In [1]:

```
import pandas as pd
import numpy as np
import yfinance as yf
import arch

import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
tickers = ['CL=F', '^DJI', 'USDGBP=X']  # Replace these tickers with the ones you need
start_date = '2020-01-01'
end_date = '2023-01-01'

data = yf.download(tickers, start=start_date, end=end_date)['Adj Close']
```

```
[*********************100%***********************]  3 of 3 completed
```

In [3]:

```
returns = data.pct_change().dropna()
```

In [4]:

```
returns.tail()
```

Out[4]:

Date	CL=F	USDGBP=X	^DJI
2022-12-26	0.000000	-0.001665	0.000000
2022-12-27	-0.000377	-0.001998	0.001133
2022-12-28	-0.007167	0.004725	-0.011006
2022-12-29	-0.007092	-0.000096	0.010497
2022-12-30	0.023724	-0.002309	-0.002214

In [5]:

```
returns.info()
```

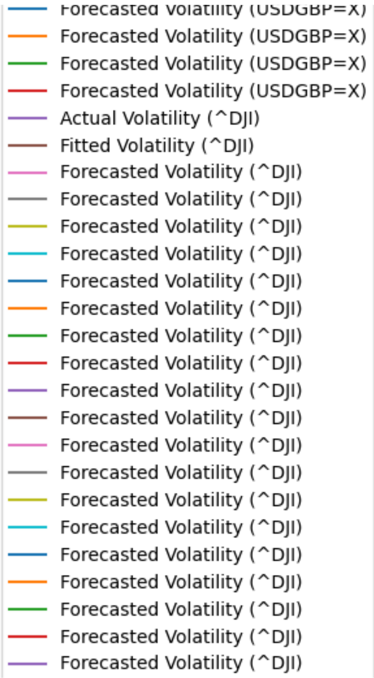
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 781 entries, 2020-01-03 to 2022-12-30
Freq: B
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   CL=F      781 non-null    float64
 1   USDGBP=X  781 non-null    float64
 2   ^DJI      781 non-null    float64
dtypes: float64(3)
memory usage: 24.4 KB
```

In [13]:

```
from statsmodels.stats.diagnostic import acorr_ljungbox
```

In [15]:

```
def check_arch_garch_effects(data):
    lb_test_results = pd.DataFrame()
    for column in data.columns:
        lb_test = acorr_ljungbox(data[column], lags=[10], return_df=True)
        lb_test_results[column] = lb_test['lb_stat']
    return lb_test_results

lb_test_results = check_arch_garch_effects(returns)
print(lb_test_results)

# Step 5: Fit an ARCH/GARCH model for each series.
def fit_arch_garch_model(data):
    models = []
    for column in data.columns:
        model = arch.arch_model(data[column], vol='Garch', p=1, q=1)
        model_fit = model.fit()
        models.append(model_fit)
    return models

arch_garch_models = fit_arch_garch_model(returns)

# Step 6: Forecast volatility for the next 3 months for each series.
forecast_horizon = 90  # 3 months * 30 days
forecast_variances = []
for model in arch_garch_models:
    forecasts = model.forecast(horizon=forecast_horizon, start=None)
    forecast_variance = forecasts.variance[-1:]
    forecast_variances.append(forecast_variance)

def plot_arch_garch_results(returns, arch_garch_models, forecast_variances):
    plt.figure(figsize=(10, 6))
    for i, column in enumerate(returns.columns):
        # Plot actual volatility
        actual_volatility = np.sqrt(arch_garch_models[i].conditional_volatility)
        plt.plot(returns.index, actual_volatility, label=f'Actual Volatility ({colum

        # Plot fitted volatility
        plt.plot(arch_garch_models[i].conditional_volatility, label=f'Fitted Volatil

        # Plot forecasted volatility
        forecast_index = pd.date_range(start=returns.index[-1], periods=len(forecast
        forecast_volatility = np.sqrt(forecast_variances[i])
        plt.plot(forecast_index, forecast_volatility, label=f'Forecasted Volatility

    plt.title('ARCH/GARCH Volatility')
    plt.xlabel('Date')
    plt.ylabel('Volatility')
    plt.legend()
    plt.show()

plot_arch_garch_results(returns, arch_garch_models, forecast_variances)
```



In [16]:

```
from statsmodels.tsa.vector_ar.var_model import VAR
from statsmodels.tsa.statespace.varmax import VARMAX
from statsmodels.tsa.vector_ar.vecm import VECM
```

In [17]:

```
data_for_var = returns.dropna()
```

In [18]:

```
def fit_var_model(data, lag_order):
    model = VAR(data)
    model_fit = model.fit(lag_order)
    return model_fit

lag_order = 2  # Replace with an appropriate lag order
var_model_fit = fit_var_model(data_for_var, lag_order)
print(var_model_fit.summary())
```

```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Sun, 30, Jul, 2023
Time:                     12:25:23
--------------------------------------------------------------------
No. of Equations:         3.00000    BIC:                   -22.8603
Nobs:                     779.000    HQIC:                  -22.9376
Log likelihood:           5657.95    FPE:                1.04074e-10
AIC:                     -22.9859    Det(Omega_mle):     1.01319e-10
--------------------------------------------------------------------
Results for equation CL=F
==========================================================================
====
                 coefficient       std. error           t-stat
prob
--------------------------------------------------------------------------
----
const              -0.002687         0.004140           -0.649
0.516
L1.CL=F             0.400343         0.035239           11.361
0.000
L1.USDGBP=X        -0.234090         0.692104           -0.338
0.735
L1.^DJI            -0.987221         0.276914           -3.565
0.000
L2.CL=F            -0.238229         0.035286           -6.751
0.000
L2.USDGBP=X        -0.867130         0.647767           -1.339
0.181
L2.^DJI             0.287236         0.296303            0.969
0.332
==========================================================================
====

Results for equation USDGBP=X
==========================================================================
====
                 coefficient       std. error           t-stat
prob
--------------------------------------------------------------------------
----
const               0.000196         0.000211            0.930
0.352
L1.CL=F            -0.001335         0.001794           -0.744
0.457
L1.USDGBP=X        -0.001124         0.035230           -0.032
0.975
L1.^DJI            -0.167607         0.014096          -11.891
0.000
L2.CL=F            -0.001253         0.001796           -0.697
0.485
L2.USDGBP=X        -0.019331         0.032973           -0.586
0.558
L2.^DJI            -0.085944         0.015083           -5.698
0.000
==========================================================================
====

Results for equation ^DJI
```

```
===========================================================================
====
                 coefficient       std. error           t-stat
prob
---------------------------------------------------------------------------
----
const               0.000349         0.000538            0.649
0.517
L1.CL=F             0.010682         0.004582            2.331
0.020
L1.USDGBP=X        -0.188323         0.089993           -2.093
0.036
L1.^DJI            -0.197089         0.036007           -5.474
0.000
L2.CL=F            -0.009073         0.004588           -1.977
0.048
L2.USDGBP=X         0.091368         0.084228            1.085
0.278
L2.^DJI             0.132997         0.038528            3.452
0.001
===========================================================================
====

Correlation matrix of residuals
                CL=F  USDGBP=X      ^DJI
CL=F        1.000000 -0.015737  0.122574
USDGBP=X   -0.015737  1.000000 -0.032794
^DJI        0.122574 -0.032794  1.000000
```

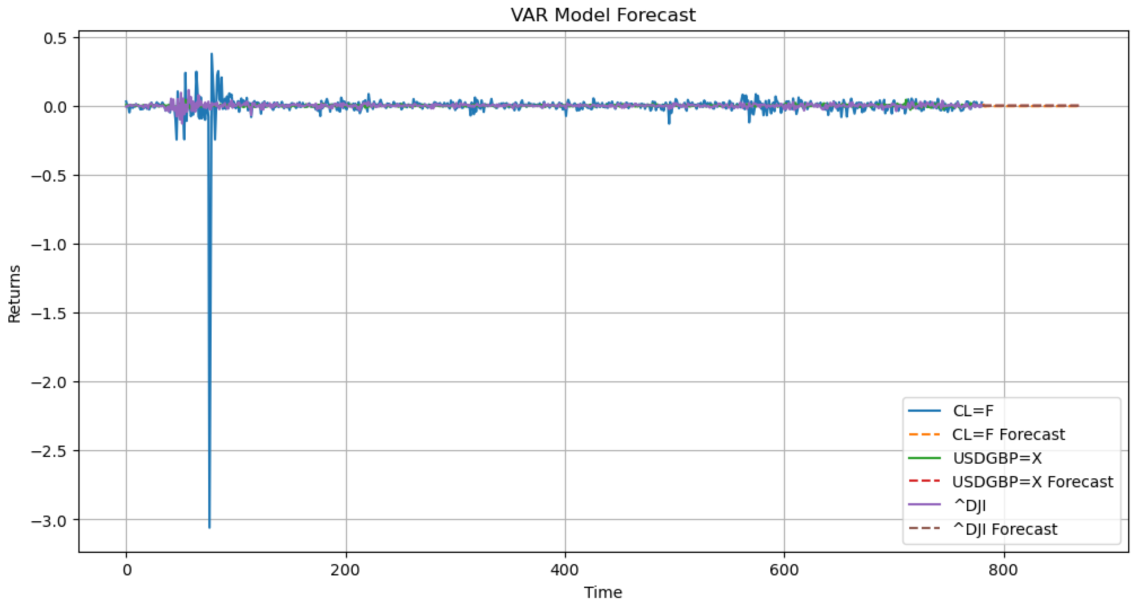
In [19]:

```python
def fit_vecm_model(data, r):
    model = VECM(data, k_ar_diff=1, coint_rank=r)
    model_fit = model.fit()
    return model_fit

coint_rank = 1  # Replace with an appropriate cointegration rank
vecm_model_fit = fit_vecm_model(data_for_var, coint_rank)
print(vecm_model_fit.summary())
```

```
========
                  coef    std err          z      P>|z|      [0.025
0.975]
-----------------------------------------------------------------------
--------
L1.CL=F         0.0059      0.002      3.745      0.000       0.003
0.009
L1.USDGBP=X    -0.1442      0.030     -4.800      0.000      -0.203
-0.085
L1.^DJI         0.1218      0.014      8.892      0.000       0.095
0.149
Det. terms outside the coint. relation & lagged endog. parameters for e
quation ^DJI
=======================================================================
========
                  coef    std err          z      P>|z|      [0.025
0.975]
-----------------------------------------------------------------------
--------
L1.CL=F         0.0222      0.004      5.188      0.000       0.014
```

In [23]:

```python
# Forecast for VAR
var_forecast = var_model_fit.forecast(data_for_var.values[-lag_order:], forecast_ste

# Plot the VAR forecast
plt.figure(figsize=(12, 6))
plt.title("VAR Model Forecast")
for i, col in enumerate(data_for_var.columns):
    plt.plot(np.arange(data_for_var.shape[0]), data_for_var[col], label=col)
    plt.plot(np.arange(data_for_var.shape[0], data_for_var.shape[0] + forecast_steps
             var_forecast[:, i], label=f"{col} Forecast", linestyle='dashed')

plt.xlabel("Time")
plt.ylabel("Returns")
plt.legend()
plt.grid(True)
plt.show()
```



In [ ]:

```python

```

In [ ]:

```python

```