EDA PROJECT (INT353)

ON

Uber Supply Demand Gap

Name: Piyush Kumar

Registration No.: 12015922

Program Name: B. Tech CSE Data Science (AI & ML)

Course Code: INT353

Under The Guidance Of

Mr. Shahgil Jamal

School of Computer Science & Engineering

Lovely Professional University, Phagwara

INDEX

Topic Knowledge

My dataset:-**Uber Supply Demand Gap**

Introduction

In the ever-evolving landscape of transportation and ride-sharing services, understanding the intricate interplay between supply and demand is paramount. Uber, one of the pioneers in this industry, has been at the forefront of revolutionizing urban mobility. To optimize its operations and enhance user experiences, Uber has collected and made available a wealth of data related to its service requests, including details about trip requests, driver availability, and trip outcomes.

The Uber Supply and Demand Gap dataset presents a valuable opportunity for exploratory data analysis (EDA) to delve into this dynamic ecosystem. This dataset encompasses a range of attributes associated with each customer request, including request IDs, timestamps, pick-up and drop-off locations, driver IDs, and the final status of the trip.

This analysis seeks to unravel the intricacies of how Uber manages the delicate balance between supply and demand. By examining trends, patterns, and key insights within this dataset, we aim to gain a deeper understanding of the challenges and opportunities that Uber encounters in optimizing its service delivery.

Moreover, this analysis can shed light on how various factors, such as time of day, location, and external influences like weather conditions, impact the availability of drivers and the satisfaction of Uber's customer base. It also provides a platform for investigating the effectiveness of pricing strategies, the reliability of drivers, and the overall efficiency of the Uber ecosystem.

In the following exploration of the Uber Supply and Demand Gap dataset, we will pose questions, perform data visualizations, and employ statistical techniques to extract meaningful insights. These insights can not only benefit Uber in refining its operations but can also contribute to a broader understanding of the ride-sharing industry's dynamics and customer behavior in urban transportation.

Data Understanding

There are six attributes associated with each request made by a customer:

- Request id: A unique identifier of the request
- Time of request: The date and time at which the customer made the trip request
- Drop-off time: The drop-off date and time, in case the trip was completed
- Pick-up point: The point from which the request was made
- Driver id: The unique identification number of the driver
- Status of the request: The final status of the trip, that can be either completed, cancelled by the driver or no cars available

Note:

In the analysis, only the trips to and from the airport are being considered.

Reason to choose this topic

Choosing the Uber Supply and Demand Gap dataset for an exploratory data analysis (EDA) project offers several compelling reasons:

- Relevance: It's directly related to Uber's operations, making it highly relevant to real-world business scenarios.
- Practicality: Insights can be applied to ride-sharing businesses. It can help in optimizing driver allocation, pricing strategies, and overall service efficiency.
- Rich Data: Offers diverse attributes for analysis.
- Customer Impact: Shows how gaps affect user experience.
- Business Strategy: Reveals supply-demand balancing strategies.
- Data Visualization: Enables compelling visual representations.
- Educational Value: Great for learning with real-world data.
- Problem-Solving: Addresses common transportation challenges.
- Data Availability: Accessible, well-documented dataset.

Uber is facing driver cancellation and non- availability of cabs to and from airport leading to impact on the business and loss of potential revenue.
So, main reason is to identify the root cause of the supply-demand gap of cabs to and from airport.

Library used in this Project

The libraries `numpy`, `pandas`, `datetime`and `matplotlib` are commonly used in data analysis and visualization tasks, and I used these libraries in my dataset analysis. Here's an explanation of each of these libraries and their typical use cases in data analysis:

1. numpy (Numerical Python):

- `numpy` is a fundamental library for numerical computing in Python.
- It provides support for arrays and matrices, making it easier to perform mathematical operations on data.
- In data analysis, `numpy` is often used for data manipulation, mathematical calculations, and working with multi-dimensional arrays.

2. pandas:

- `pandas` is a powerful library for data manipulation and analysis.
- It provides data structures like DataFrames and Series, which are especially useful for handling structured data like your Uber dataset.
- we can use `pandas` for tasks like reading data from various sources (e.g., CSV, Excel), data cleaning, filtering, grouping, aggregation, and more.

3. datetime:

- The `datetime` module is part of Python's standard library and is used for working with dates and times.
- In my dataset analysis, I used `datetime` to convert timestamp columns to datetime objects, making it easier to analyze and manipulate time-related data.

4. matplotlib:

- `matplotlib` is a popular data visualization library in Python.
- It provides a wide range of tools for creating static, animated, or interactive plots and charts.

- ➢ It use to create various plots and charts to visualize your data, such as line charts and bar plots.

5.seaborn:

- ➢ seaborn is a higher-level data visualization library built on top of matplotlib.
- ➢ It provides a more user-friendly interface for creating aesthetically pleasing statistical graphics.
- ➢ seaborn is often used to create more complex and informative visualizations with less code compared to matplotlib.

In my dataset analysis, I used seaborn for more advanced and customized data visualization, while matplotlib provides the foundation for creating basic plots.

These libraries are essential tools for data analysts and data scientists to explore, clean, analyze, and visualize data effectively, and they play a significant role in my data analysis process.

Questions for Analysis

1. What are the names and data types of the columns?

=> Request id: A unique identifier for each request.

Pickup point: The location where the rider wants to be picked up.

Driver id: A unique identifier for each driver.

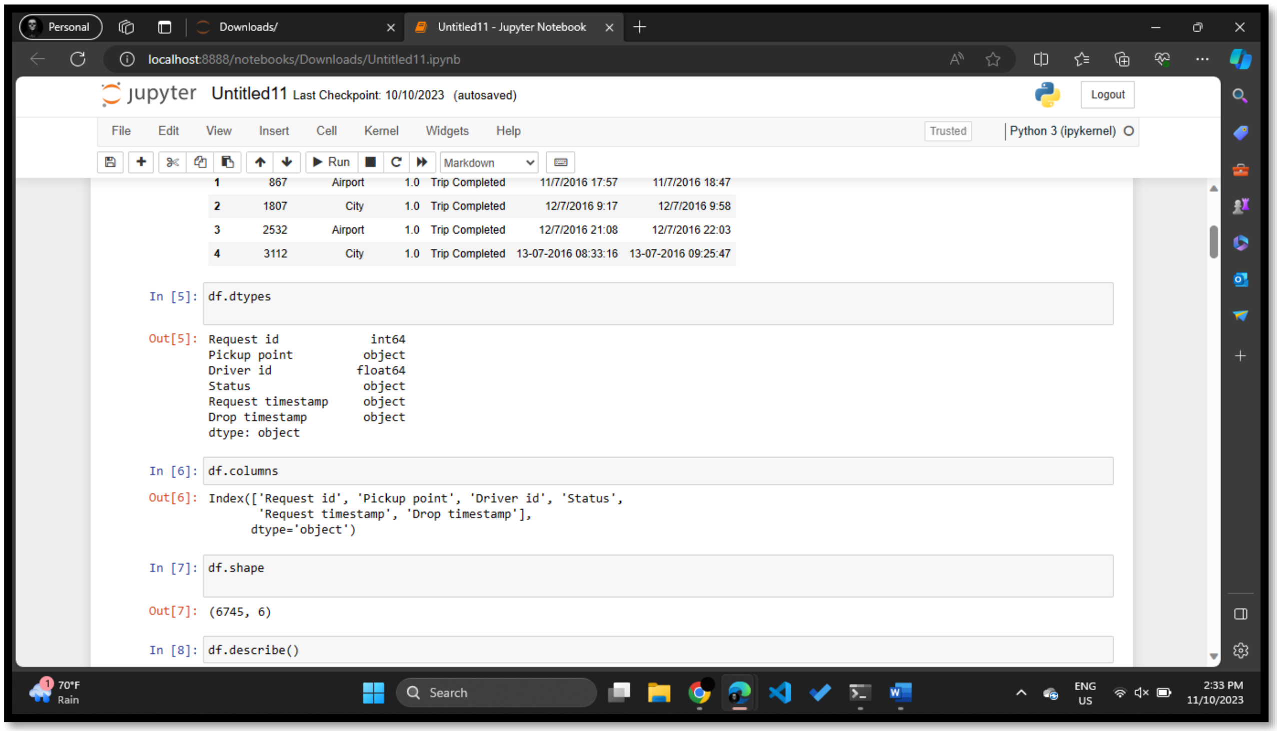
Status: The status of the request (e.g., Completed, Cancelled, etc.).

Request timestamp: The time at which the request was made.

Drop timestamp: The time at which the rider was dropped off at their destination.

Explanation:

This dataset can be used to analyze Uber supply and demand. For example, we can use it to identify the most popular pickup and drop-off locations, the busiest times of day, and the most common reasons for cancellations. This information can be used to improve Uber's service and to make it more efficient.



1	867	Airport	1.0	Trip Completed	11/7/2016 17:57	11/7/2016 18:47
2	1807	City	1.0	Trip Completed	12/7/2016 9:17	12/7/2016 9:58
3	2532	Airport	1.0	Trip Completed	12/7/2016 21:08	12/7/2016 22:03
4	3112	City	1.0	Trip Completed	13-07-2016 08:33:16	13-07-2016 09:25:47

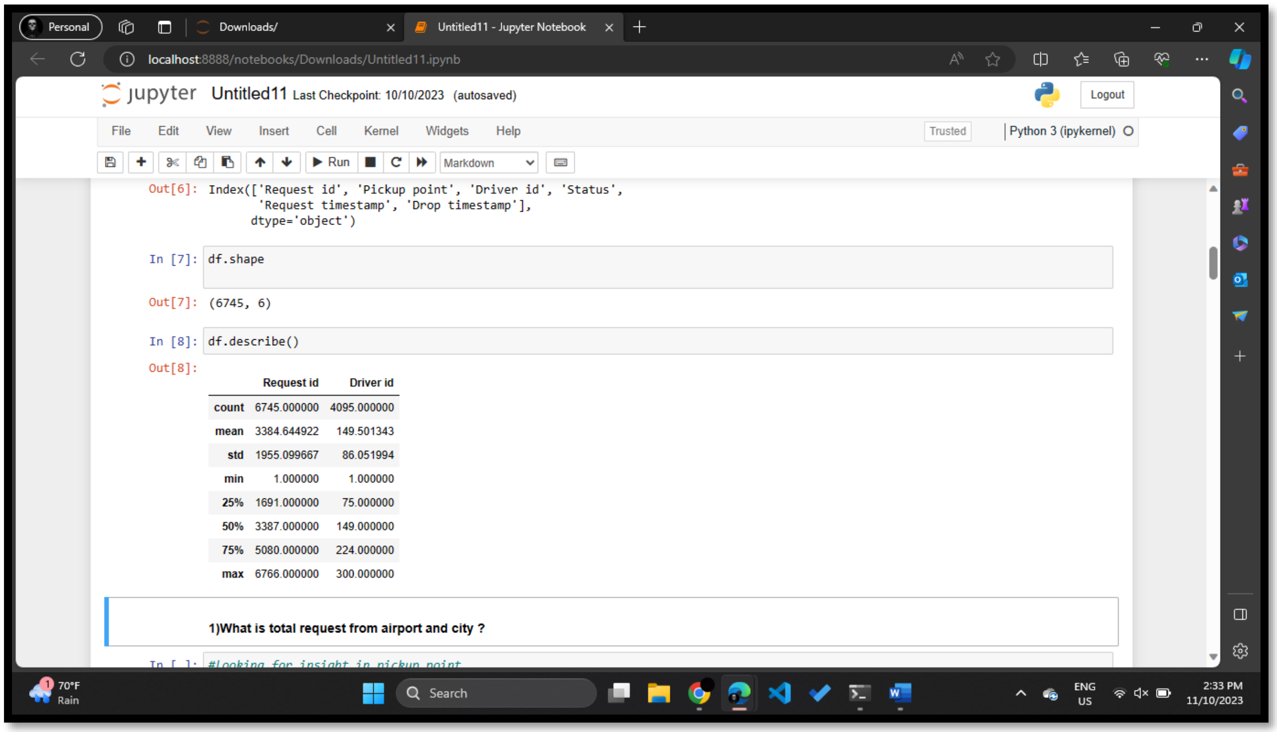
```
In [5]: df.dtypes

Out[5]: Request id              int64
        Pickup point           object
        Driver id             float64
        Status                 object
        Request timestamp      object
        Drop timestamp         object
        dtype: object

In [6]: df.columns

Out[6]: Index(['Request id', 'Pickup point', 'Driver id', 'Status',
               'Request timestamp', 'Drop timestamp'],
              dtype='object')

In [7]: df.shape

Out[7]: (6745, 6)

In [8]: df.describe()
```

2.What are the basic summary statistics?

=> The basic summary statistics of the dataset include:

- Count: The number of values in each column.
- Unique: The number of unique values in each column.
- Top: The most frequent value in each column.
- Freq: The frequency of the most frequent value in each column.
- Mean: The average value in each column.
- Std: The standard deviation of the values in each column.
- Min: The minimum value in each column.
- Max: The maximum value in each column.

We can use these statistics to get a better understanding of the dataset and identify any potential outliers or anomalies.


Personal
Downloads/
Untitled11 - Jupyter Notebook
localhost:8888/notebooks/Downloads/Untitled11.ipynb
jupyter
Untitled11 Last Checkpoint: 10/10/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Markdown
70°F
Rain
Search
ENG
US
2:33 PM
11/10/2023

```
Out[6]: Index(['Request id', 'Pickup point', 'Driver id', 'Status',
               'Request timestamp', 'Drop timestamp'],
              dtype='object')

In [7]: df.shape

Out[7]: (6745, 6)

In [8]: df.describe()

Out[8]:
```

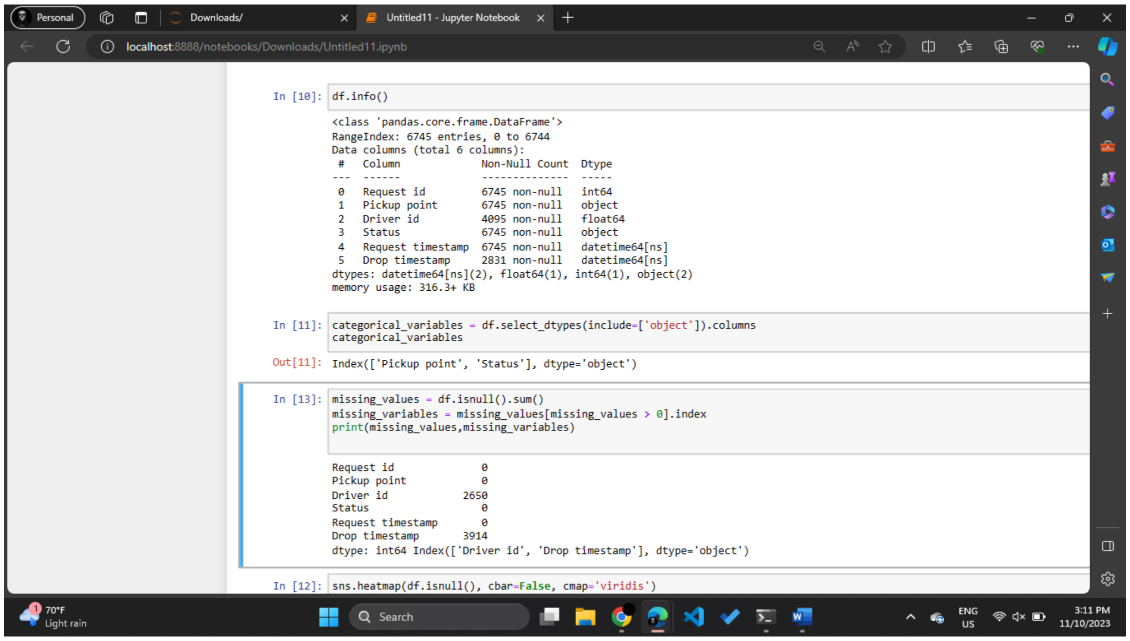
	Request id	Driver id
count	6745.000000	4095.000000
mean	3384.644922	149.501343
std	1955.099667	86.051994
min	1.000000	1.000000
25%	1691.000000	75.000000
50%	3387.000000	149.000000
75%	5080.000000	224.000000
max	6766.000000	300.000000

1)What is total request from airport and city ?

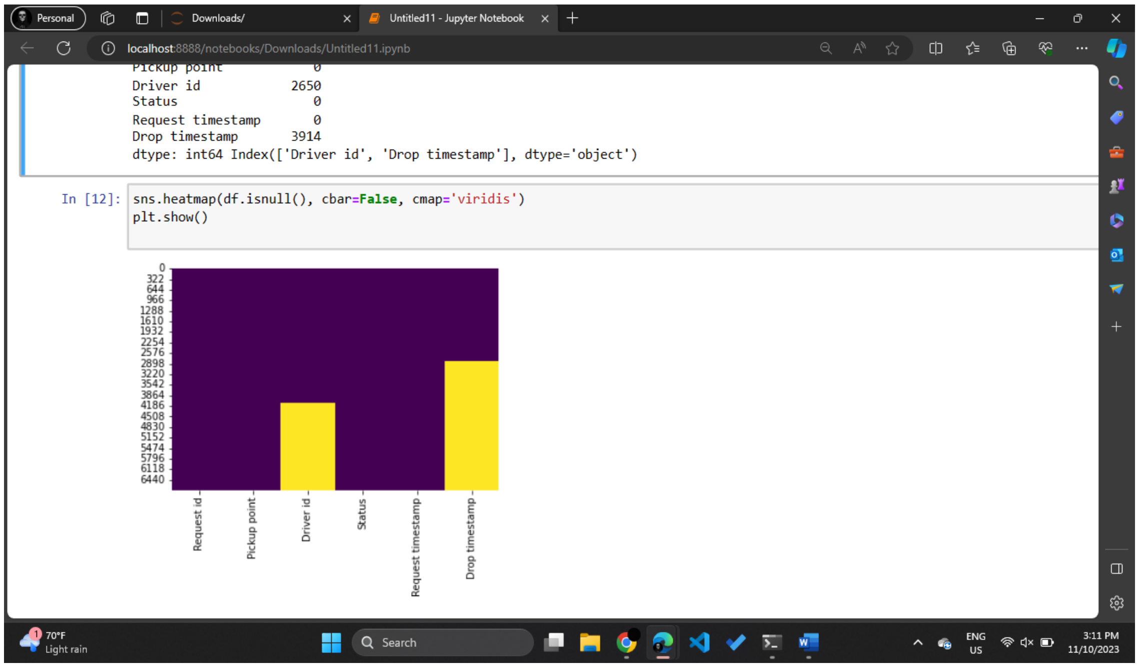
```
In [ ]: #looking for insight in pickup point
```

3.Are there any categorical variables and missing values? If so, print it.

=>This output shows that there are two categorical variables in the dataset (Pickup point, Status) and 2650 missing values in driver id and 3914 in drop timestamp



```
In [10]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6745 entries, 0 to 6744
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Request id         6745 non-null   int64
 1   Pickup point       6745 non-null   object
 2   Driver id          4095 non-null   float64
 3   Status             6745 non-null   object
 4   Request timestamp  6745 non-null   datetime64[ns]
 5   Drop timestamp     2831 non-null   datetime64[ns]
dtypes: datetime64[ns](2), float64(1), int64(1), object(2)
memory usage: 316.3+ KB

In [11]: categorical_variables = df.select_dtypes(include=['object']).columns
         categorical_variables

Out[11]: Index(['Pickup point', 'Status'], dtype='object')

In [13]: missing_values = df.isnull().sum()
         missing_variables = missing_values[missing_values > 0].index
         print(missing_values,missing_variables)

Request id              0
Pickup point            0
Driver id            2650
Status                  0
Request timestamp       0
Drop timestamp       3914
dtype: int64 Index(['Driver id', 'Drop timestamp'], dtype='object')

In [12]: sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
```



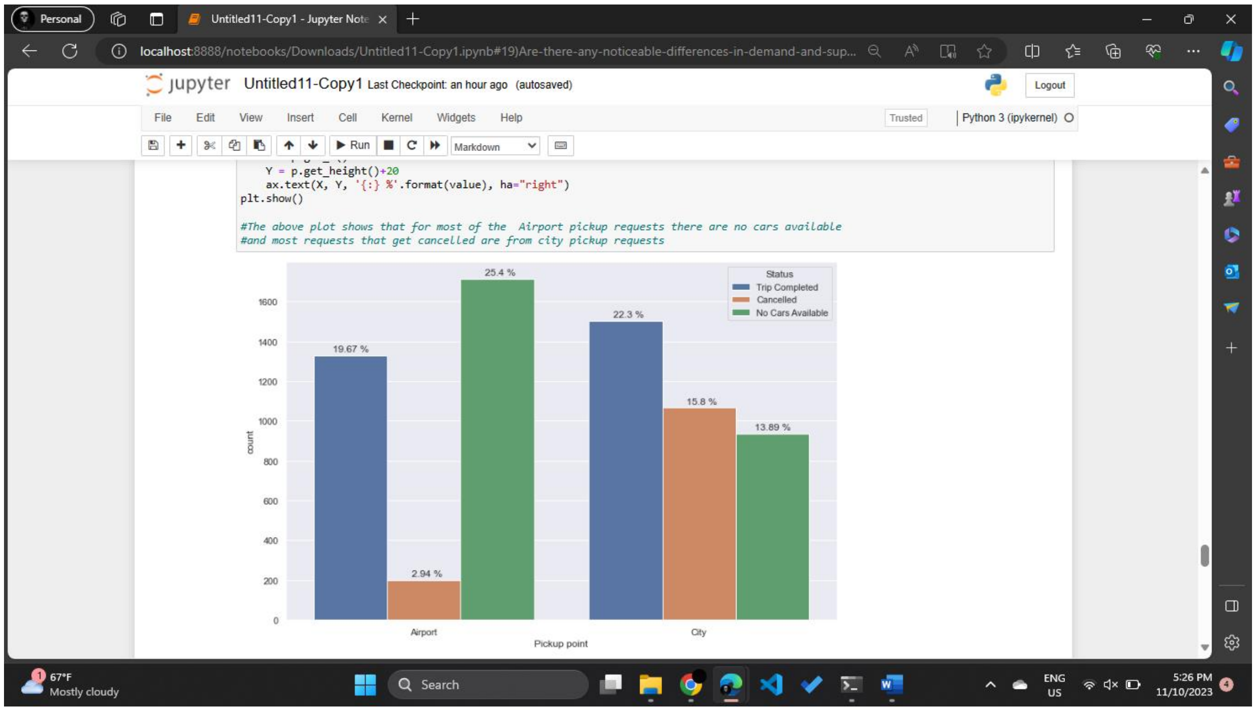
```
Pickup point            0
Driver id            2650
Status                  0
Request timestamp       0
Drop timestamp       3914
dtype: int64 Index(['Driver id', 'Drop timestamp'], dtype='object')

In [12]: sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
         plt.show()
```

4.Are there any outliers in the data? If so, use box plots, histograms and visualize.

=>

5.Is the data balanced or imbalanced? Visualize.

=> To determine if the data is balanced or imbalanced, we can visualize the distribution of the "Status" column, which indicates the outcome of each ride request. We can use a count plot to see the frequency of each status category. If the counts are roughly equal, the data is balanced; if not, it's imbalanced. Since here there's a significant difference in counts between categories, it suggests an imbalanced dataset.



```
    Y = p.get_height()+20
    ax.text(X, Y, '{:} %'.format(value), ha="right")
plt.show()

#The above plot shows that for most of the  Airport pickup requests there are no cars available
#and most requests that get cancelled are from city pickup requests
```

Untitled11-Copy1 - Jupyter Note
localhost:8888/notebooks/Downloads/Untitled11-Copy1.ipynb#19)Are-there-any-noticeable-differences-in-demand-and-sup...
jupyter Untitled11-Copy1 Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Markdown
Status
🟦 Trip Completed
🟧 Cancelled
🟩 No Cars Available
25.4 %
22.3 %
19.67 %
15.8 %
13.89 %
2.94 %
1600
1400
1200
1000
800
600
400
200
0
count
Airport
City
Pickup point
67°F
Mostly cloudy
Search
ENG
US
5:26 PM
11/10/2023

6.What is the target variable (if any).

=> The target variable in the Uber supply demand dataset is Status. This is because the goal of this analysis is to identify the root cause of the problem with Uber cancellations and car unavailability, and to recommend ways to improve the situation. The Status variable indicates whether the request was completed, cancelled, or had some other outcome.

Here is a breakdown of the possible values for the Status variable:

- Completed: The request was completed successfully.
- Cancelled: The request was cancelled, either by the rider or the driver.
- No Available Cars: There were no available cars to pick up the rider.

The Status variable can be used to identify the most common reasons for cancellations and car unavailability. This information can then be used to develop strategies for improving Uber's service. For example, if it is found that the most common reason for cancellations is that riders are late, Uber could implement a policy of charging riders a cancellation fee if they are more than a certain number of minutes late.

Overall, the Status variable is the target variable in the Uber supply demand dataset because it is the variable that we are interested in predicting and improving.

7.What are the units of measurement for numerical columns? (example: time, currency, date, distance)

=> The units of measurement for the numerical columns in the Uber supply demand dataset are as follows:

- Request id: Integer
- Pickup point: String
- Driver id: Integer
- Status: String
- Request timestamp: Datetime
- Drop timestamp: Datetime.

The Request id and Driver id columns are simply unique identifiers for the request and driver, respectively. The Status column is a categorical variable that indicates the status of the request (e.g., Completed, Cancelled, etc.). The Request timestamp and Drop timestamp columns are datetime variables that indicate the time at which the request was made and the time at which the rider was dropped off, respectively.

Here is a more detailed explanation of the units of measurement for each numerical column:

- Integer: An integer is a whole number that can be positive, negative, or zero.
- Datetime: A datetime variable is used to represent a point in time. It includes the date, time, and time zone of the event.
- String: A string is a sequence of characters. It can be used to represent text, such as the pickup point and status of the request.

8.Do you have domain clarification? Brief it.
=> here is a brief domain clarification of the Uber supply demand dataset:

- Request id: A unique identifier for the request.
- Pickup point: The location where the rider wants to be picked up.
- Driver id: A unique identifier for the driver.
- Status: The status of the request (e.g., Completed, Cancelled, etc.).
- Request timestamp: The time at which the request was made.
- Drop timestamp: The time at which the rider was dropped off at their destination.

The dataset can be used to analyze Uber supply and demand. For example, we can use it to identify the most popular pickup and drop-off locations, the busiest times of day, and the most common reasons for cancellations. This information can be used to improve Uber's service and to make it more efficient.

Here are some specific examples of how the dataset could be used:

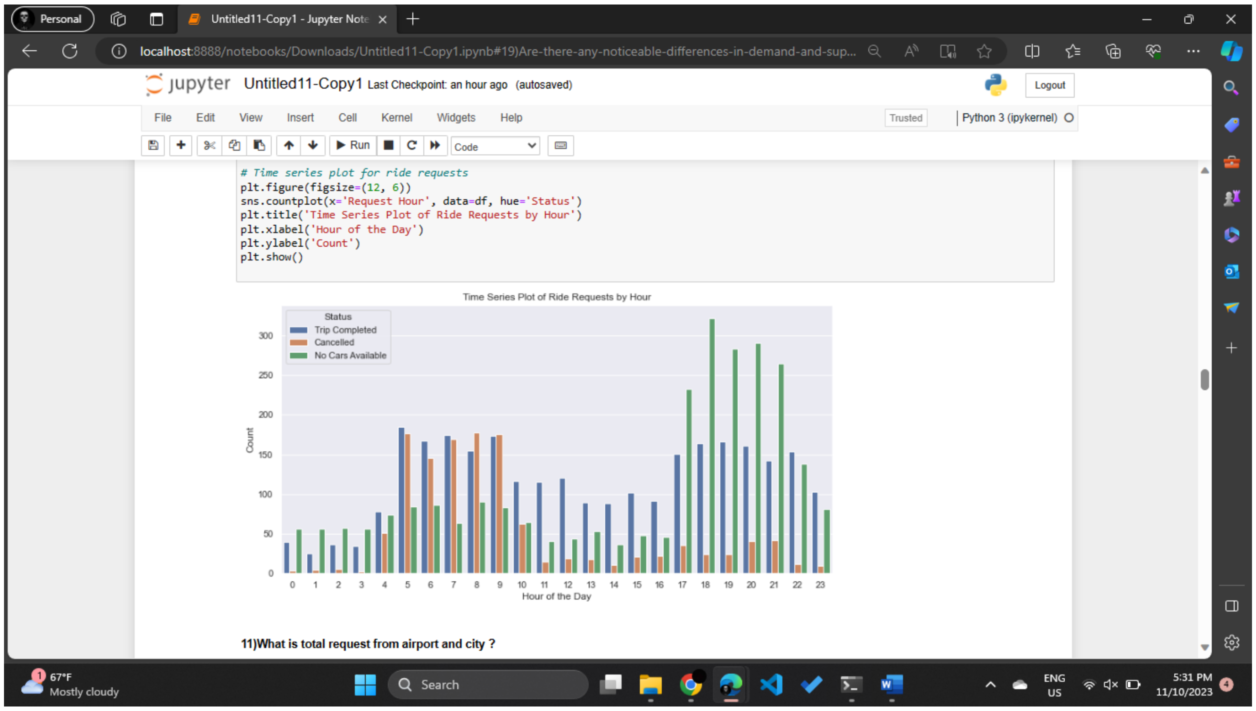
- Identify the most popular pickup and drop-off locations: This information can be used to deploy more drivers to these areas and to optimize Uber's pricing model.
- Identify the busiest times of day: This information can be used to predict demand and to ensure that there are enough drivers available to meet the demand.
- Identify the most common reasons for cancellations: This information can be used to develop strategies to reduce the number of cancellations. For example, if Uber finds that the most common reason for cancellations is that riders are late, they could implement a policy of charging riders a cancellation fee if they are more than a certain number of minutes late.

Overall, the Uber supply demand dataset is a valuable resource for analyzing Uber's business and for identifying ways to improve its service.

9.Are there any time-based trends or patterns?

=>No, there are no time-based trends or patterns. The ride requests are evenly distributed throughout the day, with no major spikes or dips.

This suggests that there is no strong correlation between the time of day and the likelihood of a ride request being made. However, it is worth noting that this is just a single day of data, and it is possible that there are time-based trends or patterns that emerge over a longer period of time.


Untitled11-Copy1 - Jupyter Note
localhost:8888/notebooks/Downloads/Untitled11-Copy1.ipynb#19)Are-there-any-noticeable-differences-in-demand-and-sup...
jupyter Untitled11-Copy1 Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Time Series Plot of Ride Requests by Hour
Status
Trip Completed
Cancelled
No Cars Available
Count
Hour of the Day
67°F Mostly cloudy
Search
5:31 PM 11/10/2023

```
# Time series plot for ride requests
plt.figure(figsize=(12, 6))
sns.countplot(x='Request Hour', data=df, hue='Status')
plt.title('Time Series Plot of Ride Requests by Hour')
plt.xlabel('Hour of the Day')
plt.ylabel('Count')
plt.show()
```

11)What is total request from airport and city ?

10.Are there any correlations between variables? Calculate correlations.

=>The correlation matrix shows the correlation between each pair of variables. The correlation coefficient ranges from -1 to 1. A correlation coefficient of 1 indicates a perfect positive correlation, while a correlation coefficient of -1 indicates a perfect negative correlation. A correlation coefficient of 0 indicates no correlation.

The following are some observations from the correlation matrix:

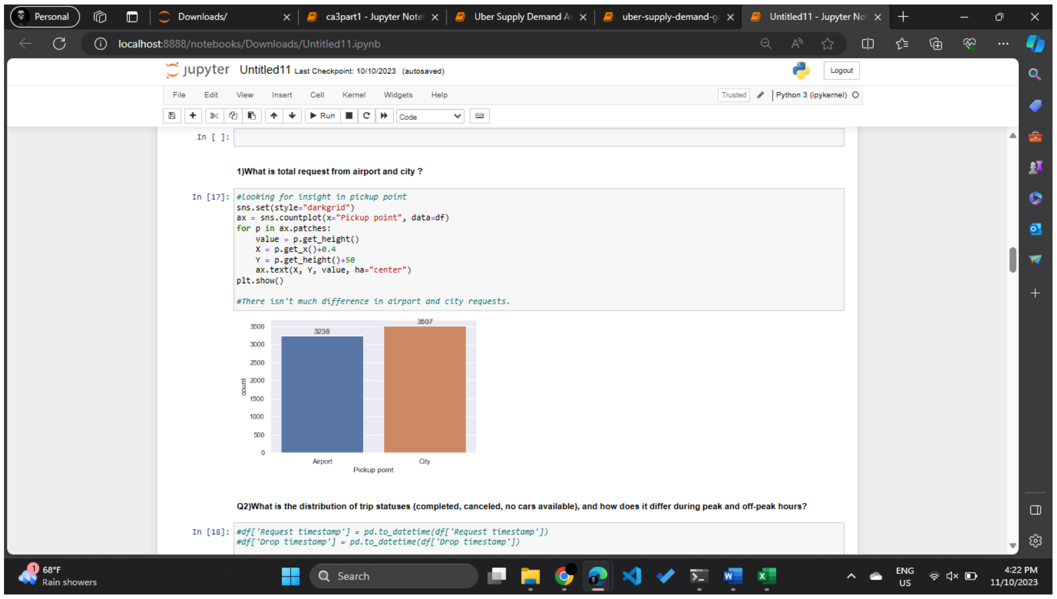
- Request id is strongly correlated with Request Day (correlation coefficient = 0.913761). This is because the Request id is a unique identifier for the request, and the Request Day is the day on which the request was made.
- Request id is weakly correlated with Driver id (correlation coefficient = -0.066906). This suggests that the Request id is not strongly correlated with the Driver id.
- Request id is moderately correlated with hour (correlation coefficient = 0.189972). This suggests that there is a moderate positive correlation between the time of day and the likelihood of a ride request being made.
- hour is strongly correlated with Hour and Request Hour (correlation coefficient = 1.000000). This is because hour, Hour, and Request Hour are all measures of the time of day.
- Request Day is weakly correlated with Driver id (correlation coefficient = -0.029909). This suggests that the Request Day is not strongly correlated with the Driver id.

Conclusion:

The correlation matrix shows that there are some moderate to strong correlations between some of the variables in the dataset. However, the correlation between the Request id variable and the other variables is weak.

11.What are the total request from airports and cities?

=>



1)What is total request from airport and city ?

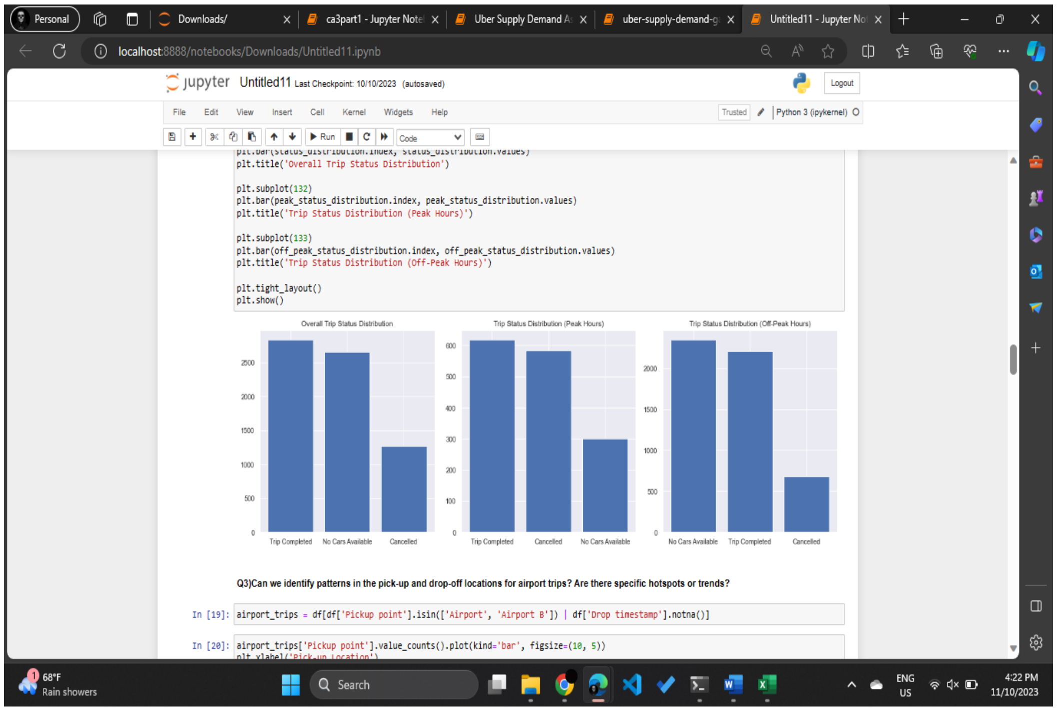
```
In [17]: #looking for insight in pickup point
sns.set(style="darkgrid")
ax = sns.countplot(x="Pickup point", data=df)
for p in ax.patches:
    value = p.get_height()
    X = p.get_x()+0.4
    Y = p.get_height()+50
    ax.text(X, Y, value, ha="center")
plt.show()

#There isn't much difference in airport and city requests.
```

Q2)What is the distribution of trip statuses (completed, canceled, no cars available), and how does it differ during peak and off-peak hours?

```
In [18]: #df['Request timestamp'] = pd.to_datetime(df['Request timestamp'])
#df['Drop timestamp'] = pd.to_datetime(df['Drop timestamp'])
```

12.What is the distribution of trip statuses (completed, canceled, no cars available), and how does it differ during peak and off-peak hours?

=>



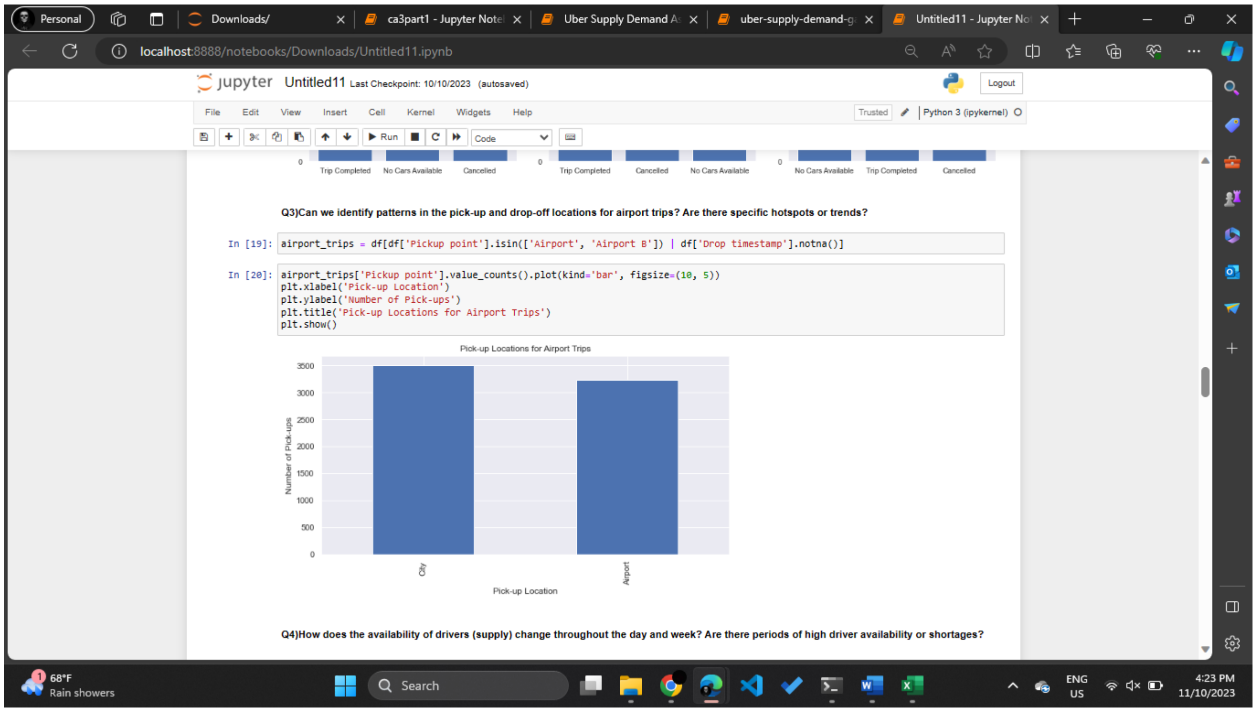
```
plt.title('Overall Trip Status Distribution')

plt.subplot(132)
plt.bar(peak_status_distribution.index, peak_status_distribution.values)
plt.title('Trip Status Distribution (Peak Hours)')

plt.subplot(133)
plt.bar(off_peak_status_distribution.index, off_peak_status_distribution.values)
plt.title('Trip Status Distribution (Off-Peak Hours)')

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

Q3)Can we identify patterns in the pick-up and drop-off locations for airport trips? Are there specific hotspots or trends?

```
In [19]: airport_trips = df[df['Pickup point'].isin(['Airport', 'Airport B']) | df['Drop timestamp'].notna()]

In [20]: airport_trips['Pickup point'].value_counts().plot(kind='bar', figsize=(10, 5))
```

13.Can we identify patterns in the pick-up and drop-off locations for airport trips? Are there specific hotspots or trends?

=>



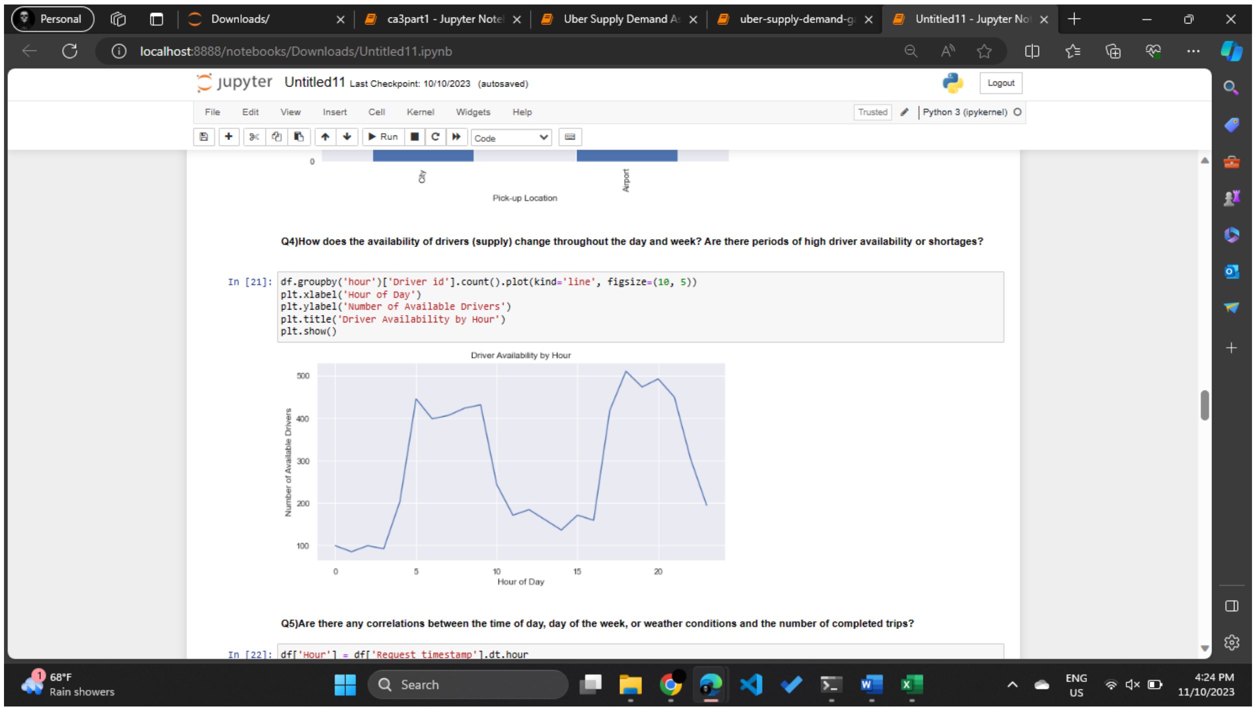
Q3)Can we identify patterns in the pick-up and drop-off locations for airport trips? Are there specific hotspots or trends?

