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.

## Data Preparation

* 1. Loading the dataset
     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.

## Data Cleaning

### Fixing Columns

* + 1. **Fix the index**  
         
       I simply check for the index and it looked good, from 0 to 21
    2. **Combine the two airport\_fee columns**

Airport\_fee and airport\_fee, two columns were appearing which I combined to handle redundancy.

### Handling Missing Values

* + 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()
  + 1. **Handling missing values in passenger\_count**

Missing values in Passenger\_Count handled using median.

* + 1. **Handle missing values in RatecodeID**  
       Missing values in RatecodeID were handled using **mode** which were populate maximum appearing values.
    2. **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}")

### Handling Outliers and Standardising Values

* + 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

## Exploratory Data Analysis

### General EDA: Finding Patterns and Trends

* + 1. **Classify variables into categorical and numerical**# Numerical columns (int and float)

**numerical\_cols = df2.select\_dtypes(include=['int64', 'float64']).columns.tolist()**

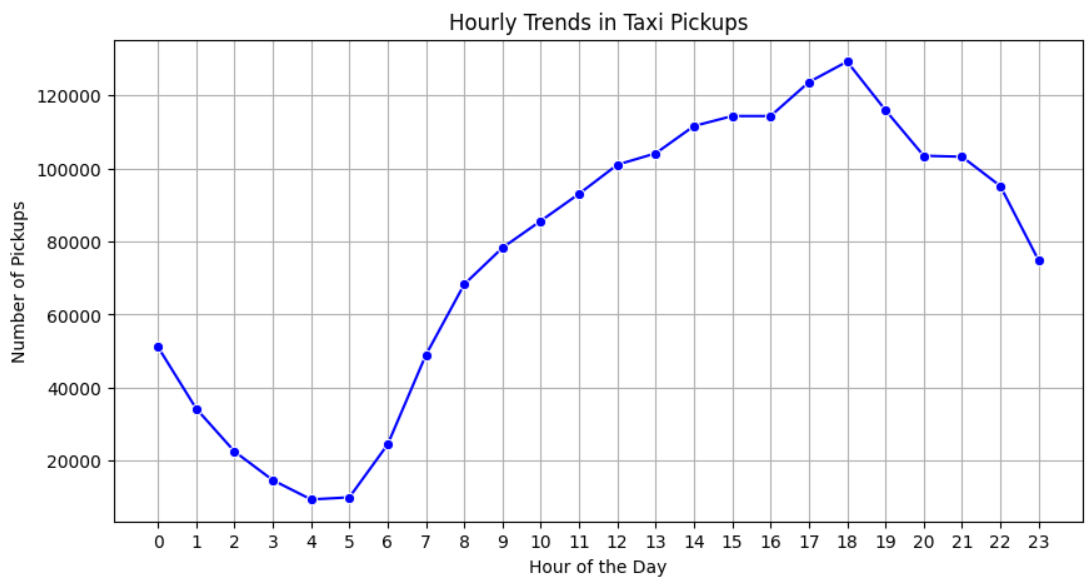
# Categorical columns (object and category)

**categorical\_cols = df2.select\_dtypes(include=['object', 'category']).columns.tolist()**

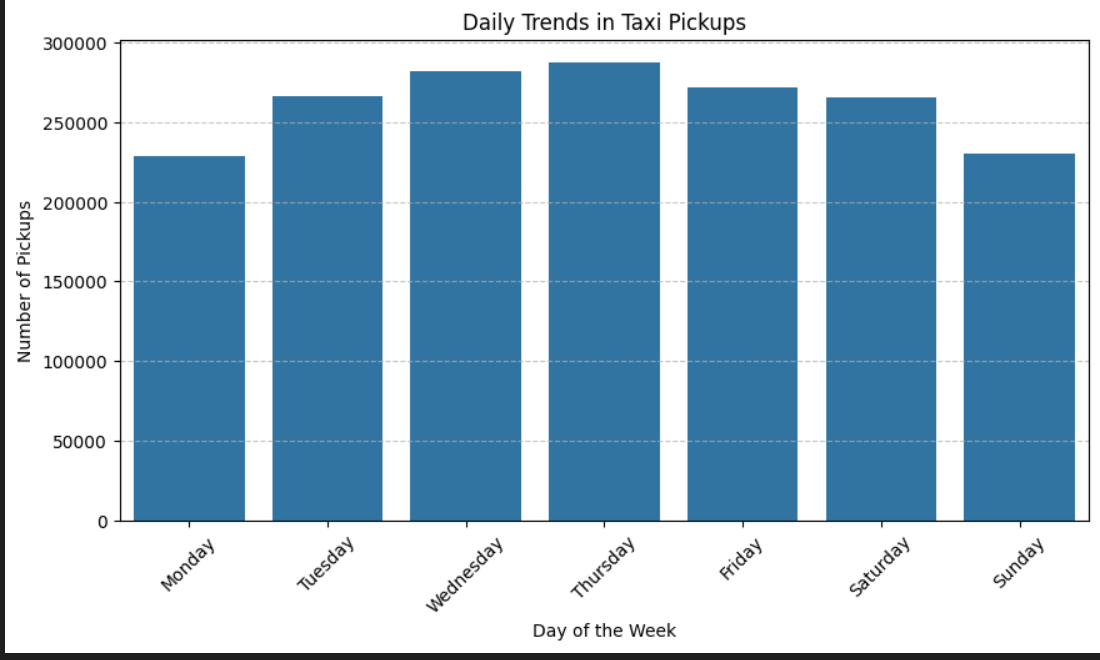
Used above commands to categorised the columns.

* + 1. **Analyse the distribution of taxi pickups by hours, days of the week, and months**

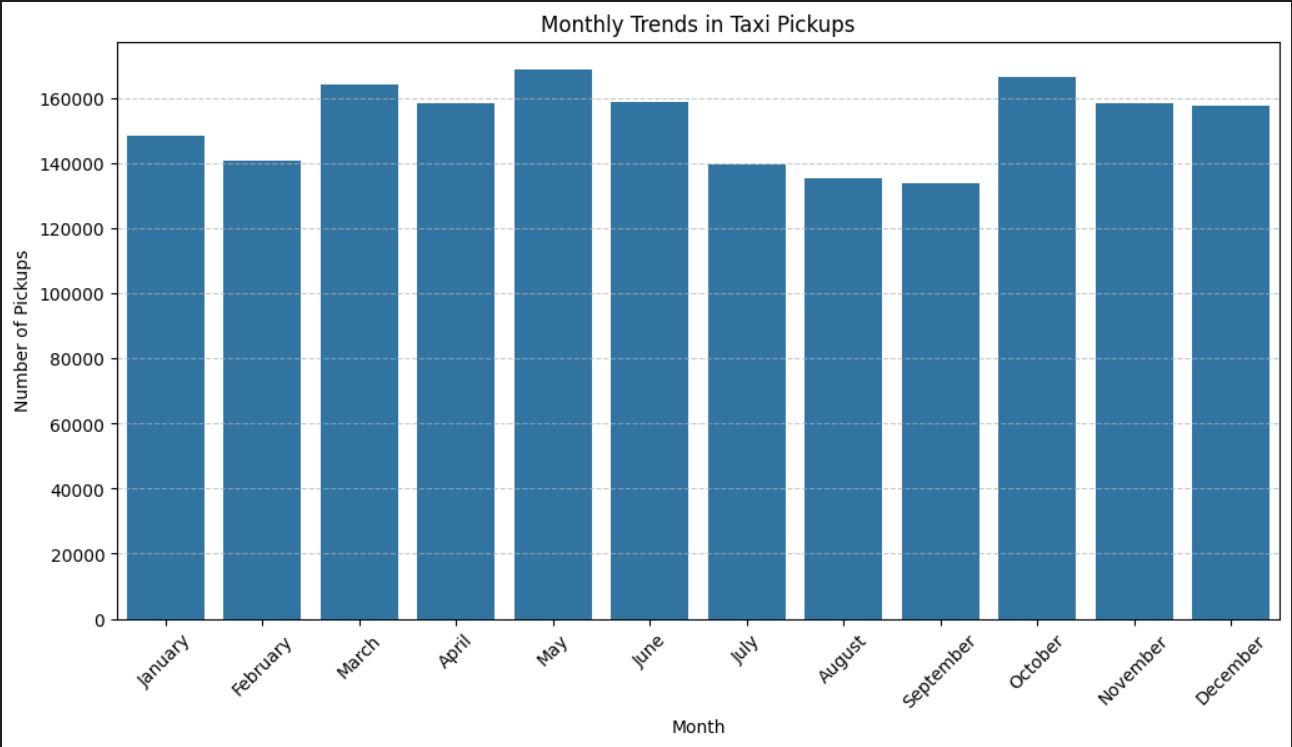
1. **Taxi pickup by hours:** During evening hours taxi pickups increases. Value is highest @6 PM.



