MTA Turnstile Data:

Exploratory Data Analysis

Khulud Mansour Amer October 9,2021

Email: kho5hmasour@gmail.com

In the third week of Data Science Bootcamp, I start my first project, which was exploratory data analysis of the MTA turnstile data of NY sub way stations using Python paired with pandas, matplotlib, SQLalchemy, and seaborn.

Background:

I start working at Senior connections nonprofit organization located to the capital area Richmond-Virginia state in January 2021 as a bank account financial analyst. In May 8,2021 the senior connections organization have decided to launch a new location located to NY city and to serve this goal, my manager Ms. Jean Adams decides to sponsor a fundraising in New York city to support the organization financially. Our plan was to target the busiest location in the city, so we agree on sending our team to NY subway stations to collect as much signatures as we can hoping to fill the event space with people passionate about increasing the participation of old people in society by improving quality of life for seniors and find usefulness in their daily task.

As a member of the team, I have been asked to utilize publically accessible MTA data to optimize a perfect location for the team Seniors Connections sending to collect the most signatures, ideally from those who will attend the gala and contribute to our cause.

Setup:

We used historical data from MTA's turnstile (http://web.mta.info/developers/turnstile.html),

Deciding to focus on data from June, July, August 2021.

To be specific we took 12 weeks, 4 weeks from each month, as per the code snippet below:

```
# Source: http://web.mta.info/developers/turnstile.html
def get_data(week_nums):
    url = "http://web.mta.info/developers/data/nyct/turnstile/turnstile_{}.txt"
    dfs = []
    for week_num in week_nums:
        file_url = url.format(week_num)
        dfs.append(pd.read_csv(file_url))
    return pd.concat(dfs)

week_nums = [210605,210612,210619,210626,210703,210710,210717,210724,210807,210814,210821,210828]
turnstiles_df = get_data(week_nums)
```

```
turnstiles_df.to_csv('data.csv',index=False)
```

Preparing the data:

Start with sqlalchemy part:

SQL Part

```
from sqlalchemy import create_engine
engine=create_engine("sqlite:///MTA.db")
turnstiles_df=pd.read_sql('SELECT "from MTAdata;',engine)
turnstiles_df.head(20)
```

73]:

	C/A	UNIT	SCP	STATION	LINENAME	DIVISION	DATE	TIME	DESC	ENTRIES	EXITS
0	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/29/2021	00:00:00	REGULAR	7578734	2590325
1	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/29/2021	04:00:00	REGULAR	7578740	2590327
2	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/29/2021	08:00:00	REGULAR	7578749	2590340
3	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/29/2021	12:00:00	REGULAR	7578789	2590386
4	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/29/2021	16:00:00	REGULAR	7578897	2590418
5	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/29/2021	20:00:00	REGULAR	7579021	2590439
6	A002	R051	02-00-00	59 ST	NQR456W	BMT	05/30/2021	00:00:00	REGULAR	7579078	2590451

Cleaning data:

Check to verify that "C/A", "UNIT", "SCP", "STATION", "DATE_TIME" columns are uniqueRemove the duplicate by using drop. duplicate & group by entries since we are targeting the most traffic stations at the entries only, code as address

Drop Duplicates

```
# we have two entries for same times - deal with it by applying drop_duplicates
turnstiles_df.sort_values(["C/A", "UNIT", "SCP", "STATION", "DATE_TIME"],
                          inplace=True, ascending=False)
turnstiles_df.drop_duplicates(subset=["C/A", "UNIT", "SCP", "STATION", "DATE_TIME"], inplace=True)
```

```
(turnstiles_df
 .groupby(["C/A", "UNIT", "SCP", "STATION", "DATE_TIME"])
 .ENTRIES.count()
 .reset_index()
 .sort_values("ENTRIES", ascending=False)).head(20)
```

below,

Work off of the daily maximum `ENTRIES` calculations and recall that the `ENTRIES` column contains **cumulative entries** on each day. Then calculate **daily entries**, i.e., the number of new entries gained each day.

Next, apply reverse entries and handle the negative values before we finalize our daily count column as shown in the code bellow:

```
turnstiles_daily[turnstiles_daily["ENTRIES"] < turnstiles_daily["PREV_ENTRIES"]].head(20)
```

	C/A	UNIT	SCP	STATION	DATE	ENTRIES	PREV_DATE	PREV_ENTRIES
4369	A011	R080	01-03-00	57 ST-7 AV	05/30/2021	885601428	05/29/2021	885601601.0
4370	A011	R080	01-03-00	57 ST-7 AV	05/31/2021	885601312	05/30/2021	885601428.0
4371	A011	R080	01-03-00	57 ST-7 AV	06/01/2021	885601073	05/31/2021	885601312.0

```
def get_daily_counts(row, max_counter):
    counter = abs(row["ENTRIES"] - row["PREV_ENTRIES"])
    if counter > max_counter:
        counter = row["ENTRIES"]
    return counter
# If counter is > 1Million, then the counter might have been reset.
# Just set it to zero as different counters have different cycle limits
_ = turnstiles_daily.apply(get_daily_counts, axis=1, max_counter=100000)
```

I set the max counter to 100000.
Then, combine turnstiles that fall within the same ControlArea/Unit/Station combo. There are some ControlArea/Unit/Station groups that have a single turnstile, but most have multiple turnstiles -- same value for the C/A, UNIT and STATION columns, different values for the SCP column, and sum the counts from each turnstile belonging to that combo using pd. groupby.

After that I group data by stations and data to find a sum of daily entries

Come up with daily time series for each STATION, by adding up all the turnstiles in a station.

```
station_daily = turnstiles_daily. groupby(["STATION", "DATE"])['DAILY_ENTRIES'].sum().reset_index()
```

```
station_daily. head(20)
```

Finding out the stations with the highest traffic during the time I investigate

The code I use:

```
station_totals = station_daily.groupby('STATION').sum()\
  .sort_values('DAILY_ENTRIES', ascending=False)\
  .reset_index()
```

	STATION	DAILY_ENTRIES
0	34 ST-PENN STA	5286459.0
1	34 ST-HERALD SQ	4024449.0
2	GRD CNTRL-42 ST	3777897.0
3	42 ST-PORT AUTH	3575376.0
4	86 ST	3446357.0
5	23 ST	3441532.0
6	14 ST-UNION SQ	3292190.0
7	125 ST	3266507.0
8	TIMES SQ-42 ST	2937559.0
9	PATH NEW WTC	2848597.0
10	FLUSHING-MAIN	2822127.0
11	FULTON ST	2812066.0
12	JKSN HT-ROOSVLT	2587628.0
13	59 ST	2548531.0
14	96 ST	2489110.0
15	CANAL ST	2465474.0
16	59 ST COLUMBUS	2364520.0
17	14 ST	2043962.0
18	72 ST	1930045.0

Next,

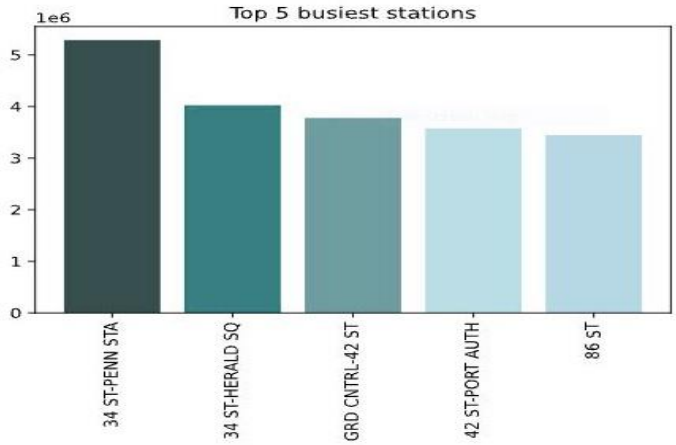
Focus on the top 5 stations:

This step is very important because it targets the most traffic entries among all the stations.

It helps senior connections members to determine the best location to collect signatures from travelers.

Not only that, moreover, but the mobility of the team would have being very difficult if we did not target specific locations and it would consume lots of time and wasting energy.

The top 5 stations are:



* 34 ST-PENN STA

* 34 ST-HERALD SQ

* GRD CNTRL-42 ST

* 42 ST-PORT AUTH

* 86 ST

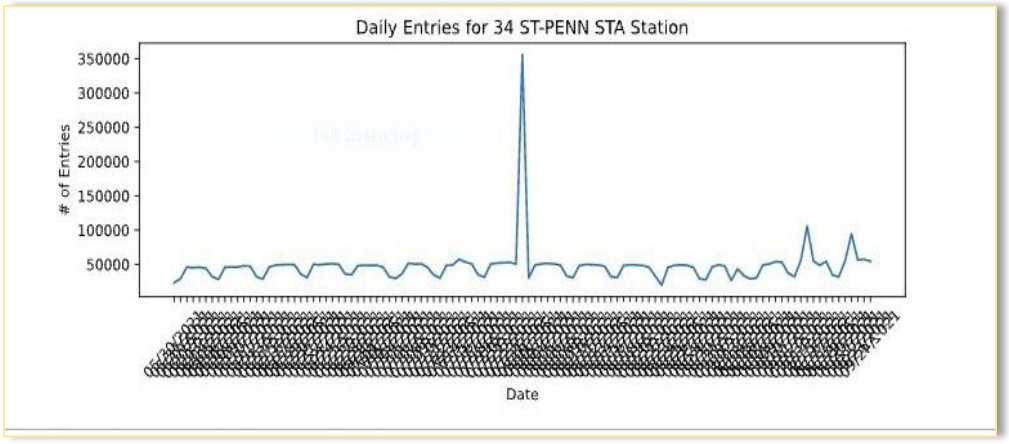
Visualization:

Plot the daily time series for a station.
I pick 34 ST-PENN STA station

```
#Plot the daily time series for a station.
station_daily_34_ST = station_daily[station_daily['STATION'] == '34 ST-PENN STA']
station_daily_34_ST.head()
```

	STATION	DATE	DAILY_ENTRIES
6768	34 ST-PENN STA	05/30/2021	23224.0
6769	34 ST-PENN STA	05/31/2021	29271.0
6770	34 ST-PENN STA	06/01/2021	46271.0
6771	34 ST-PENN STA	06/02/2021	45021.0
6772	34 ST-PENN STA	06/03/2021	45969.0

```
plt.figure(figsize=(15,5))
plt.plot(station_daily_34_ST['DATE'], station_daily_34_ST['DAILY_ENTRIES'])
plt.ylabel('# of Entries')
plt.xlabel('Date')
plt.xticks(rotation=45)
plt.title('Daily Entries for 34 ST-PENN STA Station')
```



Finally,

Make one list of counts for **one** week for one station. Monday's count, Tuesday's count, etc. so it's a list of 7 counts. Make the same list for another week, and another week, and another week, using pandas datetime day of week
The code shown below:

```
station_daily_34_ST['DAY_OF_WEEK_NUM'] =
pd.to_datetime(station_daily_34_ST['DATE']).dt.dayofweek
```

```
station_daily_34_ST['WEEK_OF_YEAR'] = pd.to_datetime(station_daily_34_ST['DATE']).dt.week
```

```
station_daily_34_ST.head()
```

	STATION	DATE	DAILY_ENTRIES	DAY_OF_WEEK_NUM	WEEK_OF_YEAR
6768	34 ST-PENN STA	05/30/2021	23224.0	6	21
6769	34 ST-PENN STA	05/31/2021	29271.0	0	22
6770	34 ST-PENN STA	06/01/2021	46271.0	1	22
6771	34 ST-PENN STA	06/02/2021	45021.0	2	22
6772	34 ST-PENN STA	06/03/2021	45969.0	3	22

Final plot of the top three busiest stations :

`plt.plot(week_count_list)` for every `week_count_list` I create.

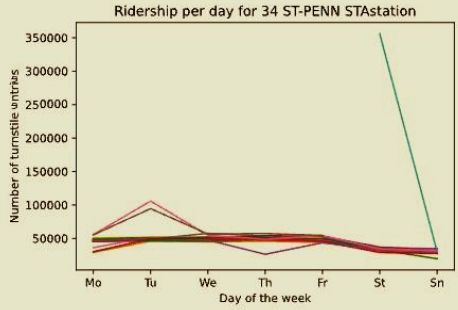
A rainbow plot of weekly commute numbers on top of each other.

34 ST-PENN STA

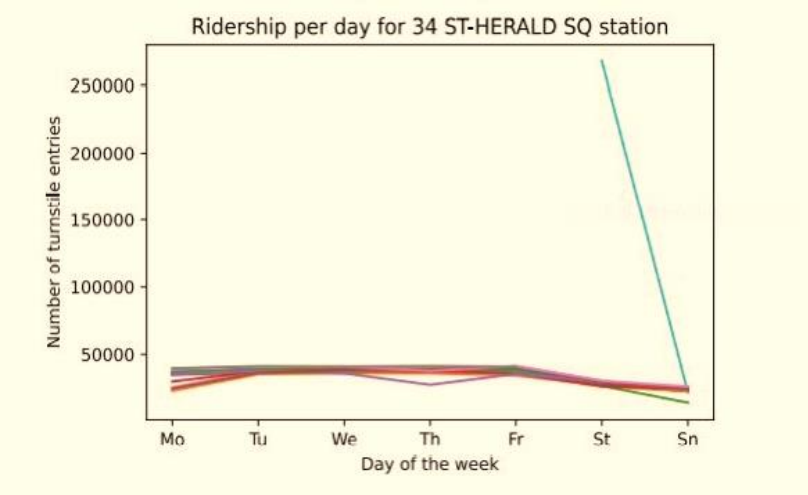
```
for i, group in station_daily_34_ST.groupby('WEEK_OF_YEAR'):
    plt.plot(group['DAY_OF_WEEK_NUM'], group['DAILY_ENTRIES'])

plt.xlabel('Day of the week')
plt.ylabel('Number of turnstile entries')
plt.xticks(np.arange(7),['Mo','Tu','We','Th','Fr','St','Sn'])
plt.title('Ridership per day for 34 ST-PENN STAstation')
```

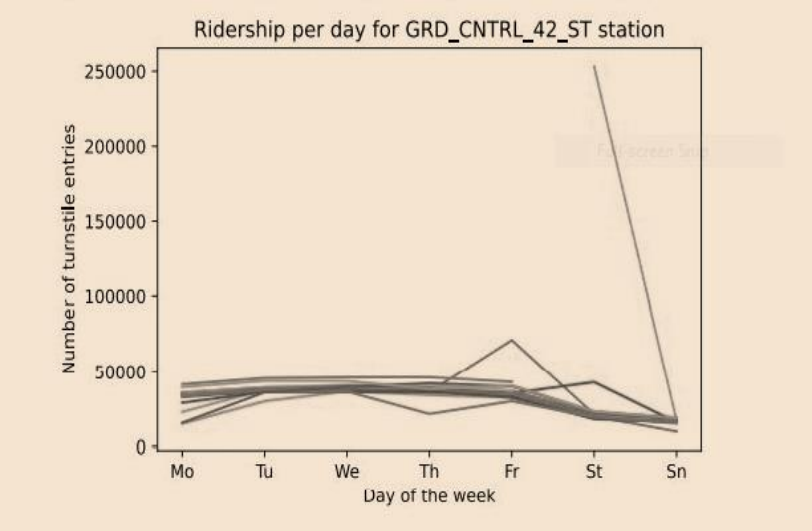
```
Text(0.5, 1.0, 'Ridership per day for 34 ST-PENN STAstation')
```



34 ST- HERALD SQ



GRD-CNTRL_42_ST



END OF PROJECT.