# UBER TRIP ANALYSIS

Uber uses a mixture of internal and external data to estimate fares. Uber calculates fares automatically using street traffic data, GPS data and its own algorithms that make alterations based on the time of the journey. It also analyses external data like public transport routes to plan various services.

## Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

## Load The Dataset

In [2]:
```
data=pd.read_csv(r'C:\Users\user\Downloads\uber-raw-data-sep14.csv')
```

In [3]:
```
data["Date/Time"] = data["Date/Time"].map(pd.to_datetime)
```

## Basic Chacks

In [12]:
```
data.head()
```

Out[12]:

| | Date/Time | Lat | Lon | Base | Day | Weekday | Hour |
|---|---|---|---|---|---|---|---|
| **0** | 2014-09-01 00:01:00 | 40.2201 | -74.0021 | B02512 | 1 | 0 | 0 |
| **1** | 2014-09-01 00:01:00 | 40.7500 | -74.0027 | B02512 | 1 | 0 | 0 |
| **2** | 2014-09-01 00:03:00 | 40.7559 | -73.9864 | B02512 | 1 | 0 | 0 |
| **3** | 2014-09-01 00:06:00 | 40.7450 | -73.9889 | B02512 | 1 | 0 | 0 |
| **4** | 2014-09-01 00:11:00 | 40.8145 | -73.9444 | B02512 | 1 | 0 | 0 |

In [13]:
```
data.tail()
```

Out[13]:

| | Date/Time | Lat | Lon | Base | Day | Weekday | Hour |
|---|---|---|---|---|---|---|---|
| **1028131** | 2014-09-30 22:57:00 | 40.7668 | -73.9845 | B02764 | 30 | 1 | 22 |
| **1028132** | 2014-09-30 22:57:00 | 40.6911 | -74.1773 | B02764 | 30 | 1 | 22 |
| **1028133** | 2014-09-30 22:58:00 | 40.8519 | -73.9319 | B02764 | 30 | 1 | 22 |
| **1028134** | 2014-09-30 22:58:00 | 40.7081 | -74.0066 | B02764 | 30 | 1 | 22 |
| **1028135** | 2014-09-30 22:58:00 | 40.7140 | -73.9496 | B02764 | 30 | 1 | 22 |

In [14]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1028136 entries, 0 to 1028135
Data columns (total 7 columns):
 #   Column     Non-Null Count    Dtype
---  ------     --------------    -----
 0   Date/Time  1028136 non-null  datetime64[ns]
 1   Lat        1028136 non-null  float64
 2   Lon        1028136 non-null  float64
 3   Base       1028136 non-null  object
 4   Day        1028136 non-null  int64
 5   Weekday    1028136 non-null  int64
 6   Hour       1028136 non-null  int64
dtypes: datetime64[ns](1), float64(2), int64(3), object(1)
memory usage: 54.9+ MB
```

In [15]:

```
data.dtypes
```

Out[15]:

```
Date/Time    datetime64[ns]
Lat                 float64
Lon                 float64
Base                 object
Day                   int64
Weekday               int64
Hour                  int64
dtype: object
```

In [16]:

```
data.describe()
```

Out[16]:

| | Lat | Lon | Day | Weekday | Hour |
|---|---|---|---|---|---|
| **count** | 1.028136e+06 | 1.028136e+06 | 1.028136e+06 | 1.028136e+06 | 1.028136e+06 |
| **mean** | 4.073922e+01 | -7.397182e+01 | 1.555385e+01 | 2.961477e+00 | 1.409235e+01 |
| **std** | 4.082861e-02 | 5.831413e-02 | 8.448335e+00 | 1.942572e+00 | 5.971244e+00 |
| **min** | 3.998970e+01 | -7.477360e+01 | 1.000000e+00 | 0.000000e+00 | 0.000000e+00 |
| **25%** | 4.072040e+01 | -7.399620e+01 | 8.000000e+00 | 1.000000e+00 | 1.000000e+01 |
| **50%** | 4.074180e+01 | -7.398310e+01 | 1.600000e+01 | 3.000000e+00 | 1.500000e+01 |
| **75%** | 4.076120e+01 | -7.396280e+01 | 2.300000e+01 | 5.000000e+00 | 1.900000e+01 |
| **max** | 4.134760e+01 | -7.271630e+01 | 3.000000e+01 | 6.000000e+00 | 2.300000e+01 |

In [17]:

```
data.shape
```

Out[17]:

```
(1028136, 7)
```

In [18]:

```
data.isnull().sum()
```

Out[18]:

```
Date/Time    0
Lat          0
Lon          0
Base         0
Day          0
Weekday      0
Hour         0
dtype: int64
```

In [19]:

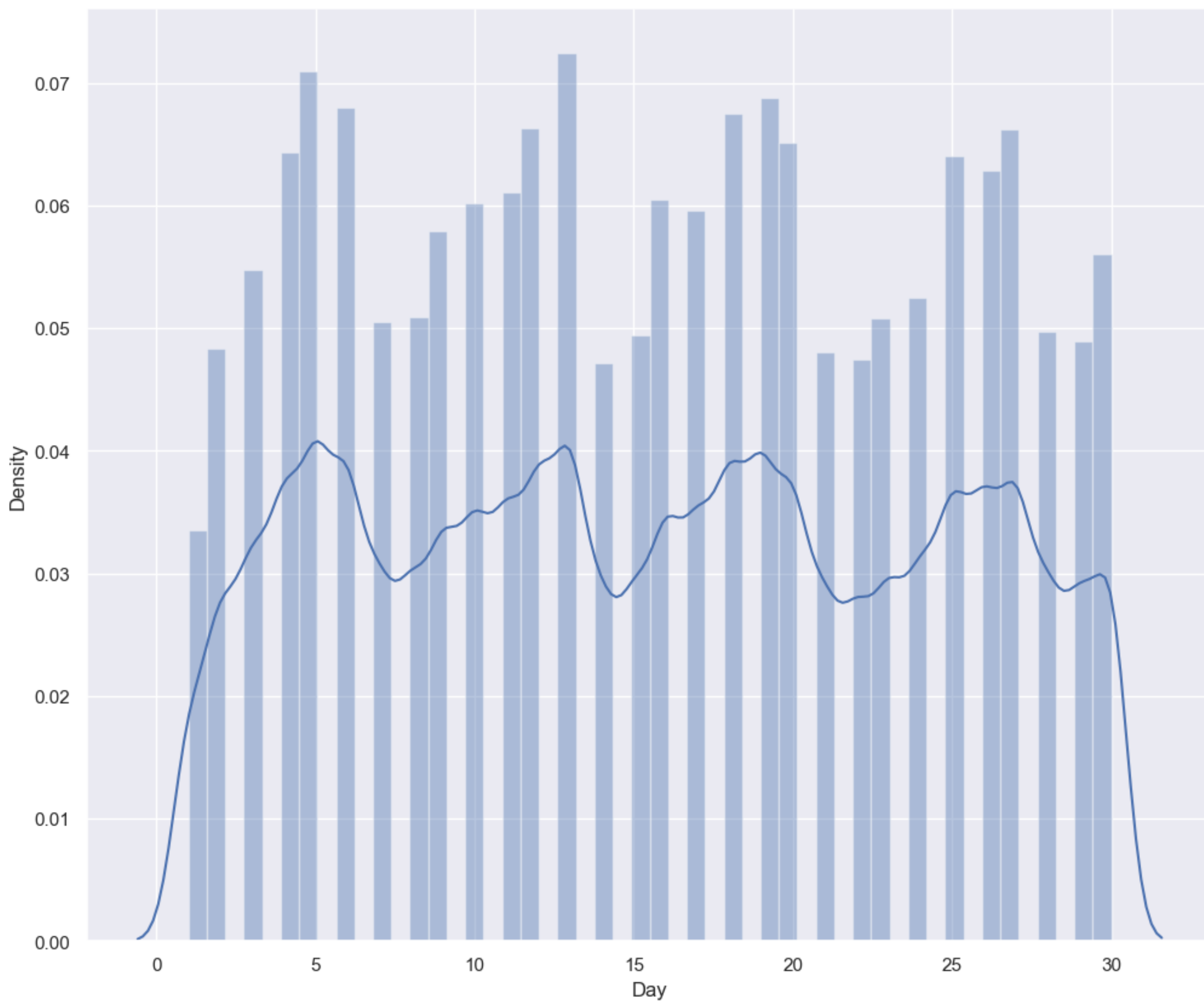
```
data["Day"] = data["Date/Time"].apply(lambda x: x.day)
data["Weekday"] = data["Date/Time"].apply(lambda x: x.weekday())
data["Hour"] = data["Date/Time"].apply(lambda x: x.hour)
print(data.head())
```

```
            Date/Time      Lat      Lon    Base  Day  Weekday  Hour
0 2014-09-01 00:01:00  40.2201 -74.0021  B02512    1        0     0
1 2014-09-01 00:01:00  40.7500 -74.0027  B02512    1        0     0
2 2014-09-01 00:03:00  40.7559 -73.9864  B02512    1        0     0
3 2014-09-01 00:06:00  40.7450 -73.9889  B02512    1        0     0
4 2014-09-01 00:11:00  40.8145 -73.9444  B02512    1        0     0
```

In [23]:

```python
import warnings
warnings.filterwarnings('ignore')
sns.set(rc={'figure.figsize':(12, 10)})
sns.distplot(data["Day"])
```

Out[23]: <Axes: xlabel='Day', ylabel='Density'>

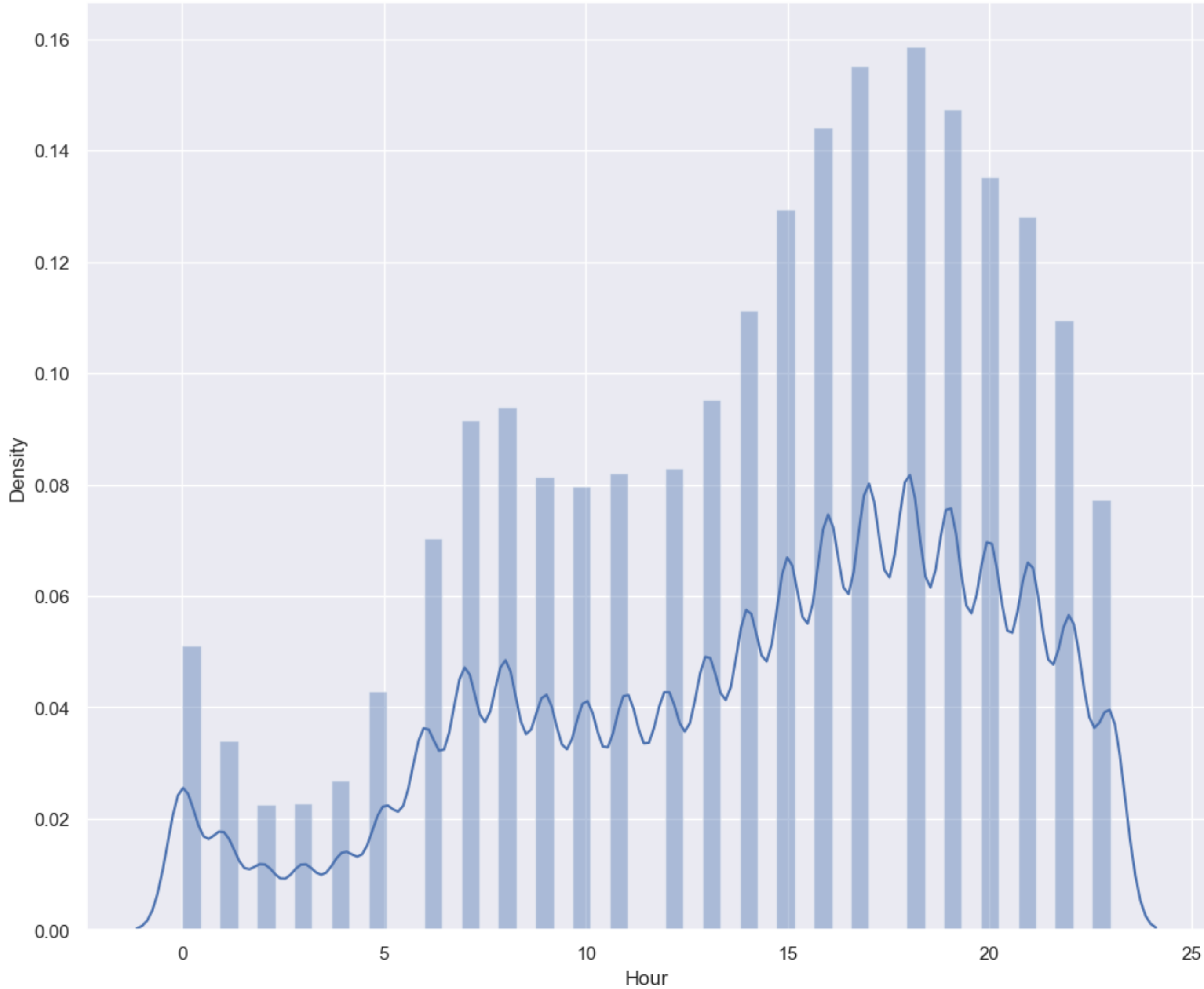



In [24]:

```python
sns.distplot(data["Hour"])
```

Out[24]: <Axes: xlabel='Hour', ylabel='Density'>

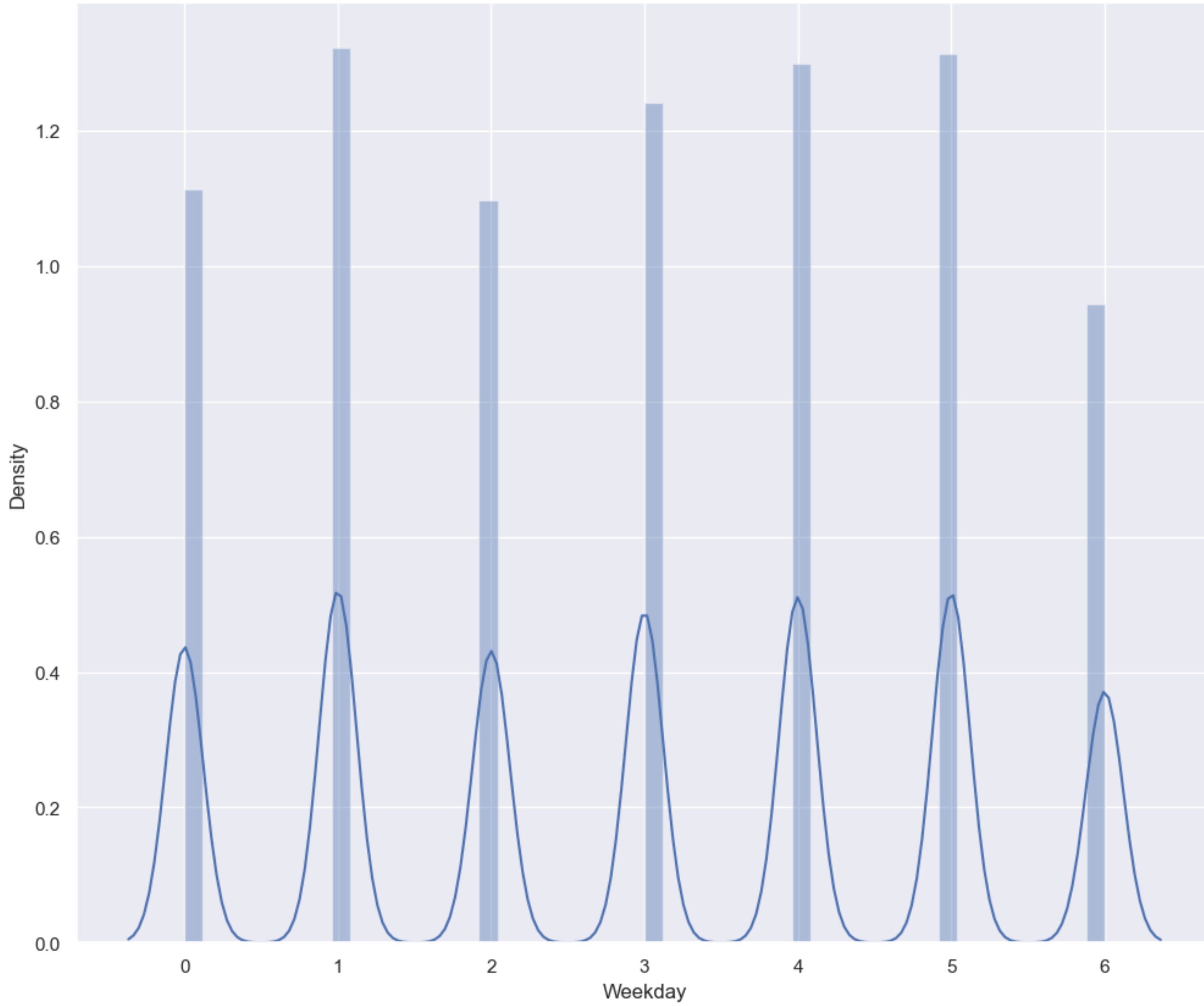
Loading [MathJax]/extensions/Safe.js




In [25]:
```
sns.distplot(data["Weekday"])
```

Out[25]:
```
<Axes: xlabel='Weekday', ylabel='Density'>
```

Loading [MathJax]/extensions/Safe.js

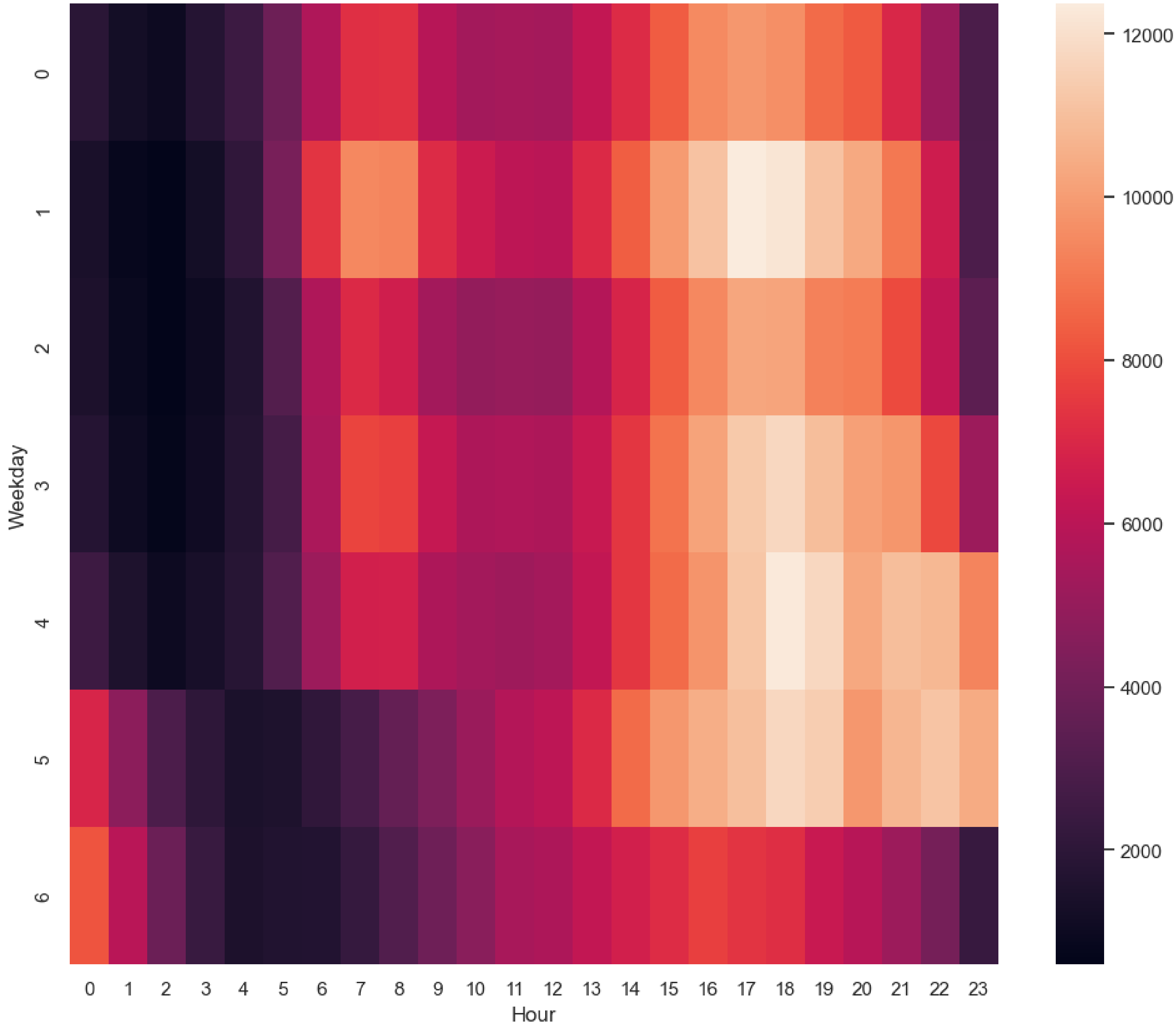



In [26]:

```
# Correlation of Weekday and Hour
df = data.groupby(["Weekday", "Hour"]).apply(lambda x: len(x))
df = df.unstack()
sns.heatmap(df, annot=False)
```

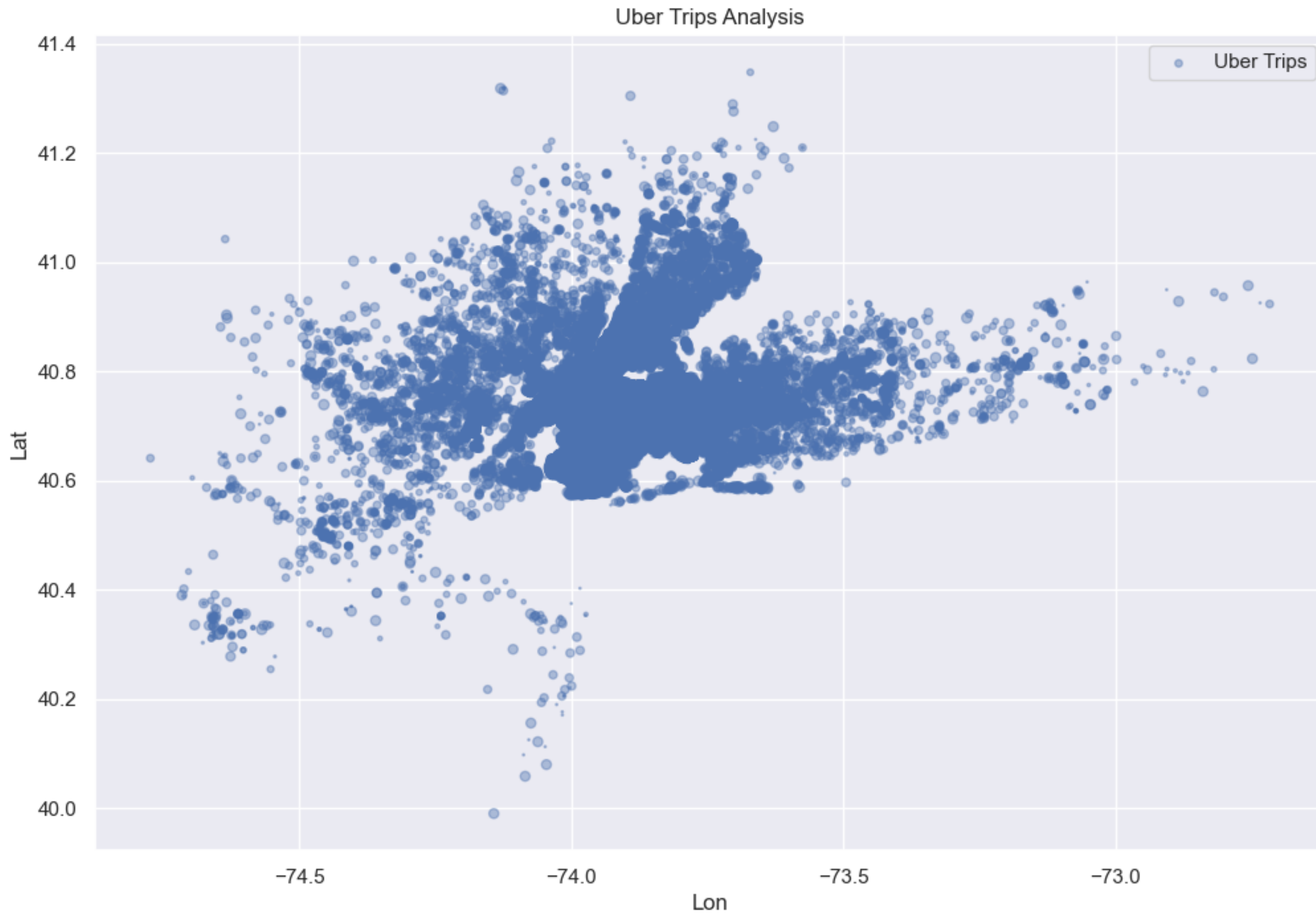
Out[26]:

```
<Axes: xlabel='Hour', ylabel='Weekday'>
```




In [27]:
```
data.plot(kind='scatter', x='Lon', y='Lat', alpha=0.4, s=data['Day'], label='Uber Trips'
figsize=(12, 8), cmap=plt.get_cmap('jet'))
plt.title("Uber Trips Analysis")
plt.legend()
plt.show()
```

Loading [MathJax]/extensions/Safe.js




## Summary

So this is how we can analyze the Uber trips for New York City. Some of the conclusions that I got from this analysis are:

1)Monday is the most profitable day for Uber

2)On Saturdays less number of people use Uber

3)6 pm is the busiest day for Uber

4)On average a rise in Uber trips start around 5 am.

5)Most of the Uber trips originate near the Manhattan region in New York.

In [ ]: