

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

Sampling strategy: As asked in assignment to keep the total entries to around 250,000 to 300,000, I have decided to set sample fraction as 0.008% (as 5% of data was too high which was suggested in starter file) per date-hour from each month's parquet file.

In this methods, it reads each monthly parquet file one by one, and for each date in the month it iterates over 24 hours and randomly samples a fraction of trips for that date-hour.

As part of sampling, it also **derives new columns pickup_date and pickup_hour** from pickup datetime.

All sampled rows are concatenated into a single **nyc_taxi_2023_sampled.parquet** which represents the full year's sampled data.

Final sampled file contains **303838 rows and 22 columns**.

```
Final sampled yearly shape: (303838, 22)
<class 'pandas.core.frame.DataFrame'
RangeIndex: 303838 entries, 0 to 303837
Data columns (total 22 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   VendorID          303838 non-null  int64  
 1   tpep_pickup_datetime 303838 non-null  datetime64[us]
 2   tpep_dropoff_datetime 303838 non-null  datetime64[us]
 3   passenger_count     293620 non-null  float64 
 4   trip_distance       303838 non-null  float64 
 5   RatecodeID          293620 non-null  float64 
 6   store_and_fwd_flag   293620 non-null  object  
 7   PULocationID        303838 non-null  int64  
 8   DOLocationID        303838 non-null  int64  
 9   payment_type         303838 non-null  int64  
 10  fare_amount          303838 non-null  float64 
 11  extra                303838 non-null  float64 
 12  mta_tax               303838 non-null  float64 
 13  tip_amount             303838 non-null  float64 
 14  tolls_amount            303838 non-null  float64 
 15  improvement_surcharge 303838 non-null  float64 
 16  total_amount            303838 non-null  float64 
 17  congestion_surcharge   293620 non-null  float64 
 18  airport_fee              23775 non-null  float64 
 19  pickup_date              303838 non-null  object  
 20  pickup_hour              303838 non-null  int32  
 21  Airport_fee               269845 non-null  float64 
dtypes: datetime64[us](2), float64(13), int32(1), int64(4), object(2)
memory usage: 49.8+ MB
```

After combining the data files into one DataFrame, also converted the new DataFrame to a CSV file and store it as **nyc_taxi_2023_sampled.csv** file to use

directly in further data operations.

Load sampled file nyc_taxi_2023_sampled.parquet

VendorID	tpg.pickup_datetime	tpg.dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PUlocationID	DOlocationID	payment_type	...	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount	
0	2	2023-01-01 00:07:18	2023-01-01 00:23:15	1.0	7.74	1.0	N	138	256	2	...	0.5	0.00	0.0	1.0	41.15
1	2	2023-01-01 00:16:41	2023-01-01 00:21:46	2.0	1.24	1.0	N	161	237	1	...	0.5	2.58	0.0	1.0	15.48
2	2	2023-01-01 00:14:03	2023-01-01 00:24:36	3.0	1.44	1.0	N	237	141	2	...	0.5	0.00	0.0	1.0	16.40
3	2	2023-01-01 00:24:30	2023-01-01 00:29:55	1.0	0.54	1.0	N	143	142	2	...	0.5	0.00	0.0	1.0	11.50
4	2	2023-01-01 00:04:00	2023-01-01 01:01:00	NaN	19.24	NaN	None	66	107	0	...	0.5	5.93	0.0	1.0	35.57

2. Data Cleaning

2.1. Fixing Columns

2.1.1. Fix the index

Fix the index and drop any column that are not needed

Reset index on a working copy of DataFrame (df).

NOTE:'pickup_date', 'pickup_hour' are derived columns which I will use in further assignment so not dropping them here.

Starting shape (df): (303838, 22)

After reset_index (df_clean): (303838, 22)

2.1.2. Combine the two airport_fee columns

Create unified column taking first non-null value across variants, fill remaining NaN with 0

Drop original duplicates except the unified one

Shape after combining columns: (303838, 21)

2.1.3. Fix columns with negative (monetary) values

check where values of fare amount are negative

Total rows: 303,838

Negative fare rows: 0

No negative fare_amount values found.

Did you notice something different in the `RatecodeID` column for above records?

Analyse RatecodeID for the negative fare amounts

Negative fare rows: 0

No negative fare_amount rows found.

Find which columns have negative values

Columns with negative values (top rows):						
	column	neg_count	neg_pct	min_neg	max_neg	distinct_neg_values_sample
1	mta_tax	11	0.003620	-0.50	-0.50	[-0.5]
2	improvement_surcharge	11	0.003620	-1.00	-1.00	[-1.0]
3	total_amount	11	0.003620	-4.00	-1.50	[-4.0, -3.25, -1.5]
4	congestion_surcharge	6	0.001975	-2.50	-2.50	[-2.5]
5	airport_fee	2	0.000658	-1.75	-1.75	[-1.75]
0	extra	1	0.000329	-2.50	-2.50	[-2.5]

Fix these negative values

For Columns mta_tax, improvement_surcharge, total_amount, congestion_surcharge, extra replace negative values with 0

```
Before df shape: (303838, 21)
After df shape: (303838, 21)
```

2.2. Handling Missing Values

2.2.1. Find the proportion of missing values in each column

proportion of missing values in each column:	
	missing_pct
passenger_count	3.363
RatecodeID	3.363
store_and_fwd_flag	3.363
congestion_surcharge	3.363
VendorID	0.000
mta_tax	0.000
pickup_date	0.000
airport_fee	0.000
total_amount	0.000
improvement_surcharge	0.000
tolls_amount	0.000
tip_amount	0.000
fare_amount	0.000
extra	0.000
tpep_pickup_datetime	0.000
payment_type	0.000
DOLocationID	0.000
PULocationID	0.000
trip_distance	0.000
tpep_dropoff_datetime	0.000
pickup_hour	0.000

2.2.2. Handling missing values in passenger_count

Impute NaN values in 'passenger_count' **with mode** (most frequent value).

The variable passenger_count represents a discrete count of passengers in a taxi trip and typically takes small integer values (e.g., 1–6). Because this feature is categorical–discrete in nature, using the mode is the most appropriate imputation strategy.

2.2.3. Handle missing values in RatecodeID

Impute NaN values in 'RatecodeID' **with mode** (most frequent value)

The variable RatecodeID is categorical data so mode is appropriate imputation strategy.

2.2.4. Impute NaN in congestion_surcharge

Impute NaN values in 'congestion_surcharge' **with median**

Median avoids overestimating or underestimating charges, which could happen with mean imputation in skewed distributions.

It preserves the central tendency of the surcharge without being influenced by extreme or incorrect values.

Handle any remaining missing values

Found only 1 column with missing value

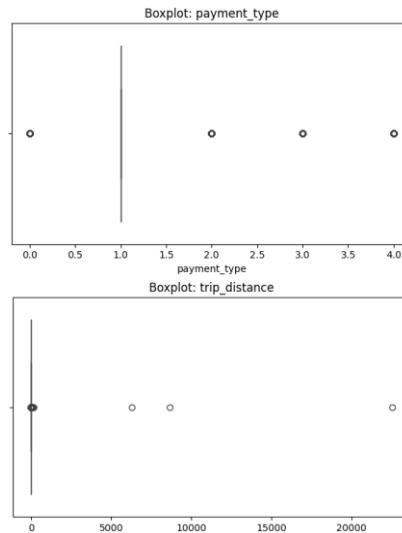
column	dtype	pct_missing
6 store_and_fwd_flag	object	3.363

Impute missing values in 'store_and_fwd_flag' with **mode** (most frequent value)

2.3. Handling Outliers and Standardising Values

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

Used IQR method: detect values outside [Q1 - 1.5*IQR, Q3 + 1.5*IQR]



Fix outliers

Removed 5 rows for: passenger_count > 6

Passenger count more than 6 is treated as invalid.

Removed 5 rows for: passenger_count > 6											
VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID
0	2	2023-11-30 00:13:36	2023-11-30 00:13:39	8.0	0.00	5.0	N	90	264		
1	2	2023-02-19 17:19:13	2023-02-19 17:57:24	9.0	16.79	5.0	N	186	1		
2	2	2023-04-09 09:22:54	2023-04-09 09:23:22	7.0	0.00	5.0	N	125	125		
3	2	2023-05-29 02:35:04	2023-05-29 02:35:16	7.0	0.00	5.0	N	256	256		
4	2	2023-09-18 13:07:26	2023-09-18 14:05:27	8.0	31.71	5.0	N	48	219		

Removed 6 rows for: trip_distance<=0.1 & fare_amount>300

Trip distance is <=0.1 and fare amount is > \$300 is treated as invalid.

Removed 7 rows for: trip_distance==0 & fare_amount==0 & different_zones

Trip distance is 0 and fare amount is also 0 but pickup and drop locations are

different treated as invalid.

Removed 3 rows for: trip_distance>250

Trip distance > 250 is treated as invalid.

Set payment_type==0 to 5 (Unknown) for 10215 rows

As payment type 0 is invalid but rows count is big, considered imputing with payment_type = 5 which is to represent Unknown payment type.

Tip Amount: tip_amount can be zero so no filter applied.

Do any columns need standardising?

Numeric coercion for **passenger_count**, make datatype **from float64 to Int32**.

VendorID, RatecodeID, payment_type can be treated as **categorical** value rather than numeric so converted accordingly.

Drop rows with invalid **VendorID == 6**.

```
<class 'pandas.core.frame.DataFrame'>
Index: 303743 entries, 0 to 303837
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype  
 --- 
 0   VendorID        303743 non-null  object  
 1   tpep_pickup_datetime 303743 non-null  datetime64[us]
 2   tpep_dropoff_datetime 303743 non-null  datetime64[us]
 3   passenger_count    303743 non-null  int32  
 4   trip_distance      303743 non-null  float64 
 5   RatecodeID         303743 non-null  object  
 6   store_and_fwd_flag 303743 non-null  object  
 7   PULocationID      303743 non-null  int64  
 8   DOLocationID      303743 non-null  int64  
 9   payment_type       303743 non-null  object  
 10  fare_amount        303743 non-null  float64 
 11  extra              303743 non-null  float64 
 12  mta_tax             303743 non-null  float64 
 13  tip_amount          303743 non-null  float64 
 14  tolls_amount        303743 non-null  float64 
 15  improvement_surcharge 303743 non-null  float64 
 16  total_amount        303743 non-null  float64 
 17  congestion_surcharge 303743 non-null  float64 
 18  airport_fee         303743 non-null  float64 
 19  pickup_date         303743 non-null  datetime64[ns]
 20  pickup_hour         303743 non-null  int32  
dtypes: Int32(1), datetime64[ns](1), datetime64[us](2), float64(10), int32(1), int64(2), object(4)
memory usage: 49.0e+07
```

3. Exploratory Data Analysis

3.1. General EDA: Finding Patterns and Trends

3.1.1. Classify variables into categorical and numerical

Categorise the variables into Numerical or Categorical.

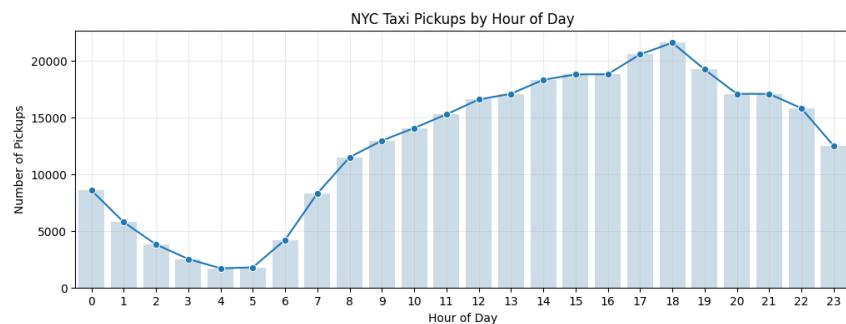
- `VendorID`: category
- `tpep_pickup_datetime`: datetime
- `tpep_dropoff_datetime`: datetime
- `passenger_count`: Numerical
- `trip_distance`: Numerical
- `RatecodeID`: category
- `PULocationID`: category
- `DOLocationID`: category
- `payment_type`: category
- `pickup_hour`: Numerical
- `trip_duration`: Numerical

The following monetary parameters belong in the same category, is it categorical or numerical?

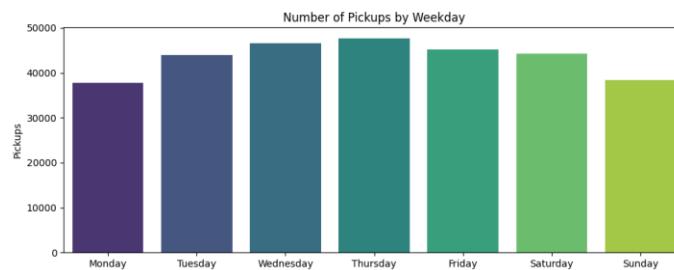
- `fare_amount` : Numerical
- `extra` : Numerical
- `mta_tax` : Numerical
- `tip_amount` : Numerical
- `tolls_amount` : Numerical
- `improvement_surcharge` : Numerical
- `total_amount` : Numerical
- `congestion_surcharge` : Numerical
- `airport_fee` : Numerical

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

Find and show the hourly trends in taxi pickups



Find and show the daily trends in taxi pickups (days of the week)



Show the monthly trends in pickups



Financial Analysis

Take a look at the financial parameters like `fare_amount`, `tip_amount`, `total_amount`, and also `trip_distance`. Do these contain zero/negative values?

Yes, found below listed zero/negative values.

shape=(303743, 24)		
	Zero Count	Negative Count
column		
trip_distance	5996	0
fare_amount	100	0
tip_amount	69750	0
total_amount	50	0

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

Create a df with non zero entries for the selected parameters

Selected 'fare_amount', 'total_amount', 'trip_distance' columns for analysis.

The distance might be 0 in cases where pickup and drop is in the same zone, so only considered **trip_distance zero where pickup and drop locations are different**.

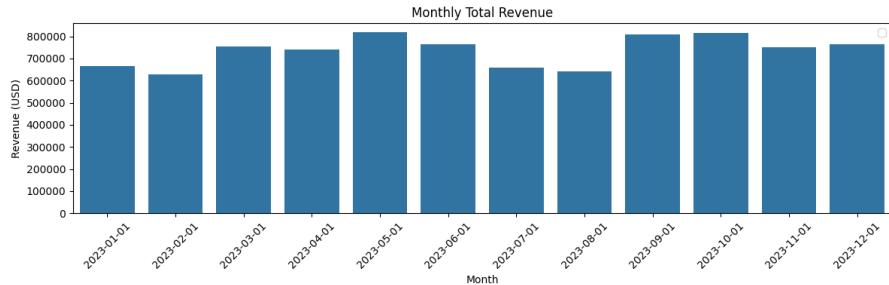
Found below listed columns with zero values which are dropped.

Rows with fare_amount == 0: **100**

Rows with total_amount == 0: **50**

Rows with trip_distance == 0: **3484**

3.1.4. Analyse the monthly revenue trends

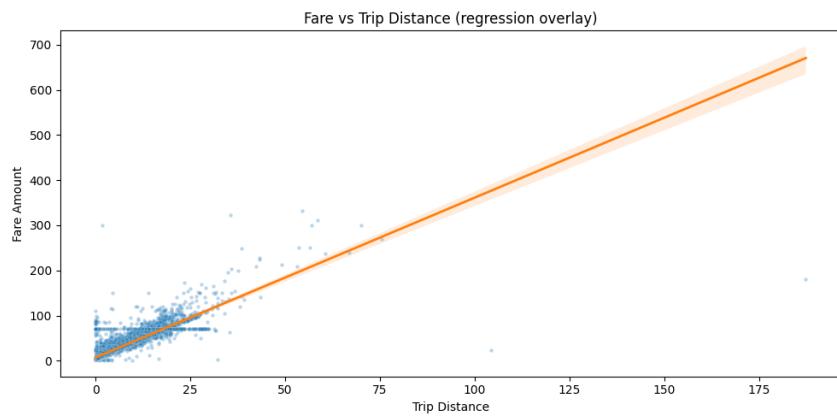


3.1.5. Find the proportion of each quarter's revenue in the yearly revenue

Overall quarterly revenue totals and proportion (% of total revenue):		
quarter	revenue_total	pct_of_total
0	2048394.50	23.25
1	2324558.24	26.39
2	2106950.85	23.92
3	2328830.56	26.44

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

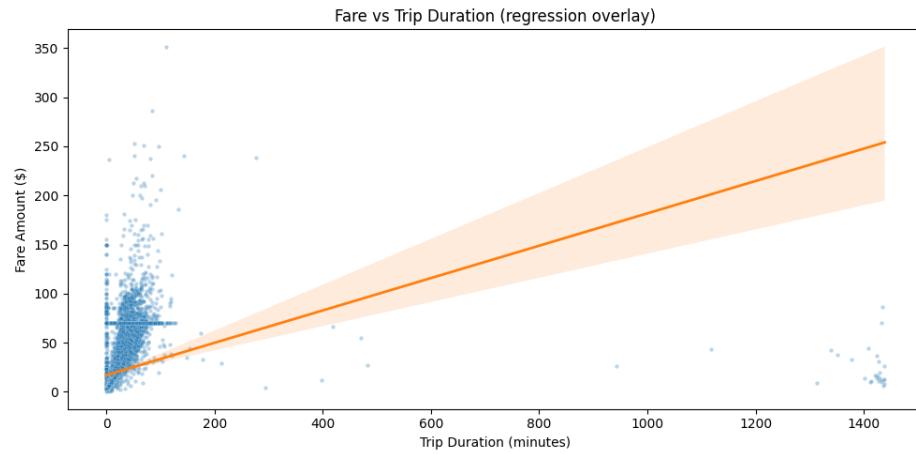
Spearman correlation = 0.9223



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

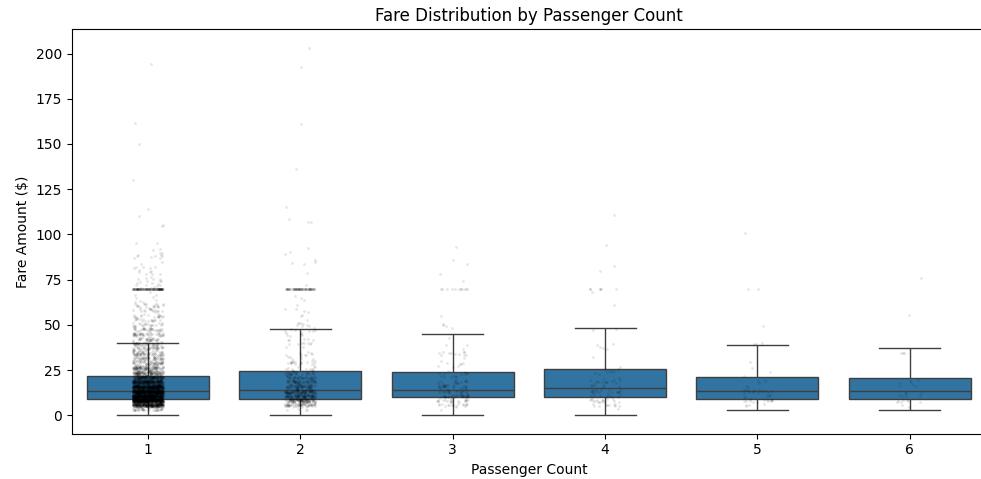
Show relationship between fare and trip duration

Spearman correlation = 0.9363



Show relationship between fare and number of passengers

Fare summary by passenger_count:				
	passenger_count	trips	avg_fare	median_fare
0	1	228093	19.79	13.5
1	2	44045	22.26	14.2
2	3	10992	21.98	14.2
3	4	6052	23.26	14.9
4	5	3768	18.85	13.5
5	6	2567	18.98	13.5

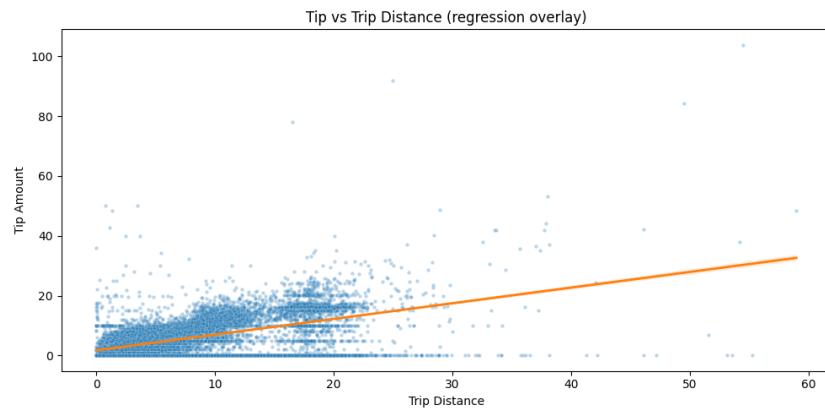


Show relationship between tip and trip distance

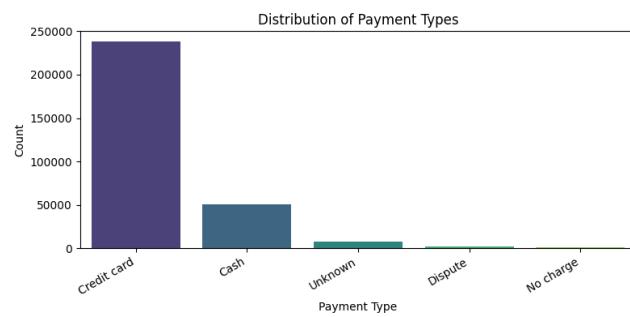
Pearson $r = 0.5882$

Spearman $r = 0.4314$

Longer trips tend to receive higher tips, but the relationship is **moderate rather than strong**, so both Pearson and Spearman correlations confirm a **positive association**



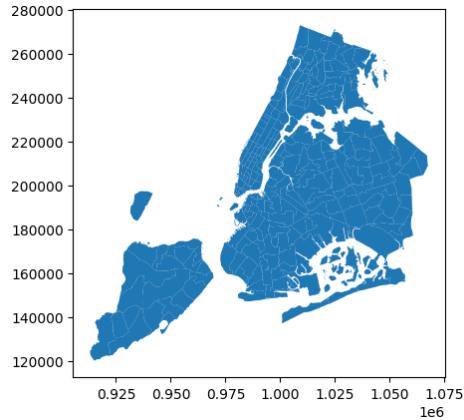
3.1.8. Analyse the distribution of different payment types



3.1.9. Load the taxi zones shapefile and display it

	OBJECTID	Shape_Leng	Shape_Area	zone	LocationID	borough	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...

```
<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
```



3.1.10. Merge the zone data with trips data

Merged zones and trip records using locationID and PULocationID.

	PULocationID	LocationID	zone	borough
0	138	138.0	LaGuardia Airport	Queens
1	161	161.0	Midtown Center	Manhattan
2	237	237.0	Upper East Side South	Manhattan
3	143	143.0	Lincoln Square West	Manhattan
4	66	66.0	DUMBO/Vinegar Hill	Brooklyn

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

Group data by location and calculate the number of trips.

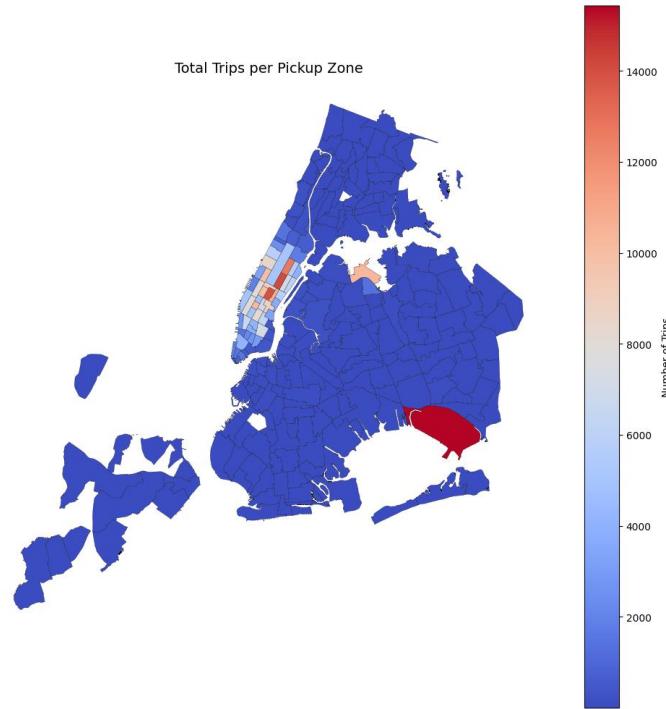
	PULocationID	num_trips
0	1	42
1	3	9
2	4	338
3	5	2
4	6	3

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

Merge trip counts back to the zones GeoDataFrame

	trip_count_pickup	borough	LocationID	zone
131	15434.0	Queens	132	JFK Airport
236	14063.0	Manhattan	237	Upper East Side South
160	13906.0	Manhattan	161	Midtown Center
235	12663.0	Manhattan	236	Upper East Side North
161	10759.0	Manhattan	162	Midtown East
137	10209.0	Queens	138	LaGuardia Airport
185	10140.0	Manhattan	186	Penn Station/Madison Sq West
229	9910.0	Manhattan	230	Times Sq/Theatre District
141	9845.0	Manhattan	142	Lincoln Square East
169	8824.0	Manhattan	170	Murray Hill

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



Can you try displaying the zones DF sorted by the number of trips?

OBJECTID	Shape_Leng	Shape_Area	zone	LocationID	borough	geometry	trip_count_pickup	
131	132	0.245479	0.002038	JFK Airport	132	Queens	MULTIPOLYGON (((1032791.001 181085.006, 103283...	15434.0
236	237	0.042213	0.000096	Upper East Side South	237	Manhattan	POLYGON ((993633.442 216961.016, 993507.232 21...	14063.0
160	161	0.035804	0.000072	Midtown Center	161	Manhattan	POLYGON ((991081.026 214453.698, 990952.644 21...	13906.0
235	236	0.044252	0.000103	Upper East Side North	236	Manhattan	POLYGON ((995940.048 221122.92, 995812.322 220...	12663.0
161	162	0.035270	0.000048	Midtown East	162	Manhattan	POLYGON ((992224.354 214415.293, 992096.999 21...	10759.0

3.1.14. Conclude with results

Busiest Hours: Commute peaks across morning and evening, mid-day steady, and late-night low.

Busiest Days: Weekends and Fridays typically higher trip volumes than midweek.

Busiest Months: Clear seasonality with summer months peaking, holiday-related dips present.

Revenue Trends

Monthly Revenue: Upward trend into summer, slight softening post-peak, stable baseline outside peak season.

Quarterly Revenue: Q3 highest, followed by Q2, Q4 and Q1 lower reflecting seasonality.

Fare Relationships

Distance & Fare: Strong positive relationship, fare increases near-linearly with trip distance, with higher variance at long distances.

Duration & Fare: Positive association, longer trips cost more, tempered by traffic and time-based components.

Passenger Count & Fare: Weak and limited impact, slight increases for higher counts, but not a primary driver.

Tip Relationships

Distance & Tip: Mild positive correlation, longer trips see higher average tips, though variability is large.

Geography

Busiest Pickup Zones: High activity concentrated in central Manhattan (Midtown, Lower Manhattan) and airport zones (JFK, LGA).

Choropleth Insight: Trip counts cluster around transit hubs, commercial cores, and major connectors, peripheral zones show lower intensity.

3.2. Detailed EDA: Insights and Strategies

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

PULocationID	DOLocationID	pickup_hour	speed_mph
48124	226	145	18 0.026569
62426	260	129	17 0.040746
17788	113	113	13 0.043484
46450	209	232	13 0.043579
18363	113	235	22 0.048105

How does identifying high-traffic, high-demand routes help us?

Summary

Capacity Planning: Align driver supply with peak routes and hours to reduce wait times and increase fulfilment rates.

Dispatch Optimization: Prioritize matching and repositioning toward high-demand corridors to cut empty miles and idle time.

Dynamic Pricing: Set smarter surge thresholds where demand is persistent, improving revenue while managing service reliability.

Fleet Efficiency: Identify bottlenecks, reroute or time-shift drivers to avoid congestion, lowering trip duration variability and fuel costs.

Service Design: Add preferred pickup/drop-off points and micro-hubs in hot zones to streamline boarding and reduce cancellations.

Customer Experience: Shorter ETAs and more consistent fares on busy routes improve satisfaction and repeat usage.

Infrastructure Insight: Surface chronic slow corridors for stakeholder dialogue (city, airports) on curb space, signals, or lane policies.

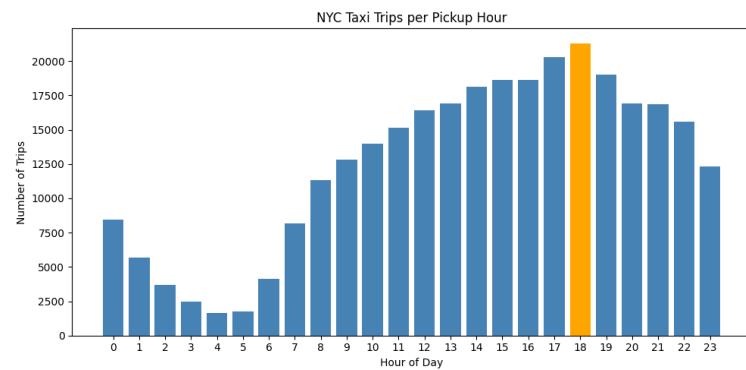
Partner Strategy: Target marketing and partnerships (events, venues, business districts) along top routes to lift utilization.

Forecasting & Staffing: Improve demand forecasts, schedule driver shifts and incentives around predictable route peaks.

Risk Management: Spot routes with high delay or incident rates, adjust guidance, training, or safety measures accordingly.

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

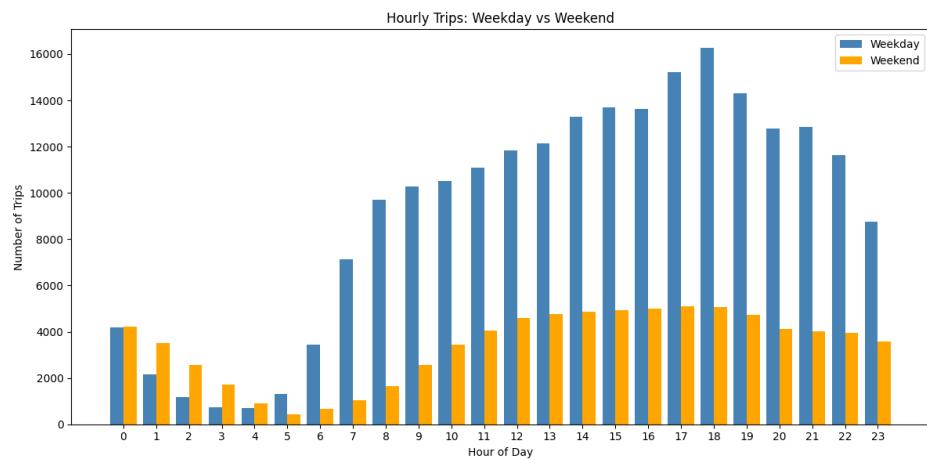
Busiest hour: **18:00 with 21307 trips**



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

Five busiest hours (scaled by sampling fraction):		
pickup_hour	sampled_trips	actual_trips
18	21307	2663375
17	20307	2538375
19	19019	2377375
16	18622	2327750
15	18610	2326250

3.2.4. Compare hourly traffic on weekdays and weekends



What can you infer from the above patterns? How will finding busy and quiet hours for each day help us?

Summary:

Demand Shape: Weekdays show pronounced commute peaks (morning/evening), weekends shift later with flatter midday peaks and softer late night tails. This indicates different rider intents (work vs leisure) and timing.

Staffing & Supply: Align driver availability to peak hours per day. Schedule more drivers for weekday commute windows, shift coverage later on weekends to cut ETAs, cancellations, and idle time.

Dynamic Pricing: Calibrate surge rules by hour/day to smooth demand-supply gaps. Lower thresholds during predictable peaks, relax during quiet hours to maintain affordability and utilization.

Dispatch & Repositioning: Proactively move drivers toward emerging hotspots before peaks (pre-commute staging), reduce empty miles, and improve match rates during busy hours.

Marketing & Incentives: Target quiet hours with promotions or driver incentives to lift utilization, and reward peak-hour reliability to retain drivers where service is most needed.

Operational Efficiency: Use quiet hours for maintenance/training and busy hours for maximum throughput, plan break rotations and fueling to avoid peak-time bottlenecks.

Forecasting & SLAs: Hour-by-hour patterns improve short-term forecasts and service level targets (ETAs, acceptance rates) tailored to each day's rhythm.

Customer Experience: Communicate expected busy periods, suggest off-peak alternatives, and optimize pickup points in peak corridors to reduce wait and variability.

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

Top 10 Pickup Zones:		
	PULocationID	total_pickups
117	132	15434
215	237	14063
145	161	13906
214	236	12663
146	162	10759
123	138	10209
166	186	10140
288	230	9910
127	142	9845
154	170	8824

Top 10 Dropoff Zones:		
	DOLocationID	total_dropoffs
226	236	13387
227	237	12583
153	161	11737
220	230	9095
162	170	8904
154	162	8576
134	142	8501
229	239	8319
133	141	7801
66	68	7666

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

Top 10 zones by pickups/dropoffs ratio:	
zone	pickup_drop_ratio
East Elmhurst	8.138554
JFK Airport	4.337830
LaGuardia Airport	2.643449
Penn Station/Madison Sq West	1.513885
Central Park	1.364693
West Village	1.360981
Greenwich Village South	1.338164
Midtown East	1.254548
Garment District	1.185339
Midtown Center	1.184800

Bottom 10 zones by pickups/dropoffs ratio:	
zone	pickup_drop_ratio
Whitestone	0.033898
Ocean Parkway South	0.041667
Glen Oaks	0.043478
Windsor Terrace	0.043796
Newark Airport	0.046460
Bayside	0.049180
Riverdale/North Riverdale/Fieldston	0.060000
Murray Hill-Queens	0.065217
Glendale	0.065574
Queensboro Hill	0.066667

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

During night hours (11pm to 5am) find the top 10 pickup and dropoff zones
Note that the top zones should be of night hours and not the overall top zones

Top 10 pickup zones during night hours :		
LocationID	zone	night_pickups
79	East Village	2588
132	JFK Airport	2330
249	West Village	2084
48	Clinton East	1689
148	Lower East Side	1609
114	Greenwich Village South	1389
230	Times Sq/Theatre District	1384
186	Penn Station/Madison Sq West	1146
164	Midtown South	1016
138	LaGuardia Airport	995

Top 10 dropoff zones during night hours:		
LocationID	zone	night_dropoffs
79	East Village	1383
48	Clinton East	1176
170	Murray Hill	1022
68	East Chelsea	973
107	Gramercy	957
141	Lenox Hill West	890
263	Yorkville West	857
249	West Village	768
236	Upper East Side North	752
90	Flatiron	743

3.2.8. Find the revenue share for nighttime and daytime hours

Filter for night hours (11 PM to 5 AM)

Revenue share night and day time hours:

Night: revenue = \$1,067,057.93, **share = 12.11%**

Day: revenue = \$7,742,737.72, **share = 87.89%**

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

For instance, suppose the average fare per mile for trips with 3 passengers is 3 USD/mile, then the fare per mile per passenger will be 1 USD/mile.

passenger_count	avg_fare_per_mile_per_passenger
1	17.38
2	9.17
3	6.07
4	6.92
5	2.65
6	2.09

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

Compare the average fare per mile for different days and for different times of the day

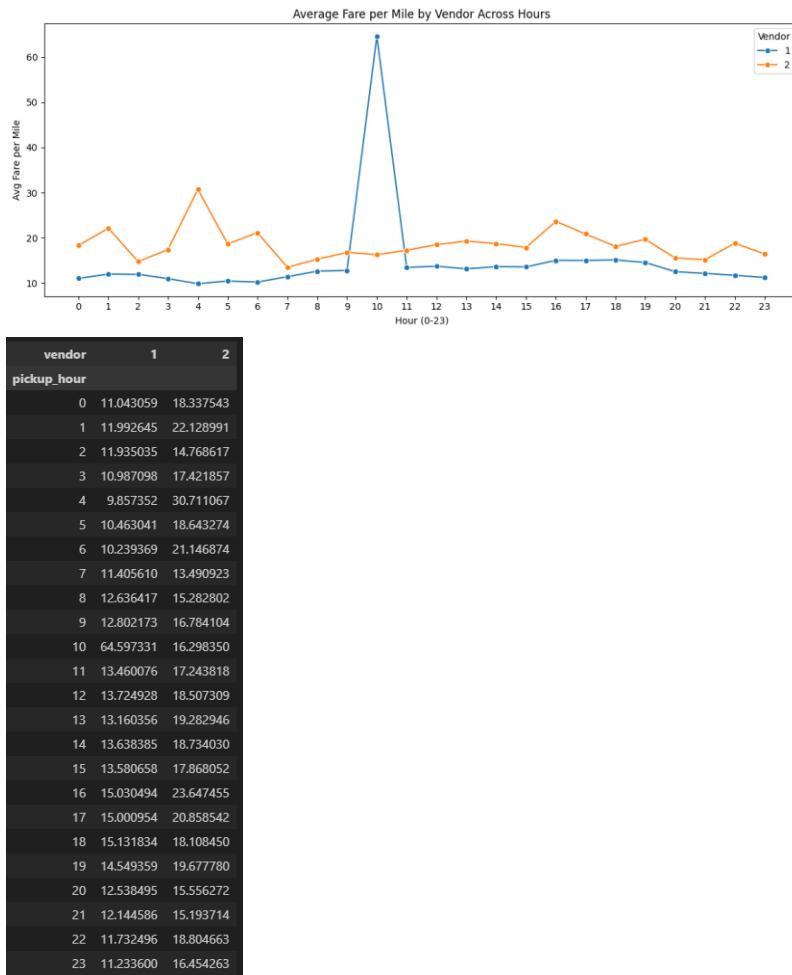
hour_of_day	avg_fare_per_mile
0	16.713528
1	19.972707
2	14.119125
3	16.016258
4	26.020969
5	16.371254
6	17.995639
7	12.873920
8	14.535945
9	15.672374
10	30.063646
11	16.183583
12	17.211646
13	17.544149
14	17.359797
15	16.707947
16	21.322169
17	19.299952
18	17.342673
19	18.414410
20	14.824466
21	14.468450
22	17.203306
23	15.262888

day_of_week	avg_fare_per_mile
Monday	16.004423
Tuesday	21.911992
Wednesday	17.334284
Thursday	18.673340
Friday	15.487594
Saturday	16.564001
Sunday	17.065971

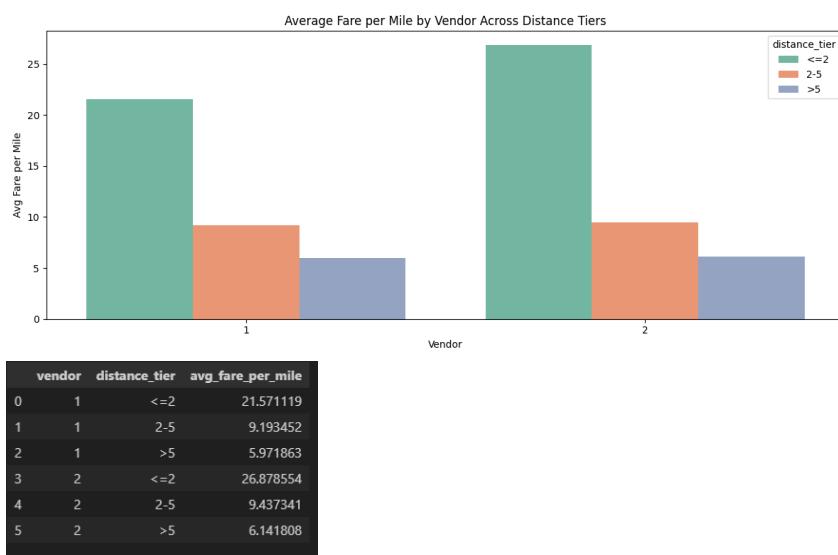
3.2.11. Analyse the average fare per mile for the different vendors

Compare fare per mile for different vendors

Average fare per mile by vendor across hours of the day

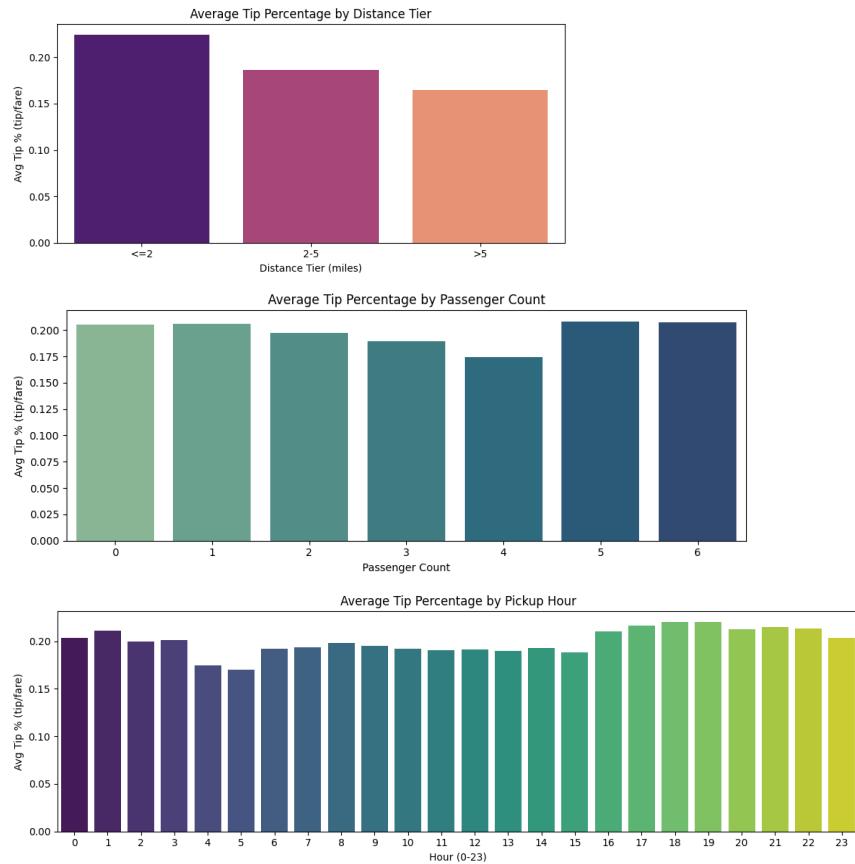


3.2.12. Compare the fare rates of different vendors in a distance-tiered fashion



3.2.13. Analyse the tip percentages

Tip percentage analysis by distance tiers, passenger counts, and pickup hour

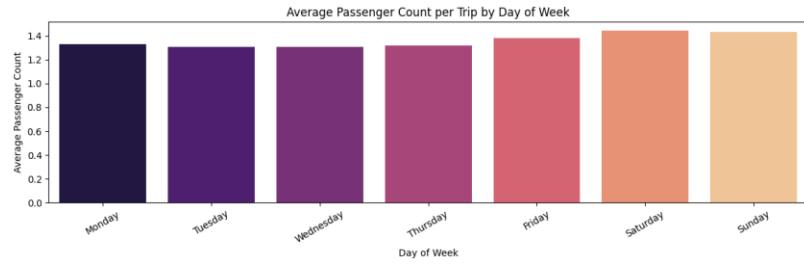
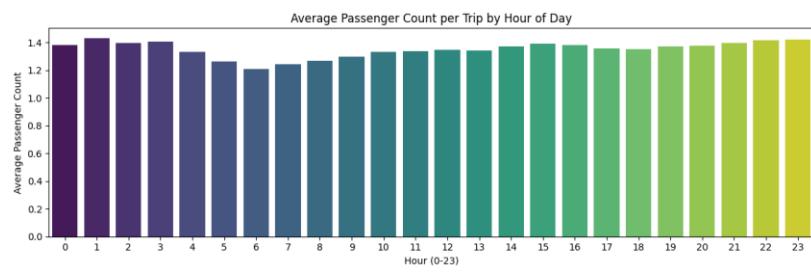


	pickup_hour	avg_tip_pct	trips
0	0	0.203629	8323
1	1	0.211328	5594
2	2	0.200134	3656
3	3	0.201669	2408
4	4	0.174502	1574
5	5	0.170263	1703
6	6	0.192011	4074
7	7	0.194027	8108
8	8	0.198304	11236
9	9	0.195063	12751
10	10	0.192466	13835
11	11	0.190573	14996
12	12	0.191666	16285
13	13	0.190053	16803
14	14	0.192692	18006
15	15	0.188623	18460
16	16	0.210114	18451
17	17	0.216637	20133
18	18	0.220397	21165
19	19	0.220432	18888
20	20	0.212781	16796
21	21	0.214734	16762
22	22	0.213828	15497
23	23	0.203386	12204

	distance_tier	avg_tip_pct	trips
0	<=2	0.22434	163612
1	2-5	0.18608	82216
2	>5	0.16478	51880

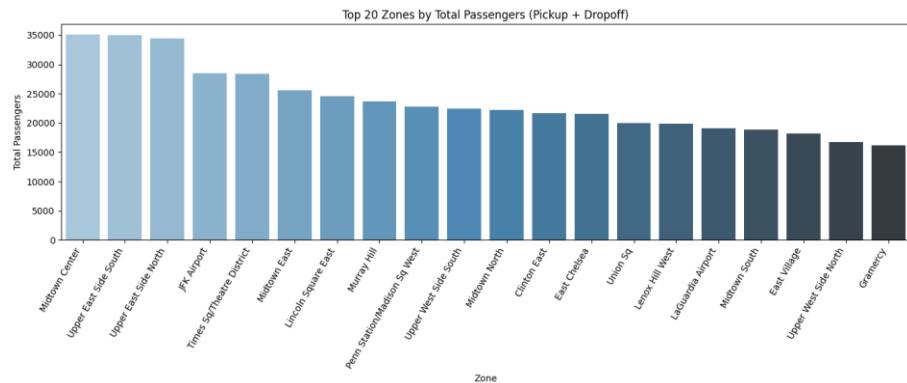
	passenger_count_clean	avg_tip_pct	trips
0	0	0.205318	4540
1	1	0.205870	226281
2	2	0.197267	43726
3	3	0.189255	10888
4	4	0.174000	5949
5	5	0.208218	3766
6	6	0.207250	2558

3.2.14. Analyse the trends in passenger count



	pickup_hour	avg_passenger_count	trips
0	0	1.382168	8423
1	1	1.430739	5667
2	2	1.396988	3718
3	3	1.407347	2450
4	4	1.330037	1618
5	5	1.264434	1732
6	6	1.209053	4109
7	7	1.244425	8162
8	8	1.266543	11319
9	9	1.294534	12844
10	10	1.331327	13959
11	11	1.336682	15130
12	12	1.347144	16437
13	13	1.3419	16926
14	14	1.370709	18149
15	15	1.390435	18610
16	16	1.382612	18622
17	17	1.354016	20307
18	18	1.350964	21307
19	19	1.370314	19019
20	20	1.374408	16904
21	21	1.395636	16864
22	22	1.414215	15603
23	23	1.420698	12301
	day_of_week	avg_passenger_count	trips
0	Monday	1.30784	37396
1	Tuesday	1.306854	43493
2	Wednesday	1.306971	46089
3	Thursday	1.319445	47163
4	Friday	1.378513	44622
5	Saturday	1.443986	43632
6	Sunday	1.430541	37785

3.2.15. Analyse the variation of passenger counts across zones



Zone	pickup_total_pass	pickup_trips	pickup_avg_pass_per_trip	dropoff_total_pass	dropoff_trips	dropoff_avg_pass_per_trip	overall_total_passengers	
151	Midtown Center	19047	13906	1.369697	16056	11737	1.367982	35103
226	Upper East Side South	18409	14063	1.309038	16590	12583	1.318446	34999
225	Upper East Side North	16831	12663	1.329148	17587	13387	1.313737	34418
119	JFK Airport	22980	15434	1.488921	5488	3558	1.54244	28468
219	Times Sq/Theatre District	14541	9910	1.467306	13803	9095	1.517647	28344
152	Midtown East	14200	10759	1.319825	11371	8576	1.32591	25571
132	Lincoln Square East	13268	9845	1.347689	11342	8501	1.334196	24610
160	Murray Hill	11739	8824	1.330349	11989	8904	1.346473	23728
176	Penn Station/Madison Sq West	13533	10140	1.334615	9206	6698	1.37444	22739
228	Upper West Side South	11209	8396	1.33504	11250	8319	1.352326	22459

For a more detailed analysis, we can use the `zones_with_trips` GeoDataFrame
 Create a new column for the average passenger count in each zone.

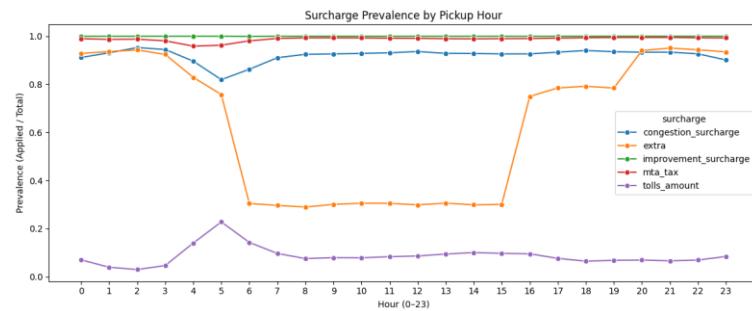
Add average passenger count per zone to zones_with_trips GeoDataFrame

	zone	avg_passenger_count	trips
252	Willets Point	2.0	1
5	Arrrochar/Fort Wadsworth	2.0	3
153	Marine Park/Floyd Bennett Field	2.0	1
119	Highbridge Park	2.0	1
66	Dyker Heights	1.8	5
46	Clairemont/Bathgate	1.727273	11
11	Battery Park	1.691729	133
193	Randalls Island	1.666667	9
194	Red Hook	1.653846	26
227	Sunset Park West	1.631579	19
174	Oakland Gardens	1.625	8
0	Newark Airport	1.571429	42
156	Maspeth	1.565217	23
134	Kew Gardens Hills	1.555556	9
159	Middle Village	1.5	6
260	World Trade Center	1.498435	1597
131	JFK Airport	1.488921	15434
42	Central Park	1.484152	5048
56	Corona	1.47619	42
55	Corona	1.47619	42

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

Surcharge prevalence: by hour

surcharge	trips_applied	total_trips	prevalence
improvement_surcharge	300162	300180	0.999940
mta_tax	297577	300180	0.991329
congestion_surcharge	278351	300180	0.927280
extra	182135	300180	0.606753
tolls_amount	24203	300180	0.080628



4. Conclusions

4.1. Final Insights and Recommendations

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

Summary

Busiest Hours: Clear commuter peaks on weekdays (morning/evening), weekend peaks shift later with flatter midday demand.

Busiest Days/Months: Fridays and weekends lead trips, seasonal peak in summer with post-peak softening.

Busiest Zones: Central Manhattan corridors (Midtown, Lower Manhattan) and airport zones (JFK, LGA) dominate pickups and dropoffs, including night hours.

Revenue & Pricing

Night vs Day: Nighttime revenue is a smaller but material share, late-night hotspots cluster around entertainment districts and airports.

Fare Drivers: Fare scales strongly with distance and moderately with duration, passenger count has weak impact.

Vendor/Tier Effects: Fare-per-mile varies by vendor and distance tier, shorter trips (≤ 2 miles) show higher per-mile rates due to base fees.

Customer Experience

Tips: Tip percentage rises modestly with distance, low tips more prevalent at certain late-night hours and shorter rides.

Surcharges: Congestion and airport fees concentrate by zone and hour (Manhattan, airport corridors), tolls vary by route selection.

Routing & Bottlenecks

Slow Routes: Specific PU to DO corridors exhibit low speeds during peak hours, speeds improve off-peak. Pre-commute staging reduces delays.

Night Traffic: 23:00-05:59 has distinct top zones, optimize late-night coverage with fewer zones but deeper presence.

Geography

Choropleth Insights: Trip volumes cluster around transit hubs, business districts, and connectors, peripheral zones show lower intensity.

Zone Mix: Several zones have high solo rider share, others skew to group travel (higher average passenger counts).

Recommendations

Supply Allocation: Increase driver availability in weekday commute windows and shift supply later on weekends, stage drivers in Midtown, Financial District, and airports before peaks.

Dynamic Pricing: Tune surge thresholds by hour/day and zone, lower thresholds during predictable peaks to improve reliability, relax during quiet hours to sustain utilization.

Dispatch & Repositioning: Real-time repositioning toward emerging hotspots, pre-commute staging reduces empty miles and improves match rates.

Route Optimization: Prefer faster corridors during identified slow-hour windows, guide drivers to avoid known bottlenecks (signal-heavy avenues, construction zones) using zone-level speed metrics.

Night Operations: Focus coverage on top late-night pickup/dropoff zones, add micro-hubs and safe, well-lit pickup points to reduce cancellations and wait times.

Vendor Strategy: For short trips, consider fee transparency and incentives to mitigate higher per-mile base effects, benchmark vendors' fare-per-mile by tier to align quality and pricing.

Surcharge Awareness: Communicate surcharge expectations in high-incidence zones/hours, consider routing alternatives to minimize tolls when ETAs are comparable.

Tip Uplift: Encourage tipping via UX prompts after longer trips and at hours with historically higher tips, address low-tip cohorts (short rides, certain hours) with service cues and driver guidance.

Staffing & Maintenance: Schedule maintenance/training during quiet hours, use driver incentives to cover late-night peaks and early-morning gaps.

Forecasting & SLAs: Set hour-by-hour SLAs (ETAs, acceptance rates) per zone, integrate seasonal effects and event calendars for proactive staffing.

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.

Core Positioning

Peak Corridors: Stage cabs pre-peak in Midtown East/West, Lower Manhattan, and Financial District, maintain rolling buffers near Penn Station, Grand Central, and major hotel clusters.

Airport Hubs: Keep dedicated pools at JFK/LGA with dynamic spillover to nearby zones, increase late-night presence for flight banks and early-morning departures.

Nightlife Anchors: From 22:00-02:00, position near entertainment districts (West Village, Lower East Side, Williamsburg connectors) and transit transfer points.

Event-Driven Zones: Add flexible capacity near stadiums/convention centers on event calendars, deploy 60-90 minutes pre-end to absorb exit surges.

Temporal Tactics

Weekday Commute: 07-10 and 16-19 - pre-stage in business districts and major transit nodes, prioritize fast-merge corridors identified with higher throughput.

Weekend Shift: 11-15 and 20-01 - pivot toward shopping/entertainment zones, reduce early-morning coverage except airport rings.

Shoulder Hours: 05-07 and 21-23 - thinner spread with micro-hubs at multi-line subway stations, rebalance toward emerging hotspots via live heatmaps.

Micro-Hubs & Routing

Micro-Hubs: Define safe, high-visibility pickup spots per busy zone to cut cancellations, rotate cabs through hubs to minimize idle clustering.

Fast Corridors: Use speed audit insights to favor quicker east-west/avenue combinations at peak, avoid historically slow streets under construction or signal-heavy segments.

Toll-Aware Paths: Where ETAs are comparable, pick lower-toll routes for price-sensitive corridors, keep high-reliability toll routes for tight SLAs.

Supply Controls

Tiered Buffers: Maintain minimum cab counts per top-10 pickup zones, add overflow pool that repositions based on real-time demand spikes.

Dynamic Reposition: Every 15–30 minutes, pull 10–15% of idle cabs from quiet zones into adjacent growing hotspots, use contiguous-zone flows to reduce empty miles.

Driver Shifts: Align shifts to peaks (weekday commute, weekend late-night) and seasonality, schedule maintenance/training in quiet windows.

Pricing & Incentives

Surge Guardrails: Calibrate by hour/day/zone to smooth demand-supply gaps without over-suppressing trips, lower thresholds in predictable peaks, relax off-peak.

Incentives: Offer short bonuses for coverage gaps (late-night airport returns, early-morning city entries), reward adherence to hub placements in peak windows.

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

Time-Based Tiers:

Peak Weekday Commute (07-10, 16-19): Slightly lower surge thresholds in top zones to maintain reliability, cap max multiplier to protect competitiveness, use micro surges in adjacent zones to preposition supply.

Weekend Late Peak (20-01): Allow modest surges in nightlife corridors, pair with driver incentives to ensure coverage and stabilize ETAs.

Zone-Aware Pricing:

Airport Corridors: Transparent airport fee messaging, moderate surge near flight banks, offer bundled fixed-fare options for common airport routes to reduce price anxiety and boost conversion.

High-Congestion Zones (Manhattan core): Consider small congestion add-ons with ETA guarantees, offer alternative routing discounts for comparable travel times on lower-toll paths.

Distance-Tier Optimization:

Short Trips (≤ 2 miles): Base-fee weight drives high per-mile costs, introduce “short-hop” discount window during quiet hours to stimulate demand and improve utilization.

Mid Trips (2–5 miles): Keep standard per-mile, test small dynamic adjustments by hour to smooth demand.

Long Trips (>5 miles): Volume discounts on per-mile beyond a threshold, maintain consistent minimums to avoid under pricing on very long trips.

Vendor Benchmarking:

Monitor competitor fare-per-mile by tier and hour, set competitive guardrails (e.g., within 5-10% in top corridors). If your vendor shows higher short-trip per-mile, offset via promo codes or loyalty accruals for frequent short rides.

Surcharge Hygiene:

Clearly itemize surcharges in high-incidence zones/hours (congestion, airport), minimize surprises. Where tolls add little ETA benefit, default to lower-toll routes, keep premium toll routes for SLA-critical trips with explicit user opt-in.