Yahoo Finance Script Update Specification

[**Enclosures 2**](https://docs.google.com/document/d/1dKG7qvuSGz4iKflH1uwZE_H3eBD1pEUQTcwwEVuvY6Q/edit#heading=h.unkn0mxw5io2)

[**What this code does 2**](https://docs.google.com/document/d/1dKG7qvuSGz4iKflH1uwZE_H3eBD1pEUQTcwwEVuvY6Q/edit#heading=h.1wyb3tcnqdxq)

[**Job Instructions 3**](https://docs.google.com/document/d/1dKG7qvuSGz4iKflH1uwZE_H3eBD1pEUQTcwwEVuvY6Q/edit#heading=h.cmwbmetrrrkh)

[**Why I am doing this 3**](https://docs.google.com/document/d/1dKG7qvuSGz4iKflH1uwZE_H3eBD1pEUQTcwwEVuvY6Q/edit#heading=h.r8i9jdxg2w7p)

[**Don't forget 3**](https://docs.google.com/document/d/1dKG7qvuSGz4iKflH1uwZE_H3eBD1pEUQTcwwEVuvY6Q/edit#heading=h.hwl1i9pvzgli)

[**Calculation of the averages 3**](https://docs.google.com/document/d/1dKG7qvuSGz4iKflH1uwZE_H3eBD1pEUQTcwwEVuvY6Q/edit#heading=h.9dlmdgzf5txh)

Enclosures

I have already given you the stockave.py file, which is a newer version of your script **yahoo\_2.py**. I made a few additions, mainly on the Google Sheets part of the script.

What this code does

This Python script automates the process of extracting stock market data from Yahoo Finance and updates a Google Sheet with the retrieved information. Here's a summary of the code:

Import required libraries and modules, such as **subprocess**, **datetime**, **gspread**, **selenium**, and **webdriver\_manager**.

1. Install necessary Python packages using subprocess.check\_call.
2. Configure the Chrome WebDriver with certain options (e.g., headless mode, ignoring certificate errors, etc.) for web scraping.
3. Define the Google Sheets API scopes and authenticate using a JSON key file.
4. Connect to a Google Sheet and select the 'Stock Basket' and 'MarketData' worksheets.
5. Iterate through the stocks listed in the 'Stock Basket' worksheet and extract historical data for each stock from Yahoo Finance.
6. Calculate the average closing prices for different timeframes (50-day, 20-day, etc.) using the extracted data.
7. Insert the extracted data and calculated averages into the 'MarketData' worksheet.
8. Apply conditional formatting to the worksheet based on a boolean rule.

The script starts by setting up the necessary dependencies and configurations for the Chrome WebDriver and Google Sheets API.

It then connects to the specified Google Sheet and iterates through a list of stock symbols in the 'Stock Basket' worksheet.

For each stock, the script:

* visits the stock's historical data page on Yahoo Finance and extracts the closing prices
* calculates the averages for different timeframes

It then appends this data to the 'MarketData' worksheet and applies conditional formatting to the worksheet based on a boolean rule.

Job Instructions

In this job, I need you to separate the existing script into two separate Python scripts.

The first script, named **acquire.py**, will be responsible for scraping stock market data from Yahoo Finance and storing it in a DataFrame.

The second script, **named process.py**, will process the DataFrame and insert the data into a Google Sheet. Additionally, I need you to calculate the 200-day average from the historical data, similar to the existing 20-day and 50-day averages.

The output columns are:

* Date
* Symbol
* CompanyName
* ClosingPrice
* TwoHundredDayAverage
* FiftyDayAverage
* TwentyDayAverage
* TwoHundredDayAverageLast
* FiftyDayAverageLast
* TwentyDayAverageLast

Why am I doing this?

I am separating concerns by dividing the work between a scraping specialist and a Google Sheets specialist.

This will also be helpful when I turn this into a web app, as having discrete functions will improve the overall design and organization of the code. I will replace Google Sheets with a database.

Don't forget

* Add a parser for arguments to pass the DataFrame between the two scripts.
* Remove libraries from **acquire.py** that only pertain to **process.py** (e.g., **gspread**, **oauth2client**, and **gspread\_formatting**).
* Add more comments throughout the code to explain the steps.
* Instead of printing '------> New Data Added to MarketData worksheet.', show the stock code being processed to indicate progress.
* Any other items that you think will help us debug and test this quicker.

Technical Items

Passing the DataFrame over

To pass the DataFrame from acquire.py to process.py, you can save the DataFrame to a CSV file in acquire.py and then read the CSV file in process.py.

Here's how you can do it:

* In acquire.py, after you have populated the DataFrame, save it to a CSV file:

df.to\_csv('stock\_data.csv', index=False)

* Now, in process.py, you can read the CSV file into a DataFrame:

import pandas as pd

…

df = pd.read\_csv('stock\_data.csv')

* Once you have read the CSV file into a DataFrame in process.py, you can create a function to process the DataFrame and insert the data into the Google Sheet:

def process\_dataframe(df):

    # Your existing code to insert the data into the Google Sheet goes here

if \_\_name\_\_ == "\_\_main\_\_":

    process\_dataframe(df)

The process\_dataframe() function in process.py accepts the DataFrame as an argument and processes it according to your existing code that inserts the data into the Google Sheet.

Calculation of the averages

To calculate the TwoHundredDayAverage and TwoHundredDayAverageLast, follow these steps, similar to the existing calculation for the FiftyDayAverage and FiftyDayAverageLast:

1. Initialize variables hd\_total and lhd\_total to store the sum of the closing prices for the 200-day range and the last 200-day range, respectively.
2. Iterate through the historical data and add the closing price to hd\_total for the first 200 days, and add the closing price to lhd\_total for days 2 to 201.
3. Calculate the TwoHundredDayAverage by dividing hd\_total by 200, and calculate the TwoHundredDayAverageLast by dividing lhd\_total by 200.
4. Add these values to the DataFrame before inserting it into the Google Sheet.