Task 1

Interface with a stock price data feed

Interface with a stock price data feed and set up your system for analysis of the data

You've been asked to assist with some development to add a chart to a trader's dashboard allowing them to better identify under/over-valued stocks.

The trader would like to be able to monitor two historically correlated stocks and be able to visualize when the correlation between the two weakens (i.e. one stock moves proportionally more than the historical correlation would imply). This could indicate a potential trade strategy to simultaneously buy the relatively underperforming stock and sell the relatively outperforming stock. Assuming the two prices subsequently converge, the trade should be profitable.

Most data visualization for our traders is built on JPMorgan Chase's Perspective data visualization software, which is now open source. If you want to explore that, a link is provided in the resources section.

Before implementing this request using perspective, first you'll need to interface with the relevant financial data feed and make the necessary adjustments to facilitate the monitoring of potential trade opportunities.

Acceptance Criteria

- *getDataPoint* function should return correct tuple of stock name, bid_price, ask_price and price. Note: price of a stock = average of bid and ask
- *getRatio* function should return the ratio of the two stock prices
- main function should output correct stock info, prices and ratio
- Upload a git patch file as the submission to this task
- Bonus: All unit tests inside client_test.py, added/existing have to pass

Solution:

1. Set the environment using Replit (an collaborative browser based IDE)
2. Fixed the broken client datafeed script in the repository by making the required adjustments to it.
3. Generated a patch file of the changes I made

Task 2

Use JPMorgan Chase frameworks and tools

Implement the Perspective open source code in preparation for data visualization

Typically, traders monitor stock prices and trading strategies by having data displayed visually on their screens in chart form. Often these charts will be accompanied by alerts that notify users when certain events occur or when preset price thresholds are hit.

JPMorgan Chase created the Perspective tool over many years to allows users to present and manipulate data feeds visually in web applications.

Perspective provides a set of flexible data transforms, such as pivots, filters, and aggregations. It utilizes bleeding-edge browser technology such as Web Assembly and Apache Arrow and is unmatched in browser performance. It is engineered for reliability and production-vetted on the JPMorgan Chase trading floor and is now available to the development community as Open Source. If you want to explore that, a link is provided in the resources section.

Acceptance Criteria:

- This ticket is done when the graph displayed in the client-side web application is a continuously updating line graph whose y axis is the stock's top_ask_price and the x-axis is the timestamp of the stock. The continuous updates to the graph should be the result of continuous requests and responses to and from the server for the stock data.

- This ticket is done when the graph is also able to aggregate duplicated data retrieved from the server

Solution:

1. Set the environment by using NPM and NVM(node version manager)
2. Fixed the broken typescript files in repository to make the web application output correctly
3. Generated a patch file of the changes I made.

Task 3

Display data visually for traders

Use Perspective to create the chart for the trader's dashboard

Being able to access and adjust data feeds is critical to any trading analysis and stock price monitoring. From the previous tasks, we now have the adjusted data set up on your systems and being piped into Perspective.

For traders to have a complete picture of all the trading strategies being monitored, several screens typically display an assortment of live and historical data at their workstation.

Given there is a lot of information and data being produced at once, visualizing data in a clear manner with UI/UX considerations accounted for is critical to providing traders with the tools to improve their performance.

Acceptance Criteria:

- This ticket is done when the numbers from the python script render properly in the live perspective graph. This means the ratio between the two stock prices is tracked and displayed. The upper and lower bounds must be shown on the graph too. And finally, alerts are shown whenever these bounds are crossed by the ratio (the guide below will also give more detail and visuals to help you understand these requirements better)

- This ticket is done when a patch file is uploaded along with a video or audio explanation of the final chart you have produced

Solution:

1. Set the environment by using NPM and NVM(node version manager)
2. Fixed the broken typescript files in repository to make the web application output correctly
3. Generated a patch file of the changes I made.