Name : Sarvagya Singh **SAPID : 60009200030**

Experiment 1

Aim: Analysis of Volume Data and Historical Price for one Stock on Yahoo Finance Dataset.

Stock Market Data: Obtaining Data, Visualization & Analysis in Python

Objective:

- Get historical data for stocks
- Plot the stock market data and analyse the performance
- Get the fundamental, futures and options data

Theory:

Historical Stock Price Data

Stocks are very volatile instruments and therefore very important to thoroughly analyse the price behaviour before making any trading decisions. Hence fetching and analysing the prices is crucial. The stock data can be downloaded from different packages such as yahoo finance, quandl and alpha vantage. This data can be fetched the daily and minute level from yahoo finance. It is essential to understand data structures, data analysis, dealing with financial data, and for generating trading signals. This experiment will build learning on how to fetch various data like pricing data of stocks, fundamental.

Yahoo Finance Market Data

Yahoo Finance: One of the first sources to get historical daily price-volume stock market data is Yahoo finance. The yfinance module has the download method which can be used to download the stock market data. To get stock market data for different geographies, use the ticker symbol on Yahoo finance.

Intraday or Minute Frequency Stock Data

yfinance module can be used to fetch the minute level stock market data. It returns the stock market data for the last 7 days.
It takes the following parameters:

1. **ticker:** The name of the tickers for stock market data. stock market data for multiple tickers can be separated by space
2. **period:** The number of days/month of stock market data required. The valid frequencies are 1d, 5d, 1mo, 3mo, 6mo, 1y, 2y, 5y, 10y, ytd, max
3. **interval:** The frequency of the stock market data. The valid intervals are 1m, 2m, 5m, 15m, 30m, 60m, 90m, 1h, 1d, 5d, 1wk, 1mo, 3mo

Minute level data

Through yfinance, fetch the data of minute frequency. download for other frequency by just tweaking the interval parameter on line no 8 below. Following values are supported in the interval: 1m, 5m, 15m, 30m, 60m.

This is used to analyse data, create a trading strategy and analyse the performance of the strategy using the pyfolio package. It computes the Sharpe ratio, Sortino ratio, maximum drawdowns and many other metrics.

List of stock tickers in S&P500

The next logical requirement is how to get the tickers that make up favourite index or are listed on a specific exchange. For that, get the tickers from the Wikipedia page and use those tickers to get data from yfinance.

Resample Stock Data

Convert 1-minute data to 1-hour data or Resample Stock Data

During strategy modelling, one might be required to work with a custom frequency of stock market data such as 15 minutes or 1 hour or even 1 month. If minute level data is available, then one can easily construct the 15 minutes, 1 hour or daily candles by resampling them. So, there is no need to purchase them separately. In this case, use the pandas resample method to convert the stock market data to the frequency of your choice. The implementation of these is shown below where a 1-minute frequency data is converted to 10-minute frequency data. The first step is to define the dictionary with the conversion logic. For example, to get the open value the first value will be used, to get the high value the maximum value will be used and so on. Yahoo finance has limited set of minute level data.

Stock Market Data Visualization and Analysis

After having the stock market data, the next step is to create trading strategies and analyse the performance. The ease of analysing the performance is the key advantage of the Python.

Analyse the cumulative returns, drawdown plot, different ratios such as

- Sharpe ratio,
- Sortino ratio, and
- Calmar ratio.

Conclusion

To be able to use the Python codes to fetch the stock market data of your favourites stocks, build the strategies using this stock market data and analyse this data.

<u>Lab Experiment to be done by students:</u>

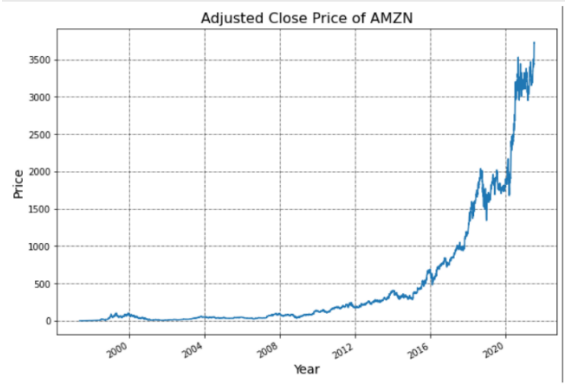
1. Fetch Historical Price & Volume Data for one Stock
2. Analysis of Volume Data and Historical Price for one Stock.
3. Implementation of time period specific setting.
4. enforce a Frequency Setting Frequency Settings (Intraday)
5. Implement financial model for Stocks Splits and Dividend Prediction.

6. Export yahoo finance data CSV/Excel
7. Importing financial Data for multiple Stocks.

Practice Problems:

Data for single stocks

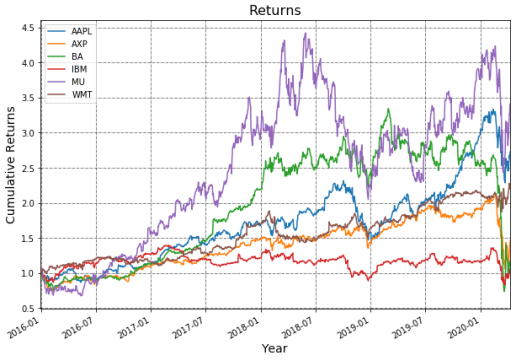
1. Get data for AAPL from 2016 to 2019 and plot the adjusted closing price of the data.
2. Improve the plot by resizing, giving appropriate labels and adding grid lines for better readability.



Data for multiple stocks

3. Fetch the data of multiple stocks and store it in a dataframe data. Then calculate the daily returns and plot the cumulative returns of all the stock prices using matplotlib package.

Date	AAPL	WMT	IBM	MU	BA	AXP
2014-12-31	102.11	76.12	130.81	35.01	114.15	86.39
2015-01-02	101.14	76.14	132.13	34.75	114.12	86.37
2015-01-05	98.29	75.92	130.05	33.78	113.33	84.08
2015-01-06	98.30	76.50	127.25	32.87	111.99	82.29
2015-01-07	99.68	78.53	126.42	32.10	113.73	84.09



Minute level Data Analysis

4. Fetch the data at minute frequency : fetch the stock market data for MSFT for the past 5 days of 1-minute frequency. Get daily and minute level historical stock data using yahoo finance. And get the list of stock tickers in S&P500.

```
[**********100%**********]  1 of 1 completed
```

Datetime	Open	High	Low	Close	Volume
2021-07-02 09:30:00-04:00	272.820007	273.070007	272.500000	273.070007	775960
2021-07-02 09:31:00-04:00	273.029999	273.100006	272.929993	273.089996	40240
2021-07-02 09:33:00-04:00	273.769989	273.769989	273.640015	273.769989	439991
2021-07-02 09:34:00-04:00	274.000000	274.040009	274.000000	274.040009	192699
2021-07-02 09:35:00-04:00	274.000000	274.190002	273.730011	273.859985	144486

Get stock market data for multiple tickers

5. get the tickers that make up your favourite index or are listed on a specific exchange.

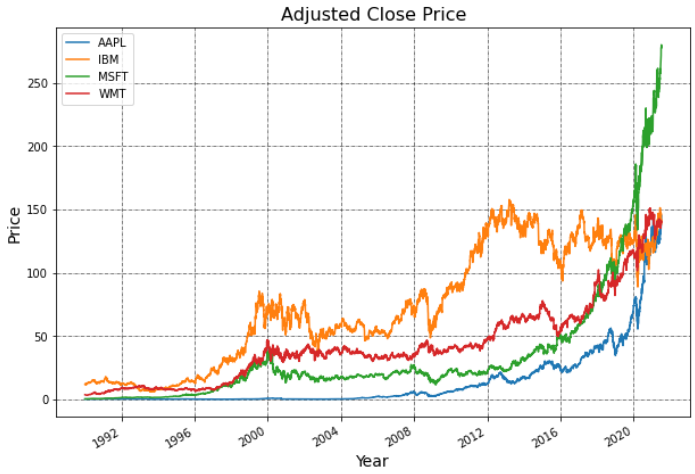
6. To get the stock market data of multiple stock tickers, create a list of tickers and call the **yfinance download method** for each stock ticker.

```
  Symbol             Security SEC filings             GICS Sector  \
0    MMM           3M Company     reports             Industrials
1    ABT  Abbott Laboratories     reports             Health Care
2   ABBV          AbbVie Inc.     reports             Health Care
3   ABMD              Abiomed     reports             Health Care
4    ACN            Accenture     reports  Information Technology

                GICS Sub-Industry    Headquarters Location Date first added  \
0        Industrial Conglomerates      St. Paul, Minnesota       1976-08-09
1           Health Care Equipment  North Chicago, Illinois       1964-03-31
2                  Pharmaceuticals  North Chicago, Illinois       2012-12-31
3           Health Care Equipment   Danvers, Massachusetts       2018-05-31
4  IT Consulting & Other Services          Dublin, Ireland       2011-07-06
```

Date	AAPL	IBM	MSFT	WMT
1990-01-02	0.267212	11.971801	0.390554	3.768927
1990-01-03	0.269005	12.078699	0.392754	3.768927
1990-01-04	0.269903	12.216123	0.404306	3.748934
1990-01-05	0.270799	12.185585	0.394405	3.708946
1990-01-08	0.272592	12.261941	0.400455	3.758933

7. Plot all Close Prices



S&P 500 Stock Tickers

8. analyse the stock market data for all the stocks which make up S&P 500. It gets the list of stocks from the Wikipedia page and then fetches the stock market data from yahoo finance.
9. Resample Stock Data: Convert 1-minute data to 1-hour data or Resample Stock Data

```
[**********100%**********]  1 of 1 completed
```

Datetime	Open	High	Low	Close	Volume
2021-07-02 09:30:00-04:00	272.820007	274.570007	272.500000	274.459991	2064942
2021-07-02 09:40:00-04:00	274.359985	275.179993	274.179993	275.000000	1198341
2021-07-02 09:50:00-04:00	275.000000	275.056091	274.709991	274.750000	646755
2021-07-02 10:00:00-04:00	274.739990	275.039886	274.434998	275.015015	672251
2021-07-02 10:10:00-04:00	275.010010	275.049988	274.559998	274.720001	405374

12. Create a simple buy and hold strategy for illustration purpose with four stocks namely:
    - **Apple**
    - **Amazon**

- **Microsoft**
- **Walmart**

Analyse the performance, using the **pyfolio tear sheet** .

Start date	2016-07-12
End date	2021-07-09
Total months	59

	Backtest
Annual return	36.4%
Cumulative returns	371.8%
Annual volatility	22.1%
Sharpe ratio	1.52
Calmar ratio	1.51
Stability	0.97
Max drawdown	-24.1%
Omega ratio	1.34
Sortino ratio	2.25
Skew	-0.02
Kurtosis	9.15
Tail ratio	0.93
Daily value at risk	-2.6%

Sarvagya Singh

60009200030 -- K1

FMC -- lab1

In [ ]:
```
pip install yfinance
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting yfinance
  Downloading yfinance-0.2.12-py2.py3-none-any.whl (59 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 59.2/59.2 KB 426.7 kB/s eta 0:00:00
Requirement already satisfied: pandas>=1.3.0 in /usr/local/lib/python3.8/dist-packages (from yfinance) (1.3.5)
Collecting beautifulsoup4>=4.11.1
  Downloading beautifulsoup4-4.11.2-py3-none-any.whl (129 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 129.4/129.4 KB 5.7 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.8/dist-packages (from yfinance) (1.22.4)
Collecting requests>=2.26
  Downloading requests-2.28.2-py3-none-any.whl (62 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.8/62.8 KB 3.8 MB/s eta 0:00:00
Requirement already satisfied: lxml>=4.9.1 in /usr/local/lib/python3.8/dist-packages (from yfinance) (4.9.2)
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.8/dist-packages (from yfinance) (0.0.11)
Collecting cryptography>=3.3.2
  Downloading cryptography-39.0.1-cp36-abi3-manylinux_2_28_x86_64.whl (4.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.2/4.2 MB 13.0 MB/s eta 0:00:00
Collecting frozendict>=2.3.4
  Downloading frozendict-2.3.5-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (111 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 111.2/111.2 KB 3.7 MB/s eta 0:00:00
Requirement already satisfied: appdirs>=1.4.4 in /usr/local/lib/python3.8/dist-packages (from yfinance) (1.4.4)
Collecting html5lib>=1.1
  Downloading html5lib-1.1-py2.py3-none-any.whl (112 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 112.2/112.2 KB 2.7 MB/s eta 0:00:00
Requirement already satisfied: pytz>=2022.5 in /usr/local/lib/python3.8/dist-packages (from yfinance) (2022.7.1)
Collecting soupsieve>1.2
  Downloading soupsieve-2.4-py3-none-any.whl (37 kB)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.8/dist-packages (from cryptography>=3.3.2->yfinance) (1.15.1)
Requirement already satisfied: six>=1.9 in /usr/local/lib/python3.8/dist-packages (from html5lib>=1.1->yfinance) (1.15.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.8/dist-packages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.8/dist-packages (from pandas>=1.3.0->yfinance) (2.8.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests>=2.26->yfinance) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests>=2.26->yfinance) (2022.12.7)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests>=2.26->yfinance) (1.24.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.8/dist-packages (from requests>=2.26->yfinance) (3.0.1)
Requirement already satisfied: pycparser in /usr/local/lib/python3.8/dist-packages (from cffi>=1.12->cryptography>=3.3.2->yfinance) (2.21)
Installing collected packages: soupsieve, requests, html5lib, frozendict, cryptography, beautifulsoup4, yfinance
  Attempting uninstall: requests
    Found existing installation: requests 2.25.1
    Uninstalling requests-2.25.1:
      Successfully uninstalled requests-2.25.1
  Attempting uninstall: html5lib
    Found existing installation: html5lib 1.0.1
    Uninstalling html5lib-1.0.1:
      Successfully uninstalled html5lib-1.0.1
  Attempting uninstall: beautifulsoup4
    Found existing installation: beautifulsoup4 4.6.3
    Uninstalling beautifulsoup4-4.6.3:
      Successfully uninstalled beautifulsoup4-4.6.3
Successfully installed beautifulsoup4-4.11.2 cryptography-39.0.1 frozendict-2.3.5 html5lib-1.1 requests-2.28.2 soupsieve-2.4 yfinance-0.2.12
```

In [ ]:
```
import pandas as pd
import yfinance as yf
```

In [ ]:
```
Ge = yf.download("GE")
```

```
[*********************100%***********************]  1 of 1 completed
```

In [ ]:
```
Ge
```

Out[ ]:

Date	Open	High	Low	Close	Adj Close	Volume
1962-01-02	4.691347	4.769536	4.644433	4.675709	0.786456	345317
1962-01-03	4.652252	4.652252	4.613158	4.628796	0.778565	236606
1962-01-04	4.628796	4.667890	4.534969	4.574063	0.769359	294159
1962-01-05	4.574063	4.581882	4.378590	4.456780	0.749632	436442
1962-01-08	4.456780	4.456780	4.316039	4.448961	0.748316	495593
...	...	...	...	...	...	...
2023-02-17	83.589996	83.820000	82.230003	83.040001	83.040001	4424200
2023-02-21	82.139999	83.379997	81.809998	82.260002	82.260002	4071800
2023-02-22	82.160004	84.330002	82.099998	83.190002	83.190002	8642500
2023-02-23	83.489998	84.410004	81.900002	82.940002	82.940002	8077200
2023-02-24	81.760002	83.779999	81.650002	83.550003	83.550003	6946800

15393 rows × 6 columns

In [ ]:

In [ ]:

In [ ]:
```
ticker = "GE"
```

In [ ]:
```
GE=yf.download(ticker,start= "2023-01-01",end = "2023-02-03")
```

```
[*********************100%***********************]  1 of 1 completed
```

In [ ]:
```
GE
```

Out[ ]:

Date	Open	High	Low	Close	Adj Close	Volume
2023-01-03	65.612801	66.424667	65.214676	66.338799	66.338799	8204933
2023-01-04	68.410004	70.199997	66.750000	70.199997	70.199997	16784600
2023-01-05	69.900002	71.550003	69.000000	71.290001	71.290001	12770200
2023-01-06	72.010002	72.330002	70.750000	71.940002	71.940002	10389000
2023-01-09	72.000000	73.910004	71.959999	72.669998	72.669998	7572900
2023-01-10	72.199997	75.370003	72.199997	75.269997	75.269997	9255500
2023-01-11	75.820000	77.699997	75.510002	77.690002	77.690002	10159300
2023-01-12	77.879997	79.209999	77.290001	78.860001	78.860001	11498700
2023-01-13	78.779999	80.599998	78.500000	80.199997	80.199997	11044800
2023-01-17	79.919998	80.660004	79.760002	80.489998	80.489998	8492200
2023-01-18	80.940002	81.180000	79.139999	79.269997	79.269997	7834300
2023-01-19	77.940002	78.750000	76.669998	76.860001	76.860001	10411700
2023-01-20	77.589996	77.750000	75.669998	77.680000	77.680000	8257900
2023-01-23	77.610001	80.019997	77.540001	79.769997	79.769997	10957500
2023-01-24	78.000000	80.849998	77.470001	80.699997	80.699997	13381800
2023-01-25	79.779999	81.349998	79.430000	80.790001	80.790001	8501500
2023-01-26	81.480003	81.580002	80.120003	81.139999	81.139999	6511200
2023-01-27	81.000000	83.989998	80.900002	83.230003	83.230003	7789100
2023-01-30	82.410004	82.750000	80.669998	80.830002	80.830002	5726900
2023-01-31	80.449997	80.910004	79.470001	80.480003	80.480003	6883500
2023-02-01	80.269997	82.470001	80.010002	82.320000	82.320000	7272300
2023-02-02	82.190002	84.029999	81.900002	83.940002	83.940002	8293000

In [ ]:
```
GE.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 22 entries, 2023-01-03 to 2023-02-02
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Open       22 non-null     float64
 1   High       22 non-null     float64
 2   Low        22 non-null     float64
 3   Close      22 non-null     float64
 4   Adj Close  22 non-null     float64
 5   Volume     22 non-null     int64
dtypes: float64(5), int64(1)
memory usage: 1.2 KB
```

In [ ]:
```
yf.download(ticker , period = "ytd")
```

```
[*********************100%***********************]  1 of 1 completed
```

Out[ ]:

Date	Open	High	Low	Close	Adj Close	Volume
2023-01-03	65.612801	66.424667	65.214676	66.338799	66.338799	8204933
2023-01-04	68.410004	70.199997	66.750000	70.199997	70.199997	16784600
2023-01-05	69.900002	71.550003	69.000000	71.290001	71.290001	12770200
2023-01-06	72.010002	72.330002	70.750000	71.940002	71.940002	10389000
2023-01-09	72.000000	73.910004	71.959999	72.669998	72.669998	7572900
2023-01-10	72.199997	75.370003	72.199997	75.269997	75.269997	9255500
2023-01-11	75.820000	77.699997	75.510002	77.690002	77.690002	10159300
2023-01-12	77.879997	79.209999	77.290001	78.860001	78.860001	11498700
2023-01-13	78.779999	80.599998	78.500000	80.199997	80.199997	11044800
2023-01-17	79.919998	80.660004	79.760002	80.489998	80.489998	8492200
2023-01-18	80.940002	81.180000	79.139999	79.269997	79.269997	7834300
2023-01-19	77.940002	78.750000	76.669998	76.860001	76.860001	10411700
2023-01-20	77.589996	77.750000	75.669998	77.680000	77.680000	8257900
2023-01-23	77.610001	80.019997	77.540001	79.769997	79.769997	10957500
2023-01-24	78.000000	80.849998	77.470001	80.699997	80.699997	13381800
2023-01-25	79.779999	81.349998	79.430000	80.790001	80.790001	8501500
2023-01-26	81.480003	81.580002	80.120003	81.139999	81.139999	6511200
2023-01-27	81.000000	83.989998	80.900002	83.230003	83.230003	7789100
2023-01-30	82.410004	82.750000	80.669998	80.830002	80.830002	5726900
2023-01-31	80.449997	80.910004	79.470001	80.480003	80.480003	6883500
2023-02-01	80.269997	82.470001	80.010002	82.320000	82.320000	7272300
2023-02-02	82.190002	84.029999	81.900002	83.940002	83.940002	8293000
2023-02-03	83.510002	83.690002	81.820000	81.959999	81.959999	5777400
2023-02-06	81.470001	82.440002	81.239998	82.349998	82.349998	4329700
2023-02-07	81.980003	82.370003	80.540001	82.110001	82.110001	4702300
2023-02-08	81.690002	82.599998	81.300003	81.959999	81.959999	3407400
2023-02-09	82.339996	82.629997	80.440002	80.790001	80.790001	3861400
2023-02-10	80.809998	81.349998	80.400002	81.290001	81.290001	3490100
2023-02-13	81.199997	82.919998	81.080002	82.680000	82.680000	4081600
2023-02-14	82.129997	83.809998	81.959999	83.540001	83.540001	5368500
2023-02-15	82.610001	84.790001	82.440002	84.769997	84.769997	5831400
2023-02-16	83.610001	84.540001	82.980003	84.050003	84.050003	4505900
2023-02-17	83.589996	83.820000	82.230003	83.040001	83.040001	4424200
2023-02-21	82.139999	83.379997	81.809998	82.260002	82.260002	4071800
2023-02-22	82.160004	84.330002	82.099998	83.190002	83.190002	8642500
2023-02-23	83.489998	84.410004	81.900002	82.940002	82.940002	8077200
2023-02-24	81.760002	83.779999	81.650002	83.550003	83.550003	6946800

In [ ]:
```
yf.download(ticker , period = "5d")
```

```
[*********************100%***********************]  1 of 1 completed
```

Out[ ]:

Date	Open	High	Low	Close	Adj Close	Volume
2023-02-17	83.589996	83.820000	82.230003	83.040001	83.040001	4424200
2023-02-21	82.139999	83.379997	81.809998	82.260002	82.260002	4071800
2023-02-22	82.160004	84.330002	82.099998	83.190002	83.190002	8642500
2023-02-23	83.489998	84.410004	81.900002	82.940002	82.940002	8077200
2023-02-24	81.760002	83.779999	81.650002	83.550003	83.550003	6946800

In [ ]:
```
yf.download(ticker , period = "1mo")
```

```
[*********************100%***********************]  1 of 1 completed
```

Out[ ]:

Date	Open	High	Low	Close	Adj Close	Volume
2023-02-24	81.760002	83.779999	81.650002	83.550003	83.550003	6946800

In [ ]:
```
yf.download(ticker , period = "10d")
```

```
[*********************100%***********************]  1 of 1 completed
```

Out[ ]:

Date	Open	High	Low	Close	Adj Close	Volume
2023-02-10	80.809998	81.349998	80.400002	81.290001	81.290001	3490100
2023-02-13	81.199997	82.919998	81.080002	82.680000	82.680000	4081600
2023-02-14	82.129997	83.809998	81.959999	83.540001	83.540001	5368500
2023-02-15	82.610001	84.790001	82.440002	84.769997	84.769997	5831400
2023-02-16	83.610001	84.540001	82.980003	84.050003	84.050003	4505900
2023-02-17	83.589996	83.820000	82.230003	83.040001	83.040001	4424200
2023-02-21	82.139999	83.379997	81.809998	82.260002	82.260002	4071800
2023-02-22	82.160004	84.330002	82.099998	83.190002	83.190002	8642500
2023-02-23	83.489998	84.410004	81.900002	82.940002	82.940002	8077200
2023-02-24	81.760002	83.779999	81.650002	83.550003	83.550003	6946800

In [ ]:
```
yf.download(ticker , period = "10y")
```

```
[*********************100%***********************]  1 of 1 completed
```

Out[ ]:

Date	Open	High	Low	Close	Adj Close	Volume
2013-02-25	141.115707	141.295868	136.972321	136.972321	112.056107	8596245
2013-02-26	137.933105	139.013992	137.572815	138.413498	113.235146	6880403
2013-02-27	138.233353	140.515228	138.113251	140.335068	114.807190	4942011
2013-02-28	140.335068	140.875519	139.314240	139.434341	114.070305	6613672
2013-03-01	138.233353	140.154922	138.113251	139.254181	113.922905	6865516
...	...	...	...	...	...	...
2023-02-17	83.589996	83.820000	82.230003	83.040001	83.040001	4424200
2023-02-21	82.139999	83.379997	81.809998	82.260002	82.260002	4071800
2023-02-22	82.160004	84.330002	82.099998	83.190002	83.190002	8642500
2023-02-23	83.489998	84.410004	81.900002	82.940002	82.940002	8077200
2023-02-24	81.760002	83.779999	81.650002	83.550003	83.550003	6946800

2519 rows × 6 columns

In [ ]:
```
GE=yf.download(ticker, period = "5d",interval ="1m")
```

```
[*********************100%***********************]  1 of 1 completed
```

In [ ]:
```
GE.head(10)
```

Out[ ]:

Datetime	Open	High	Low	Close	Adj Close	Volume
2023-02-17 09:30:00-05:00	83.589996	83.699997	83.500000	83.620003	83.620003	158135
2023-02-17 09:31:00-05:00	83.580002	83.739998	83.500000	83.620003	83.620003	14655
2023-02-17 09:32:00-05:00	83.699997	83.815002	83.699997	83.769997	83.769997	3282
2023-02-17 09:33:00-05:00	83.750000	83.820000	83.723000	83.724998	83.724998	9949
2023-02-17 09:34:00-05:00	83.713402	83.769997	83.644997	83.660004	83.660004	8209
2023-02-17 09:35:00-05:00	83.629997	83.709999	83.580002	83.660004	83.660004	7608
2023-02-17 09:36:00-05:00	83.620003	83.620003	83.495003	83.495003	83.495003	6705
2023-02-17 09:37:00-05:00	83.470001	83.499100	83.400002	83.474998	83.474998	9783
2023-02-17 09:38:00-05:00	83.466599	83.510002	83.440002	83.459999	83.459999	13546
2023-02-17 09:39:00-05:00	83.480003	83.519997	83.279999	83.290001	83.290001	14038

In [ ]:
```
GE.describe()
```

Out[ ]:

	Open	High	Low	Close	Adj Close	Volume
count	1950.000000	1950.000000	1950.000000	1950.000000	1950.000000	1950.000000
mean	82.855813	82.894762	82.817301	82.856026	82.856026	14627.345128
std	0.601664	0.604757	0.597718	0.599837	0.599837	20726.058945
min	81.760002	81.860001	81.650002	81.800003	81.800003	931.000000
25%	82.339996	82.385401	82.309998	82.342375	82.342375	5497.250000
50%	82.779999	82.819851	82.739998	82.773399	82.773399	8695.500000
75%	83.290001	83.338747	83.239998	83.290001	83.290001	15028.750000
max	84.320000	84.410004	84.269997	84.360001	84.360001	350015.000000

In [ ]:
```
yf.download(ticker, period = "5d",prepost=True,interval ="1m")
```

```
[*********************100%***********************]  1 of 1 completed
```

Out[ ]:

Datetime	Open	High	Low	Close	Adj Close	Volume
2023-02-17 04:00:00-05:00	83.91	83.91	83.91	83.91	83.91	0
2023-02-17 05:43:00-05:00	83.29	83.86	83.29	83.76	83.76	0
2023-02-17 06:13:00-05:00	83.76	83.76	83.76	83.76	83.76	0
2023-02-17 06:53:00-05:00	83.76	83.76	83.76	83.76	83.76	0
2023-02-17 07:00:00-05:00	83.75	83.78	83.71	83.78	83.78	0
...	...	...	...	...	...	...
2023-02-24 19:32:00-05:00	83.41	83.41	83.41	83.41	83.41	0
2023-02-24 19:35:00-05:00	83.41	83.41	83.41	83.41	83.41	0
2023-02-24 19:52:00-05:00	83.52	83.52	83.41	83.41	83.41	0
2023-02-24 19:54:00-05:00	83.41	83.41	83.41	83.41	83.41	0
2023-02-24 19:58:00-05:00	83.41	83.41	83.41	83.41	83.41	0

2490 rows × 6 columns

In [ ]:
```
yf.download("GE", period = "5d",prepost=True,interval ="1m")
```

```
[*********************100%***********************]  1 of 1 completed
```

Out[ ]:

Datetime	Open	High	Low	Close	Adj Close	Volume
2023-02-17 04:00:00-05:00	83.91	83.91	83.91	83.91	83.91	0
2023-02-17 05:43:00-05:00	83.29	83.86	83.29	83.76	83.76	0
2023-02-17 06:13:00-05:00	83.76	83.76	83.76	83.76	83.76	0
2023-02-17 06:53:00-05:00	83.76	83.76	83.76	83.76	83.76	0
2023-02-17 07:00:00-05:00	83.75	83.78	83.71	83.78	83.78	0
...	...	...	...	...	...	...
2023-02-24 19:32:00-05:00	83.41	83.41	83.41	83.41	83.41	0
2023-02-24 19:35:00-05:00	83.41	83.41	83.41	83.41	83.41	0
2023-02-24 19:52:00-05:00	83.52	83.52	83.41	83.41	83.41	0
2023-02-24 19:54:00-05:00	83.41	83.41	83.41	83.41	83.41	0
2023-02-24 19:58:00-05:00	83.41	83.41	83.41	83.41	83.41	0

2490 rows × 6 columns

In [ ]:
```
GE=yf.download(ticker, period = "5y", actions = True)
```

```
[*********************100%***********************]  1 of 1 completed
```

In [ ]:
```
GE
```

Out[ ]:

Date	Open	High	Low	Close	Adj Close	Volume	Dividends	Stock Splits
2018-02-26	86.410858	88.152283	83.768692	87.972137	84.843300	24183587	0.0	0.0
2018-02-27	88.092239	91.154747	87.071396	87.071396	83.974602	15491287	0.0	0.0
2018-02-28	87.671890	87.791992	84.609383	84.729477	81.715973	14756906	0.0	0.0
2018-03-01	85.029724	85.630219	83.768692	84.189034	81.194756	15300060	0.0	0.0
2018-03-02	84.008888	85.510117	83.888786	84.789528	81.773872	12428067	0.0	0.0
...	...	...	...	...	...	...	...	...
2023-02-17	83.589996	83.820000	82.230003	83.040001	83.040001	4424200	0.0	0.0
2023-02-21	82.139999	83.379997	81.809998	82.260002	82.260002	4071800	0.0	0.0
2023-02-22	82.160004	84.330002	82.099998	83.190002	83.190002	8642500	0.0	0.0
2023-02-23	83.489998	84.410004	81.900002	82.940002	82.940002	8077200	0.0	0.0
2023-02-24	81.760002	83.779999	81.650002	83.550003	83.550003	6946800	0.0	0.0

1259 rows × 8 columns

In [ ]:
```
GE[GE["Dividends"]>0]
```

Out[ ]:

Date	Open	High	Low	Close	Adj Close	Volume	Dividends	Stock Splits
2018-06-15	81.126526	81.186569	79.865494	79.865494	77.708633	16016872	0.720591	0.0
2018-09-14	75.241699	76.562782	74.340958	76.142441	74.792244	11652071	0.720591	0.0
2018-12-19	45.877621	47.739147	45.337177	45.997719	45.244228	36549039	0.060049	0.0
2019-03-08	58.266979	60.452770	57.954723	59.828259	58.910545	14013932	0.062451	0.0
2019-06-28	64.886810	65.573769	64.761902	65.573769	64.630081	7048767	0.062451	0.0
2019-09-13	58.142078	59.016392	57.080406	58.329430	57.552143	7300003	0.062451	0.0
2019-12-20	69.195938	69.570648	68.071815	68.883682	68.027367	11791781	0.062451	0.0
2020-03-06	60.827480	60.952381	57.330212	58.704136	58.031937	23772718	0.062451	0.0
2020-06-26	41.342701	41.530056	40.281029	40.468384	40.064976	14316712	0.062451	0.0
2020-09-25	37.845432	38.532398	37.595627	38.157688	37.839752	12774564	0.062451	0.0
2020-12-18	67.322403	69.445747	67.072601	67.509758	67.008850	18374920	0.062451	0.0
2021-03-05	86.182671	86.245117	82.060890	84.933647	84.365631	16809202	0.062451	0.0
2021-06-25	82.185791	82.685402	81.811089	82.185791	81.698280	5462760	0.062451	0.0
2021-09-24	80.140518	81.342705	80.093674	81.030441	80.612419	8402720	0.062451	0.0
2021-12-20	70.257614	70.335678	68.829041	70.241997	69.940811	7702909	0.062451	0.0
2022-03-07	69.219360	69.640907	66.580795	66.651054	66.424881	9120848	0.062451	0.0
2022-06-27	52.669788	53.005463	51.631538	52.076504	51.961758	8708622	0.062451	0.0
2022-09-26	50.374706	51.647152	50.124901	50.234192	50.185703	13392343	0.062451	0.0
2022-12-14	64.309135	65.058548	63.497269	63.598751	63.598751	9794014	0.062451	0.0

In [ ]:
```
GE.loc['2019-08-01':"2019-08-15"]
```

Out[ ]:

Date	Open	High	Low	Close	Adj Close	Volume	Dividends	Stock Splits
2019-08-01	64.761902	65.511322	62.326309	62.950821	62.044868	14078462	0.0	0.0
2019-08-02	62.513660	62.950821	61.327087	62.451210	61.552448	10295461	0.0	0.0
2019-08-05	61.139736	61.264637	60.015614	60.327869	59.459675	11247692	0.0	0.0
2019-08-06	60.640125	60.889931	59.203747	59.765808	58.905693	10631948	0.0	0.0
2019-08-07	58.953941	59.141296	57.642467	59.078846	58.228622	12296063	0.0	0.0
2019-08-08	59.391102	59.516003	58.204529	59.266197	58.413280	7911232	0.0	0.0
2019-08-09	59.141296	59.203747	56.955505	57.142857	56.320492	12284918	0.0	0.0
2019-08-12	56.455894	56.768150	55.893833	56.518345	55.704967	10500181	0.0	0.0
2019-08-13	57.642467	59.890709	57.517563	58.391880	57.551537	11853477	0.0	0.0
2019-08-14	57.704918	58.079624	55.956284	56.393444	55.581860	14282285	0.0	0.0
2019-08-15	53.333332	53.895393	47.775177	50.023418	49.303516	64217923	0.0	0.0

In [ ]:
```
GE[GE["Stock Splits"]>0]
```

Out[ ]:

Date	Open	High	Low	Close	Adj Close	Volume	Dividends	Stock Splits
2019-02-26	67.384857	68.071815	65.761124	66.572990	65.482460	16651495	0.0	1.040
2021-08-02	81.561279	83.692429	78.399689	78.532394	78.066551	31878326	0.0	0.125
2023-01-04	68.410004	70.199997	66.750000	70.199997	70.199997	16784600	0.0	1.281

In [ ]:
```
GE.to_csv("GE.csv")
```

In [ ]:
```
pd.read_csv("GE.csv",parse_dates = ["Date"],index_col="Date")
```

Out[ ]:

Date	Open	High	Low	Close	Adj Close	Volume	Dividends	Stock Splits
2018-02-26	86.410858	88.152283	83.768692	87.972137	84.843300	24183587	0.0	0.0
2018-02-27	88.092239	91.154747	87.071396	87.071396	83.974602	15491287	0.0	0.0
2018-02-28	87.671890	87.791992	84.609383	84.729477	81.715973	14756906	0.0	0.0
2018-03-01	85.029724	85.630219	83.768692	84.189034	81.194756	15300060	0.0	0.0
2018-03-02	84.008888	85.510117	83.888786	84.789528	81.773872	12428067	0.0	0.0
...	...	...	...	...	...	...	...	...
2023-02-17	83.589996	83.820000	82.230003	83.040001	83.040001	4424200	0.0	0.0
2023-02-21	82.139999	83.379997	81.809998	82.260002	82.260002	4071800	0.0	0.0
2023-02-22	82.160004	84.330002	82.099998	83.190002	83.190002	8642500	0.0	0.0
2023-02-23	83.489998	84.410004	81.900002	82.940002	82.940002	8077200	0.0	0.0
2023-02-24	81.760002	83.779999	81.650002	83.550003	83.550003	6946800	0.0	0.0

1259 rows × 8 columns

In [ ]:
```
pd.read_csv("GE.csv",index_col="Date")
```

Out[ ]:

Date	Open	High	Low	Close	Adj Close	Volume	Dividends	Stock Splits
2018-02-26	86.410858	88.152283	83.768692	87.972137	84.843300	24183587	0.0	0.0
2018-02-27	88.092239	91.154747	87.071396	87.071396	83.974602	15491287	0.0	0.0
2018-02-28	87.671890	87.791992	84.609383	84.729477	81.715973	14756906	0.0	0.0
2018-03-01	85.029724	85.630219	83.768692	84.189034	81.194756	15300060	0.0	0.0
2018-03-02	84.008888	85.510117	83.888786	84.789528	81.773872	12428067	0.0	0.0
...	...	...	...	...	...	...	...	...
2023-02-17	83.589996	83.820000	82.230003	83.040001	83.040001	4424200	0.0	0.0
2023-02-21	82.139999	83.379997	81.809998	82.260002	82.260002	4071800	0.0	0.0
2023-02-22	82.160004	84.330002	82.099998	83.190002	83.190002	8642500	0.0	0.0
2023-02-23	83.489998	84.410004	81.900002	82.940002	82.940002	8077200	0.0	0.0
2023-02-24	81.760002	83.779999	81.650002	83.550003	83.550003	6946800	0.0	0.0

1259 rows × 8 columns

In [ ]:
```
tickers = ["GE","AAPL","META"]
```

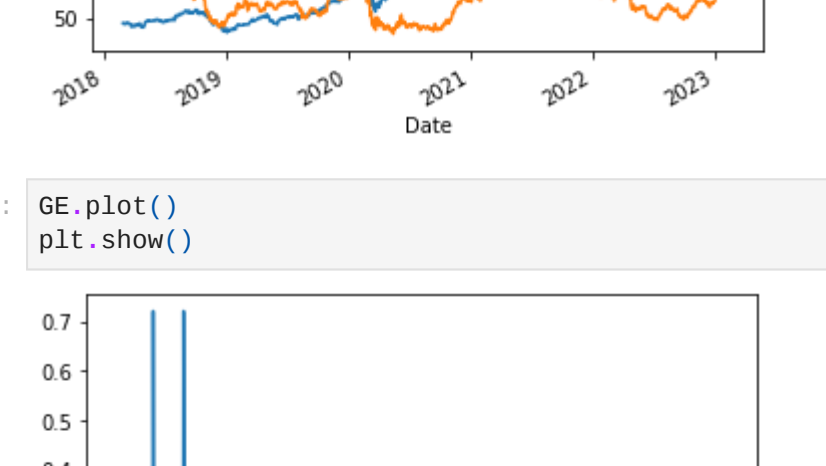
In [ ]:
```
stocks=yf.download(tickers,period="5y")
```

```
[*********************100%***********************]  3 of 3 completed
```

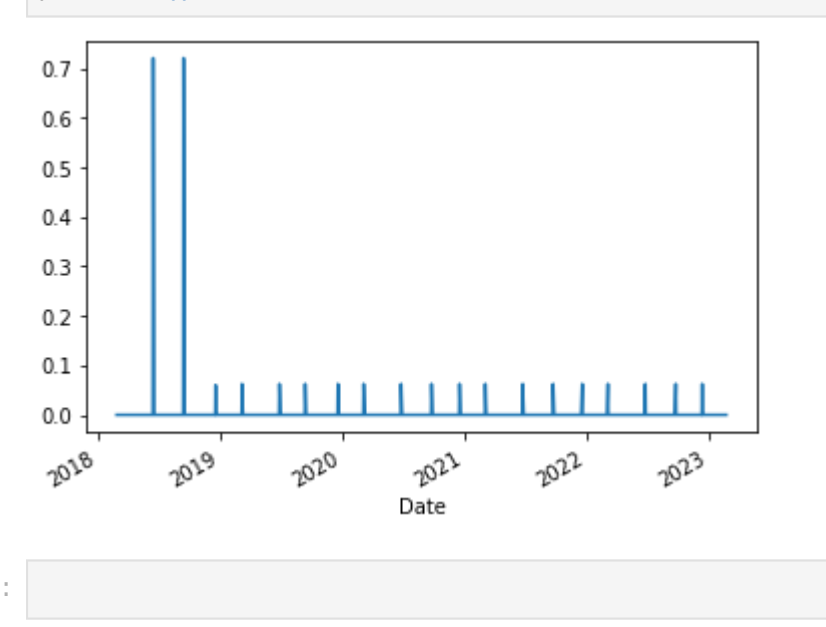
In [ ]:
```
s=stocks.Close
```

In [ ]:
```
import matplotlib.pyplot as plt
```

In [ ]:
```
s.plot()
plt.show()
```



In [ ]:
```
GE.plot()
plt.show()
```



In [ ]: