GWP 3

Theme 2:

Statistical Related Risk: Volatility and Correlation

Volatility and Correlation are two of the most important statistical risk that investors has to deal with when investing in the financial market

Statistical Related Risk:Volatility

Definition

Volatility is one one the major subject of discussion in the stock market as it affect a lot of investors, to define volatility I will cite five(5) different authors:

Hull, J. C. (2017). Options, Futures, and Other Derivatives (10th Ed.). Pearson. "Volatility is a measure of the amount by which an asset price is expected to fluctuate over a given period of time."

Fama, E. F. (1965). The Behavior of Stock-Market Prices. The Journal of Business, 38(1), 34-105. "The volatility of common stock prices is of interest to investors and students of the behavior of securities markets. Roughly, volatility is the magnitude of fluctuations in the market value of the common stock of a corporation."

Jorion, P. (2007). Financial Risk Manager Handbook (5th Ed.). John Wiley & Sons. "Volatility is a statistical measure of the dispersion of returns for a given security or market index. It is a measure of the degree of variation of a security's price over time."

Schwager, J. D. (1996). Market Wizards: Interviews with Top Traders. HarperCollins. "Volatility is the magnitude of price moves, regardless of direction, that the market has experienced over a specified period."

Pindyck, R. S., & Rubinfeld, D. L. (1998). Econometric Models and Economic Forecasts (4th Ed.). McGraw-Hill. "Volatility refers to the degree to which a time series deviates from its mean or trend over time.

Volatility Equation

Standard deviation is a measure of volatility as it calculates the degree of variation of an asset's price or returns around its mean. The formula for standard deviation is:

$\sigma = \text{sqrt}(\Sigma(x - \mu)^2 / N)$

where: x = each observation of the variable μ = the mean of the variable N = the total number of observations

In the context of financial markets, the variable is typically the returns of an asset, and the standard deviation is a measure of its volatility.

To annualize the standard deviation of daily returns, the formula for statistical volatility multiplies the standard deviation by the square root of the number of trading days in a year:

$SV = \sigma * \text{sqrt}(N)$

(Source: Hull, J. C. (2017). Options, Futures, and Other Derivatives (10th Ed.). Pearson.)

Types of Volatility

Historical volatility: based on past price fluctuations of an asset. (Hull, 2017)

Implied volatility: a market-based measure of the expected future volatility of an asset, based on the prices of options on that asset. (Hull, 2017)

Realized volatility: quantifies the actual amount of price fluctuation of an asset over a specific time period. (Jorion, 2007)

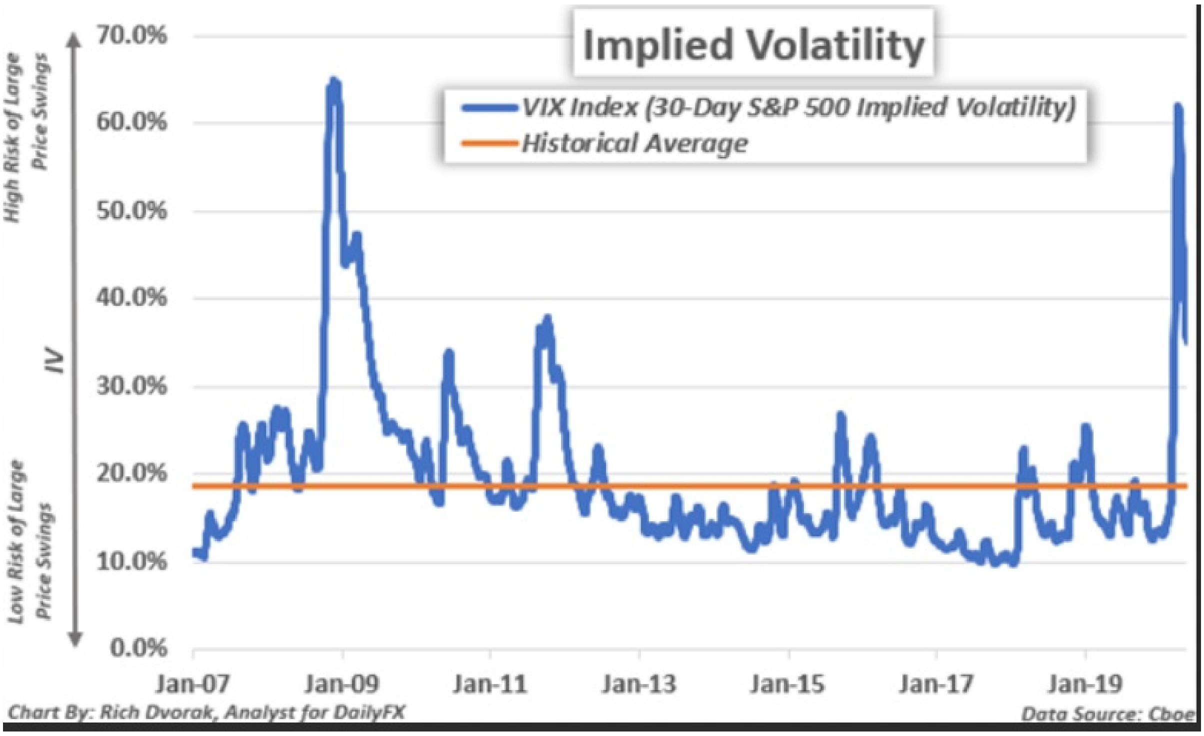
Conditional volatility: measures the volatility of an asset that is dependent on certain conditions, such as changes in interest rates or economic indicators. (Engle, 2002)

Relative volatility: compares the volatility of one asset to another, usually within the same market or industry. (Hull, 2017)

Volatility Diagram:

some diagrams on different types of volatility


RELIANCE
O:789.53 H:790.98 V:5.63 M
C:783.28 L:780.60 D:08/28/2017
Historical Volatility
HIST VOL (10,252,1)
945.30
900.00
850.00
800.00
750.00
700.00
30
21.06
10
Jul
14
Aug
14
Sep
14
Oct
14
Nov
1D 5D 1M 3M 6M YTD 1Y 5Y All


Implied Volatility
VIX Index (30-Day S&P 500 Implied Volatility)
Historical Average
High Risk of Large Price Swings
IV
Low Risk of Large Price Swings
70.0%
60.0%
50.0%
40.0%
30.0%
20.0%
10.0%
0.0%
Jan-07
Jan-09
Jan-11
Jan-13
Jan-15
Jan-17
Jan-19
Chart By: Rich Dvorak, Analyst for DailyFX
Data Source: Cboe

Statistical related Rsik : Correlation

Definition

Correlation is also one of the statistical related risk, and is known as the directional movement between two variables.

I have provided below 5 definitions of correlation by different authors:

Bodie, Z., Kane, A., & Marcus, A. J. (2014). Investments (10th Ed.). McGraw-Hill Education. "Correlation measures the degree to which two variables move together over time. In finance, the correlation coefficient is used to measure the extent to which the returns on two assets are related." (p. 320)

Hull, J. C. (2017). Options, Futures, and Other Derivatives (10th Ed.). Pearson. "Correlation measures the extent to which the prices of two different assets are related." (p. 160)

Jorion, P. (2007). Financial Risk Manager Handbook (5th Ed.). John Wiley & Sons. "Correlation measures the strength of the linear relationship between two variables, such as the returns on two different assets." (p. 93)

Pindyck, R. S., & Rubinfeld, D. L. (1998). Econometric Models and Economic Forecasts (4th Ed.). McGraw-Hill. "Correlation measures the degree to which two time series move together. In finance, correlation is often used to describe the relationship between the returns on two different assets." (p. 719)

Wilmott, P., Howison, S., & Dewynne, J. (2013). The Mathematics of Financial Derivatives: A Student Introduction (2nd Ed.). Cambridge University Press. "Correlation is a measure of the linear dependence between two random variables, such as the returns on two different assets. In finance, correlation is used to describe the relationship between the returns on different assets in a portfolio." (p. 31)

Equation:

Statistical risk, also known as systematic risk, is a type of market risk that is related to the overall market movements and cannot be diversified away by holding a diversified portfolio of assets. Correlation is a statistical measure that helps to quantify the degree to which two assets are related to each other in terms of their price movements. In finance, correlation is often used to analyze the relationship between different asset classes or securities, such as stocks, bonds, and commodities.

The calculation of correlation involves measuring the strength and direction of the linear relationship between two variables. In the financial market, correlation is typically measured using the correlation coefficient, which ranges from -1 to +1. A correlation coefficient of +1 indicates a perfect positive correlation, where the two assets move in the same direction at the same time, while a correlation coefficient of -1 indicates a perfect negative correlation, where the two assets move in opposite directions. A correlation coefficient of 0 indicates no correlation, meaning that there is no linear relationship between the two assets.

It's important to note that correlation is not causation, and a high correlation between two assets does not necessarily imply a causal relationship between them. Therefore, it is important to exercise caution when interpreting correlation results and to consider other factors, such as fundamental analysis and market trends, when making investment decisions.

The formula for calculating the correlation coefficient is:

$r = Cov(X,Y) / (\sigma X * \sigma Y)$

where: $Cov(X,Y)$ = the covariance of the two variables X and Y

σX = the standard deviation of X

σY = the standard deviation of Y

Types of Correlation

There are several ways to calculate correlation in the financial market, including:

Pearson correlation coefficient: The Pearson correlation coefficient is the most commonly used method for calculating correlation. It measures the linear relationship between two variables, and is calculated by dividing the covariance between the two variables by the product of their standard deviations. The formula for Pearson correlation coefficient is:

$r = (\Sigma(x - \bar{x})(y - \bar{y})) / (\sqrt{\Sigma(x - \bar{x})^2}) (\sqrt{\Sigma(y - \bar{y})^2}))$

where: r = Pearson correlation coefficient

x = the values of the first variable

y = the values of the second variable

$\bar{x}$ = the mean of the first variable

$\bar{y}$ = the mean of the second variable

Spearman rank correlation coefficient: The Spearman rank correlation coefficient is used when the variables are not normally distributed, or when the relationship between the variables is non-linear. It measures the degree of association between two variables by calculating the correlation between their ranked values.

The formula for Spearman rank correlation coefficient is:

$r = 1 - 6\Sigma d^2 / n(n^2-1)$

where: r = Spearman rank correlation coefficient

d = the difference between the ranks of the paired observations

n = the number of observations

Kendall Correlation Coefficient: Kendall is another method for calculating the rank correlation coefficient, and is used when the variables are not normally distributed, or when the relationship between the variables is non-linear. It measures the degree of association between two variables by comparing the number of concordant and discordant pairs of observations.

The formula for Kendall correlation coefficient is:

$\tau = (c - d) / (0.5n(n-1))$

where: τ = Kendall correlation coefficient

c = the number of concordant pairs of observations

d = the number of discordant pairs of observations

n = the number of observations

Weighted correlation: Weighted correlation is used when the observations are not equally important, and assigns weights to each observation based on their importance. The weights are then used in the calculation of the correlation coefficient.

The formula for weighted correlation is:

$r = \Sigma(wxy) / \sqrt{(\Sigma(wx^2)\Sigma(wy^2))}$

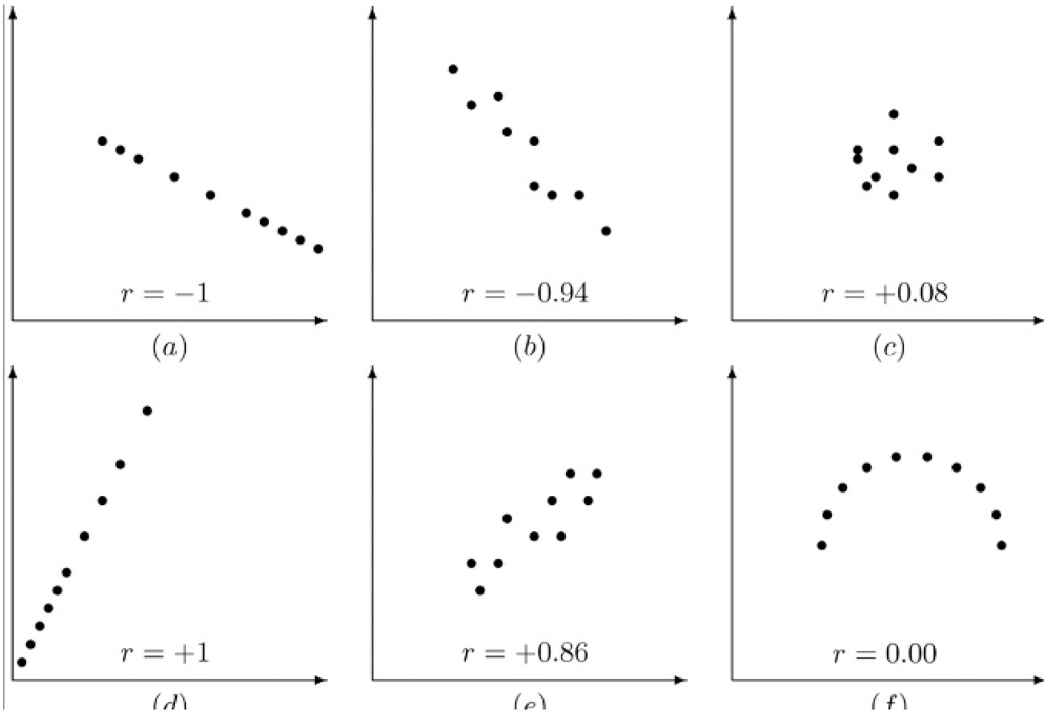
where: r = correlation coefficient

w = weight assigned to each observation

x = the values of the first variable

y = the values of the second variable

Diagram of different correlation coefficient



Reference:

Bodnar, T., Parolya, N., & Schmid, W. (2018). Robustness of alternative correlation measures under market stress: Evidence from European stock markets. Journal of Empirical Finance, 49, 62-80. https://doi.org/10.1016/j.jempfin.2018.07.002

Bodnar, T., Parolya, N., & Schmid, W. (2019). Dynamic conditional correlation models: A review and some developments. Econometrics, 7(3), 34.

Bollerslev, T., Engle, R. F., & Wooldridge, J. M. (1988). A capital asset pricing model with time-varying covariances. Journal of Political Economy, 96(1), 116-131.

Brown, S. J., & Warner, J. B. (1980). Measuring security price performance. Journal of Financial Economics, 8(3), 205-258. https://doi.org/10.1016/0304-405X(80)90002-1

Engle, R. F. (2002). Dynamic conditional correlation: A simple class of multivariate generalized autoregressive conditional heteroskedasticity models. Journal of Business & Economic Statistics, 20(3), 339-350.

Fabozzi, F. J., & Markowitz, H. M. (2011). The theory and practice of investment management: Asset allocation, valuation, portfolio construction, and strategies (2nd ed.). John Wiley & Sons.

Fama, E. F. (1965). The Behavior of Stock-Market Prices. The Journal of Business, 38(1), 34-105.

Giot, P., & Laurent, S. (2003). Market risk in commodity markets: A VaR approach. Energy Economics, 25(5), 435-457. https://doi.org/10.1016/S0140-9883(03)00022-4

Granger, C. W., & Newbold, P. (1974). Spurious regressions in econometrics. Journal of Econometrics, 2(2), 111-120. https://doi.org/10.1016/0304-4076(74)90034-7

Hull, J. C. (2017). Options, Futures, and Other Derivatives (10th Ed.). Pearson.

Jorion, P. (2007). Financial Risk Manager Handbook (5th Ed.). John Wiley & Sons.

Li, M., & Su, L. (2015). Dependence and risk analysis of China's regional housing markets. Habitat International, 48, 55-64. https://doi.org/10.1016/j.habitatint.2015.03.023

Patton, A. J. (2011). Copula-based models for financial time series. Handbook of Financial Time Series, 437-460.

Pindyck, R. S., & Rubinfeld, D. L. (1998). Econometric Models and Economic Forecasts (4th Ed.). McGraw-Hill.

Poon, S. H., & Granger, C. W. (2003). Forecasting volatility in financial markets: A review. Journal of Economic Literature, 41(2), 478-539.

Schwager, J. D. (1996). Market Wizards: Interviews with Top Traders. HarperCollins. "Types of Volatility" - Investopedia. (n.d.). Retrieved April 30, 2023, from https://www.investopedia.com/terms/t/typesofvolatility.asp

"Volatility: Types and Calculation." Corporate Finance Institute. (n.d.). Retrieved April 30, 2023, from https://corporatefinanceinstitute.com/resources/knowledge/trading-investing/volatility/

In [1]:
```
%pip install pandas-datareader
```

```
Requirement already satisfied: pandas-datareader in c:\users\p3002745\anaconda3\lib\site-packages (0.10.0)
Requirement already satisfied: pandas>=0.23 in c:\users\p3002745\anaconda3\lib\site-packages (from pandas-datarea
der) (1.3.4)
Requirement already satisfied: requests>=2.19.0 in c:\users\p3002745\anaconda3\lib\site-packages (from pandas-dat
areader) (2.26.0)
Requirement already satisfied: lxml in c:\users\p3002745\anaconda3\lib\site-packages (from pandas-datareader) (4.
9.2)
Requirement already satisfied: pytz>=2017.3 in c:\users\p3002745\anaconda3\lib\site-packages (from pandas>=0.23->
pandas-datareader) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\p3002745\anaconda3\lib\site-packages (from pand
as>=0.23->pandas-datareader) (2.8.2)
Requirement already satisfied: numpy>=1.17.3 in c:\users\p3002745\anaconda3\lib\site-packages (from pandas>=0.23-
>pandas-datareader) (1.20.3)
Requirement already satisfied: six>=1.5 in c:\users\p3002745\anaconda3\lib\site-packages (from python-dateutil>=2
.7.3->pandas>=0.23->pandas-datareader) (1.16.0)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\p3002745\anaconda3\lib\site-packages (from r
equests>=2.19.0->pandas-datareader) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\p3002745\anaconda3\lib\site-packages (from reque
sts>=2.19.0->pandas-datareader) (1.26.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\p3002745\anaconda3\lib\site-packages (from requests
>=2.19.0->pandas-datareader) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in c:\users\p3002745\anaconda3\lib\site-packages (from requests>=2.19
.0->pandas-datareader) (3.2)
Note: you may need to restart the kernel to use updated packages.
```

CORRELATION

A correlation is a statistical indicator of the relationship among two different variables. In a scatterplot, the fit of the data can be shown graphically. We can typically evaluate the relationship between the variables and decide whether or not they are related using scatterplot. In addition, the correlation is measured in integer from -1 to +1 which shows if there is positive correlation or negative correlation.

In [7]:

```python
#pandas and NumPy imports
import pandas as pd
from pandas import Series,DataFrame
import numpy as np

# For Visualization
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
%matplotlib inline

# For reading stock data from yahoo
import yfinance as yf
from pandas_datareader import data

# For time stamps
from datetime import datetime
#import datetime

# For division in Python 3
from __future__ import division
###################################
#!/usr/bin/python
import warnings
import pandas_datareader.data as pdr
warnings.simplefilter('ignore', FutureWarning)
#############################################
from functools import reduce
from tqdm import tqdm
```

In [8]:

```python
# The tech stocks we'll use for this analysis
tech_list = ['AAPL','GOOG','MSFT','AMZN']

# Set up End and Start times for data grab
end = datetime.now()
start = datetime(end.year - 1,end.month,end.day)


#For loop for grabing yahoo finance data and setting as a dataframe

for stock in tech_list:
    # Set DataFrame as the Stock Ticker  df=data.DataReader('AAPL','yahoo','2016/1/1','2017/1/1')
    globals()[stock] = yf.download(stock, start=start, end=end)
```

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

In [9]:

```python
# Summary Stats for Apple stocks
AAPL.describe()
```

Out[9]:

	Open	High	Low	Close	Adj Close	Volume
count	252.000000	252.000000	252.000000	252.000000	252.000000	2.520000e+02
mean	149.446032	151.462064	147.673135	149.696012	149.335922	7.850940e+07
std	10.839053	10.678778	11.048599	10.941798	10.958731	2.487611e+07
min	126.010002	127.769997	124.169998	125.019997	124.829399	3.031996e+07
25%	142.120003	143.987495	139.974998	142.472496	141.981434	6.292415e+07
50%	148.884995	150.930000	147.264999	149.375000	148.933762	7.370535e+07
75%	156.657501	158.277496	154.204994	156.979996	156.441681	8.804190e+07
max	173.750000	176.149994	173.119995	174.550003	173.995270	1.826020e+08

In [10]:

```python
# General Info about Apple Stock
AAPL.info()
```

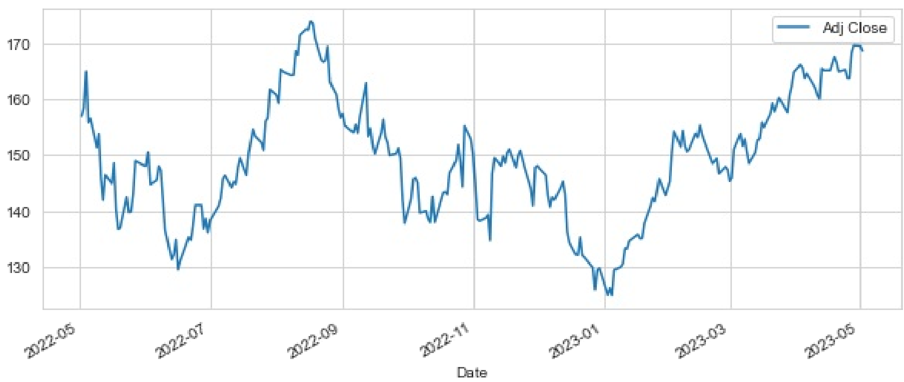
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 252 entries, 2022-05-02 to 2023-05-02
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Open       252 non-null    float64
 1   High       252 non-null    float64
 2   Low        252 non-null    float64
 3   Close      252 non-null    float64
 4   Adj Close  252 non-null    float64
 5   Volume     252 non-null    int64  
dtypes: float64(5), int64(1)
memory usage: 13.8 KB
```

In [11]:

```python
# Historical view of the closing price of Apple stock
AAPL['Adj Close'].plot(legend=True,figsize=(10,4))
```

Out[11]:

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

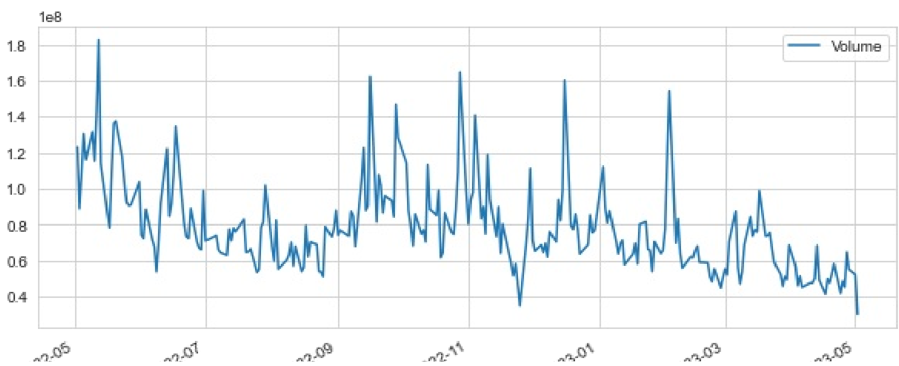


In [12]:

```python
# Historical view of the total volume of Apple stock traded each day
AAPL['Volume'].plot(legend=True,figsize=(10,4))
```

Out[12]:

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



In [13]:

```python
# Calculation to grab all the closing prices for the tech stock list into one DataFrame
closing_df = yf.download(['AAPL','GOOG','MSFT','AMZN'],start,end)['Adj Close']
```

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

In [14]:

```python
# Quick look of the data frame
closing_df.head()
```

Out[14]:

Date	AAPL	AMZN	GOOG	MSFT
2022-05-02	157.008896	124.500000	117.156998	281.706329
2022-05-03	158.519745	124.253502	118.129501	279.042511
2022-05-04	165.020370	125.928497	122.574997	287.162842
2022-05-05	155.826065	116.406998	116.746498	274.655548
2022-05-06	156.562683	114.772499	115.660004	272.061005

In [15]:

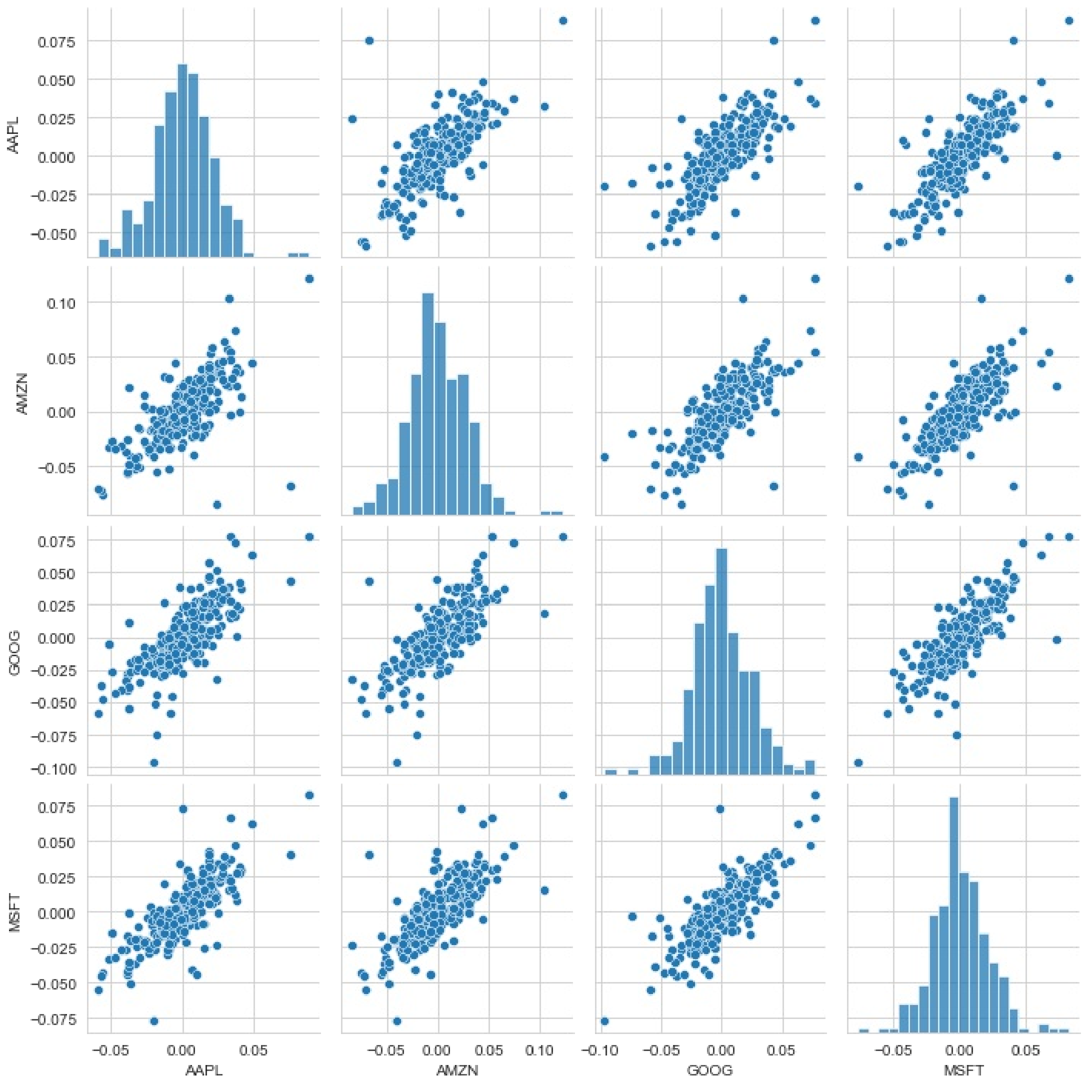
```python
# Calculate the daily return percent of all stocks and store them in a new tech returns DataFrame
tech_rets = closing_df.pct_change()
```

In [16]:

```python
# Correlation analysis for every possible combination of stocks in our technology stock ticker list.
sns.pairplot(tech_rets.dropna())
```

Out[16]:

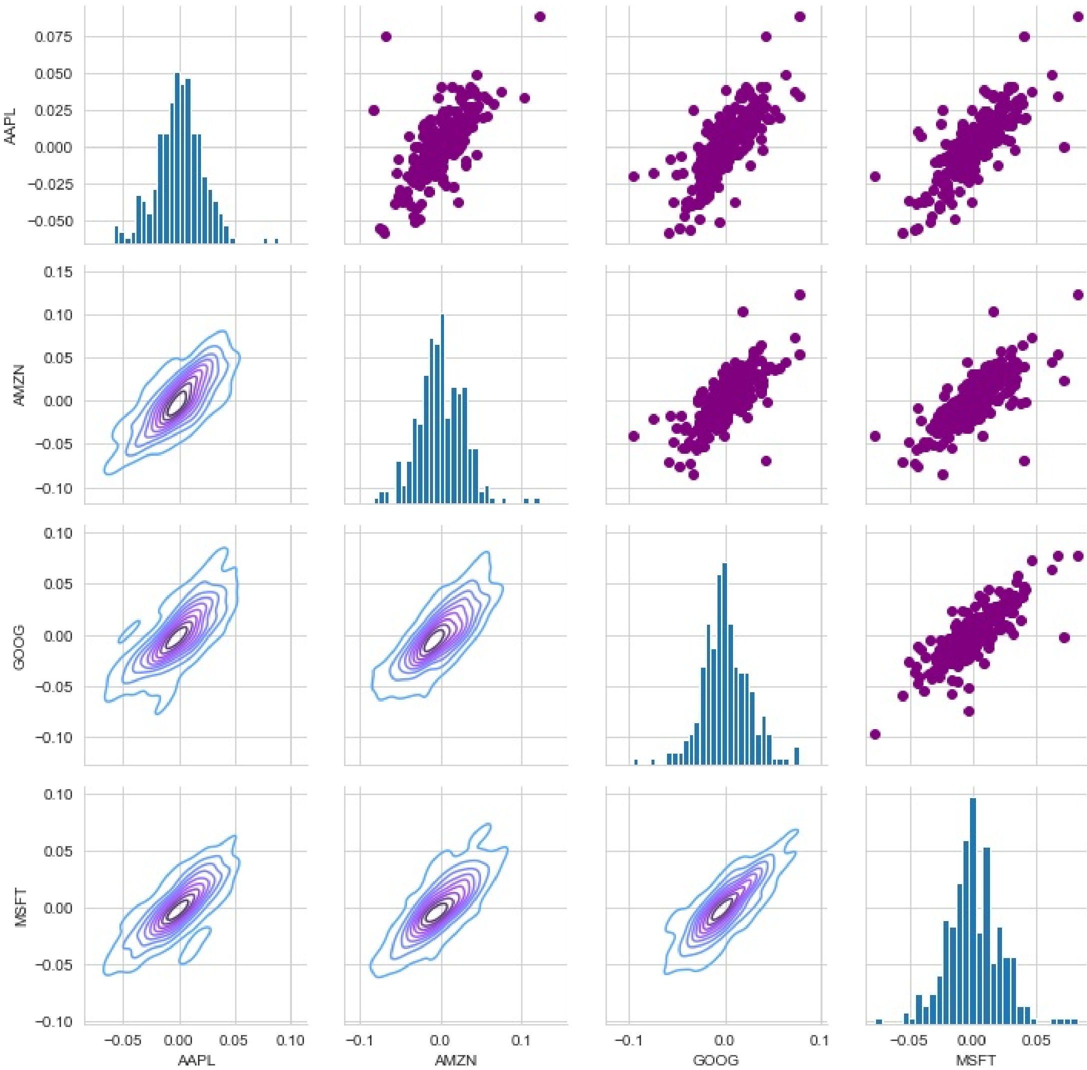
```
<seaborn.axisgrid.PairGrid at 0x2097d9c4fa0>
```



In [17]:

```
# Mixed plot to visualize the correlation between all technology stocks
returns_fig = sns.PairGrid(tech_rets.dropna())

returns_fig.map_upper(plt.scatter,color='purple')

returns_fig.map_lower(sns.kdeplot,cmap='cool_d')

returns_fig.map_diag(plt.hist,bins=30)
```

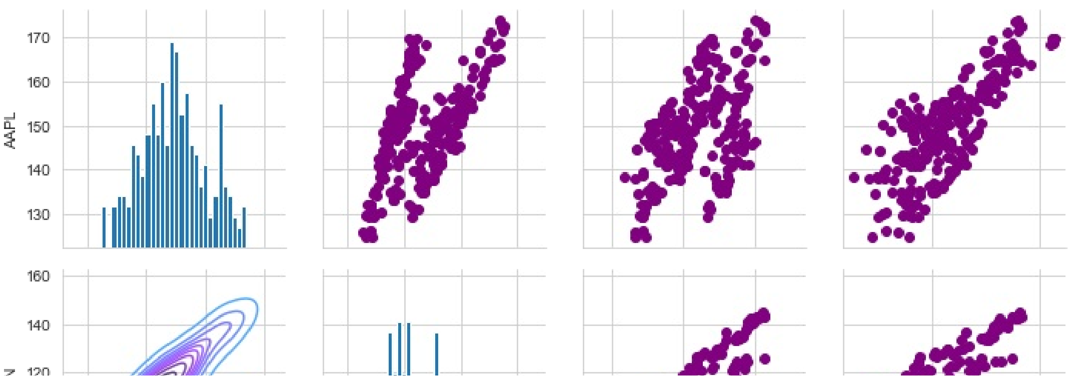
Out[17]: <seaborn.axisgrid.PairGrid at 0x2097e31f370>

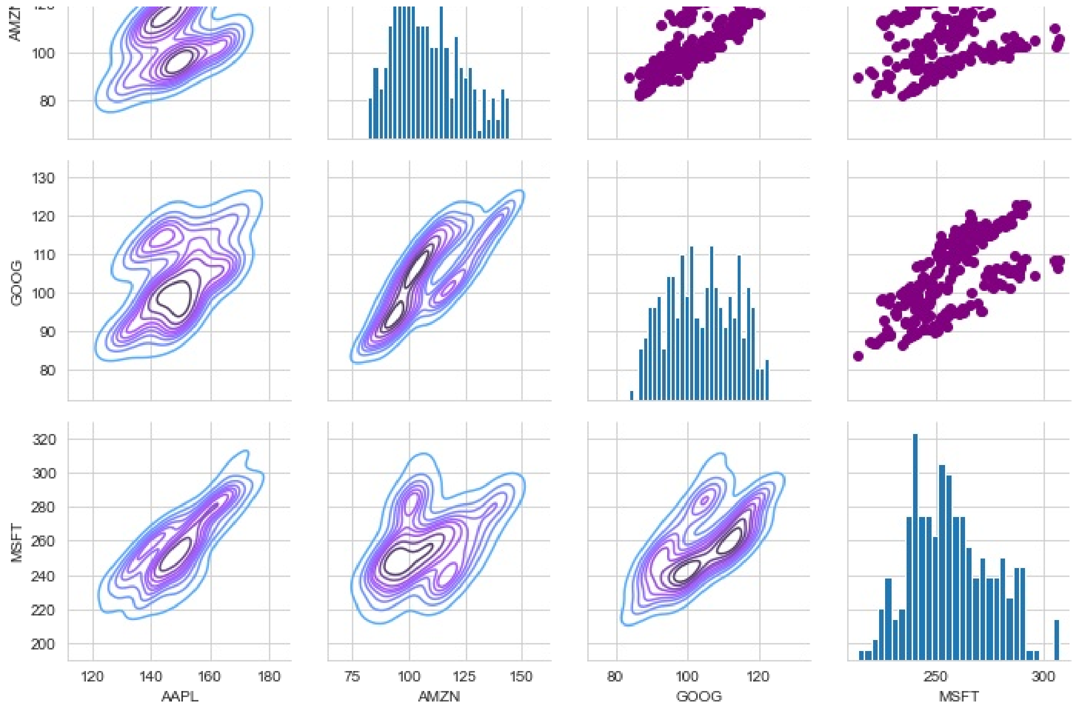


In [18]:

```
# Correlation analysis by using mixed types of plots for the closing price of all technology stocks
returns_fig = sns.PairGrid(closing_df)

returns_fig.map_upper(plt.scatter,color='purple')

returns_fig.map_lower(sns.kdeplot,cmap='cool_d')

returns_fig.map_diag(plt.hist,bins=30)
```

Out[18]: <seaborn.axisgrid.PairGrid at 0x2097e333760>





COEFFICIENT CORRELATION FORMULA

$$r_{xy} = \frac{\mathrm{cov}(x, y)}{\delta_x \delta_y}$$

$$\mathrm{cov}(x, y) = \Sigma(x_i - \bar{x})(y_i - \bar{y})$$

$$\delta_x = \sqrt{\Sigma(x_i - \bar{x})^{2}}$$

$$\delta_y = \sqrt{\Sigma(y_i - \bar{y})^{2}}$$

OR

$$r_{xy} = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\Sigma(x_i - \bar{x})^{2} \Sigma(y_i - \bar{y})^{2}}}$$

In [19]:

```
np.random.seed(123)

def process_date(df):
    df = df.reset_index().rename(columns={'Date': 'date'})
    df.date = df.date.dt.date

    return df

representation_symbol_mapping = {
    'bond': '^TNX',
    'stock': 'AAPL',
    'crypto': 'BTC-USD',
    'crypto_etf': 'BTF',
    'equity_etf': 'SPY',
}

START = '2022-01-01'
END = '2022-12-31'
```

In [20]:

```
def get_representative_data():
    dfs = []

    for asset_kind_str, symbol in representation_symbol_mapping.items():
        df = yf.download(symbol, start=START, end=END)

        if 'Adj Close' in df:
            df = df.rename(columns={'Adj Close': asset_kind_str})[[asset_kind_str]]
        else:
            df = df.rename(columns={'Close': asset_kind_str})[[asset_kind_str]]

        df = process_date(df)
        dfs.append(df)

    df = reduce(lambda left, right: pd.merge(left, right, on=['date'], how='inner'), dfs)
```

```python
    return df
```

In [21]:

```python
df = get_representative_data()

def cal_return(df):
    fields = [col for col in df.columns if col != 'date']
    for field in fields:
        df[field] = df[field] / df[field].shift(1) - 1

    df['bond'] *= -1
    df = df.dropna(subset=[col for col in df.columns if col != 'date'])

    return df
```

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

In [22]:

```python
df = cal_return(df)
```

In [23]:

```python
def get_statistics(df):
    fields = [col for col in df.columns if col != 'date']
    df_stats = df.agg(
        {
            field: ['mean', 'std', 'var', 'skew', 'kurt'] for field in fields
        }

    )

    return df_stats
```

In [24]:

```python
df_stats = get_statistics(df)
df_stats
```

Out[24]:

	bond	stock	crypto	crypto_etf	equity_etf
mean	-0.003858	-0.001074	-0.003278	-0.003096	-0.000709
std	0.027622	0.022471	0.040119	0.041634	0.015297
var	0.000763	0.000505	0.001610	0.001733	0.000234
skew	-0.041607	0.326067	-0.897183	-0.432430	0.049228
kurt	0.451780	1.079766	5.357589	3.260793	0.353271

In [25]:

```python
cov = df.cov()
cov
```

Out[25]:

	bond	stock	crypto	crypto_etf	equity_etf
bond	0.000763	0.000051	0.000064	0.000041	0.000049
stock	0.000051	0.000505	0.000447	0.000487	0.000304
crypto	0.000064	0.000447	0.001610	0.001506	0.000346
crypto_etf	0.000041	0.000487	0.001506	0.001733	0.000375
equity_etf	0.000049	0.000304	0.000346	0.000375	0.000234

In [26]:

