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

```
Number of rows in the combined DataFrame: 1911120

DataFrame has been downsampled to 300,000 rows.

Number of rows in the downsampled DataFrame: 300000

DataFrame saved to 'sampled_nyc_taxi_data.csv'
```

2. Data Cleaning

2.1. Fixing Columns

2.1.1. Fix the index

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULo
0	2	2023-09-08 12:47:27	2023-09-08 13:11:38	1.0	2.54	1.0	N	
1	2	2023-04-10 09:25:07	2023-04-10 09:48:36	1.0	10.98	1.0	N	
2	2	2023-05-15 18:21:44	2023-05-15 18:31:34	1.0	1.38	1.0	N	
3	2	2023-01-07 22:26:46	2023-01-07 22:36:29	1.0	4.97	1.0	N	

2.1.2. Combine the two airport_fee columns

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULoc
0	2	2023-10-21 15:07:15	2023-10-21 15:13:20	NaN	0.80	NaN	NaN	
1	2	2023-09-08 12:47:27	2023-09-08 13:11:38	1.0	2.54	1.0	N	
2	2	2023-04-10 09:25:07	2023-04-10 09:48:36	1.0	10.98	1.0	N	
3	2	2023-05-15 18:21:44	2023-05-15 18:31:34	1.0	1.38	1.0	N	
4	2	2023-01-07 22:26:46	2023-01-07 22:36:29	1.0	4.97	1.0	N	

5 rows x 21 columns

2.2. Handling Missing Values

2.2.1. Find the proportion of missing values in each column

	0
VendorID	0.000000
tpep_pickup_datetime	0.000000
tpep_dropoff_datetime	0.000000
passenger_count	3.399333
trip_distance	0.000000
RatecodeID	3.399333
store_and_fwd_flag	3.399333
PULocationID	0.000000
DOLocationID	0.000000
payment_type	0.000000
fare_amount	0.000000
extra	0.000000
mta_tax	0.000000
tip_amount	0.000000
tolls_amount	0.000000
improvement_surcharge	0.000000
total_amount	0.000000

2.2.2. Handling missing values in passenger_count

```
        VendorID tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0              2  2023-10-21 15:07:15   2023-10-21 15:13:20              NaN
21             1  2023-10-31 18:11:58   2023-10-31 18:22:17              NaN
43             1  2023-09-15 19:10:20   2023-09-15 19:11:00              NaN
79             1  2023-11-29 07:51:07   2023-11-29 08:08:40              NaN
85             1  2023-01-14 23:53:32   2023-01-15 00:11:22              NaN
...          ...                  ...                   ...              ...
299923         1  2023-11-23 13:42:48   2023-11-23 13:56:07              NaN
299956         2  2023-07-06 04:37:18   2023-07-06 04:53:47              NaN
299959         2  2023-07-10 18:29:11   2023-07-10 18:37:24              NaN
299975         2  2023-10-29 03:36:28   2023-10-29 03:40:50              NaN
299981         2  2023-05-20 19:32:59   2023-05-20 19:45:35              NaN

        trip_distance  RatecodeID store_and_fwd_flag  PULocationID  \
0                0.80         NaN                NaN           137
21               0.90         NaN                NaN           142
43               0.00         NaN                NaN           161
79               0.00         NaN                NaN           142
85               0.00         NaN                NaN           249
...               ...         ...                ...           ...
299923           0.00         NaN                NaN           236
299956           8.18         NaN                NaN           166
299959           1.15         NaN                NaN            68
```

2.2.3. Handle missing values in RatecodeID

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payment_type
0	2	2023-10-21 15:07:15	2023-10-21 15:13:20	1.0	0.80	1.0	NaN	137	233	0
1	2	2023-09-08 12:47:27	2023-09-08 13:11:38	1.0	2.54	1.0	N	113	246	1
2	2	2023-04-10 09:25:07	2023-04-10 09:48:36	1.0	10.98	1.0	N	264	264	1
3	2	2023-05-15 18:21:44	2023-05-15 18:31:34	1.0	1.38	1.0	N	143	48	1
4	2	2023-01-07 22:26:46	2023-01-07 22:36:29	1.0	4.97	1.0	N	132	130	2

5 rows × 21 columns

2.2.4. Impute NaN in congestion_surcharge

```
Column 'store_and_fwd_flag' has missing values.
```

```
Column 'airport_fee' has missing values.
```

