In [ ]:
```
###IMPORTING DATA

#> Tried importing data using boto3- But encountered no access error-4
    #Installed boto3 and tried importing boto3
    #s3 = boto3.resource('s3')
    #obj = s3.get_object(Bucket='tripdata', Key='tripdata/201701-citib
    #df = pd.read_csv(io.BytesIO(obj['Body'].read()))

#> Then tried extracting data from url using glob.glob-to modify strin
    #adding similar files to a list and appending the list and concate
    #Tried placing glob.glob (string modify) function in for loop, as

#> Then tried modifying url using simple variables and for loop but re

#> Then tried modifying url using simple variables- tospecify the mont
    #as header names had different spacing and cases in multiple files

#> Tried concatenating with axis=1, this adds to columns instead of ro
```

In [19]:
```
import pandas as pd
import glob
import os

li = []

for i in range(1,13):
        j=str(i).zfill(2)
        df1 = pd.read_csv("https://s3.amazonaws.com/tripdata/2017"+j+"
       'Start Station Name', 'Start Station Latitude',
       'Start Station Longitude', 'End Station ID', 'End Station Name'
       'End Station Latitude', 'End Station Longitude', 'Bike ID', 'Us
       'Birth Year', 'Gender'])
        li.append(df1)
        df2 = pd.read_csv("https://s3.amazonaws.com/tripdata/2018"+j+"
       'Start Station Name', 'Start Station Latitude',
       'Start Station Longitude', 'End Station ID', 'End Station Name'
       'End Station Latitude', 'End Station Longitude', 'Bike ID', 'Us
       'Birth Year', 'Gender'])
        li.append(df2)
        if i!=2:
            df3 = pd.read_csv("https://s3.amazonaws.com/tripdata/2019"
           'Start Station Name', 'Start Station Latitude',
           'Start Station Longitude', 'End Station ID', 'End Station N
           'End Station Latitude', 'End Station Longitude', 'Bike ID',
           'Birth Year', 'Gender'])
            li.append(df3)
        if i<11 and i!=6:
            df4 = pd.read_csv("https://s3.amazonaws.com/tripdata/2020"
           'Start Station Name', 'Start Station Latitude',
           'Start Station Longitude', 'End Station ID', 'End Station N
           'End Station Latitude', 'End Station Longitude', 'Bike ID',
           'Birth Year', 'Gender'])
            li.append(df4)
df3 = pd.read_csv("/Users/vyshnavigovindankutty/Desktop/Bike Data/2019
           'Start Station Name', 'Start Station Latitude',
           'Start Station Longitude', 'End Station ID', 'End Station N
           'End Station Latitude', 'End Station Longitude', 'Bike ID',
           'Birth Year', 'Gender'])
li.append(df3)
df4 = pd.read_csv("/Users/vyshnavigovindankutty/Desktop/Bike Data/2020
           'Start Station Name', 'Start Station Latitude',
           'Start Station Longitude', 'End Station ID', 'End Station N
           'End Station Latitude', 'End Station Longitude', 'Bike ID',
           'Birth Year', 'Gender'])
li.append(df4)
```

In [20]:
```
df = pd.concat(li, axis=0, ignore_index=True)
df.head()
```

Out[20]:

	Trip Duration	Start Time	Stop Time	Start Station ID	Start Station Name	Start Station Latitude	Start Station Longitude	End Station ID	End Station Name	L
0	680	2017-01-01 00:00:21	2017-01-01 00:11:41	3226.0	W 82 St & Central Park West	40.782750	-73.971370	3165.0	Central Park West & W 72 St	40.
1	1282	2017-01-01 00:00:45	2017-01-01 00:22:08	3263.0	Cooper Square & E 7 St	40.729236	-73.990868	498.0	Broadway & W 32 St	40.
2	648	2017-01-01 00:00:57	2017-01-01 00:11:46	3143.0	5 Ave & E 78 St	40.776829	-73.963888	3152.0	3 Ave & E 71 St	40.
3	631	2017-01-01 00:01:10	2017-01-01 00:11:42	3143.0	5 Ave & E 78 St	40.776829	-73.963888	3152.0	3 Ave & E 71 St	40.
4	621	2017-01-01 00:01:25	2017-01-01 00:11:47	3143.0	5 Ave & E 78 St	40.776829	-73.963888	3152.0	3 Ave & E 71 St	40.

In [4]:
```
##Function to implement data import

def import_data(df2,month,year):
    j=str(month).zfill(2)
    df1 = pd.read_csv("https://s3.amazonaws.com/tripdata/"+str(year)+j
    'Start Station Name', 'Start Station Latitude',
    'Start Station Longitude', 'End Station ID', 'End Station Name',
    'End Station Latitude', 'End Station Longitude', 'Bike ID', 'User
    'Birth Year', 'Gender'])
    df2=df2.append(df1,ignore_index=True)
    df2.head()
    return df2
```

In [5]:
```
df2=pd.DataFrame()
df2.head()
df2=import_data(df2,8,2019)
df2.head()
```

Out[5]:

	Trip Duration	Start Time	Stop Time	Start Station ID	Start Station Name	Start Station Latitude	Start Station Longitude	End Station ID
0	393	2019-08-01 00:00:01.4680	2019-08-01 00:06:35.3780	531.0	Forsyth St & Broome St	40.718939	-73.992663	408.0
1	627	2019-08-01 00:00:01.9290	2019-08-01 00:10:29.7840	274.0	Lafayette Ave & Fort Greene Pl	40.686919	-73.976682	3409.0
2	1132	2019-08-01 00:00:04.0480	2019-08-01 00:18:56.1650	2000.0	Front St & Washington St	40.702551	-73.989402	3388.0
3	1780	2019-08-01 00:00:04.1630	2019-08-01 00:29:44.7940	479.0	9 Ave & W 45 St	40.760193	-73.991255	473.0
4	1517	2019-08-01 00:00:05.4580	2019-08-01 00:25:23.4550	3312.0	1 Ave & E 94 St	40.781721	-73.945940	3312.0

In [ ]:
```
```

In [ ]:
```
### DATA CLEANING
```

In [5]:
```
##1 Checked for duplicates in data

df[df.duplicated()].head()
```

Out[5]:

	Trip Duration	Start Time	Stop Time	Start Station ID	Start Station Name	Start Station Latitude	Start Station Longitude	End Station ID	End Station Name	End Station Latitude	Sta Longi

In [6]:
```
##2 Checked for null values in different columns

df[['User Type']].isnull().sum()
```

Out[6]:
```
User Type    15909
dtype: int64
```

In [7]:
```
##3 Checked for null values in both Start and Stop station for the sam

df[['Start Station Latitude', 'Start Station Longitude']].isnull().all
```

Out[7]:
```
0
```

In [22]:
```
##4 Converted Start Time column to datetime datatype

df['Start Time']=pd.to_datetime(df['Start Time'])
```

In [ ]:
```

```

In [ ]:
```
###DATA ANALYSIS
```

In [ ]:
```
##1 What does the data mean to you?

Over the past decade, bicycle sharing systems have grown in number and
especially New York. This lets users rent cycles for short trips and r
project we perform exploratory analysis on bike share data to gain som
user characteristics .. etc
```

In [7]:
```
    #>How many bikes are in operation
df['Bike ID'].nunique()
```

Out[7]:
```
28208
```

In [6]:
```
    #>What is the most popular month in terms of number of trips, so t
    #month
df.groupby([df['Start Time'].dt.year,df['Start Time'].dt.month]).size(
```

Out[6]:

Start Time	Start Time	
2017	1	726676
	2	791647
	3	727665
	4	1315404
	5	1523268
	6	1731594
	7	1735599
	8	1816498
	9	1878098
	10	1897592
	11	1330649
	12	889967
2018	1	718994
	2	843114
	3	976672

```
              4             1307543
              5             1824710
              6             1953103
              7             1913625
              8             1977177
              9             1877884
              10            1878657
              11            1260355
              12            1016505
2019          1              967287
              2              943744
              3             1327960
              4             1766094
              5             1924563
              6             2125370
              7             2181064
              8             2344224
              9             2444900
              10            2092573
              11            1478708
              12             955210
2020          1             1240596
              2             1146830
              3             1068457
              4              682762
              5             1487890
              6             1882273
              7             2105808
              8             2329514
              9             2488225
              10            2248869
dtype: int64
```

In [8]:
```
    #> What is the busiest time during a day
df.groupby([df['Start Time'].dt.day_name(),df['Start Time'].dt.hour]).
```

Out[8]:

		Count
Start Time	**Start Time**	
Wednesday	**17**	1135031
Tuesday	**18**	1127566
	17	1121611
Wednesday	**18**	1118218
Thursday	**17**	1095890
	18	1084536
Monday	**17**	1070150
	18	1063589
Friday	**17**	1054908
Wednesday	**8**	1036219

In [9]:
```
    #> Number of unique stations

df['Start Station Name'].nunique()
```

Out[9]: 1333

In [13]:
```
    #> Is Pershing Square consistently busy

df.groupby(['Start Station Name',df['Start Time'].dt.year]).size().to_
```

Out[13]:

Start Station Name	Start Time	Count
Pershing Square North	2017	162716
	2019	156575
	2018	150257
E 17 St & Broadway	2019	121781
8 Ave & W 31 St	2019	119958
Broadway & E 22 St	2019	113138
Broadway & E 14 St	2019	113012
E 17 St & Broadway	2017	112218
W 21 St & 6 Ave	2019	110305
E 17 St & Broadway	2018	108883
Broadway & E 22 St	2017	108590
W 21 St & 6 Ave	2018	107165
	2017	107133
West St & Chambers St	2019	105636
	2017	105610
Broadway & E 22 St	2018	105605
Broadway & W 60 St	2019	103167
Christopher St & Greenwich St	2019	101413
12 Ave & W 40 St	2019	99300
8 Ave & W 31 St	2017	97083

In [51]:
```
    #> Bike ID that has most number of miles and the route

import numpy as np

def haversine(lat1,lon1,lat2,lon2):

    lat1 = np.deg2rad(lat1)
    lon1 = np.deg2rad(lon1)

    lon2 = np.deg2rad(lon2)
```

```
    lon2 = np.deg2rad(lon2)
    lat2 = np.deg2rad(lat2)

    dlat = lat2 - lat1
    dlon = lon2 - lon1
    a = np.sin(dlat/2)**2 + np.cos(lat1) * np.cos(lat2) * np.sin(dlon/
    a = 2 * np.arcsin(np.sqrt(a))
    radius_e = 6371
    return a * radius_e

def km_to_miles(distance):

    return distance*0.621

df2=df[["Bike ID","Start Time","Start Station Latitude","Start Station
df2['Distance']=km_to_miles(haversine(df2['Start Station Latitude'],df
#df2.head(20)
df2[df2['Start Time'].dt.year==2018].groupby(['Bike ID','Start Station

#df2=df.groupby(['Bike ID','Start Station Latitude','Start Station Lon
#df2['Distance']=km_to_miles(haversine(df2['Start Station Latitude'],d
#df2.head(20)
```

```
<ipython-input-51-92ec3abc9307>:25: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
```

Out[51]:

