In [1]:
```
!pip install pmdarima==2.0.3
!pip install --force-reinstall numpy==1.25.2
```

```
Collecting pmdarima==2.0.3
  Downloading pmdarima-2.0.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux
_2_28_x86_64.whl.metadata (7.8 kB)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.11/dist-packages (from
pmdarima==2.0.3) (1.4.2)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/local/lib/python3.11/
dist-packages (from pmdarima==2.0.3) (3.0.12)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.11/dist-packages (from
pmdarima==2.0.3) (2.0.2)
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.11/dist-packages (from
pmdarima==2.0.3) (2.2.2)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.11/dist-packages
(from pmdarima==2.0.3) (1.6.1)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.11/dist-packages (from
pmdarima==2.0.3) (1.15.2)
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python3.11/dist-packages
(from pmdarima==2.0.3) (0.14.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.11/dist-packages (from pmdar
ima==2.0.3) (2.4.0)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/python3.11/dist-
packages (from pmdarima==2.0.3) (75.2.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packa
ges (from pandas>=0.19->pmdarima==2.0.3) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from
pandas>=0.19->pmdarima==2.0.3) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (fro
m pandas>=0.19->pmdarima==2.0.3) (2025.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-package
s (from scikit-learn>=0.22->pmdarima==2.0.3) (3.6.0)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.11/dist-packages (from
statsmodels>=0.13.2->pmdarima==2.0.3) (1.0.1)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-packages (fr
om statsmodels>=0.13.2->pmdarima==2.0.3) (24.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from pyth
on-dateutil>=2.8.2->pandas>=0.19->pmdarima==2.0.3) (1.17.0)
Downloading pmdarima-2.0.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2
_28_x86_64.whl (1.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.9/1.9 MB 16.8 MB/s eta 0:00:00
Installing collected packages: pmdarima
Successfully installed pmdarima-2.0.3
Collecting numpy==1.25.2
  Downloading numpy-1.25.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadat
a (5.6 kB)
Downloading numpy-1.25.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (18.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.2/18.2 MB 85.0 MB/s eta 0:00:00
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 2.0.2
    Uninstalling numpy-2.0.2:
      Successfully uninstalled numpy-2.0.2
ERROR: pip's dependency resolver does not currently take into account all the packages that a
re installed. This behaviour is the source of the following dependency conflicts.
tensorflow 2.18.0 requires numpy<2.1.0,>=1.26.0, but you have numpy 1.25.2 which is incompati
ble.
thinc 8.3.6 requires numpy<3.0.0,>=2.0.0, but you have numpy 1.25.2 which is incompatible.
blosc2 3.3.2 requires numpy>=1.26, but you have numpy 1.25.2 which is incompatible.
Successfully installed numpy-1.25.2
```

Libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
import numpy as np
```

Loading Data and displaying few rows data

In [2]:
```
df = pd.read_csv('Non seasonal nvda data.csv', parse_dates=['Date'])

df.head(10)
```

Out[2]:

	Date	Open	High	Low	Close	Volume
0	2025-04-07	87.46	101.75	86.62	97.64	611041250
1	2025-04-04	98.91	100.13	92.11	94.31	532273812
2	2025-04-03	103.51	105.63	101.60	101.80	338769406
3	2025-04-02	107.29	111.98	106.79	110.42	220601203
4	2025-04-01	108.52	110.20	106.47	110.15	222614000
5	2025-03-31	105.13	110.96	103.65	108.38	299212719
6	2025-03-28	111.49	112.87	109.07	109.67	229872500
7	2025-03-27	111.35	114.45	110.66	111.43	236902094
8	2025-03-26	118.73	118.84	112.71	113.76	296431719
9	2025-03-25	120.55	121.29	118.92	120.69	167447203

Data cleaning

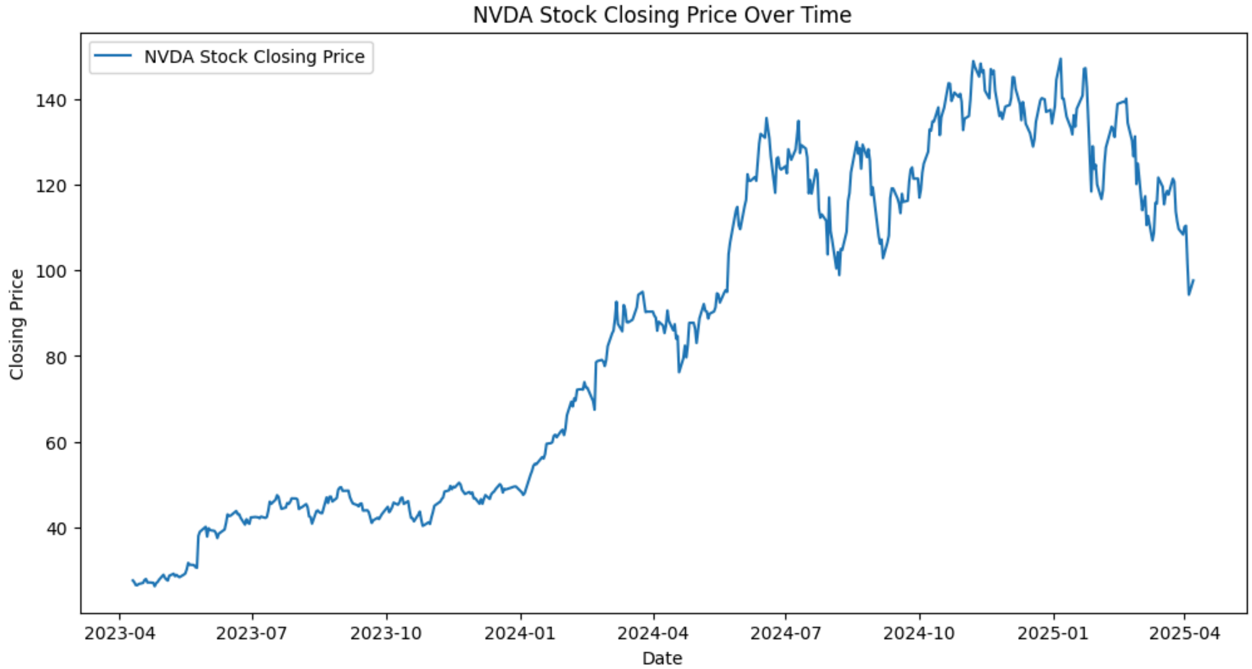
In [3]:
```
# Drop empty value rows
df = df.dropna()

# Drop duplicate rows
df = df.drop_duplicates()

# Convert date column
df['Date'] = pd.to_datetime(df['Date'])

# Check for Null values
print(df.isnull().sum())
```

```
Date      0
Open      0
High      0
Low       0
Close     0
Volume    0
dtype: int64
```

#Data Visualization

Closing Price visual reprentation

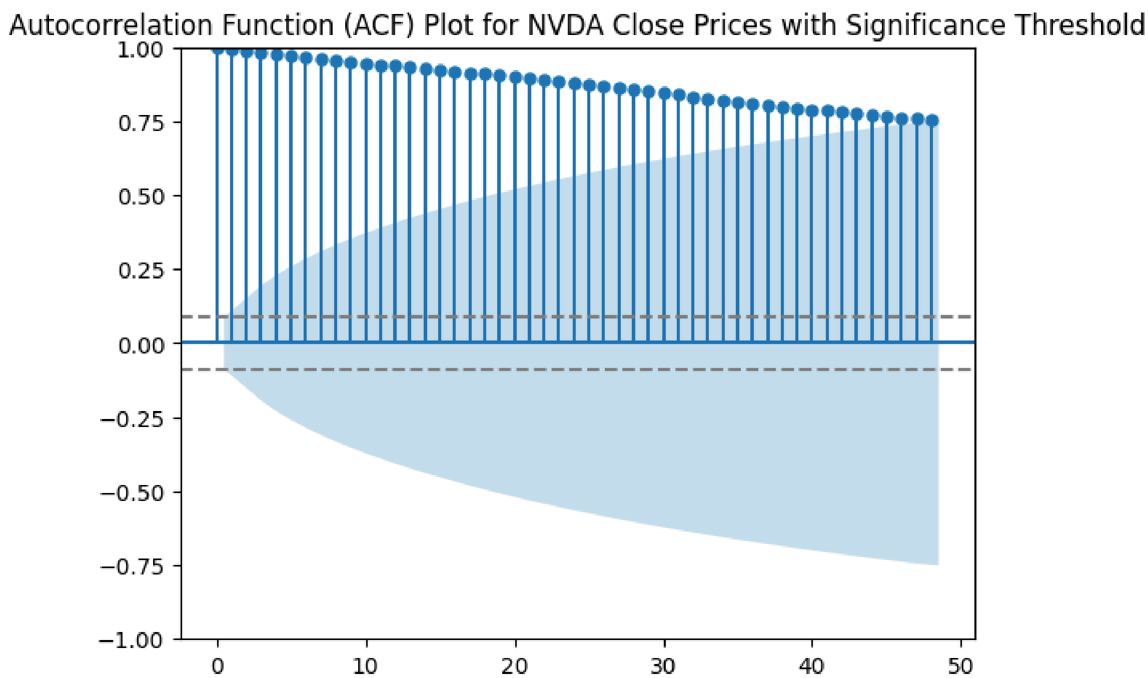
In [4]:
```
plt.figure(figsize=(12, 6))
plt.plot(df['Date'], df['Close'], label='NVDA Stock Closing Price')
plt.title('NVDA Stock Closing Price Over Time')
plt.xlabel('Date')
plt.ylabel('Closing Price')
plt.legend()
plt.show()
```



ACF plot

In [5]:
```
from statsmodels.graphics.tsaplots import plot_acf

plt.figure(figsize=(12, 6))
acf_plot_close = plot_acf(df['Close'].dropna(), lags=48, alpha=0.05)  # alpha sets confide

# Add a dotted line at the significance threshold
plt.axhline(y=-1.96/np.sqrt(len(df['Close'])), linestyle='--', color='gray')
plt.axhline(y=1.96/np.sqrt(len(df['Close'])), linestyle='--', color='gray')

plt.title('Autocorrelation Function (ACF) Plot for NVDA Close Prices with Significance Thr
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```



1. Strong Positive Autocorrelation at Short Lags: What happened yesterday with the stock price really influences what happens today. They're strongly linked.
2. Gradually Decreasing Autocorrelation: As the lag increases, the autocorrelation generally decreases. This suggests that the influence of past prices on the current price weakens over time.
3. Significant Autocorrelation Beyond the Threshold: For quite a while, even as you go back several days, there's still a noticeable pattern. This suggests that the time series is not completely random and that past prices have a predictive power for future prices.

#Augmented Dickey–Fuller test (Stationary test) The ADF test is a statistical test used to determine if a time series is stationary or not.

In [6]:
```
result_original = adfuller(df['Close'])
print('ADF Statistic (Original):', result_original[0])
print('p-value (Original):', result_original[1])
print('Critical Values (Original):', result_original[4])
```

```
ADF Statistic (Original): -0.2988577938344866
p-value (Original): 0.9256659470708057
Critical Values (Original): {'1%': -3.4435761493506294, '5%': -2.867372960189225, '10%': -2.5
698767442886696}
```

1. We can see that P- value is significantly greater than 0.05.
2. The ADF Statistic (-0.2988577938344866) is greater than all the critical values provided.
3. So this is non-stationary therefore we need to make it stationary by differencing.

Differencing: It helps to stabilize the mean of a time series by removing trends and seasonality.

In [7]:
```
#Differenced Once
df['Close'] = df['Close'].diff()
```

In [8]:
```
df.head()
```

Out[8]:

	Date	Open	High	Low	Close	Volume
0	2025-04-07	87.46	101.75	86.62	NaN	611041250
1	2025-04-04	98.91	100.13	92.11	-3.33	532273812
2	2025-04-03	103.51	105.63	101.60	7.49	338769406
3	2025-04-02	107.29	111.98	106.79	8.62	220601203
4	2025-04-01	108.52	110.20	106.47	-0.27	222614000

After Differencing we got some NaN value in our column so we are filling it.

In [9]:
```
df['Close'].fillna(method='bfill', inplace=True)
```

```
<ipython-input-9-7170cc8430d6>:1: FutureWarning: A value is trying to be set on a copy of a D
ataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the inter
mediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: val
ue}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inpl
ace on the original object.


  df['Close'].fillna(method='bfill', inplace=True)
<ipython-input-9-7170cc8430d6>:1: FutureWarning: Series.fillna with 'method' is deprecated an
d will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  df['Close'].fillna(method='bfill', inplace=True)
```

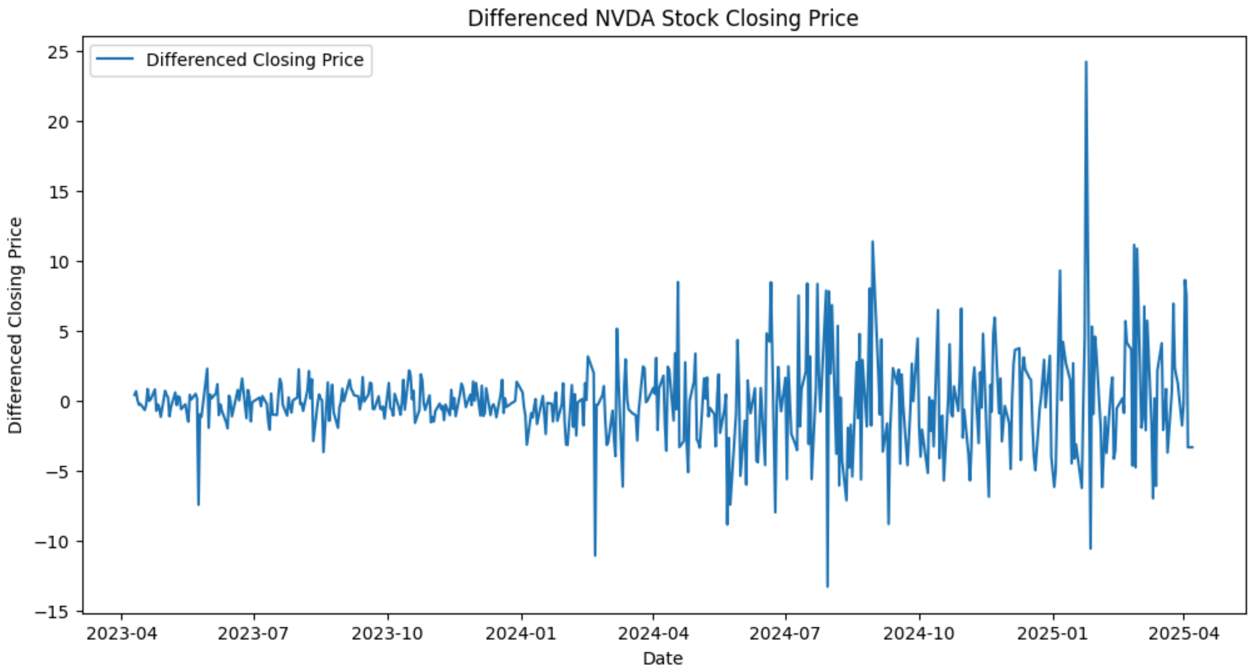
In [10]:
```
df.head()
```

Out[10]:

	Date	Open	High	Low	Close	Volume
0	2025-04-07	87.46	101.75	86.62	-3.33	611041250
1	2025-04-04	98.91	100.13	92.11	-3.33	532273812
2	2025-04-03	103.51	105.63	101.60	7.49	338769406
3	2025-04-02	107.29	111.98	106.79	8.62	220601203
4	2025-04-01	108.52	110.20	106.47	-0.27	222614000

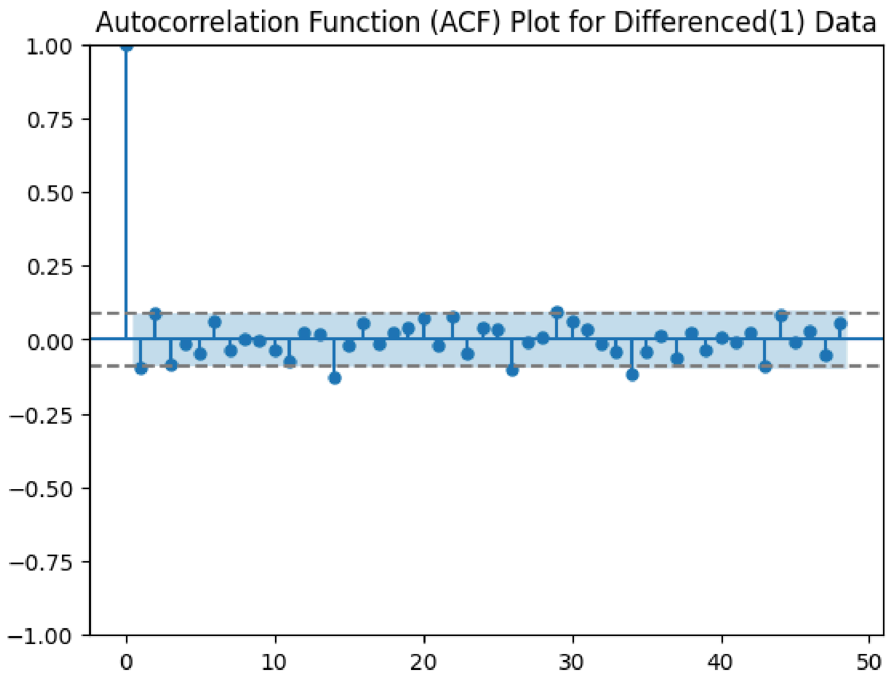
In [11]:

```
# Plotting differenced closing prices
plt.figure(figsize=(12, 6))
plt.plot(df['Date'], df['Close'], label='Differenced Closing Price')
plt.title('Differenced NVDA Stock Closing Price')
plt.xlabel('Date')
plt.ylabel('Differenced Closing Price')
plt.legend()
plt.show()
```



In [12]:
```
# ACF Plot for 'Close'
plt.figure(figsize=(12, 6))
acf_plot_close = plot_acf(df['Close'].dropna(), lags=48, alpha=0.05)  # alpha sets confide

# Add a dotted line at the significance threshold
plt.axhline(y=-1.96/np.sqrt(len(df['Close'])), linestyle='--', color='gray')
plt.axhline(y=1.96/np.sqrt(len(df['Close'])), linestyle='--', color='gray')

plt.title('Autocorrelation Function (ACF) Plot for Differenced(1) Data')
plt.show()
```

<Figure size 1200x600 with 0 Axes>

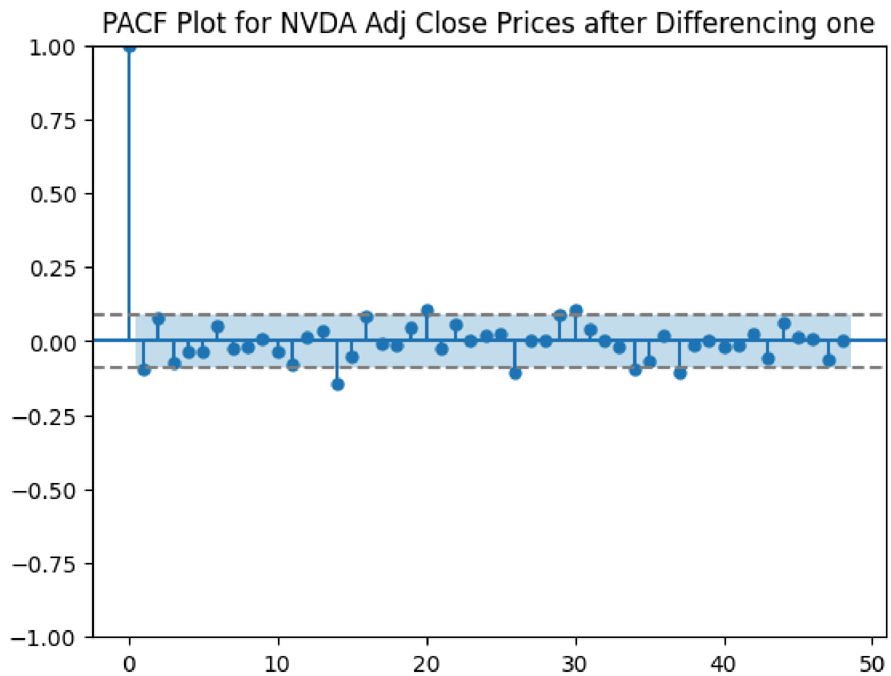


PACF PLOT

p is the order of Autoregressive (AR). From PACF plot we can find the value of p. q is the order of Moving Average (MA). From ACF plot we can find the value of q

In [13]:
```
from statsmodels.graphics.tsaplots import plot_pacf

# PACF plot for 'Close'
plt.figure(figsize=(12, 6))
pacf_plot_close = plot_pacf(df['Close'].dropna(), lags=48, alpha=0.05)  # Adjust 'lags' an

# Add a dotted line at the significance threshold
plt.axhline(y=-1.96/np.sqrt(len(df['Close'])), linestyle='--', color='gray')
plt.axhline(y=1.96/np.sqrt(len(df['Close'])), linestyle='--', color='gray')

plt.title('PACF Plot for NVDA Adj Close Prices after Differencing one')
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```



In [14]:
```
from pmdarima.arima.utils import ndiffs
```

In [15]:
```
ndiffs(df['Close'], test='adf')
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'for
ce_all_finite' was renamed to 'ensure_all_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
```

Out[15]:
```
0
```

Fitting the ARIMA Model

In [16]:
```
pip install --upgrade statsmodels
```

```
Requirement already satisfied: statsmodels in /usr/local/lib/python3.11/dist-packages (0.14.
4)
Requirement already satisfied: numpy<3,>=1.22.3 in /usr/local/lib/python3.11/dist-packages (f
rom statsmodels) (1.25.2)
Requirement already satisfied: scipy!=1.9.2,>=1.8 in /usr/local/lib/python3.11/dist-packages
(from statsmodels) (1.15.2)
Requirement already satisfied: pandas!=2.1.0,>=1.4 in /usr/local/lib/python3.11/dist-packages
(from statsmodels) (2.2.2)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.11/dist-packages (from
statsmodels) (1.0.1)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-packages (fr
om statsmodels) (24.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packa
ges (from pandas!=2.1.0,>=1.4->statsmodels) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from
pandas!=2.1.0,>=1.4->statsmodels) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (fro
m pandas!=2.1.0,>=1.4->statsmodels) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from pyth
on-dateutil>=2.8.2->pandas!=2.1.0,>=1.4->statsmodels) (1.17.0)
```

Model Selection

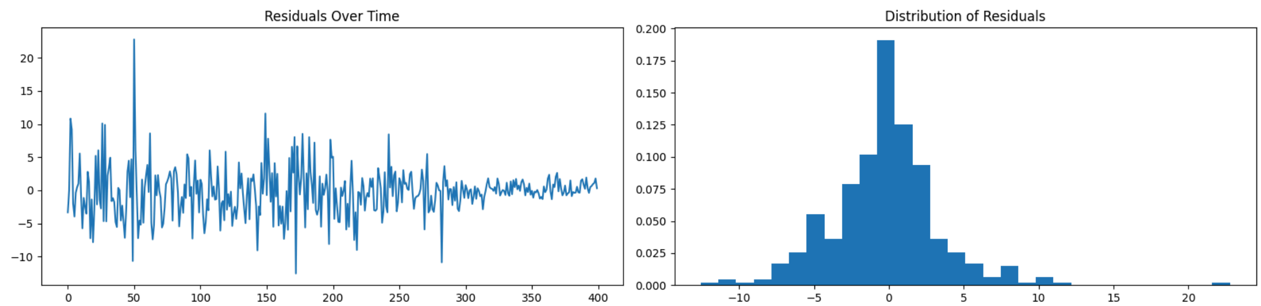
In [17]:
```
from statsmodels.tsa.arima.model import ARIMA

# ARIMA Model
model = ARIMA(df['Close'], order=(1, 1, 1))
result = model.fit()

# Print the summary
print(result.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                  501
Model:                 ARIMA(1, 1, 1)   Log Likelihood               -1299.373
Date:                Mon, 05 May 2025   AIC                           2604.745
Time:                        21:13:20   BIC                           2617.389
Sample:                             0   HQIC                          2609.707
                                - 501
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.0950      0.032     -2.922      0.003      -0.159      -0.031
ma.L1         -0.9946      0.011    -93.239      0.000      -1.016      -0.974
sigma2        10.4875      0.403     25.996      0.000       9.697      11.278
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):               818.61
Prob(Q):                              0.90   Prob(JB):                         0.00
Heteroskedasticity (H):               0.07   Skew:                             0.65
Prob(H) (two-sided):                  0.00   Kurtosis:                         9.13
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

In [18]:
```
from pmdarima import auto_arima

model = auto_arima(df['Close'], seasonal=False,
                      start_p=0, start_q=0, max_p=5, max_q=5,
                      d=1,
                      trace=True, error_action='ignore', suppress_warnings=True, stepwise=
print(model.summary())
```

```
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.8998      0.033    -27.043      0.000      -0.965      -0.835
ar.L2         -0.6339      0.046    -13.633      0.000      -0.725      -0.543
ar.L3         -0.5334      0.046    -11.583      0.000      -0.624      -0.443
ar.L4         -0.4034      0.048     -8.346      0.000      -0.498      -0.309
ar.L5         -0.2396      0.035     -6.824      0.000      -0.308      -0.171
sigma2        12.0387      0.478     25.211      0.000      11.103      12.975
===================================================================================
Ljung-Box (L1) (Q):                   0.40   Jarque-Bera (JB):               613.08
Prob(Q):                              0.53   Prob(JB):                         0.00
Heteroskedasticity (H):               0.07   Skew:                             0.55
Prob(H) (two-sided):                  0.00   Kurtosis:                         8.31
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

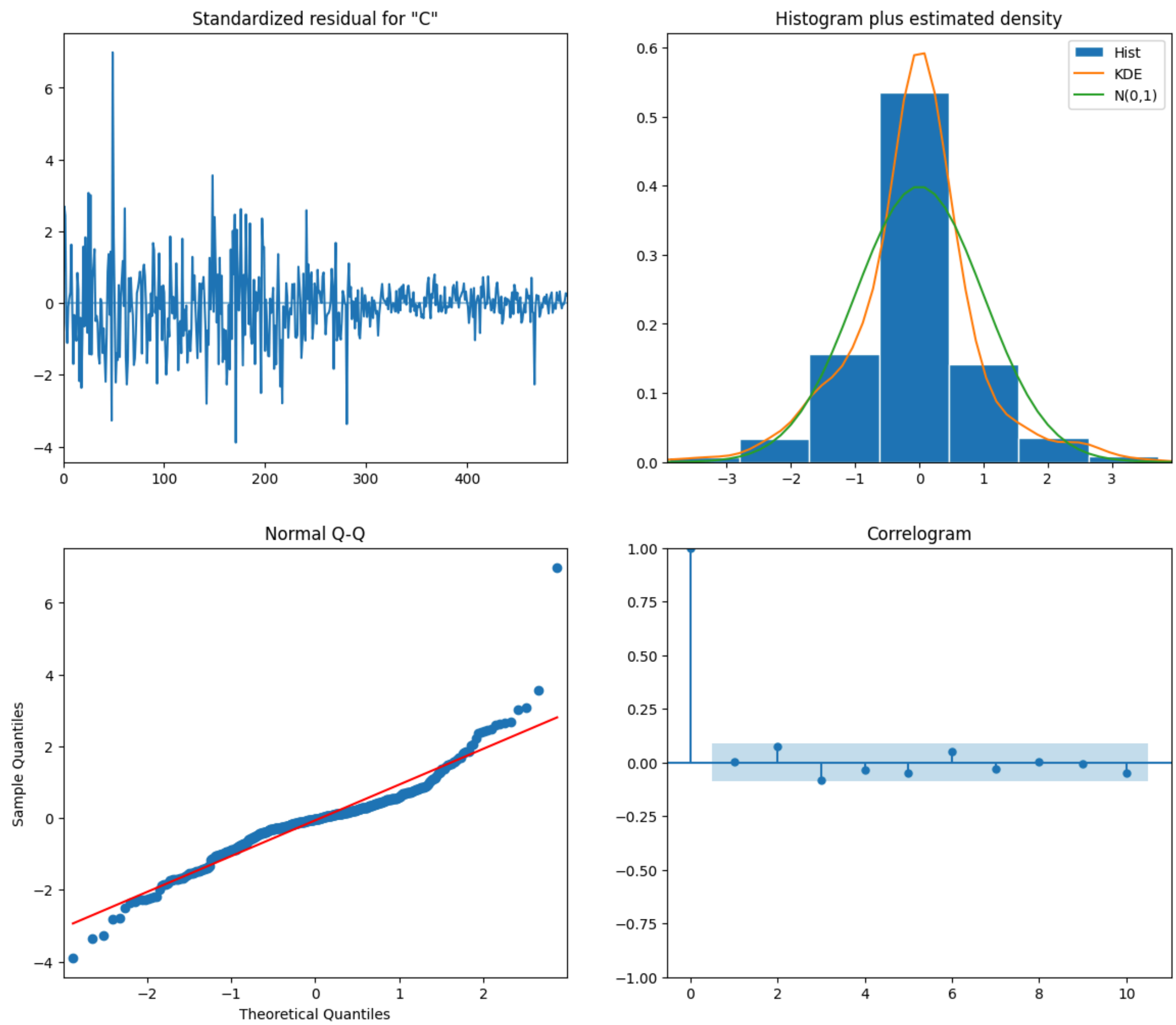
In [54]:
```
# Plot Residual Errors
residuals = pd.DataFrame(result.resid)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 4))

ax1.plot(residuals)
ax1.set_title("Residuals Over Time")

ax2.hist(residuals, density=True, bins=30)
ax2.set_title("Distribution of Residuals")

plt.tight_layout()
plt.show()

```



In [50]:
```python
import pandas as pd
from statsmodels.stats.diagnostic import acorr_ljungbox

data = df['Close']

lb_test = acorr_ljungbox(data, lags=20, return_df=True)

print(lb_test)
```

```
      lb_stat  lb_pvalue
1    4.512393   0.033650
2    8.500464   0.014261
3   12.273141   0.006504
4   12.342104   0.014981
5   13.367349   0.020169
6   15.331503   0.017829
7   15.981075   0.025290
8   15.986613   0.042572
9   15.987730   0.067138
10  16.630124   0.082960
11  19.345174   0.055172
12  19.684989   0.073285
13  19.897743   0.097790
14  28.117391   0.013726
15  28.265553   0.019965
16  29.993440   0.018036
17  30.074825   0.025810
18  30.401521   0.033721
19  31.323128   0.037195
20  34.233329   0.024589
```

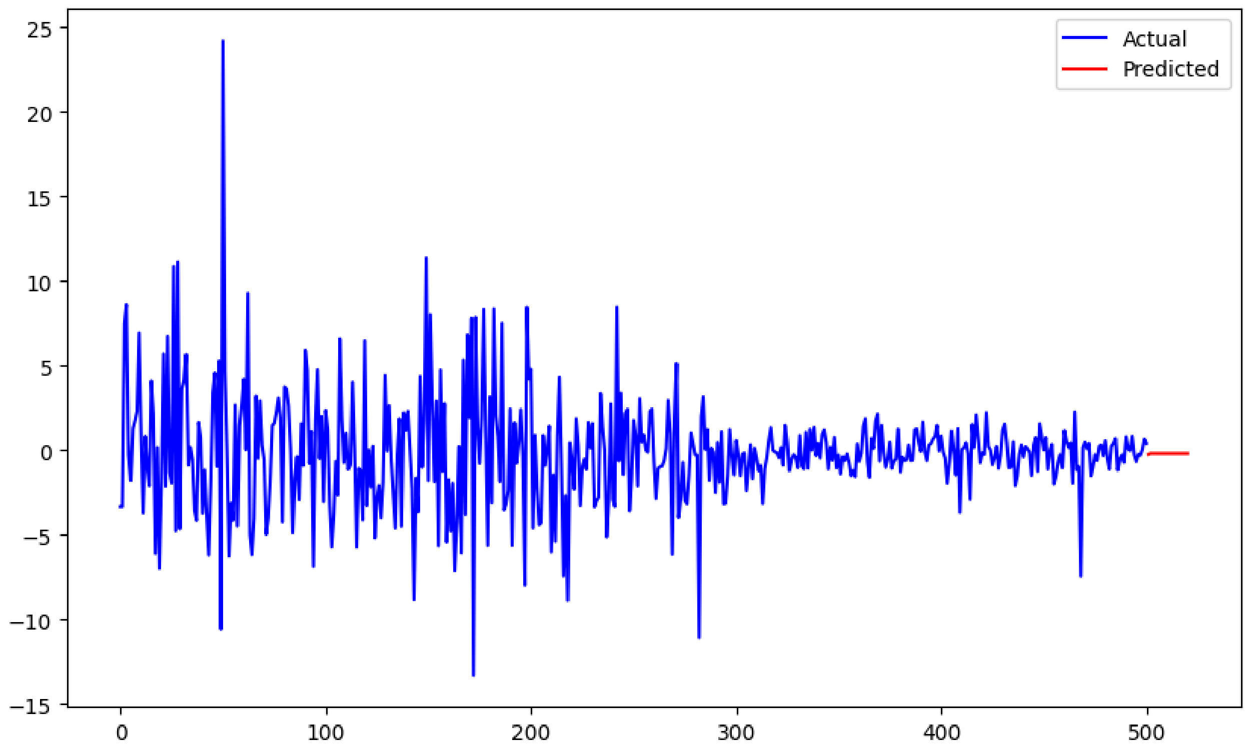
In [22]:
```
result.plot_diagnostics(figsize=(14,12))
plt.show()
```



In [51]:
```
from scipy.stats import shapiro
import numpy as np

# Example dataset
data = residuals.copy()

# Run Shapiro-Wilk test
stat, p = shapiro(data)

# Print results
print("Shapiro-Wilk Test Statistic:", stat)
print("p-value:", p)

if p > 0.05:
    print("Data appears to be normally distributed (fail to reject H0).")
else:
    print("Data does not appear to be normally distributed (reject H0).")

```

```
Shapiro-Wilk Test Statistic: 0.9219184111044073
p-value: 1.8984881534724404e-15
Data does not appear to be normally distributed (reject H0).
```

In [34]:
```
from statsmodels.tsa.arima.model import ARIMA

# ARIMA Model
model = ARIMA(df['Close'], order=(1, 1, 1))

# Fit the model
result = model.fit()

# Get forecast
forecast = result.forecast(steps=20)

# Plot Actual vs Predicted
fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(df['Close'], label='Actual', color='blue')
ax.plot(range(len(df), len(df) + len(forecast)), forecast, label='Predicted', color='red')
ax.legend()

plt.show()

# Print the summary
print(result.summary())
```



```
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                  501
Model:                 ARIMA(1, 1, 1)   Log Likelihood               -1299.373
Date:                Mon, 05 May 2025   AIC                           2604.745
Time:                        21:16:19   BIC                           2617.389
Sample:                             0   HQIC                          2609.707
                                - 501
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.0950      0.032     -2.922      0.003      -0.159      -0.031
ma.L1         -0.9946      0.011    -93.239      0.000      -1.016      -0.974
sigma2        10.4875      0.403     25.996      0.000       9.697      11.278
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):               818.61
Prob(Q):                              0.90   Prob(JB):                         0.00
Heteroskedasticity (H):               0.07   Skew:                             0.65
Prob(H) (two-sided):                  0.00   Kurtosis:                         9.13
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

In [35]:

```python
df2= df[['Close']].copy()
df2.describe()
```

Out[35]:

	Close
count	501.000000
mean	-0.146487
std	3.247951
min	-13.290000
25%	-1.490000
50%	-0.220000
75%	1.190000
max	24.200000

Train and Test

In [36]:

```python
from sklearn.metrics import mean_squared_error

n = int(len(df2) * 0.8)
train = df2['Close'][:n]
test = df2['Close'][n:]
```

In [37]:

```python
print(len(train))
print(len(test))
```

```
400
101
```

In [38]:

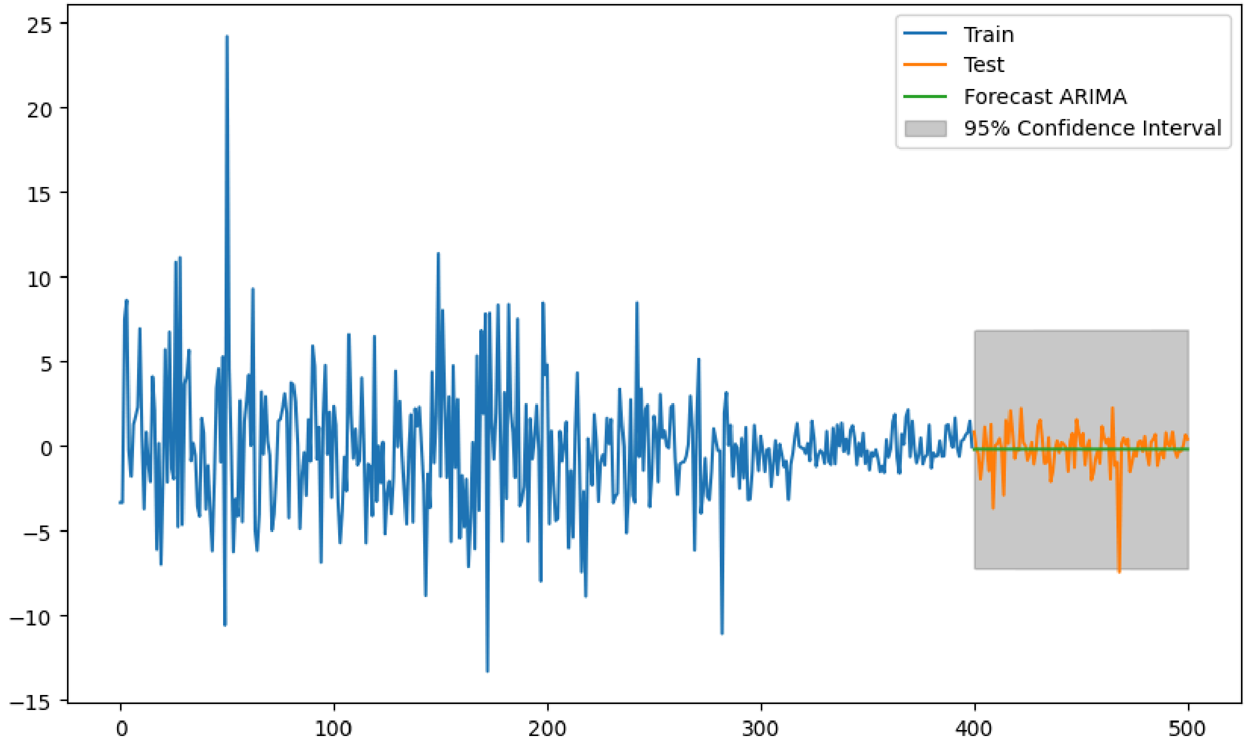
```python
model = ARIMA(train, order= (1,1,1))
result= model.fit()
```

