Link to Github repository: https://github.com/arundelm/206finalproject.git

A. The goals for your project including what APIs/websites you planned to work with and what data you planned to gather (10 points)

For our project, we set out to analyze how different financial assets—Bitcoin, the S&P 500, Gold, and Oil—have performed over time when adjusted for inflation using the Consumer Price Index (CPI). In addition to adjusting for inflation, we also wanted to see the individual asset performance and volatility across this time period as well. We initially planned to pull data from several public APIs: CoinGecko for Bitcoin, Yahoo Finance and Alpha Vantage for stocks and gold, and FRED for CPI and oil prices, but CoinGecko and other Bitcoin APIs were giving SSLErrors, causing us to also use Yahoo Finance for Bitcoin prices. All the data was stored in a SQLite database so we could run analysis like price-to-CPI ratios, asset correlations, monthly returns, and monthly volatility. We then visualized these results with graphs to grow our understanding of these trends.

B. The goals that were achieved including what APIs/websites you actually worked with and what data you did gather (10 points)

We successfully met our data collection goals using a combination of APIs. Due to SSL issues with CoinGecko and then CoinCap, we pivoted to using the Yahoo Finance API for Bitcoin data. We used Yahoo Finance's module for retrieving S&P 500 prices and Alpha Vantage for obtaining monthly gold prices. For each asset (Bitcoin, S&P500, Oil) we collected 1 price for each month since 2016. For gold, we collected an open and close price for each month since 2016, which we then used for our shared integer key. If a gold price increased over that month, it corresponds to a 1 and decreased corresponds to a 0 in our Gold_Change table. The FRED API was used to collect data on the Consumer Price Index (CPI) and oil prices. We collected 1 datapoint per month for the CPI since 2016 as well. All this information was processed and stored in a SQLite database. We grouped together everything into one table since they all had the same corresponding dates, except for our Gold_Change table. We then selected and cleaned the data to perform our calculations and generate visualizations based on normalized prices, monthly returns, volatility, and asset correlations.

C. The problems that you faced (10 points)

Several technical issues arose throughout the project. The most immediate obstacle was an SSL certificate error while attempting to retrieve data from CoinGecko in an Anaconda environment, prompting us to switch to the CoinCap API. After some time, this error persisted for CoinCap as well, which caused us to pivot to Yahoo Finance, which worked well. Additionally, Alpha Vantage imposed strict API rate limits, which made it difficult to retrieve large amounts of historical data at once, which also allowed us to reach the 25 items at a time constraint. We also encountered formatting inconsistencies in the FRED data, especially with missing or null values represented as a period, which required filtering. Finally, aligning the data from all sources was a challenge due to differing date formats and granularities—some sources provided daily data while others used monthly intervals. We solved the issue misaligning dates by

representing all datasets with the YYYY-MM format, which, if an asset price was measured at a slightly different time than another, it would all still be within the same row as the broad year month format encompassed all differing API data formats. Also, during our grading session, we were instructed to fix our split table, which caused us to have to re-do all of the SQL queries. We put everything into a combined table and then used a Gold_Change separate table for the shared integer key as well as for our JOIN in the code.

D. The calculations from the data in the database (i.e. a screenshot) (10 points)

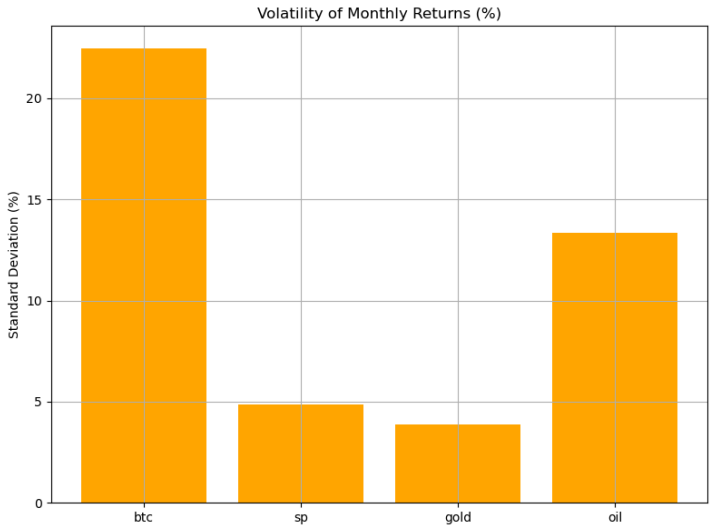
After consolidating the data into a single merged table, we calculated a variety of metrics. These included price-to-CPI ratios for each asset, which were normalized to illustrate their performance relative to inflation. We only displayed the first 20 rows in order to not clutter the calculations output. We also computed the average monthly percentage returns and the volatility of monthly returns for Bitcoin, the S&P 500, Gold, and Oil. A correlation matrix was produced to assess the relationships between these assets and inflation. Lastly, we counted the number of months in which gold went up and down over the 100 months. All these calculations were documented in the calculations_output.txt file, which includes the first ten rows of price-to-CPI ratios, the correlation matrix, average monthly returns, and volatility statistics. During our grading session, we were instructed to fix the format of the asset/cpi table, so we did and now it looks much more readable and organized.

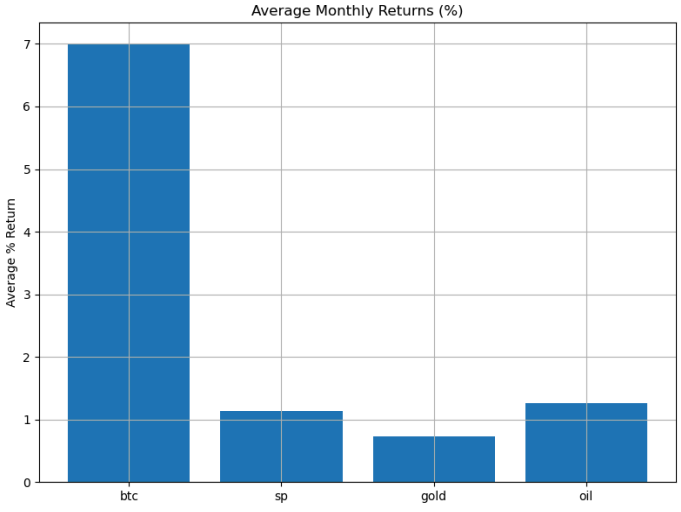
```
Price-to-CPI Ratios (First 20 Rows):
Date       BTC/CPI  SP500/CPI  Gold/CPI  Oil/CPI
2016-07      2.80        0.87       0.53      0.20
2016-08      2.60        0.90       0.54      0.17
2016-09      2.39        0.90       0.52      0.18
2016-10      2.52        0.89       0.52      0.20
2016-11      2.90        0.87       0.51      0.19
2016-12      3.07        0.90       0.46      0.21
2017-01      3.96        0.92       0.45      0.21
2017-02      3.98        0.93       0.47      0.22
2017-03      4.84        0.98       0.49      0.22
2017-04      4.39        0.96       0.49      0.21
2017-05      5.53        0.98       0.49      0.20
2017-06      9.37        1.00       0.49      0.20
2017-07     10.21        0.99       0.48      0.18
2017-08     11.71        1.01       0.49      0.20
2017-09     19.08        1.01       0.51      0.19
2017-10     17.60        1.02       0.49      0.21
2017-11     26.05        1.04       0.49      0.22
2017-12     41.16        1.07       0.49      0.24
2018-01     56.71        1.08       0.50      0.24
2018-02     41.03        1.13       0.51      0.26

Correlation Matrix:
        btc     sp500     gold      oil       cpi
btc     1.00      0.93      0.83      0.56      0.77
sp500   0.93      1.00      0.91      0.65      0.91
gold    0.83      0.91      1.00      0.42      0.86
oil     0.56      0.65      0.42      1.00      0.71
cpi     0.77      0.91      0.86      0.71      1.00

Average Monthly Returns (%):
btc: 6.99%
sp: 1.13%
gold: 0.73%
oil: 1.27%
```

