Module: ECS765P - Big Data Processing
Name: Nicholas Reeves
Student Number: 220708234
Assignment: Coursework

Introduction

There are two files and folder containing the data used to produce the plots that are include in the accompanying zip file.

taxi.py: spark job used to produce all data used in the report
Visualization Script - Nicholas Reeves 220708234: notebook used to produce plots

All results were obtained through **one spark job** using the **taxi.py** script. **The application ID follows**

App ID:spark-c64405dffb6847da8db93556a379a1ab

Task 1

Output for both Green Trip and Yellow trip dataframes below

```
Green Trip Schema
root
 |-- lpep_pickup_datetime: string (nullable = true)
 |-- lpep_dropoff_datetime: string (nullable = true)
 |-- PULocationID: string (nullable = true)
 |-- DOLocationID: string (nullable = true)
 |-- passenger_count: string (nullable = true)
 |-- trip_distance: string (nullable = true)
 |-- fare_amount: string (nullable = true)
 |-- extra: string (nullable = true)
 |-- mta_tax: string (nullable = true)
 |-- tip_amount: string (nullable = true)
 |-- tolls_amount: string (nullable = true)
 |-- ehail_fee: string (nullable = true)
 |-- total_amount: string (nullable = true)
 |-- payment_type: string (nullable = true)
 |-- trip_type: string (nullable = true)
 |-- congestion_surcharge: string (nullable = true)
 |-- taxi_type: string (nullable = true)
 |-- Pickup_Borough: string (nullable = true)
 |-- Pickup_Zone: string (nullable = true)
 |-- Pickup_service_zone: string (nullable = true)
 |-- Dropoff_Borough: string (nullable = true)
 |-- Dropoff_Zone: string (nullable = true)
 |-- Dropoff_service_zone: string (nullable = true)

None
There are 465295 records in the green trip dataframe
There are 23 columns in the green trip dataframe
```

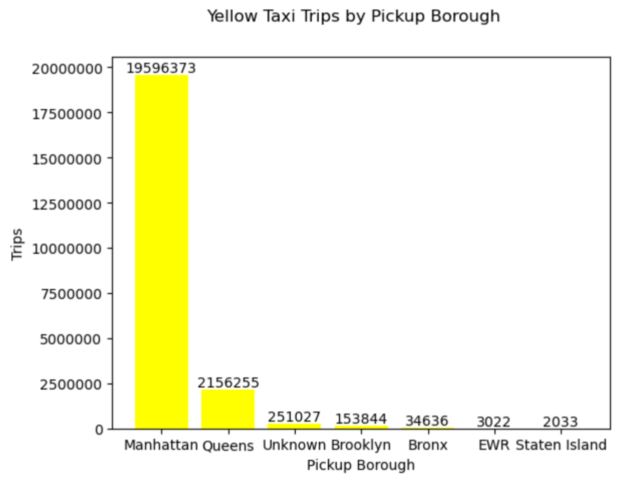
```
Yellow Trip Schema
root
 |-- tpep_pickup_datetime: string (nullable = true)
 |-- tpep_dropoff_datetime: string (nullable = true)
 |-- passenger_count: string (nullable = true)
 |-- trip_distance: string (nullable = true)
 |-- PULocationID: string (nullable = true)
 |-- DOLocationID: string (nullable = true)
 |-- payment_type: string (nullable = true)
 |-- fare_amount: string (nullable = true)
 |-- extra: string (nullable = true)
 |-- mta_tax: string (nullable = true)
 |-- tip_amount: string (nullable = true)
 |-- tolls_amount: string (nullable = true)
 |-- total_amount: string (nullable = true)
 |-- congestion_surcharge: string (nullable = true)
 |-- airport_fee: string (nullable = true)
 |-- taxi_type: string (nullable = true)
 |-- Pickup_Borough: string (nullable = true)
 |-- Pickup_Zone: string (nullable = true)
 |-- Pickup_service_zone: string (nullable = true)
 |-- Dropoff_Borough: string (nullable = true)
 |-- Dropoff_Zone: string (nullable = true)
 |-- Dropoff_service_zone: string (nullable = true)

None
There are 22197190 records in the yellow trip dataframe
There are 22 columns in the yellow trip dataframe
```

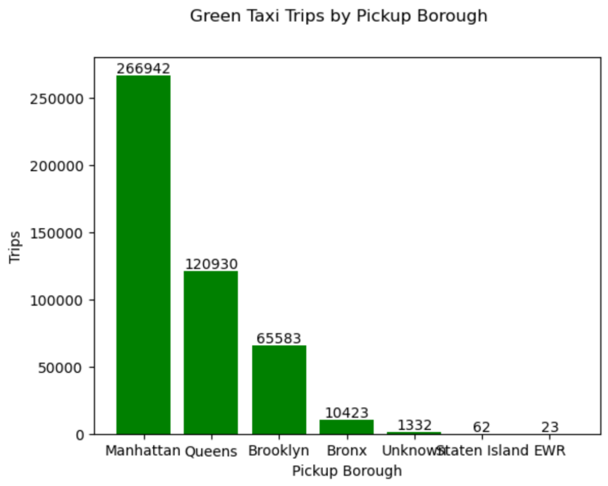
As can be observed in the print statements below each schema, there are 465,295 trips in the green taxi dataframe and 23 columns. There are 22,197,190 trips in the yellowtaxi dataframe and 22 columns. The schemas for both data frames match the specifications.

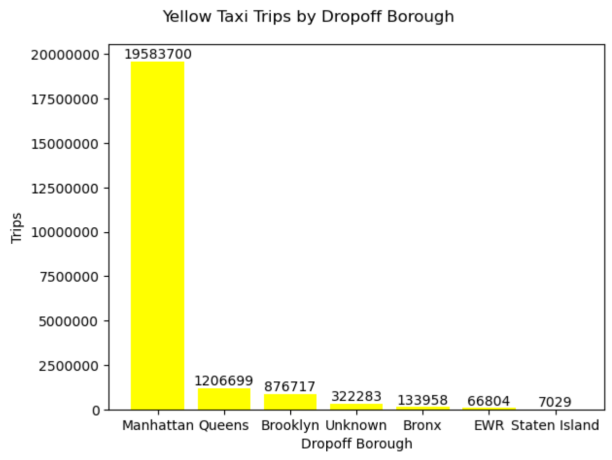
Dataframes were obtained by joining to the taxizone lookup table twice and renaming additional columns. The joining field 'LocationID' in the lookup table was dropped after each join.

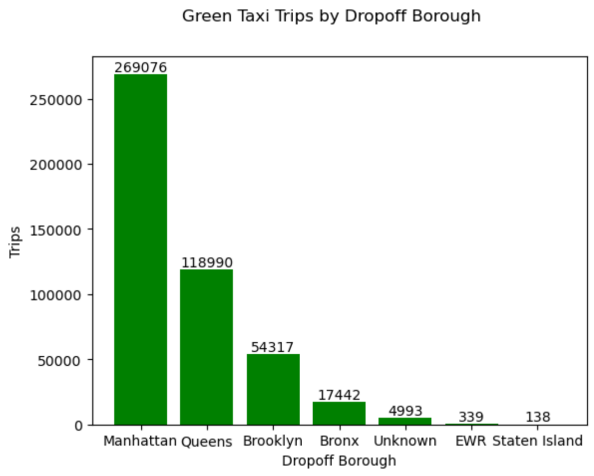
Task 2

See below for four plots of the number of trips by pickup and dropoff borough respectively.





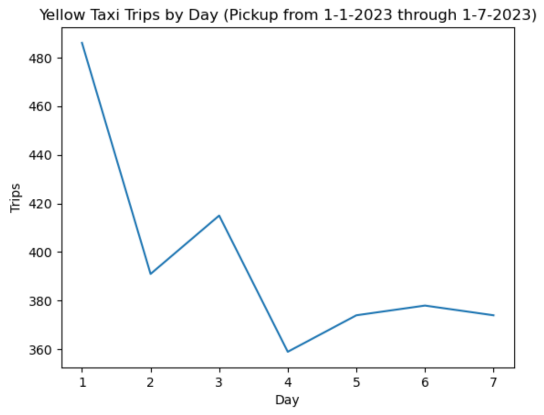




Results were obtained by grouping by the pickup or dropoff borough and counting the total number of rows. Results were then sorted by count in descending order to aid in future tasks.

Task 3

The plot below displays the number of yellow taxi trips with a fare greater than 50 USD and distance less than 1 mile in the first week of January 2023.

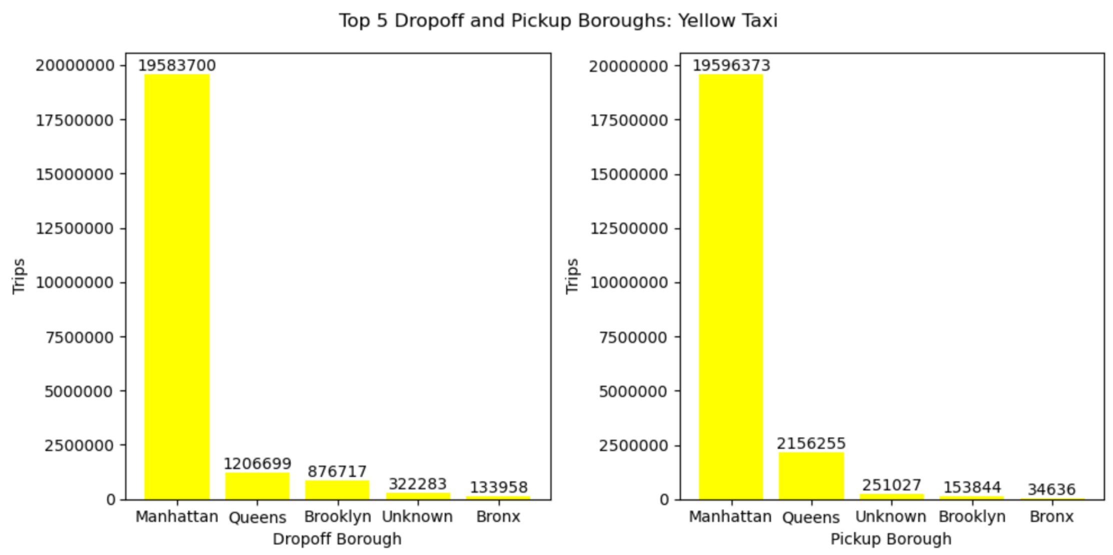


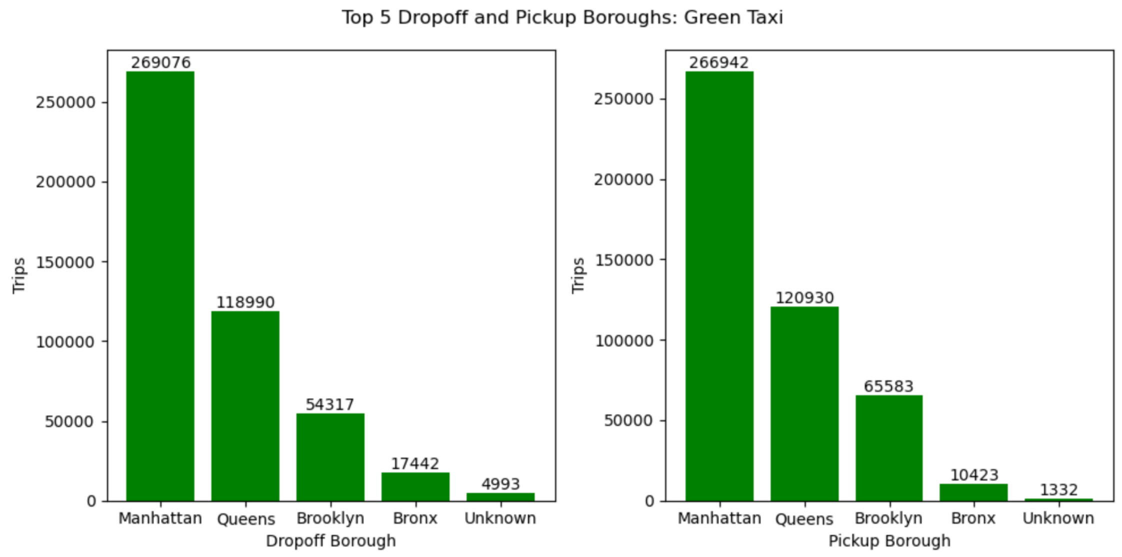
The data was obtained by defining a UDF that removes the timestamp from the imported timestamp strings. The remaining string was converted to a new date type column. The fare and distance columns were also converted to floats as to easier apply the comparison filters and to aid any future computations that included them.

After column conversions, the ‘between’ function was used to filter the DF to only include records that had a pickup date that fell within the relevant date range. A second and third filter were applied to remove any trips with fare less than or equal to 50 dollars and distance greater than or equal to a mile.

Task 4

The plots below display the top five pickup and dropoff boroughs for each taxi service.



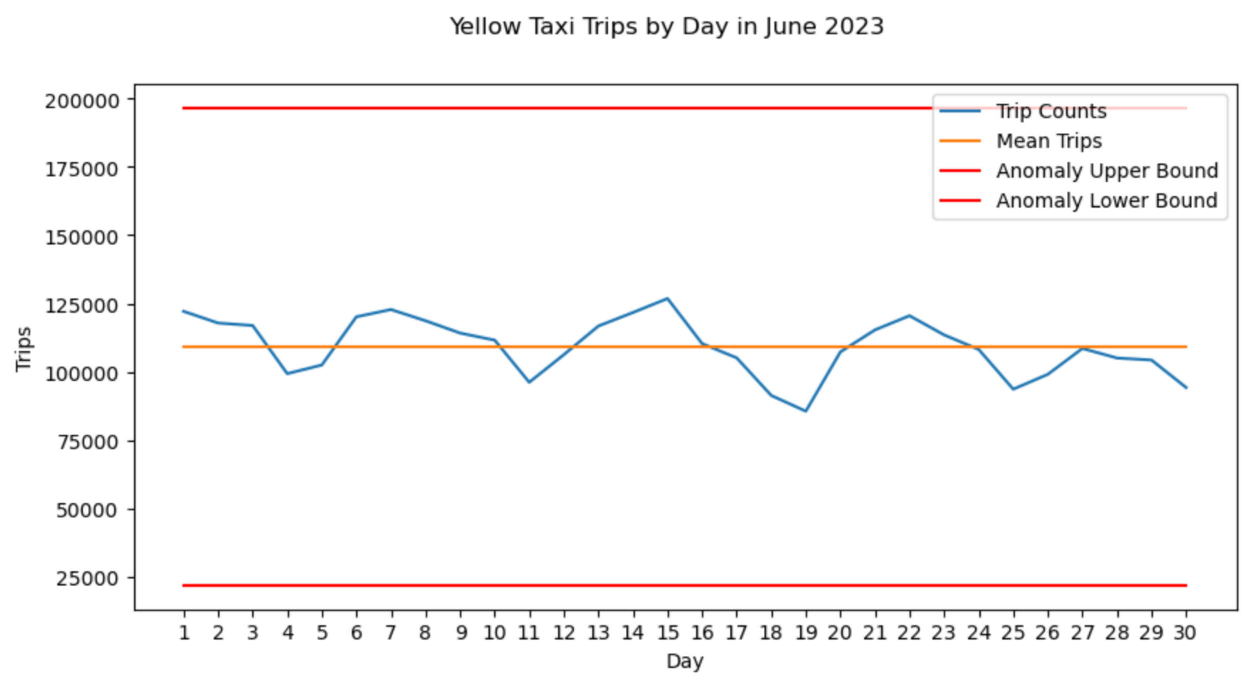


As can be seen, Manhattan and Queens are the first and second most popular boroughs for both pickup and dropoff locations across services.

Results were obtained by using the data produced in task 2. The data had already been sorted, so the only transformation needed was to select the first five records from each dataframe.

Task 5

The plot above shows the number of yellow taxi trips by day in June 2023 (blue).



The orange line represents the mean number of trips each day and was calculated by counting the total number of yellow taxi trips in June and dividing it by the number of days. The mean number trips a day in June was 109199.63

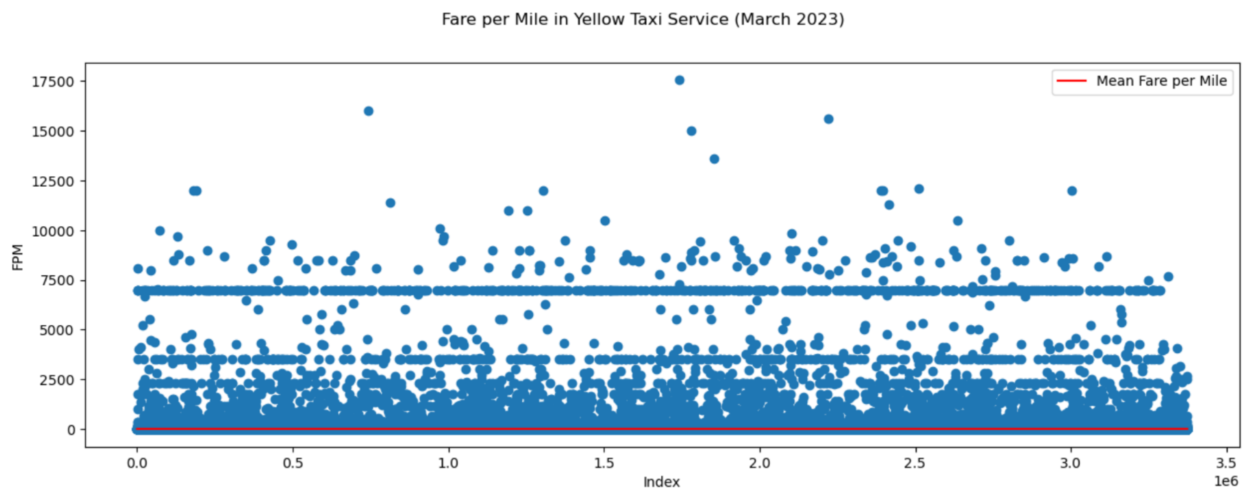
I defined an anomaly as a day in which the trip count was greater than 180% of the mean, or less than 20% of the mean. The red lines represent these boundaries. There were no anomalous days that occurred in June.

There is a clear week based trend in the volume of trips. The number of trips appears to reach a local minima and maxima roughly every seven days. For example, the number of trips reaches a local minima on the 4th, 11th, 19th, and 25th. June 4th 2023 was a sunday, so this likely corresponds to weekly drop in the yellow taxi trip volume around sunday/monday. A similar weekly trend in the local maxima can be observed at midweek.

Data was obtained by applying the same date filtering process that was used Task 3 to reduce the set of yellow taxi trips to the relevant month. Mean trip count was calculated using the filtered dataset. Next, the dataframe was grouped by the day of pickup and counts were aggregated. Columns containing the mean number of trips and anomaly bounds were added to the final dataframe for ease in plotting.

Task 6

The plot below shows the fare per mile (FPM) calculations for all yellow taxi trips that in March 2023. The red line represents the mean fare per mile in the month. Fare per mile is defined as the fare amount over the distance in miles. Mean fare per mile is the average FPM in March 2023.



As you can see there are a large number of points that fall well above the mean FPM, with values as high as 17500 USD to mile. To understand these trips better, I filtered the data frame to only include records with a fare per mile greater than 10000. These trips are considerably different in FPM than the mean of 10.48

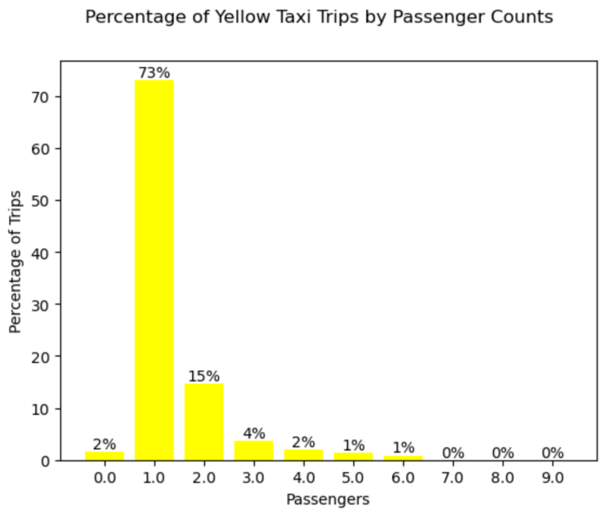
	fare_per_mile	mean_fpm	fare_amount	trip_distance
71257	10000.000224	10.478527	100.00	0.01
180565	12000.000268	10.478527	120.00	0.01
190446	12000.000268	10.478527	120.00	0.01
740741	16000.000358	10.478527	160.00	0.01
812848	11400.000255	10.478527	114.00	0.01
973670	10100.000226	10.478527	101.00	0.01
1193275	11000.000246	10.478527	110.00	0.01

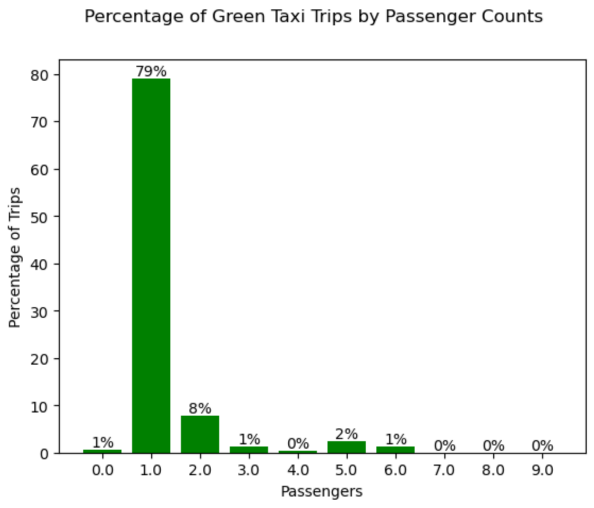
It was clear from visual inspection of the data that a notable amount of these records can be explained by short trips with large taxi fares.

These plots were created using the full output from the spark job. The task 6 csv I submitted is a random sample of the original dataframe as the output was too large to upload. The visualization and dataframe output will look slightly different when the task 6 section of the notebook is run.

The data was produced by applying the same method of date filtering used in task five and task three. Fare per mile was calculated by dividing the fare amount column by the trip distance column. Mean fare per mile was calculated by summing the fare per mile column and dividing the total by the number of yellow taxi trips in the month.

Task 7





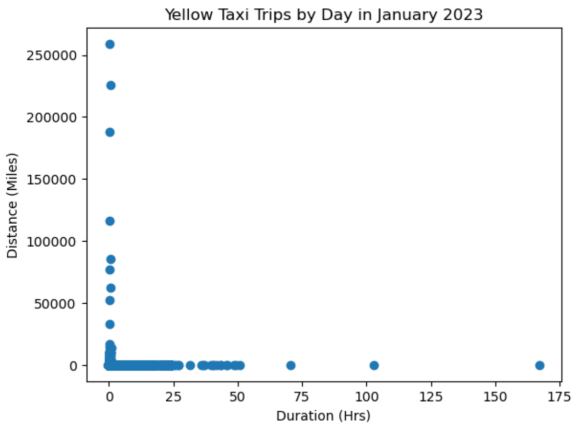
The plots above show the percentage of total trips by passenger count for both taxi services. Note that 2.8% of the yellow taxi trips and 7% of the green taxi trips had a blank number of passengers entered and are not displayed in the plot.

Solo trips are much more common than trips with any other number of passengers. 73% of yellow taxi trips and 79% of green taxi trips had one passenger. The next most common number of passengers was two, making up 15% and 8% of the total trips in the dataset for yellow and green respectively.

Data was obtained by taking the full dataframes, grouping by the passenger number column, and applying counts. Percentages were obtained by dividing the aggregated counts by the total number of trips in each dataframe.

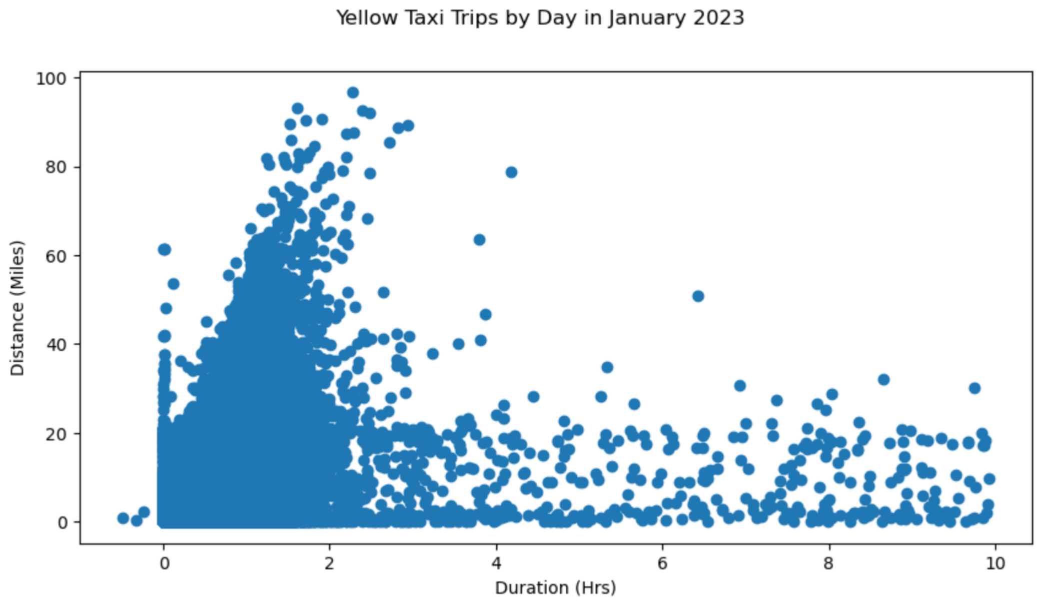
Task 8

The following section describes the relationship between trip distance (miles) and duration (hours) for all yellow taxi trips that occurred in January 2023.



The plot above shows the relationship between duration and distance for all yellow taxi trips in January 2023. It is difficult to interpret any trend as there are some outliers with either extremely long durations or distances that skew the scale.

To improve interpretability I removed trips with distance greater than 100 miles or duration longer than 10 hours from the plot. It is a 50 mile drive from South Staten Island and a 10 hour drive from NYC to Charlotte North Carolina. For these reasons I think its possible these records are inaccurate representations of the trips that occurred.



A clear relationship emerges with the outliers excluded. An increase in duration appears to relate to an increase in trip distance for trips with duration less than 2 hours.

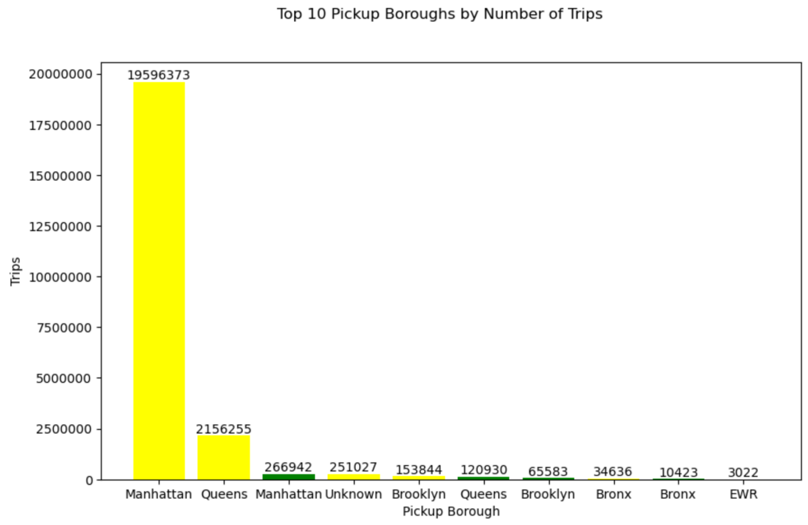
Some trips can be observed on the y axis that took close to 0 hours but traversed 20-60 miles. Considering the legal speed limit in most of the U.S. is 60 miles per hour, I think it is fair to say these are likely to be data errors.

The relationship appears to disappear with trips longer than 2-2.5 hours. Its possible this could be explained by data issues or extreme cases of traffic.

This data was constructed by using the date filtering method used in prior tasks to reduce the yellow taxi data to trips that occurred in January 2023. Datetime strings for pickup and dropoff times were then converted to timestamps and the unix timestamp function was applied. From there, difference in seconds was calculated using subtraction to retrieve trip duration in seconds. This value was divided by 3600 to receive trip duration in hours.

Task 9

The following barchart displayed the top 10 boroughs by number of pickups and taxi service.



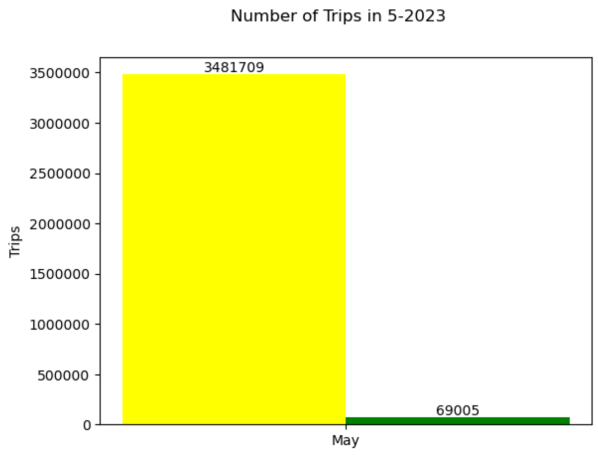
The yellow taxi service makes up six of the 10 bars and the green taxi service makes up four of the 10 bars. The most pickups occurred in Manhattan by the yellow taxi service.

The green taxi company is less active in all boroughs than the yellow taxi company, however it appears to be more evenly distributed in its pickup activity across boroughs. The yellow taxi service is heavily skewed towards service in Manhattan performing almost 10 times more pickups in this borough than in its second most active borough (Queens).

This plot was created using the pickup counts by borough data produced in task 2. Service color columns were created for both the yellow and green dataframes. The dataframes are combined using a union and then sorted in descending order based on the counts. The first 10 records are saved.

Task 10

The month with the most total trips is shown below.



There were 3,550,714 trips in May 2023. 3,481,709 of these were yellow taxi trips and 69,005 were green taxi trips.

This data was produced by grouping the green and yellow dataframes by pickup month and aggregating counts of the rows. The month column was constructed using the PySpark SQL function ‘month’ and applying it to the date type column for pickup date that was produced in task 3.

The yellow and green dataframes were combined using an inner join on pickup month. Combined trip count was calculated by adding the two count columns in the joined dataset. This dataframe was sorted in descending order by the total counts and the first record was selected.