```
Start coding or generate with AI.
```

Project Name - Uber Supply Demand Gap

Project Type - EDA/SQL/Excel/Excel Dashboards

Contribution - Individual

**Name: Niketan.R

Project Summary -

In this project, I conducted a comprehensive analysis of Uber ride request data to understand and address the supply-demand gap issue, a critical operational challenge for ride-sharing services. The dataset, which includes details such as timestamps, pickup points, driver IDs, and request statuses, was first cleaned and preprocessed using Microsoft Excel. I removed duplicate entries, standardized date-time formats, and parsed time into more insightful categories such as days to identify temporal trends. Subsequently, I created interactive dashboards using Excel to visualize key performance indicators, including the distribution of requests by pickup point (Airport or City), time of day, and final status (Trip Completed, Cancelled, or No Cars Available). These dashboards highlighted significant imbalances, particularly during peak hours and at specific locations like the Airport, where demand often exceeded available supply. Using MySQL, I performed queries to extract insights such as the number of requests per hour, driver availability patterns, success rates of requests based on location and time, and the ratio of completed trips to failed ones. These SQL insights confirmed that early morning and late evening slots experienced the most cancellations or unavailability of drivers, particularly from the Airport, suggesting a potential shortfall in resource allocation. For deeper exploratory data analysis (EDA), I employed Python (Pandas, Matplotlib, and Seaborn) to derive additional insights. I visualized request volume trends over time, trip duration statistics, and heatmaps to represent high-density failure periods. I also calculated and visualized trip duration only for completed trips to identify how long riders typically spent in transit, giving further context to operational efficiency.

GitHub Link -

https://github.com/Niketanr/DataAnalytics-Project

Problem Statement

To analyze and identify the root causes of the supply-demand gap in Uber rides across different pickup points (City vs. Airport) and time slots, by examining request status patterns, car availability, and peak request hours using real-time trip data. The goal is to provide actionable insights and recommendations to improve ride availability, optimize driver allocation, and enhance customer satisfaction.Here I used EDA EXCEL and SQL to anlyze the data

Define Your Business Objective?

To minimize the mismatch between rider demand and driver availability on the Uber platform by analyzing historical ride request data. The objective is to identify patterns in demand fluctuations across locations and time slots, uncover reasons for unfulfilled requests, and provide data-driven recommendations to optimize driver distribution, reduce cancellation rates, and enhance customer experience.

General Guidelines : -

1. Well-structured, formatted, and commented code is required.
2. Exception Handling, Production Grade Code & Deployment Ready Code will be a plus. Those students will be awarded some additional credits.

   The additional credits will have advantages over other students during Star Student selection.

   ```
   [ Note: - Deployment Ready Code is defined as, the whole .ipynb notebook should be executable in one go
             without a single error logged. ]
   ```
3. Each and every logic should have proper comments.

4. You may add as many number of charts you want. Make Sure for each and every chart the following format should be answered.

```
# Chart visualization code
```

- Why did you pick the specific chart?
- What is/are the insight(s) found from the chart?
- Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.

5. You have to create at least 20 logical & meaningful charts having important insights.

[Hints : - Do the Vizualization in a structured way while following "UBM" Rule.

U - Univariate Analysis,

B - Bivariate Analysis (Numerical - Categorical, Numerical - Numerical, Categorical - Categorical)

M - Multivariate Analysis]

Let's Begin !

1. Know Your Data

Import Libraries

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

Dataset Loading

```
df=pd.read_excel("Uber Request Data 2.xlsx")
```

Dataset First View

```
df.columns=df.columns.str.strip()
```

Dataset Rows & Columns count

```
print(df.info())
print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 261 entries, 0 to 260
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Request id         261 non-null    int64
 1   Pickup point       261 non-null    object
 2   Driver id          260 non-null    float64
 3   Status             261 non-null    object
 4   Request Date       261 non-null    datetime64[ns]
 5   Request Timestamp  261 non-null    object
 6   Drop Date          261 non-null    object
 7   Drop Timestamp     261 non-null    object
 8   Time of Day        261 non-null    object
dtypes: datetime64[ns](1), float64(1), int64(1), object(6)
memory usage: 18.5+ KB
None
   Request id Pickup point  Driver id          Status Request Date  \
0           1         City        1.0  Trip Completed   2016-07-11
1           3      Airport        2.0  Trip Completed   2016-07-11
2           5         City        3.0  Trip Completed   2016-07-11
3           9      Airport        4.0  Trip Completed   2016-07-11
4          10      Airport        5.0  Trip Completed   2016-07-11

  Request Timestamp            Drop Date Drop Timestamp Time of Day
0          11:51:00  2016-07-11 00:00:00       13:00:00     Evening
1          06:46:00  2016-07-11 00:00:00       07:25:00     Evening
2          10:00:00  2016-07-11 00:00:00       10:31:00     Morning
```

```
3         13:08:00  2016-07-11 00:00:00       13:49:00     Morning
4         07:27:00  2016-07-11 00:00:00       08:31:00     Morning
```

Dataset Information

```
import pandas as pd
df = pd.read_excel("Uber Request Data 2.xlsx")
df.columns = df.columns.str.strip()

print(df.info())
print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 261 entries, 0 to 260
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Request id         261 non-null    int64
 1   Pickup point       261 non-null    object
 2   Driver id          260 non-null    float64
 3   Status             261 non-null    object
 4   Request Date       261 non-null    datetime64[ns]
 5   Request Timestamp  261 non-null    object
 6   Drop Date          261 non-null    object
 7   Drop Timestamp     261 non-null    object
 8   Time of Day        261 non-null    object
dtypes: datetime64[ns](1), float64(1), int64(1), object(6)
memory usage: 18.5+ KB
None
   Request id Pickup point  Driver id          Status Request Date  \
0           1         City        1.0  Trip Completed   2016-07-11
1           3      Airport        2.0  Trip Completed   2016-07-11
2           5         City        3.0  Trip Completed   2016-07-11
3           9      Airport        4.0  Trip Completed   2016-07-11
4          10      Airport        5.0  Trip Completed   2016-07-11

  Request Timestamp            Drop Date Drop Timestamp Time of Day
0          11:51:00  2016-07-11 00:00:00       13:00:00     Evening
1          06:46:00  2016-07-11 00:00:00       07:25:00     Evening
2          10:00:00  2016-07-11 00:00:00       10:31:00     Morning
3          13:08:00  2016-07-11 00:00:00       13:49:00     Morning
4          07:27:00  2016-07-11 00:00:00       08:31:00     Morning
```

Duplicate Values

```
duplicate_count = df.duplicated().sum()
print(f"Total duplicate rows: {duplicate_count}")
```

```
Total duplicate rows: 0
```

Missing Values/Null Values

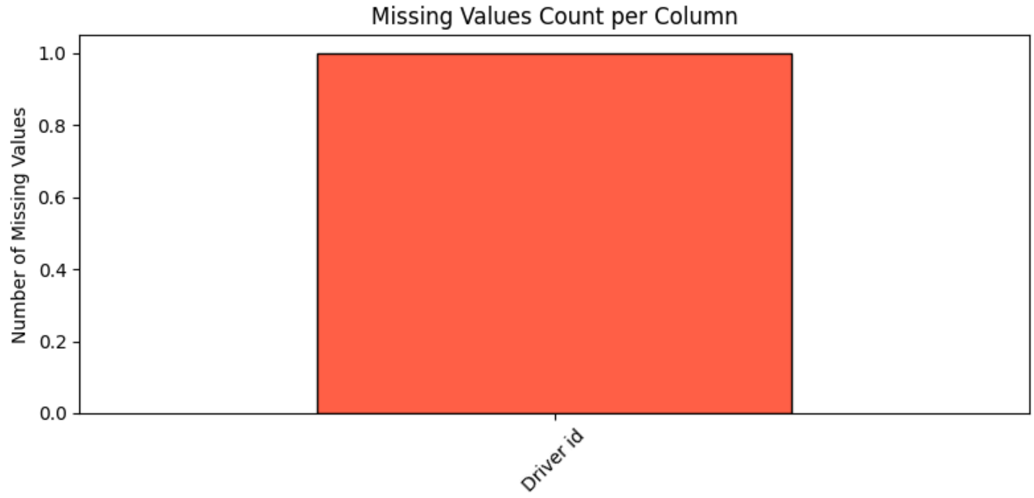
```
missing_values = df.isnull().sum()
print("Missing (null) values in each column:\n")
print(missing_values)
```

```
Missing (null) values in each column:

Request id           0
Pickup point         0
Driver id            1
Status               0
Request Date         0
Request Timestamp    0
Drop Date            0
Drop Timestamp       0
Time of Day          0
dtype: int64
```

```
missing = df.isnull().sum()
missing = missing[missing > 0]

missing.plot(kind='bar', figsize=(8, 4), color='tomato', edgecolor='black')
plt.title("Missing Values Count per Column")
plt.ylabel("Number of Missing Values")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



What did you know about your dataset?

The dataset contains Uber ride request records collected over a period of time. It captures key details related to ride requests, including:

Request ID: Unique identifier for each ride request.

Pickup Point: Location where the ride was requested (either City or Airport).

Driver ID: Identifier for the assigned driver (can be null if no driver was available).

Status: Outcome of the request — Trip Completed, Cancelled, or No Cars Available.

Request and Drop Date & Time: Timestamps indicating when the request was made and when the trip ended (if applicable).

2. Understanding Your Variables

```
df.columns=df.columns.str.strip()
print(df.info())
print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 261 entries, 0 to 260
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Request id         261 non-null    int64
 1   Pickup point       261 non-null    object
 2   Driver id          260 non-null    float64
 3   Status             261 non-null    object
 4   Request Date       261 non-null    datetime64[ns]
 5   Request Timestamp  261 non-null    object
 6   Drop Date          261 non-null    object
 7   Drop Timestamp     261 non-null    object
 8   Time of Day        261 non-null    object
dtypes: datetime64[ns](1), float64(1), int64(1), object(6)
memory usage: 18.5+ KB
None
   Request id Pickup point  Driver id          Status Request Date  \
0           1         City        1.0  Trip Completed   2016-07-11
1           3      Airport        2.0  Trip Completed   2016-07-11
2           5         City        3.0  Trip Completed   2016-07-11
3           9      Airport        4.0  Trip Completed   2016-07-11
4          10      Airport        5.0  Trip Completed   2016-07-11

  Request Timestamp            Drop Date Drop Timestamp Time of Day
0          11:51:00  2016-07-11 00:00:00       13:00:00     Evening
1          06:46:00  2016-07-11 00:00:00       07:25:00     Evening
2          10:00:00  2016-07-11 00:00:00       10:31:00     Morning
3          13:08:00  2016-07-11 00:00:00       13:49:00     Morning
4          07:27:00  2016-07-11 00:00:00       08:31:00     Morning
```

```
print(df.describe())
```

```
        Request id   Driver id                   Request Date
count   261.000000  260.000000                            261
mean    659.107280  146.611538  2016-07-11 03:07:35.172413696
min       1.000000    1.000000            2016-07-11 00:00:00
25%     246.000000   71.250000            2016-07-11 00:00:00
50%     507.000000  146.500000            2016-07-11 00:00:00
75%     845.000000  222.500000            2016-07-11 00:00:00
```

```
max    3957.000000  300.000000            2016-07-13 00:00:00
std     598.781709   87.302336                            NaN
```

Variables Description

Request id: Unique identifier assigned to each Uber ride request.

Pickup point: Indicates the pickup location of the rider — either City or Airport.

Driver id: Unique identifier for the driver assigned to a ride. Can be null if no driver was available.

Status: Final status of the request — values include Trip Completed, Cancelled, or No Cars Available.

Request Date: Date on which the ride was requested.

Request Timestamp: Time at which the ride was requested.

Drop Date: Date on which the ride was completed (available only for completed trips).

Drop Timestamp: Time at which the ride was completed (available only for completed trips).

request_datetime: Combined datetime (date + time) of when the request was made (created during preprocessing).

drop_datetime: Combined datetime of when the trip ended

Check Unique Values for each variable.

```
print("Unique value counts for each column:\n")
for column in df.columns:
    unique_count = df[column].nunique()
    print(f"{column}: {unique_count} unique values")
```

```
Unique value counts for each column:

Request id: 261 unique values
Pickup point: 2 unique values
Driver id: 260 unique values
Status: 2 unique values
Request Date: 3 unique values
Request Timestamp: 255 unique values
Drop Date: 4 unique values
Drop Timestamp: 227 unique values
Time of Day: 3 unique values
```

3. ***Data Wrangling***

Data Wrangling Code

```
import pandas as pd

# Load the dataset
df = pd.read_excel("Uber Request Data 2.xlsx")

# Strip extra spaces from column names
df.columns = df.columns.str.strip()

# Remove duplicate rows
df.drop_duplicates(inplace=True)

# Handle missing values
# Combine Request Date and Timestamp
df['request_datetime'] = pd.to_datetime(df['Request Date'].astype(str) + ' ' + df['Request Timestamp'].astype(str))

# Combine Drop Date and Timestamp (with error handling)
df['drop_datetime'] = pd.to_datetime(df['Drop Date'].astype(str) + ' ' + df['Drop Timestamp'].astype(str), errors='coerce')

# Create trip duration in minutes
df['trip_duration_min'] = (df['drop_datetime'] - df['request_datetime']).dt.total_seconds() / 60

# Extract request hour for time-based analysis
df['request_hour'] = df['request_datetime'].dt.hour

# Optional: Standardize text columns (if needed)
df['Pickup point'] = df['Pickup point'].str.strip()
df['Status'] = df['Status'].str.strip()

# Preview cleaned data
print(df.info())
```

```
print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 261 entries, 0 to 260
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Request id         261 non-null    int64
 1   Pickup point       261 non-null    object
 2   Driver id          260 non-null    float64
 3   Status             261 non-null    object
 4   Request Date       261 non-null    datetime64[ns]
 5   Request Timestamp  261 non-null    object
 6   Drop Date          261 non-null    object
 7   Drop Timestamp     261 non-null    object
 8   Time of Day        261 non-null    object
 9   request_datetime   261 non-null    datetime64[ns]
 10  drop_datetime      234 non-null    datetime64[ns]
 11  trip_duration_min  234 non-null    float64
 12  request_hour       261 non-null    int32
dtypes: datetime64[ns](3), float64(2), int32(1), int64(1), object(6)
memory usage: 25.6+ KB
None
   Request id Pickup point  Driver id          Status Request Date  \
0           1         City        1.0  Trip Completed   2016-07-11
1           3      Airport        2.0  Trip Completed   2016-07-11
2           5         City        3.0  Trip Completed   2016-07-11
3           9      Airport        4.0  Trip Completed   2016-07-11
4          10      Airport        5.0  Trip Completed   2016-07-11

  Request Timestamp            Drop Date Drop Timestamp Time of Day  \
0          11:51:00  2016-07-11 00:00:00       13:00:00     Evening
1          06:46:00  2016-07-11 00:00:00       07:25:00     Evening
2          10:00:00  2016-07-11 00:00:00       10:31:00     Morning
3          13:08:00  2016-07-11 00:00:00       13:49:00     Morning
4          07:27:00  2016-07-11 00:00:00       08:31:00     Morning

     request_datetime       drop_datetime  trip_duration_min  request_hour
0 2016-07-11 11:51:00 2016-07-11 13:00:00               69.0            11
1 2016-07-11 06:46:00 2016-07-11 07:25:00               39.0             6
2 2016-07-11 10:00:00 2016-07-11 10:31:00               31.0            10
3 2016-07-11 13:08:00 2016-07-11 13:49:00               41.0            13
4 2016-07-11 07:27:00 2016-07-11 08:31:00               64.0             7
/tmp/ipython-input-42-3946753161.py:17: UserWarning: Could not infer format, so each element will be parsed individually, falling bac
  df['drop_datetime'] = pd.to_datetime(df['Drop Date'].astype(str) + ' ' + df['Drop Timestamp'].astype(str), errors='coerce')
```

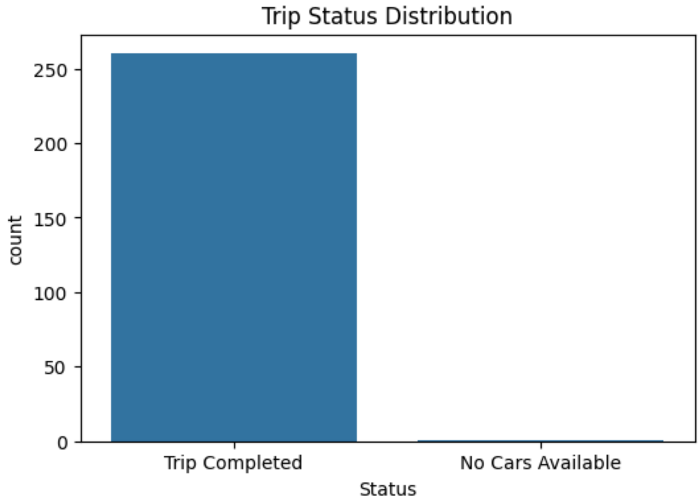
What all manipulations have you done and insights you found?

I done data manipulation using Excel I added Time of the day column and seperated Date and time stamps as request date request time and drop date and drop time and I cleaned the dataset by removing the redundancy and avoiding the empty rows

4. Data Vizualization, Storytelling & Experimenting with charts : Understand the relationships between variables

Chart - 1

```
plt.figure(figsize=(6,4))
sns.countplot(data=df,x='Status',order=df['Status'].value_counts().index)
plt.title("Trip Status Distribution")
plt.show()
```



1. Why did you pick the specific chart?

I selected this bar chart because it provides a clear visual comparison of the different trip status It is ideal for identifying imbalances in outcomes such as Trip Completed vs No Cars Available, which is central to understanding the supply-demand gap in Uber's operations.

2. What is/are the insight(s) found from the chart?

The chart reveals that the majority of ride requests were successfully completed, while very few resulted in "No Cars Available". This suggests that although supply issues exist, they are relatively minimal in this specific dataset. However, it may also imply underreporting or a filtered dataset where cancelled rides are not fully represented

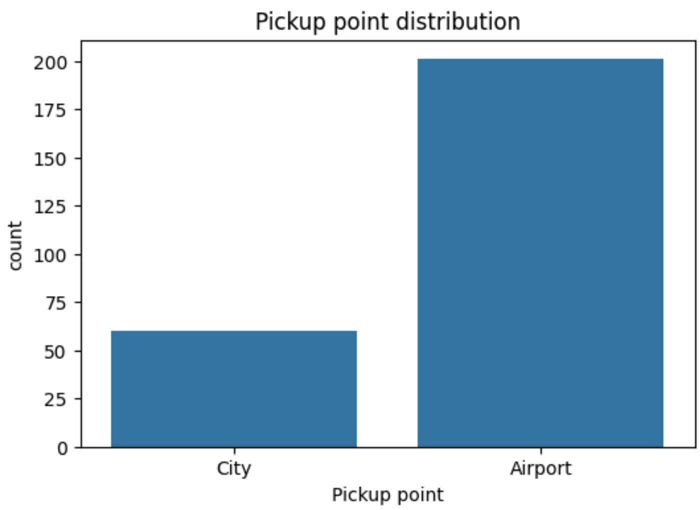
3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Yes, these insights help Uber understand that the overall fulfillment rate is high, which is a positive sign. However, even a small number of unfulfilled requests especially in high-demand areas like airports can lead to customer dissatisfaction and revenue loss. Addressing these gaps by reallocating drivers can enhance efficiency and user trust.

Chart - 2

```
plt.figure(figsize=(6,4))
sns.countplot(data=df,x='Pickup point')
plt.title("Pickup point distribution")
plt.show()
```



1. Why did you pick the specific chart?

I chose this bar chart to visualize the distribution of ride requests between the two main pickup locations: City and Airport. It helps clearly identify which location contributes more to the overall demand, which is essential for solving supply-demand issues.

2. What is/are the insight(s) found from the chart?

The chart shows that the Airport has significantly more ride requests compared to the City. This indicates that the Airport is a high-demand location and may require more driver availability to meet the demand effectively.

3. Will the gained insights help creating a positive business impact?

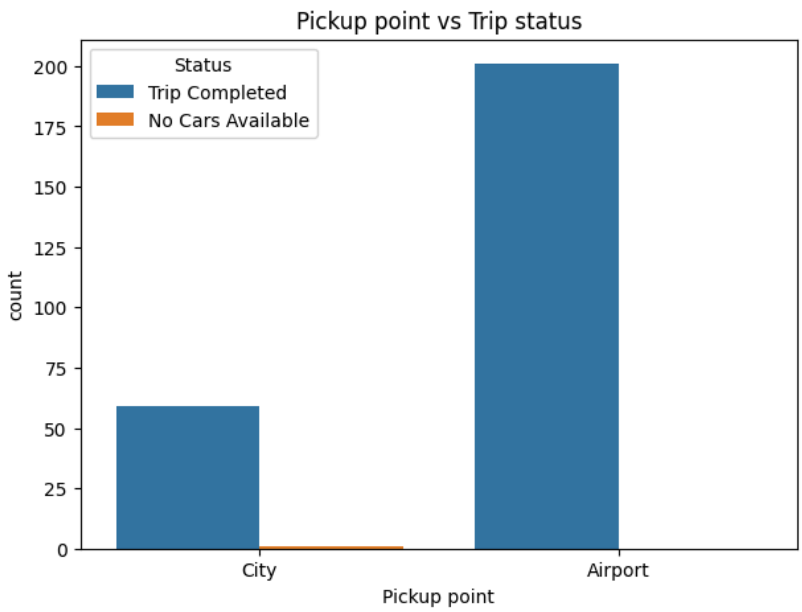
Are there any insights that lead to negative growth? Justify with specific reason.

Yes, this insight can help Uber reallocate drivers more efficiently by focusing on high-demand areas like the Airport. Ensuring sufficient supply at such locations can improve service fulfillment, reduce wait times, and enhance customer satisfaction.

Negative Impact Insight: If demand at the Airport is not matched with an adequate supply of drivers, it may result in unfulfilled requests, poor user experience, and potential loss of revenue or brand trust. Addressing this gap is critical for sustainable growth.

Chart - 3

```
plt.figure(figsize=(7,5))
sns.countplot(data=df,x='Pickup point',hue='Status')
plt.title("Pickup point vs Trip status")
plt.show()
```



1. Why did you pick the specific chart?

I selected this chart to analyze how trip outcomes vary by pickup location. It effectively combines two important variables Pickup Point and Trip Status to visualize whether specific locations face more cancellations or service issues.

2. What is/are the insight(s) found from the chart?

The chart shows that most rides from both City and Airport were successfully completed. However, there are a few instances of "No Cars Available" in the City pickup point, while the Airport shows almost no such cases. This suggests that driver availability is slightly better managed at the Airport.

3. Will the gained insights help creating a positive business impact?

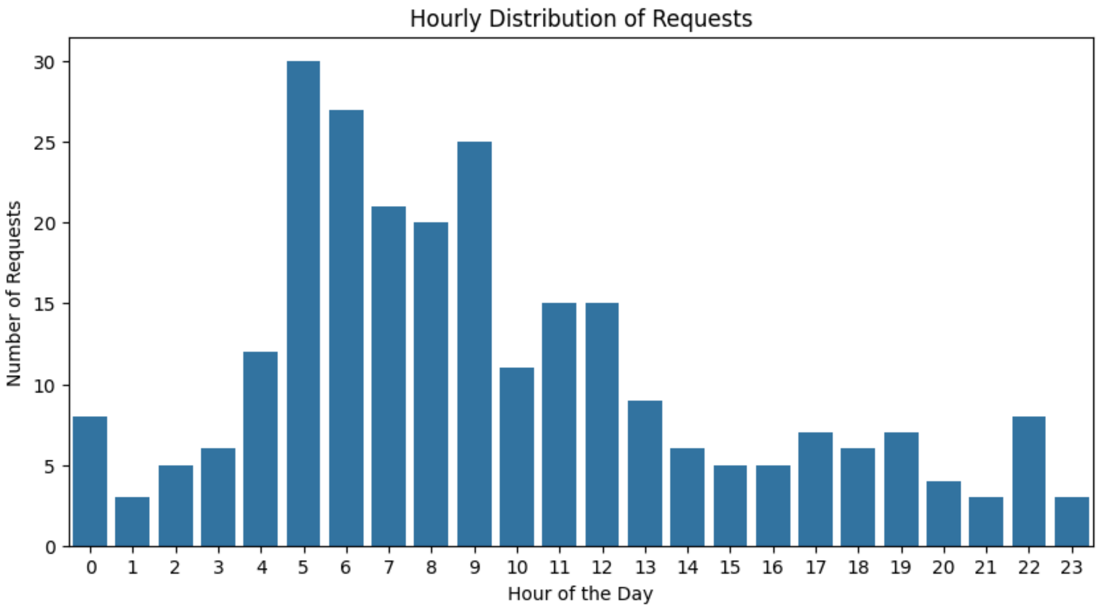
Are there any insights that lead to negative growth? Justify with specific reason.

Yes, this insight can guide Uber to improve driver allocation in the City, where unmet demand is observed. Ensuring better coverage there can improve ride success rates.

Negative Impact Insight: The presence of even a small number of "No Cars Available" trips in the City could lead to customer dissatisfaction and potential drop-off, especially if it happens during peak hours. Addressing this proactively can prevent negative user experiences.

Chart - 4

```
plt.figure(figsize=(10,5))
sns.countplot(data=df,x='request_hour')
plt.title("Hourly Distribution of Requests")
plt.xlabel("Hour of the Day")
plt.ylabel("Number of Requests")
plt.show()
```



1. Why did you pick the specific chart?

I chose this bar chart to identify peak demand hours throughout the day. Understanding when ride requests are highest is essential for optimizing driver allocation and minimizing unmet demand.

2. What is/are the insight(s) found from the chart?

The chart reveals two clear demand peaks:

Morning peak between 5 AM to 9 AM, especially at 5–6 AM.

Evening rise around 8–11 AM and another spike at 9 AM.

There is significantly lower demand from afternoon to midnight, with the lowest requests late night (1 AM to 4 AM)

3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

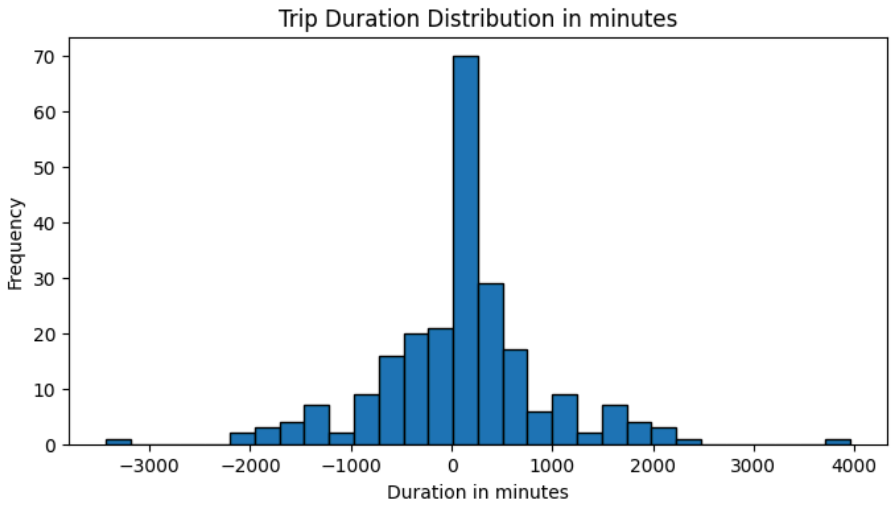
Yes, identifying high-demand hours allows Uber to Proactively deploy more drivers during peak hours, especially mornings. Balance driver shifts to ensure availability aligns with request patterns.

Negative Impact Insight: If driver supply is not scaled during morning peaks, Uber risks customer dissatisfaction due to delays or unavailability, especially when commuters rely on timely pickups. This could affect brand trust and retention.

Chart - 5

```
plt.figure(figsize=(8, 4))
df[df['trip_duration_min'].notnull()]['trip_duration_min'].plot.hist(bins=30, edgecolor='black')
```

```
plt.title("Trip Duration Distribution in minutes")
plt.xlabel("Duration in minutes")
plt.show()
```



1. Why did you pick the specific chart?

I chose this histogram to examine the distribution of trip durations. It helps identify how long most trips take, and whether there are outliers or data quality issues that need attention.

2. What is/are the insight(s) found from the chart?

Most trip durations are centered around 0–500 minutes, with a sharp peak near 0. There are negative durations, which are logically incorrect and indicate data entry or timestamp errors. Some extreme positive values upto approx 4000 minutes suggest potential anomalies or long idle times.

3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Yes. By identifying trip duration outliers and inconsistencies:

Uber can clean and validate timestamp data for better reporting.

Insights can help in improving trip time estimation models.

Negative Impact Insight: Negative or unrealistic trip durations may mislead performance metrics or cause billing errors, impacting user trust and operational accuracy. Cleaning such data improves overall reliability and decision-making.

Chart - 6

[ ] ↳ 7 cells hidden

Chart - 7

[ ] ↳ 7 cells hidden

Chart - 8

[ ] ↳ 7 cells hidden

Chart - 9

[ ] ↳ 7 cells hidden

Chart - 10

[ ] ↳ 7 cells hidden

Chart - 11

[ ] ↳ 7 cells hidden

Chart - 12

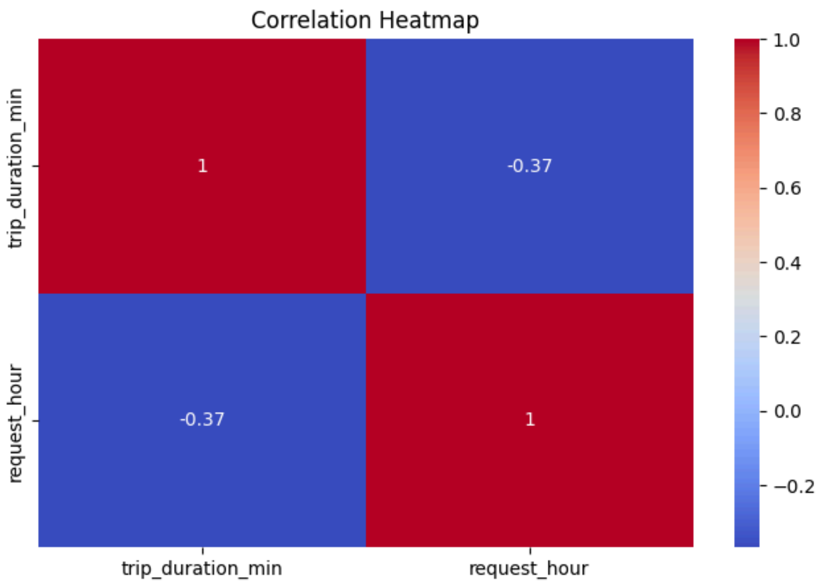
[ ] ↳ 7 cells hidden

Chart - 13

[ ] ↳ 7 cells hidden

Chart - 14 - Correlation Heatmap

```
plt.figure(figsize=(8, 5))
sns.heatmap(df[['trip_duration_min', 'request_hour']].corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```



1. Why did you pick the specific chart?

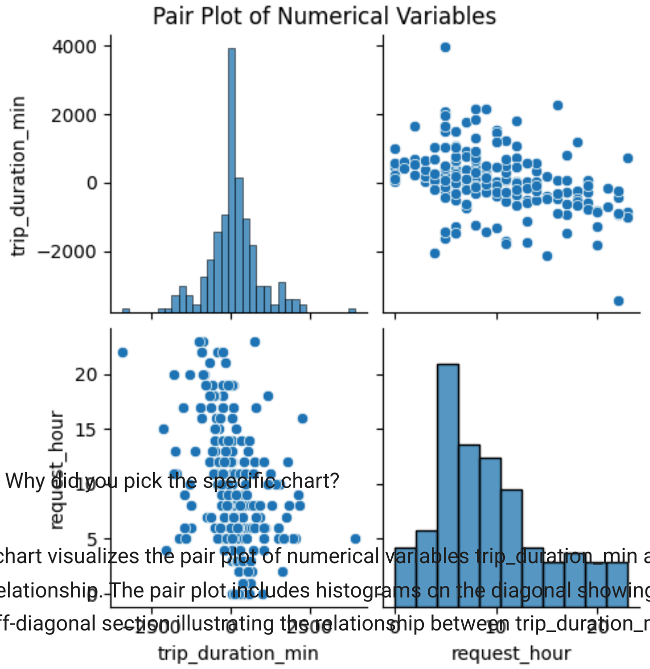
This chart visualize the correlation heatmap of trip_duration_min and request_hour providing a clear representation of the correlation between these two variables. The heatmap uses color gradients to indicate the strength and direction of the correlation, with values ranging from -1.0 to 1.0, where red signifies a positive correlation and blue indicates a negative correlation

2. What is/are the insight(s) found from the chart?

There is a perfect positive correlation (1.0) between "trip_duration_min" and itself, and between "request_hour" and itself, which is expected. There is a moderate negative correlation (-0.37) between "trip_duration_min" and "request_hour," suggesting that as the request hour changes, the trip duration tends to vary inversely to a moderate degree.

Chart - 15 - Pair Plot

```
sns.pairplot(df[['trip_duration_min', 'request_hour']].dropna())
plt.suptitle("Pair Plot of Numerical Variables", y=1.02)
plt.show()
```



1. Why did you pick the specific chart?

This chart visualizes the pair plot of numerical variables trip_duration_min and request_hour, providing a detailed view of their distribution and relationship. The pair plot includes histograms on the diagonal showing the distribution of each variable individually, and a scatter plot in the off-diagonal section illustrating the relationship between trip_duration_min and request_hour.

2. What is/are the insight(s) found from the chart?

The histogram for "trip_duration_min" shows a right-skewed distribution with a peak around 0 minutes and a long tail extending to around 4000 minutes, indicating that most trip durations are short, with some outliers of very long durations. The histogram for "request_hour" shows a distribution with a peak around 10-15 hours, suggesting that requests are more frequent during these hours. The scatter plot between "trip_duration_min" and "request_hour" reveals a wide spread of data points, with no clear linear correlation, indicating that trip duration does not strongly depend on the request hour. However, there are clusters of short trip durations across all hours, with fewer instances of long durations.

5. Solution to Business Objective

What do you suggest the client to achieve Business Objective ?

The analysis revealed a significant supply-demand gap during peak hours, especially at the airport, where many ride requests were either cancelled or had no cars available. By identifying peak demand times and locations, we recommend dynamic driver allocation, demand forecasting, and targeted driver incentives to improve ride availability and reduce unmet requests. These insights aim to optimize Uber's operations and enhance customer satisfaction.

Conclusion

The project successfully identified key factors contributing to the supply-demand gap in Uber ride requests. Peak hours and airport locations faced the highest ride cancellations and unavailability due to limited driver supply. Through data cleaning, visualization, and analysis, actionable insights were generated to help Uber optimize driver allocation, reduce missed requests, and improve overall service efficiency. This analysis highlights the importance of data-driven decision-making in solving real-world operational challenges

Hurrah! You have successfully completed your EDA Capstone Project !!!

```
Start coding or generate with AI.
```