Analazing the Casablanca Stock Exchange data project

Part I : Data Collection

The stock data from Casablanca's Stock Exchange will be the primary focus of our project, and will be the foundation upon which we will gather data and analyze it.

First of all, we go to Casablanca's Stock Exchange website (https://www.casablanca-bourse.com/bourseweb/index.aspx) and we choose the sector "Construction and building materials".

One of the instruments we have had choosen is LAFARGEHOLCIM MAR (https://www.casablanca-bourse.com/bourseweb/en/Company.aspx?codeValeur=3800&cat=7).You can see down bellow the char showing the history from 01/01/2020 till 01/12/2022.

We downloaded the data. The data was stored in a .aspx file.

Note that,ASPX stands for Active Server Pages Extended. A file with .aspx extension is a webpage generated using Microsoft ASP.NET framework running on web servers.

Task 1 :

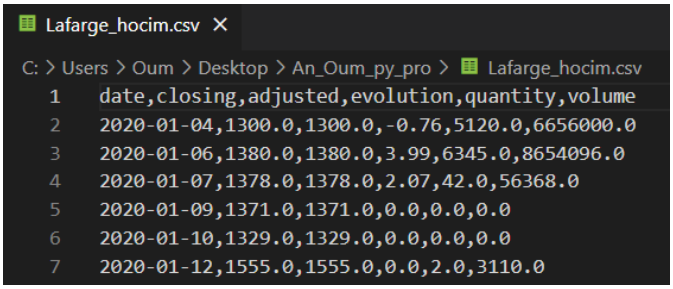
- Create a function that tikes the file we downloaded and returns a CSV file with the following columns : *date,closing,adjusted,evolution,quantity and volume*.

In [4]:
```
from bs4 import BeautifulSoup
import pandas as pd
import plotly.graph_objs as go
import plotly.offline as pyo
from datetime import datetime
import warnings
warnings.filterwarnings("ignore")
import numpy as np

#We have taken data from the sector "Construction and building materials".
#We have choosen the following Instruments : Afric Industries SA, Colorado,
#Jet Contractors, Lafargeholcim MAR and TGCC S.A a.

def scrapper(file_name):
#file_name is the name of the instrument.
#The dictionary "dic_files" contains the names of instrument as keys and the value as a path to the file containing the data.

    dic_files={"Africa_industries":"data_py_pro/Africa_industries.aspx",
           "Colorado":"data_py_pro/colorado.aspx",
           "Jet_contractors":"data_py_pro/Jet_contractors.aspx",
           "Lafarge_hocim":"data_py_pro/Lafarge_holcim.aspx",
           "Tgcc":"data_py_pro/Tgcc.aspx"}

#we used "with open() as" so it closes automatically the file.
    with open(dic_files[file_name],"r") as myfile:

#We read the data in the file and we store it in a variable, using globals() to make the string
#"file_name" become a variable  called file_name.
        globals()[file_name]=myfile.read()

#We store the content after applying BeautifulSoup in final_file
        final_file=BeautifulSoup(globals()[file_name],'lxml')

#Since we have in the original file ',' we should replace it with '.' to avoid any confusion.
    final_file1=[]
    for f in final_file.find_all("span"):
            f=f.get_text().replace(",",".")
            final_file1.append(f)

    list_of_columns = ["date","closing","adjusted","evolution","quantity","volume"]
    data=[]
#The jum is 6 since we have six columns. We used len(final_file1) because every file has diffrent size.
    for i in range(0,len(final_file1),6):
        data.append([final_file1[i],
        final_file1[i+1],
        final_file1[i+2],
        final_file1[i+3],
        final_file1[i+4],
        final_file1[i+5]])

    data_df= pd.DataFrame(data,columns=list_of_columns)
#Converting the values in the "date" column of a DataFrame data_df to datetime objects.
    data_df["date"]= pd.to_datetime(data_df["date"])
#Concerting the rest of the columns into float objects.
    data_df[["closing","adjusted","evolution","quantity","volume"]]= data_df[["closing","adjusted","evolution","quantity","volume
    data_df = data_df.sort_values('date', ascending=True)
#The last step is to convert our dataframe into a csv file.
    data_csv= data_df.to_csv("{}.csv".format(file_name),index=False)

    return data_csv
```

In [5]:
```
%%timeit
#scrapper("Jet_contractors")
#scrapper("Colorado")
#scrapper("Africa_industries")
#scrapper("Tgcc")

scrapper("Lafarge_hocim")
```

```
347 ms ± 24.8 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Here is a snipt of the file **Lafarge_hocim.csv**.



Part II : Data Processing

After collecting the data, we will be in this part analazing it.

Tasks :

Task 2 :

- Create a class Stock that will represent stock in our project.

Task 3 :

- Creat a method in Stock responsible for visualisation.

Task 4 :

- Inside Stock, create the following instance methods:
  - A method that returns the maximum value in the Dataframe.
  - A method that returns the minimum value in the Dataframe.
  - A method that returns the maximum volume in the Dataframe.
  - A method that returns the maximum quantity in the Dataframe.

Task 6

- Inside Stock, create a method that calculates the momentum for the given stock for any value of N.

Task 7 :

- Inside Stock, create a method that calculates the simple moving average for the given stock.

Task 8 :

- Extend the method you implemented previously for visualization to include the two indicators you just implemented.

In [16]:

```
class Stock():
    #Task 2
    def __init__(self,name):
        '''The constructor of the class Stock. It has as an attributes:
        - name : which will be the name of the instrument.
        - data : It is a Dataframe containing our data that we got using the method scrapper(name).'''
        self.name = name
        scrapper(name)
        self.data=pd.read_csv(name+".csv")

    #Task 3
    def visualize(self):
        '''visualize() plot our data in the y axis we have the value and volume. For the x-axis we have the dates.'''
        graph1 = go.Scatter(x=self.data.date, y=self.data.closing, name='Value')
        graph2 = go.Scatter(x=self.data.date, y=self.data.volume, name='Volume', yaxis='y2')
        data = [graph1, graph2]
        layout = go.Layout(title=self.name, xaxis=dict(title='Date'), yaxis=dict(title='Value'),
                           yaxis2=dict(title='Volume', overlaying='y', side='right'))
        layout.update(shapes=
        [
            dict(
                type='rect',
                x0='2022-01-01',
                y0=0,
                x1='2022-12-31',
                y1=self.data["closing"].max(),
                opacity=0
                )
        ])
        fig = go.Figure(data=data, layout=layout)
        pyo.iplot(fig)
    #Task 4
    def maximum_value(self):
        '''maximum_value() gives the maximum value in the stock.'''
        return self.data["closing"].max()

    def minimum_value(self):
        '''minimum_value() gives the minimum value in the stock.'''
        return self.data["closing"].min()

    def maximum_volume(self):
        '''maximum_volume() gives the maximum volume in the stock.'''
        return self.data["volume"].max()

    def  maximum_quantity(self):
        '''maximum_quantity() gives the maximum quantity in the stock.'''
        return self.data["quantity"].max()

    #Task 6
    def momentum(self,N=list):
        '''Method momentum computes values from a given list N and adds len(N) as columns to the original DataFrame.'''
        self.N=N
        for i in self.N:
            self.data["momentum"+"={}".format(i)]=np.nan
            for j in range(0,len(self.data)):
                if j >= i :
                    self.data["momentum"+"={}".format(i)][j] = self.data["closing"][j]-self.data["closing"][j-i]
        return self.data
    #Task 7
    def sma(self,window):
        '''Method sma computes value of simple moving avreage of a window taken as parameter and
        add it as a column to our original DataFrame.'''
        self.window=window
        void_nparray = np.full((self.window-1),np.nan)
        weights = np.repeat(1.0 , self.window)/ self.window
        sma_unfinished = np.convolve(self.data["closing"],weights,"valid")
        sma = np.concatenate((void_nparray, sma_unfinished), axis=0)
        self.data["sma"+"={}".format(self.window)]=pd.DataFrame(sma)
        return self.data
    #Task 8
    def visualize_mom_sma(self):
        '''Method visualize_mom_sma that looks for the value of sma in the list of the momentums in order to compare
        the evolution of both plots following the same parameter.'''
        graph1 = go.Scatter(x=self.data["date"], y=self.data["momentum"+"={}".format(self.window)]
                            , name="Momentum"+"={}".format(self.window))
        graph2 = go.Scatter(x=self.data["date"], y=self.data["sma"+"={}".format(self.window)],
                            name="sma"+"={}".format(self.window), yaxis='y2')
        data = [graph1, graph2]
        layout = go.Layout(title=self.name, xaxis=dict(title='Date'),yaxis=dict(title="momentum"+"={}".format(self.N[0])),
                           yaxis2=dict(title="sma"+"={}".format(self.window), overlaying='y', side='right'))
        layout.update(shapes=[dict(type='rect',x0='2022-01-01',y0=0,x1='2022-12-31',y1=self.data["closing"].max(),opacity=0)])
        fig = go.Figure(data=data, layout=layout)
```

```
pyo.iplot(fig)
```

Task 2 :

Let's creat an object from the class stock for the instrument '**Lafargeholcim MAR**'.

In [7]:
```
lafargeholcim = Stock("Lafarge_hocim")
lafargeholcim.data
```

Out[7]:

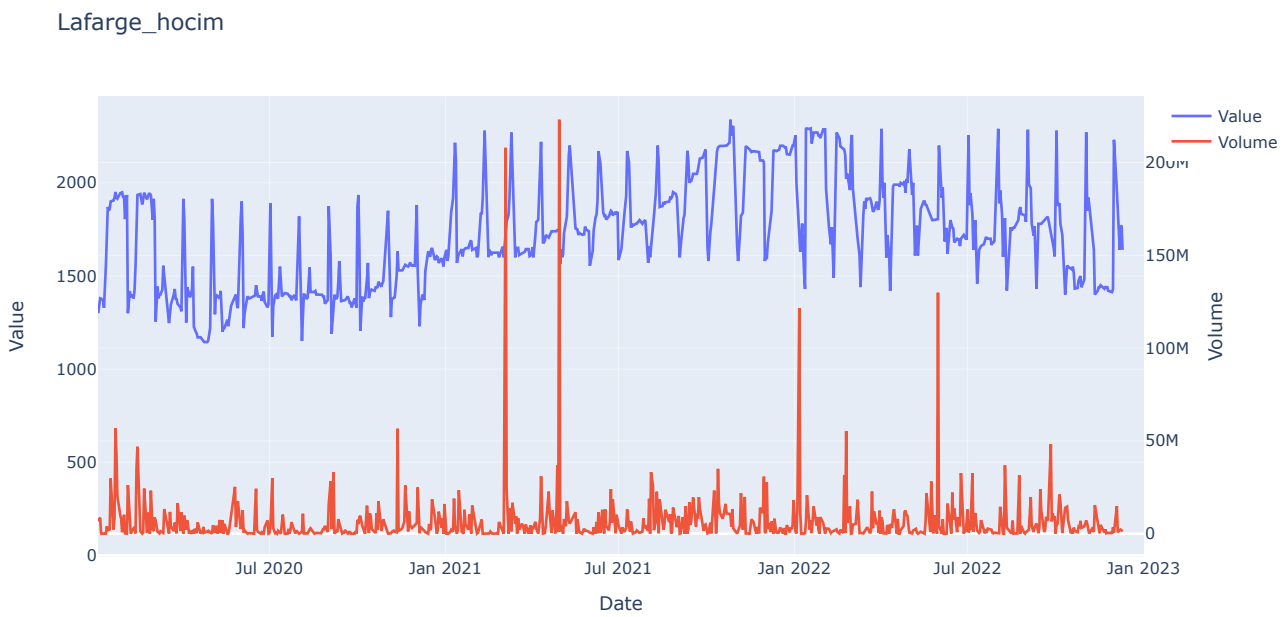
	date	closing	adjusted	evolution	quantity	volume
0	2020-01-04	1300.0	1300.0	-0.76	5120.0	6656000.0
1	2020-01-06	1380.0	1380.0	3.99	6345.0	8654096.0
2	2020-01-07	1378.0	1378.0	2.07	42.0	56368.0
3	2020-01-09	1371.0	1371.0	0.00	0.0	0.0
4	2020-01-10	1329.0	1329.0	0.00	0.0	0.0
...	...	...	...	...	...	...
727	2022-12-05	1860.0	1860.0	-1.59	392.0	729215.0
728	2022-12-07	1640.0	1640.0	2.31	1255.0	2058425.0
729	2022-12-08	1770.0	1770.0	1.67	1275.0	2256375.0
730	2022-12-09	1766.0	1766.0	0.00	1002.0	1769596.0
731	2022-12-10	1640.0	1640.0	0.00	701.0	1149640.0

732 rows × 6 columns

Task 3 :

Let's use the method visualize() to see our data as a plot.

In [8]:
```
lafargeholcim.visualize()
```



Task 4 :

Let's return the maximum closing from the stock of **__Lafargeholcim MAR__.**

In [9]:
```
lafargeholcim.maximum_value()
```

Out[9]:
```
2339.0
```

Let's return the minimum closing from the stock of **Lafargeholcim MAR**.

In [10]:
```
lafargeholcim.minimum_value()
```

Out[10]:
```
1145.0
```

Let's return the maximum volume from the stock of **Lafargeholcim MAR**.

In [11]:
```
lafargeholcim.maximum_volume()
```

Out[11]:
```
223103485.0
```

Let's return the maximum quantity from the stock of **Lafargeholcim MAR**.

In [12]:
```
lafargeholcim.maximum_quantity()
```

Out[12]:
```
127853.0
```

Task 6 :

Let's calculate the momentum for the stock of __Lafargeholcim MAR__ for a list of from (1 day to 5 days) and add it to DataFrame .

In [13]:
```
lafargeholcim.momentum([1,2,3,4,5])
```

Out[13]:

	date	closing	adjusted	evolution	quantity	volume	momentum=1	momentum=2	momentum=3	momentum=4	momentum=5
0	2020-01-04	1300.0	1300.0	-0.76	5120.0	6656000.0	NaN	NaN	NaN	NaN	NaN
1	2020-01-06	1380.0	1380.0	3.99	6345.0	8654096.0	80.0	NaN	NaN	NaN	NaN
2	2020-01-07	1378.0	1378.0	2.07	42.0	56368.0	-2.0	78.0	NaN	NaN	NaN
3	2020-01-09	1371.0	1371.0	0.00	0.0	0.0	-7.0	-9.0	71.0	NaN	NaN
4	2020-01-10	1329.0	1329.0	0.00	0.0	0.0	-42.0	-49.0	-51.0	29.0	NaN
...	...	...	...	...	...	...	...	...	...	...	...
727	2022-12-05	1860.0	1860.0	-1.59	392.0	729215.0	-100.0	-371.0	431.0	447.0	447.0
728	2022-12-07	1640.0	1640.0	2.31	1255.0	2058425.0	-220.0	-320.0	-591.0	211.0	227.0
729	2022-12-08	1770.0	1770.0	1.67	1275.0	2256375.0	130.0	-90.0	-190.0	-461.0	341.0
730	2022-12-09	1766.0	1766.0	0.00	1002.0	1769596.0	-4.0	126.0	-94.0	-194.0	-465.0
731	2022-12-10	1640.0	1640.0	0.00	701.0	1149640.0	-126.0	-130.0	0.0	-220.0	-320.0

732 rows × 11 columns

Task 7 :

Let's compute the simple moving average for the stock of __Lafargeholcim MAR__ for a 3 days value and add it to DataFrame .

In [14]:
```
lafargeholcim.sma(3)
```

Out[14]:

	date	closing	adjusted	evolution	quantity	volume	momentum=1	momentum=2	momentum=3	momentum=4	momentum=5	sma=3
0	2020-01-04	1300.0	1300.0	-0.76	5120.0	6656000.0	NaN	NaN	NaN	NaN	NaN	NaN
1	2020-01-06	1380.0	1380.0	3.99	6345.0	8654096.0	80.0	NaN	NaN	NaN	NaN	NaN
2	2020-01-07	1378.0	1378.0	2.07	42.0	56368.0	-2.0	78.0	NaN	NaN	NaN	1352.666667
3	2020-01-09	1371.0	1371.0	0.00	0.0	0.0	-7.0	-9.0	71.0	NaN	NaN	1376.333333
4	2020-01-10	1329.0	1329.0	0.00	0.0	0.0	-42.0	-49.0	-51.0	29.0	NaN	1359.333333
...	...	...	...	...	...	...	...	...	...	...	...	...
727	2022-12-05	1860.0	1860.0	-1.59	392.0	729215.0	-100.0	-371.0	431.0	447.0	447.0	2017.000000
728	2022-12-07	1640.0	1640.0	2.31	1255.0	2058425.0	-220.0	-320.0	-591.0	211.0	227.0	1820.000000
729	2022-12-08	1770.0	1770.0	1.67	1275.0	2256375.0	130.0	-90.0	-190.0	-461.0	341.0	1756.666667
730	2022-12-09	1766.0	1766.0	0.00	1002.0	1769596.0	-4.0	126.0	-94.0	-194.0	-465.0	1725.333333
731	2022-12-10	1640.0	1640.0	0.00	701.0	1149640.0	-126.0	-130.0	0.0	-220.0	-320.0	1725.333333

732 rows × 12 columns

Task 8 :

We compare the 2 DataFrames data["momentum=3"] and data["sma=3"], in order to compare momentum of 3 days to sma of 3 days.

In [17]:
```
lafargeholcim.visualize_mom_sma()
```