MRS / Spark / HDInsight / AML Blob Storage - Customer Churn Prediction Demo

Instructions



Author: Min Qiu, Katherine Zhao

Date: May 2016

# Introduction

* **Architecture**
* **Prerequisites**

# Demo Setup Instructions

# Demo Instructions

# Demo Workthrough

* **Step 1: Upload sample data and reference using HADOOP**
* **Step 2: Create hive tables using HIVE**
* **Step 3: Partition sample data using HIVE**
* **Step 4: Feature Engineering Train data using SPARK**
* **Step 5: Train and Persist Model using MRS with SPARK**
* **Step 6: Feature Engineering all data using SPARK**
* **Step 7: Score Model using MRS with SPARK**
* **Step 8: Deploy AML web service for lookup using MRS**
* **Step 9: Joseph’s Mart to using AML web service**

# Visualization

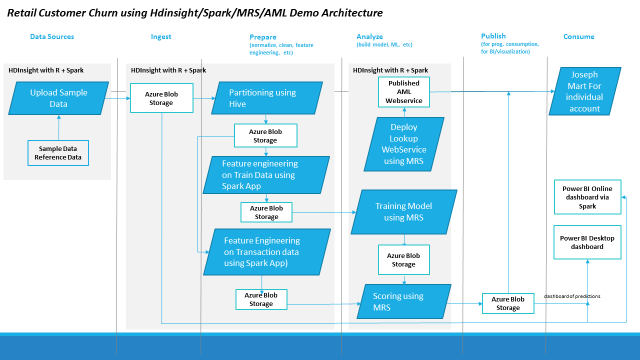
* **Visualization using POWER BI desktop**
* **Visualization using POWER BI online with SPARK**

Cleanup

# INTRODUCTION

This demo demonstrates how to use [Microsoft R Server](https://azure.microsoft.com/en-us/documentation/articles/hdinsight-hadoop-r-server-install-r-studio/), [Azure HDInsight with R on Linux](https://azure.microsoft.com/en-us/documentation/articles/hdinsight-hadoop-r-server-get-started/), [Azure Machine Learning](https://azure.microsoft.com/en-us/services/machine-learning/), Spark, Scala, Hive, etc. to build an end-to-end, cloud solution for Retail Customer Churn. The demo attempts to simulate the real-world use case of data placement/storage, feature engineering, model retraining, prediction, and visualization.

## Architecture



## Prerequisites

### **Subscription requirements.**

An Azure subscription: Before you begin, you must have an Azure subscription that have access to Azure HDInsight, Azure Blob Storage, etc. See [Get Azure Free Trail](https://azure.microsoft.com/en-us/documentation/videos/get-azure-free-trial-for-testing-hadoop-in-hdinsight/) for more information.

### **HDInsight Cluster on Linux with R Server**.

First, you need a HDInsight Cluster on Linux with R Server to deploy in order to run this demo.

* + Deploy an HDInsight cluster running Linux with R Server (with Azure Storage) with an SSH password. You can find the instruction [here](https://azure.microsoft.com/en-us/documentation/articles/hdinsight-hadoop-r-server-get-started/#create-the-cluster).
  + Follow all the steps the “Create the Cluster” section. Followings are some recommendations from us.
  + In Step 5, we recommend you creating a new resource group for the demo.
  + In Step 7, we recommend you creating a new storage account for the demo. Also, DO NOT name the default container as “customerchurnintern” or “customerchurnresult”.
  + In Step 8, it’s better to change “Number of Worker nodes” to 8 and remain all D4 nodes as default.

The deployment of the Cluster takes about 20-40 minutes. In the meanwhile, you can continue with the remaining prerequisites setup.

### **Power BI.**

You need a Power BI account, an Online access and a Desktop software for this demo.

* + **Power BI Account: could be gotten from**[here](https://powerbi.microsoft.com/en-us/landing/signin/)
  + **Online: access Power BI online** [here](https://powerbi.microsoft.com/)
  + **Desktop:** download and install Power BI Desktop[here](https://powerbi.microsoft.com/en-us/desktop/)**.**

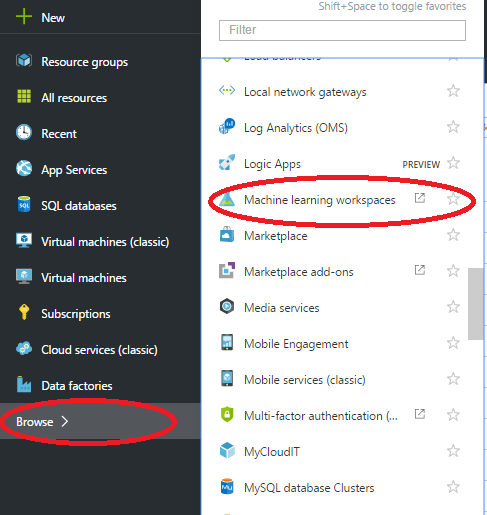
### **Windows side requirements.**

* + A Secure Shell (SSH) client: An SSH client is used to remotely connect to the HDInsight cluster and run commands directly on the cluster. Linux, Unix, and OS X systems provide an SSH client through the SSH command. For Windows systems, we recommend [PuTTY](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html).
  + (OPTIONAL) A SCP client: For Windows, we recommend [WinSCP](https://winscp.net/eng/download.php).
  + Visual Studio with Azure SDK. Please install Visual Studio and Azure SDK by following the step 1 in this instruction [here](https://blogs.msdn.microsoft.com/xiaoyong/2015/05/04/how-to-write-and-submit-hive-queries-using-visual-studio/).

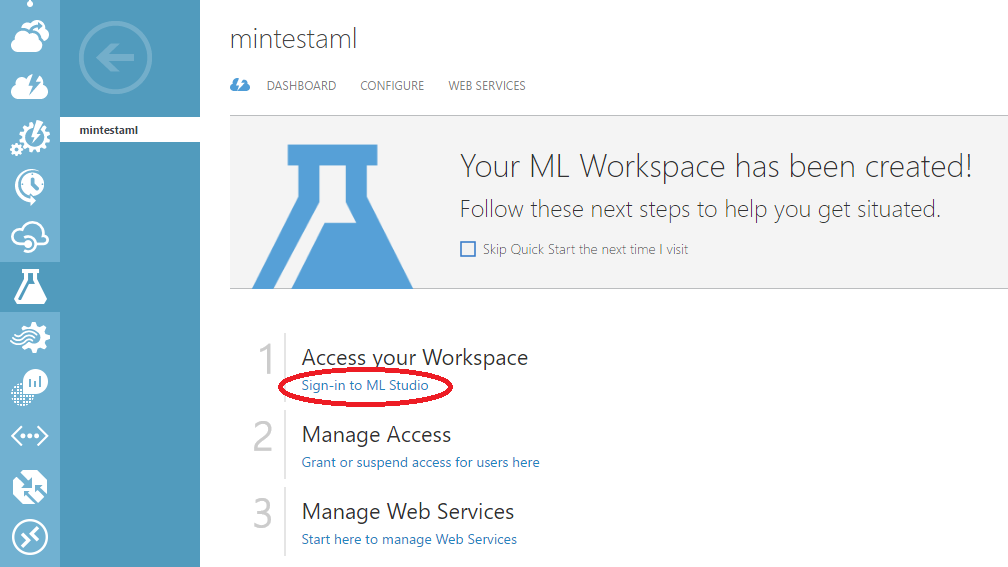
### **Azure Machine Learning Workspace.**

You need a AML workspace pre-created.

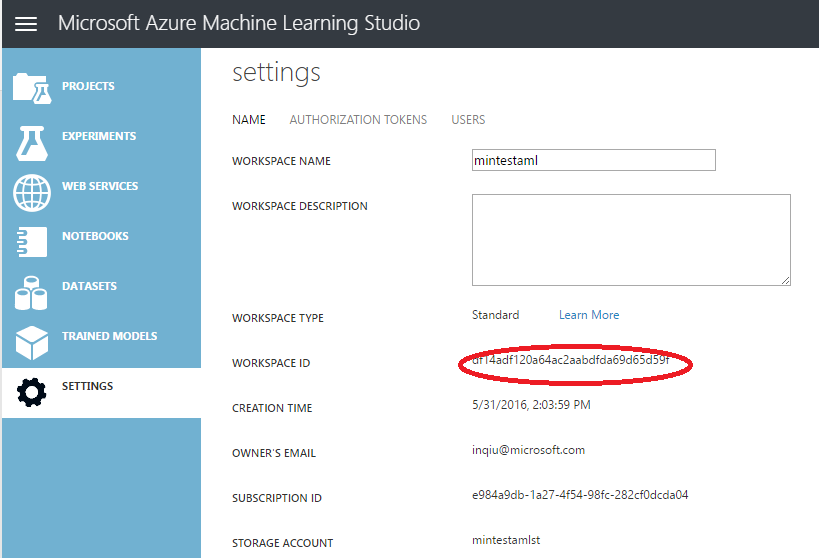
* + Go to portal.azure.com. Choose the Machine Learning Workspace like below which will open the old azure portal.

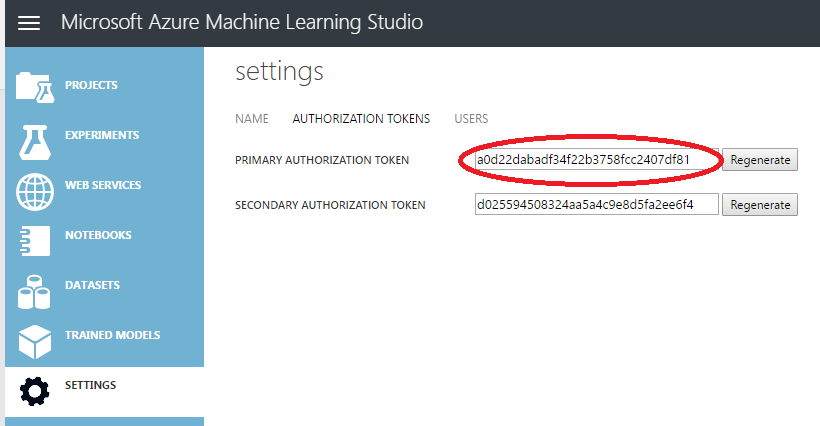


* + Following the instruction [here](https://azure.microsoft.com/en-us/documentation/articles/machine-learning-create-workspace/) to create a new workspace if you don’t have one. Please note, you can create a new storage account in the step 5.
  + Find out the Workspace ID and Authorization token as followings.
    - Open **Azure Machine Learning** through Portal. It will open the Azure Machine Learning portal.
    - Select the workspace you want to use. And **Sign-in to ML Studio.**



* + - Go to SETTINGS in the AML Studio. Find the **WORKSPACE ID** under the NAME tab and find **PRIMARY AUTHORIZATION TOKEN** under the **AUTHORIZATION TOKENS** tab. This information will be needed later to for publishing Azure Machine Learning web service in Microsoft R Server.





# DEMO SETUP INSTRUCTIONS

Note in the section all *instructions* are shown in blue italics. Talking points are shown in black normal text.

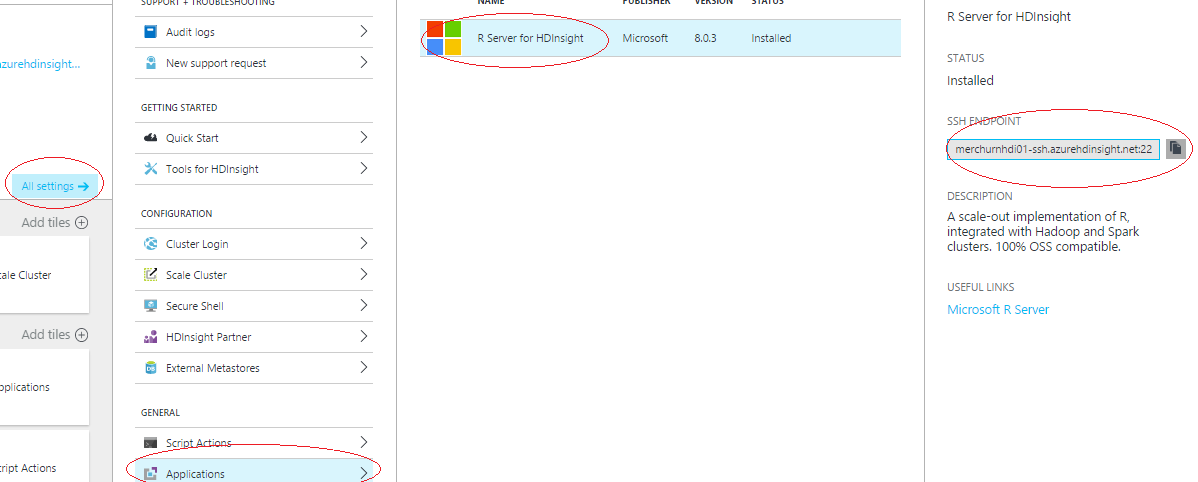
Now we explain how to connect to the edge node of an HDInsight cluster with R Server, install R Studio Server, install required packages and libraries for R, AzureML and Spark etc.

## *Identify the edge node of the cluster.*

For an HDInsight cluster with R Server, following is the naming convention for the edge node.

* + Edge node – R-Server.CLUSTERNAME-ssh.azurehdinsight.net

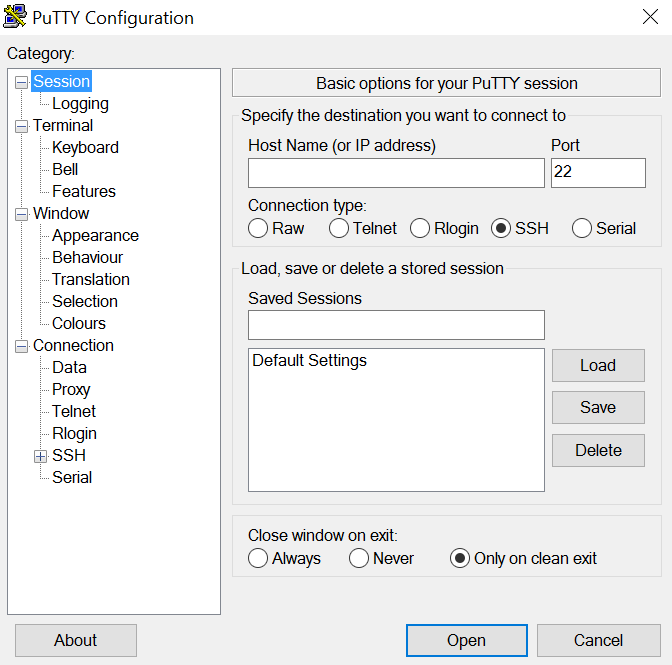
You can also find the R-Server.CLUSTERNAME-ssh.azurehdinsight.net address in the Azure portal by selecting your cluster, then **All Settings**, **Applications**, and **RServer**. This will display the SSH Endpoint information for the edge node. If you click on the copy button to get the address, you need to remove the port number “:22” from the end.



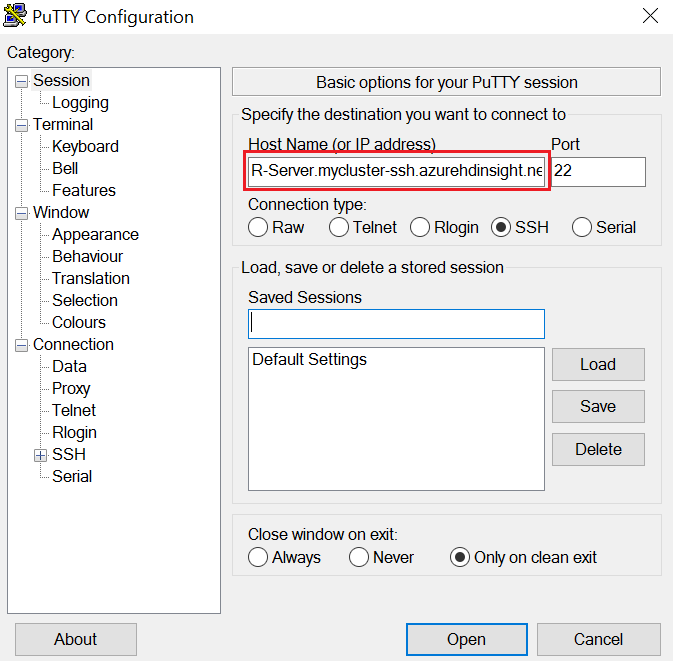
## *Connect to the R Server edge node which is on Linux using PuTTY.*

See more details [Connect to a Linux-based HDInsight cluster using PuTTY](https://azure.microsoft.com/en-us/documentation/articles/hdinsight-hadoop-linux-use-ssh-windows/#connect-to-a-linux-based-hdinsight-cluster).

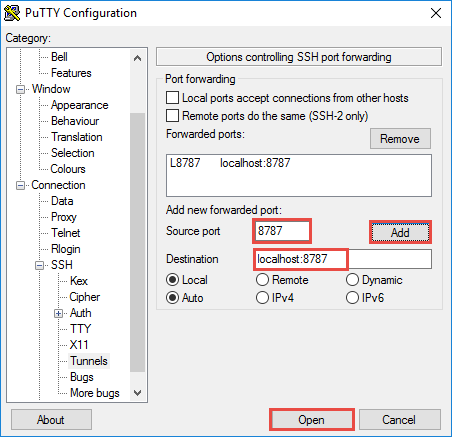
* + *Open PuTTY.*



* + *In Category, select Session. From the Basic options for your PuTTY session screen, enter the SSH Endpoint (edge node address, for example, “R-Server.mycluster-ssh.azurehdinsight.net”) of your HDInsight server in the Host name (or IP address) field.*



* + *In order to run R Server script on a R Studio Server and open the JosephMart website, you need to create two SSH tunnels in the current PuTTY session.* 
    1. *In the****Category****section to the left of the dialog, expand****Connection****, expand****SSH****, and then select****Tunnels****.*
    2. *Provide the following information on the****Options controlling SSH port forwarding*** *form for R Studio:*
       1. **Source port** - The port on the client that you wish to forward. For example, **8787**.
       2. **Destination** - The destination that must be mapped to the local client machine. For example, **localhost:8787**.



* + - 1. *Click****Add****to add the settings.*
    1. *Provide the following information on the****Options controlling SSH port forwarding*** *form for JosephMart website in a similar way like in II:*

1. **Source port** - The port on the client that you wish to forward. Input **3000**.
2. **Destination** - The destination that must be mapped to the local client machine. Input **localhost:3000**.
3. *Click****Add****to add the settings.*
   * *To save the connection information for future use, enter a name for this connection under* ***Saved Sessions****, and then click* ***Save****. The connection will be added to the list of saved sessions*.
   * *Click* ***Open*** *to connect to the cluster. When prompted, you should use the username and password for ssh when you entered during the cluster deployment.*

## *Upload the zip file to the edge node using any ftp client, for example, WinSCP.*

* + Connect to edge node via SSH client using the same credentials of the Edge Node. In WinSCP, you can import the PuTTY configurations by clicking **Tools -> Import Sites -> Choose a PuTTY site to import -> Edit** and enter the SSH user name and password to connect.
  + Download the file “CumstomerChurn\_MRS\_SPARK.zip” and put under the home directory for the user.
* *Unzip the zip file by in PuTTY.*

In the home directory of the connected PuTTY session (same connection as above), type in the below commands to unzip the zip file.

*$ unzip CumstomerChurn\_MRS\_SPARK.zip –d $HOME/customer\_churn\_demo*

*$ cd $HOME/customer\_churn\_demo*

## *Run setup\_demo shell script.*

Under the directory you unzip the files, run followings. This will automatically install all the required elements. You will be asked if you want to install something that would take 21 MB during this installation. You need to enter “Y” in the PuTTY session to continue.

*$ cd $HOME/customer\_churn\_demo*

*$ chmod +x \*.sh*

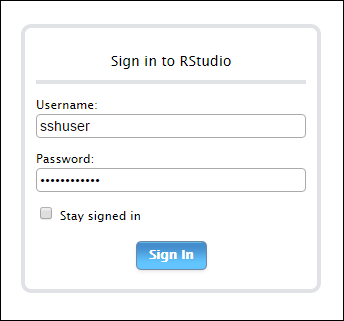
*$ ./setup\_demo.sh*

This installation takes about 10 minutes.

## *Test R Studio Server.*

To test if the R Studio Server successfully installed and R Studio client works, open a web browser and enter the following URL based on the port you entered for the tunnel.

* 1. *Open* [*http://localhost:8787/*](http://localhost:8787/) *in a web browser in your local machine.*
  2. *Enter the SSH username and password to connect to the cluster when prompted.*



Now we are ready to run the demo.

# DEMO INSTRUCTIONS

## PowerPoint

Use the PowerPoint deck to introduce the demo. When you get to the “Demo” slide, follow the script and instructions below.

## Prepare Demo

Make sure to set up your computer prior to the demo following the section “DEMO SETUP INSTRUCTIONS”

* Open a SSH session to the Edge Node of the HDInsight cluster.
* Open <http://localhost:8787/> to run R Studio Client.

Open the following applications as well:

* Go to azure portal -> the HDInsight deployed -> Cluster Dashboard -> Jupyter Notebook and log in using HDInsight’s username and password.
* PowerPoint – open “Customer Churn.pptx” under the directory you unzipped in the Windows machine.
* Visual Studio
  + Open the project CustomerChurnSparkDemo.hiveproj under the directory you unzipped in the Windows machine.
  + Click VIEW > Server Explorer and sign in with your Azure account. Then right click on the “HDInsight” Node, select “Connect to a Hadoop Cluster”.
* Power BI desktop.

## Run Demo

Note in the section all *instructions* are shown in blue italics. Talking points are shown in black normal text.

## Live Demo

We will be showing the integration of R into different Microsoft products in this demo: HDInsight, Hive, Spark, R Studio, Azure Machine Learning and Power BI.

There is only one shell script you need to run the whole demo. The name is run\_demo.sh which is under the directory you unzip the files in the Edge Node. SSH to the Edge Node, Run as in followings. it will automatically run all steps of demo.

*$ chmod +x \*.sh*

*$ ./run\_demo.sh*

It will prompt for the following parameters:

1. **storagename**: This is the existing storage you choose in the **Data Source** or the name of **New** storage when you deploy the Cluster.
2. **HDIcontainer**: This is the container name the HDInsight cluster default associated with. if you don’t change anything during the cluster deployment, it usually uses the default value which is the same as the CLUSTERNAME.
3. **wsID**: This is ID of the Azure Machine Learning Workspace you choose to use
4. **wsAuth**: This is the Authorization token of the Azure Machine Learning Workspace you choose to use
5. **churnPeriod**: This is the period you want to set to define the customer churn. Default value is 21 days.
6. **churnThreshold**: This is the threshold you want to set to define the customer churn. The threshold defines as the number transactions a customer has at the churnPeriod. Default value is 0, which means a customer churned if he/she doesn’t have any transaction during the churnPeriod.

The script will pause on each step so you can check the result for each step like in DEMO WALKTHROUGH below.

# DEMO WALKTHROUGH

As we state above, this demo is a one-command demo. The following tells how to check the results of each step.

Let’s start in Visual Studio where we have connected to HDInsight Cluster.

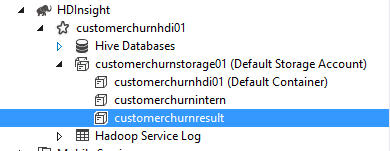
*In Visual Studio, open Server Explorer -> Sign in using your Azure account -> connect to the Azure Subscription -> HDInsight*

*In Visual Studio, also open the project CustomerChurnSparkDemo.hiveproj under the directory you unzipped in the Windows machine.* Here you can see all the code.

## Step 1: Upload sample data and reference data to Azure Blob Storage using HADOOP

First of all, the demo automatically creates two containers customerchurnintern and customerchurnresult in the Azure Blob Storage besides using the default container associated with the HDInsight Cluster.

*In Visual Studio, open Server Explorer -> HDInsight -> {CLUSTERSTORAGENAME} -> customerchurnresult*

. 

customerchurnresult: stores data we need to use for visualization due to the performance consideration.

* /customerchurn/data/sampledata/activity
* /customerchurn/data/sampledata/user
* /customerchurn/data/referencedata/age
* /customerchurn/data/referencedata/region
* /customerchurn/data/predictions

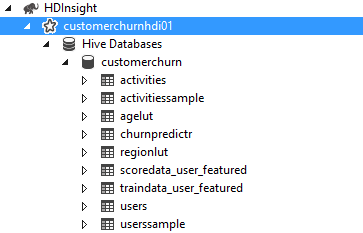
customerchurnintern: stores all the partitioned data and intermediate data.

The sample data for activities and users come with the demo package are respectively loaded to Azure Blob Storage under customerchurn/data/sampledata/activity and customerchurn/data/sampledata/user folders in a container customerchurnresult.

## Step 2: Create hive tables using HIVE

We are using hive tables to interact with the Azure Blob storage. The database for the demo in hive is customerchurn. We create two hive tables activitiessample and userssample on top of the sample data and two lookup tables agelut and regionlut. We also create the tables for the Step 3, 4.

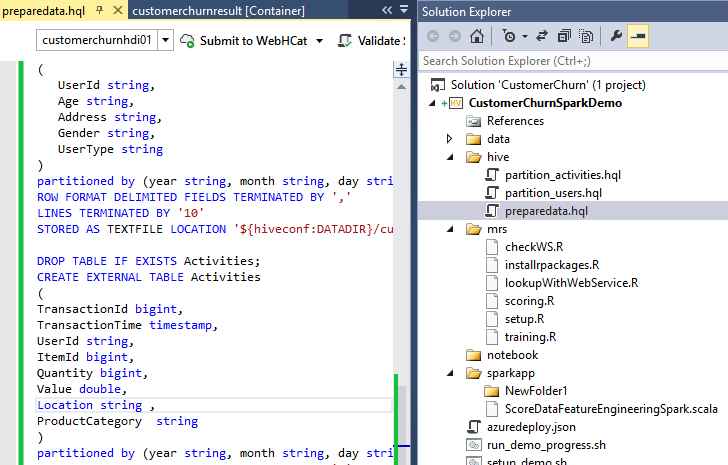
*In Visual Studio, open Server Explorer -> HDInsight -> {CLUSTERMNAME} ->Hive Databases -> customerchurn*



*View columns of tables by click the table to expend*

You can see the columns of each table.

*In Visual Studio, open the file prepareddata.hql under hive folder in the solution explorer to view the hive query:*

**

We will now interact with these tables in Hive and Spark.

## Step 3: Partition sample data using HIVE

In order to simulate real-world use case, we use Hive to partition the sample data into different partitions on year/month/day because in real-world case the data could be streamed by such as use [Azure Stream Analytics](https://azure.microsoft.com/en-us/services/stream-analytics/?cdn=disable) and stored in the similar way.

This step will run about 8 minutes.

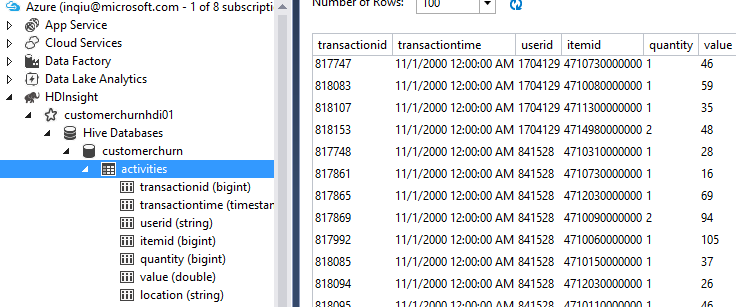
There are two partitioned hive tables users and activities created in hive on top of the partitioned azure blobs.

For hive table definition:

*In Visual Studio, open Server Explorer -> HDInsight -> {CLUSTERMNAME} ->Hive Databases -> customerchurn*

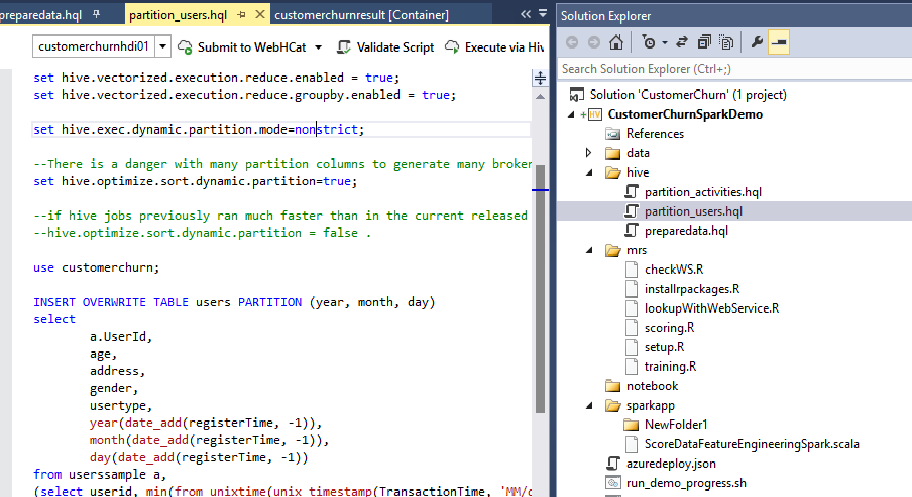
For hive table data:

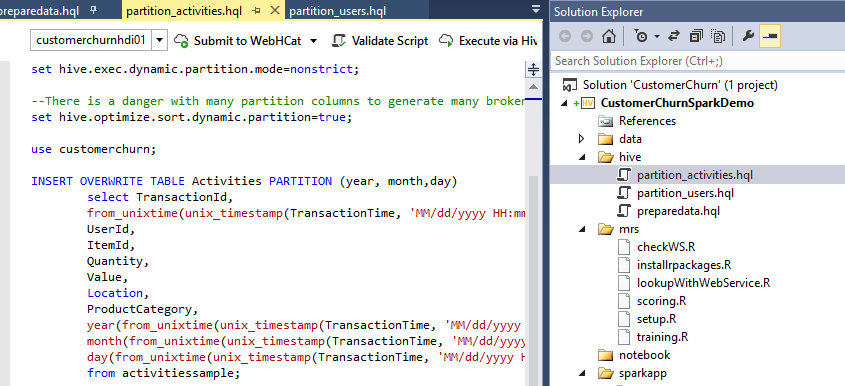
*In Visual Studio, open Server Explorer -> HDInsight -> {CLUSTERMNAME} ->Hive Databases -> customerchurn -> {table} -> right click -> top 100 rows*

**

For hive query how to partition:

*In Visual Studio, open the file partition\_users.hql under hive folder in the solution explorer to view the hive query:*

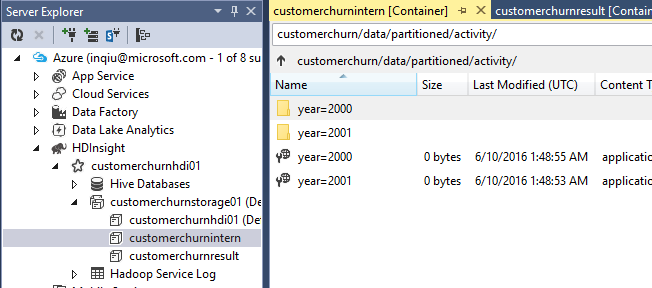


*In Visual Studio, open the file partition\_activities.hql under hive folder in the solution explorer to view the hive query:* 

