SI 206 Final Project Report
Analysis of S&P 500 Index and Sectors
By Kedi Jiang, Annabel Zhuang
Link: https://github.com/annazjf/SI-206-Final-Project-JK

Goals

There are 2 main goals for the project. Firstly, we figure out whether each sector in S&P 500 has been in the ranking for the same number of years, and what sector is relatively new in the ranking. Secondly, we took a close look into 6 representative companies in 3 sectors of S&P 500: healthcare, information technology, and energy and analyzed which sector/companies' stock follows that trend of S&P 500 index most closely. From there, we hope to conclude which sector under the S&P 500 has the strongest positive correlation with the overall stock market index of S&P 500.

API/Website/Library we've worked with and what data we've gathered and used

- **Alpha Vantage (API)**:
    - Time series data from different sectors of S&P 500
        - Information Technology: Apple(AAPL), Microsoft(MSFT)
        - Energy: Exxon Mobil (XOM), Chevron (CVX)
        - Health Care: Johnson & Johnson (JNJ), Eli Lilly & Co. (LLY)
    - In form of Json file
    - We extract: Dates and closing stock price columns
- **Yahoo Finance(yfinance library)**
    - Times series data of S&P 500 index, in the form of csv file
    - We extract: Dates and closing stock price columns
- **Wikipedia (Website):**
    - Data of S&P 500 headquarter Sector, date added.

Goals we've achieved

- We calculated the rate of change of closing stock price and index using sqlite UPDATE command. We plotted the dates against the rate of change for each company using matplotlib. This is meant to compare how the S & P 500 index changes from a day to the previous day with how the stock price for a company changed. If the rate of change is similar on the same day, then we can say that the company's stock price has a positive correlation with the index. After calculation, we created a new column to the calculation file for the gap between index and closing price, and computed the average. In the end, we found that Exxon Mobil, an American multinational oil and gas corporation, follows the trend of the S & P 500 index most closely.
- For the goal of comparing the number of years that each sector has been in S & P 500, we calculated the days gap between the "newest_added_date" and the "oldest_added_date" for each sector with the GROUP BY command, and we wrote the outcome into the result file. Then we plotted the sectors against the number of years inside the list using matplotlib. From the plot, we found that except for "Real Estate" and "Communication Services", all the sectors have been in S&P 500 for about 65 years. The "Real Estate" is the newest added sector in the list for 20.79 years and follow by the "Communication Services" sector for 35.78 years.

Problems that I faced:

I struggled to only extract the data I wanted and grab only 25 rows of data at a time, but we were able to fix the problem by adding a unique id column to our csv file, and loading sequentially to our database.

Calculation File:

Result file output:


```
{
  "Communication Services": {
    "newest_added_date": "2022-04-11",
    "oldest_added_date": "1976-06-30",
```

```
    "difference_years": 45.78
  },
  "Consumer Discretionary": {
    "newest_added_date": "2021-03-22",
    "oldest_added_date": "1957-03-04",
    "difference_years": 64.05
  },
  "Consumer Staples": {
    "newest_added_date": "2023-03-15",
    "oldest_added_date": "1957-03-04",
    "difference_years": 66.03
  },
  "Energy": {
    "newest_added_date": "2022-10-12",
    "oldest_added_date": "1957-03-04",
    "difference_years": 65.61
  },
  "Financials": {
    "newest_added_date": "2022-11-01",
    "oldest_added_date": "1957-03-04",
    "difference_years": 65.66
  },
  "Health Care": {
    "newest_added_date": "2023-03-15",
    "oldest_added_date": "1957-03-04",
    "difference_years": 66.03
  },
  "Industrials": {
    "newest_added_date": "2022-09-19",
    "oldest_added_date": "1957-03-04",
    "difference_years": 65.54
  },
  "Information Technology": {
    "newest_added_date": "2023-03-20",
    "oldest_added_date": "1957-03-04",
    "difference_years": 66.04
  },
  "Materials": {
    "newest_added_date": "2019-06-07",
    "oldest_added_date": "1957-03-04",
```

```
    "difference_years": 62.26
  },
  "Real Estate": {
    "newest_added_date": "2022-09-19",
    "oldest_added_date": "2001-12-03",
    "difference_years": 20.79
  },
  "Utilities": {
    "newest_added_date": "2022-10-03",
    "oldest_added_date": "1957-03-04",
    "difference_years": 65.58
  }
}
```

There are 6 more files of calculation result

CVX_roc.csv

```
Date,Company ROC,S&P500 ROC
2023-04-19,0.004886664704150795,0.005988193161113811
2023-04-18,-0.0009374267635340788,8.42690983805877e-05
2023-04-17,0.0023457658925637886,-0.000854489519194225
2023-04-14,0.00889304937982688,-0.0032952622489341504
2023-04-13,-0.0020296914868939593,0.002073664659441411
2023-04-12,-0.013190772270323727,-0.013089093405503106
2023-04-11,-0.0035331527499705235,0.0041520523068737224
2023-04-10,-0.0054958042784541234,4.1354188044385406e-05
2023-04-06,-0.003802959177609997,-0.0009953113657404464
2023-04-05,0.013301521025946852,-0.003566398372990578
2023-04-04,-0.004944666823640237,0.0024985979631609897
2023-04-03,0.005383341221012758,0.005830772911119979
2023-03-31,-0.039952927331568064,-0.00368521543043231
2023-03-30,-0.004719293944594326,-0.014231094669152922
2023-03-29,-0.0094217624237944,-0.005682790708887308
2023-03-28,-0.008516722615939354,-0.014037414436131606
2023-03-27,-0.0115367734654210 5,0.0015763243835945212
2023-03-24,-0.010085632730732657,-0.0016442463072129491
2023-03-23,-0.0098679994873766,-0.0056081782089648586
2023-03-22,0.01003106393994294,-0.00297564782693561
2023-03-21,0.020759915422566857,0.01673879835883166
2023-03-20,-0.029690540455715207,-0.012815816483241151
2023-03-17,-0.014490878509509697,-0.008839568874392354
2023-03-16,0.012800315084678932,0.011142238734136958
```

AAPL_roc.csv

```
Date,Company ROC,S&P500 ROC
2023-04-19,0.005880588058805819,0.005988193161113811
2023-04-18,-0.006920002386207699,8.42690983805877e-05
2023-04-17,-0.007448789571694654,-0.000854489519194225
2023-04-14,-0.00012104339405665926,-0.0032952622489341504
2023-04-13,0.0021185158283396543,0.002073664659441411
2023-04-12,-0.032978980430055614,-0.013089093405503106
2023-04-11,0.004372267332917033,0.0041520523068737224
2023-04-10,0.0076492537313432194,4.1354188044385406e-05
2023-04-06,0.016231562056409278,-0.0009953113657404464
2023-04-05,-0.005465808332321181,-0.003566398372990578
2023-04-04,0.01141914997557404,0.0024985979631609897
2023-04-03,0.0032602789349755,0.005830772911119979
2023-03-31,-0.007642775470903183,-0.00368521543043231
2023-03-30,-0.015403274711946585,-0.014231094669152922
2023-03-29,-0.009793052475979325,-0.005682790708887308
2023-03-28,-0.019406605710020553,-0.014037414436131606
2023-03-27,0.003996194100856298,0.0015763243835945212
2023-03-24,0.012446297700277982,-0.0016442463072129491
2023-03-23,-0.008237129485179365,-0.005608178208964858
```

JNJ_roc.csv

```
Date,Company ROC,S&P500 ROC
2023-04-19,-0.00641887761340024,0.005988193161113811
2023-04-18,-0.00935211960868769,8.42690983805877e-05
2023-04-17,0.02894230172039002,-0.000854489519194225
2023-04-14,0.0010261362950444615,-0.0032952622489341504
2023-04-13,0.001628075253256212,0.002073664659441411
2023-04-12,-0.013184034675817385,-0.013089093405503106
2023-04-11,0.002135187896535034,0.0041520523068737224
2023-04-10,0.0003043769404028912,4.1354188044385406e-05
2023-04-06,0.005051119766309716,-0.0009953113657404464
2023-04-05,0.002785346654556512,-0.003566398372990578
2023-04-04,-0.04299257291226378,0.0024985979631609897
2023-04-03,-0.01034765600353344,0.005830772911119979
2023-03-31,-0.011794708320050968,-0.00368521543043231
2023-03-30,-0.010129032258064472,-0.014231094669152922
2023-03-29,-0.000782115622759542,-0.005682790708887308
2023-03-28,-0.009718870262866148,-0.014037414436131606
2023-03-27,0.009748386246871415,0.0015763243835945212
2023-03-24,-0.004240052185257702,-0.0016442463072129491
```

