Problem Statement :

Uber has received some complaints from their customers facing problems related to ride cancellations by the driver and non-availability of cars for a specific route in the city.

The uneven supply-demand gap for cabs from City to Airport and vice-versa is causing a bad effect on customer relationships as well as Uber is losing out on its revenue.

The aim of analysis is to identify the root cause of the problem (i.e. cancellation and non-availability of cars) and recommend ways to tackle the situation.

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

```
df = pd.read_csv("/content/uber-data.csv")
```

```
df.head()
```

	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

```
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   object
 5   Drop timestamp     2831 non-null   object
dtypes: float64(1), int64(1), object(4)
memory usage: 316.3+ KB
```

```
df.isna().sum()
```

```
Request id              0
Pickup point            0
Driver id            2650
Status                  0
Request timestamp       0
Drop timestamp       3914
dtype: int64
```

```
df.isna().sum()*100/len(df)
```

```
Request id            0.000000
Pickup point          0.000000
Driver id            39.288362
Status                0.000000
Request timestamp     0.000000
Drop timestamp       58.028169
dtype: float64
```

```
df.shape
```

```
(6745, 6)
```

```
df["Request timestamp"] = pd.to_datetime(df["Request timestamp"])
```

```
df["Drop timestamp"] = pd.to_datetime(df["Drop timestamp"])
```

```
df.head()
```

	Request id	Pickup point	Driver id	Status	Request timestamp	Drop timestamp
0	619	Airport	1.0	Trip Completed	2016-11-07 11:51:00	2016-11-07 13:00:00
1	867	Airport	1.0	Trip Completed	2016-11-07 17:57:00	2016-11-07 18:47:00
2	1807	City	1.0	Trip Completed	2016-12-07 09:17:00	2016-12-07 09:58:00

```
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
```

```
df["Status"].value_counts()
```

```
Trip Completed       2831
No Cars Available    2650
Cancelled            1264
Name: Status, dtype: int64
```

```
df["RequestHour"] = df["Request timestamp"].dt.hour
```

```
df.head()
```

	Request id	Pickup point	Driver id	Status	Request timestamp	Drop timestamp	RequestHour
0	619	Airport	1.0	Trip Completed	2016-11-07 11:51:00	2016-11-07 13:00:00	11
1	867	Airport	1.0	Trip Completed	2016-11-07 17:57:00	2016-11-07 18:47:00	17
2	1807	City	1.0	Trip Completed	2016-12-07 09:17:00	2016-12-07 09:58:00	9

```
bin = [0, 4, 9, 16, 21, 23]
label = ["Dawn", "Early Morning", "Noon", "Late Evening", "Night"]

df["TimeSlot"] = pd.cut(df["RequestHour"] , bins = bin, labels = label)
df.head()
```

	Request id	Pickup point	Driver id	Status	Request timestamp	Drop timestamp	RequestHour	TimeSlot
0	619	Airport	1.0	Trip Completed	2016-11-07 11:51:00	2016-11-07 13:00:00	11	Noon
1	867	Airport	1.0	Trip Completed	2016-11-07 17:57:00	2016-11-07 18:47:00	17	Late Evening
2	1807	City	1.0	Trip Completed	2016-12-07 09:17:00	2016-12-07 09:58:00	9	Early Morning

```
df["TimeSlot"].value_counts()
```

```
Late Evening     2342
Early Morning    2103
Noon             1224
Night             498
Dawn              479
Name: TimeSlot, dtype: int64
```

```
df["TimeSlot"].value_counts(normalize = True)*100
```

```
Late Evening     35.239242
Early Morning    31.643094
Noon             18.417093
Night             7.493229
Dawn              7.207343
Name: TimeSlot, dtype: float64
```

```
df["Status"].value_counts(normalize = True)*100
```

```
Trip Completed       41.971831
No Cars Available    39.288362
Cancelled            18.739807
Name: Status, dtype: float64
```

```
# Distinguish the Supply-Demand Gap by a new variable Cab Availability where Supply is when Trip is Completed, all else is Demand -

df["Cab Availability"] = df["Status"].apply(lambda x: "Available" if x=="Trip Completed" else "Not Available")
```

```
df["Cab Availability"].value_counts(normalize =True)*100
```

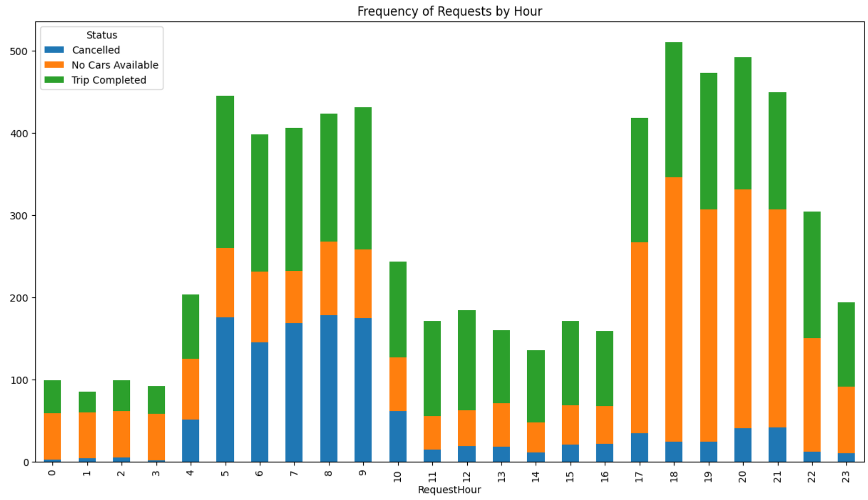
```
Not Available    58.028169
Available        41.971831
Name: Cab Availability, dtype: float64
```

```
#Frequency of Requests by Hour

# Frequency of Requests by Hour -

df.groupby(['RequestHour','Status']).size().unstack().plot(kind='bar', stacked=True, figsize=(15, 8))
plt.title('Frequency of Requests by Hour')
```

```
Text(0.5, 1.0, 'Frequency of Requests by Hour')
```




```
#RCA - insights
'''
1. Peaks are at 5-9 and 17-21 request hours
2. During the 5-9 peak hours: among the 2 problems: major problem is cab cancellations
3. During the 17-21 peak hours: among the 2 problems: major problem is unavailability of cab
'''
```

```
df["Pickup point"].value_counts(normalize = True)*100
```

```
City       51.99407
Airport    48.00593
Name: Pickup point, dtype: float64
```

```
sample = df[df["Cab Availability"] == "Not Available"]
```

```
sample["Pickup point"].value_counts(normalize = True)*100
```

```
City       51.175268
Airport    48.824732
Name: Pickup point, dtype: float64
```

```
sample["TimeSlot"].value_counts(normalize = True)*100
```

```
Late Evening     40.415045
Early Morning    32.399481
Noon             13.022049
Dawn              7.911803
Night             6.251621
Name: TimeSlot, dtype: float64
```

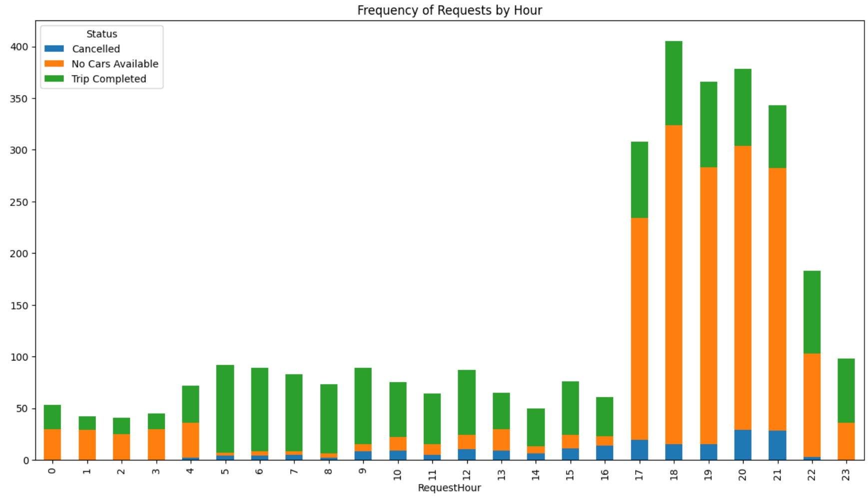
```
#Assumption:
'''
City -> Airport
Airport -> City
'''
```

```
'\nCity -> Airport\nAirport -> City\n'
```

```
#Suppy-Demand gap from Airport to City

df[df["Pickup point"] == "Airport"].groupby(['RequestHour','Status']).size().unstack().plot(kind='bar', stacked=True, figsize=(15, 8))
plt.title('Frequency of Requests by Hour')
```

```
Text(0.5, 1.0, 'Frequency of Requests by Hour')
```

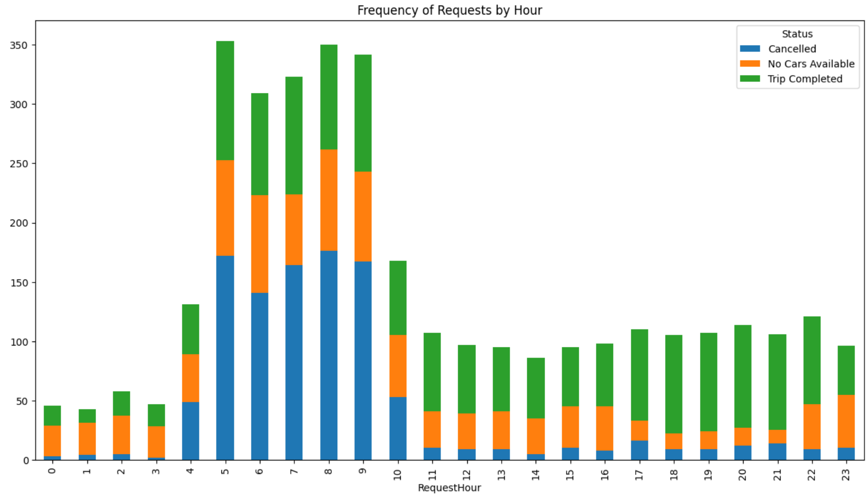



```
#obs: majority of the unavailable cabs issue is coming during the late evening speifically when pickup point is Airport
```

```
#Suppy-Demand gap from City to Airport

df[df["Pickup point"] == "City"].groupby(['RequestHour','Status']).size().unstack().plot(kind='bar', stacked=True, figsize=(15, 8))
plt.title('Frequency of Requests by Hour')
```

```
Text(0.5, 1.0, 'Frequency of Requests by Hour')
```




```
#obs: majority of the cancellations are coming early morning spcifically when the pickup point is City

'''

Direct vs Root causes:

Direct Causes                                                             Root Causes
Immediate factors that directly contribute to a problem or an event.     Underlying factors that give rise to the direct causes.
Evident and observable.                                                 Often hidden or not immediately apparent.
Addressing direct causes can resolve the immediate problem.             Addressing root causes prevents the problem from recurring
Focuses on symptoms and visible effects.                                 Focuses on the fundamental reasons behind the symptoms.
Usually associated with short-term impact.                              Often associated with long-term impact.
'''
```

```
#Competitor Analysis:

Amazon    Flipkart:

1. Market Presence:
2. User Experience
3. Pricing and Offers
4. Delivery and Logistics
5. Mobile Apps
6. Loyalty Programs
7. Marketing and Advertising



'''

HW - Uber vs Ola

1. Market Presence
2. Service Range
3. Pricing and Offers
4. Customer Base
5. User Experience
6. Ride Options and Vehicle Types
7. Safety Measures
8. Driver Incentives
9. Loyalty Programs
10. Marketing and Advertising
```