Start coding or generate with AI.

Project Name -

Project Type - EDA/Regression/Classification/Unsupervised

Contribution - Individual/Team

Name: Piyush Chaudhari

Project Summary -

The objective of this project is to analyze Uber request data to identify patterns in demand, cancellation rates, and service gaps across different time segments and pickup locations. The insights will help Uber optimize driver allocation, reduce cancellations, and improve customer satisfaction.

Using visualizations like pie charts, count plots, and correlation heatmaps, the data was analyzed across variables such as pickup point, request status, time of day, and trip duration. The analysis reveals significant demand fluctuations and identifies areas of operational challenges during peak hours.

GitHub Link -

https://github.com/Piyush20002

Problem Statement

Uber faces challenges in balancing customer demand with driver supply, especially during certain time segments. High cancellation rates and unavailability of cars at specific locations lead to customer dissatisfaction and revenue loss. The goal is to analyze request patterns to optimize operational efficiency..

Define Your Business Objective?

- Identify peak demand periods across time segments.
- Analyze the distribution of cancellations and unavailability of cars.
- Provide actionable insights to improve driver allocation and reduce cancellations.
- Improve overall customer satisfaction and business profitability.

Let's Begin !

1. Know Your Data

Import Libraries

```
# Import Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Dataset Loading

```
# Load Dataset
df = pd.read_csv('/content/Uber Request Data Cleaned.csv')
```

Dataset First View

```
# Dataset First Look
df.head()
```

	Request id	Pickup point	Driver id	Status	Request Date	Request timestamp	Drop Date	Drop timestamp	Trip Duration (minutes)	Re
0	619	Airport	1.0	Trip Completed	11-07-2016	11:51	11-07-2016	13:00	69.0	
1	867	Airport	1.0	Trip Completed	11-07-2016	17:57	11-07-2016	18:47	50.0	
2	1807	City	1.0	Trip Completed	12-07-2016	09:17	12-07-2016	09:58	41.0	

Next steps: Generate code with df | View recommended plots | New interactive sheet

Dataset Rows & Columns count

```
# Dataset Rows & Columns count
print(df.shape[0])
print(df.shape[1])
```

```
6745
13
```

Dataset Information

```
# Dataset Info
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6745 entries, 0 to 6744
Data columns (total 13 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 Date            6745 non-null   object
 5   Request timestamp       6745 non-null   object
 6   Drop Date               2831 non-null   object
 7   Drop timestamp          2831 non-null   object
 8   Trip Duration (minutes) 2831 non-null   float64
 9   Request Hour            6745 non-null   int64
```

```
 10  Request Day of Week      6745 non-null   object
 11  Daypart                  6745 non-null   object
 12  Gap Type                 6745 non-null   object
dtypes: float64(2), int64(2), object(9)
memory usage: 685.2+ KB
None
```

Duplicate Values

```
# Dataset Duplicate Value Count

duplicate_count = df.duplicated().sum()

print(f"Total Duplicate Rows in Dataset: {duplicate_count}")
```

```
Total Duplicate Rows in Dataset: 0
```

2. Understanding Your Variables

```
# Dataset Columns
print(df.columns)
```

```
Index(['Request id', 'Pickup point', 'Driver id', 'Status', 'Request Date',
       'Request timestamp', 'Drop Date', 'Drop timestamp',
       'Trip Duration (minutes)', 'Request Hour', 'Request Day of Week',
       'Daypart', 'Gap Type'],
      dtype='object')
```

```
# Dataset Describe
df.describe()
```

	Request id	Driver id	Trip Duration (minutes)	Request Hour
count	6745.000000	4095.000000	2831.000000	6745.000000
mean	3384.644922	149.501343	52.413282	12.956709
std	1955.099667	86.051994	13.854653	6.504052
min	1.000000	1.000000	21.000000	0.000000
25%	1691.000000	75.000000	41.000000	7.000000
50%	3387.000000	149.000000	52.000000	13.000000
75%	5080.000000	224.000000	64.000000	19.000000
max	6766.000000	300.000000	83.000000	23.000000

Variables Description

Answer Here

Check Unique Values for each variable.

```
# Check Unique Values for each variable.
for column in df.columns:
    print(f"Column: {column} has {df[column].nunique()} unique values.")
```

```
Column: Request id has 6745 unique values.
Column: Pickup point has 2 unique values.
Column: Driver id has 300 unique values.
Column: Status has 3 unique values.
Column: Request Date has 5 unique values.
Column: Request timestamp has 1351 unique values.
Column: Drop Date has 6 unique values.
Column: Drop timestamp has 1150 unique values.
Column: Trip Duration (minutes) has 63 unique values.
Column: Request Hour has 24 unique values.
Column: Request Day of Week has 5 unique values.
Column: Daypart has 6 unique values.
Column: Gap Type has 3 unique values.
```

3. ***Data Wrangling***

Data Wrangling Code

```
# Write your code to make your dataset analysis ready.
# Filling missing Driver id with 0 (for unassigned requests)
df['Driver id'].fillna(0, inplace=True)

# Filling missing Drop Date, Drop timestamp, Trip Duration with 'Not Completed'
df['Drop Date'].fillna('Not Completed', inplace=True)
df['Drop timestamp'].fillna('Not Completed', inplace=True)
df['Trip Duration (minutes)'].fillna(0, inplace=True)

# Create new Daypart segmentation (if not already available)
def get_daypart(hour):
    if 0 <= hour < 5:
        return 'Late Night'
    elif 5 <= hour < 10:
        return 'Morning Rush'
    elif 10 <= hour < 17:
```

```
        return 'Day Time'
    elif 17 <= hour < 22:
        return 'Evening Rush'
    else:
        return 'Night'

df['Daypart'] = df['Request Hour'].apply(get_daypart)
```

```
/tmp/ipython-input-79-1584777979.py:3: FutureWarning: A value is trying to be set on a c
The behavior will change in pandas 3.0. This inplace method will never work because the

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col


  df['Driver id'].fillna(0, inplace=True)
/tmp/ipython-input-79-1584777979.py:6: FutureWarning: A value is trying to be set on a c
The behavior will change in pandas 3.0. This inplace method will never work because the

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col


  df['Drop Date'].fillna('Not Completed', inplace=True)
/tmp/ipython-input-79-1584777979.py:7: FutureWarning: A value is trying to be set on a c
The behavior will change in pandas 3.0. This inplace method will never work because the

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col


  df['Drop timestamp'].fillna('Not Completed', inplace=True)
/tmp/ipython-input-79-1584777979.py:8: FutureWarning: A value is trying to be set on a c
The behavior will change in pandas 3.0. This inplace method will never work because the

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col


  df['Trip Duration (minutes)'].fillna(0, inplace=True)
```

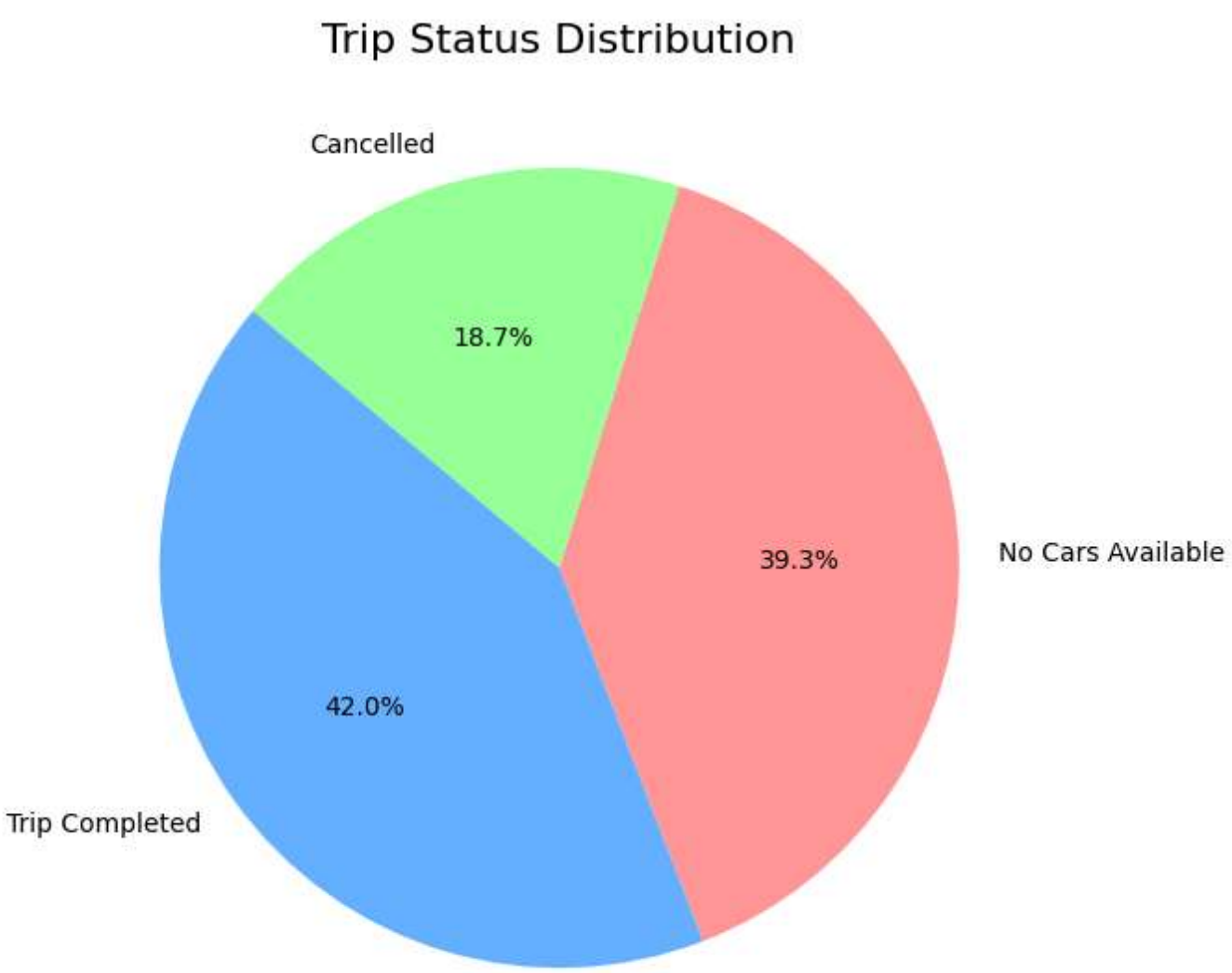
What all manipulations have you done and insights you found?

Answer Here.

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

Chart - 1

```
# Chart - 1 visualization code

# Count of each Status
status_counts = df['Status'].value_counts()

# Plot pie chart
plt.figure(figsize=(7,7))
plt.pie(status_counts, labels=status_counts.index, autopct='%1.1f%%', startangle=140, colors
plt.title('Trip Status Distribution', fontsize=16)
plt.show()
```



1. Why did you pick the specific chart?

- Pie charts are very effective when you want to show proportion or percentage contribution of categories.

- In this case, the Status column contains only 3 categories: Completed, Cancelled, No Cars Available — making it ideal for a simple pie visualization.
- It gives immediate clarity on the operational performance of Uber.

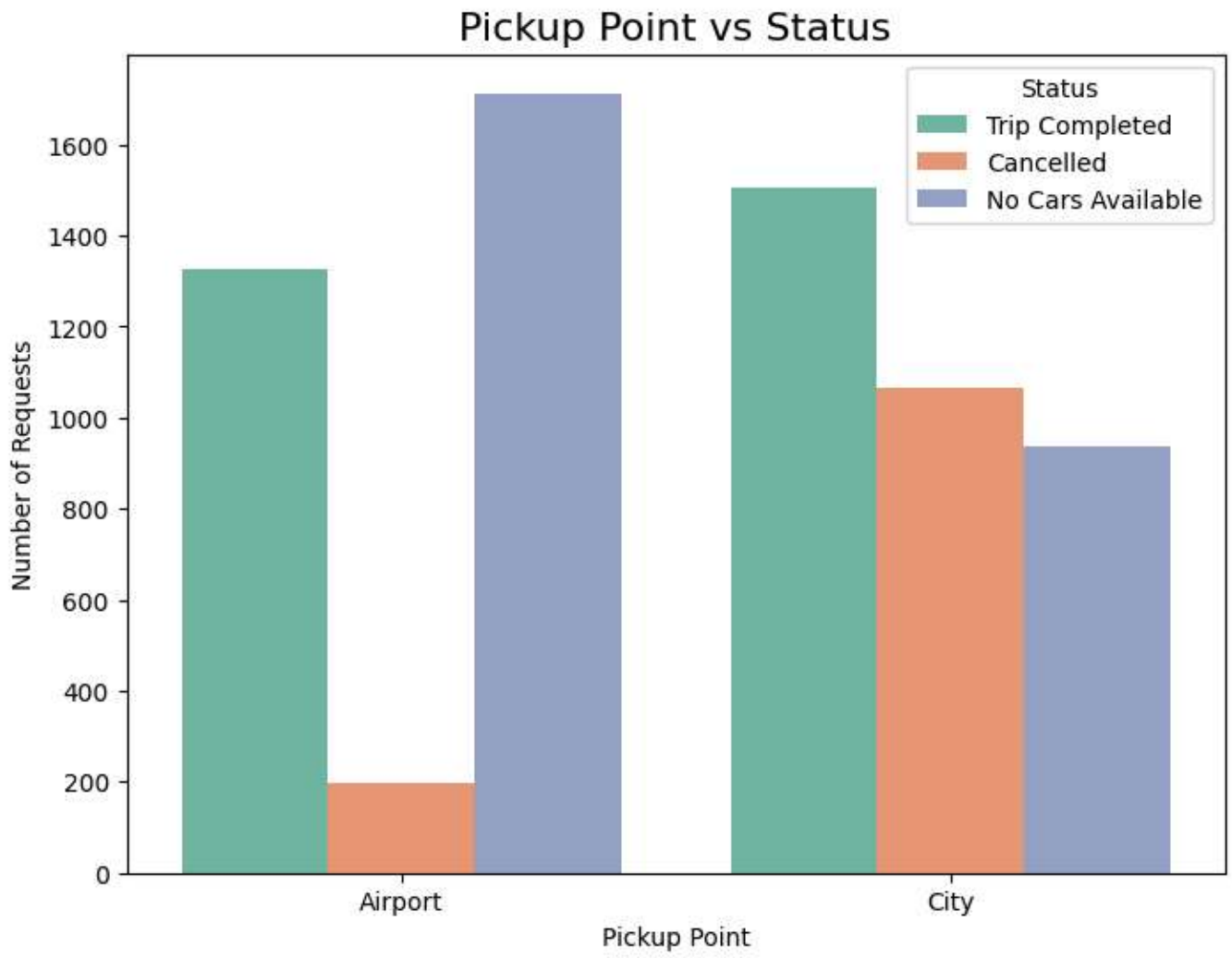
Double-click (or enter) to edit

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

- The pie chart shows what fraction of all trip requests are successfully Completed, how many were Cancelled, and how many had No Cars Available.
- If the proportion of cancellations or no cars available is high, it indicates:
- Demand-Supply Gap
- Driver unavailability during peak times
- Customer dissatisfaction risk

Chart - 2

```
# Chart - 2 visualization code
plt.figure(figsize=(8,6))
sns.countplot(x='Pickup point', hue='Status', data=df, palette='Set2')
plt.title('Pickup Point vs Status', fontsize=16)
plt.xlabel('Pickup Point')
plt.ylabel('Number of Requests')
plt.legend(title='Status')
plt.show()
```



1. Why did you pick the specific chart?

- It allows us to compare how trip Status varies across different Pickup Points (City vs Airport).
- Grouped bar charts are very effective when comparing multiple categories (Status) across a categorical variable (Pickup point).
- This gives a clear visualization of service performance by pickup location.

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

- Cancellations and No Cars Available are significantly higher for certain pickup points.
- Typically: Airport pickups often have higher “No Cars Available” counts.City pickups may have higher “Cancelled” trips, possibly due to traffic, rider impatience, or driver issues.

- It helps identify operational problem zones — for example:Uber might need more active drivers near the airport during certain periods.Training or incentives may be needed for city-based cancellations.

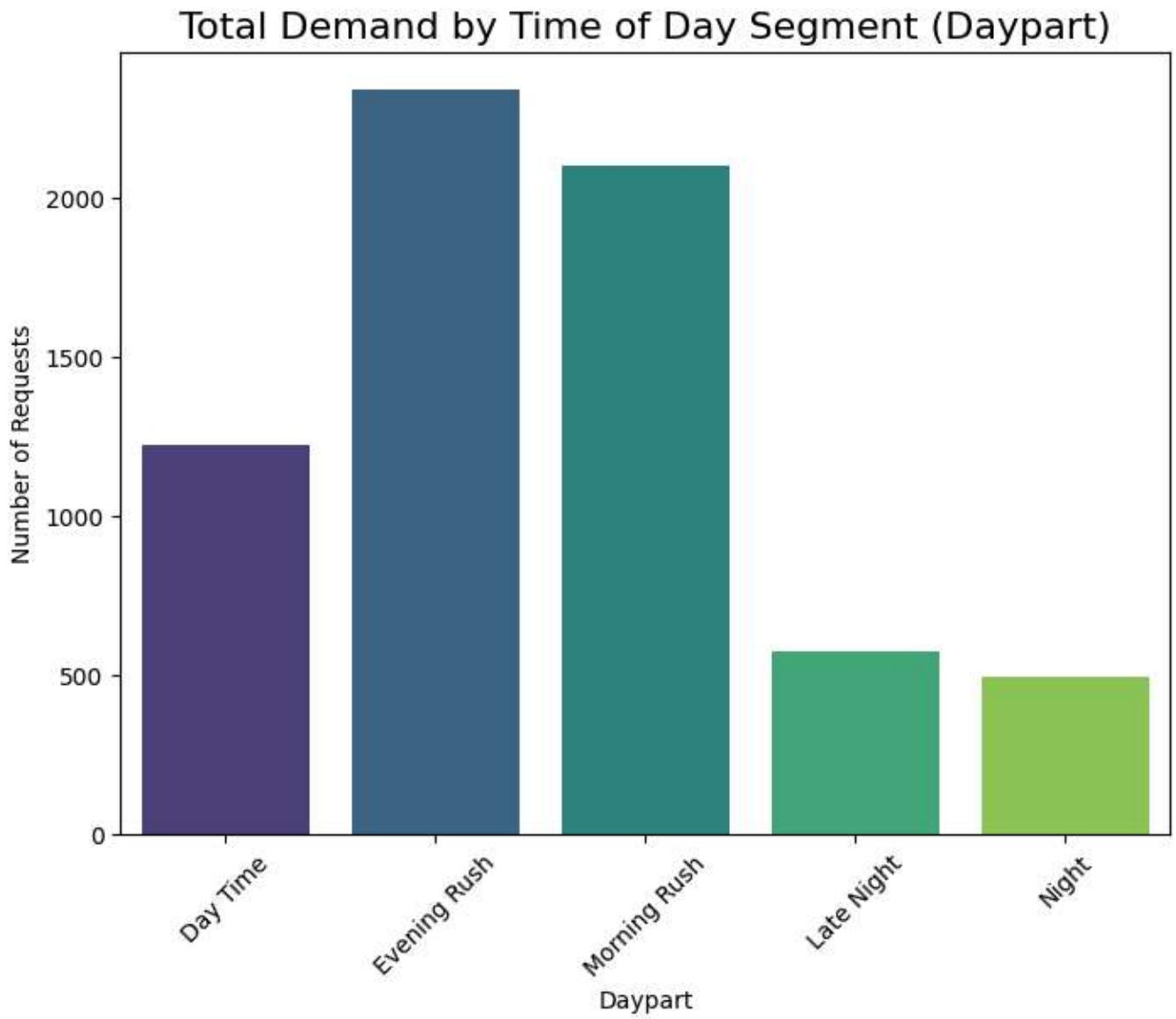
3. Will the gained insights help creating a positive business impact?

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

Chart - 3

```
# Chart - 3 visualization code

plt.figure(figsize=(8,6))
sns.countplot(x='Daypart', data=df, palette='viridis')
plt.title('Total Demand by Time of Day Segment (Daypart)', fontsize=16)
plt.xlabel('Daypart')
plt.ylabel('Number of Requests')
plt.xticks(rotation=45)
plt.show()
```

```
/tmp/ipython-input-82-1983945464.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.countplot(x='Daypart', data=df, palette='viridis')
```



1. Why did you pick the specific chart?

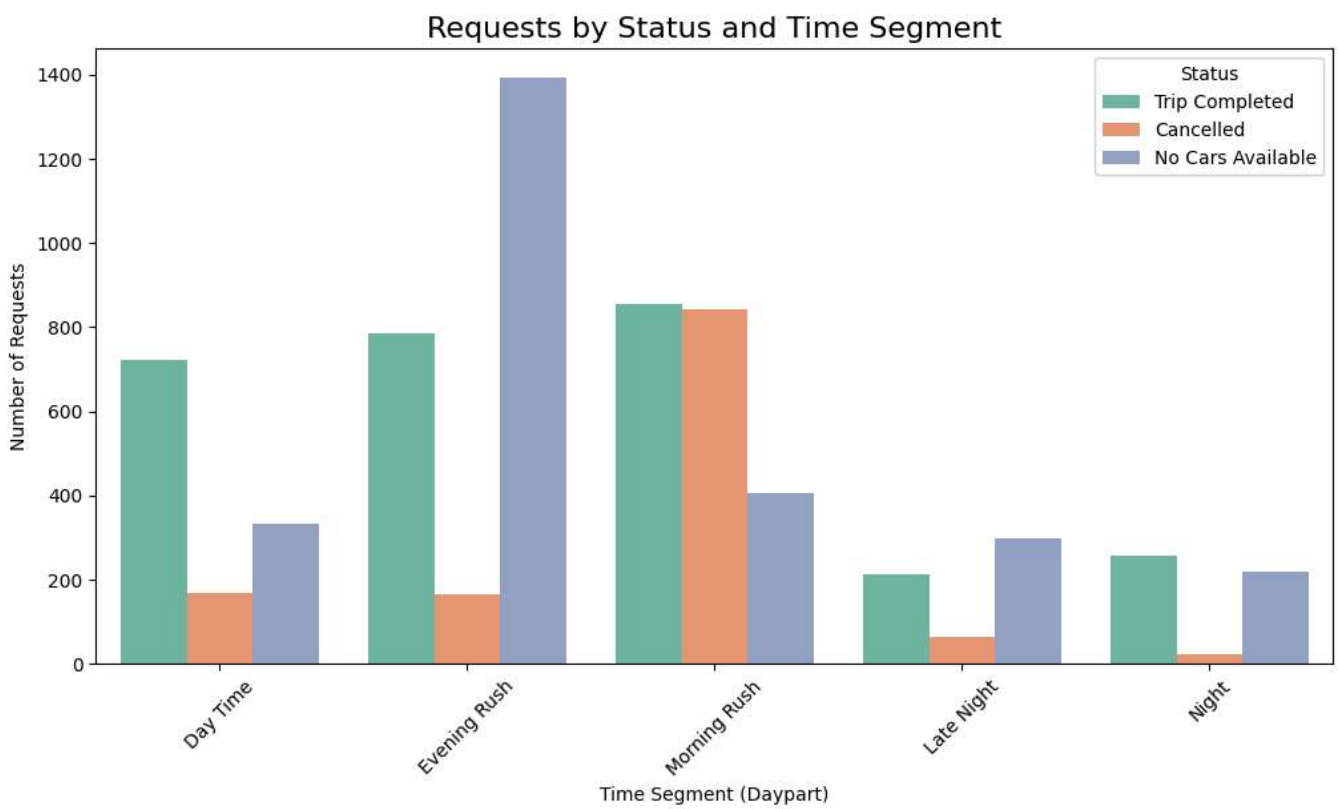
- It is ideal for showing the total volume of requests across different time segments of the day.
- Since Daypart is a categorical variable representing time segments (e.g. Morning, Afternoon, Evening, etc.), a single-variable countplot quickly shows which periods have higher or lower demand.
- This chart helps Uber identify peak demand periods easily.

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

Demand peaks during Morning and Evening Rush hours, while Late Night and Early Morning show lower demand but risk driver shortages. Uber can optimize driver allocation and offer incentives to balance supply with peak demand, improving service reliability.

Chart - 4

```
# Chart - 4 visualization code
plt.figure(figsize=(12, 6))
sns.countplot(x='Daypart', hue='Status', data=df, palette='Set2')
plt.title('Requests by Status and Time Segment', fontsize=16)
plt.xlabel('Time Segment (Daypart)')
plt.ylabel('Number of Requests')
plt.xticks(rotation=45)
plt.legend(title='Status')
plt.show()
```



1. Why did you pick the specific chart?

Answer Here.

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

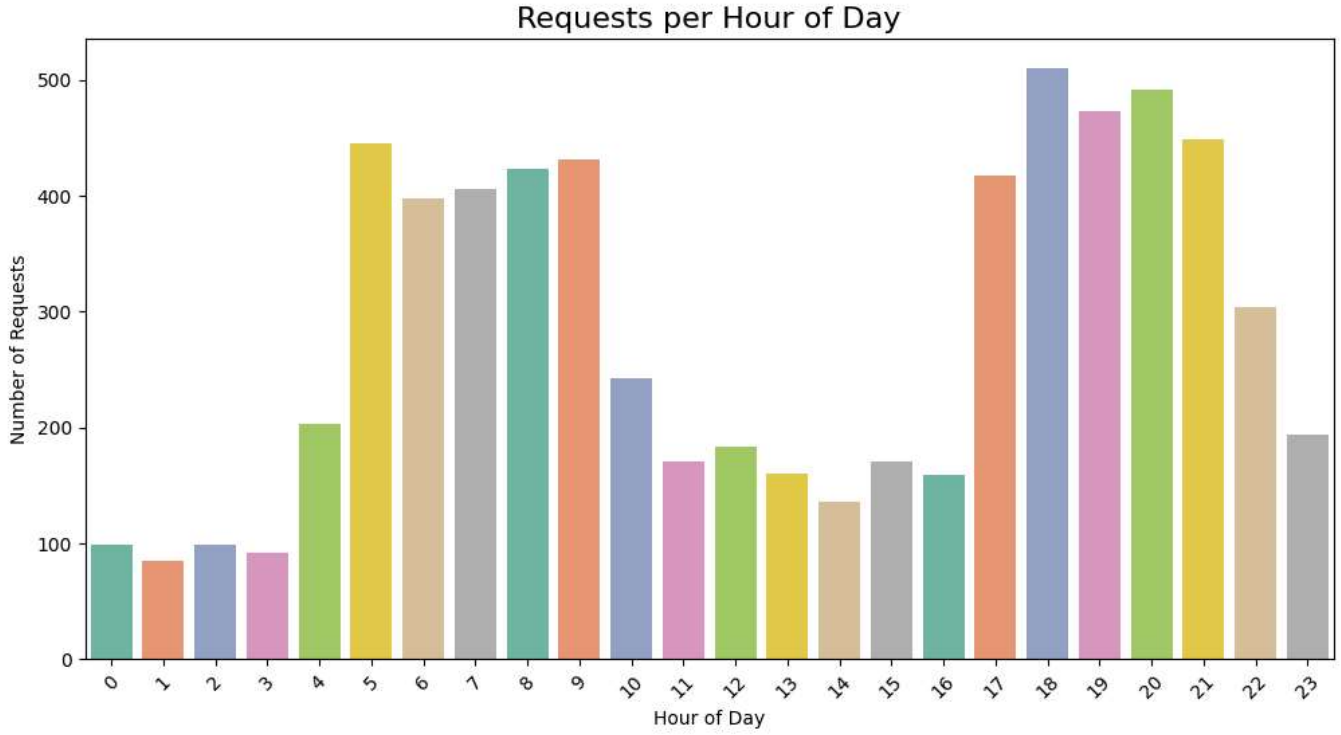
Answer Here

Chart - 5

```
# Convert timestamp to datetime if not already
df['Request timestamp'] = pd.to_datetime(df['Request timestamp'])
```

```
# Extract hour
df['Request Hour'] = df['Request timestamp'].dt.hour

# Plot
plt.figure(figsize=(12, 6))
sns.countplot(x='Request Hour', data=df, palette='Set2')
plt.title('Requests per Hour of Day', fontsize=16)
plt.xlabel('Hour of Day')
plt.ylabel('Number of Requests')
plt.xticks(rotation=45)
plt.show()
```

```
/tmp/ipython-input-84-2970832990.py:2: UserWarning: Could not infer format, so each elem
  df['Request timestamp'] = pd.to_datetime(df['Request timestamp'])
/tmp/ipython-input-84-2970832990.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.countplot(x='Request Hour', data=df, palette='Set2')
```



1. Why did you pick the specific chart?

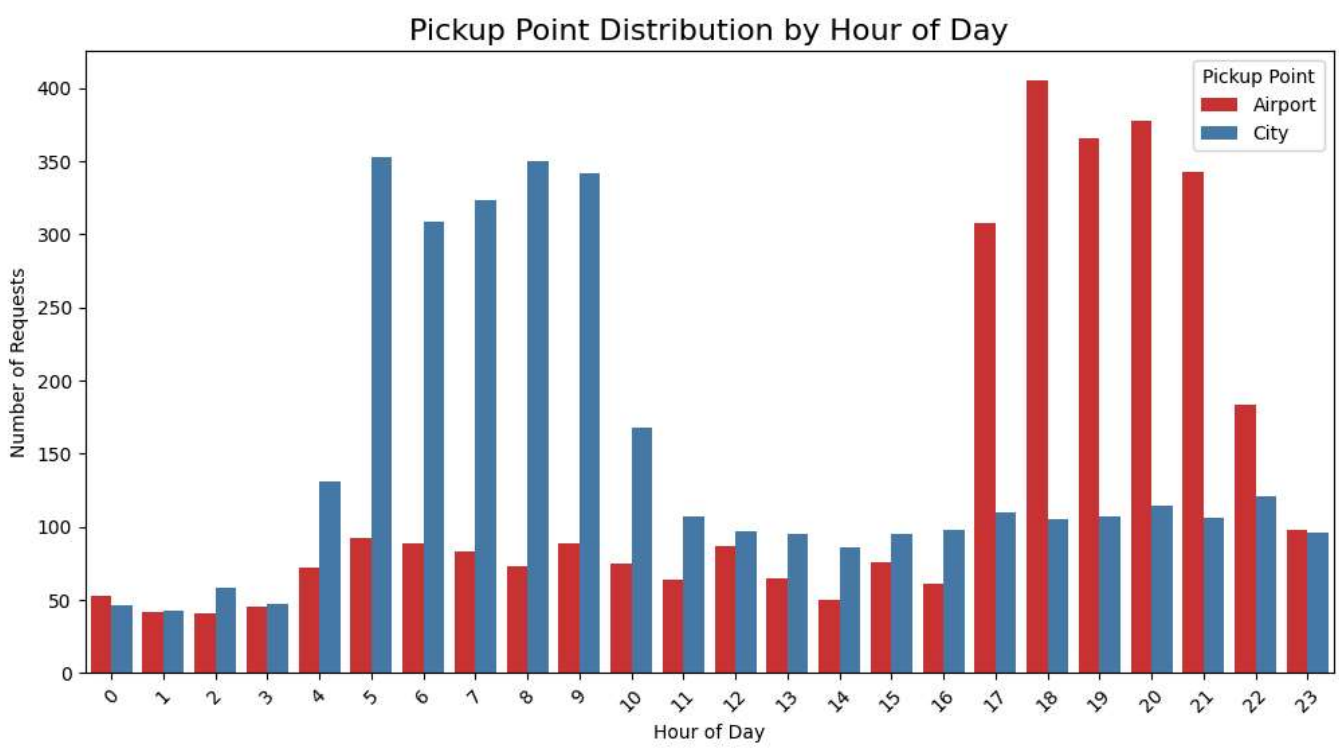
- It clearly compares trip outcomes (Status) across different time segments (Daypart).
- Grouped bar charts are effective to visualize distribution of success and failure rates by time of day.
- It helps to easily spot when cancellations or no cars are more frequent.

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

Answer Here

Chart - 6

```
# Chart - 6 visualization code
plt.figure(figsize=(12,6))
sns.countplot(x='Request Hour', hue='Pickup point', data=df, palette='Set1')
plt.title('Pickup Point Distribution by Hour of Day', fontsize=16)
plt.xlabel('Hour of Day')
plt.ylabel('Number of Requests')
plt.xticks(rotation=45)
plt.legend(title='Pickup Point')
plt.show()
```



1. Why did you pick the specific chart?

Answer Here.

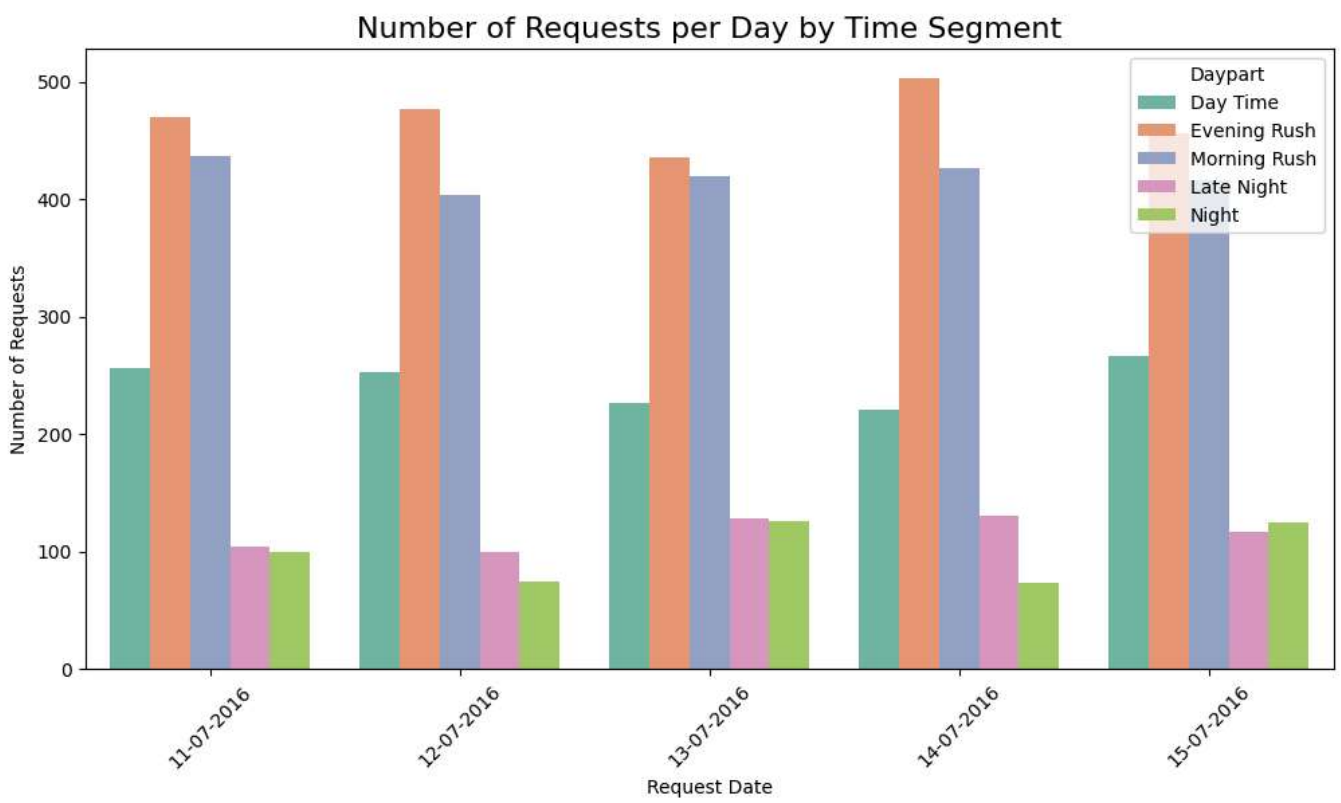
2. What is/are the insight(s) found from the chart?

Answer Here

Chart - 7

```
# Chart - 7 visualization code
plt.figure(figsize=(12,6))
sns.countplot(x='Request Date', hue='Daypart', data=df, palette='Set2')
plt.title('Number of Requests per Day by Time Segment', fontsize=16)
plt.xlabel('Request Date')
```

```
plt.ylabel('Number of Requests')
plt.xticks(rotation=45)
plt.legend(title='Daypart')
plt.show()
```



1. Why did you pick the specific chart?

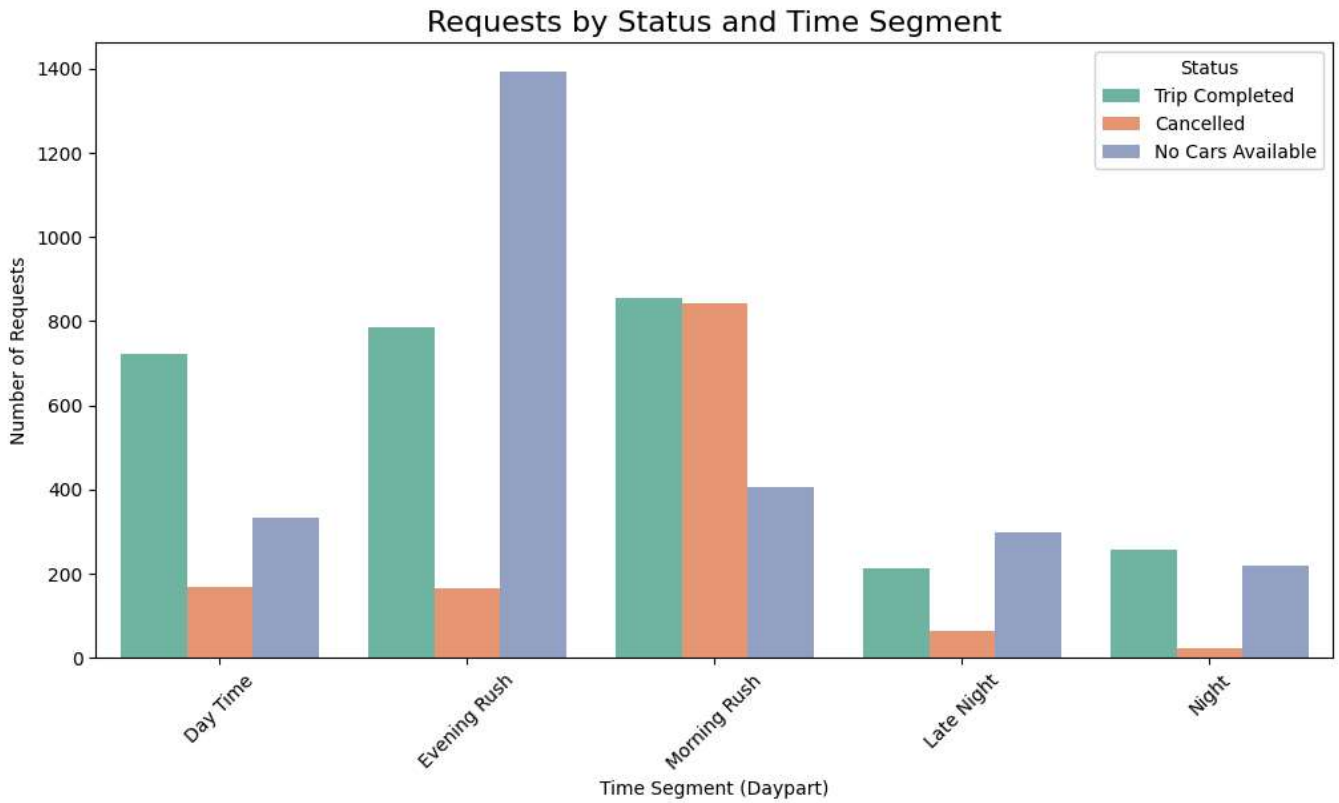
- It shows how pickup locations (City vs Airport) vary across different hours of the day.
- The hourly distribution helps identify peak hours for each pickup point.
- Grouped bar charts make it easy to compare multiple categories across continuous variables like hours.

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

- City pickups dominate during most hours of the day.
- Airport pickups spike during specific periods (often late night or early morning, depending on flight schedules).
- These patterns help Uber optimize driver allocation at both pickup points depending on time of day.Answer Here

Chart - 8

```
plt.figure(figsize=(12, 6))
sns.countplot(x='Daypart', hue='Status', data=df, palette='Set2')
plt.title('Requests by Status and Time Segment', fontsize=16)
plt.xlabel('Time Segment (Daypart)')
plt.ylabel('Number of Requests')
plt.xticks(rotation=45)
plt.legend(title='Status')
plt.show()
```



1. Why did you pick the specific chart?

- It helps visualize daily demand patterns broken down by time segments (Dayparts).
- Grouped bar charts allow easy comparison of how demand varies for each day across different times of day.
- This chart supports operational planning on a day-by-day basis.

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

- Some days show higher total demand compared to others.
- Morning and Evening Rush segments consistently have higher requests across multiple days.

- The chart highlights specific days with spikes in demand, which Uber can analyze for events, holidays, or external factors affecting demand.

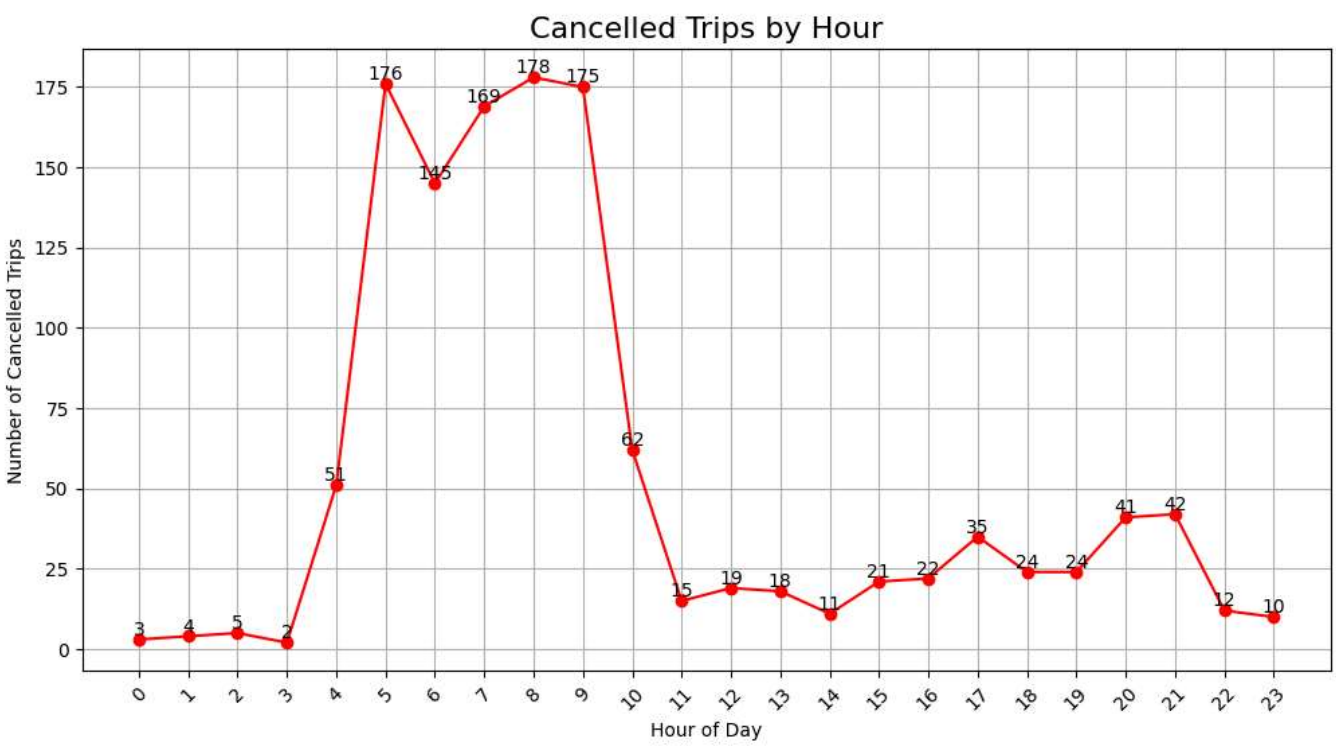
3. Will the gained insights help creating a positive business impact?

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

Answer Here

Chart - 9

```
# Chart - 9 visualization code

# timestamp is datetime
df['Request timestamp'] = pd.to_datetime(df['Request timestamp'])
df['Request Hour'] = df['Request timestamp'].dt.hour

# Filter for Cancelled trips
cancelled = df[df['Status'] == 'Cancelled']

# Group by Hour
cancelled_by_hour = cancelled.groupby('Request Hour').size()

# Plot
plt.figure(figsize=(12,6))
plt.plot(cancelled_by_hour.index, cancelled_by_hour.values, marker='o', linestyle='-', color
plt.title('Cancelled Trips by Hour', fontsize=16)
plt.xlabel('Hour of Day')
plt.ylabel('Number of Cancelled Trips')
plt.xticks(range(0, 24), rotation=45)

# Add data labels
for x, y in zip(cancelled_by_hour.index, cancelled_by_hour.values):
    plt.text(x, y, str(y), ha='center', va='bottom', fontsize=10)

plt.grid(True)
plt.show()
```



1. Why did you pick the specific chart?

I selected a line chart because it effectively shows how cancelled trips vary across different hours of the day. A line chart is suitable for representing continuous data over time (in this case, hours of the day), and makes it easy to observe trends, peaks, and drops in cancellations. The addition of data labels helps to quickly identify the exact number of cancellations at each hour.

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

- The number of cancellations is not uniform throughout the day.
- There are clear peaks in cancellations during the morning peak hours (e.g., 5 AM to 9 AM) and again during evening hours (around 5 PM to 9 PM), which likely coincide with office commute times.

- The cancellation rate is much lower during late night and early afternoon hours.
- This indicates that during peak demand hours, supply-demand mismatch or driver unavailability may be leading to a higher number of cancellations.

italicized text#### Chart - 10 - Correlation Heatmap