Part B Specification

# Completed work Part A

Imtiaz, before I get into the specification of the work to be done, I will just confirm what we have done already.

I have successfully completed testing your Python script called "Yahoo" which carries out the following steps:

1. The code installs necessary packages (pandas, selenium, webdriver\_manager, and numpy).
2. The code sets up a list of stock symbols and corresponding company names.
3. The code uses Selenium to scrape Yahoo Finance for each stock symbol, gathering the date, closing price, and data needed to calculate the 50-day and 20-day moving averages.
4. The code creates a DataFrame from the gathered data.
5. The code writes the DataFrame to an Excel file named "yahoo.xlsx".
6. The code prints out the number of extracted data after each iteration of the loop.
7. Finally, the code prints a message indicating that the extraction was successful.

Defect: Regarding the issue with the stock price column appearing as text in Excel (column D), it seems to be due to the presence of non-numeric characters and symbols in some of the scraped data. I recommend modifying the code to ensure that only numerical values are pushed to the cell values.

Perhaps this might solve that problem:

gc.loc[:, ['Date', 'Symbol','CompanyName','ClosingPrice','FiftyDayAverage','TwentyDayAverage','FiftyDayAverageLast','TwentyDayAverageLast','Check']] = gc.loc[:, ['Date', 'Symbol','CompanyName','ClosingPrice','FiftyDayAverage','TwentyDayAverage','FiftyDayAverageLast','TwentyDayAverageLast','Check']].apply(pd.to\_numeric, errors='ignore')

gc.to\_excel("yahoo.xlsx", index=False)

This code uses the apply() method to apply the pd.to\_numeric() function to all columns except for the 'Check' column, which appears to be a string column. The errors='ignore' parameter ensures that any cells that cannot be converted to a numeric format are left as they are.

Part B

I want you to create a different version of this script which carries out the following steps:

1. The code installs necessary packages (pandas, selenium, webdriver\_manager, and numpy).
2. The code reads the stock symbols and corresponding company names from the Google Sheet named "Stocks" and sheet named "Stock Basket", iterating from row two through to the last used row, and adds them to the array.
3. The code uses Selenium to scrape Yahoo Finance for each stock symbol in the array, gathering the date, closing price, and data needed to calculate the 50-day and 20-day moving averages.
4. The code creates a DataFrame from the gathered data.
5. The code writes the DataFrame to the first available row on a Sheet named "MarketData" in the Google Sheet "Stocks". It will not overwrite existing data.
6. The code checks for non-numeric characters and symbols in the stock price column and formats it as a number in the Excel file.
7. The code prints out the number of extracted data after each iteration of the loop.
8. Finally, the code prints a message indicating that the extraction was successful.

Here's a code snippet that shows how to authenticate and access a Google Spreadsheet called "Stocks" and read data from the sheet called "Stock Basket".

Before you get started, make sure you have created a project in the Google Cloud Console, enabled the Google Sheets API, and created a service account. You will also need to download the service account JSON file, which contains the credentials that allow your code to authenticate with Google Sheets API.

import gspread

from oauth2client.service\_account import ServiceAccountCredentials

# Authenticate with Google Sheets API

scope = ['https://www.googleapis.com/auth/spreadsheets']

creds = ServiceAccountCredentials.from\_json\_keyfile\_name('path/to/service\_account.json', scope)

client = gspread.authorize(creds)

# Access the Google Spreadsheet called "Stocks"

spreadsheet\_name = "Stocks"

sheet\_name = "Stock Basket"

worksheet = client.open(spreadsheet\_name).worksheet(sheet\_name)

# Read data from the worksheet into a dictionary called "stocks\_dict"

stocks\_dict = {}

for row in worksheet.get\_all\_records():

symbol = row["Symbol"]

name\_url = row["Company Name"]

stocks\_dict[symbol] = name\_url

print(stocks\_dict)

This code authenticates with the Google Sheets API using a service account JSON file, then accesses the Google Spreadsheet called "Stocks" and the sheet called "Stock Basket". It then reads data from the worksheet into a dictionary called "stocks\_dict", which is similar to the "stocks\_dict" created in the original code.

Note that you will need to replace "path/to/service\_account.json" with the actual path to your service account JSON file. Also, make sure you have installed the necessary libraries for working with Google Sheets API, namely gspread and oauth2client. You can install them using pip:

pip install gspread oauth2client

We can directly write the data frame to the Google Sheet. Here's a code snippet that writes the data frame to the first available row on the sheet named "MarketData" in the Google Sheet "Stocks". It will not overwrite existing data.

# authenticate and access the Google Sheet

# (assuming credentials are already set up)

import gspread

from oauth2client.service\_account import ServiceAccountCredentials

scope = ["https://www.googleapis.com/auth/spreadsheets"]

creds = ServiceAccountCredentials.from\_json\_keyfile\_name("credentials.json", scope)

client = gspread.authorize(creds)

sheet\_name = "MarketData"

worksheet = client.open("Stocks").worksheet(sheet\_name)

# get the last row index and add 1 to find the first empty row

last\_row\_index = len(worksheet.get\_all\_values())

first\_empty\_row\_index = last\_row\_index + 1

# write the data frame to the first empty row

worksheet.insert\_rows(gc.values.tolist(), first\_empty\_row\_index)

Note that in the insert\_rows() method, we pass in the values of the data frame as a list of lists using the .values.tolist() method. This is because the insert\_rows() method expects a list of rows, where each row is represented as a list of values.

I hope that is clear enough to complete this job will stop please contact me once you have read this specification thank you.