```
import datetime as dt
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
!pip install yfinance
!pip install pandas_datareader
import yfinance as yf
import pandas_datareader as web
import math
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: yfinance in /usr/local/lib/python3.7/dist-packages (0.1.7
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: appdirs>=1.4.4 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: lxml>=4.5.1 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: requests>=2.26 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: charset-normalizer<3,>=2 in /usr/local/lib/python3.7/dist
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packa
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: pandas_datareader in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: lxml in /usr/local/lib/python3.7/dist-packages (from panc
Requirement already satisfied: pandas>=0.23 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: charset-normalizer<3,>=2 in /usr/local/lib/python3.7/dist
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.7/dist-pa
```

```
ticker = ['TSLA', 'AAPL', 'MSFT', 'NVDA', 'AMZN', 'AMD', 'GME']
stock_data = yf.download(ticker, start = "2012-10-01", end = "2022-10-01")
```

```
[**********100%**********]  7 of 7 completed
```

```
#Portfolio Weight (equally Weighted)
#table_1 = pd.DataFrame[
weight = [1/7, 1/7, 1/7, 1/7, 1/7, 1/7, 1/7]
```

```
# Annualized Volatility (Using trailing 3 months)
stock_data_three_m = yf.download('TSLA, AAPL, MSFT, NVDA, AMZN, AMD, GME', start = "2022-07-0
end = "2022-10-01")
stock_data_three_m.dropna(inplace = True)

stock_data_three_m
stock_close = stock_data_three_m["Adj Close"]
Annual_volatility = stock_close.pct_change().std() * np.sqrt(252) * 100
Annual_volatility
```

```
[**********100%**********]  7 of 7 completed
AAPL    30.637832
AMD     50.077006
AMZN    43.809199
GME     77.382340
MSFT    29.388622
NVDA    55.231329
TSLA    49.097381
dtype: float64
```

```
# Beta against SPY
Trailing_12_months_SPY = yf.download('TSLA, AAPL, MSFT, NVDA, AMZN, AMD, GME, SPY', start = '
end = '2022-10-01')

beta_1 = Trailing_12_months_SPY['Adj Close']

returns = np.log(beta_1/beta_1.shift())

cov = returns.cov()
var = returns['SPY'].var()
```

```
[**********100%**********]  8 of 8 completed
```

```
b_TSLA_a_SPY = cov.loc['TSLA', 'SPY'] / var
b_AAPL_a_SPY = cov.loc['AAPL', 'SPY'] / var
b_MSFT_a_SPY = cov.loc['MSFT','SPY'] / var
b_NVDA_a_SPY = cov.loc['NVDA','SPY'] / var
b_AMZN_a_SPY = cov.loc['AMZN','SPY'] / var
b_AMD_a_SPY = cov.loc['AMD','SPY'] / var
b_GME_a_SPY = cov.loc['GME','SPY'] / var

print(b_TSLA_a_SPY)
print(b_AAPL_a_SPY)
print(b_MSFT_a_SPY)
print(b_NVDA_a_SPY)
print(b_AMZN_a_SPY)
print(b_AMD_a_SPY)
print(b_GME_a_SPY)
```

```
1.9094098874271694
1.2439007302686227
1.2121471080793855
2.276934482385174
1.6403531635107793
2.092113535503107
1.96821457512569
```

```
#Beta against IWM
Trailing_12_months_IWM = yf.download('TSLA, AAPL, MSFT, NVDA, AMZN, AMD, GME, IWM', start = '
end = '2022-10-01')

beta_2 = Trailing_12_months_IWM['Adj Close']
returns_2 = np.log(beta_2/beta_2.shift())

cov_2 = returns_2.cov()
var_2 = returns_2['IWM'].var()
```

```
[**********100%**********]  8 of 8 completed
```

```
b_TSLA_a_IWM = cov_2.loc['TSLA', 'IWM'] / var_2
b_AAPL_a_IWM = cov_2.loc['AAPL', 'IWM'] / var_2
b_MSFT_a_IWM = cov_2.loc['MSFT', 'IWM'] / var_2
b_NVDA_a_IWM = cov_2.loc['NVDA', 'IWM'] / var_2
b_AMZN_a_IWM = cov_2.loc['AMZN', 'IWM'] / var_2
b_AMD_a_IWM = cov_2.loc['AMD', 'IWM'] / var_2
b_GME_a_IWM = cov_2.loc['GME', 'IWM'] / var_2

print(b_TSLA_a_IWM)
print(b_AAPL_a_IWM)
print(b_MSFT_a_IWM)
print(b_NVDA_a_IWM)
print(b_AMZN_a_IWM)
print(b_AMD_a_IWM)
print(b_GME_a_IWM)
```

```
1.5930956672002419
0.8898966066669668
0.8342875975890516
1.8077307078840192
1.2143762165247858
1.7054697415319096
1.9940127685706825
```

```
#Beta against DIA

Trailing_12_months_DIA = yf.download('TSLA, AAPL, MSFT, NVDA, AMZN, AMD, GME, DIA', start = '
end = '2022-10-01')
```

```
beta_3 = Trailing_12_months_DIA['Adj Close']
returns_3 = np.log(beta_3/beta_3.shift())

cov_3 = returns_3.cov()
var_3 = returns_3['DIA'].var()
```

```
[**********100%**********]  8 of 8 completed
```

```
b_TSLA_a_DIA = cov_3.loc['TSLA', 'DIA'] / var_3
b_AAPL_a_DIA = cov_3.loc['AAPL', 'DIA'] / var_3
b_MSFT_a_DIA = cov_3.loc['MSFT', 'DIA'] / var_3
b_NVDA_a_DIA = cov_3.loc['NVDA', 'DIA'] / var_3
b_AMZN_a_DIA = cov_3.loc['AMZN', 'DIA'] / var_3
b_AMD_a_DIA = cov_3.loc['AMD', 'DIA'] / var_3
b_GME_a_DIA = cov_3.loc['GME', 'DIA'] / var_3

print(b_TSLA_a_DIA)
print(b_AAPL_a_DIA)
print(b_MSFT_a_DIA)
print(b_NVDA_a_DIA)
print(b_AMZN_a_DIA)
print(b_AMD_a_DIA)
print(b_GME_a_DIA)
```

```
1.918815573397074
1.3892295882266594
1.3227133894500074
2.373845926243097
1.7724447405815038
2.2122203014692015
2.001151931686538
```

```
Annual_stock = Trailing_12_months_SPY['Adj Close']
```

```
# Maximum Weekly Drawdown

TSLASUM = Annual_stock['TSLA'].sum()
TSLAMIN = Annual_stock['TSLA'].min()
TSLAMAX = Annual_stock['TSLA'].max()
tsla_max_drawdown = ((TSLAMIN - TSLAMAX ) / TSLAMAX) * 100
print(tsla_max_drawdown)

AAPLMIN = Annual_stock['AAPL'].min()
AAPLMAX = Annual_stock['AAPL'].max()
AAPL_max_drawdown = ((AAPLMIN - AAPLMAX ) / AAPLMAX) * 100
print(AAPL_max_drawdown)
```

```
MSFTMIN = Annual_stock['MSFT'].min()
MSFTMAX = Annual_stock['MSFT'].max()
MSFT_max_drawdown = ((MSFTMIN - MSFTMAX ) / MSFTMAX) * 100
print(MSFT_max_drawdown)

NVDAMIN = Annual_stock['NVDA'].min()
NVDAMAX = Annual_stock['NVDA'].max()
NVDA_max_drawdown = ((NVDAMIN - NVDAMAX ) / NVDAMAX) * 100
print(NVDA_max_drawdown)

AMZNMIN = Annual_stock['AMZN'].min()
AMZNMAX = Annual_stock['AMZN'].max()
AMZN_max_drawdown = ((AMZNMIN - AMZNMAX ) / AMZNMAX) * 100
print(AMZN_max_drawdown)

AMDMIN = Annual_stock['AMD'].min()
AMDMAX = Annual_stock['AMD'].max()
AMD_max_drawdown = ((AMDMIN - AMDMAX ) / AMDMAX) * 100
print(AMD_max_drawdown)

GMEMIN = Annual_stock['GME'].min()
GMEMAX = Annual_stock['GME'].max()
GME_max_drawdown = ((GMEMIN - GMEMAX ) / GMEMAX) * 100
print(GME_max_drawdown)
```

```
-48.926342817724446
-28.346214554194148
-31.67739580683949
-63.600400187117586
-44.63834407169731
-60.86714892393073
-68.4467785710266
```

```
#Average weekly drawdowns

Avg_weekly_TSLA = (Annual_stock['TSLA'].min() - Annual_stock['TSLA'].max()) / Annual_stock['T
Avg_weekly_AAPL = (Annual_stock['AAPL'].min() - Annual_stock['AAPL'].max()) / Annual_stock['A
Avg_weekly_MSFT = (Annual_stock['MSFT'].min() - Annual_stock['MSFT'].max()) / Annual_stock['M
Avg_weekly_NVDA = (Annual_stock['NVDA'].min() - Annual_stock['NVDA'].max()) / Annual_stock['N
Avg_weekly_AMZN = (Annual_stock['AMZN'].min() - Annual_stock['AMZN'].max()) / Annual_stock['A
Avg_weekly_AMD = (Annual_stock['AMD'].min() - Annual_stock['AMD'].max()) / Annual_stock['AMD'
Avg_weekly_GME = (Annual_stock['GME'].min() - Annual_stock['GME'].max()) / Annual_stock['GME'

print(Avg_weekly_TSLA)
print(Avg_weekly_AAPL)
print(Avg_weekly_MSFT)
print(Avg_weekly_NVDA)
print(Avg_weekly_AMZN)
print(Avg_weekly_AMD)
print(Avg_weekly_GME)
```

```
-0.4892634281772445
-0.2834621455419415
-0.31677395806839487
-0.6360040018711759
-0.4463834407169731
-0.6086714892393073
-0.684467785710266
```

```
#weekly drawdown redo
tickerSymbols = ['TSLA', 'AAPL', 'MSFT', 'NVDA', 'AMZN', 'AMD', 'GME']

TSLA = yf.Ticker(tickerSymbols[0])
AAPL = yf.Ticker(tickerSymbols[1])
MSFT = yf.Ticker(tickerSymbols[2])
NVDA = yf.Ticker(tickerSymbols[3])
AMZN = yf.Ticker(tickerSymbols[4])
AMD = yf.Ticker(tickerSymbols[5])
GME = yf.Ticker(tickerSymbols[6])

TSLA_df = TSLA.history(period='1y', start='2021-10-01', end='2022-10-01')
AAPL_df = AAPL.history(period='1y', start='2021-10-01', end='2022-10-01')
MSFT_df = MSFT.history(period='1y', start='2021-10-01', end='2022-10-01')
NVDA_df = NVDA.history(period='1y', start='2021-10-01', end='2022-10-01')
AMZN_df = AMZN.history(period='1y', start='2021-10-01', end='2022-10-01')
AMD_df = AMD.history(period='1y', start='2021-10-01', end='2022-10-01')
GME_df = GME.history(period='1y', start='2021-10-01', end='2022-10-01')

TSLA_low = TSLA_df['Low']
AAPL_low = AAPL_df['Low']
MSFT_low = MSFT_df['Low']
NVDA_low = NVDA_df['Low']
AMZN_low = AMZN_df['Low']
AMD_low = AMD_df['Low']
GME_low = GME_df['Low']


TSLA_high = TSLA_df['High']
AAPL_high = AAPL_df['High']
MSFT_high = MSFT_df['High']
NVDA_high = NVDA_df['High']
AMZN_high = AMZN_df['High']
AMD_high = AMD_df['High']
GME_high = GME_df['High']


TSLA_weekly_return = ( TSLA_low.mean() + TSLA_high.mean()) / TSLA_high.mean()
AAPL_weekly_return = ( AAPL_low.mean() + AAPL_high.mean()) / AAPL_high.mean()
MSFT_weekly_return = ( MSFT_low.mean() + MSFT_high.mean()) / MSFT_high.mean()
NVDA_weekly_return = ( NVDA_low.mean() + NVDA_high.mean()) / NVDA_high.mean()
```

```
AMZN_weekly_return = ( AMZN_low.mean() + AMZN_high.mean()) / AMZN_high.mean()
AMD_weekly_return = ( AMD_low.mean() + AMD_high.mean()) / AMD_high.mean()
GME_weekly_return = ( GME_low.mean() + GME_high.mean()) / GME_high.mean()
```

```
weekly = [TSLA_weekly_return, AAPL_weekly_return, MSFT_weekly_return, NVDA_weekly_return, AMZ
weeklydf = pd.DataFrame(weekly, index =['TSLA', 'AAPL', 'MSFT', 'NVDA', 'AMZN', 'AMD', 'GME']
weeklydf
```

	Average Weekly Returns
TSLA	1.949835
AAPL	1.975268
MSFT	1.976026
NVDA	1.951334
AMZN	1.968939
AMD	1.952577
GME	1.917347

```
# Total returns 10 years

total_returns_10_yrs = stock_data['Adj Close'].resample('D').ffill().pct_change()
total_return = total_returns_10_yrs.sum()
total_return
```

```
AAPL    2.342539
AMD     4.648745
AMZN    2.703305
GME     6.677241
MSFT    2.616051
NVDA    4.637565
TSLA    6.492387
dtype: float64
```

```
# Annualized total return
annualized_return = total_returns_10_yrs.mean()
print(annualized_return)
```

```
AAPL    0.000642
AMD     0.001273
AMZN    0.000740
GME     0.001829
MSFT    0.000717
NVDA    0.001270
```

```
TSLA    0.001778
dtype: float64
```

```
df_table_1 = {'Ticker':['TSLA', 'AAPL', 'MSFT', 'NVDA', 'AMZN', 'AMD', 'GME'],
              'Weight':[1/7, 1/7, 1/7, 1/7, 1/7, 1/7, 1/7],
              'Annual Vol':[49.097381, 30.637832, 29.388622, 55.231329, 43.809199, 50.077006,
              'BetaSPY':[1.90940, 1.24390, 1.21214, 2.27693, 1.64035, 2.09211, 1.96821],
              'BetaIWM':[1.59309, 0.88989, 0.83428, 1.80773, 1.21437, 1.70546, 1.99401],
              'BetaDIA':[1.91881, 1.38922, 1.32271, 2.37384, 1.77244, 2.21222, 2.00115],
              'AvgWeeklyDrawdown':[1.949835, 1.975268, 1.976026, 1.951334, 1.968939, 1.952577
              'MaxWeekDrawdown':[-0.48926, -0.28346, -0.31677, -0.63600, -0.44638, -0.60867,
              'totreturn':[6.492387, 2.342539, 2.616051, 4.637565, 2.703305, 2.703305, 6.6772
              'annualtotreturn':[0.001778, 0.000642, 0.000717, 0.001270, 0.000740, 0.001273,
#columns=['Ticker', 'Weight', 'Annual Vol', 'BetaSPY', 'BetaIWM', 'BetaDIA', 'AvgWeeklyDrawdo
table1 = pd.DataFrame(df_table_1)

table1
```

	Ticker	Weight	Annual Vol	BetaSPY	BetaIWM	BetaDIA	AvgWeeklyDrawdown	MaxWeekDra
0	TSLA	0.142857	49.097381	1.90940	1.59309	1.91881	1.949835	-0.4
1	AAPL	0.142857	30.637832	1.24390	0.88989	1.38922	1.975268	-0.2
2	MSFT	0.142857	29.388622	1.21214	0.83428	1.32271	1.976026	-0.3
3	NVDA	0.142857	55.231329	2.27693	1.80773	2.37384	1.951334	-0.6
4	AMZN	0.142857	43.809199	1.64035	1.21437	1.77244	1.968939	-0.4
5	AMD	0.142857	50.077006	2.09211	1.70546	2.21222	1.952577	-0.6
6	GME	0.142857	77.382340	1.96821	1.99401	2.00115	1.917347	-0.6

```
# P
stocks = ['TSLA', 'AAPL', 'MSFT', 'NVDA', 'AMZN', 'AMD', 'GME']
```

Double-click (or enter) to edit

```
# Correlation Against ETF
# # https://medium.com/analytics-vidhya/how-to-create-a-stock-correlation-matrix-in-python-4f

from datetime import datetime
import seaborn

start = dt.datetime(2021, 10, 1)
end = dt.datetime(2022, 10, 1)
symbols_list = ['TSLA', 'AAPL', 'MSFT', 'NVDA', 'AMZN', 'AMD', 'GME', 'SPY', 'IWM', 'DIA']
```

```
#array to store prices
symbols=[]

#array to store prices
!pip install --upgrade pandas
!pip install --upgrade pandas-datareader
symbols=[]
for ticker in symbols_list:
    r = web.DataReader(ticker, 'yahoo', start, end)
    # add a symbol column
    r['Symbol'] = ticker
    symbols.append(r)
# concatenate into df
df = pd.concat(symbols)
df = df.reset_index()
df = df[['Date', 'Adj Close', 'Symbol']]
df.head()
df_pivot=df.pivot('Date','Symbol','Adj Close').reset_index()
df_pivot.head()

df_pivot['the_portfolio'] = df_pivot[stocks].mean(axis=1)



df_pivot = df_pivot.set_index('Date')
df_pivot = df_pivot.pct_change()
df_pivot
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (1.3.5)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: pandas-datareader in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: lxml in /usr/local/lib/python3.7/dist-packages (from panc
Requirement already satisfied: pandas>=0.23 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: charset-normalizer<3,>=2 in /usr/local/lib/python3.7/dist
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packa
```

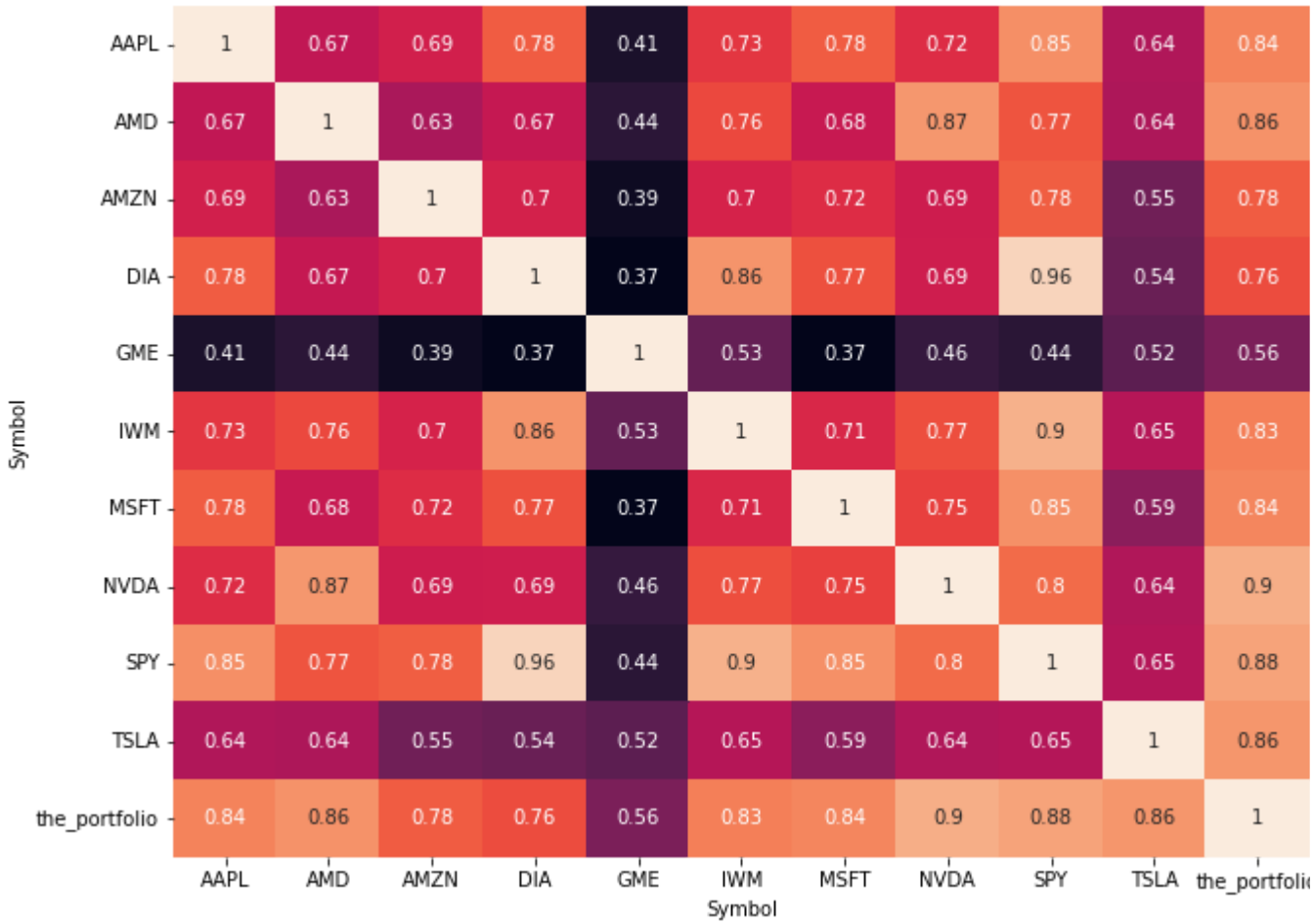
Symbol	AAPL	AMD	AMZN	DIA	GME	IWM	MSFT	NVDA

```
corr_df = df_pivot.corr(method='pearson')
#reset symbol as index (rather than 0-X)
corr_df.head().reset_index()
#del corr_df.index.name
corr_df.head(10)
```

Symbol	AAPL	AMD	AMZN	DIA	GME	IWM	MSFT	NVDA	
Symbol									
AAPL	1.000000	0.666582	0.694926	0.784407	0.411661	0.733798	0.782726	0.719279	0.8
AMD	0.666582	1.000000	0.631191	0.673519	0.438288	0.758699	0.677954	0.866123	0.7
AMZN	0.694926	0.631191	1.000000	0.698595	0.389356	0.702093	0.718726	0.687315	0.7
DIA	0.784407	0.673519	0.698595	1.000000	0.365957	0.859255	0.768873	0.688803	0.9
GME	0.411661	0.438288	0.389356	0.365957	1.000000	0.530527	0.366122	0.455927	0.4
IWM	0.733798	0.758699	0.702093	0.859255	0.530527	1.000000	0.708530	0.766755	0.9
MSFT	0.782726	0.677954	0.718726	0.768873	0.366122	0.708530	1.000000	0.747579	0.8
NVDA	0.719279	0.866123	0.687315	0.688803	0.455927	0.766755	0.747579	1.000000	0.8
SPY	0.851766	0.772184	0.784683	0.962344	0.435882	0.903508	0.854463	0.801879	1.0
TSLA	0.643037	0.637821	0.549189	0.537551	0.517511	0.650603	0.586673	0.642247	0.6

```
plt.figure(figsize=(13, 8))
seaborn.heatmap(corr_df, annot=True)
plt.figure()
```

```
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 432x288 with 0 Axes>
```

```
#Covariance of Portfolio against ETF
cov_df_SPY = df_pivot['SPY'].cov(df_pivot['the_portfolio'])

cov_df_IWM = df_pivot['IWM'].cov(df_pivot['the_portfolio'])
cov_df_DIA = df_pivot['DIA'].cov(df_pivot['the_portfolio'])

new_cov_SPY = cov_df_SPY * 10000
new_cov_IWM = cov_df_IWM * 10000
new_cov_DIA = cov_df_DIA * 10000
print(new_cov_SPY)
print(new_cov_IWM)
print(new_cov_DIA)

#print(cov_df_SPY)
#print(cov_df_IWM)
#print(cov_df_DIA)
```

```
3.2435800654600775
3.6923404884129476
2.323092185090982
```

```
from numpy.ma.core import true_divide
```

```
# Tracking Error

# Tracking Error = standard deviation (stock return - benchmark return)


closeddata = web.DataReader(['TSLA', 'AAPL', 'MSFT', 'NVDA', 'AMZN', 'AMD', 'GME', 'SPY', 'IW

closeddata['stock_return'] = closeddata['TSLA'].pct_change() + closeddata['AAPL'].pct_change(
closeddata['bench_return'] = closeddata['SPY'].pct_change() + closeddata['IWM'].pct_change()
closeddata['Tracking_Error'] = closeddata['stock_return'] - closeddata['bench_return']

closeddata['SPY_ERROR_pct'] = closeddata['SPY'].pct_change()
closeddata['IWM_ERROR_pct'] = closeddata['IWM'].pct_change()
closeddata['DIA_ERROR_pct'] = closeddata['DIA'].pct_change()

closeddata['Tracking_Error_SPY'] = closeddata['stock_return'] - closeddata['SPY_ERROR_pct']
closeddata['Tracking_Error_IWM'] = closeddata['stock_return'] - closeddata['IWM_ERROR_pct']
closeddata['Tracking_Error_DIA'] = closeddata['stock_return'] - closeddata['DIA_ERROR_pct']

TE_SPY = closeddata['Tracking_Error_SPY'].std()*100
TE_IWM = closeddata['Tracking_Error_IWM'].std()*100
TE_DIA = closeddata['Tracking_Error_DIA'].std()*100
print(TE_SPY)
print(TE_IWM)
print(TE_DIA)
TE = closeddata['Tracking_Error'].std()*100

closeddata.tail()
```

```
     13.23747411191036
     13.124680129961716
```

```
#Sharpe Ratio (using current risk-free rate)
# https://www.codearmo.com/blog/sharpe-sortino-and-calmar-ratios-python
def sharpe_ratio(return_series, N, rf):
    mean = return_series.mean() * N -rf
    sigma = return_series.std() * np.sqrt(N)
    return mean / sigma

N = 255 #255 trading days in a year
rf =0.03 #3% risk free rate as of 2022
sharpes = df_pivot.apply(sharpe_ratio, args=(N,rf,),axis=0)

sharpes
```

```
     Symbol
     AAPL            -0.015054
     AMD             -0.569866
     AMZN            -0.667581
     DIA             -0.962204
     GME             -0.120758
     IWM             -1.071019
     MSFT            -0.616208
     NVDA            -0.605076
     SPY             -0.857424
     TSLA             0.318793
     the_portfolio   -0.401212
     dtype: float64
```

```
#Annualized Volatility(252 days)
tickerSymbols = ['SPY', 'IWM', 'DIA']
stocks = ['TSLA', 'AAPL', 'MSFT', 'NVDA', 'AMZN', 'AMD', 'GME']

SPY = yf.Ticker(tickerSymbols[0])
IWM = yf.Ticker(tickerSymbols[1])
DIA = yf.Ticker(tickerSymbols[2])

SPY_df = SPY.history(period='1y', start='2021-10-01', end='2022-10-01')
IWM_df = IWM.history(period='1y', start='2021-10-01', end='2022-10-01')
DIA_df = DIA.history(period='1y', start='2021-10-01', end='2022-10-01')

SPY = SPY_df['Close']
IWM = IWM_df['Close']
DIA = DIA_df['Close']

# stocks

TSLA_df = TSLA.history(period='1y', start='2021-10-01', end='2022-10-01')
AAPL_df = AAPL.history(period='1y', start='2021-10-01', end='2022-10-01')
MSFT_df = MSFT.history(period='1y', start='2021-10-01', end='2022-10-01')
NVDA_df = NVDA.history(period='1y', start='2021-10-01', end='2022-10-01')
```

```
AMZN_df = AMZN.history(period='1y', start='2021-10-01', end='2022-10-01')
AMD_df = AMD.history(period='1y', start='2021-10-01', end='2022-10-01')
GME_df = GME.history(period='1y', start='2021-10-01', end='2022-10-01')

TSLA = TSLA_df['Close']
AAPL = AAPL_df['Close']
MSFT = MSFT_df['Close']
NVDA = NVDA_df['Close']
AMZN = AMZN_df['Close']
AMD = AMD_df['Close']
GME = GME_df['Close']

stocks_df = pd.concat([TSLA, AAPL, MSFT, NVDA, AMZN, AMD, GME], axis='columns', join='inner')
new_stocks = stocks_df.mean(axis=1)
new_stocks = new_stocks.pct_change()

#Annualized Volatility of Each Stock
etfs_df = pd.concat([SPY, IWM, DIA], axis='columns', join='inner')
etfs_df = etfs_df.pct_change()
ETFVolatility = etfs_df.std() * np.sqrt(252)
stockVolatility = new_stocks.std() * np.sqrt(252)
new_spread = stockVolatility - ETFVolatility


print(new_spread)
```

```
Close    0.204480
Close    0.159151
Close    0.242671
dtype: float64
```

```
ETFVolatility
```

```
Close    0.218993
Close    0.264321
Close    0.180801
dtype: float64
```

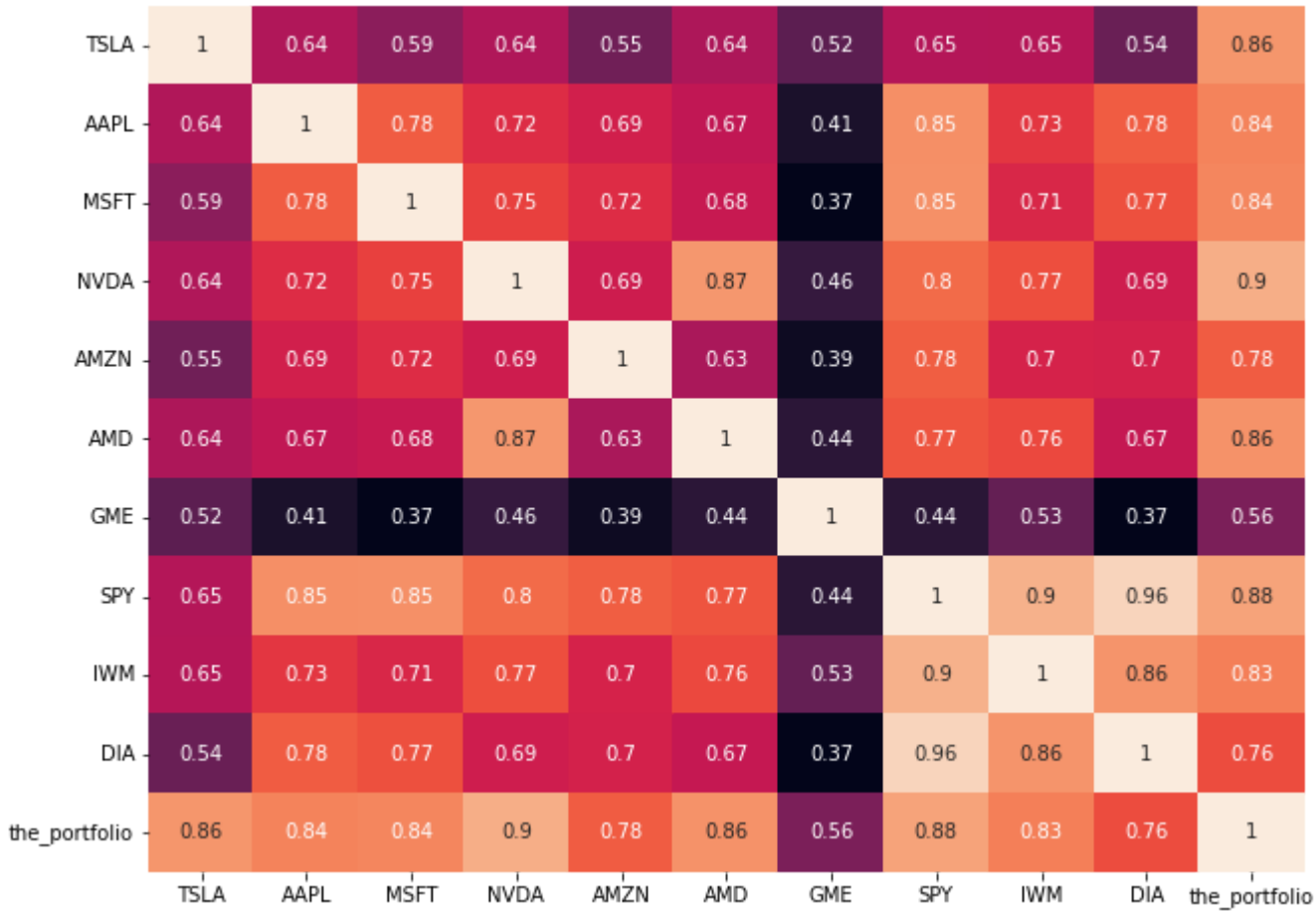
```
# Table 2
df_table_2 = {'Ticker':['SPY', 'IWM', 'DIA'],
              'Correlation':[0.88, 0.83, 0.76],
              'Covariance':[3.24358, 3.69234, 2.32309],
              'Tracking Error':[13.23718, 13.12440, 13.32778],
              'Sharpe Ratio':[-0.857424, -1.071019, -0.962204],
              'Spread':[0.204480, 0.159151, 0.242671]}

table2 = pd.DataFrame(df_table_2)
```

```
#Correlation Matrix


df_corr_matrix = pd.DataFrame(df_pivot, columns = ['TSLA', 'AAPL', 'MSFT', 'NVDA', 'AMZN', 'A

corrMatrix = df_corr_matrix.corr()
plt.figure(figsize=(13, 8))
sns.heatmap(corrMatrix,annot=True)
plt.show()
```




```
table1
```

	Ticker	Weight	Annual Vol	BetaSPY	BetaIWM	BetaDIA	AvgWeeklyDrawdown	MaxWeekDra

table2

	Ticker	Correlation	Covariance	Tracking Error	Sharpe Ratio	Spread
0	SPY	0.88	3.24358	13.23718	-0.857424	0.204480
1	IWM	0.83	3.69234	13.12440	-1.071019	0.159151
2	DIA	0.76	2.32309	13.32778	-0.962204	0.242671

```
#sources
#https://medium.com/analytics-vidhya/how-to-create-a-stock-correlation-matrix-in-python-4f32f
#https://www.codearmo.com/blog/sharpe-sortino-and-calmar-ratios-python
#https://r-shuo-wang.medium.com/portfolio-analysis-basics-returns-and-drawdowns-70c5f7a0eb3d
#
```