Public Recycling Bins in the NYC Boroughs

Background:

Recycling helps protect the environment, ensures environmental sustainability and helps to tackle climate change. The New York Department of Sanitation (DSNY) is responsible for collecting trash, cleaning the city of New York and more relevenatly, they're responible for public recycling bins.

Throughout this project, I'll explore 3 datasets to find the ratio of people per bin in each New York borough. By doing so, I'll be able to provide suggestions to further improve the recycling opportunities in New York.

The 3 datasets are:

- **MTA Turnstiles:** This dataset will be used to find the daily traffic in each station which will help get a gauge of how many people pass by each day.
- **MTA Stations:** This dataset will be used to link the stations to the New York City boroughs.
- **Public Recycling Bins:** This dataset will be used to get the number of recycling bins that are in each New York Borough.

[Dataset #1: MTA Turnstiles] (http://web.mta.info/developers/turnstile.html)

Importing the MTA Turnstiles Dataset

In [1]:
```
import pandas as pd
import datetime
import plotly.express as px
import matplotlib.pyplot as plt
```

In [2]:
```
def get_data(weeks):
    url = "http://web.mta.info/developers/data/nyct/turnstile/turnstile_{}.txt"
    dfs = []
    for week in weeks:
        file_url = url.format(week)
        dfs.append(pd.read_csv(file_url))
    return pd.concat(dfs)

weeks = [200201, 200208, 200215, 200222,
         200229, 200307, 200314, 200321,
         200328, 200404, 200411, 200418]


# February 1st to April 25th

turnstiles_df = get_data(weeks)
```

In [3]:
```
turnstiles_df
```

Out[3]:

	C/A	UNIT	SCP	STATION	LINENAME	DIVISION	DATE	TIME	DESC	ENTRIES	EXITS
0	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/25/2020	03:00:00	REGULAR	7356095	2493703
1	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/25/2020	07:00:00	REGULAR	7356105	2493714
2	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/25/2020	11:00:00	REGULAR	7356170	2493761
3	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/25/2020	15:00:00	REGULAR	7356333	2493812
4	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/25/2020	19:00:00	REGULAR	7356581	2493862
...	...	...	...	...	...	...	...	...	...	...	...
206171	TRAM2	R469	00-05-01	RIT-ROOSEVELT	R	RIT	04/17/2020	05:00:00	REGULAR	5554	514
206172	TRAM2	R469	00-05-01	RIT-ROOSEVELT	R	RIT	04/17/2020	09:00:00	REGULAR	5554	514
206173	TRAM2	R469	00-05-01	RIT-ROOSEVELT	R	RIT	04/17/2020	13:00:00	REGULAR	5554	514
206174	TRAM2	R469	00-05-01	RIT-ROOSEVELT	R	RIT	04/17/2020	17:00:00	REGULAR	5554	514
206175	TRAM2	R469	00-05-01	RIT-ROOSEVELT	R	RIT	04/17/2020	21:00:00	REGULAR	5554	514

2472168 rows × 11 columns

Cleaning the Turnstiles Data and Extracting Features

Removing the white space from column names:

In [4]:
```
turnstiles_df.columns
```

Out[4]:
```
Index(['C/A', 'UNIT', 'SCP', 'STATION', 'LINENAME', 'DIVISION', 'DATE', 'TIME',
       'DESC', 'ENTRIES',
       'EXITS                                                               '],
      dtype='object')
```

In [5]:
```
turnstiles_df.columns = turnstiles_df.columns.str.strip()
turnstiles_df.columns
```

Out[5]:
```
Index(['C/A', 'UNIT', 'SCP', 'STATION', 'LINENAME', 'DIVISION', 'DATE', 'TIME',
       'DESC', 'ENTRIES', 'EXITS'],
      dtype='object')
```

Converting date" and time columns to a single datetime column:

In [6]:
```
turnstiles_df["DATE_TIME"] = pd.to_datetime(turnstiles_df.DATE + " " + turnstiles_df.TIME,
                                            format="%m/%d/%Y %H:%M:%S")
turnstiles_df
```

Out[6]:

	C/A	UNIT	SCP	STATION	LINENAME	DIVISION	DATE	TIME	DESC	ENTRIES	EXITS	DATE_TIME
0	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/25/2020	03:00:00	REGULAR	7356095	2493703	2020-01-25 03:00:00
1	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/25/2020	07:00:00	REGULAR	7356105	2493714	2020-01-25 07:00:00
2	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/25/2020	11:00:00	REGULAR	7356170	2493761	2020-01-25 11:00:00
3	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/25/2020	15:00:00	REGULAR	7356333	2493812	2020-01-25 15:00:00
4	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/25/2020	19:00:00	REGULAR	7356581	2493862	2020-01-25 19:00:00
...	...	...	...	...	...	...	...	...	...	...	...	...
206171	TRAM2	R469	00-05-01	RIT-ROOSEVELT	R	RIT	04/17/2020	05:00:00	REGULAR	5554	514	2020-04-17 05:00:00
206172	TRAM2	R469	00-05-01	RIT-ROOSEVELT	R	RIT	04/17/2020	09:00:00	REGULAR	5554	514	2020-04-17 09:00:00
206173	TRAM2	R469	00-05-01	RIT-ROOSEVELT	R	RIT	04/17/2020	13:00:00	REGULAR	5554	514	2020-04-17 13:00:00
206174	TRAM2	R469	00-05-01	RIT-ROOSEVELT	R	RIT	04/17/2020	17:00:00	REGULAR	5554	514	2020-04-17 17:00:00
206175	TRAM2	R469	00-05-01	RIT-ROOSEVELT	R	RIT	04/17/2020	21:00:00	REGULAR	5554	514	2020-04-17 21:00:00

2472168 rows × 12 columns

Checking for duplicates:

In [7]:
```
temp = (turnstiles_df
 .groupby(["C/A", "UNIT", "SCP", "STATION", "DATE_TIME"])
 .ENTRIES.count()
 .reset_index()
 .sort_values("ENTRIES", ascending=False))

temp["ENTRIES"].unique()
```

Out[7]:
```
array([2, 1], dtype=int64)
```

The result above indicates that some values are duplicated.

In [8]:
```
turnstiles_df.sort_values(["C/A", "UNIT", "SCP", "STATION", "DATE_TIME"],
                          inplace=True, ascending=False)

# dropping duplicated rows:
turnstiles_df.drop_duplicates(subset=["C/A", "UNIT", "SCP", "STATION", "DATE_TIME"], inplace=True)
```

In [9]:
```
temp = (turnstiles_df
 .groupby(["C/A", "UNIT", "SCP", "STATION", "DATE_TIME"])
 .ENTRIES.count()
 .reset_index()
 .sort_values("ENTRIES", ascending=False))
temp["ENTRIES"].unique()
```

Out[9]:
```
array([1], dtype=int64)
```

Now, there are no duplicated values.

In [10]:
```
temp = (turnstiles_df
 .groupby(["C/A", "UNIT", "SCP", "STATION", "DATE_TIME"])
 ["EXITS"].count()
 .reset_index()
 .sort_values("EXITS", ascending=False))

temp["EXITS"].unique()
```

Out[10]:
```
array([1], dtype=int64)
```

Sanity check with the EXITS column, there are no duplicates.

Calculating daily entries and exits:

In [11]:
```
# maximum (cumulative) entries and exits
turnstiles_daily = (turnstiles_df.groupby(["C/A", "UNIT", "SCP", "STATION", "DATE"], as_index=False)['ENTRIES',
```

```
<ipython-input-11-ccb5c105043e>:2: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
  turnstiles_daily = (turnstiles_df.groupby(["C/A", "UNIT", "SCP", "STATION", "DATE"], as_index=False)['ENTRIE
S','EXITS'].max())
```

In [12]:
```
# create columns for the previous date, entries and exits for later analysis

turnstiles_daily[["PREV_DATE", "PREV_ENTRIES", "PREV_EXITS"]] = (turnstiles_daily.groupby(["C/A", "UNIT", "SCP"
                                                                 ["DATE", "ENTRIES","EXITS"].apply(lambda grp:
```

```
<ipython-input-12-5db4e0c49f8e>:3: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
  turnstiles_daily[["PREV_DATE", "PREV_ENTRIES", "PREV_EXITS"]] = (turnstiles_daily.groupby(["C/A", "UNIT", "SC
P", "STATION"])
```

In [13]:
```
# Drop the rows for the earliest date in the df since it has no previous date

turnstiles_daily.dropna(subset=["PREV_DATE"], axis=0, inplace=True)
```

In [14]:
```
# get daily entries, if counter is reset, remove the value. If counter is in reverse, take absolute value.
def get_daily_entries(row, max_counter):
    counter = row["ENTRIES"] - row["PREV_ENTRIES"]
    if counter < 0:
        counter = abs(counter)
    if counter > max_counter:
        return 0
    return counter
```

In [15]:
```
turnstiles_daily["DAILY_ENTRIES"] = turnstiles_daily.apply(get_daily_entries, axis=1, max_counter=100000)
```

In [16]:
```
# get daily exits, if counter is reset, remove the value. If counter is in reverse, take absolute value.
def get_daily_exits(row, max_counter):
    counter = row["EXITS"] - row["PREV_EXITS"]
    if counter < 0:
        counter = abs(counter)
    if counter > max_counter:
        return 0
    return counter
```

In [17]:
```
turnstiles_daily["DAILY_EXITS"] = turnstiles_daily.apply(get_daily_exits, axis=1, max_counter=100000)
```

In [18]:
```
# create new column which describes the summation of daily exits and daily entries.

turnstiles_daily["TOTAL_TRAFFIC"] = turnstiles_daily["DAILY_EXITS"] + turnstiles_daily["DAILY_ENTRIES"]
```

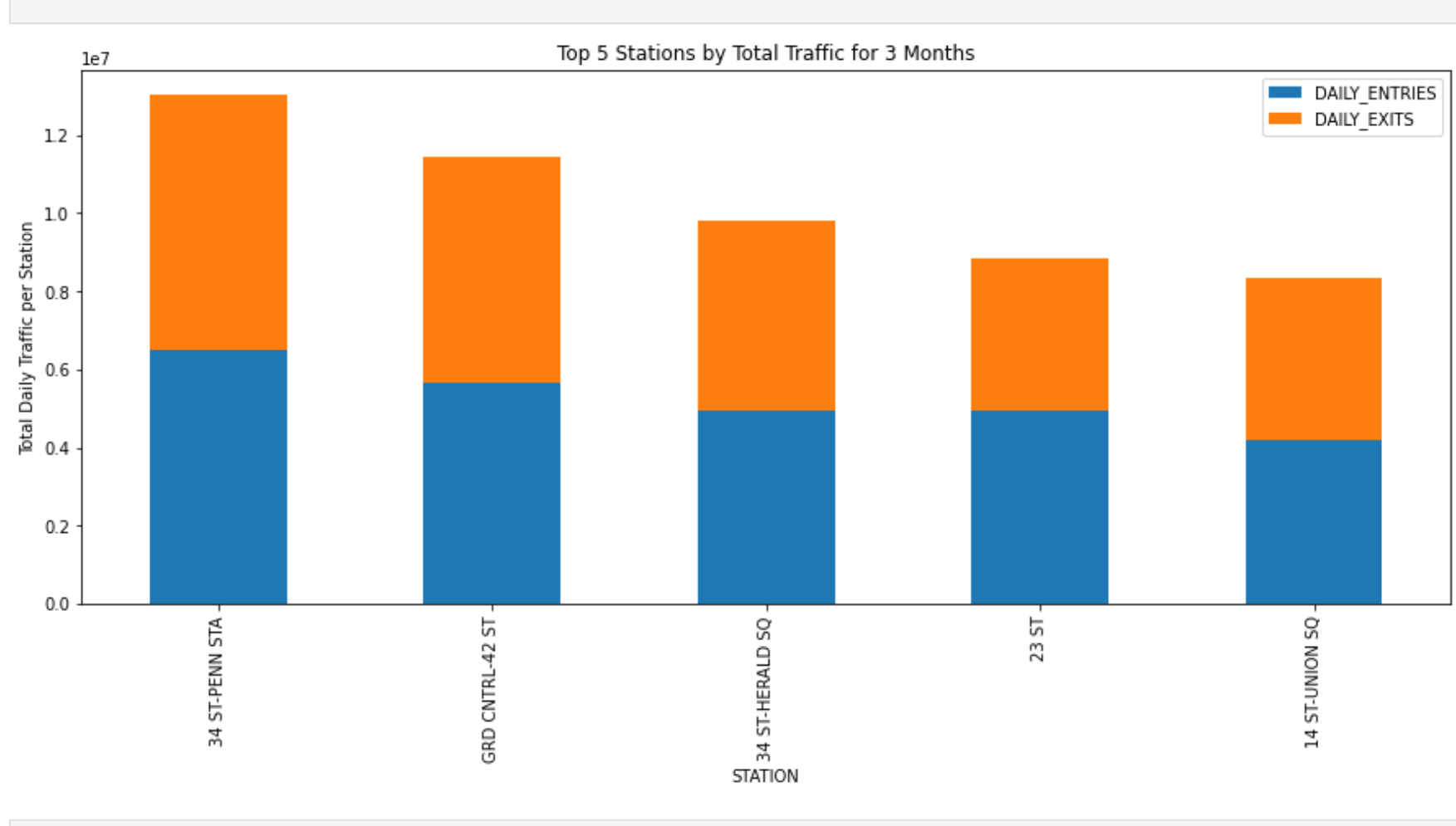
In [19]:
```
# daily entries, daily exits and total traffic per station

station_daily = turnstiles_daily.groupby(["STATION", "DATE"],
                                         as_index = False)[['DAILY_ENTRIES','DAILY_EXITS', "TOTAL_TRAFFIC"]].su
station_daily
```

Out[19]:

	STATION	DATE	DAILY_ENTRIES	DAILY_EXITS	TOTAL_TRAFFIC
0	1 AV	01/26/2020	5886.0	6500.0	12386.0
1	1 AV	01/27/2020	16326.0	18845.0	35171.0
2	1 AV	01/28/2020	17627.0	19885.0	37512.0
3	1 AV	01/29/2020	18010.0	20137.0	38147.0
4	1 AV	01/30/2020	18117.0	20421.0	38538.0
...	...	...	...	...	...
31359	ZEREGA AV	04/13/2020	257.0	313.0	570.0
31360	ZEREGA AV	04/14/2020	326.0	413.0	739.0
31361	ZEREGA AV	04/15/2020	290.0	411.0	701.0
31362	ZEREGA AV	04/16/2020	303.0	396.0	699.0
31363	ZEREGA AV	04/17/2020	315.0	399.0	714.0

31364 rows × 5 columns

In [20]:
```
# Horizontal bar chart - stacked
station_traffic = station_daily.groupby(["STATION"]
                                        ,as_index = False)[['DAILY_ENTRIES','DAILY_EXITS', "TOTAL_TRAFFIC"]].su

station_traffic = station_traffic.sort_values("TOTAL_TRAFFIC", ascending = False)

station_traffic = station_traffic.drop(["TOTAL_TRAFFIC"], axis = 1)



station_traffic[:5].plot(x='STATION', kind='bar', stacked=True,
                         title = "Top 5 Stations by Total Traffic for 3 Months", ylabel = "Total Daily Traffic
                         figsize = (15,6));
```



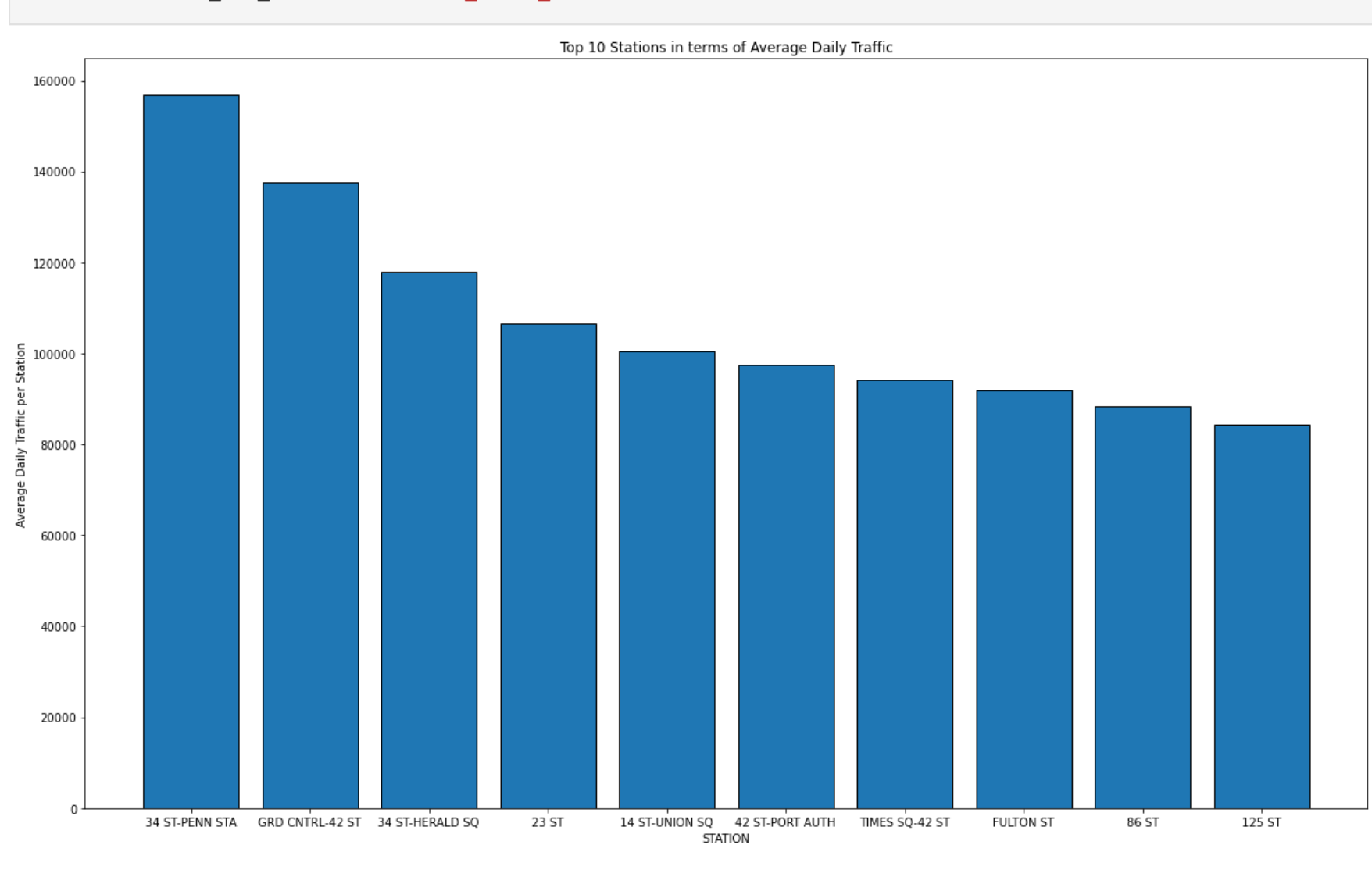
In [21]:
```
# average daily traffic per station

station_avg_traffic = station_daily.groupby(["STATION"])["TOTAL_TRAFFIC"].mean().sort_values(ascending = False)
station_avg_traffic = pd.DataFrame({"STATION": station_avg_traffic.index,
                                    "AVERAGE_DAILY_TRAFFIC" : round(station_avg_traffic)})

station_avg_traffic = station_avg_traffic.reset_index(drop=True)

station_avg_traffic
```

Out[21]:

	STATION	AVERAGE_DAILY_TRAFFIC
0	34 ST-PENN STA	156966.0
1	GRD CNTRL-42 ST	137692.0
2	34 ST-HERALD SQ	118001.0
3	23 ST	106617.0
4	14 ST-UNION SQ	100467.0
...	...	...
373	AQUEDUCT RACETR	893.0
374	BEACH 98 ST	524.0
375	TOMPKINSVILLE	379.0
376	BEACH 105 ST	277.0
377	BROAD CHANNEL	212.0

378 rows × 2 columns

In [22]:
```
# top 10 stations in terms of average daily traffic
plt.figure(figsize=(20,12));
plt.xlabel("STATION")
plt.ylabel("Average Daily Traffic per Station")
plt.title("Top 10 Stations in terms of Average Daily Traffic")
plt.bar(station_avg_traffic['STATION'][:10],
        station_avg_traffic['AVERAGE_DAILY_TRAFFIC'][:10],edgecolor='black', linewidth=1);
```



[Dataset #2: MTA Stations] (http://web.mta.info/developers/data/nyct/subway/Stations.csv)

Importing the MTA Stations dataset:

In [23]:
```
stations = pd.read_csv("http://web.mta.info/developers/data/nyct/subway/Stations.csv")
stations
```

Out[23]:

	Station ID	Complex ID	GTFS Stop ID	Division	Line	Stop Name	Borough	Daytime Routes	Structure	GTFS Latitude	GTFS Longitude	North Direction Label	South Direction Label	ADA
0	1	1	R01	BMT	Astoria	Astoria-Ditmars Blvd	Q	N W	Elevated	40.775036	-73.912034	NaN	Manhattan	0
1	2	2	R03	BMT	Astoria	Astoria Blvd	Q	N W	Elevated	40.770258	-73.917843	Ditmars Blvd	Manhattan	1
2	3	3	R04	BMT	Astoria	30 Av	Q	N W	Elevated	40.766779	-73.921479	Astoria - Ditmars Blvd	Manhattan	0
3	4	4	R05	BMT	Astoria	Broadway	Q	N W	Elevated	40.761820	-73.925508	Astoria - Ditmars Blvd	Manhattan	0
4	5	5	R06	BMT	Astoria	36 Av	Q	N W	Elevated	40.756804	-73.929575	Astoria - Ditmars Blvd	Manhattan	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
491	517	517	S15	SIR	Staten Island	Prince's Bay	SI	SIR	Open Cut	40.525507	-74.200064	St George	Tottenville	0
492	518	518	S14	SIR	Staten Island	Pleasant Plains	SI	SIR	Embankment	40.522410	-74.217847	St George	Tottenville	0
493	519	519	S13	SIR	Staten Island	Richmond Valley	SI	SIR	Open Cut	40.519631	-74.229141	St George	Tottenville	0
494	522	522	S09	SIR	Staten Island	Tottenville	SI	SIR	At Grade	40.512764	-74.251961	St George	NaN	1
495	523	523	S11	SIR	Staten Island	Arthur Kill	SI	SIR	At Grade	40.516578	-74.242096	St George	Tottenville	1

496 rows × 15 columns

From this dataset, I'm only intrested in fnding the Borough for each station.

Exploring and preparing the data:

The data has the stations in the "Stop Name" column.

In [24]:
```
print(len(station_avg_traffic["STATION"].unique()))
len(stations["Stop Name"].unique())
```

```
378
```

Out[24]:
```
379
```

By comparing the number of stations from the two datasets, it is apparent that there is one extra station in the second dataset, which might be a duplicate station.

In [25]:
```
len(stations["Stop Name"].str.strip(".").str.strip().str.upper().unique())
```

Out[25]:
```
378
```

If we strip extra dots and white spaces from the station names, we remove the duplicate station. By observing the station names, there was a duplicate of the station ("14 st-union sq") written once with a dot at the end and once without.

Let's create a dataframe to compare the station names in the two datasets. We start off by sorting both of them to ease the process of finding mismatches.

In [26]:
```
data_1_stations = list(station_avg_traffic["STATION"].unique())
data_2_stations = list(stations["Stop Name"].str.strip(".").str.strip().str.upper().unique())
data_1_stations.sort()
data_2_stations.sort()
compare = pd.DataFrame({"Data1" : data_1_stations, "Data2" : data_2_stations})
compare
```

Out[26]:

	Data1	Data2
0	1 AV	1 AV
1	103 ST	103 ST
2	103 ST-CORONA	103 ST-CORONA PLAZA
3	104 ST	104 ST
4	110 ST	110 ST
...	...	...
373	WOODLAWN	WOODSIDE-61 ST
374	WORLD TRADE CTR	WORLD TRADE CENTER
375	WTC-CORTLANDT	WTC CORTLANDT
376	YORK ST	YORK ST
377	ZEREGA AV	ZEREGA AV

378 rows × 2 columns

It is clearly apparent that the two are almost identical but they have some differences that are not consistent. For example, "103 ST-CORONA" is written with an extra "PLAZA" for the second dataset. Also, "CENTER" is shortened as "CTR" for the first data set and many other inconsistent differences.

Therefore, a better solution is to sort both datasets by the station name, then copy the first data from the second one.

In [27]:
```
stations["STATION"] = stations["Stop Name"].str.strip().str.strip(".").str.upper()
stations.drop_duplicates(subset ="STATION",
                     keep = "first", inplace = True)
stations = stations.sort_values("STATION")
station_avg_traffic = station_avg_traffic.sort_values("STATION")
station_avg_traffic
```

Out[27]:

	STATION	AVERAGE_DAILY_TRAFFIC
79	1 AV	19051.0
41	103 ST	29554.0
64	103 ST-CORONA	21252.0
315	104 ST	3571.0
138	110 ST	12017.0
...	...	...
269	WOODLAWN	4966.0
110	WORLD TRADE CTR	14772.0
133	WTC-CORTLANDT	12190.0
170	YORK ST	9922.0
330	ZEREGA AV	2985.0

378 rows × 2 columns

In [28]:
```
station_avg_traffic["STATION"] = station_avg_traffic["STATION"].str.strip().str.strip(".").str.upper()
stations["STATION"] = station_avg_traffic["STATION"].unique().copy()
```

In [29]:
```
stations_borough = stations[["STATION", "Borough"]]

# replacing borough abbreviations with whole name

stations_borough = stations_borough.replace({"Borough" : {"M" : "Manhattan", "Q" : "Queens",
                                                          "Bx" : "Bronx", "Bk" : "Brooklyn",
                                                          "SI" : "Staten Island"}}, regex = True)
```

Merging the two datasets together:

In [30]:
```
stations_borough_totals = pd.merge(station_avg_traffic, stations_borough, on = "STATION")
stations_borough_totals = stations_borough_totals.sort_values("AVERAGE_DAILY_TRAFFIC", ascending = False).reset
stations_borough_totals
```

Out[30]:

	STATION	AVERAGE_DAILY_TRAFFIC	Borough
0	34 ST-PENN STA	156966.0	Manhattan
1	GRD CNTRL-42 ST	137692.0	Manhattan
2	34 ST-HERALD SQ	118001.0	Manhattan
3	23 ST	106617.0	Manhattan
4	14 ST-UNION SQ	100467.0	Manhattan
...	...	...	...
373	AQUEDUCT RACETR	893.0	Queens
374	BEACH 98 ST	524.0	Bronx
375	TOMPKINSVILLE	379.0	Staten Island
376	BEACH 105 ST	277.0	Queens
377	BROAD CHANNEL	212.0	Queens

378 rows × 3 columns

Now, we have a dataframe that contains each station, where is it located (borough) and its average daily traffic.

[Dataset #3: Public Recycling Bins] (https://data.cityofnewyork.us/Environment/Public-Recycling-Bins/sxx4-xhzg)

Importing the Public Recycling Bins dataset:

In [31]:
```
bins = pd.read_csv("https://data.cityofnewyork.us/api/views/sxx4-xhzg/rows.csv?accessType=DOWNLOAD")
bins
```

Out[31]:

	Borough	Site type	Park/Site Name	Address	Latitude	Longitude
0	Bronx	Subproperty	227th St. Plgd	E 227 St/Bronx River Pkway	40.890849	-73.864224
1	Bronx	Subproperty	Allerton Ballfields	Allerton Ave & Moshulu Pkway	40.848891	-73.877128
2	Bronx	Outdoor	Arthur Ave & E 187 St	Arthur Ave & 187 St	40.855570	-73.887565
3	Bronx	Outdoor	Barstow Mansion	895 Shore Road, Pelham Bay Park	40.871864	-73.805549
4	Bronx	Subproperty	Bradley Playground	2001-2017 Bronx Park E	40.851889	-73.868549
...	...	...	...	...	...	...
540	Staten Island	Indoor	West Brighton Pool	899 Henderson Ave	40.637121	-74.119287
541	Staten Island	Outdoor	Willowbrook Park	Willowbrook Park	40.603832	-74.158697
542	Staten Island	Outdoor	Willowbrook Park	Willowbrook Park	40.603828	-74.161250
543	Staten Island	Outdoor	Wolfe's Pond	Wolfe's Pond	40.517368	-74.190913
544	Staten Island	Outdoor	Wolfe's Pond	Wolfe's Pond	40.518652	-74.187310

545 rows × 6 columns

For this dataset, I'm interested in finding the total number of bins per borough so that I can link it to the merged dataset above.

We can notice that the Boroughs here are written in full and not abbreviated which is why the abbreviations were removed from the previous dataset.

Exploring and preparing the data:

By observing the boroughs, we can notice a white space for the "Brooklyn" borough that needs to be stripped

In [32]:
```
bins["Borough"].unique()
```

Out[32]:
```
array(['Bronx', 'Brooklyn', 'Brooklyn ', 'Manhattan', 'Queens',
       'Staten Island'], dtype=object)
```

In [33]:
```
bins["Borough"] = bins["Borough"].str.strip()
bins["Borough"].unique()
```

Out[33]:
```
array(['Bronx', 'Brooklyn', 'Manhattan', 'Queens', 'Staten Island'],
      dtype=object)
```

Next, we find the number of bins per borough as so:

In [34]:
```
bins_per_borough = bins["Borough"].value_counts()
total_bins = pd.DataFrame({"Borough": bins_per_borough.index, "Bins" : bins_per_borough}).reset_index(drop = Tr
total_bins
```

Out[34]:

	Borough	Bins
0	Manhattan	184
1	Queens	117
2	Bronx	108
3	Brooklyn	94
4	Staten Island	42

Merging the datasets together:

In [35]:
```
station_avg_bins = pd.merge(stations_borough_totals, total_bins, on = "Borough")
station_avg_bins
```

Out[35]:

	STATION	AVERAGE_DAILY_TRAFFIC	Borough	Bins
0	34 ST-PENN STA	156966.0	Manhattan	184
1	GRD CNTRL-42 ST	137692.0	Manhattan	184
2	34 ST-HERALD SQ	118001.0	Manhattan	184
3	23 ST	106617.0	Manhattan	184
4	14 ST-UNION SQ	100467.0	Manhattan	184
...	...	...	...	...
373	BEACH 44 ST	957.0	Queens	117
374	ROCKAWAY PARK B	910.0	Queens	117
375	AQUEDUCT RACETR	893.0	Queens	117
376	BEACH 105 ST	277.0	Queens	117
377	BROAD CHANNEL	212.0	Queens	117

378 rows × 4 columns

Now, we have a dataframe that contains each station, where is it located (borough), its average daily traffic and the number of bins that are in its borough.

Next, we group by the boroughs and find the total average daily traffic per borough.

In [36]:
```
station_borough_bin = station_avg_bins.groupby(["Borough", "Bins"], as_index = False)["AVERAGE_DAILY_TRAFFIC"].
station_borough_bin
```

Out[36]:

	Borough	Bins	AVERAGE_DAILY_TRAFFIC
0	Bronx	108	719694.0
1	Brooklyn	94	1702854.0
2	Manhattan	184	2146054.0
3	Queens	117	854500.0
4	Staten Island	42	204197.0

We then find the ratio of people per bin for each borough.

In [37]:
```
station_borough_bin["People per Bin"] = round(station_borough_bin["AVERAGE_DAILY_TRAFFIC"] / station_borough_bi
station_borough_bin
```

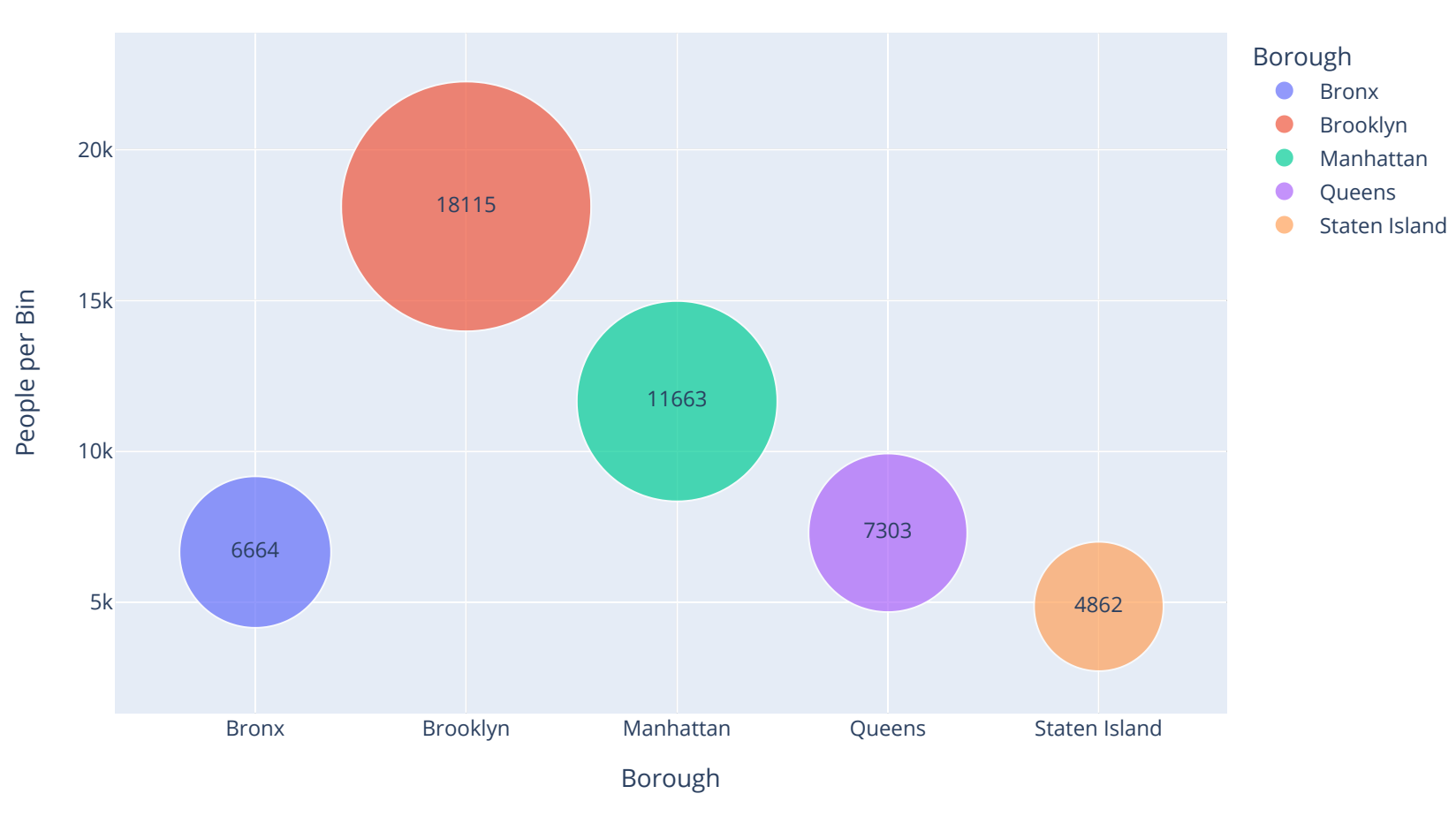
Out[37]:

	Borough	Bins	AVERAGE_DAILY_TRAFFIC	People per Bin
0	Bronx	108	719694.0	6664.0
1	Brooklyn	94	1702854.0	18115.0
2	Manhattan	184	2146054.0	11663.0
3	Queens	117	854500.0	7303.0
4	Staten Island	42	204197.0	4862.0

For example, in Bronx, there is a single bin for every 6664 people.

We then visualize the results as so:

In [38]:
```
fig = px.scatter(station_borough_bin.assign(dataType='People per Bin'), color='Borough', x='Borough', y='People
                 size='People per Bin', size_max=100, text = "People per Bin")
fig.show()
```



Conclusion

Based on the analysis, I suggest increasing the number of recycling bins in Brooklyn and Manhattan to improve recycling bins usage.