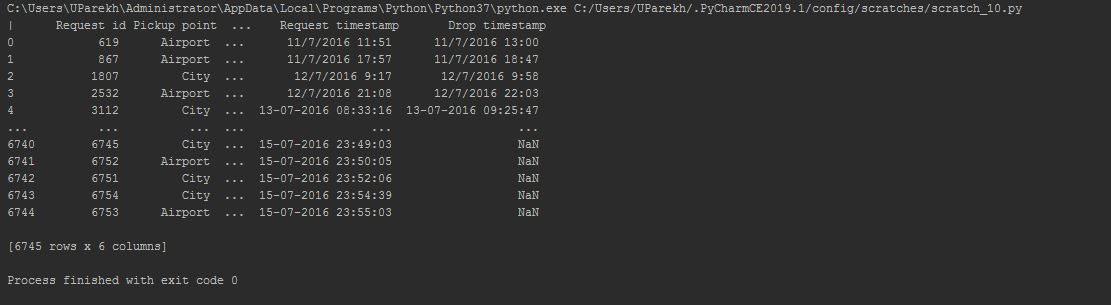
**Uber Demand Supply Analysis**

The analysis done using Python to answer the questions

df = pd.read\_csv('Uber Request Data.csv')  
  
print(df)  
  
cars\_NA = len(df[df['Status'] == 'No Cars Available'])  
trip\_cancel = len(df[df['Status'] == 'Cancelled'])  
print("No. of request not accepted due to unavailability of cars: " + str(cars\_NA))  
print("No. of unattended requests (cars not available + driver cancelled): " + str(cars\_NA+trip\_cancel))



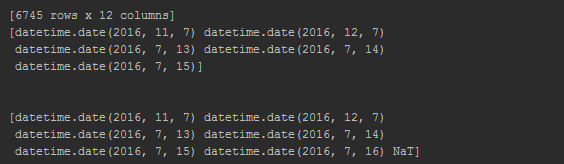


No Driver ID was generated for the number of request that were denied due to unavailabitily of cars. (i.e. 2650 requests)

No Drop timestamp for requests where driver denied and no cars available. (i.e. 3914 requests)

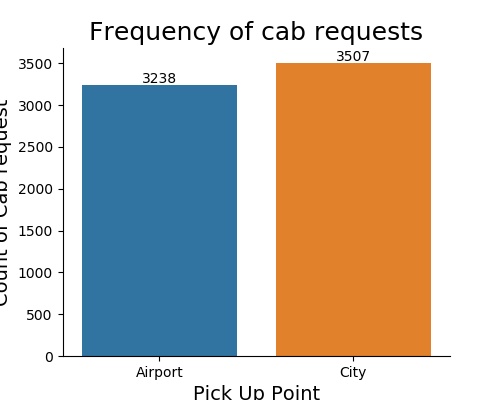
Therefore, there is no missing values in the data set. Now, we need to define the datetime format.

#Cross checking  
print(df['pick\_date'].unique())  
print("\n")  
print(df['drop\_date'].unique())  
  
  
print(df['drop\_date'].value\_counts())



There are 30 rides that ended on the next day (mid-night rides).

#Plot a countplot on "Status" column to identify count of completed trips, cancelled, and no cars available.  
plt.figure(figsize=(8, 5))  
g = sns.countplot(x="Status", data=df)  
sns.despine()  
plt.title("Frequency of request", fontsize = 18)  
plt.xlabel("Trip status", fontsize = 14)  
plt.ylabel("No. of trips", fontsize = 14)  
  
for p in g.patches:  
 g.annotate(p.get\_height(), (p.get\_x() + p.get\_width() / 2., p.get\_height() + 50), ha = 'center', va = 'center')  
  
#plt.legend(loc='upper center', fontsize=12)  
fig =plt.gcf()  
fig.set\_size\_inches(16, 8)  
plt.savefig('Uber/Number\_of\_trips.jpeg', dpi=100)

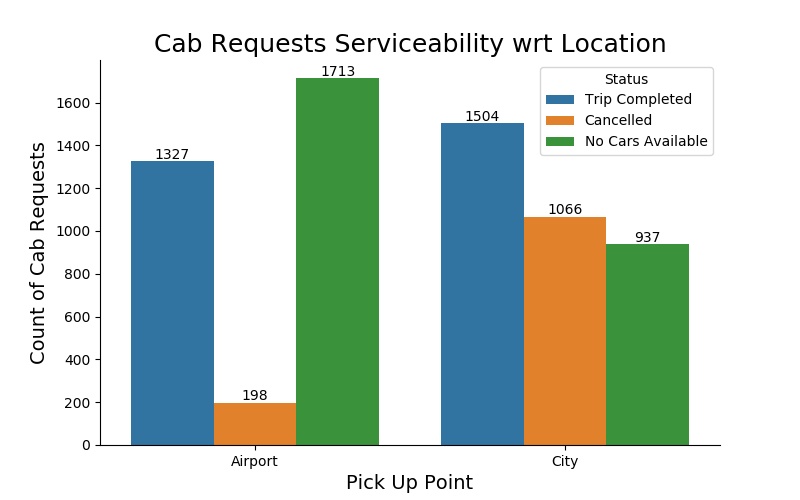


plt.figure(figsize=(5, 4))  
l = sns.countplot(x = "Pickup point", data = df)  
sns.despine()  
plt.title("Frequency of cab requests", fontsize = 18)  
plt.xlabel("Pick Up Point", fontsize = 14)  
plt.ylabel("Count of Cab request", fontsize = 14)  
  
for p in l.patches:  
 l.annotate(p.get\_height(), (p.get\_x() + p.get\_width() / 2., p.get\_height() + 70), ha = 'center', va = 'center')  
  
plt.savefig('Uber/Count\_of\_request.jpeg', dpi=100)

Both the location have approximately same number of cab requests.

Lets look into the serviceability of the requests for both the locations.

#cab requests w.r.t. location  
plt.figure(figsize=(8, 5))  
l = sns.countplot(x = "Pickup point", hue = "Status", data = df)  
sns.despine()  
plt.title("Cab Requests Serviceability wrt Location", fontsize = 18)  
plt.xlabel("Pick Up Point", fontsize = 14)  
plt.ylabel("Count of Cab Requests", fontsize = 14)  
  
for p in l.patches:  
 l.annotate(p.get\_height(), (p.get\_x() + p.get\_width() / 2., p.get\_height() + 30), ha = 'center', va = 'center')  
  
plt.savefig('Uber/Cab\_Request\_Location.jpeg', dpi=100)

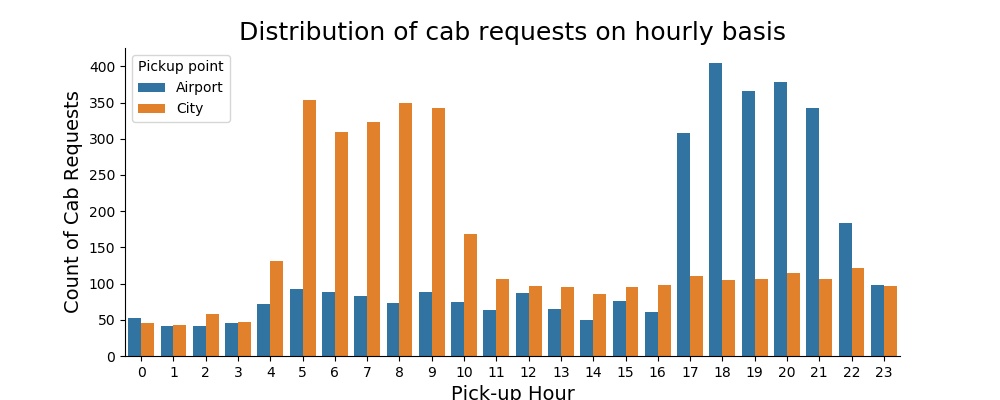


The issue of cab availability is more pressing at airport than in the city.

The number of requests cancelled from the city is higher than airport

Let us analyze the requests with respect to time

#Plot count plot for all days w.r.t. to pick up hour  
plt.figure(figsize=(10, 4))  
sns.countplot(x = "pick\_hour", hue = "Pickup point", data = df)  
sns.despine()  
plt.title("Distribution of cab requests on hourly basis", fontsize = 18)  
plt.xlabel("Pick-up Hour", fontsize = 14)  
plt.ylabel("Count of Cab Requests", fontsize = 14)  
plt.savefig('Uber/pick\_up\_by\_hr.jpeg', dpi=100)



There is a surge in the number of requests at night (5-9 PM) at the airport.

There is a surge in the number of requests in the morning (5-9 AM) in the city.

Let's divide the hours in slots for various time period of the day

2am - 5am: Pre\_Morning

5am - 10am: Morning\_Rush

10am - 5pm: Day\_Time

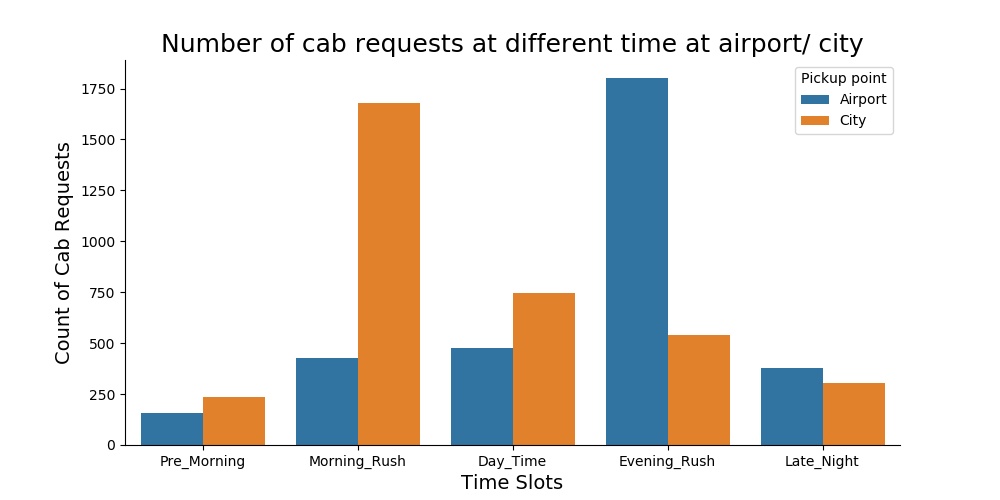
5pm - 10pm: Evening\_Rush

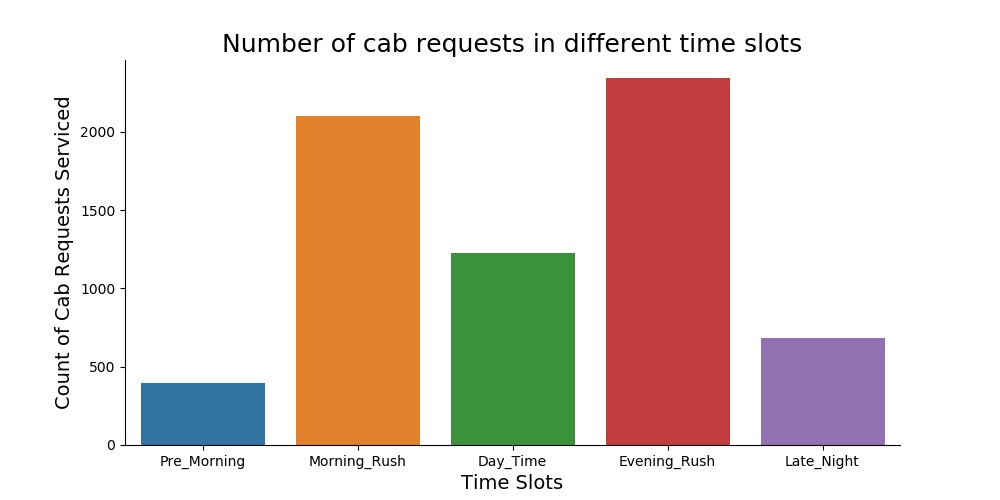
10pm - 2am: Late\_Night

# function to create a time slot for various time period of day  
def time\_period(x):  
 *'divide the time of the day into four categories'* if 2<= x < 5:  
 return "Pre\_Morning"  
 elif 5 <= x < 10:  
 return "Morning\_Rush"  
 elif 10 <= x < 17:  
 return "Day\_Time"  
 elif 17 <= x < 22:  
 return "Evening\_Rush"  
 else:  
 return "Late\_Night"  
  
df['time\_slot'] = df.pick\_hour.apply(lambda x: time\_period(x))  
  
df.time\_slot.value\_counts()

#Plot requests w.r.t. status for different time slots at airport/city  
plt.figure(figsize=(10, 5))  
sns.countplot(x = "time\_slot", data = df, hue = 'Pickup point',  
 order= ['Pre\_Morning', 'Morning\_Rush', "Day\_Time", "Evening\_Rush", "Late\_Night"])  
sns.despine()  
plt.title("Number of cab requests at different time at airport/ city", fontsize = 18)  
plt.xlabel("Time Slots", fontsize = 14)  
plt.ylabel("Count of Cab Requests", fontsize = 14)  
plt.savefig('Uber/Cab\_Request\_by\_time\_Location.jpeg', dpi=100)

#Plot requests w.r.t. status for different time slots  
plt.figure(figsize=(10, 5))  
sns.countplot(x = "time\_slot", hue = "Status", data = df,  
 order= ['Pre\_Morning', 'Morning\_Rush', "Day\_Time", "Evening\_Rush", "Late\_Night"])  
sns.despine()  
plt.title("Cab requests serviceability in different time slots", fontsize = 18)  
plt.xlabel("Time Slots", fontsize = 14)  
plt.ylabel("Count of Cab Requests Serviced", fontsize = 14)  
plt.savefig('Uber/Cab\_Request\_by\_timeSlots.jpeg', dpi=100)



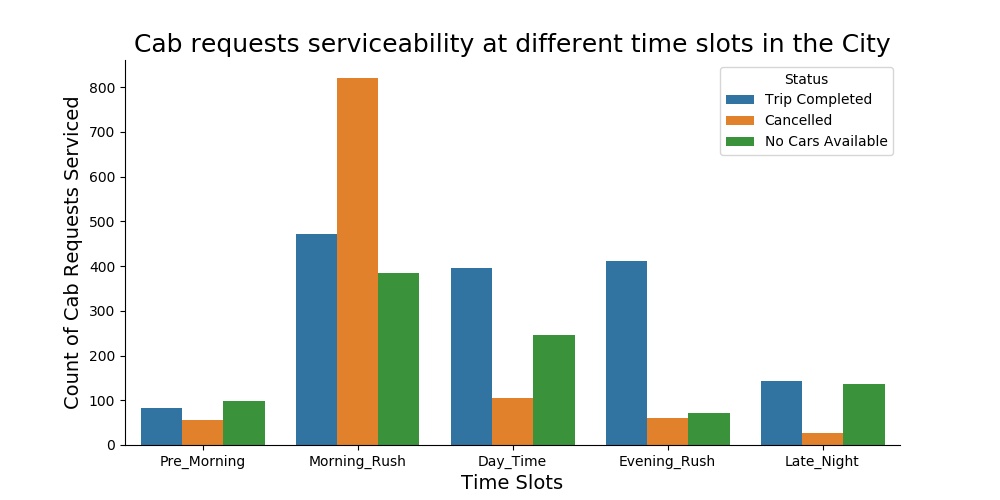


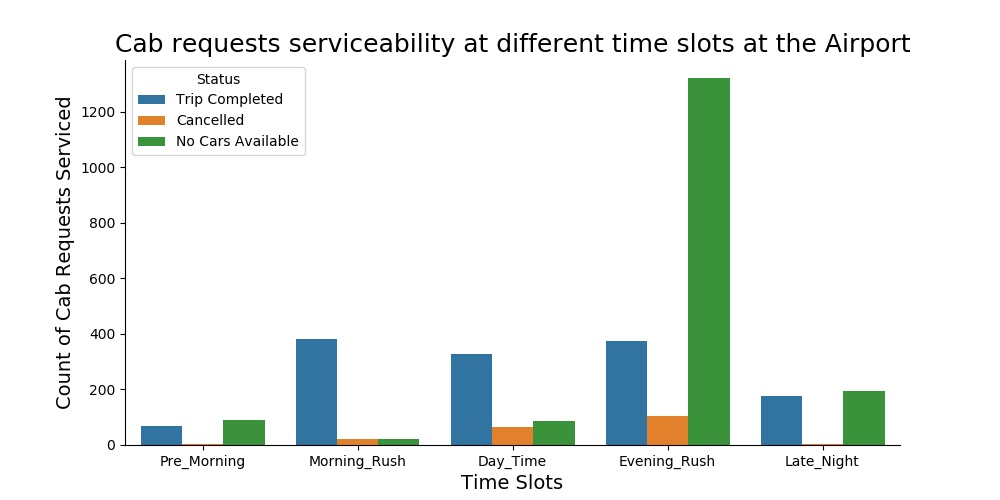
The graph discloses that there is a large amount of requests unserved due to unavailability of cabs in the evening and many requests are cancelled in the morning.

Previously, we have seen that cab availability issue is at the airport and cancellation in the city. We need to check both the issues wrt to time.

airport = df[df['Pickup point'] == 'Airport']  
city = df[df['Pickup point'] == 'City']  
  
  
#Plot requests w.r.t. status for different time slots at the airport  
plt.figure(figsize=(10, 5))  
sns.countplot(x = "time\_slot", hue = "Status", data = airport,  
 order= ['Pre\_Morning', 'Morning\_Rush', "Day\_Time", "Evening\_Rush", "Late\_Night"])  
sns.despine()  
plt.title("Cab requests serviceability at different time slots at the Airport", fontsize = 18)  
plt.xlabel("Time Slots", fontsize = 14)  
plt.ylabel("Count of Cab Requests Serviced", fontsize = 14)  
plt.savefig('Uber/Cab\_Request\_by\_time\_Airport.jpeg', dpi=100)

#Plot requests w.r.t. status for different time slots  
plt.figure(figsize=(10, 5))  
sns.countplot(x = "time\_slot", hue = "Status", data = df,  
 order= ['Pre\_Morning', 'Morning\_Rush', "Day\_Time", "Evening\_Rush", "Late\_Night"])  
sns.despine()  
plt.title("Cab requests serviceability in different time slots", fontsize = 18)  
plt.xlabel("Time Slots", fontsize = 14)  
plt.ylabel("Count of Cab Requests Serviced", fontsize = 14)  
plt.savefig('Uber/Cab\_Request\_by\_timeSlots.jpeg', dpi=100)





A large number of requests are denied in the morning.

By comparing both the graphs, it can be suggested that the less number of cabs are going to the airport from city and thus very less number of cabs are available at the airport in the evening.

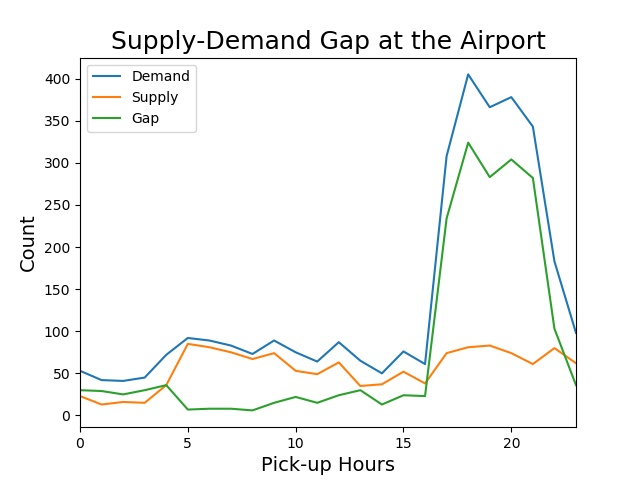
Understanding the supply-demand gap

Demand: Total number of requests for cab rides Supply: Total number of requests completed

We will look at the difference between the supply and demand for both the locations.

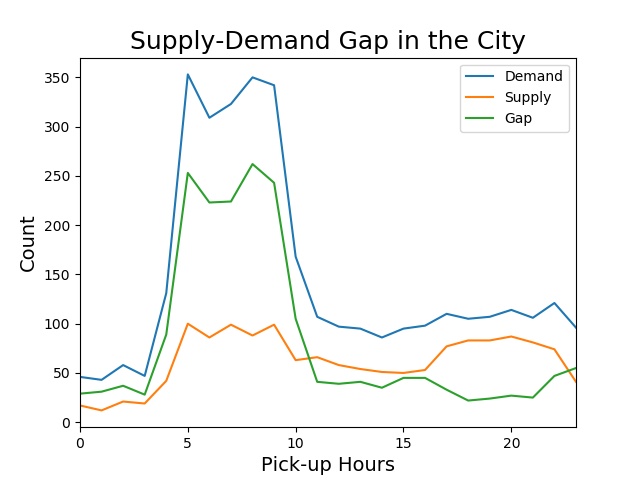
airport['supply\_gap'] = ['Supply' if x == 'Trip Completed' else 'Gap' for x in airport['Status']]  
  
  
airport\_analysis\_table = pd.pivot\_table(airport,  
 index=['pick\_hour'],  
 columns=["supply\_gap"],  
 values=["Pickup point"],  
 aggfunc={"Pickup point": len},  
 dropna="False",  
 margins='True',  
 margins\_name= "Demand",  
 fill\_value= 0)  
  
airport\_analysis\_table=airport\_analysis\_table[:-1]  
airport\_analysis\_table= airport\_analysis\_table.rename(columns={"Pickup point": ' ' })  
airport\_analysis\_table= airport\_analysis\_table.rename(index={'pick\_hour': ' ' })  
airport\_analysis\_table= airport\_analysis\_table.rename(columns={"supply\_gap": 'Request timestamp' })  
  
airport\_analysis = pd.DataFrame(airport.groupby('pick\_hour')['Status'].count())  
airport\_analysis = airport\_analysis.rename(columns = {'Status' : 'Demand'})  
airport\_analysis.head(3)  
  
  
  
airport\_completion = airport[airport['Status'] == 'Trip Completed']  
  
airport\_analysis['Supply'] = pd.DataFrame(airport\_completion.groupby('pick\_hour')['Status'].count())  
airport\_analysis.head(3)  
  
airport\_analysis['Gap'] = airport\_analysis['Demand'] - airport\_analysis['Supply']  
  
plt.figure(figsize=(8, 4))  
airport\_analysis.plot(kind = 'line')  
plt.title("Supply-Demand Gap at the Airport", fontsize = 18)  
plt.xlabel("Pick-up Hours", fontsize = 14)  
plt.ylabel("Count", fontsize = 14)  
plt.savefig('Uber/Supply\_demand\_gap\_Airport.jpeg', dpi=100)

|  |  |  |  |
| --- | --- | --- | --- |
| **supply\_gap** | **Gap** | **Supply** | **Demand** |
| **pick\_hour** |  |  |  |
| **0** | 30 | 23 | 53 |
| **1** | 29 | 13 | 42 |
| **2** | 25 | 16 | 41 |
| **3** | 30 | 15 | 45 |
| **4** | 36 | 36 | 72 |
| **5** | 7 | 85 | 92 |
| **6** | 8 | 81 | 89 |
| **7** | 8 | 75 | 83 |
| **8** | 6 | 67 | 73 |
| **9** | 15 | 74 | 89 |
| **10** | 22 | 53 | 75 |
| **11** | 15 | 49 | 64 |
| **12** | 24 | 63 | 87 |
| **13** | 30 | 35 | 65 |
| **14** | 13 | 37 | 50 |
| **15** | 24 | 52 | 76 |
| **16** | 23 | 38 | 61 |
| **17** | 234 | 74 | 308 |
| **18** | 324 | 81 | 405 |
| **19** | 283 | 83 | 366 |
| **20** | 304 | 74 | 378 |
| **21** | 282 | 61 | 343 |
| **22** | 103 | 80 | 183 |
| **23** | 36 | 62 | 98 |



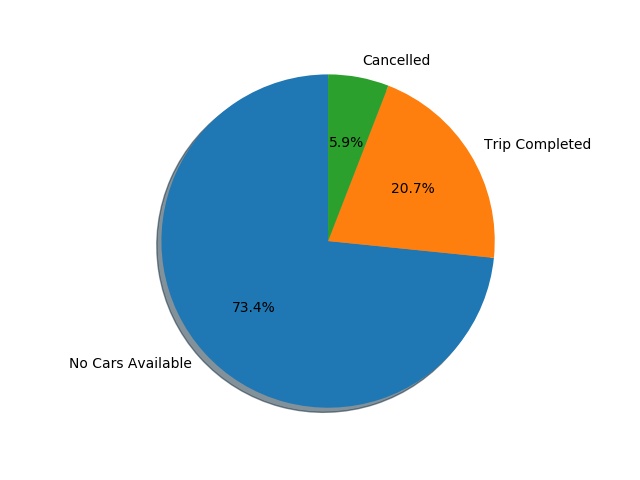
city['supply\_gap'] = ['Supply' if x == 'Trip Completed' else 'Gap' for x in city['Status']]  
  
city\_analysis\_table = pd.pivot\_table(city,  
 index=['pick\_hour'],  
 columns=["supply\_gap"],  
 values=["Pickup point"],  
 aggfunc={"Pickup point": len},  
 dropna="False",  
 margins='True',  
 margins\_name= "Demand",  
 fill\_value= 0)  
  
city\_analysis\_table=city\_analysis\_table[:-1]  
city\_analysis\_table= city\_analysis\_table.rename(columns={"Pickup point": ' ' })  
city\_analysis\_table= city\_analysis\_table.rename(index={'pick\_hour': ' ' })  
city\_analysis\_table= city\_analysis\_table.rename(columns={"supply\_gap": 'Request timestamp' })  
  
  
city\_analysis = pd.DataFrame(city.groupby('pick\_hour')['Status'].count())  
city\_analysis = city\_analysis.rename(columns = {'Status' : 'Demand'})  
  
  
city\_completion = city[city['Status'] == 'Trip Completed']  
  
city\_analysis['Supply'] = pd.DataFrame(city\_completion.groupby('pick\_hour')['Status'].count())  
  
city\_analysis['Gap'] = city\_analysis['Demand'] - city\_analysis['Supply']  
  
plt.figure(figsize=(8, 4))  
city\_analysis.plot(kind = 'line')  
plt.title("Supply-Demand Gap in the City", fontsize = 18)  
plt.xlabel("Pick-up Hours", fontsize = 14)  
plt.ylabel("Count", fontsize = 14)  
plt.savefig('Uber/Supply\_demand\_gap\_city.jpeg', dpi=100)

|  |  |  |  |
| --- | --- | --- | --- |
| **supply\_gap** | **Gap** | **Supply** | **Demand** |
| **pick\_hour** |  |  |  |
| **0** | 29 | 17 | 46 |
| **1** | 31 | 12 | 43 |
| **2** | 37 | 21 | 58 |
| **3** | 28 | 19 | 47 |
| **4** | 89 | 42 | 131 |
| **5** | 253 | 100 | 353 |
| **6** | 223 | 86 | 309 |
| **7** | 224 | 99 | 323 |
| **8** | 262 | 88 | 350 |
| **9** | 243 | 99 | 342 |
| **10** | 105 | 63 | 168 |
| **11** | 41 | 66 | 107 |
| **12** | 39 | 58 | 97 |
| **13** | 41 | 54 | 95 |
| **14** | 35 | 51 | 86 |
| **15** | 45 | 50 | 95 |
| **16** | 45 | 53 | 98 |
| **17** | 33 | 77 | 110 |
| **18** | 22 | 83 | 105 |
| **19** | 24 | 83 | 107 |
| **20** | 27 | 87 | 114 |
| **21** | 25 | 81 | 106 |
| **22** | 47 | 74 | 121 |
| **23** | 55 | 41 | 96 |

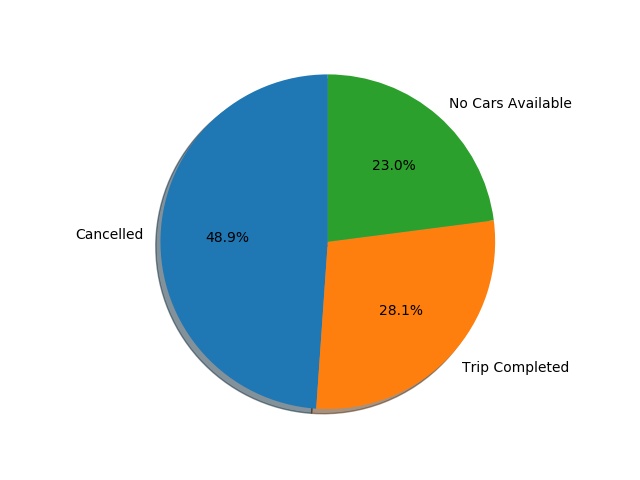


# Let's create pie charts instead of a count plots  
def pie\_chart(dataframe):  
 *"""  
 creates a pie chart  
 input: dataframe with a 'category' as index and a numerical column  
 output: pie chart  
 """* labels = dataframe.index.values  
 sizes = dataframe['Status'].values  
  
 fig1, ax1 = plt.subplots()  
 fig1 = plt.figure(figsize=(8, 4))  
 ax1.pie(sizes, labels=labels, autopct='%1.1f%%', shadow=True, startangle=90)  
 ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.  
 plt.show()  
  
#Status of trips at Airport in the evening rush time  
df\_airport = airport[airport.time\_slot == "Evening\_Rush"]  
df\_airport\_count = pd.DataFrame(df\_airport.Status.value\_counts())  
pie\_chart(df\_airport\_count)  
  
  
  
#Status of trips in the city in the morning rush time  
df\_city = city[city.time\_slot == "Morning\_Rush"]  
df\_city\_count = pd.DataFrame(df\_city.Status.value\_counts())  
pie\_chart(df\_city\_count)

**AIRPORT PIECHART**



**CITY PIE CHART**



### **Findings:**

Airport : Upon analysis, the evening time slot seems to be most problematic for pickup points as airport as the requests are not served due to unavailability of cabs. The unavailability of cabs at the airport in the evening slot may be due to the less number of cabs travelling from city to the airport (less number of request in the city at evening).

City : As per the analysis, the morning time slot is most problematic where the requests are being cancelled. Most probably the requests are being cancelled by the drivers the reason being they have to wait for a long time or return empty, as their are few request for cab at the airport in the morning.

linkcode

### **Recommendations:**

* For bridging the demand supply gap from airport to city in evening, more cabs should be present at the airport or cab pooling facility should be started.
* In morning, Uber can provide return compensation to the driver if had to return empty, this will result in less cancellation of requests by drivers in the morning.