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

How many rows are there?
After loading 2023-1.parquet, there are 3041714 rows and 19 columns

Do you think handling such a large number of rows is computationally feasible when we have to combine the data for all twelve months into one?

Loading all month's data would be possible, if we have enough resources like memory and processing cpu. Else the program will crash and cause system slowdown. Hence in general, it is good to load them separately and sample fraction of the data.

1.1.1. Sample the data and combine the files

For us to have entries of around 250,000 to 300,000 a good sampling fraction is determined to be 0.008.

number of records after sampling= 303397

2. Data Cleaning

2.1. Fixing Columns

2.1.1. Fix the index

Dropped store_and_fwd_flag column

Index fixed with df.index_reset()

2.1.2. Combine the two airport_fee columns

Data is distributed between two different columns Airport_fee and airport_fee.

airport_fee had many null columns.

Created a new column Airport_Fee_comb:

df['Airport_Fee_Comb']=df['Airport_fee'].fillna(0)+df['airport_fee'].fillna(0)

At this point dropped the Airport_fee and airport_fee columns

2.1.3 Fix columns with negative (monetary) values

Columns with negative values:

12	mta_tax	16
15	improvement_surcharge	16
16	total_amount	16
17	congestion_surcharge	15

Fixed all of these negative values by using `abs` function.

Since in my sample data there were no negative values for Fare_amount,
I couldn't find anything different for RateCodeID.

2.2. Handling Missing Values

2.2.1. Find the proportion of missing values in each column

	ColumnName	Missing_values_count	Missing_values_Percentage
3	passenger_count	11081	3.7
5	RatecodeID	11081	3.7
6	store_and_fwd_flag	11081	3.7
17	congestion_surcharge	11081	3.7
18	airport_fee	279692	92.2
19	Airport_fee	34786	11.5

2.2.2. Handling missing values in passenger_count

In my sample 3.7% of the rows had missing passenger_count. Fixed this by updating passenger_count to have mean of passenger_count.

2.2.3. Handle missing values in RatecodeID

There were 11081 missing values for RatecodeID. To impute this value, found the max of count of RatecodeID and updated the same. In this case RatecodeId 1 was the most popular, so the missing values were updated with 1.

2.2.4. Impute NaN in congestion_surcharge

There wre 11081 NaN in congestion_surcharge. Imputed these values with mean.

2.2.5 Are there missing values in other columns? Did you find NaN values in some other set of columns? Handle those missing values below.

Airport_fee and airport_fee had Nan values which were filled with 0

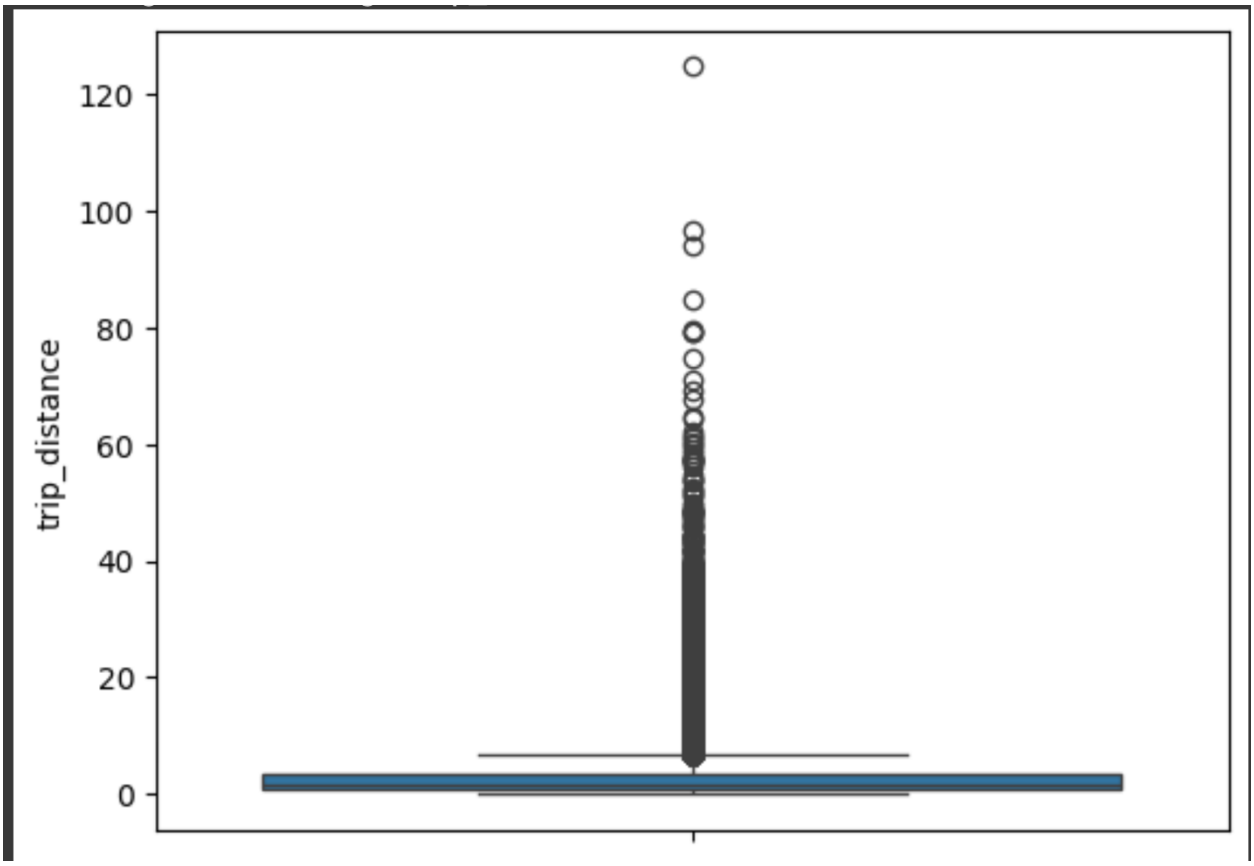
2.3. Handling Outliers and Standardising Values

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

RatecodeId had an outlier value of 99.0, and it is replaced with 1, the most popular value for RatecodeId.

For few records payment_type was 0 and they have been removed.

Trip_distance of > 200 is an outlier.



Inspected all the columns and found nothing needs standardization. Incorrect payment type was already done.

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: Numerical

tpep_dropoff_datetime: Numerical

passenger_count: Numerical

trip_distance: Numerical

RatecodeID: Categorical

PULocationID: Categorical

DOLocationID: Categorical

payment_type: Categorical

pickup_hour: Categorical

trip_duration: Numerical

fare_amount : Numerical

extra : Numerical

mta_tax : Numerical

tip_amount : Numerical

tolls_amount : Numerical

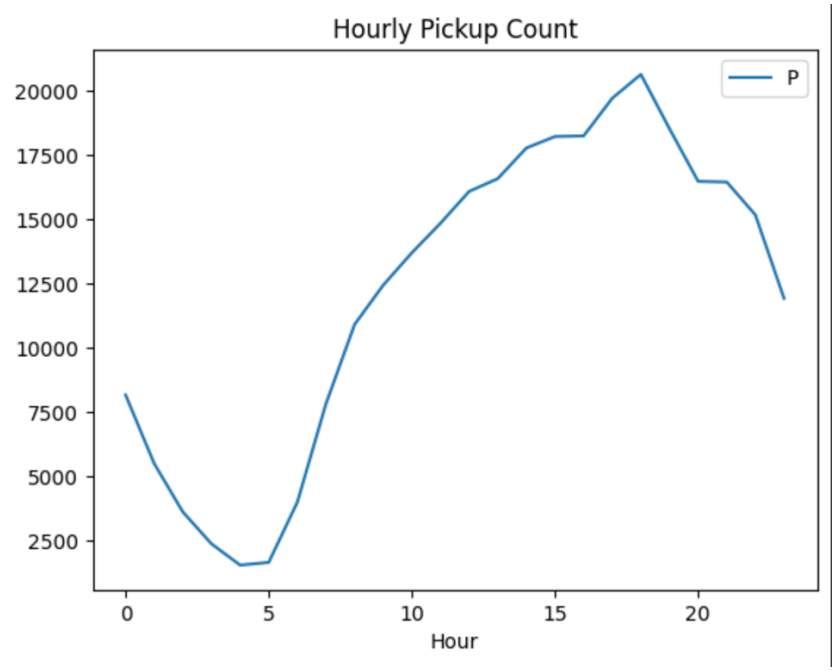
improvement_surcharge : Numerical

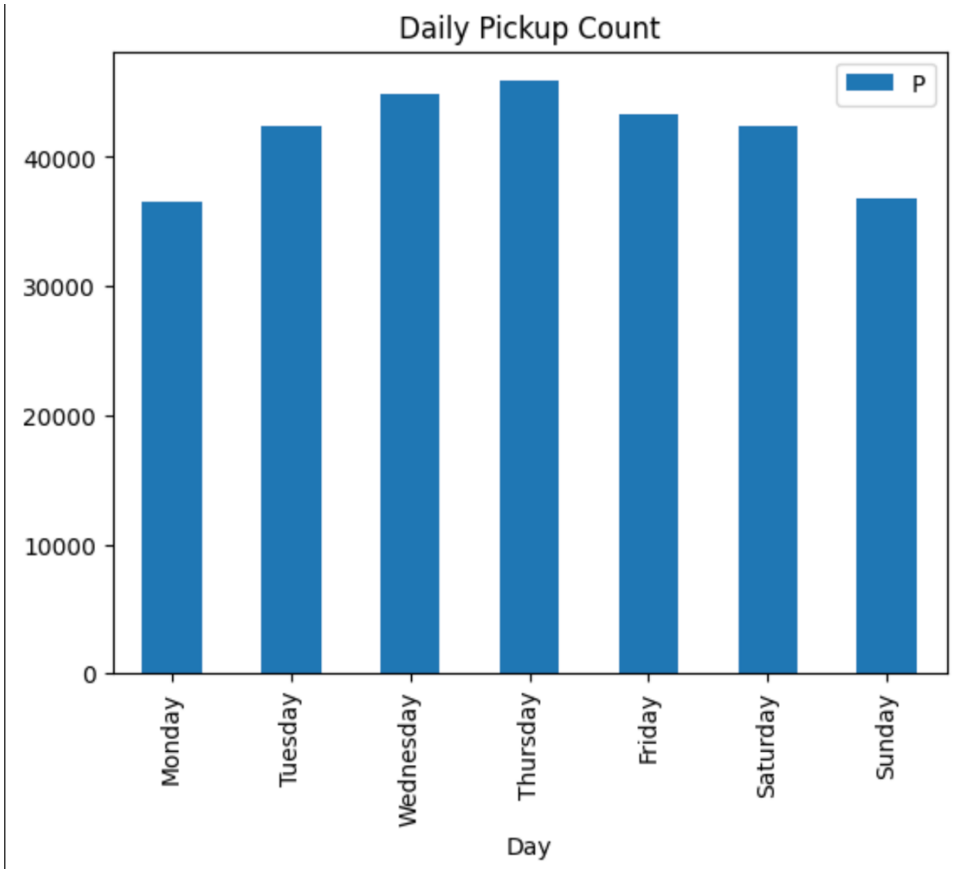
total_amount : Numerical

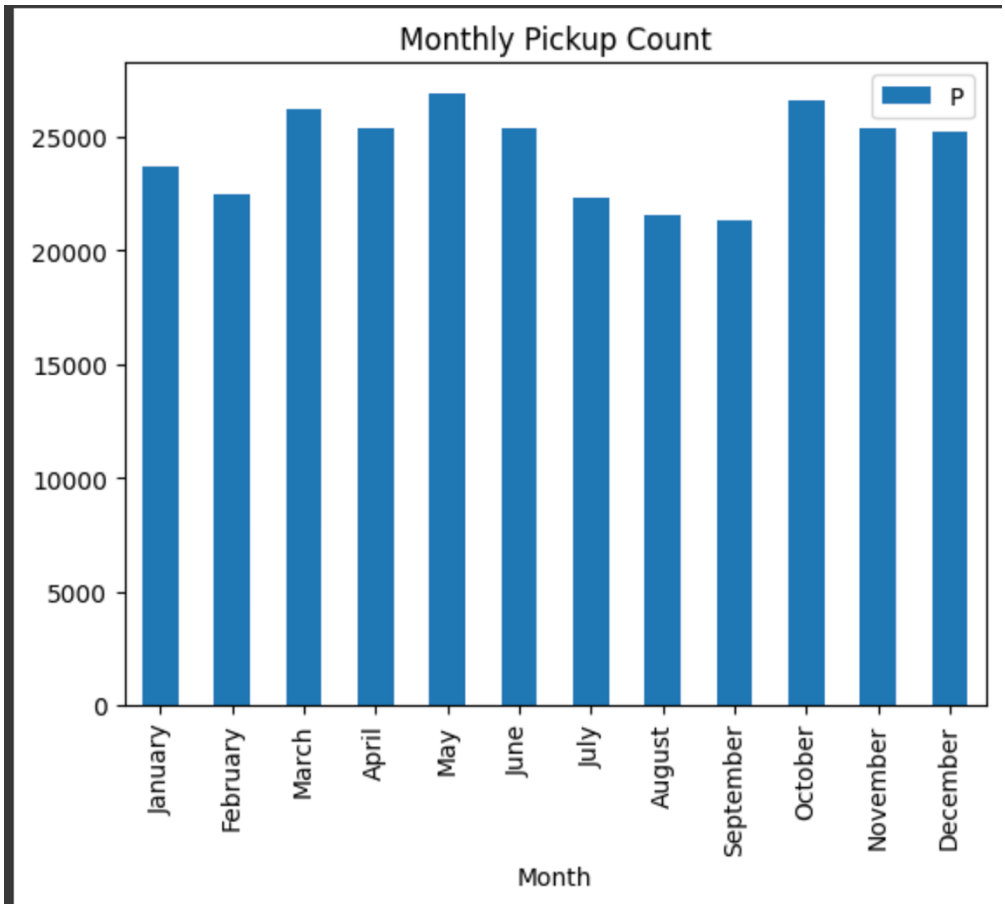
congestion_surcharge : Numerical

airport_fee : Numerical

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







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

fare_amount values and count where value<=0

fare_amount

0.0 105

Name: count, dtype: int64

tip_amount values and count where value<=0

tip_amount

0.0 65224

Name: count, dtype: int64

total_amount values and count where value<=0

total_amount

0.0 40

Name: count, dtype: int64

trip_distance values and count where value<=0

trip_distance

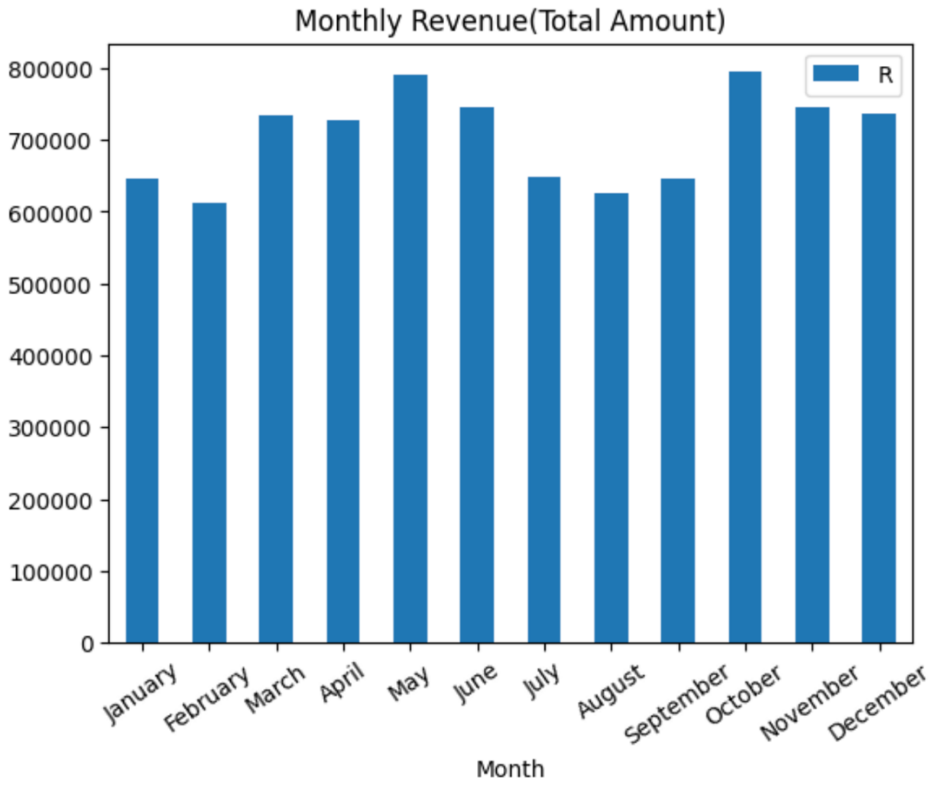
0.0 3642

Name: count, dtype: int64

The tip_count and trip_distance columns with 0 can’t be removed, since, they are genuine. Travel from one location to the same location is termed to be 0. Its possible that the drivers are not being tipped.

3.1.4. Analyse the monthly revenue trends

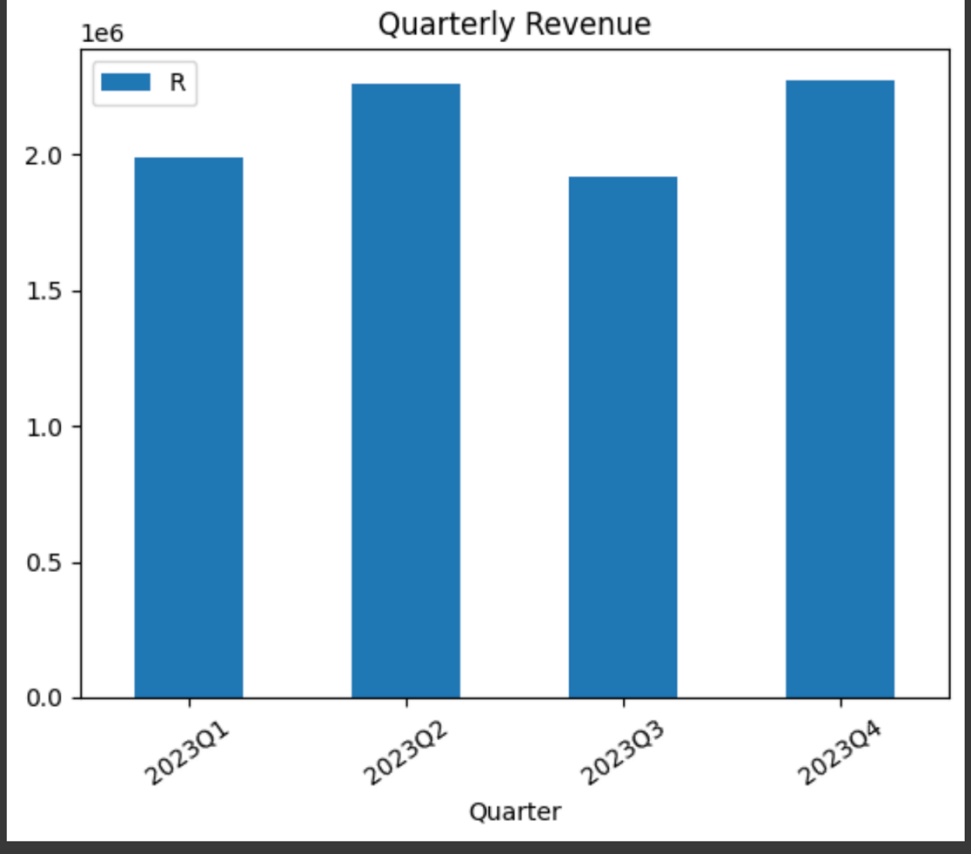
Revenue for the month based on starting time.



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

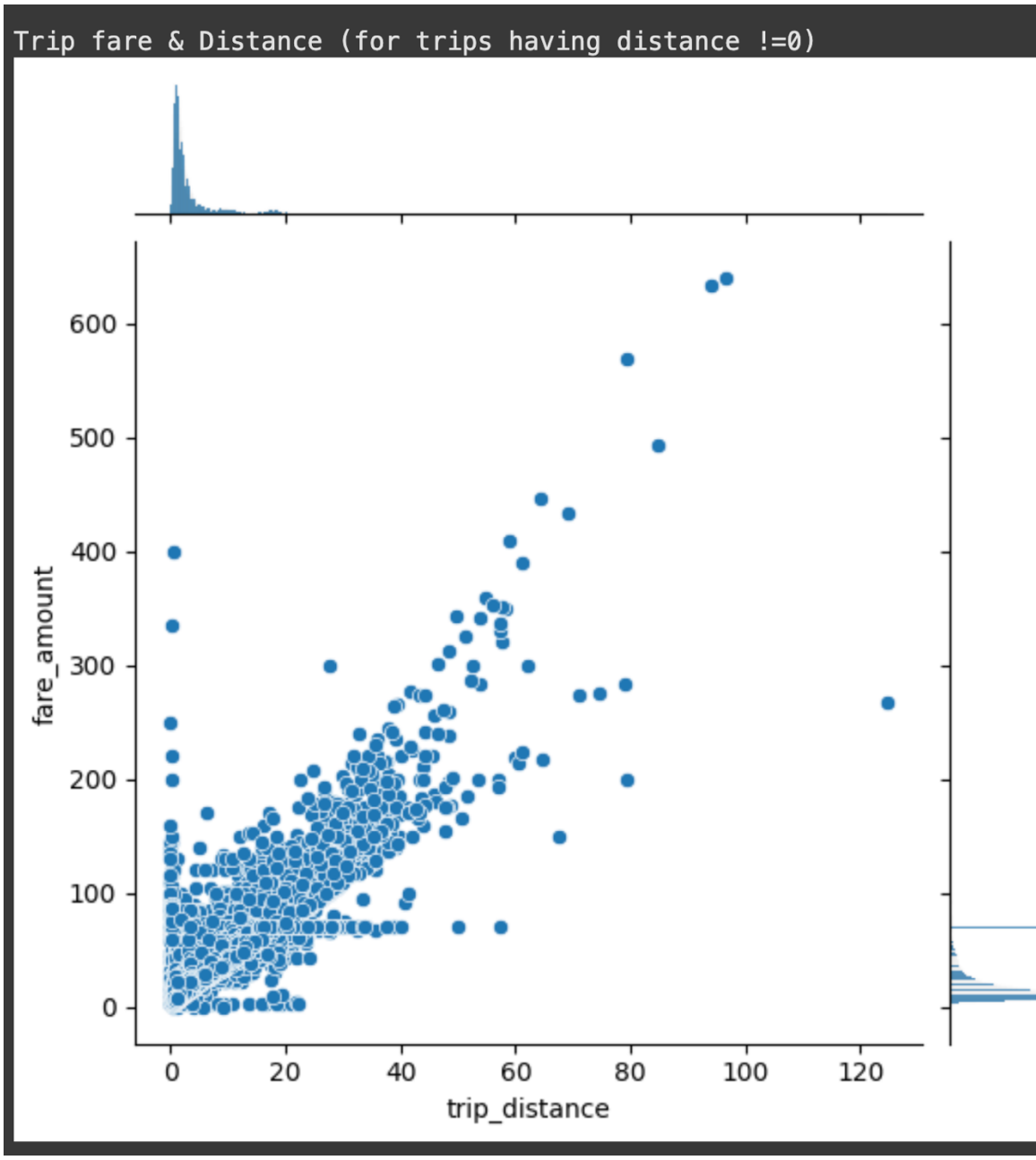
Revenue of the quarter based on starting time

	tpep_pickup_datetime	Total Quarterly Amount
0	2023Q1	1990935.42
1	2023Q2	2262966.51
2	2023Q3	1919205.40
3	2023Q4	2275593.71



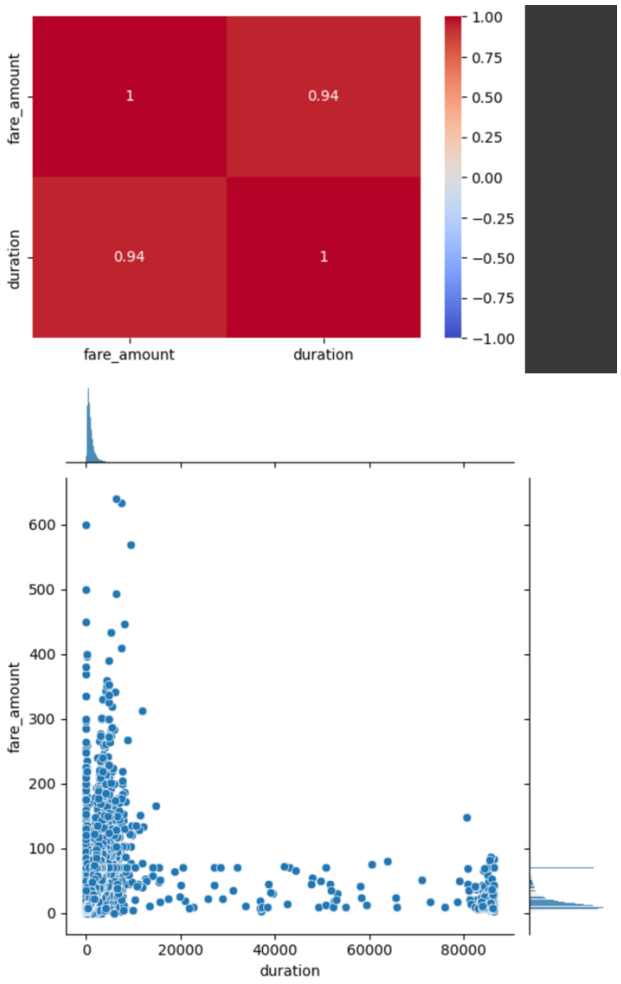
3.1.6. Analyse and visualise the relationship between distance and fare amount

As the distance increases, the fare increases.

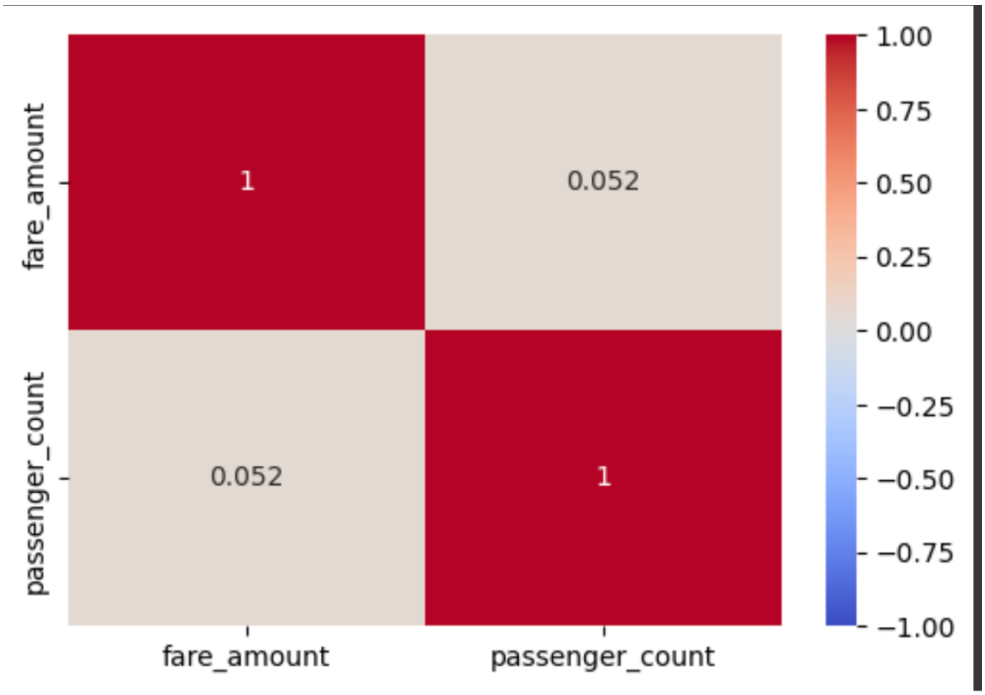


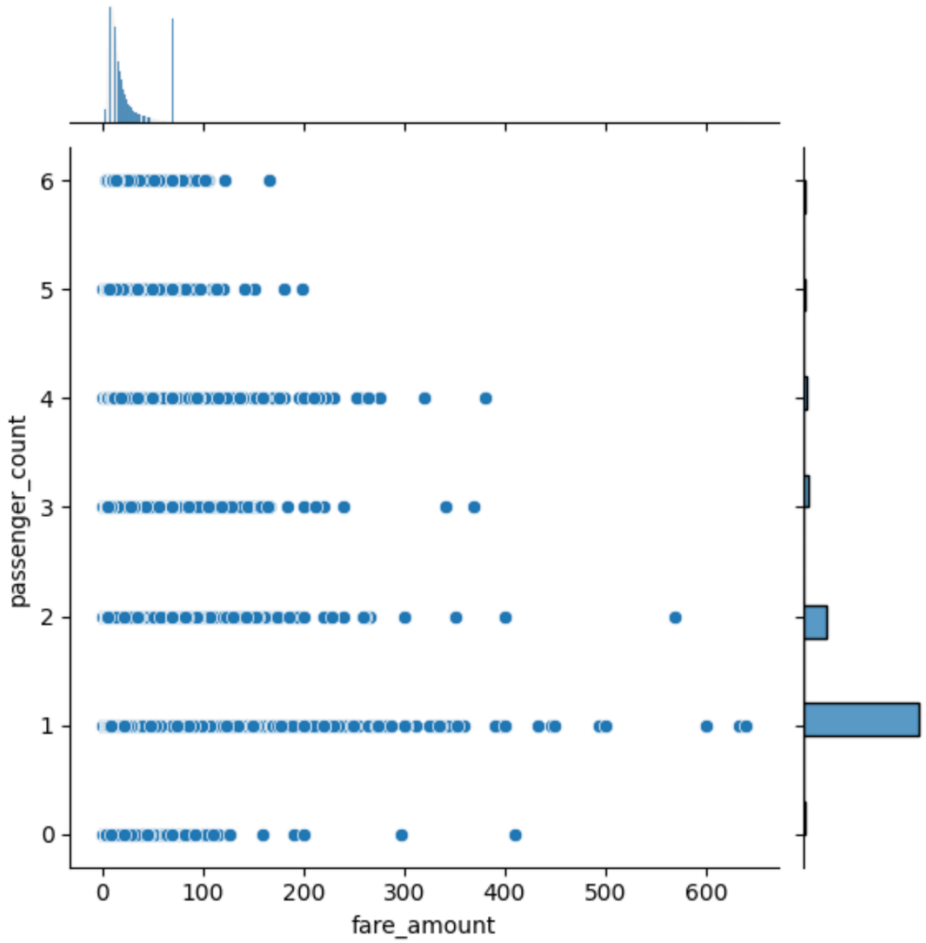
3.1.7. Analyse the relationship between fare/tips and trips/passengers

Duration and Fare are well related

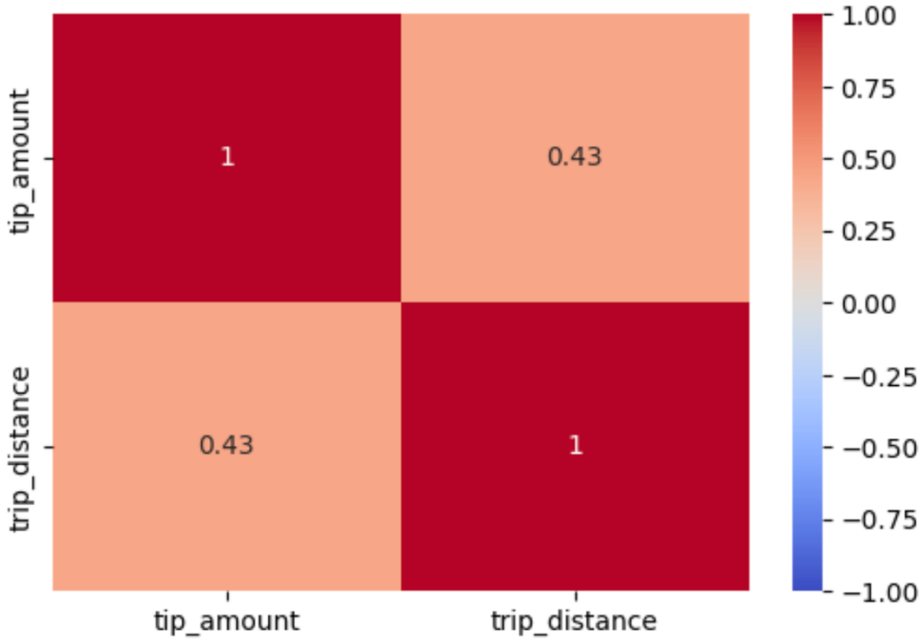


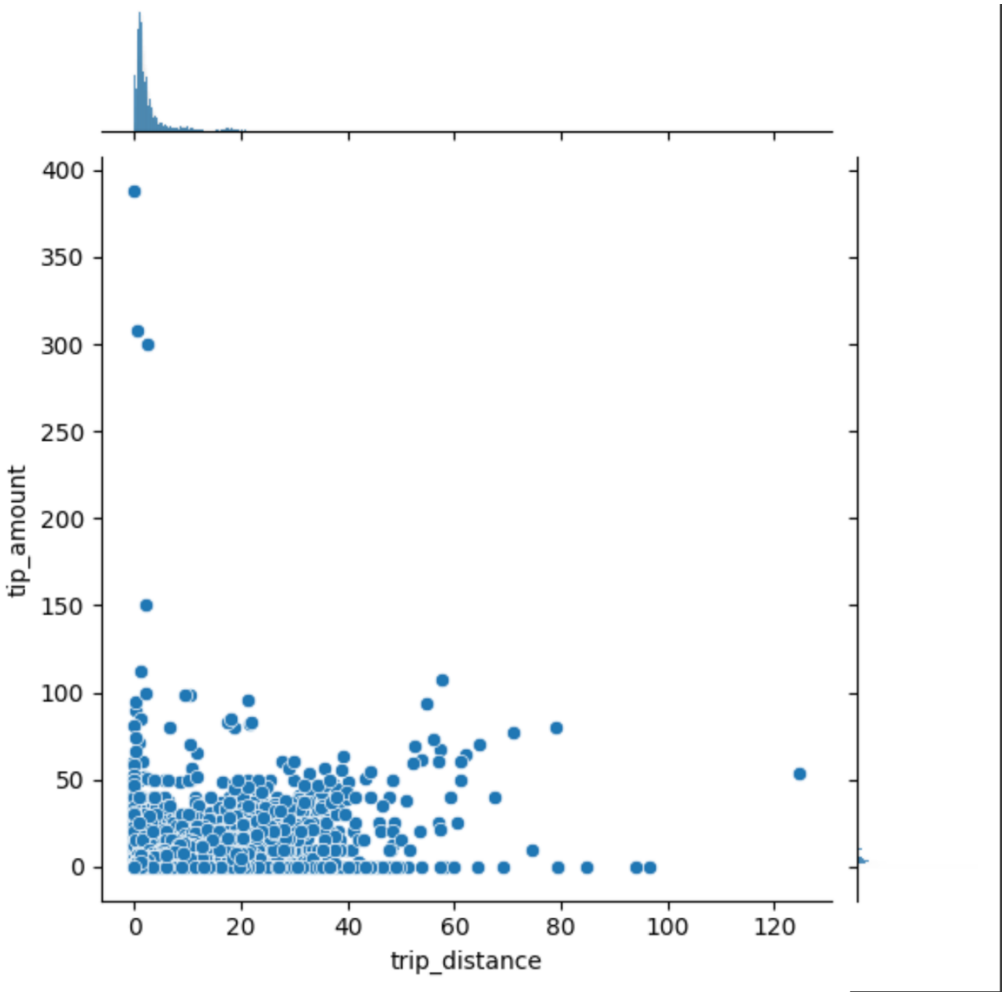
Passenger and Fare are not related


fare_amount
passenger_count
1
0.052
0.052
1
fare_amount
passenger_count
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


passenger_count
fare_amount
0
1
2
3
4
5
6
0
100
200
300
400
500
600

The tip and trip distance have some relation.



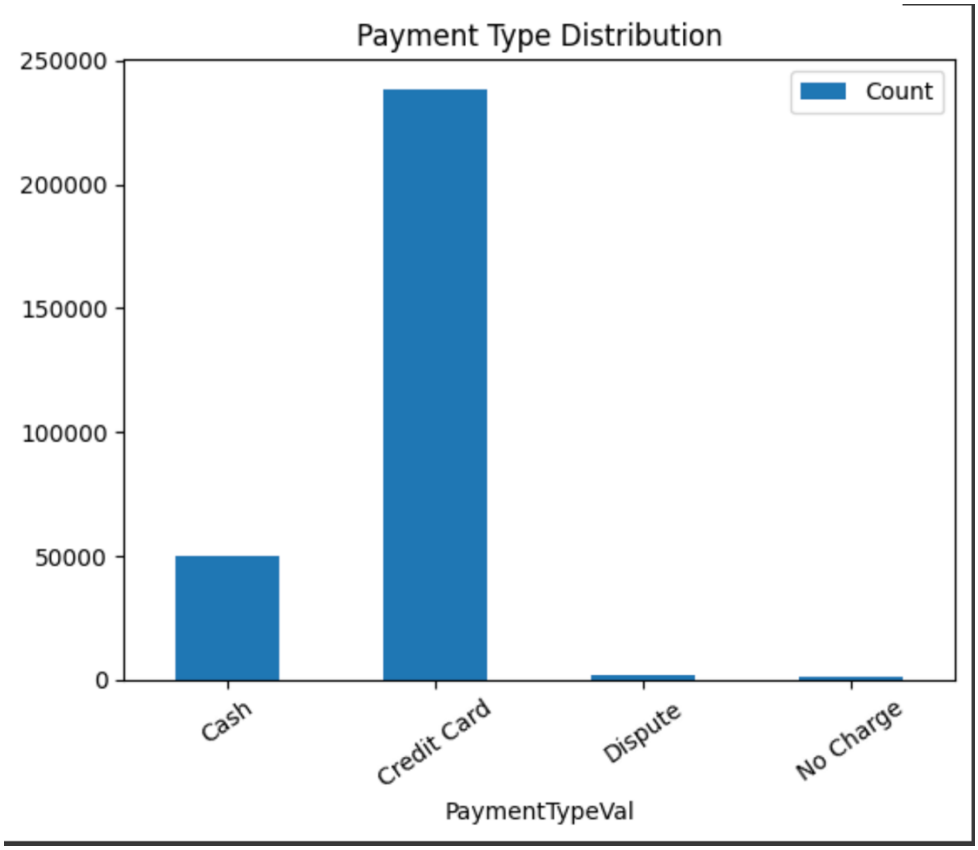


3.1.8. Analyse the distribution of different payment types

Most people use Credit card payment type followed by Cash.

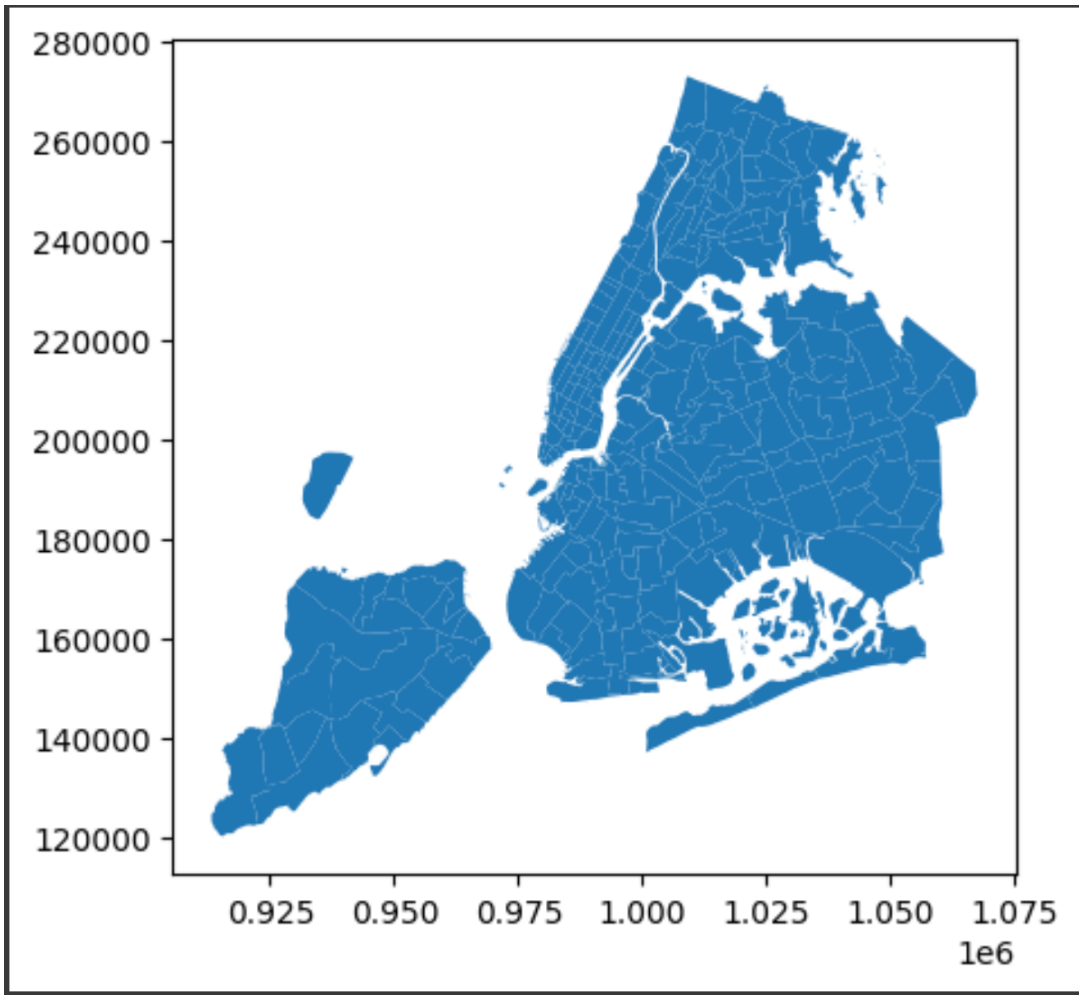
Payment Type Distribution

	payment_type	Count	PaymentTypeVal
0	1	238356	Credit Card
1	2	50302	Cash
2	3	1413	No Charge
3	4	2241	Dispute



3.1.9. Load the taxi zones shapefile and display it

taxi_zones.shp file was r



ead using gpd.read_file(shapefile) API

3.1.10. Merge the zone data with trips data

```
Data Size after merging taxi and zones data:  292334
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 292334 entries, 0 to 292333
Data columns (total 30 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   VendorID               292334 non-null  int64
 1   tpep_pickup_datetime   292334 non-null  datetime64[us]
 2   tpep_dropoff_datetime  292334 non-null  datetime64[us]
 3   passenger_count        292334 non-null  float64
 4   trip_distance          292334 non-null  float64
 5   RatecodeID             292334 non-null  float64
 6   PULocationID           292334 non-null  int64
 7   DOLocationID           292334 non-null  int64
 8   payment_type           292334 non-null  int64
 9   fare_amount            292334 non-null  float64
 10  extra                  292334 non-null  float64
 11  mta_tax                292334 non-null  float64
 12  tip_amount             292334 non-null  float64
 13  tolls_amount           292334 non-null  float64
 14  improvement_surcharge  292334 non-null  float64
 15  total_amount           292334 non-null  float64
 16  congestion_surcharge   292334 non-null  float64
 17  airport_fee            292334 non-null  float64
 18  Airport_fee            292334 non-null  float64
 19  Airport_Fee_Comb       292334 non-null  float64
 20  Hour                   292334 non-null  int32
 21  Day                    292334 non-null  category
 22  Month                  292334 non-null  category
 23  PaymentTypeVal         292334 non-null  object
 24  PUOBJECTID             289407 non-null  float64
 25  PUShape_Leng           289407 non-null  float64
 26  PUShape_Area           289407 non-null  float64
 27  PUzone                 289407 non-null  object
 28  PUborough              289407 non-null  object
 29  PUgeometry             289407 non-null  geometry
dtypes: category(2), datetime64[us](2), float64(17), geometry(1), int32(1), int64(4), object(3)
memory usage: 61.9+ MB
```

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

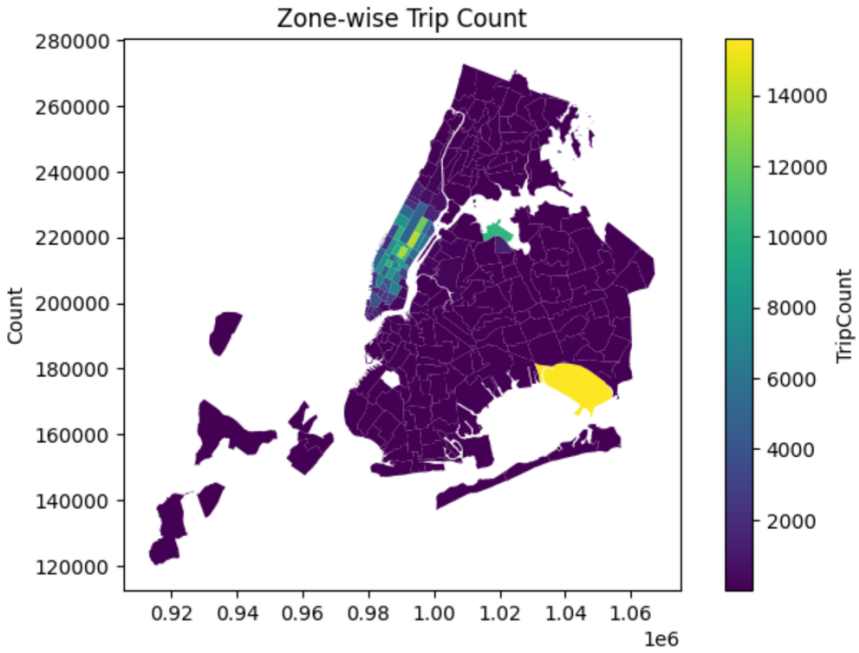
	PULocationID	TripCount
0	1	34
1	3	7
2	4	327
3	5	5
4	6	2
...	...	...
237	261	1498
238	262	3865
239	263	5558
240	264	2756
241	265	168

242 rows × 2 columns

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

```
Merged TripCount into zones dataframe


<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 263 entries, 0 to 262
Data columns (total 9 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
 7   PULocationID  240 non-null    float64
 8   TripCount     240 non-null    float64
dtypes: float64(4), geometry(1), int32(2), object(2)
memory usage: 16.6+ KB
```

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



Sorted Zones Df by using API sort_values as follows

zones.sort_values(by=['TripCount'])

	OBJECTID	Shape_Leng	Shape_Area	zone	LocationID	borough	geometry	PULocationID
156	170	0.045769	0.000074	Murray Hill	170	Manhattan	POLYGON ((991999.299 210994.739, 991972.635 21...	170
209	230	0.031028	0.000056	Times Sq/Theatre District	230	Manhattan	POLYGON ((988786.877 214532.094, 988650.277 21...	230
129	142	0.038176	0.000076	Lincoln Square East	142	Manhattan	POLYGON ((989380.305 218980.247, 989359.803 21...	142
170	186	0.024696	0.000037	Penn Station/Madison Sq West	186	Manhattan	POLYGON ((986752.603 210853.699, 986627.863 21...	186
125	138	0.107467	0.000537	LaGuardia Airport	138	Queens	MULTIPOLYGON (((1019904.219 225677.983, 102031...	138
148	162	0.035270	0.000048	Midtown East	162	Manhattan	POLYGON ((992224.354 214415.293, 992096.999 21...	162
215	236	0.044252	0.000103	Upper East Side North	236	Manhattan	POLYGON ((995940.048 221122.92, 995812.322 220...	236
147	161	0.035804	0.000072	Midtown Center	161	Manhattan	POLYGON ((991081.026 214453.698, 990952.644 21...	161
216	237	0.042213	0.000096	Upper East Side South	237	Manhattan	POLYGON ((993633.442 216961.016, 993507.232 21...	237
119	132	0.245479	0.002038	JFK Airport	132	Queens	MULTIPOLYGON (((1032791.001 181085.006, 103283...	132

3.1.14. Conclude with results

Busiest hours : 6 PM

Busiest Day : Thursday

Busiest Month: May & October

Trends in revenue collected: Max: October / Min: August

Trends in quarterly revenue: Max: Quarter 4 2023 / Min: Quarter 1 2023

As trip distances increases fare increases.

Fare doesn’t depend on the number of passenger count.

As trip duration increases, fare increases.

How tip amount depends on trip distance

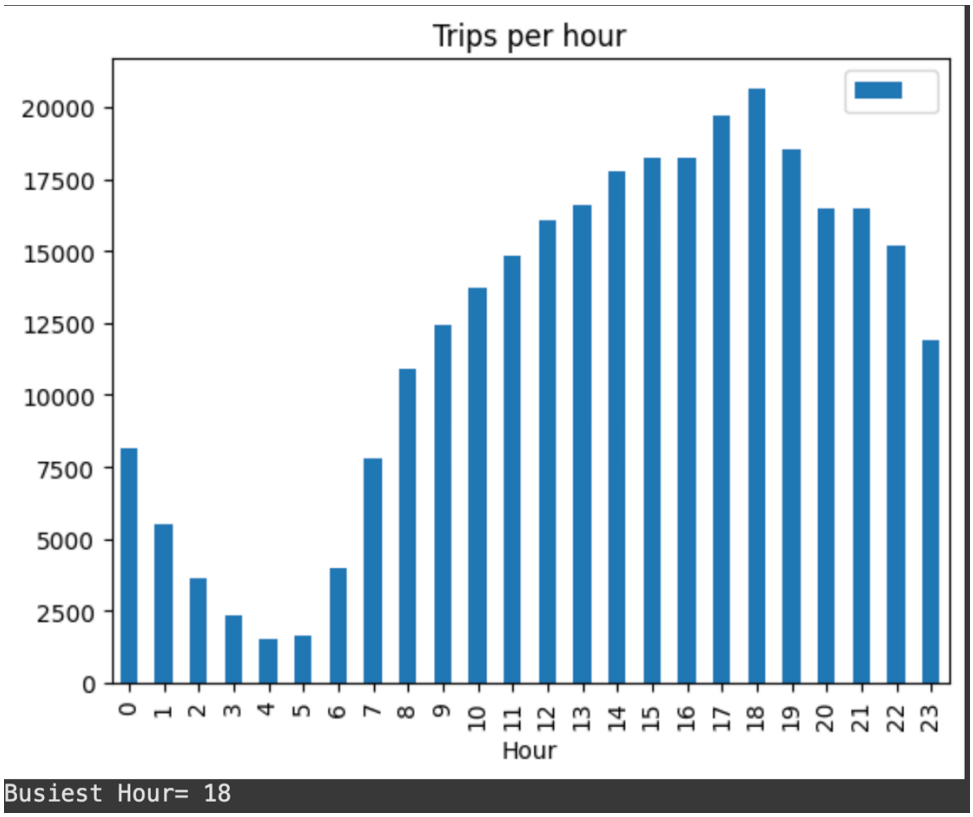
Most of the time the tip doesn’t depend on the trip distance

Busiest zones: JFK Airport

3.2. Detailed EDA: Insights and Strategies

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

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

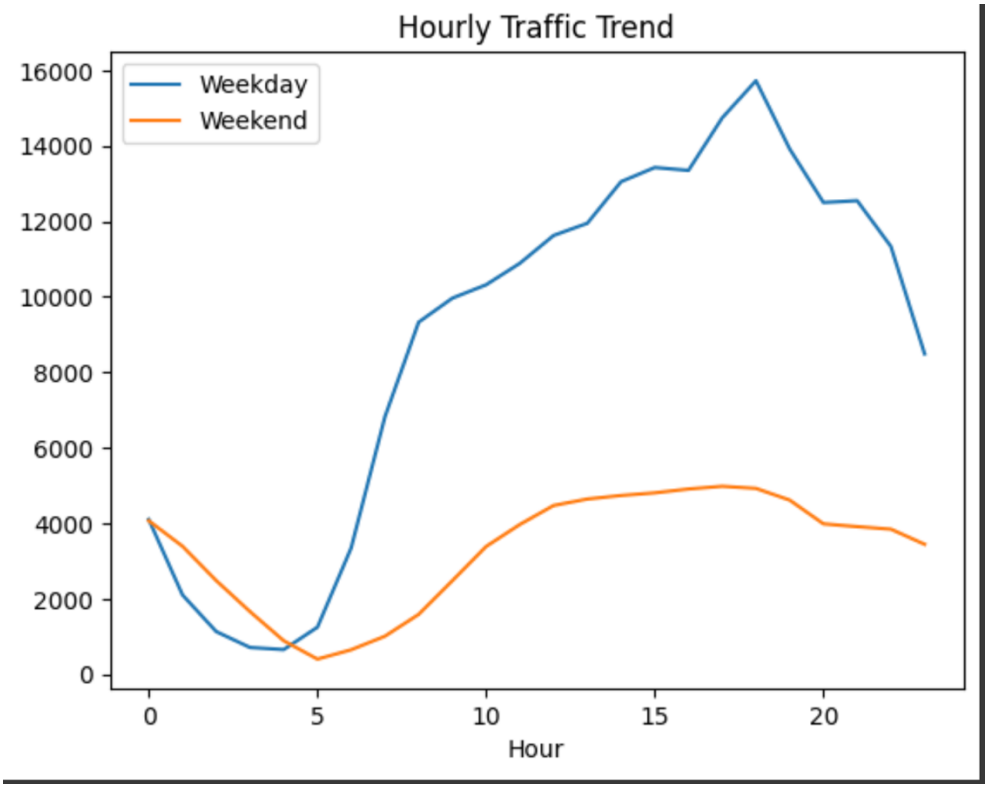


3.2.3. Scale up the number of trips from above to find the actual number of trips

3.2.4.

Hour	Sampled	Scaled
18	**20641**	**2580125.0**
17	**19714**	**2464250.0**
19	**18524**	**2315500.0**
16	**18247**	**2280875.0**
15	**18224**	**2278000.0**

3.2.5. Compare hourly traffic on weekdays and weekends



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

```
Top 10 Pickup Zones:
                          PUzone
238             Yorkville West
237             Yorkville East
236         World Trade Center
235                  Woodside
234         Woodlawn/Wakefield
233                 Woodhaven
232            Windsor Terrace
231  Williamsburg (South Side)
230  Williamsburg (North Side)
229   Williamsbridge/Olinville


Top 10 Drop Zones:
                          DOzone
255             Yorkville West
254             Yorkville East
253         World Trade Center
252                  Woodside
251         Woodlawn/Wakefield
250                 Woodhaven
249            Windsor Terrace
248  Williamsburg (South Side)
247  Williamsburg (North Side)
246   Williamsbridge/Olinville
```

3.2.7. Identify zones with high pickup and dropoff traffic during night hours (11PM to 5AM)

```
Top 10 Pickup Zones @Night:
176                 Yorkville West
175                 Yorkville East
174           World Trade Center
173                      Woodside
172                    Woodhaven
171                Windsor Terrace
170     Williamsburg (South Side)
169     Williamsburg (North Side)
168  Westchester Village/Unionport
167                  West Village



Top 10 Drop Zones @ Night:
238              Yorkville West
237              Yorkville East
236         World Trade Center
235                    Woodside
234         Woodlawn/Wakefield
233                  Woodhaven
232              Windsor Terrace
231  Williamsburg (South Side)
230  Williamsburg (North Side)
229   Williamsbridge/Olinville
```

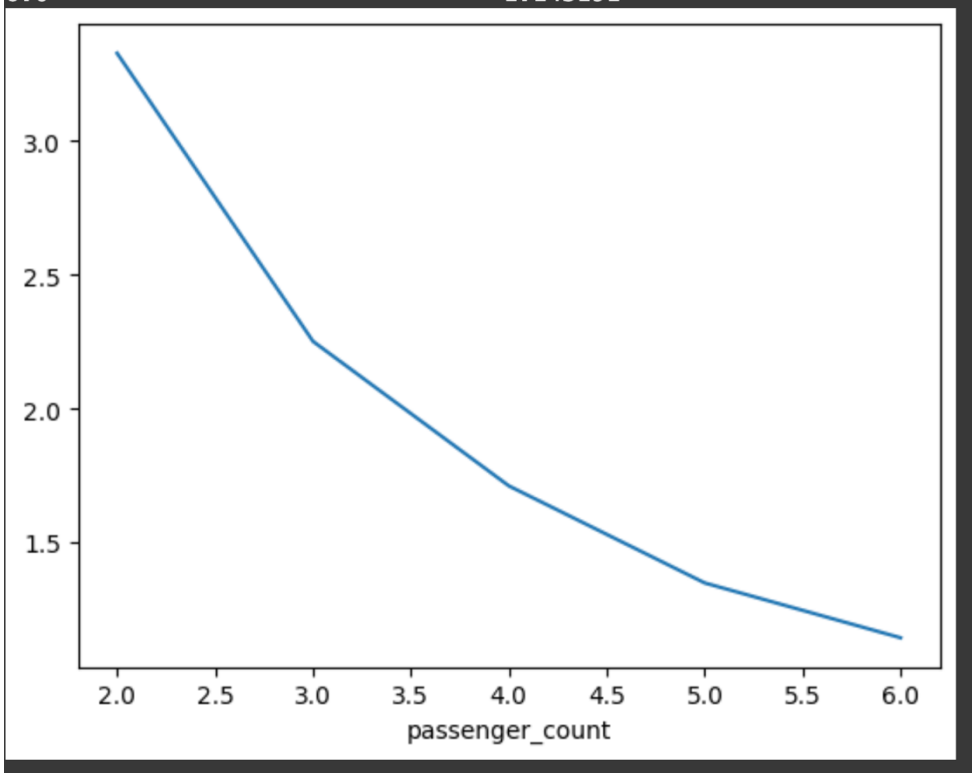
3.2.8. Identify the top zones with high traffic during night hours

3.2.9. Find the revenue share for nighttime and daytime hours

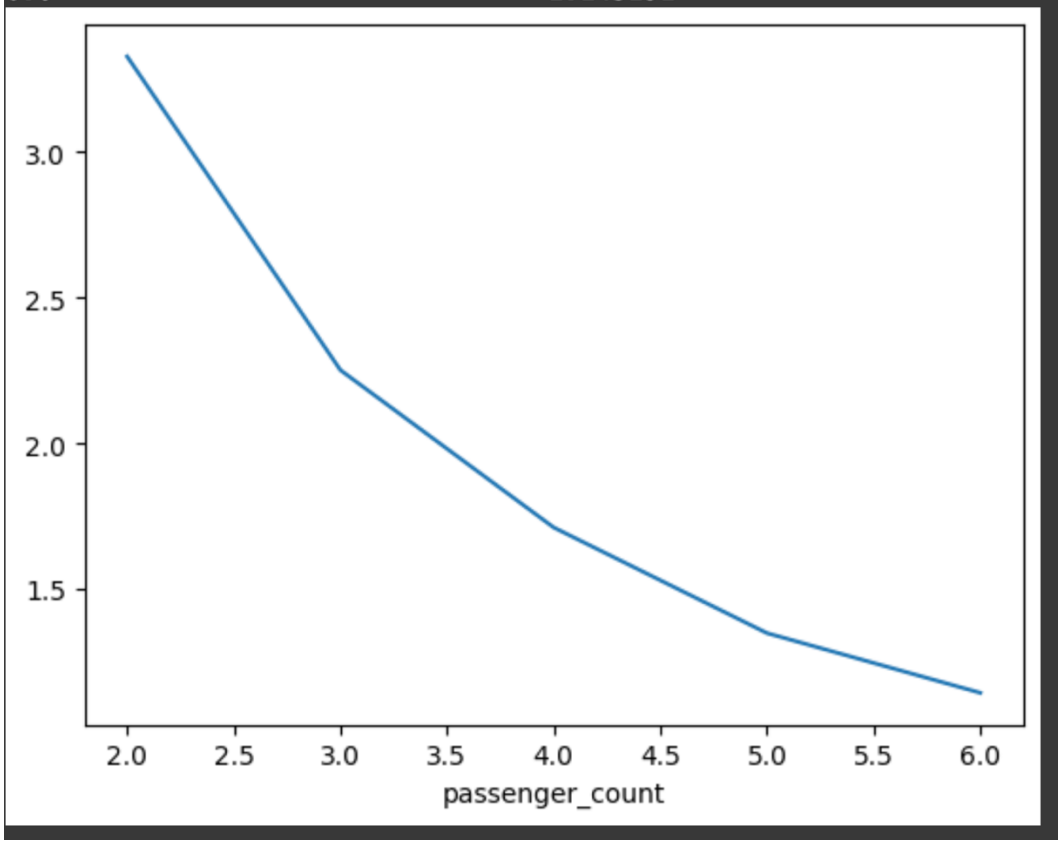
```
Total Day Time Revenue 5072399.34

Total Night Time Revenue 542008.96
```

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

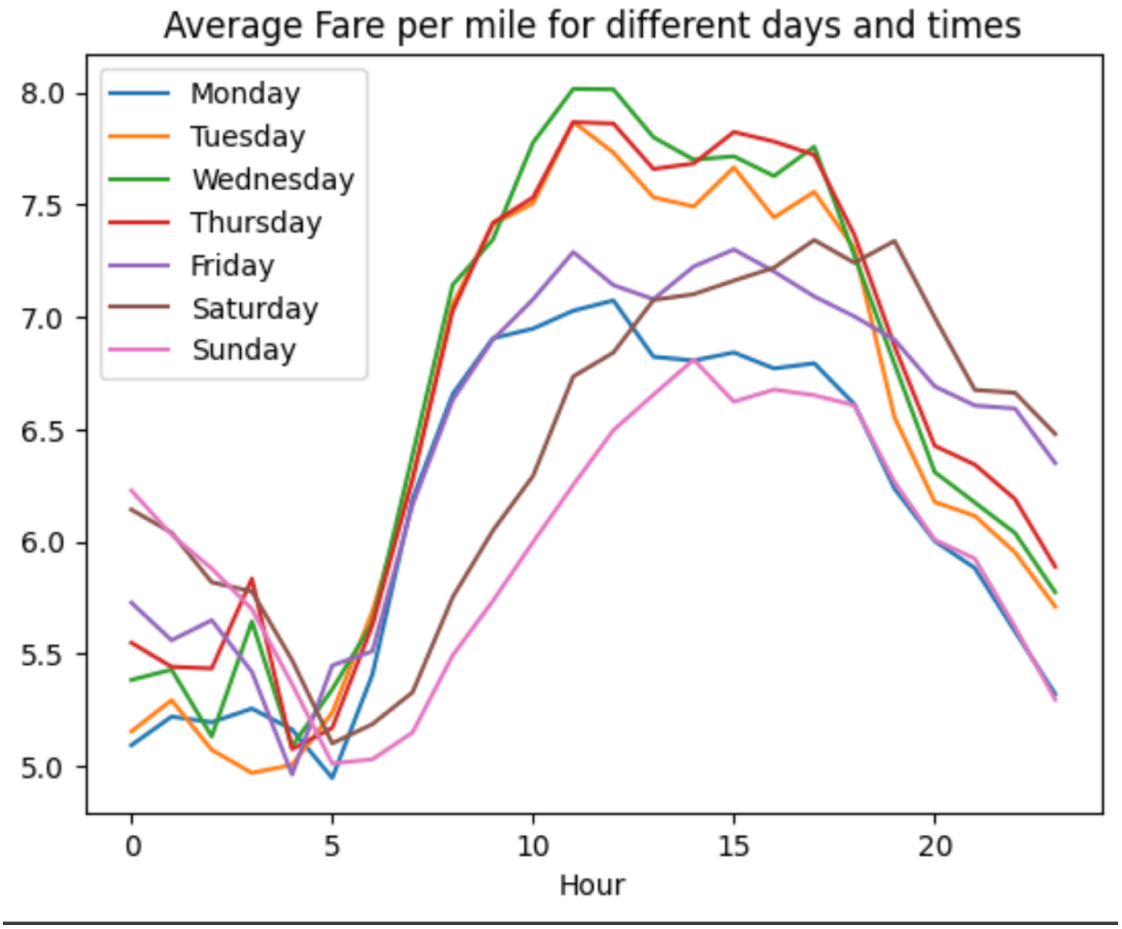
```
Fare per mile per pessanger
                 Fare_permile_perpessenger
passenger_count
2.0                                3.327751
3.0                                2.251090
4.0                                1.710815
5.0                                1.348119
6.0                                1.143191
```



```
Fare per mile per pessanger
                 Fare_permile_perpessenger
passenger_count
2.0                              3.327751
3.0                              2.251090
4.0                              1.710815
5.0                              1.348119
6.0                              1.143191
```

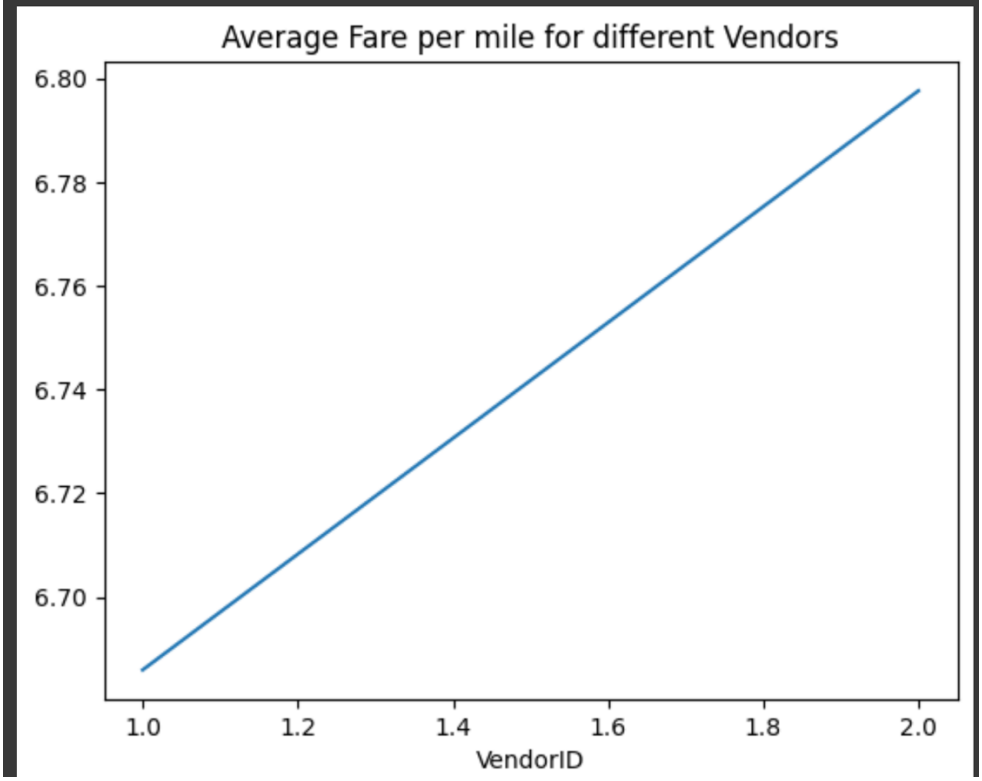


3.2.11. Find the average fare per mile by hours of the day and by days of the week



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

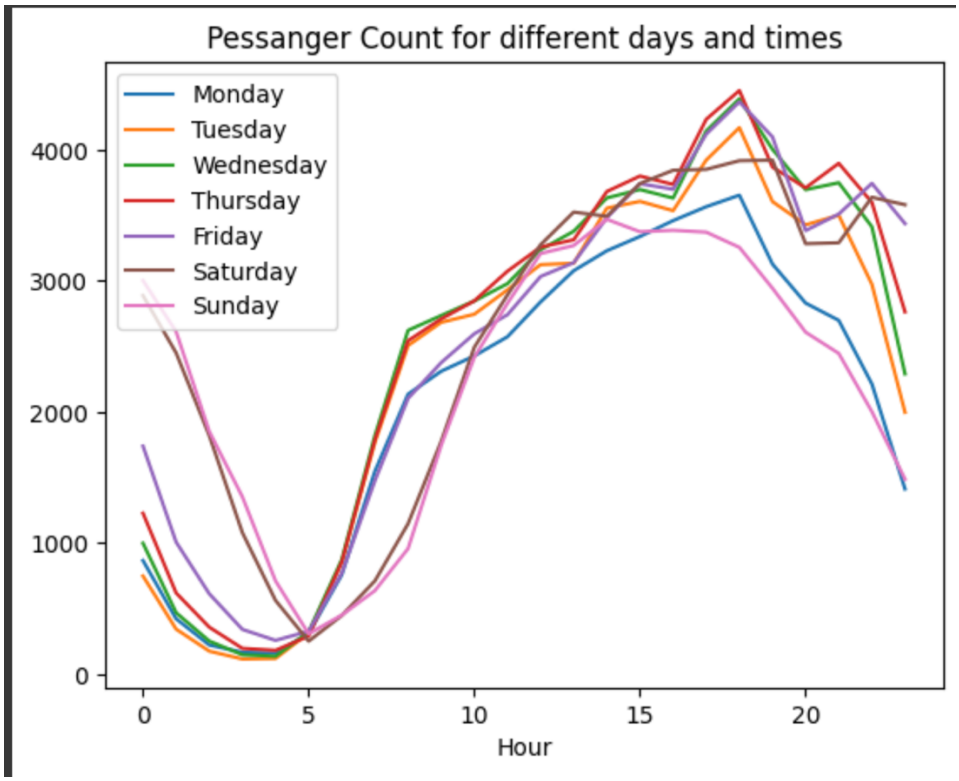
```
Avg. Fare-per-mile for different vendors
VendorID
1    6.685970
2    6.797567
Name: fare_per_mile, dtype: float64
```

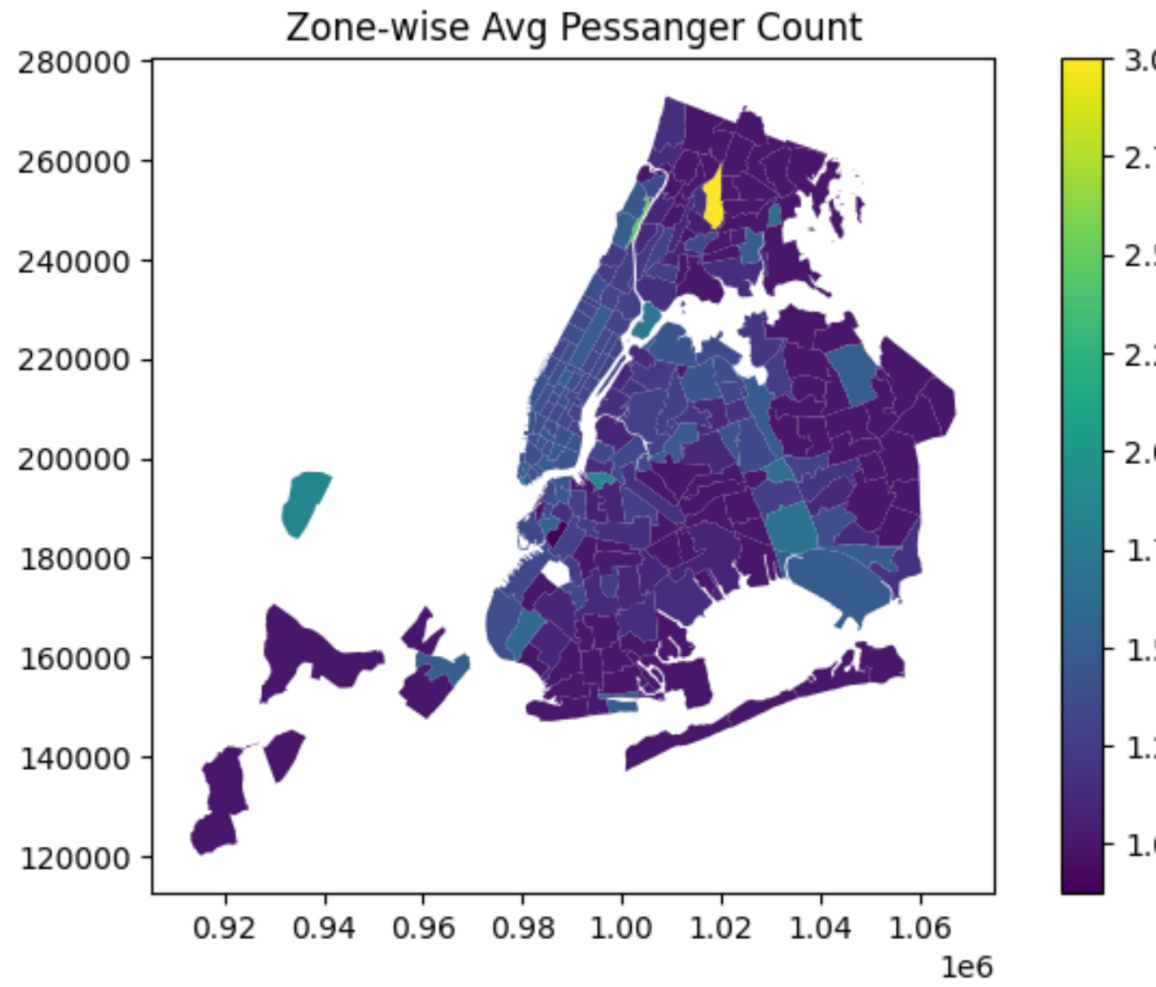


3.2.13. Compare the fare rates of different vendors in a distance-tiered fashion

3.2.14. Analyse the tip percentages

3.2.15. Analyse the trends in passenger count

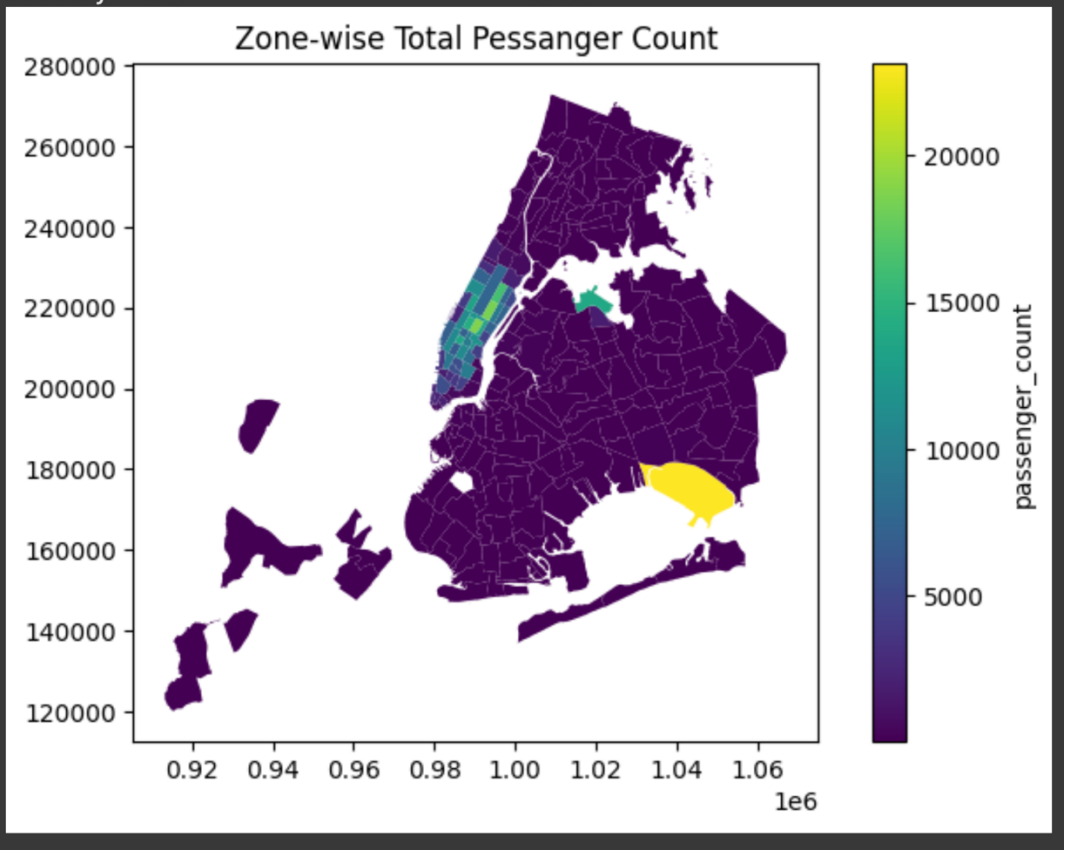




3.2.16. Analyse the variation of passenger counts across zones

Top 10 zones w.r.t pessanger count

PUzone	passenger_count
Allerton/Pelham Gardens	7.0
Alphabet City	473.0
Arden Heights	5.0
Arrochar/Fort Wadsworth	3.0
Astoria	186.0
Astoria Park	3.0
Auburndale	1.0
Baisley Park	134.0
Bath Beach	5.0
Battery Park	268.0



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

4. Conclusions

4.1. Final Insights and Recommendations

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

4.1.1.1. Allow cab sharing in JFK

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.

4.1.2.1. Have more cars for 6PM
4.1.2.2. Have more cars for JFK Geo location
4.1.2.3. Have more cars for Thursdays / Wednesday / Thursday
4.1.2.4. Months October/May/March are busy in that order and has more than usual passengers and needs more car

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