CS 620–Introduction to Data Science and Analytics, Spring 2022, HW3

You have just been hired as an analyst for an investment firm. Your first assignment is to analyze data for stocks in the S&P 500. The S&P 500 is a stock index that contains the 500 largest publicly traded companies.

You have been given two sources of data to work with. The first is an XML file that contains the Symbol (ticker), company name, sector, and industry for every stock in the S&P 500, as of summer 2016.
The second is a CSV file that contains pricing information for stocks in the S&P 500 between August 2009 and August 2010. There is one row in the CSV file for every stock, on every date that the market was open. Each row contains the date as a string, the stock's ticker, the day's opening price, the day's high price, the day's low price, the day's closing price, and the volume traded that day.

Use the provided files SP500_ind.csv and SP500_symbols.xml and the starter code.
Implement using python and associated libraries (NumPy, Pandas etc.,).

- Complete the following functions in the starter code.

```
def ticker_find(xml_root, ticker):
        """     This function takes in the root of the xml dataset and a
                Symbol (ticker).  Return the name of the ticker. If the name is
                not available in the given SP500_symbols.xml then this function will assign
                "No data in SP500" for the ticker name
                Ex: for ticker "A", the function returns Agilent Technologies Inc
        """
def calc_avg_close(csv_data_sp500, ticker):
        """     This function takes in the csv_data_sp500 and a ticker.
                Return the average closing price for the stock as a float value.
        """
def calc_vwap(csv_data_sp500, ticker):
        """This function takes in the csv_data_sp500 and a ticker.  Return the volume weighted
        average price (VWAP) of the stock as a float.  In order to do this, first find the average
        price of the stock on each day.  Then, multiply that price with the volume on that day.
        Take the sum of these values.  Finally, divide that value by the sum of all the volumes.
        (hint: average price for each day = (high + low + close)/3)
        """
def calc_sma(company_data, days):
        """
              Compute Simple Moving Average for the given days for a given company. See
              below for instructions.
        """

def calc_cma(company_data):
        """
              Compute Cumulative Moving Average for for a given company. See below for
              instructions.
        """

def display_plt(company_data):
        """
              Display the required chart. See below for instructions.
        """
```

CS 620–Introduction to Data Science and Analytics, Spring 2022, HW3

1. To perform these calculations, you should call above functions in a logical order, with the appropriate parameters. Use the given driver code. Warning: **DO NOT CHANGE** the driver code. You are free to experiment with the driver code, but don't forget to revert it back to the original form. It's advised to run the program during development for a single company (for example, ticker: "JAVA") until you get the correct results.

   Note that stocks move in and out of the S&P 500. Some stocks may be represented in the CSV file, but not in the XML file (and vice-versa). Display "No data in SP500" for the names of these tickers. For example, for ticker JAVA,

   Ticker: JAVA Name: No data in SP500 Avg: 9.01 VWAP: 9.00

2. Calculate the Simple Moving Average (SMA) for a company for a given number of days (e.g., SMA-3 for 3 days, SMA-10 for 10 days).
   Simple Moving Average is one of the core technical indicators used by traders and investors for the technical analysis of a stock. Simple moving average is calculated by adding the "Close" price of last N number of days (window size N) and then dividing it by the number of days.

   $$SMA = (\text{Sum of ticker Close value over the past N days}) / (N)$$

   Note that Pandas dataframe.rolling() function provides the feature of rolling window calculations given window=N parameter. You don't need to display the SMA values of this function. Create a new DataFrame column called 'SMA' and return the DataFrame.

3. Calculate the Cumulative Moving Average (CMA) for a company.
   The Cumulative Moving Average is the mean of all the previous closing values up to the current value (n). While calculating CMA we don't have any fixed size of the window. The size of the window (n) keeps on increasing as time passes.

   $$CMA = (C_1 + C_2 + \ldots + C_n) / (n)$$

   Hint: use the dataframe.expanding().mean() to calculate the CMA. You don't need to display the CMA values of this function. Create a new DataFrame column called 'CMA' and return the DataFrame.

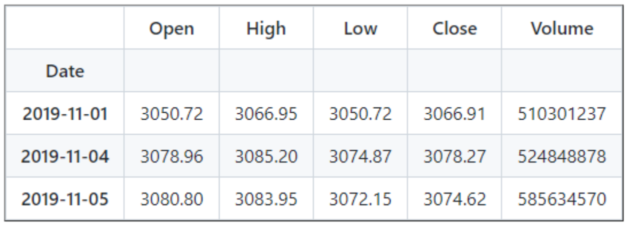
4. Generate OHLC (Open, High, Low, Close) chart using mplfinance (https://github.com/matplotlib/mplfinance) and display the data for the ticker "JAVA". See the corresponding chart image below. Your chart should display the SMA (blue line) and CMA (orange line) in the same plot.

   Don't forget to make necessary installations before you start using mplfinance

   ```
   pip install --upgrade mplfinance
   ```

   Note that mplfinance requires a DataFrame in a certain format including Date as the index. Convert the date in the given format using to_datetime function (https://pandas.pydata.org/docs/reference/api/pandas.to_datetime.html) in pandas with format='%Y%m%d'

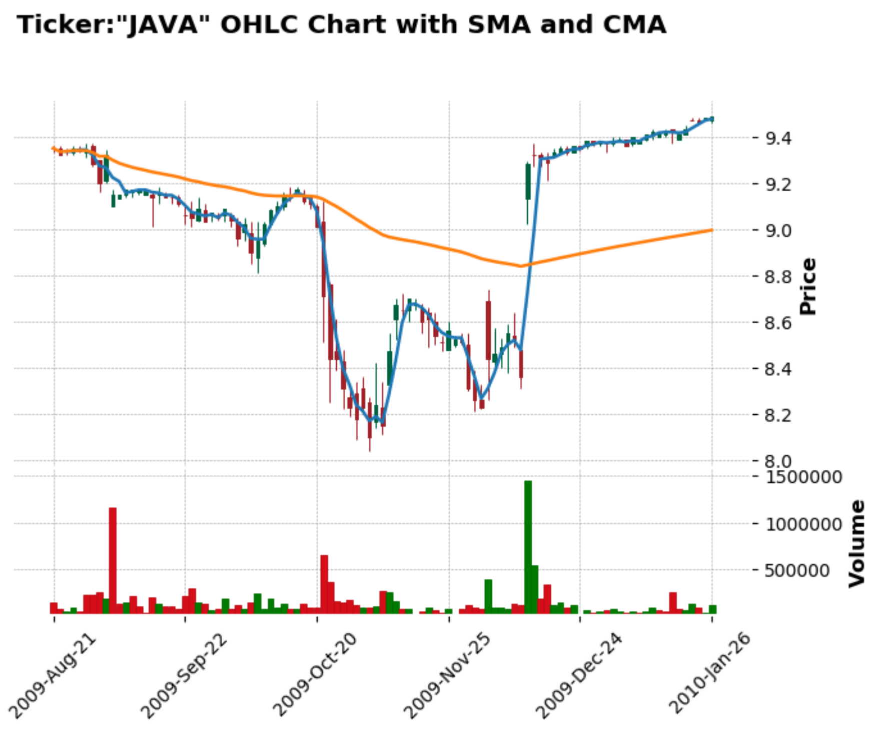
For example, the resultant DataFrame should looks like below,



	Open	High	Low	Close	Volume
Date					
2019-11-01	3050.72	3066.95	3050.72	3066.91	510301237
2019-11-04	3078.96	3085.20	3074.87	3078.27	524848878
2019-11-05	3080.80	3083.95	3072.15	3074.62	585634570

Additional information on how to add options for volume, candle type, styles, and extra plots (for SMA and CMA) are available at
https://github.com/matplotlib/mplfinance/blob/master/examples/addplot.ipynb

Your resultant chart should look like below chart for ticker "JAVA",



5. You **CANNOT USE** any regular python loops inside calc_avg_close, calc_vwap, calc_sma, calc_cma, and display_plt functions. Instead use DataFrame techniques learned in Pandas.

What to turn in: Submit your .py file to Blackboard.

> **Lastname-hw3.py** should contain the following information at the top:
> CS620
> HW3
> @author: <Your Name and UIN>