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



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

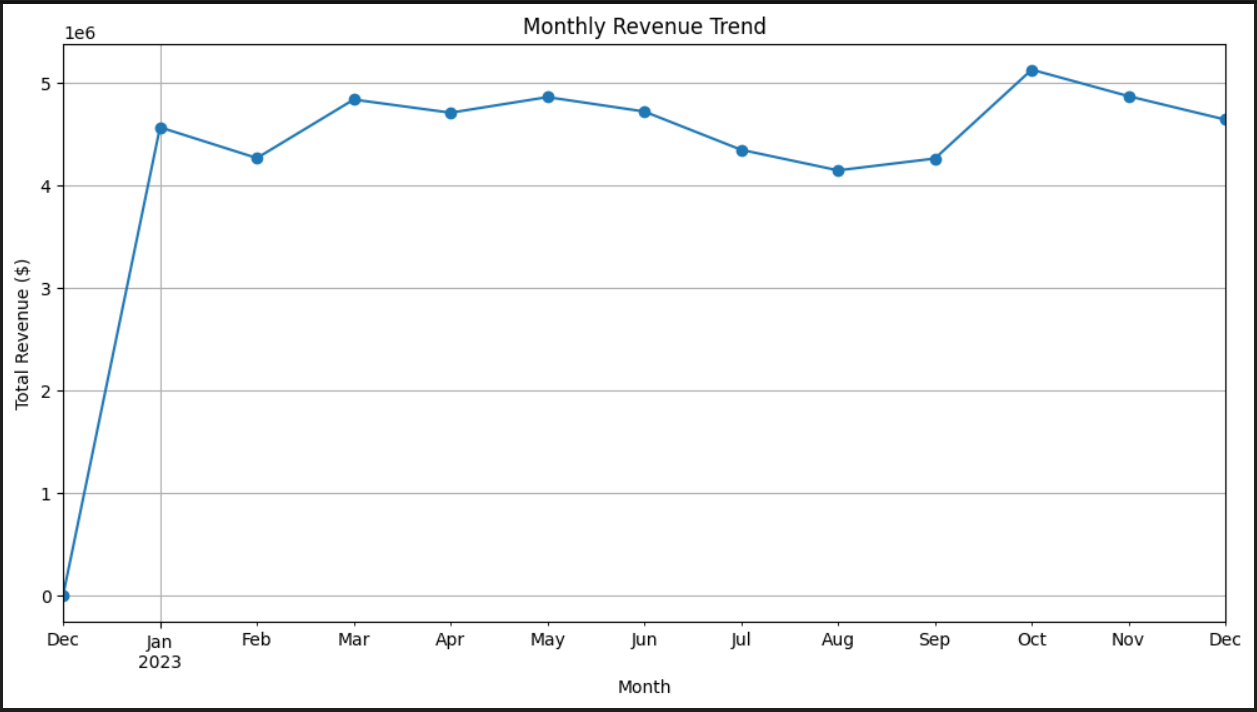


* + 1. **Filter out the zero/negative values in fares, distance and tips**columns\_to\_check = ['fare\_amount', 'tip\_amount', 'total\_amount', 'trip\_distance']

df2[columns\_to\_check].describe()

Used above code to check the negative and zero values

* + 1. **Analyse the monthly revenue trends**

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

* + 1. **Find the proportion of each quarter’s revenue in the yearly revenue  
       year\_quarter**

2022Q4 0.000067

2023Q1 24.694513

2023Q2 25.816265

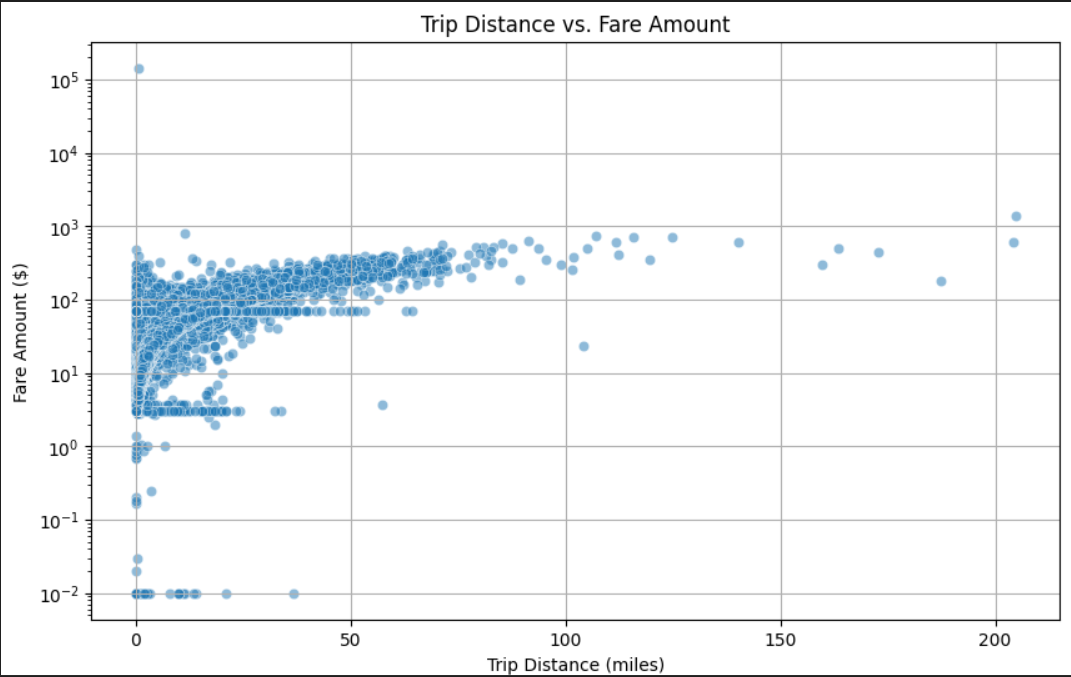
2023Q3 23.040803

2023Q4 26.448352

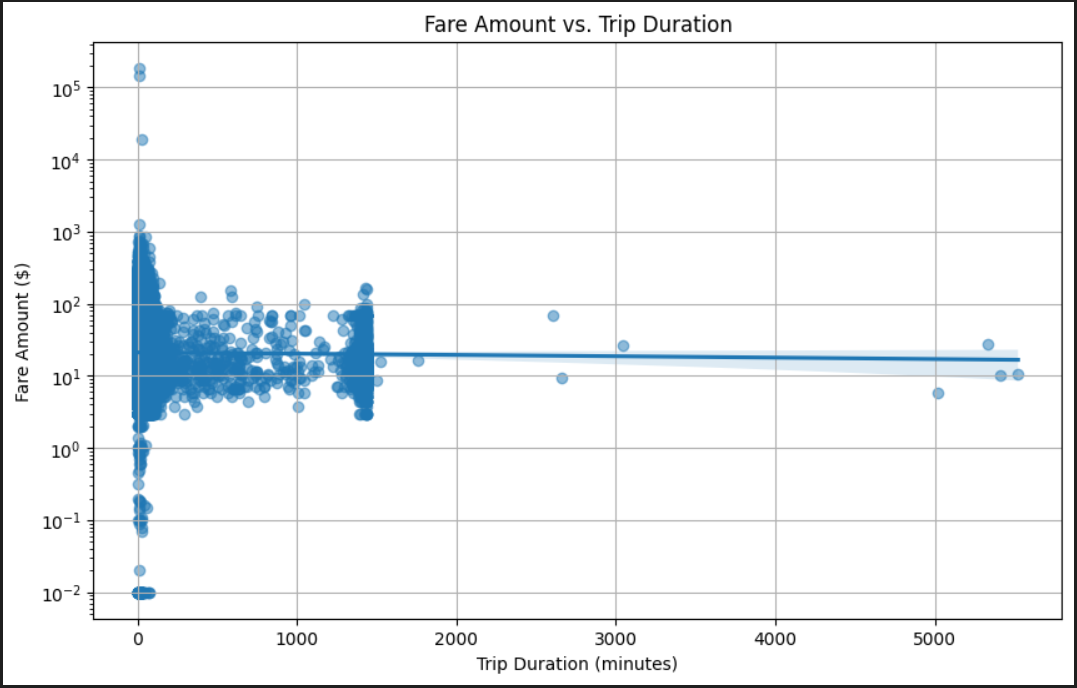
**Revenue is highest in Q4**

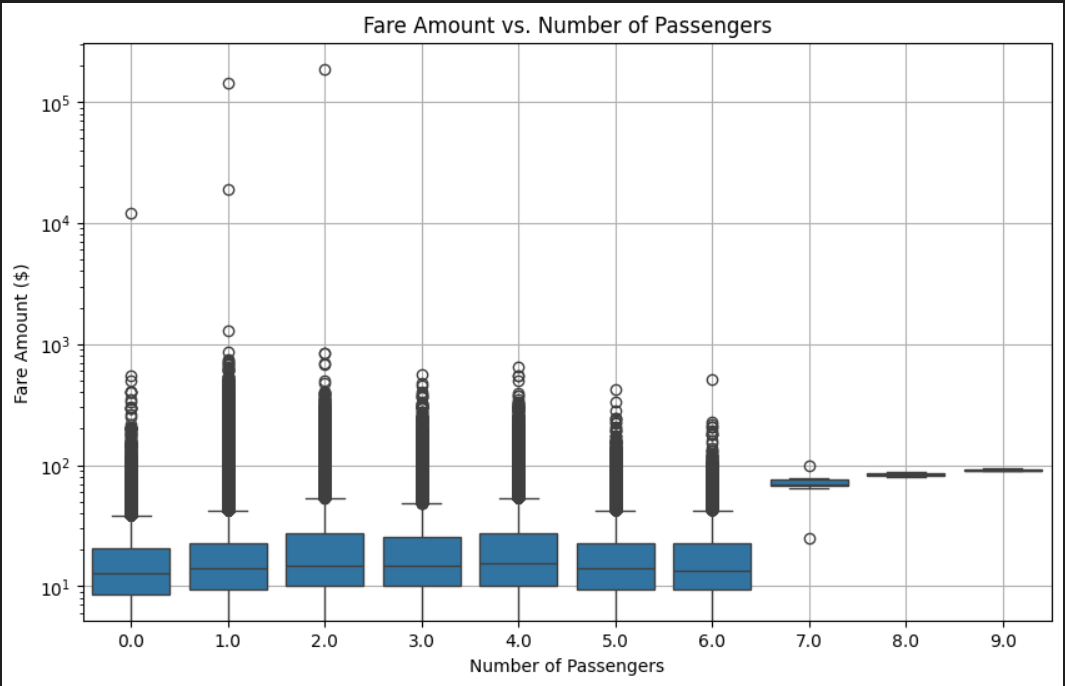
* + 1. **Analyse and visualise the relationship between distance and fare amount**

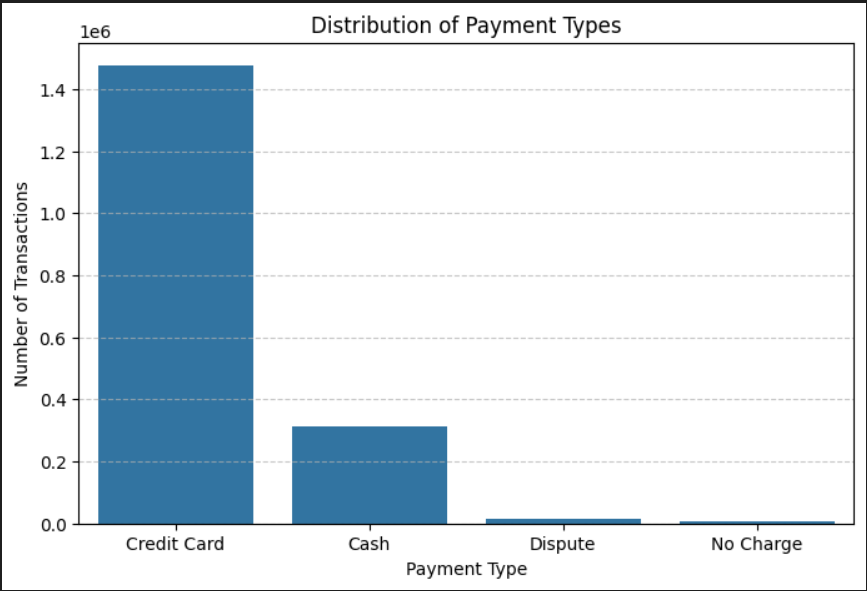
1. A positive correlation between trip distance and fare was observed
2. Correlation between Trip Distance and Fare Amount: 0.1564



* + 1. **Analyse the relationship between fare/tips and trips/passengers**

1. Correlation is negative
2. Correlation between Trip Duration and Fare Amount: -0.0002
3. Correlation between Passenger Count and Fare Amount: 0.0067 and positive

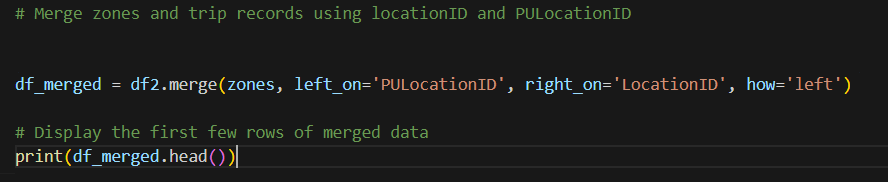
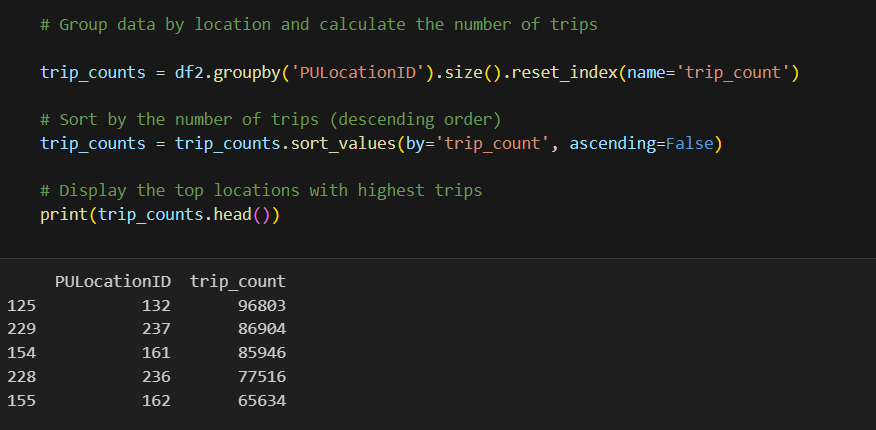
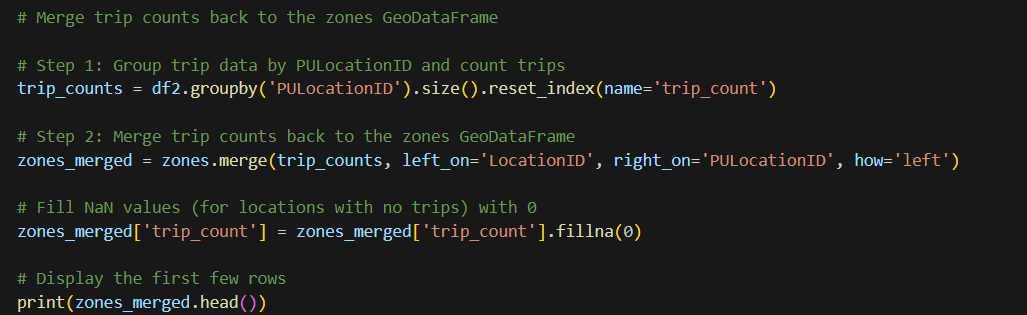
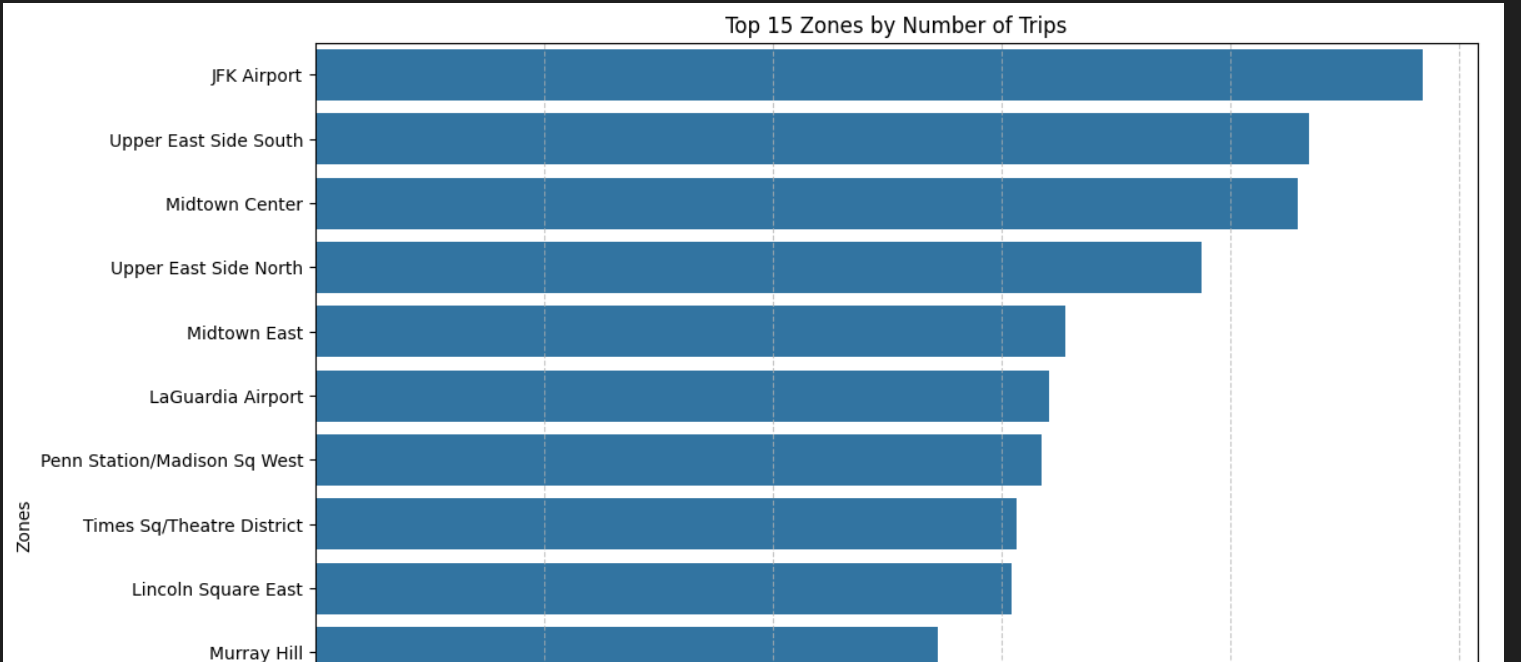


* + 1. **Analyse the distribution of different payment types**
    2. **Load the taxi zones shapefile and display it**import geopandas as gpd

# Read the shapefile using geopandas

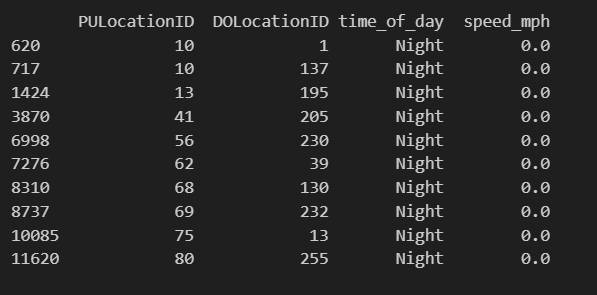
zones =gpd.read\_file(r"C:\Users\Vedan\Downloads\Datasets and Dictionary-NYC\Datasets and Dictionary\taxi\_zones\taxi\_zones.shp")

zones.head()

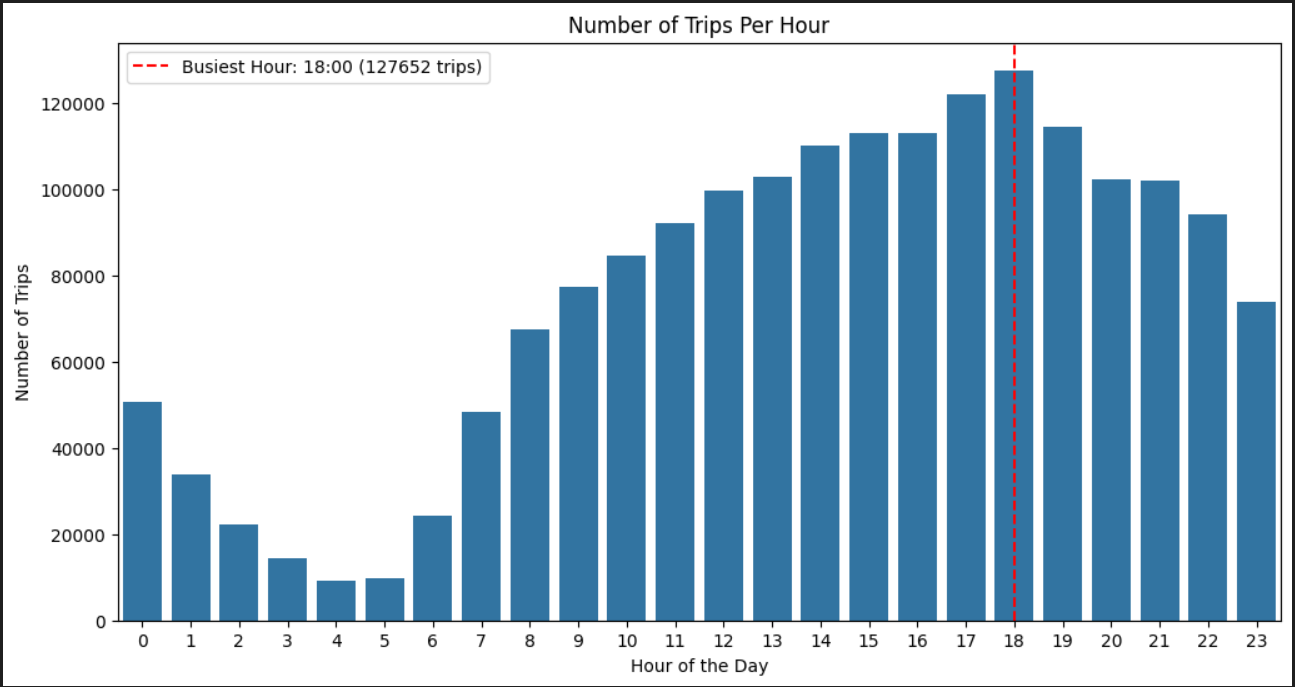
* + 1. **Merge the zone data with trips data**
    2. **Find the number of trips for each zone/location ID**
    3. **Add the number of trips for each zone to the zones dataframe  
       **
    4. **Plot a map of the zones showing number of trips  
         
       **
    5. **Conclude with results**- 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.

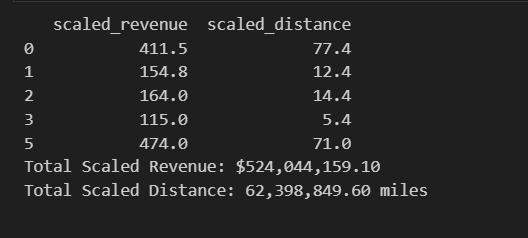
### Detailed EDA: Insights and Strategies

* + 1. **Identify slow routes by comparing average speeds on different routes**

During night time on below PUlocation speed is comparatively slow

* + 1. **Calculate the hourly number of trips and identify the busy hours**6PM is the busiest hour



* + 1. **Scale up the number of trips from above to find the actual number of trips  
       **

**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")

* + 1. **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")

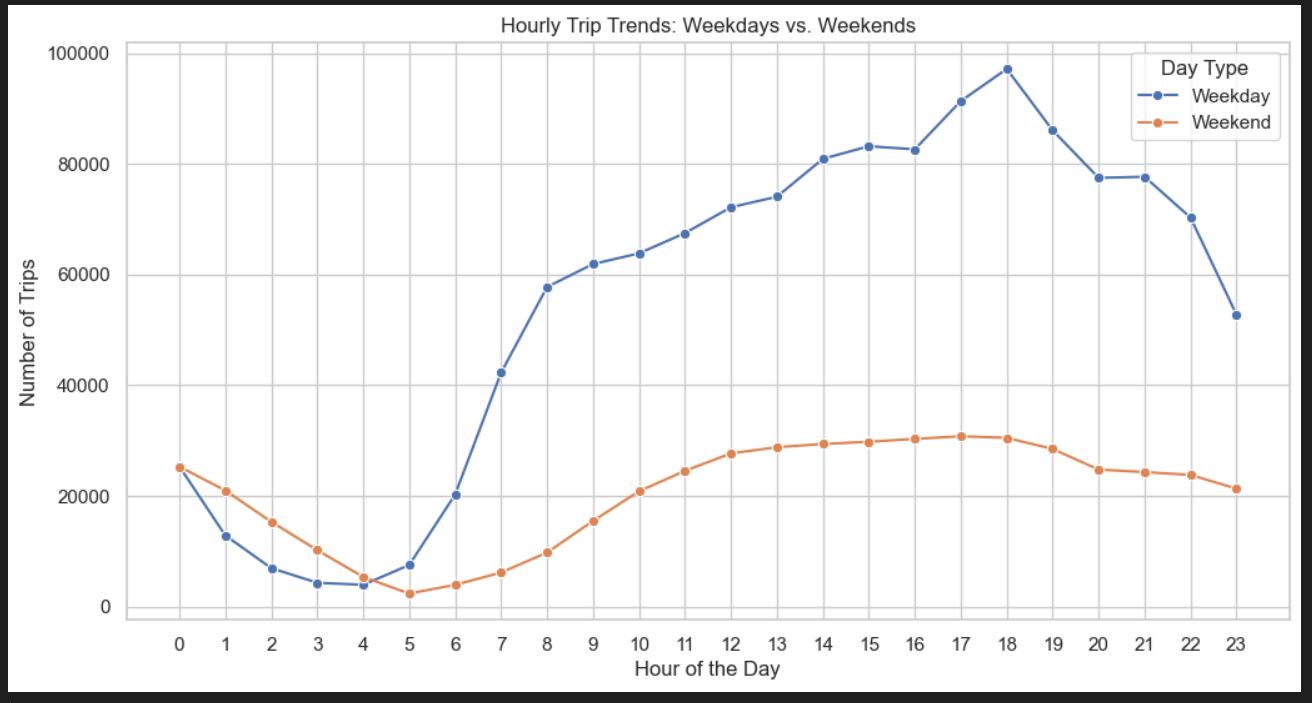
plt.ylabel("Number of Trips")

plt.title("Hourly Trip Trends: Weekdays vs. Weekends")

plt.xticks(range(0, 24))

plt.legend(title="Day Type")

plt.show()

****

* + 1. **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))

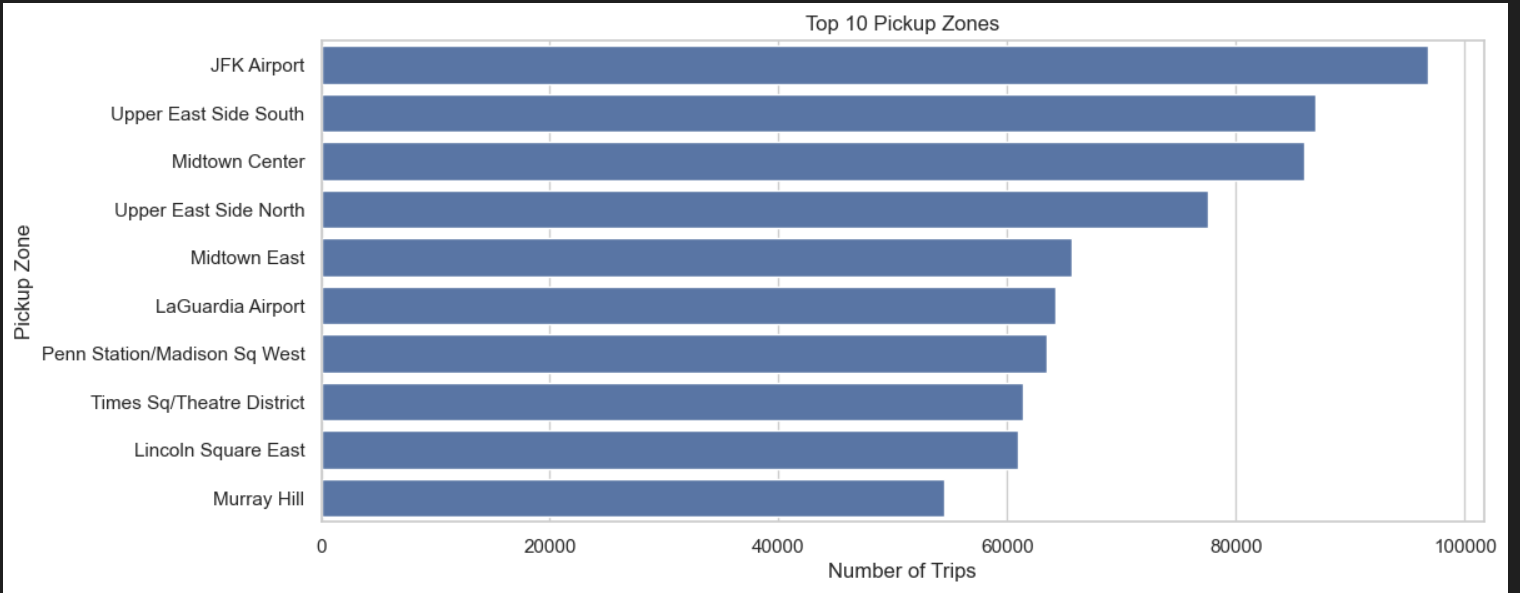
sns.barplot(x=top\_dropoff\_zones['trip\_count'], y=top\_dropoff\_zones['zone'])

plt.xlabel("Number of Trips")

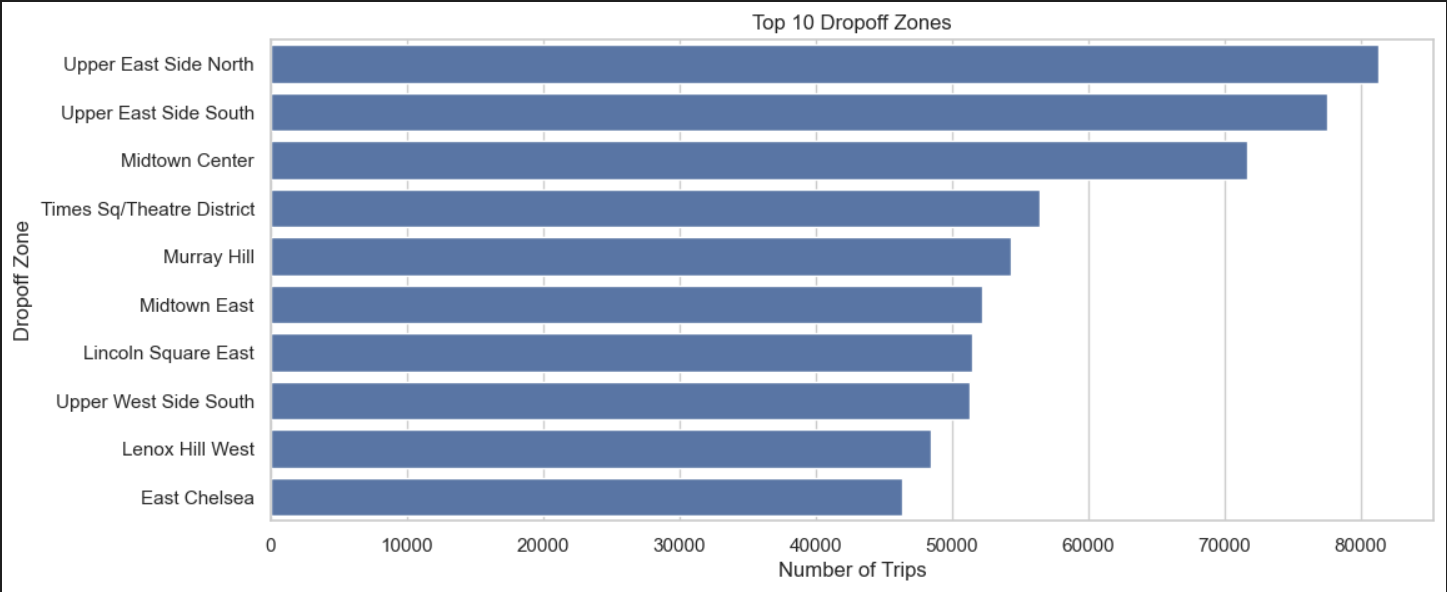
plt.ylabel("Dropoff Zone")

plt.title("Top 10 Dropoff Zones")

plt.show()

**Top 10 Pickup Zones:**

**Top 10 Dropoff Zones:**



* + 1. **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))

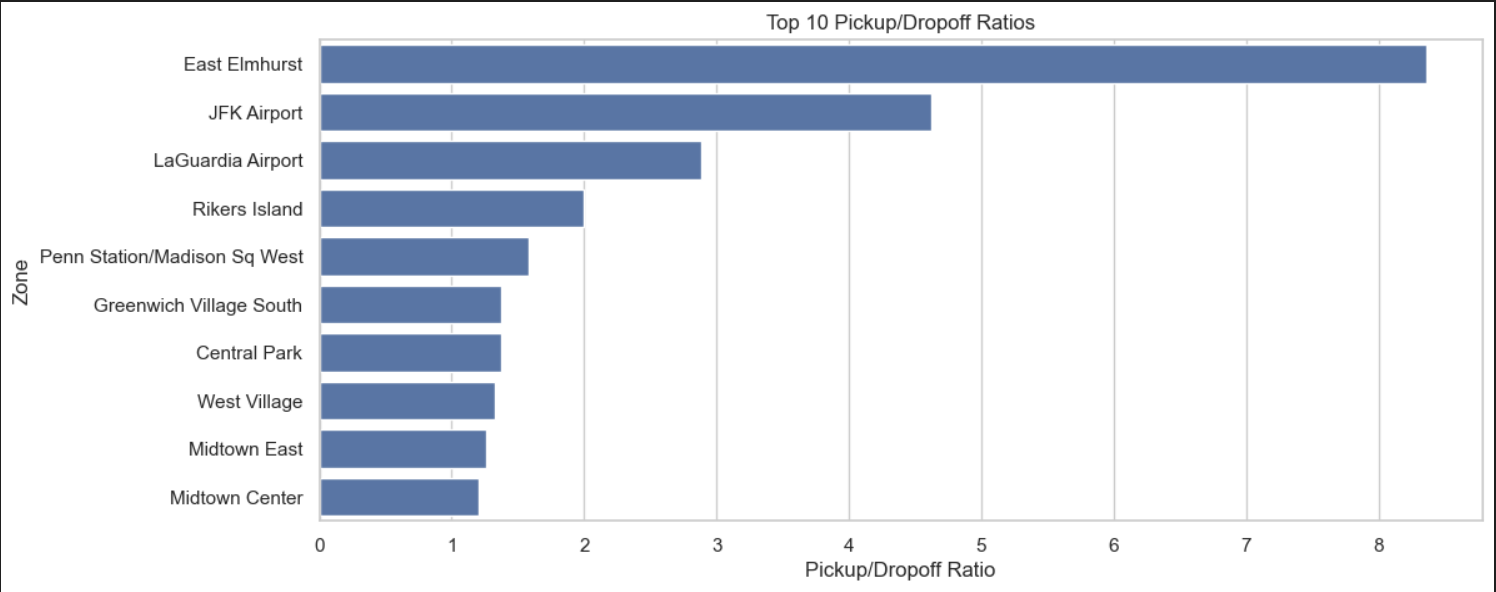
sns.barplot(x=top\_10\_ratios['pickup\_dropoff\_ratio'], y=top\_10\_ratios['zone'])

plt.xlabel("Pickup/Dropoff Ratio")

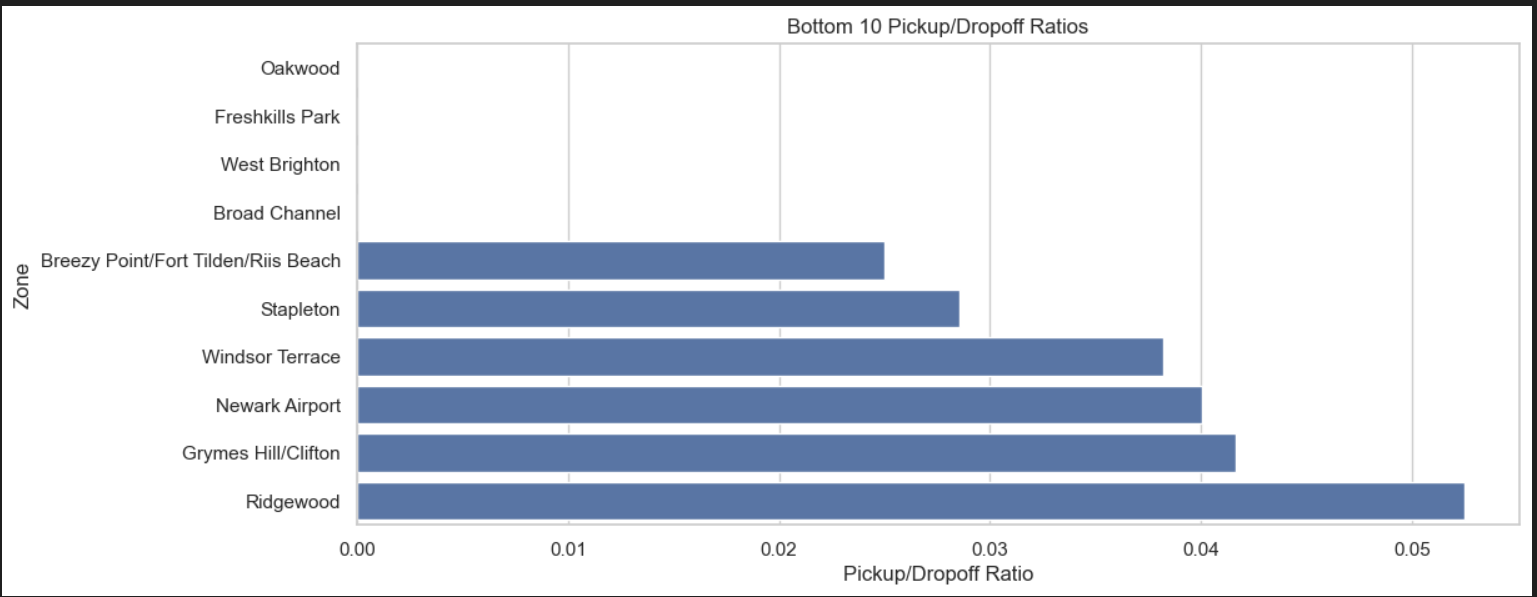
plt.ylabel("Zone")

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

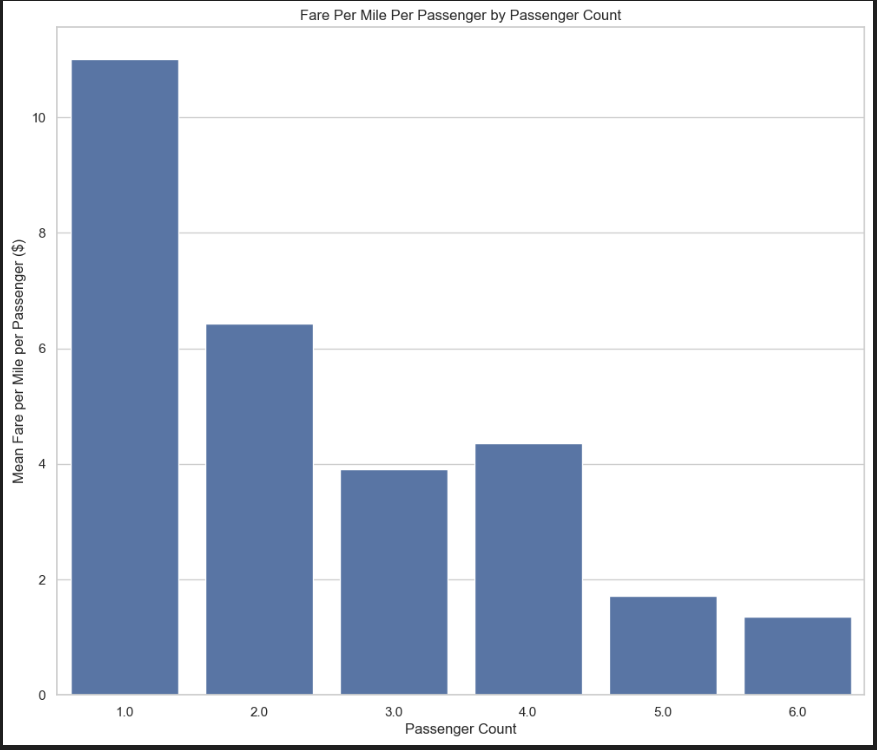
plt.show()

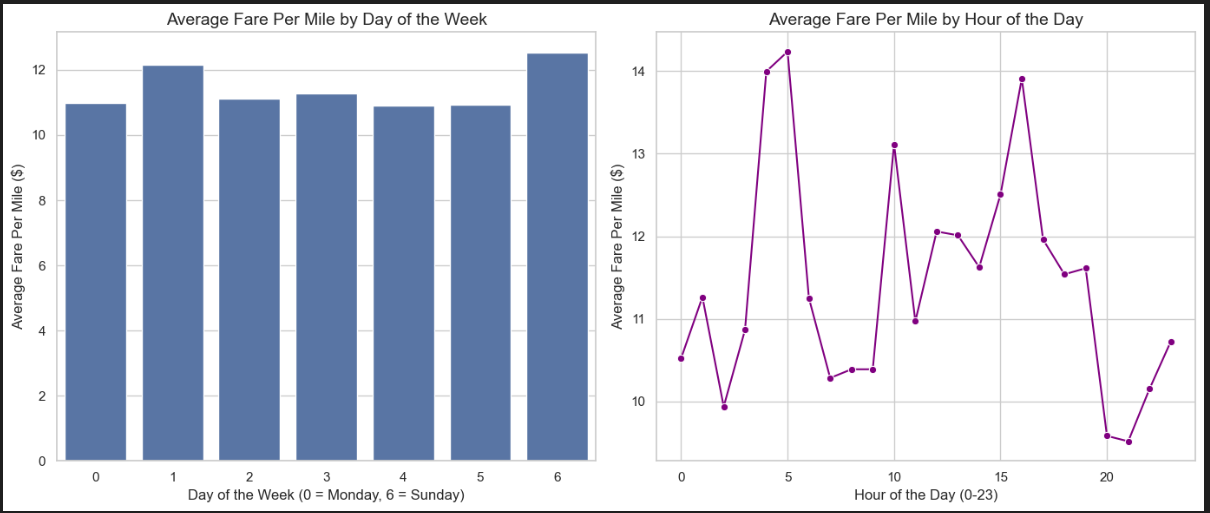
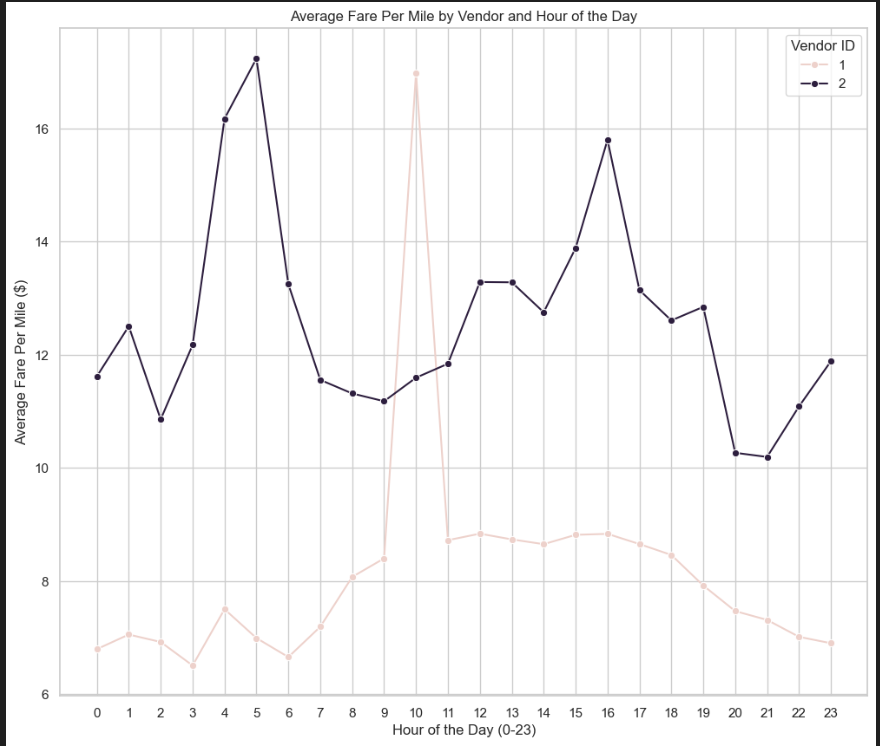
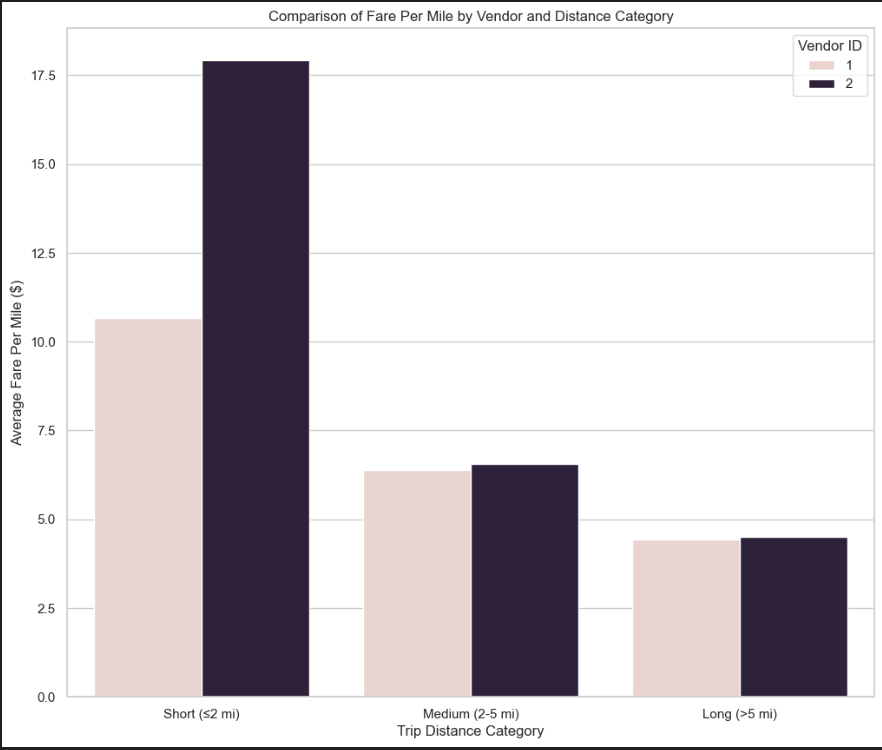
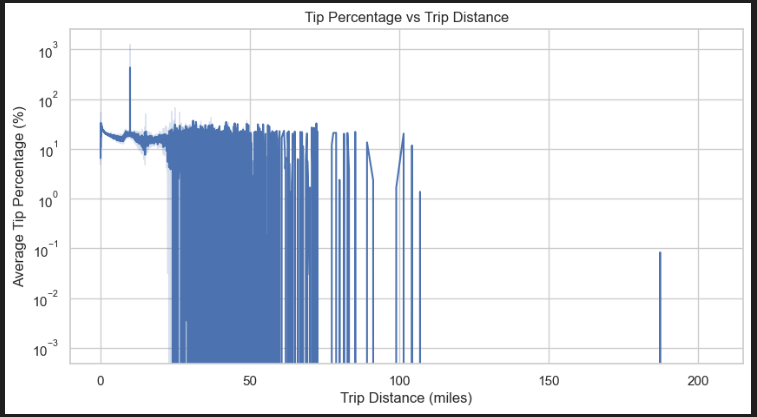
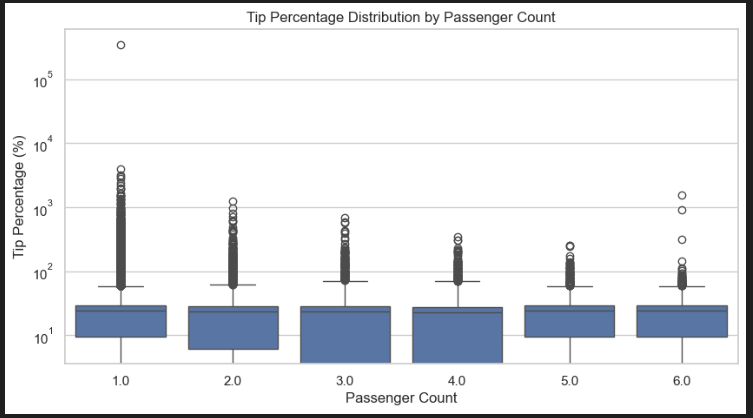
**Top 10 Pickup Ratio:**

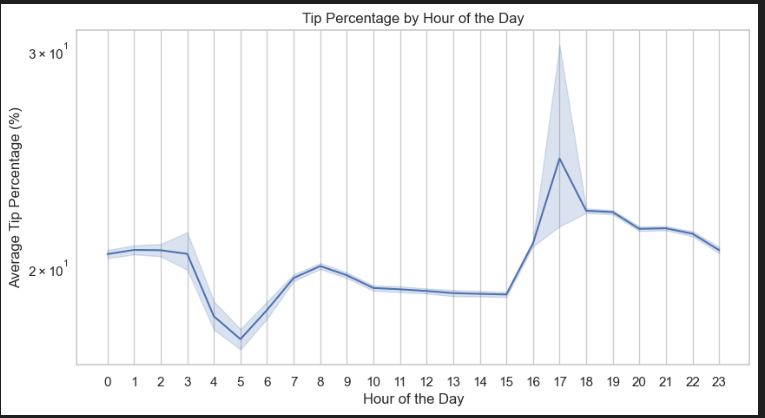
**Top 10 Dropoff Ratio:**



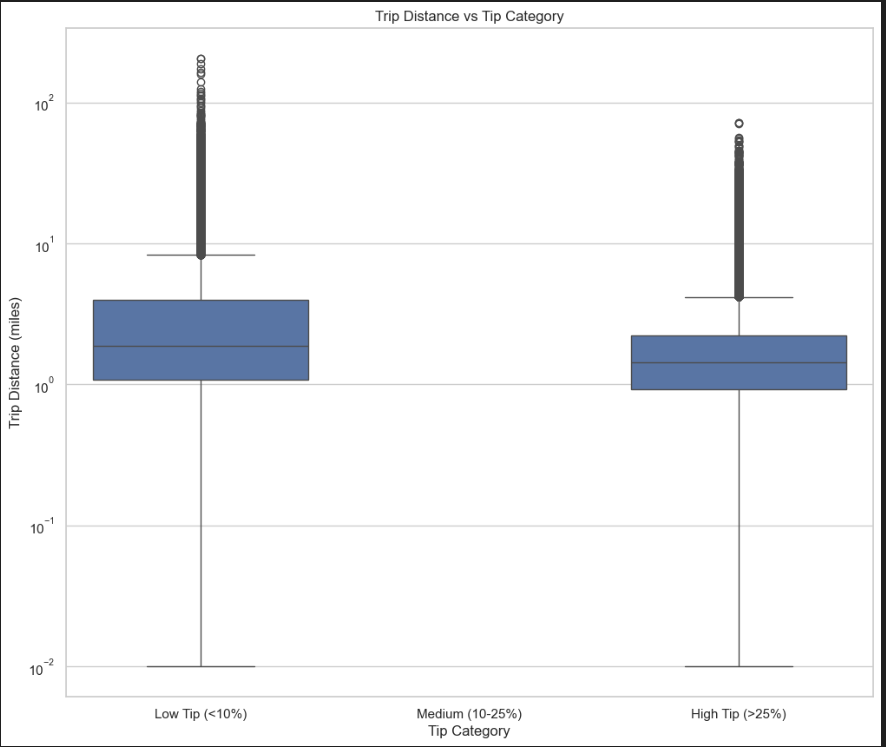
* + 1. **Identify the top zones with high traffic during night hours**
    2. **Find the revenue share for nighttime and daytime hours**
    3. **For the different passenger counts, find the average fare per mile per passenger**

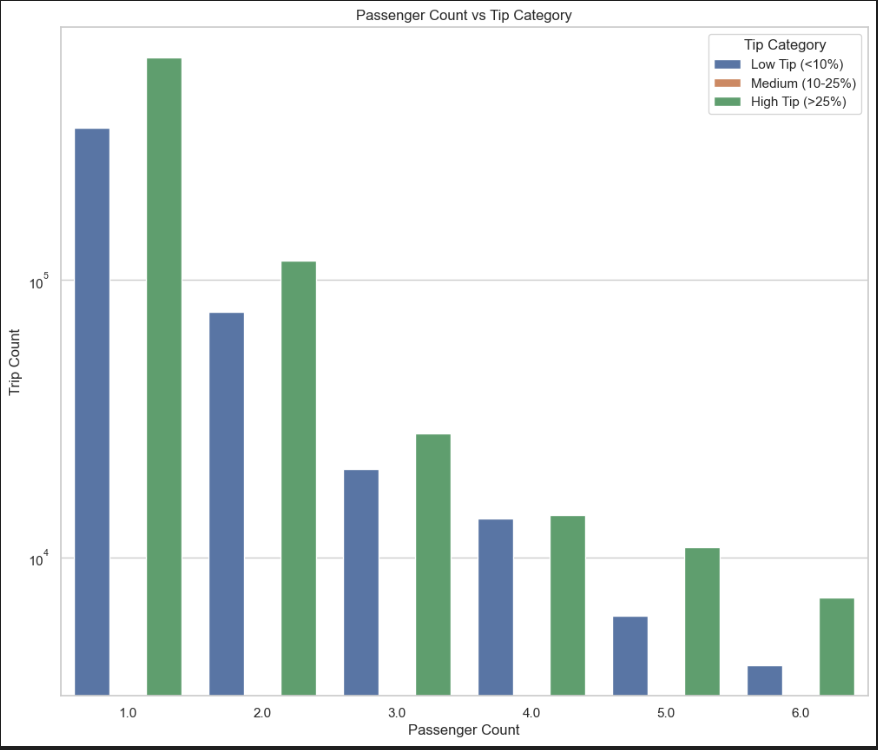
****

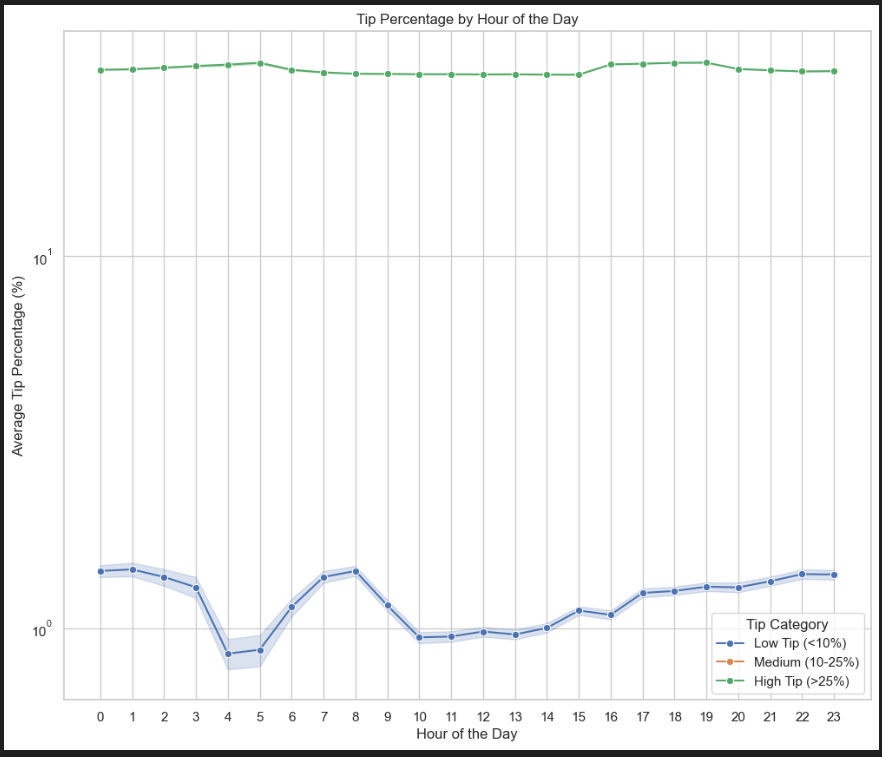
* + 1. **Find the average fare per mile by hours of the day and by days of the week**
    2. **Analyse the average fare per mile for the different vendors  
         
       **
    3. **Compare the fare rates of different vendors in a distance-tiered fashion**
    4. **Analyse the tip percentages**
    5. 

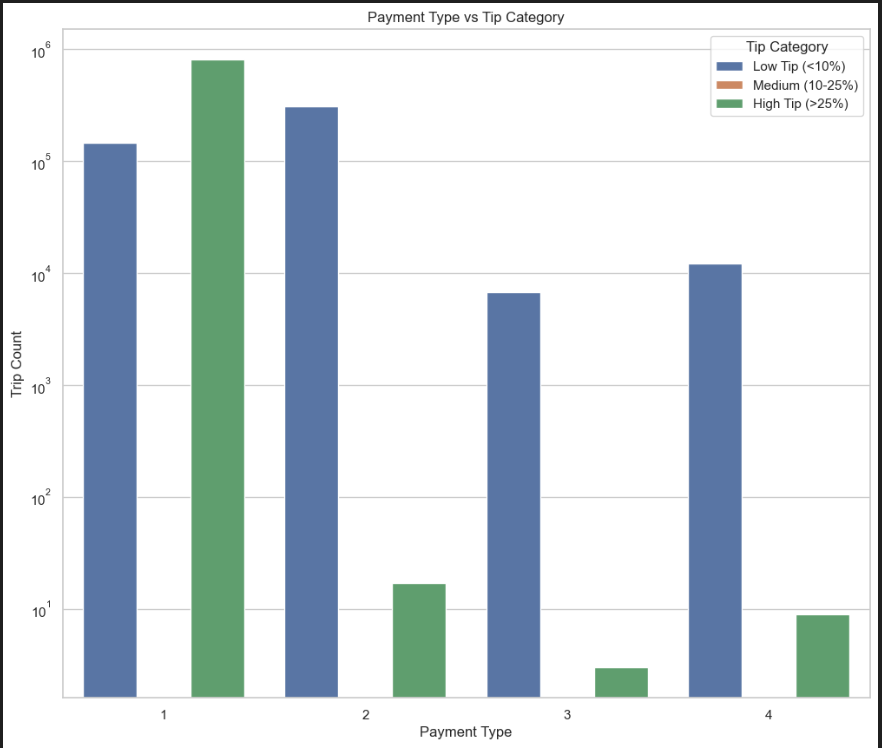


* + 1. **Analyse the trends in passenger count**

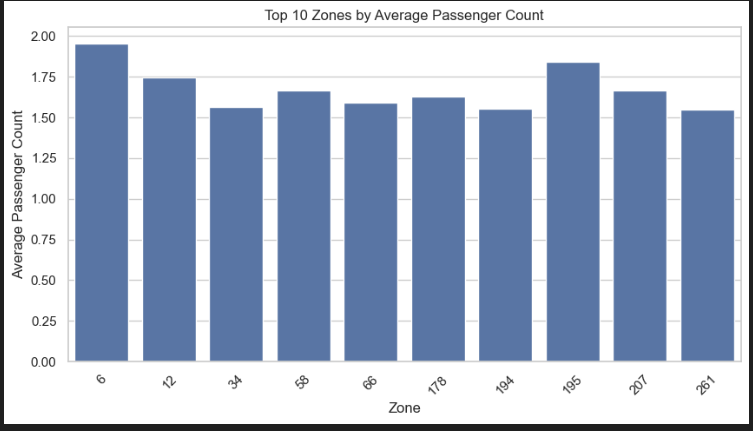


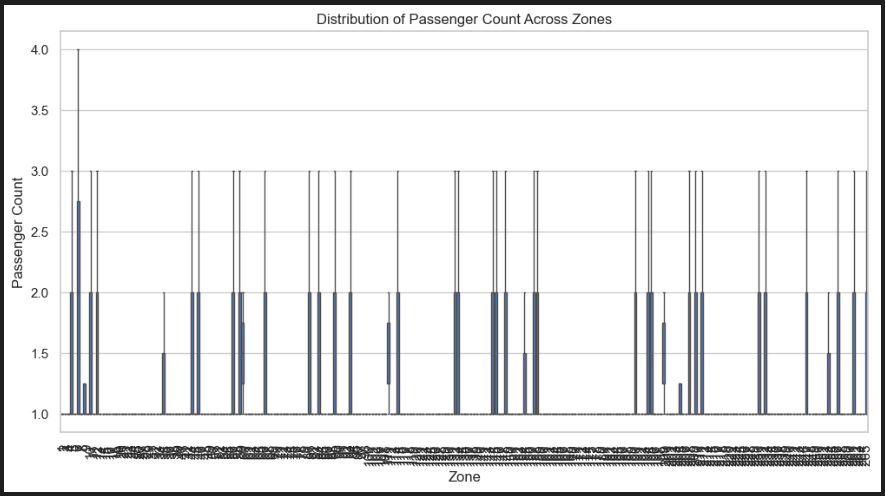


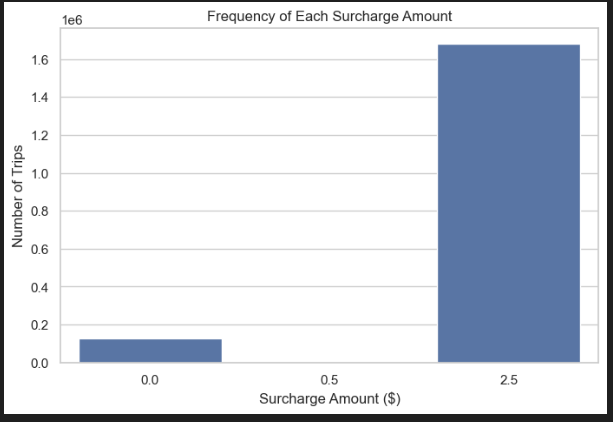


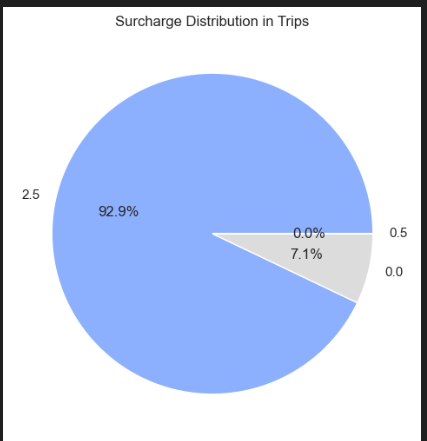


* + 1. **Analyse the variation of passenger counts across zones**





* + 1. **Analyse the pickup/dropoff zones or times when extra charges are applied more frequently.**



## Conclusions

### Final Insights and Recommendations

* + 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.
  + 1. **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.
  + 1. **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.