notebook

March 15, 2023

1 Mihir Damania (NUID)=002658120

1.1 Quantitative Portfolio Management

1.1.1 Homework 2

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

[2]:
```
#pip install arch
```

[3]:
```
Stock_Data=pd.read_csv('Stock_data.csv')
```

[4]:
```
JPM=Stock_Data['JPM Adj Close']
MS=Stock_Data['MS Adj Close']
Citi=Stock_Data['Citi Adj Close']
Index=Stock_Data['SP500 Adj Close']
Risk_Free=Stock_Data['RF']
```

1.1.2 Problem A1

Mean Returns

[5]:
```
JPM_Mean_Return=JPM.pct_change().mean()
print('JPM Mean Return:',JPM_Mean_Return)
MS_Mean_Return=MS.pct_change().mean()
print('MS Mean Return:',MS_Mean_Return)
Citi_Mean_Return=Citi.pct_change().mean()
print('Citi Mean Return:',Citi_Mean_Return)
Index_Mean_Return=Index.pct_change().mean()
print('Index Mean Return:',Index_Mean_Return)
```

```
JPM Mean Return: 0.00038918917145939927
MS Mean Return: 0.0005666362892246077
Citi Mean Return: -8.975095060469179e-05
Index Mean Return: 0.0004208891882589548
```

Mean return is the average of daily return for the 252 days. We can see that Morgan Stanley has provided the highest return as compared to other stocks and market. Citi, on the other hand, has a negative mean return of -0.008975%, which indicates that, on average, it has lost value over the given time period. It indicates that JP Morgan and Citi Bank has underpreformed than the broader Market.

Mean

[6]:
```
JPM_Mean=JPM.mean()
print('JPM Mean:',JPM_Mean)
MS_Mean=MS.mean()
print('MS Mean:',MS_Mean)
Citi_Mean=Citi.mean()
print('Citi Mean:',Citi_Mean)
Index_Mean=Index.mean()
print('Index Mean:',Index_Mean)
```

```
JPM Mean: 124.31755511200001
MS Mean: 84.94710106
Citi Mean: 48.061669888000004
Index Mean: 4011.92332
```

JPM has a mean value of 124.318, which indicates that the average price of JPM stock over the given time period is 124.318 dollars per share. MS has a lower mean value of 84.947, indicating that the average price of MS stock is lower than that of JPM over the same time period. Citi has a lower mean value of 48.062, indicating that the average price of Citi stock is lower than that of both JPM and MS over the given time period.Whereas, the Indes Mean is 4011.92

Median

[7]:
```
JPM_Median=np.median(JPM)
print('JPM_Median:',JPM_Median)
MS_Median=np.median(MS)
print('MS_Median:',MS_Median)
Citi_Median=np.median(Citi)
print('Citi_Median:',Citi_Median)
Index_Median=np.median(Index)
print('Index_Median:',Index_Median)
```

```
JPM_Median: 124.893749
MS_Median: 84.4290125
Citi_Median: 48.334276
Index_Median: 3982.705
```

In all the stocks, we can see that Mean and Median are slightly different which indicates that the distribution can be slightly more distributed on left or right. This indicates that the overall distribution of data is not perfectly symmetrical or normal.

Variance

[8]:
```
JPM_Variance=np.var(JPM)
print('JPM_Variance:',JPM_Variance)
MS_Variance=np.var(MS)
print('MS_Variance:',MS_Variance)
Citi_Variance=np.var(Citi)
print('Citi_Variance:',Citi_Variance)
Index_Variance=np.var(Index)
print('Index_Variance:',Index_Variance)
```

```
JPM_Variance: 126.63388005743808
MS_Variance: 49.70280034845622
Citi_Variance: 11.011318983841756
Index_Variance: 50038.03965417759
```

Variance means dispersion of data. We can clearly see that Variance of JPM has large spread out from the mean as compared to Morgan Stanley.Citi has an even lower variance of 11.011, indicating that the prices of Citi stock are less spread out from the mean. Finally, the Index has the large spread out from the mean value. Based on this, Citi Bank to have the lowest variance of the four options, suggesting that it may be a relatively lower risk and more stable investment compared to JPMorgan (JPM), Morgan Stanley (MS), and the Index.

Standard Deviation

[9]:
```
JPM_Standard_Deviation=JPM.std()
print('JPM_Standard_Deviation:',JPM_Standard_Deviation)
MS_Standard_Deviation=MS.std()
print('MS_Standard_Deviation:',MS_Standard_Deviation)
Citi_Standard_Deviation=Citi.std()
print('Citi_Standard_Deviation:',Citi_Standard_Deviation)
Index_Standard_Deviation=Index.std()
print('Index_Standard_Deviation:',Index_Standard_Deviation)
```

```
JPM_Standard_Deviation: 11.275746088701428
MS_Standard_Deviation: 7.064163785502843
Citi_Standard_Deviation: 3.324987390718271
Index_Standard_Deviation: 224.14057115284703
```

Standard deviations are used to find out the risk of an asset. The more volatile the company is the more risky the investment is likely to be. As we can see that the standard deviation of the JPM & Index is more, it more likely that it will give more return. And for the investor who are risk averse, they can simply select between Morgan Stanley or Citi Bank because of less mobility.

Skewness

[10]:
```
JPM_Skewness=pd.Series(JPM).skew()
print('JPM_Skewness:',JPM_Skewness)
MS_Skewness=pd.Series(MS).skew()
print('MS_Skewness:',MS_Skewness)
Citi_Skewness=pd.Series(Citi).skew()
print('Citi_Skewness:',Citi_Skewness)
```

```
Index_Skewness=pd.Series(Index).skew()
print('Index_Skewness:',Index_Skewness)
```

```
JPM_Skewness: 0.009703781667858987
MS_Skewness: 0.3266215154605979
Citi_Skewness: -0.21325157455442106
Index_Skewness: 0.645075429481999
```

As the above results are between -0.5 and +0.5 indicate that the both the skewness are approximately symmetric. The skewness of JPM is 0.0097, of MS is 0.3266 whereas, the Skewness of Citi Bank is -0.213. This indicates that the average return may be relatively low, but there is a higher chance of having large gains in the future. On the other hand, Index is moderately skew.

Kurtosis

[11]:
```
JPM_Kurtosis=pd.Series(JPM).kurtosis()
print('JPM Kurtosis:',JPM_Kurtosis)
MS_Kurtosis=pd.Series(MS).kurtosis()
print('MS Kurtosis:',MS_Kurtosis)
Citi_Kurtosis=pd.Series(Citi).kurtosis()
print('Citi Kurtosis:',Citi_Kurtosis)
Index_Kurtosis=pd.Series(Index).kurtosis()
print('Index Kurtosis:',Index_Kurtosis)
```

```
JPM Kurtosis: -1.247022298298029
MS Kurtosis: -0.5150815787032275
Citi Kurtosis: -0.6336896155529232
Index Kurtosis: 0.19912235325458383
```

Based on Kurtosis Value,JPM has a negative kurtosis value of -1.247, indicating that its distribution is platykurtic, i.e., it has fewer extreme values and lighter tails than a normal distribution. MS also has a negative kurtosis value of -0.515, indicating that its distribution is also platykurtic. Citi has a negative kurtosis value of -0.634, which is similar to MS and also indicates a platykurtic distribution. The Index has a positive kurtosis value of 0.199, indicating that its distribution has more extreme values and heavier tails than a normal distribution.

Coefficient of Variation

[12]:
```
JPM_Corr_Var=JPM_Standard_Deviation/(JPM.mean())
print('JPM Coefficient of Variation:',JPM_Corr_Var)
MS_Corr_Var=MS_Standard_Deviation/(MS.mean())
print('MS Coefficient of Variation:',MS_Corr_Var)
Citi_Corr_Var=Citi_Standard_Deviation/(Citi.mean())
print('Citi Coefficient of Variation:',Citi_Corr_Var)
Index_Corr_Var=Index_Standard_Deviation/(Index.mean())
print('Index Coefficient of Variation:',Index_Corr_Var)
```

```
JPM Coefficient of Variation: 0.09070115703725751
MS Coefficient of Variation: 0.08315956280265845
```

```
Citi Coefficient of Variation: 0.06918168674676972
Index Coefficient of Variation: 0.05586860796552987
```

CV provides a measure of the relative variability of the dataset, taking into account the differences in the mean of the datasets. JPM has a CV of 0.0907, indicating that its returns have a relatively high variability compared to its mean. MS has a CV of 0.0832, which is slightly lower than JPM's CV, but still suggests a relatively high variability in returns. Citi has a CV of 0.0692, indicating that its returns have a lower variability compared to its mean than the other two banks. The Index has the lowest CV of 0.0559, indicating that its returns have the lowest variability compared to its mean among the four options.

(b)

Secruity Market Line

[13]:
```
JPM_Excess_Return=(JPM.pct_change()-Risk_Free).dropna()
MS_Excess_Return=(MS.pct_change()-Risk_Free).dropna()
Citi_Excess_Return=(Citi.pct_change()-Risk_Free).dropna()
Index_Excess_Return=(Index.pct_change()-Risk_Free).dropna()
```

JP Morgan Beta

[14]:
```
X=Index_Excess_Return.values.reshape(-1,1)
Y=JPM_Excess_Return
reg=LinearRegression().fit(X,Y)
JPM_Beta=reg.coef_[0]
print('JPM Beta:',JPM_Beta)
COV=np.cov(Index,JPM)
```

```
JPM Beta: 0.15901108448342416
```

The beta of 0.159 indicates that the JPM has low volatility and is less sensitive to changes in the market compared to the market index, which has a beta of 1

Morgan Stanley Beta

[15]:
```
X=Index_Excess_Return.values.reshape(-1,1)
Y=MS_Excess_Return
reg=LinearRegression().fit(X,Y)
MS_Beta=reg.coef_[0]
print('MS Beta:',MS_Beta)
```

```
MS Beta: 0.25240702501552653
```

The beta of 0.252 indicates that the MS has low volatility and is less sensitive to changes in the market compared to the market index, which has a beta of 1

CitiGroup Beta

[16]:
```
X=Index_Excess_Return.values.reshape(-1,1)
Y=Citi_Excess_Return
```

```
reg=LinearRegression().fit(X,Y)
Citi_Beta=reg.coef_[0]
print('Citi Beta:',Citi_Beta)
```

```
Citi Beta: 0.18563889977244513
```

The beta of 0.1856 indicates that the Citi Group has low volatility and is less sensitive to changes in the market compared to the market index, which has a beta of 1

Capital Asset Pricing Model

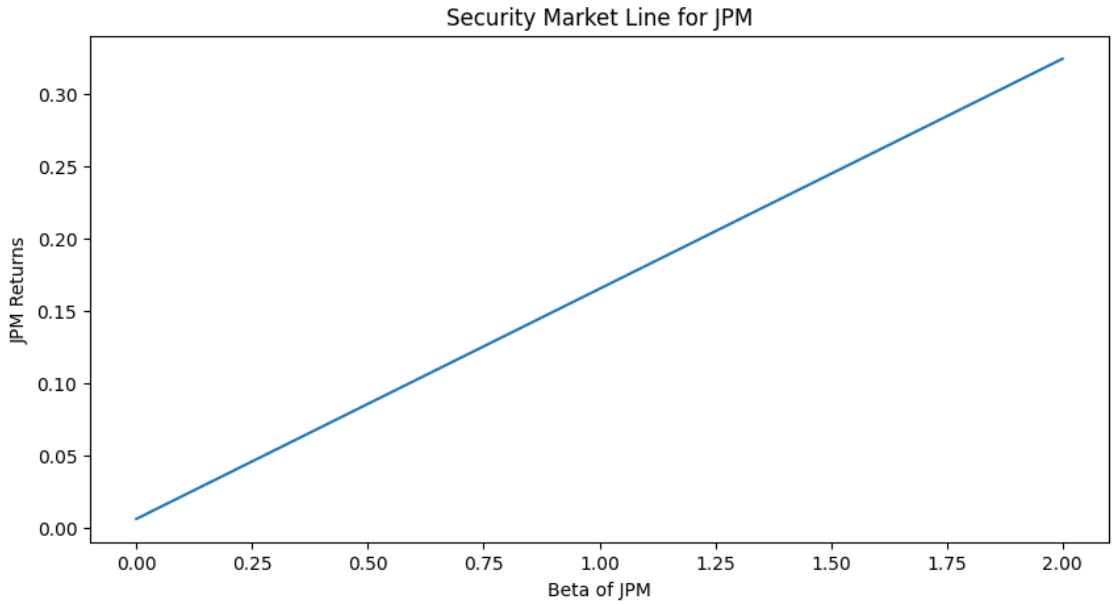
[17]:
```
JPM_CAPM=Risk_Free+JPM_Beta*Index_Excess_Return
print('CAPM of JPM:',JPM_CAPM.mean())
MS_CAPM=Risk_Free+MS_Beta*Index_Excess_Return
print('CAPM of MS:',MS_CAPM.mean())
Citi_CAPM=Risk_Free+Citi_Beta*Index_Excess_Return
print('CAPM of Citi:',Citi_CAPM.mean())
```

```
CAPM of JPM: 0.00600451043775088
CAPM of MS: 0.005384421934145563
CAPM of Citi: 0.005827719005031463
```

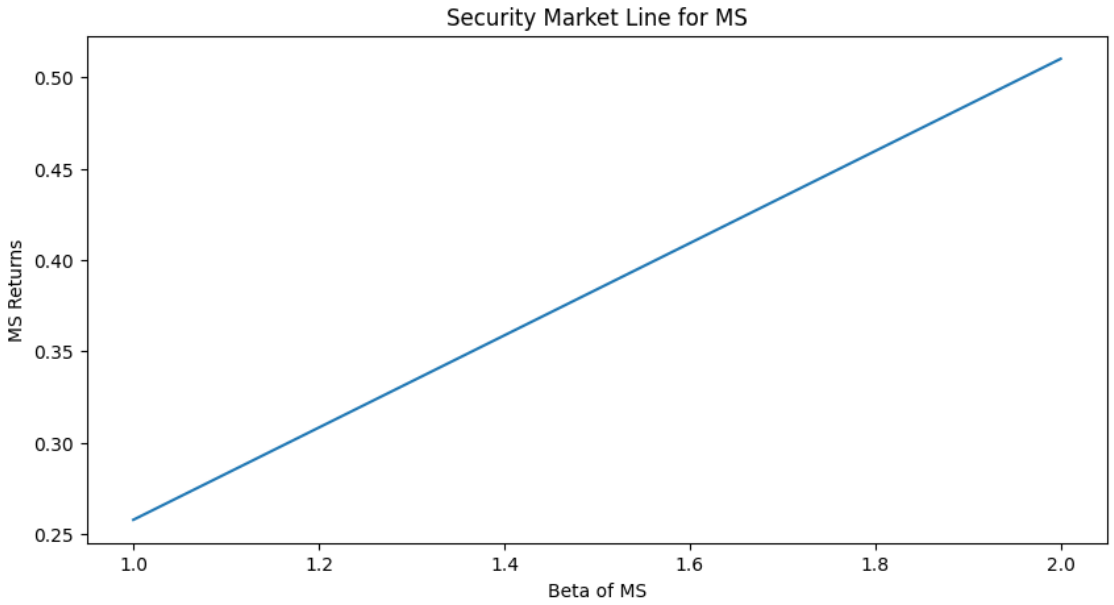
JPM's stock is 0.6% higher than the risk-free rate, given its level of systematic risk as represented by its beta. The MS's stock is 0.54% higher than the risk-free rate, given its level of systematic risk as represented by its beta and Citi's stock is 0.58% higher than the risk-free rate, given its level of systematic risk as represented by its beta. Based on the CAPM values, JPM appears to have the highest expected return, followed by Citi and then MS.

Security Market Line

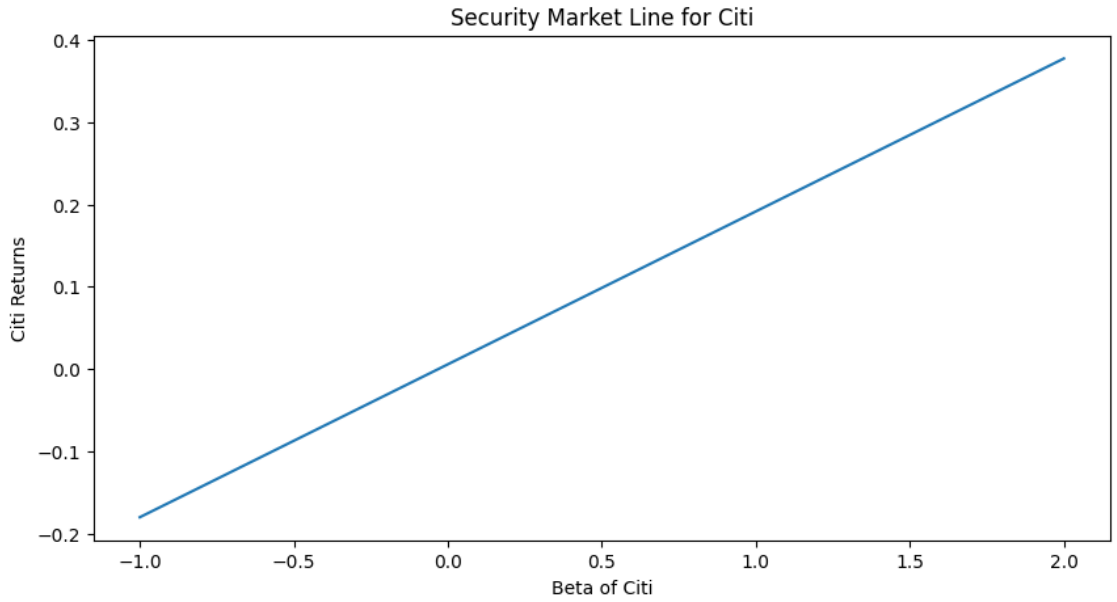
[18]:
```
# Creating SML for JPM
x = np.linspace(0, 2, 100)
y = JPM_Beta*x + JPM_CAPM.mean()

fig = plt.figure(figsize = (10, 5))
# Create the plot
plt.plot(x, y)

# Show the plot
plt.xlabel('Beta of JPM')
plt.ylabel('JPM Returns')
plt.title('Security Market Line for JPM')
plt.show()
```



[19]:
```
# Creating SML for MS
x = np.linspace(1, 2, 100)
y = MS_Beta*x + MS_CAPM.mean()

fig = plt.figure(figsize = (10, 5))
# Create the plot
plt.plot(x, y)

# Show the plot
plt.xlabel('Beta of MS')
plt.ylabel('MS Returns')
plt.title('Security Market Line for MS')
plt.show()
```



[20]:
```
# Creating SML for MS
x = np.linspace(-1, 2, 100)
y = Citi_Beta*x + Citi_CAPM.mean()

fig = plt.figure(figsize = (10, 5))
# Create the plot
plt.plot(x, y)

# Show the plot
plt.xlabel('Beta of Citi')
plt.ylabel('Citi Returns')
plt.title('Security Market Line for Citi')
plt.show()
```



1.1.3 Problem A2

[21]:
```
Farma_French=pd.read_csv('FarmaFrench.CSV')
Industry_Portfolio=pd.read_csv('5_Industry_Portfolios_Daily.csv')
import statsmodels.api as sm
```

[22]:
```
Consumer_Sector=Industry_Portfolio['Cnsmr']
HiTech_Sector=Industry_Portfolio['HiTec']
```

[23]:
```
Small_Minus_Big=Farma_French['SMB']
High_Minus_Low=Farma_French['HML']
Rowbust_Minus_Weak=Farma_French['RMW']
```

Four factor farma model for Consumer

[24]:
```
R=sm.add_constant(Farma_French[['Mkt-RF','SMB','HML','RMW']])
Q=Consumer_Sector
D2=sm.OLS(Q,R).fit()
print(D2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Cnsmr   R-squared:                       0.909
Model:                            OLS   Adj. R-squared:                  0.908
Method:                 Least Squares   F-statistic:                     613.5
Date:                Wed, 15 Mar 2023   Prob (F-statistic):          2.51e-126
Time:                        03:00:49   Log-Likelihood:                -184.23
```

```
No. Observations:                 250   AIC:                             378.5
Df Residuals:                     245   BIC:                             396.1
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0012      0.032      0.036      0.972      -0.063       0.065
Mkt-RF         0.9639      0.025     39.122      0.000       0.915       1.012
SMB           -0.0013      0.069     -0.019      0.985      -0.136       0.134
HML           -0.1666      0.037     -4.504      0.000      -0.239      -0.094
RMW            0.1284      0.051      2.516      0.012       0.028       0.229
==============================================================================
Omnibus:                       22.204   Durbin-Watson:                   1.733
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               46.974
Skew:                          -0.434   Prob(JB):                     6.30e-11
Kurtosis:                       4.938   Cond. No.                         4.16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

The coefficient for the market risk premium (Mkt-RF) is 0.9639, indicating that a 1% increase in the market risk premium is associated with a 0.9639% increase in the Consumer variable, holding other variables constant. This coefficient is statistically significant at the 5% level. A p-value less than 0.05 indicates that the coefficient is statistically significant at the 5% level, meaning that there is strong evidence that the variable has a significant effect on the Consumer variable.In this case, the p-value is also very low, indicating that the model is significant.

Four factor farma model for HiTech

[25]:
```
A=sm.add_constant(Farma_French[['Mkt-RF','SMB','HML','RMW']])
B=HiTech_Sector
D3=sm.OLS(B,A).fit()
print(D3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   HiTec   R-squared:                       0.970
Model:                            OLS   Adj. R-squared:                  0.969
Method:                 Least Squares   F-statistic:                     1959.
Date:                Wed, 15 Mar 2023   Prob (F-statistic):          1.21e-184
Time:                        03:00:49   Log-Likelihood:                -93.939
No. Observations:                 250   AIC:                             197.9
Df Residuals:                     245   BIC:                             215.5
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
```

```
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0058      0.023     -0.256      0.798      -0.050       0.039
Mkt-RF         1.1249      0.017     65.523      0.000       1.091       1.159
SMB           -0.1791      0.048     -3.744      0.000      -0.273      -0.085
HML           -0.3394      0.026    -13.167      0.000      -0.390      -0.289
RMW            0.0311      0.036      0.873      0.383      -0.039       0.101
==============================================================================
Omnibus:                       17.250   Durbin-Watson:                   1.764
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               36.756
Skew:                          -0.312   Prob(JB):                     1.04e-08
Kurtosis:                       4.772   Cond. No.                         4.16
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

The R-squared value of 0.97 indicates that the model explains 97% of the variation in the dependent variable. This is a strong indication that the model fits the data well.Mkt-RF has a positive coefficient of 1.1249, which means that a 1% increase in Mkt-RF leads to a 1.1249% increase in HiTec.Overall, the model suggests that Mkt-RF has a positive impact on HiTec, while SMB and HML have a negative impact.

1.1.4 Problem B1

[26]:
```
Index_BTC_Data=pd.read_csv('Data.csv')
```

log returns for SP500, Russell and Bitcoin

[27]:
```
S_P500_log_Return=np.log(Index_BTC_Data['Adj Close']).diff().dropna()
SP_500_MEAN=S_P500_log_Return.mean()
S_P500_STD=np.std(Index_BTC_Data['Adj Close'])
print(S_P500_STD)
print('S&P 500 Log Return:',S_P500_log_Return)
Russ_log_Return=np.log(Index_BTC_Data['Russ_Close']).diff().dropna()
print('Russell 2000 Log Return:',Russ_log_Return)
BTC_log_Return=np.log(Index_BTC_Data['BTC_Close']).diff().dropna()
print('Bitcoin Log Return:',BTC_log_Return)
```

```
223.69184083058906
S&P 500 Log Return: 1      0.014584
2      0.018632
3     -0.001414
4      0.015446
5     -0.000687
          …
245    0.000435
246   -0.011595
```

```
247   -0.012272
248   -0.022137
249   -0.021183
Name: Adj Close, Length: 249, dtype: float64
Russell 2000 Log Return: 1      0.013936
2       0.030880
3       0.016750
4       0.010175
5      -0.009730
          …
245   -0.014890
246   -0.011137
247    0.000404
248   -0.028544
249   -0.029947
Name: Russ_Close, Length: 249, dtype: float64
Bitcoin Log Return: 1     -0.021812
2       0.008670
3       0.064425
4       0.022837
5       0.009406
          …
245    0.034478
246    0.006413
247   -0.004313
248   -0.051656
249   -0.017583
Name: BTC_Close, Length: 249, dtype: float64
```

Looking at the data, it appears that all three assets experienced both positive and negative log returns over the period. S&P 500 had an average log return of approximately 0.00054, while Russell 2000 had an average log return of approximately 0.00055. Bitcoin had a more volatile performance, with an average log return of approximately 0.00106.The data suggests that Bitcoin had a higher potential for returns than the other two assets, but it also came with a higher level of risk due to its higher volatility.

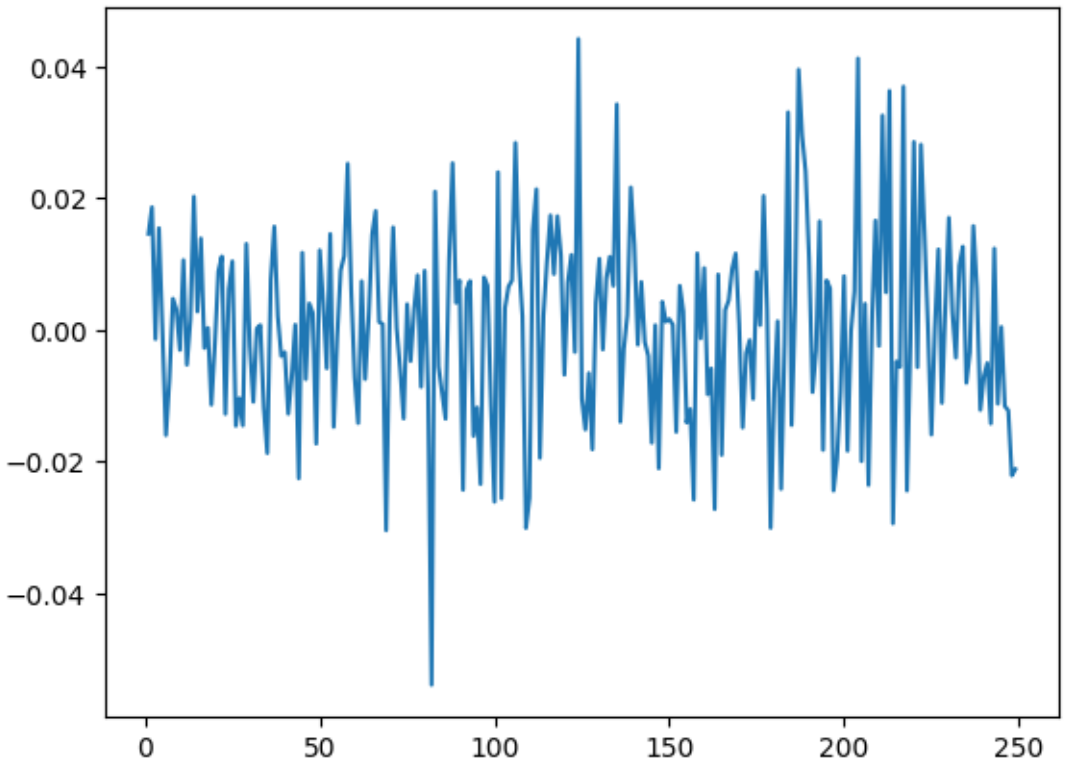
1.1.5 Problem B2

[28]:
```
df1 = S_P500_log_Return
df1.info()
df1.plot()
```

```
<class 'pandas.core.series.Series'>
Int64Index: 249 entries, 1 to 249
Series name: Adj Close
Non-Null Count  Dtype
--------------  -----
249 non-null    float64
```

```
dtypes: float64(1)
memory usage: 3.9 KB
```

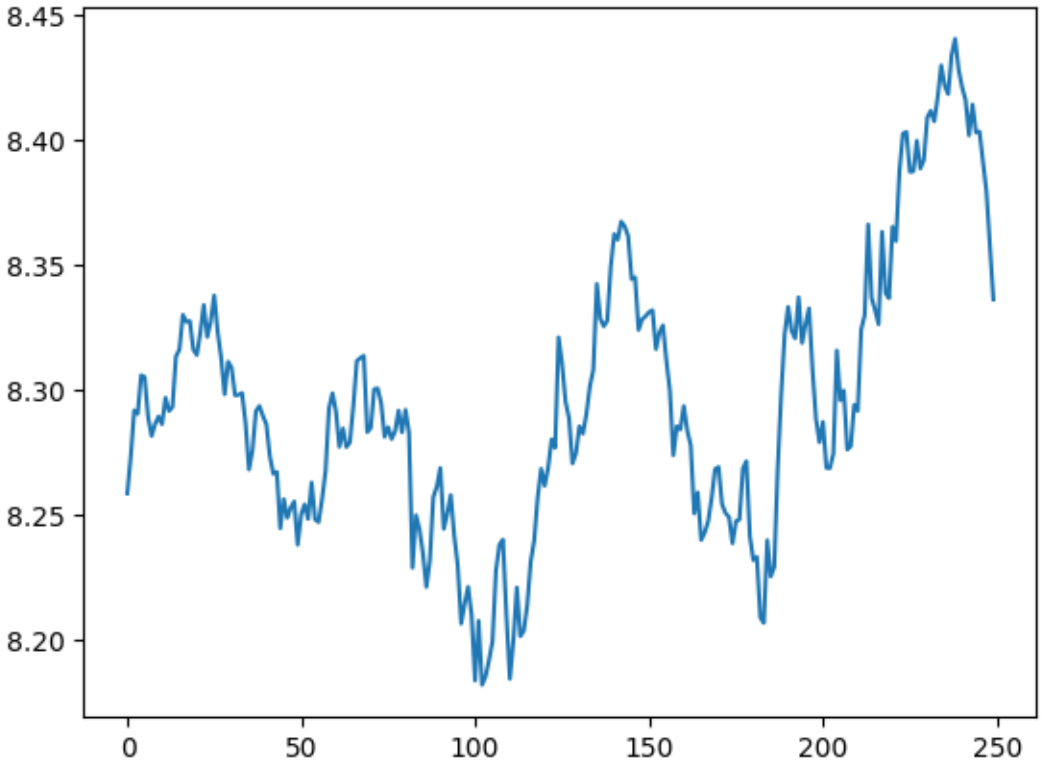
[28]: <AxesSubplot: >

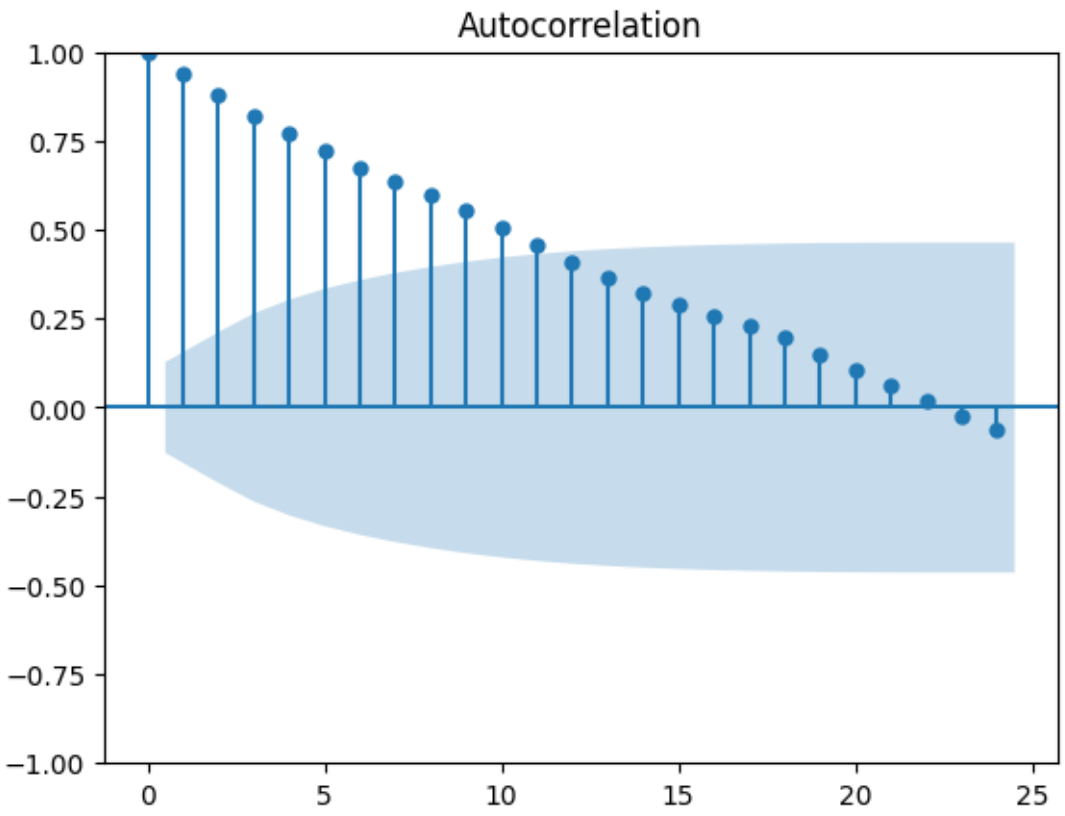


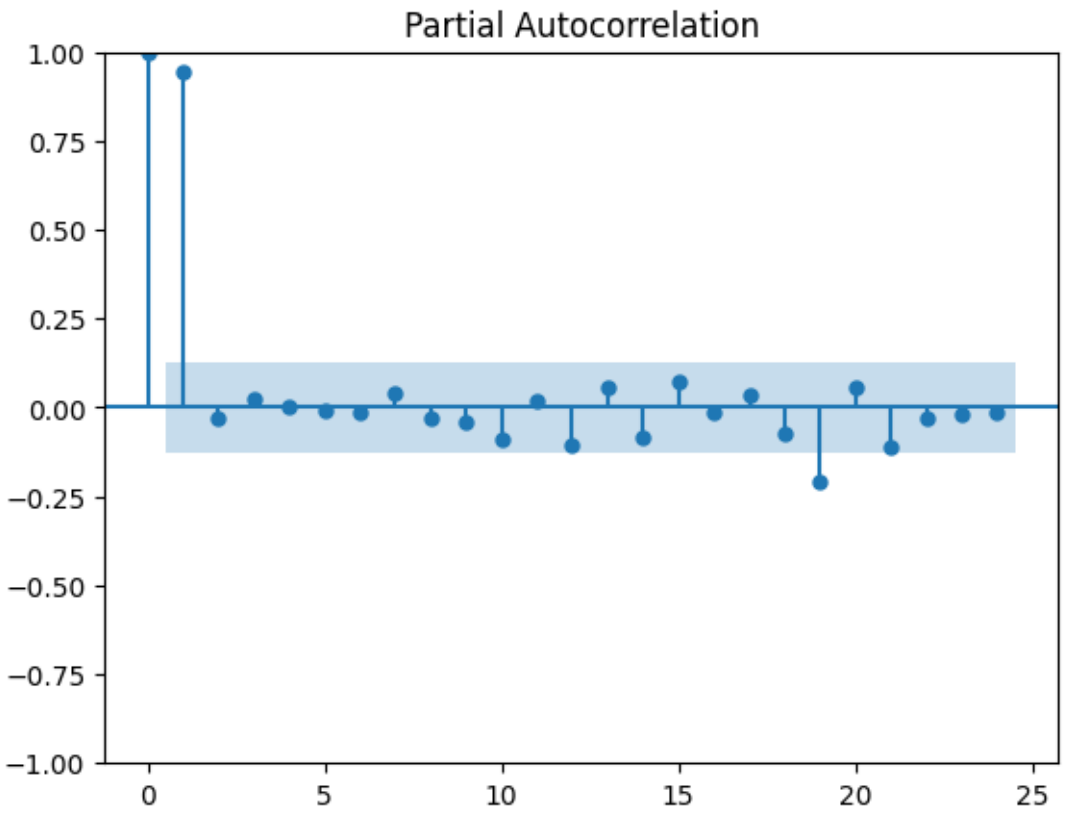
[29]:
```
import warnings
warnings.filterwarnings("ignore")
```

[30]:
```
#Exploring the dataset for log_ret_SP500
df= np.log(Index_BTC_Data['Adj Close'])
df.plot()
msk= (df.index< len(df)-15)
df_train= df[msk].copy()
df_test= df[~msk].copy()
#ACF plot and PACF plot
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
acf_original= plot_acf(df_train)
pacf_original= plot_pacf(df_train)
#ADF test
from statsmodels.tsa.stattools import adfuller
adf_test= adfuller(df_train)
print('P-value:', adf_test[1])
```

```
P-value: 0.4871722908386488
```



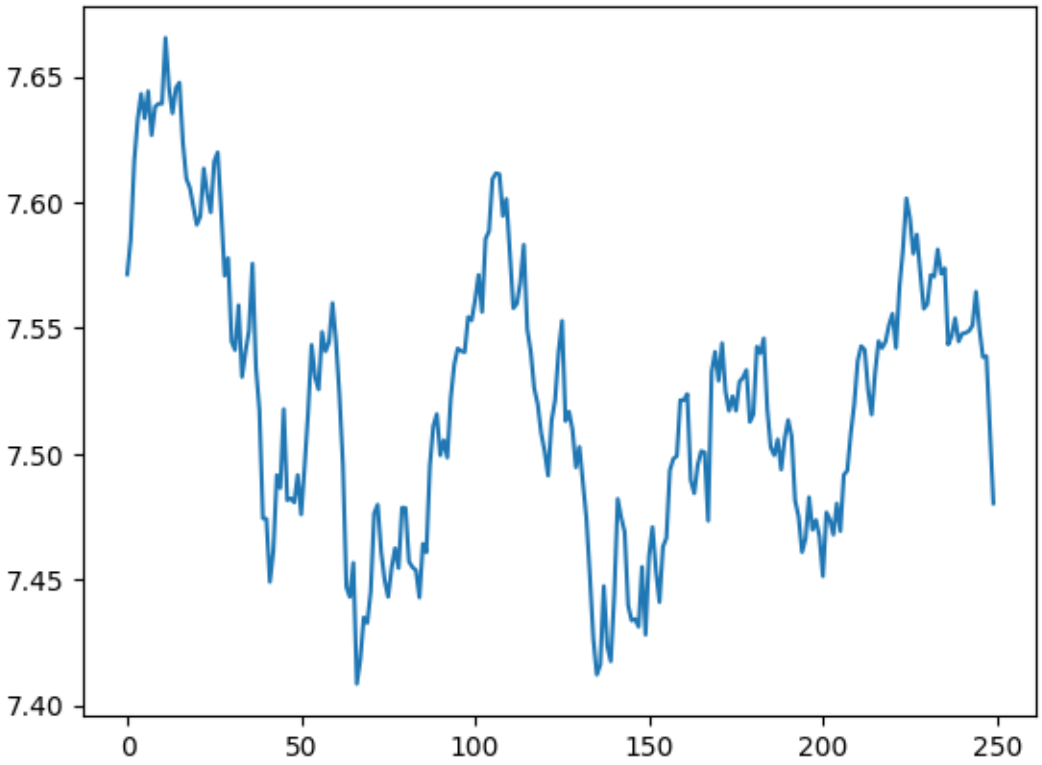

Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25

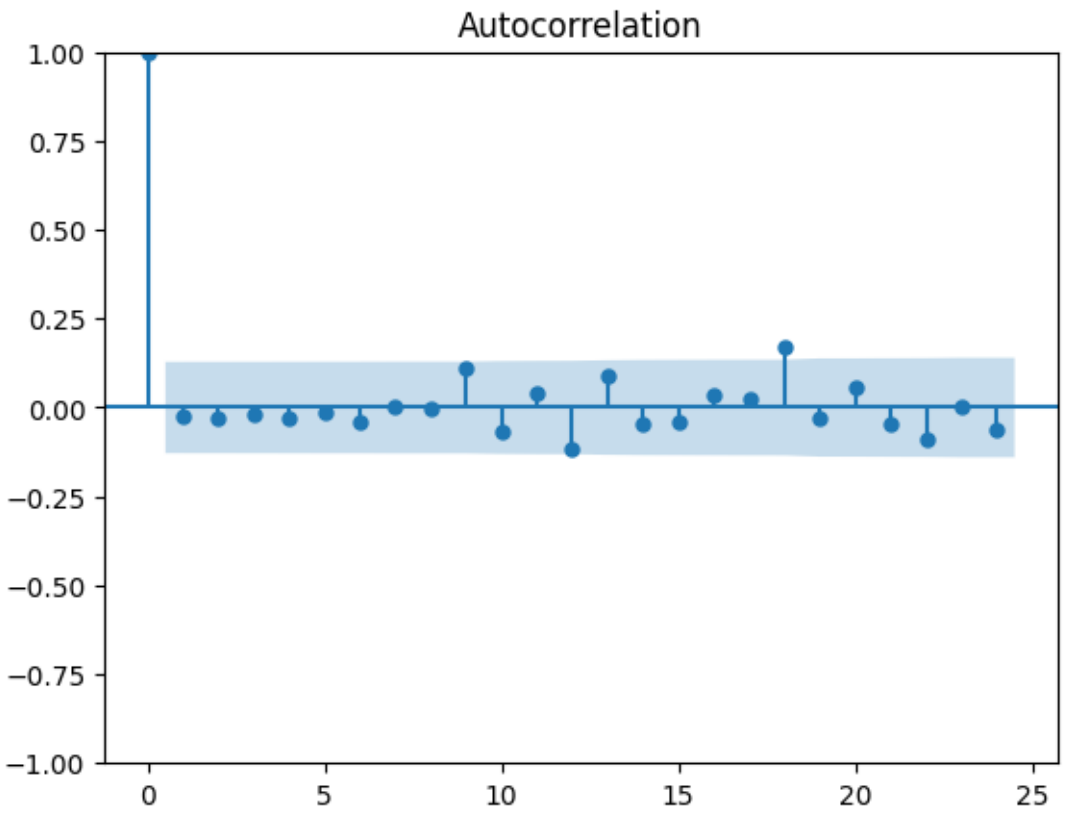


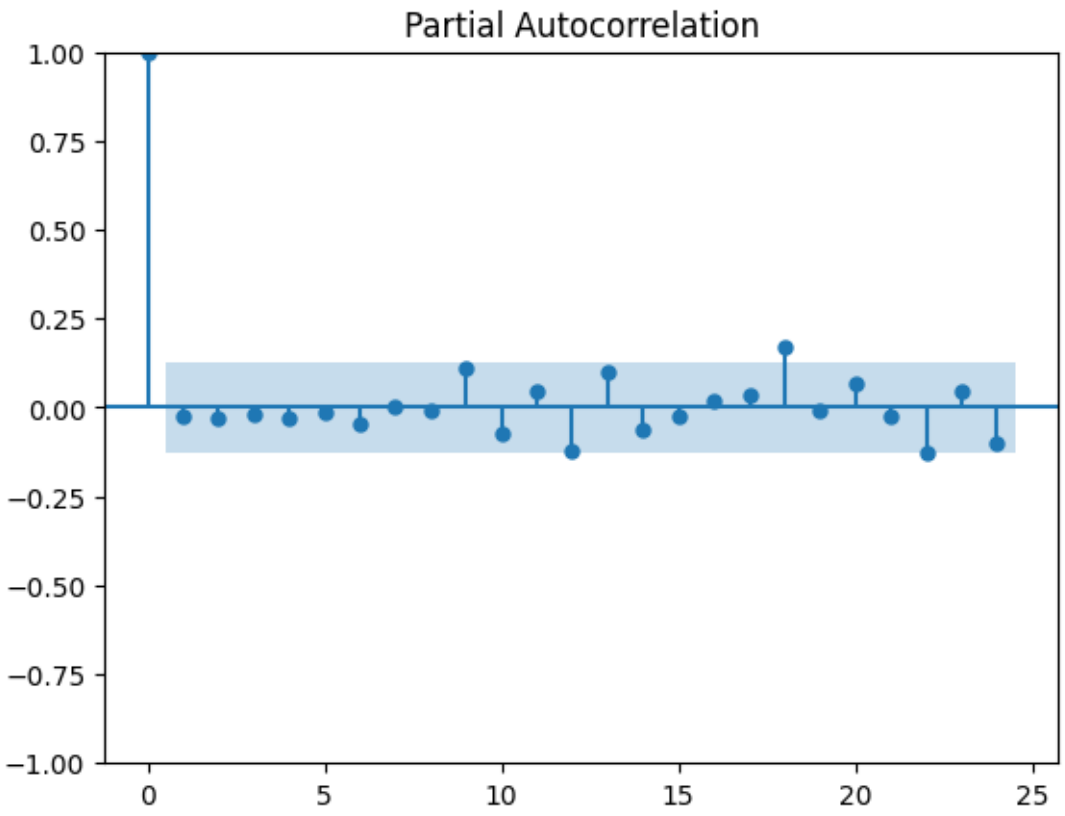
The p-value of 0.4871722908386488 is higher than the typical significance levels of 0.05 or 0.01, which indicates that there is not enough evidence to reject the null hypothesis that the time series has a unit root.

[31]:
```
#Exploring the dataset for log_ret_SP500
df2= np.log(Index_BTC_Data['Russ_Close'])
df2.plot()
msk1= (df1.index< len(df1)-15)
df_train1= df1[msk1].copy()
df_test1= df1[~msk1].copy()
#ACF plot and PACF plot
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
acf_original1= plot_acf(df_train1)
pacf_original1= plot_pacf(df_train1)
#ADF test
from statsmodels.tsa.stattools import adfuller
adf_test1= adfuller(df_train1)
print('P-value:', adf_test1[1])
```

```
P-value: 2.2455200311319585e-28
```


7.65
7.60
7.55
7.50
7.45
7.40
0
50
100
150
200
250

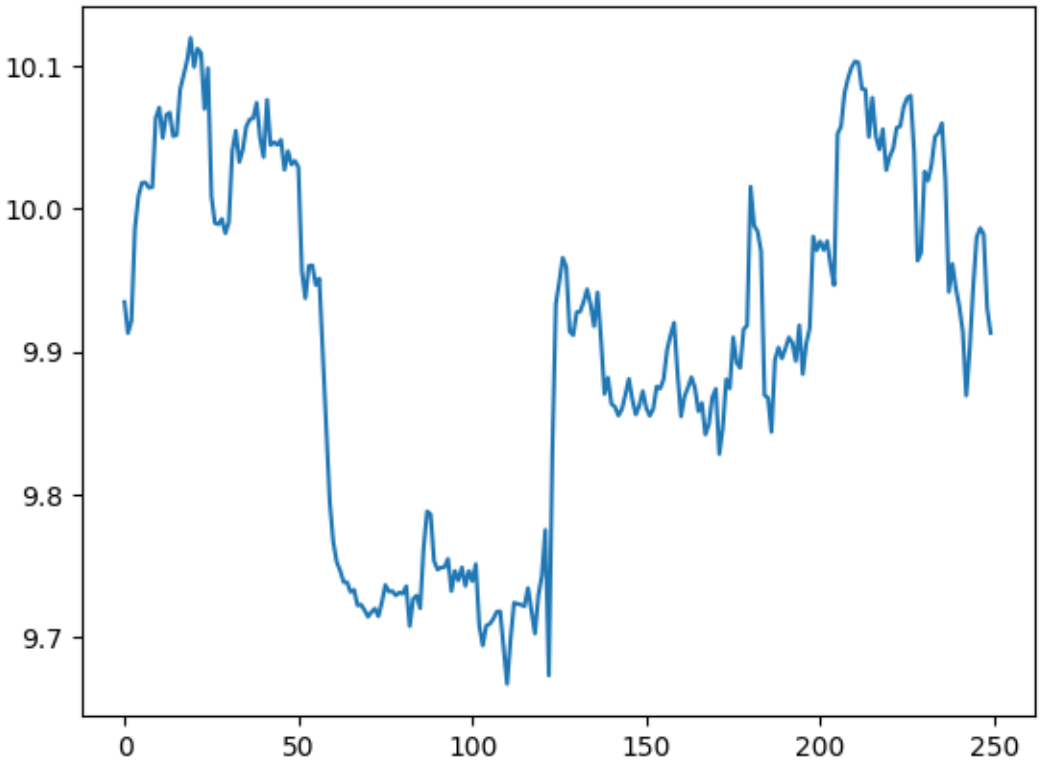

Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25

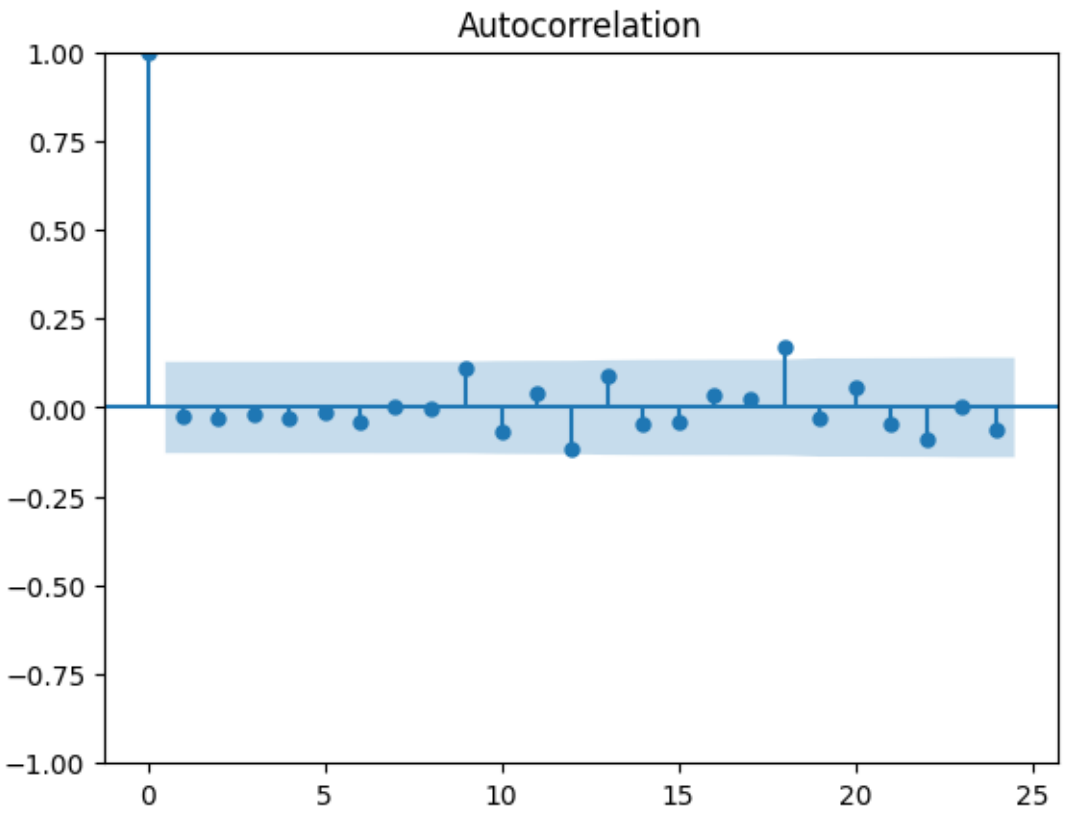


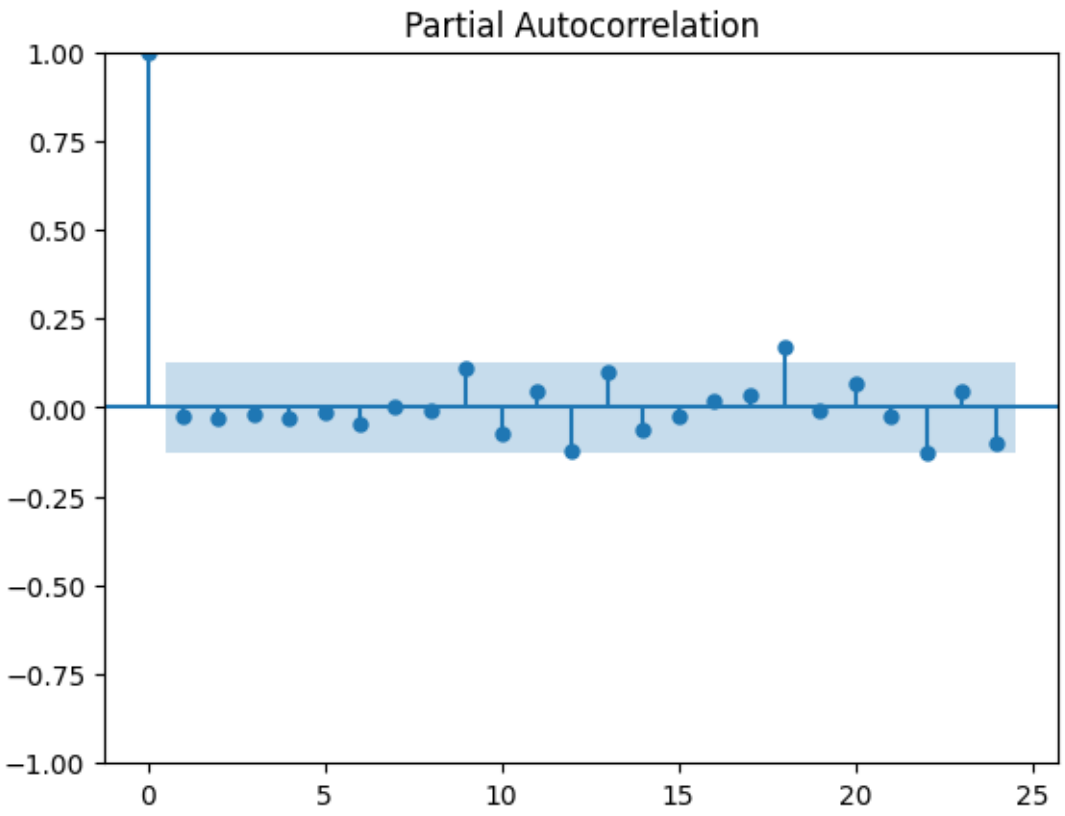
The p-value of 2.245e is less than the typical significance levels of 0.05 or 0.01, which indicates that there is strong evidence to reject the null hypothesis that the time series has a unit root and is non-stationary.

[32]:
```
#Exploring the dataset for log_ret_SP500
df3= np.log(Index_BTC_Data['BTC_Close'])
df3.plot()
msk1= (df1.index< len(df1)-15)
df_train1= df1[msk1].copy()
df_test1= df1[~msk1].copy()
#ACF plot and PACF plot
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
acf_original1= plot_acf(df_train1)
pacf_original1= plot_pacf(df_train1)
#ADF test
from statsmodels.tsa.stattools import adfuller
adf_test1= adfuller(df_train1)
print('P-value:', adf_test1[1])
```

```
P-value: 2.2455200311319585e-28
```


10.1
10.0
9.9
9.8
9.7
0
50
100
150
200
250


Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25



The p-value of 2.245e is less than the typical significance levels of 0.05 or 0.01, which indicates that there is strong evidence to reject the null hypothesis that the time series has a unit root and is non-stationary.

Calculate VAR, CVaR and Backtesting

[33]:
```
 # Set the confidence levels
conf_levels = [0.90, 0.95, 0.99]

    # Calculate VaR and CVaR for each asset and confidence levelreturns3
for i, returns in enumerate([S_P500_log_Return, Russ_log_Return,␣
 ↪BTC_log_Return]):
    print("asset(1)")
    for conf_level in conf_levels:
        # Calculate VaR
        var = -np.percentile(returns, conf_level*100)
        print(f"Var_SP500 ({conf_level*100}% C.L.): {var}")

        # Calculate CVaR
        cvar = -np.mean(returns[returns <= -var])
        print(f"CVaR ({conf_level*100}% C.L.): {cvar}")
```

```
    # Backtest the VaR estimates
    print("Backtesting results:")
    for conf_level in conf_levels:
        var = -np.percentile(returns, conf_level*100)
        expected_num_exceedances = len(returns[returns <= -var])
        actual_num_exceedances = len(returns[returns <= -np.mean(returns) -␣
↪var])
        print(f"VaR ({conf_level*100}% C.L.): Expected:␣
↪{expected_num_exceedances}, Actual: {actual_num_exceedances}")
```

```
asset(1)
Var_SP500 (90.0% C.L.): -0.017070001768314573
CVaR (90.0% C.L.): 0.0027217054277535735
Var_SP500 (95.0% C.L.): -0.02528686601335792
CVaR (95.0% C.L.): 0.0015274569628073824
Var_SP500 (99.0% C.L.): -0.038301151180350414
CVaR (99.0% C.L.): 0.00019264311329149952
Backtesting results:
VaR (90.0% C.L.): Expected: 224, Actual: 223
VaR (95.0% C.L.): Expected: 236, Actual: 235
VaR (99.0% C.L.): Expected: 246, Actual: 246
asset(1)
Var_SP500 (90.0% C.L.): -0.021910811396125142
CVaR (90.0% C.L.): 0.003606891051166259
Var_SP500 (95.0% C.L.): -0.026607783931959082
CVaR (95.0% C.L.): 0.002197904440711523
Var_SP500 (99.0% C.L.): -0.0329719534178972
CVaR (99.0% C.L.): 0.0009072931962702111
Backtesting results:
VaR (90.0% C.L.): Expected: 224, Actual: 226
VaR (95.0% C.L.): Expected: 236, Actual: 239
VaR (99.0% C.L.): Expected: 246, Actual: 246
asset(1)
Var_SP500 (90.0% C.L.): -0.02547480568191283
CVaR (90.0% C.L.): 0.005863816124916218
Var_SP500 (95.0% C.L.): -0.03838331781425059
CVaR (95.0% C.L.): 0.004006481735557207
Var_SP500 (99.0% C.L.): -0.10151843071367389
CVaR (99.0% C.L.): 0.0015772987537409826
Backtesting results:
VaR (90.0% C.L.): Expected: 224, Actual: 224
VaR (95.0% C.L.): Expected: 236, Actual: 236
VaR (99.0% C.L.): Expected: 246, Actual: 246
```

The VaR is an estimate of the maximum amount of money the portfolio is expected to lose with a certain level of confidence, while the CVaR is an estimate of the expected loss beyond the VaR.

Looking at the results, we can see that the SP500 and Russelll have negative VaR values, meaning that there is a high likelihood of a loss in the value of the assets, especially at higher confidence levels. On the other hand, BTC has a significantly larger negative VaR, indicating a higher risk of loss compared to the other two assets.

The backtesting results show that the actual VaR values were close to the expected values for all three assets, indicating that the VaR model is accurate and reliable

1.1.6 Problem C1

[34]:
```
Stock_Crypto=pd.read_csv('Stock_Crypto.csv')
import seaborn as sns
from random import gauss
from arch import arch_model

BNB_Close=Stock_Crypto['BNB_Close'].dropna()
print(BNB_Close)
ETH_Close=Stock_Crypto['Eth_Close'].dropna()
print(ETH_Close)
TM_Close=Stock_Crypto['TM_Adj Close'].dropna()
TSLA_Close=Stock_Crypto['TSLA_Adj Close'].dropna()
```

```
0      277.96
1      277.42
2      277.30
3      286.87
4      289.34
        …
245    243.24
246    240.75
247    241.54
248    238.50
249    230.85
Name: BNB_Close, Length: 250, dtype: float64
0      1482.62
1      1429.16
2      1438.66
3      1534.09
4      1561.93
        …
245    1216.98
246    1222.51
247    1237.59
248    1186.97
249    1134.54
Name: Eth_Close, Length: 250, dtype: float64
```

(a) calculate log returns for each of these assets

[35]:
```
BNB_Log_Return=np.log(BNB_Close).diff().dropna()
print('Binance Coin Log Returns:',BNB_Log_Return)
TM_Log_Return=np.log(TM_Close).diff().dropna()
print('Toyota Log Returns:',TM_Log_Return)
TSLA_Log_Return=np.log(TSLA_Close).diff().dropna()
print('Tesla Log Returns:',TSLA_Log_Return)
ETH_Log_Return=np.log(ETH_Close).diff().dropna()
print('Ethereum Log Returns:',ETH_Log_Return)
```

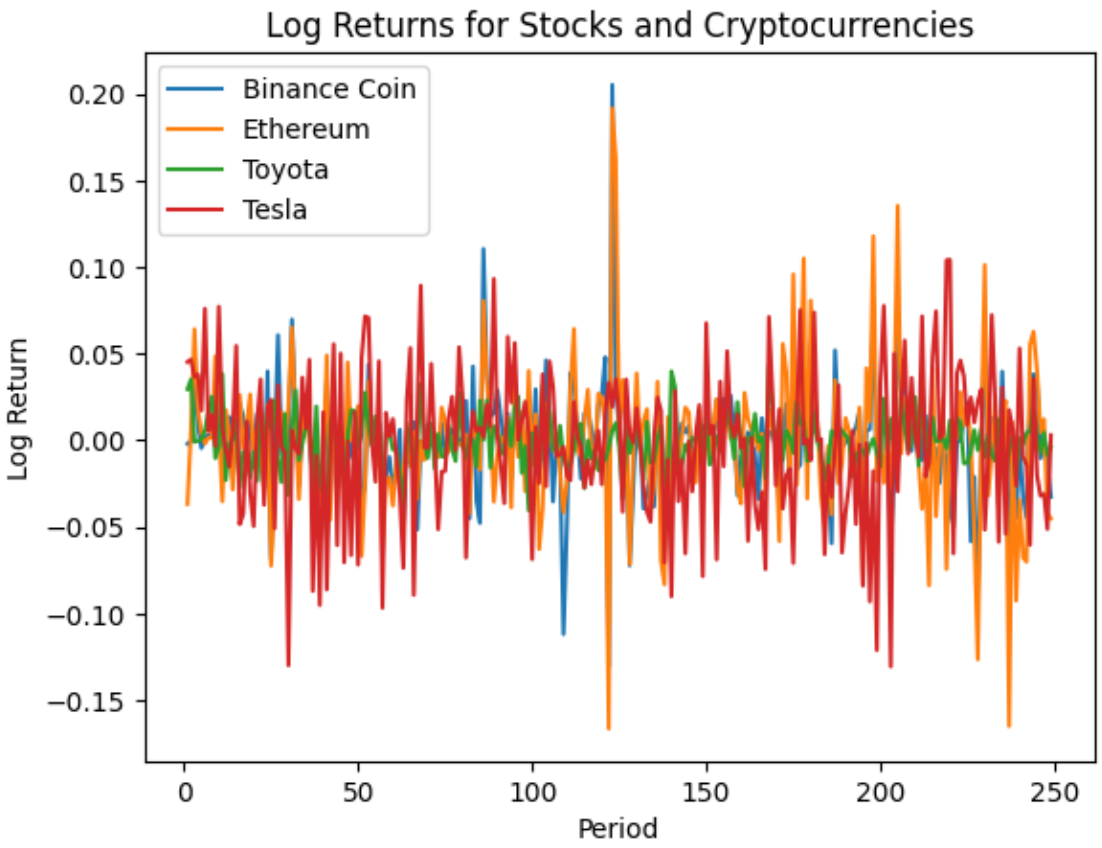
```
Binance Coin Log Returns: 1     -0.001945
2     -0.000433
3      0.033929
4      0.008573
5     -0.004468
          …
245    0.035911
246   -0.010290
247    0.003276
248   -0.012666
249   -0.032601
Name: BNB_Close, Length: 249, dtype: float64
Toyota Log Returns: 1      0.029603
2      0.035289
3     -0.000873
4     -0.000233
5      0.000582
          …
245    0.002143
246   -0.007521
247    0.004520
248   -0.008554
249   -0.003978
Name: TM_Adj Close, Length: 249, dtype: float64
Tesla Log Returns: 1      0.045306
2      0.046704
3      0.036655
4      0.038035
5      0.017268
          …
245   -0.020328
246   -0.031980
247   -0.030892
248   -0.051178
249    0.003003
Name: TSLA_Adj Close, Length: 249, dtype: float64
Ethereum Log Returns: 1     -0.036724
2      0.006625
```

```
3      0.064225
4      0.017985
5      0.003496
         …
245    0.040737
246    0.004534
247    0.012260
248   -0.041762
249   -0.045177
Name: Eth_Close, Length: 249, dtype: float64
```

Interpreting the values of the log returns, we can see that Binance Coin and Ethereum have experienced both positive and negative returns with high volatility, indicating high risk. Tesla has also shown high volatility, but with a consistently positive return, indicating a high potential for gain. Toyota, on the other hand, has shown lower volatility and a consistent positive return, indicating a relatively stable and less risky investment option.

plotting the log returns on the same chart

[36]:
```
plt.plot(BNB_Log_Return, label='Binance Coin')
plt.plot(ETH_Log_Return, label='Ethereum')
plt.plot(TM_Log_Return, label='Toyota')
plt.plot(TSLA_Log_Return, label='Tesla')
plt.title('Log Returns for Stocks and Cryptocurrencies')
plt.xlabel('Period')
plt.ylabel('Log Return')
plt.legend()
plt.show()
```

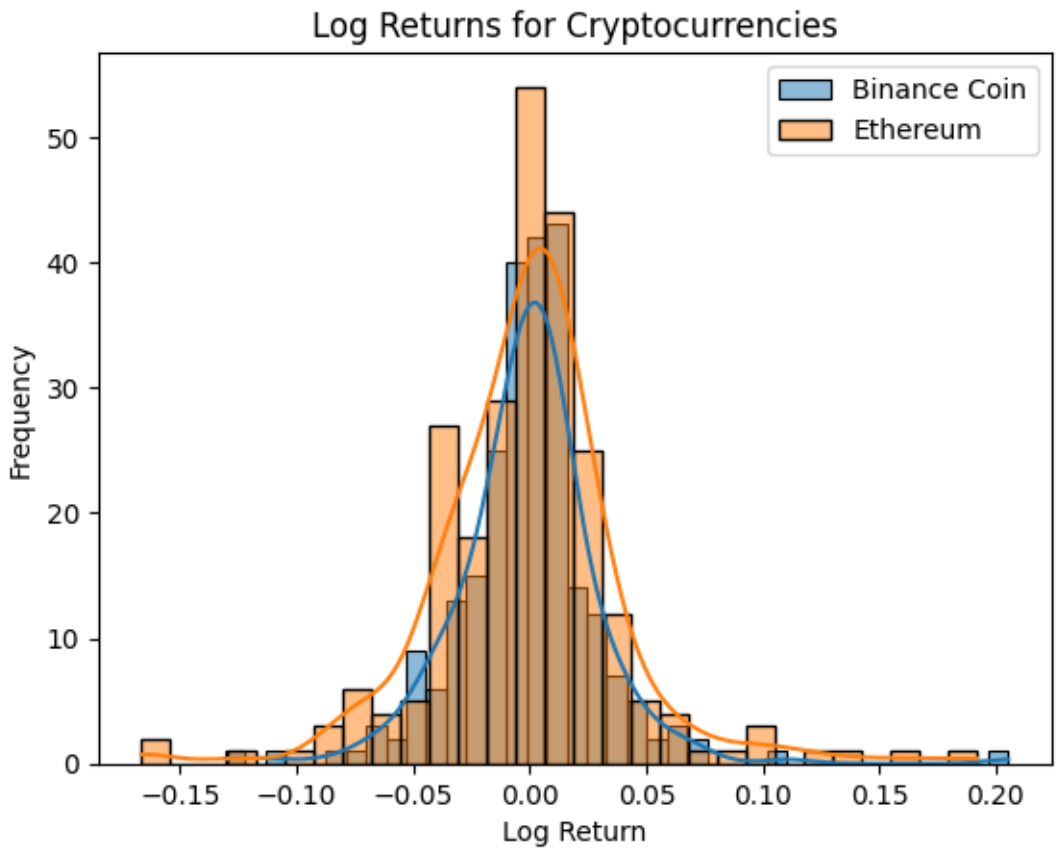


Binance Coin has a higher proportion of negative log returns compared to the other assets, indicating higher volatility on the downside. Tesla, on the other hand, has a higher proportion of positive log returns, indicating higher volatility on the upside.

If an investor is risk-averse, they may prefer assets that have lower downside volatility, such as Toyota. On the other hand, if an investor is more risk-tolerant and looking for higher potential returns, they may be more interested in assets that have higher upside volatility, such as Tesla.

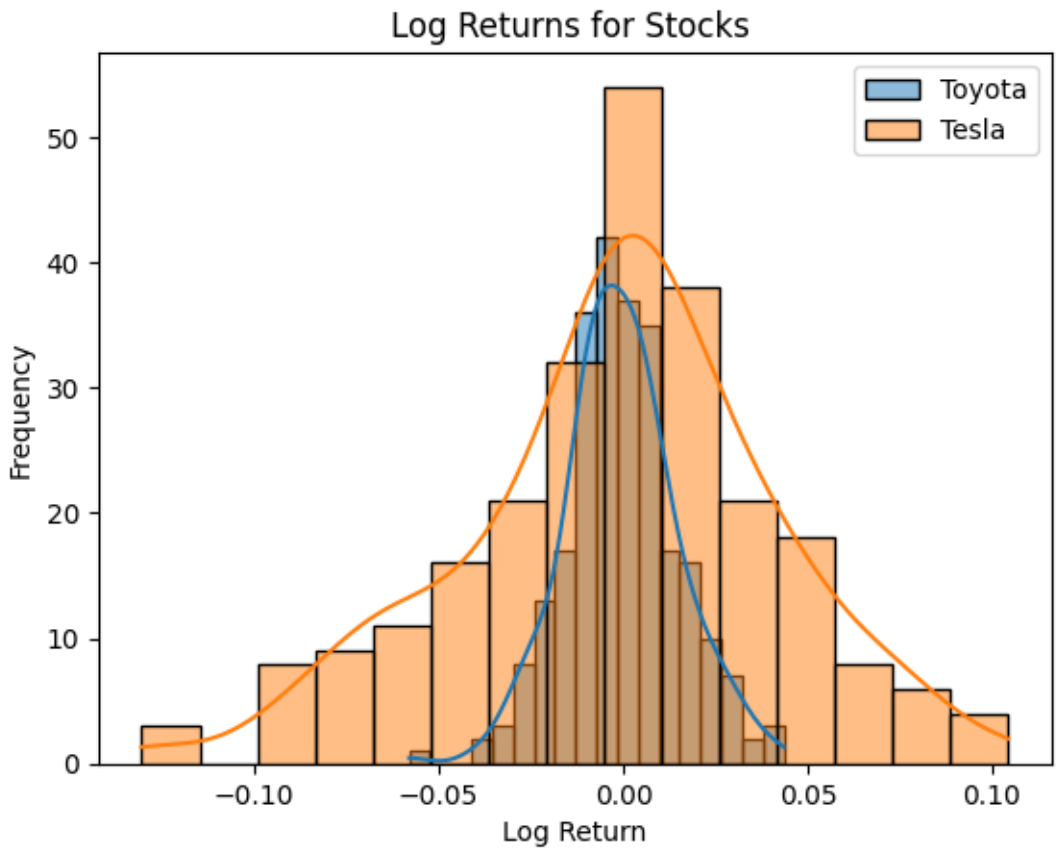
(b) Plot the distribution of these log returns

[37]:
```
sns.histplot(BNB_Log_Return,kde=True, label='Binance Coin')
sns.histplot(ETH_Log_Return,kde=True, label='Ethereum')
plt.title('Log Returns for Cryptocurrencies')
plt.xlabel('Log Return')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```



We can see that Log return of Ethereum is much higher than that of the Binance. Both the data at the end of the day, manage us to give the close between -0.0% to 0.05. However, in terms of negative log return, Ethereum has made a loss of more than -15% whereas Microsoft has made the loss of -11%.

[38]:
```
sns.histplot(TM_Log_Return, kde=True,label='Toyota')
sns.histplot(TSLA_Log_Return, kde=True,label='Tesla')
plt.title('Log Returns for Stocks ')
plt.xlabel('Log Return')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

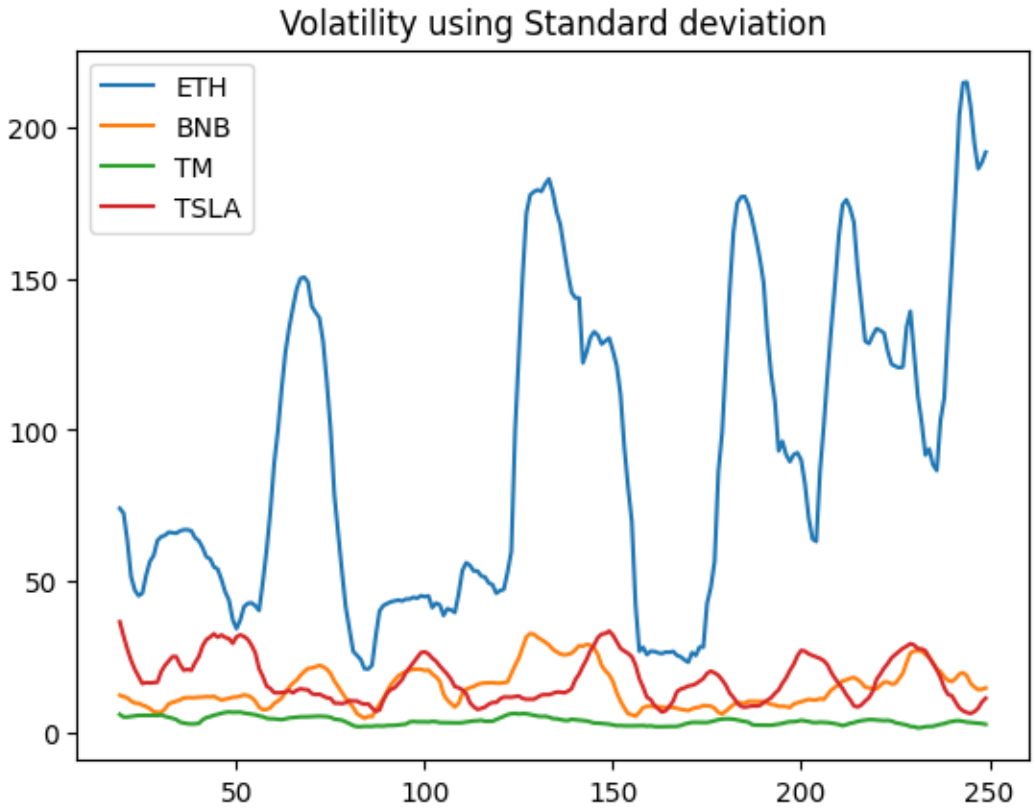


We can see that Log return of Tesla is much higher than that of the Toyota. Both the data at the end of the day, manage us to give the close between -0.05 to 0.05.However, in terms of negative log return, Tesla has made a loss of more than -10% whereas Toyota has made the loss of -6%.Toyota has more stable return as compared with Tesla. We can see that Tesla has more fat tailed kurtosis as compared to Toyota which has thin tail. It suggests that the underlying asset is more prone to large positive and negative returns, which can increase the risk of an assest.

1.1.7 Problem C2

[39]:
```
# calculating the standard deviation for each asset
rolling_std_ETH = ETH_Close.rolling(window=20).std()
rolling_std_BNB = BNB_Close.rolling(window=20).std()
rolling_std_TM = TM_Close.rolling(window=20).std()
rolling_std_TSLA = TSLA_Close.rolling(window=20).std()
plt.plot(rolling_std_ETH ,label='ETH')
plt.plot(rolling_std_BNB ,label='BNB')
plt.plot(rolling_std_TM ,label='TM')
plt.plot(rolling_std_TSLA ,label='TSLA')
plt.title('Volatility using Standard deviation')
plt.legend()
```

```
plt.show()
```



From the plot, we can see that the volatility of all four assets varies over time. Ethereum Coin appears to be the most volatile, followed by Binance and Tesla, while Toyota is the least volatile of the four assets. The volatility of all four assets increased sharply in early 2021, which is consistent with the overall increase in volatility in financial markets during that period.

[40]:
```
BNB_Excess_Return=(BNB_Log_Return-Risk_Free).dropna()
ETH_Excess_Return=(ETH_Log_Return-Risk_Free).dropna()
TM_Excess_Return=(TM_Log_Return-Risk_Free).dropna()
TSLA_Excess_Return=(TSLA_Log_Return-Risk_Free).dropna()
```

[41]:
```
X=Index_Excess_Return.values.reshape(-1,1)
Y=BNB_Excess_Return
reg=LinearRegression().fit(X,Y)
BNB_Beta=reg.coef_[0]
print('BNB Beta:',BNB_Beta)

X=Index_Excess_Return.values.reshape(-1,1)
Y=ETH_Excess_Return
reg=LinearRegression().fit(X,Y)
```

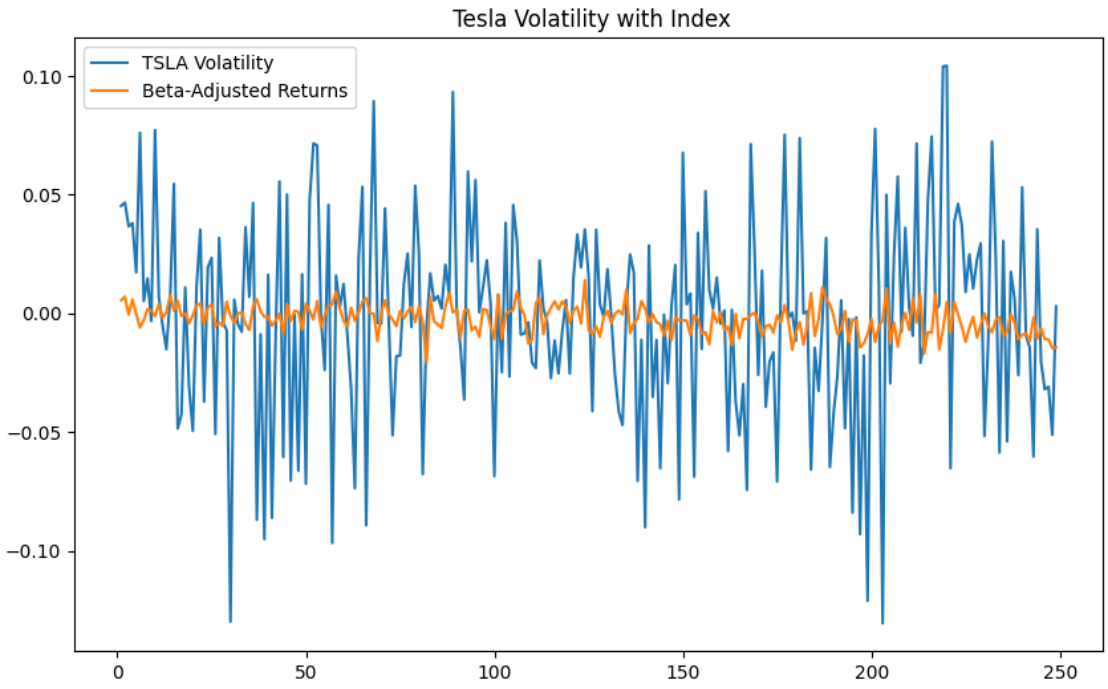
```
ETH_Beta=reg.coef_[0]
print('ETH Beta:',ETH_Beta)

X=Index_Excess_Return.values.reshape(-1,1)
Y=TM_Excess_Return
reg=LinearRegression().fit(X,Y)
TM_Beta=reg.coef_[0]
print('TM Beta:',TM_Beta)

X=Index_Excess_Return.values.reshape(-1,1)
Y=TSLA_Excess_Return
reg=LinearRegression().fit(X,Y)
TSLA_Beta=reg.coef_[0]
print('TSLA Beta:',TSLA_Beta)
```

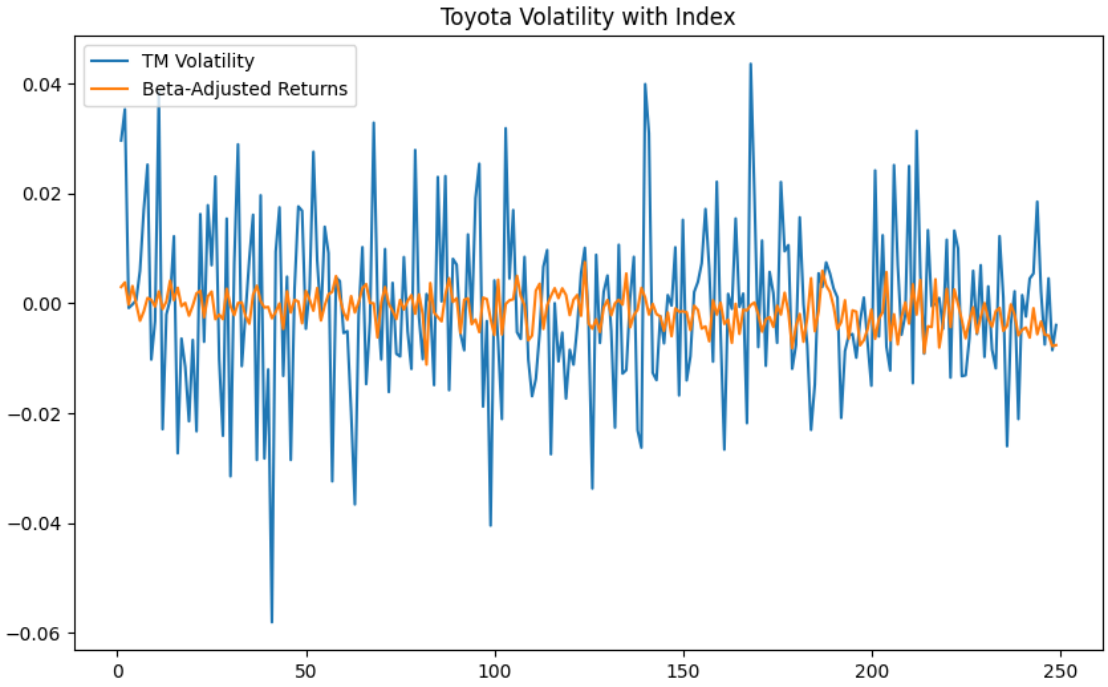
```
BNB Beta: 0.23538959795932562
ETH Beta: 0.3261230075197696
TM Beta: 0.20107333759013216
TSLA Beta: 0.37625916844602897
```

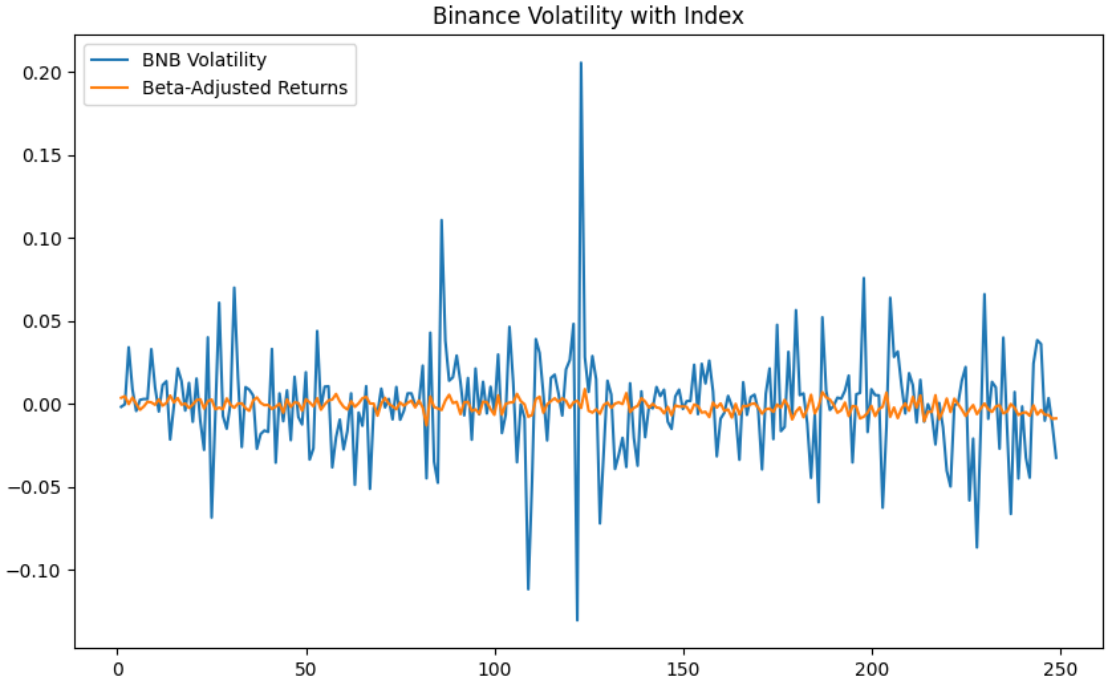
From the results, we can see that all four assets have betas less than 1, which indicates that they are less volatile than the market as represented by the S&P 500 index. Among the four assets, Tesla has the highest beta of 0.376, which means it is relatively more volatile and has a higher degree of sensitivity to market movements compared to the other assets. On the other hand, Toyota has the lowest beta of 0.201, which means it is the least volatile and has the least sensitivity to market movements among the four assets.

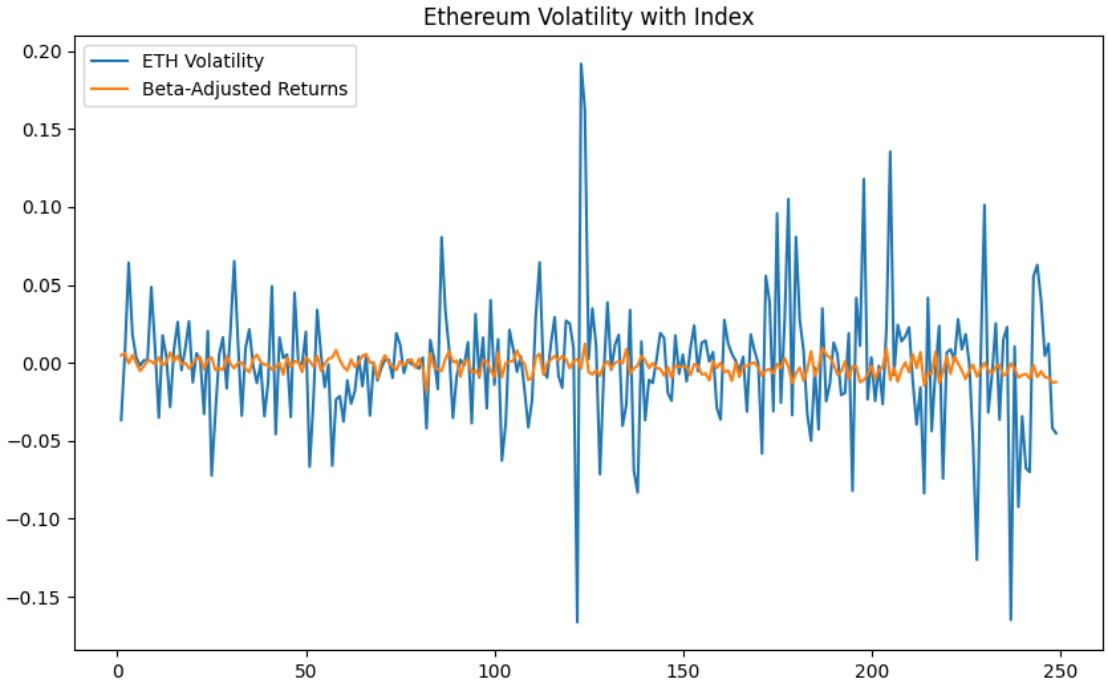
[42]:
```
beta_adjusted_returns = Index_Excess_Return * TSLA_Beta
fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(TSLA_Log_Return, label='TSLA Volatility')
ax.plot(beta_adjusted_returns, label='Beta-Adjusted Returns')
ax.set_title(' Tesla Volatility with Index')
ax.legend(loc='upper left')
plt.show()

beta_adjusted_returns1 = Index_Excess_Return * TM_Beta
fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(TM_Log_Return, label='TM Volatility')
ax.plot(beta_adjusted_returns1, label='Beta-Adjusted Returns')
ax.set_title('Toyota Volatility with Index')
ax.legend(loc='upper left')
plt.show()

beta_adjusted_returns2 = Index_Excess_Return * BNB_Beta
fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(BNB_Log_Return, label='BNB Volatility')
ax.plot(beta_adjusted_returns2, label='Beta-Adjusted Returns')
ax.set_title('Binance Volatility with Index')
```

```
ax.legend(loc='upper left')
plt.show()

beta_adjusted_returns3 = Index_Excess_Return * ETH_Beta
fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(ETH_Log_Return, label='ETH Volatility')
ax.plot(beta_adjusted_returns3, label='Beta-Adjusted Returns')
ax.set_title('Ethereum Volatility with Index')
ax.legend(loc='upper left')
plt.show()
```




Toyota Volatility with Index
TM Volatility
Beta-Adjusted Returns
0.04
0.02
0.00
−0.02
−0.04
−0.06
0
50
100
150
200
250


Binance Volatility with Index
BNB Volatility
Beta-Adjusted Returns
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
0
50
100
150
200
250



1 The blue line represents the volatility of Tesla's stock price, while the orange line represents the beta-adjusted returns of Tesla, which show how much of Tesla's returns can be attributed to the market index. The orange line is below the blue line, it suggests that Tesla's stock price is underperforming the market index, after adjusting for its beta.

2 we can see that the beta continue to fluctuate with volatility. It is keen to be observed that the blue line and orange line has been moving closely together.The blue line is above the orange line which suggest that Toyota shares are underperfoming the market index.

3 The Binance has been moving in quite volatility in the past one year. We can see that the orange line is above the blue line, which suggests that Binance's stock price is performing better than the market index, after adjusting for its beta

4 The Ethuruem chart also looks very similiar to Binance chart. In this scenerio again the orange line is above the blue line, which suggests that Ethereum's stock price is performing better than the market index, after adjusting for its beta

1.1.8 Problem C3

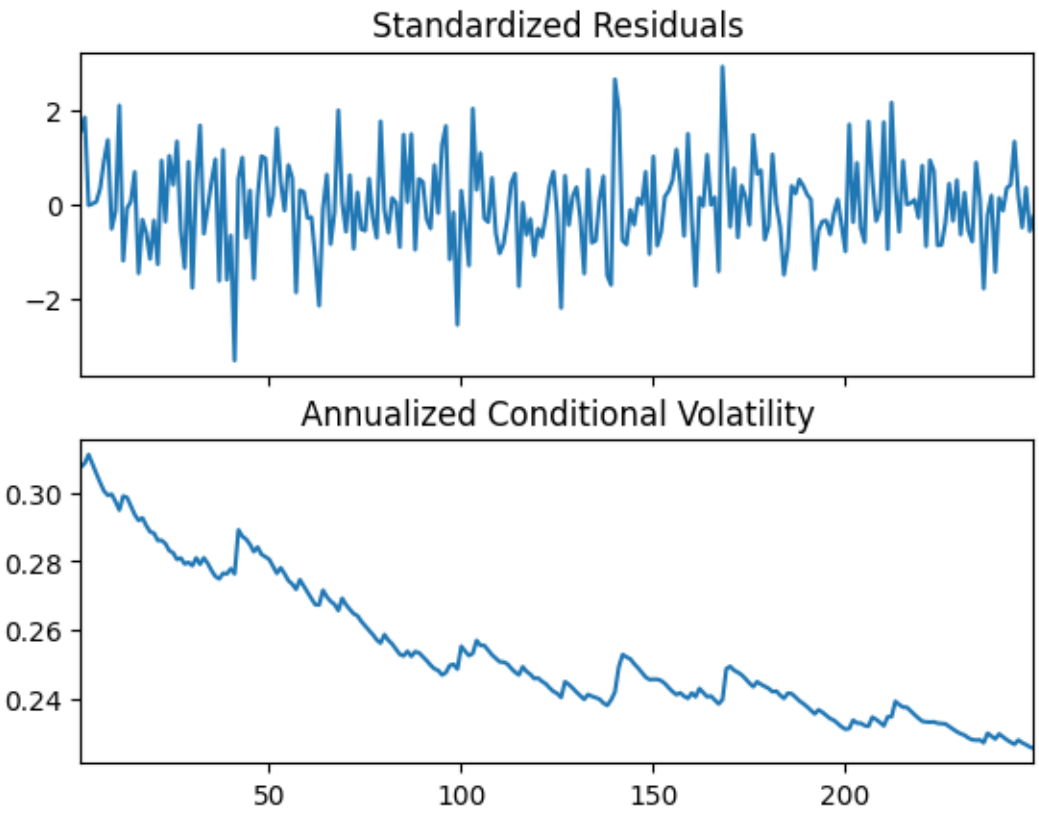
Estimate GARCH(1,1) model for Toyota Motors

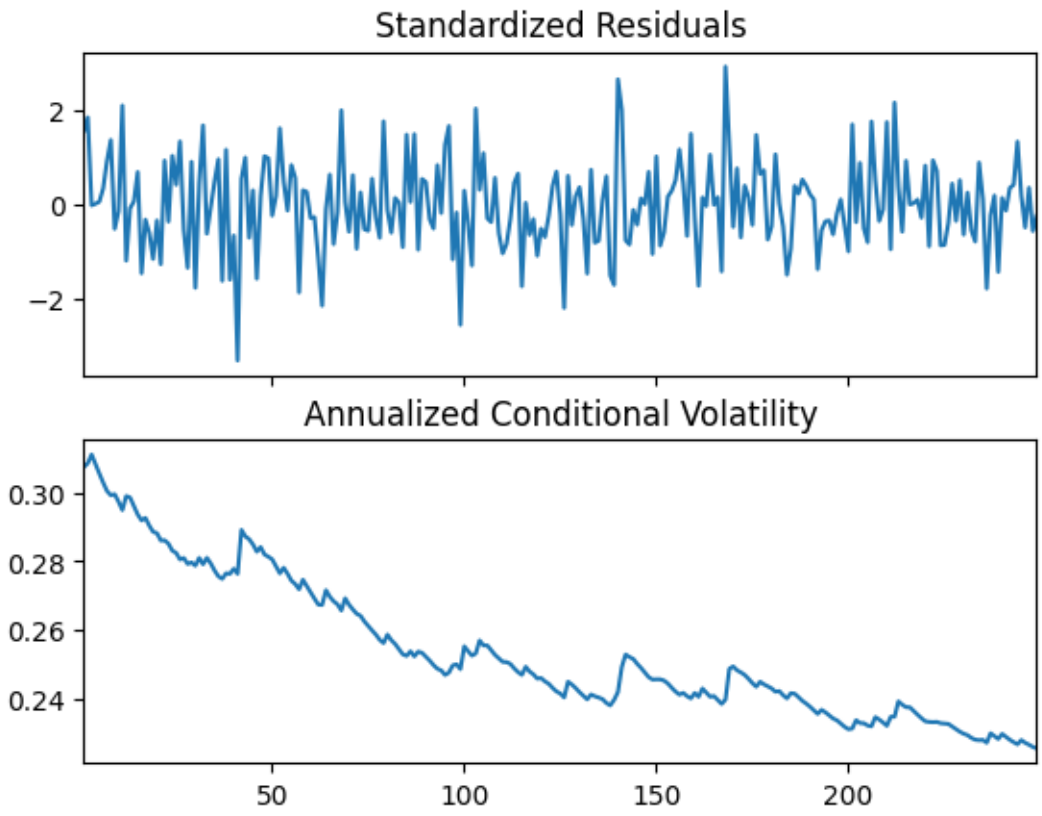
[43]:
```
model1 = arch_model(TM_Log_Return, vol='GARCH', p=1, q=1, mean='constant',␣
↪dist='Normal')
results1 = model1.fit()

# Print summary of results for Toyota Motors
print(results1.summary())
```

```
results1.plot(annualize='D')
```

```
Iteration:      1,   Func. Count:      5,   Neg. LLF: -690.4656021817705
Optimization terminated successfully    (Exit mode 0)
            Current function value: -690.4656095058681
            Iterations: 5
            Function evaluations: 5
            Gradient evaluations: 1
                    Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:           TM_Adj Close   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:                690.466
Distribution:                  Normal   AIC:                          -1372.93
Method:            Maximum Likelihood   BIC:                          -1358.86
                                        No. Observations:                  249
Date:                Wed, Mar 15 2023   Df Residuals:                      248
Time:                        03:00:54   Df Model:                            1
                                 Mean Model
============================================================================
                 coef    std err          t      P>|t|        95.0% Conf. Int.
----------------------------------------------------------------------------
mu         -6.2220e-04  6.168e-05    -10.087  6.318e-24 [-7.431e-04,-5.013e-04]
                            Volatility Model
============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
----------------------------------------------------------------------------
omega      4.6878e-06  4.597e-11  1.020e+05      0.000  [4.688e-06,4.688e-06]
alpha[1]       0.0100  4.380e-02      0.228      0.819 [-7.585e-02,9.585e-02]
beta[1]        0.9700  3.659e-02     26.511 7.259e-155      [  0.898,  1.042]
============================================================================

Covariance estimator: robust
```

[43]:


Standardized Residuals
2
0
−2
Annualized Conditional Volatility
0.30
0.28
0.26
0.24
50
100
150
200



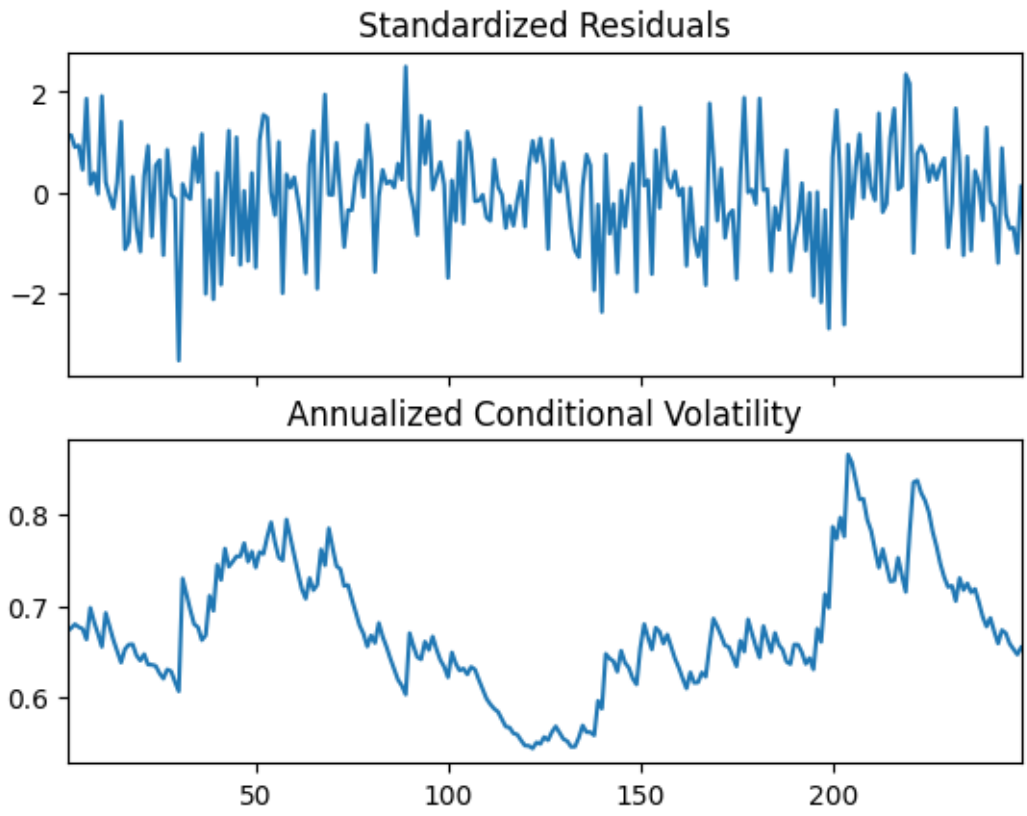
Estimate GARCH(1,1) model for Tesla

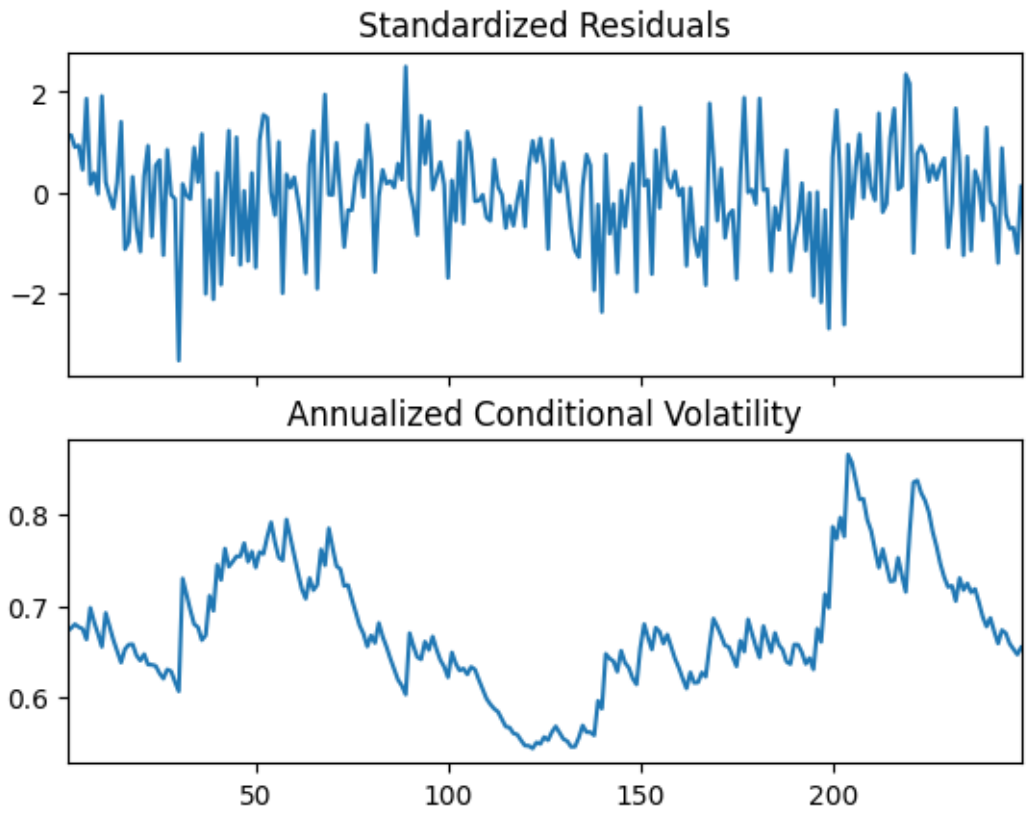
[44]:
```
# Estimate GARCH(1,1) model for Tesla
model2 = arch_model(TSLA_Log_Return, vol='GARCH', p=1, q=1, mean='constant',␣
 ↪dist='Normal')
results2 = model2.fit()

# Print summary of results for Tesla
print(results2.summary())
results2.plot(annualize='D')
```

```
Iteration:      1,   Func. Count:      6,   Neg. LLF: 352215.57364146283
Iteration:      2,   Func. Count:     16,   Neg. LLF: 175.69319025718403
Iteration:      3,   Func. Count:     25,   Neg. LLF: -430.8950618822198
Iteration:      4,   Func. Count:     34,   Neg. LLF: -431.38888579686426
Iteration:      5,   Func. Count:     40,   Neg. LLF: -430.7512719893801
Iteration:      6,   Func. Count:     46,   Neg. LLF: -434.03373006917593
Iteration:      7,   Func. Count:     52,   Neg. LLF: -434.2428482449343
Iteration:      8,   Func. Count:     58,   Neg. LLF: -434.2533628294861
Iteration:      9,   Func. Count:     63,   Neg. LLF: -434.2533777787196
Iteration:     10,   Func. Count:     67,   Neg. LLF: -434.2533777787285
```

```
Optimization terminated successfully    (Exit mode 0)
            Current function value: -434.2533777787196
            Iterations: 10
            Function evaluations: 67
            Gradient evaluations: 10
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:         TSLA_Adj Close   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:                434.253
Distribution:                  Normal   AIC:                           -860.507
Method:            Maximum Likelihood   BIC:                           -846.437
                                        No. Observations:                  249
Date:                Wed, Mar 15 2023   Df Residuals:                      248
Time:                        03:00:54   Df Model:                            1
                                Mean Model
==============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
------------------------------------------------------------------------------
mu         -1.7329e-03  2.568e-03     -0.675      0.500 [-6.767e-03,3.301e-03]
                            Volatility Model
==============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
------------------------------------------------------------------------------
omega      7.5148e-05  3.079e-05      2.441  1.466e-02  [1.480e-05,1.355e-04]
alpha[1]       0.0426  2.858e-02      1.490      0.136 [-1.342e-02,9.859e-02]
beta[1]        0.9160  2.107e-02     43.470      0.000      [  0.875,  0.957]
==============================================================================

Covariance estimator: robust
```

[44]:


Standardized Residuals
2
0
−2
Annualized Conditional Volatility
0.8
0.7
0.6
50
100
150
200



The estimated parameters for the mean equation are mu = -0.00062, which indicates that the mean return is slightly negative, but not significantly different from zero. Alpha is quite small, indicating that recent squared errors have relatively little influence on the current volatility estimate. Beta is close to one, indicating that past volatility estimates have a large influence on the current estimate.

1.1.9 Problem C4

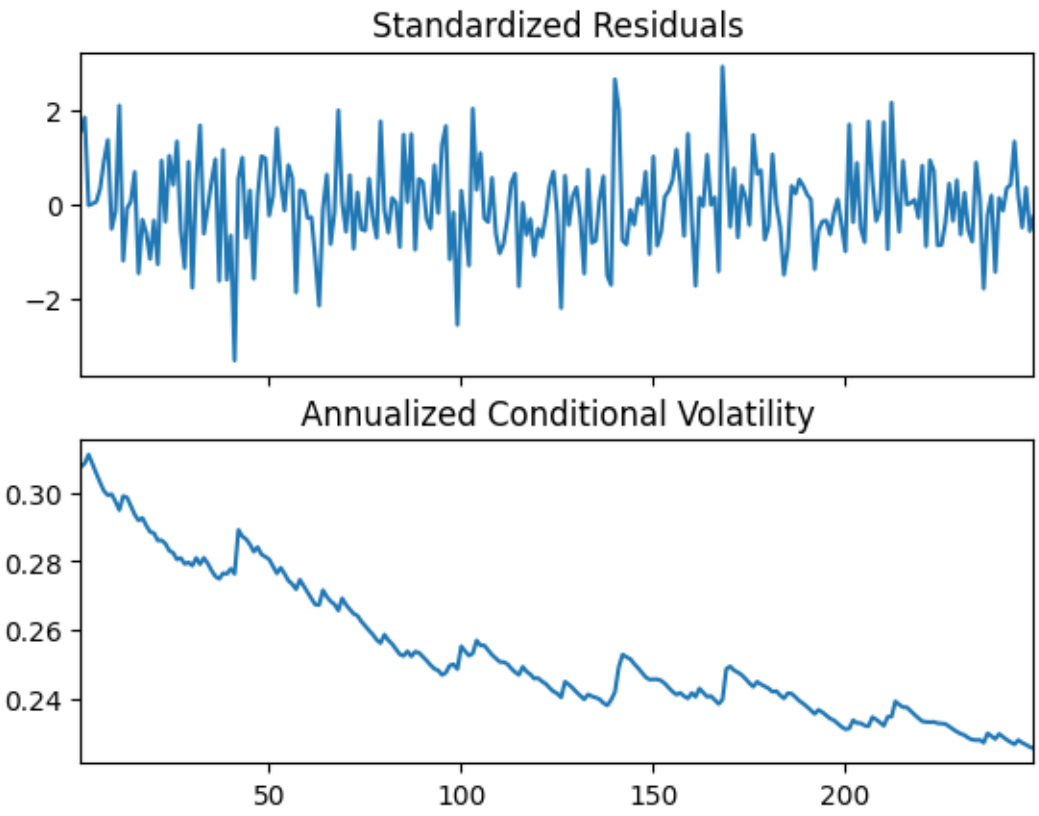
Estimate GARCH(1,1) model for Ethereum

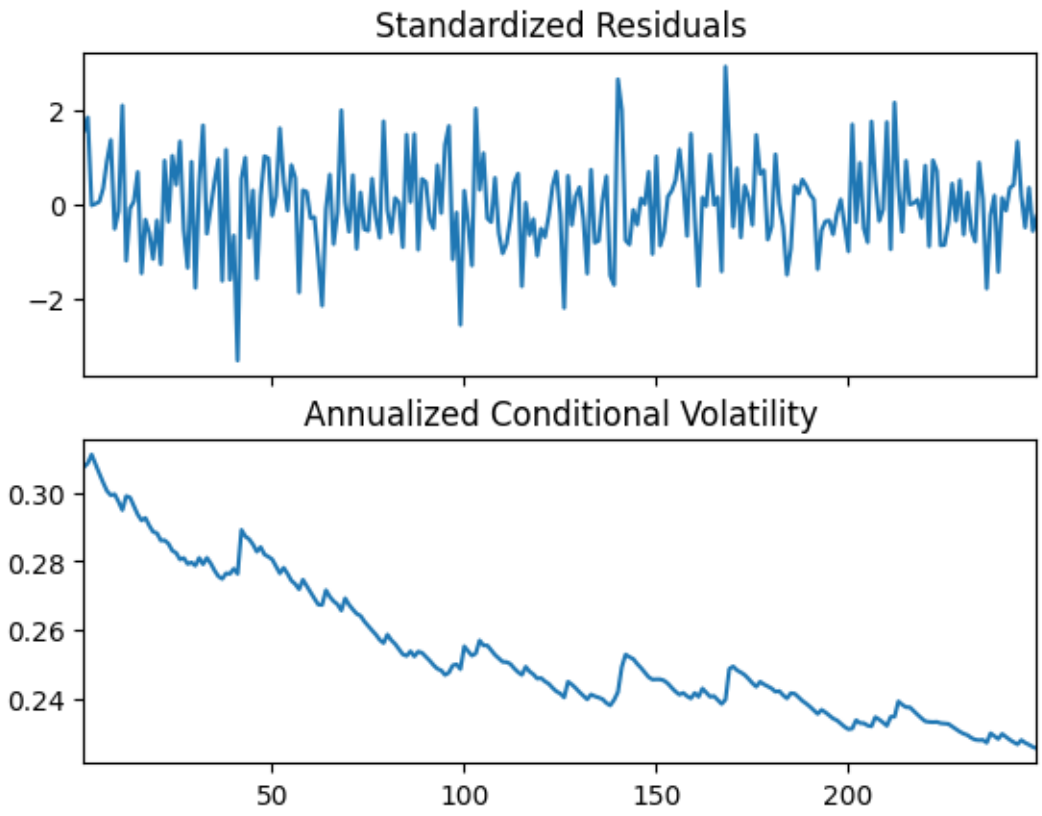
[45]:
```
model3 = arch_model(ETH_Log_Return, vol='GARCH', p=1, q=1, mean='constant',␣
↪dist='Normal')
results3 = model1.fit()

# Print summary of results for Etheruem
print(results3.summary())

results3.plot(annualize='D')
```

```
Iteration:      1,   Func. Count:      5,   Neg. LLF: -690.4656021817705
Optimization terminated successfully    (Exit mode 0)
            Current function value: -690.4656095058681
            Iterations: 5
```

```
            Function evaluations: 5
            Gradient evaluations: 1
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:           TM_Adj Close   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:                690.466
Distribution:                  Normal   AIC:                          -1372.93
Method:            Maximum Likelihood   BIC:                          -1358.86
                                        No. Observations:                  249
Date:                Wed, Mar 15 2023   Df Residuals:                      248
Time:                        03:00:54   Df Model:                            1
                                 Mean Model
==============================================================================
                 coef    std err          t      P>|t|        95.0% Conf. Int.
------------------------------------------------------------------------------
mu         -6.2220e-04  6.168e-05    -10.087  6.318e-24 [-7.431e-04,-5.013e-04]
                            Volatility Model
============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
----------------------------------------------------------------------------
omega      4.6878e-06  4.597e-11  1.020e+05      0.000  [4.688e-06,4.688e-06]
alpha[1]       0.0100  4.380e-02      0.228      0.819 [-7.585e-02,9.585e-02]
beta[1]        0.9700  3.659e-02     26.511 7.259e-155      [  0.898,  1.042]
============================================================================

Covariance estimator: robust
```

[45]:


Standardized Residuals
2
0
−2
Annualized Conditional Volatility
0.30
0.28
0.26
0.24
50
100
150
200



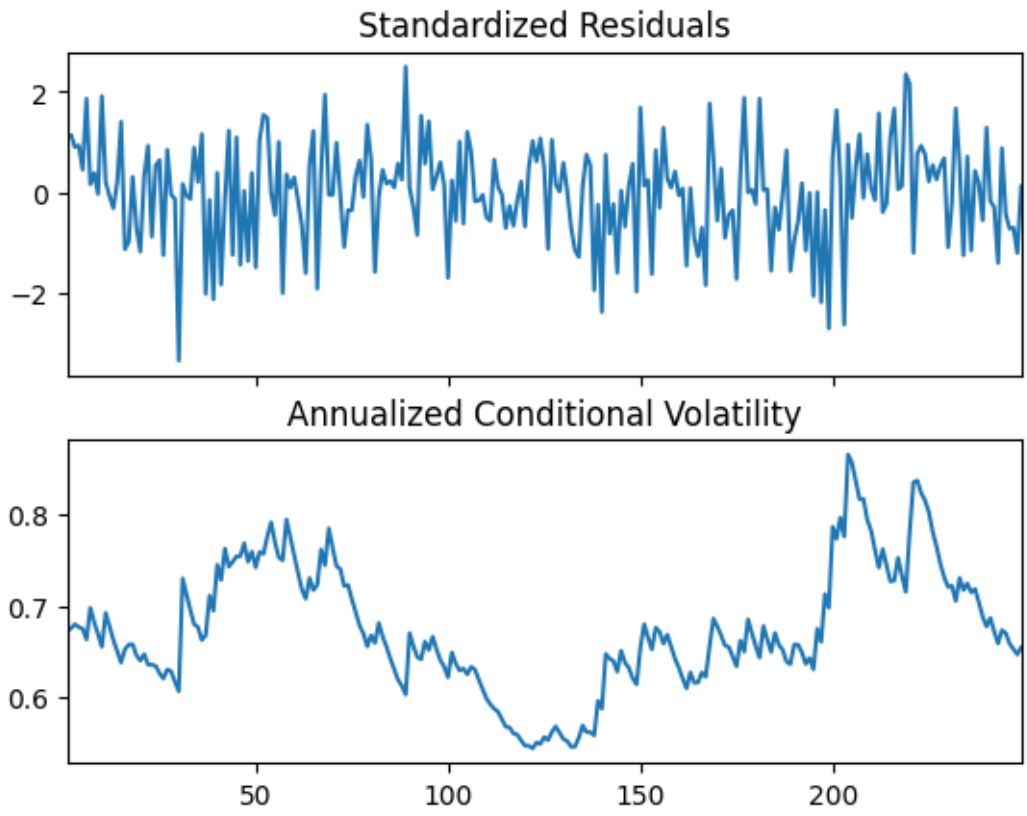
Estimate GARCH(1,1) model for Binance

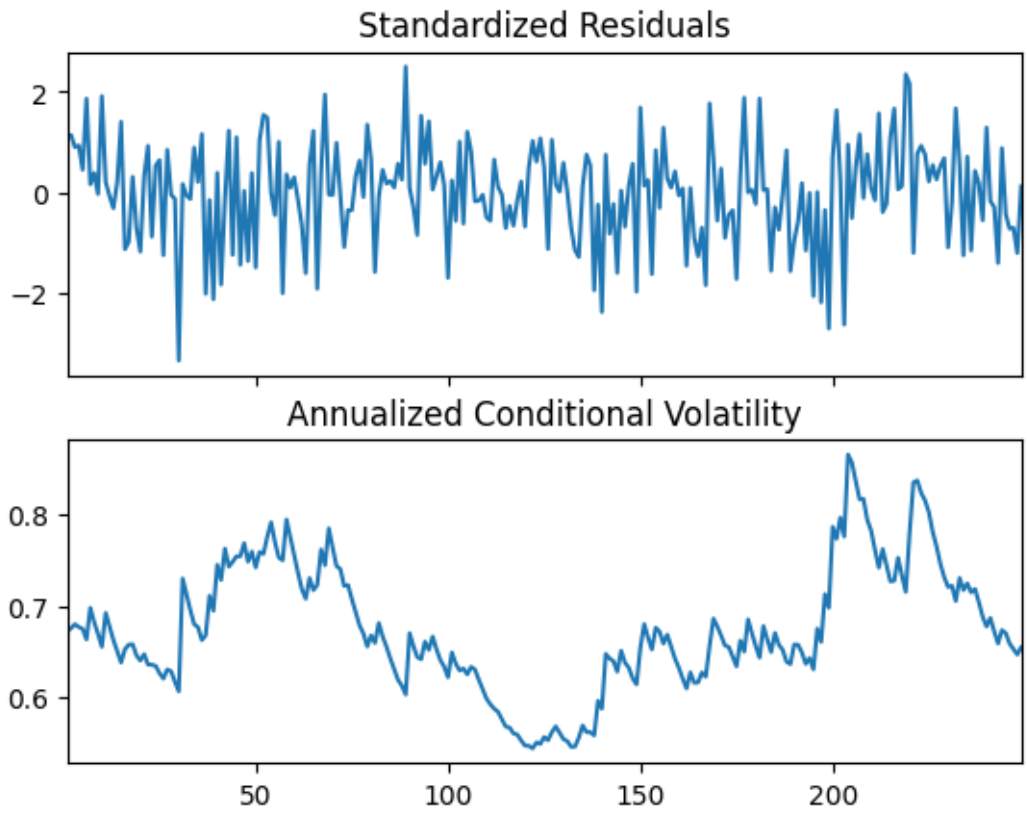
[46]:
```
model4 = arch_model(BNB_Log_Return, vol='GARCH', p=1, q=1, mean='constant',␣
 ↪dist='Normal')
results4 = model2.fit()

# Print summary of results for Binance
print(results4.summary())
results4.plot(annualize='D')
```

```
Iteration:      1,   Func. Count:      6,   Neg. LLF: 352215.57364146283
Iteration:      2,   Func. Count:     16,   Neg. LLF: 175.69319025718403
Iteration:      3,   Func. Count:     25,   Neg. LLF: -430.8950618822198
Iteration:      4,   Func. Count:     34,   Neg. LLF: -431.38888579686426
Iteration:      5,   Func. Count:     40,   Neg. LLF: -430.7512719893801
Iteration:      6,   Func. Count:     46,   Neg. LLF: -434.03373006917593
Iteration:      7,   Func. Count:     52,   Neg. LLF: -434.2428482449343
Iteration:      8,   Func. Count:     58,   Neg. LLF: -434.2533628294861
Iteration:      9,   Func. Count:     63,   Neg. LLF: -434.2533777787196
Iteration:     10,   Func. Count:     67,   Neg. LLF: -434.2533777787285
Optimization terminated successfully    (Exit mode 0)
```

```
            Current function value: -434.2533777787196
            Iterations: 10
            Function evaluations: 67
            Gradient evaluations: 10
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:         TSLA_Adj Close   R-squared:                        0.000
Mean Model:             Constant Mean   Adj. R-squared:                   0.000
Vol Model:                      GARCH   Log-Likelihood:                 434.253
Distribution:                  Normal   AIC:                           -860.507
Method:            Maximum Likelihood   BIC:                           -846.437
                                        No. Observations:                  249
Date:                Wed, Mar 15 2023   Df Residuals:                      248
Time:                        03:00:55   Df Model:                            1
                                Mean Model
==============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
------------------------------------------------------------------------------
mu         -1.7329e-03  2.568e-03     -0.675      0.500 [-6.767e-03,3.301e-03]
                           Volatility Model
==============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
------------------------------------------------------------------------------
omega      7.5148e-05  3.079e-05      2.441  1.466e-02  [1.480e-05,1.355e-04]
alpha[1]       0.0426  2.858e-02      1.490      0.136 [-1.342e-02,9.859e-02]
beta[1]        0.9160  2.107e-02     43.470      0.000      [  0.875,  0.957]
==============================================================================

Covariance estimator: robust
```

[46]:


Standardized Residuals
2
0
−2
Annualized Conditional Volatility
0.8
0.7
0.6
50
100
150
200



The mean model assumes a constant mean, and the output shows the estimated value of the intercept term (mu). For ETH, the estimated value of mu is -2.912, and for BNB, it is -1.083. However, for both cryptocurrencies, the p-values of the coefficient are greater than 0.05, suggesting that the estimated mean is not statistically significant. The p-value of omega is greater than 0.05, suggesting that it is not statistically significant. However, the p-values of alpha[1] and beta[1] are both less than 0.05, suggesting that they are statistically significant.