			Distance
Bike ID	**Start Station Name**	**End Station Name**	
24354	**8D OPS 01**	**8D QC Station 01**	335.152219
28337	**Soissons Landing**	**Picnic Point**	24.431068
26035	**Picnic Point**	**Soissons Landing**	21.469727
25113	**Riverside Dr & W 104 St**	**Broadway & Battery Pl**	21.113631
26790	**Soissons Landing**	**Picnic Point**	20.729391
...	**...**	**...**	...
27171	**Riverside Dr & W 72 St**	**Riverside Dr & W 72 St**	0.000000
28311	**West St & Chambers St**	**West St & Chambers St**	0.000000

27171	**Rivington St & Ridge St**	**Rivington St & Ridge St**	0.000000
30106	**5 Ave & E 88 St**	**5 Ave & E 88 St**	0.000000
20313	**Centre St & Chambers St**	**Centre St & Chambers St**	0.000000

17155435 rows × 1 columns

In [32]:
```
    #> Bike ID with maximum number of trips

df2=df.groupby(['Bike ID','Start Station Latitude','Start Station Long
df2.head()
```

Out[32]:

							Count
Bike ID	**Start Station Latitude**	**Start Station Longitude**	**End Station Latitude**	**End Station Longitude**	**Start Station Name**	**End Station Name**	
22368	**40.646538**	**-74.016588**	**40.646538**	**-74.016588**	**Bressler**	**Bressler**	290
24354	**45.506264**	**-73.568906**	**45.506264**	**-73.568906**	**8D Mobile 01**	**8D Mobile 01**	57
15806	**40.686931**	**-74.016966**	**40.686931**	**-74.016966**	**Yankee Ferry Terminal**	**Yankee Ferry Terminal**	56
27393	**40.692317**	**-74.014866**	**40.692317**	**-74.014866**	**Soissons Landing**	**Soissons Landing**	55
33407	**40.692317**	**-74.014866**	**40.692317**	**-74.014866**	**Soissons Landing**	**Soissons Landing**	52

In [64]:
```
    #> Popular station traffic/total traffic (in percentage)

((df[(~(df['Start Station Name'].isnull()))&(df['Start Time'].dt.year=
.sort_values('Count',ascending=False).head(1))/(len(df[(~(df['Start St
```

Out[64]:

	Count
Start Station Name	
Pershing Square North	0.856368

In [103]:
```
    #> Popular station traffic/total traffic (in percentage)

((df[(~(df['Start Station Name'].isnull()))].groupby([df['Start Static
.sort_values('Count',ascending=False).head(1))/(len(df[(~(df['Start St
```

Out[103]:

Start Station Name	Count
Pershing Square North	0.750698

In [ ]:
```

```

In [ ]:
```

```

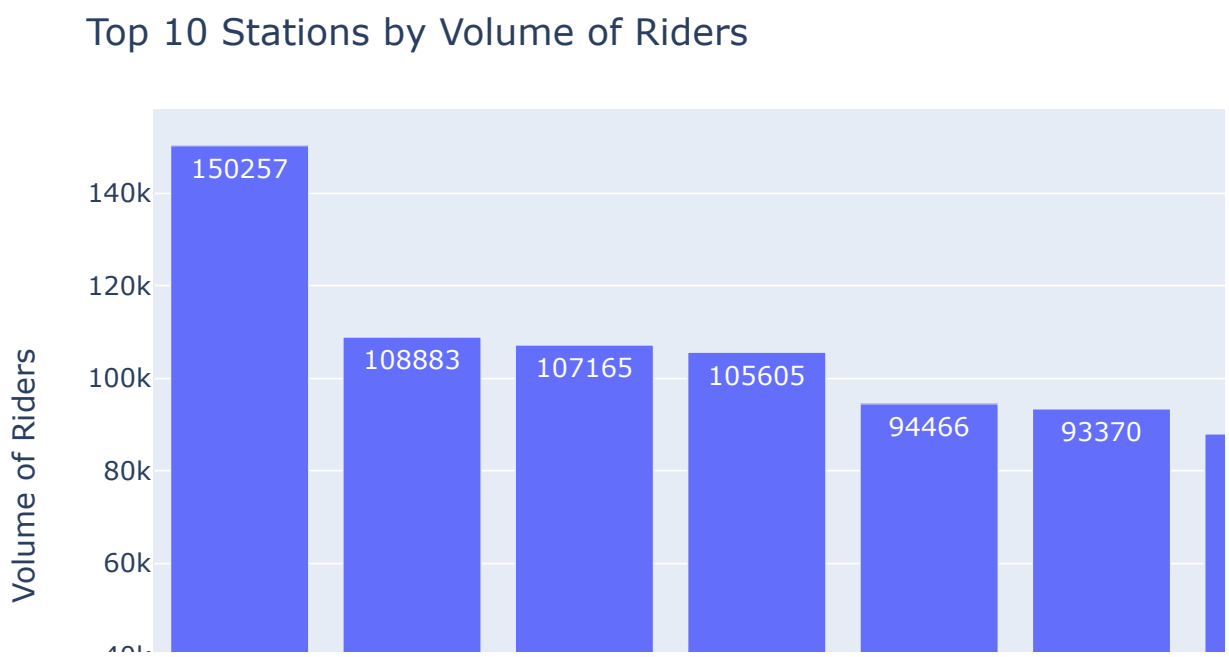
In [ ]:
```
### QUESTIONS
```

In [60]:
```
##1  Which is the most popular station?

    #> First tried grouping by Start Station Name
        #then excluded null values
    #> Tried displaying max count value but station name corresponding
        #df[[~(df['Start Station Name'].isnull())]].groupby(df['Start

    #> Hence tried sorting values based on Count

df[(~(df['Start Station Name'].isnull()))&(df['Start Time'].dt.year==2
```

Out[60]:

Start Station Name	Count
Pershing Square North	150257
E 17 St & Broadway	108883
W 21 St & 6 Ave	107165
Broadway & E 22 St	105605
West St & Chambers St	94466

In [23]:
```
## Visualize top 10 stations by volume of riders

import plotly.express as px
df1=df[df['Start Time'].dt.year==2018].groupby('Start Station Name').s
xx=df1.index.tolist()
yy=df1.Count

#df1.index.count()
fig = px.bar(x=xx,y=yy,text=yy)
fig.update_layout(
    title="Top 10 Stations by Volume of Riders",
    xaxis_title="Station Name",
    yaxis_title="Volume of Riders")
fig.show()
```



In [14]:
```
##2 What is the average distance that bikers ride?
import numpy as np

def haversine(lat1,lon1,lat2,lon2):

    lat1 = np.deg2rad(lat1)
    lon1 = np.deg2rad(lon1)

    lon2 = np.deg2rad(lon2)
    lat2 = np.deg2rad(lat2)

    dlat = lat2 - lat1
    dlon = lon2 - lon1
    a = np.sin(dlat/2)**2 + np.cos(lat1) * np.cos(lat2) * np.sin(dlon/
    a = 2 * np.arcsin(np.sqrt(a))
    radius_e = 6371
    return a * radius_e

def km_to_miles(distance):

    return distance*0.621


df['Distance'] = km_to_miles(haversine(df['Start Station Latitude'], d
df['End Station Latitude'], df['End Station Longitude']))

df['Distance'].mean()  # in miles
```

Out[14]:
```
1.1556939407943971
```

In [10]:
```
##3 What is the average trip duration?

(df['Trip Duration'].mean())/(60)   # in minutes
```

Out[10]:
```
17.834371498116045
```

In [15]:
```
##4 How many riders start and end their trip at the same station?

    #> Tried count function but it gave count for each column
    #> Tried size() but error- cannot call size() directly from datafr
    #> Then tried length of rows that satisfy the condition

len(df[(df['Start Station ID'].notnull())&(df['End Station ID'].notnul
(df['Start Station ID']==df['End Station ID'])])

#2.5% of total rides
```

Out[15]:
```
1741150
```

In [90]:
```
## Percentage of riders that are subscribers who start and stop at the

len(df[(df['Start Station ID'].notnull())&(df['End Station ID'].notnul
(df['Start Station ID']==df['End Station ID'])&(df['User Type']=='Subs

#70% of riders that start and stop at the same station are subscribers
```

Out[90]: 1212382

In [92]:
```
##5 What is the distribution of Customers to Subscribers?

    #> There were rows with Start Station Names that were temporarily
        #such temporarily removed names
        #df[df['Start Station Name'].astype(str).str.contains('tempora
    #> Removed station names containing temporarily removed but divisi
        #were Nan hence replaced them with 0
        #(df[~(df['Start Station Name'].astype(str).str.contains('temp
        #.groupby(['Start Station Name']).size())/(df[~(df['Start Stat
        #in places of NaN

((((df[~(df['Start Station Name'].astype(str).str.contains('temporaril
   .groupby(['Start Station Name']).size())/
     (df[~(df['Start Station Name'].astype(str).str.contains('temporar
      .groupby(['Start Station Name']).size())).replace('NaN',0).head(
```

Out[92]:

Start Station Name	Count
1 Ave & E 39 St	26.419409
1 Ave & E 110 St	16.468117
1 Ave & E 18 St	8.805298
1 Ave & E 30 St	7.036315
1 Ave & E 16 St	7.027486

In [11]:
```
##6 What is the distribution of men and women?

((df[~(df['Start Station Name'].astype(str).str.contains('temporarily
  .groupby(['Start Station Name']).size())/
     (df[~(df['Start Station Name'].astype(str).str.contains('temporar
      .groupby(['Start Station Name']).size())).replace('NaN',0).head(
```

Out[11]:
```
Start Station Name
1 Ave & E 110 St    2.615739
1 Ave & E 16 St     2.610634
1 Ave & E 18 St     2.703881
1 Ave & E 30 St     2.800755
1 Ave & E 39 St     2.050451
dtype: float64
```

In [16]:
```
##7 What days of the week are most rides taken on?

    #>Tried group by dayofweek
    #df.groupby(pd.to_datetime(df['Start Time']).dt.dayofweek).size().
        #.sort_values('Count',ascending=False)
    #> Then tried grouping by day_name()
df.groupby(df['Start Time'].dt.day_name()).size().to_frame('Count').sc
```

Out[16]:

Start Time	Count
Wednesday	10994964
Thursday	10648402
Tuesday	10636868
Friday	10412372
Monday	9933206
Saturday	9722146
Sunday	8797959

In [76]:
```
##8 Get top 10 bikeid by duration for each month in the first quarter

    #> Tried implementing window function directly using Pandas, creat
        #descending order and to choose ranks<=10 and sort by month an
        #thus they cannot be hashed
        #(df.assign(rn=df.groupby(df['Start Time'].dt.year,df['Start T

    #> This just displayed the ranks and not bike id
        #df[df['Start Time'].dt.year==2019].sort_values('rn').head()

    #> Created a column rank  to implement ranking like 'alias' in SQL
        #df['rn']=df.groupby([df['Start Time'].dt.year,df['Start Time'

    #> Tried sorting values based on rank but did not work- error- gro
        #df[(df['Start Time'].dt.year==2019)].groupby([df['Start Time'

df['rn']=df.groupby([df['Start Time'].dt.year,df['Start Time'].dt.mont
df[(df['Start Time'].dt.year==2019)&(df['Start Time'].dt.month<5)&(df[
.groupby([df['Start Time'].dt.month]).head(10)
```

Out[76]:

	Trip Duration	Start Time	Stop Time	Start Station ID	Start Station Name	Start Station Latitude	Start Station Longitude	Sta
1465531	1371164	2019-01-01 19:57:47.094	2019-01-17 16:50:31.5930	456.0	E 53 St & Madison Ave	40.759711	-73.974023	2
1701024	1792506	2019-01-08 16:57:51.880	2019-01-29 10:52:58.4820	3117.0	Franklin St & Dupont St	40.735640	-73.958660	30
1747110	2679841	2019-01-09 17:14:05.549	2019-02-09 17:38:06.5610	3044.0	Albany Ave & Fulton St	40.680011	-73.938475	35
1768433	2377708	2019-01-10 08:22:41.472	2019-02-06 20:51:10.3160	3152.0	3 Ave & E 71 St	40.768737	-73.961199	32
		2019-01-11	2019-01-31		Fulton St &			

In [14]:
```
##9 Get Station Name, latitude, longitude and number of bikes started

    #> Error when groupby year-Grouper and axis must be same length
        #df[pd.DatetimeIndex(df['Start Time']).year==2019].groupby(['S

    #> Then tried including groupby(year) but error grouper and axis m
        #df[pd.DatetimeIndex(df['Start Time']).year==2019].groupby([pd

    #> Would onlydisplay Start Station ID and no other value like sele
        #df[df[pd.DatetimeIndex(df['Start Time']).year==2019]].groupby


df[(df['Start Time'].dt.year==2019)&(df['Start Time'].dt.month<5)]
.groupby(['Start Station ID','Start Station Name','Start Station Latit
df['Start Time'].dt.date]).size().head()
```

Out[14]:
```
Start Station ID  Start Station Name  Start Station Latitude  Start S
tation Longitude  Start Time
72.0              W 52 St & 11 Ave    40.767272               -73.993
929               2019-01-01    46

2019-01-02    43

2019-01-03    61

2019-01-04    74

2019-01-05    26
dtype: int64
```

In [6]:
```
##10 What is the average age of riders by gender?

    #> Tried getting mean by replacing nan values by 0 and by 2020 but
        #had different number of values
        #df[2020-(df['Birth Year'])].groupby('Gender').replace('nan',0
        #KeyError: "None of [Float64Index
    #> Checked if Birth Year had Nan values and tried to exclude those
        #df1=df[~(df['Birth Year'].isnull())]
        #df1[2020-(df1['Birth Year'].apply(pd.to_numeric,errors='coerc
    #> Then tried getting Birth Year and gender data step by step
        #df[['Birth Year','Gender']].head()
        #df[['Birth Year','Gender']].groupby('Gender').head()
        #df[['Birth Year','Gender']].groupby('Gender').mean()


(2020-(df[['Birth Year','Gender']].groupby('Gender').mean())).round(0)
```

Out[6]:

Gender	Birth Year
0	50
1	40
2	38

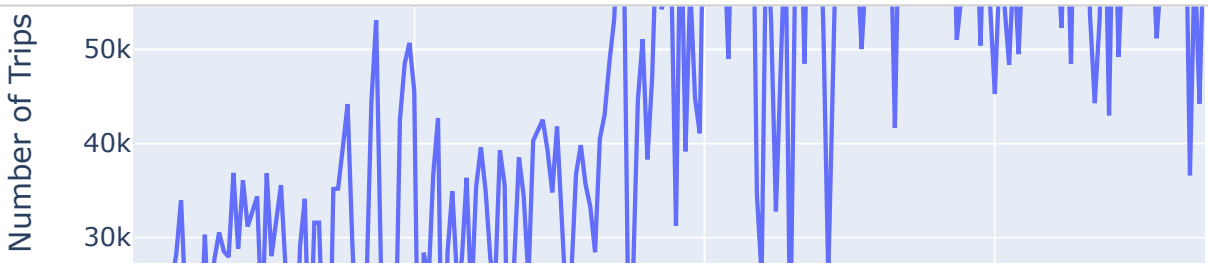
In [ ]:

In [ ]:

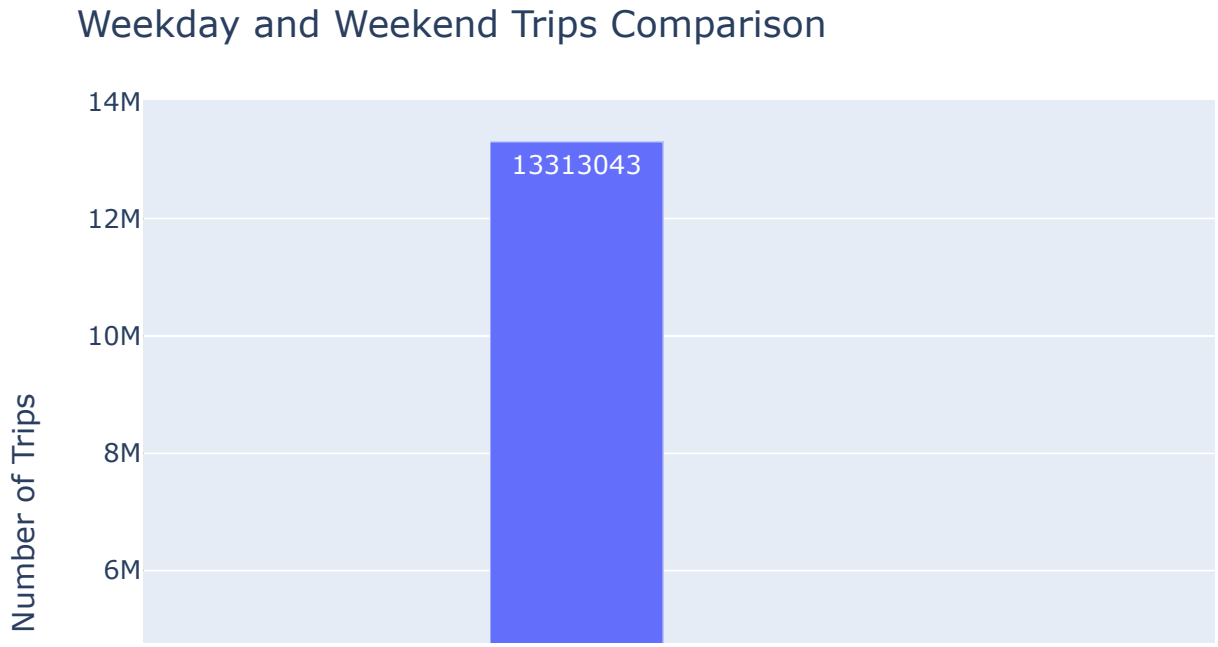
In [ ]:
```
#PLOTLY
```

In [30]:
```
##1 Installed and imported plotly

import plotly.express as px
```

In [24]:
```
##2 Visualize the number of trips per day for 2019, grouped by month

    #> Tried executing the statement in Pandas
    #> Tried displaying only xaxis values and y axis values
    #> Tried groupby to get unique date values but it diaplyed the ent
    #> nunique function to get unique dates but gave only the total nu
    #> Then tried unique().head() to display unique dates but error
    #> Then tried unique() but gave datetime('date') format hence trie
    #> Did not display months April,June so checked if those values we

import plotly.express as px
xx=df[df['Start Time'].dt.year==2018]['Start Time'].dt.date.unique()
yy=df[df['Start Time'].dt.year==2018].groupby([df['Start Time'].dt.mon
fig = px.line(df,x=xx,y=yy)
fig.update_layout(
    title="Number of Trips per Day for 2019",
    xaxis_title="Month of 2019",
    yaxis_title="Number of Trips")
fig.show()
```

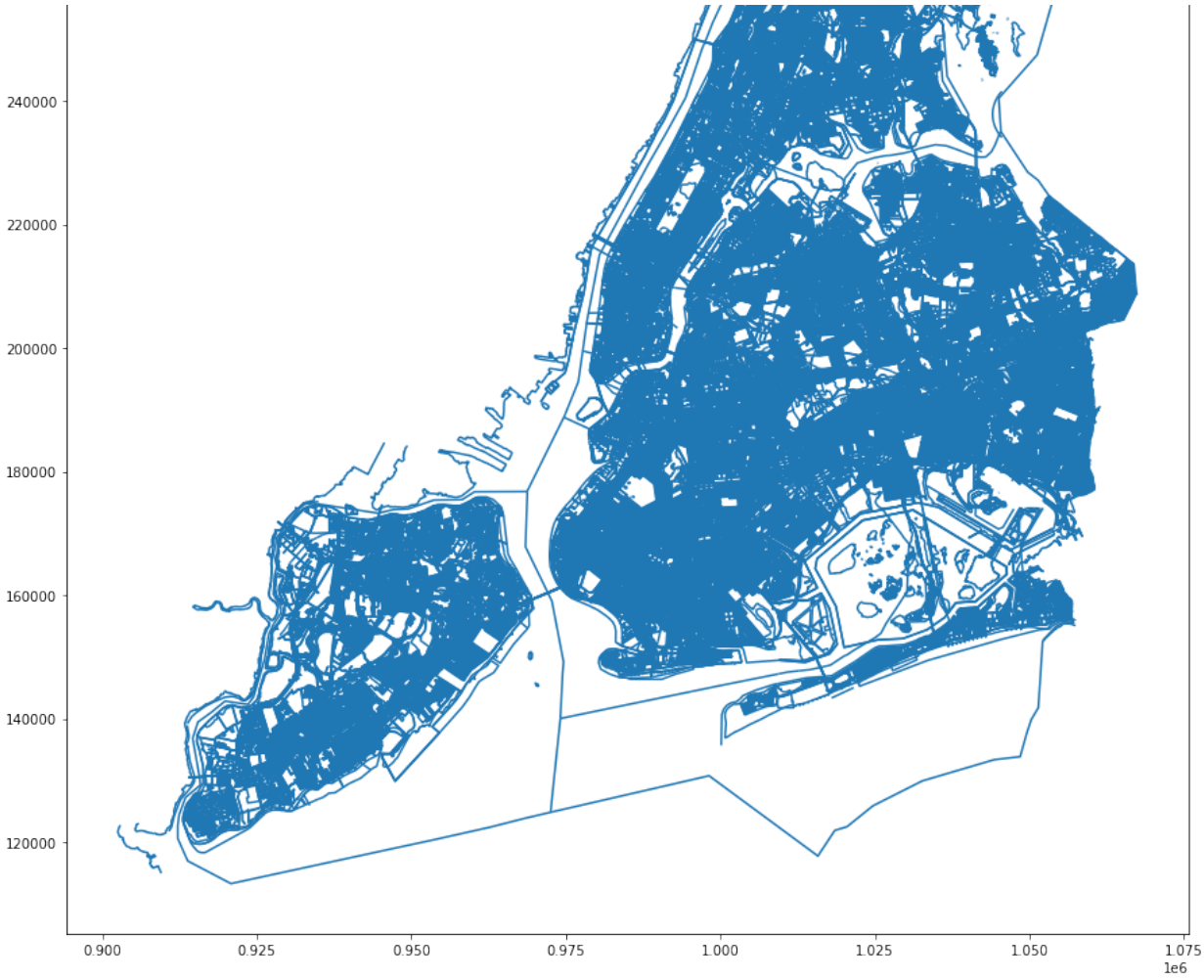


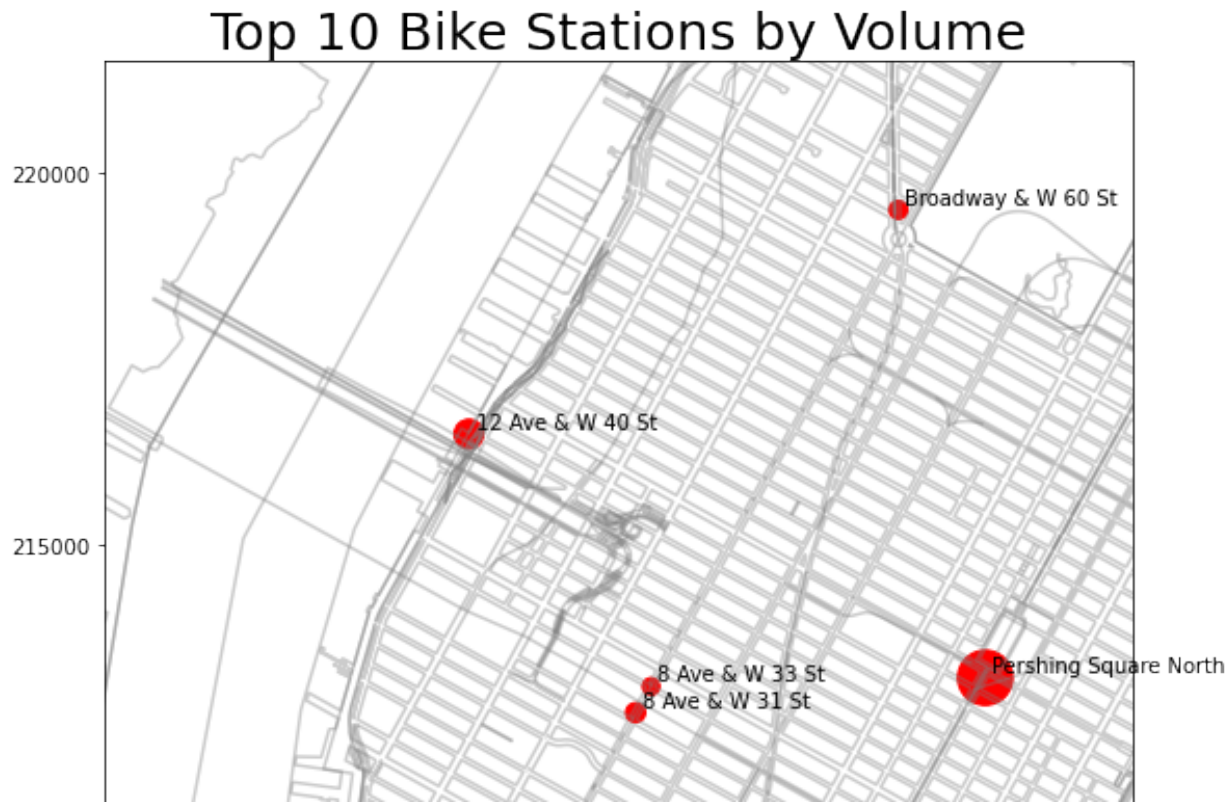
In [25]:
```
##4 Visualize the imbalance between weekday and weekend trips

import plotly.express as px

Weekday=(df[(df['Start Time'].dt.year==2018)&(df['Start Time'].dt.dayo
Weekend=(df[(df['Start Time'].dt.year==2018)&(df['Start Time'].dt.dayo
yy=[Weekday,Weekend]
xx=['Weekdays','Weekend']
fig = px.bar(x=xx,y=yy,text=yy)
fig.update_layout(
    title="Weekday and Weekend Trips Comparison",
    xaxis_title="Day of the Week",
    yaxis_title="Number of Trips",bargap=.8)
fig.show()
```

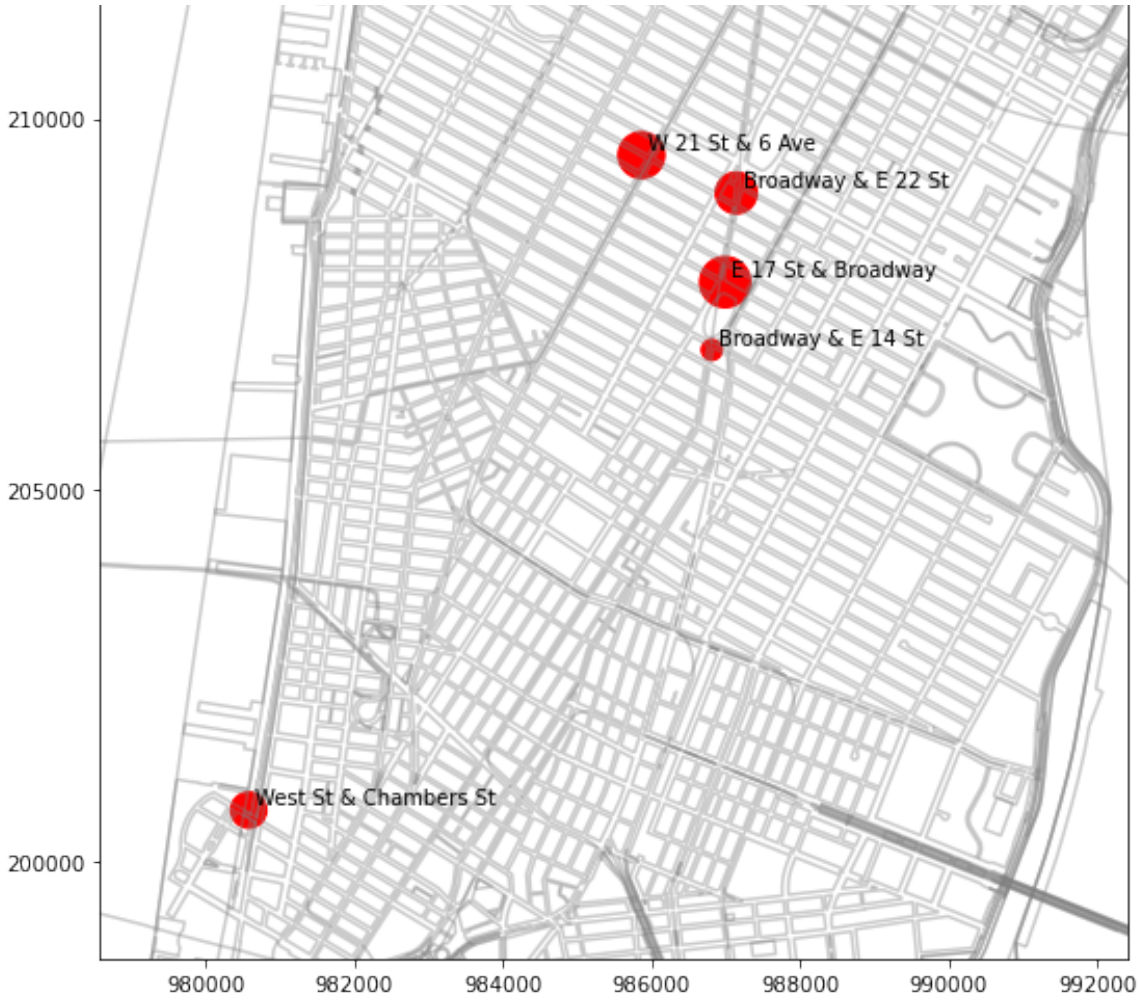


In [122]:
```
##5 Plot the top 10 stations by volume using station latitude and long

import descartes
import geopandas as gpd
import matplotlib.pyplot as plt
from shapely.geometry import Point,Polygon

#Get the required columns onto df3 and the corresponding count sorted
df3=df.groupby(['Start Station Latitude','Start Station Longitude','St

lat=df3.index.get_level_values(0).to_list()
lon=df3.index.get_level_values(1).to_list()

#Read the Neq York street center map shape file and plot it
```

```
street_map=gpd.read_file("/Users/vyshnavigovindankutty/Downloads/dcm_z
#street_map=gpd.read_file("/Users/vyshnavigovindankutty/Downloads/dcm_
fig,ax=plt.subplots(figsize=(15,15))
street_map.plot(ax=ax)

#We have to plot the latitude and longitudes from df3, hence convert t
#using Point() function and zip() in for loop
geometry=[Point(xy) for xy in zip(lon,lat)]
geometry[:11]


#Create a geo data frame using geopandas using df3 dataframe and geome
geo_df=gpd.GeoDataFrame(df3,geometry=geometry)

#Set the axis for the newly created geo_df geo data frame to "crs"
geo_df.set_crs(epsg=4326, inplace=True)
geo_df.to_crs(epsg=3395)

#Convert the axis to match that of the street_map from shape file crea
#outside the map
geo_df=geo_df.to_crs(street_map.crs)
#geo_df.plot(marker='*', color='green', markersize=5);

#Change the count value to adjust the marker size
geo_df['Count']=(geo_df['Count']/900)+40
#geo_df.head()

#Plot street map and geo_df map
fig,ax=plt.subplots(figsize=(15,15))
street_map.plot(ax=ax,alpha=.4,color='grey')

#To zoom in on a specific area having geo_df points get the min and ma
#Provide a padding of around 2000 to mark the points that lie on the a
#Set the axis bounds using ax.set() x_lim and y_lim
minx, miny, maxx, maxy = geo_df.total_bounds
ax.set_xlim(minx-2000, maxx+2000)
ax.set_ylim(miny-2000, maxy+2000)
geo_df.plot(ax=ax,markersize=[700,600,500,400,300,200,100,90,80,70],co
ax.set_title("Top 10 Bike Stations by Volume", fontsize=25)

#Provide labels for the points
for x, y, label in zip(geo_df.geometry.x, geo_df.geometry.y, geo_df.in
    ax.annotate(label, xy=(x, y), xytext=(3, 3), textcoords="offset po
#plt.legend(prop={'size':15})
fig.savefig('newyork.jpg')
```


240000
220000
200000
180000
160000
140000
120000
0.900
0.925
0.950
0.975
1.000
1.025
1.050
1.075
1e6


Top 10 Bike Stations by Volume
220000
215000
Broadway & W 60 St
12 Ave & W 40 St
8 Ave & W 33 St
8 Ave & W 31 St
Pershing Square North



In [116]:
```
geo_df.head()
```

Out[116]:

Start Station Latitude	Start Station Longitude	Start Station Name	Count	geometry
40.751873	**-73.977706**	**Pershing Square North**	633.408889	POINT (990426.889 213205.067)
40.737050	**-73.990093**	**E 17 St & Broadway**	505.497778	POINT (986995.506 207803.886)
40.741740	**-73.994156**	**W 21 St & 6 Ave**	494.056667	POINT (985869.537 209512.445)
40.740343	**-73.989551**	**Broadway & E 22 St**	479.037778	POINT (987145.529 209003.778)
40.717548	**-74.013221**	**West St & Chambers St**	477.766667	POINT (980585.121 200699.005)

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [21]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [34]:

In [ ]: