Uber Supply-Demand Gap

The aim of analysis is to identify the root cause of the problem (i.e. cancellation and non-availability of cars) and recommend ways to improve the situation. As a result of our analysis, we should be able to present to the client the root cause and possible hypotheses of the problem and recommend ways to improve them.

There are six attributes associated with each request made by a customer:

Request id: A unique identifier of the request

Time of request: The date and time at which the customer made the trip request

Drop-off time: The drop-off date and time, in case the trip was completed

Pick-up point: The point from which the request was made

Driver id: The unique identification number of the driver

Status of the request: The final status of the trip, that can be either completed, cancelled by the driver or no cars available

For this assignment, only the trips to and from the airport are being considered.

1) The following code will load the data into dataframe and prints first 5 rows

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

Out[3]:

	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

2) The following code will request for formatting timestap and drops timestamps

In [7]:
```
# formatting request timestamp and drop timestamp columns
uber_data["Request timestamp"] = pd.to_datetime(uber_data["Request timestamp"], dayfirst=True)
uber_data["Drop timestamp"] = pd.to_datetime(uber_data["Drop timestamp"], dayfirst=True)
```

In [8]:
```
# printing first few row of the formatted data
uber_data.head()
```

Out[8]:

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

3)The following code will analyse the given data and prints the few rows

In [9]:
```
# creating derived metrics from timestamp columns for further analysis and printing top few rows out of it
uber_data['Request Date'] = uber_data["Request timestamp"].dt.date
uber_data['Request Time'] = uber_data["Request timestamp"].dt.time
uber_data['Drop Date'] = uber_data["Drop timestamp"].dt.date
uber_data['Drop Time'] = uber_data["Drop timestamp"].dt.time
uber_data['Request Weekday'] = uber_data['Request timestamp'].apply(lambda x: dt.datetime.strftime(x, '%A'))
uber_data['Request Hour'] = uber_data['Request timestamp'].apply(lambda x: x.hour)
uber_data.head()
```

Out[9]:

	Request id	Pickup point	Driver id	Status	Request timestamp	Drop timestamp	Request Date	Request Time	Drop Date	Drop Time	Request Weekday	Request Hour
0	619	Airport	1.0	Trip Completed	2016-07-11 11:51:00	2016-07-11 13:00:00	2016-07-11	11:51:00	2016-07-11	13:00:00	Monday	11
1	867	Airport	1.0	Trip Completed	2016-07-11 17:57:00	2016-07-11 18:47:00	2016-07-11	17:57:00	2016-07-11	18:47:00	Monday	17
2	1807	City	1.0	Trip Completed	2016-07-12 09:17:00	2016-07-12 09:58:00	2016-07-12	09:17:00	2016-07-12	09:58:00	Tuesday	9
3	2532	Airport	1.0	Trip Completed	2016-07-12 21:08:00	2016-07-12 22:03:00	2016-07-12	21:08:00	2016-07-12	22:03:00	Tuesday	21
4	3112	City	1.0	Trip Completed	2016-07-13 08:33:16	2016-07-13 09:25:47	2016-07-13	08:33:16	2016-07-13	09:25:47	Wednesday	8

4)The following code will group data by driver id and status type

In [14]:
```
# grouping by driver id and status type and printing top 10 rows
uber_data_groupby_status_and_driverId = uber_data.groupby(["Driver id","Status"]).count()
uber_data_groupby_status_and_driverId.head(10)
```

Out[14]:

Driver id	Status	Request id	Pickup point	Request timestamp	Drop timestamp	Request Date	Request Time	Drop Date	Drop Time	Request Weekday	Request Hour
1.0	Cancelled	4	4	4	0	4	4	0	0	4	4
	Trip Completed	9	9	9	9	9	9	9	9	9	9
2.0	Cancelled	4	4	4	0	4	4	0	0	4	4
	Trip Completed	9	9	9	9	9	9	9	9	9	9
3.0	Cancelled	4	4	4	0	4	4	0	0	4	4
	Trip Completed	10	10	10	10	10	10	10	10	10	10
4.0	Cancelled	5	5	5	0	5	5	0	0	5	5
	Trip Completed	10	10	10	10	10	10	10	10	10	10
5.0	Cancelled	2	2	2	0	2	2	0	0	2	2
	Trip Completed	11	11	11	11	11	11	11	11	11	11

5) The following code filter the dataframe for which trip is either cancelled or no cab was available

In [15]:
```
# filtering the dataframe for which trip is either cancelled or no cab was available
uber_data_with_null_drop_timestamp = uber_data[uber_data["Drop timestamp"].isnull()]
uber_data_with_null_drop_timestamp.head()
```

Out[15]:

	Request id	Pickup point	Driver id	Status	Request timestamp	Drop timestamp	Request Date	Request Time	Drop Date	Drop Time	Request Weekday	Request Hour
2831	2905	City	1.0	Cancelled	2016-07-13 06:08:41	NaT	2016-07-13	06:08:41	NaT	NaT	Wednesday	6
2832	4805	City	1.0	Cancelled	2016-07-14 17:07:58	NaT	2016-07-14	17:07:58	NaT	NaT	Thursday	17
2833	5202	Airport	1.0	Cancelled	2016-07-14 20:51:37	NaT	2016-07-14	20:51:37	NaT	NaT	Thursday	20
2834	5927	City	1.0	Cancelled	2016-07-15 10:12:40	NaT	2016-07-15	10:12:40	NaT	NaT	Friday	10
2835	2347	Airport	2.0	Cancelled	2016-07-12 19:14:00	NaT	2016-07-12	19:14:00	NaT	NaT	Tuesday	19

6) The following will filter the dataframe for which trip has been completed

In [17]:
```
# filtering dataframe for which trip was completed
uber_data_with_drop_timestamp = uber_data[~uber_data["Drop timestamp"].isnull()]
uber_data_with_drop_timestamp.head()
```

Out[17]:

	Request id	Pickup point	Driver id	Status	Request timestamp	Drop timestamp	Request Date	Request Time	Drop Date	Drop Time	Request Weekday	Request Hour
0	619	Airport	1.0	Trip Completed	2016-07-11 11:51:00	2016-07-11 13:00:00	2016-07-11	11:51:00	2016-07-11	13:00:00	Monday	11
1	867	Airport	1.0	Trip Completed	2016-07-11 17:57:00	2016-07-11 18:47:00	2016-07-11	17:57:00	2016-07-11	18:47:00	Monday	17
2	1807	City	1.0	Trip Completed	2016-07-12 09:17:00	2016-07-12 09:58:00	2016-07-12	09:17:00	2016-07-12	09:58:00	Tuesday	9
3	2532	Airport	1.0	Trip Completed	2016-07-12 21:08:00	2016-07-12 22:03:00	2016-07-12	21:08:00	2016-07-12	22:03:00	Tuesday	21
4	3112	City	1.0	Trip Completed	2016-07-13 08:33:16	2016-07-13 09:25:47	2016-07-13	08:33:16	2016-07-13	09:25:47	Wednesday	8

7) The following will filter dataframe for the requests for which no cars were available

In [19]:
```
# filtering dataframe for the requests for which no cars were available
uber_data_with_no_driver_id = uber_data[uber_data["Driver id"].isnull()]
uber_data_with_no_driver_id.head()
```

Out[19]:

	Request id	Pickup point	Driver id	Status	Request timestamp	Drop timestamp	Request Date	Request Time	Drop Date	Drop Time	Request Weekday	Request Hour
4095	1362	City	NaN	No Cars Available	2016-07-11 00:02:00	NaT	2016-07-11	00:02:00	NaT	NaT	Monday	0
4096	1364	City	NaN	No Cars Available	2016-07-11 00:06:00	NaT	2016-07-11	00:06:00	NaT	NaT	Monday	0
4097	1366	City	NaN	No Cars Available	2016-07-11 00:09:00	NaT	2016-07-11	00:09:00	NaT	NaT	Monday	0
4098	2	Airport	NaN	No Cars Available	2016-07-11 00:23:00	NaT	2016-07-11	00:23:00	NaT	NaT	Monday	0
4099	7	Airport	NaN	No Cars Available	2016-07-11 00:30:00	NaT	2016-07-11	00:30:00	NaT	NaT	Monday	0

8) The following will filter dataframe for which driver id was present, so either the trip was cancelled or completed

In [20]:
```
# filtering dataframe for which driver id was present, so either the trip was cancelled or completed
uber_data_with_driver_id = uber_data[~uber_data["Driver id"].isnull()]
uber_data_with_driver_id.head()
```

Out[20]:

	Request id	Pickup point	Driver id	Status	Request timestamp	Drop timestamp	Request Date	Request Time	Drop Date	Drop Time	Request Weekday	Request Hour
0	619	Airport	1.0	Trip Completed	2016-07-11 11:51:00	2016-07-11 13:00:00	2016-07-11	11:51:00	2016-07-11	13:00:00	Monday	11
1	867	Airport	1.0	Trip Completed	2016-07-11 17:57:00	2016-07-11 18:47:00	2016-07-11	17:57:00	2016-07-11	18:47:00	Monday	17
2	1807	City	1.0	Trip Completed	2016-07-12 09:17:00	2016-07-12 09:58:00	2016-07-12	09:17:00	2016-07-12	09:58:00	Tuesday	9
3	2532	Airport	1.0	Trip Completed	2016-07-12 21:08:00	2016-07-12 22:03:00	2016-07-12	21:08:00	2016-07-12	22:03:00	Tuesday	21
4	3112	City	1.0	Trip Completed	2016-07-13 08:33:16	2016-07-13 09:25:47	2016-07-13	08:33:16	2016-07-13	09:25:47	Wednesday	8

9) the code will group by status to check total count of trip completed and cancelled

In [21]:
```
# grouping by status to check total count of trip completed and cancelledD
uber_data_with_driver_id.groupby(uber_data_with_driver_id.Status).count()
```

Out[21]:

Status	Request id	Pickup point	Driver id	Request timestamp	Drop timestamp	Request Date	Request Time	Drop Date	Drop Time	Request Weekday	Request Hour
Cancelled	1264	1264	1264	1264	0	1264	1264	0	0	1264	1264
Trip Completed	2831	2831	2831	2831	2831	2831	2831	2831	2831	2831	2831

In [22]:
```
print("Number of rows with driver id but null Drop timestamp:", uber_data_with_driver_id["Drop timestamp"].isnull().sum())
```

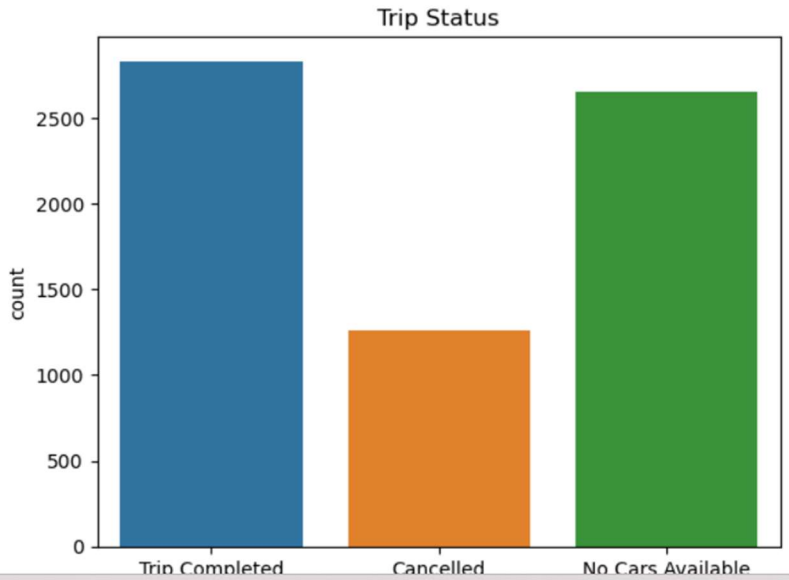
```
Number of rows with driver id but null Drop timestamp: 1264
```

10) plotting graph for trip status

In [28]:
```
# plotting graph for trip status
plt.title('Trip Status')
sns.countplot(uber_data['Status'])
```

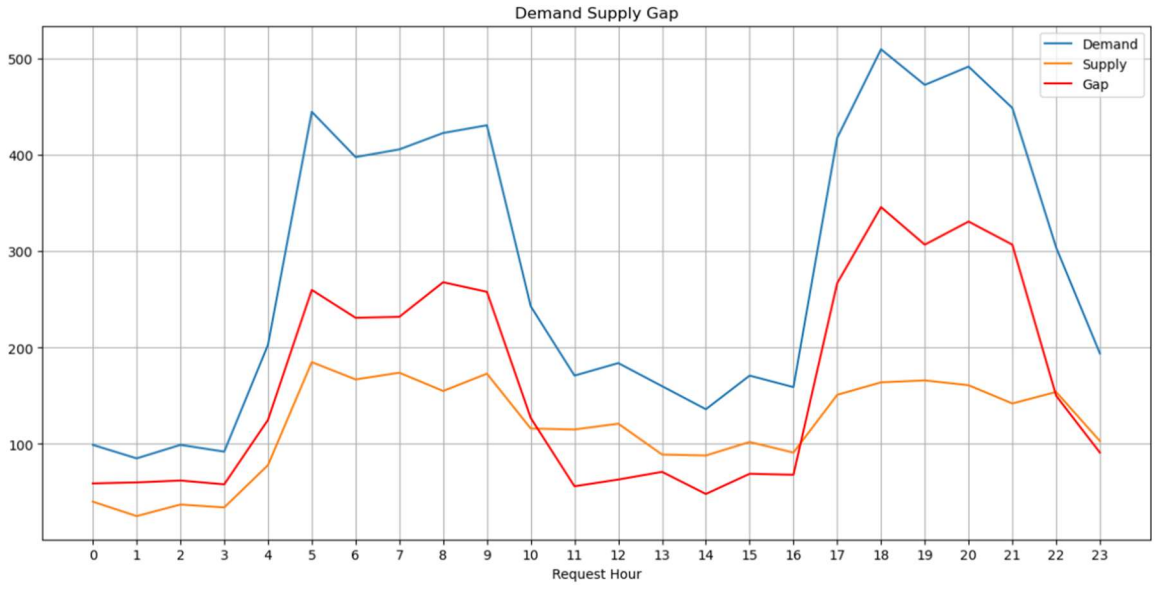
```
C:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as
ersion 0.12, the only valid positional argument will be `data`, and passing other arguments without ar
sult in an error or misinterpretation.
  warnings.warn(
```

Out[28]: <AxesSubplot:title={'center':'Trip Status'}, xlabel='Status', ylabel='count'>



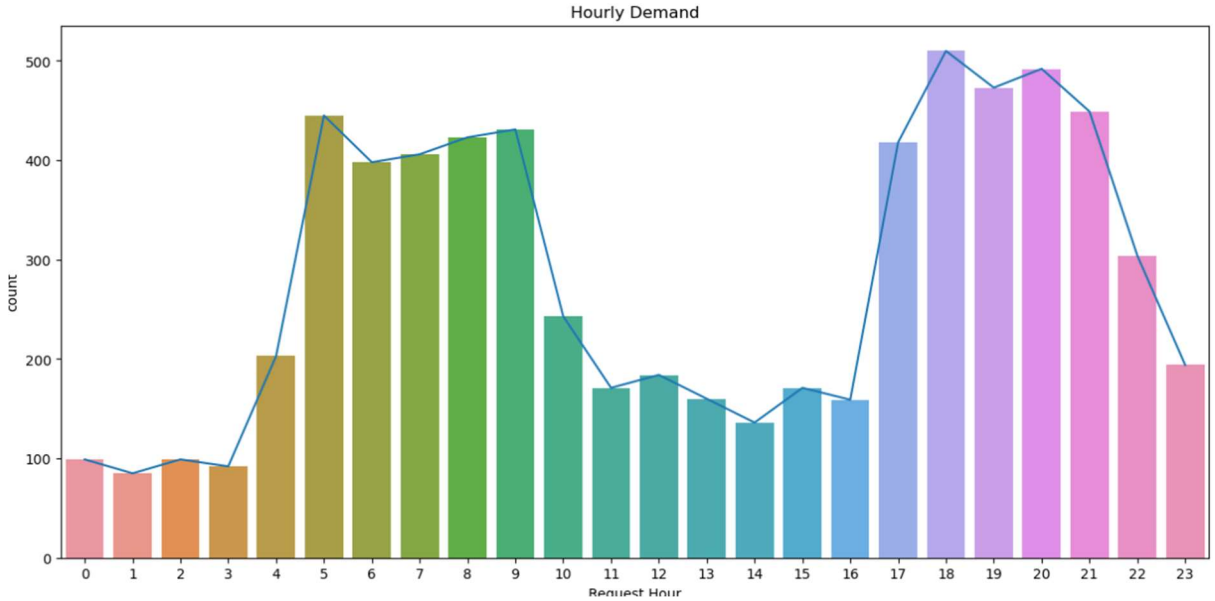
11) the following code will plot data to show demand supply graph

In [29]:
```
#plot data to show demand supply gap
fig, ax = plt.subplots(figsize=(15,7))
plt.xticks([i for i in range(0,24)])
uber_data.groupby(uber_data['Request Hour']).count()['Request id'].plot(ax=ax, label='Demand')
uber_data[uber_data['Status']=="Trip Completed"].groupby(uber_data[uber_data['Status']=="Trip Completed"]['Request Hour']).c
(uber_data.groupby(uber_data['Request Hour']).count()['Request id'] - uber_data[uber_data['Status']=="Trip Completed"].group
plt.grid("on")
plt.title("Demand Supply Gap")
plt.legend()
```



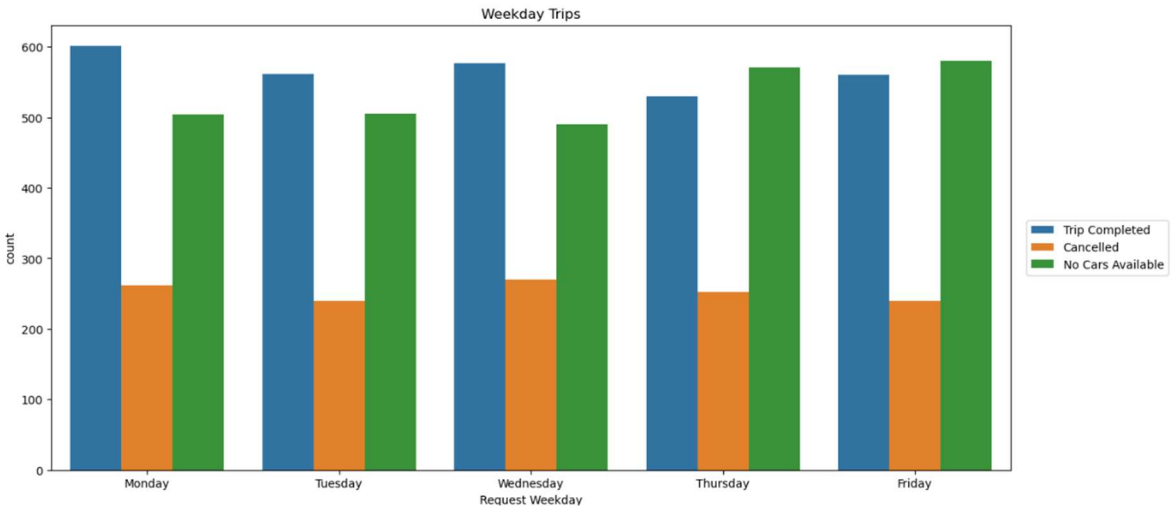
12) plot data to show hourly distribution of request

In [30]:
```
# plot data to show hourly distribution of request
fig, ax = plt.subplots(figsize=(15,7))
plt.title('Hourly Demand')
uber_data.groupby(uber_data['Request Hour']).count()['Request id'].plot(ax=ax)
sns.countplot(uber_data['Request Hour'])
```



13) plot data to show status count for each day

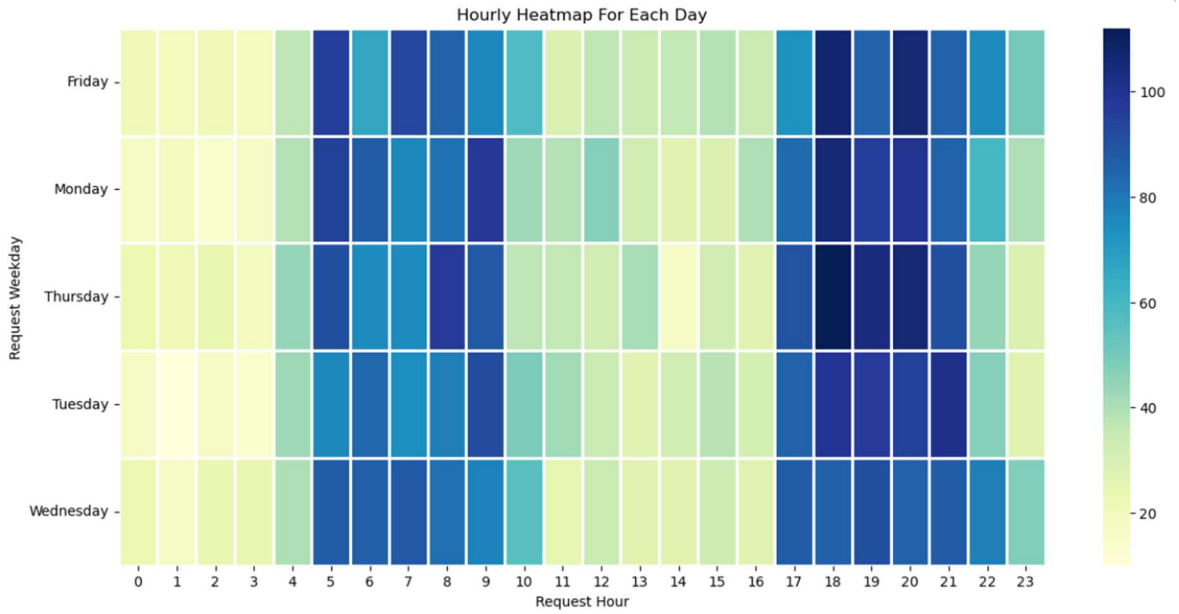
In [31]:
```
# plot data to show status count for each day
fig, ax = plt.subplots(figsize=(15,7))
plt.title("Weekday Trips")
ax = sns.countplot(x='Request Weekday',hue='Status',data=uber_data)
plt.legend(bbox_to_anchor=(1.17, 0.5), loc='right')
```



14) plot data to show hourly request heatmap

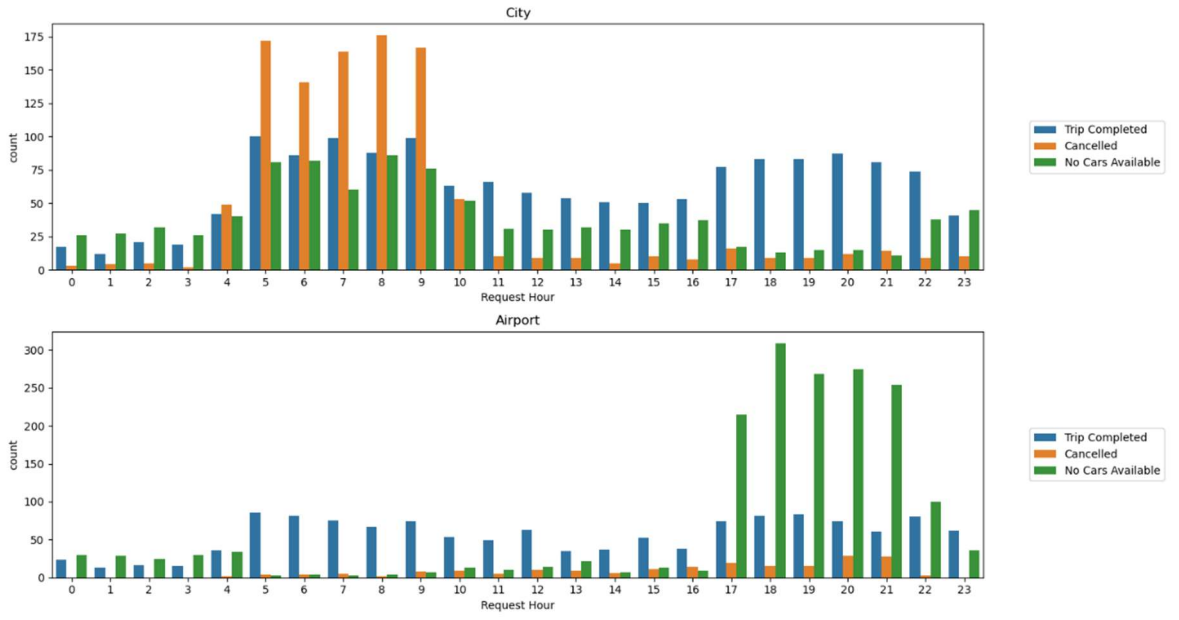
In [33]:
```
# plot data to show hourly request heatmap
fig, ax = plt.subplots(figsize=(15,7))

plt.title('Hourly Heatmap For Each Day')
sns.heatmap(uber_data_heat.pivot('Request Weekday', 'Request Hour', 'Hourly Count'), linewidths=1, cmap="YlGnBu")
plt.yticks(rotation=0)
plt.show()
```



15) plot data to get count of status for airport to city and city to airport trips

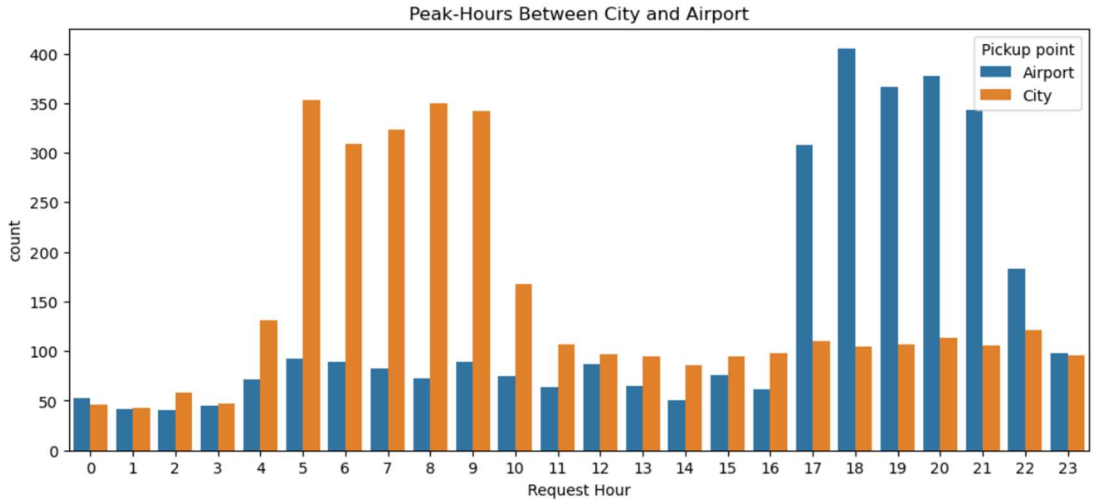
In [35]:
```
# plot data to get count of status for airport to city and city to airport trips
f = plt.figure(figsize=(15,8))
plt.subplot(2,1,1)
plt.title("City")
sns.countplot(x='Request Hour',hue='Status',data=uber_data_city)
plt.legend(bbox_to_anchor=(1.20, 0.5), loc='right')
plt.subplot(2,1,2)
plt.title("Airport")
sns.countplot(x='Request Hour',hue='Status',data=uber_data_airport)
plt.legend(bbox_to_anchor=(1.20, 0.5), loc='right')
plt.tight_layout()
```



16) plot hourly data to visualize peakhours from city to airport and airport to city trip

In [36]:
```
# plot hourly data to visualize peakhours from city to airport and airport to city trip
plt.subplots(figsize=(12,5))
plt.title("Peak-Hours Between City and Airport")
sns.countplot(x='Request Hour',hue='Pickup point',data=uber_data)
```

Out[36]: <AxesSubplot:title={'center':'Peak-Hours Between City and Airport'}, xlabel='Request Hour', ylabel='count'>



17) The final data analysis of the data