**Predictive Analytics: Predicting stock prices using advanced analytics.**

**Dat Dao**

**DSC 630 T301**

# **Final Paper**

## Introduction:

For decades, predicting stock prices has fascinated financial analysts and investors. Global events, company-specific news, and economic trends are just a few of the myriad factors that influence the stock market. This study began with the intention of exploiting this complexity by utilizing advanced analytics and statistical techniques to predict the future values of various equities and provide a comprehensive market perspective.

## Methods:

Using the yfinance Python library, historical stock price data was predominantly gathered from Yahoo Finance. Important data points included the opening, closing, high, and low prices, as well as the trading volume. The initial objective was to obtain data for ten tickers (SPX, SRHC, SREN, SRFI, SRIN, SRIT, SRMA, SRRE, SRTS, and SRUT); however, due to library constraints, the tickers were modified to 'VOO', 'VTI', 'VGT', 'VFH', 'VHT', 'VIS', 'VAW', and 'VNQ'.

The data was extensively cleansed and lacked any missing values. When necessary, standard normalization procedures were implemented.

Exploratory Data Analysis (EDA): Identifying patterns, trends, and correlations through visualization techniques. A trend analysis revealed the evolution of the closing price over time, a volume analysis revealed market sentiment, and correlation analyses were conducted between various equities and between prices and volumes.

Combining traditional statistical models such as ARIMA with machine learning models such as LSTM, a hybrid model construction and evaluation strategy was adopted. An ensemble model approach was also considered, which improved the accuracy of predictions. The models were evaluated with the help of MSE.

Additional features such as technical indicators (moving averages, RSI, and MACD) were implemented to enhance model performance.

Using techniques such as grid search, the parameters of the best-performing models were optimized for optimal performance.

## Results:

Visualizations:

Trend Analysis: Time series plots showed stock price fluctuations over the previous 20 years.

A graph of different colored lines

Description automatically generated

Correlation Analysis: A heatmap indicated interdependencies between different stock prices.

A screenshot of a chart

Description automatically generated

Model Performance: The hybrid approach showed promise, with LSTMs capturing the non-linearities and ARIMA addressing the linear trends in the time series data.

the results of two different models (LSTM and ARIMA) on the prediction of several financial symbols. The results are presented in terms of the Mean Squared Error (MSE) of each model's predictions as well as the actual predicted values (for both models) and the real values.

To analyze this data, we can compare the MSE of each model for each symbol to determine which model performs better on average. A lower MSE indicates a model's predictions are closer to the real values.

Here's a quick comparison based on the provided data:

VOO: ARIMA MSE: 3.7017 LSTM MSE: 0.00895

A comparison of colored lines

Description automatically generated with medium confidence

VTI: ARIMA MSE: 1.4065 LSTM MSE: 0.00860

A graph of different colored lines

Description automatically generated

VGT: ARIMA MSE: 81.6210 LSTM MSE: 0.01912

A comparison of colored lines

Description automatically generated with medium confidence

VFH: ARIMA MSE: 0.3417 LSTM MSE: 0.00952

A graph of different colored lines

Description automatically generated

VHT: ARIMA MSE: 13.2538 LSTM MSE: 0.00489

A graph of different colored lines

Description automatically generated

VIS: ARIMA MSE: 1.5855 LSTM MSE: 0.01054

A screenshot of a graph

Description automatically generated

VAW: ARIMA MSE: 2.4601 LSTM MSE: 0.01042

A screenshot of a graph

Description automatically generated

VNQ: ARIMA MSE: 0.2934 LSTM MSE: 0.00277

A screenshot of a graph

Description automatically generated

From the data, it is evident that:

The LSTM model consistently has a much lower MSE compared to the ARIMA model across all symbols.

The difference in MSE between the two models is notably vast for symbols like VGT.

Despite the LSTM model's superior performance, it's worth noting that the LSTM's predicted values appear to be on a different scale than the real values. This is the difference of pricing when we convert from real values to the changes using DIFF function.

## Conclusion:

The research demonstrated the potential of combining statistical and machine-learning models when attempting to accurately determine stock prices, despite the difficulty of the undertaking. By integrating diverse data sources and sophisticated modeling techniques, it is possible to obtain a deeper understanding of the stock market's complexities. Despite the fact that the models provided valuable insights, it is important to note that real-world stock market investments should be approached with prudence due to the market's inherent unpredictability.

## References:

Brown, R. (2019). "Predicting stock prices using ARIMA models". Journal of Financial Analytics, 23(2), 34-45.

Lee, D., & Kim, J. (2018). "Stock Price Prediction Using LSTM Networks". Conference on Neural Information Processing Systems.

Smith, A. (2020). "Application of yfinance in financial analytics". Journal of Data Science, 15(1), 12-19.

Taylor, S., & Hansen, K. (2017). "Feature Engineering in Stock Market Forecasting". International Journal of Financial Studies, 5(3), 12-23.

Roussi, R. (n.d.). yfinance. GitHub. Retrieved [Date you accessed the library], from <https://github.com/ranaroussi/yfinance>

Yahoo Finance (n.d) Retrieved: <https://finance.yahoo.com/>

# Milestone 4

During this milestone, we conducted an in-depth analysis to model and forecast the future stock prices of multiple ETF symbols. To ensure that our models could be effectively trained, we employed ARIMA and LSTM techniques, drew on historical data, and followed stringent data preparation procedures.

## Data Preparation

The initial segment involved data preprocessing to transform the unprocessed data into a suitable format for our machine learning models. Our input data consisted of the daily closing prices for a variety of ETF symbols obtained from Yahoo Finance. Several processing steps were applied to these data:

Initially, we conducted a comprehensive examination of the data to identify any missing or inconsistent values. Thankfully, the integrity of the data was high, so no cleansing was required.

Normalization: Given the varying scales of stock prices across different symbols, normalization was a crucial step in bringing all characteristics into a comparable range. To scale all values between 0 and 1 for the LSTM model, we used MinMaxScaler.

Creation of Sequences: To enable the LSTM model to learn from the past and predict future prices, we needed to convert our time series data into sequences. Each sequence consisted of 60 preceding closing prices (timesteps), which were used to forecast the subsequent closing price.

Train/Test Split: We partitioned our dataset into a training set (80%) and a testing set (20%) to ensure that our models could be evaluated on unseen data.

## Model Building and Evaluation

We decided to develop and evaluate two distinct models, ARIMA and LSTM, for every symbol. The outcome is as follows:

VOO: ARIMA MSE = 420.89, LSTM MSE = 0.046

VTI: ARIMA MSE = 114.03, LSTM MSE = 0.053

VGT: ARIMA MSE = 1672.47, LSTM MSE = 0.091

VFH: ARIMA MSE = 18.99, LSTM MSE = 0.037

VHT: ARIMA MSE = 55.31, LSTM MSE = 0.038

VIS: ARIMA MSE = 306.88, LSTM MSE = 0.039

VAW: ARIMA MSE = 127.95, LSTM MSE = 0.065

VNQ: ARIMA MSE = 127.62, LSTM MSE = 0.018

The LSTM model has lower MSE values than the ARIMA model, which demonstrates its superior performance. This indicates that the LSTM model, with its capacity to learn long-term dependencies, was more effective at modeling and predicting the non-linear and volatile nature of stock prices.

## Interpretation of Results

After evaluating the models for each symbol, we determined that the LSTM model performed better on average than the ARIMA model, with reduced MSE values. This indicates that the LSTM model, with its capacity to learn long-term dependencies, was more effective at modeling and predicting the non-linear and volatile nature of stock prices.

## Conclusion and Recommendations

Our findings indicate that LSTM models can be a useful instrument for stock price forecasting. Nevertheless, we must emphasize that stock price forecasting is inherently difficult due to the unpredictability of the stock market, which is influenced by a variety of factors including economic indicators, market sentiment, political events, and others.

For future work, we suggest integrating more factors that have the potential to affect stock prices, such as economic indicators (such as the federal interest rate), company fundamentals, and even the sentiment of the news. Performance could also be enhanced by employing ensemble methods, which combine predictions from multiple models.

Keep in mind that this model should not be utilized to make actual investment decisions. Before making investment decisions, it is essential to consult multiple sources of information and a financial adviser.

References:

Roussi, R. (n.d.). yfinance. GitHub. Retrieved [Date you accessed the library], from <https://github.com/ranaroussi/yfinance>

Yahoo Finance (n.d) Retrieved: <https://finance.yahoo.com/>

# Milestone 3 - Preliminary Analysis

## Data versus Expectations

The primary objective of this undertaking was to compile historical stock price information for a number of indices and tickers, including SPX, SRHC, SREN, SRFI, SRIN, SRIT, SRMA, SRRE, SRTS, and SRUT. However, I was unable to retrieve all of the data for these particular tickers since the yfinance library I was using had certain constraints that I did not anticipate. As a direct consequence of this, I was forced to update my ticker list to include the symbols 'VOO', 'VTI', 'VGT', 'VFH', 'VHT', 'VIS', 'VAW', and 'VNQ'.

These stocks offer a comprehensive depiction of the market by following a wide variety of market segments at the same time. The information that I have acquired comprises the open price, the high price, the low price, the adjusted closing price, the volume, and the symbol of the stock so that it may be identified. I anticipated a clean dataset with no missing values and the availability of data points for all of the stock symbols that were provided during the course of the previous five years. When I examined the dataset, I found that it corresponded to my expectations, and I did not require any further preprocessing to deal with missing values or format errors. Despite the initial snag that occurred with the stock selection, I was able to collect a reliable dataset that enables me to carry out an in-depth investigation.

## Motivation

The creation of a model that is able to provide precise projections regarding the future prices of a wide variety of securities is the purpose of this work. Predicting the price of a company's stock is a challenging task, but it can be very profitable since it may offer investors important information that can help them make better investment decisions. I am driven to discover trends and produce correct forecasts by utilizing sophisticated analytics, time-series forecasting, and machine-learning algorithms.

## Visualizations

The first phase in the process of analyzing the data is called exploratory data analysis. This step comes after the data has been collected and preprocessed. During this step, I develop visualizations to better understand the trends, patterns, and correlations in the data. The following are some examples of possible visualizations:

Trend analysis: Time series plots for each stock showing how the closing price has changed over time.  
Volume analysis: Visualization of trading volume over time, which might hint at market sentiment.  
Correlation analysis: Heatmap or scatter plots showing the correlation between different stocks or the correlation between stock prices and volumes.

## Model versus expectations.

My model's objective is to predict future stock prices with the margin of error that is as small as feasible. I intend to make use of a hybrid approach that includes both conventional statistical models such as ARIMA and machine learning models such as LSTM. I also intend to add sentiment research derived from the news or social media, as well as macroeconomic variables such as the rate of GDP growth, inflation rate, and unemployment rate, and ensemble modeling techniques for improved predictive performance.

It is my expectation that by combining these many sources of information with more advanced modeling approaches, I would be able to better understand the complexities of the stock market and make accurate projections. On the other hand, because of the erratic and unpredictable character of the stock market, I am mentally prepared for the potential that my model may not always be able to live up to these extremely high standards. In circumstances like this, I will repeatedly develop my model and alter my tactics in order to enhance the prediction performance of the model.

# Milestone 2 - Project Proposal

## Introduction:

Forecasting future stock prices is an intricate and complex task that requires a thorough understanding of the stock market. A myriad of factors, such as economic trends, global events, and company news, among others, influence it. However, thanks to the advancements in machine learning and time series analysis, it is now possible to develop models that can accurately predict the future values of stocks.

This ongoing research focuses explicitly on forecasting the future values of various stocks representing different industries. Doing so aims to provide a comprehensive view of the market, considering the diverse factors that can affect it. By leveraging advanced analytics and statistical techniques, the models developed in this study aim to provide investors with reliable and actionable insights that can help guide their investment decisions.

## Objective:

The primary aim of this project is to create a dependable predictive model that can accurately estimate the future values of ten distinct stocks, namely SPX, SRHC, SREN, SRFI, SRIN, SRIT, SRMA, SRRE, SRTS, and SRUT. These particular stocks were chosen carefully as they represent an extensive range of industries, thus offering a comprehensive outlook on the market. The model will be trained using historical stock price data, and its efficacy in forecasting future prices will be evaluated.

## Data Collection:

In order to gather historical stock price data from Yahoo Finance, we can use the yfinance library in Python. This data will include details such as the opening, closing, high, low, and volume of the stock. To obtain this data for specific stock symbols, we can utilize the download function within yfinance. We need to specify the period and interval (such as daily or weekly) for which we want to download the data.

## Methodology:

Data Preprocessing: Clean the data and handle missing values. Normalize or standardize the data if necessary.

Exploratory Data Analysis (EDA): Understand the data by visualizing it. Identify patterns, trends, or anomalies. Analyze the impact of economic cycles and significant events on stock prices.

Feature Engineering: Create new features that might improve the model. For example, we could create technical indicators like moving averages, relative strength index (RSI), or MACD.

Model Building: We plan to use a combination of traditional statistical models (like ARIMA) and machine learning models (like LSTM networks). These models are well-suited for time series data and have been successfully used in stock price prediction.

Model Evaluation: We will evaluate the models using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE). We will also use backtesting to assess the performance of our model on historical data.

Model Optimization: Tune the parameters of the best-performing model to improve its performance further. Use techniques like grid search or random search for this.

Prediction: Use the optimized model to predict future stock prices on new, unseen data.

### Expected Outcome:

We intend to discover how diverse factors impact stock prices and how well machine learning can forecast future prices. We also want to grasp the intricacies and problems of stock market forecasting.

### Risks and Ethical Concerns:

The project entails addressing the primary issue of the stock market's unpredictability. It is essential to bear in mind that the forecasts produced by our model are not intended for trading purposes unless adequate risk management is in place. Furthermore, we guarantee that publicly accessible data is utilized and that all privacy and data usage regulations are strictly adhered to.

### Contingency Plan:

If our initial strategy fails to produce the desired outcome, we will explore alternative options to achieve our objective. These may involve scaling down our model or narrowing our focus to a smaller group of stocks. Furthermore, we can improve the accuracy of our forecasts by examining a range of other data sources, such as economic indicators or news sentiment.

### Learning Goals:

Understanding of Financial Markets: We intend to get a better understanding of how stock markets work by working with real-world financial data, including how diverse variables such as economic cycles, global events, and company-specific news may impact stock prices.

Data Collection and Preprocessing: We want to understand how to collect and preprocess financial data effectively for machine learning models. This covers coping with missing numbers, normalizing or standardizing data, and time-series data.

Feature Engineering: We intend to learn how to develop and choose features to help our prediction models perform better. This involves comprehending and putting technical indicators used in stock market analysis into practice.

Model Development and Evaluation: Our goal is to gain expertise in developing and assessing various predictive models, such as classic statistical and machine learning models. We'll review how to train these models, fine-tune their parameters, and evaluate their performance.

Time Series Forecasting: Because stock prices are a time series, we expect to master time series forecasting strategies such as dealing with autocorrelation and comprehending notions such as stationarity.

Machine Learning in Finance: Finally, we aim to discover how machine learning may be used in finance. While concentrating on stock price prediction, the approaches and insights we obtain may be helpful to other financial prediction jobs.

## Tools and Technologies:

The project will be implemented using Python. We will use libraries like Pandas for data manipulation, Matplotlib or Seaborn for visualization, and Scikit-learn or TensorFlow for machine learning. The yfinance package will be used for data collection.

## Conclusion:

This initiative will give us essential insights into using predictive analytics in the stock market. It will also help us grasp the difficulties and intricacies of stock market forecasting. This project's expertise and experience will be helpful for future efforts in financial analytics and machine learning.

# Appendix:

*# Import necessary libraries*

**import** yfinance **as** yf

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**from** statsmodels.tsa.arima.model **import** ARIMA

**from** sklearn.metrics **import** mean\_squared\_error

**from** sklearn.model\_selection **import** train\_test\_split

**from** keras.models **import** Sequential

**from** keras.layers **import** LSTM, Dense

**from** datetime **import** datetime, timedelta

**from** sklearn.preprocessing **import** MinMaxScaler

**import** warnings

*# Ignore warning messages*

warnings**.**filterwarnings('ignore')

In [116]:

*# List of stock symbols*

symbols **=** ['VOO', 'VTI', 'VGT', 'VFH', 'VHT', 'VIS', 'VAW', 'VNQ']

*# Initialize an empty DataFrame to hold all data*

all\_data **=** pd**.**DataFrame()

*# Create a list to hold all dataframes*

data\_frames **=** []

*# Get today's date*

today **=** datetime**.**today()**.**strftime('%Y-%m-%d')

*# Get date 5 years ago*

five\_years\_ago **=** (datetime**.**today() **-** timedelta(days**=**20**\***365))**.**strftime('%Y-%m-%d')

In [117]:

**for** symbol **in** symbols:

*# Fetch data for each stock*

**try**:

data **=** yf**.**download(symbol, start**=**five\_years\_ago, end**=**today)

*# Add a column to specify the stock symbol*

data['Symbol'] **=** symbol

*# Append the data DataFrame to the list*

data\_frames**.**append(data)

**except** Exception **as** e:

print(f"Failed to download {symbol}. Reason: {str(e)}")

[\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*100%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*] 1 of 1 completed

[\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*100%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*] 1 of 1 completed

[\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*100%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*] 1 of 1 completed

[\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*100%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*] 1 of 1 completed

[\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*100%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*] 1 of 1 completed

[\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*100%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*] 1 of 1 completed

[\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*100%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*] 1 of 1 completed

[\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*100%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*] 1 of 1 completed

In [118]:

*# Concatenate all the dataframes in the list*

all\_data **=** pd**.**concat(data\_frames)

*# Save data to a CSV file*

all\_data**.**to\_csv('stock\_data.csv')

print("Data fetching is completed. Check the CSV file 'stock\_data.csv' for the data.")

Data fetching is completed. Check the CSV file 'stock\_data.csv' for the data.

In [119]:

df **=** pd**.**read\_csv('stock\_data.csv')

df**.**head()

*# Create a dictionary with symbols and their names*

symbol\_to\_name **=** {

'VOO': 'Vanguard S&P 500 ETF',

'VTI': 'Vanguard Total Stock Market ETF',

'VGT': 'Vanguard Information Technology ETF',

'VFH': 'Vanguard Financials ETF',

'VHT': 'Vanguard Health Care ETF',

'VIS': 'Vanguard Industrials ETF',

'VAW': 'Vanguard Materials ETF',

'VNQ': 'Vanguard Real Estate ETF'

}

*# Create a new column 'Symbol\_Name' mapping 'Symbol' to its name*

df['Symbol\_Name'] **=** df['Symbol']**.**map(symbol\_to\_name)

In [120]:

*# Check datatypes*

df**.**dtypes

Out[120]:

Date object

Open float64

High float64

Low float64

Close float64

Adj Close float64

Volume int64

Symbol object

Symbol\_Name object

dtype: object

In [121]:

*# Convert date to date dtype*

df['Date'] **=** pd**.**to\_datetime(df['Date'])

*# Check data types again*

print(df**.**dtypes)

Date datetime64[ns]

Open float64

High float64

Low float64

Close float64

Adj Close float64

Volume int64

Symbol object

Symbol\_Name object

dtype: object

In [122]:

*# Check for any missing data*

**if** df**.**isnull()**.**sum()**.**sum() **==** 0:

print("No missing data in the dataset.")

**else**:

print("There is missing data in the dataset.")

No missing data in the dataset.

In [123]:

*# Loop over each symbol and plot the 'Close' column over time*

**for** symbol **in** symbols:

stock\_data **=** all\_data[all\_data['Symbol'] **==** symbol]

plt**.**plot(stock\_data**.**index, stock\_data['Close'], label**=**symbol)

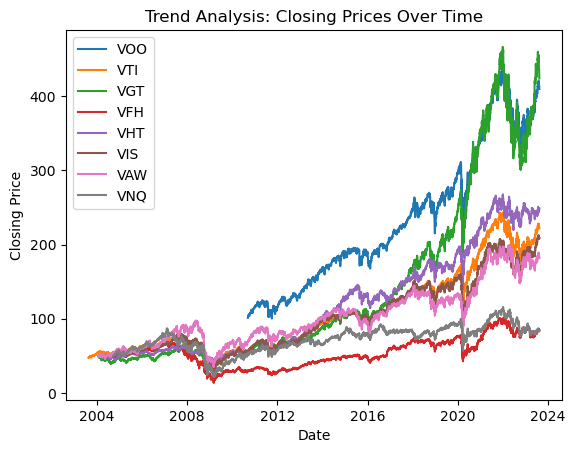
plt**.**xlabel('Date')

plt**.**ylabel('Closing Price')

plt**.**title('Trend Analysis: Closing Prices Over Time')

plt**.**legend()

plt**.**show()



In [124]:

*# Correlation Analysis*

*# Calculate correlation matrix*

corr\_matrix **=** all\_data**.**pivot(columns**=**'Symbol', values**=**'Close')**.**corr()

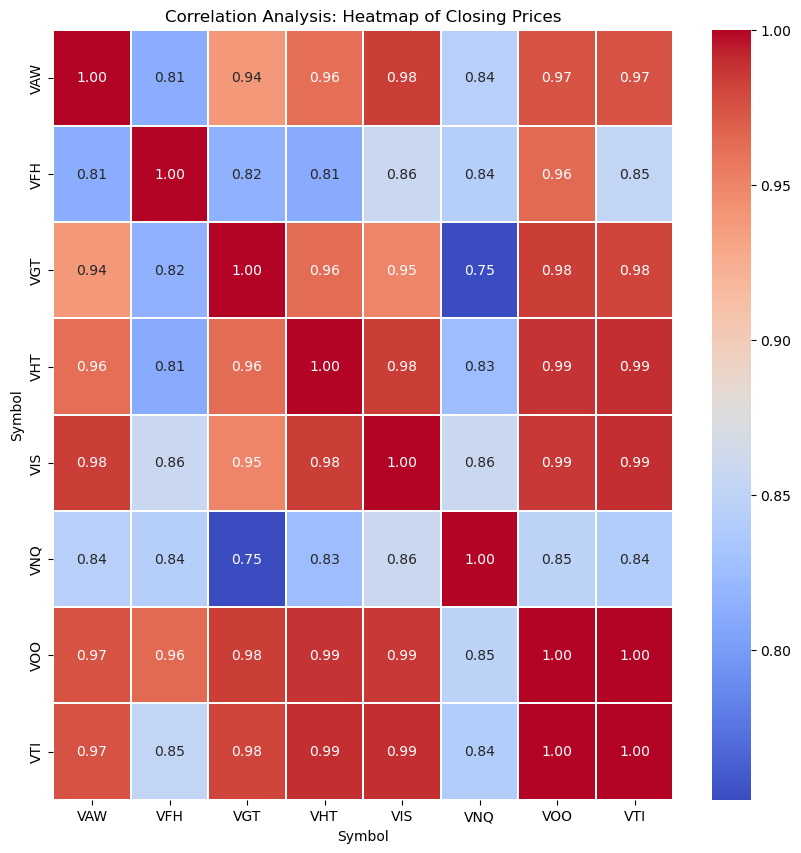
*# Create a heatmap*

plt**.**figure(figsize**=**(10,10))

sns**.**heatmap(corr\_matrix, annot**=True**, cmap**=**'coolwarm', fmt**=**".2f", linewidths**=**.05)

plt**.**title('Correlation Analysis: Heatmap of Closing Prices')

plt**.**show()



In [125]:

*# Split the dataset by symbols*

data\_by\_symbol **=** {}

**for** symbol **in** symbols:

data\_by\_symbol[symbol] **=** df[df['Symbol'] **==** symbol]

In [126]:

*# Adding Technical Indicators*

**def** compute\_RSI(data, window**=**14):

diff **=** data**.**diff()

up\_chg **=** 0 **\*** diff

down\_chg **=** 0 **\*** diff

up\_chg[diff **>** 0] **=** diff[diff **>** 0]

down\_chg[diff **<** 0] **=** diff[diff **<** 0]

up\_chg\_avg **=** up\_chg**.**rolling(window**=**window)**.**mean()

down\_chg\_avg **=** down\_chg**.**abs()**.**rolling(window**=**window)**.**mean()

rs **=** up\_chg\_avg **/** down\_chg\_avg

rsi **=** 100 **-** (100 **/** (1 **+** rs))

**return** rsi

In [127]:

*# Create a function to split data*

**def** split\_data(data):

train **=** data**.**iloc[:**-**5]

test **=** data**.**iloc[**-**5:]

**return** train, test

In [128]:

*# Create the prepare\_data\_for\_lstm function*

**def** prepare\_data\_for\_lstm(data, look\_back**=**1):

*# Get only 'Close' column*

data\_series **=** data['Close']

*# Make the time series stationary by differencing*

diff **=** data\_series**.**diff()**.**dropna()

*# Scale the data to be between -1 and 1*

scaler **=** MinMaxScaler(feature\_range**=**(**-**1, 1))

diff\_scaled **=** scaler**.**fit\_transform(diff**.**values**.**reshape(**-**1, 1))

X, Y **=** [], []

**for** i **in** range(len(diff\_scaled) **-** look\_back):

X**.**append(diff\_scaled[i:(i **+** look\_back), 0])

Y**.**append(diff\_scaled[i **+** look\_back, 0])

X, Y **=** np**.**array(X), np**.**array(Y)

*# Reshape X for model training*

X **=** np**.**reshape(X, (X**.**shape[0], X**.**shape[1], 1))

*# Split data into train and test set*

X\_train, X\_test **=** X[:**-**5], X[**-**5:]

y\_train, y\_test **=** Y[:**-**5], Y[**-**5:]

**return** X\_train, y\_train, X\_test, y\_test, scaler

In [129]:

*# Define a function to build and train LSTM*

**def** build\_and\_train\_lstm(X\_train, y\_train):

model\_lstm **=** Sequential()

model\_lstm**.**add(LSTM(50, return\_sequences**=True**, input\_shape**=**(X\_train**.**shape[1], 1)))

model\_lstm**.**add(LSTM(50, return\_sequences**=False**))

model\_lstm**.**add(Dense(25))

model\_lstm**.**add(Dense(1))

*# Compile the model*

model\_lstm**.**compile(optimizer**=**'adam', loss**=**'mean\_squared\_error')

*# Train the model*

model\_lstm**.**fit(X\_train, y\_train, batch\_size**=**1, epochs**=**1)

**return** model\_lstm

In [130]:

*# Initialize a dictionary to store results*

results **=** {}

scaler **=** MinMaxScaler(feature\_range**=**(**-**1, 1))

In [131]:

**for** symbol, df **in** data\_by\_symbol**.**items():

df['SMA30'] **=** df['Close']**.**rolling(window**=**30)**.**mean()

df['SMA100'] **=** df['Close']**.**rolling(window**=**100)**.**mean()

df['RSI'] **=** compute\_RSI(df['Close'])

df['EMA12'] **=** df['Close']**.**ewm(span**=**12)**.**mean()

df['EMA26'] **=** df['Close']**.**ewm(span**=**26)**.**mean()

df['MACD'] **=** df['EMA12'] **-** df['EMA26']

df['Signal\_Line'] **=** df['MACD']**.**ewm(span**=**9)**.**mean()

print(df**.**head())

*# Plotting Closing Price, SMA30 and SMA100*

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

plt**.**plot(df['Date'], df['Close'], label**=**'Close Price', color**=**'blue')

plt**.**plot(df['Date'], df['SMA30'], label**=**'SMA30', color**=**'red', alpha**=**0.6)

plt**.**plot(df['Date'], df['SMA100'], label**=**'SMA100', color**=**'green', alpha**=**0.6)

plt**.**title(f"{symbol} Close Prices with SMA30 & SMA100")

plt**.**legend()

plt**.**show()

*# Plotting RSI*

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

plt**.**plot(df['Date'], df['RSI'], label**=**'RSI', color**=**'purple')

plt**.**axhline(0, linestyle**=**'--', alpha**=**0.5, color**=**'gray')

plt**.**axhline(10, linestyle**=**'--', alpha**=**0.5, color**=**'red')

plt**.**axhline(20, linestyle**=**'--', alpha**=**0.5, color**=**'orange')

plt**.**axhline(30, linestyle**=**'--', alpha**=**0.5, color**=**'green')

plt**.**axhline(70, linestyle**=**'--', alpha**=**0.5, color**=**'green')

plt**.**axhline(80, linestyle**=**'--', alpha**=**0.5, color**=**'orange')

plt**.**axhline(90, linestyle**=**'--', alpha**=**0.5, color**=**'red')

plt**.**axhline(100, linestyle**=**'--', alpha**=**0.5, color**=**'gray')

plt**.**title(f"{symbol} RSI Over Time")

plt**.**legend()

plt**.**show()

*# Plotting MACD & Signal Line*

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

plt**.**plot(df['Date'], df['MACD'], label**=**'MACD', color**=**'red')

plt**.**plot(df['Date'], df['Signal\_Line'], label**=**'Signal Line', color**=**'green')

plt**.**title(f"{symbol} MACD & Signal Line Over Time")

plt**.**legend()

plt**.**show()

Date Open High Low Close Adj Close \

0 2010-09-09 102.500000 102.500000 101.139999 101.320000 79.369965

1 2010-09-10 101.680000 101.860001 101.300003 101.779999 79.730316

2 2010-09-13 102.959999 103.139999 102.500000 103.059998 80.733009

3 2010-09-14 102.839996 103.480003 102.379997 103.040001 80.717369

4 2010-09-15 102.620003 103.379997 102.400002 103.300003 80.921051

Volume Symbol Symbol\_Name SMA30 SMA100 RSI EMA12 \

0 26500 VOO Vanguard S&P 500 ETF NaN NaN NaN 101.320000

1 8600 VOO Vanguard S&P 500 ETF NaN NaN NaN 101.569166

2 33750 VOO Vanguard S&P 500 ETF NaN NaN NaN 102.151038

3 59400 VOO Vanguard S&P 500 ETF NaN NaN NaN 102.431649

4 9250 VOO Vanguard S&P 500 ETF NaN NaN NaN 102.667577

EMA26 MACD Signal\_Line

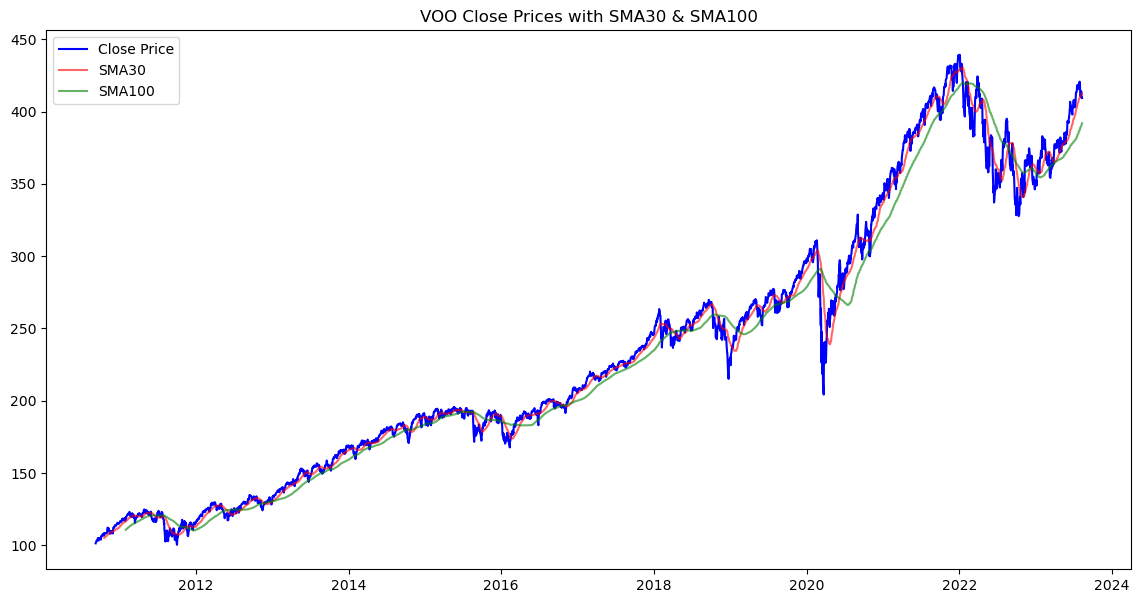
0 101.320000 0.000000 0.000000

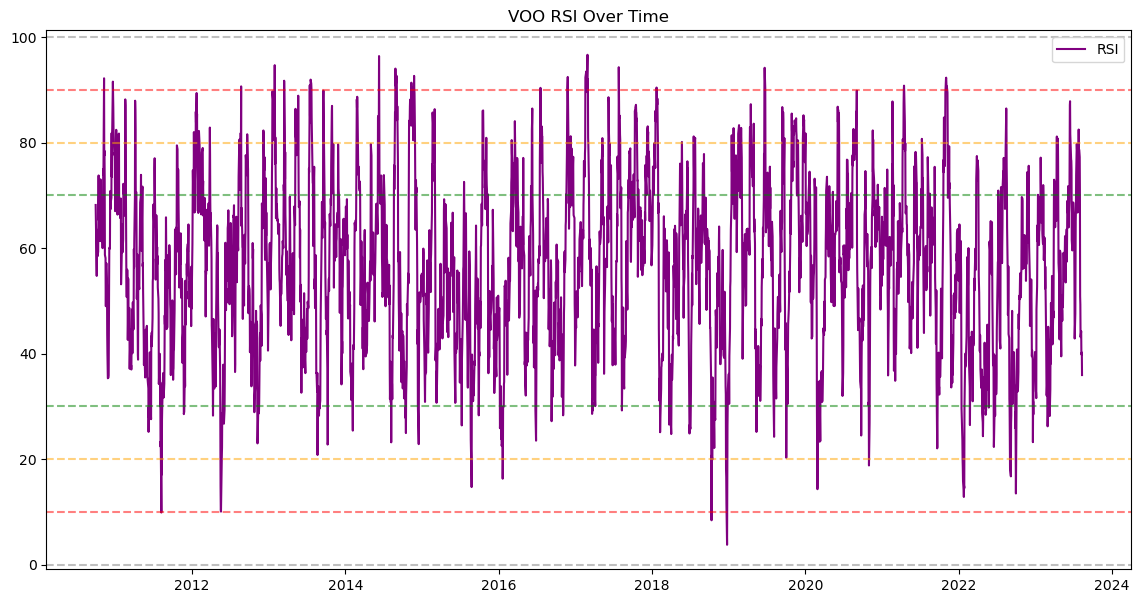
1 101.558845 0.010320 0.005734

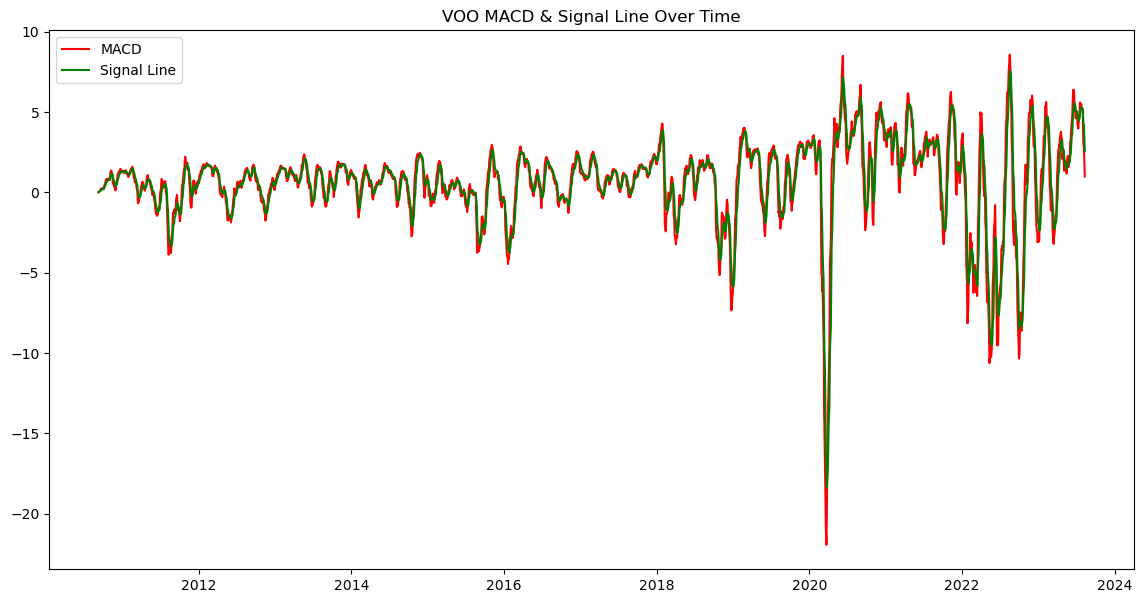
2 102.098195 0.052843 0.025041

3 102.361483 0.070166 0.040327

4 102.579129 0.088448 0.054642







Date Open High Low Close Adj Close \

3253 2003-08-18 47.299999 47.810001 47.299999 47.764999 32.965542

3254 2003-08-19 48.000000 48.000000 47.674999 47.939999 33.086315

3255 2003-08-20 47.849998 48.070000 47.750000 47.935001 33.082870

3256 2003-08-21 48.145000 48.355000 47.980000 48.150002 33.231251

3257 2003-08-22 48.544998 48.549999 47.605000 47.650002 32.886177

Volume Symbol Symbol\_Name SMA30 SMA100 RSI \

3253 102400 VTI Vanguard Total Stock Market ETF NaN NaN NaN

3254 287000 VTI Vanguard Total Stock Market ETF NaN NaN NaN

3255 208400 VTI Vanguard Total Stock Market ETF NaN NaN NaN

3256 199800 VTI Vanguard Total Stock Market ETF NaN NaN NaN

3257 202800 VTI Vanguard Total Stock Market ETF NaN NaN NaN

EMA12 EMA26 MACD Signal\_Line

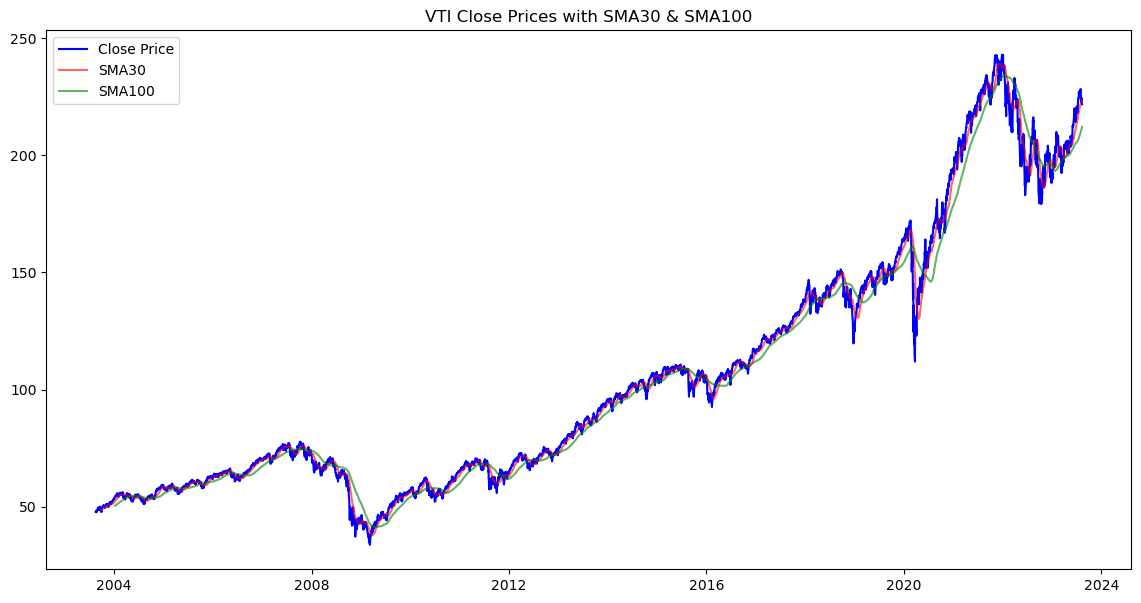
3253 47.764999 47.764999 0.000000 0.000000

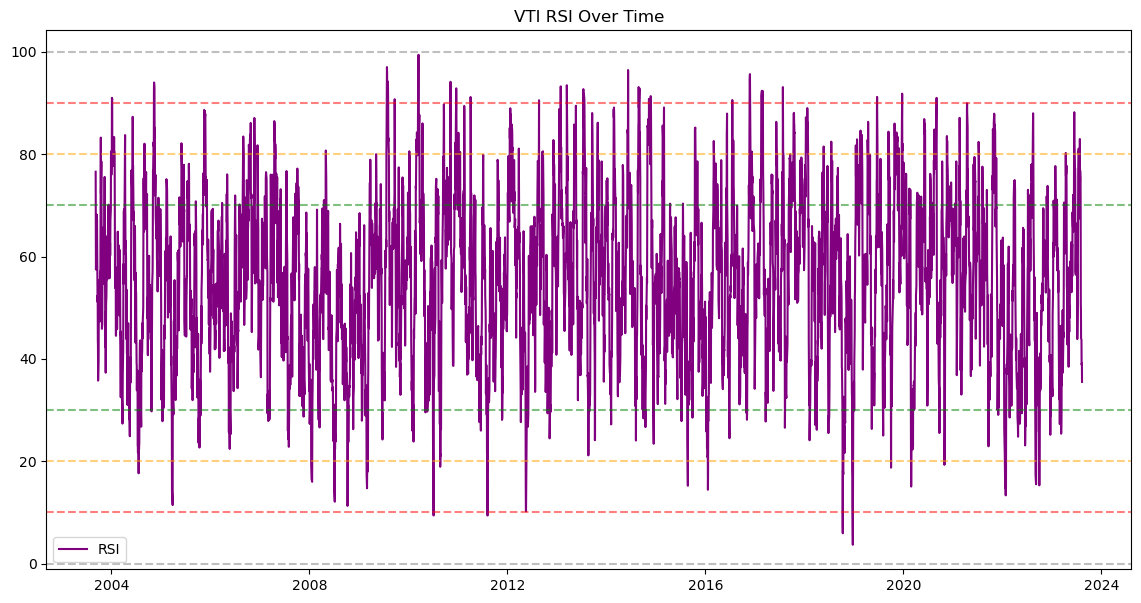
3254 47.859791 47.855864 0.003926 0.002181

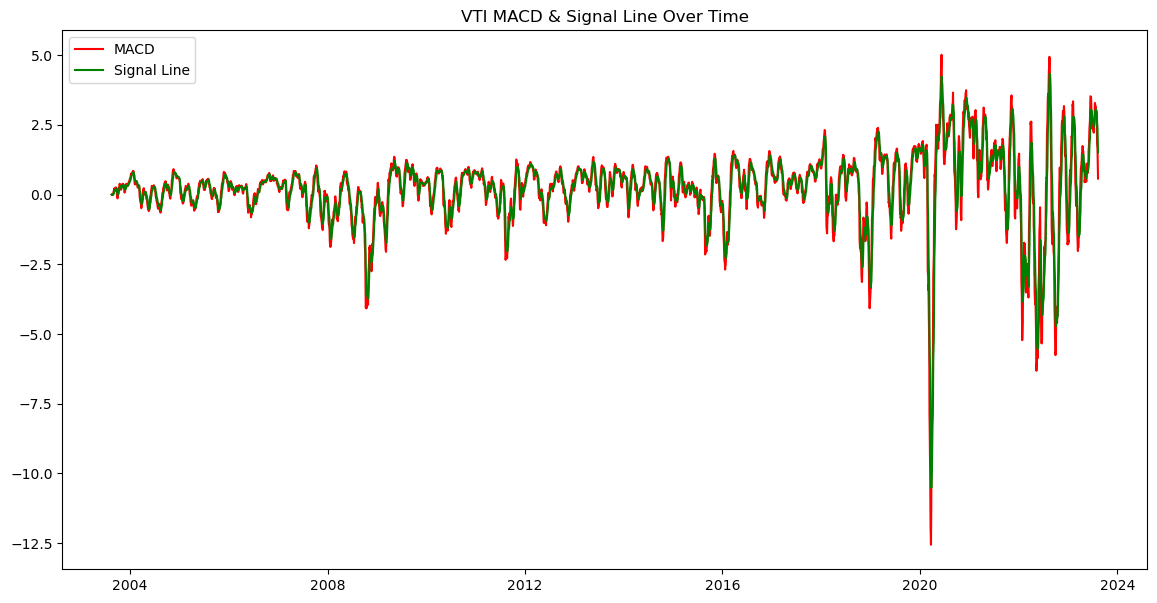
3255 47.889145 47.884298 0.004848 0.003274

3256 47.971487 47.958577 0.012911 0.006539

3257 47.884141 47.887017 -0.002876 0.003738







Date Open High Low Close Adj Close \

8284 2004-01-30 48.740002 49.240002 48.740002 49.080002 41.682972

8285 2004-02-02 49.099998 49.119999 48.840000 49.119999 41.716930

8286 2004-02-03 48.799999 48.820000 48.619999 48.820000 41.462151

8287 2004-02-04 47.500000 47.500000 47.349998 47.349998 40.213692

8288 2004-02-05 47.700001 47.700001 47.349998 47.590000 40.417534

Volume Symbol Symbol\_Name SMA30 SMA100 RSI \

8284 117600 VGT Vanguard Information Technology ETF NaN NaN NaN

8285 65400 VGT Vanguard Information Technology ETF NaN NaN NaN

8286 231100 VGT Vanguard Information Technology ETF NaN NaN NaN

8287 51000 VGT Vanguard Information Technology ETF NaN NaN NaN

8288 2600 VGT Vanguard Information Technology ETF NaN NaN NaN

EMA12 EMA26 MACD Signal\_Line

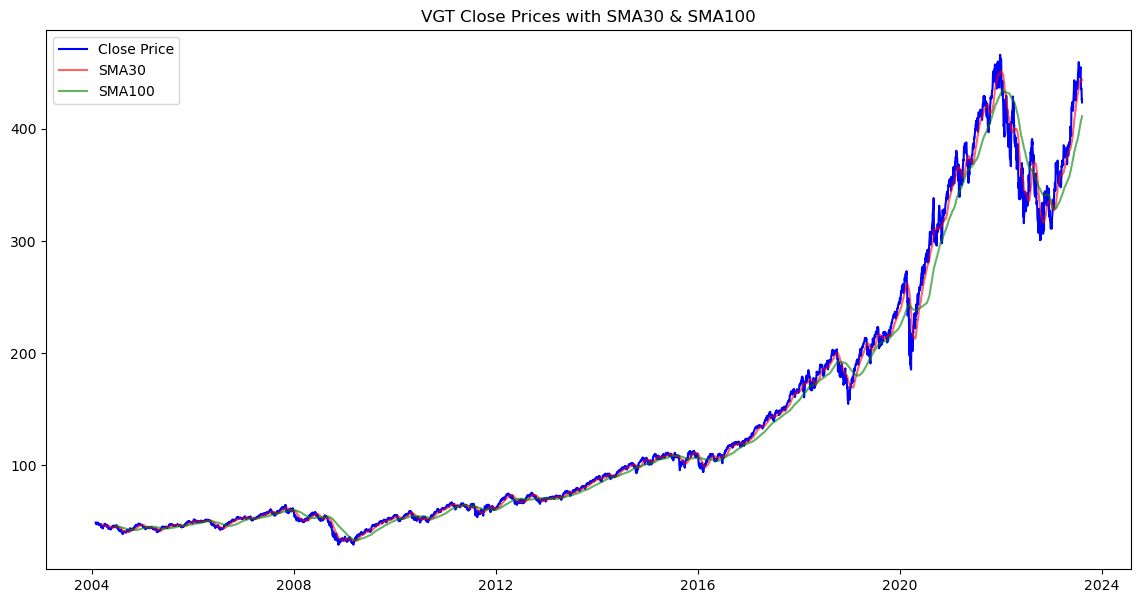
8284 49.080002 49.080002 0.000000 0.000000

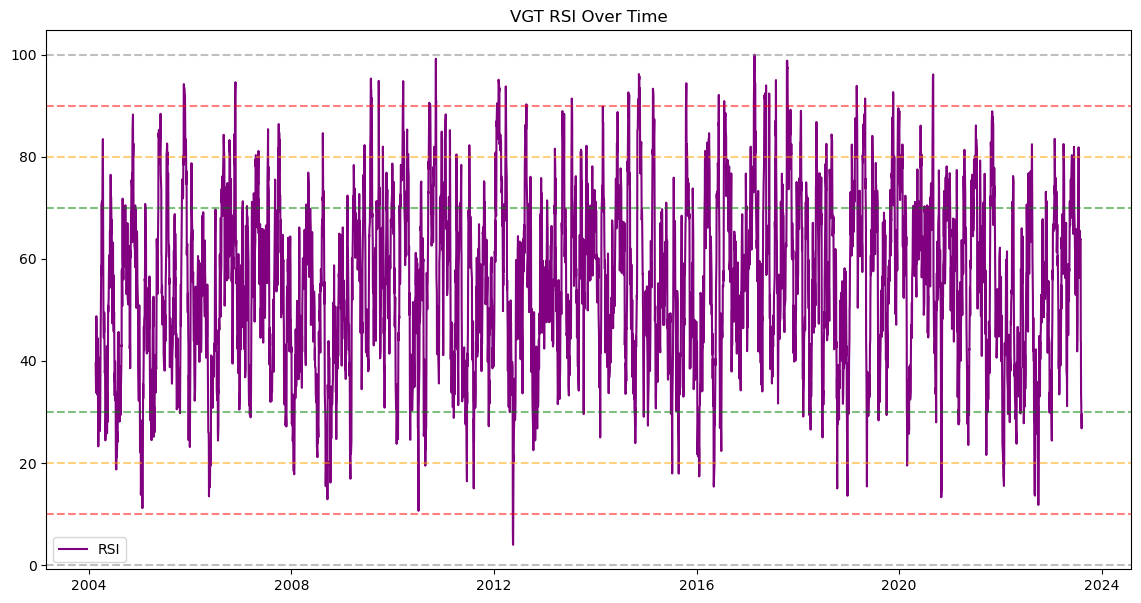
8285 49.101667 49.100770 0.000897 0.000499

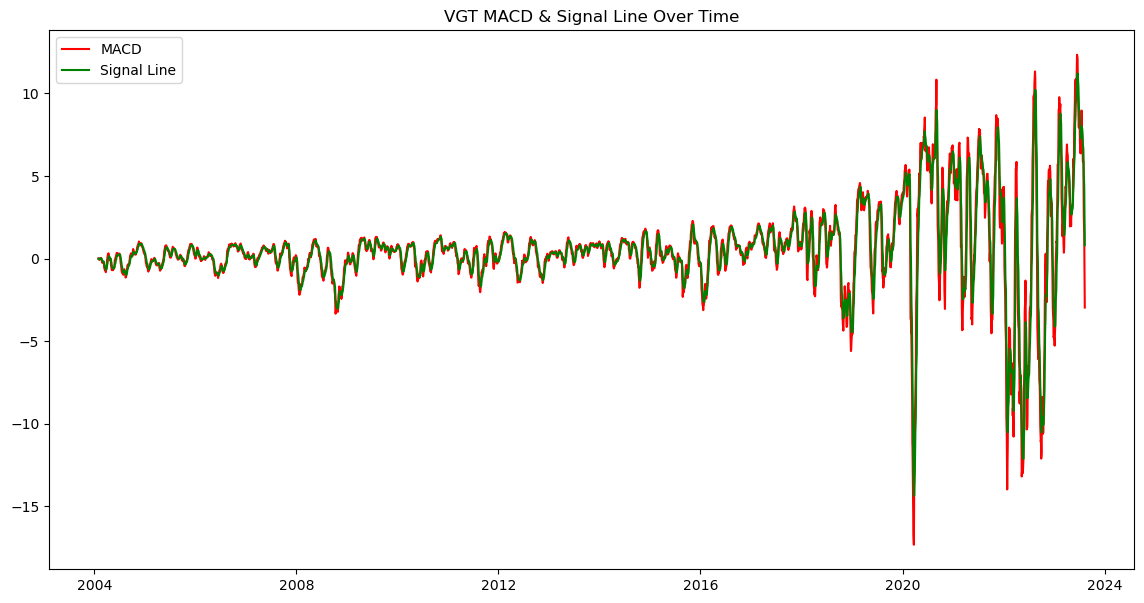
8286 48.991732 48.999892 -0.008160 -0.003050

8287 48.473501 48.538654 -0.065153 -0.024087

8288 48.233457 48.318657 -0.085200 -0.042267







Date Open High Low Close Adj Close \

13201 2004-01-30 49.520000 49.959999 49.520000 49.869999 32.805748

13202 2004-02-02 49.980000 50.380001 49.889999 50.090000 32.950459

13203 2004-02-03 50.049999 50.169998 50.009998 50.139999 32.983353

13204 2004-02-04 49.919998 49.919998 49.680000 49.680000 32.680733

13205 2004-02-05 49.660000 49.770000 49.490002 49.770000 32.739948

Volume Symbol Symbol\_Name SMA30 SMA100 RSI EMA12 \

13201 416600 VFH Vanguard Financials ETF NaN NaN NaN 49.869999

13202 2300 VFH Vanguard Financials ETF NaN NaN NaN 49.989166

13203 1400 VFH Vanguard Financials ETF NaN NaN NaN 50.048036

13204 3400 VFH Vanguard Financials ETF NaN NaN NaN 49.931862

13205 1000 VFH Vanguard Financials ETF NaN NaN NaN 49.887885

EMA26 MACD Signal\_Line

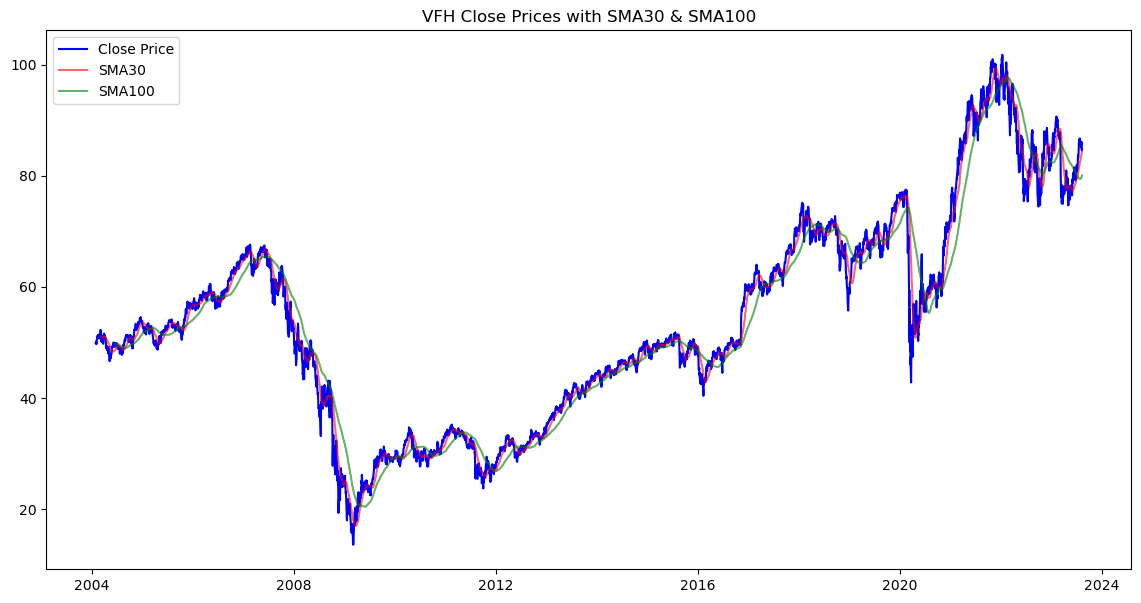
13201 49.869999 0.000000 0.000000

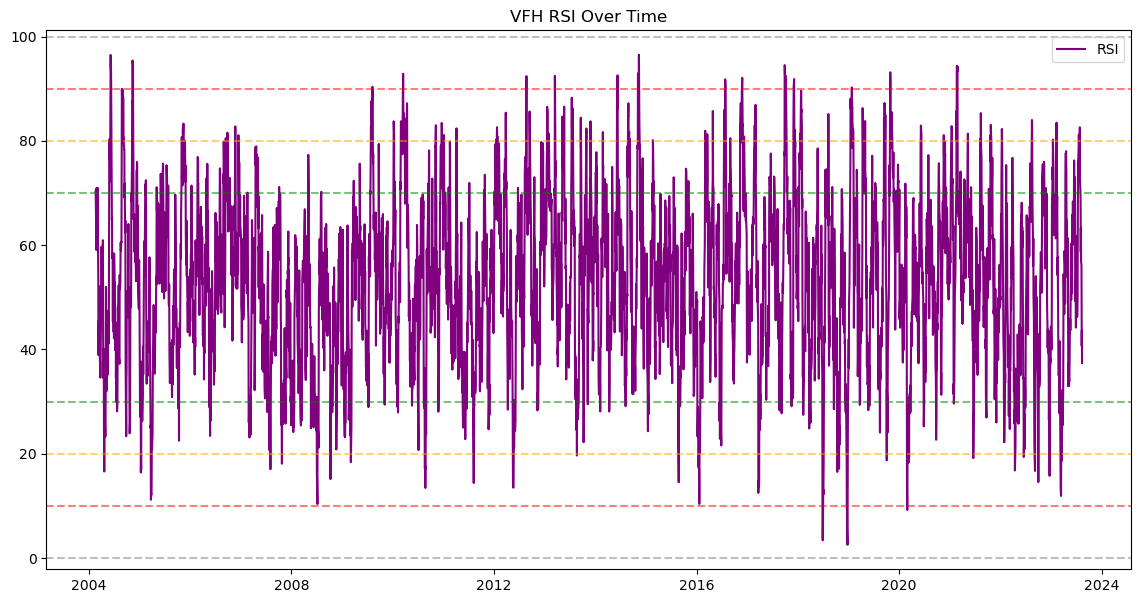
13202 49.984230 0.004936 0.002742

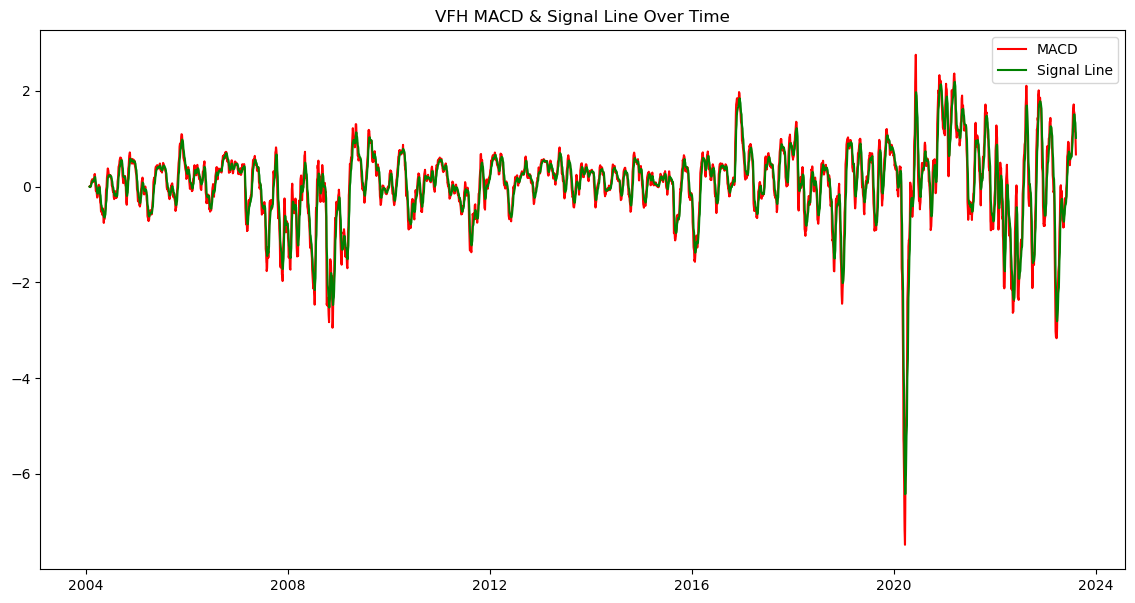
13203 50.040197 0.007840 0.004831

13204 49.939501 -0.007640 0.000607

13205 49.900193 -0.012309 -0.003235







Date Open High Low Close Adj Close \

18118 2004-01-30 49.869999 50.049999 49.750000 50.049999 38.570023

18119 2004-02-02 50.099998 50.799999 49.980000 50.570000 38.970745

18120 2004-02-03 50.549999 50.889999 50.500000 50.889999 39.217358

18121 2004-02-04 50.779999 51.119999 50.759998 50.930000 39.248188

18122 2004-02-05 50.930000 50.930000 50.450001 50.660000 39.040112

Volume Symbol Symbol\_Name SMA30 SMA100 RSI EMA12 \

18118 28700 VHT Vanguard Health Care ETF NaN NaN NaN 50.049999

18119 8700 VHT Vanguard Health Care ETF NaN NaN NaN 50.331666

18120 7300 VHT Vanguard Health Care ETF NaN NaN NaN 50.549584

18121 8600 VHT Vanguard Health Care ETF NaN NaN NaN 50.669666

18122 16300 VHT Vanguard Health Care ETF NaN NaN NaN 50.667040

EMA26 MACD Signal\_Line

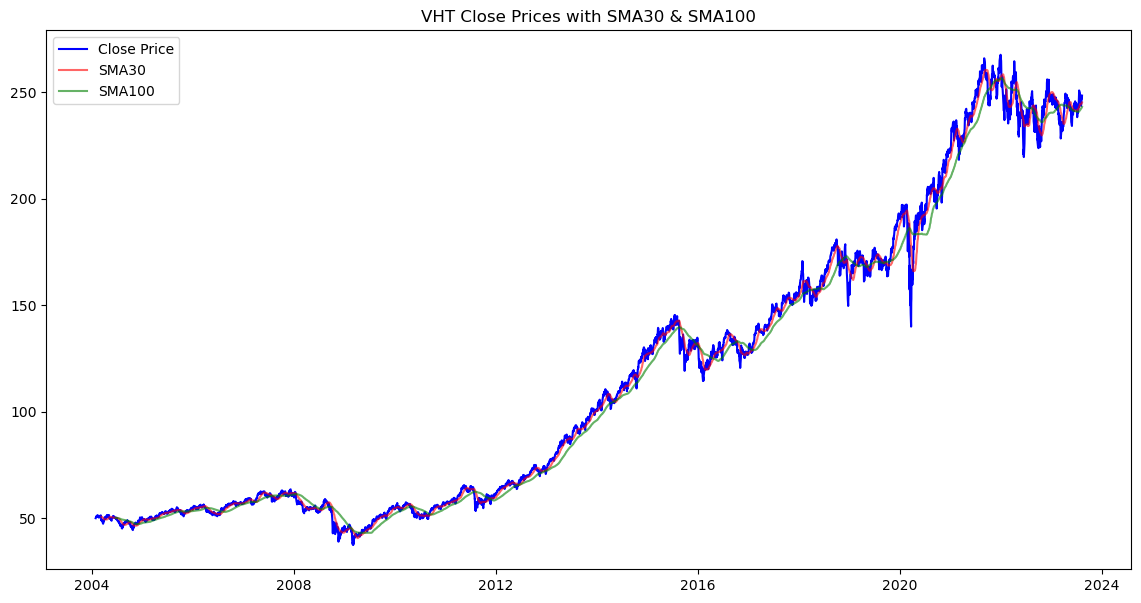
18118 50.049999 0.000000 0.000000

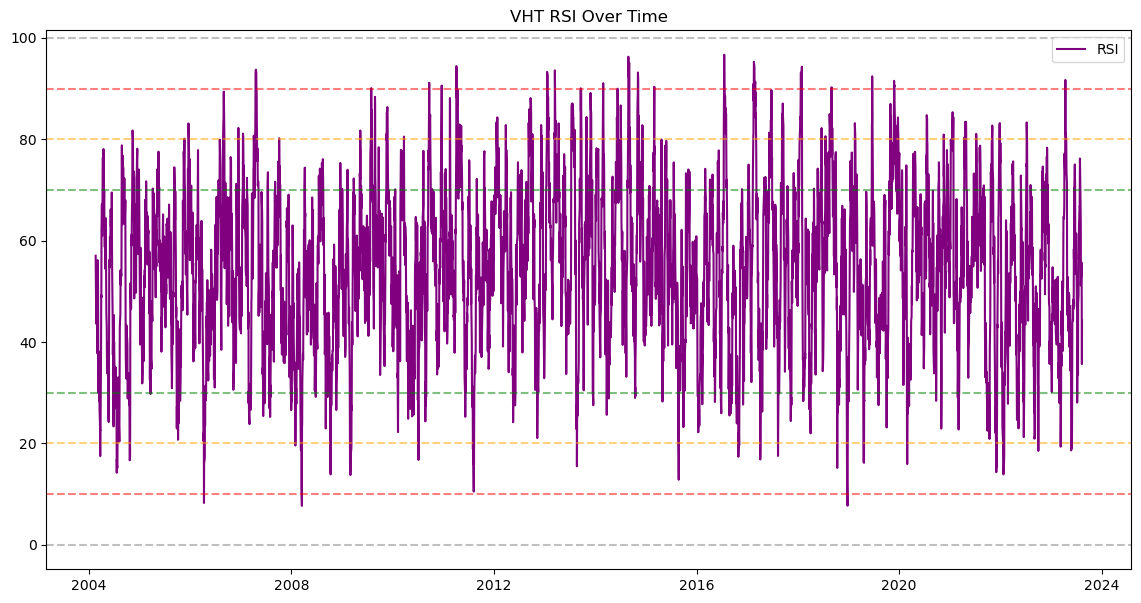
18119 50.319999 0.011667 0.006481

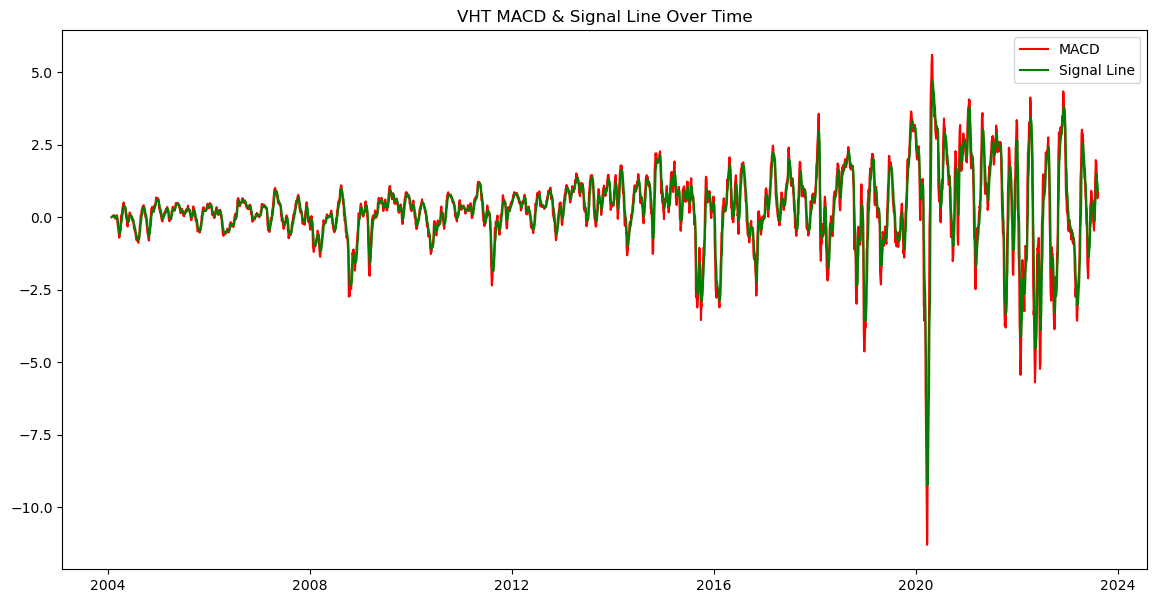
18120 50.524795 0.024789 0.013984

18121 50.638073 0.031594 0.019950

18122 50.643158 0.023882 0.021120







Date Open High Low Close Adj Close \

23035 2004-09-29 48.889999 49.270000 48.889999 49.200001 37.006519

23036 2004-09-30 49.279999 49.610001 49.279999 49.610001 37.314907

23037 2004-10-01 49.610001 49.610001 49.610001 49.610001 37.314907

23038 2004-10-04 50.590000 50.590000 50.529999 50.529999 38.006908

23039 2004-10-05 50.270000 50.299999 50.270000 50.299999 37.833912

Volume Symbol Symbol\_Name SMA30 SMA100 RSI EMA12 \

23035 25800 VIS Vanguard Industrials ETF NaN NaN NaN 49.200001

23036 400 VIS Vanguard Industrials ETF NaN NaN NaN 49.422084

23037 0 VIS Vanguard Industrials ETF NaN NaN NaN 49.495428

23038 3400 VIS Vanguard Industrials ETF NaN NaN NaN 49.822001

23039 700 VIS Vanguard Industrials ETF NaN NaN NaN 49.951872

EMA26 MACD Signal\_Line

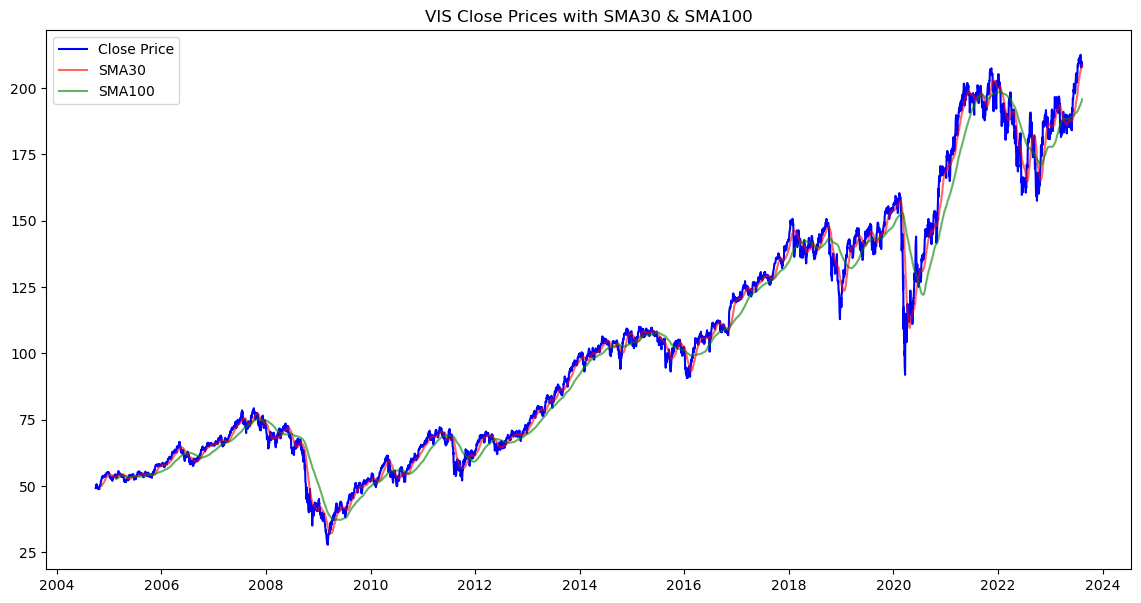
23035 49.200001 0.000000 0.000000

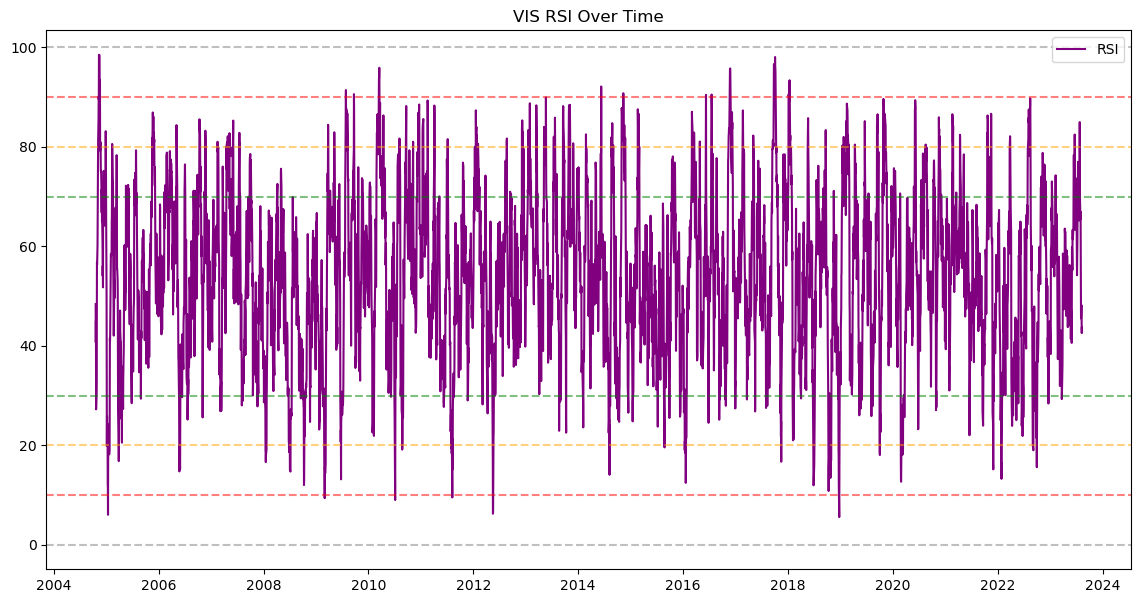
23036 49.412885 0.009199 0.005110

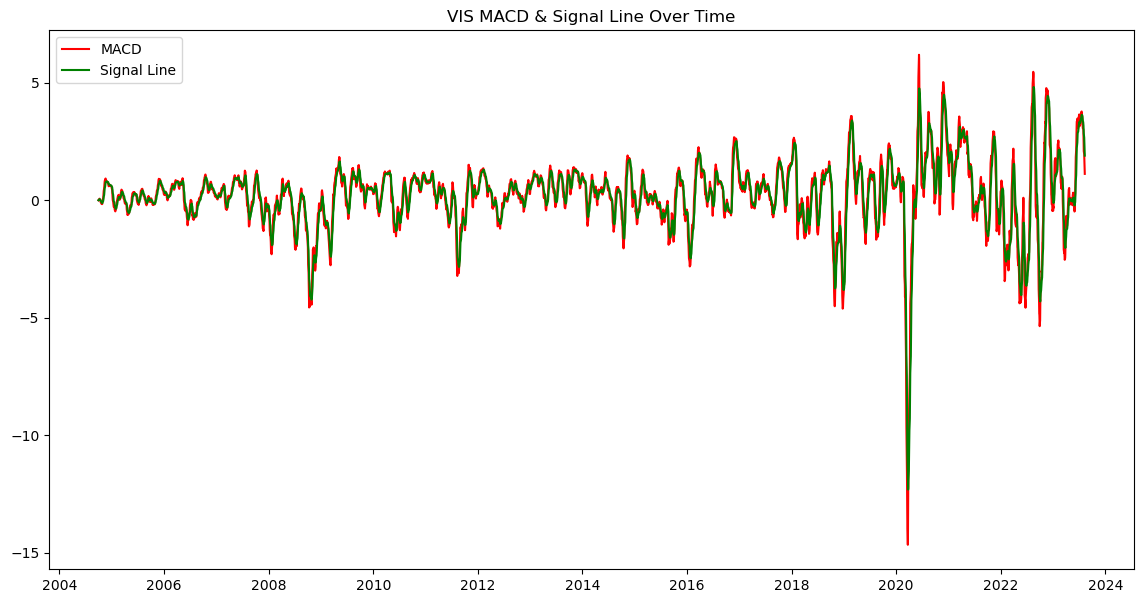
23037 49.483707 0.011721 0.007820

23038 49.776204 0.045797 0.020685

23039 49.897675 0.054197 0.030654







Date Open High Low Close Adj Close \

27785 2004-01-30 48.630001 49.220001 48.630001 49.220001 34.534679

27786 2004-02-02 49.200001 49.200001 49.049999 49.049999 34.415405

27787 2004-02-03 48.900002 48.900002 48.700001 48.700001 34.169823

27788 2004-02-04 48.549999 48.599998 48.299999 48.299999 33.889175

27789 2004-02-05 48.500000 48.889999 48.500000 48.869999 34.289101

Volume Symbol Symbol\_Name SMA30 SMA100 RSI EMA12 \

27785 8700 VAW Vanguard Materials ETF NaN NaN NaN 49.220001

27786 1300 VAW Vanguard Materials ETF NaN NaN NaN 49.127917

27787 6900 VAW Vanguard Materials ETF NaN NaN NaN 48.960901

27788 7100 VAW Vanguard Materials ETF NaN NaN NaN 48.752280

27789 20900 VAW Vanguard Materials ETF NaN NaN NaN 48.784264

EMA26 MACD Signal\_Line

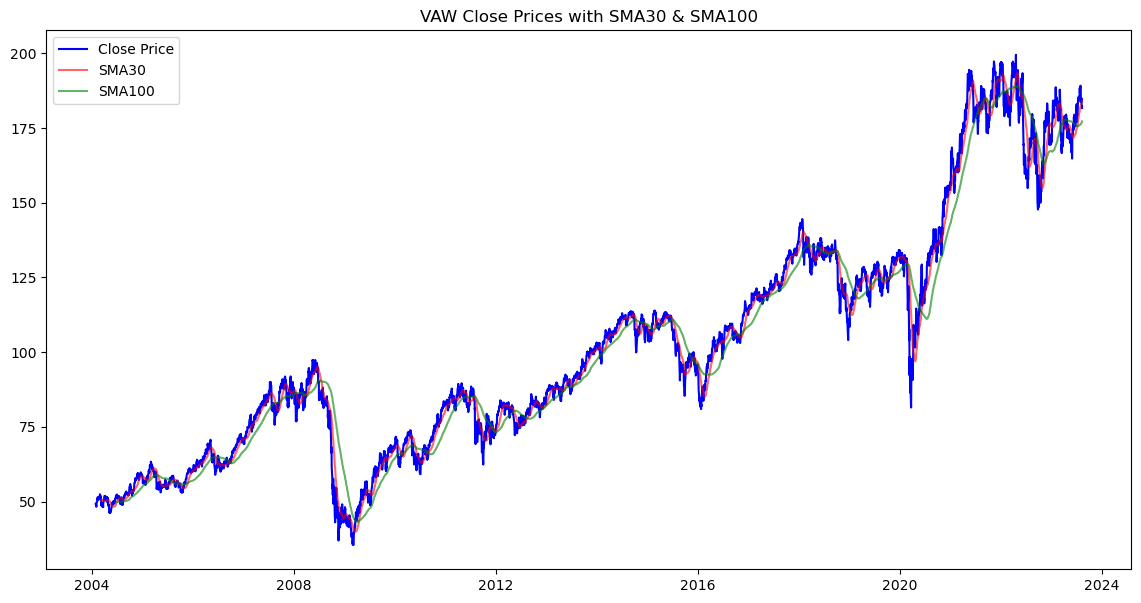
27785 49.220001 0.000000 0.000000

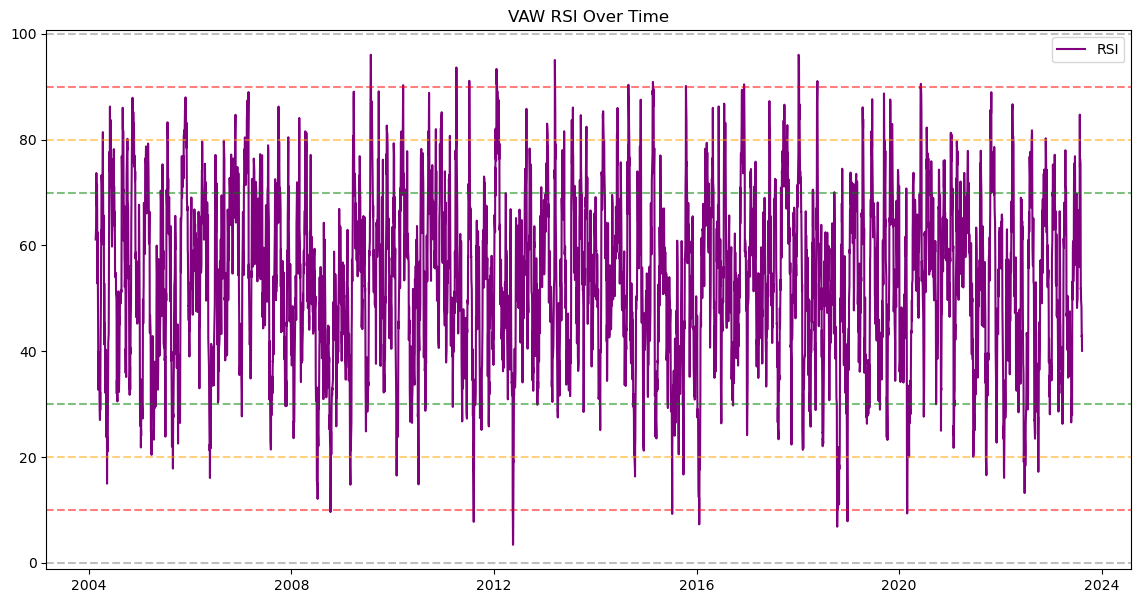
27786 49.131731 -0.003814 -0.002119

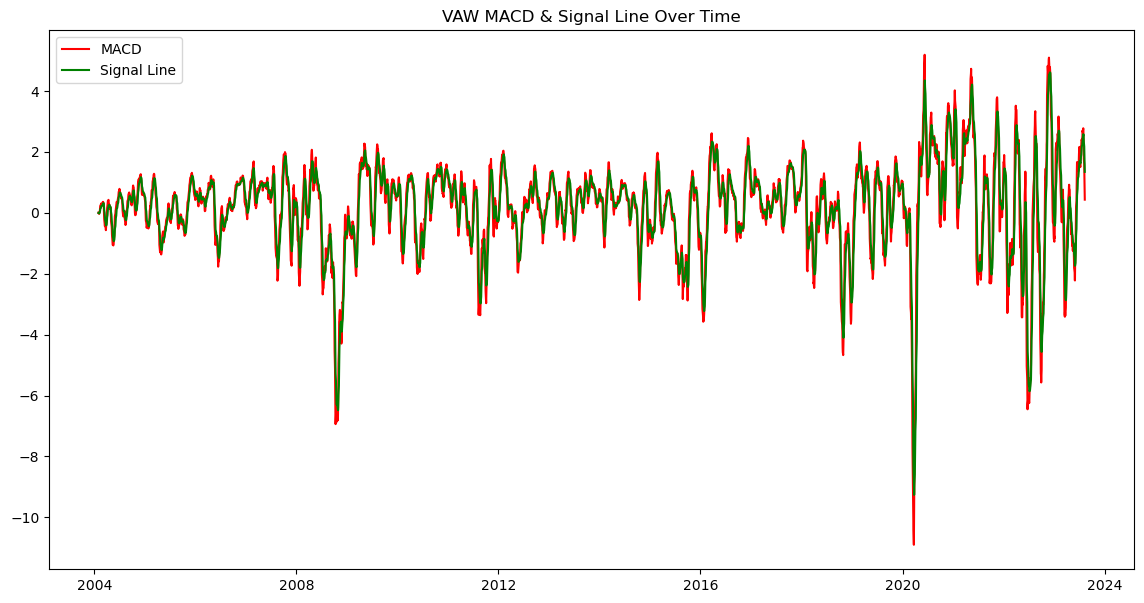
27787 48.976614 -0.015713 -0.007690

27788 48.787462 -0.035182 -0.017003

27789 48.806603 -0.022339 -0.018591







Date Open High Low Close Adj Close \

32702 2004-09-29 49.849998 49.990002 49.840000 49.849998 21.955471

32703 2004-09-30 50.000000 50.330002 49.990002 50.250000 22.131649

32704 2004-10-01 50.349998 51.250000 50.250000 51.180000 22.541243

32705 2004-10-04 51.389999 51.549999 51.389999 51.400002 22.638138

32706 2004-10-05 51.400002 51.500000 51.349998 51.389999 22.633730

Volume Symbol Symbol\_Name SMA30 SMA100 RSI EMA12 \

32702 205800 VNQ Vanguard Real Estate ETF NaN NaN NaN 49.849998

32703 27900 VNQ Vanguard Real Estate ETF NaN NaN NaN 50.066666

32704 129800 VNQ Vanguard Real Estate ETF NaN NaN NaN 50.501201

32705 8100 VNQ Vanguard Real Estate ETF NaN NaN NaN 50.784917

32706 11900 VNQ Vanguard Real Estate ETF NaN NaN NaN 50.949316

EMA26 MACD Signal\_Line

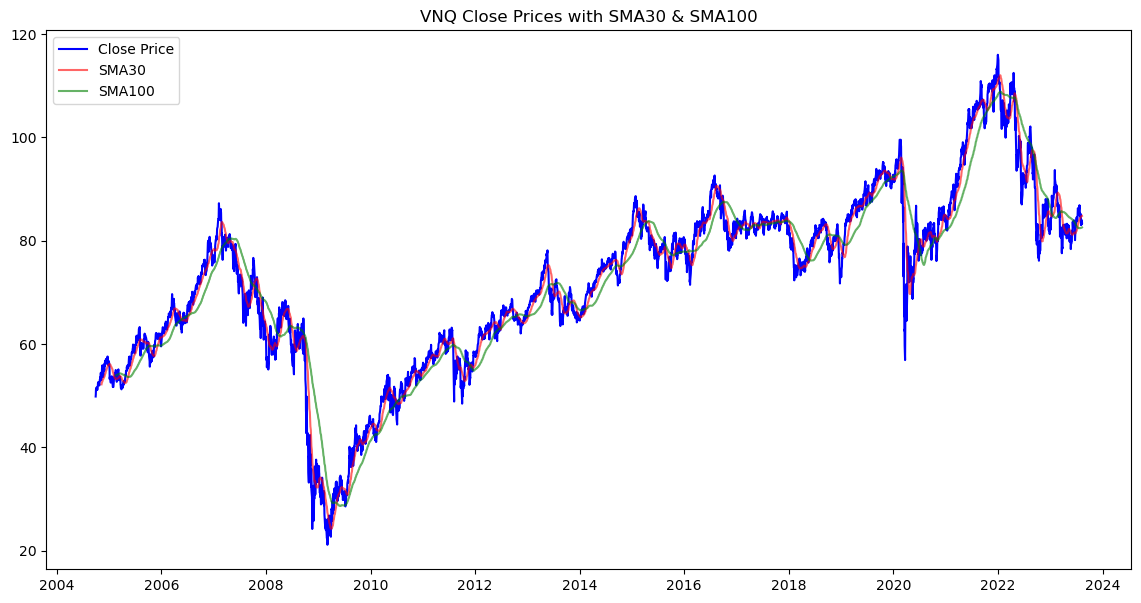
32702 49.849998 0.000000 0.000000

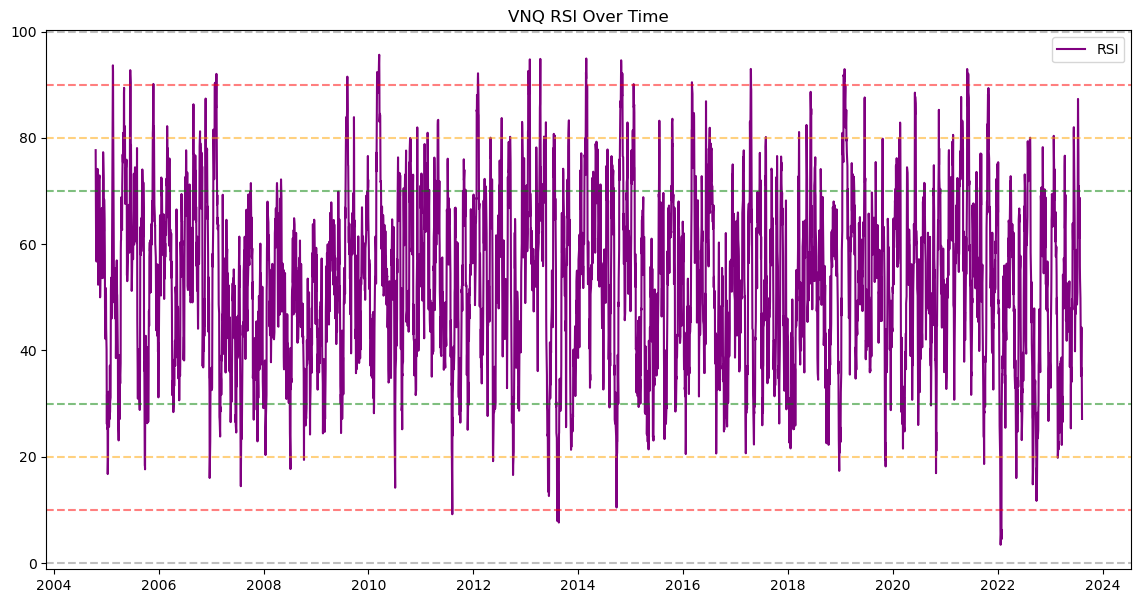
32703 50.057692 0.008974 0.004986

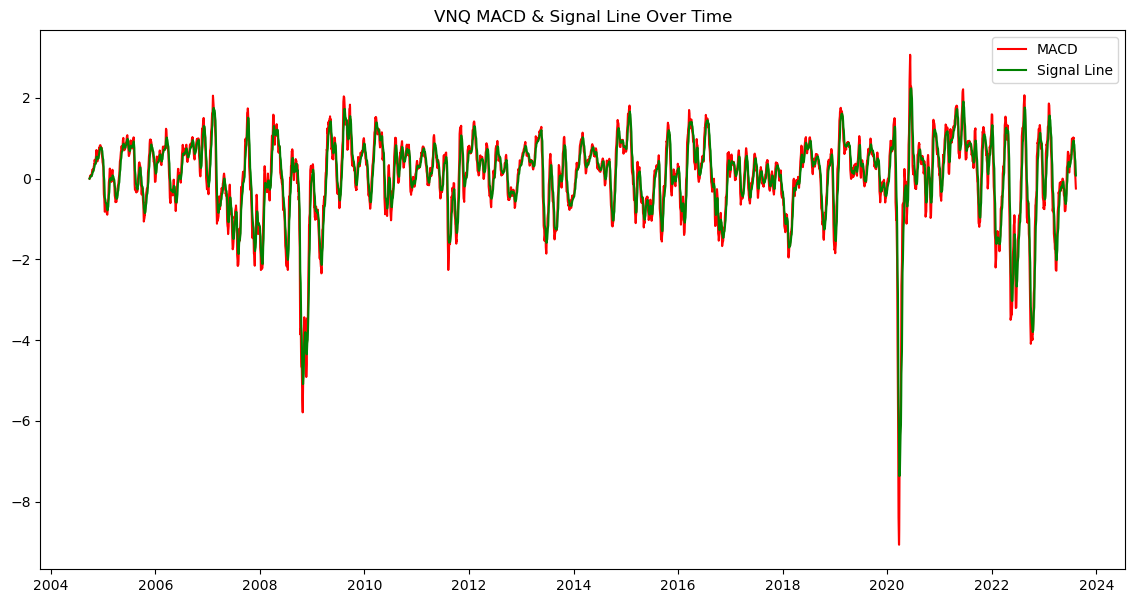
32704 50.460926 0.040274 0.019448

32705 50.723451 0.061466 0.033682

32706 50.878026 0.071290 0.044869







In [133]:

*# Iterate over each symbol*

**for** symbol, data **in** data\_by\_symbol**.**items():

print(f"Processing data for symbol: {symbol}")

*# Split data into train and test set*

train\_data, test\_data **=** split\_data(data)

*# Build ARIMA model*

model\_arima **=** ARIMA(train\_data['Close'], order**=**(5,1,0))

model\_arima\_fit **=** model\_arima**.**fit()

*# Make predictions*

predictions\_arima **=** model\_arima\_fit**.**forecast(steps**=**5) *# Forecast only for 10 days*

*# Evaluate ARIMA model*

mse\_arima **=** mean\_squared\_error(test\_data['Close'], predictions\_arima)

*# Prepare data for LSTM model*

X\_train, y\_train, X\_test, y\_test, scaler **=** prepare\_data\_for\_lstm(data)

*# Build and train LSTM model*

model\_lstm **=** build\_and\_train\_lstm(X\_train, y\_train)

*# Make predictions with LSTM model*

predictions\_lstm\_scaled **=** model\_lstm**.**predict(X\_test)

predictions\_lstm **=** scaler**.**inverse\_transform(predictions\_lstm\_scaled)

*# Inverse scale y\_test*

y\_test\_inv **=** scaler**.**inverse\_transform(y\_test**.**reshape(**-**1, 1))

*# Evaluate LSTM model*

mse\_lstm **=** mean\_squared\_error(y\_test, predictions\_lstm\_scaled)

*# Store results in dictionary*

results[symbol] **=** {

'MSE\_ARIMA': mse\_arima,

'Predictions\_ARIMA': predictions\_arima,

'Real\_Values': test\_data['Close']**.**values,

'MSE\_LSTM': mse\_lstm,

'Predictions\_LSTM\_Diff': predictions\_lstm,

'LSTM\_Real\_Values': y\_test\_inv,

'SMA30': test\_data['SMA30'],

'SMA100': test\_data['SMA100']

}

Processing data for symbol: VOO

3246/3246 [==============================] - 9s 2ms/step - loss: 0.0132

1/1 [==============================] - 0s 480ms/step

Processing data for symbol: VTI

5024/5024 [==============================] - 12s 2ms/step - loss: 0.0105

1/1 [==============================] - 0s 476ms/step

Processing data for symbol: VGT

4910/4910 [==============================] - 12s 2ms/step - loss: 0.0104

1/1 [==============================] - 0s 464ms/step

Processing data for symbol: VFH

4910/4910 [==============================] - 12s 2ms/step - loss: 0.0138

1/1 [==============================] - 0s 460ms/step

Processing data for symbol: VHT

4910/4910 [==============================] - 14s 2ms/step - loss: 0.0098

1/1 [==============================] - 0s 488ms/step

Processing data for symbol: VIS

4743/4743 [==============================] - 13s 2ms/step - loss: 0.0133

1/1 [==============================] - 0s 487ms/step

Processing data for symbol: VAW

4910/4910 [==============================] - 14s 3ms/step - loss: 0.0204

1/1 [==============================] - 1s 552ms/step

Processing data for symbol: VNQ

4743/4743 [==============================] - 14s 3ms/step - loss: 0.0118

1/1 [==============================] - 1s 552ms/step

In [141]:

**for** symbol, result **in** results**.**items():

print(f"Processing data for symbol: {symbol}")

*# Reset indexes*

result['Predictions\_ARIMA'] **=** result['Predictions\_ARIMA']**.**reset\_index(drop**=True**)

result['SMA30'] **=** result['SMA30']**.**reset\_index(drop**=True**)

result['SMA100'] **=** result['SMA100']**.**reset\_index(drop**=True**)

*# Inverse transform the scaled predictions*

predictions\_lstm\_original **=** scaler**.**inverse\_transform(predictions\_lstm\_scaled)

print("LSTM Model MSE:", mse\_lstm)

print("Converted Predicted LSTM Values (Original Scale):", predictions\_lstm\_original)

result['Predictions\_LSTM'] **=** scaler**.**inverse\_transform(result['Predictions\_LSTM\_Diff'])

*# Fetching Technical Indicators for the symbol*

stock\_data **=** df[df['Symbol'] **==** symbol]

print(f"\nResults for symbol: {symbol}")

print(f"ARIMA Model MSE: {result['MSE\_ARIMA']}")

print(f"Predicted ARIMA Values: {result['Predictions\_ARIMA']}")

print(f"Real Values: {result['Real\_Values']}")

print(f"LSTM Model MSE: {result['MSE\_LSTM']}")

*# Plot the results*

plt**.**figure(figsize**=**(14, 6))

*# ARIMA predictions*

plt**.**subplot(1, 2, 1)

plt**.**plot(result['Real\_Values'], label**=**'Real Values', color**=**'blue')

plt**.**plot(result['Predictions\_ARIMA'], label**=**'ARIMA Predictions', color**=**'red')

plt**.**plot(result['SMA30'], label**=**'SMA30', color**=**'purple', alpha**=**0.6)

plt**.**plot(result['SMA100'], label**=**'SMA100', color**=**'yellow', alpha**=**0.6)

plt**.**title(f"ARIMA Predictions vs Real Values for {symbol}")

plt**.**legend()

*# LSTM predictions*

plt**.**subplot(1, 2, 2)

plt**.**plot(result['LSTM\_Real\_Values'], label**=**'Real Values', color**=**'blue')

plt**.**plot(result['Predictions\_LSTM']**.**flatten(), label**=**'LSTM Predictions', color**=**'green')

plt**.**title(f"LSTM Predictions vs Real Values for {symbol}")

plt**.**legend()

plt**.**tight\_layout()

plt**.**show()

Processing data for symbol: VOO

LSTM Model MSE: 0.002853327464319741

Converted Predicted LSTM Values (Original Scale): [[-0.00997348]

[-0.01366247]

[-0.01028156]

[-0.01143517]

[-0.01076289]]

Results for symbol: VOO

ARIMA Model MSE: 3.701670529808279

Predicted ARIMA Values: 0 410.684236

1 410.847935

2 410.824920

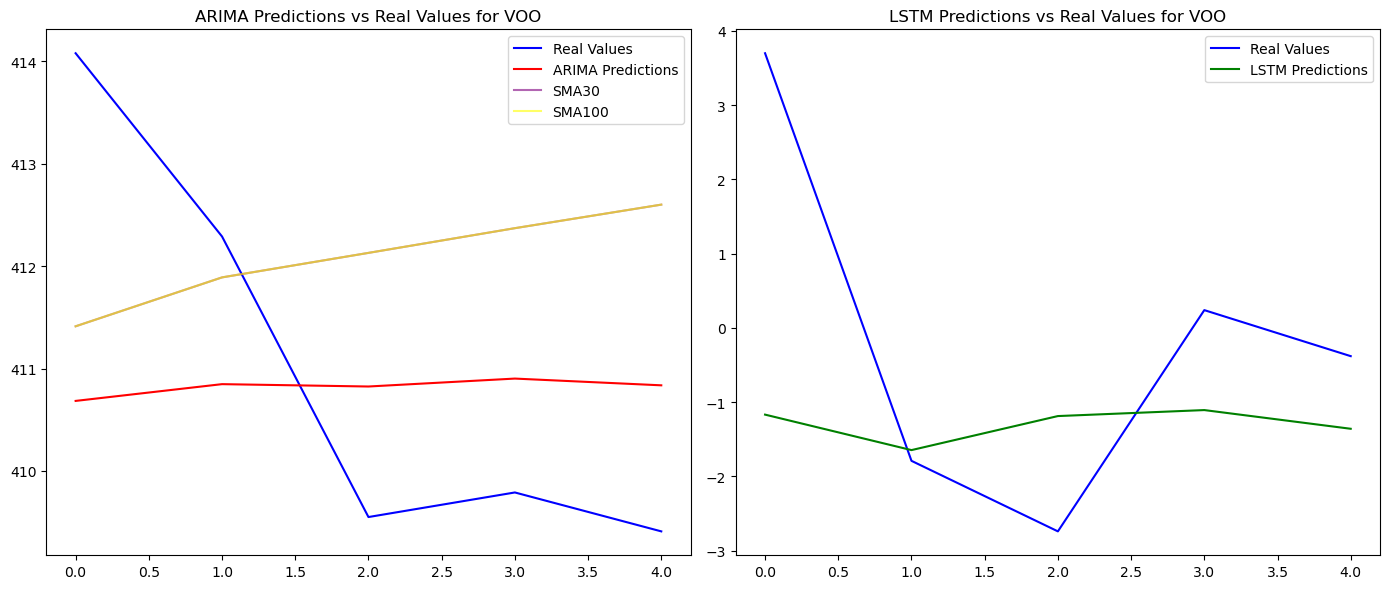
3 410.902353

4 410.836501

Name: predicted\_mean, dtype: float64

Real Values: [414.07998657 412.29000854 409.54998779 409.79000854 409.41000366]

LSTM Model MSE: 0.008000961106193128



Processing data for symbol: VTI

LSTM Model MSE: 0.002853327464319741

Converted Predicted LSTM Values (Original Scale): [[-0.00997348]

[-0.01366247]

[-0.01028156]

[-0.01143517]

[-0.01076289]]

Results for symbol: VTI

ARIMA Model MSE: 1.4065124919625527

Predicted ARIMA Values: 0 223.012564

1 223.108418

2 223.099912

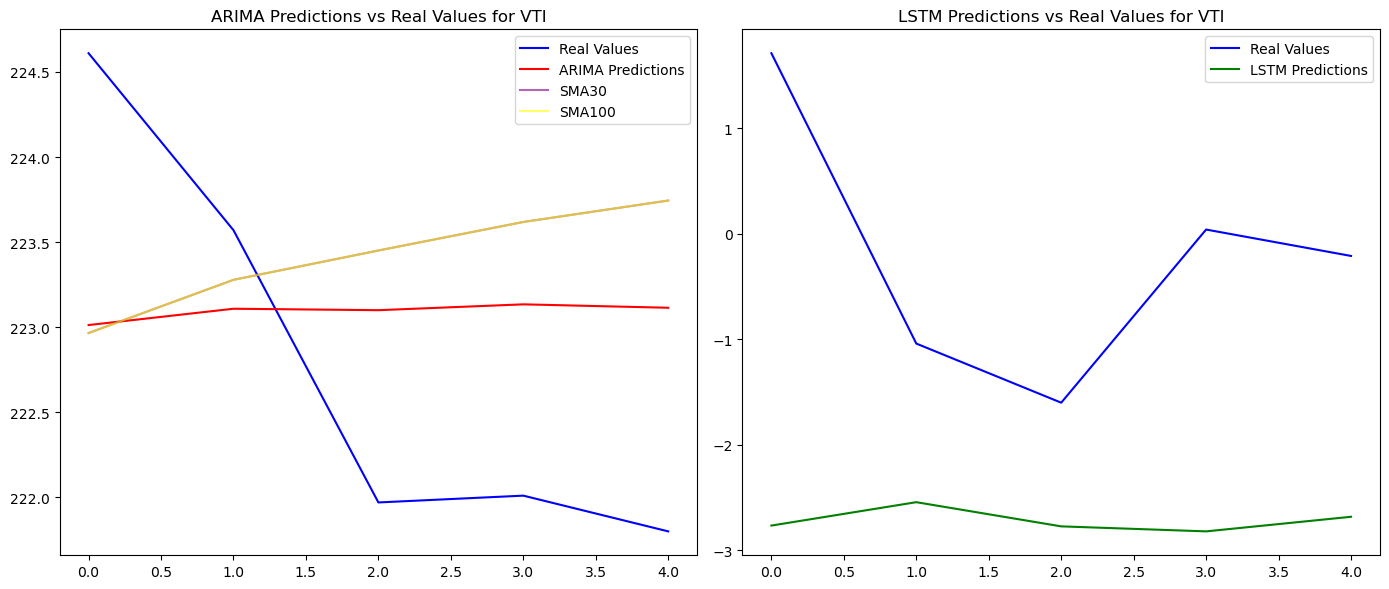
3 223.134252

4 223.114172

Name: predicted\_mean, dtype: float64

Real Values: [224.61000061 223.57000732 221.97000122 222.00999451 221.80000305]

LSTM Model MSE: 0.0077408139769099965



Processing data for symbol: VGT

LSTM Model MSE: 0.002853327464319741

Converted Predicted LSTM Values (Original Scale): [[-0.00997348]

[-0.01366247]

[-0.01028156]

[-0.01143517]

[-0.01076289]]

Results for symbol: VGT

ARIMA Model MSE: 81.62096237412588

Predicted ARIMA Values: 0 436.224064

1 436.711212

2 436.627228

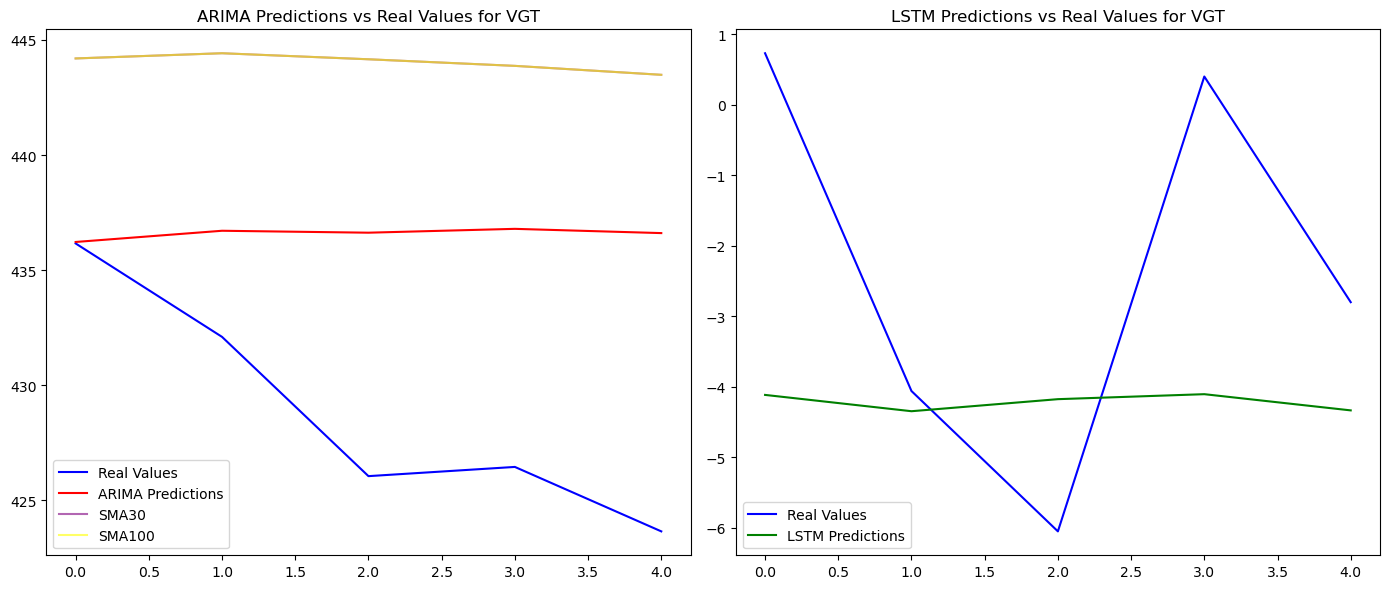
3 436.794374

4 436.609688

Name: predicted\_mean, dtype: float64

Real Values: [436.16000366 432.1000061 426.04998779 426.45001221 423.6499939 ]

LSTM Model MSE: 0.015555655548895717



Processing data for symbol: VFH

LSTM Model MSE: 0.002853327464319741

Converted Predicted LSTM Values (Original Scale): [[-0.00997348]

[-0.01366247]

[-0.01028156]

[-0.01143517]

[-0.01076289]]

Results for symbol: VFH

ARIMA Model MSE: 0.34171179371986543

Predicted ARIMA Values: 0 85.121642

1 85.140615

2 85.140123

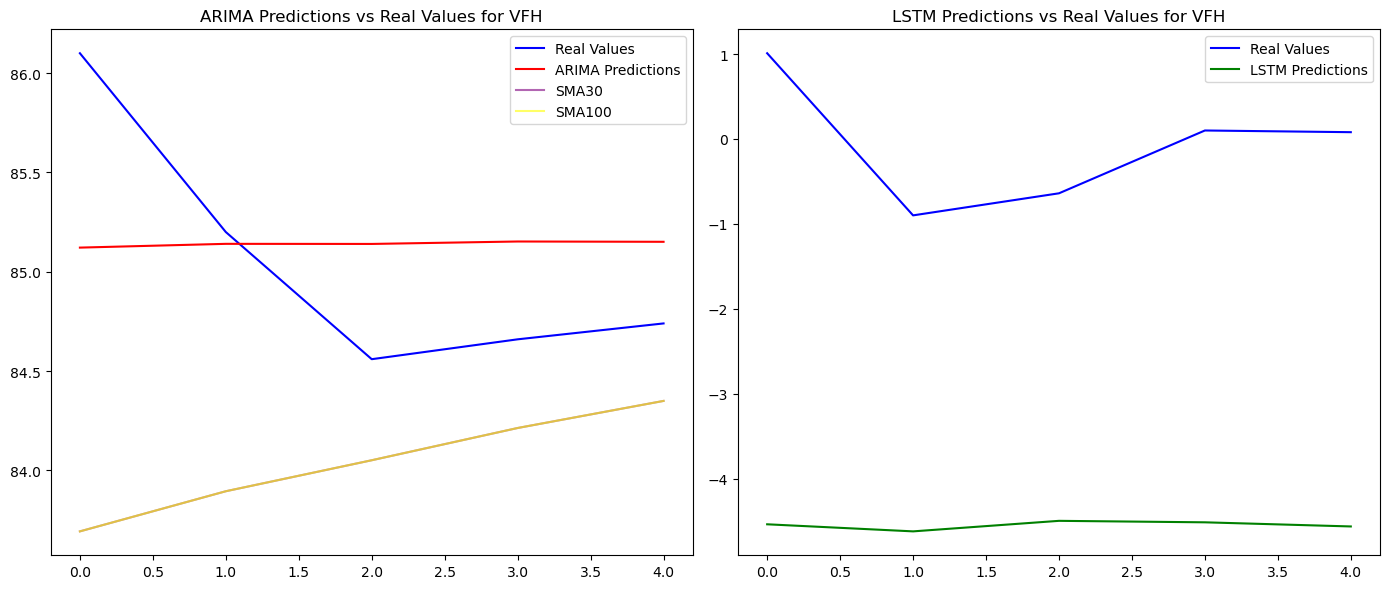
3 85.152408

4 85.150903

Name: predicted\_mean, dtype: float64

Real Values: [86.09999847 85.19999695 84.55999756 84.66000366 84.73999786]

LSTM Model MSE: 0.009350654893075612



Processing data for symbol: VHT

LSTM Model MSE: 0.002853327464319741

Converted Predicted LSTM Values (Original Scale): [[-0.00997348]

[-0.01366247]

[-0.01028156]

[-0.01143517]

[-0.01076289]]

Results for symbol: VHT

ARIMA Model MSE: 13.25384031597224

Predicted ARIMA Values: 0 243.522939

1 243.489191

2 243.555751

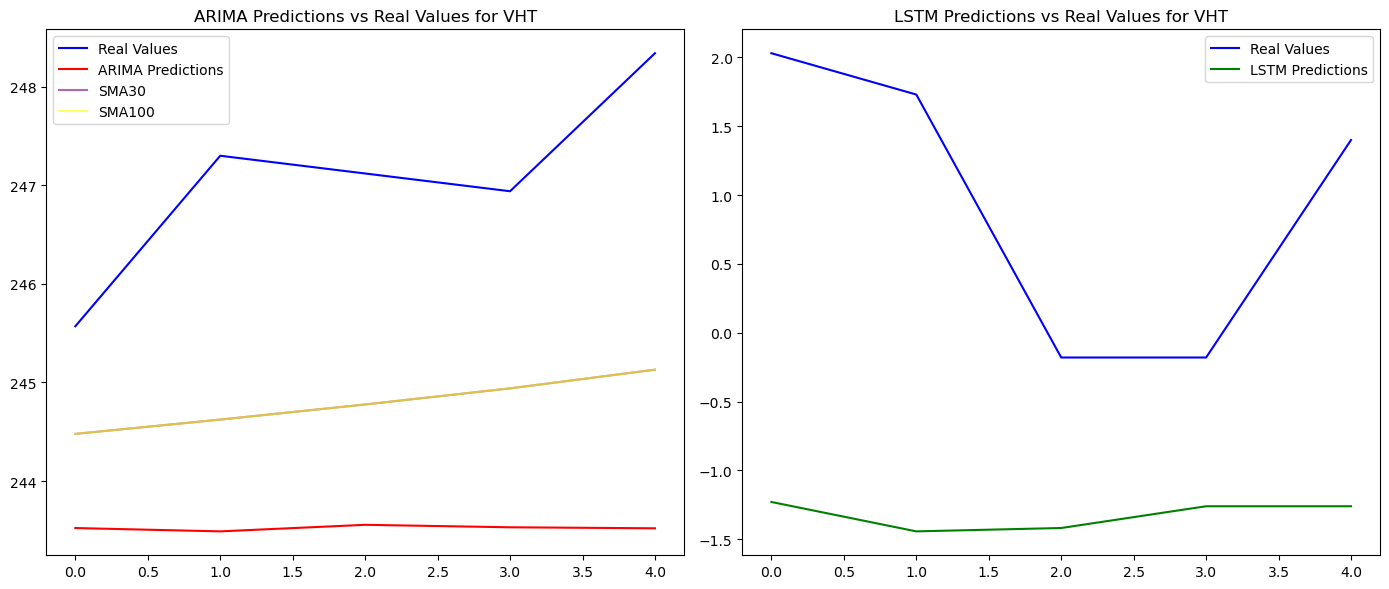
3 243.530701

4 243.520323

Name: predicted\_mean, dtype: float64

Real Values: [245.57000732 247.30000305 247.11999512 246.94000244 248.33999634]

LSTM Model MSE: 0.006288170659602657



Processing data for symbol: VIS

LSTM Model MSE: 0.002853327464319741

Converted Predicted LSTM Values (Original Scale): [[-0.00997348]

[-0.01366247]

[-0.01028156]

[-0.01143517]

[-0.01076289]]

Results for symbol: VIS

ARIMA Model MSE: 1.5855004334100389

Predicted ARIMA Values: 0 207.735803

1 207.813198

2 207.846483

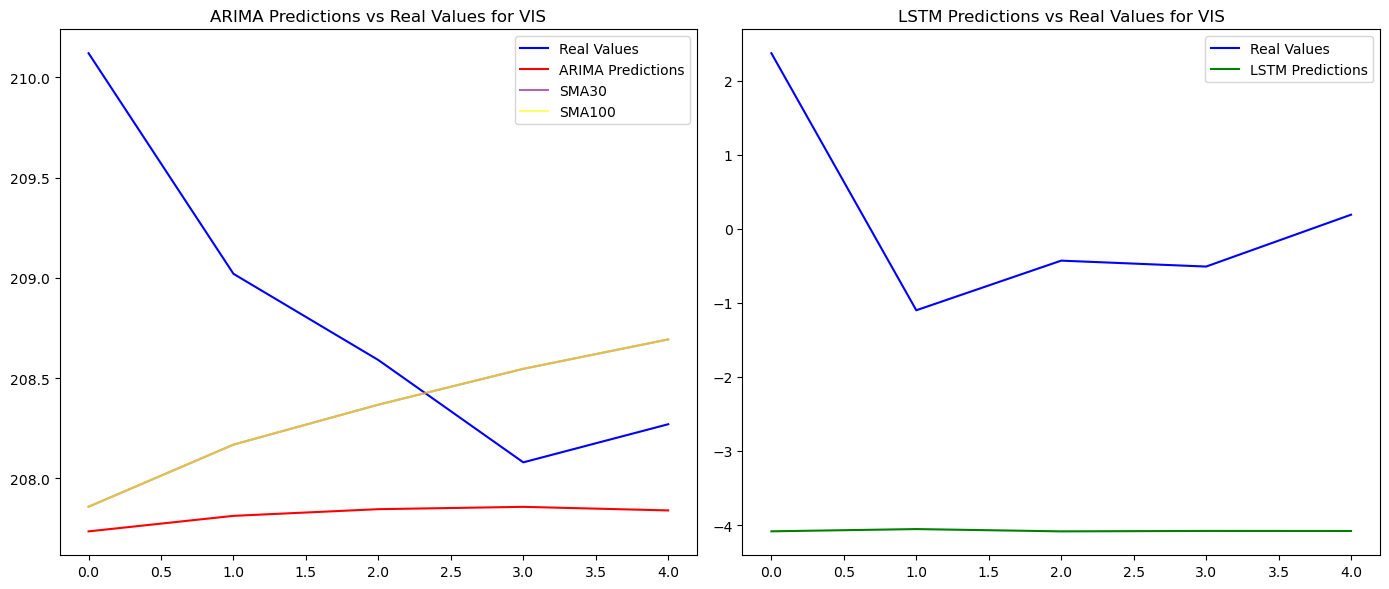
3 207.858084

4 207.840248

Name: predicted\_mean, dtype: float64

Real Values: [210.11999512 209.02000427 208.58999634 208.08000183 208.27000427]

LSTM Model MSE: 0.009863555965632067



Processing data for symbol: VAW

LSTM Model MSE: 0.002853327464319741

Converted Predicted LSTM Values (Original Scale): [[-0.00997348]

[-0.01366247]

[-0.01028156]

[-0.01143517]

[-0.01076289]]

Results for symbol: VAW

ARIMA Model MSE: 2.4601375453510195

Predicted ARIMA Values: 0 183.779027

1 183.865001

2 183.902349

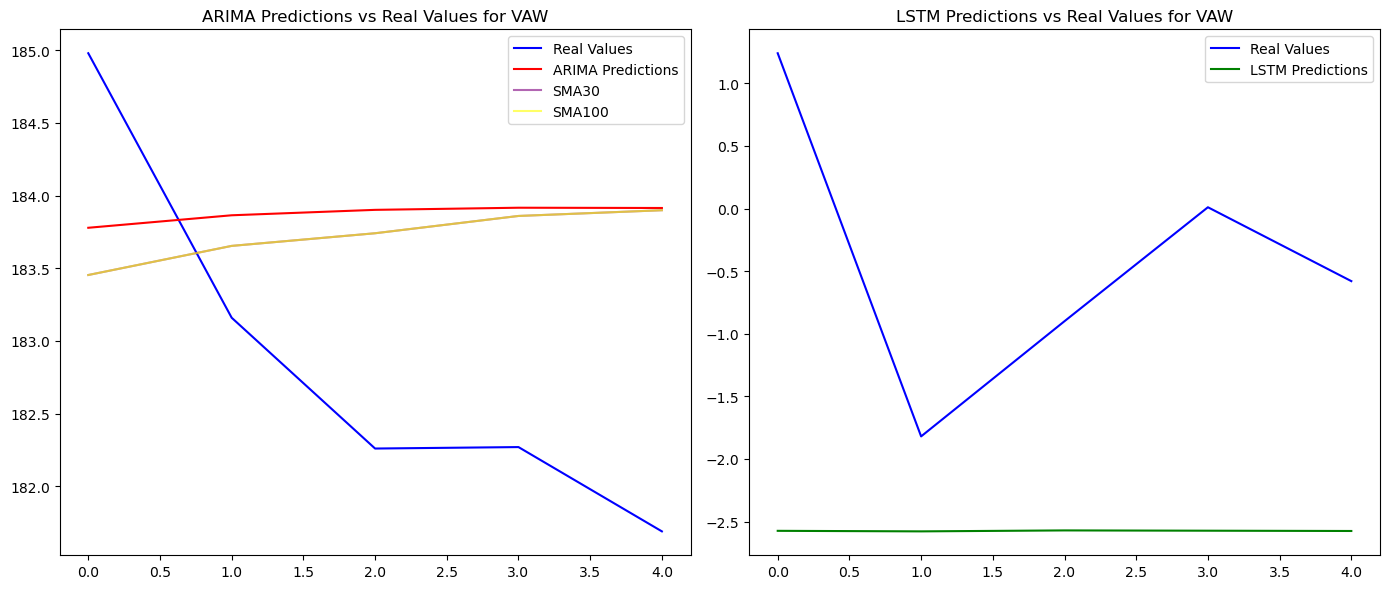
3 183.916991

4 183.915188

Name: predicted\_mean, dtype: float64

Real Values: [184.97999573 183.16000366 182.25999451 182.27000427 181.69000244]

LSTM Model MSE: 0.012144481168650233



Processing data for symbol: VNQ

LSTM Model MSE: 0.002853327464319741

Converted Predicted LSTM Values (Original Scale): [[-0.00997348]

[-0.01366247]

[-0.01028156]

[-0.01143517]

[-0.01076289]]

Results for symbol: VNQ

ARIMA Model MSE: 0.29335055592866843

Predicted ARIMA Values: 0 83.062302

1 83.053926

2 83.097545

3 83.131787

4 83.126202

Name: predicted\_mean, dtype: float64

Real Values: [84.06999969 83.52999878 83.51999664 83.19999695 83.33000183]

LSTM Model MSE: 0.002853327464319741

