Jack Basner
Jesse Goodman
Daniel Galper
Link to repository: https://github.com/dgalper/SI-206-Final-Project

1.
The goals for your project (10 points)

Our main goal for this project was to find if there was a pattern between the winning party of the United States presidential election and the way certain stock market indexes, along with several macro economic indicators perform in the year following. We chose to focus on the Dow Jones Industrial Average and the NASDAQ Composite for the indexes. On the other hand, we took a look at unemployment rate, consumer sentiment, exports of goods and services, and federal government expenditure as our economic indicators.

2.
The goals that were achieved (10 points)

We were successfully able to create 6 different visualizations to show how the 2 major stock indexes, the Dow Jones Industrial Average and the NASDAQ Composite, and 4 macro economic indicators, unemployment rate, consumer sentiment, exports of goods and services, and federal government expenditure, subsequently responded in the year directly following a presidential election. When looking at the 2 graphs of the stock indexes and their results the year after the election it is evident that the indexes did much better when the President was of the Democratic party. No Democratic year had under a 10% increase meanwhile 2 of the 3 Republican years were under 2% or negative in both indexes. Additionally, from observing the visualizations we created, it is also clear that the economic indicators were better in years following a Democrat winning the presidency. All of the data we found using our 2 APIs and the website we scraped were able to come together to produce the graphs we wanted to end up with.

3.
The problems that you faced (10 points)

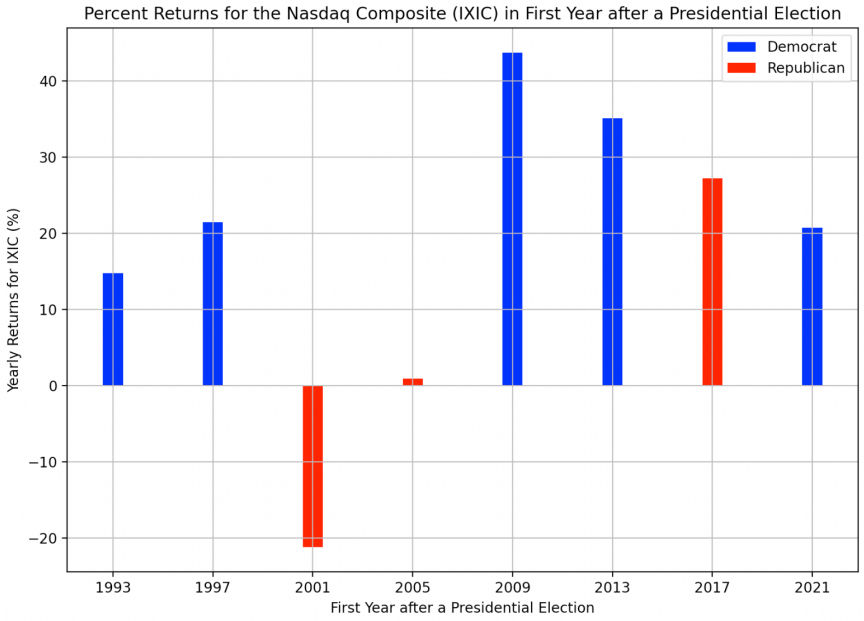
One of the first problems we faced was collecting the information we needed specifically from the election data. Originally, we planned on using 3 API's for our project but after looking online for an API for the election we were unable to find one so we shifted to web scraping. We were able to easily find an API for the stock market and economic factors, but even finding a website to scrape for the election proved difficult. The first 2 websites we found, although they contained

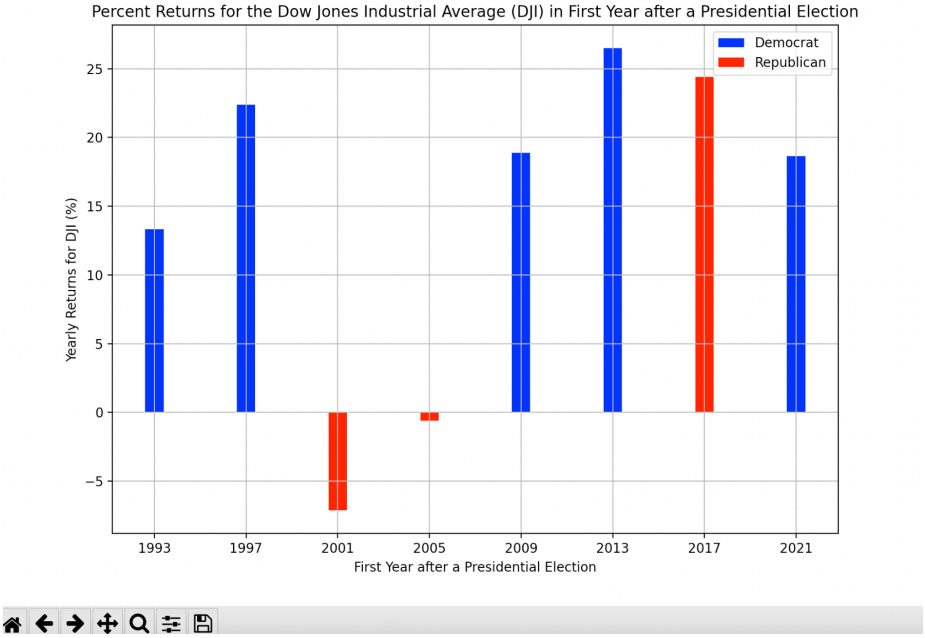
the information we needed, we were not able to scrape because the data was stored in csv files that were accessible on the site, rather than embedded in the pages html. During this, we briefly switched our focus to data surrounding the Corona Virus Pandemic but again, were unable to find a website or API that we could use. The only website that had the COVID data we needed stored their data dynamically, which meant it was unable to be scraped using beautiful soup. We went back to our original plan of using election data and were fortunately able to find a website with scrapable data and we were able to move on with our calculations. The last issue we had was trying to figure out how to select data from tables in order to create calculations and finding out how to store the information into a python variable. We were able to figure it out by using the "fetchall" function thus solving all our problems for this project.

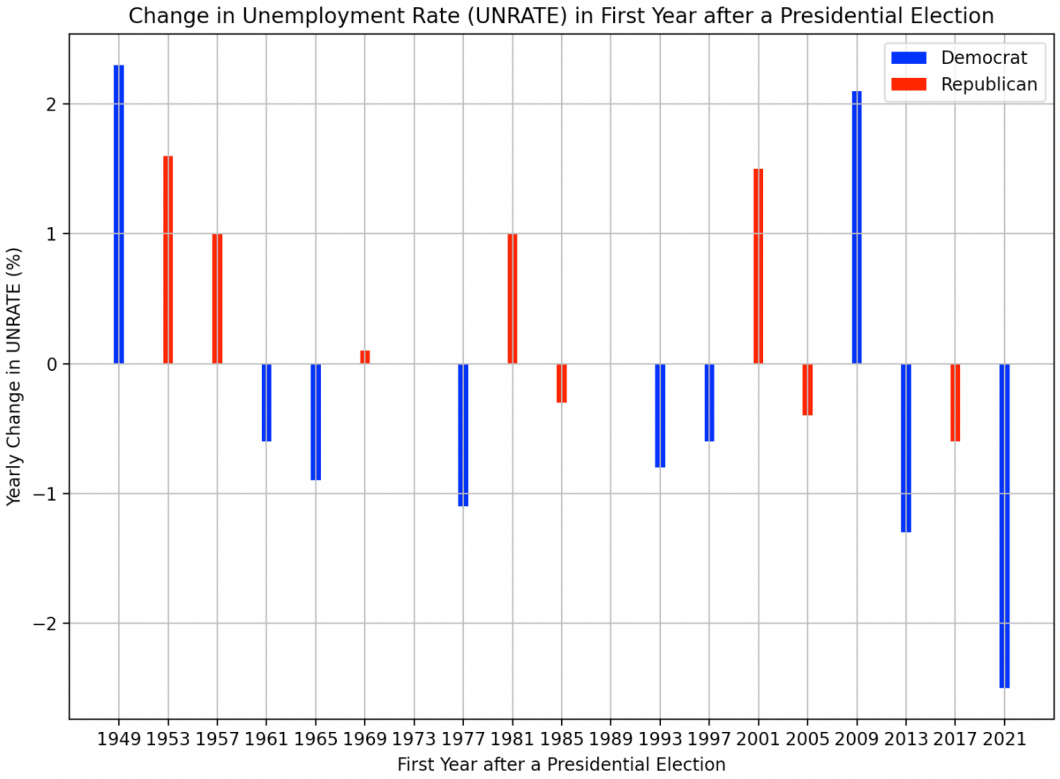
4.
Your file that contains the calculations from the data in the database (10 points)

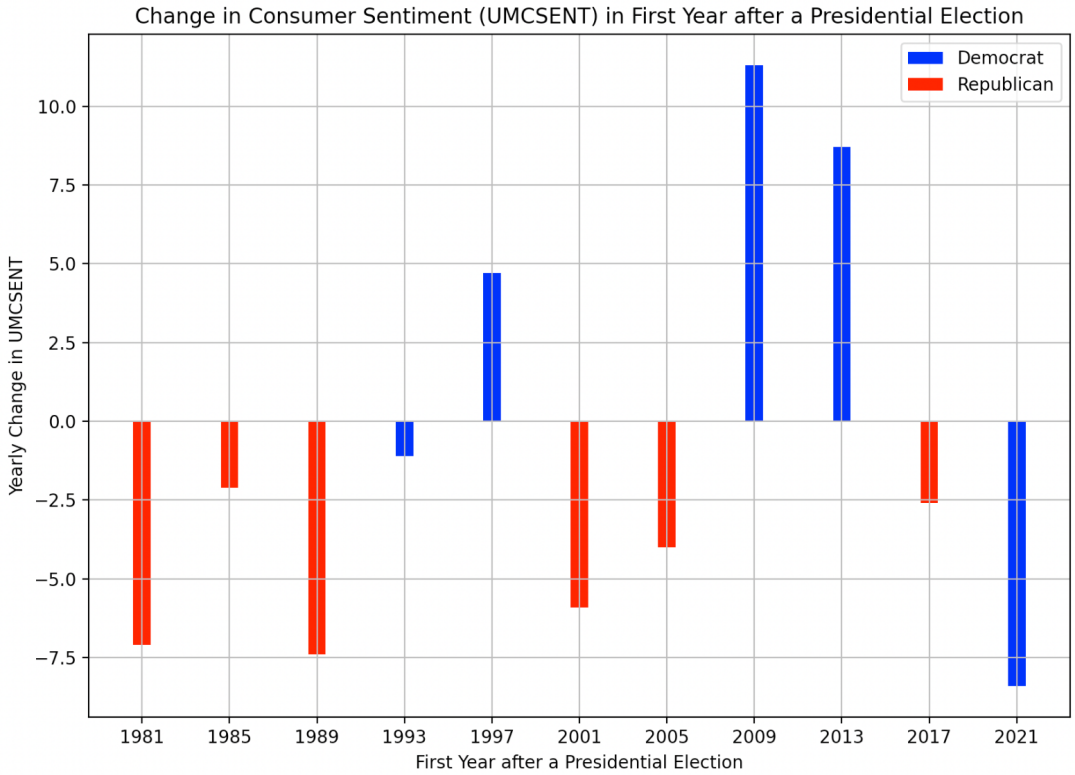
The file that contains our calculations from the data in the database is **calculations.text**. In this file we have 3 sub categories of election calculations, stock data calculations, and economic data calculations.

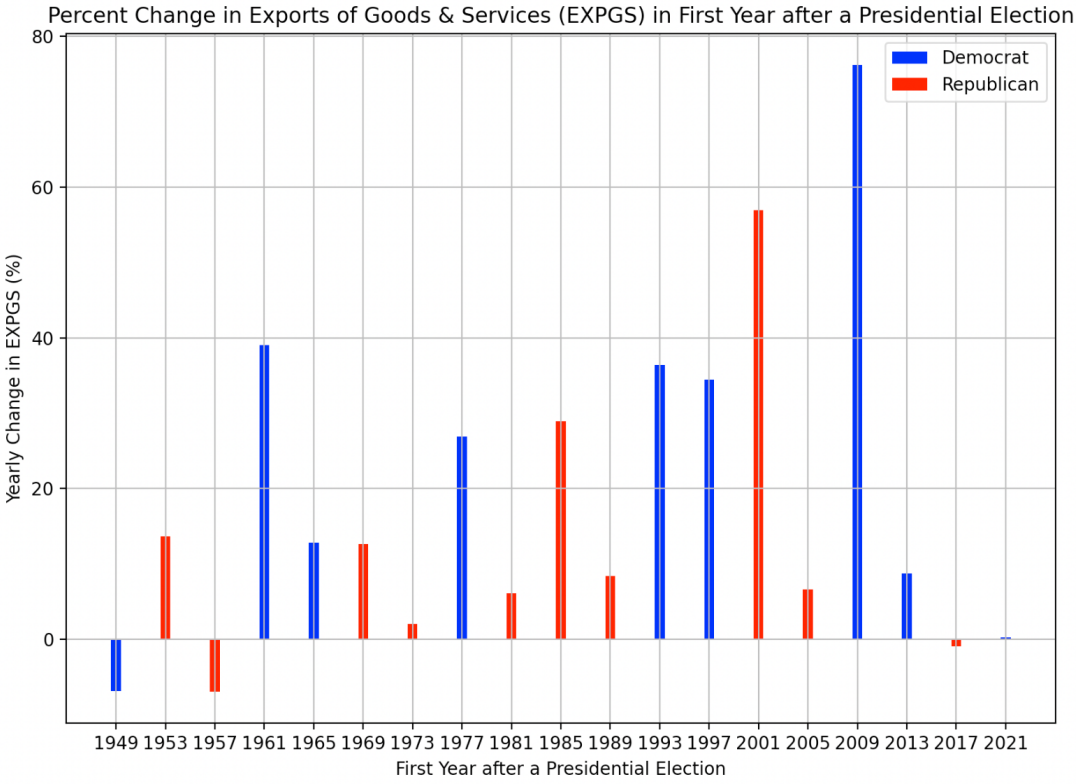
5.
The visualization that you created (i.e. screen shot or image file) (10 points)

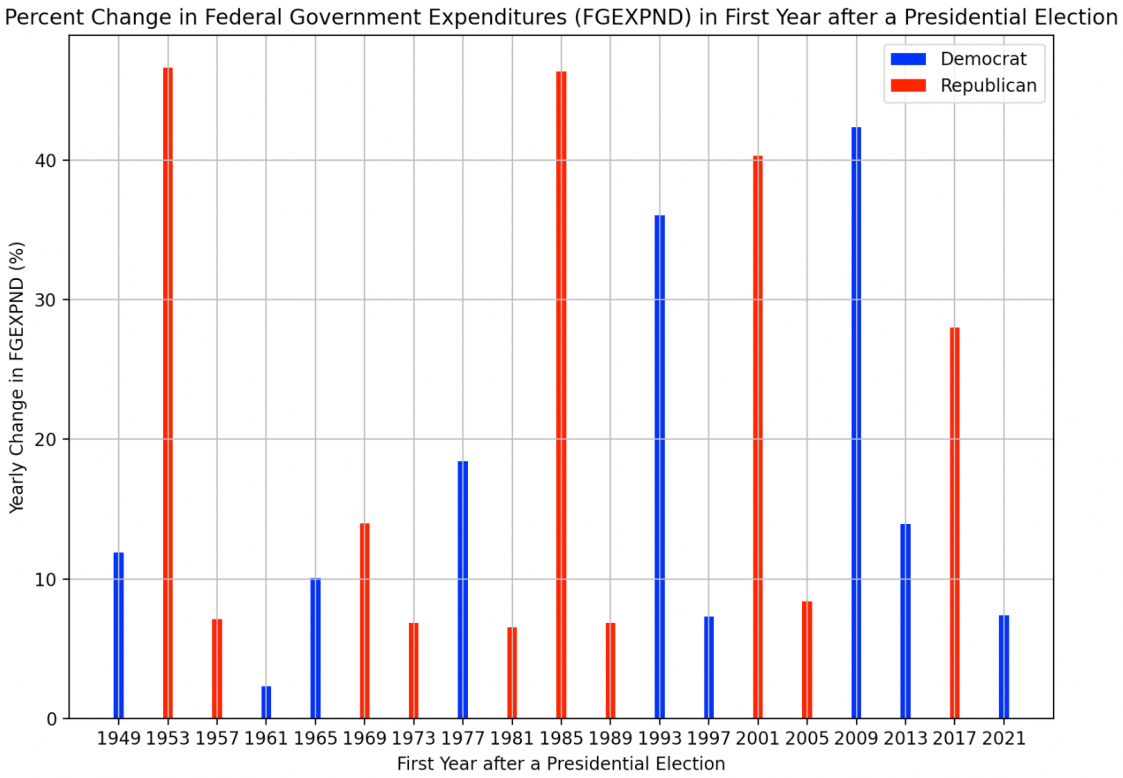

Percent Returns for the Nasdaq Composite (IXIC) in First Year after a Presidential Election
Democrat
Republican
Yearly Returns for IXIC (%)
40
30
20
10
0
−10
−20
1993
1997
2001
2005
2009
2013
2017
2021
First Year after a Presidential Election


Percent Returns for the Dow Jones Industrial Average (DJI) in First Year after a Presidential Election
Democrat
Republican
Yearly Returns for DJI (%)
25
20
15
10
5
0
−5
1993
1997
2001
2005
2009
2013
2017
2021
First Year after a Presidential Election


Change in Unemployment Rate (UNRATE) in First Year after a Presidential Election
Democrat
Republican
Yearly Change in UNRATE (%)
2
1
0
−1
−2
1949 1953 1957 1961 1965 1969 1973 1977 1981 1985 1989 1993 1997 2001 2005 2009 2013 2017 2021
First Year after a Presidential Election


Change in Consumer Sentiment (UMCSENT) in First Year after a Presidential Election
Democrat
Republican
Yearly Change in UMCSENT
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
1981
1985
1989
1993
1997
2001
2005
2009
2013
2017
2021
First Year after a Presidential Election


Percent Change in Exports of Goods & Services (EXPGS) in First Year after a Presidential Election
Democrat
Republican
Yearly Change in EXPGS (%)
80
60
40
20
0
1949 1953 1957 1961 1965 1969 1973 1977 1981 1985 1989 1993 1997 2001 2005 2009 2013 2017 2021
First Year after a Presidential Election



6.

Instructions for running your code (10 points)

1. All you have to do is run the main.py file. This will access all the data and create the database all_data.db where all of our tables are created. Next, it will perform all of the calculations using our data and write the results to calculations.text. Finally, it will create our visualizations.

7. Documentation for each function that you wrote. This includes the input and output for each function (20 points)

- election_data.py:
    - get_winning_party(year)
        - Input: a given year where a presidential election occurred
        - Output: the party (Democrat or Republican) that won that presidential election
    - compile_election_data(start_year, end_year)
        - Input: the election year to start at, the election year to end at,
        - Output: a list of dictionaries where they keys are each election year within the range provided and the values are the party that won that years election

    - enter_into_database(db_name, table_name, year_winner_list, start_index, end_index)
        - Input: takes in name of the database, name of the table, data to add to the database (year_winner_list), what index of the data to start at, what index of the data to end at
        - Output: None, creates a table called table_name in db_name and adds all of the election data from year_winner_list within the start_index and end_index range. Start_index and end_index must be within 24 of each other to ensure that only 25 items are able to be added to the database at once
    - drop_table(data_base, table_name)
        - Input: a database, a table name
        - Output: None, drops the given table from the given database
    - elections_main()
        - Input: None
        - Output: None, calls all of the above functions to create the Elections table in the all_data.db that contains a row for a presidential election year and a row for the political party that won the election that year
- stock_data.py
    - get_stock_data(symbol, year, offset)
        - Input: stock ticker symbol, year, number of days to offset the API results by
        - Output: the api call results for the desired stock and year in a dictionary
    - clear_table(data_base, table_name)
        - Input: database name, table name
        - Output: None, clears the indicated table in the database
    - drop_table(data_base, table_name)
        - Input: database name, table name
        - Output: None, deletes the indicated table from the database
    - enter_stock_data_into_database(data, table_name)
        - Input: dictionary of stock data, table name
        - Output: None, inserts the stock data into the table
    - get_yearly_data(symbol, year)
        - Input: stock ticker symbol, year
        - Output: None, uses the get_stock_data() and enter_stock_data_into_database() functions to retrieve the stock data for the whole year and enter it into the table
    - get_post_election_data(symbol, year_start, year_end)
        - Input: stock ticker symbol, start year, end year

        - Ouput: None, uses the yearly stock data tables to create a new table with start and end values for all post-election years between the start and end years (inclusive)
    - stock_data_main()
        - Input: None
        - Ouput: None, runs the get_yearly_data() function for all post-elections years since 1993 for both the Dow Jones Industrial Average and the Nasdaq Composite indexes. Then runs get_post_election_data() for both indexes.
- econ_data.py
    - get_econ_data(code, start, end, limit)
        - Input: FRED code for the economic stat, start date, end data, limit for number of observations the api will retrieve
        - Output: the api call results for the desired economic stat from the start to end date in a dictionary
    - clear_table(data_base, table_name)
        - Input: database name, table name
        - Output: None, clears the indicated table in the database
    - drop_table(data_base, table_name)
        - Input: database name, table name
        - Output: None, deletes the indicated table from the database
    - enter_econ_data_into_database(data, table_name)
        - Input: dictionary of economic data, table name
        - Output: None, inserts the data into the table
    - get_monthly_data(code, start_year, end_year)
        - Input: economic stat code, start year, end year
        - Output: None, uses the get_econ_data() and enter_econ_data_into_database() functions to retrieve the economic data from start year to end year and and enters into into a table
    - get_post_election_data(symbol, start_year, end_year)
        - Input: economic stat code, start year, end year
        - Ouput: None, uses the economic data table previously created to create a new table with start and end values for all post-election years between the start and end years (inclusive)
    - econ_data_main()
        - Input: None
        - Output: None, calls get_monthly_data() for Unemployment Rate, Consumer Sentiment, Exports of Goods & Services, and Federal Government Expenditure, and then calls get_post_election_data() for all them

- calculations.py
    - elections_calculations()
        - Input: None
        - Output: None, uses the Elections table to calculate how many of the past 30 elections were won by Democrats and Republicans and prints the results to the calculations.text file
    - average_returns(symbol)
        - Input: stock ticker symbol
        - Output: None, uses the post-election stock data tables to calculate the percentage returns for the given index in the post-elections years and records the average. Prints the results to the calculations.text file
    - stock_volatiltiy(symbol)
        - Input: stock ticker symbol
        - Output: None, uses the yearly stock data tables to calculate the relative standard deviation of the index price over the year for each post-election year and records the average. Prints the results to the calculations.text file
    - RvsD_stocks(symbol)
        - Input: stock ticker symbol
        - Output: None, splits the returns and volatility calculations by which party won the election, and records averages for each. Prints the results to the calculations.text file
    - stock_data_calculations()
        - Input: None
        - Output: runs average_returns(), stock_volatilty(), and RvsD_stocks() functions for the Dow Jones Industrial Average and Nasdaw Composite indexes
    - average_change(code)
        - Input: economic stat code
        - Output: None, uses the post-election economic data tables to calculate the change for the given stat in the post-elections years and records the average. Prints the results to the calculations.text file
    - RvsD_econ(code)
        - Input: economic stat code
        - Output: one, splits the change calculations by which party won the election, and records averages for each. Prints the results to the calculations.text file
    - econ_data_calculations()
        - Input: None

        - Output: None, runs the average_change() and RvsD_econ() functions for Unemployment Rate, Consumer Sentiment, Exports of Goods & Services, and Federal Government Expenditure
    - run_calculations()
        - Input: None
        - Output: None, runs the elections_calculations(), stock_data_calculations(), and econ_data_calculations() functions
- visualizations.py
    - stock_data_graphs(symbol)
        - Input: stock ticker symbol
        - Output: None, creates a bar graph for returns of the given index in post-elections years, color-coded by election winner
    - econ_data_graphs(code):
        - Input: economic stat code
        - Output: None, creates a bar graph for change in the given economic stat in post-election years, color-coded by election winner
    - vis_main()
        - Input: None
        - Outptut: None, runs stock_data_graphs() for the Dow Jones Industrial Average and Nasdaq Composite indexes, and runs econ_data_graphs() for Unemployment Rate, Consumer Sentiment, Exports of Goods & Services, and Federal Government Expenditure
- Main.py
    - clear_database()
        - Input: None
        - Ouput: None, clears all the tables from the database and clears the text from calculations.text
    - main(start_fresh=FALSE)
        - Input: boolean indicating whether or not to start fresh
        - Output: None, if start_fresh is true, runs clear_database(). Runs the whole program by running elections_main(), stock_data_main(), econ_data_main(), run_calculations(), and vis_main()

8.
You must also clearly document all resources you used. The documentation should be of the following form (20 points)

Date	**Issue Description**	**Location of Resource**	**Result (Did it solve the issue?)**

11/16	Marketstack API	https://marketstack.com/documentation	We used this API to get the stock information from the Dow Jones Industrial Average and NASDAQ composite
11/30	FRED API	https://fred.stlouisfed.org/docs/api/fred/index.html#General_Documentation	We used this API to get the economic information for the specific indicators we were looking at
12/5	Elections results website	https://www.presidency.ucsb.edu/statistics/elections	We used this website to scrape the winning party of the last 25 elections and put the year and winner into a dictionary
12/5	Website for help with sqlite	https://docs.python.org/3/library/sqlite3.html	We had an issue when trying to use sqlite so we used this website to help us along with figuring out how to use the information and what code to write for our databases
12/11	Website for help with matplotlib	https://www.w3schools.com/python/matplotlib_labels.asp	We had an issue when trying to create our visualizations so we used this website to help us figure out matplotlib and how to create the visualizations that we needed to display our findings
12/11	Website for help with matplotlib	https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.bar.html	We had an issue when trying to create our visualizations so we used this website to help us figure out matplotlib and how to create the visualizations that we needed to display our findings