In [39]:

```
1 print(result.summary())
```

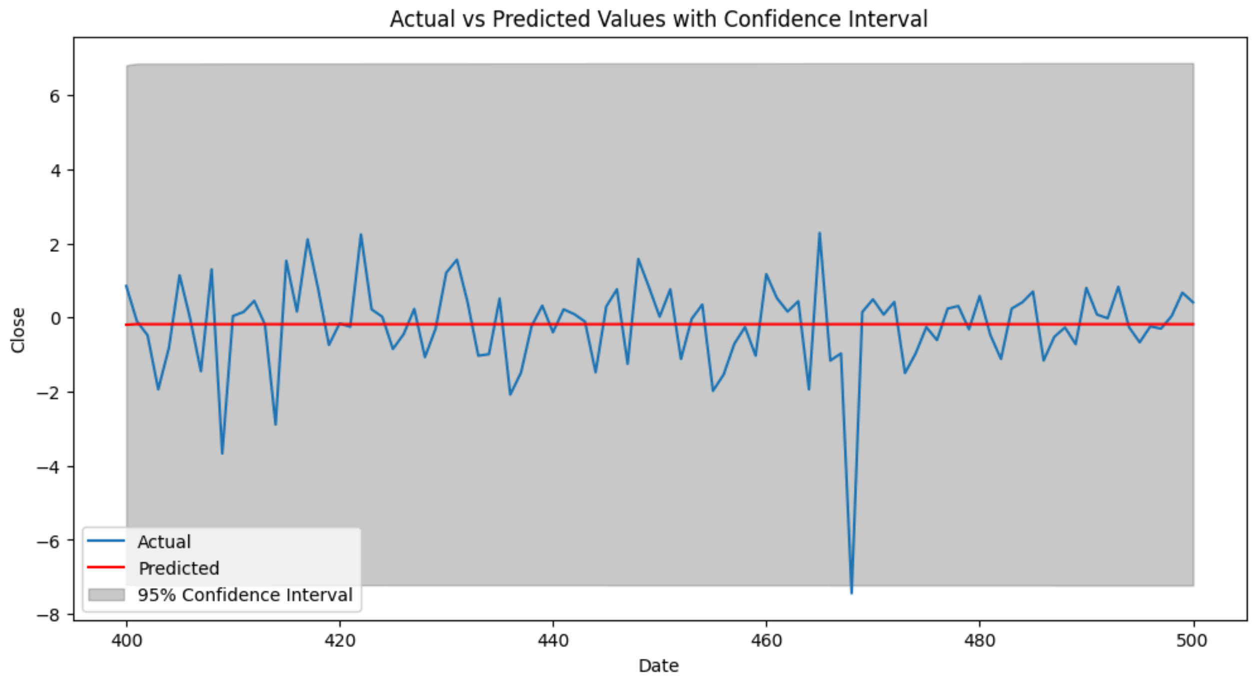
```
                               SARIMAX Results                                
==============================================================================
Dep. Variable:                  Close   No. Observations:                  400
Model:                 ARIMA(1, 1, 1)   Log Likelihood               -1076.355
Date:                Mon, 05 May 2025   AIC                           2158.711
Time:                        21:16:31   BIC                           2170.678
Sample:                             0   HQIC                          2163.450
                                - 400                                         
Covariance Type:                  opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.0976      0.040     -2.461      0.014      -0.175      -0.020
ma.L1         -0.9926      0.012    -84.473      0.000      -1.016      -0.970
sigma2        12.7604      0.610     20.934      0.000      11.566      13.955
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):               394.68
Prob(Q):                              0.91   Prob(JB):                         0.00
Heteroskedasticity (H):               0.15   Skew:                             0.66
Prob(H) (two-sided):                  0.00   Kurtosis:                         7.69
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

In [42]:
```
# Forecast using the trained model
forecast_result = result.get_forecast(steps=len(test))
forecast_mean = forecast_result.predicted_mean
confidence_interval = forecast_result.conf_int()

# Plotting
train.plot(legend=True, label='Train', figsize=(10, 6))
test.plot(legend=True, label='Test')
forecast_mean.plot(legend=True, label='Forecast ARIMA')

# Plot confidence interval
plt.fill_between(confidence_interval.index, confidence_interval.iloc[:, 0], confidence_int

plt.legend()
plt.show()
```



In [43]:

```
# Forecast using the trained model
forecast_result = result.get_forecast(steps=len(test))
forecast_mean = forecast_result.predicted_mean
confidence_interval = forecast_result.conf_int()

# Plotting actual vs predicted values
plt.figure(figsize=(12, 6))
plt.plot(test.index, test, label='Actual')
plt.plot(forecast_mean.index, forecast_mean, color='red', label='Predicted')
plt.fill_between(confidence_interval.index, confidence_interval.iloc[:, 0], confidence_int
plt.title('Actual vs Predicted Values with Confidence Interval')
plt.xlabel('Date')
plt.ylabel('Close')
plt.legend()
plt.show()
```



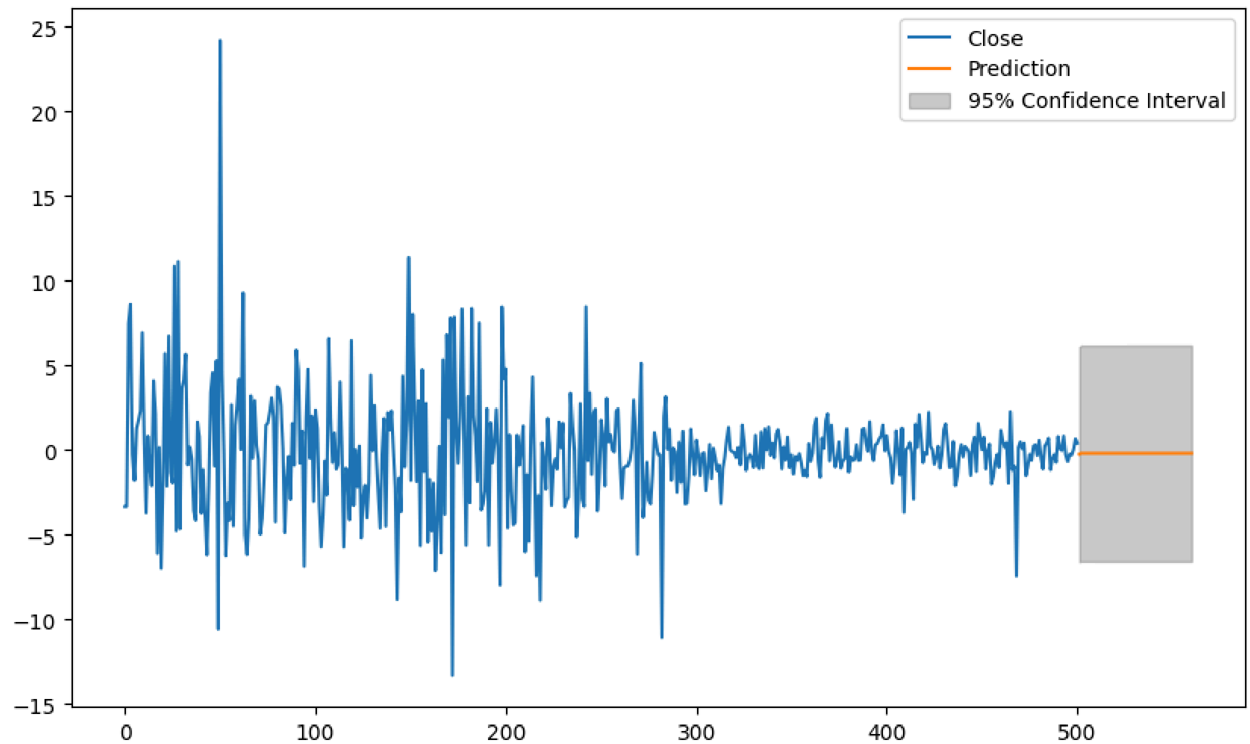
Predict Future Data

In [44]:

```
final_model= ARIMA(df2, order=(1,1,1)).fit()

predication= final_model.predict(len(df2),len(df2)+60)  #predicting for next 60 trading da
```

In [45]:

```
# Make predictions for the next 60 trading days
forecast_result = final_model.get_forecast(steps=60)
forecast_mean = forecast_result.predicted_mean
confidence_interval = forecast_result.conf_int()

# Plotting
df2.plot(legend=True, label='Train', figsize=(10, 6))
forecast_mean.plot(legend=True, label='Prediction')

# Plot confidence interval
plt.fill_between(confidence_interval.index, confidence_interval.iloc[:, 0], confidence_int

plt.legend()
plt.show()
```

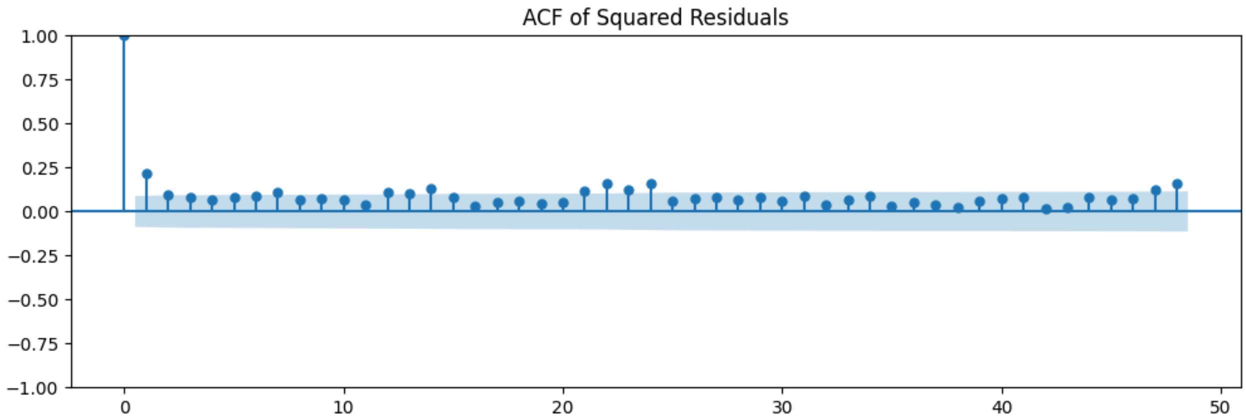


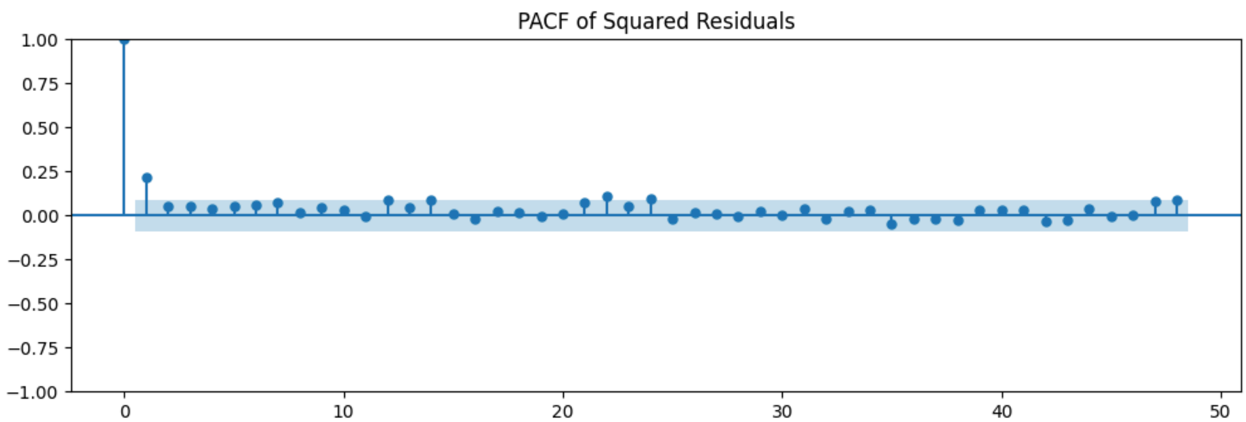
Applying ARMA + GARCH model

In [46]:
```
pip install arch
```

```
Collecting arch
  Downloading arch-7.2.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(13 kB)
Requirement already satisfied: numpy>=1.22.3 in /usr/local/lib/python3.11/dist-packages (from
arch) (1.25.2)
Requirement already satisfied: scipy>=1.8 in /usr/local/lib/python3.11/dist-packages (from ar
ch) (1.15.2)
Requirement already satisfied: pandas>=1.4 in /usr/local/lib/python3.11/dist-packages (from a
rch) (2.2.2)
Requirement already satisfied: statsmodels>=0.12 in /usr/local/lib/python3.11/dist-packages
(from arch) (0.14.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packa
ges (from pandas>=1.4->arch) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from
pandas>=1.4->arch) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (fro
m pandas>=1.4->arch) (2025.2)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.11/dist-packages (from
statsmodels>=0.12->arch) (1.0.1)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-packages (fr
om statsmodels>=0.12->arch) (24.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from pyth
on-dateutil>=2.8.2->pandas>=1.4->arch) (1.17.0)
Downloading arch-7.2.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (985 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 985.3/985.3 kB 10.3 MB/s eta 0:00:00
Installing collected packages: arch
Successfully installed arch-7.2.0
```

In [55]:
```
# Calculate residuals
residuals = final_model.resid

# Square the residuals
squared_residuals = residuals ** 2

# Plot ACF and PACF of squared residuals
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))

# ACF plot
plot_acf(squared_residuals, ax=ax1, lags=48)
ax1.set_title('ACF of Squared Residuals')

# PACF plot
plot_pacf(squared_residuals, ax=ax2, lags=48)
ax2.set_title('PACF of Squared Residuals')

plt.show()
```





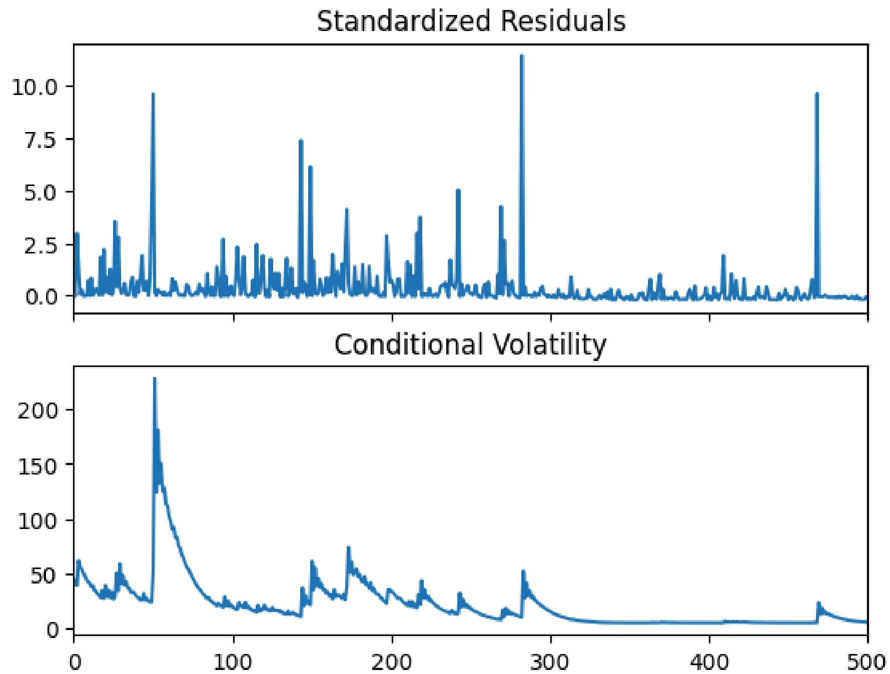
In [48]:

```python
import arch

# Fit GARCH model
garch_model = arch.arch_model(squared_residuals, vol='Garch', p=1, q=2)
garch_result = garch_model.fit()

# Display the model summary
print(garch_result.summary())
```

```
Iteration:      1,   Func. Count:      7,   Neg. LLF: 4407.444540132237
Iteration:      2,   Func. Count:     14,   Neg. LLF: 3060.4798863927226
Iteration:      3,   Func. Count:     22,   Neg. LLF: 2651.2556633005424
Iteration:      4,   Func. Count:     29,   Neg. LLF: 2270.2972455413255
Iteration:      5,   Func. Count:     36,   Neg. LLF: 2179.308537785706
Iteration:      6,   Func. Count:     42,   Neg. LLF: 2188.481812168503
Iteration:      7,   Func. Count:     49,   Neg. LLF: 2177.3174160922003
Iteration:      8,   Func. Count:     55,   Neg. LLF: 2176.624413878339
Iteration:      9,   Func. Count:     61,   Neg. LLF: 2176.1501409323137
Iteration:     10,   Func. Count:     67,   Neg. LLF: 2174.96241701514
Iteration:     11,   Func. Count:     73,   Neg. LLF: 2372.532333221814
Iteration:     12,   Func. Count:     80,   Neg. LLF: 2394.324020554337
Iteration:     13,   Func. Count:     87,   Neg. LLF: 2278.083498148742
Iteration:     14,   Func. Count:     94,   Neg. LLF: 2419.9216926077443
Iteration:     15,   Func. Count:    101,   Neg. LLF: 2298.983680414763
Iteration:     16,   Func. Count:    108,   Neg. LLF: 2255.622829630608
Iteration:     17,   Func. Count:    115,   Neg. LLF: 2702.7731728890117
Iteration:     18,   Func. Count:    123,   Neg. LLF: 2600.018979687632
Iteration:     19,   Func. Count:    130,   Neg. LLF: 2575.215822136019
Iteration:     20,   Func. Count:    137,   Neg. LLF: 2171.0706340006586
Iteration:     21,   Func. Count:    144,   Neg. LLF: 2168.0930300650916
Iteration:     22,   Func. Count:    150,   Neg. LLF: 2168.0443351656004
Iteration:     23,   Func. Count:    156,   Neg. LLF: 2168.0350849527968
Iteration:     24,   Func. Count:    162,   Neg. LLF: 2168.032484272534
Iteration:     25,   Func. Count:    168,   Neg. LLF: 2168.029915846167
Iteration:     26,   Func. Count:    174,   Neg. LLF: 2168.0278248379836
Iteration:     27,   Func. Count:    180,   Neg. LLF: 2168.027744443383
Iteration:     28,   Func. Count:    186,   Neg. LLF: 2168.0271951683
Iteration:     29,   Func. Count:    192,   Neg. LLF: 2168.0274546278006
Optimization terminated successfully    (Exit mode 0)
            Current function value: 2168.027195433203
            Iterations: 29
            Function evaluations: 202
            Gradient evaluations: 29
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:                   None   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:                -2168.03
Distribution:                  Normal   AIC:                            4346.05
Method:            Maximum Likelihood   BIC:                            4367.14
                                        No. Observations:                  501
Date:                Mon, May 05 2025   Df Residuals:                      500
Time:                        21:16:55   Df Model:                            1
                               Mean Model
========================================================================
                 coef    std err          t      P>|t|  95.0% Conf. Int.
------------------------------------------------------------------------
mu             1.1214      0.829      1.353      0.176 [ -0.503,  2.746]
                             Volatility Model
==========================================================================
                 coef    std err          t      P>|t|     95.0% Conf. Int.
--------------------------------------------------------------------------
omega          5.1748     10.064      0.514      0.607   [-14.551, 24.900]
alpha[1]       0.1893  8.564e-02      2.211  2.706e-02 [2.146e-02,  0.357]
beta[1]        0.2607      0.361      0.723      0.470   [ -0.446,  0.967]
beta[2]        0.5499      0.375      1.468      0.142   [ -0.185,  1.284]
==========================================================================

Covariance estimator: robust
```

In [49]:

```
# Plot results and diagnostics
garch_result.plot()
plt.show()
```



In [57]:
```
import pandas as pd
from arch import arch_model
from statsmodels.stats.diagnostic import acorr_ljungbox
import matplotlib.pyplot as plt

garch_model = arch_model(squared_residuals, vol='Garch', p=1, q=2)
garch_result = garch_model.fit()

# Display model summary
print(garch_result.summary())

# Get standardized residuals
residuals = garch_result.std_resid

# Perform Ljung-Box test on residuals
ljung_box = acorr_ljungbox(residuals, lags= 20, return_df=True)

print("\nLjung-Box Test Results:")
print(ljung_box)

```

```
Iteration:      1,   Func. Count:      7,   Neg. LLF: 4407.444540132237
Iteration:      2,   Func. Count:     14,   Neg. LLF: 3060.4798863927226
Iteration:      3,   Func. Count:     22,   Neg. LLF: 2651.2556633005424
Iteration:      4,   Func. Count:     29,   Neg. LLF: 2270.2972455413255
Iteration:      5,   Func. Count:     36,   Neg. LLF: 2179.308537785706
Iteration:      6,   Func. Count:     42,   Neg. LLF: 2188.481812168503
Iteration:      7,   Func. Count:     49,   Neg. LLF: 2177.3174160922003
Iteration:      8,   Func. Count:     55,   Neg. LLF: 2176.624413878339
Iteration:      9,   Func. Count:     61,   Neg. LLF: 2176.1501409323137
Iteration:     10,   Func. Count:     67,   Neg. LLF: 2174.96241701514
Iteration:     11,   Func. Count:     73,   Neg. LLF: 2372.532333221814
Iteration:     12,   Func. Count:     80,   Neg. LLF: 2394.324020554337
Iteration:     13,   Func. Count:     87,   Neg. LLF: 2278.083498148742
Iteration:     14,   Func. Count:     94,   Neg. LLF: 2419.9216926077443
Iteration:     15,   Func. Count:    101,   Neg. LLF: 2298.983680414763
Iteration:     16,   Func. Count:    108,   Neg. LLF: 2255.622829630608
Iteration:     17,   Func. Count:    115,   Neg. LLF: 2702.7731728890117
Iteration:     18,   Func. Count:    123,   Neg. LLF: 2600.018979687632
Iteration:     19,   Func. Count:    130,   Neg. LLF: 2575.215822136019
Iteration:     20,   Func. Count:    137,   Neg. LLF: 2171.0706340006586
Iteration:     21,   Func. Count:    144,   Neg. LLF: 2168.0930300650916
Iteration:     22,   Func. Count:    150,   Neg. LLF: 2168.0443351656004
Iteration:     23,   Func. Count:    156,   Neg. LLF: 2168.0350849527968
Iteration:     24,   Func. Count:    162,   Neg. LLF: 2168.032484272534
Iteration:     25,   Func. Count:    168,   Neg. LLF: 2168.029915846167
Iteration:     26,   Func. Count:    174,   Neg. LLF: 2168.0278248379836
Iteration:     27,   Func. Count:    180,   Neg. LLF: 2168.027744443383
Iteration:     28,   Func. Count:    186,   Neg. LLF: 2168.0271951683
Iteration:     29,   Func. Count:    192,   Neg. LLF: 2168.0274546278006
Optimization terminated successfully    (Exit mode 0)
            Current function value: 2168.027195433203
            Iterations: 29
            Function evaluations: 202
            Gradient evaluations: 29
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:                   None   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:               -2168.03
Distribution:                  Normal   AIC:                           4346.05
Method:            Maximum Likelihood   BIC:                           4367.14
                                        No. Observations:                  501
Date:                Mon, May 05 2025   Df Residuals:                      500
Time:                        21:52:47   Df Model:                            1
                               Mean Model
========================================================================
                 coef    std err          t      P>|t|  95.0% Conf. Int.
------------------------------------------------------------------------
mu             1.1214      0.829      1.353      0.176 [ -0.503,  2.746]
                              Volatility Model
==========================================================================
                 coef    std err          t      P>|t|    95.0% Conf. Int.
--------------------------------------------------------------------------
omega          5.1748     10.064      0.514      0.607   [-14.551, 24.900]
alpha[1]       0.1893  8.564e-02      2.211  2.706e-02 [2.146e-02,  0.357]
beta[1]        0.2607      0.361      0.723      0.470   [ -0.446,  0.967]
beta[2]        0.5499      0.375      1.468      0.142   [ -0.185,  1.284]
==========================================================================

Covariance estimator: robust

Ljung-Box Test Results:
      lb_stat  lb_pvalue
1    3.001522   0.083186
2    6.009235   0.049558
3    6.239033   0.100543
4    6.693593   0.152994
5    6.874028   0.230179
6   15.229046   0.018548
7   15.680577   0.028200
8   16.352810   0.037600
```

```
9   17.499147   0.041450
10  17.606637   0.061973
11  18.273491   0.075448
12  19.236631   0.082976
13  24.962605   0.023346
14  25.476277   0.030146
15  26.309785   0.034890
16  26.317864   0.049717
17  26.503731   0.065759
18  26.551016   0.087810
19  27.164798   0.100873
20  27.712404   0.116384
```

In [74]:

```python
import numpy as np
import matplotlib.pyplot as plt

lags = ljung_box.index.to_numpy()
p_values = ljung_box['lb_pvalue'].to_numpy()

plt.plot(lags, p_values, marker='o', color='red')
plt.xlabel('Lag')
plt.ylabel('p-value')
plt.title('Ljung-Box Test Results')
plt.axhline(y=0.05, color='blue', linestyle='--', label='Significance Level (0.05)')
plt.legend()
plt.grid(True)
plt.show()

```