```
import numpy as np
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
import pandas_datareader as pdr
import datetime
import pandas_datareader.data as web
import ffn
import plotly.express as px

import yfinance as yf

Microsoft = yf.Ticker("MSFT").history(period='5y')
Apple= yf.Ticker("AAPL").history(period='5y')
MSFT=Microsoft['Close']
AAPL= Apple['Close']
JPMorgan= yf.Ticker("JPM").history(period='5y')
JPM= JPMorgan['Close']

AAPL.mean()

88.53988112676134

AAPL.median()

69.73873138427734

MSFT.mode()

0      68.984871
1      70.406731
2      71.838028
3      78.325981
4      79.145195
5      90.327789
6      96.315254
7      98.971344
8     101.766563
9     102.081032
10    102.447182
11    105.035431
12    107.086876
13    132.242783
14    133.900253
15    134.484451
16    134.795059
17    135.438660
18    153.788254
19    204.582428
20    216.313095
21    254.080002
22    293.021790
```

```
23    298.804047
dtype: float64
```

```
MSFT.describe()
```

```
count    1259.000000
mean      177.754387
std        79.782912
min        67.939636
25%       103.273239
50%       158.399643
75%       249.507164
max       341.606354
Name: Close, dtype: float64
```

```
msft_return= MSFT.pct_change(1)
aapl_return= AAPL.pct_change(1)
```

```
bins= 1+3.322* np.log(1259)
bins
```

```
24.71267861909532
```

```
msft_return.skew()
aapl_return.skew()
```

```
-0.10350579509128915
```

```
msft_return.kurtosis()
```

```
7.86276375372009
```

```
msft_return.var()
```

```
0.00035002410653443475
```

```
msft_return.std()
```

```
0.018708931197009483
```

```
stock= pd.concat([msft_return, aapl_return],axis=1)
stock.columns= ['MSFT_return','AAPL_return']
stock
```

```
            MSFT_return  AAPL_return
Date
2017-08-14          NaN          NaN
2017-08-15     0.000273     0.010947
2017-08-16     0.005873    -0.004022
2017-08-17    -0.016972    -0.019199
2017-08-18     0.001243    -0.002280
...                 ...          ...
2022-08-08    -0.009155    -0.002903
```

```
2022-08-09     0.007063     0.000303
2022-08-10     0.024300     0.026195
2022-08-11    -0.007401    -0.004432
2022-08-12     0.017037     0.021426

[1259 rows x 2 columns]
```

```
stock.cov()
```

```
             MSFT_return  AAPL_return
MSFT_return     0.000350     0.000288
AAPL_return     0.000288     0.000410
```

```
stock.cov()*252
```

```
             MSFT_return  AAPL_return
MSFT_return     0.088206     0.072518
AAPL_return     0.072518     0.103285
```

```
stock.corr()
```

```
             MSFT_return  AAPL_return
MSFT_return     1.000000     0.759768
AAPL_return     0.759768     1.000000
```

```
MSFT.plot(label='Microsft Close', figsize=(15,5))
AAPL.plot(label='Apple Close', figsize=(15,5))
JPM.plot(label='JP Morgan Close', figsize=(15,5))

plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f680cca6a10>
```

```
Microsoft['Volume'].plot(label='Microsft Volume', figsize=(15,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6828db72d0>
```

```
Microsoft_ttl_trded= Microsoft['Volume']* Microsoft['Close']

Microsoft_ttl_trded.plot(label='Microsft Total Traded',
figsize=(15,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6828db72d0>
```

```
Apple_ttl_traded= Apple['Close']* Apple['Volume']

Morgan_ttl_traded= JPMorgan['Close']* JPMorgan['Volume']

Microsoft_ttl_trded.plot(label='Microsft Total Traded',
figsize=(15,5))
Apple_ttl_traded.plot(label='Apple Total Traded', figsize=(15,5))
Morgan_ttl_traded.plot(label='Morgan Total Traded', figsize=(15,5))
plt.xlabel('Date')
```

```
plt.ylabel('Total Traded')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f680d190cd0>
```

```
MSFT_MA_50=Microsoft['Close'].rolling(50).mean()
AAPL_MA_50=Apple['Close'].rolling(50).mean()
JPM_MA_50=JPMorgan['Close'].rolling(50).mean()
MSFT_MA_100=Microsoft['Close'].rolling(100).mean()
AAPL_MA_100=Apple['Close'].rolling(100).mean()
JPM_MA_100=JPMorgan['Close'].rolling(100).mean()
MSFT_MA_200=Microsoft['Close'].rolling(200).mean()
AAPL_MA_200=Apple['Close'].rolling(200).mean()
JPM_MA_200=JPMorgan['Close'].rolling(200).mean()
```

```
MSFT_MA_50.plot(label='Microsft MA50', figsize=(15,5))
MSFT_MA_100.plot(label='Microsft MA100', figsize=(15,5))
MSFT_MA_200.plot(label='Microsft MA200', figsize=(15,5))
MSFT.plot(label='Microsft Close', figsize=(15,5))
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f680d11d950>
```

```
AAPL_MA_50.plot(label='Apple MA50', figsize=(15,5))
AAPL_MA_100.plot(label='Apple MA100', figsize=(15,5))
AAPL_MA_200.plot(label='Apple MA200', figsize=(15,5))
AAPL.plot(label='Apple Close', figsize=(15,5))
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f680c777d50>
```

```
JPM_MA_50.plot(label='JPMorgan MA50', figsize=(15,5))
JPM_MA_100.plot(label='JPMorgan MA100', figsize=(15,5))
JPM_MA_200.plot(label='JPMorgan MA200', figsize=(15,5))
JPM.plot(label='JPMorgan Close', figsize=(15,5))
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f680ca0b5d0>
```

```
Microsoft['MACD']= Microsoft['Close'].ewm(span=12,
adjust=False).mean()-  Microsoft['Close'].ewm(span=26,
adjust=False).mean()
Microsoft['Baseline']=0
```

```
Microsoft['MACD'].plot(label='MSFT MACD', figsize=(15,5))
Microsoft['Baseline'].plot(label='Baseline')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f680c46c350>
```

```
JPMorgan['MACD']= JPMorgan['Close'].ewm(span=12, adjust=False).mean()-
JPMorgan['Close'].ewm(span=26, adjust=False).mean()
```

```
JPMorgan['Baseline']=0

JPMorgan['MACD'].plot(label='JPM MACD', figsize=(15,5))
JPMorgan['Baseline'].plot(label='Baseline')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f680c3ba750>
```

```
Apple['MACD']= Apple['Close'].ewm(span=12, adjust=False).mean()-
Apple['Close'].ewm(span=26, adjust=False).mean()
Apple['Baseline']=0

Apple['MACD'].plot(label='Apple MACD', figsize=(15,5))
Apple['Baseline'].plot(label='Baseline')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f68228c50d0>
```

```
window_of_days= 20
number_std=2

rolling_mean= Microsoft['Close'].rolling(window_of_days).mean()
rolling_std= Microsoft['Close'].rolling(window_of_days).std()

Microsoft['Rolling_Mean']= rolling_mean
Microsoft['Bollinger_High']= rolling_mean+ rolling_std*number_std
Microsoft['Bolling_low']= rolling_mean- rolling_std*number_std

Microsoft['Rolling_Mean'].plot(label='MSFT Mean', figsize=(15,5))
Microsoft['Bollinger_High'].plot(label='MSFT High', figsize=(15,5))
Microsoft['Bolling_low'].plot(label='MSFT Low', figsize=(15,5))
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f680c41e350>
```

```
import yfinance as yf
etf = ['RELIANCE.NS','SUNPHARMA.NS','ITC.NS','BPCL.NS'] #and any
tickers you'd add to be retrived
Stock = yf.download(tickers=etf, period='5y')
Portfolio=Stock['Close']
```

```
[**********************100%**********************]  4 of 4 completed
```

```
Portfolio_return= Portfolio/Portfolio.shift(1)
Portfolio_return
```

Date	BPCL.NS	ITC.NS	RELIANCE.NS	SUNPHARMA.NS
2017-08-14	NaN	NaN	NaN	NaN
2017-08-16	1.011707	1.029309	0.995582	1.029462
2017-08-17	1.020147	1.002865	1.000639	1.005250
2017-08-18	1.019445	1.006607	1.005296	0.962929
2017-08-21	0.988973	0.999113	0.993018	0.979581

```
...              ...       ...          ...          ...
2022-08-05  1.007336  1.004362     0.985264     0.997224
2022-08-08  0.968044  1.007399     1.013082     0.993941
2022-08-10  1.008138  0.993773     1.005979     1.010215
2022-08-11  1.000152  0.984094     1.003330     1.001468
2022-08-12  1.016903  1.007510     1.016171     0.991749

[1236 rows x 4 columns]
```

```
Portfolio_Cov= Portfolio_return.cov()
Portfolio_Cov
```

```
               BPCL.NS    ITC.NS  RELIANCE.NS  SUNPHARMA.NS
BPCL.NS       0.000601  0.000134     0.000179      0.000118
ITC.NS        0.000134  0.000283     0.000088      0.000085
RELIANCE.NS   0.000179  0.000088     0.000401      0.000110
SUNPHARMA.NS  0.000118  0.000085     0.000110      0.000414
```

```
annualised_cov= Portfolio_Cov*252
annualised_cov
```

```
               BPCL.NS    ITC.NS  RELIANCE.NS  SUNPHARMA.NS
BPCL.NS       0.151569  0.033731     0.045174      0.029824
ITC.NS        0.033731  0.071316     0.022126      0.021440
RELIANCE.NS   0.045174  0.022126     0.101089      0.027774
SUNPHARMA.NS  0.029824  0.021440     0.027774      0.104343
```

```
covariance_mkt= annualised_cov.iloc[0,1]
covariance_mkt
```

```
0.03373144960149097
```

```
portfolio_weights= np.array([0.25,0.25,0.25,0.25])
```

```
portfolio_return_weighted= Portfolio_return.mul(portfolio_weights,
axis=1)
portfolio_return_weighted
```

```
             BPCL.NS    ITC.NS  RELIANCE.NS  SUNPHARMA.NS
Date
2017-08-14       NaN       NaN          NaN           NaN
2017-08-16  0.252927  0.257327     0.248895      0.257365
2017-08-17  0.255037  0.250716     0.250160      0.251313
2017-08-18  0.254861  0.251652     0.251324      0.240732
2017-08-21  0.247243  0.249778     0.248254      0.244895
...              ...       ...          ...           ...
2022-08-05  0.251834  0.251090     0.246316      0.249306
2022-08-08  0.242011  0.251850     0.253271      0.248485
2022-08-10  0.252034  0.248443     0.251495      0.252554
2022-08-11  0.250038  0.246023     0.250833      0.250367
2022-08-12  0.254226  0.251878     0.254043      0.247937
```

```
[1236 rows x 4 columns]
```

```
Portfolio_return['Portfolio']=
portfolio_return_weighted.sum(axis=1).dropna()
Portfolio_return
```

```
             BPCL.NS    ITC.NS  RELIANCE.NS  SUNPHARMA.NS  Portfolio
Date
2017-08-14       NaN       NaN          NaN           NaN   0.000000
2017-08-16  1.011707  1.029309     0.995582      1.029462   1.016515
2017-08-17  1.020147  1.002865     1.000639      1.005250   1.007225
2017-08-18  1.019445  1.006607     1.005296      0.962929   0.998569
2017-08-21  0.988973  0.999113     0.993018      0.979581   0.990171
...              ...       ...          ...           ...        ...
2022-08-05  1.007336  1.004362     0.985264      0.997224   0.998546
2022-08-08  0.968044  1.007399     1.013082      0.993941   0.995616
2022-08-10  1.008138  0.993773     1.005979      1.010215   1.004526
2022-08-11  1.000152  0.984094     1.003330      1.001468   0.997261
2022-08-12  1.016903  1.007510     1.016171      0.991749   1.008083

[1236 rows x 5 columns]
```

```
tickers='NIFTYBEES.NS'
NIFTY50 = yf.download(tickers=tickers, period='5y')
Portfolio_return['Benchmark']= NIFTY50['Close']
```

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

```
Portfolio_return['Benchmark']=
Portfolio_return['Benchmark'].pct_change(1).dropna()
Portfolio_return= Portfolio_return.dropna()
```

```
Portfolio_return['RF Rate']=0.073
Portfolio_return
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

```
             BPCL.NS    ITC.NS  RELIANCE.NS  SUNPHARMA.NS
Portfolio  \
Date

2017-08-16  1.011707  1.029309     0.995582      1.029462   1.016515
```

```
2017-08-17  1.020147  1.002865     1.000639      1.005250   1.007225

2017-08-18  1.019445  1.006607     1.005296      0.962929   0.998569

2017-08-21  0.988973  0.999113     0.993018      0.979581   0.990171

2017-08-22  1.025314  1.004616     0.997219      1.021496   1.012161

...              ...       ...          ...           ...        ...

2022-08-05  1.007336  1.004362     0.985264      0.997224   0.998546

2022-08-08  0.968044  1.007399     1.013082      0.993941   0.995616

2022-08-10  1.008138  0.993773     1.005979      1.010215   1.004526

2022-08-11  1.000152  0.984094     1.003330      1.001468   0.997261

2022-08-12  1.016903  1.007510     1.016171      0.991749   1.008083
```

Date	Benchmark	RF Rate
2017-08-16	0.011384	0.073
2017-08-17	0.000745	0.073
2017-08-18	-0.005403	0.073
2017-08-21	-0.009950	0.073
2017-08-22	0.001511	0.073
...	...	...
2022-08-05	0.000423	0.073
2022-08-08	0.008137	0.073
2022-08-10	0.001310	0.073
2022-08-11	0.006805	0.073
2022-08-12	0.002704	0.073

```
[1235 rows x 7 columns]
```

```
Covariance_portfolio=Portfolio_return.cov()*252
```

```
covariance_market= Covariance_portfolio.iloc[3,4]
covariance_market
```

```
0.04584519005332168
```

```
Market_variance= Portfolio_return['Benchmark'].var()*252
Market_variance
```

```
16.60810364196477
```

```
portfolio_beta= covariance_market/Market_variance
portfolio_beta
```

```
0.0027604108838459843
```

```
Portfolio_return_mean= Portfolio_return['Portfolio'].mean()
```

```
risk_free_rate= 0.073
```

```
alpha= Portfolio_return -
( risk_free_rate+portfolio_beta*(Portfolio_return-risk_free_rate))
alpha
```

```
             BPCL.NS    ITC.NS  RELIANCE.NS  SUNPHARMA.NS
Portfolio  \
Date

2017-08-16  0.936116  0.953669     0.920035      0.953821   0.940910

2017-08-17  0.944532  0.927298     0.925078      0.929677   0.931646

2017-08-18  0.943832  0.931030     0.929723      0.887472   0.923014

2017-08-21  0.913444  0.923557     0.917478      0.904078   0.914639

2017-08-22  0.949685  0.929045     0.921668      0.945878   0.936569

...              ...       ...          ...           ...        ...

2022-08-05  0.931757  0.928791     0.909746      0.921673   0.922992

2022-08-08  0.892573  0.931820     0.937487      0.918399   0.920070

2022-08-10  0.932556  0.918231     0.930404      0.934628   0.928955

2022-08-11  0.924593  0.908579     0.927762      0.925905   0.921710

2022-08-12  0.941297  0.931931     0.940567      0.916212   0.932502


            Benchmark  RF Rate
Date
2017-08-16  -0.061446      0.0
2017-08-17  -0.072056      0.0
2017-08-18  -0.078187      0.0
2017-08-21  -0.082721      0.0
2017-08-22  -0.071292      0.0
...               ...      ...
2022-08-05  -0.072377      0.0
2022-08-08  -0.064684      0.0
```

```
2022-08-10  -0.071492      0.0
2022-08-11  -0.066012      0.0
2022-08-12  -0.070102      0.0

[1235 rows x 7 columns]
```

```
Portfolio_return['Portfolio']= Portfolio_return['Portfolio']
Portfolio_return
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Date	BPCL.NS	ITC.NS	RELIANCE.NS	SUNPHARMA.NS	Portfolio	\
2017-08-16	1.011707	1.029309	0.995582	1.029462	1.016515	
2017-08-17	1.020147	1.002865	1.000639	1.005250	1.007225	
2017-08-18	1.019445	1.006607	1.005296	0.962929	0.998569	
2017-08-21	0.988973	0.999113	0.993018	0.979581	0.990171	
2017-08-22	1.025314	1.004616	0.997219	1.021496	1.012161	
...	...	...	...	...	...	...
2022-08-05	1.007336	1.004362	0.985264	0.997224	0.998546	
2022-08-08	0.968044	1.007399	1.013082	0.993941	0.995616	
2022-08-10	1.008138	0.993773	1.005979	1.010215	1.004526	
2022-08-11	1.000152	0.984094	1.003330	1.001468	0.997261	
2022-08-12	1.016903	1.007510	1.016171	0.991749	1.008083	

Date	Benchmark	RF Rate
2017-08-16	0.011384	0.073

```
2017-08-17   0.000745    0.073
2017-08-18  -0.005403    0.073
2017-08-21  -0.009950    0.073
2017-08-22   0.001511    0.073
...               ...      ...
2022-08-05   0.000423    0.073
2022-08-08   0.008137    0.073
2022-08-10   0.001310    0.073
2022-08-11   0.006805    0.073
2022-08-12   0.002704    0.073

[1235 rows x 7 columns]
```

```
var99= np.percentile(Portfolio_return['Portfolio'],1)
var99
```

```
0.9681457460311951
```