2.3. Handling Outliers and Standardising Values

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

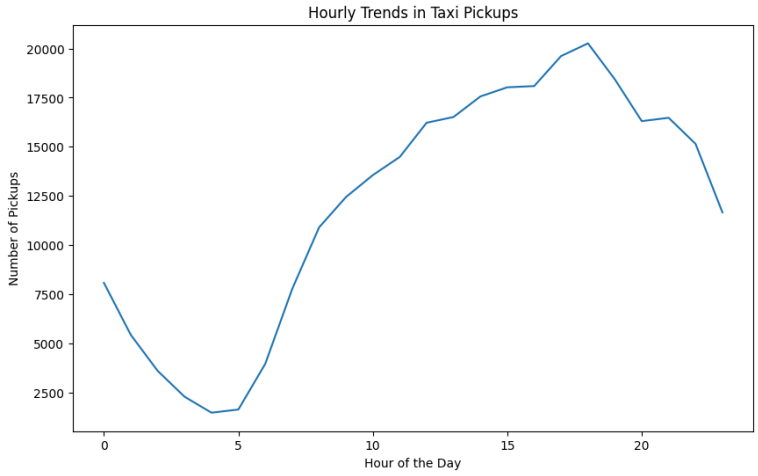
	VendorID	passenger_count	trip_distance	RatecodeID	PULocationID	DOLocationID	payment_type	fare_amount	extra	mta
count	300000.000000	300000.000000	300000.000000	300000.000000	300000.000000	300000.000000	300000.000000	300000.000000	300000.000000	300000.00
mean	1.734427	1.374683	3.631771	1.613237	165.370613	164.080367	1.165023	19.847815	1.585613	0.49
std	0.446368	0.868327	51.039032	7.270209	63.996930	69.827925	0.509377	18.444596	1.826963	0.04
min	1.000000	1.000000	0.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.00
25%	1.000000	1.000000	1.040000	1.000000	132.000000	114.000000	1.000000	9.300000	0.000000	0.50
50%	2.000000	1.000000	1.790000	1.000000	162.000000	162.000000	1.000000	13.500000	1.000000	0.50
75%	2.000000	1.000000	3.400000	1.000000	234.000000	234.000000	1.000000	21.900000	2.500000	0.50
max	6.000000	6.000000	22562.670000	99.000000	265.000000	265.000000	4.000000	1375.000000	14.250000	0.80

3. Exploratory Data Analysis

3.1. General EDA: Finding Patterns and Trends

3.1.1. Classify variables into categorical and numerical

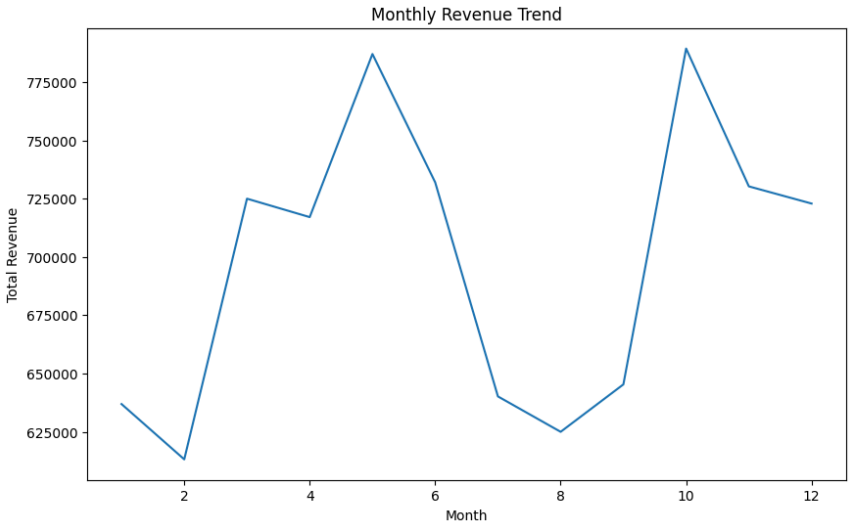
3.1.2. Analyse the distribution of taxi pickups by hours, days of the week, and months



