Basic Information

- The rapid growth of data collection has led to a new era of information.
- Data is being used to create more efficient systems and this is where Recommendation Systems come into play.
- They are used to pick up cycle from near by them and Drop Any near by station
- Where do riders go?
- When do they ride?
- How far do they go?
- Which stations are most popular?
- What days of the week are most rides taken on?
- We've heard all of these questions and we're glad to provide the data that shows you the answers from our first trip to today.

Objective

- In this System we will building a bike sharing System Using **Capitalbikeshare-tripdata**
- For the company like capital bikeshare we will provide something new which can help them to manage their business to increase profit and save time.

Read Data form CSV file

Station

```
import numpy as np
import pandas as pd

from subprocess import check_output
stations = pd.read_csv('C:/Users/Kevin/Desktop/Dataset/station.csv')
trips = pd.read_csv('C:/Users/Kevin/Desktop/Dataset/trip.csv', error_bad_lines=False)

weathers = pd.read_csv('C:/Users/Kevin/Desktop/Dataset/weather.csv')

stations.head(5)
```

trip

```
In [39]: import numpy as np
         import pandas as pd

         from subprocess import check_output
         stations = pd.read_csv('C:/Users/Raju/Desktop/dataset/station.csv')
         trips = pd.read_csv('C:/Users/Raju/Desktop/dataset/trip.csv', error_bad_lines=False)

         weathers = pd.read_csv('C:/Users/Raju/Desktop/dataset/weather.csv')



         trips.head(5)
```

weather

In [43]:
```
import numpy as np
import pandas as pd

from subprocess import check_output
stations = pd.read_csv('C:/Users/Raju/Desktop/dataset/station.csv')
trips = pd.read_csv('C:/Users/Raju/Desktop/dataset/trip.csv', error_bad_lines=False)

weathers = pd.read_csv('C:/Users/Raju/Desktop/dataset/weather.csv')


weathers.head(5)
```

Display Data From CSV file

station

	station_id	name	lat	long	install_date	install_dockcount	modification_date	current_dockcount	decommission_date
0	BT-01	3rd Ave & Broad St	47.618418	-122.350964	10/13/2014	18	NaN	18	NaN
1	BT-03	2nd Ave & Vine St	47.615829	-122.348564	10/13/2014	16	NaN	16	NaN
2	BT-04	6th Ave & Blanchard St	47.616094	-122.341102	10/13/2014	16	NaN	16	NaN
3	BT-05	2nd Ave & Blanchard St	47.613110	-122.344208	10/13/2014	14	NaN	14	NaN
4	CBD-03	7th Ave & Union St	47.610731	-122.332447	10/13/2014	20	NaN	20	NaN

trip

	trip_id	starttime	stoptime	bikeid	tripduration	from_station_name	to_station_name	from_station_id	to_station_id	usertype	gender	birthyear
0	431	10/13/2014 10:31	10/13/2014 10:48	SEA00298	985.935	2nd Ave & Spring St	Occidental Park / Occidental Ave S & S Washing...	CBD-06	PS-04	Member	Male	1960.0
1	432	10/13/2014 10:32	10/13/2014 10:48	SEA00195	926.375	2nd Ave & Spring St	Occidental Park / Occidental Ave S & S Washing...	CBD-06	PS-04	Member	Male	1970.0
2	433	10/13/2014 10:33	10/13/2014 10:48	SEA00486	883.831	2nd Ave & Spring St	Occidental Park / Occidental Ave S & S Washing...	CBD-06	PS-04	Member	Female	1988.0

weather

	Date	Max_Temperature_F	Mean_Temperature_F	Min_TemperatureF	Max_Dew_Point_F	MeanDew_Point_F	Min_Dewpoint_F	Max_Humidity	Mean_Humi
0	10/13/2014	71	62.0	54	55	51	46	87	
1	10/14/2014	63	59.0	55	52	51	50	88	
2	10/15/2014	62	58.0	54	53	50	46	87	
3	10/16/2014	71	61.0	52	49	46	42	83	
4	10/17/2014	64	60.0	57	55	51	41	87	

Fields of Dataset:

The first dataset(TRIP) contains the following features:

- trip_id: numeric ID of bike trip taken
- starttime: day and time trip started, in PST
- stoptime: day and time trip ended, in PST
- bikeid: ID attached to each bike
- tripduration: time of trip in seconds
- from_station_name: name of station where trip originated
- to_station_name: name of station where trip terminated
- from_station_id: ID of station where trip originated
- to_station_id: ID of station where trip terminated
- usertype: "Short-Term Pass Holder" is a rider who

purchased a 24-Hour or 3-Day Pass; "Member" is a rider

who purchased a Monthly or an Annual Membership

- gender: gender of rider
- birthyear: birth year of rider

The Second dataset STATION contains the following features:

- station_id: station ID number
- name: name of station
- lat: station latitude
- long: station longitude
- install_date: date that station was placed in service
- install_dockcount: number of docks at each station on the

installation date

- modification_date: date that station was modified, resulting in a change in location or dock count
- current_dockcount: number of docks at each station on

8/31/2016

- decommission_date: date that station was placed out of

service

The third dataset WEATHER contains the following features:

- Weather dataset contains daily weather information in the service area

Total Records and Attributes of Dataset

In [76]:
```
#print stations total Record,Attribute
print("Record,Attribute : ",stations.shape)
```

```
Record,Attribute :  (58, 9)
```

In [77]:
```
#print trips total Records & Attributes
print("Record,Attribute : ",trips.shape)
```

```
Record,Attribute :  (286857, 12)
```

In [78]:
```
#print weathers total Records & Attributes
print("Record,Attribute : ",weathers.shape)
```

```
Record,Attribute :  (689, 21)
```

Find mean quantile and shape (Trip)

In [47]:
```
trips.describe()
```

Out[47]:

	trip_id	tripduration	birthyear
count	286858.000000	286858.000000	181554.000000
mean	112431.781746	1178.354284	1979.759107
std	76565.086482	2038.697070	10.167110
min	431.000000	60.008000	1931.000000
25%	43051.000000	387.925750	1974.000000
50%	103486.500000	624.846500	1983.000000
75%	179544.750000	1118.483250	1987.000000
max	255245.000000	28794.398000	1999.000000

Find mean quantile and shape (Station)

```
In [46]: stations.describe()
```

Out[46]:

	lat	long	install_dockcount	current_dockcount
count	58.000000	58.000000	58.000000	58.000000
mean	47.624796	-122.327242	17.586207	16.517241
std	0.019066	0.014957	3.060985	5.117021
min	47.598488	-122.355230	12.000000	0.000000
25%	47.613239	-122.338735	16.000000	16.000000
50%	47.618591	-122.328207	18.000000	18.000000
75%	47.627712	-122.316691	18.000000	18.000000
max	47.666145	-122.284119	30.000000	26.000000

Find mean quantile and shape (Weather)

```
In [44]: weathers.describe()
```

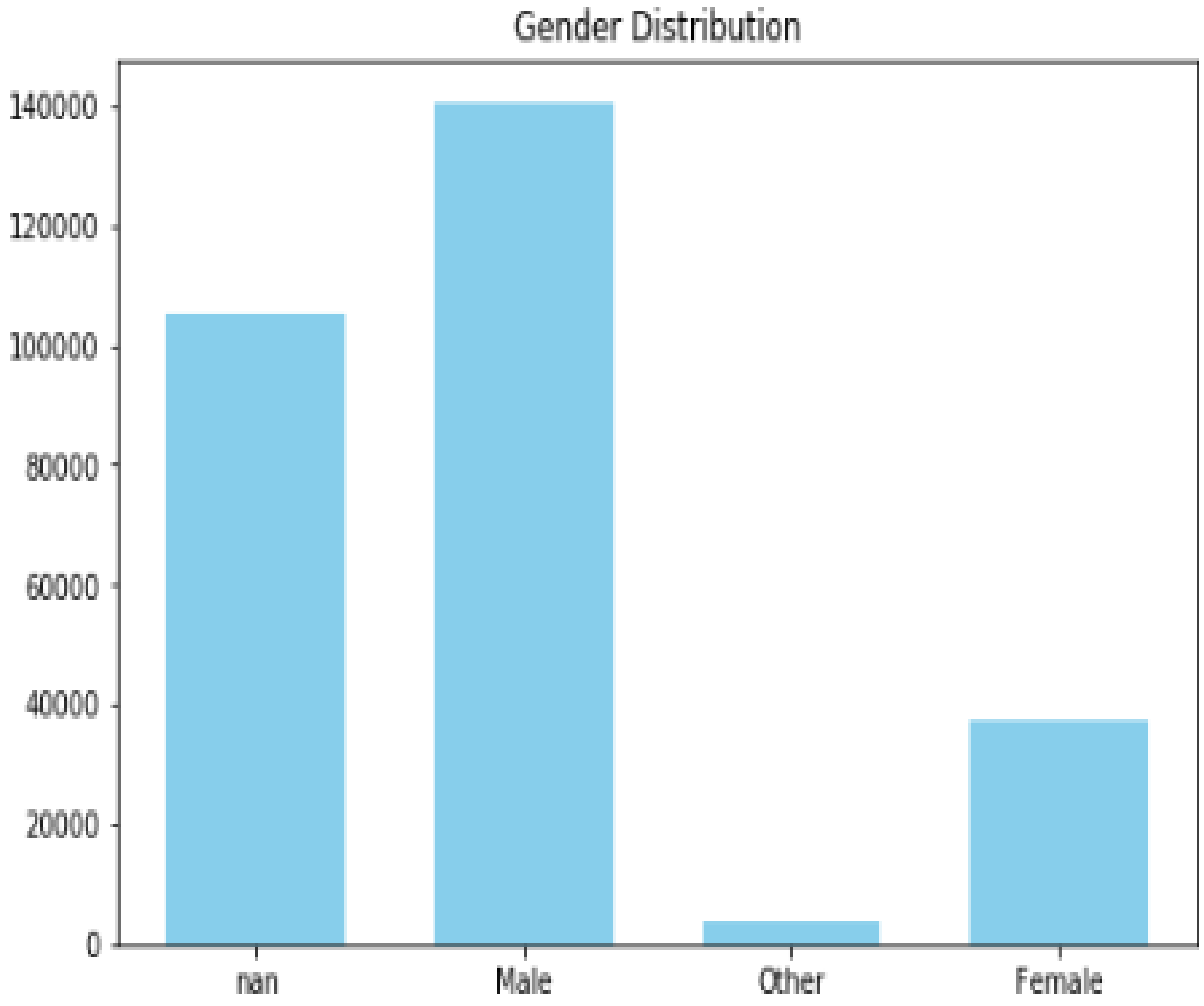
Out[44]:

	Max_Temperature_F	Mean_Temperature_F	Min_TemperatureF	Max_Dew_Point_F	MeanDew_Point_F	Min_Dewpoint_F	Max_Humidity	Mean_Humidity	Mi
count	689.000000	688.000000	689.000000	689.000000	689.000000	689.000000	689.000000	689.000000	
mean	64.027576	56.584302	49.454282	48.571843	45.021771	40.873730	84.541364	68.506531	
std	12.427843	10.408058	9.451437	7.501230	7.914025	8.854608	9.718948	12.701871	
min	39.000000	33.000000	23.000000	10.000000	4.000000	1.000000	40.000000	24.000000	
25%	55.000000	48.000000	43.000000	44.000000	41.000000	36.000000	78.000000	60.000000	
50%	63.000000	56.000000	50.000000	50.000000	46.000000	42.000000	86.000000	70.000000	
75%	73.000000	65.000000	57.000000	54.000000	51.000000	47.000000	90.000000	79.000000	
max	98.000000	83.000000	70.000000	77.000000	59.000000	57.000000	100.000000	95.000000	

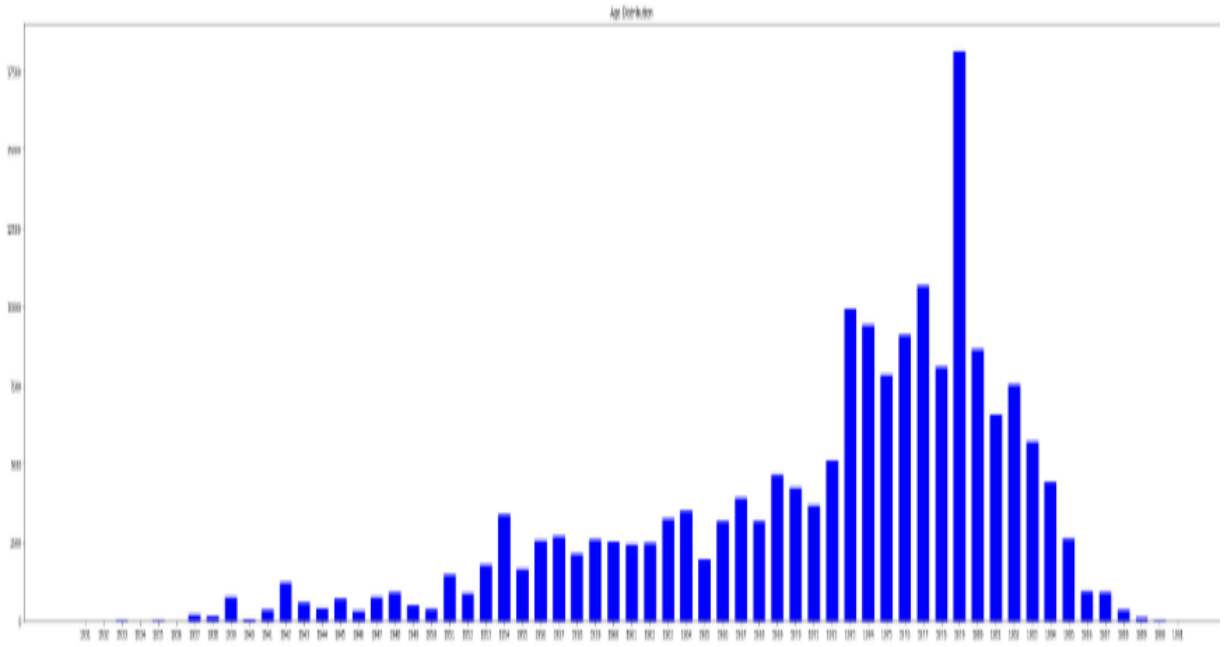
Bike Trip Graph (Gender)

```
import matplotlib.pyplot as plt

trips.gender

from collections import defaultdict

# count genders
counts = defaultdict(int)
for e in trips.gender:
    counts[e] += 1
```

```
# Gender distribution
y = [e[1] for e in counts.items()][:4]
x = range(len(y))
cols = [e[0] for e in counts.items()][:4]

plt.figure(figsize=(8,5))
plt.bar(x, y, color='skyblue', width=1/1.5)
plt.xticks(x, cols)
plt.title('Gender Distribution')
plt.show()
```

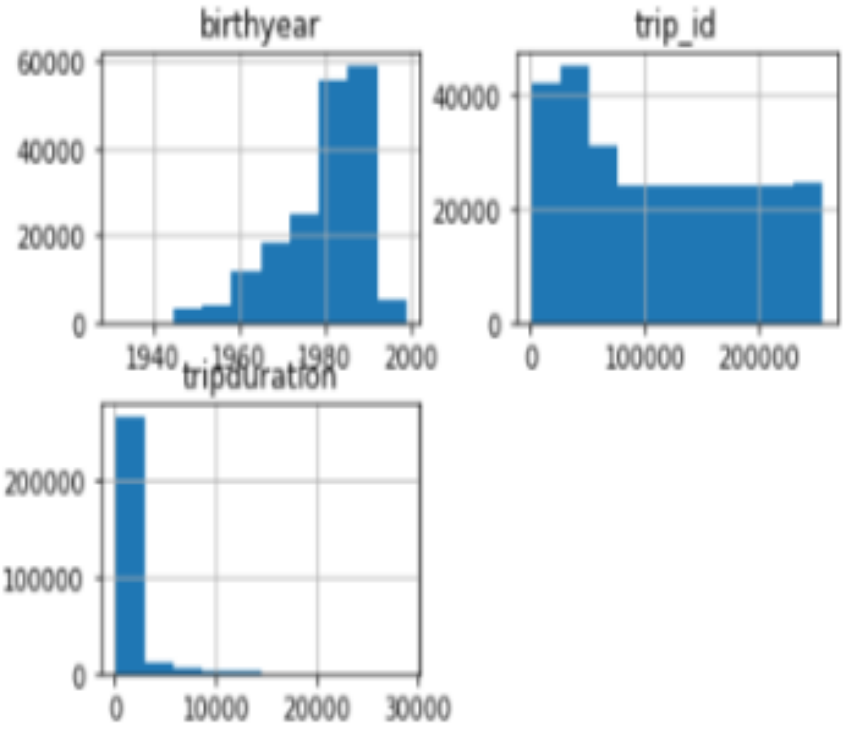

Gender Distribution
140000
120000
100000
80000
60000
40000
20000
0
nan
Male
Other
Female

Bike Trip Graph (Age Distribution)



Trip Data Histogram

```
In [79]: trips.hist()
         plt.show()
```



Data manipulation on Trip Dataset

- drop the id columns that that are unnecessary
- trips.drop(['trip_id','bikeid'], axis = 1, inplace = True)

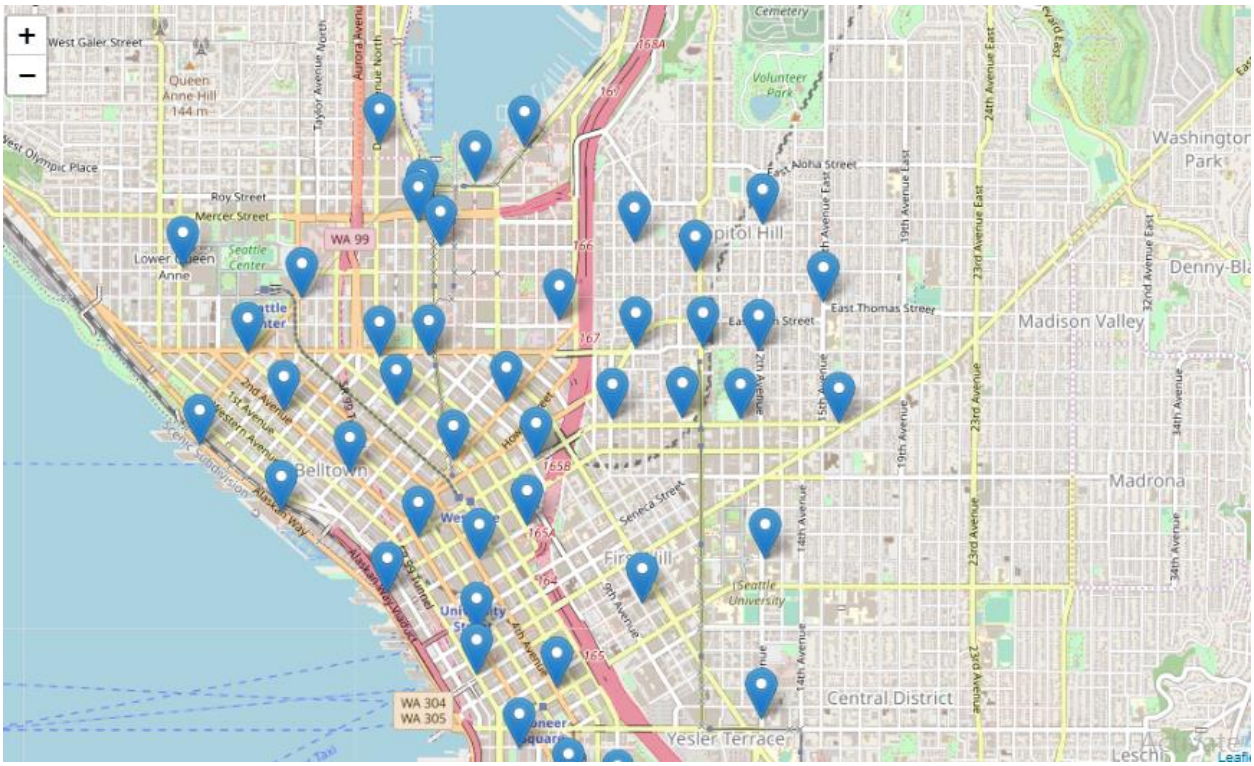
```
34]: trips.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 286858 entries, 0 to 286857
Data columns (total 10 columns):
starttime              286858 non-null object
stoptime               286858 non-null object
tripduration           286858 non-null float64
from_station_name      286858 non-null object
to_station_name        286858 non-null object
from_station_id        286858 non-null object
to_station_id          286858 non-null object
usertype               286858 non-null object
gender                 181558 non-null object
birthyear              181554 non-null float64
dtypes: float64(2), object(8)
memory usage: 21.9+ MB
```

Populating missing values from the gender column

```
In [54]: # Populating missing values from the gender column
         trips.gender.value_counts()

Out[54]: Male      140565
         Female     37562
         Other       3431
         Name: gender, dtype: int64
```

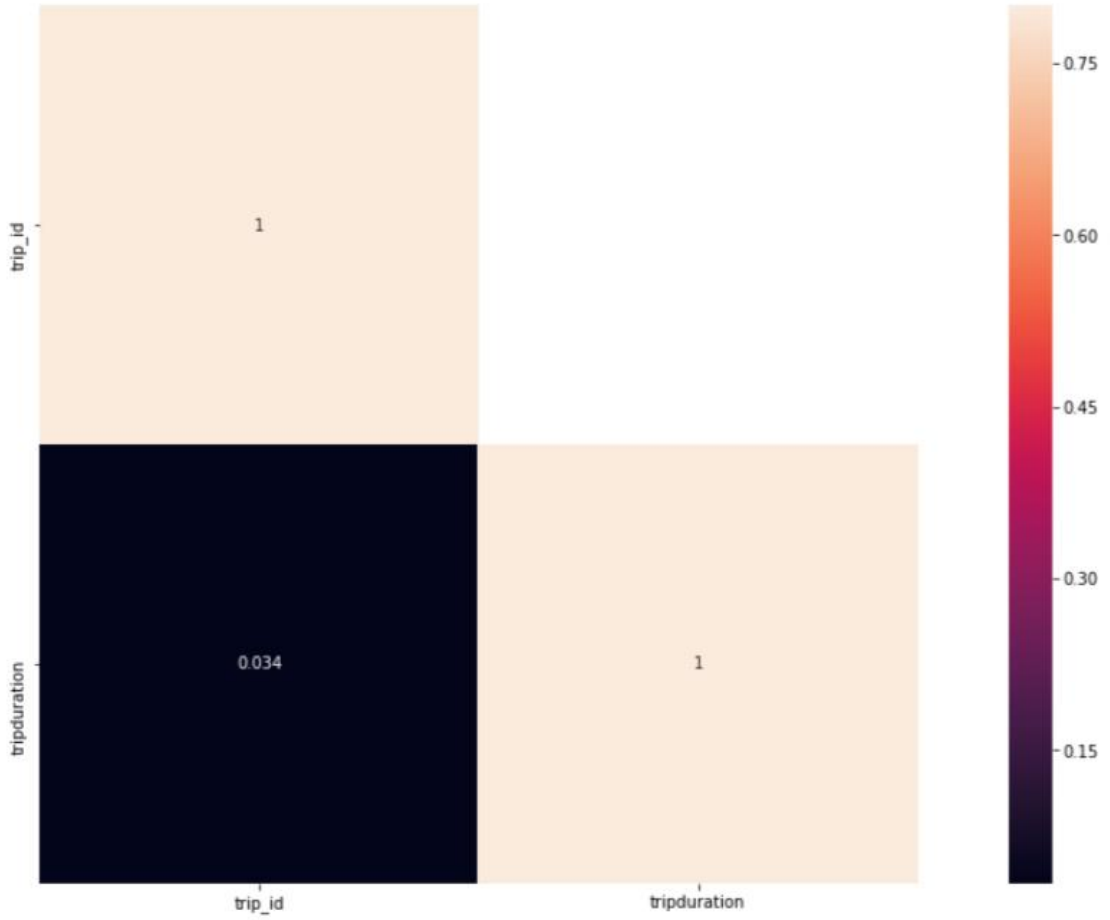
Stations map plotting.



Corrmatt graph of Trip

```
corrMatt = dailyData[["trip_id","starttime","stoptime","bikeid","tripduration","from_station_id","to_station_id"]].corr()
mask = np.array(corrMatt)
mask[np.tril_indices_from(mask)] = False
fig,ax= plt.subplots()
fig.set_size_inches(20,10)
sn.heatmap(corrMatt, mask=mask,vmax=.8, square=True,annot=True)
```

```
: <matplotlib.axes._subplots.AxesSubplot at 0x1659b6e8fd0>
```

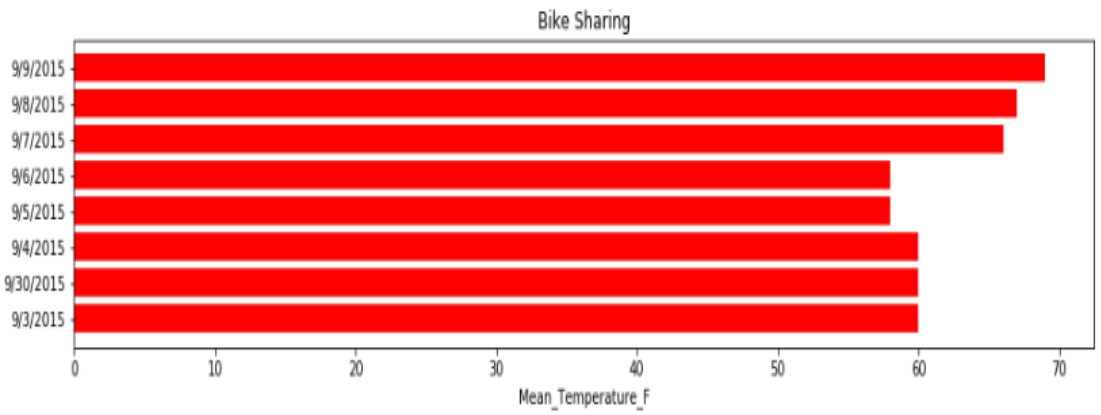


weather temperature

In [6]:
```
# plot Mean_Temperature_F for Date in weather
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from subprocess import check_output
stations = pd.read_csv('D:/Studies/semester 4/project/cycle-share-dataset/station.csv')
trips = pd.read_csv('D:/Studies/semester 4/project/cycle-share-dataset/trip.csv', error_bad_lines=False)

weathers = pd.read_csv('D:/Studies/semester 4/project/cycle-share-dataset/weather.csv')

plt.figure(figsize=(15,3))
pop= weathers.sort_values('Date', ascending=False)
plt.barh(pop['Date'].head(8),pop['Mean_Temperature_F'].head(8), align='center',
        color='red')
plt.gca().invert_yaxis()
plt.xlabel("Mean_Temperature_F")
plt.title("Bike Sharing")
```

```
Skipping line 50794: expected 12 fields, saw 20
```

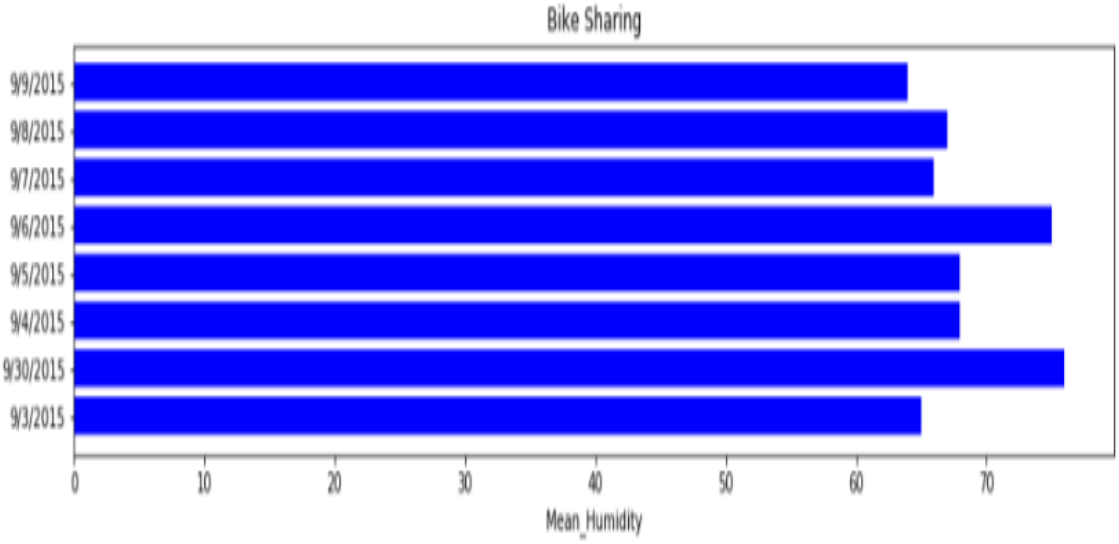
Out[6]: Text(0.5,1,'Bike Sharing')



weather humidity

In [9]:
```
# plot Mean_Humidity for Date in weather
plt.figure(figsize=(15,3))
pop= weathers.sort_values('Date', ascending=False)
plt.barh(pop['Date'].head(8),pop['Mean_Humidity'].head(8), align='center',
        color='Blue')
plt.gca().invert_yaxis()
plt.xlabel("Mean_Humidity")
plt.title("Bike Sharing")
```

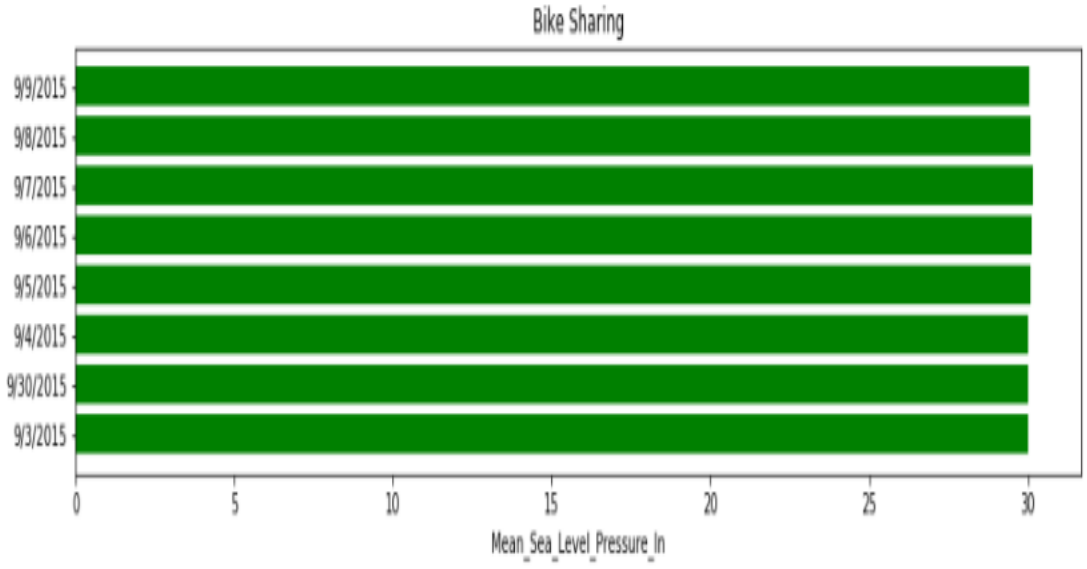
Out[9]: Text(0.5,1,'Bike Sharing')



weather sea level

In [10]:
```
# plot Mean_Sea_Level_Pressure_In for Date in weather
plt.figure(figsize=(15,3))
pop= weathers.sort_values('Date', ascending=False)
plt.barh(pop['Date'].head(8),pop['Mean_Sea_Level_Pressure_In'].head(8), align='center',
        color='green')
plt.gca().invert_yaxis()
plt.xlabel("Mean_Sea_Level_Pressure_In")
plt.title("Bike Sharing")
```

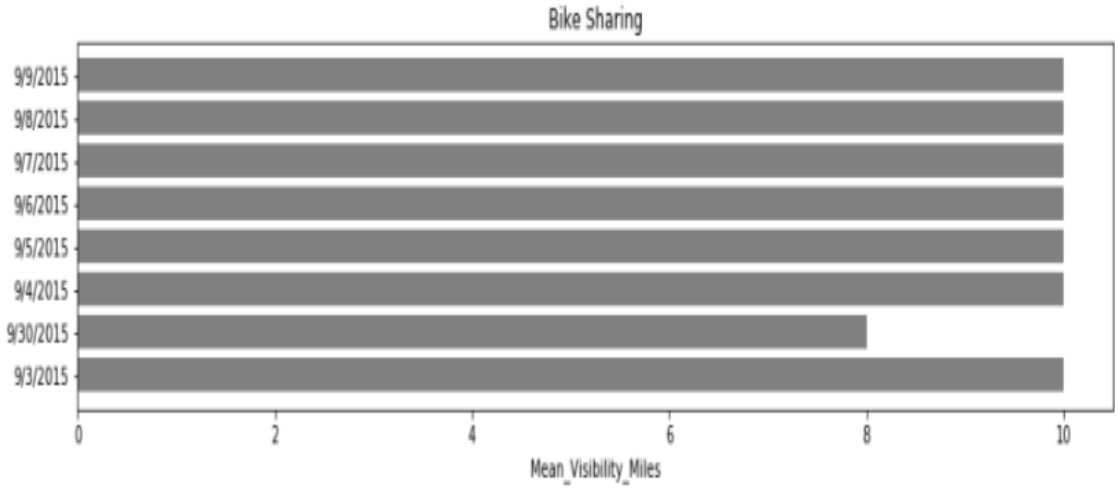
Out[10]: Text(0.5,1,'Bike Sharing')



weather visibility

In [13]:
```
# plot Mean_Visibility_Miles for Date in weather
plt.figure(figsize=(15,3))
pop= weathers.sort_values('Date', ascending=False)
plt.barh(pop['Date'].head(8),pop['Mean_Visibility_Miles'].head(8), align='center',
        color='grey')
plt.gca().invert_yaxis()
plt.xlabel("Mean_Visibility_Miles")
plt.title("Bike Sharing")
```

Out[13]: Text(0.5,1,'Bike Sharing')



Weather speed

In [11]:
```
# plot Mean_Wind_Speed_MPH for Date in weather
plt.figure(figsize=(15,3))
pop= weathers.sort_values('Date', ascending=False)
plt.barh(pop['Date'].head(8),pop['Mean_Wind_Speed_MPH'].head(8), align='center',
        color='yellow')
plt.gca().invert_yaxis()
plt.xlabel("Mean_Wind_Speed_MPH")
plt.title("Bike Sharing")
```

Out[11]: Text(0.5,1,'Bike Sharing')