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

I loaded one file from the 12 files and then realized that it is a big data. So went ahead with sampling.

1.1.1. Sample the data and combine the files

I have used path from the D drive from my computer and took sampled of **1%** for each date and hour.

For the hour and date, I have used the column**, tpep_pickup_datetime** to extract the hour and date.

Then using joblib I saved the 1% data file to my computer in the form of parquet file named **“Samples File - 1pct”**

I named the DataFrame **“sampled_df”**

2. Data Cleaning

2.1. Fixing Columns

- **Fix the index**
  I brough the columns “hour” and “date” in index 1 and 2.

I removed the column **“tpep_pickup_datetime”**, but later in Section 3, I had to restore it due to the requirement for pickup time data.

2.1.1. Combine the two airport_fee columns

- Replaced NaN values with 1 to enable adding the two columns.
- Then I added the two columns in one column named **“total_airport_fee”**

2.2. Handling Missing Values

2.2.1. Find the proportion of missing values in each column

```
sampled_df.isna().sum()

VendorID                      0
hour                          0
tpep_dropoff_datetime         0
passenger_count           14406
trip_distance                 0
RatecodeID                14406
store_and_fwd_flag        14406
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      14406
date                          0
trip_duration                 0
total_airport_fee             0
dtype: int64
```

Columns having null values: passenger_count, RatecodeID, store_and_fwd_flag, congestion_surcharge

```
numeric_df = sampled_df.select_dtypes(include='number')

negative_counts = (numeric_df < 0).sum()

print(negative_counts)

VendorID                  0
hour                      0
passenger_count           0
trip_distance             0
RatecodeID                0
PULocationID              0
DOLocationID              0
payment_type              0
fare_amount               0
extra                     0
mta_tax                  17
tip_amount                0
tolls_amount              0
improvement_surcharge    20
total_amount             20
congestion_surcharge     12
trip_duration            22
total_airport_fee         5
dtype: int64
```

Columns having negative values: mta_tax, improvemnet_surcharge, total_amount, congestion_surcharge, trip_duration, total_airport_fee

2.2.2. Handling missing values in passenger_count

Number of null entries in 'passenger_count': 14406
Median of entries in 'passenger_count': 1.0
So I replaced the missing passenger count by median value.

Zero entries in 'passenger_count': 6907
Percentage of zero entries in 'passenger_count': 1.54%, hence we may drop this small number of rows.
The passenger count rows with zero entries is dropped.

2.2.3. Handle missing values in RatecodeID

Number of null entries in 'RatecodeID': 14406
Percentage of null entries in 'RatecodeID': 3.2547326469611426% .

Unusual value "99" is seen in the RatecodeID data.

Dropped RatecodeID with 99 and replace NaN values with 1 (i .e the usual ratecode ID as per data).

2.2.4. Impute NaN in congestion_surcharge

The count of null values in congestion_surcharge is:14418

The mean of congestion_surcharge is:2.32

The median of congestion_surcharge is:2.5

Null values are replaced with median value "2.5"

Impute NaN in store_and_fwd_flag

N: 423337

Y: 2377

Null: 14406

Total: 440120

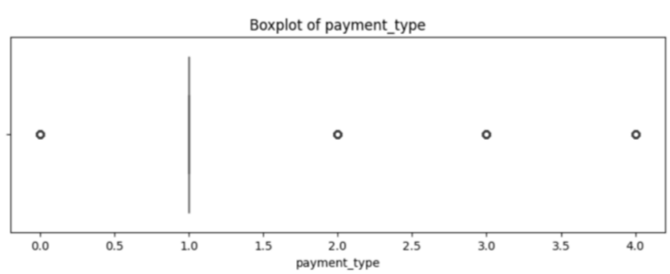
Null values are replaced by N

2.3. Handling Outliers and Standardising Values

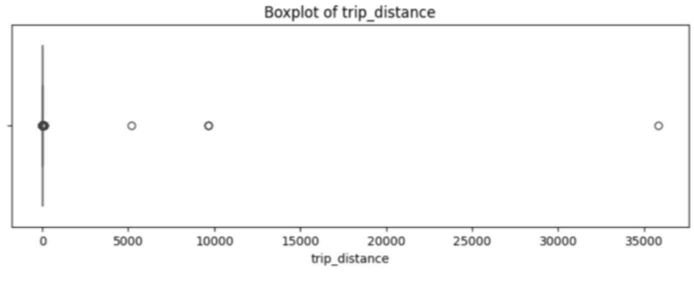
2.3.1. Check outliers in payment type, trip distance and tip amount columns

The outliers are identified using box plot.

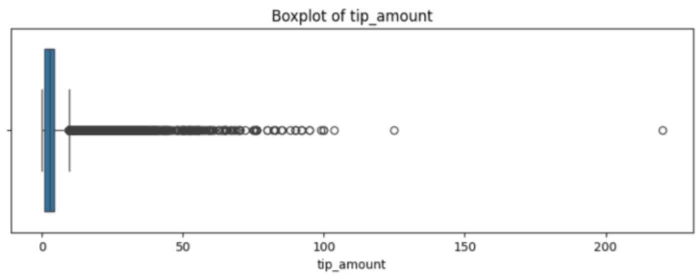
Payment type:



Trip distance:



Tip amount:



3. Exploratory Data Analysis

3.1. General EDA: Finding Patterns and Trends

Classify variables into categorical and numerical

Categorise the varaibles into Numerical or Categorical.

VendorID: Categorical

tpep_pickup_datetime: Date time - Numerical

tpep_dropoff_datetime: Date time - Numerical

passenger_count:Categorical

trip_distance: Numerical

RatecodeID:Categorical

PULocationID: Numerical

DOLocationID: Numerical

payment_type:Categorical

pickup_hour:Categorical

trip_duration: Numerical

fare_amount: Numerical

extra: Numerical

mta_tax: Numerical

tip_amount: Numerical

tolls_amount: Categorical

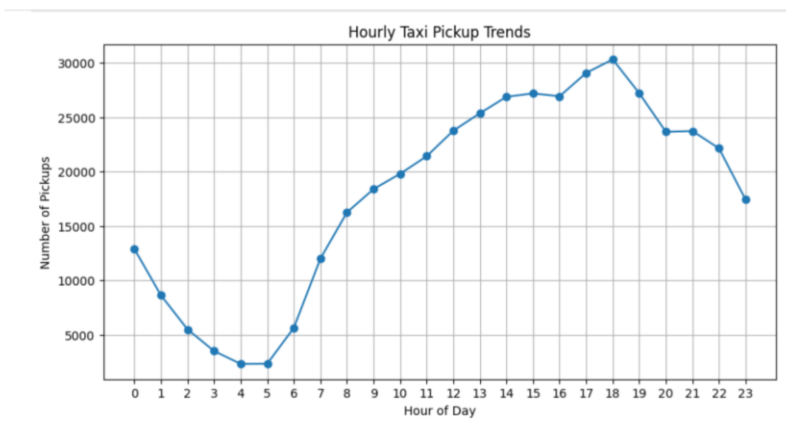
improvement_surcharge: Categorical

total_amount: Numerical

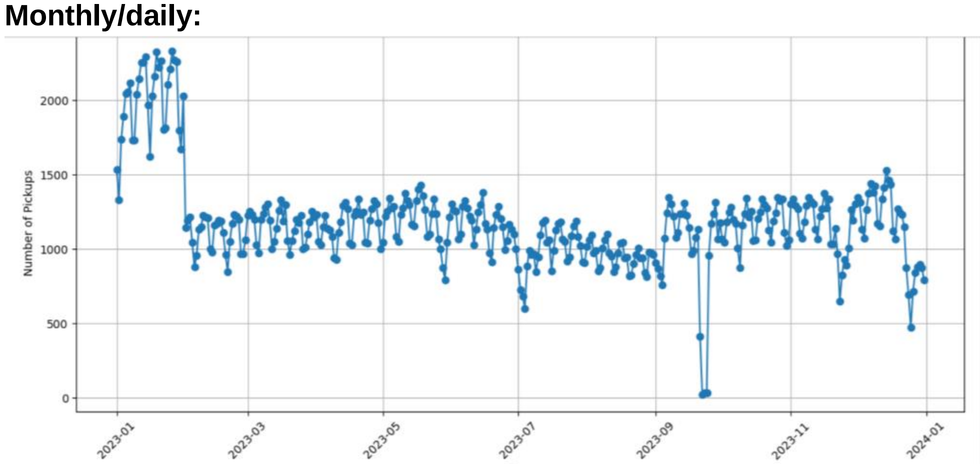
congestion_surcharge: Categorical

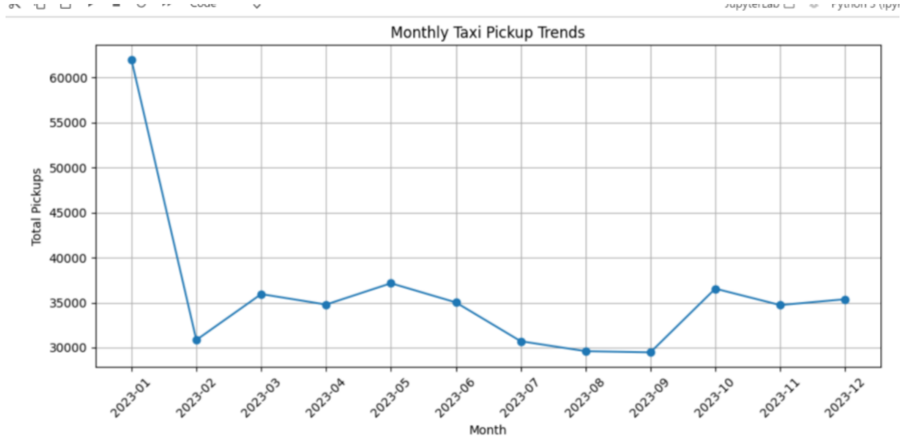
airport_fee: Numerical

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



Monthly/daily:



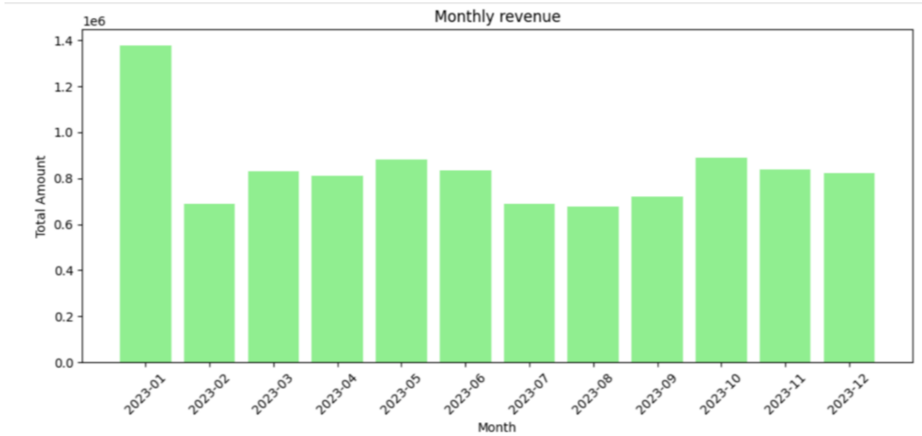


3.1.2. Filter out the zero/negative values in fares, distance and tips

I had filtered negative columns in the second section.

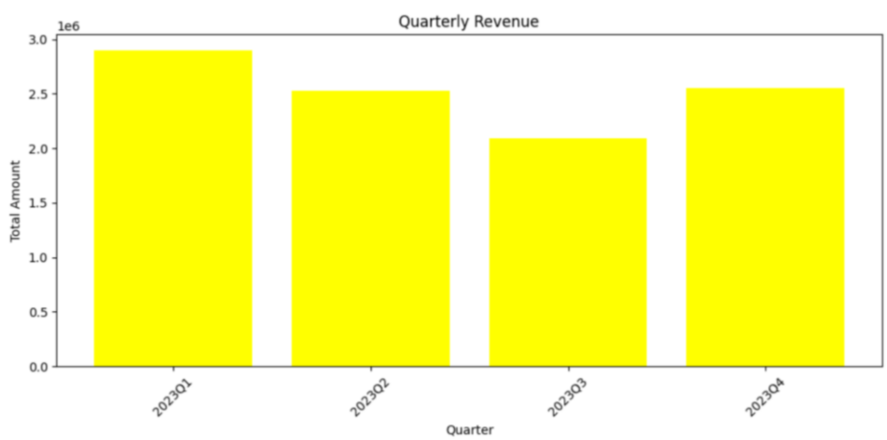
Analyse the monthly revenue trends

	month	total_amount
0	2023-01	1377752.22
1	2023-02	689164.54
2	2023-03	831487.49
3	2023-04	810260.33
4	2023-05	883369.36
5	2023-06	834821.79
6	2023-07	689553.27
7	2023-08	676301.55
8	2023-09	722130.83
9	2023-10	890484.14
10	2023-11	837455.22
11	2023-12	823104.01

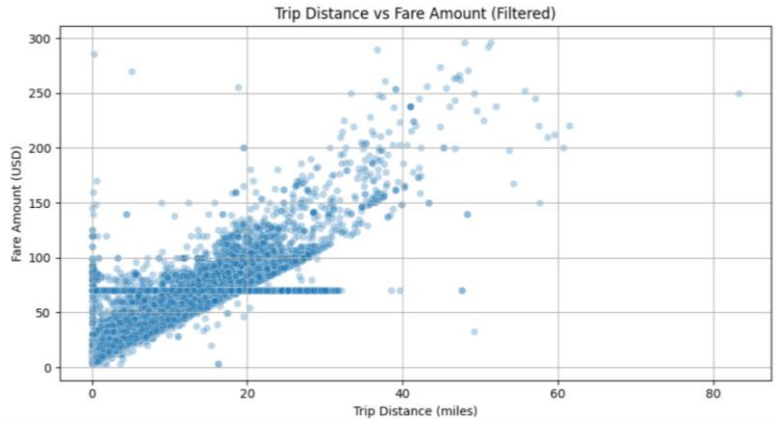


3.1.3. Find the proportion of each quarter's revenue in the yearly revenue

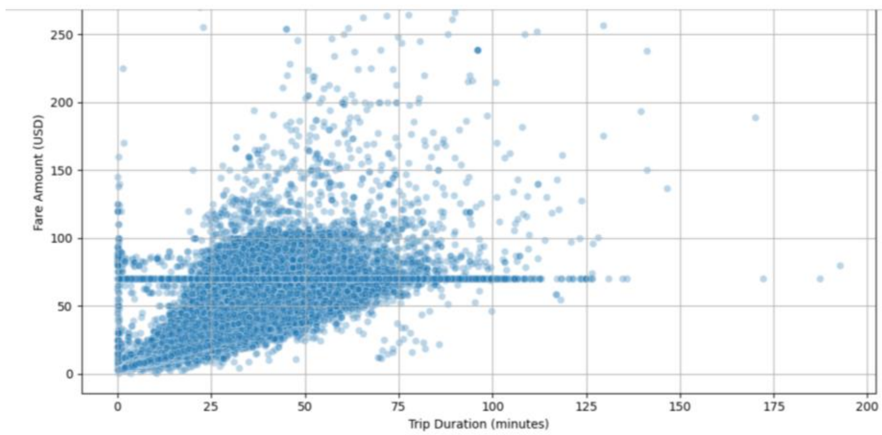
	quarter	total_amount
0	2023Q1	2898404.25
1	2023Q2	2528451.48
2	2023Q3	2087985.65
3	2023Q4	2551043.37

