Report: Optimizing NYC Taxi Operations

Include your visualizations, 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

In this section, I imported the necessary warnings and libraries required for analysis. We have multiple files containing trip records, so I uploaded the folder and combined all the files into a single sample file by extracting 5% of data from each file. Before merging the files into the sample file, I read each one based on the month, day, and hour. After reading the files, I combined the monthly data and appended it to a new DataFrame (the sample file). Once the sample was created, I saved it in both Excel and Parquet formats to use for further analysis.

After loading the sample data, I first verified that all columns were present using the df.info() function, and then confirmed that data was available in each row using the df.head() function.

2. Data Cleaning

2.1. Fixing Columns

2.1.1. Fix the index

After uploading the sample Parquet file, I verified that the data was loaded correctly using the df.info() and df.head() functions. I then examined all data types, checked for missing values, and counted the number of unique values in each column, sorting them in ascending order.

Next, I reset the index using the code df.reset_index(drop=True, inplace=True). During the data review, I identified that the column store_and_fwd_flag was not important for further analysis. This column indicates whether a trip record was temporarily stored in the vehicle's memory before being sent to the vendor (values: 'Y' for yes, 'N' for no).

After this result of column will be:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1996077 entries, 0 to 1996076
Data columns (total 21 columns):
 #   Column                 Dtype
---  ------                 -----
 0   VendorID               int64
 1   tpep_pickup_datetime   datetime64[ns]
 2   tpep_dropoff_datetime  datetime64[ns]
 3   passenger_count        float64
 4   trip_distance          float64
 5   RatecodeID             float64
 6   PULocationID           int64
 7   DOLocationID           int64
 8   payment_type           int64
 9   fare_amount            float64
 10  extra                  float64
 11  mta_tax                float64
 12  tip_amount             float64
 13  tolls_amount           float64
 14  improvement_surcharge  float64
 15  total_amount           float64
 16  congestion_surcharge   float64
 17  airport_fee            float64
 18  date                   object
 19  hour                   int32
 20  Airport_fee            float64
dtypes: datetime64[ns](2), float64(13), int32(1), int64(4), object(1)
memory usage: 312.2+ MB
```

2.1.2. Combine the two airport_fee columns.

While performing the data analysis, I found that there were two columns with the same name. To resolve this, I combined the data from both columns and dropped one of them (Airport_fee). After dropping this column, the dataset includes only the remaining relevant columns shown below.Data columns (total 20 columns):

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1996077 entries, 0 to 1996076
Data columns (total 20 columns):
 #   Column                 Dtype
---  ------                 -----
 0   VendorID               int64
 1   tpep_pickup_datetime   datetime64[ns]
 2   tpep_dropoff_datetime  datetime64[ns]
 3   passenger_count        float64
 4   trip_distance          float64
 5   RatecodeID             float64
 6   PULocationID           int64
 7   DOLocationID           int64
 8   payment_type           int64
 9   fare_amount            float64
 10  extra                  float64
 11  mta_tax                float64
 12  tip_amount             float64
 13  tolls_amount           float64
 14  improvement_surcharge  float64
 15  total_amount           float64
 16  congestion_surcharge   float64
 17  airport_fee            float64
 18  date                   object
 19  hour                   int32
dtypes: datetime64[ns](2), float64(12), int32(1), int64(4), object(1)
memory usage: 297.0+ MB
```

2.1.3 ->

In this section, I checked for negative values in the fare_amount column and removed all rows where the fare was either negative or zero. This is because the fare amount cannot be negative, and a value of zero typically indicates that no travel occurred. I also examined the RatecodeID values associated with negative fares.

During the analysis, I concluded that negative values should not exist in this dataset, as it represents travel data—where values should either be zero or positive. I identified all columns containing negative values and handled them based on the provided documentation. Specifically:

- *For the extra column, I replaced all negative values with 0.*
- *For the mta_tax column, I replaced negative values with 0.5.*
- *For all other numeric columns with negative values, I replaced them with the respective column median.*

After cleaning and replacing all negative values, the dataset is represented as follows:

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	PULocationID	DOLocationID	payment_type	
count	1.996077e+06	1996077	1996077	1.927697e+06	1.996077e+06	1.927697e+06	1.996077e+06	1.996077e+06	1.996077e+06	1
mean	1.733091e+00	2023-07-02 20:01:33.013654016	2023-07-02 20:18:59.015912192	1.369223e+00	3.860656e+00	1.634046e+00	1.652876e+02	1.640619e+02	1.163790e+00	1
min	1.000000e+00	2022-12-31 23:51:30	2022-12-31 23:56:06	0.000000e+00	0.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	0.000000e+00	0
25%	1.000000e+00	2023-04-02 16:11:46	2023-04-02 16:30:22	1.000000e+00	1.050000e+00	1.000000e+00	1.320000e+02	1.140000e+02	1.000000e+00	9
50%	2.000000e+00	2023-06-27 15:47:29	2023-06-27 16:05:23	1.000000e+00	1.790000e+00	1.000000e+00	1.620000e+02	1.620000e+02	1.000000e+00	1
75%	2.000000e+00	2023-10-06 19:39:57	2023-10-06 19:55:13	1.000000e+00	3.400000e+00	1.000000e+00	2.340000e+02	2.340000e+02	1.000000e+00	2
max	6.000000e+00	2023-12-31 23:57:51	2024-01-01 20:50:55	9.000000e+00	1.263605e+05	9.900000e+01	2.650000e+02	2.650000e+02	4.000000e+00	1
std	4.477028e-01	NaN	NaN	8.926300e-01	1.281454e+02	7.388978e+00	6.399840e+01	6.980435e+01	5.082399e-01	1

fare_amount	extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount	congestion_surcharge	airport_fee	hour
1.996077e+06	1.996077e+06	1.996077e+06	1.996077e+06	1.996077e+06	1.996077e+06	1.996077e+06	1.927697e+06	1.927697e+06	1.996077e+06
1.991496e+01	1.587761e+00	4.953096e-01	3.546390e+00	5.972019e-01	9.990478e-01	2.897851e+01	2.307772e+00	1.429360e-01	1.426458e+01
0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
9.300000e+00	0.000000e+00	5.000000e-01	1.000000e+00	0.000000e+00	1.000000e+00	1.596000e+01	2.500000e+00	0.000000e+00	1.100000e+01
1.350000e+01	1.000000e+00	5.000000e-01	2.850000e+00	0.000000e+00	1.000000e+00	2.100000e+01	2.500000e+00	0.000000e+00	1.500000e+01
2.190000e+01	2.500000e+00	5.000000e-01	4.420000e+00	0.000000e+00	1.000000e+00	3.094000e+01	2.500000e+00	0.000000e+00	1.900000e+01
1.431635e+05	2.080000e+01	4.000000e+00	2.230800e+02	1.430000e+02	1.000000e+00	1.431675e+05	2.500000e+00	1.750000e+00	2.300000e+01
1.029504e+02	1.829217e+00	4.849072e-02	4.053963e+00	2.189708e+00	2.843222e-02	1.038516e+02	6.660463e-01	4.648817e-01	5.807630e+00

2.2. Handling Missing Values

2.2.1. Find the proportion of missing values in each column

In this section, I calculated the proportion of missing values in each column to decide whether to drop or replace them. If a column has a very low proportion of missing values, we can safely drop those rows as it will not significantly impact the data analysis. However, if the proportion of missing values is high, we should replace them using appropriate methods such as the median, mode, or mean, depending on the nature of the data.
After handling the missing values, the data is presented as follows:

```
VendorID                0.000000
tpep_pickup_datetime    0.000000
tpep_dropoff_datetime   0.000000
passenger_count         0.034257
trip_distance           0.000000
RatecodeID              0.034257
PULocationID            0.000000
DOLocationID            0.000000
payment_type            0.000000
fare_amount             0.000000
extra                   0.000000
mta_tax                 0.000000
tip_amount              0.000000
tolls_amount            0.000000
improvement_surcharge   0.000000
total_amount            0.000000
congestion_surcharge    0.034257
airport_fee             0.034257
date                    0.000000
hour                    0.000000
dtype: float64
```

2.2.2. Handling missing values in passenger_count

In this section, I checked for null (NaN) values. For the passenger_count column, I replaced all NaN values with the median, since it is a numeric column. I also identified rows where passenger_count was zero and replaced those values with the median as

well, instead of dropping them. This approach ensures data integrity while handling invalid or missing entries.
After these changes, the resulting data is represented as follows:

	passenger_count
count	1.996077e+06
mean	1.372197e+00
std	8.642891e-01
min	1.000000e+00
25%	1.000000e+00
50%	1.000000e+00
75%	1.000000e+00
max	9.000000e+00

2.2.3. Handle missing values in RatecodeID

In this section, find out the missing value of RatecodeID and replace it with mode, as it is a categorical column

2.2.4. Impute NaN in congestion_surcharge

The `congestion_surcharge` column contains numerical values, so I replaced all its null values with the median.Several other columns, such as `airport_fee`, also had missing values. I replaced those missing values with the respective column medians. After these replacements, there are no null values remaining in the dataset.The updated table now looks like this:

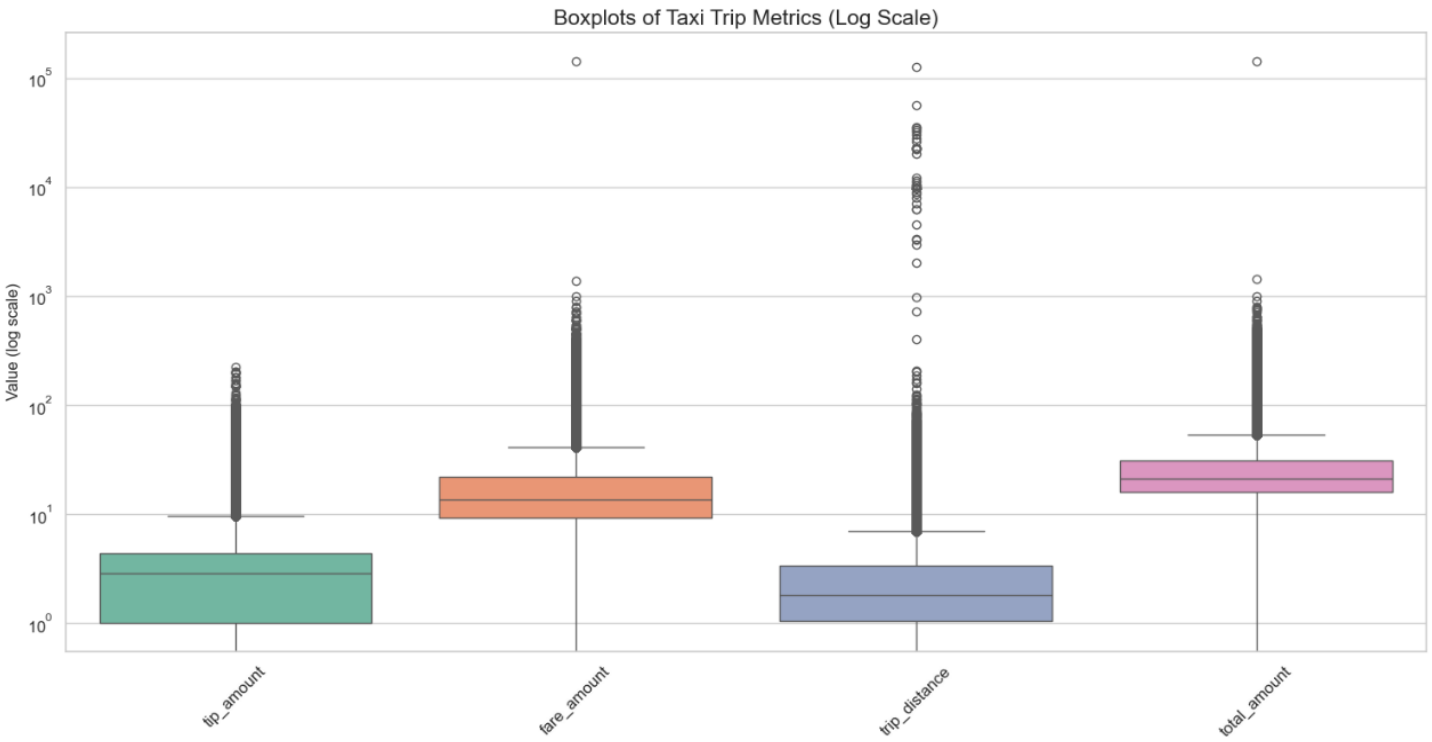
```
VendorID                0
tpep_pickup_datetime    0
tpep_dropoff_datetime   0
passenger_count         0
trip_distance           0
RatecodeID              0
PULocationID            0
DOLocationID            0
payment_type            0
fare_amount             0
extra                   0
mta_tax                 0
tip_amount              0
tolls_amount            0
improvement_surcharge   0
total_amount            0
congestion_surcharge    0
airport_fee             0
date                    0
hour                    0
dtype: int64
```

2.3. Handling Outliers and Standardising Values

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

After cleaning the data by removing null values and zeros, I identified outliers in the numeric columns: tip_amount, fare_amount, trip_distance, and total_amount.

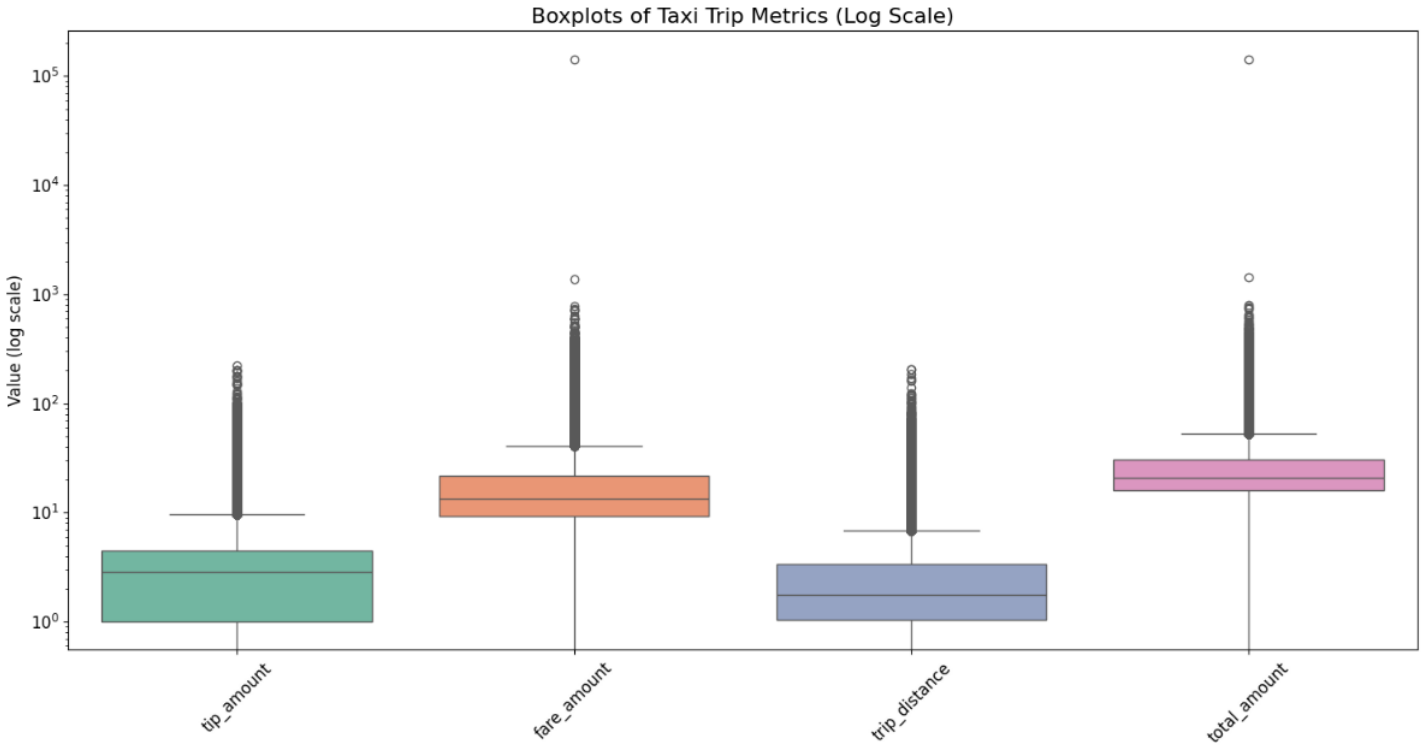
Before remove outlier graph represent like



After analyzing the data, I found that many outliers were caused by errors in recording trip details, so I addressed them accordingly:

- I dropped all rows where passenger_count was greater than 6.
- I removed entries where the trip_distance was nearly 0 but the fare_amount exceeded 300.
- I removed rows where both trip_distance and fare_amount were 0, but the pickup and drop-off zones were different, indicating inconsistent data.
- For trip_distance, I retained only values less than or equal to 250.
- In the payment_type column, I removed rows with a value of 0, as 0 does not correspond to any valid payment type.
- In the RatecodeID column, I removed rows with a value of 99.0, as this was identified as an outlier.

After removing these outliers perform the graph is look like



we can see the difference between both diagram before outliers and after outliers implement, differences are ->

 The first plot shows a larger number of extreme outliers across all four metrics (tip_amount, fare_amount, trip_distance, total_amount) compared to the second plot.

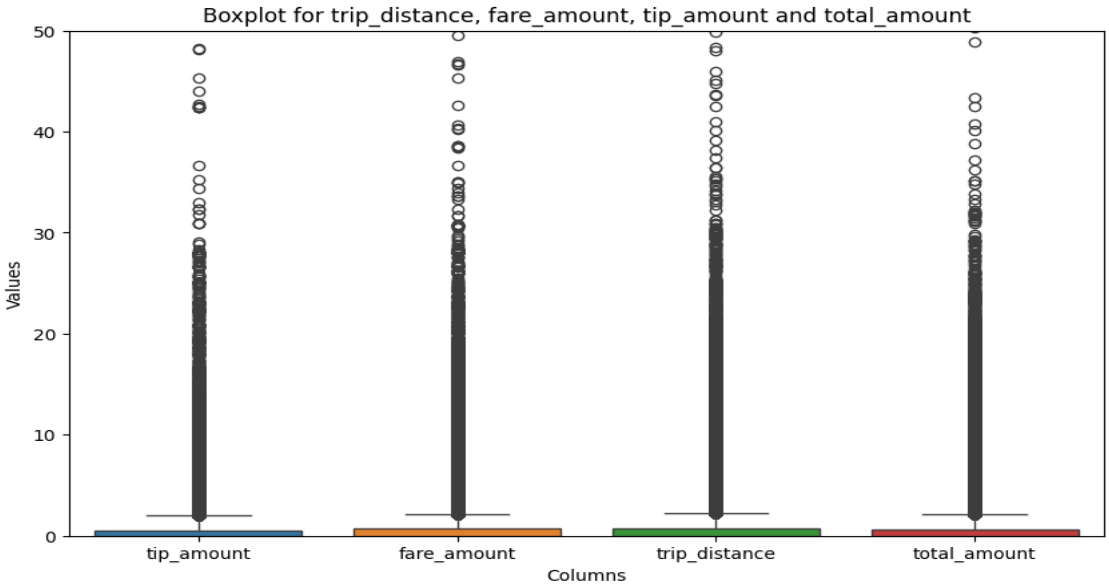
 After outlier removal (as shown in the second plot), the distributions appear tighter with fewer extreme values, especially in fare_amount and trip_distance.

 The central tendency (median) and interquartile ranges are more consistent in the cleaned data, indicating improved data quality.

 Overall, the second plot represents a cleaner dataset that is more suitable for analysis, with reduced skewness and fewer anomalies.

I also perform some standardization on the columns by using IQR (Q3 - Q1), applying a logarithmic transformation, and replacing outliers with the median value.

After doing this, the graph will look like:



After doing this, I also checked the data types of the columns and corrected date, passenger_count, and RatecodeID. The date column, which was of type object, should be converted to datetime. passenger_count should not be in float, as it always represents an integer, and similarly, RatecodeID should be of type int, as there is no precision value defined for it. After these corrections, the data will look like ->

```
<class 'pandas.core.frame.DataFrame'>
Index: 1916559 entries, 0 to 1996076
Data columns (total 20 columns):
 #   Column                 Dtype
---  ------                 -----
 0   VendorID               int64
 1   tpep_pickup_datetime   datetime64[ns]
 2   tpep_dropoff_datetime  datetime64[ns]
 3   passenger_count        int32
 4   trip_distance          float64
 5   RatecodeID             int32
 6   PULocationID           int64
 7   DOLocationID           int64
 8   payment_type           int64
 9   fare_amount            float64
 10  extra                  float64
 11  mta_tax                float64
 12  tip_amount             float64
 13  tolls_amount           float64
 14  improvement_surcharge  float64
 15  total_amount           float64
 16  congestion_surcharge   float64
 17  airport_fee            float64
 18  date                   datetime64[ns]
 19  hour                   int32
dtypes: datetime64[ns](3), float64(10), int32(3), int64(4)
memory usage: 285.1 MB
```

Finally we can say that -> To fix the outliers, we have followed the following steps:

Some points to consider:

- Removal of entries where trip_distance is nearly 0 and fare_amount is more than 300.
- Removal of entries where both trip_distance and fare_amount are 0, but the pickup and dropoff zones are different.
- Consider entries where trip_distance is less than or equal to 250 miles.
- Consider entries where payment_type is greater than 0.

Additionally, I have also removed entries where RatecodeID has a value of 99. After this, I performed standardization on the data. Finally, I converted the data types of a few columns as needed. For example, the date column was converted to datetime type, and passenger_count and RatecodeID were converted to integers."

3. Exploratory Data Analysis

3.1. General EDA: Finding Patterns and Trends

3.1.1. Classify variables into categorical and numerical

Categorical variable ->

. VendorID

. RatecodeID

. PULocationID

. DOLocationID

. payment_type

Numerical Variable ->

. fare_amount

. extra

. mta_tax

. tip_amount

. tolls_amount

. improvement_surcharge

. total_amount

. congestion_surcharge

. airport_fee

. passenger_count

. trip_distance

. pickup_hour

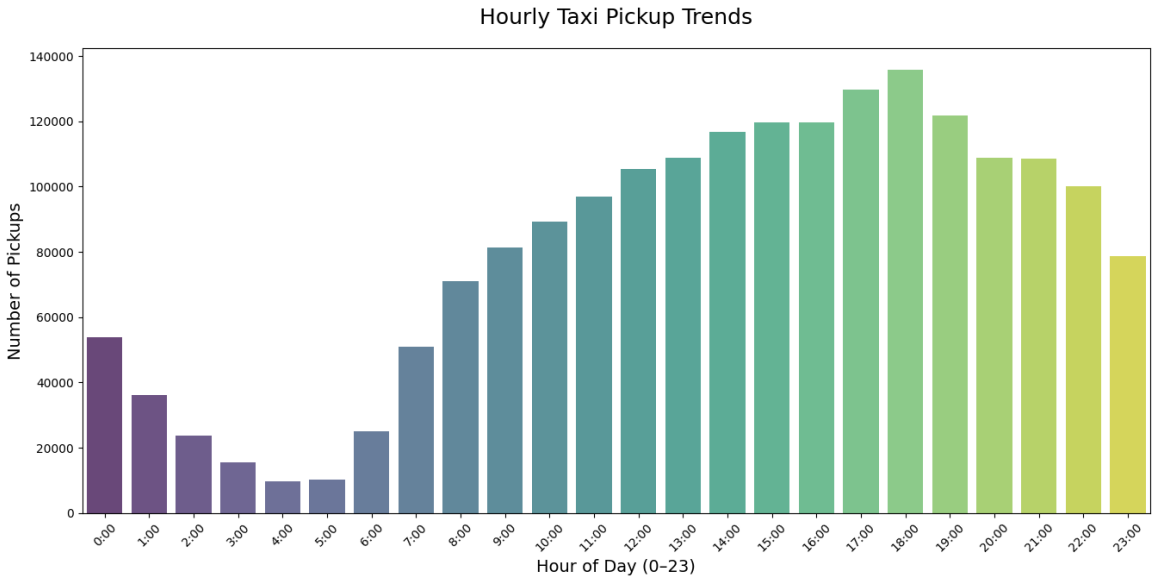
. trip_duration

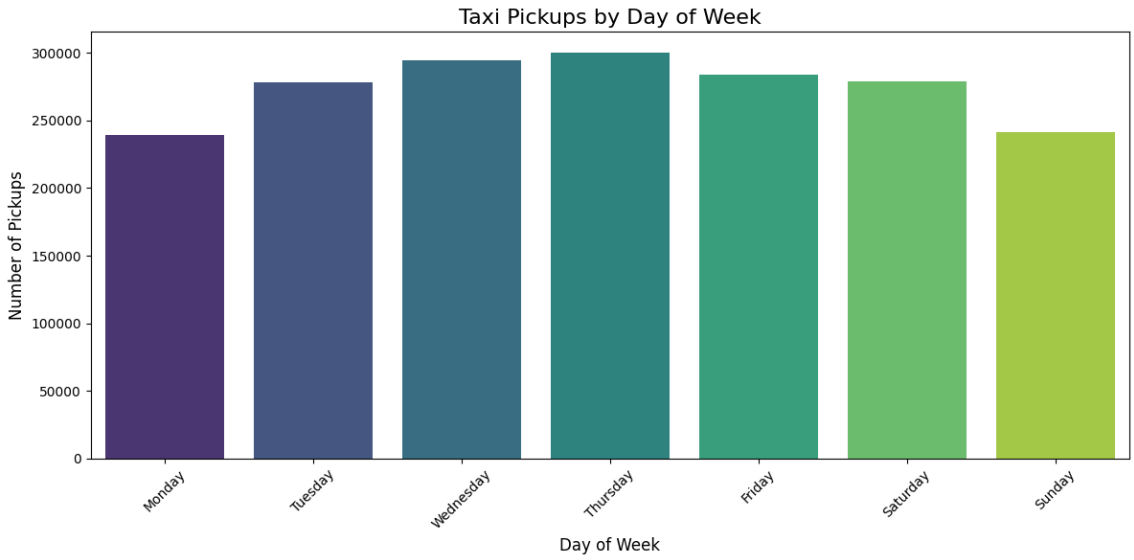
DateTime varibable

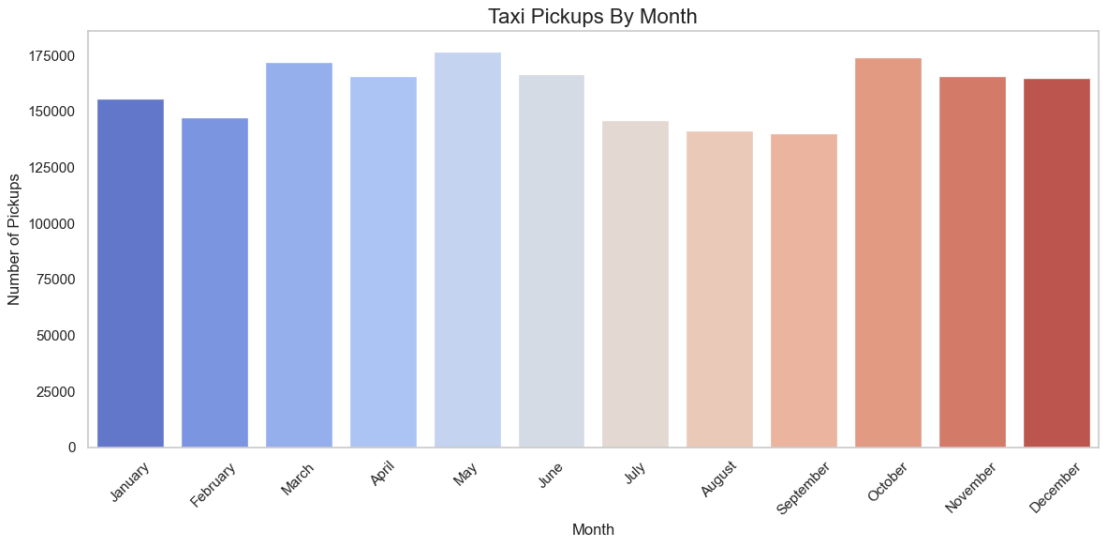
. tpep_pickup_datetime

. tpep_dropoff_datetime

3.1.2. Analyse the distribution of taxi pickups by hours, days of the week, and months In this section, I find the hourly trend in taxi pickup then data will look like







In the above graph, we can say
1. **Hourly Trends**: Taxi pickups peak between 16:00 and 19:00, with a significant drop in the early morning hours (especially from 2:00 to 5:00).

2. **Day of Week Trends**: Thursday sees the highest number of pickups, followed closely by Wednesday and Friday, while Monday and Sunday have the lowest.

3. **Monthly Trends**: March, May, and October have the highest pickup counts, whereas July through September show a noticeable dip in activity.

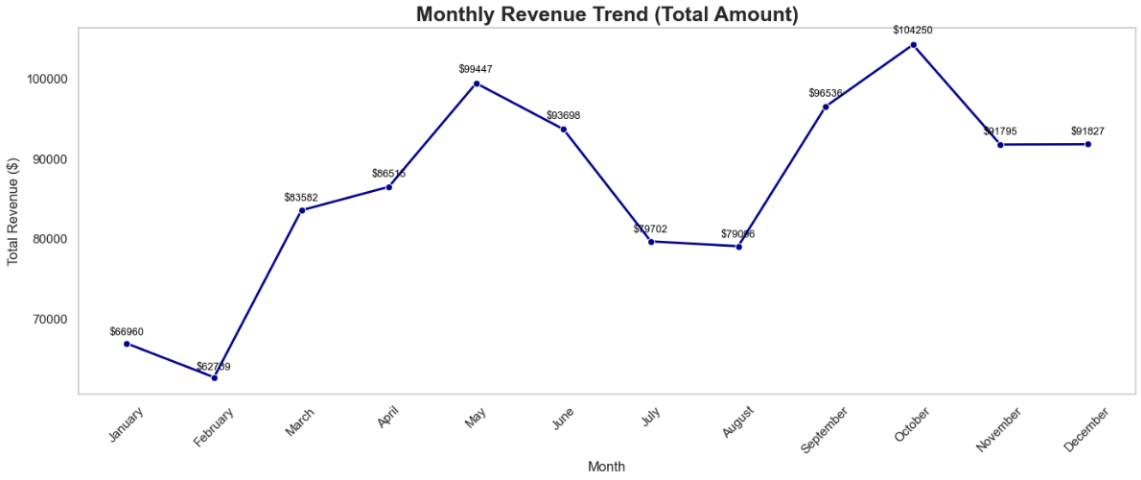
4 These patterns suggest stronger demand during weekday evenings and certain months, likely influenced by work schedules and seasonal factors.

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

In this section, removing rows with zero values helps with data cleaning, especially if zeros are invalid or represent missing data. For instance, a fare_amount of zero might indicate an incomplete or erroneous record. Therefore, I removed all rows containing zero values in the columns fare_amount, tip_amount, total_amount, and trip_distance. Filtering out zero values could also improve performance by reducing memory usage and processing time on large datasets. Moreover, excluding zeros can lead to more accurate statistical analysis by avoiding skewed results.

3.1.4. Analyse the monthly revenue trends

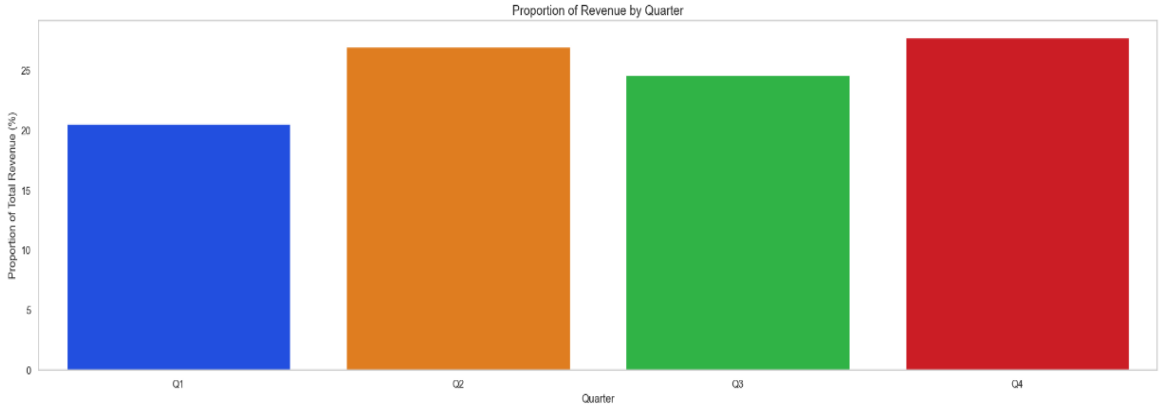
for analysis the monthly revenue, we have to group month and find total amount then graph look like ->



The monthly revenue trend shows peaks in May and October, with October generating the highest revenue at over $104,000. Revenue dips are observed in February, July, and August, with the lowest in February. Overall, the data suggests a cyclical pattern, likely driven by seasonal demand and external events.

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

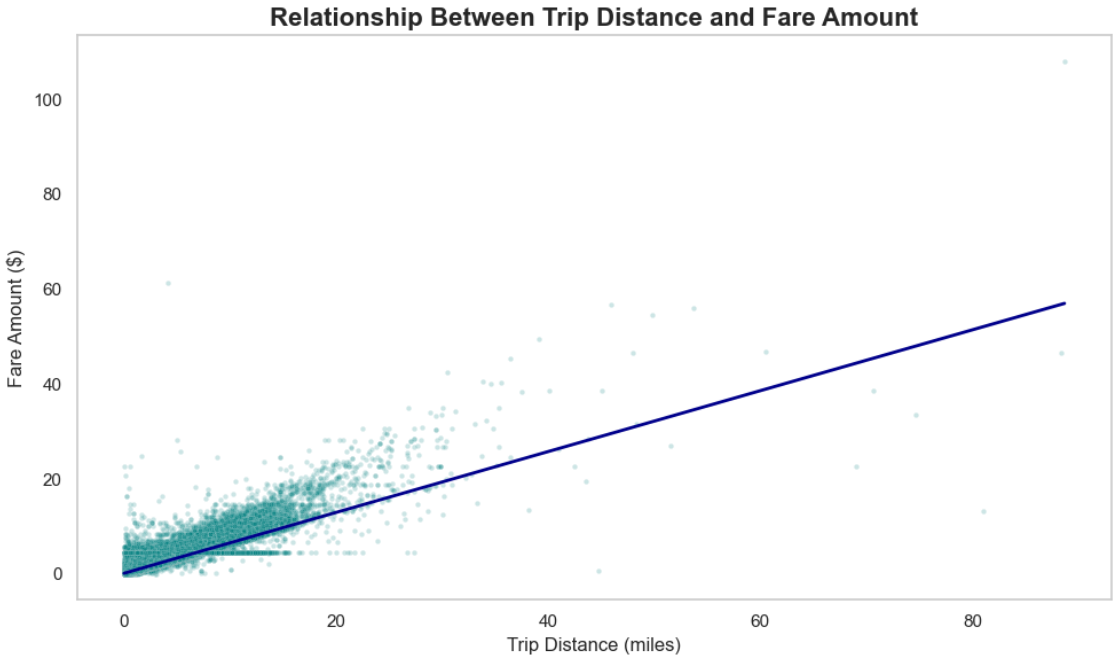
For calculating the proportion of each quarter, firstly create a column of quarter and then group quarter and sum the total amount. The graph will look like ->



The chart shows that **Q4** contributes the highest proportion of total revenue, followed closely by **Q2**. **Q1** has the lowest revenue share among all quarters. This suggests stronger business performance in the second half of the year, especially during the final quarter.

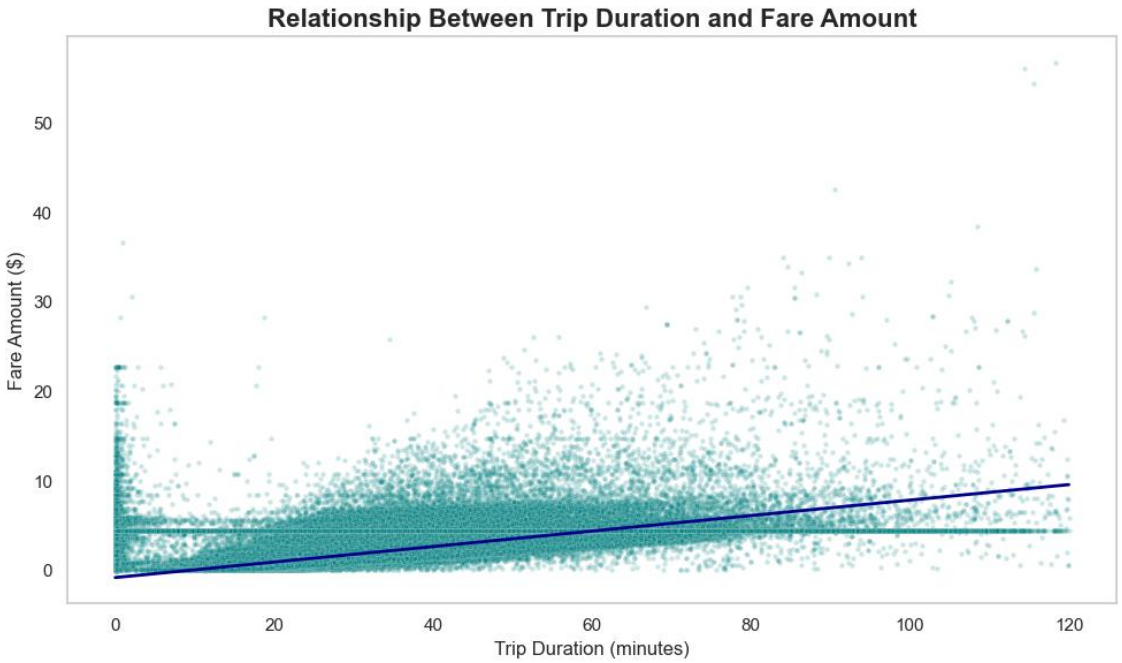
3.1.6. Analyse and visualise the relationship between distance and fare amount

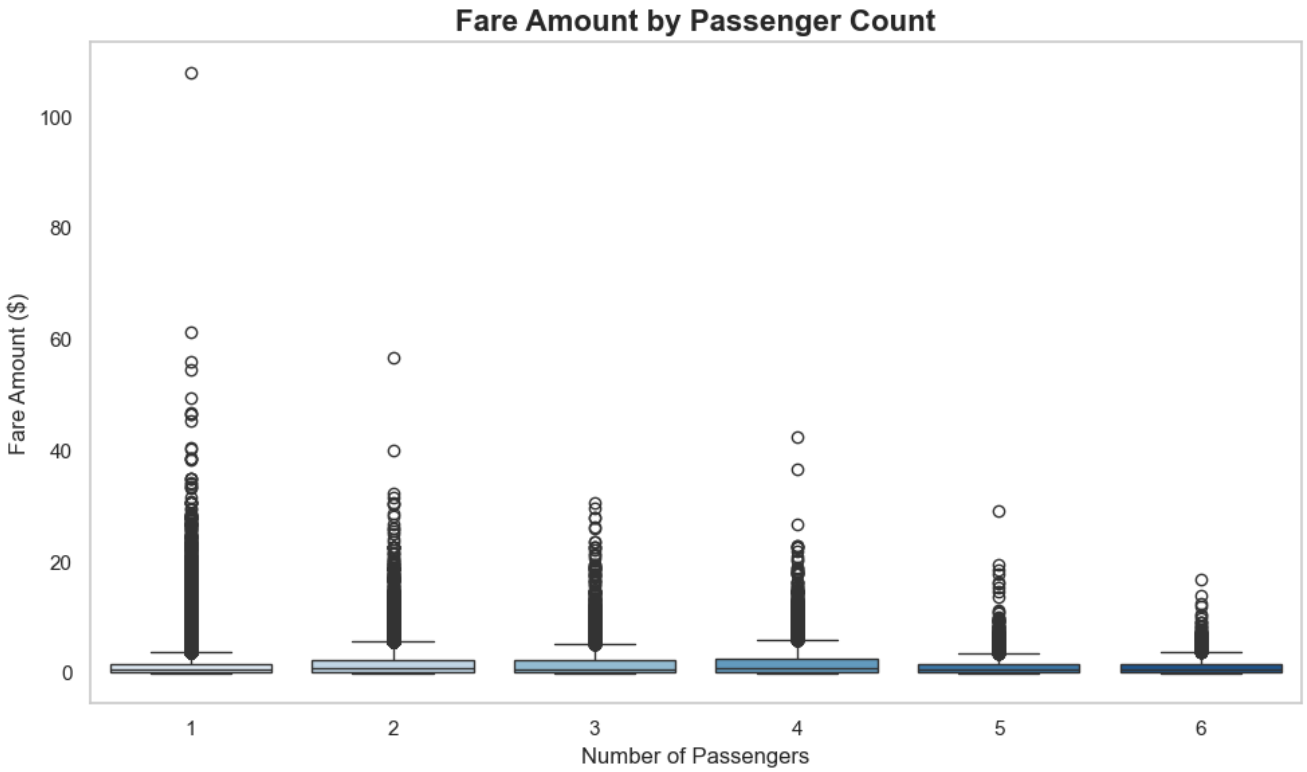
The relationship between trip distance and fare amount look like ->

