Uber Basic Data Analysis

This notebook contains a basic analysis through some visualizations of the Uber Pickups in New York City data set.

The analysis is broken up into 3 sections:

- Data Loading and Preparation.
- Exploration and visualization of pickups from April to September 2014.
- Conclusion.

1. Data Loading and Preparation

1.1 Loading Modules

In [1]:
```
import pandas as pd
import numpy as np

#Visualization modules
import matplotlib.pyplot as plt
import seaborn as sns


%matplotlib inline
from matplotlib import cm #Colormap
```

1.2 Loading Data

In [2]:
```
#Load the datasets

df_apr14=pd.read_csv("data/uber-raw-data-apr14.csv")
df_may14=pd.read_csv("data/uber-raw-data-may14.csv")
df_jun14=pd.read_csv("data/uber-raw-data-jun14.csv")
df_jul14=pd.read_csv("data/uber-raw-data-jul14.csv")
df_aug14=pd.read_csv("data/uber-raw-data-aug14.csv")
df_sep14=pd.read_csv("data/uber-raw-data-sep14.csv")
```

In [3]:
```
#Merge the dataframes into one
df = pd.concat([df_apr14, df_may14, df_jun14, df_jul14, df_aug14, df_sep14], ignore
```

1.3 Data Preparation

In [4]:
```
df.head()
```

Out[4]:

	Date/Time	Lat	Lon	Base
0	4/1/2014 0:11:00	40.7690	-73.9549	B02512
1	4/1/2014 0:17:00	40.7267	-74.0345	B02512
2	4/1/2014 0:21:00	40.7316	-73.9873	B02512
3	4/1/2014 0:28:00	40.7588	-73.9776	B02512
4	4/1/2014 0:33:00	40.7594	-73.9722	B02512

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4534327 entries, 0 to 4534326
Data columns (total 4 columns):
 #   Column     Dtype  
---  ------     -----  
 0   Date/Time  object 
 1   Lat        float64
 2   Lon        float64
 3   Base       object 
dtypes: float64(2), object(2)
memory usage: 138.4+ MB
```

In [6]:

```
# Renaming the Date/Time Column
df = df.rename(columns={'Date/Time': 'Date_time'})

# Converting the Date_time type into Datetime
df['Date_time'] = pd.to_datetime(df['Date_time'])

# Adding useful columns
df['Month'] = df['Date_time'].dt.month_name()
df['Weekday'] = df['Date_time'].dt.day_name()
df['Day'] = df['Date_time'].dt.day
df['Hour'] = df['Date_time'].dt.hour
df['Minute'] = df['Date_time'].dt.minute
```

In [7]:

```
df.head()
```

Out[7]:

	Date_time	Lat	Lon	Base	Month	Weekday	Day	Hour	Minute
0	2014-04-01 00:11:00	40.7690	-73.9549	B02512	April	Tuesday	1	0	11
1	2014-04-01 00:17:00	40.7267	-74.0345	B02512	April	Tuesday	1	0	17
2	2014-04-01 00:21:00	40.7316	-73.9873	B02512	April	Tuesday	1	0	21
3	2014-04-01 00:28:00	40.7588	-73.9776	B02512	April	Tuesday	1	0	28
4	2014-04-01 00:33:00	40.7594	-73.9722	B02512	April	Tuesday	1	0	33

In [8]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4534327 entries, 0 to 4534326
Data columns (total 9 columns):
 #   Column     Dtype         
---  ------     -----         
 0   Date_time  datetime64[ns]
 1   Lat        float64       
 2   Lon        float64       
 3   Base       object        
 4   Month      object        
 5   Weekday    object        
 6   Day        int32         
 7   Hour       int32         
 8   Minute     int32         
dtypes: datetime64[ns](1), float64(2), int32(3), object(3)
memory usage: 259.5+ MB
```

In [9]:

```
df.describe(include = 'all')
```

Out[9]:

	Date_time	Lat	Lon	Base	Month	Weekday	
count	4534327	4.534327e+06	4.534327e+06	4534327	4534327	4534327	4
unique	NaN	NaN	NaN	5	6	7	
top	NaN	NaN	NaN	B02617	September	Thursday	
freq	NaN	NaN	NaN	1458853	1028136	755145	
mean	2014-07-11 18:50:50.578151424	4.073926e+01	-7.397302e+01	NaN	NaN	NaN	1
min	2014-04-01 00:00:00	3.965690e+01	-7.492900e+01	NaN	NaN	NaN	1
25%	2014-05-28 15:18:00	4.072110e+01	-7.399650e+01	NaN	NaN	NaN	9
50%	2014-07-17 14:45:00	4.074220e+01	-7.398340e+01	NaN	NaN	NaN	1
75%	2014-08-27 21:55:00	4.076100e+01	-7.396530e+01	NaN	NaN	NaN	2
max	2014-09-30 22:59:00	4.211660e+01	-7.206660e+01	NaN	NaN	NaN	3
std	NaN	3.994991e-02	5.726670e-02	NaN	NaN	NaN	8

2 Exploration and Visualization

Through our exploration we are going to visualize and analyse:

- The number of trips by hour
- The number of trips by month
- The number of trips by weekday
- The number of trips by day
- The number of trips by hour and month
- The number of trips by weekday and hour
- The number of trips by weekday and month

2.1 Trips by hour

In [10]:

```
#Grouping by Hour
df_hour_grouped = df.groupby(['Hour']).count()

#Creating the sub dataframe
```

```
df_hour = pd.DataFrame({'Number_of_trips':df_hour_grouped.values[:,0]}, index = df_

df_hour.head()
```

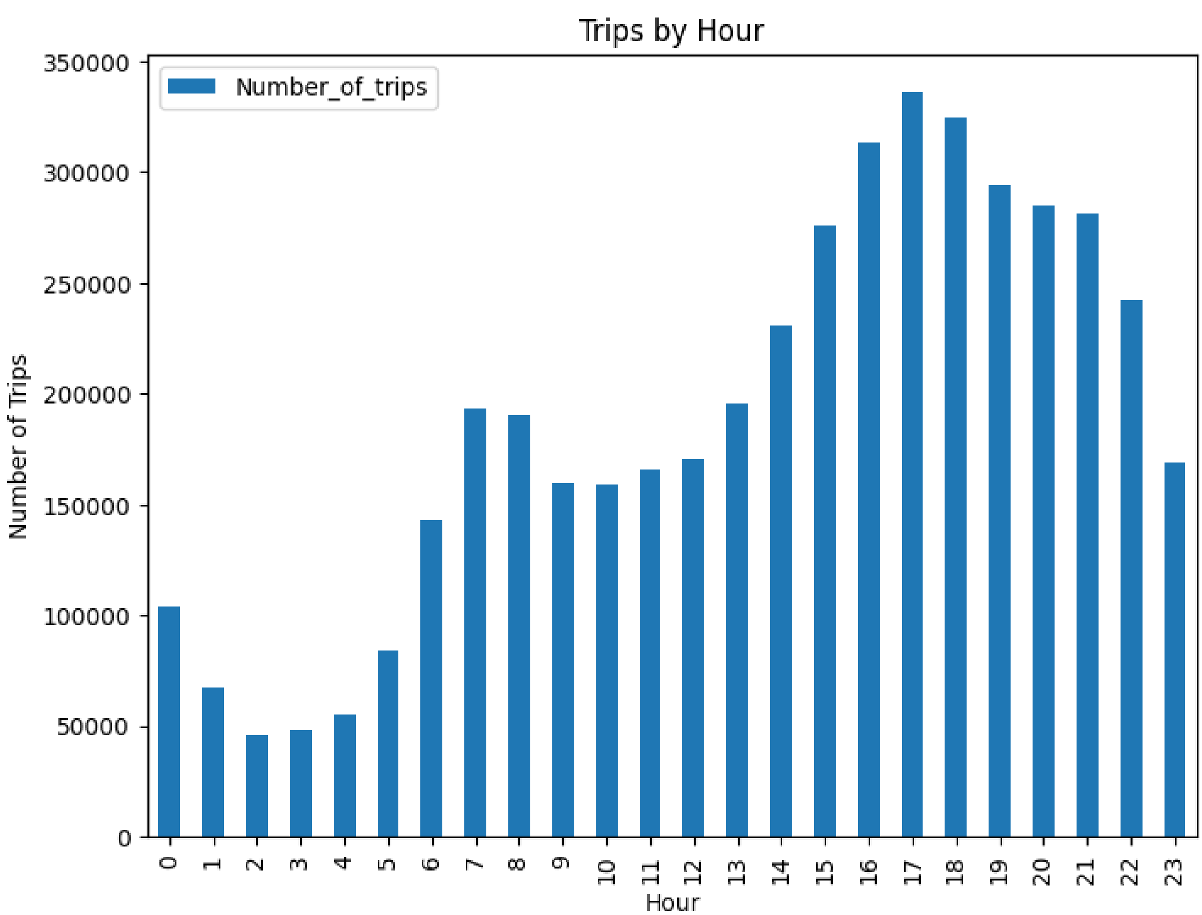
Out[10]:

	Number_of_trips
Hour	
0	103836
1	67227
2	45865
3	48287
4	55230

Plotting the results

In [11]:

```
df_hour.plot(kind='bar', figsize=(8,6))

plt.ylabel('Number of Trips')
plt.title('Trips by Hour')

plt.show()
```



Analysing the results

In [12]:
```
#The highest number of trips by hour
max_Number_of_trips_hour = max(df_hour['Number_of_trips'])
max_hour = df_hour[df_hour['Number_of_trips'] == 336190].index[0]

print('The highest number of trips by hour is {} trip, that corresponds to the peak
```

```
The highest number of trips by hour is 336190 trip, that corresponds to the peak hou
r 17:00.
```

We observe that the number of trips are higher around 16:00 and 18:00, with a spike at 17:00. It matches the end of a working day in the United States (16:30), the time when the workers go home.

We can say that the majority of Uber's clients are workers.

2.2 Trips by month

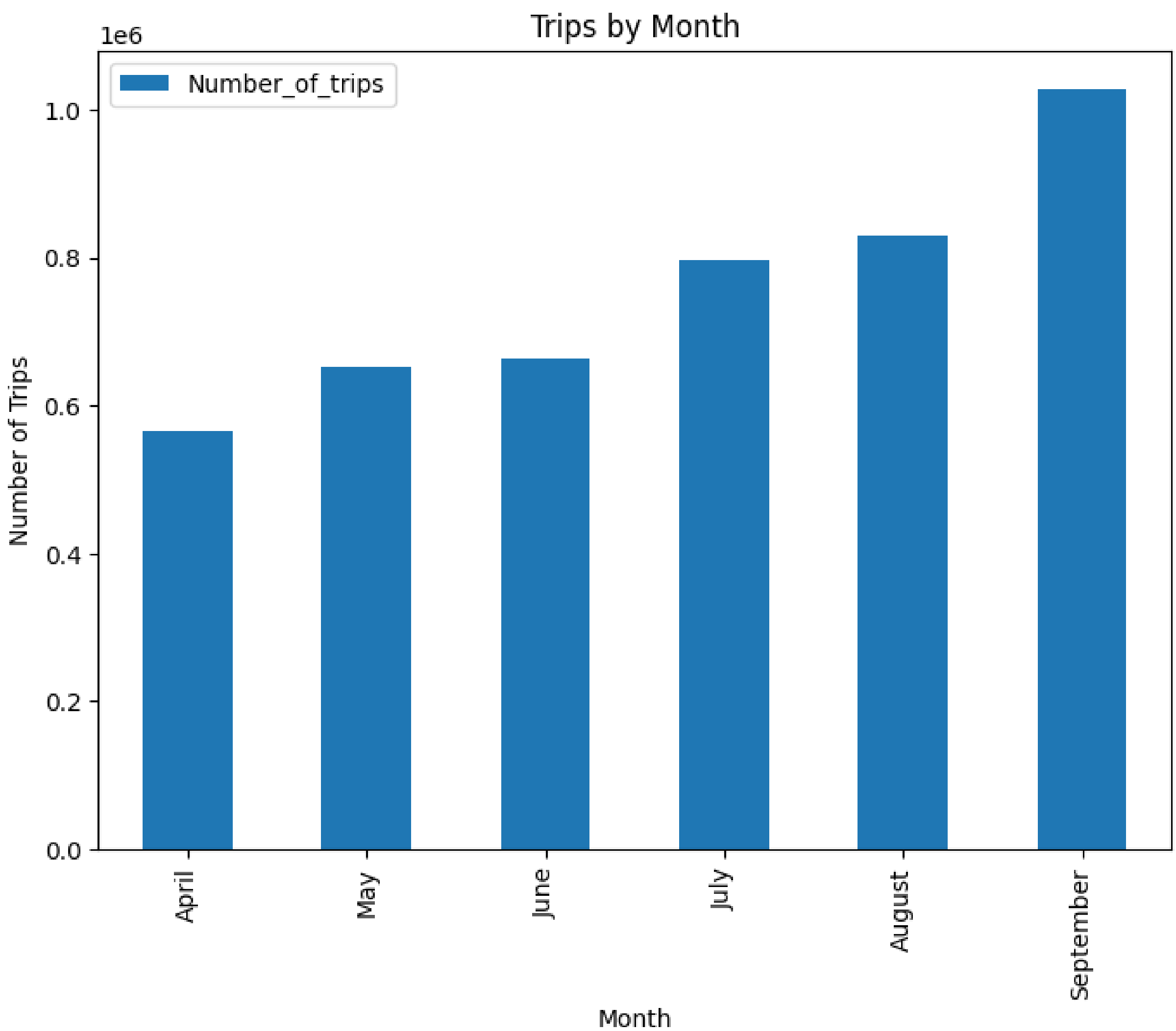
In [13]:
```
#Grouping by Month
df_month_grouped = df.groupby(['Month'], sort=False).count()

#Creating the sub dataframe
df_month = pd.DataFrame({'Number_of_trips':df_month_grouped.values[:,0]}, index = d

df_month
```

Out[13]:

	Number_of_trips
Month	
April	564516
May	652435
June	663844
July	796121
August	829275
September	1028136

Plotting the results

In [14]:
```
df_month.plot(kind='bar', figsize=(8,6))

plt.ylabel('Number of Trips')
plt.title('Trips by Month')

plt.show()
```



Analysing the results

We observe that the number of trips increases each month, with a peak increase between August and September.

In [15]:
```
number_of_trips_aug = df_month.loc['August'].values
number_of_trips_sep = df_month.loc['September'].values

ratio_month = (((number_of_trips_sep - number_of_trips_aug) / number_of_trips_aug)
ratio_month = round(ratio_month)

print('The ratio of the increase from August to September is {} %.'.format(ratio_mo
```

```
The ratio of the increase from August to September is 24 %.
```

From our results we can say that from April to September 2014, Uber was in a continuous improvement process.

2.3 Trips by weekday

In [16]:
```
#Grouping by Weekday
```

```
df_weekday_grouped = df.groupby(['Weekday'], sort = False).count()

#Creating the grouped DataFrame
df_weekday = pd.DataFrame({'Number_of_trips':df_weekday_grouped.values[:,0]}, index

df_weekday
```

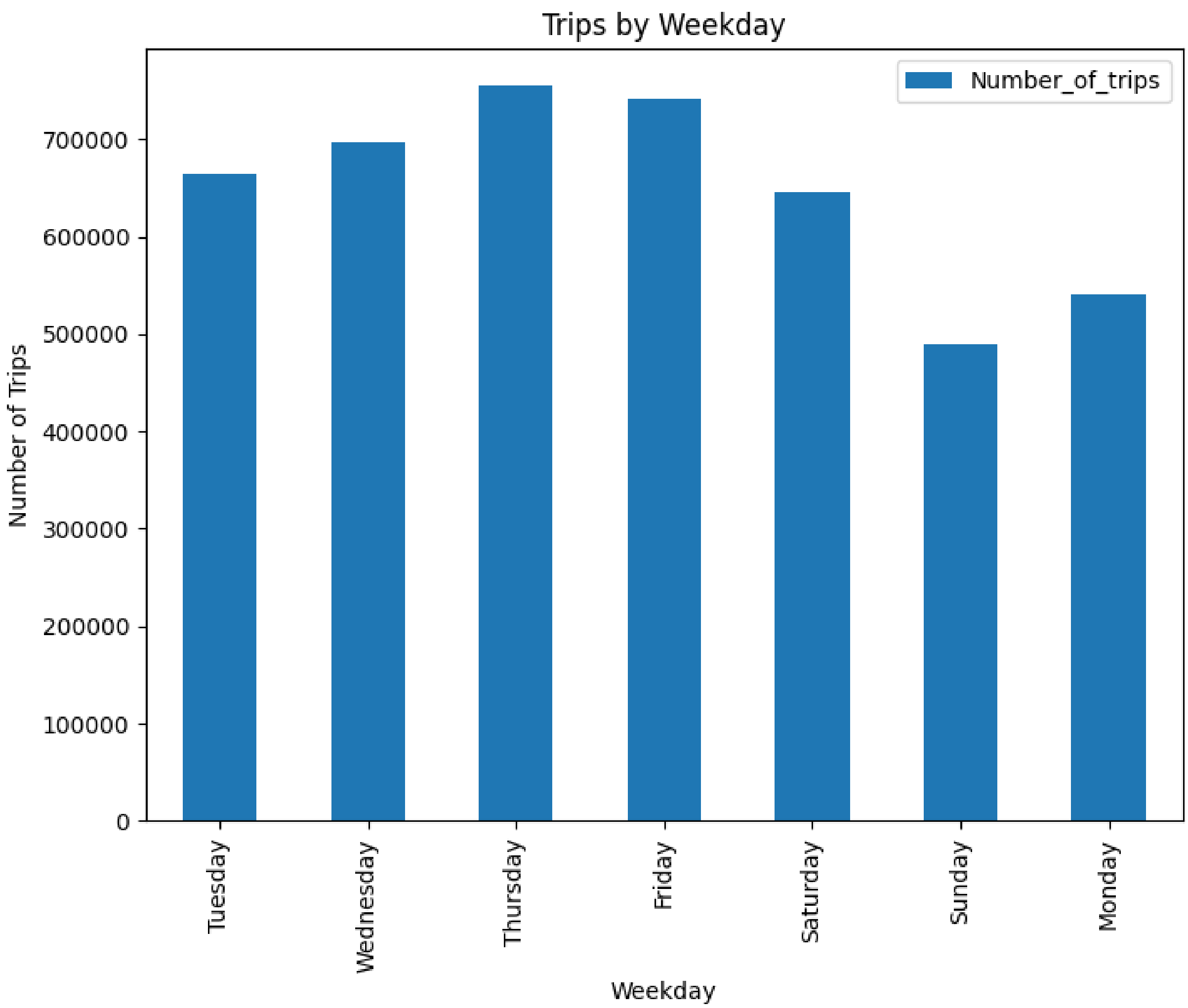
Out[16]:

	Number_of_trips
Weekday	
Tuesday	663789
Wednesday	696488
Thursday	755145
Friday	741139
Saturday	646114
Sunday	490180
Monday	541472

Plotting the results

In [17]:

```
df_weekday.plot(kind='bar', figsize=(8,6))

plt.ylabel('Number of Trips')
plt.title('Trips by Weekday')

plt.show()
```



Analysing the results

In [18]:

```
#Getting the minimum number of trips by weekday
min_number_of_trips_weekday = min(df_weekday['Number_of_trips'])

#Getting the weekday where the number of trips is minimal
min_weekday = df_weekday[df_weekday['Number_of_trips'] == min_number_of_trips_weekd

print('The lowest number of trips by weekday is {} trip, that corresponds to {}.'.f
```

```
The lowest number of trips by weekday is 490180 trip, that corresponds to Sunday.
```

In [19]:

```
#Getting the mean number of trips in the weekend - Non working day
mean_number_of_trips_weekend = ((df_weekday.loc['Saturday'] + df_weekday.loc['Sunda

#Getting the mean number of trips for the rest of the week- Working day
mean_number_of_trips_workday = (((df_weekday.loc['Monday'] + df_weekday.loc['Tuesda

ratio_weekday = (((mean_number_of_trips_workday - mean_number_of_trips_weekend) / m
ratio_weekday = round(ratio_weekday, 1)

print('The mean number of trips during working days is {}% higher than the mean num
```

```
The mean number of trips during working days is 19.6% higher than the mean number of
trips during weekends.
```

As the ratio between workdays and weekends only 19.6%, and because of the low number of trips on Monday, it cannot be said that people use Uber on workdays more than on weekends.

We need to investigate more to find out why the number of trips on mondays is as low.

2.4 Trips by day

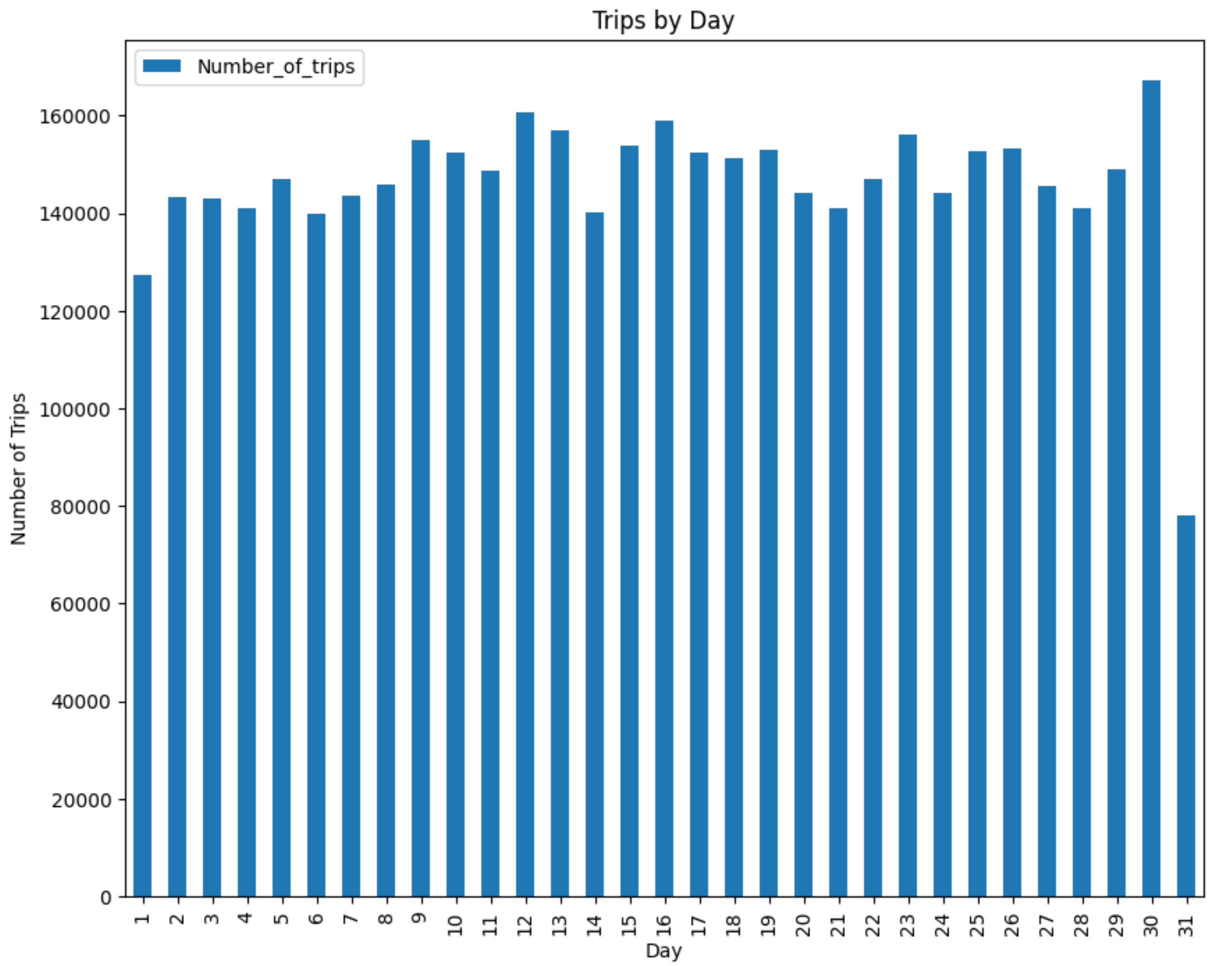
In [20]:
```
#Grouping by Day
df_day_grouped = df.groupby(['Day']).count()

#Creating the grouped DataFrame
df_day = pd.DataFrame({'Number_of_trips':df_day_grouped.values[:,0]}, index = df_da

df_day.head()
```

Out[20]:

Day	Number_of_trips
1	127430
2	143201
3	142983
4	140923
5	147054

Plotting the results

In [21]:
```
df_day.plot(kind='bar', figsize=(10,8))

plt.ylabel('Number of Trips')
plt.title('Trips by Day')

plt.show()
```



Analysing the results

The number of trips for the day 31 is a lot less than the others because April, June and September have 30 days.

The day with the highest number of trips is the 30. There's not much variation from day to day.

2.5 Trips by hour and month

In [22]:
```
#Grouping by Hour and Month
df_hour_month_grouped = df.groupby(['Hour','Month']).count()

#Creating the grouped DataFrame
df_hour_month = pd.DataFrame({'Number_of_trips':df_hour_month_grouped.values[:,1]},

df_hour_month.head(10)
```

Out[22]:

Hour	Month	Number_of_trips
0	April	11910
	August	21451
	July	17953
	June	14514
	May	13875
	September	24133
1	April	7769
	August	14471
	July	11527
	June	9167

In [23]:

```
#Reseting the Index
df_hour_month.reset_index(inplace= True)
df_hour_month.head()
```

Out[23]:

	Hour	Month	Number_of_trips
0	0	April	11910
1	0	August	21451
2	0	July	17953
3	0	June	14514
4	0	May	13875

In [24]:

```
#Preparing the Number of trips data
#We create a Numpy array that includes the Number of trips data then reshape it to
data_hour_month = df_hour_month['Number_of_trips'].values.reshape(24,6)
data_hour_month
```

Out[24]:
```
array([[11910, 21451, 17953, 14514, 13875, 24133],
       [ 7769, 14471, 11527,  9167,  8186, 16107],
       [ 4935, 10105,  8562,  6189,  5372, 10702],
       [ 5040, 10376,  9199,  6937,  5946, 10789],
       [ 6095, 11774, 10040,  7701,  6945, 12675],
       [ 9476, 16525, 14932, 11955, 10789, 20262],
       [18498, 24907, 23456, 22030, 21015, 33307],
       [24924, 34064, 32545, 30834, 27413, 43314],
       [22843, 34566, 33387, 29771, 25460, 44477],
       [17939, 30195, 28486, 24298, 20507, 38542],
       [17865, 30706, 28558, 23584, 20801, 37634],
       [18774, 31778, 30120, 24155, 22055, 38821],
       [19425, 32106, 30900, 25233, 23595, 39193],
       [22603, 35764, 35832, 28937, 27699, 45042],
       [27190, 40644, 41357, 34428, 34363, 52643],
       [35324, 48197, 46053, 41586, 43087, 61219],
       [42003, 53481, 52403, 48162, 49127, 68224],
       [45475, 57122, 58260, 50452, 51508, 73373],
       [43003, 55390, 57268, 45013, 48965, 75040],
       [38923, 53008, 52332, 38203, 42387, 69660],
       [36244, 51674, 51859, 40108, 40731, 63988],
       [36964, 51354, 49528, 40791, 42217, 60606],
       [30645, 46008, 42218, 35614, 35556, 51817],
       [20649, 33609, 29346, 24182, 24836, 36568]], dtype=int64)
```

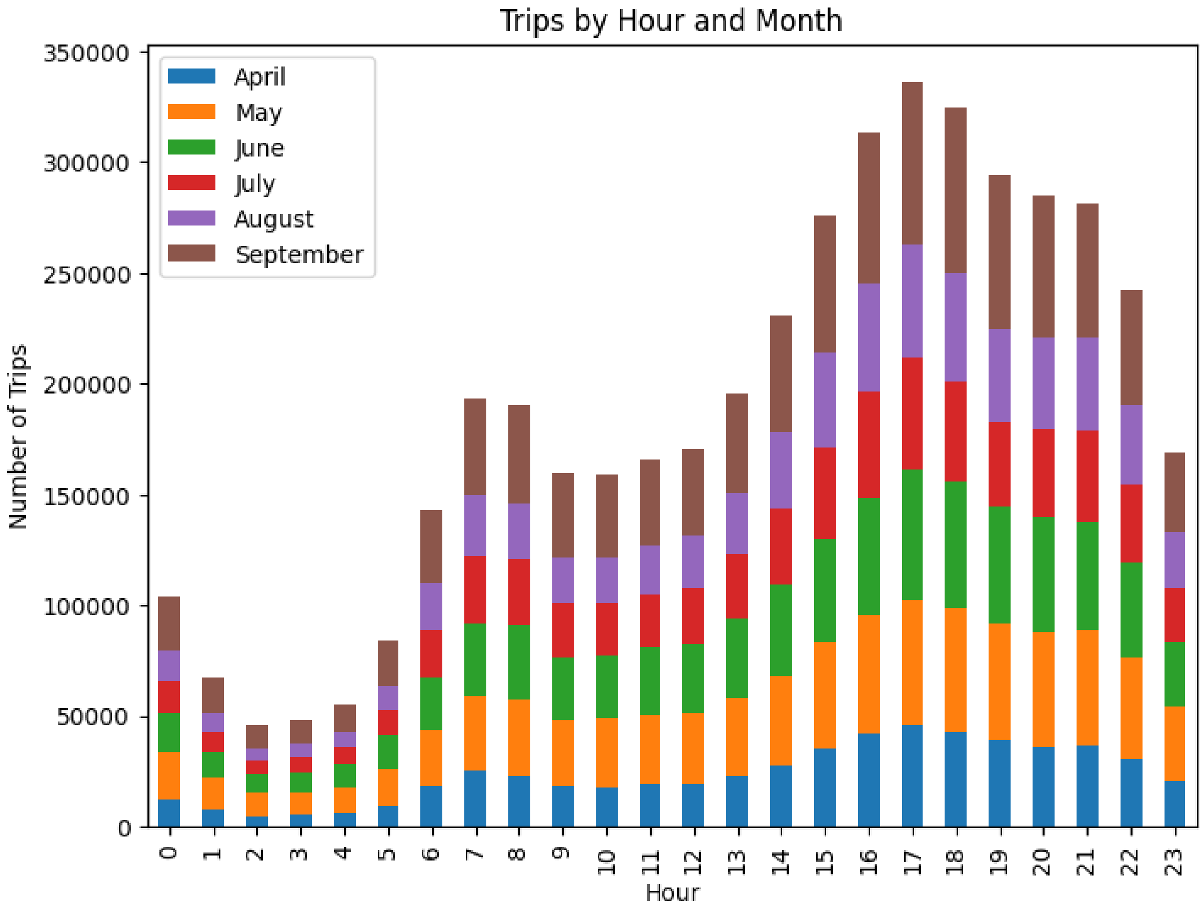
In [25]:
```
df_hour_month = pd.DataFrame(data = data_hour_month, index = df_hour_month['Hour'].
df_hour_month.head()
```

Out[25]:

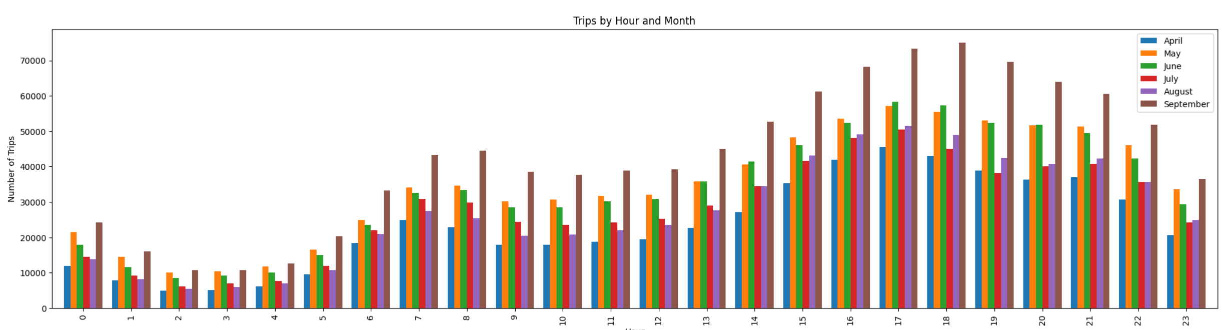
	April	May	June	July	August	September
0	11910	21451	17953	14514	13875	24133
1	7769	14471	11527	9167	8186	16107
2	4935	10105	8562	6189	5372	10702
3	5040	10376	9199	6937	5946	10789
4	6095	11774	10040	7701	6945	12675

Plotting the results

In [26]:
```
df_hour_month.plot(kind='bar', figsize=(8,6), stacked=True)

plt.xlabel('Hour')
plt.ylabel('Number of Trips')
plt.title('Trips by Hour and Month')

plt.show()
```



In [27]:
```
df_hour_month.plot(kind='bar', figsize=(25,6),width=0.8)

plt.xlabel('Hour')
plt.ylabel('Number of Trips')
plt.title('Trips by Hour and Month')

plt.show()
```



2.6 Trips by weekday and hour

In [28]:
```
#Grouping by Hour and weekday
df_weekday_hour_grouped = df.groupby(['Weekday','Hour'], sort = False).count()

#Creating the grouped DataFrame
df_weekday_hour = pd.DataFrame({'Number_of_trips':df_weekday_hour_grouped.values[:,
```

```python
df_weekday_hour
```

Out[28]:

		Number_of_trips
Weekday	**Hour**	
Tuesday	**0**	6237
	1	3509
	2	2571
	3	4494
	4	7548
...	**...**	...
Monday	**19**	34159
	20	32849
	21	28925
	22	20158
	23	11811

168 rows × 1 columns

In [29]:

```python
#Reseting the Index
df_weekday_hour.reset_index(inplace= True)

#Preparing the Number of trips data
data_weekday_hour = df_weekday_hour['Number_of_trips'].values.reshape(7,24)

df_weekday_hour = pd.DataFrame(data = data_weekday_hour, index = df_weekday_hour['W
df_weekday_hour.head()
```

Out[29]:

	0	1	2	3	4	5	6	7	8	9	...	
Tuesday	6237	3509	2571	4494	7548	14241	26872	36599	33934	25023	...	3
Wednesday	7644	4324	3141	4855	7511	13794	26943	36495	33826	25635	...	3
Thursday	9293	5290	3719	5637	8505	14169	27065	37038	35431	27812	...	3
Friday	13716	8163	5350	6930	8806	13450	23412	32061	31509	25230	...	3
Saturday	27633	19189	12710	9542	6846	7084	8579	11014	14411	17669	...	3

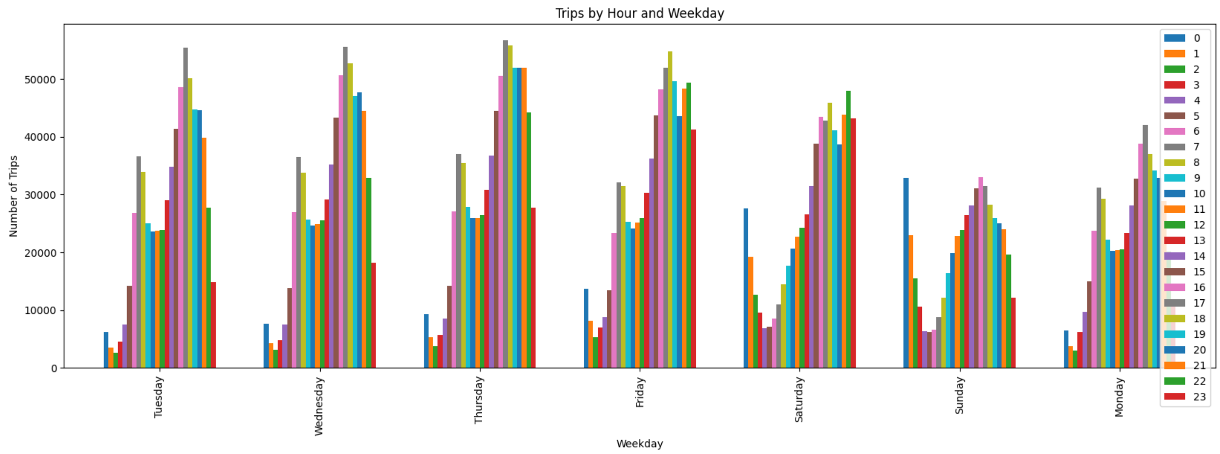
5 rows × 24 columns

Plotting the results

In [30]:

```python
df_weekday_hour.plot(kind='bar', figsize=(20,6), width = 0.7)
```

```python
plt.xlabel('Weekday')
plt.ylabel('Number of Trips')
plt.title('Trips by Hour and Weekday')

plt.show()
```



Analysing the results

We see that in working days there's a pulse at 7:00 and 8:00, it corresponds to the hour where the employees go to work. This pulse is not present on weekend days.

At the same time we see that on weekend days the number of trips around midnight, 1:00 and 2:00 is higher than on working days.

2.7 Trips by weekday and month

In [31]:
```python
#Grouping by Weekday and Month
df_month_weekday_grouped = df.groupby(['Month','Weekday'], sort=False).count()

#Creating the grouped DataFrame
df_month_weekday = pd.DataFrame({'Number_of_trips':df_month_weekday_grouped.values[

df_month_weekday.head(10)
```

Out[31]:

Month	Weekday	Number_of_trips
April	Tuesday	91185
	Wednesday	108631
	Thursday	85067
	Friday	90303
	Saturday	77218
	Sunday	51251
	Monday	60861
May	Thursday	128921
	Friday	133991
	Saturday	102990

In [32]:

```
#Reseting the Index
df_month_weekday.reset_index(inplace= True)

#Preparing the Number of trips
data_month_weekday = df_month_weekday['Number_of_trips'].values.reshape(6,7)

df_month_weekday = pd.DataFrame(data = data_month_weekday, index = df_month_weekday
df_month_weekday.head()
```

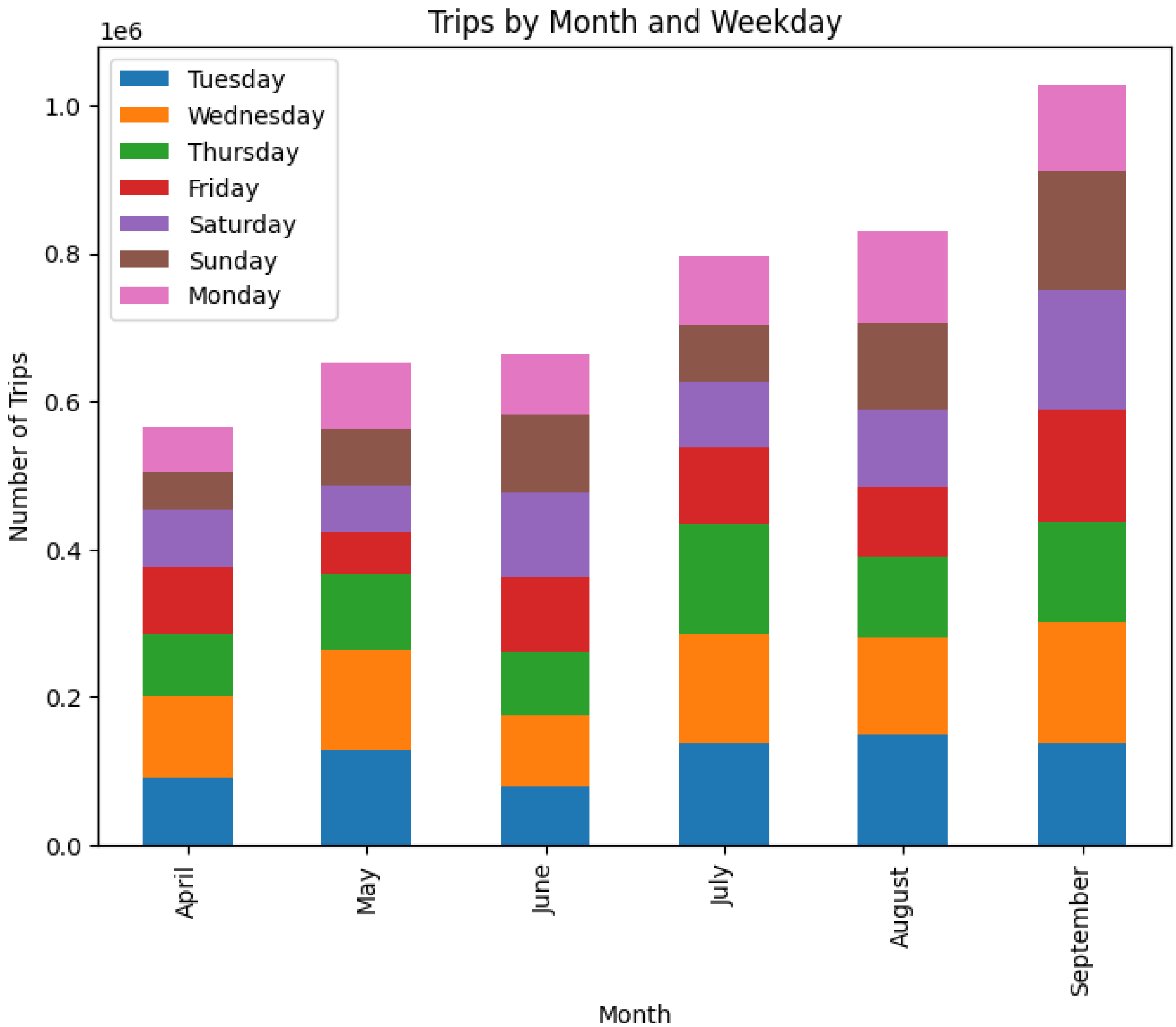
Out[32]:

	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Monday
April	91185	108631	85067	90303	77218	51251	60861
May	128921	133991	102990	56168	63846	76662	89857
June	79656	94655	88134	99654	115325	105056	81364
July	137454	147717	148439	102735	90260	76327	93189
August	148674	132225	110246	91633	107124	115256	124117

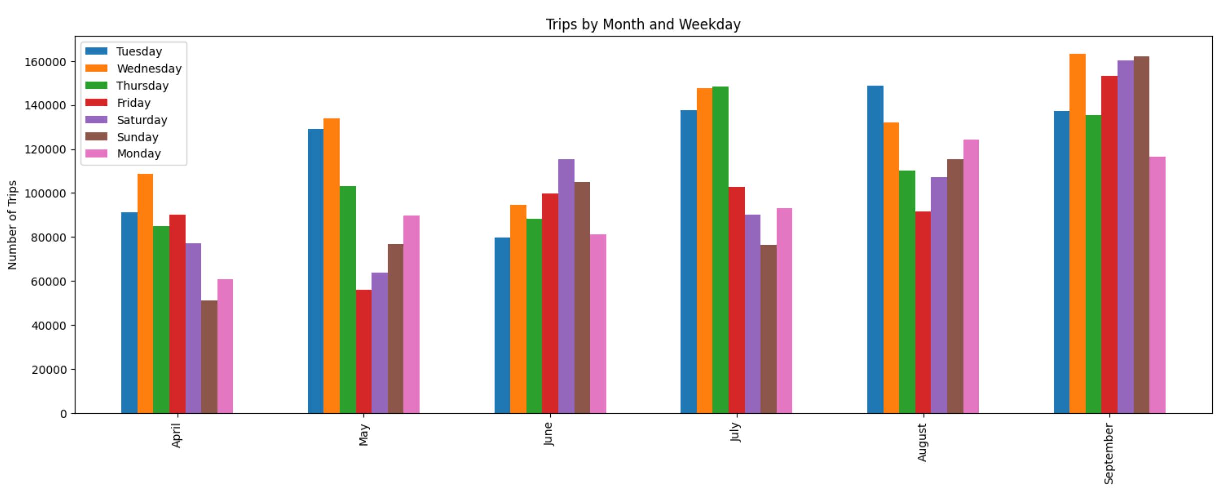
Plotting the results

In [33]:

```
df_month_weekday.plot(kind='bar', figsize=(8,6), stacked = True)

plt.xlabel('Month')
plt.ylabel('Number of Trips')
plt.title('Trips by Month and Weekday')

plt.show()
```



In [34]:
```
df_month_weekday.plot(kind='bar', figsize=(18,6), width = 0.6)

plt.xlabel('Month')
plt.ylabel('Number of Trips')
plt.title('Trips by Month and Weekday')

plt.show()
```



3. Heatmap

Through our exploration we are going to visualize:

- Heatmap by Hour and Day.
- Heatmap by Hour and Weekday.
- Heatmap by Month and Day.
- Heatmap by Month and Weekday.

In [35]:
```
#Defining a function that counts the number of rows
def count_rows(rows):
    return len(rows)
```

3.1 Heatmap by Hour and Day

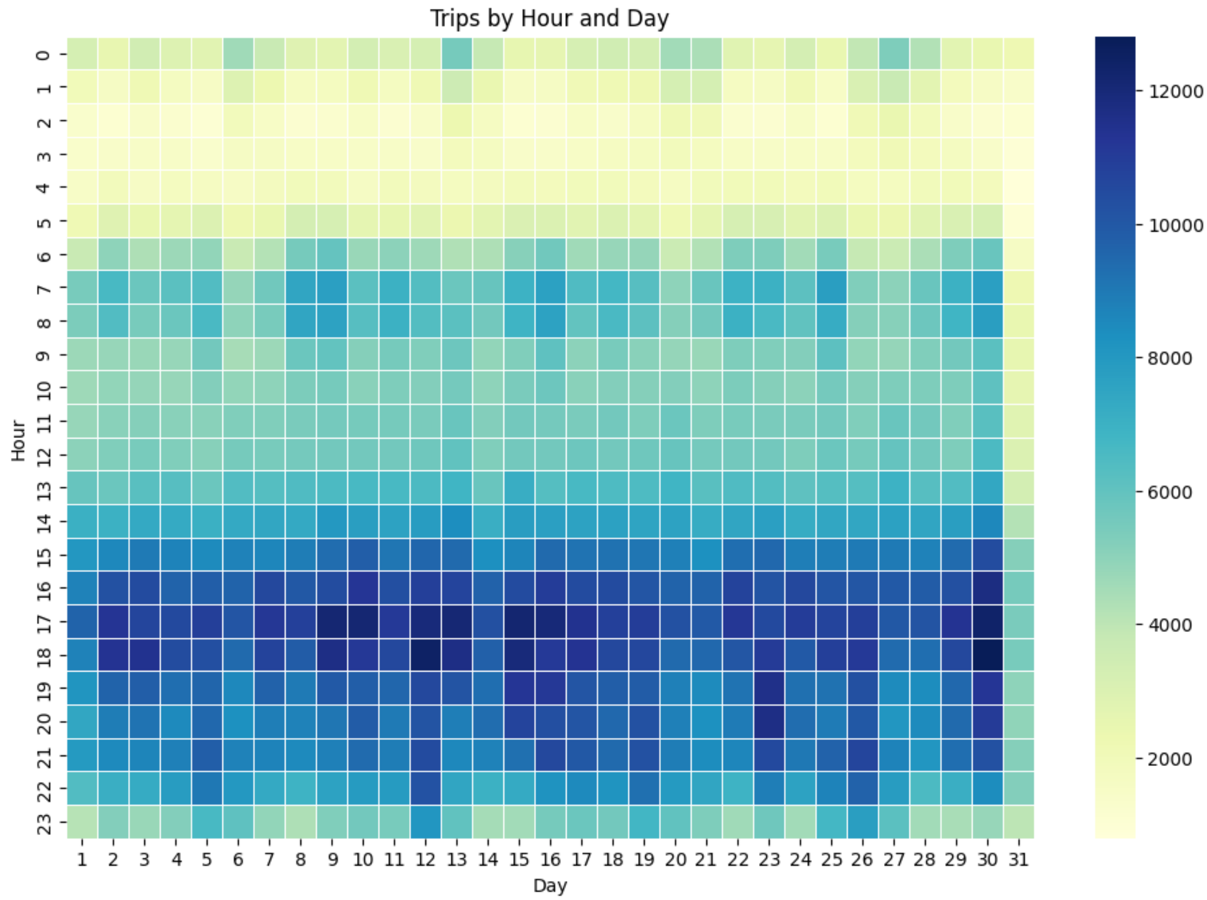
In [36]:
```
#Creating the hour and day dataframe
df_hour_day = df.groupby('Hour Day'.split()).apply(count_rows).unstack()
df_hour_day.head()
```

Out[36]:

Day	1	2	3	4	5	6	7	8	9	10	...	22	23	2
Hour														
0	3247	2480	3415	2944	2786	4623	3715	2882	2708	3351	...	2852	2605	327
1	1982	1600	2176	1718	1567	2932	2326	1672	1757	2167	...	1662	1608	209
2	1284	1109	1434	1217	1047	1902	1533	1147	1246	1495	...	1207	1163	148
3	1331	1442	1489	1449	1278	1599	1622	1509	1501	1466	...	1503	1465	152
4	1458	1897	1578	1737	1648	1584	1793	1921	1911	1614	...	1984	1994	182

5 rows × 31 columns

In [37]:
```
plt.figure(figsize = (12,8))

#Using the seaborn heatmap function
ax = sns.heatmap(df_hour_day, cmap=cm.YlGnBu, linewidth = .5)
ax.set(title="Trips by Hour and Day");
```



Analysing the results

We see that the number of trips in increasing throughout the day, with a peak demand in the evening between 16:00 and 18:00.

It corresponds to the time where employees finish their work and go home.

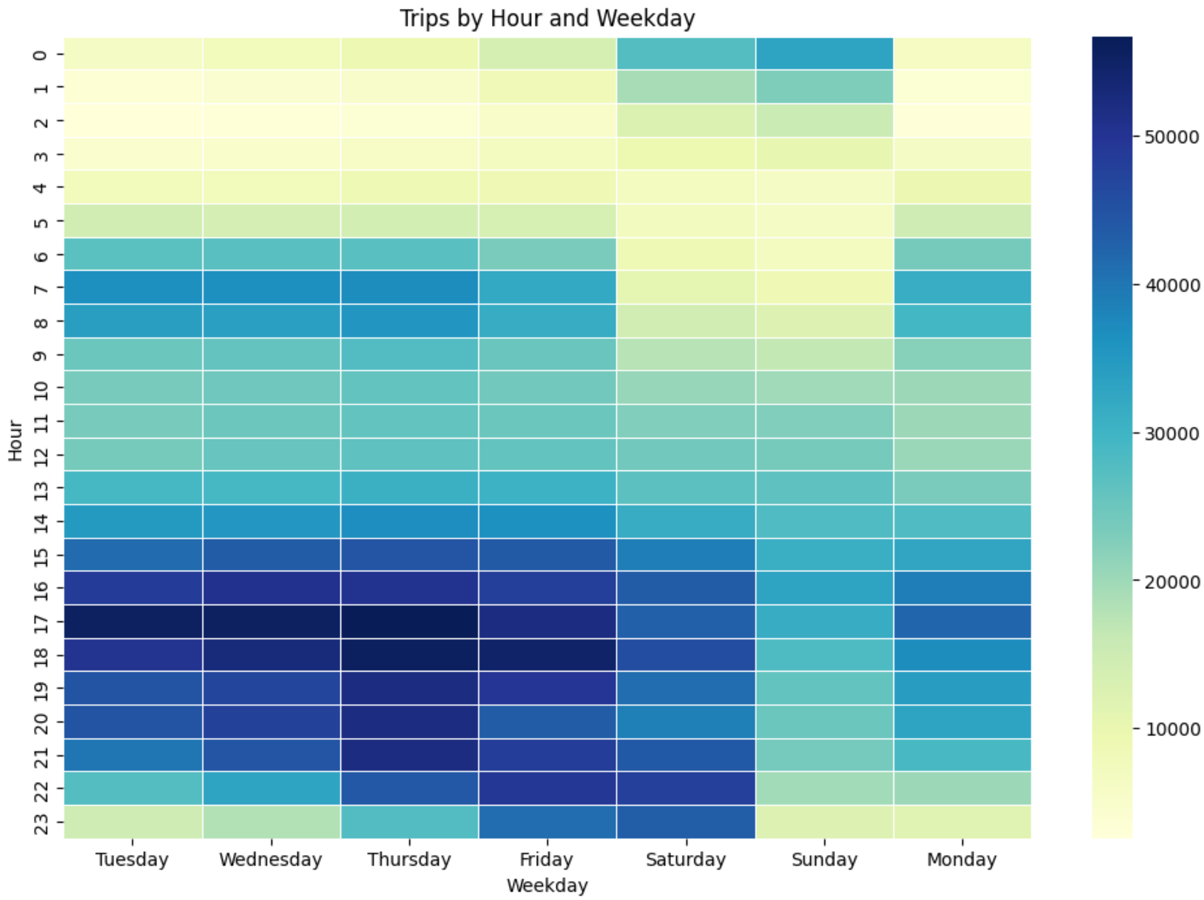
3.2 Heatmap by Hour and Weekday

In [38]:
```
df_hour_weekday = df.groupby('Hour Weekday'.split(), sort = False).apply(count_rows
df_hour_weekday.head()
```

Out[38]:

Weekday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Monday
Hour							
0	6237	7644	9293	13716	27633	32877	6436
1	3509	4324	5290	8163	19189	23015	3737
2	2571	3141	3719	5350	12710	15436	2938
3	4494	4855	5637	6930	9542	10597	6232
4	7548	7511	8505	8806	6846	6374	9640

```
In [39]: plt.figure(figsize = (12,8))

         ax = sns.heatmap(df_hour_weekday, cmap=cm.YlGnBu, linewidth = .5)
         ax.set(title="Trips by Hour and Weekday");
```



Analysing the results

We can see that on working days (From Monday to Friday) the number of trips is higher from 16:00 to 21:00. It shows even better what we said from the first heatmap.

On Friday the number of trips remains high until 23:00 and continues on early Saturday. It corresponds to the time where people come out from work, then go out for dinner or drink before the weekend.

We can notice the same pattern on Saturday, people tend to go out at night, the number of trips remains on high until early Sunday.

3.3 Heatmap by Day and Month

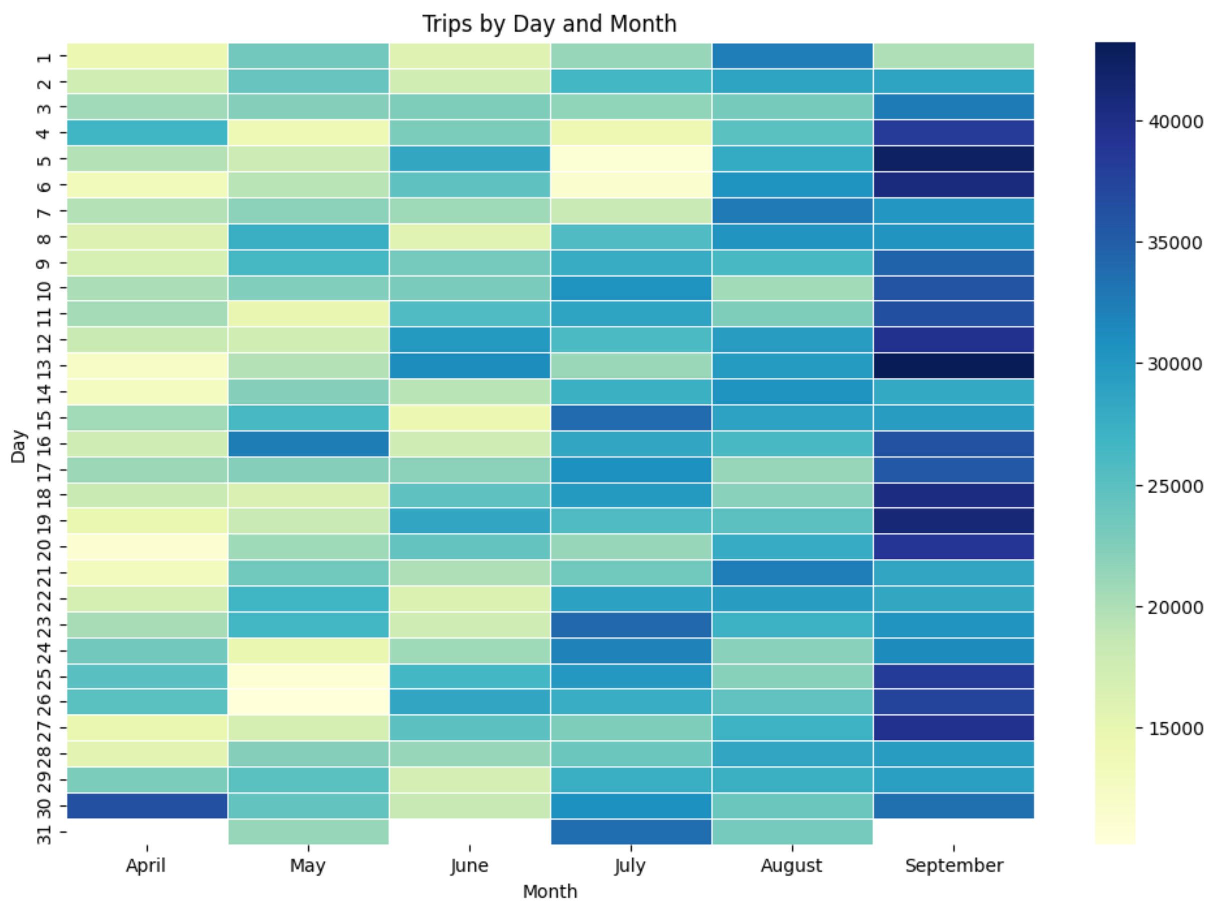
```
In [40]: df_day_month = df.groupby('Day Month'.split(), sort = False).apply(count_rows).unst
         df_day_month.head()
```

Out[40]:

Month	April	May	June	July	August	September
Day						
1	14546.0	23375.0	15967.0	21228.0	32353.0	19961.0
2	17474.0	24235.0	17503.0	26480.0	28678.0	28831.0
3	20701.0	22234.0	22674.0	21597.0	23146.0	32631.0
4	26714.0	13918.0	22831.0	14148.0	24952.0	38360.0
5	19521.0	17859.0	28371.0	10890.0	28094.0	42319.0

In [41]:

```python
plt.figure(figsize = (12,8))

ax = sns.heatmap(df_day_month, cmap = cm.YlGnBu, linewidth = .5)
ax.set(title="Trips by Day and Month");
```



Analysing the results

We observe that the number of trips increases each month, we can say that from April to September 2014, Uber was in a continuous improvement process.

We can notice from the visualization a dark spot, it corresponds to the 30 April. The number of trips that day was extreme compared to the rest of the month.

Unfortunatly we have not been able to find any factual information to explain the pulse. A successful marketing strategy can be assumed to be in place that days. So as the analysis go on we consider that day an outliner.

In [42]:
```
#The number of trips the 30th of April
max_april = max(df_day_month['April'])

#The mean number of trips the rest of April
mean_rest_april = df_day_month['April'][0:29].sum() / 29

ratio_april = round(max_april / mean_rest_april)
print('The number of trips the 30th of April is {} times higher than the mean numbe
```

```
The number of trips the 30th of April is 2 times higher than the mean number of trip
s during the rest of the month
```

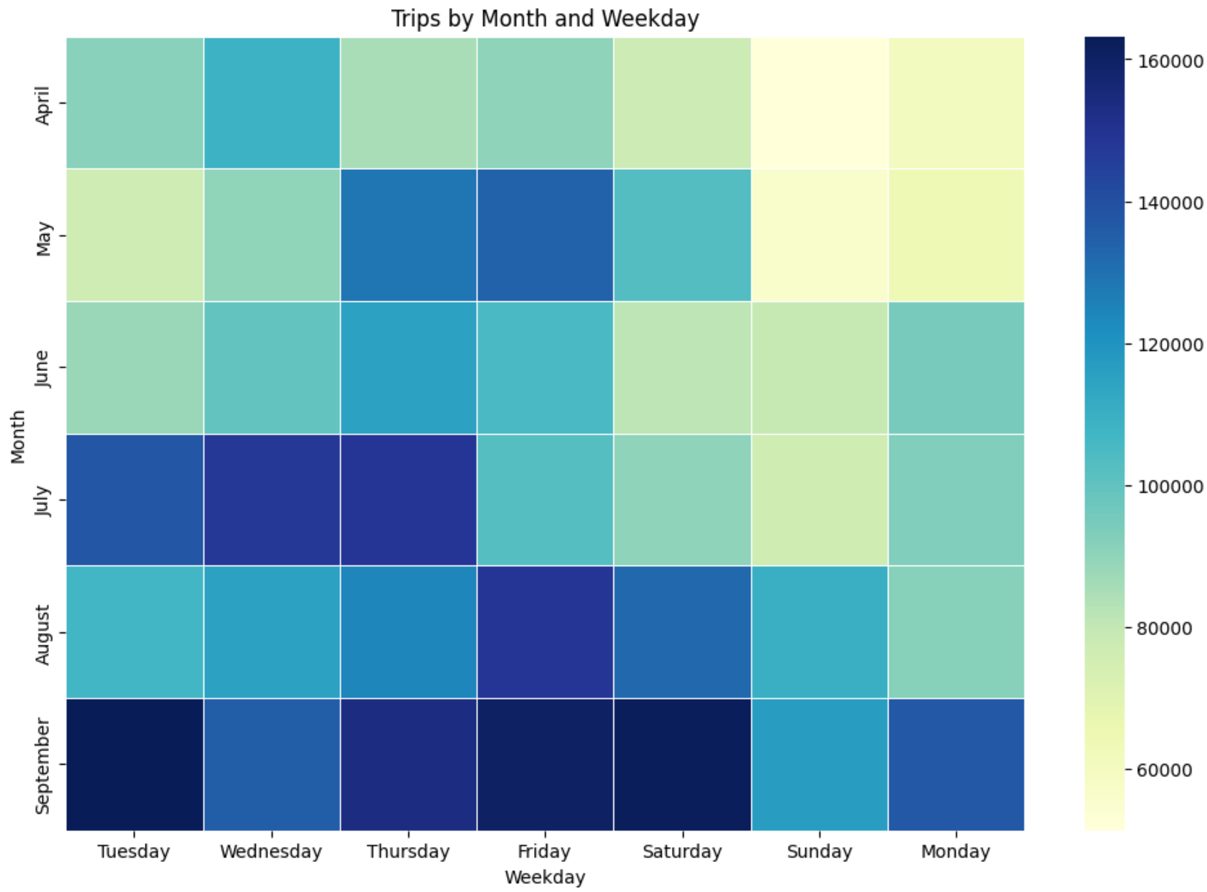
3.4 Heatmap by Month and Weekday

In [43]:
```
df_month_weekday = df.groupby('Month Weekday'.split(), sort = False).apply(count_ro
df_month_weekday.head()
```

Out[43]:

Weekday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Monday
Month							
April	91185	108631	85067	90303	77218	51251	60861
May	76662	89857	128921	133991	102990	56168	63846
June	88134	99654	115325	105056	81364	79656	94655
July	137454	147717	148439	102735	90260	76327	93189
August	107124	115256	124117	148674	132225	110246	91633

In [44]:
```
plt.figure(figsize = (12,8))

ax = sns.heatmap(df_month_weekday, cmap= cm.YlGnBu, linewidth = .5)
ax.set(title="Trips by Month and Weekday");
```



4. Conclusion

Through our analysis of the Uber Pickups in New York City data set in 2014, we managed to get the following informations:

- The peak demand hour 17:00.
- The main customer category are workers.
- An indicator of Uber's improvement from April to September.
- People tend to use Uber to go to work around 7:00 and 8:00 on working days.
- People tend to use Uber late at night (around midnight) during weekends.
- We should investigate why people don't use uber on Mondays as much as they do on other working days.