A.Y.: 2022-23 **Class/Sem: T.Y.B.Tech/ Sem-VI** **Sub:** Computational Finance

Experiment 3

Aim: Visualize and prepare data analysis on stock data prices, volume, and moving averages using a web API-Alpha Vantage

Theory: Alpha Vantage offers free stock APIs in JSON and CSV formats for realtime and historical stock market data, forex, commodity, cryptocurrency feeds and over. One of the main investing principles is that past performance even if it is not an indicator of future performance. However, it is good to look at the historical price and volume charts to get a sense of the range a stock has been trading in, notice trends and patterns, and locate the price levels at which investors are particularly active.
Alpha Vantage provides enterprise-grade financial market data through a set of powerful and developer-friendly data APIs and spreadsheets. From traditional asset classes (e.g., stocks, ETFs, mutual funds) to economic indicators, from foreign exchange rates to commodities, from fundamental data to technical indicators, Alpha Vantage is your one-stop-shop for real-time and historical global market data delivered through cloud-based APIs, Excel, and Google Sheets.
the Alpha Academy is an open knowledge base for essential content in quantitative investing, machine learning, software development, block chain technologies and more, all delivered to you by industry experts.

TIME_SERIES_INTRADAY Trending

This API returns intraday time series of the equity specified, covering extended trading hours where applicable (e.g., 4:00am to 8:00pm Eastern Time for the US market). The intraday data is derived from the Securities Information Processor (SIP) market-aggregated data. You can query both raw (as-traded) and split/dividend-adjusted intraday data from this endpoint.

This API returns the most recent 1-2 months of intraday data and is best suited for short-term/medium-term charting and trading strategy development. If you are targeting a deeper intraday history, please use the Extended Intraday API.

Lab Experiment to be done by students:

1. Get free API key by filling out a couple of fields on this link: https://www.alphavantage.co/support/#api-key
2. Get Historical Stock Price and Volume Data from Web API alpha Vantage
3. Set specific Time Periods for given stock data.
4. Analyse Stock Splits and Dividends
5. Create a DatetimeIndex and year wise analysis
6. Perform Frequency Settings (Intraday) at various time intervals
7. Plot bands using Technical Indicators
8. Implement foreign exchange Currencies / FX
9. Display daily digital Cryptocurrencies
10. Plot any technical indicator you are interested in or zoom into the specific period. Moving averages smooth out the sharp increases or decreases, with 50, 100, and 200-day averages.

Expected output:

FMC - LAB-3

Name: Sarvagya Singh

SAP: 60009200030

Div/Batch: K/K1

Lab 3: Visualize and prepare data analysis on stock data prices, volume, and moving averages usinga web API-Alpha Vantage

Getting Data from Alpha Vantage

In [ ]:

```
mykey = "DP3VNT98IOLXIA8D"#"INSERT YOUR PESONAL KEY HERE!"
```

Getting Historical Stock Price and Volume Data

In [ ]:

```
pip install alpha_vantage
```

In [ ]:

```
pip install yfinance
```

In [ ]:

```
import pandas as pd
from alpha_vantage.timeseries import TimeSeries
```

In [ ]:

```
ts = TimeSeries(key=mykey, output_format='pandas')
```

In [ ]:

```
GE = ts.get_daily_adjusted("GE")
```

In [ ]:

```
GE
```

Out[ ]:

```
(            1. open  2. high  3. low  4. close  5. adjusted close  6. volume  \
 date
 2023-04-14    93.86   95.640   93.81     95.44          95.440000  5755636.0
 2023-04-13    94.38   94.690   93.47     94.30          94.300000  8814584.0
 2023-04-12    94.26   94.670   93.81     94.10          94.100000  6997746.0
 2023-04-11    94.51   94.960   93.88     93.91          93.910000  5317712.0
 2023-04-10    93.83   94.990   93.61     94.36          94.360000  5985450.0
 ...             ...      ...     ...       ...                ...        ...
 2022-11-25    87.70   88.380   87.35     88.14          68.675017  1894075.0
 2022-11-23    87.03   87.960   86.94     87.87          68.464644  3655685.0
 2022-11-22    86.30   87.585   86.00     87.30          68.020524  4234987.0
 2022-11-21    85.11   85.995   84.73     85.89          66.921911  3269761.0
 2022-11-18    86.49   87.150   84.85     85.48          66.602456  4084167.0

             7. dividend amount  8. split coefficient
 date
```

```
2023-04-14                0.0                   1.0
2023-04-13                0.0                   1.0
2023-04-12                0.0                   1.0
2023-04-11                0.0                   1.0
2023-04-10                0.0                   1.0
...                       ...                   ...
2022-11-25                0.0                   1.0
2022-11-23                0.0                   1.0
2022-11-22                0.0                   1.0
2022-11-21                0.0                   1.0
2022-11-18                0.0                   1.0

[100 rows x 8 columns],
{'1. Information': 'Daily Time Series with Splits and Dividend Events',
 '2. Symbol': 'GE',
 '3. Last Refreshed': '2023-04-14',
 '4. Output Size': 'Compact',
 '5. Time Zone': 'US/Eastern'})
```

In [ ]:

```
len(GE)
```

Out[ ]:

```
2
```

In [ ]:

```
GE[1]
```

Out[ ]:

```
{'1. Information': 'Daily Time Series with Splits and Dividend Events',
 '2. Symbol': 'GE',
 '3. Last Refreshed': '2023-04-14',
 '4. Output Size': 'Compact',
 '5. Time Zone': 'US/Eastern'}
```

In [ ]:

```
GE[0]
```

Out[ ]:

date	1. open	2. high	3. low	4. close	5. adjusted close	6. volume	7. dividend amount	8. split coefficient
2023-04-14	93.86	95.640	93.81	95.44	95.440000	5755636.0	0.0	1.0
2023-04-13	94.38	94.690	93.47	94.30	94.300000	8814584.0	0.0	1.0
2023-04-12	94.26	94.670	93.81	94.10	94.100000	6997746.0	0.0	1.0
2023-04-11	94.51	94.960	93.88	93.91	93.910000	5317712.0	0.0	1.0
2023-04-10	93.83	94.990	93.61	94.36	94.360000	5985450.0	0.0	1.0
...	...	...	...	...	...	...	...	...
2022-11-25	87.70	88.380	87.35	88.14	68.675017	1894075.0	0.0	1.0
2022-11-23	87.03	87.960	86.94	87.87	68.464644	3655685.0	0.0	1.0
2022-11-22	86.30	87.585	86.00	87.30	68.020524	4234987.0	0.0	1.0
2022-11-21	85.11	85.995	84.73	85.89	66.921911	3269761.0	0.0	1.0
2022-11-18	86.49	87.150	84.85	85.48	66.602456	4084167.0	0.0	1.0

100 rows × 8 columns

Setting specific Time Periods

In [ ]:

In [ ]:

```
import pandas as pd
from alpha_vantage.timeseries import TimeSeries
```

In [ ]:

```
ts = TimeSeries(key=mykey, output_format='pandas')
```

In [ ]:

```
GE = ts.get_daily_adjusted("GE", outputsize= "compact")[0]
```

In [ ]:

```
GE
```

Out[ ]:

date	1. open	2. high	3. low	4. close	5. adjusted close	6. volume	7. dividend amount	8. split coefficient
2023-04-14	93.86	95.640	93.81	95.44	95.440000	5755636.0	0.0	1.0
2023-04-13	94.38	94.690	93.47	94.30	94.300000	8814584.0	0.0	1.0
2023-04-12	94.26	94.670	93.81	94.10	94.100000	6997746.0	0.0	1.0
2023-04-11	94.51	94.960	93.88	93.91	93.910000	5317712.0	0.0	1.0
2023-04-10	93.83	94.990	93.61	94.36	94.360000	5985450.0	0.0	1.0
...	...	...	...	...	...	...	...	...
2022-11-25	87.70	88.380	87.35	88.14	68.675017	1894075.0	0.0	1.0
2022-11-23	87.03	87.960	86.94	87.87	68.464644	3655685.0	0.0	1.0
2022-11-22	86.30	87.585	86.00	87.30	68.020524	4234987.0	0.0	1.0
2022-11-21	85.11	85.995	84.73	85.89	66.921911	3269761.0	0.0	1.0
2022-11-18	86.49	87.150	84.85	85.48	66.602456	4084167.0	0.0	1.0

100 rows × 8 columns

In [ ]:

```
GE = ts.get_daily_adjusted("GE", outputsize= "full")[0]
```

In [ ]:

```
GE.head()
```

Out[ ]:

date	1. open	2. high	3. low	4. close	5. adjusted close	6. volume	7. dividend amount	8. split coefficient
2023-04-14	93.86	95.64	93.81	95.44	95.44	5755636.0	0.0	1.0
2023-04-13	94.38	94.69	93.47	94.30	94.30	8814584.0	0.0	1.0
2023-04-12	94.26	94.67	93.81	94.10	94.10	6997746.0	0.0	1.0
2023-04-11	94.51	94.96	93.88	93.91	93.91	5317712.0	0.0	1.0
2023-04-10	93.83	94.99	93.61	94.36	94.36	5985450.0	0.0	1.0

In [ ]:

```
GE.tail()
```

Out[ ]:

date	1. open	2. high	3. low	4. close	5. adjusted close	6. volume	7. dividend amount	8. split coefficient
1999-11-05	133.19	134.81	133.19	133.75	147.080838	4688900.0	0.0	1.0
1999-11-04	132.50	133.56	130.50	131.88	145.024455	4353600.0	0.0	1.0
1999-11-03	132.88	132.94	130.00	131.38	144.474620	4589000.0	0.0	1.0
1999-11-02	129.69	133.13	128.19	129.00	141.857406	6340600.0	0.0	1.0
1999-11-01	133.63	134.38	129.25	129.38	142.275281	6795500.0	0.0	1.0

Stock Splits and Dividends

In [ ]:

```
import pandas as pd
from alpha_vantage.timeseries import TimeSeries
import yfinance as yf
```

In [ ]:

```
ts = TimeSeries(key=mykey, output_format='pandas')
```

In [ ]:

```
ticker = "GE"
```

In [ ]:

```
GE = ts.get_daily_adjusted(ticker, outputsize = "full")[0]
```

In [ ]:

```
GE.head()
```

Out[ ]:

date	1. open	2. high	3. low	4. close	5. adjusted close	6. volume	7. dividend amount	8. split coefficient
2023-04-14	93.86	95.64	93.81	95.44	95.44	5755636.0	0.0	1.0
2023-04-13	94.38	94.69	93.47	94.30	94.30	8814584.0	0.0	1.0
2023-04-12	94.26	94.67	93.81	94.10	94.10	6997746.0	0.0	1.0
2023-04-11	94.51	94.96	93.88	93.91	93.91	5317712.0	0.0	1.0
2023-04-10	93.83	94.99	93.61	94.36	94.36	5985450.0	0.0	1.0

In [ ]:

```
yf.download(ticker, start = GE.index[1]).head()
```

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

Out[ ]:

Date	Open	High	Low	Close	Adj Close	Volume
2023-04-13	94.379997	94.690002	93.470001	94.300003	94.300003	8814600
2023-04-14	93.860001	95.639999	93.809998	95.440002	95.440002	5754600

In [ ]:

```
GEa = ts.get_daily_adjusted(ticker, outputsize = "full")[0]
```

In [ ]:

```
GEa.head()
```

Out[ ]:

date	1. open	2. high	3. low	4. close	5. adjusted close	6. volume	7. dividend amount	8. split coefficient
2023-04-14	93.86	95.64	93.81	95.44	95.44	5755636.0	0.0	1.0
2023-04-13	94.38	94.69	93.47	94.30	94.30	8814584.0	0.0	1.0
2023-04-12	94.26	94.67	93.81	94.10	94.10	6997746.0	0.0	1.0
2023-04-11	94.51	94.96	93.88	93.91	93.91	5317712.0	0.0	1.0
2023-04-10	93.83	94.99	93.61	94.36	94.36	5985450.0	0.0	1.0

In [ ]:

```
GEa.iloc[:, -1].value_counts()
```

Out[ ]:

```
1.000    5898
1.281       1
0.125       1
3.000       1
Name: 8. split coefficient, dtype: int64
```

In [ ]:

```
GEa[GEa.iloc[:, -1] == 3]
```

Out[ ]:

date	1. open	2. high	3. low	4. close	5. adjusted close	6. volume	7. dividend amount	8. split coefficient
2000-05-08	52.13	52.88	51.63	52.44	173.966741	3892167.0	0.0	3.0

In [ ]:

```
GEa.loc["2000-05-03": "2000-05-10"]
```

Out[ ]:

date	1. open	2. high	3. low	4. close	5. adjusted close	6. volume	7. dividend amount	8. split coefficient
2000-05-10	51.50	52.06	50.06	50.63	167.962168	15059400.0	0.0	1.0
2000-05-09	52.38	52.69	50.88	52.13	172.938333	13439400.0	0.0	1.0
2000-05-08	52.13	52.88	51.63	52.44	173.966741	3892167.0	0.0	3.0
2000-05-05	154.00	160.00	153.50	158.00	174.718695	6895300.0	0.0	1.0
2000-05-04	157.44	157.50	152.75	154.00	170.295437	5137000.0	0.0	1.0
2000-05-03	159.50	160.00	154.56	156.06	172.573415	5531600.0	0.0	1.0

Creating a DatetimeIndex

In [ ]:

```
GE.head()
```

Out[ ]:

	1. open	2. high	3. low	4. close	5. adjusted close	6. volume	7. dividend amount	8. split coefficient

date	1. open	2. high	3. low	4. close	5. adjusted close	6. volume	7. dividend amount	8. split coefficient
2023-04-14	93.86	95.64	93.81	95.44	95.44	5755636.0	0.0	1.0
2023-04-13	94.38	94.69	93.47	94.30	94.30	8814584.0	0.0	1.0
2023-04-12	94.26	94.67	93.81	94.10	94.10	6997746.0	0.0	1.0
2023-04-11	94.51	94.96	93.88	93.91	93.91	5317712.0	0.0	1.0
2023-04-10	93.83	94.99	93.61	94.36	94.36	5985450.0	0.0	1.0

In [ ]:

```
GE.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 5901 entries, 2023-04-14 to 1999-11-01
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   1. open               5901 non-null   float64
 1   2. high               5901 non-null   float64
 2   3. low                5901 non-null   float64
 3   4. close              5901 non-null   float64
 4   5. adjusted close     5901 non-null   float64
 5   6. volume             5901 non-null   float64
 6   7. dividend amount    5901 non-null   float64
 7   8. split coefficient  5901 non-null   float64
dtypes: float64(8)
memory usage: 414.9 KB
```

In [ ]:

```
GE.index[0]
```

Out[ ]:

```
Timestamp('2023-04-14 00:00:00')
```

In [ ]:

```
GE.loc["2017-05-03":"2017-05-09"]
```

Out[ ]:

date	1. open	2. high	3. low	4. close	5. adjusted close	6. volume	7. dividend amount	8. split coefficient
2017-05-09	29.08	29.090	28.86	28.93	161.966651	23644067.0	0.0	1.0
2017-05-08	29.16	29.245	29.00	29.07	162.750451	21584760.0	0.0	1.0
2017-05-05	29.13	29.240	29.11	29.22	163.590237	16687305.0	0.0	1.0
2017-05-04	29.27	29.310	29.05	29.20	163.478265	19544097.0	0.0	1.0
2017-05-03	28.92	29.290	28.85	29.23	163.646222	26774100.0	0.0	1.0

In [ ]:

```
GE.loc["2017"]
```

Out[ ]:

date	1. open	2. high	3. low	4. close	5. adjusted close	6. volume	7. dividend amount	8. split coefficient
2017-12-29	17.27	17.530	17.27	17.45	100.178159	75906686.0	0.00	1.0
2017-12-28	17.35	17.400	17.25	17.36	99.661481	60756258.0	0.00	1.0
2017-12-27	17.46	17.630	17.31	17.38	99.776298	58655208.0	0.00	1.0

date	1. open	2. high	3. low	4. close	5. adjusted close	6. volume	7. dividend amount	8. split coefficient
2017-12-26	17.45	17.660	17.40	17.43	100.063342	55337900.0	0.12	1.0
2017-12-22	17.51	17.560	17.40	17.50	99.778261	46370400.0	0.00	1.0
...	...	...	...	...	...	...	...	...
2017-01-09	31.64	31.660	31.43	31.46	174.734094	21262120.0	0.00	1.0
2017-01-06	31.58	31.770	31.36	31.61	175.567219	22120800.0	0.00	1.0
2017-01-05	31.57	31.750	31.31	31.52	175.067344	25856523.0	0.00	1.0
2017-01-04	31.75	31.830	31.62	31.70	176.067094	21438996.0	0.00	1.0
2017-01-03	31.67	31.835	31.40	31.69	176.011552	32149537.0	0.00	1.0

251 rows × 8 columns

In [ ]:

```
GE.index = pd.to_datetime(GE.index)
```

In [ ]:

```
GE.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 5901 entries, 2023-04-14 to 1999-11-01
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   1. open               5901 non-null   float64
 1   2. high               5901 non-null   float64
 2   3. low                5901 non-null   float64
 3   4. close              5901 non-null   float64
 4   5. adjusted close     5901 non-null   float64
 5   6. volume             5901 non-null   float64
 6   7. dividend amount    5901 non-null   float64
 7   8. split coefficient  5901 non-null   float64
dtypes: float64(8)
memory usage: 414.9 KB
```

In [ ]:

```
type(GE.index[0])
```

Out[ ]:

```
pandas._libs.tslibs.timestamps.Timestamp
```

In [ ]:

```
GE.loc["2017"]
```

Out[ ]:

date	1. open	2. high	3. low	4. close	5. adjusted close	6. volume	7. dividend amount	8. split coefficient
2017-12-29	17.27	17.530	17.27	17.45	100.178159	75906686.0	0.00	1.0
2017-12-28	17.35	17.400	17.25	17.36	99.661481	60756258.0	0.00	1.0
2017-12-27	17.46	17.630	17.31	17.38	99.776298	58655208.0	0.00	1.0
2017-12-26	17.45	17.660	17.40	17.43	100.063342	55337900.0	0.12	1.0
2017-12-22	17.51	17.560	17.40	17.50	99.778261	46370400.0	0.00	1.0
...	...	...	...	...	...	...	...	...
2017-01-09	31.64	31.660	31.43	31.46	174.734094	21262120.0	0.00	1.0
2017-01-06	31.58	31.770	31.36	31.61	175.567219	22120800.0	0.00	1.0
2017-01-05	31.57	31.750	31.31	31.52	175.067344	25856523.0	0.00	1.0
2017-01-04	31.75	31.830	31.62	31.70	176.067094	21438996.0	0.00	1.0

date	1. open	2. high	3. low	4. close	5. adjusted close	6. volume	7. dividend amount	8. split coefficient
2017-01-03	31.67	31.835	31.40	31.69	176.011552	32149537.0	0.00	1.0

251 rows × 8 columns

Frequency Settings (Intraday)

In [ ]:

```
import pandas as pd
from alpha_vantage.timeseries import TimeSeries
```

In [ ]:

```
ts = TimeSeries(key=mykey, output_format='pandas')
```

In [ ]:

```
ts.get_monthly_adjusted("MSFT")[0]
```

Out[ ]:

date	1. open	2. high	3. low	4. close	5. adjusted close	6. volume	7. dividend amount
2023-04-14	286.52	292.080	281.64	286.14	286.1400	2.255364e+08	0.00
2023-03-31	250.76	289.270	245.61	288.30	288.3000	7.477520e+08	0.00
2023-02-28	248.00	276.760	245.47	249.42	249.4200	6.155408e+08	0.68
2023-01-31	243.08	249.830	219.35	247.81	247.1859	6.661681e+08	0.00
2022-12-30	253.87	263.915	233.87	239.82	239.2160	5.913665e+08	0.00
...	...	...	...	...	...	...	...
2000-04-28	94.44	96.500	65.00	69.75	21.8029	1.129073e+09	0.00
2000-03-31	89.62	115.000	88.94	106.25	33.2123	1.014094e+09	0.00
2000-02-29	98.50	110.000	88.12	89.37	27.9359	6.672438e+08	0.00
2000-01-31	117.37	118.620	94.87	97.87	30.5928	6.374376e+08	0.00
1999-12-31	91.06	119.940	90.87	116.75	36.4945	6.304889e+08	0.00

281 rows × 7 columns

In [ ]:

```
ts.get_intraday("MSFT", outputsize= "full", interval = "60min")[0]
```

Out[ ]:

date	1. open	2. high	3. low	4. close	5. volume
2023-04-13 20:00:00	289.7200	289.8000	289.66	289.7100	9580.0
2023-04-13 19:00:00	289.5700	289.6535	289.36	289.6200	8553.0
2023-04-13 18:00:00	289.3300	289.8400	289.33	289.7194	13513.0
2023-04-13 17:00:00	289.8400	290.3200	287.94	289.3500	1425148.0
2023-04-13 16:00:00	288.7701	289.9000	288.60	289.8700	4821906.0
...	...	...	...	...	...
2023-02-16 09:00:00	269.1200	270.9100	264.51	265.0310	311053.0
2023-02-16 08:00:00	268.7500	269.1900	268.10	268.5200	30240.0
2023-02-16 07:00:00	270.0000	270.0000	268.41	268.4100	20481.0

	1. open	2. high	3. low	4. close	5. volume
2023-02-16 05:00:00	270.0800	270.3900	269.95	270.0000	9713.0

date

624 rows × 5 columns

In [ ]:

```
ts.get_intraday("MSFT", outputsize= "full", interval = "1min")[0]
```

Out[ ]:

date	1. open	2. high	3. low	4. close	5. volume
2023-04-14 19:59:00	286.60	286.65	286.50	286.50	829.0
2023-04-14 19:54:00	286.50	286.50	286.50	286.50	731.0
2023-04-14 19:51:00	286.45	286.46	286.45	286.46	825.0
2023-04-14 19:49:00	286.37	286.40	286.37	286.40	701.0
2023-04-14 19:48:00	286.33	286.33	286.33	286.33	587.0
...	...	...	...	...	...
2023-03-31 04:11:00	284.10	284.10	284.10	284.10	322.0
2023-03-31 04:10:00	284.10	284.10	284.10	284.10	428.0
2023-03-31 04:06:00	284.10	284.10	284.10	284.10	595.0
2023-03-31 04:04:00	284.22	284.22	284.10	284.10	722.0
2023-03-31 04:01:00	283.95	284.30	283.95	284.30	1560.0

6317 rows × 5 columns

In [ ]:

```
import pandas as pd
from alpha_vantage.timeseries import TimeSeries
```

In [ ]:

```
ts = TimeSeries(key=mykey, output_format='pandas')
```

In [ ]:

```
msft = ts.get_daily_adjusted("MSFT", outputsize= "full")[0]
```

In [ ]:

```
close = msft.loc["1999-11-09":, "4. close"].to_frame()
```

In [ ]:

```
close
```

Out[ ]:

date	4. close
2023-04-14	286.14
2023-04-13	289.84
2023-04-12	283.49
2023-04-11	282.83
2023-04-10	289.39
...	...

	4. close
1999-11-15	87.00
date	
1999-11-12	89.19
1999-11-11	89.62
1999-11-10	87.12
1999-11-09	88.87

5895 rows × 1 columns

Technical Indicators

In [ ]:

```
import pandas as pd
from alpha_vantage.techindicators import TechIndicators
import matplotlib.pyplot as plt
```

In [ ]:

```
ti = TechIndicators(key= mykey, output_format='pandas')
```

In [ ]:

```
ti
```

Out[ ]:

```
<alpha_vantage.techindicators.TechIndicators at 0x7f397909bf70>
```

In [ ]:

```
sma = ti.get_sma("MSFT", interval = "daily", time_period = 50)[0]
```

In [ ]:

```
sma.head()
```

Out[ ]:

	SMA
date	
2000-01-11	31.2645
2000-01-12	31.3485
2000-01-13	31.4438
2000-01-14	31.5704
2000-01-18	31.7177

In [ ]:

```
close.head()
```

Out[ ]:

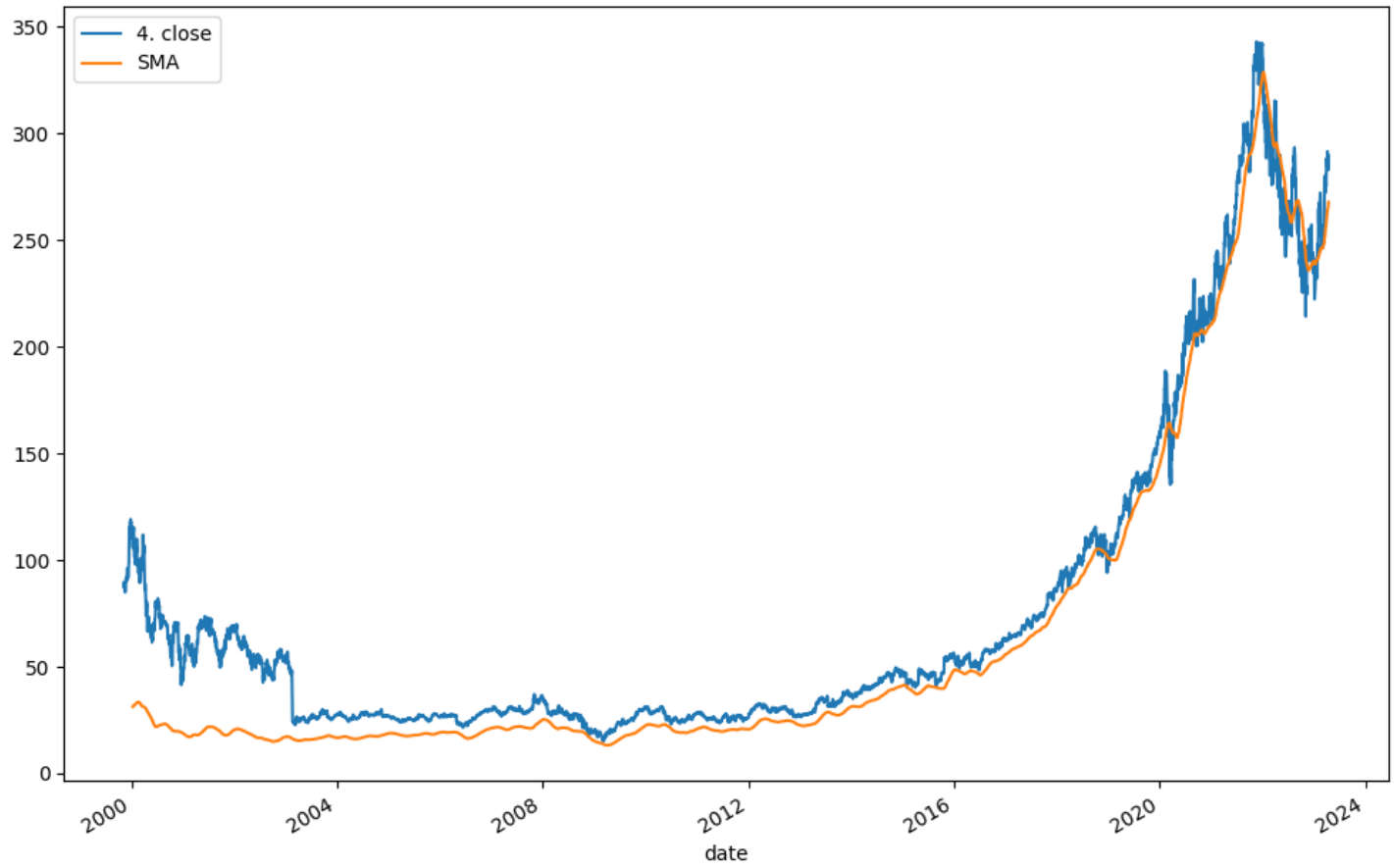
	4. close
date	
2023-04-14	286.14
2023-04-13	289.84
2023-04-12	283.49
2023-04-11	282.83
2023-04-10	289.39

In [ ]:

```
close["SMA"] = sma
```

In [ ]:

```
close.plot(figsize = (12, 8))
plt.show()
```



In [ ]:

```
bbands = ti.get_bbands("MSFT", interval = "daily", time_period = 50)[0]
```

In [ ]:

```
bbands.head()
```

Out[ ]:

date	Real Upper Band	Real Middle Band	Real Lower Band
2023-04-14	295.0512	267.8523	240.6534
2023-04-13	294.2086	267.1718	240.1350
2023-04-12	293.1315	266.3187	239.5059
2023-04-11	292.6851	265.4909	238.2967
2023-04-10	291.9745	264.7850	237.5954

In [ ]:

```
bbands["Close"] = close.iloc[:, 0]
```

In [ ]:

```
bbands
```

Out[ ]:

Out [ ]:

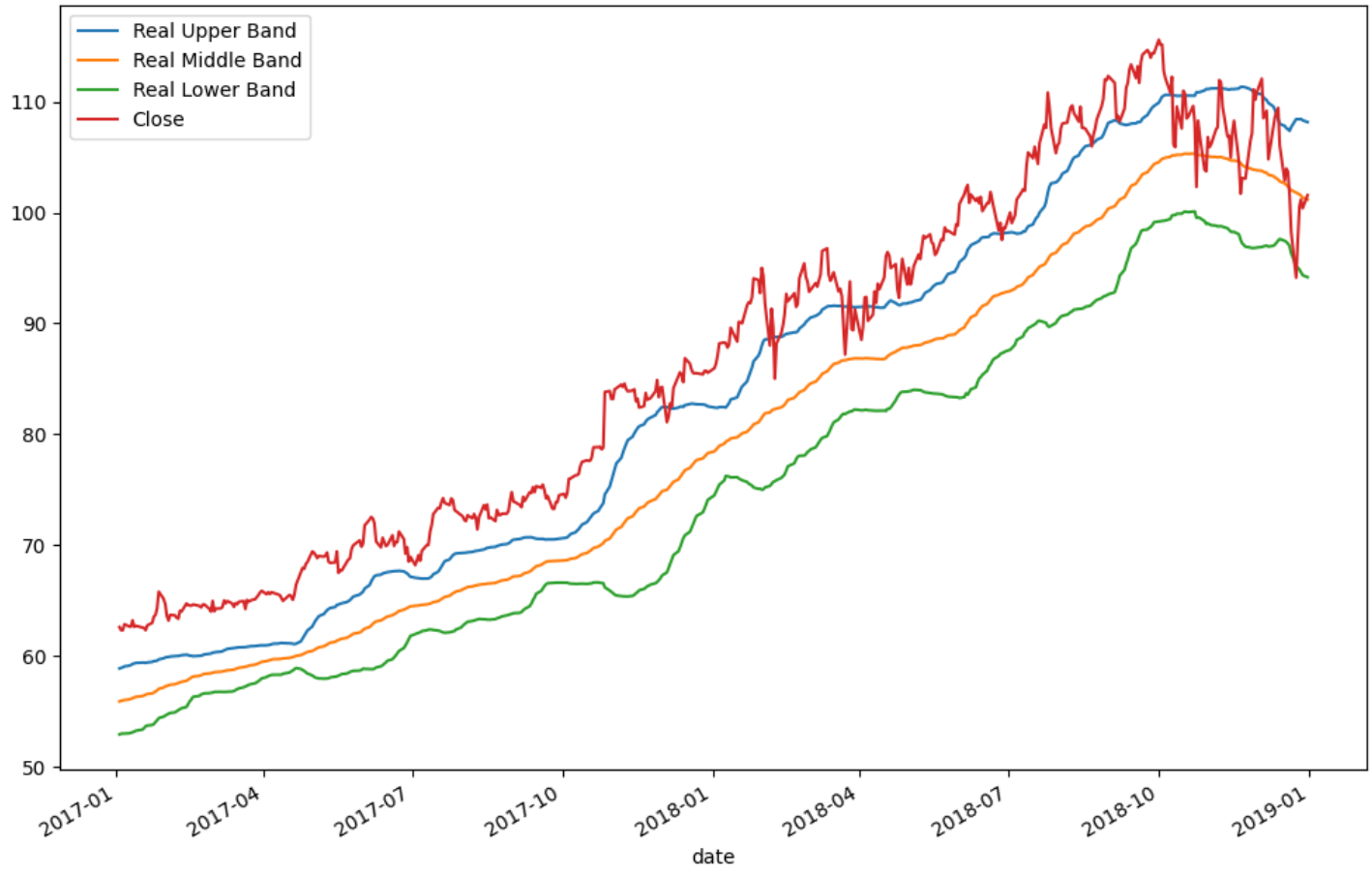
date	Real Upper Band	Real Middle Band	Real Lower Band	Close
2023-04-14	295.0512	267.8523	240.6534	286.14
2023-04-13	294.2086	267.1718	240.1350	289.84
2023-04-12	293.1315	266.3187	239.5059	283.49
2023-04-11	292.6851	265.4909	238.2967	282.83
2023-04-10	291.9745	264.7850	237.5954	289.39
...	...	...	...	...
2000-01-18	39.1660	31.7177	24.2694	115.31
2000-01-14	38.9617	31.5704	24.1791	112.25
2000-01-13	38.8066	31.4438	24.0811	107.81
2000-01-12	38.7153	31.3485	23.9816	105.81
2000-01-11	38.6465	31.2645	23.8825	109.37

5852 rows × 4 columns

In [ ]:

```
bbands.loc["2017-01-02": "2018-12-31"].plot(figsize = (12, 8))
plt.show()
```

```
<ipython-input-65-e85c8d8742a6>:1: FutureWarning: Value based partial slicing on non-mono
tonic DatetimeIndexes with non-existing keys is deprecated and will raise a KeyError in a
future Version.
  bbands.loc["2017-01-02": "2018-12-31"].plot(figsize = (12, 8))
```



In [ ]:

```
macd = ti.get_macd("MSFT", interval = "daily")[0]
```

In [ ]:

```
macd.head()
```

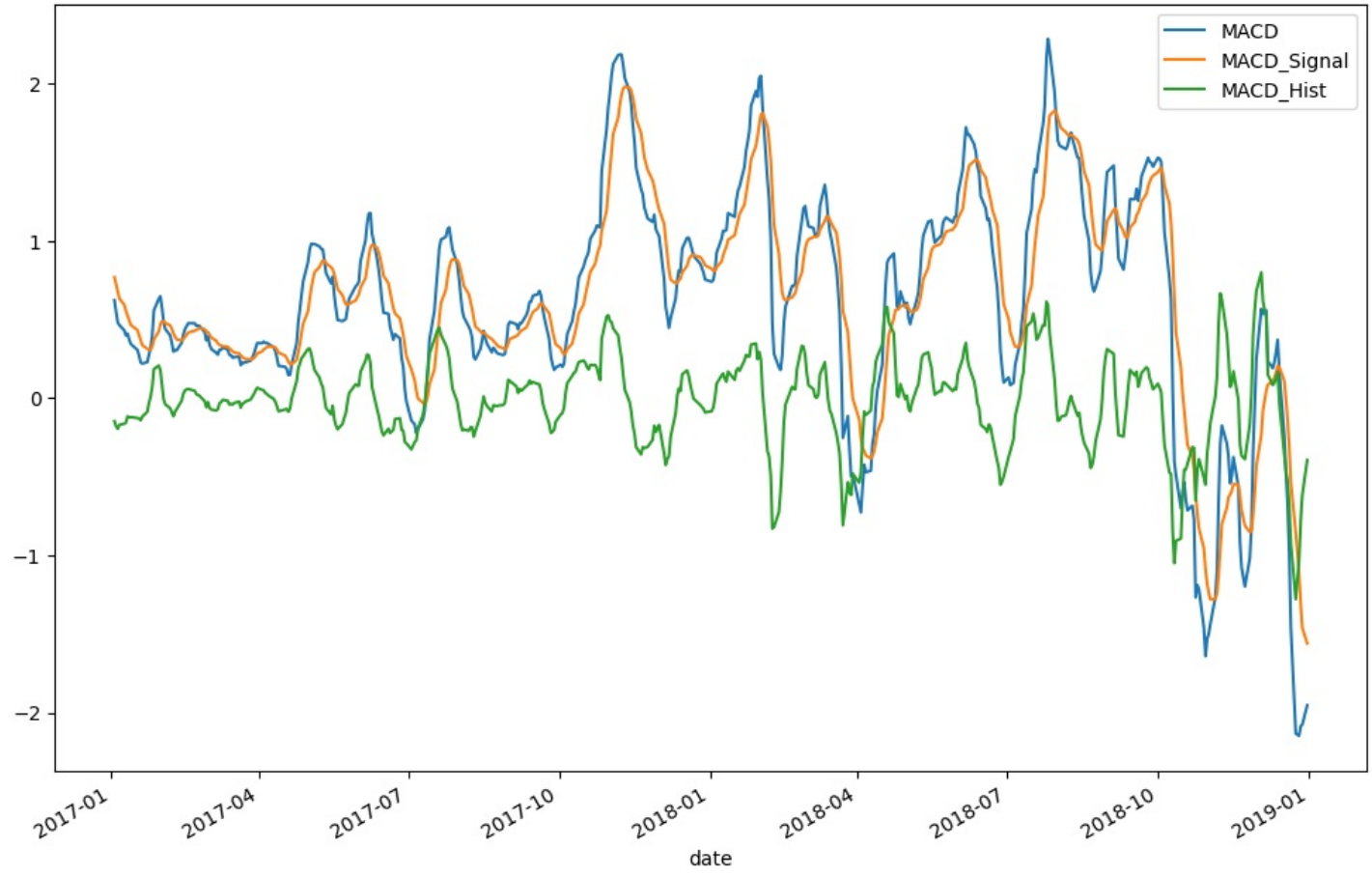
Out[ ]:

date	MACD	MACD_Signal	MACD_Hist
2023-04-14	6.5325	7.0655	-0.5330
2023-04-13	6.8940	7.1987	-0.3047
2023-04-12	6.8785	7.2749	-0.3964
2023-04-11	7.4052	7.3740	0.0312
2023-04-10	8.0369	7.3662	0.6707

In [ ]:

```
macd.loc["2017-01-02": "2018-12-31"].plot(figsize = (12, 8))
plt.show()
```

```
<ipython-input-68-dc05ad952808>:1: FutureWarning: Value based partial slicing on non-mono
tonic DatetimeIndexes with non-existing keys is deprecated and will raise a KeyError in a
future Version.
  macd.loc["2017-01-02": "2018-12-31"].plot(figsize = (12, 8))
```



In [ ]:

```
import pandas as pd
from alpha_vantage.foreignexchange import ForeignExchange
```

In [ ]:

```
fx = ForeignExchange(key= mykey, output_format='pandas')
```

In [ ]:

```
fx
```

Out[ ]:

```
<alpha_vantage.foreignexchange.ForeignExchange at 0x7f3977e3daf0>
```

In [ ]:

```
eurusd = fx.get_currency_exchange_daily("EUR", "USD", outputsize= "full")[0]
```

In [ ]:

```
eurusd
```

Out[ ]:

date	1. open	2. high	3. low	4. close
2023-04-14	1.10502	1.10759	1.09710	1.09910
2023-04-13	1.09914	1.10677	1.09750	1.10447
2023-04-12	1.09106	1.10004	1.09093	1.09914
2023-04-11	1.08586	1.09280	1.08569	1.09095
2023-04-10	1.08995	1.09173	1.08300	1.08590
...	...	...	...	...
2004-02-20	1.26970	1.27620	1.24890	1.25320
2004-02-19	1.27040	1.27450	1.26470	1.27320
2004-02-18	1.28340	1.29270	1.26550	1.27050
2004-02-17	1.28340	1.28770	1.27490	1.28540
2004-02-16	1.27400	1.27890	1.27110	1.27630

5000 rows × 4 columns

In [ ]:

```
usdeur = fx.get_currency_exchange_daily("USD", "EUR", outputsize= "full")[0]
```

In [ ]:

```
usdeur
```

Out[ ]:

date	1. open	2. high	3. low	4. close
2023-04-14	0.9047	0.9111	0.9027	0.9093
2023-04-13	0.9096	0.9105	0.9033	0.9050
2023-04-12	0.9159	0.9161	0.9086	0.9094
2023-04-11	0.9203	0.9205	0.9149	0.9161
2023-04-10	0.9165	0.9229	0.9155	0.9205
...	...	...	...	...
2014-11-28	0.8023	0.8044	0.8004	0.8028
2014-11-27	0.7993	0.8024	0.7983	0.8022
2014-11-26	0.8014	0.8034	0.7980	0.7993
2014-11-25	0.8037	0.8059	0.8007	0.8014
2014-11-24	0.8081	0.8085	0.8032	0.8036

2190 rows × 4 columns

In [ ]:

In [ ]:

```
#fx.get_currency_exchange_intraday("EUR", "USD", interval = "60min", outputsize= "full")[0] # PREMIUM now!
```

In [ ]:

```
#fx.get_currency_exchange_intraday("EUR", "USD", interval = "1min", outputsize= "full")[0] # PREMIUM now!
```

Cryptocurrencies

In [ ]:

```
import pandas as pd
from alpha_vantage.cryptocurrencies import CryptoCurrencies
```

In [ ]:

```
cc = CryptoCurrencies(key= mykey, output_format='pandas')
```

In [ ]:

```
cc
```

Out[ ]:

```
<alpha_vantage.cryptocurrencies.CryptoCurrencies at 0x7f3977daf640>
```

In [ ]:

```
BTC = cc.get_digital_currency_daily(symbol= "BTC", market = "EUR")
```

In [ ]:

```
BTC[1]
```

Out[ ]:

```
{'1. Information': 'Daily Prices and Volumes for Digital Currency',
 '2. Digital Currency Code': 'BTC',
 '3. Digital Currency Name': 'Bitcoin',
 '4. Market Code': 'EUR',
 '5. Market Name': 'Euro',
 '6. Last Refreshed': '2023-04-15 00:00:00',
 '7. Time Zone': 'UTC'}
```

In [ ]:

```
BTC[0]
```

Out[ ]:

date	1a. open (EUR)	1b. open (USD)	2a. high (EUR)	2b. high (USD)	3a. low (EUR)	3b. low (USD)	4a. close (EUR)	4b. close (USD)	5. volume	6. cap
2023-04-15	27703.579449	30466.93	27820.579080	30595.60	27653.376996	30411.72	27679.728510	30440.70	1363.167690	1363.
2023-04-14	27618.932712	30373.84	28188.300000	31000.00	27248.083800	29966.00	27703.579449	30466.93	75984.194520	75984.
2023-04-13	27177.222051	29888.07	27820.033500	30595.00	27146.778687	29854.59	27618.932712	30373.84	51934.117310	51934.
2023-04-12	27461.250999	30200.43	27720.919800	30486.00	26949.287820	29637.40	27177.222051	29888.07	62049.484510	62049.
2023-04-11	26949.242355	29637.35	27779.115000	30550.00	26906.187000	29590.00	27461.241906	30200.42	67990.076210	67990.

	...	...	...	...	...	...	...	...	...	...
2020-07-24	8732.25341[illegible]	9608.87	8762.924100	9667.00	8605.165992	9460.44	8672.781540	9537.86	43931.136205	43931.[illegible]
2020-07-23	8654.862888	9518.16	8787.475200	9664.00	8584.092069	9440.33	8732.253411	9603.27	51856.233500	51856.[illegible]
2020-07-22	8538.327000	9390.00	8678.359200	9544.00	8421.027300	9261.00	8654.862888	9518.16	48815.004107	48815.[illegible]
2020-07-21	8329.897254	9160.78	8581.727889	9437.73	8322.641040	9152.80	8538.327000	9390.00	60413.582486	60413.[illegible]
2020-07-20	8373.734607	9208.99	8385.128136	9221.52	8302.818300	9131.00	8329.897254	9160.78	35458.764082	35458.[illegible]

1000 rows × 10 columns

In [ ]:

```
cc.get_digital_currency_daily(symbol= "ETH", market = "USD")[0]
```

Out[ ]:

date	1a. open (USD)	1b. open (USD)	2a. high (USD)	2b. high (USD)	3a. low (USD)	3b. low (USD)	4a. close (USD)	4b. close (USD)	5. volume	6. market cap (USD)
2023-04-15	2099.99	2099.99	2106.11	2106.11	2092.36	2092.36	2099.30	2099.30	1.172463e+04	1.172463e+04
2023-04-14	2012.00	2012.00	2128.76	2128.76	2009.22	2009.22	2099.99	2099.99	8.963074e+05	8.963074e+05
2023-04-13	1917.40	1917.40	2023.00	2023.00	1899.55	1899.55	2012.01	2012.01	6.761645e+05	6.761645e+05
2023-04-12	1889.86	1889.86	1933.00	1933.00	1852.65	1852.65	1917.39	1917.39	6.308444e+05	6.308444e+05
2023-04-11	1910.21	1910.21	1937.37	1937.37	1881.11	1881.11	1889.86	1889.86	4.052035e+05	4.052035e+05
...	...	...	...	...	...	...	...	...	...	...
2020-07-24	275.31	275.31	287.34	287.34	266.00	266.00	279.15	279.15	9.260331e+05	9.260331e+05
2020-07-23	263.75	263.75	279.98	279.98	259.70	259.70	275.30	275.30	1.068775e+06	1.068775e+06
2020-07-22	245.53	245.53	269.61	269.61	241.51	241.51	263.74	263.74	7.184177e+05	7.184177e+05
2020-07-21	236.00	236.00	246.66	246.66	235.57	235.57	245.59	245.59	6.374621e+05	6.374621e+05
2020-07-20	239.12	239.12	239.72	239.72	234.05	234.05	236.01	236.01	3.966579e+05	3.966579e+05

1000 rows × 10 columns

In [ ]:

```
!jupyter nbconvert --to html "/content/60009200040_FMC_K2_Lab3.ipynb"
```

```
[NbConvertApp] Converting notebook /content/60009200040_FMC_K2_Lab3.ipynb to html
[NbConvertApp] Writing 1143880 bytes to /content/60009200040_FMC_K2_Lab3.html
```