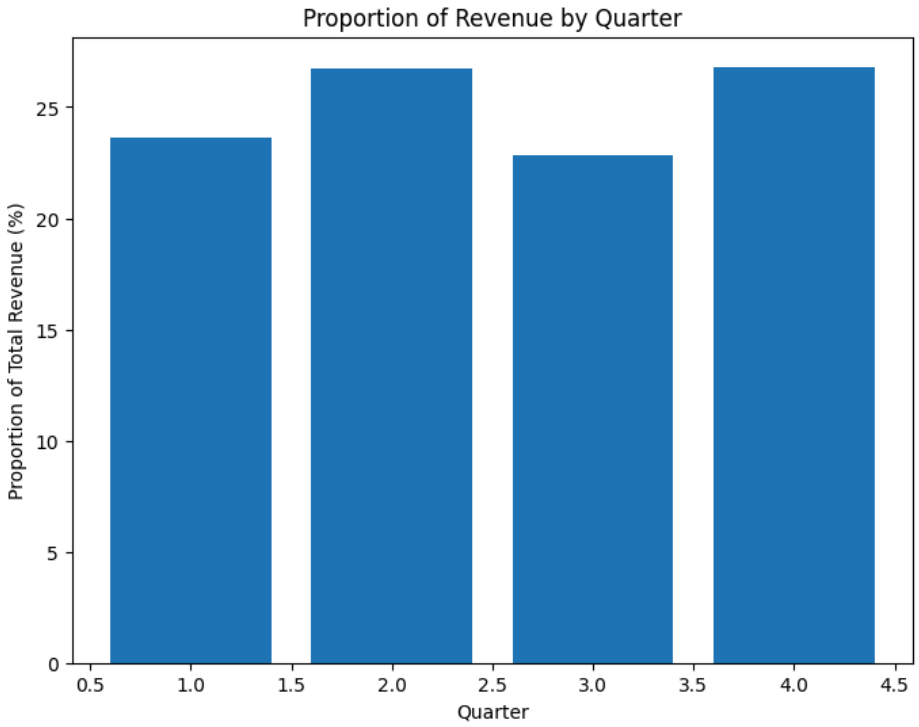
3.1.3. Filter out the zero/negative values in fares, distance and tips

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payment_type	...	tolls_amount	improvement_surcharge	total_amount	congestion_surcharge	airport_fee	date	hour	pickup_hour
1	2	2023-09-08 12:47:27	2023-09-08 13:11:38	1.0	2.54	1.0	N	113	246	1	...	0.00	1.0	30.24	2.5	0.00	2023-09-08	12	12
2	2	2023-04-10 09:25:07	2023-04-10 09:48:36	1.0	10.98	1.0	N	264	264	1	...	6.55	1.0	67.57	2.5	1.75	2023-04-10	9	9
3	2	2023-05-15 18:21:44	2023-05-15 18:31:34	1.0	1.38	1.0	N	143	48	1	...	0.00	1.0	19.90	2.5	0.00	2023-05-15	18	18
5	2	2023-01-14 16:02:47	2023-01-14 16:33:02	1.0	10.60	1.0	N	138	163	1	...	6.55	1.0	74.75	2.5	1.25	2023-01-14	16	16
6	2	2023-03-12 20:39:37	2023-03-12 20:45:52	1.0	0.70	1.0	N	236	141	1	...	0.00	1.0	14.03					

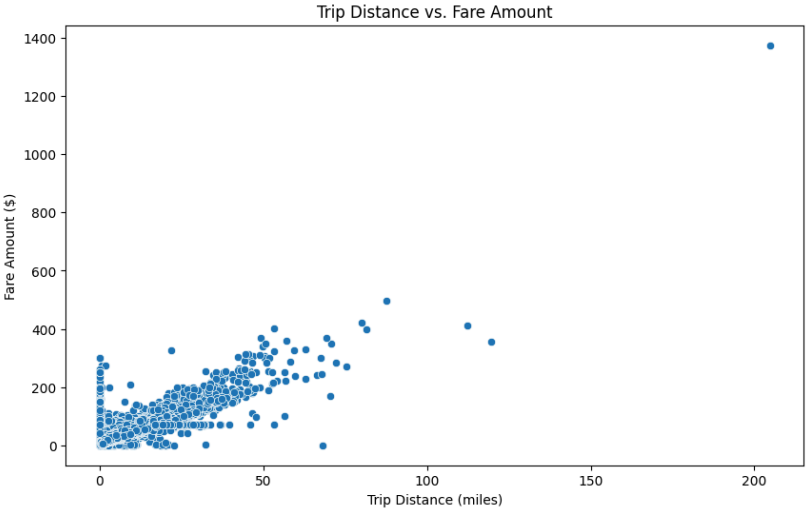
3.1.4. Analyse the monthly revenue trends

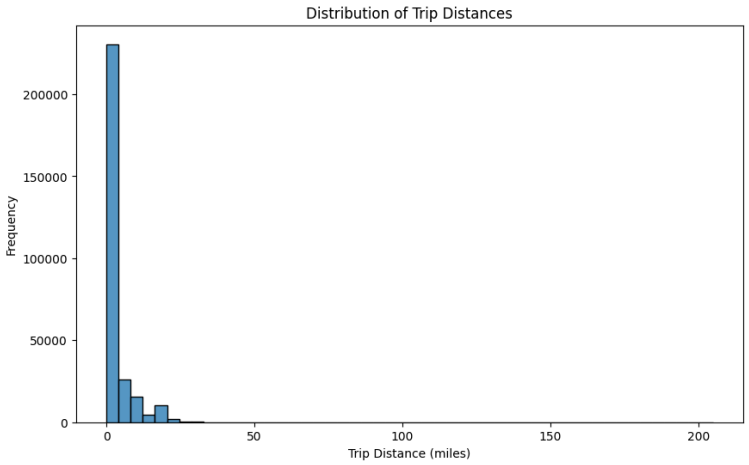


3.1.5. Find the proportion of each quarter's revenue in the yearly revenue

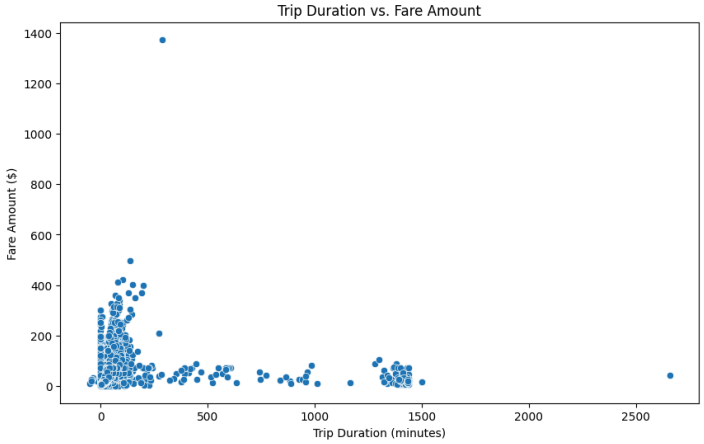


3.1.6. Analyse and visualise the relationship between distance and fare amount

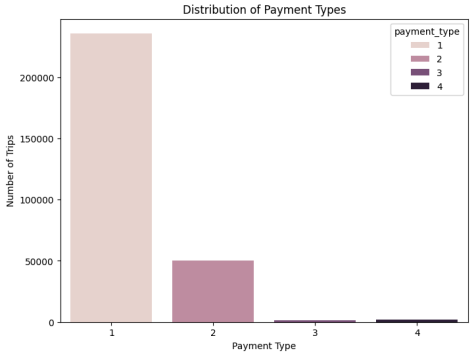




3.1.7. Analyse the relationship between fare/tips and trips/passengers



3.1.8. Analyse the distribution of different payment types



3.1.9. Load the taxi zones shapefile and display it

	OBJECTID	Shape_Leng	Shape_Area	zone	LocationID	borough	
0	1	0.116357	0.000782	Newark Airport	1	EWR	POLYGON ((933100.918 192536.086, 93309
1	2	0.433470	0.004866	Jamaica Bay	2	Queens	MULTIPOLYGON (((1033269.244 172126.008
2	3	0.084341	0.000314	Allerton/Pelham Gardens	3	Bronx	POLYGON ((1026308.77 256767.698, 10264
3	4	0.043567	0.000112	Alphabet City	4	Manhattan	POLYGON ((992073.467 203714.076, 99206
4	5	0.092146	0.000498	Arden Heights	5	Staten Island	POLYGON ((935843.31 144283.336, 936046

3.1.10. Merge the zone data with trips data

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payment_type
0	2	2023-09-08 12:47:27	2023-09-08 13:11:38	1.0	2.54	1.0	N	113	246	1
1	2	2023-04-10 09:25:07	2023-04-10 09:48:36	1.0	10.98	1.0	N	264	264	1
2	2	2023-05-15 18:21:44	2023-05-15 18:31:34	1.0	1.38	1.0	N	143	48	1
3	2	2023-01-07 22:26:46	2023-01-07 22:36:29	1.0	4.97	1.0	N	132	130	2
4	2	2023-01-14 16:02:47	2023-01-14 16:33:02	1.0	10.60	1.0	N	138	163	1

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

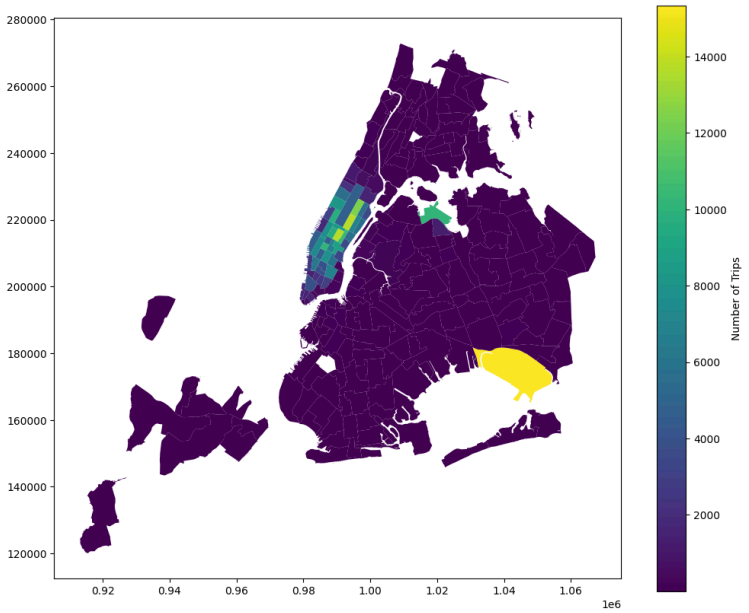
PULocationID	PULocationID
1	27
3	5
4	324
6	5
7	135
...	...
261	1526
262	3674
263	5485
264	2761
265	172

241 rows × 1 columns

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

	OBJECTID	Shape_Leng	Shape_Area	zone	LocationID	borough	geometry	trip_count
0	1	0.116357	0.000782	Newark Airport	1	EWR	POLYGON ((933100.918 192536.086, 933091.011 19...	27
2	3	0.084341	0.000314	Allerton/Pelham Gardens	3	Bronx	POLYGON ((1026308.77 256767.698, 1026495.593 2...	5
3	4	0.043567	0.000112	Alphabet City	4	Manhattan	POLYGON ((992073.467 203714.076, 992068.667 20...	324
5	6	0.150491	0.000606	Arrochar/Fort Wadsworth	6	Staten Island	POLYGON ((966568.747 158679.855, 966615.256 15...	5
6	7	0.107417	0.000390	Astoria	7	Queens	POLYGON ((1010804.218 218919.641, 1011049.165 ...	135

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



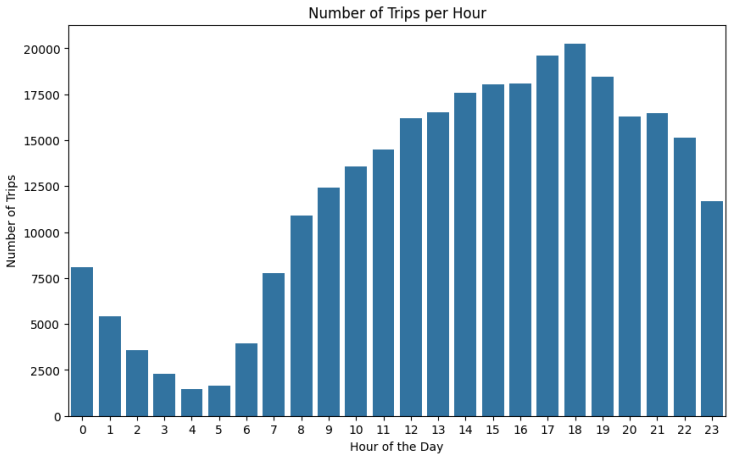
3.1.14. Conclude with results

3.2. Detailed EDA: Insights and Strategies

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

	pickup_hour	route	speed
130	0	162_162	0.000000
32931	1	107_263	-1.478981
2884	2	48_48	0.000000
3994	3	136_136	0.000000
10712	4	132_132	0.000000
369	5	50_50	0.000000
4453	6	140_140	0.000000
478	7	132_162	0.000000
449	8	13_13	0.000000
2967	9	7_193	0.000000
1611	10	132_223	0.000000
170	11	43_43	0.000000
2774	12	145_145	0.000000
382	13	193_193	0.000000
192	14	236_239	0.000000
985	15	263_263	0.000000
4248	16	145_145	0.000000
2933	17	148_148	0.000000
741	18	161_230	0.000000
915	19	143_142	0.000000
155	20	132_132	0.000000
327	21	113_113	0.000000
1311	22	161_264	0.000000
2577	23	265_265	0.000000

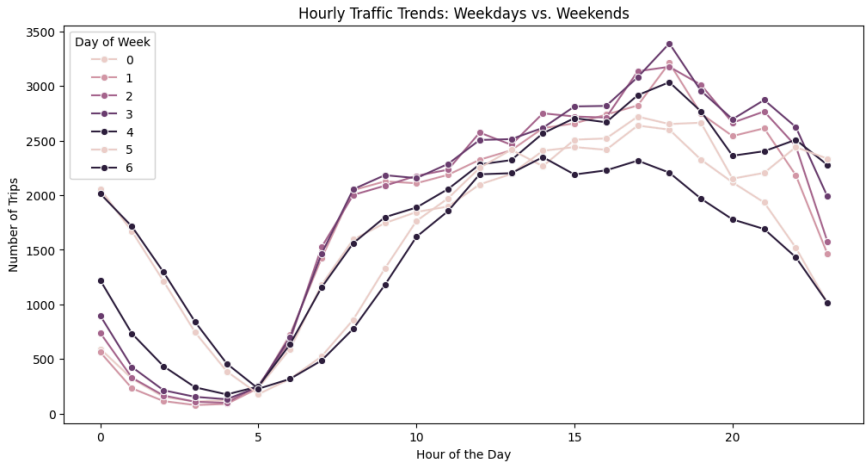
3.2.2. Calculate the hourly number of trips and identify the busy hours



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

```
Actual number of trips in the five busiest hours:
pickup_hour
18    202650.0
17    196220.0
19    184340.0
16    180920.0
15    180310.0
Name: pickup_hour, dtype: float64
```

3.2.4. Compare hourly traffic on weekdays and weekends



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

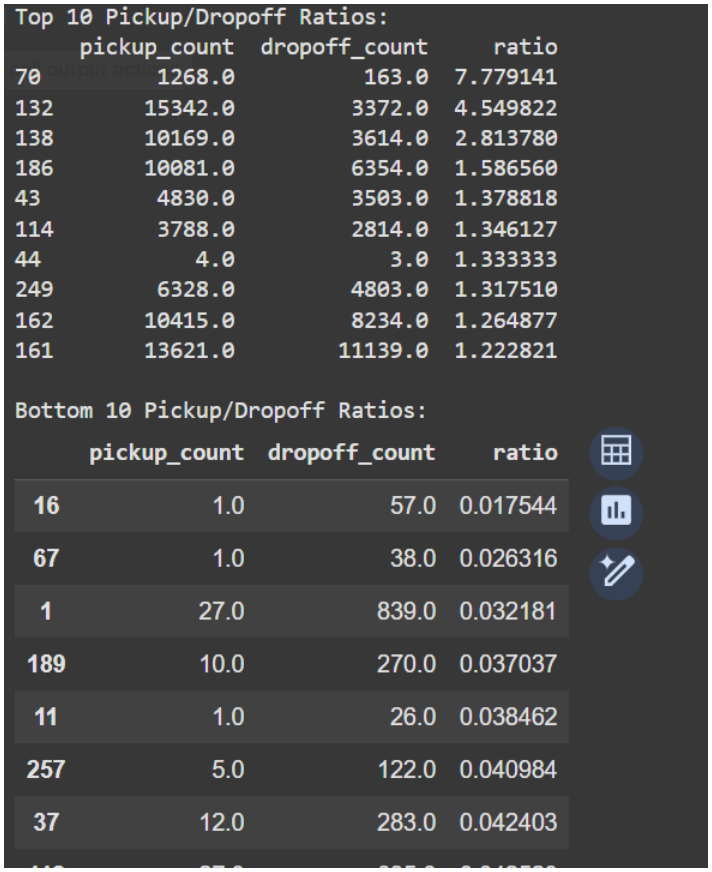
```
Top 10 Pickup Zones:
```

```
Index([132, 237, 161, 236, 162, 138, 186, 142, 230, 170], dtype='int64',
name='PULocationID')
```

```
Top 10 Dropoff Zones:
```

```
Index([236, 237, 161, 230, 170, 162, 239, 142, 141, 68], dtype='int64',
name='DOLocationID')
```

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



Top 10 Pickup/Dropoff Ratios:

	pickup_count	dropoff_count	ratio
70	1268.0	163.0	7.779141
132	15342.0	3372.0	4.549822
138	10169.0	3614.0	2.813780
186	10081.0	6354.0	1.586560
43	4830.0	3503.0	1.378818
114	3788.0	2814.0	1.346127
44	4.0	3.0	1.333333
249	6328.0	4803.0	1.317510
162	10415.0	8234.0	1.264877
161	13621.0	11139.0	1.222821

Bottom 10 Pickup/Dropoff Ratios:

	pickup_count	dropoff_count	ratio
16	1.0	57.0	0.017544
67	1.0	38.0	0.026316
1	27.0	839.0	0.032181
189	10.0	270.0	0.037037
11	1.0	26.0	0.038462
257	5.0	122.0	0.040984
37	12.0	283.0	0.042403

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

```
Top 10 Pickup Zones (Night Hours):
PULocationID
79     2445
132    2254
249    1941
48     1641
148    1553
114    1390
230    1247
186    1075
164     946
68      925
Name: count, dtype: int64

Top 10 Dropoff Zones (Night Hours):
```

DOLocationID	count
79	1293
48	1086
170	951
68	944

3.2.8. Find the revenue share for nighttime and daytime hours

```
Nighttime Revenue Share: 11.99%

Daytime Revenue Share: 88.01%
```

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

passenger_count	
1.0	10.718841
2.0	6.076498
3.0	3.114451
4.0	3.552935
5.0	1.808098

```
6.0      1.304975

dtype: float64
```

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

	day_of_week	hour_of_day	fare_per_mile
0	0	0	18.031211
1	0	1	5.971719
2	0	2	5.968780
3	0	3	43.354939
4	0	4	5.760712
...	...	...	...
163	6	19	13.943246
164	6	20	15.164402
165	6	21	14.525767
166	6	22	24.918348
167	6	23	13.488585

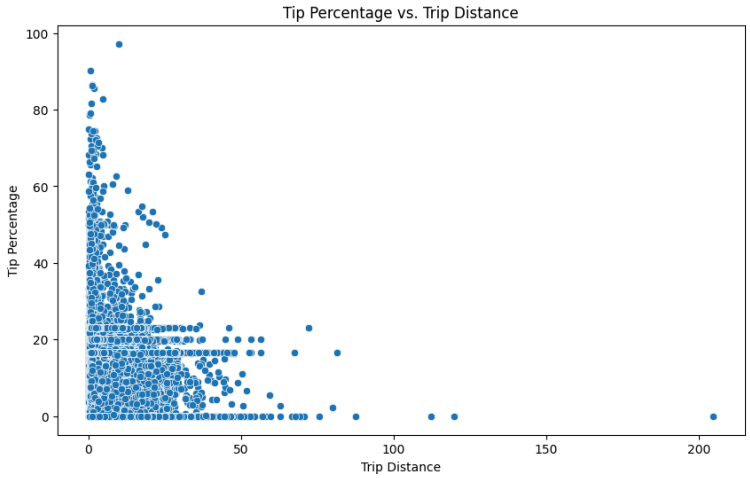
3.2.11. Analyse the average fare per mile for the different vendors

	VendorID	hour_of_day	fare_per_mile
0	1	0	6.748959
1	1	1	6.547865
2	1	2	6.688823
3	1	3	6.691887
4	1	4	7.302470
5	1	5	8.397071
6	1	6	6.623016
7	1	7	7.096034
8	1	8	8.044384

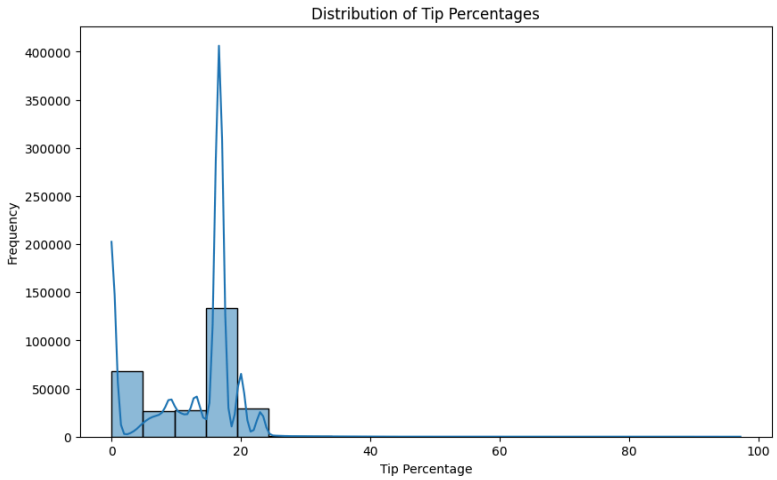
3.2.12. Compare the fare rates of different vendors in a distance-tiered fashion

	VendorID	trip_distance	fare_per_mile
0	1	0-2 miles	9.980938
1	1	2-5 miles	6.374158
2	1	5+ miles	4.421025
3	2	0-2 miles	16.987121
4	2	2-5 miles	6.539500
5	2	5+ miles	4.502676

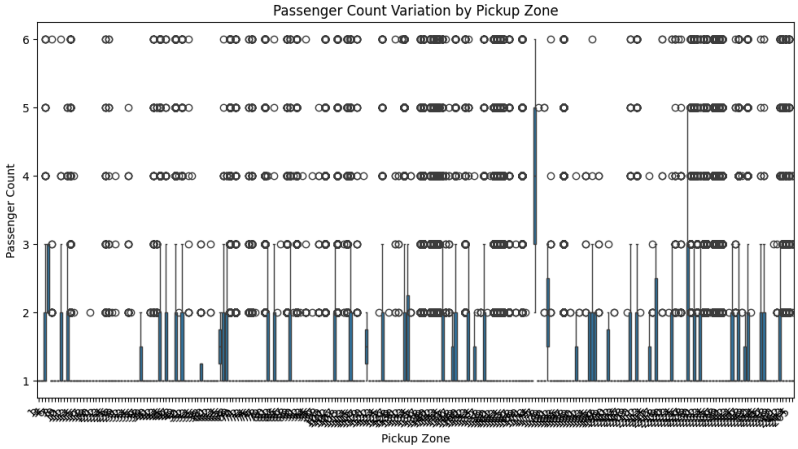
3.2.13. Analyse the tip percentages



3.2.14. Analyse the trends in passenger count



3.2.15. Analyse the variation of passenger counts across zones



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

```
Frequency of extra:
extra
0.00     108073
2.50      72868
1.00      56923
5.00      21138
3.50      16803
7.50       2561
6.00       2435
4.25       1007
9.25        968
1.75        464
3.75        392
6.75        380
2.75        334
8.75        322
10.25       275
7.75        237
1.25        216
11.75       191
2.25        137
6.25        125
10.00        97
9.75         81
11.25        60
```

4. Conclusions

4.1. Final Insights and Recommendations

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

```
# Recommendations to optimize routing and dispatching

# 1. Dynamic Pricing: Implement surge pricing during peak hours and in high-demand zones.
#    - Identify peak hours and high-demand zones using the hourly pickup/dropoff trends and top zones analysis.
#    - Adjust prices based on the ratio of pickups/dropoffs to optimize supply and demand.
#    - Consider time-of-day and day-of-week variations in demand when adjusting prices.

# 2. Optimized Vehicle Dispatching:
#    - Allocate more vehicles to high-demand zones during peak hours.
#    - Use real-time data to predict demand and proactively adjust vehicle distribution.
```

```
#    - Prioritize dispatching to areas with high pickup/dropoff ratios.
#    - Consider using predictive models to forecast demand and optimize
vehicle allocation in advance.

# 3. Route Optimization:
#    - Use algorithms to determine the most efficient routes for drivers
based on current traffic, demand, and pickup/dropoff locations.
#    - Consider incorporating real-time traffic information into routing
algorithms.
#    - Factor in the average trip distance and fare rates for different
vendors and distance tiers when calculating optimal routes.

# 4. Nighttime Operations:
#    - Increase the number of vehicles operating during night hours
(11PM-5AM) in zones with high night time demand.
#    - Consider offering different pricing strategies for night trips
based on revenue shares.

# 5. Consider passenger counts:
#    - Adjust pricing based on the average fare per mile per passenger for
different passenger counts.

# 6. Improve Customer Experience:
#    - Analyze customer tipping behaviors and identify factors that
influence tip percentages.
#    - Investigate low tip areas (distance, passenger count, time of day)
to understand and improve these aspects.
#    - Consider rewarding drivers with high tip percentages or good
customer service ratings.


# 7. Address Surcharges:
#    - Analyze the frequency and locations of extra charges to understand
their prevalence.
#    - Investigate zones and times with high surcharge applications to
understand root causes and explore potential solutions.

# Example:
# During weekdays (Monday-Friday) between 5 pm and 8 pm, increase the
number of available vehicles in zones 7, 231, 4, and 263.
```

```
# Apply a 20% surcharge during peak hours in these zones.
# During the early morning and late-night hours, reduce the vehicle supply
in areas of low demand and reposition to high demand zones
```

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.

```
# Position more cabs in high-demand zones during peak hours, as identified
by analyzing pickup/dropoff ratios across different times, days, and
months.  This ensures sufficient supply to meet increased demand.
```

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

```
# Data-Driven Pricing Adjustments

# 1. Time-Based Adjustments:
#    - Implement dynamic pricing based on hourly and daily demand
fluctuations.
#    - Offer lower fares during off-peak hours and days to attract
price-sensitive customers and fill empty vehicles.
#    - Consider a tiered pricing strategy for different time blocks (e.g.,
rush hour, daytime, nighttime).  Refer to 3.2.8 for revenue shares.

# 2. Location-Based Adjustments:
#    - Consider the average fare per mile in different zones (identified
in 3.2.15) to establish competitive pricing.
#    - Implement zone-specific surge pricing during peak hours or special
events.

# 3. Distance-Based Adjustments:
#    - Consider a minimum fare for short trips and gradually increase the
price per mile for longer journeys.
```

```
# 4. Passenger Count Adjustments:
#    - Adjust pricing based on the average fare per mile per passenger for
different passenger counts (3.2.9).
#    - This will ensure fair pricing for shared rides and incentivize
riders to opt for shared trips where possible.

# 5. Vendor Competitiveness:
#    - Adjust prices to undercut competitors in specific zones or times
when possible while still maintaining profitability.

# 6. Surcharge Optimization:
#    - Evaluate the impact of existing surcharges on revenue and customer
satisfaction.
#    - Re-evaluate or refine surcharge strategies based on this analysis.

# 7. Customer Behavior (Tipping):
#   - While not directly impacting pricing, understanding the factors
influencing tips (trip distance, passenger count, time of pickup) can
indirectly impact pricing.
#   - For example, offering slightly lower fares to incentivize higher
tips could be considered, or offering higher base fares for long trips.


# Implementation:
# - A/B testing: Implement changes gradually and test their impact on
revenue, customer behavior, and market share.
# - Data Monitoring: Continuously monitor key metrics (revenue, ride
volume, customer satisfaction) to measure the success of pricing
adjustments.
# - Machine Learning: Utilize machine learning techniques to predict
demand and dynamically adjust pricing in real-time.
```