For data in blob container:

*In Visual Studio, open Server Explorer-> HDInsight -> {CLUSTERSTORAGENAME} -> customerchurnintern -> data -> partitioned -> activities (or users)*

*Go to each partitions to see data.*



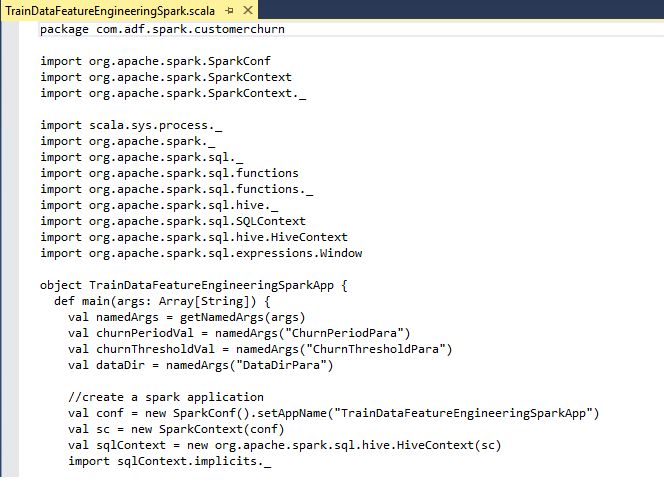
## Step 4: Feature Engineering using SPARK

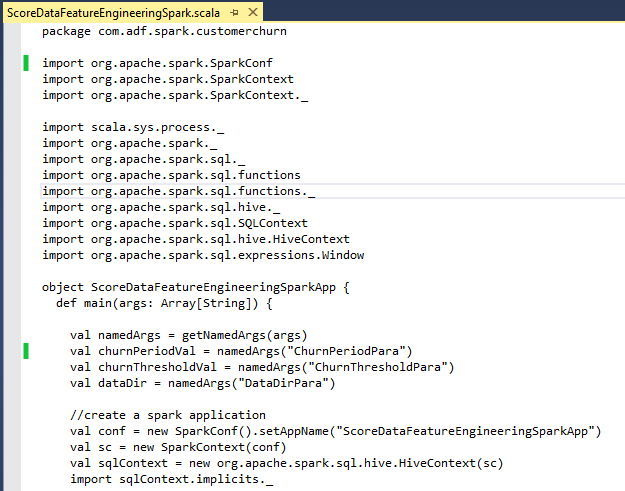
We use the Spark to do the feature engineering.

We create two self-contained applications using the Spark API in order to make it expendable to integrate with other Azure product such as Azure Data Factory. We will walk through application in Scala (with sbt).

*In Visual Studio, open the file TrainDataFeatureEngineeringSpark.scala under sparkapp/training/src/main/scala and ScoreDataFeatureEngineeringSpark.scala under sparkapp/scoring/src/main/scala folder in the solution explorer:*

The applications are very similar except the data set size.

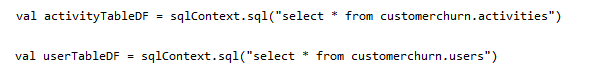
**



Note that the applications should be defined a main() method.

Both applications need three parameters: ChurnPeriod, ChurnThreshold, DataDir.

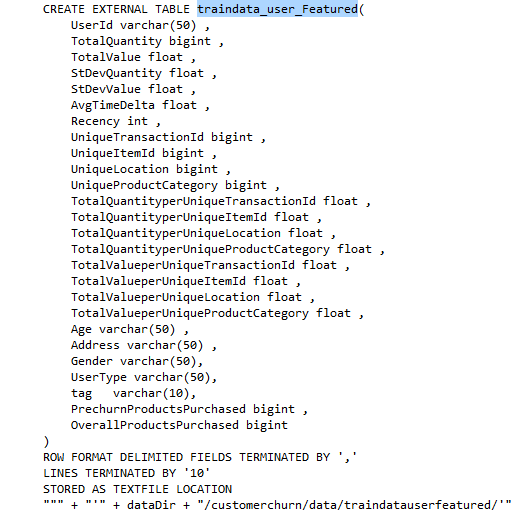
In both applications, we create two dataframes in Spark, one for activities, one for users. Both of them use the sqlContext to read the data from the partitioned hive tables.

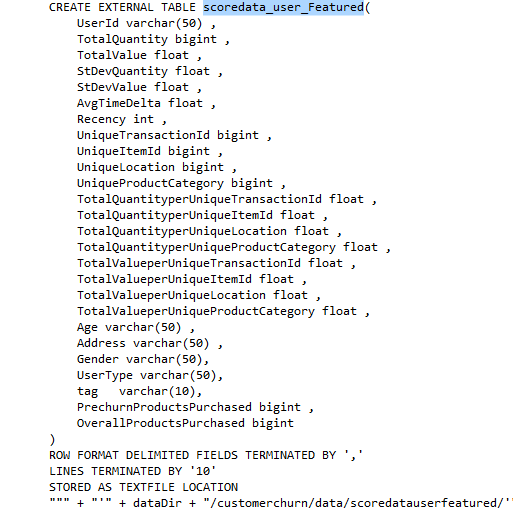


We join these two dataframes to get the features we need for modeling and prediction.

In order to save the data with the features, we create external HIVE tables in the Spark application which traindata\_user\_Featured for train data and scoredata\_user\_Featured for all data.

Both of the table have the following columns which you can explore them in the Hive Database and in spark application code.





As we stated before, both train data set and scoring data set use the same sample data. The reason is we try to demonstrate two different data flow independently. For train data set we choose 70% of the sample data randomly by using the sample method in Spark and for scoring data set we use all sample data. We then save the featured data through hive table to Azure Blob.





In the shell script, we first use open source build tool sbt to build the Spark applications.

* The sbt will produce the Spark Application for feature engineering of train data under yourWorkDirectory/sparkapp/training/target/scala-2.10 using the configurations in build.sbt. The name of the Spark application is generated by the configurations defined in build.sbt.
* The sbt will produce the Spark Application for feature engineering of all data under yourWorkDirectory/sparkapp/scoring/target/scala-2.10 using the configurations in build.sbt. The name of the Spark application is generated by the configurations defined in build.sbt.

Then we use spark-submit to call the spark applications in the shell script.

After this step complete, you can see the featured data in Blob and in Hive database

For Blob data

*In Visual Studio, open Server -> HDInsight -> {CLUSTERSTORAGENAME} -> customerchurnintern -> data -> traindatauserfeatured (or scoredatauserfeatured)*

For hive table data:

*In Visual Studio, open Server Explorer -> HDInsight -> {CLUSTERMNAME} ->Hive Databases -> customerchurn -> traindata\_user\_featured -> right click -> top 100 rows*

*In Visual Studio, open Server Explorer -> HDInsight -> {CLUSTERMNAME} ->Hive Databases -> customerchurn -> scoredata\_user\_featured -> right click -> top 100 rows*

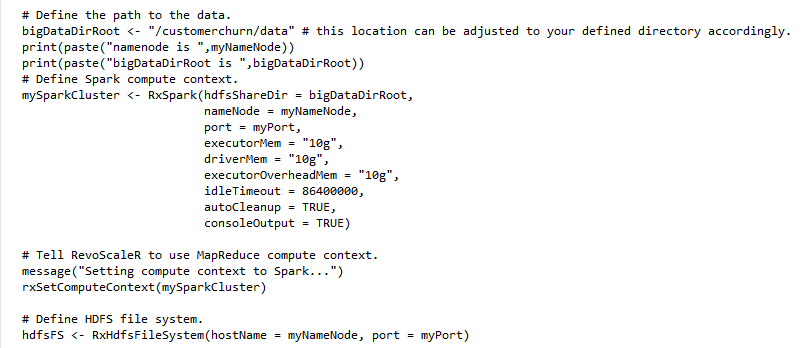
## Step 5: Train and persist Model using MRS

Now we have the featured data for train data set. let’s build a model.

*In Visual Studio, open the file setup.R under mrs folder in the solution explorer:*

Point out the code in setup.R that setup the computeContext.

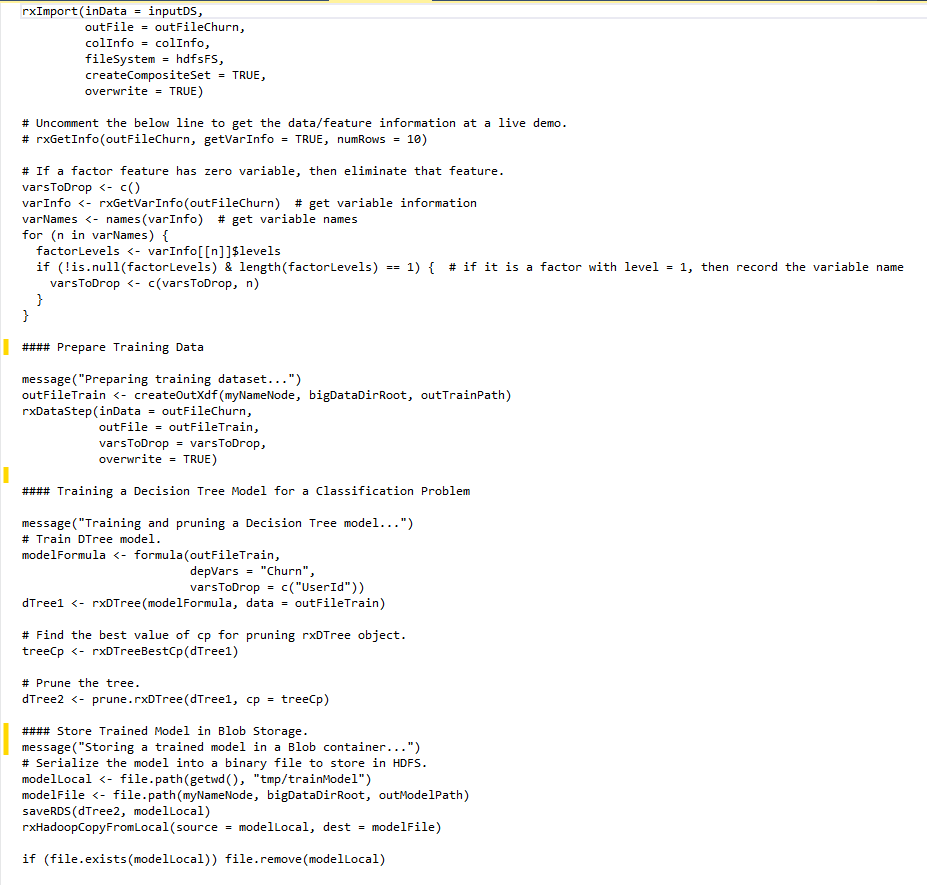
We use the spark computeContext for RevoScaleR.



*In Visual Studio, open the file training.R under mrs folder in the Solution Explorer.*

Point out the code in training.R that builds the model.

As you can see in the code, we import the data from Blob storage produced in Step 4 for train data. Then we train a Decision Tree Model for this binary classification problem by using Spark compute context. After the model trained, we serialize the model into a binary file to store through HDFS to Blob storage.



You can find the stored model named trainModel in the Blob storage.

*In Visual Studio, open Server Explorer -> HDInsight -> {CLUSTERSTORAGENAME} -> customerchurnintern -> data -> mrs*

## Step 6: Feature engineering all data using Spark

We must featurize all the customer activity and user data before make prediction. The feature engineering uses the same method we use for the train data. Please see Step 4.

## Step 7: Scoring using MRS

After we build a model and complete the feature engineering for all data, now we can do scoring to get the prediction on all customers.

*In Visual Studio, open the file scoring.R under mrs folder in the solution explorer:*

Point out the code in scoring.R that does the prediction.

As you can see in the code, we import the data from Blob storage produced in step 4 for all data. Then we load the saved model produced in step 5 and unserialize it. Then we use rxPredict to do the scoring by using Spark compute context. We then save the prediction results through HDFS to Blob storage.



You can find the prediction results in the Blob storage.

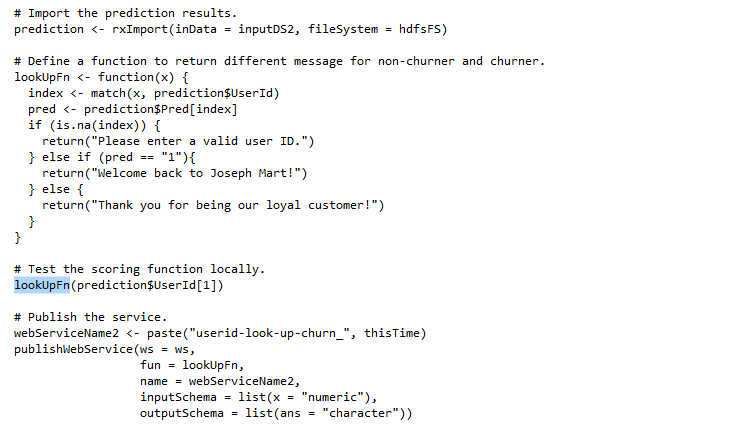
*In Visual Studio, open Server Explorer -> HDInsight -> {CLUSTERSTORAGENAME} -> customerchurnintern -> data -> predictions*

## Step 8: Deploy AML web service for lookup

In MRS, you can also publish Azure Machine Learning. We deploy a lookup web service so we can use an web application such as Joseph Mart to show some prediction results.

The run\_demo.sh will prompt the name of the web service it deployed in Step 8.

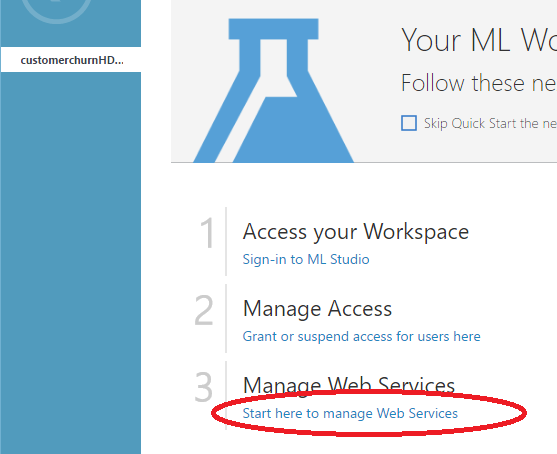
*In Visual Studio, open the file lookupWithWebService.R under mrs folder in the Solution Explorer.*



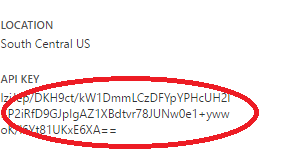
## Step 9: Joseph’s Mart to use AML web service

First, we need to find out the Find out the web service URL and API Key as followings

* *Open* ***Azure Machine Learning*** *through Portal*. It will open the Azure Machine Learning portal.
* *Select the workspace you want to use. And* ***Start here to manage Web Services****.*

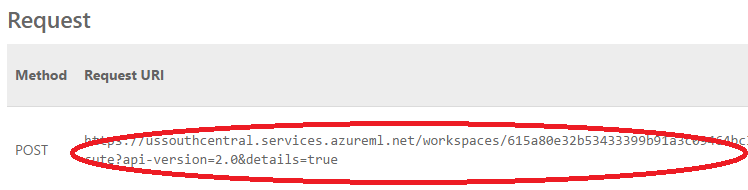
**

* *Select the web service you just deployed in Step 8*
* *Select the default endpoint. Scroll to the bottom of the page, On the right, you can find the API KEY*

**

* *On the same page, click on “****REQUEST/RESPONSE****”, it will open a new web page.*

*Copy the* ***Request URL*** *in the Request Section.*

**

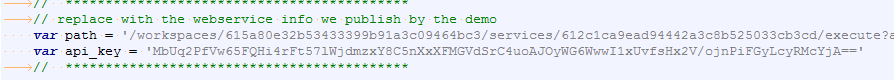
Next we need to make some changes to the web Server file.

* *SSH to the Edge Node, go to the work directory. Run the followings.*

*$ cd $HOME/customer\_churn\_demo/Website*

* *Open the file server.js*
* *Use any editor in Linux such as vi to change the path and API Key in this file according to the URL and API Key you get above.*

Please DO NOT use the <https://ussouthcentral.services.azureml.net> part from the URL. The two variable should look like the following



Next, start the Webserver.

*$ node server.js*

Now we can run the web application.

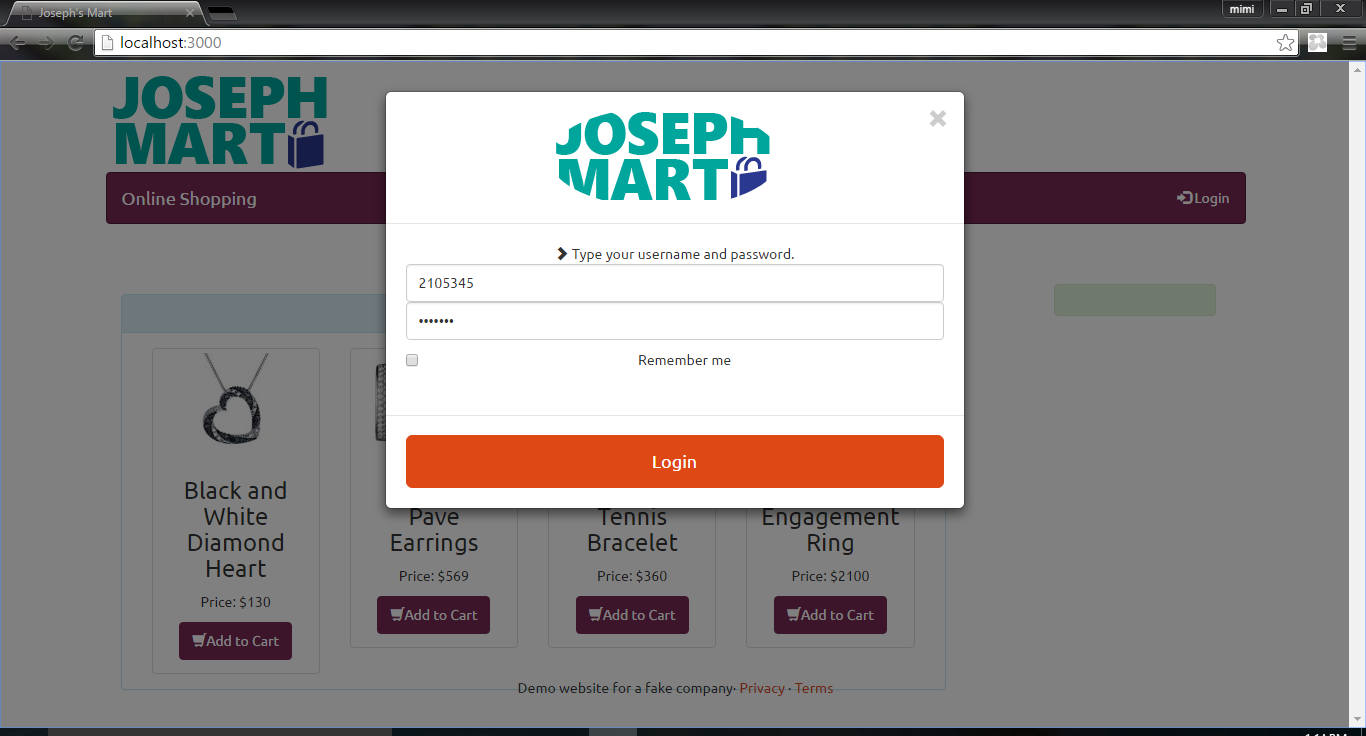
*In your Windows machine, open any web browser, put localhost:3000 in the address.*

It will bring up the Joseph’s Mart web application.

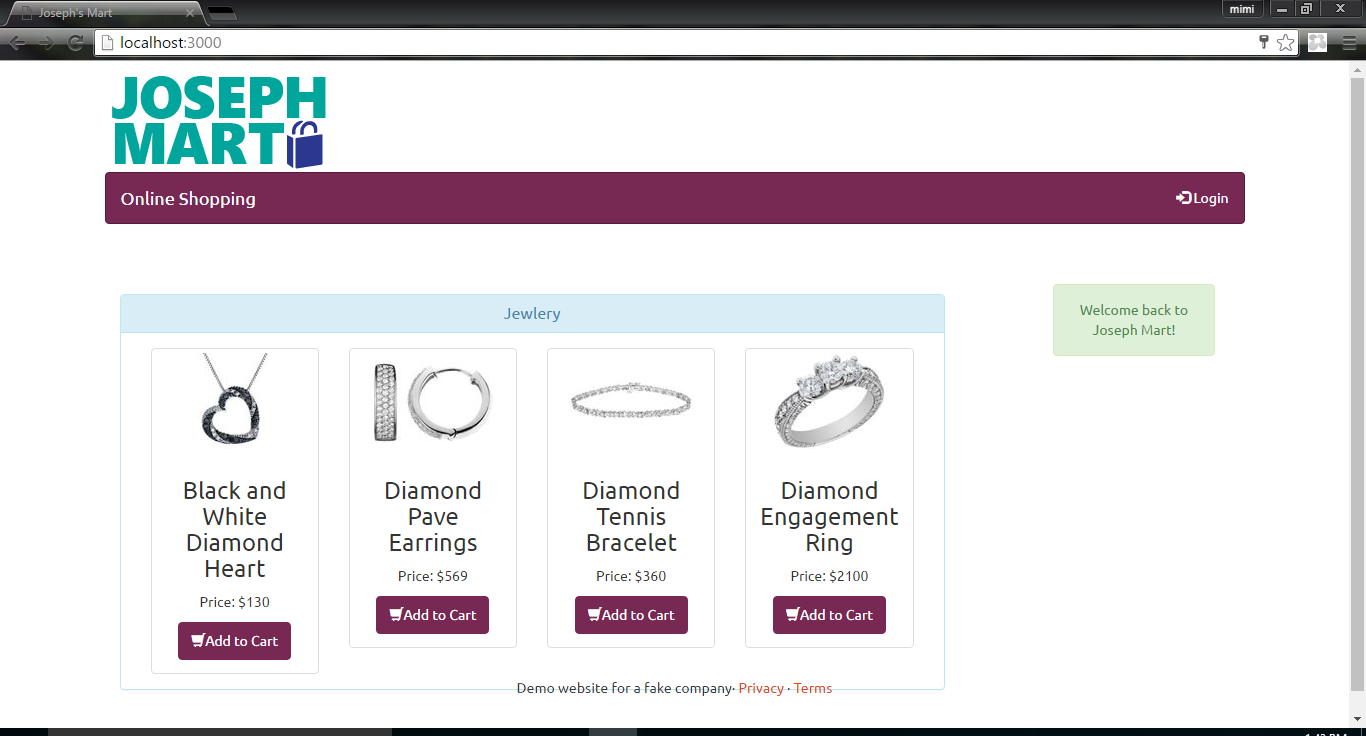
*Click login on the upper right corner.*

*Input any userId from the users.csv file as username and anything as password.*

(you can use these two userids for demo: 1029383, 1206241)



It will call the Azure Machine Learning Web Service to find out the prediction produced in step 7 for this userId. For user predicted as Churn user, the webpage will display “Welcome back to Joseph Mart!”, for user predicted as Non-Churn user, the webpage will display “Thank you for being our loyal customer!”



# VISUALIZATION

We provide two approaches to do the visualizations in this demo. Both approaches use Power BI Desktop as the development tool to build reports and then publish the pre-build reports to Power BI Online for building dashboards. Power BI Desktop has great capabilities of performing different data editing, transformation, joining and etc. and it has the flexibility of connecting to the following Azure products:

* Microsoft Azure SQL Database
* Microsoft Azure SQL Data Warehouse
* Microsoft Azure Marketplace
* Microsoft Azure HDInsight
* Microsoft Azure Blob Storage
* Microsoft Azure Table Storage
* Azure HDInsight Spark (Beta)
* Microsoft Azure DocumentDB (Beta)
* Microsoft Azure Data Lake Store (Beta)

The two connections we are using for the two approaches are: *Microsoft Azure Blob Storage* and *Azure HDInsight Spark (Beta)*.

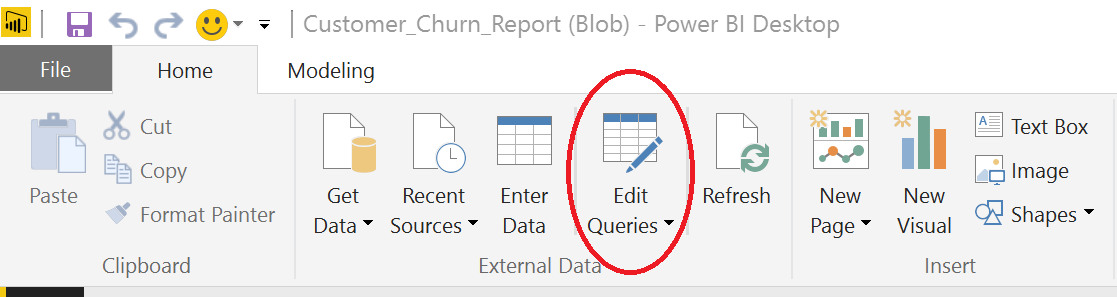
## PoweBI Desktop with Blob Storage Connection

The demo comes with a pre-build Power BI desktop file (*Customer\_Churn\_Report (Blob).pbix*) that has the connection to Blob Storage. You only need to change the connection information to connect to the data source from the live demo.

*In Visual Studio, double click the file “Customer\_Churn\_Report (Blob).pbix” under Power BIDesktop folder in the solution explorer, it will open the Power BI desktop.*

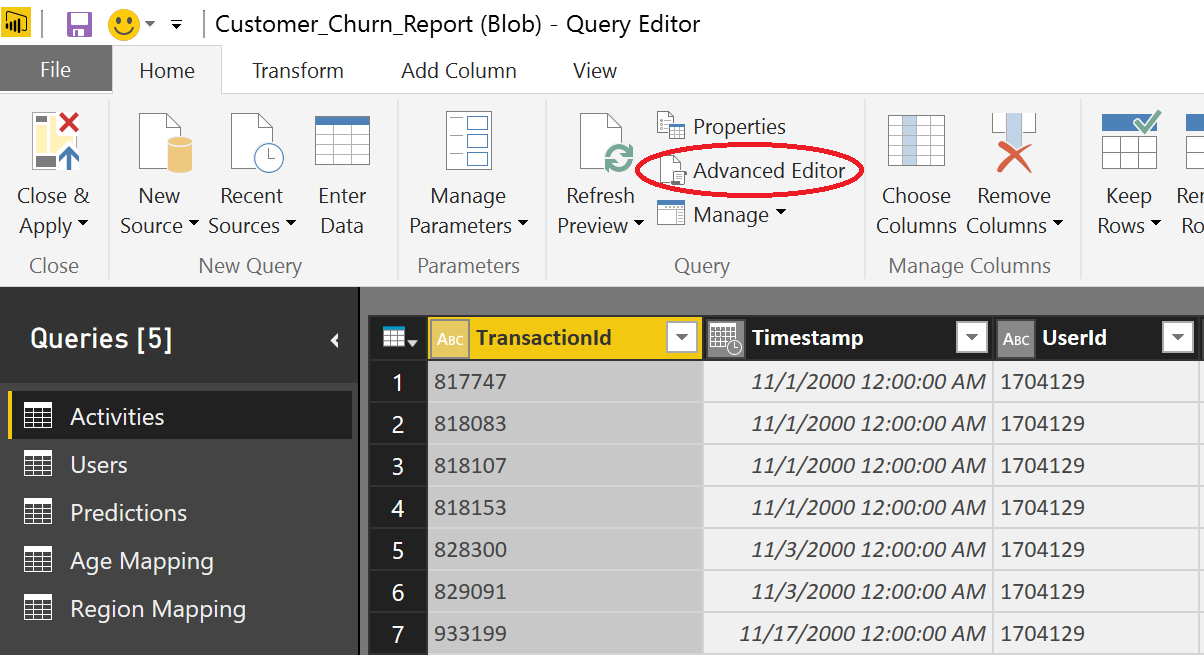
First of all, let’s change the connection info first by following the instruction below.

*In Power BI desktop, click on “Edit Queries” after open the file “Customer\_Churn\_Report (Blob).pbix”.*

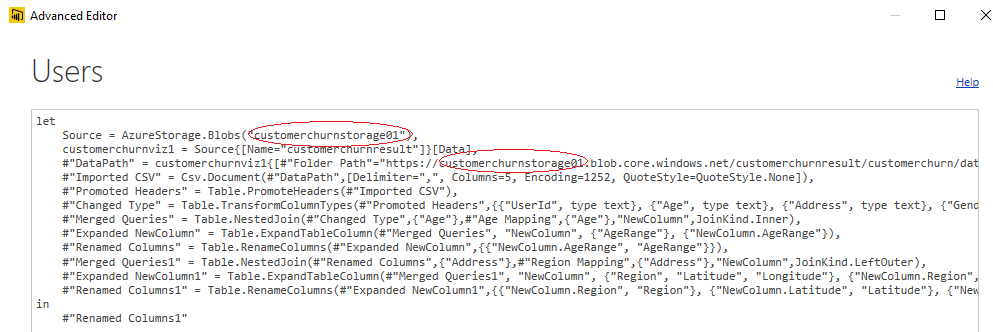
**

This will open “Query Editor” Window. In the left panel, it lists 5 queries.

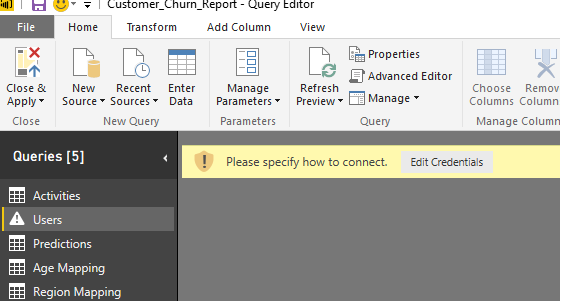
*Choose one of the queries on the left panel, click “Advanced Editor” on the menu.*



*Change all the storage names in the query.*

**

*Provide credentials (storage key) if you see this.*

**

*Click “Refresh Preview” -> “Refresh All”.*

This will refresh the data and dashboard.

*Click “Close & Apply” to save the changes of the queries.*

*Now, repeat the above steps to change for all the queries.*

## PoweBI Desktop with Azure HDInsight Spark Connection (Beta)

Azure HDInsight [now offers](http://blogs.technet.com/b/dataplatforminsider/archive/2015/07/10/microsoft-lights-up-interactive-insights-on-big-data-with-spark-for-azure-hdinsight-and-power-bi.aspx) a fully managed Spark service. This capability allows for scenarios such as iterative machine learning and interactive data analysis. Power BI allows you to directly connect to the data in [Spark on HDInsight](http://go.microsoft.com/fwlink/?LinkId=616226) offering simple and live exploration.

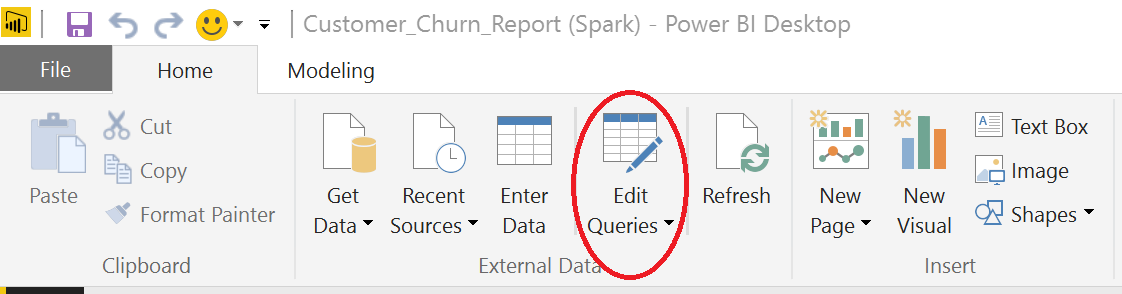
Power BI allows you to connect directly to your Spark cluster and explore and monitor data without requiring a data model as an intermediate cache. This offers interactive exploration of your data and automatically refreshes the visuals without requiring a scheduled refresh.

The demo also comes with a pre-build Power BI desktop file (*Customer\_Churn\_Report (Spark).pbix*) that has connection to Azure HDInsight Spark. Again, you only need to change the connection information to connect to the data source from the live demo.

*In Visual Studio, double click the file “Customer\_Churn\_Report (Spark).pbix” under Power BIDesktop folder in the solution explorer, it will open the Power BI desktop.*

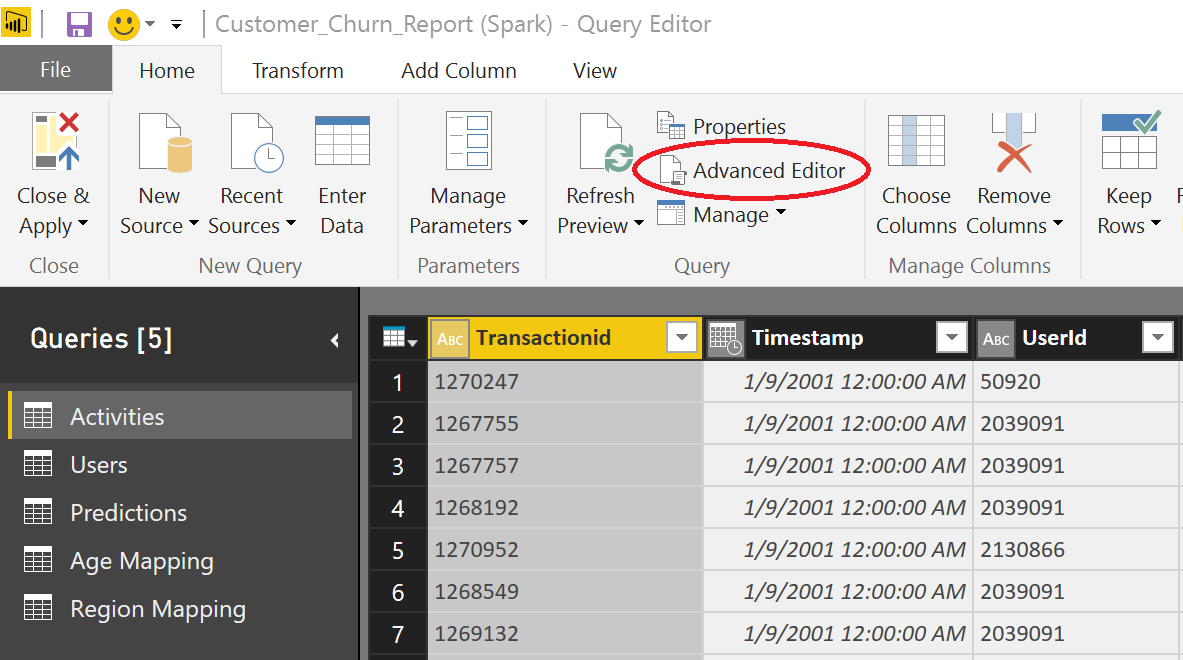
Now, let’s change the connection info first by following the instruction below.

*In Power BI desktop, click on “Edit Queries” after open the file “Customer\_Churn\_Report (Spark).pbix”.*

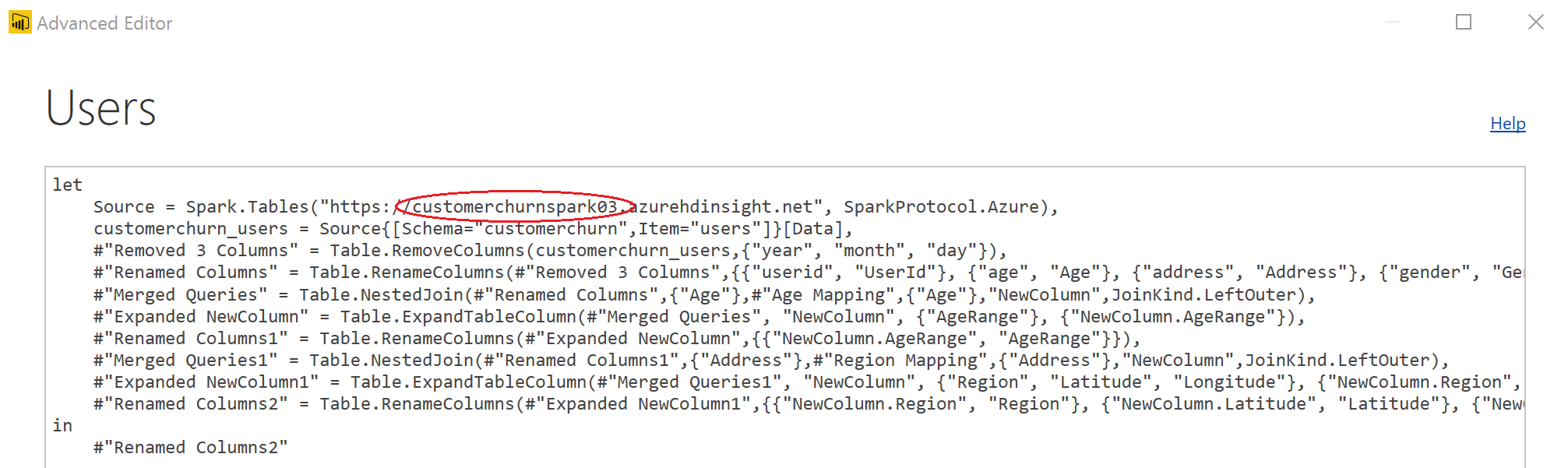
**

This will open “Query Editor” Window. In the left panel, it lists 5 queries.

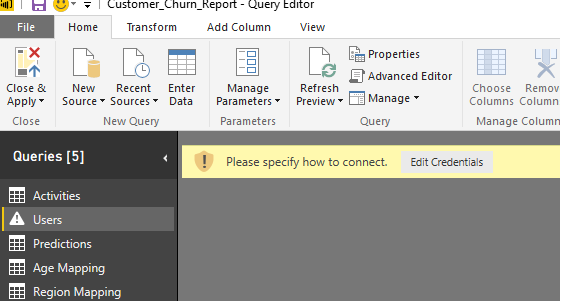
*Choose one of the queries on the left panel, click “Advanced Editor” on the menu.*



*Change the storage name in the query. (you might notice in this step that Azure HDInsight Spark connection retrieves the data slightly different than Blob Storage connection does. In the Spark connection, you only need to edit the storage name once in each query.)*

**

*Provide credentials (storage key) if you see this.*

**

*Click “Refresh Preview” -> “Refresh All”.*

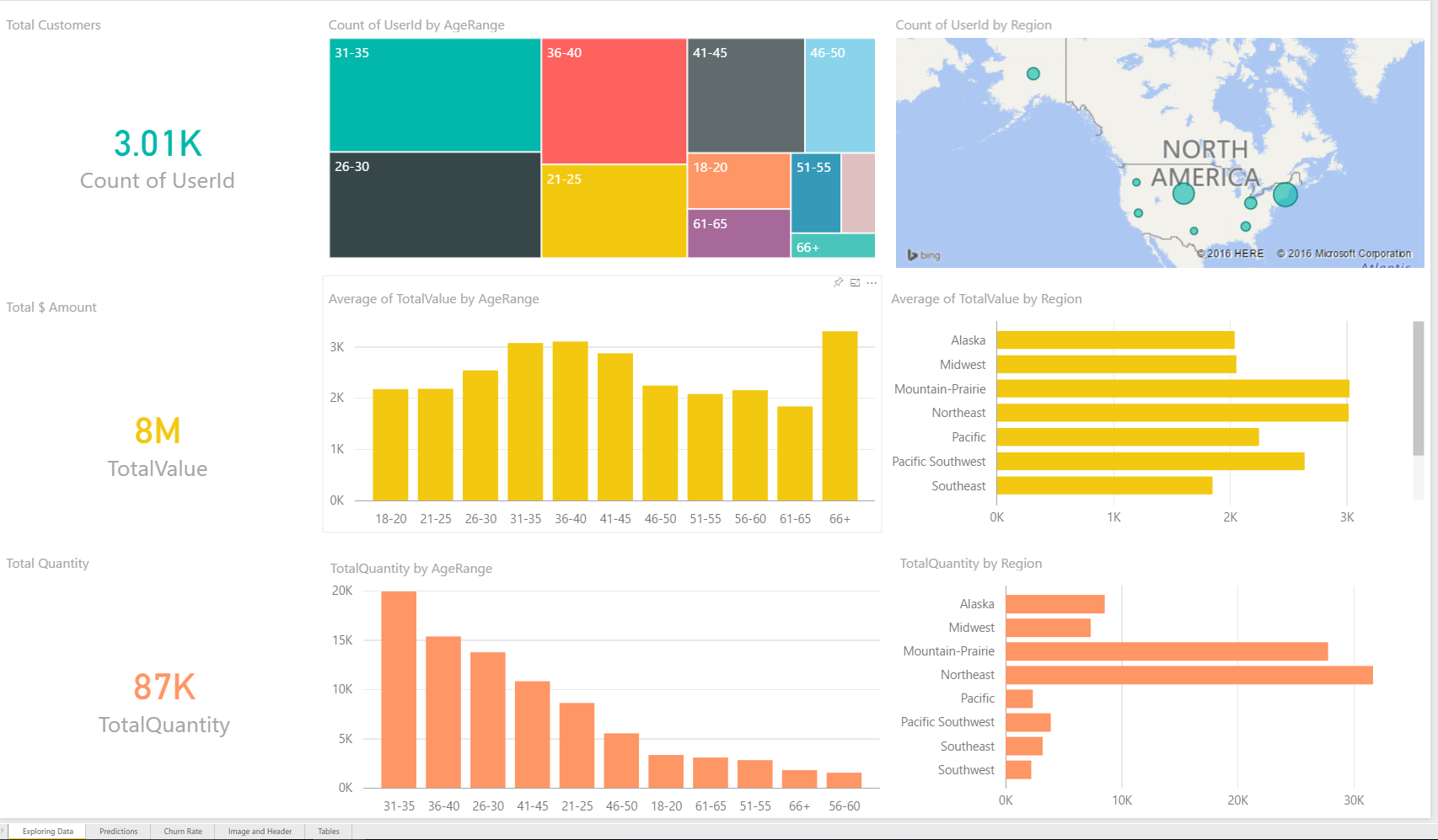
This will refresh the data and dashboard.

*Click “Close & Apply” to save the changes of the queries.*

*Now, repeat the above steps to change for all the queries.*

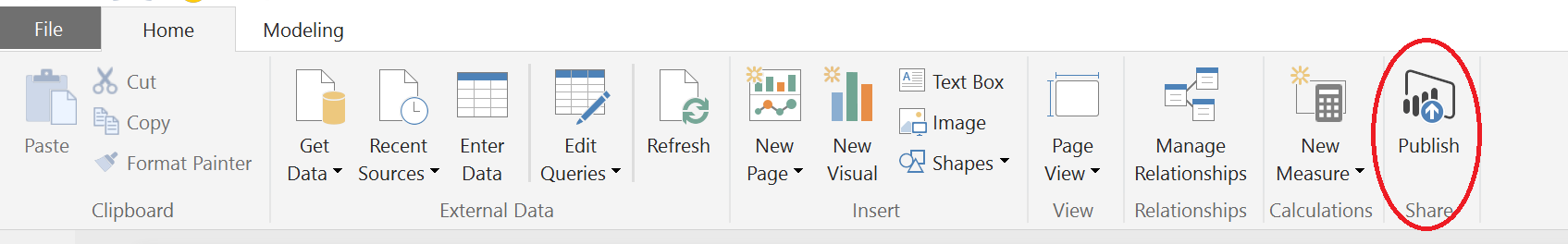
## Publish Reports in Power BI Desktop (Same for Blob and Spark)

After successfully changing the connection, you will see 5 tabs (Exploring Data, Predictions, Churn Rate, Image and Header and Tables) are created in the report. Each tab contains different visualizations that can be used for building the dashboards later.

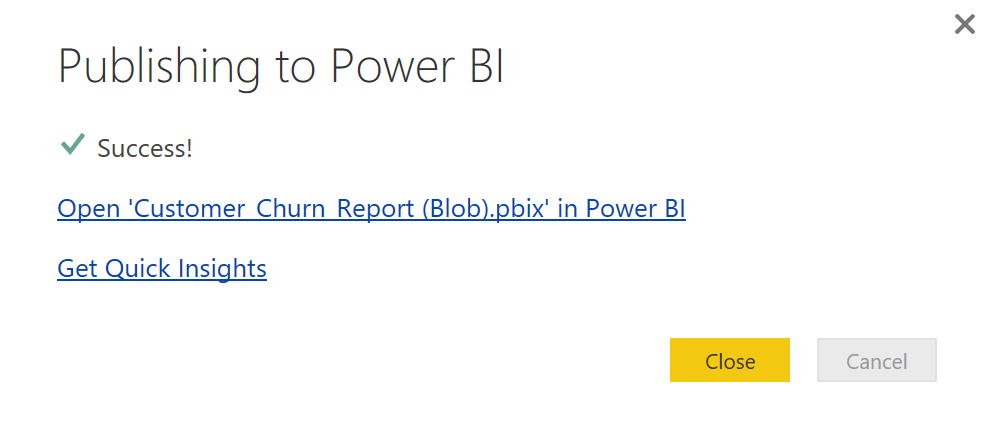


In order to view the report in Power BI Online, we need to publish it from the Power BI Desktop.

Click “Publish” and sign in with you credentials. Then you can choose a destination, such as My Workspace, that you want the report to be published.



You will see the below message when the report is published successfully.



## PoweBI Online (Same for Blob and Spark)

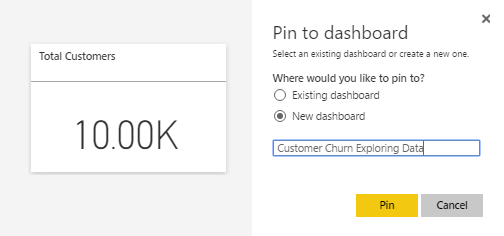
In this demo, we use Power BI Online to build two dashboards based on the pre-build reports in Power BI Desktop files.

Click on “Open ‘Customer\_Churn\_Report (Blob).pbix’ in Power BI” in the above figure or go to [https://Power BI.microsoft.com/](https://powerbi.microsoft.com/) and sign in with your credentials.

The published reports will be listed under “Reports” section, so you will see two reports are published in our demo “Customer\_Churn\_Report (Blob)” and “Customer\_Churn\_Report (Spark)”. Now, we are using the report connected to Blob as an example and the process is the same for the report connected to Spark.

Click “Reports” -> “Customer\_Churn\_Report (Blob)” to review the report. The report will look the same as it is in the Power BI Desktop.

Click “Pin visual” on the right upper corner of each visualization to pin it to a new dashboard.

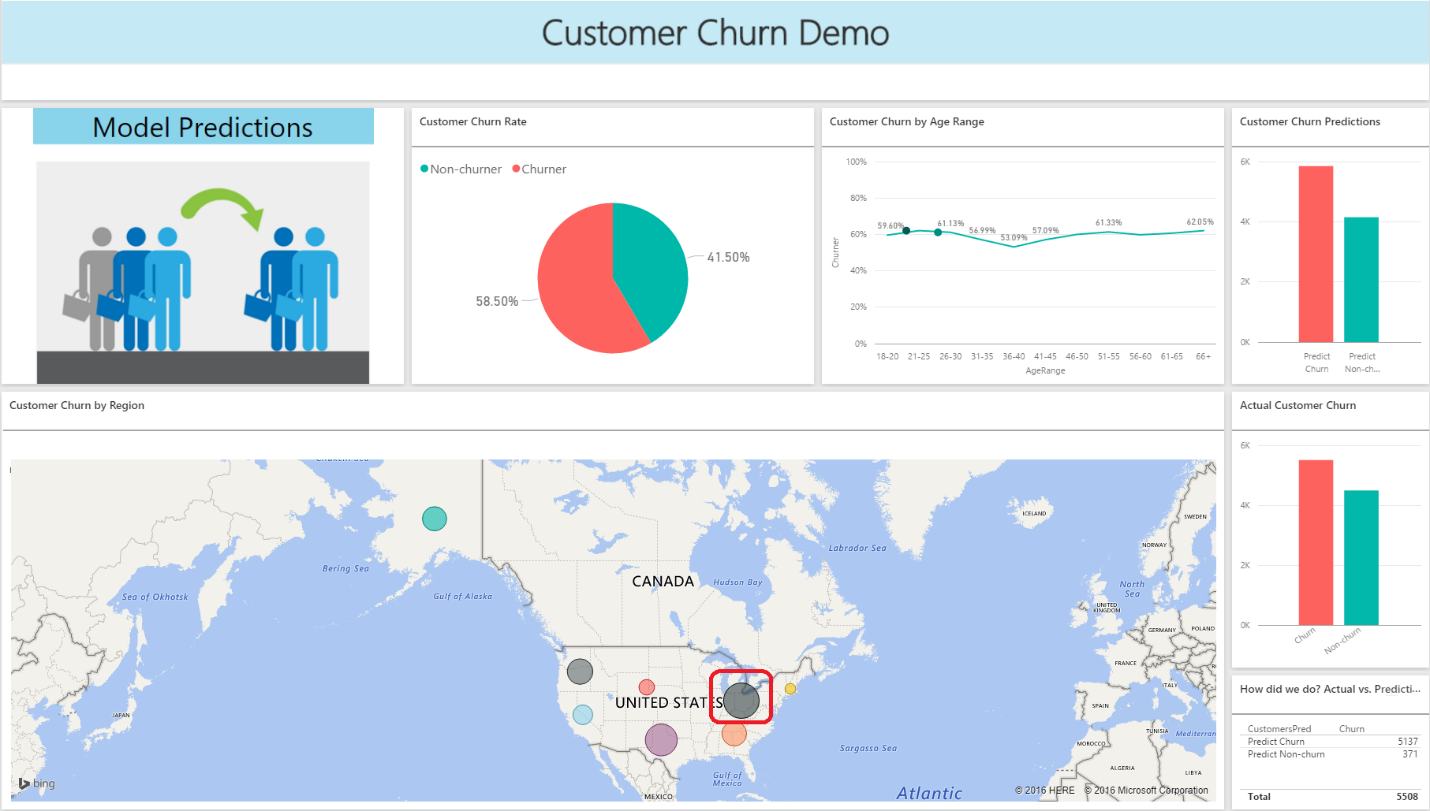


We pin all visualizations for exploring the data to a **“Customer Churn Exploring Data”** dashboard and pin all visualizations for prediction results to a **“Customer Churn Model Predictions”** dashboard. In each dashboard, we move the pinned visualizations to their best positions.

In the **Customer Churn Exploring Data** dashboard below, we can see 10,000 users made 285,000 products purchases in 2000 and 2001 that contribute to $27 million revenue. Looking at users at different age ranges, 31-35 age group has the largest number of users and makes the largest number of purchases than any other groups, but users over 66 have the highest average of total value (revenue). In terms of region (click on the “Count of UserId by Region” while talking), the Northeast region has the largest number of users and makes the largest number of purchases, and the purchases also have high values.



In the **Customer Churn Model Predictions** dashboard below, we can see there are 58.5% users churned based our prediction. If we look at user in different age ranges, we will find users over 66 years old are more likely to churn than other age groups and users in 36-40 age group are less likely to churn. Speaking of the region (click on the largest bubble in the map while talking), users in Midwest has the highest churn rate and users in Northeast has the lowest churn rate. In terms of the promotion, we could target users over 66 years old, who has the highest purchase power, and users in Midwest.

****

# CLEANUP

If you create a separate resource group in the portal, you can simple delete the resource group which will clean everything.