```
#LIBRARIES
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import chart_studio.plotly as py
import plotly.graph_objs as go
import plotly.express as px
import folium

from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
```

```
import os # to call files or use files and datasets from the local drive
```

```
os.listdir(r"N:\datasets\Datasets") #this list calls all the files from the os
```

```
['.ipynb_checkpoints',
 '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',
 'Untitled1.ipynb']
```

here R or r is used to read all the files even if there is error is in backslash or frontslash

```
df = pd.read_csv(r"N:\datasets\Datasets/uber-raw-data-janjune-15_sample.csv")
```

```
df #print the data frame
```

	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

100000 rows × 4 columns

```
df.shape #printing the shape to check the data frame
```

```
(100000, 4)
```

Data preprocessing ot data cleaning

```
df.duplicated
```

```
<bound method DataFrame.duplicated of       Dispatching_base_num         Pickup_date Affiliated_base_num  \
0                   B02617  2015-05-02 21:43:00              B02764
1                   B02682  2015-01-20 19:52:59              B02682
2                   B02617  2015-03-19 20:26:00              B02617
3                   B02764  2015-04-10 17:38:00              B02764
4                   B02764  2015-03-23 07:03:00              B00111
...                    ...                  ...                 ...
99995               B02764  2015-04-13 16:12:00              B02764
99996               B02764  2015-03-06 21:32:00              B02764
99997               B02598  2015-03-19 19:56:00              B02598
99998               B02682  2015-05-02 16:02:00              B02682
99999               B02764  2015-06-24 16:04:00              B02764

       locationID
0             237
1             231
2             161
3             107
4             140
...           ...
99995         234
99996          24
99997          17
99998          68
99999         125

[100000 rows x 4 columns]>
```

df.duplicated() is typically used when you want to directly access the boolean Series indicating duplicates.

```
df.duplicated() # when you add paranthesis it consideres itself into two dimensional true or flase types
```

```
0        False
1        False
2        False
3        False
4        False
         ...
99995    False
99996    False
99997    False
99998    False
99999    False
Length: 100000, dtype: bool
```

df.duplicated().sum() is used to count the total number of duplicate rows in the DataFrame, which can be useful for reporting or further processing

```
df.duplicated().sum()
```

```
54
```

drop_duplicates,To drop all the duplicates and here inplace refers as to whatever places it is false it make true and rearrages the dataframe

```
df.drop_duplicates(inplace=True)
```

```
df1 = df.copy()
```

```
df1.drop_duplicates(inplace = True)
```

```
df1.shape # recheck again if the duplicates are removed and shape is
```

```
(99946, 4)
```

```
df1
```

	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 × 4 columns

Here im checking what kind of data is availabe in each type of columns

```
df1.dtypes
```

```
Dispatching_base_num    object
Pickup_date             object
Affiliated_base_num     object
locationID               int64
dtype: object
```

Here im checking if isnull is making meaning if there are any null values

```
df1.isnull().sum()
```

```
Dispatching_base_num       0
Pickup_date                0
Affiliated_base_num     1116
locationID                 0
dtype: int64
```

```
df1.isnull()
```

	Dispatching_base_num	Pickup_date	Affiliated_base_num	locationID
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
...	...	...	...	...
99995	False	False	False	False
99996	False	False	False	False
99997	False	False	False	False
99998	False	False	False	False
99999	False	False	False	False

99946 rows × 4 columns

Checking the date and time

```
df1['Pickup_date'][0]
```

```
'2015-05-02 21:43:00'
```

Also checking the type of the date and time is it a str or int or it should be in datetime format

```
type(df1['Pickup_date'][0])
```

```
str
```

