MTA Usage and Covid-19 Case Rate

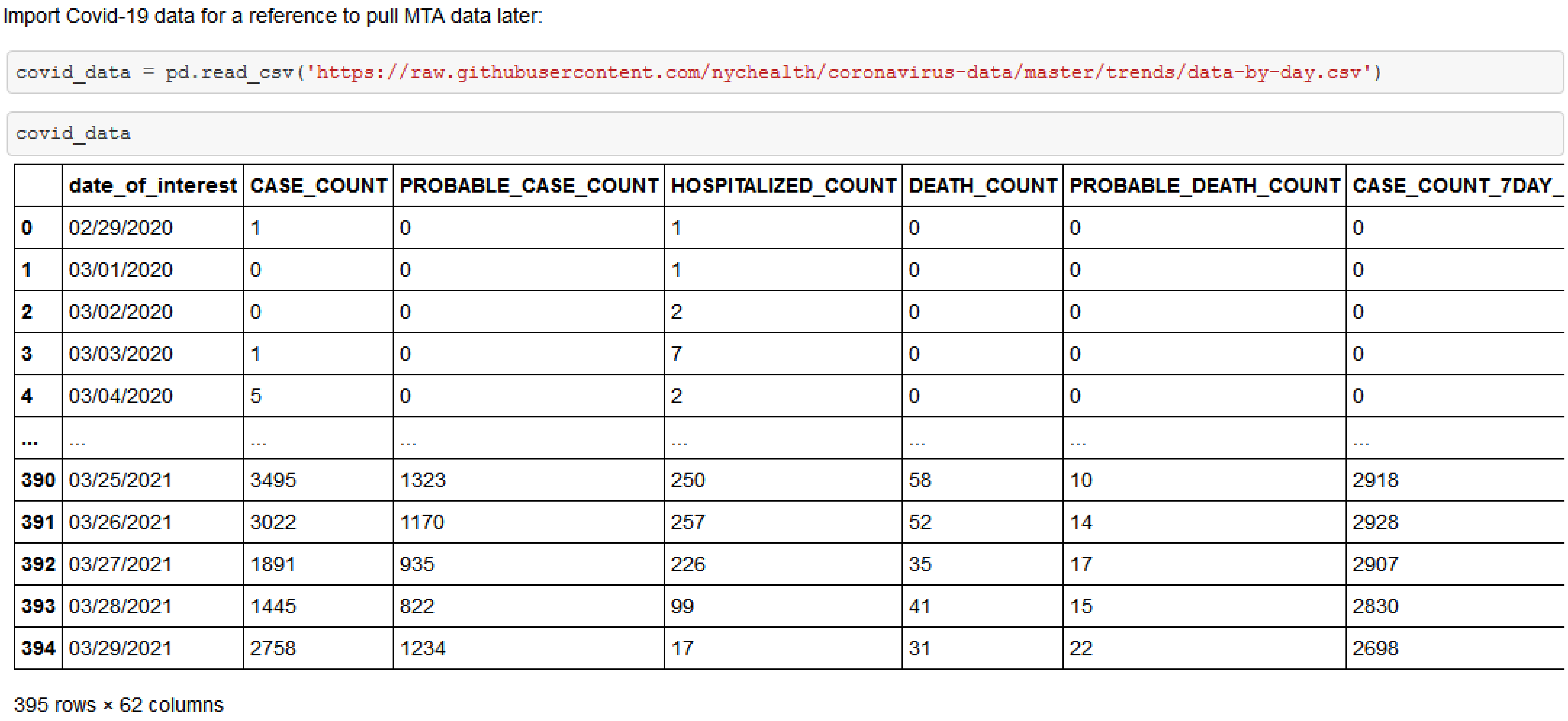
NICHOLAS DELL'AQUILO

Background

- Coronavirus- how did it impact MTA traffic?
- Is there a pattern connecting riding the subway and case rate?

This data is all easily publicly available; it should be used for analysis, so we can better understand this situation if something similar happens again in the future.

Importing Data



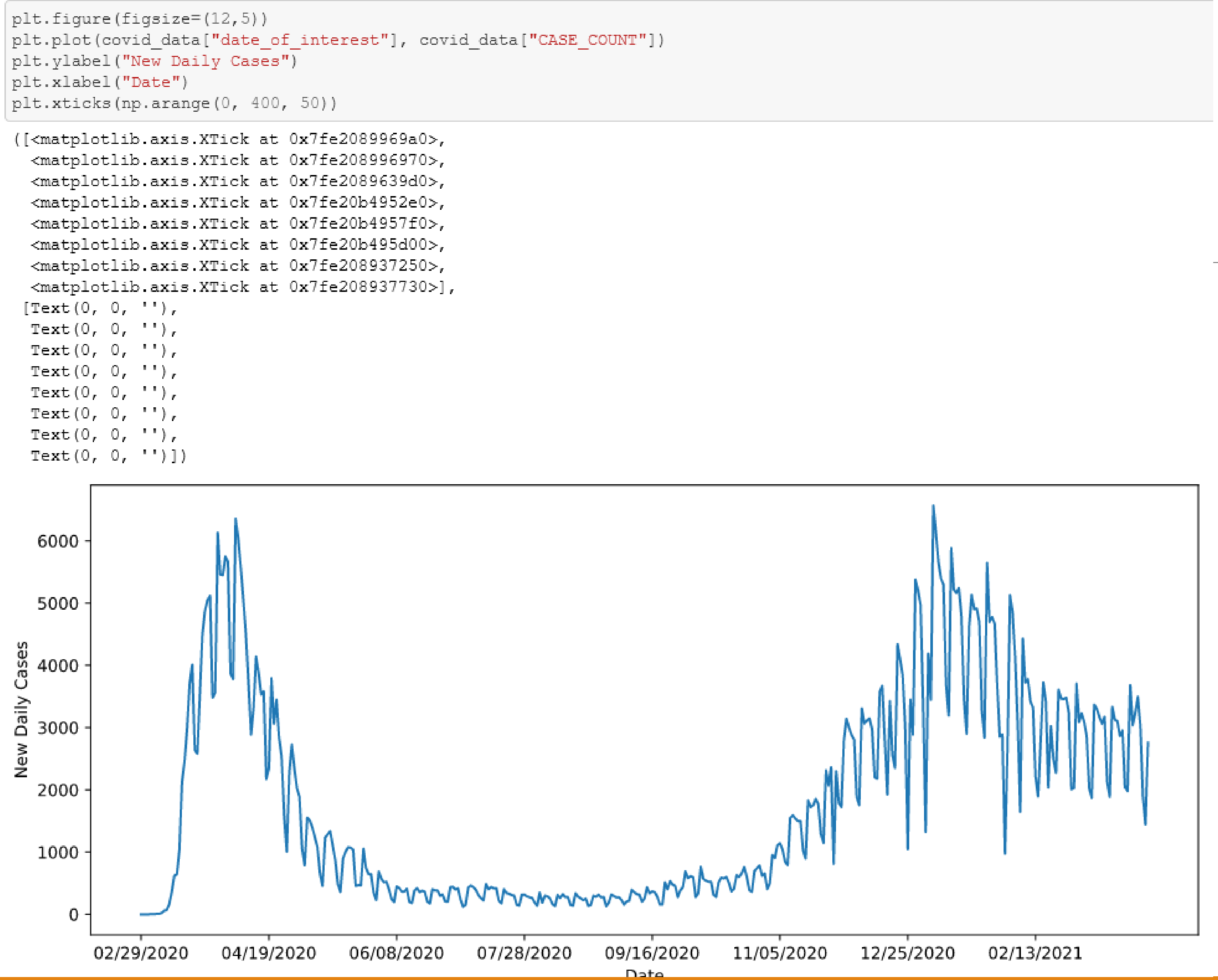
Import Covid-19 data for a reference to pull MTA data later:

```
covid_data = pd.read_csv('https://raw.githubusercontent.com/nychealth/coronavirus-data/master/trends/data-by-day.csv')
```

```
covid_data
```

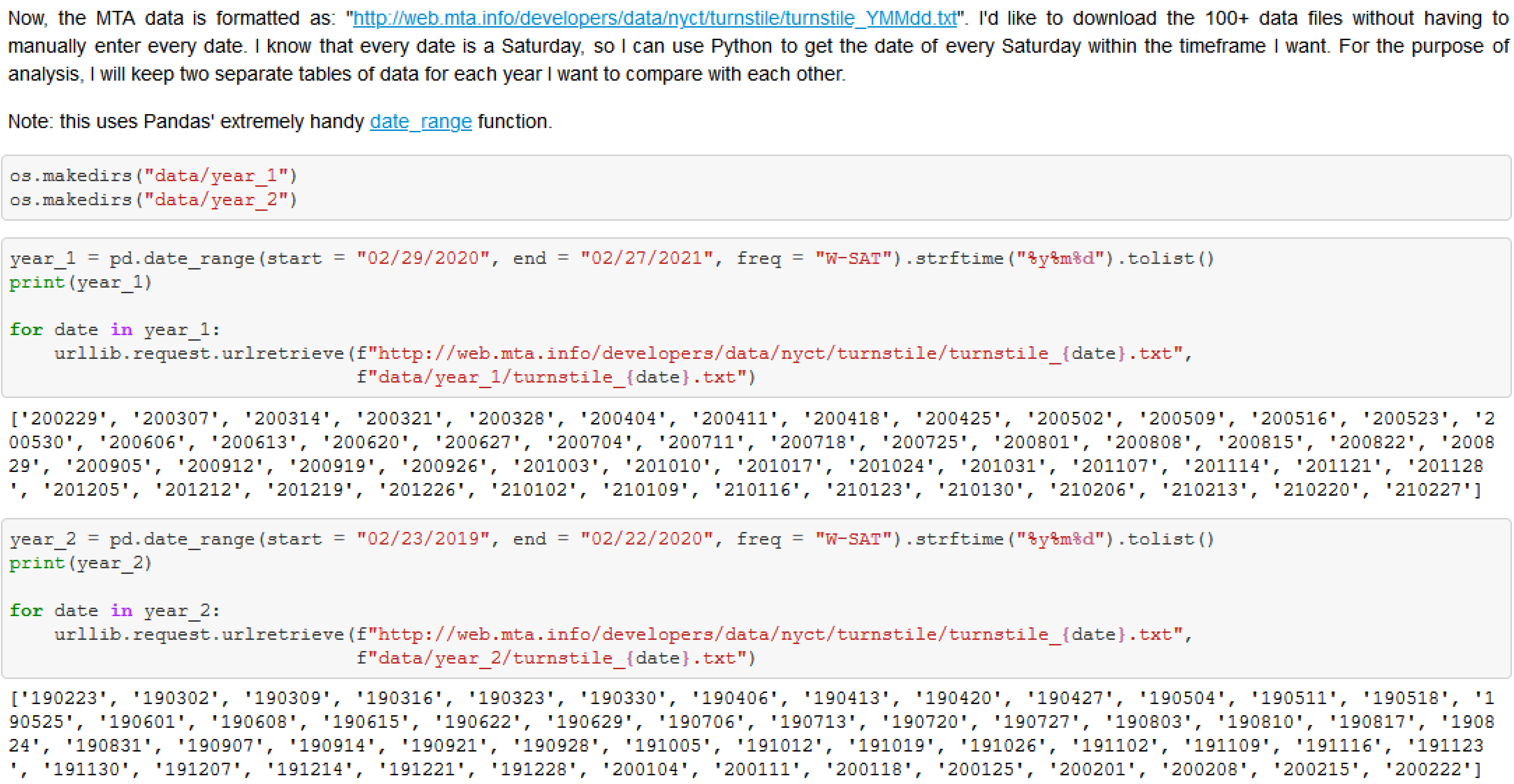
	date_of_interest	CASE_COUNT	PROBABLE_CASE_COUNT	HOSPITALIZED_COUNT	DEATH_COUNT	PROBABLE_DEATH_COUNT	CASE_COUNT_7DAY_
0	02/29/2020	1	0	1	0	0	0
1	03/01/2020	0	0	1	0	0	0
2	03/02/2020	0	0	2	0	0	0
3	03/03/2020	1	0	7	0	0	0
4	03/04/2020	5	0	2	0	0	0
...	...	...	...	...	...	...	...
390	03/25/2021	3495	1323	250	58	10	2918
391	03/26/2021	3022	1170	257	52	14	2928
392	03/27/2021	1891	935	226	35	17	2907
393	03/28/2021	1445	822	99	41	15	2830
394	03/29/2021	2758	1234	17	31	22	2698

395 rows × 62 columns



```
plt.figure(figsize=(12,5))
plt.plot(covid_data["date_of_interest"], covid_data["CASE_COUNT"])
plt.ylabel("New Daily Cases")
plt.xlabel("Date")
plt.xticks(np.arange(0, 400, 50))
```

```
([<matplotlib.axis.XTick at 0x7fe2089969a0>,
  <matplotlib.axis.XTick at 0x7fe208996970>,
  <matplotlib.axis.XTick at 0x7fe2089639d0>,
  <matplotlib.axis.XTick at 0x7fe20b4952e0>,
  <matplotlib.axis.XTick at 0x7fe20b4957f0>,
  <matplotlib.axis.XTick at 0x7fe20b495d00>,
  <matplotlib.axis.XTick at 0x7fe208937250>,
  <matplotlib.axis.XTick at 0x7fe208937730>],
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



Now, the MTA data is formatted as: "http://web.mta.info/developers/data/nyct/turnstile/turnstile_YMMdd.txt". I'd like to download the 100+ data files without having to manually enter every date. I know that every date is a Saturday, so I can use Python to get the date of every Saturday within the timeframe I want. For the purpose of analysis, I will keep two separate tables of data for each year I want to compare with each other.

Note: this uses Pandas' extremely handy date_range function.

```
os.makedirs("data/year_1")
os.makedirs("data/year_2")
```

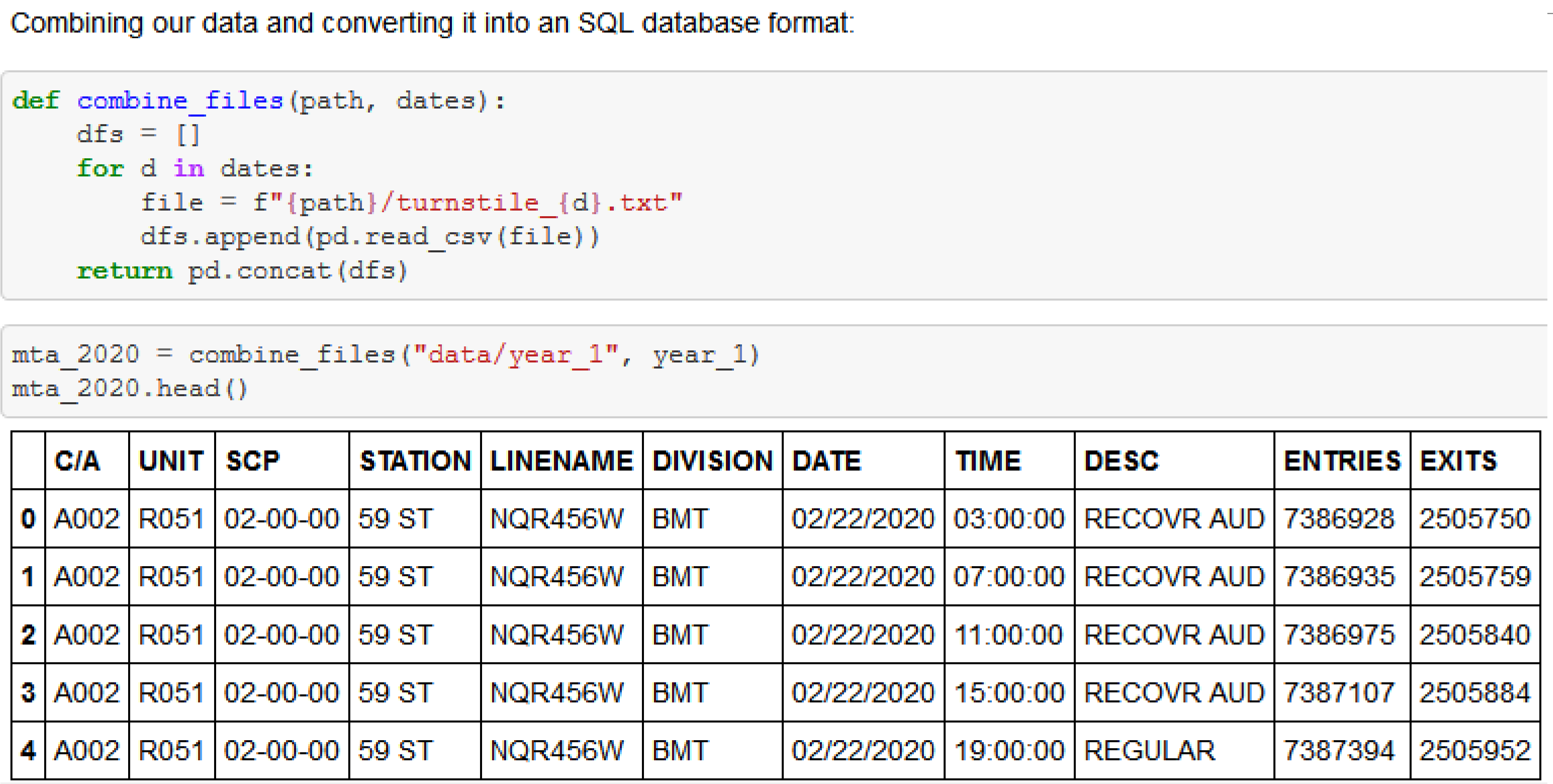
```
year_1 = pd.date_range(start = "02/29/2020", end = "02/27/2021", freq = "W-SAT").strftime("%y%m%d").tolist()
print(year_1)

for date in year_1:
    urllib.request.urlretrieve(f"http://web.mta.info/developers/data/nyct/turnstile/turnstile_{date}.txt",
                               f"data/year_1/turnstile_{date}.txt")
```

```
['200229', '200307', '200314', '200321', '200328', '200404', '200411', '200418', '200425', '200502', '200509', '200516', '200523', '2
00530', '200606', '200613', '200620', '200627', '200704', '200711', '200718', '200725', '200801', '200808', '200815', '200822', '2008
29', '200905', '200912', '200919', '200926', '201003', '201010', '201017', '201024', '201031', '201107', '201114', '201121', '201128
', '201205', '201212', '201219', '201226', '210102', '210109', '210116', '210123', '210130', '210206', '210213', '210220', '210227']
```

```
year_2 = pd.date_range(start = "02/23/2019", end = "02/22/2020", freq = "W-SAT").strftime("%y%m%d").tolist()
print(year_2)

for date in year_2:
    urllib.request.urlretrieve(f"http://web.mta.info/developers/data/nyct/turnstile/turnstile_{date}.txt",
                               f"data/year_2/turnstile_{date}.txt")
```

```
['190223', '190302', '190309', '190316', '190323', '190330', '190406', '190413', '190420', '190427', '190504', '190511', '190518', '1
90525', '190601', '190608', '190615', '190622', '190629', '190706', '190713', '190720', '190727', '190803', '190810', '190817', '1908
24', '190831', '190907', '190914', '190921', '190928', '191005', '191012', '191019', '191026', '191102', '191109', '191116', '191123
', '191130', '191207', '191214', '191221', '191228', '200104', '200111', '200118', '200125', '200201', '200208', '200215', '200222']
```

Combining data files

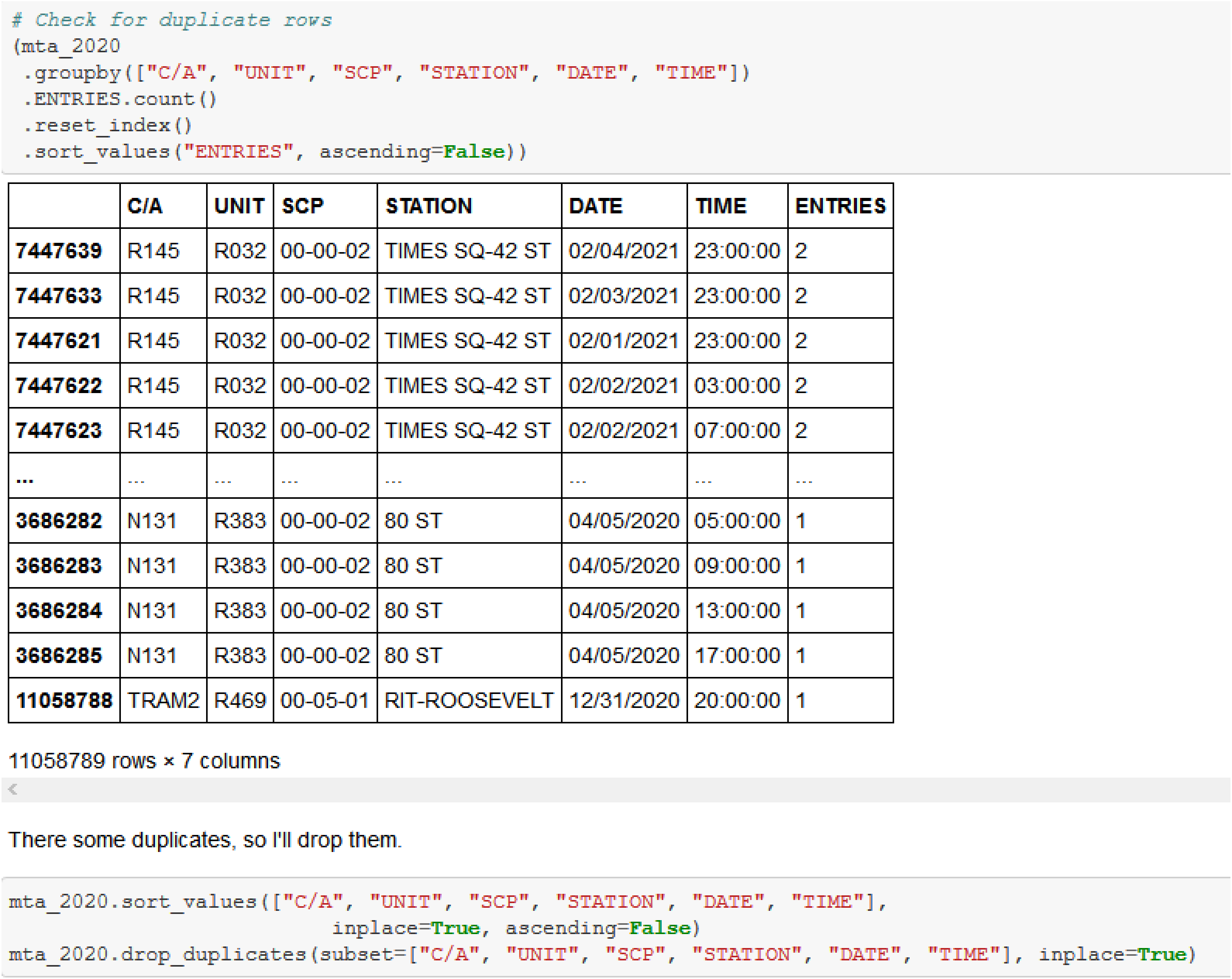


Combining our data and converting it into an SQL database format:

```
def combine_files(path, dates):
    dfs = []
    for d in dates:
        file = f"{path}/turnstile_{d}.txt"
        dfs.append(pd.read_csv(file))
    return pd.concat(dfs)
```

```
mta_2020 = combine_files("data/year_1", year_1)
mta_2020.head()
```

	C/A	UNIT	SCP	STATION	LINENAME	DIVISION	DATE	TIME	DESC	ENTRIES	EXITS
0	A002	R051	02-00-00	59 ST	NQR456W	BMT	02/22/2020	03:00:00	RECOVR AUD	7386928	2505750
1	A002	R051	02-00-00	59 ST	NQR456W	BMT	02/22/2020	07:00:00	RECOVR AUD	7386935	2505759
2	A002	R051	02-00-00	59 ST	NQR456W	BMT	02/22/2020	11:00:00	RECOVR AUD	7386975	2505840
3	A002	R051	02-00-00	59 ST	NQR456W	BMT	02/22/2020	15:00:00	RECOVR AUD	7387107	2505884
4	A002	R051	02-00-00	59 ST	NQR456W	BMT	02/22/2020	19:00:00	REGULAR	7387394	2505952



```
# Check for duplicate rows
(mta_2020
 .groupby(["C/A", "UNIT", "SCP", "STATION", "DATE", "TIME"])
 .ENTRIES.count()
 .reset_index()
 .sort_values("ENTRIES", ascending=False))
```

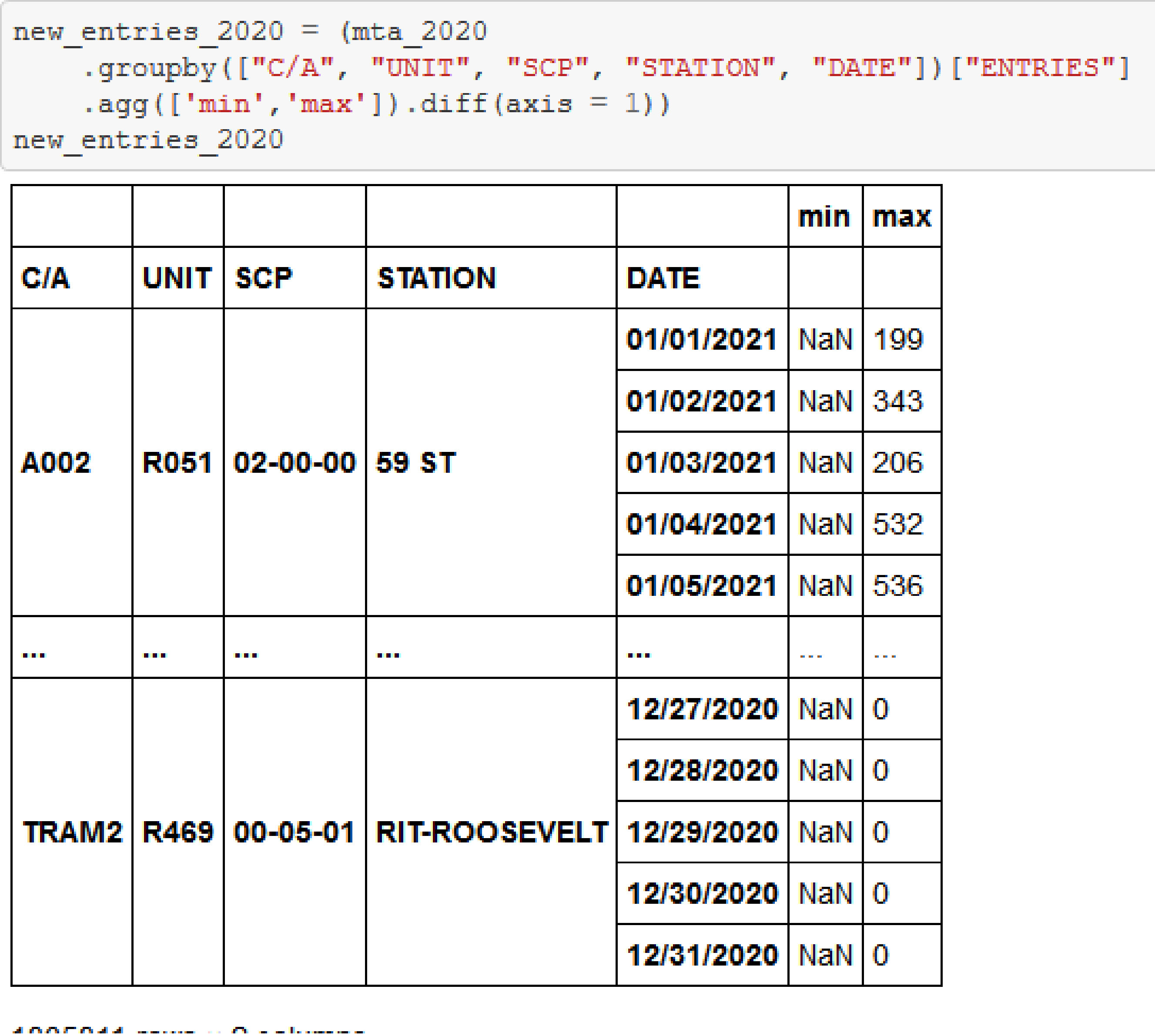
	C/A	UNIT	SCP	STATION	DATE	TIME	ENTRIES
7447639	R145	R032	00-00-02	TIMES SQ-42 ST	02/04/2021	23:00:00	2
7447633	R145	R032	00-00-02	TIMES SQ-42 ST	02/03/2021	23:00:00	2
7447621	R145	R032	00-00-02	TIMES SQ-42 ST	02/01/2021	23:00:00	2
7447622	R145	R032	00-00-02	TIMES SQ-42 ST	02/02/2021	03:00:00	2
7447623	R145	R032	00-00-02	TIMES SQ-42 ST	02/02/2021	07:00:00	2
...	...	...	...	...	...	...	...
3686282	N131	R383	00-00-02	80 ST	04/05/2020	05:00:00	1
3686283	N131	R383	00-00-02	80 ST	04/05/2020	09:00:00	1
3686284	N131	R383	00-00-02	80 ST	04/05/2020	13:00:00	1
3686285	N131	R383	00-00-02	80 ST	04/05/2020	17:00:00	1
11058788	TRAM2	R469	00-05-01	RIT-ROOSEVELT	12/31/2020	20:00:00	1

11058789 rows × 7 columns

There some duplicates, so I'll drop them.

```
mta_2020.sort_values(["C/A", "UNIT", "SCP", "STATION", "DATE", "TIME"],
                            inplace=True, ascending=False)
mta_2020.drop_duplicates(subset=["C/A", "UNIT", "SCP", "STATION", "DATE", "TIME"], inplace=True)
```

Pandas function for calculating new entries per day

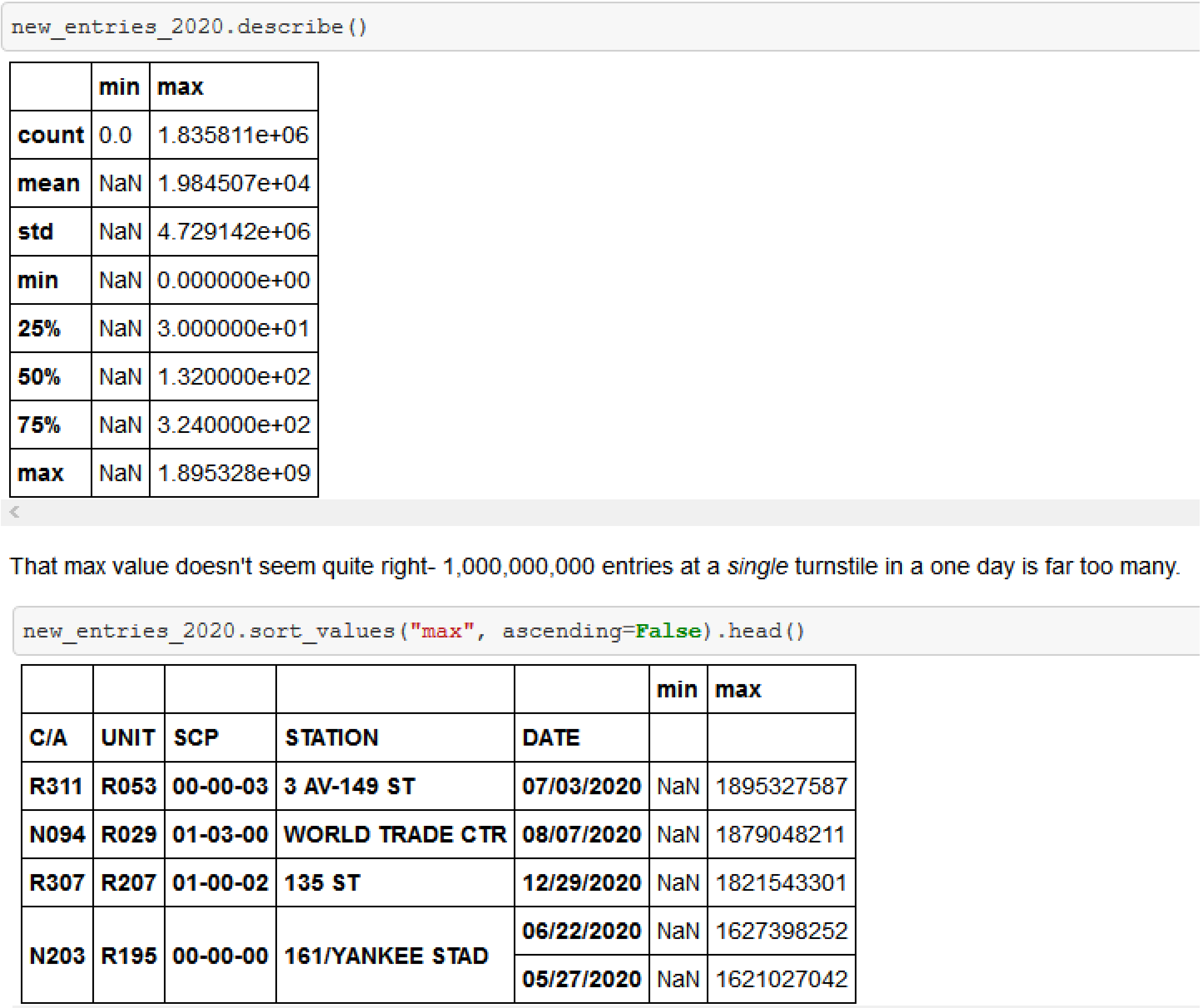


```
new_entries_2020 = (mta_2020
    .groupby(["C/A", "UNIT", "SCP", "STATION", "DATE"])["ENTRIES"]
    .agg(['min','max']).diff(axis = 1))
new_entries_2020
```

					min	max
C/A	**UNIT**	**SCP**	**STATION**	**DATE**		
A002	**R051**	**02-00-00**	**59 ST**	**01/01/2021**	NaN	199
				01/02/2021	NaN	343
				01/03/2021	NaN	206
				01/04/2021	NaN	532
				01/05/2021	NaN	536
...	**...**	**...**	**...**	**...**	...	...
TRAM2	**R469**	**00-05-01**	**RIT-ROOSEVELT**	**12/27/2020**	NaN	0
				12/28/2020	NaN	0
				12/29/2020	NaN	0
				12/30/2020	NaN	0
				12/31/2020	NaN	0

1005011 rows × 2 columns

Analyzing data: are there outliers that need to be removed?



```
new_entries_2020.describe()
```

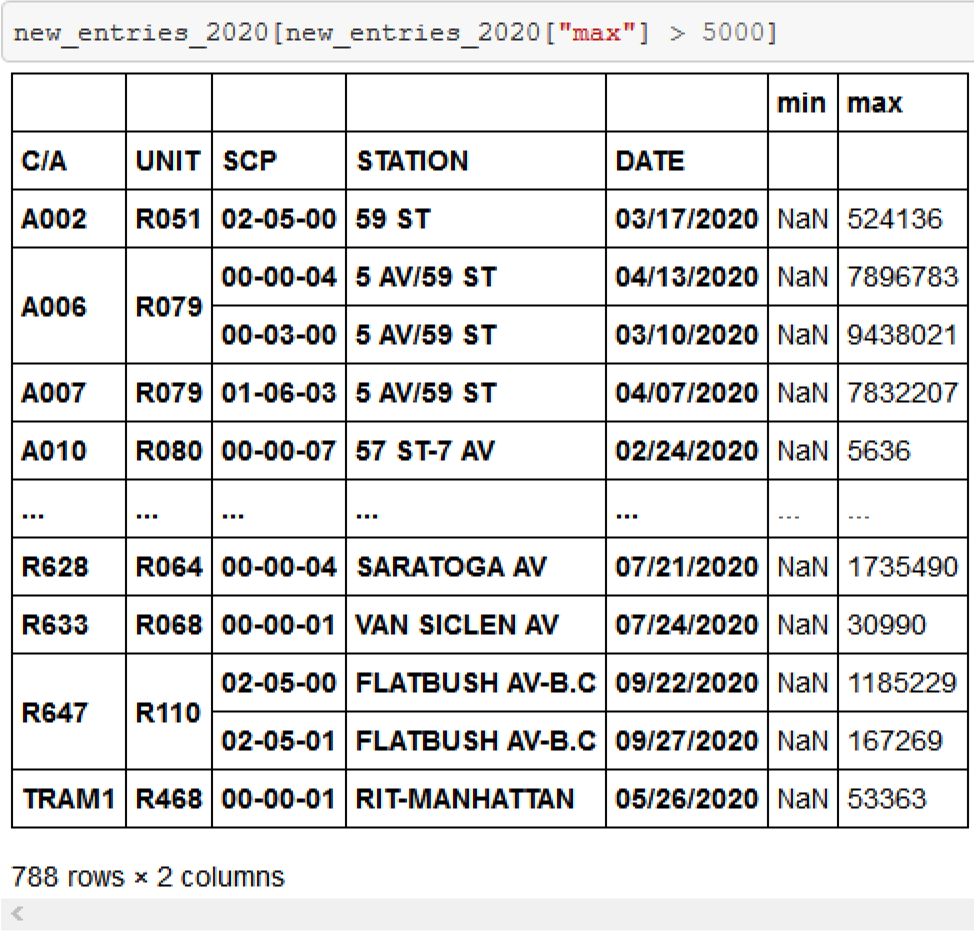
	min	max
count	0.0	1.835811e+06
mean	NaN	1.984507e+04
std	NaN	4.729142e+06
min	NaN	0.000000e+00
25%	NaN	3.000000e+01
50%	NaN	1.320000e+02
75%	NaN	3.240000e+02
max	NaN	1.895328e+09

That max value doesn't seem quite right- 1,000,000,000 entries at a *single* turnstile in a one day is far too many.

```
new_entries_2020.sort_values("max", ascending=False).head()
```

					min	max
C/A	**UNIT**	**SCP**	**STATION**	**DATE**		
R311	**R053**	**00-00-03**	**3 AV-149 ST**	**07/03/2020**	NaN	1895327587
N094	**R029**	**01-03-00**	**WORLD TRADE CTR**	**08/07/2020**	NaN	1879048211
R307	**R207**	**01-00-02**	**135 ST**	**12/29/2020**	NaN	1821543301
N203	**R195**	**00-00-00**	**161/YANKEE STAD**	**06/22/2020**	NaN	1627398252
				05/27/2020	NaN	1621027042

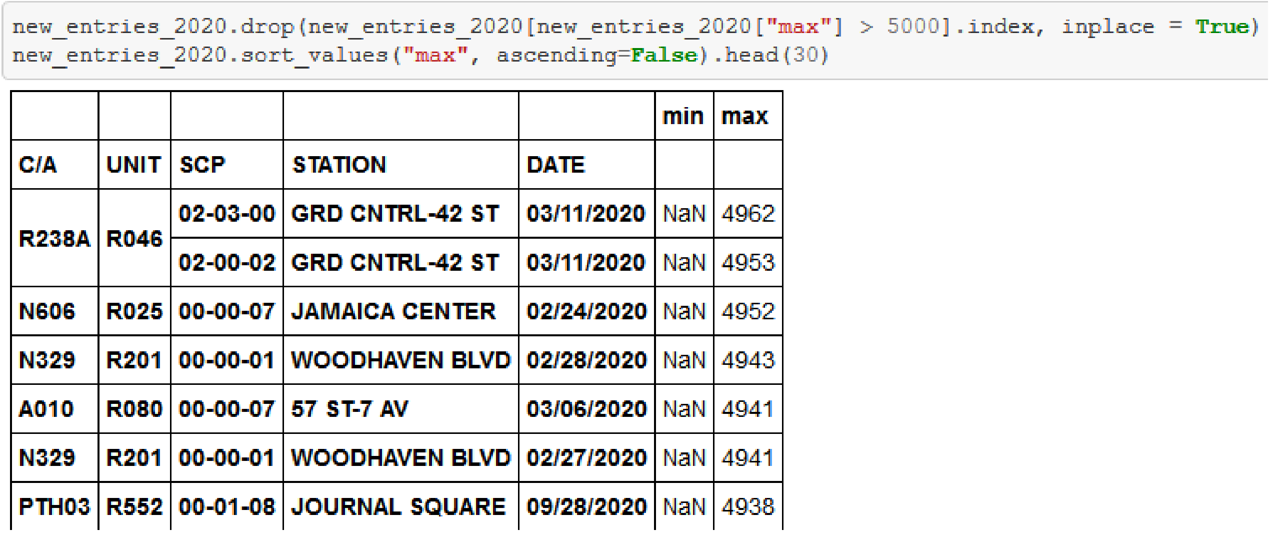
Cleaning data – Drop outliers



```
new_entries_2020[new_entries_2020["max"] > 5000]
```

					min	max
C/A	UNIT	SCP	STATION	DATE		
A002	R051	02-05-00	59 ST	03/17/2020	NaN	524136
A006	R079	00-00-04	5 AV/59 ST	04/13/2020	NaN	7896783
		00-03-00	5 AV/59 ST	03/10/2020	NaN	9438021
A007	R079	01-06-03	5 AV/59 ST	04/07/2020	NaN	7832207
A010	R080	00-00-07	57 ST-7 AV	02/24/2020	NaN	5636
...	...	...	...	...	...	...
R628	R064	00-00-04	SARATOGA AV	07/21/2020	NaN	1735490
R633	R068	00-00-01	VAN SICLEN AV	07/24/2020	NaN	30990
R647	R110	02-05-00	FLATBUSH AV-B.C	09/22/2020	NaN	1185229
		02-05-01	FLATBUSH AV-B.C	09/27/2020	NaN	167269
TRAM1	R468	00-00-01	RIT-MANHATTAN	05/26/2020	NaN	53363

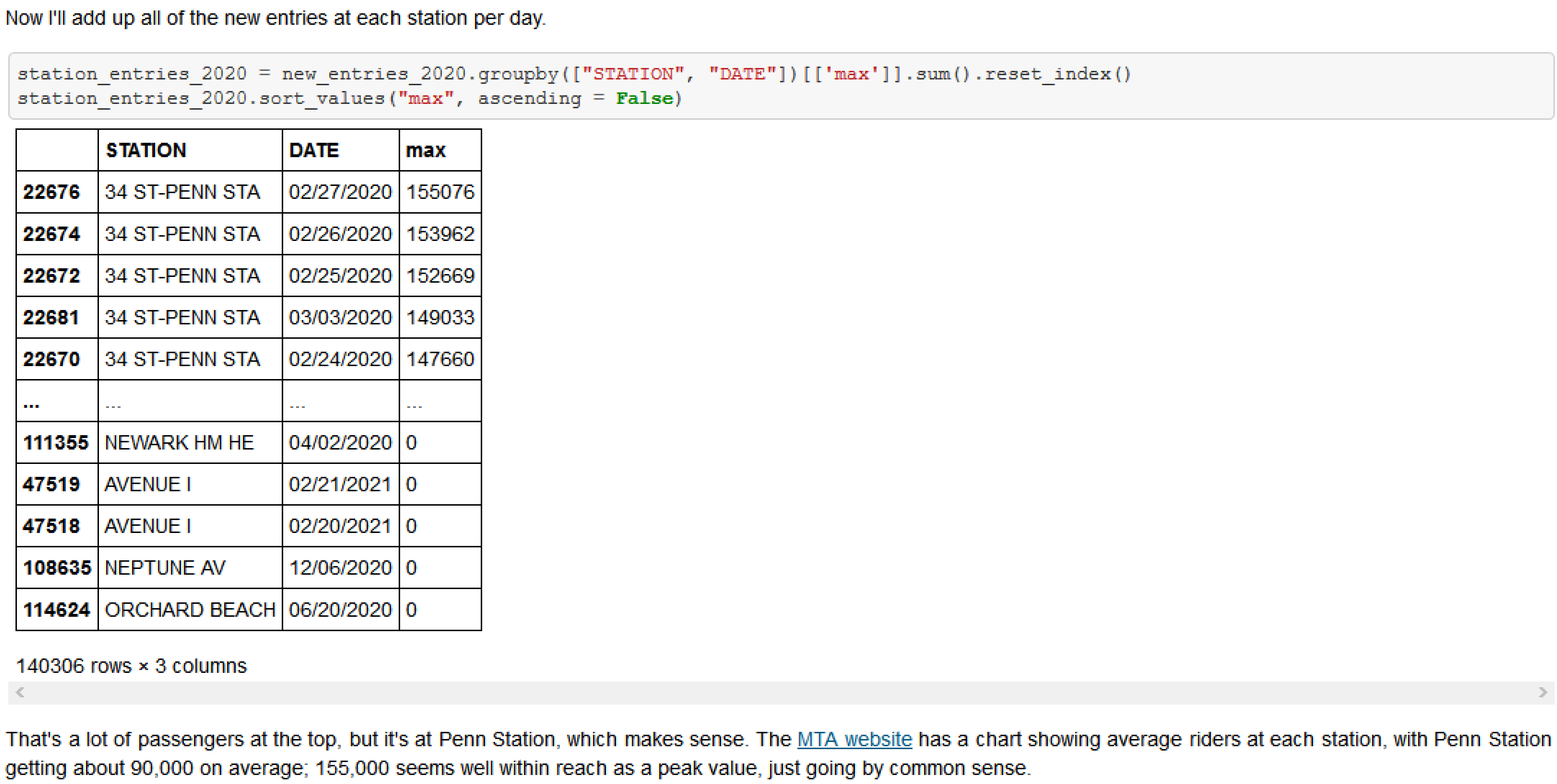
788 rows × 2 columns



```
new_entries_2020.drop(new_entries_2020[new_entries_2020["max"] > 5000].index, inplace = True)
new_entries_2020.sort_values("max", ascending=False).head(30)
```

					min	max
C/A	UNIT	SCP	STATION	DATE		
R238A	R046	02-03-00	GRD CNTRL-42 ST	03/11/2020	NaN	4962
		02-00-02	GRD CNTRL-42 ST	03/11/2020	NaN	4953
N606	R025	00-00-07	JAMAICA CENTER	02/24/2020	NaN	4952
N329	R201	00-00-01	WOODHAVEN BLVD	02/28/2020	NaN	4943
A010	R080	00-00-07	57 ST-7 AV	03/06/2020	NaN	4941
N329	R201	00-00-01	WOODHAVEN BLVD	02/27/2020	NaN	4941
PTH03	R552	00-01-08	JOURNAL SQUARE	09/28/2020	NaN	4938

Choosing the limit to be 5,000 was admittedly arbitrary, but 788 is a miniscule number of rows to lose out of the ~1.8 million of the entire table. A turnstile with 5,000 entries in 24 hours would require someone to go through it about every 17 seconds, which at least seems physically possible.



Now I'll add up all of the new entries at each station per day.

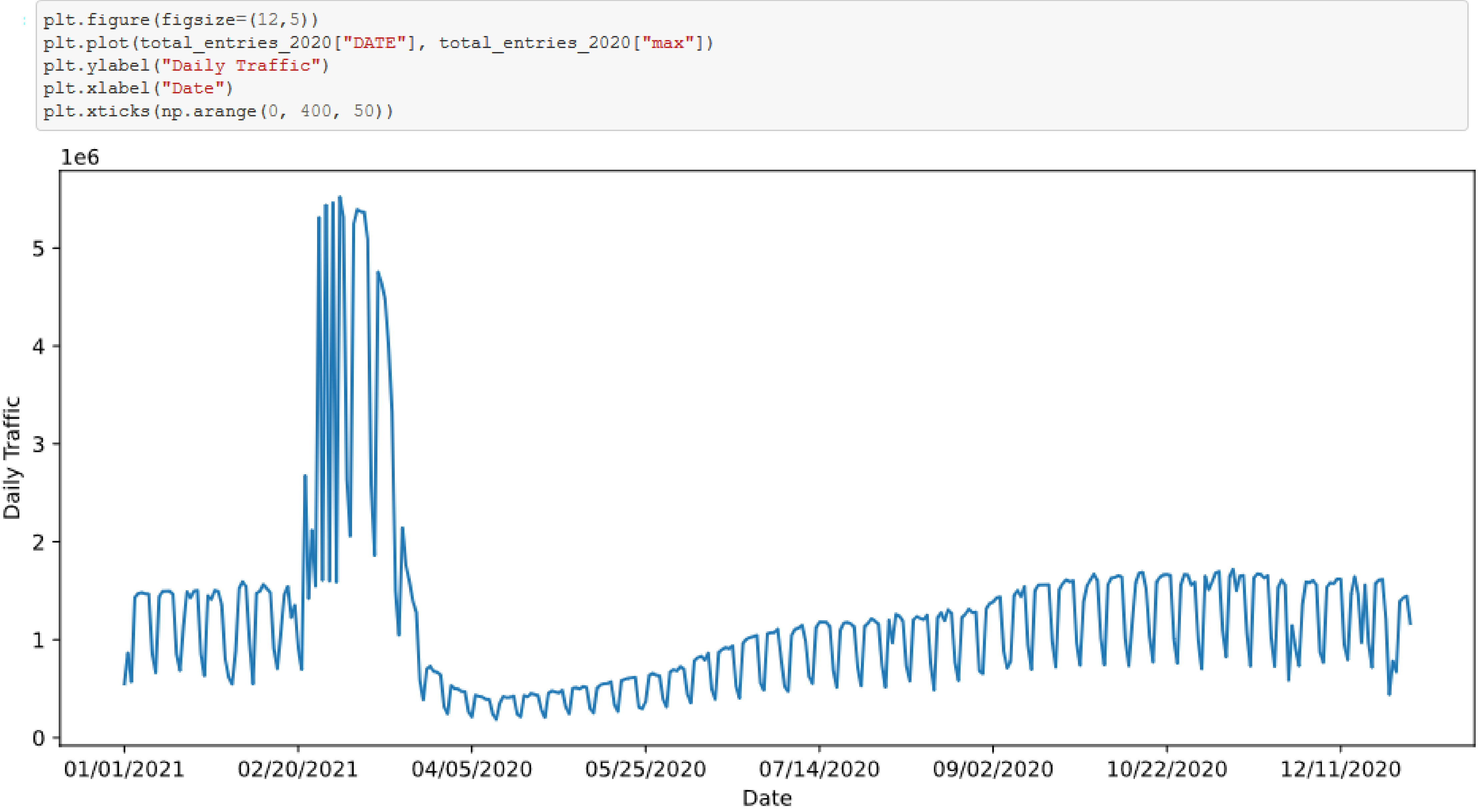
```
station_entries_2020 = new_entries_2020.groupby(["STATION", "DATE"])[['max']].sum().reset_index()
station_entries_2020.sort_values("max", ascending = False)
```

	STATION	DATE	max
22676	34 ST-PENN STA	02/27/2020	155076
22674	34 ST-PENN STA	02/26/2020	153962
22672	34 ST-PENN STA	02/25/2020	152669
22681	34 ST-PENN STA	03/03/2020	149033
22670	34 ST-PENN STA	02/24/2020	147660
...	...	...	...
111355	NEWARK HM HE	04/02/2020	0
47519	AVENUE I	02/21/2021	0
47518	AVENUE I	02/20/2021	0
108635	NEPTUNE AV	12/06/2020	0
114624	ORCHARD BEACH	06/20/2020	0

140306 rows × 3 columns

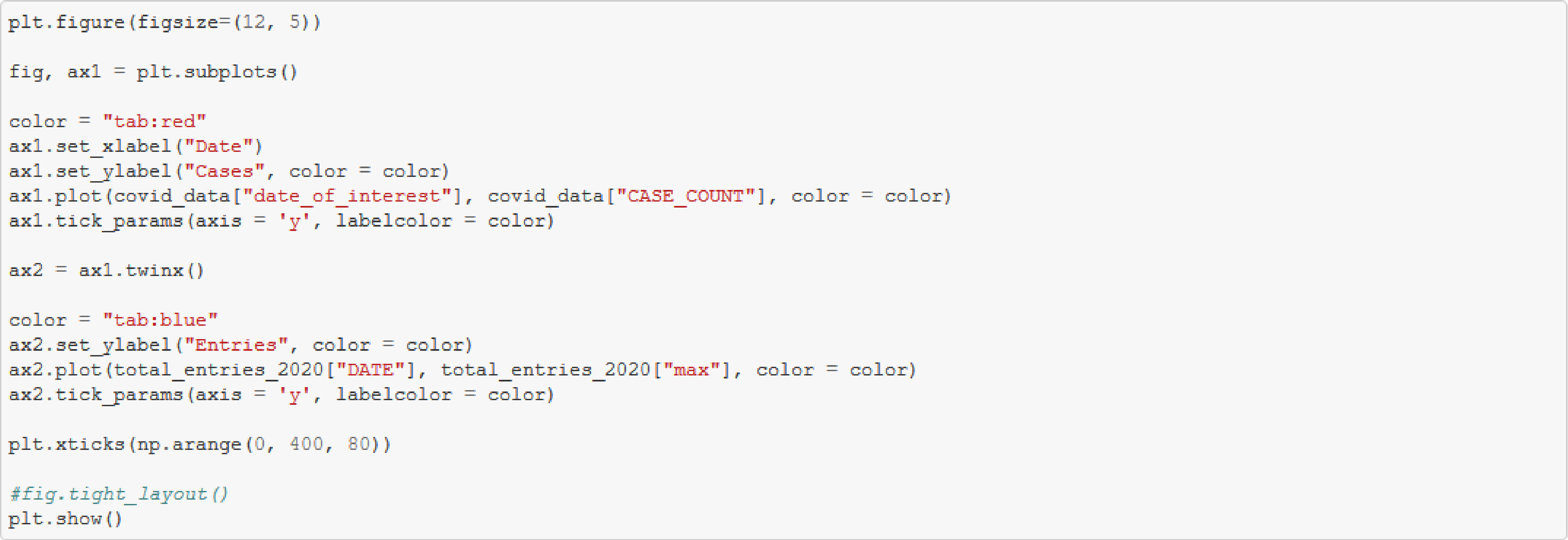
That's a lot of passengers at the top, but it's at Penn Station, which makes sense. The MTA website has a chart showing average riders at each station, with Penn Station getting about 90,000 on average; 155,000 seems well within reach as a peak value, just going by common sense.

http://web.mta.info/nyct/facts/ridership/ridership_sub.htm

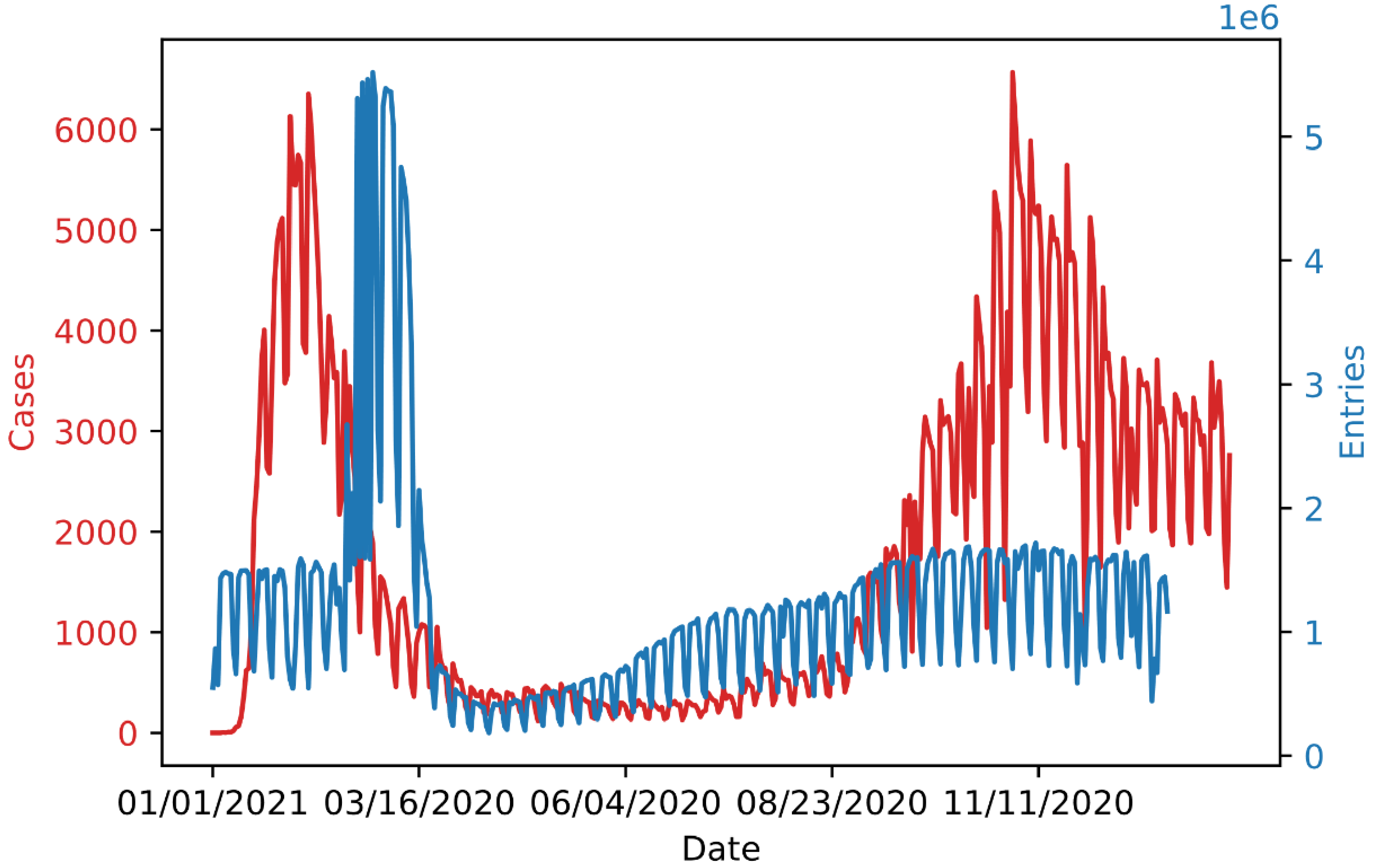


```
plt.figure(figsize=(12,5))
plt.plot(total_entries_2020["DATE"], total_entries_2020["max"])
plt.ylabel("Daily Traffic")
plt.xlabel("Date")
plt.xticks(np.arange(0, 400, 50))
```

Plotting multiple Dataframes on same chart



```
plt.figure(figsize=(12, 5))

fig, ax1 = plt.subplots()

color = "tab:red"
ax1.set_xlabel("Date")
ax1.set_ylabel("Cases", color = color)
ax1.plot(covid_data["date_of_interest"], covid_data["CASE_COUNT"], color = color)
ax1.tick_params(axis = 'y', labelcolor = color)

ax2 = ax1.twinx()

color = "tab:blue"
ax2.set_ylabel("Entries", color = color)
ax2.plot(total_entries_2020["DATE"], total_entries_2020["max"], color = color)
ax2.tick_params(axis = 'y', labelcolor = color)

plt.xticks(np.arange(0, 400, 80))

#fig.tight_layout()
plt.show()
```


1e6
6000
5000
4000
3000
2000
1000
0
Cases
5
4
3
2
1
0
Entries
01/01/2021
03/16/2020
06/04/2020
08/23/2020
11/11/2020
Date

Further Analysis

- Understand/fix spike in data
- Compare MTA usage with previous years
- Analyze usage of stations vs. case rate by borough

Thank You!