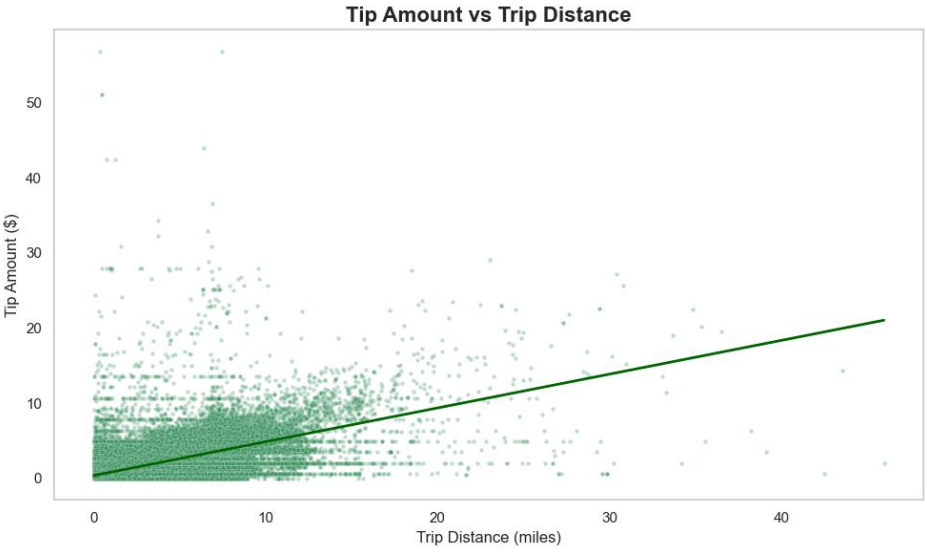


The scatter plot shows a **positive linear relationship** between trip distance and fare amount—longer trips generally result in higher fares. Most data points are concentrated at **shorter distances (under 10 miles)** with lower fare amounts. The trend line suggests consistent fare increases with distance, though there are a few outliers with unusually high fares.

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





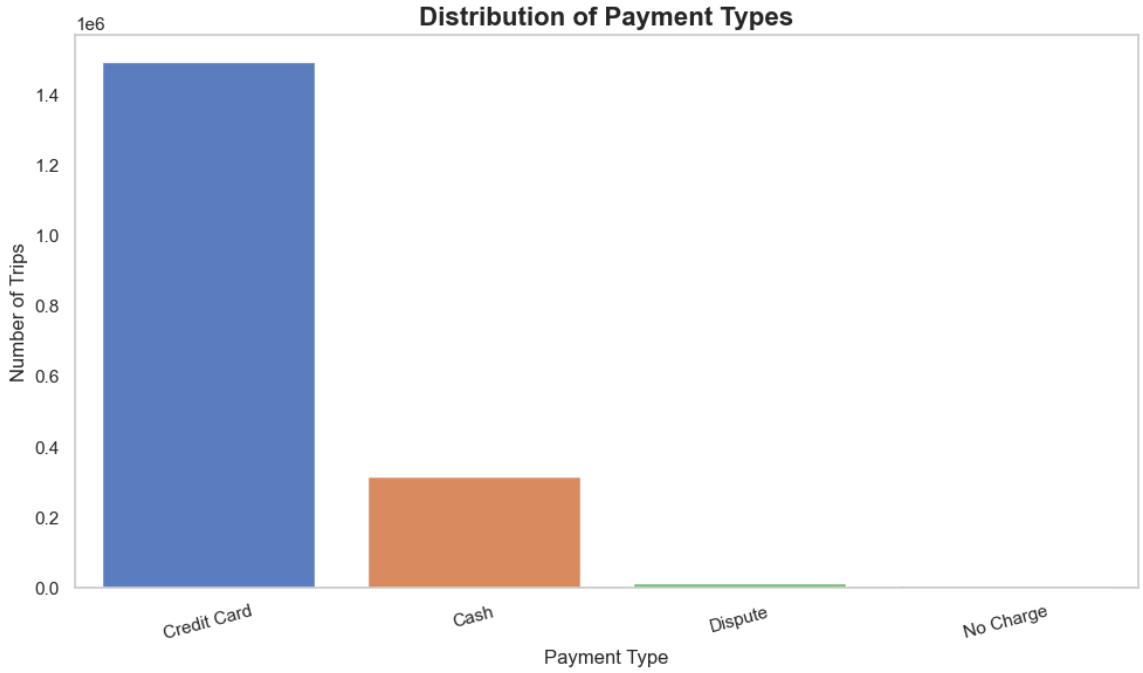


Observation of above diagram are ->

1. **Trip Duration vs Fare Amount**: There's a weak but positive relationship, indicating longer trips generally lead to higher fares, though with significant variability and many low-fare long-duration trips.
2. **Fare Amount by Passenger Count**: Fare amounts remain relatively consistent regardless of passenger count, with most fares clustering around the lower end, and occasional high-fare outliers.
3. **Tip Amount vs Trip Distance**: Tip amounts tend to increase slightly with trip distance, but like fares, are mostly concentrated at lower values with a few generous tips on longer trips.

Overall, **trip distance** has a stronger correlation with both **fare and tip amounts**, while **passenger count** has minimal impact.

3.1.8. Analyse the distribution of different payment types

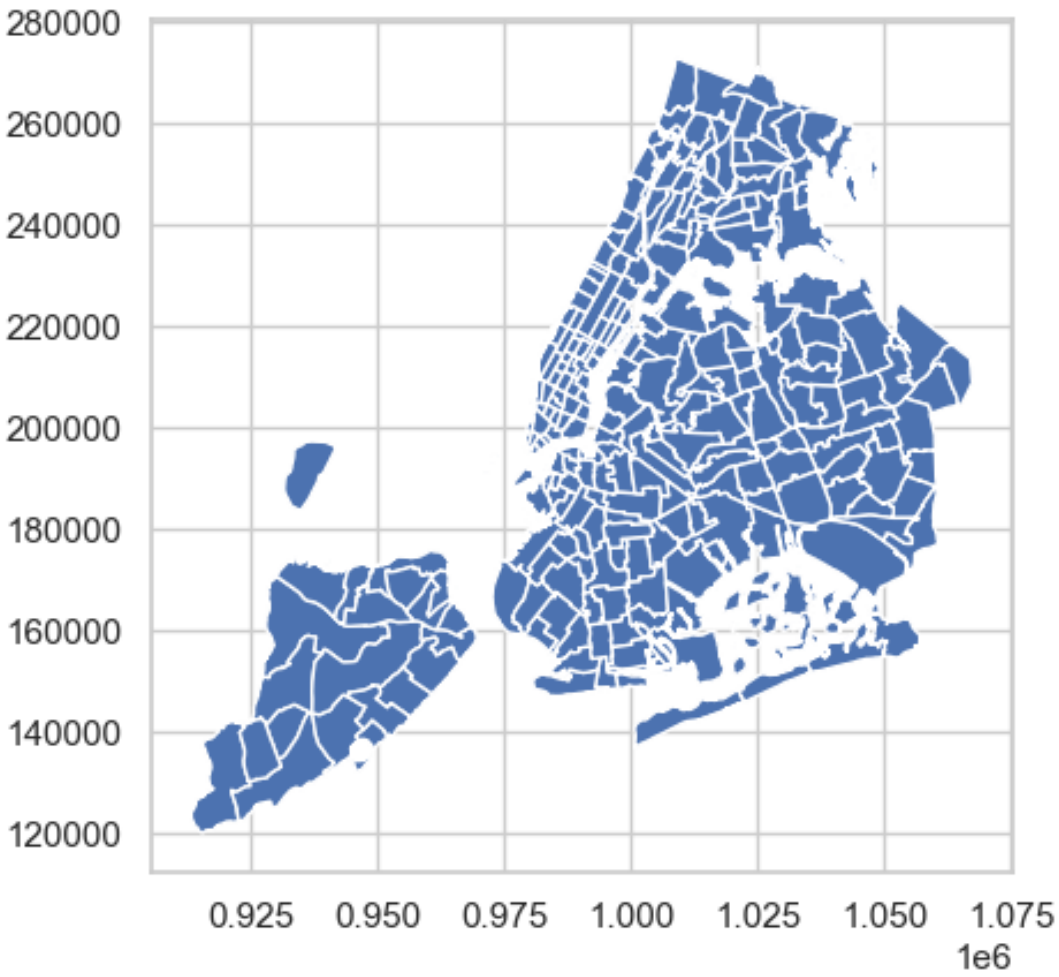


According to above diagram, we can say that the majority of trips were paid using credit cards, with over 1.4 million transactions. Cash payments accounted for significantly fewer trips, followed by a minimal number of disputes. "No Charge" trips were the least common, indicating rare instances of free or voided rides.

3.1.9. Load the taxi zones shapefile and display it

In this section, we upload the taxi zones shapefile into a variable named zones. For loading the file, we import geopandas and represent the data with the code zones.head(). The table displays the first few rows of the zones data.

	OBJECTID	Shape_Leng	Shape_Area	zone	LocationID	borough	geometry
0	1	0.116357	0.000782	Newark Airport	1	EWR	POLYGON ((933100.918 192536.086, 933091.011 19...
1	2	0.433470	0.004866	Jamaica Bay	2	Queens	MULTIPOLYGON (((1033269.244 172126.008, 103343...
2	3	0.084341	0.000314	Allerton/Pelham Gardens	3	Bronx	POLYGON ((1026308.77 256767.698, 1026495.593 2...
3	4	0.043567	0.000112	Alphabet City	4	Manhattan	POLYGON ((992073.467 203714.076, 992068.667 20...
4	5	0.092146	0.000498	Arden Heights	5	Staten Island	POLYGON ((935843.31 144283.336, 936046.565 144...



3.1.10. Merge the zone data with trips data

we merge the zones and trip records using LocationId and PULocationID with left join.

Code:

```
df_geo = df.merge( zones[['LocationID', 'zone', 'borough']],
    left_on='PULocationID',
    right_on='LocationID',
    how='left')
df_geo.head()
```

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

For finding the number of trip with location id , we need to group by PULocationID and count the trip count.

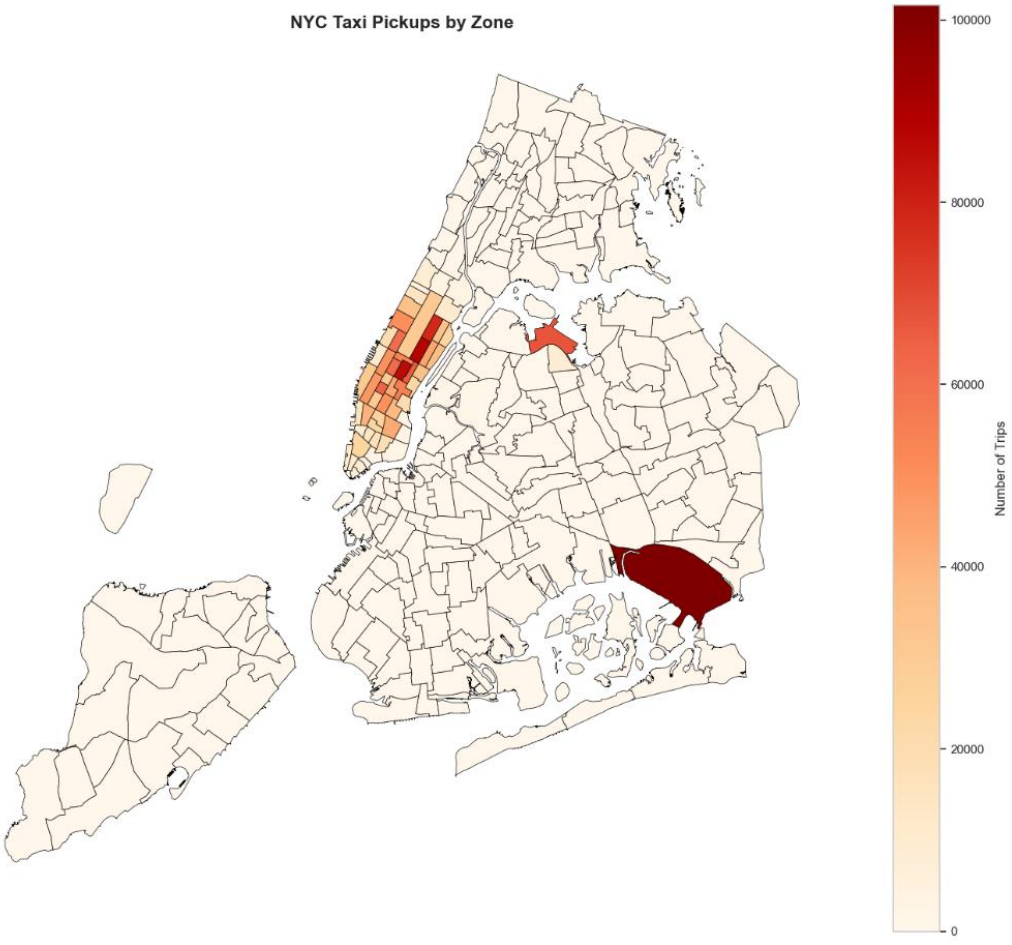
	PULocationID	trip_count
0	1	218
1	2	2
2	3	5
3	4	1794
4	6	25

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

In this section, we merge zones with PULocationID and trip_count using a left join and fill NaN values with 0. The resulting table displays the combined data.