This is the way to convert a pandas to datetime pd.to_datetime(df['Pickup_date'])

```
pd.to_datetime(df1['Pickup_date'])
```

```
0       2015-05-02 21:43:00
1       2015-01-20 19:52:59
2       2015-03-19 20:26:00
3       2015-04-10 17:38:00
4       2015-03-23 07:03:00
               ...
99995   2015-04-13 16:12:00
99996   2015-03-06 21:32:00
99997   2015-03-19 19:56:00
99998   2015-05-02 16:02:00
99999   2015-06-24 16:04:00
Name: Pickup_date, Length: 99946, dtype: datetime64[ns]
```

```
df1['Pickup_date'] = pd.to_datetime(df1['Pickup_date'])
```

```
df1['Pickup_date'].dtype
```

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

```
df1['Pickup_date'][0] #So now its got converted to a timestamp for the normal time
```

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

```
type(df1['Pickup_date'][0]) # now the pandas is converted it to the timestamp
```

```
pandas._libs.tslibs.timestamps.Timestamp
```

```
df1.dtypes #here now if you see the pickupdate is converted from an object to timestamp
```

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

Categrorical data can be either object or bool it can be taken as groups/ categroies. numerical data is either integer which is discrete, float which is a continuous

Unsupported Cell Type. Double-Click to inspect/edit the content.

problem statement which month is the highest pickup of uber in NEW YORK

```
df1
```

	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 × 4 columns

dt.month is used to return months in the form of numbers

```
df1['Pickup_date'].dt.month
```

```
0        5
1        1
2        3
3        4
4        3
        ..
99995    4
99996    3
99997    3
99998    5
99999    6
Name: Pickup_date, Length: 99946, dtype: int32
```

```
df1['month'] = df1['Pickup_date'].dt.month_name()# it prints all the months name
```

```
df1['month']
```

```
0            May
1        January
2          March
3          April
```

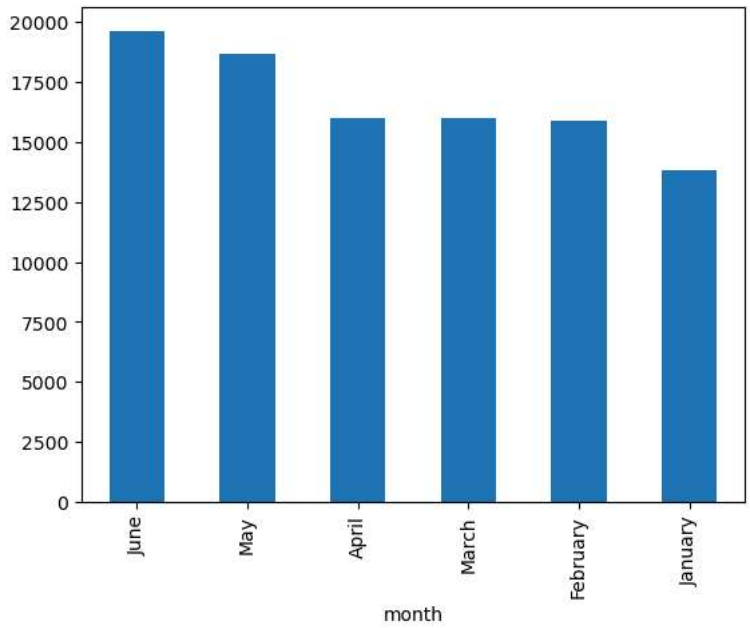
```
4          March
          ...
99995      April
99996      March
99997      March
99998        May
99999       June
Name: month, Length: 99946, dtype: object
```

```
df1['month'].value_counts()
```

```
month
June        19620
May         18660
April       15982
March       15969
February    15896
January     13819
Name: count, dtype: int64
```

```
df1['month'].value_counts().plot(kind = 'bar')
```

```
<Axes: xlabel='month'>
```



for different days to execute

```
df1['Pickup_date'].dt.day_name()
```

```
0         Saturday
1          Tuesday
2         Thursday
3           Friday
4           Monday
            ...
99995       Monday
99996       Friday
99997     Thursday
99998     Saturday
99999    Wednesday
Name: Pickup_date, Length: 99946, dtype: object
```

