```
# Importing the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Downloading the data
! gdown '1ZhqCqM5xtMNsun-xBhfvtrYH2RlM1Ue5'
```

```
Downloading...
From: https://drive.google.com/uc?id=1ZhqCqM5xtMNsun-xBhfvtrYH2RlM1Ue5
To: /content/uber-data.csv
100% 395k/395k [00:00<00:00, 112MB/s]
```

```
df = pd.read_csv('/content/uber-data.csv',dayfirst = True,na_values = 'NA')
df.head()
```

	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

```
# Checking the shape of the data
df.shape
```

```
(6745, 6)
```

```
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
```

```
# Checking the data type -

df.dtypes
```

	0
Request id	int64
Pickup point	object
Driver id	float64
Status	object
Request timestamp	object
Drop timestamp	object

```
#Checking the statistics of the data
df.describe()
```

	Request id	Driver id
count	6745.000000	4095.000000
mean	3384.644922	149.501343
std	1955.099667	86.051994
min	1.000000	1.000000
25%	1691.000000	75.000000
50%	3387.000000	149.000000
75%	5080.000000	224.000000
max	6766.000000	300.000000

```
# Checking for the total null values
df.isna().sum()
```

	0
Request id	0
Pickup point	0
Driver id	2650
Status	0
Request timestamp	0
Drop timestamp	3914

```
# Convert Request timestamp column to datetime dtype

df['Request timestamp_1'] = pd.to_datetime(df['Request timestamp'],format = '%d-%m-%Y %H:%M',errors = 'coerce')
df['Request timestamp_2'] = pd.to_datetime(df['Request timestamp'],format = '%d/%m/%Y %H:%M',errors = 'coerce')
df['Request timestamp'] = df['Request timestamp_1'].combine_first(df['Request timestamp_2'])

# Convert Drop timestamp column to datetime dtype

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

```
# Dropping the unwanted rows
df.drop(['Request timestamp_1','Request timestamp_2','Drop timestamp_1','Drop timestamp_2'],inplace = True,axis = 1)
```

```
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  2674 non-null   datetime64[ns]
 5   Drop timestamp     0 non-null      datetime64[ns]
dtypes: datetime64[ns](2), float64(1), int64(1), object(2)
memory usage: 316.3+ KB
```

Extracting new features from the existing ones

```
# Extract hour from the Request timestamp -

df['Request_hour'] = df['Request timestamp'].dt.hour
df.head()
```

	Request id	Pickup point	Driver id	Status	Request timestamp	Drop timestamp	Request_hour
0	619	Airport	1.0	Trip Completed	2016-07-11 11:51:00	NaT	11.0
1	867	Airport	1.0	Trip Completed	2016-07-11 17:57:00	NaT	17.0
2	1807	City	1.0	Trip Completed	2016-07-12 09:17:00	NaT	9.0
3	2532	Airport	1.0	Trip Completed	2016-07-12 21:08:00	NaT	21.0
4	3112	City	1.0	Trip Completed	NaT	NaT	NaN

```
# Separate 5 different timeslots from the Hour - Dawn, Early Morning, Noon, Late Evening, Night -

df['Timeslot'] = df['Request_hour'].apply(lambda x : 'Dawn' if x <= 4 else ('Early Morning' if x <= 9 else ('Afternoon' if x <= 16 else (
```

```
# using cut method

Times = [0,4,9,16,21,24]
Values = ['Dawn','Early Morning','Afternoon','Late Evening','Night']
pd.cut(df['Request_hour'],bins = Times,labels = Values)
```

	Request_hour
0	Afternoon
1	Late Evening
2	Early Morning
3	Late Evening
4	NaN
...	...
6740	NaN
6741	NaN
6742	NaN
6743	NaN
6744	NaN

6745 rows × 1 columns

```
# Distinguish the Supply-Demand Gap by a new variable Cab Availability where Supply is when Trip is Completed, all else is Demand -

df['Cab Availability'] = df['Status'].apply(lambda x : 'Available' if x == 'Trip Completed' else 'Not Available')
```

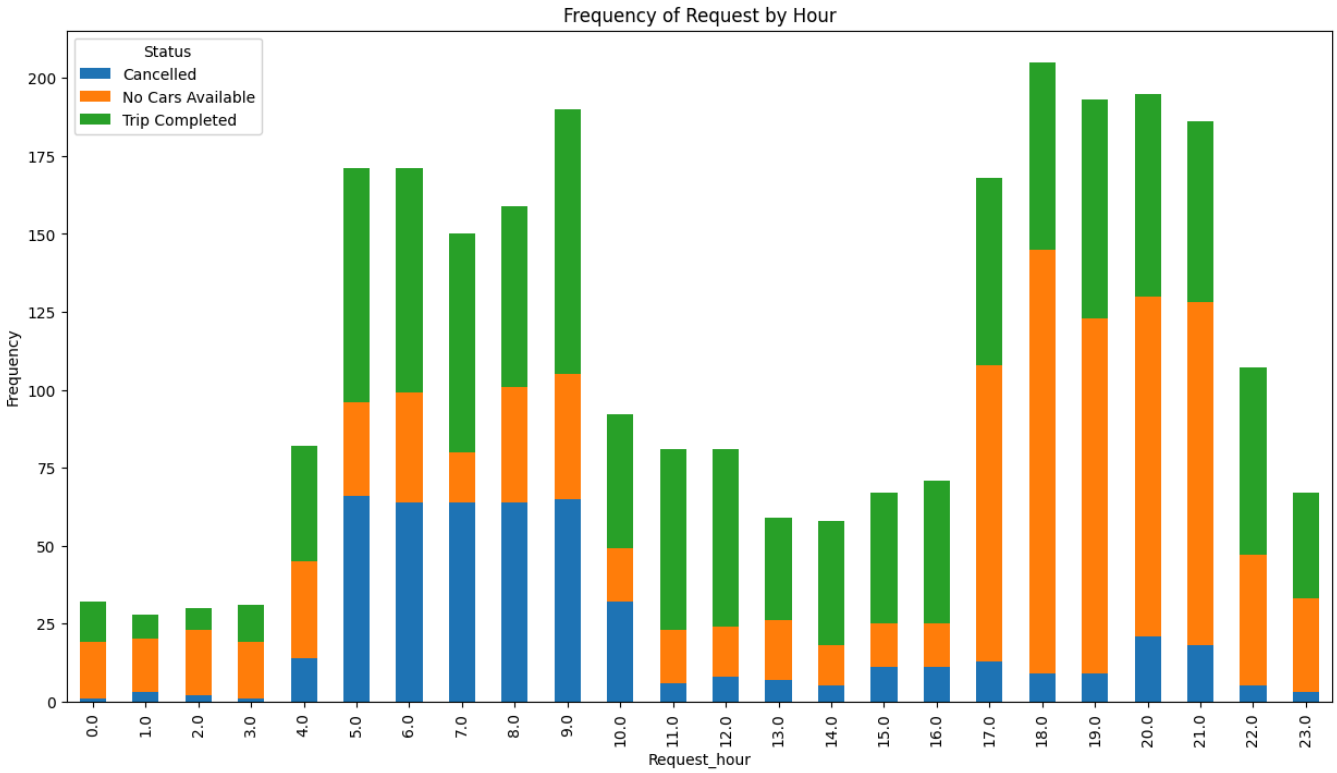
```
df.head()
```

	Request id	Pickup point	Driver id	Status	Request timestamp	Drop timestamp	Request_hour	Timeslot	Cab Availability
0	619	Airport	1.0	Trip Completed	2016-07-11 11:51:00	NaT	11.0	Afternoon	Available
1	867	Airport	1.0	Trip Completed	2016-07-11 17:57:00	NaT	17.0	Late Evening	Available
2	1807	City	1.0	Trip	2016-07-12	NaT	9.0	Early	Available

```
# Frequency of Requests by Hour -

df.groupby(['Request_hour','Status']).size().unstack().plot( kind = 'bar',stacked = True,figsize = (15,8))
plt.title('Frequency of Request by Hour')
plt.ylabel('Frequency')
```

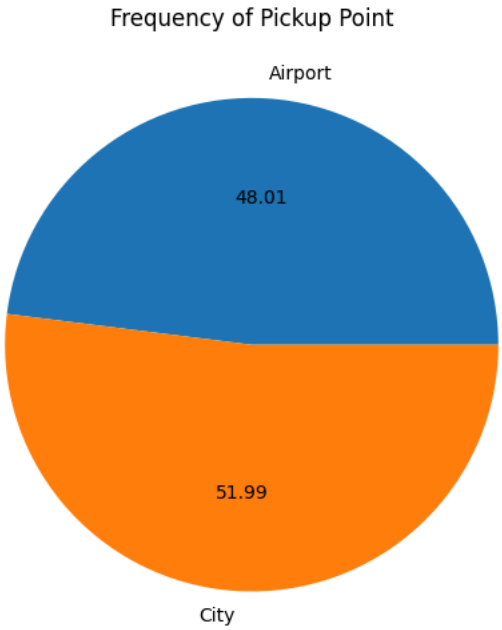
```
Text(0, 0.5, 'Frequency')
```




```
# Types of Requests (city-airport or airport-city) -

df.groupby(['Pickup point']).size().plot(kind = 'pie',stacked = True,figsize = (6,6),autopct = '%.2f',table =True)
plt.title('Frequency of Pickup Point')
```

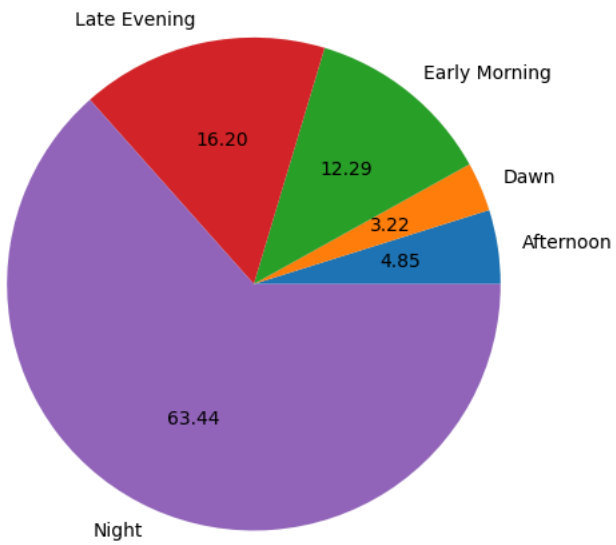
```
Text(0.5, 1.0, 'Frequency of Pickup Point')
```



Airport	City
3238	3507

```
# Distribution of Time Slots -
df[df['Cab Availability'] == 'Not Available'].groupby(['Timeslot']).size().plot(kind = 'pie',stacked = True,figsize= (6,6),autopct = '%
```

```
<Axes: >
```



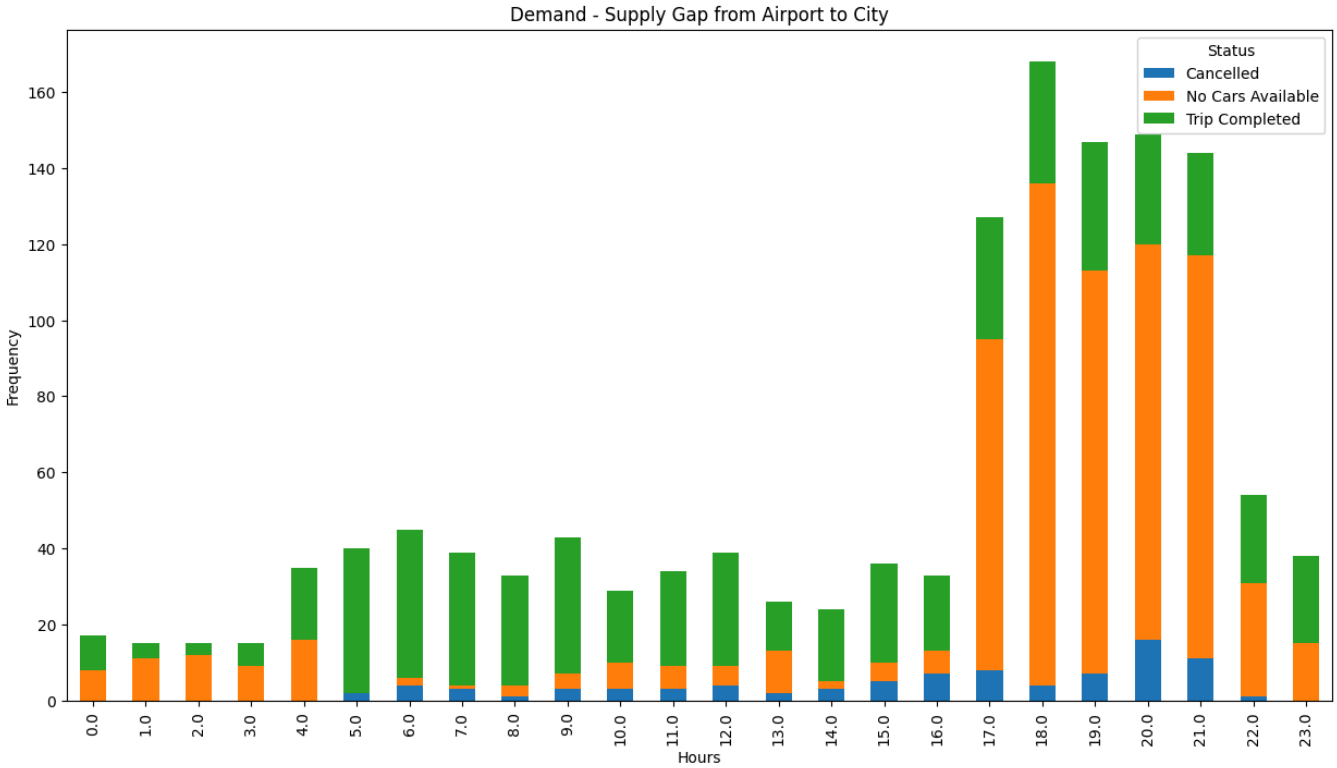
Afternoon	Dawn	Early Morning	Late Evening	Night
190	126	481	634	2483

Observation: Late Evenings and Early Mornings are not recommended for Airport-City transport or vice versa.

```
# Demand-Supply Gap from Airport to City -

df[df['Pickup point'] == 'Airport'].groupby(['Request_hour','Status']).size().unstack().plot(kind = 'bar',stacked = True,figsize = (15,
plt.title('Demand - Supply Gap from Airport to City')
plt.xlabel('Hours')
plt.ylabel('Frequency')
```

```
Text(0, 0.5, 'Frequency')
```



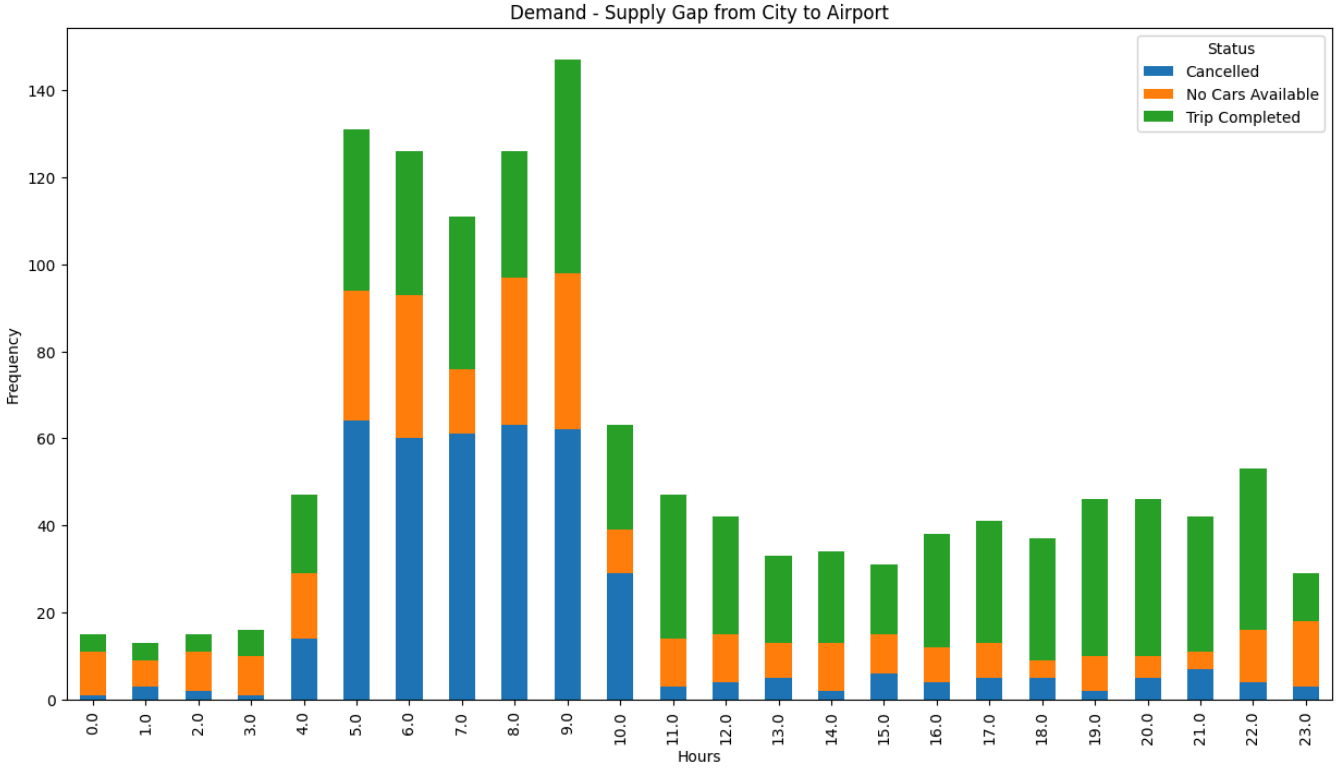
Observation:

There is very high demand for cabs from Airport to City between 5:00 PM – 9:00 PM But the supply is very less due primarily due to 'No Cabs Available'

```
# Demand-Supply Gap from City to Airport -

df[df['Pickup point'] == 'City'].groupby(['Request_hour','Status']).size().unstack().plot(kind = 'bar',stacked = True,figsize = (15,8))
plt.title('Demand - Supply Gap from City to Airport')
plt.xlabel('Hours')
plt.ylabel('Frequency')
```

```
Text(0, 0.5, 'Frequency')
```

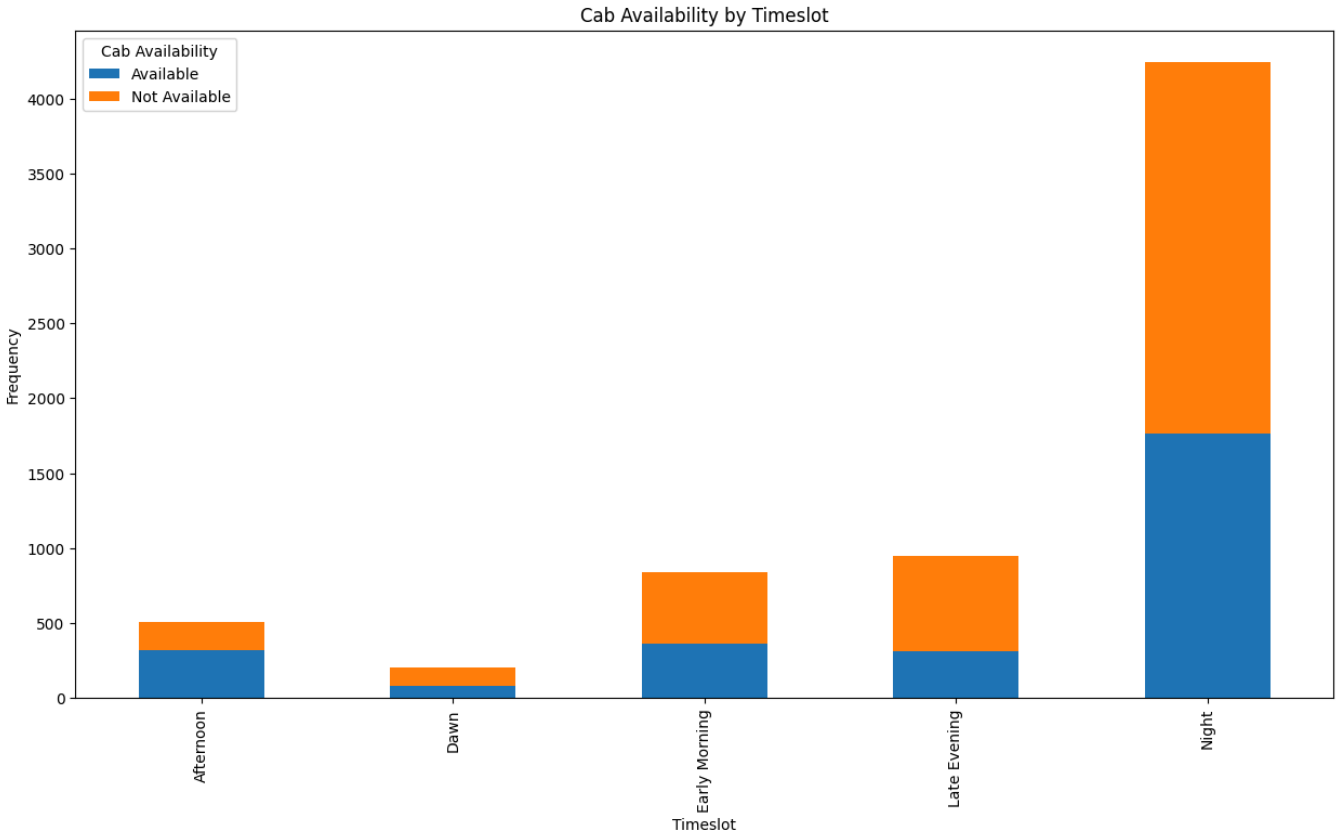


Observation:

There is very high demand for cabs from City to Airport between 5:00 AM – 9:00 AM But the supply is very less primarily due to Ride Cancellations

```
# Time slots where highest gap exists -
df.groupby(['Timeslot','Cab Availability']).size().unstack().plot(kind = 'bar',stacked = True,figsize = (15,8))
plt.title('Cab Availability by Timeslot')
plt.xlabel('Timeslot')
plt.ylabel('Frequency')
```

```
Text(0, 0.5, 'Frequency')
```



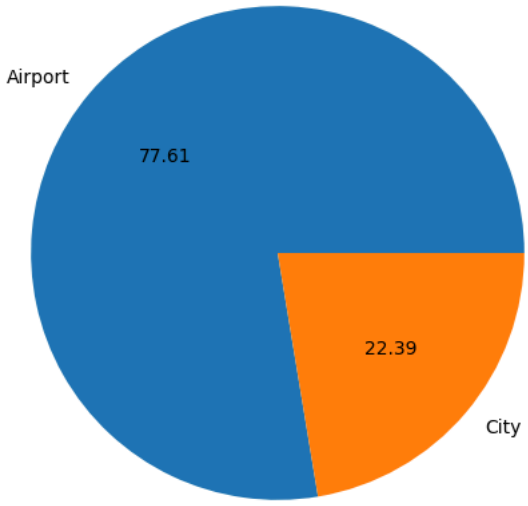
Observation:

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

```
#  Types of Requests (city-airport or airport-city) for which the gap is the most severe in the identified time slots -

df[df['Timeslot'] == 'Late Evening'].groupby(['Pickup point']).size().plot(kind = 'pie',stacked = True,figsize = (6,6),autopct = '%.2f'
plt.ylabel('')
```

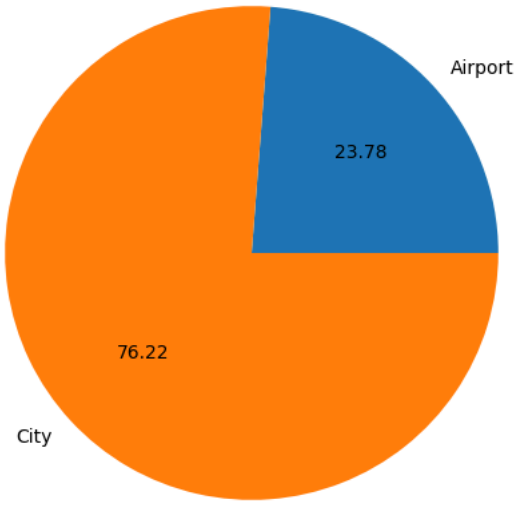
```
Text(0, 0.5, '')
```



```
#  Types of Requests (city-airport or airport-city) for which the gap is the most severe in the identified time slots -

df[df['Timeslot'] == 'Early Morning'].groupby(['Pickup point']).size().plot(kind = 'pie',stacked = True,figsize = (6,6),autopct = '%.2f
plt.ylabel('')
```

```
Text(0, 0.5, '')
```



Reason for Supply-Demand gap :

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

Recommendations :

1. Awarding incentive for waiting time will encourage the drivers to wait at Airport.
2. Drivers could be compensated for taking the night shifts hence covering the 00:00 – 5:00 time slot.
3. Seeing this analysis trends, few cabs could be placed in Airports proactively.
4. Drivers to be rewarded for the Airport rides making up for the loss in time.
5. The cab discovery range to be increased for Airport location, so that the search for cabs would be on a wider range.