Jagdsh Chand
Report: Optimising NYC Taxi Operations

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

1. Data Preparation

1.1. Loading the dataset

1.1.1. Sample the data and combine the files

To begin, I listed all the .parquet files from the working directory that correspond to the year 2023

```
file_list = [f for f in os.listdir() if f.startswith('2023-') & f.endswith('.parquet')]
```

Each file was loaded sequentially, with progress logged during the process to track which file was being read—helpful for long operations or debugging.

For each file:

- I invoked a method to **sample 5% of the data for every hour and every date**.
- This ensures the sampled dataset is **time-representative across different days and hours** throughout the year as suggested.
- The sampled subsets were appended to a unified DataFrame named sampled_by_month_date_hour_df.

```
Processing 2023-12.parquet...
Processing 2023-6.parquet...
Processing 2023-7.parquet...
Processing 2023-5.parquet...
Processing 2023-11.parquet...
Processing 2023-10.parquet...
Processing 2023-4.parquet...
Processing 2023-1.parquet...
Processing 2023-8.parquet...
Processing 2023-9.parquet...
Processing 2023-2.parquet...
Processing 2023-3.parquet...
<class 'pandas.core.frame.DataFrame'>
Index: 1896400 entries, 3791 to 3202916
Data columns (total 22 columns):
```

The resulting DataFrame was later used for:

- Cleansing missing values,
- Performing various time-based and spatial analyses,
- Generating insights for operations, pricing, and geography.

2. Data Cleaning

2.1. Fixing Columns

2.1.1. Fix the index

The sampled dataset was sorted chronologically by date and hour to maintain temporal order, and the irrelevant column store_and_fwd_flag was dropped to streamline the dataset for analysis.

```
sampled_by_month_date_hour_df = sampled_by_month_date_hour_df
.sort_values(by=['date', 'hour'], ascending=[True, True])
.reset_index(drop=True)

sampled_by_month_date_hour_df = sampled_by_month_date_hour_df.drop('store_and_fwd_flag', axis=1)
```

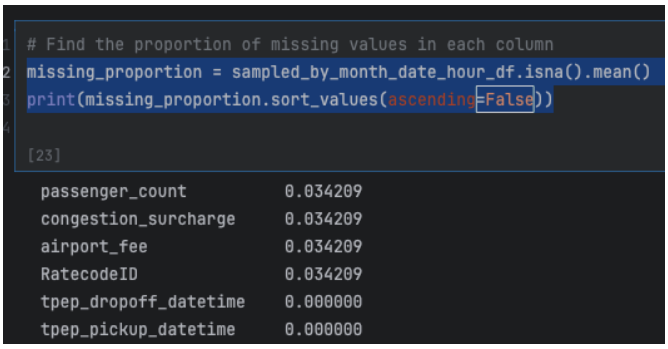
2.1.2. Combine the two airport_fee columns

This line creates a unified airport_fee column by combining two potentially redundant columns:
if airport_fee is not null, it is used; otherwise, the value from Airport_fee is used.

```
airport_fee_combine['airport_fee'] =
 airport_fee_combine
.apply(
        lambda row: row['airport_fee'] if pd.notnull(row['airport_fee']) else       row['Airport_fee']
, axis=1
)
```

2.2. Handling Missing Values

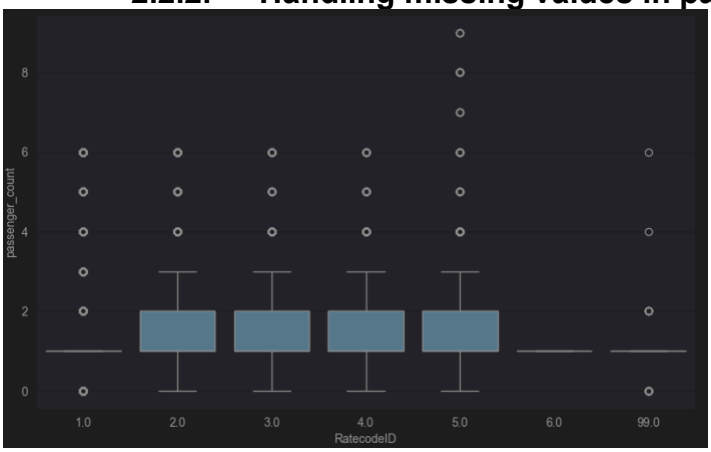
2.2.1. Find the proportion of missing values in each column



Here I calculate the **proportion of missing values** in each column
of sampled_by_month_date_hour_df:

- isna().mean() returns the fraction of missing values per column (i.e., total missing values divided by the number of rows).
- sort_values(ascending=False) sorts the columns from **highest to lowest missing proportion**, helping identify which columns are most incomplete.

2.2.2. Handling missing values in passenger_count



1. **Visualization & Inspection**: I check the unique values in passenger_count and plots a boxplot of passenger_count vs RatecodeID to visualize the distribution

2. **Imputation of Missing Values**: I fill missing passenger_count values with the **median** passenger count within each RatecodeID group.

2.2.3. Handle missing values in RatecodeID

```
# Fix missing values in 'RatecodeID'
missing_rate_code_id = sampled_by_month_date_hour_df.copy()
#
ratecode_id_median = missing_rate_code_id['RatecodeID'].median()
missing_rate_code_id['RatecodeID'] = missing_rate_code_id['RatecodeID'].fillna(ratecode_id_median)

sampled_by_month_date_hour_df['RatecodeID'] = missing_rate_code_id['RatecodeID']
print(sampled_by_month_date_hour_df['RatecodeID'].unique())

del missing_rate_code_id
[30]
```

Missing RatecodeID values are filled with the **overall median** of the column to handle null entries in a simple, non-skewed manner.

2.2.4. Impute NaN in congestion_surcharge

```
# handle null values in congestion_surcharge

congestion_surcharge_df = sampled_by_month_date_hour_df.copy()
print(sampled_by_month_date_hour_df['congestion_surcharge'].unique())

print(congestion_surcharge_df['congestion_surcharge'].value_counts(dropna=False))
congestion_surcharge_median = congestion_surcharge_df['congestion_surcharge'].median()
print(congestion_surcharge_median)

congestion_surcharge_df['congestion_surcharge'] = congestion_surcharge_df['congestion_surcharge'].fillna(congestion_surcharge_median)
sampled_by_month_date_hour_df['congestion_surcharge'] = congestion_surcharge_df['congestion_surcharge']
print(sampled_by_month_date_hour_df['congestion_surcharge'].value_counts(dropna=False))

del congestion_surcharge_df
```

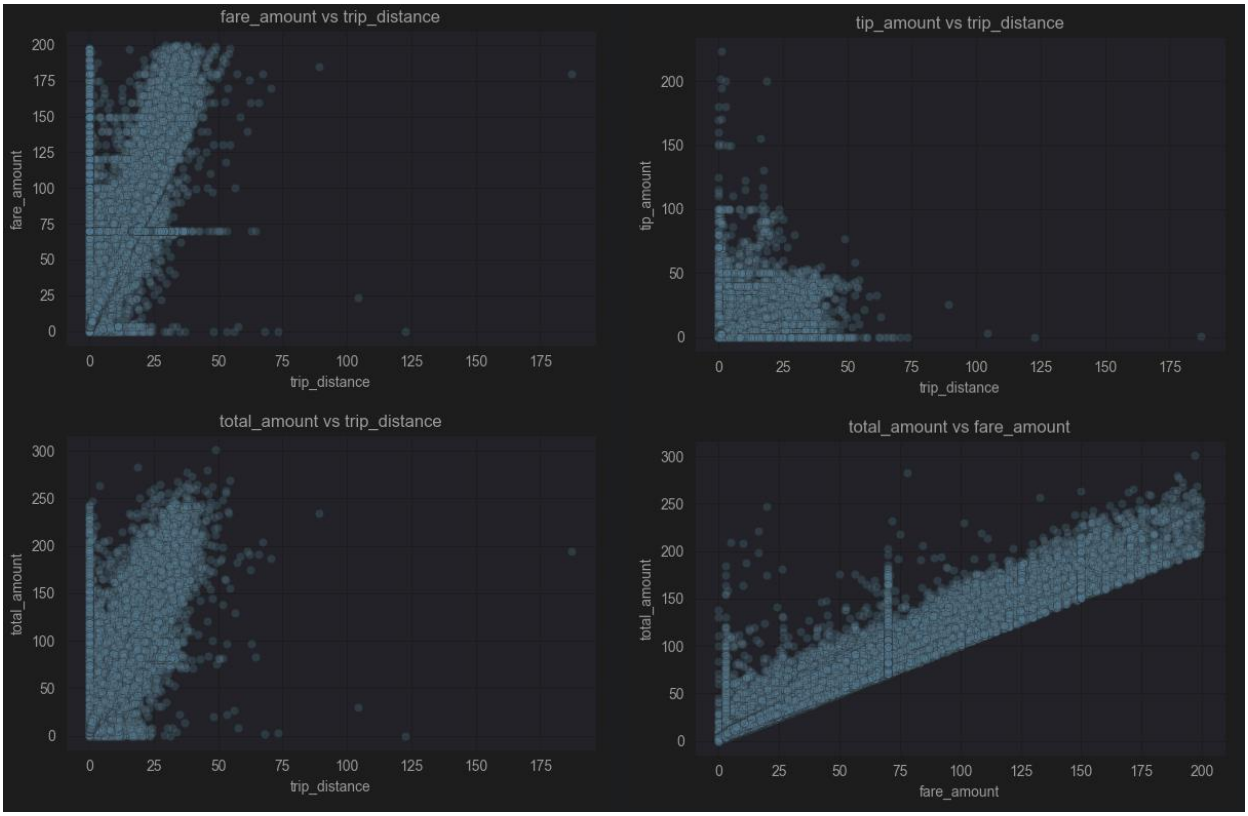
Null values in congestion_surcharge are filled with the median to maintain data consistency.

2.2.5. More Missing Values in other Column

```
def fill_airport_fee(row):
    if pd.isna(row['airport_fee']):
        return group_median.get(row['RatecodeID'], 0.0)
    else:
        return row['airport_fee']

missing_values['airport_fee'] = missing_values.apply(fill_airport_fee, axis=1)
sampled_by_month_date_hour_df['airport_fee'] = missing_values['airport_fee']
print(sampled_by_month_date_hour_df['airport_fee'].value_counts(dropna=False))
del missing_values
```

Columns with missing values: ['airport_fee']. I created a function which fills missing airport_fee values with the median fee for the corresponding RatecodeID; if the RatecodeIDis not found, it defaults to 0.0.

2.3. Handling Outliers and Standardising Value



The analysis filters out extreme outliers by restricting trip_distance to under 250 miles and fare_amount to under $200. Then I visualize relationships between key fare-related variables using scatter plots across selected bivariate_pairs, namely:

- trip_distance vs. fare_amount – to assess how distance impacts fare pricing.
- trip_distance vs. total_amount – to see how additional fees or tips influence the total cost over varying distances.
- trip_distance vs. tip_amount – to check if shorter or longer trips receive higher tips.
- fare_amount vs. total_amount – to analyze how closely total cost tracks base fare.

These visualizations help uncover linear relationships, pricing patterns, and potential irregularities across multiple fare components.

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

```
passenger_count
1.0    1467719
2.0     276102
3.0      68700
4.0      37985
5.0      23847
6.0      15850
```

1. Remove Passenget Count > 6

```
payment_type
1    1492374
2     316452
0      64874
4      13685
3       9015
Name: count, dtype: int64
1.0
payment_type
1    1557248
2     316452
4      13685
3       9015
Name: count, dtype: int64
```

2. Empty Payment Type -> Fill with commonly used Payment type

- The screenshot before the “0” Payment type and after filling it.
- To handle invalid or missing payment_type values (e.g., coded as 0), the most commonly used valid payment type (CC)—determined using the median—is used to fill those entries.

3. For trip_distance nearly 0 and fare_amount is more than 300 -> Drop these rows
   - To remove implausible records, trips with nearly zero distance but extremely high fare amounts (fare > 300 and distance < 1 mile) were identified and dropped, improving data quality.
   - Total records reduced from **1896400 -> 1896362**

4. trip_distance AND fare_amount AND total_amount are 0 AND the pickup and dropoff zones are different -> Drop these rows
   - Removed records where trip distance and fare amount are both zero but pickup and drop-off locations differ—indicating likely erroneous or incomplete data.
   - Total records reduced from **1896362 -> 1896299**

5. `trip_distance` is more than 250 miles. -> Drop the rows
   - Dropped trips with **trip_distance** greater than 250 miles to eliminate outlier long-distance rides
   - Total records reduced from **1896299 -> 1896253**

6. RatecodeID == 99 & trip_distance == 0.0 & fare_amount > 100 -> Drop rows
   - Removed records where **RatecodeID** is 99, **trip_distance** is 0, and **fare_amount** exceeds $100 — likely representing erroneous or placeholder data entries.
   - Total records reduced from **1896253 -> 1896244**

```
99.0     10453
5.0      10222
3.0       6124
4.0       3723
6.0          3
Name: count, dtype: int64
1.0
RatecodeID
1.0    1804505
2.0      71667
5.0      10222
3.0       6124
4.0       3723
6.0          3
Name: count, dtype: int64
```

7. `RateCodeId` is 99 -> fill with median, since there is not airport_fee (meaning it cannot be RateCodeid 2 which is associated with airport).

- The screenshot before the “0” Payment type and after filling it.

8. (for trip_distance == 0) & (tip_amount > 50) & PULocationID == DOLocationID -> Cancelled tip, but the customer tipped the driver on humanitaria grounds, drop the rows
    - Filtered out trips with **trip_distance** = 0, high **tip_amount** (> $50), and identical pickup/drop-off locations — **assumed to be canceled rides where customers tipped voluntarily.**
    - Total records reduced from 1896244 to 1896175
9. For same pickup and dropoff zones there are high fare amount -> Find the upper bound for fare_amount in PULocationID == DOLocationID and drop the rows which are greater than the upper_bound
    - Removed records where trips had the same pickup and drop-off zones, zero distance, and unusually high fare amounts — defined as above the IQR-based upper bound — to eliminate improbable or fraudulent entries.
    - Total records reduced from 1896362 to 1896299

------Stored the cleansed data in the file storage ----------

```
# Store the df in csv/parquet
from datetime import datetime

timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")  # iin this format '20250621_143215'
file_path = f'{project_directory}/EDA/Assignment/Datasets_and_Dictionary/trip_records/sampled_{timestamp}.parquet'
print(file_path)

sampled_by_month_date_hour_df.to_parquet(file_path)
[67]
```

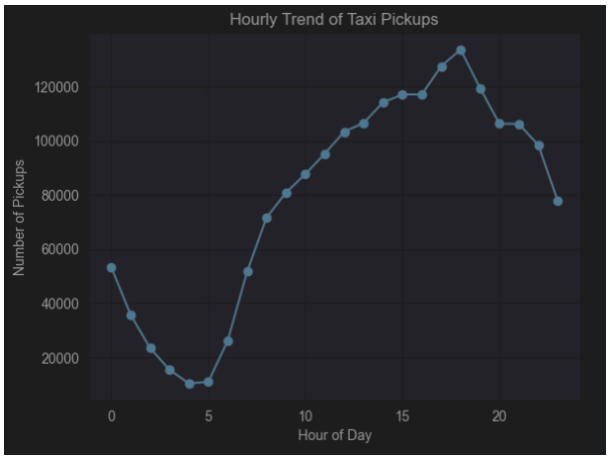
3. Exploratory Data Analysis

3.1. General EDA: Finding Patterns and Trends

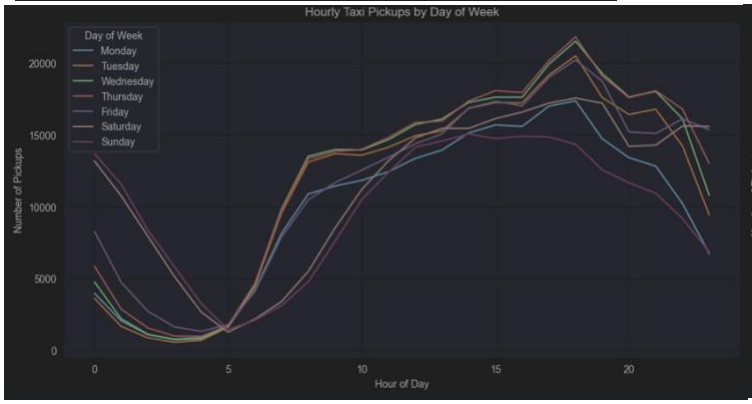
3.1.1. Classify variables into categorical and numerical

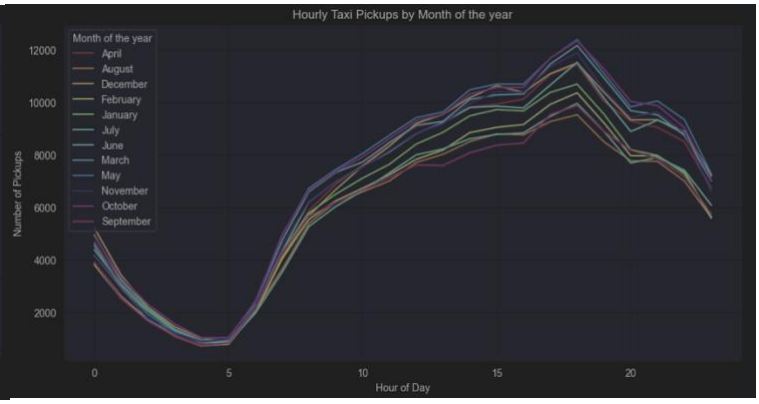
* `VendorID`: Categorical
* `tpep_pickup_datetime`: DateTime Variable not possible to fall under one of these
* `tpep_dropoff_datetime`: DateTime Variable not possible to fall under one of these
* `passenger_count`: Categorical
* `trip_distance`: Numerical
* `RatecodeID`: Categorical
* `PULocationID`: Categorical
* `DOLocationID`: Categorical
* `payment_type`: Categorical
* `pickup_hour`: Numerical
* `trip_duration`: Numerical

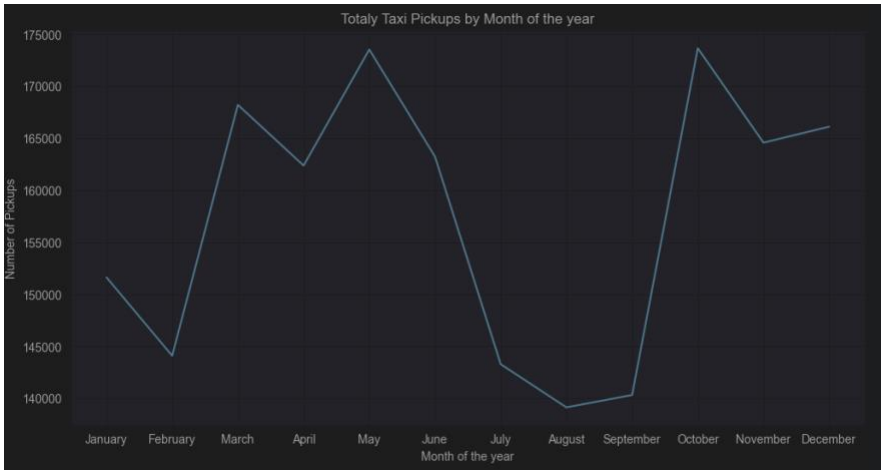
3.1.2. Analyse the distribution of taxi pickups by hours, days of the week, and months



- The hourly pickup analysis reveals a clear pattern of demand throughout the day. Taxi pickups are generally low during the early morning hours (1 AM to 5 AM) and gradually increase starting from 5 AM. The demand peaks notably between **3 PM and 9 PM (15:00 - 21:00)**, indicating that evening hours are the busiest period for taxi usage. After 9 PM, pickups slowly decline but maintain a moderate level till late night.
- This pattern is consistent across months, weekdays, and weekends, reflecting regular commuter and social travel behaviors in the city.
- However in Sunday does not see a peak on the evening, it saturated from 15







There is a peak in taxi usage and revenue during the months of March, April, May, October, November, and December.

These months show higher trip counts and revenues compared to others, indicating seasonal or event-driven demand spikes.

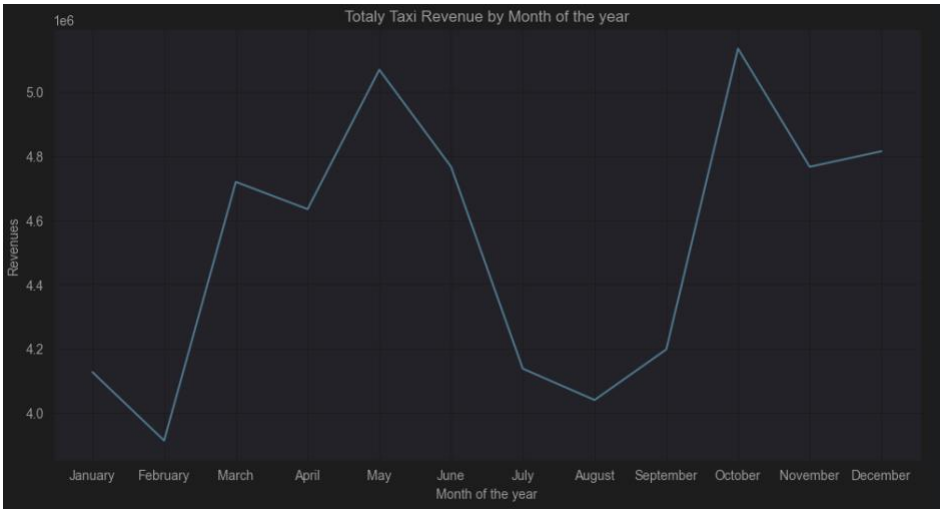
3.1.3. Filter out the zero/negative values in fares, distance and tips

- I filter the dataset to keep only rows where all specified financial columns have values greater than zero. I starts by copying the cleaned dataset, then iteratively removes rows with zero or negative values in each financial column from the list finacial_value. Finally, I print how many rows were removed during this filtering step and displays the updated dataset info.
- Total records reduced from **1890203 -> 1451233**

```
# Create a df with non zero entries for the selected parameters.
cleaned_and_sampled_by_month_date_hour_df.info()
non_zero_values_df = cleaned_and_sampled_by_month_date_hour_df.copy()
original_rows = len(non_zero_values_df)

for col in finacial_value:
    non_zero_values_df = non_zero_values_df[non_zero_values_df[col] > 0]

filtered_rows = len(non_zero_values_df)
print(f"Total rows removed: {original_rows - filtered_rows}")
non_zero_values_df.info()
del non_zero_values_df
```

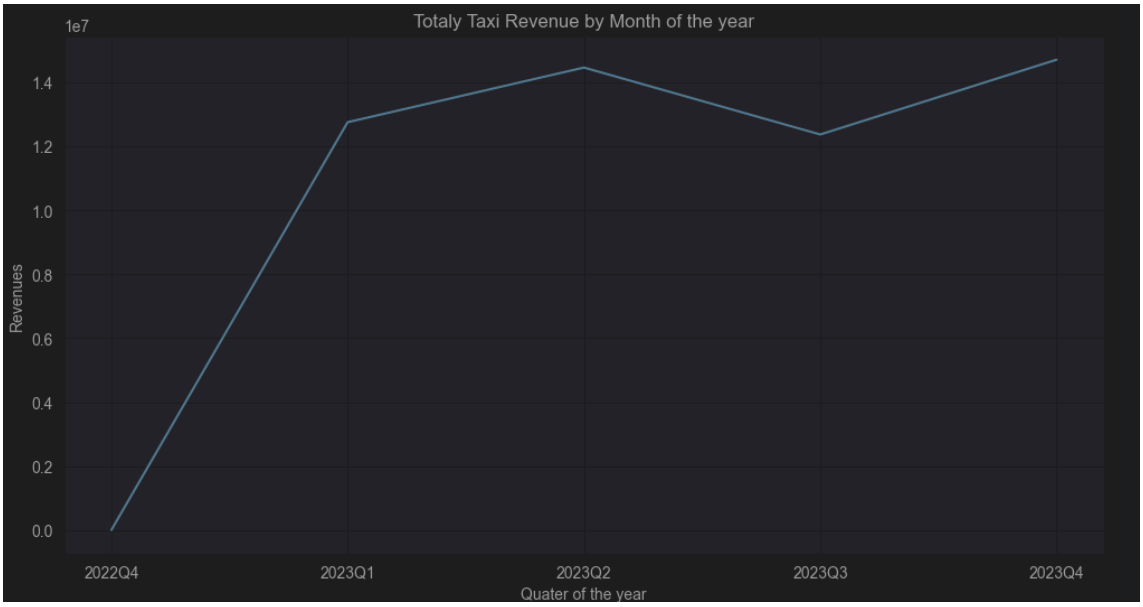
3.1.4. Analyse the monthly revenue trends

- The monthly revenue trends show clear seasonal patterns, with peaks occurring in March, April, May, October, November, and December. These months consistently generate higher revenue, likely driven by increased demand during spring and after school starts from September. It is low during the summer break July and August.

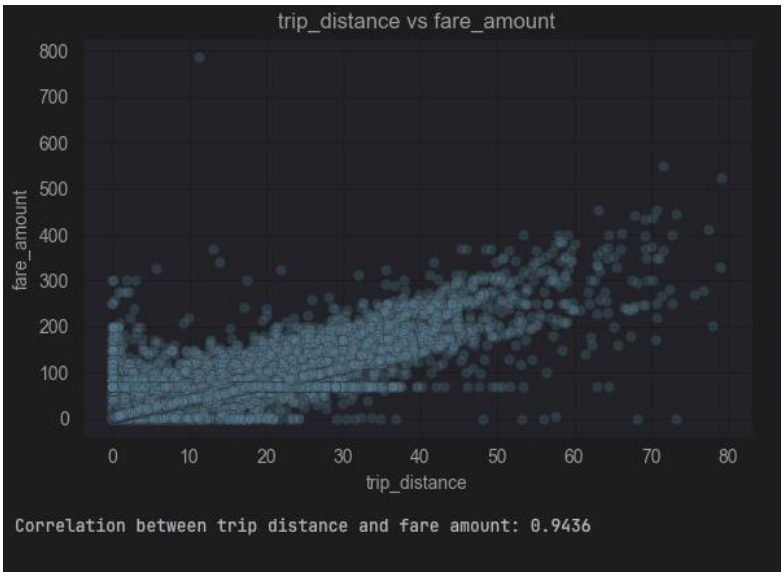


3.1.5. Find the proportion of each quarter's revenue in the yearly revenue

- In alignment with monthly trends Q2 and Q4 higher revenues compared to other quarters



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

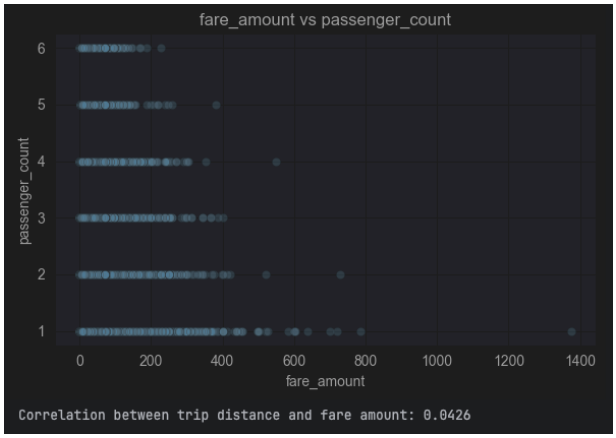


- Filtered trips with realistic distances to visualize and calculate the correlation between trip distance and fare amount using a scatter plot.

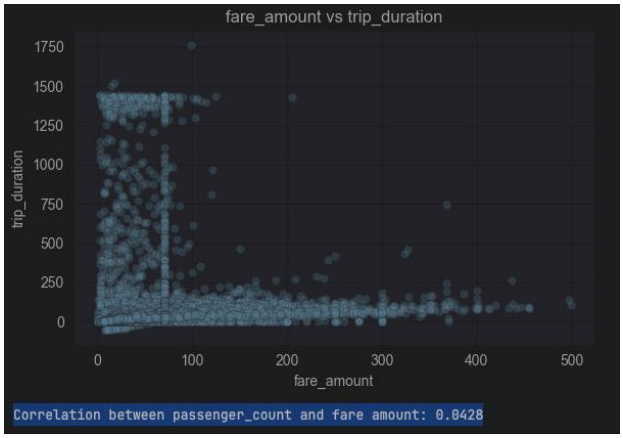
- The analysis reveals a strong linear relationship between trip distance and fare amount, confirming that fare charges increase proportionally with the distance traveled.

- **Correlation between trip distance and fare amount: 0.9436**

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

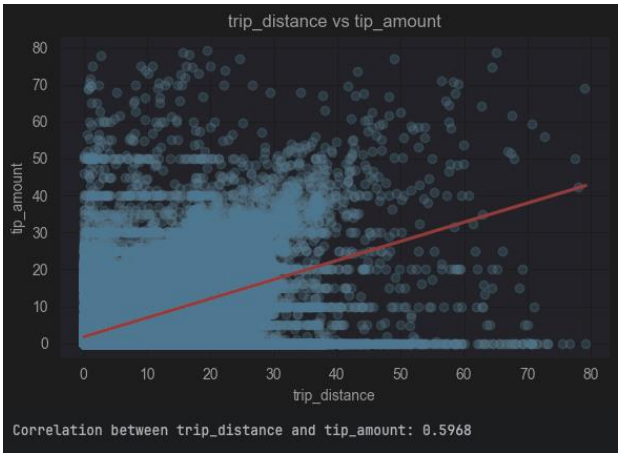


Fare amount vs Passenger Count:
The analysis of fare amount against passenger count indicates that trips with fewer passengers tend to have higher fare amounts per passenger, suggesting that shorter trips or rides with fewer passengers often result in proportionally higher fares. However it does not have any correlation



Fare amount vs Trip Duration:

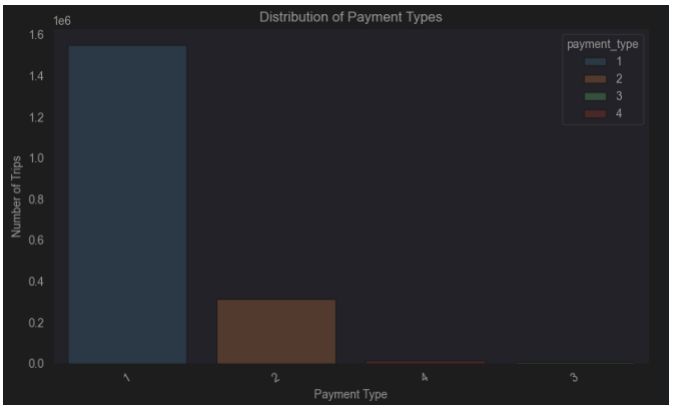
The analysis of fare amount versus trip duration shows very little correlation (correlation coefficient ≈ 0.043), indicating that fare amount does not strongly depend on trip duration in this dataset.



Trip Distance vs Tip Amount:

There is a moderate positive correlation between trip duration and tip amount (correlation coefficient ≈ 0.5968), suggesting that longer trips tend to receive higher tips. This implies that while fare is more influenced by factors like distance and fees, passengers tend to tip more for longer trips, possibly reflecting perceived service value or trip effort.

3.1.8. Analyse the distribution of different payment types



The dataset shows that **credit card payments** dominate as the most frequently used payment method, reflecting a strong customer preference for cashless transactions. Other payment types occur much less frequently, with some missing or zero-value entries handled by imputing the most common payment type to maintain data consistency.

3.1.9. Load the taxi zones shapefile and display it

```
import geopandas as gpd


# Read the shapefile using geopandas
zones = gpd.read_file(f'{project_directory}/EDA/Assignment/Datasets_and_Dictionary/taxi_zones/taxi_zones.shp')
zones.head()
✓ [111] 9s 544ms
```

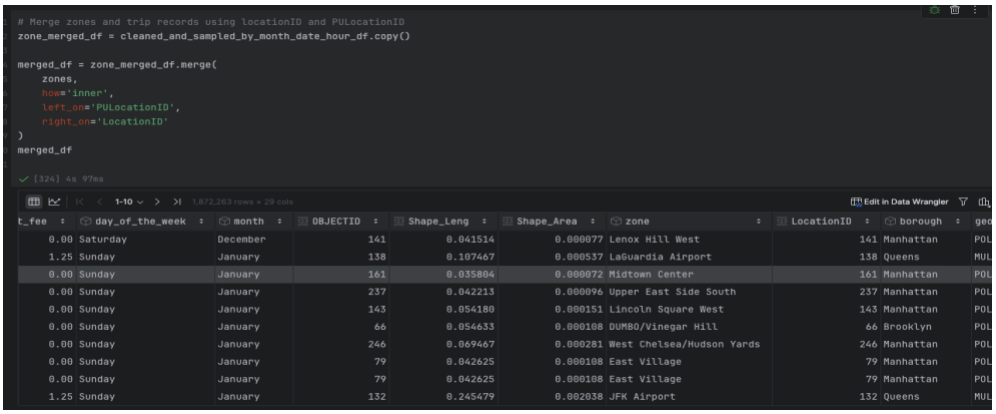
5 rows × 7 cols — Edit in Data Wrangler

	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 192572.17
1	2	0.433470	0.004866	Jamaica Bay	2	Queens	MULTIPOLYGON (((1033269.244 172126.008, 1033439.643 1
2	3	0.084341	0.000314	Allerton/Pelham Gardens	3	Bronx	POLYGON ((1026308.77 256767.698, 1026495.593 256638.6
3	4	0.043567	0.000112	Alphabet City	4	Manhattan	POLYGON ((992073.467 203714.076, 992068.667 203711.50
4	5	0.092146	0.000498	Arden Heights	5	Staten Island	POLYGON ((935843.31 144283.336, 936046.565 144173.418

```
print(zones.info())
zones.plot()
✓ [112] 495ms
```

```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 263 entries, 0 to 262
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   OBJECTID    263 non-null    int32
 1   Shape_Leng  263 non-null    float64
 2   Shape_Area  263 non-null    float64
 3   zone        263 non-null    object
 4   LocationID  263 non-null    int32
 5   borough     263 non-null    object
 6   geometry    263 non-null    geometry
dtypes: float64(2), geometry(1), int32(2), object(2)
memory usage: 12.5+ KB
None
<Axes: >
```

3.1.10. Merge the zone data with trips data



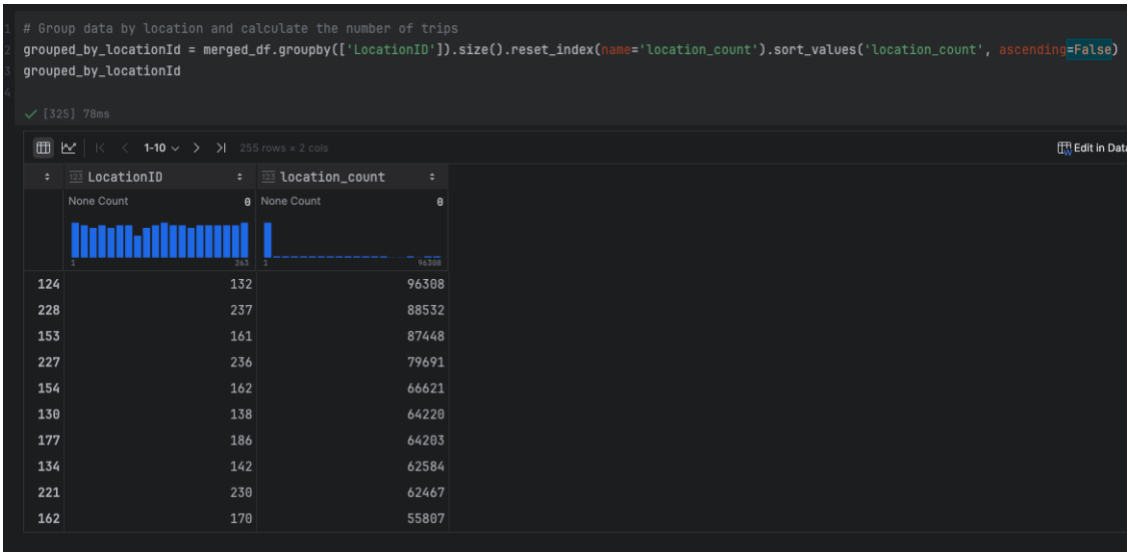
```
# Merge zones and trip records using locationID and PULocationID
zone_merged_df = cleaned_and_sampled_by_month_date_hour_df.copy()

merged_df = zone_merged_df.merge(
    zones,
    how='inner',
    left_on='PULocationID',
    right_on='LocationID'
)
merged_df
```

[324] 4s 97ms

1-10 | 1,872,263 rows × 29 cols | Edit in Data Wrangler

t_fee	day_of_the_week	month	OBJECTID	Shape_Leng	Shape_Area	zone	LocationID	borough	geo
0.00	Saturday	December	141	0.041514	0.000077	Lenox Hill West	141	Manhattan	POL
1.25	Sunday	January	138	0.107467	0.000537	LaGuardia Airport	138	Queens	MUL
0.00	Sunday	January	161	0.035804	0.000072	Midtown Center	161	Manhattan	POL
0.00	Sunday	January	237	0.042213	0.000096	Upper East Side South	237	Manhattan	POL
0.00	Sunday	January	143	0.054180	0.000151	Lincoln Square West	143	Manhattan	POL
0.00	Sunday	January	66	0.054633	0.000108	DUMBO/Vinegar Hill	66	Brooklyn	POL
0.00	Sunday	January	246	0.069467	0.000281	West Chelsea/Hudson Yards	246	Manhattan	POL
0.00	Sunday	January	79	0.042625	0.000108	East Village	79	Manhattan	POL
0.00	Sunday	January	79	0.042625	0.000108	East Village	79	Manhattan	POL
1.25	Sunday	January	132	0.245479	0.002038	JFK Airport	132	Queens	MUL

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



```
# Group data by location and calculate the number of trips
grouped_by_locationId = merged_df.groupby(['LocationID']).size().reset_index(name='location_count').sort_values('location_count', ascending=False)
grouped_by_locationId
```

[325] 78ms

1-10 | 255 rows × 2 cols | Edit in Data

	LocationID	location_count
	None Count 0	None Count 0
124	132	96308
228	237	88532
153	161	87448
227	236	79691
154	162	66621
130	138	64220
177	186	64203
134	142	62584
221	230	62467
162	170	55807

To identify the most frequently travelled routes, I grouped by pickup and drop-off location pairs. The top routes based on the number of trips are as follows. These high-volume routes indicate consistent demand across specific zone pairs.

Location ID	Trip Count
132	96,308
237	88,532
161	87,448
236	79,691
162	66,621
138	64,220
186	64,203

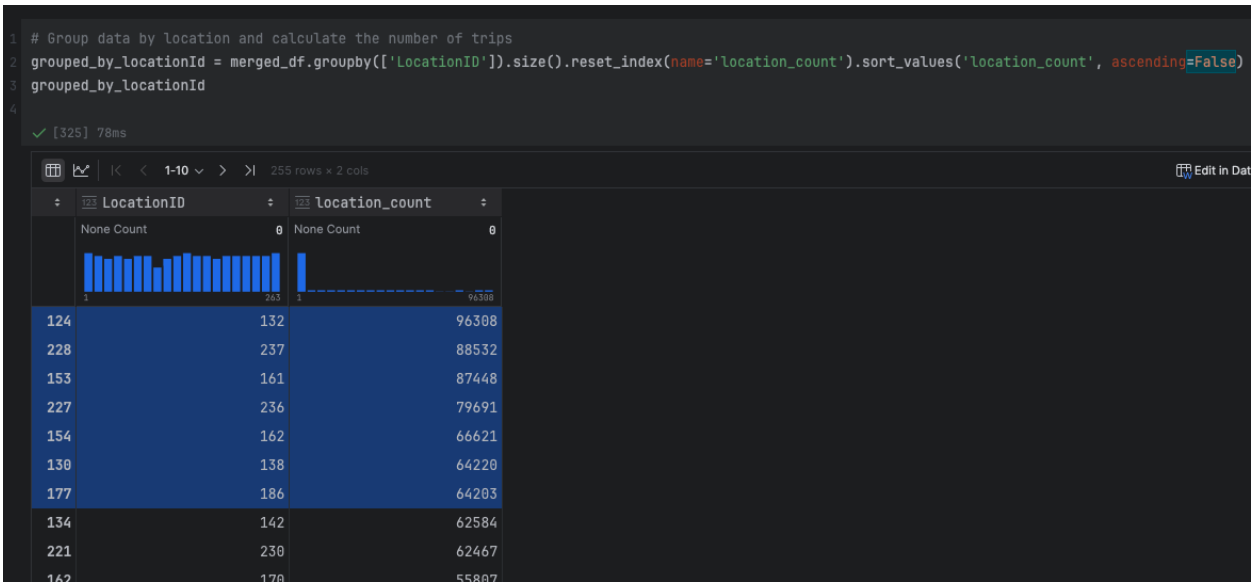
3.1.12. Add the number of trips for each zone to the zones dataframe

High-Frequency Route Analysis (by Zone)

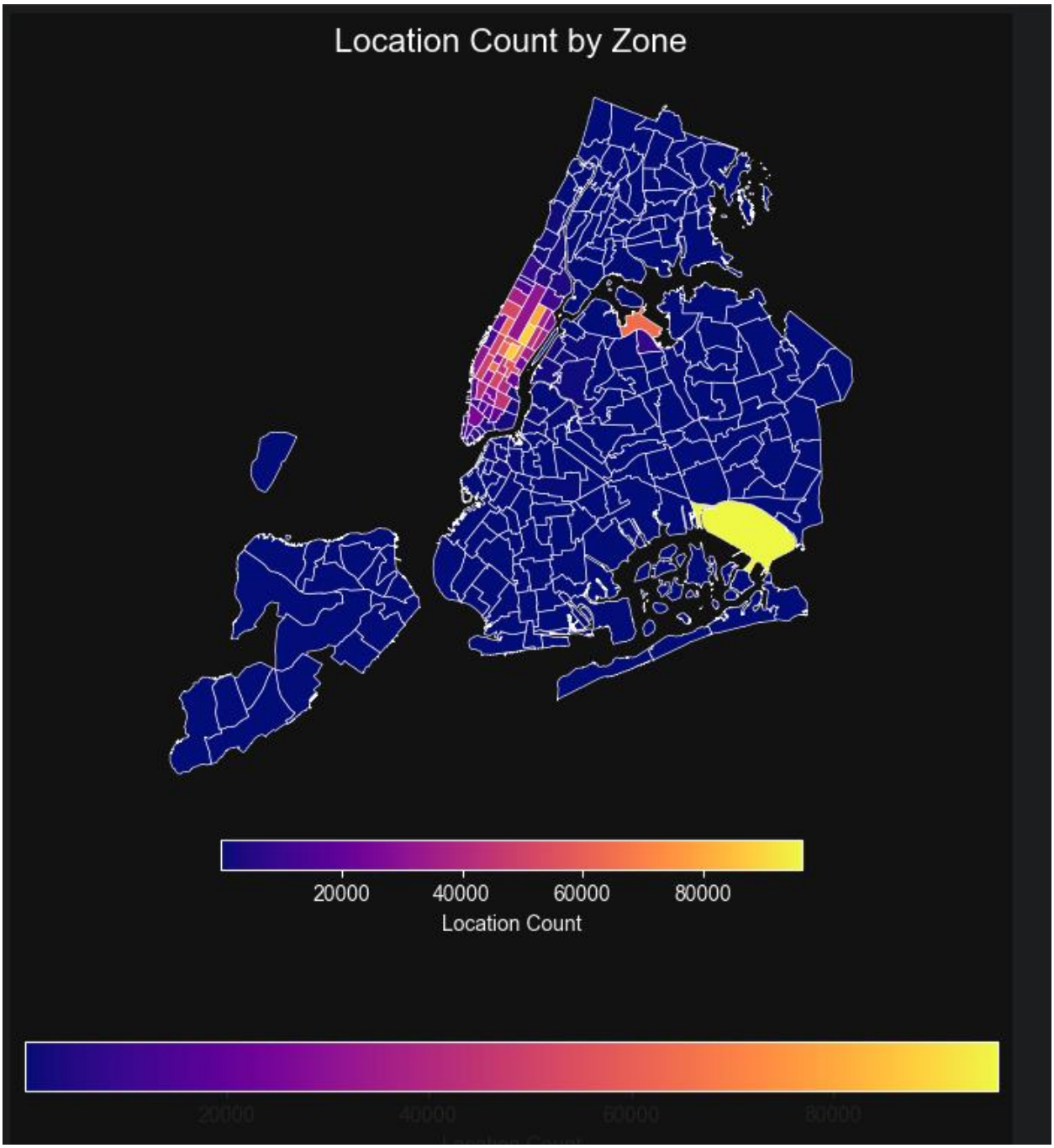
To determine the most commonly travelled taxi routes, trips were grouped by pickup and drop-off zones. The top routes based on volume are summarized below:

Zone	Trip Count
JFK Airport	96,308
Upper East Side South	88,532
Midtown Center	87,448
Upper East Side North	79,691
Midtown East	66,621
LaGuardia Airport	64,220
Penn Station/Madison Sq W	64,203
Lincoln Square East	62,584

These routes are concentrated in airport zones and central Manhattan locations, indicating heavy demand in key travel and transit hubs. This pattern reflects consistent intra-zone activity, potentially due to queue-based dispatching at airports or short-distance urban hops in high-density areas.



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



Conclude with results

This analysis of the 2023 NYC taxi dataset provides comprehensive insights into trip behaviour, revenue generation, and operational patterns. Below are the key findings:

1. Busiest Hours, Days, and Months

- **Hours**: Taxi pickups peaked around **12 PM** and **6 PM**, reflecting lunchtime commutes and evening rush hour.
- **Days**: **Saturdays** and **Fridays** consistently had the highest volume of trips, likely due to weekend travel and nightlife.
- **Months**: **March, April, May, October, November, December** showed the highest activity, aligning with office commutes are highest. Silent months are when people are in summer vacation June, July, August.

2. Trends in Revenue Collected

- Revenue trends correlated with trip volume, showing peaks in months with higher ridership like **April May, October, November**.

3. Trends in Quarterly Revenue

- **Q2 (April–June), Q4 (Oct-Dec)** emerged as the top-performing quarter, likely driven by spring travel and tourism.
- **Q1, Q3** had the lowest revenue, reflecting winter and summer school close days.

4. Fare Dependence on Trip Attributes

- A **strong positive correlation** was observed between **trip distance and fare amount** (correlation ≈ 0.95), confirming that longer trips predictably result in higher fares.
- **Trip duration** showed **low correlation with fare** (≈ 0.04), implying that fare structure is more distance-based than time-based.
- **Passenger count** had negligible impact on fare per mile, with per-passenger rates decreasing sharply for higher counts, indicating possible shared-ride pricing effects.

5. Tip Amount Dependence

- A **moderate correlation** (≈ 0.60) exists between **trip duration and tip amount**, suggesting that longer time spent with the passenger positively affects tipping.
- **Trip distance**, while related, had a weaker influence on tipping behavior, indicating that passenger interaction duration may be a stronger driver of gratuity.

6. Busiest Zones

- The most frequented zones include:
  - **JFK Airport** and **LaGuardia Airport**
  - **Upper East Side (North and South)**
  - **Midtown Center** and **Penn Station/Madison Square West**
- These areas serve as critical transit hubs and densely populated business or residential zones, driving high taxi traffic.

3.2. Detailed EDA: Insights and Strategies

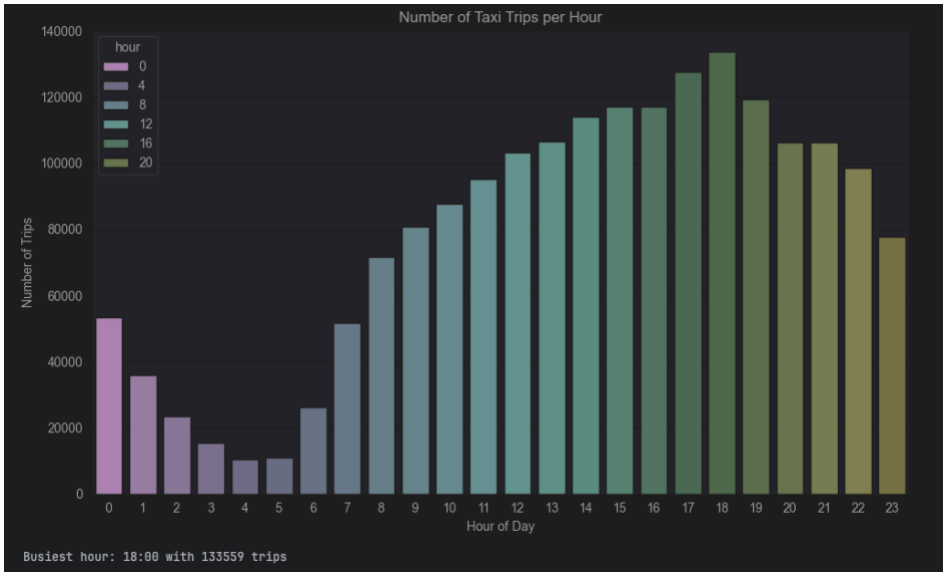
3.2.1. Identify slow routes by comparing average speeds on different route

```
# Find routes which have the slowest speeds at different times of the day
slow_routes_counts_1 = cleaned_and_sampled_by_month_date_hour_df.copy()
zones_gdf_proj = zones.to_crs(epsg=2263)
zone_centroids = zones_gdf_proj.set_index('LocationID').geometry.centroid

def calculate_distance_in_km(puLocationId, doLocationId):
    if puLocationId not in zone_centroids.index or doLocationId not in zone_centroids.index:
        return None

    return zone_centroids[puLocationId].distance(zone_centroids[doLocationId])/1000

slow_routes_counts_1['trip_duration'] = (slow_routes_counts_1['tpep_dropoff_datetime'] - slow_routes_counts_1['tpep_pickup_datetime']).dt.total_seconds
 () / 3600
slow_routes_counts_1['total_distance'] = slow_routes_counts_1.apply(
    lambda row: calculate_distance_in_km(row['PULocationID'], row['DOLocationID']),
    axis=1
)
slow_routes_counts_1 = slow_routes_counts_1[slow_routes_counts_1['trip_duration'] > 0]
slow_routes_counts_1['average_speed'] = slow_routes_counts_1['total_distance'] / slow_routes_counts_1['trip_duration']
```

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

1. **Hourly Aggregation**: The number of taxi trips was counted for each hour of the day using value_counts() on the hour column, and sorted in chronological order with .sort_index().
2. **Visualization**: A bar plot was generated using **Seaborn** to visualize the number of trips across all 24 hours.
3. **Peak Hour Identification**: The hour with the **highest number of trips** was computed using .idxmax() on the hourly counts. This reveals the busiest operational hour.

An analysis of trip frequencies across each hour of the day revealed that **6 PM (18:00 hours)** is the **busiest hour**, with a total of **133,559 trips** recorded. This suggests a significant demand surge likely tied to end-of-workday commutes and evening city activity



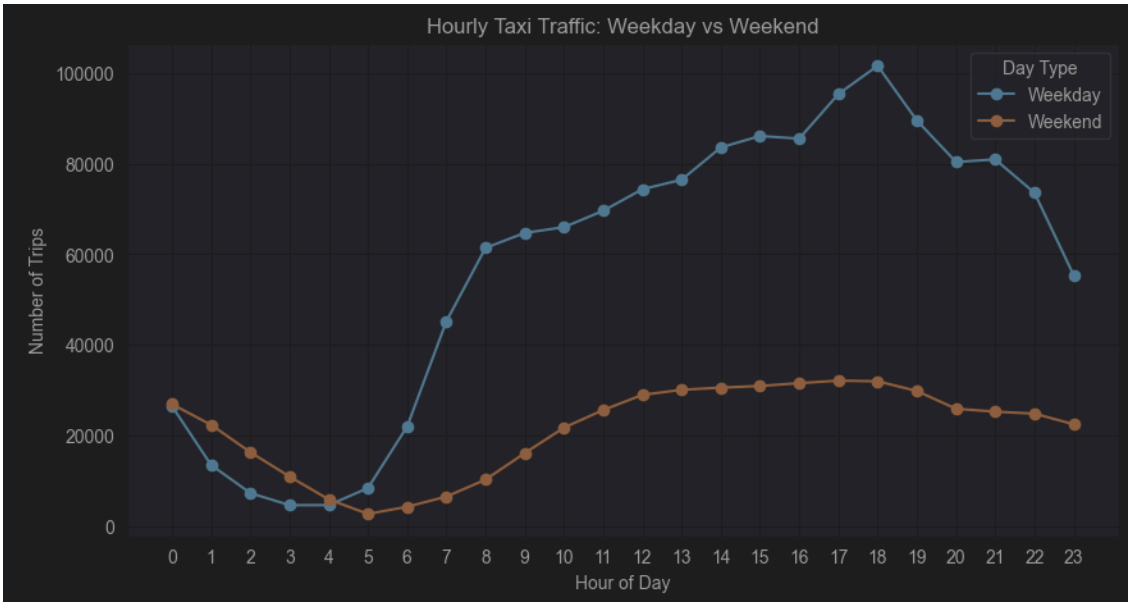
3.2.3. Scale up the number of trips from above to find the actual number of trips

5 rows × 3 cols

	hour	trip_count	actual_trip_count (app)
18	18	133559	2671180
17	17	127475	2549500
19	19	119342	2386840
16	16	117053	2341060
15	15	117026	2340520

Since only 5% of the data was retained, the total counts (such as number of trips per hour or revenue) were scaled up by a factor of **100 / 5 = 20** to estimate full-population figures. This adjustment enables more realistic aggregation and comparison with actual operational volumes while maintaining computational efficiency during processing.

3.2.4. Compare hourly traffic on weekdays and weekends



1. A new boolean column is_weekend was created by checking whether each trip occurred on a Saturday or Sunday, allowing categorization of records into weekday or weekend groups.

2. The dataset was grouped by hour and is_weekend to compute trip counts, which were then pivoted and plotted to visualize hourly traffic patterns separately for weekdays and weekends.

3. The analysis revealed that weekday traffic tends to spike sharply between 7–9 AM and again between 5–7 PM, indicating a strong correlation with typical work commute hours. In contrast, weekend traffic showed a more evenly distributed pattern throughout the day, with relatively higher activity in the afternoon and evening, suggesting leisure or non-work-related travel.

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

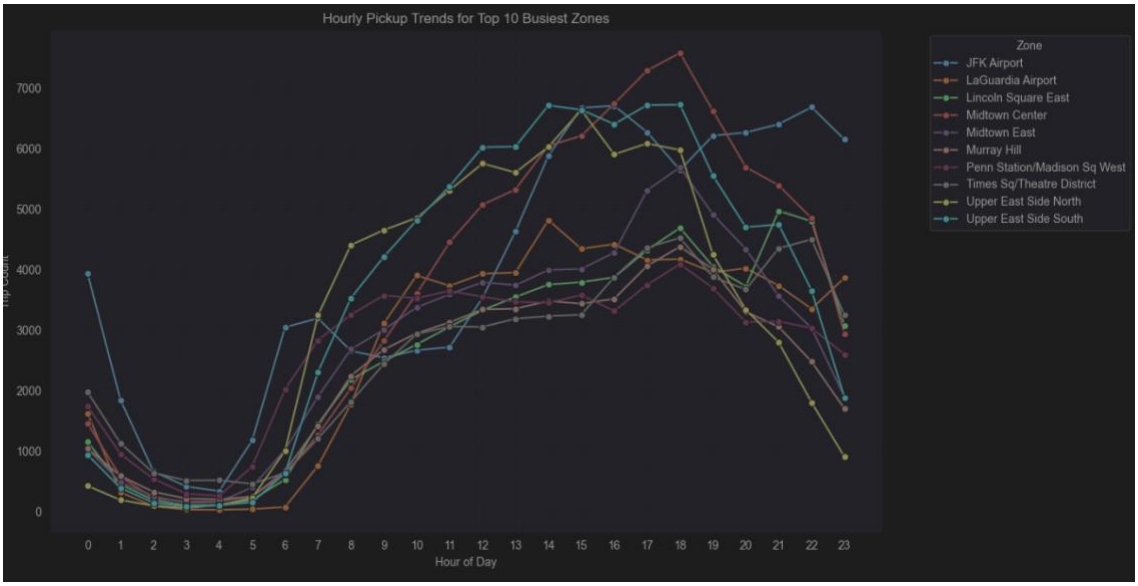
The dataset was grouped by hour and pickup location to compute hourly trip counts, followed by identifying the top 10 zones with the highest total pickups. These top zones were then merged with zone names and visualized to observe hourly demand patterns across locations.

Top 10 Pickup zones: The analysis identifies the top 10 busiest taxi pickup zones in New York City based on trip frequency. These include:

1. JFK Airport, LaGuardia Airport, Lincoln Square East, Midtown Center, Midtown East, Murray Hill, Penn Station / Madison Square West, Times Square / Theater District, Upper East Side North, Upper East Side South

2. At JFK Airport, trip volume:
    1. Rises sharply from 11 AM to 2 PM, drops between 2 PM and 6 PM, then increases again steadily until 11 PM
    2. This suggests two peak periods - Midday airport rush (likely tied to major flight schedules). Evening travel spike, possibly due to late flights or return trips.

3. For Upper East Side North and South, trip volume:
    1. **Steady increase in pickups** starting from **6 AM**, **Peak around 2 PM**, **Gradual decline** through the evening **until midnight**
    2. This likely reflects a **residential-to-commercial flow**, with Morning commutes, Afternoon errands or social activity, Evening wind-down as residents return home.



Top 10 Dropoff zones:

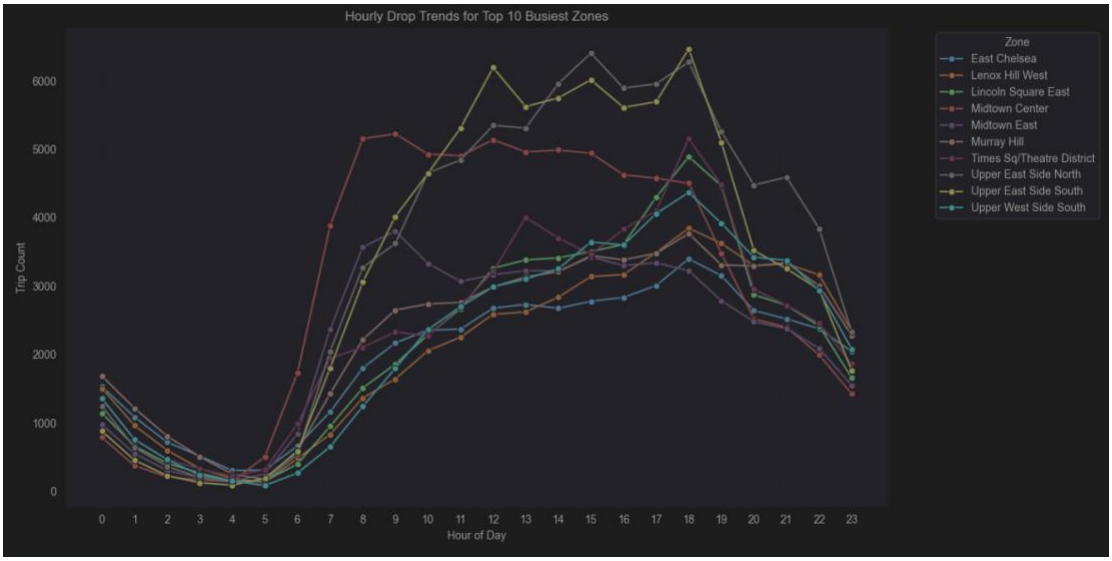
Observations:

- For Upper East Side North / South**,** dropoffs show**:**
  - **Two clear peaks**: one around **11 AM**, another around **6 PM (18:00 hrs)**
  - Patterns of **Morning arrivals** (commuters, appointments, errands), **Evening returns or visits**

- For Midtown Center dropoffs**:**
  - **Early peak at 7 AM**, due to morning commutes
  - **High volume sustained** through the day, indicating consistent daytime activity (work, tourism, etc.)
  - **Steep decline between 6–8 PM**, suggesting end of business hours or shift in activity zones

- For Times Square dropoffs:
  - Peaks occur at **12 PM (noon)** and **6 PM (18:00 hrs)**
  - Likely reflecting **lunch-time visitors and evening crowds** for entertainment, dining, and tourism.

- The **top 10 dropoff zones** do **not include airports**, unlike pickups where airports rank high.This implies that while airports are major origins for trips, passengers typically **get dropped off in other**



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

Top 10 pickup/dropoff ratio - Bottom 10 pickup/dropoff ratio

	zone	pickup_dropoff_ratio
70	East Elmhurst	8.210166
128	JFK Airport	4.372072
134	LaGuardia Airport	2.635315
182	Penn Station/Madison Sq West	1.558477
110	Greenwich Village South	1.377347
42	Central Park	1.369026
245	West Village	1.332877
158	Midtown East	1.242697
157	Midtown Center	1.188330
100	Garment District	1.187529

	zone	pickup_dropoff_ratio
0	Newark Airport	0.013390
241	West Brighton	0.032258
106	Great Kills	0.038462
111	Grymes Hill/Clifton	0.040000
26	Breezy Point/Fort Tilden/Riis Beach	0.051282
253	Windsor Terrace	0.051572
247	Westerleigh	0.058824
124	Inwood Hill Park	0.059829
64	Douglaston	0.063415
248	Whitestone	0.068493

- **East Elmhurst (8.21)**: Extremely high pickup-to-dropoff ratio, meaning many more pickups than dropoffs. This could indicate it's primarily a **residential or origin-heavy area**.
- JFK Airport (4.37) and LaGuardia Airport (2.64): Airports have notably higher pickups compared to dropoffs, consistent with their role as trip origins.
- The lowest (0.0684, 0.063) are Whitestone & Douglaston less populated area.

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

Night time pickup - Night time drop off

	PULocationID	pickup_count	LocationID	zone
71	79	16437	79.0	East Village
115	132	14511	132.0	JFK Airport
223	249	13119	249.0	West Village
42	48	10804	48.0	Clinton East
131	148	10167	148.0	Lower East Side
99	114	9174	114.0	Greenwich Village South
205	230	8464	230.0	Times Sq/Theatre District
165	186	7112	186.0	Penn Station/Madison Sq West
146	164	6439	164.0	Midtown South
60	68	6368	68.0	East Chelsea

	DOLocationID	dropoff_count	LocationID	zone
78	79	8685	79.0	East Village
46	48	7190	48.0	Clinton East
165	170	6476	170.0	Murray Hill
67	68	6099	68.0	East Chelsea
103	107	5929	107.0	Gramercy
136	141	5527	141.0	Lenox Hill West
256	263	5259	263.0	Yorkville West
242	249	5103	249.0	West Village
223	230	4814	230.0	Times Sq/Theatre District
143	148	4603	148.0	Lower East Side

Nighttime Pickup Counts in Key Zones:

- East Village: 16,437 pickups — highest among night pickups, indicating vibrant nightlife or residential demand.
- JFK Airport : 14,511 pickups — still high, reflecting late-night travel demand at the airport.
- West Village: 13,119 pickups — active nightlife or late return

Nighttime Dropoff Counts in Key Zones:

- East Village: 8,685 dropoffs — fewer than pickups, indicating more people start trips here than end them.
- Clinton East: 7,190 dropoffs
- Murray Hill: 6,476 dropoffs — notable dropoff location, likely a residential or business area.

3.2.8. Find the revenue share for nighttime and daytime hours

- Daytime revenue share: ~89.08% of total revenue occurs during the day.
- Nighttime revenue share: ~10.92% of total revenue occurs at night.

```
# Filter for night hours (11 PM to 5 AM)
revenue_share_df = cleaned_and_sampled_by_month_date_hour_df.copy()

night_revenue_df = revenue_share_df[(revenue_share_df['hour'] >= 23) | (revenue_share_df['hour'] <= 5)]
night_revenue = night_revenue_df['total_amount'].sum()

day_revenue_df = revenue_share_df[(revenue_share_df['hour'] < 23) | (revenue_share_df['hour'] > 5)]
day_revenue = day_revenue_df['total_amount'].sum()

print(f" Day's revenue share: {(day_revenue / (day_revenue + night_revenue))*100} %")
print(f" Night's revenue share: {(night_revenue / (day_revenue + night_revenue))*100} %")

del revenue_share_df
del day_revenue_df
del day_revenue
del night_revenue_df
del night_revenue
✓ [250] 986ms
```

```
Day's revenue share: 89.08305565995119 %
Night's revenue share: 10.91694434004882 %
```

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

- Fare per mile per passenger is high ($10.86) when the passenger count is 1 and reduces as the passenger count increases with 6 passenger ($1.35)

6 rows × 2 cols

	passenger_count	fare_per_mile_per_passenger
0	1.0	10.862568
1	2.0	6.431558
2	3.0	3.907755
3	4.0	4.347749
4	5.0	1.709614
5	6.0	1.350658

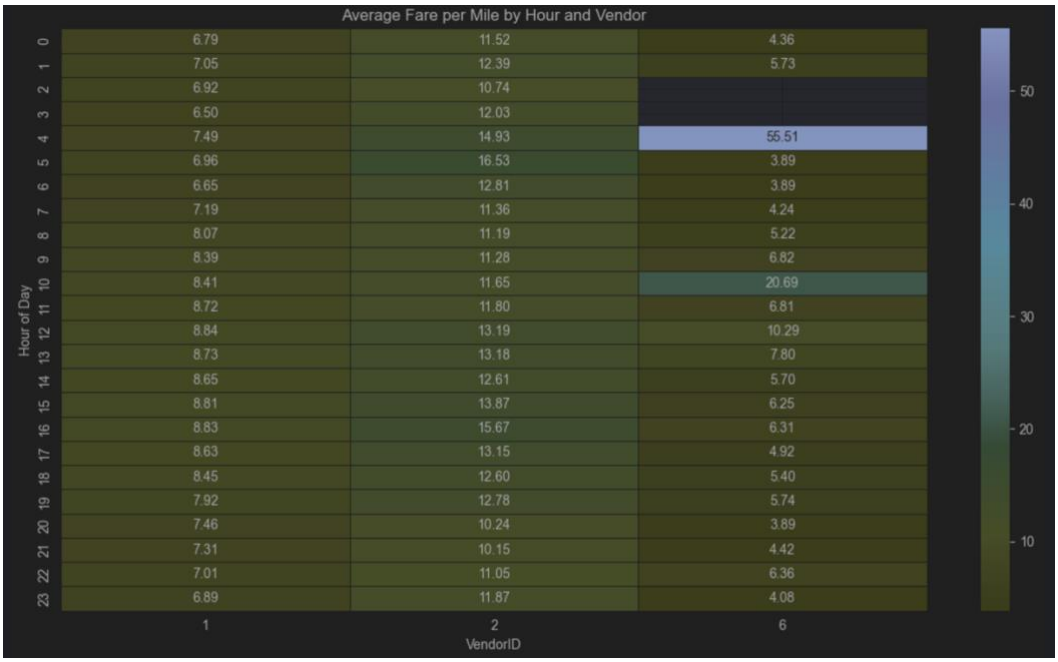
3.2.10. Find the average fare per mile by hours of the day and by days of the week

- The highest fare per mile is consistently observed between 10:00 AM to 6:00 PM, peaking around:
  - Wednesday 12:00 AM **–** $9.03/mile
  - Thursday 1:00 PM **–** $8.89/mile
  - Tuesday 11:00 AM **–** $9.03/mile
- This suggests mid-day weekday rides may be **shorter but higher-priced**, likely driven by short-hop business travel or traffic congestion.

Hour of Day	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
0	5.98	6.42	6.38	6.40	6.61	6.92	7.06
1	6.17	6.28	6.29	6.26	6.91	7.06	6.95
2	5.95	6.07	6.75	6.35	6.57	6.82	6.97
3	6.05	7.84	6.55	6.26	6.34	6.71	6.91
4	5.83	6.89	6.18	5.68	5.85	6.21	6.30
5	5.94	5.82	5.98	5.86	5.93	5.62	5.48
6	6.35	6.46	6.39	6.47	6.44	6.49	6.03
7	7.07	7.27	7.37	7.22	6.97	6.38	5.91
8	7.76	8.03	7.99	8.01	7.68	6.58	6.28
9	8.05	8.36	8.41	8.25	8.04	7.00	6.57
10	8.04	8.64	8.57	8.57	8.08	7.26	6.90
11	8.08	9.03	8.94	8.89	8.47	7.61	7.33
12	8.06	8.91	9.03	8.89	8.31	7.80	7.42
13	7.96	8.71	8.88	8.79	8.23	8.08	7.52
14	7.93	8.58	8.74	8.72	8.15	7.92	7.58
15	7.94	8.66	8.80	8.75	8.35	8.01	7.69
16	7.82	8.50	8.62	8.86	8.25	8.05	7.59
17	7.88	8.63	8.66	8.64	8.12	8.29	7.57
18	7.72	8.16	8.41	8.39	8.04	8.27	7.48
19	7.27	7.65	7.88	7.87	8.07	8.41	7.35
20	6.80	7.05	7.26	7.36	7.60	8.00	6.94
21	6.69	6.89	6.96	7.09	7.36	7.56	6.66
22	6.43	6.66	6.81	6.94	7.33	7.63	6.63
23	6.07	6.47	6.48	6.67	7.13	7.28	6.32

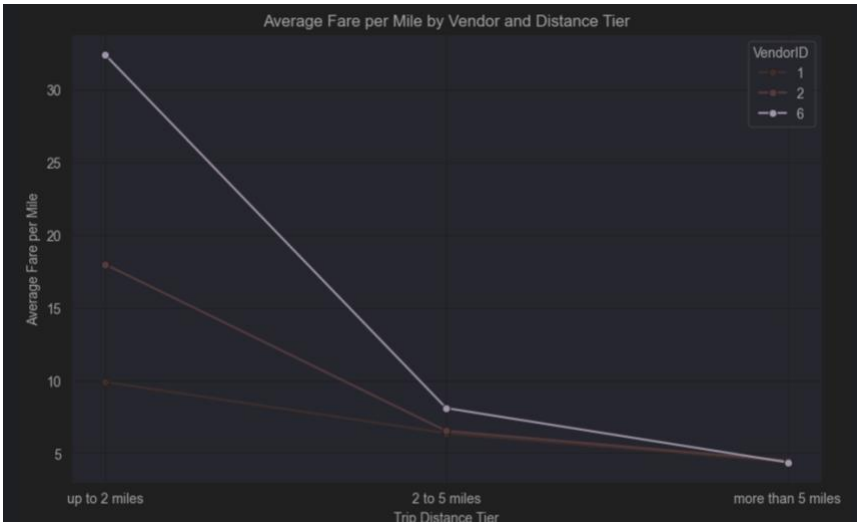
3.2.11. Analyse the average fare per mile for the different vendors

- The highest fare per mile for vendor is for Vendor 6 with peak on early morning 4, 10, 12 AM
- On the other hand other vendors have constant fare per mile while Vendor 6 has varying fare per mile across hours going low around 1, 5, 6, 7 AM, 8 PM etc.



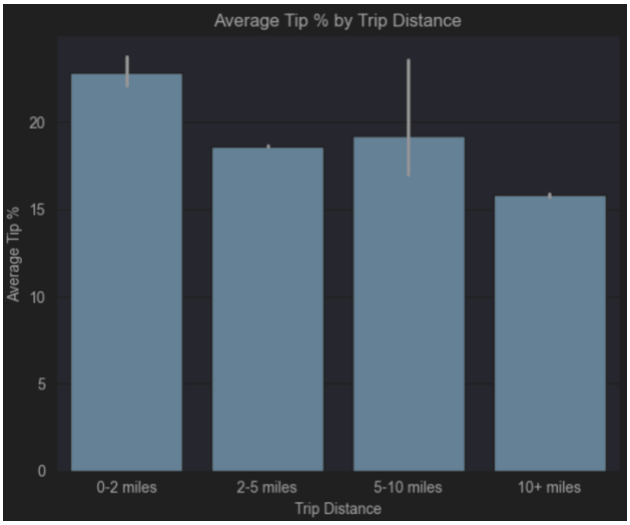
3.2.12. Compare the fare rates of different vendors in a distance-tiered fashion

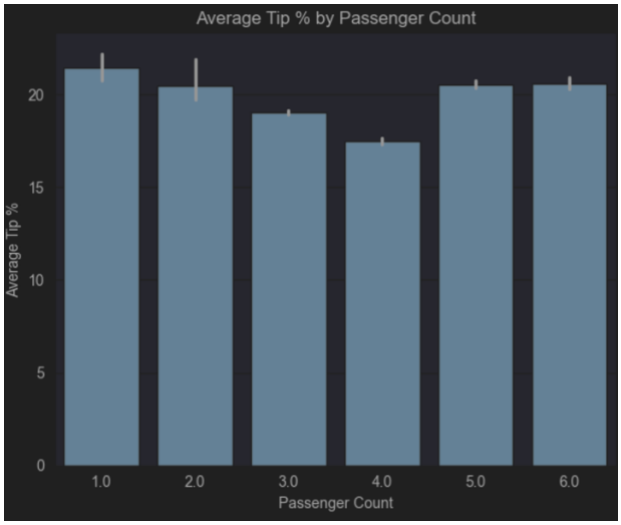
- Vendor 6 have high fare per mile for trip distance lesser than 2 miles, and almost same for 2 and above.

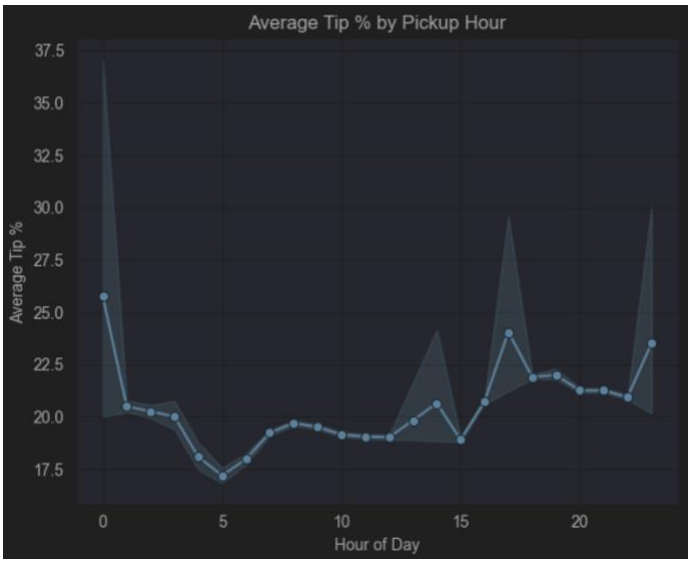


3.2.13. Analyse the tip percentages

- Average tip is higher for cases when the trip distance is between 0-2 miles, when it is 1 passenger, and during midnight.
- Average tip is lower for cases when the trip distance is greater than 10+miles, when it is 4 passengers, early morning around 5 AM.



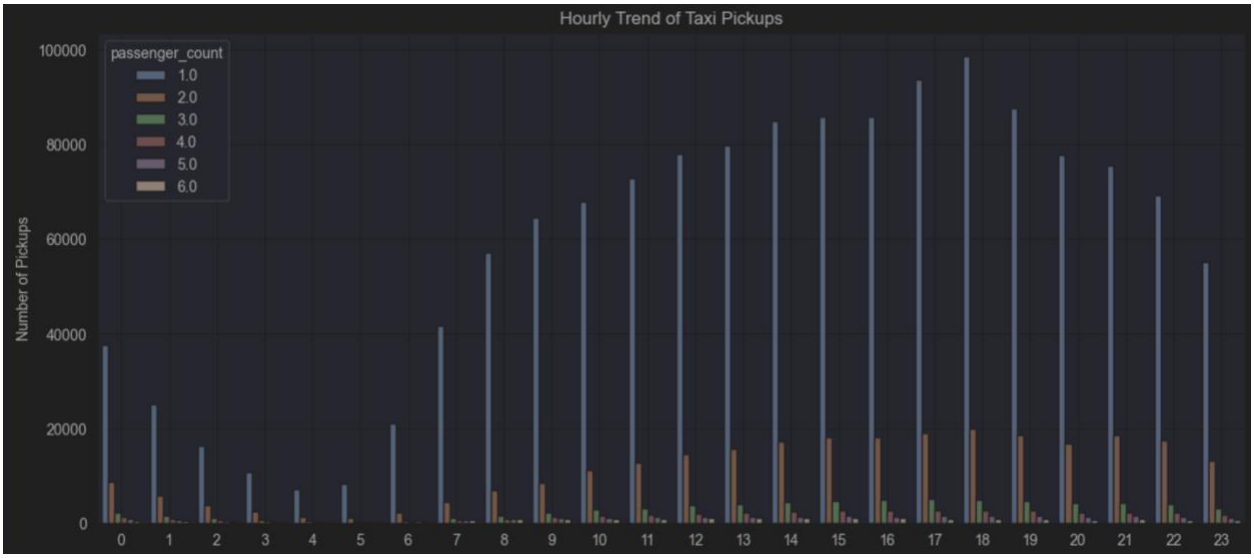




3.2.14. Analyse the trends in passenger count

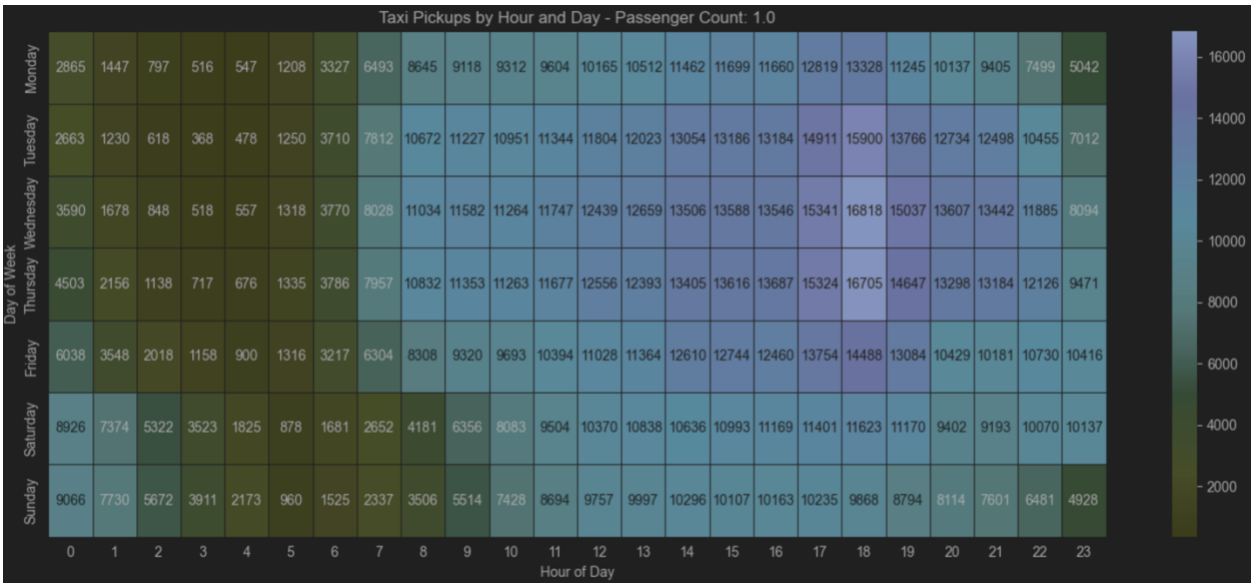
Passenger count across hour of the day:

- Single passenger count is always high across hours compared to other passenger count
- Two passenger count gradually increases during the day and peaks at 18 and remains saturated.

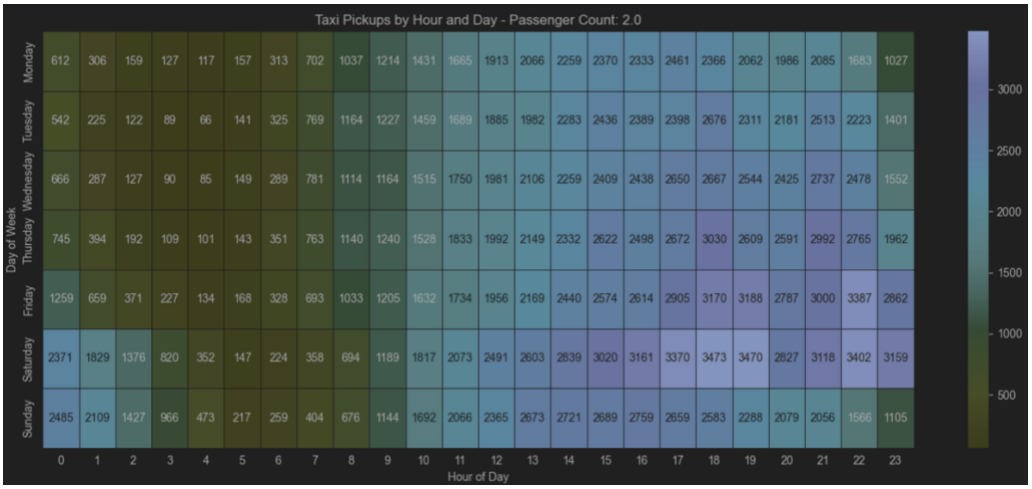


Passenger count across hour of the day and days of the week:

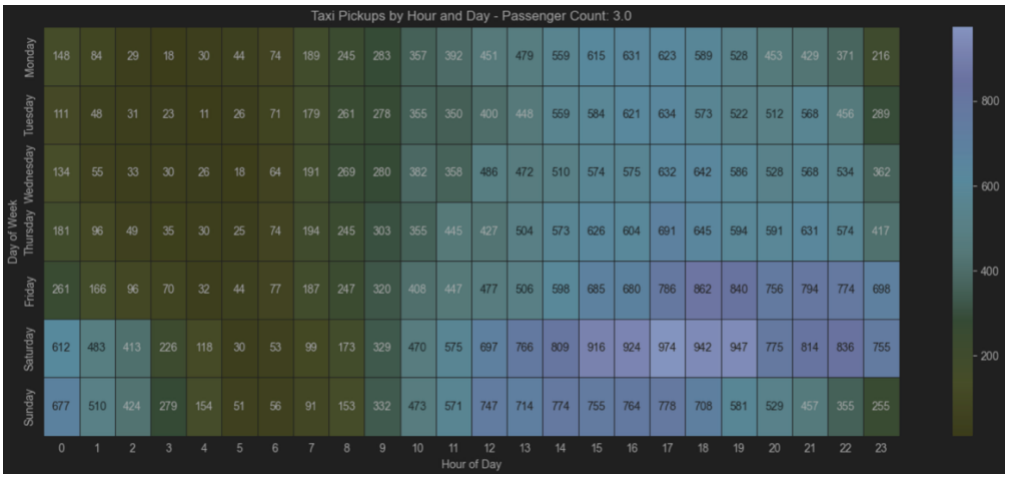
Single Passenger: It always higher in weekdays from 8 AM till midnight. This signifies these are office commuters.



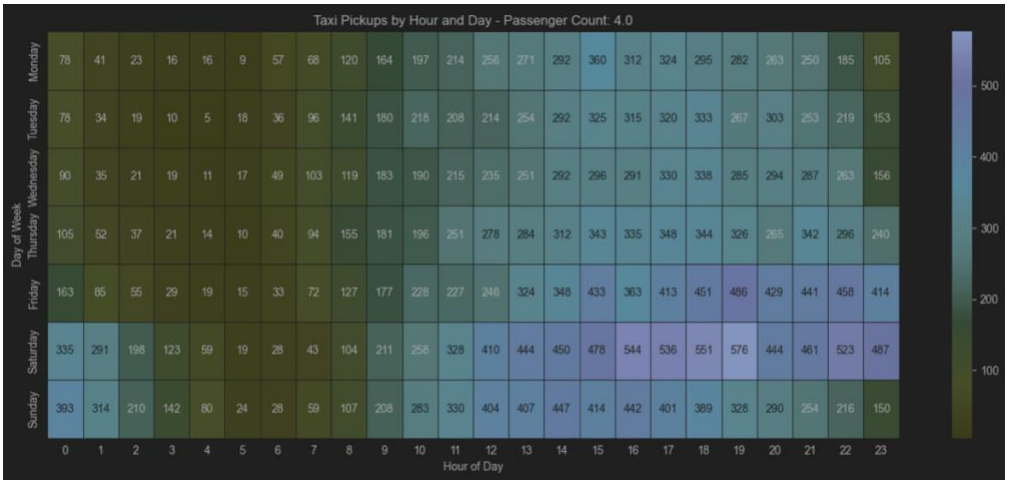
Two passenger: It starts to increase from midday till 8-9 PM in the night. However in Friday and Saturday it extends till midnight, maybe because of party times.



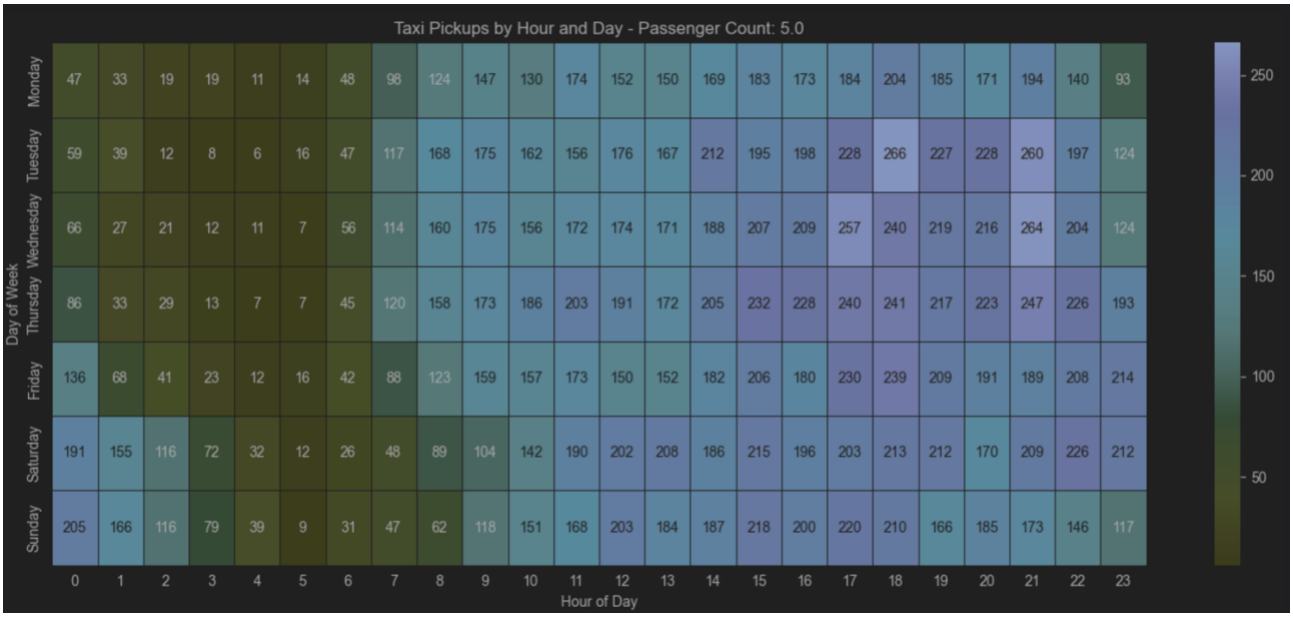
Three Passenger: This is heavily seen on Friday, Saturday and Sunday miday suggesting travelers, tourists



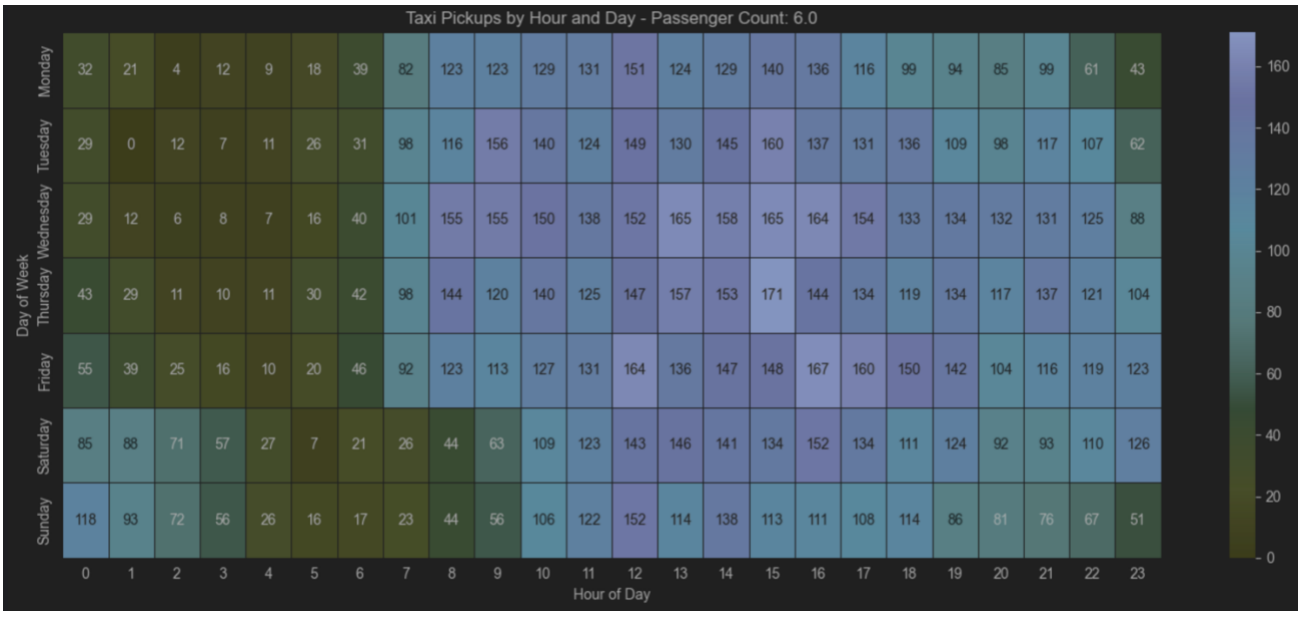
Four Passenger: This is same as three passengers travellers, tourists.



Five Passenger: it increases from 13 hour of the day in weekdays and goes till 10. Suggesting social errands completion, office completion people travel together for getting dropped.



Six Passenger: It is peak on Wednesday, Thursday Friday throughout the day.

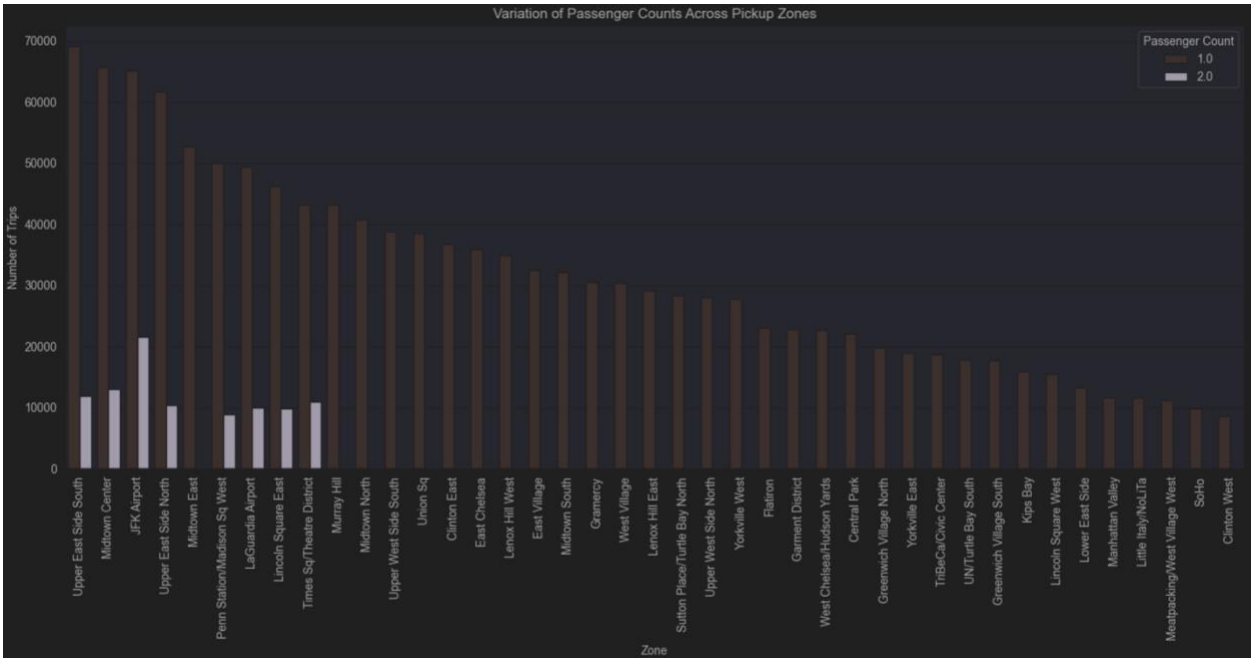


3.2.15. Analyse the variation of passenger counts across zones

- Single-passenger trips dominate across zones, especially the **busiest ones** like Upper East Side South, Midtown Center, JFK Airport, Upper East Side North, Midtown East. show significantly higher trip volumes for **passenger count = 1**, indicating that solo travelers form the bulk of taxi demand.

- A few zones show have **2-passenger trips**, especially JFK Airport**,** LaGuardia Airport**,** Times Sq/Theatre District**,** Midtown Center suggesting travelers are more likely to share rides or travel in pairs (e.g., tourists, airport transfers)

- Low-passenger-count zones Zones such as *SoHo*, *Clinton West*, and *Little Italy/NoLita* have lower total trip counts and minimal 2-passenger trips—likely due to narrower streets, walkability, or alternative transit usage



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

Frequency of Surcharge applied for pickup location:

- LaGuardia Airport has the highest rate of extra charges, with 98.7% of tipped trips including an extra charge.
- **Central Manhattan** areas (e.g., Midtown Center**,** Times Square**,** Union Sq) show **63%–65% frequency** of extra charges, indicative of high traffic volumes during surcharge hours.
- **Alphabet City** and **Hudson Sq** also reflect elevated surcharge application, despite having fewer trips

20 rows × 4 cols

	zone	extra_charge_frequency	extra_charge_tip	total_tip_counts
115	LaGuardia Airport	0.987247	63401	64220
60	East Elmhurst	0.832480	6992	8399
124	Lower East Side	0.784731	15398	19622
93	Greenwich Village South	0.766493	19229	25087
69	East Village	0.733644	33193	45244
218	West Village	0.730590	30611	41899
120	Little Italy/NoLiTa	0.697588	11972	17162
134	Meatpacking/West Village West	0.676611	11227	16593
183	SoHo	0.672843	9514	14140
221	Willets Point	0.666667	2	3
37	Clinton East	0.658137	33022	50175
200	Times Sq/Theatre District	0.654506	40885	62467
204	Union Sq	0.644395	32694	50736
1	Alphabet City	0.644169	1508	2341
140	Midtown South	0.641832	28231	43985
215	West Chelsea/Hudson Yards	0.638368	20295	31792
201	TriBeCa/Civic Center	0.636317	16242	25525
137	Midtown Center	0.634377	55475	87448
102	Hudson Sq	0.628498	5862	9327
92	Greenwich Village North	0.626565	16369	26125

Frequency of Surcharge applied for dropoff location:

- LaGuardia Airport stands out with the highest surcharge frequency of 89.2%**.** Out of 24,369 tipped rides, 21,732 included a surcharge. This likely reflects standardized surcharges for airport drop-offs.
- North Corona (79.5%)**,** Carroll Gardens (79.0%)**,** Bay Terrace/Fort Totten (78.3%), and Cobble Hill (78.2%) also exhibit high surcharge frequencies.
- Many zones Park Slope, Clinton Hill, Williamsburg Astoria, Whitestone, East Williamsburg show above 74% surcharge frequencies

20 rows · 20 rows × 4 cols

	zone	extra_charge_frequency	extra_charge_tip	total_tip_counts
134	LaGuardia Airport	0.891789	21732	24369
169	North Corona	0.794667	298	375
39	Carroll Gardens	0.790470	1294	1637
14	Bay Terrace/Fort Totten	0.783251	159	203
51	Cobble Hill	0.782186	966	1235
247	Whitestone	0.778082	284	365
80	East Williamsburg	0.775763	1882	2426
240	West Brighton	0.774194	24	31
106	Great Kills	0.769231	20	26
218	Steinway	0.765745	1994	2604
108	Greenpoint	0.763882	3182	4166
177	Park Slope	0.761914	3805	4994
252	Windsor Terrace	0.761006	605	795
48	Clinton Hill	0.760517	2061	2710
251	Williamsburg (South Side)	0.757088	3151	4162
6	Astoria	0.754092	4100	5437
250	Williamsburg (North Side)	0.752449	3687	4900
185	Prospect Heights	0.750581	1291	1720
53	Columbia Street	0.743781	299	402
64	Douglaston	0.741463	152	205

Frequency of Surcharge applied for hour of the day:

- 9 PM (21:00) shows the highest surcharge frequency of 94.2%, followed closely by:
  - 10 PM – 93.6% 8 PM – 93.3% 2 AM – 93.2% 1 AM – 92.8%

	hour	extra_charge_tip	total_tip_counts	extra_charge_frequency
21	21	100058	106175	0.942388
22	22	92039	98378	0.935565
20	20	99131	106257	0.932936
2	2	21972	23583	0.931688
1	1	33240	35807	0.928310
23	23	72026	77839	0.925320
0	0	49176	53379	0.921261
3	3	14163	15532	0.911859
4	4	8500	10502	0.809370
18	18	104722	133559	0.784088
19	19	93086	119342	0.779994
17	17	98945	127475	0.776191
5	5	8517	11060	0.770072
16	16	87260	117053	0.745474
10	10	26881	87697	0.306521
6	6	8006	26231	0.305211
14	14	34534	114115	0.302625
13	13	32096	106520	0.301314
9	9	24328	80812	0.301044
11	11	28599	95196	0.300422

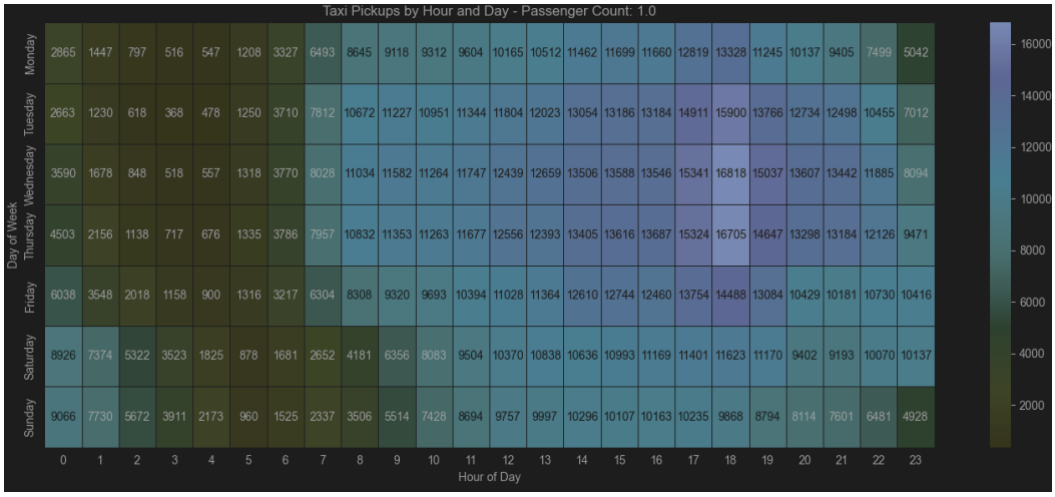
4. Conclusions

4.1. Final Insights and Recommendations

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

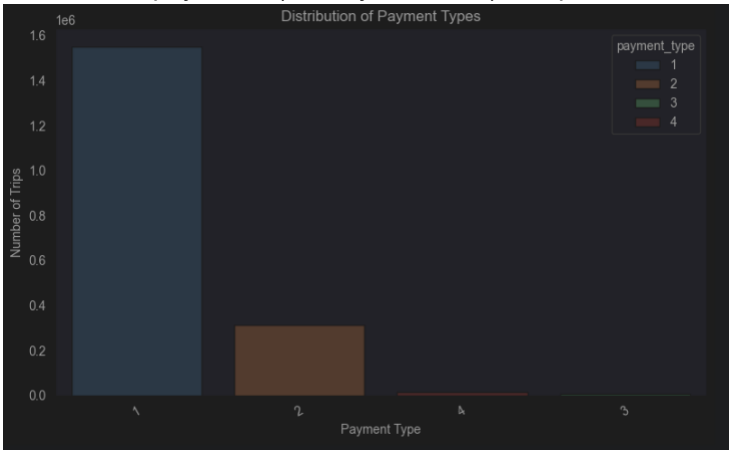
Based on Temporal conclusions mentioned here.

- **Deploy Hybrid or Electric Vehicles for Morning Shifts**
  - Most morning taxi trips have a passenger count of 1, especially during typical commuting hours (e.g., 7–12). Can assign **EVs or hybrids** to cover morning peak hours when trips are short and passenger load is low.
  - This reduces per-passenger emissions, especially for high-frequency, single-rider commutes.



- **Dynamic Routing:**
  - Use speed and trip duration data to identify and avoid congested routes during peak traffic hours.
  - Implement route optimization focusing on zones with historically slow speeds.

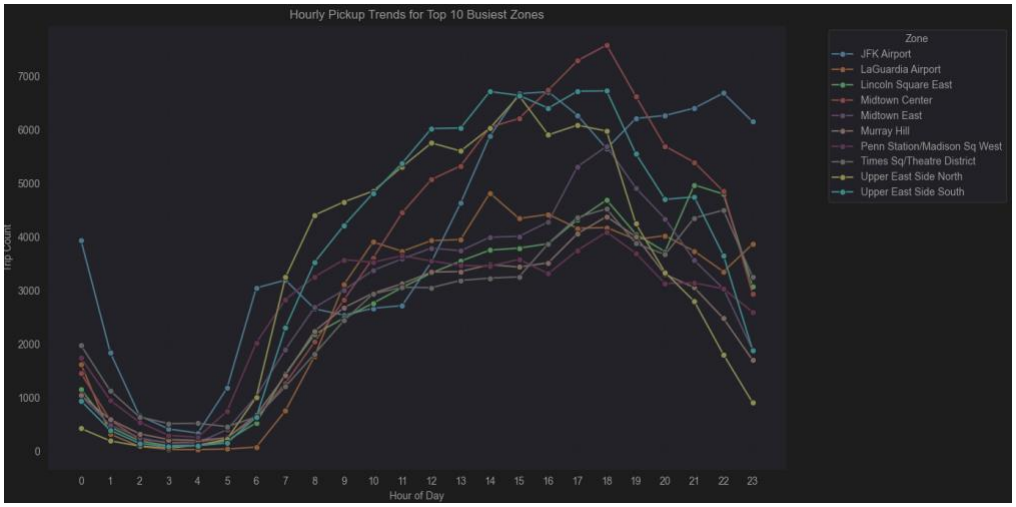
- **Payment and Tip Monitoring:**
  - Since tips are higher on shorter trips, incentivize drivers to optimize for these where feasible.
  - Promote cashless payments (already dominant) for operational ease and fraud reduction.



- **Nighttime Strategy:**
  - Given the 11% revenue share at night, optimize night shifts focusing on zones with highest night pickups/dropoffs (East Village, JFK, West Village).
  - Consider demand-responsive pricing or promotions to encourage balanced trip flows.

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.

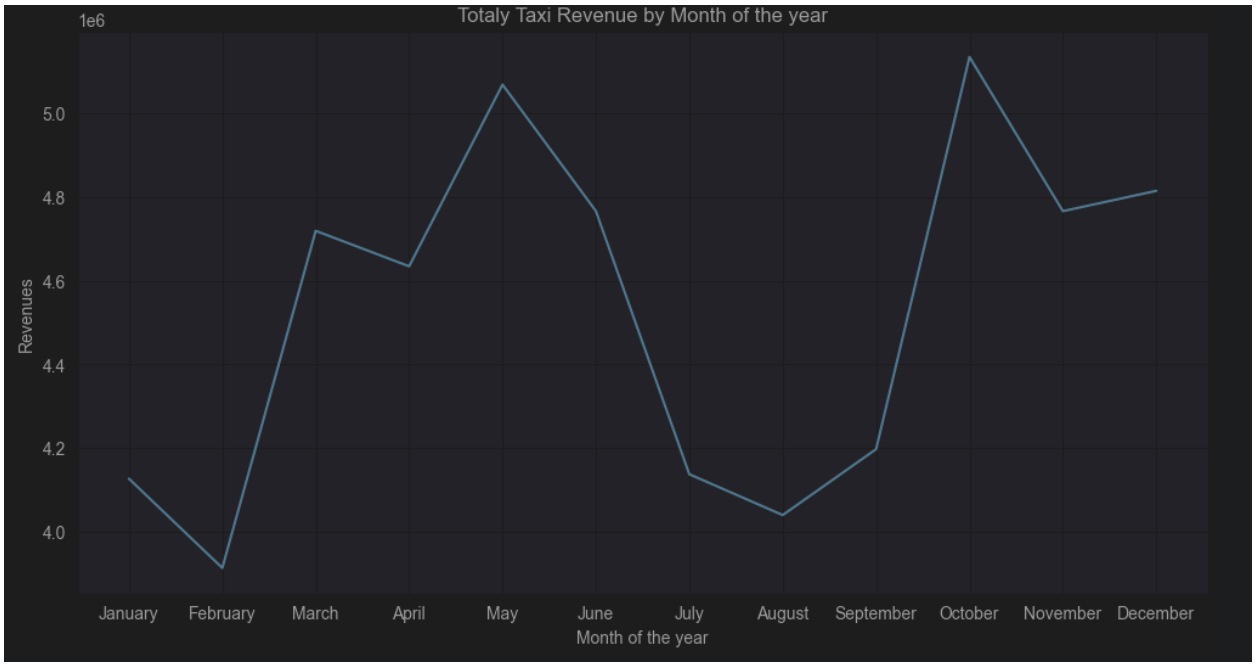
- Increase vehicle availability during peak hours (11:00–21:00), especially in high-demand zones (airports, Midtown, Upper east side North / South,).
- Vehicle availability is every important post 15 every day as that is the peak trips are taken
- Reduce idle resources during low demand (early morning hours 01:00–06:00)



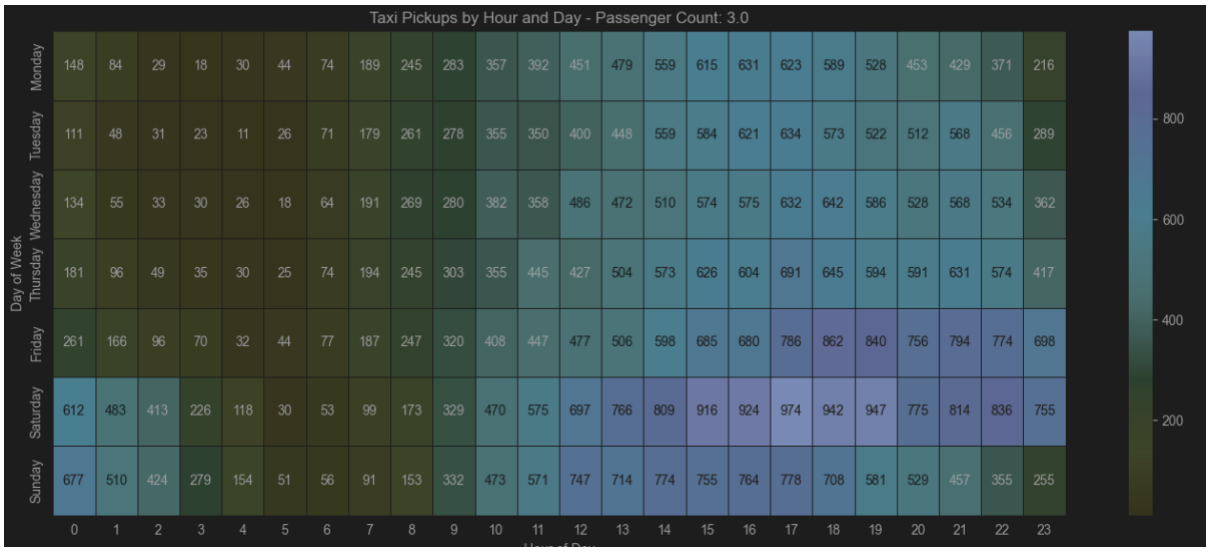
- While **East Elmhurst** has highest pick ratio, need to keep some cabs in reserve in those areas.

	zone	pickup_dropoff_ratio
70	East Elmhurst	8.210166
128	JFK Airport	4.372072
134	LaGuardia Airport	2.635315
182	Penn Station/Madison Sq West	1.558477
110	Greenwich Village South	1.377347
42	Central Park	1.369026
245	West Village	1.332877
158	Midtown East	1.242697
157	Midtown Center	1.188330
100	Garment District	1.187529

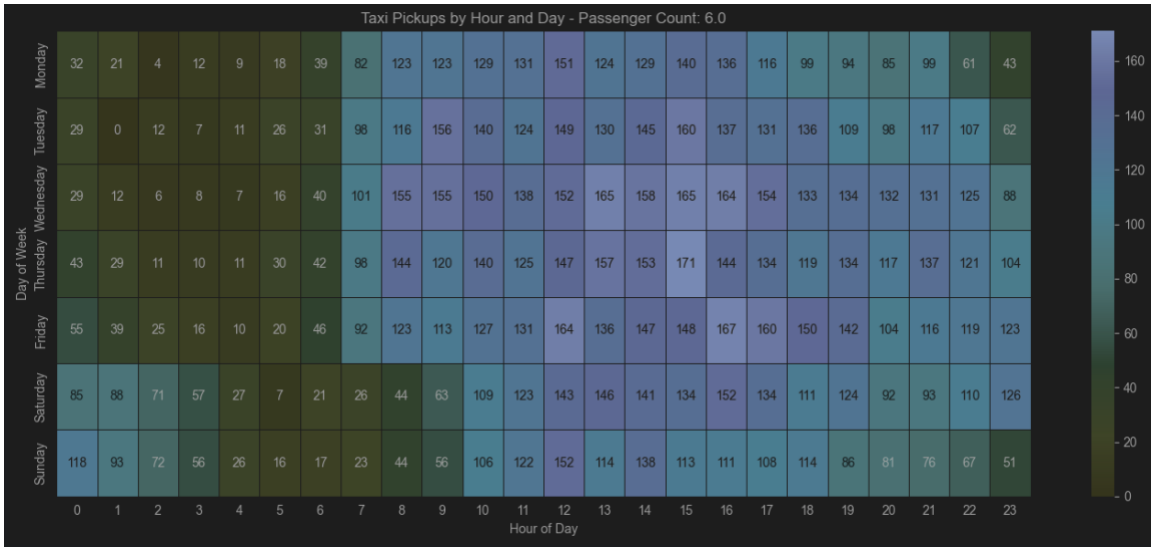
- Keep cabs proportionally high in Q2 and Q4, while in summer holidays we can limit the cabs.



- Since trips with more than 3 passengers are more common during firday evening, Saturday, Sunday we can deploy four seater car for the same

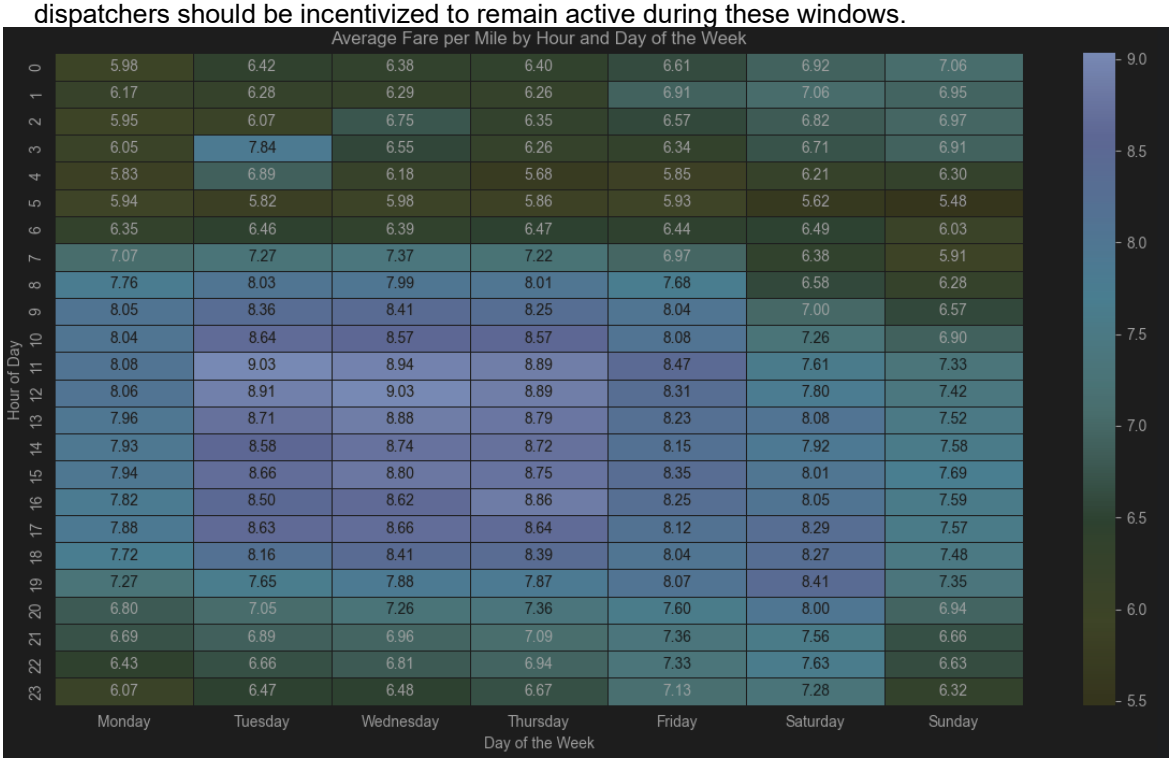


- Since trips with more than 6 passengers are more common during midday hours, deploying 6-seater vehicles between late morning and 6 PM can better accommodate tourist groups and improve service efficiency.



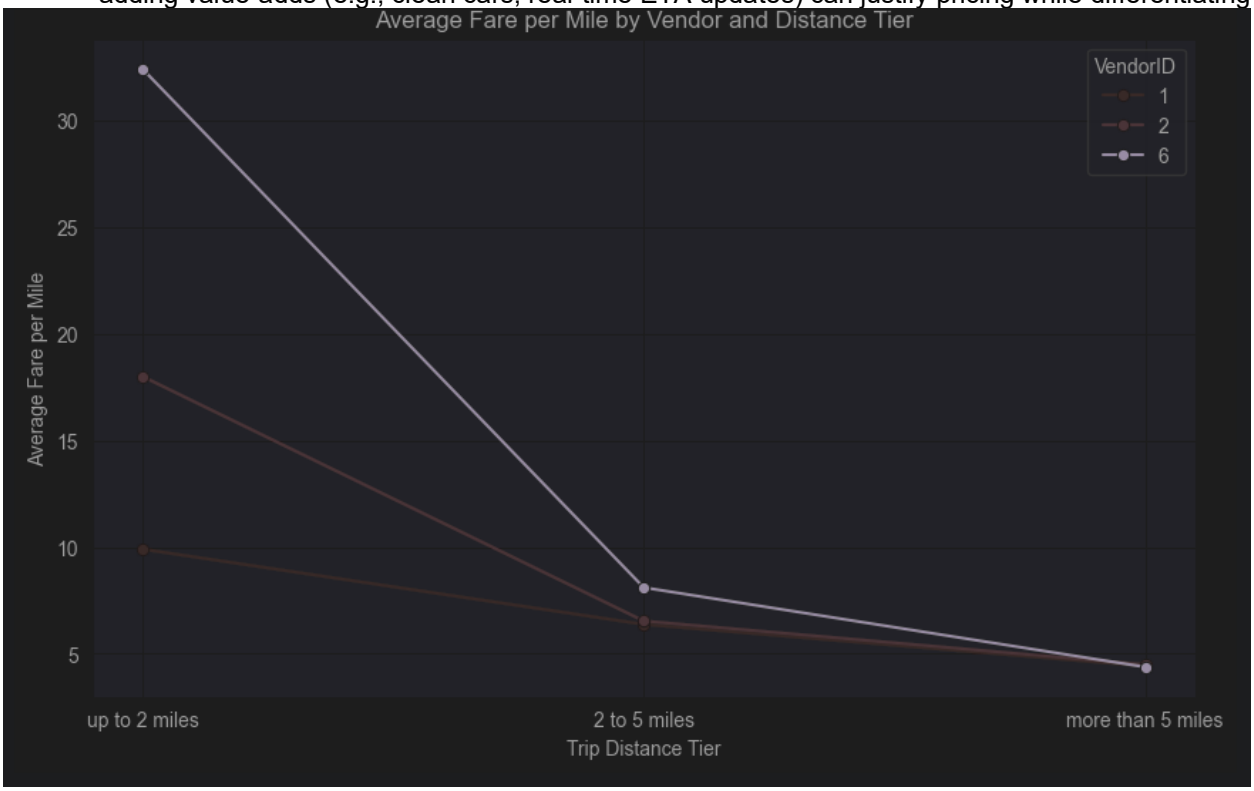
4.1.3. Propose data-driven adjustments to the pricing strategy to maximize revenue while maintaining competitive rates with other vendors.

- Midday trips yield higher fare per mile especially on Tuesday, Wednesday Thursday. Drivers and dispatchers should be incentivized to remain active during these windows.



Tiered Pricing Based on Distance

- Short trips (<2 miles): Fare per mile is very high, especially with Vendor 6. This could deter customers. Introduce discounts or flat-rate pricing to attract more riders for short urban commutes.
- Medium trips (2–5 miles): This is a competitive segment. Since most vendors charge similarly, adding value-adds (e.g., clean cars, real-time ETA updates) can justify pricing while differentiating



Time-Based Dynamic Pricing:

- Use dynamic pricing during peak hours (e.g., 15:00–21:00), especially in months with high demand (March, April, May, Oct–Dec).
- Consider night-time discounts (1:00–6:00) to improve utilization, since revenue share during night is only ~11%.

Passenger-Based Fare Adjustments

- Low passenger counts have high fare-per-mile-per-passenger.
- Encourage shared rides or implement multi-passenger fare reductions to improve efficiency.
    - Example: For 1–2 passengers, offer slightly lower per-mile rates to increase ridership and reduce empty miles.

6 rows 6 rows × 2 cols

	passenger_count	fare_per_mile_per_passenger
0	1.0	10.862568
1	2.0	6.431558
2	3.0	3.907755
3	4.0	4.347749
4	5.0	1.709614
5	6.0	1.350658