UBER DATA ANALYSIS

A COURSE PROJECT REPORT

By

Dikcha Singh (RA2011027010096)

Vijay Chandar (RA2011027010083)

Santhana Lakshmi (RA2011027010129)

Under the guidance of

Mrs.D.Hemavathi

Assistant professor

Department of Data Science and Business Systems

In partial fulfilment for the Course

of

18CSE396T – DATA SCIENCE

in

Department of Data Science and Business Systems

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Chengalpattu District

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this mini project report " **Uber Data Analysis** " is the bonafide work of Dikcha Singh (RA2011027010096) , Vijay Chandar (RA2011027010083), Santhana Lakshmi (RA2011027010129) who carried out the project work under my supervision.

Mrs. D.Hemavathi
Assistant professor
Department of Data Science and Business Systems
SRM institute of science and technology

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our Registrar **Dr. S. Ponnusamy**, for his encouragement.

We express our profound gratitude to our Dean of College of Engineering and Technology, **Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to the Chairperson, School of Computing **Dr. Revathi Venkataraman**, for imparting confidence to complete my course project

We wish to express my sincere thanks to Course Audit Professor **Dr. Annapurani Panaiyappan**, Professor and Head, Department of Networking and Communications and Course Coordinators for their constant encouragement and support.

We are highly thankful to our course project faculty **Mrs D.Hemavathi**, Assistant Professor , Department of DSBS for her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our HoD, Professor **Dr. M. Lakshmi ,** Department of DSBS and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project

TABLE OF CONTENTS

CHAPTERS	CONTENTS	PAGENO.
1.	Abstract	
2.	Problem Statement	
3.	Objectives	
4.	Dataset Description	
5.	Report	
6.	Code	
7.	Confusion Matrix	
8.	Output	
9.	Conclusion	

Abstract:

Uber has trouble balancing supply and demand, particularly for requests to and from airports. The problem results from requests to the airport being cancelled or from cars leaving the airport not being available. Both Uber's user base and revenue are impacted by this. The aim of analysis is to identify the root cause of the problem (i.e. cancellation and non-availability of cars to and from the airport) and recommend ways to improve the situation. As a result of the analysis, we should be able to present to the client the root cause(s) and possible hypotheses of the problem(s) and recommend ways to improve them.

Project statement:

You may have some experience of travelling to and from the airport. Have you ever used Uber or any other cab service for this travel? Did you at any time face the problem of cancellation by the driver or non-availability of cars?

Well, if these are the problems faced by customers, these very issues also impact the business of Uber. If drivers cancel the request of riders or if cars are unavailable, Uber loses out on its revenue.

Objectives:

The goal of analysis is to pinpoint the underlying cause of the issue (such as cancellations and car shortages) and make suggestions for how to make things better.

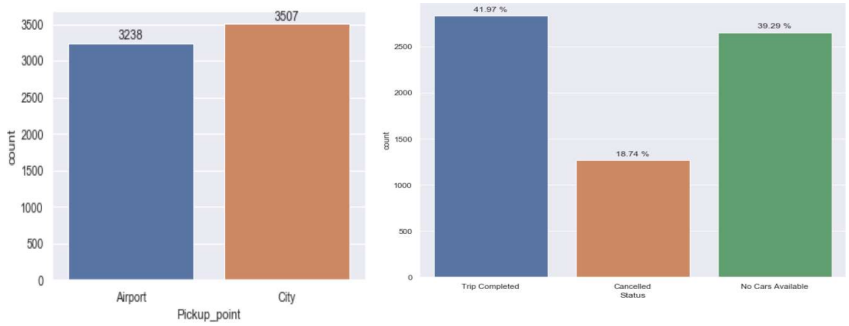
Dataset Description:

Link:https://www.kaggle.com/code/goyalshalini93/uber-supply-demand-gap-analysis-eda/data

The data used is only to and back from airport. The span of the data is of
5 days.6 attributes provided for in the csv file: -

1) Request id-id of the request made. unique in nature
2) Pickup point-location from where the request is being made.
3) Driver id-unique id of the driver.
4) Status –status of request i.e., whether it’s been completed, cancelled or the cab is notavailable.
5) Request timestamp-date and time of the request
6) Drop timestamp-date and time of completion of request

Problem:



- Overall request is the same in city and airport
- From the request, around 19% of the cabs get cancelled
- From the request, around 39% of the cabs are not available.

Problem Solving Methodology:

Data collection and cleaning

- Import the data
- Identifying the data quality issues and clean the data
- Format date and time variables
- Extract new variables required for analysis

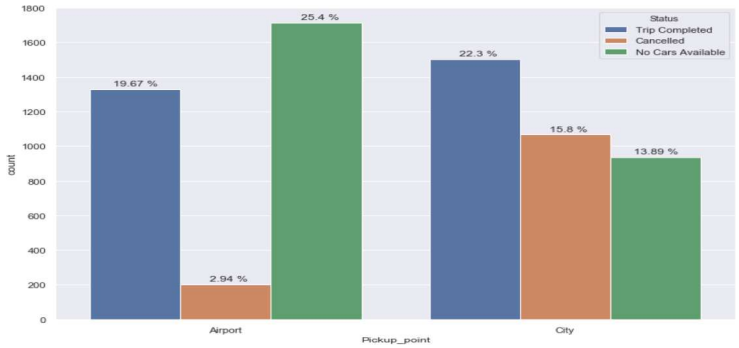
Data exploration and analysis

- Analyze different variables
- Analyze variables across different time slots
- Identify the types of requests, time slots and locations that constitute for the supply demand gap.

Outcome and recommendations

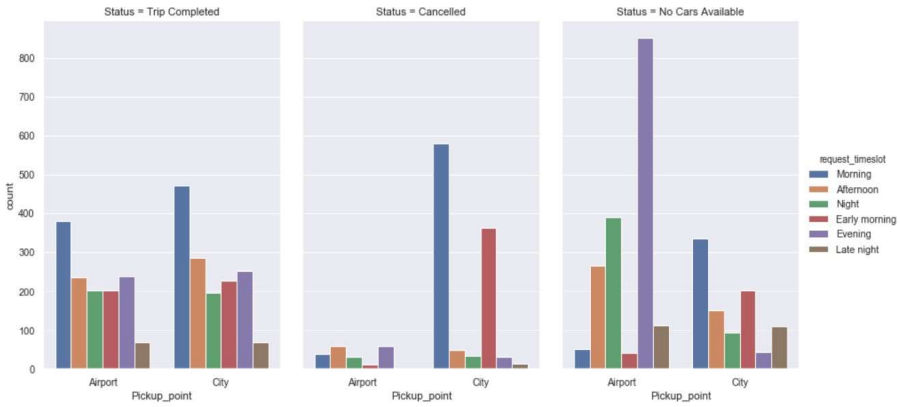
- Visualizing the problem
- Presenting the observations
- Recommendation to resolve the problem

Further analysis of the problem based on location:



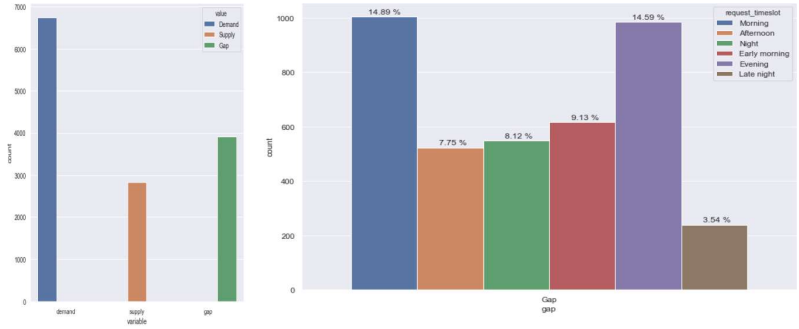
- The % of cabs that get cancelled from city is 16%
- 'No cars available' - is mainly at the airport – 25%

Further analysis of the problem based on time:



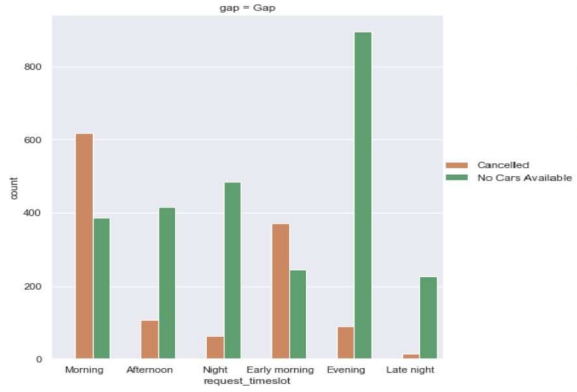
- The cabs that get cancelled in the city are during the morning hours(5am to 9am)
- No cars are available in the airport are during evening hours (5pm to 10 pm)

Analysis of supply-demand gap:



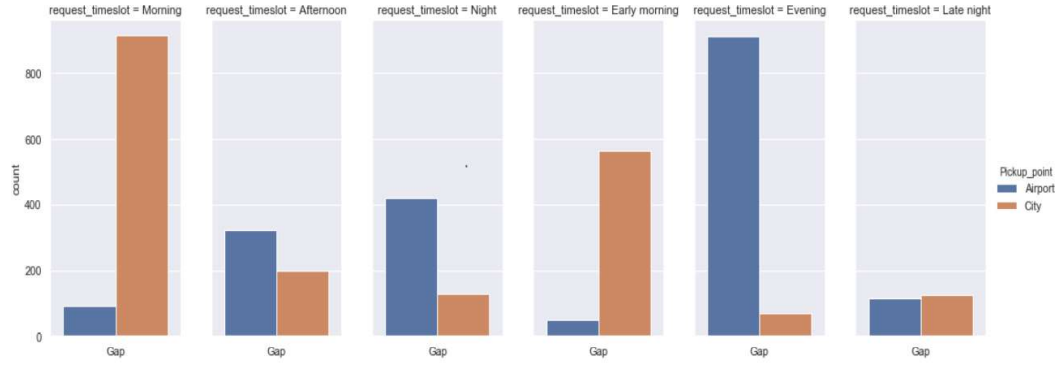
- There is gap of 58% in the supply of cabs.
- In that 58% : 15% gap happens in the morning and evening timeslot each.

Supply-demand gap based on time:



- As we saw that 15% of the gap that exist in the morning is due to cancellation
- 15% of the gap that exist in the evening is due to no cars availability.

Supply-demand gap based on location:



- 15% of the gap that exist in the morning due to cancellation is at City
- 15% of the gap that exist in the evening due to no cars availability is at Airport.

Code:

Libraries Used:

Pandas: Is a fast, powerful, flexible and easy-to-use open-source data analysis and manipulation tool, built on top of the Python programming language.

Datetime: While date and time arithmetic is supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation.

Numpy: Is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays

Matplotlib: Plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

Seaborn: Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Imports

In [12]:

```
import pandas as pd
from datetime import datetime

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

sns.set(style="white")
```

Data Load

In [13]:

```
# Load the Uber Request Data

uber_data = pd.read_csv('Uber Request Data.csv')
```

Explore Data

In [14]:

```
uber_data.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
```

You can notice nulls in columns "Driver id" and "Drop timestamp". These need not be handled as they are not used in the analysis and are going to be dropped from the data set.

In [15]:

```
uber_data.head()
```

Out[15]:

	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

We can see that the date time format in the Request timestamp column is not consistent. This will be handled below to bring all the values to a single date time format for the ease of future analysis.

Data Formatting

In [16]:

```
uber_data['Request timestamp'] = uber_data['Request timestamp'].apply(lambda x: str(x).replace('/','-'))
```

Replaced '/' with '-' for uniformity.

In [17]:

```
uber_data['Request timestamp'] = uber_data['Request timestamp'].apply(lambda x: str(x)+':00' if str(x).count(':') == 1 else str(x))
```

Imputing the seconds value of some timestamps where it is missing to 00 seconds.

In [18]:

```
uber_data['Request timestamp'] = uber_data['Request timestamp'].apply(lambda x: datetime.strptime(str(x), "%d-%m-%Y %H:%M:%S"))
```

Bringing column values to single format "%d-%m-%Y %H:%M:%S"

In [19]:

```
# Examining the data post processing

uber_data.head()
```

Out[19]:

	Request id	Pickup point	Driver id	Status	Request timestamp	Drop timestamp
0	619	Airport	1.0	Trip Completed	2016-07-11 11:51:00	11/7/2016 13:00
1	867	Airport	1.0	Trip Completed	2016-07-11 17:57:00	11/7/2016 18:47
2	1807	City	1.0	Trip Completed	2016-07-12 09:17:00	12/7/2016 9:58
3	2532	Airport	1.0	Trip Completed	2016-07-12 21:08:00	12/7/2016 22:03
4	3112	City	1.0	Trip Completed	2016-07-13 08:33:16	13-07-2016 09:25:47

In [20]:

```
uber_data.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   object
dtypes: datetime64[ns](1), float64(1), int64(1), object(3)
memory usage: 316.3+ KB
```

Dervied Metric(s)

In [21]:

```
# Extract hour from the date to analyze the supply/demand at each hour of a typical day in the data set.

uber_data['Request Hour'] = uber_data['Request timestamp'].dt.hour
```

In [22]:

```
# Create new column Gap which helps during the analysis

uber_data['Supply'] = uber_data['Status'].apply(lambda x: 1 if x == 'Trip Completed' else 0)

uber_data['Gap'] = uber_data['Status'].apply(lambda x: 'No' if x == 'Trip Completed' else 'Yes')
```

In [23]:

```
# Create new column Time Slot which helps during the analysis

TIME_SLOTS = {
    0: 'Late Night',
    1: 'Midnight',
    2: 'Midnight',
    3: 'Early Morning',
    4: 'Early Morning',
    5: 'Early Morning',
    6: 'Morning',
    7: 'Morning',
    8: 'Morning',
    9: 'Morning',
    10: 'Morning',
    11: 'Morning',
    12: 'After Noon',
    13: 'After Noon',
    14: 'After Noon',
    15: 'After Noon',
    16: 'Evening',
    17: 'Evening',
    18: 'Evening',
    19: 'Night',
    20: 'Night',
    21: 'Night',
    22: 'Late Night',
    23: 'Late Night',
}

uber_data['Time Slot'] = uber_data['Request Hour'].apply(lambda x: TIME_SLOTS.get(x))
```

In [24]:

```
# Supply and Demand numbers for analysis

gap_count = pd.DataFrame(index=[0], columns=['Request Hour','Supply','Demand','Gap Count']).dropna()

gap_count['Request Hour'] = uber_data.pivot_table(values = 'Supply', index = 'Request Hour', aggfunc = sum).index

gap_count['Supply'] = uber_data.pivot_table(values = 'Supply', index = 'Request Hour', aggfunc = sum)['Supply']

gap_count['Demand'] = uber_data.pivot_table(values = 'Status', index = 'Request Hour', aggfunc = len)['Status']

gap_count['Gap Count'] = gap_count['Demand'] - gap_count['Supply']
```

Data Cleanup

In [25]:

```
# Dropping columns that are not required for the current analysis
```

```
uber_data = uber_data.drop(columns=['Drop timestamp','Request timestamp','Supply','Driver id','Request id'])
```

Data Verification

In [26]:

```
uber_data.head()
```

Out[26]:

	Pickup point	Status	Request Hour	Gap	Time Slot
0	Airport	Trip Completed	11	No	Morning
1	Airport	Trip Completed	17	No	Evening
2	City	Trip Completed	9	No	Morning
3	Airport	Trip Completed	21	No	Night
4	City	Trip Completed	8	No	Morning

In [27]:

```
# Validating if the numbers are correct against the original data set count

gap_count['Demand'].sum()
```

Out[27]:

```
6745
```

Data Segmentation

In [28]:

```
# Data frame that contains data pertaining to all unfulfilled requests => Cancellation or Unavailability

uber_gap = uber_data.loc[(uber_data['Status'] == 'No Cars Available') | (uber_data['Status'] == 'Cancelled')]

# Data frame that contains data pertaining to fulfilled requests => Trip completed

uber_supply = uber_data.loc[uber_data['Status'] == 'Trip Completed']

# Data frames taht contain data pertaining to fulfilled requests that have drop/pickup location as Airport respectively

uber_supply_airport_inflow = uber_supply.loc[uber_supply['Pickup point'] == 'City']
uber_supply_airport_outflow = uber_supply.loc[uber_supply['Pickup point'] == 'Airport']

# Other data frames with segmented information for airport & city like Cancelled data, No cars available data and
# Trip complete data etc

uber_cancelled = uber_data.loc[uber_data['Status'] == 'Cancelled']
uber_nocar = uber_data.loc[uber_data['Status'] == 'No Cars Available']

uber_cancelled_aiport = uber_data.loc[(uber_data['Status'] == 'Cancelled') & (uber_data['Pickup point'] == 'Airport')]
uber_cancelled_city = uber_data.loc[(uber_data['Status'] == 'Cancelled') & (uber_data['Pickup point'] == 'City')]

uber_nocar_aiport = uber_data.loc[(uber_data['Status'] == 'No Cars Available') & (uber_data['Pickup point'] == 'Airport')]
uber_nocar_city = uber_data.loc[(uber_data['Status'] == 'No Cars Available') & (uber_data['Pickup point'] == 'City')]

uber_tripcomplete_aiport = uber_data.loc[(uber_data['Status'] == 'Trip Completed') & (uber_data['Pickup point'] == 'Airport')]
uber_tripcomplete_city = uber_data.loc[(uber_data['Status'] == 'Trip Completed') & (uber_data['Pickup point'] == 'City')]
```

Most pressing problems for Uber

Frequency of requests that get cancelled or show 'no cars available'

In [29]:

```
# Distinct status counts to be matched with the plot data below

uber_data['Status'].value_counts()
```

Out[29]:

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

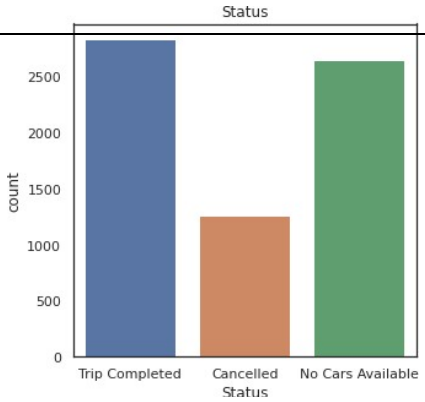
Choice of plots in the analysis that follows

Count Plot => Used to plot segmented categorical data along with their frequencies or counts like the pickup point data across various hours of the day to show the contrinution by each of them.

Dist Plot => Used to plot density of a categorical variable like how the car inflow/outflow to/from the airport looks like to identify the time slots where there is inflow to airport or vice versa.

In [30]:

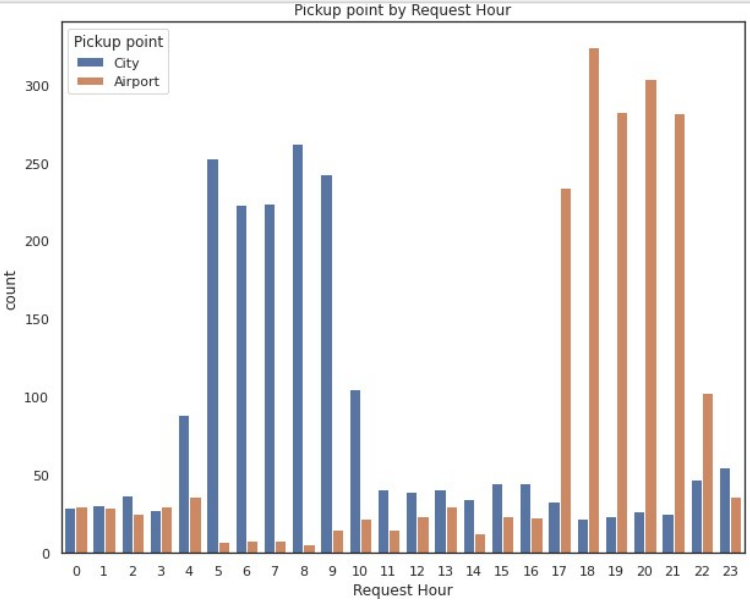
```
# Below plot shows the frequency of requests across different Status

plt.figure(figsize=(5,5))
plt.title('Status')
sns.countplot(x="Status", data=uber_data)
plt.show()
```

