CIS5200 Term Project Tutorial

Authors: Himani Batra; Katia Kermoyan; Nishant Shristiraj; Srihitha Reddy Sivinnagari

Instructor: Jongwook Woo

Date: 05/18/2017

Lab Tutorial

Himani Batra (hbatra@calstatela.edu)
Katia Kermoyan (kkermoy@calstatela.edu)
Nishant Shristiraj (nshrist@calstatela.edu)
Srihitha Reddy Sivinnagari (ssivann@calstatela.edu)

Big Data Analysis of Taxi Services in Chicago and New York

Objective

Kaggle is a platform for predictive modelling and analytics competitions in which statisticians and data miners compete to produce the best models for predicting and describing the datasets uploaded by companies and users. For this project, we used two data sets of total size 2.4GB about taxi services in Chicago and New York from kaggle.com. Both datasets include similar columns like total amount of fare, time taken for each ride, pick up and drop off locations, distance travelled and company/ vendor. The difference between them is that the Chicago dataset contains 12 month-data about Chicago taxi rides in 2016, while the New York dataset contains data about New York taxi rides in January and February of 2014.

The analysis performed on the Datasets are:

- Which companies have the highest earnings in Chicago in 2016?
- Which taxi-ids generate the highest fare with the respective number of rides?
- What is the total monthly comparison between Credit and Cash payments?
- What is the total number of monthly pick-ups in Chicago?
- Which communities in Chicago have the highest demand for pick-ups and Drop-offs?
- Which community of Chicago has the maximum miles travelled by taxi in 2016?
- Which hour of the day and which day of the month has most pickups in New York?
- What are the regions with highest demand and longest distance travelled in New York?

Which is the highest earning vendor in New York?

Introduction

This project aims at performing data analysis and providing insights on Chicago Taxi dataset and New York Taxi Dataset using HIVE, PIG and presenting visualization on Microsoft Excel - Power view, 3D maps and Tableau. Also, a predictive analysis on both the dataset have been presented using the Linear Regression model in Azure ML. In this tutorial, through each analysis we did you'll learn how to use BigInsights to:

- Load data from local desktop(windows) to Linux shell and unzip zip files
- Extract CSV files
- Download and upload files to HDFS
- Create Hive tables to query the datasets for analysis
- Access the tables in the warehouse of Hive in Ambari
- Create tables to query the datasets using PIG
- Use Microsoft Excel to connect to BigInsights (directly or using an ODBC connection) to retrieve the analyzed data
- Use Excel 3D Map for 3D visualization – Highest Earning companies in Chicago and Communities with highest miles

- Use Tableau for visualization of the analyzed data
- Use Azure ML for prediction analysis

Prerequisites

Everything you need to go through the scripts and queries is already provisioned with the cluster. To export the analyzed data to Microsoft Excel, you must meet the following requirements:

- You must have **Microsoft Excel 2010**, **2013 or 2016** installed.
- You must have your Excel PowerView and 3D Map enabled.
- Tableau 10.1 or higher installed for visualization of the analyzed data
- You must have Microsoft Hive ODBC Driver to import data from Hive into Excel. Select either the 32-bit or 64-bit version based on your version of Microsoft Excel. But, BigInsights does not support it yet as of Sept 2016.
- Azure ML

Platform Specifications

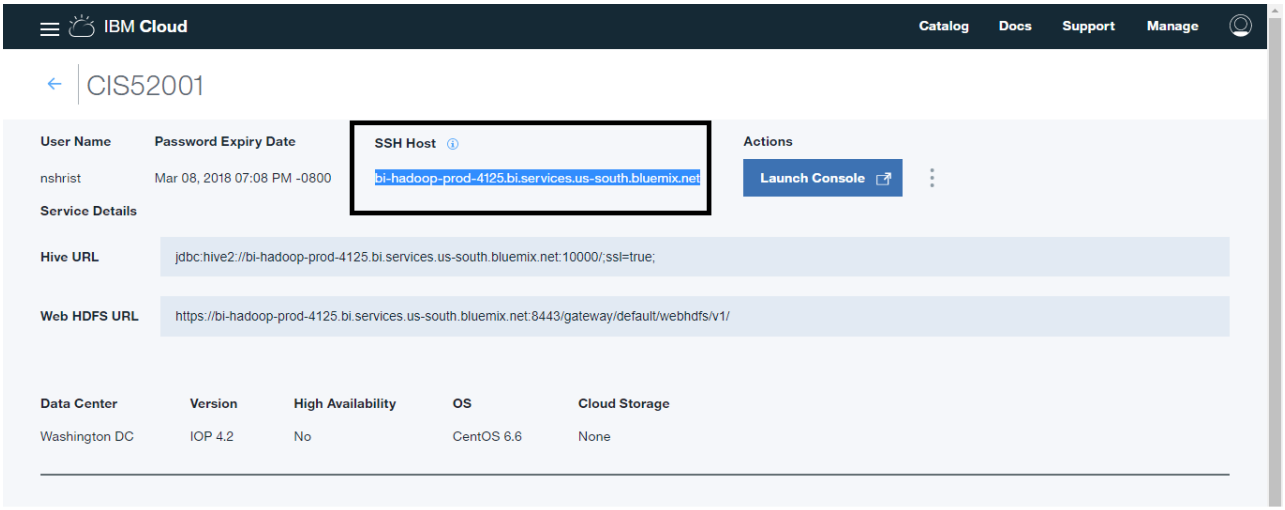
- IBM Bluemix BigInsights
- CPU Speed: 2.3 Ghz
- # of CPU cores: 4 VCPUs (Data Node), 12 VCPUs (Management Node)
- # of nodes: 1 Data Node, 1 Management Node
- Total Memory Size: 1 TB SATA

Chicago and New York Dataset Loaded into BigInsights

You need to download the dataset to your local desktop from the following links, in order to retrieve the .zip and .tar files.

- https://www.kaggle.com/chicago/chicago-taxi-rides-2016
- https://www.kaggle.com/kentonnlp/2014-new-york-city-taxi-trips

Once the files are downloaded, you need to remotely access your BigInsights, that you have executes in your IBM Bluemix using *ssh*.



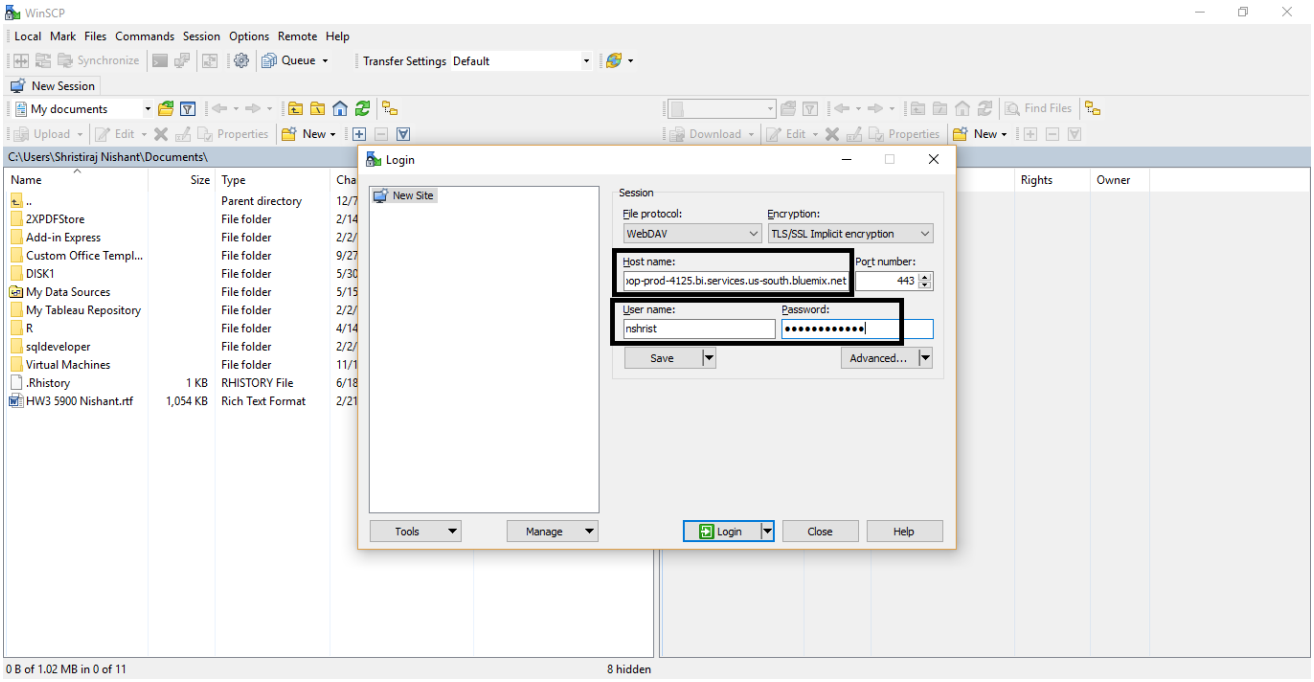
Note: ssh host can be found inside the cluster you created in BigInsights.

The size of the dataset is nearly 2.4 GB. You can access the downloaded data from your local desktop and place it in remote server of your cluster using **WinSCP**. WinSCP (Windows Secure Copy) is a free and open-source SFTP, FTP, WebDAV and SCP client for Microsoft Windows. Its main function is secure file transfer between a local and a remote computer. Beyond this, WinSCP offers basic file manager and file synchronization functionality. For secure transfers, it uses Secure Shell (SSH) and supports the SCP protocol in addition to SFTP.

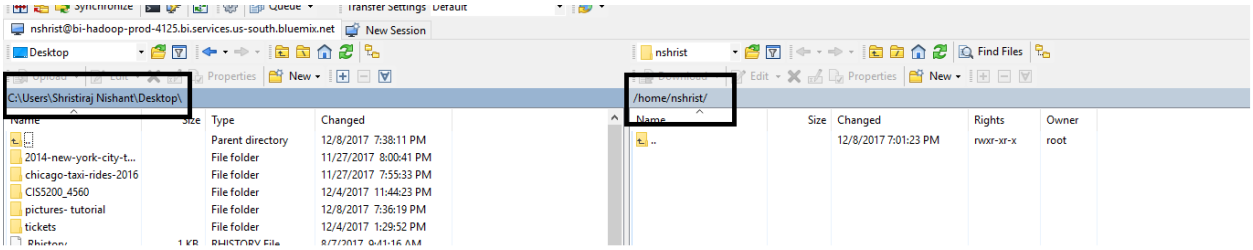
You can download and install WinSCP from the following link:

- https://winscp.net/eng/download.php

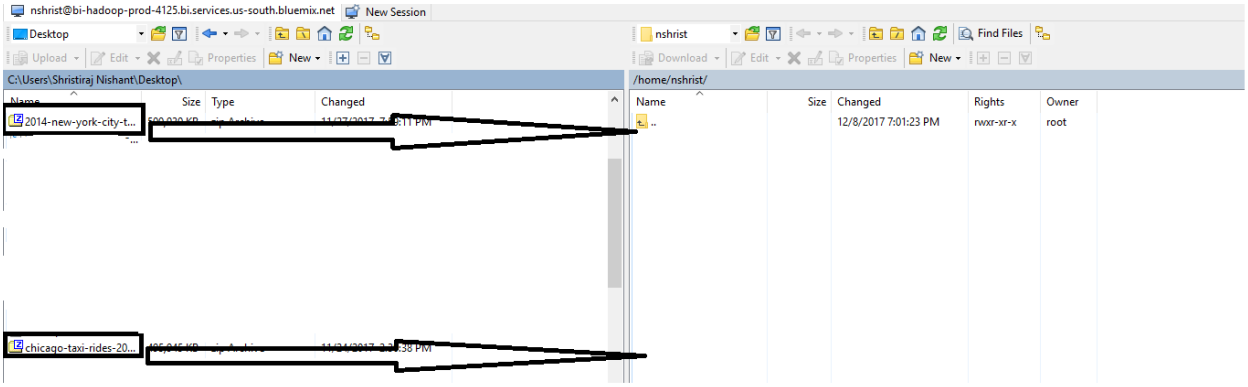
Note: For installation, accept the terms and condition, choose *Full Upgrade* when prompted in SetUp Type. After the installation, open WinSCP, enter the *ssh* host of your cluster, along with your username and password you used for creating the cluster., and press login.



Once you are logged in, you will see two different sides, on the left is your local desktop and right is the remote host. You should be in the folder on the local host, where you have saved your downloaded files. In this case the files are saved on desktop.



Drag and drop the required files to the remote host from the local desktop in WinSCP. This process might take few minutes.



After this process, we need to upload the data in HDFS. Before we upload these files to HDFS we have to create a directory in HDFS. Run the following HDFS commands to create to store these file into a directory in HDFS.

```
$ hdfs dfs -mkdir /user/nshrist/chicago
$ hdfs dfs -mkdir /user/nshrist/newyork
```

```
login as: nshrist
nshrist@bi-hadoop-prod-4125.bi.services.us-south.bluemix.net's password:
IBM's internal systems must only be used for conducting IBM's business or for pu
rposes authorized by IBM management
Use is subject to audit at any time by IBM management
-bash-4.1$ hdfs dfs -mkdir /user/nshrist/chicago
-bash-4.1$ hdfs dfs -mkdir /user/nshrist/newyork
-bash-4.1$
```

We need to unzip the files to get the files in the respective directories.

```
$ unzip Chicago-taxi-rides-2016.zip
```

```
-bash-4.1$ unzip chicago-taxi-rides-2016.zip
Archive:  chicago-taxi-rides-2016.zip
  inflating: chicago_taxi_trips_2016_01.csv
  inflating: chicago_taxi_trips_2016_02.csv
  inflating: chicago_taxi_trips_2016_03.csv
  inflating: chicago_taxi_trips_2016_04.csv
  inflating: chicago_taxi_trips_2016_05.csv
  inflating: chicago_taxi_trips_2016_06.csv
  inflating: chicago_taxi_trips_2016_07.csv
  inflating: chicago_taxi_trips_2016_08.csv
  inflating: chicago_taxi_trips_2016_09.csv
  inflating: chicago_taxi_trips_2016_10.csv
  inflating: chicago_taxi_trips_2016_11.csv
  inflating: chicago_taxi_trips_2016_12.csv
  inflating: column_remapping.json
  inflating: data_dictionary.csv
-bash-4.1$
```

Among the files, chicago_taxi_trips_2016_01, …02, refers to the respective months in a year. For example, 01 – January 02- February, and so on.

After retrieving these files, we need to put them into the respective directory for Chicago, by using the following code.

```
$ hdfs dfs -put *.csv chicago
```

```
-bash-4.1$  hdfs dfs -put *.csv chicago
-bash-4.1$
```

We need to repeat similar process with the New York data set. For putting the files we cannot use "*.csv" as we already have Chicago csv files. Since New York has just two files, we can put them manually in the respective directory. Use the following codes:

```
$ unzip 2014-new-york-city-taxi-trips.zip
$ hdfs dfs -put nyc_taxi_data_2014.csv.gz newyork
```

```
-bash-4.1$ unzip 2014-new-york-city-taxi-trips.zip
Archive:  2014-new-york-city-taxi-trips.zip
replace nyc_taxi_data_2014.csv.gz? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: nyc_taxi_data_2014.csv.gz
-bash-4.1$ hdfs dfs -put nyc_taxi_data_2014.csv newyork
put: `nyc_taxi_data_2014.csv': No such file or directory
-bash-4.1$ hdfs dfs -put nyc_taxi_data_2014.csv.gz newyork
-bash-4.1$
```

Create Hive tables to query the Chicago dataset

The following Hive statement creates an external table that allows Hive to query data stored in HDFS. External tables preserve the data in the original file format, while allowing Hive to perform queries against data within the file.

The following Hive statements will create a table **area**, by describing the fields within the files, delimiter (comma) between the fields, and location of the file. This will allow us to create Hive queries over the dataset.

Open hive shell environment as follows:

```
$ hive
```

In the hive shell, we need to copy and paste the following HiveQL code to create an external table "area".

Note: Make sure to replace "nshrist" with your account name in the following HQL code.

```
DROP TABLE IF EXISTS area;

CREATE EXTERNAL TABLE IF NOT EXISTS area(taxi_id int,
trip_start_timestamp timestamp,
trip_end_timestamp timestamp,
trip_seconds Int,
trip_miles float,
pickup_census_tract Int,
dropoff_census_tract Int,
pickup_community_area Int,
dropoff_community_area Int,
fare float,
tips float,
tolls float,
extras float,
trip_total float,
payment_type String,
company Int,
pickup_latitude Int,
pickup_longitude Int,
dropoff_latitude Int,
dropoff_longitude Int)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION '/user/nshrist/chicago';
```

Then, in hive shell, we need to check if area is shown.

```
hive> show tables;
```

Now, we can query the content of this table.

```
hive> select * from area limit 4;
```

```
hive> select * from area limit 4;
OK
NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    NULL    N
ULL     NULL    NULL    NULL    payment_type    NULL    NULL    NULL    NULL    N
ULL
85      2016-01-13 06:15:00     2016-01-13 06:15:00     180     0.4     NULL    N
ULL     24      24      4.5     0.0     0.0     0.0     4.5     Cash    107     1
99      510     199     510
2776    2016-01-22 09:30:00     2016-01-22 09:45:00     240     0.7     NULL    N
ULL     NULL    NULL    4.45    4.45    0.0     0.0     8.9     Credit Card     N
ULL     NULL    NULL    NULL    NULL
3168    2016-01-31 21:30:00     2016-01-31 21:30:00     0       0.0     NULL    N
ULL     NULL    NULL    42.75   5.0     0.0     0.0     47.75   Credit Card     1
19      NULL    NULL    NULL    NULL
Time taken: 0.107 seconds, Fetched: 4 row(s)
hive>
```

Create Hive Queries to Analyze Chicago Dataset

Analysis 1: Which companies have the highest earnings in Chicago in 2016?

For any business it is very useful to analyze the competitors. A clear idea about our competition helps us analyze and grow our business.

This analysis is carried out to find out companies doing the highest business based on their total income in 2016 in Chicago. The query groups the sum of total income for all months by the company code in a table called Company2. The companies are sorted out in descending order and top 50 companies are extracted from the dataset.

Copy and paste the following code in hive shell. The code will take about 30 seconds to execute.

```
---- Create the table company2 to extract the top 50 companies

CREATE TABLE IF NOT EXISTS Company2 as select company, sum(trip_total) as Total_F,
MAX(tips) from area Group by company ORDER BY Total_F DESC LIMIT 50;
```

After the query is executed, you should get the following output on the hive shell.

```
Stage-Stage-1: Map: 8  Reduce: 9   Cumulative CPU: 166.9 sec
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 4.23 sec   H
Total MapReduce CPU Time Spent: 2 minutes 51 seconds 130 msec
OK
Time taken: 147.361 seconds
hive>
```

In the hive shell, we need to check if Company2 is shown.

```
hive> show tables;

 ---- check the data of table Company2

hive> select * from Company2 limit 10;
```

You should get the following output after executing these codes.

```
hive> show tables;
OK
area
company2
Time taken: 0.046 seconds, Fetched: 2 row(s)
hive>  select * from Company2 limit 10;
OK
NULL    1.2637672617945094E8    450.0
107     8.486805805637527E7     496.5
101     2.8505380579580575E7    275.0
8       2.1218263000320364E7    196.75
119     2.013593595373038E7     99.99
109     1.98796181591572 46E7   139.25
82      7816315.779690586       400.0
10      5639657.199833024       120.12
92      4865292.870012505       112.15
90      3504771.61040915        52.04
Time taken: 0.094 seconds, Fetched: 10 row(s)
hive>
```

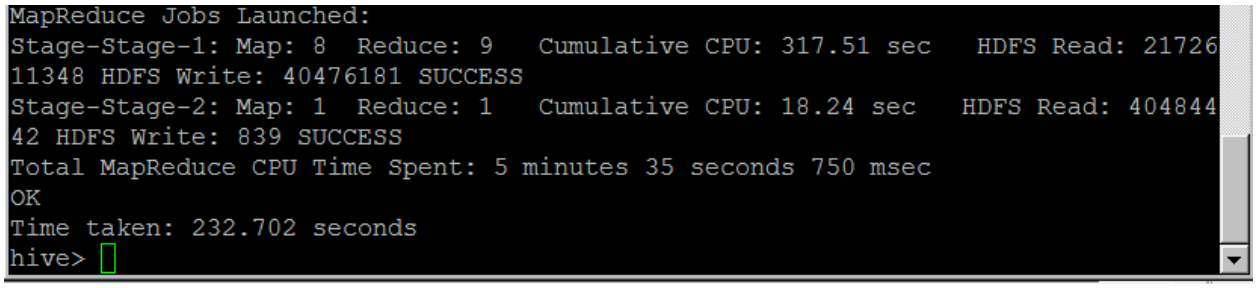
Analysis 2: Which taxi-ids generate the highest fare with the respective number of rides?

This analysis is carried out to find the top performing people in Taxi industry in Chicago 2016. In this query each taxi id is associated with a person's license number. The query groups the taxi id, pickup_community_area and dropoff_community_area by total income and total number of rides for all month in the table called taxi. The data is sorted in descending order and will give the top 20 taxi ids.

Copy and paste the following code in hive shell. The code will take about 30 seconds to execute.

```
---- Create the table taxi to extract the

create table if not exists taxi as select taxi_id, count(taxi_id) as cnt, pickup_community_area,
count(pickup_community_area) as total_pickup, dropoff_community_area,
count(dropoff_community_area ) as dropoff, SUM(trip_total) as sum from area group by taxi_id,
pickup_community_area, dropoff_community_area order by sum desc limit 20;
```

After the query is executed, you should get the following output on the hive shell.



In the hive shell, we need to check if taxi is shown.

```
hive> show tables;

 ---- check the data of table taxi

hive> select * from taxi limit 10;
```

You should get the following output after executing these codes.

```
hive> show tables;
OK
area
company2
taxi
Time taken: 0.078 seconds, Fetched: 3 row(s)
hive> select * from taxi limit 10;
OK
6818    10770   NULL    0       NULL    0       142615.94998073578
3747    3614    NULL    0       NULL    0       130969.80994129181
3237    10050   NULL    0       NULL    0       129518.95002222061
4986    5322    NULL    0       NULL    0       123320.54004955292
8028    7438    NULL    0       NULL    0       123178.91998624802
7118    8412    NULL    0       NULL    0       122477.53996062279
3869    9647    NULL    0       NULL    0       121743.54999017715
5372    5924    NULL    0       NULL    0       121136.36999177933
8164    9420    NULL    0       NULL    0       120320.47002530098
4870    8486    NULL    0       NULL    0       119089.25
Time taken: 0.126 seconds, Fetched: 10 row(s)
hive>
```

Analysis 3: What is the total monthly comparison between Credit and Cash payments?

This analysis is performed to show the comparison between credit card payment and cash payments for taxi travel for each month in Chicago 2016. The query grouped the payment type by total fare for each month. We have analyzed the data separately for each month.

In the hive shell, we need to copy and paste the following HiveQL code to create an external table "payment".

Note: Make sure to replace "hbatra" with your account name in the following HQL code.

```
CREATE EXTERNAL TABLE IF NOT EXISTS payment

(trip_id int, trip_start_timestamp timestamp, trip_end_timestamp timestamp, trip_seconds Int,
trip_miles float, pickup_census_tract Int, dropoff_census_tract Int, pickup_community_area Int,
dropoff_community_area Int, fare float, tips float, tolls float, extras float, trip_total float,
payment_type String, company Int, pickup_latitude Int, pickup_longitude Int, dropoff_latitude Int,
dropoff_longitude Int)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE LOCATION '/user/hbatra/project'

TBLPROPERTIES("skip.header.line.count"="1");
```

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS payment(trip_id int, trip_start_timest
amp timestamp, trip_end_timestamp timestamp, trip_seconds Int, trip_miles float,
 pickup_census_tract Int, dropoff_census_tract Int, pickup_community_area Int, d
ropoff_community_area Int, fare float, tips float, tolls float, extras float, tr
ip_total float, payment_type String, company Int, pickup_latitude Int, pickup_lo
ngitude Int, dropoff_latitude Int, dropoff_longitude Int) ROW FORMAT DELIMITED F
IELDS TERMINATED BY ',' STORED AS TEXTFILE LOCATION '/user/hbatra/project' TBLPR
OPERTIES("skip.header.line.count"="1");
OK
Time taken: 15.691 seconds
hive>
```

To load the data into hive table use the LOAD DATA command. The next few command will lead you through the process of loading the file into hive table.

1. Use the LOAD DATA command statement in Hive to load the file chicago_taxi_trips_2016_01 Into the area table.

