SI 206 Final Project Report

404: Mask Not Found (Adam Sturza, Dane Taylor)

TABLE OF CONTENTS

GOALS

What we achieved &
problems faced

01

INITIAL GOALS

#1

Compare COVID-19 infection data to mask-usage rates, sorted by county and state

#2

Use at least 2 APIs and/or websites

#3

Create at least two visualizations of notable correlations

GOALS ACHIEVED & PROBLEMS

Goals Achieved

- Calculated proportion rates of "always" and "never" mask-wearing relative to a county's total # of cases and deaths
- Reported rates of "always" and "never" mask-wearing relative to counties with the highest # of cases and deaths
- Used two websites to gather data
- Created four visualizations

Problems

- Had trouble figuring out what data to output in .txt file → averaged county data over each state
- Couldn't limit table to 25 items at first → created three separate files on top of main.py that load county, mask use, and state data
- Minor confusion with syntax, API and Pandas documentation → in resources

CALCULATIONS

02

CALCULATIONS

- Data calculated from LoadCounties.py, LoadMasks.py, and LoadStates.py
- Averaged county data from Covid.db over state data to calculate proportion rates and total cases, hospitalizations, & deaths for each state
- (Michigan's rates in case you were interested →)

data.txt

```
---County Statistics Converted to State Data---


Alabama's population, on average, reports never wearing masks 8.2% of the time and always
wearing masks 47.5% of the time. Alabama has reported an accumulated 269877 cases, 26331
hospitalizations, and 3889 deaths.

Alaska's population, on average, reports never wearing masks 5.6% of the time and always
wearing masks 46.2% of the time. Alaska has reported an accumulated 36549 cases, 799
hospitalizations, and 136 deaths.

Arizona's population, on average, reports never wearing masks 5.0% of the time and always
wearing masks 65.5% of the time. Arizona has reported an accumulated 364276 cases, 28248
hospitalizations, and 6950 deaths.

Arkansas's population, on average, reports never wearing masks 9.1% of the time and always
wearing masks 46.3% of the time. Arkansas has reported an accumulated 168228 cases, 9401
hospitalizations, and 2660 deaths.

California's population, on average, reports never wearing masks 3.2% of the time and
always wearing masks 71.5% of the time. California has reported an accumulated 1369751
cases and 19926 deaths (hospitalizations not reported).

Colorado's population, on average, reports never wearing masks 3.9% of the time and always
wearing masks 54.7% of the time. Colorado has reported an accumulated 262061 cases, 14868
hospitalizations, and 3437 deaths.

Connecticut's population, on average, reports never wearing masks 1.8% of the time and
always wearing masks 77.9% of the time. Connecticut has reported an accumulated 126905
```

```
Michigan's population, on average, reports never wearing masks 4.5% of the time and always
wearing masks 57.5% of the time. Michigan has reported an accumulated 424373 cases and
10303 deaths (hospitalizations not reported).
```

CALCULATIONS (CONT'D)

- Data calculated from same files outputted to same data.txt file
- Analyzed notable proportions rates of mask-wearers in specific counties of reporting "always" or "never" wearing mask

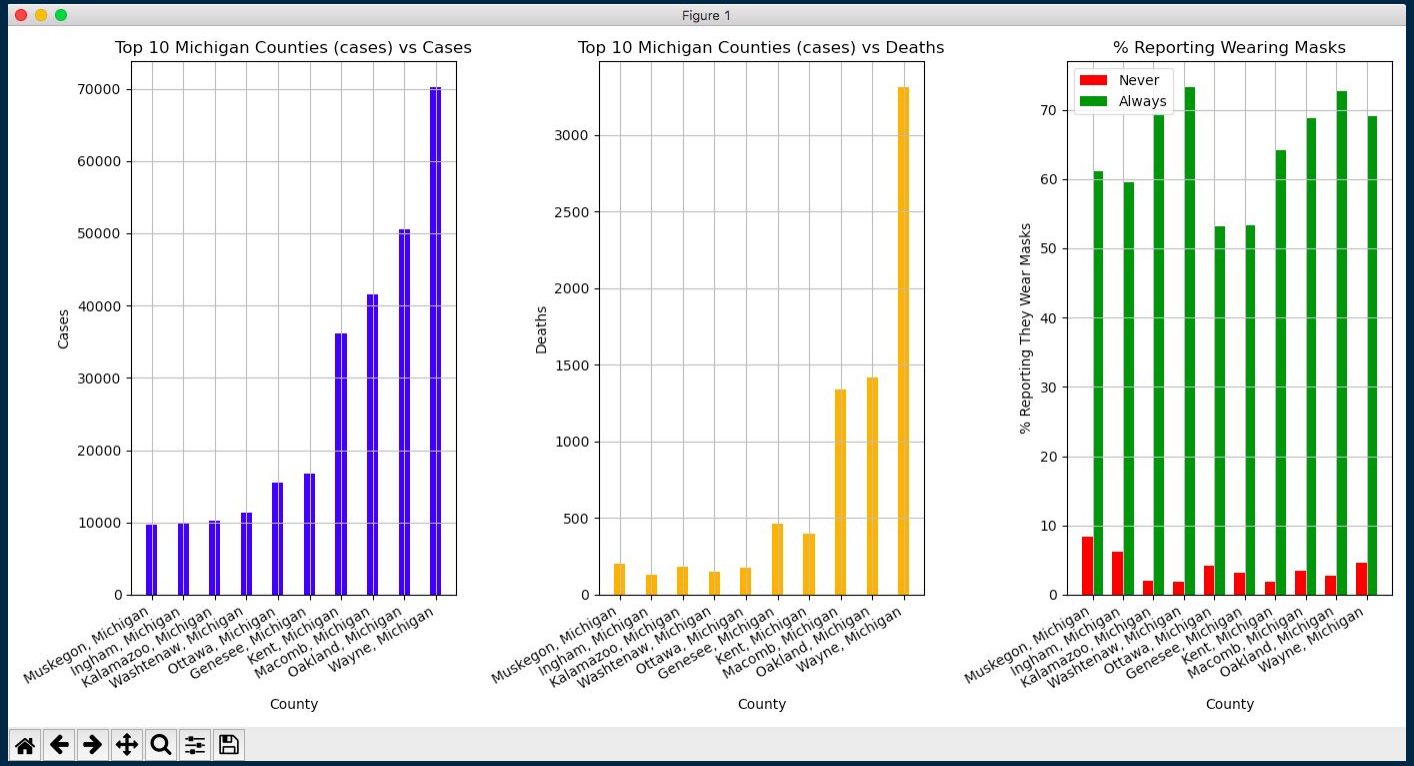
data.txt

```
---Calculated COVID Data Output File :: Here are some of the calculations we did---


Millard, Utah was the county with the highest proportion of its population reporting it
never wore masks at 43.2%. The county has reported 513 cases and 1 deaths.

Inyo, California was the county with the highest proportion of its population reporting it
always wore masks at 88.9%. The county has reported 332 cases and 16 deaths.

Los Angeles, California was the county with the most reported cases at 449982 cases. 78.6%
of the population reported always wearing masks, while 2.1% reported never wearing masks.

Los Angeles, California was the county with the most reported deaths at 7909 deaths. 78.6%
of the population reported always wearing masks, while 2.1% reported never wearking masks.
```

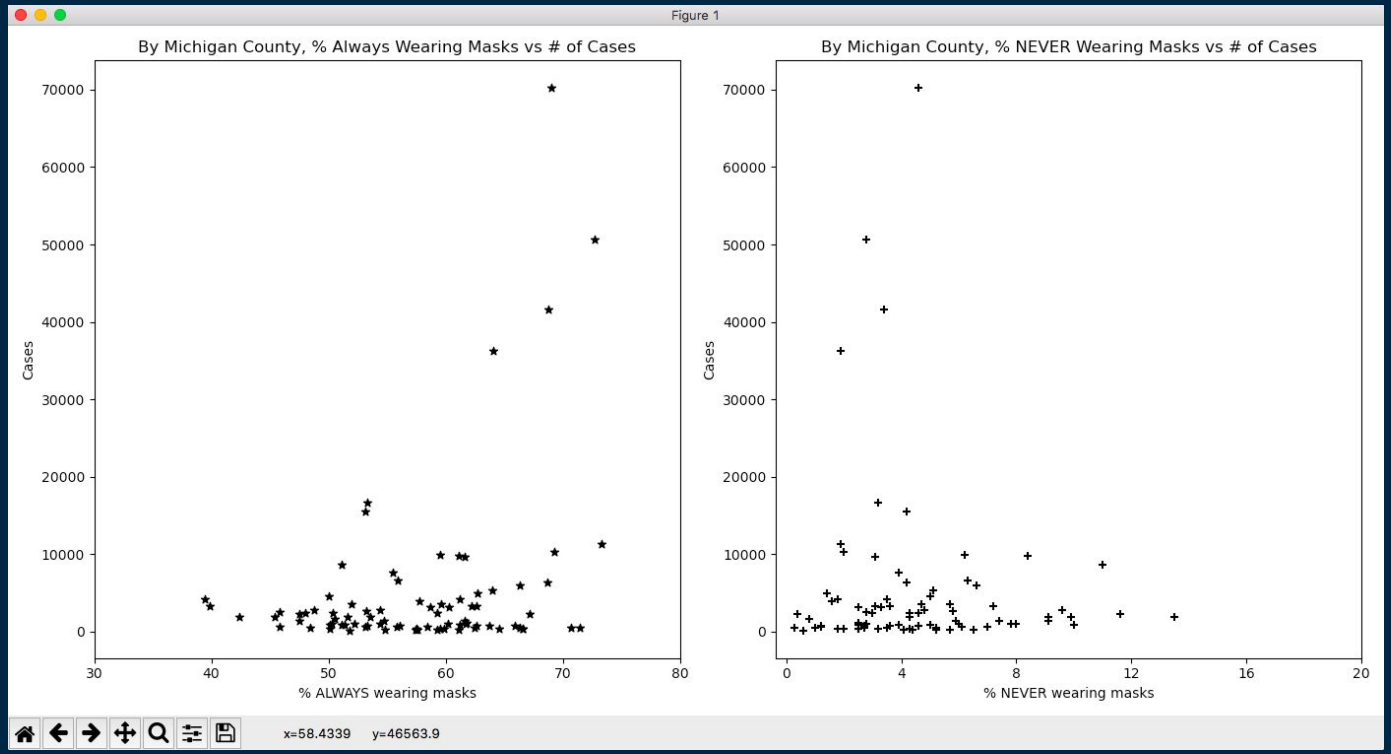
VISUALIZATIONS

03

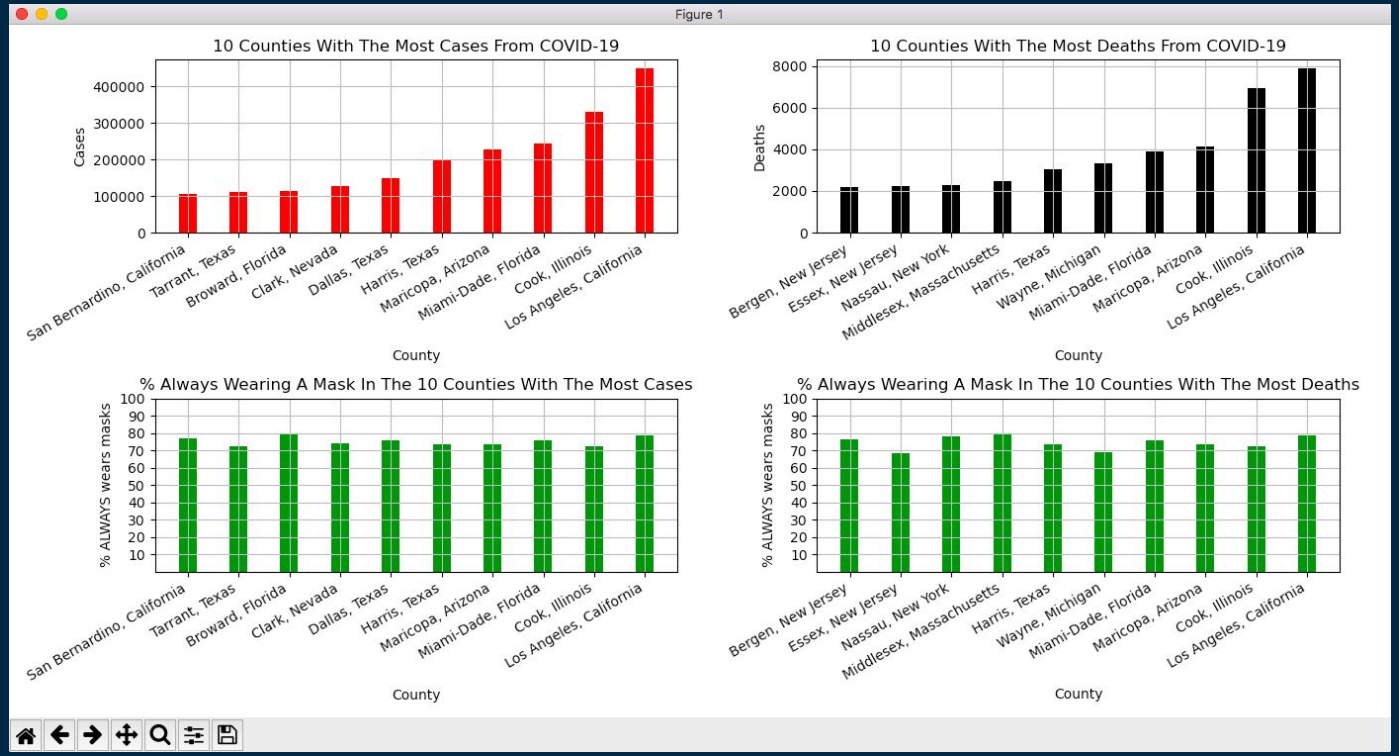
VISUALIZATIONS



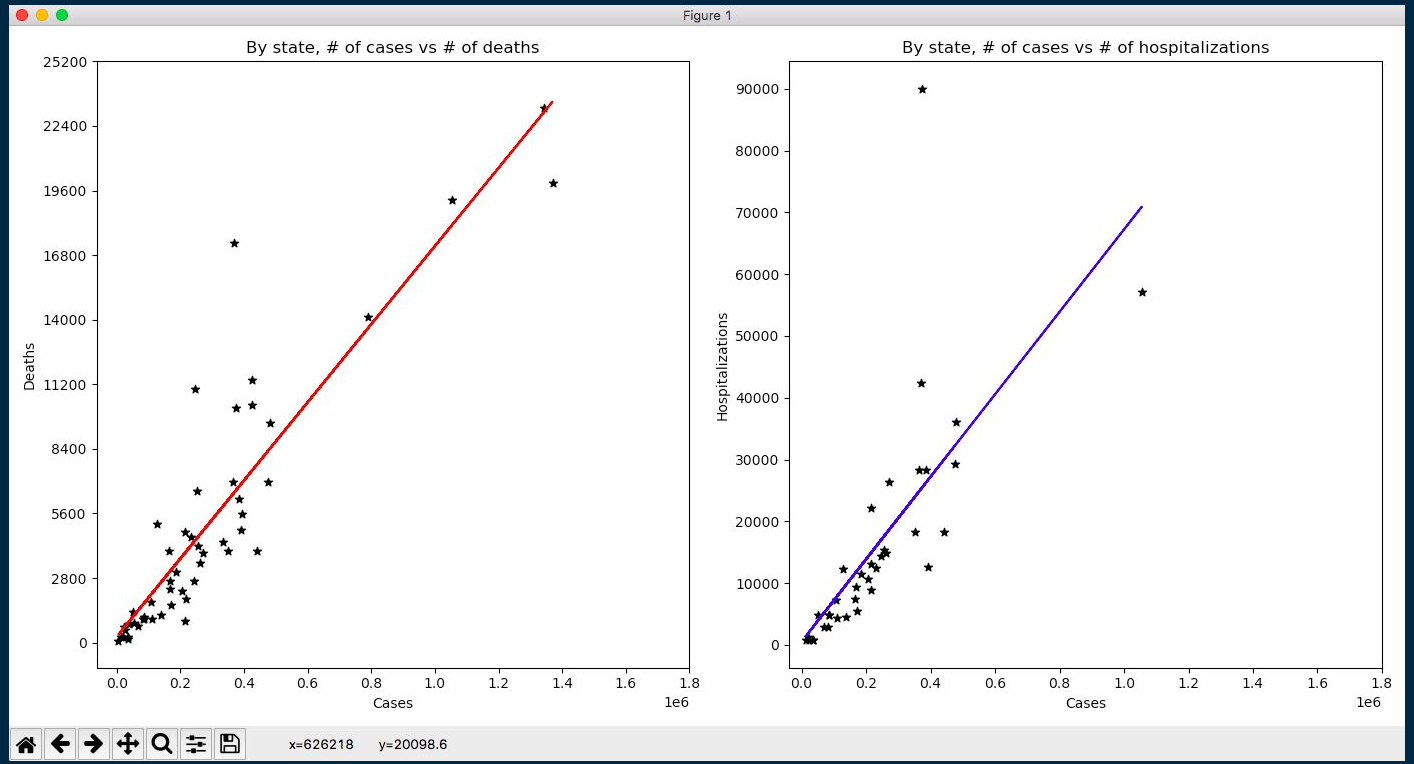
VISUALIZATIONS (CONT'D)



VISUALIZATIONS (CONT'D)



VISUALIZATIONS (CONT'D)



INSTRUCTIONS

04

INSTRUCTIONS

#1

Clear files

Make sure that a "Covid.db" file or any pycache folders **do not** exist in your files. If they do, **delete them**.

#2

Load county data

Open a source-code editor and run LoadCounties.py **five times**. This creates the database table for each county's COVID-19 cases and deaths. After you do this, the screen should say "total number of items - 3245".

#3

Load mask data

Now, run LoadMaskUse.py **five times**. This creates the database table for each county's mask-wearing statistics that fall under "always" and "never". After you do this, the screen should say "total number of items - 3142".

INSTRUCTIONS (CONT'D)

#4

Load state data

Now, run LoadStates.py **one time**. This creates the database table for each state's COVID-19 statistics for cases, deaths, and hospitalizations. After you do this, the screen should say "total number of items - 56".

#5

Run main file

Now, run main.py. This file writes the calculations made to a file called "data.txt" and produces the visualizations made.

#6

Observe results!

There will appear four data visualizations (which utilize matplotlib). In the folder, there will a file called "data.txt" which contains our calculations, and a Covid database (covid.db) which should contain 3 tables in total.

DOCUMENTATION

of code (and tables)

05

LoadCounties.py

```
insertIntoDatabase(df, row, cur, conn):
    """Takes in the name of the Pandas dataframe, the index of the row of data to insert, and the SQLite database cursor and connector as inputs. Loads that row of data from the Pandas dataframe into the 'Counties' SQL table. Does not return anything."""

load25(cur, conn):
    """Takes in the SQLite database cursor and connector as inputs. Establishes the Pandas connection with the website where the raw data is stored, creates the table 'Counties' if it does not already exist, and if there are not already 100 entries in the SQL table, loads 25 entries. Otherwise loads the rest of the entries. Does not return anything."""
```

LoadMaskUse.py

```
insertIntoDatabase(df, row, cur, conn):
    """Takes in the name of the Pandas dataframe, the index of the row of data to insert, and the SQLite database cursor and connector as inputs. Loads that row of data from the Pandas dataframe into the 'Mask_use' SQL table. Does not return anything."""

load25(cur, conn):
    """Takes in the SQLite database cursor and connector. Establishes the Pandas connection with the website where the raw data is stored, creates the table 'Mask_use' if it does not already exist, and if there are not already 100 entries in the SQL table, loads 25 entries. Otherwise loads the rest of the entries. Does not return anything."""
```

LoadStates.py

```
load25(cur, conn):
    """Takes in the SQLite database cursor and connector as inputs.
Establishes the Pandas connection with the website where the raw data is
stored and creates the table 'States' if it does not already exist. A dictionary
maps United States state abbreviations to full names and loads all the data
into the 'States' SQL table. Does not return anything."""
```

main.py

```
setUpDatabase():
    """Creates an SQL database with the name 'Covid', which can be located
at the path main.py is located at. Returns the cursor and connection to the
database."""

calculateWriteData(cur, conn):
    """Performs an SQL join on the 'Counties' and 'Mask_use' tables and
computes various averages as well as finds the counties with the greatest
statistics (cases, deaths, etc.). Additionally, computes counts for each state for
cases and deaths, and average proportions by state of responses to ALWAYS
and NEVER wearing masks. Creates 'data.txt' output file and writes data to it.
Returns list of county data fetched from the SQL join and a dictionary of data
by state."""
```

main.py (CONT'D)

```
visualizations(county_data, stateD):
    """Takes the list of county data and dictionary of state data returned by calculateWriteData() and creates four visualizations, each containing a few subplots of matplotlib graphs, which can be clicked through. Does not return anything."""

main():
    """Runs setUpDatabase(), saves the variables returned from running calculateWriteData(), and runs visualizations() with the same variables as inputs. Finally, closes the database connection."""
```

Table created from LoadCounties.py

- Contains up-to-date data related to COVID-19 **cases** and **deaths** for each county in all states

Table: Counties

	county_id	county_name	county_state	cases	d
	Filter	Filter	Filter	Filter	Fil
1	1001	Autauga	Alabama	3005	
2	1003	Baldwin	Alabama	9728	
3	1005	Barbour	Alabama	1223	
4	1007	Bibb	Alabama	1293	
5	1009	Blount	Alabama	3299	
6	1011	Bullock	Alabama	713	
7	1013	Butler	Alabama	1236	
8	1015	Calhoun	Alabama	7096	
9	1017	Chambers	Alabama	1906	
10	1019	Cherokee	Alabama	1093	
11	1021	Chilton	Alabama	2391	
12	1023	Choctaw	Alabama	440	
13	1025	Clarke	Alabama	1671	
14	1027	Clay	Alabama	963	
15	1029	Cleburne	Alabama	821	
16	1031	Coffee	Alabama	2657	
17	1033	Colbert	Alabama	3536	
18	1035	Conecuh	Alabama	738	
19	1037	Coosa	Alabama	393	
20	1039	Covington	Alabama	2370	
21	1041	Crenshaw	Alabama	745	
22	1043	Cullman	Alabama	5032	
23	1045	Dale	Alabama	2528	
24	1047	Dallas	Alabama	2384	

Table created from LoadMaskUse.py

- Contains data related to mask-wearing for each county in all states based on a 1-5 scale of "**never**" and "**always**" (only populating with those two)
- **county_id** is a shared key for Counties and Mask_use tables

Table: Mask_use

	county_id	response_never	response_always
	Filter	Filter	Filter
1	1001	0.053	0.444
2	1003	0.083	0.436
3	1005	0.067	0.491
4	1007	0.02	0.572
5	1009	0.053	0.459
6	1011	0.031	0.5
7	1013	0.102	0.451
8	1015	0.152	0.442
9	1017	0.117	0.56
10	1019	0.135	0.52
11	1021	0.06	0.618
12	1023	0.049	0.568
13	1025	0.049	0.43
14	1027	0.148	0.329
15	1029	0.151	0.368
16	1031	0.101	0.466
17	1033	0.082	0.51
18	1035	0.099	0.399
19	1037	0.055	0.384
20	1039	0.187	0.356
21	1041	0.06	0.416
22	1043	0.13	0.379
23	1045	0.089	0.46
24	1047	0.095	0.524

Table created from LoadStates.py

- Contains up-to-date data related to COVID-19 **positive cases**, **deaths**, and **hospitalizations** for each state

Table: States

	state	positive	deaths	hospitalized
	Filter	Filter	Filter	Filter
1	Alaska	35720	143	799
2	Alabama	269877	3889	26331
3	Arkansas	170924	2660	9401
4	American Samoa	0	0	NULL
5	Arizona	364276	6950	28248
6	California	1341700	19876	NULL
7	Colorado	260581	2724	14868
8	Connecticut	127715	5146	12257
9	District of Columbia	23136	697	NULL
10	Delaware	39912	793	NULL
11	Florida	1040727	19423	57185
12	Georgia	443822	9806	36039
13	Guam	7004	113	NULL
14	Hawaii	18842	262	1325
15	Iowa	213390	2683	NULL
16	Idaho	109705	1032	4372
17	Illinois	787573	14116	NULL
18	Indiana	381617	6242	28367
19	Kansas	168295	1786	5417
20	Kentucky	200632	2072	10678
21	Louisiana	251123	6584	NULL
22	Massachusetts	256844	11004	14397
23	Maryland	215027	4846	22095
24	Maine	13348	227	768

RESOURCES

06

RESOURCES

Date	Issue Description	Location of Resource	Result (did it solve the issue?)
11/28/20	Used Pandas user guide documentation to better understand how it works with databases, CSVs, and transferring data	https://pandas.pydata.org/docs/user_guide/index.html#user-guide	Yes, we were able to use the Pandas library to manipulate data from CSV raw text into SQL 25 items at a time
11/29/20	Was confused on how to load multiple files into database	https://medium.com/coriers/how-to-load-multiple-files-into-your-database-with-python-and-sql-94e9c417da47	Yes, we created multiple files that fed into the Covid database
12/2/20	Wanted to know how to limit items to 25 using Pandas	Intuitively found a way to check if the initial entries in Pandas were already in the SQL table, and loaded an additional 25 entries if not	Yes, we were able to limit the items to 25 each

RESOURCES (CONT'D)

Date	Issue Description	Location of Resource	Result (did it solve the issue?)
12/3/20	Needed help rounding digits in .txt file where python floats would go on forever	https://www.w3schools.com/python/ref_func_round.asp	Was able to successfully implement the round() function in certain cases when python floats got funky
12/3/20	Used matplotlib usage guide for various tips on using package	https://matplotlib.org/3.3.3/tutorials/introductory/usage.html	Yes, we were able to create numerous visualizations
12/3/20	Was having trouble finding out how to make multiple subplots	https://stackoverflow.com/questions/31726643/how-do-i-get-multiple-subplots-in-matplotlib	Yes, we made multiple subplots on a few of the visualizations

RESOURCES (CONT'D)

Date	Issue Description	Location of Resource	Result (did it solve the issue?)
12/4/20	Wanted to figure out a simple way to plot regression lines on matplotlib	https://numpy.org/doc/stable/reference/generated/numpy.polyfit.html	Utilized numpy.polyfit() with degree of 1 in order to make fitted regression lines on two of the final visualizations
12/3/20	Needed a simplified way to assign digits and axis labels to xticks and yticks in order to enhance appeal of the visualizations	https://numpy.org/doc/stable/reference/generated/numpy.arange.html	Implemented np.arange() and other Numpy helper functions to specify axis digits and labels

The end!

Thank you
https://github.com/taylomd/final-project-fall20