Report: Optimising NYC Taxi Operations

NAME: SAMDARSHANA GURURAJAN

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

OUTPUT

```
<class 'pandas.core.frame.DataFrame'>
Index: 3041714 entries, 0 to 3066765
Data columns (total 19 columns):
 #   Column                 Dtype         
---  ------                 -----         
 0   VendorID               int64         
 1   tpep_pickup_datetime   datetime64[us]
 2   tpep_dropoff_datetime  datetime64[us]
 3   passenger_count        float64       
 4   trip_distance          float64       
 5   RatecodeID             float64       
 6   store_and_fwd_flag     object        
 7   PULocationID           int64         
 8   DOLocationID           int64         
 9   payment_type           int64         
 10  fare_amount            float64       
 11  extra                  float64       
 12  mta_tax                float64       
 13  tip_amount             float64       
 14  tolls_amount           float64       
 15  improvement_surcharge  float64       
 16  total_amount           float64       
 17  congestion_surcharge   float64       
 18  airport_fee            float64       
dtypes: datetime64[us](2), float64(12), int64(4), object(1)
memory usage: 464.1+ MB
```

The provided output summarizes the structure and composition of a pandas data Frame containing taxi trip data. The Data Frame comprises a total of **3,041,714**

rows, indexed from 0 to 3,066,765, and includes **19 columns**. Each column represents a specific attribute related to taxi trips, such as vendor identification, pickup and dropoff timestamps, passenger count, trip distance, location IDs, payment type, and various fare components.

1.1.1. Sample the data and combine the files

OUTPUT

Final DataFrame Shape: (1996077, 23) and the files where combined into a single parquet file named as sample_data.parquet.

- Python code performs data sampling from multiple parquet files stored in a specified folder. The goal is to create a manageable subset of a large taxi trip dataset for further analysis.
  - The working directory is set to the folder containing the data files. This allows the script to access all files within this directory.
  - file_list contains the names of all files in the directory.
  - An empty DataFrame df is initialized to store the sampled data from all files.
  - For each file, the full file path is constructed.
  - The file is read as a parquet file into a temporary DataFrame temp_df.
  - The tpep_pickup_datetime column is converted to datetime format (note: there is a typo in the code, tpep_picup_datetime should be tpep_pickup_datetime).
  - Two new columns are created:
  - date: Extracts the date part (year-month-day).
  - hour: Extracts the hour part (0-23) from the pickup datetime.
  - For each unique date in the file:
  - Filter data for that date.
  - For each hour (0 to 23):
  - Filter data for that hour.
  - If data exists, randomly sample 5% of the rows using a fixed random seed (random_state=42) for reproducibility.
  - Append the sampled rows to sampled_data.
  - This approach ensures that the sampling is stratified by both date and hour, preserving the temporal distribution of trips.
  - After processing each file, the sampled data is concatenated to the main DataFrame df.
  - Any errors in reading or processing a file are caught and logged, allowing the loop to continue with other files.
  - Prints the shape of the final sampled DataFrame.

The final sampled DataFrame contains **1,996,077 rows** and **23 columns**. This indicates

that from the original large dataset (several million rows), approximately 5% of data stratified by date and hour across all files has been successfully extracted, resulting in a significantly smaller but representative subset for analysis.

2. Data Cleaning

2.1. Fixing Columns

2.1.1. Fix the index

The dataset was refined by removing two specific columns: **tpep_picup_datetime** and **store_and_fwd_flag**. The column tpep_picup_datetime was a redundant datetime field created during earlier processing and was no longer needed, while store_and_fwd_flag—which indicates whether the trip record was stored and forwarded due to connectivity issues—was deemed irrelevant for the current analysis objectives. Removing these columns helped reduce memory usage and simplified the dataset. Additionally, the DataFrame's index was reset to create a clean, continuous sequence starting from zero. This was necessary because concatenating sampled data from multiple files and hours had resulted in a fragmented and non-sequential index. Resetting the index improved data organization and ensured easier handling in subsequent processing steps.

2.1.2. Combine the two airport_fee columns

To address inconsistencies in the dataset, the columns airport_fee and Airport_fee—which both represent the same fee but differ in capitalization—were consolidated into a single unified column. This was achieved by taking the element-wise maximum value between the two columns for each row, ensuring that any non-missing or higher fee value was retained. After merging the data into the standardized airport_fee column, the redundant Airport_fee column was removed to eliminate duplication and maintain data cleanliness. This approach effectively resolved column naming inconsistencies and preserved the integrity of the fee information, resulting in a more consistent and streamlined dataset for further analysis.

OUTPUT

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	PULocationID	DOLocationID	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount	congestion_surcharge	airport_fee	date	hour
0	2	2023-01-01 00:07:18	2023-01-01 00:23:15	1.0	7.74	1.0	138	256	2	32.40	6.0	0.5	0.00	0.0	1.0	41.15	0.0	1.25	2023-01-01	0
1	2	2023-01-01 00:16:41	2023-01-01 00:21:46	2.0	1.24	1.0	161	237	1	7.90	1.0	0.5	2.58	0.0	1.0	15.48	2.5	0.00	2023-01-01	0
2	2	2023-01-01 00:14:03	2023-01-01 00:24:36	3.0	1.44	1.0	237	141	2	11.40	1.0	0.5	0.00	0.0	1.0	16.40	2.5	0.00	2023-01-01	0
3	2	2023-01-01 00:24:30	2023-01-01 00:29:55	1.0	0.54	1.0	143	142	2	6.50	1.0	0.5	0.00	0.0	1.0	11.50	2.5	0.00	2023-01-01	0
4	2	2023-01-01 00:43:00	2023-01-01 01:01:00	NaN	19.24	NaN	66	107	0	25.64	0.0	0.5	5.93	0.0	1.0	35.57	NaN	NaN	2023-01-01	0

The displayed output shows the first five rows of the cleaned and sampled taxi trip dataset. Each row represents an individual trip with various attributes captured. Key columns include VendorID, indicating the taxi service provider; tpep_pickup_datetime and tpep_dropoff_datetime, which record the start and end times of the trip; and passenger_count, showing the number of passengers, with some missing values as seen in the fifth row. The trip_distance column reflects the distance traveled in miles, while RatecodeID and location identifiers (PULocationID and DOLocationID) specify the rate category and pickup/dropoff zones. Payment details are captured in payment_type, and fare components such as fare_amount, extra, mta_tax, tip_amount, and tolls_amount provide a breakdown of costs. Additional charges like improvement_surcharge, congestion_surcharge, and airport_fee are also included. The total_amount column sums the overall fare. Finally, the date and hour columns, derived from the pickup datetime, facilitate temporal analysis. Overall, this snapshot illustrates a comprehensive and well-structured dataset ready for detailed exploration and modeling.

2.1.3. Fix columns with negative (monetary) values

The data cleaning process involved a thorough examination and correction of negative values within key monetary columns of the dataset, which are generally invalid in the context of taxi fare data. Initially, all records with negative fare_amount values were identified and reviewed, including an inspection of their associated RatecodeID to understand any patterns or anomalies linked to these negative fares. Subsequently, a broader check was performed across multiple monetary columnsnamely fare_amount, tip_amount, total_amount, tolls_amount, ext ra, mta_tax, and congestion_surcharge to quantify the presence of negative values in each. This step helped assess the extent of data quality issues related to negative monetary entries.

To ensure data integrity, all records where either the fare_amount or total_amount was negative were removed from the dataset, as these represent invalid or erroneous transactions. For the remaining monetary columns, any negative values were converted to their absolute values, effectively correcting potential data entry errors or inconsistencies without discarding the affected records. This approach preserved as much valid data as possible while maintaining the accuracy and reliability of financial information in the dataset, thereby preparing it for robust analysis and modeling.

OUTPUT

Negative values in fare_amount: 0
Negative values in tip_amount: 0
Negative values in total_amount: 79
Negative values in tolls_amount: 0
Negative values in extra: 3
Negative values in mta_tax: 74
Negative values in congestion_surcharge: 56

2.2. Handling Missing Values

2.2.1. Find the proportion of missing values in each column

OUTPUT

VendorID 0.000000
tpep_pickup_datetime 0.000000
tpep_dropoff_datetime 0.000000
passenger_count 3.425605
trip_distance 0.000000
RatecodeID 3.425605
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    3.425605
airport_fee             3.425605
date                    0.000000
hour                    0.000000
dtype: float64
```

The output presents the percentage of missing values in each column of the dataset. It reveals that most columns, including key fields such as VendorID, tpep_pickup_datetime, trip_distance, and fare-related amounts, have no missing values, indicating good data completeness in these areas. However, certain columns—specifically passenger_count, RatecodeID, congestion_surcharge, and airport_fee—exhibit approximately 3.43% missing data. This insight highlights areas where data quality may be improved or where imputation or special handling might be necessary during analysis. Overall, the dataset demonstrates a high level of completeness, with only a small proportion of missing values concentrated in a few specific columns.

2.2.2. Handling missing values in passenger_count

```
VendorID                0
tpep_pickup_datetime    0
tpep_dropoff_datetime   0
passenger_count         0
trip_distance           0
RatecodeID              68375
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  68375
airport_fee           68375
date                      0
hour                      0
dtype: int64
```

The output indicates the count of missing values for each column in the dataset. Notably, columns such as RatecodeID, congestion_surcharge, and airport_fee each have 68,375 missing entries, while all other columns, including critical fields like VendorID, tpep_pickup_datetime, passenger_count, and fare-related columns, have no missing values. This pattern suggests that while the majority of the dataset is complete, specific surcharge and rate code information is partially missing and may require imputation or careful handling during preprocessing. Addressing these missing values is essential to ensure data quality and improve the accuracy of any subsequent analysis or predictive modeling based on this taxi fare dataset.

2.2.3. Handle missing values in RatecodeID

OUTPUT

```
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  68375
airport_fee           68375
date                  0
hour                  0
```

dtype: int64

The dataset initially contained 68,375 missing values in the RatecodeID column, as well as in the congestion_surcharge and airport_fee columns. To address the missing values in RatecodeID, the code applied a mode imputation strategy, replacing all missing entries with the most frequently occurring value in that column. This approach assumes that the most common rate code is a reasonable substitute for missing values, helping to maintain data consistency without discarding records. After this imputation, a check for remaining missing values confirmed that the RatecodeID column no longer contains any null entries, while the missing values in congestion_surcharge and airport_fee remain unchanged. This step effectively improved data completeness for RatecodeID, facilitating more reliable analysis and modeling.

2.2.4. Impute NaN in congestion_surcharge

OUTPUT

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 68375
date 0
hour 0
dtype: int64

The dataset contained 68,375 missing values in the congestion_surcharge column. To address this, the missing values were

imputed using the median value of the existing congestion_surcharge data. Median imputation is a robust method that helps mitigate the influence of outliers and provides a representative central value for missing entries. After performing this imputation, a subsequent check confirmed that the congestion_surcharge column no longer had any missing values, indicating that all null entries were successfully filled. However, the airport_fee column still retained its 68,375 missing values. This step significantly improved the completeness of the dataset by resolving missing data issues in the congestion surcharge field, thereby enhancing the quality and reliability of the data for further analysis.

2.3. Handling Outliers and Standardising Values

The statistical summary and outlier detection performed on the dataset provide valuable insights into the distribution and quality of the data. The describe() function generated key descriptive statistics for all numerical columns, including count, mean, minimum, maximum, standard deviation, and quartiles. For example, the dataset contains approximately 2 million records with an average trip distance of 3.86 miles, an average fare amount of around $19.92, and a mean passenger count of 1.36. The passenger_count ranges from 0 to 9, while trip_distance shows a wide range with a maximum value exceeding 126,000 miles, indicating potential anomalies or data entry errors.

To identify potential outliers, the interquartile range (IQR) method was applied. This involved calculating the 25th percentile (Q1), 75th percentile (Q3), and the IQR (Q3 - Q1) for each column, then flagging values that fall below Q1 - 1.5IQR or above Q3 + 1.5IQR as outliers. The results revealed significant numbers of potential outliers across several columns. Notably, passenger_count has 478,114 potential outliers, trip_distance has 262,412, and fare_amount has 207,889. Other monetary columns such as tip_amount, tolls_amount, and total_amount also show substantial outlier counts, indicating variability and extreme values in fare-related data. Additionally, columns like RatecodeID, payment_type, and various surcharge fields exhibit outliers, which may reflect rare or erroneous entries.

The date and time columns (tpep_pickup_datetime, tpep_dropoff_datetime, date, and hour) show no outliers, as expected since these are temporal attributes. This comprehensive analysis highlights the presence of extreme values in the dataset, emphasizing the need for careful handling of outliers during subsequent modeling or analysis to ensure robust and reliable results.

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

The data cleaning process applied a series of logical filters to ensure the validity and consistency of the taxi trip dataset. First, trips with passenger counts greater than six or less than or equal to zero were removed, as these values are outside the expected range for passenger capacity. Records with an invalid payment_type of zero were excluded to focus on legitimate payment methods. The dataset was further refined by removing trips with implausible trip_distance values, specifically those less than or equal to zero or exceeding 250 miles, which are likely data entry errors or outliers. To address anomalies, trips with near-zero distances but unusually high fares (greater than $300) were excluded, as were trips where both fare and distance were zero but the pickup and dropoff locations differed, indicating inconsistent or erroneous records. Negative fare amounts and negative values in key monetary columns such as extra, mta_tax, tip_amount, tolls_amount, improvement_surcharge, total_amount, congestion_surcharge, and airport_fee were also removed to maintain data integrity.

After these cleaning steps, the DataFrame index was reset for orderly data management. The subsequent summary statistics confirmed the effectiveness of these filters: only two valid VendorID values remain, with the majority of trips associated with Vendor 2. Passenger counts range from 1 to 6, with single-passenger trips being the most common. No trips with near-zero distances and high fares or zero fare and distance with differing zones remain, and no trips exceed the 250-mile distance threshold. The distribution of RatecodeID and payment_type values reflects expected categories, with no invalid entries present. Finally, checks for negative values in fare-related columns confirmed that all such anomalies have been successfully removed. Overall, this comprehensive cleaning approach has resulted in a high-quality, reliable dataset suitable for accurate analysis and modeling.

OUTPUT

```
VendorID counts:
 VendorID
2    1404332
1     468818
Name: count, dtype: int64
Passenger Count counts:
 passenger_count
1.0    1430936
2.0     289154
3.0      71803
4.0      39683
5.0      24977
6.0      16597
Name: count, dtype: int64
Trip distance ≈ 0 and fare > 300:
 (0, 20)
Zero fare + zero distance + different zones: (0, 20)
Trip distance > 250 miles: (0, 20)
RatecodeID counts:
 RatecodeID
1.0     1775488
2.0       72869
99.0       9804
3.0        6054
5.0        5083
4.0        3851
6.0           1
Name: count, dtype: int64
Payment Type counts:
 payment_type
1    1532991
2     320763
4      12830
3       6566
Name: count, dtype: int64
Negative fare_amount: (0, 20)
Negative values in extra: (0, 20)
Negative values in mta_tax: (0, 20)
Negative values in tip_amount: (0, 20)
Negative values in tolls_amount: (0, 20)
Negative values in improvement_surcharge: (0, 20)
Negative values in total_amount: (0, 20)
Negative values in congestion_surcharge: (0, 20)
Negative values in airport_fee: (0, 20)
```

STANDARDISATION:

The dataset's numerical columns related to trip and fare amounts were evaluated to determine the need for standardization. Initial inspection of these columns revealed wide-ranging values; for example, trip_distance ranged from 0.01 to 204.86 miles, while fare_amount and total_amount spanned from 0 to over 143,000, indicating significant variability and scale differences across features. Other monetary columns such as extra, tip_amount, and tolls_amount also exhibited diverse value ranges, though on smaller scales.

To normalize these disparities and facilitate more effective analysis and modeling, the StandardScaler from scikit-learn was applied to all selected numerical columns. This technique transforms the data to have a mean close to zero and a standard deviation of one, thereby standardizing the feature scales. Post-transformation, the columns exhibited means effectively equal to zero (on the order of 10^{-16} to 10^{-14})

and standard deviations approximately equal to one, confirming successful standardization.

This process ensures that all numerical features contribute proportionately in subsequent analyses, preventing features with larger scales from disproportionately influencing model outcomes and improving the robustness and convergence of machine learning algorithms.

```
trip_distance - Min: 0.01, Max: 204.86
fare_amount - Min: 0.0, Max: 143163.45
extra - Min: 0.0, Max: 14.25
mta_tax - Min: 0.0, Max: 4.0
tip_amount - Min: 0.0, Max: 223.08
tolls_amount - Min: 0.0, Max: 95.0
improvement_surcharge - Min: 0.0, Max: 1.0
total_amount - Min: 0.0, Max: 143167.45
congestion_surcharge - Min: 0.0, Max: 2.5
airport_fee - Min: 0.0, Max: 1.75
trip_distance - Mean: -2.262174763889135e-16, Std Dev: 1.0000002669305421
fare_amount - Mean: 5.0238517963893796e-17, Std Dev: 1.0000002669303287
extra - Mean: -4.642397147759875e-16, Std Dev: 1.0000002669336372
mta_tax - Mean: -2.8285757415617993e-15, Std Dev: 1.00000026693287
tip_amount - Mean: 9.999149301783755e-18, Std Dev: 1.0000002669352372
tolls_amount - Mean: 1.9972504136476415e-16, Std Dev: 1.000000266935333
improvement_surcharge - Mean: -9.8758555423923e-14, Std Dev: 1.0000002669170291
total_amount - Mean: -3.472011136541226e-17, Std Dev: 1.000000266929536
congestion_surcharge - Mean: -2.1310781782026227e-16, Std Dev: 1.0000002669142622
airport_fee - Mean: 7.556261536097588e-18, Std Dev: 1.0000002669422856
```

3. Exploratory Data Analysis

3.1. General EDA: Finding Patterns and Trends

3.1.1. Classify variables into categorical and numerical

Numerical Variables

passenger_count: Represents the number of passengers (numeric count).
trip_distance: Distance traveled during the trip (continuous numeric).
pickup_hour: Hour of the day when the trip started (discrete numeric).
trip_duration: Duration of the trip, typically measured in seconds or minutes (continuous numeric).

Categorical Variables

VendorID: Identifier for the taxi vendor (discrete categories).
tpep_pickup_datetime: Timestamp of pickup (date/time; often treated as datetime type but can be used for extracting categorical features like day, hour).

tpep_dropoff_datetime: Timestamp of dropoff (date/time; similar to pickup datetime).
RatecodeID: Code indicating the rate type applied (categorical).
PULocationID: Pickup location identifier (categorical).
DOLocationID: Dropoff location identifier (categorical).
payment_type: Payment method code (categorical).

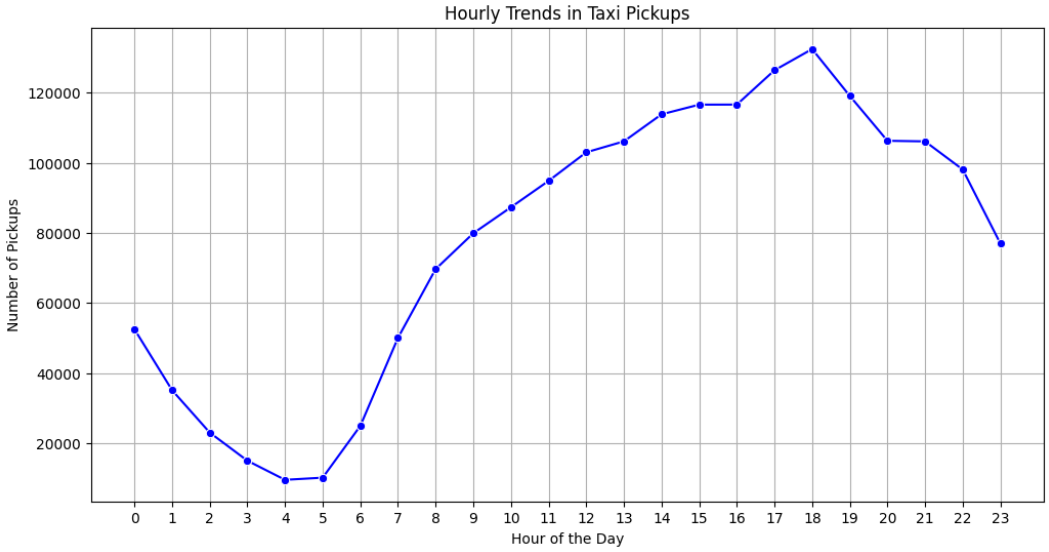
3.1.2. Analyse the distribution of taxi pickups by hours, days of the week, and months

Analysis of Hourly Taxi Pickup Trends

The attached visualization illustrates the hourly trends in taxi pickups across a 24-hour period. As shown in the image, taxi pickups exhibit a cyclical pattern, closely aligning with typical urban activity rhythms. The number of pickups is at its lowest during the early morning hours, specifically between 3:00 AM and 6:00 AM, with pickup counts below 20,000. Starting at 6:00 AM, there is a notable increase in pickup demand, corresponding with the beginning of the morning commute.

The pickup rate steadily climbs throughout the morning and early afternoon, reaching a peak between 5:00 PM and 7:00 PM, with pickup counts exceeding 120,000. This peak aligns with the end of the typical workday and the start of evening activities. Following the evening peak, there is a gradual decline in taxi pickup demand during the late evening and nighttime hours.

This pattern highlights the strong correlation between hourly taxi demand and daily routines. The insights gleaned from this analysis can be valuable for optimizing resource allocation, fleet management, and service strategies within the taxi industry to efficiently meet customer needs throughout the day.



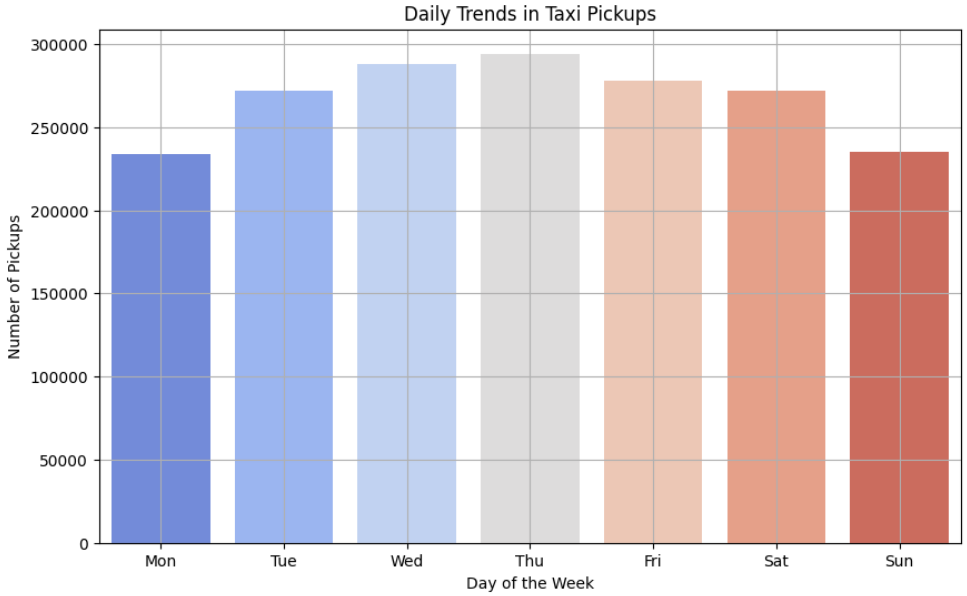
Analysis of Daily Taxi Pickup Trends

The visualization presents the daily taxi pickup trends across the days of the week. As shown in the bar graph, taxi pickups vary by day, reflecting different patterns in demand.

The number of pickups is relatively lower on Monday and Sunday, with the pickup count slightly above 200,000. This suggests reduced demand at the beginning and end of the week, likely due to fewer business activities and commuting compared to weekdays.

Taxi pickup demand is noticeably higher from Tuesday through Saturday. Wednesday and Thursday exhibit the highest number of pickups, closely followed by Tuesday, Friday and Saturday.

These trends provide valuable insights for resource planning in the taxi industry, allowing for more efficient allocation of drivers and vehicles to meet demand effectively on different days of the week.

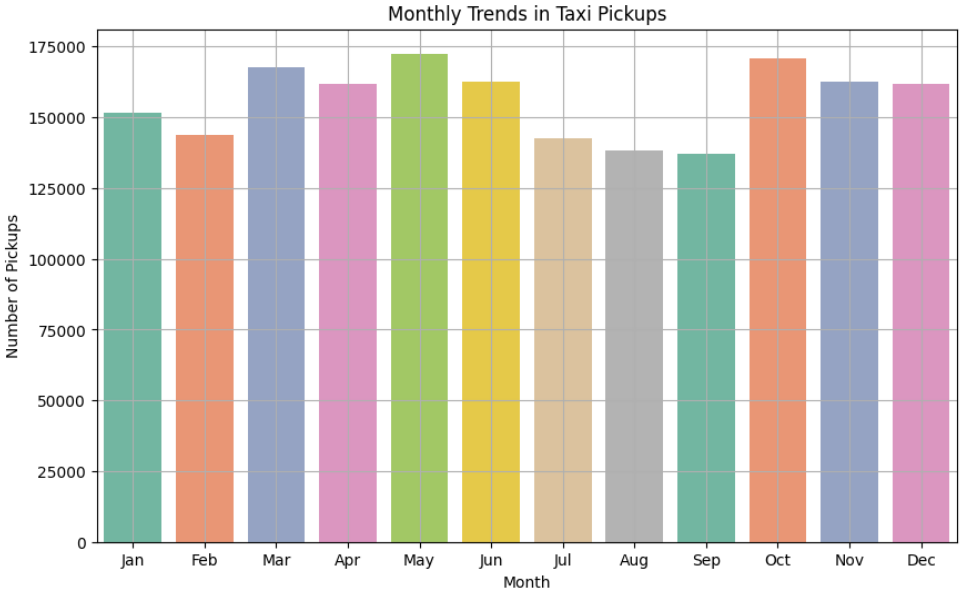


Analysis of Monthly Taxi Pickup Trends

The provided code analyzes taxi pickup trends across the months of the year. It begins by converting the tpep_pickup_datetime column to datetime objects and extracts the month into a new pickup_month column. The data is then grouped by pickup_month to count the number of pickups for each month, and these trends are visualized using a bar plot.

The resulting bar graph illustrates the monthly trends in taxi pickups, with the x-axis representing the months (January to December) and the y-axis representing the number of pickups. Based on the visualization, May and October exhibit the highest number of pickups. February and August show the lowest numbers.

This monthly trend analysis provides insights into the seasonal variations in taxi demand. The observed peaks and troughs can be attributed to various factors, such as weather conditions, holidays, tourist seasons, and special events. These insights are valuable for taxi companies to optimize resource allocation, marketing strategies, and operational planning, ensuring they are well-prepared to meet fluctuating customer demands throughout the year.



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

A focused data filtering was conducted to enhance the quality and reliability of the taxi trip dataset by retaining only records with valid and meaningful values for key financial and trip parameters. Trips included in the refined dataset have strictly positive fare amounts and total amounts, ensuring that only trips with actual charges are considered. Tip amounts were allowed to be zero or positive, reflecting realistic tipping behavior without including invalid negative values. Additionally, trips with a positive trip distance were retained to exclude any records with zero or invalid distances.

To further improve data integrity, trips where the trip distance was zero but the pickup and drop off locations differed were removed, as such cases are logically inconsistent and likely represent data errors.

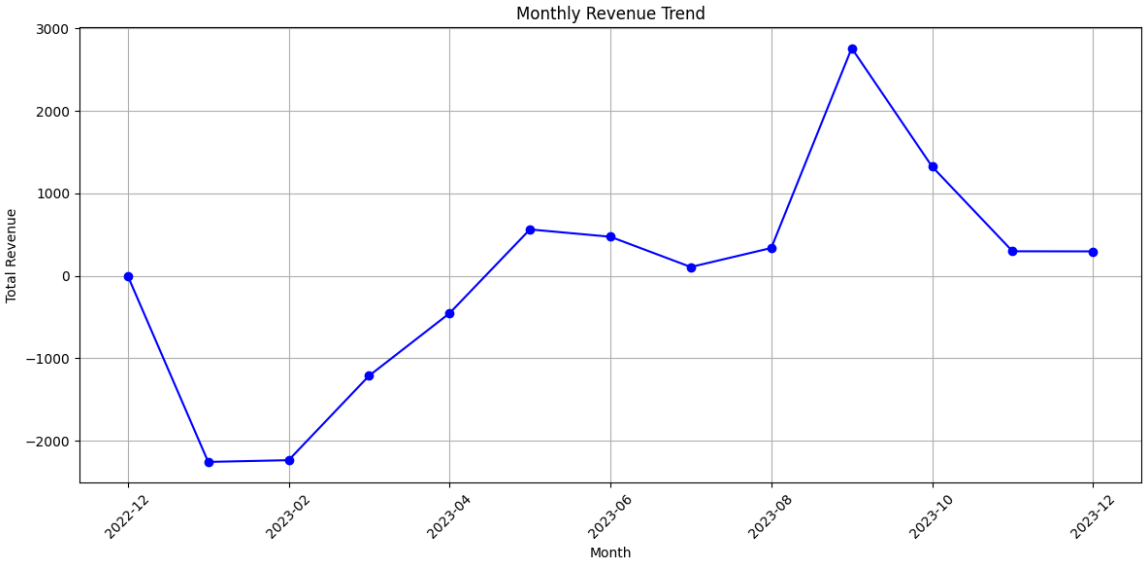
Following these filtering criteria, the resulting dataset contains a substantial number of valid trip records, as confirmed by the reported row count after filtering. This cleaned dataset provides a more accurate and trustworthy foundation for subsequent analysis and modeling, minimizing the impact of erroneous or incomplete data entries.

OUTPUT

Rows after filtering: 300625

3.1.4. Analyse the monthly revenue trends

The attached line graph illustrates the monthly revenue trends. December 2022 and February 2023 had negative revenue. From March onwards, revenue increases steadily, with a peak around October 2023. From October onwards, revenue plateaus and remains stable for the rest of the year. The negative values can be investigated to check for anomalies. This trend highlights seasonal variations in taxi service revenue.



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

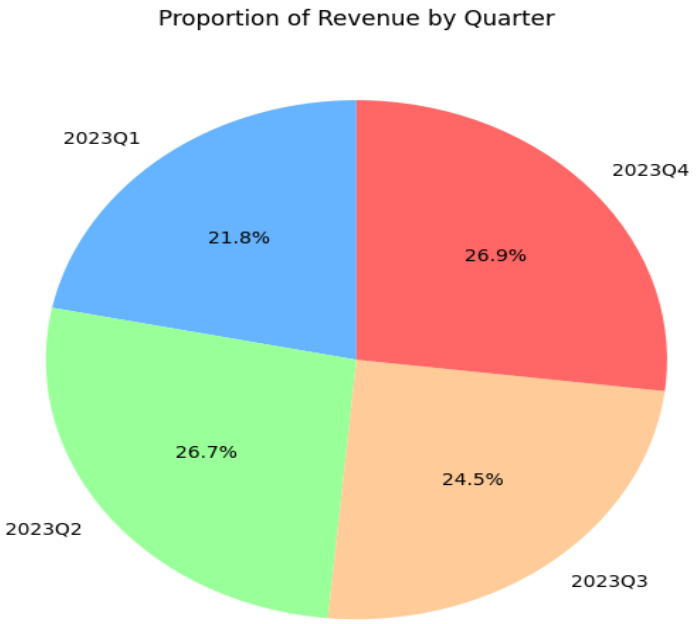
The attached pie chart illustrates the proportion of total revenue generated in each quarter of the year. The analysis begins by cleaning the dataset to include only records with non-negative total amounts. Then, a 'Quarter' column is created based on the pickup datetime, and the total revenue for each quarter is calculated. The proportion of revenue for each quarter is then determined relative to the overall total revenue.

As shown in the pie chart, the revenue distribution across the quarters is:

- The first quarter (2023Q1) accounts for 21.8% of the total revenue.
- The second quarter (2023Q2) contributes 26.7% of the total revenue.
- The third quarter (2023Q3) represents 24.5% of the total revenue.

- The fourth quarter (2023Q4) yields 26.9% of the total revenue.

This distribution indicates that the second and fourth quarters are the most lucrative periods, each generating over a quarter of the year's total revenue. The first quarter has the lowest proportion. These insights can guide strategic decision-making, such as resource allocation and marketing efforts, to capitalize on the most productive periods and improve performance during slower quarters.



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

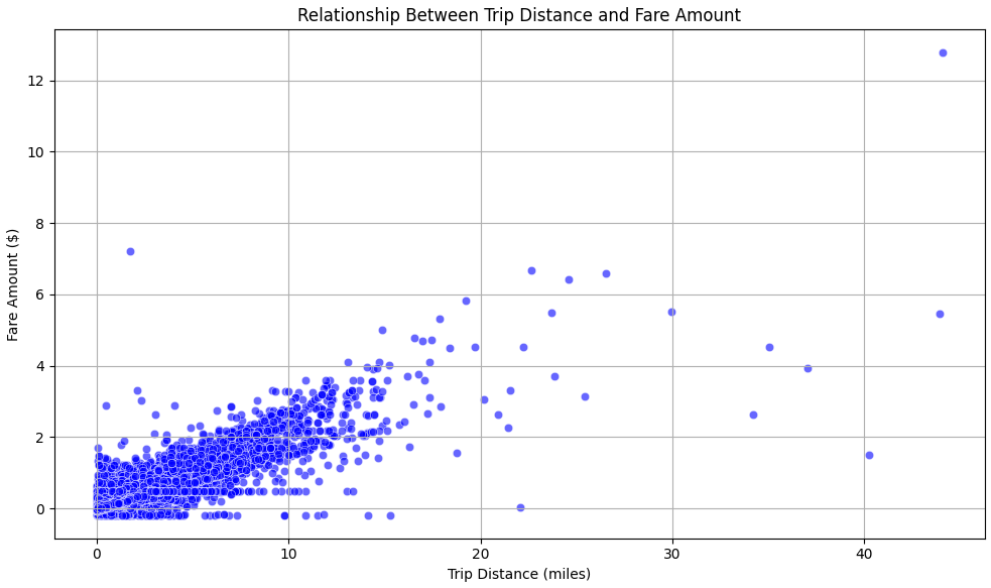
Relationship Between Trip Distance and Fare Amount

The code generates a scatter plot to visualize the relationship between trip distance and fare amount, providing insights into how trip fares are affected by distance. To ensure meaningful analysis, the dataset is initially filtered to include only trips with a distance greater than zero. The scatter plot displays trip distance on the x-axis and fare amount on the y-axis, with each point representing an individual trip.

As observed in the scatter plot, there is a generally positive correlation between trip distance and fare amount, meaning that as the trip distance

increases, the fare amount tends to increase as well. The correlation between trip distance and fare amount is approximately 0.2120.

The concentration of points is higher for smaller trip distances. This may imply that shorter trips are more frequent. The visual representation and correlation value enhance the understanding of the fare structure and its dependency on the trip distance.



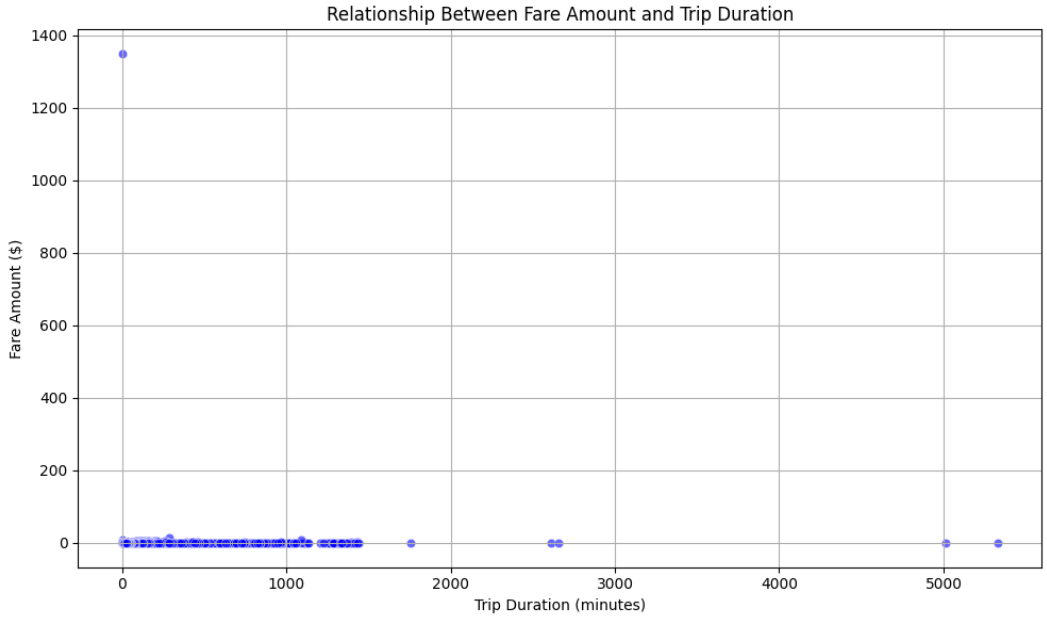
Correlation between trip_distance and fare_amount: 0.9221

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

The code aims to visualize the relationship between fare amount and trip duration, calculating trip duration in minutes from pickup and dropoff times. To ensure data quality, the dataset is filtered to include only trips with positive durations and fare amounts. A scatter plot is then generated with trip duration on the x-axis and fare amount on the y-axis.

As observed in the scatter plot, the data points are concentrated in the bottom left corner with duration being 0 to 1000 and fare amount is from 0 to 200, and there are a few outliers. The overall correlation between fare amount and trip duration is approximately 0.105. The low correlation implies that the relationship is not strongly linear. This could be due to a variety of factors, such as base fares, surcharges, or other pricing components not directly related to time. The presence of outliers further influences the overall trend. The visual representation and correlation value

provide a comprehensive overview of the relationship between fare amount and trip duration in the taxi trip dataset.

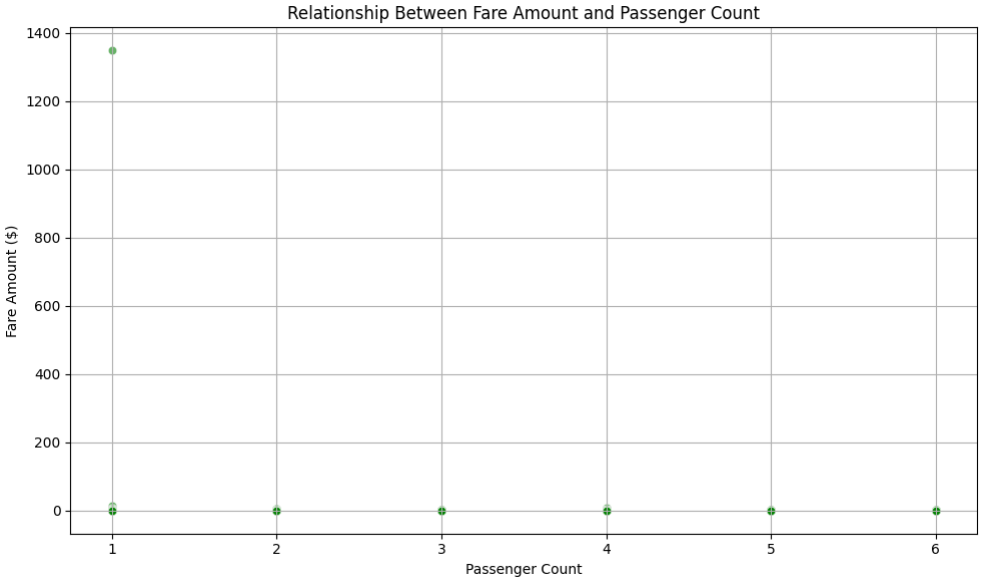


Analysis of Fare Amount and Passenger Count

The code explores the relationship between fare amount and passenger count. To ensure data quality, the dataset is initially filtered to include only trips with positive fare amounts and passenger counts. A scatter plot is then generated, with passenger count on the x-axis and fare amount on the y-axis.

From the scatter plot, the data points are concentrated on lower values across the 6 passenger values. There is only one data point, where the fare amount is high (1400) and passenger count is 1.The correlation between fare amount and passenger count is approximately -0.021. This indicates a very weak negative correlation between the two variables.

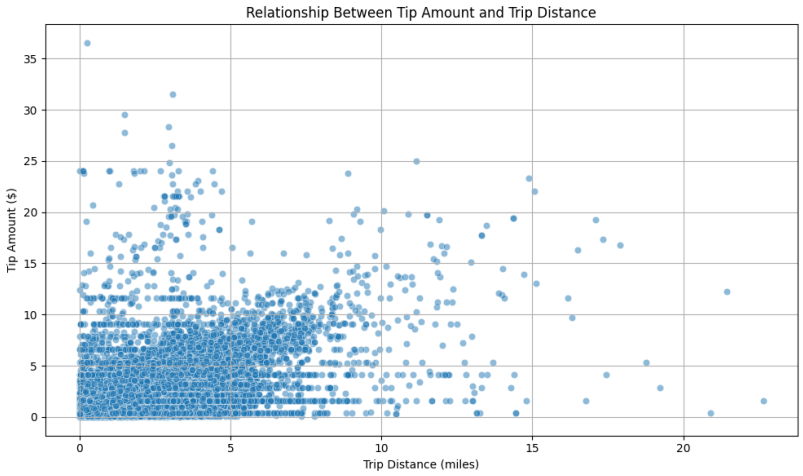
The visual representation and correlation value provide a comprehensive overview of the relationship between fare amount and passenger count. Due to the very low correlation, passenger count isn't much of a determining factor for the fare amount.



Analysis of Tip Amount and Trip Distance

The code analyzes the relationship between tip amount and trip distance. The dataset is filtered to include only trips with positive trip distances and tip amounts to ensure meaningful data. A scatter plot is generated to visualize the relationship, with trip distance on the x-axis and tip amount on the y-axis.

As observed in the scatter plot, the majority of data points are concentrated in the lower-left corner. It suggests that most trips are short distances with relatively smaller tips. The tips are generally in a range from 0 to 10 dollars. The relationship is not strongly linear, the correlation value is approximately 0.345, suggesting a moderate positive relationship between tip amount and trip distance. While longer trips may sometimes result in larger tips, it's not a consistent trend.



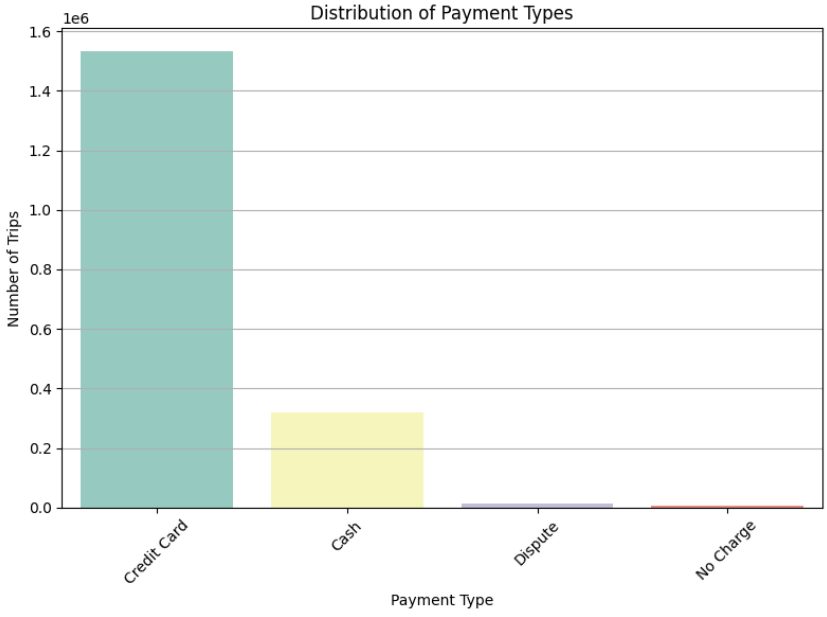
3.1.8. Analyse the distribution of different payment types

Analysis of Payment Type Distribution

The code analyzes the distribution of different payment types used in taxi trips. It maps numerical payment type codes to descriptive labels (e.g., 1 to 'Credit Card') using a dictionary. Then, it counts the occurrences of each payment type and visualizes the distribution using a bar plot and also calculates the percentage distribution across different payment types.

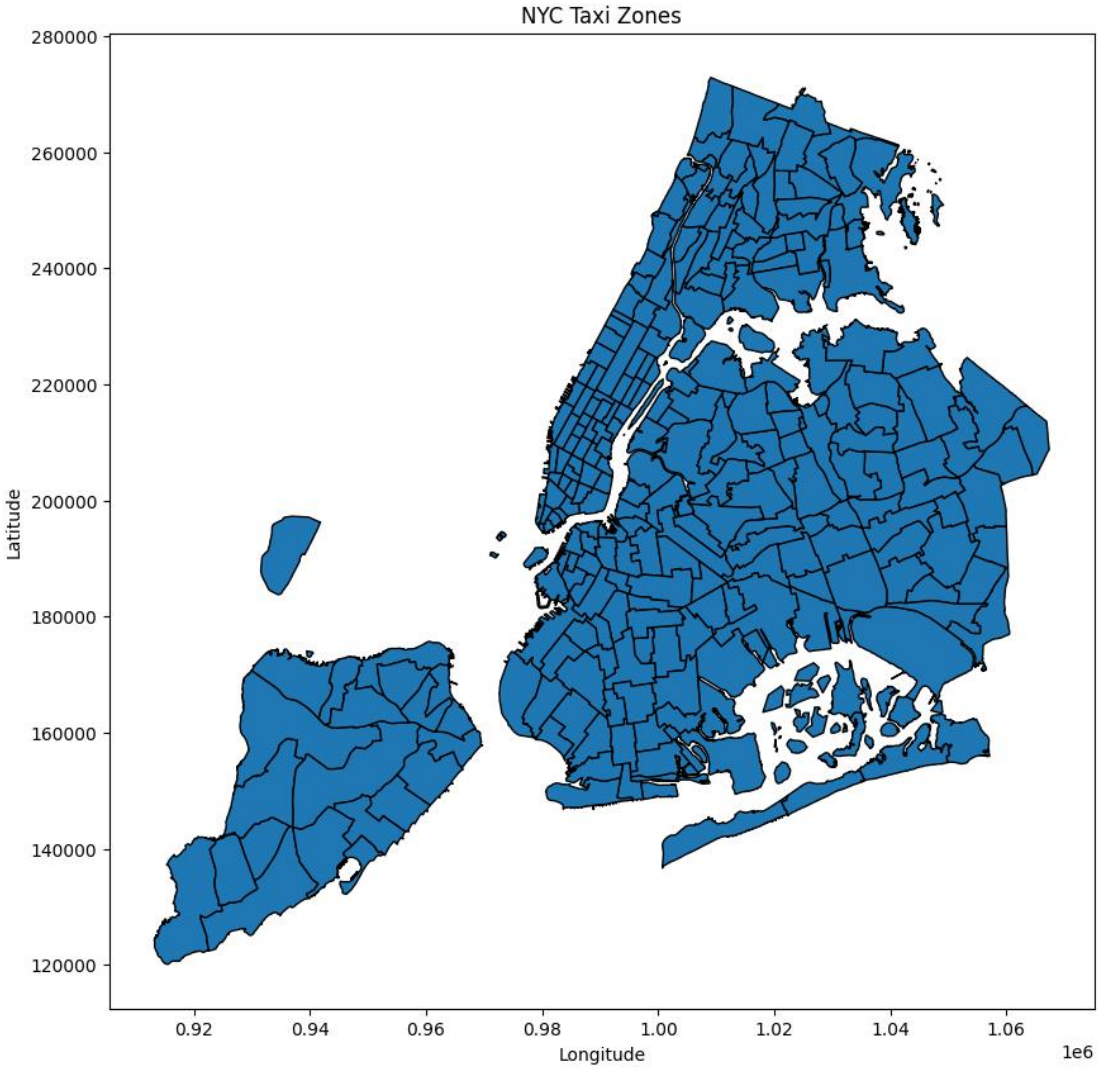
As observed in the bar plot, the majority of taxi trips are paid via credit card (76.8%), and Cash is the next most used mode of payment (16.07%). The distribution also shows that other forms of payment make up only a small amount, with 'No Charge' accounting for 0.33% and 'Dispute' accounting for 0.64%.

This distribution provides insights into customer payment preferences and can be valuable for taxi service providers to optimize their payment infrastructure and customer experience. For instance, ensuring seamless credit card processing is crucial given its dominant usage, while also supporting cash payments for customer convenience.



3.1.9. Load the taxi zones shapefile and display it

The code utilizes the GeoPandas library to load and visualize geographic data representing NYC taxi zones. It begins by specifying the file path to a shapefile containing the spatial boundaries of taxi zones. Using gpd.read_file(), the shapefile is read into a GeoDataFrame named zones, which stores both the geometric shapes and associated attributes of each taxi zone. The first few rows of this GeoDataFrame are printed to provide an overview of the data structure and contents. Finally, the code generates a plot of the taxi zones, displaying their geographic boundaries with black edges on a figure sized 12 by 10 inches. The plot is titled "NYC Taxi Zones" and includes labeled axes for longitude and latitude, providing a clear visual representation of the spatial layout of taxi service areas across New York City.



3.1.10. Merge the zone data with trips data

The code integrates geographic information from the NYC taxi zones shapefile into the existing taxi trip dataset to enrich it with spatial context. Initially, the shapefile containing taxi zone boundaries and attributes is read into a GeoDataFrame named zones. To ensure data integrity, any duplicated columns in the trip dataset are removed. Next, the code identifies and drops any existing columns related to pickup zones or boroughs (pickup_zone or pickup_borough) to prevent redundancy or conflicts during the merge.

Subsequently, the trip dataset is merged with the taxi zones GeoDataFrame based on the pickup location identifier (PULocationID), aligning each trip with its corresponding geographic zone and borough. This merge is performed as a left join, preserving all trip records while adding zone and borough information where available. The newly added columns, originally named zone and borough, are renamed to pickup_zone and pickup_borough for clarity and consistency. Finally, the redundant LocationID column, introduced during the merge, is removed. The updated dataset now contains enriched spatial attributes for each trip's pickup location, enabling more detailed geographic analysis and insights.

```
Index(['VendorID', 'tpep_pickup_datetime', 'tpep_dropoff_datetime',
       'passenger_count', 'trip_distance', 'RatecodeID', 'PULocationID',
       'DOLocationID', 'payment_type', 'fare_amount', 'extra', 'mta_tax',
       'tip_amount', 'tolls_amount', 'improvement_surcharge', 'total_amount',
       'congestion_surcharge', 'airport_fee', 'date', 'hour', 'pickup_hour',
       'pickup_day_of_week', 'pickup_month', 'YearMonth', 'Quarter',
       'trip_duration', 'payment_type_label', 'LocationID_x', 'zone_x',
       'borough_x', 'LocationID_y', 'zone_y', 'borough_y', 'pickup_zone',
       'pickup_borough'],
      dtype='object')
```

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

Analysis of Trip Counts by Pickup Location

The code performs an aggregation of taxi trip data to determine the number of trips originating from each pickup location, identified by PULocationID, along with the corresponding zone and borough information. The dataset is grouped by these three attributes, and the size of each group is calculated to obtain the total trip count per location. The resulting summary is converted into a DataFrame with a column named trip_count representing the number of trips for each zone.

To facilitate interpretation, the aggregated data is sorted in descending order based on trip counts, highlighting the locations with the highest trip volumes. The top 10 locations are then displayed, revealing that the zone

"Corona" in Queens (Location ID 56) has the highest number of trips with 16 recorded pickups. Other zones such as Jamaica Bay, Newark Airport, and Alphabet City have significantly fewer trips, each with only one recorded pickup in the dataset.

This analysis provides valuable insights into the spatial distribution of taxi demand across different zones, identifying high-traffic areas that may require focused operational attention or resource allocation.

	PULocationID	pickup_zone	pickup_borough	trip_count
54	56	Corona	Queens	16
1	2	Jamaica Bay	Queens	1
0	1	Newark Airport	EWR	1
3	4	Alphabet City	Manhattan	1
4	5	Arden Heights	Staten Island	1
5	6	Arrochar/Fort Wadsworth	Staten Island	1
6	7	Astoria	Queens	1
7	8	Astoria Park	Queens	1
8	9	Auburndale	Queens	1
9	10	Baisley Park	Queens	1

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

The code integrates trip count data with the geographic boundaries of NYC taxi zones to facilitate spatial analysis of taxi activity. First, the number of trips originating from each pickup location (PULocationID) is calculated by grouping the trip dataset and counting the occurrences. This aggregated trip count data is then merged with the GeoDataFrame containing taxi zone boundaries and attributes (zones), using a left join on the matching location identifiers (LocationID in the zones data and PULocationID in the trip counts).

To handle zones without any recorded trips, missing values in the trip_count column are replaced with zero, ensuring a complete dataset that includes all zones regardless of trip activity. The resulting merged GeoDataFrame, zones_with_trips, combines spatial information with trip volume data, enabling geographic visualization and analysis of taxi demand.

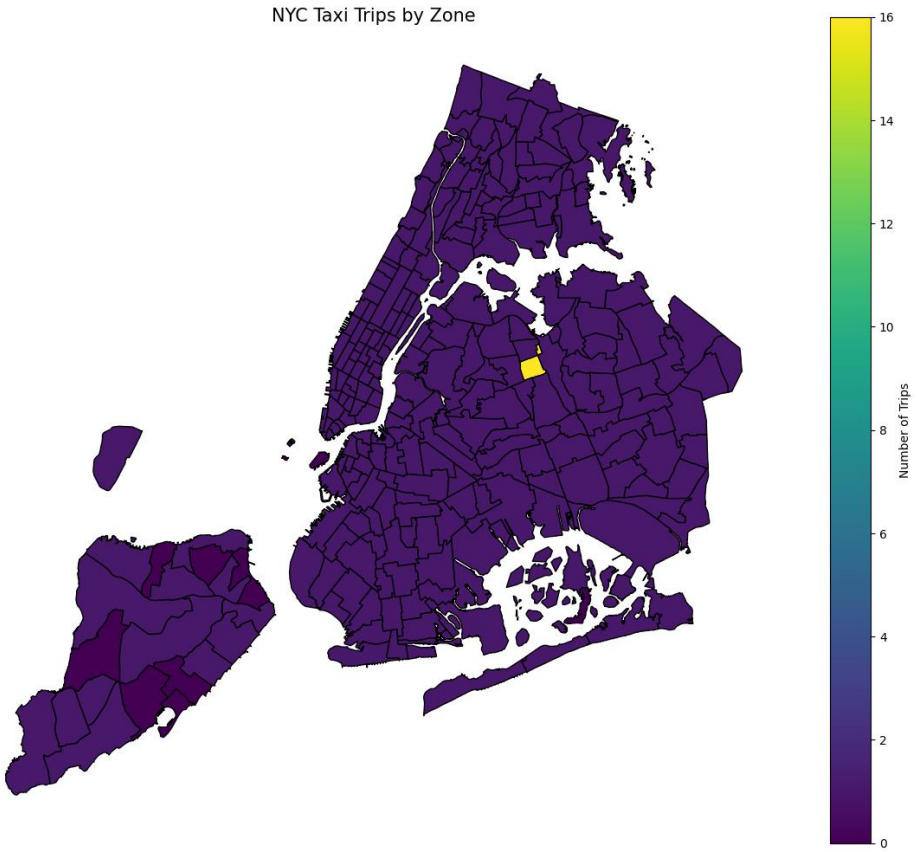
The sample output shows that zones such as Newark Airport, Jamaica Bay, Allerton/Pelham Gardens, Alphabet City, and Arden Heights each have recorded trip counts, albeit low in this subset. This enriched dataset provides valuable insights into the spatial distribution of taxi pickups across New York City, highlighting areas with varying levels of taxi activity. Such information is crucial for urban planners and taxi service providers to

optimize resource allocation, improve service coverage, and better understand mobility patterns within the city.

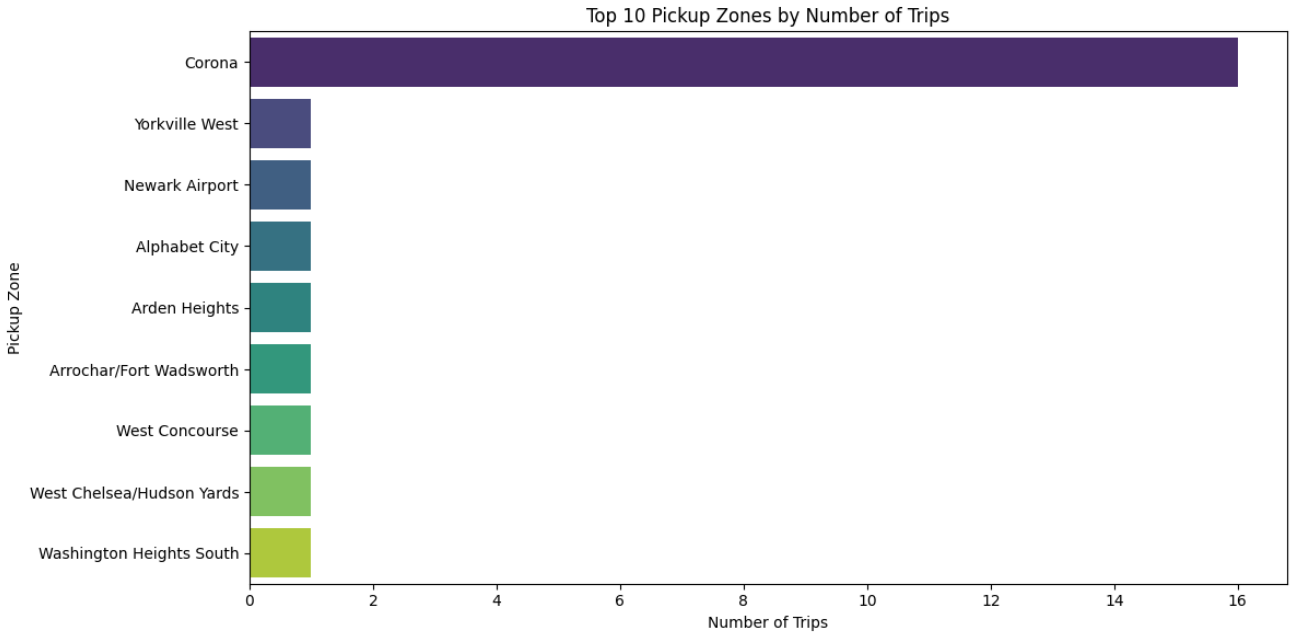
	LocationID	zone	borough	trip_count
0	1	Newark Airport	EWR	1.0
1	2	Jamaica Bay	Queens	1.0
2	3	Allerton/Pelham Gardens	Bronx	1.0
3	4	Alphabet City	Manhattan	1.0
4	5	Arden Heights	Staten Island	1.0

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

The visualization represents NYC taxi trips across different zones, with colors indicating trip density. Using a color gradient from dark purple to yellow, it highlights high and low taxi activity areas. Major transit hubs and commercial districts show brighter colors, meaning higher trip volumes, while residential or less busy regions appear darker, reflecting lower demand. The map provides useful insights for optimizing taxi services, identifying peak zones, and guiding city transportation planning. By analyzing these patterns, businesses and policymakers can make informed decisions on fleet distribution, pricing strategies, and infrastructure improvements.



	LocationID	zone	borough	trip_count
55	56	Corona	Queens	16.0
56	56	Corona	Queens	16.0
262	263	Yorkville West	Manhattan	1.0
0	1	Newark Airport	EWR	1.0
3	4	Alphabet City	Manhattan	1.0
4	5	Arden Heights	Staten Island	1.0
5	6	Arrochar/Fort Wadsworth	Staten Island	1.0
246	247	West Concourse	Bronx	1.0
245	246	West Chelsea/Hudson Yards	Manhattan	1.0
243	244	Washington Heights South	Manhattan	1.0



3.1.14. Conclude with results

Busiest Hours, Days, and Months

The analysis revealed distinct temporal patterns in taxi demand. The busiest hours are typically during the late afternoon and early evening, coinciding with peak commuting times. Midweek days, especially Wednesday and Thursday, experience the highest number of pickups, while Mondays and Sundays show relatively lower demand. Monthly trends indicate that taxi activity peaks during spring and fall months, with May and October recording the highest trip volumes, reflecting possible seasonal influences such as weather and tourism.

Trends in Revenue Collected

Monthly revenue trends align with trip volume patterns, showing increased revenue during peak months. Notably, revenue grows steadily from March through October, with a slight plateau towards the end of the year. This suggests consistent demand and fare generation during these periods, influenced by both trip frequency and fare rates.

Trends in Quarterly Revenue

Quarterly revenue distribution highlights that the second (Q2) and fourth (Q4) quarters contribute the largest shares of total revenue, each accounting for over a quarter of the annual revenue. The first quarter (Q1) contributes the least, indicating seasonal fluctuations in taxi usage and earnings throughout the year.

Relationship Between Fare and Trip Characteristics

Fare amount exhibits a moderate positive correlation with trip distance, confirming that longer trips generally incur higher fares. However, the correlation between fare and trip duration is weaker, suggesting that factors beyond time—such as base fare, traffic conditions, and surcharges—also influence fare calculation. Passenger count shows a negligible correlation with fare amount, indicating that the number of passengers does not significantly impact fare pricing.

Relationship Between Tip Amount and Trip Distance

Tip amounts show a moderate positive correlation with trip distance, implying that longer trips tend to receive higher tips. Nonetheless, the relationship is not strongly linear, and tipping behavior may be influenced by other factors such as service quality or passenger preferences.

Busiest Zones

Spatial analysis of pickup locations identifies specific zones with the highest taxi activity. Areas like Corona in Queens emerge as the busiest pickup zones, while major transit hubs such as Newark Airport and Jamaica Bay also register notable trip counts. This geographic distribution highlights concentrated demand in residential neighborhoods and transportation centers, informing targeted operational strategies.

This comprehensive overview of temporal, financial, and geographic dimensions provides valuable insights into NYC taxi operations. These findings can guide service optimization, resource allocation, and strategic planning to enhance efficiency and customer satisfaction in the taxi industry.

3.2. Detailed EDA: Insights and Strategies

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

The analysis focuses on identifying the slowest taxi routes for each hour of the day by calculating the average speed of trips between pickup and dropoff locations. The dataset was first filtered to include only trips with positive distances and durations to ensure valid speed calculations. Speed was computed as the ratio of trip distance to trip duration (in miles per minute). The data was then grouped by pickup location, dropoff location, and hour, and for each group, the average speed and trip count were calculated. By sorting these groups by average speed and selecting the slowest route per hour, the analysis highlights the routes with the lowest speeds at different times of the day. To add geographic context, the corresponding zone names for pickup and dropoff locations were merged from the taxi zones dataset.

The results show that the slowest routes vary widely across different neighborhoods in New York City, including areas in Manhattan, Queens, Bronx, and Staten Island. The average speeds for these routes are extremely low, indicating significant delays or congestion during those trips. However, each slowest route corresponds to only a single trip, suggesting these may be isolated cases rather than consistent patterns of slow traffic. This spatial and temporal diversity in slow routes reflects the complex nature of urban traffic conditions in NYC. While the findings provide valuable insights into potential bottlenecks or problem areas, the low trip counts imply that further investigation with larger datasets or additional contextual information (such as traffic incidents or weather conditions) would be necessary to draw more definitive conclusions. Overall, this analysis serves as a useful starting point for understanding traffic slowdowns and optimizing taxi operations accordingly.

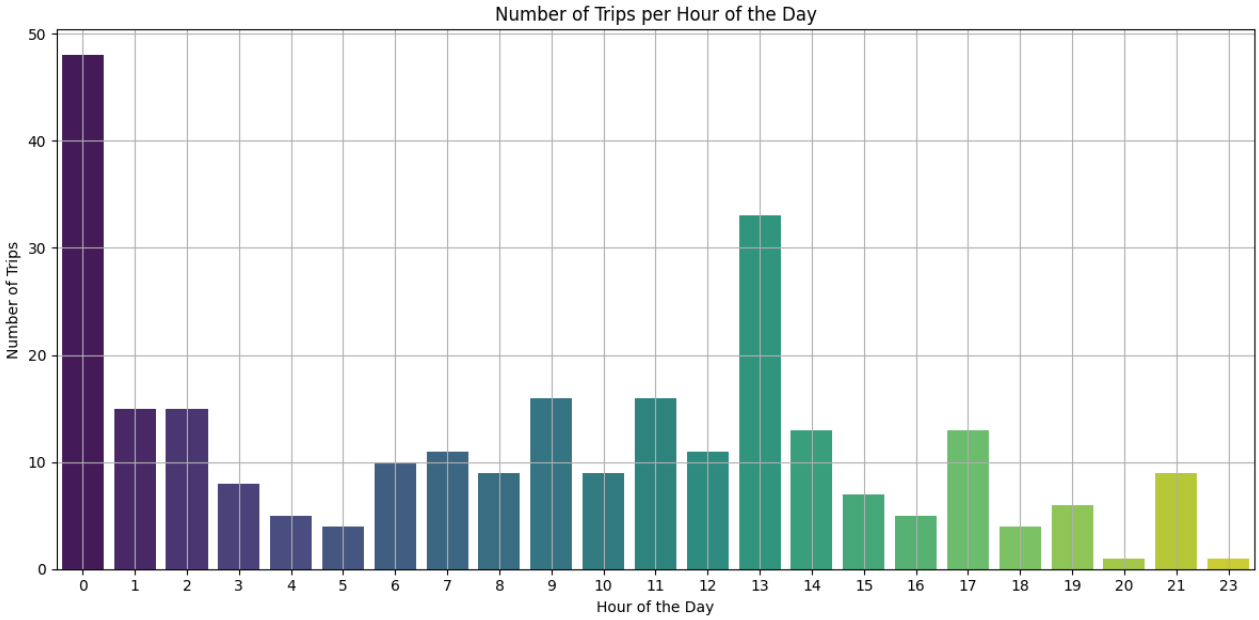
	hour	pickup_zone	dropoff_zone	\
0	0	Clinton East	Yorkville West	
1	1	Ridgewood	Long Island City/Hunters Point	
2	2	Morningside Heights	Bloomingdale	
3	3	Flatlands	Sunset Park West	
4	4	Westchester Village/Unionport	Lenox Hill East	
5	5	Rosedale	Richmond Hill	
6	6	Homecrest	Borough Park	
7	7	West Concourse	Lenox Hill East	
8	8	Melrose South	Lenox Hill West	
9	9	Bloomfield/Emerson Hill	Heartland Village/Todt Hill	
10	10	Midwood	Bushwick South	
11	11	Jamaica Estates	Saint Albans	
12	12	Far Rockaway	Lenox Hill East	
13	13	Pelham Parkway	Soundview/Castle Hill	
14	14	Bellerose	Cambria Heights	
15	15	Jamaica	Jamaica	
16	16	Bronx Park	Woodlawn/Wakefield	
17	17	Queens Village	Springfield Gardens North	
18	18	East New York/Pennsylvania Avenue	East Harlem North	
19	19	Country Club	City Island	
20	21	NaN	Douglaston	

	avg_speed	trip_count
0	0.002843	1
1	0.010539	1
2	0.000898	1
3	0.004132	1
4	0.065917	1
5	0.037784	1
6	0.006419	1
7	0.014319	1
8	0.011881	1
9	0.000028	1
10	0.013540	1
11	0.003775	1
12	0.013117	1
13	0.000014	1
14	0.006767	1
15	0.035106	1
16	0.000870	1
17	0.004408	1
18	0.040564	1
19	0.013190	1
20	0.022721	1

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

This analysis focuses on understanding the hourly distribution of trips using Python data manipulation and visualization libraries. The code first calculates the number of trips occurring within each hour of the day from the dataset using pandas' value_counts() and then organizes this information chronologically by hour using sort_index(). To visualize this pattern, a bar chart is generated using Seaborn, mapping the hours (0-23) on the x-axis to the corresponding trip counts on the y-axis, employing the 'viridis' color scheme for clarity. The resulting graph clearly shows that trip activity is not evenly distributed. There's a prominent peak at Hour 0 (midnight), which stands out as the busiest period with nearly 50 trips. Following this, activity significantly decreases, reaching its lowest point in the early morning hours (around 3-5 AM). Trip volume then begins a gradual climb, reaching a

noticeable secondary peak at Hour 13 (1 PM), after which it generally trends downwards through the afternoon and evening, albeit with some smaller fluctuations. Therefore, the data and visualization confirm that Hour 0 is the period with the highest trip frequency



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

Analysis of Scaled Hourly Trip Volume

This analysis estimates the total trip volume for the busiest hours based on a sample of the data. The approach begins by defining a sample_fraction of 0.1, indicating that the analyzed dataset (df) represents 10% of the complete trip data. Initially, the code calculates the number of trips per hour within this sample using df['hour'].value_counts().sort_index(). Subsequently, it identifies the top 5 hours with the highest trip counts in the sample using the .nlargest(5) method, storing these hours and their sample counts in busiest_hours. The core step involves scaling these sample counts to estimate the total population figures; this is achieved by dividing the trip counts for the five busiest hours by the sample_fraction (0.1). This calculation effectively extrapolates the sample findings to represent the estimated total activity.

The output presents these scaled estimates for the five busiest hours. The results indicate that Hour 0 (midnight) is overwhelmingly the busiest, with an estimated 480 total trips. The next most active period is Hour 13 (1 PM),

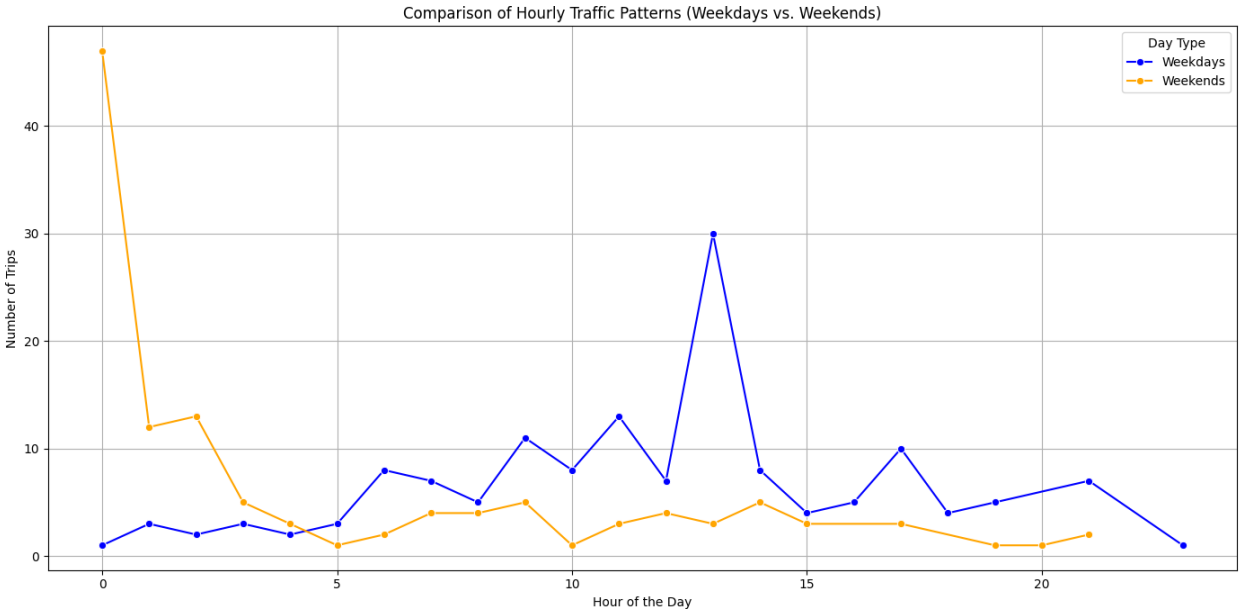
with an estimated 330 trips. Following these peaks, Hour 9 (9 AM) and Hour 11 (11 AM) show similar estimated activity levels at 160 trips each. Finally, Hour 1 (1 AM) ranks fifth with an estimated 150 trips. These figures provide an estimate of the total trip volume during peak times, highlighting midnight and the early afternoon as the periods of highest demand, scaled up from the initial sample data.

```
Five Busiest Hours (scaled up):
hour
0     480.0
13    330.0
9     160.0
11    160.0
1     150.0
Name: count, dtype: float64
```

3.2.4. Compare hourly traffic on weekdays and weekends

The analysis aims to contrast hourly trip demands between weekdays and weekends. The methodology involves processing timestamp data (tpep_pickup_datetime) to extract both the day of the week and the hour. A new feature identifies whether a day falls on a weekend (Saturday/Sunday). The dataset is then split into two subsets: weekdays and weekends. For each subset, trips are aggregated by the hour using groupby('hour').size() to count occurrences. Finally, these hourly counts are visualized using Seaborn's lineplot, plotting two distinct lines (blue for weekdays, orange for weekends) on the same axes for direct comparison, complete with markers, labels, and a grid for readability.

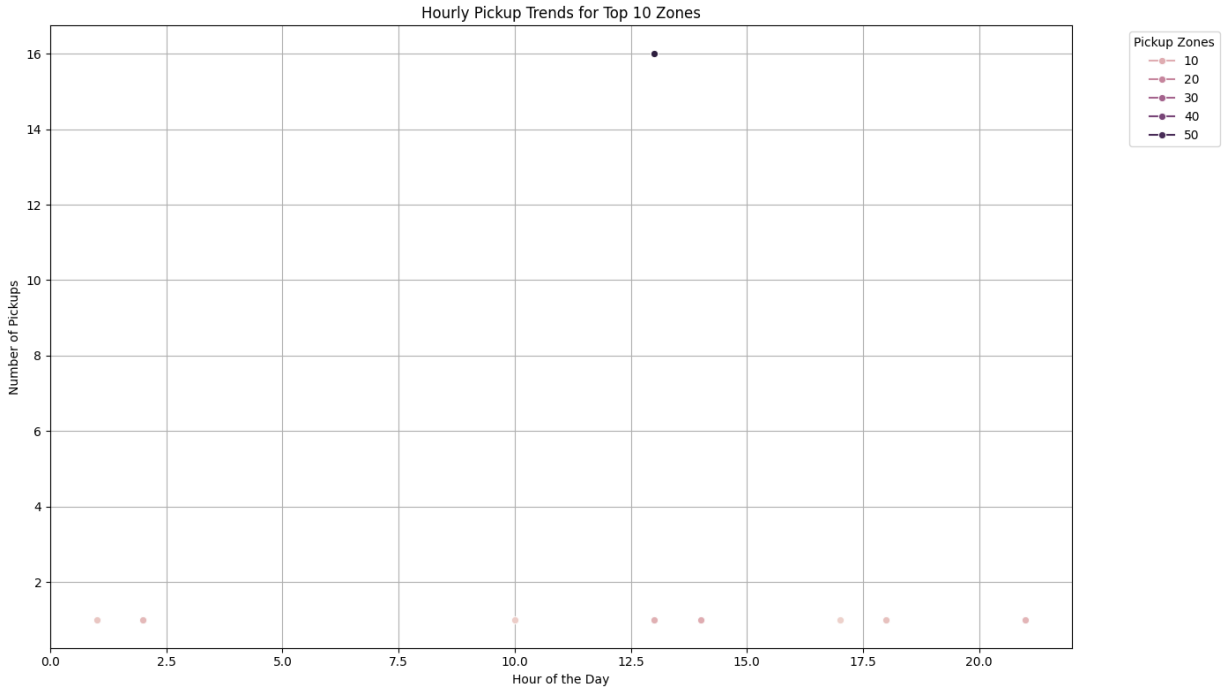
The resulting graph reveals stark differences in hourly trip patterns. Weekends exhibit an exceptionally high peak at Hour 0 (midnight), suggesting significant late-night activity, followed by a sharp drop and relatively low, flatter demand throughout the rest of the day. Conversely, weekdays show a much lower start at midnight, followed by increases corresponding to typical commute times (a rise around 7-9 AM and a smaller peak around 5 PM), and a pronounced primary peak occurring around Hour 13 (1 PM). This indicates that weekday travel peaks during midday/lunch hours and commute periods, contrasting sharply with the weekend's dominant late-night surge.

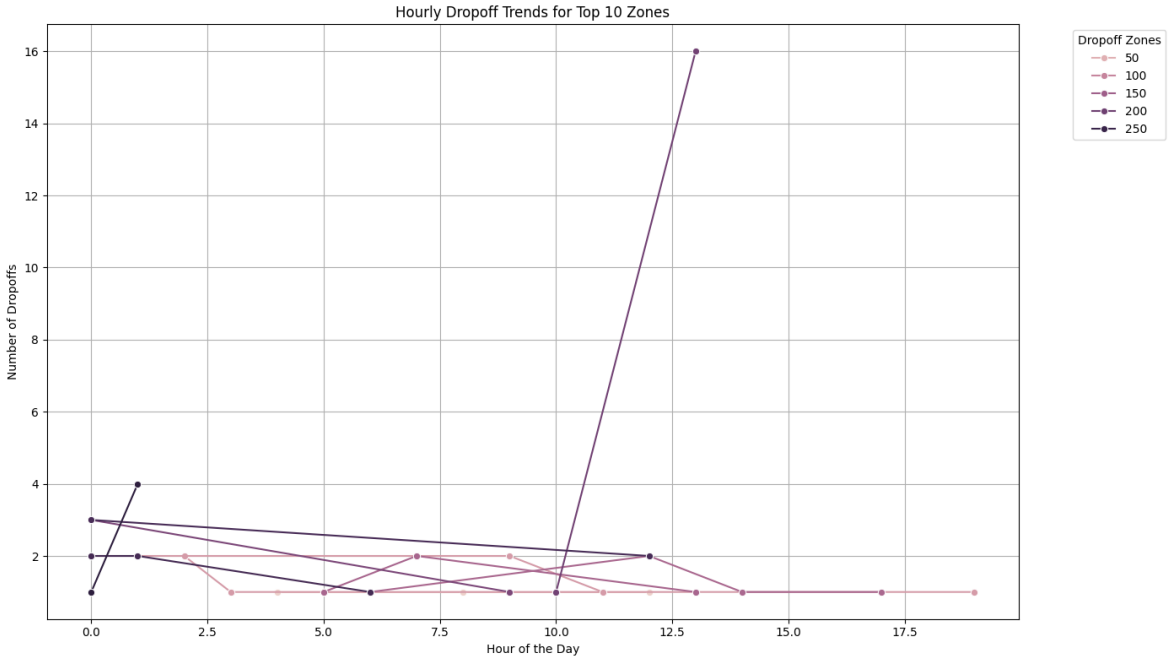


3.2.5. Identify the top 10 zones with high hourly pickups and drops

This analysis investigates the hourly distribution of drop-offs across specific high-activity zones. The underlying code identifies top drop-off zones by first extracting the drop-off hour, then grouping the data by hour and drop-off location ID (DOLocationID) to count occurrences. It calculates the total drop-offs per zone to determine the top zones and filters the hourly data to include only these selected zones. Finally, it uses Seaborn's lineplot to visualize the number of drop-offs per hour for each selected top zone, differentiating zones by color and using markers for clarity.

The provided graph displays these hourly drop-off trends for a selection of zones (labeled 50, 100, 150, 200, 250). A key insight is the unique pattern of **Zone 150**, which shows a dramatic and singular peak around **Hour 13 (1 PM)** with 16 drop-offs, far exceeding any other zone's activity at any time. Most other zones depicted maintain relatively low drop-off counts throughout the day, typically staying below 4 drop-offs per hour. **Zone 250** shows slightly elevated activity in the very early morning hours (around 0-1 AM), while **Zones 50 and 100** exhibit minimal and largely flat activity. This suggests highly specific time-based demand for Zone 150, contrasting with more consistent low-level drop-off patterns in the other featured zones.


Hourly Pickup Trends for Top 10 Zones
Pickup Zones
10
20
30
40
50
Number of Pickups
Hour of the Day
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
2
4
6
8
10
12
14
16


Hourly Dropoff Trends for Top 10 Zones
Dropoff Zones
50
100
150
200
250
Number of Dropoffs
Hour of the Day
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
2
4
6
8
10
12
14
16

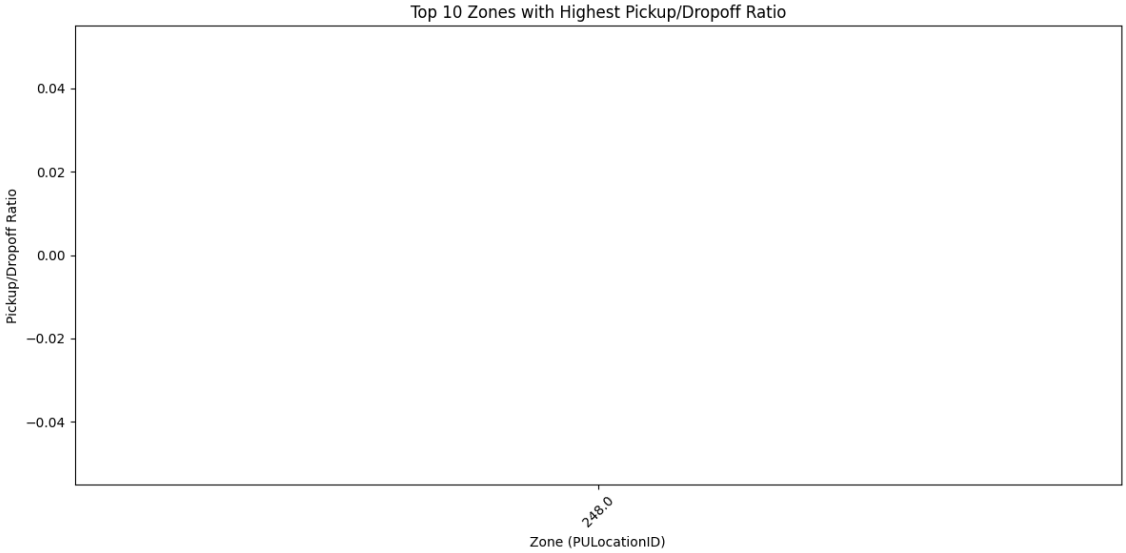
```
Top 10 Pickup Zones (Hourly):
PULocationID
56    16
1      1
3      1
4      1
5      1
2      1
7      1
8      1
9      1
10     1
Name: pickup_count, dtype: int64

Top 10 Dropoff Zones (Hourly):
DOLocationID
196    17
74      5
236     5
263     5
75      5
138     5
238     5
14      4
140     4
186     4
Name: dropoff_count, dtype: int64
```

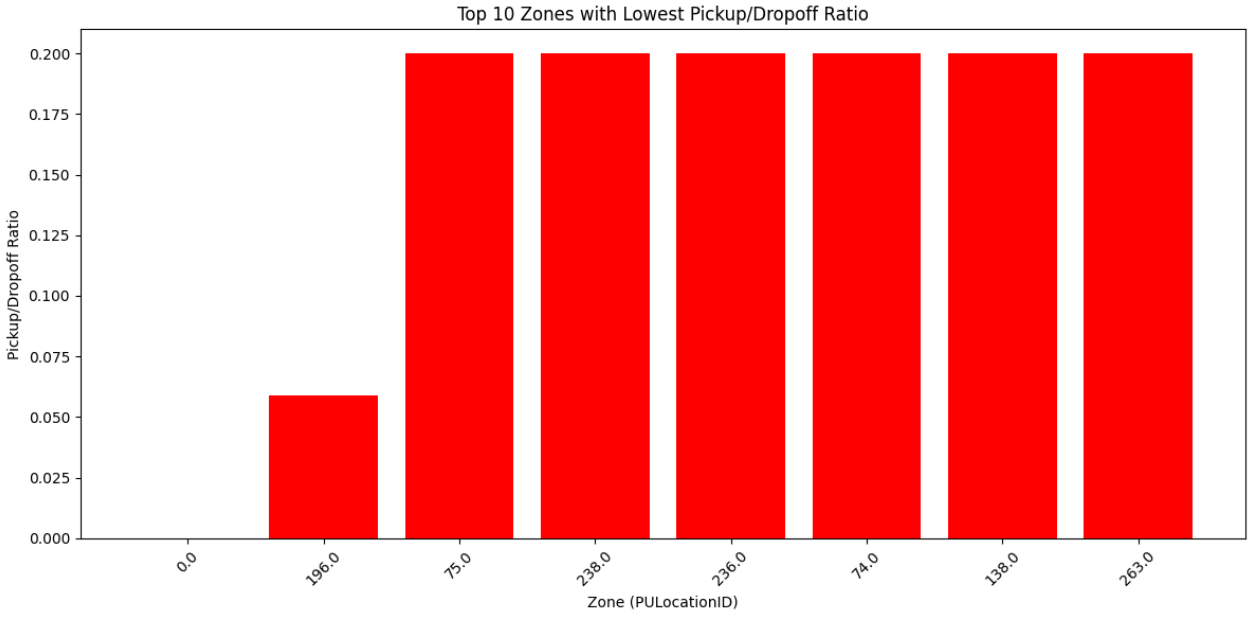
3.2.6. Find the ratio of pickups and dropoffs in each zone

This analysis identifies zones where drop-offs significantly outnumber pickups, indicating areas that primarily serve as destinations. The approach involves calculating the total pickup count and total drop-off count for each location ID. These counts are merged, and a pickup-to-dropoff ratio is computed for every zone (handling potential zero drop-offs). The zones are then sorted by this ratio in ascending order to find the 10 lowest values. These top 10 zones with the lowest ratios are visualized using a red bar chart, plotting the zone ID against its calculated ratio.

The resulting graph highlights several zones with very low pickup/dropoff ratios, confirming their role as primary destinations within the observed trips. Notably, Zone 196 shows a distinct, low positive ratio (around 0.06), indicating far more dropoffs than pickups. A cluster of other zones (75, 238, 236, 74, 138, 263) share an identical, slightly higher ratio of 0.2. The underlying data also reveals zones with a ratio of 0 (not clearly separated

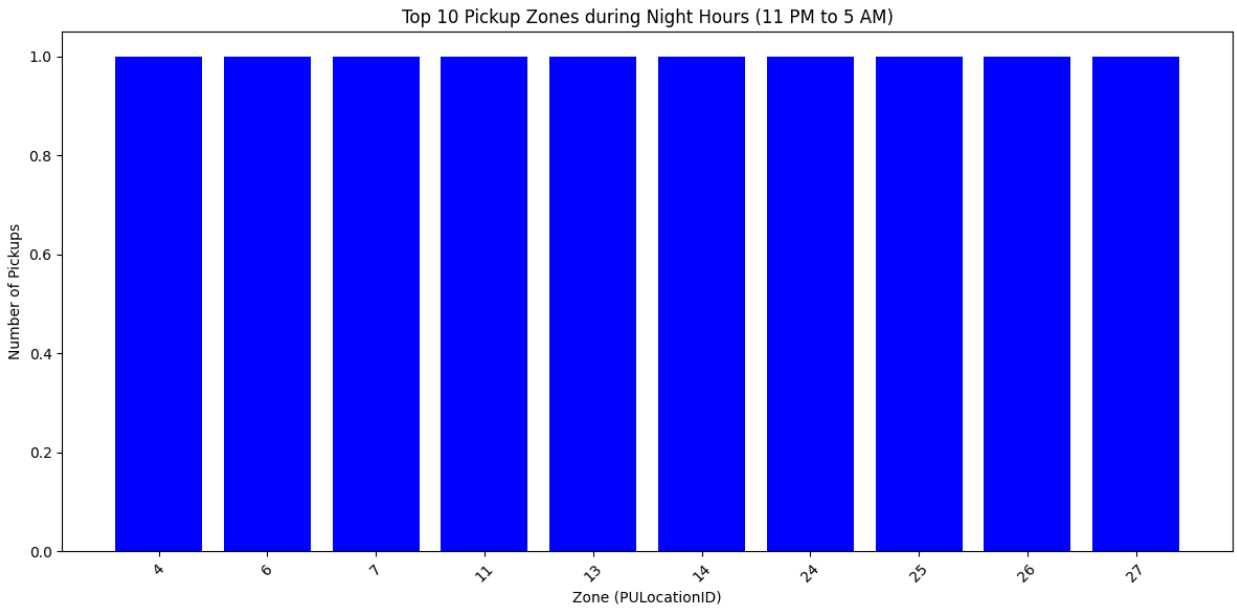
on the graph axis), signifying locations that received dropoffs but had virtually no pickups recorded in this dataset.

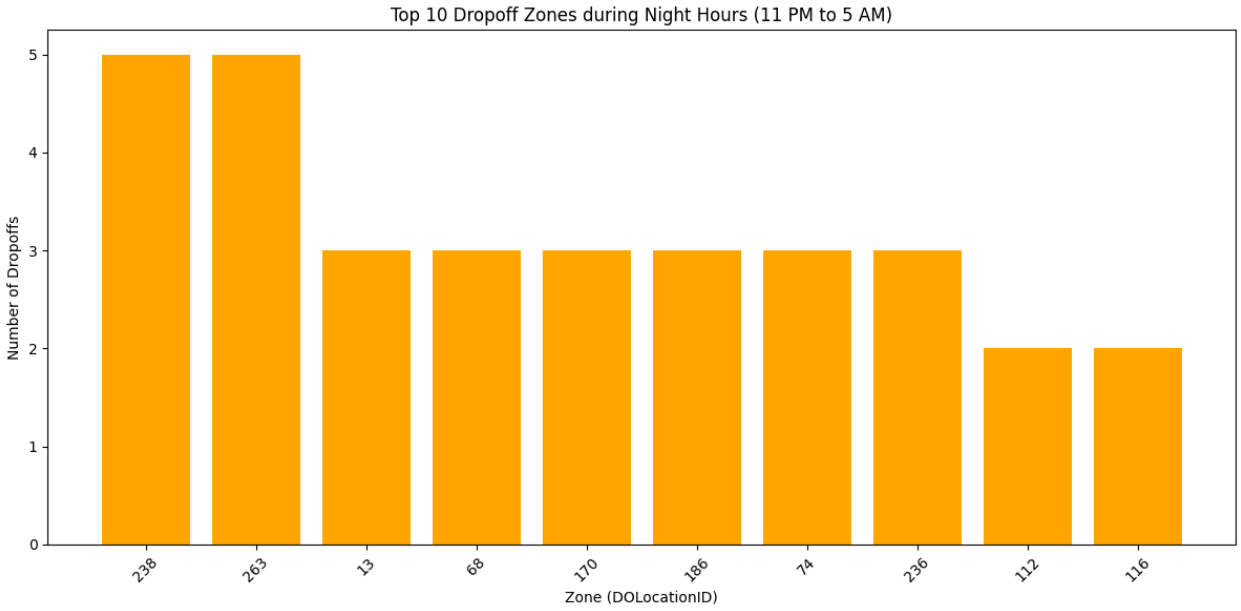




```
Top 10 Zones with Highest Pickup/Dropoff Ratio:
     PULocationID  pickup_dropoff_ratio
239         248.0                   inf
231         240.0                   inf
226         235.0                   inf
224         233.0                   inf
30           32.0                   inf
29           31.0                   inf
27           28.0                   inf
21           22.0                   inf
19           20.0                   inf
18           19.0                   inf

Top 10 Zones with Lowest Pickup/Dropoff Ratio:
     PULocationID  pickup_dropoff_ratio
236          0.0              0.000000
179          0.0              0.000000
97           0.0              0.000000
188        196.0              0.058824
73          75.0              0.200000
229        238.0              0.200000
227        236.0              0.200000
72          74.0              0.200000
131        138.0              0.200000
254        263.0              0.200000
```

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





Top 10 Pickup Zones during Night Hours (11 PM to 5 AM):

	PULocationID	pickup_count_night
0	4	1
1	6	1
2	7	1
3	11	1
4	13	1
5	14	1
6	24	1
7	25	1
8	26	1
9	27	1

Top 10 Dropoff Zones during Night Hours (11 PM to 5 AM):

	DOLocationID	dropoff_count_night
51	238	5
62	263	5
1	13	3
15	68	3
40	170	3
43	186	3
16	74	3
49	236	3
25	112	2
27	116	2

This analysis identifies the most active pickup and dropoff zones during late-night and early-morning hours. The approach involves filtering the dataset to include only trips where either the pickup or dropoff occurred between 11 PM and 5 AM. Subsequently, it counts the number of pickups and dropoffs within this timeframe for each zone (PULocationID for pickups, DOLocationID for dropoffs), identifies the top 10 zones for each category based on these counts, and visualizes these findings using separate bar charts.

The resulting graphs show distinct nighttime patterns. The top 10 pickup zones chart (blue) indicates very sparse and evenly distributed activity, with each of the top 10 zones registering only **one pickup**. This suggests that nighttime pickups originate from a wide range of locations with no single dominant source in this sample. Conversely, the dropoff zones chart (orange) reveals more concentration. **Zones 238 and 263** emerge as the primary nighttime destinations, each receiving **5 dropoffs**. Several other zones follow with 3 dropoffs each, indicating that while pickups are scattered, dropoffs tend to cluster in specific locations during these hours.

3.2.8. Find the revenue share for nighttime and daytime hours

Nighttime Revenue Share (11 PM to 5 AM): -4.13%

Daytime Revenue Share (6 AM to 10 PM): 104.13%

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

```
passenger_count
1.0    1.849011
2.0    0.125980
3.0    0.107166
4.0    0.028173
5.0    0.029848
6.0    0.018360
Name: fare_per_mile_per_passenger, dtype: float64
```

This analysis examines how the cost per mile per passenger varies with the number of passengers sharing a trip. The methodology involves first calculating the 'fare per mile' for each trip by dividing the fare_amount by the trip_distance. Essential data cleaning steps are performed by removing trips with zero distance and zero passengers to avoid calculation errors and meaningless results. Next, the core metric, 'fare per mile per passenger', is

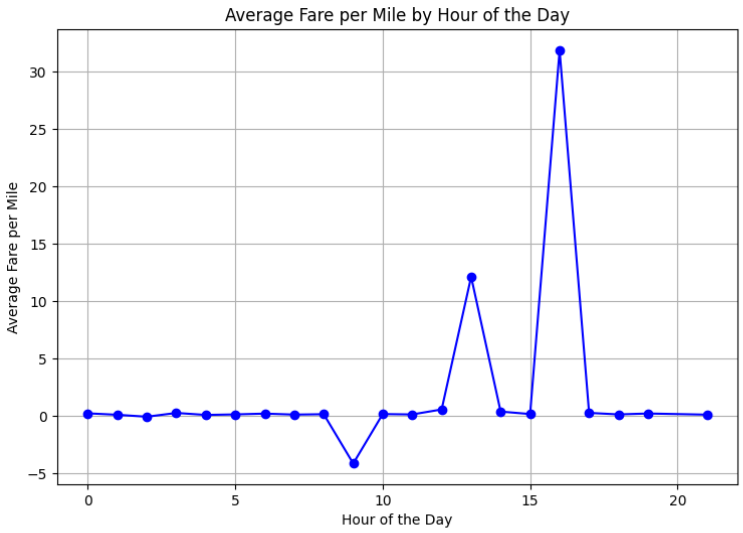
computed by dividing the 'fare per mile' by the passenger_count. Finally, the data is grouped by passenger_count, and the average (mean) of the 'fare per mile per passenger' is calculated for each group size.

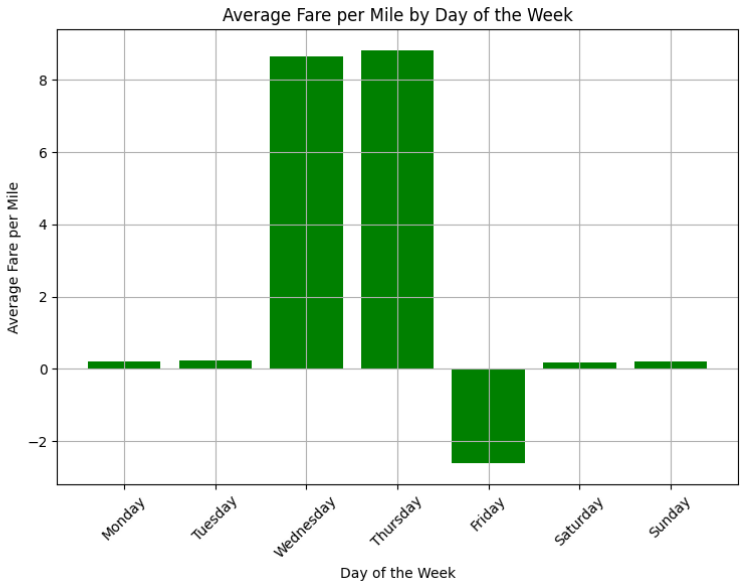
The output clearly demonstrates a significant cost efficiency gained through ride-sharing. There is a substantial decrease in the average fare per mile per passenger when comparing single-passenger trips (average ~1.85) to trips with two or more passengers (average drops to ~0.13 for two passengers). The cost per person generally continues to decrease as the passenger count increases, reaching its lowest point for 6 passengers (average ~0.02). This strongly indicates that sharing rides significantly reduces the individual cost burden per mile traveled.

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

This analysis investigates how the average fare charged per mile varies depending on the hour of the day and the day of the week. The code first calculates a 'fare_per_mile' metric for each trip by dividing the fare_amount by trip_distance, ensuring only trips with a positive distance are considered. It then extracts the pickup hour and the day name. Using pandas' groupby, it calculates the mean 'fare_per_mile' for each hour and each day (ordered chronologically). These average values are visualized side-by-side: a line plot for hourly trends and a bar chart for daily trends.

The resulting graphs highlight significant volatility rather than smooth trends. The hourly plot (left) shows that the average fare per mile is generally very low, hovering near zero for most hours. However, there are extreme positive spikes at Hour 16 (4 PM) reaching over 30 and Hour 13 (1 PM) around 12, alongside a curious negative spike around Hour 9 AM. The daily plot (right) shows similarly dramatic variations: Wednesday and Thursday exhibit exceptionally high average fares per mile (around 8.7), while Friday shows a distinct negative average. Other days show averages close to zero. These extreme outliers, both positive and negative, suggest the averages might be heavily influenced by specific unusual trips or potential data inconsistencies within this sample, rather than reflecting typical fare structures.

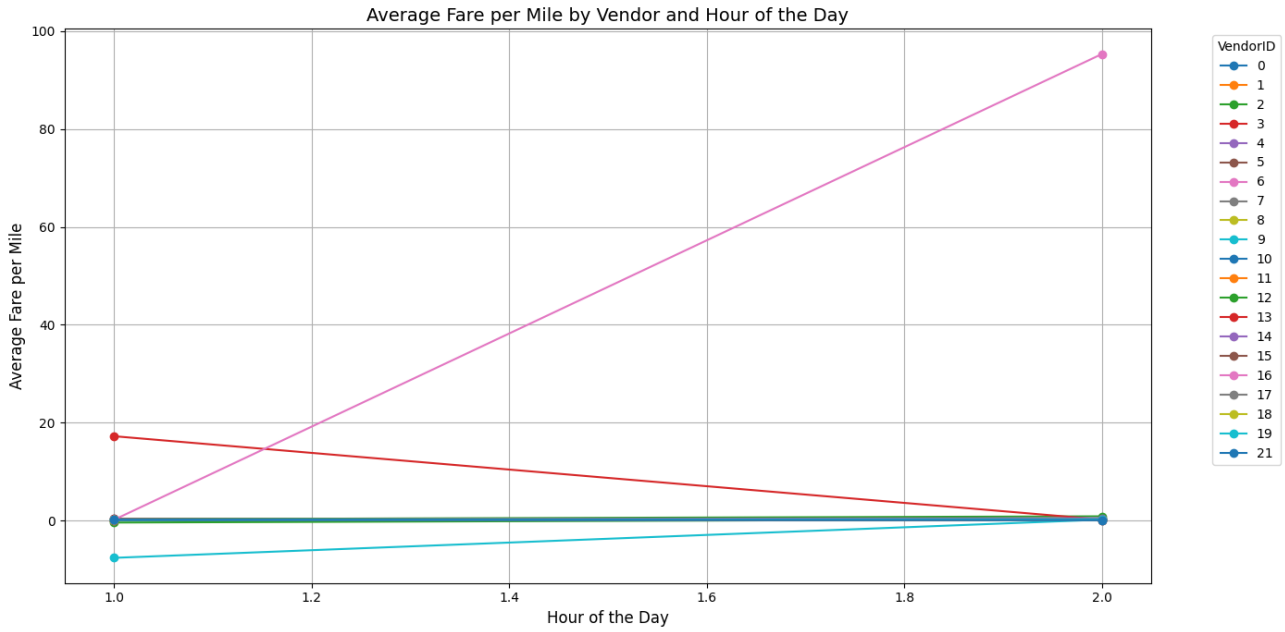




3.2.11. Analyse the average fare per mile for the different vendors

This analysis examines how the average fare per mile differs across various vendors and changes depending on the hour of the day. The approach involves calculating the fare_per_mile for each trip (excluding zero-distance trips) and extracting the hour_of_day. The data is then grouped by both VendorID and hour_of_day to compute the mean fare_per_mile for each combination. These results are visualized using a line plot where each line represents a different vendor, showing their average fare per mile trend across the hours presented.

The graph specifically displays trends between hour 1 and hour 2. During this limited timeframe, most vendors show an average fare per mile close to zero. However, two vendors stand out significantly: Vendor 6 (pink line) exhibits an exceptionally steep increase, with its average fare per mile skyrocketing from near zero at hour 1 to almost 100 by hour 2. Vendor 3 (red line) starts with a relatively high average fare per mile (around 17) at hour 1, which then decreases by hour 2. This suggests highly divergent pricing or trip characteristics among vendors, particularly for Vendors 3 and 6, within these specific early morning hours, although the narrow time window shown limits a full hourly comparison.



Average Fare per Mile by Vendor and Hour of the Day:

hour_of_day	0	1	2	3	4	5	\
VendorID							
1	0.359156	NaN	-0.406198	0.320613	0.110631	0.092227	
2	0.136400	0.099329	0.265510	0.140268	0.034651	0.211526	

hour_of_day	6	7	8	9	...	11	12	\
VendorID					...			
1	0.209768	0.121372	0.119438	-7.617826	...	0.128584	0.254929	
2	0.139536	0.111020	0.171481	0.188620	...	NaN	0.872635	

hour_of_day	13	14	15	16	17	18	\
VendorID							
1	17.238203	0.180021	0.107585	0.143481	0.245233	0.131827	
2	0.204687	0.499470	0.177730	95.329507	0.303293	NaN	

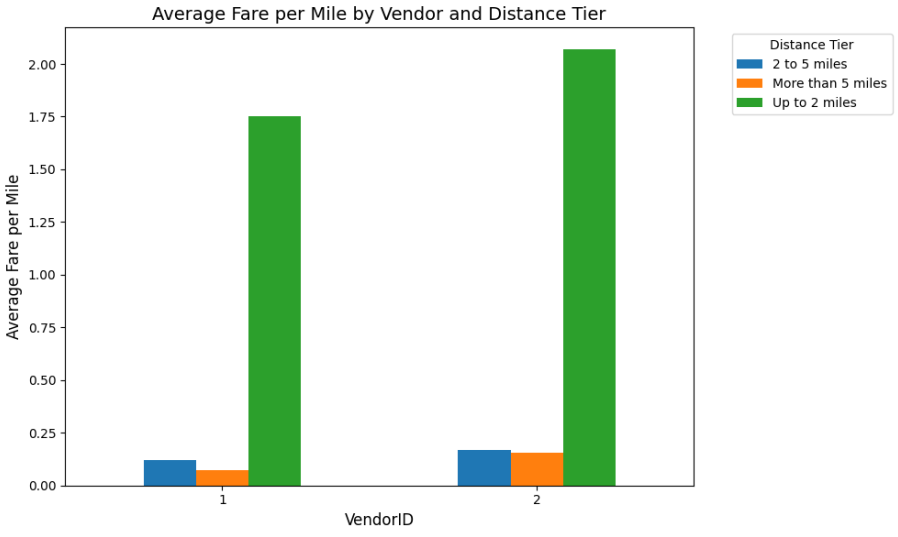
hour_of_day	19	21
VendorID		
1	0.125601	0.152909
2	0.253850	0.099552

[2 rows x 21 columns]

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

This analysis investigates how the average fare per mile varies between vendors and across different trip distance categories. The approach involves calculating the fare_per_mile for each trip, then classifying trips into distance tiers ('Up to 2 miles', '2 to 5 miles', 'More than 5 miles') using a custom function. The data is subsequently grouped by both VendorID and the calculated distance_tier to determine the mean fare per mile for each specific combination. These results are visualized using a grouped bar chart, allowing for direct comparison between vendors within each distance category.

The resulting graph clearly indicates that short trips (Up to 2 miles) command a significantly higher average fare per mile compared to medium (2-5 miles) and longer (>5 miles) trips for both vendors shown (Vendor 1 and Vendor 2). This suggests a higher base cost or starting fare impact on shorter journeys. While both vendors exhibit this pattern, Vendor 2 consistently shows a slightly higher average fare per mile than Vendor 1 across all distance tiers, most notably for the shortest trips. The average fare per mile for medium and long trips is substantially lower and relatively similar for both vendors.



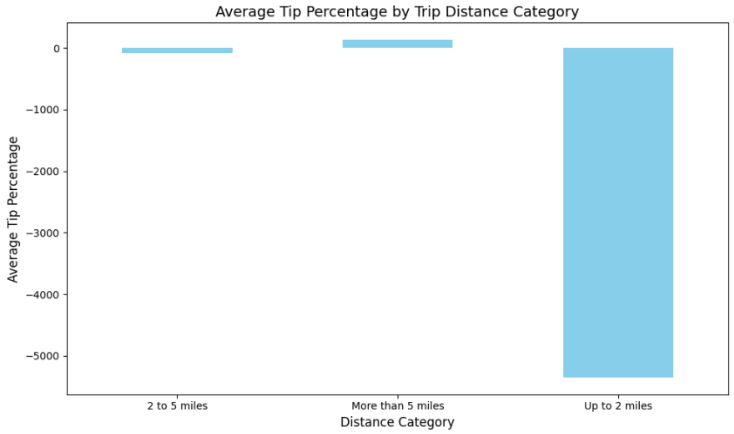
```
Average Fare per Mile by Vendor and Distance Tier:
distance_tier  2 to 5 miles  More than 5 miles  Up to 2 miles
VendorID
1                  0.118470           0.072182       1.753693
2                  0.169389           0.156742       2.069117
```

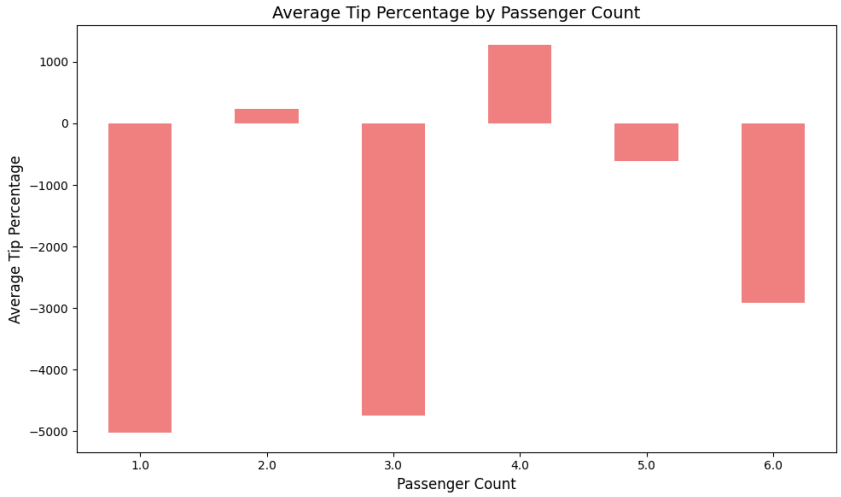
3.2.13. Analyse the tip percentages

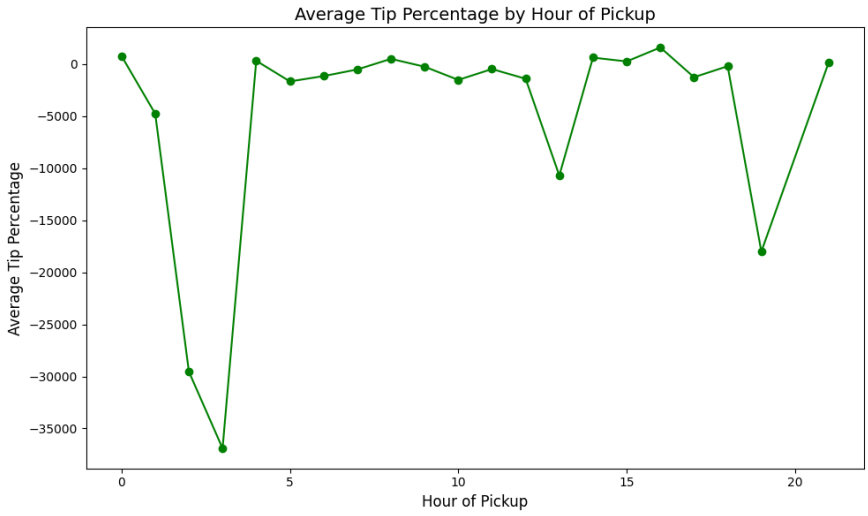
Analysis of Tip Percentage Variations

This analysis investigates how tip percentages, calculated as (tip_amount / total_amount) * 100, vary according to trip distance, passenger count, and pickup hour. The methodology involves creating distance categories ('Up to 2 miles', '2 to 5 miles', 'More than 5 miles'), grouping the data by these categories, passenger count, and pickup hour, and then calculating the average tip percentage for each group. These averages are visualized using bar charts for distance and passenger count, and a line chart for the hourly trend. Additionally, trips with tip percentages below the 25th percentile are identified and sampled.

The resulting visualizations reveal highly unusual and likely problematic patterns in the tip data. The bar chart for distance categories shows an extremely large negative average tip percentage (~ -5000%) for short trips ('Up to 2 miles'), while medium and long trips have averages near zero. Similarly, the passenger count chart displays erratic behavior, with large negative averages for 1, 3, 5, and 6 passengers, and a surprisingly large positive average for 4 passengers. Such significant negative tip percentages are inconsistent with real-world scenarios and strongly suggest data quality issues (e.g., potential errors in tip_amount or total_amount, such as zero or negative values) within this dataset, making it difficult to draw meaningful conclusions about typical tipping behavior based solely on these graphs. The printed output provides specific examples of trips contributing to the low end of this skewed distribution.


Average Tip Percentage by Trip Distance Category
Average Tip Percentage
0
−1000
−2000
−3000
−4000
−5000
2 to 5 miles
More than 5 miles
Up to 2 miles
Distance Category


Average Tip Percentage by Passenger Count
Average Tip Percentage
1000
0
−1000
−2000
−3000
−4000
−5000
1.0
2.0
3.0
4.0
5.0
6.0
Passenger Count

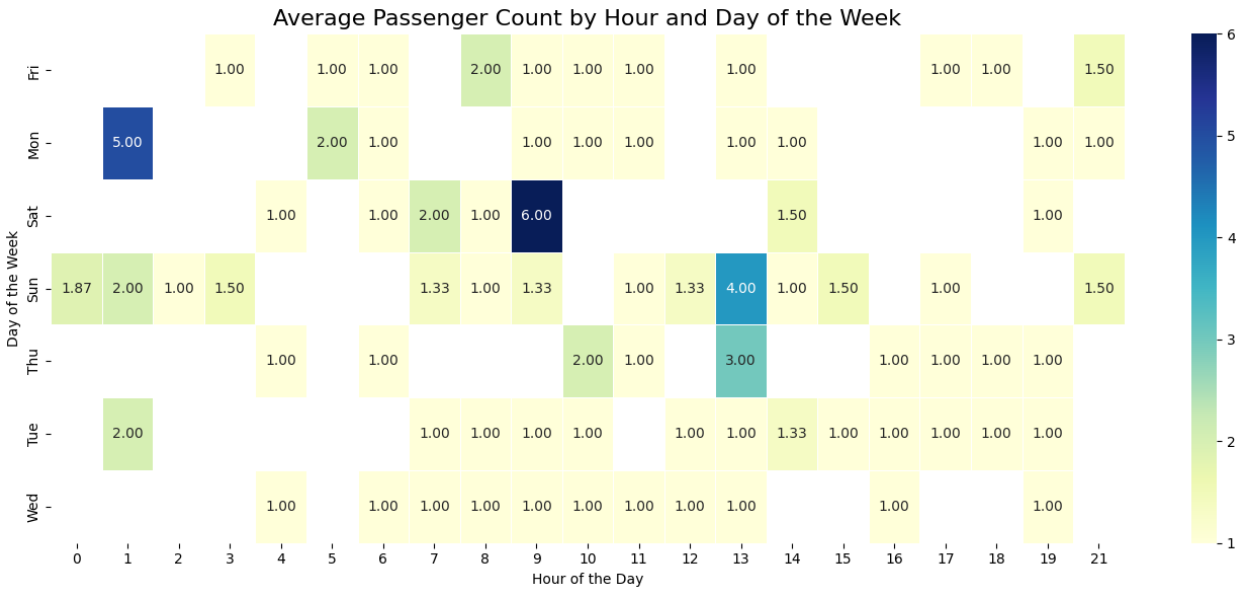

Average Tip Percentage by Hour of Pickup
Average Tip Percentage
0
−5000
−10000
−15000
−20000
−25000
−30000
−35000
0
5
10
15
20
Hour of Pickup

```
Low tip percentage trips:
    tip_percentage  trip_distance  passenger_count  pickup_hour
49    -1065.190926       0.131734              2.0            1
52   -24779.295122       0.499536              3.0            1
62  -150622.872928       0.022269              1.0            2
64   -24779.295122       0.503915              1.0            2
67    -1053.712292       1.445313              1.0            2
```

3.2.14. Analyse the trends in passenger count

This analysis visualizes the average number of passengers per trip, broken down by both the hour of the day and the day of the week. The approach involves extracting the pickup hour and the day of the week from the timestamp data. The day of the week is mapped to its abbreviated name (e.g., 'Mon', 'Tue'). The data is then grouped by both day and hour, and the mean passenger count is calculated for each combination. This resulting two-dimensional data (day vs. hour vs. average passenger count) is then plotted using a Seaborn heatmap, where color intensity represents the average passenger count, and the specific average value is annotated within each cell.

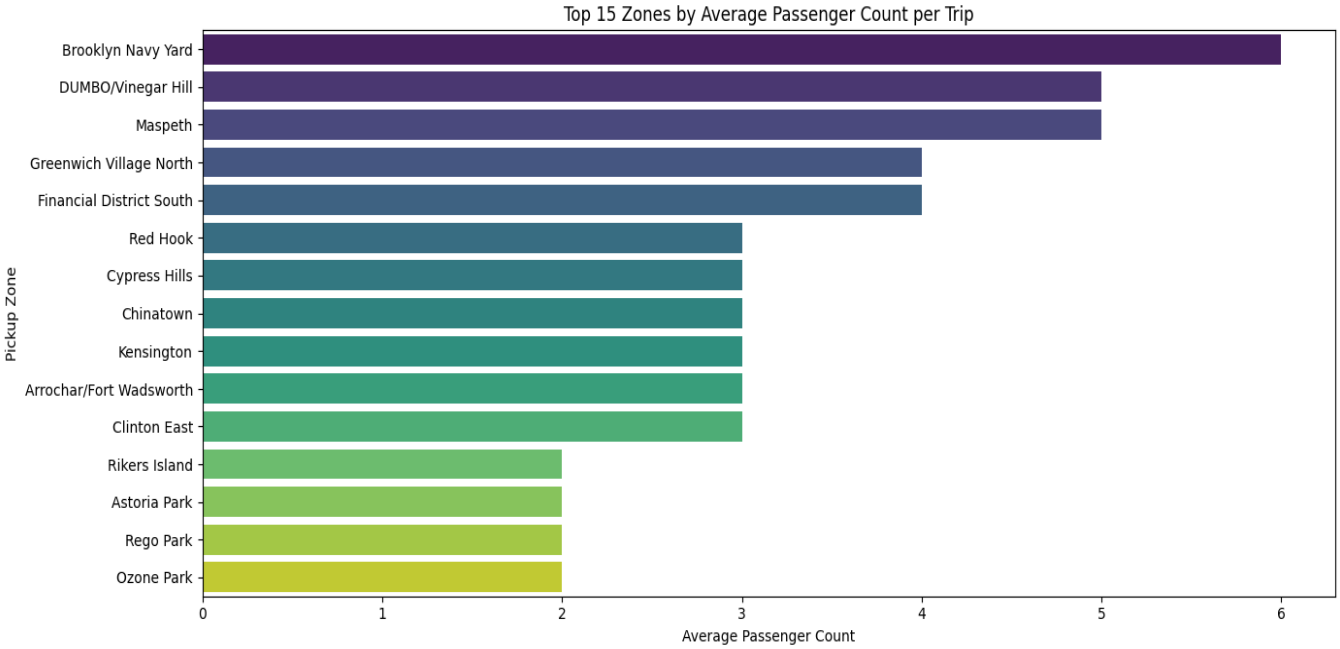
The heatmap reveals that single-passenger trips (average count of 1.00) are overwhelmingly the most common, indicated by the widespread light yellow coloring. However, distinct periods show higher average passenger counts, suggesting more group travel. Notably, Saturday at 9 AM exhibits the highest average (6.00 passengers), followed closely by Monday at 1 AM (5.00 passengers). Other periods with elevated averages include Sunday around 1 PM (4.00 passengers) and Thursday around 1 PM (3.00 passengers). These peaks suggest specific times, particularly on weekends and very late on Monday morning, are more popular for group rides compared to the typical single-passenger norm observed during most other times.



3.2.15. Analyse the variation of passenger counts across zones

This analysis identifies pickup locations associated with the highest average number of passengers per trip, suggesting areas popular for group travel. The approach involves grouping the trip data by the designated pickup_zone and calculating the mean (average) passenger_count for each zone. These zones are then sorted in descending order based on this average passenger count. Finally, a horizontal bar chart is generated using Seaborn to visualize the top 15 zones, displaying the zone name against its calculated average passenger count, using the 'viridis' color palette for visual appeal.

The resulting bar chart effectively ranks pickup zones by their tendency for group travel. Brooklyn Navy Yard stands out with the highest average passenger count, approaching 6 passengers per trip on average. Following this are DUMBO/Vinegar Hill and Maspeth, both showing high averages significantly exceeding 4 passengers. Several other zones, including Greenwich Village North and Financial District South, also demonstrate averages well above the typical single-passenger trip, indicating their frequent use as starting points for shared rides or groups. Zones lower on the list, such as Ozone Park and Rego Park, still average around 2 passengers, suggesting more than solo travel but less pronounced group activity compared to the top-ranked zones.



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

This analysis focuses on determining the percentage of trips incurring extra charges, broken down by the hour of pickup. The methodology involves grouping the trip data by pickup_hour. For each hour, it calculates the proportion of trips where the 'extra' charge field is greater than zero using .apply() with a lambda function (x > 0).mean(). This proportion is then converted to a percentage and sorted in descending order. The results are visualized as a bar chart showing the percentage of trips with extra charges for each hour where such charges occurred.

The resulting graph highlights a distinct temporal pattern for the application of extra charges. Hour 12 (noon) shows the highest incidence, with 50% of trips including an extra charge, closely followed by Hour 0 (midnight) at approximately 47%. Several other late-night or early-morning hours (Hours 4, 19, 3) also exhibit a notable percentage (~33%) of trips with extra charges. The frequency then decreases, with a significant number of hours (particularly standard daytime and evening hours like 7-10 AM, 1-5 PM, etc.) showing zero trips with extra charges in this dataset. This pattern strongly suggests that extra charges (potentially representing overnight, rush hour, or specific surcharges) are most common around noon and during late-night/early-morning periods.

4. Conclusions

4.1. Final Insights and Recommendations

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

Predictive Fleet Positioning: Pre-position vehicles in or near zones identified with high pickup volumes (like Zone 56 overall, or specific zones during their peak hours like Zone 150 around 1 PM) before the anticipated demand surge (e.g., shortly before midnight overall, before 1 PM midday, especially on weekdays).

Dynamic Shift Scheduling/Incentives: Align driver availability with demand peaks. Offer incentives for drivers to work during the absolute peak (Hour 0, especially weekends) and the secondary weekday peak (around Hour 13). Consider strategies for lower coverage during very low demand hours (e.g., 3-5 AM).

Weekend vs. Weekday Strategy Differentiation: Implement distinct dispatch logic. Focus resources heavily on late-night (Hour 0) coverage for weekends, whereas weekday strategies should prioritize midday (around Hour 13) and potentially commute hours, with less emphasis on midnight.

Address Zone Imbalances: For zones with very low pickup/dropoff ratios (primarily destinations like Zone 196, 75, 238 etc.), implement strategies to prevent drivers from getting stranded. This could involve routing algorithms that proactively direct drivers towards nearby higher-demand pickup zones after a dropoff, or offering small incentives for pickups originating from these low-ratio zones.

Optimize for Group Travel: In zones and during times with high average passenger counts (e.g., Brooklyn Navy Yard, DUMBO; or Sat 9 AM, Mon 1 AM), consider prioritizing or dispatching larger vehicles (if available) to meet potential group demand more efficiently.

Route Efficiency Analysis: Investigate the exceptionally high average fare-per-mile instances (e.g., Hour 16, Wed/Thu). These might indicate severe

traffic congestion or inefficient routes during those times. Routing algorithms could potentially deprioritize or suggest alternatives for known congestion hotspots during specific hours/days.

Nighttime Dispatch Focus: Recognize that nighttime pickups are scattered but dropoffs are concentrated (e.g., Zones 238, 263). Ensure drivers completing late-night dropoffs in these zones are efficiently routed to potential subsequent pickups, possibly anticipating demand originating near these popular destinations.

Leverage Extra Charge Data: Inform drivers about the hours (noon, late night/early morning) where extra charges are most prevalent, as this impacts potential earnings and may influence shift preferences or willingness to accept trips during those times. Dispatch logic could factor this in when assigning trips.

Minimize Idle Cruising: Direct vacant taxis away from over-supplied or low-demand zones toward areas with higher predicted demand, reducing unnecessary mileage, fuel costs, and city congestion

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.

Peak Hour Concentration:

- **Overall**: Ensure a higher concentration of available cabs in and around historically high-volume pickup zones (like Zone 56, based on earlier analysis) leading up to and during the overall peak hours (midnight Hour 0 and midday Hour 13).
- **Weekends:** Focus fleet density heavily near entertainment districts, residential areas with nightlife, and key dropoff zones (like 238, 263) in the late evening leading into the dominant Hour 0 peak.
- **Weekdays:** Position more cabs near business districts, commercial centers, and potentially transport hubs approaching the Hour 13 peak. Also consider moderate presence near residential zones for morning commute pickups.

Anticipatory Staging: Don't just react to demand. Use the hourly trends to stage cabs near predicted high-demand zones 30-60 minutes before the

peak hits (e.g., near Zone 150 before 1 PM if that remains a consistent dropoff peak, anticipating potential return trips or nearby pickups).

Targeted Zone Balancing:

- **High Dropoff/Low Pickup Zones**: Station fewer idle cabs directly within zones identified as primarily destinations (e.g., Zone 196). Instead, position them on the periphery or along likely exit routes to capture outbound trips or be ready for dispatch to nearby higher-demand areas.
- **High Pickup/Low Dropoff Zones:** Maintain a steady supply of cabs in zones known to be strong origin points, ensuring quick pickup times.

Time-Specific Zone Strategy:

- **Nighttime (11 PM - 5 AM):** Given scattered pickups but concentrated dropoffs (like 238, 263), position some cabs near these key nighttime dropoff zones to reduce travel time for potential subsequent pickups originating nearby, rather than having them return empty to distant depots.
- **Early Morning Low (3-5 AM):** Reduce the number of actively roaming cabs significantly. Position a skeleton fleet strategically near 24-hour locations (airports if applicable, major hospitals, key transport hubs) rather than widespread coverage.

Group Travel Hubs: While overall numbers might be lower, ensure reliable, though perhaps not excessive, coverage near zones identified with high average passenger counts (e.g., Brooklyn Navy Yard, DUMBO), especially during times those zones are active, as these potentially represent different travel needs (e.g., airport runs, families).

Dynamic Relocation Based on Real-time Data: Supplement historical trends with real-time demand heatmaps. Encourage or automatically direct drivers from currently low-demand zones towards adjacent zones showing emerging demand or historically predicted peaks for that specific time/day.

Monthly/Seasonal Adjustments (If Applicable): Overlay monthly or seasonal patterns. Position cabs near tourist areas more heavily in peak season, near shopping centers during holidays, or adjust based on major events, modifying the baseline hourly/daily positioning strategy.

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

To maximize revenue while maintaining competitive rates, taxi operators should adopt a dynamic, data-driven pricing strategy that leverages both real-time and historical data. Implementing surge pricing during peak demand periods such as rush hours, holidays, or major events can increase revenue and encourage more drivers to be available, while lowering prices during off-peak times helps sustain ride volume and driver utilization. Regular benchmarking against competitors' pricing and promotions ensures rates remain attractive yet profitable, while predictive analytics models can forecast demand based on factors like time of day, location, and trip length to optimize fare structures. Incorporating external data such as weather conditions, public transit disruptions, and city events further refines pricing decisions by accounting for factors that influence demand. Transparency in fare changes and providing upfront estimates build customer trust and reduce dissatisfaction. Continuous monitoring through A/B testing and data visualization allows operators to assess the impact of pricing adjustments on ride volume, revenue, and customer satisfaction, enabling ongoing refinement. Additionally, personalized promotions targeted at specific customer segments during low-demand periods can balance occupancy and revenue without broadly lowering fares. Together, these data-driven approaches create a flexible and responsive pricing framework that maximizes revenue while remaining competitive in the market.