uber_casestudy

January 12, 2025

[144]:
```
"""
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
"""
```

[144]:
```
'\nProblem Statement :\nUber has received some complaints from their customers
facing problems related to ride cancellations by the driver \nand non-
availability of cars for a specific route in the city.\n\nThe uneven supply-
demand gap for cabs from City to Airport and vice-versa is causing a bad effect
on \ncustomer relationships as well as Uber is losing out on its revenue.\n\nThe
aim of analysis is to identify the root cause of the problem (i.e. cancellation
and non-availability of cars) \nand recommend ways to tackle the situation\n'
```

[145]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[146]:
```
import warnings
warnings.simplefilter('ignore')
sns.set_style('whitegrid')
```

[147]:
```
#importing the data
df=pd.read_csv(r"C:\Users\samvj\Downloads\uber-data.
 ↪csv",dayfirst=True,na_values='NA')
df.head()
```

[147]:
```
   Request id Pickup point  Driver id          Status    Request timestamp  \
0         619      Airport        1.0  Trip Completed      11/7/2016 11:51
1         867      Airport        1.0  Trip Completed      11/7/2016 17:57
2        1807         City        1.0  Trip Completed       12/7/2016 9:17
3        2532      Airport        1.0  Trip Completed      12/7/2016 21:08
4        3112         City        1.0  Trip Completed  13-07-2016 08:33:16

         Drop timestamp
0       11/7/2016 13:00
1       11/7/2016 18:47
2        12/7/2016 9:58
3       12/7/2016 22:03
4  13-07-2016 09:25:47
```

[148]:
```
print("number of rows:{}".format(df.shape[0]))
print("number of rows:{}".format(df.shape[1]))
```

```
number of rows:6745
number of rows:6
```

[149]:
```
#checking the data info
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
```

[150]:
```
df["Driver id"].nunique()
```

[150]:
```
300
```

[151]:
```
df.head()
```

[151]:
```
   Request id Pickup point  Driver id          Status    Request timestamp  \
0         619      Airport        1.0  Trip Completed      11/7/2016 11:51
1         867      Airport        1.0  Trip Completed      11/7/2016 17:57
2        1807         City        1.0  Trip Completed       12/7/2016 9:17
```

```
3         2532      Airport         1.0  Trip Completed      12/7/2016 21:08
4         3112         City         1.0  Trip Completed  13-07-2016 08:33:16

         Drop timestamp
0       11/7/2016 13:00
1       11/7/2016 18:47
2        12/7/2016 9:58
3       12/7/2016 22:03
4  13-07-2016 09:25:47
```

[152]:
```
#changing the date format into the desired format
df['Request timestamp_1'] = pd.to_datetime(df['Request timestamp'],␣
 ↪format='%d-%m-%Y %H:%M:%S', errors='coerce')
df['Request timestamp_2']=pd.to_datetime(df['Request timestamp'], format='%d/%m/
 ↪%Y %H:%M', errors='coerce')
df['Request timestamp']=df['Request timestamp_2'].combine_first(df['Request␣
 ↪timestamp_1'])
```

[153]:
```
df['Drop timestamp_1'] = pd.to_datetime(df['Drop timestamp'], format='%d-%m-%Y␣
 ↪%H:%M:%S', errors='coerce')
df['Drop timestamp_2']=pd.to_datetime(df['Drop timestamp'], format='%d/%m/%Y %H:
 ↪%M', errors='coerce')
df['Drop timestamp']=df['Drop timestamp_2'].combine_first(df['Drop␣
 ↪timestamp_1'])
```

[154]:
```
df.head()
```

[154]:
```
   Request id Pickup point  Driver id          Status   Request timestamp  \
0         619      Airport        1.0  Trip Completed 2016-07-11 11:51:00
1         867      Airport        1.0  Trip Completed 2016-07-11 17:57:00
2        1807         City        1.0  Trip Completed 2016-07-12 09:17:00
3        2532      Airport        1.0  Trip Completed 2016-07-12 21:08:00
4        3112         City        1.0  Trip Completed 2016-07-13 08:33:16

       Drop timestamp Request timestamp_1 Request timestamp_2  \
0 2016-07-11 13:00:00                 NaT 2016-07-11 11:51:00
1 2016-07-11 18:47:00                 NaT 2016-07-11 17:57:00
2 2016-07-12 09:58:00                 NaT 2016-07-12 09:17:00
3 2016-07-12 22:03:00                 NaT 2016-07-12 21:08:00
4 2016-07-13 09:25:47 2016-07-13 08:33:16                 NaT

     Drop timestamp_1    Drop timestamp_2
0                NaT 2016-07-11 13:00:00
1                NaT 2016-07-11 18:47:00
2                NaT 2016-07-12 09:58:00
3                NaT 2016-07-12 22:03:00
4 2016-07-13 09:25:47                 NaT
```

[155]:
```
df.drop(columns=['Request timestamp_1', 'Request timestamp_2',
                 'Drop timestamp_1', 'Drop timestamp_2'],
                  inplace=True)
```

[156]:
```
df.head()
```

[156]:
```
   Request id Pickup point  Driver id          Status   Request timestamp  \
0         619      Airport        1.0  Trip Completed 2016-07-11 11:51:00
1         867      Airport        1.0  Trip Completed 2016-07-11 17:57:00
2        1807         City        1.0  Trip Completed 2016-07-12 09:17:00
3        2532      Airport        1.0  Trip Completed 2016-07-12 21:08:00
4        3112         City        1.0  Trip Completed 2016-07-13 08:33:16

       Drop timestamp
0 2016-07-11 13:00:00
1 2016-07-11 18:47:00
2 2016-07-12 09:58:00
3 2016-07-12 22:03:00
4 2016-07-13 09:25:47
```

[157]:
```
#checking for null values
df.isnull().sum()/len(df)*100
```

[157]:
```
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
```

[158]:
```
"""
here we focus on the drop timestamp
we assume that the drop timestamp is more than 50% is due to the cancellations␣
 ↪of trip by the drivers
by taking consideration on the request timestamp
"""
```

[158]:
```
' \nhere we focus on the drop timestamp\nwe assume that the drop timestamp is
more than 50% is due to the cancellations of trip by the drivers\nby taking
consideration on the request timestamp\n'
```

[159]:
```
#Feature engineering
df["RequestHour"]= df["Request timestamp"].dt.hour
```

[160]:
```
df.head()
```

[160]:
```
   Request id Pickup point  Driver id          Status   Request timestamp  \
0         619      Airport        1.0  Trip Completed 2016-07-11 11:51:00
1         867      Airport        1.0  Trip Completed 2016-07-11 17:57:00
2        1807         City        1.0  Trip Completed 2016-07-12 09:17:00
3        2532      Airport        1.0  Trip Completed 2016-07-12 21:08:00
4        3112         City        1.0  Trip Completed 2016-07-13 08:33:16

       Drop timestamp  RequestHour
0 2016-07-11 13:00:00           11
1 2016-07-11 18:47:00           17
2 2016-07-12 09:58:00            9
3 2016-07-12 22:03:00           21
4 2016-07-13 09:25:47            8
```

[161]:
```
#categorising the features for further analysis
df["TimeSlot"] = np.where(df["RequestHour"] <= 4, "Dawn",
                  np.where(df["RequestHour"] <= 9, "Early Morning",
                  np.where(df["RequestHour"] <= 16, "Noon",
                  np.where(df["RequestHour"] <= 21, "Late Evening", "Night"))))
```

[162]:
```
df.head()
```

[162]:
```
   Request id Pickup point  Driver id          Status   Request timestamp  \
0         619      Airport        1.0  Trip Completed 2016-07-11 11:51:00
1         867      Airport        1.0  Trip Completed 2016-07-11 17:57:00
2        1807         City        1.0  Trip Completed 2016-07-12 09:17:00
3        2532      Airport        1.0  Trip Completed 2016-07-12 21:08:00
4        3112         City        1.0  Trip Completed 2016-07-13 08:33:16

       Drop timestamp  RequestHour       TimeSlot
0 2016-07-11 13:00:00           11           Noon
1 2016-07-11 18:47:00           17   Late Evening
2 2016-07-12 09:58:00            9  Early Morning
3 2016-07-12 22:03:00           21   Late Evening
4 2016-07-13 09:25:47            8  Early Morning
```

[163]:
```
df["Status"].unique()
```

[163]:
```
array(['Trip Completed', 'Cancelled', 'No Cars Available'], dtype=object)
```

[164]:
```
df["cab Availability"] = np.where(df["Status"] == 'Trip␣
 ↪Completed','Available','Not Available')
```

[165]:
```
df.head()
```

[165]:
```
   Request id Pickup point  Driver id          Status   Request timestamp  \
0         619      Airport        1.0  Trip Completed 2016-07-11 11:51:00
```

```
1          867      Airport         1.0  Trip Completed 2016-07-11 17:57:00
2         1807         City         1.0  Trip Completed 2016-07-12 09:17:00
3         2532      Airport         1.0  Trip Completed 2016-07-12 21:08:00
4         3112         City         1.0  Trip Completed 2016-07-13 08:33:16
```

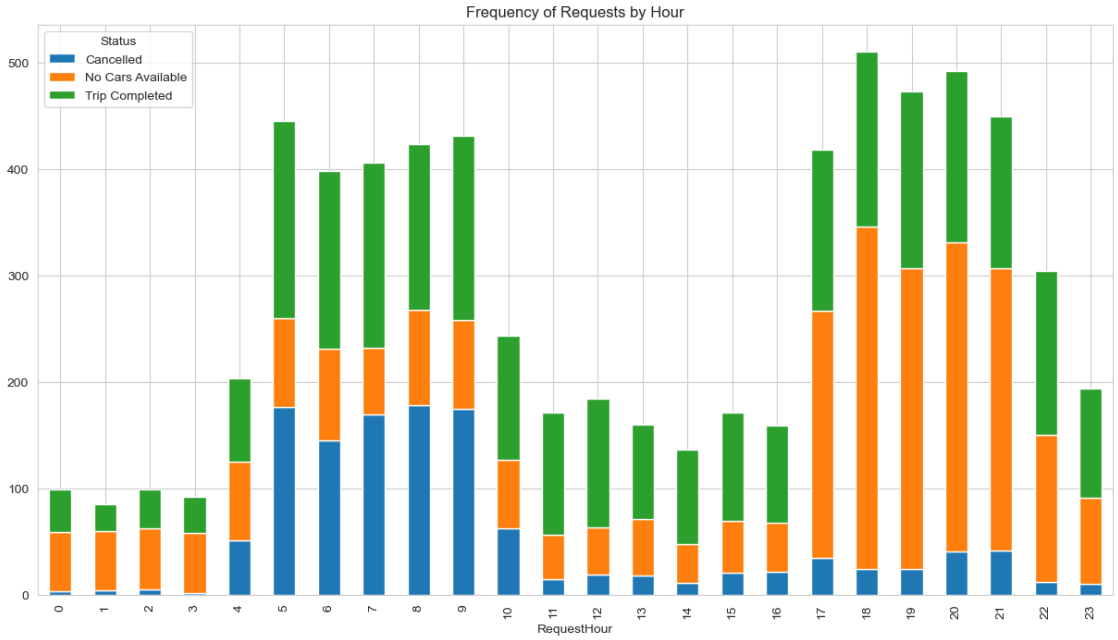
	Drop timestamp	RequestHour	TimeSlot	cab Availability
0	2016-07-11 13:00:00	11	Noon	Available
1	2016-07-11 18:47:00	17	Late Evening	Available
2	2016-07-12 09:58:00	9	Early Morning	Available
3	2016-07-12 22:03:00	21	Late Evening	Available
4	2016-07-13 09:25:47	8	Early Morning	Available

[166]:
```
df['cab Availability'].value_counts(normalize=True)*100
```

[166]:
```
cab Availability
Not Available    58.028169
Available        41.971831
Name: proportion, dtype: float64
```

[167]:
```
# status of cab availability during different time zone
df.groupby(['RequestHour','Status']).size().unstack().
  ↪plot(kind='bar',stacked=True,figsize=(15,8))
plt.title("Frequency of Requests by Hour")
```

[167]:
```
Text(0.5, 1.0, 'Frequency of Requests by Hour')
```



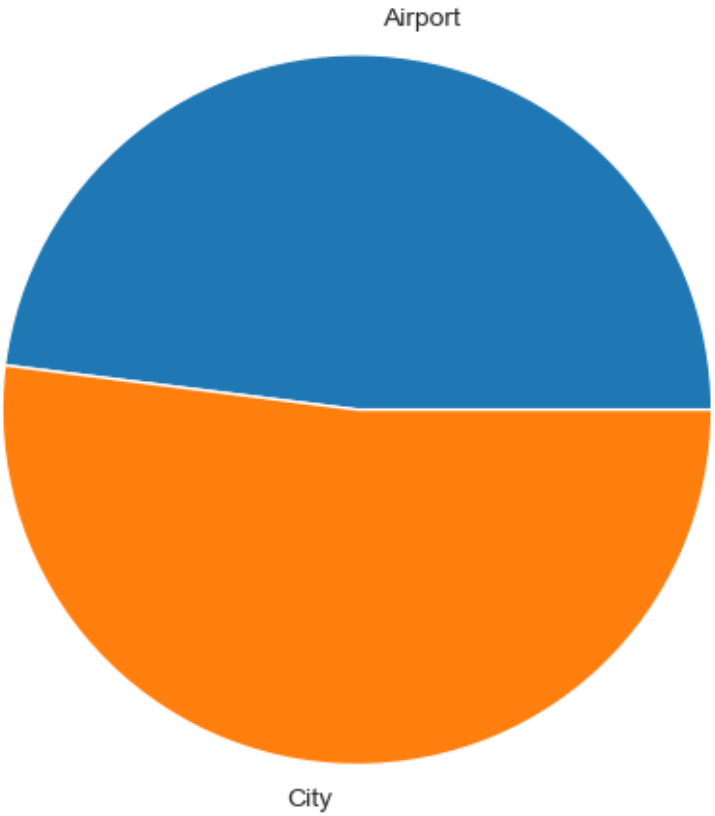
[168]:
```
"""
(5-9)-peak of cancellation (morning)
(17-21) -peak of cab not available (evening 5-9)

"""
```

[168]:
```
'\n(5-9)-peak of cancellation (morning)\n(17-21) -peak of cab not available
(evening 5-9)\n\n'
```

[169]:
```
# comparing the no of cab available at both pickup point
df.groupby(['Pickup point']).size().plot(kind="pie", stacked=True, figsize=(6,
 ↪6), table=True)
plt.ylabel("")
```

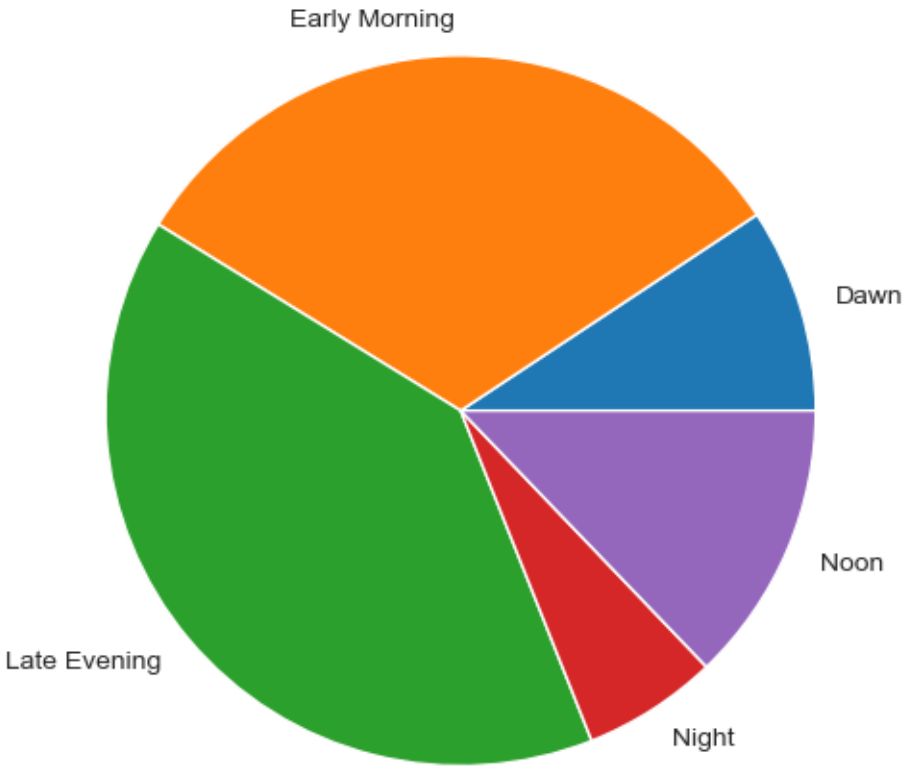
[169]:
```
Text(0, 0.5, '')
```



Airport	City
3238	3507

[170]:
```
#availability of cabs in different timeslot
df[(df["cab Availability"] == "Not Available")].groupby(["TimeSlot"]).size().
 ↪plot(kind="pie", stacked = True , figsize = (6,6), table = True)
plt.ylabel("")
```

[170]:
```
Text(0, 0.5, '')
```



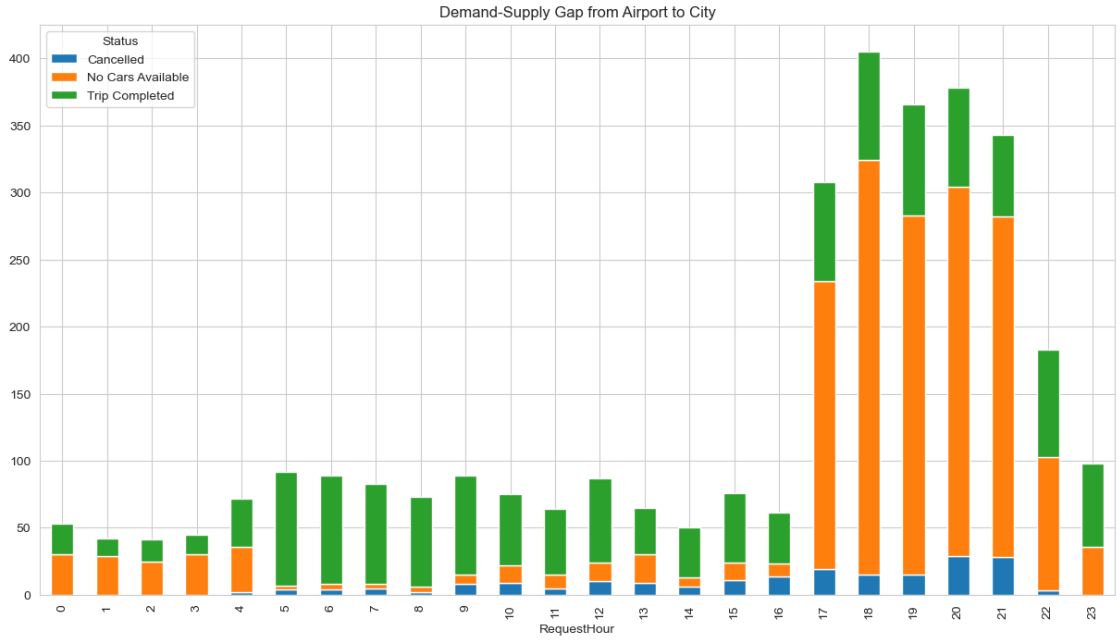
Dawn	Early Morning	Late Evening	Night	Noon
364	1249	1558	241	502

[171]:
```
"""
Observation:
    late evenings and early mornings are not recommended for the airport␣
 ↪transportation and vice_versa
"""
```

[171]:
```
'\nObservation:\n    late evenings and early mornings are not recommended for
the airport transportation and vice_versa\n'
```

[172]:
```
#Analysing the data of Airport pickup point
df[(df["Pickup point"] == "Airport")].groupby(['RequestHour','Status']).size().
 ↪unstack().plot(kind="bar", stacked = True, figsize = (15, 8))
plt.title("Demand-Supply Gap from Airport to City")
```

[172]: Text(0.5, 1.0, 'Demand-Supply Gap from Airport to City')

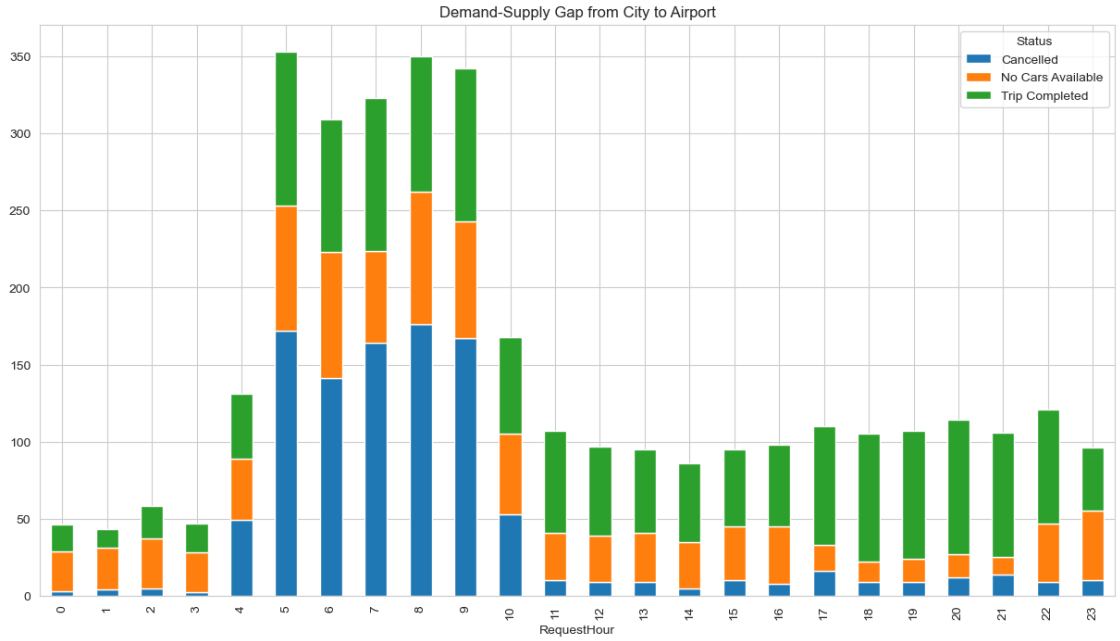


[173]:
```
"""
observation
    There ia a high demand for cabs during the time frame of 5.00 PM to 9.00 PM
    But he supply is very less due to no cabs availability
"""
```

[173]: '\nobservation\n   There ia a high demand for cabs during the time frame of 5.00 PM to 9.00 PM\n   But he supply is very less due to no cabs availability\n'

[174]:
```
#Analysing the data of City pickup point
df[(df["Pickup point"] == "City")].groupby(['RequestHour','Status']).size().
 ↪unstack().plot(kind="bar", stacked = True, figsize = (15, 8))
plt.title("Demand-Supply Gap from City to Airport")
```

[174]: Text(0.5, 1.0, 'Demand-Supply Gap from City to Airport')

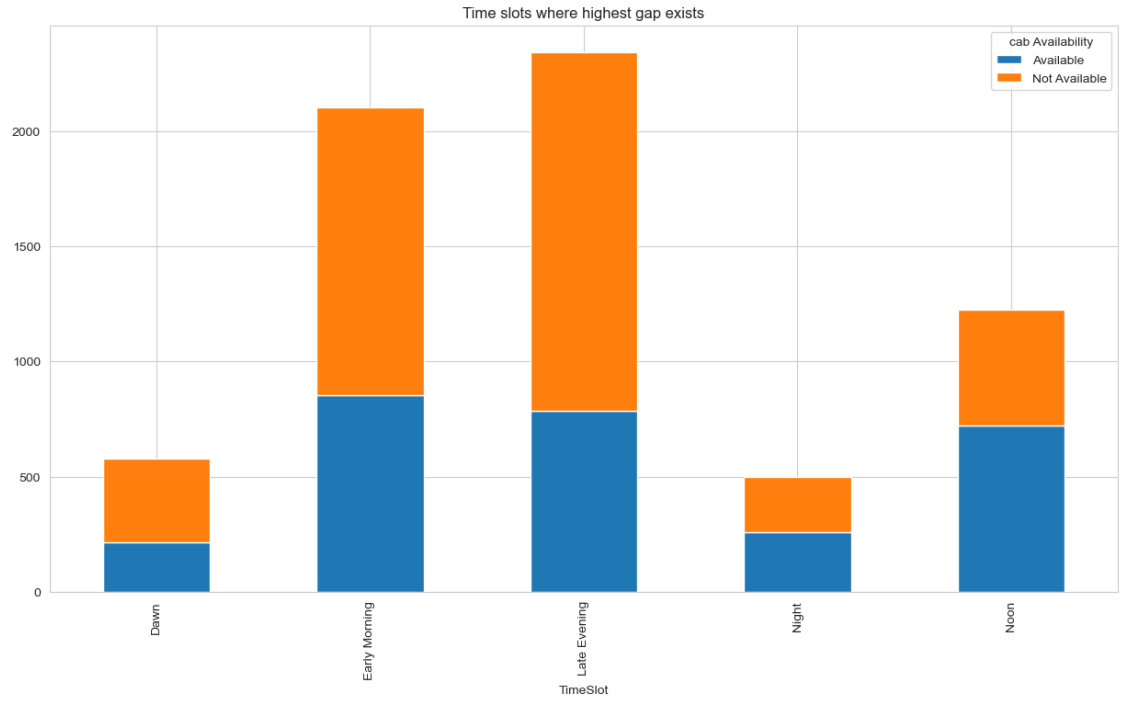


[175]:
```
"""
observation:
    There is very high demand for cabs from City to Airport between 5:00 AM - 9:
 ↪00 AM
    But the supply is very less primarily due to Ride Cancellations
"""
```

[175]:
```
'\nobservation:\n    There is very high demand for cabs from City to Airport
between 5:00 AM - 9:00 AM\n    But the supply is very less primarily due to Ride
Cancellations\n'
```

[176]:
```
# Analysing the data of different timeslot for car availability
df.groupby(["TimeSlot","cab Availability"]).size().unstack().plot(kind="bar",␣
 ↪stacked = True ,figsize=(15,8))
plt.title('Time slots where highest gap exists')
```

[176]:
```
Text(0.5, 1.0, 'Time slots where highest gap exists')
```



[177]:
```
"""
observation:
    Among the assumed time slots, we can see that the Late Evening and Early␣
 ↪Morning time slots has got the highest gap.
    This means that during evening & morning hours the probability of getting a␣
 ↪cab is very less
    """
```

[177]: '\nobservation:\n   Among the assumed time slots, we can see that the Late Evening and Early Morning time slots has got the highest gap.\n   This means that during evening & morning hours the probability of getting a cab is very less\n   '

[178]:
```
"""
Reason for the supply demand Gap:
    In the Supply-Demand graph from Airport to City, between 5:00 PM to 9:00 PM␣
 ↪there is very high demand for
        cabs because the supply is very low due to 'No Cars Available'
    The 'No Cars Available' is due to the fact that in the previous hours fewer␣
 ↪people travelled from City -
        Airport and so fewer cars are available in near Airpor
    Likewise, in Supply-Demand graph from City - Airport, between 5:00 AM to 9:
 ↪00 AM, there is very high
        demand for cabs because the supply is very low due to Ride Cancellations
```

```
    This is because there were fewer trips to Airport that completed in the␣
 ↪previous hours, so now the cabs have to come from a
        ong distance (City) to pickup the passenger and then they have to wait␣
 ↪for the passenger's arrival, so the drivers cancel the trip
"""
```

[178]:
```
"\nReason for the supply demand Gap:\n    In the Supply-Demand graph from
Airport to City, between 5:00 PM to 9:00 PM there is very high demand for\n
cabs because the supply is very low due to 'No Cars Available'\n    The 'No Cars
Available' is due to the fact that in the previous hours fewer people travelled
from City - \n        Airport and so fewer cars are available in near Airpor\n
Likewise, in Supply-Demand graph from City - Airport, between 5:00 AM to 9:00
AM, there is very high \n        demand for cabs because the supply is very low
due to Ride Cancellations\n    This is because there were fewer trips to Airport
that completed in the previous hours, so now the cabs have to come from a \n
ong distance (City) to pickup the passenger and then they have to wait for the
passenger's arrival, so the drivers cancel the trip\n"
```

[179]:
```
"""
Recommendations:
    Awarding Incentive for drivers who are waiting at the airport.
    Have a fleet stationed at the airport/ also can be done for part time␣
 ↪drivers.
    Provide free parking at airport.
    Refreshment booth for driver waiting at the airport.
    priority for High fare rides or long-distance rides.
    Drivers could be compensated for taking the night shifts hence covering the␣
 ↪00:00 - 5:00 time slot.
    Streaks maintained for consistent performance of such rides andbonus issued
"""
```

[179]:
```
'\nRecommendations:\n    Awarding Incentive for drivers who are waiting at the
airport.\n    Have a fleet stationed at the airport/ also can be done for part
time drivers.\n    Provide free parking at airport.\n    Refreshment booth for
driver waiting at the airport.\n    priority for High fare rides or long-
distance rides.\n    Drivers could be compensated for taking the night shifts
hence covering the 00:00 - 5:00 time slot.\n    Streaks maintained for
consistent performance of such rides andbonus issued\n'
```