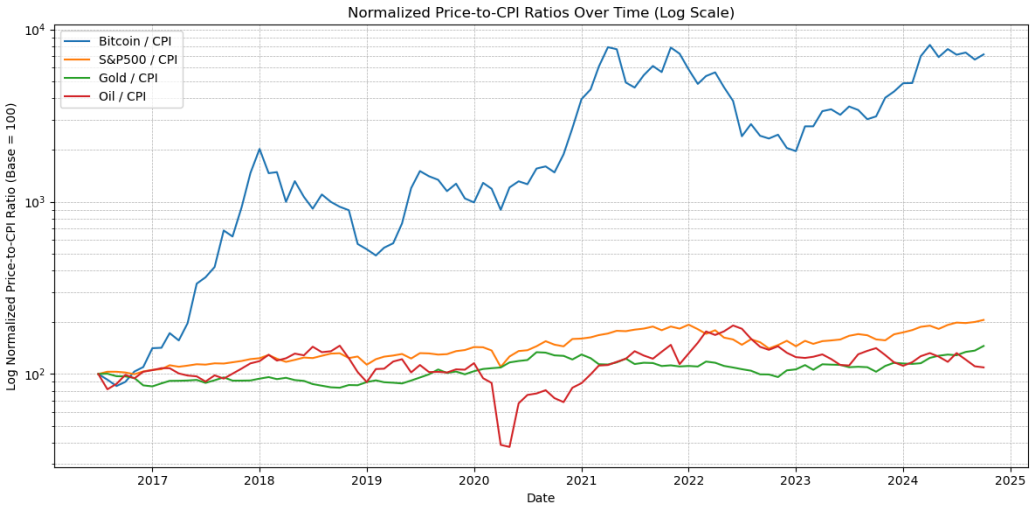
```
Average Monthly Returns (%):
btc: 6.99%
sp: 1.13%
gold: 0.73%
oil: 1.27%

Volatility of Monthly Returns (%):
btc: 22.44%
sp: 4.87%
gold: 3.87%
oil: 13.35%

Gold Price Movement Counts:
Months Gold Went UP:   56
Months Gold Went DOWN: 44
```

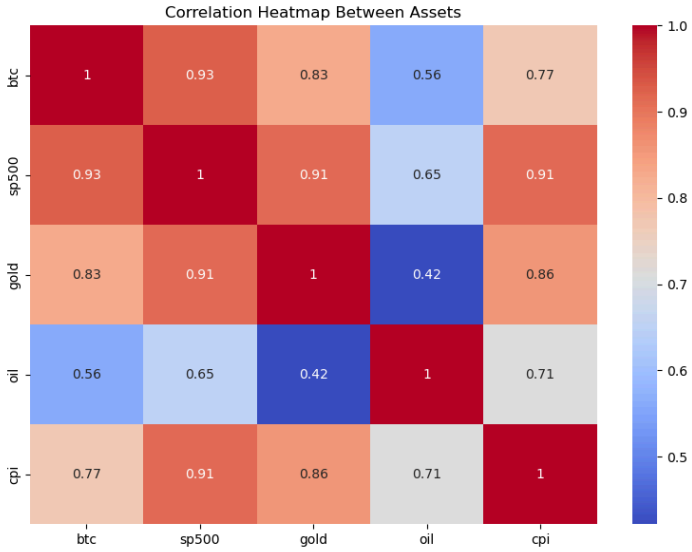
E. The visualization that you created (i.e. screenshot or image file) (10 points)

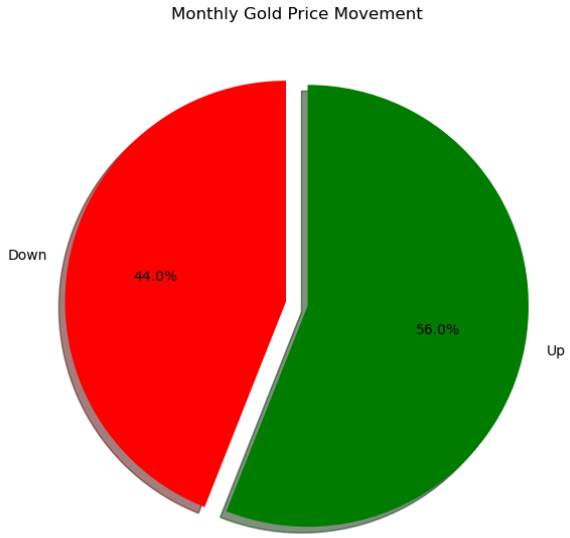
We created multiple visualizations to support and illustrate our analysis. The price_to_cpi_ratio.png graph shows normalized price-to-CPI ratios over time for each asset, providing a comparative view of how they performed relative to inflation. We used a log scale in order to equally display each asset on the graph as Bitcoin, with the largest returns since 2016 dominated the original graph. The correlation_heatmap.png displays a correlation matrix between asset prices and CPI values. We also plotted a scatterplot of Bitcoin versus Gold prices (btc_vs_gold_scatter.png), a bar chart of average monthly returns (avg_monthly_returns.png), and a volatility comparison chart (volatility_bar_chart.png). We included a pie chart of the number of up and down months gold has had in the last 100 months (gold_up_down_chart.png). Each of these visualizations were generated using matplotlib and seaborn, and saved as PNG files.

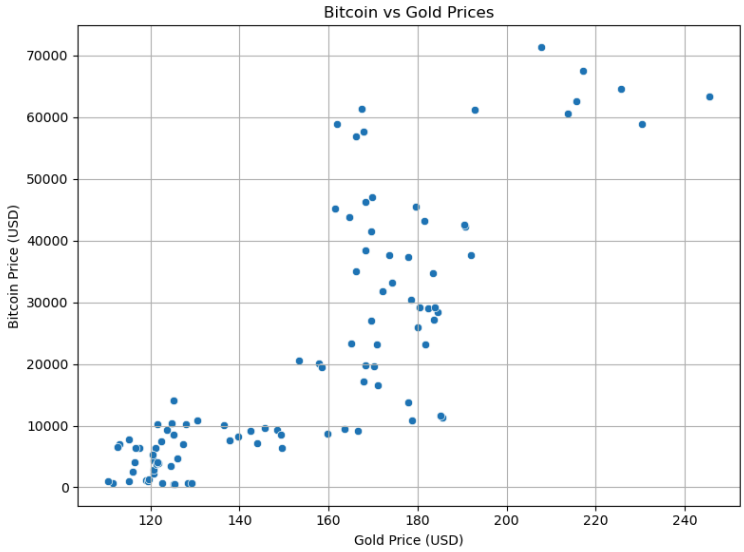












F. Instructions for running your code (10 points)

To run the project, first create a file named api_key.txt and insert your API keys: the first line should contain your Alpha Vantage key, and the second line your FRED key. Then, run fetch_all.py which runs: fetch_bitcoin.py, fetch_sp500_gld.py, and fetch_cpi_oil.py. Each time this is run, it will populate the table with 25 data points from each asset and the CPI. After being run 4 times, it will be full with 100 total rows and running the file any more will just result in a message stating that there are already 100 data points. Finally, run calculations.py, which will

generate the calculations in calculations_output.txt and each graph as well. These scripts will fetch the data, store it into the financial_data.db SQLite database, perform calculations, and generate visual output files (PNG graphs and calculations_output.txt). Ensure you have installed the necessary libraries including pandas, matplotlib, seaborn, requests, sqlite3, and yfinance as well.

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

Each function used in our project is well-documented in the code.

The get_api_key(index) function takes an index as input and returns the corresponding API key from the file. Index of 1 corresponds to AlphaVantage API key, and an index of 2 corresponds to the FRED API key.

Each of these functions begin with 07-2016 (July 2016) and store up until 10-2024 (October 2024) which is 100 months. They add 25 at a time sequentially, so first 25 on the first iteration, then the next 25, and so on until 100.

The fetch_and_store_bitcoin() function retrieves Bitcoin data from CoinCap, processes it into monthly data, and stores it in the Bitcoin_Prices table. It has no input or output as it solely interacts with the SQLite database. Upon each call, it will add 25 Bitcoin prices (1 per month since 2016) to the table until 100 is reached.

The fetch_and_store_sp500() function pulls S&P 500 data using Yahoo Finance and stores it in the SP500_Prices table. It also has no input or output. It adds 25 prices (1 per month since 2016) to the table upon each iteration until 100 is reached.

The fetch_and_store_gold() function fetches daily GLD data from Alpha Vantage, aggregates it by month, and stores it in the Gold_Prices table. It has no input or output. This differs from the others as it stores an open and close price for that month 25 at a time until 100 is reached. If the price increases from open to close, then a 1 is added additionally to the table under gold_change. If it decreases, then a 0 is added. This integer corresponds to our Gold_Change table which holds 0 corresponding to “down” and 1 corresponding to “up”. We then use this in our calculations.

The fetch_and_store_cpi() downloads CPI data from FRED and stores it in CPI_Data. It has no input or output and just stores 25 months of CPI values upon each iteration until 100 is reached.

Similarly, fetch_and_store_oil() gets oil prices from FRED, aggregates them monthly, and stores them in Oil_Prices. It has no input or output and also just stores 25 months of oil prices until 100 is reached.

Finally, calculations.py performs all the data merging, metric calculations, graphing, and file output generation needed for analysis. It has no input and its output is in calculations_output.txt where all of the written calculation output is placed. Additionally, it creates all of the visualizations of this project.

H. 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?)
3/29/2025	Getting a new SSL issue with Anaconda and CoinGecko	HW6 - caching to files/not writing unless the database is already empty	Yes, it avoided repeated API calls since we had already stored the data. It then began to stop working though
3/30/2025	Continuing to get the SSL CoinGecko issue	Free api github repo - CoinCap	Yes, we substituted CoinCap for CoinGecko and it worked
4/02/2025	Got the SSL issue once again with CoinCap after trying to repeatedly use it	Free api github repo- yfinance	Yes, it permanently resolved the issue as we were able to consistently obtain the bitcoin data
4/03/2025	Trouble matching rows across different assets due to inconsistent date formats (some were full dates, others just year-month)	Stack Overflow	Yes, helped us decide to normalize all dates in the database to `YYYY-MM` format, allowing easier joins across all dataset
4/07/2025	Needed to determine how to structure two tables to share an integer key to meet the project rubric	ChatGPT	Yes, we started by referencing SP500_Prices to SP500_Dates but after the grading session switched to creating a Gold_Change file
4/10/2025	Needed a new API for oil prices after inconsistent FRED data response	FRED API documentation	Yes, FRED API worked when throttled, but we had to add error checks such as seeing if the value == '.' or was missing

4/12/2025	Git pull + push errors (couldn't push unless pull + divergent branches)	Github documentation and ChatGPT	Yes, resolved by running `git pull --rebase` before pushing again. We were able to sync local and remote commits.
4/17/2025	Had a split table and needed to put all data into one	SQLite documentation	Yes, it avoided our split table as we made Combined_Prices and used that table for the project
4/17/2025	Needed to represent the movement of gold prices (up/down) visually in a clear format	Matplotlib documentaiton	Yes, we implemented a pie chart with exploded slices for clarity and labeled percent values for readability.