RJD Final Project Documentation Report

Group Names: Raina Gebara, Julia Coffman, and Divum Mittal
Github Repository Link: https://github.com/coffma/RJD-FinalProject/tree/main

1. The goals for your project including what APIs/websites you planned to work with and what data you planned to gather

Originally, our group had planned on using health data to see a correlation between weather, the number of people who would consider themselves active, and data about the health status from each state. The goal was for this data to show that there is a correlation between weather and activity, we most certainly know that there is a correlation between activity and health status. Overall, we wanted to test additional factors to see their importance/significance. We were going to access data from Strava, a fitness app, and gather information about what state each person recorded a workout from. In addition, we were going to do similar extraction with the fitbit website. However, after careful consideration this data was too protected to properly access and therefore we could not gather the appropriate data in addition to the quantity of data necessary to complete this project to the fullest extent.

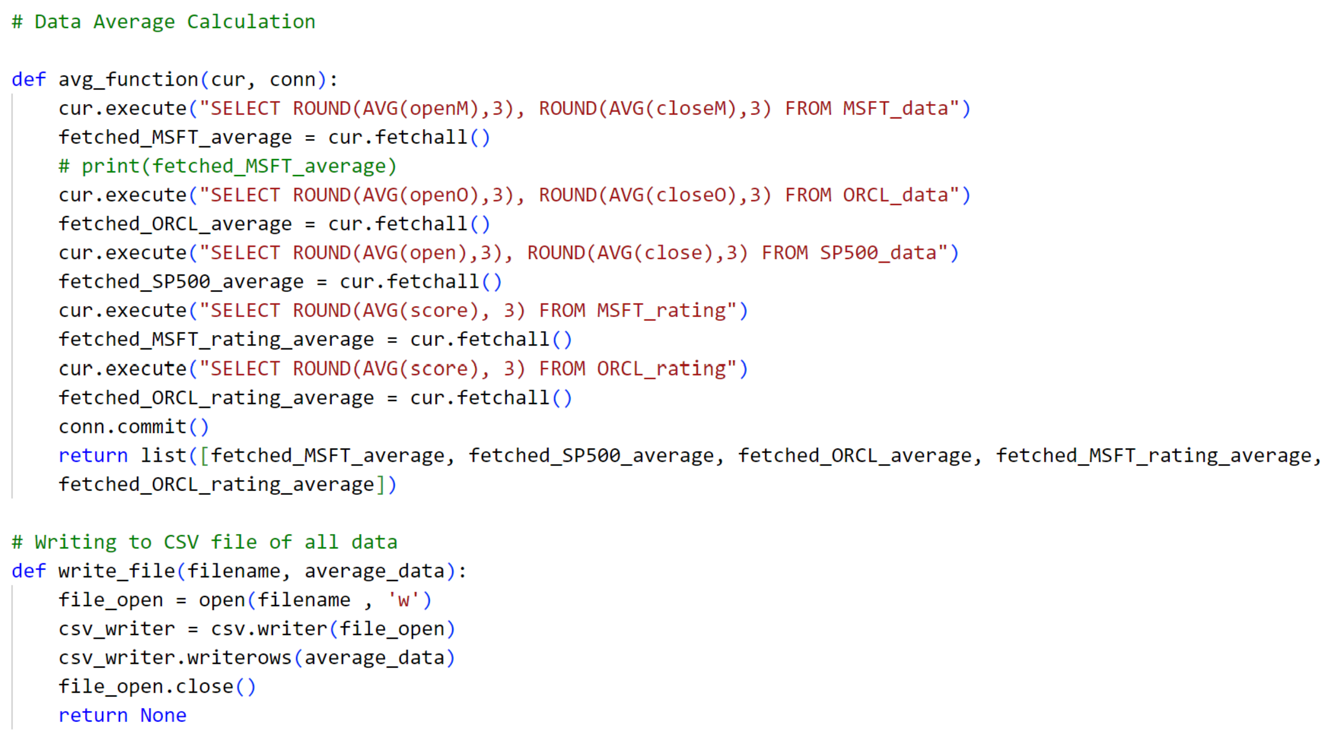
2. The goals that were achieved including what APIs/websites you actually worked with and what data you did gather

We gathered data about different technology/software companies and compared their data to both the S&P 500 and Rating Date. We believed that these resources would give us a clear picture into how these stocks do over the span of the last 100 weeks and the last 100 days. The APIs that we used included alpha advantage, market stack, and financial modeling prep. We collected data about the high, low, open, and close values for Oracle, Microsoft, and the S&P 500. We compared these values to rating data. Rating data is used to enhance the buyers understanding of how the stock is performing. The higher the rating, the better the stock is doing and the more highly recommended it is for the consumer to invest. We then calculated the average for the open and the close of each day. These values were all written to a csv file. We also calculated the average for the rating value of each day for the Oracle and Microsoft data. We did this by joining the tables together on the date column. This calculated data was then written to a csv file. Finally we created 6 visualizations in which they showcased the total of the number of data points per-quarter, the high values for the microsoft stock for each data point, the number of high and mid rating values, the average high for the oracle and microsoft data, and finally the open prices for the S&P 500 over the past 100 days. All of these visualizations helped us to better understand our data.

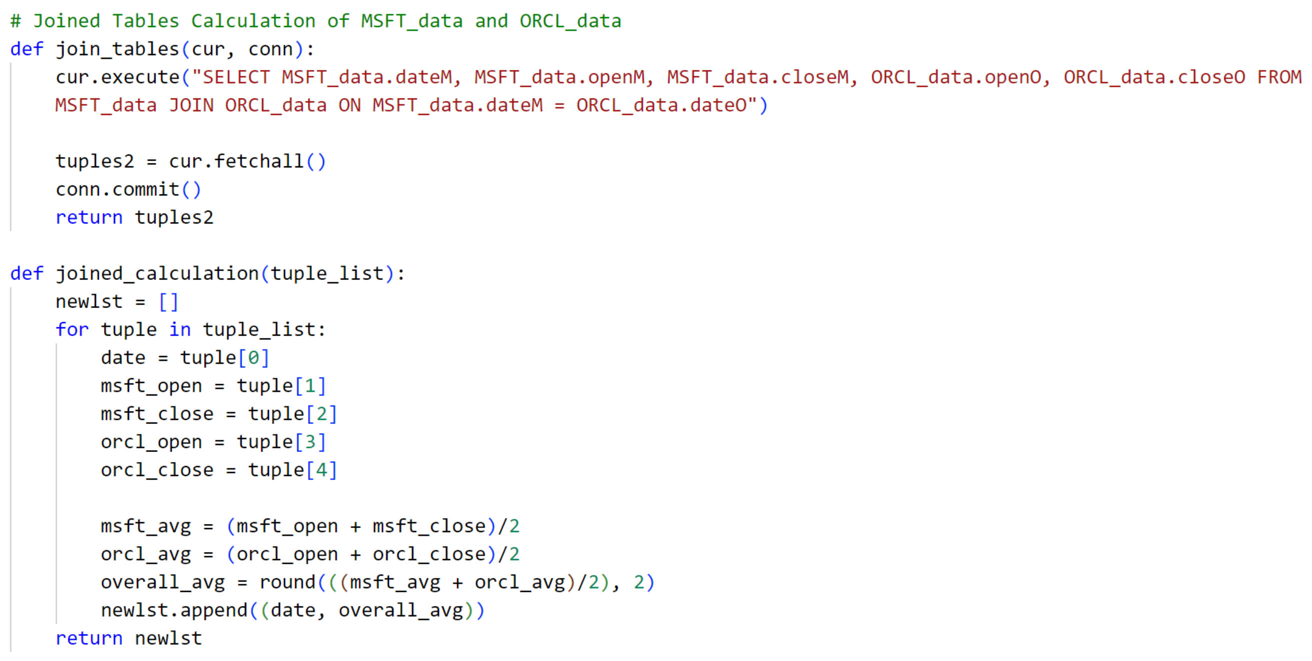
3. The problems you faced

We faced a few problems with getting APIs that gave us access to the data that we needed. As stated before we changed the general topic of our project, this was done because we were unable to access the health data due to not being a developer and ensuring customer privacy meant that API keys were not given out freely. We also had some slight difficulty in writing the join statement. This statement aims to join two tables within a database together on a common column. Additionally, one of our APIs had a limit to the number of calls that could be made which required us to create additional accounts and ensure that the file was only being run when necessary. Another issue we ran into was determining if it was more effective to display the average rating for each stock or the frequency of each rating. We chose frequency to better show the fluctuations the stock may undergo rather than just providing an average investment recommendation.

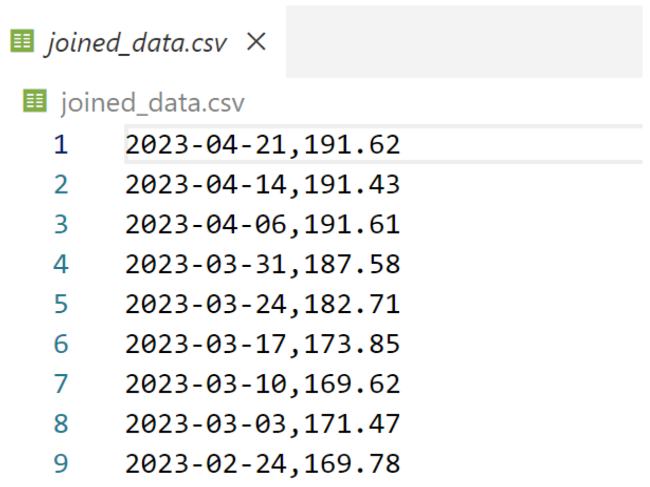
4. The calculations from the data in the database (i.e. a screenshot)



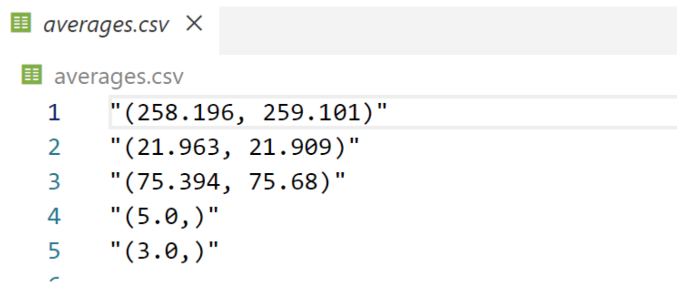
```
# Data Average Calculation

def avg_function(cur, conn):
    cur.execute("SELECT ROUND(AVG(openM),3), ROUND(AVG(closeM),3) FROM MSFT_data")
    fetched_MSFT_average = cur.fetchall()
    # print(fetched_MSFT_average)
    cur.execute("SELECT ROUND(AVG(openO),3), ROUND(AVG(closeO),3) FROM ORCL_data")
    fetched_ORCL_average = cur.fetchall()
    cur.execute("SELECT ROUND(AVG(open),3), ROUND(AVG(close),3) FROM SP500_data")
    fetched_SP500_average = cur.fetchall()
    cur.execute("SELECT ROUND(AVG(score), 3) FROM MSFT_rating")
    fetched_MSFT_rating_average = cur.fetchall()
    cur.execute("SELECT ROUND(AVG(score), 3) FROM ORCL_rating")
    fetched_ORCL_rating_average = cur.fetchall()
    conn.commit()
    return list([fetched_MSFT_average, fetched_SP500_average, fetched_ORCL_average, fetched_MSFT_rating_average,
    fetched_ORCL_rating_average])

# Writing to CSV file of all data
def write_file(filename, average_data):
    file_open = open(filename , 'w')
    csv_writer = csv.writer(file_open)
    csv_writer.writerows(average_data)
    file_open.close()
    return None
```



```python
# Joined Tables Calculation of MSFT_data and ORCL_data
def join_tables(cur, conn):
    cur.execute("SELECT MSFT_data.dateM, MSFT_data.openM, MSFT_data.closeM, ORCL_data.openO, ORCL_data.closeO FROM
    MSFT_data JOIN ORCL_data ON MSFT_data.dateM = ORCL_data.dateO")

    tuples2 = cur.fetchall()
    conn.commit()
    return tuples2

def joined_calculation(tuple_list):
    newlst = []
    for tuple in tuple_list:
        date = tuple[0]
        msft_open = tuple[1]
        msft_close = tuple[2]
        orcl_open = tuple[3]
        orcl_close = tuple[4]

        msft_avg = (msft_open + msft_close)/2
        orcl_avg = (orcl_open + orcl_close)/2
        overall_avg = round(((msft_avg + orcl_avg)/2), 2)
        newlst.append((date, overall_avg))
    return newlst
```



```
2023-04-21,191.62
2023-04-14,191.43
2023-04-06,191.61
2023-03-31,187.58
2023-03-24,182.71
2023-03-17,173.85
2023-03-10,169.62
2023-03-03,171.47
2023-02-24,169.78
```

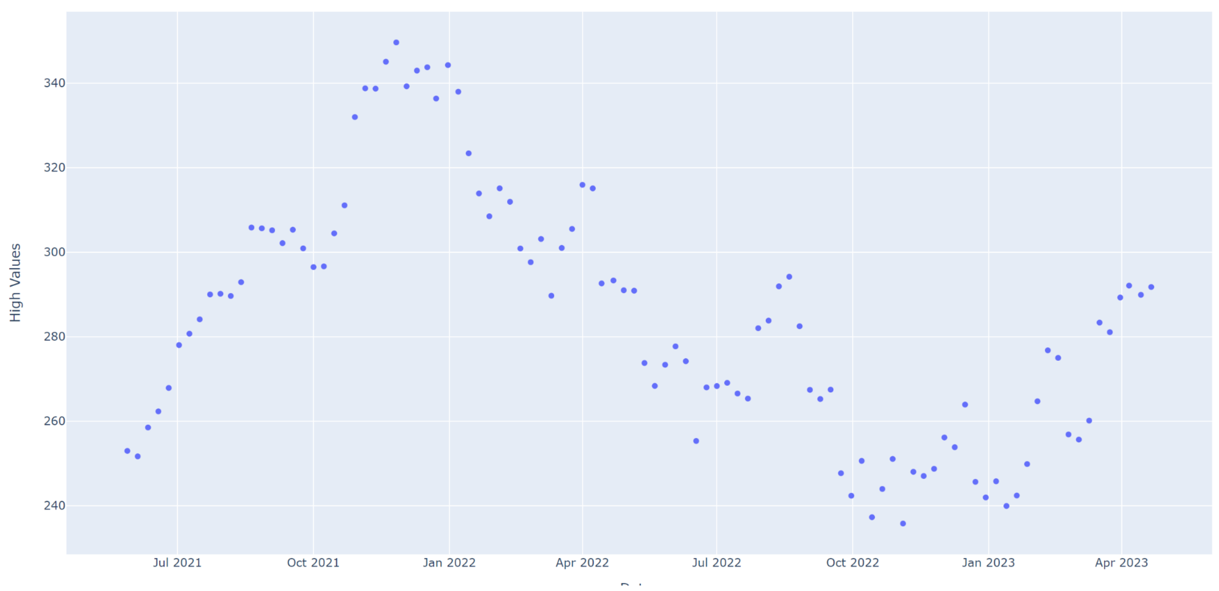


```
"(258.196, 259.101)"
"(21.963, 21.909)"
"(75.394, 75.68)"
"(5.0,)"
"(3.0,)"
```

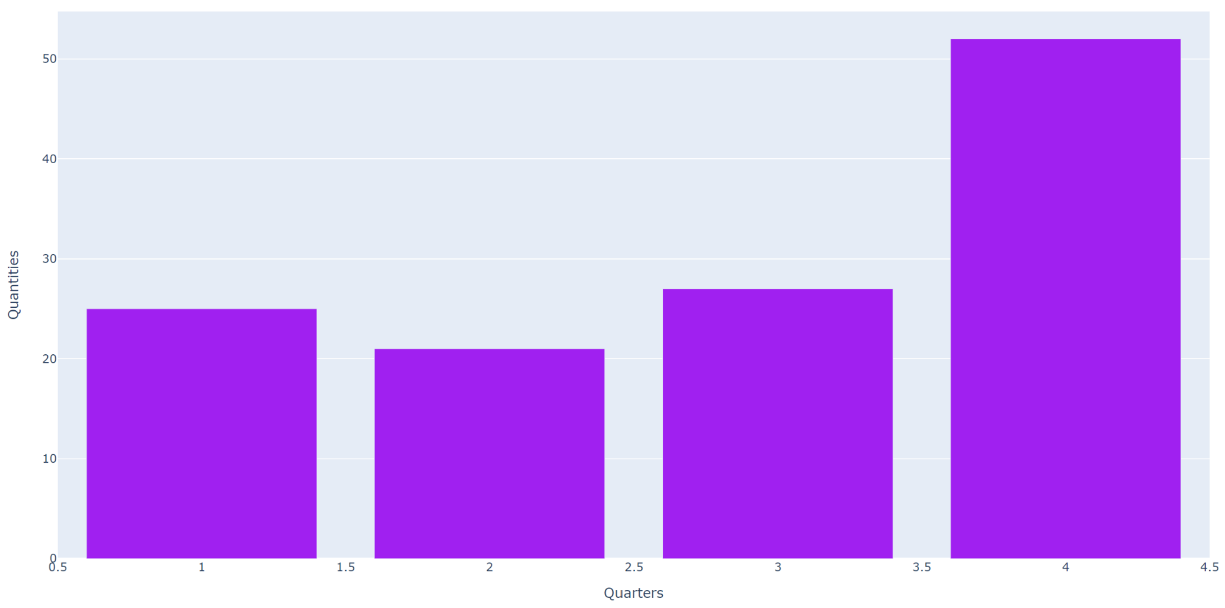
The above are two snippets from the CSV files that we wrote into after running the calculation functions and as a result of running the "Calculations_and_Visualizations.py" file.

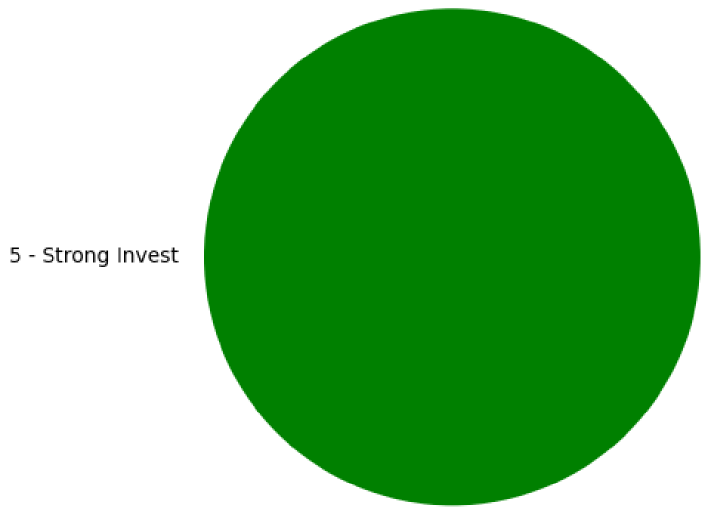
5. The visualization that you created (i.e. screenshots)

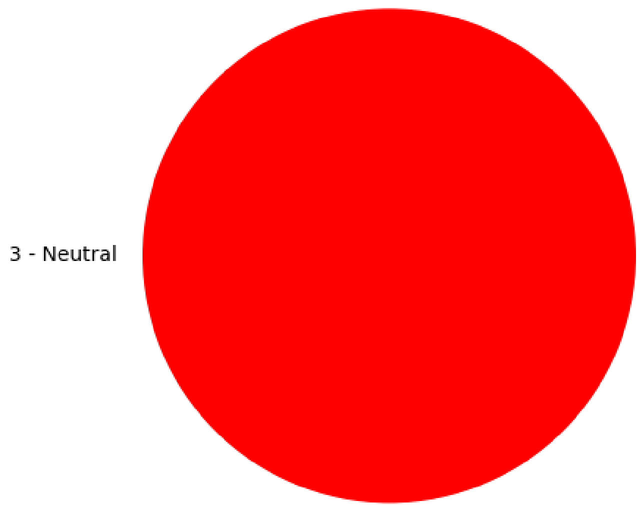
Average High Value For Different Dates

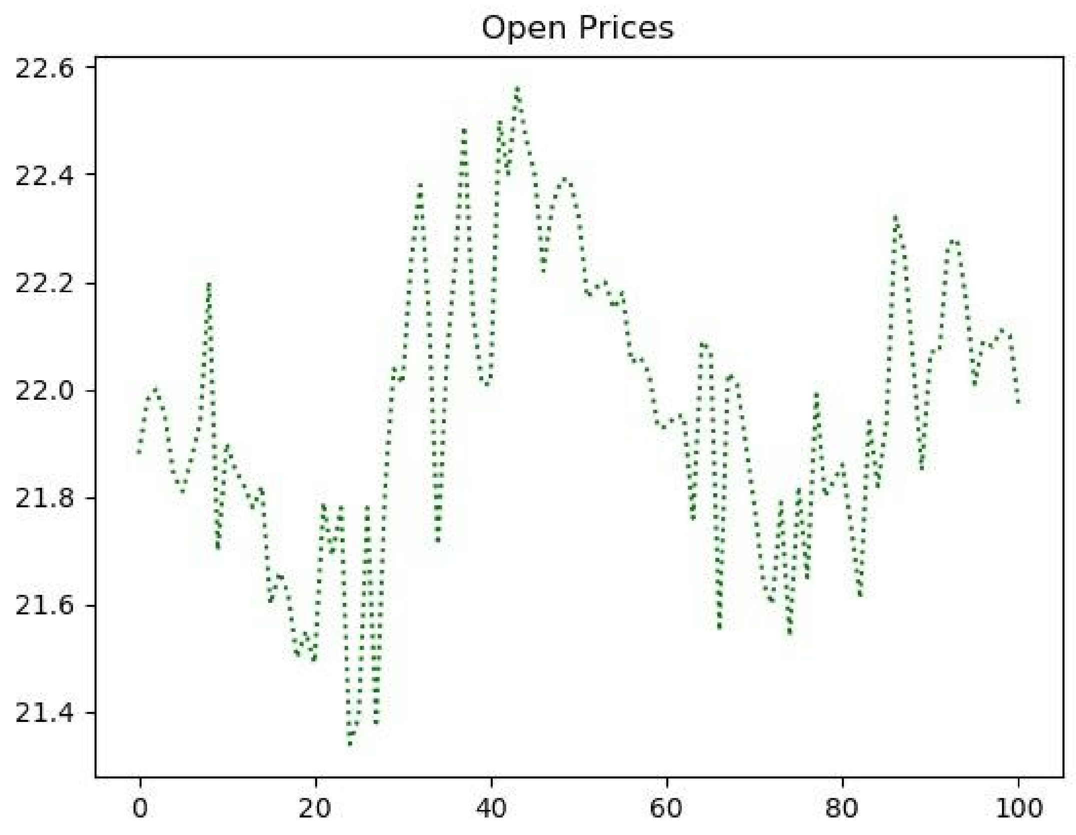


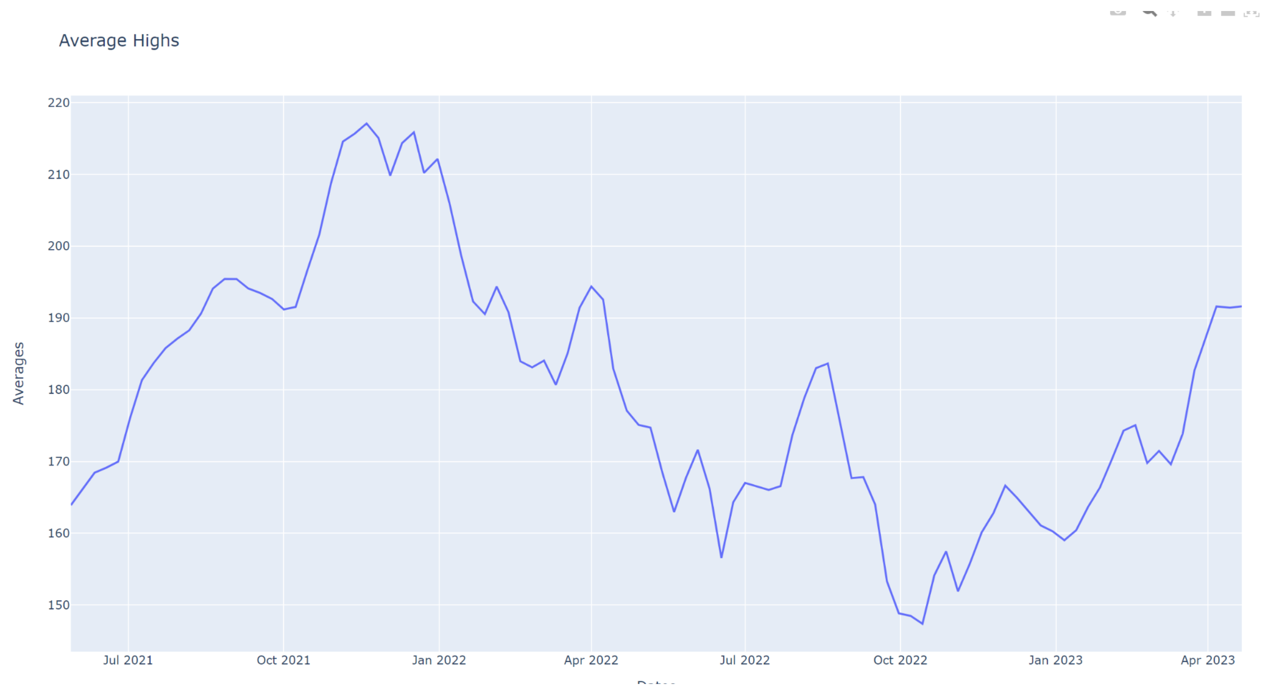
Total Data Points Per Quarter




5 - Strong Invest


3 - Neutral


Open Prices
22.6
22.4
22.2
22.0
21.8
21.6
21.4
0
20
40
60
80
100


Average Highs
Averages
220
210
200
190
180
170
160
150
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Jan 2023
Apr 2023

6. Instructions for running your code

To run our code, you must click run on the file that is entitled “Database_Creation.py”. With each time that you run this code file 25 data points will be added to specific tables within the “final206.db” database. In order to gather 100 data points within the database, one must run the file a total of four times. Next, you will run the “Calculations_and_Visualizations.py” file in order to calculate averages of the data points and create visualizations. Within each of the API data tables, the averages were calculated for Rating, Microsoft, Oracle, and the S&P 500 with those written into CSV files. Additionally, the calculations of overall averages were calculated from a joined table of Microsoft and Oracle data. Two separate csv files will be created with one showcasing the averages in the “averages.csv” and the joined average data points in the “joined_data.csv” for further reference. Lastly, all the visualizations, including a bar chart, two pie charts, a scatterplot, and a linear regression will pop up for the user.

7. Documentation for each function that you wrote. This includes describing the input and output for each function

NAME	**DESCRIPTION**	**INPUT**	**OUTPUT**
make_empty_file	This is simply making the file that is the database	none	Open file name
open_database	Creating the path for the database	Open file name	Cur conn
make_integer_key_table	Creating the table that will be used as a key to denote what quarter the data is from based on the date	Cur conn	none
get_data	This calls the API and loads the data into a json file	Symbol (of the company)	Dictionary of dictionaries containing all of the API data
get_sp500_data	This grabs all of the S&P data	url	A list of dictionaries
get_rating_data	Getting all of the data from the API	company	List of dictionaries

create_table_MSFT	This is creating the table within the database for the microsoft data	Data, cur, conn	none
create_table_ORCL	This is creating another table within the database for all of the Oracle data	Data, cur, conn	None
make_integer_SP500_table	Creating table within database for the S&P data	Data, cur, conn	None
make_oracle_rating_table	Creating table within database for the rating of Oracle	Data, cur, conn	None
make_msft_rating_table	Creating table within database for the rating of microsoft	Data, cur, conn	None
avg_function	Calculates the average of the open and close values from the MSFT data and ORCL data	Cur, conn	None
write_file	Writing the csv file with all of the calculated averages from our data	Filename, average_data	None (new file will be created)
join_tables	Joining the MSFT and ORCL tables on the date	Cur, conn	None
joined_calculation	Joins the average data with the date and creates a list of tuples	List of tuples	List of tuples
joined_plot	Creates the visualization for the average from the joined data	Return from joined_calculation	None, figure appears
final_quarter	Counts the number of	Cur, conn	List of each total

	dates from each quarter		
make_bar_graph	Creates a bar graph for the totals from each quarter	Cur, conn	None, figure will appear
get_high_data	Will get a list of all of the high values from the MSFT data and a list of all of the dates contained in the MSFT data	Cur, conn	Listed highs, listed dates
make_scatter_plot	Creates a scatter plot of the high data over the past 4 years	listed_elements(highs), listed_dates	None, figure will appear
pie_chart	Creates a pie chart for the investments that are a strong and a neutral investment, this is for the MSFT and ORCL data	company	None, figure will appear
sp_scatter	Creating the visualization for the S&P 500 data	Cur conn	Non, figure will appear
main()	Calls all of the functions in the correct order	None	None

8. You must clearly document all resources you used. The documentation should be of the following form

The resources that were used for help in completing this project were going to office hours and meeting with peer instructors through the school of information. In addition we used slides that were presented to us during class and past homeworks that had been completed.

Date	Issue Description	Location of Resource	Result

04/20/23	Select Statements	W3schools	The issue was solved through their insight into properly selecting individual items
04/18/23	Dictionary Handling	W3schools	The issue was solved through reading about the proper use of key values