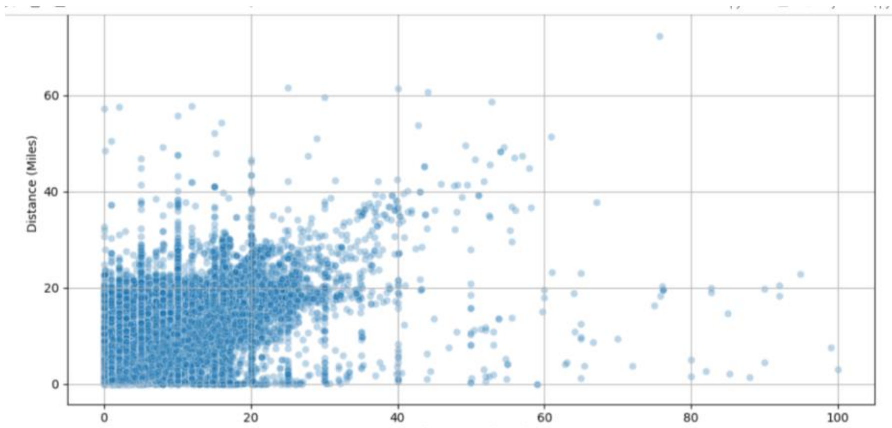


3.1.4. Analyse and visualise the relationship between distance and fare amount



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





3.1.6. Analyse the distribution of different payment types

Payment Type Percentages:
 payment_type
1   99.99

2	0.00
4	0.00
3	0.00

3.1.7. Load the taxi zones shapefile and display it

```
# import geopandas as gpd
import geopandas as gpd

# Read the shapefile using geopandas
#zones = # read the .shp file using gpd
zones=gpd.read_file(r"D:\upgrad\EDA\12 files of NYC yellow taxi data\Datasets and Dictionary\taxi_zones\taxi_zones.shp")
zones.head()
```

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

3.1.8. Merge the zone data with trips data

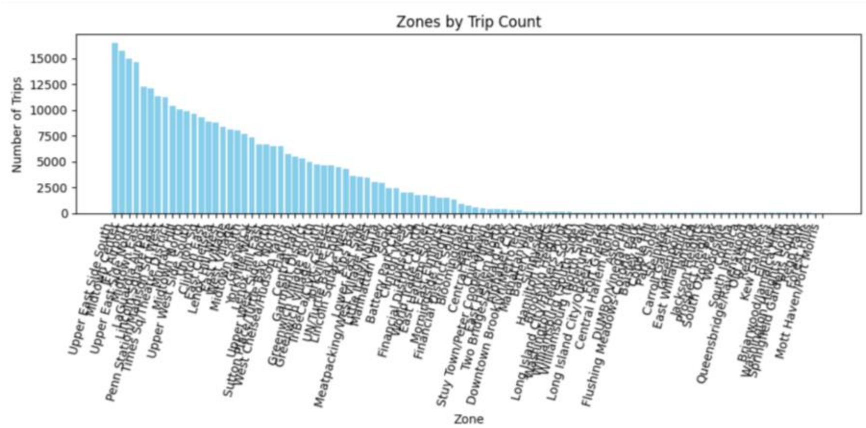
merged_df = sampled_df.merge(zones, left_on='PULocationID', right_on='LocationID', how='left')

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

```
# Group data by location and calculate the number of trips

LocationID_counts = merged_df.groupby('LocationID').size().reset_index(name='trip_
print(LocationID_counts)
```

```
     LocationID  trip_count
0           1.0           7
1           2.0           2
2           3.0           1
3           4.0         322
4           7.0          77
..          ...         ...
169       259.0           1
170       260.0          28
171       261.0        1630
172       262.0        4447
173       263.0        6651
```



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

	OBJECTID	Shape_Leng	Shape_Area	zone	LocationID	borough	geometry	trip_count
236	237	0.042213	0.000096	Upper East Side South	237	Manhattan	POLYGON ((993633.442 216961.016, 993507.232 21...	16506
160	161	0.035804	0.000072	Midtown Center	161	Manhattan	POLYGON ((991081.026 214453.698, 990952.644 21...	15756
131	132	0.245479	0.002038	JFK Airport	132	Queens	MULTIPOLYGON (((1032791.001 181085.006, 103283...	14989
235	236	0.044252	0.000103	Upper East Side North	236	Manhattan	POLYGON ((995940.048 221122.92, 995812.322 220...	14656
161	162	0.035270	0.000048	Midtown East	162	Manhattan	POLYGON ((992224.354 214415.293, 992096.999 21...	12269
137	138	0.107467	0.000537	LaGuardia Airport	138	Queens	MULTIPOLYGON (((1019904.219 225677.983, 102031...	12067
141	142	0.038176	0.000076	Lincoln Square East	142	Manhattan	POLYGON ((989380.305 218980.247, 989359.803 21...	11304
185	186	0.024696	0.000037	Penn Station/Madison Sq West	186	Manhattan	POLYGON ((986752.603 210853.699, 986627.863 21...	11230
229	230	0.031028	0.000056	Times Sq/Theatre District	230	Manhattan	POLYGON ((988786.877 214532.094, 988650.277 21...	10381
169	170	0.045769	0.000074	Murray Hill	170	Manhattan	POLYGON ((991999.299 210994.739, 991972.635 21...	10103
162	163	0.034177	0.000041	Midtown North	163	Manhattan	POLYGON ((989412.663 219020.943, 990045.841 21...	9884
238	239	0.063626	0.000205	Upper West Side South	239	Manhattan	POLYGON ((991168.979 226252.992, 991955.565 22...	9642
233	234	0.036072	0.000073	Union Sq	234	Manhattan	POLYGON ((987029.847 207022.299, 987048.27 206...	9260
47	48	0.043747	0.000094	Clinton East	48	Manhattan	POLYGON ((986694.313 214463.846, 986568.184 21...	8880
67	68	0.049337	0.000111	East Chelsea	68	Manhattan	POLYGON ((983690.405 209040.369, 983550.612 20...	8789

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

3.1.12. Conclude with results

- Busiest hours, days and months:
  Busiest months in descending order: January, October, May, November, June, march, Descember, April, September, February, July, August.
  Busiest dates: January 2023
  Hours: 10^{th} -24^{th} hour. Highest picups in 19^{th} hour
- Trends in revenue collected:
  Monthly revenue collected:
  0 2023-01 1377752.22
  1 2023-02 689164.54
  2 2023-03 831487.49
  3 2023-04 810260.33
  4 2023-05 883369.36

5 2023-06 834821.79
6 2023-07 689553.27
7 2023-08 676301.55
8 2023-09 722130.83
9 2023-10 890484.14
10 2023-11 837455.22
11 2023-12 823104.01

- Trends in quarterly revenue:
  quarter total_amount
  0 2023Q1 2898404.25
  1 2023Q2 2528451.48
  2 2023Q3 2087985.65
  3 2023Q4 2551043.37

- How fare depends on trip distance, trip duration and passenger counts:
  - fare amount is mostly equally proportionate to Trip distance.
  - Generally long trip duration implies long distances. Hence, fare in crease as duration increases, but- some entries that needs to be considered where fare is the same even if durations are long.
  - Graphical representations indicate that passenger count is inversely proportional to fare amount.

- How tip amount depends on trip distance:
  - Tip amount is equally proportionate to trip distances.
- Busiest zones:

	OBJECTID	Shape_Leng	Shape_Area	zone	LocationID	borough	geometry	trip_count
236	237	0.042213	0.000096	Upper East Side South	237	Manhattan	POLYGON ((993633.442 216961.016, 993507.232 21...	16506
160	161	0.035804	0.000072	Midtown Center	161	Manhattan	POLYGON ((991081.026 214453.698, 990952.644 21...	15756
131	132	0.245479	0.002038	JFK Airport	132	Queens	MULTIPOLYGON (((1032791.001 181085.006, 103283...	14989
235	236	0.044252	0.000103	Upper East Side North	236	Manhattan	POLYGON ((995940.048 221122.92, 995812.322 220...	14656
161	162	0.035270	0.000048	Midtown East	162	Manhattan	POLYGON ((992224.354 214415.293, 992096.999 21...	12269
137	138	0.107467	0.000537	LaGuardia Airport	138	Queens	MULTIPOLYGON (((1019904.219 225677.983, 102031...	12067
141	142	0.038176	0.000076	Lincoln Square East	142	Manhattan	POLYGON ((989380.305 218980.247, 989359.803 21...	11304
185	186	0.024696	0.000037	Penn Station/Madison Sq West	186	Manhattan	POLYGON ((986752.603 210853.699, 986627.863 21...	11230
229	230	0.031028	0.000056	Times Sq/Theatre District	230	Manhattan	POLYGON ((988786.877 214532.094, 988650.277 21...	10381
169	170	0.045769	0.000074	Murray Hill	170	Manhattan	POLYGON ((991999.299 210994.739, 991972.635 21...	10103
162	163	0.034177	0.000041	Midtown North	163	Manhattan	POLYGON ((989412.663 219020.943, 990045.841 21...	9884
238	239	0.063626	0.000205	Upper West Side South	239	Manhattan	POLYGON ((991168.979 226252.992, 991955.565 22...	9642
233	234	0.036072	0.000073	Union Sq	234	Manhattan	POLYGON ((987029.847 207022.299, 987048.27 206...	9260
47	48	0.043747	0.000094	Clinton East	48	Manhattan	POLYGON ((986694.313 214463.846, 986568.184 21...	8880

3.2. Detailed EDA: Insights and Strategies

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

```
merged_df['speed']=merged_df['trip_distance']/merged_df['trip_duration']
print("Speed is calculated in units: miles/minute")
merged_df['speed'].head(20)
```

```
Speed is calculated in units: miles/minute

0     0.193388
1     0.255670
2     0.472015
3     0.181818
4     0.226214
5     0.173913
6     0.532686
7     0.523675
8     0.306569
9     0.429412
10    0.148662
11    0.408987
12    0.177841
13    0.128878
14    0.168782
15    0.173576
16    0.301898
17    0.434191
18    0.158463
19    0.226829
Name: speed, dtype: float64
```

fast zones

```
sort_merged_df[['speed','zone']].head(40)
```

	speed	zone
274447	733.771176	Midtown South
314966	623.512782	Morningside Heights
329667	623.512782	Morningside Heights
170004	243.600000	Manhattan Valley
57211	216.000000	Battery Park City
968	194.400000	Flatbush/Ditmas Park
279837	194.400000	Flatbush/Ditmas Park
121092	164.100000	Jackson Heights
88811	132.000000	Lenox Hill West
245739	74.210526	Washington Heights North
78873	68.666667	Lenox Hill West
115204	37.400000	West Village
130637	36.000000	Times Sq/Theatre District
238546	26.890909	East Elmhurst

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

	hour	hourly_trip_count
18	18	23693
17	17	22358
19	19	21471
16	16	20147
15	15	20034

```
hourly_sorted_trip['hourly_trip_count'].max()
```

```
23693
```

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

```python
# Scale up the number of trips

# Fill in the value of your sampling fraction and use that to scale up the numbers
sample_fraction = 0.01
sampled_trips=zones_with_counts['trip_count'].sum()


total_trips=sampled_trips/sample_fraction
print("total number of trips\n:",total_trips)
print("\nSampled trips\n:",sampled_trips)
```

```
total number of trips
: 32795300.0

Sampled trips
: 327953
```

3.2.4. Compare hourly traffic on weekdays and weekends

```python
merged_df[['day_name','trip_count']].value_counts()
```

```
day_name   trip_count
Thursday   52562        52562
Wednesday  51055        51055
Tuesday    48843        48843
Friday     48782        48782
Saturday   47613        47613
Sunday     41216        41216
Monday     40882        40882
Name: count, dtype: int64
```

```
day_name   hour  trip_count
Thursday   18    52562          4054
Wednesday  18    51055          3947
Tuesday    18    48843          3766
Thursday   17    52562          3663
           19    52562          3610
Wednesday  17    51055          3548
           19    51055          3485
Tuesday    17    48843          3473
Friday     18    48782          3464
           17    48782          3394
Thursday   21    52562          3381
Friday     19    48782          3364
Tuesday    19    48843          3333
Wednesday  21    51055          3266
           20    51055          3240
Thursday   15    52562          3191
Monday     18    40882          3190
Tuesday    21    48843          3170
Wednesday  14    51055          3120
Tuesday    16    48843          3119
Thursday   20    52562          3106
           16    52562          3062
Wednesday  16    51055          3053
Tuesday    15    48843          3033
Thursday   22    52562          3030
Tuesday    20    48843          3013
Wednesday  15    51055          3002
```

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

```python
pick_drop_zone = merged_df[['zone', 'dropoff_hour','pickup_hour' , 'speed']].value_counts()
pick_drop_zone.head(10)
```

```
zone                     dropoff_hour         pickup_hour                    speed
Upper West Side North    2023-01-04 11:00:00  2023-01-03 17:40:00.000000002  0.148846    6
Greenwich Village North  2023-08-09 09:00:00  2023-08-08 19:04:00.000000002  0.180144    5
Battery Park City        2023-04-21 18:00:00  2023-04-20 22:01:00.000000002  0.220183    5
Kips Bay                 2023-12-11 20:00:00  2023-12-11 12:58:00.000000002  0.110900    5
East Village             2023-02-13 13:00:00  2023-02-13 05:18:00.000000000  0.181818    5
Upper West Side South    2023-06-23 11:00:00  2023-06-22 16:34:00.000000002  0.097649    5
Lenox Hill West          2023-11-21 09:00:00  2023-11-20 23:24:00.000000000  0.104167    5
Greenwich Village South  2023-02-24 20:00:00  2023-02-24 07:25:00.000000002  0.080265    5
Lenox Hill West          2023-05-04 15:00:00  2023-05-03 13:13:00.000000002  0.232321    5
Lincoln Square West      2023-12-18 08:00:00  2023-12-18 01:46:59.999999999  0.167292    5
Name: count, dtype: int64
```

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

NaN in dropoff zone: 0
Top 10

:	PULocationID	pickup_zone	pickup_count	DOLocationID	dropoff_zone	dropoff_count	pickup_dropoff_ratio
0	70.0	East Elmhurst	1501.0	70.0	East Elmhurst	96.0	15.635417
1	132.0	JFK Airport	14989.0	132.0	JFK Airport	3145.0	4.765978
2	138.0	LaGuardia Airport	12067.0	138.0	LaGuardia Airport	4176.0	2.889607
3	186.0	Penn Station/Madison Sq West	11230.0	186.0	Penn Station/Madison Sq West	6955.0	1.614666
4	114.0	Greenwich Village South	4673.0	114.0	Greenwich Village South	3411.0	1.369979
5	249.0	West Village	8037.0	249.0	West Village	5969.0	1.346457
6	43.0	Central Park	5291.0	43.0	Central Park	4054.0	1.305131
7	162.0	Midtown East	12269.0	162.0	Midtown East	9695.0	1.265498
8	93.0	Flushing Meadows-Corona Park	68.0	93.0	Flushing Meadows-Corona Park	55.0	1.236364
9	161.0	Midtown Center	15756.0	161.0	Midtown Center	12760.0	1.234796

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

Night hours are considered from hours<4th and >22nd on the scale of 0 to 23.

```python
nigh_hour_data = zone_counts_hourly[(zone_counts_hourly['hour'] < 4) | (zone_counts_hourly['hour'] > 22)].value_counts()
nigh_hour_data.head(10)
```

PULocationID	hour	pickup_count_hour	DOLocationID	dropoff_count_hour	dropoff_zone	pickup_zone	
56.0	1	4.0	56.0	1.0	Corona	Corona	4
4.0	0	47.0	4.0	74.0	Alphabet City	Alphabet City	1
189.0	0	3.0	189.0	13.0	Prospect Heights	Prospect Heights	1
186.0	23	506.0	186.0	191.0	Penn Station/Madison Sq West	Penn Station/Madison Sq West	1
	3	38.0	186.0	37.0	Penn Station/Madison Sq West	Penn Station/Madison Sq West	1
	2	105.0	186.0	53.0	Penn Station/Madison Sq West	Penn Station/Madison Sq West	1
	1	169.0	186.0	71.0	Penn Station/Madison Sq West	Penn Station/Madison Sq West	1
	0	323.0	186.0	133.0	Penn Station/Madison Sq West	Penn Station/Madison Sq West	1
181.0	23	8.0	181.0	98.0	Park Slope	Park Slope	1
	1	5.0	181.0	41.0	Park Slope	Park Slope	1

Name: count, dtype: int64

3.2.8. Find the revenue share for nighttime and daytime hours

Total amount(Dat time): 9783594.000000004

Total amount(Night time): 1073475.9300000002

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

Trips with 1 passenger

Total amount for trips with 1 passenger is: 7377952.77

Total distance in miles for trips with 1 passenger is: 870686.84

Hance per mile fare is: 8.473715727689187

Therefore, fare per mile per passenger will be : 8.473715727689187 USD/mile

Trips with 2 passenger

Total amount for trips with 2 passenger is: 1608448.38

Total distance in miles for trips with 2 passenger is: 198926.78

Hance per mile fare is: 8.085630200217386

Therefore, fare per mile per passenger will be : 4.042815100108693 USD/mile

Trips with 3 passenger

Total amount for trips with 3 passenger is: 377650.14

Total distance in miles for trips with 3 passenger is: 45007.09

Hance per mile fare is: 8.390903299902305

Therefore, fare per mile per passenger will be : 2.796967766634102 USD/mile

Trips with 4 passenger

Total amount for trips with 4 passenger is: 204198.07

Total distance in miles for trips with 4 passenger is: 24942.83

Hance per mile fare is: 8.186644017539308

Therefore, fare per mile per passenger will be : 2.046661004384827 USD/mile

Trips with 5 passenger

Total amount for trips with 5 passenger is: 131340.47

Total distance in miles for trips with 5 passenger is: 15248.56

Hance per mile fare is: 8.613303157806376

Therefore, fare per mile per passenger will be : 1.7226606315612751 USD/mile

Trips with 6 passenger

Total amount for trips with 6 passenger is: 84004.17

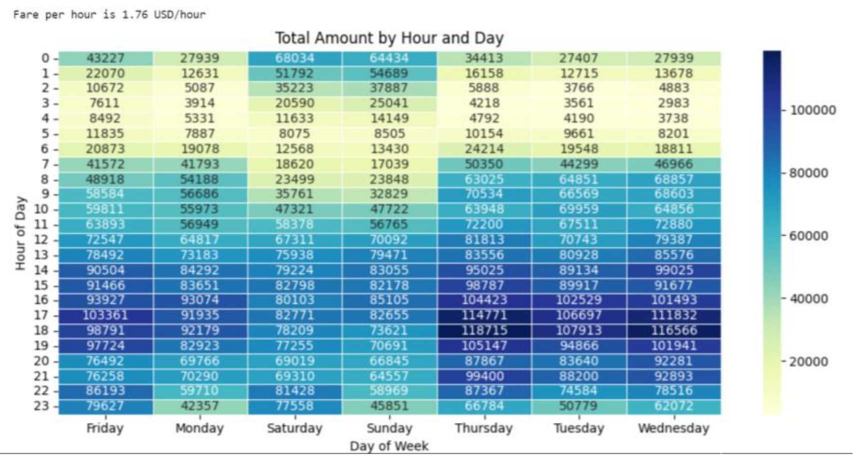
Total distance in miles for trips with 6 passenger is: 9587.18

Hance per mile fare is: 8.762135476751244

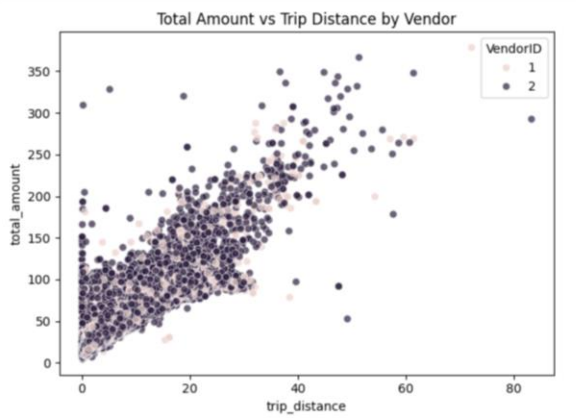
Therefore, fare per mile per passenger will be : 1.460355912791874 USD/mile

Average per passenger per mile fare is: 3.4238626905283263

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

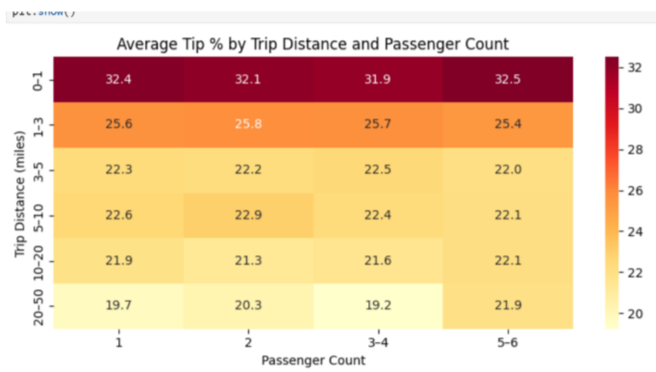


3.2.10. Analyse the average fare per mile for the different vendors

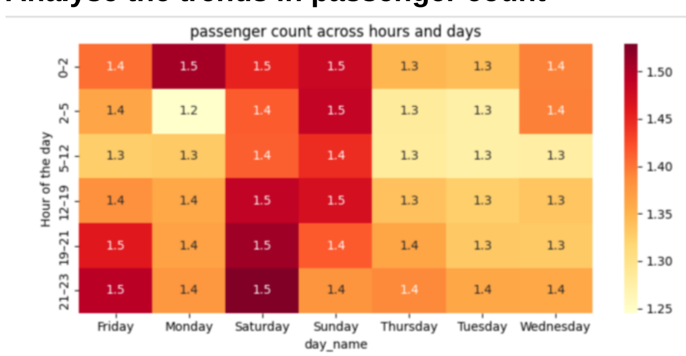


3.2.11. Compare the fare rates of different vendors in a distance-tiered fashion

3.2.12. Analyse the tip percentages



3.2.13. Analyse the trends in passenger count

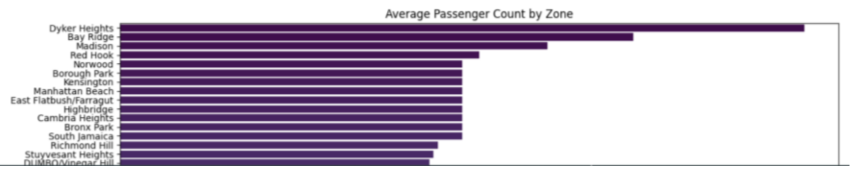


3.2.14. Analyse the variation of passenger counts across zones

Due to the size, whole graph is not inserted here.

```python
# How does passenger count vary across zones
zone_passenger_summary = merged_df.groupby('zone')['passenger_count'].agg(['mean', 'median', 'sum', 'count']).reset_index()
zone_passenger_summary = zone_passenger_summary.sort_values('mean', ascending=False)

plt.figure(figsize=(14, 25))
sns.barplot(data=zone_passenger_summary, x='mean', y='zone', palette='viridis')
plt.xticks(rotation=90)
plt.title('Average Passenger Count by Zone')
plt.ylabel('Zone')
plt.xlabel('Average Passenger Count')
plt.tight_layout()
plt.show()
```



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

<u>improvement_surcharge</u>

count	330953.000000
mean	0.999588
std	0.017172
min	0.000000
25%	1.000000
50%	1.000000
75%	1.000000
max	1.000000

<u>congestion_surcharge</u>

count	330953.000000
mean	2.371087
std	0.552867
min	0.000000
25%	2.500000
50%	2.500000
75%	2.500000

max 2.500000

extra

count	330953.000000
mean	1.643726
std	1.838072
min	0.000000
25%	0.000000
50%	1.000000
75%	2.500000
max	14.250000

4. Conclusions

4.1. Final Insights and Recommendations

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

4.1.2. Suggestions on strategically positioning cabs across different zones to make best use of insights uncovered by analysing trip trends across time, days and months.

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