MINI – PROJECT 2

IST – 652

SCRIPTING FOR DATA ANALYSIS

FALL 2022

SEMI - STRUCTURED DATA

PREPARED BY : **HENDI KUSHTA**

Table of Contents

Data and Source....3
Data Exploration and Cleaning....3
Data Analysis - Questions....3
Scripting....3
Output Files....13

Data and Source

The main outline of the assignment is to write a program that reads data found in formats such as JSON (or another semi-structured data format), from a Mongo DB collection, from a file, or from the web. I have chosen 2 datasets both of them are JSON files and are about New York City bike share used for public transportation. The data is found in company's website https://ride.citibikenyc.com/system-data. The data that I have used can be found in the attached links: https://gbfs.citibikenyc.com/gbfs/en/station_information.json and https://gbfs.citibikenyc.com/gbfs/en/station_status.json .

The first dataset has 1795 entries and 17 columns while the second dataset has 1795 entries and 21 columns.

Data Exploration and Cleaning

Firstly I used head() function to check first rows.
Next step was to check the number of rows and columns.
After that I checked the dataset column names in case I needed to change any label.
I have checked datasets info, if there is any null values and if there was, I have droped.
I have merged two different dataset with one another to create a new dataset with more information about the stations. Changed variables name when needed to make a better analysis.
I changed the label listed_in to genre which seems to be better.
There are some other data cleaning and exploration in 3 questions and analysis I have prepared in this mini project.

Data Analysis - Questions

There are three questions I have tried to answer in this first mini-project.
1. Find which are top 10 mos active stations.
2. Which stations are out_of service? Finding the number of bikes that are found in these stations and comparing with the number of bikes in active stations
3. Compare number of bikes available with number of electronic bikes available in top stations

Scripting

Import the necessary packages that I have used in my mini project.
First assigned the json url to a variable. Then got a response from the url and read the response.
Printed first 500 characters from the response.
Load the json file as a dictionary data type, and checked for the keys of this json file.
Create a list of the stations through checking keys inside the keys.

```
# import libraries
import urllib.request
import json
```

```
# first json url
station_information_URL = "https://gbfs.citibikenyc.com/gbfs/en/station_information.json"
```

```
# get response
response = urllib.request.urlopen(station_information_URL)
type(response)
```

```
http.client.HTTPResponse
```

```
# read response
json_string = response.read().decode('utf-8')
json_string[:500]
```

```
'{"data":{"stations":[{"has_kiosk":true,"rental_uris":{"ios":"https://bkn.lft.to/lastmile_qr_scan","androi
d":"https://bkn.lft.to/lastmile_qr_scan"},"lat":40.76727216,"short_name":"6926.01","region_id":"71","name":"W
52 St & 11 Ave","external_id":"66db237e-0aca-11e7-82f6-3863bb44ef7c","legacy_id":"72","station_id":"72","capac
ity":55,"lon":-73.99392888,"station_type":"classic","eightd_station_services":[],"electric_bike_surcharge_waiv
er":false,"rental_methods":["CREDITCARD","KEY"],"eightd_has_key_d'
```

```
# load json as dictionary
eq_parsed_json = json.loads(json_string)
type(eq_parsed_json)
```

```
dict
```

```
# check json file dict keys
eq_parsed_json.keys()
```

```
dict_keys(['data', 'last_updated', 'ttl'])
```

```
# check all information from data key
eq_parsed_json['data']['stations']
```

Below is a better representation of an element in the list created before.

```
# pretty print the dict bike1
print(json.dumps(bike1, indent=4))
```

```
{
    "has_kiosk": true,
    "rental_uris": {
        "ios": "https://bkn.lft.to/lastmile_qr_scan",
        "android": "https://bkn.lft.to/lastmile_qr_scan"
    },
    "lat": 40.76727216,
    "short_name": "6926.01",
    "region_id": "71",
    "name": "W 52 St & 11 Ave",
    "external_id": "66db237e-0aca-11e7-82f6-3863bb44ef7c",
    "legacy_id": "72",
    "station_id": "72",
    "capacity": 55,
    "lon": -73.99392888,
    "station_type": "classic",
    "eightd_station_services": [],
    "electric_bike_surcharge_waiver": false,
    "rental_methods": [
        "CREDITCARD",
        "KEY"
    ],
    "eightd_has_key_dispenser": false
}
```

The next step is to convert semi – structured data to structured data. Imported pandas library.
Converted semi-structured json file to structured.
I have checked the information about the dataset information

```
# import library
import pandas as pd
```

```
# convert semi-structured data to structured
df = pd.json_normalize(json.loads(json.dumps(bikel)))
df
```

15]:

	has_kiosk	lat	short_name	region_id	name	external_id	legacy_id	station_id	capacity	lon	station_type	eightd_station_services	e
0	True	40.767272	6926.01	71	W 52 St & 11 Ave	66db237e-0aca-11e7-82f6-3863bb44ef7c	72	72	55	-73.993929	classic	[]	

```
# Do the same for all dataset
df = pd.json_normalize(json.loads(json.dumps(bikelist)))
```

```
# print first records of the dataset
df_stations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1795 entries, 0 to 1794
Data columns (total 21 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   station_id                      1795 non-null   object
 1   num_bikes_disabled              1795 non-null   int64
 2   is_returning                    1795 non-null   int64
 3   last_reported                   1795 non-null   int64
 4   station_status                  1795 non-null   object
 5   num_docks_disabled              1795 non-null   int64
 6   eightd_has_available_keys       1795 non-null   bool
 7   num_bikes_available             1795 non-null   int64
 8   is_installed                    1795 non-null   int64
 9   is_renting                      1795 non-null   int64
 10  num_ebikes_available            1795 non-null   int64
 11  num_docks_available             1795 non-null   int64
 12  legacy_id                       1795 non-null   object
 13  eightd_active_station_services  5 non-null      object
 14  valet.valet_revision            5 non-null      float64
 15  valet.active                    5 non-null      object
 16  valet.station_id                5 non-null      object
 17  valet.off_dock_capacity         5 non-null      float64
 18  valet.dock_blocked_count        5 non-null      float64
 19  valet.off_dock_count            5 non-null      float64
 20  valet.region                    5 non-null      object
dtypes: bool(1), float64(4), int64(9), object(7)
memory usage: 282.3+ KB
```

check the datatype of one of the variables of the new dataset. Then I have selected only the necessary columns for my analysis.
Secondly, I have followed the same steps for my second dataset to convert to structured data.

Read second json file dataset

```
# first json url
# get response
# read response
status_URL = "https://gbfs.citibikenyc.com/gbfs/en/station_status.json"
response1 = urllib.request.urlopen(status_URL)
json_string = response1.read().decode('utf-8')
json_string[:500]
```

21]:
```
'{"data":{"stations":[{"station_id":"72","num_bikes_disabled":0,"is_returning":1,"last_reported":1669751360,"st
ation_status":"active","num_docks_disabled":0,"eightd_has_available_keys":false,"num_bikes_available":3,"is_ins
talled":1,"is_renting":1,"num_ebikes_available":2,"num_docks_available":52,"legacy_id":"72"},{"station_id":"7
9","num_bikes_disabled":5,"is_returning":1,"last_reported":1669751898,"station_status":"active","num_docks_disa
bled":0,"eightd_has_available_keys":false,"num_bikes_availa'
```

```
# load json as dictionary
eq_parsed_json = json.loads(json_string)
```

```
# check json file dict keys
eq_parsed_json.keys()
```

23]:
```
dict_keys(['data', 'last_updated', 'ttl'])
```

```
# check all information from data key
stationlist = eq_parsed_json['data']['stations']
```

```
# convert semi-structured data to structured
df_stations = pd.json_normalize(json.loads(json.dumps(stationlist)))
```

```
df_stations
```

26]:

	station_id	num_bikes_disabled	is_returning	last_reported	station_status	num_docks_disabled	eightd_has_available_keys	num_bikes_available	is
0	72	0	1	1669751360	active	0	False	3	
1	79	5	1	1669751898	active	0	False	24	
2	82	1	1	1669751164	active	0	False	23	
3	83	0	1	1669751579	active	0	False	45	
4	116	0	1	1669751990	active	0	False	72	
...	...	...	...	...	...	...	...	...	...
1790	4986	5	0	1669745979	active	0	False	14	
1791	5001	1	1	1669751074	active	0	False	2	

I have selected the main variables from the second dataset and then merged the 2 dataset with each other based on station_id.

```
# merge datasets with one another
df_merged = pd.merge(df_bike_station_information, df_bike_station_status, on='station_id')
```

```
df_merged.head()
```

]:

	capacity	lat	lon	has_kiosk	name	station_id	station_status	num_docks_available	num_bikes_disabled	num_docks_disabled	num
0	55	40.767272	-73.993929	True	W 52 St & 11 Ave	72	active	52	0	0	
1	33	40.719116	-74.006667	True	Franklin St & W Broadway	79	active	4	5	0	
2	27	40.711174	-74.000165	True	St James Pl & Pearl St	82	active	3	1	0	
3	62	40.683526	-73.976323	True	Atlantic Ave & Fort Greene Pl	83	active	16	0	0	
4	74	40.741776	-74.001497	True	W 17 St & 8 Ave	116	active	1	0	0	

I have shown again the information about the new merged dataset, checked if there is any null values, and if there was, I have removed them.

```
# get information related with the dataset after merging
# columns, how many rows and columns, non null values, data type of each column
df_merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1795 entries, 0 to 1794
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   capacity              1795 non-null   int64
 1   lat                   1795 non-null   float64
 2   lon                   1795 non-null   float64
 3   has_kiosk             1795 non-null   bool
 4   name                  1795 non-null   object
 5   station_id            1795 non-null   object
 6   station_status        1795 non-null   object
 7   num_docks_available   1795 non-null   int64
 8   num_bikes_disabled    1795 non-null   int64
 9   num_docks_disabled    1795 non-null   int64
 10  num_bikes_available   1795 non-null   int64
 11  num_ebikes_available  1795 non-null   int64
 12  is_renting            1795 non-null   int64
dtypes: bool(1), float64(2), int64(7), object(3)
memory usage: 184.1+ KB
```

```
# check for null values
df_merged.isnull().sum()
```

```
capacity                0
lat                     0
lon                     0
has_kiosk               0
name                    0
station_id              0
station_status          0
num_docks_available     0
num_bikes_disabled      0
num_docks_disabled      0
num_bikes_available     0
num_ebikes_available    0
is_renting              0
dtype: int64
```

```
# drop null values if any
df_merged = df_merged.dropna()
```

After that, I showed some statistics about numeric variables.

```
# get statistical analysis for numerical variables
df_merged.describe().round(2)
```

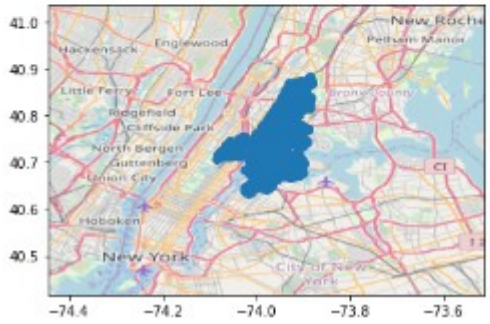
	capacity	lat	lon	num_docks_available	num_bikes_disabled	num_docks_disabled	num_bikes_available	num_ebikes_available	is_renting
count	1795.00	1795.00	1795.00	1795.00	1795.00	1795.00	1795.00	1795.00	1795.00
mean	30.13	40.73	-73.91	15.32	0.79	0.23	13.55	2.45	0.96
std	16.22	0.96	1.75	12.75	1.95	2.95	15.52	4.65	0.21
min	0.00	0.00	-74.09	0.00	0.00	0.00	0.00	0.00	0.00
25%	20.00	40.70	-73.99	5.00	0.00	0.00	2.00	0.00	1.00
50%	25.00	40.74	-73.95	15.00	0.00	0.00	5.00	1.00	1.00
75%	35.00	40.79	-73.92	21.00	1.00	0.00	19.00	3.00	1.00
max	117.00	40.85	0.00	113.00	40.00	66.00	103.00	50.00	1.00

Before going on with 3 questions, I have printed a map of the bikes in New York City.

```
# plot the distribution of city bikes in New York's city map
import matplotlib.pyplot as plt

# The boundaries of the image map
map_box = [-74.4461, -73.5123, 40.4166, 41.0359]
# The name of the image of the New York map might be different.
map_img = plt.imread('map.png')
fig, ax = plt.subplots()
ax.scatter(df_merged['lon'], df_merged['lat'])

ax.imshow(map_img, extent=map_box, alpha=0.9)
plt.savefig('Bike_distribution_NYC.png', dpi=300)
plt.show()
```



Analysis 1 - Find which are top 10 most active stations

I have select columns needed for the first question, then I have changed the label for num_docks_available to bikes_in_use, and applied a condition where to be selected only active stations.
Than I have created a new column that shows the ration between bikes in use and capacity, and round it in 2 decimal digits. I order my data based on the new column created firstly and then capacity in descending order.

```
# select columns needed for the first question
# change the name of num_docks_available to bikes_in_use
# condition of only active stations
df_question1 = df_merged[["name", "capacity", "num_docks_available", "station_status"]]
df_question1=df_question1.rename(columns = {'num_docks_available':'bikes_in_use'})
df_question1 = df_question1[df_question1['station_status'] == 'active']
df_question1.head()
```

]:

	name	capacity	bikes_in_use	station_status
0	W 52 St & 11 Ave	55	52	active
1	Franklin St & W Broadway	33	4	active
2	St James Pl & Pearl St	27	3	active
3	Atlantic Ave & Fort Greene Pl	62	16	active
4	W 17 St & 8 Ave	74	1	active

```
# create a new column for the ratio between bikes in use and capacity
# round in 2 digits
# save a csv file
df_question1["Percentage_of_bike_usage"] = df_question1["bikes_in_use"]/(df_question1["capacity"])*100
df_question1["Percentage_of_bike_usage"] = df_question1["Percentage_of_bike_usage"].round(2)
df_question1.to_csv('Active Stations.csv')
```

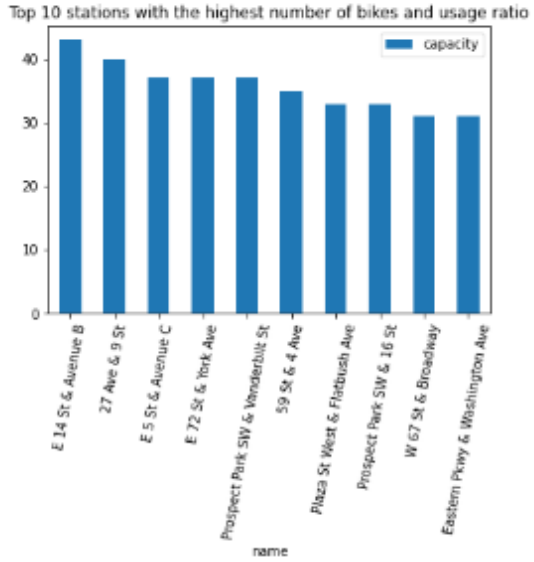
```
# order based on the most active stations
df_question1 = df_question1.sort_values(['Percentage_of_bike_usage','capacity'], ascending=False)
df_question1.head(10)
```

]:

	name	capacity	bikes_in_use	station_status	Percentage_of_bike_usage
224	E 14 St & Avenue B	43	43	active	100.0
687	27 Ave & 9 St	40	40	active	100.0
136	E 5 St & Avenue C	37	37	active	100.0
349	E 72 St & York Ave	37	37	active	100.0

Draw a barchart for the most active statitions.

```
# Draw a vertical bar chart
df_question1_top_10 = df_question1.head(10)
df_question1_top_10.plot.bar(x="name", y="capacity", rot=80, title="Top 10 stations with the highest number of b
plt.savefig('Top_stations_highest_nr_bikes.png', dpi=300)
plt.show()
```



Analysis 2 - Which stations are out_of service? Finding the number of bikes that are found in these stations and comparing with the number of bikes in active stations

Select only the necessary columns for my analysis. Count the number of active and out of service stations.
I grouped by the new dataset using 2 columns name and station_status. Then I find the number of bikes available in each station using sum aggregate function. I give a name to the column of the aggregate function and reset index.

```
# select title, type and country
netflix_country = netflix_df[['title','type', 'country']]

# drop column country and split the countries in different rows
netflix_country = (netflix_country.drop('country', axis=1).join
            (netflix_country.country.str.split(', ',expand=True).stack()
            .reset_index(drop=True, level=1).rename('country')
            ))
```

```
netflix_country
```

```
#create the new dataframe for countries and their count
netflix_country_1 = netflix_country[['title', 'country']].groupby(['country'])['title']
       .count().reset_index().sort_values(
                          'title', ascending= False).head(10).rename(columns = {'title': 'movies_count'})
netflix_country_1
```

```
#Create and merge the dataframes
netflix_country_2 = netflix_country[['title', 'country']].groupby(['country'])['title']
               .count().reset_index().sort_values('title', ascending= False).head(10)

netflix_country_3 = netflix_country[['title', 'type', 'country']].groupby(['country', 'type'])['title']
                    .count().reset_index().sort_values('title', ascending= False).rename(columns = {
                                                'title': 'movies_count'})

netflix_country_chart = netflix_country_2.merge(netflix_country_3, how = 'left', left_on = 'country',
                                                right_on = 'country')

# create a new column % which shows the percentages of movies or
# tv shows from all amount of program.
netflix_country_chart['%'] = ((netflix_country_chart['movies_count']/netflix_country_chart['title'])*100)
                                                .round(2)
```

Sort the dataset based of the number of bikes available, save the first csv file. Then I pivot my dataset and then save again another csv file this time for pivoted new dataset.

```
# sort the dataframe based on nr_of_bikes_available
df_question2.sort_values(['nr_of_bikes_available'], ascending=False)
```

	name	station_status	nr_of_bikes_available
471	Broadway & E 14 St	active	103
823	E 47 St & Park Ave	active	94
1310	Old Slip & South St	active	90
980	Grand Army Plaza & Central Park S	active	55
1135	Laight St & Hudson St	active	56
...	...	...	...
1378	Putnam Ave & Nostrand Ave	out_of_service	0
580	Central Park West & W 85 St	active	0
316	7 Ave & 22 St	active	0
1515	Stuyvesant Ave & Hart St	active	0
1514	Stuyvesant Ave & Gates Ave	active	0

1795 rows × 3 columns

```
# save to csv file
df_question2.to_csv('Active vs Out Of Service.csv')
```

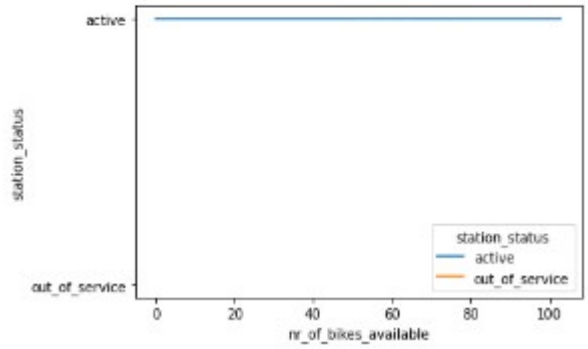
```
import numpy as np
# pivot dataset
df_question2_pivot = pd.pivot_table(df_question2, values='nr_of_bikes_available',
                                    index='station_status', columns='name')

# replace null values in the pivoted table with 0
df_question2_pivot = df_question2_pivot.replace(np.nan,0)
df_question2_pivot = df_question2_pivot.astype(int)
df_question2_pivot.head()
```

name	1 Ave & E 110 St	1 Ave & E 16 St	1 Ave & E 18 St	1 Ave & E 30 St	1 Ave & E 39 St	1 Ave & E 44 St	1 Ave & E 6 St	1 Ave & E 62 St	1 Ave & E 68 St	1 Ave & E 78 St	...	Woodside Ave & 69 St	Woodside Ave & Roosevelt Ave	Woodward Ave & Harman St	Wyckoff Av & Jefferson St	Wyckoff Av & Stanhope St	Wyckoff Ave & Gates Ave	Wyckoff St & Nevins St	Wy Metr
station_status																			
active	5	17	1	22	7	46	34	16	39	5	...	16	10	6	22	19	13	12	
out_of_service	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	

Create a line chart for active and out of service stations.

```
# plot a line graph for active and out of order stations
import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(data = df_question2, x = 'nr_of_bikes_available', y = 'station_status', hue = 'station_status')
plt.savefig('active vs out of service.png', dpi=300)
plt.show()
```



Analysis 3 - Compare number of bikes available with number of electronic bikes available in top stations.

Select the necessary columns for question 3. Then I create a new column for the ratio between bikes in use and capacity and round it in 2 digits. Replace all nan with 0.

```
# select the necessary columns for second question
df_question3 = df_merged[["name", "num_bikes_available", "num_ebikes_available"]]
df_question3
```

	name	num_bikes_available	num_ebikes_available
0	W 52 St & 11 Ave	3	2
1	Franklin St & W Broadway	24	5
2	St James Pl & Pearl St	23	0
3	Atlantic Ave & Fort Greene Pl	45	1
4	W 17 St & 5 Ave	72	3
...	...	...	...
1790	College Ave & E 169 St	14	9
1791	Lafayette Ave & Stuyvesant Ave	2	2
1792	Main St & Plymouth St	41	1
1793	55 St & 56 Dr	0	0
1794	Herkimer St & Eastern Pkwy	0	0

1795 rows × 3 columns

```
# create a new column for the ratio between bikes in use and capacity
# round in 2 digits
# replace nan with 0
df_question3["ebikes_bikes_ratio"] = df_question3["num_ebikes_available"]/(df_question3["num_bikes_available"])*
df_question3["ebikes_bikes_ratio"] = df_question3["ebikes_bikes_ratio"].round(2)
df_question3 = df_question3.replace(np.nan,0)
df_question3
```

	name	num_bikes_available	num_ebikes_available	ebikes_bikes_ratio
0	W 52 St & 11 Ave	3	2	66.67
1	Franklin St & W Broadway	24	5	20.83
2	St James Pl & Pearl St	23	0	0.00
3	Atlantic Ave & Fort Greene Pl	45	1	2.22
4	W 17 St & 5 Ave	72	3	4.17

Then I have filtered only stations where there is at least 1 electronic bike, and I order the dataset based on number of ebikes available in descending order.

```
# stations where there is at least 1 electronic bike
df_question3_at_least_1_ebike = df_question3[df_question3['num_ebikes_available'] >= 1]
df_question3_at_least_1_ebike
```

	name	num_bikes_available	num_ebikes_available	ebikes_bikes_ratio
0	W 52 St & 11 Ave	3	2	66.67
1	Franklin St & W Broadway	24	5	20.83
3	Atlantic Ave & Fort Greene Pl	45	1	2.22
4	W 17 St & 8 Ave	72	3	4.17
7	Barrow St & Hudson St	29	1	3.45
...	...	...	...	...
1785	Vernon Blvd & Queens Plaza S	27	2	7.41
1788	JC Medical Center	2	1	50.00
1790	College Ave & E 169 St	14	9	64.29
1791	Lafayette Ave & Stuyvesant Ave	2	2	100.00
1792	Main St & Plymouth St	41	1	2.44

1099 rows × 4 columns

```
# order based on number of electronic bikes
df_question3_at_least_1_ebike.sort_values(['num_ebikes_available'], ascending=False)
```

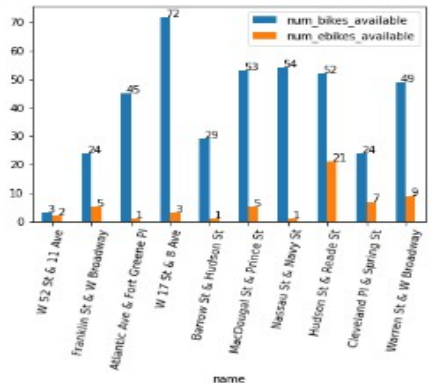
	name	num_bikes_available	num_ebikes_available	ebikes_bikes_ratio
112	E 47 St & Park Ave	94	50	53.19
214	Broadway & W 51 St	65	45	70.59
205	W 44 St & 5 Ave	61	44	72.13
58	Grand Army Plaza & Central Park S	55	41	46.59
795	W 55 St & 6 Ave	72	40	55.56
...	...	...	...	...
603	23 Ave & 27 St	6	1	16.67
605	Steinway St & 23 Ave	4	1	25.00
1216	Hazen St & 20 Ave	19	1	5.26
608	Frederick Douglass Blvd & W 129 St	16	1	6.25
1792	Main St & Plymouth St	41	1	2.44

1099 rows × 4 columns

Save a csv file. Then for the sake of plotting, I have created a new dataset with only 10 records, and plotted a bar chart for bikes and ebikes in these stations.

```
# save a csv file
df_question3_at_least_1_ebike.to_csv("ebikes vs bikes.csv")
# for plot purpose select only 10 records
df_question3_at_least_1_ebike_plot = df_question3_at_least_1_ebike.head(10)
```

```
# plot electronic bikes vs bikes
fig, ax = plt.subplots()
df_question3_at_least_1_ebike_plot.plot.bar(x = 'name', y = ['num_bikes_available', 'num_ebikes_available'], rot
for p in ax.patches:
    ax.annotate(np.round(p.get_height(),decimals=2), (p.get_x()+p.get_width()/2., p.get_height()))
plt.savefig('bikes vs ebikes', dpi=300)
plt.show()
```

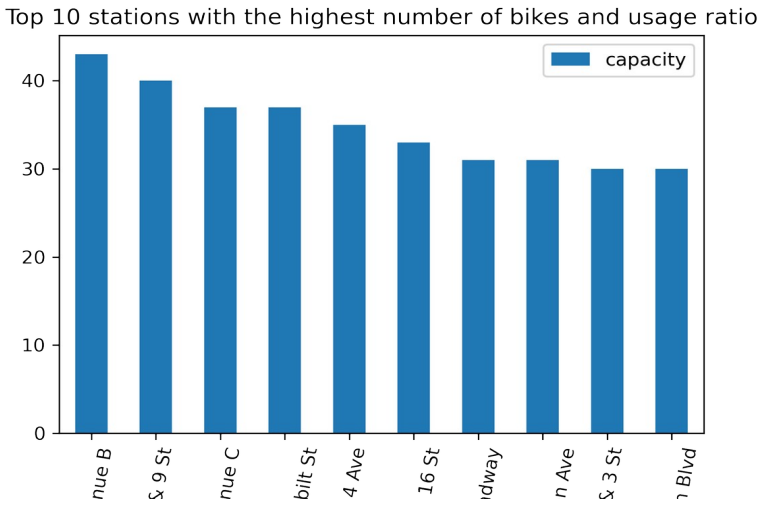


Output Files.

- Analysis 1:
    - 1 csv file

		name	capacity	bikes_in_use	station_status	Percentage_of_bike_usage
2	0	W 52 St & 11 Ave	55	52	active	94.55
3	1	Franklin St & W Broadway	33	4	active	12.12
4	2	St James Pl & Pearl St	27	3	active	11.11
5	3	Atlantic Ave & Fort Greene Pl	62	16	active	25.81
6	4	W 17 St & 8 Ave	74	1	active	1.35
7	5	Park Ave & St Edwards St	53	9	active	16.98
8	6	Lexington Ave & Classon Ave	19	17	active	89.47
9	7	Barrow St & Hudson St	31	0	active	0.0
10	8	MacDougal St & Prince St	56	0	active	0.0
11	9	Clinton St & Joralemon St	50	7	active	14.0
12	10	Nassau St & Navy St	58	2	active	3.45
13	11	Hudson St & Reade St	55	0	active	0.0
14	12	E 2 St & Avenue C	56	53	active	94.64
15	13	Cleveland Pl & Spring St	33	4	active	12.12

    - 1 plotted graph



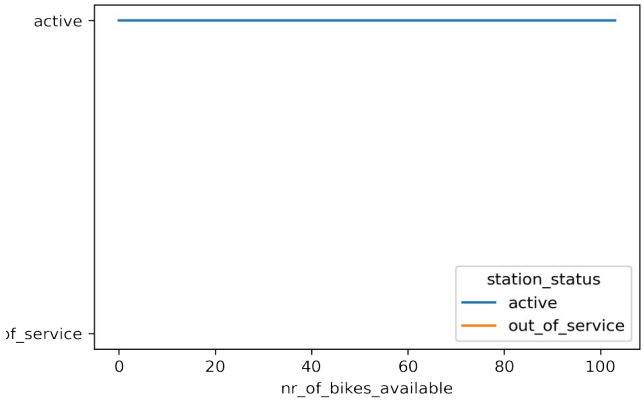
- Analysis 2:
    - 1 csv file before pivoting dataset, but just grouped one.

		name	station_status	nr_of_bikes_available
2	0	1 Ave & E 110 St	active	5
3	1	1 Ave & E 16 St	active	17
4	2	1 Ave & E 18 St	active	1
5	3	1 Ave & E 30 St	active	22
6	4	1 Ave & E 39 St	active	7
7	5	1 Ave & E 44 St	active	46
8	6	1 Ave & E 6 St	active	34
9	7	1 Ave & E 62 St	active	16
10	8	1 Ave & E 68 St	active	39

    - 1 csv file after pivoting dataset.

	station_status	1 Ave & E 110 St	1 Ave & E 16 St	1 Ave & E 18 St	1 Ave &
2	active	5	17	1	22
3	out_of_service	0	0	0	0

- 1 plotted graph



- Analysis 3:

- 1 csv file

		name	num_bikes_available	num_ebikes_available	ebikes_bikes_ratio
2	0	W 52 St & 11 Ave	3	2	66.67
3	1	Franklin St & W Broadway	24	5	20.83
4	3	Atlantic Ave & Fort Greene Pl	45	1	2.22
5	4	W 17 St & 8 Ave	72	3	4.17
6	7	Barrow St & Hudson St	29	1	3.45
7	8	MacDougal St & Prince St	53	5	9.43
8	10	Nassau St & Navy St	54	1	1.85

- 1 plotted graph