A
Project Report
On

"NYC Automated Traffic Volume Counts"

Prepared by

Saurabh Jain | saurabh_jain@csu.fullerton.edu

Pratishtha Soni| pratishthasoni@csu.fullerton.edu

A Report Submitted to

Tseng-Ching James Shen, PhD

Department of Computer Science

College of Engineering and Computer Science (ECS)

California State University, Fullerton (CSUF)

for the Fulfillment of the Requirements for the

CPSC 531-01 Advance Database Management

California State University, Fullerton
CA - 92831
December, 2022

TABLE OF CONTENTS

INTRODUCTION

Problem Statement

The vehicular traffic is increasing tremendously these days, simultaneously congestion also increases. To prevent congestion, one option is to increase the capacity by increasing the number of the existing transportation system. A second option is to develop alternatives that increase capacity by improving the efficiency of the existing transportation system. The latter focuses on building fewer lane miles.

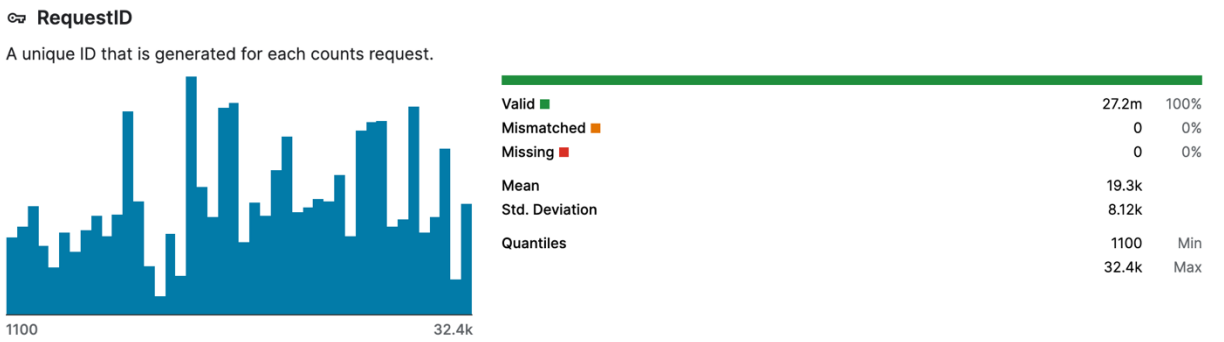
To Overcome these Problems, the New York City Department of Transportation (NYC DOT) uses Automated Traffic Recorders (ATR) to collect traffic sample volume counts at bridge crossings and roadways. These counts do not cover the entire year, and the number of days counted per location may vary from year to year.

Data Description

NYC Automated Traffic Count allows the user to view and access traffic data information of different cities in NYC. The information is displayed in the form of excel. New York City transportation uses automated traffic recorders to collect traffic volume counts at bridge crossings and roadways. The data consists of car volume, averaged every 15 mins for the duration of 2000 to 2020. This traffic data can be used to determine various type of analysis and design patterns in traffic over the year 2000 to 2020.

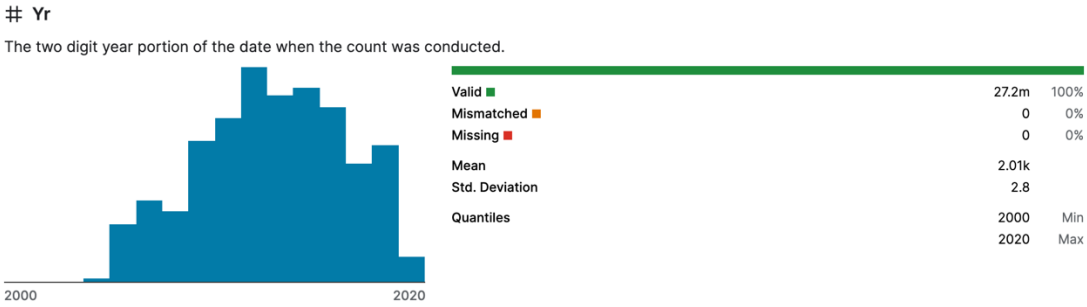
The information is collected from Kaggle in a form of 1 3GB excel file and consists of 14 rows and 27190511 columns:

RequestID - Unique ID that is generated for each count request.

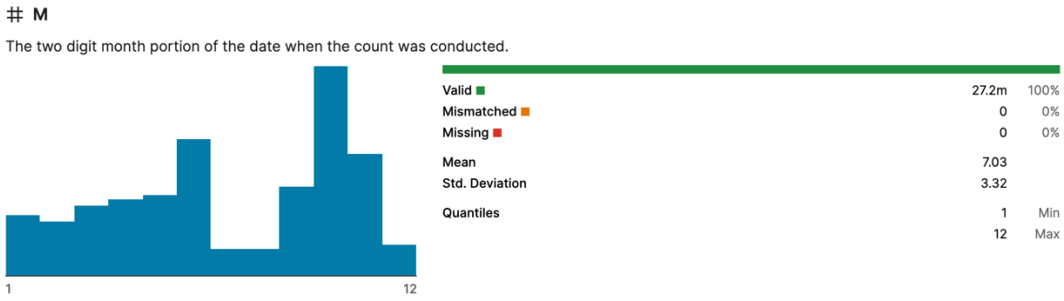


Boro - Lists which of the five administrative divisions of New York City the location is within, written as a word. Where Brooklyn data is 29% and Queens data is 26%.

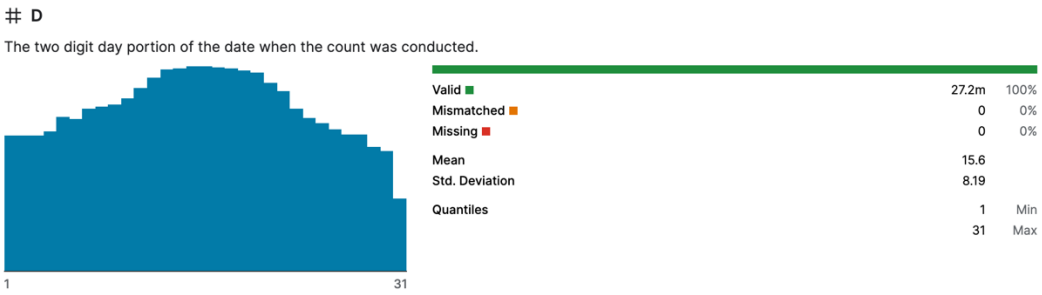
Yr - The two-digit year portion of the date when the count was conducted. (From 2000 to 2020).



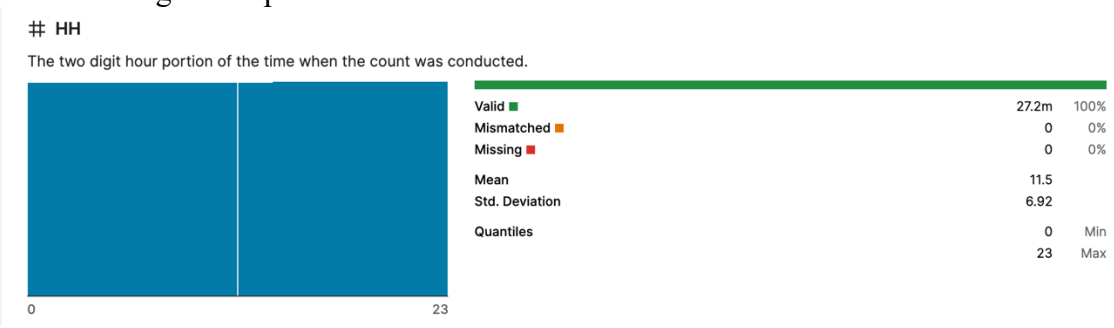
M - The two-digit month portion of the date when the count was conducted.



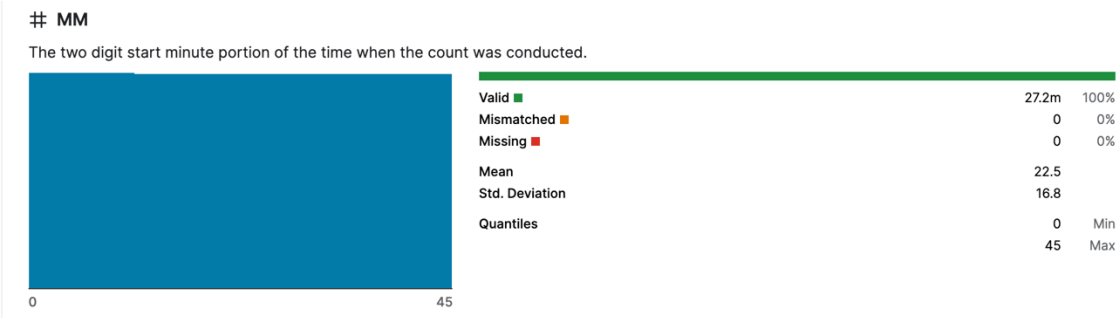
D - The two-digit day portion of the date when the count was conducted.



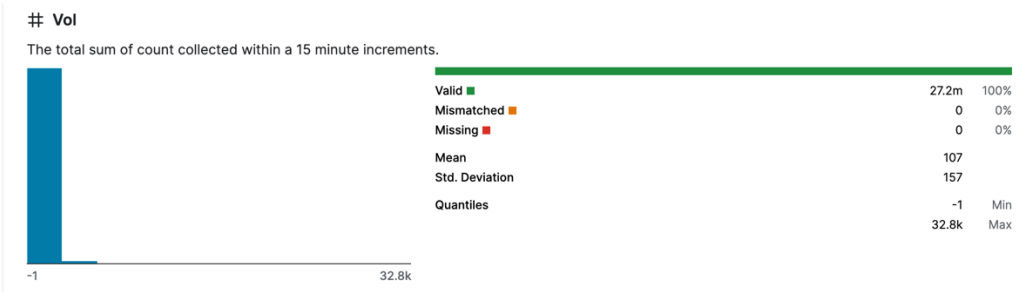
HH - The two-digit hour portion of the time when the count was conducted.



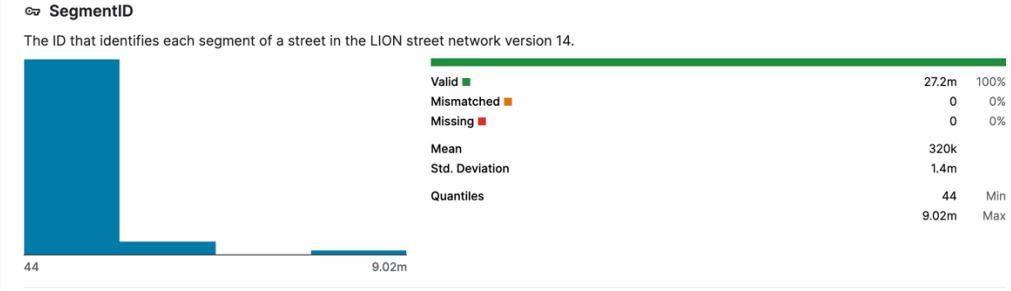
MM - The two digits start minute portion of the time when the count was conducted.



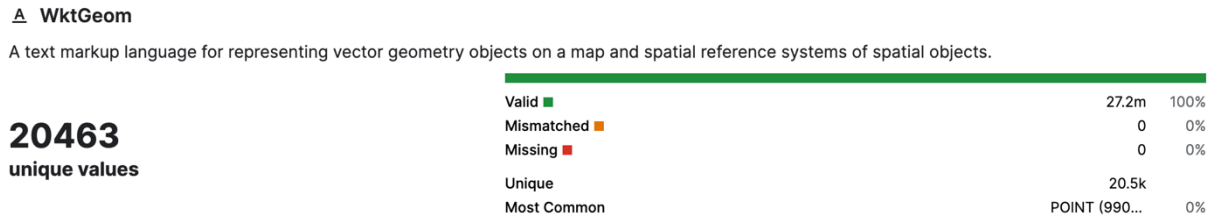
Vol - The total sum of counts collected within 15-minute increments.



SegmentID - The ID that identifies each segment of a street in the LION street network version 14.



WktGeom - A text markup language for representing vector geometry objects on a map and spatial reference systems of spatial objects.



Street - The 'On Street' where the count took place.

fromSt - The 'From Street' where the count took place.

toSt - The 'To Street' where the count took place.

Direction - The text-based direction of traffic where the count took place.

Tool and Technology Used:

- Spark
- Google Collab Notebook
- Google Cloud Platform
- Google Data Proc
- Python Matplotlib

FUNCTIONALITIES

1. Pre-Processing of Dataset:

In this step all the null values from different columns from the dataset is removed using the python language in juypter notebook. Particularly, pandas library is being used for converting the datatypes (from string to integer and vice versa) and for removing null values and for removing the spaces before, after and in between of each string/int variable.

2. Processing of the Dataset on the Cloud:

The project data is in the form of excel of size ***3 GB*** and processing this huge data on local is challenging, so processed this bunch of data using Google's cloud dataproc service which is the open-source service for running Apache Hadoop and Apache spark and other tools and frameworks. The dataproc allows a new user 300 free credits to process and play with the data.

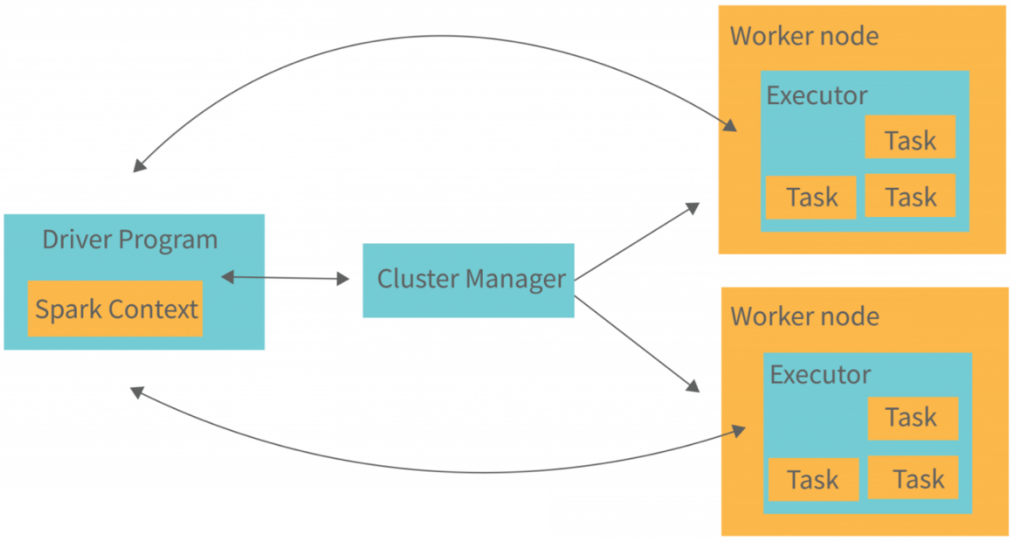
3. Data Analysis/Queries Significance:

- Enhance public safety.
- Reduce congestion.
- Improved access to travel and transit information.
- Generate cost savings to motor carriers, transit operators, toll authorities, and government agencies; and
- Reduce detrimental environmental impacts.

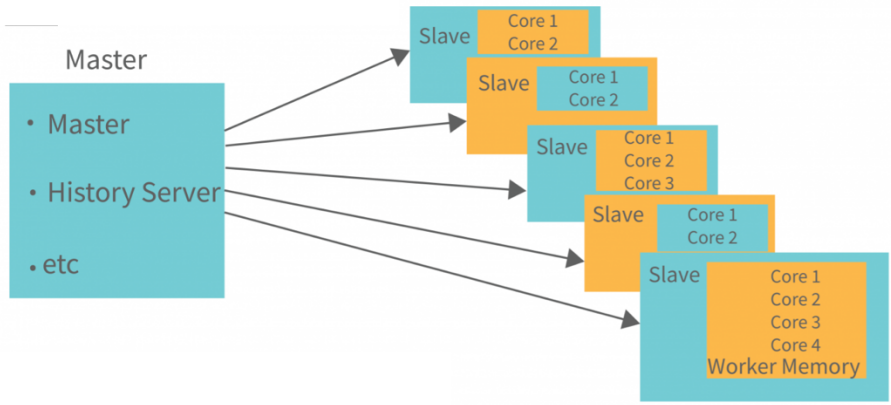
ARCHITECTURE & DESGIN

A high-level view of the Apache Spark application:

In the below diagram, spark uses the master and slave architecture. The driver program connects to a spark cluster, calls the main application, and builds a spark context (which serves as a gateway) to monitor the jobs running in the cluster. The spark context is implemented for all the operations.



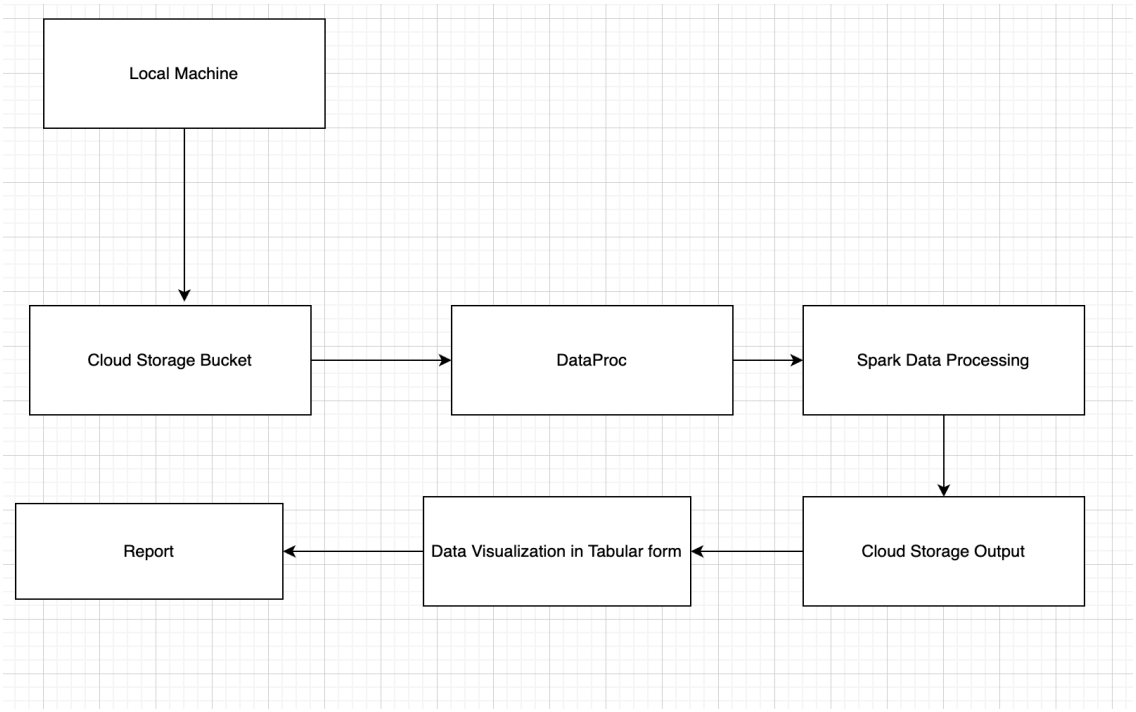
Spark Standalone Architecture



The architecture/implementation diagram of the project:

Methodology:

- **Local Machine:** The computer that you are currently logged on to as a user.
- **Cloud Storage Bucket:** Google Cloud Platform Console to upload files or folders.
- **DataProc:** Dataproc is a managed Spark and Hadoop service that lets you take advantage of open-source data tools for batch processing, querying, streaming, and machine learning.
- **Spark Data Processing:** Apache Spark is an open-source, distributed processing system used for big data workloads. It utilizes in-memory caching and optimized query execution for fast queries against data of any size. Simply put, Spark is a fast and general engine for large-scale data processing.
- **Cloud Storage Output:** Google Cloud Platform Console to upload files or folders.
- **Data Visualization:** Visualization of data using Python Library.



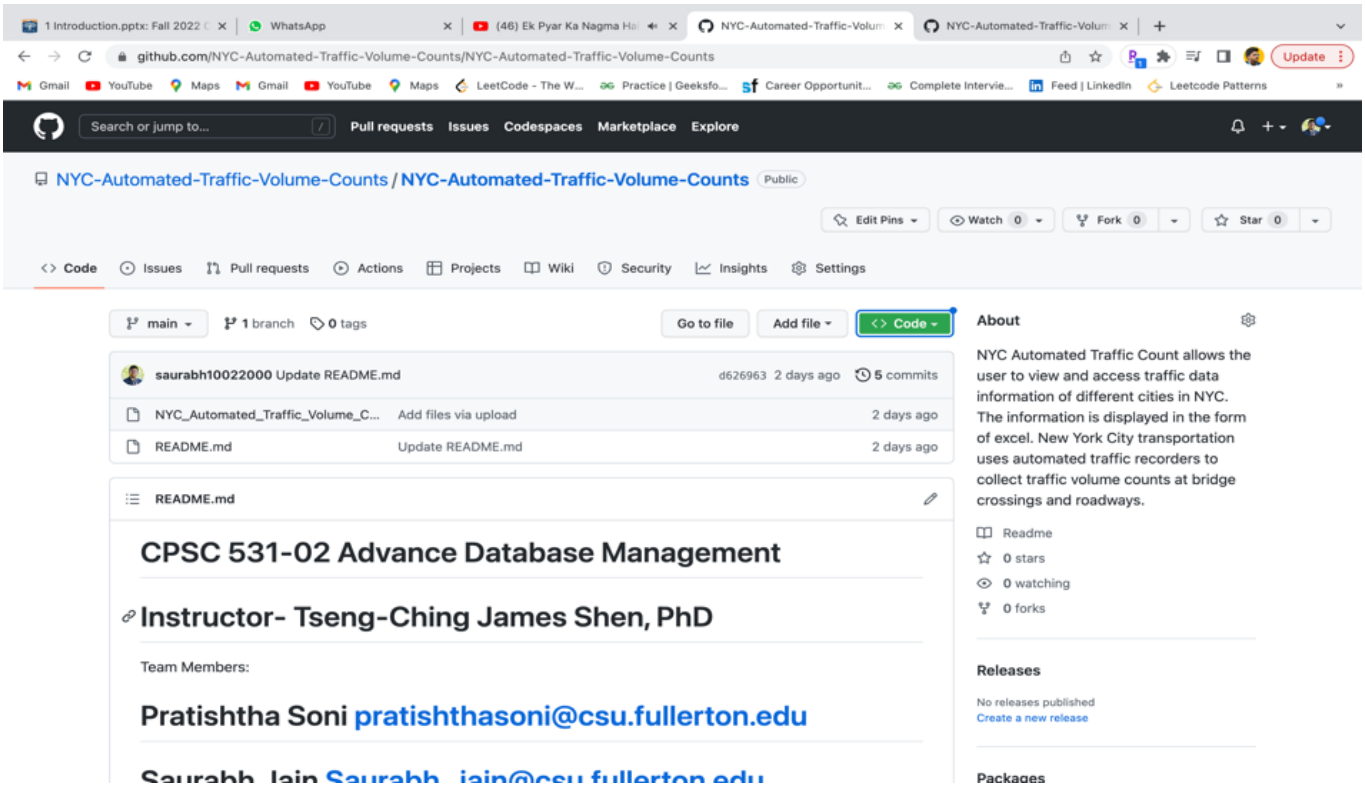
GitHub LOCATION OF CODE

Project Location: https://github.com/NYC-Automated-Traffic-Volume-Counts/NYC-Automated-Traffic-Volume-Counts.git.

Steps:

1. Download "NYC_Automated_Traffic_Volume_Counts.ipynb".
2. Download Data from Kaggle (3GB) https://www.kaggle.com/datasets/aadimator/nyc-automated-traffic-volume-counts
3. Import Data and project files to DataProc or on a Local Machine.

4. Run



DEPLOYMENT INSTRUCTIONS

Steps to Set-up Spark in Local Machine:

Install JDK

```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
```

Get Spark installer (Check the path on Spark.org)

```
!wget -q https://archive.apache.org/dist/spark/spark-2.4.3/spark-2.4.3-bin-hadoop2.6.tgz
```

Check if the file is copied

```
!ls
```

Untar the Spark installer

```
!tar -xvf spark-2.4.3-bin-hadoop2.6.tgz
```

Check the spark folder after untar

```
!ls
```

Install findspark - a Python library to find Spark

```
!pip install -q findspark
```

Set environment variables
Set Java and Spark Home based on location where they are stored

```
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-2.4.3-bin-hadoop2.6"
```

Create a local Spark Session

```
import findspark
findspark.init()
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[*]").getOrCreate()
```

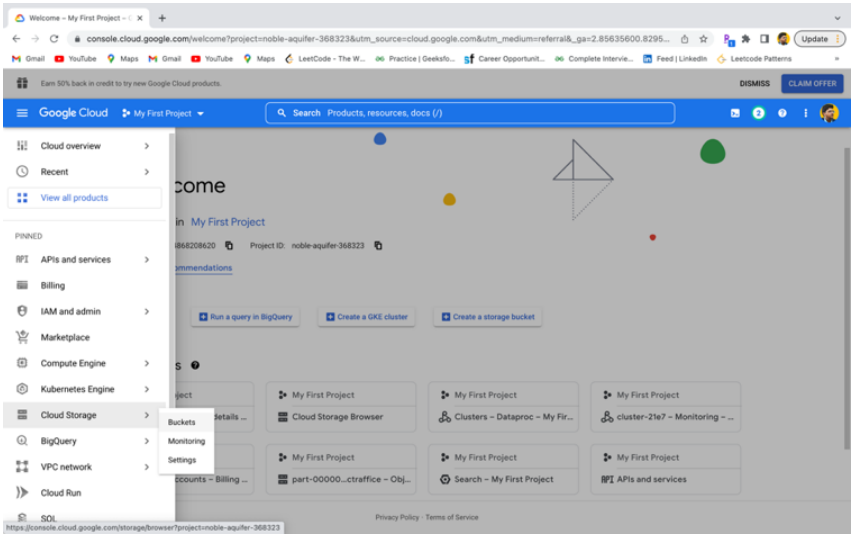
Test Installation

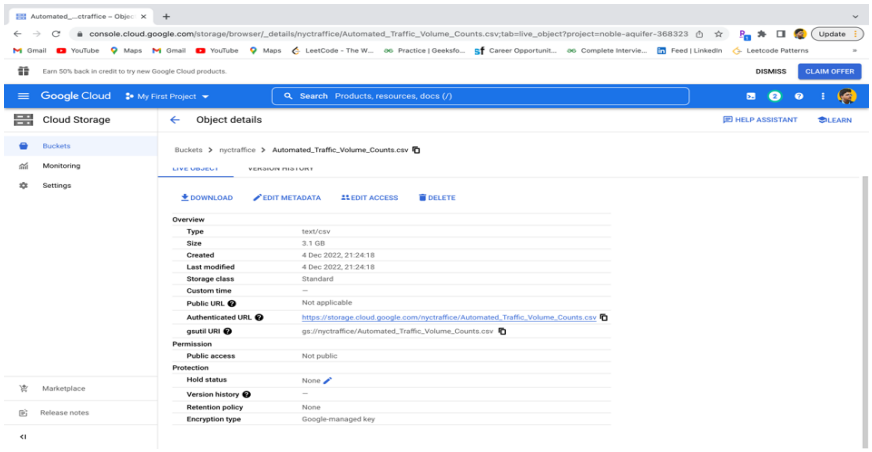
```
df = spark.createDataFrame([{"Google": "Colab","Spark": "Scala"} ,{"Google":
"Dataproc","Spark":"Python"}])
df.show()
```

STEPS TO RUN THE APPLICATION

1. Create a Bucket and Upload Data.

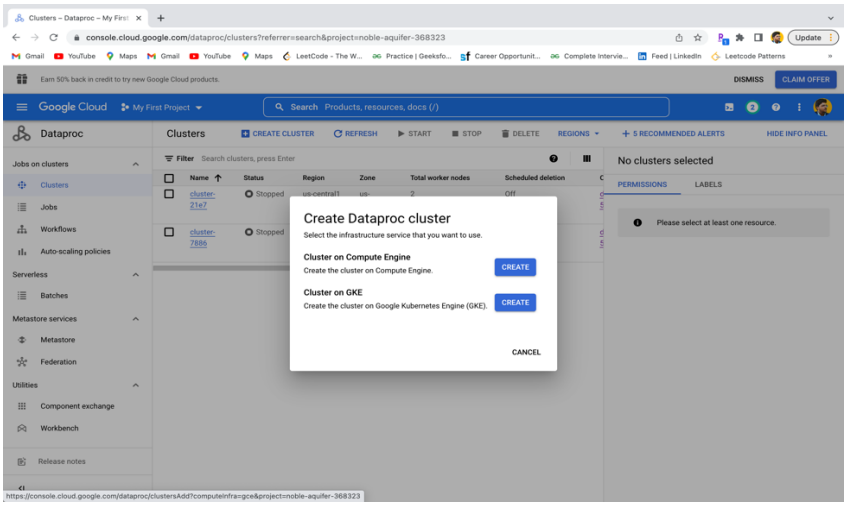
- In the Google Cloud console, go to the Cloud Storage Buckets page. Go to Buckets.
- Click Create bucket.
- On the Create a bucket page, enter your bucket information. To go to the next step, click Continue. For Name, your bucket, enter a name that meets the bucket name requirements.
- Click Create.
- Upload data and Use the Access link to import data from Bucket to SSH Console.

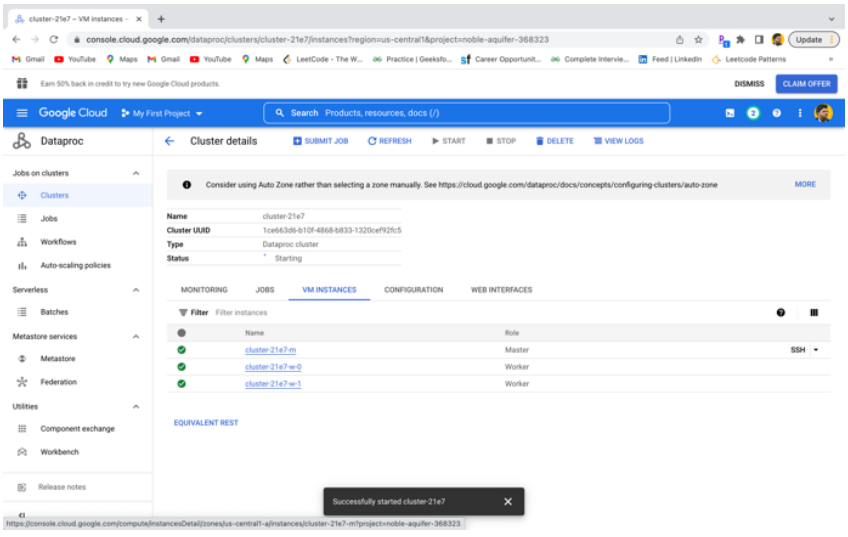




2. Open DataProc Services and create clusters.

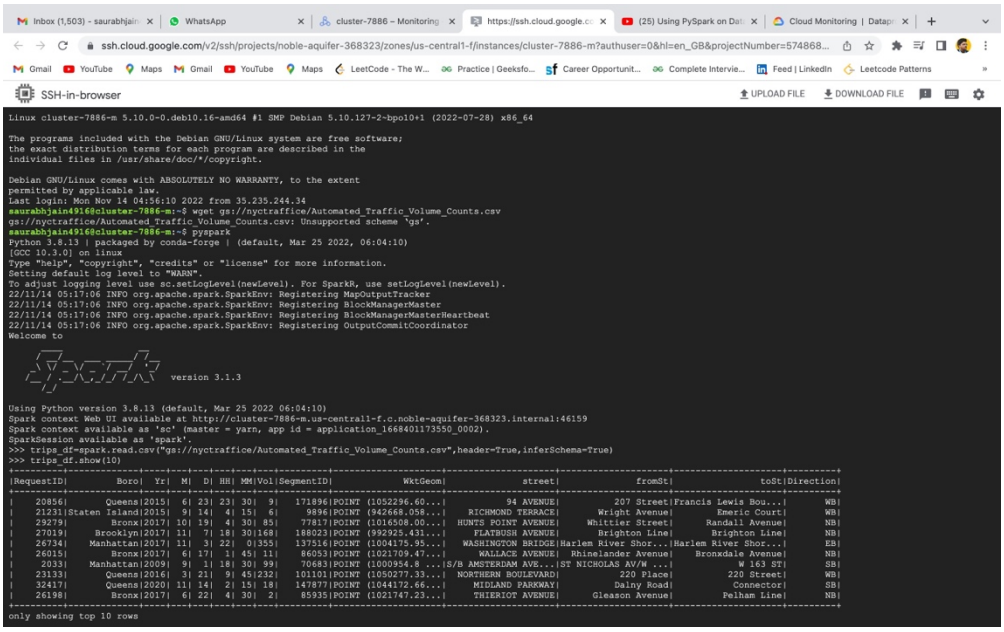
- In the Google Cloud console, go to the Dataproc Clusters page.GO TO CLUSTERS
- Click Create cluster.
- In the Create Dataproc cluster dialog, click Create in the Cluster on Compute engine row.
- In the Cluster Name field, enter example-cluster.
- In the Region and Zone lists, select a region and zone
- For all the other options, use the default settings.
- To create the cluster, click Create.Your new cluster appears in a list on the Clusters page. The status is Provisioning until the cluster is ready to use, and then the status changes to Running. Provisioning the cluster might take a couple of minutes.

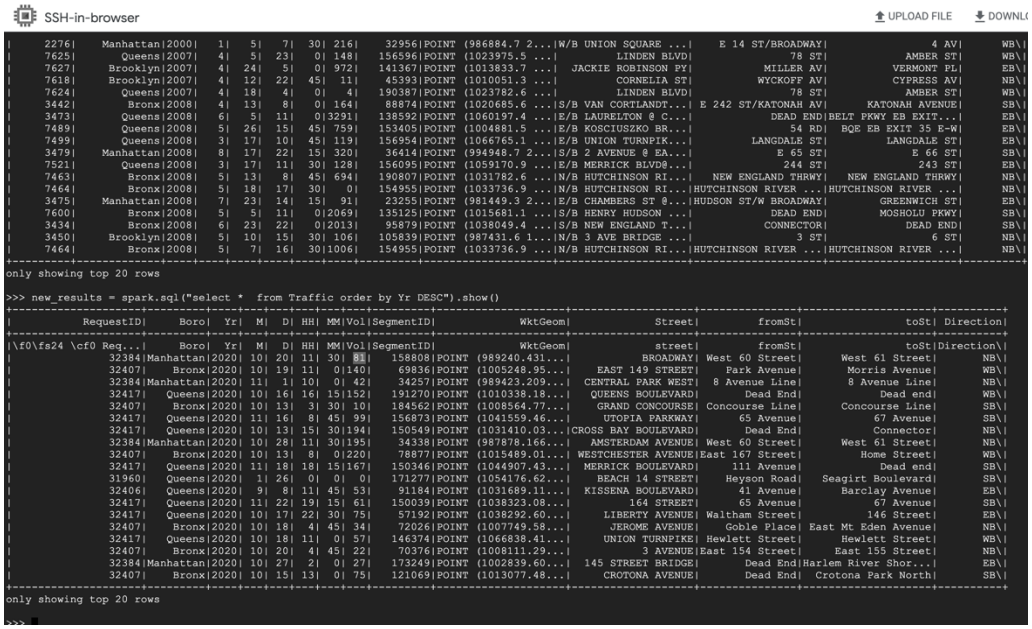




3. SSH Terminal / Jupyter Notebook to Execution

- Start pyspark.
- customerDF = spark.read.csv("write address of bucket",header=True)
  customerDF.show()
- write spark sql queries.
- Use this command to save output in bucket
  db.write.save("write address of bucket ",format='csv',header=True)

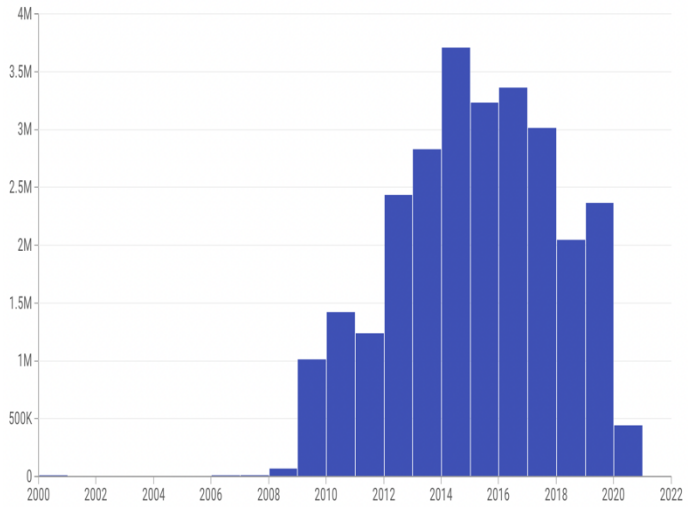




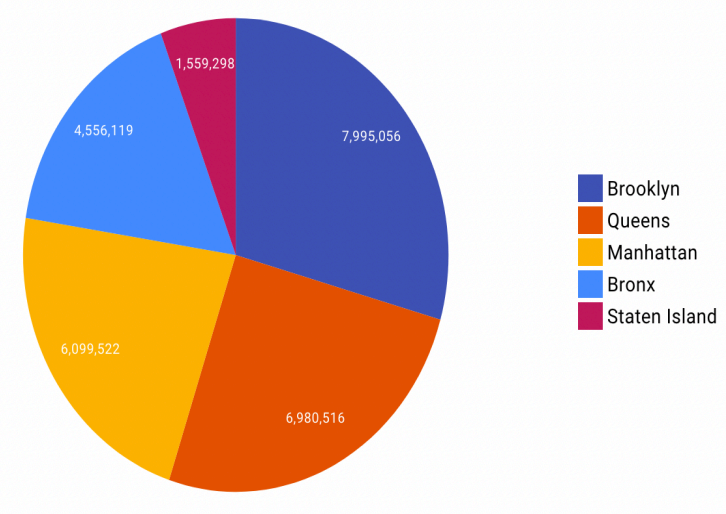
TEST RESULT

Data visualization Using NYC-Automated-Traffic-Volume-Counts

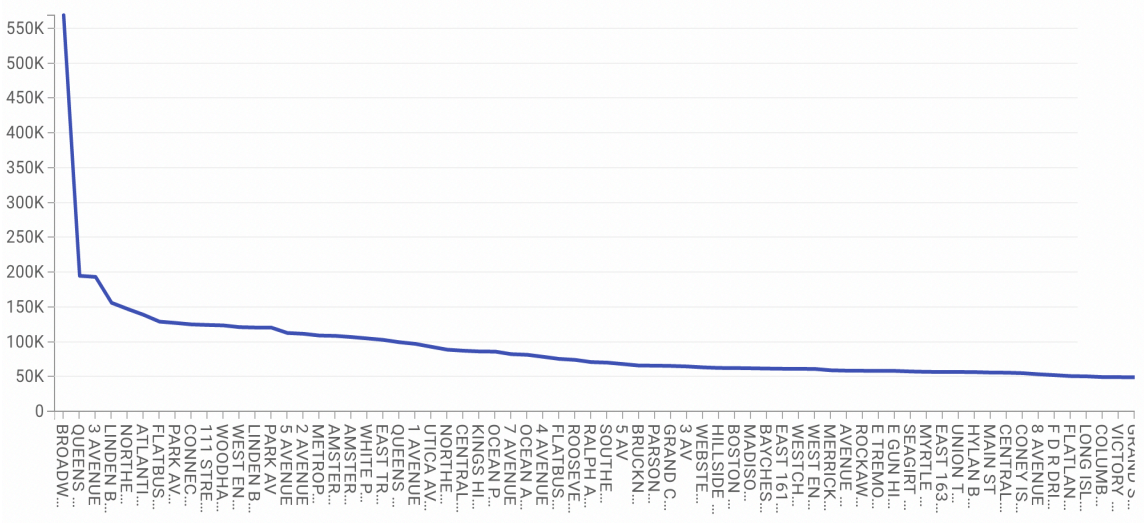
- Variation in Volume of data/vehicle: After pre-processing, the main analysis of this traffic dataset shows the number of increases in the volume of cars from the year 2000 to 2020. (Queries being displayed in the implementation section)



- Busiest Cities of NYC: After pre-processing and running queries and we extracted the top 3 busiest cities and, it found that Brooklyn is the most crowded and busiest city with the count of vehicles (7,995,056, 29%), Queens being the 2nd busiest (6,980,516, 26%) and Manhattan being the 3rd busiest (6,099,052, 22%).



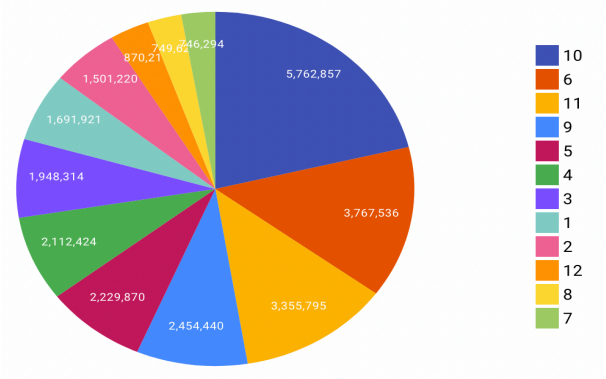
- Most visited streets: From the dataset, we found the most visited and busiest street in different cities in NYC.



- The volume of car increased every 15 mins in New York: This query displays how many numbers of cars increased city of Boro.

- Grouped according to Busiest Month in the year of different Cities: By applying group by, sorting, and count query we found that October was the busiest month in Manhattan in the entire year.

  - 1st pie chart shows the number distribution according to month.



  - 2nd pie chart shows the busiest city according to the month.