Big Data Course Project
Topic: Weather and Taxi Data
Group members : Vivek Ghatala (vrj247), Jil Kapashi (jmk854) and Shyam Joshi (srj295)

Dataset to be used:
NYC Taxi Data
NYC Weather Data
NYC MTA Data
NYC Subway Stations Co-oridnates data

Big Data Project GitHub Link :-
https://github.com/shyamrjoshi/Big-Data-Project

Project Report

Aim :-
The main aim of our analysis is to use NYC Taxi Data which when integrated with the Weather data and the Subway turnstile data provides us with a deeper insight about the usage of NYC Taxis and how it changes with Weather and how it impacts the subway usage by the population. We start by discussing what questions we want to address and then move on to the steps taken by us for our analysis, we also mention the difficulties which we encountered and how we solved them.

Questions identified to be answered :-
The questions which we came up with to be answered are as follows :-

1. How many people take the taxi and get down near a subway station (i.e the distance between the dropoff point and the nearest subway station should be less than 0.1 miles, as it is a walkable distance) and how the weather affects them? We list the top 10 subways as the output for each month and each of them for rain and snow.

2. What is the trend regarding the number of people taking taxis at a particular time of the day versus the number of people travelling by subways during particular peak hours of the day?

Experience :-

Coming up with these questions was difficult as we wanted to identify some unique questions using the datasets we had chosen. The first problem we encountered was to think the above questions through and how we were going to proceed for our analysis and what might our output look like and how the datasets would interact with each other, identifying common attributes among them and also understanding what each and every attribute stands for and also what a particular value for that attribute indicates about it and how it would be useful for our analysis. Understanding the metadata took a lot of time as there were columns with the same name occurring multiple times in the dataset (i.e MW and AW attributes in the Weather data), also we were posed with a situation of cleaning the data first, we have cleaned the NYC Taxi Data and also the Weather data according to our needs. (Like weather data might have values in any one of the three AW or MW attributes stated above so we basically had to go through the whole data in order to figure out what attributes we needed to consider)
First we started with MapReduce as it was the thing we were comfortable in, but later realised that it would take a lot of time for processing and hence we moved on to spark. However the

initial cleaning of the NYC Taxi Data and the weather data has been done through MapReduce as it was simple to implement as a starting point. With Spark came the challenges of learning Spark in detail for which we had to go through various materials and with a collaborative team effort, we were able to come up with solutions to answer both of our questions.

Question 1:-

How many people take the taxi and get down near a subway station (i.e the distance between the dropoff point and the nearest subway station should be less than 0.1 miles, as it is a walkable distance) and how the weather affects them? We list the top 10 subways as the output for each month and each of them for rain and snow.
Before reproducing any of the below mentioned Tasks please note that the header lines should be removed from all csv files before proceeding. All codes are run on Amazon EMR cluster. Upload all the files to S3 bucket.

Method/Algorithm for Question 1:-

Datasets Used :- NYC Taxi Data, Weather Data and Subway Stations Coordinates Data
Steps Performed :-

1. Cleaning NYC Taxi Data for dirty trips like once with no coordinates.
2. Cleaning weather data for changing GMT time to EST
3. First we joined taxi data with the cleaned weather data to identify the trips when there was rain or snow based on the values of MW and AW and the time of the trip and the weather at that hour.
4. Then we performed a cross product of the output received in step 3 with the Subway Stations Coordinates data to identify the trips whose dropoff locations are within 0.1 miles of the subway locations by checking the distance of dropoff location from each subway station.
5. The output in step 4 is the input for the program which filter the top 10 subway stations and the output of this step is displayed using a D3 Bar Graph.

Instructions/Commands for usage :-

First clean the NYC taxi data using the map.py and reduce.py files provided by creating a new cluster and then adding a step and selecting Streaming program and the respective map.py and reduce.py files from your bucket and then select NYC taxi data as the input file and then crating a folder for your output. No of reducers used is 1. Type “-D mapreduce.job.reduces=1” in the options field.

Then Clean the weather data using spark the command will be
Get the map file from your bucket into hdfs
Hdfs dfs -get s3://yourpath map.py
Hdfs dfs -put map.py map.py

Get the weather data from your bucket into hdfs
Hdfs dfs -get s3://yourpath weatherdata.csv
Hdfs dfs -put weatherdata.csv weatherdata.csv

Running the cleaning script
Spark-submit map.py weatherdata.csv >> cleanedweatherdata.csv

Now use the join.py in your bucket and moving it to hdfs
Hdfs dfs -get s3://yourpath taxi_weather_rain_join.py
Hdfs dfs -put taxi_weather_rain_join.py taxi_weather_rain_join.py

Now, get the output of the NYC taxidata from your bucket into hdfs :-
Hdfs dfs -get s3://yourpath/part-00000 cleanedtaxidata.csv
Hdfs dfs -put cleanedtaxidata.csv cleanedtaxidata.csv

Now, use the output of the Clean on the weather data and the NYC taxidata as the input for the taxi_weather_rain_join.py spark program
Spark-submit taxi_weather_rain_join.py cleanedtaxidata.csv cleanedweatherdata.csv >> taxiweatherjoinoutputformonth.csv

Now, move the output of this join to your bucket
Hdfs dfs -put taxiweatherjoinoutputformonth.csv taxiweatherjoinoutputformonth.csv
Hdfs dfs -put taxiweatherjoinoutputformonth.csv
s3://yourpath/taxiweatherjoinoutputformonth.csv

Now, use the output of the join for the input to the crossproduct
First, move the subwaycross.py file from your s3 bucket to hdfs
Hdfs dfs -get s3://yourbucket/subwaycross.py subwaycross.py
Hdfs dfs -put subwaycross.py subwaycross.py

Now, move the subway.csv(containing the subway stations with their coordinates) into hdfs
Hdfs dfs -get s3://yourbucket/subway.csv subway.csv
Hdfs dfs -put subway.csv subway.csv
Spark-submit subwaycross.py taxiweatherjoinoutputformonth.csv subway.csv >> subwaytaxiweatherjoincross.csv

Now, move the output of this crossproduct to your bucket
Hdfs dfs -put subwaytaxiweatherjoincross.csv subwaytaxiweatherjoincross.csv
Hdfs dfs -put subwaytaxiweatherjoincross.csv s3://yourpath/subwaytaxiweatherjoincross.csv

Now, move the subwaystats.py(to find out the top 10 subway stations) into hdfs
Hdfs dfs -get s3://yourbucket/subwaystats.py subwaystats.py
Hdfs dfs -put subwaystats.py subwaystats.py

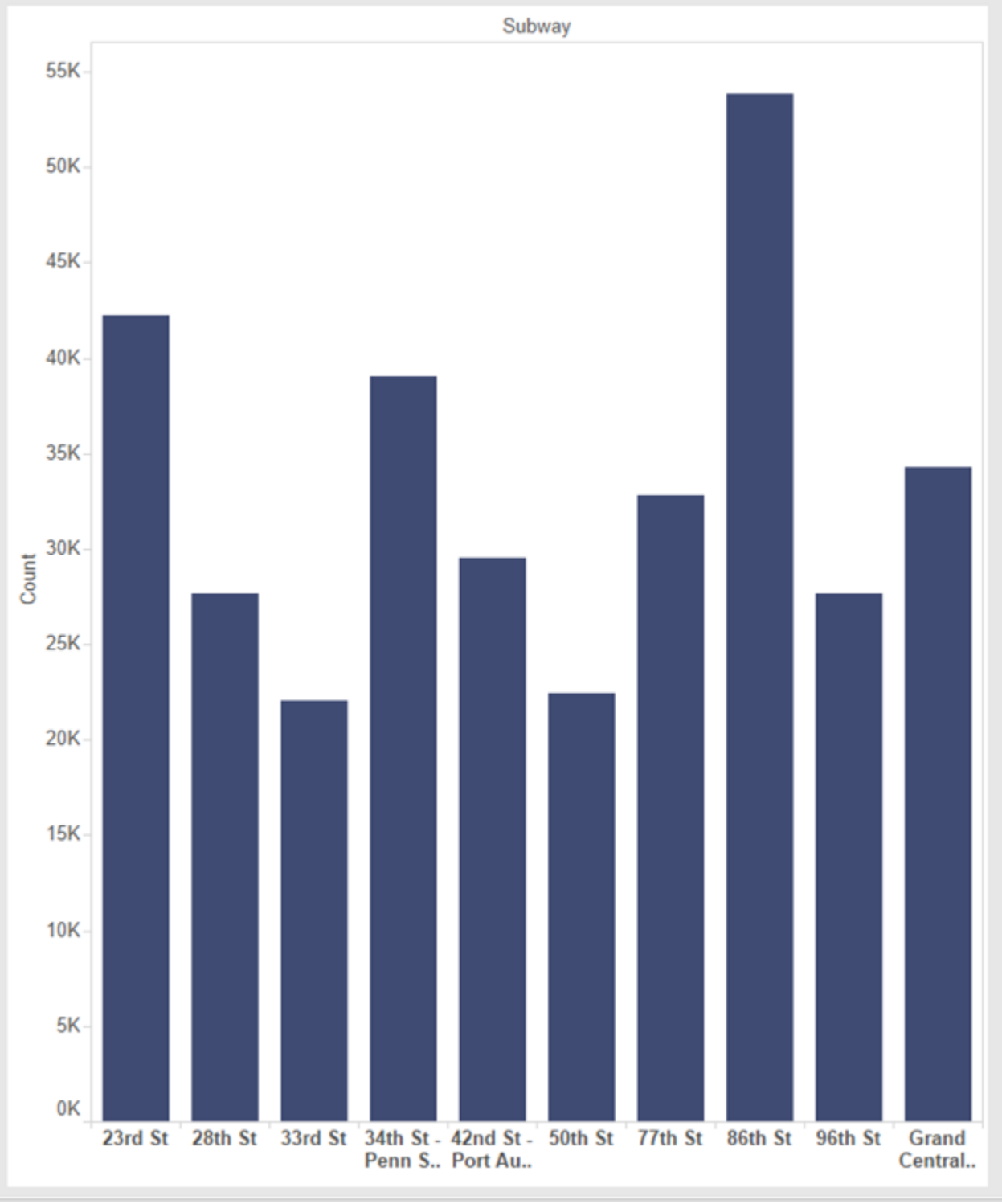
Now use the output of the cross product as the input for subwaystats.py
Spark-submit subwaystats.py subwaytaxiweatherjoincross.csv >> toptensubway.csv
Move this file to your bucket
Hdfs dfs -put toptensubway.csv s3://yourbucket/toptensubway.csv

Repeating the same procedure for snow just replace the join.py file by taxi_weather_snow_join.py file during the join step

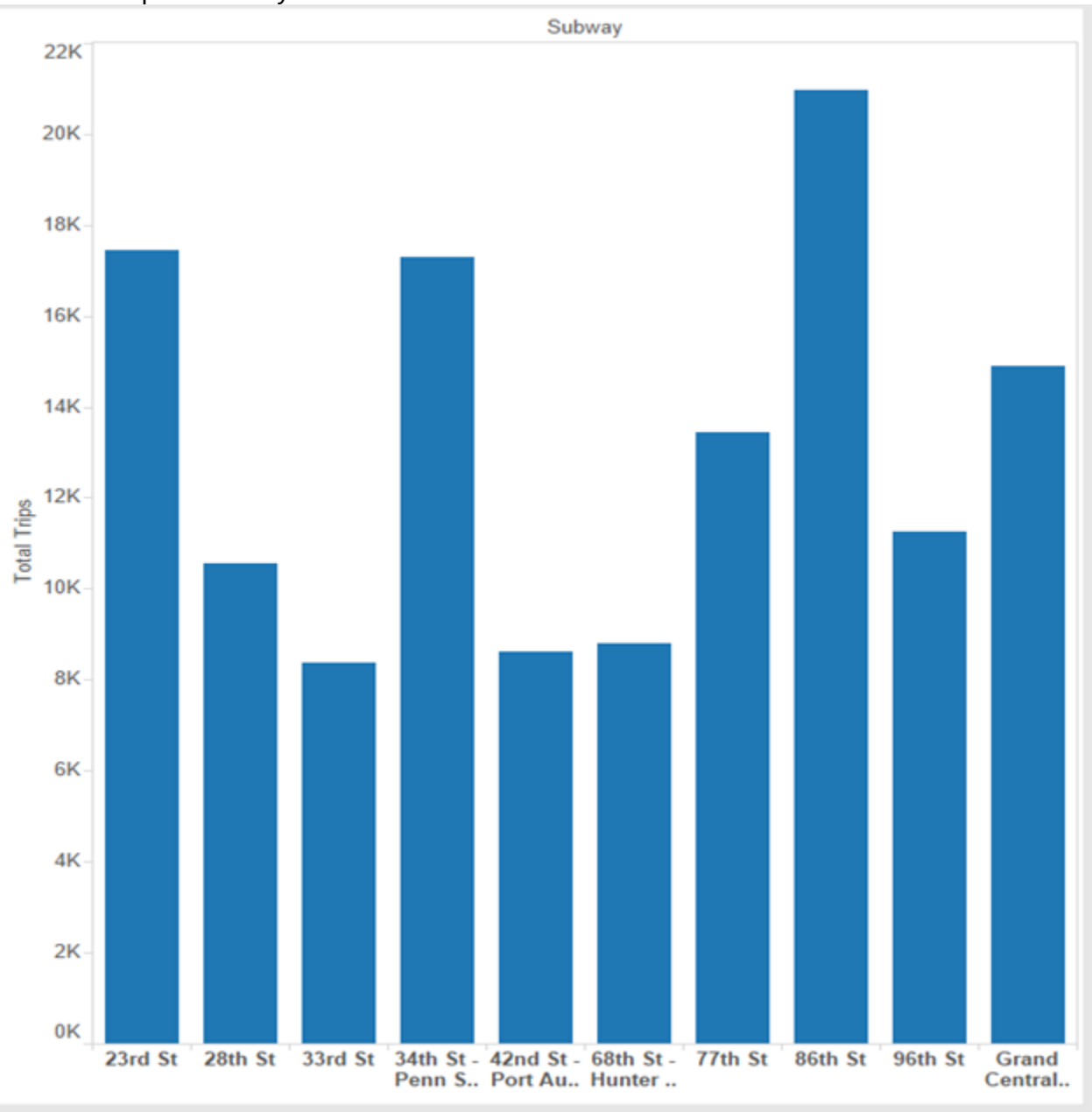
Now, we performed a D3 bar graph visualization on this output. In order to run our D3 code you just need to change the file being passed inside D3 .js file
D3 visualizations reference :- D3 Official Website
https://github.com/d3/d3/wiki/Gallery
http://bl.ocks.org/Caged/6476579

The output of our D3 visualizations is as below for rain and snow :-

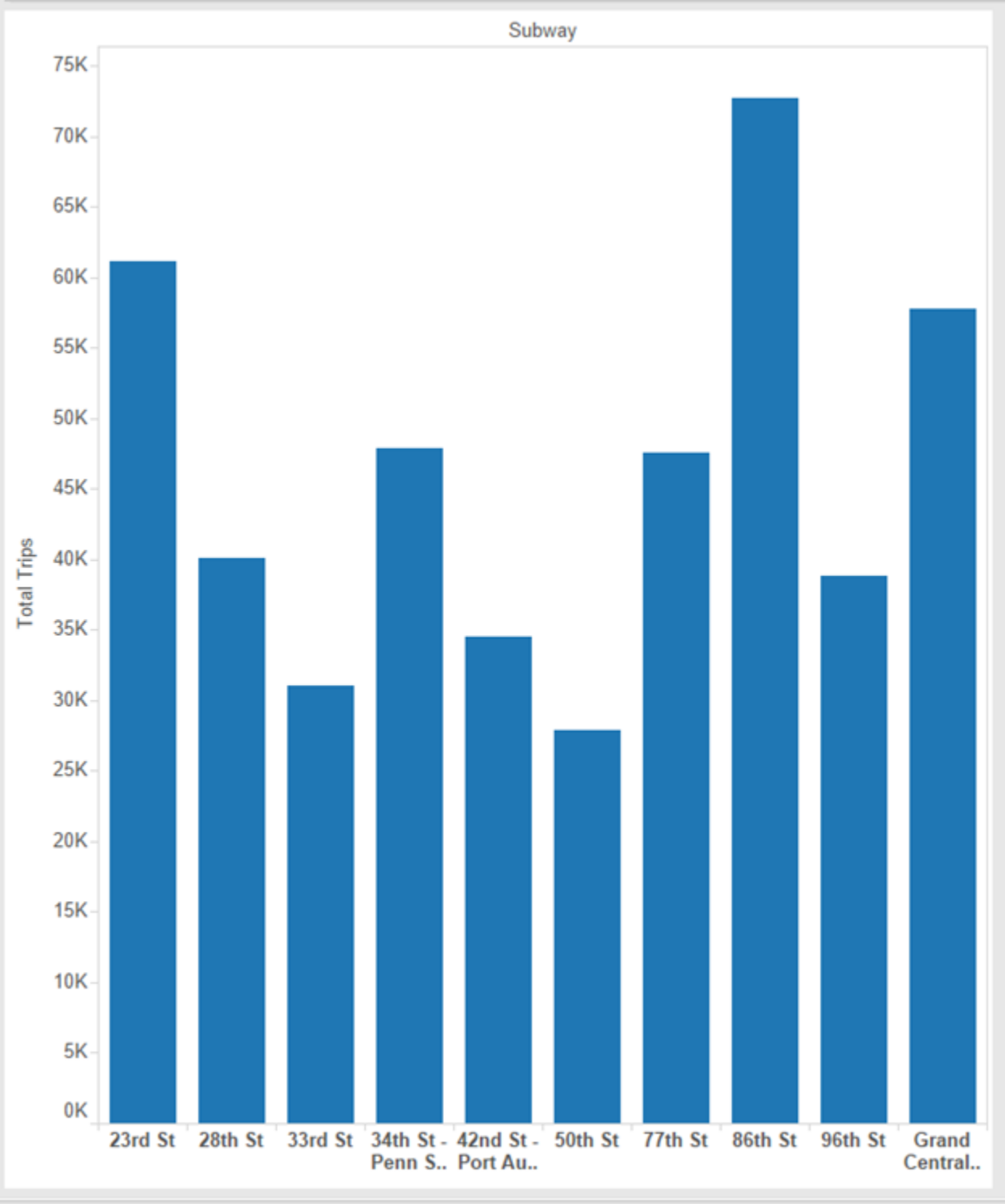
Rain Jan Top 10 Subway Stations :-



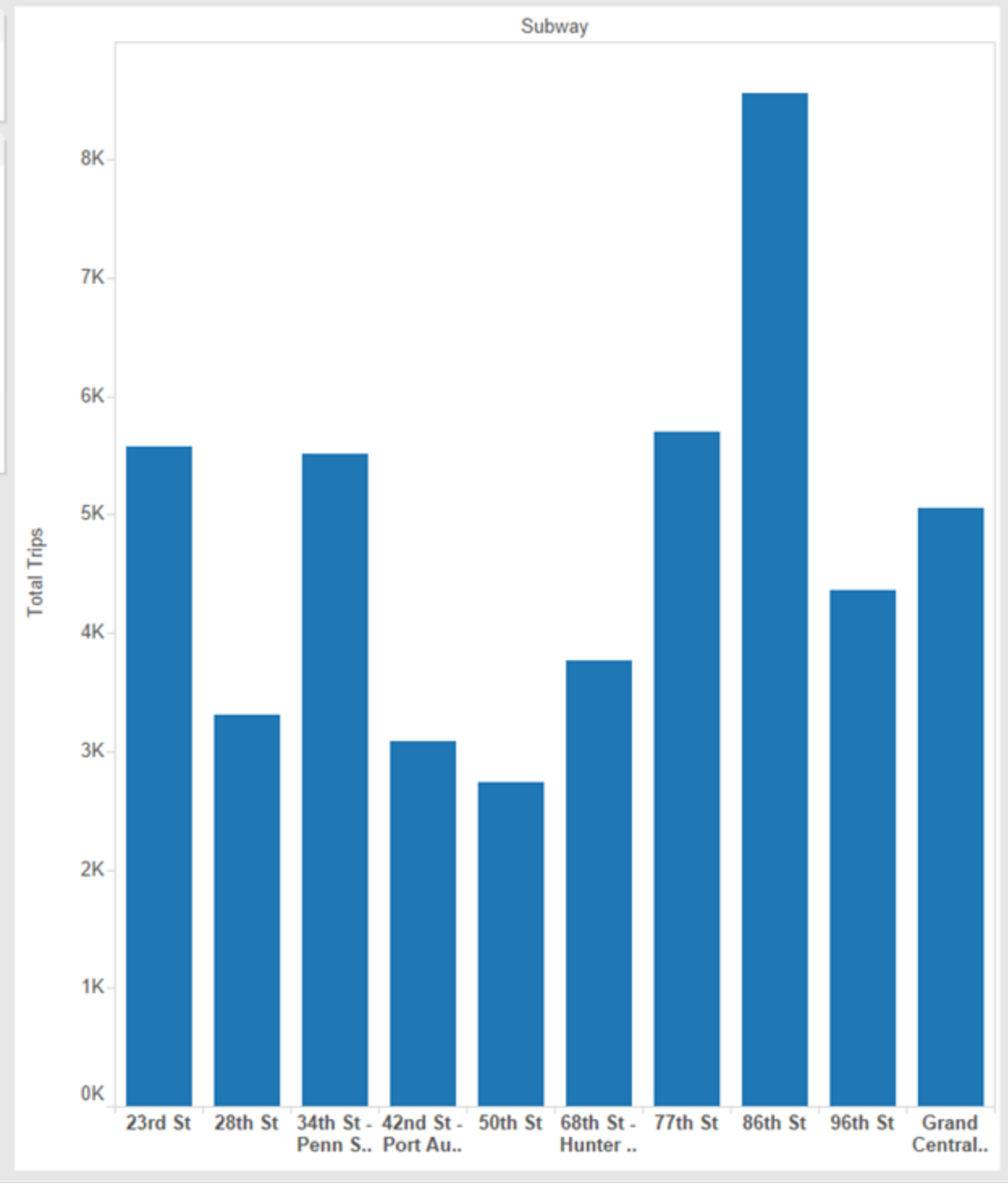
Rain Feb Top 10 Subway Stations :-



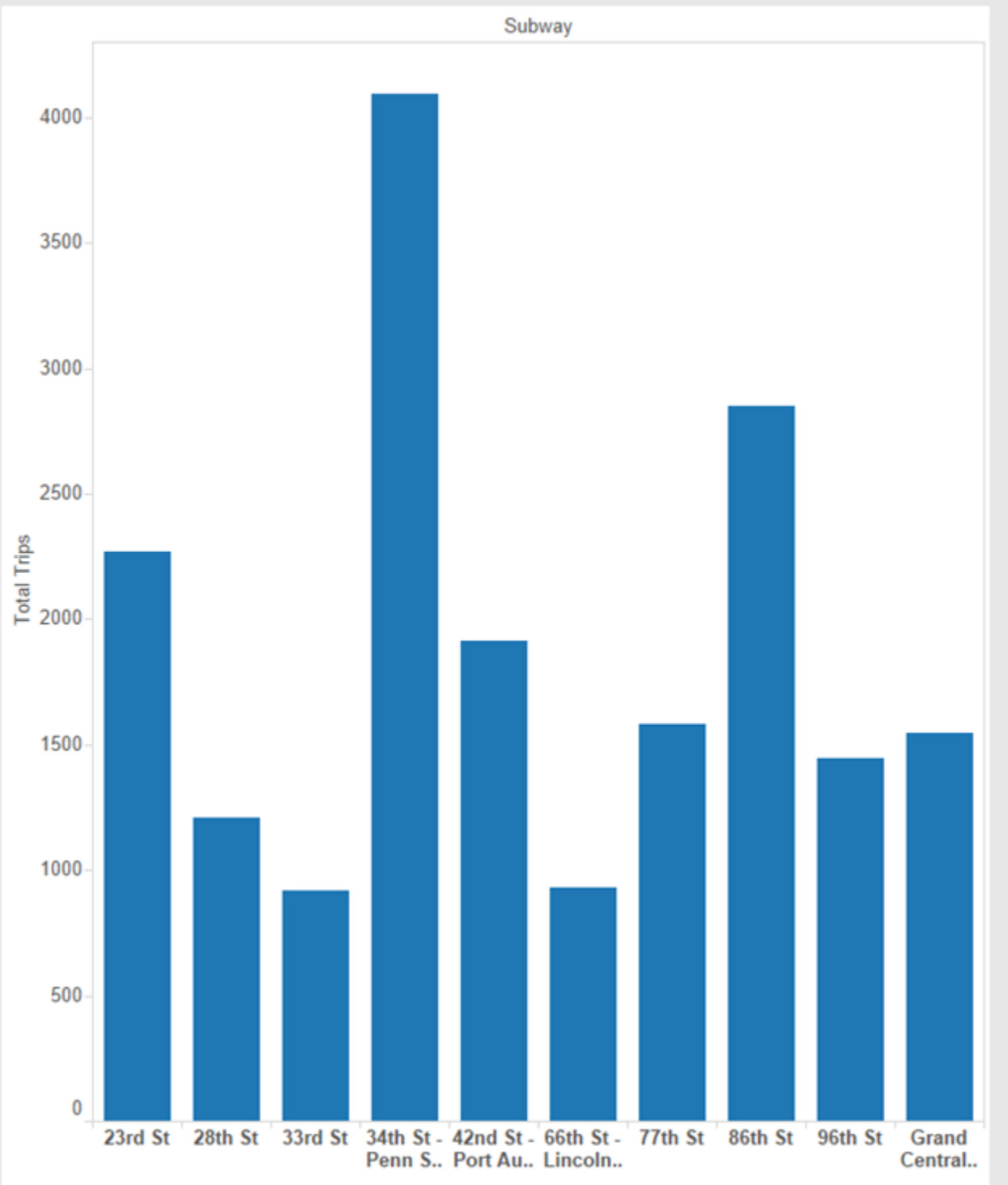
Rain March Top 10 Subway Stations :-



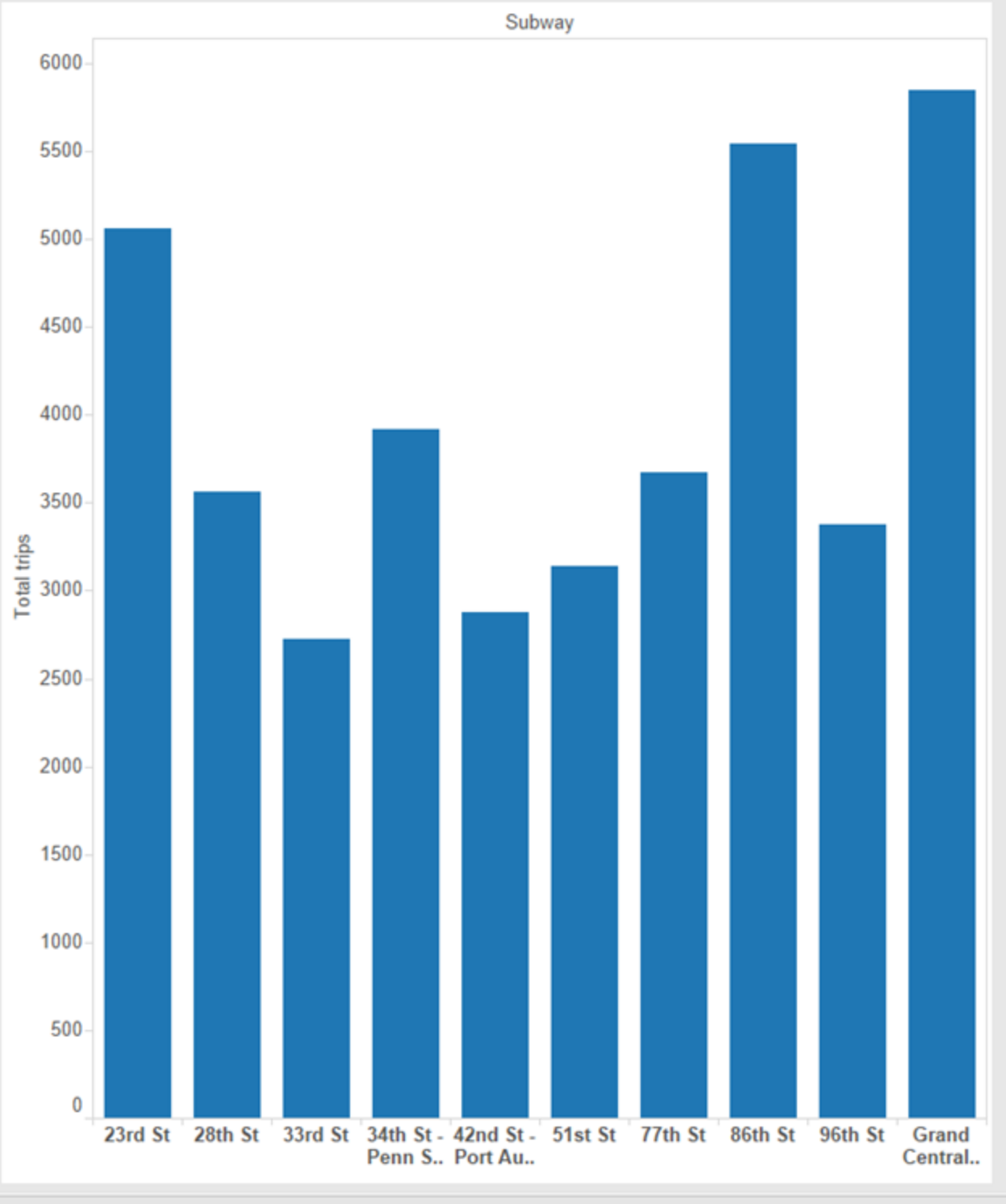
Snow Jan Top 10 Subway Stations :-



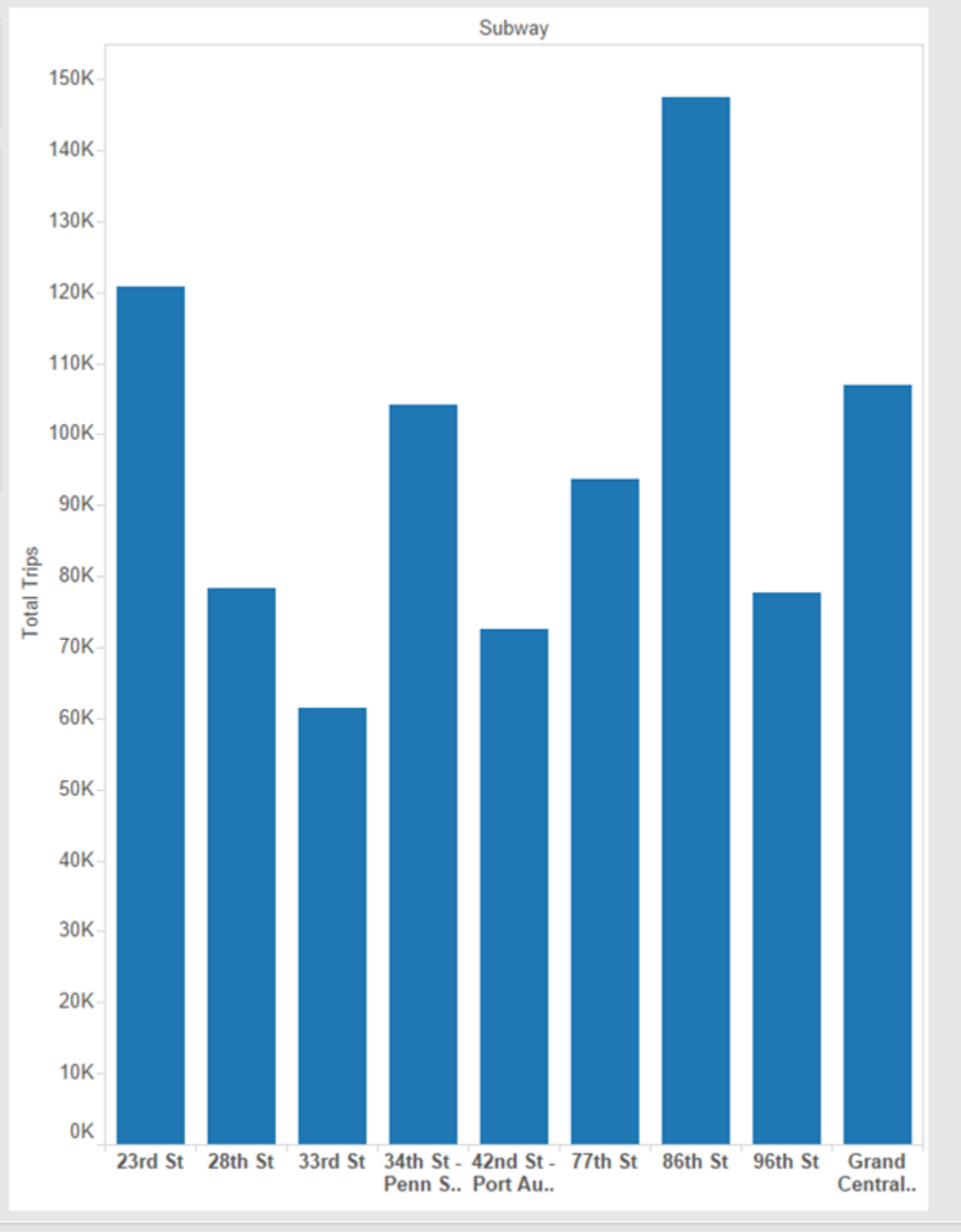
Snow Feb Top 10 Subway Stations :-



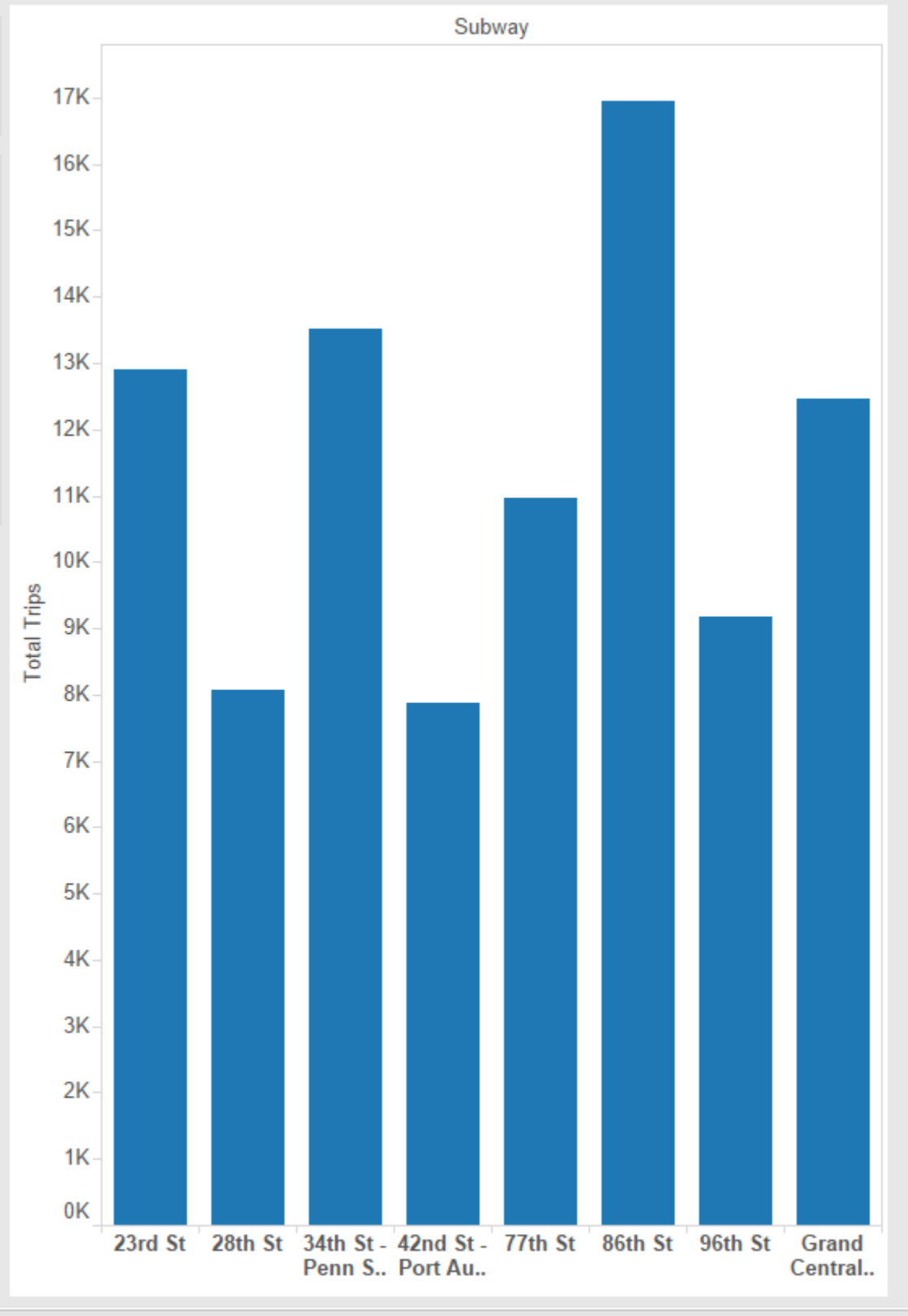
Snow March Top 10 Subway Stations :-



Rain Top 10 Subway Stations aggregated over 3 months :-



Snow Top 10 Subway stations aggregated over 3 months :-



Analysis for Question 1:-

According to our visualizations created we can conclude that during rain and snow the most popular subway station is 86th Street near where the passengers get off.
On the basis of the results we have got we feel that the NYC Taxis can increase the number of cabs in the areas where the Top 10 subway stations are located so that their evenue could incerease.

Question 2 :-

What is the trend regarding the number of people taking taxis at a particular time of the day versus the number of people traveling by subways during particular peak hours of the day?

Method/Algorithm for Question 2 :-

Datasets Used :- NYC Taxi Data, Subway Turnstile Data

1. Use the previously cleaned NYC taxi data
2. Run taxi_revenue.py on the NYC taxi data in order to calculate the per hour average revenue of the NYC taxi day per day per month
3. Calculate the total number of entries at all the subway stations per hour per month by running subway_revenue.py on Subway Turnstile Data
4. Analyse the output of the 2 and 3 step by visualizations in order to gain deeper insights

Instructions/Commands for Question 2 :-

First please note that we are using the cleaned NYC taxi data from the first question

Now, move the taxi_revenue.py from your bucket to hdfs
Hdfs dfs -get s3://yourbucket/taxi_revenue.py taxi_revenue.py
Hdfs dfs -put taxi_revenue.py taxi_revenue.py

Now run taxi_revenue.py on your cleaned NYC taxi data
Spark-submit taxi_revenue.py taxidataclean.csv >> averagetaxirevenue.csv

Now move the averagetaxirevenue.csv from hdfs to your bucket
Hdfs dfs -put averagetaxirevenue.csv averagetaxirevenue.csv

Perform visualizations on averagetaxirevenue.csv

Now, move the subway_revenue.py from your bucket to hdfs
Hdfs dfs -get s3://yourbucket/subway_revenue.py subway_revenue.py
Hdfs dfs -put subway_revenue.py subway_revenue.py

Now run subway_revenue.py on your Subway Turnstile data

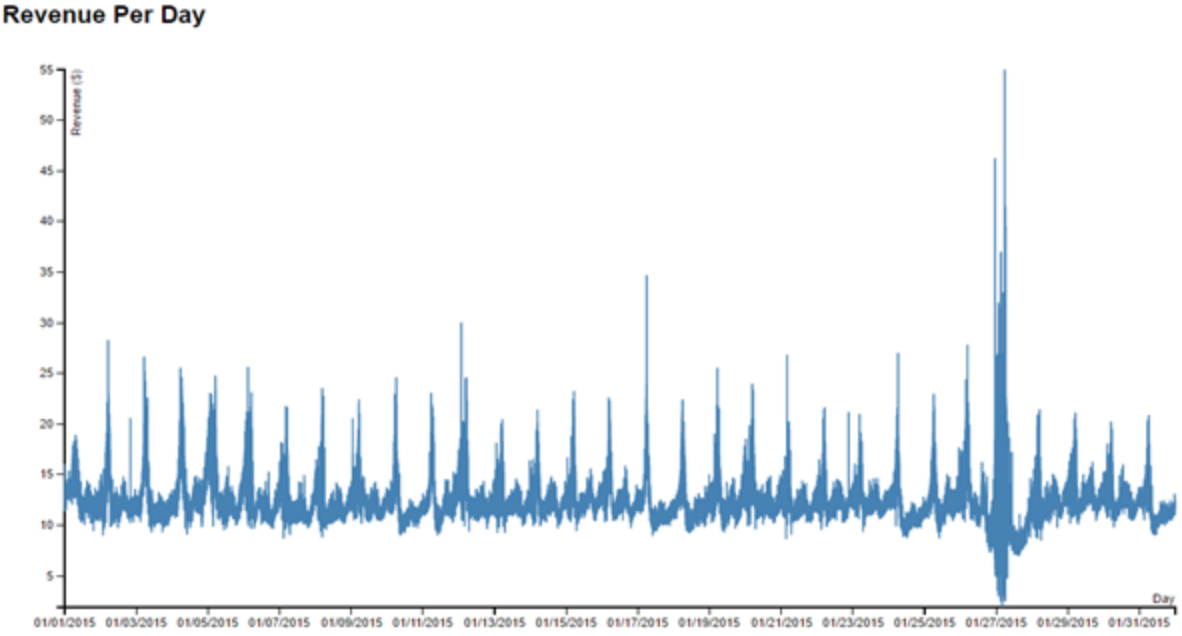
First move Subway Turnstile data from your bucket to hdfs
Hdfs dfs -get s3://yourbucket/subwayturnstiledata.csv subwayturnstiledata.csv
Hdfs dfs -put subwayturnstiledata.csv subwayturnstiledata.csv

Now, run subway_revenue.py on Subway Turnstile Data
Spark-submit subway_revenue.py subwayturnstiledata.csv >> subwayanalysis.csv

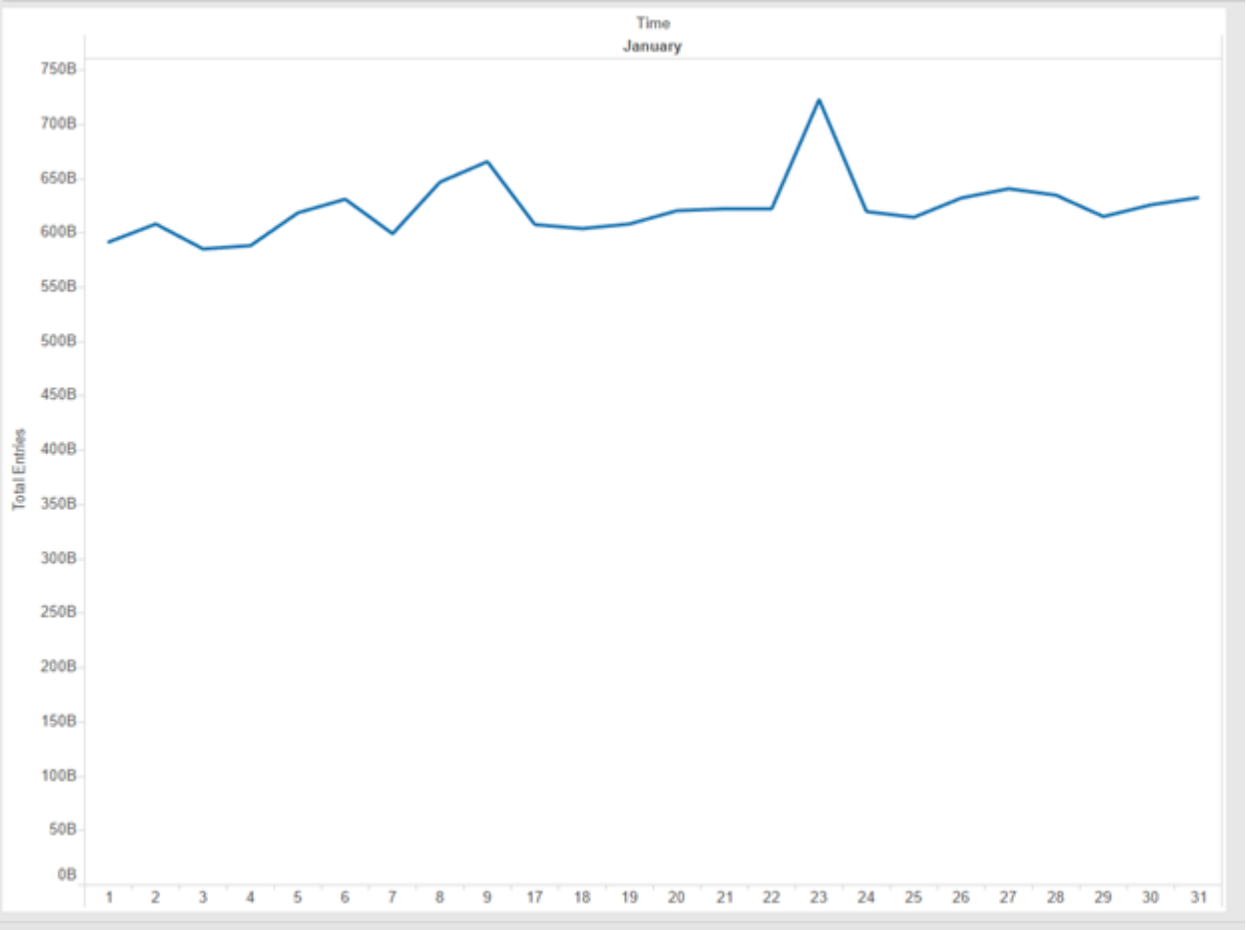
Now, move subwayanalysis.csv to your bucket and perform visualizations on it
Hdfs dfs -put subwayanalysis s3://yourbucket/subwayanalysis.csv

The visualizations for question 2 have been performed in Tableau :-
Following are the screenshots of our visualizations :
Subway Entries (No.of People) per month
Taxi trips per month per hour

Taxi Revenue for January 2015



Subway Revenue for January 2015



Analysis for Question 2 :-

Looking at the visualizations we can see that the number of trips drops between 4PM and 5PM and then rises sharply in the late evening hours as people return to their home from work or go out for a party. The number of trips decreases after 8PM till early morning and then increase around 8AM-9AM the normal work time for NY. The sharp decrease between 4PM and 5PM is something unusual as people leave work around 4-5PM this should be marked by an increase in the number of trips but it is exactly the opposite. approximately a 15% drop in the number of taxi rides between 4-5PM. on searching regarding this the probable reason for this is the shift change for the taxis. Most of the taxis are used by 2 drivers and each work for a 12 hour shift. The handover of taxicabs takes place at this time between 4-5PM. To meet the demand for taxi during this time there needs to be a change in the way this hand over takes place. The change in the shift handover is a decision that involves other factors and parameters like the profit made by each driver and their availability. The would be a tough decision to take for the T&LC management.

On plotting the revenue graph per day, we observed that for January 2015 there was a steep rises on January 27, 2015 as there was a Blizzard and people preferred taking cabs rather than

subways to their residences in order to evade the weather conditions. Subway revenue remain as normal. This is shown below

References :-
http://www.nytimes.com/2011/01/12/nyregion/12taxi.html?_r=0
http://www.cnn.com/2015/01/27/us/new-day-5-things/

Running Times of Tasks :-

Question 1:-

Cleaning NYC Taxi Data :- on an average 4 minutes per month
Cleaning Weather Data :- Less than 1 minute
Taxi Data and Weather data join :- on an average 5 minutes per month
Joined data output cross product with subway coordinates data :- on an average 27 minutes per month
Subway stats on the cross join output :- on an average 2 minutes per month

Question 2:-

Generate per hour revenue per month :- on an average of 3 minutes per month
Generate subway revenue per month :- on an average of 1 minute per month

Cluster Configuration :-

AWS - EMR - Select spark while creating cluster

Optimizations Performed :-

1. We have used the broadcast functionality of spark in order to broadcast subway.csv to all the worker nodes, before this it was taking 3 hours of time to perform the cross product which was brought down significantly to 26 minutes. Master node will distribute the subway.csv file to all the worker nodes. Each node will open the subway.csv file and create the dictionary, since the subway.csv file is 24KB all the calculations could be performed inside the memory which improved the time drastically.

2. To measure the distance between the taxi drop off points (latitude and longitude) and subway station coordinates (latitude and longitude) we use the Vincenty's formulae

References :-

https://en.wikipedia.org/wiki/Vincenty%27s_formulae

Individual Contributions :-

1. Identifying Questions :- Shyam, Jil and Vivek.
2. Cleaning Data Scripts - Taxi :- Vivek and Jil.
3. Cleaning Data Scripts - Weather :- Vivek and Shyam.
4. Understanding Metadata for datasets :- Vivek, Shyam and Jil.

5. Question 1 implementation :- Shyam, Vivek and Jil.
6. Question 1 D3 Visualizations :- Shyam.
7. Question 1 Analysis :- Shyam, Jil and Vivek.
8. Question 2 implementation :- Vivek, Shyam and Jil.
9. Question 2 Tableau Visualizations :- Vivek, Shyam and Jil.
10. Question 2 Analysis :- Shyam and Vivek.
11. Weekly Project Documentation Update :- Jil.
12. Github Repo Creation :- Shyam.
13. Final Project Documentation :- Vivek, Shyam and Jil.
14. Question 1 Optimizations Idea :- Vivek.