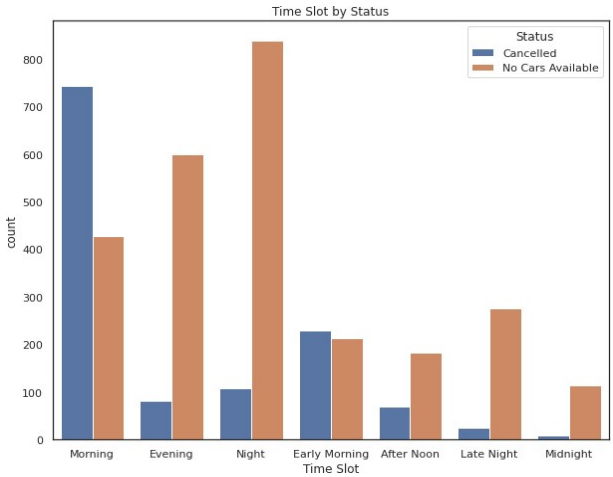


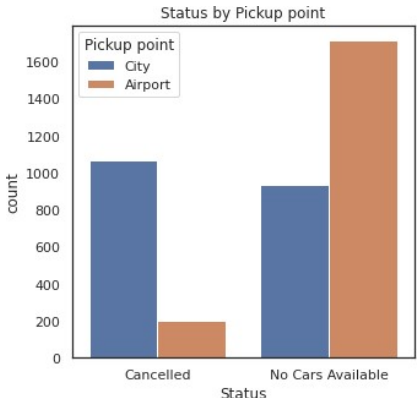
Most problematic types of requests

In [31]:

```
# Pickup point by Request Hour was chosen from Gap data frame to show which specific pickup point contributes to the
# high demand at which request hours

plt.figure(figsize=(10,8))
plt.title('Pickup point by Request Hour')
sns.countplot(x="Request Hour", hue="Pickup point", data=uber_gap)
plt.show()

# Time Slot by Status was chosen from Gap data frame to show which specific time slot contributes to the
# cancellations/no cars available problems

plt.figure(figsize=(10,8))
plt.title('Time Slot by Status')
sns.countplot(x="Time Slot", hue="Status", data=uber_gap)
plt.show()

# Pickup point by Status was chosen from Gap data frame to show which pickup point contributes to the supply-demand gap

plt.figure(figsize=(5,5))
plt.title('Status by Pickup point')
sns.countplot(x="Status", hue="Pickup point", data=uber_gap)
plt.show()
```







Inference

We can infer the below from the plots shown above:

1) Cab cancellations are relatively more when the request is from city to airport and the time slot in which this is happening is 4 AM to 10 AM i.e., in the early morning and morning time slots.

2) No cars available are relatively more when the request is from airport to city and the time slot in which this is happening is 5 PM to 10 PM i.e., in the evening and the night time slots.

Gap between supply and demand

Numbers never lie

In [32]:

```
# Examining the supply and demand numbers to verify against the next plot for each request hour

gap_count
```

Out[32]:

	Request Hour	Supply	Demand	Gap Count
0	0	40	99	59
1	1	25	85	60
2	2	37	99	62
3	3	34	92	58
4	4	78	203	125
5	5	185	445	260
6	6	167	398	231
7	7	174	406	232
8	8	155	423	268
9	9	173	431	258
10	10	116	243	127
11	11	115	171	56
12	12	121	184	63
13	13	89	160	71
14	14	88	136	48
15	15	102	171	69
16	16	91	159	68
17	17	151	418	267
18	18	164	510	346
19	19	166	473	307
20	20	161	492	331
21	21	142	449	307
22	22	154	304	150
23	23	103	194	91

Supply is defined as any request with status "Trip Completed"

Demand is all the requests

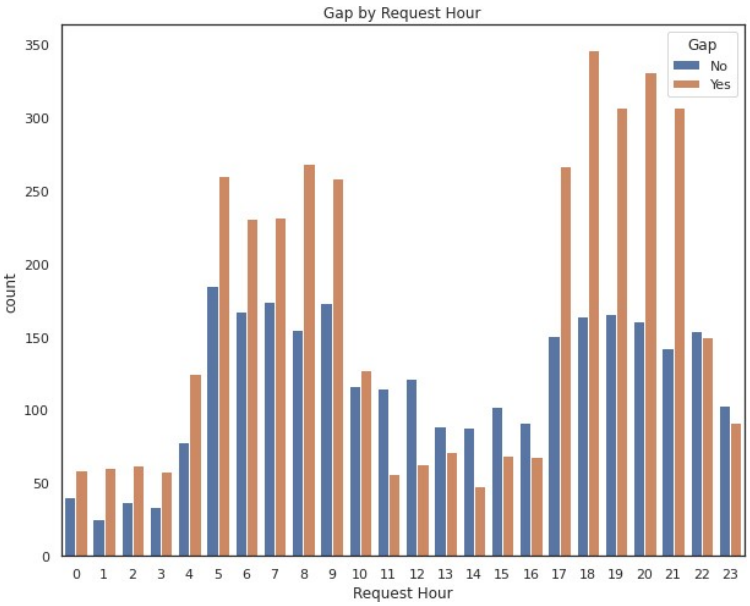
Gap is defined as any request with status either "Cancelled" or "No Cars Available"

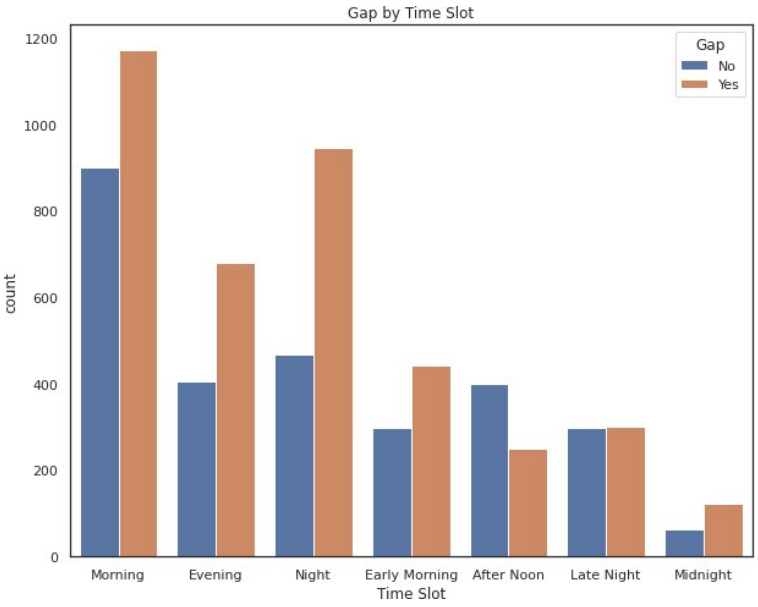
Time slots when the highest gap exists

In [33]:

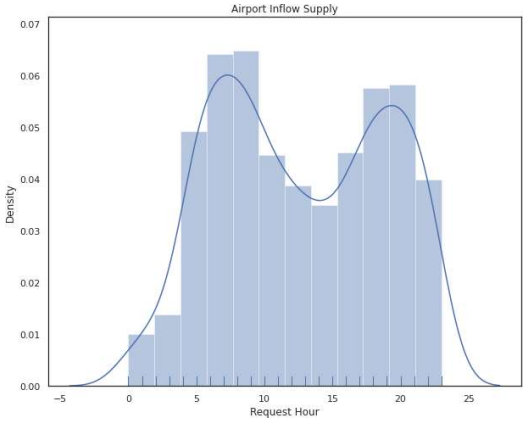
```
# Request hour by Gap from overall data was chosen to analyze which request hours are contributing to the supply-demand gap
```

```
plt.figure(figsize=(10,8))
plt.title('Gap by Request Hour')
sns.countplot(x="Request Hour", hue="Gap", data=uber_data)
plt.show()

# Time Slot by Gap from overall data was chosen to analyze which time slots are contributing to the supply-demand gap

plt.figure(figsize=(10,8))
plt.title('Gap by Time Slot')
sns.countplot(x="Time Slot", hue="Gap", data=uber_data)
plt.show()

# Same as above but with distribution plot

plt.figure(figsize=(10,8))
plt.title('Gap by Request Hour')
sns.distplot(uber_gap['Request Hour'], rug=True, hist=False)
plt.show()
```






```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2103: FutureWarning: The `axis` variable is no longer used and will be removed. Instead, assign variables directly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```

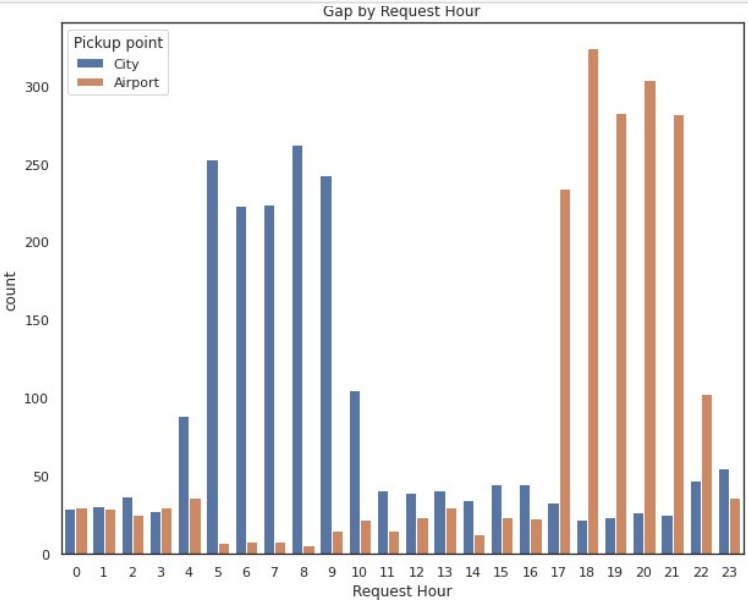


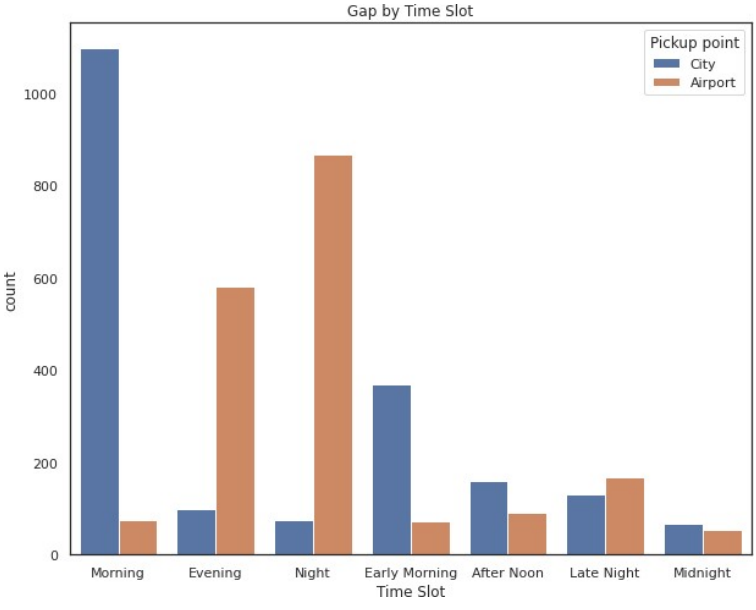
Clearly, the highest gap exists in time slots 4 AM to 10 AM and 5 PM to 10 PM i.e., early morning, morning, evening and night time slots.

Types of requests for which the gap is the most

In [34]:

```
# Request hour by Pickup point from Gap data fame to analyze which pickup points contribute to the supply-demand gap

plt.figure(figsize=(10,8))
plt.title('Gap by Request Hour')
sns.countplot(x="Request Hour", hue="Pickup point", data=uber_gap)
plt.show()

plt.figure(figsize=(10,8))
plt.title('Gap by Time Slot')
sns.countplot(x="Time Slot", hue="Pickup point", data=uber_gap)
plt.show()
```





We can infer the below from the plots shown above:

1) In the time slot 4 AM to 10 AM i.e., early morning and morning time slots, supply-demand gap is high for requests to airport from city.

2) In the time slot 5 PM to 10 PM i.e., evening and night time slots, supply-demand gap is high for request to city from airport.

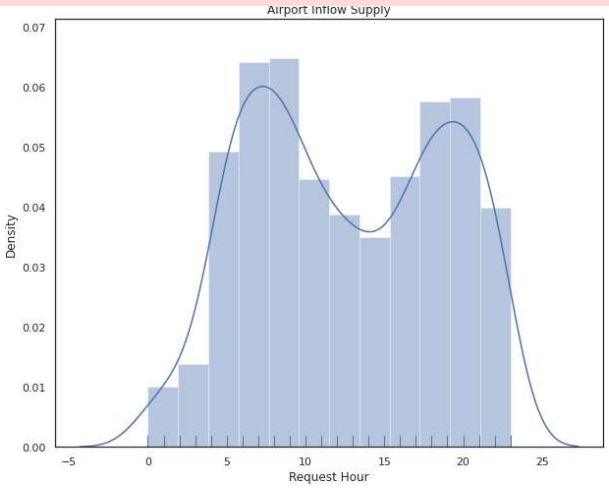
Reason for Supply-Demand Gap

Supply Plots

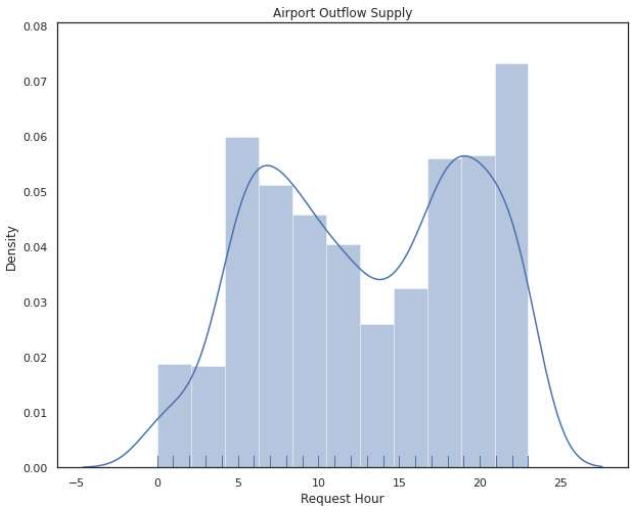
In [35]:

```
# Plot Airport inflow for finding the distribution of cars towards airport across the day

plt.figure(figsize=(10,8))
plt.title('Airport Inflow Supply')
sns.distplot(uber_supply_airport_inflow['Request Hour'], rug=True)
plt.show()

# Plot Airport outflow for finding the distribution of cars from airport across the day

plt.figure(figsize=(10,8))
plt.title('Airport Outflow Supply')
sns.distplot(uber_supply_airport_outflow['Request Hour'], rug=True)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2103: FutureWarning: The `axis` variable is no longer used and will be removed. Instead, assign variables directly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```




```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2103: FutureWarning: The `axis` variable is no longer used and will be removed. Instead, assign variables directly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```

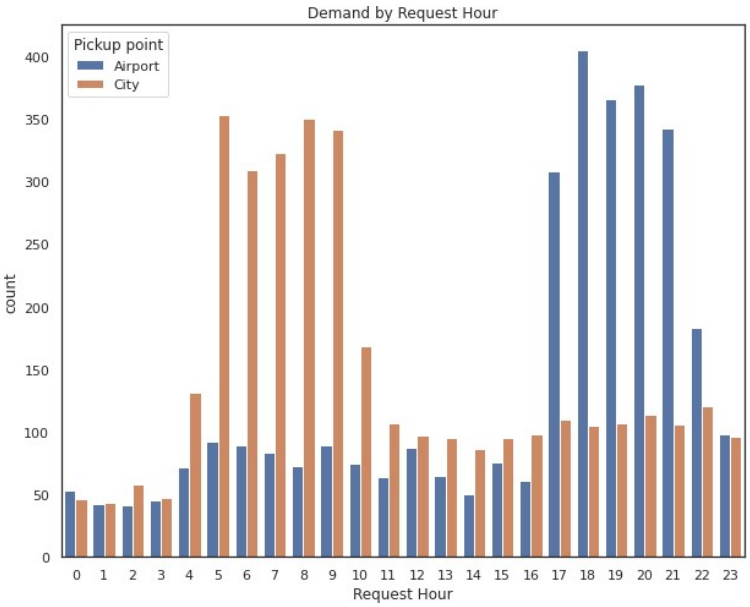


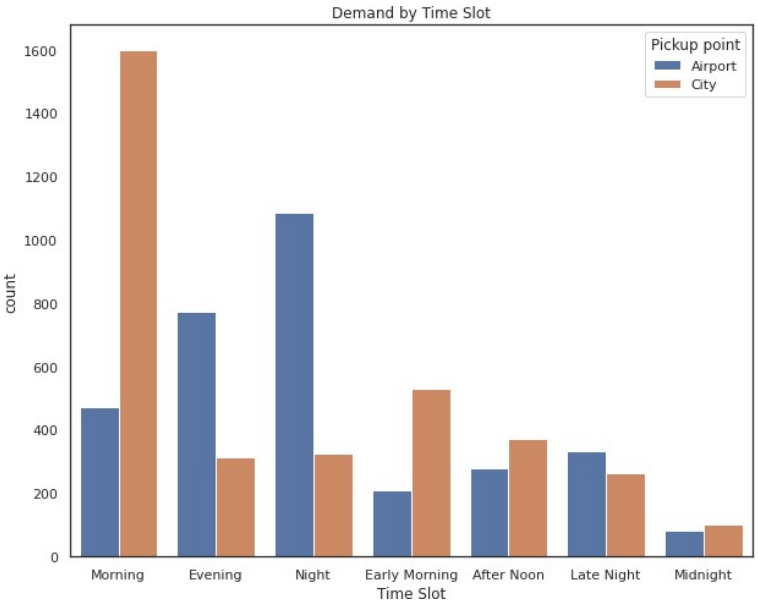
Demand Plots

In [36]:

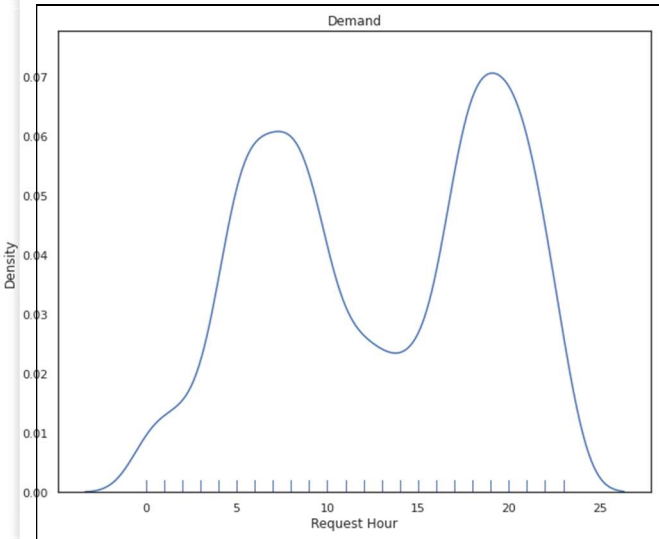
```
# Plots to show demand of cars across the day at various hours/time slots
```

```
plt.figure(figsize=(10,8))
plt.title('Demand by Request Hour')
sns.countplot(x="Request Hour", hue="Pickup point", data=uber_data)
plt.show()

plt.figure(figsize=(10,8))
plt.title('Demand by Time Slot')
sns.countplot(x="Time Slot", hue="Pickup point", data=uber_data)
plt.show()

plt.figure(figsize=(10,8))
plt.title('Demand')
sns.distplot(uber_data['Request Hour'], rug=True, hist=False)
plt.show()
```





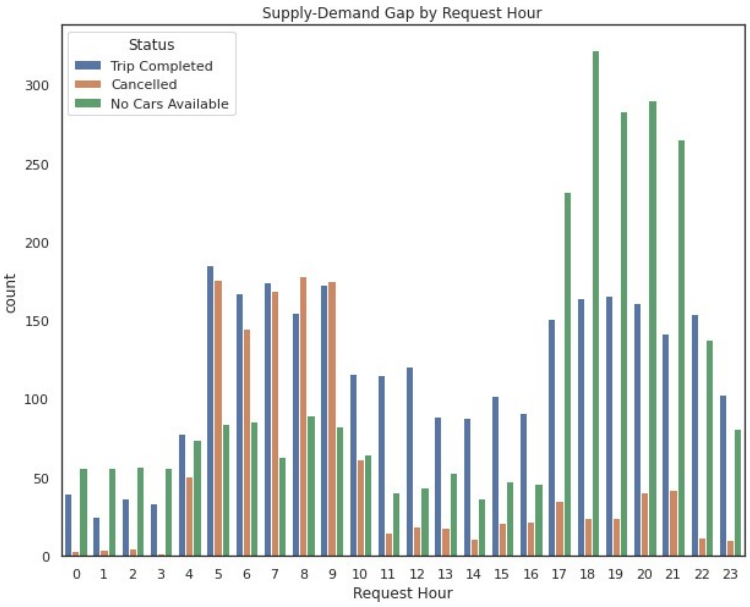

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2103: FutureWarning: The `axis` variable is no longer used and will be removed. Instead, assign variables directly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```

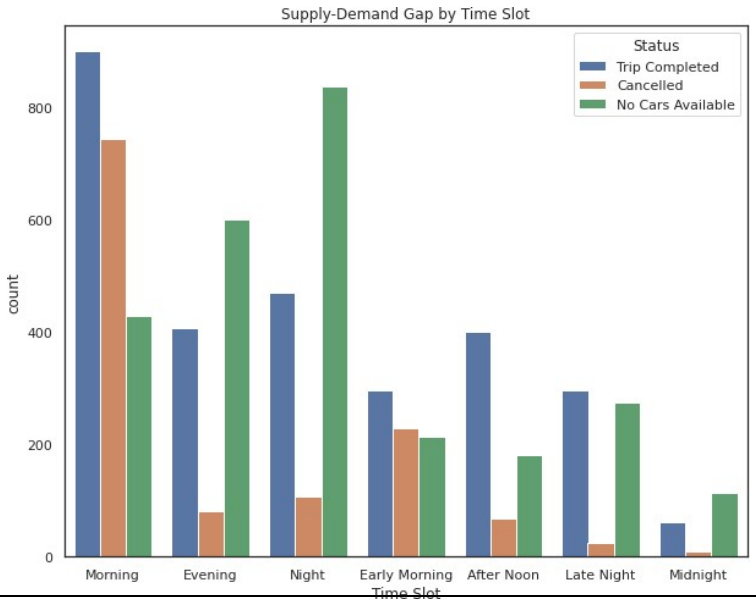


Supply-Demand Gap Plots

In [37]:

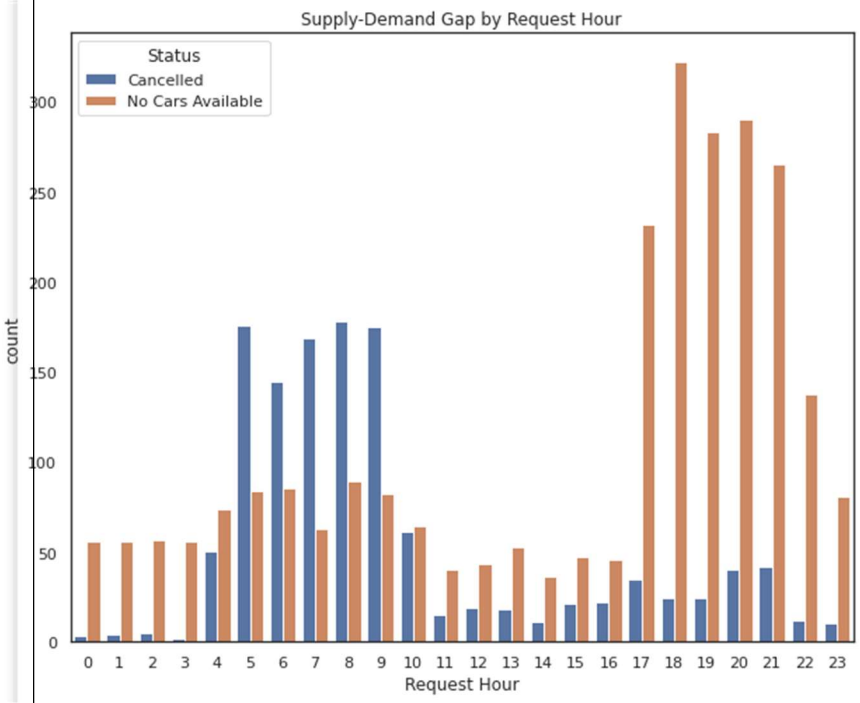
```
# Putting in all together from the data overall data set which shows supply (Trip Completed), demand (Cancellation
# /No Car) and gap

plt.figure(figsize=(10,8))
plt.title('Supply-Demand Gap by Request Hour')
sns.countplot(x="Request Hour", hue="Status", data=uber_data)
plt.show()

plt.figure(figsize=(10,8))
plt.title('Supply-Demand Gap by Time Slot')
sns.countplot(x="Time Slot", hue="Status", data=uber_data)
plt.show()
```

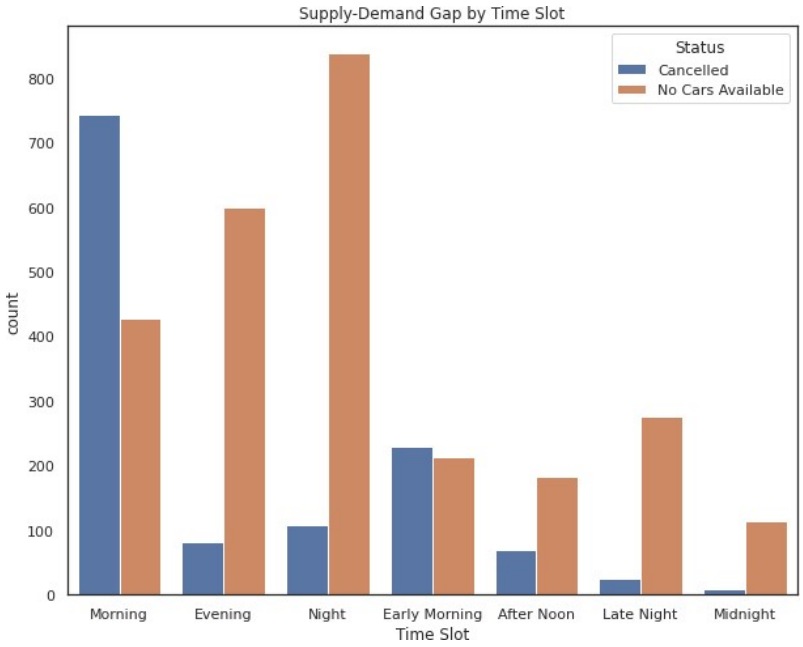




In [38]:

```
# Same as above but the data considered is the Gap data

plt.figure(figsize=(10,8))
plt.title('Supply-Demand Gap by Request Hour')
sns.countplot(x="Request Hour", hue="Status", data=uber_gap)
plt.show()

plt.figure(figsize=(10,8))
plt.title('Supply-Demand Gap by Time Slot')
sns.countplot(x="Time Slot", hue="Status", data=uber_gap)
plt.show()
```





Both "Car Cancellations" from 4 AM to 10 AM and "No Cars Available" from 5 PM to 10 PM are pressing problems for Uber.

1) Car cancellations being more from 4 AM to 10 AM could be due to the fact that the airport doesn't have more arrivals in the same time slot and also till 5 PM there by less number of requests from airport to city. Hence drivers don't want to get stranded at the airport without any requests for a long time which could be used to earn more money if they were in city.

2) No cars available being more from 5 PM to 10 PM could be due to the fact that the airport doesn't have more departures in the same time slot and also till 4 AM there by causing less inflow to airport causing not many cars available at the airport. This could also be due to the time of the day which is when the working hours end for most of the drivers.

Additional Plots for EDA

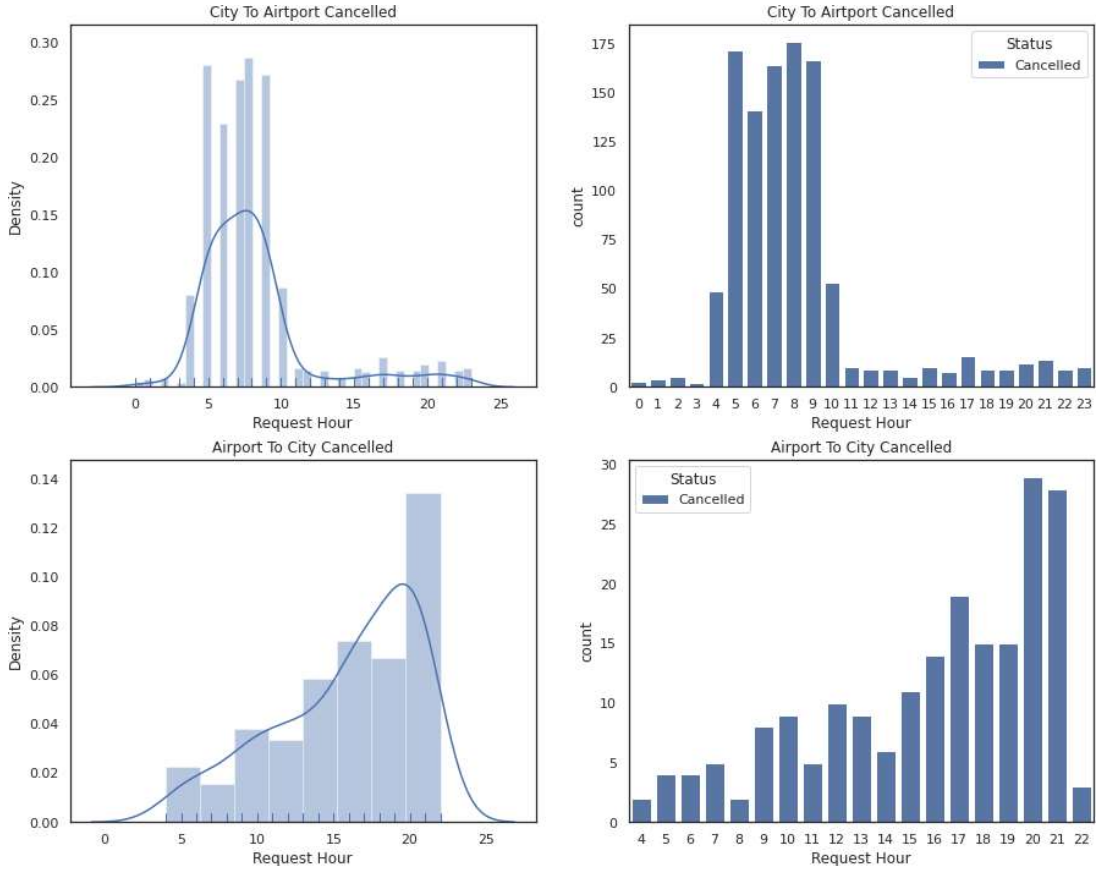
Cancelled Requests

In [39]:

```
# figure size
plt.figure(figsize=(15,12))

# subplot 1
plt.subplot(2, 2, 1)
plt.title('City To Airtport Cancelled')
sns.distplot(uber_cancelled_city['Request Hour'], rug=True)

# subplot 2
plt.subplot(2, 2, 2)
plt.title('City To Airtport Cancelled')
sns.countplot(x="Request Hour", hue="Status", data=uber_cancelled_city)

# subplot 3
plt.subplot(2, 2,  3)
plt.title('Airport To City Cancelled')
sns.distplot(uber_cancelled_aiport['Request Hour'], rug=True)

# subplot 4
plt.subplot(2,  2,  4)
plt.title('Airport To City Cancelled')
sns.countplot(x="Request Hour", hue="Status", data=uber_cancelled_aiport)

plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2103: FutureWarning: The `axis` variable is no longer used and will be removed. I
nstead, assign variables directly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level fu
nction for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2103: FutureWarning: The `axis` variable is no longer used and will be removed. I
nstead, assign variables directly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```



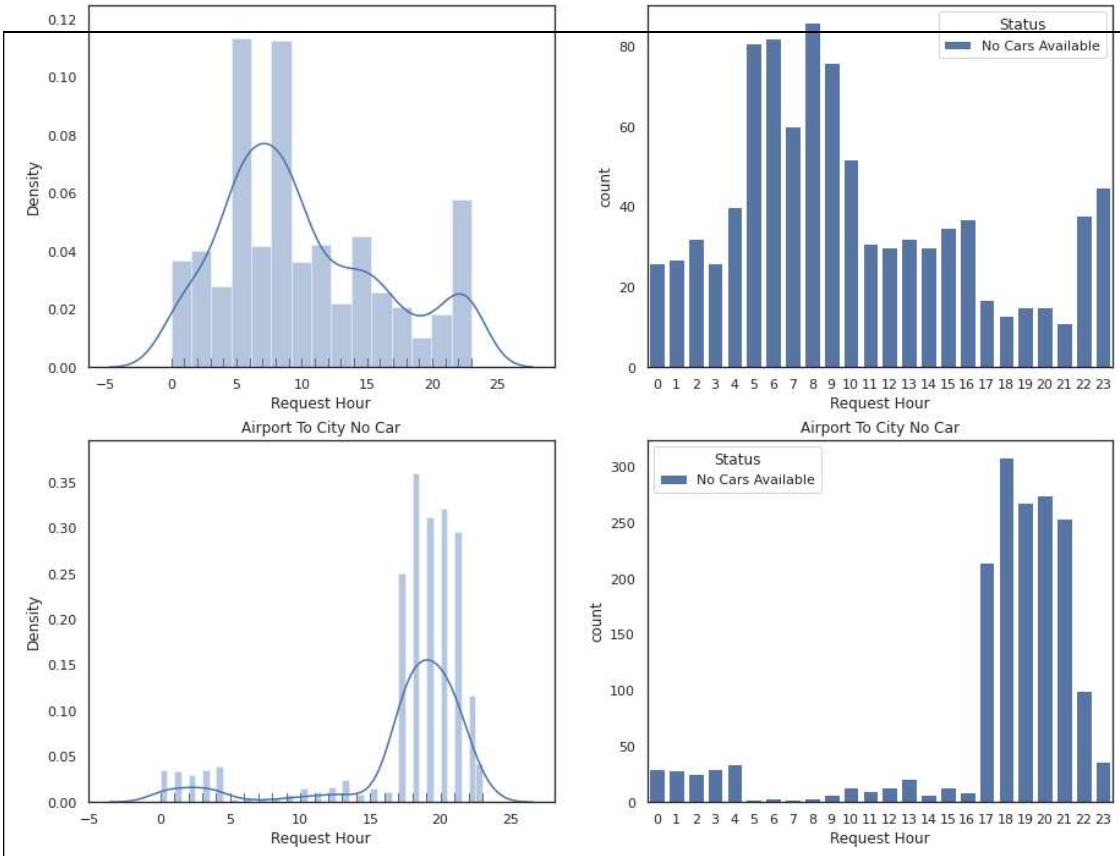
Clearly, city to airport cancellations are standing out between 4 AM to 10 AM i.e, early morning and morning time slots and airport to city cancellations are showing spike at 8 and 9 PM i.e., in the night time slot.

No Cars Available Requests

In [40]:

```
# figure size
plt.figure(figsize=(15,12))

# subplot 1
plt.subplot(2, 2,  1)
plt.title('City To Airport No Car')
sns.distplot(uber_nocar_city['Request Hour'], rug=True)

# subplot 2
plt.subplot(2, 2,  2)
plt.title('City To Airport No Car')
sns.countplot(x="Request Hour", hue="Status", data=uber_nocar_city)

# subplot 3
plt.subplot(2, 2, 3)
plt.title('Airport To City No Car')
sns.distplot(uber_nocar_aiport['Request Hour'], rug=True)

# subplot 4
plt.subplot(2, 2, 4)
plt.title('Airport To City No Car')
sns.countplot(x="Request Hour", hue="Status", data=uber_nocar_aiport)

plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level fu
nction for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2103: FutureWarning: The `axis` variable is no longer used and will be removed. I
nstead, assign variables directly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level fu
nction for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2103: FutureWarning: The `axis` variable is no longer used and will be removed. I
nstead, assign variables directly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```



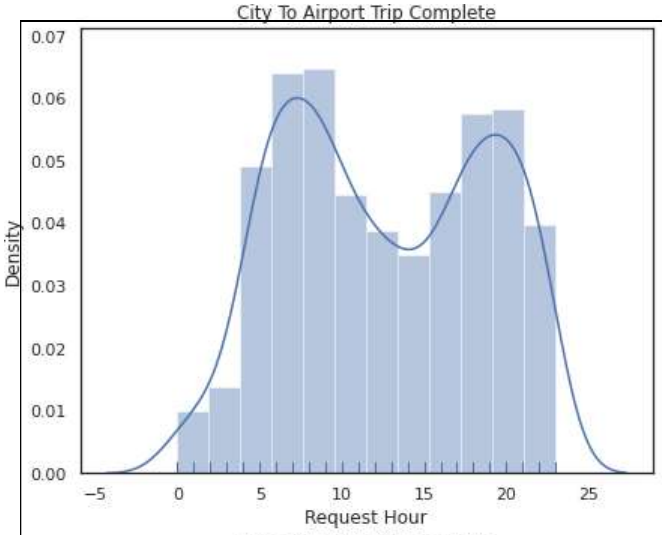
Clearly, airport to city no cars available are standing out between 5 PM to 10 PM i.e, evening and night time slots and airport to city no cars available are showing spike 4 AM and 10 AM i.e., in the early morning and morning time slots.

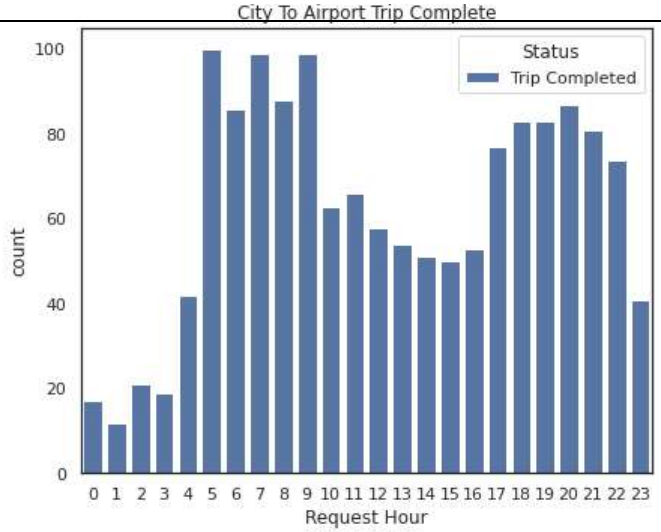
Trip Completed Requests

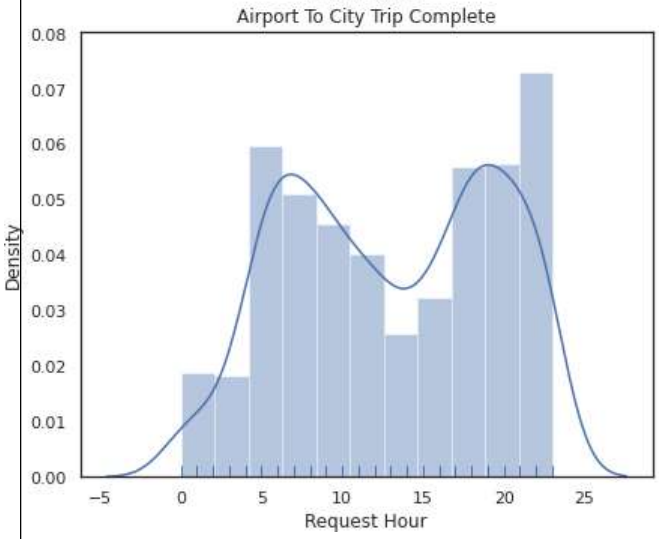
In [41]:

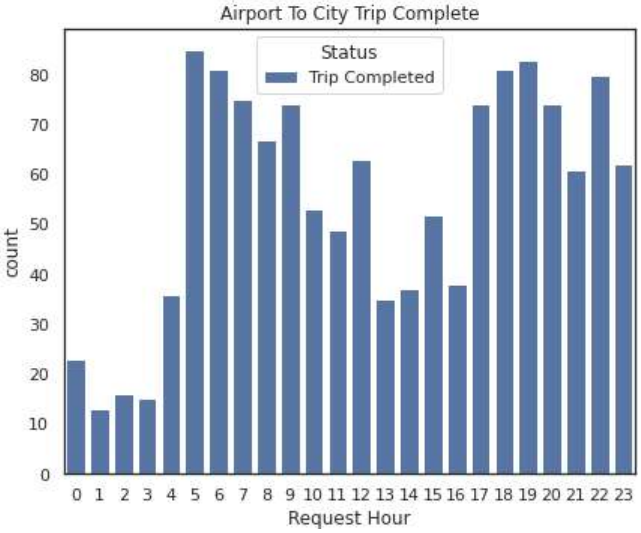
```
# figure size
plt.figure(figsize=(15,12))

# subplot 1
plt.subplot(2, 2, 1)
plt.title('City To Airport Trip Complete')
sns.distplot(uber_tripcomplete_city['Request Hour'], rug=True)

# subplot 2
plt.subplot(2, 2, 2)
plt.title('City To Airport Trip Complete')
sns.countplot(x="Request Hour", hue="Status", data=uber_tripcomplete_city)

# subplot 3
plt.subplot(2, 2, 3)
plt.title('Airport To City Trip Complete')
sns.distplot(uber_tripcomplete_aiport['Request Hour'], rug=True)

# subplot 4
plt.subplot(2, 2, 4)
plt.title('Airport To City Trip Complete')
sns.countplot(x="Request Hour", hue="Status", data=uber_tripcomplete_aiport)

plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2103: FutureWarning: The `axis` variable is no longer used and will be removed. Instead, assign variables directly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2103: FutureWarning: The `axis` variable is no longer used and will be removed. Instead, assign variables directly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
```


City To Airport Trip Complete
Density
Request Hour

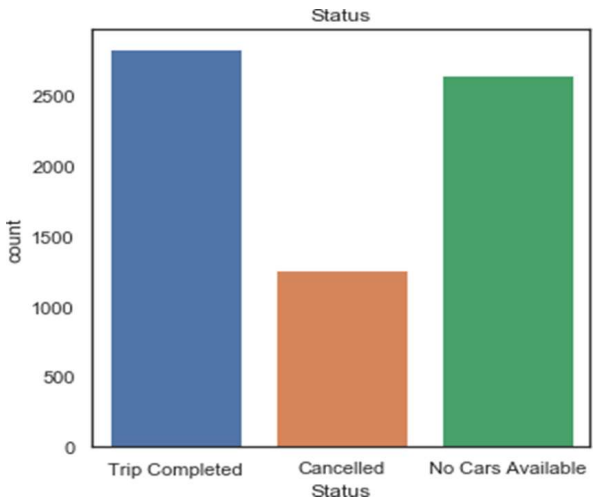

City To Airport Trip Complete
Status
Trip Completed
count
Request Hour


Airport To City Trip Complete
Density
Request Hour


Airport To City Trip Complete
Status
Trip Completed
count
Request Hour

Report:

Frequency:



The adjacent plot shows the frequency of requests that get cancelled or show 'no cars available'.

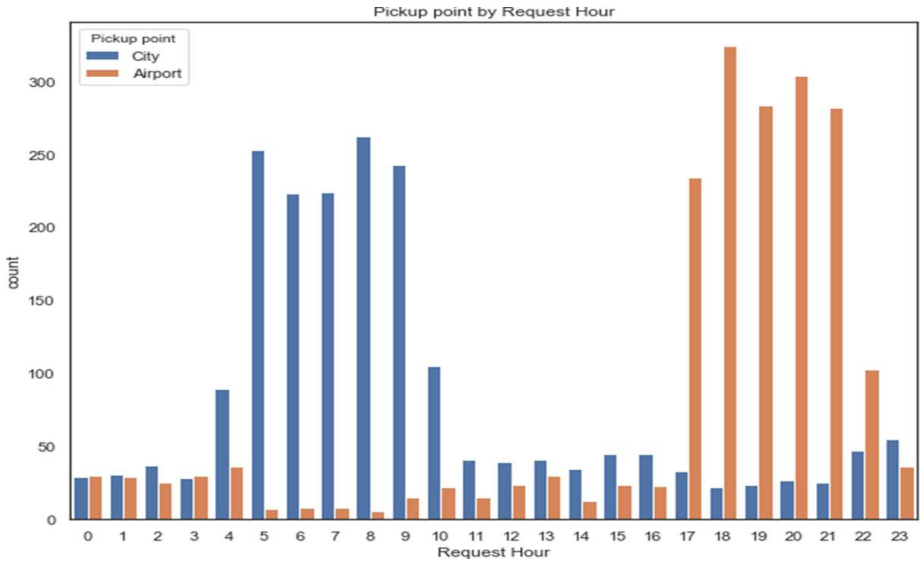
Trip Completed : 2831 No Cars Available : 2650 Cancelled : 1264 Total : 6745

Most problematic types of requests:

Cab cancellations are more when the request is from city to airport.

No cars available are more when the request is from airport to city.

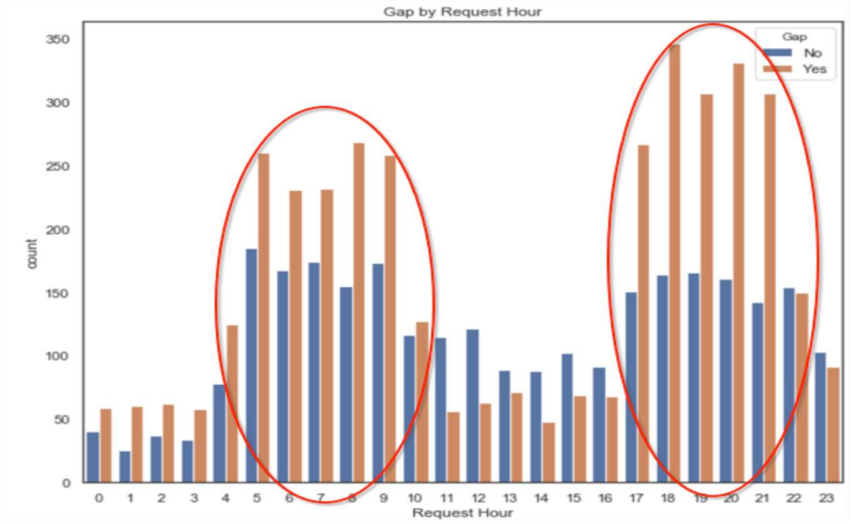
Time slots of problematic requests:



Cab cancellations to airport are more in the time slot 4 AM to 10 AM.

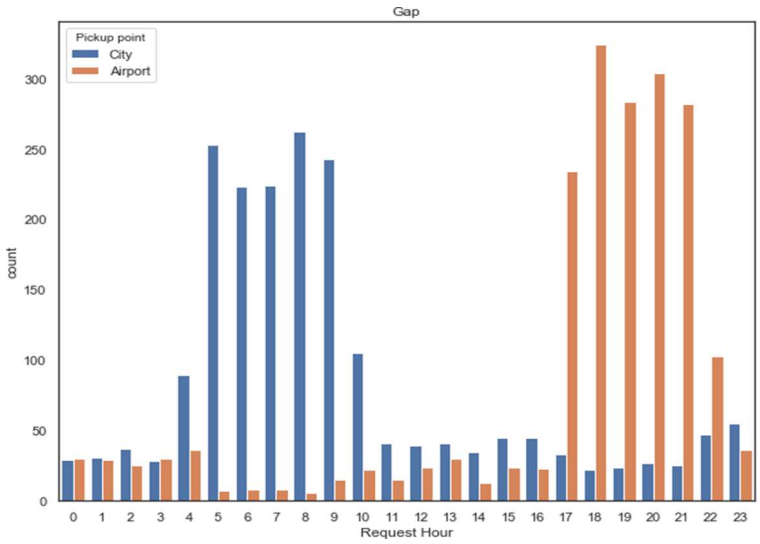
No cars available from airport are more in the time slot 5 PM to 10 PM.

Time slots of the highest gap:



Clearly, the highest gap exists in time slots 4 AM to 10 AM and 5 PM to 10 PM.

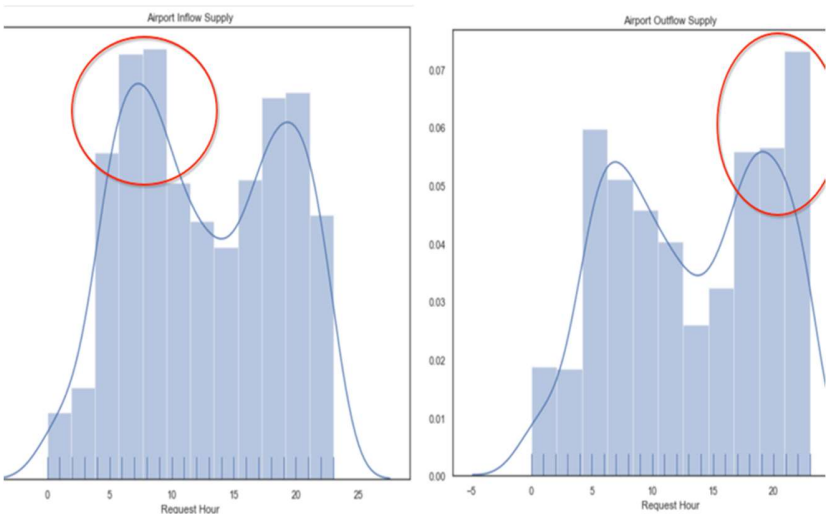
Types of request with high gap:



In the time slot 4 AM to 10 AM, supply-demand gap is high for requests to airport from city.

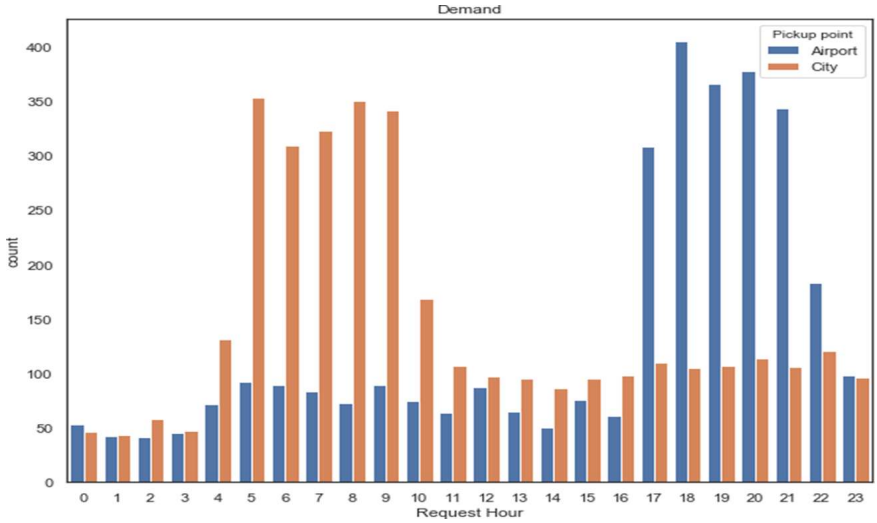
In the time slot 5 PM to 10 PM, supply-demand gap is high for request to city from airport.

Supply:



The airport inflow and outflow of cabs is relatively high in the mornings (4 AM to 10 AM) and in the nights (5 PM to 10 PM) respectively.

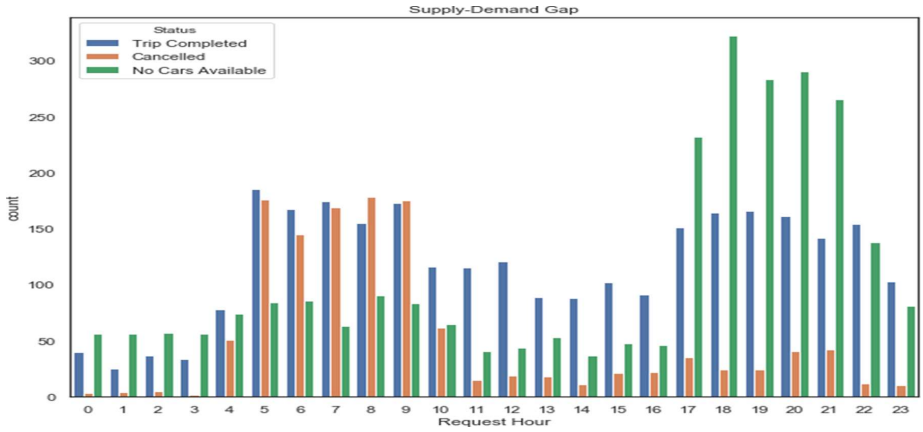
Demand:



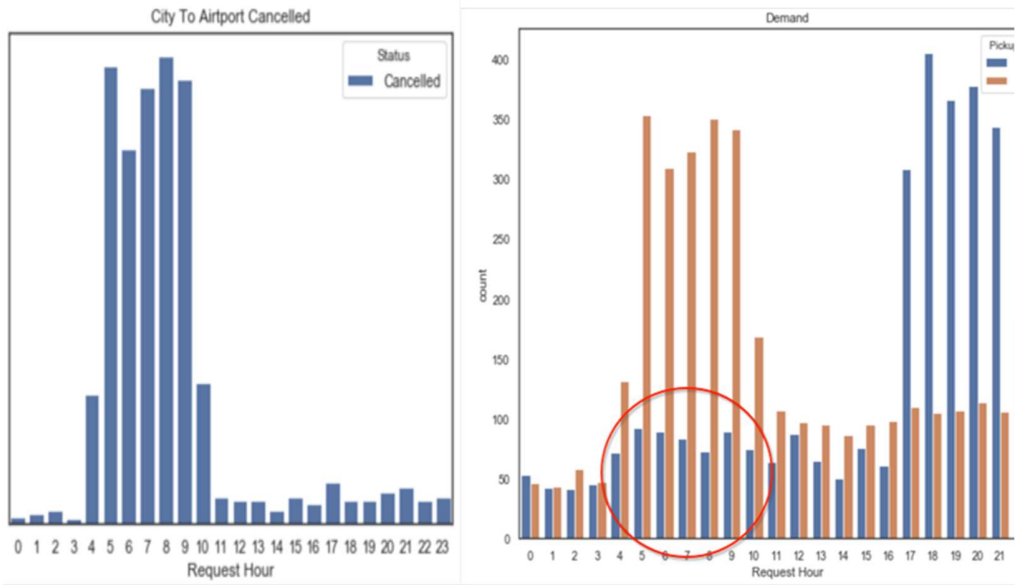
The demand for airport going cabs is more in the mornings in the time slot 4 AM to 10 AM. This could be due to the fact that the departure timings of most airlines fall in this time slot. The demand for city going cabs is more in the nights in the time slot 5 PM to 10 PM.

This could be due to the fact that the arrival timings of most airlines fall in this time slot.

Supply Demand and Gap:



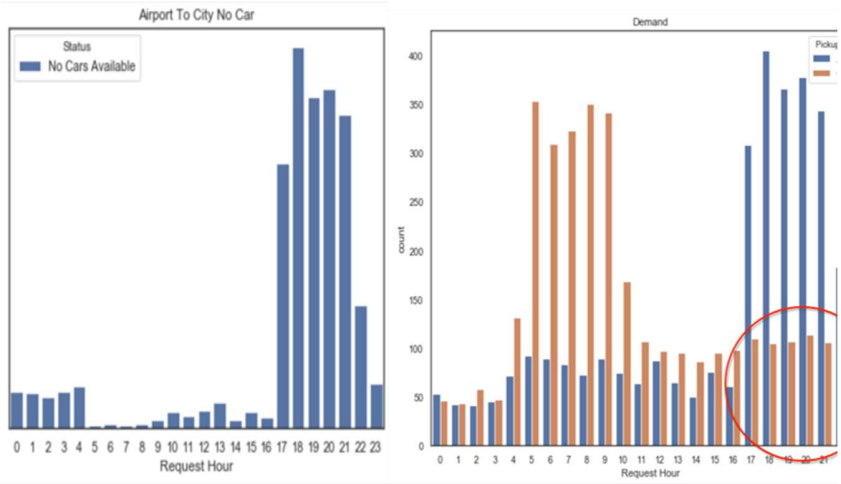
Cancellations to airport:



The adjacent plot shows that the demand for cabs towards city from airport is low from 4AM to 10 AM which could mean that the arrivals at the airport is minimum in this time slot.

Hence, drivers who go to airport during this time will have to wait a lot of time to get a request to get back into city there by wasting time which could be used to make more money if they are in city causing supply- demand gap.

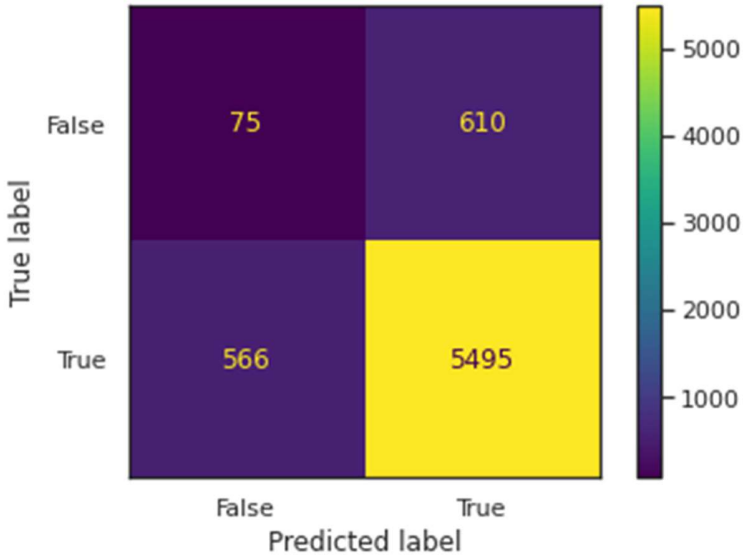
No cars available from airport:



The adjacent plot shows that the demand for cabs towards airport is low from 5 PM to 10 PM which could mean that the departures at the airport are minimum in this time slot.

Hence, there is less airport inflow there by leading to no cars available situation for the passengers coming in flights during this time slot causing the supply- demand gap.

Confusion Matrix:



Recommended Solutions:

Uber Self Drive Away Airport Cars:

To Airport:-

- A minimum of three hours prior to the time of boarding, the customer enters the flight information and the pickup locationin the app.
- The user can either pick up the automobile at a nearby "UberShop" or have the Uber driver drop it off at the pickup point.
- User drives the vehicle to the airport and hands it off to an Uber ground agent there on a platform set aside for pickups and drops.
- User is charged accordingly.This frees user from the tension of whether he/she gets the cab ontime for airport drop and completely driver independent.

From Airport:-

- User uses the app to book car at the airport by providing their flight details.
- The airport ground staff of Uber to drive the car to the arrival platform minutes after the arrival and hand over to the user.
- Users drive the car to the “Uber Shop” or a location which is nearest to their drop location and will be charged accordingly.
- User is charged accordingly.This meets the demand of the late night arrival passengers.

Uber Airport Shuttle:

Uber will launch Airport Shuttle service from the locations where there is a higher demand for Airport cabs at the designated times by evaluating historical data and identifying the hot spots within the city. This service will be offered every 30 minutes. User reserves a spot in the shuttle service at least three hours prior to the scheduled departure time. Shuttle service is provided by a van with room for up to 7 passengers and their bags. To accommodate the demand of late-night arrival customers, the same service might be made accessible from the airport within the designated time periods.

Conclusion:

The supply and demand are vastly out of balance. There is typically a disparity between supply and demand. In comparison to being completed, a lot more rides are unfinished (cancelled or unavailable). Regardless matter how we interpret the facts, the latency exceeds the supply. During peak hours, the lag actually decreases even more, but it still exceeds the completed/supplied trips. To put it another way, the supply-demand imbalance is obvious. It might be fixed by concentrating more on peak times first since they add to the gap that already exists. That ought to be the starting point.

Analysis of conclusion:

Pickup point: City

According to the above plots, the morning time slot has the largest negative gap, which is troublesome because cancellations are occurring. The early rush might be to blame for the cancellation of the request. Given that they might have gotten multiple rides within the city rather than only one to the airport, it may be assumed that the drivers most likely cancelled the request for the airport.

Pickup point: Airport

According to the aforementioned plots, the nighttime time slot has the biggest negative gap, which is an issue because there aren't any cars there. There might not be enough automobiles available because they are not necessarily in the airport area, which could be the cause of the lack of cars.