Lil’ Zeus Food Truck

METIS PROJECT 1:

MTA ANALYSIS

Introduction

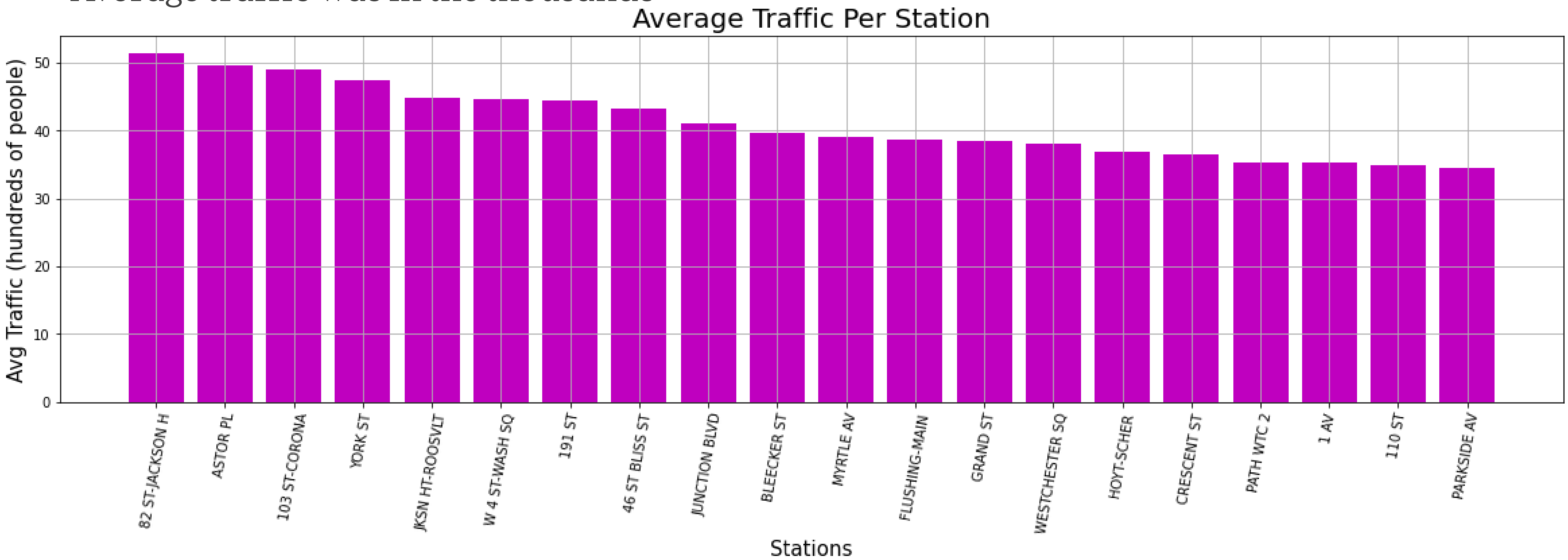
- Motivation was to provide Lil' Zeus with ideal times and locations to sell food
- Objective was to find the busiest stations and the times of the day
- The final result was a couple of graphs displaying this information

Methodology

- Data used were MTA turnstile data for the months of Jan, Feb, Mar of 2022
- Tools used were Jupyter Notebook, Stack Overflow, python libraries of pandas, numpy, matplotlib, datetime, and mpl_toolkits
- Primary metrics: DATE, TIME, STATION, ENTRIES/EXITS (TRAFFIC)
- Isolated turnstiles/stations, averaged the data when grouped, selected the top 20

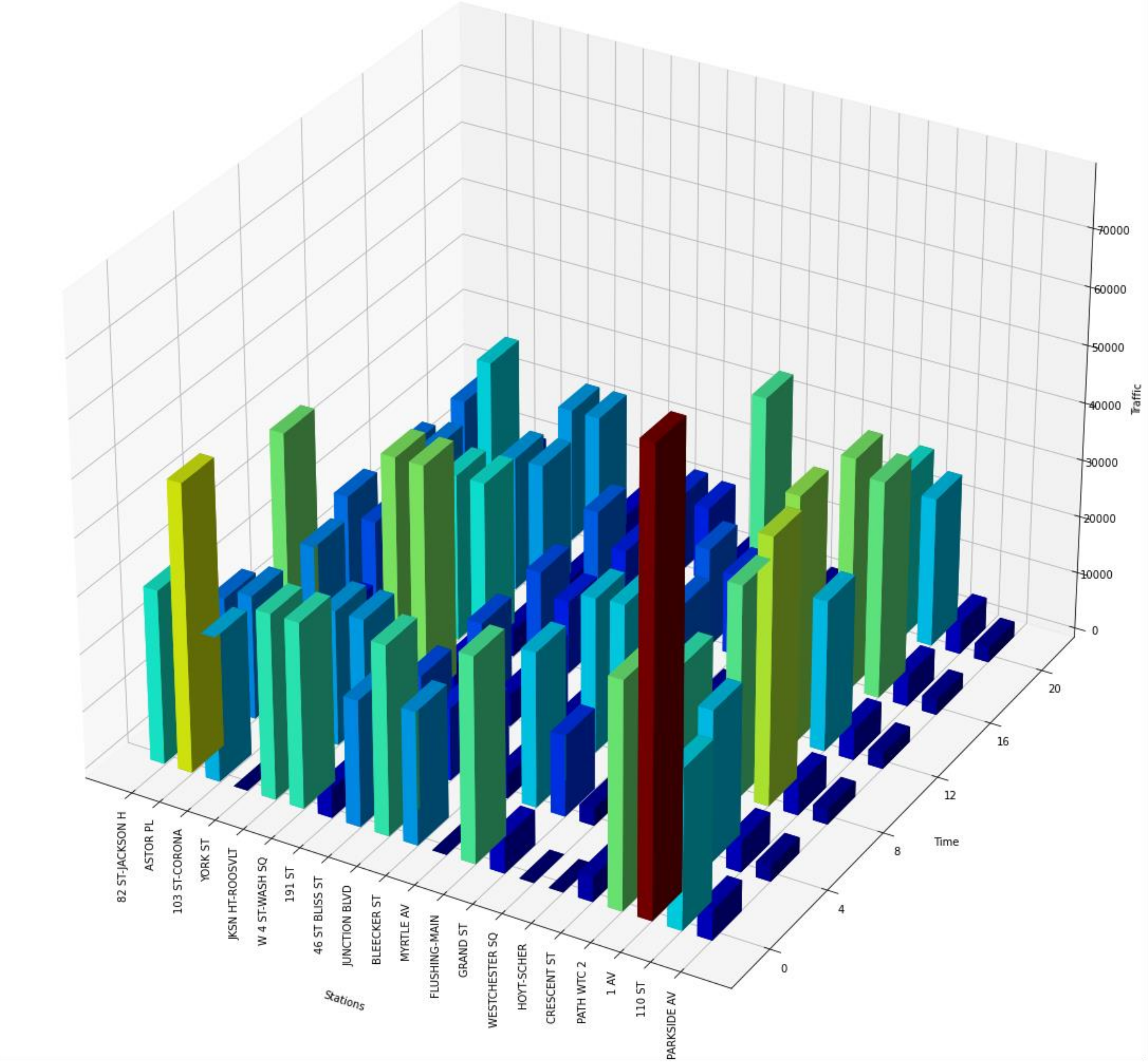
Results

- Average traffic was in the thousands



Results cont.

- Higher traffic at around 8am
- Peaks again 6pm



Conclusion

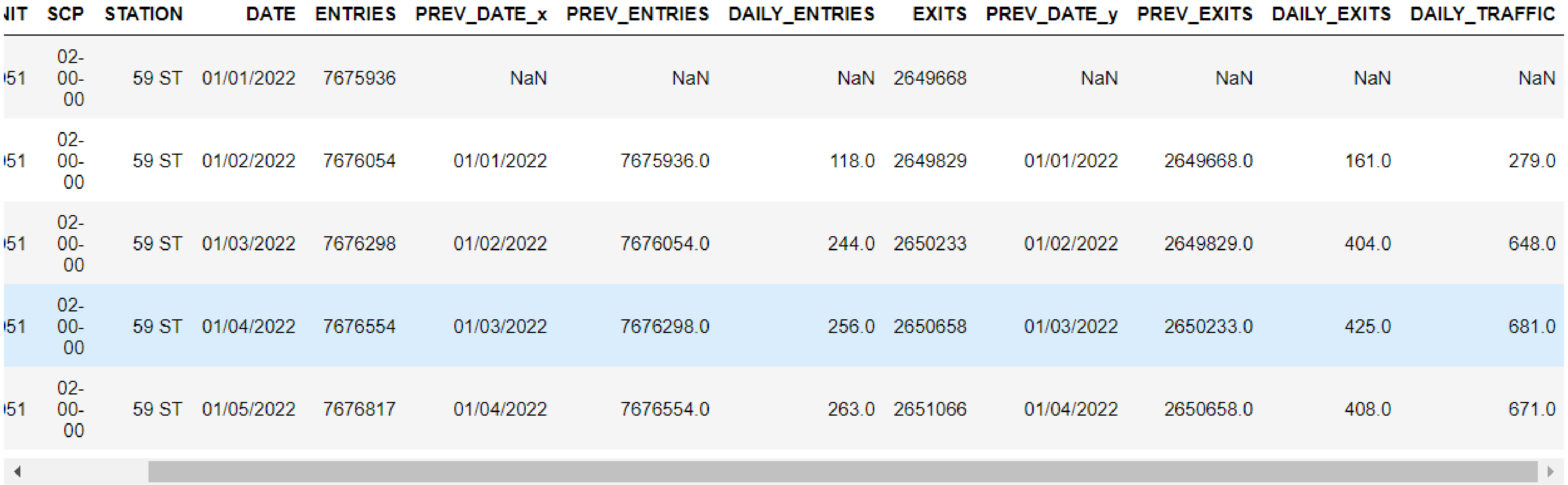
- Recommendations: Clean, Check, Clean, Check
- Insights: Less traffic during Lunch

Future Work

- More rigorous examination of the data to find outliers/anomalies
- Deeper research into 3D plots
- Insights as to what makes these stations so busy
- Investigation into why some data is

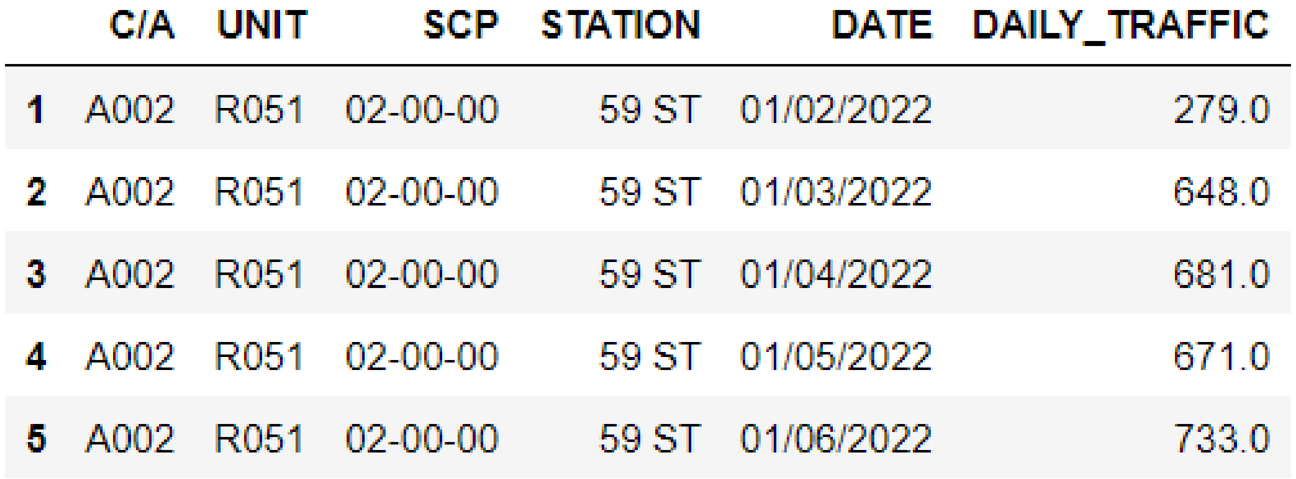
Appendix cont.

- Necessary to create additional columns to get average daily traffic

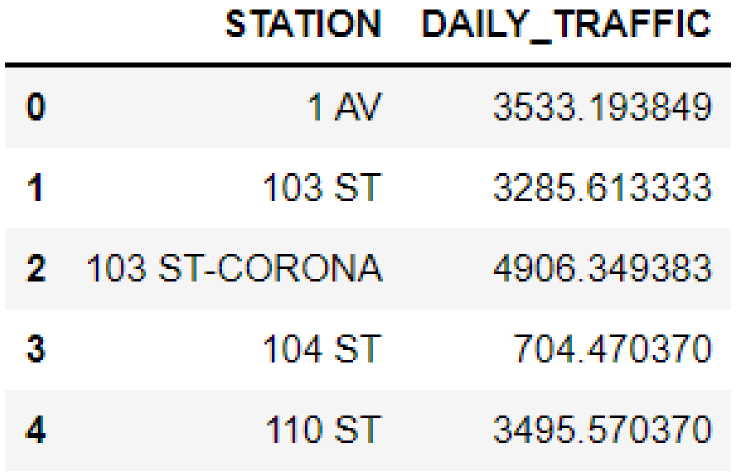


NIT	SCP	STATION	DATE	ENTRIES	PREV_DATE_x	PREV_ENTRIES	DAILY_ENTRIES	EXITS	PREV_DATE_y	PREV_EXITS	DAILY_EXITS	DAILY_TRAFFIC
51	02-00-00	59 ST	01/01/2022	7675936	NaN	NaN	NaN	2649668	NaN	NaN	NaN	NaN
51	02-00-00	59 ST	01/02/2022	7676054	01/01/2022	7675936.0	118.0	2649829	01/01/2022	2649668.0	161.0	279.0
51	02-00-00	59 ST	01/03/2022	7676298	01/02/2022	7676054.0	244.0	2650233	01/02/2022	2649829.0	404.0	648.0
51	02-00-00	59 ST	01/04/2022	7676554	01/03/2022	7676298.0	256.0	2650658	01/03/2022	2650233.0	425.0	681.0
51	02-00-00	59 ST	01/05/2022	7676817	01/04/2022	7676554.0	263.0	2651066	01/04/2022	2650658.0	408.0	671.0

Appendix cont.



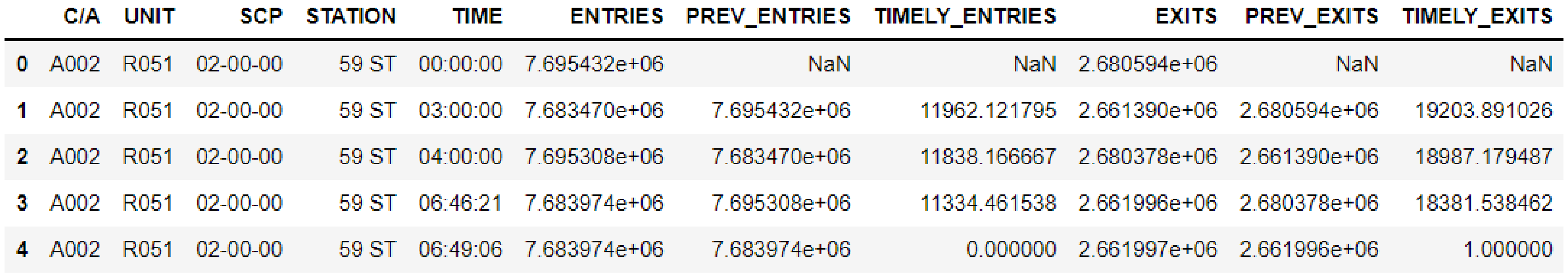
	C/A	UNIT	SCP	STATION	DATE	DAILY_TRAFFIC
1	A002	R051	02-00-00	59 ST	01/02/2022	279.0
2	A002	R051	02-00-00	59 ST	01/03/2022	648.0
3	A002	R051	02-00-00	59 ST	01/04/2022	681.0
4	A002	R051	02-00-00	59 ST	01/05/2022	671.0
5	A002	R051	02-00-00	59 ST	01/06/2022	733.0



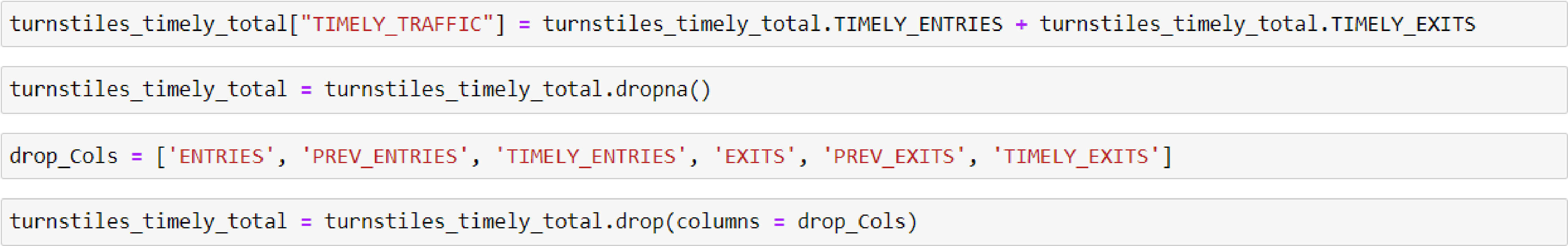
	STATION	DAILY_TRAFFIC
0	1 AV	3533.193849
1	103 ST	3285.613333
2	103 ST-CORONA	4906.349383
3	104 ST	704.470370
4	110 ST	3495.570370

Appendix

- Similar process for timely traffic



	C/A	UNIT	SCP	STATION	TIME	ENTRIES	PREV_ENTRIES	TIMELY_ENTRIES	EXITS	PREV_EXITS	TIMELY_EXITS
0	A002	R051	02-00-00	59 ST	00:00:00	7.695432e+06	NaN	NaN	2.680594e+06	NaN	NaN
1	A002	R051	02-00-00	59 ST	03:00:00	7.683470e+06	7.695432e+06	11962.121795	2.661390e+06	2.680594e+06	19203.891026
2	A002	R051	02-00-00	59 ST	04:00:00	7.695308e+06	7.683470e+06	11838.166667	2.680378e+06	2.661390e+06	18987.179487
3	A002	R051	02-00-00	59 ST	06:46:21	7.683974e+06	7.695308e+06	11334.461538	2.661996e+06	2.680378e+06	18381.538462
4	A002	R051	02-00-00	59 ST	06:49:06	7.683974e+06	7.683974e+06	0.000000	2.661997e+06	2.661996e+06	1.000000



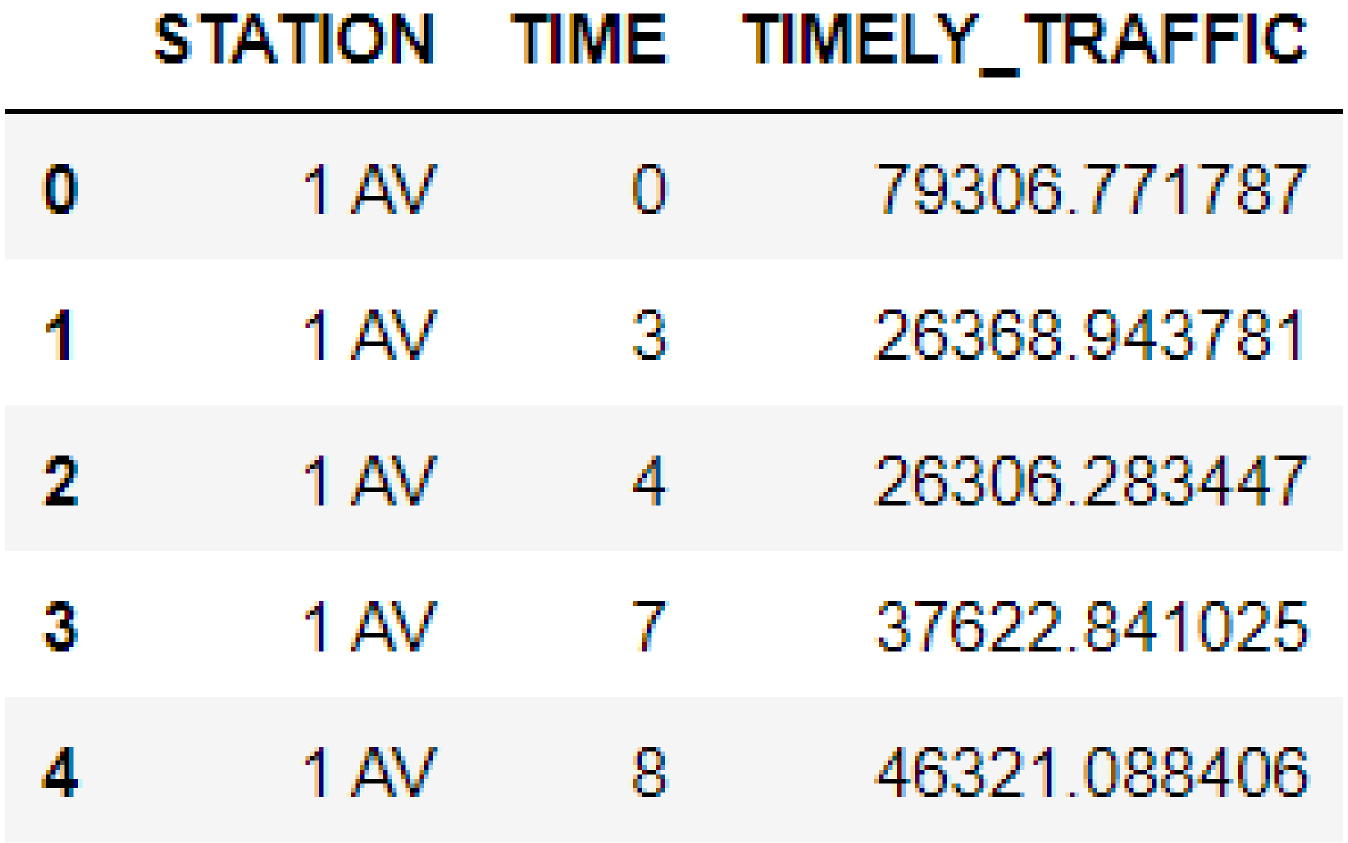
```
turnstiles_timely_total["TIMELY_TRAFFIC"] = turnstiles_timely_total.TIMELY_ENTRIES + turnstiles_timely_total.TIMELY_EXITS
```

```
turnstiles_timely_total = turnstiles_timely_total.dropna()
```

```
drop_Cols = ['ENTRIES', 'PREV_ENTRIES', 'TIMELY_ENTRIES', 'EXITS', 'PREV_EXITS', 'TIMELY_EXITS']
```

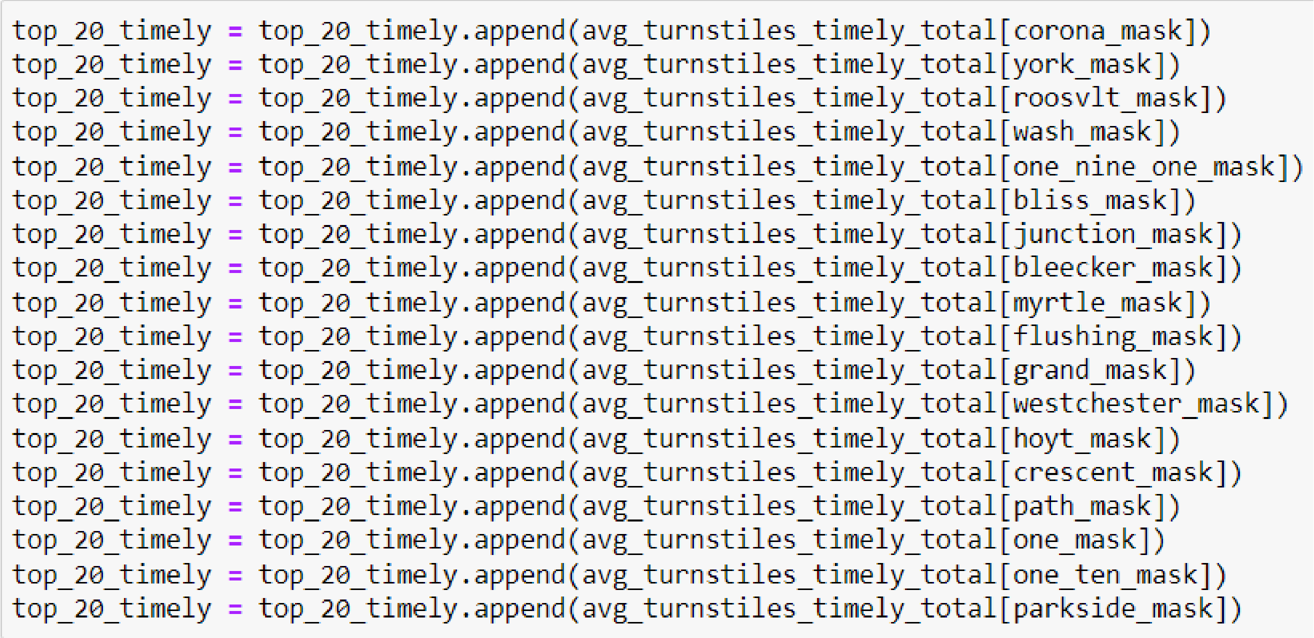
```
turnstiles_timely_total = turnstiles_timely_total.drop(columns = drop_Cols)
```

Appendix cont.

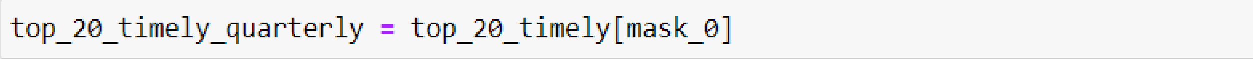


	STATION	TIME	TIMELY_TRAFFIC
0	1 AV	0	79306.771787
1	1 AV	3	26368.943781
2	1 AV	4	26306.283447
3	1 AV	7	37622.841025
4	1 AV	8	46321.088406

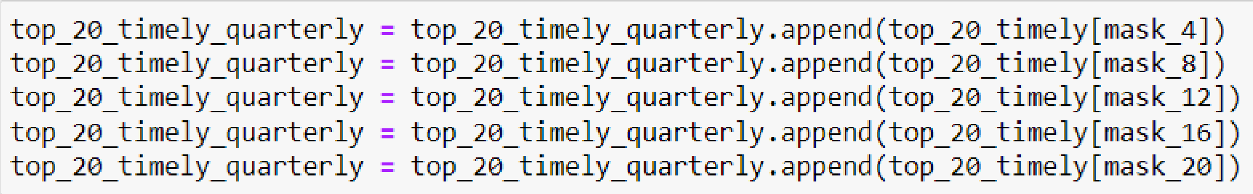
Appendix cont.



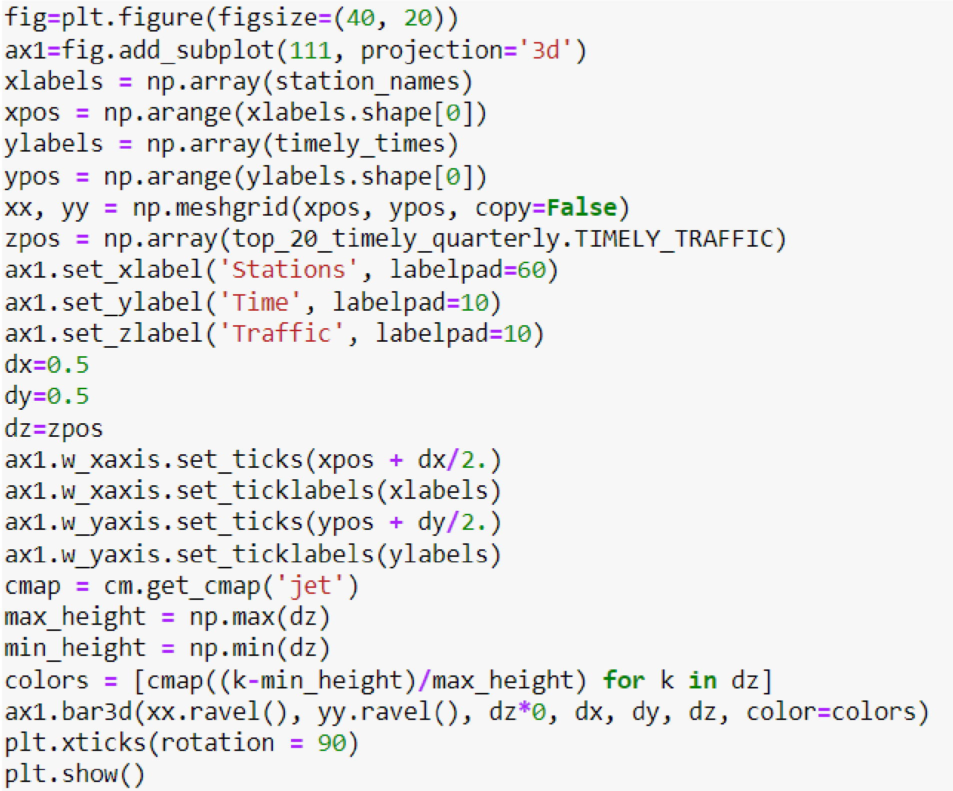
```
top_20_timely = top_20_timely.append(avg_turnstiles_timely_total[corona_mask])
top_20_timely = top_20_timely.append(avg_turnstiles_timely_total[york_mask])
top_20_timely = top_20_timely.append(avg_turnstiles_timely_total[roosvlt_mask])
top_20_timely = top_20_timely.append(avg_turnstiles_timely_total[wash_mask])
top_20_timely = top_20_timely.append(avg_turnstiles_timely_total[one_nine_one_mask])
top_20_timely = top_20_timely.append(avg_turnstiles_timely_total[bliss_mask])
top_20_timely = top_20_timely.append(avg_turnstiles_timely_total[junction_mask])
top_20_timely = top_20_timely.append(avg_turnstiles_timely_total[bleecker_mask])
top_20_timely = top_20_timely.append(avg_turnstiles_timely_total[myrtle_mask])
top_20_timely = top_20_timely.append(avg_turnstiles_timely_total[flushing_mask])
top_20_timely = top_20_timely.append(avg_turnstiles_timely_total[grand_mask])
top_20_timely = top_20_timely.append(avg_turnstiles_timely_total[westchester_mask])
top_20_timely = top_20_timely.append(avg_turnstiles_timely_total[hoyt_mask])
top_20_timely = top_20_timely.append(avg_turnstiles_timely_total[crescent_mask])
top_20_timely = top_20_timely.append(avg_turnstiles_timely_total[path_mask])
top_20_timely = top_20_timely.append(avg_turnstiles_timely_total[one_mask])
top_20_timely = top_20_timely.append(avg_turnstiles_timely_total[one_ten_mask])
top_20_timely = top_20_timely.append(avg_turnstiles_timely_total[parkside_mask])
```



```
top_20_timely_quarterly = top_20_timely[mask_0]
```



```
top_20_timely_quarterly = top_20_timely_quarterly.append(top_20_timely[mask_4])
top_20_timely_quarterly = top_20_timely_quarterly.append(top_20_timely[mask_8])
top_20_timely_quarterly = top_20_timely_quarterly.append(top_20_timely[mask_12])
top_20_timely_quarterly = top_20_timely_quarterly.append(top_20_timely[mask_16])
top_20_timely_quarterly = top_20_timely_quarterly.append(top_20_timely[mask_20])
```



```
fig=plt.figure(figsize=(40, 20))
ax1=fig.add_subplot(111, projection='3d')
xlabels = np.array(station_names)
xpos = np.arange(xlabels.shape[0])
ylabels = np.array(timely_times)
ypos = np.arange(ylabels.shape[0])
xx, yy = np.meshgrid(xpos, ypos, copy=False)
zpos = np.array(top_20_timely_quarterly.TIMELY_TRAFFIC)
ax1.set_xlabel('Stations', labelpad=60)
ax1.set_ylabel('Time', labelpad=10)
ax1.set_zlabel('Traffic', labelpad=10)
dx=0.5
dy=0.5
dz=zpos
ax1.w_xaxis.set_ticks(xpos + dx/2.)
ax1.w_xaxis.set_ticklabels(xlabels)
ax1.w_yaxis.set_ticks(ypos + dy/2.)
ax1.w_yaxis.set_ticklabels(ylabels)
cmap = cm.get_cmap('jet')
max_height = np.max(dz)
min_height = np.min(dz)
colors = [cmap((k-min_height)/max_height) for k in dz]
ax1.bar3d(xx.ravel(), yy.ravel(), dz*0, dx, dy, dz, color=colors)
plt.xticks(rotation = 90)
plt.show()
```