```
LOAD DATA INPATH '/user/hbatra/chicago/chicago_taxi_trips_2016_01.csv' OVERWRITE INTO TABLE payment;
```

```
hive> LOAD DATA INPATH '/user/hbatra/chicago/chicago_taxi_trips_2016_01.csv' OVERWRITE INTO TABLE payment;
Loading data to table default.payment
Table default.payment stats: [numFiles=1, numRows=0, totalSize=184423109, rawDataSize=0]
OK
Time taken: 1.783 seconds
hive>
```

2. The LOAD DATA INPATH command moves the file to the table's directory. Verify that the file is no longer present in the original directory.

```
hdfs dfs -ls chicago/chicago_taxi_trips_2016_01.csv
```

```
-bash-4.1$ hdfs dfs -ls chicago/chicago_taxi_trips_2016_01.csv
ls: `chicago/chicago_taxi_trips_2016_01.csv': No such file or directory
-bash-4.1$
```

3. Now we have to create an external table payment_1 that is a structured table format for the 1st month of Chicago data

```
Drop table if exists payment_1;
```

```
Create external table payment_1
(payment_type String,fare float)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION '/user/hbatra/payment_1/';
```

```
hive> Drop table if exists payment_1;
OK
Time taken: 0.114 seconds
hive> create external table payment_1 (payment_type String,fare float)ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED
 AS TEXTFILE LOCATION '/user/hbatra/payment_1/';
OK
Time taken: 0.159 seconds
hive>
```

4. Now you take the data from payment to payment_1 using select statement as follows:

```
INSERT OVERWRITE TABLE payment_1

Select payment_type,sum(fare) as Total_fare from payment group by payment_type,order by
Total_fare;
```

```
Table default.payment_1 stats: [numFiles=1, numRows=7, totalSize=123, rawDataSize=116]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 18.53 sec   HDFS Read: 184563929 HDFS Write: 327 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 4.39 sec   HDFS Read: 5595 HDFS Write: 197 SUCCESS
Total MapReduce CPU Time Spent: 22 seconds 920 msec
OK
Time taken: 63.427 seconds
hive>
```

In the hive you can check the content for payment_1 and payment table. You can compare the content of both the tables and think about why payment_1 is better for querying.

```
Hive> show tables;

Select * from payment_1 limit 10;

Select * from payment limit 10;
```

```
hive> show tables;
OK
payment
payment_1
Time taken: 0.63 seconds, Fetched: 2 row(s)
hive> select * from payment_1 limit 10;
OK
Prcard  2259.41
Pcard   4147.9
Dispute 11615.65
Unknown 33465.16
No Charge       122957.08
Cash    1.0177316E7
Credit Card     1.2085902E7
Time taken: 0.492 seconds, Fetched: 7 row(s)
hive> select * from payment limit 10;
OK
85      2016-01-13 06:15:00     2016-01-13 06:15:00     180     0.4     NULL    NULL    24      24      4.5   0
.0      0.0     0.0     4.5     Cash    107     199     510     199     510
2776    2016-01-22 09:30:00     2016-01-22 09:45:00     240     0.7     NULL    NULL    NULL    NULL    4.45  4
.45     0.0     0.0     8.9     Credit Card     NULL    NULL    NULL    NULL    NULL
3168    2016-01-31 21:30:00     2016-01-31 21:30:00     0       0.0     NULL    NULL    NULL    NULL    42.75 5
.0      0.0     0.0     47.75   Credit Card     119     NULL    NULL    NULL    NULL
4237    2016-01-23 17:30:00     2016-01-23 17:30:00     480     1.1     NULL    NULL    6       6       7.0   0
.0      0.0     0.0     7.0     Cash    NULL    686     500     686     500
5710    2016-01-14 05:45:00     2016-01-14 06:00:00     480     2.71    NULL    NULL    32      NULL    10.25 0
.0      0.0     0.0     10.25   Cash    NULL    385     478     NULL    NULL
1987    2016-01-08 18:15:00     2016-01-08 18:45:00     1080    6.2     NULL    NULL    8       3       17.75 0
.0      0.0     0.0     17.75   Cash    NULL    599     346     660     120
4986    2016-01-14 04:30:00     2016-01-14 05:00:00     1500    18.4    NULL    NULL    NULL    NULL    45.0  1
2.0     0.0     0.0     57.0    Credit Card     NULL    NULL    NULL    NULL    NULL
6400    2016-01-26 04:15:00     2016-01-26 04:15:00     60      0.2     NULL    NULL    16      16      3.75  0
.0      0.0     0.0     3.75    Cash    107     527     24      527     24
7418    2016-01-22 11:30:00     2016-01-22 11:45:00     180     0.0     NULL    504     8       32      5.0   2
.0      0.0     1.5     8.5     Credit Card     82      210     470     744     605
6450    2016-01-07 21:15:00     2016-01-07 21:15:00     0       0.0     NULL    NULL    NULL    NULL    3.25  0
.0      0.0     1.5     4.75    Cash    NULL    NULL    NULL    NULL    NULL
Time taken: 0.083 seconds, Fetched: 10 row(s)
hive>
```

The above analysis is for only one month. Repeat the same for rest 11 months and then combine the data in excel.

Analysis 4: What is the total number of monthly pick-ups in Chicago?

This analysis is performed to show in which month the frequency of taxi travel is maximum in Chicago in 2016. In the query we will count the trip ID for each month. We have analyzed the data separately for each month.

Since we are moving the data from hdfs to hive table. We have to again put all the files in Chicago directory. The files are already in Hadoop file system so can we directly put them in Chicago directory using the command below:

```
Hdfs dfs –put *.csv chicago
```

```
-bash-4.1$  hdfs dfs -put *.csv chicago
-bash-4.1$
```

In the hive shell, we need to copy and paste the following HiveQL code to create an external table "payment".

Note: Make sure to replace "hbatra" with your account name in the following HQL code.

```
CREATE EXTERNAL TABLE IF NOT EXISTS frequency

(trip_id int, trip_start_timestamp timestamp, trip_end_timestamp timestamp, trip_seconds Int, trip_miles float, pickup_census_tract Int, dropoff_census_tract Int, pickup_community_area Int, dropoff_community_area Int, fare float, tips float, tolls float, extras float, trip_total float, payment_type String, company Int, pickup_latitude Int, pickup_longitude Int, dropoff_latitude Int, dropoff_longitude Int)

 ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE LOCATION '/user/hbatra/project1'

TBLPROPERTIES("skip.header.line.count"="1");
```

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS frequency
    > (trip_id int, trip_start_timestamp timestamp, trip_end_timestamp timestamp, trip_seconds Int, trip_miles float, pickup_censu
tract Int, dropoff_census_tract Int, pickup_community_area Int, dropoff_community_area Int, fare float, tips float, tolls float, e
ras float, trip_total float, payment_type String, company Int, pickup_latitude Int, pickup_longitude Int, dropoff_latitude Int, di
off_longitude Int)
    >  ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE LOCATION '/user/hbatra/project1'
    > TBLPROPERTIES("skip.header.line.count"="1");
OK
Time taken: 6.572 seconds
hive>
```

Then load the data into hive table use the LOAD DATA command. The next few command will lead you through the process of loading the file into hive table.

1. Use the LOAD DATA command statement in Hive to load the file chicago_taxi_trips_2016_01 Into the area table.

```
LOAD DATA INPATH '/user/hbatra/chicago/chicago_taxi_trips_2016_01.csv' OVERWRITE INTO TABLE frequency;
```

```
hive> LOAD DATA INPATH '/user/hbatra/chicago/chicago_taxi_trips_2016_01.csv' OVERWRITE INTO TABLE frequency;
Loading data to table default.frequency
Table default.frequency stats: [numFiles=1, numRows=0, totalSize=184423109, rawDataSize=0]
OK
Time taken: 0.226 seconds
hive>
```

2. The LOAD DATA INPATH command moves the file to the table's directory. Verify that the file is no longer present in the original directory.

hdfs dfs -ls chicago/chicago_taxi_trips_2016_01.csv

```
-bash-4.1$ hdfs dfs -ls chicago/chicago_taxi_trips_2016_01.csv
ls: `chicago/chicago_taxi_trips_2016_01.csv': No such file or directory
-bash-4.1$
```

3. Now we have to create an external table frequency_1 that is a structured table format for the 1st month of Chicago data.

Drop table if exists frequency_1;

Create external table frequency_1
(trip_id int)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION '/user/hbatra/frequency_1/';

```
hive> Drop table if exists frequency_1;
OK
Time taken: 0.103 seconds
hive> Create external table frequency_1
    > (trip_id int)
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE LOCATION '/user/hbatra/frequency_1/';
OK
Time taken: 0.073 seconds
hive>
```

4. Now you take the data from payment to payment_1 using select statement as follows:

```
INSERT OVERWRITE TABLE frequency_1 Select count(trip_id) from frequency;
```

```
hive> INSERT OVERWRITE TABLE frequency_1 Select count(trip_id) from frequency;
Query ID = hbatra_20171210171545_708ae679-745a-4eec-9f1c-9160941f9623
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1512862017005_0003, Tracking URL = http://bi-hadoop-prod-4130.bi.services.us-
tion_1512862017005_0003/
Kill Command = /usr/iop/4.2.0.0/hadoop/bin/hadoop job  -kill job_1512862017005_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-12-10 17:16:07,292 Stage-1 map = 0%,  reduce = 0%
2017-12-10 17:16:12,988 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.66 sec
2017-12-10 17:16:18,578 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 7.02 sec
MapReduce Total cumulative CPU time: 7 seconds 20 msec
Ended Job = job_1512862017005_0003
Loading data to table default.frequency_1
Table default.frequency_1 stats: [numFiles=1, numRows=1, totalSize=8, rawDataSize=7]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.02 sec   HDFS Read: 184564845 HDFS Write: 8
Total MapReduce CPU Time Spent: 7 seconds 20 msec
OK
Time taken: 35.341 seconds
hive>
```

In the hive you can check the content for frequency_1 and frequency table. You can compare the content of both the tables and think about why frequency_1 is better for querying.

```
Hive> show tables;

Select * from frequency_1;

Select * from frequency limit 5;
```

```
hive> Select * from frequency_1;
OK
1705782
Time taken: 0.136 seconds, Fetched: 1 row(s)
hive> Select * from frequency limit 5;
OK
85      2016-01-13 06:15:00     2016-01-13 06:15:00     180     0.4     NULL    NULL    24      24      4.5     0.0     0.0     0.04
.5      Cash    107     199     510     199     510
2776    2016-01-22 09:30:00     2016-01-22 09:45:00     240     0.7     NULL    NULL    NULL    NULL    4.45    4.45    0.0     0.08
.9      Credit Card     NULL    NULL    NULL    NULL    NULL
3168    2016-01-31 21:30:00     2016-01-31 21:30:00     0       0.0     NULL    NULL    NULL    NULL    42.75   5.0     0.0     0.04
7.75    Credit Card     119     NULL    NULL    NULL    NULL
4237    2016-01-23 17:30:00     2016-01-23 17:30:00     480     1.1     NULL    NULL    6       6       7.0     0.0     0.0     0.07
.0      Cash    NULL    686     500     686     500
5710    2016-01-14 05:45:00     2016-01-14 06:00:00     480     2.71    NULL    NULL    32      NULL    10.25   0.0     0.0     0.01
0.25    Cash    NULL    385     478     NULL    NULL
Time taken: 0.22 seconds, Fetched: 5 row(s)
hive>
```

The above analysis is for only one month. Repeat the same for rest 11 months and then combine the data in excel.

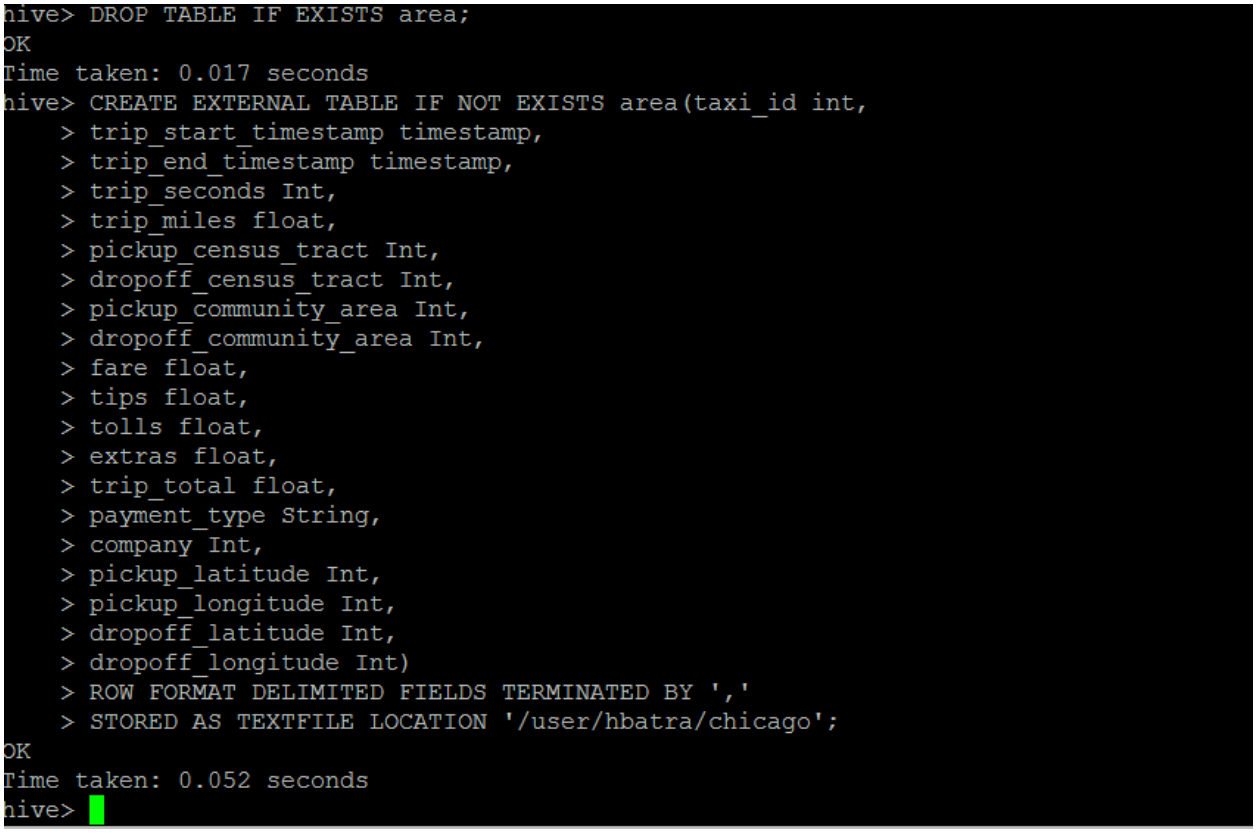
Analysis 5: Which communities in Chicago have the highest demand for pick-ups and Drop-offs?

This Analysis is performed to see the top 5 communities of Chicago with highest pick-ups and drop-offs in 2016. The query groups the pick-up community area and drop-off community area for all the months by the communities in a table community. The communities are sorted out in descending order and top 5 communities are extracted from the dataset.

In the hive shell, we need to copy and paste the following HiveQL code to create an external table “area”.

Note: Make sure to replace “hbatra” with your account name in the following HQL code.

```
DROP TABLE IF EXISTS area;

CREATE EXTERNAL TABLE IF NOT EXISTS area(taxi_id int,
trip_start_timestamp timestamp,
trip_end_timestamp timestamp,
trip_seconds Int,
trip_miles float,
pickup_census_tract Int,
dropoff_census_tract Int,
pickup_community_area Int,
dropoff_community_area Int,
fare float,
tips float,
tolls float,
extras float,
trip_total float,
payment_type String,
company Int,
pickup_latitude Int,
pickup_longitude Int,
dropoff_latitude Int,
dropoff_longitude Int)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION '/user/hbatra/chicago';
```



Copy and paste the following code in hive shell. The code will take about 30 seconds to execute.

---- Create the table pick_drop to extract the data for top 20 communities.

```
CREATE TABLE IF NOT EXISTS Pick_Drop as select pickup_community_area,
count(pickup_community_area ) as Total_Community_Count, dropoff_community_area,
count(dropoff_community_area ) as Total_Dropoff_Count from area GROUP BY
pickup_community_area ,dropoff_community_area ORDER BY Total_Community_Count DESC
LIMIT 20;
```

After the query is executed, you should get the following output on the hive shell.

```
2017-12-10 18:17:44,074 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 3.62 sec
MapReduce Total cumulative CPU time: 3 seconds 620 msec
Ended Job = job_1512862017005_0005
Moving data to: hdfs://bi-hadoop-prod-4130.bi.services.us-south.bluemix.net:8020/apps/hive/warehouse/pick_drop
Table default.pick_drop stats: [numFiles=1, numRows=20, totalSize=383, rawDataSize=363]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 7  Reduce: 8   Cumulative CPU: 151.93 sec   HDFS Read: 1988032331 HDFS Write: 102734 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 3.62 sec   HDFS Read: 110097 HDFS Write: 457 SUCCESS
Total MapReduce CPU Time Spent: 2 minutes 35 seconds 550 msec
OK
Time taken: 116.835 seconds
hive>
```

In the hive shell, we need to check if pick_drop is shown.

```
hive> show tables;

 ---- check the data of table pick_drop

hive> select * from pick_drop limit 5;
```

You should get the following output after executing these codes.

```
hive> show tables;
OK
area
frequency
frequency_1
payment
payment_1
pick_drop
Time taken: 0.029 seconds, Fetched: 6 row(s)
hive> select * from pick_drop limit 5;
OK
8       2003161 8       2003161
32      1342191 8       1342191
8       1314026 32      1314026
32      990121  32      990121
32      544027  28      544027
Time taken: 0.235 seconds, Fetched: 5 row(s)
hive>
```

Create Pig Table to Query and Analyze the Chicago Dataset

Analysis 6: Which communities of Chicago has the maximum miles travelled by taxi in 2016?

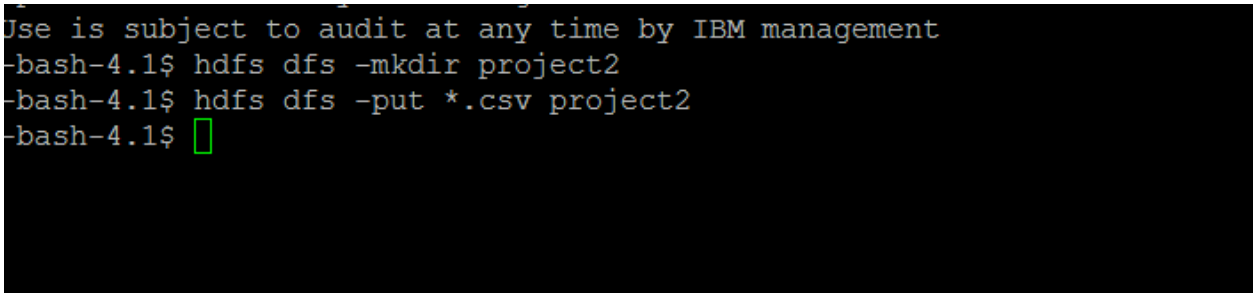
This analysis is performed to show in which community the frequency of taxi travel is maximum in Chicago 2016. The query is written in pig and groups the pick-up community area for all the months by the total number of miles. The data is then stored in the table summiles.

We will create a directory project2.

Hdfs dfs –mkdir project2

Run the following hdfs command to upload the files.

```
Hdfs dfs –put *.csv project2
```



The code for extraction has been wriiten in pig script.

```
Vi summiles.pig
```

```
chicagoNew = load '/user/hbatra/project2/' USING PigStorage(',') AS (taxi_id:Int, trip_start_timestamp:chararray, tri
p_end_timestamp:chararray, trip_seconds:Int, trip_miles:float, pickup_census_tract:Int, dropoff_census_tract:Int, pic
kup_community_area:Int, dropoff_community_area:Int, fare:float, tips:float, tolls:float, extras:float, trip_total:flo
at, payment_type:chararray, company:Int, pickup_latitude:Int, pickup_longitude:Int, dropoff_latitude:Int, dropoff_lon
gitude:Int);

describe chicagoNew ;

grpmiles = GROUP chicagoNew by pickup_community_area;

summiles = FOREACH grpmiles GENERATE group as pickup_community_area, SUM(chicagoNew.trip_miles) as sum_tripmiles;

STORE summiles INTO '/user/hbatra/output/Summiles_community' USING PigStorage(',');
~
~
~
~
```

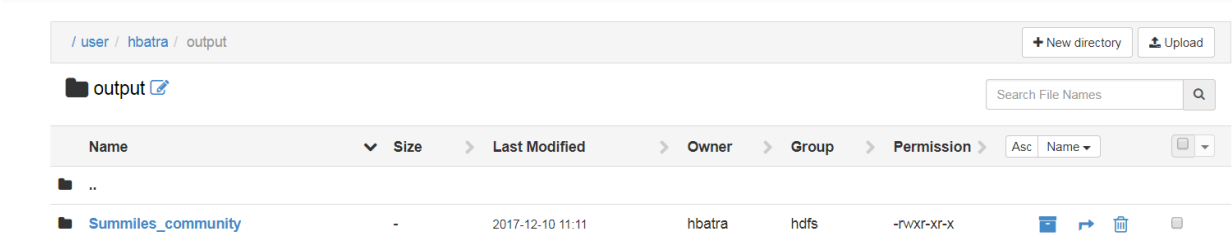
Now we will save the script by entering **escape :wq** and run the script with below command. To execute the job it will take a minute to complete the job. If the job is successfully completed you will get a SUCCESS at the end.

Pig summiles.pig

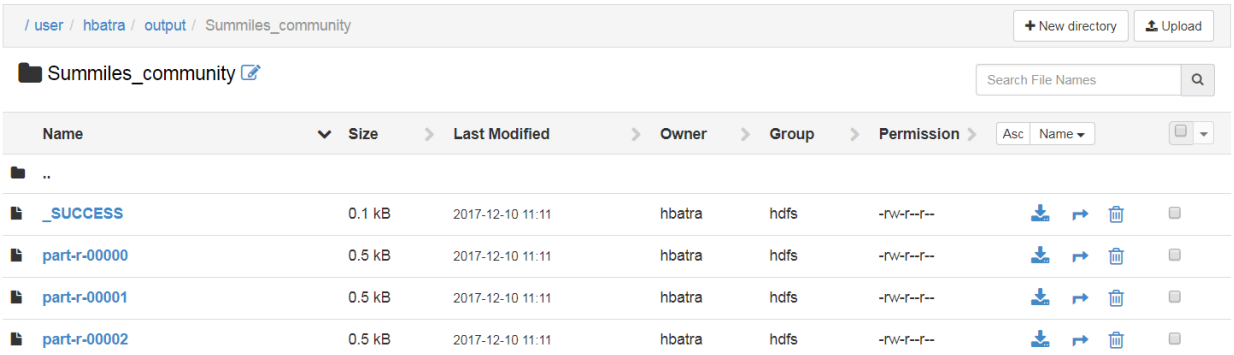
```
2017-12-10 19:11:49,576 [main] INFO  org.apache.hadoop.yarn.client.api.impl.TimelineClientImpl - Timeline service add
ress: http://bi-hadoop-prod-4130.bi.services.us-south.bluemix.net:8188/ws/v1/timeline/
2017-12-10 19:11:49,576 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at bi-hado
op-prod-4130.bi.services.us-south.bluemix.net/173.16.127.1:8050
2017-12-10 19:11:49,581 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed.
 FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2017-12-10 19:11:49,667 [main] WARN  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher -
 Encountered Warning ACCESSING_NON_EXISTENT_FIELD 366 time(s).
2017-12-10 19:11:49,667 [main] WARN  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher -
 Encountered Warning FIELD_DISCARDED_TYPE_CONVERSION_FAILED 234 time(s).
2017-12-10 19:11:49,668 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher -
 Success!
2017-12-10 19:11:49,700 [main] INFO  org.apache.pig.Main - Pig script completed in 2 minutes, 34 seconds and 560 mill
iseconds (154560 ms)
-bash-4.1$
```

Once the job is complete open Ambari and go to **File Browser** and look for the newly created folder **Output** under user/hbatra.

Click on **Output** folder you will find a sub folder name **Summiles_community**

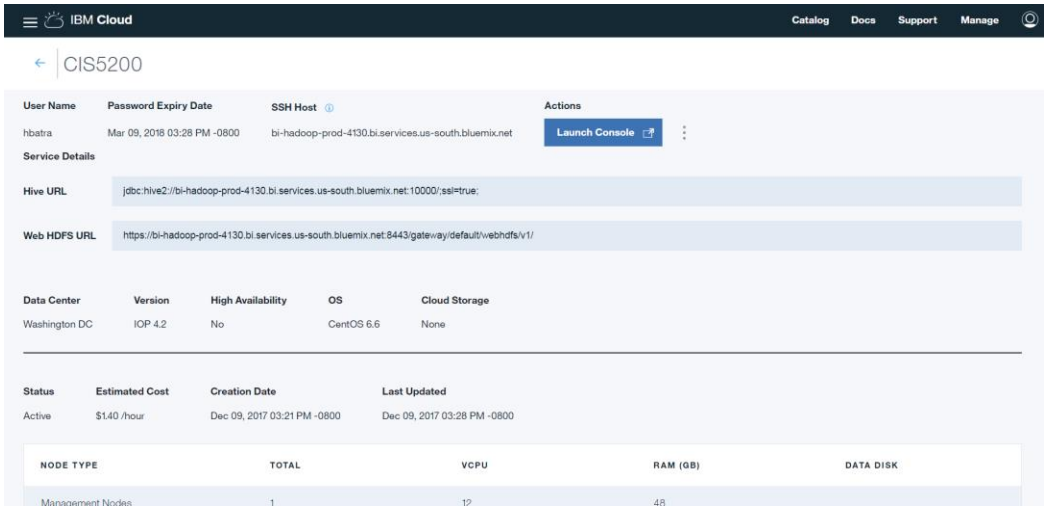


Click on the folder **Summiles_community** you will see the output files part-r-0000, part-r-0001 and part-r-0002.



Loading the Data into Excel:

Now go to the BigInsights cluster page to open Ambari web page.



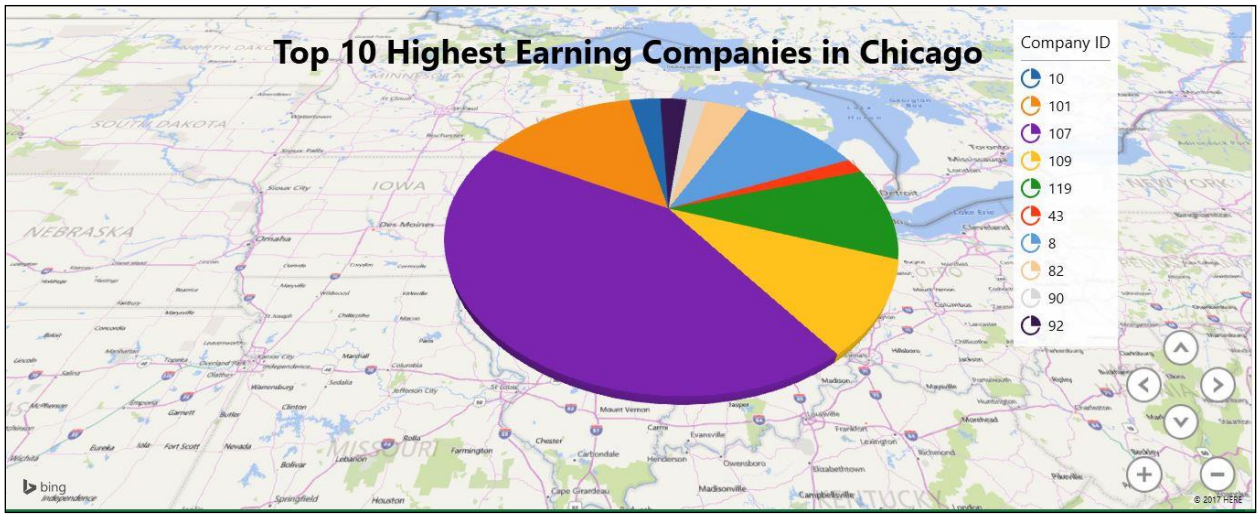
And, you need to go and open “File Browser” to find out the analyzed data in the respective directories as mentioned in the codes. You need to click on the file to download to your local computer in order to open it in Excel.

You have to open the file in Excel with delimiter (separator) specified while creating the analyzed data file on HDFS.

Analysis 1: Which companies have the highest earnings in Chicago in 2016?

The downloaded file is opened in Excel (delimited by comma) and we have used the 3-D Map Power View column under the Insert tab for visualization. After that select stacked columns.

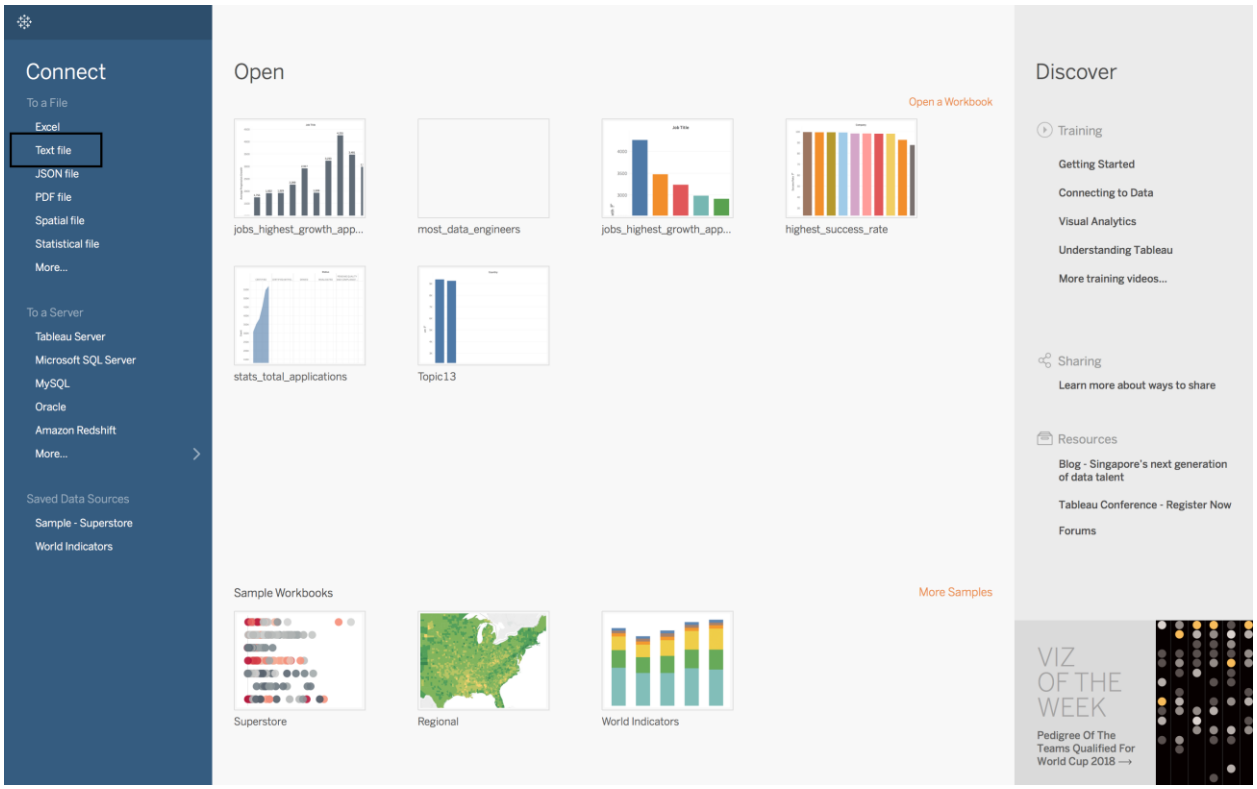
The output is should look like the image below, shown in a 3D map with pie chart to represent the top 10 Highest Taxi Companies in Chicago in 2016.

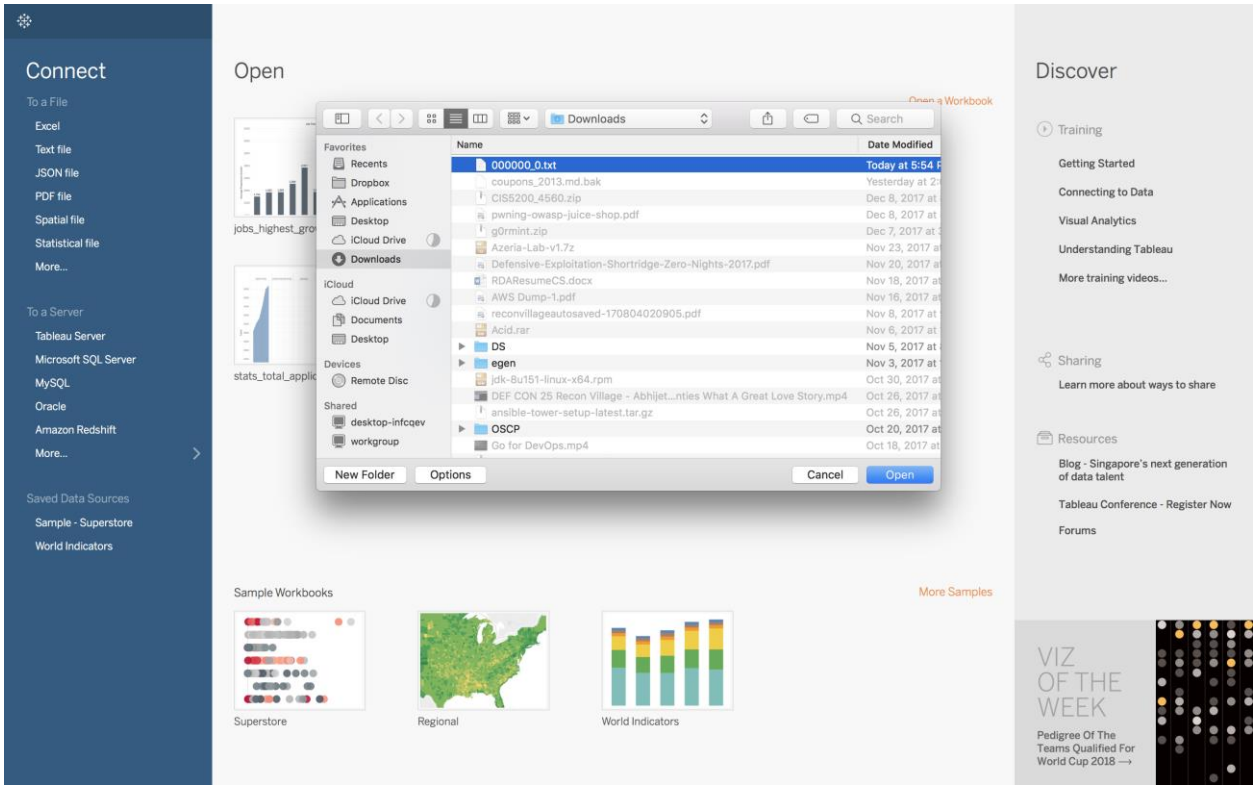


Analysis 2: Which taxi-ids generate the highest fare with the respective number of rides?

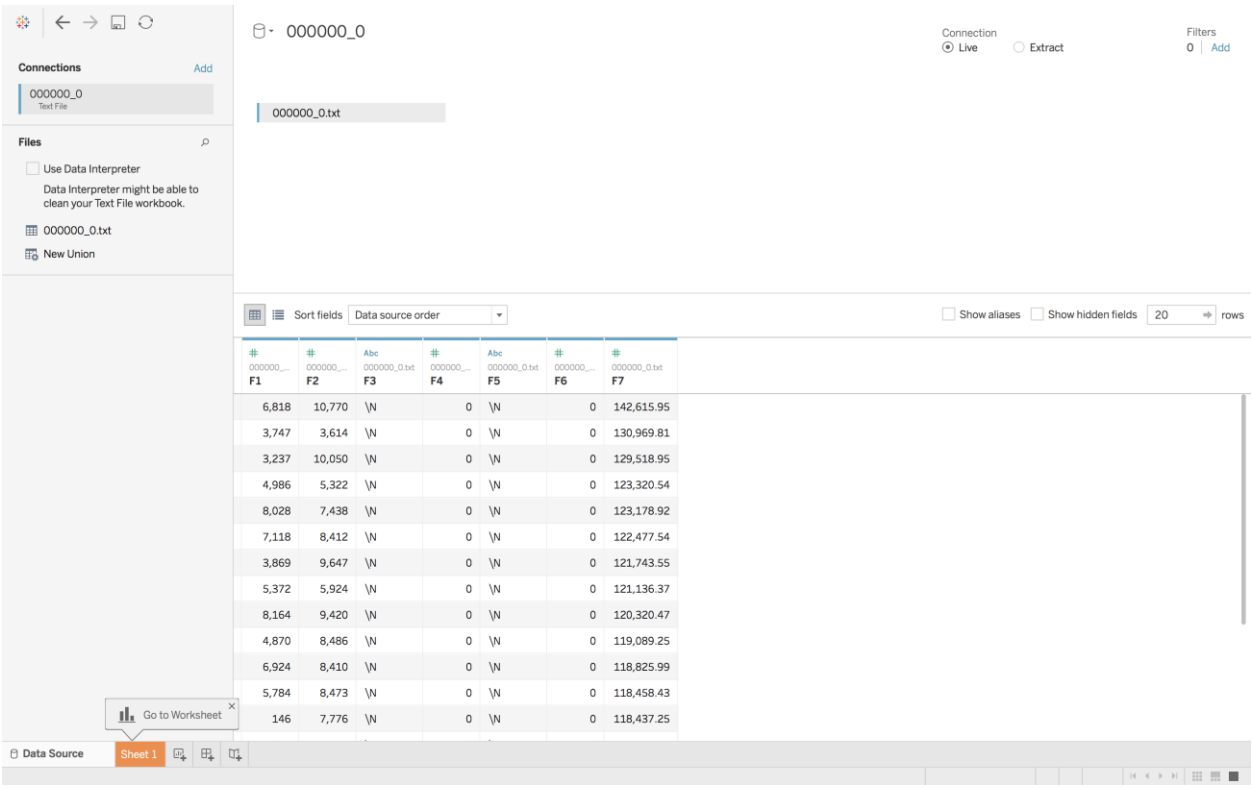
This visualization was done using tableau 10.3. The downloaded excel file was opened in tableau in the following manner:

Open Tableau software, and upload the text file downloaded from File Browser in Ambari, as:

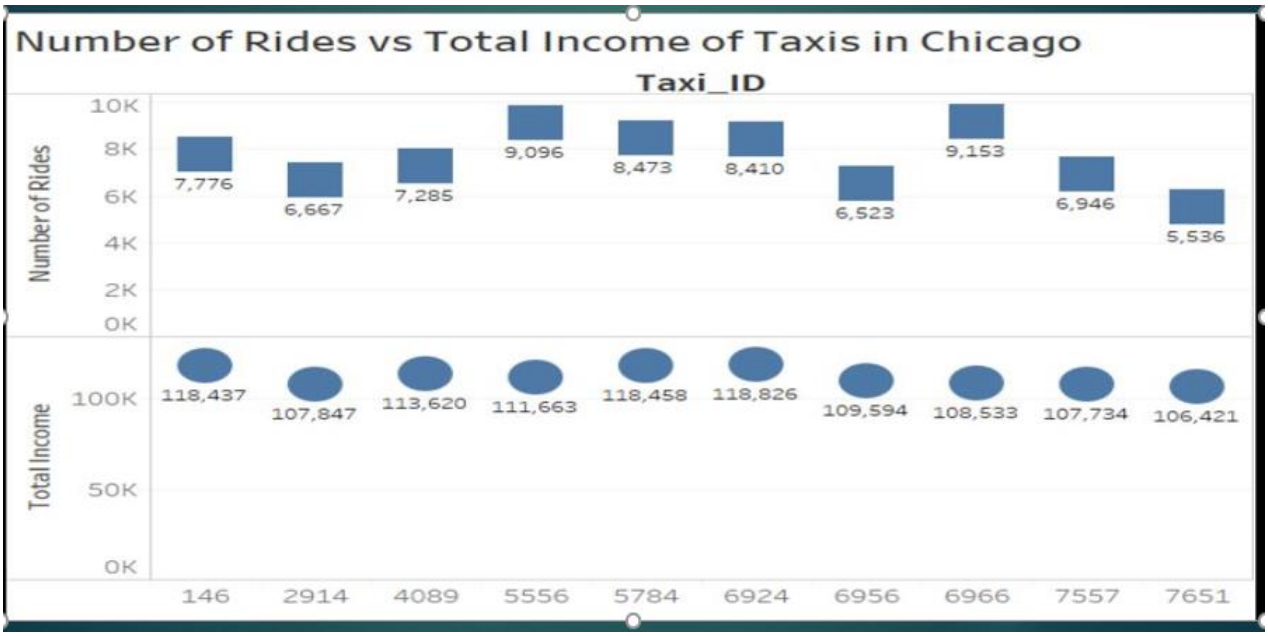




Rename FI as “Taxi ID”, F2 as “Number of Rides”, and F7 as “Total Amount” in the data source of Tableau.

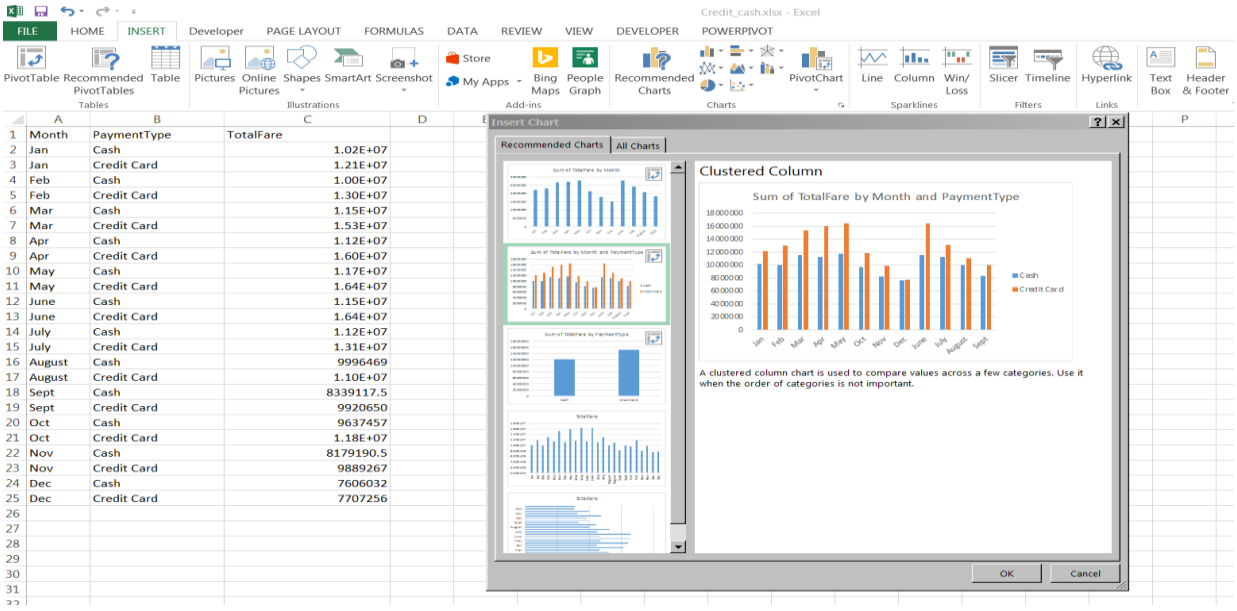


The final output should look like the image shown below (result is filtered in Tableau to get Top 10 Taxi Ids in Chicago):

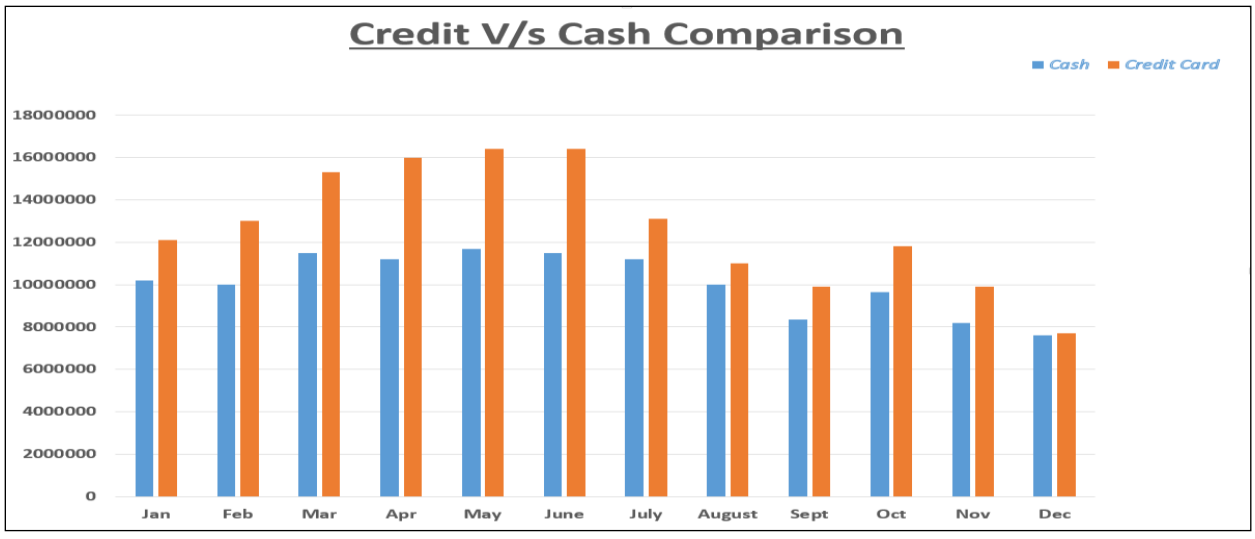


Analysis 3: What is the total monthly comparison between Credit and Cash payments?

The downloaded file is opened in excel and we have used a vertical bar chart to show the comparison between credit card payments and cash card payments for taxi travel for each month in Chicago 2016.



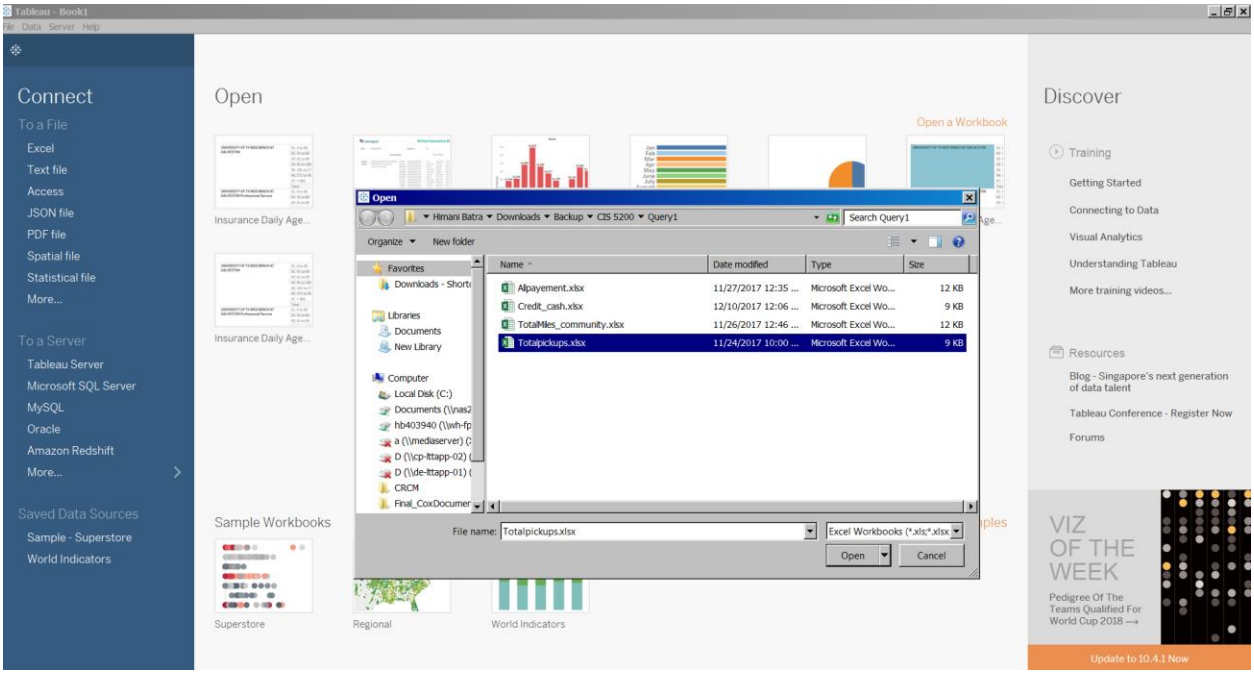
Month	PaymentType	TotalFare
Jan	Cash	1.02E+07
Jan	Credit Card	1.21E+07
Feb	Cash	1.00E+07
Feb	Credit Card	1.30E+07
Mar	Cash	1.15E+07
Mar	Credit Card	1.53E+07
Apr	Cash	1.12E+07
Apr	Credit Card	1.60E+07
May	Cash	1.17E+07
May	Credit Card	1.64E+07
June	Cash	1.15E+07
June	Credit Card	1.64E+07
July	Cash	1.12E+07
July	Credit Card	1.31E+07
August	Cash	9996469
August	Credit Card	1.10E+07
Sept	Cash	8339117.5
Sept	Credit Card	9920650
Oct	Cash	9637457
Oct	Credit Card	1.18E+07
Nov	Cash	8179190.5
Nov	Credit Card	9889267
Dec	Cash	7606032
Dec	Credit Card	7707256



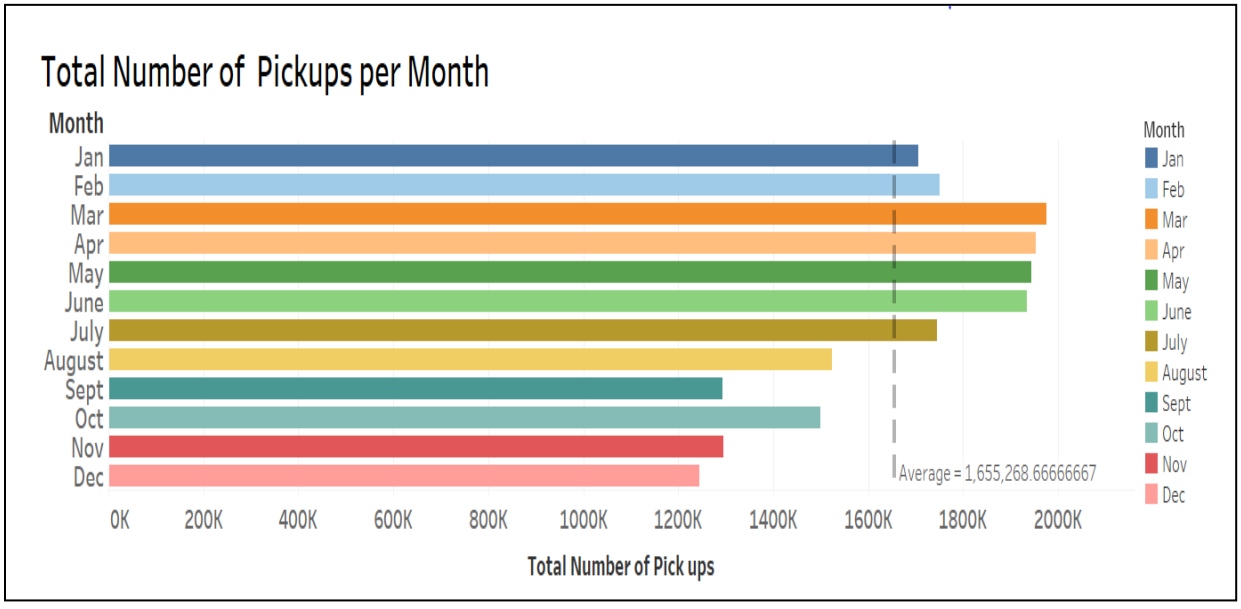
Analysis 4: What is the total number of monthly pick-ups in Chicago?

The desired file was downloaded to Excel and opened in the tableau as shown below. In tableau, we have used the horizontal bar chart to represent the total number of pick-ups for each month.

The below step shows how to import the analyzed excel file to tableau.



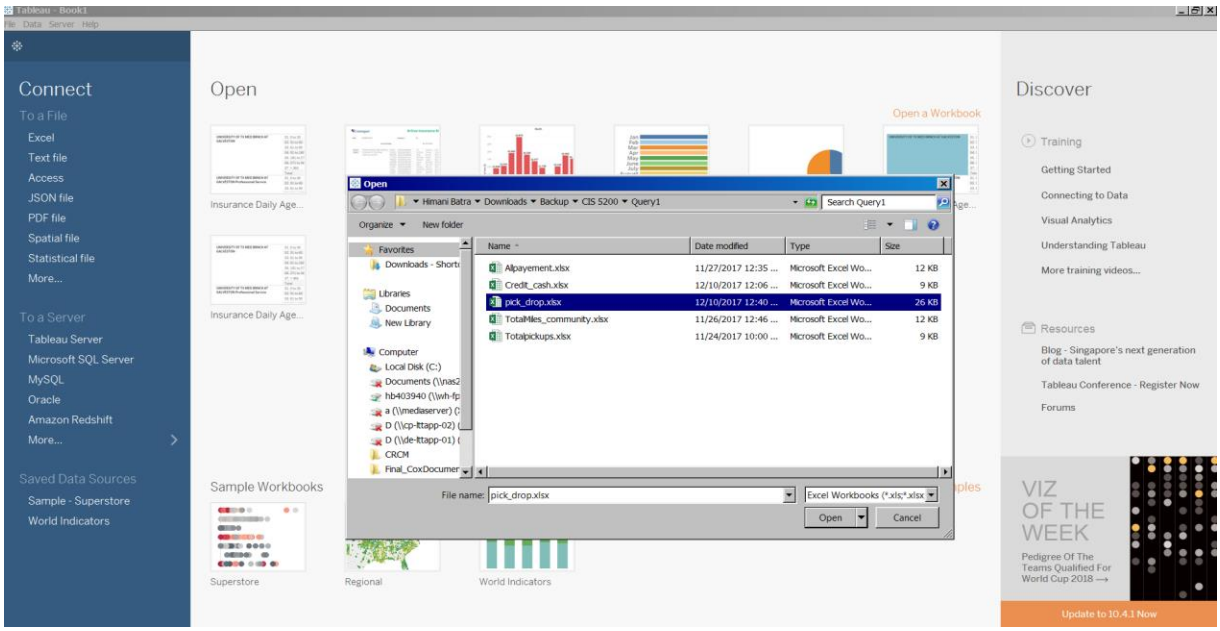
The horizontal bar shows the number of pick-ups are more in summer (March, April, May) as compared to winters (November, December , January). Also, the reference line shows the Average pick-ups is approx. 1,655,268.667.



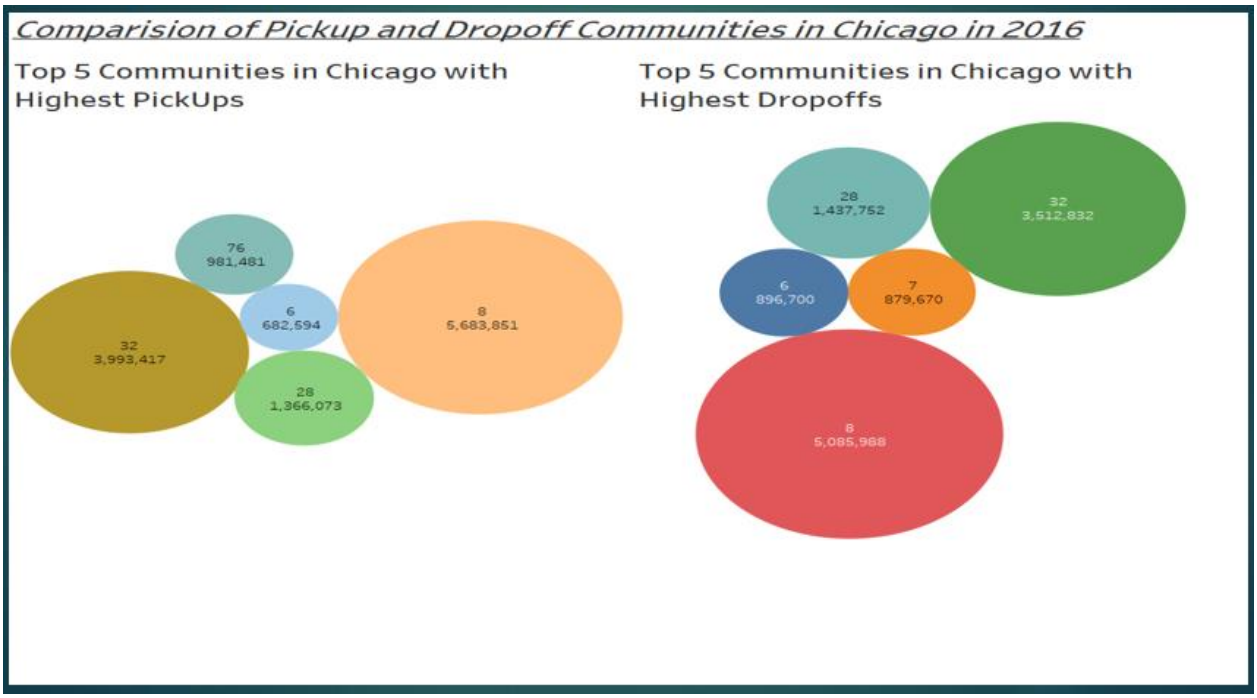
Analysis 5: Which communities in Chicago have the highest demand for pick-ups and Drop-offs?

The desired file was downloaded to Excel and opened in the tableau as shown below. In tableau, we have used the bubble chart to represent the most active community. The bigger the size of the bubble the more pick-ups and drop-offs are happening in that community.

The below step shows how to import the analyzed excel file to tableau

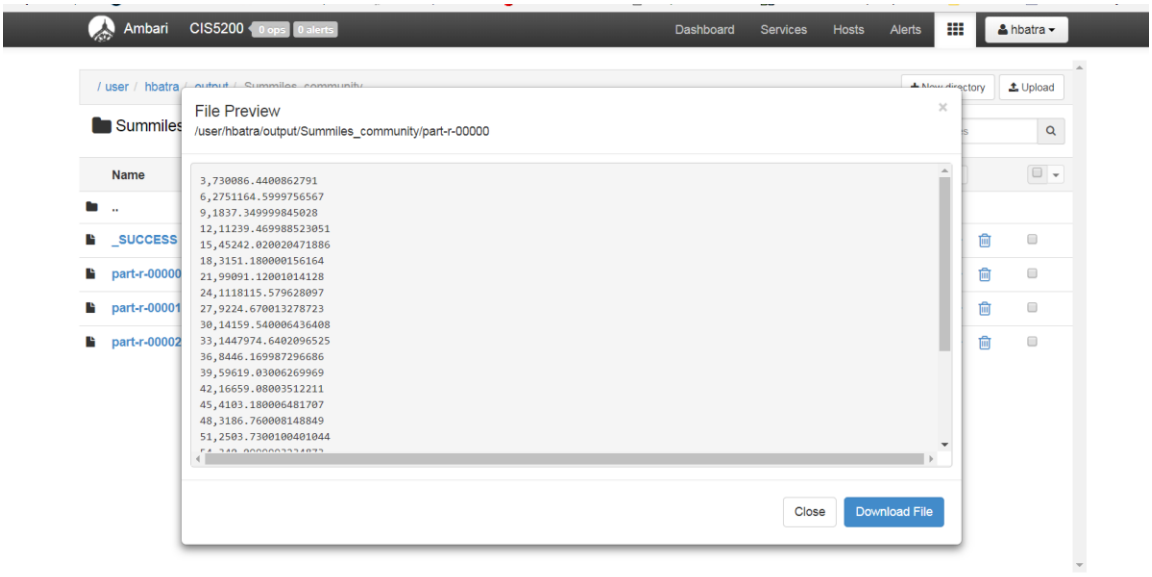


The bubble chart output shows the maximum number of pick-ups and drop-offs are having in community 8.

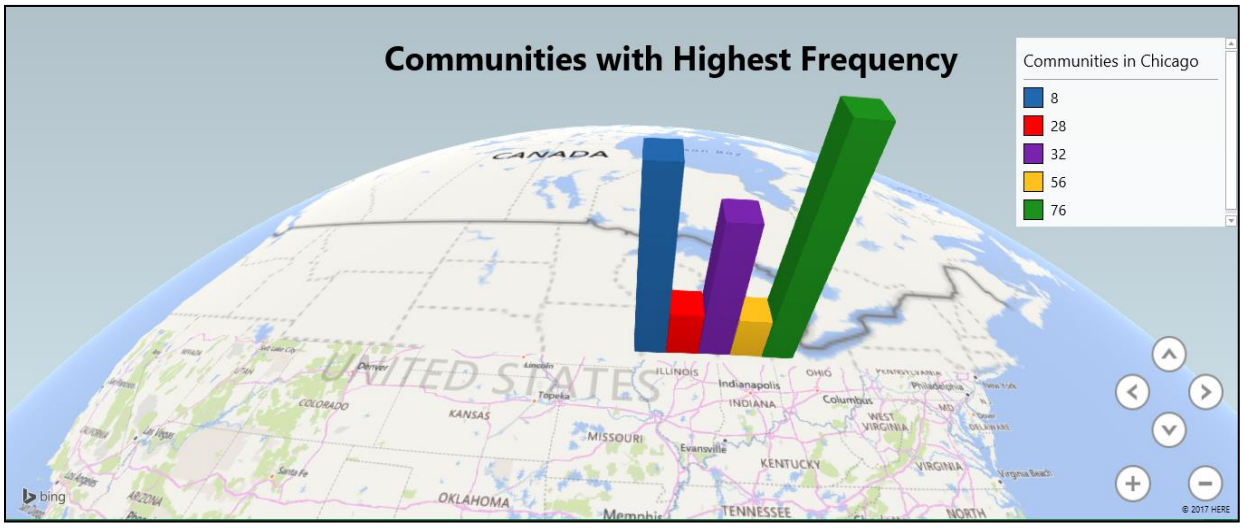


Analyses 6: Which communities of Chicago have the maximum miles travelled by taxi in 2016?

Open the file in Ambari under the folder Output and download the file.



Open the downloaded table in Excel with delimiter (separator) “comma”. Then, go to “Insert” tab to find out the menu “3D Maps” enabled. You need to click and open “3D Map”. If it complaints that 3D Map cannot be open, then you need to make sure if you insert headers into the first row. Now you select columns you want to filter.



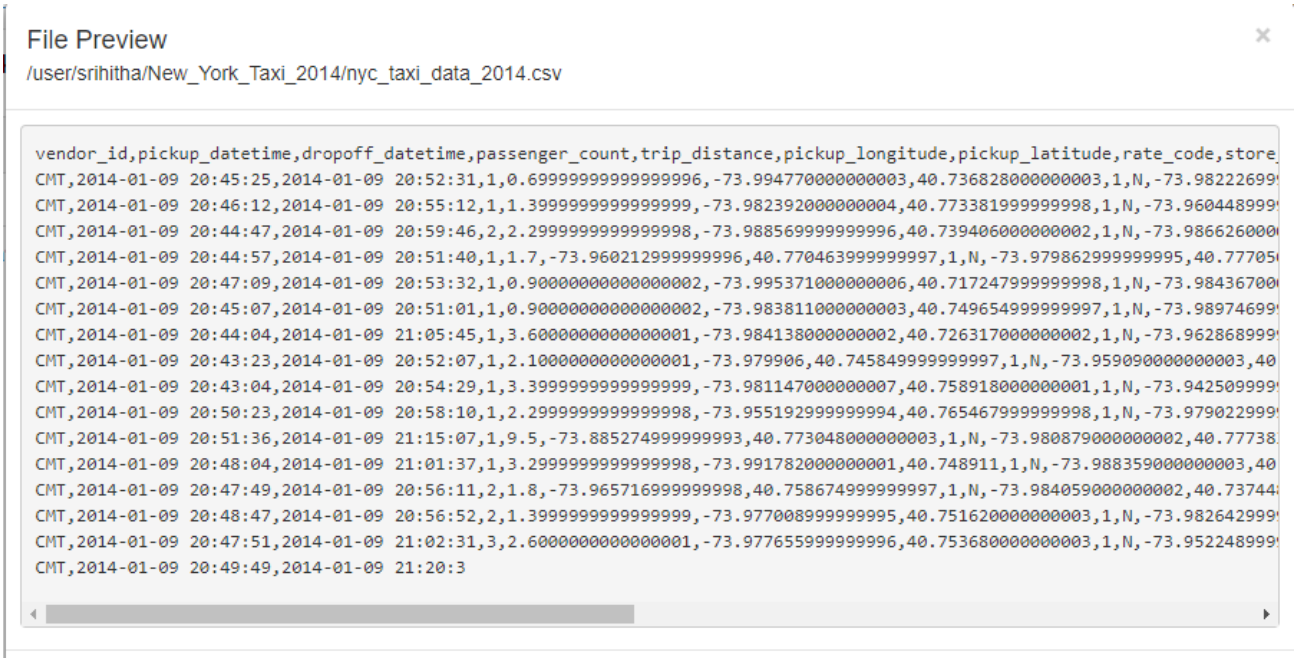
New York Dataset Loaded into Big Insights

Analysis 7: Which is the highest earning vendor in New York?

Navigate to /user/srihitha/New_York_Taxi_2014/nyc_taxi_data_2014.csv to make sure if it has the files uploaded,

hdfs dfs -ls

```
-bash-4.1$ hdfs dfs -ls
Found 1 items
drwxr-xr-x   - srihitha hdfs          0 2017-12-10 22:40 New_York_Taxi_2014
-bash-4.1$
```



Open the Hive interface by typing in `Pig` in the terminal:

```
pig
```

We're now going to create a table from our CSV using a Pig query. Copy and paste the following query to run the command and create the table.

```
NwYrk = LOAD '/user/mahisiri/NewYork/nyc_taxi_data_2014.csv' USING PigStorage(',')
AS (vendor_id:chararray,
pickup_datetime:chararray,
dropoff_datetime:chararray,
passenger_count:int,
trip_distance:long,
pickup_longitude:double,
pickup_latitude:double,
rate_code:int,
store_and_fwd_flag:chararray,
dropoff_longitude:double,
dropoff_latitude:double,
payment_type:chararray,
fare_amount:long,
surcharge:long,
mta_tax:long,
tip_amount:long,
tolls_amount:long,
total_amount:long);
```

DESCRIBE shows what the types of the columns in the table. You can see the columns in the table by executing:

```
DESCRIBE NwYrk;
```

```
grunt> NwYrk = LOAD '/user/srihitha/New_York_Taxi_2014/nyc_taxi_data_2014.csv' USING PigStorage(',')
>> AS (vendor_id:chararray,
>> pickup_datetime:chararray,
>> dropoff_datetime:chararray,
>> passenger_count:int,
>> trip_distance:long,
>> pickup_longitude:double,
>> pickup_latitude:double,
>> rate_code:int,
>> store_and_fwd_flag:chararray,
>> dropoff_longitude:double,
>> dropoff_latitude:double,
>> payment_type:chararray,
>> fare_amount:long,
>> surcharge:long,
>> mta_tax:long,
>> tip_amount:long,
>> tolls_amount:long,
>> total_amount:long);
grunt> DESCRIBE NwYrk;
NwYrk: {vendor_id: chararray,pickup_datetime: chararray,dropoff_datetime: chararray,passenger_count: int,trip_distance: long,pickup_longitude: double,pickup_latitude: d
ouble,rate_code: int,store_and_fwd_flag: chararray,dropoff_longitude: double,dropoff_latitude: double,payment_type: chararray,fare_amount: long,surcharge: long,mta_tax:
 long,tip_amount: long,tolls_amount: long,total_amount: long}
grunt> by_vendorid_travels = GROUP NwYrk BY (vendor_id, total_amount);
grunt> by_vendorid_travels_counts = FOREACH by_vendorid_travels GENERATE
>>     FLATTEN(group) AS (vendor_id, total_amount), COUNT(NwYrk) AS vendor_travel_count;
grunt>
```

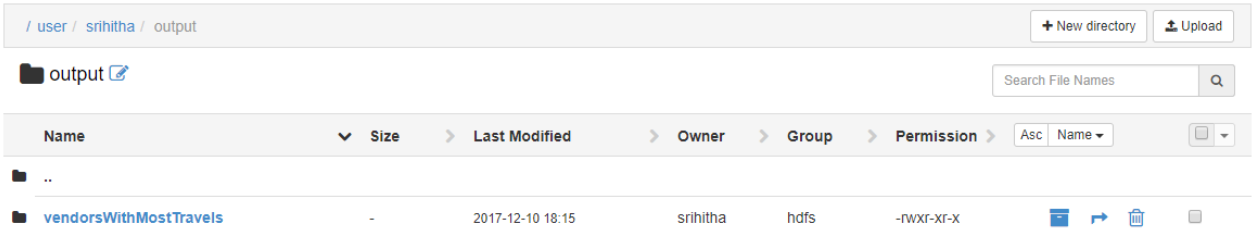
We now have group vendor_id and total_amount to find which vendor has more revenue. In this step, you will use the STORE command to output a relation into a new file in HDFS. Enter the following command to output the crelation to a folder named output/ vendorsWithMostTravels with

The full script should be:

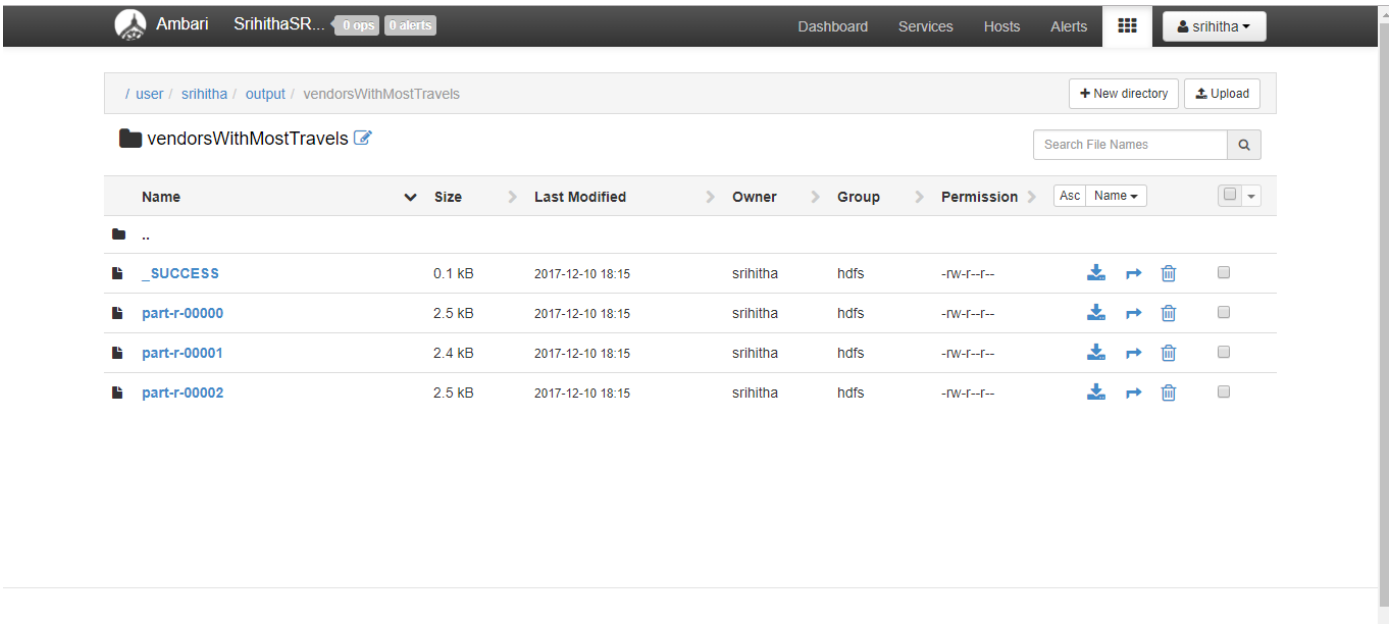
```
by_vendorid_travels = GROUP NwYrk BY (vendor_id, total_amount);

by_vendorid_travels_counts = FOREACH by_vendorid_travels GENERATE
        FLATTEN(group) AS (vendor_id, total_amount), COUNT(NwYrk) AS
vendor_travel_count;

STORE by_vendorid_travels_counts INTO 'output/vendorsWithMostTravels' USING
PigStorage(',');
```

```
Input(s):
Successfully read 15000000 records (2534170220 bytes) from: "/user/srihitha/New_York_Taxi_2014/nyc_taxi_data_2014.csv"

Output(s):
Successfully stored 696 records (7656 bytes) in: "hdfs://bi-hadoop-prod-4220.bi.services.us-south.bluemix.net:8020/user/srihitha/output/vendorsWithMostTravels"

Counters:
Total records written : 696
Total bytes written : 7656
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1512941238126_0002


2017-12-11 02:15:28,555 [main] INFO  org.apache.hadoop.yarn.client.api.impl.TimelineClientImpl - Timeline service address: http://bi-hadoop-prod-4220.bi.services.us-sou
th.bluemix.net:8188/ws/v1/timeline/
2017-12-11 02:15:28,555 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at bi-hadoop-prod-4220.bi.services.us-south.bluemix.net/173.1
6.136.1:8050
2017-12-11 02:15:28,558 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to j
ob history server
2017-12-11 02:15:28,698 [main] INFO  org.apache.hadoop.yarn.client.api.impl.TimelineClientImpl - Timeline service address: http://bi-hadoop-prod-4220.bi.services.us-sou
th.bluemix.net:8188/ws/v1/timeline/
2017-12-11 02:15:28,698 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at bi-hadoop-prod-4220.bi.services.us-south.bluemix.net/173.1
6.136.1:8050
2017-12-11 02:15:28,703 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to j
ob history server
2017-12-11 02:15:28,779 [main] INFO  org.apache.hadoop.yarn.client.api.impl.TimelineClientImpl - Timeline service address: http://bi-hadoop-prod-4220.bi.services.us-sou
th.bluemix.net:8188/ws/v1/timeline/
2017-12-11 02:15:28,780 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at bi-hadoop-prod-4220.bi.services.us-south.bluemix.net/173.1
6.136.1:8050
2017-12-11 02:15:28,783 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to j
ob history server
2017-12-11 02:15:28,810 [main] WARN  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Encountered Warning FIELD_DISCARDED_TYPE_CONVERSIO
N_FAILED 13 time(s).
2017-12-11 02:15:28,811 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
```

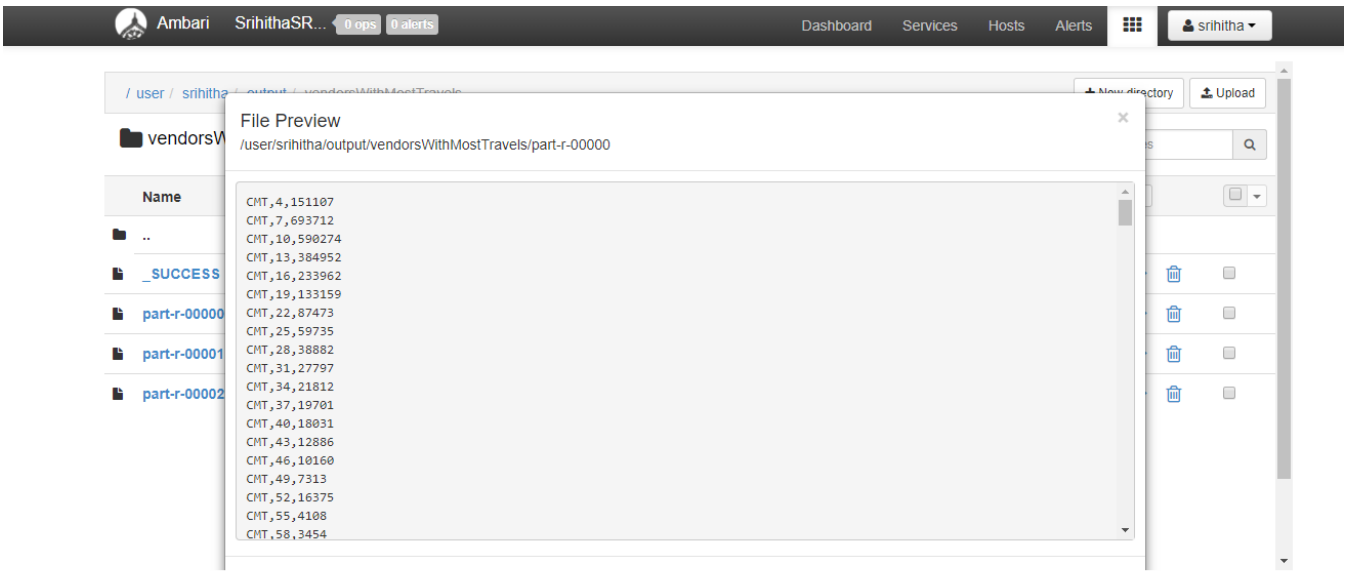
Open **Files Browser** at Ambari and click on "`/user/srihitha/output`" folder. You will find a subfolder named "`vendorsWithMostTravels`".



Click on "`vendorsWithMostTravels`" folder. You will see an output file called "`part-r-00000`", "`part-r-00001`" and "`part-r-00002`":



Click on the file "`part-r-00000`" to see the File Preview:



Then, using "Files View" of Ambari, you can download *000000-0* text file at /user/srihitha/output/vendorsWithMostTravels/part-r-00000 directory of Pig Storage to your desktop computer. You actually can connect from Tableau to connect the cluster to get this file but IBM Analytics Engine does not have the connector, which you tried in Excel as some of you successfully connect it.

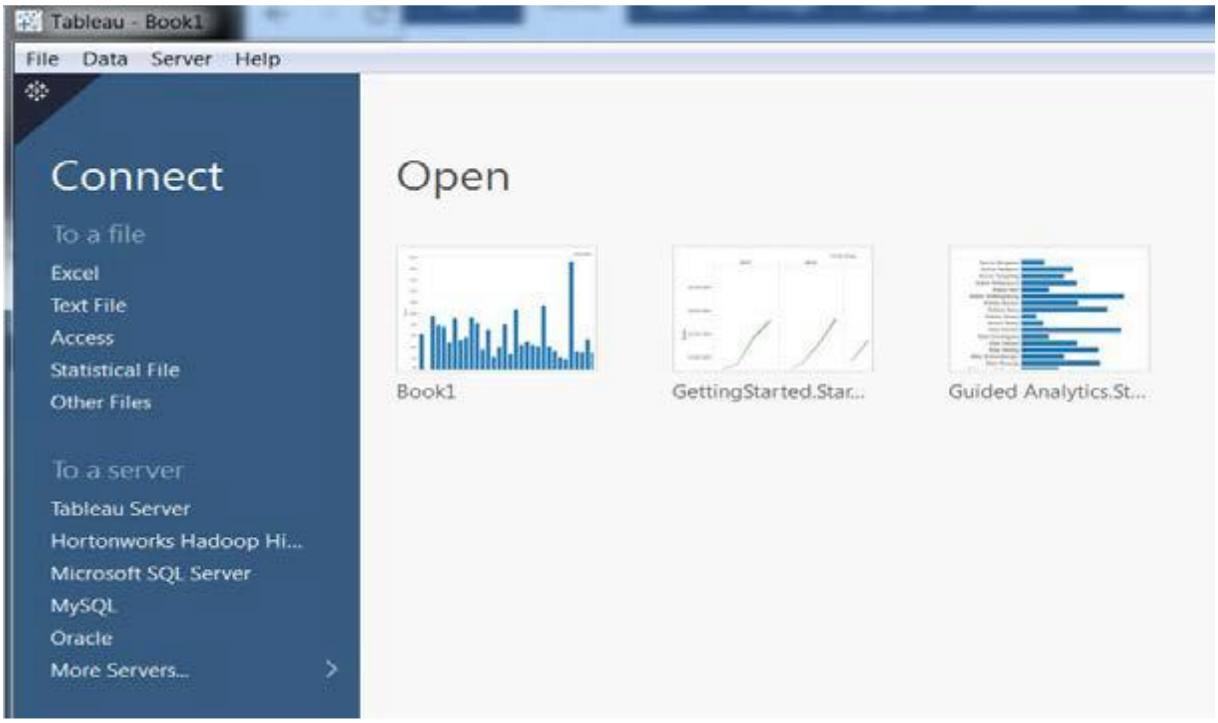
TABLEAU TO IMPORT HADOOP FILE AT ANALYTICS ENGINE

Open your tableau at your local computer (or lab computer on campus). Computer room on campus may or may not have tableau with the license key. In that case, you may use your local computer.

TABLEAU TO OPEN DATA FILE DIRECTLY FROM TABLEAU AND VISUALIZATION

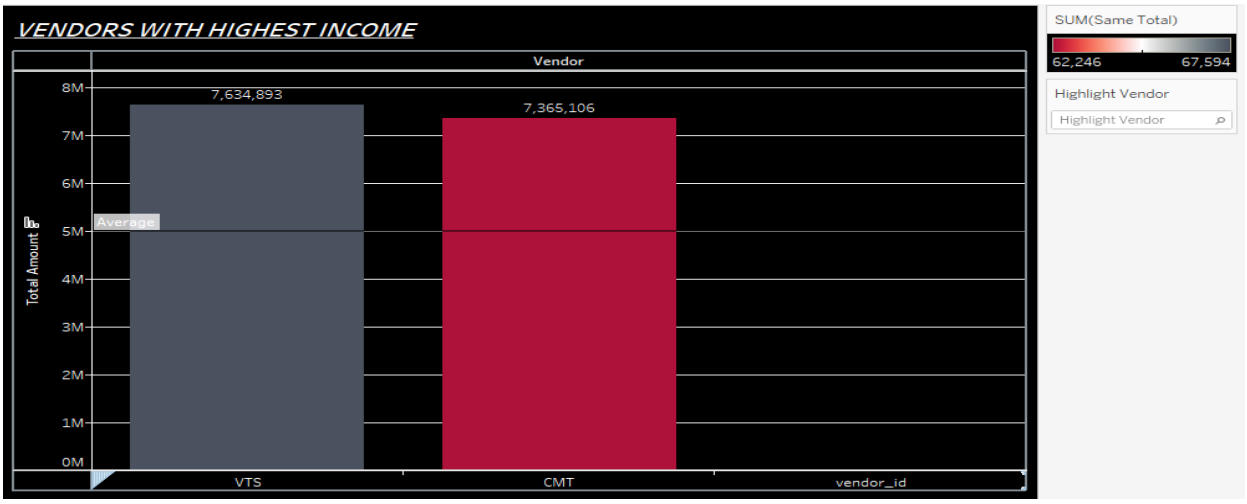
1. Download all the output files from Ambari, export the text file to excel and append all the files.

2. Open your Tableau to connect your server. You need to select **Text File** to open the file **000000-0**.



3. You will see the following data at **Data Source .**

5. Select **Sheet 1** next to **Data Source**, which will present the following frame. Drag vendor_id and total_amount to the canvas.



Analysis 8: Which hour of the day and which day of the month has most pickups in New York?

```
$ hive
```

The following Hive statement creates an external table that allows Hive to query data stored in HDFS. External tables preserve the data in the original file format, while allowing Hive to perform queries against the data within the file. The Hive statement below creates a new table named biz by describing the fields within the files, the delimiter (comma) between fields.

```
drop table if exists NewYork_counts_travels;


CREATE EXTERNAL TABLE IF NOT EXISTS NewYork_counts_travels(vendor_id String,
pickup_datetime timestamp,
dropoff_datetime timestamp,
passenger_count int,
trip_distance float,
pickup_longitude float,
pickup_latitude float,
```

```
rate_code Int,
store_and_fwd_flag String,
dropoff_longitude float,
dropoff_latitude float,
payment_type String,
fare_amount float,
surcharge float,
mta_tax float,
tip_amount float,
tolls_amount float,
total_amount float)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION ' /user/srihitha/New_York_Taxi_2014/'
TBLPROPERTIES ('skip.header.line.count'='1');
```

Select Query to make sure that the data exists in the hdfs directory.

```
SELECT COUNT(*) FROM NewYork_counts_travels;
```

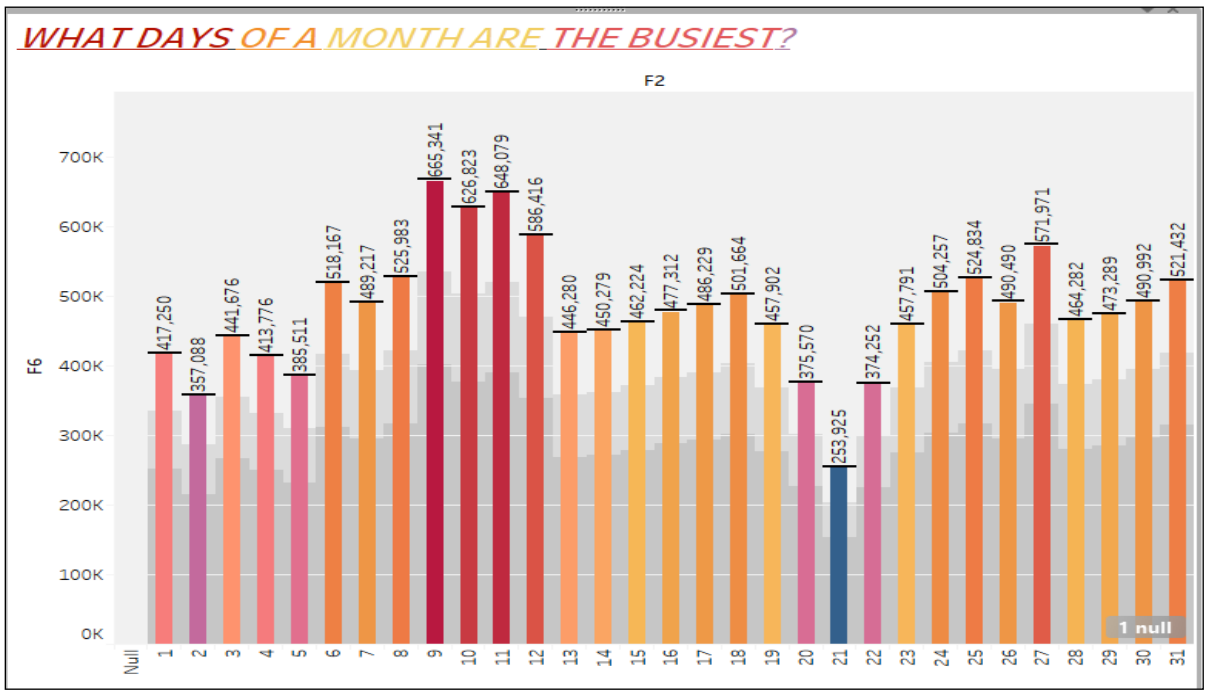
Now you run the following 2 HiveQLs in order to create a table "NY_passenger_pickup_counts", who tweets mostly in the order.

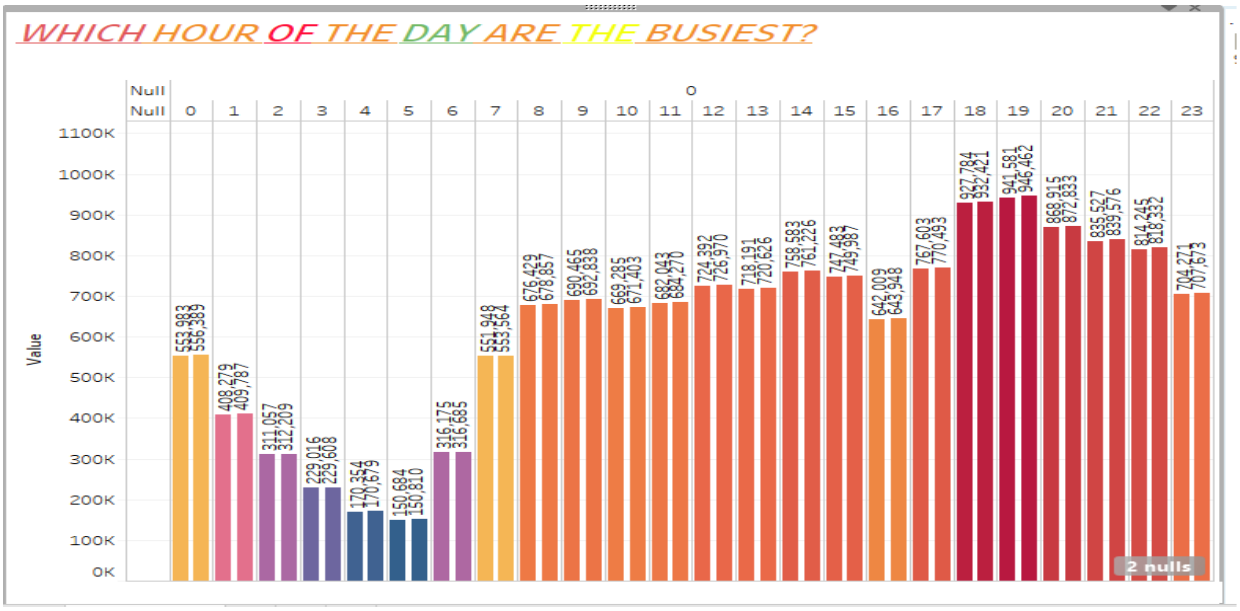
```
drop table if exists NY_passenger_pickup_counts;
```

```
CREATE TABLE IF NOT EXISTS NY_passenger_pickup_counts
AS SELECT pickup_datetime,
COUNT(DISTINCT(pickup_longitude)) AS PLttd,
COUNT(DISTINCT(pickup_latitude)) AS PLngttd
```

```
FROM NewYork_counts GROUP BY pickup_datetime;
```

The final visuals in Tableau should like the image below:

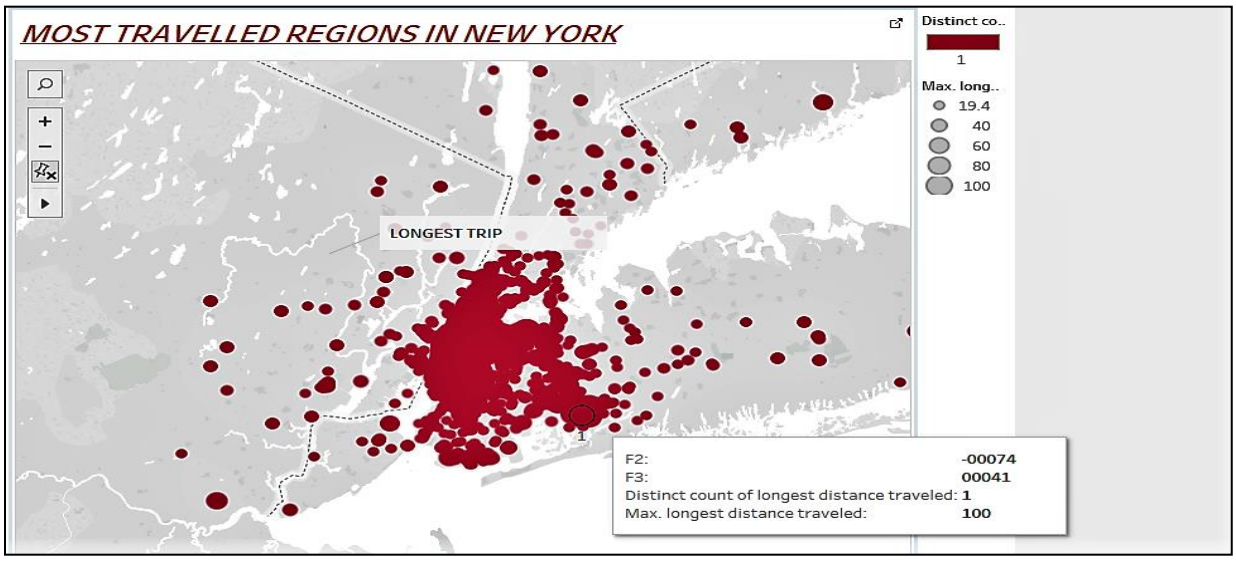




Analysis 9: What are the regions with highest demand and longest distance travelled in New York?

```
drop table if exists NY_Distance_travelled_most_main;


CREATE TABLE IF NOT EXISTS  NY_Distance_travelled_most_main
AS SELECT NewYork_counts_travels.pickup_longitude, NewYork_counts_travels.pickup_latitude,
MAX(NewYork_counts_travels.trip_distance) AS longest
FROM NewYork_counts_travels GROUP BY NewYork_counts_travels.pickup_longitude,
NewYork_counts_travels.pickup_latitude
ORDER BY longest DESC;
```

The final Geo Map visual in Tableau should look like the following image.



PREDICTIVE ANALYSIS USING AZURE MACHINE LEARNING

Objective

The main objective of this tutorial is to perform predictive analysis on the total_amount of New York Motor Taxi data set and on trip_total of Chicago Taxi dataset using Microsoft Azure Machine Learning Studio, based on the features like vendors, communities, pickup latitudes and longitudes, drop off latitudes and longitudes, tips, tolls, fare amount, taxes and extras.

Step 1: Creating an Azure ML Experiment

 Azure ML offers a free-tier account, which you can use to complete this tutorial.

Step 2: Sign up for a Microsoft Account

 If you do not already have a Microsoft account, sign up for one at https://signup.live.com/ . You don't need to use your school email account to sign up but you can use any email account.

Step 3: Sign up for a free Azure ML

1. Browse to http://bit.ly/azureml_login and click **Get started now**.

2. When prompted, choose the option to sign in, and sign in with your Microsoft account credentials.

3. On the **Welcome** page, watch the overview video if you want to see an introduction to Azure ML Studio. Then close the **Welcome** page by clicking the checkmark icon.

Step 4: Creating an Azure ML Experiment for Linear Regression Model for New York Data Set

1. Azure ML enables us to create experiments in which we can manipulate data, create predictive models, and visualize the results.

2. In this tutorial, you will create a simple experiment in which you will explore a new York and Chicago datasets that contains details on the numerous taxi rides of New York city and of Chicago from which you would like to predict score of the total_amount and trip_total which sre the summation of fare, tips, tolls, taxes and extras of taxi rides based on '**Vendors_id**', '**picku_datetime**', '**trip_distance**', '**pickup_latitude**', '**pickup_longitude**', '**dropff_latitude**', '**dropoff_longitude**' and '**payment_type**' in New York and '**taxi_id**', '**trip_miles**', '**pickup_community_area**', '**dropoff_community_area**' and '**company**' in Chicago data set.

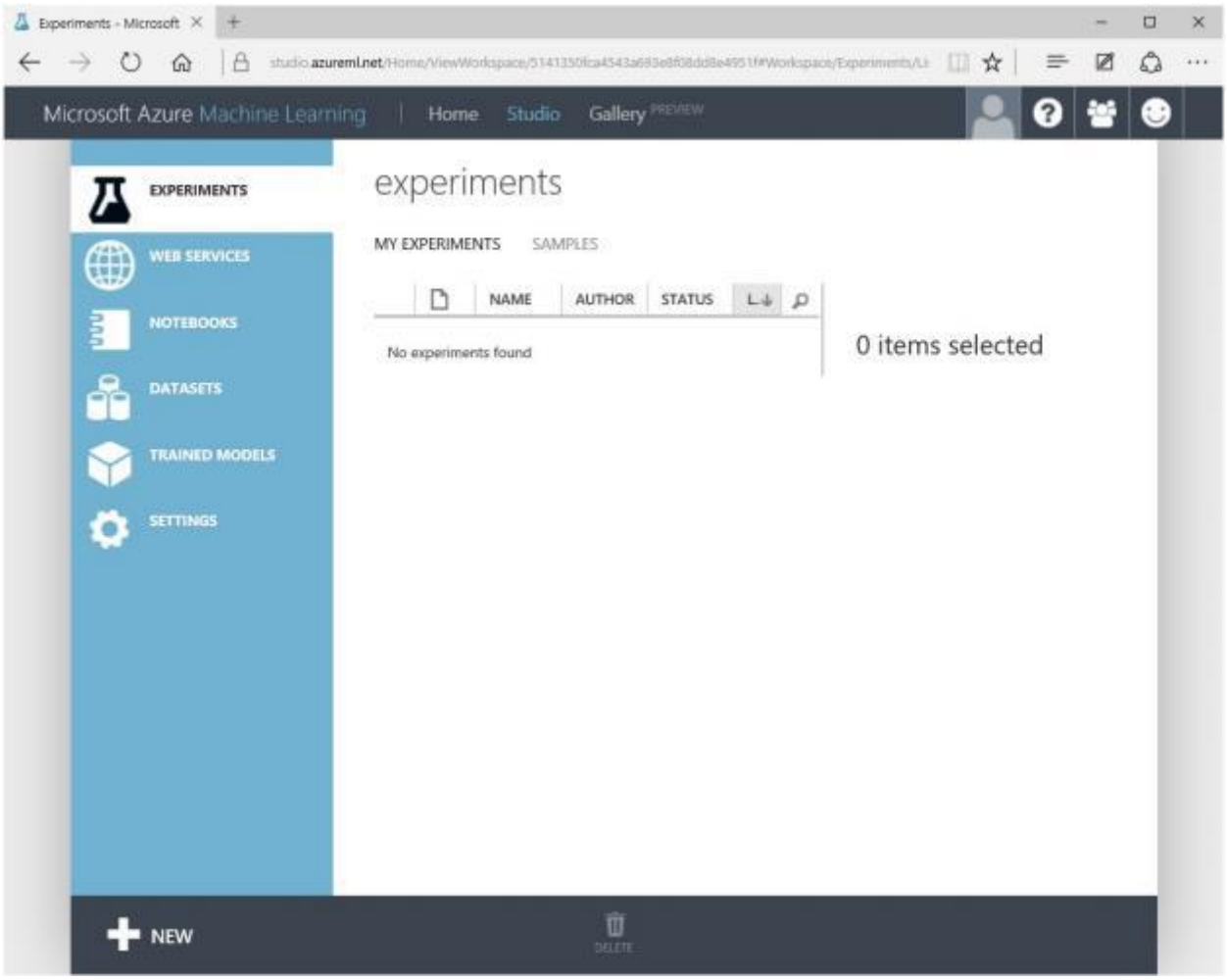
Sign into Azure ML Studio

1. Open a browser and browse to https://studio.azureml.net.

2. Click **Sign In** and sign in using the Microsoft account associated with your free Azure ML account.

3. If the Welcome page is displayed, close it by clicking the **OK** icon (which looks like a checkmark). Then, if the New page (containing a collection of Microsoft samples) is displayed, close it by clicking the Close icon (which looks like an X).

6. You should now be in Azure ML Studio with the Experiments page selected, which looks like the following image (if not, click the Studio tab at the top of the page).



5. In the Studio, at the bottom left, click **NEW**. Then in the collection of Microsoft samples, select **Blank Experiment**.

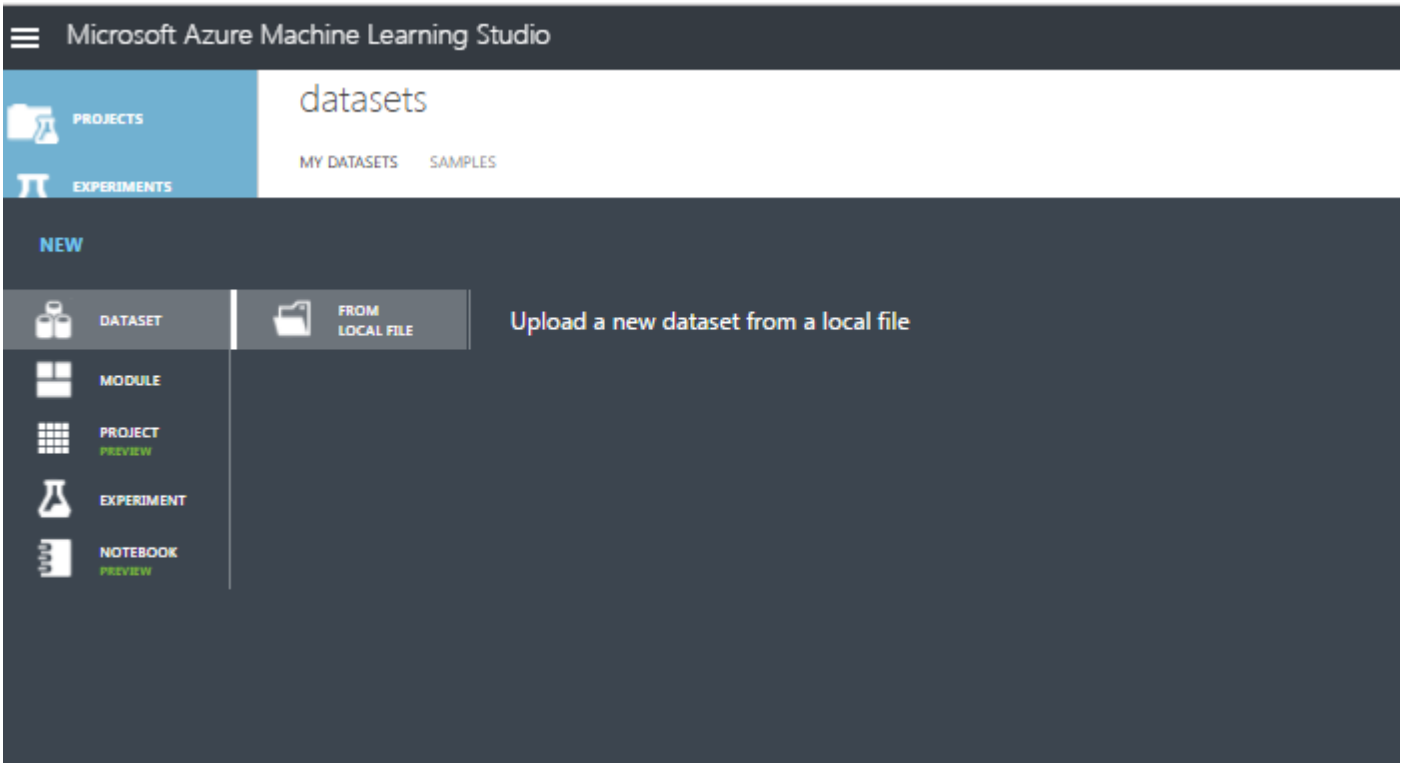
6. Change the title of your experiment from "Experiment created on today's date" to "**New York fare prediction**".

Uploading a Data File to Azure ML:

When you need to create, an experiment based on your own data or data you have obtained from a third-party, you must begin by uploading the data to Azure ML. To predict the Trip_Total and Total_amount based on number of rides, the dataset must be uploaded.

1. Open the **nyc_taxi_data_2014** file in the folder where you extracted the lab files, using either a spreadsheet application such as Microsoft Excel, or a text editor such as Microsoft Windows Notepad.

2. With the **New York fare prediction** experiment open, at the bottom left, click NEW. Then in the NEW dialog box, click the DATASET tab as shown in the below image.



3. Click **FROM LOCAL FILE**. Then in the Upload a new dataset dialog box, browse to select the **nyc_taxi_data_2014** file from the folder where you extracted the lab files on your local computer and enter the following details as shown in the image below, and then click the OK icon.

- **This is a latest version of an existing dataset**: Unselected
- **Enter a name for the new dataset**: **nyc_taxi_data**
- **Select a type for the new dataset**: Generic CSV file with a header (.csv)
- **Provide an optional description**: Taxi Rides in NYC

4. Wait for the upload of the dataset to be completed, and then on the experiment items pane, expand **Saved Datasets** and **My Datasets** to verify that the dataset is listed.

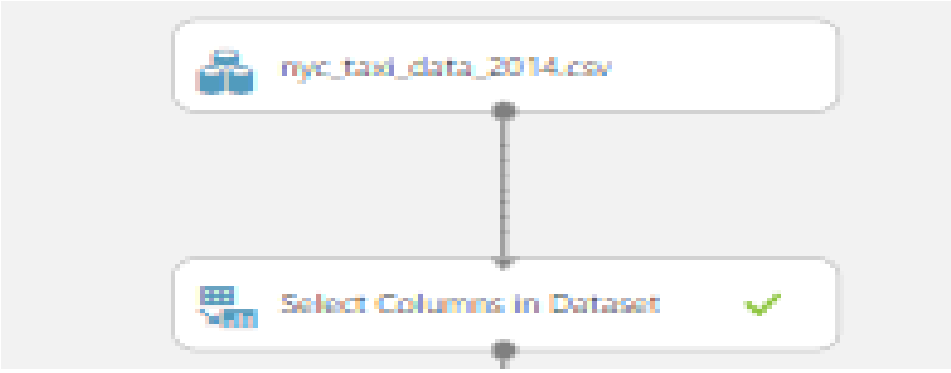
Visualize the Dataset in Azure ML:

1. Drag the **nyc_taxi_data_2014.csv** dataset to the canvas for the **New York fare prediction** experiment.

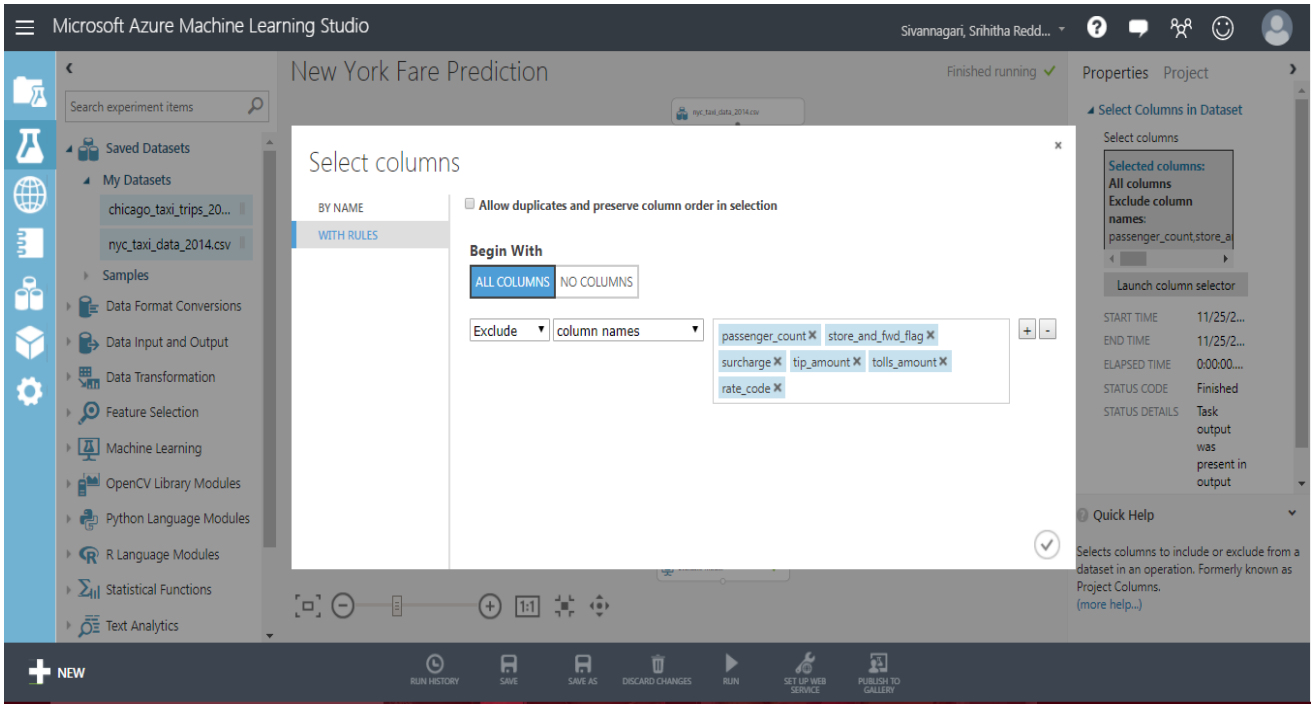
2. Right-click the output port for the **nyc_taxi_data_2014.csv** dataset on the canvas and click **Visualize** to view the data in the dataset.

3. Verify that the dataset contains the data you viewed in the source file, and then close the dataset.

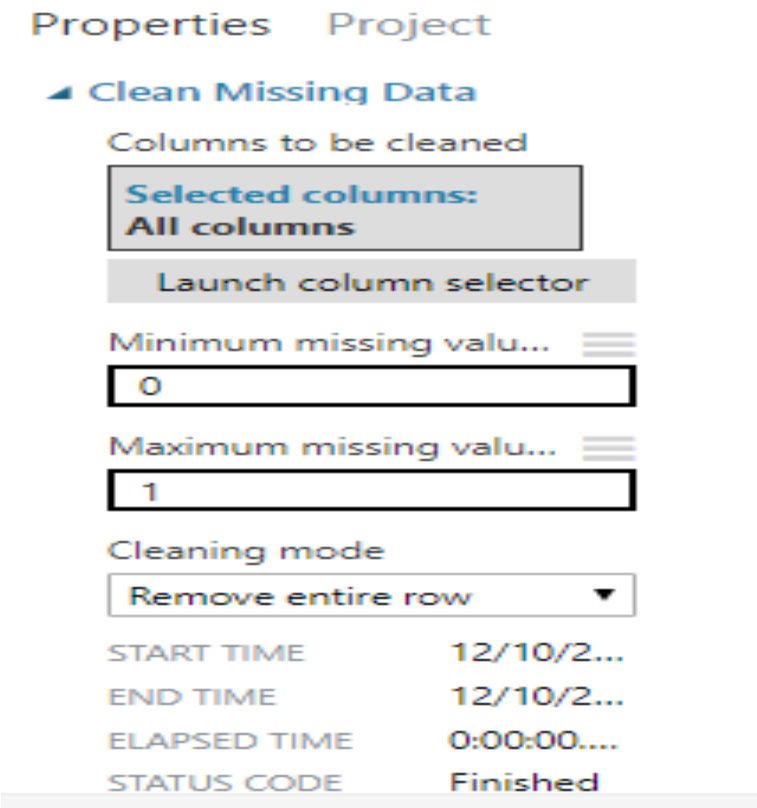
5. Search for the **Select Columns in Dataset (Project Columns)** module and drag it onto your canvas. Connect the Results Dataset output of the **nyc_taxi_data_2014.csv** module to the input port of the **Select Columns in Dataset (Project Columns)** module.



6. In the properties pane, select **launch column selector** and select the with rules option. Under **ALL COLUMNS** exclude the column names **Passenger_count**, **Store_and_fwd_flag, surcharge, tip_amount, tolls_amount,** and **rate_code.**

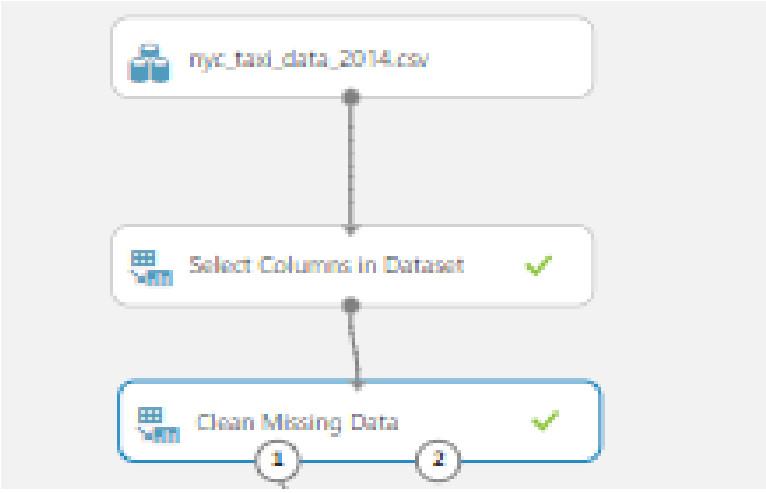


7. Drag the **Clean Missing Data** module to the experiment canvas and connect it to the **Select Columns in Dataset** module. In the **Properties** pane, select **Remove entire row** under **Cleaning mode**. This directs **Clean Missing Data** to clean the data by removing rows that have any missing values. Double-click the module and type the comment "Remove missing value rows."



8. Run the experiment by clicking **RUN** at the bottom of the page.

9. When the experiment has finished running, all the modules have a green check mark to indicate that they finished successfully. Notice also the **Finished running** status in the upper-right corner.



10. Drag another **Select Columns in Dataset** module to the experiment canvas. Connect the left output port of the **Clean Missing Data** module to the input of the **Select Columns in Dataset** module.

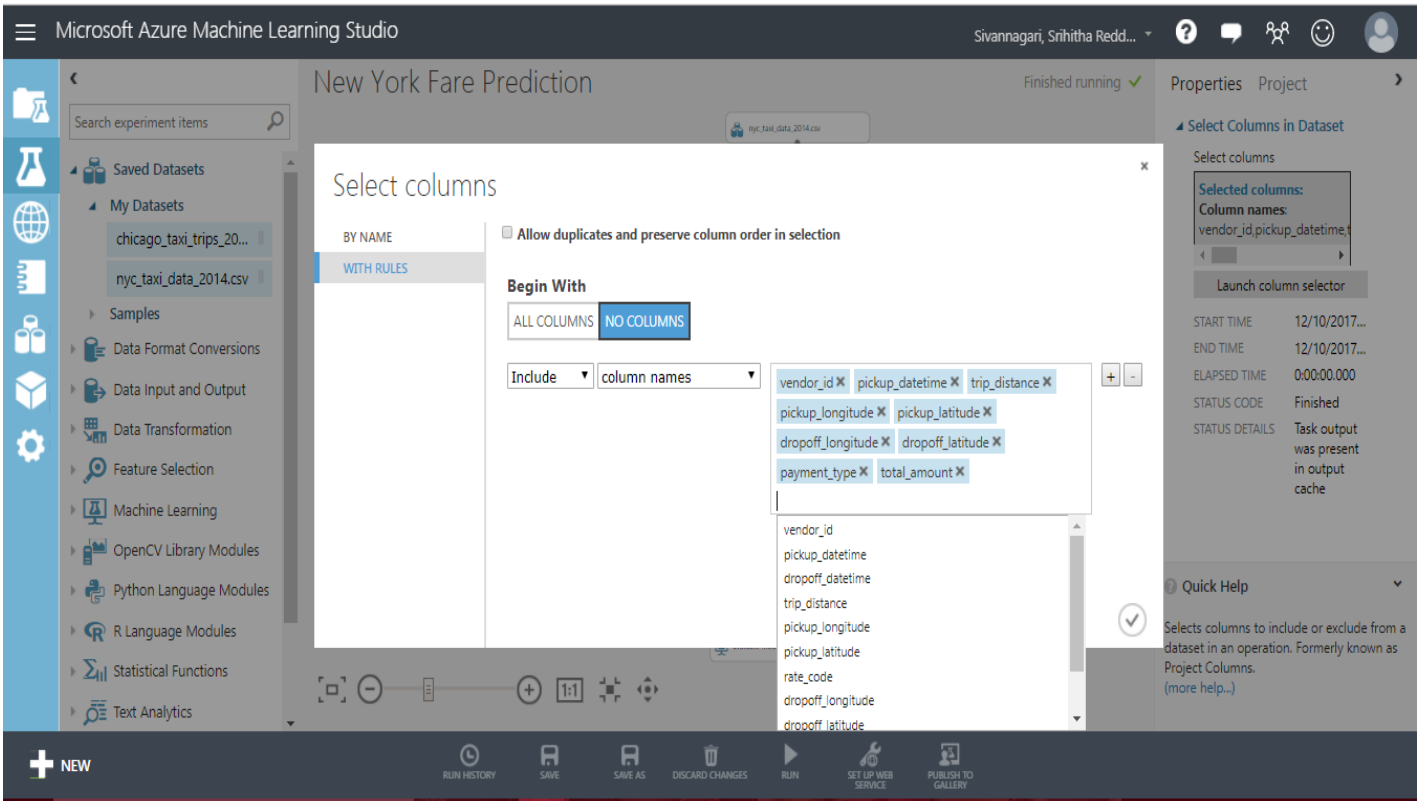
11. Double-click the module and type "Select features for prediction."

12. Click **Launch column selector** in the **Properties** pane.

13. Click **With rules**.

14. Under **Begin With**, click **No columns**. In the filter row, select **Include** and **column names** and select our list of column names in the text box. This directs the module to not pass through any columns (features) except the ones that we specify.

15. Click the check mark (OK) button.

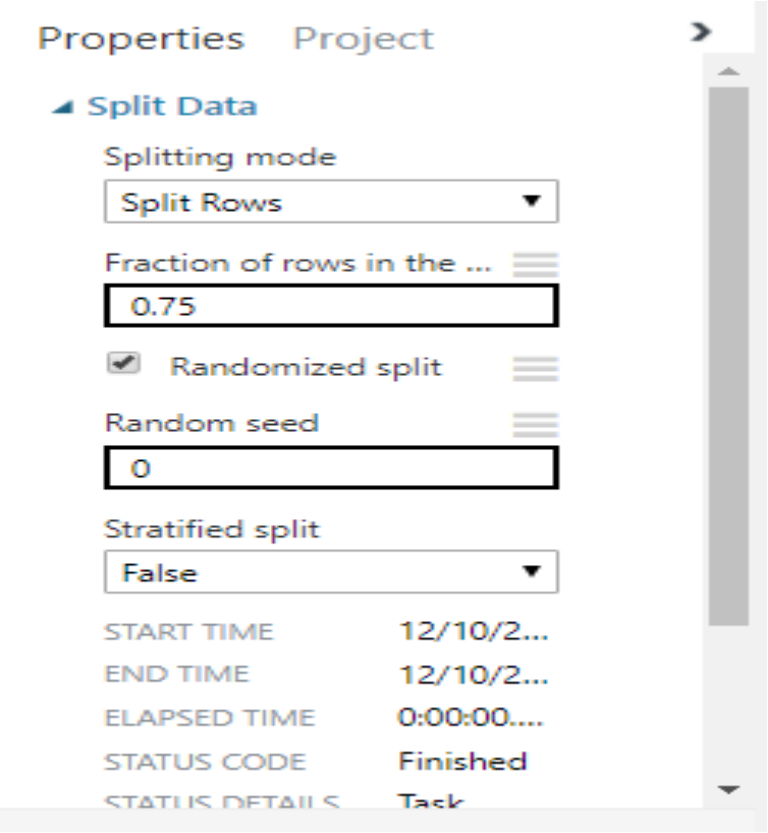


16. This produces a filtered dataset containing only the features we want to pass to the learning algorithm we'll use in the next step. Later, you can return and try again with a different selection of features.

17. Because we want to predict total_amount, which is a number, we'll use a regression algorithm. For this tutorial, we'll use a simple *linear regression* model.

18. Select and drag the **Split Data** module to the experiment canvas and connect it to the last **Select Columns in Dataset** module.

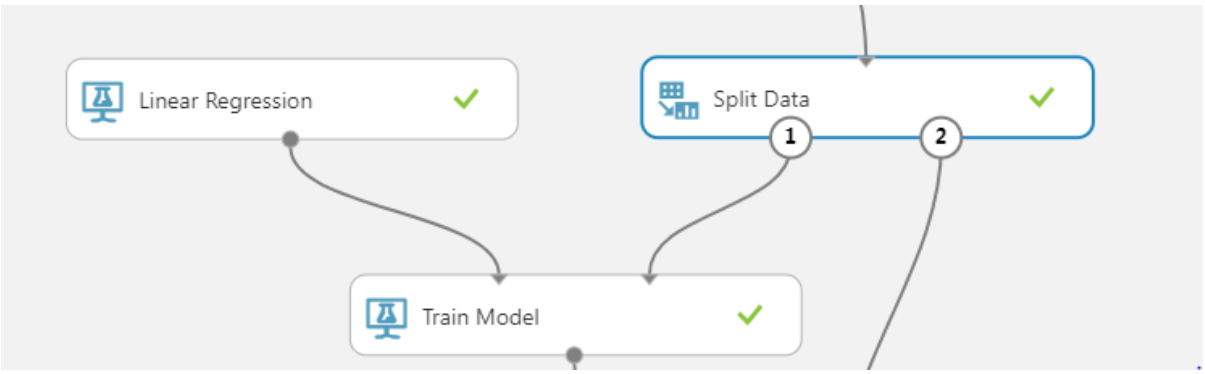
19. Click the **Split Data** module to select it. Find the **Fraction of rows in the first output dataset** (in the **Properties** pane to the right of the canvas) and set it to 0.75. This way, we'll use 75 percent of the data to train the model, and hold back 25 percent for testing (later, you can experiment with using different percentages).



20. Run the experiment. When the experiment is run, the **Select Columns in Dataset** and **Split Data** modules pass column definitions to the modules we'll be adding next.

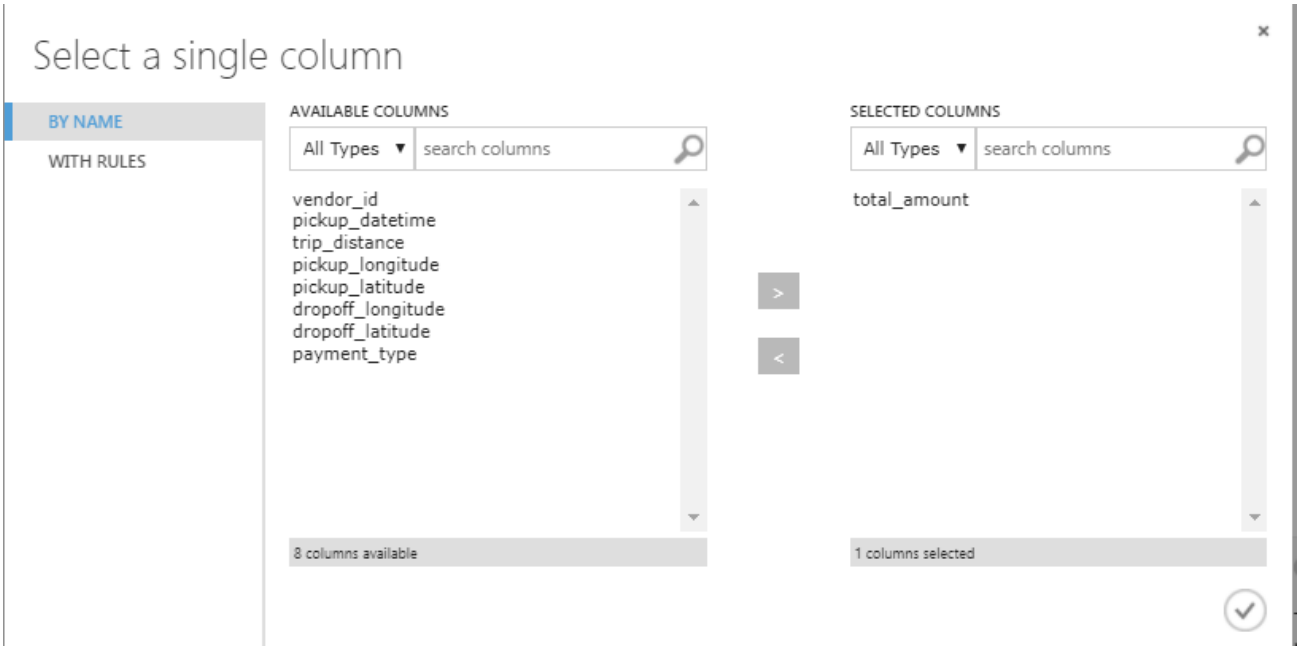
21. To select the learning algorithm, expand the **Machine Learning** category in the module palette to the left of the canvas, and then expand **Initialize Model**. This displays several categories of modules that can be used to initialize machine learning algorithms. For this experiment, select the **Linear Regression** module under the **Regression** category, and drag it to the experiment canvas. (You can also find the module by typing "linear regression" in the palette Search box.)

22. Find and drag the **Train Model** module to the experiment canvas. Connect the output of the **Linear Regression** module to the left input of the **Train Model** module, and connect the training data output (left port) of the **Split Data** module to the right input of the **Train Model** module.



23. Click the **Train Model** module, click **Launch column selector** in the **Properties** pane, and then select the **total_amount** column. This is the value that our model is going to predict.

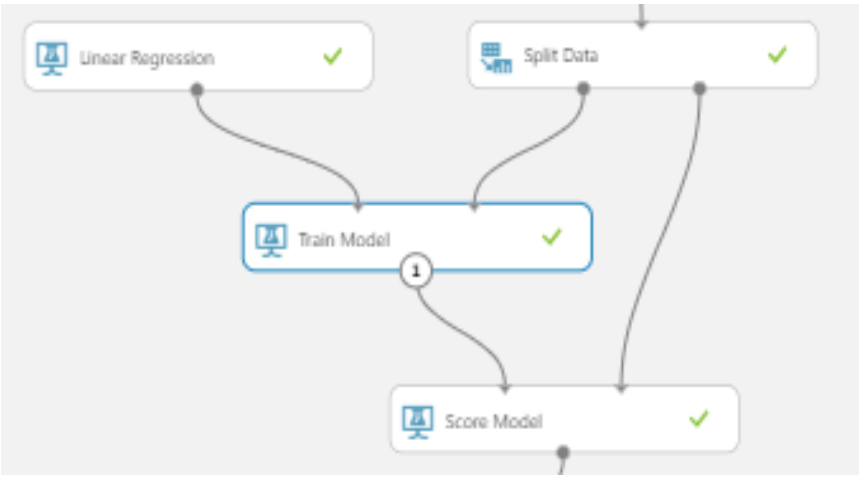
    You select the **total_amount** column in the column selector by moving it from the **Available columns** list to the **Selected columns** list.



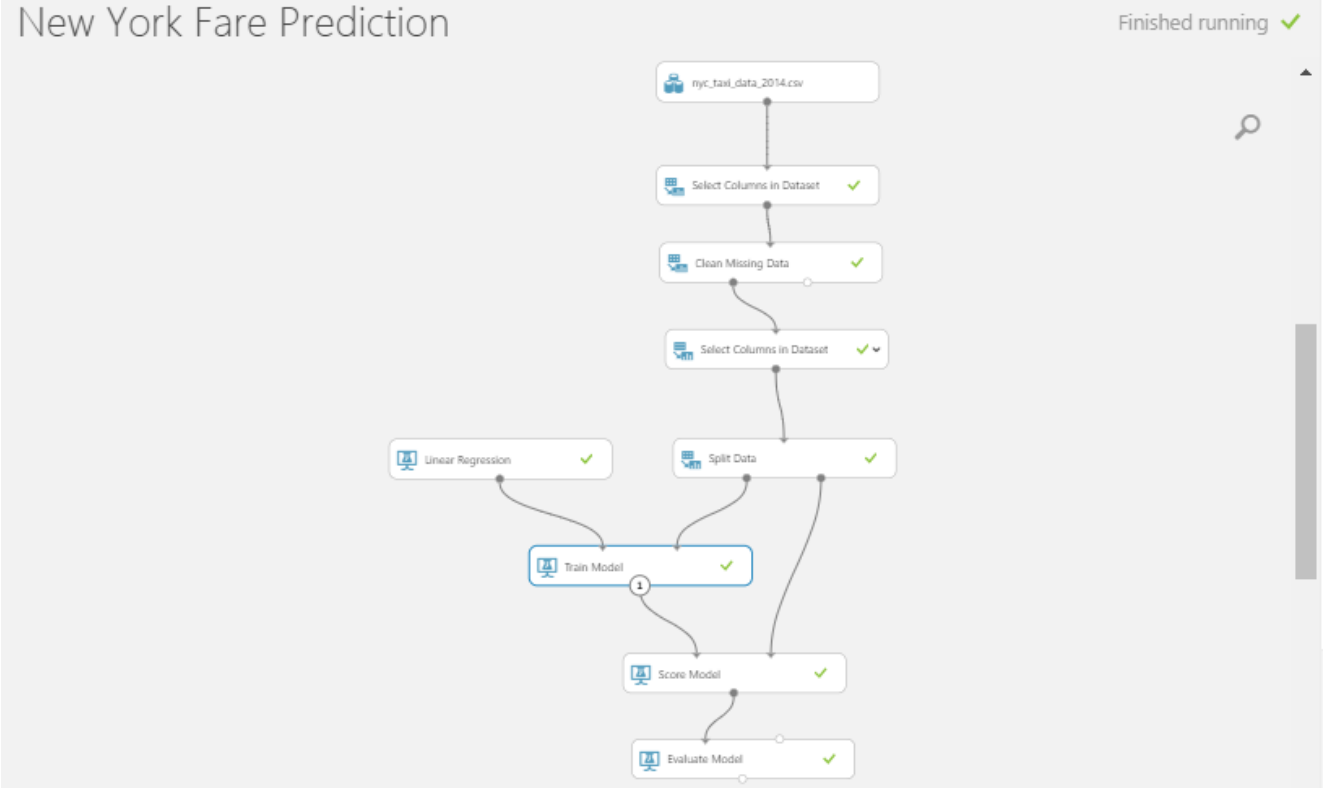
24. Run the experiment.

25. Find and drag the **Score Model** module to the experiment canvas. Connect the output of the **Train Model** module to the left input port of **Score Model**. Connect the test data output (right port) of the **Split Data** module to the right input port of **Score Model.**

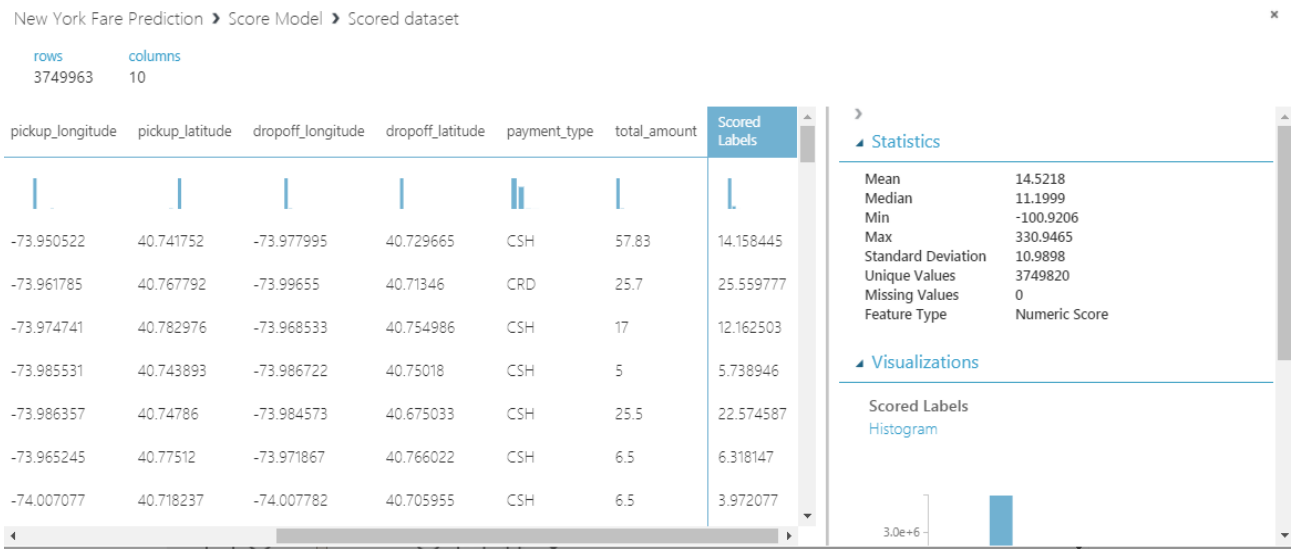
26. Search for **Evaluate model** and connect to the output port of Score model.



We now have a trained regression model that can be used to score new New York data to make total_amount predictions.

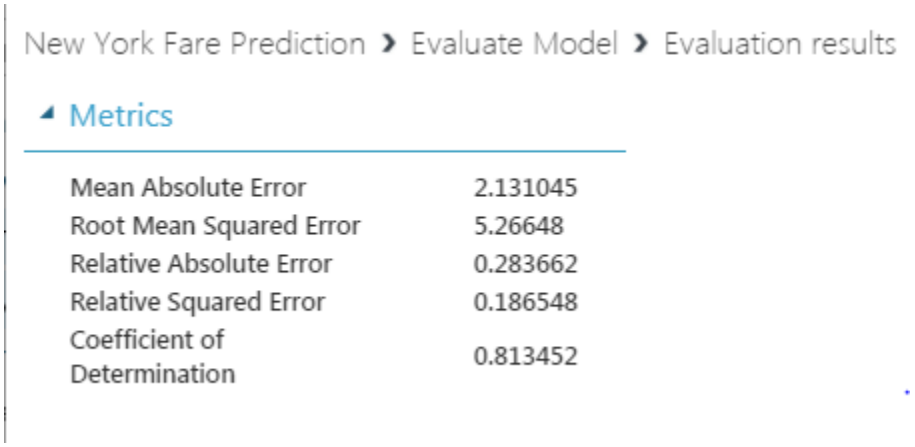


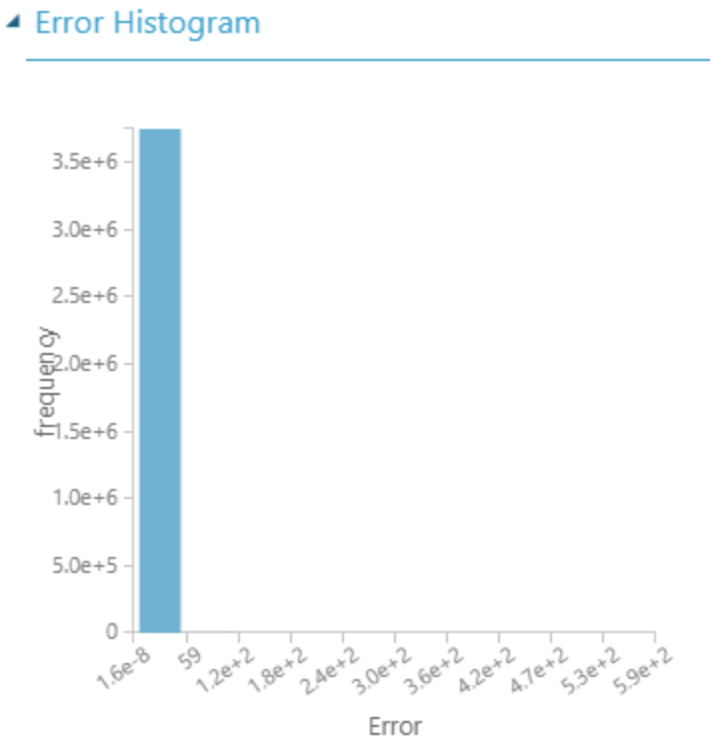
27. Run the experiment and view the output from the **Score Model** module (click the output port of **Score Model** and select **Visualize**). The output shows the predicted values for price and the known values from the test data.



28. Run the experiment.

29. To view the output from the **Evaluate Model** module, click the output port, and then select **Visualize**.





Step 5: Creating an Azure ML Experiment for Linear Regression Model for Chicago Taxi Data Set

1. Azure ML enables us to create experiments in which we can manipulate data, create predictive models, and visualize the results.

2. In this tutorial, you will create a simple experiment in which you will explore a new York and Chicago datasets that contains details on the numerous taxi rides of New York city and of Chicago from which you would like to predict score of the trip_total which are the summation of fare, tips, tolls, taxes and extras of taxi rides based on **'taxi_id'**, **'trip_miles'**, **'pickup_community_area'**, **'dropoff_community_area'** and **'company'** in Chicago data set.

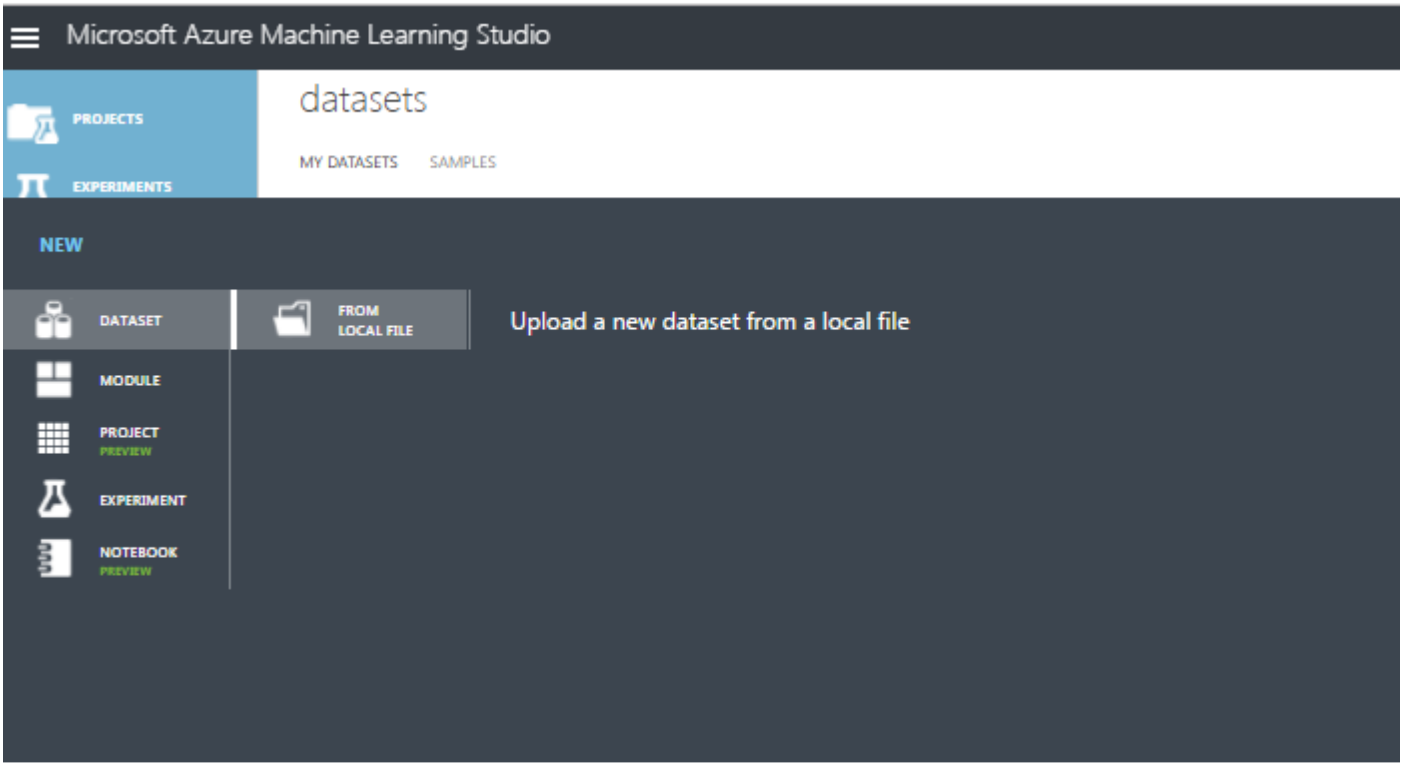
Sign into Azure ML Studio

1. Open a browser and browse to https://studio.azureml.net.

2. Click **Sign In** and sign in using the Microsoft account associated with your free Azure ML account.

3. If the Welcome page is displayed, close it by clicking the **OK** icon (which looks like a checkmark). Then, if the New page (containing a collection of Microsoft samples) is displayed, close it by clicking the Close icon (which looks like an X).

7. You should now be in Azure ML Studio with the Experiments page selected, which looks like the following image (if not, click the Studio tab at the top of the page).



0

3. Click **FROM LOCAL FILE**. Then in the Upload a new dataset dialog box, browse to select the **Chicago_taxi_trips_2016.csv** file from the folder where you extracted the lab files on your local computer and enter the following details as shown in the image below, and then click the OK icon.
• **This is a latest version of an existing dataset**: Unselected
• **Enter a name for the new dataset**: **Chicago_taxi_data**
• **Select a type for the new dataset**: Generic CSV file with a header (.csv)
• **Provide an optional description**: Taxi Rides in Chicago

4. Wait for the upload of the dataset to be completed, and then on the experiment items pane, expand **Saved Datasets** and **My Datasets** to verify that the dataset is listed.

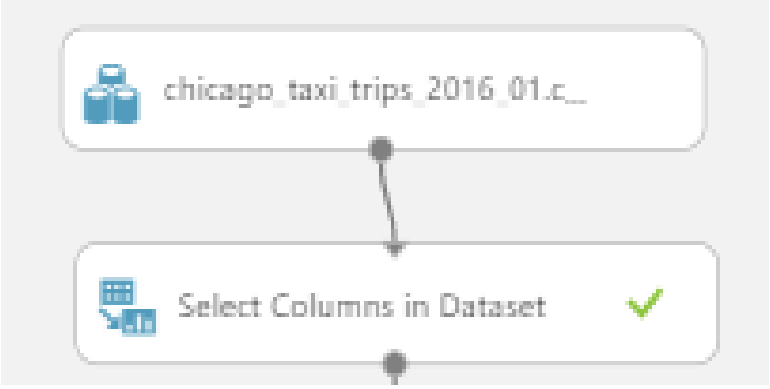
Visualize the Dataset in Azure ML:

1. Drag the **Chicago_taxi_trips_2016.csv** dataset to the canvas for the **Chicago fare prediction** experiment.

2. Right-click the output port for the **Chicago_taxi_trips_2016.csv** dataset on the canvas and click **Visualize** to view the data in the dataset.

3. Verify that the dataset contains the data you viewed in the source file, and then close the dataset.

4. Search for the **Select Columns in Dataset (Project Columns)** module and drag it onto your canvas. Connect the Results Dataset output of the **nyc_taxi_data_2014.csv** module to the input port of the **Select Columns in Dataset (Project Columns)** module.



5.Drag another **Select Columns in Dataset** module to the experiment canvas. Connect the left output port of the **Clean Missing Data** module to the input of the **Select Columns in Dataset** module.

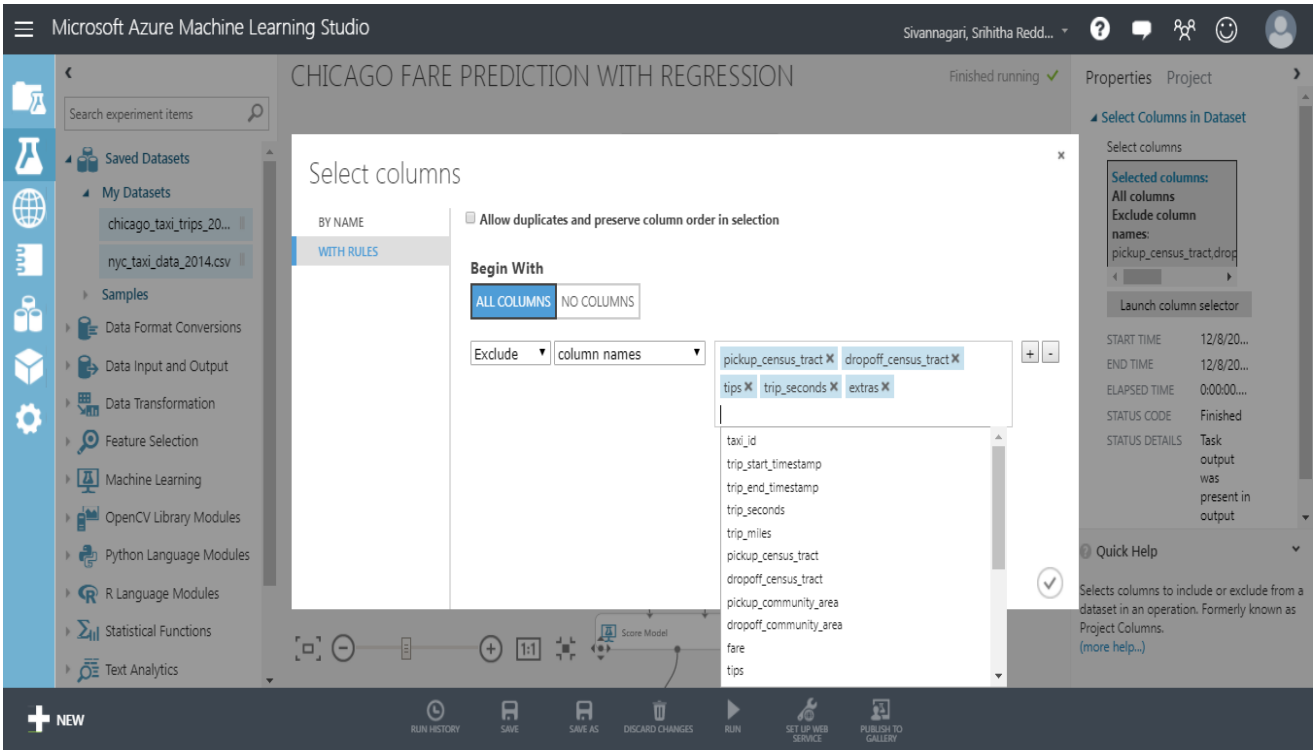
6.Double-click the module and type "Select features for prediction."

7.Click **Launch column selector** in the **Properties** pane.

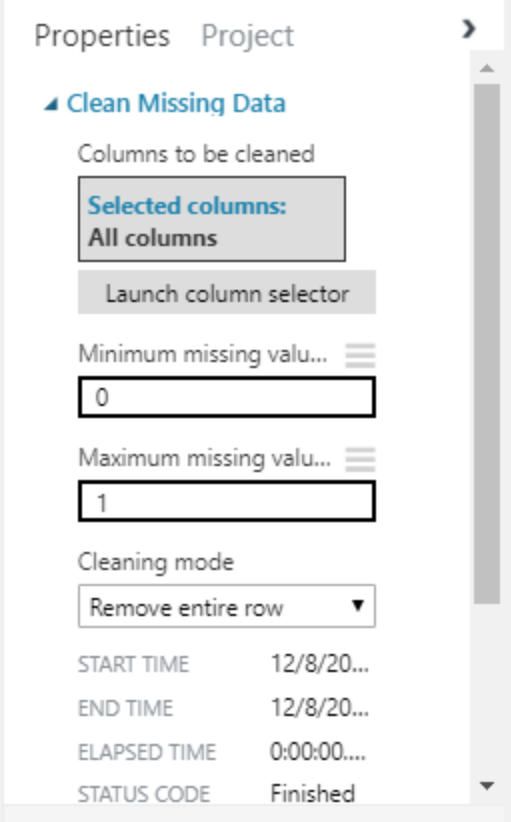
8.Click **With rules**.

9.Under **Begin With**, click **No columns**. In the filter row, select **Include** and **column names** and select our list of column names in the text box. This directs the module to not pass through any columns (features) except the ones that we specify.

10.Click the check mark (OK) button.

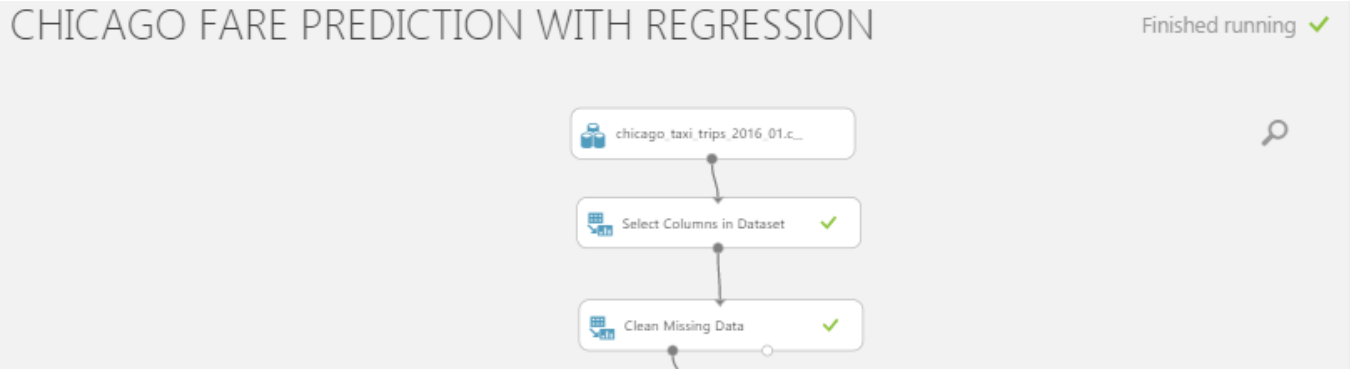


11. Drag the **Clean Missing Data** module to the experiment canvas and connect it to the **Select Columns in Dataset** module. In the **Properties** pane, select **Remove entire row** under **Cleaning mode**. This directs **Clean Missing Data** to clean the data by removing rows that have any missing values. Double-click the module and type the comment "Remove missing value rows."



12. Run the experiment by clicking **RUN** at the bottom of the page.

13. When the experiment has finished running, all the modules have a green check mark to indicate that they finished successfully. Notice also the **Finished running** status in the upper-right corner.



14. Drag another **Select Columns in Dataset** module to the experiment canvas. Connect the left output port of the **Clean Missing Data** module to the input of the **Select Columns in Dataset** module.

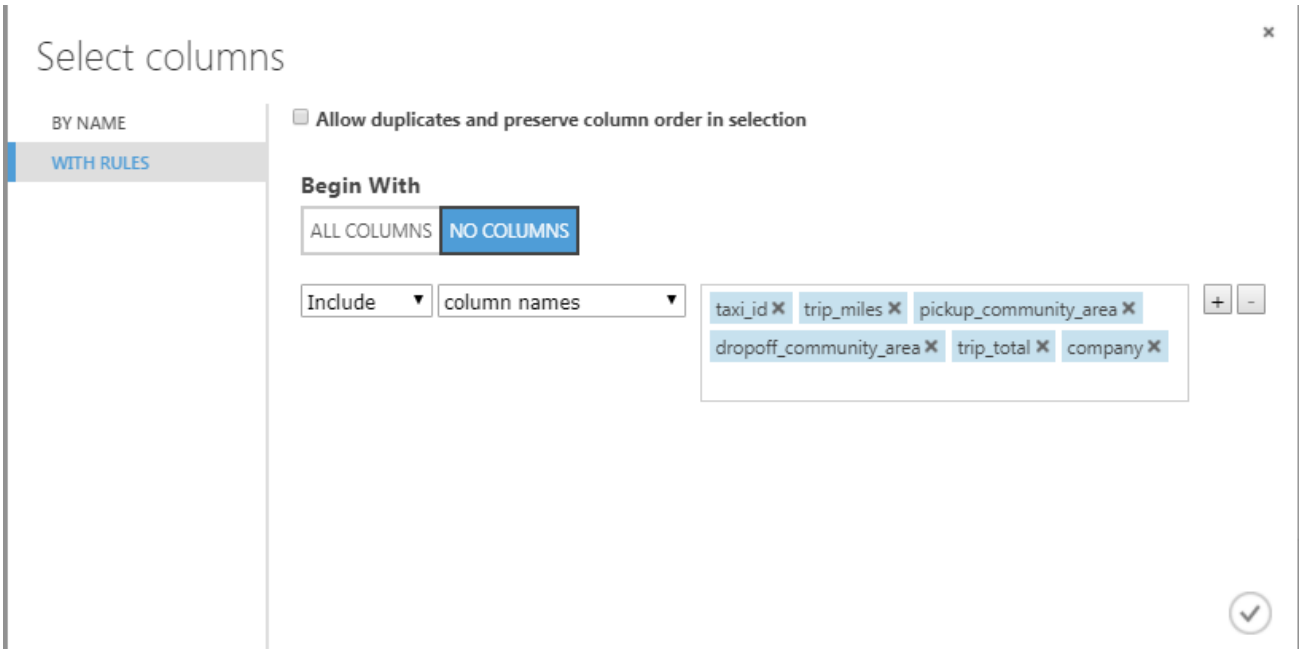
15. Double-click the module and type "Select features for prediction."

16.Click **Launch column selector** in the **Properties** pane.

17.Click **With rules**.

18.Under **Begin With**, click **No columns**. In the filter row, select **Include** and **column names** and select our list of column names in the text box. This directs the module to not pass through any columns (features) except the ones that we specify.

19.Click the check mark (OK) button.

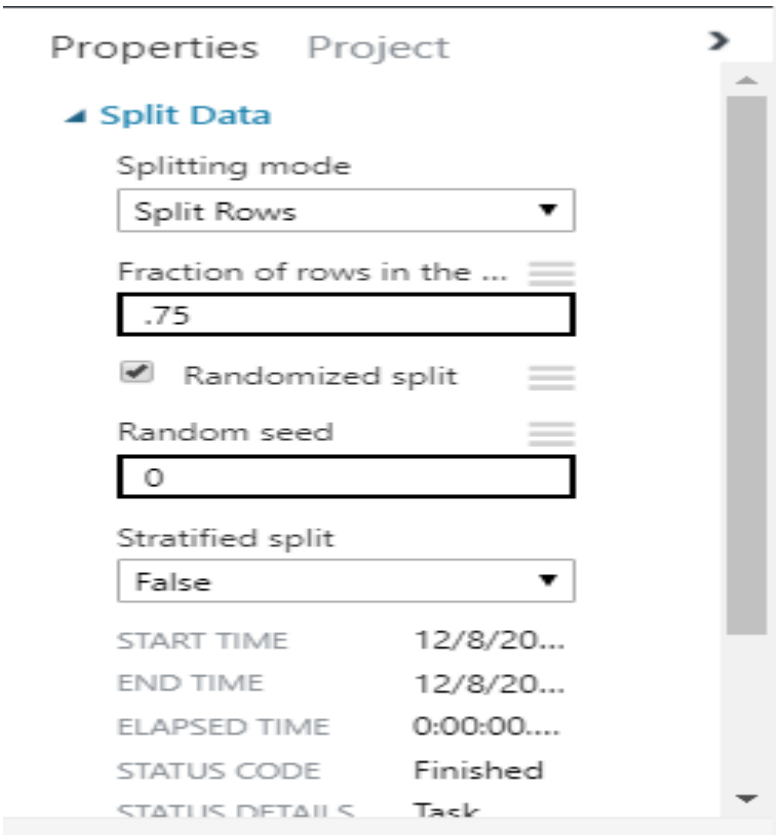


20. This produces a filtered dataset containing only the features we want to pass to the learning algorithm we'll use in the next step. Later, you can return and try again with a different selection of features.

21.Because we want to predict total_amount, which is a number, we'll use a regression algorithm. For this tutorial, we'll use a simple ***linear regression model***.

22.Select and drag the **Split Data** module to the experiment canvas and connect it to the last **Select Columns in Dataset** module.

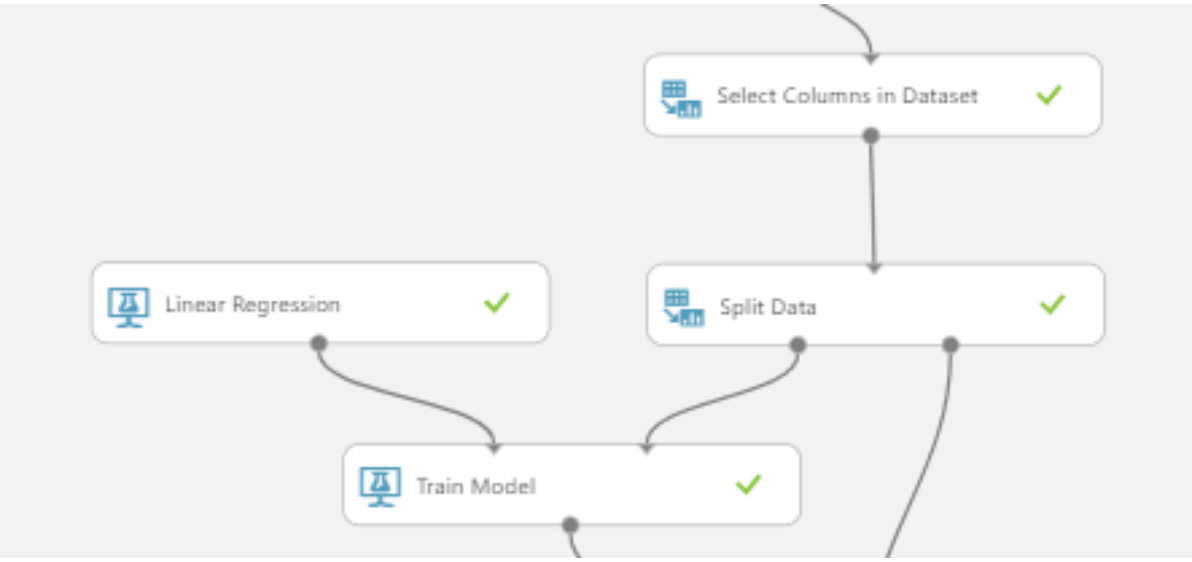
23.Click the **Split Data** module to select it. Find the **Fraction of rows in the first output dataset** (in the **Properties** pane to the right of the canvas) and set it to 0.75. This way, we'll use 75 percent of the data to train the model, and hold back 25 percent for testing (later, you can experiment with using different percentages).



24. Run the experiment. When the experiment is run, the **Select Columns in Dataset** and **Split Data** modules pass column definitions to the modules we'll be adding next.

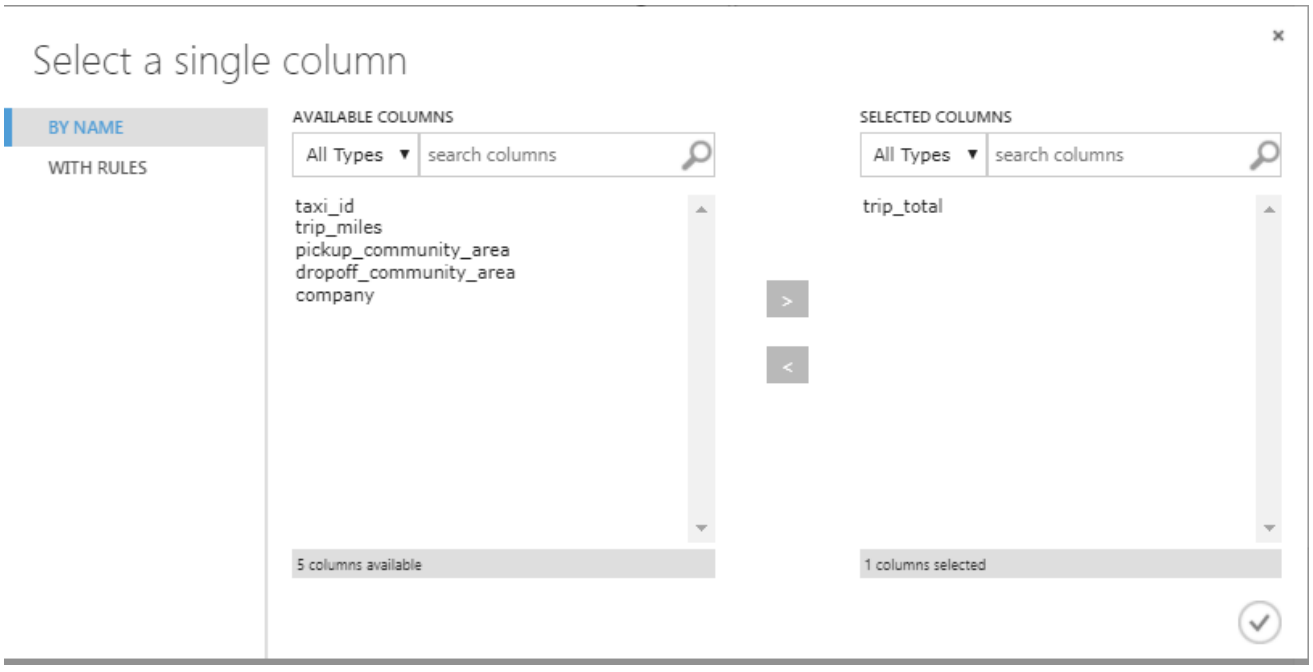
25.To select the learning algorithm, expand the **Machine Learning** category in the module palette to the left of the canvas, and then expand **Initialize Model**. This displays several categories of modules that can be used to initialize machine learning algorithms. For this experiment, select the **Linear Regression** module under the **Regression** category, and drag it to the experiment canvas. (You can also find the module by typing "linear regression" in the palette Search box.)

26. Find and drag the **Train Model** module to the experiment canvas. Connect the output of the **Linear Regression** module to the left input of the **Train Model** module, and connect the training data output (left port) of the **Split Data** module to the right input of the **Train Model** module.



27. Click the **Train Model** module, click **Launch column selector** in the **Properties** pane, and then select the **total_amount** column. This is the value that our model is going to predict.

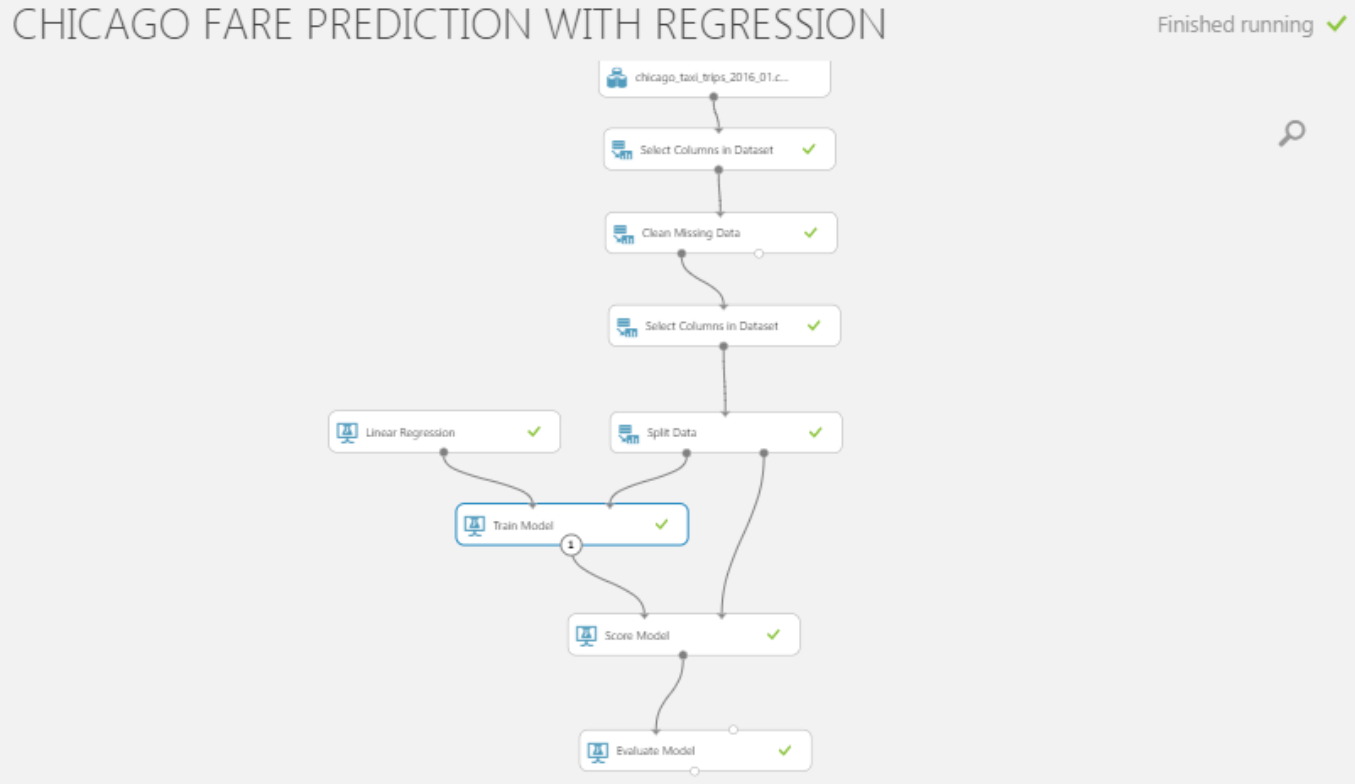
You select the **total_amount** column in the column selector by moving it from the **Available columns** list to the **Selected columns** list.



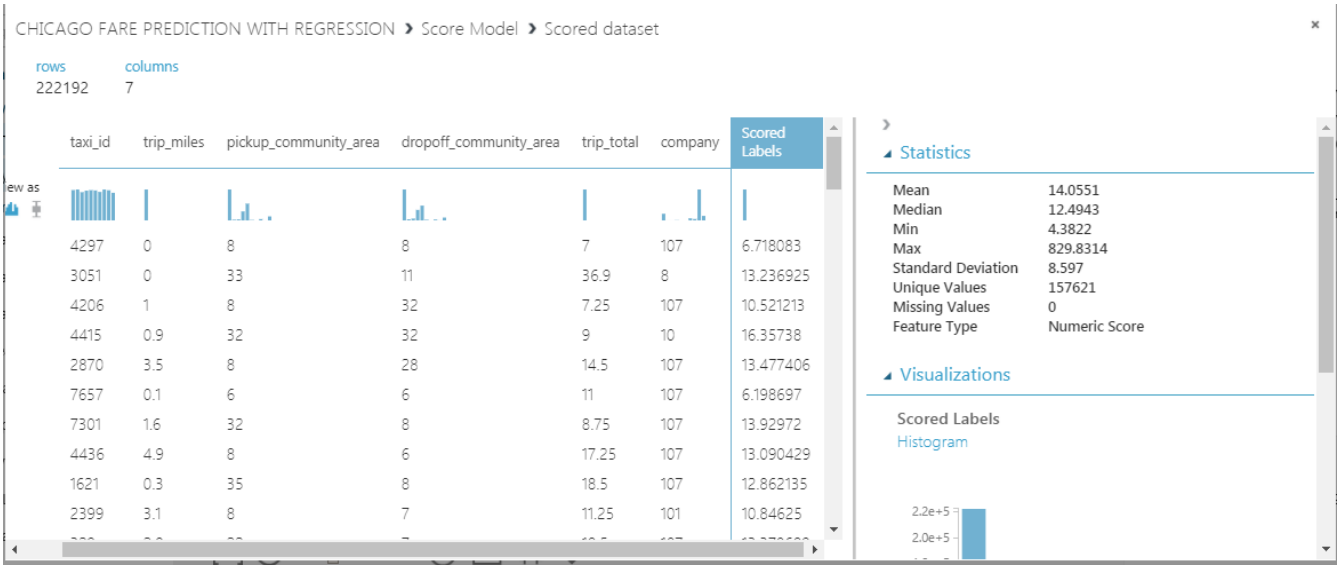
28. Run the experiment.

29.Find and drag the **Score Model** module to the experiment canvas. Connect the output of the **Train Model** module to the left input port of **Score Model**. Connect the test data output (right port) of the **Split Data** module to the right input port of **Score Model.**

30. Search for **Evaluate model** and connect to the output port of Score model.

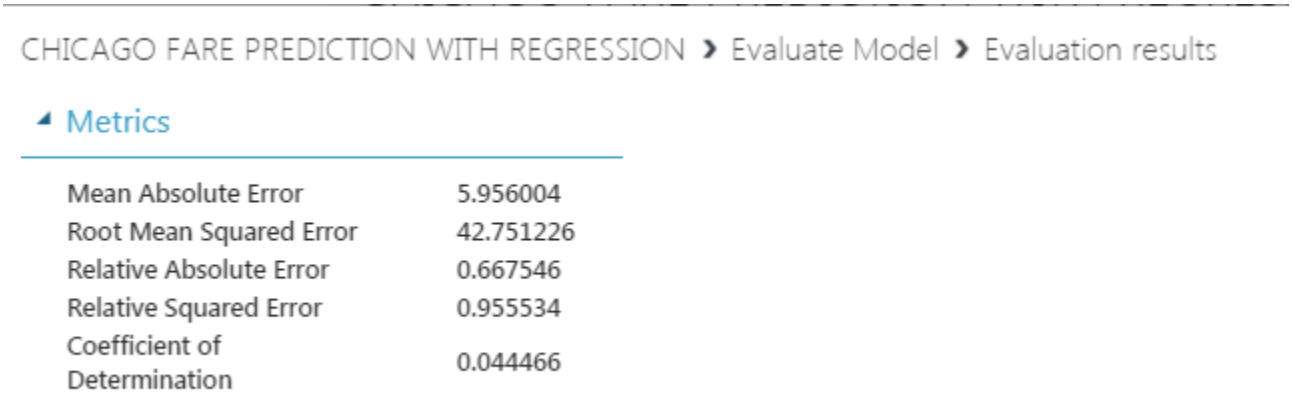


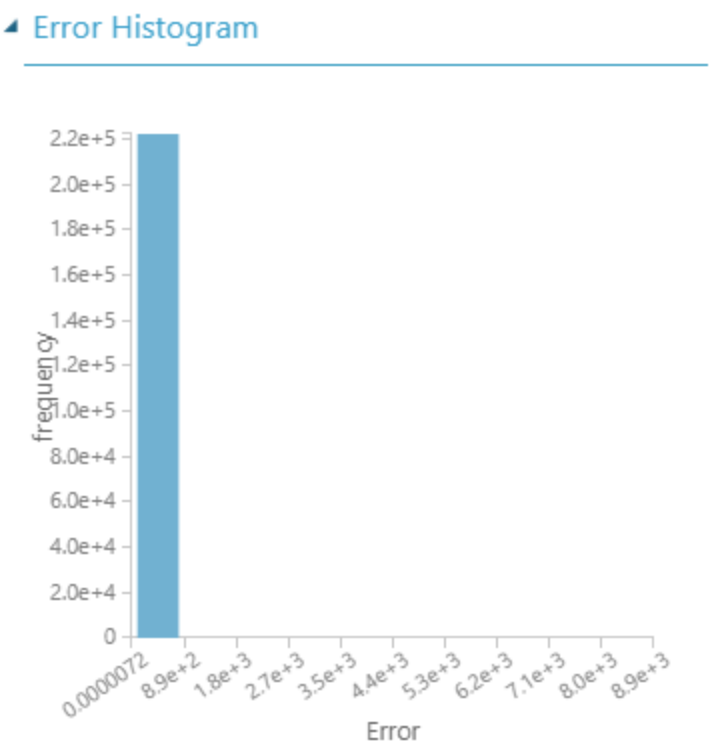
31.Run the experiment and view the output from the **Score Model** module (click the output port of **Score Model** and select **Visualize**). The output shows the predicted values for price and the known values from the test data.



30. 32. Run the experiment.

31. To view the output from the **Evaluate Model** module, click the output port, and then select **Visualize**.





Step 6: Evaluation results for the experiments

The following statistics are shown for our model:+

- **Mean Absolute Error** (MAE): The average of absolute errors (an *error* is the difference between the predicted value and the actual value).
- **Root Mean Squared Error** (RMSE): The square root of the average of squared errors of predictions made on the test dataset.
- **Relative Absolute Error**: The average of absolute errors relative to the absolute difference between actual values and the average of all actual values.
- **Relative Squared Error**: The average of squared errors relative to the squared difference between the actual values and the average of all actual values.
- **Coefficient of Determination**: Also known as the **R squared value**, this is a statistical metric indicating how well a model fits the data.

For each of the error statistics, smaller is better. A smaller value indicates that the predictions more closely match the actual values. For **Coefficient of Determination**, the closer its value is to one (1.0), the better the predictions.

References:

[1] https://docs.microsoft.com/en-us/azure/machine-learning/studio/create-experiment

[2] http://hortonworks.com/hadoop-tutorial/making-things-tick-with-tableau/

[3] http://hadooptutorial.info/tableau-integration-with-hadoop/