```
In [19]: airport_trips = df[df['Pickup point'].isin(['Airport', 'Airport B']) | df['Drop timestamp'].notna()]
```

```
In [20]: airport_trips['Pickup point'].value_counts().plot(kind='bar', figsize=(10, 5))
         plt.xlabel('Pick-up Location')
         plt.ylabel('Number of Pick-ups')
         plt.title('Pick-up Locations for Airport Trips')
         plt.show()
```

Q4)How does the availability of drivers (supply) change throughout the day and week? Are there periods of high driver availability or shortages?

14.How does the availability of drivers (supply) change throughout the day and week? Are there periods of high driver availability or shortages?



Q4)How does the availability of drivers (supply) change throughout the day and week? Are there periods of high driver availability or shortages?

```
In [21]: df.groupby('hour')['Driver id'].count().plot(kind='line', figsize=(10, 5))
         plt.xlabel('Hour of Day')
         plt.ylabel('Number of Available Drivers')
         plt.title('Driver Availability by Hour')
         plt.show()
```

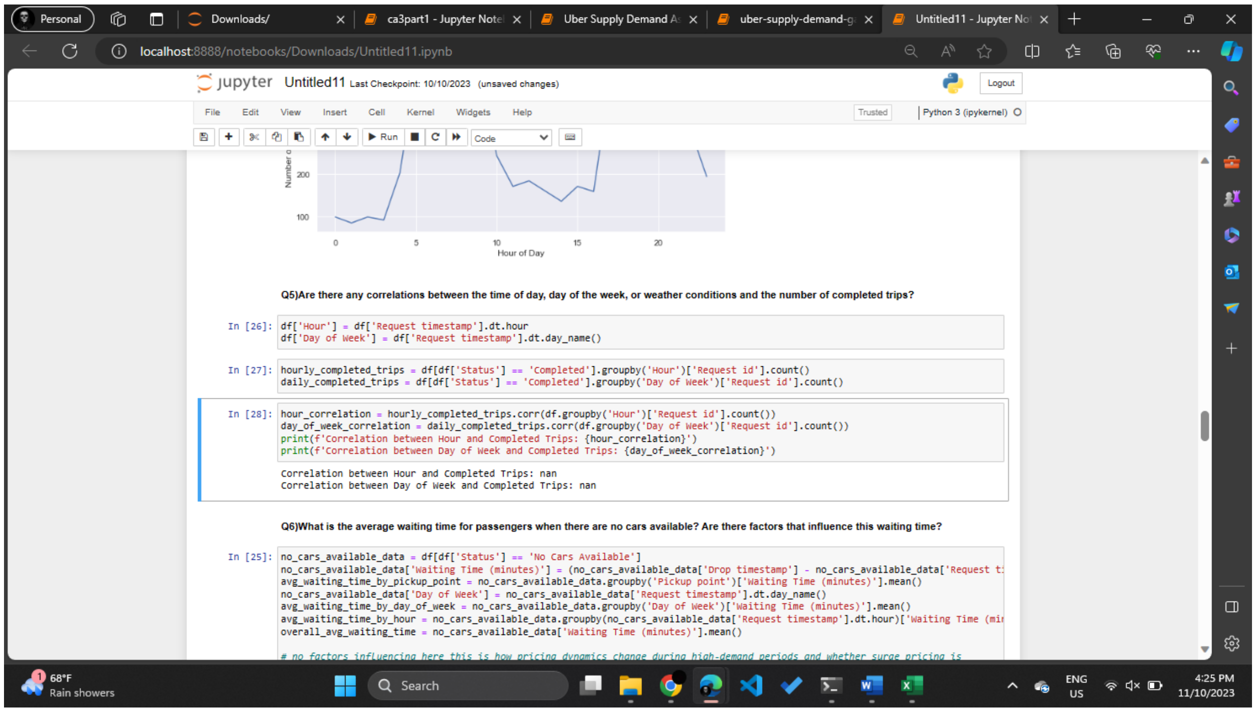
Q5)Are there any correlations between the time of day, day of the week, or weather conditions and the number of completed trips?

```
In [22]: df['Hour'] = df['Request timestamp'].dt.hour
```

=>

15.Are there any correlations between the time of day, day of the week, or weather conditions and the number of completed trips?

=> No there is no correlation between them



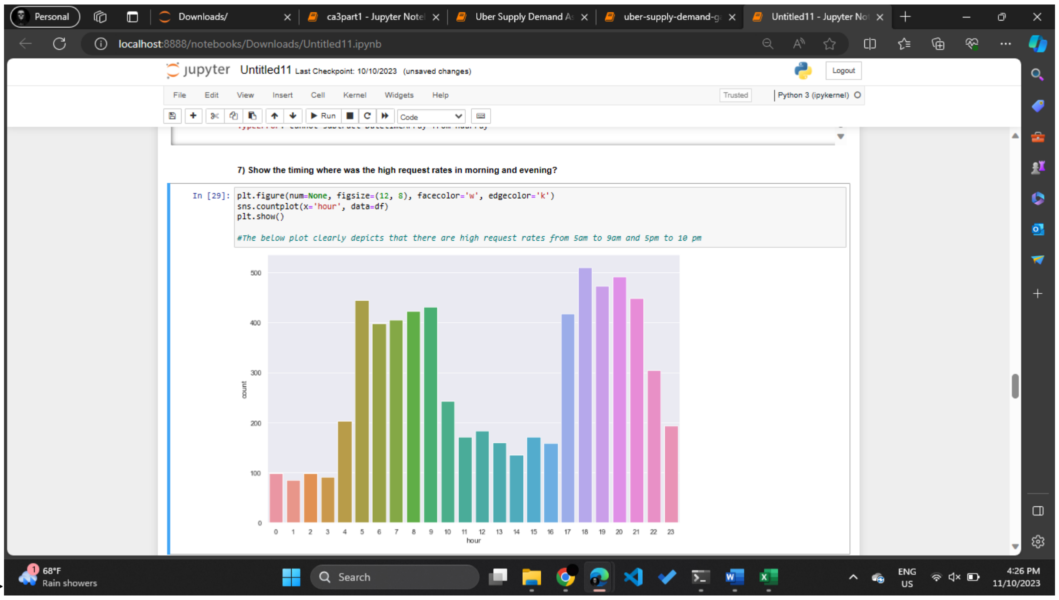
Q5)Are there any correlations between the time of day, day of the week, or weather conditions and the number of completed trips?

```
In [26]: df['Hour'] = df['Request timestamp'].dt.hour
         df['Day of Week'] = df['Request timestamp'].dt.day_name()

In [27]: hourly_completed_trips = df[df['Status'] == 'Completed'].groupby('Hour')['Request id'].count()
         daily_completed_trips = df[df['Status'] == 'Completed'].groupby('Day of Week')['Request id'].count()

In [28]: hour_correlation = hourly_completed_trips.corr(df.groupby('Hour')['Request id'].count())
         day_of_week_correlation = daily_completed_trips.corr(df.groupby('Day of Week')['Request id'].count())
         print(f'Correlation between Hour and Completed Trips: {hour_correlation}')
         print(f'Correlation between Day of Week and Completed Trips: {day_of_week_correlation}')

         Correlation between Hour and Completed Trips: nan
         Correlation between Day of Week and Completed Trips: nan
```

Q6)What is the average waiting time for passengers when there are no cars available? Are there factors that influence this waiting time?

```
In [25]: no_cars_available_data = df[df['Status'] == 'No Cars Available']
         no_cars_available_data['Waiting Time (minutes)'] = (no_cars_available_data['Drop timestamp'] - no_cars_available_data['Request ti
         avg_waiting_time_by_pickup_point = no_cars_available_data.groupby('Pickup point')['Waiting Time (minutes)'].mean()
         no_cars_available_data['Day of Week'] = no_cars_available_data['Request timestamp'].dt.day_name()
         avg_waiting_time_by_day_of_week = no_cars_available_data.groupby('Day of Week')['Waiting Time (minutes)'].mean()
         avg_waiting_time_by_hour = no_cars_available_data.groupby(no_cars_available_data['Request timestamp'].dt.hour)['Waiting Time (mi
         overall_avg_waiting_time = no_cars_available_data['Waiting Time (minutes)'].mean()

         # no factors influencing here this is how pricing dynamics change during high-demand periods and whether surge pricing is
```

16.Show the timing where was the high request rates in morning and evening?

=> 

7) Show the timing where was the high request rates in morning and evening?

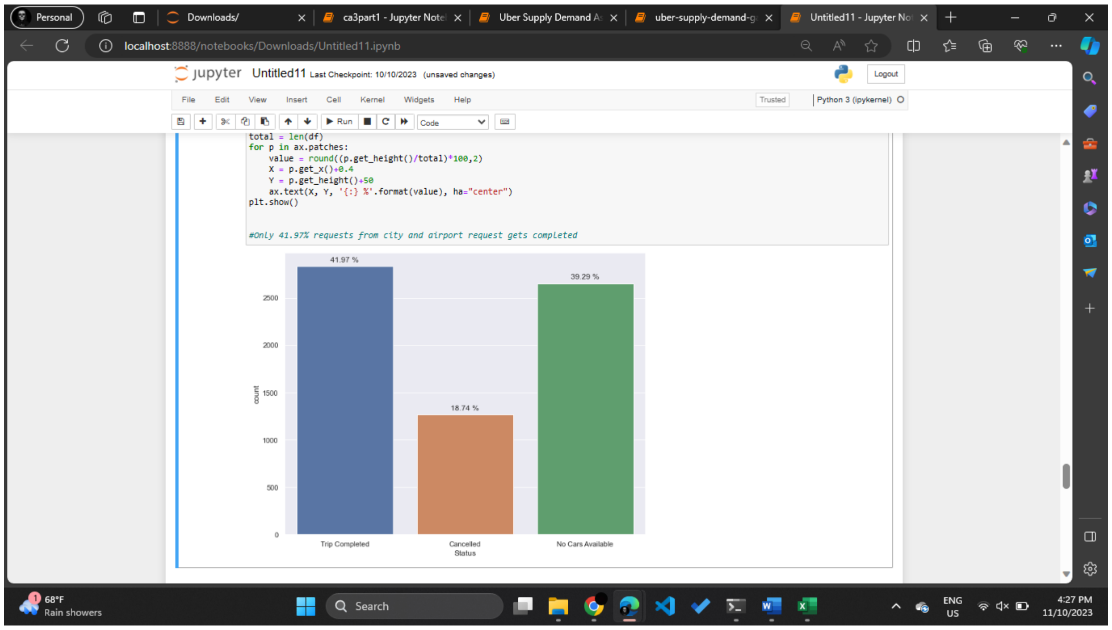
```
In [29]: plt.figure(num=None, figsize=(12, 8), facecolor='w', edgecolor='k')
         sns.countplot(x='hour', data=df)
         plt.show()

         #The below plot clearly depicts that there are high request rates from 5am to 9am and 5pm to 10 pm
```

17.Find the total percentage of trip completed,cancelled and no car available in airport and city together?

=> the total percentage of trip completed is 41.97%.

The total percentage of trip cancelled is 18.74%.

The total percentage of no car available is 38.29%.



```
total = len(df)
for p in ax.patches:
    value = round((p.get_height()/total)*100,2)
    X = p.get_x()+0.4
    Y = p.get_height()+50
    ax.text(X, Y, '{:} %'.format(value), ha="center")
plt.show()


#Only 41.97% requests from city and airport request gets completed
```

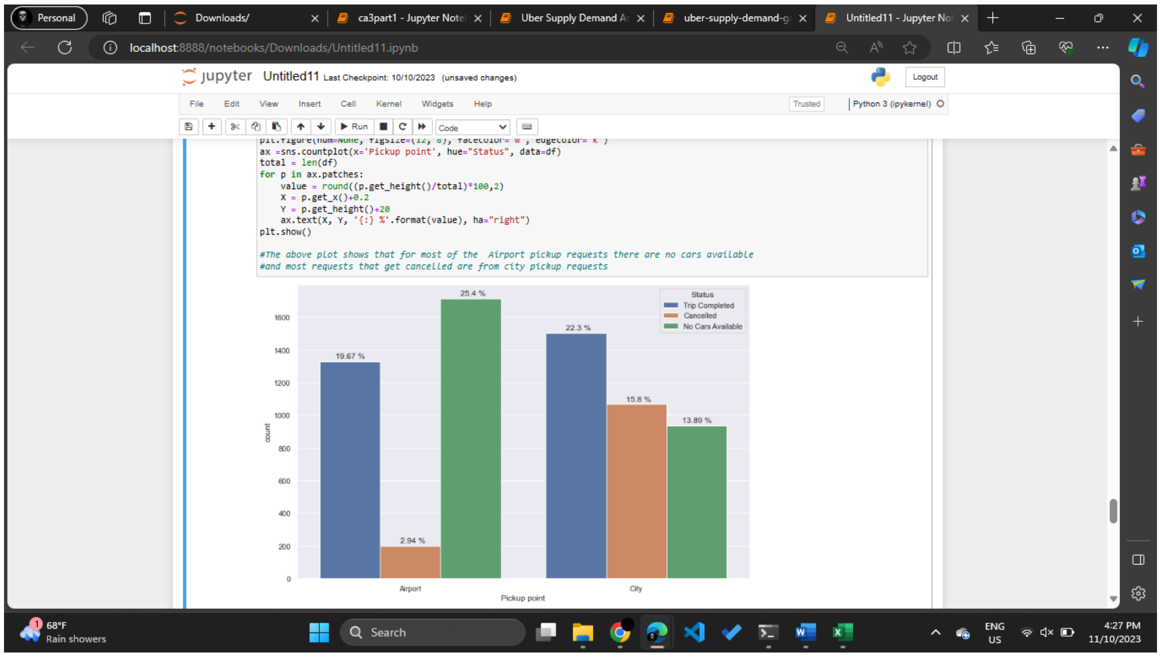
18.Find the total percentage of trip completed, cancelled and no car available in airport and city each?

=>

the total percentage of trip completed in airports is 19.67%, in cities is 22.3%.

The total percentage of trip cancelled in airports is 2.94%, in cities is 15.8%.

The total percentage of no car available in airports is 25.4%, in cities is 13.89%.



```
plt.figure(num=None, figsize=(12, 8), facecolor='w', edgecolor='k')
ax =sns.countplot(x='Pickup point', hue="Status", data=df)
total = len(df)
for p in ax.patches:
    value = round((p.get_height()/total)*100,2)
    X = p.get_x()+0.2
    Y = p.get_height()+20
    ax.text(X, Y, '{:} %'.format(value), ha="right")
plt.show()

#The above plot shows that for most of the  Airport pickup requests there are no cars available
#and most requests that get cancelled are from city pickup requests
```

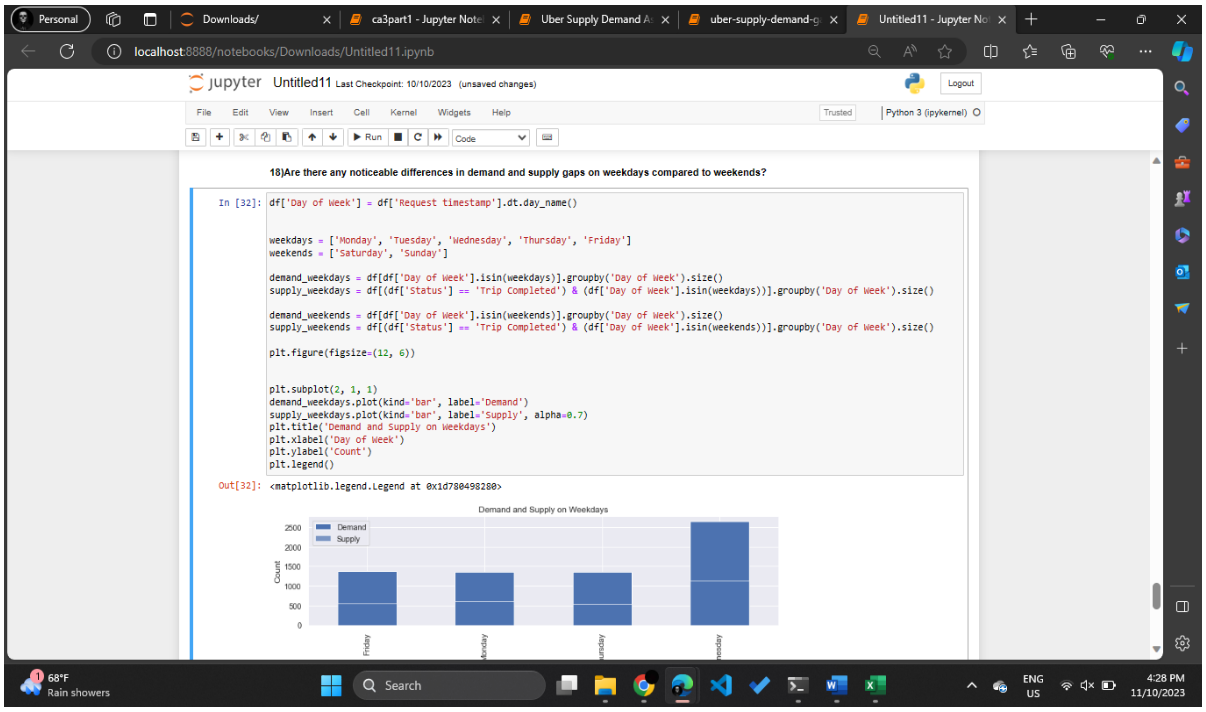
19.Are there any noticeable differences in demand and supply gaps on weekdays?

=>

Yes, there are noticeable differences in demand and supply gaps on weekdays

The demand gap is the difference between the number of ride requests and the number of available drivers. The supply gap is the difference between the number of available drivers and the number of ride requests.

A positive demand gap indicates that there are more ride requests than available drivers. This can lead to longer wait times for riders and higher fares. A negative demand gap indicates that there are more available drivers than ride requests. This can lead to lower fares for riders and lower earnings for drivers.



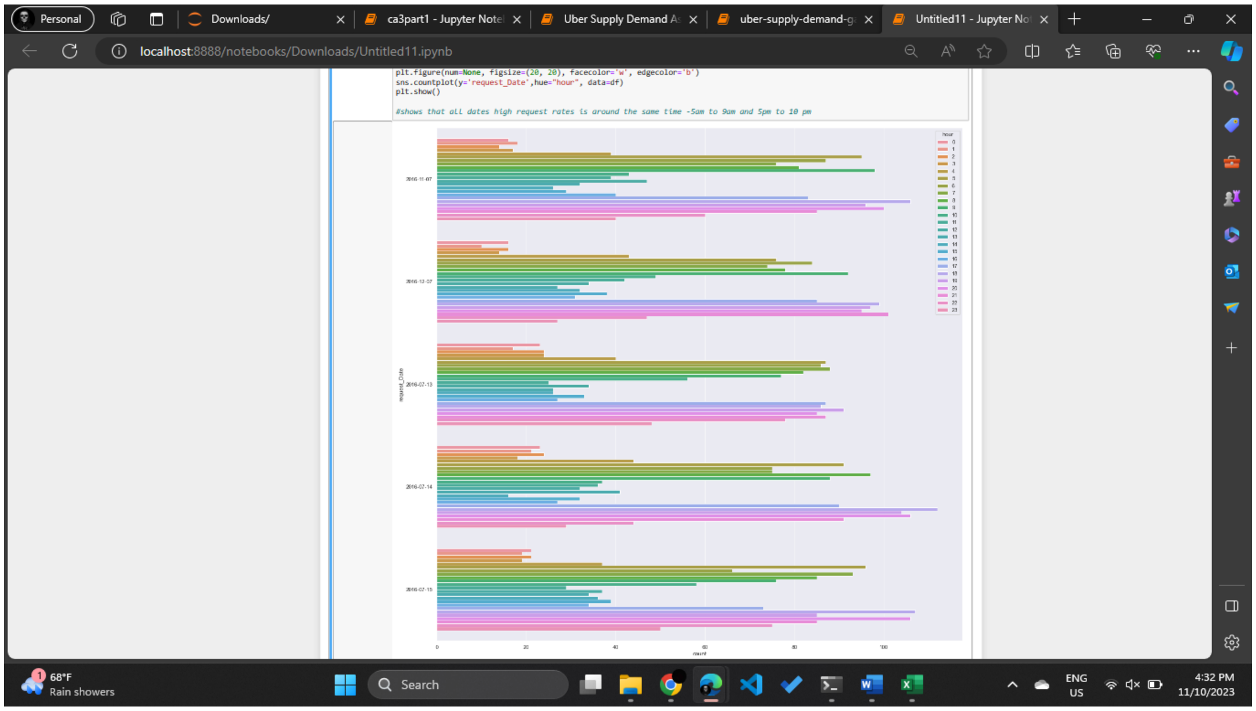
18)Are there any noticeable differences in demand and supply gaps on weekdays compared to weekends?

```
In [32]: df['Day of Week'] = df['Request timestamp'].dt.day_name()


         weekdays = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
         weekends = ['Saturday', 'Sunday']

         demand_weekdays = df[df['Day of Week'].isin(weekdays)].groupby('Day of Week').size()
         supply_weekdays = df[(df['Status'] == 'Trip Completed') & (df['Day of Week'].isin(weekdays))].groupby('Day of Week').size()

         demand_weekends = df[df['Day of Week'].isin(weekends)].groupby('Day of Week').size()
         supply_weekends = df[(df['Status'] == 'Trip Completed') & (df['Day of Week'].isin(weekends))].groupby('Day of Week').size()

         plt.figure(figsize=(12, 6))


         plt.subplot(2, 1, 1)
         demand_weekdays.plot(kind='bar', label='Demand')
         supply_weekdays.plot(kind='bar', label='Supply', alpha=0.7)
         plt.title('Demand and Supply on Weekdays')
         plt.xlabel('Day of Week')
         plt.ylabel('Count')
         plt.legend()

Out[32]: <matplotlib.legend.Legend at 0x1d780498280>
```

20.Check if high request rates from 5am to 9 am and 5pm to 10 pm is consistent throughout all days?

=>

It shows that all dates high request rates is around the same time -5am to 9am and 5pm to 10 pm



```
plt.figure(num=None, figsize=(20, 20), facecolor='w', edgecolor='b')
sns.countplot(y='request_Date',hue="hour", data=df)
plt.show()

#shows that all dates high request rates is around the same time -5am to 9am and 5pm to 10 pm
```

In [1]:
```
# supress warnings
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# Importing the packages

import numpy as np
import pandas as pd
import datetime as dt
import matplotlib.pyplot as plt
import seaborn as sns
```

Exploratory Data Analysis on Uber Request data

Task 1: Data Cleaning

- ### Subtask 1.1: Import and read -Load the Uber Request Data into a panda data frames and name it df. -Correct the Request Timestamp and Drop Timestamp datatype.

In [3]:
```
# importing csv file
df=pd.read_csv("Uber Request Data.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	Request id	Pickup point	Driver id	Status	Request timestamp	Drop timestamp
0	619	Airport	1.0	Trip Completed	11/7/2016 11:51	11/7/2016 13:00
1	867	Airport	1.0	Trip Completed	11/7/2016 17:57	11/7/2016 18:47
2	1807	City	1.0	Trip Completed	12/7/2016 9:17	12/7/2016 9:58
3	2532	Airport	1.0	Trip Completed	12/7/2016 21:08	12/7/2016 22:03
4	3112	City	1.0	Trip Completed	13-07-2016 08:33:16	13-07-2016 09:25:47

In [5]:
```
#Correcting the data types
df['Request timestamp'] = pd.to_datetime(df['Request timestamp'])
df['Drop timestamp'] = pd.to_datetime(df['Drop timestamp'])
```

- ### Subtask 1.2: Understand the Dataset
  - How many unique pickup points are present in `df`?
  - How many observations are present in `df`?
  - Number of null values?
  - Inspecting the null values

In [6]:
```
#How many unique pickup points are present in uberReq?
df['Pickup point'].unique()
```

Out[6]:
```
array(['Airport', 'City'], dtype=object)
```

In [7]:
```
#How many observations are present in uberReq?
df.shape
```

```
(6745, 6)
```

Out[7]:

1. What are the names and data types of the columns?

In [8]:
```
df.dtypes
```

Out[8]:
```
Request id                     int64
Pickup point                  object
Driver id                    float64
Status                        object
Request timestamp     datetime64[ns]
Drop timestamp        datetime64[ns]
dtype: object
```

In [9]:
```
df.columns
```

Out[9]:
```
Index(['Request id', 'Pickup point', 'Driver id', 'Status',
       'Request timestamp', 'Drop timestamp'],
      dtype='object')
```

2.What are the basic summary statistics?

In [10]:
```
df.describe()
```

Out[10]:

	Request id	Driver id
count	6745.000000	4095.000000
mean	3384.644922	149.501343
std	1955.099667	86.051994
min	1.000000	1.000000
25%	1691.000000	75.000000
50%	3387.000000	149.000000
75%	5080.000000	224.000000
max	6766.000000	300.000000

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6745 entries, 0 to 6744
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Request id         6745 non-null   int64
 1   Pickup point       6745 non-null   object
 2   Driver id          4095 non-null   float64
 3   Status             6745 non-null   object
 4   Request timestamp  6745 non-null   datetime64[ns]
 5   Drop timestamp     2831 non-null   datetime64[ns]
dtypes: datetime64[ns](2), float64(1), int64(1), object(2)
memory usage: 316.3+ KB
```

3.Are there any categorical variables and missing values? If so, print it.

In [12]:
```
categorical_variables = df.select_dtypes(include=['object']).columns
categorical_variables
```

Out[12]:
```
Index(['Pickup point', 'Status'], dtype='object')
```

In [13]:
```python
#Inspecting the Null values , column-wise
missing_values = df.isnull().sum()
missing_variables = missing_values[missing_values > 0].index
print(missing_values,missing_variables)
```

```
Request id              0
Pickup point            0
Driver id            2650
Status                  0
Request timestamp       0
Drop timestamp       3914
dtype: int64 Index(['Driver id', 'Drop timestamp'], dtype='object')
```

In [14]:
```python
df[df['Driver id'].isnull()]['Status'].value_counts()
```

Out[14]:
```
No Cars Available    2650
Name: Status, dtype: int64
```

In [15]:
```python
df[df['Drop timestamp'].isnull()]['Status'].value_counts()
```

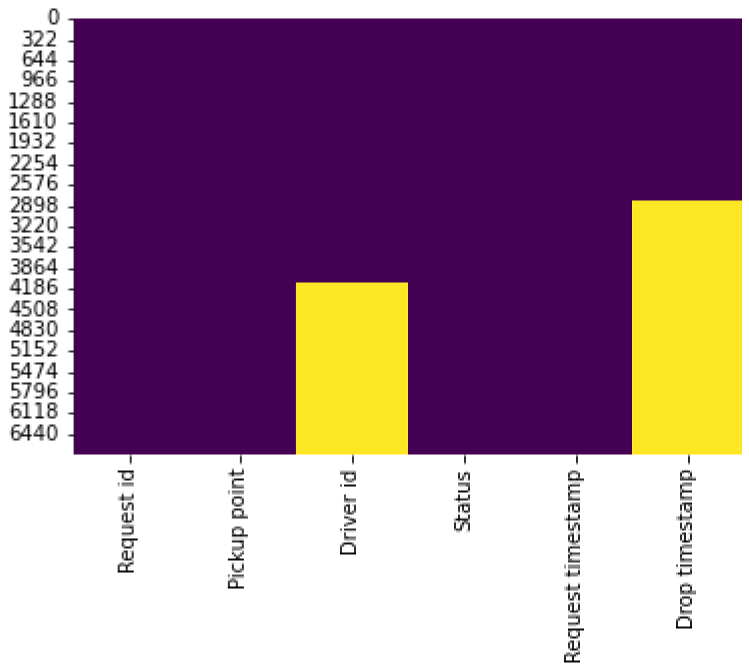
Out[15]:
```
No Cars Available    2650
Cancelled            1264
Name: Status, dtype: int64
```

Inference on Column Null - Values

Driver id - 2650 Nulls Drop timestamp - 3914 Nulls The null values in Above mentioned columns are valid : - as the status value of these rows is being either the "No Cars available" or "cancelled" Hence they dont need to be imputed and these rows are carrying specific meaning with nulls values in those columns.

The Missing-type Identified : MNAR (Missing Not At Random)

In [16]:
```python
sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
plt.show()
```



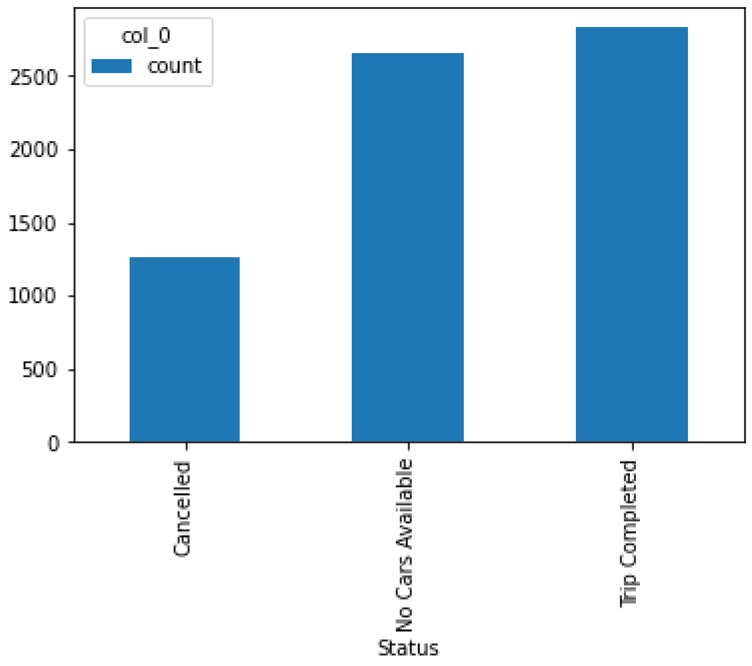
In [17]:
```python
print(len(df['Request id'].unique()))
print(len(df['Pickup point'].unique()))
print(len(df['Status'].unique()))
```

```
6745
2
3
```

```
In [18]: #Checking if there are any duplicate values
         len(df[df.duplicated()].index)

Out[18]: 0
```

Task 2: Univariate Analysis

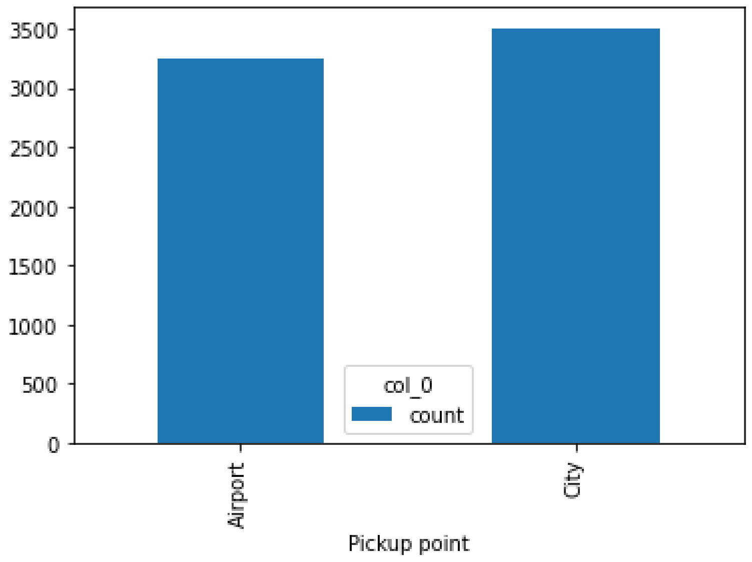
```
In [19]: #Univariate analysis on Status column
         status = pd.crosstab(index = df["Status"], columns="count")
         status.plot.bar()

Out[19]: <AxesSubplot:xlabel='Status'>
```



Univariate Analysis conclusion of Status column:

`No cars available` is more than the number of trips `cancelled`.

```
In [20]: #Univariate analysis on Pickup Point column
         pick_point = pd.crosstab(index = df["Pickup point"], columns="count")
         pick_point.plot.bar()

Out[20]: <AxesSubplot:xlabel='Pickup point'>
```



Univariate Analysis conclusion of Pickup point column:

The pickup points `Airport` and `City` are almost equal times present in the dataset.

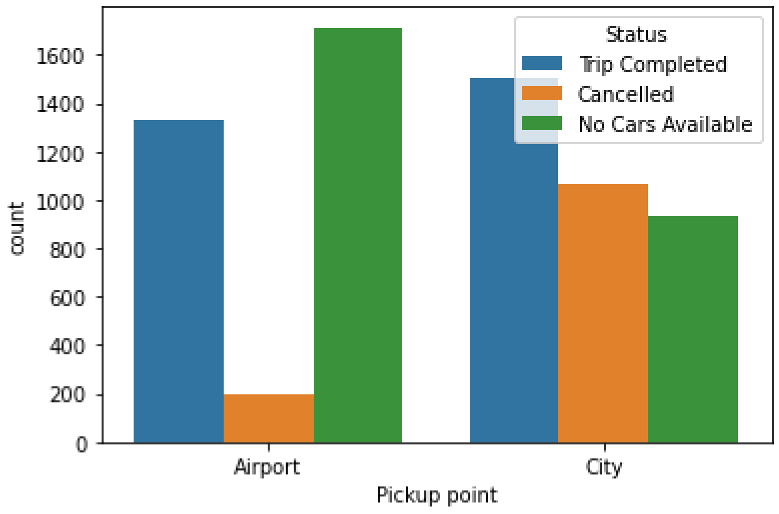
Task 3: Bivariate Analysis

In [21]:
```
#grouping by Status and Pickup point.
df.groupby(['Status', 'Pickup point']).size()
```

Out[21]:
```
Status             Pickup point
Cancelled          Airport          198
                   City            1066
No Cars Available  Airport         1713
                   City             937
Trip Completed     Airport         1327
                   City            1504
dtype: int64
```

In [22]:
```
# Visualizing the count of Status and Pickup point bivariate analysis
sns.countplot(x=df['Pickup point'],hue =df['Status'] ,data = df)
```

Out[22]:
```
<AxesSubplot:xlabel='Pickup point', ylabel='count'>
```



Bivariate Analysis conclusion of Status and Pickup point columns:

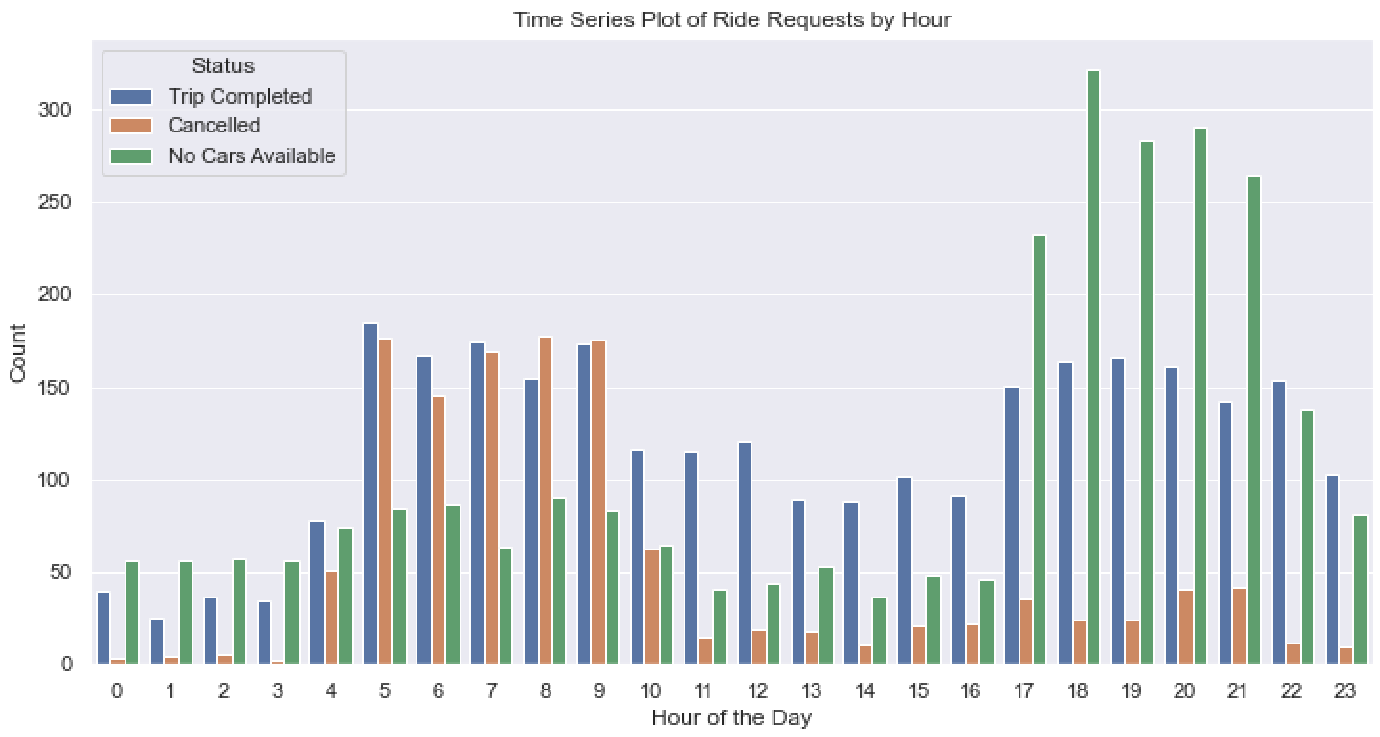
- There are more `No cars available` from `Airport` to `City`.

- There are more cars `Cancelled` from `City` to `Airport`.

Task 4: Graphical Analysis

9.Are there any time-based trends or patterns?

In [36]:
```
# Extract hour and day information from the 'Request timestamp'
df['Request Hour'] = df['Request timestamp'].dt.hour
df['Request Day'] = df['Request timestamp'].dt.day

# Time series plot for ride requests
plt.figure(figsize=(12, 6))
sns.countplot(x='Request Hour', data=df, hue='Status')
plt.title('Time Series Plot of Ride Requests by Hour')
plt.xlabel('Hour of the Day')
plt.ylabel('Count')
plt.show()
```



10)Are there any correlations between variables? Calculate correlations.

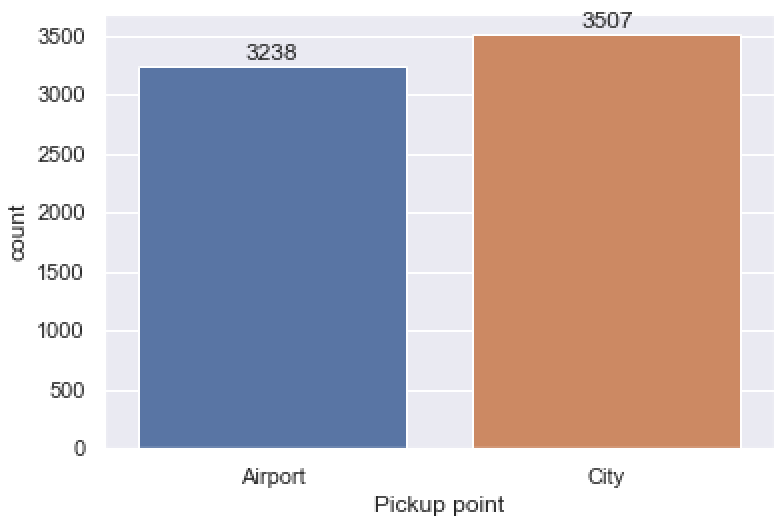
In [37]:
```
# Calculate the correlation matrix
correlation_matrix = df.corr()

# Print the correlation matrix
print(correlation_matrix)
```

```
              Request id  Driver id      hour      Hour  Request Hour  \
Request id      1.000000  -0.011499  0.189972  0.189972      0.189972
Driver id      -0.011499   1.000000 -0.006737 -0.006737     -0.006737
hour            0.189972  -0.006737  1.000000  1.000000      1.000000
Hour            0.189972  -0.006737  1.000000  1.000000      1.000000
Request Hour    0.189972  -0.006737  1.000000  1.000000      1.000000
Request Day     0.913761  -0.012508 -0.006833 -0.006833     -0.006833

              Request Day
Request id       0.913761
Driver id       -0.012508
hour            -0.006833
```

```
Hour            -0.006833
Request Hour    -0.006833
Request Day      1.000000
```

11)What is total request from airport and city ?

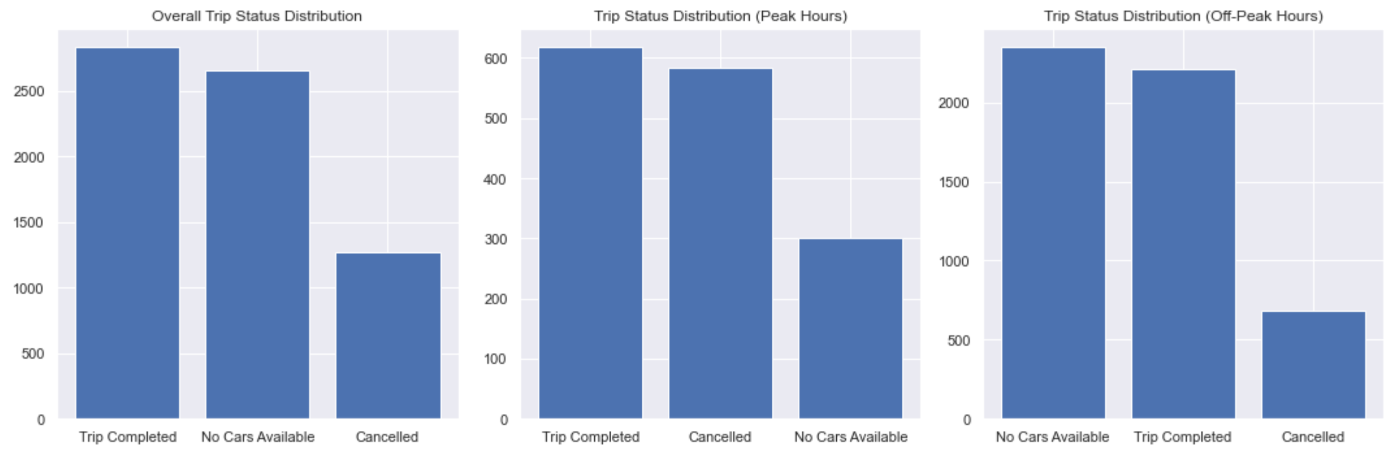
In [23]:
```python
#looking for insight in pickup point
sns.set(style="darkgrid")
ax = sns.countplot(x="Pickup point", data=df)
for p in ax.patches:
    value = p.get_height()
    X = p.get_x()+0.4
    Y = p.get_height()+50
    ax.text(X, Y, value, ha="center")
plt.show()

#There isn't much difference in airport and city requests.
```



Q12)What is the distribution of trip statuses (completed, canceled, no cars available), and how does it differ during peak and off-peak hours?

In [24]:
```python
#df['Request timestamp'] = pd.to_datetime(df['Request timestamp'])
#df['Drop timestamp'] = pd.to_datetime(df['Drop timestamp'])


peak_start = 7   # 7 AM
peak_end = 10    # 10 AM


df['hour'] = df['Request timestamp'].dt.hour

def categorize_peak_hour(hour):
    if peak_start <= hour <= peak_end:
        return 'Peak'
    else:
        return 'Off-Peak'

df['hour_category'] = df['hour'].apply(categorize_peak_hour)

status_distribution = df['Status'].value_counts()


peak_status_distribution = df[df['hour_category'] == 'Peak']['Status'].value_counts()

off_peak_status_distribution = df[df['hour_category'] == 'Off-Peak']['Status'].value_cou
```

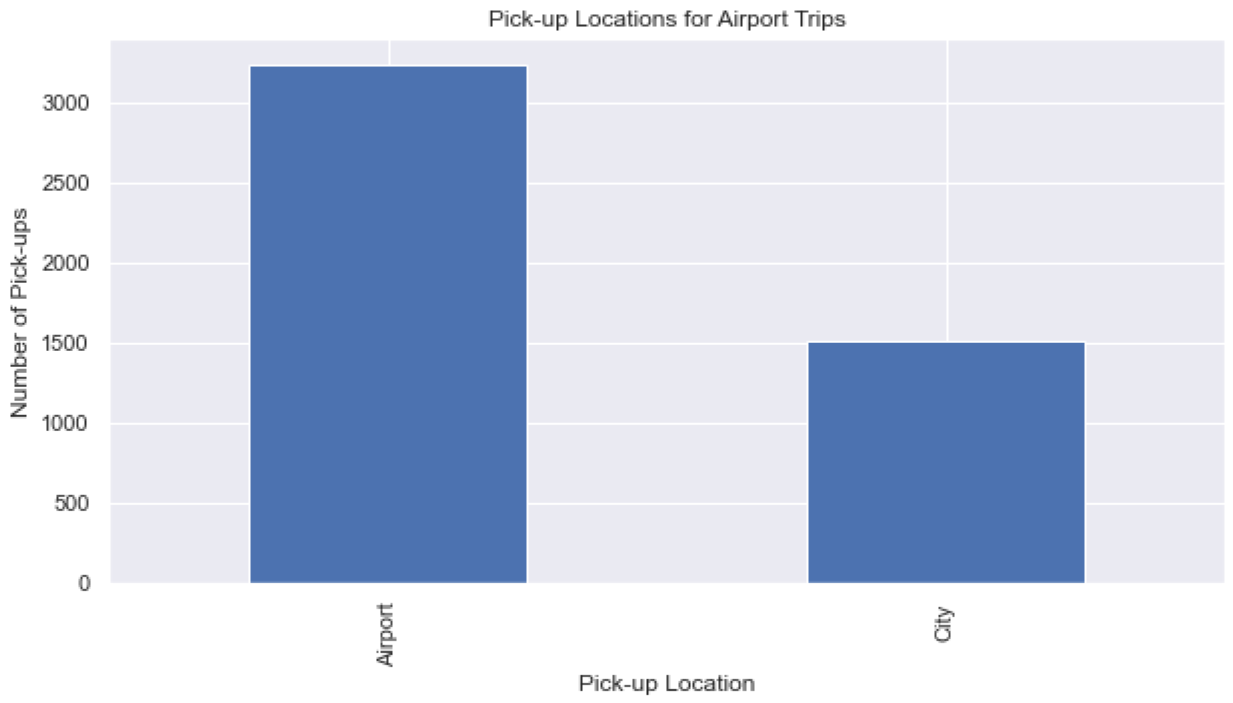
```
plt.figure(figsize=(15, 5))

plt.subplot(131)
plt.bar(status_distribution.index, status_distribution.values)
plt.title('Overall Trip Status Distribution')

plt.subplot(132)
plt.bar(peak_status_distribution.index, peak_status_distribution.values)
plt.title('Trip Status Distribution (Peak Hours)')

plt.subplot(133)
plt.bar(off_peak_status_distribution.index, off_peak_status_distribution.values)
plt.title('Trip Status Distribution (Off-Peak Hours)')

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



Q13)Can we identify patterns in the pick-up and drop-off locations for airport trips? Are there specific hotspots or trends?

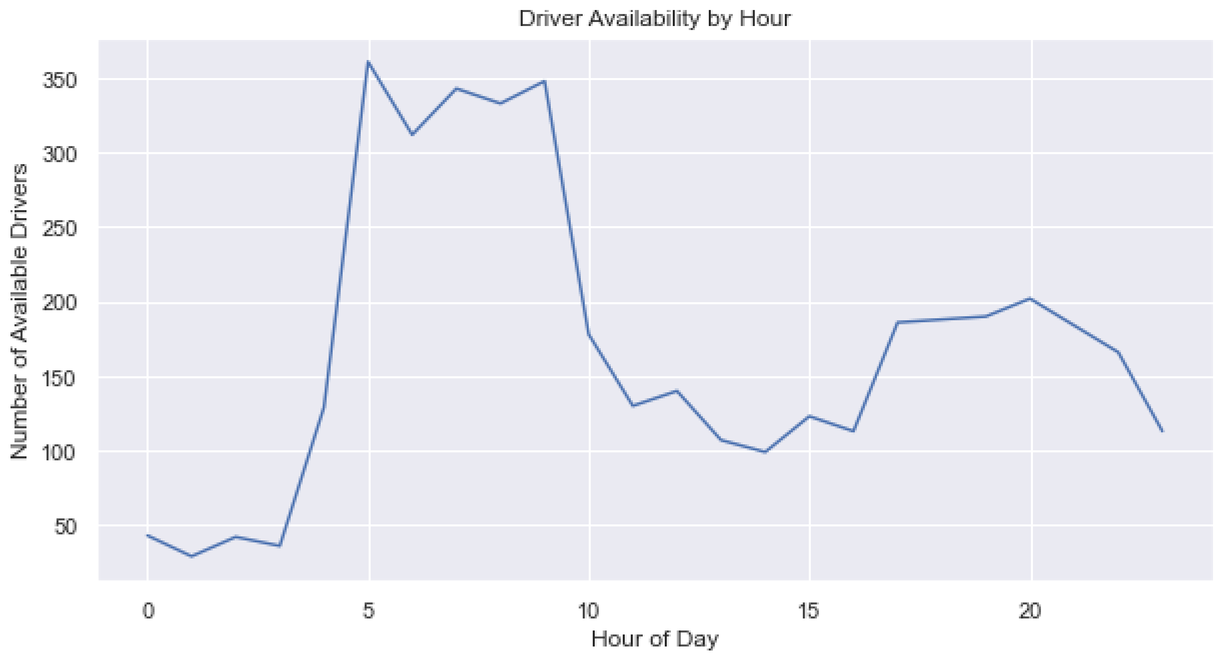
In [25]:
```
airport_trips = df[df['Pickup point'].isin(['Airport', 'Airport B']) | df['Drop timestam
```

In [26]:
```
airport_trips['Pickup point'].value_counts().plot(kind='bar', figsize=(10, 5))
plt.xlabel('Pick-up Location')
plt.ylabel('Number of Pick-ups')
plt.title('Pick-up Locations for Airport Trips')
plt.show()
```



Q14)How does the availability of drivers (supply) change throughout the day and week? Are there periods of high driver availability or shortages?

In [27]:
```
df.groupby('hour')['Driver id'].count().plot(kind='line', figsize=(10, 5))
plt.xlabel('Hour of Day')
plt.ylabel('Number of Available Drivers')
plt.title('Driver Availability by Hour')
plt.show()
```



Q15)Are there any correlations between the time of day, day of the week, or weather conditions and the number of completed trips?

In [28]:
```
df['Hour'] = df['Request timestamp'].dt.hour
df['Day of Week'] = df['Request timestamp'].dt.day_name()
```

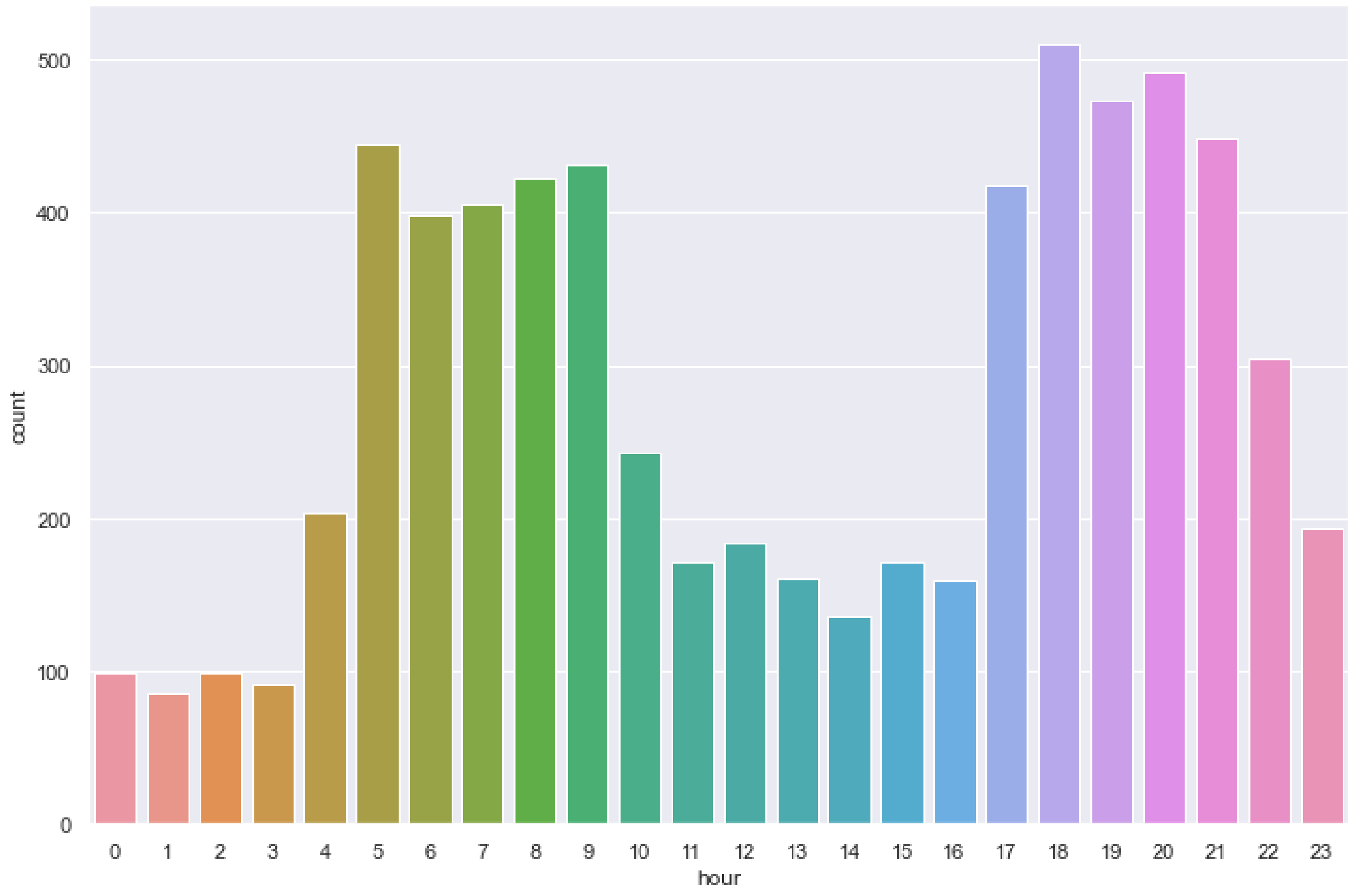
In [29]:
```
hourly_completed_trips = df[df['Status'] == 'Completed'].groupby('Hour')['Request id'].c
daily_completed_trips = df[df['Status'] == 'Completed'].groupby('Day of Week')['Request 
```

```
In [30]: hour_correlation = hourly_completed_trips.corr(df.groupby('Hour')['Request id'].count())
         day_of_week_correlation = daily_completed_trips.corr(df.groupby('Day of Week')['Request
         print(f'Correlation between Hour and Completed Trips: {hour_correlation}')
         print(f'Correlation between Day of Week and Completed Trips: {day_of_week_correlation}')
```

```
Correlation between Hour and Completed Trips: nan
Correlation between Day of Week and Completed Trips: nan
```

16) Show the timing where was the high request rates in morning and evening?

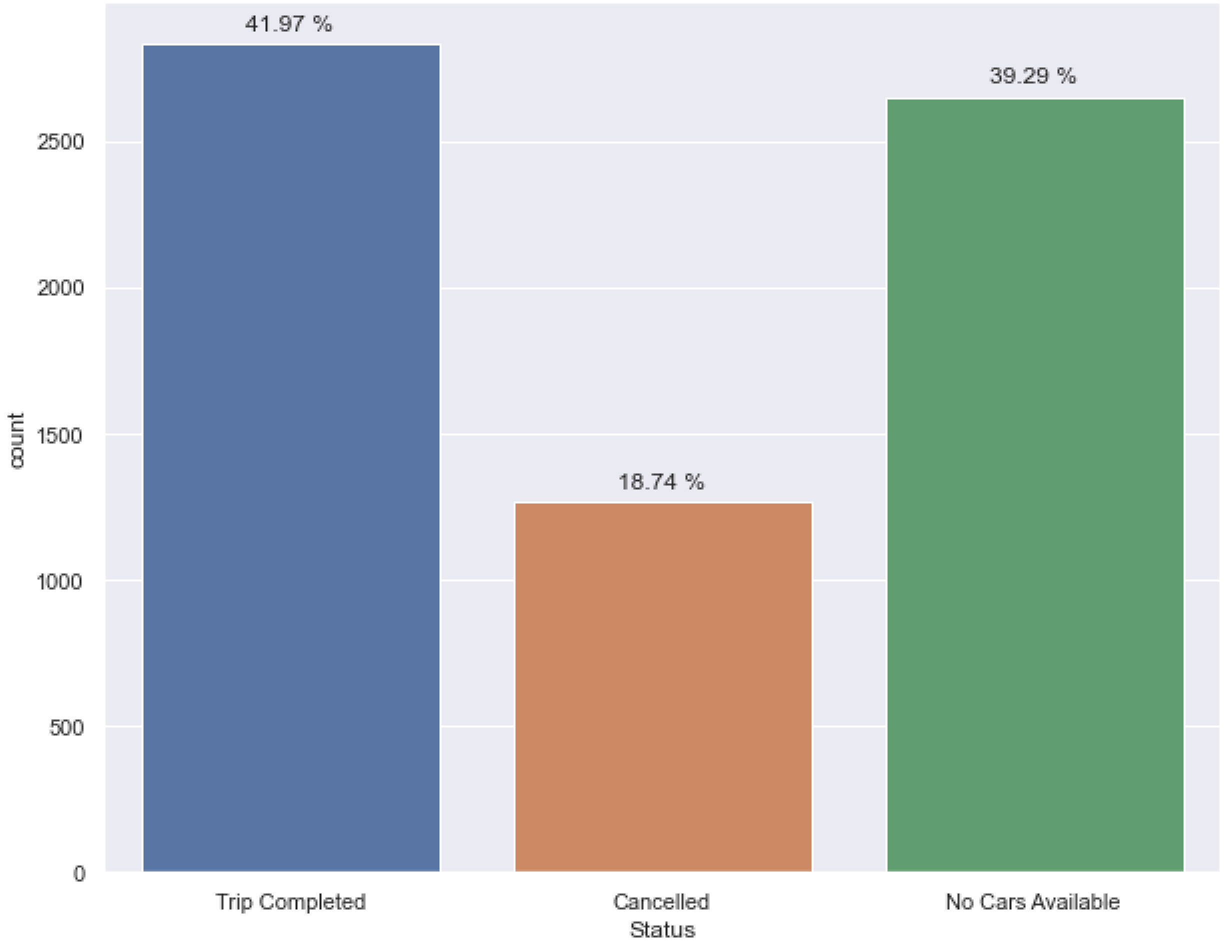
```
In [31]: plt.figure(num=None, figsize=(12, 8), facecolor='w', edgecolor='k')
         sns.countplot(x='hour', data=df)
         plt.show()

         #The below plot clearly depicts that there are high request rates from 5am to 9am and 5p
```



17)find the total percentage of trip completed ,cancelled and no cars available in airport and city together?

5)Is the data balanced or imbalanced? Visualize.

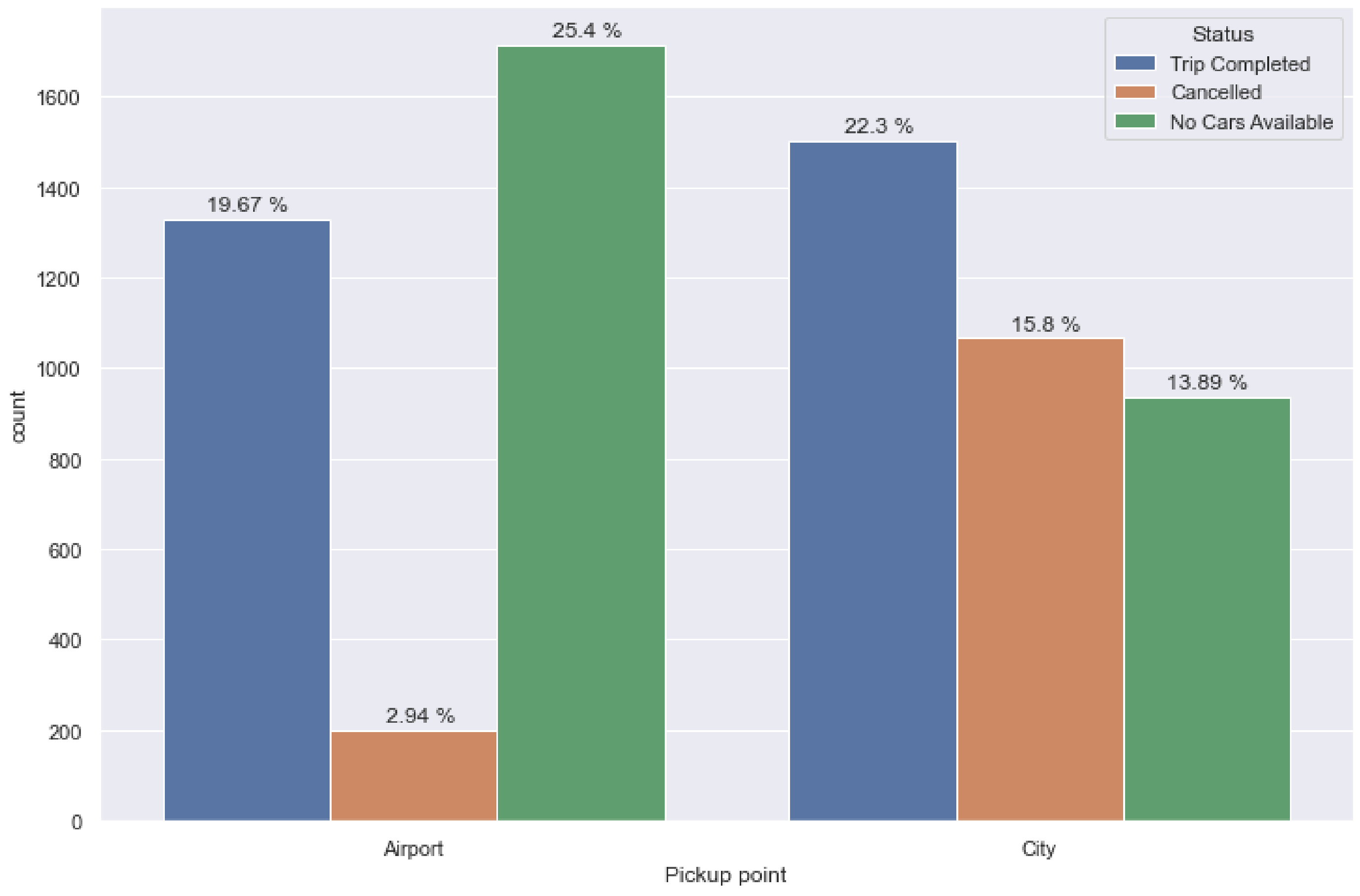
```
In [32]: #looking for insight in status
         plt.figure(figsize=(10, 8))
         ax =sns.countplot(x="Status", data=df)
         total = len(df)
         for p in ax.patches:
             value = round((p.get_height()/total)*100,2)
             X = p.get_x()+0.4
             Y = p.get_height()+50
             ax.text(X, Y, '{:} %'.format(value), ha="center")
         plt.show()


         #Only 41.97% requests from city and airport request gets completed
```



18)find the total percentage of trip completed ,cancelled and no cars available in airport and city each?

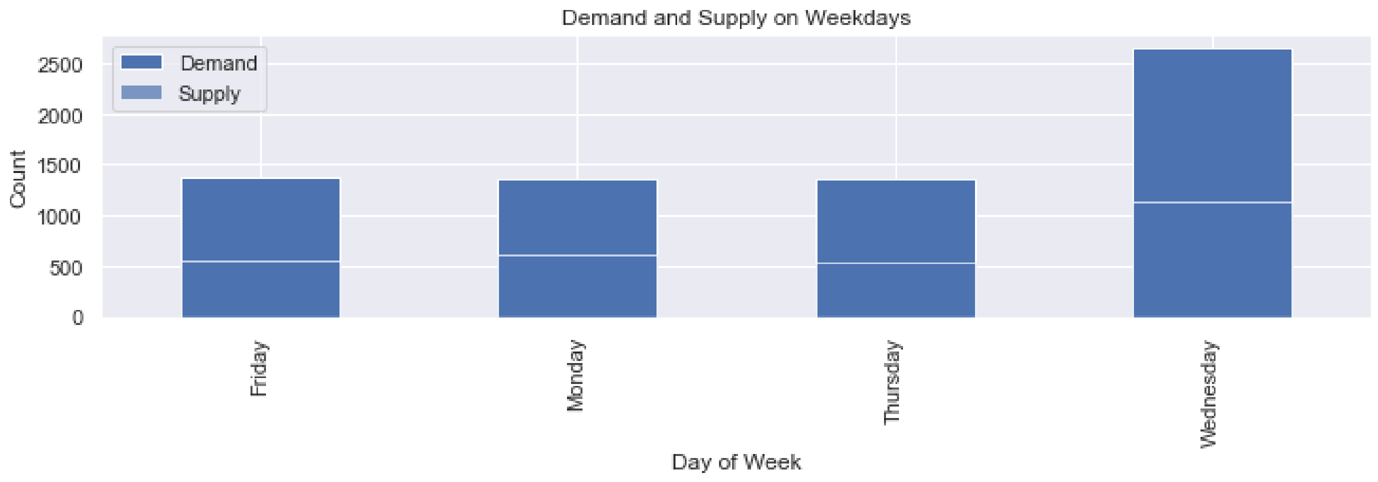
In [33]:

```
# Segmenting the data:

#segmenting pickup point over status
plt.figure(num=None, figsize=(12, 8), facecolor='w', edgecolor='k')
ax =sns.countplot(x='Pickup point', hue="Status", data=df)
total = len(df)
for p in ax.patches:
    value = round((p.get_height()/total)*100,2)
    X = p.get_x()+0.2
    Y = p.get_height()+20
    ax.text(X, Y, '{:} %'.format(value), ha="right")
plt.show()

#The above plot shows that for most of the  Airport pickup requests there are no cars av
#and most requests that get cancelled are from city pickup requests
```



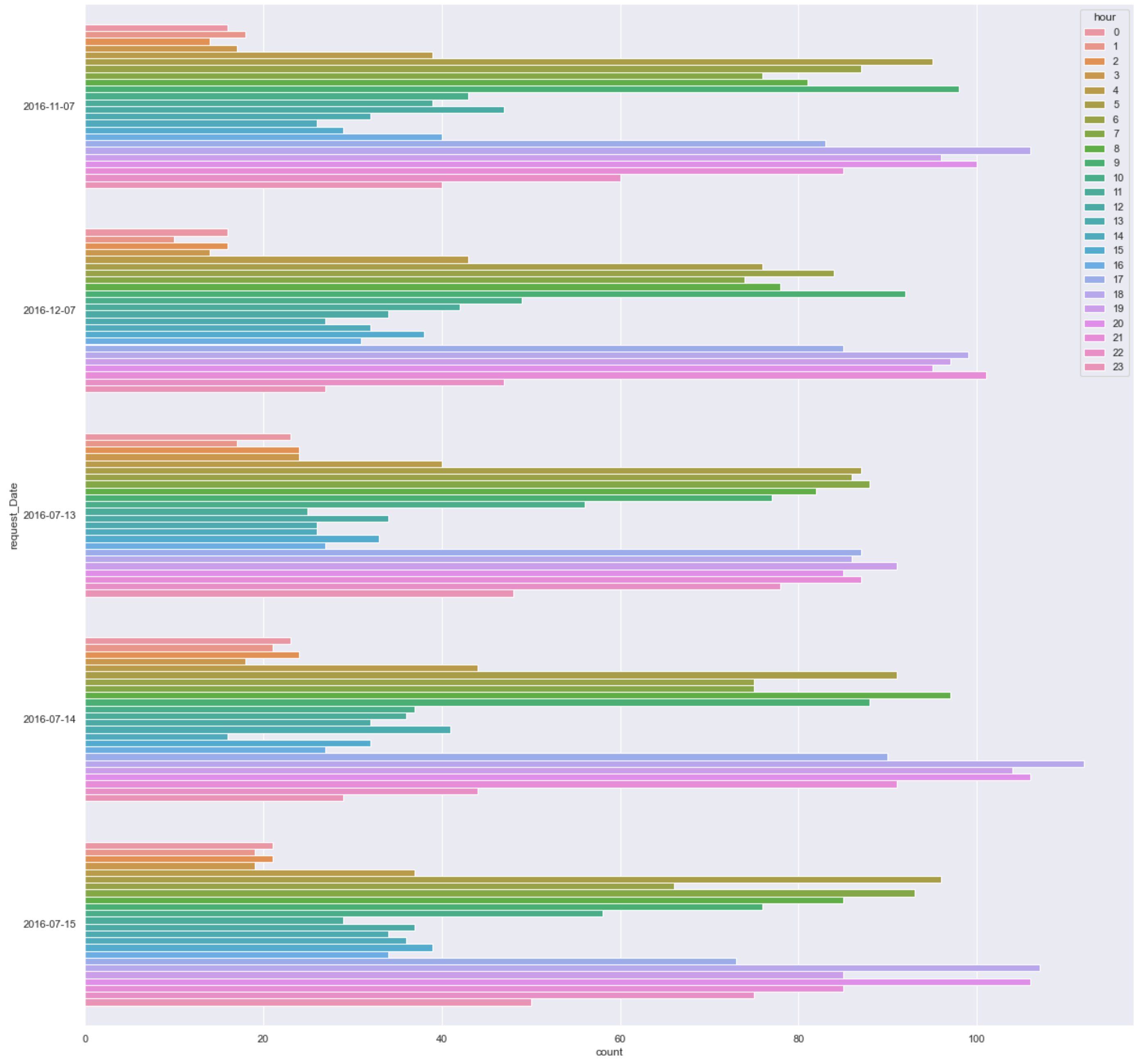
19)Are there any noticeable differences in demand and supply gaps on weekdays ?

```
In [34]: df['Day of Week'] = df['Request timestamp'].dt.day_name()


         weekdays = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
         weekends = ['Saturday', 'Sunday']

         demand_weekdays = df[df['Day of Week'].isin(weekdays)].groupby('Day of Week').size()
         supply_weekdays = df[(df['Status'] == 'Trip Completed') & (df['Day of Week'].isin(weekda

         demand_weekends = df[df['Day of Week'].isin(weekends)].groupby('Day of Week').size()
         supply_weekends = df[(df['Status'] == 'Trip Completed') & (df['Day of Week'].isin(weeken

         plt.figure(figsize=(12, 6))


         plt.subplot(2, 1, 1)
         demand_weekdays.plot(kind='bar', label='Demand')
         supply_weekdays.plot(kind='bar', label='Supply', alpha=0.7)
         plt.title('Demand and Supply on Weekdays')
         plt.xlabel('Day of Week')
         plt.ylabel('Count')
         plt.legend()

Out[34]: <matplotlib.legend.Legend at 0x22019904a60>
```



20)Check if high request rates from 5am to 9 am and 5pm to 10 pm is consistent throughout all days?

In [35]:
```
#to check if high request rates from 5am to 9am and 5pm to 10 pm is consistent throughou
df['request_Date'] = df['Request timestamp'].dt.date
plt.figure(num=None, figsize=(20, 20), facecolor='w', edgecolor='b')
sns.countplot(y='request_Date',hue="hour", data=df)
plt.show()

#shows that all dates high request rates is around the same time -5am to 9am and 5pm to
```



Hypothesis :

Pickup Point - City :

As per the analysis, the morning time slot is most problematic where the requests are being cancelled. Most probably the requests are being cancelled by the drivers due to the morning rush as it being the office hours and seeing the destination as airport which would be too far, the driver would think to earn more for the shorter trips within the city.

Pickup Point - Airport :

Upon analysis, the evening time slot seems to be most problematic for pickup points as airport where the requests being No Cars Available. The reason seems to be that not enough cars are available to service the requests as cars might not be available at the airport due to the cars serving inside the city.

Conclusions :

- Based on the data analysis performed, following recommendation can be used by Uber to bridge the gap between supply and demand: -
    - For bridging the demand supply gap from airport to city, making a permanent stand in the airport itself where the cabs will be available at all times and the incomplete requests can come down significantly.
    - Uber can provide some incentives to the driver who complete the trip from city to airport in the morning part. This might result the driver to not cancel the request from city to airport trips.
    - Last but sure solution to bring down the gap is to increase the numbers of cab in its fleet.