

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

#> Tried importing data using boto3- But encountered no access error-4
04
    #Installed boto3 and tried importing boto3
    #s3 = boto3.resource('s3')
    #obj = s3.get_object(Bucket='tripdata', Key='tripdata/201701-citibike-tripdata.csv.zip')
    #df = pd.read_csv(io.BytesIO(obj['Body'].read()))

#> Then tried extracting data from url using glob.glob-to modify string to match the right url for each file and
    #adding similar files to a list and appending the list and concatenating the list but resulted in 0 files in list
    #Tried placing glob.glob (string modify) function in for loop, as part of list ..etc

#> Then tried modifying url using simple variables and for loop but resulted in columns being duplicated

#> Then tried modifying url using simple variables- to specify the month in for loop specifying the header names
    #as header names had different spacing and cases in multiple files

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

```
In [70]: 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+"-citibike-tripdata.csv.zip", index_col=None, header=0, names=['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', 'End Station Longitude', 'Bike ID', 'User Type', 'Birth Year', 'Gender'])
    li.append(df1)
    df2 = pd.read_csv("https://s3.amazonaws.com/tripdata/2018"+j+"-citibike-tripdata.csv.zip", index_col=None, header=0, names=['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', 'End Station Longitude', 'Bike ID', 'User Type', 'Birth Year', 'Gender'])
    li.append(df2)
```

```

ation', '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', 'End Station Longitude', 'Bike ID', 'Us
er Type',
      'Birth Year', 'Gender'])
    li.append(df2)
    if i!=2:
        df3 = pd.read_csv("https://s3.amazonaws.com/tripdata/2019"
+j+ "-citibike-tripdata.csv.zip", index_col=None, header=0, names=[ 'Trip
Duration', 'Start Time', 'Stop Time', 'Start Station ID',
      'Start Station Name', 'Start Station Latitude',
      'Start Station Longitude', 'End Station ID', 'End Station N
ame',
      'End Station Latitude', 'End Station Longitude', 'Bike ID',
      'User Type',
      'Birth Year', 'Gender'])
        li.append(df3)
    if i<11 and i!=6:
        df4 = pd.read_csv("https://s3.amazonaws.com/tripdata/2020"
+j+ "-citibike-tripdata.csv.zip", index_col=None, header=0, names=[ 'Trip
Duration', 'Start Time', 'Stop Time', 'Start Station ID',
      'Start Station Name', 'Start Station Latitude',
      'Start Station Longitude', 'End Station ID', 'End Station N
ame',
      'End Station Latitude', 'End Station Longitude', 'Bike ID',
      'User Type',
      'Birth Year', 'Gender'])
        li.append(df4)
df3 = pd.read_csv("/Users/vyshnavigovindankutty/Desktop/Bike Data/2019
02-citibike-tripdata.csv", index_col=None, header=0, names=[ 'Trip Durat
ion', 'Start Time', 'Stop Time', 'Start Station ID',
      'Start Station Name', 'Start Station Latitude',
      'Start Station Longitude', 'End Station ID', 'End Station N
ame',
      'End Station Latitude', 'End Station Longitude', 'Bike ID',
      'User Type',
      'Birth Year', 'Gender'])
li.append(df3)
df4 = pd.read_csv("/Users/vyshnavigovindankutty/Desktop/Bike Data/2020
06-citibike-tripdata.csv", index_col=None, header=0, names=[ 'Trip Durat
ion', 'Start Time', 'Stop Time', 'Start Station ID',
      'Start Station Name', 'Start Station Latitude',
      'Start Station Longitude', 'End Station ID', 'End Station N
ame',
      'End Station Latitude', 'End Station Longitude', 'Bike ID',
      'User Type',
      'Birth Year', 'Gender'])
li.append(df4)

```

```
In [71]: df = pd.concat(li, axis=0, ignore_index=True)
df.head()
```

Out[71]:

	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
+"-citibike-tripdata.csv.zip", index_col=None, header=0,names=['Trip D
uration', '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', 'End Station Longitude', 'Bike ID', 'User
Type',
    '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	End Station Longitude
--	------------------	---------------	--------------	------------------------	--------------------------	------------------------------	-------------------------------	----------------------	------------------------	----------------------------	-----------------------------

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 same row as they are important fields

df[['Start Station Latitude', 'Start Station Longitude']].isnull().all(axis=1).sum()
```

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

```
In [75]: ##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 popularity in different cities, especially New York. This lets users rent cycles for short trips and return them at the destination station. In this project we perform exploratory analysis on bike share data to gain some insights on bike traffic, average trip duration, user types, user characteristics .. etc
```

```
In [48]: ##2
         #>How has subscriber numbers changed from 2017 to 2018
df[(df['User Type']=='Subscriber')&(df['Start Time'].dt.year==2018)].groupby([df['Start Time'].dt.month,'User Type']).size().head()
```

```
Out[48]: Start Time  User Type
2017      Subscriber    14579325
2018      Subscriber    15614825
2019      Subscriber    16782128
2020      Subscriber    11366053
dtype: int64
```

```
In [41]: #>How many bikes are in operation
df['Bike ID'].nunique()
```

```
Out[41]: 28196
```

```
In [42]: #>What is the most popular month in terms of number of trips, so that we can provide some incentives on that
#month
df.groupby([df['Start Time'].dt.year,df['Start Time'].dt.month]).size().head(24)
```

```
Out[42]: 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
dtype: int64
```

```
In [47]: #> What is the busiest time during a day  
df.groupby([df['Start Time'].dt.dayofweek,df['Start Time'].dt.hour]).s  
ize().to_frame('Count').sort_values('Count',ascending=False).head(10)
```

Out[47]:

Count		
Start Time	Start Time	
2	17	1097303
	18	1086364
1	18	1080738
	17	1075547
3	17	1054514
	18	1049078
0	17	1023918
	18	1019251
4	17	1011324
2	8	1007867

In [ ]:

In [ ]: *### QUESTIONS*

```
In [72]: ##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
to it was not getting displayed.
#df[~(df['Start Station Name'].isnull())].groupby(df['Start
Station Name']).size().to_frame('Count').max()

#> Hence tried sorting values based on Count

df[~(df['Start Station Name'].isnull())].groupby(df['Start Station Name']).size().to_frame('Count').sort_values('Count',ascending=False).head(1)
```

Out[72]:

Count
Start Station Name
Pershing Square North 534071

```
In [73]: ##2 What is the average distance that bikers ride?
import numpy as np

def haversine(lat1,lon1,lat2,lon2):
    lon1 = np.deg2rad(lon1)
    lat1 = np.deg2rad(lat1)
    lon2 = np.deg2rad(lon2)
    lat2 = np.deg2rad(lat2)

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

df['Distance'] = (haversine(df['Start Station Latitude'], df['Start Station Longitude'], df['End Station Latitude'], df['End Station Longitude'])*0.621)

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

Out[73]: 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 dataframe, it can only be called after a groupby  
         #> Then tried length of rows that satisfy the condition  
  
         len(df[(df['Start Station ID'].notnull())&(df['End Station ID'].notnull())&(df['Start Station ID']==df['End Station ID'])])
```

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

```
In [66]: ##5 What is the distribution of Customers to Subscribers?

#> There were rows with Start Station Names that were temporarily
removed- checked for Station names containing
#such temporarily removed names
#df[df['Start Station Name'].astype(str).str.contains('tempora
rily removed')]
#> Removed station names containing temporarily removed but divisi
on caused Nan values where column values
#were Nan hence replaced them with 0
#(df[~(df['Start Station Name'].astype(str).str.contains('temp
orarily removed')) & (df['User Type']=='Customer')])
#.groupby(['Start Station Name']).size()/(df[~(df['Start Stat
ion Name'].astype(str).str.contains('temporarily removed'))& (df['User
Type']=='Subscriber')].groupby(['Start Station Name']).size()).replace
('NaN',0).head()
#in places of NaN

(((df[~(df['Start Station Name'].astype(str).str.contains('temporarily
removed')) & (df['User Type']=='Customer')].groupby(['Start Station Na
me']).size())/
(df[~(df['Start Station Name'].astype(str).str.contains('temporar
ily removed'))& (df['User Type']=='Subscriber')].groupby(['Start Stati
on Name']).size())).replace('NaN',0).head()).to_frame('Count').sort_va
lues('Count')
```

Out[66]:

	Count
Start Station Name	
1 Ave & E 30 St	0.069303
1 Ave & E 16 St	0.070881
1 Ave & E 18 St	0.088980
1 Ave & E 110 St	0.154909
1 Ave & E 39 St	0.264194

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

((df[~(df['Start Station Name'].astype(str).str.contains('temporarily
removed')) & (df['Gender']==1)].groupby(['Start Station Name']).size()
)/
  (df[~(df['Start Station Name'].astype(str).str.contains('temporar
ily removed'))& (df['Gender']==2)].groupby(['Start Station Name']).siz
e()))).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 [12]: ##7 What days of the week are most rides taken on?

df.groupby(pd.to_datetime(df['Start Time']).dt.dayofweek).size().reset
_index(name='Count').sort_values('Count',ascending=False)
```

```
Out[12]:
```

	Start Time	Count
2	2	10206249
3	3	9847210
1	1	9735297
4	4	9567056
0	0	9054811
5	5	8969367
6	6	8078228

```
In [76]: ##8 Get top 10 bikeid by duration for each month in the first quarter
of 2019

#> Tried implementing window function directly using Pandas, creat
ing a column rn with trip duration ranked in
#descending order and to choose ranks<=10 and sort by month an
d year but error- Series' objects are mutable,
#thus they cannot be hashed
#(df.assign(rn=df.groupby(df['Start Time'].dt.year,df['Start T
ime'].dt.month)['Trip Duration'].rank(method='first', ascending=False)
).query('rn <= 10').sort_values([df['Start Time'].dt.year,df['Start Ti
me'].dt.month, 'rn']))

#> 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'
].dt.month])['Trip Duration'].rank(ascending=False);

#> Tried sorting values based on rank but did not work- error- gro
upby has no sort_values attribute
#df[(df['Start Time'].dt.year==2019)].groupby([df['Start Time'
].dt.month])['Start Time', 'rn'].sort_values(['rn']).head()

df['rn']=df.groupby([df['Start Time'].dt.year,df['Start Time'].dt.mont
h])['Trip Duration'].rank(ascending=False);
df[(df['Start Time'].dt.year==2019)&(df['Start Time'].dt.month<5)&(df[
'rn']<11)].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	St
<b>1465531</b>	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
<b>1701024</b>	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
<b>1747110</b>	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	30
<b>1768433</b>	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	30

<b>1832775</b>	1702688	2019-01-11 22:06:10.539	2019-01-31 15:04:19.3930	3042.0	Fulton St & Utica Ave	40.679427	-73.929891	30
<b>1999371</b>	1059371	2019-01-17 07:38:37.582	2019-01-29 13:54:48.8950	229.0	Great Jones St	40.727434	-73.993790	30
<b>2041870</b>	1427844	2019-01-18 09:13:14.402	2019-02-03 21:50:38.6820	468.0	Broadway & W 56 St	40.765265	-73.981923	30
<b>2060370</b>	1058966	2019-01-18 17:53:20.263	2019-01-31 00:02:47.1240	340.0	Madison St & Clinton St	40.712690	-73.987763	30
<b>2061005</b>	1034663	2019-01-18 18:04:02.954	2019-01-30 17:28:26.3530	465.0	Broadway & W 41 St	40.755136	-73.986580	30
<b>2268207</b>	1047413	2019-01-26 18:50:05.617	2019-02-07 21:46:59.3990	472.0	E 32 St & Park Ave	40.745712	-73.981948	30
<b>8228062</b>	2571434	2019-03-04 17:23:10.667	2019-04-03 12:40:24.9710	2008.0	Little West St & 1 Pl	40.705693	-74.016777	30
<b>8301022</b>	1461968	2019-03-06 17:54:46.547	2019-03-23 17:00:54.8470	3064.0	Myrtle Ave & Lewis Ave	40.696820	-73.937569	30
<b>8421451</b>	2754673	2019-03-09 20:02:11.709	2019-04-10 18:13:25.5980	334.0	W 20 St & 7 Ave	40.742388	-73.997262	30
<b>8450905</b>	2969781	2019-03-11 08:28:14.341	2019-04-14 17:24:35.5970	3358.0	Garfield Pl & 8 Ave	40.671198	-73.974841	30
<b>8883338</b>	1339052	2019-03-19 20:47:04.806	2019-04-04 08:44:37.6620	3521.0	Lenox Ave & W 111 St	40.798786	-73.952300	30
<b>8942071</b>	1908974	2019-03-20 23:21:50.400	2019-04-12 01:38:04.6990	3521.0	Lenox Ave & W 111 St	40.798786	-73.952300	30
<b>8952338</b>	1630514	2019-03-21 12:21:26.469	2019-04-09 09:16:41.2680	504.0	1 Ave & E 16 St	40.732219	-73.981656	30
<b>8986679</b>	1797632	2019-03-22 17:46:16.709	2019-04-12 13:06:49.6490	3457.0	E 58 St & Madison Ave	40.763026	-73.972095	30
<b>9207436</b>	1510516	2019-03-27 09:10:00.859	2019-04-13 20:45:17.5790	3427.0	Lafayette St & Jersey St	40.724305	-73.996010	30
<b>9312730</b>	1334968	2019-03-29 07:30:19.837	2019-04-13 18:19:48.5680	361.0	Allen St & Hester St	40.716059	-73.991908	30

<b>13200795</b>	2123501	2019-04-01 18:58:27.387	2019-04-26 08:50:08.6210	3016.0	Kent Ave & N 7 St	40.720368	-73.961651	30
<b>13485990</b>	2210301	2019-04-06 17:52:14.930	2019-05-02 07:50:36.5900	3584.0	Eastern Pkwy & Franklin Ave	40.670777	-73.957680	30
<b>13486136</b>	2245318	2019-04-06 17:53:26.095	2019-05-02 17:35:24.3130	3584.0	Eastern Pkwy & Franklin Ave	40.670777	-73.957680	30
<b>13493510</b>	2572595	2019-04-06 19:16:14.428	2019-05-06 13:52:50.3740	3364.0	Carroll St & 5 Ave	40.675162	-73.981483	30
<b>13543432</b>	2264557	2019-04-07 15:48:01.604	2019-05-03 20:50:38.9830	3042.0	Fulton St & Utica Ave	40.679427	-73.929891	30
<b>13549066</b>	1891712	2019-04-07 16:44:46.529	2019-04-29 14:13:18.8950	3467.0	W Broadway & Spring Street	40.724947	-74.001659	30
<b>13558480</b>	2330556	2019-04-07 18:39:27.501	2019-05-04 18:02:03.7510	3714.0	Division Av & Hooper St	40.706842	-73.954435	30
<b>13607798</b>	1974452	2019-04-08 16:59:10.273	2019-05-01 13:26:42.6120	3498.0	Pleasant Ave & E 120 St	40.797477	-73.931185	30
<b>13733204</b>	1964853	2019-04-10 14:59:25.995	2019-05-03 08:46:59.5460	3541.0	Amsterdam Ave & W 125 St	40.813358	-73.956461	30
<b>13752516</b>	1952891	2019-04-10 18:20:48.647	2019-05-03 08:48:59.9510	533.0	Broadway & W 38 St	40.752996	-73.987216	30
<b>68330251</b>	1947736	2019-02-01 12:51:51.503	2019-02-24 01:54:07.5690	458.0	11 Ave & W 27 St	40.751396	-74.005226	30
<b>68417577</b>	2403037	2019-02-04 13:25:32.159	2019-03-04 08:56:09.9670	224.0	Spruce St & Nassau St	40.711464	-74.005524	30
<b>68434115</b>	1372092	2019-02-04 18:17:00.153	2019-02-20 15:25:12.4170	3691.0	28 Ave & 44 St	40.764089	-73.910651	30
<b>68470677</b>	1982217	2019-02-05 14:31:32.226	2019-02-28 13:08:29.8770	3518.0	Lenox Ave & W 126 St	40.808442	-73.945209	30
<b>68633473</b>	1449369	2019-02-09 16:06:20.648	2019-02-26 10:42:30.2720	3505.0	Lexington Ave & E 127 St	40.805726	-73.936322	30

<b>68664706</b>	1266532	2019-02-10 21:12:30.990	2019-02-25 13:01:23.9350	3629.0	Adam Clayton Powell Blvd & W 126 St	40.809495	-73.947765	36
<b>68679026</b>	1303733	2019-02-11 10:43:05.169	2019-02-26 12:51:58.9290	498.0	Broadway & W 32 St	40.748549	-73.988084	36
<b>68681280</b>	1383937	2019-02-11 12:22:09.131	2019-02-27 12:47:46.7190	364.0	Lafayette Ave & Classon Ave	40.689004	-73.960239	36
<b>68770030</b>	1330980	2019-02-14 11:24:40.396	2019-03-01 21:07:40.8470	267.0	Broadway & W 36 St	40.750977	-73.987654	36
<b>68836095</b>	1304306	2019-02-15 18:53:52.400	2019-03-02 21:12:19.3810	3046.0	Marcus Garvey Blvd & Macon St	40.682601	-73.938037	36

```
In [14]: ##9 Get Station Name, latitude, longitude and number of bikes started
from each station using startstationid* for each day for the first qua
rter of 2019

#> Error when groupby year-Grouper and axis must be same length
#df[pd.DatetimeIndex(df['Start Time']).year==2019].groupby(['S
tart Station ID',pd.DatetimeIndex(df['Start Time']).date]).size().rese
t_index(name='Count')[['Start Station Name','Start Station Latitude','
Start Station Longitude','Start Station ID','Count']]

#> Then tried including groupby(year) but error grouper and axis m
ust be same length
#df[pd.DatetimeIndex(df['Start Time']).year==2019].groupby([pd
.DatetimeIndex(df['Start Time']).year,pd.DatetimeIndex(df['Start Time'
]).date]).agg('count').head()

#> Would onlydisplay Start Station ID and no other value like sele
ct multiple columns in MySQL
#df[df[pd.DatetimeIndex(df['Start Time']).year==2019]].groupby
([pd.DatetimeIndex(df['Start Time']).year,pd.DatetimeIndex(df['Start T
ime']).month]).size()

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

```
Out[14]: Start Station ID  Start Station Name  Start Station Latitude  Start
Station Longitude  Start Time
72.0                W 52 St & 11 Ave      40.767272                -73.99
3929                2019-01-01      46

2019-01-02      43

2019-01-03      61

2019-01-04      74

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



```
In [15]: ##10 What is the average age of riders by gender?

#> Tried getting mean by replacing nan values by 0 and by 2020 but
got key error as columns
#had different number of values
#df[2020-(df['Birth Year'])].groupby('Gender').replace('nan',0)
).mean()
#KeyError: "None of [Float64Index
#> Checked if Birth Year had Nan values and tried to exclude those
but still got the same error
#df1=df[~(df['Birth Year'].isnull())]
#df1[2020-(df1['Birth Year'].apply(pd.to_numeric,errors='coerce'))].groupby('Gender').head()
#> 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())
```

Out[15]:

	Birth Year
Gender	
0	49.797344
1	39.793961
2	38.282270

In [ ]:

In [ ]: #PLOTLY

In [ ]: ##1 Installed and imported plotly

```
import plotly.express as px
```

```
In [19]: ##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 entire table
#> nunique function to get unique dates but gave only the total number of unique dates, in this case 306
#> Then tried unique().head() to display unique dates but error
#> Then tried unique() but gave datetime('date') format hence tried unique().tolist()
#> Did not display months April, June so checked if those values were present

import plotly.express as px
xx=df[df['Start Time'].dt.year==2019]['Start Time'].dt.date.unique()
yy=df[df['Start Time'].dt.year==2019].groupby([df['Start Time'].dt.month,df['Start Time'].dt.date]).size()
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 [59]: ##3 Visualize top 10 stations by volume of riders

import plotly.express as px
df1=df.groupby('Start Station Name').size().to_frame('Count').sort_val
ues('Count',ascending=False).head(10)
xx=df1.index.tolist()
yy=df1.Count

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

```
In [81]: ##4 Visualize the imbalance between weekday and weekend trips

Weekday=(df[df['Start Time'].dt.dayofweek<5].groupby([df['Start Time']
.dt.dayofweek]).size()).sum()
Weekend=(df[df['Start Time'].dt.dayofweek>4].groupby([df['Start Time']
.dt.dayofweek]).size()).sum()
yy=[Weekday,Weekend]
xx=['Weekdays','Weekend']
fig = px.bar(x=xx,y=yy)
fig.update_layout(
    title="Weekday and Weekend Trips Comparison",
    xaxis_title="Day of the Week",
    yaxis_title="Number of Trips")
fig.show()
```

In [ ]:

In [ ]:

In [ ]: