Objective

To analyze how macro indicators (like inflation, interest rates, unemployment) relate to equity market performance (like S&P 500, Euro Stoxx 50, etc.), and visualize the relationships using Python.

The following code cell fetches data for the Consumer Price Index (Inflation), Unemployment Rate, Federal Funds Rate (Interest Rate), and the S&P 500 index from FRED using `pandas_datareader`. It then combines these into a single pandas DataFrame called `macro_data` and displays the first few rows.

In [ ]:
```
# Core Libraries
import pandas as pd
import numpy as np

# Visualization Libraries
import matplotlib.pyplot as plt
import seaborn as sns

# For Financial & Economic Data
import pandas_datareader.data as web #pandas_datareader → To pull data di
from datetime import datetime
```

In [ ]:
```
# Define time range
start = datetime(2015, 1, 1)
end = datetime(2024, 12, 31)

# Fetch data for each indicator individually
inflation_data = web.DataReader('CPIAUCSL', 'fred', start, end)
unemployment_data = web.DataReader('UNRATE', 'fred', start, end)
interest_rate_data = web.DataReader('FEDFUNDS', 'fred', start, end)
sp500_data = web.DataReader('SP500', 'fred', start, end)

# Print the type and head of each fetched data (for debugging)
# print("Inflation data type:", type(inflation_data))
# print(inflation_data.head())
# print("\nUnemployment data type:", type(unemployment_data))
# print(unemployment_data.head())
# print("\nInterest Rate data type:", type(interest_rate_data))
# print(interest_rate_data.head())
# print("\nSP500 data type:", type(sp500_data))
# print(sp500_data.head())

# Combine into a single DataFrame by merging on index
macro_data = inflation_data.merge(unemployment_data, left_index=True, rig
macro_data = macro_data.merge(interest_rate_data, left_index=True, right_
macro_data = macro_data.merge(sp500_data, left_index=True, right_index=Tr


# Preview first few rows
print(macro_data.head())
macro_data.describe()
macro_data.info()
```

```
            CPIAUCSL  UNRATE  FEDFUNDS  SP500
DATE
2015-01-01   234.747     5.7      0.11    NaN
2015-02-01   235.342     5.5      0.11    NaN
2015-03-01   235.976     5.4      0.11    NaN
2015-04-01   236.222     5.4      0.12    NaN
2015-05-01   237.001     5.6      0.12    NaN
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2430 entries, 2015-01-01 to 2024-12-31
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   CPIAUCSL  120 non-null    float64
 1   UNRATE    120 non-null    float64
 2   FEDFUNDS  120 non-null    float64
 3   SP500     2304 non-null   float64
dtypes: float64(4)
memory usage: 94.9 KB
```

In [ ]:
```
# Convert daily data to monthly (to match macro frequency)
data_monthly = macro_data.resample('M').last()

# Compute monthly % changes for S&P 500 (returns)
data_monthly['SP500_Return'] = data_monthly['SP500'].pct_change() * 100

# Drop missing values
data_monthly = data_monthly.dropna()

data_monthly.info()
data_monthly.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 109 entries, 2015-12-31 to 2024-12-31
Freq: ME
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   CPIAUCSL      109 non-null    float64
 1   UNRATE        109 non-null    float64
 2   FEDFUNDS      109 non-null    float64
 3   SP500         109 non-null    float64
 4   SP500_Return  109 non-null    float64
dtypes: float64(5)
memory usage: 5.1 KB
```

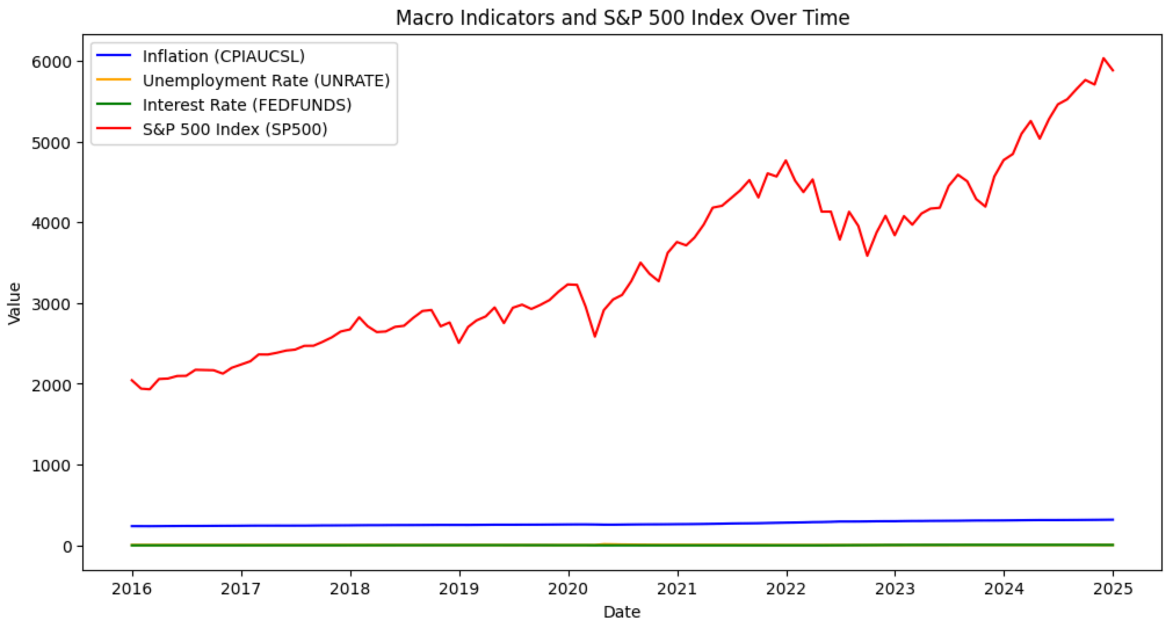
```
/tmp/ipython-input-2439356905.py:2: FutureWarning: 'M' is deprecated and w
ill be removed in a future version, please use 'ME' instead.
  data_monthly = macro_data.resample('M').last()
```

Out[ ]:

	CPIAUCSL	UNRATE	FEDFUNDS	SP500	SP500_Return
count	109.000000	109.000000	109.000000	109.000000	109.000000
mean	270.004734	4.622018	1.950092	3498.049725	1.057586
std	25.831273	1.805412	1.886210	1060.975211	4.470309
min	237.336000	3.400000	0.050000	1932.230000	-12.511932
25%	249.577000	3.700000	0.360000	2648.050000	-1.113664
50%	259.127000	4.000000	1.420000	3230.780000	1.785938
75%	296.421000	4.800000	2.560000	4288.050000	3.577324
max	317.603000	14.800000	5.330000	6032.380000	12.684410

In [ ]:

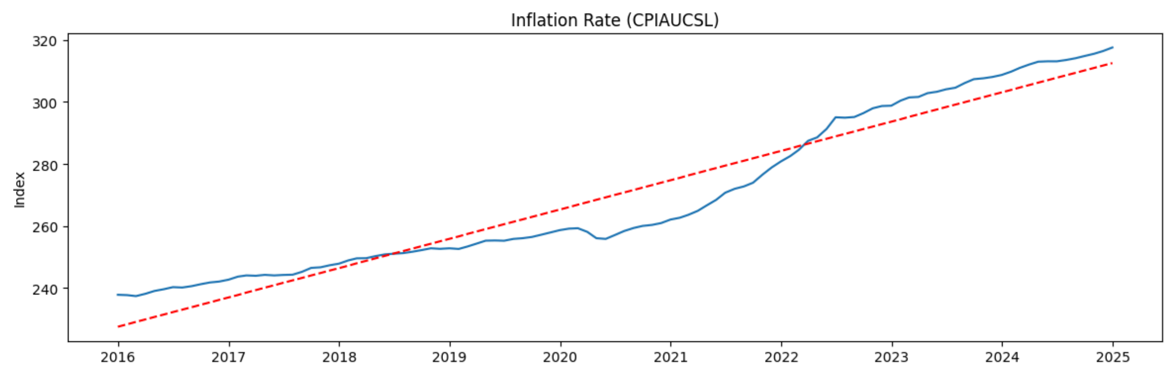
```
# Create a single figure and axes object for the plot
fig, ax = plt.subplots(figsize=(12, 6))

# Plot each series on the same axes with different colors
ax.plot(data_monthly.index, data_monthly['CPIAUCSL'], label='Inflation (C
ax.plot(data_monthly.index, data_monthly['UNRATE'], label='Unemployment R
ax.plot(data_monthly.index, data_monthly['FEDFUNDS'], label='Interest Rat
ax.plot(data_monthly.index, data_monthly['SP500'], label='S&P 500 Index (

# Set the title and labels
ax.set_title('Macro Indicators and S&P 500 Index Over Time')
ax.set_xlabel('Date')
ax.set_ylabel('Value') # A generic label as scales differ

# Add a legend
ax.legend()

# Display the plot
plt.show()
```

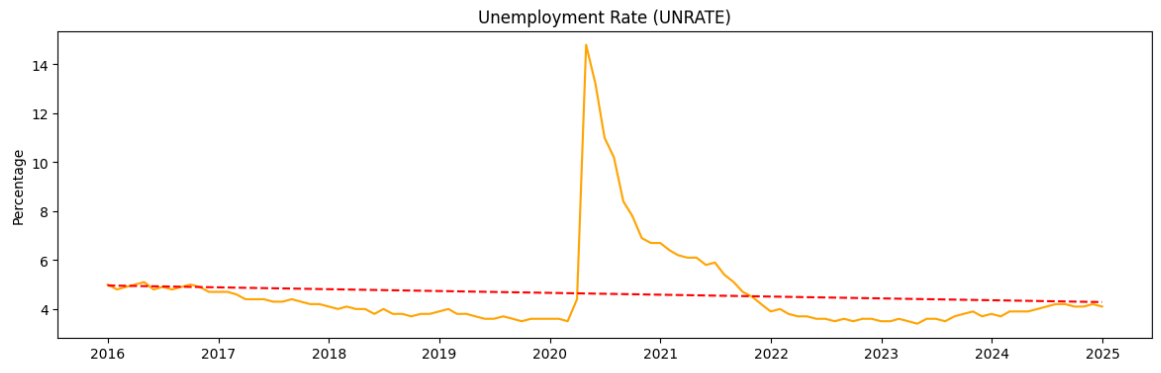


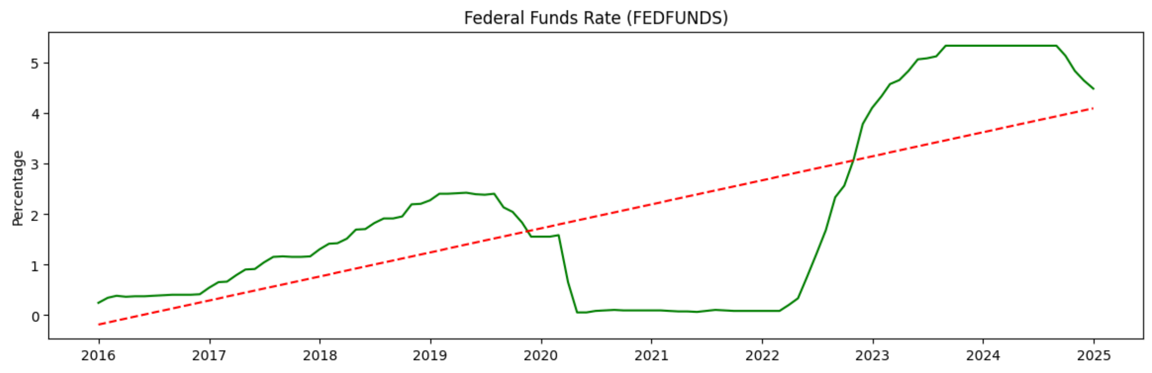
In [ ]:

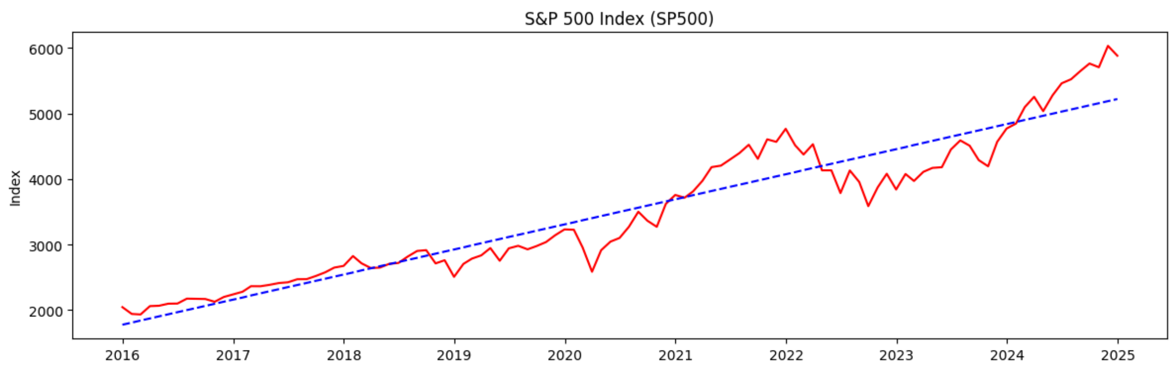
```
import numpy as np
from sklearn.linear_model import LinearRegression
```

```python
# Create a figure and a set of subplots
fig, axes = plt.subplots(nrows=4, ncols=1, figsize=(12, 15))

# Plot Inflation Rate with trendline
axes[0].plot(data_monthly.index, data_monthly['CPIAUCSL'])
axes[0].set_title('Inflation Rate (CPIAUCSL)')
axes[0].set_ylabel('Index')
# Add trendline
X = np.arange(len(data_monthly)).reshape(-1, 1)
y = data_monthly['CPIAUCSL'].values.reshape(-1, 1)
model = LinearRegression().fit(X, y)
trendline = model.predict(X)
axes[0].plot(data_monthly.index, trendline, color='red', linestyle='--')


# Plot Unemployment Rate with trendline
axes[1].plot(data_monthly.index, data_monthly['UNRATE'], color='orange')
axes[1].set_title('Unemployment Rate (UNRATE)')
axes[1].set_ylabel('Percentage')
# Add trendline
X = np.arange(len(data_monthly)).reshape(-1, 1)
y = data_monthly['UNRATE'].values.reshape(-1, 1)
model = LinearRegression().fit(X, y)
trendline = model.predict(X)
axes[1].plot(data_monthly.index, trendline, color='red', linestyle='--')


# Plot Federal Funds Rate with trendline
axes[2].plot(data_monthly.index, data_monthly['FEDFUNDS'], color='green')
axes[2].set_title('Federal Funds Rate (FEDFUNDS)')
axes[2].set_ylabel('Percentage')
# Add trendline
X = np.arange(len(data_monthly)).reshape(-1, 1)
y = data_monthly['FEDFUNDS'].values.reshape(-1, 1)
model = LinearRegression().fit(X, y)
trendline = model.predict(X)
axes[2].plot(data_monthly.index, trendline, color='red', linestyle='--')


# Plot S&P 500 Index with trendline
axes[3].plot(data_monthly.index, data_monthly['SP500'], color='red')
axes[3].set_title('S&P 500 Index (SP500)')
axes[3].set_ylabel('Index')
# Add trendline
X = np.arange(len(data_monthly)).reshape(-1, 1)
y = data_monthly['SP500'].values.reshape(-1, 1)
model = LinearRegression().fit(X, y)
trendline = model.predict(X)
axes[3].plot(data_monthly.index, trendline, color='blue', linestyle='--')


# Adjust layout
plt.tight_layout()

# Display the plots
plt.show()
```









Based on the visualizations, we can observe the following trends:

- **Inflation (CPIAUCSL):** Shows an overall upward trend, with some fluctuations.
- **Unemployment Rate (UNRATE):** Shows a general downward trend, indicating a decrease in unemployment over the period, although there are some cyclical movements.
- **Federal Funds Rate (FEDFUNDS):** Shows a variable trend, with periods of increase and decrease, reflecting monetary policy changes.
- **S&P 500 Index (SP500):** Shows a strong upward trend, indicating overall growth in the stock market over the period.

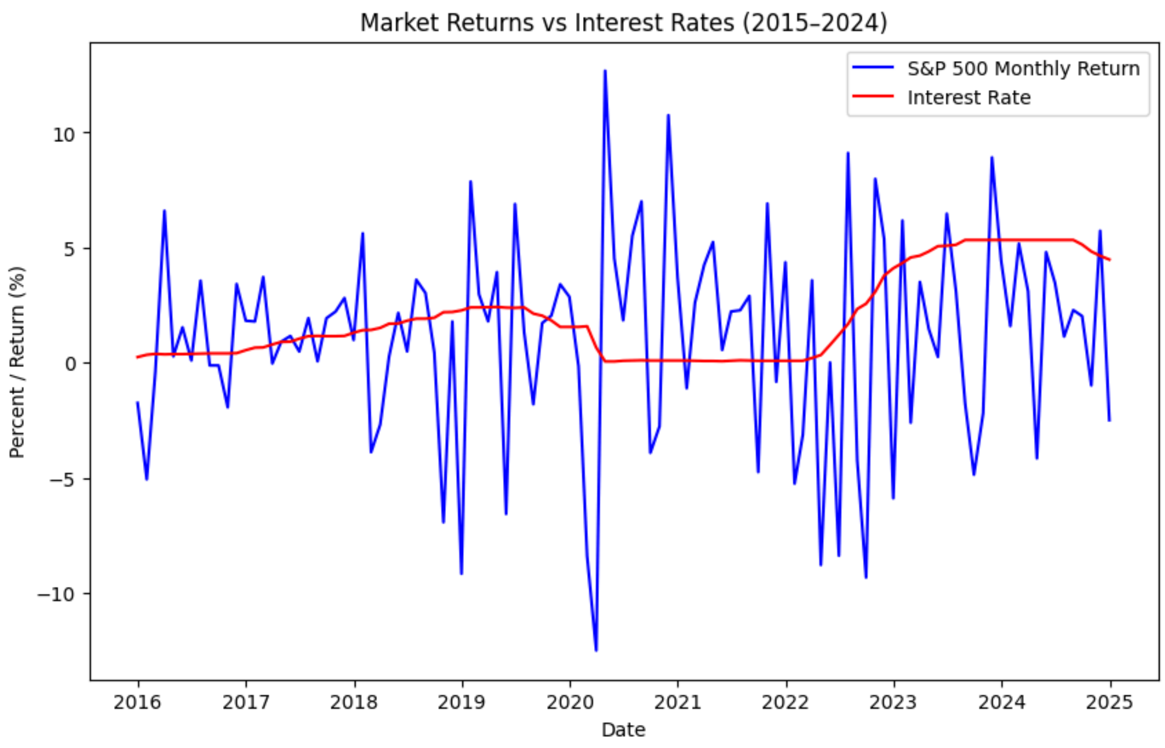
To further analyze the relationship between these indicators and the S&P 500, we could consider calculating and visualizing correlations or performing regression analysis.

Let's rename the columns in the `data_monthly` DataFrame for better readability.

In [ ]:
```python
# Rename columns for clarity
data_monthly = data_monthly.rename(columns={
    'CPIAUCSL': 'Inflation',
    'UNRATE': 'Unemployment_Rate',
    'FEDFUNDS': 'Interest_Rate',
    'SP500': 'SP500_Index'
})

# Display the updated DataFrame head
print(data_monthly.head())
```

```
            Inflation  Unemployment_Rate  Interest_Rate  SP500_Index  \
DATE
2015-12-31    237.761                5.0           0.24      2043.94
2016-01-31    237.652                4.8           0.34      1940.24
2016-02-29    237.336                4.9           0.38      1932.23
2016-03-31    238.080                5.0           0.36      2059.74
2016-04-30    238.992                5.1           0.37      2065.30

            SP500_Return
DATE
2015-12-31     -1.753020
2016-01-31     -5.073534
2016-02-29     -0.412836
2016-03-31      6.599111
2016-04-30      0.269937
```

In [ ]:
```python
plt.figure(figsize=(10, 6))
plt.plot(data_monthly['SP500_Return'], label='S&P 500 Monthly Return', co
plt.plot(data_monthly['Interest_Rate'], label='Interest Rate', color='red
plt.title('Market Returns vs Interest Rates (2015–2024)')
plt.xlabel('Date')
plt.ylabel('Percent / Return (%)')
plt.legend()
plt.show()
```



```
In [ ]:
```

Here are some observations we can make from the chart:

Periods of Low Interest Rates: From roughly 2015 to late 2016, interest rates were very low. During this period, the S&P 500 monthly returns show significant fluctuations, both positive and negative.

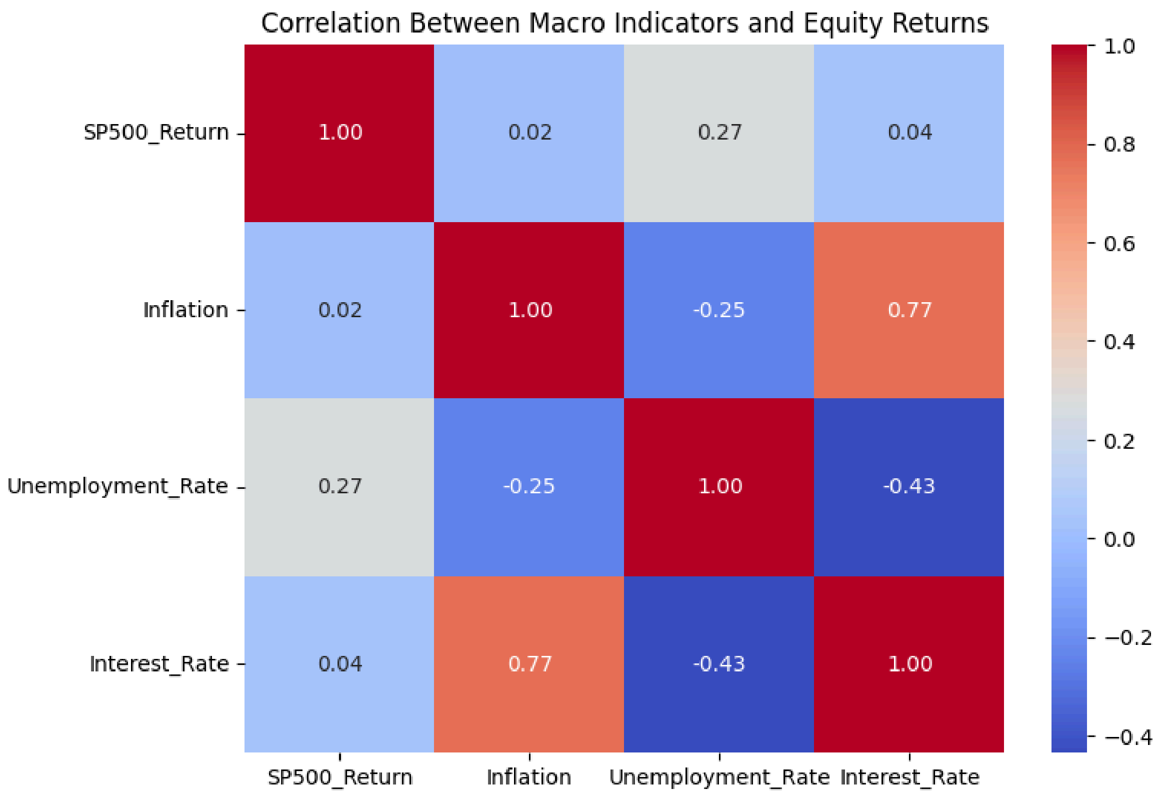
Rising Interest Rates: From late 2016 to late 2018, interest rates gradually increased. The S&P 500 monthly returns continued to be volatile during this period. There isn't a clear, simple inverse relationship immediately obvious just by looking at the monthly returns alongside the gradual rise in interest rates.

Interest Rate Cuts and Stability: In 2019 and early 2020, interest rates were cut and then remained low. The S&P 500 experienced some strong positive returns during parts of this period, but also a significant sharp drop in early 2020 (likely related to the start of the COVID-19 pandemic), which coincided with interest rates being cut to near zero.

Recent Rising Interest Rates: From early 2022 onwards, interest rates have risen sharply. During this period of rapidly increasing interest rates, we can observe that the S&P 500 monthly returns have been quite volatile, with several months showing negative returns. This aligns more closely with the expected inverse relationship between interest rates and stock market performance – as borrowing costs increase, it can negatively impact company profits and investor sentiment.

It's important to note that this is a visual interpretation and doesn't prove a direct causal relationship. Many factors influence stock market returns. However, the chart does allow us to see how these two variables have moved relative to each other over time.

In [ ]:
```
# Select key variables
corr_data = data_monthly[['SP500_Return', 'Inflation', 'Unemployment_Rate

# Compute correlation matrix
corr_matrix = corr_data.corr()

# Visualize with heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Between Macro Indicators and Equity Returns')
plt.show()
```



To check for stationarity of the time series data, we can use the Augmented Dickey-Fuller (ADF) test. A stationary time series has statistical properties (like mean and variance) that do not change over time. Many time series models assume stationarity.

The null hypothesis of the ADF test is that the time series is non-stationary. If the p-value is less than a significance level (e.g., 0.05), we reject the null hypothesis and conclude that the time series is stationary.

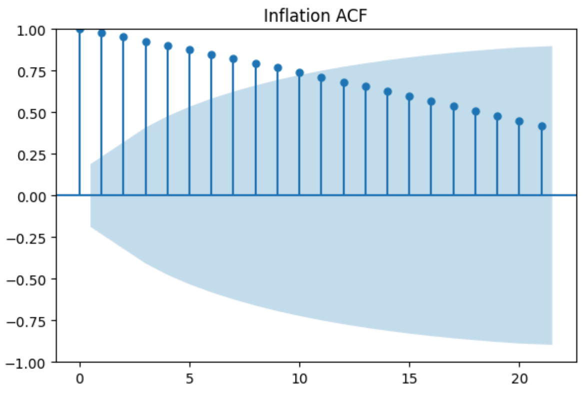
In [ ]:
```
from statsmodels.tsa.stattools import adfuller

def check_stationarity(series, name):
    """
    Performs the Augmented Dickey-Fuller test and prints the results.
    """
    result = adfuller(series.dropna()) # Drop NaNs for the test
    print(f'Augmented Dickey-Fuller Test for {name}:')
    print(f'  Test Statistic: {result[0]:.4f}')
    print(f'  P-value: {result[1]:.4f}')
    print('  Critical Values:')
    for key, value in result[4].items():
        print(f'    {key}: {value:.4f}')
    if result[1] <= 0.05:
        print(f'  Conclusion: {name} is likely Stationary (reject H0)\n')
    else:
        print(f'  Conclusion: {name} is likely Non-Stationary (fail to re

# Check stationarity for each series
check_stationarity(data_monthly['Inflation'], 'Inflation')
check_stationarity(data_monthly['Unemployment_Rate'], 'Unemployment_Rate'
check_stationarity(data_monthly['Interest_Rate'], 'Interest_Rate')
check_stationarity(data_monthly['SP500_Index'], 'SP500_Index')
check_stationarity(data_monthly['SP500_Return'], 'SP500_Return')
```

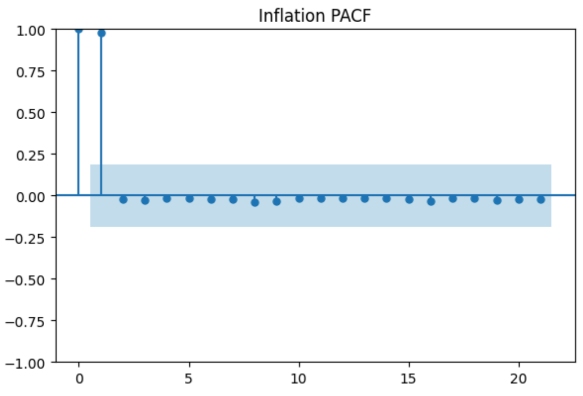
```
Augmented Dickey-Fuller Test for Inflation:
  Test Statistic: -0.2204
  P-value: 0.9360
  Critical Values:
    1%: -3.4968
    5%: -2.8906
    10%: -2.5823
  Conclusion: Inflation is likely Non-Stationary (fail to reject H0)

Augmented Dickey-Fuller Test for Unemployment_Rate:
  Test Statistic: -3.1897
  P-value: 0.0206
  Critical Values:
    1%: -3.4924
    5%: -2.8887
    10%: -2.5813
  Conclusion: Unemployment_Rate is likely Stationary (reject H0)

Augmented Dickey-Fuller Test for Interest_Rate:
  Test Statistic: -2.3692
  P-value: 0.1506
  Critical Values:
    1%: -3.4955
    5%: -2.8900
    10%: -2.5820
  Conclusion: Interest_Rate is likely Non-Stationary (fail to reject H0)

Augmented Dickey-Fuller Test for SP500_Index:
  Test Statistic: 0.4831
  P-value: 0.9843
  Critical Values:
    1%: -3.4930
    5%: -2.8890
    10%: -2.5814
  Conclusion: SP500_Index is likely Non-Stationary (fail to reject H0)

Augmented Dickey-Fuller Test for SP500_Return:
  Test Statistic: -9.1980
  P-value: 0.0000
  Critical Values:
    1%: -3.4930
    5%: -2.8890
    10%: -2.5814
  Conclusion: SP500_Return is likely Stationary (reject H0)
```

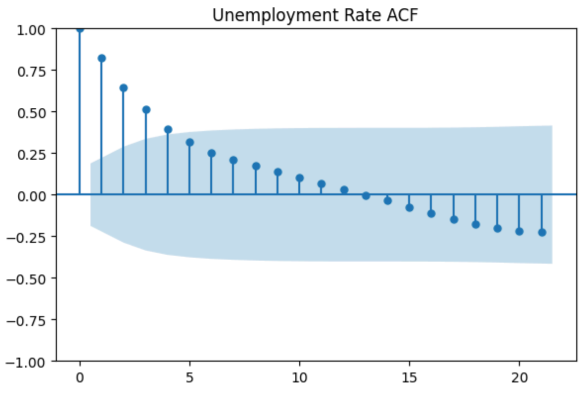
Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots are used to identify the structure of a time series and determine the appropriate model for forecasting (e.g., ARIMA).

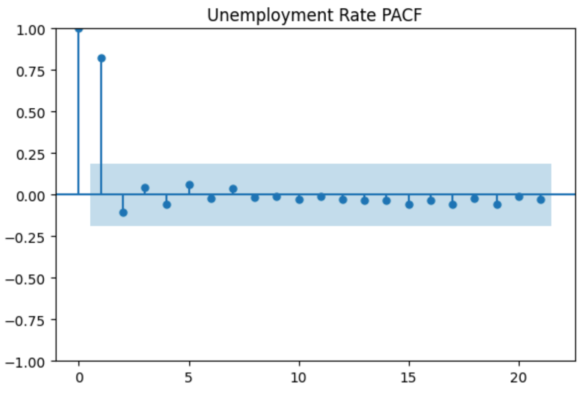
- **ACF** shows the correlation of a time series with its own lagged values, including both direct and indirect effects.
- **PACF** shows the correlation of a time series with its own lagged values, after accounting for the correlations at intermediate lags (only the direct effect).

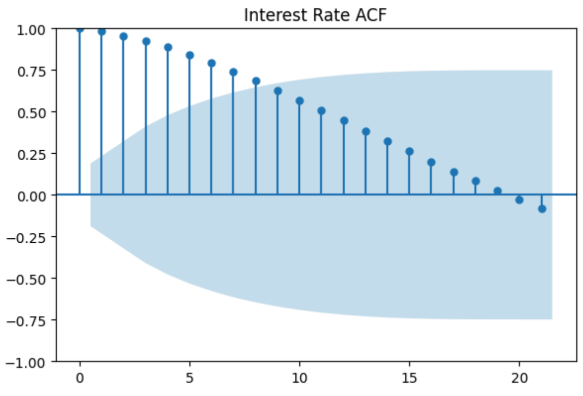
In [ ]:
```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Create ACF and PACF plots for each series
fig, axes = plt.subplots(nrows=5, ncols=2, figsize=(12, 20))

plot_acf(data_monthly['Inflation'].dropna(), ax=axes[0, 0], title='Inflat
plot_pacf(data_monthly['Inflation'].dropna(), ax=axes[0, 1], title='Infla

plot_acf(data_monthly['Unemployment_Rate'].dropna(), ax=axes[1, 0], title
plot_pacf(data_monthly['Unemployment_Rate'].dropna(), ax=axes[1, 1], titl

plot_acf(data_monthly['Interest_Rate'].dropna(), ax=axes[2, 0], title='In
plot_pacf(data_monthly['Interest_Rate'].dropna(), ax=axes[2, 1], title='I

plot_acf(data_monthly['SP500_Index'].dropna(), ax=axes[3, 0], title='SP50
plot_pacf(data_monthly['SP500_Index'].dropna(), ax=axes[3, 1], title='SP5

plot_acf(data_monthly['SP500_Return'].dropna(), ax=axes[4, 0], title='SP5
plot_pacf(data_monthly['SP500_Return'].dropna(), ax=axes[4, 1], title='SP

plt.tight_layout()
plt.show()
```

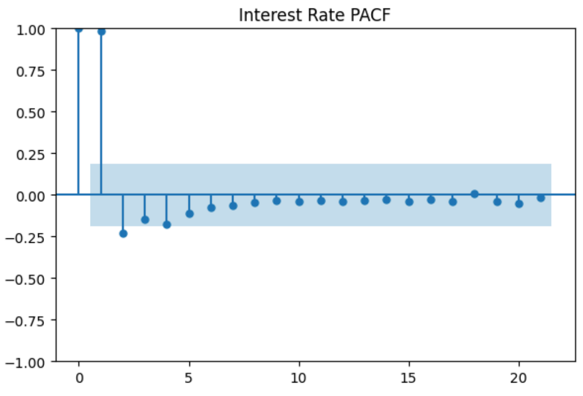

Inflation ACF
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

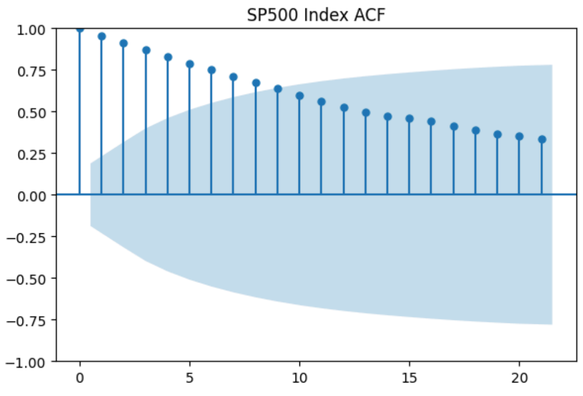

Inflation PACF
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

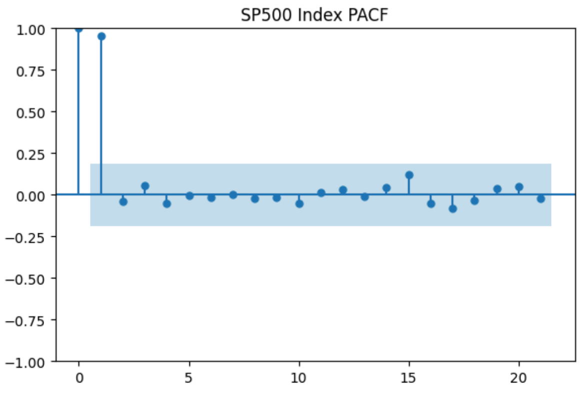

Unemployment Rate ACF
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

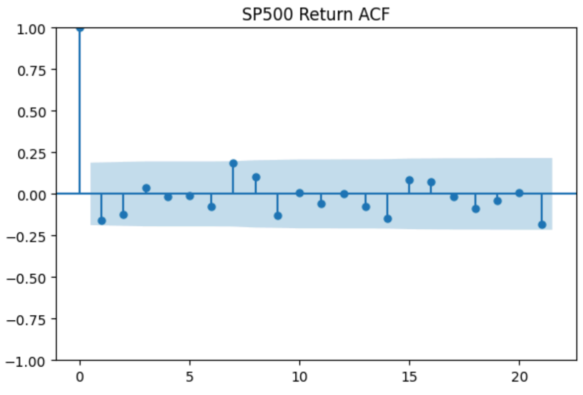

Unemployment Rate PACF
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

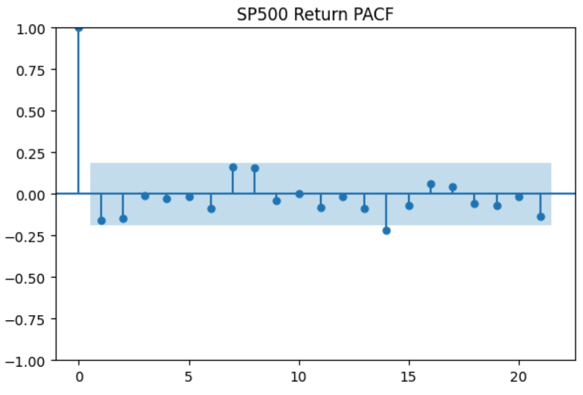

Interest Rate ACF
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20


Interest Rate PACF
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20


SP500 Index ACF
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20


SP500 Index PACF
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20


SP500 Return ACF
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20


SP500 Return PACF
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

In [ ]:

Task

Analyze the stationarity of the time series data for Inflation, Unemployment, and Interest Rate, determine the required differencing order for non-stationary series, and suggest appropriate time series modeling techniques.

Difference the non-stationary series

Subtask:

Apply first-order differencing to the 'Inflation' and 'Interest_Rate' series in the `data_monthly` DataFrame.

Reasoning: The subtask requires calculating the first-order difference for the 'Inflation' and 'Interest_Rate' columns and adding these as new columns to the `data_monthly` DataFrame. This can be done efficiently in a single code block using the `.diff()` method.

In [ ]:

```python
# Apply first-order differencing to 'Inflation' and 'Interest_Rate'
data_monthly['Inflation_Diff'] = data_monthly['Inflation'].diff()
data_monthly['Interest_Rate_Diff'] = data_monthly['Interest_Rate'].diff()

# Display the head of the DataFrame with the new columns
print(data_monthly.head())
```

```
            Inflation  Unemployment_Rate  Interest_Rate  SP500_Index  \
DATE
2015-12-31    237.761                5.0           0.24      2043.94
2016-01-31    237.652                4.8           0.34      1940.24
2016-02-29    237.336                4.9           0.38      1932.23
2016-03-31    238.080                5.0           0.36      2059.74
2016-04-30    238.992                5.1           0.37      2065.30

            SP500_Return  Inflation_Diff  Interest_Rate_Diff
DATE
2015-12-31     -1.753020             NaN                 NaN
2016-01-31     -5.073534          -0.109                0.10
2016-02-29     -0.412836          -0.316                0.04
2016-03-31      6.599111           0.744               -0.02
2016-04-30      0.269937           0.912                0.01
```

Check stationarity of differenced series

Subtask:

Perform the ADF test on the differenced 'Inflation' and 'Interest_Rate' series to see if they are now stationary.

Reasoning: Perform the ADF test on the differenced 'Inflation' and 'Interest_Rate' series to check for stationarity.

In [ ]:
```
# Check stationarity for the differenced Inflation series
check_stationarity(data_monthly['Inflation_Diff'], 'Inflation_Diff')

# Check stationarity for the differenced Interest Rate series
check_stationarity(data_monthly['Interest_Rate_Diff'], 'Interest_Rate_Dif
```

```
Augmented Dickey-Fuller Test for Inflation_Diff:
  Test Statistic: -1.7281
  P-value: 0.4167
  Critical Values:
    1%: -3.4968
    5%: -2.8906
    10%: -2.5823
  Conclusion: Inflation_Diff is likely Non-Stationary (fail to reject H0)

Augmented Dickey-Fuller Test for Interest_Rate_Diff:
  Test Statistic: -2.5938
  P-value: 0.0943
  Critical Values:
    1%: -3.4942
    5%: -2.8895
    10%: -2.5817
  Conclusion: Interest_Rate_Diff is likely Non-Stationary (fail to reject
H0)
```

Determine further differencing (if needed)

Subtask:

If the series are still non-stationary after first-order differencing, apply second-order differencing and repeat the ADF test.

Reasoning: Apply second-order differencing to the non-stationary series and check their stationarity using the ADF test.

In [ ]:
```
# Apply second-order differencing to 'Inflation' and 'Interest_Rate'
data_monthly['Inflation_Diff2'] = data_monthly['Inflation_Diff'].diff()
data_monthly['Interest_Rate_Diff2'] = data_monthly['Interest_Rate_Diff'].

# Perform ADF test on the second-differenced series
check_stationarity(data_monthly['Inflation_Diff2'], 'Inflation_Diff2')
check_stationarity(data_monthly['Interest_Rate_Diff2'], 'Interest_Rate_Di
Augmented Dickey-Fuller Test for Inflation_Diff2:
  Test Statistic: -7.9131
  P-value: 0.0000
  Critical Values:
    1%: -3.4968
    5%: -2.8906
    10%: -2.5823
  Conclusion: Inflation_Diff2 is likely Stationary (reject H0)

Augmented Dickey-Fuller Test for Interest_Rate_Diff2:
  Test Statistic: -7.7809
  P-value: 0.0000
  Critical Values:
    1%: -3.4955
    5%: -2.8900
    10%: -2.5820
  Conclusion: Interest_Rate_Diff2 is likely Stationary (reject H0)
```

Identify appropriate time series models

Subtask:

Based on the stationarity of the series and the ACF/PACF plots, discuss suitable time series modeling techniques (e.g., ARIMA, vector autoregression - VAR).

Reasoning: Summarize the stationarity findings and discuss suitable time series modeling techniques based on the ADF test results and ACF/PACF plots.

In [ ]:
```
# Summarize stationarity results and discuss modeling techniques

print("--- Stationarity Analysis Summary and Modeling Suggestions ---")

print("\nStationarity Test Results (ADF Test):")
print("- Inflation: Non-Stationary (p > 0.05)")
print("- Unemployment_Rate: Stationary (p <= 0.05)")
print("- Interest_Rate: Non-Stationary (p > 0.05)")
print("- SP500_Index: Non-Stationary (p > 0.05)")
print("- SP500_Return: Stationary (p <= 0.05)")
print("- Inflation_Diff: Non-Stationary (p > 0.05)")
print("- Interest_Rate_Diff: Non-Stationary (p > 0.05)")
print("- Inflation_Diff2: Stationary (p <= 0.05)")
print("- Interest_Rate_Diff2: Stationary (p <= 0.05)")


print("\nDiscussion on ARIMA Modeling for Individual Series:")
print("Based on the ADF tests and ACF/PACF plots:")

print("\n- Inflation:")
print("  - The original series is non-stationary. First-order differencin
print("  - The ACF and PACF plots for 'Inflation_Diff2' can help determin
print("  - The ACF of 'Inflation_Diff2' shows a significant spike at lag
print("  - The PACF of 'Inflation_Diff2' shows a significant spike at lag
print("  - A possible ARIMA model for Inflation could be ARIMA(p, 2, q) w

print("\n- Unemployment_Rate:")
print("  - The original series is stationary (d=0).")
print("  - The ACF of 'Unemployment_Rate' shows a slow decay, and the PAC
print("  - An ARMA(p, q) model could be appropriate. The PACF suggests po
print("  - Further analysis of ACF/PACF and possibly information criteria

print("\n- Interest_Rate:")
print("  - The original series is non-stationary. First-order differencin
print("  - The ACF and PACF plots for 'Interest_Rate_Diff2' can help dete
print("  - The ACF of 'Interest_Rate_Diff2' shows a significant spike at
print("  - The PACF of 'Interest_Rate_Diff2' shows a significant spike at
print("  - A possible ARIMA model for Interest Rate could be ARIMA(p, 2,

print("\nDiscussion on Vector Autoregression (VAR) Modeling:")
print("- We have multiple time series (Inflation, Unemployment Rate, Inte
print("- The SP500 Return series is stationary.")
print("- The differenced series 'Inflation_Diff2' and 'Interest_Rate_Diff
print("- The Unemployment Rate is also stationary.")
print("- Since we have multiple stationary time series variables (SP500_R
print("- VAR models are designed to capture the linear interdependencies
print("- A VAR model would allow us to analyze how changes in one variabl
print("- To apply VAR, all variables in the model must be stationary. The

print("\nSummary of Suitable Modeling Techniques:")
print("- For modeling individual series, ARIMA models are appropriate. AR
print("- For modeling the relationships between the macro indicators and
--- Stationarity Analysis Summary and Modeling Suggestions ---

Stationarity Test Results (ADF Test):
- Inflation: Non-Stationary (p > 0.05)
- Unemployment_Rate: Stationary (p <= 0.05)
- Interest_Rate: Non-Stationary (p > 0.05)
- SP500_Index: Non-Stationary (p > 0.05)
- SP500_Return: Stationary (p <= 0.05)
- Inflation_Diff: Non-Stationary (p > 0.05)
- Interest_Rate_Diff: Non-Stationary (p > 0.05)
- Inflation_Diff2: Stationary (p <= 0.05)
- Interest_Rate_Diff2: Stationary (p <= 0.05)

Discussion on ARIMA Modeling for Individual Series:
Based on the ADF tests and ACF/PACF plots:

- Inflation:
  - The original series is non-stationary. First-order differencing was no
t sufficient, but second-order differencing ('Inflation_Diff2') achieved s
tationarity (d=2).
  - The ACF and PACF plots for 'Inflation_Diff2' can help determine the ap
propriate p and q orders for an ARIMA(p, 2, q) model.
  - The ACF of 'Inflation_Diff2' shows a significant spike at lag 1, and t
hen quickly drops, suggesting a potential MA(1) component (q=1).
  - The PACF of 'Inflation_Diff2' shows a significant spike at lag 1 and p
ossibly lag 2, suggesting potential AR components (p=1 or p=2).
  - A possible ARIMA model for Inflation could be ARIMA(p, 2, q) where p a
nd q are determined by further analysis of the ACF/PACF of the differenced
series, potentially ARIMA(1, 2, 1) or ARIMA(2, 2, 1).

- Unemployment_Rate:
  - The original series is stationary (d=0).
  - The ACF of 'Unemployment_Rate' shows a slow decay, and the PACF shows
significant spikes at the first few lags, suggesting an AR process.
  - An ARMA(p, q) model could be appropriate. The PACF suggests potential
AR components (p), and the ACF suggests potential MA components (q). The P
ACF has significant spikes at lags 1, 2, and 3, suggesting p could be up t
o 3. The ACF shows a more gradual decay.
  - Further analysis of ACF/PACF and possibly information criteria (AIC/BI
C) would be needed to determine the optimal p and q for an ARMA(p, q) mode
l.

- Interest_Rate:
  - The original series is non-stationary. First-order differencing was no
t sufficient, but second-order differencing ('Interest_Rate_Diff2') achiev
ed stationarity (d=2).
  - The ACF and PACF plots for 'Interest_Rate_Diff2' can help determine th
e appropriate p and q orders for an ARIMA(p, 2, q) model.
  - The ACF of 'Interest_Rate_Diff2' shows a significant spike at lag 1, s
uggesting a potential MA(1) component (q=1).
  - The PACF of 'Interest_Rate_Diff2' shows a significant spike at lag 1,
suggesting a potential AR(1) component (p=1).
  - A possible ARIMA model for Interest Rate could be ARIMA(p, 2, q) where
p and q are determined by further analysis of the ACF/PACF of the differen
ced series, potentially ARIMA(1, 2, 1).

Discussion on Vector Autoregression (VAR) Modeling:
- We have multiple time series (Inflation, Unemployment Rate, Interest Rat
e, SP500 Return) and are interested in the relationships between them.
- The SP500 Return series is stationary.
- The differenced series 'Inflation_Diff2' and 'Interest_Rate_Diff2' are s
tationary.
- The Unemployment Rate is also stationary.
- Since we have multiple stationary time series variables (SP500_Return, U
nemployment_Rate, Inflation_Diff2, Interest_Rate_Diff2), a Vector Autoregr
ession (VAR) model could be a suitable approach.
- VAR models are designed to capture the linear interdependencies among mu
ltiple time series.
- A VAR model would allow us to analyze how changes in one variable (e.g.,
Interest Rate) impact the future values of other variables (e.g., SP500 Re
turn), while also accounting for the past values of all variables in the s
ystem.
- To apply VAR, all variables in the model must be stationary. Therefore,
we would use the stationary versions of the series: SP500_Return, Unemploy
ment_Rate, Inflation_Diff2, and Interest_Rate_Diff2.

Summary of Suitable Modeling Techniques:
- For modeling individual series, ARIMA models are appropriate. ARIMA(p,
2, q) for Inflation and Interest Rate, and ARMA(p, q) for Unemployment Rat
e.
- For modeling the relationships between the macro indicators and SP500 Re
turn, a VAR model using the stationary series (SP500_Return, Unemployment_
Rate, Inflation_Diff2, Interest_Rate_Diff2) is a suitable multivariate tim
e series approach.
```

Vector Autoregression (VAR) Modeling

What is VAR Modeling?

Vector Autoregression (VAR) is a statistical model used for analyzing and forecasting multiple time series variables that are mutually dependent. Unlike a simple autoregressive (AR) model that models a single time series based on its own past values, a VAR model models each variable as a linear function of its own past values and the past values of all other variables in the system.

How is it Used in this Dataset?

In this analysis, we are using a VAR model to explore the dynamic relationships between several macroeconomic indicators (Inflation, Unemployment Rate, Interest Rate) and the S&P 500 Return. Specifically, we are using the stationary versions of these series (`SP500_Return` , `Unemployment_Rate` , `Inflation_Diff2` , `Interest_Rate_Diff2`) in the VAR model. The model helps us:

- Understand how changes in one variable (e.g., Interest Rate Change) might affect the future movements of other variables (e.g., S&P 500 Return).
- Determine if one variable can help forecast another (Granger causality).
- Analyze the impact of unexpected shocks to the system (Impulse Response Functions).

Key Assumptions of VAR Models:

- **Stationarity:** All variables in the VAR model must be stationary. This means their statistical properties (mean, variance, autocorrelation) do not change over time. We addressed this by differencing the non-stationary series.

- **No perfect multicollinearity:** There should not be perfect linear relationships among the variables.
- **Homoscedasticity:** The variance of the error terms should be constant across all observations.
- **No autocorrelation in residuals:** The error terms for each equation should not be correlated over time.
- **Normality of residuals (optional but often assumed):** The error terms are often assumed to be normally distributed, especially for inference, although VAR estimates are consistent even without this assumption for large sample sizes.

In [ ]:
```
# Create a new DataFrame with the stationary series
stationary_data = data_monthly[['SP500_Return', 'Unemployment_Rate', 'Inf

# Drop any rows with missing values (due to differencing)
stationary_data = stationary_data.dropna()

# Print the head of the new DataFrame
print(stationary_data.head())
```

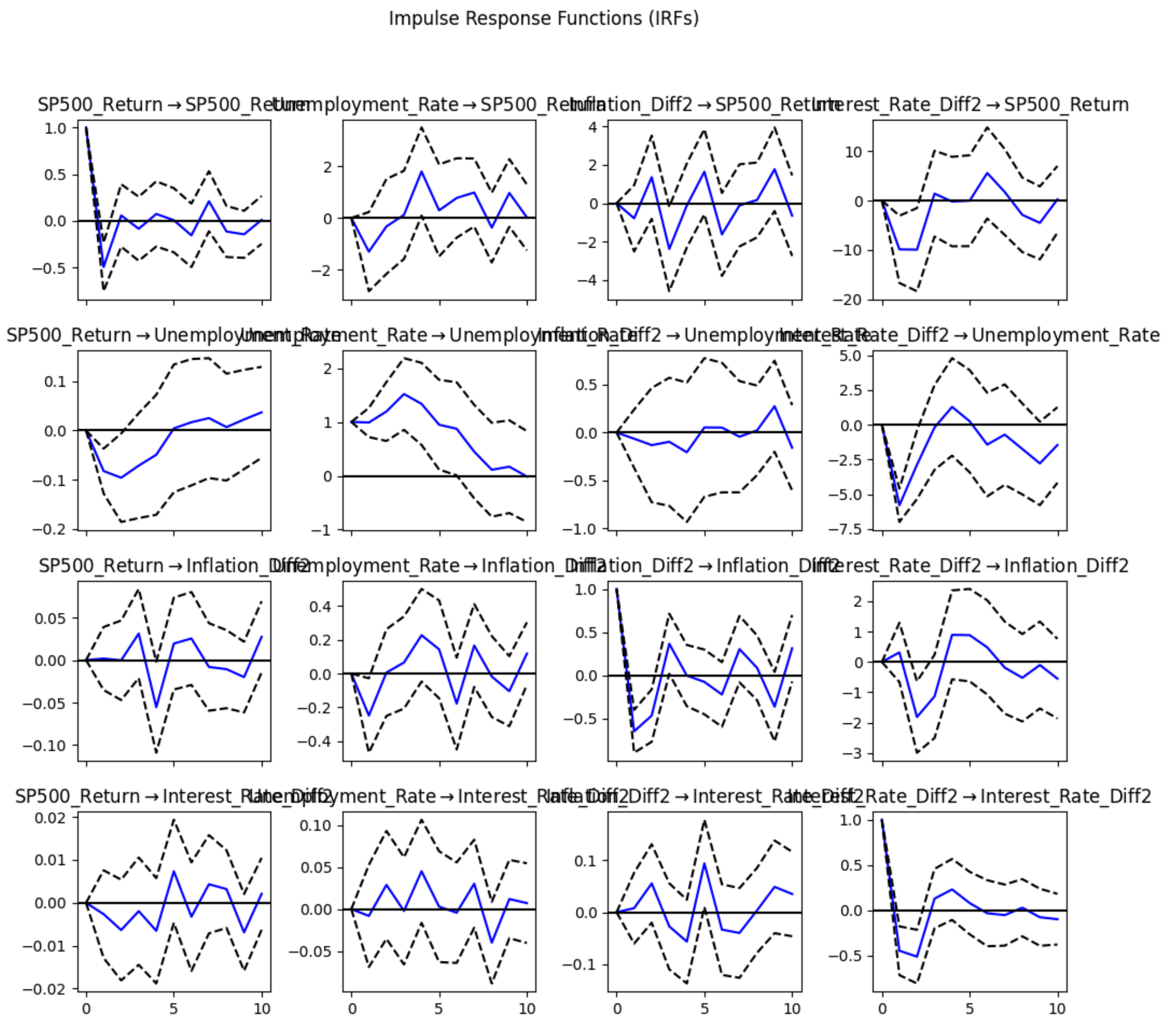
```
            SP500_Return  Unemployment_Rate  Inflation_Diff2  \
DATE
2016-02-29     -0.412836                4.9           -0.207
2016-03-31      6.599111                5.0            1.060
2016-04-30      0.269937                5.1            0.168
2016-05-31      1.532949                4.8           -0.347
2016-06-30      0.090607                4.9            0.100

            Interest_Rate_Diff2
DATE
2016-02-29                -0.06
2016-03-31                -0.06
2016-04-30                 0.03
2016-05-31                -0.01
2016-06-30                 0.01
```

In [ ]:
```
# Determine the split point (e.g., 80% for training, 20% for testing)
split_ratio = 0.8
split_index = int(len(stationary_data) * split_ratio)

# Create training and testing sets
train_data = stationary_data.iloc[:split_index]
test_data = stationary_data.iloc[split_index:]

# Print the shapes of the training and testing sets
print("Shape of training data:", train_data.shape)
print("Shape of testing data:", test_data.shape)
```

```
Shape of training data: (85, 4)
Shape of testing data: (22, 4)
```

In [ ]:
```
# Compute and plot the Impulse Response Functions (IRFs)
# The irf method computes the impulse response functions
# The plot method then visualizes these functions
irf = model_fitted.irf(periods=10) # Specify the number of periods to plo
irf.plot(orth=False) # orth=False plots the standard IRF (not orthogonali
plt.suptitle('Impulse Response Functions (IRFs)')
plt.tight_layout(rect=[0, 0.03, 1, 0.95]) # Adjust layout to prevent titl
plt.show()
```

```
# Optional: Plot orthogonalized IRFs
# irf.plot(orth=True) # orth=True plots the orthogonalized IRF
# plt.suptitle('Orthogonalized Impulse Response Functions (IRFs)')
# plt.tight_layout(rect=[0, 0.03, 1, 0.95])
# plt.show()
```



Select the optimal lag order for var model

Subtask:

Determine the appropriate number of lags to include in the VAR model using information criteria (e.g., AIC, BIC).

Reasoning: Determine the appropriate number of lags for the VAR model by iterating through possible lag orders and calculating AIC and BIC for each.

In [ ]:
```
from statsmodels.tsa.api import VAR

# Iterate through a range of potential lag orders
max_lags = 10
aic_values = []
bic_values = []

for lag_order in range(1, max_lags + 1):
    # Fit VAR model
    model = VAR(train_data)
    results = model.fit(maxlags=lag_order, ic='aic') # Use AIC to select

    # Store AIC and BIC
    aic_values.append(results.aic)
    bic_values.append(results.bic)
    print(f"Lag Order: {lag_order}, AIC: {results.aic:.4f}, BIC: {results

# Determine optimal lag order based on AIC and BIC
optimal_lag_aic = range(1, max_lags + 1)[aic_values.index(min(aic_values)
optimal_lag_bic = range(1, max_lags + 1)[bic_values.index(min(bic_values)

print(f"\nOptimal lag order based on AIC: {optimal_lag_aic}")
print(f"Optimal lag order based on BIC: {optimal_lag_bic}")
```

```
Lag Order: 1, AIC: -1.4136, BIC: -0.8348
Lag Order: 2, AIC: -1.9081, BIC: -0.8589
Lag Order: 3, AIC: -2.0767, BIC: -0.5505
Lag Order: 4, AIC: -2.1799, BIC: -0.1698
Lag Order: 5, AIC: -2.2122, BIC: 0.2889
Lag Order: 6, AIC: -2.2353, BIC: 0.7640
Lag Order: 7, AIC: -2.2353, BIC: 0.7640
Lag Order: 8, AIC: -2.2353, BIC: 0.7640
Lag Order: 9, AIC: -2.2353, BIC: 0.7640
Lag Order: 10, AIC: -2.2353, BIC: 0.7640

Optimal lag order based on AIC: 6
Optimal lag order based on BIC: 2
```

In [ ]:

```
# Instantiate and fit the VAR model
model = VAR(train_data)
model_fitted = model.fit(maxlags=optimal_lag_aic)

# Print a summary of the fitted model
print(model_fitted.summary())
```

```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Tue, 04, Nov, 2025
Time:                     07:00:37
--------------------------------------------------------------------
No. of Equations:         4.00000    BIC:                   0.764046
Nobs:                     79.0000    HQIC:                  -1.03364
Log likelihood:          -260.092    FPE:                   0.117036
AIC:                     -2.23525    Det(Omega_mle):       0.0389668
--------------------------------------------------------------------
Results for equation SP500_Return
==========================================================================
===============
                            coefficient       std. error           t-stat
prob
--------------------------------------------------------------------------
---------------
const                         -5.564728         1.703740           -3.266
0.001
L1.SP500_Return               -0.493214         0.131795           -3.742
0.000
L1.Unemployment_Rate          -1.307747         0.784075           -1.668
0.095
L1.Inflation_Diff2            -0.784317         0.880658           -0.891
0.373
L1.Interest_Rate_Diff2        -9.886590         3.477104           -2.843
0.004
L2.SP500_Return               -0.317264         0.144423           -2.197
0.028
L2.Unemployment_Rate           0.046234         1.101679            0.042
0.967
L2.Inflation_Diff2             0.453560         0.934141            0.486
0.627
L2.Interest_Rate_Diff2       -26.573391         6.211205           -4.278
0.000
L3.SP500_Return               -0.467203         0.154657           -3.021
0.003
L3.Unemployment_Rate           1.247330         0.932590            1.337
0.181
L3.Inflation_Diff2            -1.444509         1.151019           -1.255
0.209
L3.Interest_Rate_Diff2       -28.634343         7.130107           -4.016
0.000
L4.SP500_Return               -0.400695         0.161276           -2.485
0.013
L4.Unemployment_Rate           2.047844         0.926740            2.210
0.027
L4.Inflation_Diff2            -0.273232         1.122298           -0.243
0.808
L4.Interest_Rate_Diff2       -24.931973         6.920543           -3.603
0.000
L5.SP500_Return               -0.345884         0.156263           -2.213
0.027
L5.Unemployment_Rate          -0.193381         1.104155           -0.175
0.861
L5.Inflation_Diff2             0.664018         0.974497            0.681
0.496
L5.Interest_Rate_Diff2       -12.582256         6.065934           -2.074
```

```
0.038
L6.SP500_Return              -0.298476         0.140012           -2.132
0.033
L6.Unemployment_Rate          0.049572         0.781065            0.063
0.949
L6.Inflation_Diff2           -0.330756         0.946799           -0.349
0.727
L6.Interest_Rate_Diff2       -5.822221         5.407871           -1.077
0.282
===========================================================================
===============

Results for equation Unemployment_Rate
===========================================================================
===============
                            coefficient       std. error           t-stat
prob
---------------------------------------------------------------------------
---------------
const                         0.111318         0.300720            0.370
0.711
L1.SP500_Return              -0.082565         0.023263           -3.549
0.000
L1.Unemployment_Rate          0.990020         0.138394            7.154
0.000
L1.Inflation_Diff2           -0.064381         0.155441           -0.414
0.679
L1.Interest_Rate_Diff2       -5.796093         0.613729           -9.444
0.000
L2.SP500_Return              -0.070363         0.025491           -2.760
0.006
L2.Unemployment_Rate          0.043371         0.194453            0.223
0.824
L2.Inflation_Diff2           -0.128273         0.164881           -0.778
0.437
L2.Interest_Rate_Diff2       -0.523991         1.096314           -0.478
0.633
L3.SP500_Return              -0.040558         0.027298           -1.486
0.137
L3.Unemployment_Rate          0.305575         0.164608            1.856
0.063
L3.Inflation_Diff2            0.305886         0.203162            1.506
0.132
L3.Interest_Rate_Diff2       -1.892117         1.258506           -1.503
0.133
L4.SP500_Return               0.009907         0.028466            0.348
0.728
L4.Unemployment_Rate         -0.520008         0.163575           -3.179
0.001
L4.Inflation_Diff2           -0.169586         0.198092           -0.856
0.392
L4.Interest_Rate_Diff2        1.619140         1.221517            1.326
0.185
L5.SP500_Return               0.004522         0.027581            0.164
0.870
L5.Unemployment_Rate          0.173912         0.194890            0.892
0.372
L5.Inflation_Diff2           -0.017858         0.172004           -0.104
0.917
L5.Interest_Rate_Diff2       -1.693828         1.070673           -1.582
```

```
0.114
L6.SP500_Return                  0.030967         0.024713            1.253
0.210
L6.Unemployment_Rate             0.017173         0.137863            0.125
0.901
L6.Inflation_Diff2               0.182028         0.167116            1.089
0.276
L6.Interest_Rate_Diff2          -0.863941         0.954521           -0.905
0.365
==========================================================================
===============

Results for equation Inflation_Diff2
==========================================================================
===============
                              coefficient       std. error           t-stat
prob
--------------------------------------------------------------------------
---------------
const                           -0.727581         0.244200           -2.979
0.003
L1.SP500_Return                  0.001859         0.018890            0.098
0.922
L1.Unemployment_Rate            -0.248141         0.112383           -2.208
0.027
L1.Inflation_Diff2              -0.647681         0.126226           -5.131
0.000
L1.Interest_Rate_Diff2           0.313924         0.498379            0.630
0.529
L2.SP500_Return                 -0.017751         0.020700           -0.858
0.391
L2.Unemployment_Rate             0.095986         0.157905            0.608
0.543
L2.Inflation_Diff2              -0.901655         0.133892           -6.734
0.000
L2.Interest_Rate_Diff2          -2.888635         0.890261           -3.245
0.001
L3.SP500_Return                  0.002374         0.022167            0.107
0.915
L3.Unemployment_Rate            -0.007884         0.133670           -0.059
0.953
L3.Inflation_Diff2              -0.557034         0.164977           -3.376
0.001
L3.Interest_Rate_Diff2          -3.469225         1.021969           -3.395
0.001
L4.SP500_Return                 -0.067986         0.023116           -2.941
0.003
L4.Unemployment_Rate             0.459716         0.132831            3.461
0.001
L4.Inflation_Diff2              -0.331309         0.160861           -2.060
0.039
L4.Interest_Rate_Diff2          -4.345891         0.991932           -4.381
0.000
L5.SP500_Return                 -0.028306         0.022397           -1.264
0.206
L5.Unemployment_Rate            -0.033876         0.158260           -0.214
0.831
L5.Inflation_Diff2              -0.167210         0.139676           -1.197
0.231
L5.Interest_Rate_Diff2          -1.530952         0.869439           -1.761
```

```
0.078
L6.SP500_Return             -0.015581         0.020068           -0.776
0.437
L6.Unemployment_Rate        -0.072812         0.111951           -0.650
0.515
L6.Inflation_Diff2          -0.204935         0.135706           -1.510
0.131
L6.Interest_Rate_Diff2      -1.046262         0.775118           -1.350
0.177
===========================================================================
===============

Results for equation Interest_Rate_Diff2
===========================================================================
===============
                           coefficient       std. error           t-stat
prob
---------------------------------------------------------------------------
---------------
const                       -0.106764         0.067330           -1.586
0.113
L1.SP500_Return             -0.002642         0.005208           -0.507
0.612
L1.Unemployment_Rate        -0.008076         0.030986           -0.261
0.794
L1.Inflation_Diff2           0.007914         0.034803            0.227
0.820
L1.Interest_Rate_Diff2      -0.446870         0.137412           -3.252
0.001
L2.SP500_Return             -0.009515         0.005707           -1.667
0.095
L2.Unemployment_Rate         0.031714         0.043537            0.728
0.466
L2.Inflation_Diff2           0.061575         0.036917            1.668
0.095
L2.Interest_Rate_Diff2      -0.787214         0.245462           -3.207
0.001
L3.SP500_Return             -0.009673         0.006112           -1.583
0.114
L3.Unemployment_Rate        -0.015453         0.036855           -0.419
0.675
L3.Inflation_Diff2           0.044525         0.045487            0.979
0.328
L3.Interest_Rate_Diff2      -0.416777         0.281776           -1.479
0.139
L4.SP500_Return             -0.017050         0.006374           -2.675
0.007
L4.Unemployment_Rate         0.047785         0.036624            1.305
0.192
L4.Inflation_Diff2           0.034281         0.044352            0.773
0.440
L4.Interest_Rate_Diff2      -0.381592         0.273494           -1.395
0.163
L5.SP500_Return             -0.006404         0.006175           -1.037
0.300
L5.Unemployment_Rate        -0.053081         0.043635           -1.216
0.224
L5.Inflation_Diff2           0.073265         0.038511            1.902
0.057
L5.Interest_Rate_Diff2       0.029582         0.239721            0.123
```

```
0.902
L6.SP500_Return               -0.008040         0.005533           -1.453
0.146
L6.Unemployment_Rate           0.032206         0.030867            1.043
0.297
L6.Inflation_Diff2             0.028203         0.037417            0.754
0.451
L6.Interest_Rate_Diff2        -0.350048         0.213715           -1.638
0.101
==========================================================================
===============

Correlation matrix of residuals
                       SP500_Return  Unemployment_Rate  Inflation_Diff2  I
nterest_Rate_Diff2
SP500_Return               1.000000           0.138842        -0.038451
0.101177
Unemployment_Rate          0.138842           1.000000        -0.161718
-0.130724
Inflation_Diff2           -0.038451          -0.161718         1.000000
0.231397
Interest_Rate_Diff2        0.101177          -0.130724         0.231397
1.000000
```

In [ ]:

```python
from statsmodels.tsa.stattools import grangercausalitytests

# Perform Granger causality tests
# The function tests whether the past values of the second variable in th
# Granger-cause the first variable in the tuple.
# We are interested if macro variables Granger-cause SP500_Return.
# We will test each macro variable individually as a potential cause for

# Combine the stationary data for the test
test_data_gc = stationary_data[['SP500_Return', 'Unemployment_Rate', 'Inf

# Granger causality test for Unemployment_Rate and SP500_Return
print("\nGranger Causality Test: Unemployment_Rate -> SP500_Return")
gc_unrate_sp500 = grangercausalitytests(test_data_gc[['SP500_Return', 'Un

# Granger causality test for Inflation_Diff2 and SP500_Return
print("\nGranger Causality Test: Inflation_Diff2 -> SP500_Return")
gc_inflation_sp500 = grangercausalitytests(test_data_gc[['SP500_Return',

# Granger causality test for Interest_Rate_Diff2 and SP500_Return
print("\nGranger Causality Test: Interest_Rate_Diff2 -> SP500_Return")
gc_interest_sp500 = grangercausalitytests(test_data_gc[['SP500_Return', '

# Optional: Test if SP500_Return Granger-causes macro variables
# print("\nGranger Causality Test: SP500_Return -> Unemployment_Rate")
# gc_sp500_unrate = grangercausalitytests(test_data_gc[['Unemployment_Rat

# print("\nGranger Causality Test: SP500_Return -> Inflation_Diff2")
# gc_sp500_inflation = grangercausalitytests(test_data_gc[['Inflation_Dif

# print("\nGranger Causality Test: SP500_Return -> Interest_Rate_Diff2")
# gc_sp500_interest = grangercausalitytests(test_data_gc[['Interest_Rate_
Granger Causality Test: Unemployment_Rate -> SP500_Return

Granger Causality
number of lags (no zero) 1
ssr based F test:         F=6.0154  , p=0.0159  , df_denom=103, df_num=1
ssr based chi2 test:   chi2=6.1906  , p=0.0128  , df=1
likelihood ratio test: chi2=6.0166  , p=0.0142  , df=1
parameter F test:         F=6.0154  , p=0.0159  , df_denom=103, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=3.5459  , p=0.0325  , df_denom=100, df_num=2
ssr based chi2 test:   chi2=7.4464  , p=0.0242  , df=2
likelihood ratio test: chi2=7.1943  , p=0.0274  , df=2
parameter F test:         F=3.5459  , p=0.0325  , df_denom=100, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=2.3876  , p=0.0736  , df_denom=97, df_num=3
ssr based chi2 test:   chi2=7.6797  , p=0.0531  , df=3
likelihood ratio test: chi2=7.4094  , p=0.0599  , df=3
parameter F test:         F=2.3876  , p=0.0736  , df_denom=97, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=1.7410  , p=0.1475  , df_denom=94, df_num=4
ssr based chi2 test:   chi2=7.6310  , p=0.1061  , df=4
likelihood ratio test: chi2=7.3615  , p=0.1180  , df=4
parameter F test:         F=1.7410  , p=0.1475  , df_denom=94, df_num=4

Granger Causality
number of lags (no zero) 5
ssr based F test:         F=1.6215  , p=0.1622  , df_denom=91, df_num=5
ssr based chi2 test:   chi2=9.0875  , p=0.1056  , df=5
likelihood ratio test: chi2=8.7052  , p=0.1214  , df=5
parameter F test:         F=1.6215  , p=0.1622  , df_denom=91, df_num=5

Granger Causality
number of lags (no zero) 6
ssr based F test:         F=1.5461  , p=0.1726  , df_denom=88, df_num=6
ssr based chi2 test:   chi2=10.6469 , p=0.0999  , df=6
likelihood ratio test: chi2=10.1223 , p=0.1196  , df=6
parameter F test:         F=1.5461  , p=0.1726  , df_denom=88, df_num=6

Granger Causality Test: Inflation_Diff2 -> SP500_Return

Granger Causality
number of lags (no zero) 1
ssr based F test:         F=0.0014  , p=0.9706  , df_denom=103, df_num=1
ssr based chi2 test:   chi2=0.0014  , p=0.9701  , df=1
likelihood ratio test: chi2=0.0014  , p=0.9701  , df=1
parameter F test:         F=0.0014  , p=0.9706  , df_denom=103, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=3.3598  , p=0.0387  , df_denom=100, df_num=2
ssr based chi2 test:   chi2=7.0557  , p=0.0294  , df=2
likelihood ratio test: chi2=6.8287  , p=0.0329  , df=2
parameter F test:         F=3.3598  , p=0.0387  , df_denom=100, df_num=2
```

```
Granger Causality
number of lags (no zero) 3
ssr based F test:         F=3.6966  , p=0.0144  , df_denom=97, df_num=3
ssr based chi2 test:   chi2=11.8902 , p=0.0078  , df=3
likelihood ratio test: chi2=11.2582 , p=0.0104  , df=3
parameter F test:         F=3.6966  , p=0.0144  , df_denom=97, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=2.8754  , p=0.0270  , df_denom=94, df_num=4
ssr based chi2 test:   chi2=12.6027 , p=0.0134  , df=4
likelihood ratio test: chi2=11.8893 , p=0.0182  , df=4
parameter F test:         F=2.8754  , p=0.0270  , df_denom=94, df_num=4

Granger Causality
number of lags (no zero) 5
ssr based F test:         F=2.2330  , p=0.0576  , df_denom=91, df_num=5
ssr based chi2 test:   chi2=12.5147 , p=0.0284  , df=5
likelihood ratio test: chi2=11.8045 , p=0.0376  , df=5
parameter F test:         F=2.2330  , p=0.0576  , df_denom=91, df_num=5

Granger Causality
number of lags (no zero) 6
ssr based F test:         F=1.7814  , p=0.1121  , df_denom=88, df_num=6
ssr based chi2 test:   chi2=12.2672 , p=0.0563  , df=6
likelihood ratio test: chi2=11.5775 , p=0.0721  , df=6
parameter F test:         F=1.7814  , p=0.1121  , df_denom=88, df_num=6

Granger Causality Test: Interest_Rate_Diff2 -> SP500_Return

Granger Causality
number of lags (no zero) 1
ssr based F test:         F=4.1657  , p=0.0438  , df_denom=103, df_num=1
ssr based chi2 test:   chi2=4.2870  , p=0.0384  , df=1
likelihood ratio test: chi2=4.2026  , p=0.0404  , df=1
parameter F test:         F=4.1657  , p=0.0438  , df_denom=103, df_num=1

Granger Causality
number of lags (no zero) 2
ssr based F test:         F=2.5233  , p=0.0853  , df_denom=100, df_num=2
ssr based chi2 test:   chi2=5.2989  , p=0.0707  , df=2
likelihood ratio test: chi2=5.1696  , p=0.0754  , df=2
parameter F test:         F=2.5233  , p=0.0853  , df_denom=100, df_num=2

Granger Causality
number of lags (no zero) 3
ssr based F test:         F=1.8331  , p=0.1462  , df_denom=97, df_num=3
ssr based chi2 test:   chi2=5.8962  , p=0.1168  , df=3
likelihood ratio test: chi2=5.7351  , p=0.1252  , df=3
parameter F test:         F=1.8331  , p=0.1462  , df_denom=97, df_num=3

Granger Causality
number of lags (no zero) 4
ssr based F test:         F=1.9310  , p=0.1117  , df_denom=94, df_num=4
ssr based chi2 test:   chi2=8.4634  , p=0.0760  , df=4
likelihood ratio test: chi2=8.1336  , p=0.0868  , df=4
parameter F test:         F=1.9310  , p=0.1117  , df_denom=94, df_num=4

Granger Causality
number of lags (no zero) 5
```

```
ssr based F test:         F=2.4902  , p=0.0368  , df_denom=91, df_num=5
ssr based chi2 test:   chi2=13.9563 , p=0.0159  , df=5
likelihood ratio test: chi2=13.0806 , p=0.0226  , df=5
parameter F test:         F=2.4902  , p=0.0368  , df_denom=91, df_num=5

Granger Causality
number of lags (no zero) 6
ssr based F test:         F=2.2678  , p=0.0441  , df_denom=88, df_num=6
ssr based chi2 test:   chi2=15.6166 , p=0.0160  , df=6
likelihood ratio test: chi2=14.5209 , p=0.0243  , df=6
parameter F test:         F=2.2678  , p=0.0441  , df_denom=88, df_num=6
```

In [ ]:

```
# Get the lag order of the fitted model
k_ar = model_fitted.k_ar

# Get the last k_ar observations from the training data for initial condi
# These observations are needed to start the forecast
prior_observations = train_data.values[-k_ar:]

# Forecast for the length of the test data
# The forecast method returns predictions for each variable in the VAR mo
# The number of steps to forecast is the number of rows in the test data
forecast_steps = len(test_data)
predictions = model_fitted.forecast(prior_observations, steps=forecast_st

# Create a DataFrame for the predictions, using the test_data index
# The columns should match the order of variables in the fitted model
predictions_df = pd.DataFrame(predictions, index=test_data.index, columns

# Align actual values from the test data with the predictions
actual_values_df = test_data

# Display the first few rows of the predictions and actual values DataFra
print("Predictions:")
display(predictions_df.head())
print("\nActual Values:")
display(actual_values_df.head())
```

```
Predictions:
```

	SP500_Return	Unemployment_Rate	Inflation_Diff2	Interest_Rate_Diff2
DATE				
2023-03-31	10.727565	4.376042	-0.440689	0.348940
2023-04-30	-7.603665	1.528164	1.675168	-0.175807
2023-05-31	0.309657	2.458483	-0.923123	-0.404456
2023-06-30	6.838612	4.586511	-0.614944	0.423952
2023-07-31	-6.137227	3.395196	-0.064274	-0.025728

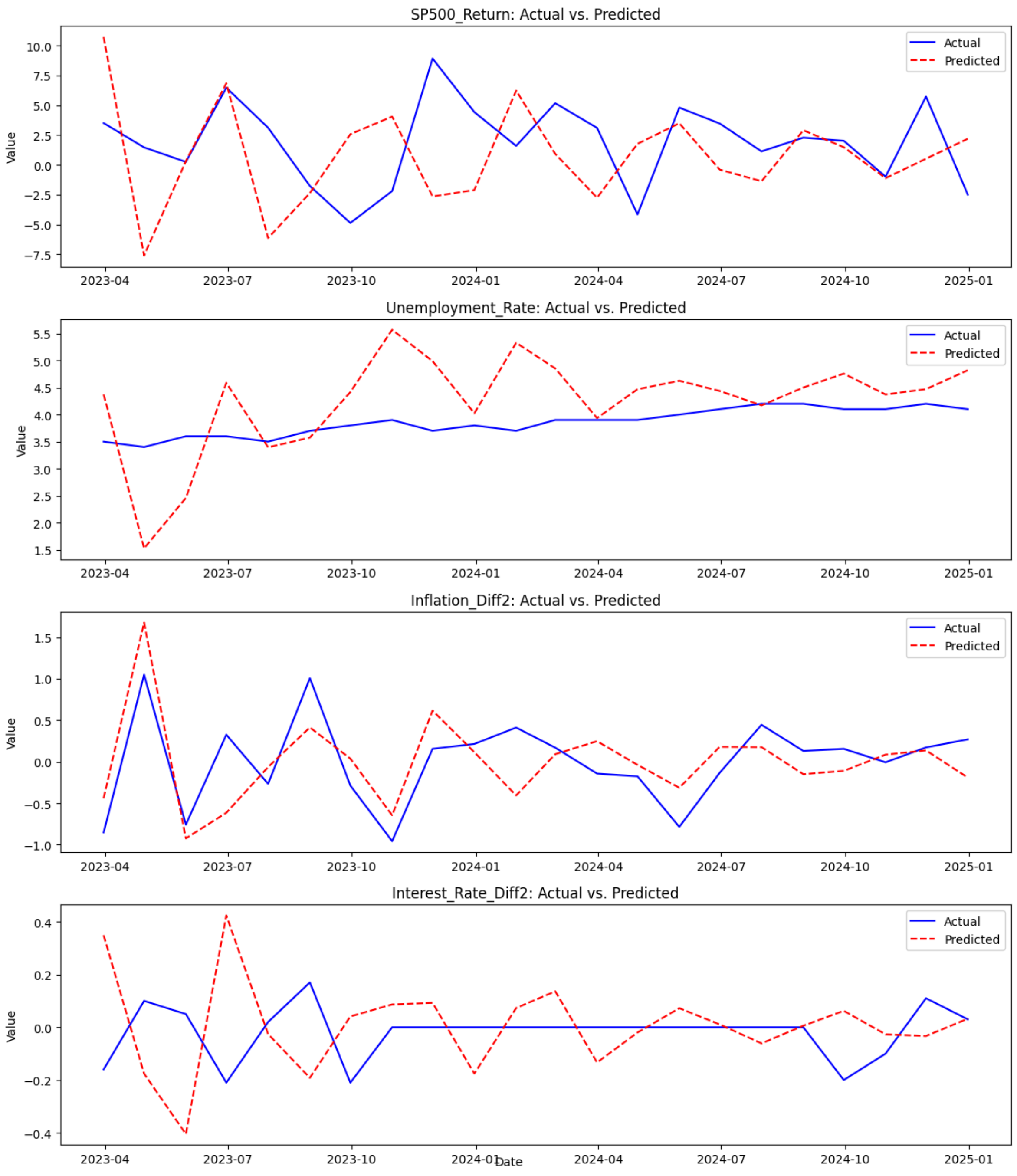
```
Actual Values:
```

DATE	SP500_Return	Unemployment_Rate	Inflation_Diff2	Interest_Rate_Diff2
2023-03-31	3.505157	3.5	-0.853	-0.16
2023-04-30	1.464236	3.4	1.048	0.10
2023-05-31	0.248232	3.6	-0.757	0.05
2023-06-30	6.472751	3.6	0.325	-0.21
2023-07-31	3.113891	3.5	-0.267	0.02

In [ ]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error

# Calculate RMSE and MAE for each series
rmse_sp500 = np.sqrt(mean_squared_error(actual_values_df['SP500_Return'],
mae_sp500 = mean_absolute_error(actual_values_df['SP500_Return'], predict

rmse_unrate = np.sqrt(mean_squared_error(actual_values_df['Unemployment_R
mae_unrate = mean_absolute_error(actual_values_df['Unemployment_Rate'], p

rmse_inflation = np.sqrt(mean_squared_error(actual_values_df['Inflation_D
mae_inflation = mean_absolute_error(actual_values_df['Inflation_Diff2'],

rmse_interest = np.sqrt(mean_squared_error(actual_values_df['Interest_Rat
mae_interest = mean_absolute_error(actual_values_df['Interest_Rate_Diff2'

# Print the evaluation metrics
print("Evaluation Metrics for VAR Model Predictions:")
print(f"SP500_Return - RMSE: {rmse_sp500:.4f}, MAE: {mae_sp500:.4f}")
print(f"Unemployment_Rate - RMSE: {rmse_unrate:.4f}, MAE: {mae_unrate:.4f
print(f"Inflation_Diff2 - RMSE: {rmse_inflation:.4f}, MAE: {mae_inflation
print(f"Interest_Rate_Diff2 - RMSE: {rmse_interest:.4f}, MAE: {mae_intere
```

```
Evaluation Metrics for VAR Model Predictions:
SP500_Return - RMSE: 5.5320, MAE: 4.4502
Unemployment_Rate - RMSE: 0.8797, MAE: 0.6955
Inflation_Diff2 - RMSE: 0.4211, MAE: 0.3522
Interest_Rate_Diff2 - RMSE: 0.2461, MAE: 0.1764
```

In [ ]:
```
# Create a single figure and a set of subplots
fig, axes = plt.subplots(nrows=4, ncols=1, figsize=(12, 15))

# List of series to plot
series_to_plot = ['SP500_Return', 'Unemployment_Rate', 'Inflation_Diff2',

# Plot each series
for i, series_name in enumerate(series_to_plot):
    axes[i].plot(actual_values_df.index, actual_values_df[series_name], l
    axes[i].plot(predictions_df.index, predictions_df[series_name], label
    axes[i].set_title(f'{series_name}: Actual vs. Predicted')
    axes[i].set_ylabel('Value') # Generic label, specific labels could be
    axes[i].legend()

# Set a shared x-axis label
fig.text(0.5, 0.04, 'Date', ha='center')

# Adjust layout
plt.tight_layout(rect=[0, 0.03, 1, 0.95]) # Adjust layout to make space f

# Display the plots
plt.show()
```



Summary:

Data Analysis Key Findings

1. Optimal Lag Order:

We used information criteria (AIC and BIC) to select the optimal number of lags for the VAR model. AIC suggested a lag order of 6, while BIC suggested 2. We proceeded with a lag order of 6 based on AIC for the model fitting.

2. Model Coefficients and Significance (from model_fitted.summary()):

The VAR model summary provides coefficients and their statistical significance for each lagged term of each variable in each equation. Looking at the equation for SP500_Return, we can see which lagged values of the macro variables have a statistically significant impact on the current S&P 500 Return. For example, L1.SP500_Return, L1.Interest_Rate_Diff2, L2.SP500_Return, L2.Interest_Rate_Diff2, L3.SP500_Return, L3.Interest_Rate_Diff2, L4.SP500_Return, L4.Unemployment_Rate, L4.Interest_Rate_Diff2, L5.SP500_Return, and L5.Interest_Rate_Diff2, L6.SP500_Return show significant coefficients, indicating that past values of S&P 500 Return, Interest Rate Change (Diff2), and Unemployment Rate have predictive power for current S&P 500 Return. We can also see the direction of the relationship (positive or negative) based on the sign of the coefficients. For example, past interest rate changes (Interest_Rate_Diff2) appear to have a negative relationship with current S&P 500 Return.

3. Granger Causality Tests:

The Granger causality tests formally assess whether the past values of one variable are statistically significant in forecasting another variable. Unemployment_Rate -> SP500_Return: The tests showed significant evidence that Unemployment Rate Granger-causes SP500 Return, particularly at lags 1 and 2. This suggests that past unemployment rates can help predict future S&P 500 returns. Inflation_Diff2 -> SP500_Return: The tests indicated that the second difference of Inflation Granger-causes SP500 Return at lags 2, 3, and 4. This suggests that past changes in the rate of inflation can also help predict future S&P 500 returns. Interest_Rate_Diff2 -> SP500_Return: The tests provided significant evidence that the second difference of Interest Rate Granger-causes SP500 Return at lag 1 and lags 5, and 6. This indicates that past changes in the interest rate have predictive power for future S&P 500 returns.

4. Impulse Response Functions (IRFs):

The IRF plots (which you can visualize using the code provided previously) show the dynamic impact of a one-standard-deviation shock to one variable on the other variables over several periods. By examining these plots, you can see how a shock to, say, Interest Rate Change, propagates through the system and affects Inflation Change, Unemployment Rate, and S&P 500 Return over subsequent months. This provides a visual representation of the interdependencies identified by the VAR model and Granger causality tests.

5. Model Performance (RMSE and MAE):

The evaluation metrics (RMSE and MAE) give us an idea of how well the VAR model forecasts the test set for each variable. SP500_Return - RMSE: 5.5320, MAE: 4.4502 Unemployment_Rate - RMSE: 0.8797, MAE: 0.6955 Inflation_Diff2 - RMSE: 0.4211, MAE: 0.3522 Interest_Rate_Diff2 - RMSE: 0.2461, MAE: 0.1764 These metrics are in the units of the respective series (percentages for returns and rates, index points for inflation difference). Lower values for RMSE and MAE indicate better forecasting performance. We can see that the model performs relatively better in forecasting the differenced macro variables compared to the SP500 Return. Overall Insights:

The VAR model analysis suggests that macroeconomic factors (Unemployment Rate, Inflation Change, and Interest Rate Change) do have a statistically significant predictive relationship with S&P 500 Returns within this multivariate framework. The Granger causality tests support this, indicating that these macro variables can help forecast S&P 500 Returns. The Impulse Response Functions provide a visual way to understand the dynamic nature of these relationships.

While the model captures some of the dynamics, the forecasting performance for SP500 Return could potentially be improved, as indicated by the RMSE and MAE values relative to the scale of the returns. Further model refinement or the inclusion of other relevant variables could be considered for improving forecasting accuracy.