LLY_roc.csv

```
Date,Company ROC,S&P500 ROC
2023-04-19,-0.011579200128064819,0.005988193161113811
2023-04-18,-0.0015115933813804148,8.42690983805877e-05
2023-04-17,0.006596199075450779,-0.000854489519194225
2023-04-14,0.00639183563851214,-0.0032952622489341504
2023-04-13,0.0011208069810262297,0.002073664659441411
2023-04-12,-0.016260162601625928,-0.013089093405503106
2023-04-11,-0.01357539628776586,0.0041520523068737224
2023-04-10,0.00813097461817377,4.1354188044385406e-05
2023-04-06,0.003514986376021854,-0.0009953113657404464
2023-04-05,-0.016427271986749602,-0.003566398372990578
2023-04-04,-0.02103577738515902,0.002498597963160989 7
2023-04-03,-0.010941289267384793,0.005830772911119979
2023-03-31,-0.02087016023265094 6,-0.00368521543043231
2023-03-30,-0.00794944965348558,-0.014231094669152922
2023-03-29,-0.014147758959758118,-0.005682790708887308
2023-03-28,0.005299669514990838,-0.014037414436131606
```

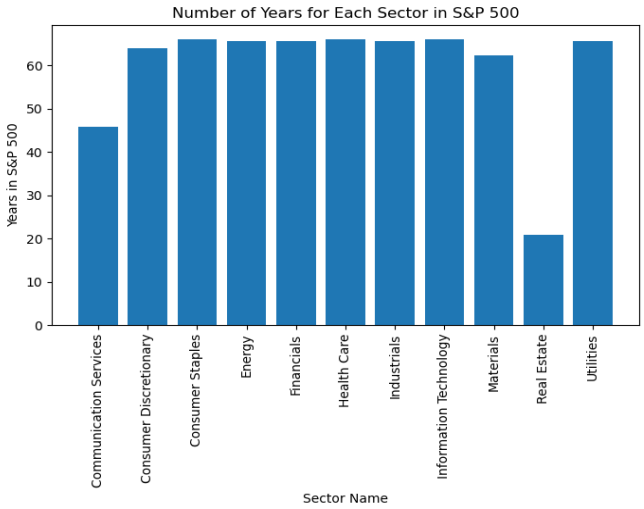
MSFT_roc.csv

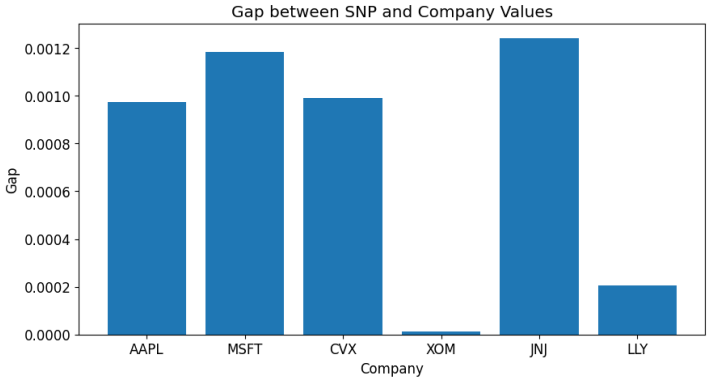
```
Date,Company ROC,S&P500 ROC
2023-04-19,0.008178672538534042,0.005988193161113811
2023-04-18,-0.0002773444271103626,8.42690983805877e-05
2023-04-17,0.0014911398550473587,-0.000854489519194225
2023-04-14,-0.00921052631578956,-0.0032952622489341504
2023-04-13,0.012930733207520756,0.002073664659441411
2023-04-12,-0.021908639249240844,-0.013089093405503106
2023-04-11,-0.0023281244488342623,0.0041520523068737224
2023-04-10,0.023194144892691733,4.1354188044385406e-05
2023-04-06,0.007636753170462133,-0.0009953113657404464
2023-04-05,-0.02489711934156395,-0.003566398372990578
2023-04-04,0.009988042484349835,0.002498597963160989 7
2023-04-03,0.0001741068319521 2537,0.005830772911119979
2023-03-31,0.0037252376144552906,-0.00368521543043231
2023-03-30,-0.014741588622962192,-0.014231094669152922
2023-03-29,-0.012462594613624433,-0.005682790708887308
2023-03-28,-0.018822858365120577,-0.014037414436131606
2023-03-27,0.004178323583911555,0.001576324383594521 2
2023-03-24,0.015160286561979874,-0.0016442463072129491
2023-03-23,-0.010371743236981746,-0.0056081782089648586
2023-03-22,-0.019340200244903853,-0.00297564782693561
```

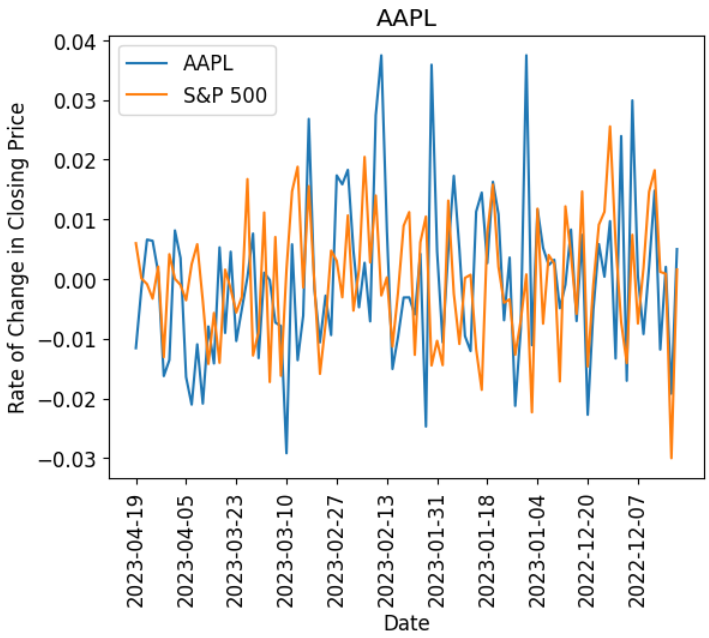
XOM_roc.csv

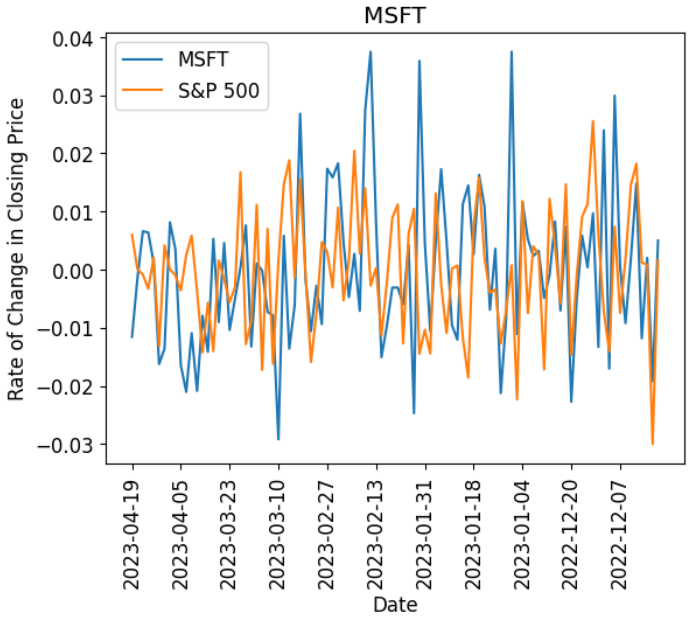
```
Date,Company ROC,S&P500 ROC
2023-04-19,0.008042199930819722,0.005988193161113811
2023-04-18,0.0031740585056189807,8.42690983805877e-05
2023-04-17,-0.0191551222849324,-0.000854489519194225
2023-04-14,0.011769834350479461,-0.0032952622489341504
2023-04-13,-0.0024127531236536074,0.002073664659441411
2023-04-12,-0.004664420834412992,-0.013089093405503106
2023-04-11,0.001041395469929622,0.0041520523068737224
2023-04-10,-0.007022106631989494,4.1354188044385406e-05
2023-04-06,0.004452592980618045,-0.0009953113657404464
2023-04-05,0.016862233811386334,-0.003566398372990578
2023-04-04,-0.016839046072313863,0.0024985979631609897
2023-04-03,0.009650495565988518,0.005830772911119979
2023-03-31,-0.05571342461035046,-0.00368521543043231
2023-03-30,-0.0015502462155754305,-0.014231094669152922
2023-03-29,-0.004840624714585818,-0.005682790708887308
2023-03-28,-0.016886930983847186,-0.014037414436131606
2023-03-27,-0.012322628827483265,0.0015763243835945212
2023-03-24,-0.021455576559546278,-0.0016442463072129491
2023-03-23,-0.00115908432338457,-0.0056081782089648586
2023-03-22,0.011410888695484062,-0.00297564782693561
2023-03-21,0.023424801606272135,0.01673879835883166
2023-03-20,-0.04288116591928254,-0.012815816483241151
2023-03-17,-0.025475841874084915,-0.008839568874392354
2023-03-16,0.01191907051282049,0.011142238734136958
```

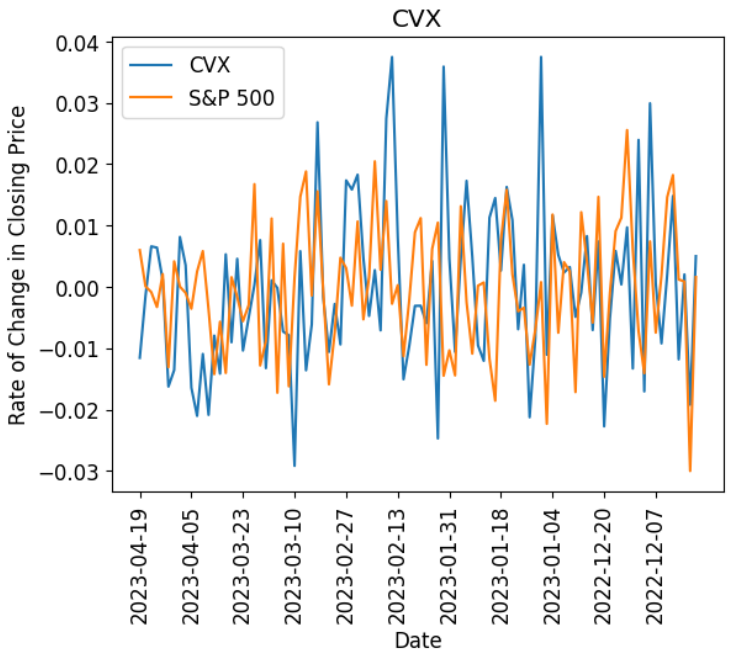
Visualizations:

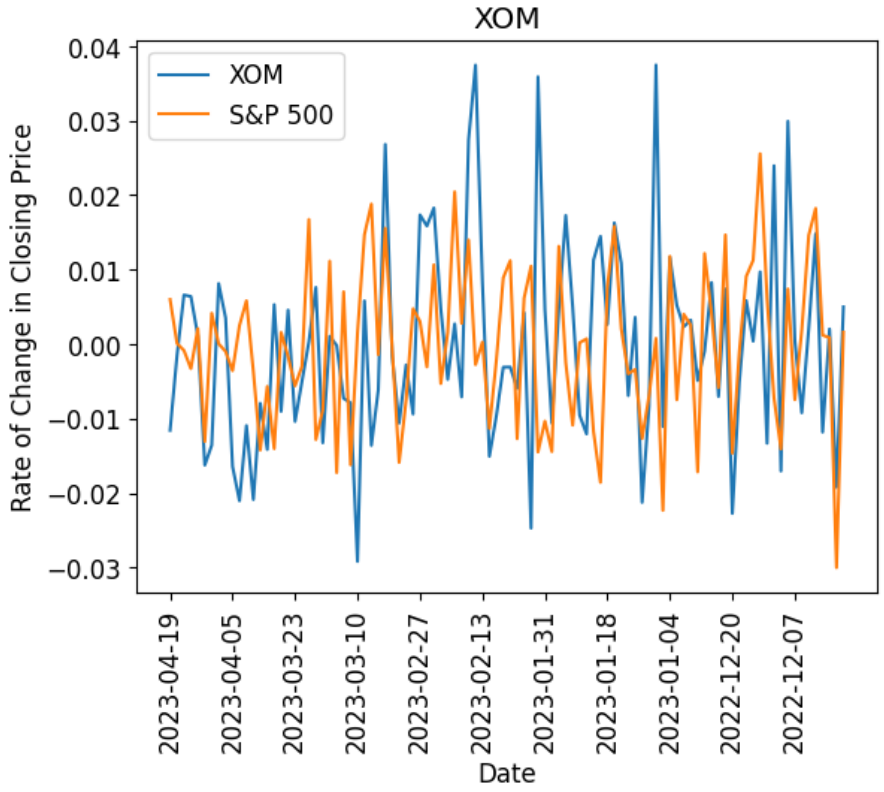


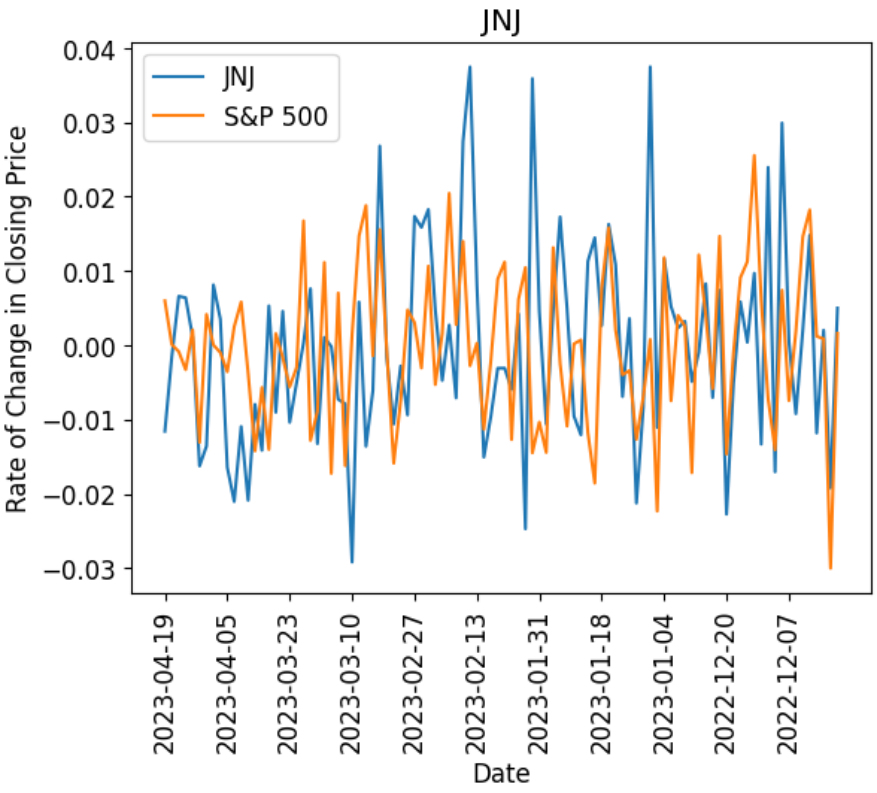


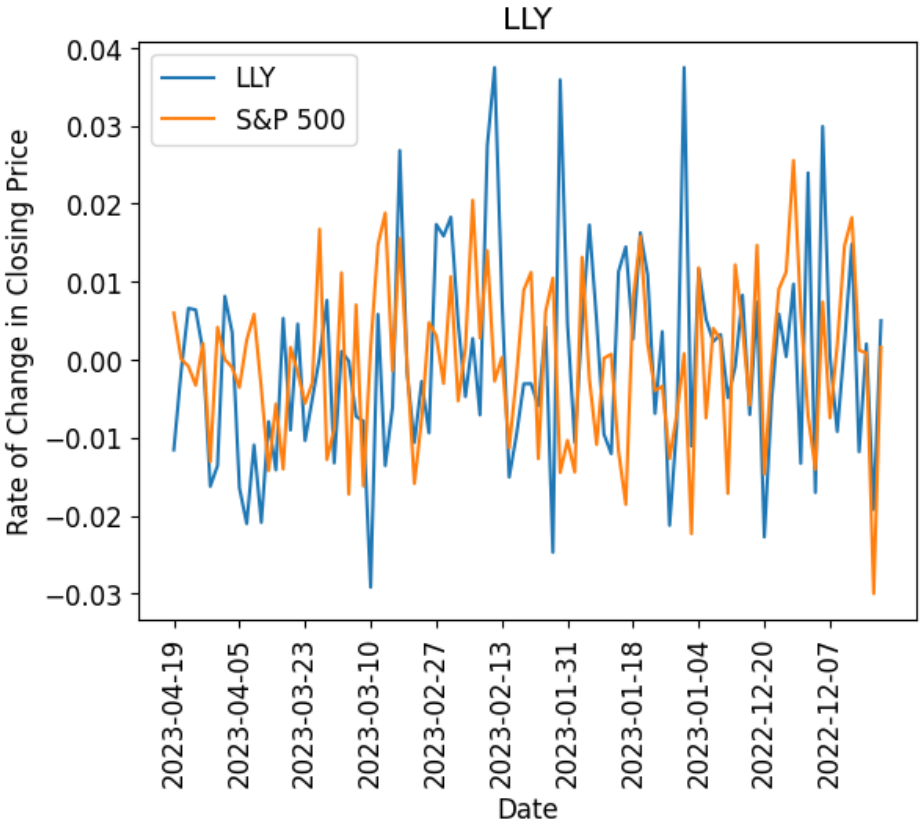

AAPL
AAPL
S&P 500
Rate of Change in Closing Price
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
2023-04-19
2023-04-05
2023-03-23
2023-03-10
2023-02-27
2023-02-13
2023-01-31
2023-01-18
2023-01-04
2022-12-20
2022-12-07
Date


MSFT
MSFT
S&P 500
Rate of Change in Closing Price
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
2023-04-19
2023-04-05
2023-03-23
2023-03-10
2023-02-27
2023-02-13
2023-01-31
2023-01-18
2023-01-04
2022-12-20
2022-12-07
Date


CVX
CVX
S&P 500
Rate of Change in Closing Price
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
2023-04-19
2023-04-05
2023-03-23
2023-03-10
2023-02-27
2023-02-13
2023-01-31
2023-01-18
2023-01-04
2022-12-20
2022-12-07
Date


XOM
XOM
S&P 500
Rate of Change in Closing Price
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
2023-04-19
2023-04-05
2023-03-23
2023-03-10
2023-02-27
2023-02-13
2023-01-31
2023-01-18
2023-01-04
2022-12-20
2022-12-07
Date


JNJ
JNJ
S&P 500
Rate of Change in Closing Price
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
2023-04-19
2023-04-05
2023-03-23
2023-03-10
2023-02-27
2023-02-13
2023-01-31
2023-01-18
2023-01-04
2022-12-20
2022-12-07
Date


LLY
LLY
S&P 500
Rate of Change in Closing Price
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
2023-04-19
2023-04-05
2023-03-23
2023-03-10
2023-02-27
2023-02-13
2023-01-31
2023-01-18
2023-01-04
2022-12-20
2022-12-07
Date

Instructions for running the code:

I. gather_Annabel.py

First run the gather_Annabel.py file, you will get a csv file and the two tables within the database called (founding data). Those two tables are companies and Sector.
Then you can see a bar chart visualization demonstrating the number of years for each sector being in S&P 500 ranking.

II. DataGathering.ipynb

1. Run All, this will gather all data from Alpha Vantage API and Yahoo finance, and store different tables into company_stock.db. Note that each table only has 25 rows for now, but we have more data to read.
2. Run cell 13 three more times until it turns all tables except SP500 table 100 rows.
3. Run cell 19 three more times, which will load 100 rows to the SP500 table.

III. Data_Calc_Visualization

1. Run all, this will do all the calculation needed and show graphs

IV. Combine_db.py

1. Run the file to combine all tables in founding_data.db and company_stock.db.

Code documentation:

I. gather_Annabel.py

1. get_SP_companies_table() function performs web scraping on the Wikipedia page for the S&P 500 companies table. It uses the BeautifulSoup library to parse the HTML content of the page from a specific URL, extract the data from rows, and return a list of lists containing the rows of the table.

2. write_row(rows, filename): This function takes the rows from the scraping and writes them to a csv file with the filename indicated.

3. setUpDatabase(db_name): This function takes in a database name, creates a connection to a SQLite database with that name, sets up a cursor object, and returns the cursor and the connection.

4. creat_dict(cur2, conn2): This function creates the table and the sector dictionary with the 11 GICS (Global Industry Classification Standard) sector names as keys and their corresponding sector id as values. The id is the Sector_id in the companies table to make sure there is no duplicate string data. It returns the dictionary

5. createdb(cur2, conn2, dicti): This function takes in the cursor, connection, and the sector dictionary that was created previously. It creates a table named “companies” in the

SQLite database, reads the data from the csv file created in the write_row() function, and inserts it into the table. It also takes the id in the dictionary that passed in as the sector_id in the table. It returns None.

6. get_sector_dictionary(cur, conn): This function takes in the connection and cursor. It queries the database to get the newest and oldest dates added for each sector, calculates the difference in years between them, and creates a dictionary with each sector name as a key and another dictionary as the value. The inner dictionary has the key and value of newest_added_date, oldest_added_date, and difference_year. The output is the dictionary.

7. plot_the_result(dic): This function takes the dictionary returned from get_sector_dictionary() and creates a bar graph using the Matplotlib library with the sector names on the x-axis and the difference_years on the y-axis, indicating the number of years for each sector in S&P 500. It returns None.

V. DataGathering.ipynb

Function	**Description**
make_company_json(company_stock_name)	This function retrieves the daily adjusted time series data for a specified company's stock from the Alpha Vantage API and saves it in a JSON file. The file name will be the company_stock_name followed by the ".json" extension. Parameters: company_stock_name: A string parameter that specifies the stock symbol for the company whose data needs to be retrieved and saved in a JSON file. Return: None
read_data(filename)	Description: This function reads the data from a JSON file with the specified filename and returns the data in a dictionary format. Parameters: filename: A string parameter that specifies the name of the JSON file to be read. Return Value: Returns the data in a dictionary format.
open_database(db_name)	Description: This function opens a connection to the specified database and returns a cursor object and a connection object. Parameters: db_name: A string parameter that specifies the name of the database to connect to. Return Value: Returns a cursor object and a connection object.
make_stock_close_tb(data, cur, conn,company)	Description: This function creates a table with the name of the given company in the database specified by the connection object. It inserts the data , 25 rows at a time, from the given dictionary object containing the daily closing prices of the stock. It stores the date and closing price for each day in a separate row of the table. Parameters: data: A dictionary object containing the daily closing prices of the stock. cur: A cursor object representing the database cursor.

	conn: A connection object representing the database connection. company: A string parameter that specifies the name of the company for which the data is being stored in the database. Return Value: This function does not return any value. It inserts data into the specified database.

VI. Data_Calc_Visualization

extract_Data(company)	Description: This function extracts the data from the specified company's table and the S&P 500 table in the specified database. It returns a tuple containing lists of dates, company closing prices, and S&P 500 closing prices. Parameters: company: A string parameter that specifies the name of the company for which the data is being extracted from the database. Return Value: Returns a tuple containing three lists: dates: A list of dates on which the data was recorded. company_close: A list of closing prices of the company's stock. sp500_close: A list of closing prices of the S&P 500 index.
get_roc(company_close,sp500_close)	Description: This function calculates the rate of change (ROC) for a given company's closing prices and S&P 500 index closing prices. It returns a tuple containing two lists of ROC values, one for the company and one for the S&P 500 index. Parameters: company_close: A list of the company's closing prices. sp500_close: A list of the S&P 500 index's closing prices. Return Value: Returns a tuple containing two lists: company_roc: A list of rate of change values for the company's closing prices. sp500_roc: A list of rate of change values for the S&P 500 index's closing prices.
write_to_file(which_company, dates, company_roc, sp500_roc)	Description: This function writes a CSV file containing the dates, rate of change values for a given company's closing prices, and rate of change values for the S&P 500 index's closing prices. The file is named after the given company, and it is saved in the current working directory. Parameters: which_company: A string representing the name of the company. dates: A list of dates corresponding to the rate of change values. company_roc: A list of rate of change values for the given company's closing prices. sp500_roc: A list of rate of change values for the S&P 500 index's closing prices. Return Value: This function does not return anything.
read_new_csv_to_db(csv_file, cur,conn)	Description: Creates a new table in the database if it doesn't already exist. Reads the data from the specified CSV file and inserts it into the table. Commits the changes to the database. Parameters:

	csv_file : str : the name of the CSV file to be read and inserted into the database. cur : sqlite3.Cursor : the cursor object for the database connection. conn : sqlite3.Connection : the connection object for the database. Returns:None
graph_trend(dates, company_roc, sp500_roc, company_name)	Plots a line graph of the rate of change in closing price of a company and the S&P 500 over time. Args: dates: A list of strings representing the dates. company_roc: A list of floats representing the rate of change in closing price for the company. sp500_roc: A list of floats representing the rate of change in closing price for the S&P 500. company_name: A string representing the name of the company. Returns: None.

VII. Combine_db.py

copy_database(src_file, dst_file):	Copies a SQLite database from the source file to the destination file. Args: src_file (str): The path to the source database file. dst_file (str): The path to the destination database file. Returns: None.

Documentation of resources used:

Date	Issue Description	Location of Resource	Result
4/22/2023	Hope to auto increment the id	https://stackoverflow.com/questions/2935658/beautifulsoup-get-the-contents-of-a-specific-table	I referenced the practice on this website, and it solved the problem.
4/19/2023	Learn how to use Alpha Vantage API	https://www.alphavantage.co/documentation/#time-series-data	Referred to the example code and got the information I want
4/19/2023	Learn how to get 6 month of S&P 500 index	https://github.com/ranaroussi/yfinance	Found the yfinance library