for hourly

```
df1['Pickup_date'].dt.hour
```

```
0        21
1        19
```

```
2        20
3        17
4         7
         ..
99995    16
99996    21
99997    19
99998    16
99999    16
Name: Pickup_date, Length: 99946, dtype: int32
```

for every minute

```
df1['Pickup_date'].dt.minute
```

```
0        43
1        52
2        26
3        38
4         3
         ..
99995    12
99996    32
99997    56
99998     2
99999     4
Name: Pickup_date, Length: 99946, dtype: int32
```

printing the first 5 values

```
df1.head(5)
```

	Dispatching_base_num	Pickup_date	Affiliated_base_num	locationID	month
0	B02617	2015-05-02 21:43:00	B02764	237	May
1	B02682	2015-01-20 19:52:59	B02682	231	January
2	B02617	2015-03-19 20:26:00	B02617	161	March
3	B02764	2015-04-10 17:38:00	B02764	107	April
4	B02764	2015-03-23 07:03:00	B00111	140	March

printing the last five values

```
df1.tail(5)
```

	Dispatching_base_num	Pickup_date	Affiliated_base_num	locationID	month
99995	B02764	2015-04-13 16:12:00	B02764	234	April
99996	B02764	2015-03-06 21:32:00	B02764	24	March
99997	B02598	2015-03-19 19:56:00	B02598	17	March
99998	B02682	2015-05-02 16:02:00	B02682	68	May
99999	B02764	2015-06-24 16:04:00	B02764	125	June

```
df1['Day_of_Week'] = df1['Pickup_date'].dt.day_name()
```

to do cross tabulation

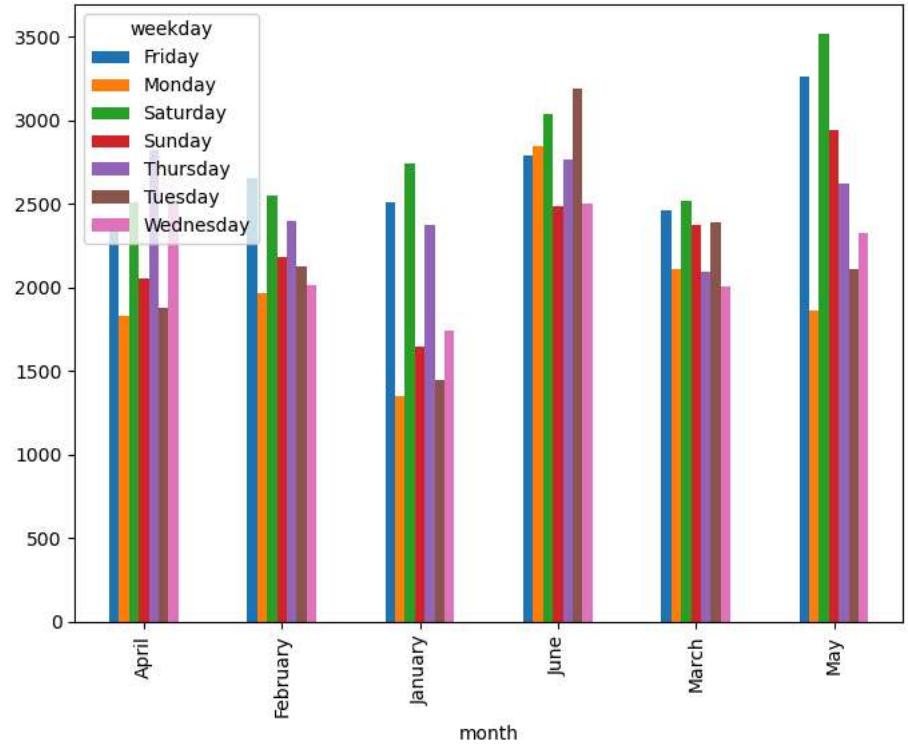
```
df1['weekday'] = df1['Pickup_date'].dt.day_name()
pivot = pd.crosstab(index=df1['month'], columns=df1['weekday'])
```

```
1 pivot
```

