog box that appears. Visual Studio will now launch.



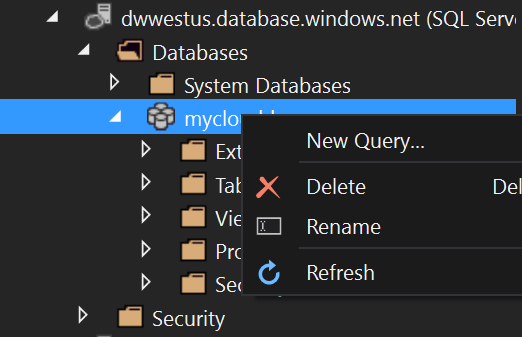
1. Fill in the password specified when the data warehouse was created.



1. Click ‘Connect’
2. Expand out the object tree within the SQL Server Object Explorer pane.



1. Right click the database name and select ‘New Query’. A new query window will open.

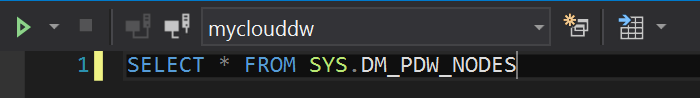


1. Ensure that the database in the drop down displays the name of your new data warehouse. If not, change it to the correct context be-fore continuing.



1. Type the following into the query window

|  |
| --- |
| SELECT \* FROM SYS.DM\_PDW\_NODES |

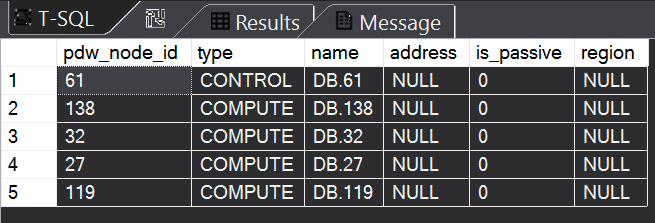


This query will show us how many compute nodes we are using. For more information on scaling and compute resources please see the following URL:

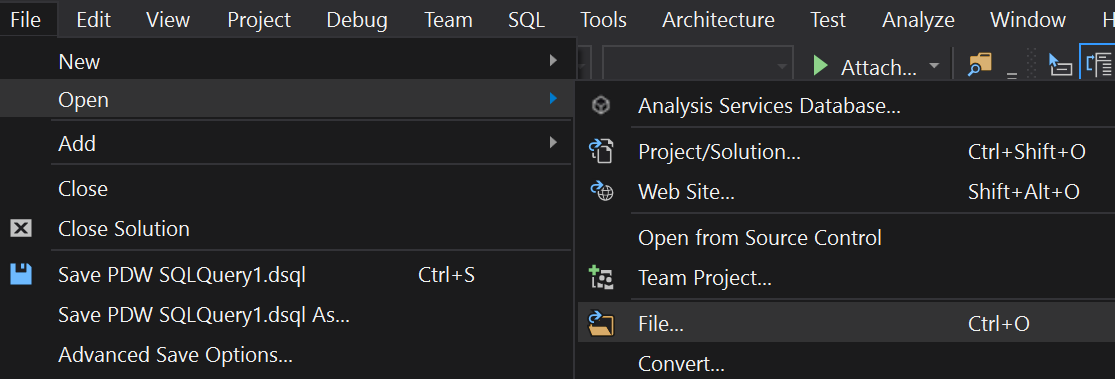
<https://azure.microsoft.com/en-gb/documentation/articles/sql-data-warehouse-develop-concurrency/>

1. The result should be as per the screenshot.

DWU 400 = 1 Control node & 4 Compute Nodes



1. Open the script **Documents\xxx\Script.dsql** by selecting File, Open, File….
2. Browse to the above path and document



1. Select the file and click ‘Open’
2. Click the ‘Connect’ icon on the toolbar above the query window



1. Fill in the details for your server:

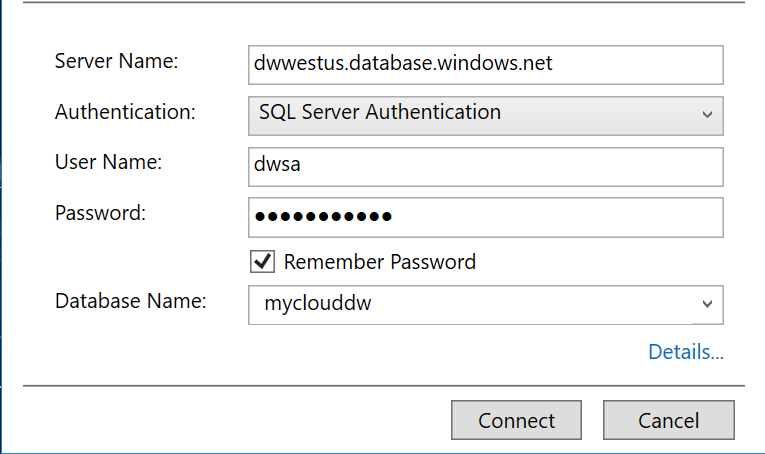
Server Name: dwname.database.windows.net

Authentication = SQL Server Authentication

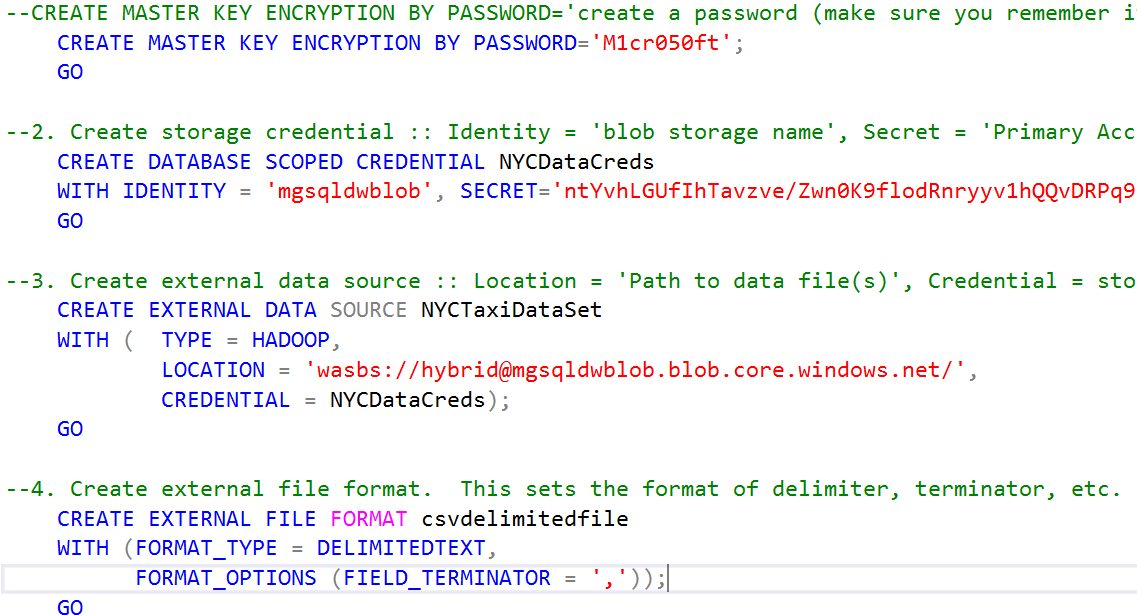
User Name = login name

Password = password

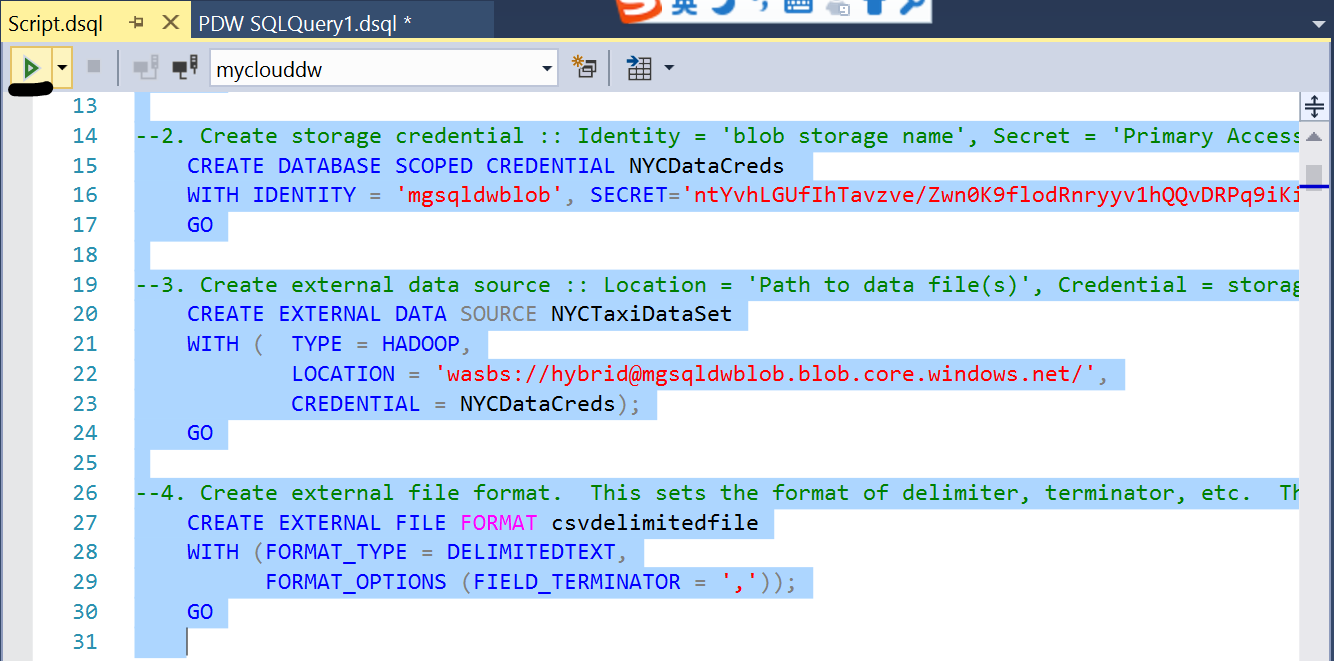
Database Name = use the drop down to select the name of your data warehouse.



1. Click ‘Connect’
2. Review the script by scrolling through it. Do not execute the script at this time.



1. Highlight the first sections (1, 2, 3 & 4) and execute using keystroke combination ‘CTRL-SHIFT-E’ or by using the green arrow at the top of the query window.

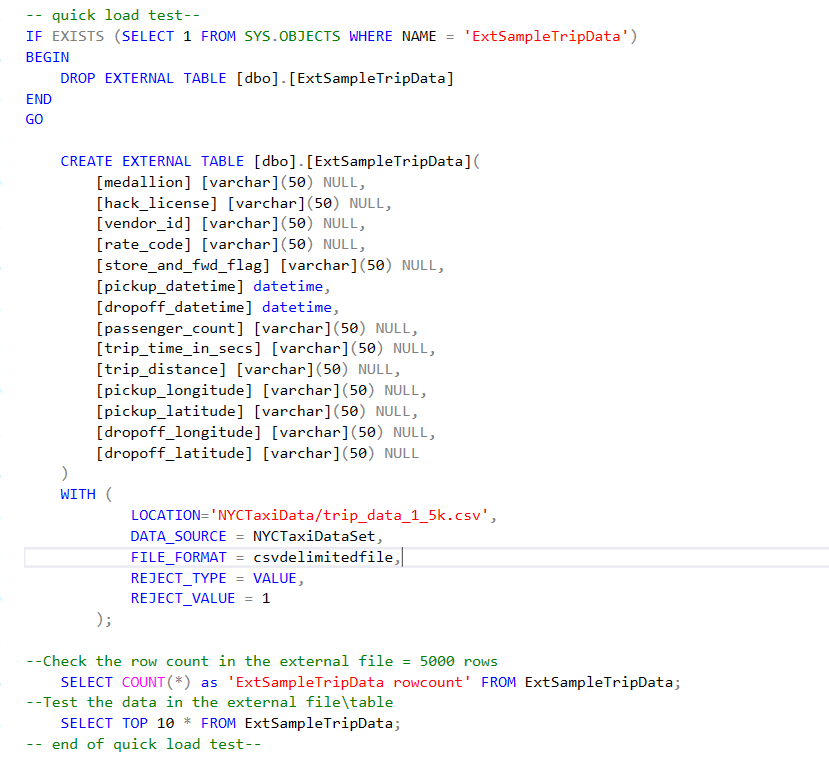


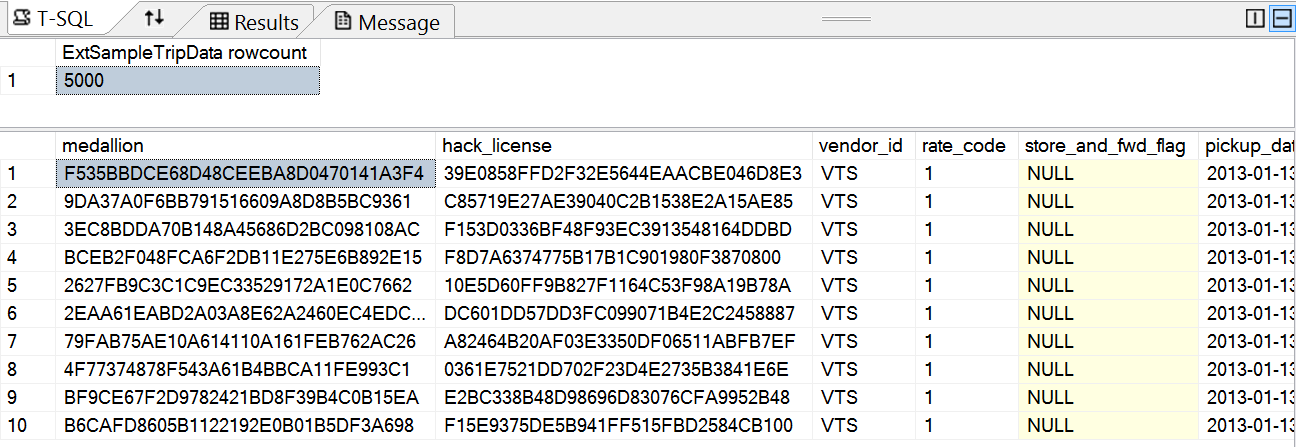
Now we have created the storage credentials we can query and load the data we have in the Azure Blob Store.

Firstly, lets test our connectivity to the data store where we have sample data which we can test with.

The sample data file has 5000 rows.

1. Execute the quick load test section of the script





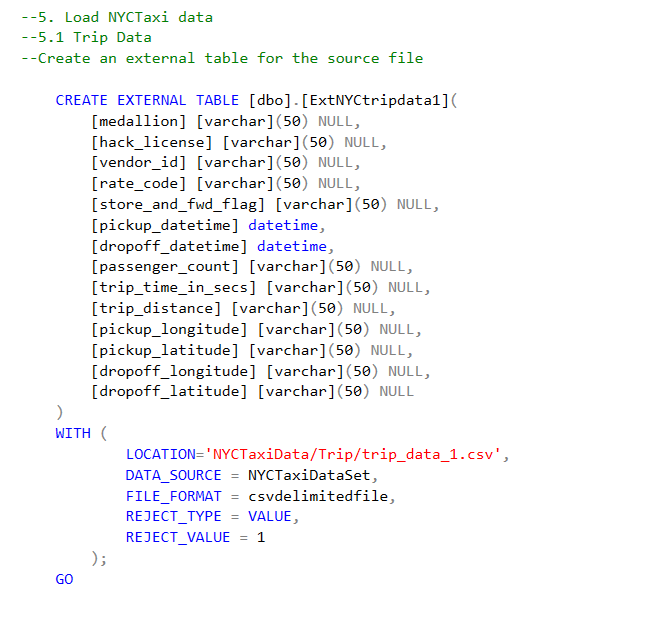
Now the query has completed and we can see the rowcount of 5000 plus a TOP 10 output from the text file in Azure Blob Storage.

Good, so we know we can read the source data. Now for loading something a little larger.

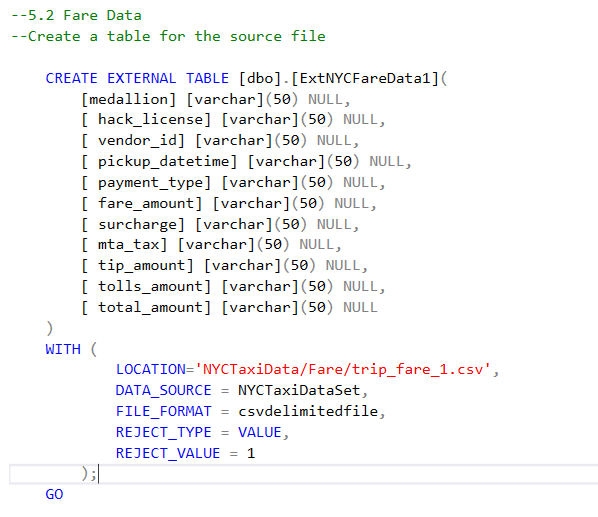
Due to the size of the source data (29GB) we will not load the entire data set. The data set has two primary file sources; tripdata and faredata, as well as a sample data set. We will load the sample data set and one of each source type.

First we will declare the external table metadata for the tripdata file.

1. Highlight and execute section 5.1 of the script.



1. Now we will do the same for the faredata file. Highlight and execute section 5.2 of the script.



Now we will read the data from the file(s) and write the data into a relational table.

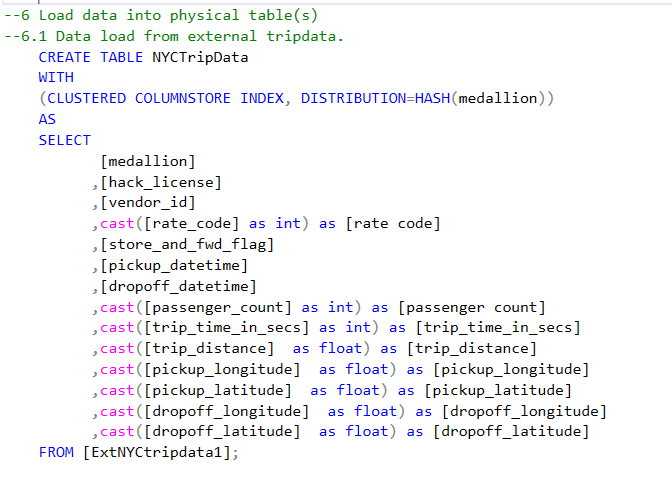
First, we will do the tripdata data.

We read from our external data using our declared EXTER-NAL TABLE (ExtNYCTripdata1).

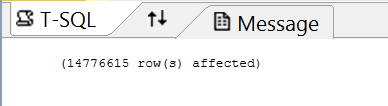
We will insert the data into a physical table using CTAS (CREATE TABLE AS SELECT).

The tripdata file contains 14,776,615 rows and will load in approximately 90 seconds.

1. Highlight and execute section 6.1 of the script.



1. Upon completion we can see how many rows were loaded. 36. Check the output matches the screenshot (rowcount)



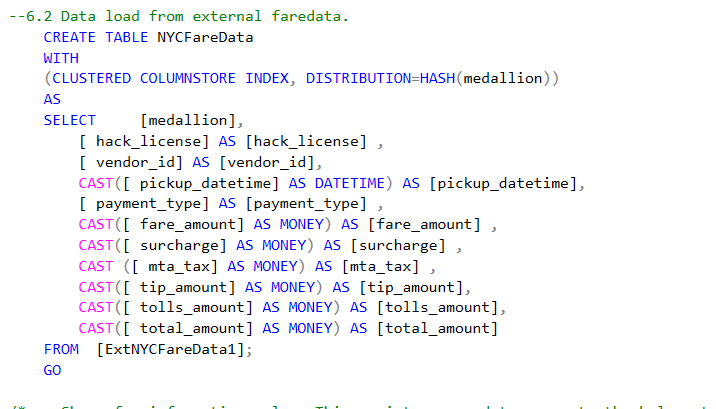
Next, we will do the faredata data.

Similarly, we read from our external data using our de-clared EXTERNAL TABLE (ExtNYCFaredata1).

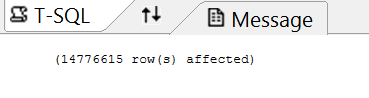
We will insert the data into a physical table using CTAS (CREATE TABLE AS SELECT).

The faredata file contains 14,776,615 rows and will load in approximately 90 seconds.

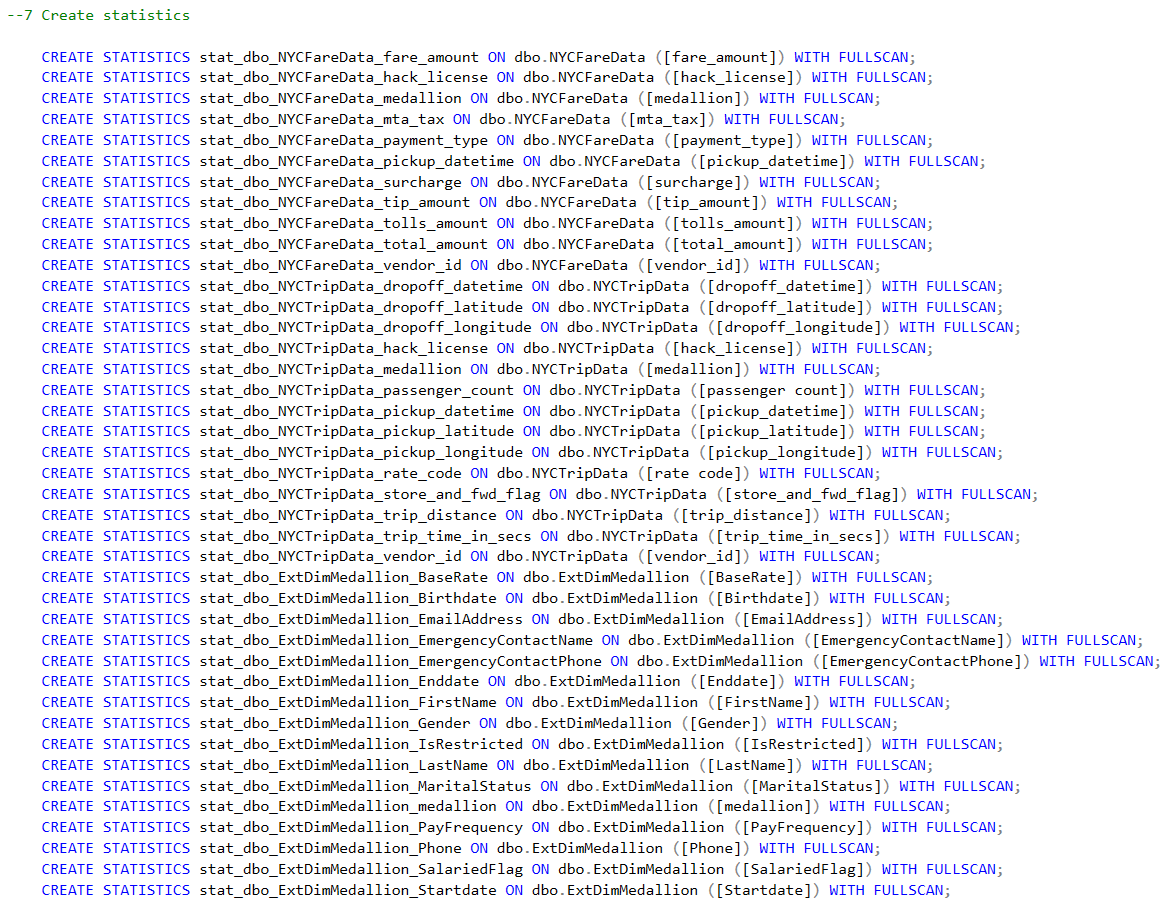
1. Highlight and execute section 6.2 of the script.



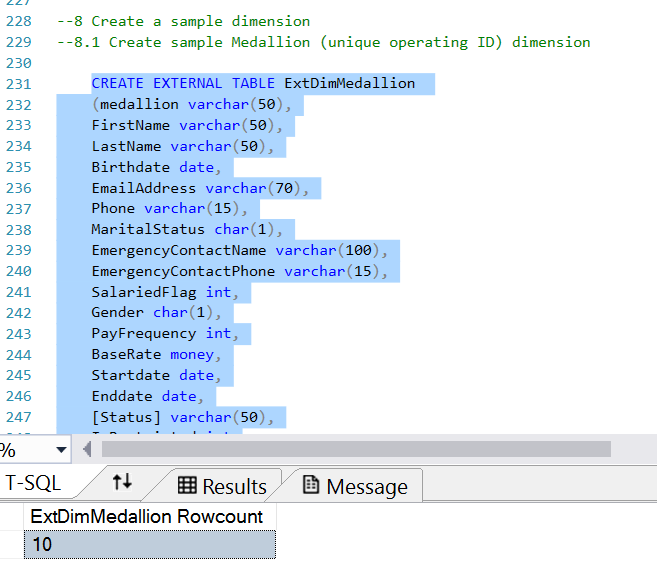
1. Upon completion we can see how many rows were loaded. 36. Check the output matches the screenshot (rowcount)



1. Use the script shown to create the statistics statements.



1. Execute Step 8 in the script (Create sample medallion dimension



Statistics enable the Cost Based Optimizer in knowing where the data exists (extent \ page), when it was last queried, generating the most optimal execution plan, etc.

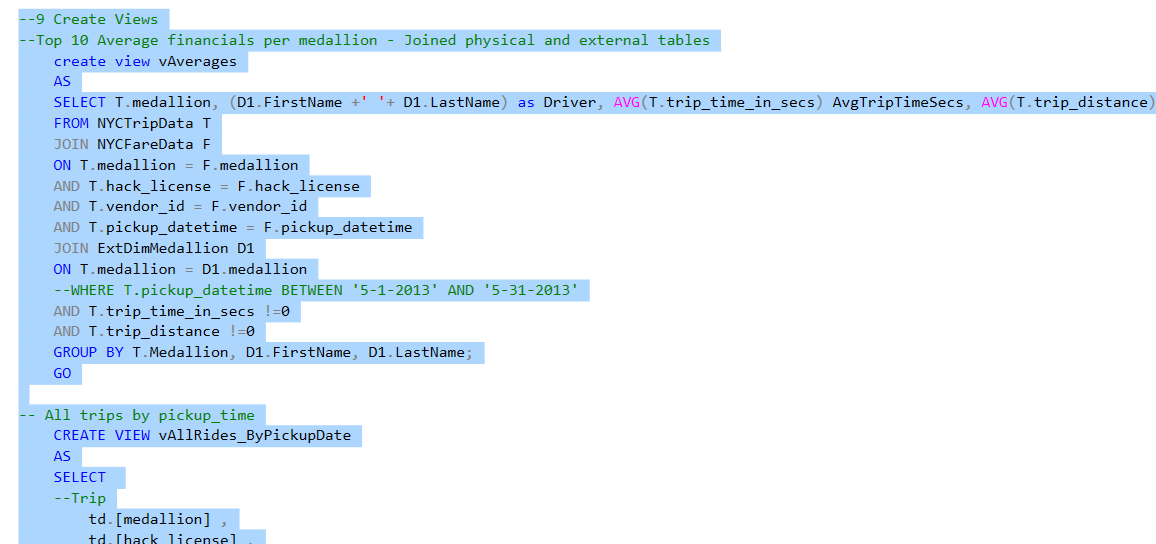
By executing the CREATE STATISTICS statements against all columns in the physical tables (an approach known as ‘covering statistics’) we enable support for faster data access across all columns.

The statistics script (all statements executed as a single batch) should complete in approximately 50 seconds.

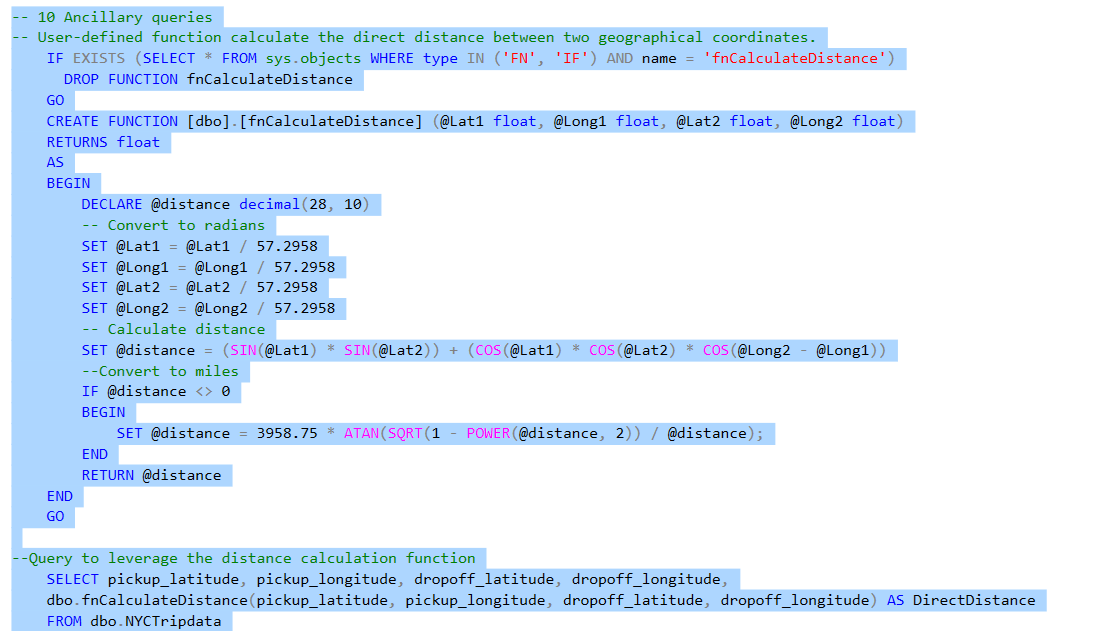
1. Execute Step 7 in the script (the CREATE STATISTICS statements) as a single query batch. Select the statements as a single body of text and use keystroke CTRL-SHIFT-E or press the green arrow on the top pf the query window.

Ok, so now we have data loaded and ready to query. Let’s execute step 9 of the script which creates some useful views over the data.

1. Execute step 9 on the script (Create views). Highlight all of step 9 and execute using CTRL-SHIFT-E or by pressing the green arrow on the query window.



1. Execute step 10 of the script (Ancillary queries). It is recommended that you execute each query individually in order to understand the query and the output produced.



You have just …. created an Azure SQL Data Warehouse, configured access to a cloud data store, queried external data, loaded external data using Polybase, created statistics, prepared data for querying, created supporting data views, created a user defined function, and executed some queries against the data. You have just achieved in less than an hour what enterprises have typically taken weeks or months to achieve.

**Step by step:**

<https://azure.microsoft.com/en-us/documentation/articles/sql-data-warehouse-load-polybase-guide/>

<https://azure.microsoft.com/en-us/documentation/articles/sql-data-warehouse-get-started-load-with-polybase/>

### Exercise 3: Azure SQL Data Warehouse integrated with PowerBI

1. ***Pre-requisites***

A Power BI subscription is required (free or premium)

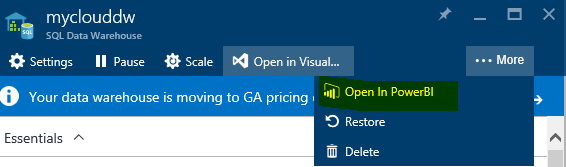
1. We will use the **Microsoft Edge** browser for this lab part.
2. 1. **Browser** (Separate private Mode Browser tabs)
3. Home Page <http://portal.azure.com>
4. Username: **<your live account\azure login>**
5. Password: **<your password>**
6. Home Page http://www.powerbi.com
7. Username: [??????@**yourcompany.com**](mailto:??????@yourcompany.com)
8. Password: ??????
9. ***Scenario***

Storing data at scale is one thing – but what we really want to do is get insight from that data. With PowerBI we can gain insight into any data to make informed business decisions efficiently and safely.

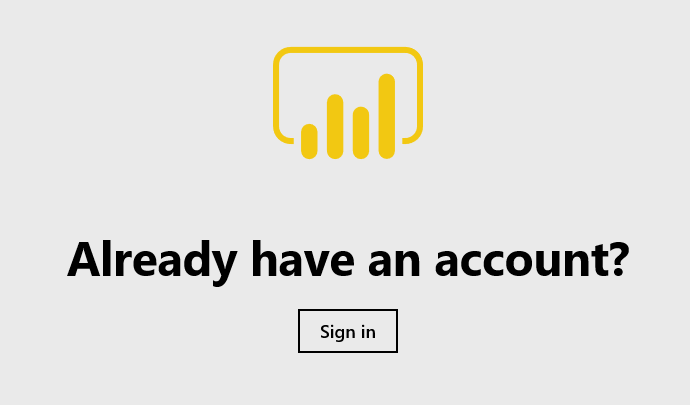
1. ***Steps***

Let’s see how easy it is to hook up the Azure SQL datawarehouse to PowerBI

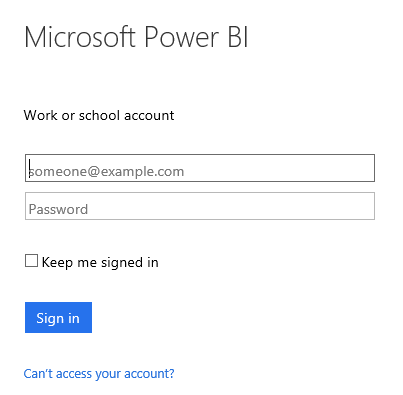
1. In the Azure portal, click the data warehouse tile that you pinned to your dashboard. The blade will open and you will see your data warehouse properties.
2. Click on ‘Open in Power BI’ button at the top of the blade to navigate to Power BI.



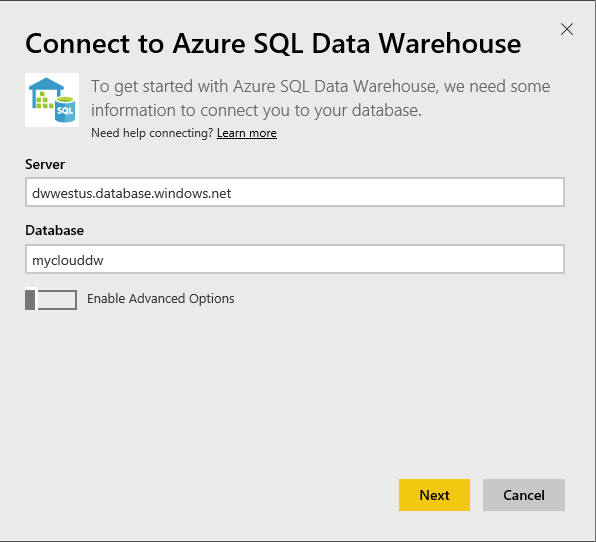
1. Log into Power BI (if required). Click the Sign In button on PowerBi.com.



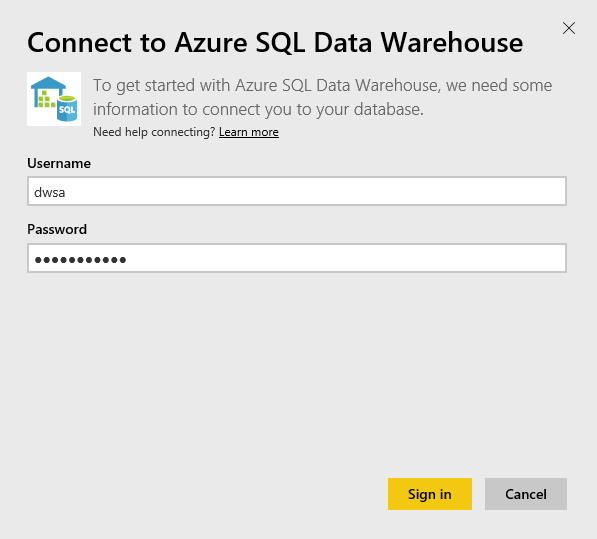
1. Provide your credentials when prompted.



1. Power BI should open to the connection page

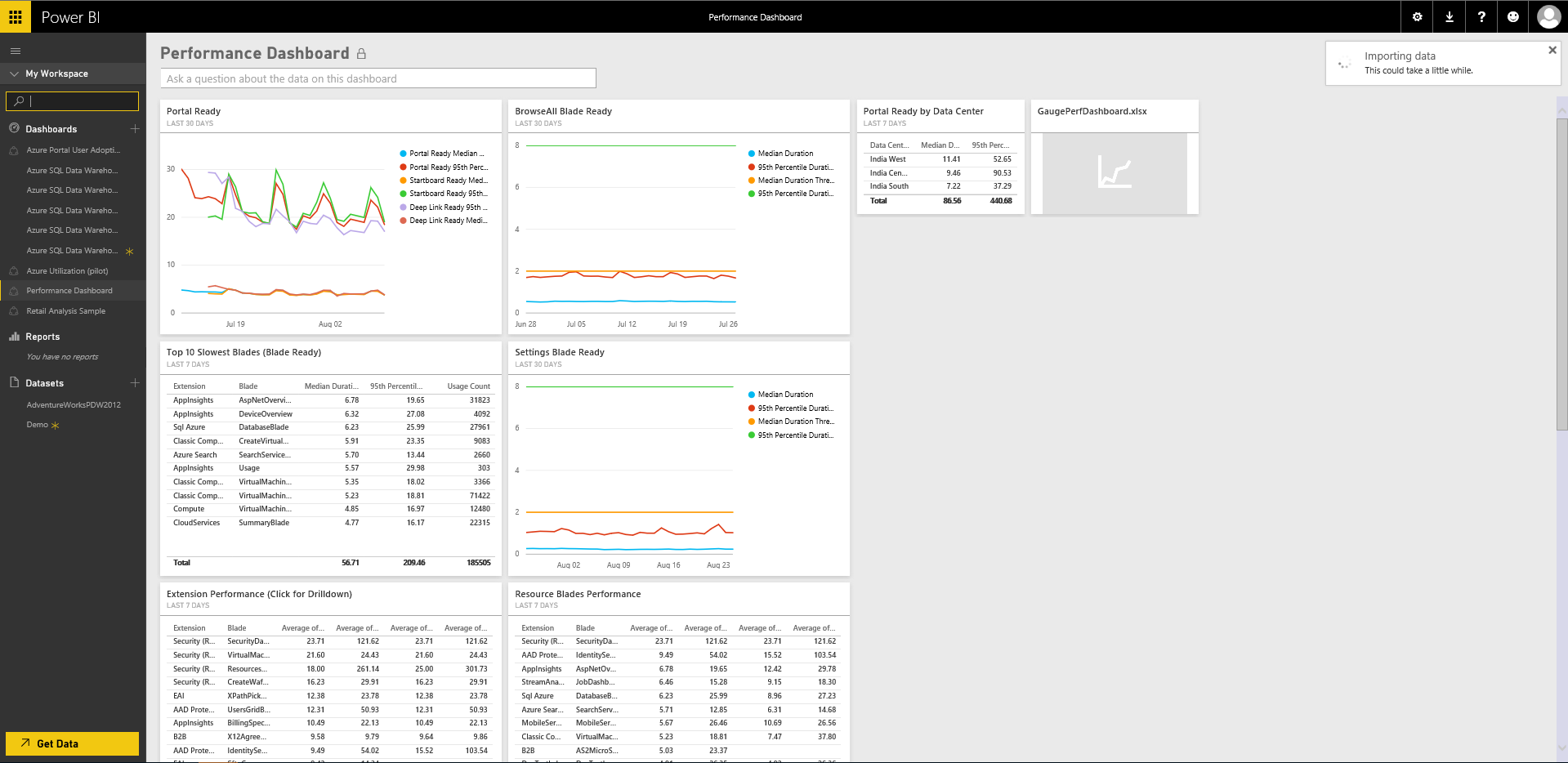


1. Click Next
2. Next page has login filled in except for password. Fill in the password for your account and then click Connect.



NOTE: If the Connect dialog does not display the information, return to the Azure portal and click the “Open in Power BI” link again.

1. After clicking connect Power BI will redirect to the main page and a connection will be established. The loading icon may be shown in the upper right corner spot. Once completed you should be presented with a clean report sheet (blank sheet of ‘digital’ paper).



If the sheet didn’t show for some reason (maybe it was the first time logging into Power BI), look for a new dataset (located on the lower left hand side of the workspace under Datasets) which will be called myclouddw. Double click on the dataset which will open a blank report.

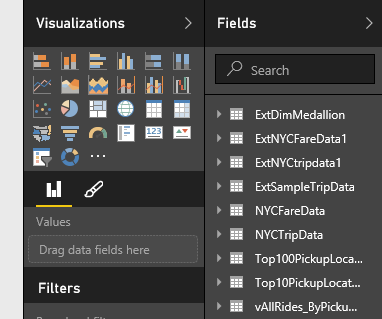
Continue.



We can see the tables\views and respective fields available on the right hand side of the Power BI screen.

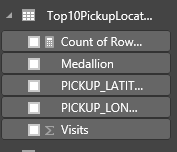
We will use one of our views we created previously to create a stunning BI report over the data we just loaded into our data warehouse.

1. Familiarize yourself with the layout pf the Power BI screen. Don’t click or create anything yet.



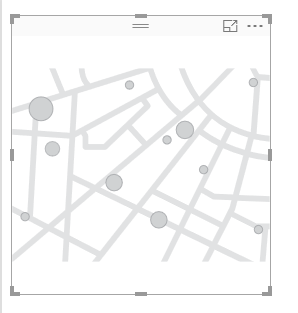
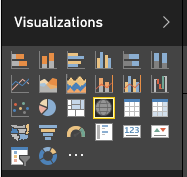
To create appealing and insightful visualizations we can simply click on a visual which will add it to the report, or we can drag data fields onto the report page directly from the field list.

1. Navigate down the Fields list to the vTop10PickupLocations view and expand it so you can see the fields.



1. We will create a map visual using the table Top10PickupLocations, which is the last object in the field list.

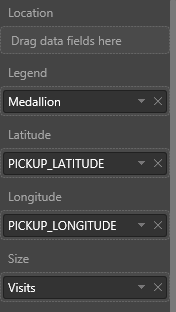
Select the Map visualization from the Visualizations box. An empty map pane will appear on the report sheet.



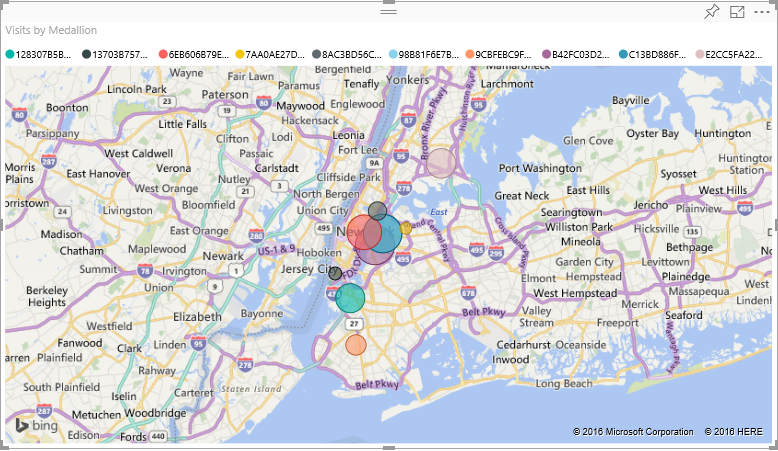
1. If we click on the map object, we see fields appear under the Visualizations box. We need to place data fields into the visualization fields in order to create the relationship.

Drag the following data fields onto the respective visualization fields:

* PICKUP\_LATITUDE -> Latitude 
* PICKUP\_LONGITUDE -> Longitude 
* Visits -> Size 
* Medallion -> Legend



1. Resize the visual to suit your preference. Resizing can be easily done by simply dragging\expanding the borders of the visual.



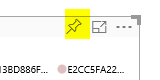
1. Click Save on the menu bar at the top of the report page.



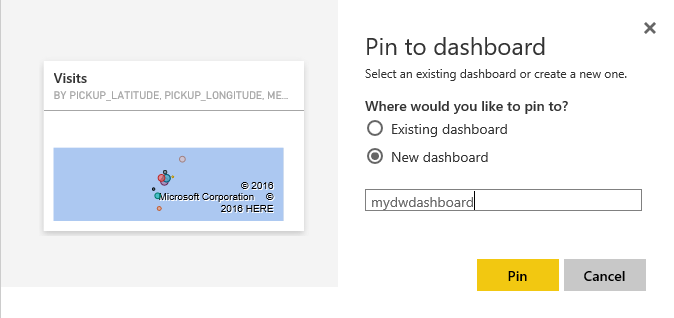
1. Enter the report name and save it.

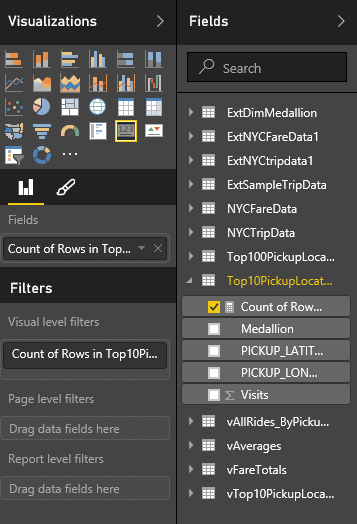
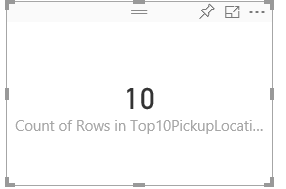


1. To pin the tiles to your dashboard. Click Pin button.

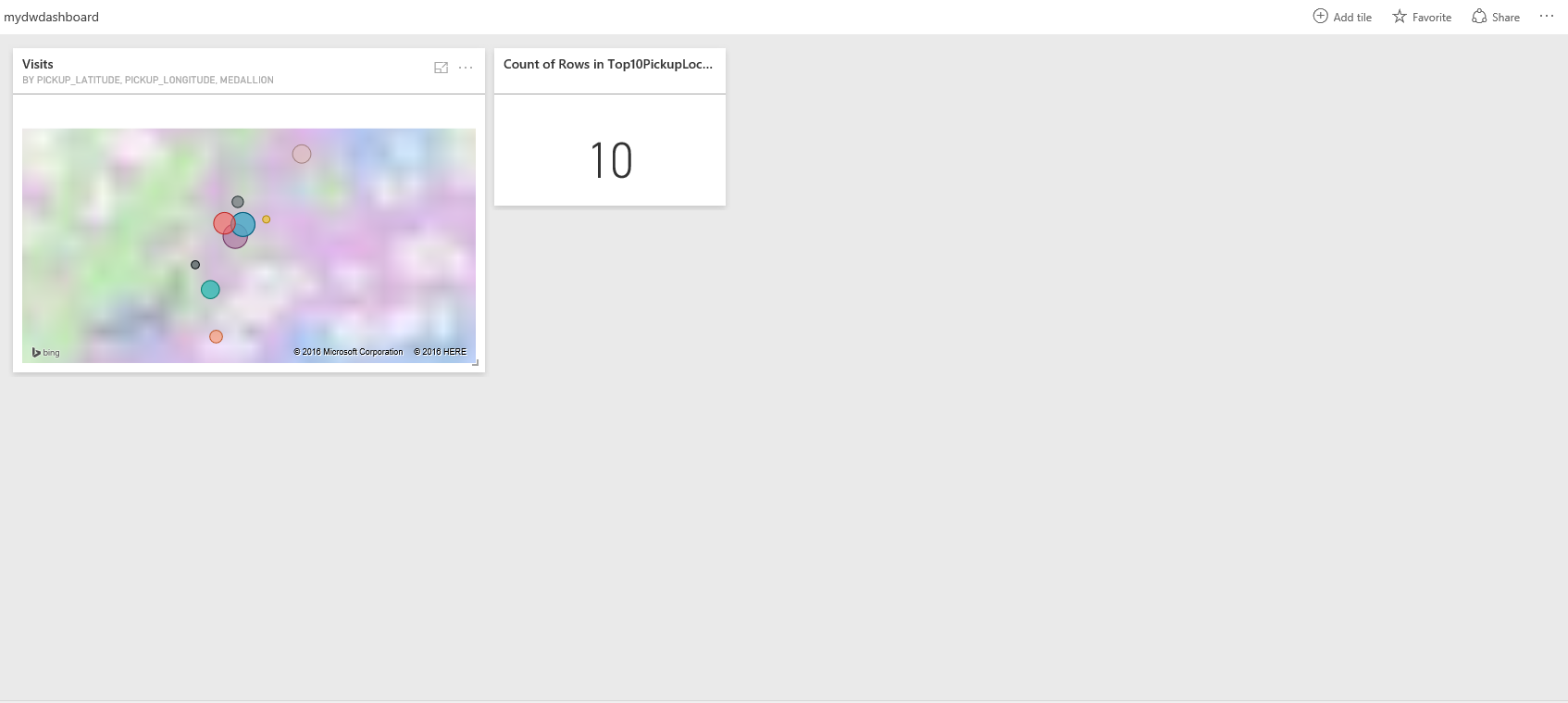


1. Create a new dashboard and click Pin.



1. You can create another card chart to show the Top 10 Pickup location. Then pin it to the dashboard.
2. 
3. You will see the new item with star icon under Dashboard. Click it to see the dashboard.





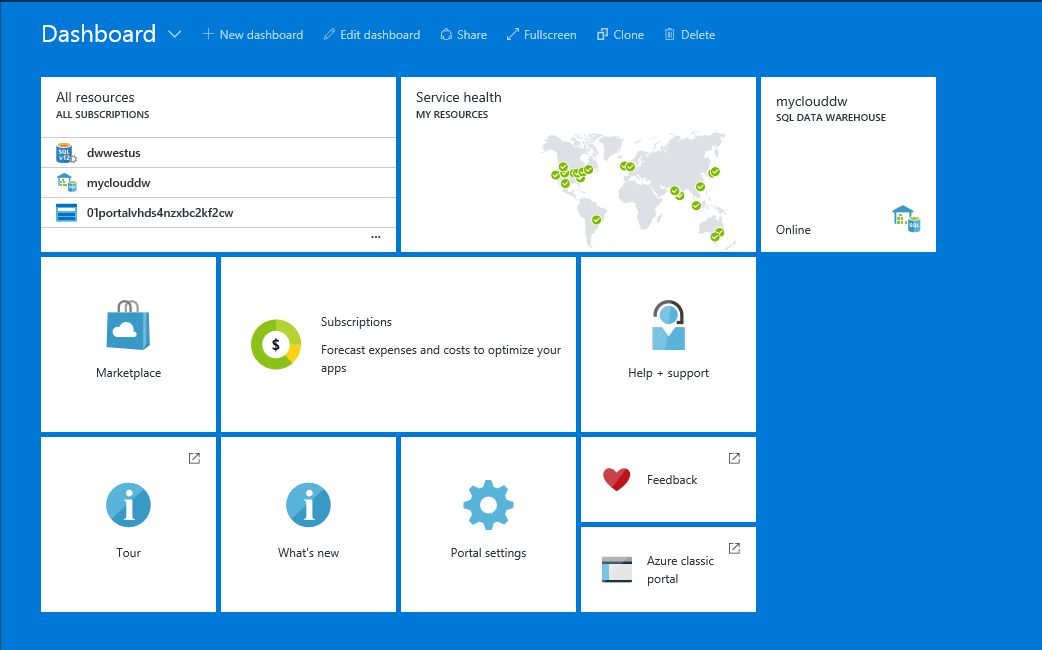
**IMPORTANT – PLEASE READ**

If you are using your own Azure subscription, please perform the following steps to pause your Azure SQL Data Warehouse. It is highly recommended that you Pause or delete your data warehouse at the end of the lab to avoid any compute charges when the data warehouse is not in use.

To pause our data warehouse, we will use the Azure portal.

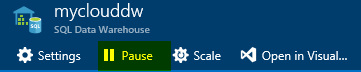
We must open our data warehouse blade because we want to manipulate the data warehouse, specifically the service state.

1. Within the Azure portal, click on the data warehouse tile you pinned to the dashboard when you created the data warehouse.

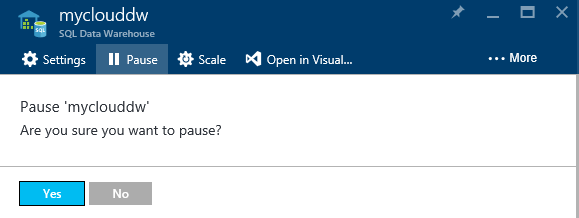


1. Using the integrated management experience that Azure offers we can quickly pause, resume or scale our data warehouse. We will pause our data warehouse.

Click the **Pause** button on the blade toolbar



1. Azure provides a confirmation prompt to pause the service. We should accept the confirmation.



1. Click Yes on the Pause dialogue prompt.
2. The data warehouse will now pause meaning that you will not be charged for any compute after the operation has completed.

**Step by step:**

<https://azure.microsoft.com/en-us/documentation/articles/sql-data-warehouse-integrate-power-bi/>

* 1. **Appendix A: Structure**
  2. **DB Structure**
  3. Custom schema (NYC Taxi dataset)
  4. **West US Data Centre**:
* Resource Group = rgmyclouddw
* SQL DW Server = dwwestus.database.windows.net
* Database = myclouddw