Report: Optimising NYC Taxi Operations

Include your visualisations, analysis, results, insights, and outcomes. Explain your methodology and approach to the tasks. Add your conclusions to the sections.

1. Data Preparation

1.1. Loading the dataset

1.1.1. Sample the data and combine the files

The dataset was loaded from 12 parquet files and combined into a single DataFrame. A sample of the 5% of data was extracted for efficiency in processing.

2. Data Cleaning

2.1. Fixing Columns

2.1.1. Fix the index

I simply check for the index and it looked good, from 0 to 21

2.1.2. Combine the two airport_fee columns

Airport_fee and airport_fee, two columns were appearing which I combined to handle redundancy.

2.2. Handling Missing Values

2.2.1. Find the proportion of missing values in each column

Using below command I got the percentage of missing values in data

- 100 * df2.isnull().mean()

2.2.2. Handling missing values in passenger_count

Missing values in Passenger_Count handled using median.

2.2.3. Handle missing values in RatecodeID

Missing values in RatecodeID were handled using **mode** which were populate maximum appearing values.

2.2.4. Impute NaN in congestion_surcharge

I have replaced NaN values in Congestion Surcharge with median to handle null values.

Code- #replacing nulls with median

df2['congestion_surcharge'] = df2['congestion_surcharge'].fillna(df2['congestion_surcharge'].median())

#Printing the result

missing_values = df2['congestion_surcharge'].isnull().sum()

print(f"Missing values in 'congestion_surcharge': {missing_values}")

2.3. Handling Outliers and Standardising Values

2.3.1. Check outliers in payment type, trip distance and tip amount columns

Outliers were checked in payment type, trip distance, and tip amount and standardised to ensure the fare consistency

3. Exploratory Data Analysis

3.1. General EDA: Finding Patterns and Trends

3.1.1. Classify variables into categorical and numerical

Numerical columns (int and float)

numerical_cols = df2.select_dtypes(include=['int64', 'float64']).columns.tolist()

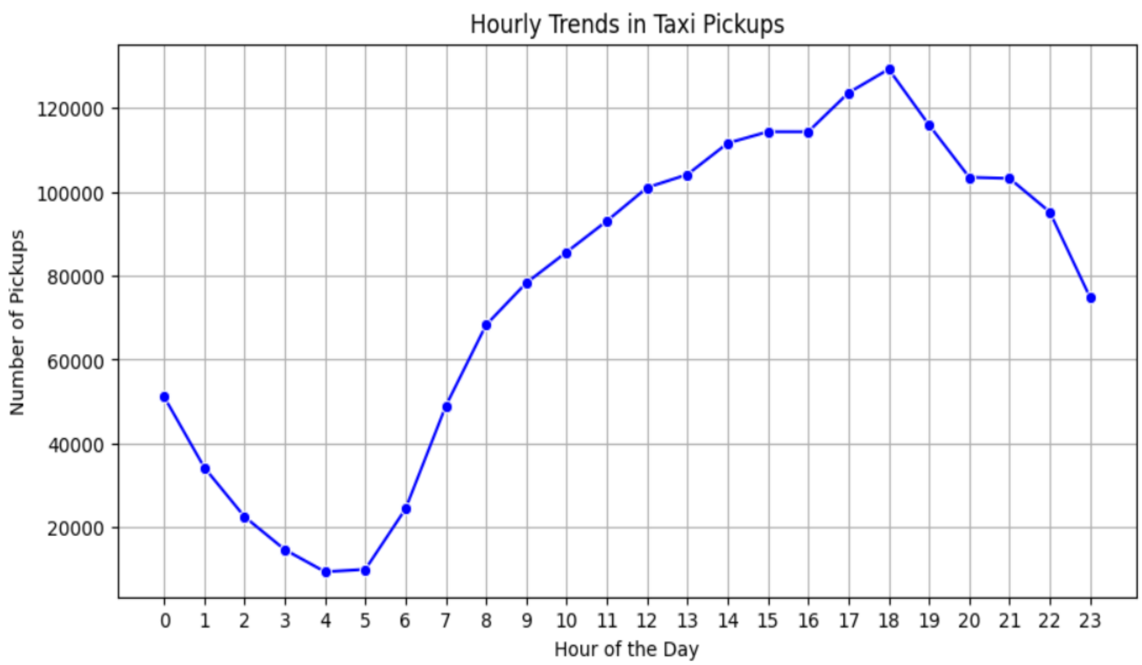
Categorical columns (object and category)

```
categorical_cols = df2.select_dtypes(include=['object',
'category']).columns.tolist()
```

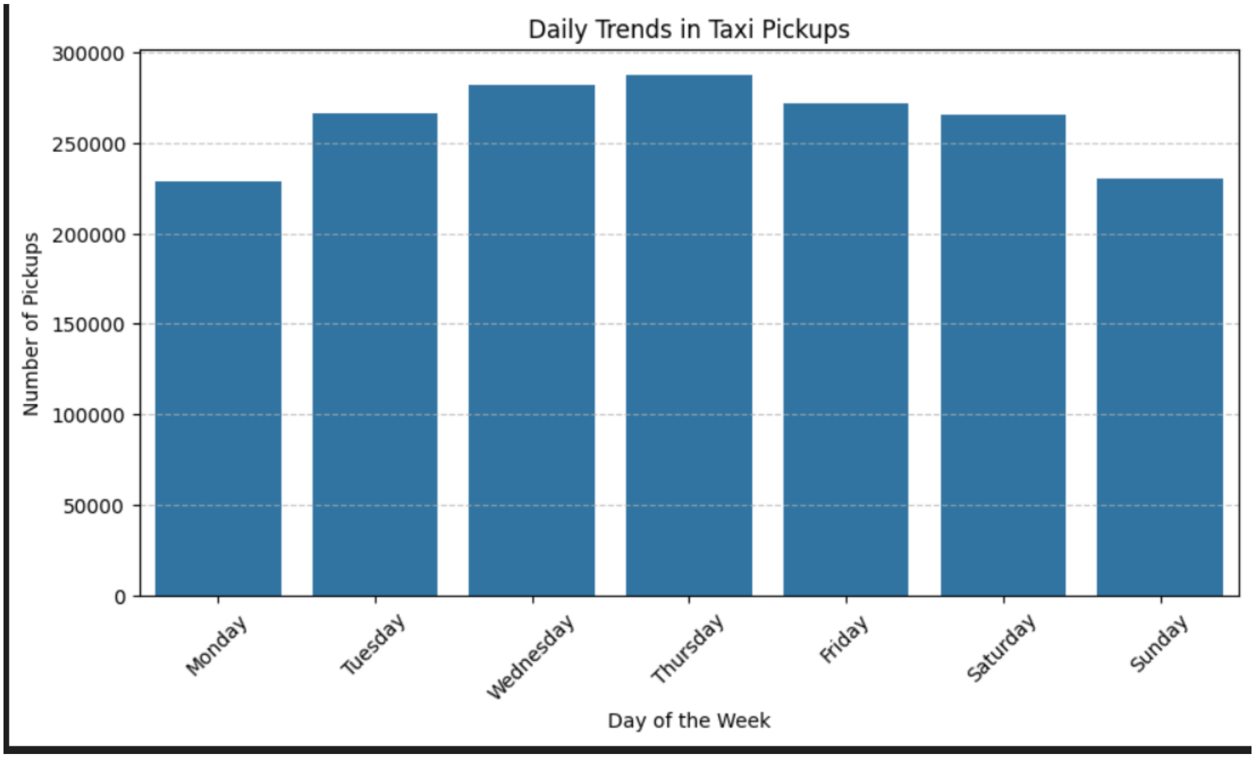
Used above commands to categorised the columns.

3.1.2. Analyse the distribution of taxi pickups by hours, days of the week, and months

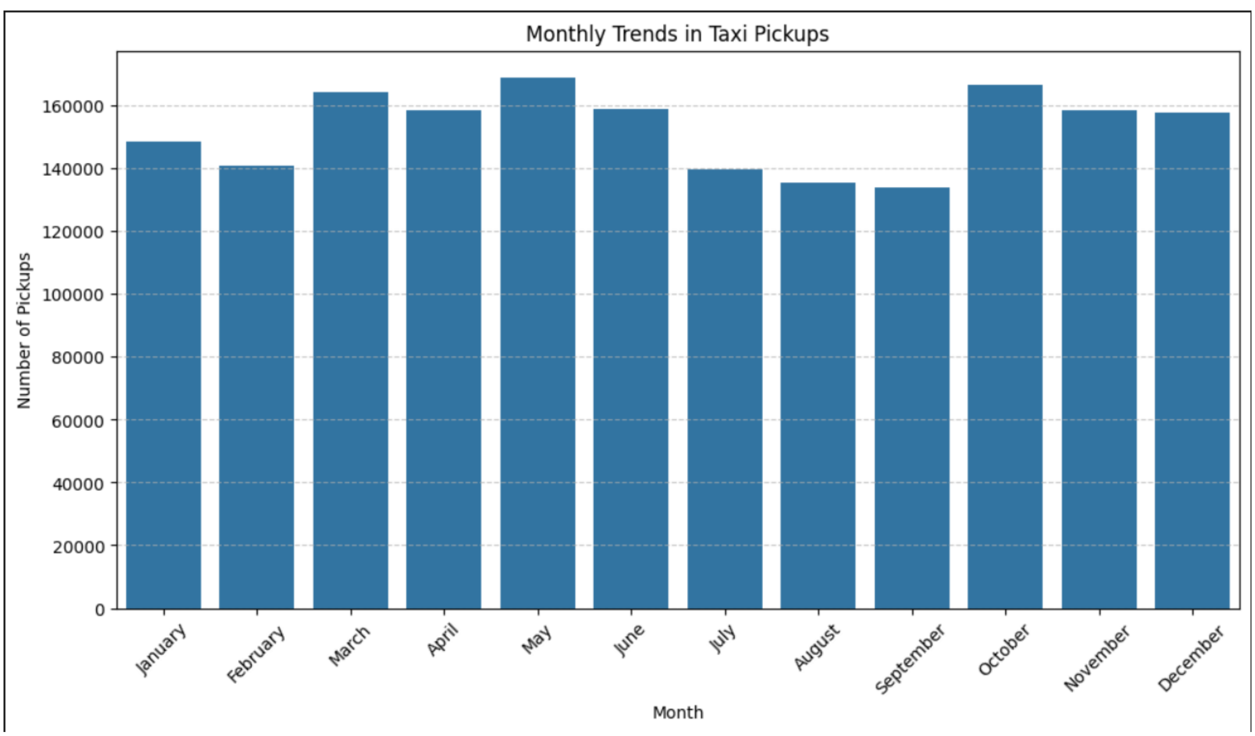
a. **Taxi pickup by hours:** During evening hours taxi pickups increases. Value is highest @6 PM.



b. **Taxi pickup by Day:** During mid of the week taxi pickups increases. Value is highest on thrusday



a. **Taxi pickup by Month:** Taxi pickup in quarter second and quarter four are highest.



3.1.3. Filter out the zero/negative values in fares, distance and tips

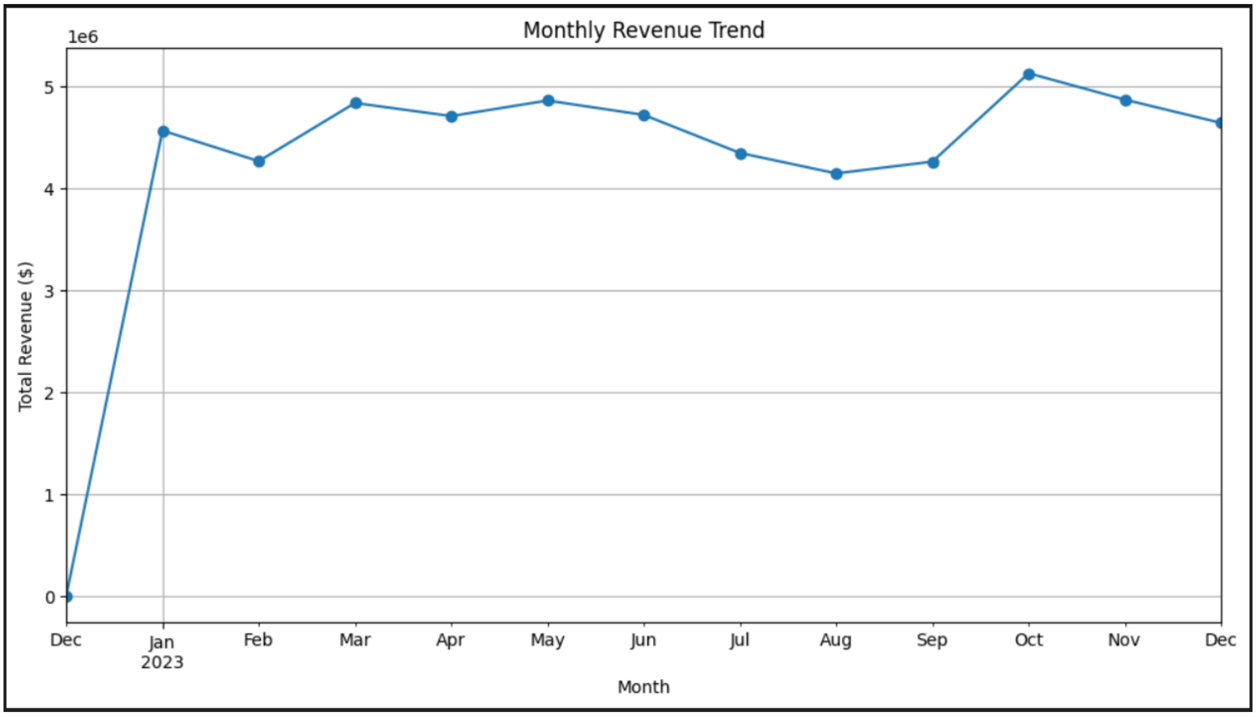
columns_to_check = ['fare_amount', 'tip_amount', 'total_amount', 'trip_distance']

df2[columns_to_check].describe()

Used above code to check the negative and zero values

3.1.4. Analyse the monthly revenue trends

Monthly revenue is highest in Oct month'23 and lowest in Feb and Aug'23



3.1.5. Find the proportion of each quarter's revenue in the yearly revenue

year_quarter

2022Q4 0.000067

2023Q1 24.694513

2023Q2 25.816265

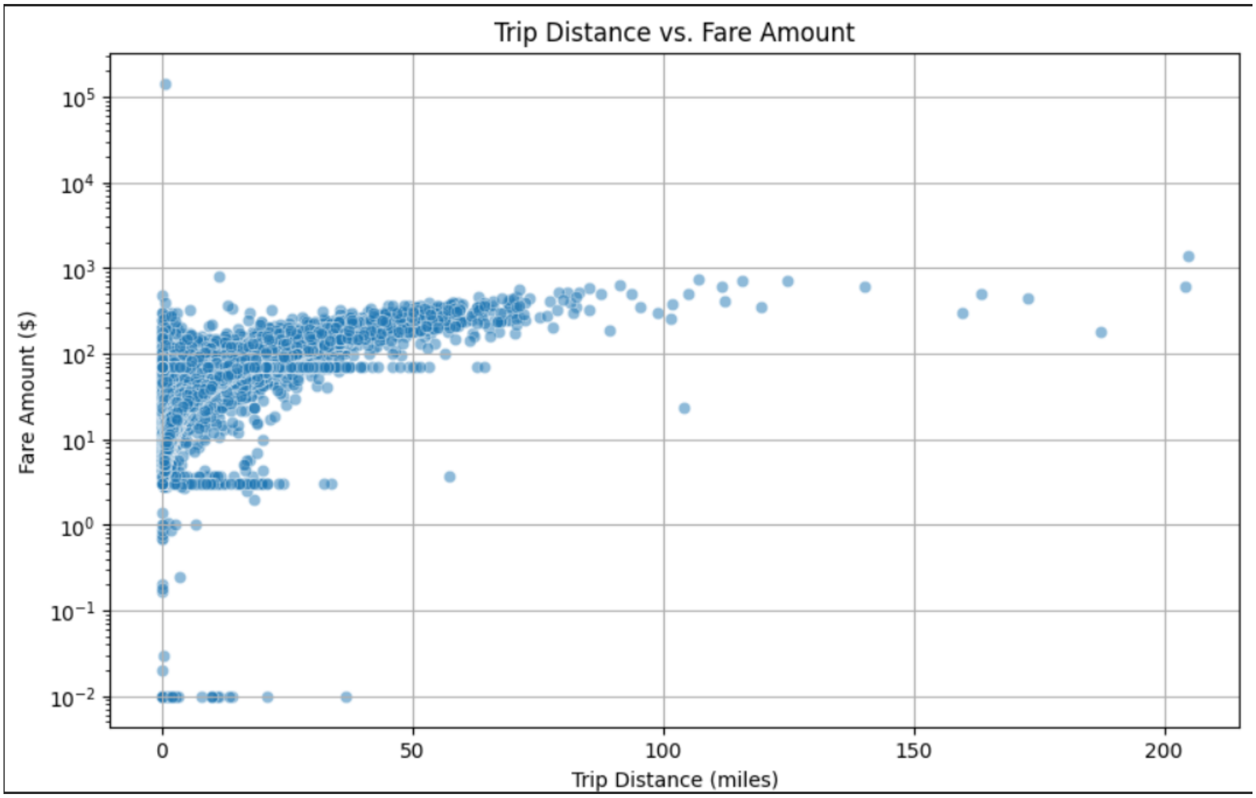
2023Q3 23.040803

2023Q4 26.448352

Revenue is highest in Q4

3.1.6. Analyse and visualise the relationship between distance and fare amount

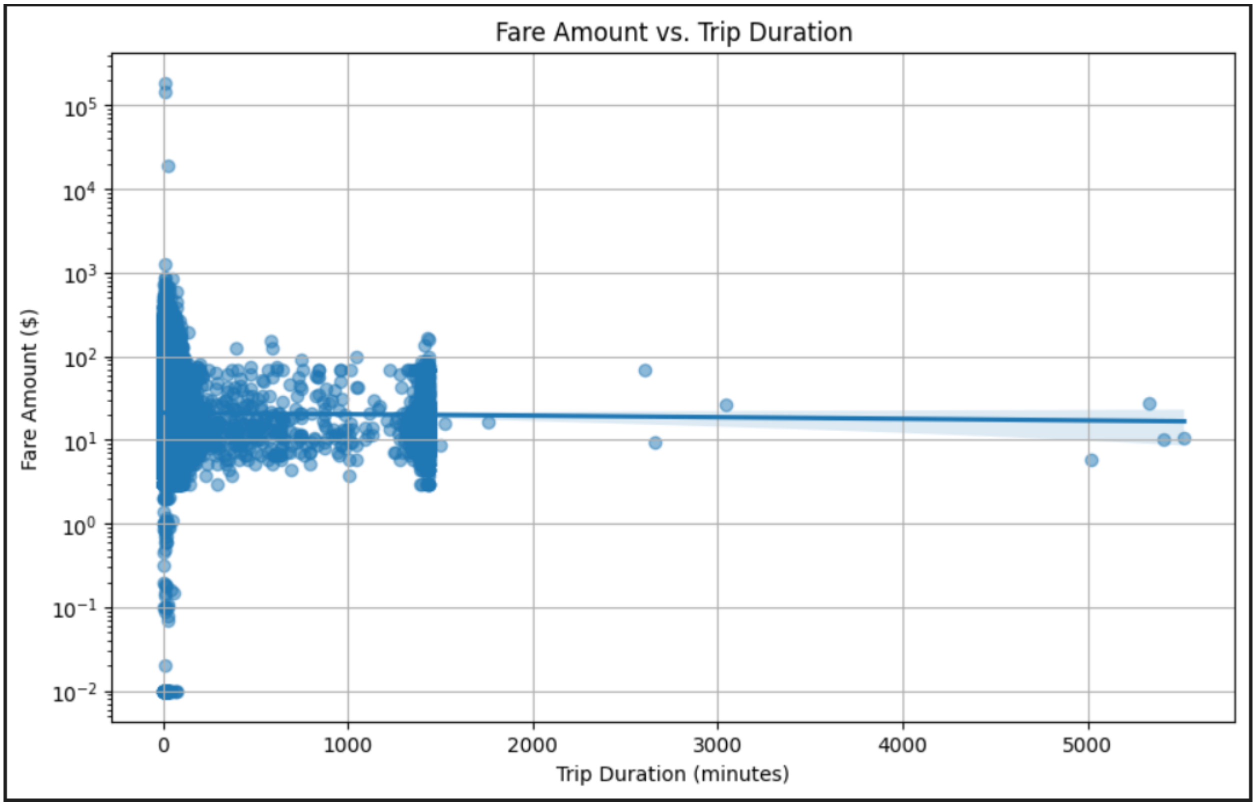
a. A positive correlation between trip distance and fare was observed
b. Correlation between Trip Distance and Fare Amount: 0.1564



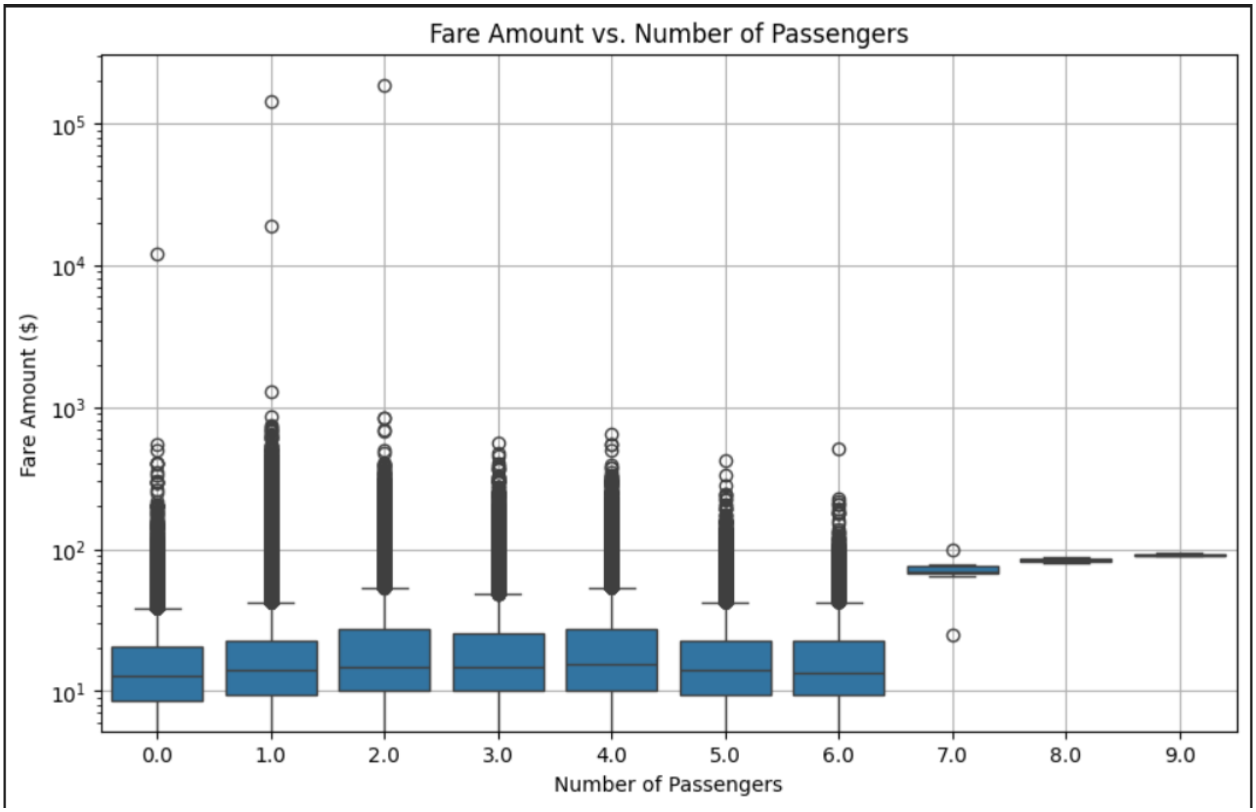
3.1.7. Analyse the relationship between fare/tips and trips/passengers

A. Correlation is negative

B. Correlation between Trip Duration and Fare Amount: -0.0002

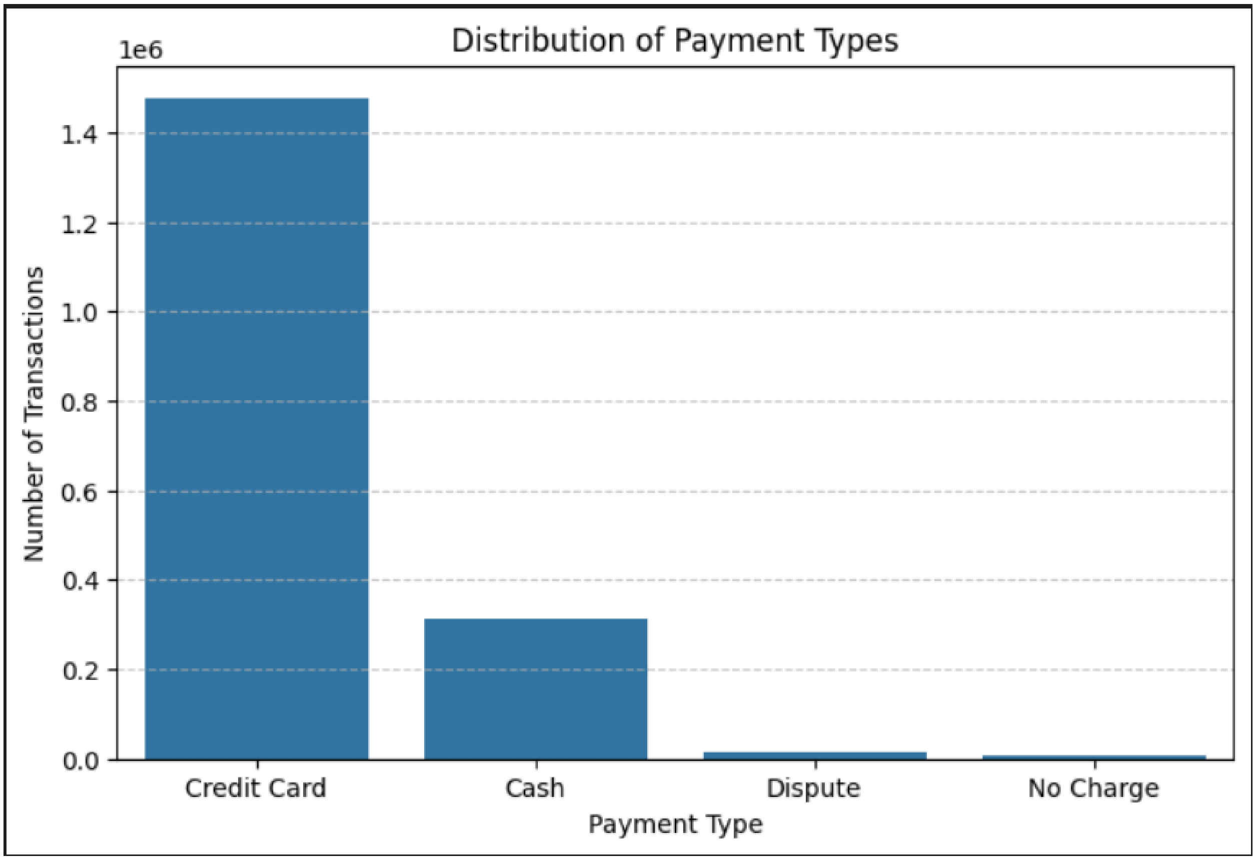


a. Correlation between Passenger Count and Fare Amount: 0.0067 and positive



Fare Amount vs. Number of Passengers
Fare Amount ($)
10^5
10^4
10^3
10^2
10^1
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
Number of Passengers

3.1.8. Analyse the distribution of different payment types



3.1.9. Load the taxi zones shapefile and display it

import geopandas as gpd

Read the shapefile using geopandas

zones =gpd.read_file(r"C:\Users\Vedan\Downloads\Datasets and Dictionary-NYC\Datasets and Dictionary\taxi_zones\taxi_zones.shp")

zones.head()

3.1.10. Merge the zone data with trips data

```
# Merge zones and trip records using locationID and PULocationID


df_merged = df2.merge(zones, left_on='PULocationID', right_on='LocationID', how='left')

# Display the first few rows of merged data
print(df_merged.head())
```

3.1.11. Find the number of trips for each zone/location ID

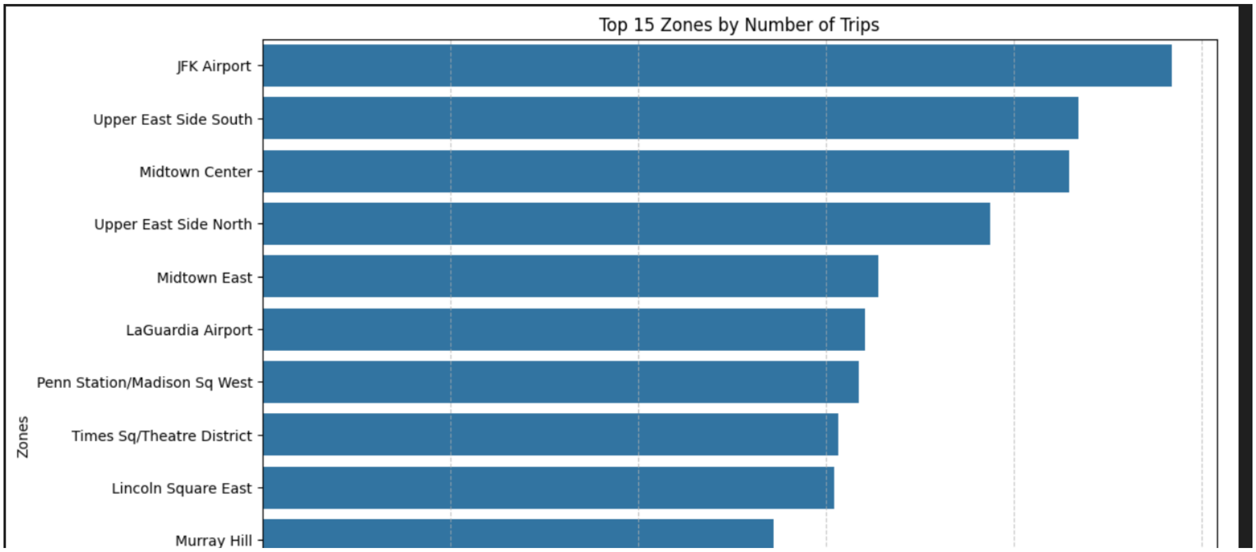
```
# Group data by location and calculate the number of trips

trip_counts = df2.groupby('PULocationID').size().reset_index(name='trip_count')

# Sort by the number of trips (descending order)
trip_counts = trip_counts.sort_values(by='trip_count', ascending=False)

# Display the top locations with highest trips
print(trip_counts.head())
```

```
     PULocationID  trip_count
125           132       96803
229           237       86904
154           161       85946
228           236       77516
155           162       65634
```

3.1.12. Add the number of trips for each zone to the zones dataframe

```
# Merge trip counts back to the zones GeoDataFrame

# Step 1: Group trip data by PULocationID and count trips
trip_counts = df2.groupby('PULocationID').size().reset_index(name='trip_count')

# Step 2: Merge trip counts back to the zones GeoDataFrame
zones_merged = zones.merge(trip_counts, left_on='LocationID', right_on='PULocationID', how='left')

# Fill NaN values (for locations with no trips) with 0
zones_merged['trip_count'] = zones_merged['trip_count'].fillna(0)

# Display the first few rows
print(zones_merged.head())
```

3.1.13. Plot a map of the zones showing number of trips



3.1.14. Conclude with results

- The analysis discloses that taxi demand peaks in the evenings, mid-week, and during Q4 of the year. A positive correlation exists between distance and fare, while tip percentages vary based on factors like distance and pickup time. Optimizing taxi dispatching in high-traffic zones and adjusting pricing strategies based on peak demand can enhance operational efficiency and revenue.

3.2. Detailed EDA: Insights and Strategies

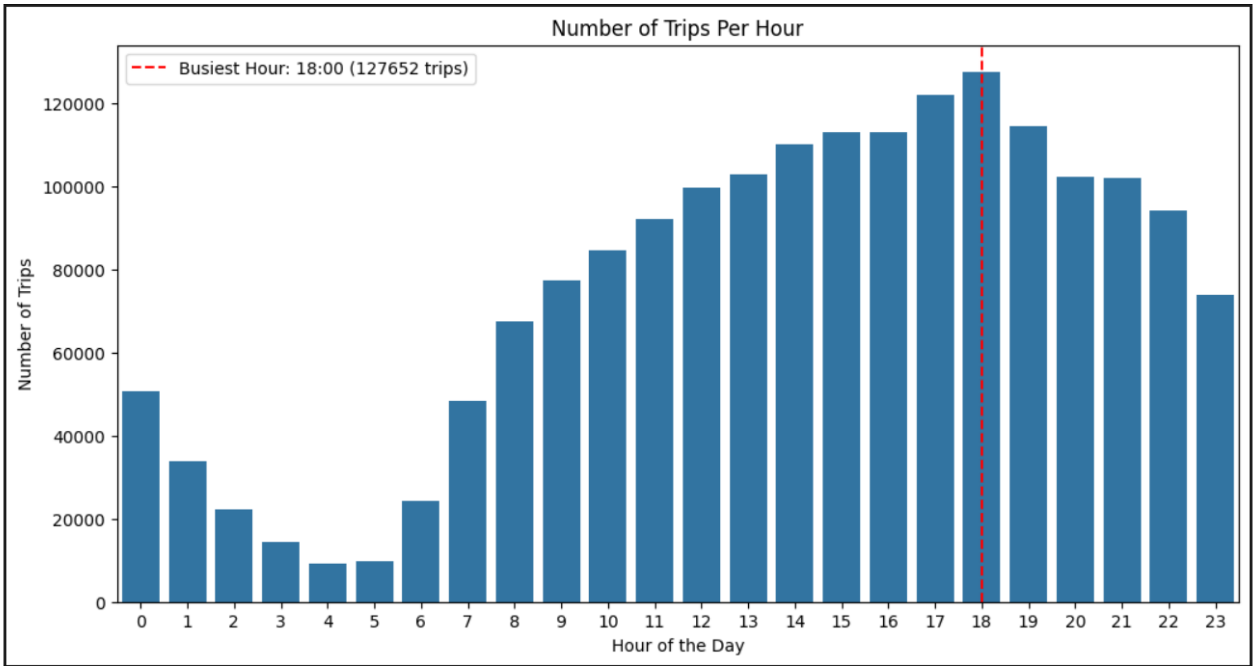
3.2.1. Identify slow routes by comparing average speeds on different routes

During night time on below PUlocation speed is comparatively slow

	PULocationID	DOLocationID	time_of_day	speed_mph
620	10	1	Night	0.0
717	10	137	Night	0.0
1424	13	195	Night	0.0
3870	41	205	Night	0.0
6998	56	230	Night	0.0
7276	62	39	Night	0.0
8310	68	130	Night	0.0
8737	69	232	Night	0.0
10085	75	13	Night	0.0
11620	80	255	Night	0.0

3.2.2. Calculate the hourly number of trips and identify the busy hours

6PM is the busiest hour



3.2.3. Scale up the number of trips from above to find the actual number of trips

```
   scaled_revenue  scaled_distance
0           411.5             77.4
1           154.8             12.4
2           164.0             14.4
3           115.0              5.4
5           474.0             71.0
Total Scaled Revenue: $524,044,159.10
Total Scaled Distance: 62,398,849.60 miles
```

Code- # Define your sampling fraction

sampling_fraction = 0.1 #Taking 10% for sampling

Scale up the number of trips

df['scaled_revenue'] = df2['total_amount'] / sampling_fraction

df['scaled_distance'] = df2['trip_distance'] / sampling_fraction

Display the first few rows

print(df[['scaled_revenue', 'scaled_distance']].head())

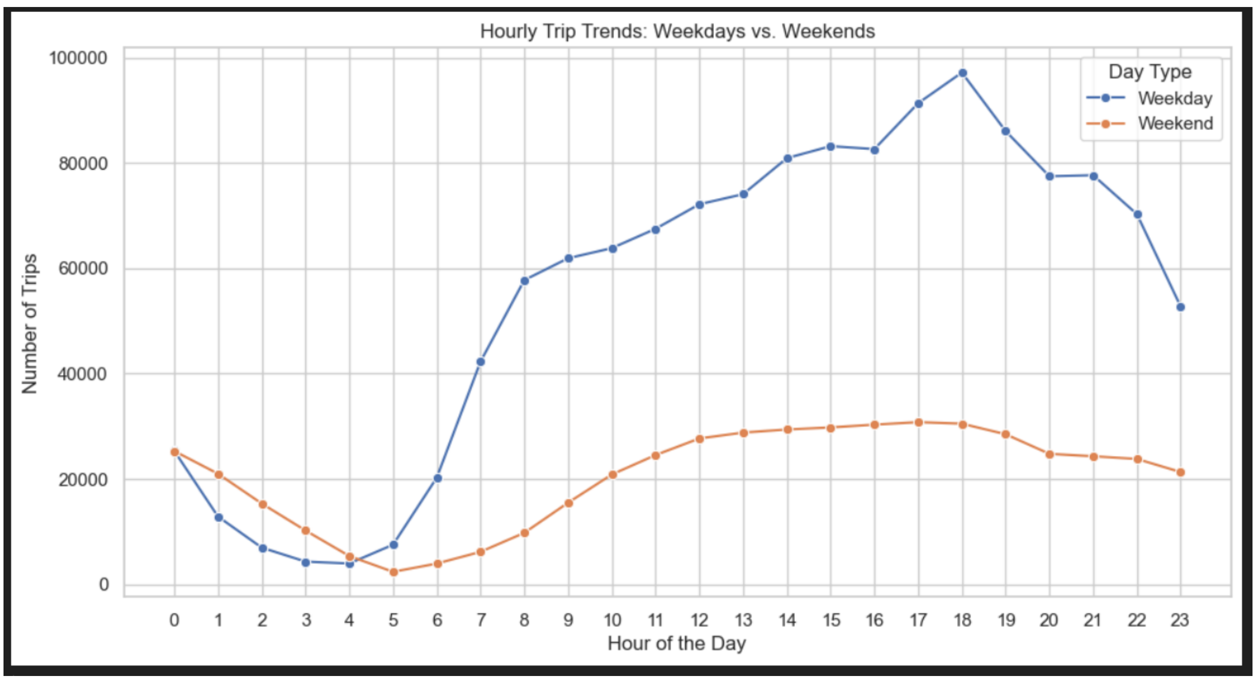
```
# Print total scaled values
total_scaled_revenue = df['scaled_revenue'].sum()
total_scaled_distance = df['scaled_distance'].sum()

print(f"Total Scaled Revenue: ${total_scaled_revenue:,.2f}")
print(f"Total Scaled Distance: {total_scaled_distance:,.2f} miles")
```

3.2.4. Compare hourly traffic on weekdays and weekends

Code-

```
# Ensure datetime conversion
df['tpep_pickup_datetime'] = pd.to_datetime(df2['tpep_pickup_datetime'])

# Extract hour and day of the week
df['hour'] = df['tpep_pickup_datetime'].dt.hour
df['day_of_week'] = df['tpep_pickup_datetime'].dt.dayofweek  # Monday=0, Sunday=6

# Define weekday vs weekend
df['day_type'] = df['day_of_week'].apply(lambda x: 'Weekday' if x < 5 else 'Weekend')

# Aggregate trip counts per hour
trips_per_hour = df.groupby(['day_type', 'hour']).size().reset_index(name='trip_count')

# Plot
plt.figure(figsize=(12, 6))
sns.set_theme(style="whitegrid")
```

```
sns.lineplot(data=trips_per_hour, x='hour', y='trip_count', hue='day_type', marker="o")

# Labels and title
plt.xlabel("Hour of the Day")
plt.ylabel("Number of Trips")
plt.title("Hourly Trip Trends: Weekdays vs. Weekends")
plt.xticks(range(0, 24))
plt.legend(title="Day Type")
plt.show()
```



3.2.5.Identify the top 10 zones with high hourly pickups and drops

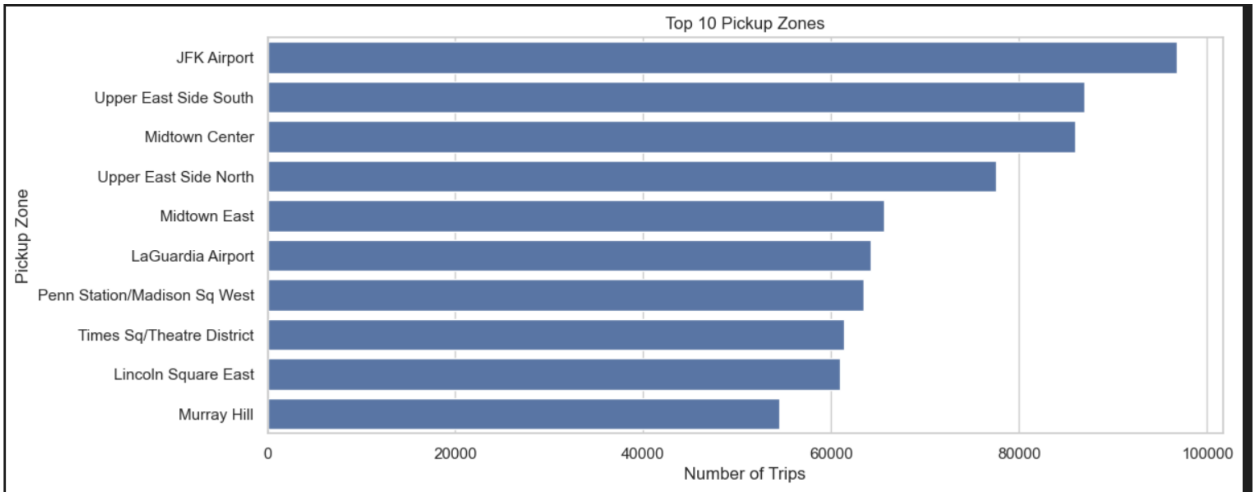
Code- # Find top 10 pickup and dropoff zones

Step 1: Count top 10 pickup zones

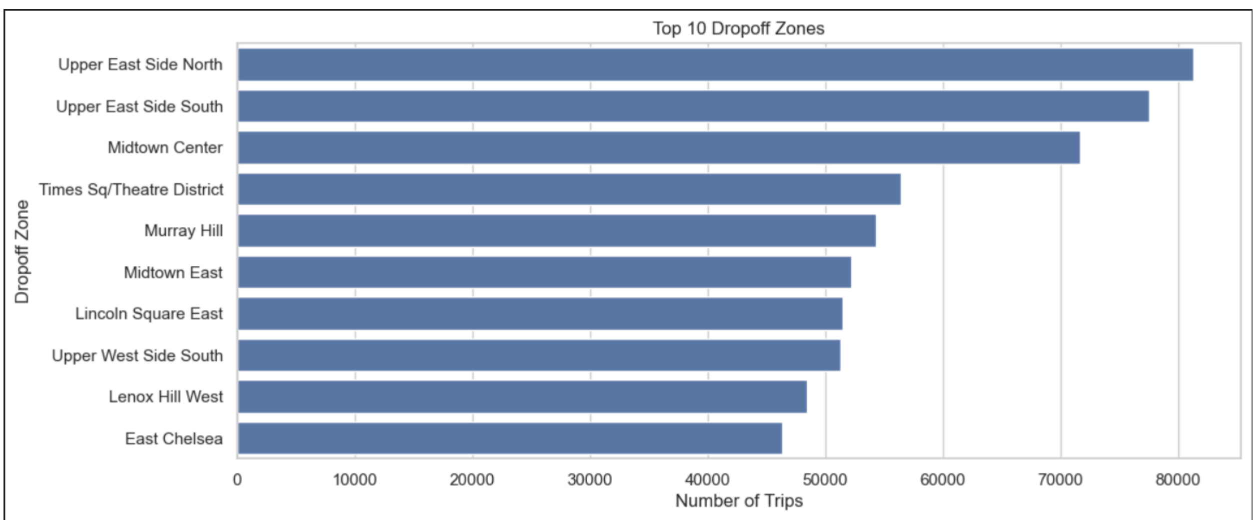
```
top_pickup_zones = df2['PULocationID'].value_counts().head(10).reset_index()

top_pickup_zones.columns = ['LocationID', 'trip_count']


# Step 2: Count top 10 dropoff zones

top_dropoff_zones = df2['DOLocationID'].value_counts().head(10).reset_index()

top_dropoff_zones.columns = ['LocationID', 'trip_count']


# Step 3: Merge with zone names (if available)

if 'zone' in zones.columns:  # Assuming 'zones' has a mapping of LocationID to zone names

    top_pickup_zones = top_pickup_zones.merge(zones[['LocationID', 'zone']],
on='LocationID', how='left')

    top_dropoff_zones = top_dropoff_zones.merge(zones[['LocationID', 'zone']],
on='LocationID', how='left')


# Step 4: Plot Top 10 Pickup Zones

plt.figure(figsize=(12, 5))

sns.barplot(x=top_pickup_zones['trip_count'], y=top_pickup_zones['zone'])

plt.xlabel("Number of Trips")

plt.ylabel("Pickup Zone")

plt.title("Top 10 Pickup Zones")

plt.show()


# Step 5: Plot Top 10 Dropoff Zones

plt.figure(figsize=(12, 5))

sns.barplot(x=top_dropoff_zones['trip_count'], y=top_dropoff_zones['zone'])

plt.xlabel("Number of Trips")

plt.ylabel("Dropoff Zone")
```

```
plt.title("Top 10 Dropoff Zones")

plt.show()
```

Top 10 Pickup Zones:



Top 10 Dropoff Zones:



3.2.6. Find the ratio of pickups and dropoffs in each zone

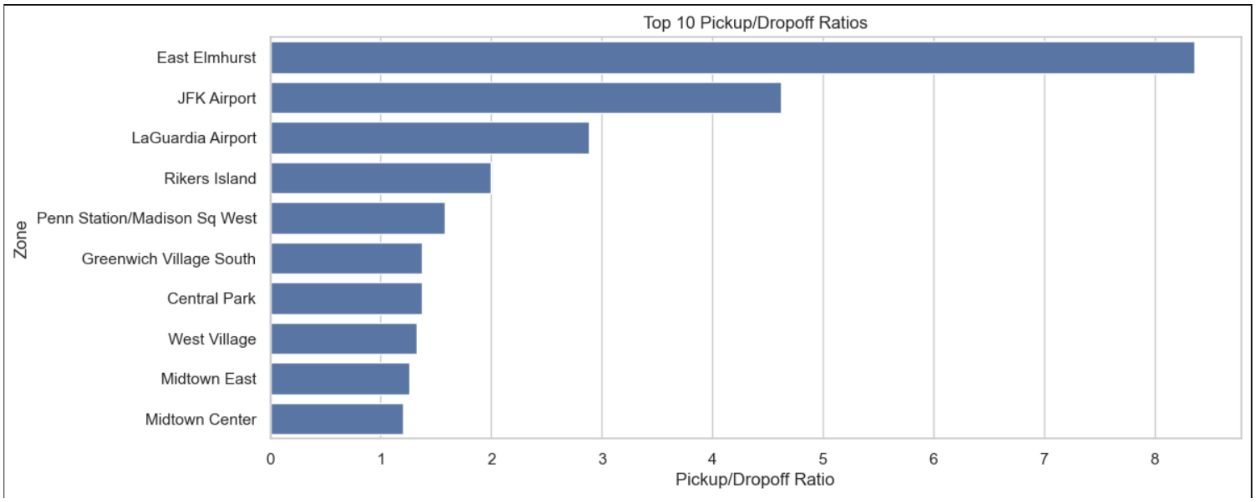
Code: # Find the top 10 and bottom 10 pickup/dropoff ratios

```
# Step 1: Count total pickups and dropoffs

pickup_counts = df2['PULocationID'].value_counts().reset_index()
```

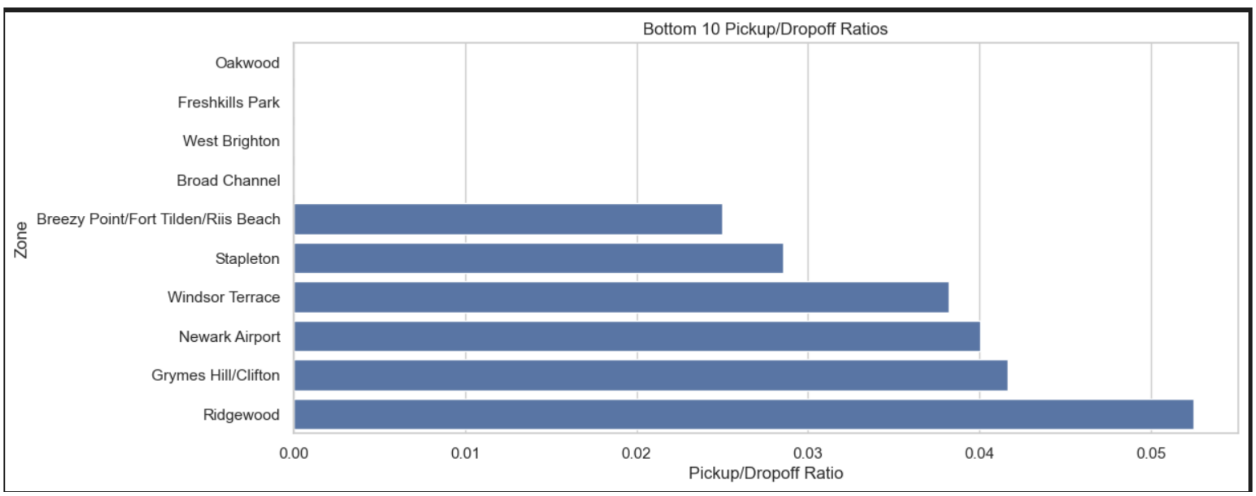
```
pickup_counts.columns = ['LocationID', 'pickup_count']


dropoff_counts = df2['DOLocationID'].value_counts().reset_index()

dropoff_counts.columns = ['LocationID', 'dropoff_count']


# Step 2: Merge pickup and dropoff counts

zone_ratios = pickup_counts.merge(dropoff_counts, on='LocationID', how='outer').fillna(0)


# Step 3: Compute the pickup/dropoff ratio

zone_ratios['pickup_dropoff_ratio'] = zone_ratios['pickup_count'] /
(zone_ratios['dropoff_count'] + 1)  # Avoid division by zero


# Step 4: Merge with zone names

if 'zone' in zones.columns:

    zone_ratios = zone_ratios.merge(zones[['LocationID', 'zone']], on='LocationID',
how='left')


# Step 5: Sort and extract top/bottom 10

top_10_ratios = zone_ratios.sort_values(by='pickup_dropoff_ratio',
ascending=False).head(10)

bottom_10_ratios = zone_ratios.sort_values(by='pickup_dropoff_ratio',
ascending=True).head(10)


# Step 6: Plot the Top 10 Ratios

plt.figure(figsize=(12, 5))

sns.barplot(x=top_10_ratios['pickup_dropoff_ratio'], y=top_10_ratios['zone'])

plt.xlabel("Pickup/Dropoff Ratio")

plt.ylabel("Zone")
```

```
plt.title("Top 10 Pickup/Dropoff Ratios")

plt.show()
```

Top 10 Pickup Ratio:



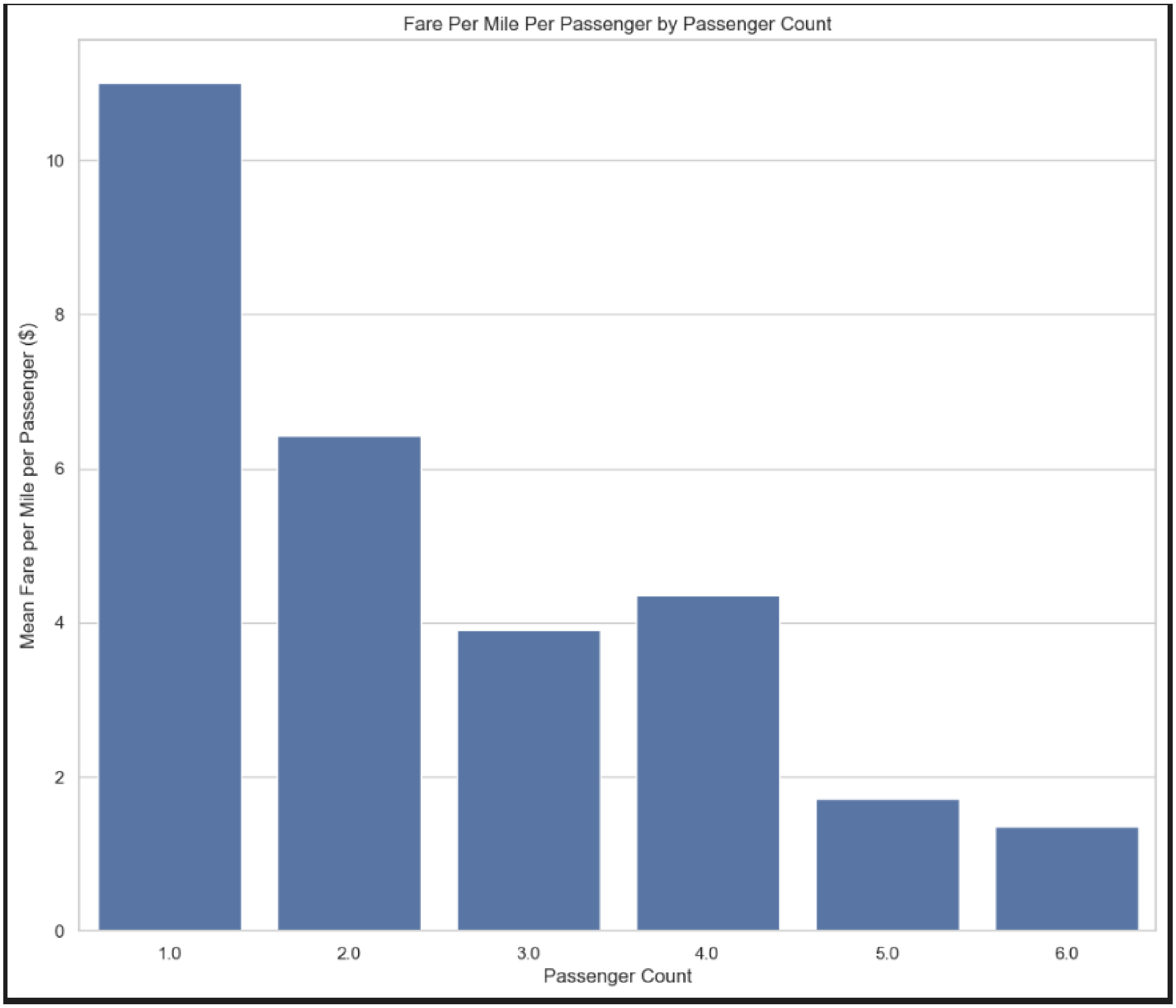
Top 10 Dropoff Ratio:



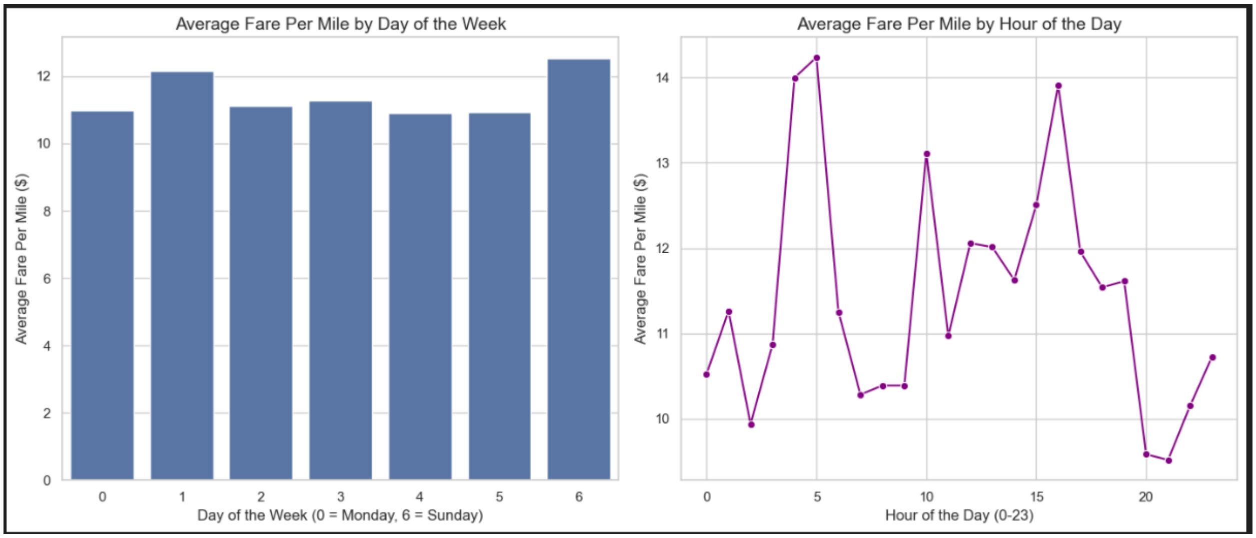
3.2.7. Identify the top zones with high traffic during night hours

3.2.8. Find the revenue share for nighttime and daytime hours

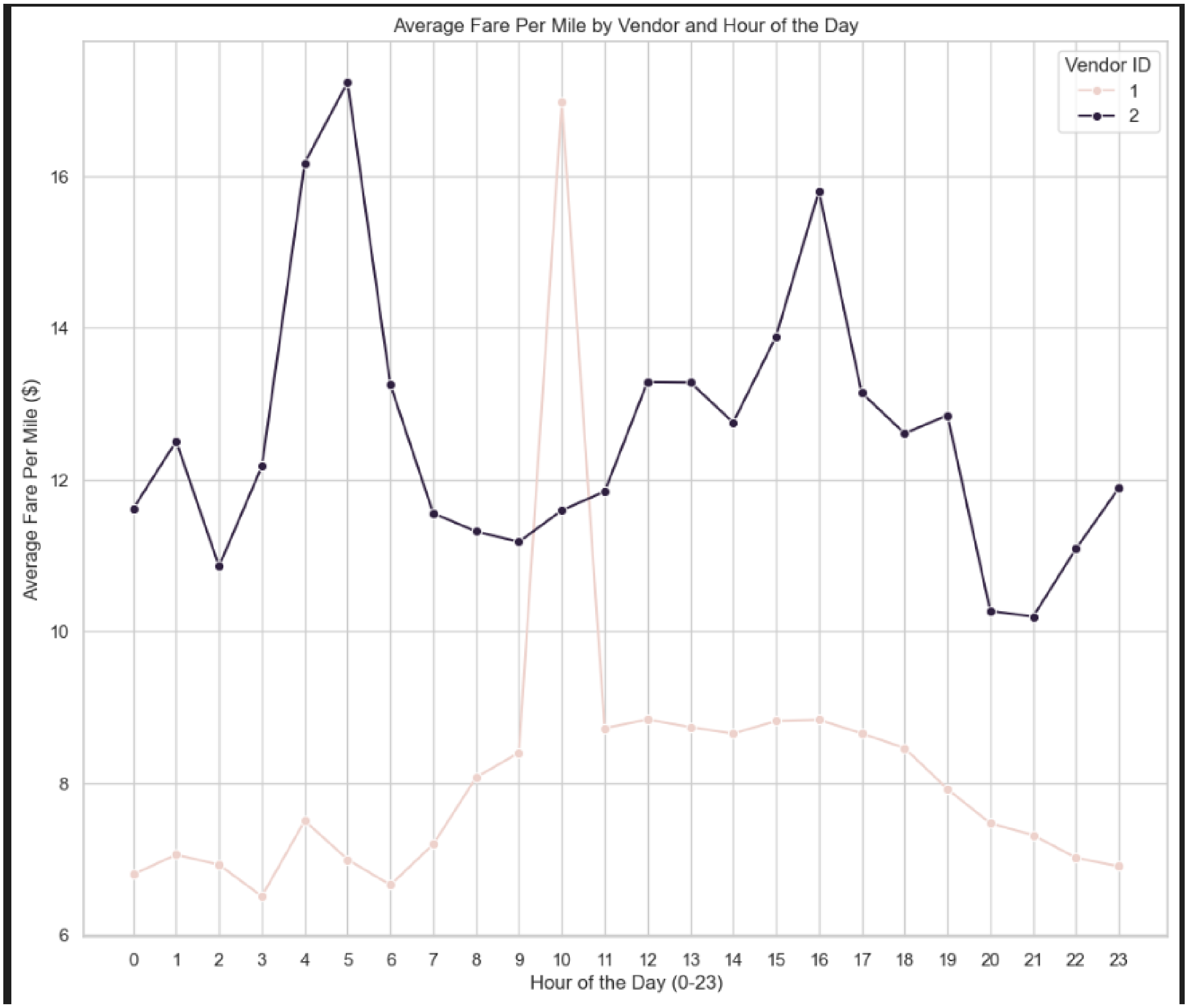
3.2.9. For the different passenger counts, find the average fare per mile per passenger


Fare Per Mile Per Passenger by Passenger Count
Mean Fare per Mile per Passenger ($)
0
2
4
6
8
10
1.0
2.0
3.0
4.0
5.0
6.0
Passenger Count

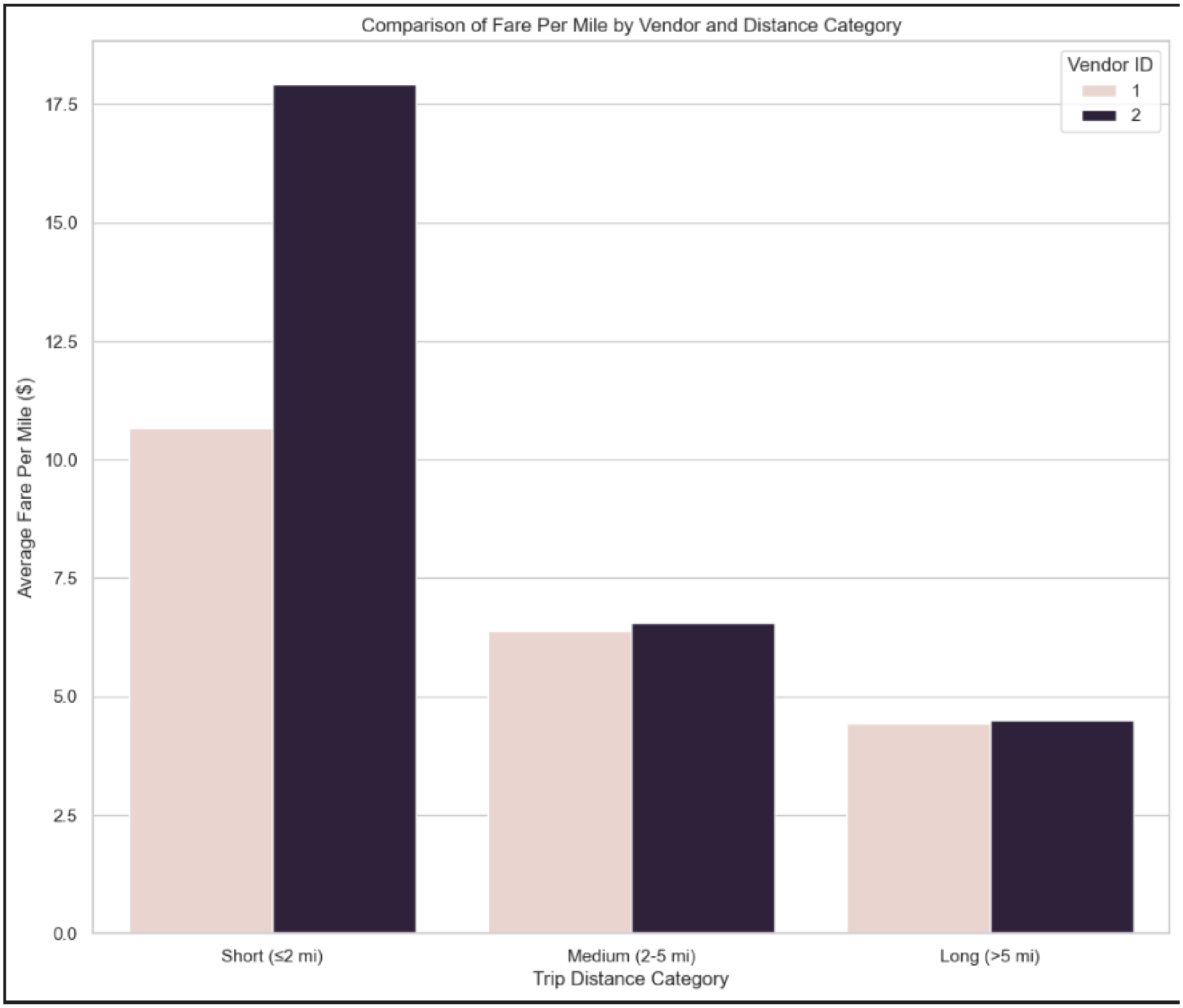
3.2.10. Find the average fare per mile by hours of the day and by days of the week



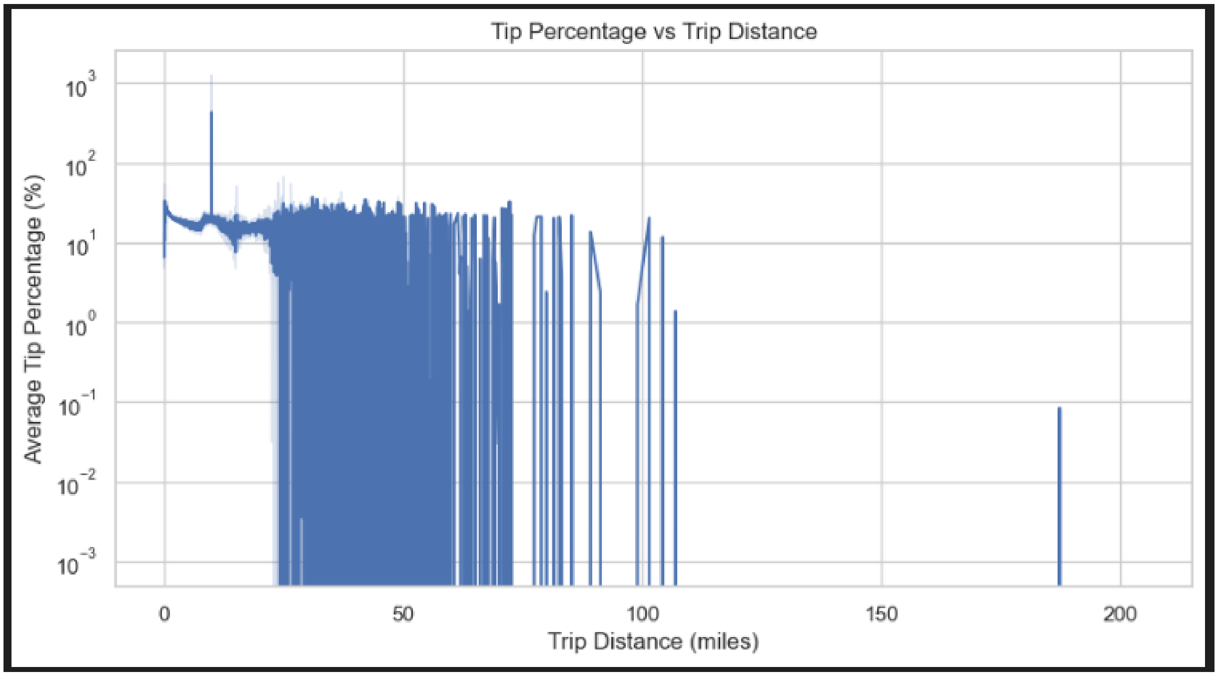
3.2.11. Analyse the average fare per mile for the different vendors

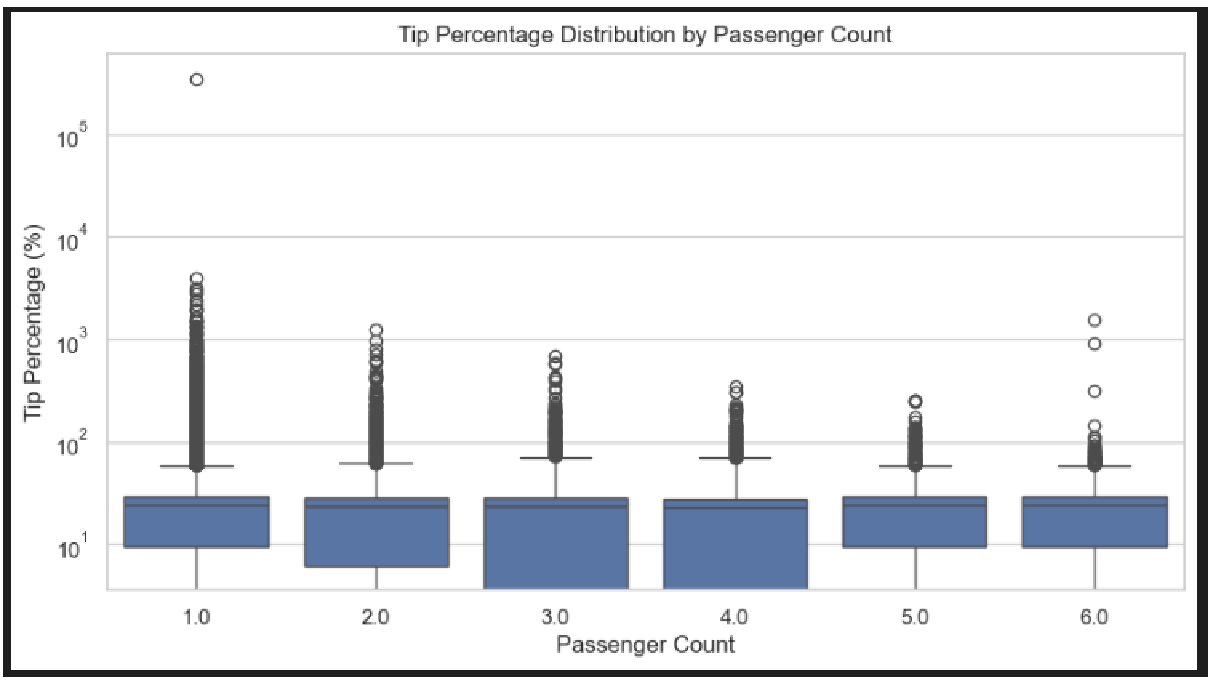

Average Fare Per Mile by Vendor and Hour of the Day
Vendor ID
1
2
Average Fare Per Mile ($)
16
14
12
10
8
6
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour of the Day (0-23)

3.2.12. Compare the fare rates of different vendors in a distance-tiered fashion

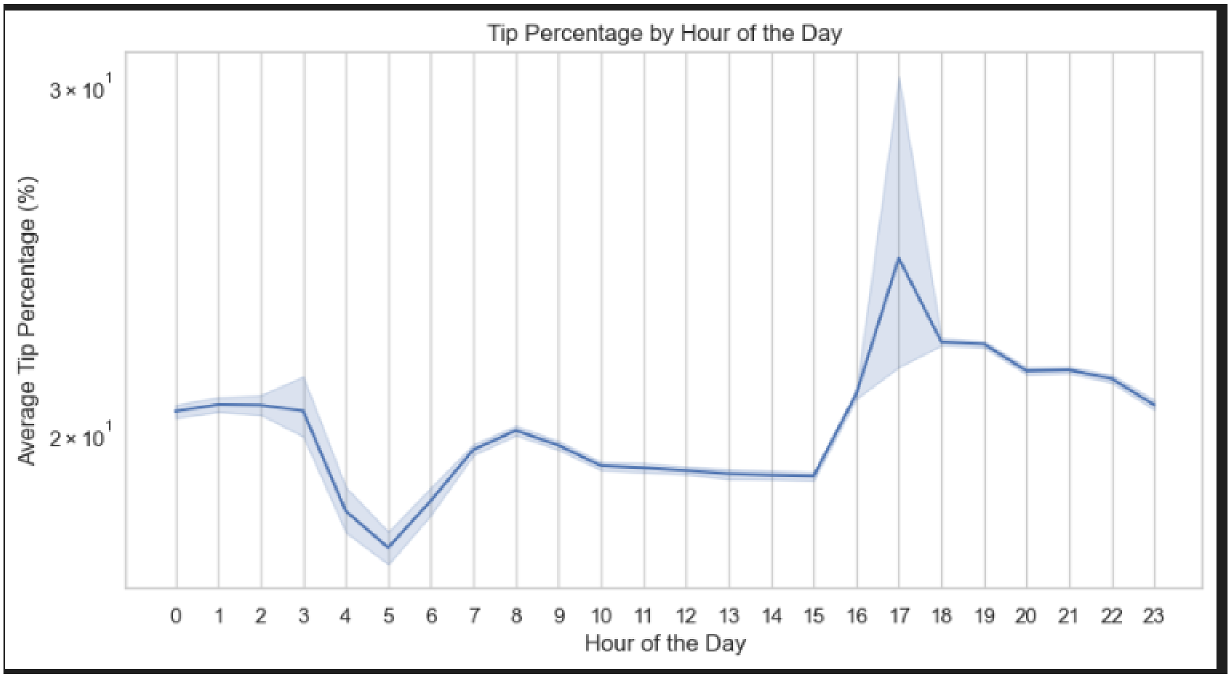


3.2.13. Analyse the tip percentages

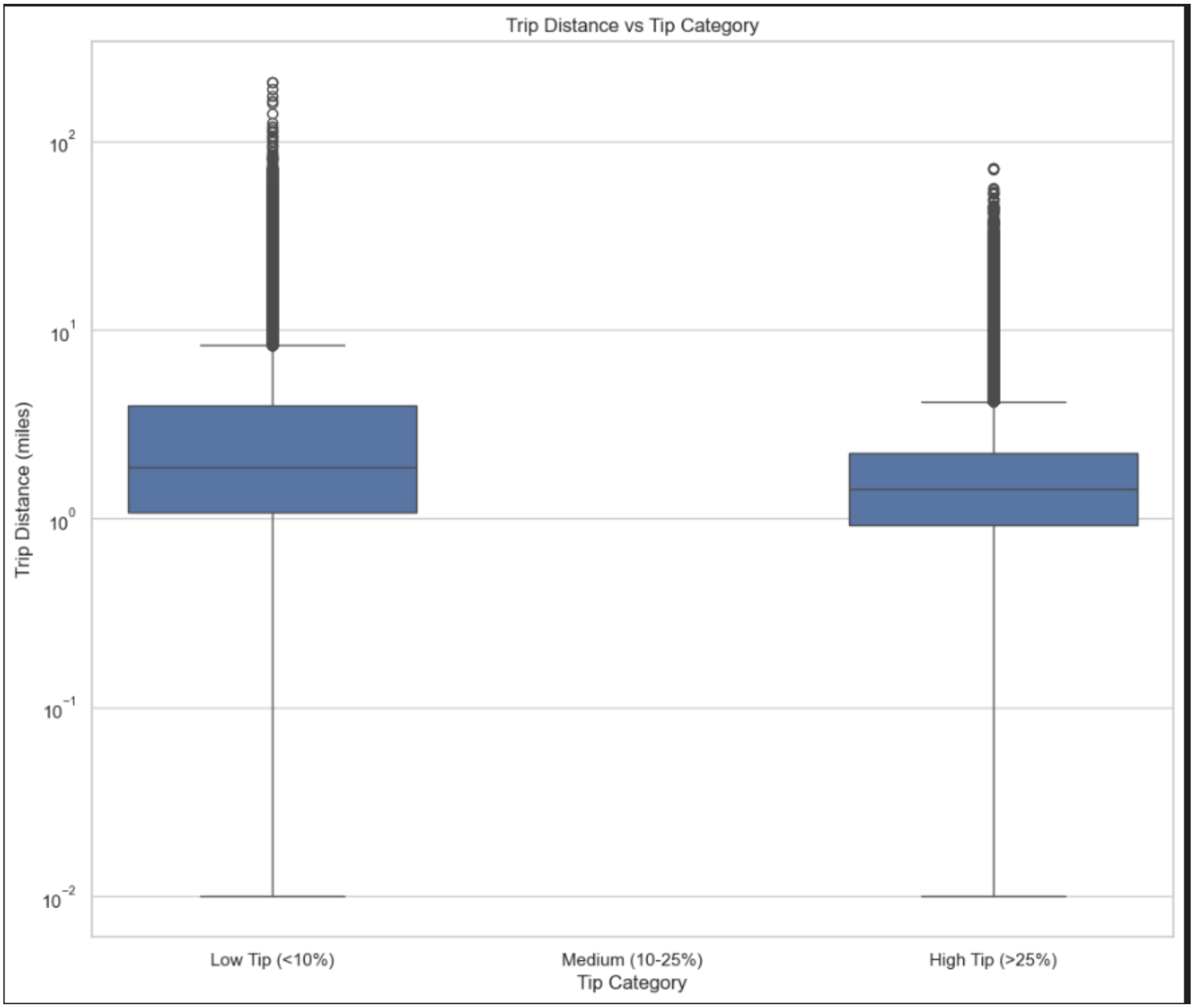


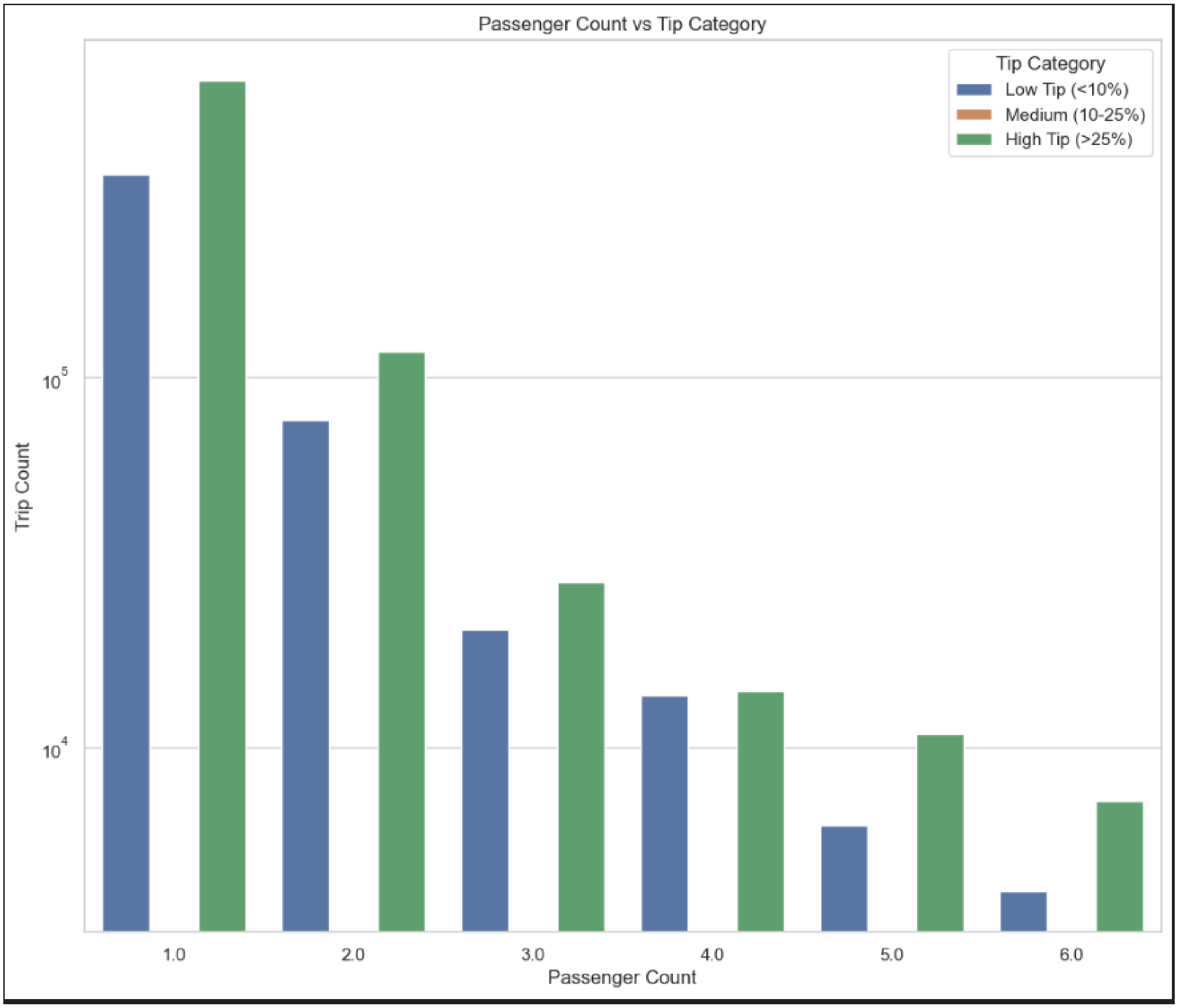


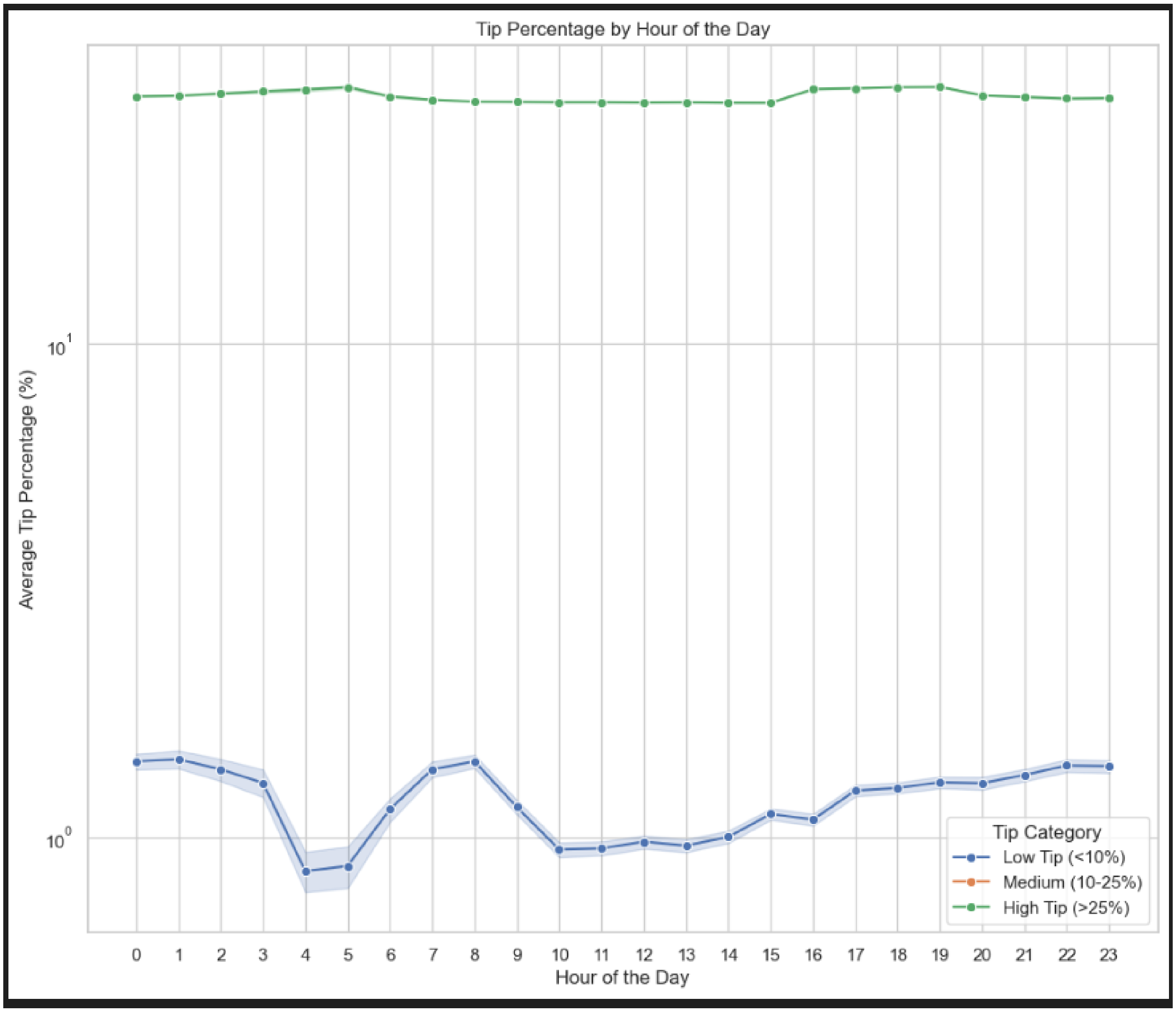
3.2.14.

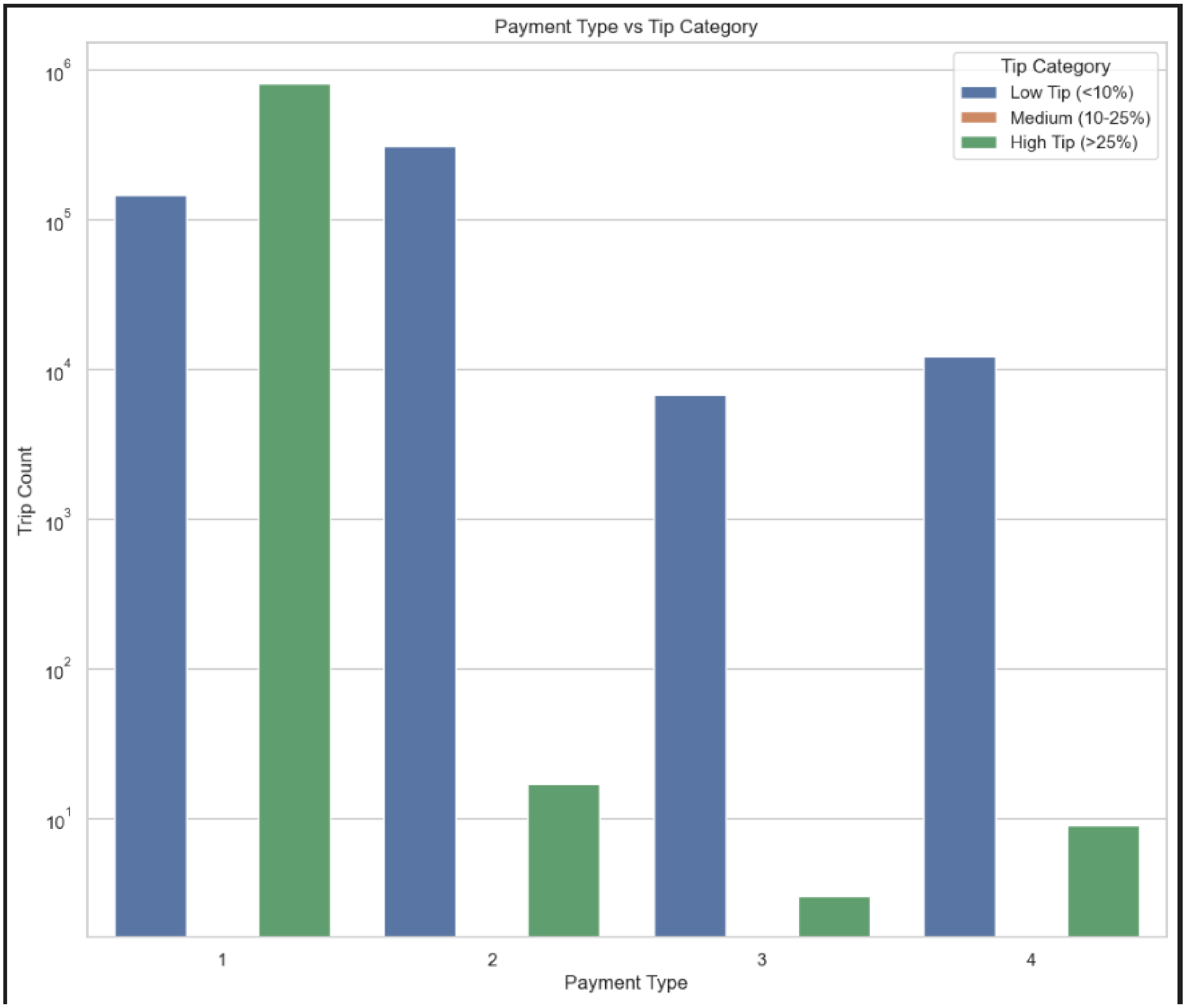


3.2.15. Analyse the trends in passenger count

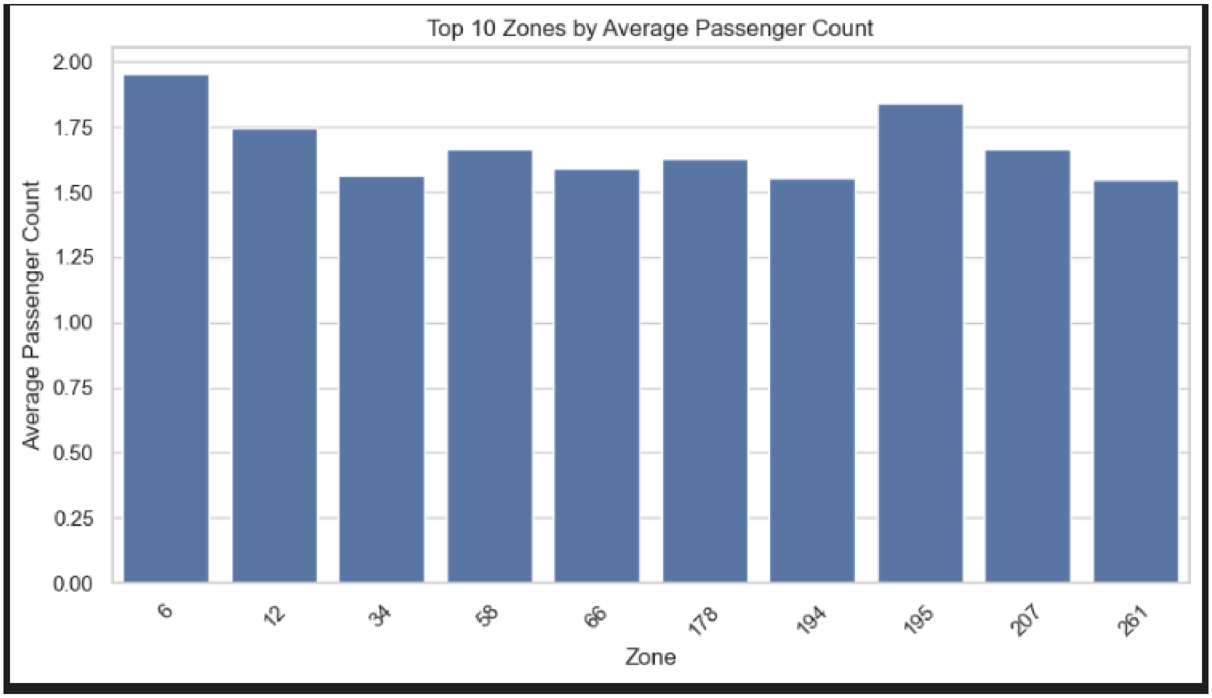

Trip Distance vs Tip Category
Trip Distance (miles)
10^2
10^1
10^0
10^{-1}
10^{-2}
Low Tip (<10%)
Medium (10-25%)
High Tip (>25%)
Tip Category

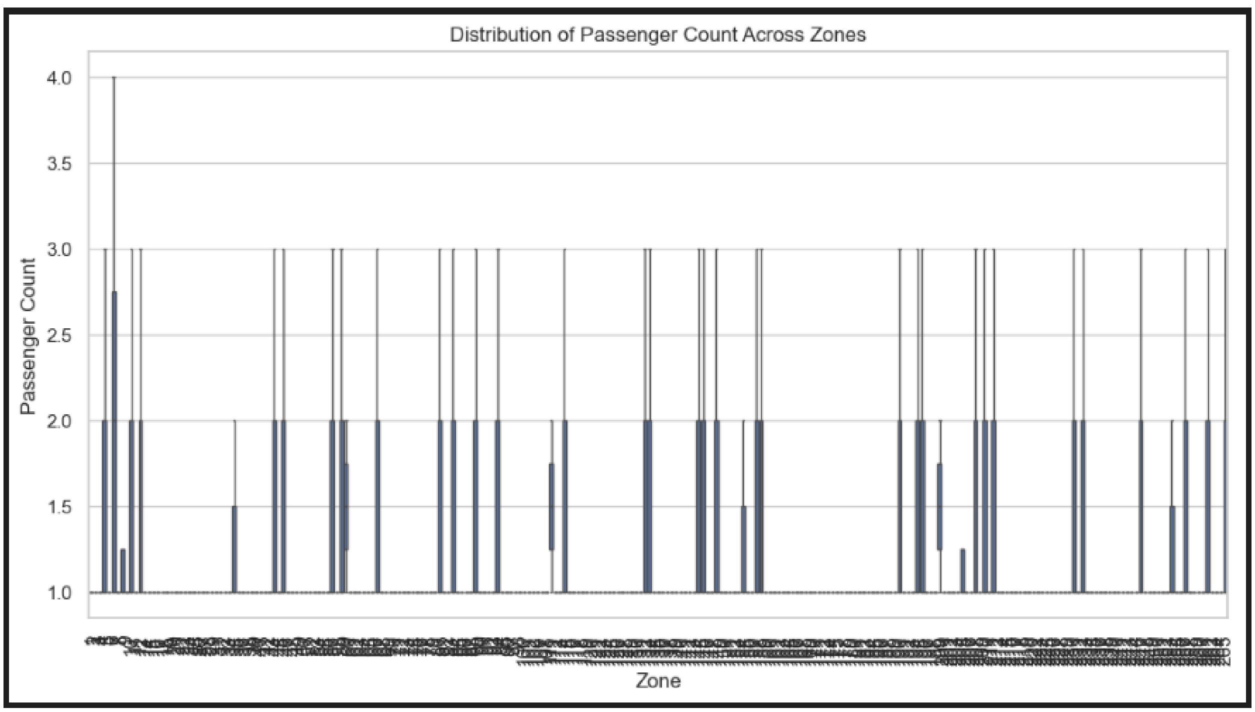

Passenger Count vs Tip Category
Tip Category
Low Tip (<10%)
Medium (10-25%)
High Tip (>25%)
Trip Count
10^5
10^4
1.0
2.0
3.0
4.0
5.0
6.0
Passenger Count


Tip Percentage by Hour of the Day
Average Tip Percentage (%)
10^1
10^0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
Hour of the Day
Tip Category
Low Tip (<10%)
Medium (10-25%)
High Tip (>25%)

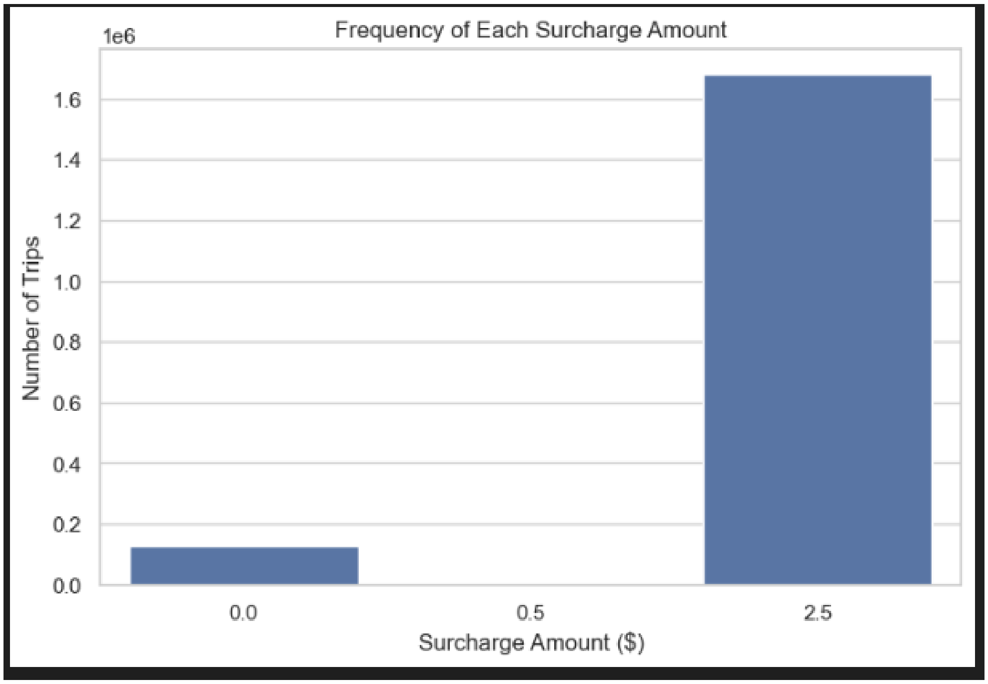


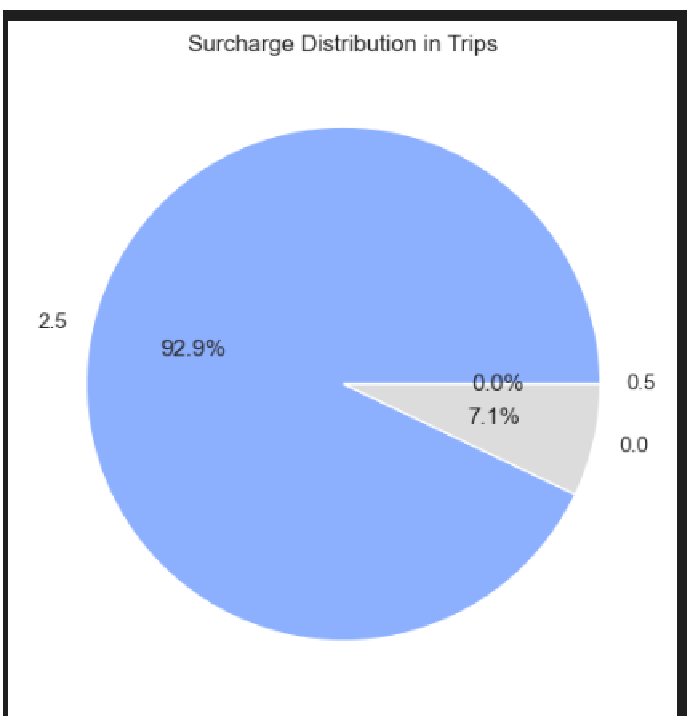
3.2.16. Analyse the variation of passenger counts across zones





3.2.17. Analyse the pickup/dropoff zones or times when extra charges are applied more frequently.


Frequency of Each Surcharge Amount
1e6
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Number of Trips
0.0
0.5
2.5
Surcharge Amount ($)


Surcharge Distribution in Trips
2.5
92.9%
0.0%
7.1%
0.5
0.0

4. Conclusions

4.1. Final Insights and Recommendations

4.1.1. Recommendations to optimize routing and dispatching based on demand patterns and operational inefficiencies.

- Since taxi demand peaks at **6 PM and midweek (Thursday)**, routing algorithms should prioritize these high-traffic areas for cab availability.

- Assign more taxis to **high-revenue quarters (Q2 & Q4)** and areas with high trip volumes.
- Use historical trip data to predict demand surges and dispatch cabs in advance.

4.1.2. Suggestions on strategically positioning cabs across different zones to make best use of insights uncovered by analysing trip trends across time, days and months.

- Position cabs near the top 10 pickup zones identified in the report.
- Since nighttime has slow-moving traffic in certain zones, increase availability in well-lit and high-demand areas.
- Deploy more cabs in Q2 and Q4, as these have the highest revenue.

4.1.3. Propose data-driven adjustments to the pricing strategy to maximize revenue while maintaining competitive rates with other vendors.

- Charge higher rates for longer trips, as there's a positive correlation between distance and fare.
- Differentiate pricing for single vs. multiple passengers to maximize per-mile revenue.