Importing required Libraries and Dataset

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import warnings
warnings.filterwarnings("ignore")

df = pd.read_csv('Oracle Dataset.csv')

df['Date'] = pd.to_datetime(df['Date'])

df.set_index('Date', inplace=True)
```

Data Preprocessing

In [2]:
```
print("Missing values in the dataset:", df.isnull().sum())
```

```
Missing values in the dataset: Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

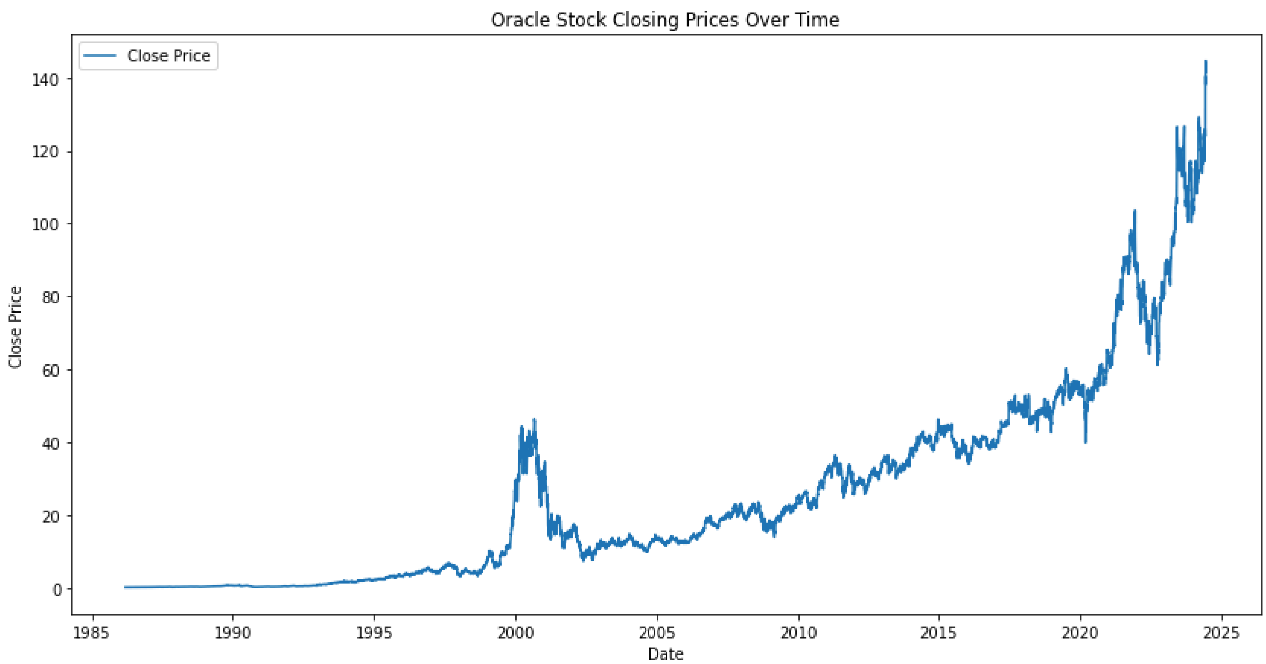
Descriptive Statistics

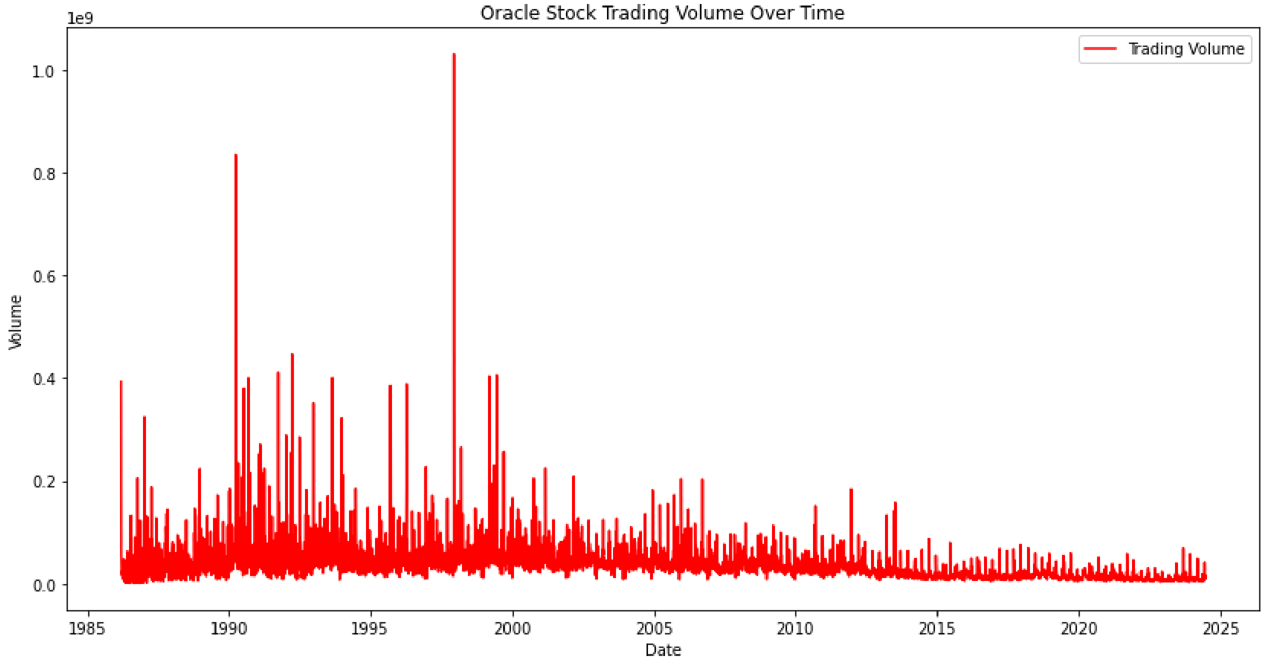
In [3]:
```
print("Descriptive statistics of the dataset:", df.describe())
```

```
Descriptive statistics of the dataset:              Open          High           Low  
Close    Adj Close  \
count  9647.000000  9647.000000  9647.000000  9647.000000  9647.000000
mean     25.276109    25.581687    24.986783    25.285539    22.724261
std      27.240417    27.511509    26.993151    27.256435    26.436486
min       0.041667     0.043981     0.040123     0.041667     0.033906
25%       3.064815     3.129630     2.972222     3.062500     2.492072
50%      16.520000    16.940001    16.150000    16.530001    13.451090
75%      39.311250    39.740002    38.965000    39.290001    34.002430
max     145.320007   145.320007   141.949997   144.639999   144.639999

             Volume
count  9.647000e+03
mean   3.562601e+07
std    3.292784e+07
min    3.888000e+05
25%    1.551595e+07
50%    2.984175e+07
75%    4.553488e+07
max    1.030963e+09
```

Visualizations

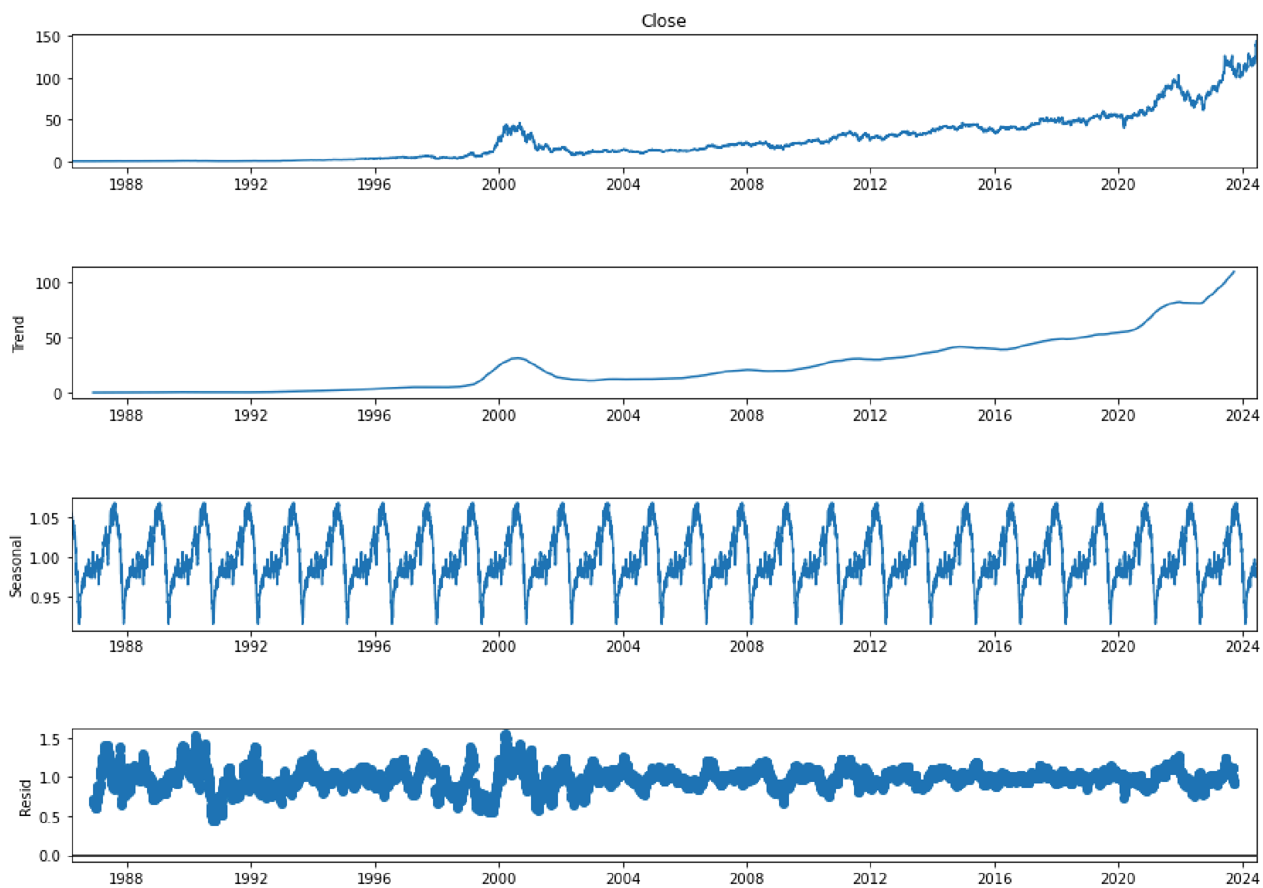
In [15]:
```
plt.figure(figsize=(14, 7))
plt.plot(df['Close'], label='Close Price')
plt.title('Oracle Stock Closing Prices Over Time')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.legend()
plt.show()

plt.figure(figsize=(14, 7))
plt.plot(df['Volume'], label='Trading Volume', color='red')
plt.title('Oracle Stock Trading Volume Over Time')
plt.xlabel('Date')
plt.ylabel('Volume')
plt.legend()
plt.show()
```





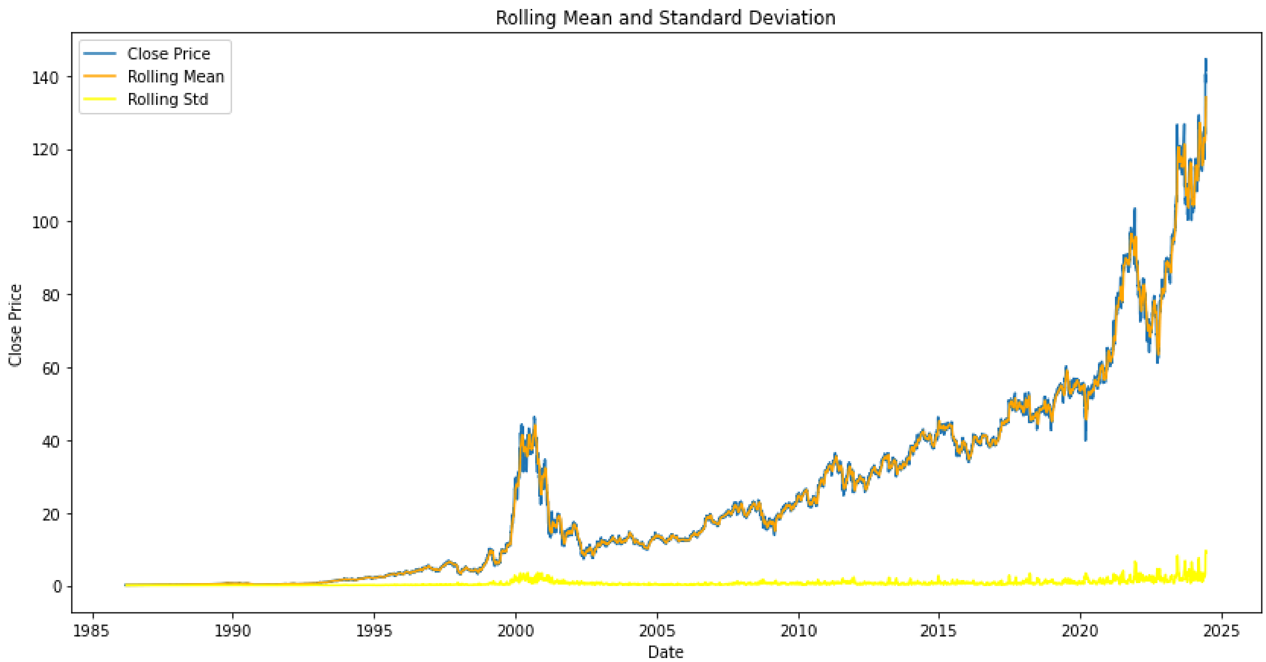
Time Series Decomposition

In [5]:
```
decomposition = seasonal_decompose(df['Close'], model='multiplicative', period=365)
fig = decomposition.plot()
fig.set_size_inches(14, 10)
plt.show()
```



Rolling Statistics

In [18]:
```
rolling_mean = df['Close'].rolling(window=12).mean()
rolling_std = df['Close'].rolling(window=12).std()

plt.figure(figsize=(14, 7))
plt.plot(df['Close'], label='Close Price')
plt.plot(rolling_mean, label='Rolling Mean', color='orange')
plt.plot(rolling_std, label='Rolling Std', color='yellow')
plt.title('Rolling Mean and Standard Deviation')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```



Stationarity Test

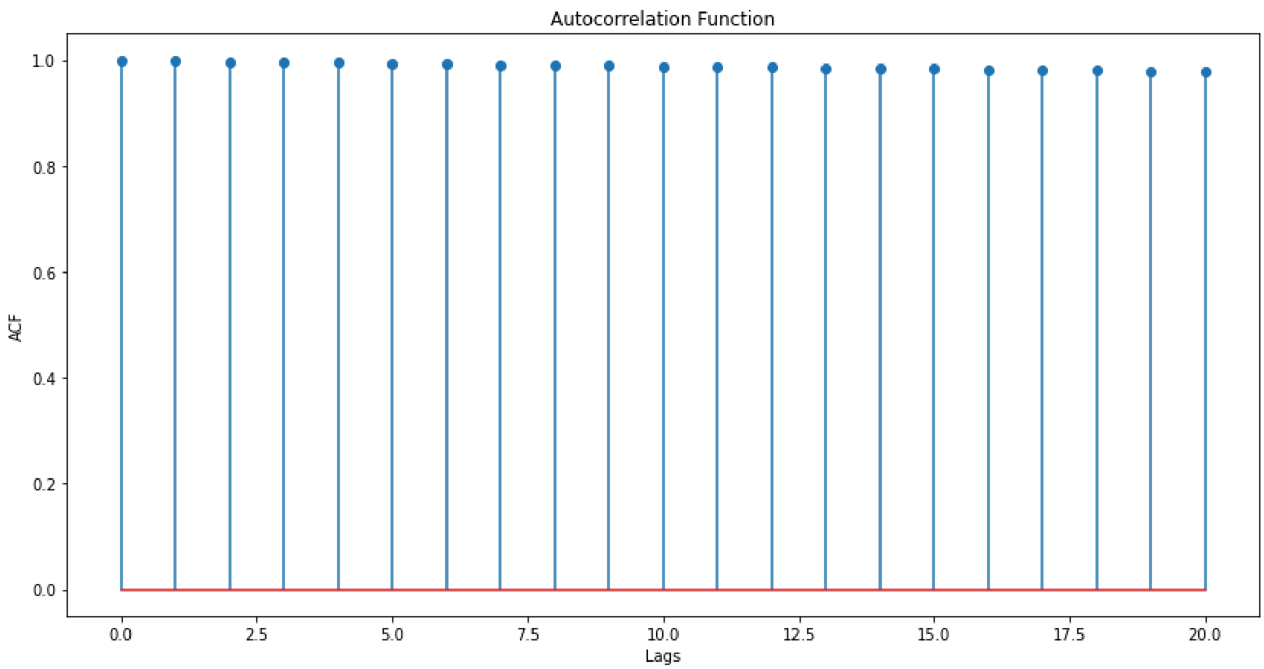
In [7]:
```
from statsmodels.tsa.stattools import adfuller

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

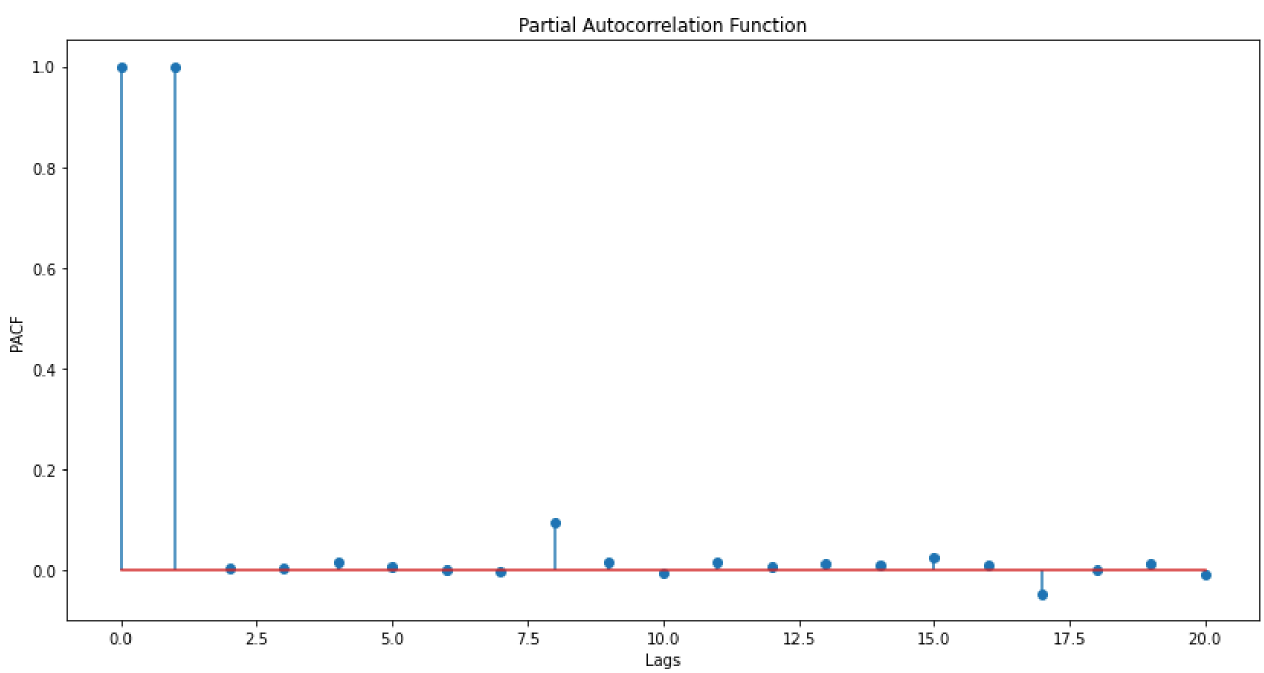
```
ADF Statistic: 2.8385579860917933
p-value: 1.0
Critical Values: {'1%': -3.4310300838145107, '5%': -2.8618405552764714, '10%': -2.566929
9804591974}
```

Autocorrelation and Partial Autocorrelation

In [8]:
```
from statsmodels.tsa.stattools import acf
from statsmodels.tsa.stattools import pacf
```

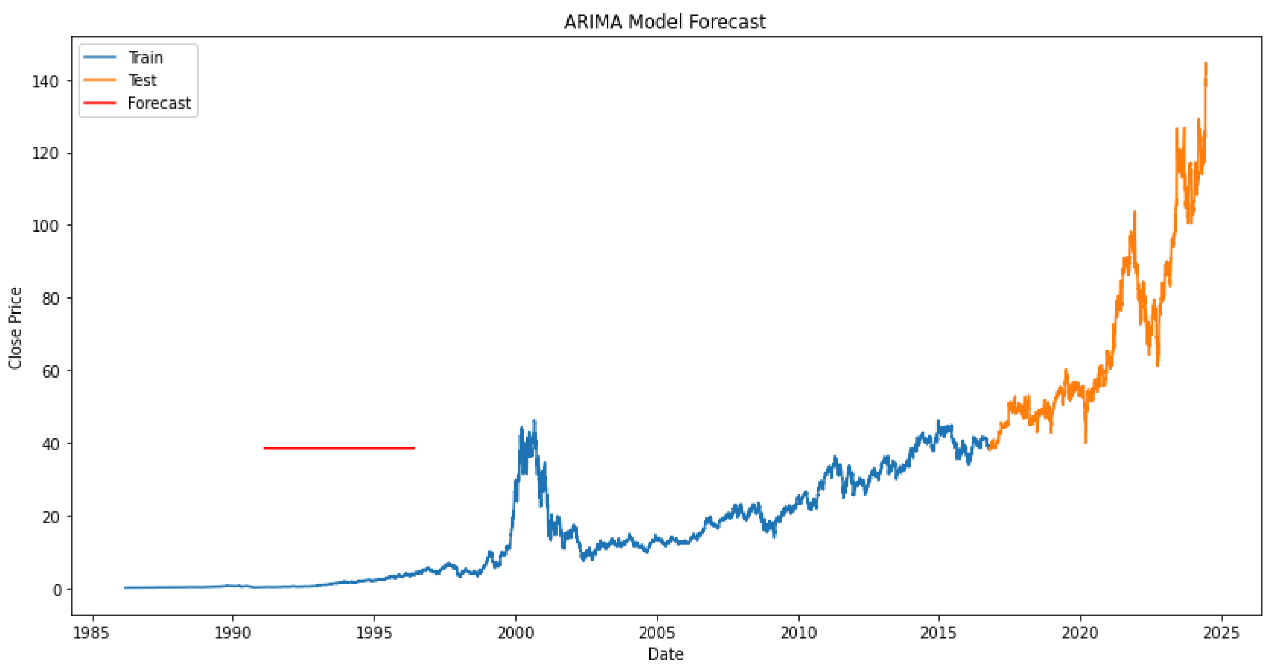
```
lag_acf = acf(df['Close'], nlags=20)
lag_pacf = pacf(df['Close'], nlags=20)

plt.figure(figsize=(14, 7))
plt.stem(lag_acf)
plt.title('Autocorrelation Function')
plt.xlabel('Lags')
plt.ylabel('ACF')
plt.show()

plt.figure(figsize=(14, 7))
plt.stem(lag_pacf)
plt.title('Partial Autocorrelation Function')
plt.xlabel('Lags')
plt.ylabel('PACF')
plt.show()
```





ARIMA Modeling

In [10]:
```
from statsmodels.tsa.arima.model import ARIMA

train_size = int(len(df) * 0.8)
train, test = df['Close'][:train_size], df['Close'][train_size:]

model = ARIMA(train, order=(5, 1, 0))
model_fit = model.fit()

forecast = model_fit.forecast(steps=len(test))
plt.figure(figsize=(14, 7))
plt.plot(train, label='Train')
plt.plot(test, label='Test')
plt.plot(forecast, label='Forecast', color='red')
plt.title('ARIMA Model Forecast')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```



GARCH Modeling

In [11]:
```
%pip install arch

from arch import arch_model

returns = df['Close'].pct_change().dropna()
model_garch = arch_model(returns, vol='Garch', p=1, q=1)
model_garch_fit = model_garch.fit()

forecast_garch = model_garch_fit.forecast(horizon=5)
```

```
Collecting arch
  Downloading arch-5.6.0-cp38-cp38-win_amd64.whl (857 kB)
```

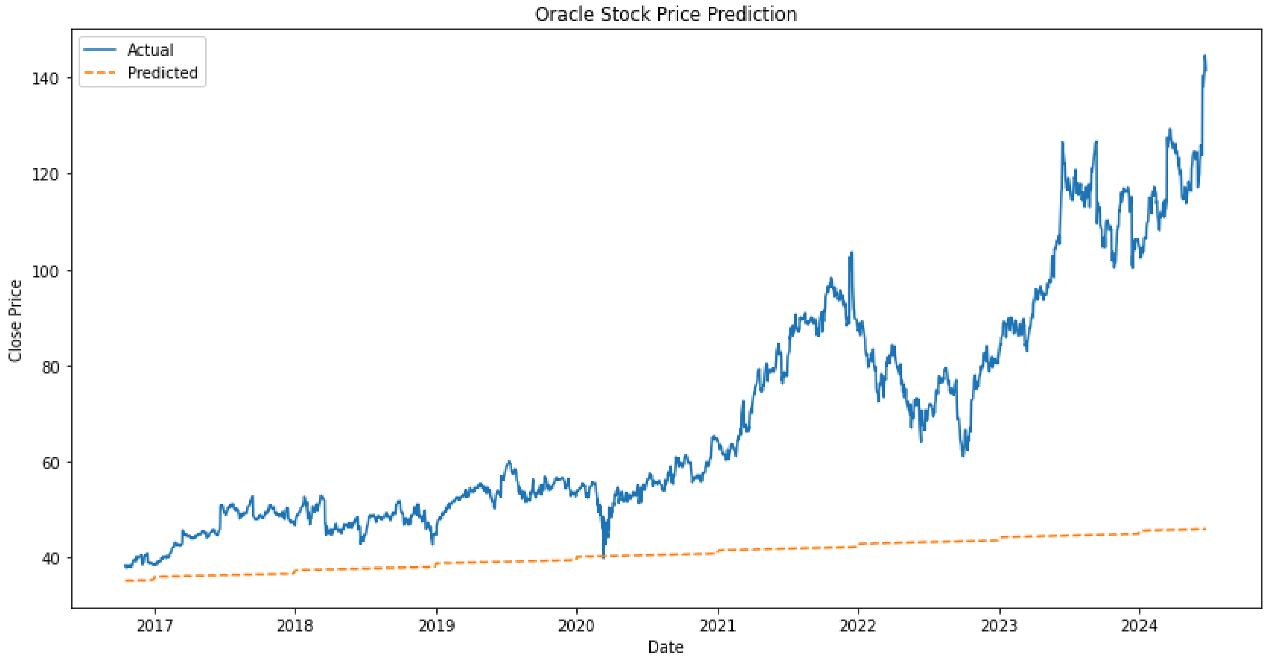
```
Requirement already satisfied: statsmodels>=0.11 in c:\users\abdul\anaconda3\lib\site-pa
ckages (from arch) (0.12.0)
Requirement already satisfied: scipy>=1.3 in c:\users\abdul\anaconda3\lib\site-packages
(from arch) (1.5.2)
Requirement already satisfied: pandas>=1.0 in c:\users\abdul\anaconda3\lib\site-packages
(from arch) (1.1.3)
Requirement already satisfied: numpy>=1.17 in c:\users\abdul\anaconda3\lib\site-packages
(from arch) (1.19.2)
Collecting property-cached>=1.6.4
  Downloading property_cached-1.6.4-py2.py3-none-any.whl (7.8 kB)
Requirement already satisfied: patsy>=0.5 in c:\users\abdul\anaconda3\lib\site-packages
(from statsmodels>=0.11->arch) (0.5.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\abdul\anaconda3\lib\si
te-packages (from pandas>=1.0->arch) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in c:\users\abdul\anaconda3\lib\site-package
s (from pandas>=1.0->arch) (2020.1)
Requirement already satisfied: six in c:\users\abdul\anaconda3\lib\site-packages (from p
atsy>=0.5->statsmodels>=0.11->arch) (1.15.0)
Installing collected packages: property-cached, arch
Successfully installed arch-5.6.0 property-cached-1.6.4
Note: you may need to restart the kernel to use updated packages.
Iteration:      1,   Func. Count:      6,   Neg. LLF: 3642376969870.3047
Iteration:      2,   Func. Count:     18,   Neg. LLF: -22205.220237460904
Optimization terminated successfully    (Exit mode 0)
            Current function value: -22205.220224298704
            Iterations: 6
            Function evaluations: 18
            Gradient evaluations: 2
```

Predictive Modeling and Feature Engineering

In [12]:

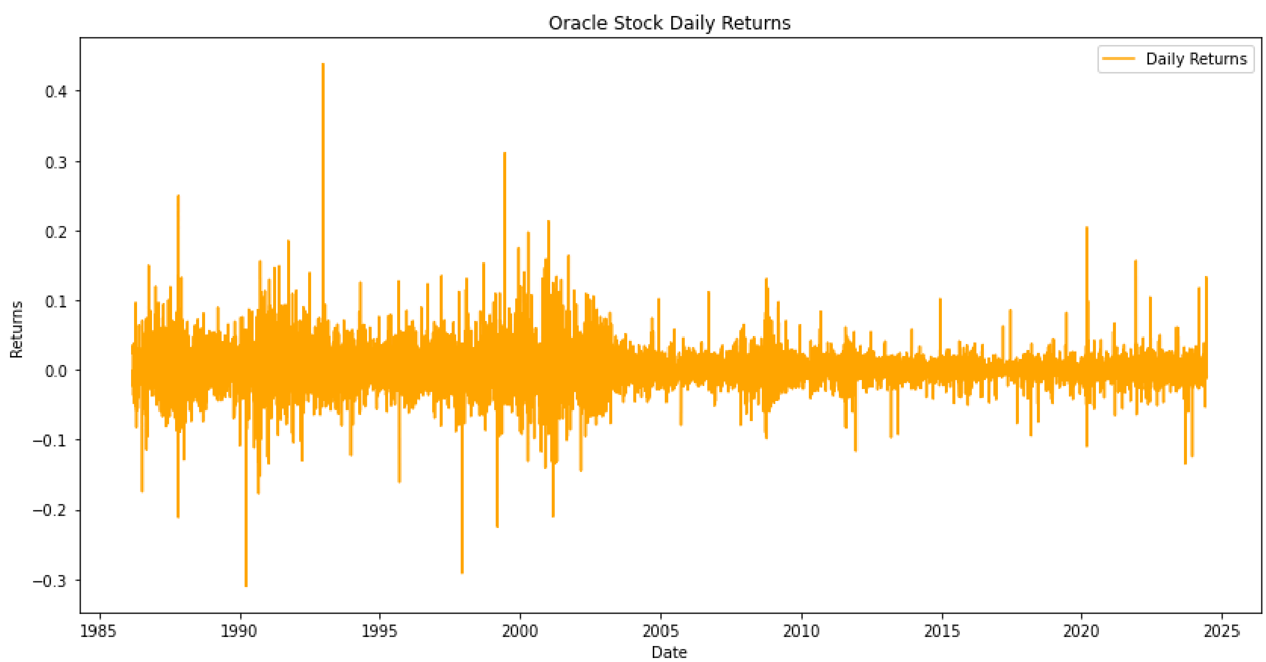
```
df['Year'] = df.index.year
df['Month'] = df.index.month
df['Day'] = df.index.day

X = df[['Year', 'Month', 'Day']]
y = df['Close']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)

model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R^2 Score: {r2}')

plt.figure(figsize=(14, 7))
plt.plot(y_test.index, y_test, label='Actual')
plt.plot(y_test.index, y_pred, label='Predicted', linestyle='--')
plt.title('Oracle Stock Price Prediction')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```

```
Mean Squared Error: 1260.1612472827792
R^2 Score: -1.1420148618511128
```



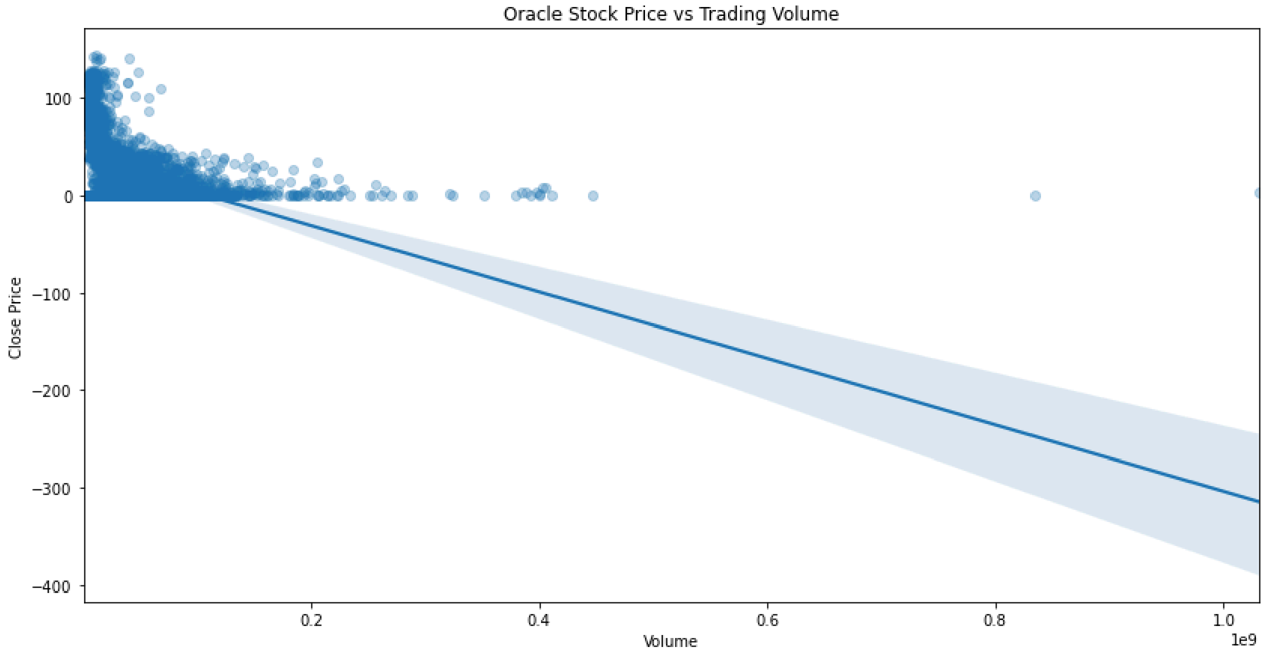
Volatility Analysis

In [19]:
```
df['Returns'] = df['Close'].pct_change()
plt.figure(figsize=(14, 7))
plt.plot(df['Returns'], label='Daily Returns', color='orange')
plt.title('Oracle Stock Daily Returns')
plt.xlabel('Date')
plt.ylabel('Returns')
plt.legend()
plt.show()
```



Volume Analysis

In [14]:
```
plt.figure(figsize=(14, 7))
sns.regplot(x='Volume', y='Close', data=df, scatter_kws={'alpha':0.3})
plt.title('Oracle Stock Price vs Trading Volume')
plt.xlabel('Volume')
plt.ylabel('Close Price')
plt.show()
```



In [ ]: