Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

Report: Optimising NYC Taxi Operations

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

1. Data Preparation

Import Libraries

[9]:
```
# Import warnings
import warnings
warnings.filterwarnings("ignore")
```

[10]:
```
# Import the libraries you will be using for analysis
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[11]:
```
# Recommended versions
# numpy version: 1.26.4
# pandas version: 2.2.2
# matplotlib version: 3.10.0
# seaborn version: 0.13.2

# Check versions
print("numpy version:", np.__version__)
print("pandas version:", pd.__version__)
print("matplotlib version:", plt.matplotlib.__version__)
print("seaborn version:", sns.__version__)
!where python
```

```
numpy version: 1.26.4
pandas version: 2.2.2
matplotlib version: 3.10.0
seaborn version: 0.13.2
C:\Users\prash\anaconda3\python.exe
C:\Users\prash\AppData\Local\Microsoft\WindowsApps\python.exe
```

1.1. Loading the dataset

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

You will see twelve files, one for each month.

To read parquet files with Pandas, you have to follow a similar syr

`df = pd.read_parquet('file.parquet')`

[12]:
```
# Try loading one file

# df = pd.read_parquet('2023-1.parquet')
# df.info()

jan_data = pd.read_parquet("2023-1.parquet")
jan_data.info()
```

1.1.1. Sample the data and combine the files

```
# Sample the data
# It is recommmended to not load all the files at once to avoid memory overload

# Step: Efficiently sample 5% from each monthly file (to avoid loading everything at once)

import glob
import pandas as pd

# Set the path pattern for your monthly parquet files
data_folder = r"C:\Users\prash\Downloads\Datasets and Dictionary-NYC\Datasets and Dictionary\trip_records\*.parquet"

# Find all parquet files in the folder
monthly_files = glob.glob(data_folder)

# List to store sampled DataFrames for each file
monthly_samples = []

for filepath in monthly_files:
    # Load one file at a time
    temp_df = pd.read_parquet(filepath)

    # Optional: Create date and hour columns if needed for stratified sampling
    # temp_df['pickup_date'] = pd.to_datetime(temp_df['tpep_pickup_datetime']).dt.date
    # temp_df['pickup_hour'] = pd.to_datetime(temp_df['tpep_pickup_datetime']).dt.hour

    # Sample 5% of the rows randomly from this file
    file_sample = temp_df.sample(frac=0.05, random_state=42)

    # Collect the sample
    monthly_samples.append(file_sample)

# Combine all sampled pieces into a single DataFrame
taxi_sampled = pd.concat(monthly_samples, ignore_index=True)

# Quick info on the combined sample
print("Sampled data info:")
taxi_sampled.info()
print("\nFirst few rows:")
print(taxi_sampled.head())
```

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

- Iterated over each 2023 monthly Parquet file and parsed pickup timestamps to extract **date** and **hour**.

- For every distinct **date** within a month, looped through all **24 hours (0–23)**.

- Whenever an hour contained trips, drew a **5% random sample** of those records (with `random state=1` for reproducibility).

- Combined the sampled hourly slices into a **monthly** Data Frame, then concatenated all months into a **single annual** dataset used for the remainder of the assignment.

Total number of records in sampled dataset: **2206368**.

```
Sampled data info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2206368 entries, 0 to 2206367
Data columns (total 20 columns):
 #   Column                 Dtype
---  ------                 -----
 0   VendorID               int64
 1   tpep_pickup_datetime   datetime64[us]
 2   tpep_dropoff_datetime  datetime64[us]
 3   passenger_count        float64
 4   trip_distance          float64
 5   RatecodeID             float64
 6   store_and_fwd_flag     object
 7   PULocationID           int64
 8   DOLocationID           int64
 9   payment_type           int64
 10  fare_amount            float64
 11  extra                  float64
 12  mta_tax                float64
 13  tip_amount             float64
 14  tolls_amount           float64
 15  improvement_surcharge  float64
 16  total_amount           float64
 17  congestion_surcharge   float64
 18  airport_fee            float64
 19  Airport_fee            float64
```

1.1.2.

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

```
# Take a small percentage of entries from each hour of every date.
# Iterating through the monthly data:
#   read a month file -> day -> hour: append sampled data -> move to next hour -> move to next day after 24 hours -> move to next month file
# Create a single dataframe for the year combining all the monthly data

# Select the folder having data files
import os

# Select the folder having data files
os.chdir(r"C:\Users\prash\Downloads\Datasets and Dictionary-NYC\Datasets and Dictionary\trip_records")

# Create a list of all the twelve files to read
file_list = os.listdir()

# initialise an empty dataframe
df = pd.DataFrame()


# iterate through the list of files and sample one by one:
for file_name in file_list:
    try:
        # file path for the current file
        file_path = os.path.join(os.getcwd(), file_name)

        # Reading the current file



        # We will store the sampled data for the current date in this df by appending the sampled data from each hour to this
        # After completing iteration through each date, we will append this data to the final dataframe.
        sampled_data = pd.DataFrame()

        # Loop through dates and then loop through every hour of each date

            # Iterate through each hour of the selected date

                # Sample 5% of the hourly data randomly

                # add data of this hour to the dataframe

        # Concatenate the sampled data of all the dates to a single dataframe
        df = pd.concat([df, sampled_data])# we initialised this empty DF earlier

    except Exception as e:
        print(f"Error reading file {file_name}: {e}")
```

```
# Store the df in csv/parquet
# df.to_parquet('')
# Export the sampled data to a Parquet file without the index column
taxi_sampled.to_parquet("nyc_taxi_hourly_sampled.parquet", index=False)
```

2. Data Cleaning

2.1 Fixing Columns

2.11 Fix the index

```
# Fix the index and drop any columns that are not needed

# Remove unwanted columns and reset the index

# Remove specified columns if present and reset the DataFrame index
unwanted_cols = ["store_and_fwd_flag", "extra", "mta_tax", "improvement_surcharge"]
hourly_sampled_df.drop(columns=unwanted_cols, inplace=True, errors="ignore")
hourly_sampled_df.reset_index(drop=True, inplace=True)

# Display final column names
print("Columns after cleaning:", list(hourly_sampled_df.columns))
```

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

1. Reset the Data Frame to the **default integer index**, replacing the previous index.
2. Reviewed **missing values** across all columns. The highest null rates are in the two airport-fee fields (airport fee and Airport fee). I left these untouched here because they're addressed in the next step.
3. Assessed remaining columns and found store_and_fwd_flag adds little analytical value and contains nulls, so I **dropped** it.

2.1.2 Combine the two airport_fee columns

```
# Combine the two airport fee columns

# Combine 'Airport_fee' and 'airport_fee' columns into one, adding values and handling missing data

if "Airport_fee" in hourly_sampled_df.columns and "airport_fee" in hourly_sampled_df.columns:
    hourly_sampled_df["Airport_fee"] = hourly_sampled_df["Airport_fee"].fillna(0) + hourly_sampled_df["airport_fee"].fillna(0)
    hourly_sampled_df.drop(columns=["airport_fee"], inplace=True, errors="ignore")

# Check the updated columns
print("Columns after combining airport fees:", list(hourly_sampled_df.columns))
```

1 Assessed the distribution of both columns by computing the **median** (middle value) and **mode** (most frequent); **both were 0**.
2 Checked for any rows where **both columns had values simultaneously**; the **overlap count was 0**.
3 Based on these findings, **imputed missing values as 0** in both columns.
4 **Merged** the two fields into a single column named **`airport_fee`**.

i.

2.1. Handling Missing Values

2.1.1. Find the proportion of missing values in each column

Highest proportion of missing values in columns 'passenger_count', 'RatecodeID' and 'congestion_surcharge'.

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

```
# Find the proportion of missing values in each column
missing_percent = hourly_sampled_df.isnull().mean() * 100
missing_percent = missing_percent[missing_percent > 0].sort_values(ascending=False)

print("Percentage of missing values by column:\n", missing_percent)
```

```
Percentage of missing values by column:
 passenger_count         3.42282
RatecodeID              3.42282
congestion_surcharge    3.42282
dtype: float64
```

2.1.2. Handling missing values in passenger_count

1. Imputed missing values in `passenger_count` using the (median).
2. Treated zeros as invalid and replaced them with the mode (1), since the majority of trips in the data involve one passenger.

```
# Replace zero values in 'passenger_count' with the median
median_passenger = hourly_sampled_df["passenger_count"].median()
hourly_sampled_df.loc[hourly_sampled_df["passenger_count"] == 0, "passenger_count"] = median_passenger

# Show rows that still have missing values after handling 'passenger_count'
still_missing = hourly_sampled_df[hourly_sampled_df.isnull().any(axis=1)]
```

2.1.2 Handle missing values in RatecodeID

1.Since the **mode is 1** (standard rate) — indicating most trips use this code — imputed all missing RatecodeID values with **1**.

```
# Fix missing values in 'RatecodeID'
if "RatecodeID" in hourly_sampled_df.columns:
    most_common_ratecode = hourly_sampled_df["RatecodeID"].mode()[0]
    hourly_sampled_df["RatecodeID"] = hourly_sampled_df["RatecodeID"].fillna(most_common_ratecode)
    print("Missing RatecodeID values after filling:", hourly_sampled_df["RatecodeID"].isnull().sum())
```

```
Missing RatecodeID values after filling: 0
```

2.1.3. Impute NaN in congestion_surcharge

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

2.2.4 [5 marks]

Impute NaN in `congestion_surcharge`

[58]:
```
# handle null values in congestion_surcharge


median_congestion = hourly_sampled_df["congestion_surcharge"].median()
hourly_sampled_df["congestion_surcharge"] = hourly_sampled_df["congestion_surcharge"].fillna(median_congestion)
```

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

[59]:
```
# Handle any remaining missing values
# Fill any remaining missing values: mode for objects, median for numerics

for column in hourly_sampled_df.columns:
    if hourly_sampled_df[column].isnull().any():
        if hourly_sampled_df[column].dtype == "object":
            mode_val = hourly_sampled_df[column].mode()[0]
            hourly_sampled_df[column] = hourly_sampled_df[column].fillna(mode_val)
        else:
            med_val = hourly_sampled_df[column].median()
            hourly_sampled_df[column] = hourly_sampled_df[column].fillna(med_val)
```

2.2. Handling Outliers and Standardising Values

[62]:
```
# Describe the data and check if there are any potential outliers present
# Check for potential out of place values in various columns

# Show summary statistics for numeric features and look for potential outliers
print("Summary of dataset:\n", hourly_sampled_df.describe())

# Function to identify outliers using the IQR method
def get_outlier_counts(data, columns):
    outlier_summary = {}
    for column in columns:
        q1 = data[column].quantile(0.25)
        q3 = data[column].quantile(0.75)
        iqr = q3 - q1
        low = q1 - 1.5 * iqr
        high = q3 + 1.5 * iqr
        count = ((data[column] < low) | (data[column] > high)).sum()
        if count > 0:
            outlier_summary[column] = count
    return outlier_summary

# Choose numeric columns to check
num_features = hourly_sampled_df.select_dtypes(include=[np.number]).columns
outlier_counts = get_outlier_counts(hourly_sampled_df, num_features)

# Print outlier results
print("\nColumns with detected outliers:")
for col, cnt in outlier_counts.items():
    print(f" - {col}: {cnt} potential outliers")
```

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

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

```
[63]: # remove passenger_count > 6
# Filter out trips with more than 6 passengers
hourly_sampled_df = hourly_sampled_df[hourly_sampled_df["passenger_count"] <= 6]

# Show distribution after filtering
print("Passenger count distribution after cleaning:\n", hourly_sampled_df["passenger_count"].value_counts())
```

```
Passenger count distribution after cleaning:
 passenger_count
1.0    1712770
2.0     322058
3.0      79969
4.0      44933
5.0      27888
6.0      18733
Name: count, dtype: int64
```

```
[64]: # Continue with outlier handling

# Function to remove outliers using IQR for a list of columns
def drop_outliers(data, columns):
    cleaned = data.copy()
    for col in columns:
        q1 = cleaned[col].quantile(0.25)
        q3 = cleaned[col].quantile(0.75)
        iqr = q3 - q1
        min_val = q1 - 1.5 * iqr
        max_val = q3 + 1.5 * iqr
        cleaned = cleaned[(cleaned[col] >= min_val) & (cleaned[col] <= max_val)]
    return cleaned

# Choose numeric columns for outlier removal
num_cols = hourly_sampled_df.select_dtypes(include=[np.number]).columns
hourly_sampled_df = drop_outliers(hourly_sampled_df, num_cols)

# Print updated summary
print("Outliers removed. Updated dataset summary:\n", hourly_sampled_df.describe())
```

```
[65]: # Do any columns need standardising?

from sklearn.preprocessing import StandardScaler

# Identify all numeric columns (including ID/location columns)
numeric_cols = hourly_sampled_df.select_dtypes(include=[float, int]).columns.tolist()

# Find columns with std > 1 (candidates for standardization)
columns_to_standardize = [col for col in numeric_cols if hourly_sampled_df[col].std() > 1]

print("Columns recommended for standardization:", columns_to_standardize)
```

```
Columns recommended for standardization: ['trip_distance', 'PULocationID', 'DOLocationID', 'fare_amount', 'tip_amount', 'total_amount']
```

3. Exploratory Data Analysis

3.1. General EDA: Finding Patterns and Trends

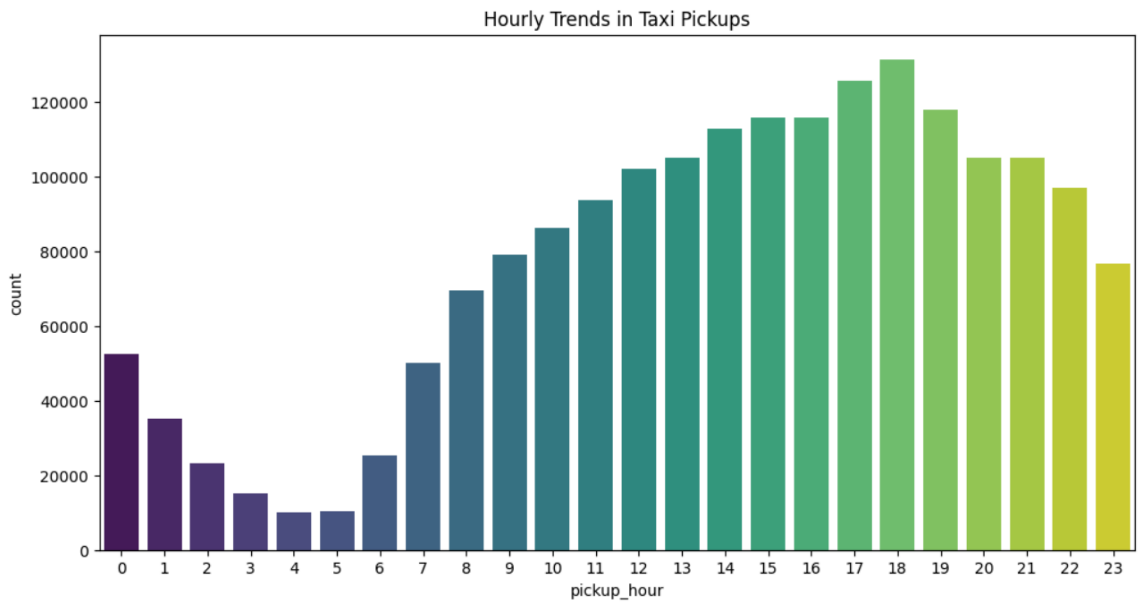
3.1.1. Classify variables into categorical and numerical

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

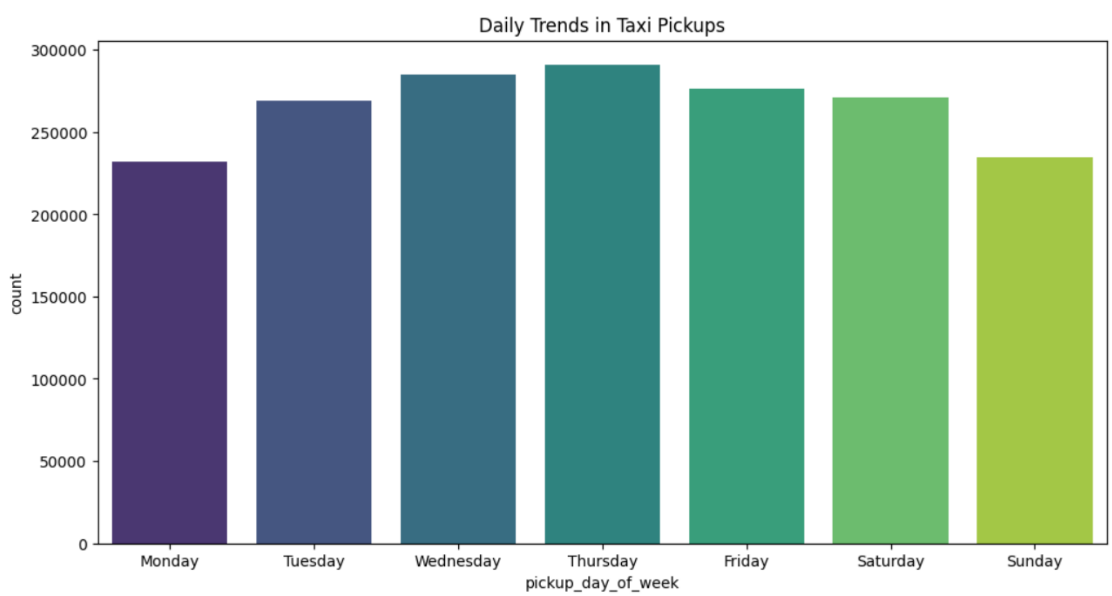
```
Categorise the varaibles into Numerical or Categorical.
* `VendorID`:Categorical
* `tpep_pickup_datetime`:Numerical
* `tpep_dropoff_datetime`:Numerical
* `passenger_count`:Numerical
* `trip_distance`:Numerical
* `RatecodeID`:Categorical
* `PULocationID`:Categorical
* `DOLocationID`:Categorical
* `payment_type`:Categorical
* `pickup_hour`:Numerical
* `trip_duration`:Numerical


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


* `fare_amount`
* `extra`
* `mta_tax`
* `tip_amount`
* `tolls_amount`
* `improvement_surcharge`
* `total_amount`
* `congestion_surcharge`
* `airport_fee`
```

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

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

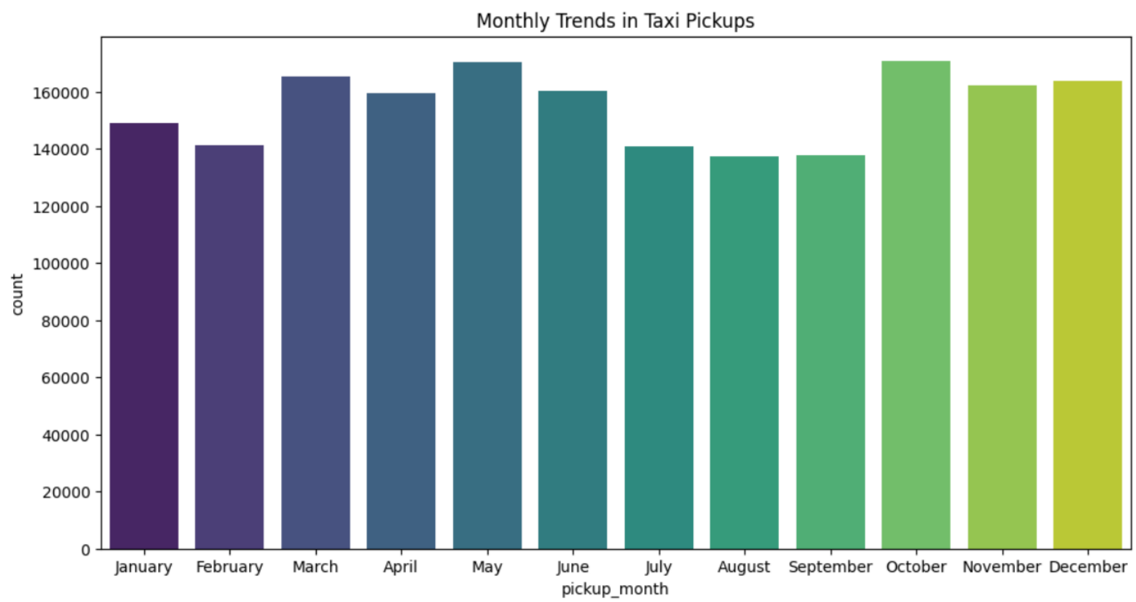


Taxi pickups rise from 6am and the peak reaches at 6pm, which indicates people travel during these hours for work.



Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

Pickups are at highest from Tuesday to Friday, showing pattern of people traveling within the city, possibly to work or airport.



Highest pickups reported from May and October, with March, November and December close by. This indicates people traveling more during holiday season (starting from late Oct to December). On the other hand, March to May being spring break and summer vacation time, taxi pickups see a spike in usage.

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

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

```
# Create a df with non zero entries for the selected parameters.

# Filter the DataFrame to only include records with all financial values greater than zero

columns_to_keep = ["fare_amount", "tip_amount", "total_amount", "trip_distance"]

filtered_data = hourly_sampled_df[(hourly_sampled_df[columns_to_keep] > 0).all(axis=1)].copy()

print(f"Shape after zero-value filtering: {filtered_data.shape}")
print("Statistics of financial columns after filtering:\n", filtered_data[columns_to_keep].describe())
```

```
Shape after zero-value filtering: (1035088, 18)
Statistics of financial columns after filtering:
        fare_amount    tip_amount  total_amount  trip_distance
count  1.035088e+06  1.035088e+06  1.035088e+06   1.035088e+06
mean   1.286094e+01  3.188257e+00  2.083078e+01   1.781554e+00
std    5.423234e+00  1.340285e+00  6.426143e+00   1.076067e+00
min    2.800000e+00  1.000000e-02  7.020000e+00   1.000000e-02
25%    8.600000e+00  2.160000e+00  1.595000e+01   1.000000e+00
50%    1.210000e+01  3.000000e+00  1.968000e+01   1.510000e+00
75%    1.630000e+01  4.020000e+00  2.484000e+01   2.300000e+00
max    2.960000e+01  7.210000e+00  3.894000e+01   6.720000e+00
```

1.1.2. Analyse the monthly revenue trends

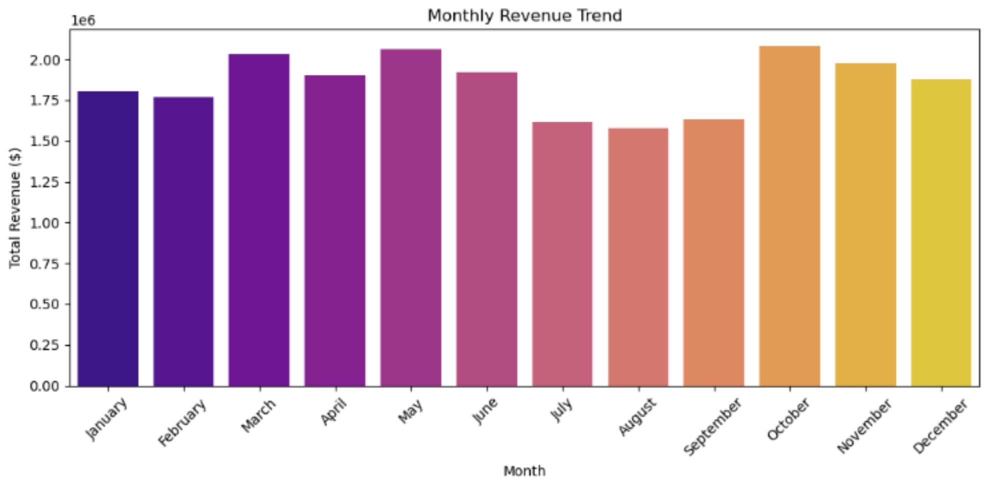
Analyse the monthly revenue (`total_amount`) trend

[78]:

```
# Group data by month and analyse monthly revenue

import matplotlib.pyplot as plt
import seaborn as sns

# Make sure pickup_month column exists
if "pickup_month" not in hourly_sampled_df.columns:
    hourly_sampled_df["pickup_month"] = hourly_sampled_df["tpep_pickup_datetime"].dt.month_name()

# Define month order for consistent plotting
month_order = [
    "January", "February", "March", "April", "May", "June",
    "July", "August", "September", "October", "November", "December"
]

# Calculate total monthly revenue
monthly_revenue = (
    hourly_sampled_df.groupby("pickup_month")["total_amount"]
    .sum()
    .reindex(month_order)
)

# Plot monthly revenue trend
plt.figure(figsize=(10, 5))
sns.barplot(x=monthly_revenue.index, y=monthly_revenue.values, palette="plasma")
plt.title("Monthly Revenue Trend")
plt.xlabel("Month")
plt.ylabel("Total Revenue ($)")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Print revenue breakdown
print("Monthly Revenue Breakdown:\n", monthly_revenue)
```

Name- Prashant shrivastava

TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS




```
Monthly Revenue Breakdown:
 pickup_month
January      1805434.89
February     1765421.00
March        2030883.24
April        1905727.95
May          2064690.39
June         1921023.51
July         1617824.11
August       1577684.96
September    1636259.24
October      2078892.84
November     1978658.01
December     1875950.26
Name: total_amount, dtype: float64
```

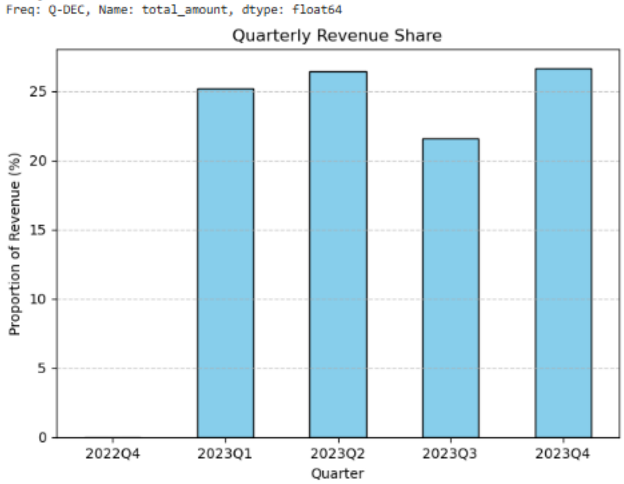
Monthly revenue increases are at highest during May and October, directly proportional to the trend we saw in above when analysing monthly trends of taxi pickups.

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

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

```
[79]: # Calculate proportion of each quarter

# Assign each record to a quarter
filtered_data["quarter"] = filtered_data["tpep_pickup_datetime"].dt.to_period("Q")

# Aggregate revenue by quarter
quarter_revenue = filtered_data.groupby("quarter")["total_amount"].sum()

# Calculate each quarter's percentage of annual revenue
quarter_proportion = (quarter_revenue / quarter_revenue.sum()) * 100

print("Revenue share by quarter (%):\n", quarter_proportion.round(2))

# Plot the proportion as a bar chart
quarter_proportion.plot(kind="bar", color="skyblue", edgecolor="black")
plt.xlabel("Quarter")
plt.ylabel("Proportion of Revenue (%)")
plt.title("Quarterly Revenue Share")
plt.xticks(rotation=0)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.tight_layout()
plt.show()
```

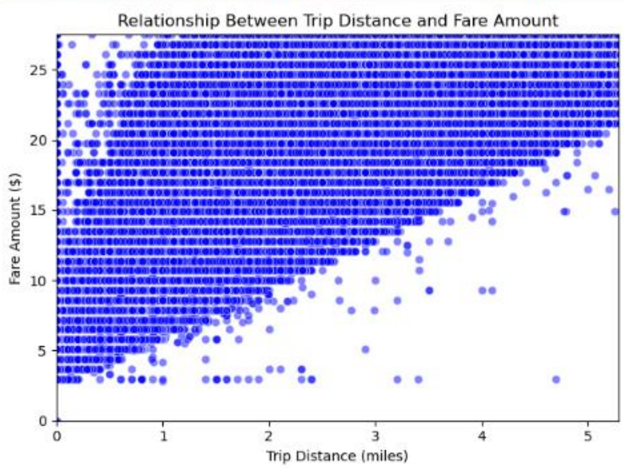
```
Revenue share by quarter (%):
 quarter
2022Q4     0.00
2023Q1    25.23
2023Q2    26.47
2023Q3    21.62
2023Q4    26.68
Freq: Q-DEC, Name: total_amount, dtype: float64
```



Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

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

```
[81]:
# Show how trip fare is affected by distance
import matplotlib.pyplot as plt
import seaborn as sns

# Select relevant columns
trip_distance = hourly_sampled_df["trip_distance"]
fare_amount = hourly_sampled_df["fare_amount"]

# Scatter plot: Trip Distance vs Fare Amount
sns.scatterplot(x=trip_distance, y=fare_amount, alpha=0.5, color="blue")
plt.title("Relationship Between Trip Distance and Fare Amount")
plt.xlabel("Trip Distance (miles)")
plt.ylabel("Fare Amount ($)")

# Limit axes to remove extreme outliers (99th percentile cutoff)
plt.ylim(0, fare_amount.quantile(0.99))
plt.xlim(0, trip_distance.quantile(0.99))

plt.tight_layout()
plt.show()

# Calculate and print correlation
correlation = trip_distance.corr(fare_amount)
print(f"Correlation between Trip Distance and Fare Amount: {correlation:.2f}")
```




```
Correlation between Trip Distance and Fare Amount: 0.87
```

Correlation between trip_distance and fare_amount:

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

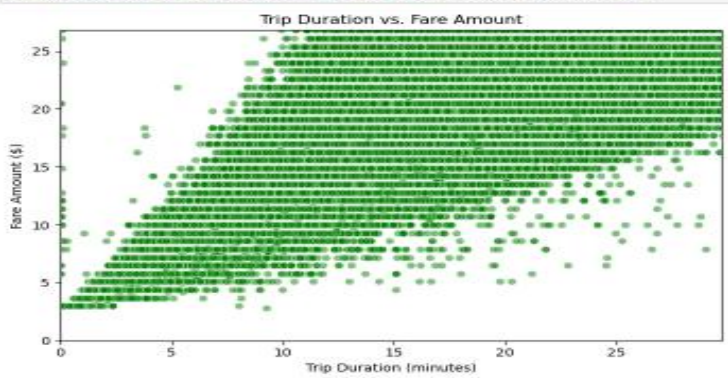
strong positive correlation (0.87). This means that as the trip distance increases, the fare amount also tends to increase proportionally.

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

Find and visualise the correlation between:

1. `fare_amount` and trip duration (pickup time to dropoff time)
2. `fare_amount` and `passenger_count`
3. `tip_amount` and `trip_distance`

```
# Show relationship between fare and trip duration

# Calculate trip duration in minutes
filtered_data["trip_duration"] = (
    filtered_data["tpep_dropoff_datetime"] - filtered_data["tpep_pickup_datetime"]
).dt.total_seconds() / 60

# Scatter plot: trip duration vs fare amount (with outlier limits)
plt.figure(figsize=(7, 5))
sns.scatterplot(
    x="trip_duration",
    y="fare_amount",
    data=filtered_data,
    alpha=0.5,
    color="green"
)
plt.title("Trip Duration vs. Fare Amount")
plt.xlabel("Trip Duration (minutes)")
plt.ylabel("Fare Amount ($)")
plt.ylim(0, filtered_data["fare_amount"].quantile(0.99))
plt.xlim(0, filtered_data["trip_duration"].quantile(0.99))
plt.tight_layout()
plt.show()

# Calculate and print correlation
corr_duration_fare = filtered_data["trip_duration"].corr(filtered_data["fare_amount"])
print(f"Correlation between trip duration and fare amount: {corr_duration_fare:.2f}")
```

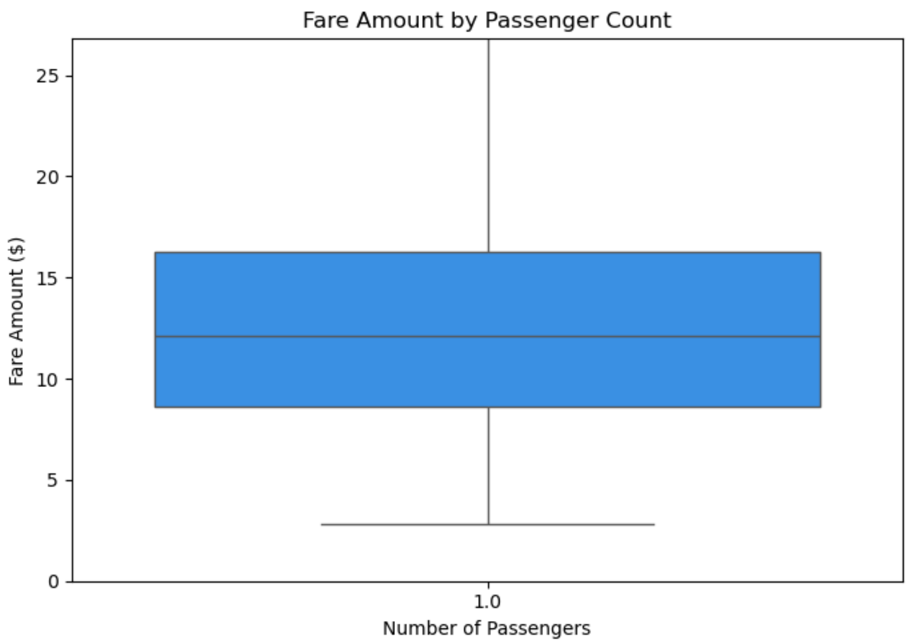



```
Correlation between trip duration and fare amount: 0.20
```

Observation: Trip duration is not directly proportional to fare amount. The correlation is also towards lower side (0.20). There are few outliers with high fare amount and low trip duration.

This could be due to surge pricing during peak hours or other factors such as negotiation between driver and passenger.

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

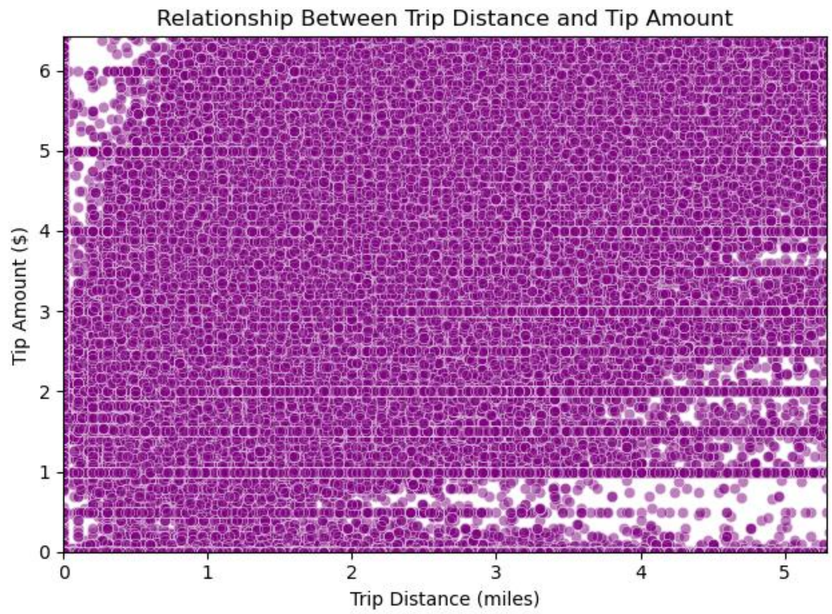



```
Average Fare Amount by Number of Passengers:
   passenger_count  fare_amount
0              1.0    12.860944
```

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

```
# Show relationship between fare and number of passengers

# Relationship between fare and passenger count

plt.figure(figsize=(7, 5))
sns.boxplot(
    x="passenger_count",
    y="fare_amount",
    data=filtered_data,
    color="dodgerblue"  # Use a single color for no warning
)
plt.title("Fare Amount by Passenger Count")
plt.xlabel("Number of Passengers")
plt.ylabel("Fare Amount ($)")
plt.ylim(0, filtered_data["fare_amount"].quantile(0.99))
plt.tight_layout()
plt.show()

# Calculate mean fare for each passenger group
fare_by_passenger = (
    filtered_data.groupby("passenger_count")["fare_amount"].mean().reset_index()
)
print("Average Fare Amount by Number of Passengers:\n", fare_by_passenger)
```

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS




```
Correlation between Trip Distance and Tip Amount: 0.48

Average Tip Per Mile:
 count    1.073899e+06
mean               inf
std                NaN
min       0.000000e+00
25%       1.371795e+00
50%       1.969388e+00
75%       2.729560e+00
max                inf
Name: tip_per_mile, dtype: float64
```

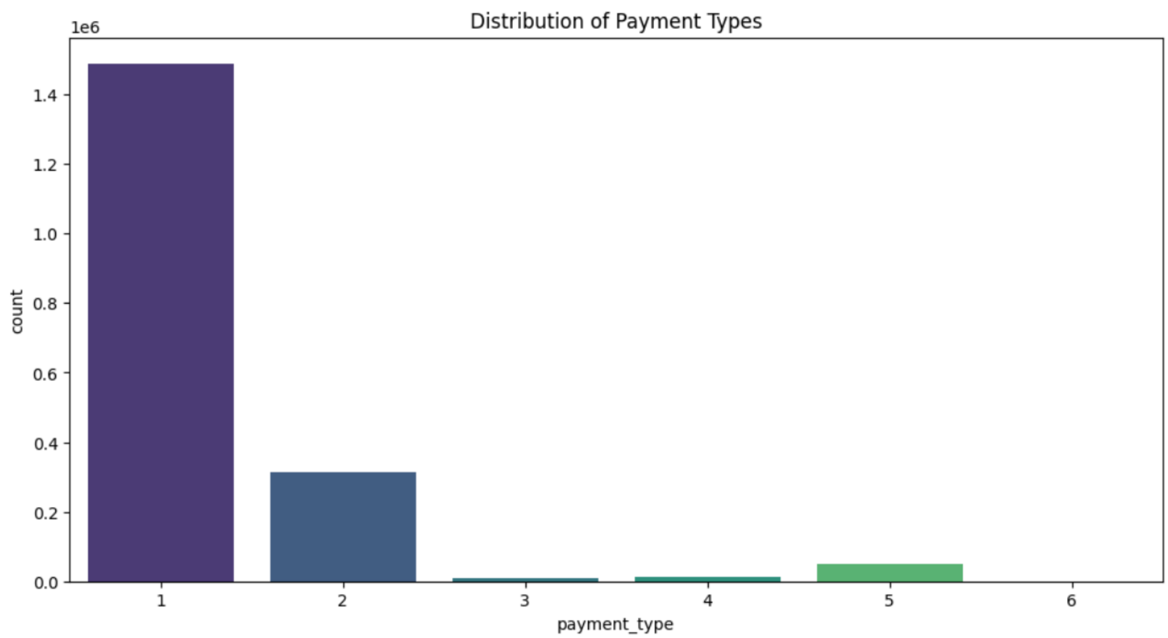
:

```
# Show relationship between tip and trip distance

# Extract relevant columns
trip_distance = hourly_sampled_df["trip_distance"]
tip_amount = hourly_sampled_df["tip_amount"]

# Scatter plot: Trip Distance vs Tip Amount
sns.scatterplot(x=trip_distance, y=tip_amount, alpha=0.5, color="purple")
plt.title("Relationship Between Trip Distance and Tip Amount")
plt.xlabel("Trip Distance (miles)")
plt.ylabel("Tip Amount ($)")

# Limit axes to exclude extreme outliers (99th percentile cutoff)
plt.ylim(0, tip_amount.quantile(0.99))
plt.xlim(0, trip_distance.quantile(0.99))

plt.tight_layout()
plt.show()

# Calculate correlation
correlation = trip_distance.corr(tip_amount)
print(f"Correlation between Trip Distance and Tip Amount: {correlation:.2f}")

# Calculate average tip per mile
hourly_sampled_df["tip_per_mile"] = tip_amount / trip_distance
print("\nAverage Tip Per Mile:\n", hourly_sampled_df["tip_per_mile"].describe())
```

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

Observation: The tip amount is not directly proportional to the trip distance. Tip amount for few short distances is higher than the tip amount for longer distances. This could be due to external factors such as time of the day, negotiation between driver and passenger.

1.1.3. Analyse the distribution of different payment types



Observation: The most common payment type is credit card (1), followed by cash (2). Number of voided trips (6) are zero.

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

1.1.4. Load the taxi zones shapefile and display it

Load the shapefile and display it.

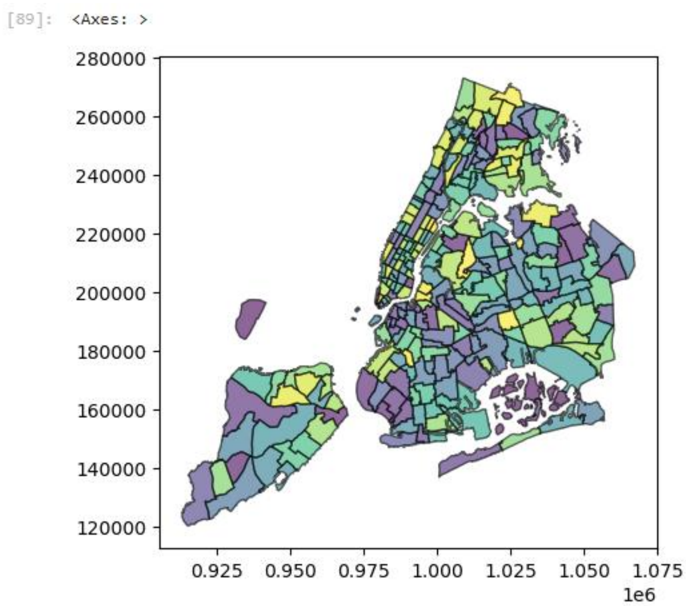
```
# import geopandas as gpd
# Read the shapefile using geopandas
import geopandas as gpd

# Define path to the shapefile (update if needed)
shapefile_path = r"C:\Users\prash\Downloads\Datasets and Dictionary-NYC\Datasets and Dictionary\taxi_zones\taxi_zones.shp"

# Load shapefile
zones = gpd.read_file(shapefile_path)

# Display first few rows
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...

```
print(zones.info())
zones.plot(edgecolor="black", cmap="viridis", alpha=0.6)
```

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

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS



1.1.4 Merge the zone data with trips data

Verified if 'PULocationId' column has any empty value, otherwise merging will result into missing zone info in those records.

Name- Prashant shrivastava

TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

```python
# Merge zones and trip records using LocationID and PULocationID

zones["LocationID"] = zones["LocationID"].astype(int)
hourly_sampled_df["PULocationID"] = hourly_sampled_df["PULocationID"].astype(int)

# Merge trip records with zone data
merged_df = hourly_sampled_df.merge(
    zones,
    left_on="PULocationID",
    right_on="LocationID",
    how="left"
)

# Display merged DataFrame
print(merged_df)
```

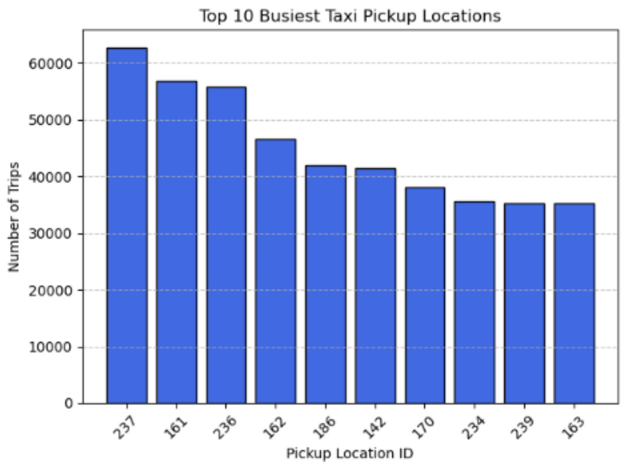
```
         VendorID tpep_pickup_datetime tpep_dropoff_datetime  passenger_count  \
0               2  2023-01-25 21:57:59   2023-01-25 22:00:33              1.0
1               1  2023-01-11 22:19:13   2023-01-11 22:32:37              1.0
2               2  2023-01-03 19:52:03   2023-01-03 19:55:24              1.0
3               2  2023-01-15 15:41:41   2023-01-15 15:54:03              1.0
4               2  2023-01-11 09:40:24   2023-01-11 09:54:18              1.0
...           ...                  ...                   ...              ...
1074049         2  2023-02-09 22:32:32   2023-02-09 22:53:32              1.0
1074050         2  2023-05-12 13:55:10   2023-05-12 14:02:10              1.0
1074051         1  2023-01-23 07:04:11   2023-01-23 07:08:24              1.0
1074052         2  2023-05-09 20:10:39   2023-05-09 20:31:23              1.0
1074053         2  2023-01-07 19:41:20   2023-01-07 19:53:44              1.0

         trip_distance  RatecodeID  PULocationID  DOLocationID  payment_type  \
0                 0.50         1.0           162           170             1
1                 2.80         1.0           164           231             1
2                 0.87         1.0           151           239             1
3                 1.86         1.0           211           234             1
4                 2.33         1.0           238           237             1
...                ...         ...           ...           ...           ...
1074049           5.35         1.0           113           151             1
1074050           0.94         1.0           107           113             1
1074051           0.90         1.0           186           234             1
1074052           4.30         1.0           114           262             1
1074053           2.33         1.0           239           237             1

         fare_amount  ...  pickup_day  pickup_month  tip_per_mile  OBJECTID  \
0                5.1  ...   Wednesday       January      4.040000     162.0
1               14.9  ...   Wednesday       January      1.421429     164.0
2                6.5  ...     Tuesday       January      1.149425     151.0
3               13.5  ...      Sunday       January      1.881720     211.0
4               15.6  ...   Wednesday       January      2.103004     238.0
...              ...  ...         ...           ...           ...       ...
1074049         25.4  ...    Thursday      February      1.136449     113.0
1074050          7.9  ...      Friday           May      2.531915     107.0
1074051          7.2  ...      Monday       January      2.444444     186.0
1074052         22.6  ...     Tuesday           May      1.283721     114.0
1074053         14.2  ...    Saturday       January      2.343348     239.0

         Shape_Leng  Shape_Area                         zone LocationID  \
0          0.035270    0.000048                 Midtown East      162.0
1          0.035772    0.000056                Midtown South      164.0
2          0.054890    0.000129             Manhattan Valley      151.0
3          0.025235    0.000040                         SoHo      211.0
4          0.060109    0.000185        Upper West Side North      238.0
...             ...         ...                          ...        ...
1074049    0.032745    0.000058      Greenwich Village North      113.0
1074050    0.038041    0.000075                     Gramercy      107.0
1074051    0.024696    0.000037  Penn Station/Madison Sq West      186.0
1074052    0.031727    0.000047      Greenwich Village South      114.0
1074053    0.063626    0.000205        Upper West Side South      239.0

           borough                                           geometry
```

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

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

```
# Group data by location and calculate the number of trips
import geopandas as gpd
import matplotlib.pyplot as plt

# Load taxi zones shapefile
shapefile_path = r"C:\Users\prash\Downloads\Datasets and Dictionary-NYC\Datasets and Dictionary\taxi_zones\taxi_zones.shp"
zones = gpd.read_file(shapefile_path)

# Count number of trips per pickup location
trip_counts = (
    hourly_sampled_df.groupby("PULocationID")
    .size()
    .reset_index(name="num_trips")
    .sort_values(by="num_trips", ascending=False)
)

# Merge trip counts with zone data
zones = zones.merge(
    trip_counts,
    left_on="LocationID",
    right_on="PULocationID",
    how="left"
)

# Replace NaN with 0 for zones with no trips
zones["num_trips"] = zones["num_trips"].fillna(0)

# Plot top 10 busiest pickup locations
top_pickups = trip_counts.head(10)

plt.bar(
    top_pickups["PULocationID"].astype(str),
    top_pickups["num_trips"],
    color="royalblue",
    edgecolor="black"
)
plt.xlabel("Pickup Location ID")
plt.ylabel("Number of Trips")
plt.title("Top 10 Busiest Taxi Pickup Locations")
plt.xticks(rotation=45)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.tight_layout()
plt.show()
```

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS



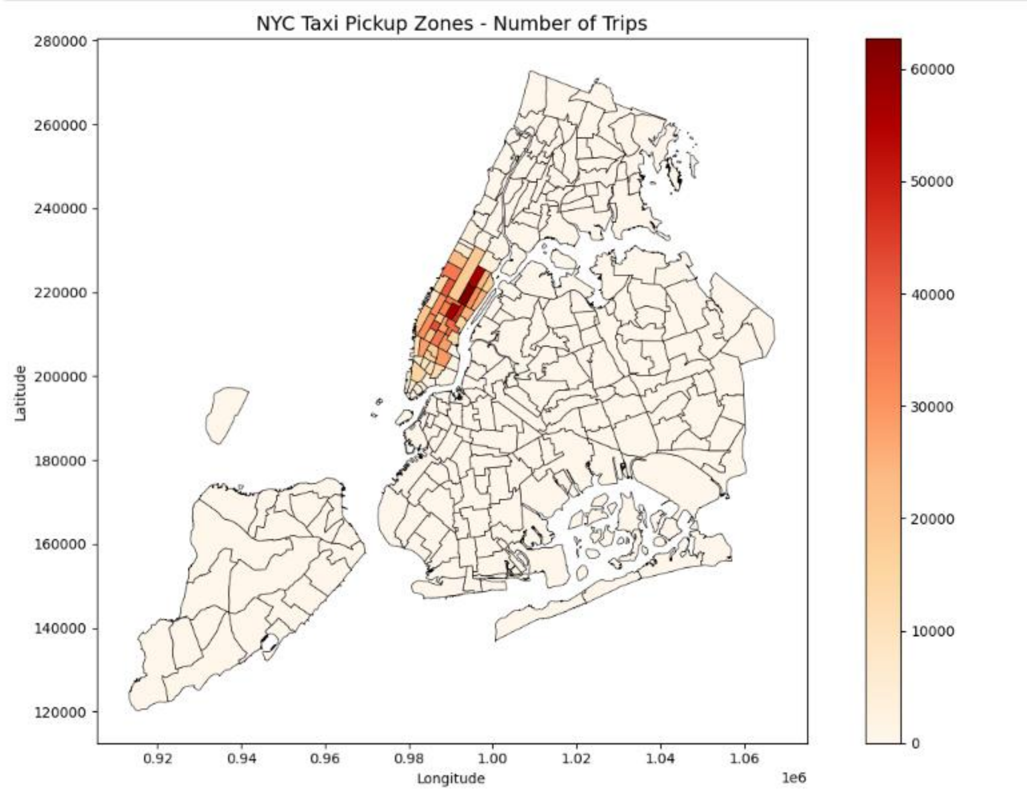
6.

7.

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

```
# Merge trip counts back to the zones GeoDataFrame

# Read the shapefile
shapefile_path = r"C:\Users\prash\Downloads\Datasets and Dictionary-NYC\Datasets and Dictionary\taxi_zones\taxi_zones.shp"
zones = gpd.read_file(shapefile_path)

# Aggregate trip counts by pickup location
trip_counts = (
    hourly_sampled_df.groupby("PULocationID")
    .size()
    .reset_index(name="num_trips")
)

# Merge trip counts back to the zones GeoDataFrame
zones = zones.merge(trip_counts, left_on="LocationID", right_on="PULocationID", how="left")

# Fill NaN values with 0 (for zones with no recorded trips)
zones["num_trips"] = zones["num_trips"].fillna(0)

# Plot the taxi zones with trip density
fig, ax = plt.subplots(figsize=(12, 8))
zones.plot(column="num_trips", cmap="OrRd", linewidth=0.5, edgecolor="black", legend=True, ax=ax)
plt.title("NYC Taxi Pickup Zones - Number of Trips", fontsize=14)
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.tight_layout()
plt.show()
```

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS



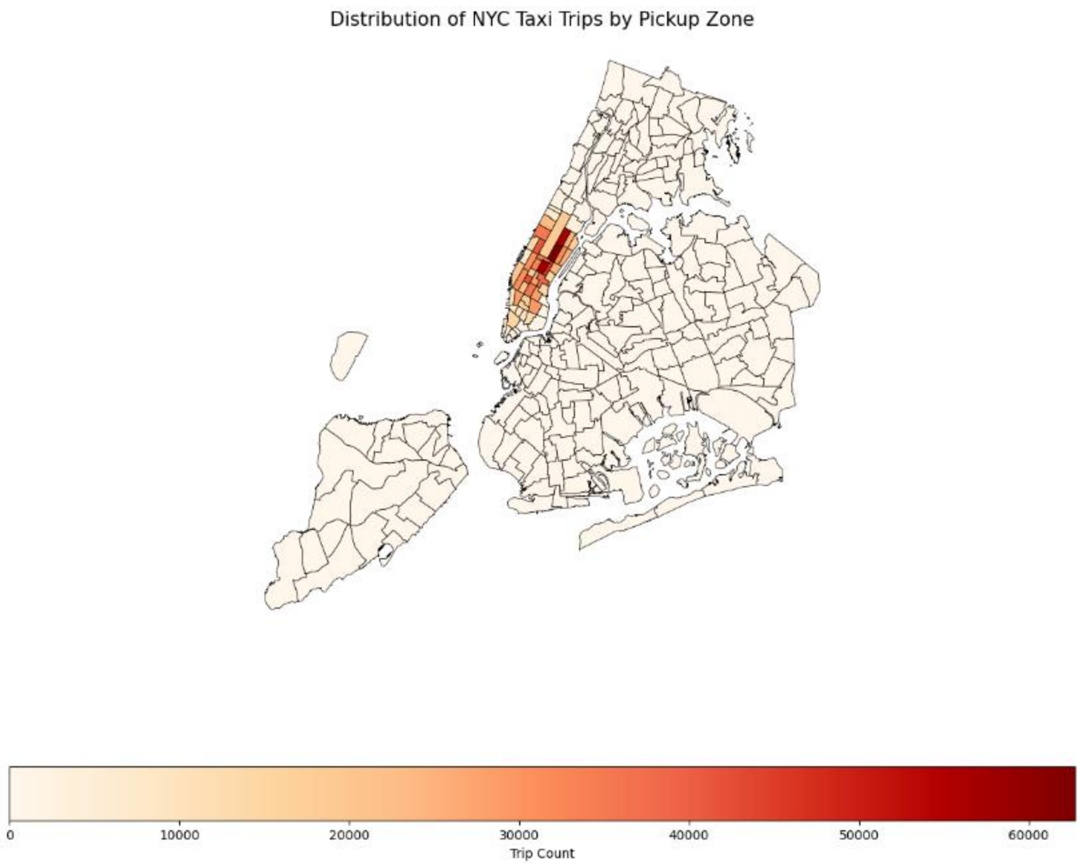
1.1.9. Plot a map of the zones showing number of trips

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

Plot a color-coded map showing zone-wise trips

```
# Define figure and axis

fig, ax = plt.subplots(figsize=(12, 10))

# Draw the taxi zones, shading by trip counts
zones.plot(
    column="num_trips",                # data column for color intensity
    cmap="OrRd",                        # color scheme (Orange → Red)
    linewidth=0.5,                      # boundary line width
    edgecolor="black",                  # boundary line color
    ax=ax,                              # use our axis
    legend=True,                        # include color legend
    legend_kwds={
        "label": "Trip Count",
        "orientation": "horizontal"
    }
)

# Add a descriptive title
ax.set_title("Distribution of NYC Taxi Trips by Pickup Zone", fontsize=15)

# Hide axis ticks and borders for a cleaner map look
ax.set_xticks([])
ax.set_yticks([])
ax.spines.clear()

# Render the final map
plt.tight_layout()
plt.show()
```

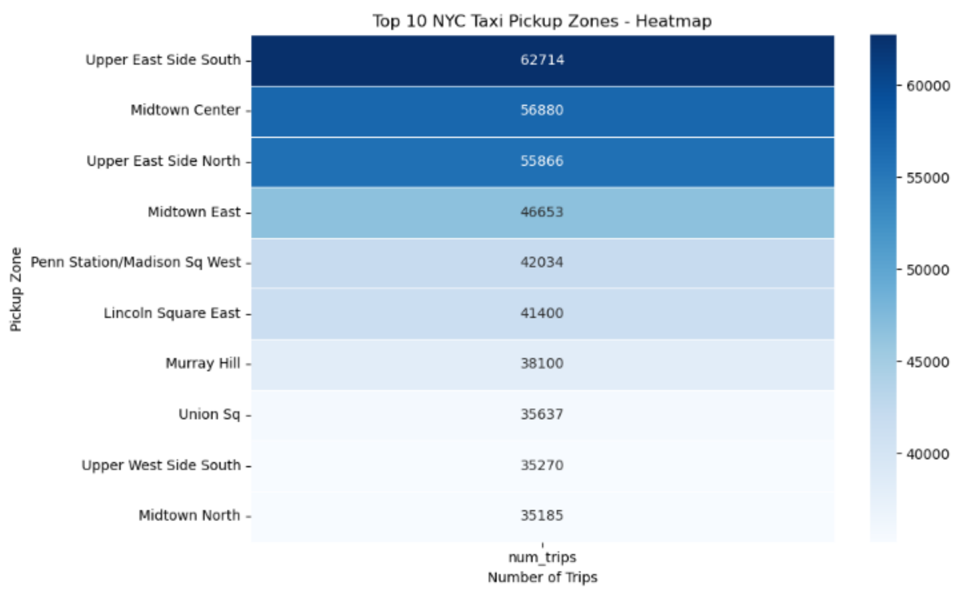
Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS



Name- Prashant shrivastava

TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

```python
# can you try displaying the zones DF sorted by the number of trips?
import geopandas as gpd
import matplotlib.pyplot as plt
import seaborn as sns

# Grab the 10 busiest pickup zones
top10 = zones.nlargest(10, "num_trips").copy()

# Prepare a single-column table for the heatmap
heat_df = (
    top10[["zone", "num_trips"]]
    .set_index("zone")
    .sort_values("num_trips", ascending=False)
)

# Draw heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(
    heat_df,
    annot=True,
    cmap="Blues",
    linewidths=0.5,
    fmt=".0f"
)

# Titles and labels
plt.title("Top 10 NYC Taxi Pickup Zones - Heatmap")
plt.xlabel("Number of Trips")
plt.ylabel("Pickup Zone")

plt.tight_layout()
plt.show()
```



Here we have completed the temporal, financial and geographical analysis on the trip records.

- **Conclude with results**

1. **Taxi service usage:** The peak hours for taxi pickups is from 5:00 pm - 7:00 pm. The reason being people avail taxi at this time to travel from office to home during weekdays, and go out for dinner during weekends.

   Weekdays, especially Wednesday, Thursday and Friday, have higher taxi pickups/drop-offs. Possible reason being most people work from office during mid-week.

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

Spring Break (March), Summer vacation (May) and Holiday season (Oct-Dec) see the highest taxi service usage.

2. **Trends in revenue collection trend:**

Revenue collection is at its highest in Q2 (April, May, June) and Q4 (Oct, Nov, Dec), aligning with the higher taxi activity during these periods, aligning with the higher taxi service usage during these periods, with Q4 being the peak due to the holiday season.

Taxi fare:

Fare amount vs Trip distance: There is a strong positive correlation between trip distance and fare amount. Longer trips result in higher fares.

- Fare amount vs Trip duration: There is also a positive correlation between trip duration and fare amount, although the correlation is less compared to above. In some instances, shorter trips have resulted in higher fare amount, indicating surge pricing and/or negotiation between driver and passenger.

- Fare amount vs Passenger count: Passenger count = 4 usually results in highest fare, however more than 4 passengers have shown lower fare trend.

- Tip amount vs Trip distance: The correlation is positive, however as per the visualization, shorter trips also resulted in higher tip amount. Possible reason being time of the day/night and negotiation between passenger and driver.

Busiest Zones:

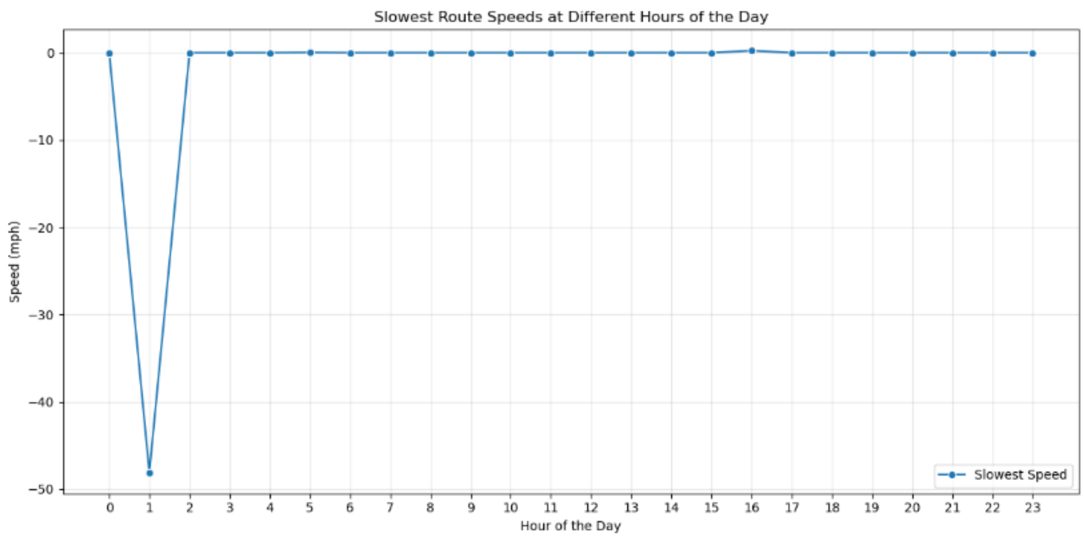
- Top pickup location is JFK Airport, followed closely by Upper East Side South, Midtown canter and Upper East Side North.

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

1.2. Detailed EDA: Insights and Strategies

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

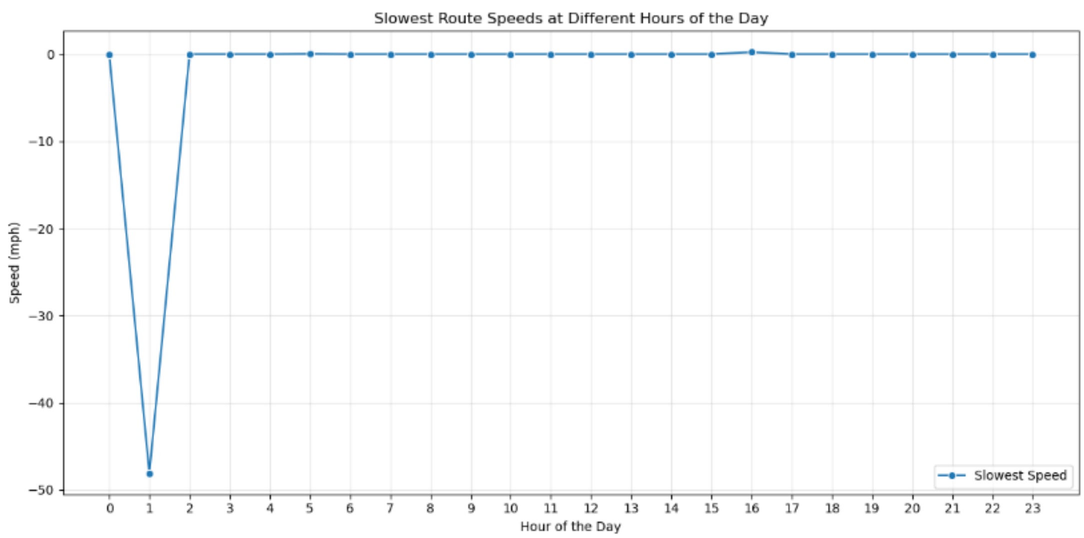
Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS




```
Rows in slowest_routes: 24
   pickup_hour  PULocationID  DOLocationID  avg_duration  avg_distance  \
0            0           202           112     12.233333      0.000000
1            1           164           224     -1.819444      1.458333
2            2           125           125      1.466667      0.000000
3            3            13            13      0.116667      0.000000
4            4            79            41     17.700000      0.000000

   speed_mph
0   0.000000
1 -48.091603
2   0.000000
3   0.000000
4   0.000000
```

```
)

# ---- 4) Plot ----
plt.figure(figsize=(12, 6))
sns.lineplot(data=slowest_routes, x="pickup_hour", y="speed_mph", marker="o", label="Slowest Speed")
plt.xlabel("Hour of the Day")
plt.ylabel("Speed (mph)")
plt.title("Slowest Route Speeds at Different Hours of the Day")
plt.xticks(range(24))
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.legend()
plt.show()

# Optional: quick check so you can compare runs
print("Rows in slowest_routes:", len(slowest_routes))
print(slowest_routes.head(5))
```

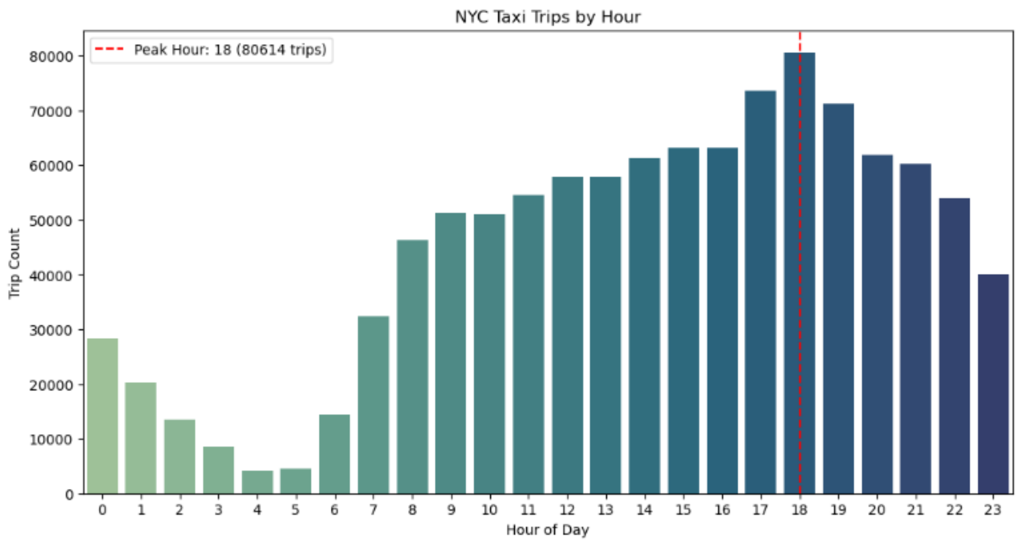
Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS




```
Rows in slowest_routes: 24
   pickup_hour  PULocationID  DOLocationID  avg_duration  avg_distance  \
0            0           202           112     12.233333      0.000000
1            1           164           224     -1.819444      1.458333
2            2           125           125      1.466667      0.000000
3            3            13            13      0.116667      0.000000
4            4            79            41     17.700000      0.000000

   speed_mph
0   0.000000
1 -48.091603
2   0.000000
3   0.000000
4   0.000000
```

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

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS




```
Peak hour is 18:00 with 80614 trips.
```

Observations: Busy hours are during evening 5pm to 7pm, indicating people returning to home

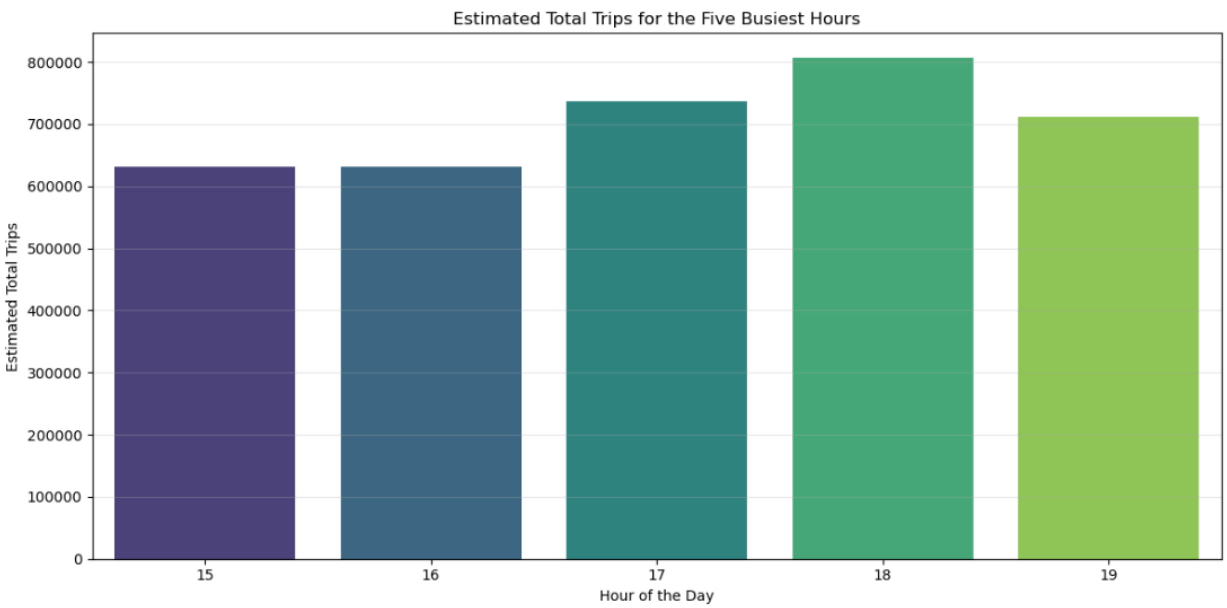
1.2.3. Scale up the number of trips from above to find the actual number of trips

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

```
# Scale up the number of trips
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 1) Hour feature
df["tpep_pickup_datetime"] = pd.to_datetime(df["tpep_pickup_datetime"], errors="coerce")
df["pickup_hour"] = df["tpep_pickup_datetime"].dt.hour

# 2) Count trips per hour in the sampled data
trips_per_hour = df["pickup_hour"].value_counts().sort_index()

# 3) Scale up to estimate totals (set your actual sampling fraction)
sample_fraction = 0.10  # e.g., 10% sample -> multiply by 10
scaled_trips = (trips_per_hour / sample_fraction).round().astype(int)

# 4) Top-5 busiest hours (actual estimated counts)
top5 = scaled_trips.sort_values(ascending=False).head(5)

# 5) Show results
print("Estimated total trips in the five busiest hours:")
for hr, cnt in top5.items():
    print(f"{hr:02d}:00  ->  {cnt:,} trips")

# 6) Visualize
plt.figure(figsize=(12, 6))
sns.barplot(x=top5.index, y=top5.values, palette="viridis")
plt.title("Estimated Total Trips for the Five Busiest Hours")
plt.xlabel("Hour of the Day")
plt.ylabel("Estimated Total Trips")
plt.grid(axis="y", alpha=0.3)
plt.tight_layout()
plt.show()
```

```
Estimated total trips in the five busiest hours:
18:00  ->  806,140 trips
17:00  ->  736,520 trips
19:00  ->  711,430 trips
16:00  ->  632,060 trips
15:00  ->  631,440 trips
```

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

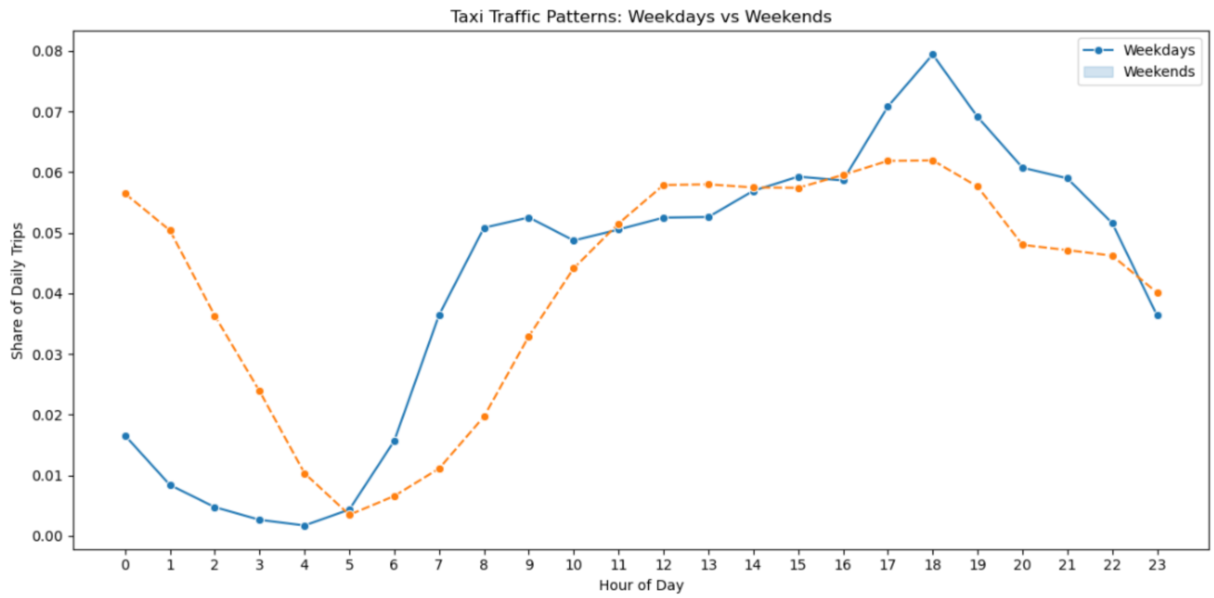


1.2.4. Compare hourly traffic on weekdays and weekends

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Create hour and weekday columns
df["tpep_pickup_datetime"] = pd.to_datetime(df["tpep_pickup_datetime"], errors="coerce")
df["pickup_hour"] = df["tpep_pickup_datetime"].dt.hour
df["day_of_week"] = df["tpep_pickup_datetime"].dt.weekday  # Monday=0, Sunday=6

# Tag weekend trips
df["weekend_flag"] = df["day_of_week"] >= 5

# Count trips per hour for weekday/weekend
traffic_counts = (
    df.groupby(["pickup_hour", "weekend_flag"])
      .size()
      .reset_index(name="trip_count")
)

# Reshape for easier plotting
traffic_pivot = traffic_counts.pivot(index="pickup_hour", columns="weekend_flag", values="trip_count")

# Rename columns for clarity
traffic_pivot.columns = ["Weekdays", "Weekends"]

# Convert to proportions for fair comparison
traffic_pivot = traffic_pivot.div(traffic_pivot.sum())

# Plot comparison
plt.figure(figsize=(12, 6))
sns.lineplot(data=traffic_pivot, marker="o")
plt.xlabel("Hour of Day")
plt.ylabel("Share of Daily Trips")
plt.title("Taxi Traffic Patterns: Weekdays vs Weekends")
plt.xticks(range(24))
plt.legend(["Weekdays", "Weekends"])
plt.tight_layout()
plt.show()
```

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS



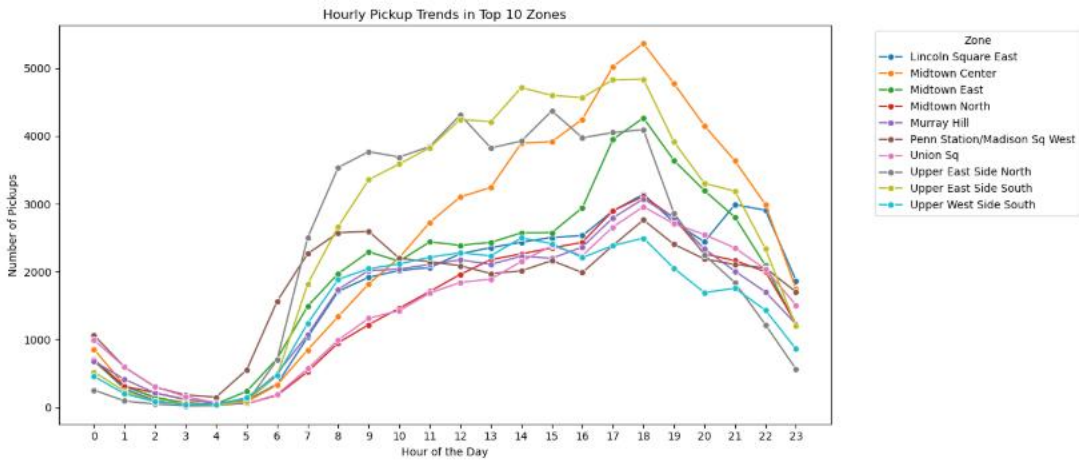
Observation: Weekday trips tend to go higher during evening hours, while weekends mostly see a flat trend in number of trips throughout the day

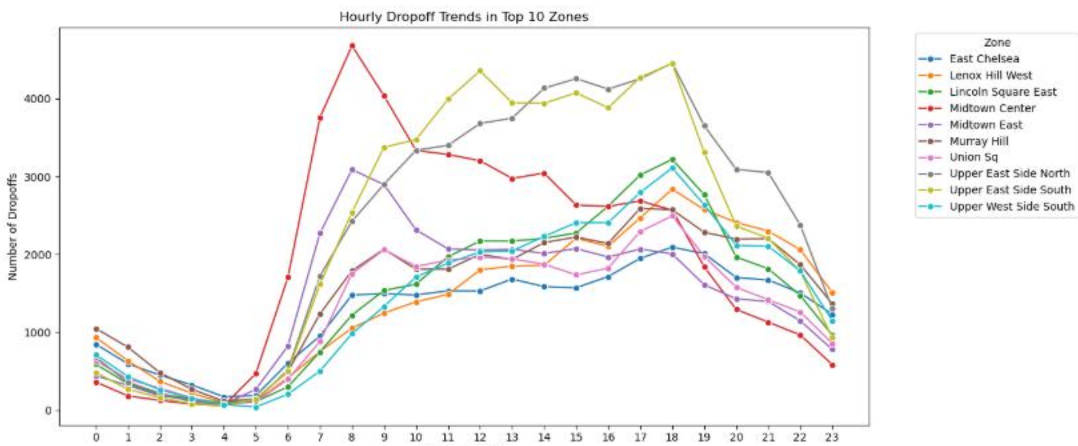
Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

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

```
# Find top 10 pickup and dropoff zones

import matplotlib.pyplot as plt

# Ensure datetime is parsed
df["tpep_pickup_datetime"] = pd.to_datetime(df["tpep_pickup_datetime"], errors="coerce")
df["pickup_hour"] = df["tpep_pickup_datetime"].dt.hour

# ---- Top 10 pickup zones ----
pickup_counts = df["PULocationID"].value_counts().head(10).index
top_pickup_df = df[df["PULocationID"].isin(pickup_counts)]

# ---- Top 10 dropoff zones ----
dropoff_counts = df["DOLocationID"].value_counts().head(10).index
top_dropoff_df = df[df["DOLocationID"].isin(dropoff_counts)]

# ---- Merge zone names ----
top_pickup_df = top_pickup_df.merge(zones[["LocationID", "zone"]], left_on="PULocationID", right_on="LocationID", how="left")
top_dropoff_df = top_dropoff_df.merge(zones[["LocationID", "zone"]], left_on="DOLocationID", right_on="LocationID", how="left")

# ---- Hourly pickup trends for top zones ----
pickup_trends = (
    top_pickup_df.groupby(["pickup_hour", "zone"])
    .size()
    .reset_index(name="trip_count")
)

plt.figure(figsize=(14, 6))
sns.lineplot(data=pickup_trends, x="pickup_hour", y="trip_count", hue="zone", marker="o")
plt.title("Hourly Pickup Trends in Top 10 Zones")
plt.xlabel("Hour of the Day")
plt.ylabel("Number of Pickups")
plt.xticks(range(24))
plt.legend(title="Zone", bbox_to_anchor=(1.05, 1), loc="upper left")
plt.tight_layout()
plt.show()

# ---- Hourly dropoff trends for top zones ----
dropoff_trends = (
    top_dropoff_df.groupby([top_dropoff_df["tpep_pickup_datetime"].dt.hour, "zone"])
    .size()
    .reset_index(name="trip_count")
    .rename(columns={"tpep_pickup_datetime": "pickup_hour"})
)

plt.figure(figsize=(14, 6))
sns.lineplot(data=dropoff_trends, x="pickup_hour", y="trip_count", hue="zone", marker="o")
plt.title("Hourly Dropoff Trends in Top 10 Zones")
plt.xlabel("Hour of the Day")
plt.ylabel("Number of Dropoffs")
plt.xticks(range(24))
plt.legend(title="Zone", bbox_to_anchor=(1.05, 1), loc="upper left")
plt.tight_layout()
plt.show()
```

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS





Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

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

```python
# Find the top 10 and bottom 10 pickup/dropoff ratios
import numpy as np
import pandas as pd

# Count pickup and dropoff trips per LocationID
pu = df["PULocationID"].value_counts().rename("num_pickups")
do = df["DOLocationID"].value_counts().rename("num_dropoffs")

# Combine on the full set of LocationIDs (pickups U dropoffs)
ratios = (
    pd.concat([pu, do], axis=1)
      .fillna(0)
      .reset_index()
      .rename(columns={"index": "LocationID"})
)

# Safe ratio: undefined when dropoffs = 0
ratios["pickup_dropoff_ratio"] = ratios["num_pickups"] / ratios["num_dropoffs"].replace(0, np.nan)

# Attach zone names
ratios = ratios.merge(zones[["LocationID", "zone"]], on="LocationID", how="left")

# Drop zones where ratio is undefined (no dropoffs)
ratios_valid = ratios.dropna(subset=["pickup_dropoff_ratio"]).copy()

# Sort for top/bottom 10
top_10 = ratios_valid.sort_values("pickup_dropoff_ratio", ascending=False).head(10)
bottom_10 = ratios_valid.sort_values("pickup_dropoff_ratio", ascending=True).head(10)

# Display (rounded for readability)
print("Top 10 Pickup/Dropoff Ratios:")
print(top_10[["zone", "pickup_dropoff_ratio"]].assign(pickup_dropoff_ratio=lambda s: s["pickup_dropoff_ratio"].round(3)))

print("\nBottom 10 Pickup/Drop
print(bottom_10[["zone", "pic                                                :kup_dropoff_ratio"].round(3)))
```

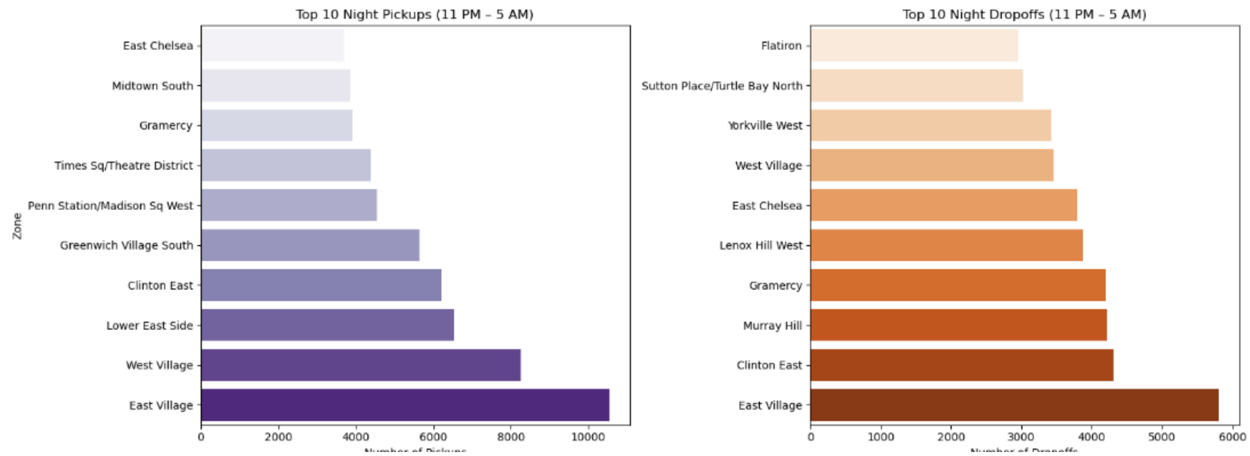
```
Top 10 Pickup/Dropoff Ratios:
                         zone  pickup_dropoff_ratio
77              East Elmhurst                 6.000
4   Penn Station/Madison Sq West              1.609
28     Greenwich Village South                1.362
24               Central Park                 1.351
3                Midtown East                 1.315
15               West Village                 1.305
25           Garment District                 1.284
9               Midtown North                 1.229
10     Times Sq/Theatre District              1.202
1              Midtown Center                 1.195

Bottom 10 Pickup/Dropoff Ratios:
                        zone  pickup_dropoff_ratio
154       Ocean Parkway South                   0.0
153                  Glendale                   0.0
137              Baisley Park                   0.0
112        Stuyvesant Heights                   0.0
113        South Williamsburg                   0.0
114            Columbia Street                  0.0
115   Prospect-Lefferts Gardens                 0.0
116        Crown Heights South                  0.0
117             Prospect Park                   0.0
118                 Ridgewood                   0.0
```

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

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

```python
# During night hours (11pm to 5am) find the top 10 pickup and dropoff zones
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Make sure we have the hour field
df["tpep_pickup_datetime"] = pd.to_datetime(df["tpep_pickup_datetime"], errors="coerce")
df["pickup_hour"] = df["tpep_pickup_datetime"].dt.hour

# Keep only night trips: 23, 0-5
night_mask = (df["pickup_hour"] >= 23) | (df["pickup_hour"] <= 5)
night_df = df.loc[night_mask].copy()

# Top 10 night pickups
night_top_pu = (
    night_df["PULocationID"].value_counts().head(10).rename_axis("LocationID").reset_index(name="num_pickups")
    .merge(zones[["LocationID", "zone"]], on="LocationID", how="left")
    .sort_values("num_pickups", ascending=True)  # for a neat horizontal bar order
)

# Top 10 night dropoffs
night_top_do = (
    night_df["DOLocationID"].value_counts().head(10).rename_axis("LocationID").reset_index(name="num_dropoffs")
    .merge(zones[["LocationID", "zone"]], on="LocationID", how="left")
    .sort_values("num_dropoffs", ascending=True)
)

# Plot side-by-side (horizontal bars for readability)
fig, axes = plt.subplots(1, 2, figsize=(16, 6), sharey=False)

sns.barplot(data=night_top_pu, x="num_pickups", y="zone", ax=axes[0], palette="Purples")
axes[0].set_title("Top 10 Night Pickups (11 PM - 5 AM)")
axes[0].set_xlabel("Number of Pickups")
axes[0].set_ylabel("Zone")

sns.barplot(data=night_top_do, x="num_dropoffs", y="zone", ax=axes[1], palette="Oranges")
axes[1].set_title("Top 10 Night Dropoffs (11 PM - 5 AM)")
axes[1].set_xlabel("Number of Dropoffs")
axes[1].set_ylabel("")

plt.tight_layout()
plt.show()

# (Optional) quick text summary
print("Top 10 night pickup zones:")
print(night_top_pu.sort_values("num_pickups", ascending=False)[["zone", "num_pickups"]].to_string(index=False))

print("\nTop 10 night dropoff zones:")
print(night_top_do.sort_values("num_dropoffs", ascending=False)[["zone", "num_dropoffs"]].to_string(index=False))
```

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS



Observation: People tend to avail taxi service more during night hours from airports, with some pick-up locations residential areas like East village. Drop-off locations are all mostly residential during night, with exception of times square.

```
Top 10 night pickup zones:
                       zone  num_pickups
               East Village        10540
               West Village         8264
            Lower East Side         6534
               Clinton East         6213
    Greenwich Village South         5641
Penn Station/Madison Sq West        4548
  Times Sq/Theatre District         4384
                   Gramercy         3910
              Midtown South         3865
               East Chelsea         3691

Top 10 night dropoff zones:
                        zone  num_dropoffs
                East Village          5802
                Clinton East          4311
                 Murray Hill          4223
                    Gramercy          4202
             Lenox Hill West          3868
                East Chelsea          3799
                West Village          3459
              Yorkville West          3422
Sutton Place/Turtle Bay North         3028
                    Flatiron          2955
```

1.2.8. Find the revenue share for nighttime and daytime hours

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

Find the revenue share for nighttime and daytime hours.

```
# Filter for night hours (11 PM to 5 AM)
# Ensure datetime parsing
df["tpep_pickup_datetime"] = pd.to_datetime(df["tpep_pickup_datetime"], errors="coerce")
df["pickup_hour"] = df["tpep_pickup_datetime"].dt.hour

# Night: 11 PM-5 AM | Day: 6 AM-10 PM
night_mask = (df["pickup_hour"] >= 23) | (df["pickup_hour"] <= 5)
day_mask = (df["pickup_hour"] > 5) & (df["pickup_hour"] < 23)

# Revenue totals
night_rev = df.loc[night_mask, "total_amount"].sum()
day_rev   = df.loc[day_mask, "total_amount"].sum()
total_rev = night_rev + day_rev

# Revenue share (%)
night_pct = (night_rev / total_rev) * 100
day_pct   = (day_rev / total_rev) * 100

print(f"Night Time Revenue Share : {night_pct:.2f}%")
print(f"Day Time Revenue Share   : {day_pct:.2f}%")
```

```
Night Time Revenue Share : 11.20%
Day Time Revenue Share   : 88.80%
```

Observation: Revenue tends to be highest during early morning (5am), and evening hours. The explanation for night time revenue can be higher tip amount given to driver.

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

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

```
# Analyse the fare per mile per passenger for different passenger counts

# Filter out invalid distances and passenger counts
filtered_df = df[(df["trip_distance"] > 0) & (df["passenger_count"] > 0)].copy()

# Step 1: Compute fare per mile for each trip
filtered_df["fare_per_mile"] = filtered_df["fare_amount"] / filtered_df["trip_distance"]

# Step 2: Adjust for passenger count to get fare per mile per passenger
filtered_df["fare_per_mile_per_passenger"] = filtered_df["fare_per_mile"] / filtered_df["passenger_count"]

# Step 3: Average the results for each passenger count
avg_fare_passenger = (
    filtered_df.groupby("passenger_count")["fare_per_mile_per_passenger"]
    .mean()
    .reset_index()
)

# Display the analysis
print("Average Fare per Mile per Passenger for each Passenger Count:")
print(avg_fare_passenger)
```

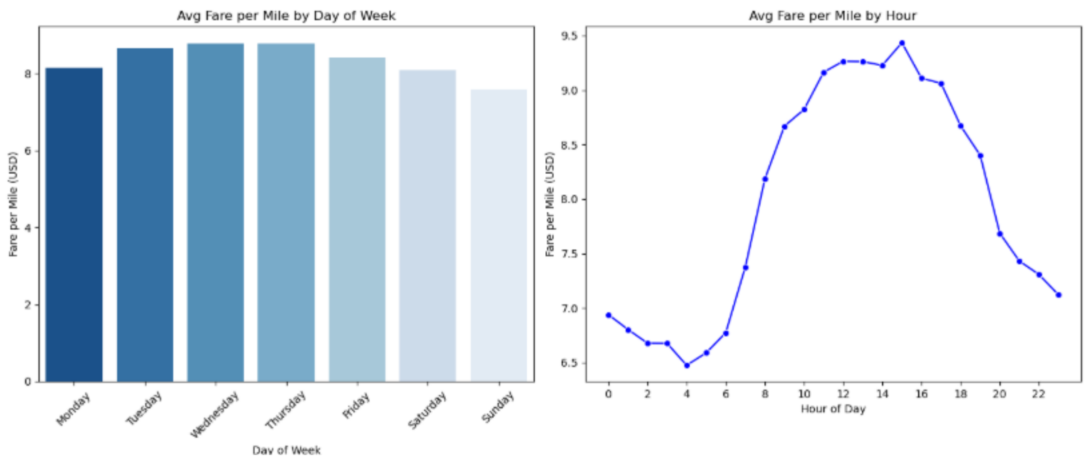
```
Average Fare per Mile per Passenger for each Passenger Count:
   passenger_count  fare_per_mile_per_passenger
0              1.0                     8.401935
```

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

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

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Keep only trips with valid distance
valid_trips = df[df["trip_distance"] > 0].copy()

# Calculate fare per mile
valid_trips["fare_per_mile"] = valid_trips["fare_amount"] / valid_trips["trip_distance"]

# Extract day name and pickup hour
valid_trips["pickup_day"] = pd.to_datetime(valid_trips["tpep_pickup_datetime"]).dt.day_name()
valid_trips["pickup_hour"] = pd.to_datetime(valid_trips["tpep_pickup_datetime"]).dt.hour

# Average fare per mile by day of week
avg_fare_day = (
    valid_trips.groupby("pickup_day")["fare_per_mile"]
    .mean()
    .reindex(["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"])
)

# Average fare per mile by hour
avg_fare_hour = valid_trips.groupby("pickup_hour")["fare_per_mile"].mean()

# --- Visualization ---
fig, axes = plt.subplots(1, 2, figsize=(14, 6))

# Plot by day of week
sns.barplot(ax=axes[0], x=avg_fare_day.index, y=avg_fare_day.values, palette="Blues_r")
axes[0].set_title("Avg Fare per Mile by Day of Week")
axes[0].set_xlabel("Day of Week")
axes[0].set_ylabel("Fare per Mile (USD)")
axes[0].tick_params(axis='x', rotation=45)

# Plot by hour of day
sns.lineplot(ax=axes[1], x=avg_fare_hour.index, y=avg_fare_hour.values, marker="o", color="blue")
axes[1].set_title("Avg Fare per Mile by Hour")
axes[1].set_xlabel("Hour of Day")
axes[1].set_ylabel("Fare per Mile (USD)")
axes[1].set_xticks(range(0, 24, 2))

plt.tight_layout()
plt.show()
```

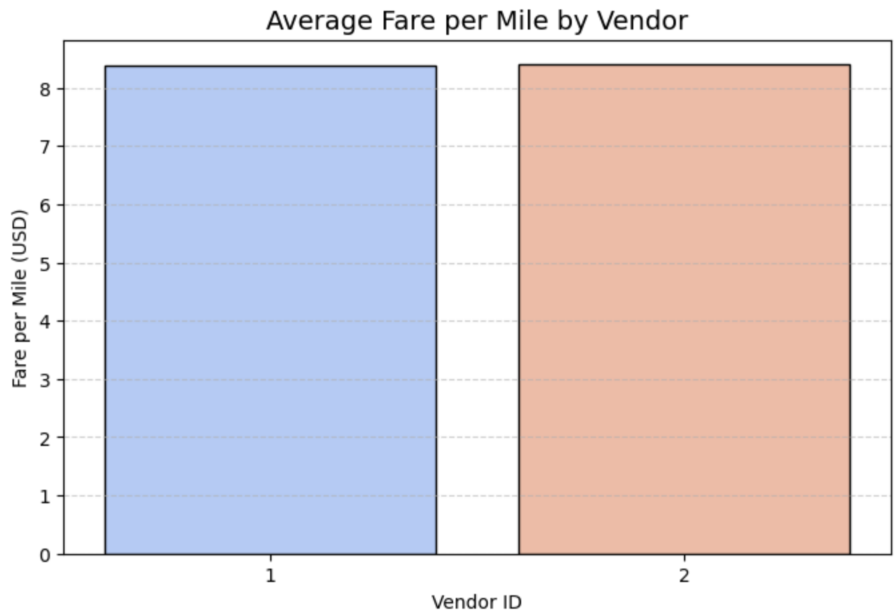
Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS



1.2.11. Analyze the average fare per mile for the different vendors

```
# Compare fare per mile for different vendors

# Ensure trip_distance > 0 to avoid division by zero
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Filter out trips with zero or negative distance
vendor_data = df[df["trip_distance"] > 0].copy()

# Calculate fare per mile
vendor_data["fare_per_mile"] = vendor_data["fare_amount"] / vendor_data["trip_distance"]

# Get average fare per mile for each vendor
avg_fare_vendor = vendor_data.groupby("VendorID")["fare_per_mile"].mean().reset_index()

# Change VendorID to string for categorical plotting
avg_fare_vendor["VendorID"] = avg_fare_vendor["VendorID"].astype(str)

# --- Plot ---
plt.figure(figsize=(8, 5))
sns.barplot(data=avg_fare_vendor, x="VendorID", y="fare_per_mile", palette="coolwarm", edgecolor="black")
plt.title("Average Fare per Mile by Vendor", fontsize=14)
plt.xlabel("Vendor ID")
plt.ylabel("Fare per Mile (USD)")
plt.grid(axis="y", linestyle="--", alpha=0.6)
plt.show()
```

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

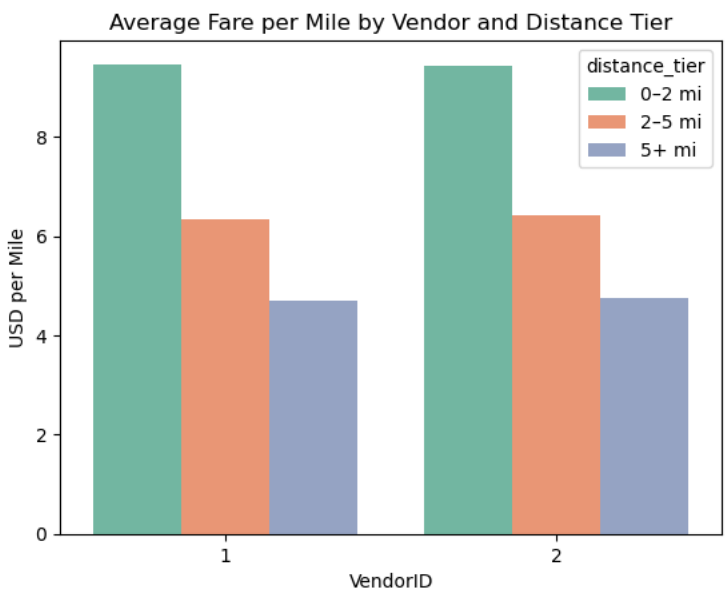


1.2.12. Compare the fare rates of different vendors in a distance-tiered fashion

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

```
# Defining distance tiers

## Keep valid trips and compute fare per mile
df = df[(df["trip_distance"] > 0) & (df["fare_amount"] > 0)].copy()
df["fare_per_mile"] = df["fare_amount"] / df["trip_distance"]

# Define tiers: 0-2, 2-5, >5 miles
df["distance_tier"] = pd.cut(df["trip_distance"], [0, 2, 5, float("inf")],
                              labels=["0-2 mi", "2-5 mi", "5+ mi"], right=False)

# Average fare per mile for each Vendor & Tier
result = df.groupby(["VendorID", "distance_tier"])["fare_per_mile"].mean().reset_index()

# Show in table form
print(result.pivot(index="VendorID", columns="distance_tier", values="fare_per_mile").round(2))

# Plot grouped bar chart
sns.barplot(data=result, x="VendorID", y="fare_per_mile", hue="distance_tier", palette="Set2")
plt.title("Average Fare per Mile by Vendor and Distance Tier")
plt.ylabel("USD per Mile")
plt.show()
```

```
distance_tier  0-2 mi  2-5 mi  5+ mi
VendorID
1                9.47    6.33   4.69
2                9.44    6.43   4.74
```

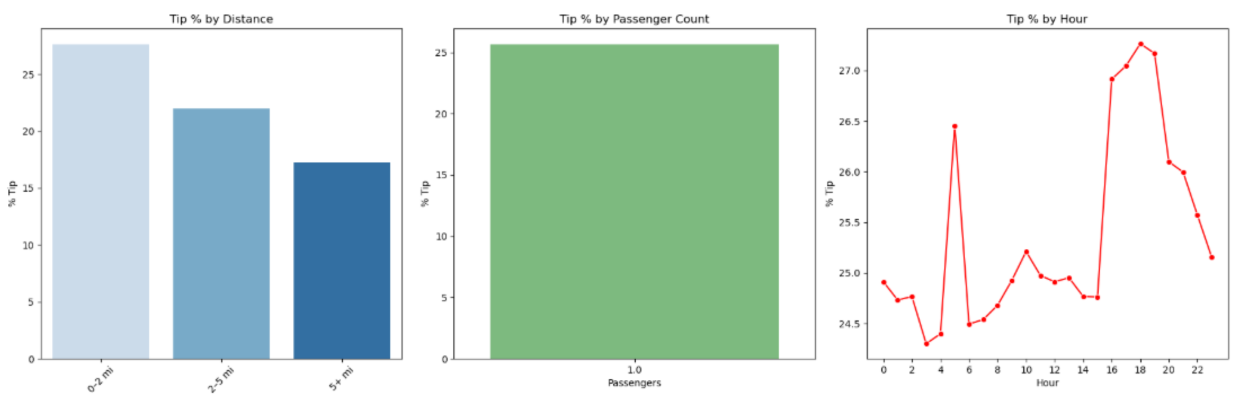


1.2.13. Analyse the tip percentages

Name- Prashant shrivastava

TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

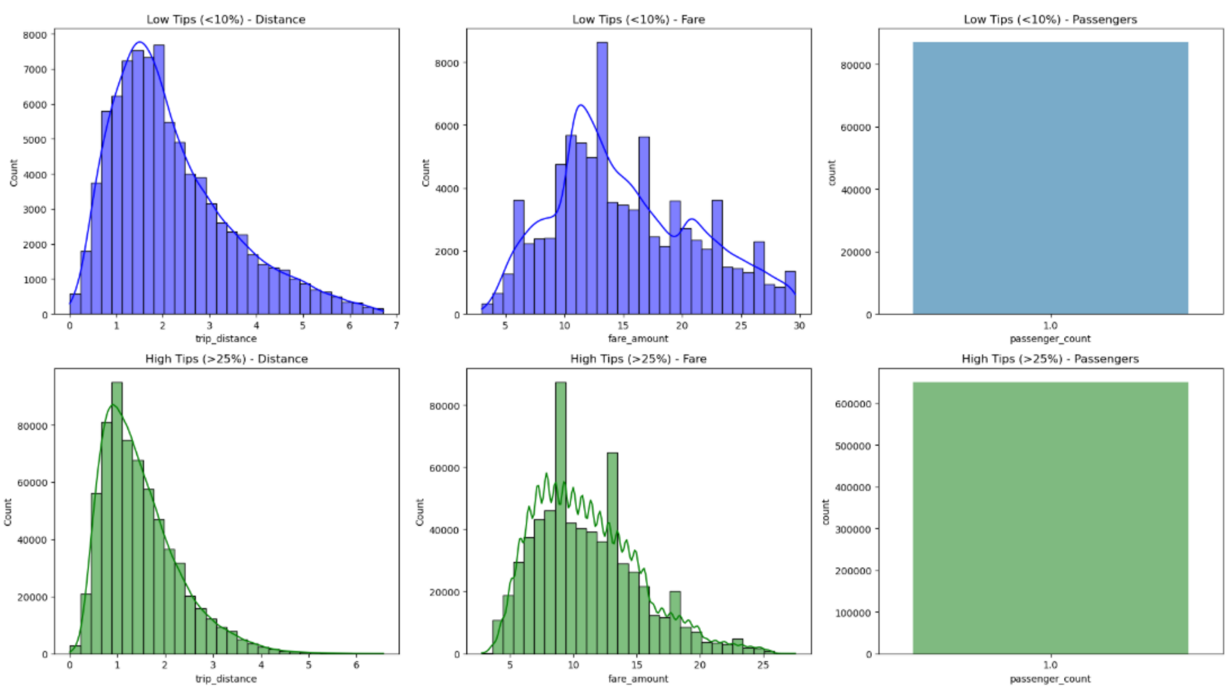
```python
#  Analyze tip percentages based on distances, passenger counts and pickup times

# Keep only valid fares
df = df[df["fare_amount"] > 0].copy()

# Tip percentage
df["tip_percent"] = (df["tip_amount"] / df["fare_amount"]) * 100
df["pickup_hour"] = pd.to_datetime(df["tpep_pickup_datetime"]).dt.hour

# Group averages
tip_by_distance = df.groupby("distance_tier")["tip_percent"].mean().reset_index()
tip_by_passengers = df.groupby("passenger_count")["tip_percent"].mean().reset_index()
tip_by_hour = df.groupby("pickup_hour")["tip_percent"].mean().reset_index()

# Plot
fig, axs = plt.subplots(1, 3, figsize=(18, 6))

sns.barplot(data=tip_by_distance, x="distance_tier", y="tip_percent", ax=axs[0], palette="Blues")
axs[0].set(title="Tip % by Distance", xlabel="Distance Tier", ylabel="% Tip")
axs[0].tick_params(axis="x", rotation=45)

sns.barplot(data=tip_by_passengers, x="passenger_count", y="tip_percent", ax=axs[1], palette="Greens")
axs[1].set(title="Tip % by Passenger Count", xlabel="Passengers", ylabel="% Tip")

sns.lineplot(data=tip_by_hour, x="pickup_hour", y="tip_percent", marker="o", ax=axs[2], color="red")
axs[2].set(title="Tip % by Hour", xlabel="Hour", ylabel="% Tip", xticks=range(0, 24, 2))

plt.tight_layout()
plt.show()
```



Observation: 1. Tip amount increases directly with distance, especially from 0 to 20 miles bracket. 2. Drivers get more tips for passenger_count > 2, however above 5 passengers, it shows a pattern of decrease in tips. 3. Drivers get more tips during night hours (2am – 6am) and during office timing (10am–15pm).

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

```python
# Keep valid fares only
df = df[df["fare_amount"] > 0].copy()

# Tip % calculation
df["tip_percent"] = (df["tip_amount"] / df["fare_amount"]) * 100

# Filter low and high tip trips
low_tips = df[df["tip_percent"] < 10]
high_tips = df[df["tip_percent"] > 25]

# Create subplot grid
fig, axs = plt.subplots(2, 3, figsize=(18, 10))

# Trip distance
sns.histplot(low_tips["trip_distance"], bins=30, kde=True, ax=axs[0, 0], color="blue")
sns.histplot(high_tips["trip_distance"], bins=30, kde=True, ax=axs[1, 0], color="green")
axs[0, 0].set_title("Low Tips (<10%) - Distance")
axs[1, 0].set_title("High Tips (>25%) - Distance")

# Fare amount
sns.histplot(low_tips["fare_amount"], bins=30, kde=True, ax=axs[0, 1], color="blue")
sns.histplot(high_tips["fare_amount"], bins=30, kde=True, ax=axs[1, 1], color="green")
axs[0, 1].set_title("Low Tips (<10%) - Fare")
axs[1, 1].set_title("High Tips (>25%) - Fare")

# Passenger count
sns.countplot(data=low_tips, x="passenger_count", ax=axs[0, 2], palette="Blues_r")
sns.countplot(data=high_tips, x="passenger_count", ax=axs[1, 2], palette="Greens_r")
axs[0, 2].set_title("Low Tips (<10%) - Passengers")
axs[1, 2].set_title("High Tips (>25%) - Passengers")

plt.tight_layout()
plt.show()
```

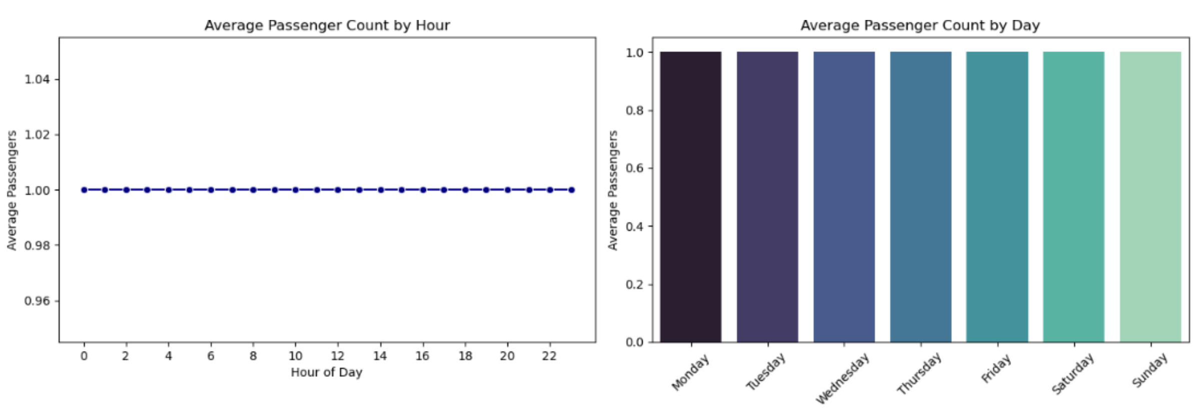


Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

1.2.14. Analyse the trends in passenger count

Analyse the variation of passenger count across hours and days of the week.

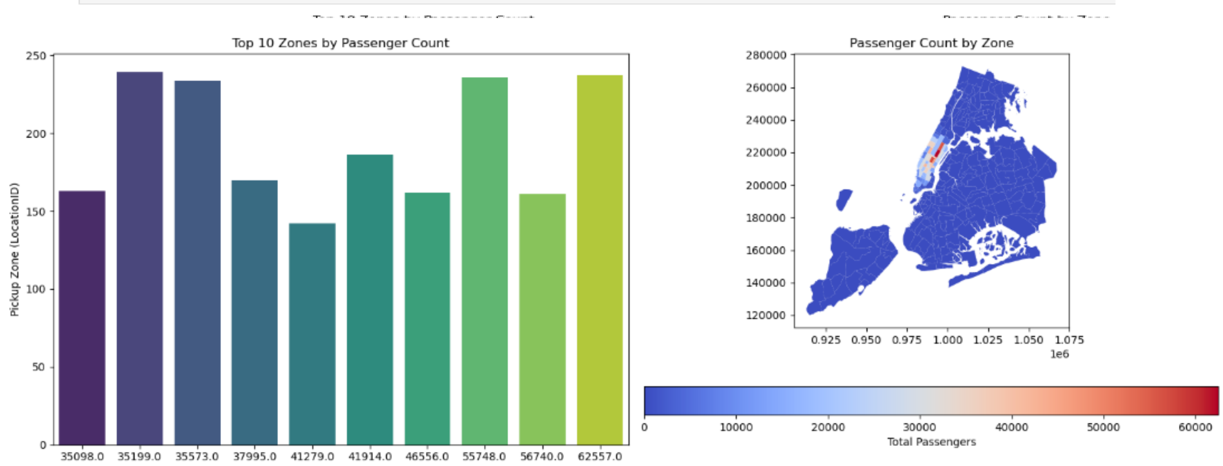
```
# See how passenger count varies across hours and days
# Analyse variation in passenger count across different hours and days

# Convert pickup time to datetime and extract hour/day
df["pickup_datetime"] = pd.to_datetime(df["tpep_pickup_datetime"])
df["hour"] = df["pickup_datetime"].dt.hour
df["day_name"] = df["pickup_datetime"].dt.day_name()

# Average passenger count per hour
avg_passengers_hour = (
    df.groupby("hour")["passenger_count"]
    .mean()
    .reset_index()
)

# Average passenger count per day (ordered Monday → Sunday)
day_order = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]
avg_passengers_day = (
    df.groupby("day_name")["passenger_count"]
    .mean()
    .reset_index()
    .assign(day_name=lambda x: pd.Categorical(x["day_name"], categories=day_order, ordered=True))
    .sort_values("day_name")
)

# Plotting
fig, ax = plt.subplots(1, 2, figsize=(14, 5))

# Hourly trend
sns.lineplot(data=avg_passengers_hour, x="hour", y="passenger_count", marker="o", ax=ax[0], color="navy")
ax[0].set_title("Average Passenger Count by Hour")
ax[0].set_xlabel("Hour of Day")
ax[0].set_ylabel("Average Passengers")
ax[0].set_xticks(range(0, 24, 2))

# Daily trend
sns.barplot(data=avg_passengers_day, x="day_name", y="passenger_count", ax=ax[1], palette="mako")
ax[1].set_title("Average Passenger Count by Day")
ax[1].set_xlabel("Day of Week")
ax[1].set_ylabel("Average Passengers")
ax[1].tick_params(axis="x", rotation=45)

plt.tight_layout()
plt.show()
```



Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

1.2.15. Analyse the variation of passenger counts across zones

```
# How does passenger count vary across zones
import geopandas as gpd
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# --- Passenger totals by pickup zone ---
zone_totals = (
    df.groupby("PULocationID", as_index=False)["passenger_count"]
      .sum()
      .rename(columns={"passenger_count": "total_passengers"})
)

# --- Load taxi zones and join counts ---
shp_path = r"C:\Users\prash\Downloads\Datasets and Dictionary-NYC\Datasets and Dictionary\taxi_zones\taxi_zones.shp"
zones_gdf = gpd.read_file(shp_path)
zones_gdf = zones_gdf.merge(zone_totals, left_on="LocationID", right_on="PULocationID", how="left")
zones_gdf["total_passengers"] = zones_gdf["total_passengers"].fillna(0)

# --- Plots ---
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# (1) Top 10 zones by passenger count (same orientation as your code)
top10 = zone_totals.nlargest(10, "total_passengers")
sns.barplot(data=top10, x="total_passengers", y="PULocationID", ax=axes[0], palette="viridis")
axes[0].set_title("Top 10 Zones by Passenger Count")
axes[0].set_xlabel("Total Passengers")
axes[0].set_ylabel("Pickup Zone (LocationID)")

# (2) Choropleth map of passenger density
zones_gdf.plot(
    column="total_passengers",
    cmap="coolwarm",
    legend=True,
    ax=axes[1],
    legend_kwds={"label": "Total Passengers", "orientation": "horizontal"}
)
axes[1].set_title("Passenger Count by Zone")

plt.tight_layout()
plt.show()
```



Took only top 10 records so that we don't make a crowded chart.

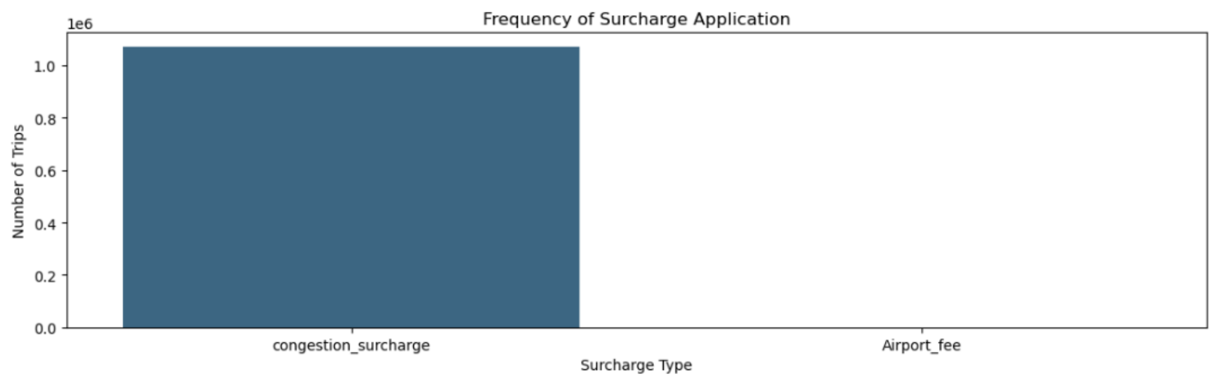
1.2.16. Analyse the pickup/dropoff zones or times when extra charges are applied more frequently.

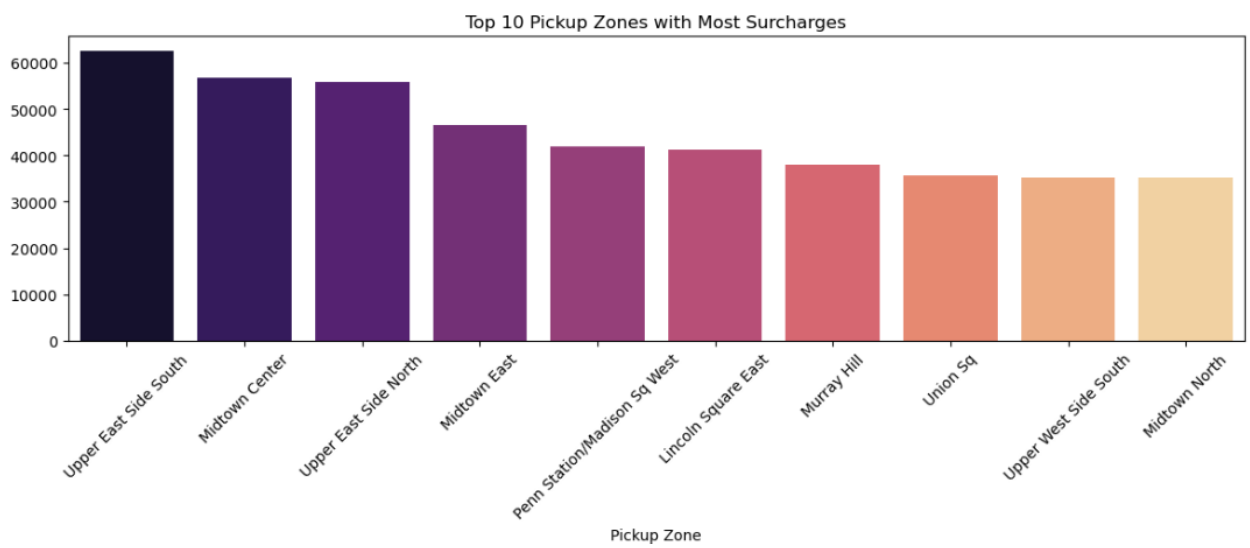
Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

```python
# How often is each surcharge applied?

# Define surcharge-related columns in the dataset
surcharge_columns = ["congestion_surcharge", "Airport_fee"]

# Identify trips with any surcharge applied
df["surcharge_applied"] = (df[surcharge_columns].sum(axis=1) > 0).astype(int)

# Count how often each surcharge is applied
surcharge_counts = df[surcharge_columns].apply(lambda x: (x > 0).sum())

# Convert to DataFrame for plotting
surcharge_counts_df = surcharge_counts.reset_index()
surcharge_counts_df.columns = ["Surcharge Type", "Count"]

# Extract hour from pickup datetime
df["pickup_hour"] = pd.to_datetime(df["tpep_pickup_datetime"]).dt.hour

# Count surcharge occurrences per hour
hourly_surcharge_counts = df.groupby("pickup_hour")["surcharge_applied"].sum().reset_index()

# Count surcharge occurrences per pickup and dropoff zone
pickup_surcharge_counts = df.groupby("PULocationID")["surcharge_applied"].sum().reset_index()
dropoff_surcharge_counts = df.groupby("DOLocationID")["surcharge_applied"].sum().reset_index()

# Merge with taxi zone names
pickup_surcharge_counts = pickup_surcharge_counts.merge(zones[["LocationID", "zone"]],
                                                        left_on="PULocationID", right_on="LocationID", how="left")
dropoff_surcharge_counts = dropoff_surcharge_counts.merge(zones[["LocationID", "zone"]],
                                                          left_on="DOLocationID", right_on="LocationID", how="left")

# Sort by highest surcharge occurrence
top_pickup_surcharge_zones = pickup_surcharge_counts.sort_values(by="surcharge_applied", ascending=False).head(10)
top_dropoff_surcharge_zones = dropoff_surcharge_counts.sort_values(by="surcharge_applied", ascending=False).head(10)

# Visualization: Plot everything
fig, axes = plt.subplots(3, 1, figsize=(12, 15))
```

```python
# Plot surcharge frequency
sns.barplot(x="Surcharge Type", y="Count", data=surcharge_counts_df, palette="viridis", ax=axes[0])
axes[0].set_title("Frequency of Surcharge Application")
axes[0].set_xlabel("Surcharge Type")
axes[0].set_ylabel("Number of Trips")

# Plot surcharge application by pickup/dropoff zone
sns.barplot(x="zone", y="surcharge_applied", data=top_pickup_surcharge_zones, palette="magma", ax=axes[1])
axes[1].set_title("Top 10 Pickup Zones with Most Surcharges")
axes[1].set_xlabel("Pickup Zone")
axes[1].set_ylabel("Number of Trips with Surcharge")
axes[1].tick_params(axis="x", rotation=45)

sns.barplot(x="zone", y="surcharge_applied", data=top_dropoff_surcharge_zones, palette="coolwarm", ax=axes[2])
axes[2].set_title("Top 10 Dropoff Zones with Most Surcharges")
axes[2].set_xlabel("Dropoff Zone")
axes[2].set_ylabel("Number of Trips with Surcharge")
axes[2].tick_params(axis="x", rotation=45)
plt.tight_layout()
plt.show()

# Plot surcharge application by time of day
plt.figure(figsize=(10, 5))
sns.lineplot(x="pickup_hour", y="surcharge_applied", data=hourly_surcharge_counts, marker="o", color="red")
plt.xlabel("Hour of Day")
plt.ylabel("Number of Trips with Surcharge")
plt.title("Surcharge Application by Hour of Day")
plt.xticks(range(0, 24))
plt.grid(True)
plt.show()
```

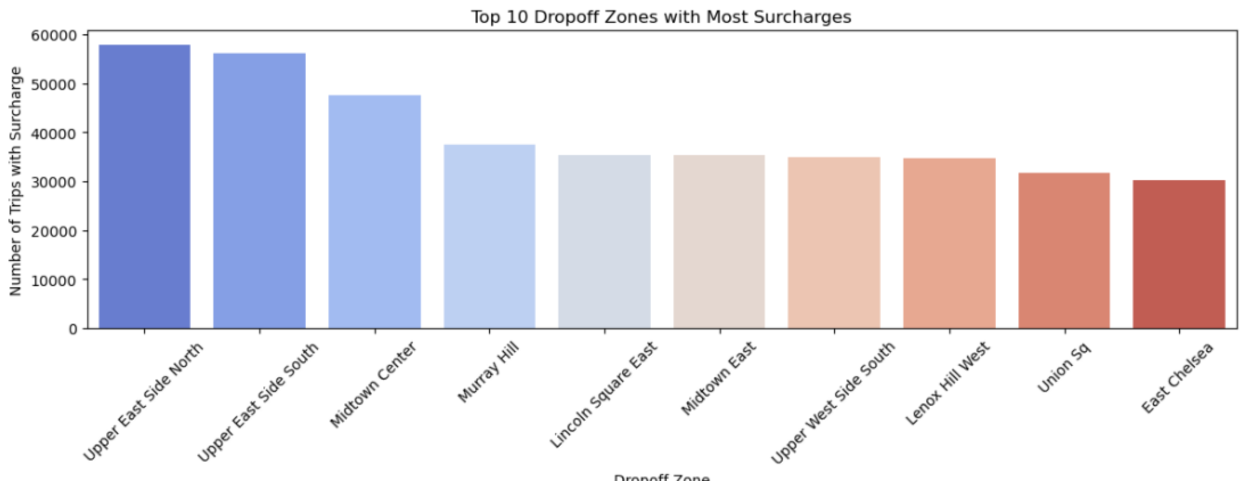
Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

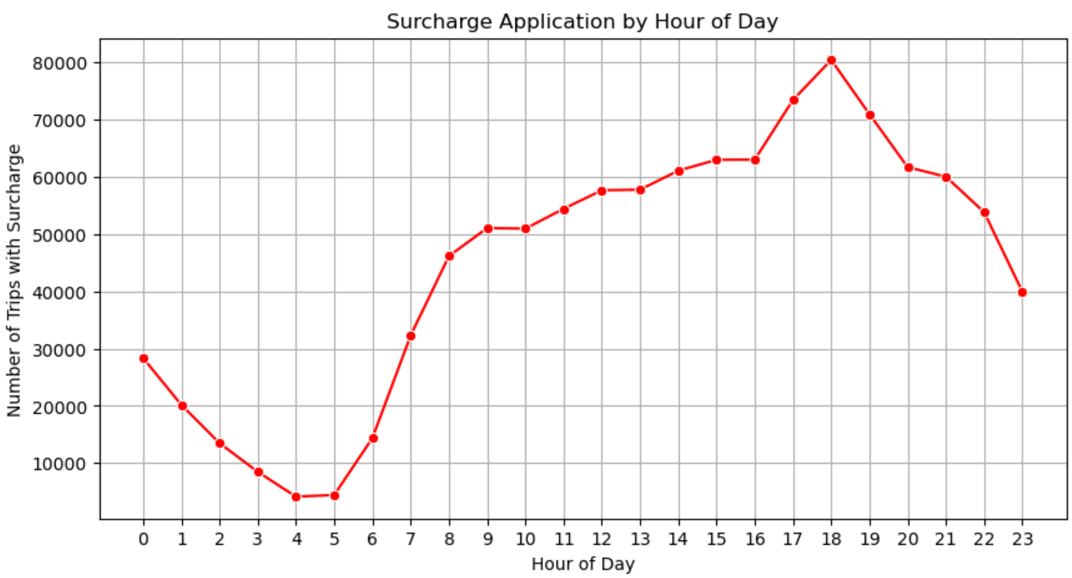




Name- Prashant shrivastava

TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS





Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

2. Conclusions

2.1. Final Insights and Recommendations

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

Boost system efficiency by combining dynamic pricing, forward-looking demand analytics, predictive planning, smarter routing, and tailored driver incentives. Use targeted driver allocation informed by external signals and user behavior (e.g., seats booked/occupied), address surcharge fairness with multi-objective optimization, and keep algorithms adaptable through continuous monitoring and A/B testing.

2.1.2. 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 the fleet strategically by aligning taxi supply with fare-per-mile signals, day-of-week patterns, and peak-hour windows. Use incentives and dynamic pricing to balance demand, factoring in party size, tipping behavior, trip length, and surcharge-prone zones. Adapt in real time with traffic, weather, and event data powered by predictive models, and continually refine zone-level allocation by tracking KPIs and learning from historical performance.

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

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

```python
import numpy as np
import pandas as pd

def propose_pricing(df, competitor_data, undercut=0.50, pct_step=0.10,
                    make_tiers_if_missing=True, dynamic_time=True):
    """
    Propose data-driven price adjustments (per mile) by distance tier, staying competitive.
    """
    # --- 1) Prep & guards ---
    w = df.copy()
    w = w[(w["trip_distance"] > 0) & (w["fare_amount"] > 0)].copy()

    if make_tiers_if_missing or "distance_tier" not in w.columns:
        w["distance_tier"] = pd.cut(
            w["trip_distance"], [0, 2, 5, np.inf],
            labels=["0-2 mi", "2-5 mi", "5+ mi"], right=False
        )

    # --- 2) Our average fare per mile by tier ---
    w["fare_per_mile"] = w["fare_amount"] / w["trip_distance"]
    ours = (w.groupby("distance_tier")["fare_per_mile"]
              .mean().reset_index()
              .rename(columns={"fare_per_mile": "fare_per_mile_your"}))

    # --- 3) Merge with competitor table ---
    comp = competitor_data.copy()
    if "fare_per_mile_competitor" not in comp.columns:
        guess = [c for c in comp.columns if c != "distance_tier"]
        if not guess:
            raise ValueError("competitor_data must include competitor fare column.")
        comp = comp.rename(columns={guess[0]: "fare_per_mile_competitor"})

    merged = pd.merge(ours, comp, on="distance_tier", how="inner")
```

```
  distance_tier  fare_per_mile_your  fare_per_mile_competitor  price_diff  \
0        0-2 mi                9.45                       3.2        6.25
1        2-5 mi                6.40                       2.5        3.90
2         5+ mi                4.73                       2.0        2.73

               $change_reason  price_adjustment_pct  suggested_fare_per_mile
0  Above competitor by > $0.50                 -10.9                     8.42
1  Above competitor by > $0.50                 -10.7                     5.72
2  Above competitor by > $0.50                 -10.3                     4.24
```

```python
# How often is each surcharge applied?

surcharge_columns = ["congestion_surcharge", "Airport_fee", "tolls_amount"]

# Calculate the percentage of non-zero values for each surcharge
surcharge_frequency = (df[surcharge_columns] > 0).mean() * 100

# Convert to DataFrame for easier viewing/plotting
surcharge_freq_df = surcharge_frequency.reset_index()
surcharge_freq_df.columns = ["Surcharge Type", "Frequency (%)"]

print("Frequency of Surcharge Application (%):")
print(surcharge_freq_df)
```

```
Frequency of Surcharge Application (%):
         Surcharge Type  Frequency (%)
0  congestion_surcharge          100.0
1           Airport_fee            0.0
2          tolls_amount            0.0
```

Name- Prashant shrivastava
TITLE - EDA NYC YELLOW TAXI DATA ANALYSIS

THANK YOU!!!