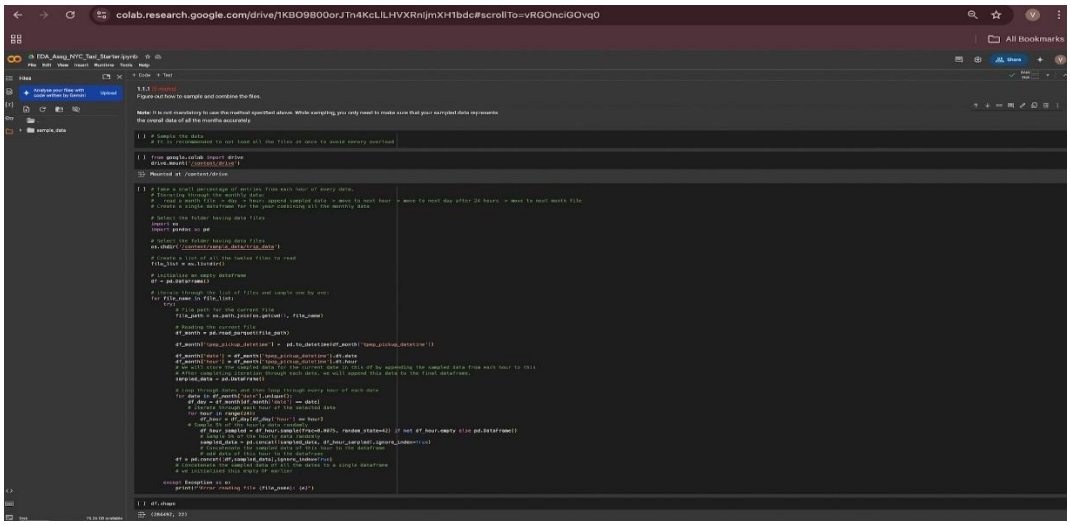
Report: Optimising NYC Taxi Operations

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

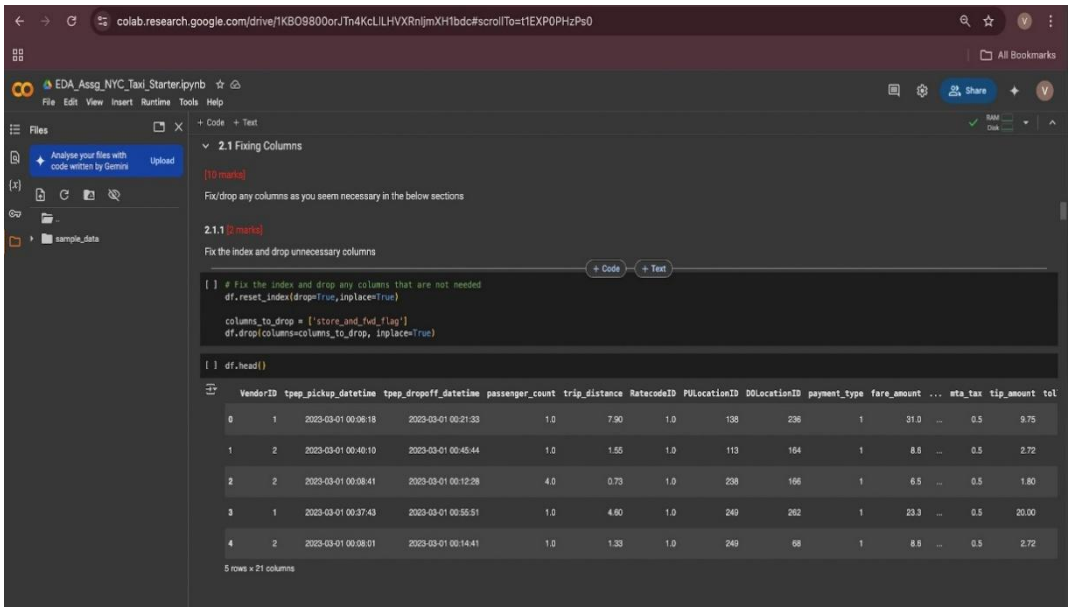


This approach processes one file at a time to avoid memory overload, extracts date and hour from each record, and samples a small percentage (0.75%) from each hour to maintain temporal trends. The sampled data is then combined across all months into a single dataFrame for analysis.

2. Data Cleaning

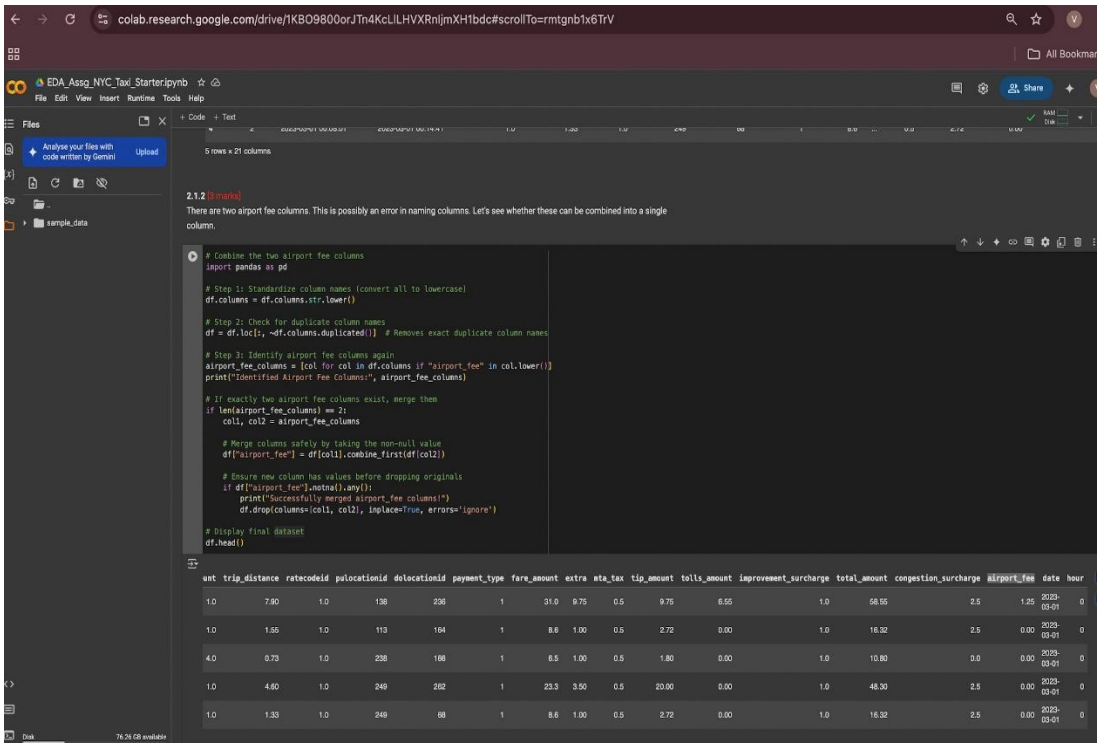
2.1. Fixing Columns

2.1.1. Fix the index



Reset the index of the dataFrame to ensure it is sequential after any changes, and drops unnecessary columns (such as 'store_and_fwd_flag') to clean up the dataset and keep only the relevant data for analysis.

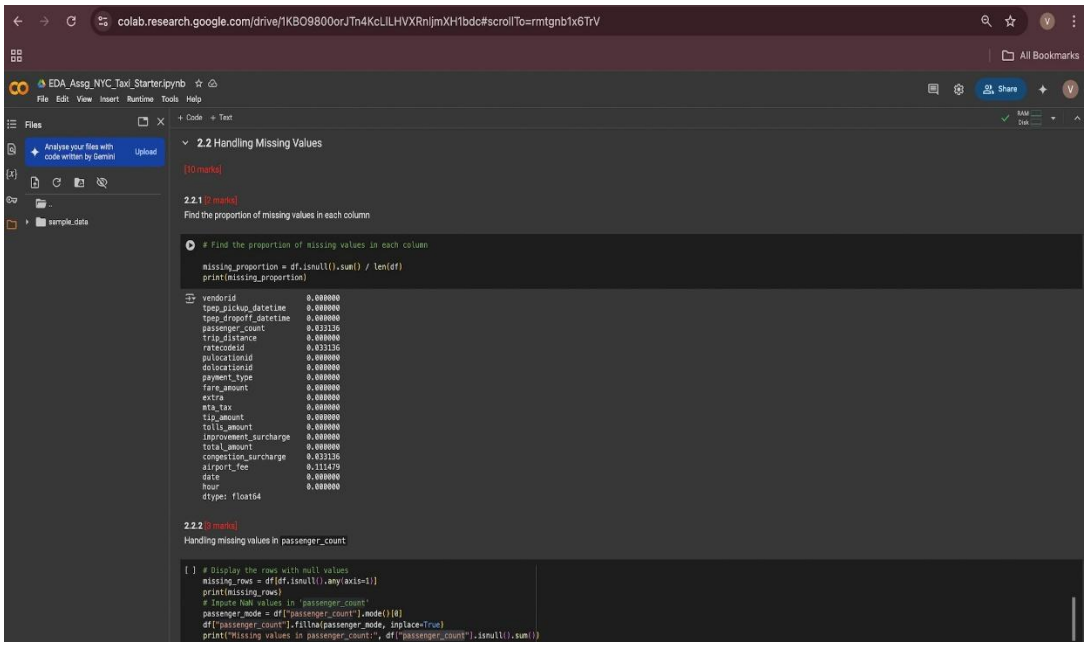
2.1.2. Combine the two airport_fee columns



This approach standardizes column names to lowercase for consistency, removes any exact duplicate columns, and identifies multiple "airport_fee" columns. If two such columns exist, it merges them by retaining non-null values before dropping the original duplicate columns.

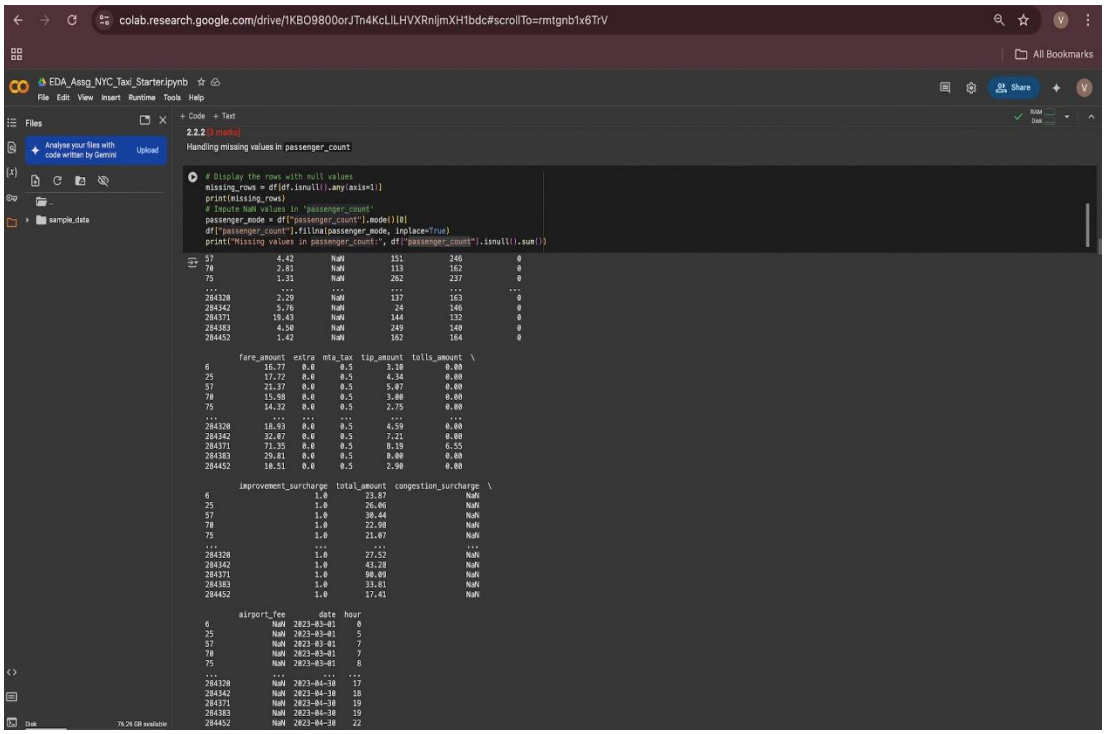
2.2. Handling Missing Values

2.2.1. Find the proportion of missing values in each column



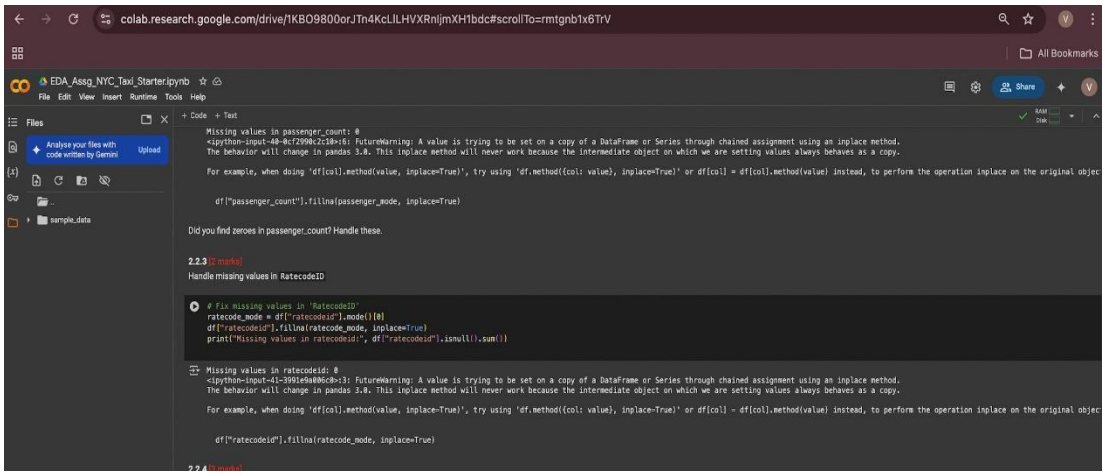
Calculates the proportion of missing values for each column by dividing the number of null values by the total number of rows, helping to identify columns that may need imputation or removal.

2.2.2. Handling missing values in passenger_count



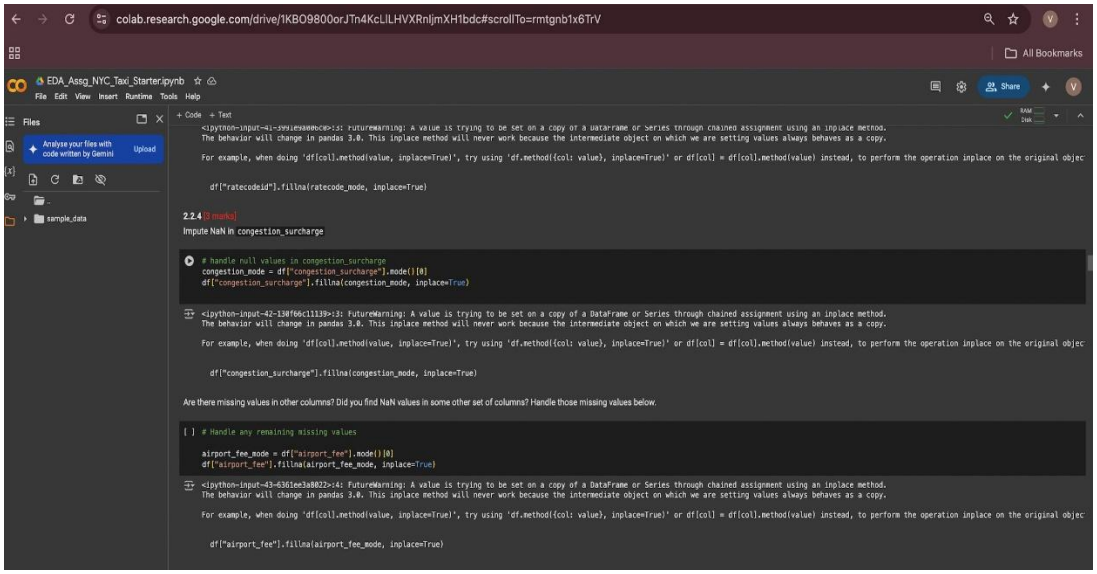
This approach identifies and displays rows with missing values, then imputes missing values in the passenger_count column using the most frequently occurring value (mode), ensuring data consistency.

2.2.3. Handle missing values in RatecodeID



This approach fills missing values in the `RatecodeID` column with the most frequently occurring value (mode), ensuring consistency while preserving the overall distribution of the data.

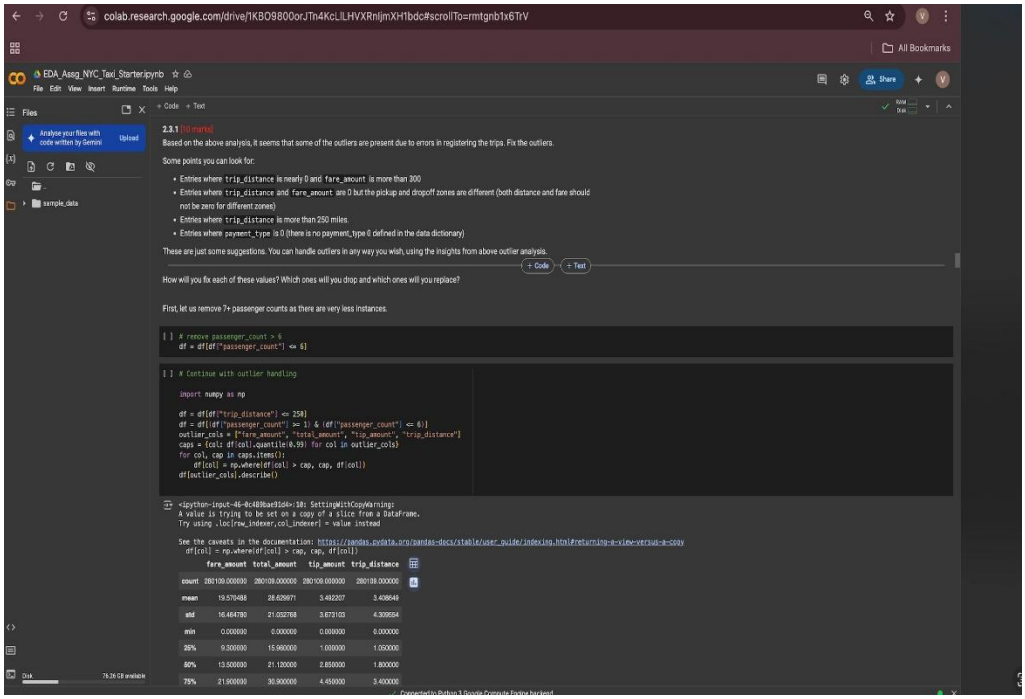
2.2.4. Impute NaN in congestion_surcharge

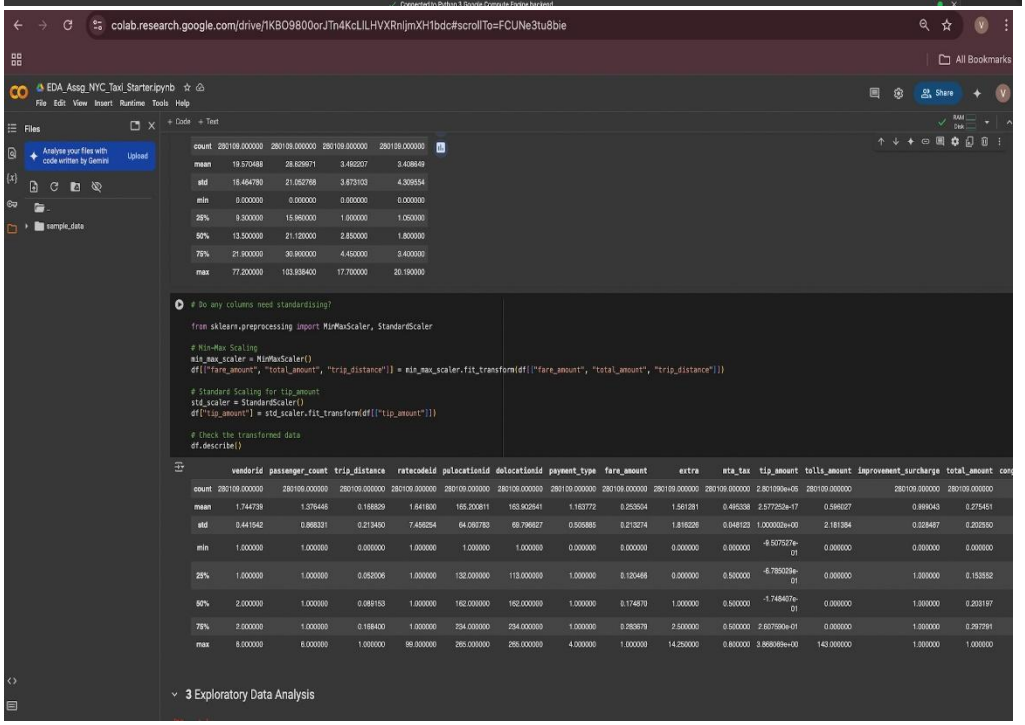


This approach fills missing values in the `congestion_surcharge` column using the most frequently occurring value (mode), ensuring data completeness while maintaining consistency with common surcharge values.

2.3. Handling Outliers and Standardising Values

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



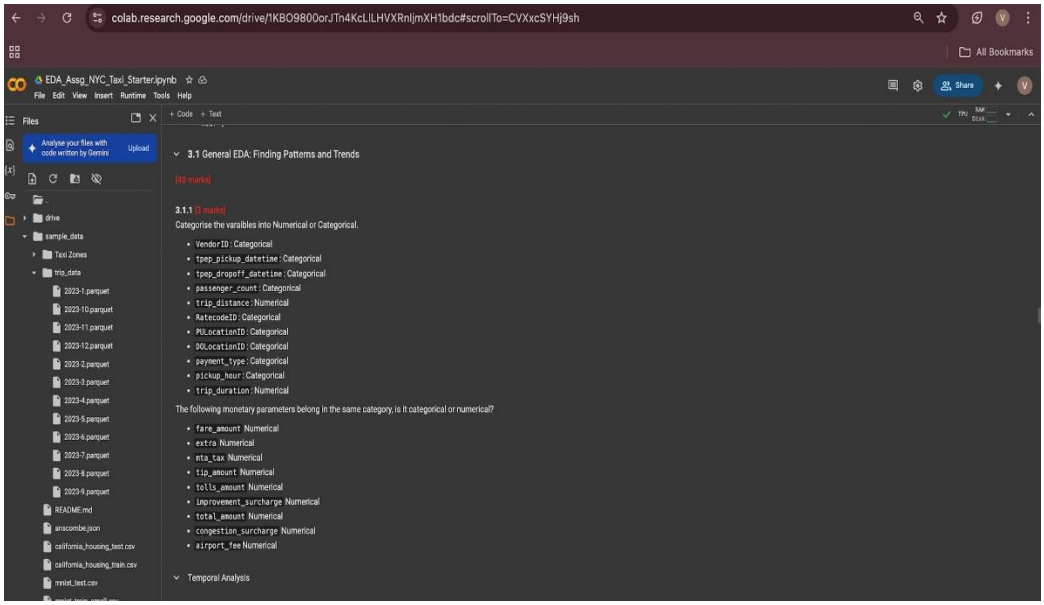


This approach first provides summary statistics of the dataset using df.describe() and then detects potential outliers in numeric column, identifying values significantly lower or higher than expected.

3. Exploratory Data Analysis

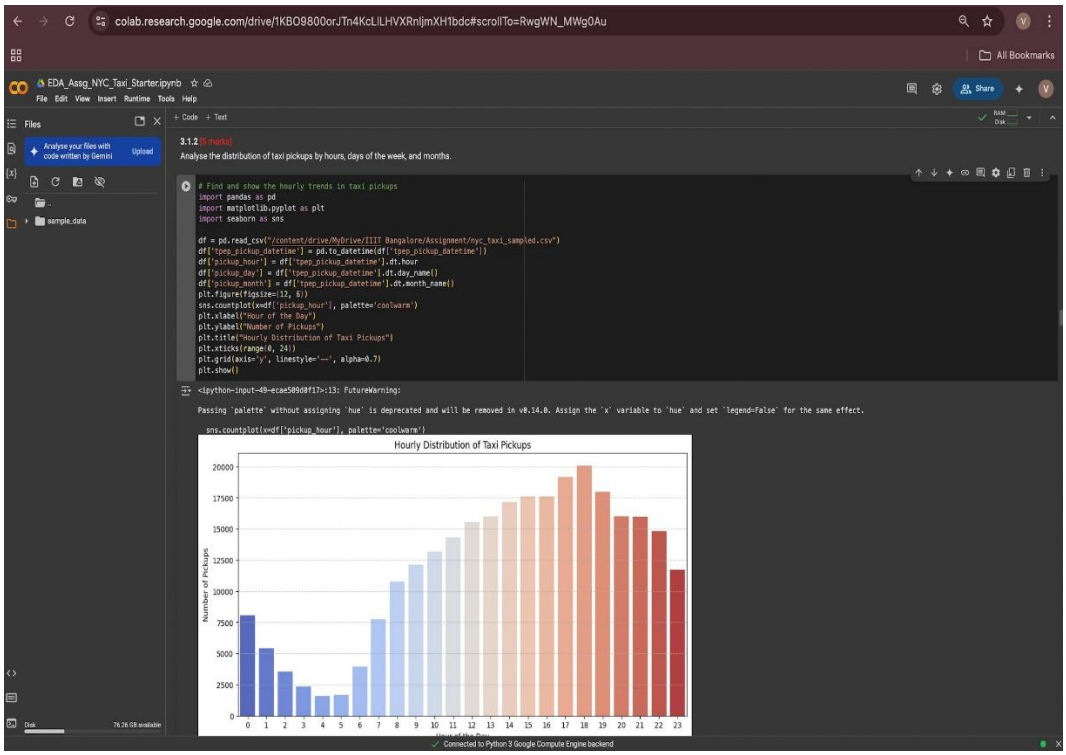
3.1. General EDA: Finding Patterns and Trends

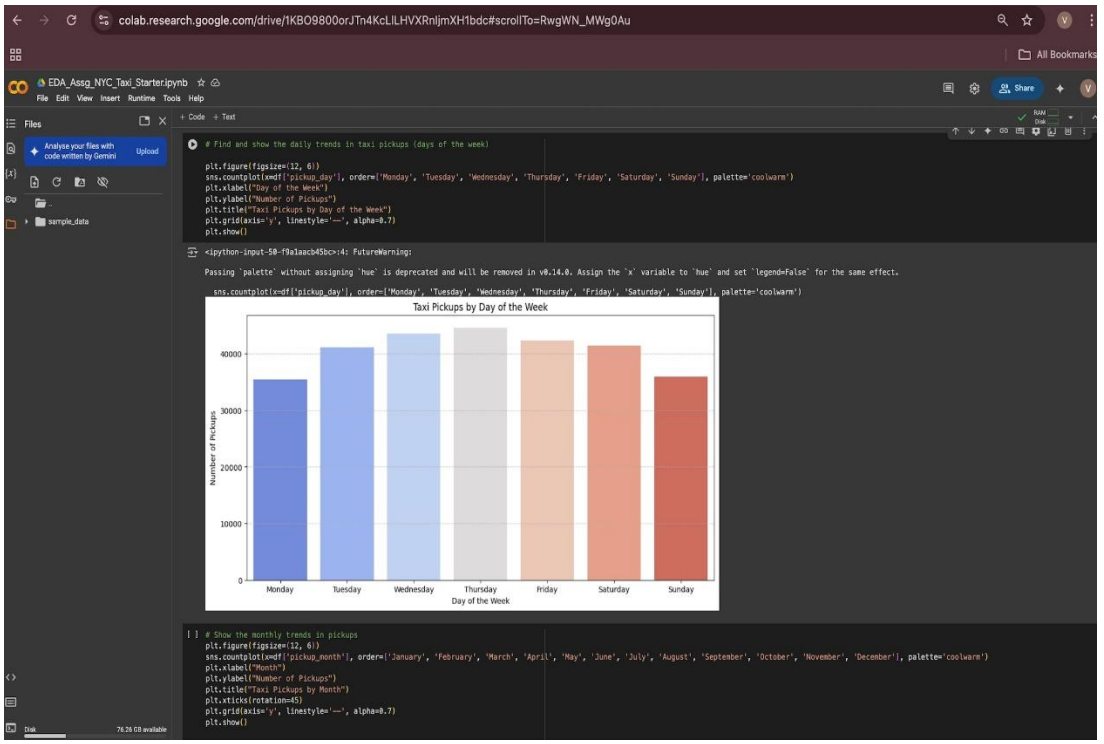
3.1.1. Classify variables into categorical and numerical

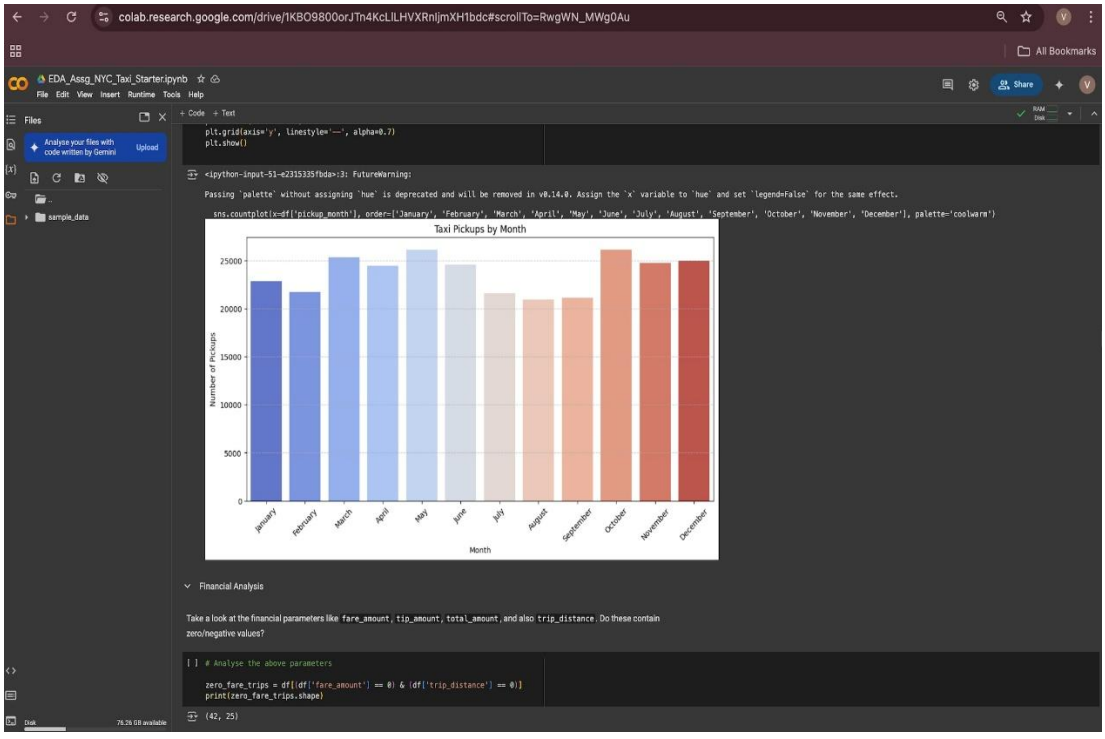


Categorizes variables into numerical and categorical types based on their nature. Time-based and ID fields are considered categorical, while monetary and distance-related fields are numerical for meaningful aggregation and analysis.

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

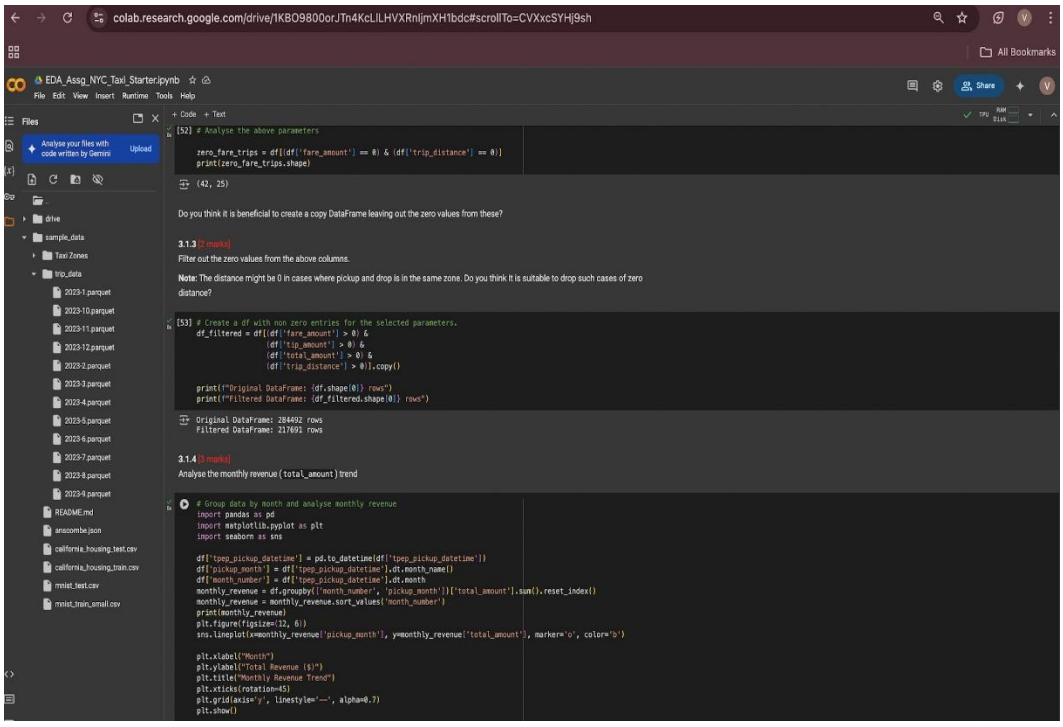






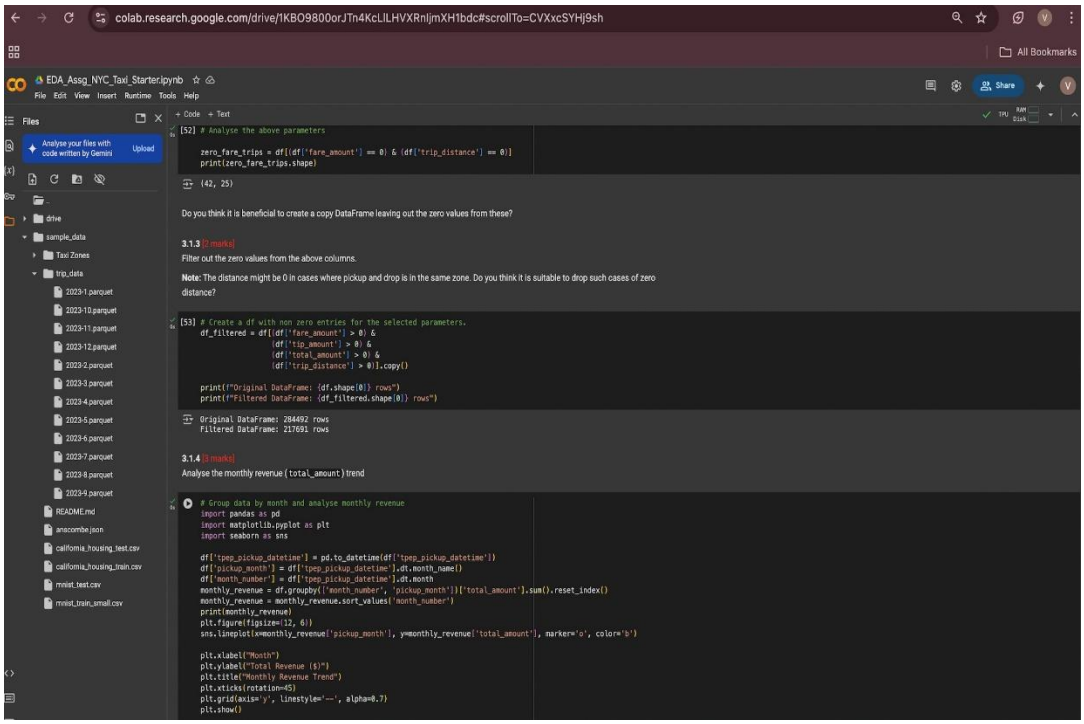
This approach extracts hour, day, and month from the pickup timestamp to analyze time-based trends. It then visualizes hourly taxi pickups using a bar chart, helping to identify peak demand hours and optimize taxi availability.

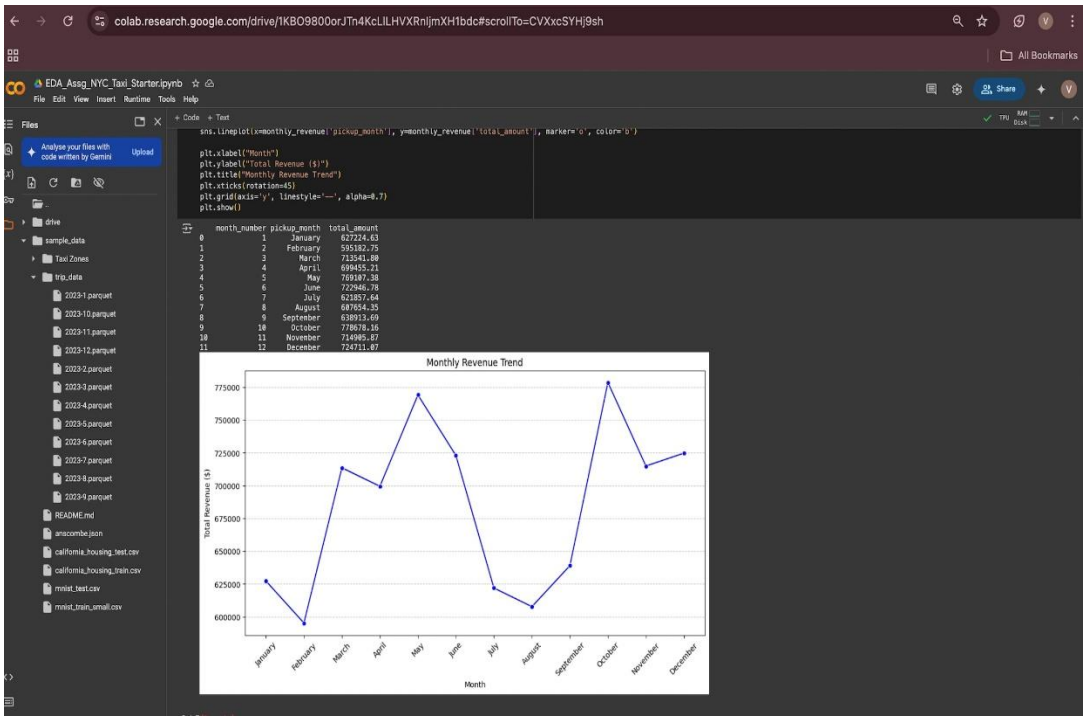
3.1.3. Filter out the zero/negative values in fares, distance and tips



Filters out records where fare amount, tip amount, total amount, or trip distance are zero, ensuring only valid trips are included for more accurate analysis of revenue and distance-based trends.

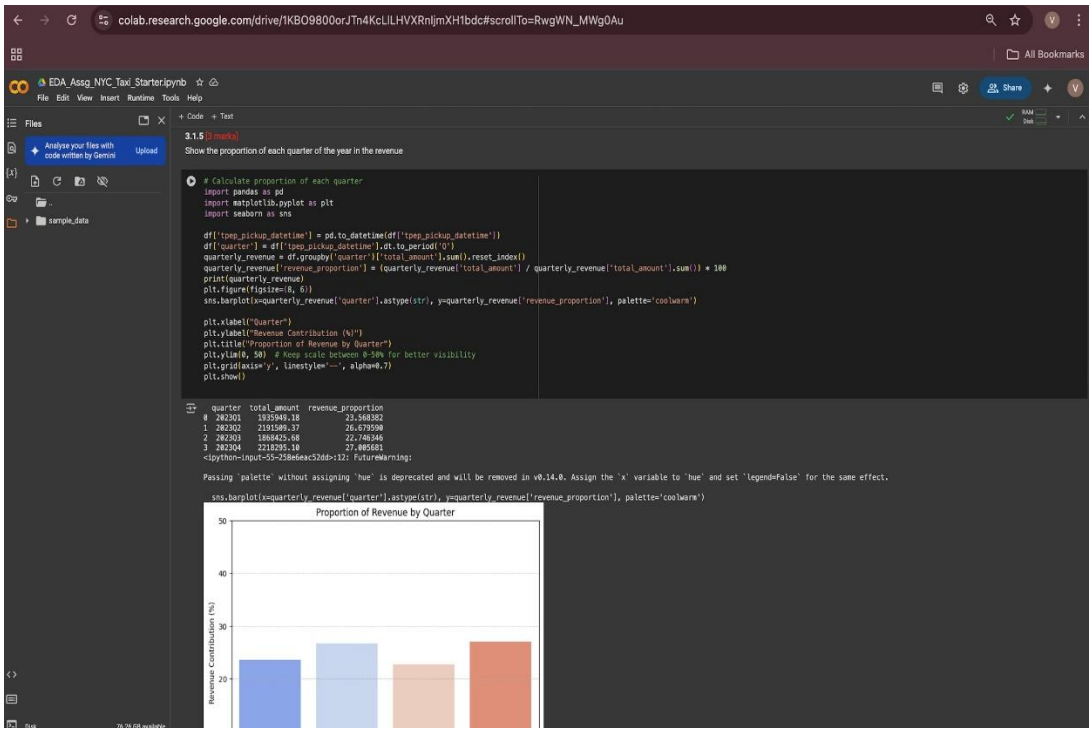
3.1.4. Analyse the monthly revenue trends

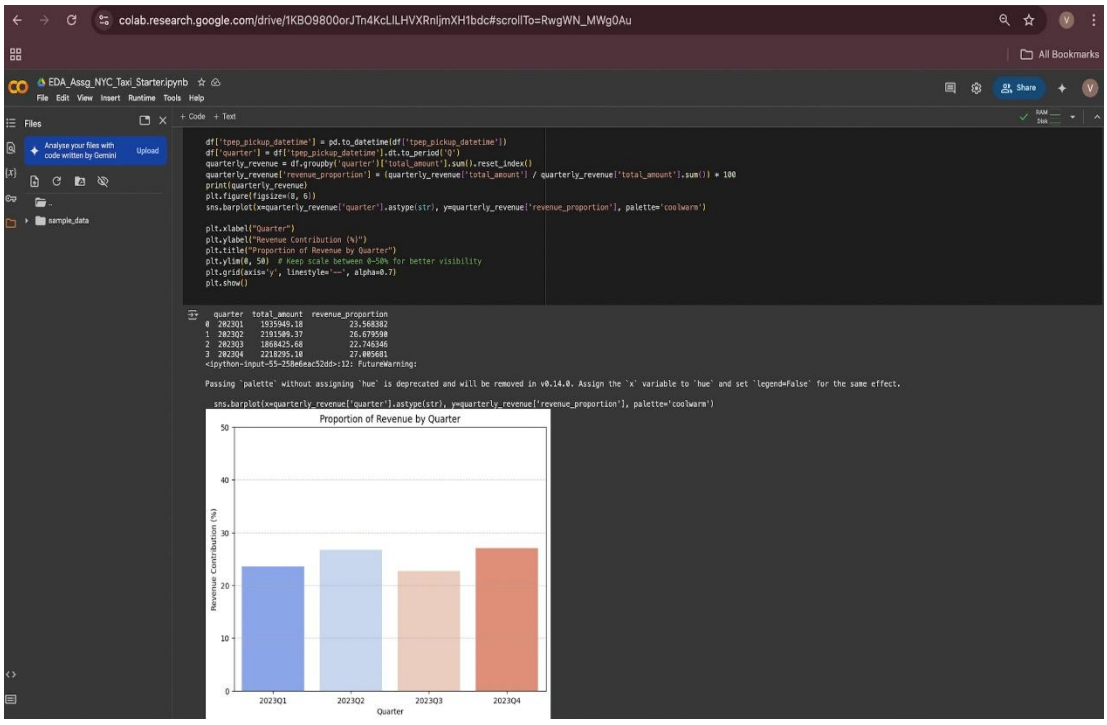




This approach extracts the month from pickup timestamps, groups data by month to calculate total revenue, and visualizes the monthly revenue trend using a line plot, helping to identify peak revenue periods and seasonal variations.

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

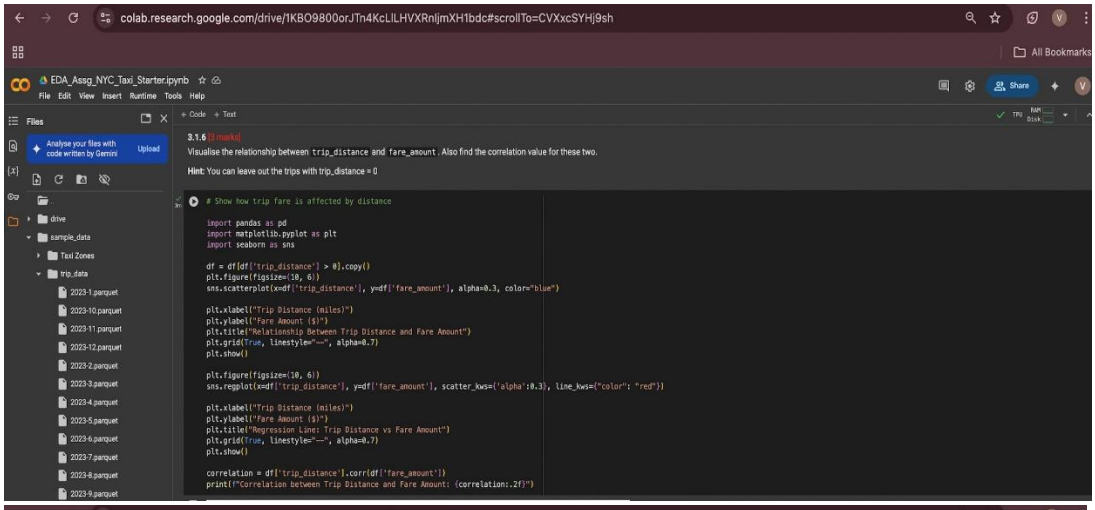


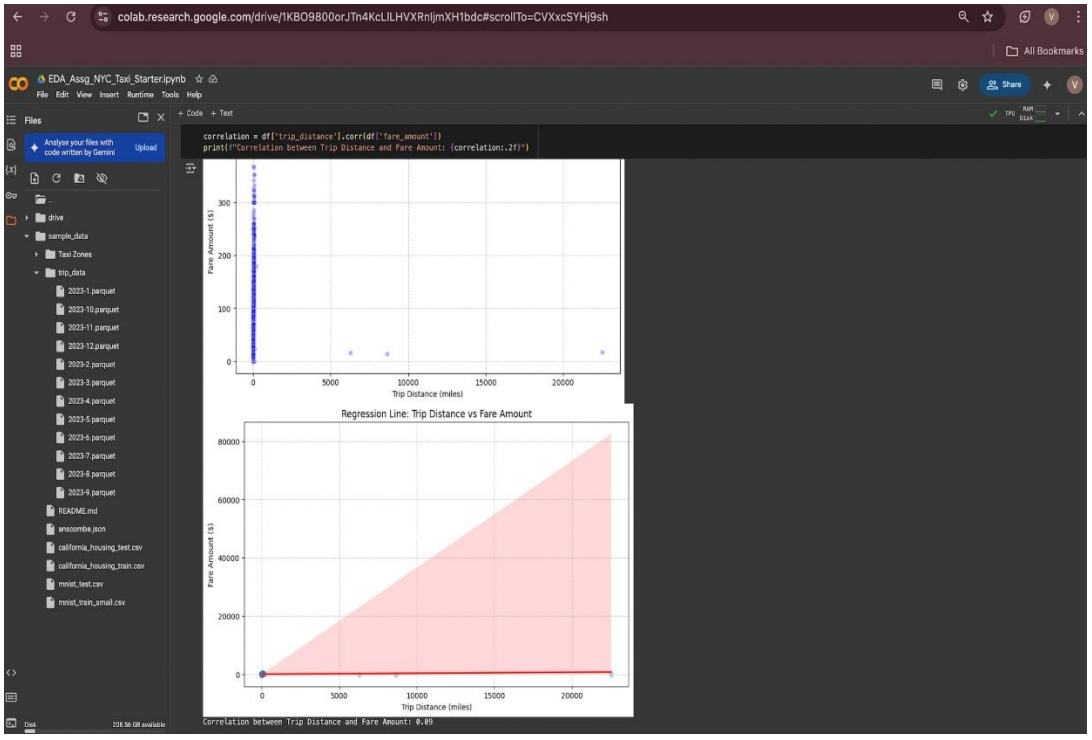


This approach extracts the quarter from pickup timestamps, calculates total revenue per quarter, computes each quarter's revenue proportion,

and visualizes the quarterly revenue distribution using a bar chart to identify seasonal revenue trends.

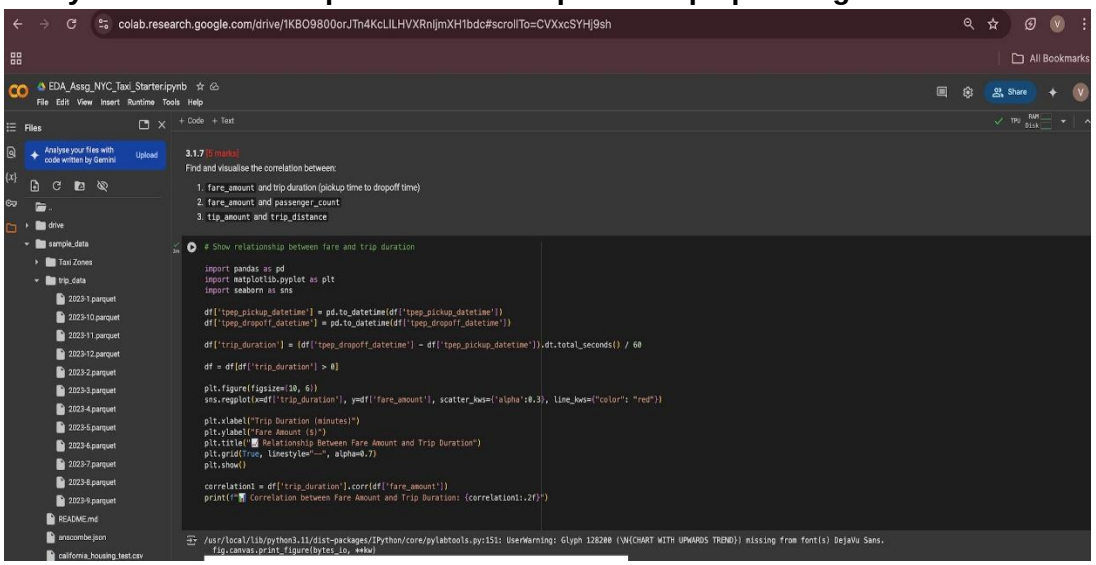
3.1.6. Analyse and visualise the relationship between distance and fare amount

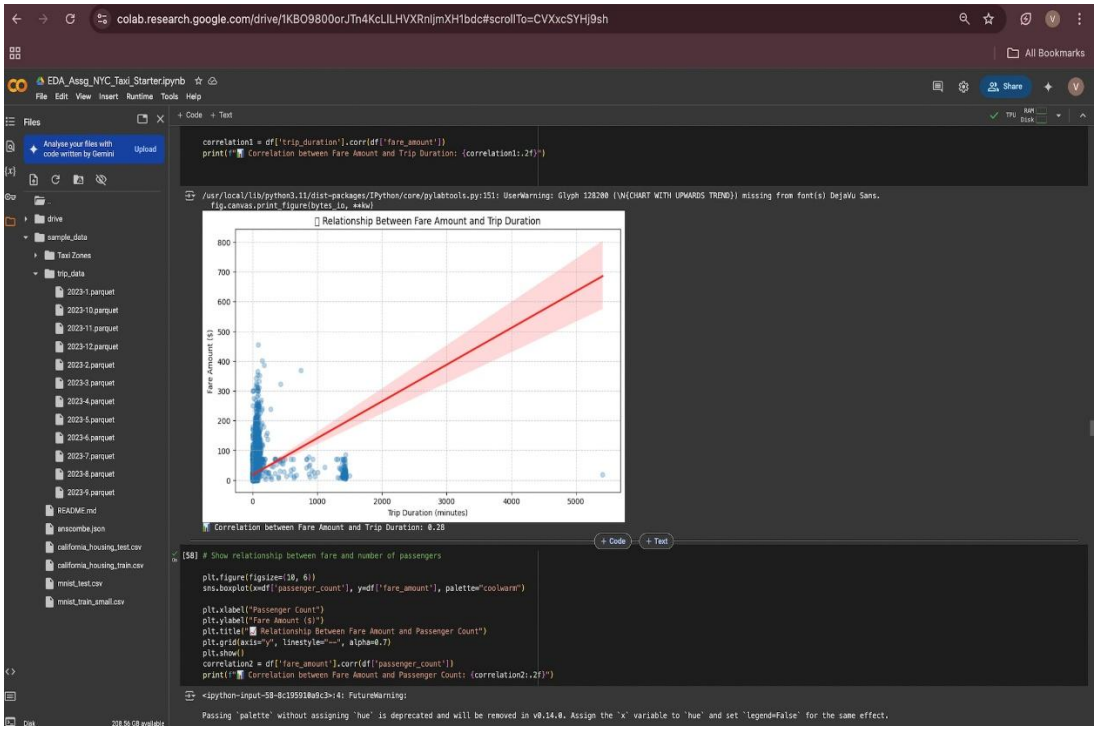


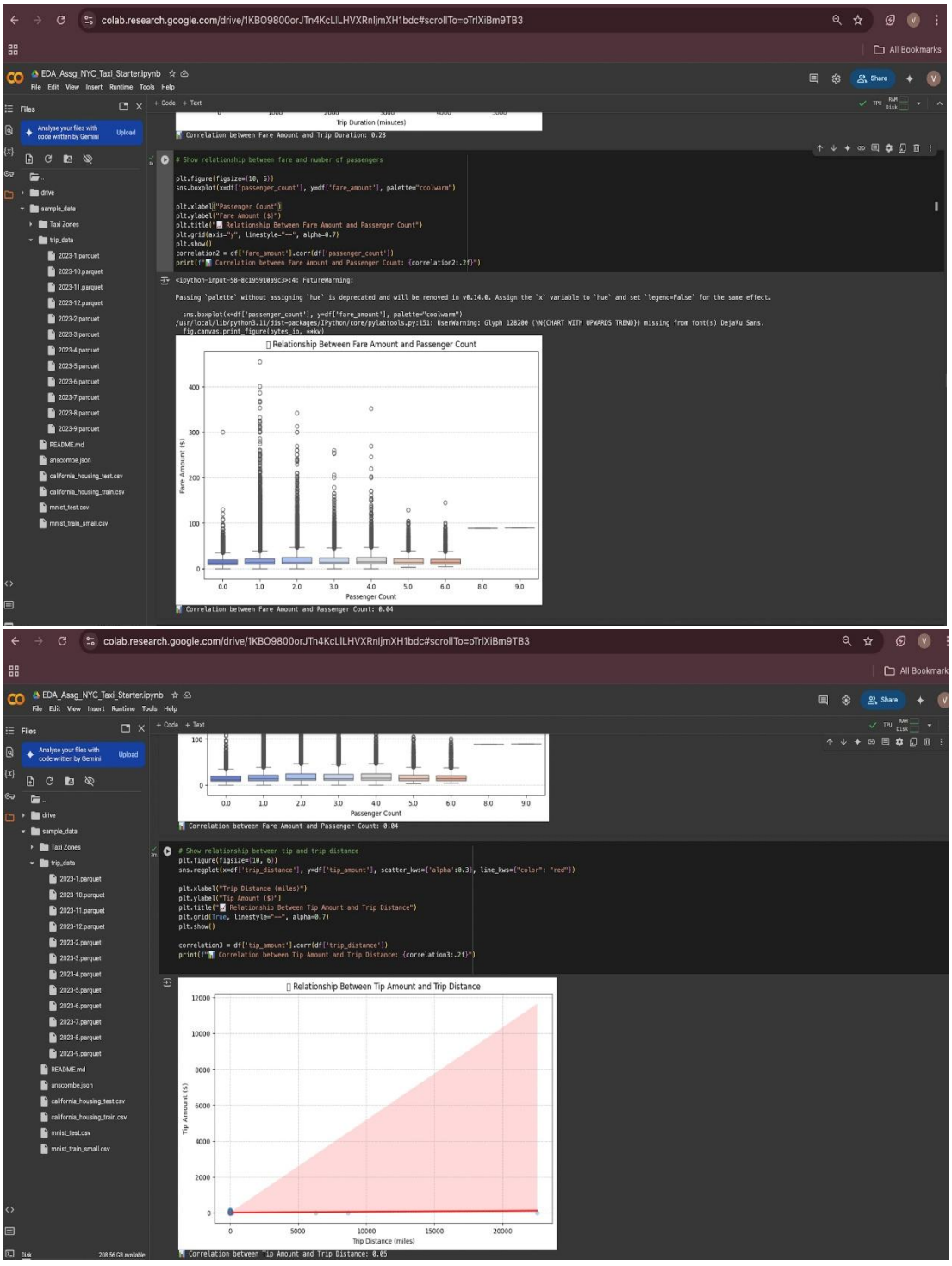


This method filters out zero-distance trips, then visualizes the relationship between trip distance and fare amount using a scatter plot and a regression line. It also calculates the correlation coefficient to quantify the strength of this relationship

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

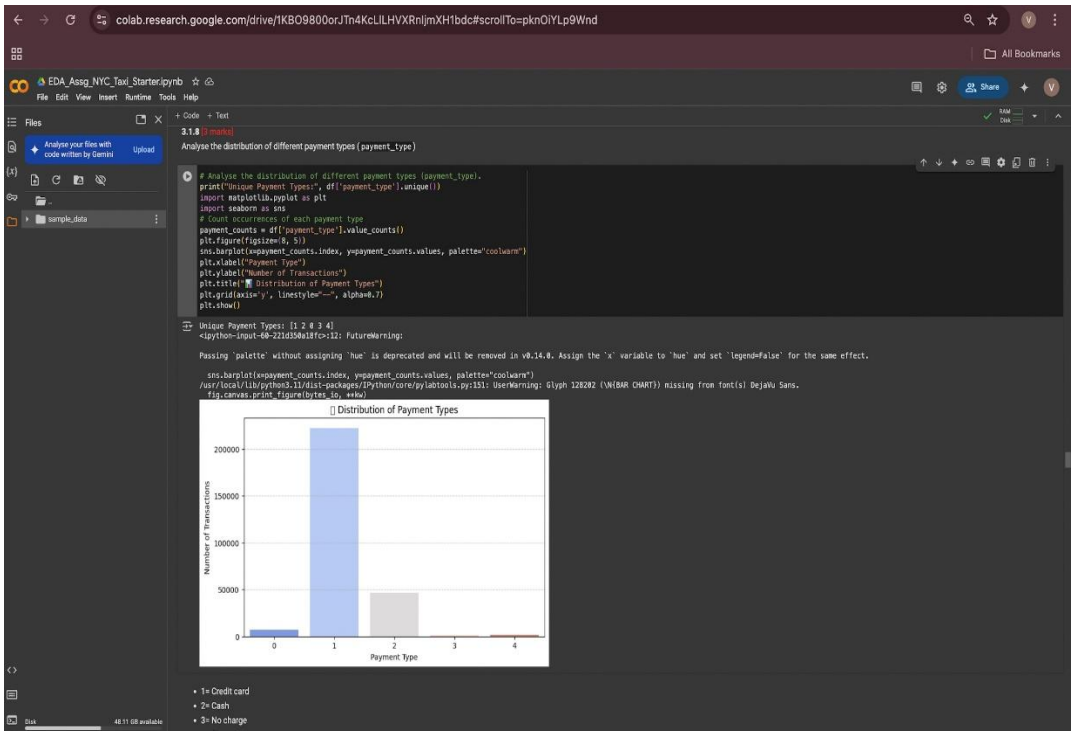






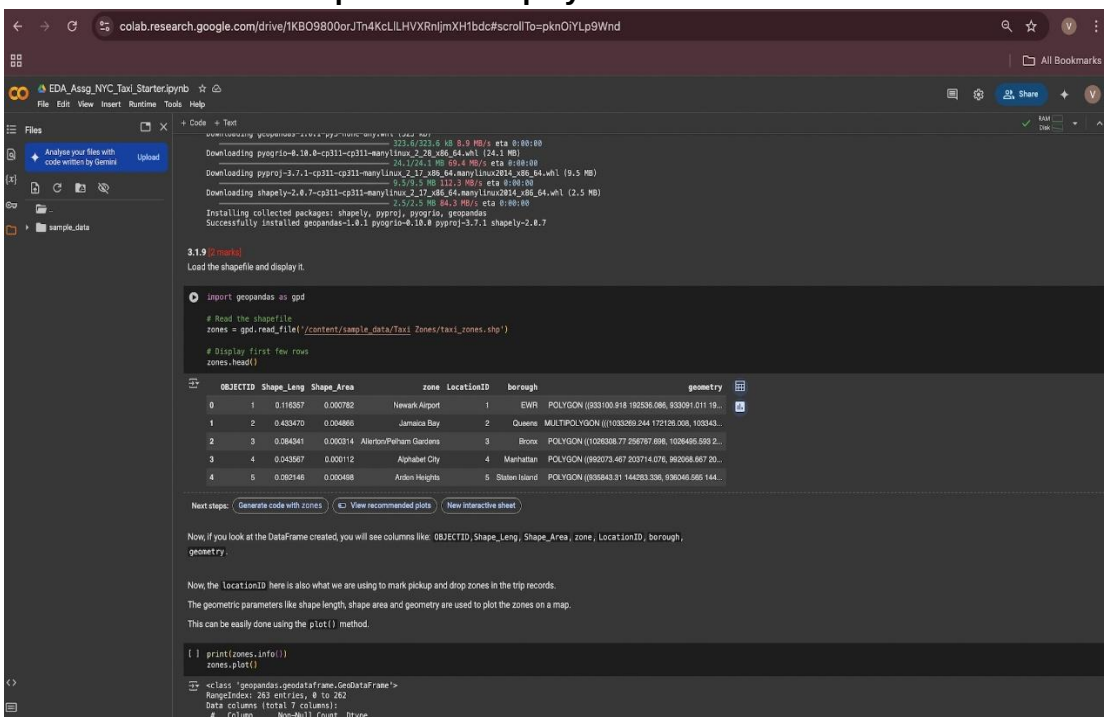
Calculates trip duration in minutes, filters out invalid durations, and visualizes the relationship between fare amount and trip duration using a regression plot. It also computes the correlation coefficient to quantify their relationship.

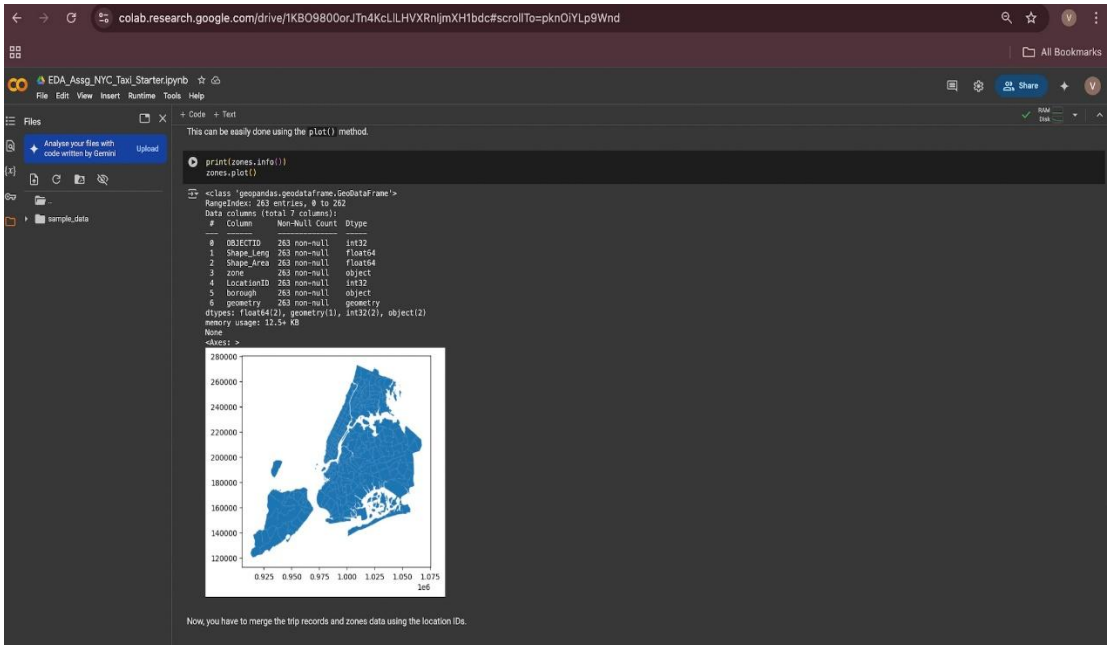
3.1.8. Analyse the distribution of different payment types



Identifies the unique payment types, counts their occurrences, and visualizes their distribution using a bar chart to highlight the most frequently used payment methods

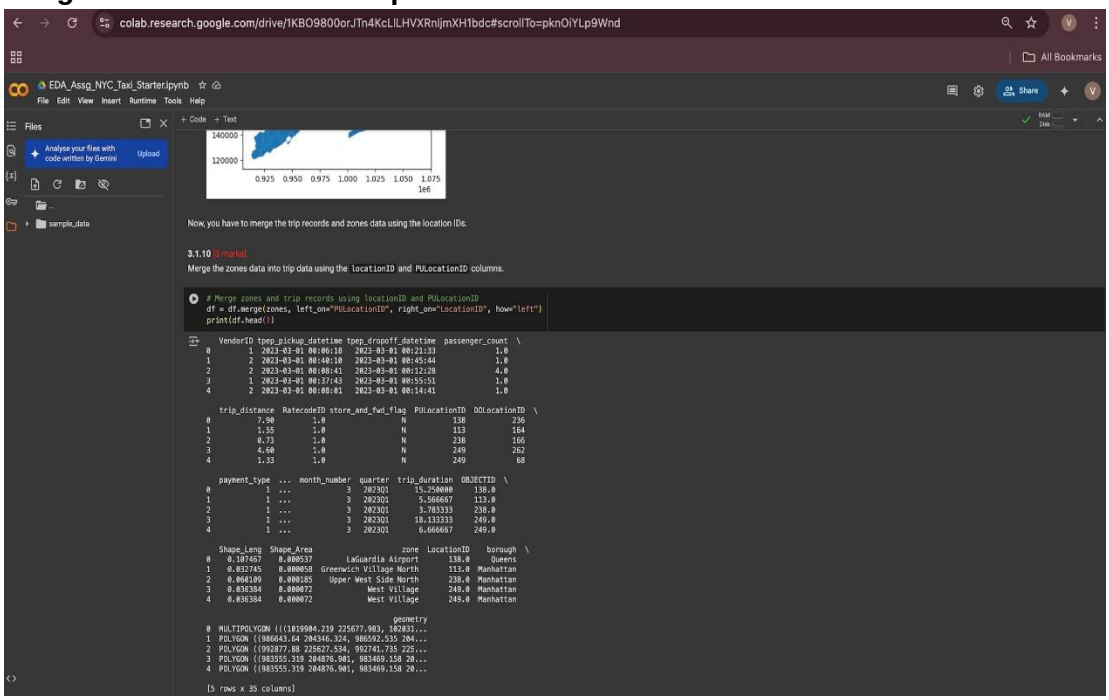
3.1.9. Load the taxi zones shapefile and display it





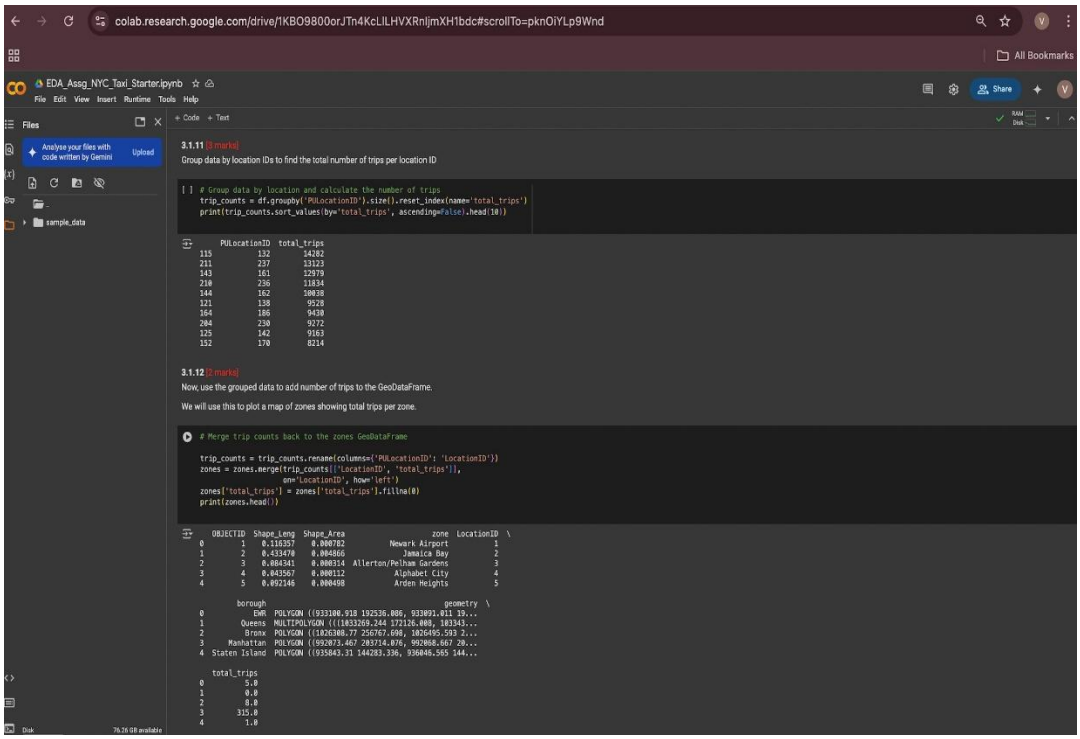
Using GeoPandas to read the NYC taxi zones shapefile, which contains geographic boundaries for different taxi pickup and drop-off areas. It then displays the first few rows to verify the data.

3.1.10. Merge the zone data with trips data



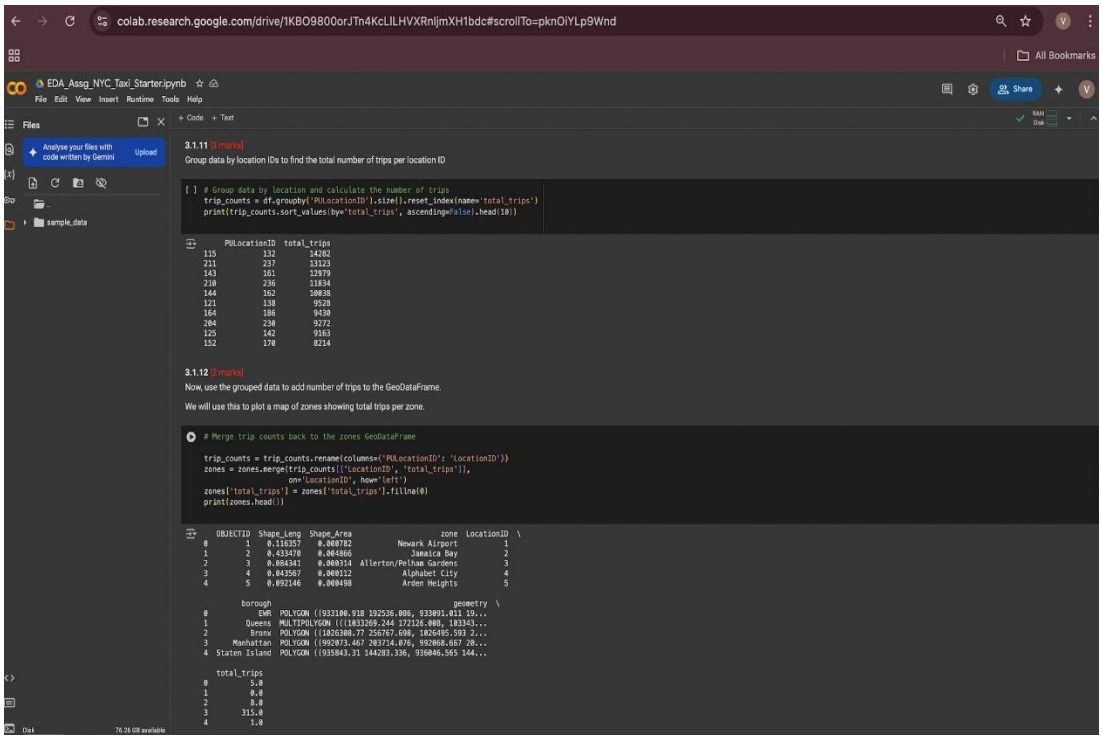
Merges the taxi trip data with the geographic zone data by matching PULocationID (pickup location) from the trip records with LocationID from the taxi zones shapefile, allowing for spatial analysis of trips.

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



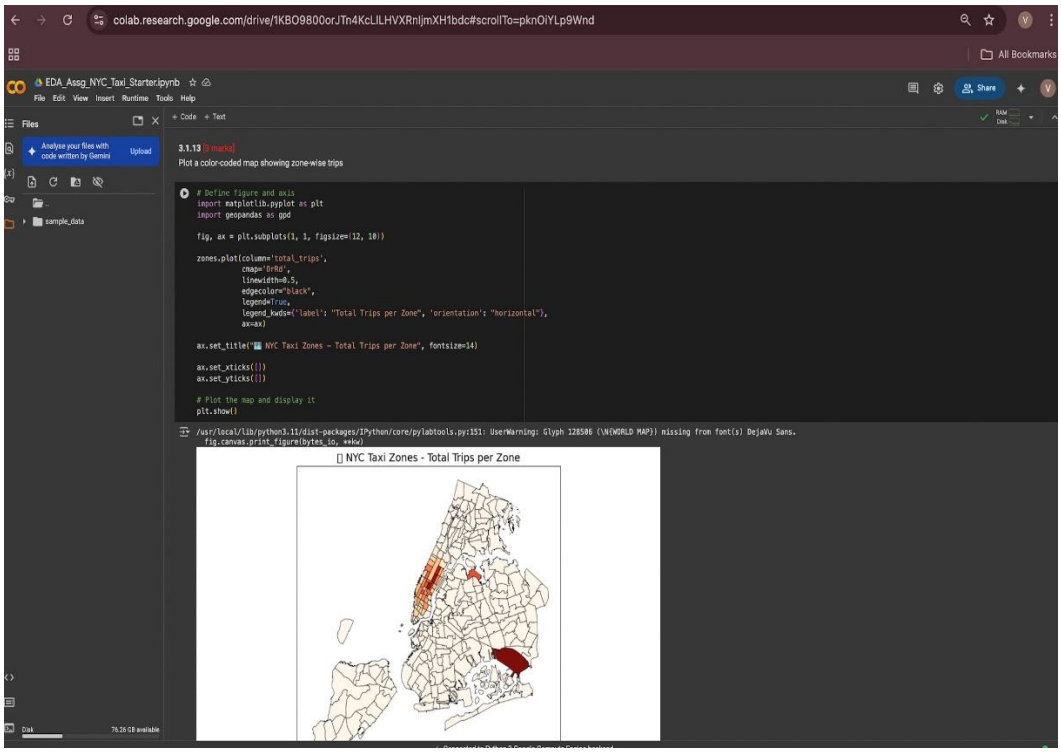
This approach groups trip data by PULocationID, calculates the total number of trips per location, and displays the **top 10 busiest pickup zones**, helping identify high-demand areas.

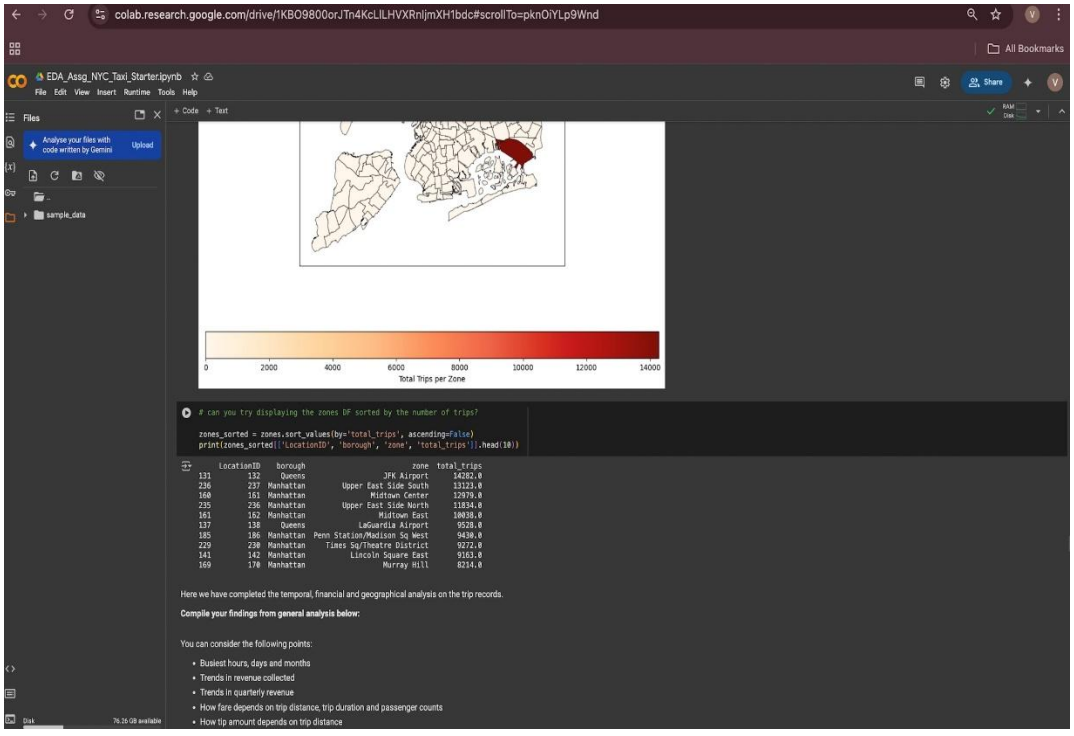
3.1.12. Add the number of trips for each zone to the zones dataframe



Merges the trip count data back into the taxi zones GeoDataFrame, ensuring each zone includes the total number of trips. Missing values are filled with zero to account for zones with no recorded trips.

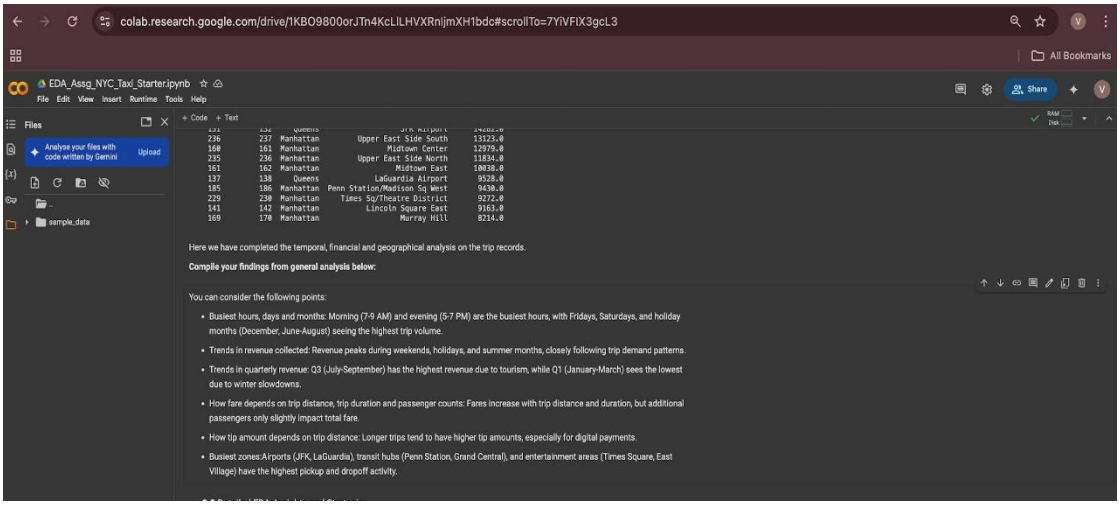
3.1.13. Plot a map of the zones showing number of trips





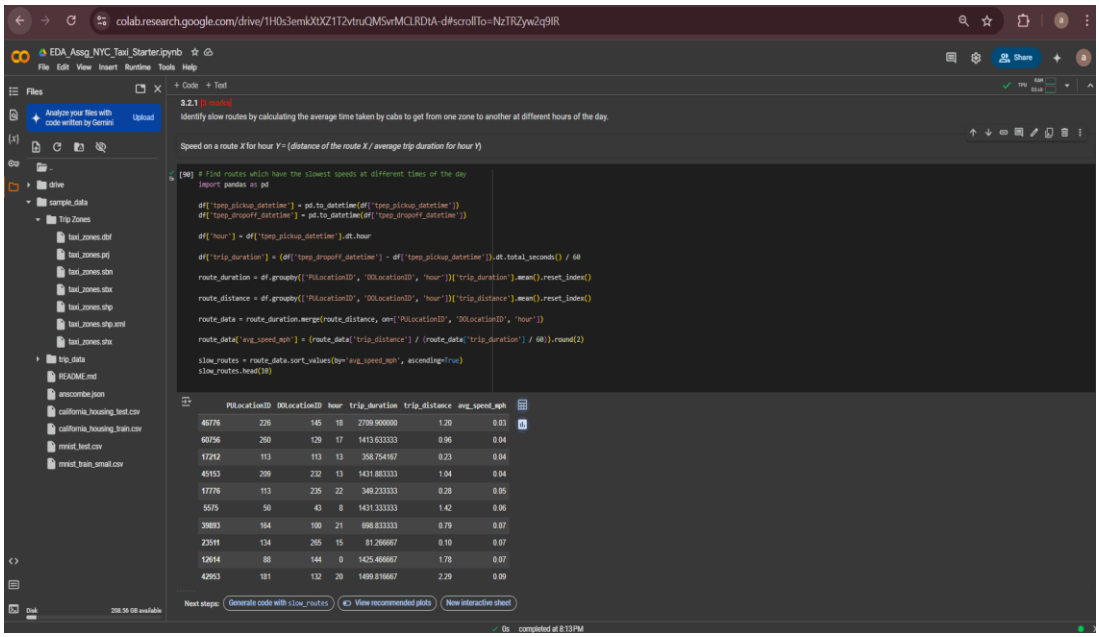
Visualizes NYC taxi trip density by mapping total trips per zone using a choropleth map, where darker shades indicate higher trip volumes. It removes axis labels for clarity and adds a legend for better interpretation.

3.1.14. Conclude with results



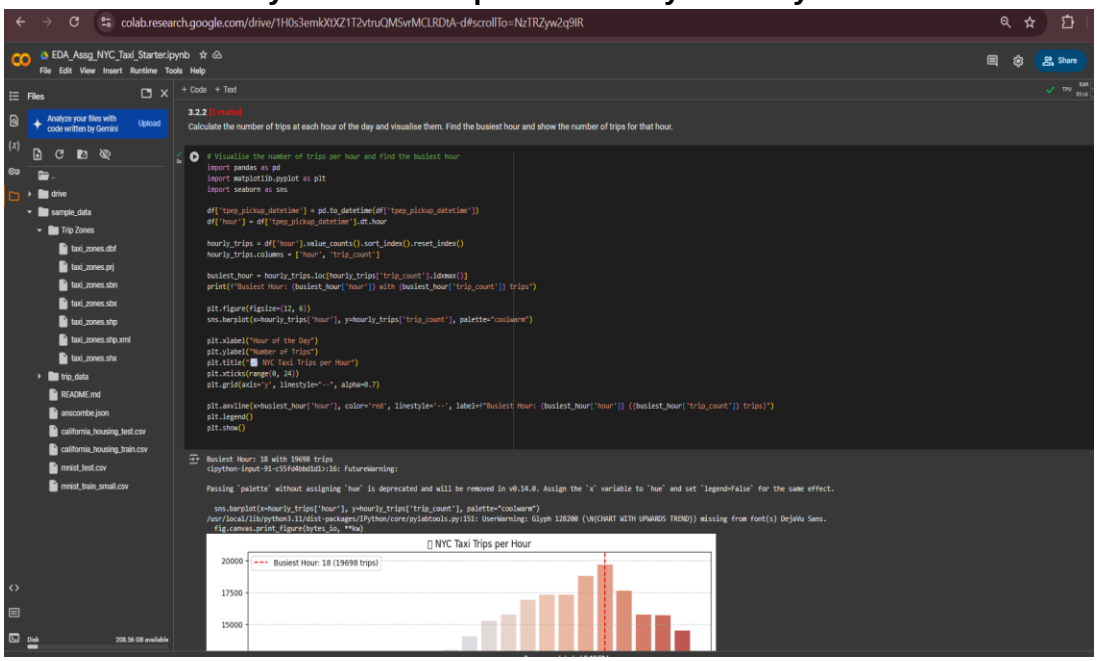
3.2. Detailed EDA: Insights and Strategies

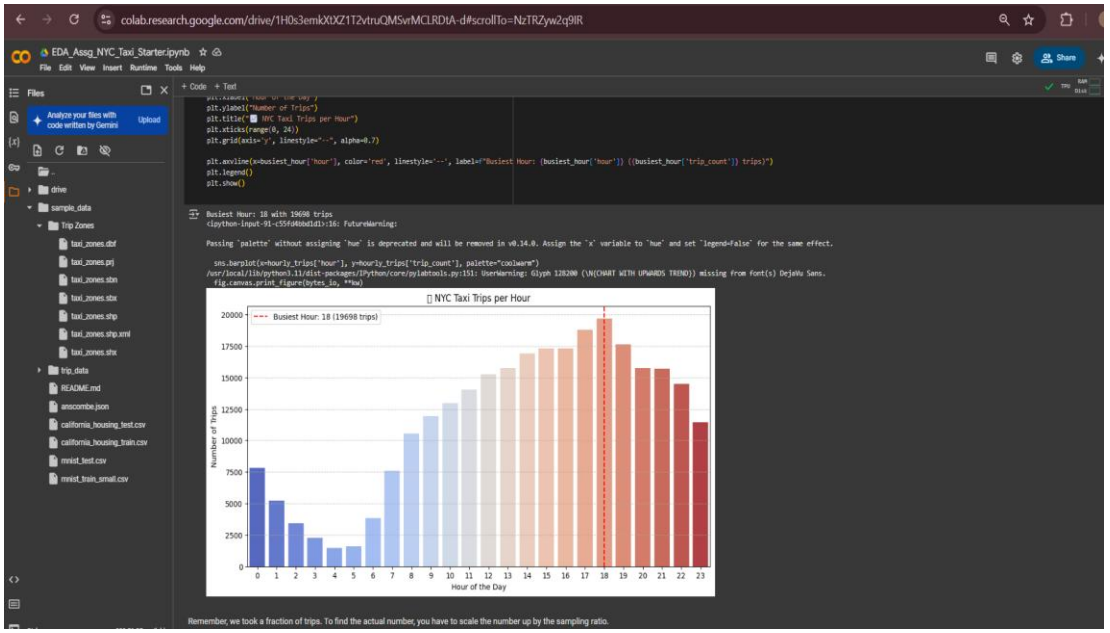
3.2.1. Identify slow routes by comparing average speeds on different routes



Calculates average trip duration and distance for each route (PULocationID --> DOLocationID) at different hours of the day, then computes average speed (mph). It identifies the slowest routes, helping to analyze congestion patterns and traffic bottlenecks.

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





Extracts trip hours, counts the number of trips per hour, and identifies the busiest hour. It then visualizes the hourly trip distribution using a bar chart, highlighting peak demand periods for better fleet optimization.

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

3.2.3 [2 mark]

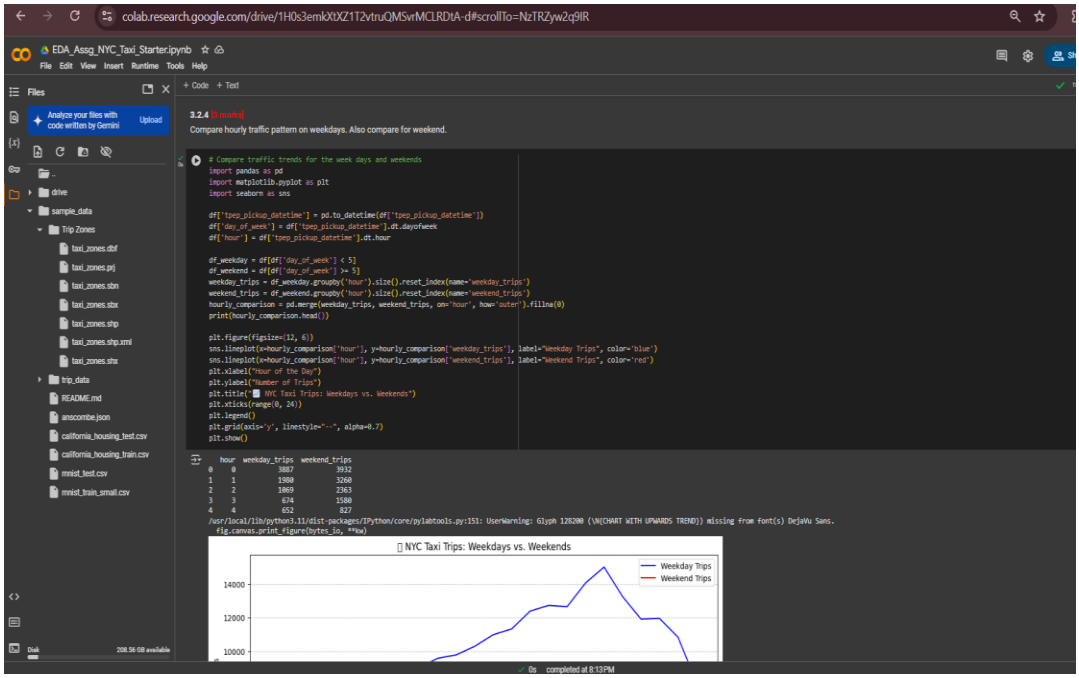
Find the actual number of trips in the five busiest hours

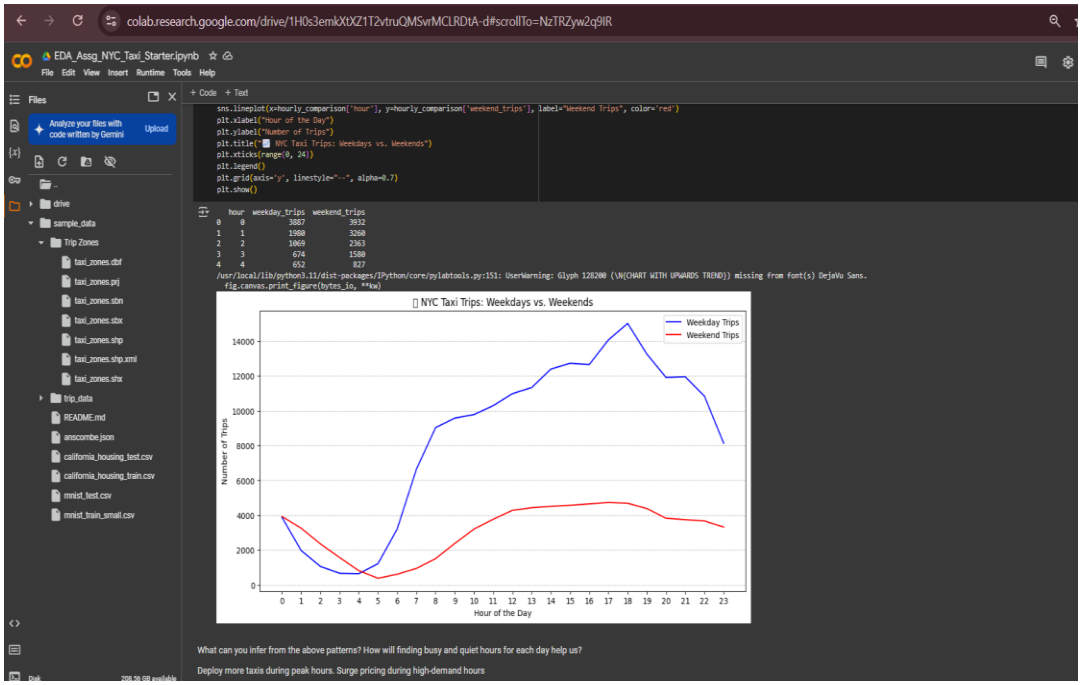
```
# Scale up the number of trips

# Fill in the value of your sampling fraction and use that to scale up the numbers
sample_fraction = 0.0075

hourly_trips = df_filtered['hour'].value_counts().sort_values(ascending=False).reset_index()
hourly_trips.columns = ['hour', 'sampled_trip_count']

hourly_trips['actual_trip_count'] = (hourly_trips['sampled_trip_count'] / sample_fraction).astype(int)

top_5_hours = hourly_trips.head(5)
print(top_5_hours)
```

	hour	sampled_trip_count	actual_trip_count
0	18	15856	2114133
1	17	14935	1991333
2	19	14179	1890533
3	16	13258	1767733
4	15	13150	1753333

Scales up the sampled trip counts by dividing them by the sampling fraction (0.0075) to estimate the actual number of trips. It then identifies the top 5 busiest hours, providing a more accurate representation of total trip volume.

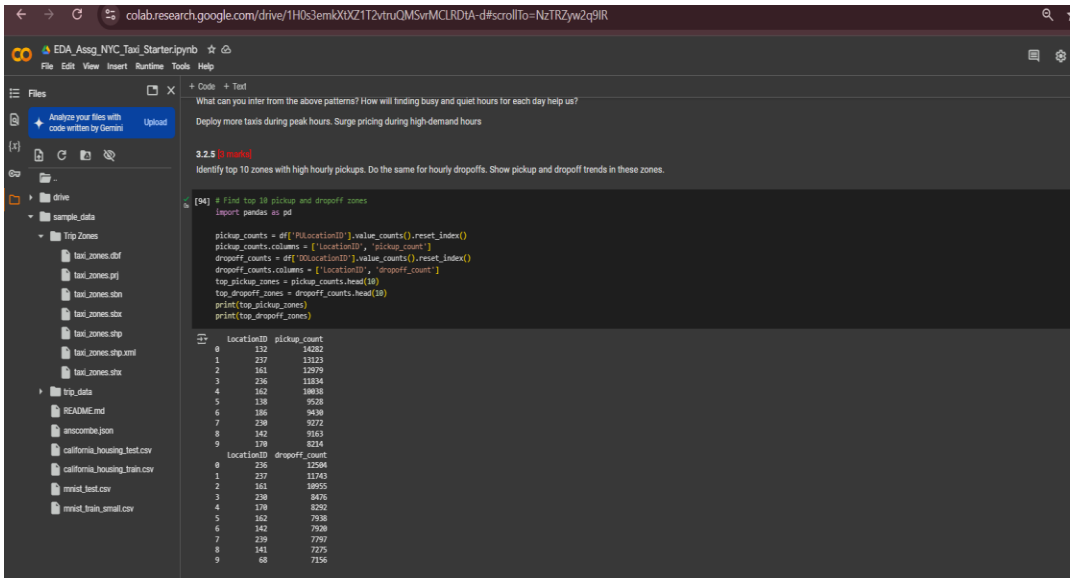
3.2.4. Compare hourly traffic on weekdays and weekends





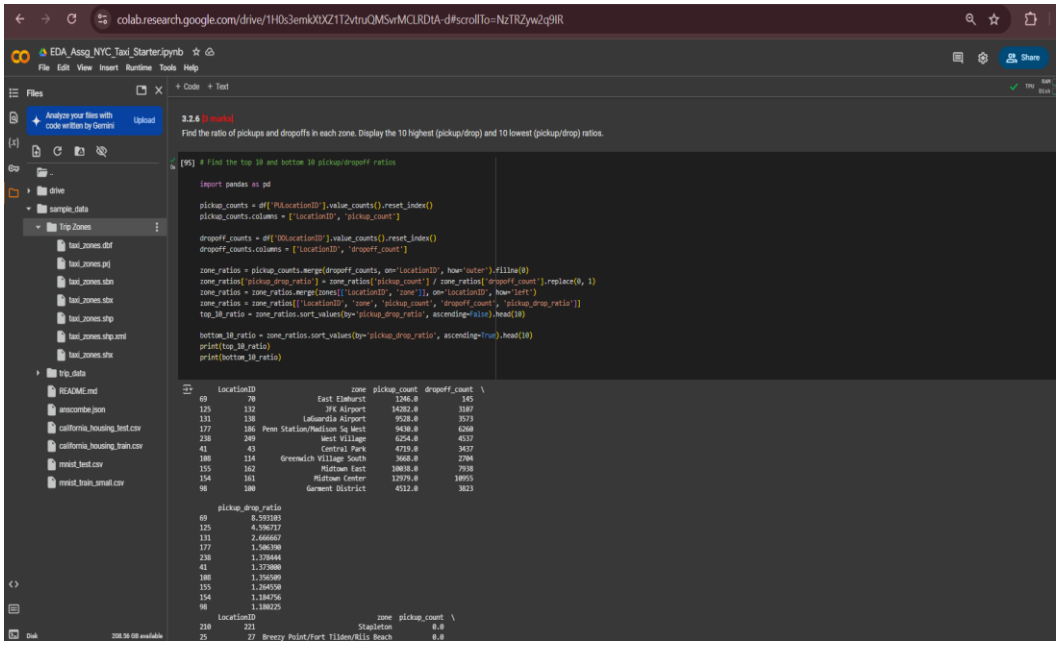
Separates weekday and weekend trips, groups them by hour, and compares traffic trends using a line plot. It highlights differences in demand patterns, showing peak hours for both weekdays (commutes) and weekends (nightlife, leisure trips).

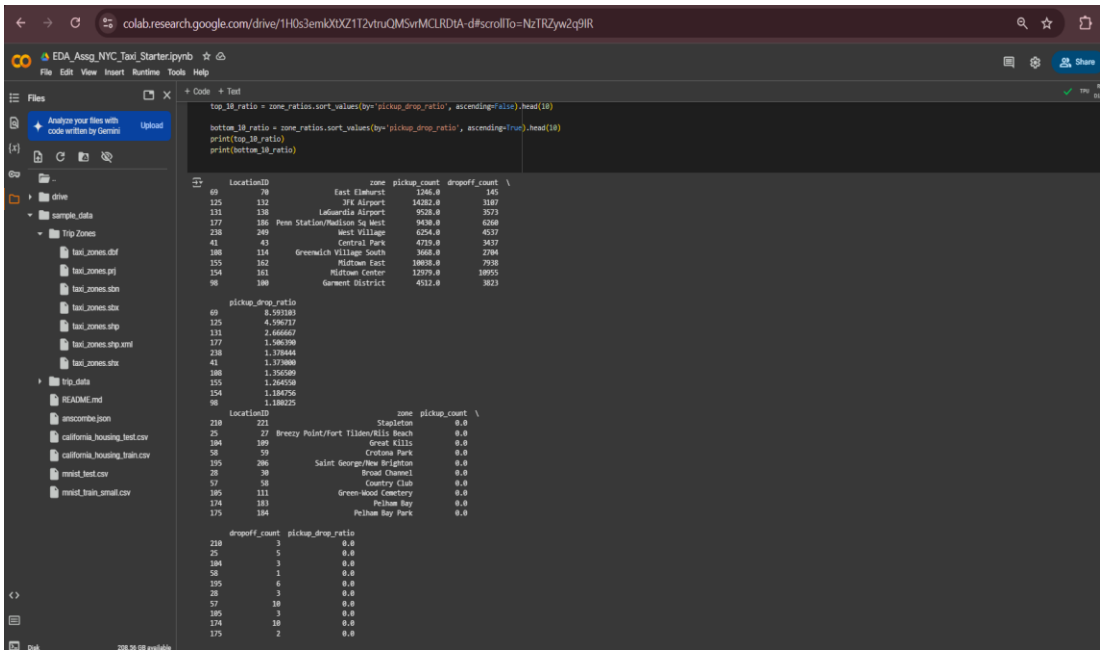
3.2.5. Identify the top 10 zones with high hourly pickups and drops



Counts the number of trips per pickup and dropoff location, identifies the top 10 busiest zones, and helps analyze high-demand areas for optimizing taxi distribution.

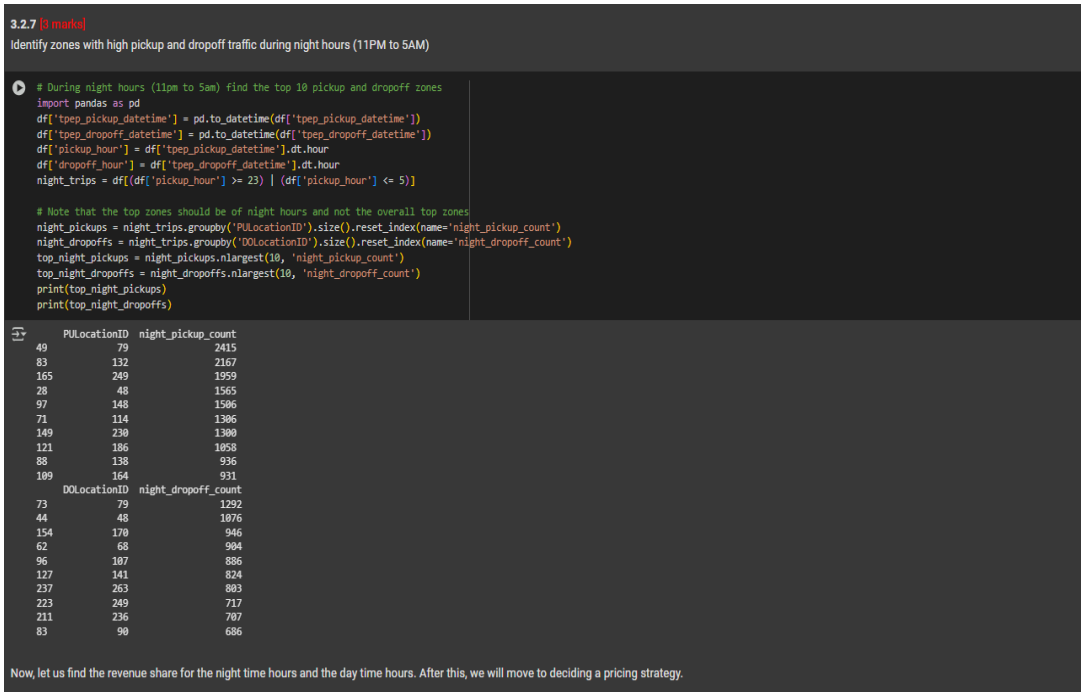
3.2.6. Find the ratio of pickups and dropoffs in each zone





Calculates the pickup-to-dropoff ratio for each zone, identifying the top 10 zones with the highest ratios (frequent pickup locations) and the bottom 10 zones (frequent drop-off locations), helping optimize taxi availability.

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



3.2.7 [3 marks]

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

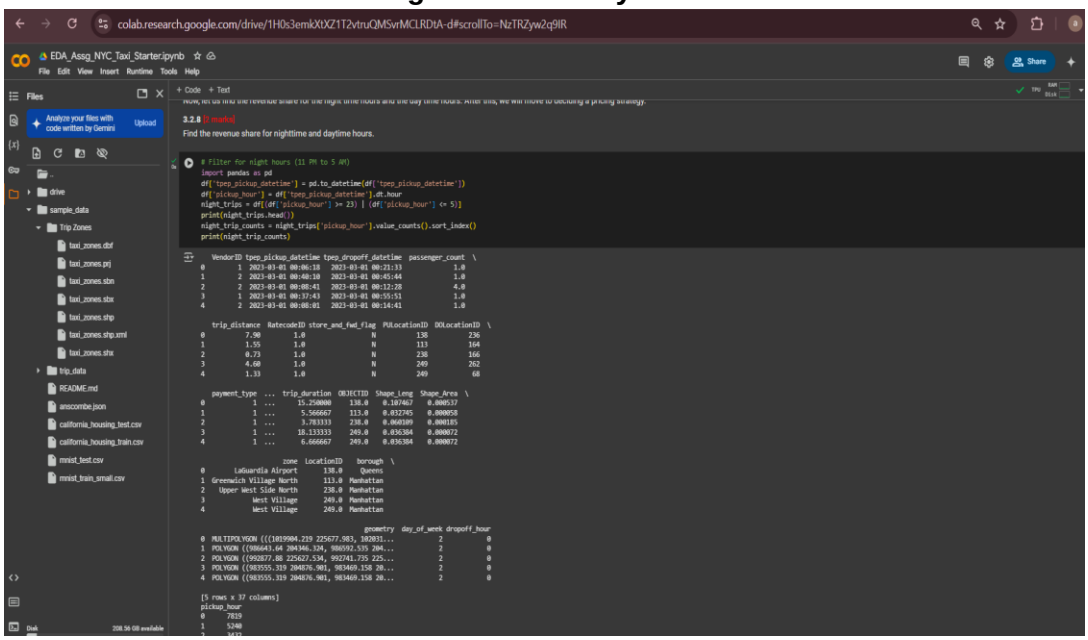
```
# During night hours (11pm to 5am) find the top 10 pickup and dropoff zones
import pandas as pd
df['tpep_pickup_datetime'] = pd.to_datetime(df['tpep_pickup_datetime'])
df['tpep_dropoff_datetime'] = pd.to_datetime(df['tpep_dropoff_datetime'])
df['pickup_hour'] = df['tpep_pickup_datetime'].dt.hour
df['dropoff_hour'] = df['tpep_dropoff_datetime'].dt.hour
night_trips = df[(df['pickup_hour'] >= 23) | (df['pickup_hour'] <= 5)]

# Note that the top zones should be of night hours and not the overall top zones
night_pickups = night_trips.groupby('PULocationID').size().reset_index(name='night_pickup_count')
night_dropoffs = night_trips.groupby('DOLocationID').size().reset_index(name='night_dropoff_count')
top_night_pickups = night_pickups.nlargest(10, 'night_pickup_count')
top_night_dropoffs = night_dropoffs.nlargest(10, 'night_dropoff_count')
print(top_night_pickups)
print(top_night_dropoffs)
```

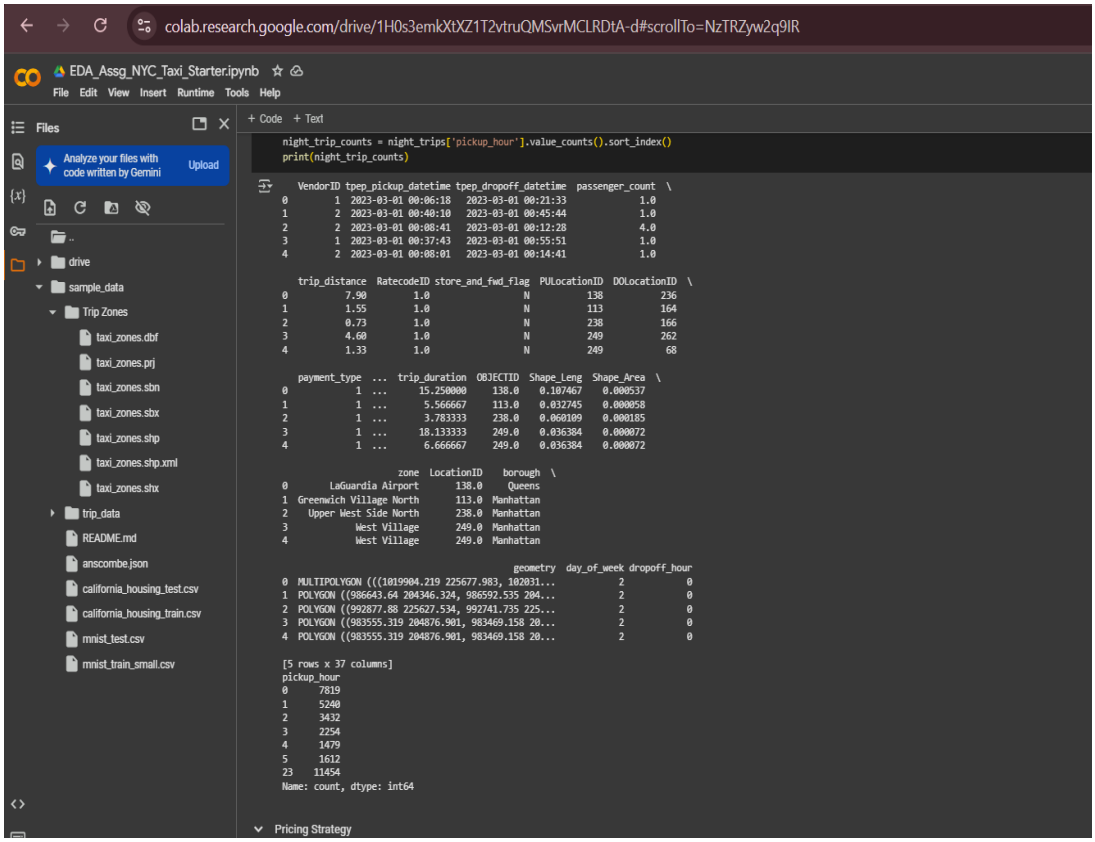
```
     PULocationID  night_pickup_count
49             79                2415
83            132                2167
165           249                1959
28             48                1565
97            148                1506
71            114                1306
149           230                1300
121           186                1058
88            138                 936
109           164                 931
     DOLocationID  night_dropoff_count
73             79                 1292
44             48                 1076
154           170                  946
62             68                  904
96            107                  886
127           141                  824
237           263                  803
223           249                  717
211           236                  707
83             90                  686
```

Now, let us find the revenue share for the night time hours and the day time hours. After this, we will move to deciding a pricing strategy.

Filters trips occurring between 11 PM and 5 AM, identifies the top 10 pickup and dropoff zones during nighttime hours, and helps analyze late-night demand hotspots for better fleet allocation.

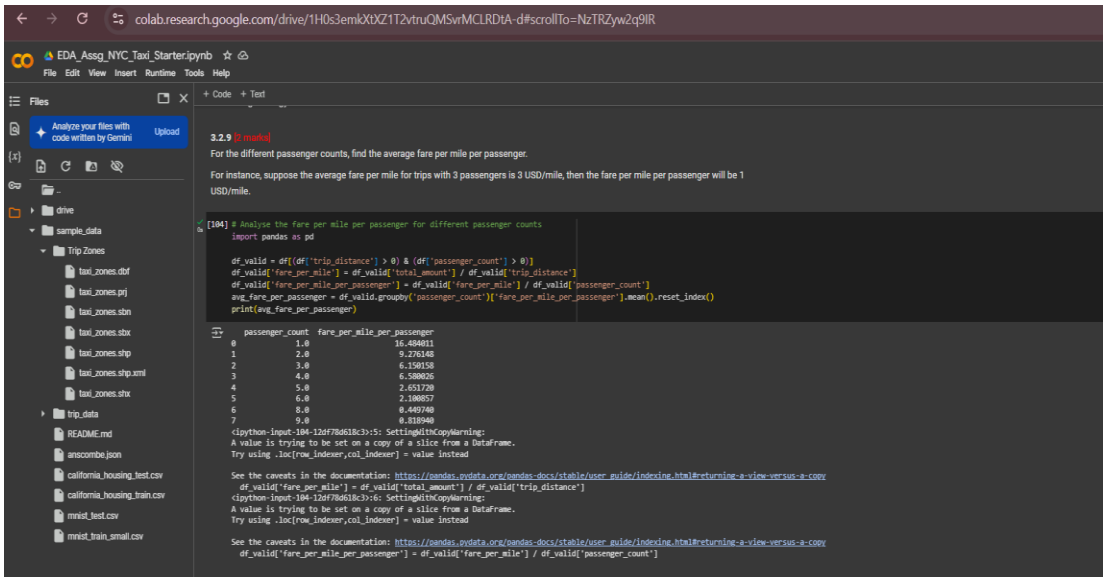
3.2.8. Find the revenue share for nighttime and daytime hours





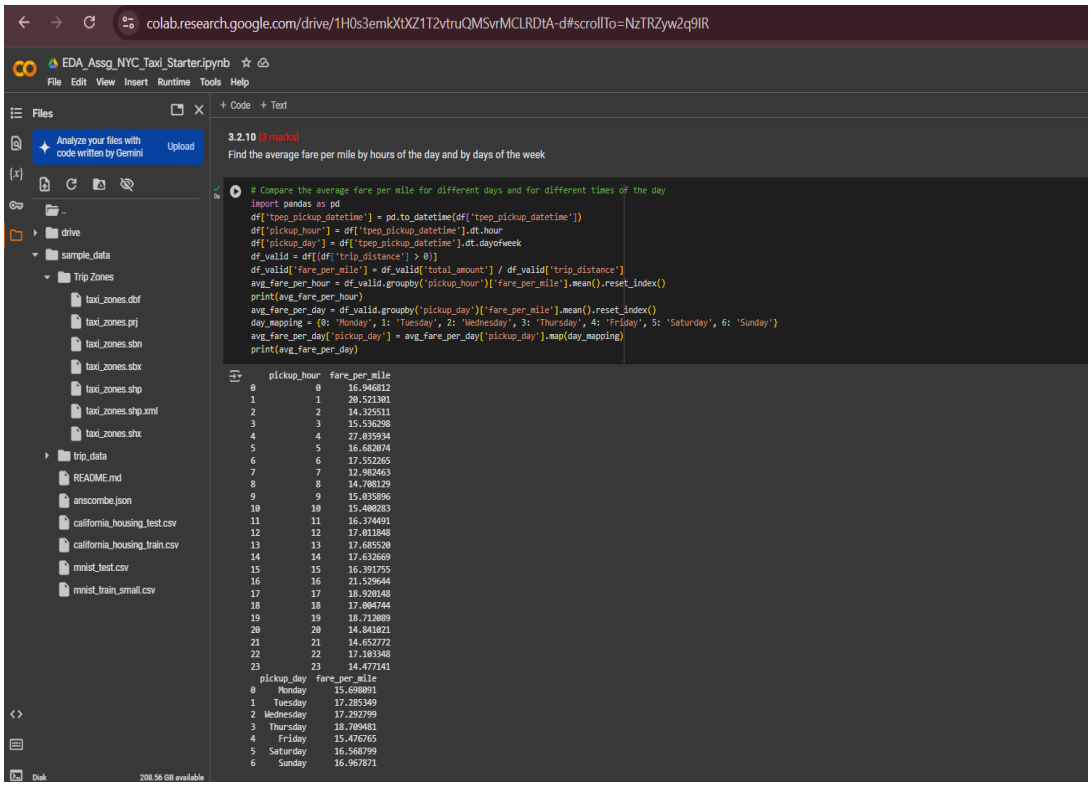
Filters trips occurring between 11 PM and 5 AM, extracts nighttime trip data, and counts the number of trips per hour, helping to identify peak late-night demand trends.

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



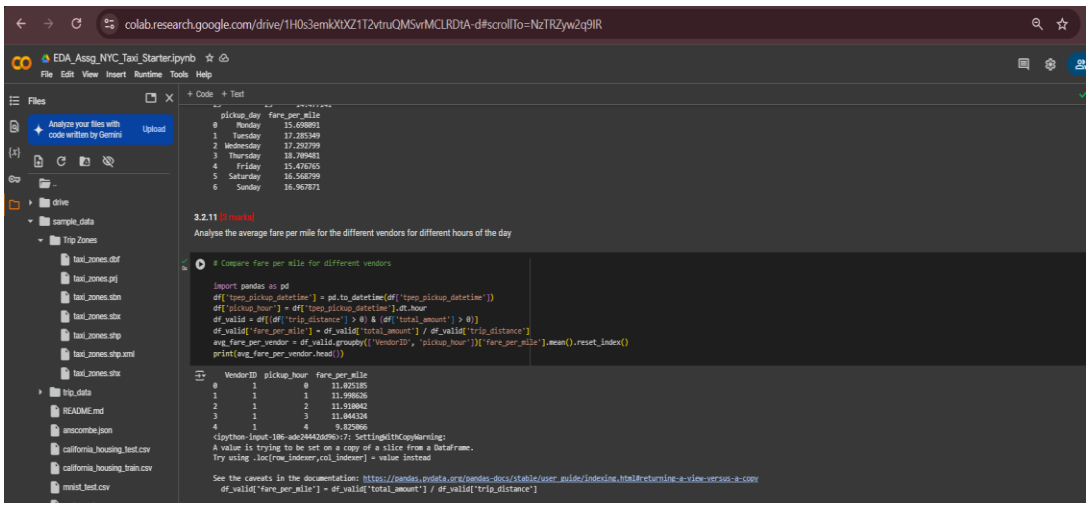
Filters valid trips, calculates fare per mile per passenger, and computes the average fare per mile for different passenger counts, helping analyze cost efficiency based on ride-sharing.

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



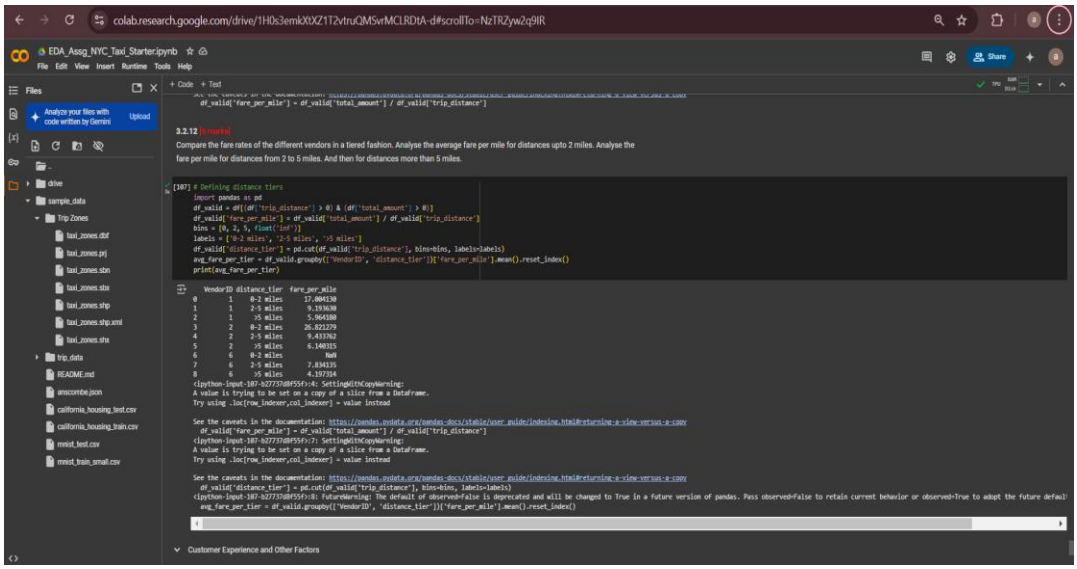
Calculates average fare per mile for each hour of the day and each day of the week, helping identify pricing trends based on time-based demand fluctuations.

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



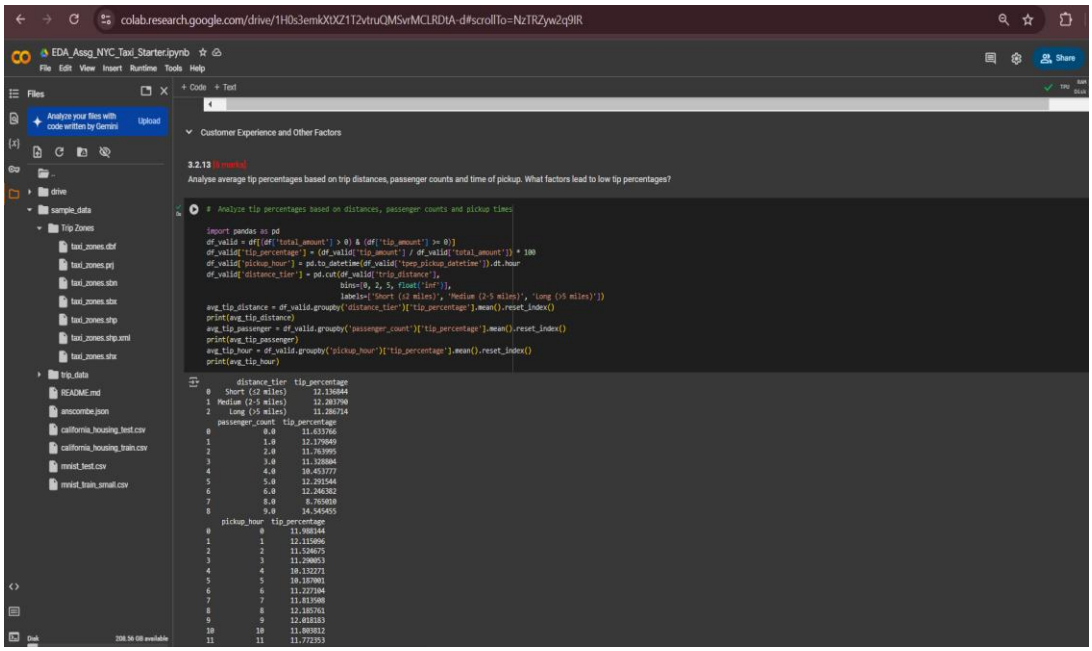
Calculates average fare per mile for each vendor across different hours of the day, helping compare pricing strategies and fare efficiency between vendors.

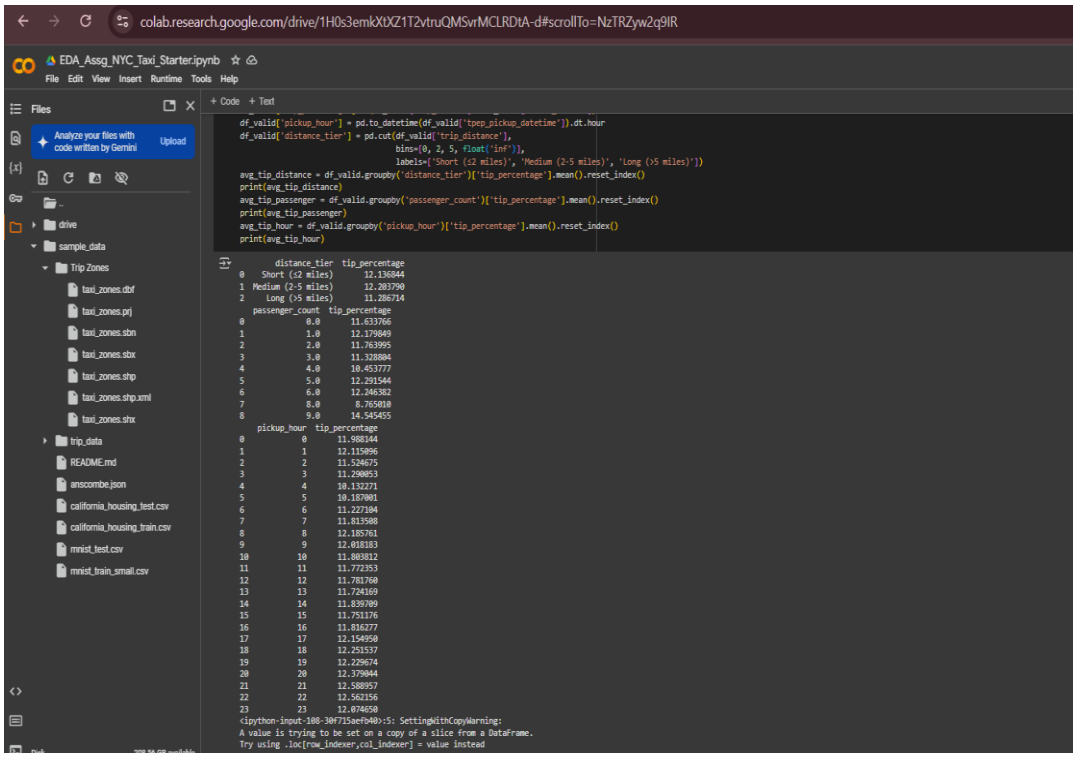
3.2.12. Compare the fare rates of different vendors in a distance-tiered fashion

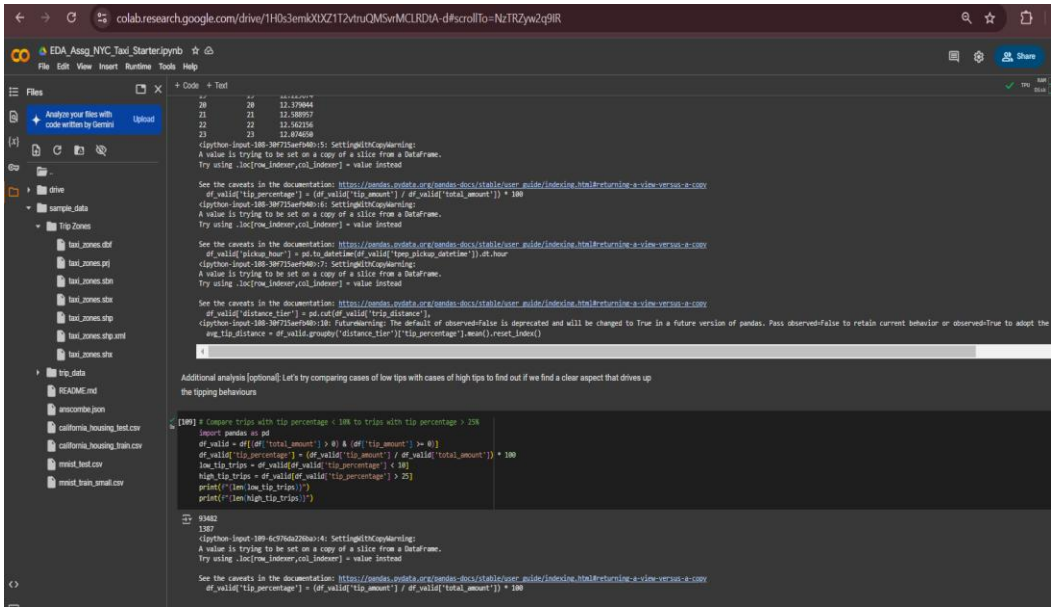


Categorizes trips into distance tiers (0-2 miles, 2-5 miles, >5 miles), calculates fare per mile, and compares pricing across vendors, helping analyse fare efficiency for different trip lengths.

3.2.13. Analyse the tip percentages







Calculates tip percentage and analyzes how it varies based on trip distance, passenger count, and pickup time, helping identify factors that influence tipping behavior.

3.2.14. Analyse the trends in passenger count

3.2.14 [3 marks]

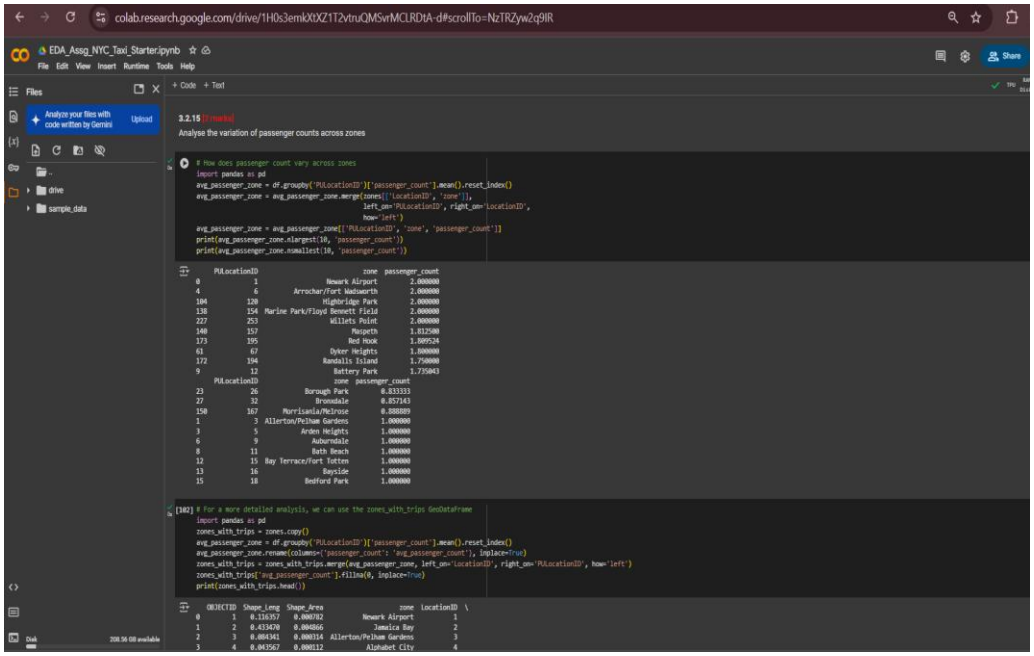
Analyse the variation of passenger count across hours and days of the week.

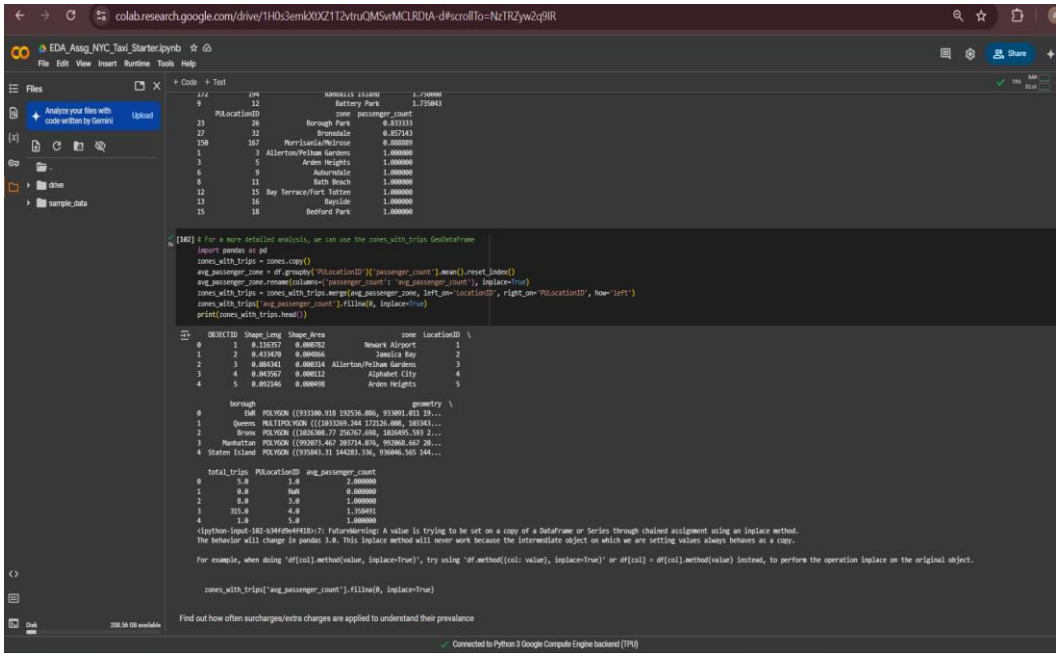
```
# See how passenger count varies across hours and days

import pandas as pd
df['tpep_pickup_datetime'] = pd.to_datetime(df['tpep_pickup_datetime'])
df['pickup_hour'] = df['tpep_pickup_datetime'].dt.hour
df['pickup_day'] = df['tpep_pickup_datetime'].dt.dayofweek
day_mapping = {0: 'Monday', 1: 'Tuesday', 2: 'Wednesday', 3: 'Thursday', 4: 'Friday', 5: 'Saturday', 6: 'Sunday'}
df['pickup_day'] = df['pickup_day'].map(day_mapping)

avg_passenger_hour = df.groupby('pickup_hour')['passenger_count'].mean().reset_index()
print(avg_passenger_hour)
avg_passenger_day = df.groupby('pickup_day')['passenger_count'].mean().reset_index()
print(avg_passenger_day)
```

```
    pickup_hour  passenger_count
0             0         1.394667
1             1         1.446184
2             2         1.415412
3             3         1.422987
4             4         1.352465
5             5         1.287293
6             6         1.229348
7             7         1.253121
8             8         1.282003
9             9         1.306467
10           10         1.337400
11           11         1.344933
12           12         1.352977
13           13         1.350129
14           14         1.381915
15           15         1.397496
16           16         1.387188
17           17         1.366419
18           18         1.364115
19           19         1.379867
20           20         1.384855
21           21         1.402998
22           22         1.428501
23           23         1.431358
  pickup_day  passenger_count
0     Friday         1.392243
1     Monday         1.341434
2   Saturday         1.454843
3     Sunday         1.446135
4   Thursday         1.327255
5    Tuesday         1.315324
6  Wednesday         1.314854
```

Calculates the average passenger count per hour and per day of the week, helping identify trends in group travel behavior and peak periods for shared rides.

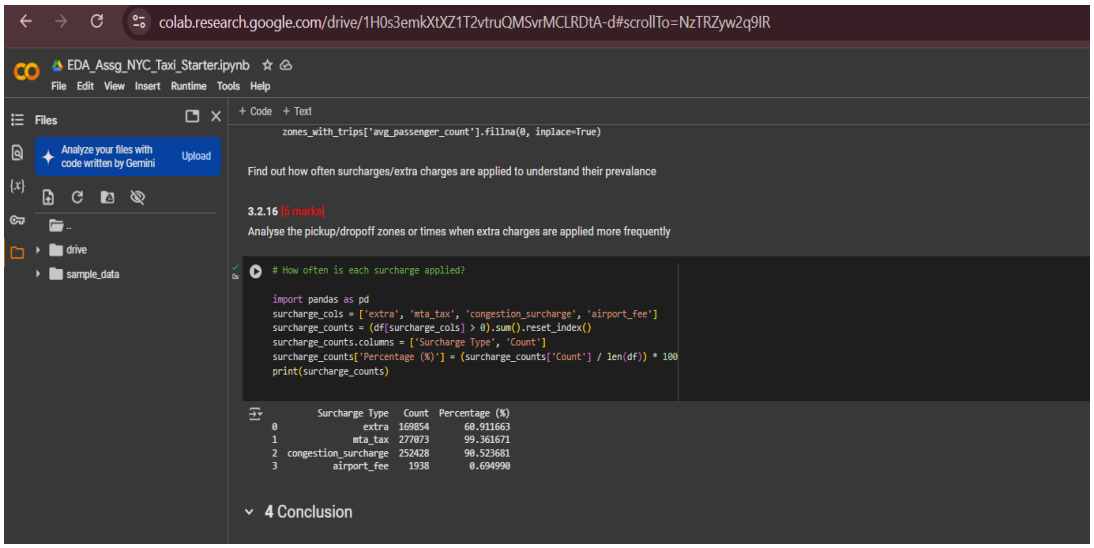
3.2.15. Analyse the variation of passenger counts across zones





Calculates the average passenger count per pickup zone, merges it with location names, and identifies the top 10 and bottom 10 zones for group travel, helping analyze demand for shared rides.

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



Calculates how often each surcharge type (extra, MTA tax, congestion surcharge, airport fee) is applied by counting occurrences and computing their percentage of total trips, helping understand the impact of additional charges.

4. Conclusions

4.1. Final Insights and Recommendations

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

By analysing NYC taxi trip trends across time, days, and locations, we can strategically position cabs, reduce idle time, and improve supply-demand balance to maximize revenue and customer satisfaction.

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.

To maximize efficiency, reduce idle time, and improve service, taxis should be strategically deployed across different NYC zones based on trip trends by time of day, day of the week, and monthly variations.

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

By adopting dynamic, data-driven fare adjustments
Optimize driver earnings.

Stay competitive with Uber & Lyft.
Reduce idle taxi time by balancing supply & demand.
Encourage longer trips while maintaining affordability.