**NYC Trip Data Analysis**

A group of yellow taxi cabs on a wet street

Description automatically generated

**By:**

Divisha Choudavaram, Graduate Student at Northeastern University

Email: [choudavaramdivisha@gmail.com](mailto:choudavaramdivisha@gmail.com)

LinkedIn: <https://www.linkedin.com/in/divisha-choudavaram/>

# **Introduction**

In this project I created a NYC Trip Analysis Dashboard using the trip records data provided by the Taxi and Limousine Commission ([here](https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page)) from the years 2016 and beyond, amounting to a total of 615 million trip records. This dashboard offers an in-depth analysis of taxi trips in New York City, unveiling key insights that encompass the busiest pickup and drop-off locations, and peak trip times throughout the day. It provides valuable insights into the traffic dynamics of NYC. Moreover, the dashboard conducts a thorough analysis of cab services, encompassing yellow taxis, Green taxis, and ride-sharing platforms such as Uber, Lyft, Juno, and Via. These insights are particularly useful for those looking to venture into the New York City taxi industry.

# **Dataset**

As mentioned above the data is from the New York City Taxi and Limousine Commission (TLC) provides access to extensive trip data, which can be found [here](https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page). This dataset contains valuable information, including pickup times, geographic coordinates, passenger counts, and various other essential variables.

The dataset comprises monthly Parquet files for each year, with separate files for yellow taxis, green taxis, and for-hire vehicles like Uber and Lyft. For this project, we have considered data from the year 2016 through 2023. The size of the data is 615 million trip records and 33 fields for yellow taxis, 47 million trip records and 34 fields for green taxis, and 20 million records with 35 fields for For-Hire vehicle records. For more detailed information about the dataset, the data descriptions for yellow taxi records can be found [here](https://www.nyc.gov/assets/tlc/downloads/pdf/data_dictionary_trip_records_yellow.pdf), green taxi records [here](https://www.nyc.gov/assets/tlc/downloads/pdf/data_dictionary_trip_records_green.pdf), and for-hire vehicle records [here](https://www.nyc.gov/assets/tlc/downloads/pdf/data_dictionary_trip_records_hvfhs.pdf).

By leveraging this rich dataset, we can gain deep insights into the intricacies of the New York City transportation ecosystem, enabling us to make informed decisions and tailor our services to the specific needs and demands of this dynamic urban environment.

# **Methodology**

Due to the large volume of data involved, we have developed a Big Data pipeline to extract and prepare the data for visualization. Our Big Data pipeline leverages a variety of technologies, including Apache Spark, Databricks, Google BigQuery, and Looker for the Visualization Dashboard. Please find the flowchart below depicting the Big Data Pipeline.

A diagram of data processing

Description automatically generated

**Figure1: Big Data Pipeline Flow Chart Diagram**

Before proceeding with the steps, the dataset should be downloaded which can be found here - <https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page>. If you scroll down, you will find the data organized by years, as shown below.

A screenshot of a computer

Description automatically generated

**Figure 2: Dataset**

In this, each year has files for yellow taxis, green taxis, and for-hire vehicles for different months, all in the following format: 'yellow\_tripdata\_2023-03.parquet,' 'green\_tripdata\_2013-03.parquet,' and 'fhvhv\_tripdata\_2023-03.parquet,' respectively. The '03' in the file names represents the month, which is March. We downloaded all these files manually and uploaded them to Google Cloud Storage, a process discussed below.

Anyone interested in running the pipeline can access these files by going to each year and downloading the files in the specified format. Specifically, you can download files from 'yellow\_tripdata\_2023-03.parquet,' 'green\_tripdata\_2013-03.parquet,' and 'fhvhv\_tripdata\_2023-03.parquet' for the years 2016 and above, totaling 270 files. However, if you just want to see how the pipeline works and don’t want to download all the 270 files, you can simply download the files as per the mentioned format from 2022 which will be around 36 into local machine and perform this pipelining.

## **Step 1:**

In the initial step, we have successfully transferred data from the NYC Taxi & Limousine Commission to a Google Cloud Storage Bucket. First one needs to have to Google Cloud console account which can be found here - <https://console.cloud.google.com/>. One can login to this using google account and create a new project by "Select a Project" on the top as shown below.

A screenshot of a computer

Description automatically generated

**Figure 3: Creating Google Project**

Then click on “New Project” and the below page open.

A screenshot of a computer

Description automatically generated

**Figure 4: Giving Project Details**

Give the project name and in my case it is “NycTaxiDataAnalysis” and click on create. Once the new project is created, we will use this for the entire pipeline.

To create a Google Cloud Storage Bucket, access the Google Cloud Console, select the project created – and then search for 'Cloud Storage,' and click 'Create' in the upper menu, as illustrated below.

A screenshot of a computer

Description automatically generated

**Figure 2: Creating a Google Cloud Storage Bucket**

The Google Storage Bucket appears as shown below, allowing us to upload files into the bucket by clicking on the 'Upload' option. Here you can use the upload option to upload the files downloaded into local machine.

A screenshot of a computer

Description automatically generated

**Figure 3: Uploading files into Google Storage Bucket**

I made this data public as this data is already available on internet and connection to databricks would be easy through public access. So, for each file file uploaded, the public access is given. For this we can click on the 3 dots beside the file and give public access -> edit access -> and give the public access as shown in the screenshot below.

A screenshot of a computer

Description automatically generated

**Figure 4: Public access to the files in Storage Bucket**

Once the files are uploaded the next step is to pre-process the data for visualization.

## **Step 2:**

We pre-processed the data using Apache Spark, a unified analytics engine for large-scale data processing capable of handling both batch and real-time streaming data workloads. Apache Spark is renowned for its speed, scalability, and fault tolerance. Our Apache Spark jobs are executed within the Databricks environment, a unified analytics platform that seamlessly integrates Apache Spark, Databricks Runtime, and a cloud-based workspace, offering users a single platform for data engineering, data science, and machine learning. The cloud provider used for Databricks is the Google Cloud Console.

One can open the Databricks website(<https://www.databricks.com/> ) -> Click on Get started for free -> Create the databricks account -> Give the cloud provider as “Google Cloud Platform” . Once you do this process, one can see a page where it asks for “Take me to Databricks on Google Cloud”. Click on that and you will see the below page.

A screenshot of a computer

Description automatically generated

**Figure 5: Creating Databricks Account**

Here Databricks platform should be activated as I already activated it doesn’t show me to activate, but a new user will have an option to activate and by click on it the service will be activated. Once activate, you can click on “Manage the Provider” which takes you to Databricks platform and asks for signin with google, Give the google account details over there which will open the databricks platform.

Then click on the “workspace” on the top and the page will be opened where you have to click on “create workspace” as shown below.

A screenshot of a computer

Description automatically generated

**Figure 6: Creating workspace on databricks**

The work space asks for the google project id for which you can open google cloud concole -> select the project and you will be able to find id beside as shown below.

A screenshot of a computer

Description automatically generated

**Figure 7: Selecting project ID**

Copy the project ID and paste it here and give the workspace details and choose any nearest area and click on “save”. If you don’t see the new workspace. Please refresh the page as it takes time and wait till it shows as status as “Running”. Once status changes, open workspace -> it again asks for credentials -> login with google account. Ths takes you to your workspace.

To connect Google Cloud Storage to Databricks, we first need to create a service account on the Google Cloud Console and provide the service account details in Databricks. You can create the service account by selecting 'Service account' on the Google Cloud Console, as shown below, and then clicking 'Create Service Account'.

A screenshot of a computer

Description automatically generated

**Figure 8: Creating Service Account**

The service account name can be given in "Service account details" as shown below.

A screenshot of a service account

Description automatically generated

**Figure 9: Service account details**

Once the details are given, we have to give “Storage Admin” access in the "Grant this service account access to project" section as shown below.

A screenshot of a computer

Description automatically generated

**Figure 10: Giving Storage Admin Access**

While 'Storage Admin' access is sufficient for connecting to Databricks, we will also establish a connection between Databricks and Google BigQuery going forward. Therefore, the roles shown below, which are essential for writing data into BigQuery, have been granted at this stage. So the **below roles should also be given At this stage** as the service account cannot edited once cliked on “Done”.

A screenshot of a computer

Description automatically generated

**Figure 11: Roles required for connecting Databricks to Google Big Query**

After giving the necessary roles, we just have to click on “Done” and the service account will be created which looks like below as shown in Figure 8. The highlighted part in the figure 8 should be given in Databricks which we will discuss going forward.

A screenshot of a computer

Description automatically generated

**Figure 12: Service account**

As workspace is created above, now we have to create a new ‘Notebook’ in that where we perform our data-preprocessing as shown below.

A screenshot of a computer

Description automatically generated

**Figure 13: Creating new ‘Notebook’ on Databricks**

After creating the notebook, to execute any operations within it, we must first attach a compute resource, as illustrated below. This computer resource window will automatically popup whenever a you run a query, so we can attach this while running the queries. Here, we can select the runtime environment as Spark.

A screenshot of a computer error

Description automatically generated

**Figure 14: Attaching Computer resources.**

Now, to load the data from Google Cloud Storage, we need to provide the email address highlighted in Figure 9. To do this, click on 'Cluster' at the top of the notebook and then select 'Configuration,' as shown below.

A screenshot of a computer

Description automatically generated

**Figure 15: Configuring GCP to Databricks**

In the 'Configuration' section, under 'Advanced Options,' paste the email highlighted in Figure 8 into the 'Google Service Account' field, as demonstrated below.

A screenshot of a computer

Description automatically generated

**Figure 16: Linking GCP to Databricks**

Now to get the data from GCP to Databricks, the below code is used which gets all the files present in the Google Cloud storage bucket. This code should be run on the notebook created in databricks. The notebook submitted with this report named “nyctaxidataanalysis\_notebook” has all the code that is required for this project. Once after creating the notebook run each and every step till you get the code where you run “big query part as we have to do some steps on big query before running that part. That “BigQuery” chunk of code can be identified easily as it has word “bigquery” in it.

A close up of text

Description automatically generated

**Figure 17: Code to get all the files present in GCP to Databricks**

Now the below code gets all the files after 2016 and stores in 3 dataframes named “yellow\_data”, “green\_data” and “hvfhv\_data” as shown below.

A screenshot of a computer code

Description automatically generated

**Figure 18: Code to read after the year 2015.**

After reading the data, several data pre-processing steps were undertaken, including converting the pickup and dropoff date-time values into datetime format, introducing additional columns (for instance, creating 'Rate Code Name' based on the provided 'Rate Code ID' in compliance with the Data Dictionary), and renaming columns for clarity. We also extracted location geometries from a shape file, which was acquired from the Google Cloud Platform, similar to our data files. All the pre-processing steps can be run from the file named “nyctaxidataanalysis\_notebook” with this submission.

The culmination of these efforts resulted in the creation of a consolidated data frame. This data frame incorporates details from all categories, encompassing yellow taxis, green taxis, and for-hire vehicles while ensuring the presence of common columns across the three data frames. For a comprehensive breakdown of the steps taken, please refer to the 'nyctaxidataanalysis\_notebook' attached to this project submission.

With the data now prepared, the next step is to store it in Google BigQuery, which is a data warehousing tool. We use BigQuery because it's a system that can easily connect to various dashboarding tools. Additionally, Google BigQuery is well-suited for handling the large volumes of data involved in this project.

## **Step 3:**

Now, it's time to load the data into Google BigQuery. On BigQuery also, we have to choose the same project as we used for storing the data. Before loading, it's essential to create a Service account in Google Cloud Storage and provide the necessary details in Databricks to establish the connection with BigQuery. However, it's worth noting that we've already completed this process as depicted in Figure 7, where we granted the required permissions and configured the service account, as illustrated in Figures 11 and 12.

Now, before loading the data into Google BigQuery, it's necessary to create a dataset in Google BigQuery, which serves as a database for storing the four dataframes as tables. You can create the dataset by clicking on the three dots next to the project name and selecting 'Create dataset,' as illustrated below.

A screenshot of a computer

Description automatically generated

**Figure 19: Creating new Dataset in Google BigQuery**

Now the dataset details can be given after clicking on “Create Dataset” as shown below.

A screenshot of a computer

Description automatically generated

**Figure 20: Giving datasets details.**

After creating the dataset in Google BigQuery, we can proceed to load the data into it using the provided code within the Databricks notebook.

A close-up of a computer screen

Description automatically generated**Figure 21: Code to load the yellow data into Big Query**

A screenshot of a computer

Description automatically generated**Figure 22: Code to load the green data into Big Query**

A white background with red and black text

Description automatically generated

**Figure 23: Code to load the for-hire data into Big Query**

A white background with black text

Description automatically generated

**Figure 24: Code to load the combined data frame’s data into Big Query**

The Big data loaded into the Big Query looks like below.

A screenshot of a search engine

Description automatically generated

**Figure 25: The data tables in dataset**

Now, this data can be easily integrated into dashboarding tools, and for this project, we have selected Looker as our dashboarding tool.

## **Step 4:**

Before visualizing the data in Looker, let's start by connecting BigQuery's data to Looker, as demonstrated below. In Looker, you can navigate to a new page, click on 'Add Data,' and then select 'BigQuery.' From there, you can choose the project where your BigQuery data is stored, select the dataset, and choose the table you wish to load, as shown below. We have loaded all the data into Looker, as we intend to create four dashboard pages.

A screenshot of a computer

Description automatically generated

**Figure 26: Loading Data from Big Query in Looker**

## **Step 5:**

The last step is to visualize in the Looker dashboard. The dashboard created can be found [here](https://lookerstudio.google.com/s/pZAdcWwrzVM).