SI 206 Data Oriented Programming Final Project

Project Name: APIs, SQL, and Visualization
Project Team: April Tsai & Morgan Bo
GitHub Repository: https://github.com/aprilts/finalprojectsi206/tree/main

Research Topic

Considering the time frame between January 2020 and April 2021, what are the effects of the COVID-19 pandemic on Tesla and Gamestock average monthly stock prices?

Our Project Goals

1. To be able to successfully grab data from two API's and collect its information and store it into databases that reshape/redefine that data
2. To be able to make visualizations using our calculations and data that provide a visual element to further justify our findings
3. To discover and uncover new pieces of information from a JSON that may be overlooked or is not as noticeable. As a programmer, eager to capitalize on these moments and successfully implement better shown data.

Goals Achieved

1. Accessed two API's and retrieved data from them
2. Successfully manipulated the data to be stored in a Database to find calculations (averages, sums, counts)
3. Able to create a clean line chart(s) that described what we had initially intended to do. In other words, we did not change/modify what our project we had wanted to do.
4. Able to persevere through moments that felt complicated/questionable and kept moving forward; readjusted accordingly with any obstacles that came our way.

Problems Faced

1. While we already had a research topic in mind, finding a suitable API to pull data from was a problem that we faced early on in our project.
2. The next problem we faced was figuring out how to pull the data from the APIs.
3. We struggled with storing 25 or fewer items into the database each time we ran the code.
4. Another problem we encountered was with JOIN. We had trouble understanding how and where to use JOIN to select data from our database.
5. We had minor issues with GitHub, particularly merge problems.
6. We also had problems with MatPlotLib when creating our visualizations.

The resource documentation log at the bottom of the report details how we were able to solve the problems that we faced and the resources that we used when solving our problems.

Calculations

The following is the output found in calculations.txt as a result of our program's calculations. The actual output file can be found in the project zip file.

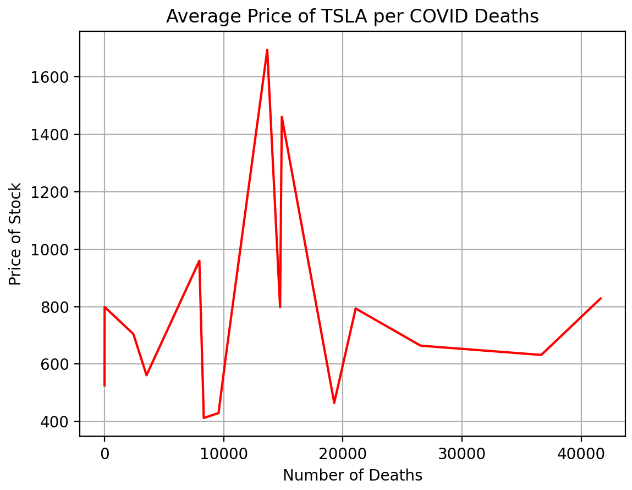
```
Month, Number of Deaths, Stock Name, Average Stock Price
JANUARY 2020, 3, GME, 4.923290476190475
JANUARY 2020, 3, TSLA, 525.5592857142858
FEBRUARY 2020, 11, GME, 3.9658578947368412
FEBRUARY 2020, 11, TSLA, 798.6007894736841
MARCH 2020, 3525, GME, 3.9769340909090913
MARCH 2020, 3525, TSLA, 560.9640909090909
APRIL 2020, 26524, GME, 4.557147619047619
APRIL 2020, 26524, TSLA, 663.8328571428572
MAY 2020, 14730, GME, 4.71288
MAY 2020, 14730, TSLA, 798.0407499999999
JUNE 2020, 7957, GME, 4.58345
JUNE 2020, 7957, TSLA, 959.7679568181816
JULY 2020, 14875, GME, 4.154181818181818
JULY 2020, 14875, TSLA, 1460.8416
AUGUST 2020, 13654, GME, 4.769738095238095
AUGUST 2020, 13654, TSLA, 1694.7205238095235
SEPTEMBER 2020, 8329, GME, 8.485171428571428
SEPTEMBER 2020, 8329, TSLA, 411.9617452380952
OCTOBER 2020, 9565, GME, 12.307438636363635
OCTOBER 2020, 9565, TSLA, 429.02982727272723
NOVEMBER 2020, 19273, GME, 12.476440000000004
NOVEMBER 2020, 19273, TSLA, 464.7093775
DECEMBER 2020, 36644, GME, 16.607077272727274
DECEMBER 2020, 36644, TSLA, 631.6401340909092
JANUARY 2021, 41617, GME, 83.38475000000001
JANUARY 2021, 41617, TSLA, 828.1166236842105
FEBRUARY 2021, 21072, GME, 79.06658157894735
FEBRUARY 2021, 21072, TSLA, 793.240189473684
MARCH 2021, 10273, GME, 189.13425869565216
MARCH 2021, 10273, TSLA, 655.9332913043479
APRIL 2021, 2421, GME, 166.06669642857142
APRIL 2021, 2421, TSLA, 707.4925107142857

NUMBER OF DEATHS PER MONTH (1/2020 - 4/2021) FOR ALL AGES 5 - 74
JANUARY 2020 : 3
FEBRUARY 2020 : 11
MARCH 2020 : 3525
APRIL 2020 : 26524
MAY 2020 : 14730
JUNE 2020 : 7957
JULY 2020 : 14875
```

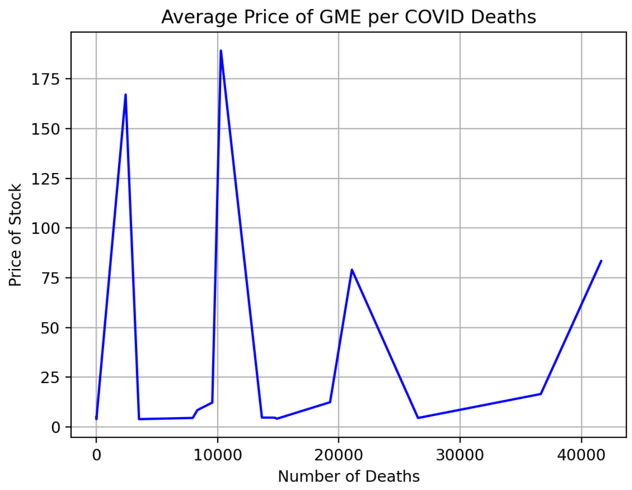
AUGUST 2020 : 13654
SEPTEMBER 2020 : 8329
OCTOBER 2020 : 9565
NOVEMBER 2020 : 19273
DECEMBER 2020 : 36644
JANUARY 2021 : 41617
FEBRUARY 2021 : 21072
MARCH 2021 : 10273
APRIL 2021 : 2421

AVERAGE TESLA STOCK PRICES PER MONTH (1/2020 - 4/2021)
JANUARY 2020 : 525.5592857142858
FEBRUARY 2020 : 798.6007894736841
MARCH 2020 : 560.9640909090909
APRIL 2020 : 663.8328571428572
MAY 2020 : 798.0407499999999
JUNE 2020 : 959.7679568181816
JULY 2020 : 1460.8416
AUGUST 2020 : 1694.7205238095235
SEPTEMBER 2020 : 411.9617452380952
OCTOBER 2020 : 429.02982727272723
NOVEMBER 2020 : 464.7093775
DECEMBER 2020 : 631.6401340909092
JANUARY 2021 : 828.1166236842105
FEBRUARY 2021 : 793.240189473684
MARCH 2021 : 655.9332913043479
APRIL 2021 : 707.4925107142857

AVERAGE GAMESTOP STOCK PRICES PER MONTH (1/2020 - 4/2021)
JANUARY 2020 : 4.923290476190475
FEBRUARY 2020 : 3.9658578947368412
MARCH 2020 : 3.9769340909090913
APRIL 2020 : 4.557147619047619
MAY 2020 : 4.71288
JUNE 2020 : 4.58345
JULY 2020 : 4.154181818181818
AUGUST 2020 : 4.769738095238095
SEPTEMBER 2020 : 8.485171428571428
OCTOBER 2020 : 12.307438636363635
NOVEMBER 2020 : 12.476440000000004
DECEMBER 2020 : 16.607077272727274
JANUARY 2021 : 83.38475000000001
FEBRUARY 2021 : 79.06658157894735
MARCH 2021 : 189.13425869565216
APRIL 2021 : 166.06669642857142

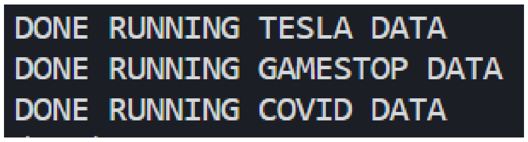
Visualizations





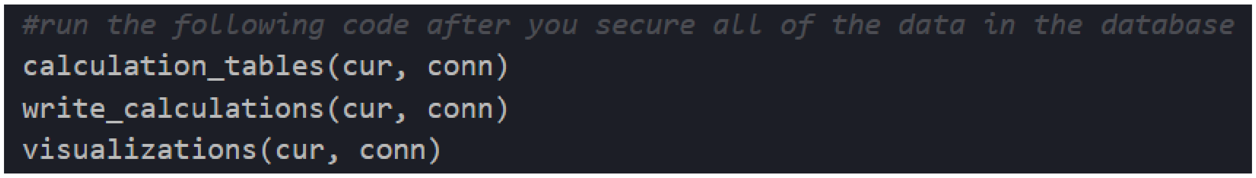
Code Instructions

Continue running the program until you see the following (approximately 14 times). You must see all three messages. This means that all of the data has been inserted into the database.



```
DONE RUNNING TESLA DATA
DONE RUNNING GAMESTOP DATA
DONE RUNNING COVID DATA
```

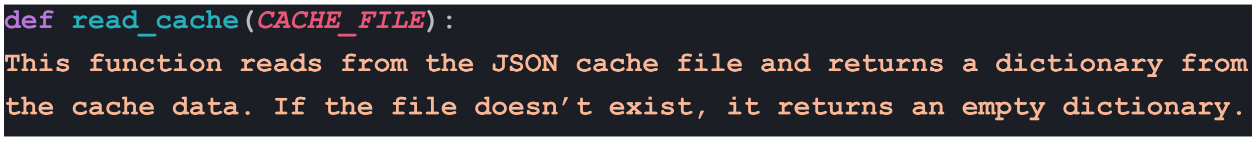
Afterwards, go down to main() to uncomment out three functions (calculation_tables(), write_calculations(), and visualizations()) and run the program one more time.



```
#run the following code after you secure all of the data in the database
calculation_tables(cur, conn)
write_calculations(cur, conn)
visualizations(cur, conn)
```

Note: If the program has issues opening the database, run the program again or delete the database file and re-run the program.

Function Documentation



```
def read_cache(CACHE_FILE):
This function reads from the JSON cache file and returns a dictionary from
the cache data. If the file doesn't exist, it returns an empty dictionary.
```

```
def write_cache(CACHE_FILE, CACHE_DICTIONARY):
This function encodes the cache dictionary (CACHE_DICT) into JSON format
and writes the JSON to the cache file (CACHE_FNAME) to save the search
results.
def create_request_url(symbol):
This function prepares and returns the request url for the API call to
AlphaAdvantage. The documentation of the API parameters is at
https://www.alphavantage.co/documentation/
```

```
def get_stocks_data_from_cache(symbol, CACHE_FILE):
This function uses the passed stock symbol to generate a request_url,
checks if the url is in the dictionary returned by read_cache.

If the request_url exists as a key in the dictionary, it should print
"Using cache for <title>" and return the results for that request_url.
If the request_url does not exist in the dictionary, the function should
print "Fetching data for <title>" and make a call to the API to get data.

If data is found for the stock, it should add them to a dictionary (key is
the request_url, and value is the results)and write out the dictionary to
a file using write_cache.

If there was an exception during the search (for reasons such as no
network connection, etc), it should print out "Exception" and return None.
```

```
def get_covid_data(group_in, state_in, sex_in):
This function takes in a group either ‘By Month’ or ‘By Year’, a selected
U.S. state, and a gender (M/F) and returns a list of dictionaries based on
the selected parameters. This will provide us with the correct type of
COVID details. For example get_covid_data(‘By Month’, ‘United States’,
‘All Sexes’) will return us COVID information on the number of COVID
deaths in all of the United States for all genders per Month.
```

```
def read_data_from_file(filename):
This function opens the passed in file, reads the contents of the file
which is in JSON format, parses the JSON string, and converts the data
into a Python Dictionary. The function then returns the Python Dictionary.
```

```
def setUpDatabase(db_name):
This function sets up the database for SQL, sets up a connection to the
database, and creates a cursor. The function returns both cur and conn.
```

```
def get_stock_dates(symbol, data):
As we are only working with stock data starting from January 2020 to April
2021, the function grabs the keys (in the format of dates) of the
stock_data that we pass in. The function uses regex to find keys that
correspond to the appropriate time frame, and returns the list of keys.
```

```
def set_up_tesla_stocks_table(data, cur, conn, time_frame):
This function takes in cur, conn, stock_data, and the keys from time_frame
which were generated by get_stock_dates(), sets up the Tesla_Stock table
in the database, counts the current number of rows in the database, and
inserts the appropriate data depending on the number of rows in the
database. The function does not return anything, but prints "DONE RUNNING
TESLA DATA" once all of the data has been inserted into the database.
```

```
def set_up_gamestop_stocks_table(data, cur, conn, time_frame):
This function takes in cur, conn, stock_data, and the keys from time_frame
which were generated by get_stock_dates(), sets up the Gamestop_Stock
table in the database, counts the current number of rows in the database,
and inserts the appropriate data depending on the number of rows in the
database. The function does not return anything, but prints "DONE RUNNING
GAMESTOP DATA" once all of the data has been inserted into the database.
```

```
def set_up_time_table(data, cur, conn):
This function takes in cur, conn, and covid_data to create a timetable in
the database which includes the appropriate months and year with a unique
month id associated with each month.
```

```
def get_selected_age_group(data):
Since there are so many groups to choose from, and some age groups
overlap, we wanted to select data from the original JSON list of
dictionaries and make sure we were selecting age groups that did not
overlap. As such, this function takes in covid_data and returns a specific
list of age_groups (5-74) that do not overlap.
```

```
def set_up_age_group_table(data, cur, conn):
This function takes in cur, conn, and covid_data to create the age group
table which includes the appropriate age groups (5-14, 15-24, 25-34,
35-44, 45-54, 55-64, 65-74) and a unique age group id that corresponds to
each group to prevent the data overlapping.
```

```
def set_up_covid_deaths_table(data, cur, conn, age_groups):
This function takes in covid_data, cur, conn, and a list of age_groups
from get_selected_age_group to create a data table called COVID_Deaths
with a month/year ID, age group ID, and the number of deaths corresponding
to those two variables. For every run of our program, we made sure to make
this function grab a new age group from a set of 16 months (1/2020 -
4/2021) on with the number of COVID deaths. That way, we subdivided the
number of deaths by each age group and passed them into our database each
time we ran the program.
```

```
def tesla_monthly_average_price(cur, conn):
This function selects data from the Tesla_Stock table in the database and
calculates the average daily price by adding up the high and low prices
and dividing by two. Afterwards, the function sorts the data based on each
month, and then calculates the average price per month. The function then
returns a list of tuples that includes the month and its average price.
```

```
def gamestop_monthly_average_price(cur, conn):
This function selects data from the Gamestop_Stock table in the database
and calculates average daily price by adding up the high and low prices
and dividing by two. Afterwards, the function sorts the data based on each
month, and then calculates the average price per month. The function then
returns a list of tuples that includes the month and its average price.
```

```
def calculation_tables(cur, conn):
This function creates two new tables in the database - Average_Stock_Price
and Monthly_Deaths. These two tables store data from our calculations
regarding the average stock price of Tesla and Gamestop and the number of
COVID-19 deaths per month. The function takes in SQL cur and conn, but
does not return anything.
```

```
def write_calculations(cur, conn):
This function joins information from the database stored in
calculation_tables() and writes our calculations into a text file. The
function takes in SQL cur and conn, but does not return anything.
```

```
def visualizations(cur, conn):
This function takes in cur and conn to generate two visualizations using
our existing data and calculations. The function does not return anything.
```

Resources Documentation

Date	Issue Description	Resource	Result
April 21st	While I already had the link to my API, I was trying to figure out how to pull the necessary data from the API.	API Tutorial Documentation HW7	To pull the data directly from the API, I referenced instructions from the API Tutorial website, previous code from HW7, and the documentation from my API website. I was able to successfully create a request url to get the data.
April 22nd	I was not sure how to store 25 or fewer items into the database each time I ran the code.	Office Hours Project Partner	My partner and I had a brainstorming session to see if we could come up with a way to only store 25 or fewer items at a time. We thought about counting the current rows in the database and executing if statements depending on the number of rows. My partner then went to office hours to check if this was a feasible way to do it.
April 23rd	I was having trouble understanding how and where to use JOIN to select data from my database.	W3Schools JOIN Project Partner HW8	After discussing potential ways to use JOIN with my partner and referencing W3Schools and HW8, I was able to successfully select data from my database using JOIN.
April 25th	I had accidentally modified files in a github repository that my partner had modified. Which this was in part because I did not "git pull" right away.	Project Partner Office Hours (Uche)	I happened to just delete the entire project off of my local computer and do a git clone of the most recent push to the github from my project partner. This had saved me from all the additional details as to who's branch is who's and what new lines of code, either form me or my partner, would be selected for a push..so I decided to save myself the burden of having to do that.
April 26th	I had already downloaded MatplotLib but for some reason no graphs were appearing on my screen after finish my code that should have executed visualizations	Lecture Slides and Lecture Recordings	I had to redownload MatPlotlib once again. I'm not too sure what the error was, but after redownloaded MatPlotlib, I was able to get my graphs to appear by testing Discussion 12's assignment first, and then my Project second!