```
# Install yfinance and matplotlib package
!pip install yfinance
!pip install matplotlib
```

```
# Import yfinance and matplotlib
import yfinance as yf
import matplotlib.pyplot as plt
import seaborn as sb
# Load Packages
import numpy as np
import pandas as pd
from pandas_datareader import data
```

Step 1: Choose seven (7) assets (preferably something with significant historical data)

```
# Read Data
test = data.DataReader(['SPY', 'AAPL','MSFT','TSLA','JNJ','UNH','NVDA'], 'yahoo', start='2
test.head()
```

Attributes	Adj Close						
Symbols	SPY	AAPL	MSFT	TSLA	JNJ	UNH	NVDA
Date							
2020-06-30	299.575195	90.074974	199.925888	215.962006	133.604034	286.754120	94.8(
2020-07-01	301.673615	89.904610	201.094925	223.925995	133.366531	289.456940	95.1:
2020-07-02	303.334930	89.904610	202.627457	241.731995	133.927048	289.972198	95.9-
2020-07-06	308.017609	92.309570	206.989273	274.316010	135.836639	294.395782	98.2'
2020-07-07	304.840698	92.023155	204.582428	277.971985	135.713104	288.406891	98.5:

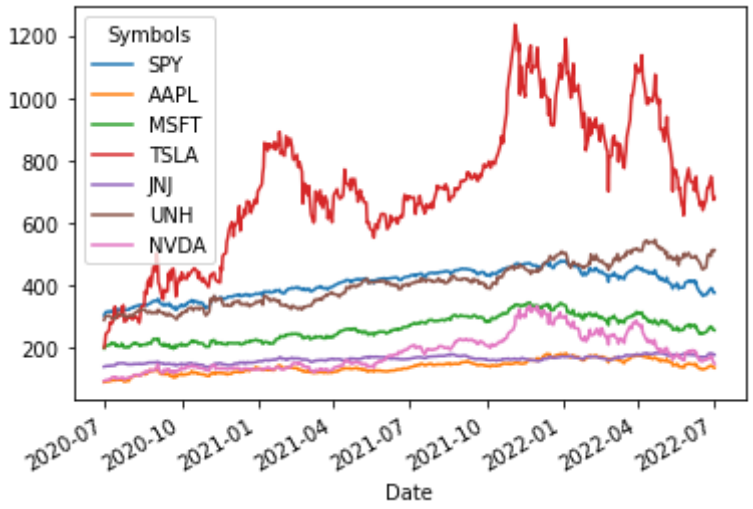
5 rows × 42 columns

Step 2: Retrieve daily open or close data on your assets for the previous 2 years

```
data_open=test['Open']
data_open.head()
```

Symbols	SPY	AAPL	MSFT	TSLA	JNJ	UNH
Date						
2020-06-30	303.989990	90.019997	197.880005	201.300003	139.399994	288.570007
2020-07-01	309.570007	91.279999	203.139999	216.600006	140.690002	295.829987
2020-07-02	314.239990	91.962502	205.679993	244.296005	141.250000	300.500000

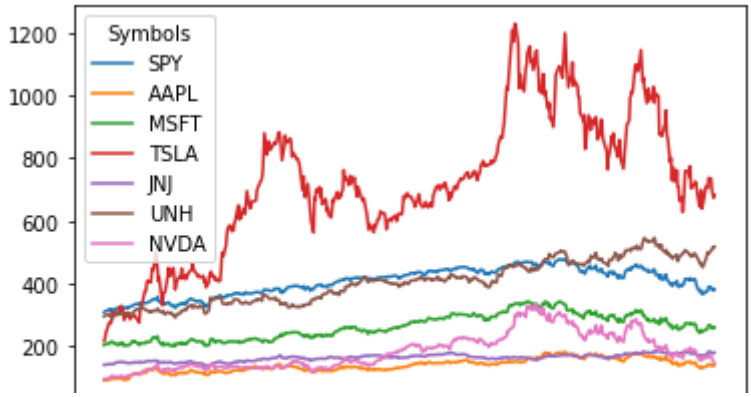
```
data_open.plot()
plt.show()
```




```
data_close=test['Close']
data_close.head()
```

Symbols	SPY	AAPL	MSFT	TSLA	JNJ	UNH
Date						
2020-06-30	308.359985	91.199997	203.509995	215.962006	140.630005	294.950012
2020-07-01	310.519989	91.027496	204.699997	223.925995	140.380005	297.730011
2020-07-02	312.230011	91.027496	206.259995	241.731995	140.970001	298.260010

```
data_close.plot()
plt.show()
```




```
# Closing price
test = test['Adj Close']
test.head()
```

Symbols	SPY	AAPL	MSFT	TSLA	JNJ	UNH
Date						
2020-06-30	299.575134	90.074982	199.925919	215.962006	133.604034	286.754150
2020-07-01	301.673615	89.904594	201.094910	223.925995	133.366531	289.456879
2020-07-02	303.334961	89.904594	202.627457	241.731995	133.927063	289.972198

Step 3: Calculate the mean, variance, and correlation matrix for all assets

```
# Log of percentage change
tesla = test.pct_change().apply(lambda x: np.log(1+x))
tesla.head()
```

Symbols	SPY	AAPL	MSFT	TSLA	JNJ	UNH	NVD
Date							
2020-06-30	NaN	NaN	NaN	NaN	NaN	NaN	Na
2020-07-01	0.006980	-0.001893	0.005830	0.036213	-0.001779	0.009381	0.00339
2020-07-02	0.005492	0.000000	0.007592	0.076514	0.004194	0.001779	0.00859

```
var_tesla = tesla.var()
var_tesla
```

```
Symbols
SPY     0.000124
AAPL    0.000410
MSFT    0.000304
TSLA    0.001762
JNJ     0.000105
UNH     0.000204
NVDA    0.001033
dtype: float64
```

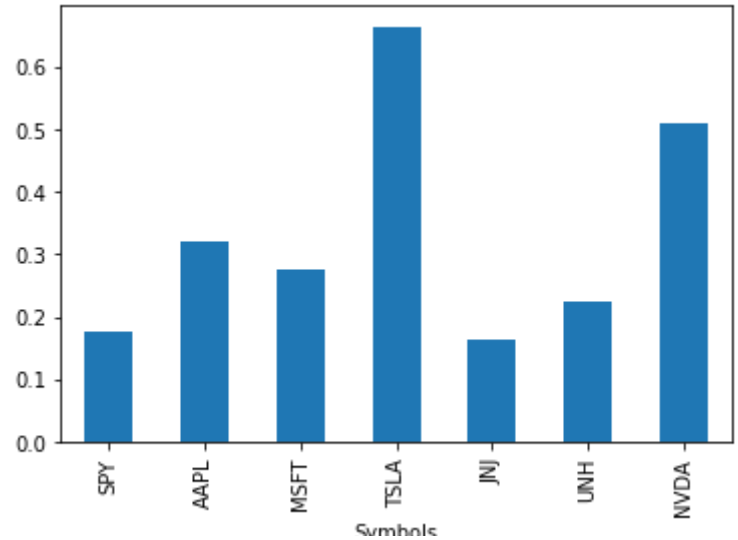
```
# Volatility
tesla_vol = np.sqrt(var_tesla * 250)

tesla_vol
```

```
Symbols
SPY     0.176232
AAPL    0.320006
MSFT    0.275544
TSLA    0.663637
JNJ     0.162112
UNH     0.225650
NVDA    0.508132
dtype: float64
```

```
test.pct_change().apply(lambda x: np.log(1+x)).std().apply(lambda x: x*np.sqrt(250)).plot(
```

```
<AxesSubplot:xlabel='Symbols'>
```

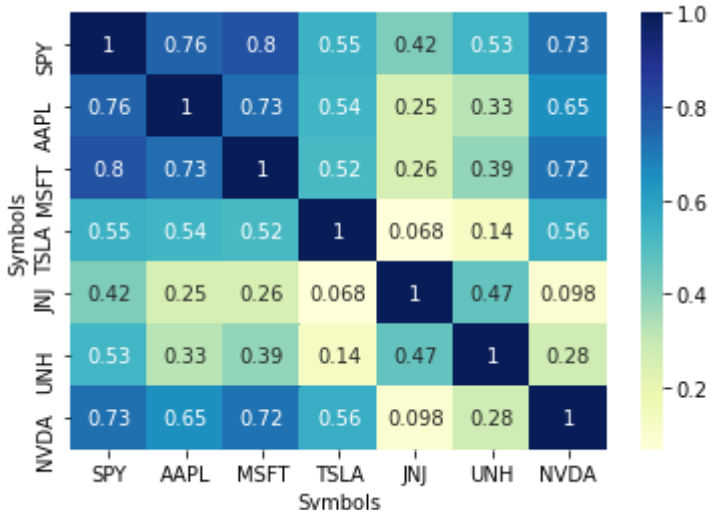



```
# Log of Percentage change
test1 = test.pct_change().apply(lambda x: np.log(1+x))
test1.head()
```

Symbols	SPY	AAPL	MSFT	TSLA	JNJ	UNH	NVD
Date							
2020-06-30	NaN	NaN	NaN	NaN	NaN	NaN	Nal

```
dataplot = sb.heatmap(test1.corr(), cmap="YlGnBu", annot=True)

# displaying heatmap
plt.show()
```



Step 4: Calculate efficient frontier and optimal weights for your portfolio (Don't forget the output!)

[ ] ↳ *3 cells hidden*

Step 5: Use MatPlotLib or Seaborn libraries to graph the results (matrix and frontier)Step 6: Output the calculated optimal weighting (identify portfolio weightings for optimum Sharpe Ratio)along with the efficient frontier and correlation matrix charts. Output to a saved file.

[ ] ↳ *15 cells hidden*