uber_analysis_yawar_sofi

July 21, 2023

```
[1]: # 1.. Lets Read data for Analysis
```

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

```
[3]: import os
```

```
[4]: os.listdir(r"C:\Users\Lenovo\OneDrive\Desktop\Datasets")
```

```
[4]: ['other-American_B01362.csv',
      'other-Carmel_B00256.csv',
      'other-Dial7_B00887.csv',
      'other-Diplo_B01196.csv',
      'other-Federal_02216.csv',
      'other-FHV-services_jan-aug-2015.csv',
      'other-Firstclass_B01536.csv',
      'other-Highclass_B01717.csv',
      'other-Lyft_B02510.csv',
      'other-Prestige_B01338.csv',
      'other-Skyline_B00111.csv',
      'Uber-Jan-Feb-FOIL.csv',
      'uber-raw-data-apr14.csv',
      'uber-raw-data-aug14.csv',
      'uber-raw-data-janjune-15.csv',
      'uber-raw-data-janjune-15_sample.csv',
      'uber-raw-data-jul14.csv',
      'uber-raw-data-jun14.csv',
      'uber-raw-data-may14.csv',
      'uber-raw-data-sep14.csv']
```

```
[5]: uber_15 = pd.read_csv(r"C:\Users\Lenovo\OneDrive\Desktop\Datasets/
      ↪uber-raw-data-janjune-15_sample.csv")
```

```
[6]: uber_15.shape
```

```
[6]: (100000, 4)
```

1 2.. Lets Perform Data pre-processing/Data cleaning !

```
check data-type , check missing values , check whether duplicated values or not !
ie Prepare Data for Analysis !
```

[7]:
```
type(uber_15)
```

[7]:
```
pandas.core.frame.DataFrame
```

[8]:
```
uber_15.duplicated().sum()
```

[8]:
```
54
```

[9]:
```
uber_15.drop_duplicates(inplace=True)
```

[10]:
```
uber_15.duplicated().sum()
```

[10]:
```
0
```

[11]:
```
uber_15.shape
```

[11]:
```
(99946, 4)
```

[12]:
```
uber_15.dtypes
```

[12]:
```
Dispatching_base_num    object
Pickup_date             object
Affiliated_base_num     object
locationID               int64
dtype: object
```

[13]:
```
uber_15.isnull().sum()
```

[13]:
```
Dispatching_base_num       0
Pickup_date                0
Affiliated_base_num     1116
locationID                 0
dtype: int64
```

2 3.. Which month have max. Uber pickups in New York City ?¶

[14]:
```
uber_15['Pickup_date'][0]
```

[14]:
```
'2015-05-02 21:43:00'
```

[15]:
```
type(uber_15['Pickup_date'][0])
```

[15]:
```
str
```

```
[16]: uber_15['Pickup_date'] = pd.to_datetime(uber_15['Pickup_date'])
```

```
[17]: uber_15['Pickup_date'].dtype
```

```
[17]: dtype('<M8[ns]')
```

```
[18]: uber_15['Pickup_date'][0]
```

```
[18]: Timestamp('2015-05-02 21:43:00')
```

```
[19]: type(uber_15['Pickup_date'][0])
```

```
[19]: pandas._libs.tslibs.timestamps.Timestamp
```

```
[20]: uber_15.dtypes
```

```
[20]: Dispatching_base_num            object
      Pickup_date             datetime64[ns]
      Affiliated_base_num             object
      locationID                       int64
      dtype: object
```

```
[21]: uber_15
```

```
[21]:       Dispatching_base_num         Pickup_date Affiliated_base_num  locationID
      0                   B02617 2015-05-02 21:43:00              B02764         237
      1                   B02682 2015-01-20 19:52:59              B02682         231
      2                   B02617 2015-03-19 20:26:00              B02617         161
      3                   B02764 2015-04-10 17:38:00              B02764         107
      4                   B02764 2015-03-23 07:03:00              B00111         140
      …                      …                   …                 …          …
      99995               B02764 2015-04-13 16:12:00              B02764         234
      99996               B02764 2015-03-06 21:32:00              B02764          24
      99997               B02598 2015-03-19 19:56:00              B02598          17
      99998               B02682 2015-05-02 16:02:00              B02682          68
      99999               B02764 2015-06-24 16:04:00              B02764         125

      [99946 rows x 4 columns]
```

```
[22]: uber_15['month'] = uber_15['Pickup_date'].dt.month_name()
```

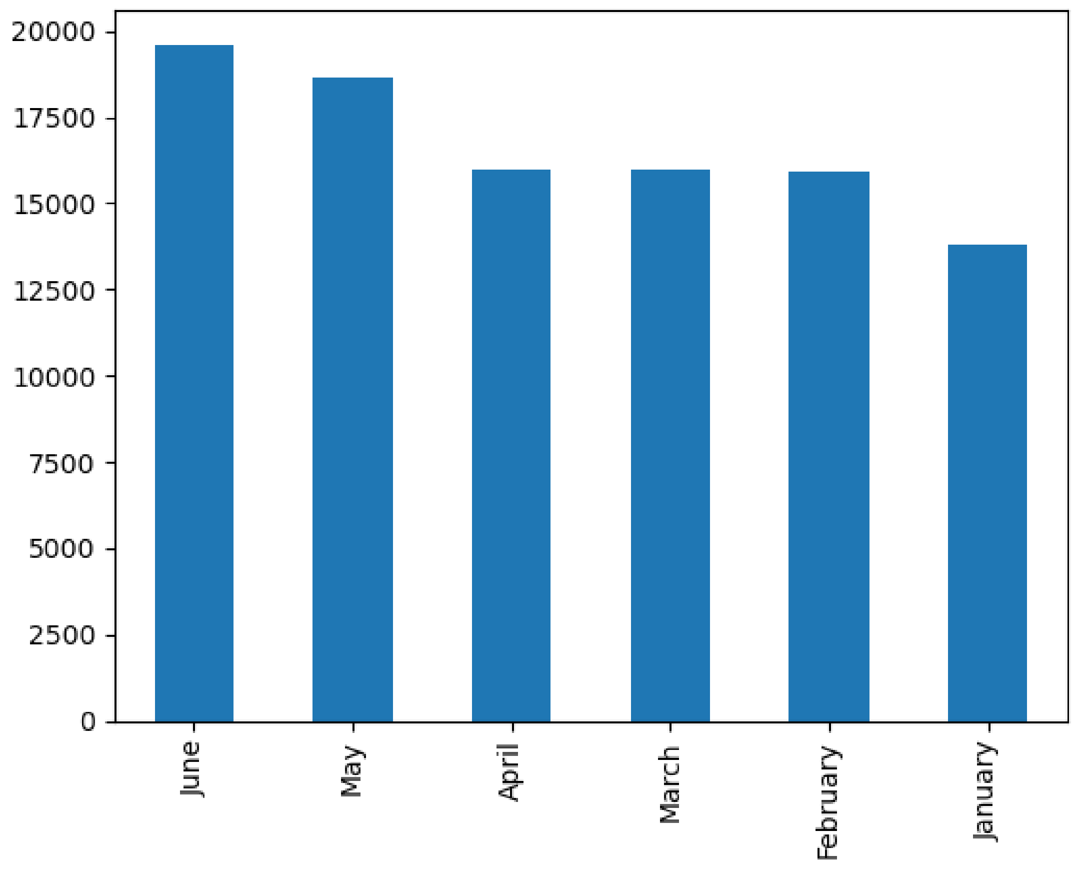
```
[23]: uber_15['month']
```

```
[23]: 0             May
      1         January
      2           March
      3           April
      4           March
```

```
              …
99995      April
99996      March
99997      March
99998        May
99999       June
Name: month, Length: 99946, dtype: object
```

[24]:
```
uber_15['month'].value_counts().plot(kind='bar')
```

[24]:
```
<Axes: >
```



[25]:
```
'''
Inference : June seems to have max Uber Pickups

'''
```

[25]:
```
'\nInference : June seems to have max Uber Pickups \n\n'
```

[26]:
```
## extracting dervied features (weekday ,day ,hour ,month ,minute) from␣
↪'Pickup_date'..

uber_15['weekday'] = uber_15['Pickup_date'].dt.day_name()
uber_15['day'] = uber_15['Pickup_date'].dt.day
uber_15['hour'] = uber_15['Pickup_date'].dt.hour
uber_15['minute'] = uber_15['Pickup_date'].dt.minute
```

[27]:
```
uber_15.head(4)
```

[27]:
```
  Dispatching_base_num         Pickup_date Affiliated_base_num  locationID  \
0               B02617 2015-05-02 21:43:00              B02764         237
1               B02682 2015-01-20 19:52:59              B02682         231
2               B02617 2015-03-19 20:26:00              B02617         161
3               B02764 2015-04-10 17:38:00              B02764         107

     month   weekday  day  hour  minute
0      May  Saturday    2    21      43
1  January   Tuesday   20    19      52
2    March  Thursday   19    20      26
3    April    Friday   10    17      38
```

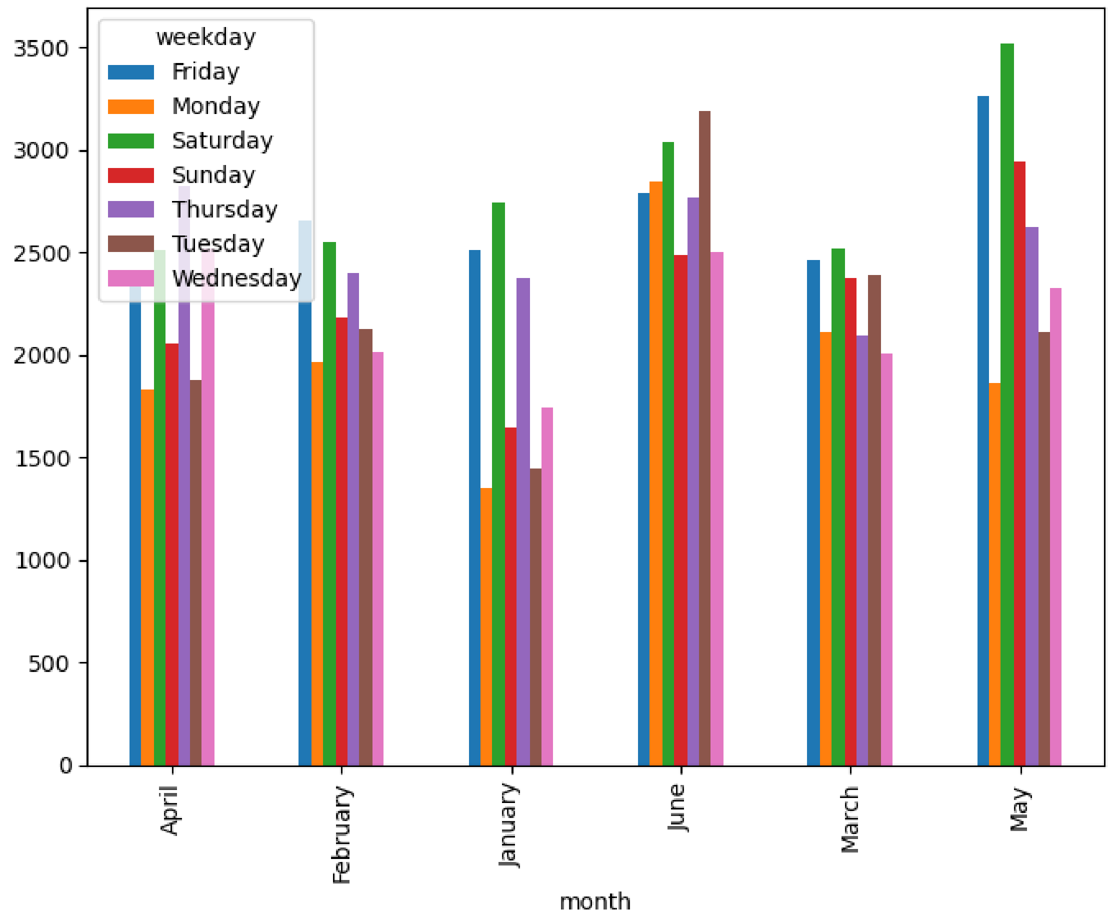
[28]:
```
## pd.crosstab() is used to create pivot table ..

pivot = pd.crosstab(index=uber_15['month'] , columns=uber_15['weekday'])
```

[29]:
```
pivot
```

[29]:
```
weekday   Friday  Monday  Saturday  Sunday  Thursday  Tuesday  Wednesday
month
April       2365    1833      2508    2052      2823     1880       2521
February    2655    1970      2550    2183      2396     2129       2013
January     2508    1353      2745    1651      2378     1444       1740
June        2793    2848      3037    2485      2767     3187       2503
March       2465    2115      2522    2379      2093     2388       2007
May         3262    1865      3519    2944      2627     2115       2328
```

[30]:
```
## grouped-bar plot using Pandas ..
pivot.plot(kind= 'bar' , figsize=(8,6))
```

[30]:
```
<Axes: xlabel='month'>
```



[31]:
```
'''

On Saturday & Friday, u are getting more Uber pickups in each month , it seems␣
↪that New Yorkers used to go for
shopping , Malls , fun activities alot on these days

'''
```

[31]:
```
'\n\nOn Saturday & Friday, u are getting more Uber pickups in each month , it
seems that New Yorkers used to go for \nshopping , Malls , fun activities alot
on these days\n\n'
```

3 4.. Lets Find out Hourly Rush in New york city on all days

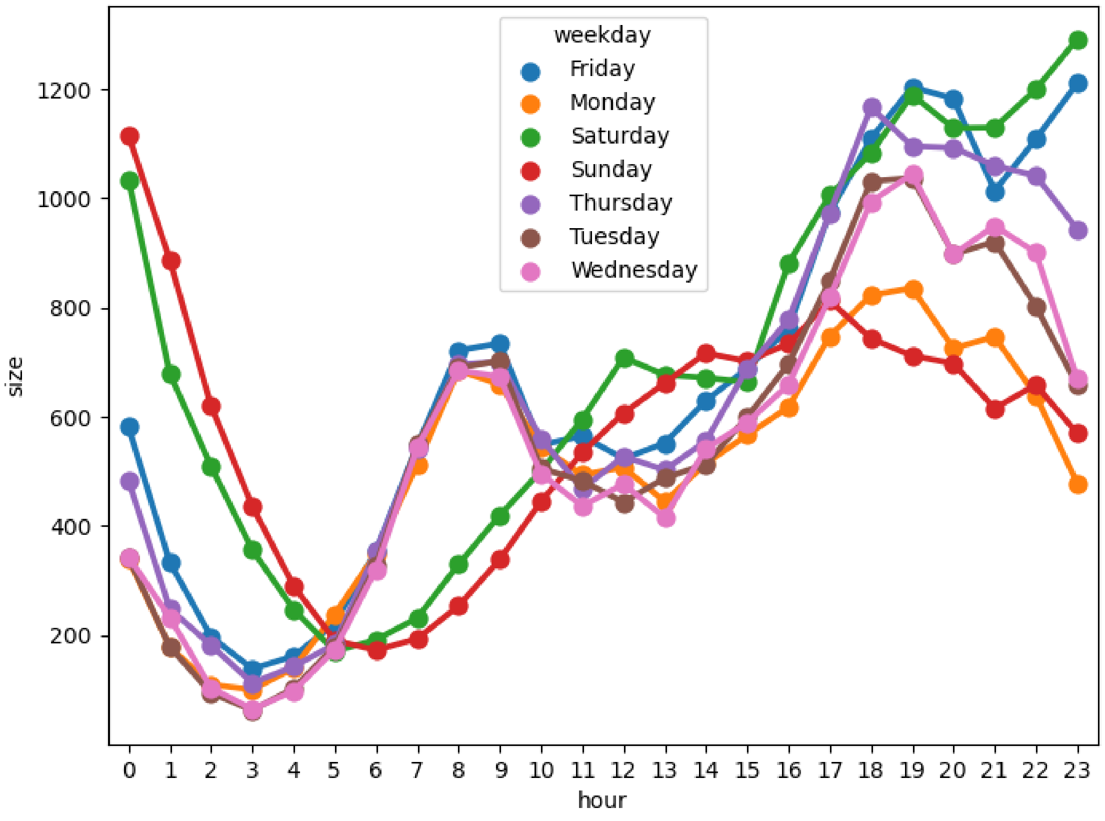
[32]:
```
summary = uber_15.groupby(['weekday' , 'hour'] , as_index=False).size()
```

[33]:
```
summary
```

```
[33]:        weekday  hour  size
      0       Friday     0   581
      1       Friday     1   333
      2       Friday     2   197
      3       Friday     3   138
      4       Friday     4   161
      ..         ...   ...   ...
      163  Wednesday    19  1044
      164  Wednesday    20   897
      165  Wednesday    21   949
      166  Wednesday    22   900
      167  Wednesday    23   669

      [168 rows x 3 columns]
```

```
[34]: ## pointplot between 'hour' & 'size' for all the weekdays..

      plt.figure(figsize=(8,6))
      sns.pointplot(x="hour" , y="size" , hue="weekday" , data=summary)
```

```
[34]: <Axes: xlabel='hour', ylabel='size'>
```



[35]:
```
'''
It's interesting to see that Saturday and Sunday exhibit similar demand␣
 ↪throughout the late night/morning/afternoon,
but it exhibits opposite trends during the evening. In the evening, Saturday␣
 ↪pickups continue to increase throughout the evening,
but Sunday pickups takes a downward turn after evening..

We can see that there the weekdays that has the most demand during the late␣
 ↪evening is Friday and Saturday,
which is expected, but what strikes me is that Thursday nights also exhibits␣
 ↪very similar trends as Friday and Saturday nights.

It seems like New Yorkers are starting their 'weekends' on Thursday nights. :)



'''
```

[35]:
```
"\nIt's interesting to see that Saturday and Sunday exhibit similar demand
throughout the late night/morning/afternoon, \nbut it exhibits opposite trends
during the evening. In the evening, Saturday pickups continue to increase
throughout the evening,\nbut Sunday pickups takes a downward turn after
evening..\n\nWe can see that there the weekdays that has the most demand during
the late evening is Friday and Saturday, \nwhich is expected, but what strikes
me is that Thursday nights also exhibits very similar trends as Friday and
Saturday nights.\n\nIt seems like New Yorkers are starting their 'weekends' on
Thursday nights. :)\n\n\n"
```

4 5.. Which Base_number has most number of Active Vehicles ??

[36]:
```
uber_15.columns
```

[36]:
```
Index(['Dispatching_base_num', 'Pickup_date', 'Affiliated_base_num',
       'locationID', 'month', 'weekday', 'day', 'hour', 'minute'],
      dtype='object')
```

[37]:
```
os.listdir(r"C:\Users\Lenovo\OneDrive\Desktop\Datasets")
```

[37]:
```
['other-American_B01362.csv',
 'other-Carmel_B00256.csv',
 'other-Dial7_B00887.csv',
 'other-Diplo_B01196.csv',
 'other-Federal_02216.csv',
 'other-FHV-services_jan-aug-2015.csv',
 'other-Firstclass_B01536.csv',
 'other-Highclass_B01717.csv',
 'other-Lyft_B02510.csv',
 'other-Prestige_B01338.csv',
 'other-Skyline_B00111.csv',
 'Uber-Jan-Feb-FOIL.csv',
 'uber-raw-data-apr14.csv',
 'uber-raw-data-aug14.csv',
 'uber-raw-data-janjune-15.csv',
 'uber-raw-data-janjune-15_sample.csv',
 'uber-raw-data-jul14.csv',
 'uber-raw-data-jun14.csv',
 'uber-raw-data-may14.csv',
 'uber-raw-data-sep14.csv']
```

[38]:
```
uber_foil = pd.read_csv(r"C:\Users\Lenovo\OneDrive\Desktop\Datasets/
 ↪Uber-Jan-Feb-FOIL.csv")
```

[39]:
```
uber_foil.shape
```

[39]:
```
(354, 4)
```

[40]:
```
uber_foil.head(3)
```

[40]:
```
  dispatching_base_number      date  active_vehicles  trips
0                  B02512  1/1/2015              190   1132
1                  B02765  1/1/2015              225   1765
2                  B02764  1/1/2015             3427  29421
```

[41]:
```
### establishing the entire set-up of Plotly..
```

[42]:
```
import chart_studio.plotly as py
import plotly.graph_objs as go
import plotly.express as px

from plotly.offline import download_plotlyjs , init_notebook_mode , plot , iplot
## iplot() when working in a Jupyter Notebook to display the plot in the␣
 ↪notebook.
## U have to do a proper setup of plotly , otherwise plotly plots gets open in␣
 ↪a web-browser instead of Jupyter notebook
```

[43]:
```
init_notebook_mode(connected=True)
```

[44]:
```
uber_foil.columns
```

[44]:
```
Index(['dispatching_base_number', 'date', 'active_vehicles', 'trips'],
dtype='object')
```

[45]:
```
px.box(x='dispatching_base_number' , y='active_vehicles' , data_frame=uber_foil)
```

```
[46]: ### if u need distribution +  5-summary stats of data , its good to go with␣
       ↪violinplot
      px.violin(x='dispatching_base_number' , y='active_vehicles' ,␣
       ↪data_frame=uber_foil)
```

5 6.. Collect entire data & Make it ready for the Data Analysis..¶

```
[47]: files = os.listdir(r"C:\Users\Lenovo\OneDrive\Desktop\Datasets")[-8:]
```

```
[48]: files.remove('uber-raw-data-janjune-15.csv')
```

```
[49]: files
```

```
[49]: ['uber-raw-data-apr14.csv',
       'uber-raw-data-aug14.csv',
       'uber-raw-data-janjune-15_sample.csv',
       'uber-raw-data-jul14.csv',
       'uber-raw-data-jun14.csv',
       'uber-raw-data-may14.csv',
       'uber-raw-data-sep14.csv']
```

```
[50]: files.remove('uber-raw-data-janjune-15_sample.csv')
```

```
[51]: files
```

```
[51]: ['uber-raw-data-apr14.csv',
       'uber-raw-data-aug14.csv',
       'uber-raw-data-jul14.csv',
       'uber-raw-data-jun14.csv',
       'uber-raw-data-may14.csv',
       'uber-raw-data-sep14.csv']
```

```
[52]: #blank dataframe
      final = pd.DataFrame()

      path = r"C:\Users\Lenovo\OneDrive\Desktop\Datasets"

      for file in files :
          current_df = pd.read_csv(path+'/'+file)
          final = pd.concat([current_df , final])
```

```
[53]: final.shape
```

```
[53]: (4534327, 4)
```

[54]:
```
### After Collecting entire data ,u might ask is : Do we have duplicate entires␣
↪in data ?
### We are going to remove duplicates data when the entire row is duplicated
```

[55]:
```
### first lets figure out total observations where we have duplicate values..
final.duplicated().sum()
```

[55]:
```
82581
```

[56]:
```
## drop duplicate rows ..
final.drop_duplicates(inplace=True)
```

[57]:
```
final.shape
```

[57]:
```
(4451746, 4)
```

[58]:
```
final.head(3)
```

[58]:
```
           Date/Time      Lat      Lon    Base
0  9/1/2014 0:01:00  40.2201 -74.0021  B02512
1  9/1/2014 0:01:00  40.7500 -74.0027  B02512
2  9/1/2014 0:03:00  40.7559 -73.9864  B02512
```

6 Dataset Information :

The dataset contains information about the Datetime, Latitude, Longitude and Base of each uber ride that happened in the month of July 2014 at New York City, USA Date/Time : The date and time of the Uber pickup

```
Lat : The latitude of the Uber pickup

Lon : The longitude of the Uber pickup

Base : The TLC base company code affiliated with the Uber pickup
```

The Base codes are for the following Uber bases: B02512 : Unter B02598 : Hinter B02617 : Weiter B02682 : Schmecken B02764 : Danach-NY

7 7.. at what locations of New York City we are getting rush ??

[59]:
```
### ie where-ever we have more data-points or more density, it means more rush␣
↪is at there !
```

[60]:
```
rush_uber = final.groupby(['Lat' , 'Lon'] , as_index=False).size()
```

```
[61]: rush_uber.head(6)
```

```
[61]:        Lat      Lon  size
      0  39.6569 -74.2258     1
      1  39.6686 -74.1607     1
      2  39.7214 -74.2446     1
      3  39.8416 -74.1512     1
      4  39.9055 -74.0791     1
      5  39.9196 -74.1112     1
```

```
[62]: import folium
```

```
[63]: basemap = folium.Map()
```

```
[64]: basemap
```

```
[64]: <folium.folium.Map at 0x2469a5fd300>
```

```
[65]: from folium.plugins import HeatMap
```

```
[66]: HeatMap(rush_uber).add_to(basemap)
```

```
[66]: <folium.plugins.heat_map.HeatMap at 0x2469a5db220>
```

```
[67]: basemap
```

```
[67]: <folium.folium.Map at 0x2469a5fd300>
```

```
[68]: We can see a number of hot spots here. Midtown Manhattan is clearly a huge␣
        ↪bright spot
      & these are made from Midtown to Lower Manhattan followed by Upper Manhattan␣
        ↪and the Heights of Brooklyn.
```

```
  Cell In[68], line 3

    ^
SyntaxError: invalid non-printable character U+200B
```

8 8.. Examine rush on Hour and Weekday (Perform Pair wise Analysis)¶

```
[ ]: final.columns
```

[ ]:
```
final.head(3)
```

[ ]:
```
final.dtypes
```

[ ]:
```
final['Date/Time'][0]
```

[ ]:
```
### converting 'Date/Time' feature into date-time..
final['Date/Time'] = pd.to_datetime(final['Date/Time'] , format="%m/%d/%Y %H:%M:
 ↪%S")
```

[ ]:
```
final['Date/Time']
```

[ ]:
```
### extracting 'weekday' & 'hour' from 'Date/Time' feature.

final['day'] = final['Date/Time'].dt.day
final['hour'] = final['Date/Time'].dt.hour
```

[ ]:
```
final.head(4)
```

[ ]:
```
'''
Earlier we have learnt how to create pivot table using pd.crosstab() , now let␣
 ↪me show u one more way to build
pivot_table without pd.crosstab()

'''
```

[ ]:
```
pivot = final.groupby(['day' , 'hour']).size().unstack()
```

[ ]:
```
pivot
### pivot table is all about  , we have Rows*columns & having value in each␣
 ↪cell !
```

9 9.. How to Automate Your Analysis..?

[ ]:
```
### styling dataframe
pivot.style.background_gradient()
```

[ ]:
```
## creating a user-defined function..

def gen_pivot_table(df , col1 , col2):

    pivot = final.groupby([col1 , col2]).size().unstack()
    return pivot.style.background_gradient()

```

[ ]:
```
final.columns
```

[ ]:
```
gen_pivot_table(final , "day" , "hour")
```

[ ]:
```
```