

```
pip install yfinance
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: yfinance in /usr/local/lib/python3.7/dist-packages (0.1.74)
Requirement already satisfied: requests>=2.26 in /usr/local/lib/python3.7/dist-packages (from yfinance) (2.28)
Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.7/dist-packages (from yfinance) (1.21.6)
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.7/dist-packages (from yfinance)
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.7/dist-packages (from yfinance) (1.3.
Requirement already satisfied: lxml>=4.5.1 in /usr/local/lib/python3.7/dist-packages (from yfinance) (4.9.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas>
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.24.0->y
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests>=2
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests
Requirement already satisfied: charset-normalizer<3,>=2 in /usr/local/lib/python3.7/dist-packages (from reque
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests>=2.26->y
```

```
# importing all the necessary modules
```

```
import pandas as pd
import yfinance as yf
import datetime as dt
import numpy as np
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

▼ PART 1: Create a table showing constituent (stocks) risk analysis in the equal-weight portfolio analysis as of the current date.

```
# setting list of items into variables: tickers and etf, then using yfinance to retrieve data, passing in ticker a
tickers=['AAPL', 'GE', 'MSFT', 'MCD', 'JPM', 'SBUX', 'TSLA']
etf=['SPY', 'IWM', 'DIA']
df= yf.download(tickers+etf, period = '10y')
```

```
[*****100%*****] 10 of 10 completed
```

```
# taking only the Adj Close column from the dataframe and looking at the first 5 dataset
df=df['Adj Close']
df.head()
```

	AAPL	DIA	GE	IWM	JPM	MCD	MSFT	SBUX	SPY	TSLA
Date										
2012-07-23	18.438765	102.170410	123.597275	67.811508	26.135422	67.235947	23.906790	21.289309	111.781586	6.132
2012-07-24	18.349909	101.325111	122.859001	66.920990	26.355501	66.570724	23.800636	21.238823	110.821754	5.968
2012-07-25	17.557487	101.848434	123.043571	67.069405	26.689405	66.555603	23.539360	21.209373	110.846626	5.790
2012-07-26	17.554741	103.514915	126.488754	67.645653	27.175074	67.281311	23.808805	22.046637	112.675301	5.626
2012-07-27	17.868647	105.092827	128.703568	69.252083	27.994654	67.424965	24.298697	19.972401	114.752205	5.902

```
# create portfolio column by taking the mean of the tickers across row, then take the percentage change between th
df['portfolio'] = df[tickers].mean(axis=1)
returns= df.pct_change()
```

```
# creating the first table, with index set to tickers
table_1=pd.DataFrame(index= tickers)
```

```
# share weighted
table_1['weight']=1/len(tickers)
'''
price weighted - alternate method: not in the final table
1/7 x(AAPL + GOOG + MSFT + MCD + JPM + SBUX + TSLA) = (df.mean(axis=1)).mean()
table_1['p_weighted']=(df.mean(axis=1)).mean()/ df.mean()
'''
```

```
# Annualized Volatility calculation(using trailing 3-months)
returns_trail_3_month= returns[-63:] # trailing 3-months percentage change
table_1['annual. volat.']= (returns_trail_3_month.var()/returns_trail_3_month.std())**(1/np.sqrt(4))
```

```

'''
volatility = np.sqrt(returns_trail_3_month.var())
table_1['volatility_annual'] = (volatility*np.sqrt(252/4))
'''

# calculate beta
beta= returns[-252:].cov()/returns[-252:].var()
for item in etf:
    table_1[item + ' beta']= beta[item]

'''
log_returns = np.log(data/data.shift())
cov = log_returns.cov()
beta_SPY = []
beta_IWM = []
beta_DIA = []
for stock in tickers:
    var = log_returns[stock].var()
    for bench in benches:
        beta = cov.loc[bench, stock]/var
        if bench == 'SPY':
            beta_SPY.append(beta)
        if bench == 'IWM':
            beta_IWM.append(beta)
        if bench == 'DIA':
            beta_DIA.append(beta)
'''

'\nlog_returns = np.log(data/data.shift())\ncov = log_returns.cov()\nbeta_SPY = []\nbeta_IWM = []\nbeta_DIA =
returns[stock].var()\n    for bench in benches:\n        beta = cov.loc[bench, stock]/var\n            if bench =
a)\n        if bench == 'IWM':\n            beta_IWM.append(beta)\n            if bench == 'DIA':\n                bet

# Drawdown caluclation
drawdown_5roll= returns[-252:].rolling(5).max()- returns[-252:].rolling(5).min()
table_1['avg_drawdown']=drawdown_5roll.mean()
table_1['max_drawdown']= drawdown_5roll.max()

```

```

'''
def Max_drawdown(df, window=5, draw_type = 'max') :
    if draw_type == 'avg':
        Roll_Max = df.rolling(window,min_periods=1).mean()
    else:
        Roll_Max = df.rolling(window,min_periods=1).max()

    Daily_Drawdown= df/Roll_Max -1

    Max_Daily_Drawdown = Daily_Drawdown.rolling(window, min_periods=1).min()

    return Max_Daily_Drawdown

avg_drawdown = (Max_drawdown(data[tickers], draw_type = 'avg').resample('Y').mean()).to_numpy()
avg_drawdown

max_drawdown = (Max_drawdown(data[tickers]).resample('Y').mean()).to_numpy()
max_drawdown
'''


'\ndef Max_drawdown(df, window=5, draw_type = 'max') :
    if draw_type == 'avg':
        Roll_Max = df.rolling(window,min_periods=1).mean()
    else:
        Roll_Max = df.rolling(window,min_periods=1).max()
    Daily_Drawdown= df/Roll_Max -1
    Max_Daily_Drawdown = Daily_Drawdown.rolling(window, min_periods=1).min()
    return Max_Daily_Drawdown
navg_drawdown = (Max_drawdown(data[tickers], draw_type = 'avg').resample('Y').mean()).to_numpy()
nmax_drawdown = (Max_drawdown(data[tickers]).resample('Y').mean()).to_numpy()

# Calculating total return and annualized total returns
table_1['total_return']= df.pct_change(len(df)-1)[-1:].T
table_1['annualized returns']= table_1.total_return**(1/np.sqrt(252))

'''
table_1['total_return']= (df[tickers].pct_change(len(df)-1).iloc[-1])
table_1['annualized returns']= table_1.total_return**(1/np.sqrt(252))
'''

```

```
\ntable_1['total_return'] = (df[tickers].pct_change(len(df)-1).iloc[-1])\ntable_1['annualized returns'] = tabl
table_1.T
```

	AAPL	GE	MSFT	MCD	JPM	SBUX	TSLA	
weight	0.142857	0.142857	0.142857	0.142857	0.142857	0.142857	0.142857	
annual. volat.	0.160189	0.162785	0.151832	0.119110	0.142105	0.164500	0.217774	
SPY beta	1.256238	1.075426	1.206873	0.560692	0.874446	1.130200	1.930769	
IWM beta	0.841479	0.909269	0.771039	0.352354	0.647306	0.871695	1.559033	
DIA beta	1.364488	1.358478	1.275809	0.731947	1.184283	1.297671	1.884880	
avg_drawdown	0.042526	0.046989	0.041438	0.025024	0.037634	0.043161	0.088774	
max_drawdown	0.096724	0.116774	0.089921	0.071181	0.080772	0.149301	0.222318	
total_return	7.356850	-0.448289	9.890629	2.777592	3.390976	2.926384	132.191452	
annualized returns	1.133957	NaN	1.155296	1.066470	1.079959	1.069981	1.360264	

▼ PART 2: Create a table showing Portfolio Risk against the three ETFs

```
# creating the second table, with index set to etf
table_2 = pd.DataFrame(index= etf)
```

```
# Calculating correlation and covariance of portfolio against ETF
table_2['correlation'] = returns[etf+['portfolio']].corr().portfolio
table_2['covariance'] = returns[etf+['portfolio']].cov().portfolio * 10000
```

```
# Tracking error calculation : Standard Deviation of (P - B), where P is portfolio return and B is benchmark retu
for item in etf:
    table_2.loc[item, 'tracking_error'] = (returns[item] - returns.portfolio).std() * 100
```

```
# Calculate sharpe ratio: SR = R_p-R_f/std_P
excess_return = (returns.portfolio[-252:] - (0.02 / np.sqrt(252)))
table_2['sharpe'] = 0
for item in etf:
    table_2.loc[item, 'sharpe'] = (excess_return / (returns[-252:].portfolio- returns[-252:][item]).std())[-1]

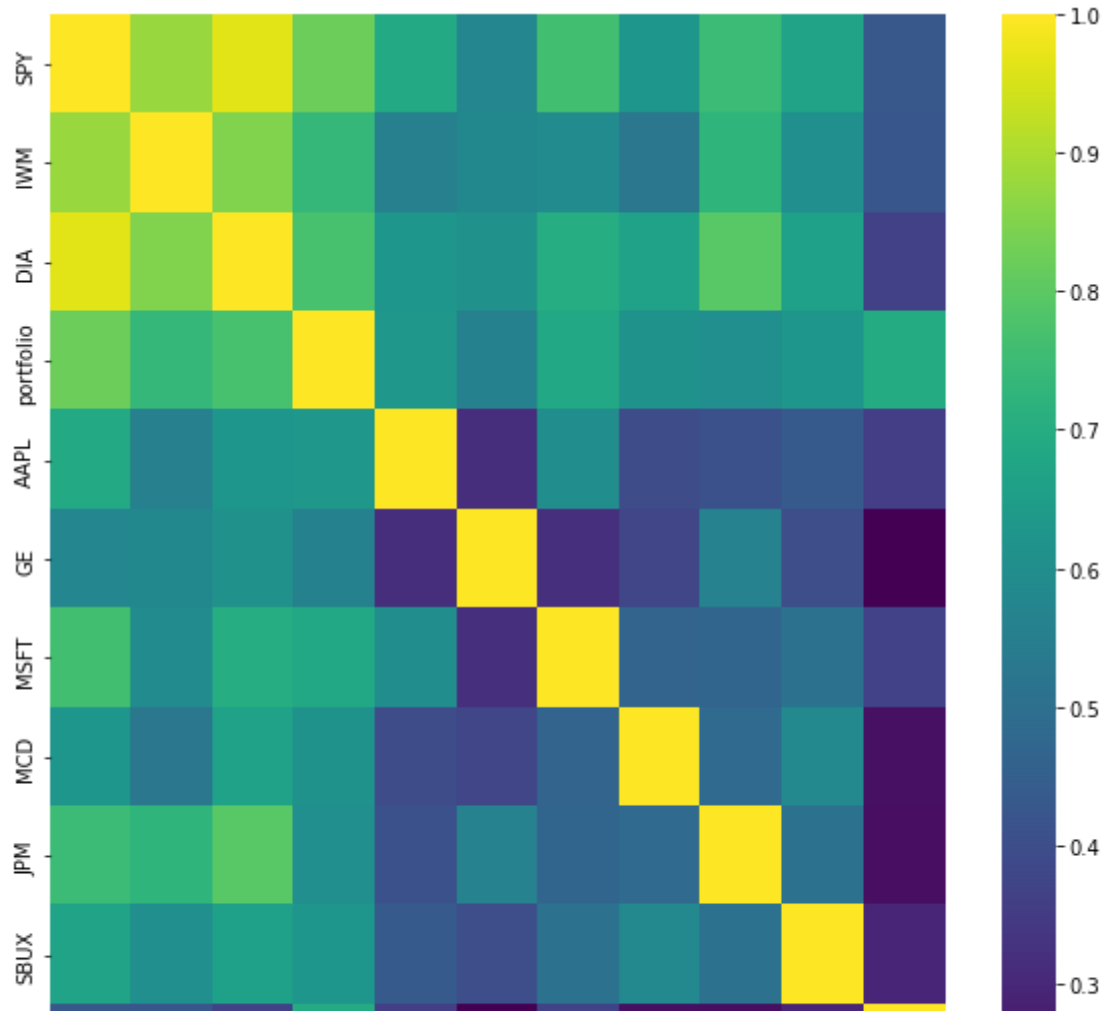
# Calculate volatility spread
table_2["volatility spread"] = returns[etf + ['portfolio']][-252:].std() ** (1/252)
table_2.T
```

	SPY	IWM	DIA	
correlation	0.821756	0.736038	0.770869	
covariance	1.289910	1.450811	1.221064	
tracking_error	0.857359	1.030724	0.945181	
sharpe	-0.230779	-0.229894	-0.200835	
volatility spread	0.982825	0.983705	0.982113	

PART 3: Create a correlation matrix showing the correlations between the equal-weighted portfolio, 3 ETFs, and your 7 stocks.

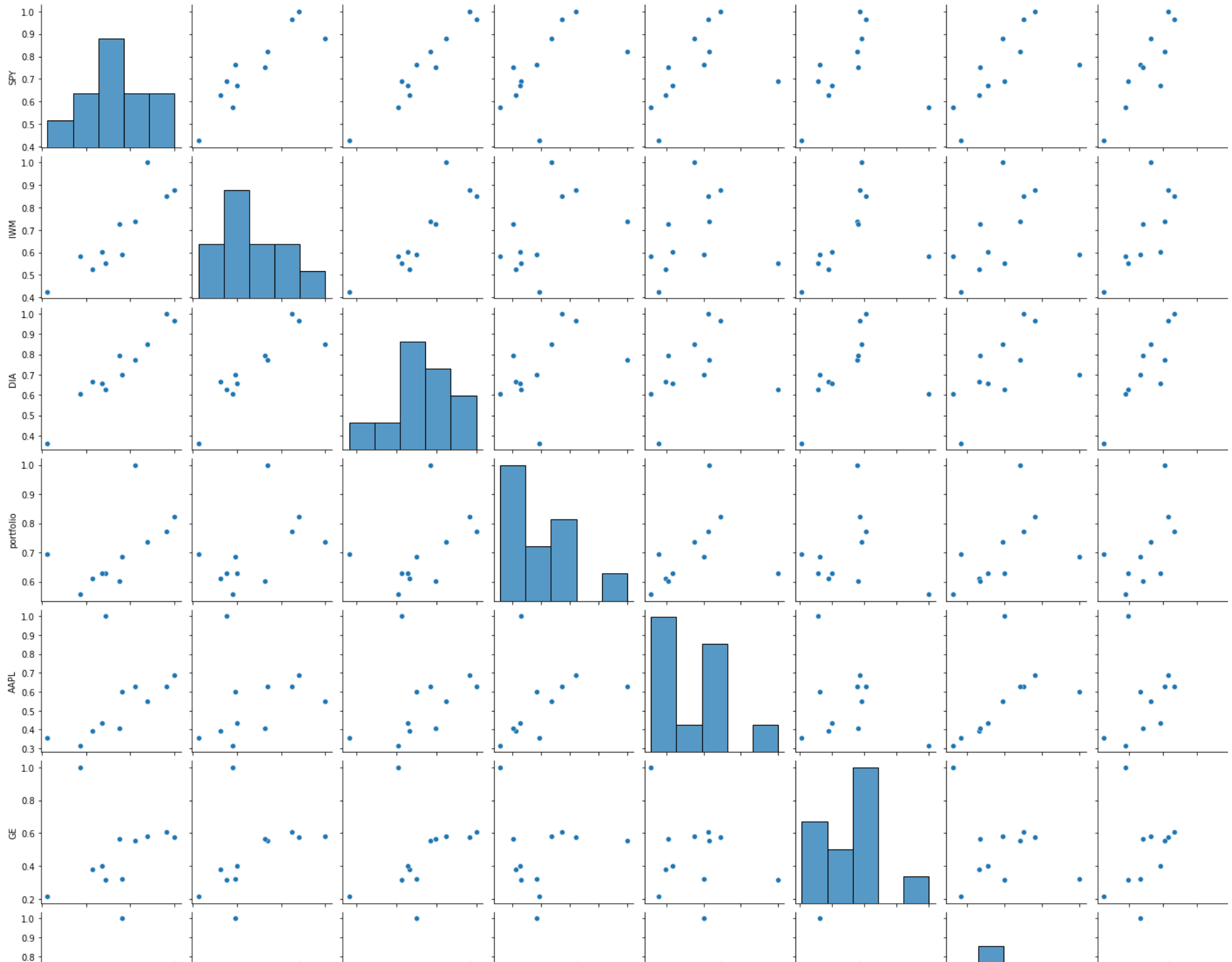
```
fig, ax = plt.subplots(figsize=(10, 10))
sns.heatmap(returns[etf+['portfolio']+tickers].corr(), cmap = 'viridis')

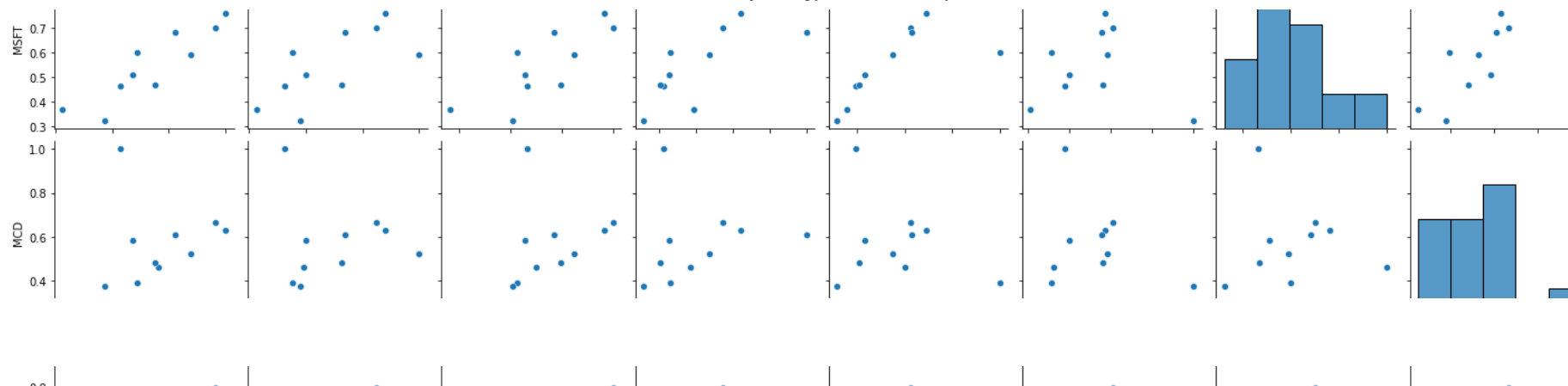
plt.show()
```



```
sns.pairplot(returns[etf+['portfolio']+tickers].corr())
```

<seaborn.axisgrid.PairGrid at 0x7fb090d08810>





▼ CODE SOURCE:

```
'''
We want acknowledge the cited works from professor John Driescher
'''

'\nCode Source:\nWe want acknowledge the cited works from professor John Driescher\n'
```

✓ 45s completed at 1:26 PM

