STOCK PRICE PREDICTION

DATA LOADING AND PREPROCESSING

INTRODUCTION:

-In this section, we will embark on the initial steps of our stock price predictionproject by collecting and preprocessing historical stock market data.

-Accurate and well-organized data is the foundation of any successful predictive modeling project. We will utilize Python and relevant libraries to accomplish this task.

DATA COLLECTION:

DATA SOURCE:

-To begin our project, we need historical stock market data. You can obtain such data from various sources, including financial data providers, APIs, or public datasets.

-For the purpose of this project, we assume that we have already obtained a dataset in a suitable format.

- This dataset typically contains information about a stock's daily performance, including open price, close price, high price, low price, and trading volume.

LOADING THE DATA:

We will employ the Pandas library to load our dataset. Pandas is a powerful data manipulation library in Python that provides an efficient and flexible way to work with structured data.

INPUT:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import
train_test_split
from sklearn.preprocessing import
MinMaxScaler


# Assuming you have a CSV file with
historical stock price data
from google.colab import files
upload=files.upload()
import matplotlib.pyplot as plt
data = pd.read_csv('stock_data.csv')
```

OUTPUT:

Once we've loaded our data, we can inspect its structure and quality.

DATA PROCESSOING:

DATA EXPLORATION:

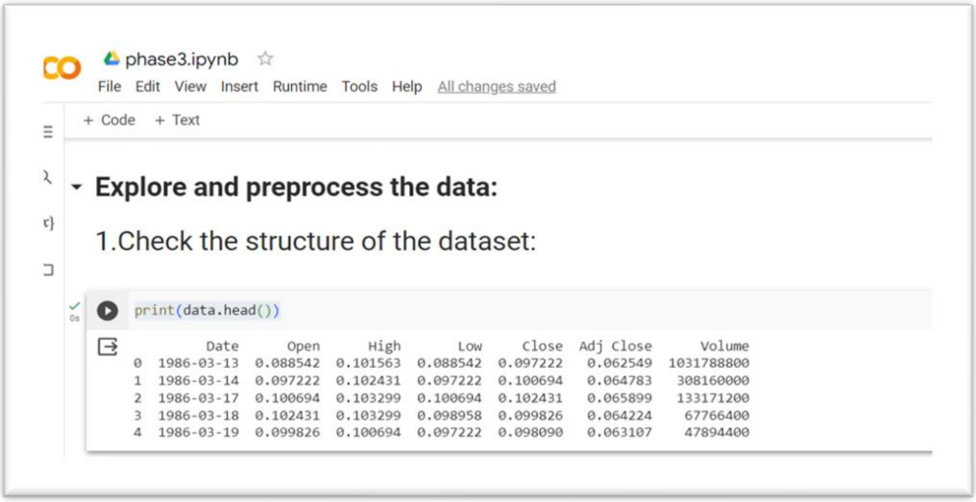
Before we dive into preprocessing, it's crucial to explore the dataset. Some key steps in this phase include:

1.Check the structure of the dataset:

INPUT:

```
print(data.head())
```

OUPUT:



2.Handle missing data, if any:

INPUT:

```
a=data.dropna(inplace=True)   # Remove rows with
missing values
print(a)
```

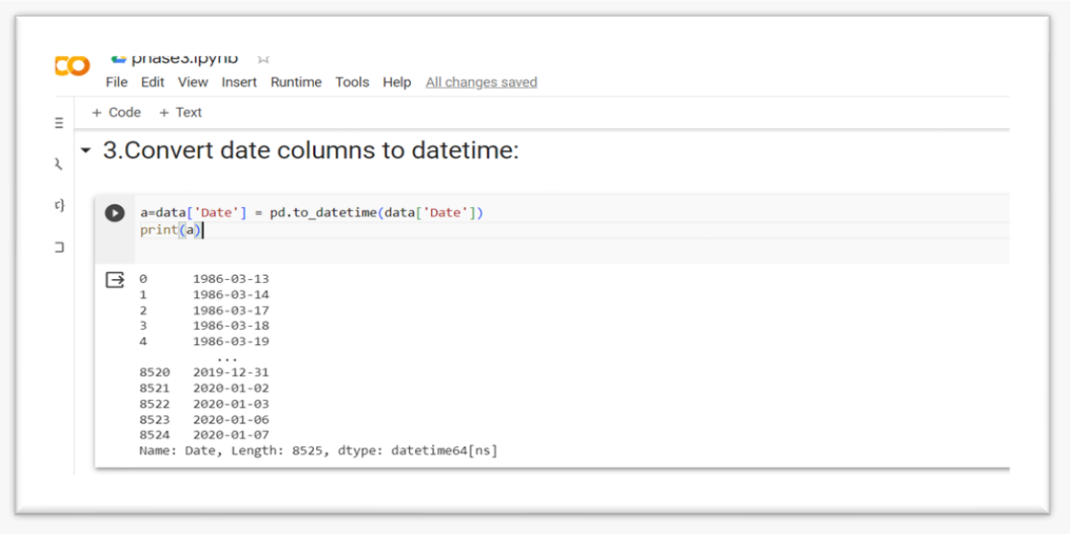
OUTPUT:

3.Convert date columns to datetime:

INPUT:

```
a=data['Date'] = pd.to_datetime(data['Date'])
print(a)
```

OUPUT:



4.Sort the data by date:

INPUT:

```
a=data.sort_values(by='Date',
inplace=True)
print(a)
```

OUPUT:


4.Sort the data by date:

```
[22] a=data.sort_values(by='Date', inplace=True)
     print(a)

None
```

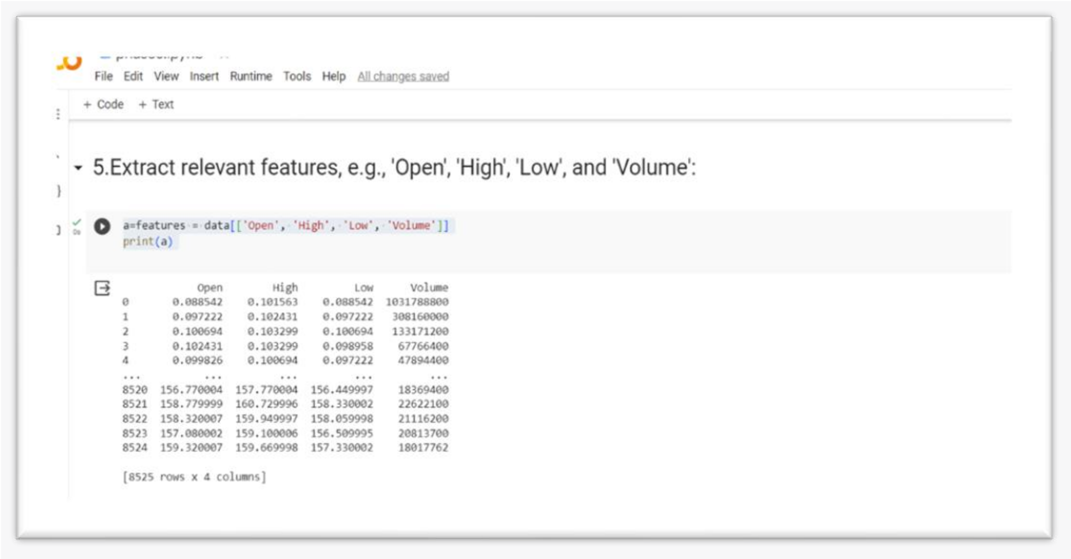
5.Extract relevant **features**, e.g., 'Open', 'High', 'Low', and 'Volume':

-In our stock price prediction model, we need to decide which features to include. Common features for stock prediction models include open price, high price, low price, and trading volume. We extract these features from the dataset:

INPUT:

```
 a=features = data[['Open', 'High', 'Low',
'Volume']]
print(a)
```

OUPUT:



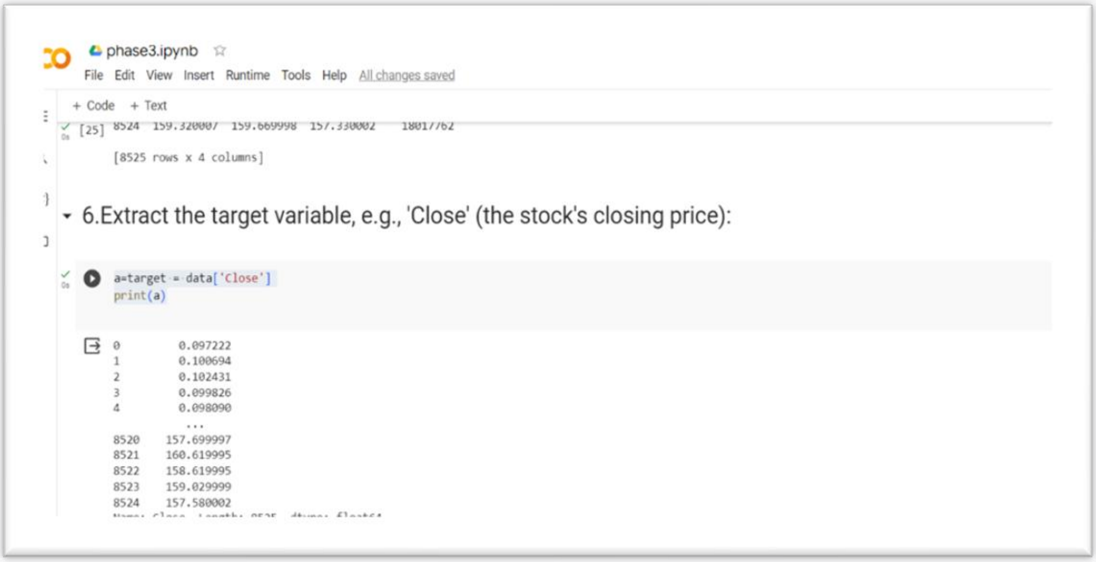
6.Extract the **target variable**, e.g., 'Close' (the stock's closing price):

The target variable, which we aim to predict, is typically the closing price of the stock. We extract this target variable from the dataset:

INPUT:

```
a=target = data['Close']
print(a)
```

OUPUT:



7.Normalize the features using Min-Max scaling:

INPUT:

```
scaler = MinMaxScaler()
a=features = scaler.fit_transform(features)
print(a)
```

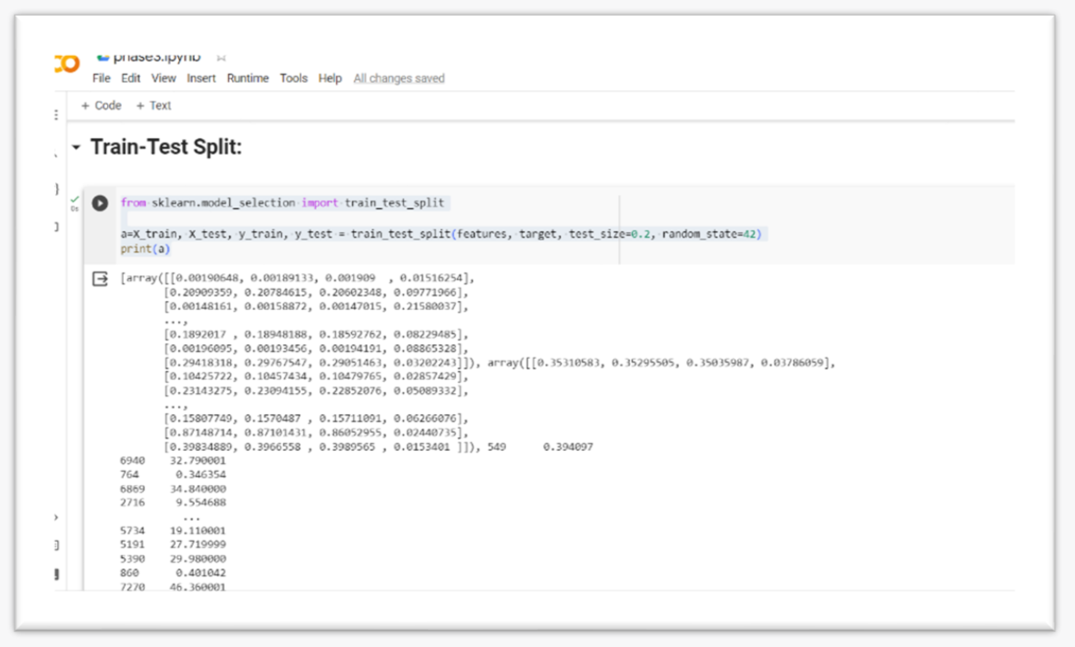
OUPUT:

Train-Test Split:

INPUT:

```
from sklearn.model_selection import
train_test_split

a=X_train, X_test, y_train, y_test =
train_test_split(features, target, test_size=0.2,
random_state=42)
print(a)
```

OUPUT:

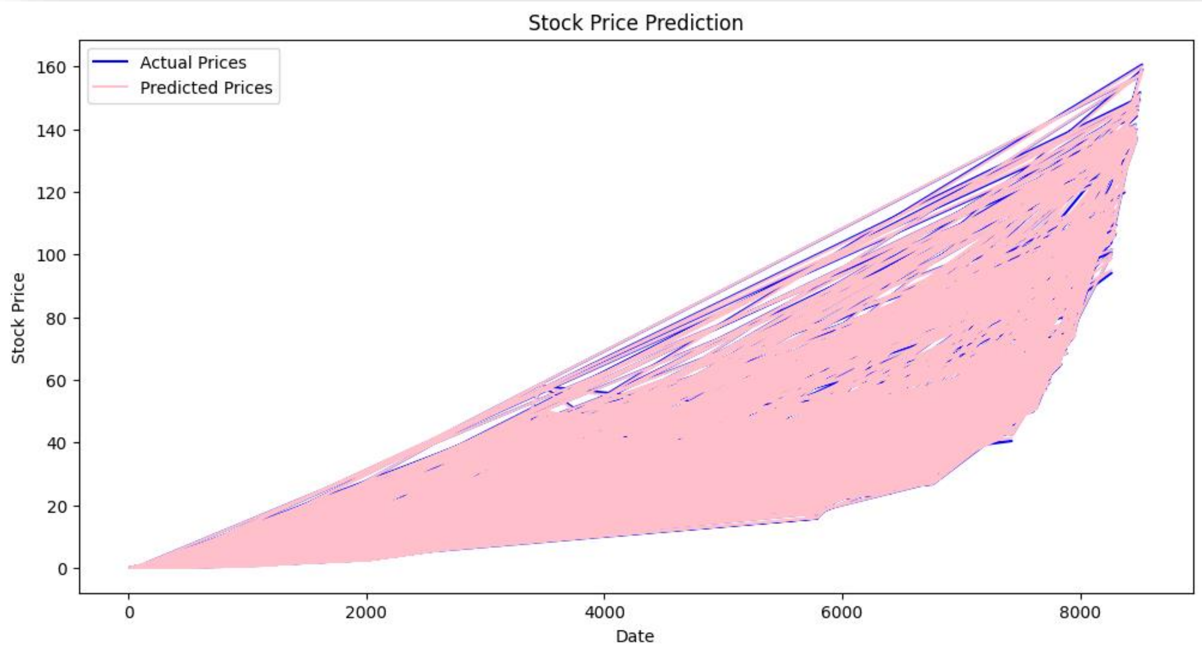


LINEAR REGRESSION MODEL AND VISUALIZING THE RESULTS:

INPUT:

```
# Import the linear regression model
from sklearn.linear_model import
LinearRegression
from sklearn.metrics import mean_squared_error

# Initialize and train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Calculate the Mean Squared Error (MSE) to
evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")

# Visualize the predicted vs. actual stock
prices
plt.figure(figsize=(12, 6))
plt.plot(y_test.index, y_test.values,
label='Actual Prices', color='blue')
plt.plot(y_test.index, y_pred, label='Predicted
Prices', color='pink')
plt.title('Stock Price Prediction')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```

OUTPUT:



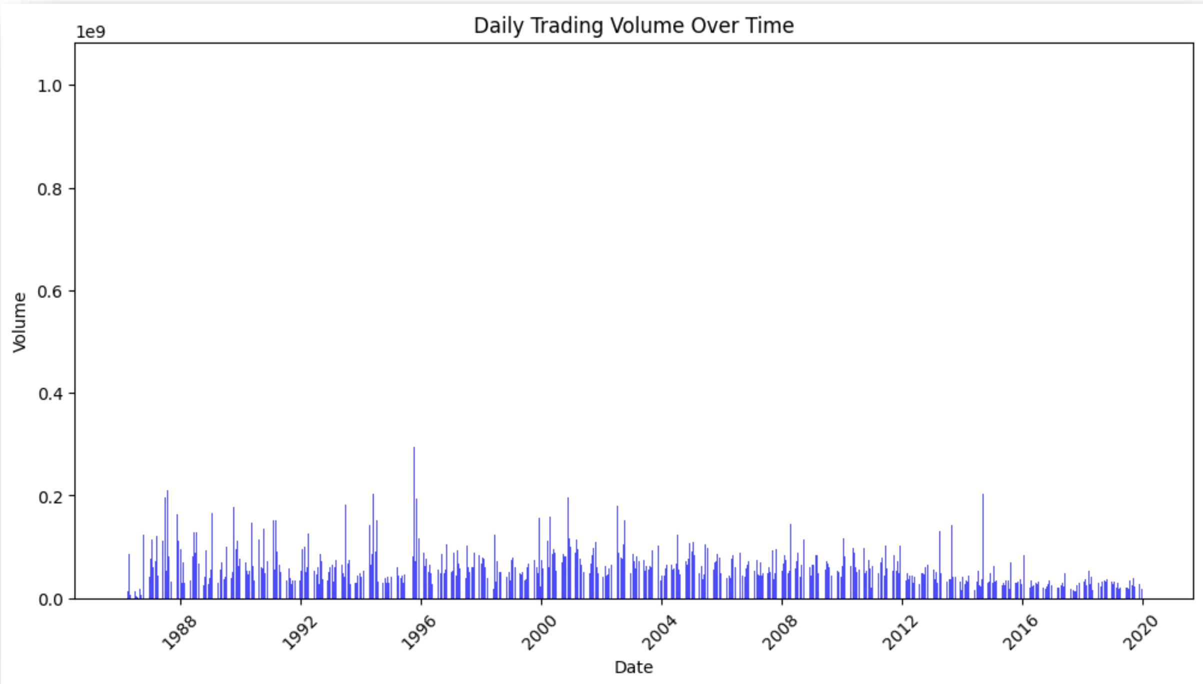
CREATE A BAR CHART TO VISUALIZE THE DAILY TRADING VOLIUME OVER TIME:

INPUT:

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset (assuming you've already loaded
it)
# Example data:
# data = pd.read_csv('stock_data.csv')

# Assuming 'Date' and 'Volume' columns exist in
your dataset
dates = data['Date']
volume = data['Volume']
```

```python
# Create a bar chart to visualize daily trading volume
plt.figure(figsize=(12, 6))
plt.bar(dates, volume, color='blue', alpha=0.7)
plt.xlabel('Date')
plt.ylabel('Volume')
plt.title('Daily Trading Volume Over Time')
plt.xticks(rotation=45)  # Rotate x-axis labels for readability

plt.show()
```

OUPUT:

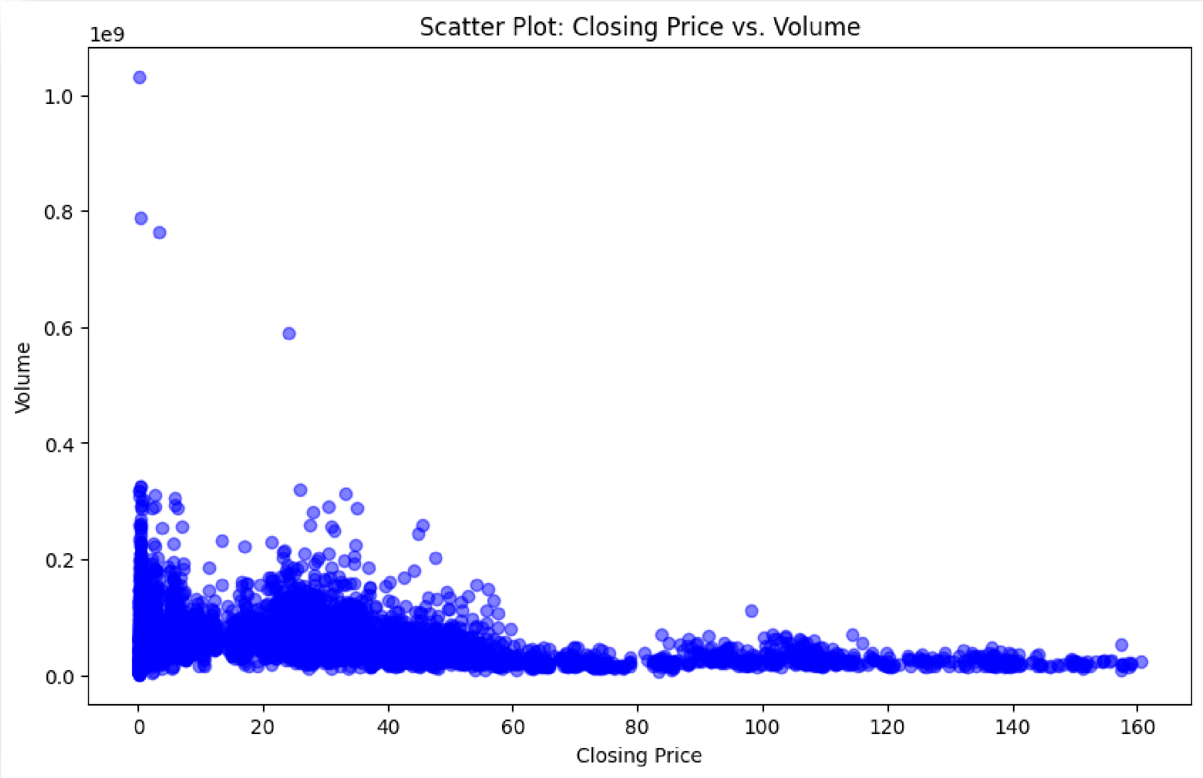


CREATE A SCATTER PLOT TO VISUALIZE THE RELATIONSHIP BETWEEN THE STOCK'S CLOSING PRICE AND ITS TRADING VOLUME OVER TIME:

INPUT:

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset (assuming you've already
loaded it)
# Example data:
# data = pd.read_csv('stock_data.csv')

# Assuming 'Close' and 'Volume' columns exist in
your dataset
close_price = data['Close']
volume = data['Volume']

# Create a scatter plot to visualize the
relationship between closing price and volume
plt.figure(figsize=(10, 6))
plt.scatter(close_price, volume, alpha=0.5,
color='blue')
plt.xlabel('Closing Price')
plt.ylabel('Volume')
plt.title('Scatter Plot: Closing Price vs.
Volume')

plt.show()
```

OUTPUT:



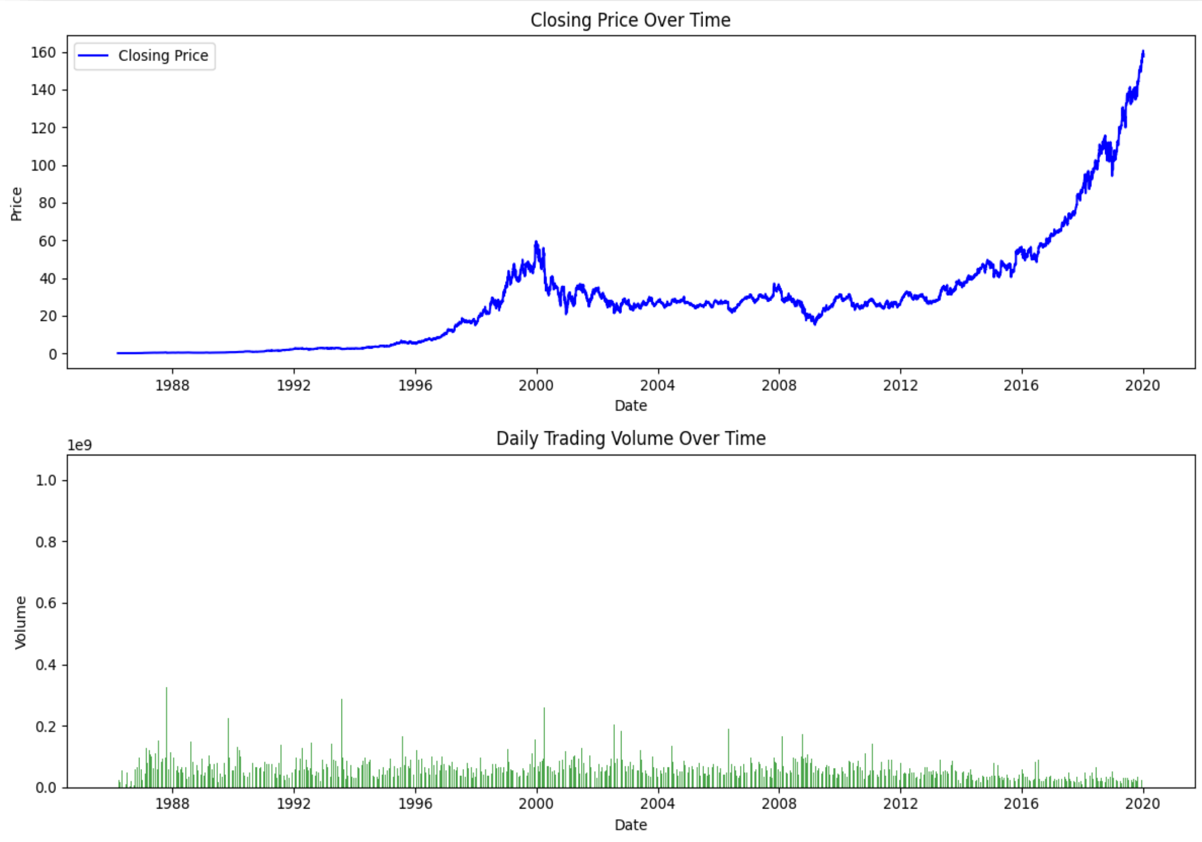
RESULT VISUALIZATION:

INPUT:

```
import pandas as pd
import matplotlib.pyplot as plt

# Load your stock price data into a
DataFrame (data assumed to be loaded)

# Convert the 'Date' column to a datetime
object
data['Date'] = pd.to_datetime(data['Date'])

# Set the 'Date' column as the index
data.set_index('Date', inplace=True)
```

```
# Create subplots for multiple
visualizations
fig, axes = plt.subplots(nrows=2, ncols=1,
figsize=(12, 8))

# Plot 1: Closing price over time
axes[0].plot(data.index, data['Close'],
label='Closing Price', color='blue')
axes[0].set_title('Closing Price Over Time')
axes[0].set_xlabel('Date')
axes[0].set_ylabel('Price')
axes[0].legend()

# Plot 2: Daily trading volume over time
axes[1].bar(data.index, data['Volume'],
color='green', alpha=0.7)
axes[1].set_title('Daily Trading Volume Over
Time')
axes[1].set_xlabel('Date')
axes[1].set_ylabel('Volume')

# Ensure the plots don't overlap
plt.tight_layout()

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

OUTPUT:



CONCLUSION:

-In this section, we have successfully loaded and preprocessed the historical stock market data, making it ready for model development. In the next steps, you can proceed with building and evaluating your stock price prediction model using various machine learning or time-series analysis techniques.

-These initial data preprocessing steps are crucial for ensuring the quality and suitability of the dataset for your prediction model. By following these steps, you have set a solid foundation for your stock price prediction project.