SUBJECT : DSC 550 TERM PROJECT - TOOLS FOR INVESTOR'S - S&P 500 AND DOW 30 INDICES.

NAME : ANBUSELVAN MAHALINGAM

SUBJECT : DSC 550 TERM PROJECT MILESTONE 1 - TOOLS FOR INVESTOR'S - S&P 500 AND DOW 30 INDICES.

Term Project Milestone 1: Data Selection and EDA

Business Problem Overview:

The USA stock market, is a complex and dynamic system, influenced by a myriad of factors ranging from economic indicators to global events. Investors and financial analysts continually seek tools and methods to better understand and predict market movements to make informed investment decisions.

This project focuses on the S&P 500 and Dow Jones Industrial Average (Dow 30), two of the most prominent stock indices in the United States, representing a broad spectrum of the market's health.

Goals and objective of the project:

1. To Develop a predictive model to forecast the future movements of the S&P 500 and Dow 30 indices(future closing prices of the S&P 500 and Dow 30 indices).
2. To provide a tool that aids investors and analysts in anticipating market changes, thereby facilitating better investment strategy formulation.

Scope:

For the project we are considering only two indices for analysis and the same can be performed for relating multiple other indices as an expantion to this project.

Data Selection

For this project, we focus on the S&P 500 and Dow Jones Industrial Average (Dow 30) indices. The dataset encompasses 5 years of historical data. Reference: Yahoofinance.com for obtainng these indices historical data.

S&P 500: https://finance.yahoo.com/quote/%5EGSPC/history?p=%255EGSPC

The Standard and Poor's 500, or simply the S&P 500,is a stock market index tracking the stock performance of 500 of the largest companies listed on stock exchanges in the United States. It is one of the most commonly followed equity indices and includes approximately 80% of the total market capitalization of U.S. public companies.

Dow 30: https://finance.yahoo.com/quote/%5EDJI/history?p=%255EDJI

The Dow Jones Industrial Average, Dow Jones, or simply the Dow, is a stock market index of 30 prominent companies listed on stock exchanges in the United States.

In [1]:
```
# EDA - to explore and uderstad the initial data

#To load the data into dataframes and merge it together.

import pandas as pd

# To load the datasets
sp500_data_df = pd.read_csv('S&P500-5yearData.csv')
dow30_data_df = pd.read_csv('DOW30-5yearData.csv')

# To convert 'Date' columns to datetime
sp500_data_df['Date'] = pd.to_datetime(sp500_data_df['Date'])
dow30_data_df['Date'] = pd.to_datetime(dow30_data_df['Date'])

# Merge the datasets on 'Date' and rename fields with suffix.
merged_data = pd.merge(sp500_data_df, dow30_data_df, on='Date', suffixes=('_sp500',
merged_data.head(n=5)
```

Out[1]:

	Date	Open_sp500	High_sp500	Low_sp500	Close_sp500	Adj Close_sp500	Volume_sp500
0	2019-01-22	2657.879883	2657.879883	2617.270020	2632.899902	2632.899902	3923950000
1	2019-01-23	2643.479980	2653.189941	2612.860107	2638.699951	2638.699951	3358770000
2	2019-01-24	2638.840088	2647.199951	2627.010010	2642.330078	2642.330078	3449230000
3	2019-01-25	2657.439941	2672.379883	2657.330078	2664.760010	2664.760010	3821000000
4	2019-01-28	2644.969971	2644.969971	2624.060059	2643.850098	2643.850098	3630820000

In [2]:
```
# to get overall info about the dataset icnluding the datatype and null value on th
merged_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1258 entries, 0 to 1257
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Date             1258 non-null   datetime64[ns]
 1   Open_sp500       1258 non-null   float64
 2   High_sp500       1258 non-null   float64
 3   Low_sp500        1258 non-null   float64
 4   Close_sp500      1258 non-null   float64
 5   Adj Close_sp500  1258 non-null   float64
 6   Volume_sp500     1258 non-null   int64
 7   Open_dow30       1258 non-null   float64
 8   High_dow30       1258 non-null   float64
 9   Low_dow30        1258 non-null   float64
 10  Close_dow30      1258 non-null   float64
 11  Adj Close_dow30  1258 non-null   float64
 12  Volume_dow30     1258 non-null   int64
dtypes: datetime64[ns](1), float64(10), int64(2)
memory usage: 137.6 KB
```

In [3]:
```
# To get comprehensive statistical overview of dataset
merged_data.describe()
```

Out[3]:

	Open_sp500	High_sp500	Low_sp500	Close_sp500	Adj Close_sp500	Volume_sp500	O
count	1258.000000	1258.000000	1258.000000	1258.000000	1258.000000	1.258000e+03	
mean	3777.426413	3800.220382	3753.356557	3778.343116	3778.343116	4.306184e+09	3(
std	630.107259	631.949095	628.910413	630.461755	630.461755	1.064672e+09	
min	2290.709961	2300.729980	2191.860107	2237.399902	2237.399902	1.296530e+09	1(
25%	3152.619995	3182.530029	3137.015014	3154.027405	3154.027405	3.662092e+09	2(
50%	3917.295044	3949.779908	3891.505005	3918.785034	3918.785034	4.048580e+09	32
75%	4329.079956	4361.334961	4291.397705	4328.559814	4328.559814	4.687198e+09	3(
max	4804.509766	4842.069824	4785.870117	4839.810059	4839.810059	9.976520e+09	3(

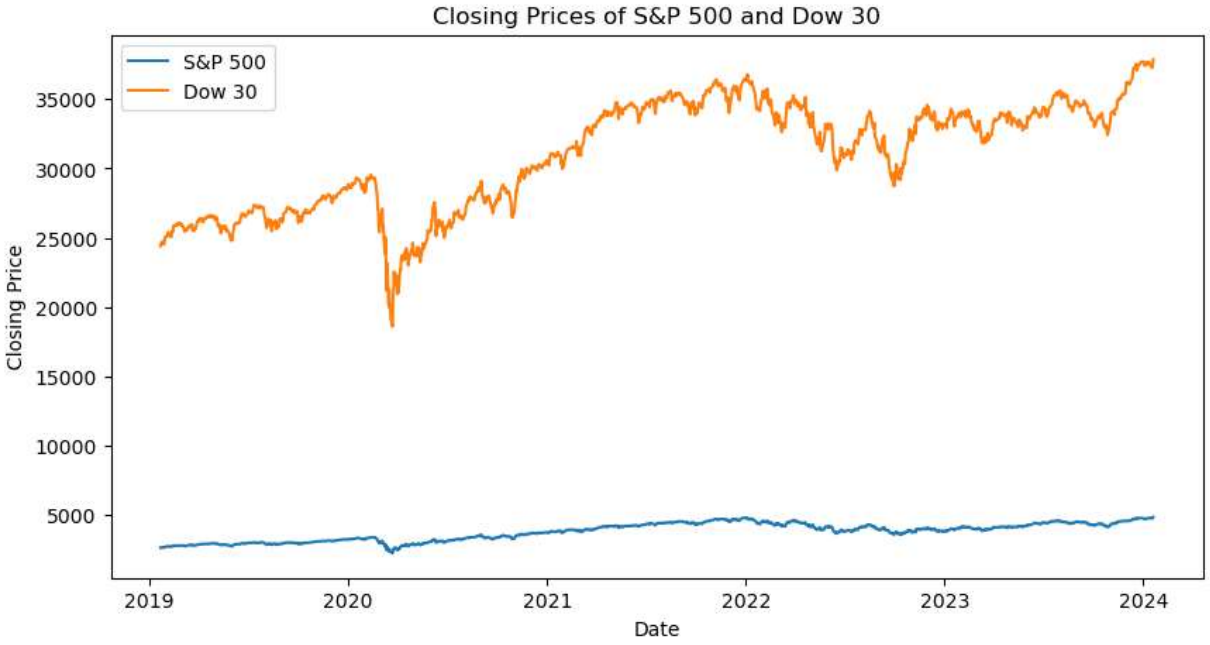
S&P 500 the average closing price is around 3778, and the the mean volume is approximately 4.31 billion, reflecting the average trading activity.

Dow 30 the average closing price is approximately 31000,and the mean volume is around 343 million, suggesting lower trading volumes compared to the S&P 500.

In [4]:
```
# To perform trend analysis of closing prices
# import matplotlib libary
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 5)) # to define the size for the isualization
plt.plot(merged_data['Date'], merged_data['Close_sp500'], label='S&P 500')
```

```
plt.plot(merged_data['Date'], merged_data['Close_dow30'], label='Dow 30')
plt.title('Closing Prices of S&P 500 and Dow 30') # to add title for the isualizati
plt.xlabel('Date') # to add x axis label
plt.ylabel('Closing Price') # to add y - axis label
plt.legend() # To add legends for user information
plt.show()
```



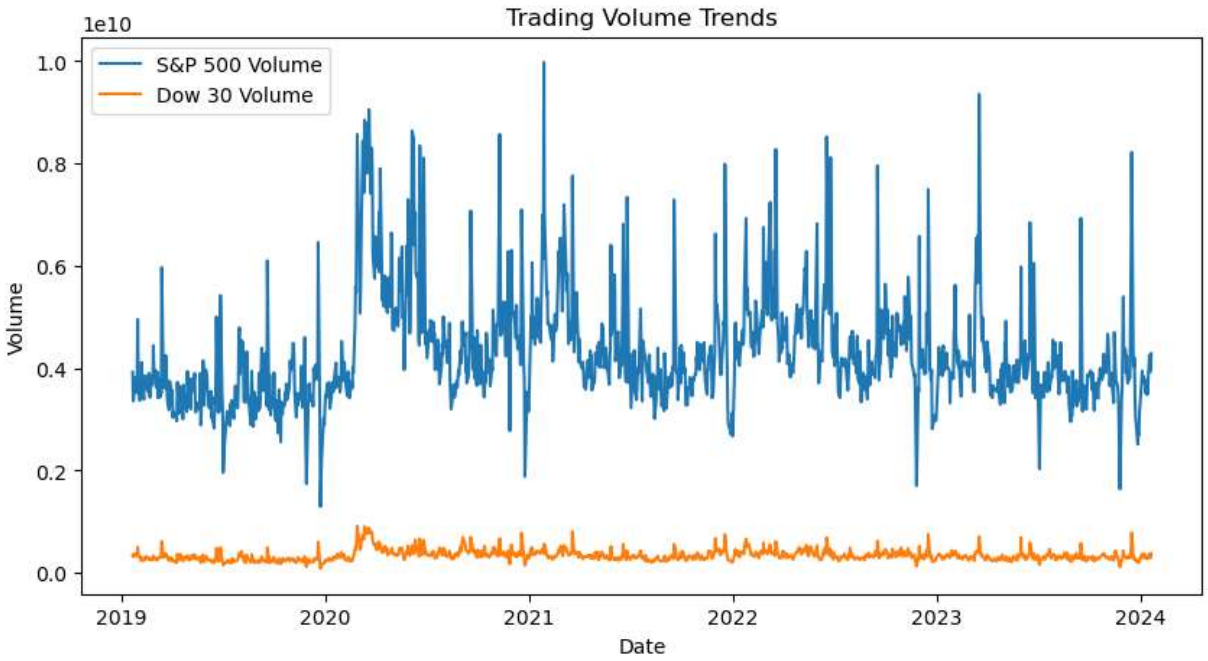
The long-term trend for both indices S&P 500 and DOW 30 has been positive over the last five years. This information can be crucial for long-term investors, suggesting that despite periods of decline, the overall trajectory has been one of growth.

There was a slight down trend and volatility around the year 2020, and both the indices piked up later and gained considerale gowth. The 2020 down trend could be the reason of political reason, or pandamic period reason.

The trends between the two indices seem to move in a correlated fashion, rising and falling together,which suggests a potential shared market drivers.

In [5]:
```
#to perform Volume Trend Analysis against Date

plt.figure(figsize=(10, 5)) # to define the size for the visualization
plt.plot(merged_data['Date'], merged_data['Volume_sp500'], label='S&P 500 Volume')
plt.plot(merged_data['Date'], merged_data['Volume_dow30'], label='Dow 30 Volume')
plt.title('Trading Volume Trends') # to add title for the visualization
plt.xlabel('Date') # to add x axis label
plt.ylabel('Volume') # to ad y - axis label
plt.legend()  # To add legends for user information
plt.show()
```



The S&P 500 shows significantly higher trading volumes compared to the Dow 30.

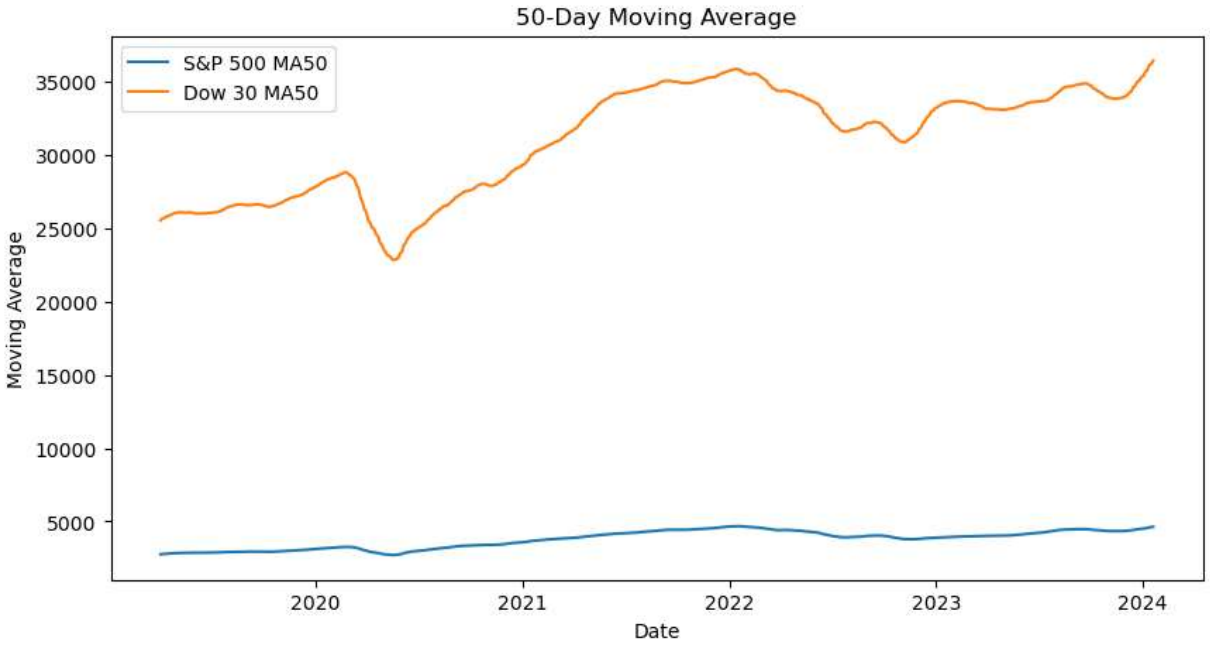
The S&P 500's volume displays spikes volatility in volume this depicts the investor sentiments about buy ad sell. The trading volume for the Dow 30 appears relatively consistent with fewer spikes, which may indicate less frequent trading or lower liquidity in comparison to the S&P 500.

Around 2020 both the indices had some down trend in volume, this can be oelated with the price trend visualization.

In [6]:
```
# To calculate 50 day moving averages
# 50-day moving average is a key indicator used by traders and investors to underst

# add ew eleet to the dataframe  by adding mean for the window of 50 day period
merged_data['MA50_sp500'] = merged_data['Close_sp500'].rolling(window=50).mean()
merged_data['MA50_dow30'] = merged_data['Close_dow30'].rolling(window=50).mean()

plt.figure(figsize=(10, 5))
plt.plot(merged_data['Date'], merged_data['MA50_sp500'], label='S&P 500 MA50')
plt.plot(merged_data['Date'], merged_data['MA50_dow30'], label='Dow 30 MA50')
plt.title('50-Day Moving Average')
plt.xlabel('Date')
plt.ylabel('Moving Average')
plt.legend()
plt.show()
```

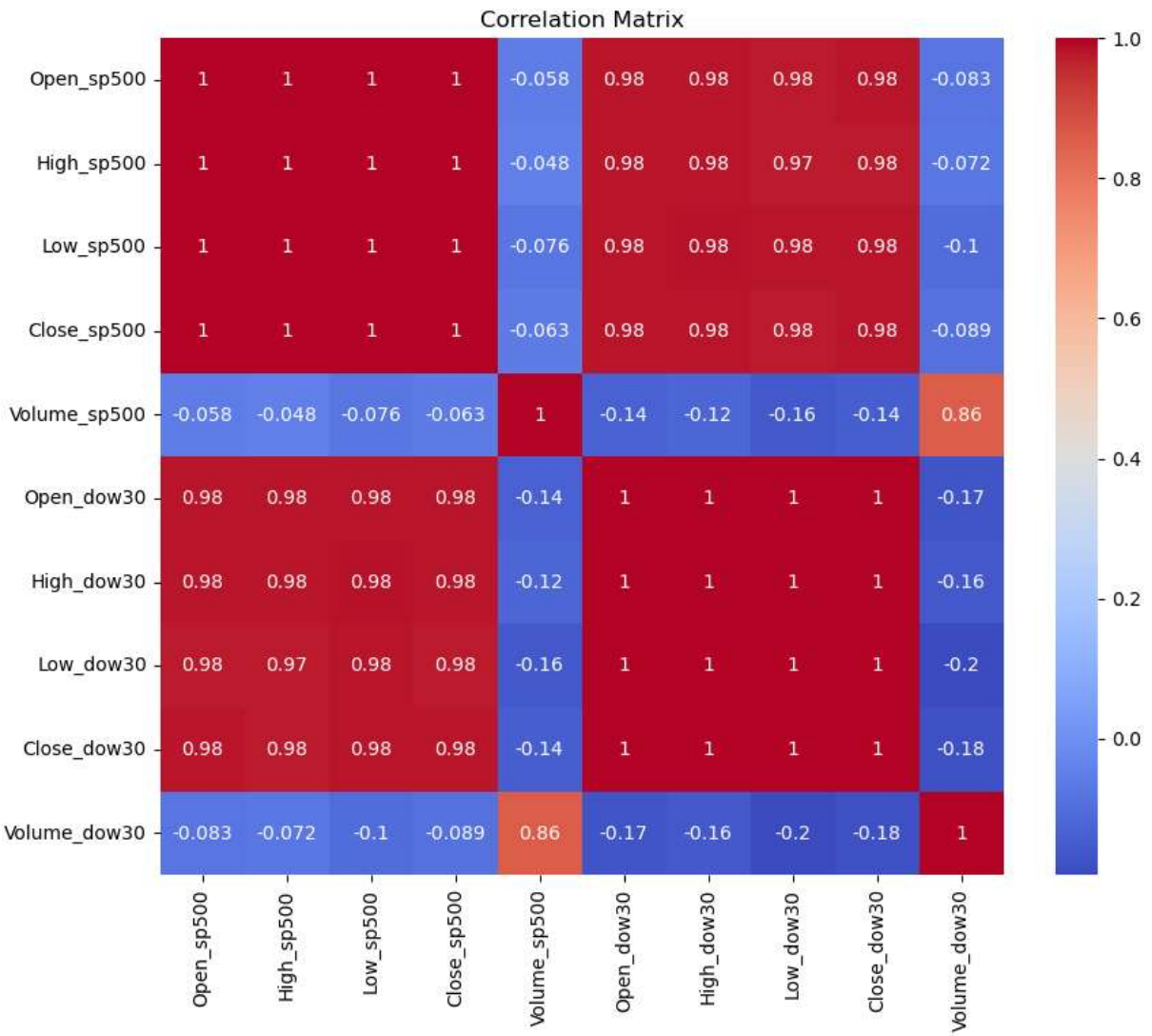


The MA50 of the Dow 30 appears to be at a higher level compared to the S&P 500 same as the closin price trend analysis.

In [7]:
```
# To perfor correlation Analysis on the iices.
import seaborn as sns

# To select columns for correlation analysis
correlation_data = merged_data[['Open_sp500', 'High_sp500', 'Low_sp500', 'Close_sp5
                               'Open_dow30', 'High_dow30', 'Low_dow30', 'Close_dow
#create coelatio matrix
correlation_matrix = correlation_data.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm') # heatmap representing
plt.title('Correlation Matrix')
plt.show()
```



In [ ]:

Co-relation between the vaiables. here -1 (perfect negative correlation) to +1 (perfect positive correlation)

The opening, high, low, and closing prices of the S&P 500 and Dow 30 also exhibit a high degree of positive correlation with each other.

The opening, high, low, and closing prices of the S&P 500 and Dow 30 also exhibit a high degree of positive correlation with each other.

Conlusion

The EDA has provided a strong foundation for developing predictive models, formulating investment strategies, and conducting further in-depth analyses.

The S&P 500 and Dow 30,show similar price movement patterns indicating that they may be similarly affected by overall market conditions.

The consistent upward trend in their moving averages across the analyzed period signifies a robust market performance, which can instill confidence in long-term market growth despite short-term fluctuations.

The 50-day moving averages provided a perspective of the market trends, filtering out daily fluctuations. Both indices displayed a relatively stable upward movement in their respective MA50s.

The opening, high, low, and closing prices of the S&P 500 and Dow 30 also exhibit a high degree of positive correlation with each other

Based on the Exploratoy data analysis conclusion, we can conduct futher analysis on getting investor sentiment, and develop a robust machine learning model to predit the closing price for the indices in upoming project milestones.

/==========

SUBJECT : DSC 550 TERM PROJECT MILESTONE 2

```
1.Dropped un-wanted features/columns from the dataset - 'Adj
Close_sp500' & Adj Close_dow30 are not useful for our model

2.Added new features weekly. monthly, quaterly moving average
closin cost
    The 7-Day Moving Average - short-term cost as mean price
    The 30-Day Moving Average - medium-term cost as mean price
    The 90-Day Moving Average - longer-term cost as mean price

3.Performed visual trend analysis for short, medium and long term
closing cost to predict the trend Bull/Bear.
```

/==========

Data Preparation Data preparation is critical step as this directly impacts accuracy, realiability and perormance of the model.i this milestone 2 we will be preparing dataset for time series predition model and trend analysis .

we will make ue of merged_data - merged data frame from milestone 1

Data Preparation - Remove un wanted or less used column/features

```
The adjusted closing price is a way of measuring the value of a
stock after accounting for any corporate actions that may have
affected its price, such as dividends, splits, or mergers.The
adjusted closing price is a way of measuring the value of a stock
after accounting for any corporate actions that may have affected
its price, such as dividends, splits, or mergers.
```

Here the Adj Close_sp500 & Adj Close_dow30 not much useful assuming the divident and maket decisions related to adjusted closign price and histoical impact will continue same in future.

In [8]:
```
merged_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1258 entries, 0 to 1257
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype         
---  ------           --------------  -----         
 0   Date             1258 non-null   datetime64[ns]
 1   Open_sp500       1258 non-null   float64       
 2   High_sp500       1258 non-null   float64       
 3   Low_sp500        1258 non-null   float64       
 4   Close_sp500      1258 non-null   float64       
 5   Adj Close_sp500  1258 non-null   float64       
 6   Volume_sp500     1258 non-null   int64         
 7   Open_dow30       1258 non-null   float64       
 8   High_dow30       1258 non-null   float64       
 9   Low_dow30        1258 non-null   float64       
 10  Close_dow30      1258 non-null   float64       
 11  Adj Close_dow30  1258 non-null   float64       
 12  Volume_dow30     1258 non-null   int64         
 13  MA50_sp500       1209 non-null   float64       
 14  MA50_dow30       1209 non-null   float64       
dtypes: datetime64[ns](1), float64(12), int64(2)
memory usage: 157.2 KB
```

In [9]:
```
# to remove column/feature - Adj_Close_sp500 & Adj_Close_dow30

# Assuming 'Adj Close_sp500' & Adj Close_dow30 are not useful for our model
merged_data.drop(columns=['Adj Close_sp500', 'Adj Close_dow30'], inplace=True)
```

In [10]:
```
# Valdiation - after dropping the un-wanted features
merged_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1258 entries, 0 to 1257
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   Date           1258 non-null   datetime64[ns]
 1   Open_sp500     1258 non-null   float64       
 2   High_sp500     1258 non-null   float64       
 3   Low_sp500      1258 non-null   float64       
 4   Close_sp500    1258 non-null   float64       
 5   Volume_sp500   1258 non-null   int64         
 6   Open_dow30     1258 non-null   float64       
 7   High_dow30     1258 non-null   float64       
 8   Low_dow30      1258 non-null   float64       
 9   Close_dow30    1258 non-null   float64       
 10  Volume_dow30   1258 non-null   int64         
 11  MA50_sp500     1209 non-null   float64       
 12  MA50_dow30     1209 non-null   float64       
dtypes: datetime64[ns](1), float64(10), int64(2)
memory usage: 137.6 KB
```

Creating new Features for moving average

A moving average is a technical indicator that shows the average price of a stock over a specific period of time. It helps to smooth out the price fluctuations and identify the trend direction, we already ahve 50 day movign aearage features MA50_sp500 & MA50_dow30.

we will be creating new features Moving Avearage for wekly, monthly and quaterly timeframe.This is to to help our trend and predition model.

In [11]:
```
# To create new features moving average for hot term (weekly and monthly roughly 30
# and logn tern moving average 90 days Quaterly

# To calculate moving averages for the S&P 500 index
merged_data['MA_7_sp500'] = merged_data['Close_sp500'].rolling(window=7).mean()
merged_data['MA_30_sp500'] = merged_data['Close_sp500'].rolling(window=30).mean()
merged_data['MA_90_sp500'] = merged_data['Close_sp500'].rolling(window=90).mean()

# To calculate moving averages for the Dow 30 index
merged_data['MA_7_dow30'] = merged_data['Close_dow30'].rolling(window=7).mean()
merged_data['MA_30_dow30'] = merged_data['Close_dow30'].rolling(window=30).mean()
merged_data['MA_90_dow30'] = merged_data['Close_dow30'].rolling(window=90).mean()
```

In [12]:
```
# Valdiation after the Featue ceation
merged_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1258 entries, 0 to 1257
Data columns (total 19 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   Date           1258 non-null   datetime64[ns]
 1   Open_sp500     1258 non-null   float64       
 2   High_sp500     1258 non-null   float64       
 3   Low_sp500      1258 non-null   float64       
 4   Close_sp500    1258 non-null   float64       
 5   Volume_sp500   1258 non-null   int64         
 6   Open_dow30     1258 non-null   float64       
 7   High_dow30     1258 non-null   float64       
 8   Low_dow30      1258 non-null   float64       
 9   Close_dow30    1258 non-null   float64       
 10  Volume_dow30   1258 non-null   int64         
 11  MA50_sp500     1209 non-null   float64       
 12  MA50_dow30     1209 non-null   float64       
 13  MA_7_sp500     1252 non-null   float64       
 14  MA_30_sp500    1229 non-null   float64       
 15  MA_90_sp500    1169 non-null   float64       
 16  MA_7_dow30     1252 non-null   float64       
 17  MA_30_dow30    1229 non-null   float64       
 18  MA_90_dow30    1169 non-null   float64       
dtypes: datetime64[ns](1), float64(16), int64(2)
memory usage: 196.6 KB
```

In [13]:

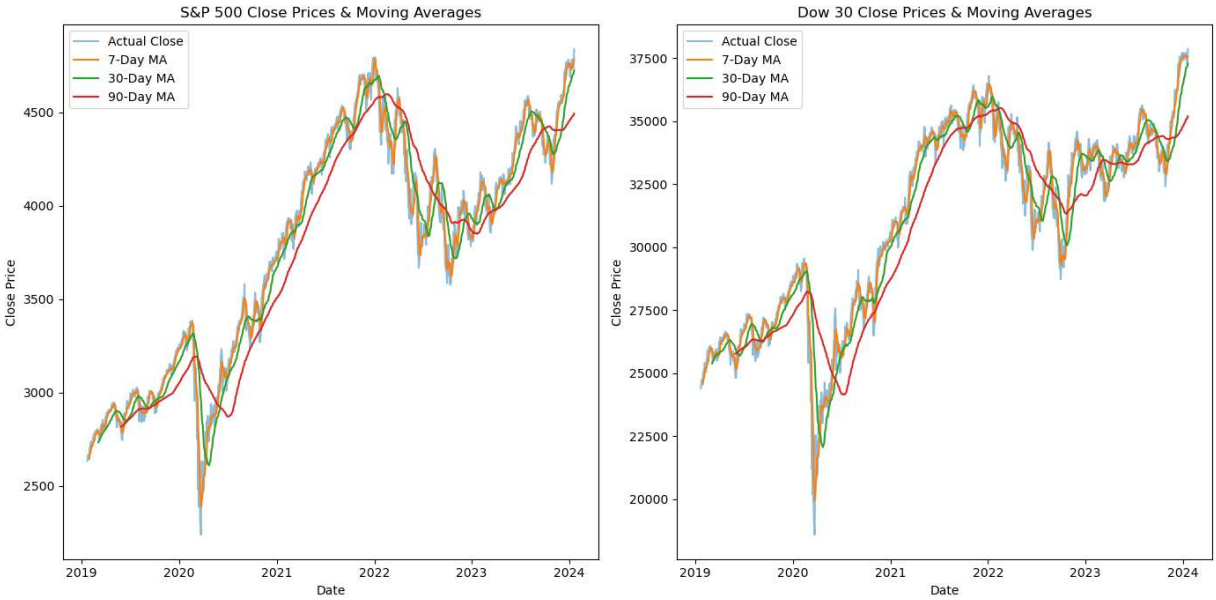
```
# Tend analysis ases on moving aveaages - To plot moving aerages and the actual clo
import matplotlib.pyplot as plt

plt.figure(figsize=(14, 7))

# Plot for S&P 500
plt.subplot(1, 2, 1)
plt.plot(merged_data['Date'], merged_data['Close_sp500'], label='Actual Close', alp
plt.plot(merged_data['Date'], merged_data['MA_7_sp500'], label='7-Day MA')
plt.plot(merged_data['Date'], merged_data['MA_30_sp500'], label='30-Day MA')
plt.plot(merged_data['Date'], merged_data['MA_90_sp500'], label='90-Day MA')
plt.title('S&P 500 Close Prices & Moving Averages')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.legend()

# Plot for Dow 30
plt.subplot(1, 2, 2)
plt.plot(merged_data['Date'], merged_data['Close_dow30'], label='Actual Close', alp
plt.plot(merged_data['Date'], merged_data['MA_7_dow30'], label='7-Day MA')
plt.plot(merged_data['Date'], merged_data['MA_30_dow30'], label='30-Day MA')
plt.plot(merged_data['Date'], merged_data['MA_90_dow30'], label='90-Day MA')
plt.title('Dow 30 Close Prices & Moving Averages')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.legend()

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



Here both the S&P 500 and the Dow 30 almost follows the same pattern over the past 5 years, There was a sharp dip around 2020 this could be the reason of COVID period. For the logn term both the index outperormed the market and hows the sing of bull.

The actual closing price - Daily

```
  It is the most volatile line on the chart, with all the daily
market movements reflected.It shows the real-time market
sentiment and is sensitive to daily market news, economic reports,
and other events
  There was a sharp dip around 2020 this could be the reason of
COVID period and market down trend.
```

The 7-Day Moving Average - short-term trend line

```
It helps to identify the short-term trend and momentum.
If the 7-day MA is trending upward and crosses above the longer-
term MAs, it is a Bull sign,
If the 7-day MA is trending downward and crosses below the longer-
term MAs it is a Bear sign.
Curently for the year of 2024, we can see the Bullish trend which
started from the year 2023 mid approximately.
```

The 30-Day Moving Average - medium-term trend line

```
The 30-day MA mediul term trend direction can be established when
comparing the actual closing prices to the 30-day MA.
If the actual closin price stays above the 30-day MA, it is a Bull
sign.
If the actual closin price stays below the 30-day MA, it is a Bear
sign.
```

The 90-Day Moving Average - longer-term trend line

```
The 90-day MA - long term trend direction can be established when
comparing the actual closing prices to the 90-day MA.
If the actual closin price stays above the 90-day MA, it is a Bull
sign.
If the actual closin price stays below the 90-day MA, it is a Bear
sign.

Based on investors financial plan and their goal investor can
follow the short,medium and long term trend analysis and plan their
investment options.

Here the similar activity can be performed for any stock or index
and identify their trend over the period of time same as above.
```

```
/=============================================================
```

SUBJECT : DSC 550 TERM PROJECT MILESTONE 3

```
Overview and Reason for selecting ARIMA model.
From milestone 3 reducing the scope of the poject to S&P 500 alone
and the same model fit process can be applied for DOW 30 index.

    MILESTONE 3.1 To find best p, d, q parameter values for the
best model fit
    MILESTONE 3.2 To fit the ARIMA Model
    MILESTONE 3.3 To valdiate the model fit with atual test data
    MILESTONE 3.4 To print, Summarise and interpretation of metrics
    MILESTONE 3.5 To Forecast S&P 500 INDEX closing price - future
values with the model
```

```
/=============================================================
```

Overview and Reason for selecting the model.

Time series analysis - Forecasting Models:

```
Several models can be used to forecast future values based on past
data. Some of these include

ARIMA (Autoregressive Integrated Moving Average): Useful for
modeling time series that show non-seasonal patterns.

Seasonal ARIMA (SARIMA): An extension of ARIMA that supports
univariate time series data with a seasonal component.
```

SARIMA model requries high computational power and takes lot of time to generate output, for the simplicity we are choosing ARIMA model here.

Considering the goal of predicting stock index/price based on hsitorical data we choose the time series analysis ARIMA model's for its simple computational power need.
Here we conside S&P 500 stock index historical data and apply ARIMA model fit with optimized p, d, q parameter values to forecast the future values.

In [14]:
```
# To display the dataset information.
merged_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1258 entries, 0 to 1257
Data columns (total 19 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   Date           1258 non-null   datetime64[ns]
 1   Open_sp500     1258 non-null   float64       
 2   High_sp500     1258 non-null   float64       
 3   Low_sp500      1258 non-null   float64       
 4   Close_sp500    1258 non-null   float64       
 5   Volume_sp500   1258 non-null   int64         
 6   Open_dow30     1258 non-null   float64       
 7   High_dow30     1258 non-null   float64       
 8   Low_dow30      1258 non-null   float64       
 9   Close_dow30    1258 non-null   float64       
 10  Volume_dow30   1258 non-null   int64         
 11  MA50_sp500     1209 non-null   float64       
 12  MA50_dow30     1209 non-null   float64       
 13  MA_7_sp500     1252 non-null   float64       
 14  MA_30_sp500    1229 non-null   float64       
 15  MA_90_sp500    1169 non-null   float64       
 16  MA_7_dow30     1252 non-null   float64       
 17  MA_30_dow30    1229 non-null   float64       
 18  MA_90_dow30    1169 non-null   float64       
dtypes: datetime64[ns](1), float64(16), int64(2)
memory usage: 196.6 KB
```

In [15]:
```
# MILESTONE 3.1 To find best p, d, q parameter values for the best model fit

from statsmodels.tsa.arima.model import ARIMA
import itertools
import warnings

p = d = q = range(0, 4)  # To define the parameter range
pdq = list(itertools.product(p, d, q))  # to generate all combinations of p, d, q

warnings.filterwarnings("ignore")  # To ignore any warnings during the loop
#merged_data.index = pd.to_datetime(merged_data.index)

best_aic = float("inf")
```

```
best_params = None

for param in pdq:
    try:
        model = ARIMA(merged_data['Close_sp500'], order=param)
        results = model.fit()
        if results.aic < best_aic:
            best_aic = results.aic
            best_params = param
    except:
        continue

print(f"Best ARIMA Parameters: {best_params}, AIC: {best_aic}")
```

```
Best ARIMA Parameters: (2, 2, 3), AIC: 13117.70780378316
```

In [16]:
```
# MILESTONE 3.2 To fit the ARIMA Model

from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
import numpy as np

# To fit the model with the best p, d, q parameter
model = ARIMA(merged_data['Close_sp500'], order=(2, 2, 3))
model_fit = model.fit()

# To forecast future values for next 5 period
forecast = model_fit.forecast(steps=5)

last_training_date = merged_data['Date'].iloc[-1]
last_training_index = merged_data.index[-1]  # Last date in the training dataset
print(f"Last Business day from training dataset: {last_training_date}")
print(f"Last index from training dataset: {last_training_index}")

# To print next 5 trading day's forecasted values, 2024-01-22 to 2024-01-26
print(f"Forecasted Values of next 5 business days from last business day 2024-01-22
```

```
Last Business day from training dataset: 2024-01-19 00:00:00
Last index from training dataset: 1257
Forecasted Values of next 5 business days from last business day 2024-01-22 to 2024-
01-26
1258    4843.351933
1259    4843.208616
1260    4846.742907
1261    4847.066569
1262    4849.783500
Name: predicted_mean, dtype: float64
```

In [17]:
```
#MILESTONE 3.3 To valdiate the model fit with atual test data

#To load the test dataset for validation(new dataset next to last test dataset date
test_data_path = 'GSPC.csv'  # Placeholder path for test data
test_data = pd.read_csv(test_data_path)
test_data.head(n=5)
```

Out[17]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2024-01-22	4853.419922	4868.410156	4844.049805	4850.430176	4850.430176	4297610000
1	2024-01-23	4856.799805	4866.479980	4844.370117	4864.600098	4864.600098	3912800000
2	2024-01-24	4888.560059	4903.680176	4865.939941	4868.549805	4868.549805	4330030000
3	2024-01-25	4886.660156	4898.149902	4869.339844	4894.160156	4894.160156	4020430000
4	2024-01-26	4888.910156	4906.689941	4881.470215	4890.970215	4890.970215	3353400000

In [18]:
```
# Actual values from the dataset for the model's evaluation

test_data['Date'] = pd.to_datetime(test_data['Date'])
test_data.set_index('Date', inplace=True)

test_dates = test_data.index[:5]  # Extract dates for the 5 entries to match foreca

# Print the dates and the corresponding actual closing prices
for date in test_dates:
    print(f"Date: {date}, Actual Closing Price: {test_data.loc[date, 'Close']}")
```

```
Date: 2024-01-22 00:00:00, Actual Closing Price: 4850.430176
Date: 2024-01-23 00:00:00, Actual Closing Price: 4864.600098
Date: 2024-01-24 00:00:00, Actual Closing Price: 4868.549805
Date: 2024-01-25 00:00:00, Actual Closing Price: 4894.160156
Date: 2024-01-26 00:00:00, Actual Closing Price: 4890.970215
```

In [19]:
```
#MILESTONE 3.4 To print, summarise and interpretation of metrics.

# Forecasted values for the next 5 business days
forecasted_values = np.array([4843.351933, 4843.208616, 4846.742907, 4847.066569, 4

# Actual closing prices for the corresponding dates
actual_values = np.array([4850.430176, 4864.600098, 4868.549805, 4894.160156, 4890.

# Calculate MSE and RMSE
mse = mean_squared_error(actual_values, forecasted_values)
rmse = np.sqrt(mse)

print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

```
Mean Squared Error (MSE): 979.4778511047301
Root Mean Squared Error (RMSE): 31.29661085652455
```

The Mean Squared Error (MSE) between the forecasted values and the actual closing prices for the specified dates is approximately 979.48, and the Root Mean Squared Error (RMSE) is approximately

31.30.

The ARIMA(2, 2, 3) model showed a reasonable ability to forecast future stock index closing prices over a short horizon. The forecasts were relatively close to the actual prices, with deviations within an expected range given market volatility. The model's performance, as indicated by an RMSE of about 31.30, suggests that while the forecasts were not perfect, they captured the general direction and magnitude of price movements to a useful extent.

In [20]:
```
# To print the summary of the model fit
print(model_fit.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:            Close_sp500   No. Observations:                 1258
Model:                 ARIMA(2, 2, 3)   Log Likelihood               -6552.854
Date:                Sun, 18 Feb 2024   AIC                          13117.708
Time:                        01:13:38   BIC                          13148.522
Sample:                             0   HQIC                         13129.289
                                - 1258
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.7648      0.027    -65.067      0.000      -1.818      -1.712
ar.L2         -0.8899      0.025    -35.081      0.000      -0.940      -0.840
ma.L1          0.6867      0.086      7.942      0.000       0.517       0.856
ma.L2         -0.8967      0.135     -6.663      0.000      -1.160      -0.633
ma.L3         -0.7898      0.071    -11.144      0.000      -0.929      -0.651
sigma2      1979.2820    161.809     12.232      0.000    1662.143    2296.421
===================================================================================
Ljung-Box (L1) (Q):                   0.31   Jarque-Bera (JB):               710.57
Prob(Q):                              0.58   Prob(JB):                         0.00
Heteroskedasticity (H):               1.10   Skew:                            -0.57
Prob(H) (two-sided):                  0.33   Kurtosis:                         6.50
===================================================================================

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

Summary:

This model involves,
- Second-order differencing (d=2)
- Two autoregressive (AR) terms (p=2)
- Three moving average (MA) terms (q=3)

To evaluate the fit of ARIMA model and compare it with other models:

These metrics can be compared with other ARIMA model's Goodnes of fit can be measure and best model and its parameters are

choosen based on lowest AIC, BIC.
Akaike Information Criterion (AIC) :13117.708
Bayesian Information Criterion (BIC) : 13148.522
Hannan-Quinn Information Criterion (HQIC)13129.289

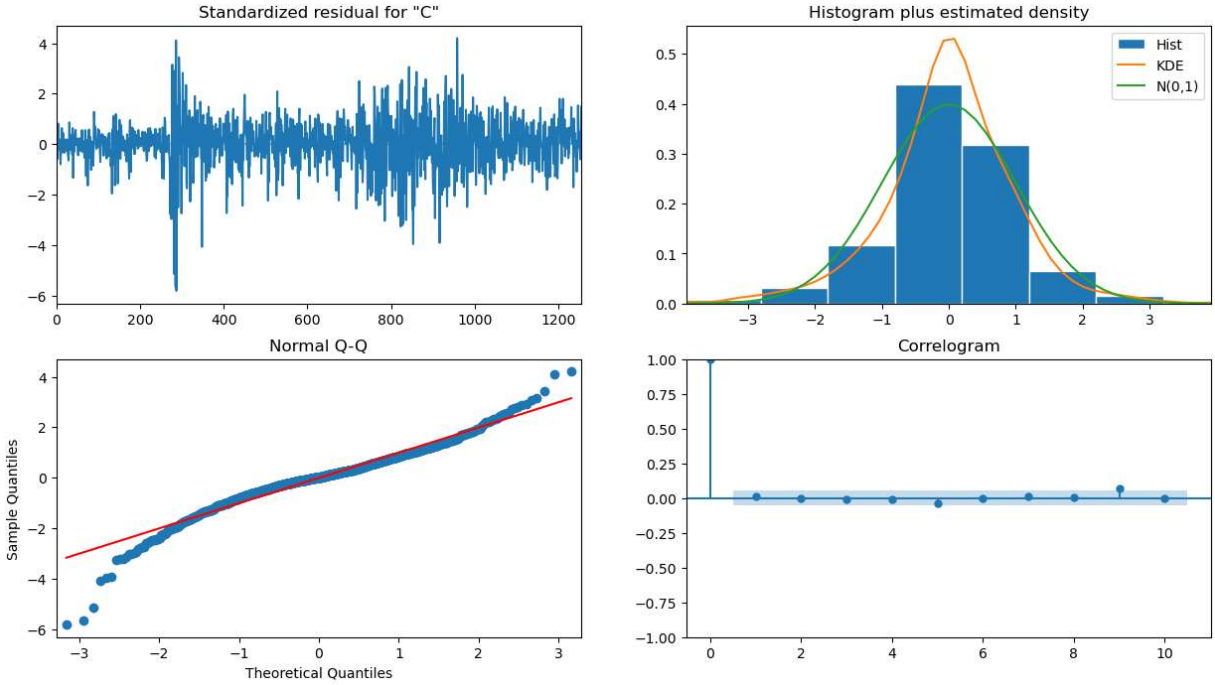
we already perfomed generating all combinations of p, d, q within range 0 to 4. and identified best parameter model, so assuming this is the best model fit with bett AIC,BIC and HQIC.

Coefficients Interpretation:
Autoregressive(AR) coefficients (ar.L1:-1.7648, ar.L2:-0.8899):This indicates that the model finds a significant negative relationship between the current value of the series and its past values.
Moving average (MA) coefficients (ma.L1:0.6867, ma.L2:-0.8967, ma.L3:-0.8967) indicates the model's attempt to correct for any autocorrelation not captured by the AR part giving us about significant moving average effects in the data.
sigma2: represents the variance of the model's residuals. The estimated value suggests the level of noise or error terms in the model.
std err Interpretation:
The smaller values indicating the precision of these estimates.

Confidence intervals ([0.025 0.975]) provide a range of values within which the true parameter value is expected to fall 95% of the time. Here none of these intervals include 0 further confirms the statistical significance of the coefficients.

In [21]:
```
# Residuals diagnostics
model_fit.plot_diagnostics(figsize=(15, 8))
plt.show()

# Optionally, Ljung-Box test
from statsmodels.stats.diagnostic import acorr_ljungbox
ljung_box, p_value = acorr_ljungbox(model_fit.resid)
print(f'Ljung-Box test: {ljung_box[:10]}')
print(f'p-Value: {p_value[:10]}')
```



Ljung-Box test: lb_stat
p-Value: lb_pvalue

SUMMARY:

Standardized Residuals Plot:

The plot suggests that there may be some outliers or noise, given a few spikes, but there's no apparent trend or seasonality, which is good.

Histogram and Estimated Density:

the distribution of residuals alongside a Kernel Density Estimate (KDE) and the normal distribution (N(0,1)).The residuals has a slight skew, as indicated by the density being higher on the left and the peak being off-center.

Normal Q-Q Plot:

Quantile-quantile plot compares the distribution of residuals to a normal distribution. Points lying on the red line suggest that the residuals are normally distributed. Deviations from this line, especially in the tails, would indicate deviations from normality.the plot shows some deviation in the tails, suggesting that the residuals may not be entirely normal, which could indicate the presence of outliers or heavy tails in the distribution of residuals.

Correlogram (ACF Plot):

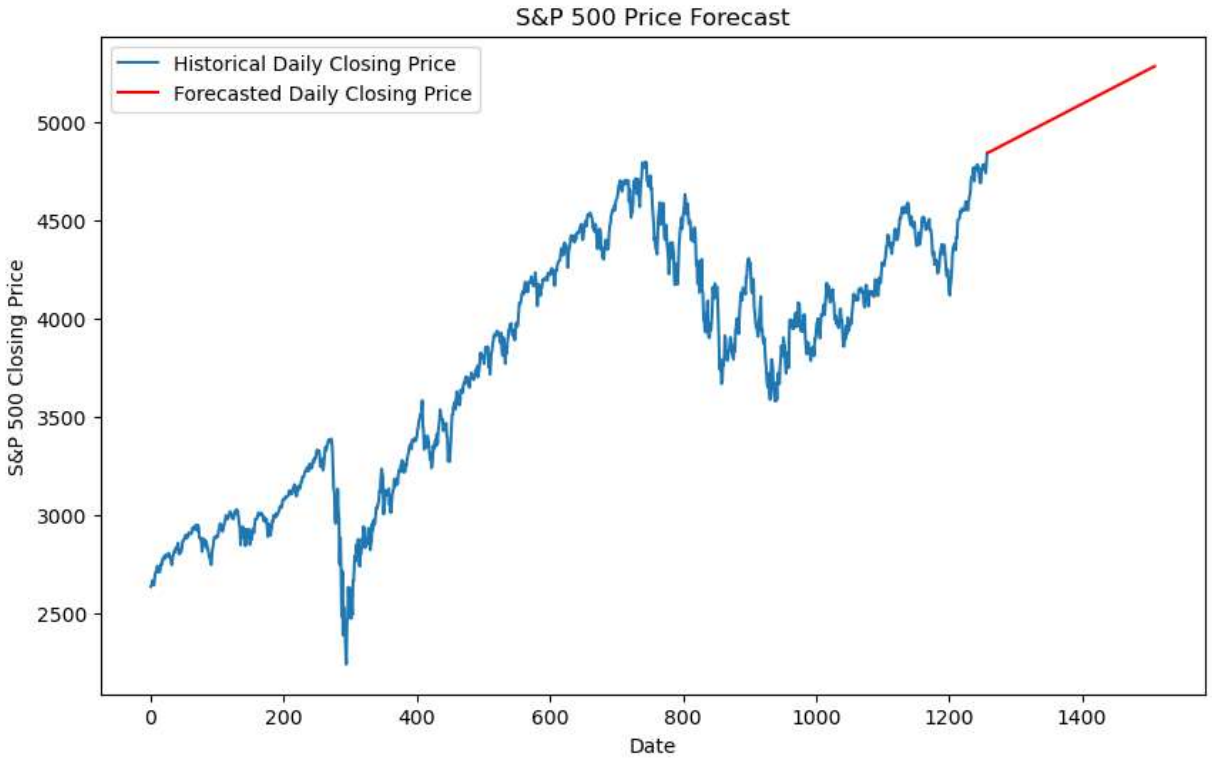
This plot shows the autocorrelation of the residuals at different lags. Ideally, all dots should fall within the blue band, which represents the confidence interval. If they fall outside, it suggests significant autocorrelation at that lag. the correlogram shows that autocorrelations are within the confidence

band for all lags, indicating no significant autocorrelation.

In [22]:
```
# MILESTONE 3.5 To Forecast S&P 500 INDEX closing price - future values with the mo

# Forecast the next 252 bussines/trading days(approxmtely one trading year)
forecast = model_fit.forecast(steps=252)

# Plot the forecast alongside the historical data
plt.figure(figsize=(10,6))
plt.plot(merged_data['Close_sp500'].index, merged_data['Close_sp500'], label='Histo
plt.plot(forecast.index, forecast, label='Forecasted Daily Closing Price', color='r
plt.xlabel('Date')
plt.ylabel('S&P 500 Closing Price')
plt.title('S&P 500 Price Forecast')
plt.legend()
plt.show()
```



Conclusion:

The ARIMA(2, 2, 3) model showed a reasonable ability to forecast future stock index closing prices over a short horizon. The forecasts were relatively close to the actual prices, with deviations within an expected range given market volatility. The model's performance, as indicated by an RMSE of about 31.30, suggests that while the forecasts were not perfect, they captured the general direction and magnitude of price movements to a useful extent.

Further Improvements:

Here the model doesn't include complex market influencing factors like, job market report, consumer interest, inflation and government stand on political and inflation matters.The model can be further expanded and improved with these information for best model fit to get more accurate forecast.
The same model fit process and evaluation metrics can be applied for DOW 30 index.

In [ ]: