Jupyter Notebook for Data Demand-Supply analysis for Bolt Company

Done by Nato Jorjiashvili

In [1]:

```python
#import libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

In [2]:

```python
#reading data
demand_df = pd.read_excel('Demand Data.xlsx')
supply_df = pd.read_excel('Supply Data.xlsx')

demand_df['Date'] = pd.to_datetime(demand_df['Date'])
supply_df['Date'] = pd.to_datetime(supply_df['Date'])
```

In [3]:

```python
# Check for missing values
print(demand_df.isnull().sum())
print(supply_df.isnull().sum())

# Check data types
print(demand_df.dtypes)
print(supply_df.dtypes)
```

```
Date                           0
People saw 0 cars (unique)     0
People saw +1 cars (unique)    0
Coverage Ratio (%)             0
dtype: int64
Date                        0
Active drivers              0
Online (h)                  0
Has booking (h)             0
Waiting for booking (h)     0
Hours per active driver     0
Rides per online hour       0
Finished Rides             45
dtype: int64
Date                           datetime64[ns]
People saw 0 cars (unique)              int64
People saw +1 cars (unique)             int64
Coverage Ratio (%)                      int64
dtype: object
Date                       datetime64[ns]
Active drivers                      int64
Online (h)                          int64
Has booking (h)                     int64
Waiting for booking (h)             int64
Hours per active driver           float64
Rides per online hour             float64
Finished Rides                    float64
dtype: object
```

In [4]:

```
missing_data1=demand_df.isnull()
missing_data1.head(5)
```

Out[4]:

	Date	People saw 0 cars (unique)	People saw +1 cars (unique)	Coverage Ratio (%)
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	False	False

In [5]:

```
missing_data2=supply_df.isnull()
missing_data2.head(5)

for column in missing_data2.columns.values.tolist():
    print(column)
    print(missing_data2[column].value_counts())
    print("")
```

```
Date
False    840
Name: Date, dtype: int64

Active drivers
False    840
Name: Active drivers, dtype: int64

Online (h)
False    840
Name: Online (h), dtype: int64

Has booking (h)
False    840
Name: Has booking (h), dtype: int64

Waiting for booking (h)
False    840
Name: Waiting for booking (h), dtype: int64

Hours per active driver
False    840
Name: Hours per active driver, dtype: int64

Rides per online hour
False    840
Name: Rides per online hour, dtype: int64

Finished Rides
False    795
True      45
Name: Finished Rides, dtype: int64
```

In [6]:

```
supply_df["Finished Rides"].replace(np.nan, 0)
```

Out[6]:

```
0      12.0
1      28.0
2      16.0
3      15.0
4      36.0
       ... 
835     0.0
836     0.0
837     1.0
838     2.0
839     6.0
Name: Finished Rides, Length: 840, dtype: float64
```

In [7]:

```
missing_data2=supply_df.isnull()
missing_data2.head(5)

for column in missing_data2.columns.values.tolist():
    print(column)
    print(missing_data2[column].value_counts())
    print("")
```

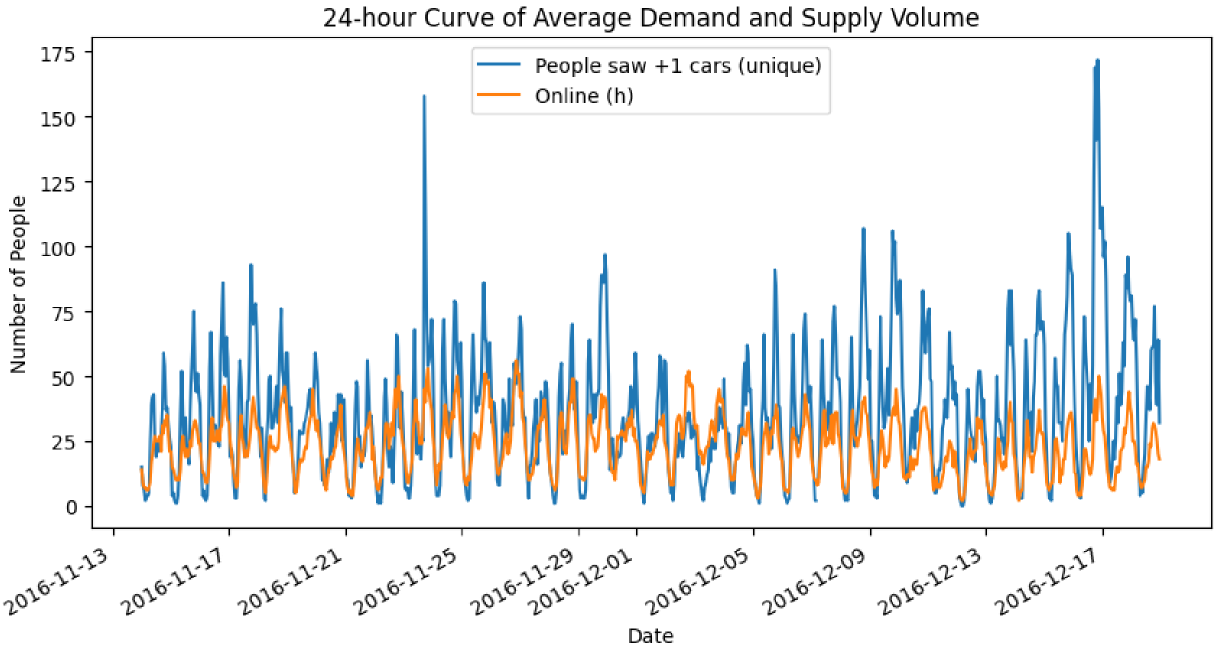
```
Date
False    840
Name: Date, dtype: int64

Active drivers
False    840
Name: Active drivers, dtype: int64

Online (h)
False    840
Name: Online (h), dtype: int64

Has booking (h)
False    840
Name: Has booking (h), dtype: int64

Waiting for booking (h)
False    840
Name: Waiting for booking (h), dtype: int64

Hours per active driver
False    840
Name: Hours per active driver, dtype: int64

Rides per online hour
False    840
Name: Rides per online hour, dtype: int64

Finished Rides
False    795
True      45
Name: Finished Rides, dtype: int64
```

In [8]:

```
# Question 1: Identify the time periods that are critical to us
# Create a new column in the supply data to calculate the average number of finished r
supply_df['Avg Finished Rides'] = supply_df['Finished Rides'] / supply_df['Online (h)'

# Create a new data frame to merge the demand and supply data
merged_data = pd.merge(demand_df, supply_df, on='Date', how='outer')
```

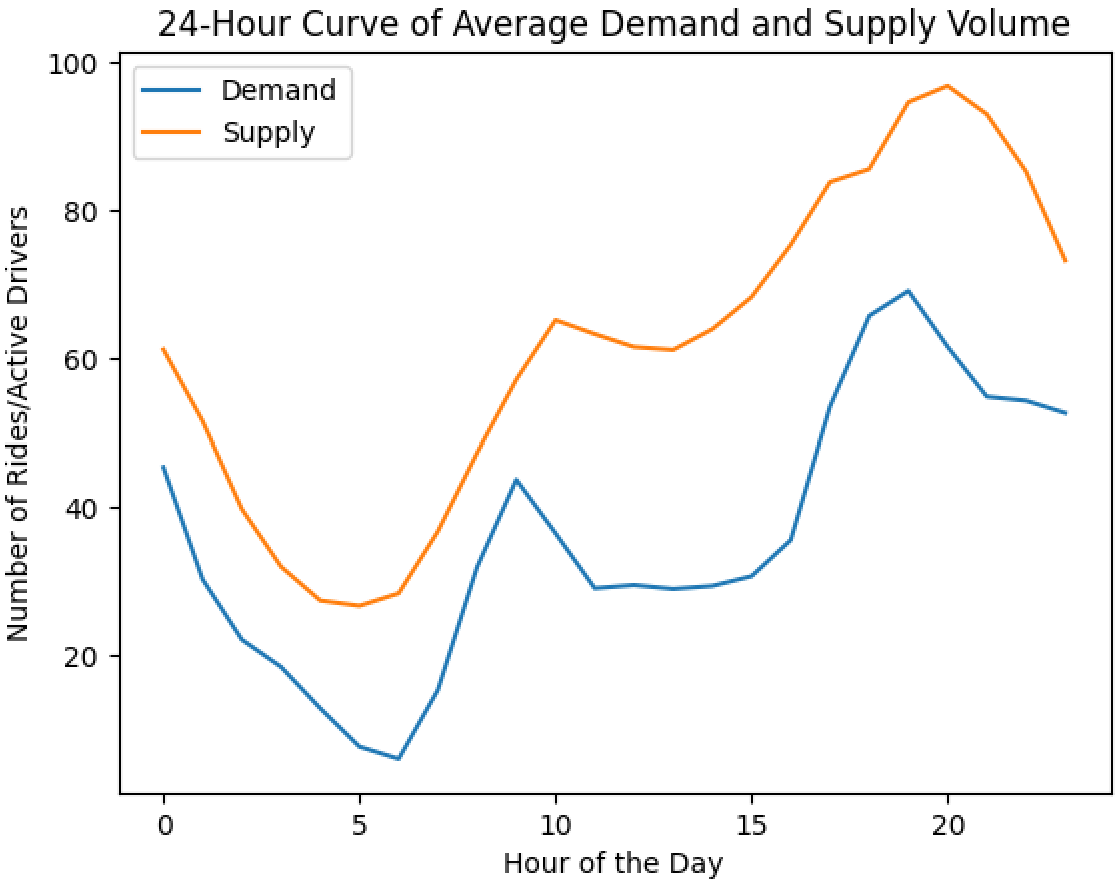
In [9]:

```
# Plot the 24-hour curve of average demand and supply volume
merged_data.plot(x='Date', y=['People saw +1 cars (unique)', 'Online (h)'], kind='line
plt.xlabel('Date')
plt.ylabel('Number of People')
plt.title('24-hour Curve of Average Demand and Supply Volume')
plt.show()
```



In [10]:

```python
# Calculate the average demand and supply volume for each hour of the day
merged_data["Hour"] = pd.to_datetime(merged_data["Date"]).dt.hour
hourly_demand = merged_data.groupby("Hour")["People saw +1 cars (unique)"].mean()
hourly_supply = merged_data.groupby("Hour")["Active drivers"].mean()

# Plot a 24-hour curve
plt.plot(hourly_demand.index, hourly_demand, label="Demand")
plt.plot(hourly_supply.index, hourly_supply, label="Supply")
plt.title("24-Hour Curve of Average Demand and Supply Volume")
plt.xlabel("Hour of the Day")
plt.ylabel("Number of Rides/Active Drivers")
plt.legend()
plt.show()
```



In [11]:

```
merged_data['Day'] = merged_data['Date'].dt.day_name()
merged_data.columns
```

Out[11]:

```
Index(['Date', 'People saw 0 cars (unique)', 'People saw +1 cars (unique)',
       'Coverage Ratio (%)', 'Active drivers', 'Online (h)', 'Has booking (h)',
       'Waiting for booking (h)', 'Hours per active driver',
       'Rides per online hour', 'Finished Rides', 'Avg Finished Rides', 'Hour',
       'Day'],
      dtype='object')
```

In [12]:

```
# Identify undersupplied hours during a weekly period (Monday to Sunday)
merged_data['day_of_week'] = merged_data['Date'].dt.day_name()

merged_data.head()
```

Out[12]:

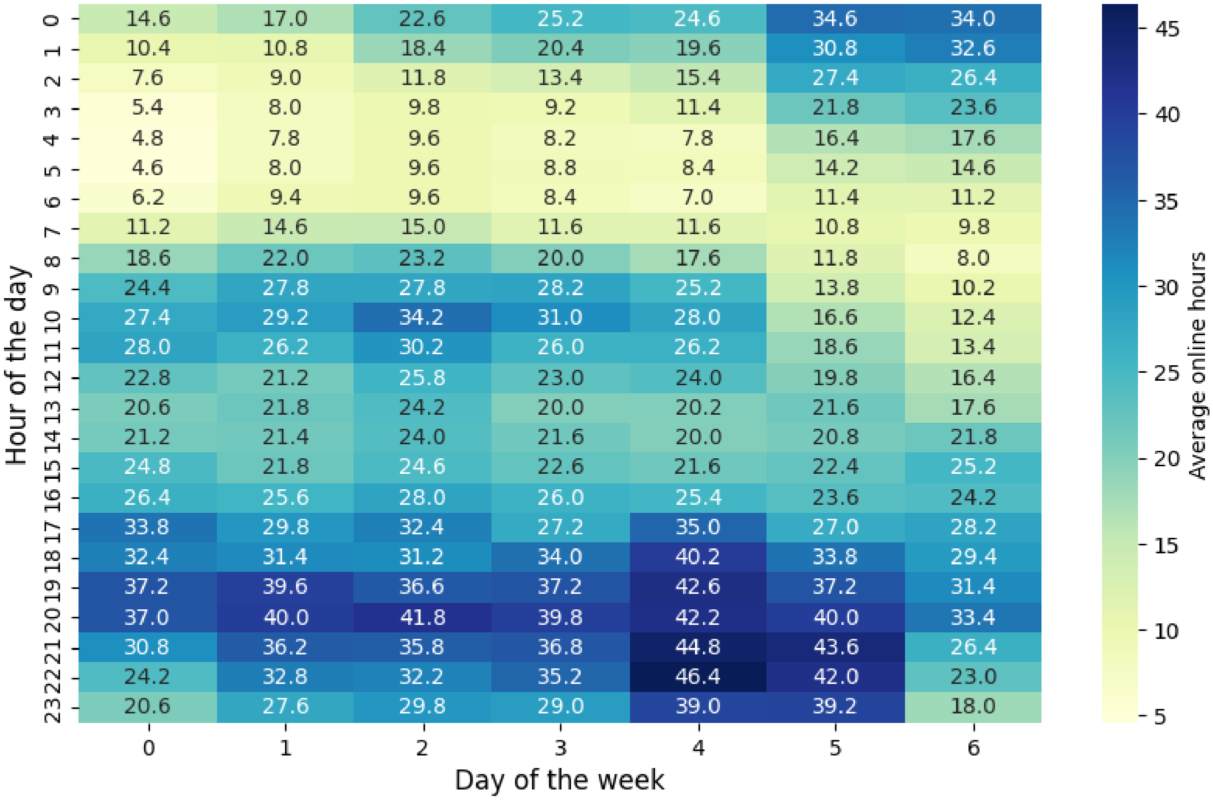
	Date	People saw 0 cars (unique)	People saw +1 cars (unique)	Coverage Ratio (%)	Active drivers	Online (h)	Has booking (h)	Waiting for booking (h)	Hours per active driver	Rides per online hour	Finishe Ride
0	2016-12-18 23:00:00	9.0	32.0	78.0	52	18	6	11	0.3	0.67	12.(
1	2016-12-18 22:00:00	29.0	64.0	69.0	59	20	11	9	0.3	1.40	28.(
2	2016-12-18 21:00:00	5.0	39.0	89.0	72	25	7	18	0.3	0.64	16.(
3	2016-12-18 20:00:00	13.0	48.0	79.0	86	29	7	23	0.3	0.52	15.(
4	2016-12-18 19:00:00	12.0	77.0	87.0	82	31	14	17	0.4	1.16	36.(

In [13]:

```
supply_df['Date'] = pd.to_datetime(supply_df['Date'])
supply_df['Hour'] = supply_df['Date'].dt.hour
supply_hourly = supply_df.groupby('Hour').mean()[['Active drivers', 'Online (h)']]

supply_df['Weekday'] = supply_df['Date'].dt.weekday
supply_df_weekly = supply_df.groupby(['Weekday', 'Hour']).mean()[['Active drivers', 'O
supply_df_weekly = supply_df.groupby(['Weekday', 'Hour']).mean()[['Active drivers', 'O
```

In [14]:

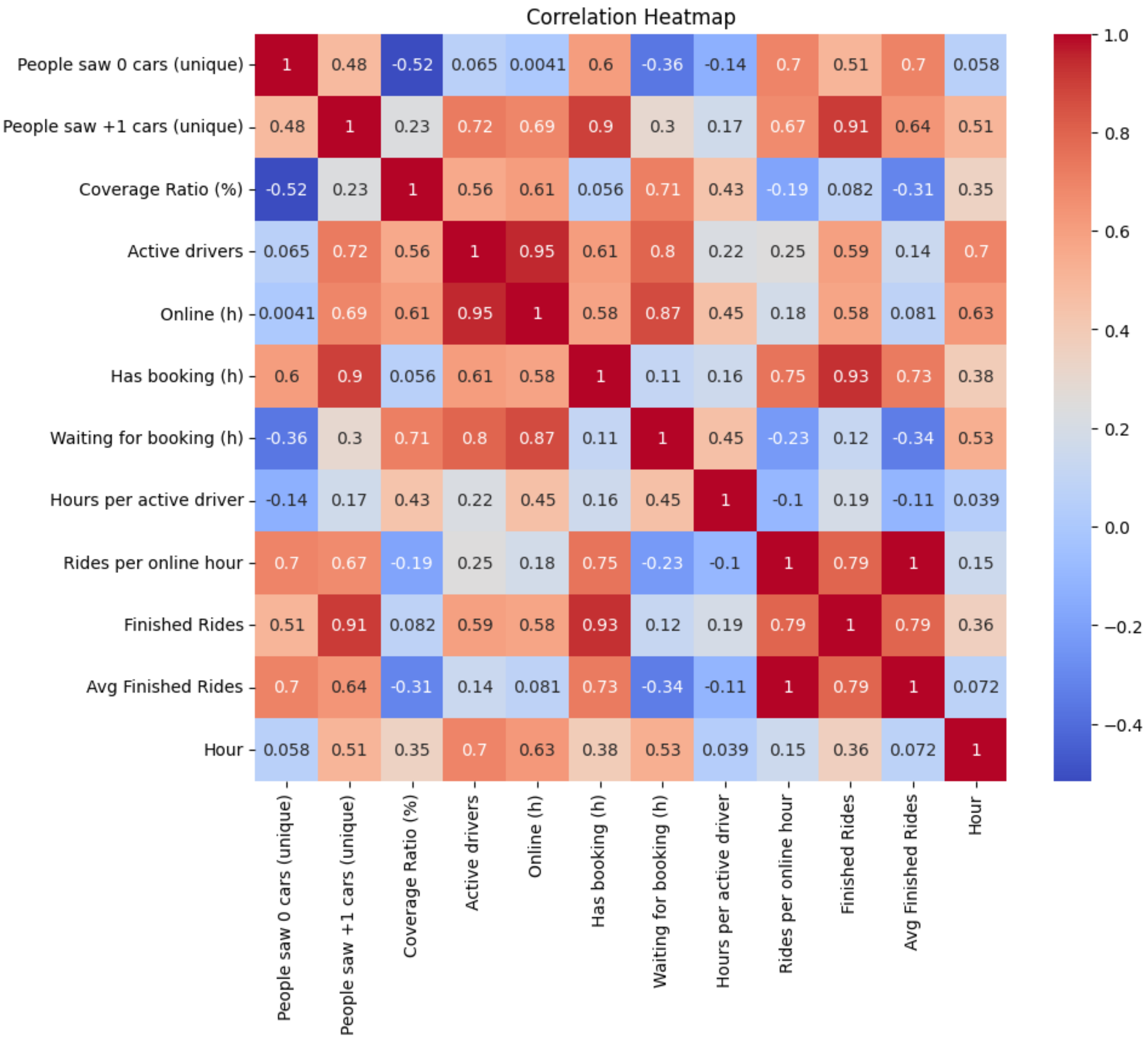
```
fig, ax = plt.subplots(figsize=(10, 6))
sns.heatmap(supply_df_weekly['Online (h)'].unstack().T, cmap='YlGnBu', annot=True, fmt

# Set axis labels and tick labels
ax.set_xlabel('Day of the week', fontsize=12)
ax.set_ylabel('Hour of the day', fontsize=12)
ax.tick_params(axis='both', which='major', labelsize=10)

# Rotate x-axis tick labels
ax.tick_params(axis='x', rotation=0)
```



Hour of the day	0	1	2	3	4	5	6
0	14.6	17.0	22.6	25.2	24.6	34.6	34.0
1	10.4	10.8	18.4	20.4	19.6	30.8	32.6
2	7.6	9.0	11.8	13.4	15.4	27.4	26.4
3	5.4	8.0	9.8	9.2	11.4	21.8	23.6
4	4.8	7.8	9.6	8.2	7.8	16.4	17.6
5	4.6	8.0	9.6	8.8	8.4	14.2	14.6
6	6.2	9.4	9.6	8.4	7.0	11.4	11.2
7	11.2	14.6	15.0	11.6	11.6	10.8	9.8
8	18.6	22.0	23.2	20.0	17.6	11.8	8.0
9	24.4	27.8	27.8	28.2	25.2	13.8	10.2
10	27.4	29.2	34.2	31.0	28.0	16.6	12.4
11	28.0	26.2	30.2	26.0	26.2	18.6	13.4
12	22.8	21.2	25.8	23.0	24.0	19.8	16.4
13	20.6	21.8	24.2	20.0	20.2	21.6	17.6
14	21.2	21.4	24.0	21.6	20.0	20.8	21.8
15	24.8	21.8	24.6	22.6	21.6	22.4	25.2
16	26.4	25.6	28.0	26.0	25.4	23.6	24.2
17	33.8	29.8	32.4	27.2	35.0	27.0	28.2
18	32.4	31.4	31.2	34.0	40.2	33.8	29.4
19	37.2	39.6	36.6	37.2	42.6	37.2	31.4
20	37.0	40.0	41.8	39.8	42.2	40.0	33.4
21	30.8	36.2	35.8	36.8	44.8	43.6	26.4
22	24.2	32.8	32.2	35.2	46.4	42.0	23.0
23	20.6	27.6	29.8	29.0	39.0	39.2	18.0

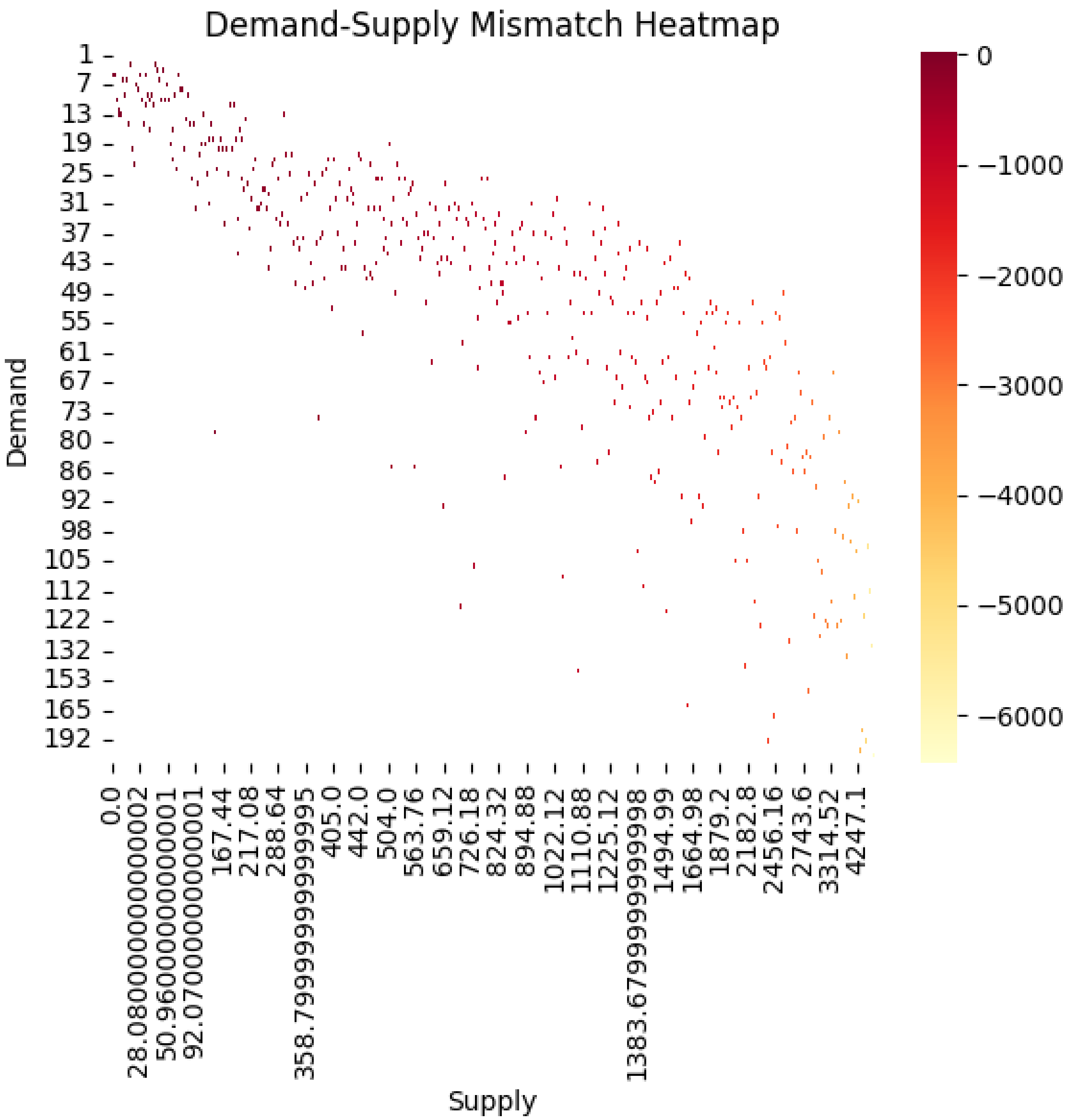
In [15]:

```
# Compute the correlation matrix
corr = merged_data.corr()

# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```



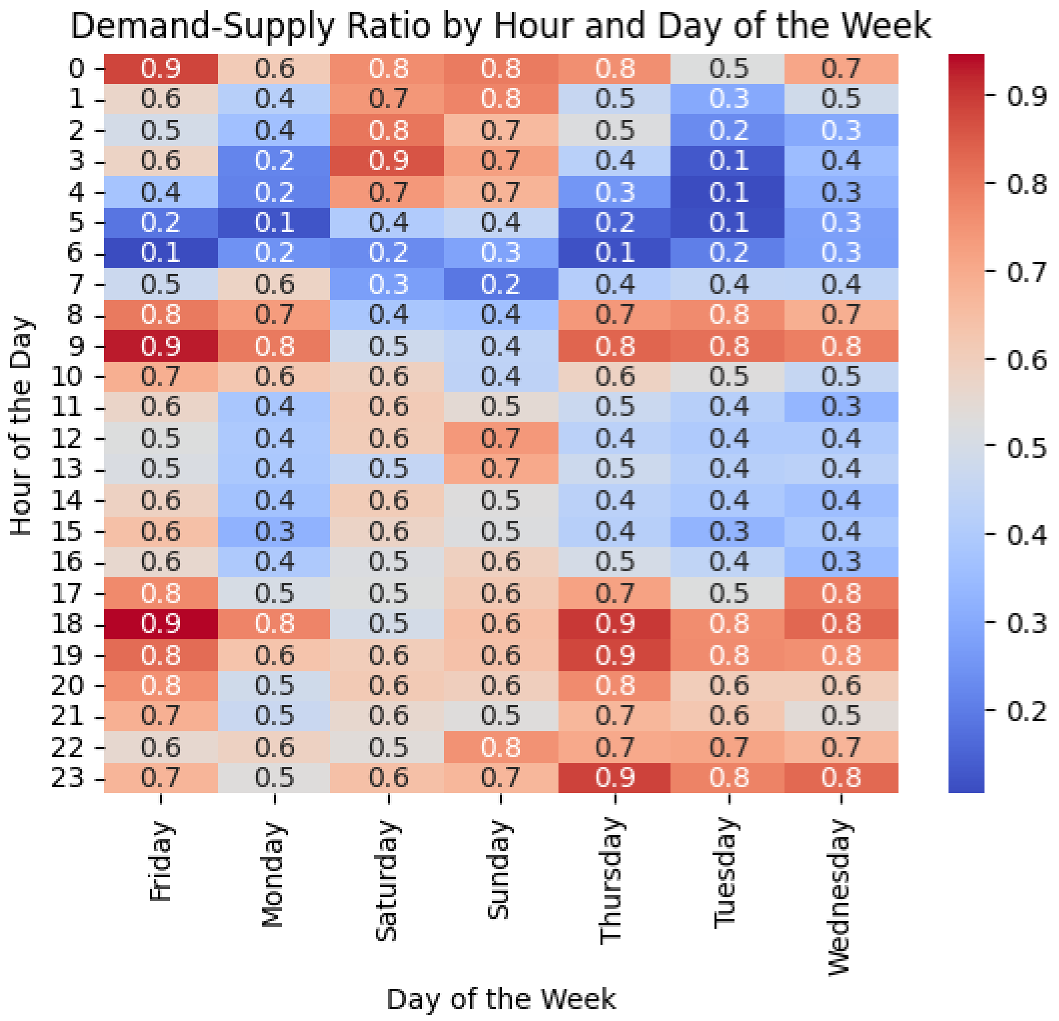
In [16]:

```
# Rename columns for better clarity
demand_df = demand_df.rename(columns={"People saw 0 cars (unique)": "Demand 0 cars",
                                      "People saw +1 cars (unique)": "Demand +1 ca
                                      "Coverage Ratio (%)": "Coverage Ratio"})
supply_df = supply_df.rename(columns={"Online (h)": "Online hours",
                                      "Has booking (h)": "Has booking hours",
                                      "Waiting for booking (h)": "Waiting for book
                                      "Hours per active driver": "Hours per driver
                                      "Rides per online hour": "Rides per hour"})

# Merge demand and supply data on date
demand_supply_data = pd.merge(demand_df, supply_df, on="Date")

# Calculate the demand-supply mismatch
demand_supply_data["Demand"] = demand_supply_data["Demand 0 cars"] + demand_supply_dat
demand_supply_data["Supply"] = demand_supply_data["Active drivers"] * demand_supply_da
demand_supply_data["Mismatch"] = demand_supply_data["Demand"] - demand_supply_data["Su

# Create a heatmap of demand-supply mismatch
heatmap_data = pd.pivot_table(demand_supply_data, values="Mismatch", index="Demand", c
sns.heatmap(heatmap_data, cmap="YlOrRd")
plt.title("Demand-Supply Mismatch Heatmap")
plt.show()
```



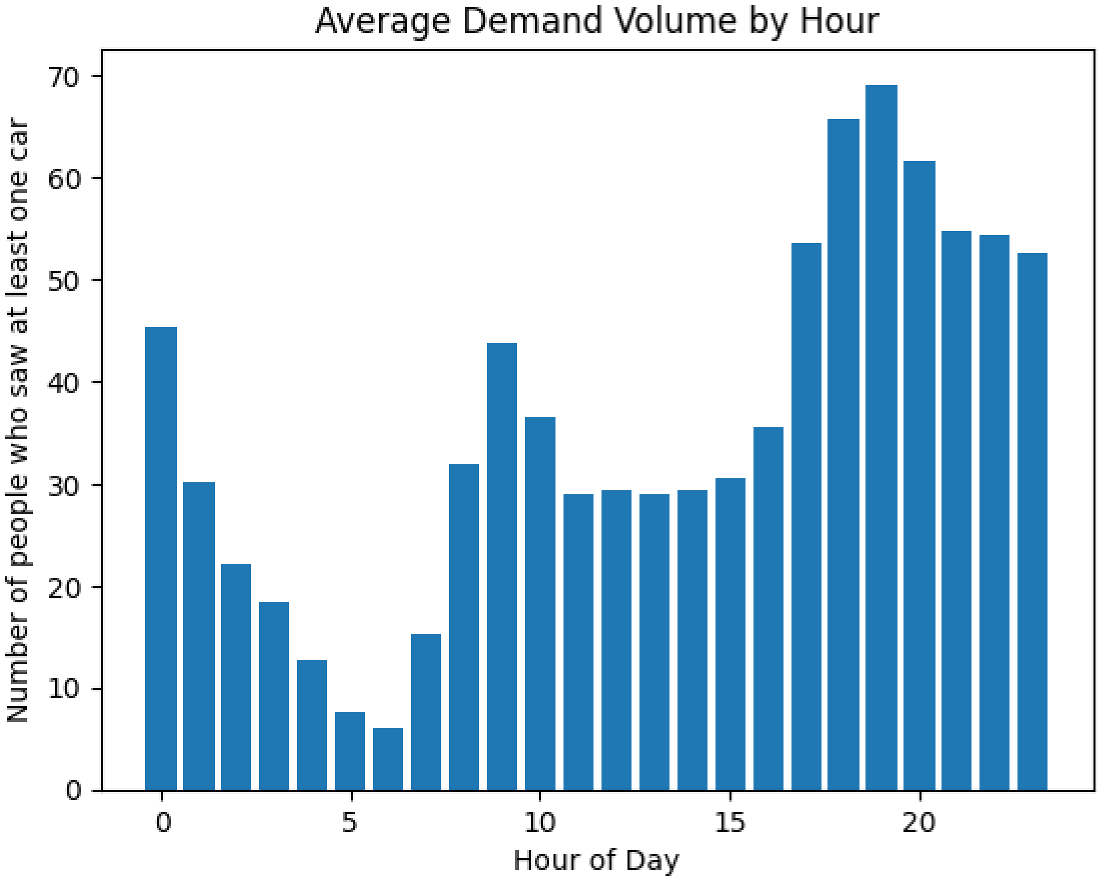
In [17]:

```
# Create a pivot table with the average demand and supply volume for each hour and day
hourly_pivot = merged_data.pivot_table(values=["People saw +1 cars (unique)", "Active

# Calculate the demand-supply ratio
ratio = hourly_pivot["People saw +1 cars (unique)"] / hourly_pivot["Active drivers"]

# Create a heatmap
sns.heatmap(ratio, cmap="coolwarm", annot=True, fmt=".1f")
plt.title("Demand-Supply Ratio by Hour and Day of the Week")
plt.xlabel("Day of the Week")
plt.ylabel("Hour of the Day")
plt.show()
```



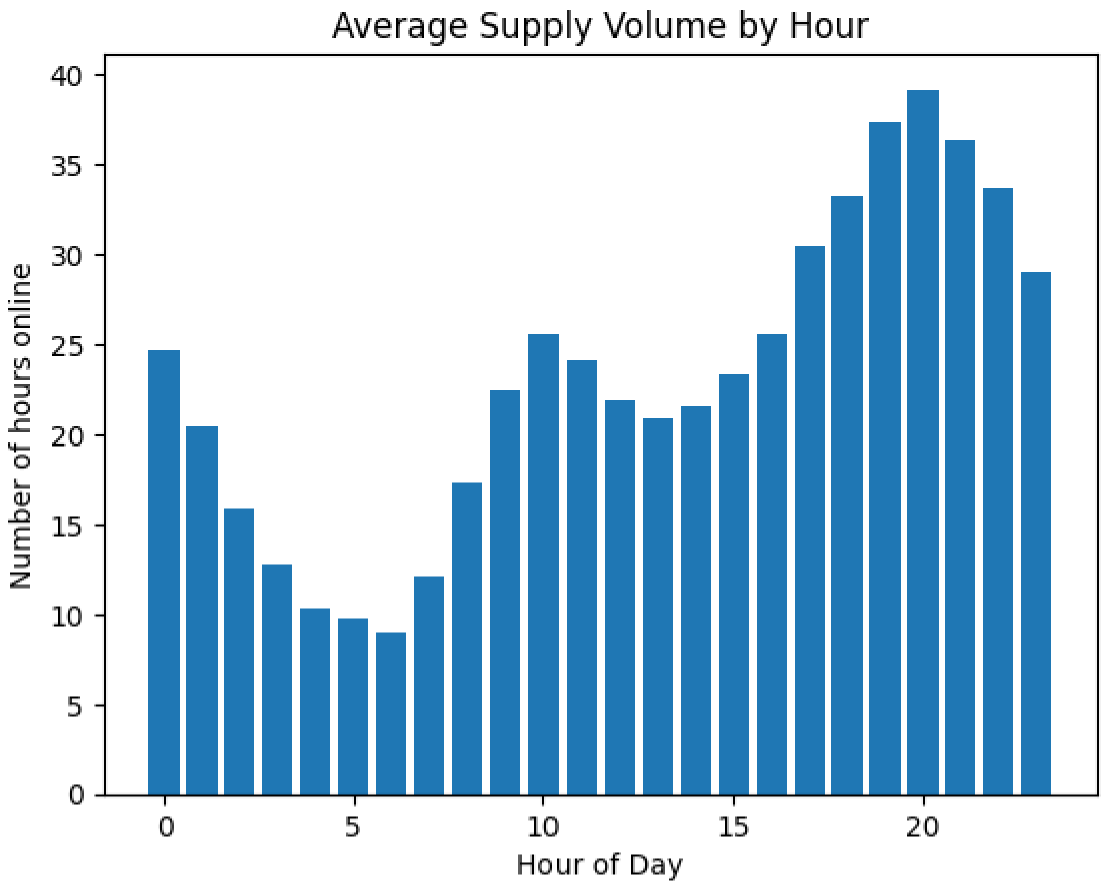
In [18]:

```
# Calculate the average demand and supply volume over a 24-hour period
merged_data['Hour'] = merged_data['Date'].dt.hour
demand_df_avg = merged_data.groupby('Hour')['People saw +1 cars (unique)'].mean()
supply_df_avg = merged_data.groupby('Hour')['Online (h)'].mean()

# Create the average demand volume graph
plt.bar(demand_df_avg.index, demand_df_avg)
plt.title('Average Demand Volume by Hour')
plt.xlabel('Hour of Day')
plt.ylabel('Number of people who saw at least one car')
plt.show()
```



In [19]:

```
# Create the average supply volume graph
plt.bar(supply_df_avg.index, supply_df_avg)
plt.title('Average Supply Volume by Hour')
plt.xlabel('Hour of Day')
plt.ylabel('Number of hours online')
plt.show()
```



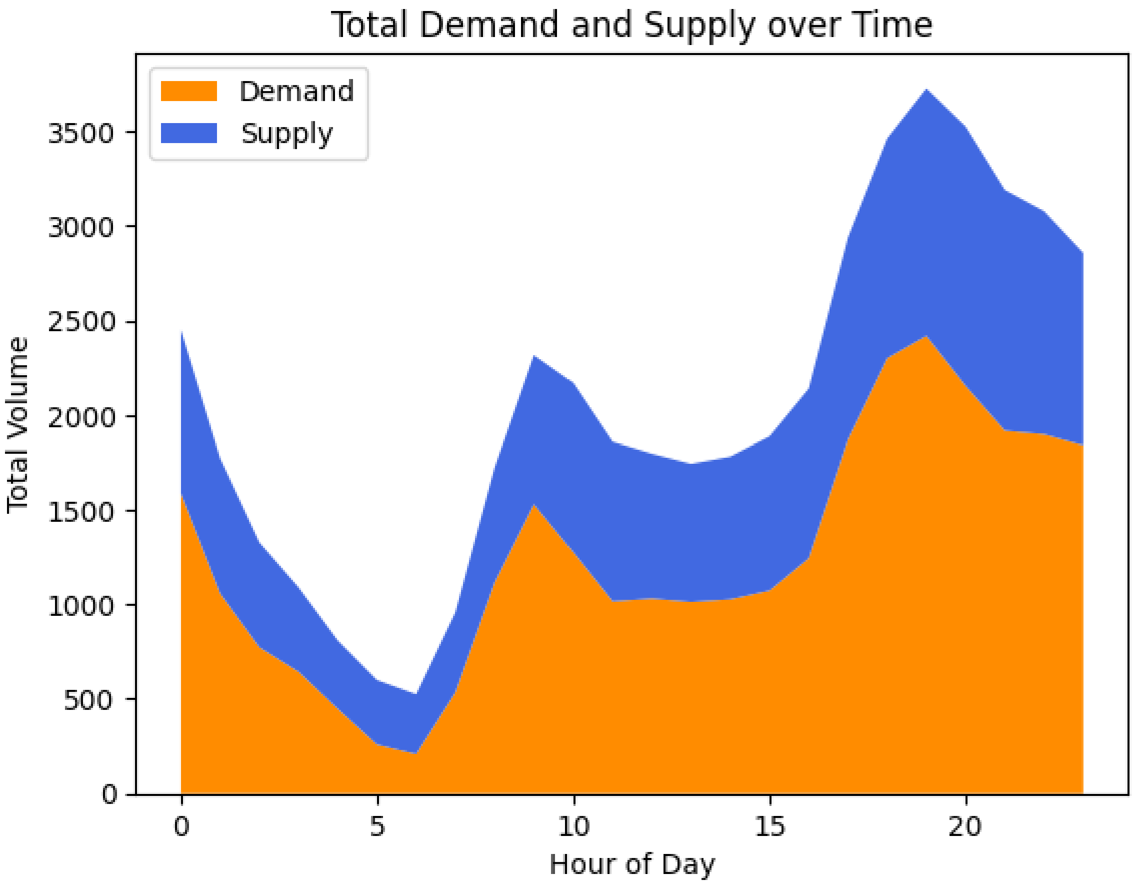
In [20]:
```
merged_data.head()
```

Out[20]:

	Date	People saw 0 cars (unique)	People saw +1 cars (unique)	Coverage Ratio (%)	Active drivers	Online (h)	Has booking (h)	Waiting for booking (h)	Hours per active driver	Rides per online hour	Finishe Ride
0	2016-12-18 23:00:00	9.0	32.0	78.0	52	18	6	11	0.3	0.67	12.(
1	2016-12-18 22:00:00	29.0	64.0	69.0	59	20	11	9	0.3	1.40	28.(
2	2016-12-18 21:00:00	5.0	39.0	89.0	72	25	7	18	0.3	0.64	16.(
3	2016-12-18 20:00:00	13.0	48.0	79.0	86	29	7	23	0.3	0.52	15.(
4	2016-12-18 19:00:00	12.0	77.0	87.0	82	31	14	17	0.4	1.16	36.(

In [21]:
```
# Calculate the total demand and supply volume over a 24-hour period
df_demand_total = merged_data.groupby('Hour')['People saw +1 cars (unique)'].sum()
```

```
df_supply_total = merged_data.groupby('Hour')['Online (h)'].sum()

# Create the stacked area chart
plt.stackplot(df_demand_total.index, df_demand_total, df_supply_total, labels=['Demand
plt.legend(loc='upper left')
plt.title('Total Demand and Supply over Time')
plt.xlabel('Hour of Day')
plt.ylabel('Total Volume')
plt.show()
```

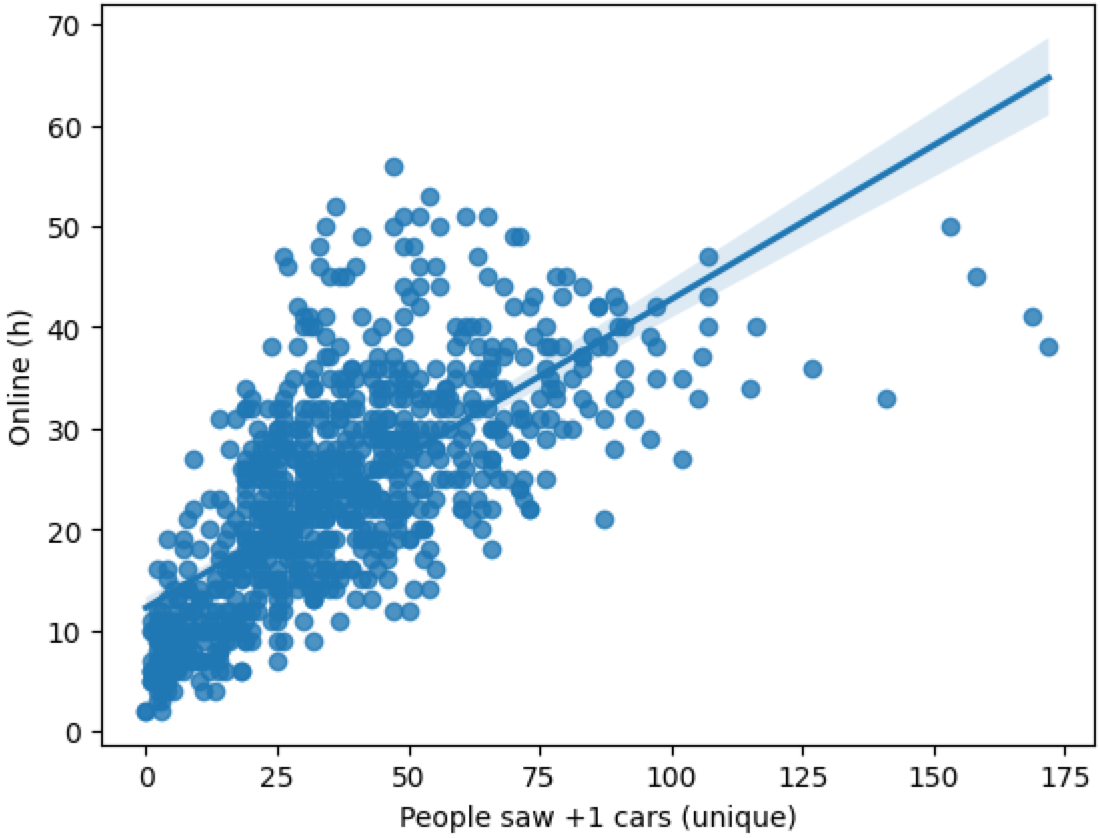


In [22]:

```
sns.regplot(x="People saw +1 cars (unique)",y="Online (h)", data=merged_data)
```

Out[22]:

```
<AxesSubplot:xlabel='People saw +1 cars (unique)', ylabel='Online (h)'>
```



In [23]:

```
# Question 2: Calculate the number of online hours required to ensure good coverage ra
peak_hours = merged_data[(merged_data['People saw 0 cars (unique)'] > merged_data['Peo

#peak_hours.head()
required_online_hours = peak_hours['People saw +1 cars (unique)'].sum() / peak_hours['
print('Number of Online Hours Required for Good Coverage Ratio During Peak Hours:', re
```

```
Number of Online Hours Required for Good Coverage Ratio During Peak Hours: 2322.89177
79339426
```

In [24]:

```
# Question 3: Calculate the earning we can guarantee to attract more supply during pea
revenue = (0.2 * peak_hours['People saw +1 cars (unique)'].sum() * 10) / peak_hours['O
print('Revenue During Peak Hours:', revenue)
guaranteed_earning = (0.8 * peak_hours['People saw +1 cars (unique)'].sum() * 10) / pe
print('Earning We Can Guarantee to Attract More Supply During Peak Hours:', guaranteed
```

```
Revenue During Peak Hours: 3.857142857142857
Earning We Can Guarantee to Attract More Supply During Peak Hours: 15.428571428571429
```