In [ ]:
```
#Import libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
import numpy as np
```

In [2]:
```
#Write code to load the dataset here
df = pd.read_csv("UberRequestData.csv")
```

In [3]:
```
#use the following few lines to explore the data or clean it
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
```

Q1. Null Values

Which columns have null values and how many?

a) Request id - 2650 Driver id -3914

b) Request id- 3914 Driver id - 2650

c) Driver id - 2650 Drop time stamp -3914

d) There are no null values in the dataset.

In [4]:
```
#We simply use isnull().sum() on whole df
print (df.isnull().sum())
```

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

Q2. Choose Correct

Select the correct statements from the options given below. (More than one may be correct)

a) We need to delete the records containing the null values as they won't be useful for our analysis.

b) Only the records containing null values in the Driver id need to be eliminated.

c) There are a total of 1264 requests which have a status of "cancelled"

d) We need to keep the records containing null values for both the columns and using them analyse the problems faced by Uber.

In [5]:
```
# value_counts() on Status
df['Status'].value_counts()
```

Out[5]:
```
Trip Completed       2831
No Cars Available    2650
Cancelled            1264
Name: Status, dtype: int64
```

Q3. Busy Hour

Now that you have understood the significance of the null values in this case study, the next step would involve fixing the Request Time-Stamp and Drop-Time Stamp columns. Go ahead and convert it into a date-time object. After that report back the hour in which the most number of requests are being made.
a) 18:00-19:00

b) 16:00-17:00

c) 14:00-15:00

d) 13:00-14:00

e) 19:00-20:00

In [7]:
```
#write your solution here

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

In [9]:
```
df['Request hour']=df['Request timestamp'].apply(lambda x: x.hour)
df['Request hour'].value_counts()
```

Out[9]:
```
18    510
20    492
19    473
21    449
5     445
9     431
8     423
17    418
7     406
6     398
22    304
10    243
4     203
23    194
12    184
11    171
15    171
13    160
16    159
14    136
0      99
2      99
3      92
1      85
Name: Request hour, dtype: int64
```

a) 18:00-19:00 hour has most requests

Q4. Cancelled Dates

Next, extract the date from the request timestamp and find the date when the most number of trips were cancelled.

a) 2016-07-13

b) 2016-11-07

c) 2016-07-14

d) 2016-07-15

e) 2016-12-07

In [10]:
```
#write your code here
df['Request date']=df['Request timestamp'].apply(lambda x: x.date())
df[df['Status']=='Cancelled']['Request date'].value_counts()
```

Out[10]:
```
2016-07-13    270
2016-11-07    262
2016-07-14    252
2016-12-07    240
2016-07-15    240
Name: Request date, dtype: int64
```

The highest number of cancel1ation is on 2016-07-13

Correct Statements II

Choose the correct statements from the ones given below.

(More than one may be correct)

a) In general, the five hour time period in which the most number of trips are completed is approximately 5-10.

b) Across all the days, the number of trips from the City that get cancelled is nearly 9 times less than the trips when there were no cars available.

c) Across all the days, the number of trips from the Airport that get cancelled is comparable to the number of trips when there were no cars available.

d) The number of trips getting completed from City is higher than the trips completed from the airport.

In [11]:
```
#Checking option (a)
Q5a=df[df['Status']=='Trip Completed']['Request hour'].value_counts().sort_index()
for i in range(24):
    Q5a[i]=sum(Q5a[i:i+5])
Q5a
```

```
6     785
7     733
8     680
9     614
10    529
11    515
12    491
13    521
14    596
15    674
16    733
17    784
18    787
19    726
20    560
21    399
22    257
23    103
Name: Request hour, dtype: int64
```

We see that the maximum Trips completed in 5 hour window is in 5-10 (854)

In [12]:
```
#for options b and c
len(df[(df['Pickup point']=='City') & (df['Status']=='Cancelled')])
```

Out[12]: 1066

In [13]:
```
len(df[(df['Pickup point']=='Airport') & (df['Status']=='Cancelled')])
```

Out[13]: 198

In [14]:
```
len(df[(df['Status']=='No Cars Available')])
```

Out[14]: 2650

In [15]:
```
#checking for option d
len(df[(df['Pickup point']=='City') & (df['Status']=='Trip Completed')])
```

Out[15]: 1504

In [16]:
```
len(df[(df['Pickup point']=='Airport') & (df['Status']=='Trip Completed')])
```

Out[16]: 1327

Ans: a) In general, the five hour time period in which the most number of trips are completed is approximately 5-10.
d) The number of trips getting completed from City is higher than the trips completed from the airport.

Q6. Request Time

Once you've analysed requests for all the days at an aggregate level, it's time to find patterns in individual days and see if there is anything common across the days. Analyse the distribution of requests across all the days and choose the correct option.

a) The most number of requests are obtained only in the morning hours across all the days.

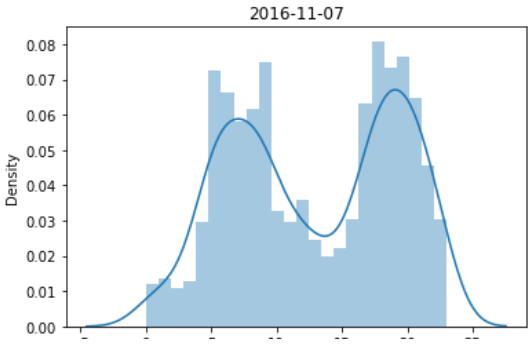
b) The most number of requests are obtained only in the evening hours across all the days.

c) The most number of requests are obtained both in the morning as well as the evening hours across all the days.

```
d) None of these
```

In [17]:
```
#write code here
unique_dates=df['Request date'].unique()
for i in unique_dates:
    sns.distplot(df[df["Request date"]==i]['Request hour'], bins=24)
    plt.title(str(i))
    plt.show()
```

```
C:\Users\Edison George\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a depreca
ted function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level fun
ction with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
Option c is correct statement
```

In [ ]:

Q7. Request Time II

Now that you've observed that the number of requests across all the days is similar, you need to check it across the pick-up points as well. Analyse the distribution of requests across each of the pick-up points exclusively and choose the correct option:

a) The majority of requests from City are obtained during evening hours.

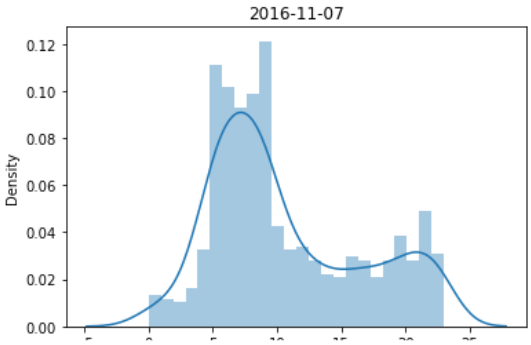
b) The majority of requests from Airport are obtained during morning hours.

c) The majority of requests from City are obtained during morning hours.

d) Both a) and b)

In [18]:
```
for i in unique_dates:
    sns.distplot(df[(df["Request date"]==i) & (df['Pickup point']=='City')]['Request hour'], bins=24)
    plt.title(str(i))
    plt.show()
```

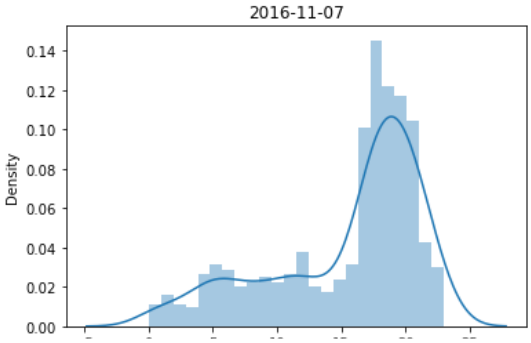
```
C:\Users\Edison George\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a depreca
ted function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level fun
ction with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```




```
From above graph option(c) is true
```

In [19]:
```
for i in unique_dates:
    sns.distplot(df[(df["Request date"]==i) & (df['Pickup point']=='Airport')]['Request hour'], bins=24)
    plt.title(str(i))
    plt.show()
```

```
C:\Users\Edison George\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a depreca
ted function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level fun
ction with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Q8. Request Time III

Now that you have realised the patterns of requests coming from both the city and the airport, the next step is to divide the entire time period to certain slots and then analyse them using plots. Use the following classification table to create the slots and then choose the correct option.

Hour Range(x)	Time Slot
x < 5	Early morning
5 <= x < 10	Morning
10 <= x < 17	Day Time
17 <= x < 22	Evening
22 <= x	Late Night

a) The maximum number of requests across all the days happens in the Evening.

b) The minimum number of requests across all the days happens in Early Morning.

c) The minimum number of requests across all the days happens in Late Night.

d) Both a) and c)

In [20]:
```
def time_period(x):
    if x < 5:
        return "Early Morning"
    elif 5 <= x < 10:
        return "Morning"
    elif 10 <= x < 17:
        return "Day Time"
    elif 17 <= x < 22:
        return "Evenning"
    else:
        return "Late Night"
df['Time slot'] = df['Request hour'].apply(lambda x: time_period(x))
df['Time slot'].value_counts()
```

Out[20]:
```
Evenning         2342
Morning          2103
Day Time         1224
Early Morning     578
Late Night        498
Name: Time slot, dtype: int64
```

Q9. Bar

After this, you decide to plot a bar plot with the time slots in the X-axis and the number of requests in the Y-axis and observe the Status of trips for each time slot. This way you'll get the number of trip request across each time slot and their status as well.

To make things easier, you created an additional column called 'Count' whose value is fixed at 1 for all the rows(df["Count"]=1). Then in order to create a bar plot as mentioned in the first statement, you decided to use sns.barplot() function.

What are the parameters that you need to take here?

a) Take time slots as x and Count as y

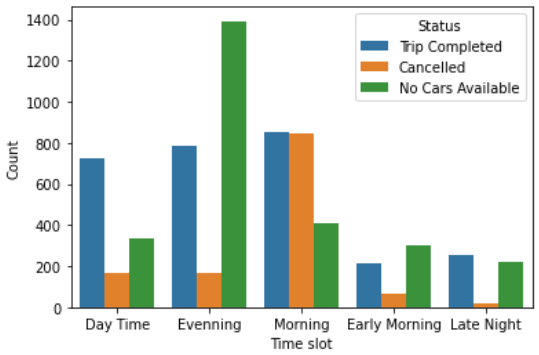
b) Take the Status as hue and df as the data

c) Both (a) and (b)

d) The parameters mentioned above are insufficient

In [21]:
```
df['Count']=1
sns.barplot(data = df, x = 'Time slot', y='Count', hue= 'Status',estimator = np.sum)
```

Out[21]: <AxesSubplot:xlabel='Time slot', ylabel='Count'>



Q10. Correct Bar

Once you have created the bar plot correctly, what major issues are you able to observe? Choose the correct options amongst the following-

a) A majority of requests in the evening are having a “No cars available” status

b) The major issue in the morning slot is that it has a lot of cancellations

c) Both (a) and (b)

d) None of the above

In [ ]:
```
Ans: c) Both (a) and (b)
```

Q11. Demand Supply

Now that you’ve understood the major issues in the morning and the evening time slots, it’s time to compute the demand and supply gap. Here,

Demand - Number of Requests Uber Gets
Supply - Number of Requests Uber is able to fulfil

Demand supply gap = Demand - Supply.

Compute the demand-supply gap for both the morning and the evening slots for the Pick-up points having the most number of requests in that time_slot

a) Morning - 1205 Evening - 1427

b) Morning - 1427 Evening - 1205

c) Morning - 1249 Evening - 1558

d) Morning - 1558 Evening - 1249

In [22]:
```
demandsupply_City=pd.pivot_table(df[df['Pickup point']=='City'], values='Count', index='Time slot', columns ='Status', aggfu
demandsupply_City['Gap']=demandsupply_City['Cancelled'].add(demandsupply_City['No Cars Available'])
demandsupply_City
```

Out[22]:

Status	Cancelled	No Cars Available	Trip Completed	Gap
Time slot				
Day Time	104	247	395	351
Early Morning	63	151	111	214
Evenning	60	71	411	131
Late Night	19	83	115	102
Morning	820	385	472	1205

In [23]:
```
demandsupply_Airport=pd.pivot_table(df[df['Pickup point']=='Airport'], values='Count', index='Time slot', columns ='Status',
demandsupply_Airport['Gap']=demandsupply_Airport['Cancelled'].add(demandsupply_Airport['No Cars Available'])

demandsupply_Airport
```

Out[23]:

Status	Cancelled	No Cars Available	Trip Completed	Gap
Time slot				
Day Time	64	87	327	151
Early Morning	2	148	103	150
Evenning	106	1321	373	1427
Late Night	3	136	142	139
Morning	23	21	382	44

```
a) Morning - 1205 Evening - 1427 is the correct option
```