	LocationID	zone	borough	trip_count
0	1	Newark Airport	EWR	218
1	2	Jamaica Bay	Queens	2
2	3	Allerton/Pelham Gardens	Bronx	5
3	4	Alphabet City	Manhattan	1794
4	5	Arden Heights	Staten Island	0

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



Darker red areas indicate higher pickup volumes, with two major hotspots visible: one in Midtown/Lower Manhattan (west side) and another in eastern Queens/JFK Airport area. Most other zones show minimal taxi activity (pale colors). The visualization effectively highlights the concentration of taxi services in specific high-demand areas of New York City.

3.1.14. Conclude with results

1. **Monthly Revenue Trends** show a peak in May and October, with the highest revenue in October ($104,250), indicating seasonal fluctuations in taxi usage and potential demand surges during certain months.
2. **Quarterly Revenue Proportions** reveal that Q2 and Q4 contribute the most to total annual revenue, while Q1 lags behind, suggesting increased ride activity in spring and late fall.
3. **Trip Distance vs Fare Amount** demonstrates a strong positive correlation—longer trips generally cost more, as expected in distance-based fare systems.
4. **Trip Duration vs Fare Amount** also shows a positive trend, but with more dispersion, indicating that traffic conditions or waiting time may influence fare inconsistencies.
5. **Fare Amount by Passenger Count** suggests that the number of passengers has minimal impact on fare, with most rides priced similarly regardless of passenger count—possibly due to flat rates or shared ride policies.
6. **Tip Amount vs Trip Distance** shows a slight positive correlation; longer trips often receive higher tips, though the variance is significant and many short trips still earn tips.
7. **Payment Type Distribution** is dominated by credit card payments, followed by cash. Disputes and no-charge trips are extremely rare, indicating digital payments are preferred.
8. **Pickup Location Heatmap** highlights two primary hotspots: Midtown/Lower Manhattan and JFK Airport. These areas see the most taxi activity, aligning with business centers and transport hubs, while other regions remain relatively quiet.

3.2. Detailed EDA: Insights and Strategies

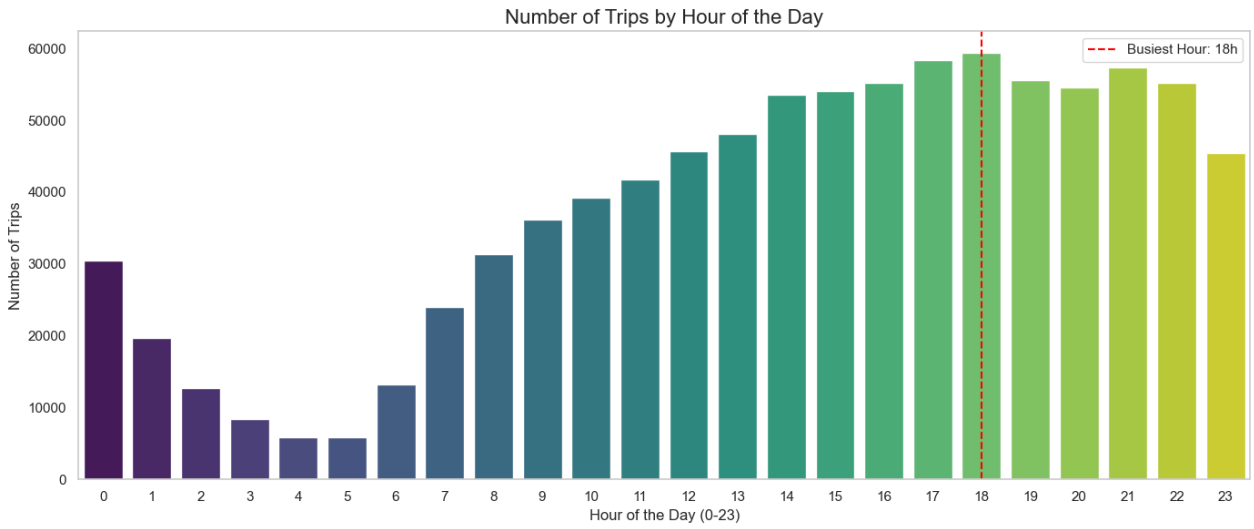
3.2.1. Identify slow routes by comparing average speeds on different route

To find out route which have slowest speed at different times of day, we have to calculate duration in minutes and filter outliers then after we group data (PULocationID, DOLocationID, pickup_hour) then after we calculate slow_routers which we explained in code . The result will look like:

	PULocationID	DOLocationID	pickup_hour	avg_distance	avg_duration_min	trip_count	avg_speed_mph
0	186	233	9	0.065502	21.815104	32	0.180156
1	113	170	11	0.052402	17.051333	25	0.184391
2	113	170	17	0.050461	15.008642	27	0.201728
3	162	186	17	0.070772	20.422414	29	0.207926
4	186	163	18	0.076419	21.653205	26	0.211754
5	163	186	16	0.082705	23.220707	33	0.213701
6	186	162	10	0.088256	24.028070	57	0.220381
7	238	142	12	0.049961	12.908333	34	0.232229
8	186	163	12	0.104279	26.938667	25	0.232260
9	162	186	16	0.081962	20.815812	39	0.236248

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

After calculating the number of trips by hour of day, Result will look like:



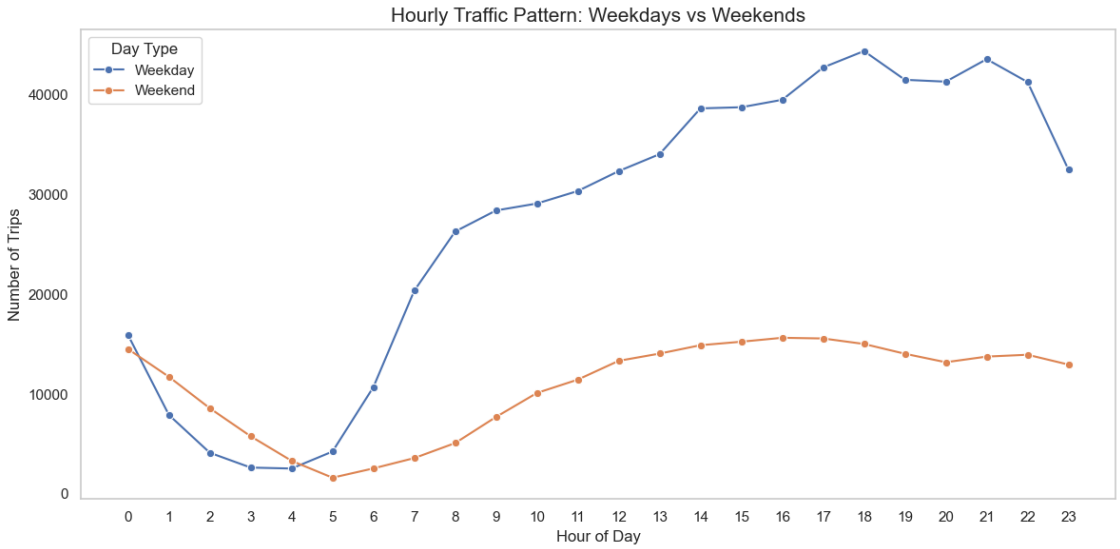
The bar chart shows taxi trip frequency throughout the day in NYC. Trip volumes are lowest during early morning hours (3-5 AM), then steadily increase from 6 AM onward, reaching peak activity at 6 PM (nearly 60,000 trips) marked as the busiest hour. Evening hours maintain high trip volumes before gradually declining after 10 PM, demonstrating clear patterns of urban mobility corresponding to typical daily commuting and activity cycles.

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

After scale up the number of trips then actual number of trips will be ->.

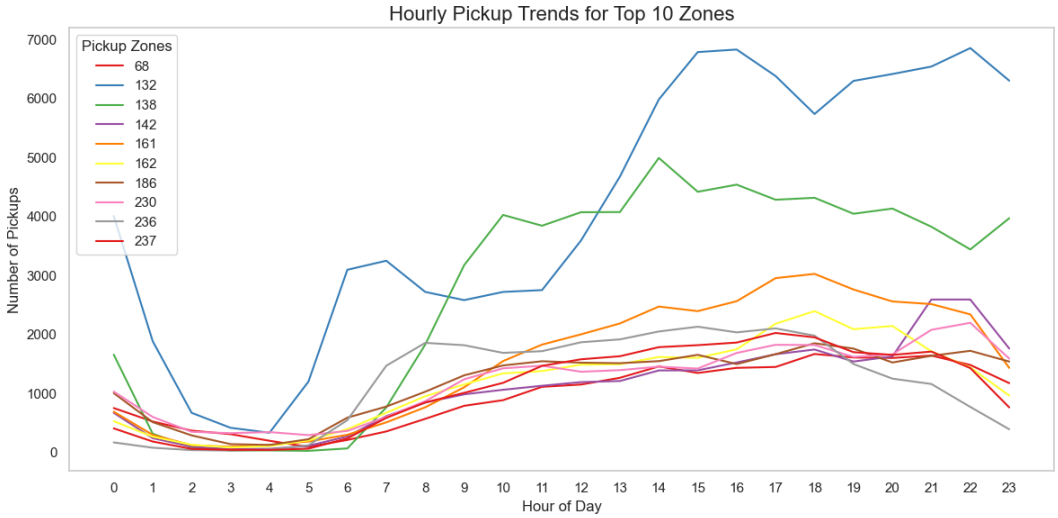
```
Top 5 Busiest Hours with Estimated Actual Trips:
    hour  sampled_trip_count  estimated_actual_trips
18    18               59419                 5941900
17    17               58326                 5832600
21    21               57344                 5734400
19    19               55556                 5555600
22    22               55232                 5523200
```

3.2.4. Compare hourly traffic on weekdays and weekends

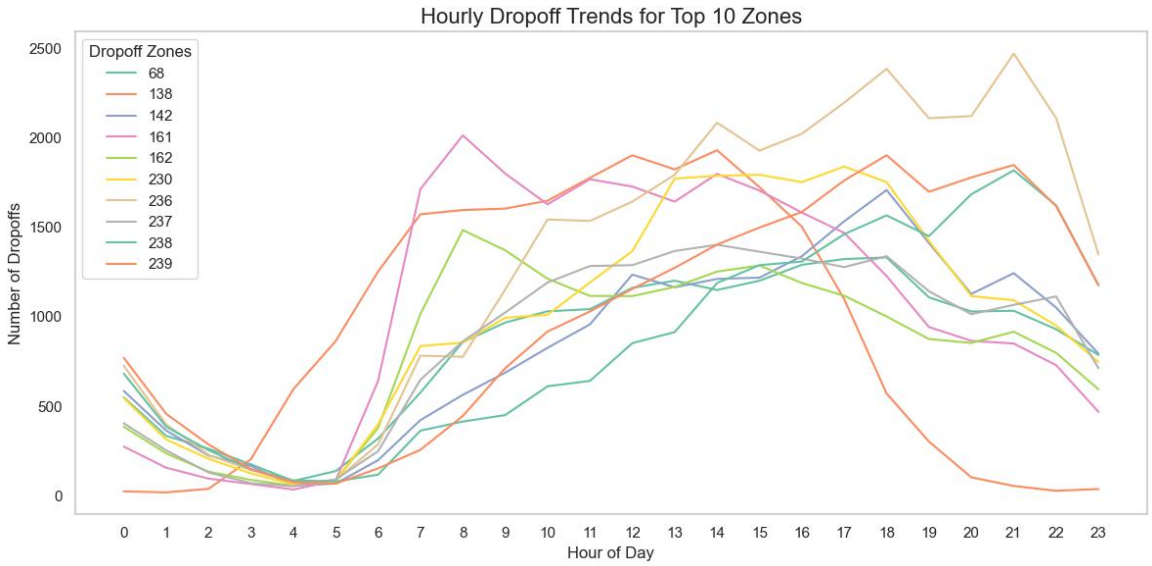


The line graph compares taxi trip patterns between weekdays and weekends across 24 hours. Weekday trips (blue line) show distinct commuting patterns with morning and evening peaks, reaching approximately 45,000 trips during evening rush hours. Weekend trips (orange line) remain consistently lower throughout the day, averaging around 15,000 trips during peak hours, and show a more gradual rise and fall without the sharp commuting peaks seen on weekdays.

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



The line graph shows hourly taxi pickup patterns for the top 10 zones in NYC. Zone 132 (blue line) dominates with the highest pickup volume, peaking at nearly 7,000 pickups during evening hours (15-18). Zone 142 (green line) ranks second, maintaining consistent activity between 4,000-5,000 pickups throughout the day. All zones follow a similar pattern with minimal activity during early morning hours (2-5 AM) before steadily increasing, though each zone has distinct peaks and volumes. While most zones show evening peaks around 17-20 hours, some zones like 68 and 132 show interesting morning patterns, suggesting different neighborhood characteristics or purposes (residential vs. business).



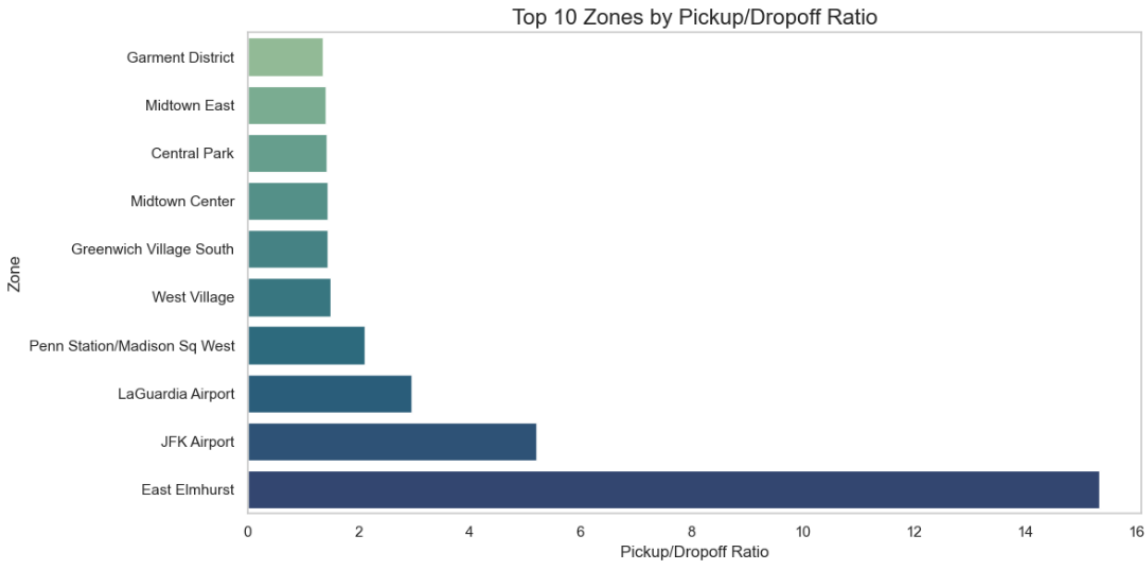
The line graph displays hourly dropoff patterns for the top 10 taxi zones in NYC. All zones show minimal activity during early morning hours (2-4 AM) before increasing, with most zones reaching peak dropoffs between 15-21 hours (3-9 PM). Zone 237 (light brown line) shows the highest dropoff volume, reaching nearly 2,500 dropoffs during evening peaks, while Zone 138 (orange line) displays an unusual pattern with high morning activity but virtually no evening dropoffs, suggesting it might be primarily a business or commuter destination.

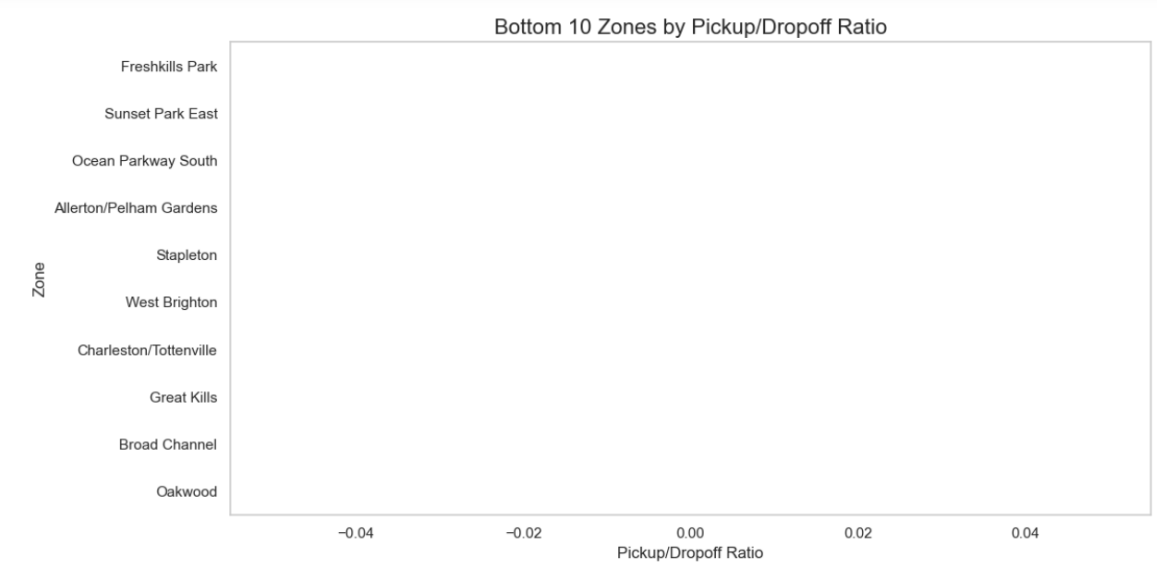
3.2.6. Find the ratio of pickups and dropoffs in each zone

Top 10 Zones by Pickup/Dropoff Ratio:

```
Top 10 Zones by Pickup/Dropoff Ratio:
                            zone  pickup_count  dropoff_count  \
70                 East Elmhurst        8266.0          539.0
132                  JFK Airport       98038.0        18853.0
138              LaGuardia Airport       65950.0        22287.0
186  Penn Station/Madison Sq West       28412.0        13440.0
249                 West Village       19037.0        12654.0
114       Greenwich Village South       12005.0         8241.0
161                Midtown Center       36742.0        25299.0
43                   Central Park       13580.0         9451.0
162                  Midtown East       28041.0        19746.0
100               Garment District       12367.0         9064.0

     pickup_dropoff_ratio
70              15.335807
132              5.200127
138              2.959124
186              2.113988
249              1.504425
114              1.456741
161              1.452310
43               1.436885
162              1.420085
100              1.364409
```

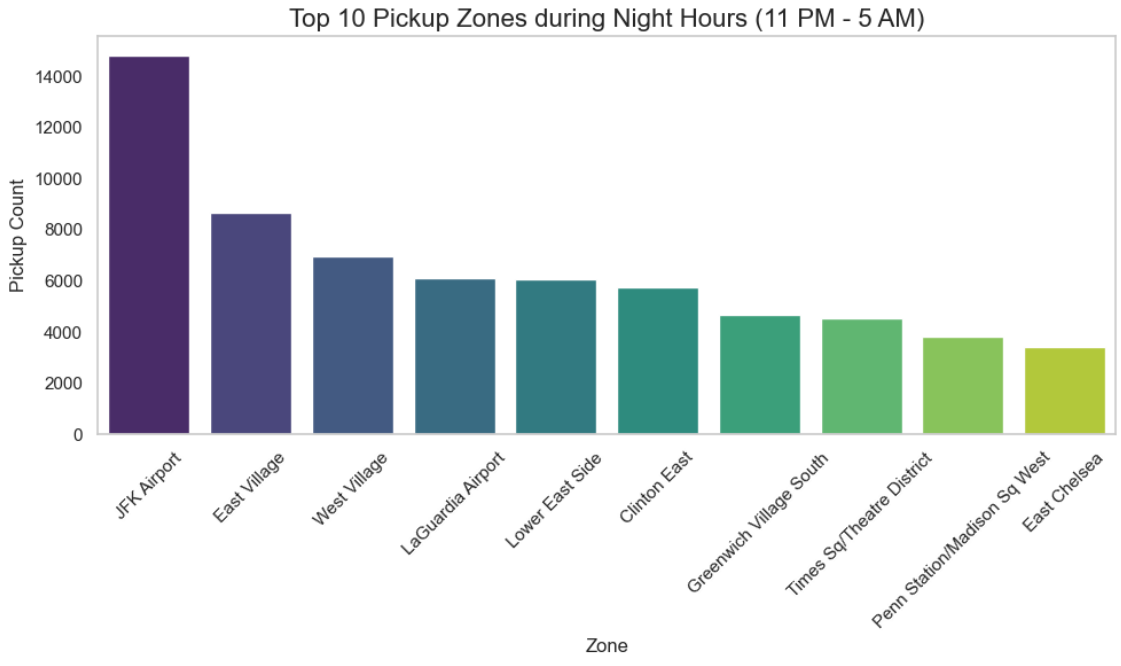
```
Bottom 10 Zones by Pickup/Dropoff Ratio:
                        zone  pickup_count  dropoff_count  \
99             Freshkills Park           0.0            1.0
227          Sunset Park East           0.0          283.0
178       Ocean Parkway South           0.0          144.0
3     Allerton/Pelham Gardens           0.0          111.0
221                 Stapleton           0.0           33.0
245             West Brighton           0.0           31.0
44     Charleston/Tottenville           0.0            6.0
109                Great Kills           0.0           25.0
30              Broad Channel           0.0           15.0
176                   Oakwood           0.0           12.0

     pickup_dropoff_ratio
99                    0.0
227                   0.0
178                   0.0
3                     0.0
221                   0.0
245                   0.0
44                    0.0
109                   0.0
30                    0.0
176                   0.0
```

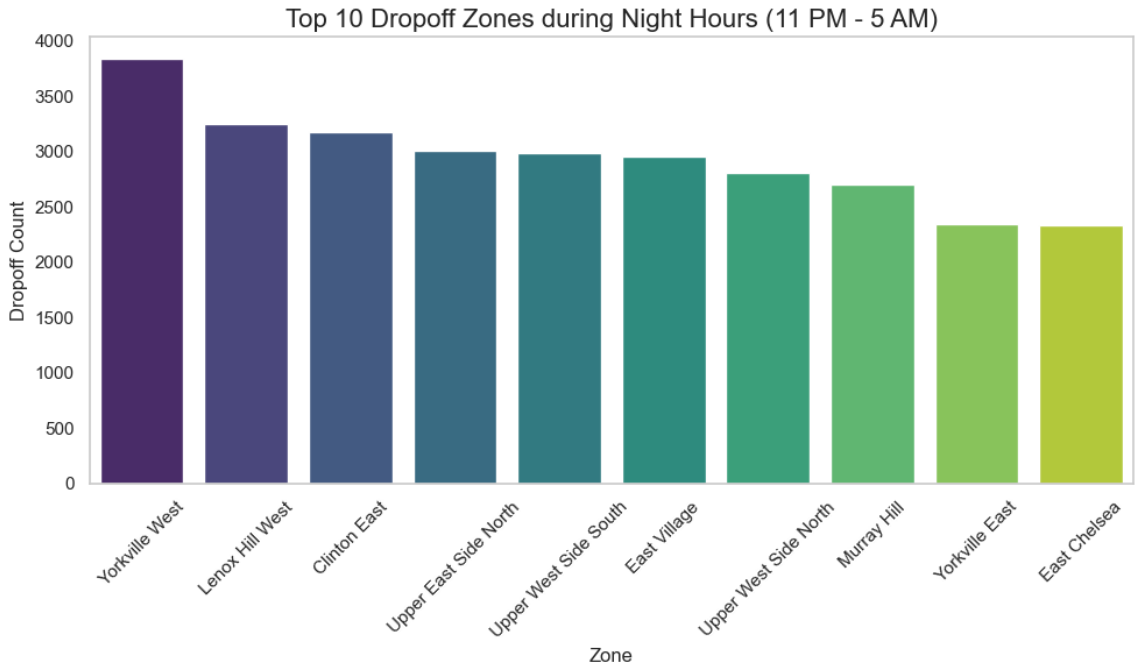




Based on the above data, airports and transit hubs dominate pickup-to-dropoff ratios. JFK Airport (5.2:1) and LaGuardia (3.0:1) show extremely high pickup demand, while East Elmhurst has the most extreme ratio at 15.3:1. Manhattan locations like Penn Station, Midtown Center, and the Village areas maintain solid ratios between 1.4-2.1:1, indicating their importance as origin points in the city's transportation network.

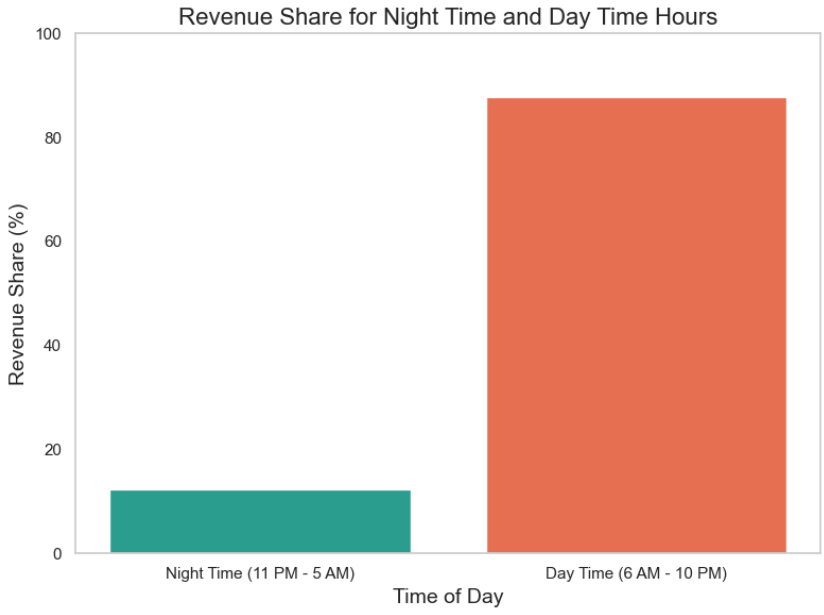
3.2.7. Identify the top zones with high traffic during night hours





The nighttime taxi data shows a clear pattern where JFK Airport dominates as the top pickup zone (>14,000 rides), followed by East Village and West Village. For dropoffs, Yorkville West leads with nearly 4,000 rides, with Lenox Hill West and Clinton East following closely. Notable is the absence of overlap between top pickup and dropoff zones - airports and popular nightlife areas generate rides, while residential neighborhoods like Yorkville and Upper East/West Side are destinations. This suggests distinctive nighttime travel flows across NYC.

3.2.8. Find the revenue share for nighttime and daytime hours



The chart shows a significant disparity in taxi revenue distribution between night and day hours in NYC. Daytime operations (6 AM - 10 PM) generate approximately 87% of total

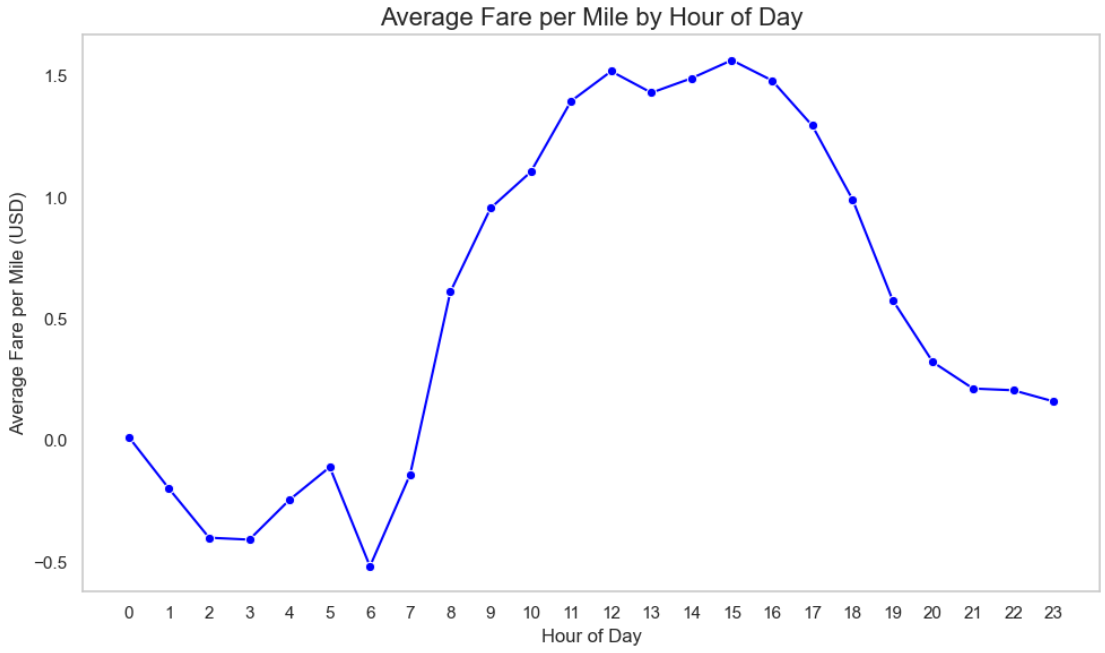
revenue, while nighttime hours (11 PM - 5 AM) account for only about 13%. This 6.7:1 ratio highlights the overwhelming economic importance of daytime taxi service despite covering 16 hours versus night's 6-hour window. The data suggests that while nightlife creates notable travel patterns as seen in previous charts, the taxi industry's financial foundation remains firmly rooted in daytime service.

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

	passenger_count	avg_fare_per_mile
0	1	1.444605
1	2	1.380790
2	3	1.468744
3	4	1.467013
4	5	1.307002
5	6	1.285484

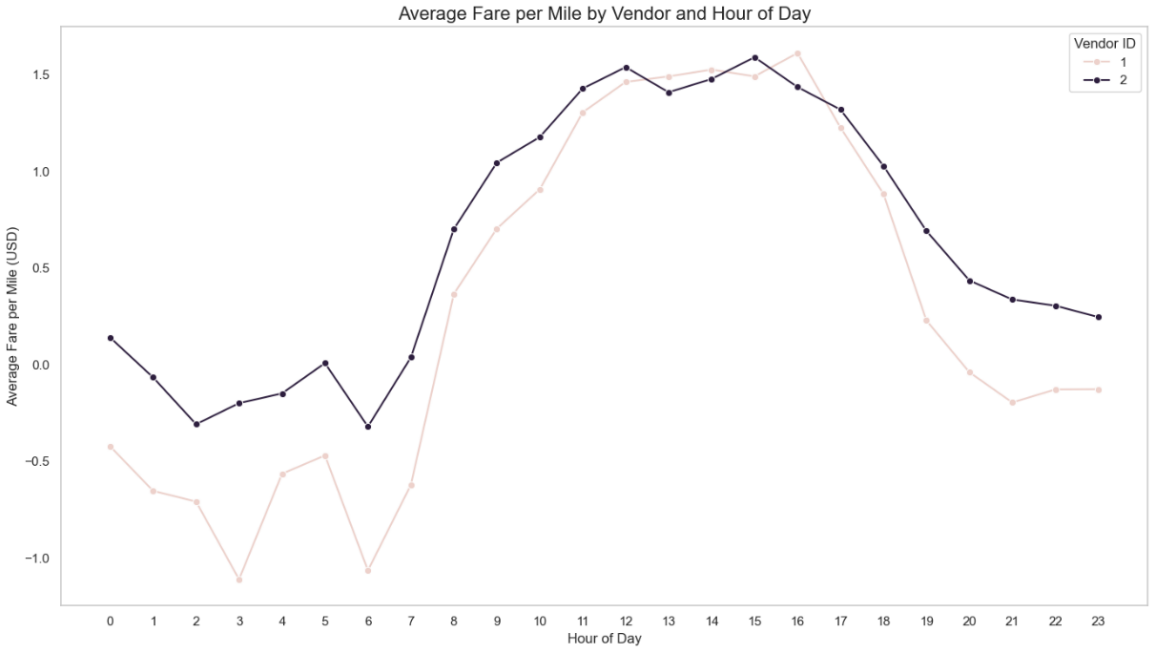
The data reveals that medium-sized groups (3-4 passengers) pay the highest fares per mile at approximately $1.47, while larger groups (5-6 passengers) receive the best value at around $1.29-$1.31 per mile. Single riders fall in the middle range at $1.44, suggesting taxi pricing may reflect vehicle utilization efficiency rather than a straightforward p er-person discount structure. This non-linear pricing pattern indicates potential optimization opportunities for taxi ser vices based on passenger count

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



The graph shows dramatic variation in fare per mile by hour, with negative values during early morning hours (midnight to 7 AM) suggesting potential data anomalies or special fare structures. Prices peak during midday and afternoon (11 AM to 5 PM) at approximately $1.50 per mile, then steadily decline through evening hours. This pattern indicates higher pricing during business hours when demand and traffic congestion are highest, with significantly lower rates during off-peak overnight periods.

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



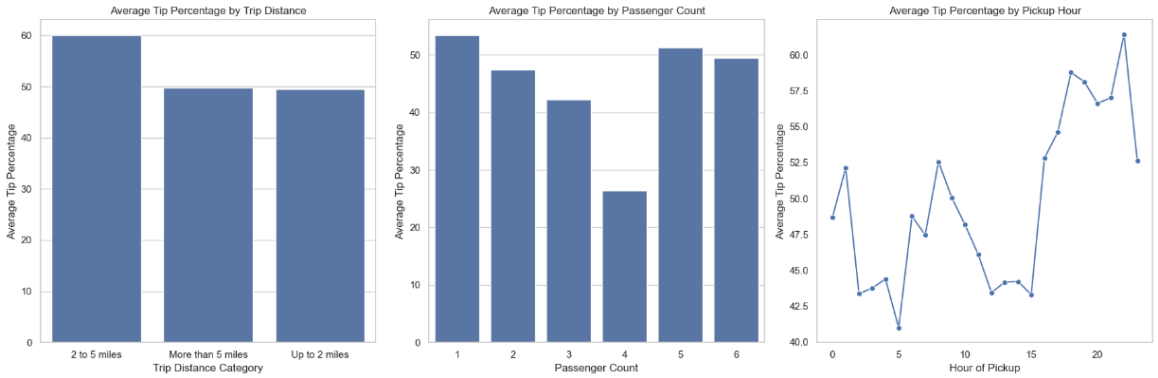
The graph reveals significant pricing differences between two taxi vendors throughout the day, with Vendor 2 (black line) consistently charging higher fares per mile than Vendor 1 (pink line), especially during overnight hours. Both vendors show similar daytime pricing patterns with peak rates ($1.50-$1.60) from 11 AM to 4 PM, but Vendor 1's rates drop more dramatically after 6 PM, reaching negative values in early morning hours. These negative values for both vendors during overnight periods (12 AM-7 AM) suggest either data anomalies or special fare structures, though Vendor 2 maintains comparatively higher rates throughout the entire 24-hour cycle.

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

	VendorID	distance_tier	fare_per_mile
0	1	2 to 5 miles	0.682840
1	1	More than 5 miles	0.656553
2	1	Up to 2 miles	0.620869
3	2	2 to 5 miles	0.698610
4	2	More than 5 miles	0.657379
5	2	Up to 2 miles	0.935280

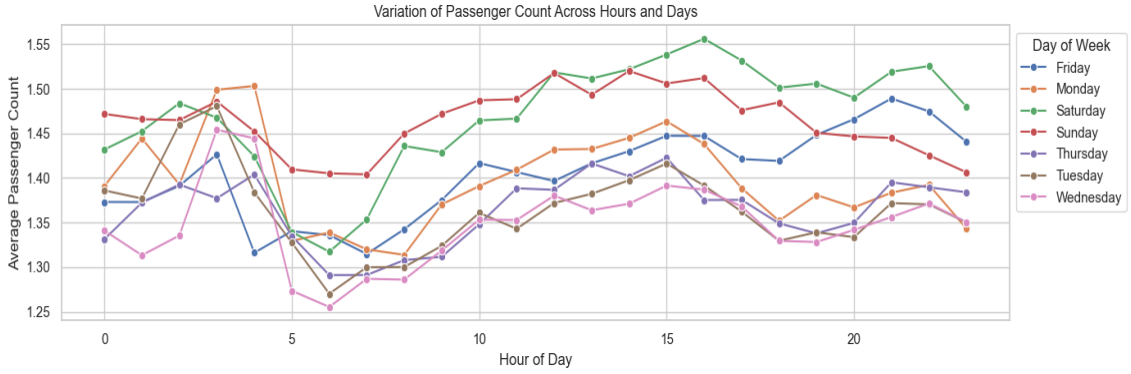
The data reveals distinct pricing strategies between vendors based on trip distance. Vendor 2 charges substantially more for short trips (up to 2 miles) at $0.94/mile versus Vendor 1's $0.62/mile, representing a 50% premium. For me dium (2-5 miles) and long trips (5+ miles), both vendors offer comparable rates around $0.68-$0.70/mile and $0.66/ mile respectively. This suggests Vendor 2 targets higher profits on short trips, while both vendors maintain similar co mpetitive pricing for longer journeys.

3.2.13. Analyse the tip percentages



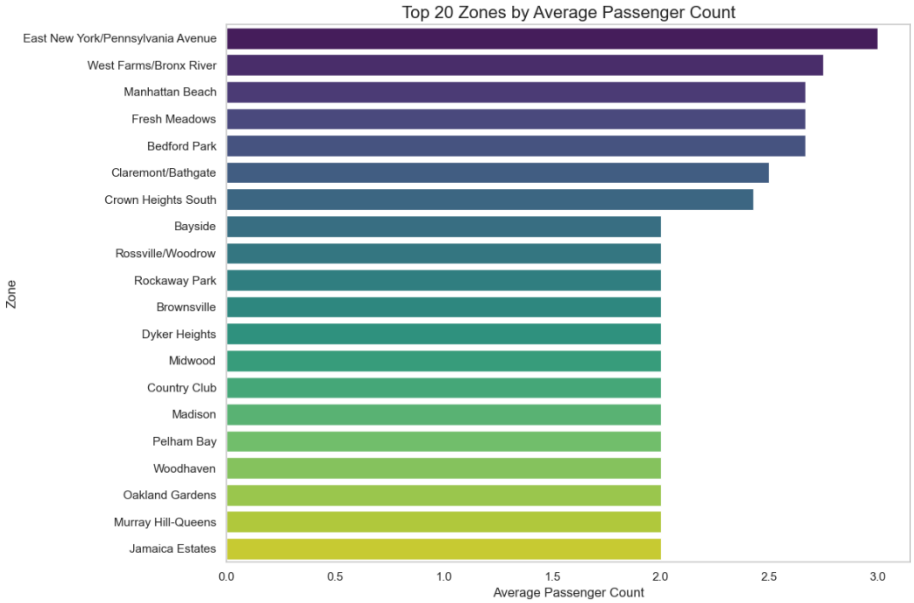
The data reveals that shorter trips (2 to 5 miles) receive significantly higher tip percentages than longer trips. Tipping behavior varies considerably by passenger count, with 1 and 5-6 passengers tipping most generously while 4 passengers tip the least. Tips also fluctuate throughout the day, showing a general upward trend with peaks around hours 20-21 (8-9 PM), suggesting evening riders are more generous tippers than daytime riders.

3.2.14. Analyse the trends in passenger count

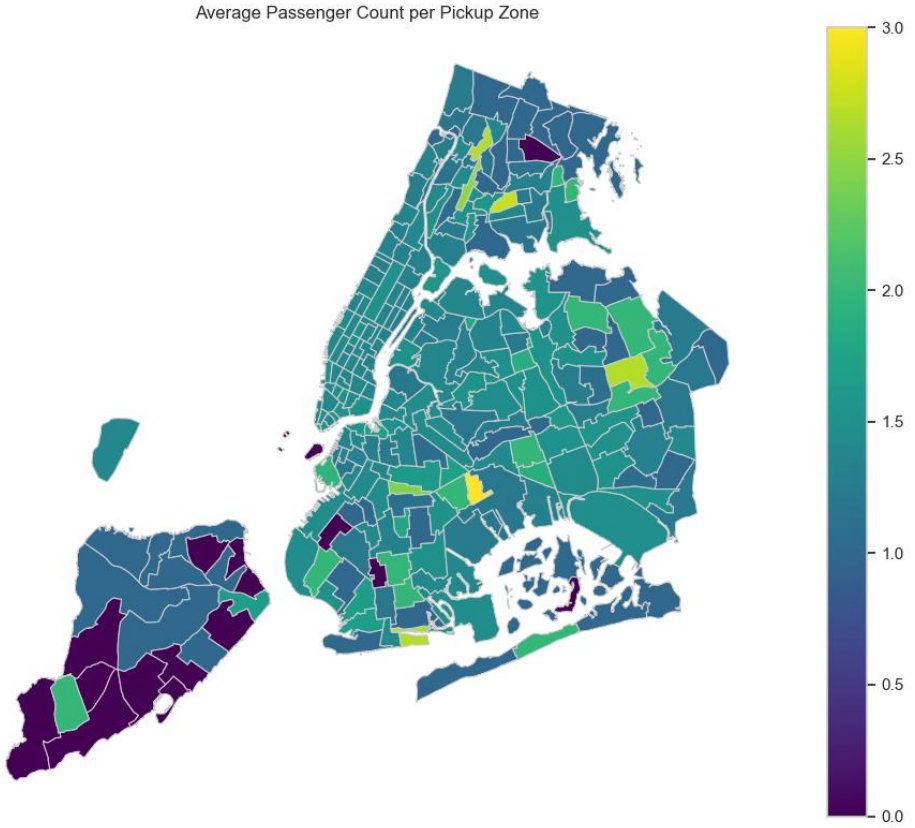


The graph shows Friday has the highest average passenger count, peaking at approximately 1.55 passengers around hour 15 (3 PM). Weekdays (Monday-Thursday) generally have lower passenger counts, particularly during midday hours (5-10). All days show a similar pattern with drops in early morning hours and increases in late afternoon/evening, suggesting passenger grouping behavior follows consistent daily rhythms with weekend variations.

3.2.15. Analyse the variation of passenger counts across zones

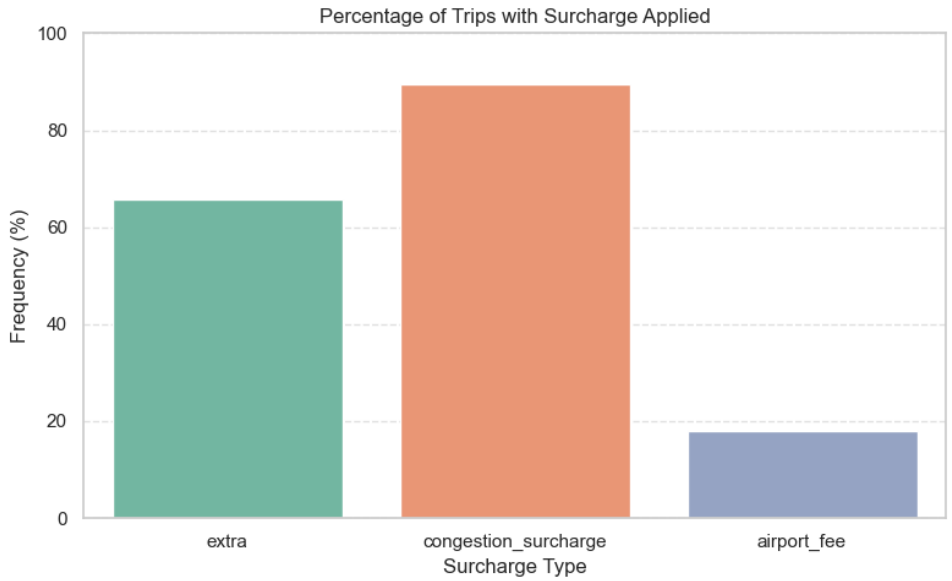


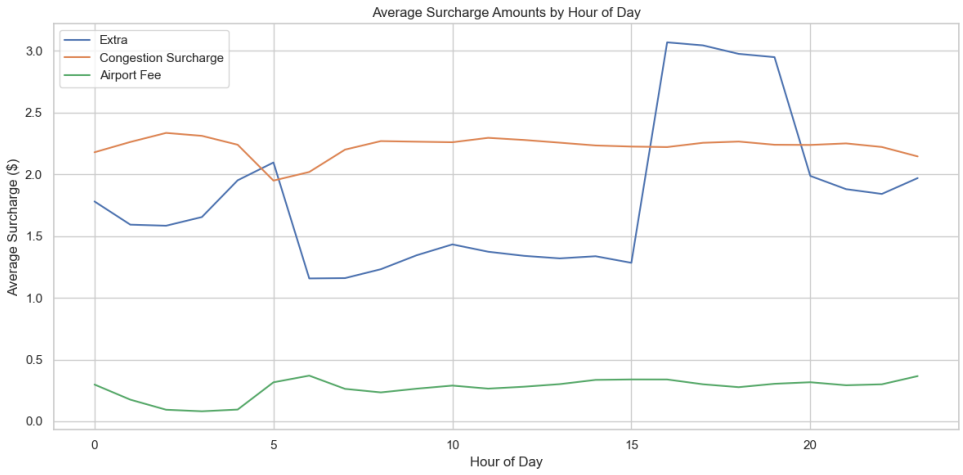
East New York/Pennsylvania Avenue leads with the highest average passenger count (nearly 3 passengers per ride), followed closely by West Farms/Bronx River. The top five zones (East New York, West Farms, Manhattan Beach, Fresh Meadows, and Bedford Park) all maintain averages above 2.5 passengers per ride, significantly higher than zones ranking 10-20 which cluster around 2.0 passengers. This indicates certain neighborhoods consistently have larger group travel behaviors, potentially due to factors like residential density, family demographics, or limited transit alternatives.



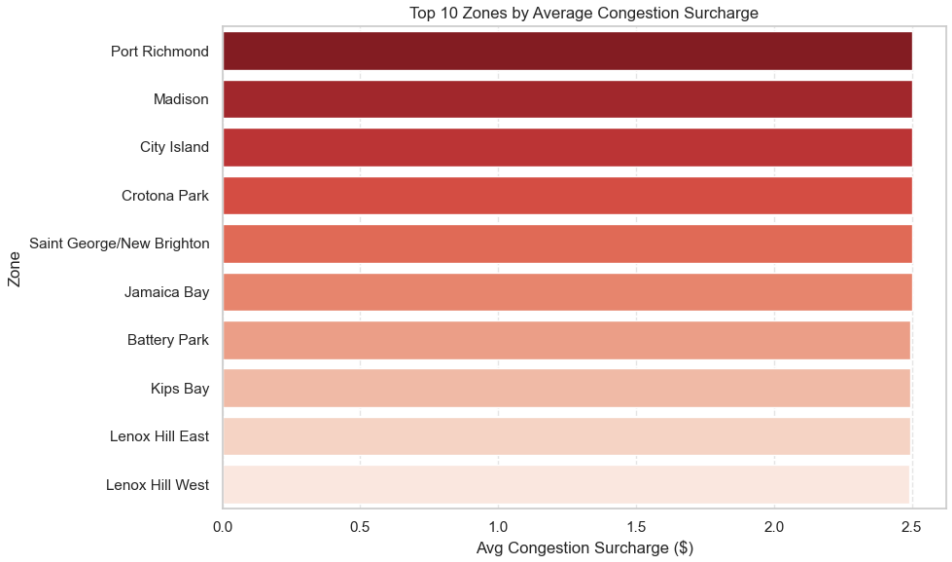
The map reveals significant geographical variation in average passenger counts across New York City zones. Several scattered bright yellow areas (primarily in eastern Brooklyn and parts of Queens) show the highest passenger counts (2.5-3.0 per ride), while Staten Island and northern Manhattan display the lowest counts (dark purple/blue, often below 0.5). The majority of Manhattan and central Brooklyn maintain moderate passenger counts (teal color, approximately 1.5 passengers per pickup), suggesting different travel behaviors based on neighborhood characteristics.

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





Congestion surcharges are the most frequently applied fee type, affecting nearly 90% of trips, while airport fees are the least common at approximately 18% of trips. The "extra" surcharge applies to about 65% of rides. Looking at surcharge amounts by time of day, the "extra" surcharge shows dramatic hourly fluctuations, peaking significantly between hours 16-19 (4-7 PM) at around $3. Congestion surcharges remain relatively stable throughout the day (consistently $2.00-2.30), while airport fees are consistently the lowest charge (under $0.50), with minimal variation across different hours.



The data reveals that all top 10 zones have nearly identical congestion surcharges of approximately $2.50, with Port Richmond leading marginally. The "extra" surcharge shows dramatic time-of-day variation, spiking significantly during evening rush hours (4-7 PM) to over $3.00, while dropping to around $1.20 during mid-morning hours. Congestion surcharges are the most commonly applied fee (approximately 90% of trips), followed by "extra" surcharges (65%), with airport fees being the least frequent (less than 20%).

4. Conclusions

4.1. Final Insights and Recommendations

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

Smart Scheduling -> Put more taxis on the road during busy times: weekday mornings (7-10AM), evenings (4-8PM), and weekend nights Reduce taxis during slow periods to save money
Better Route Planning -> Use traffic updates to help drivers avoid busy areas like Times Square Position drivers where rides will be needed before demand increases
Strategic Coverage -> Focus on busy pickup spots like Midtown and airports Add more taxis in growing areas like Brooklyn during evenings
Data Cleanup -> Create better checks for unusual fares and trip distances Look closely at areas with strange patterns to fix problems
Tailored Services -> Offer shared rides during office hours when many people travel similar routes Provide premium options for late-night downtown customers

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.

Taxi Positioning Strategy Made Simple
Best Places for Taxis

Midtown Manhattan

When: Weekday mornings (7-10AM) and evenings (4-8PM), especially Tuesday-Thursday
Why: Office workers, tourists, regular commuters
Do: Have extra taxis ready during office rush hours

Airports (JFK, LGA)

When: Every day, especially 5-11PM on weekends and holidays
Why: Travelers with luggage need taxis
Do: Match taxi availability with flight arrivals

Downtown and Financial District

When: Weekday mornings (7-9AM) and evenings (5-7PM)
Why: Business people taking short to medium trips
Do: Focus on start and end of business hours

Brooklyn and Queens

When: Evenings and weekends
Why: People going home or out for fun
Do: Move taxis from Manhattan to these areas at night and on weekends

Outer Boroughs (Bronx, Staten Island)

When: Early mornings and weekends
Why: Fewer transportation options
Do: Offer special booking options during quieter times

Time Tips

Weekdays: Focus on business areas and commuter routes
Weekends: Cover entertainment spots, airports, and neighborhoods

Seasonal Changes -> Add more taxis in tourist areas from May-September and during holiday shopping seasons

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

Simple Pricing Strategy

1. Match Prices to Different Customer Types
   Work Travelers: These people care more about being on time than saving a few dollars. Offer reliable service during weekday commute hours. Weekend Fun-Seekers: These customers watch their spending more closely. Give them group discount options. Tourists: These riders want simple pricing and safe service from airports and landmarks. Offer clear flat rates.
2. Reward Regular Riders
   Offer monthly ride passes for frequent customers Give discounts or cashback to people who use the service often
3. Smarter Price Adjustments
   Use traffic, weather, and other data to spot when rides will be difficult Add small price increases ($1-3) only when necessary, not just because it's busy
4. Simple Flat Rates Between Popular Places
   Set fixed prices for common routes like airport trips Customers prefer knowing the exact fare before riding
5. Stay Competitive Without Losing Money
   Keep track of what Uber, Lyft and others are charging Price just below competitors when possible, but not so low that you lose money.
   This approach keeps prices fair while still making profit, keeps customers coming back, and stays competitive without relying on surprise price increases.