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

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

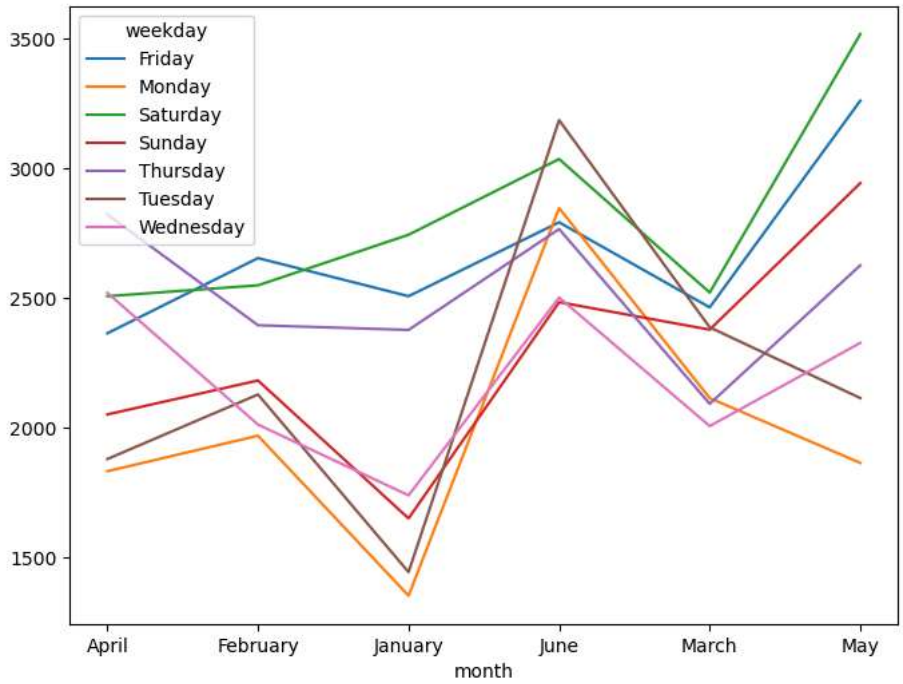
```
<Axes: xlabel='month'>
```



Analying the rush hour in new york [Data Analysis]

```
1 pivot.plot(kind='line', figsize=(8,6))
```

```
<Axes: xlabel='month'>
```




```
df1['hour'] = df1['Pickup_date'].dt.hour
```

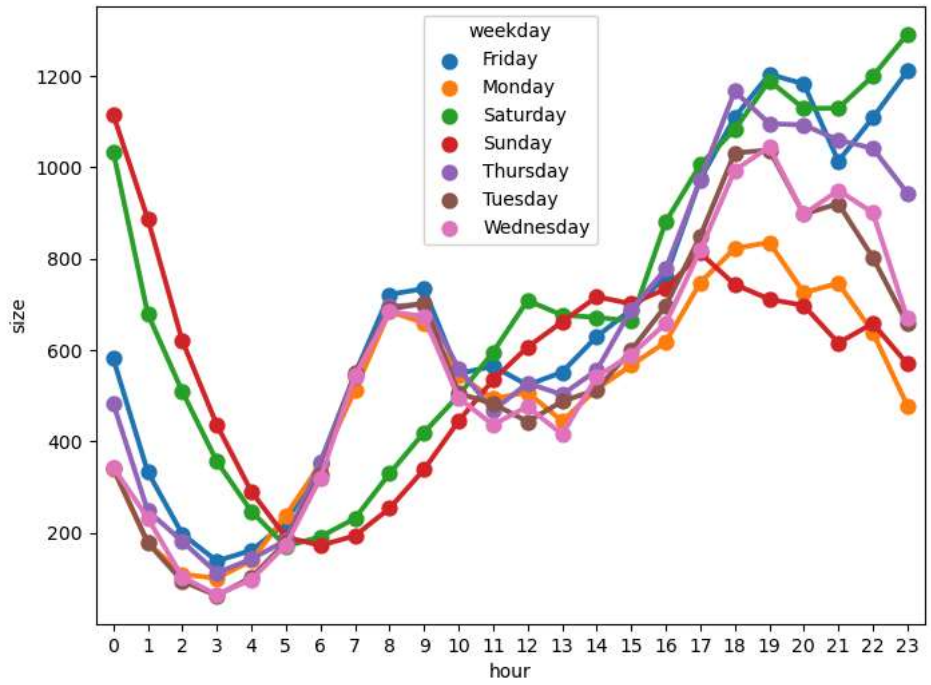
```
summary = df1.groupby(['weekday', 'hour'], as_index=False).size()
```

```
summary
```

	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 × 3 columns

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



Which base number has the most amount of active vechiles??

```
df1.columns
```

```
Index(['Dispatching_base_num', 'Pickup_date', 'Affiliated_base_num',
       'locationID', 'month', 'Day_of_Week', 'weekday', 'hour'],
      dtype='object')
```

```
os.listdir(r"N:\datasets\Datasets") #this list calls all the files from the os
```

```
['.ipynb_checkpoints',
 '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',
 'Untitled1.ipynb']
```

```
df_foil = pd.read_csv(r'N:\datasets\Datasets\Uber-Jan-Feb-FOIL.csv')
```

```
df_foil
```

	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
3	B02682	1/1/2015	945	7679
4	B02617	1/1/2015	1228	9537
...	...	...	...	...
349	B02764	2/28/2015	3952	39812
350	B02617	2/28/2015	1372	14022
351	B02682	2/28/2015	1386	14472
352	B02512	2/28/2015	230	1803
353	B02765	2/28/2015	747	7753

354 rows × 4 columns

```
df_foil.head()
```

	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
3	B02682	1/1/2015	945	7679
4	B02617	1/1/2015	1228	9537

```
init_notebook_mode(connected=True)
```

```
df_foil.columns
```

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

BOXPLOT USING PLOTLY

```
px.box(x='dispatching_base_number', y='active_vehicles', data_frame=df_foil)
```

Violin plot -> which means distance + box

```
px.violin(x='dispatching_base_number', y='active_vehicles', data_frame=df_foil)
```

```
os.listdir(r"N:\datasets\Datasets") #this list calls all the files from the os
```

```
['.ipynb_checkpoints',
 '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',
 'Untitled1.ipynb']
```

Now lets combine data into one big data or some new data file

```
files = os.listdir(r"N:\datasets\Datasets")[-9:-1]
files
```

```
['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']
```

```
files.remove('uber-raw-data-janjune-15.csv')
files
```

```
['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']
```

```
files.remove('uber-raw-data-janjune-15_sample.csv')
files
```

```
['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']
```

Concating all these files together

```
final_df = pd.DataFrame()

path = r"N:\datasets\Datasets"

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

```
final_df.shape
```

```
(4534327, 4)
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
final_df.duplicated().sum()
```

```
82581
```

```
final_df.drop_duplicates(inplace = True)
```

```
1 final_df.shape
```

```
(4451746, 4)
```

```
1 final_df
```

	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
3	9/1/2014 0:06:00	40.7450	-73.9889	B02512
4	9/1/2014 0:11:00	40.8145	-73.9444	B02512
...	...	...	...	...
564511	4/30/2014 23:22:00	40.7640	-73.9744	B02764
564512	4/30/2014 23:26:00	40.7629	-73.9672	B02764
564513	4/30/2014 23:31:00	40.7443	-73.9889	B02764
564514	4/30/2014 23:32:00	40.6756	-73.9405	B02764
564515	4/30/2014 23:48:00	40.6880	-73.9608	B02764

4451746 rows × 4 columns

At what locations of new york city we are getting rush

```
1 rush_uber = final_df.groupby(['Lat','Lon'], as_index=False).size()
```

Folium is a map

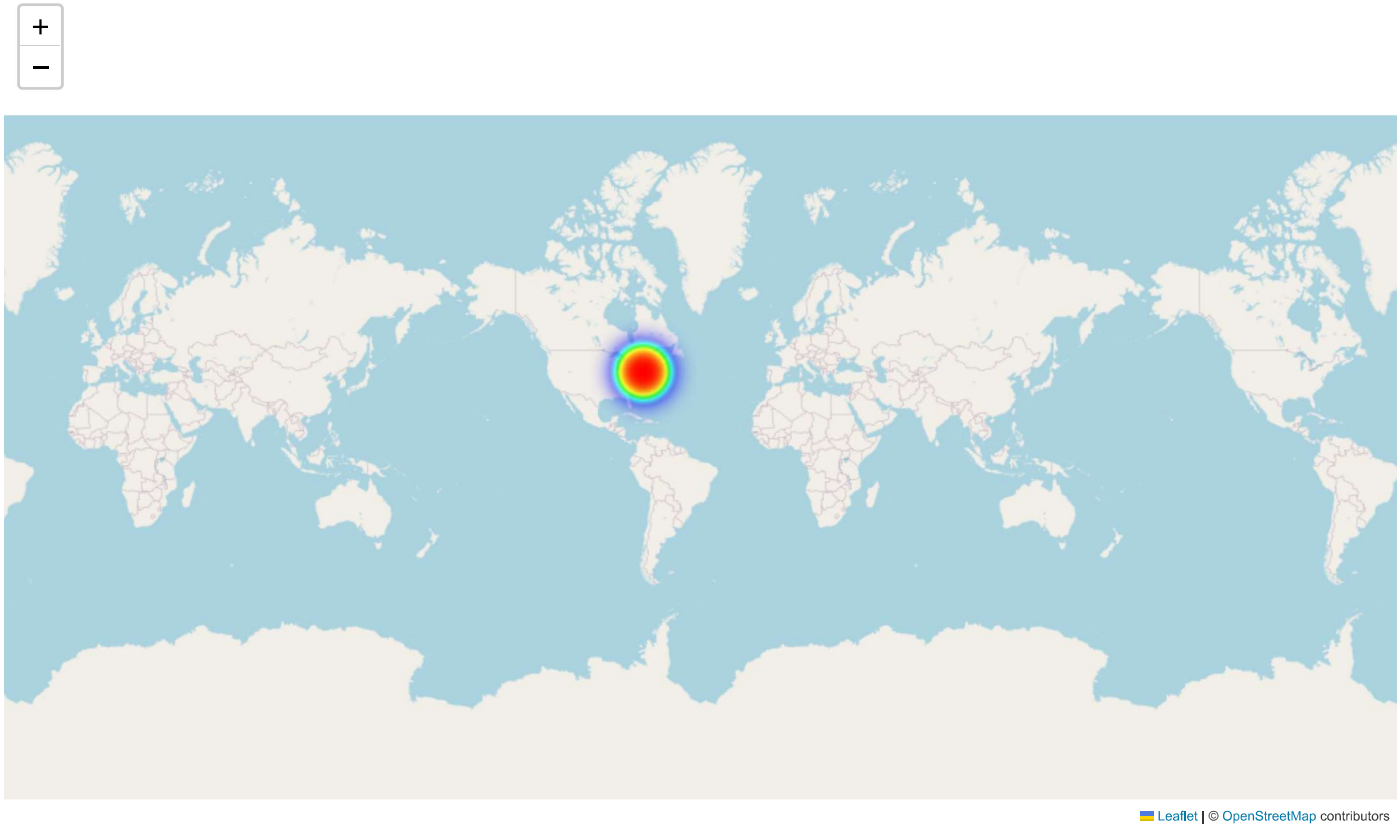
```
1 basemap = folium.Map()
2 basemap
```


```
1 from folium.plugins import HeatMap
```

```
1 HeatMap(rush_uber).add_to(basemap)
```

```
<folium.plugins.heat_map.HeatMap at 0x2810b5952d0>
```

```
1 basemap
```



Examine Rush on hour and weekday(Performing pairwise analysis)

```
1 final_df.columns
```

```
Index(['Date/Time', 'Lat', 'Lon', 'Base'], dtype='object')
```

```
1 final_df.head()
```

	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
3	9/1/2014 0:06:00	40.7450	-73.9889	B02512
4	9/1/2014 0:11:00	40.8145	-73.9444	B02512

```
1 final_df.dtypes
```

```
Date/Time     object
Lat          float64
Lon          float64
Base          object
dtype: object
```

```
1 final_df['Date/Time'][0] # here we will get 6 indexs because we have concat 6.csv files for 0 right so
```

```
0    9/1/2014 0:01:00
0    5/1/2014 0:02:00
0    6/1/2014 0:00:00
0    7/1/2014 0:03:00
0    8/1/2014 0:03:00
```

```
0    4/1/2014 0:11:00
Name: Date/Time, dtype: object
```

```
final_df['Date/Time']=pd.to_datetime(final_df['Date/Time'], format="%m/%d/%Y %H:%M:%S")
final_df['Date/Time']
```

```
0        2014-09-01 00:01:00
1        2014-09-01 00:01:00
2        2014-09-01 00:03:00
3        2014-09-01 00:06:00
4        2014-09-01 00:11:00
                 ...
564511   2014-04-30 23:22:00
564512   2014-04-30 23:26:00
564513   2014-04-30 23:31:00
564514   2014-04-30 23:32:00
564515   2014-04-30 23:48:00
Name: Date/Time, Length: 4451746, dtype: datetime64[ns]
```

```
final_df['Date/Time'].dtype
```

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

for day

```
final_df['day'] = final_df['Date/Time'].dt.day
final_df['day']
```

```
0          1
1          1
2          1
3          1
4          1
          ..
564511    30
564512    30
564513    30
564514    30
564515    30
Name: day, Length: 4451746, dtype: int32
```

for hour

```
final_df['hour'] = final_df['Date/Time'].dt.hour
final_df['hour']
```

```
0          0
1          0
2          0
3          0
4          0
          ..
564511    23
564512    23
564513    23
564514    23
564515    23
Name: hour, Length: 4451746, dtype: int32
```

```
final_df.head()
```

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

```
1 final_df.groupby(['day','hour']). size()
```

```
day  hour
1    0       3178
     1       1944
     2       1256
     3       1308
     4       1429
             ...
31   19      4898
     20      4819
     21      5064
     22      5164
     23      3961
Length: 744, dtype: int64
```

```
1 final_df.groupby(['day','hour'], as_index=False). size()
```

	day	hour	size
0	1	0	3178
1	1	1	1944
2	1	2	1256
3	1	3	1308
4	1	4	1429
...	...	...	...
739	31	19	4898
740	31	20	4819
741	31	21	5064
742	31	22	5164
743	31	23	3961

744 rows × 3 columns

```
1 pivot1 = final_df.groupby(['day','hour']). size().unstack()
2 pivot1
```

hour	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19	20	21	22	23
day																					
1	3178	1944	1256	1308	1429	2126	3664	5380	5292	4617	...	6933	7910	8633	9511	8604	8001	7315	7803	6268	4050
2	2435	1569	1087	1414	1876	2812	4920	6544	6310	4712	...	6904	8449	10109	11100	11123	9474	8759	8357	6998	5160
3	3354	2142	1407	1467	1550	2387	4241	5663	5386	4657	...	7226	8850	10314	10491	11239	9599	9026	8531	7142	4686
4	2897	1688	1199	1424	1696	2581	4592	6029	5704	4744	...	7158	8515	9492	10357	10259	9097	8358	8649	7706	5130
5	2733	1541	1030	1253	1617	2900	4814	6261	6469	5530	...	6955	8312	9609	10699	10170	9430	9354	9610	8853	6518