Lucas Faudman 05311800
SI 206 4/26/2020

1.The goals for your project (10 points)

Goals	**Achieved?**
Collect cryptocurrency price data over time	**Yes**
Collect stock price data over time	**Yes**
Collect COVID-19 statistical data over time	**Yes**
Prevent duplicate data or timestamps being added to database	**Yes**
Use a SQL JOIN to select and calculate time series data for cryptocurrency price, stock price or COVID-19 statistical data	**Yes**
Calculate percent change by day for a given variable	**Yes**
Calculate relative change by day between a pair of two given variables	**No**
Write calculations to an output file.	**Yes**
Generate scatter/line plots for cryptocurrency price, stock price or COVID-19 statistical data with Plotly	**Yes**
Generate candlestick plots for cryptocurrency and stock price data with Plotly	**Yes**
Generate candlestick plots for COVID-19 statistical data with Plotly	**No**
Plot a pair of variables so they can be compared in a single graph with 2 y-axis	**Yes**
Plot a pair of variables so they can be compared in a single graph relatively with 1 y-axis	**No**
Allow dynamic manipulation of the variables and date range used in a given plot based on user input.	**Yes**
Allow dynamic manipulation of the plot style of a given plot based on user input.	**Yes**
Deploy an interactive visualization tool by using dash to create a Flask app with interactive callbacks based on current state similar to the useState Hook in React.js.	**Yes**

2. The goals that were achieved (10 points)

The goals that I achieved are marked above with a **Yes** and the goals I did not achieve are marked **No**.

3. The problems that you faced (10 points)

Problems	Solution
Initially I could not prevent duplicating timestamp data between multiple tables.	Create a new table dates for tracking date_ids and their corresponding timestamps.
Initially I could not accurately order data when selecting from the database.	Add “ORDER BY dates.timestamp” to the end of my SELECT statement.
Attempting to create a table that already existed would raise a sqlite3.OperationalError and cause get.py to crash.	Use ‘CREATE TABLE IF NOT EXISTS” instead of “CREATE TABLE”
When plotting two candlestick plots the fact that both displayed green/red made it hard to differentiate between traces.	Count # of candlesticks and change color scheme from green/red to cyan/grey if the number is greater than 0.
When attempting to use a db in an function wrapped with @app.callback, visualize.py would crash due to sqlite3.ProgrammingError since “SQLite objects created in a thread can only be used in that same thread”	Initialize a new DbProcessor inside the body of the @app.callback wrapped function.

4. Your file that contains the calculations from the data in the database (10 points)
See the “calculations” directory for multiple exams of calculation output files in JSON format.

5. The visualization that you created (i.e. screen shot or image file) (10 points)
See “visualization_examples.mov” for a variety of example visualizations.

6. Instructions for running your code (10 points)

BEFORE YOU BEGIN:

1. Unzip “Lucas_Faudman_SI206_Final.zip”
2. Open a terminal and navigate to the unzipped folder “Lucas_Faudman_SI206_Final”

TO GET DATA:

1. In the terminal, run the command “python3 get.py” to insert new items into the database. (This needs to be run ~30-100 times to fill the database with enough data to produce useful plots)
2. (Optional) Use the following command line arguments to specify custom parameters to speed up this process if necessary:

Arg Parser	**Default**	**Example Command**
--db_filename	db.sqlite	python3 get.py --db_filename mydb.sqlite
--max_inserts_per_table	1	python3 get.py --max_inserts_per_table 100
--max_inserts_per_execution	20	python3 get.py --max_inserts_per_execution 99999

TO VISUALIZE THE DATA IN THE DATABASE:

1. In the terminal, run the command “python3 visualize.py”
2. Open a browser a go to the link “http://127.0.0.1:8050/”
3. Now as you change your selections in the dropdown menus and radio buttons, the plots will automatically update to your selections.

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

See filename “documentation.txt” and/or the docstrings under each function in the .py files.

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?)
4/25	SQL error when attempting to create a table that already exists	https://www.w3schools.com/sql/sql_exists.asp	**No**
4/25	SQL error when attempting to create a table that already exists	https://www.sqlitetutorial.net/sqlite-create-table/	**Yes**
4/25	SQL INSERT failed without parenthesis	https://www.w3schools.com/sql/sql_insert.asp	**Yes**
4/25	SQL BETWEEN col1, col2 failed without AND	https://www.w3schools.com/sql/sql_between.asp	**Yes**
4/25	Gaps in Plotly scatter plots were not filling in.	https://plotly.com/python/line-charts/	**Yes**
4/26	Dash callbacks fail with more than 1 Input or Output in the @app.callback wrapper	https://dash.plotly.com/basic-callbacks	**No**
4/26	Dash callbacks fail with more than 1 Input or Output in the @app.callback wrapper	https://dash.plotly.com/interactive-graphing	**Yes**
4/26	Plotly rendering of x-axis was difficult to read when using integer timestamps rather than datetime.datetime objects	https://stackoverflow.com/questions/9744775/how-to-convert-integer-timestamp-to-python-datetime	**Yes**
4/26	App layout rendered elements in a vertical layout always rather than wrapping	https://css-tricks.com/snippets/css/a-guide-to-flexbox/	**Yes**