```python
corr = df.corr()
corr
```

Out[26]:

	bond	stock	crypto	crypto_etf	equity_etf
bond	1.000000	0.082968	0.058033	0.035643	0.116676
stock	0.082968	1.000000	0.496039	0.520172	0.885435
crypto	0.058033	0.496039	1.000000	0.901861	0.563882
crypto_etf	0.035643	0.520172	0.901861	1.000000	0.588086
equity_etf	0.116676	0.885435	0.563882	0.588086	1.000000

Volatility

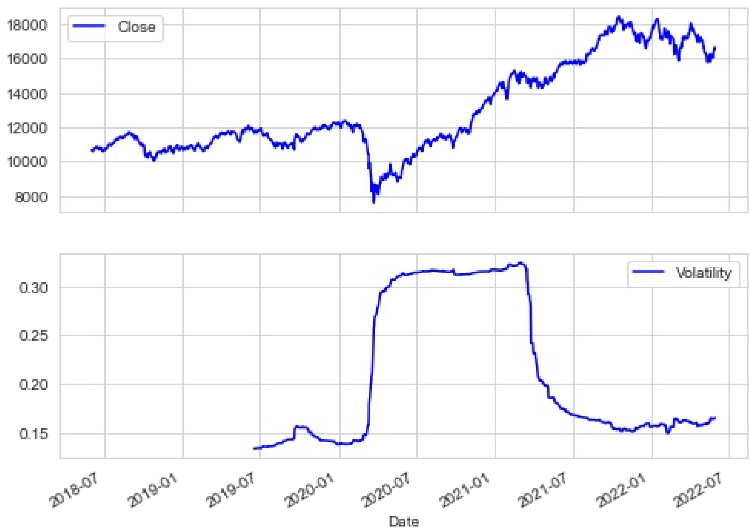
In [27]:

```python
## Computing Volatility
# Load the required modules and packages
import numpy as np
import pandas as pd
import yfinance as yf

# Pull NIFTY data from Yahoo finance
NIFTY = yf.download('^NSEI',start='2018-6-1', end='2022-6-1')

# Compute the logarithmic returns using the Closing price
NIFTY['Log_Ret'] = np.log(NIFTY['Close'] / NIFTY['Close'].shift(1))

# Compute Volatility using the pandas rolling standard deviation function
NIFTY['Volatility'] = NIFTY['Log_Ret'].rolling(window=252).std() * np.sqrt(252)
print(NIFTY.tail(15))

# Plot the NIFTY Price series and the Volatility
NIFTY[['Close', 'Volatility']].plot(subplots=True, color='blue',figsize=(8, 6))
```

```
[*********************100%***********************]  1 of 1 completed
                    Open          High           Low         Close  \
Date
2022-05-11  16270.049805  16318.750000  15992.599609  16167.099609
2022-05-12  16021.099609  16041.950195  15735.750000  15808.000000
2022-05-13  15977.000000  16083.599609  15740.849609  15782.150391
2022-05-16  15845.099609  15977.950195  15739.650391  15842.299805
2022-05-17  15912.599609  16284.250000  15900.799805  16259.299805
2022-05-18  16318.150391  16399.800781  16211.200195  16240.299805
2022-05-19  15917.400391  15984.750000  15775.200195  15809.400391
2022-05-20  16043.799805  16283.049805  16003.849609  16266.150391
2022-05-23  16290.950195  16414.699219  16185.750000  16214.700195
2022-05-24  16225.549805  16262.799805  16078.599609  16125.150391
2022-05-25  16196.349609  16223.349609  16006.950195  16025.799805
2022-05-26  16105.000000  16204.450195  15903.700195  16170.150391
2022-05-27  16296.599609  16370.599609  16221.950195  16352.450195
2022-05-30  16527.900391  16695.500000  16506.150391  16661.400391
2022-05-31  16578.449219  16690.750000  16521.900391  16584.550781

               Adj Close  Volume   Log_Ret  Volatility
Date
2022-05-11  16167.099609  284300 -0.004502    0.158307
2022-05-12  15808.000000  314900 -0.022462    0.159796
2022-05-13  15782.150391  369100 -0.001637    0.159679
2022-05-16  15842.299805  217600  0.003804    0.159529
2022-05-17  16259.299805  295700  0.025981    0.161462
2022-05-18  16240.299805  290400 -0.001169    0.161105
2022-05-19  15809.400391  313900 -0.026891    0.163393
2022-05-20  16266.150391  252400  0.028482    0.165009
2022-05-23  16214.700195  293800 -0.003168    0.164607
2022-05-24  16125.150391  249800 -0.005538    0.164620
2022-05-25  16025.799805  243300 -0.006180    0.164523
2022-05-26  16170.150391  314300  0.008967    0.163803
2022-05-27  16352.450195  274100  0.011211    0.164165
2022-05-30  16661.400391  251400  0.018717    0.165196
2022-05-31  16584.550781  651600 -0.004623    0.165170
```

Out[27]:

```
array([<AxesSubplot:xlabel='Date'>, <AxesSubplot:xlabel='Date'>],
      dtype=object)
```



In [28]:
```
# Sharpe Ratio function
def sharpe(returns, daily_risk_free_rate, days=252):
  volatility = returns.std()
  sharpe_ratio = (returns.mean() - daily_risk_free_rate) / volatility * np.sqrt(days)
  return sharpe_ratio

# Sortino Ratio function
def sortino(returns, daily_risk_free_rate, days=252):
  volatility = returns.std()
  sortino_ratio = (expected_returns - daily_risk_free_rate) / volatility * np.sqrt(days)
  return sortino_ratio
```

In [29]:
```
sharpe(NIFTY['Log_Ret'], 0)
```

Out[29]:
```
0.536108951256729
```

In [31]:
```
import datetime
```

In [32]:
```
# fetch multiple asset data
def getMultiAssetData(ticketList, date_from, date_to):
    def getData(ticker):
        data = yf.download(ticker, date_from, date_to)
        return data
    datas = map(getData, tickerList)
    return pd.concat(datas, keys=tickerList, names=['Ticker', 'Date'])

date_from = datetime.date(2018, 1, 1)
date_to = datetime.date(2020, 8, 31)
tickerList = ['AAPL', 'AMZN', 'JWN', 'PG']
multiData = getMultiAssetData(tickerList, date_from, date_to)
df = multiData.copy()
```

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

In [33]:
```
df = df.loc['AAPL', :]
df.tail()
```

Out[33]:

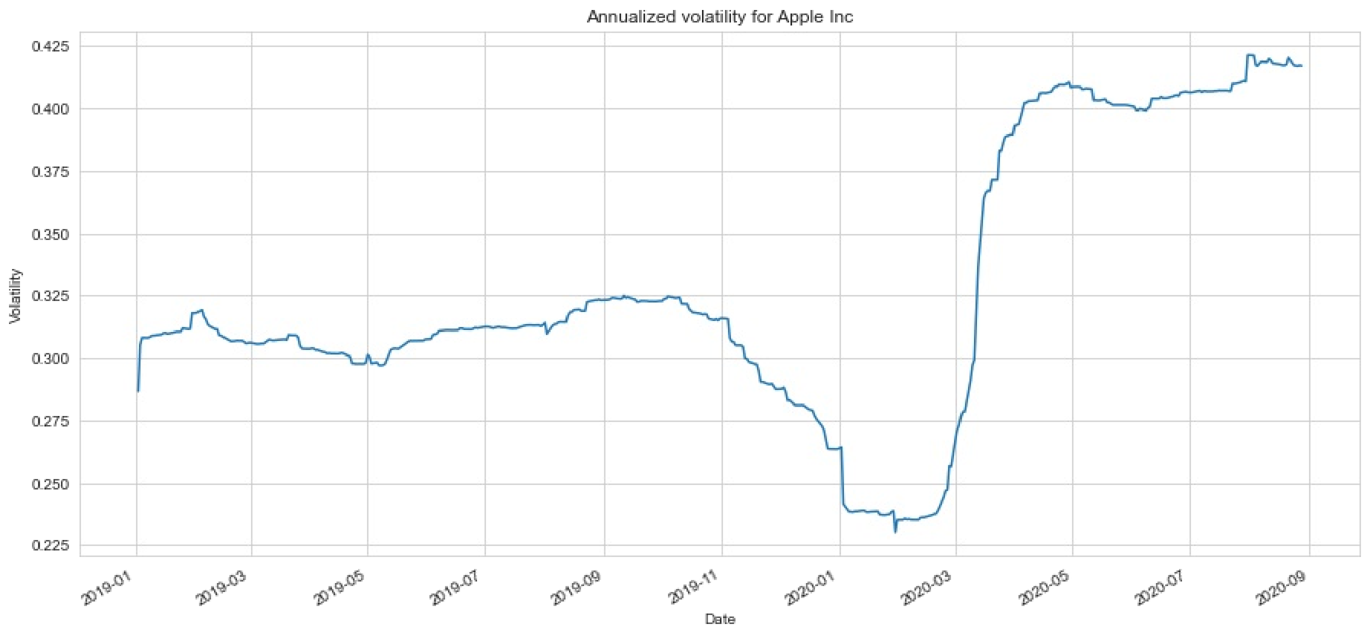
	Open	High	Low	Close	Adj Close	Volume
Date						
2020-08-24	128.697495	128.785004	123.937500	125.857498	123.961121	345937600
2020-08-25	124.697502	125.180000	123.052498	124.824997	122.944183	211495600
2020-08-26	126.180000	126.992500	125.082497	126.522499	124.616112	163022400
2020-08-27	127.142502	127.485001	123.832497	125.010002	123.126411	155552400
2020-08-28	126.012497	126.442497	124.577499	124.807503	122.926956	187630000

In [34]:
```
# compute volatility using Pandas rolling and std methods, the trading days is set to 252 days
TRADING_DAYS = 252
returns = np.log(df['Close']/df['Close'].shift(1))
returns.fillna(0, inplace=True)
volatility = returns.rolling(window=TRADING_DAYS).std()*np.sqrt(TRADING_DAYS)
volatility.tail()
```

Out[34]:
```
Date
2020-08-24    0.417332
2020-08-25    0.417219
2020-08-26    0.417071
2020-08-27    0.417351
2020-08-28    0.417170
Name: Close, dtype: float64
```

In [35]:
```
%matplotlib inline
```

```
fig = plt.figure(figsize=(15, 7))
ax1 = fig.add_subplot(1, 1, 1)
volatility.plot(ax=ax1)
ax1.set_xlabel('Date')
ax1.set_ylabel('Volatility')
ax1.set_title('Annualized volatility for Apple Inc')
plt.show()
```



In [36]:

```
# use pivot to reshape DataFrame with only Close
df = multiData.copy()
closePrice = df[['Close']]
closePrice = closePrice.reset_index()
closePriceTable = closePrice.pivot(index='Date', columns='Ticker', values='Close')
closePriceTable.tail()
```

Out[36]:

Ticker	AAPL	AMZN	JWN	PG
Date				
2020-08-24	125.857498	165.373001	15.57	138.509995
2020-08-25	124.824997	167.324493	15.54	139.059998
2020-08-26	126.522499	172.092499	14.69	138.389999
2020-08-27	125.010002	170.000000	14.79	138.210007
2020-08-28	124.807503	170.089996	15.68	138.770004

In [37]:

```
# compute volatility using Pandas rolling and std methods, the trading days is set to 252 days
TRADING_DAYS = 252
returns_portfolio = np.log(closePriceTable/closePriceTable.shift(1))
returns_portfolio.fillna(0, inplace=True)
volatility_portfolio = returns_portfolio.rolling(window=TRADING_DAYS).std()*np.sqrt(TRADING_DAYS)
volatility_portfolio.tail()
```

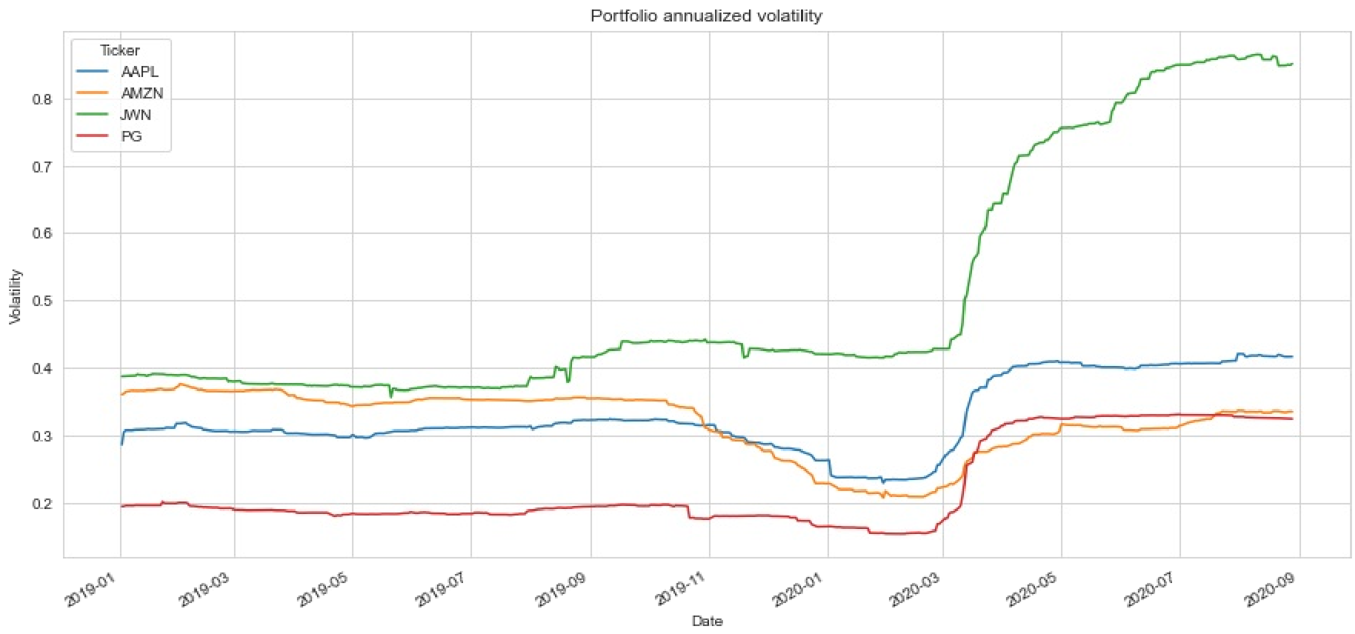
Out[37]:

Ticker	AAPL	AMZN	JWN	PG
Date				
2020-08-24	0.417332	0.334757	0.847830	0.325688
2020-08-25	0.417219	0.334777	0.847742	0.325297
2020-08-26	0.417071	0.335686	0.848871	0.325198
2020-08-27	0.417351	0.336014	0.848346	0.325138
2020-08-28	0.417170	0.335874	0.850365	0.325148

In [38]:

```
%matplotlib inline
fig = plt.figure(figsize=(15, 7))
ax2 = fig.add_subplot(1, 1, 1)
volatility_portfolio.plot(ax=ax2)
ax2.set_xlabel('Date')
ax2.set_ylabel('Volatility')
```

```
ax2.set_title('Portfolio annualized volatility')
plt.show()
```



In [39]:

```
df = multiData.copy()
df = df.loc['AAPL', :]
df.tail()
```

Out[39]:

Date	Open	High	Low	Close	Adj Close	Volume
2020-08-24	128.697495	128.785004	123.937500	125.857498	123.961121	345937600
2020-08-25	124.697502	125.180000	123.052498	124.824997	122.944183	211495600
2020-08-26	126.180000	126.992500	125.082497	126.522499	124.616112	163022400
2020-08-27	127.142502	127.485001	123.832497	125.010002	123.126411	155552400
2020-08-28	126.012497	126.442497	124.577499	124.807503	122.926956	187630000

In [40]:

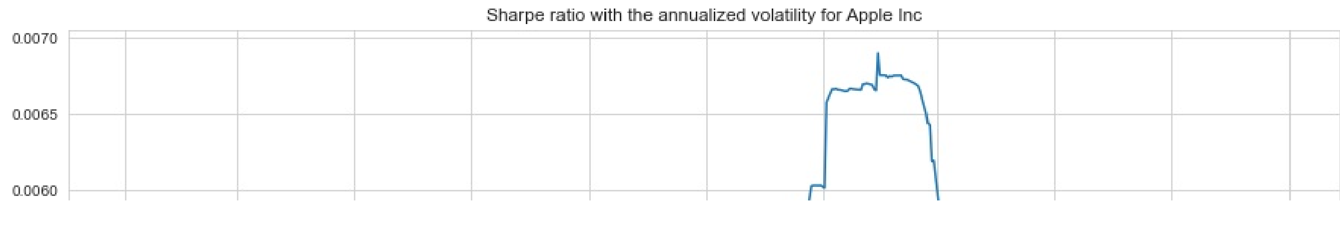
```
# compute sharpe ratio using Pandas rolling and std methods, the trading days is set to 252 days
TRADING_DAYS = 252
returns = np.log(df['Close']/df['Close'].shift(1))
returns.fillna(0, inplace=True)
volatility = returns.rolling(window=TRADING_DAYS).std()*np.sqrt(TRADING_DAYS)
sharpe_ratio = returns.mean()/volatility
sharpe_ratio.tail()
```

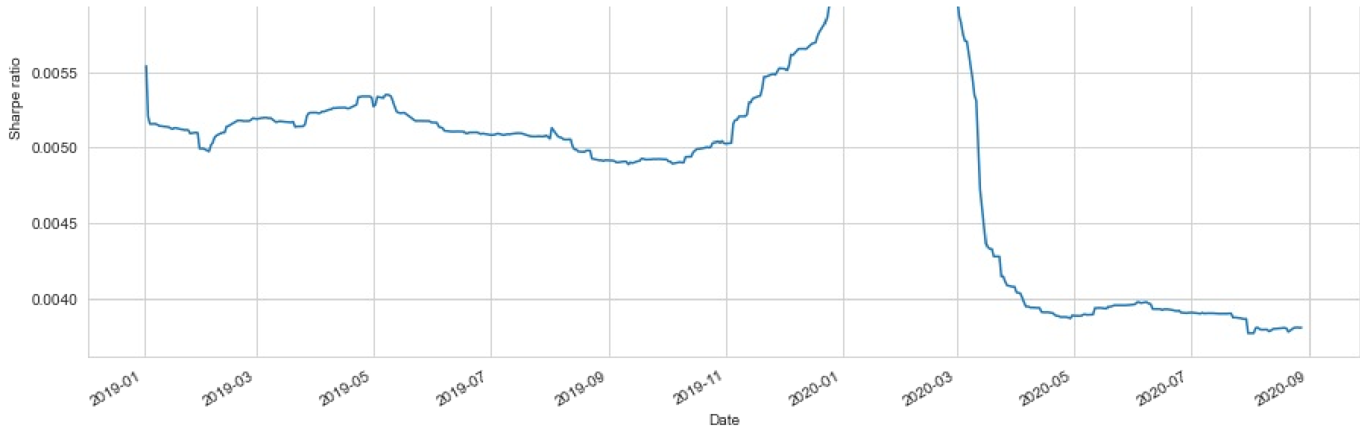
Out[40]:

```
Date
2020-08-24    0.003805
2020-08-25    0.003807
2020-08-26    0.003808
2020-08-27    0.003805
2020-08-28    0.003807
Name: Close, dtype: float64
```

In [41]:

```
%matplotlib inline
fig = plt.figure(figsize=(15, 7))
ax3 = fig.add_subplot(1, 1, 1)
sharpe_ratio.plot(ax=ax3)
ax3.set_xlabel('Date')
ax3.set_ylabel('Sharpe ratio')
ax3.set_title('Sharpe ratio with the annualized volatility for Apple Inc')
plt.show()
```





In [42]:
```python
# use pivot to reshape DataFrame with only Close
df = multiData.copy()
closePrice = df[['Close']]
closePrice = closePrice.reset_index()
closePriceTable = closePrice.pivot(index='Date', columns='Ticker', values='Close')
closePriceTable.tail()
```

Out[42]:

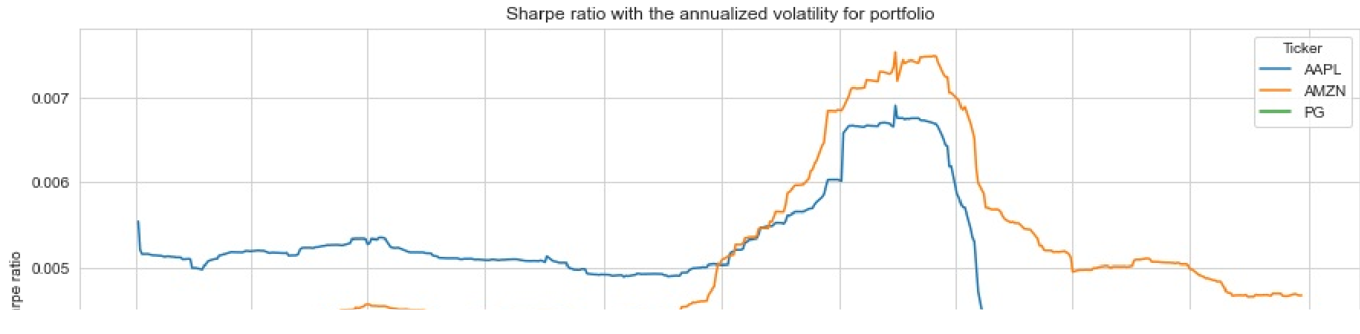
Ticker	AAPL	AMZN	JWN	PG
Date				
2020-08-24	125.857498	165.373001	15.57	138.509995
2020-08-25	124.824997	167.324493	15.54	139.059998
2020-08-26	126.522499	172.092499	14.69	138.389999
2020-08-27	125.010002	170.000000	14.79	138.210007
2020-08-28	124.807503	170.089996	15.68	138.770004

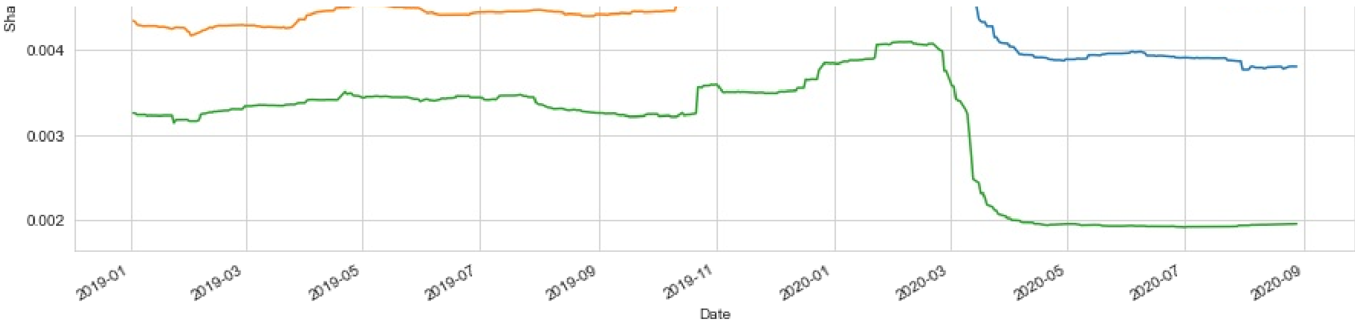
In [43]:
```python
# compute sharpe ratio using Pandas rolling and std methods, the trading days is set to 252 days
TRADING_DAYS = 252
returns_portfolio = np.log(closePriceTable/closePriceTable.shift(1))
returns_portfolio.fillna(0, inplace=True)
volatility_portfolio = returns_portfolio.rolling(window=TRADING_DAYS).std()*np.sqrt(TRADING_DAYS)
sharpe_ratio_portfolio = returns_portfolio.mean()/volatility_portfolio
sharpe_ratio_portfolio.tail()
```

Out[43]:

Ticker	AAPL	AMZN	JWN	PG
Date				
2020-08-24	0.003805	0.004687	-0.002010	0.001951
2020-08-25	0.003807	0.004686	-0.002010	0.001954
2020-08-26	0.003808	0.004674	-0.002008	0.001954
2020-08-27	0.003805	0.004669	-0.002009	0.001955
2020-08-28	0.003807	0.004671	-0.002004	0.001955

In [44]:
```python
%matplotlib inline
fig = plt.figure(figsize=(15, 7))
ax4 = fig.add_subplot(1, 1, 1)
sharpe_ratio_portfolio[sharpe_ratio_portfolio.columns[sharpe_ratio_portfolio.columns != 'JWN']].plot(ax=ax4)
ax4.set_xlabel('Date')
ax4.set_ylabel('Sharpe ratio')
ax4.set_title('Sharpe ratio with the annualized volatility for portfolio')
plt.show()
```





In [ ]:

Loading [MathJax]/extensions/Safe.js