



CIS 5200 Term Project Tutorial



Authors:

Smit Shiroya;

Linray Song;

Kathryn Joy Mak;

Pratik Parmar;

Instructor:

[Jongwook Woo](#)

Date: 12/15/2019

Chicago Divvy Bike Sharing Data

Term Project Group 5

Smit Shiroya, Linray Song, Kathryn Joy Mak, Pratik Parmar

Objectives

In this lab you will analyze and visualize Chicago Divvy Bike Sharing Data. Thus,

- You should learn how to download Chicago Divvy Bike Sharing Data to the local systems in AWS EMR.
- Then, you will learn how to upload it to HDFS.
- You will figure out how to manipulate and analyze the data in HDFS using HiveQL.

- You will also practice how to visualize the result in Excel and PowerBI.

Introduction

Divvy is Chicagoland's bike share system, with 6,000 bikes available at 570+ stations across Chicago and Evanston. Divvy provides residents and visitors with a convenient, fun and affordable transportation option for getting around and exploring Chicago. Divvy, like other bike share systems, consists of a fleet of specially designed, sturdy and durable bikes that are locked into a network of docking stations throughout the region. The bikes can be unlocked from one station and returned to any other station in the system. People use bike share to explore Chicago, commute to work or school, run errands, get to appointments or social engagements, and more.

Platform Specification

Cluster Version – AWS EMR
Number of Nodes – 5
Memory size – 150 GB
CPU – 20 vCPU
CPU speed – 2.20 GHz
HDFS capacity – 147 GB
Storage – 678 GB

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:

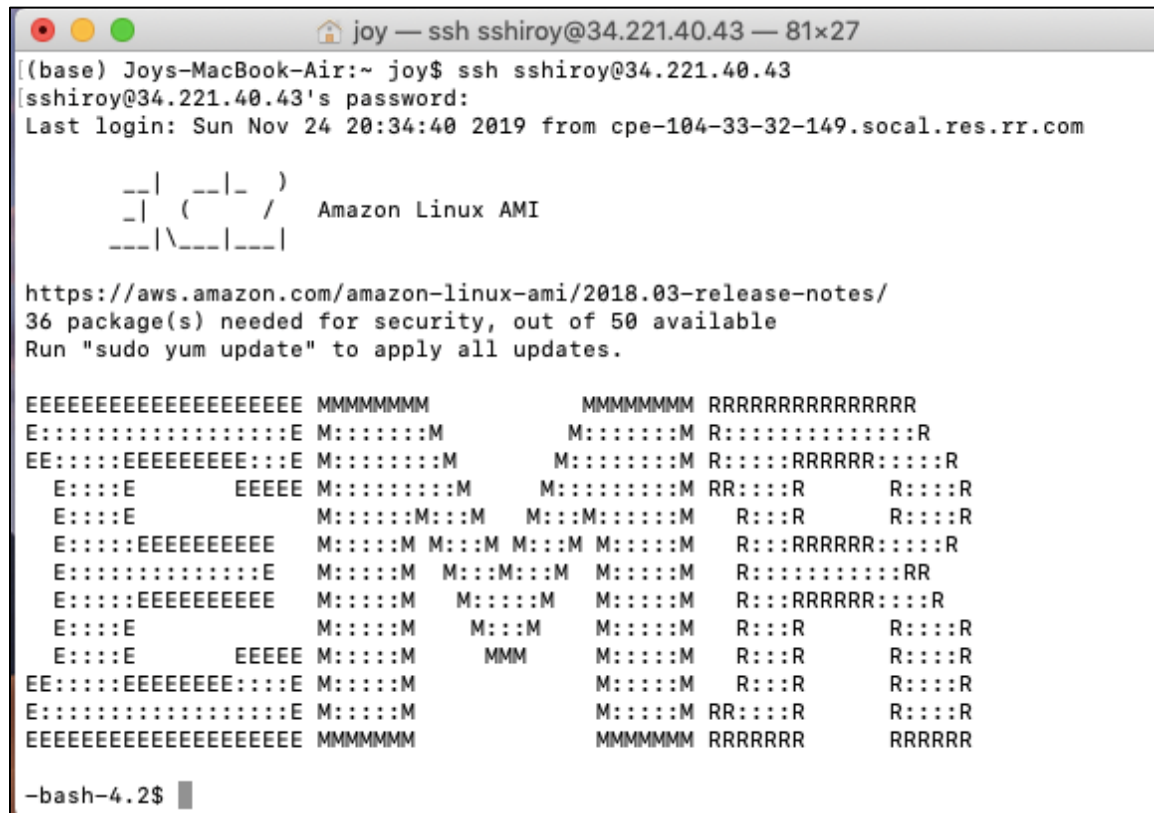
- You must have an ip address to connect to AWS EMR
- You must have **Microsoft Excel 2010, 2013 or 2016** installed.
- You must have an account with the Power-BI.

1. Connect to AWS EMR: Big Data Compute

You need to remotely access your AWS that you executed in your AWS EMR account using *ssh*. For example, for the username and ip address: **sshroy**, you need to run the following with the ip address given:

```
$ ssh sshroy@34.221.40.43
```

When asked for password, type in your username again and enter.



```
joy — ssh sshiroy@34.221.40.43 — 81x27
[(base) Joys-MacBook-Air:~ joy$ ssh sshiroy@34.221.40.43
[sshiroy@34.221.40.43's password:
Last login: Sun Nov 24 20:34:40 2019 from cpe-104-33-32-149.socal.res.rr.com

  _ _ | _ _ | _ _ )
  _ | ( _ _ /   Amazon Linux AMI
  _ _ | \ _ _ | _ _ |

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
36 package(s) needed for security, out of 50 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMMM RRRRRRRRRRRRRRRR
E::::::::::::::::::::E M::::::::M           M::::::::M R::::::::::::R
EE::::::::EEEEEEEE::::E M::::::::M           M::::::::M R::::RRRRRR::::R
  E::::E             EEEEE M::::::::M           M::::::::M RR::::R      R::::R
  E::::E             M::::::::M::M       M:::M:::M:::M   R:::R      R::::R
  E::::EEEEEEEEEEEE   M::::M M:::M M:::M M:::M M:::M   R::RRRRRR::::R
  E::::::::::::::::::E M::::M M:::M::M M:::M M:::M   R::::::::::::RR
  E::::EEEEEEEEEEEE   M::::M M:::M::M M:::M M:::M   R::RRRRRR::::R
  E::::E             M::::M M:::M M:::M M:::M   R:::R      R::::R
  E::::E             EEEEE M::::M       MMM M:::M   R:::R      R::::R
EE::::::::EEEEEEEE::::E M::::M           M::::M   R:::R      R::::R
E::::::::::::::::::E M::::M           M::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMMM RRRRRRR      RRRRRR

-bash-4.2$
```

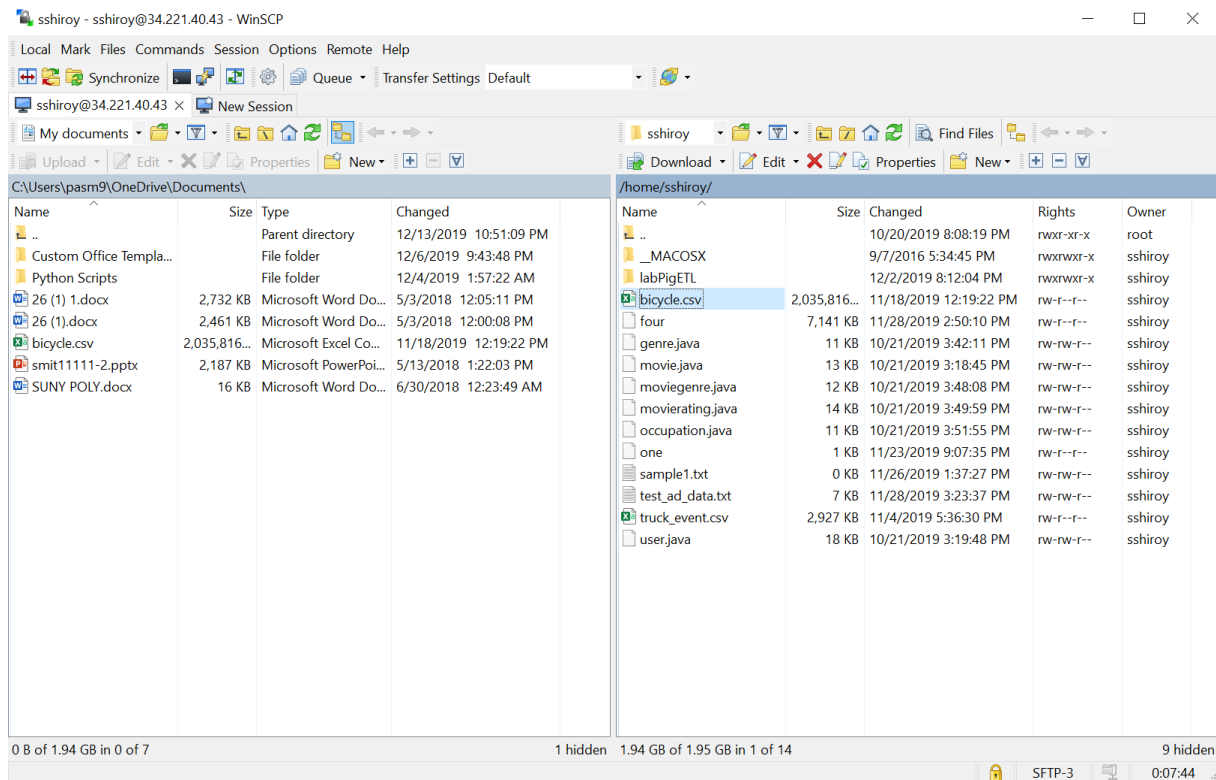
You are now connected to AWS EMR.

2. Upload Data to Server

Server IP: 129.150.122.73

- Download the dataset from the url:
- Use the software WinSCP to upload the dataset bicycle.csv of size 5GB from local system to AWS EMR Server.

You should get something like this: <https://www.kaggle.com/yingwurenjian/chicago-divvy-bicycle-sharing-data>



3. Create directories in HDFS

Run the following commands for creating directories:

```
$ hdfs dfs -mkdir Bicycle_Data
$ hdfs dfs -chmod -R o+w Bicycle_Data
$ hdfs dfs -mkdir Bicycle_Data/data
$ hdfs dfs -chmod -R o+w Bicycle_Data/data
```

You can view the directories created by the following command:

```
$ hdfs dfs -ls Bicycle_Data/data
```

```
[~bash-4.2$ hdfs dfs -ls Bicycle_Data/data
Found 3 items
-rw-r--r--    1 sshiroy hadoop 2084674565 2019-11-18 22:35 Bicycle_Data/data/bicycle.csv
drwxr-xrwx   - sshiroy hadoop          0 2019-11-24 04:54 Bicycle_Data/data/one
drwxr-xrwx   - sshiroy hadoop          0 2019-11-24 06:07 Bicycle_Data/data/two
~bash-4.2$ █
```

4. Put files in HDFS directories

Run the following commands to put dictionaries into respective folders:

```
$ hdfs dfs -put bicycle.csv Bicycle_Data/data
```

Run the following command to provide permission to the files under project folder:

```
hdfs dfs -chmod -R o+w Bicycle_Data/data
```

5. Creating Hive tables to query data

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.

Open another terminal and login into your account using ssh as in Step 1.

Open **beeline** CLI (Command Line Shell Interface) that is equivalent to **hive** CLI environment as follows. **Beeline** is for multiple users' access to Hive Server 2 of a Hadoop cluster. Press enter without any password when it asks for password.

NOTE: the following connect url is an example and it should be given by the instructor:

```
~bash-4.2$ beeline -u jdbc:hive2://localhost:10000/default -n sshiroy
Connecting to jdbc:hive2://localhost:10000/default
```

Connected to: Apache Hive (version 2.3.5-amzn-1)
Driver: Hive JDBC (version 2.3.5-amzn-1)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 2.3.5-amzn-1 by Apache Hive
0: jdbc:hive2://localhost:10000/default>

NOTE: If you see “CLOSED” in the above beeline shell prompt, it is not connected to Hive Server2.

Now you have to create your database with your username to separate your tables with other users. For example, the user **sshiroy** should run the following:

NOTE: you have to use your username.

0: jdbc:hive2://localhost:10000/default> CREATE DATABASE sshiroy; No rows affected (0.277 seconds)

0: jdbc:hive2://localhost:10000/default>show DATABASES;

```
+-----+
| database_name |
+-----+
| abaldaw      |
| amedin62     |
| bfaraji      |
| bliang12     |
| calipio      |
| default      |
| dkansar      |
| dolivas6     |
| dpatel86     |
| group5       |
| hadoop       |
| hchen53      |
| hchen53_1    |
| hchen53_2    |
| jjoshi       |
| jjoshi6      |
| jlogue       |
| jlogue_2     |
| jlu11        |
| jmonte80_2   |
| jmoon14      |
| jmoon14_2    |
| joshi        |
| jwoo5        |
```

```

| kmak6      |
| llin24     |
| llin27     |
| lsong5     |
| mbrumwe    |
| mcuriel8   |
| mcuriel8_2 |
| mduong11   |
| mmanuel5   |
| mmonter3   |
| mmonter32  |
| mrodr456   |
| pparmar2   |
| rsanto33   |
| rsanto33_b |
| skashif    |
| sshinde6   |
| sshiroy    |
| thua       |
| trucks     |
| umuslim2   |
| whe8       |
| ychen148   |
| ychen148_2 |
| yourdpatel86 |
| youruser_name |
| yqian6     |
+-----+

```

51 rows selected (0.219 seconds)

0: jdbc:hive2://localhost:10000/default>use group5; No rows affected

(0.184 seconds)

In the beeline shell CLI, you need to copy and paste the following HiveQL code to create external tables.

NOTE: Don't forget to replace **sshiroy to your account name in the following HQL code.**

```

DROP TABLE IF EXISTS b_data;

CREATE EXTERNAL TABLE IF NOT EXISTS b_data
(trip_id BIGINT, year BIGINT, month BIGINT,
week BIGINT, day BIGINT,
hour BIGINT, usertype STRING, gender STRING,
starttime TIMESTAMP, stoptime TIMESTAMP, tripduration FLOAT, temperature FLOAT, events STRING,
from_station_id BIGINT,
from_station_name STRING, latitude_start FLOAT, longitude_start FLOAT,
dpcapacity_start FLOAT, to_station_id BIGINT, to_station_name STRING)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE LOCATION

'/user/sshiroy/Bicycle_Data/data' TBLPROPERTIES ('skip.header.line.count'='1');

```

The above commands created b_data tables which will later be used for data analysis.

You can query the result by running select statement:

Select * from b_data limit 10;

Then, in the beeline shell, you need to check if the tables are shown:

```
show tables;
```

```
0: jdbc:hive2://localhost:10000/default> show tables;
```

```

+-----+
| tab_name |
+-----+
| b_data   |
| g_type   |
| u_type   |
+-----+

```

3 rows selected (0.259 seconds)

Now you can query the contents of the table:

```
select * from b_data limit 10;
```



```

3 rows selected (0.259 seconds)
0: jdbc:hive2://localhost:10000/default> select * from b_data limit 10;

```

b_data.trip_id	b_data.year	b_data.month	b_data.week	b_data.day	b_data.hour	b_data.usertype	b_data.gender	b_data.starttime	b_data.stoptime	b_data.tripduration
tion	b_data.temperature	b_data.events	b_data.from_station_id	b_data.from_station_name	b_data.latitude_start	b_data.longitude_start	b_data.dpcapacity_start	b_data.to_station_id	b_data.to_station_name	
2355134	2014	6	27	0	23	Subscriber	Male	2014-06-30 23:57:00.0	2014-07-01 00:07:00.0	10.066667
68.0		tstorms	131		Lincoln Ave & Belmont Ave	41.939365		-87.66839	15.0	303
		Broadway & Cornelia Ave								
2355133	2014	6	27	0	23	Subscriber	Male	2014-06-30 23:56:00.0	2014-07-01 00:00:00.0	4.383333
68.0		tstorms	282		Halsted St & Maxwell St	41.86458		-87.64693	15.0	22
		May St & Taylor St								
2355130	2014	6	27	0	23	Subscriber	Male	2014-06-30 23:33:00.0	2014-06-30 23:35:00.0	2.1
68.0		tstorms	327		Sheffield Ave & Webster Ave	41.921688		-87.65372	19.0	225
		Halsted St & Dickens Ave								
2355129	2014	6	27	0	23	Subscriber	Female	2014-06-30 23:26:00.0	2014-07-01 00:24:00.0	58.016666
68.0		tstorms	134		Peoria St & Jackson Blvd	41.87775		-87.649635	19.0	194
		State St & Wacker Dr								
2355128	2014	6	27	0	23	Subscriber	Female	2014-06-30 23:16:00.0	2014-06-30 23:26:00.0	10.633333
68.0		tstorms	320		Loomis St & Lexington St	41.87219		-87.6615	15.0	134
		Peoria St & Jackson Blvd								
2355127	2014	6	27	0	23	Subscriber	Male	2014-06-30 23:11:00.0	2014-06-30 23:17:00.0	5.6
68.0		tstorms	332		Halsted St & Diversey Pkwy	41.933342		-87.64875	15.0	319
		Greenview Ave & Diversey Pkwy								
2355126	2014	6	27	0	23	Subscriber	Male	2014-06-30 23:08:00.0	2014-06-30 23:13:00.0	5.066666
68.0		tstorms	174		Canal St & Madison St	41.88289		-87.63983	23.0	44

You can see the structure of the table as well

```
describe b_data;
```

```

10 rows selected (0.385 seconds)
0: jdbc:hive2://localhost:10000/default> describe b_data;
+-----+-----+-----+
| col_name | data_type | comment |
+-----+-----+-----+
| trip_id | bigint | |
| year | bigint | |
| month | bigint | |
| week | bigint | |
| day | bigint | |
| hour | bigint | |
| usertype | string | |
| gender | string | |
| starttime | timestamp | |
| stoptime | timestamp | |
| tripduration | float | |
| temperature | float | |
| events | string | |
| from_station_id | bigint | |
| from_station_name | string | |
| latitude_start | float | |
| longitude_start | float | |
| dpcapacity_start | float | |
| to_station_id | bigint | |
| to_station_name | string | |
+-----+-----+-----+
20 rows selected (0.326 seconds)
0: jdbc:hive2://localhost:10000/default> █

```

6. Creating Hive Queries to Analyze Data

```
Select * from b_data limit 5;
```

1. The below query will return the user-type count group by types.

```

SELECT usertype, COUNT(usertype) FROM b_data GROUP BY
usertype HAVING COUNT(usertype) > 1;

```

```
0: jdbc:hive2://localhost:10000/default> SELECT usertype, COUNT(usertype) FROM b_data GROUP BY usertype HAVING COUNT(usertype) > 1;
```

usertype	_c1
Dependent	178
Customer	1277
Subscriber	9493780

```
3 rows selected (40.865 seconds)
0: jdbc:hive2://localhost:10000/default> █
```

Now a table will be created using this query and stored in HDFS for visualization at a later stage.

```
CREATE TABLE IF NOT EXISTS u_type ROW FORMAT DELIMITED
FIELDS TERMINATED BY "," STORED AS TEXTFILE LOCATION
"/user/sshiroy/Bicycle_Data/data/one" AS SELECT usertype,
COUNT(usertype) FROM b_data GROUP BY usertype HAVING
COUNT(usertype) > 1;
```

Switch on to the first terminal. You can see the directory “one” has been created under project/tables and if you view the directory you can see a file has been placed there.

```
-bash-4.2$ hdfs dfs -ls Bicycle_Data/data/one
```

Found 2 items

```
-rwxr-xrwx  1 sshiroy hadoop      14 2019-11-24 04:54 Bicycle_Data/data/one/000001_0
-rwxr-xrwx  1 sshiroy hadoop     33 2019-11-24 04:54 Bicycle_Data/data/one/000006_0
```

You can view the contents of the file with the below command:

```
-bash-4.2$ hdfs dfs -cat Bicycle_Data/data/one/000006_0
```

Dependent,178

Customer,1277

Subscriber,9493780

2. The below query will return the gender count.

```
SELECT COUNT(1),gender FROM b_data GROUP BY
gender;
```

```
0: jdbc:hive2://localhost:10000/default> SELECT COUNT(1),gender FROM b_data GROUP BY gender;
```

_c0	gender
1	NULL
2378675	Female
7116560	Male

```
3 rows selected (39.809 seconds)
0: jdbc:hive2://localhost:10000/default> _
```

Now a table will be created using this query and stored in HDFS for visualization at a later stage.

```
CREATE TABLE IF NOT EXISTS g_type ROW FORMAT
DELIMITED FIELDS TERMINATED BY "," STORED AS
TEXTFILE LOCATION
"/user/sshiroy/Bicycle_Data/data/two" AS SELECT
COUNT(1),gender FROM b data GROUP BY gender;
```

Switch on to the first terminal. You can see the directory “two” has been created under project/tables and if you view the directory you can see a file has been placed there.

```
-bash-4.2$ hdfs dfs -ls Bicycle_Data/data/two
```

Found 2 items

```
-rwxr-xrwx  1 sshiroy hadoop      14 2019-11-24 04:54 Bicycle_Data/data/two/000001_0
-rwxr-xrwx  1 sshiroy hadoop     33 2019-11-24 04:54 Bicycle_Data/data/two/000006_0
```

You can view the contents of the file with the below command:

```
-bash-4.2$ hdfs dfs -cat Bicycle_Data/data/two/000006_0
```

3. The below query will return the trips per month.

```
SELECT month, COUNT(month) FROM b_data GROUP BY
month HAVING COUNT(month) > 1;
```

```
0: jdbc:hive2://localhost:10000/default> SELECT month, COUNT(month) FROM b_data GROUP BY month HAVING COUNT(month) > 1;
```

month	_c1
5	869584
10	997712
11	643556
9	1189971
2	305975
7	1279505
12	394672
3	437929
1	271715
6	1178225
4	621688
8	1304703

```
12 rows selected (36.7 seconds)
0: jdbc:hive2://localhost:10000/default>
```

Now a table will be created using this query and stored in HDFS for visualization at a later stage.

```
CREATE TABLE IF NOT EXISTS m_type ROW FORMAT DELIMITED
FIELDS TERMINATED BY "," STORED AS TEXTFILE LOCATION
"/user/sshiroy/Bicycle_Data/data/three" AS SELECT
month, COUNT(month) FROM b_data GROUP BY month HAVING
COUNT(month) > 1;
```

Switch on to the first terminal. You can see the directory “three” has been created under project/tables and if you view the directory you can see a file has been placed there.

```
-bash-4.2$ hdfs dfs -ls Bicycle_Data/data/three
Found 2 items
-rwxr-xrwx  1 sshiroy hadoop      14 2019-11-24 04:54 Bicycle_Data/data/three/000000_0
```

You can view the contents of the file with the below command:

```
-bash-4.2$ hdfs dfs -cat Bicycle_Data/data/three/000000_0
```

4. The below query will visualize the number of bikes rented from different location.

```
SELECT trip_id, starttime, latitude_start,
longitude_start FROM b_data ORDER BY starttime
DESC LIMIT 150000;
```

```
0: jdbc:hive2://localhost:10000/default> SELECT trip_id, starttime, latitude_start, longitude_start FROM b_data ORDER BY starttime DESC LIMIT 150000;
```

trip_id	starttime	latitude_start	longitude_start
17536701	2017-12-31 23:58:00.0	41.90778	-87.68585
17536698	2017-12-31 23:48:00.0	41.929546	-87.64312
17536697	2017-12-31 23:42:00.0	41.954247	-87.6544
17536696	2017-12-31 23:41:00.0	41.91369	-87.652855
17536695	2017-12-31 23:34:00.0	41.896545	-87.63093
17536694	2017-12-31 23:21:00.0	41.939743	-87.65887
17536693	2017-12-31 23:17:00.0	41.88132	-87.629524
17536692	2017-12-31 22:57:00.0	41.89057	-87.62207
17536690	2017-12-31 22:50:00.0	41.867226	-87.62596
17536691	2017-12-31 22:50:00.0	41.867226	-87.62596
17536689	2017-12-31 22:48:00.0	41.968987	-87.69603
17536688	2017-12-31 22:45:00.0	41.925602	-87.65371
17536687	2017-12-31 22:42:00.0	41.922695	-87.69715
17536686	2017-12-31 22:09:00.0	42.02102	-87.665085
17536685	2017-12-31 22:07:00.0	41.89057	-87.62207
17536684	2017-12-31 22:05:00.0	41.96522	-87.65814
17536626	2017-12-31 20:57:00.0	41.96909	-87.67424
17536616	2017-12-31 20:53:00.0	42.00445	-87.6724
17536613	2017-12-31 20:53:00.0	41.96909	-87.67424
17536612	2017-12-31 20:49:00.0	41.858166	-87.656494
17536611	2017-12-31 20:46:00.0	41.857613	-87.61941
17536610	2017-12-31 20:44:00.0	41.96909	-87.67424
17536609	2017-12-31 20:42:00.0	41.954178	-87.66436
17536601	2017-12-31 20:34:00.0	41.90231	-87.62769
17536598	2017-12-31 20:33:00.0	41.91369	-87.652855
17536593	2017-12-31 20:26:00.0	41.89766	-87.62351
17536592	2017-12-31 20:26:00.0	41.95283	-87.649994
17536591	2017-12-31 20:22:00.0	41.88103	-87.624084
17536589	2017-12-31 20:17:00.0	41.949276	-87.6463
17536580	2017-12-31 20:08:00.0	41.890762	-87.6317
17536581	2017-12-31 20:08:00.0	41.88635	-87.617516
17536582	2017-12-31 20:08:00.0	42.008595	-87.69049
17536579	2017-12-31 20:05:00.0	41.89691	-87.62174
17536578	2017-12-31 20:05:00.0	41.909397	-87.67769
17536577	2017-12-31 20:01:00.0	41.935734	-87.663574
17536529	2017-12-31 18:59:00.0	41.87785	-87.62408
17536521	2017-12-31 18:56:00.0	42.007973	-87.665504
17536522	2017-12-31 18:56:00.0	41.940105	-87.645454
17536518	2017-12-31 18:55:00.0	41.967094	-87.67903
17536517	2017-12-31 18:52:00.0	41.857555	-87.66154
17536516	2017-12-31 18:52:00.0	41.936497	-87.64754
17536515	2017-12-31 18:50:00.0	41.893993	-87.62932
17536512	2017-12-31 18:49:00.0	41.856594	-87.62754
17536513	2017-12-31 18:49:00.0	41.856594	-87.62754
17536511	2017-12-31 18:48:00.0	41.929142	-87.64908
17536509	2017-12-31 18:45:00.0	41.892155	-87.636925

Now a table will be created using this query and stored in HDFS for visualization at a later stage.

```
CREATE TABLE IF NOT EXISTS l_type ROW FORMAT DELIMITED
FIELDS TERMINATED BY "," STORED AS TEXTFILE LOCATION
"/user/sshiroy/Bicycle_Data/data/four" AS SELECT
trip_id, starttime, latitude_start, longitude_start FROM
b_data ORDER BY starttime DESC LIMIT 150000;
```

Switch on to the first terminal. You can see the directory “four” has been created under project/tables and if you view the directory you can see a file has been placed there.

```
-bash-4.2$ hdfs dfs -ls Bicycle_Data/data/four
```

Found 1 items

```
-rwxr-xrwx 1 sshiroy hadoop 7311602 2019-11-28 22:47 Bicycle_Data/data/four/000000_0
```

You can view the contents of the file with the below command:

```
-bash-4.2$ hdfs dfs -cat Bicycle_Data/data/four/000000_0
```

7. Downloading data into your PC

After the Hive tables are created, you can download it to your lab (or personal PC/Laptop) as follows.

1. Switch on to the first terminal connected to the AWS EMR to download the output file

```
$ ssh sshiroy@ipaddress  
sshiroy@ipaddress's password:
```

Run the following command to check if files are present:

```
-bash-4.1$ hdfs dfs -ls Bicycle_Data/data/one
```

```
-bash-4.2$ hdfs dfs -ls Bicycle_Data/data/one
```

Found 2 items

```
-rwxr-xrwx  1 sshiroy hadoop      14 2019-11-24 04:54 Bicycle_Data/data/one/000001_0  
-rwxr-xrwx  1 sshiroy hadoop     33 2019-11-24 04:54 Bicycle_Data/data/one/000006_0
```

You will see only one file named 000000_0 is present in all the above folders except at Bicycle_Data/data/one/.

2. Download the file to the local file systems and rename it

Since all the folders have the same name file 000000_0, after downloading into local file system you will rename the file

```
-bash-4.1$ hdfs dfs -get Bicycle_Data/data/one/000006_0
```

```
-bash-4.1$ ls
```

```
000000_0 __MACOSX
```

```
-bash-4.1$ mv 000006_0 one
```

```
-bash-4.1$ ls
```

```
__MACOSX one
```

Similarly, do it for others

```
-bash-4.1$ hdfs dfs -get project/tables/two/000000_0
```

```
-bash-4.1$ mv 000000_0 two
```

```
-bash-4.1$ hdfs dfs -get project/tables/three/000000_0
```

```
-bash-4.1$ mv 000000_0 three
```

```
-bash-4.1$ hdfs dfs -get project/tables/four/000000_0
```

```
-bash-4.1$ mv 000000_0 four
```

3. Open another terminal in order to import the output file using your lab computer (or your PC/Laptop)
- you have to download the file to your lab computer (or your PC/Laptop). For example, your output file at the AWS EMR server is located at /home/sshiroy/one and remotely copied to the file "one.csv".

When asked for password, provide the password

```
$ scp sshiroy@ipaddress:/home/sshiroy/one one.csv
$ scp sshiroy@ipaddress:/home/sshiroy/two two.csv
$ scp sshiroy@ipaddress:/home/sshiroy/three three.csv
$ scp sshiroy@ipaddress:/home/sshiroy/four four.csv
```

```
[Joys-MacBook-Air:~ joy$ scp sshiroy@34.221.40.43:/home/sshiroy/four four.csv
sshiroy@34.221.40.43's password:
four
100% 7140KB 2.5MB/s 00:02
```

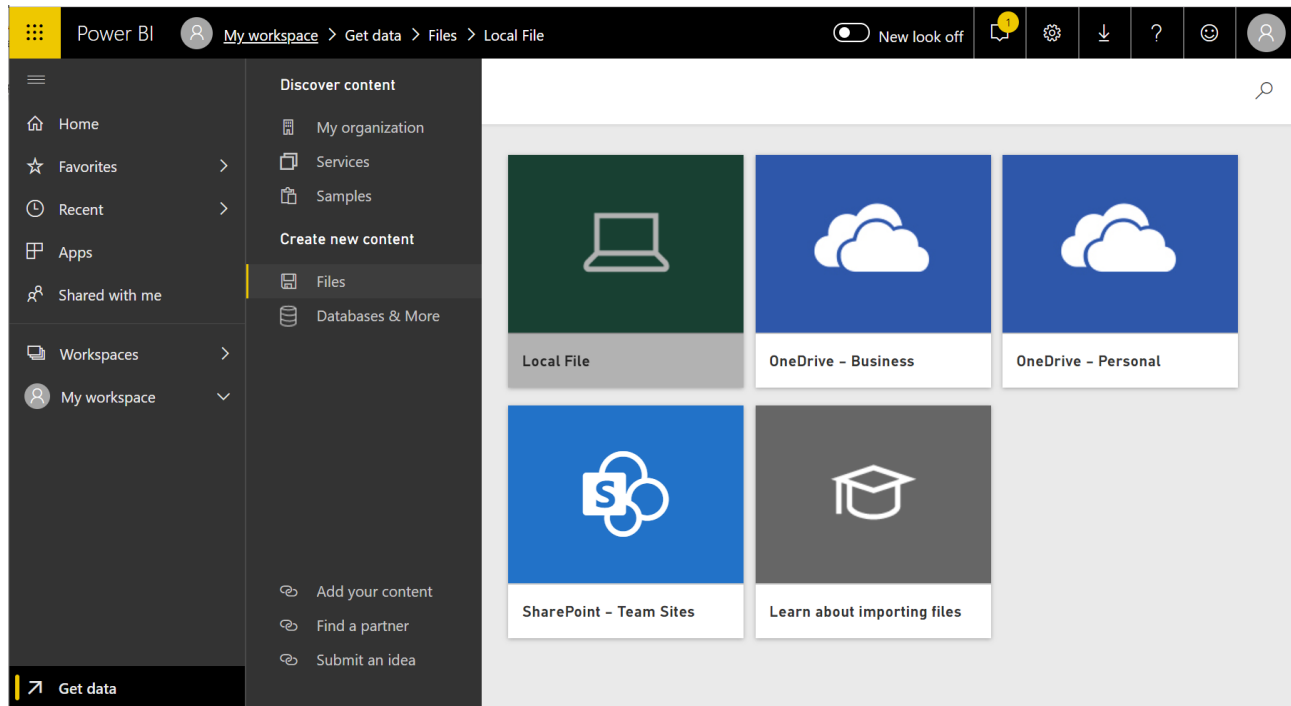
Similarly do it for others.

8. Visualizing data

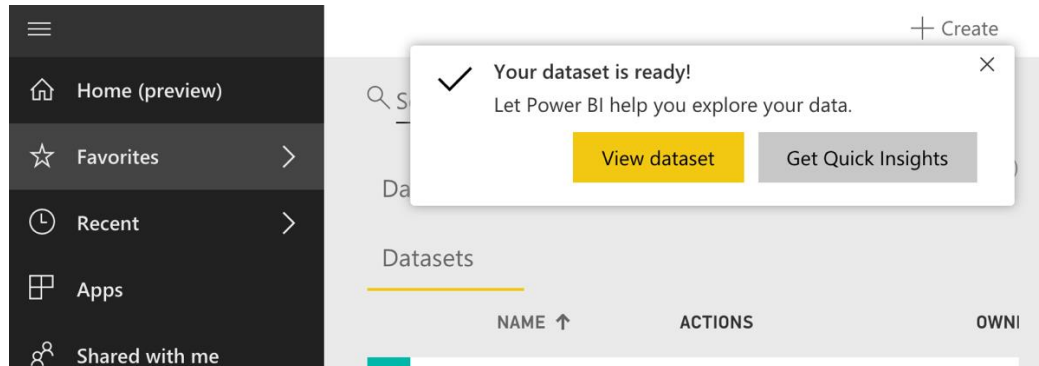
1. Breakdown of User Type

- a. Open the one.csv file with Microsoft Excel and insert column heading as Review Count and Rating. Save the document.
- b. Open a web browser and go to and sign in with your school account at:
<https://app.PowerBI.com>

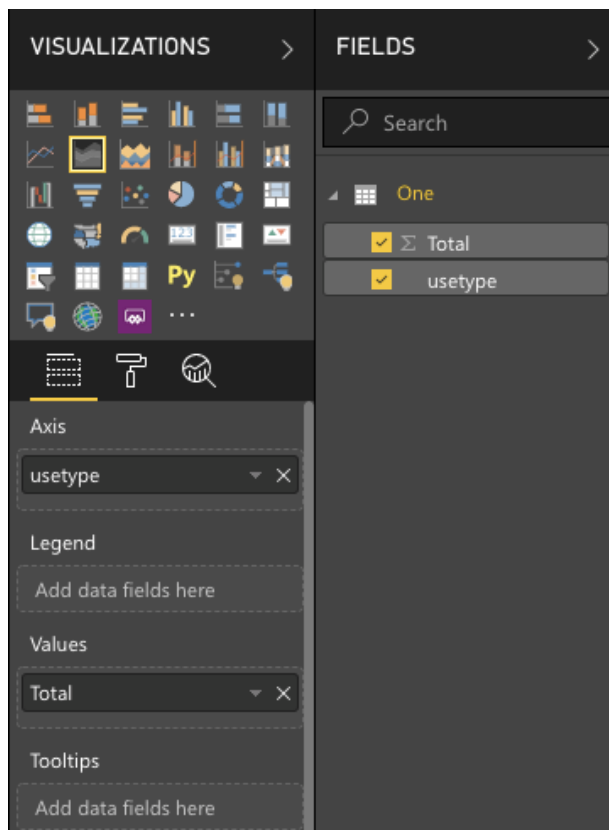
c. Click on Local File and select the one.csv



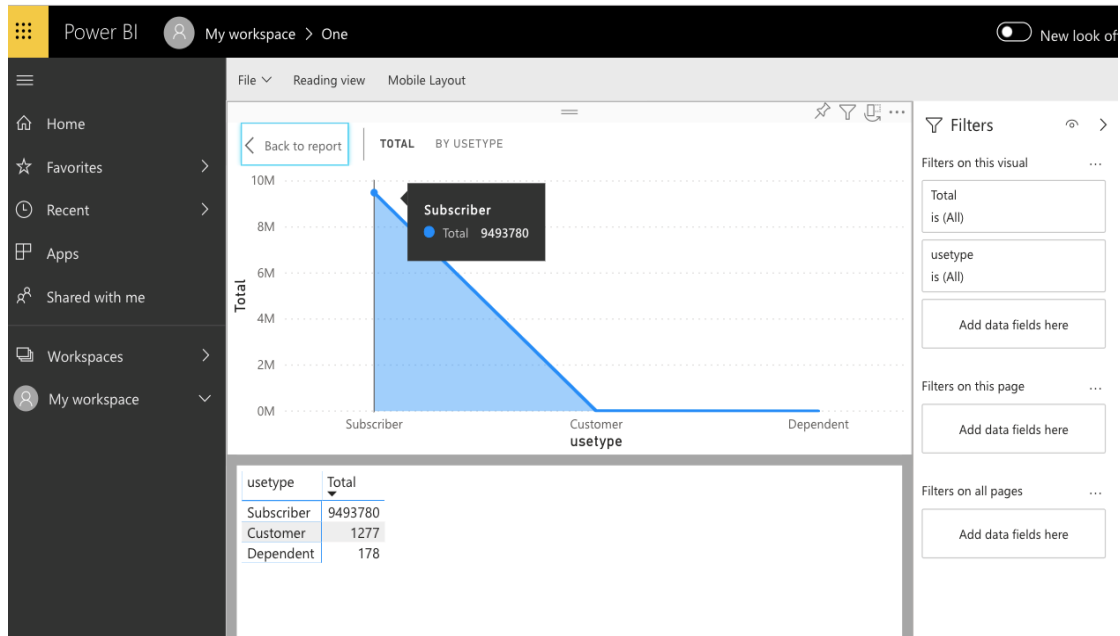
d. Click on View dataset once the file is uploaded



- e. Click on Area Chart under VISUALIZATIONS.
Drag user type and total under axis. Drag Total under values.

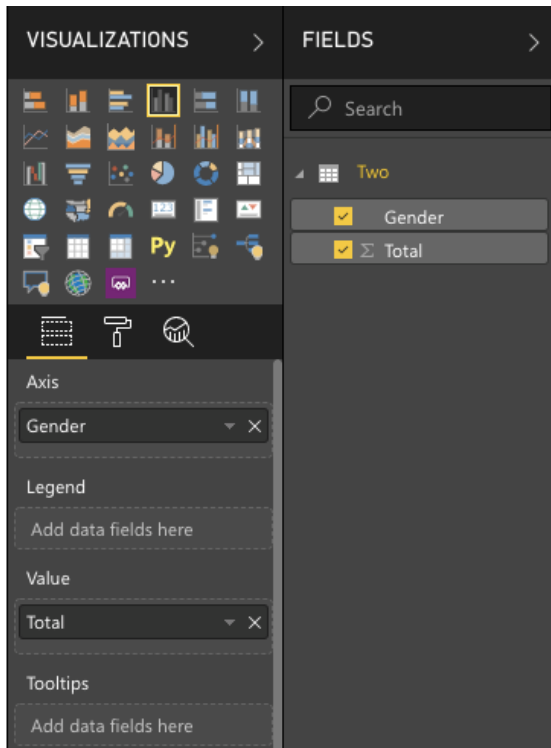


- f. Click on the format under VISUALIZATIONS and switch on the Legend
You can customize colors and other features using format



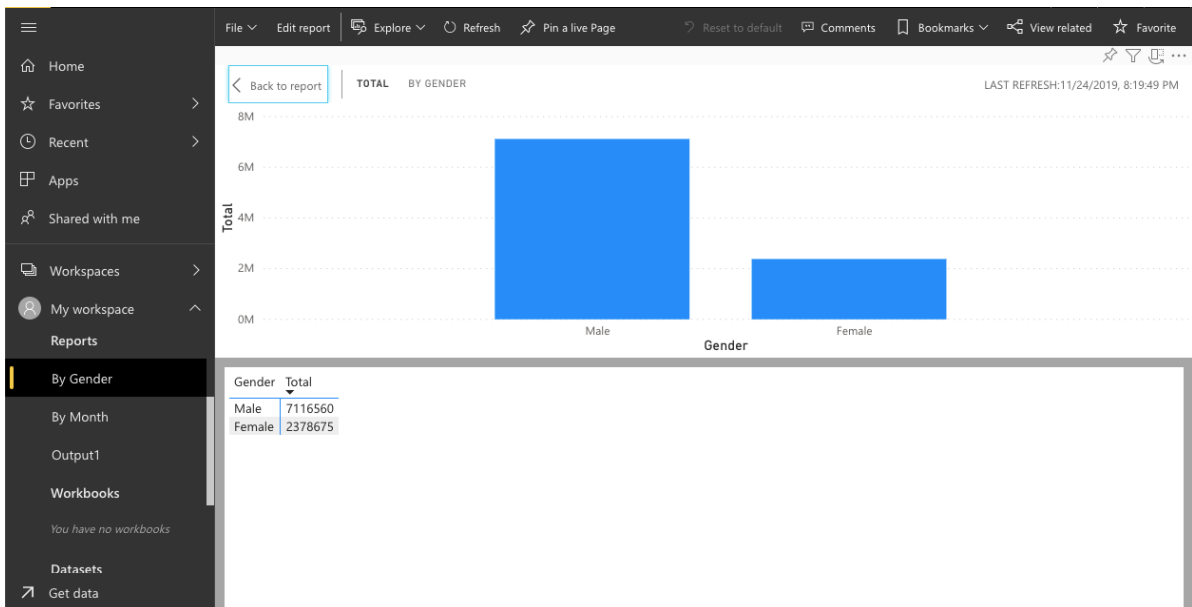
2. Gender Gap

- Open the two.csv file with Microsoft Excel and insert column heading as Review Count and User. Save the document.
- Open a web browser and go to and sign in with your school account at: <https://app.PowerBI.com>
- Click on Local File and select the two.csv
- Click on View dataset once the file is uploaded
- Click on Clustered column chart under VISUALIZATIONS. Drag gender under Axis, Drag Total Count under Value.



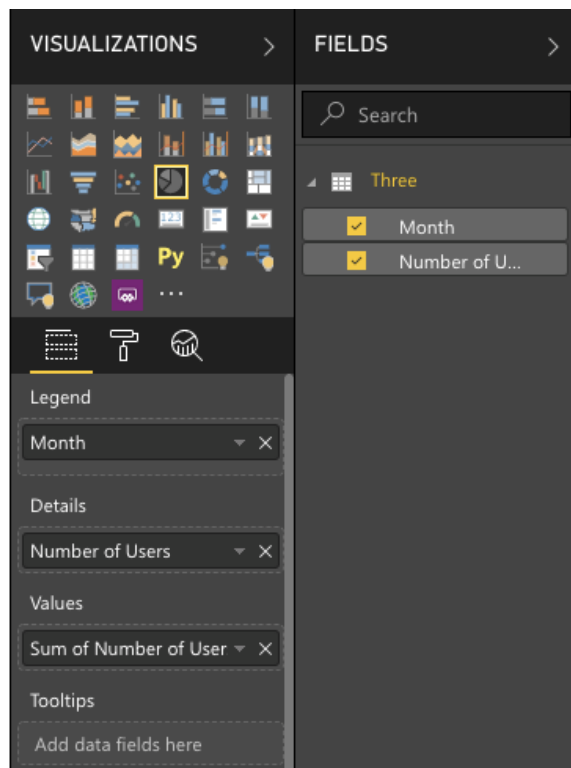
- f. Click on the format under VISUALIZATIONS and select type as “Gender” under X-Axis.

You can customize colors and other features using format

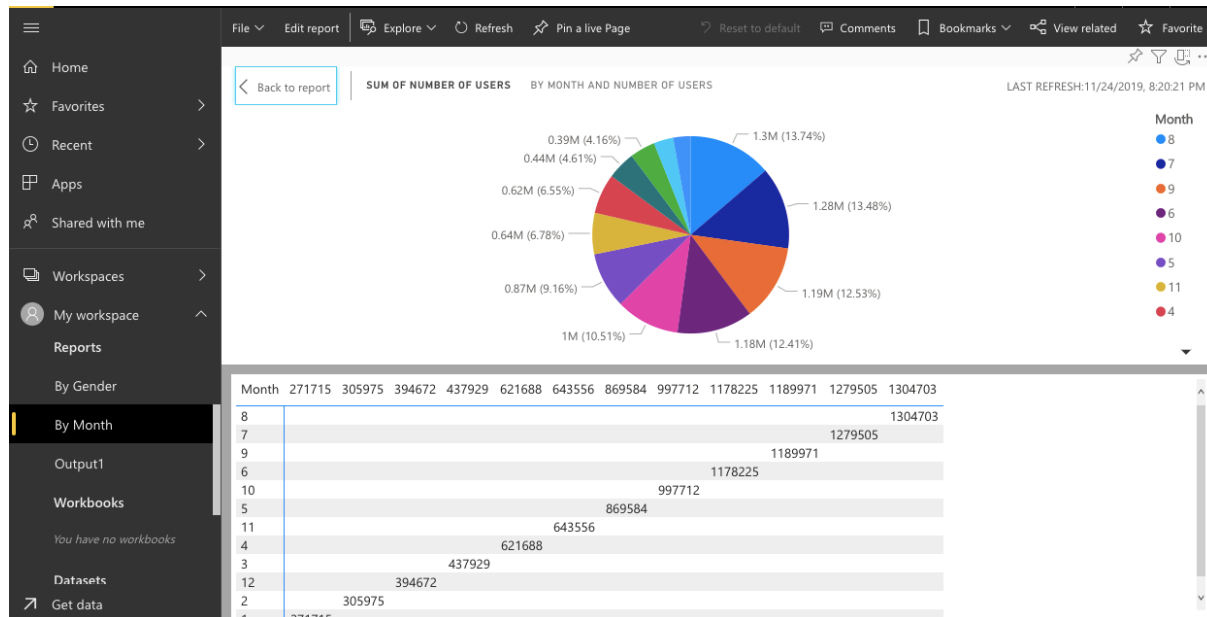


3. Percentage of Monthly Usage

- a. Open the three.csv file with Microsoft Excel and insert column heading as Count, Year and Month. Save the document.
- b. Open a web browser and go to and sign in with your school account at:
<https://app.PowerBI.com>
- c. Click on Local File and select the three.csv
- d. Click on View dataset once the file is uploaded
- e. Click on Pie chart under VISUALIZATIONS.
Drag month under Axis.
Drag number of users under detail.
Select sum of users under value.
- f. You will display values from January to December and so under filter section for month select as per the below screenshot and click apply filter

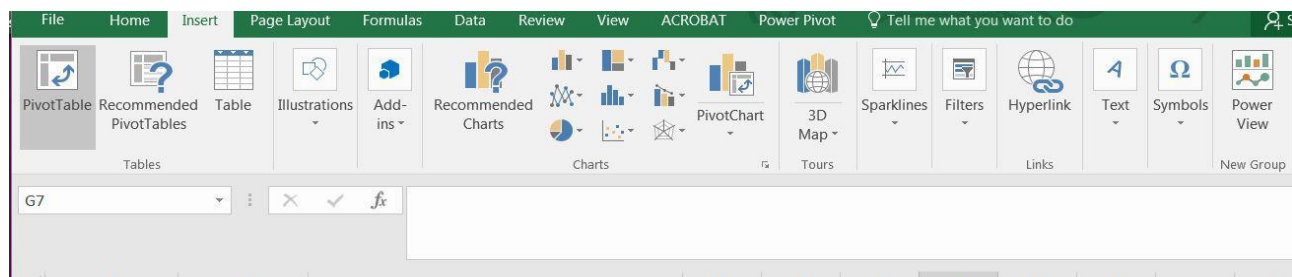


The visualization will be like as below:

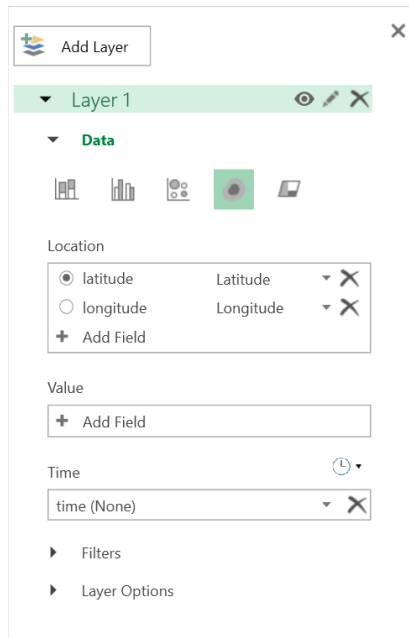


4. Distribution of Divvy Stations across Chicago

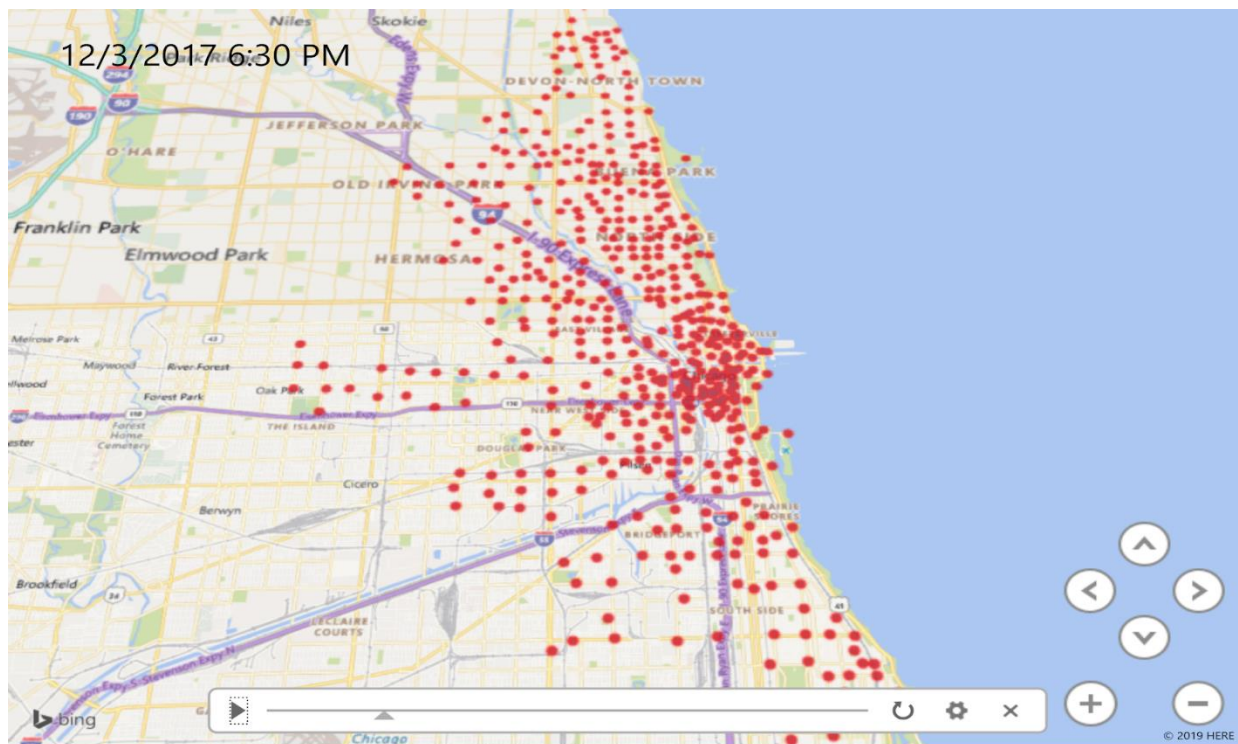
- Open the four.csv file with Microsoft Excel and insert column heading as Trip Id, Start, Longitude Start and Latitude Start. Save in.xlsx.
- Open four.xlsx in MS-excel. Go to Insert tab and click on 3D Map.



- You will see the 3D map.
NOTE: If you don't see the layer frame in the right side, you may select all data manually before opening 3D map
- Add fields as shown in the picture below



- e. You will get a view like below. You can click on **play** button to observe from where bikes have been rented



Summary

In this tutorial you learned how AWS EMR Big Data can be used to analyze different patterns of raw data using Apache Hive. You went through a flow to understand how the raw data is first uploaded to HDFS, and then loaded to Hive tables for performing queries. And, you learned how to import the results of Hive queries into Microsoft Excel and Power-BI. Finally, you learned how to create visualizations using Power-BI and 3D Map chart in MS-excel.

References:

1. Ink, Social. "84 Million Trips Taken on Shared Bikes and Scooters Across the U.S. in 2018." *National Association of City Transportation Officials*, 17 Apr. 2019, nacto.org/2019/04/17/84-million-trips-on-shared-bikes-and-scooters/.
2. Divvy Bikes. (2019). *Motivate International, Inc. "About Divvy: Company & History"*. Retrieved from www.divvybikes.com/about
3. Zhoa, J. (2017). *Chicago Divvy Bicycle Sharing Data*. Retrieved from <https://www.kaggle.com/yingwurenjian/chicago-divvy-bicycle-sharing-data#data.csv>