SPADE DS Internship Assignment-1

In [ ]:
```
##Importing Libraries
```

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:
```
#Importing dataset
```

In [2]:
```
df = pd.read_csv("Cab X Request Data.csv")
```

In [3]:
```
df.head(10)
```

Out[3]:

	Request id	Pickup point	Driver id	Status	Request timestamp	Drop timestamp
0	619	Airport	1.0	Trip Completed	11-07-2016 11:51	11-07-2016 13:00
1	867	Airport	1.0	Trip Completed	11-07-2016 17:57	11-07-2016 18:47
2	1807	City	1.0	Trip Completed	12-07-2016 09:17	12-07-2016 09:58
3	2532	Airport	1.0	Trip Completed	12-07-2016 21:08	12-07-2016 22:03
4	3112	City	1.0	Trip Completed	13-07-2016 08:33	13-07-2016 09:25
5	3879	Airport	1.0	Trip Completed	13-07-2016 21:57	13-07-2016 22:28
6	4270	Airport	1.0	Trip Completed	14-07-2016 06:15	14-07-2016 07:13
7	5510	Airport	1.0	Trip Completed	15-07-2016 05:11	15-07-2016 06:07
8	6248	City	1.0	Trip Completed	15-07-2016 17:57	15-07-2016 18:50
9	267	City	2.0	Trip Completed	11-07-2016 06:46	11-07-2016 07:25

In [4]:
```
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
```

In [5]:
```
df.describe(include="all")
```

Out[5]:

	Request id	Pickup point	Driver id	Status	Request timestamp	Drop timestamp
count	6745.000000	6745	4095.000000	6745	6745	2831
unique	NaN	2	NaN	3	4016	2282
top	NaN	City	NaN	Trip Completed	15-07-2016 19:19	13-07-2016 08:53
freq	NaN	3507	NaN	2831	8	5
mean	3384.644922	NaN	149.501343	NaN	NaN	NaN
std	1955.099667	NaN	86.051994	NaN	NaN	NaN
min	1.000000	NaN	1.000000	NaN	NaN	NaN
25%	1691.000000	NaN	75.000000	NaN	NaN	NaN
50%	3387.000000	NaN	149.000000	NaN	NaN	NaN
75%	5080.000000	NaN	224.000000	NaN	NaN	NaN
max	6766.000000	NaN	300.000000	NaN	NaN	NaN

In [6]:
```
df['Request timestamp'] = df['Request timestamp'].astype('datetime64[ns]')
df['Drop timestamp'] = df['Drop timestamp'].astype('datetime64[ns]')
```

In [7]:
```
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 [8]:
```
df['Request timestamp'].head()
```

Out[8]:
```
0   2016-11-07 11:51:00
1   2016-11-07 17:57:00
2   2016-12-07 09:17:00
3   2016-12-07 21:08:00
4   2016-07-13 08:33:00
Name: Request timestamp, dtype: datetime64[ns]
```

In [ ]:
```
#Function to Create Booking_hour Feature using Request timestamp feature
```

In [9]:
```
Booking_hour = list()
for _ in df['Request timestamp']:
    h = _.hour
    Booking_hour.append(h)

df['Booking_hour']=Booking_hour
```

In [10]:
```
df.head(15)
```

Out[10]:

	Request id	Pickup point	Driver id	Status	Request timestamp	Drop timestamp	Booking_hour
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
3	2532	Airport	1.0	Trip Completed	2016-12-07 21:08:00	2016-12-07 22:03:00	21
4	3112	City	1.0	Trip Completed	2016-07-13 08:33:00	2016-07-13 09:25:00	8
5	3879	Airport	1.0	Trip Completed	2016-07-13 21:57:00	2016-07-13 22:28:00	21
6	4270	Airport	1.0	Trip Completed	2016-07-14 06:15:00	2016-07-14 07:13:00	6
7	5510	Airport	1.0	Trip Completed	2016-07-15 05:11:00	2016-07-15 06:07:00	5
8	6248	City	1.0	Trip Completed	2016-07-15 17:57:00	2016-07-15 18:50:00	17
9	267	City	2.0	Trip Completed	2016-11-07 06:46:00	2016-11-07 07:25:00	6
10	1467	Airport	2.0	Trip Completed	2016-12-07 05:08:00	2016-12-07 06:02:00	5
11	1983	City	2.0	Trip Completed	2016-12-07 12:30:00	2016-12-07 12:57:00	12
12	2784	Airport	2.0	Trip Completed	2016-07-13 04:49:00	2016-07-13 05:23:00	4
13	3075	City	2.0	Trip Completed	2016-07-13 08:02:00	2016-07-13 09:16:00	8
14	3379	City	2.0	Trip Completed	2016-07-13 14:23:00	2016-07-13 15:35:00	14

In [11]:
```
df['Booking_hour'].describe()
```

Out[11]:
```
count    6745.000000
mean       12.956709
std         6.504052
min         0.000000
25%         7.000000
50%        13.000000
75%        19.000000
max        23.000000
Name: Booking_hour, dtype: float64
```

In [ ]:
```
#Function to divide Booking hour into timeslots
```

In [12]:
```
time_slot = list()
for i in df['Booking_hour']:
    if 0<=i<=3:
        time_slot.append("Mid_night")
    elif 4<=i<=6:
        time_slot.append("Early_morning")
    elif 7<=i<=11:
        time_slot.append("Morning")
    elif 12<=i<=15:
        time_slot.append("AfterNoon")
    elif 16<=i<=19:
        time_slot.append("Evening")
    else:
        time_slot.append("Night")

df['time_slot']=time_slot
```

In [13]:
```
df.head(15)
```

Out[13]:

	Request id	Pickup point	Driver id	Status	Request timestamp	Drop timestamp	Booking_hour	time_slot
0	619	Airport	1.0	Trip Completed	2016-11-07 11:51:00	2016-11-07 13:00:00	11	Morning
1	867	Airport	1.0	Trip Completed	2016-11-07 17:57:00	2016-11-07 18:47:00	17	Evening
2	1807	City	1.0	Trip Completed	2016-12-07 09:17:00	2016-12-07 09:58:00	9	Morning
3	2532	Airport	1.0	Trip Completed	2016-12-07 21:08:00	2016-12-07 22:03:00	21	Night
4	3112	City	1.0	Trip Completed	2016-07-13 08:33:00	2016-07-13 09:25:00	8	Morning
5	3879	Airport	1.0	Trip Completed	2016-07-13 21:57:00	2016-07-13 22:28:00	21	Night
6	4270	Airport	1.0	Trip Completed	2016-07-14 06:15:00	2016-07-14 07:13:00	6	Early_morning
7	5510	Airport	1.0	Trip Completed	2016-07-15 05:11:00	2016-07-15 06:07:00	5	Early_morning
8	6248	City	1.0	Trip Completed	2016-07-15 17:57:00	2016-07-15 18:50:00	17	Evening
9	267	City	2.0	Trip Completed	2016-11-07 06:46:00	2016-11-07 07:25:00	6	Early_morning
10	1467	Airport	2.0	Trip Completed	2016-12-07 05:08:00	2016-12-07 06:02:00	5	Early_morning
11	1983	City	2.0	Trip Completed	2016-12-07 12:30:00	2016-12-07 12:57:00	12	AfterNoon
12	2784	Airport	2.0	Trip Completed	2016-07-13 04:49:00	2016-07-13 05:23:00	4	Early_morning
13	3075	City	2.0	Trip Completed	2016-07-13 08:02:00	2016-07-13 09:16:00	8	Morning
14	3379	City	2.0	Trip Completed	2016-07-13 14:23:00	2016-07-13 15:35:00	14	AfterNoon

In [14]:
```
df.columns
```

Out[14]:
```
Index(['Request id', 'Pickup point', 'Driver id', 'Status',
       'Request timestamp', 'Drop timestamp', 'Booking_hour', 'time_slot'],
      dtype='object')
```

In [ ]:
```
#Dropping the useless columns for analysis
```

In [15]:
```
unused = ['Request id', 'Driver id','Request timestamp', 'Drop timestamp']
df.drop(unused,axis=1,inplace=True)
```

In [16]:
```
df.head(10)
```

Out[16]:

	Pickup point	Status	Booking_hour	time_slot
0	Airport	Trip Completed	11	Morning
1	Airport	Trip Completed	17	Evening
2	City	Trip Completed	9	Morning
3	Airport	Trip Completed	21	Night
4	City	Trip Completed	8	Morning
5	Airport	Trip Completed	21	Night
6	Airport	Trip Completed	6	Early_morning
7	Airport	Trip Completed	5	Early_morning
8	City	Trip Completed	17	Evening
9	City	Trip Completed	6	Early_morning

In [ ]:
```
#Function to create a supplied feature using Status Feature
```

In [17]:
```
supplied = list()
for _ in df['Status']:
    if _ == 'Trip Completed':
        supplied.append('Yes')
    else:
        supplied.append("No")

df['supplied'] = supplied
```

In [18]:
```
df.head(10)
```

Out[18]:

	Pickup point	Status	Booking_hour	time_slot	supplied
0	Airport	Trip Completed	11	Morning	Yes
1	Airport	Trip Completed	17	Evening	Yes
2	City	Trip Completed	9	Morning	Yes
3	Airport	Trip Completed	21	Night	Yes
4	City	Trip Completed	8	Morning	Yes
5	Airport	Trip Completed	21	Night	Yes
6	Airport	Trip Completed	6	Early_morning	Yes
7	Airport	Trip Completed	5	Early_morning	Yes
8	City	Trip Completed	17	Evening	Yes
9	City	Trip Completed	6	Early_morning	Yes

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6745 entries, 0 to 6744
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Pickup point  6745 non-null   object
 1   Status        6745 non-null   object
 2   Booking_hour  6745 non-null   int64 
 3   time_slot     6745 non-null   object
 4   supplied      6745 non-null   object
dtypes: int64(1), object(4)
memory usage: 263.6+ KB
```

In [20]:
```
df.describe(include="all")
```

Out[20]:

	Pickup point	Status	Booking_hour	time_slot	supplied
count	6745	6745	6745.000000	6745	6745
unique	2	3	NaN	6	2
top	City	Trip Completed	NaN	Morning	No
freq	3507	2831	NaN	1674	3914
mean	NaN	NaN	12.956709	NaN	NaN
std	NaN	NaN	6.504052	NaN	NaN
min	NaN	NaN	0.000000	NaN	NaN
25%	NaN	NaN	7.000000	NaN	NaN
50%	NaN	NaN	13.000000	NaN	NaN
75%	NaN	NaN	19.000000	NaN	NaN
max	NaN	NaN	23.000000	NaN	NaN

In [ ]:
```
#Data Visualization and Analysis.
```

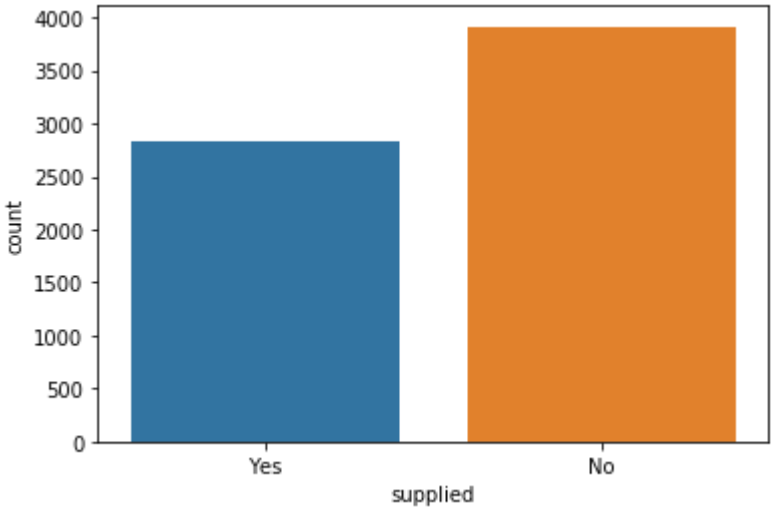
1. Visually identify the most pressing problems for X Company.

Hint: Create plots to visualize the frequency of requests that get cancelled or show 'no cars available; identify the most problematic types of requests (city to airport / airport to city etc.) and the time slots (early mornings, late evenings etc.) using plots

In [ ]:
```
#supplied vs Total NO of Bookings
```

In [21]:
```
sns.countplot(x='supplied',data=df)
```

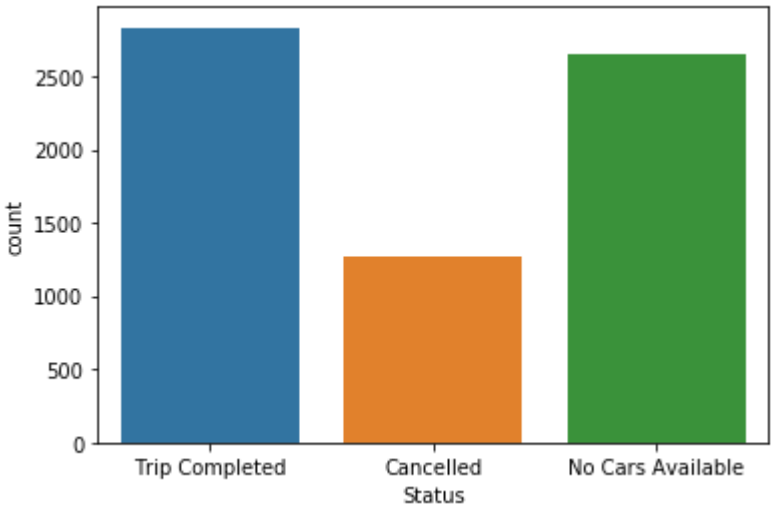
Out[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ce1e5e208>
```



In [ ]:
```
#Status vs Number of Bookings
```

In [22]:
```
sns.countplot(x='Status', data=df)
```

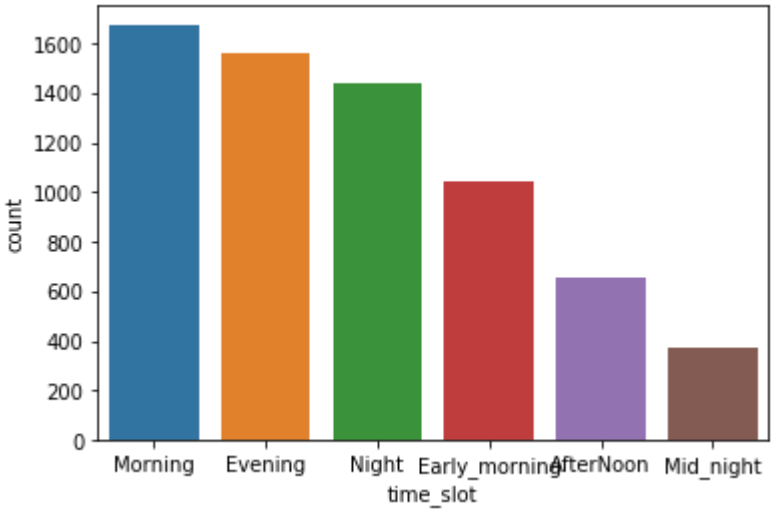
Out[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ce1f88d48>
```



In [ ]:
```
#time slot vs No of Bookings
```

In [23]:
```
sns.countplot(x='time_slot', data =df)
```

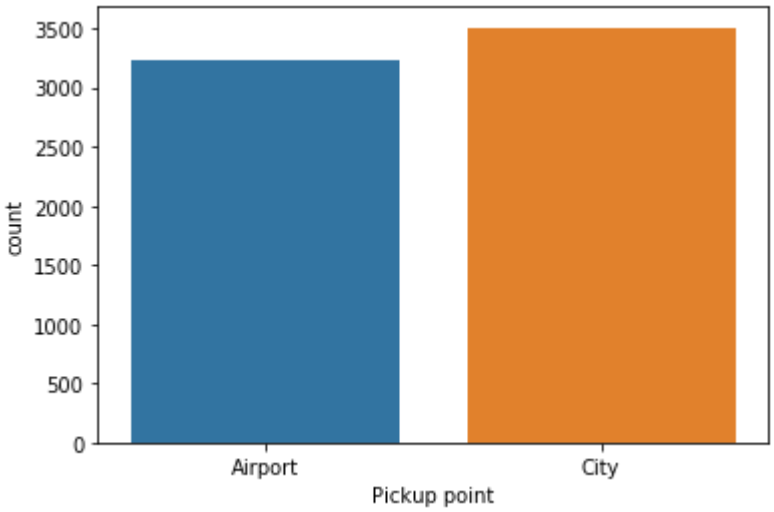
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x1ce2003848>



In [ ]:
```
#Pickup point vs No of Bookings
```

In [24]:
```
sns.countplot(x='Pickup point',data=df)
```

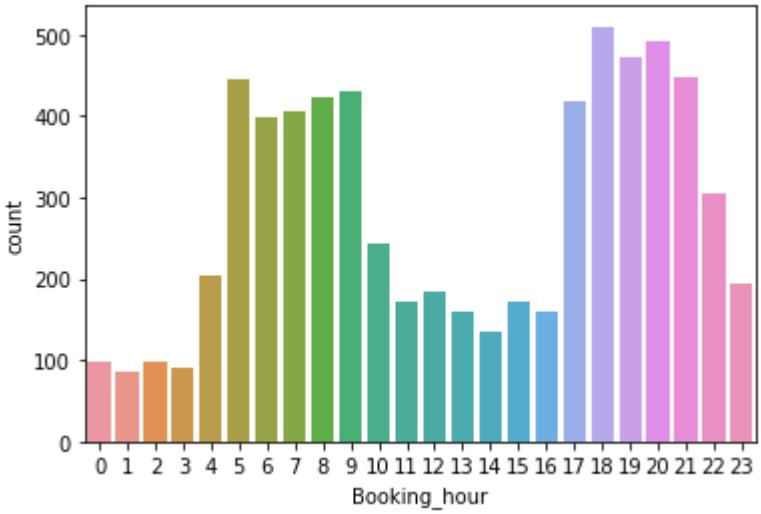
Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x1ce1fb2888>



In [ ]:
```
#Booking hour vs No of bookings
```

In [25]:
```
sns.countplot(x='Booking_hour',data=df)
```

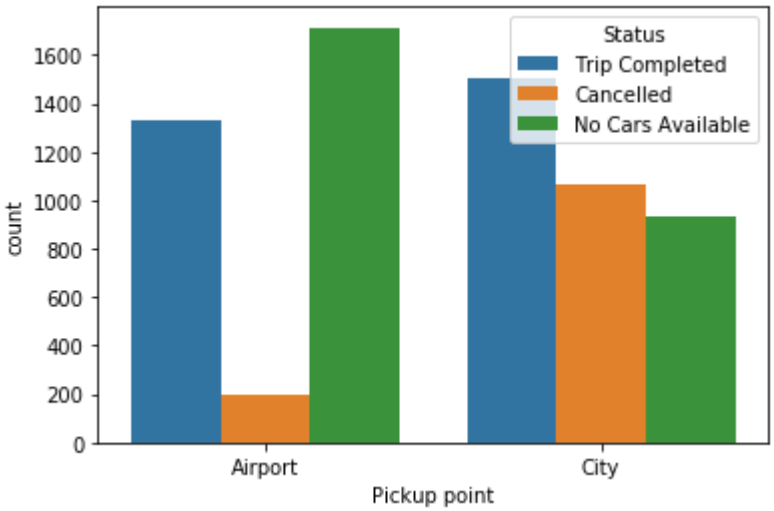
Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x1ce20f3188>



In [ ]:
```
#Pickup point & Status VS No of Bookings
```

In [26]:
```
sns.countplot(x='Pickup point',hue = 'Status',data=df)
```

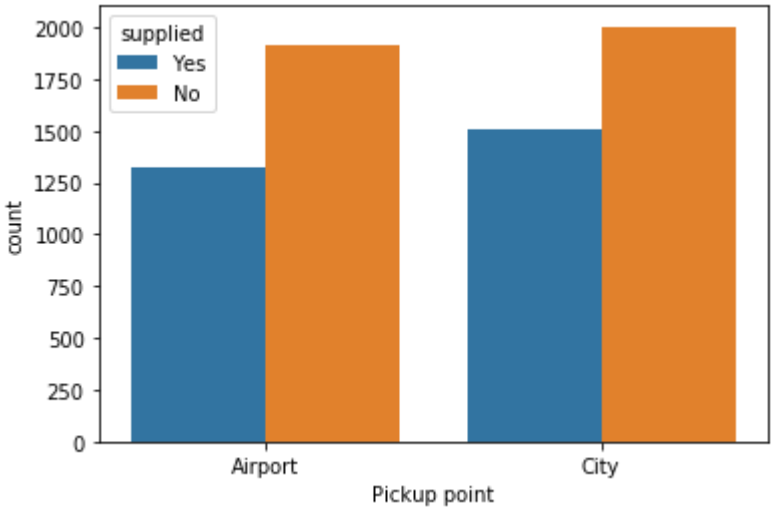
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x1ce21bf108>



In [ ]:
```
#Pickup point & Supplied VS No of Bookings
```

In [27]:
```
sns.countplot(x='Pickup point',hue = 'supplied',data=df)
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ce222d408>
```



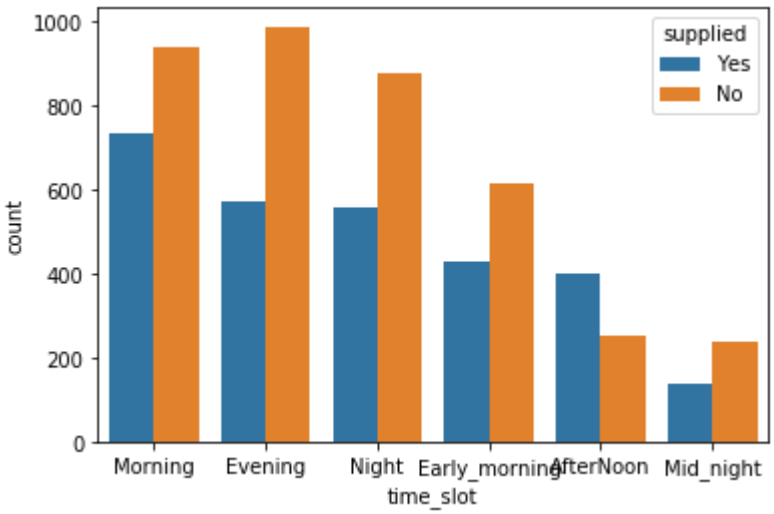
2. Find out the gap between supply and demand and show the same using plots.

Find the time slots when the highest gap exists

In [ ]:
```
#Time slot & Supplied VS No of Bookings
```

In [28]:
```
sns.countplot(x='time_slot',hue = 'supplied',data=df)
```

Out[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ce229cb08>
```

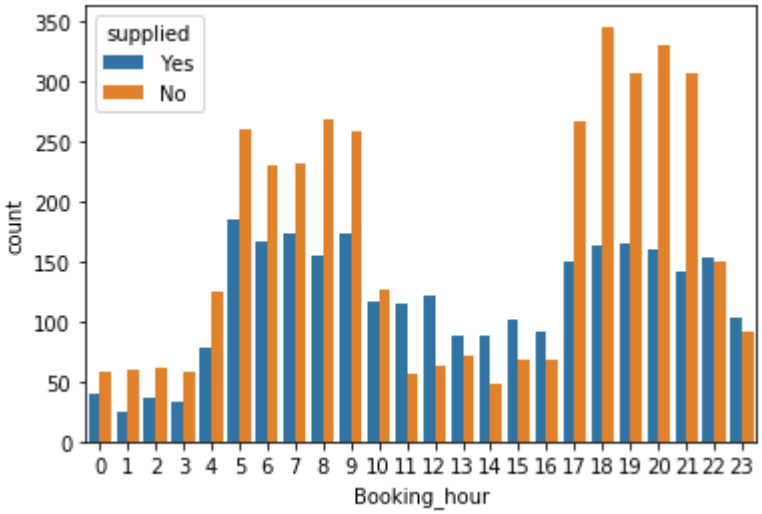


Supply demand Gap is there in each and every time slot.Top slots where the gap is more are Moring,Evening and Night Slots.

In [ ]:
```
#Booking Hour & Supplied VS No of Bookings
```

In [29]:
```
sns.countplot(x='Booking_hour',hue = 'supplied',data=df)
```

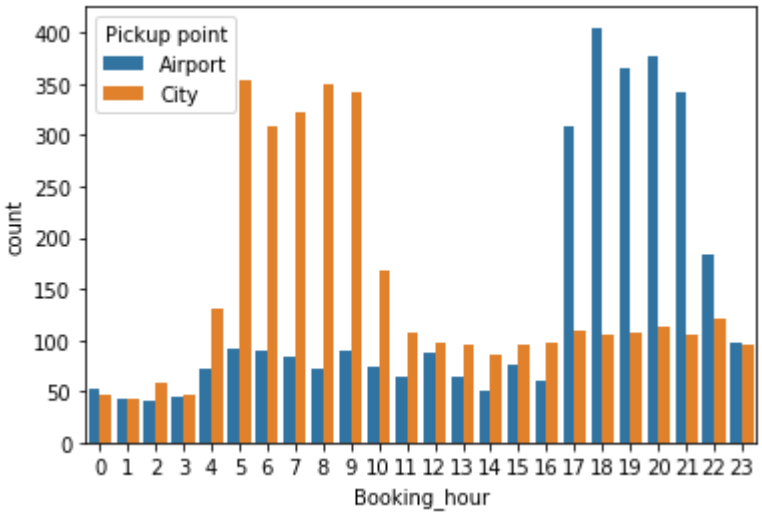
Out[29]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ce233b1c8>
```



In [ ]:
```
#Booking Hour & pickup point VS No of Bookings
```

In [30]:
```
sns.countplot(x='Booking_hour',hue = 'Pickup point',data=df)
```

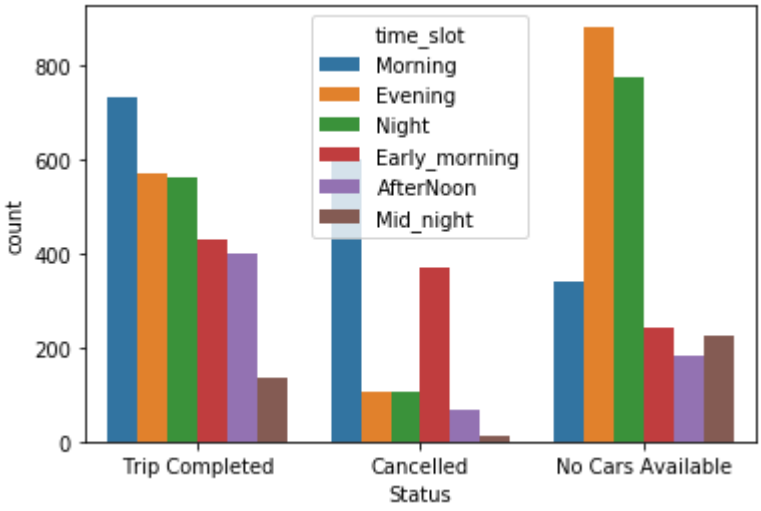
Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ce24653c8>
```



In [ ]:
```
#Status & timeslot VS No of Bookings
```

In [31]:
```
sns.countplot(x='Status',hue = 'time_slot',data=df)
```

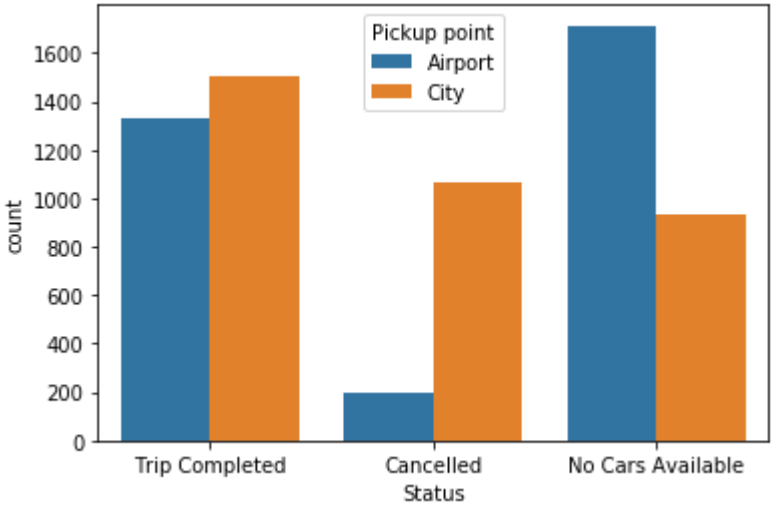
Out[31]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ce256fb48>
```



In [ ]:
```
#Status  & Pickup point VS No of Bookings
```

In [32]:
```
sns.countplot(x='Status',hue = 'Pickup point',data=df)
```

Out[32]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ce25f9fc8>
```

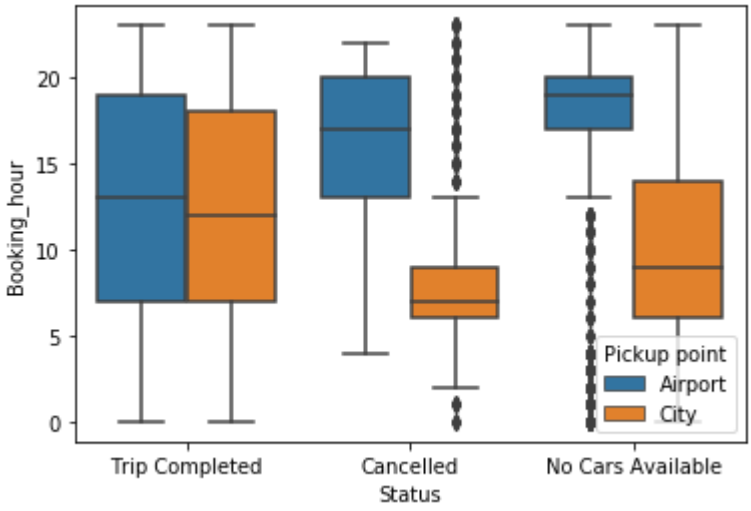


Find the types of requests (city-airport or airport-city) for which the gap is the most severe in the identified time slots

In [ ]:
```
#Box plot showing realtion between 'Status' , 'Booking hour','Pickup point'
```

In [33]:
```
sns.boxplot(x='Status',y='Booking_hour',hue='Pickup point',data=df)
```

Out[33]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ce2679d48>
```



City analysis :

1. Requests made by customers are cancelled at City in between 6 to 9 (Morining Slot) and Cars are not avilable from 6 to 14 (Morning and Afternoon slots)

Airport Analysis: ¶

1. Requests made by customers are cancelled at Airport in between 13 to 19 (AfterNoon and Evening Slots) and Cars are not avilable from 17 to 19 (Evening slot)

3. What do you think is the reason for this issue for the supply-demand gap? Write theanswer in less than 100 words. You may accompany the write-up with plot(s).

1. Accoring to me the reason might be not enough cars to service the requests as cars might not be available at the airport due to the cars serving inside the city and viceversa.
2. Accoring to me the reason for requests being cancelled by the drivers might be the morning rush 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.

4.Recommend some ways to resolve the supply-demand gap.

1. Increasing the number of cabs
2. The Company X can plan to give more profits to drivers who drive the cab for longer distance form airport to city and viceversa in the slots where there is gap between supply and demand is more.
3. .They can have a permanent stand at the airports and make sure the cabs are available round the clock.