In [1]:
```
import numpy as np
import  pandas as pd
import matplotlib.pyplot as plt
from matplotlib.dates import DateFormatter
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:
```
def read_data(path, date="Date"):
    data = pd.read_csv(path)
    data[date] = pd.to_datetime(data[date], dayfirst=True)
    return data
```

In [3]:
```
#import csv data
nvda = read_data("data/NVDA.csv")
#nvda['Close'] = nvda['Close'].apply(np.log)
display(nvda)
```

| | Date | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|---|
| **0** | 2019-04-15 | 47.424999 | 47.615002 | 45.775002 | 46.575001 | 46.258690 | 156704800 |
| **1** | 2019-04-22 | 46.337502 | 48.202499 | 43.325001 | 44.522499 | 44.220123 | 231328000 |
| **2** | 2019-04-29 | 44.602501 | 46.217499 | 43.875000 | 45.752499 | 45.441769 | 184062800 |
| **3** | 2019-05-06 | 43.875000 | 45.084999 | 41.000000 | 42.205002 | 41.918369 | 262631200 |
| **4** | 2019-05-13 | 40.875000 | 41.107498 | 38.735001 | 39.132500 | 38.866734 | 348726800 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **257** | 2024-03-18 | 903.880005 | 947.780029 | 850.099976 | 942.890015 | 942.890015 | 288579700 |
| **258** | 2024-03-25 | 939.409973 | 967.659973 | 891.229980 | 903.559998 | 903.559998 | 208706300 |
| **259** | 2024-04-01 | 902.989990 | 922.250000 | 858.799988 | 880.080017 | 880.080017 | 208939400 |
| **260** | 2024-04-08 | 887.000000 | 907.390015 | 830.219971 | 881.859985 | 881.859985 | 207522200 |
| **261** | 2024-04-15 | 890.979980 | 906.130005 | 859.289978 | 860.010010 | 860.010010 | 44307700 |

262 rows × 7 columns

In [4]:
```
searchvol = read_data("data/ai-timeline_Glimpse_Google-Trends.csv", date='T
lnsearchvol = read_data("data/ai-timeline_Glimpse_Google-Trends.csv", date=

lnsearchvol['Absolute Google Search Volume'] = lnsearchvol['Absolute Google


display(searchvol)
display(lnsearchvol)
```

| | Time (week of) | Normalized Value (0-100) | Absolute Google Search Volume |
|---|---|---|---|
| **0** | 2019-04-14 | 15 | 1360594 |
| **1** | 2019-04-21 | 14 | 1269888 |
| **2** | 2019-04-28 | 15 | 1360594 |
| **3** | 2019-05-05 | 15 | 1360594 |
| **4** | 2019-05-12 | 15 | 1360594 |
| **...** | ... | ... | ... |
| **257** | 2024-03-17 | 98 | 8889219 |
| **258** | 2024-03-24 | 100 | 9070631 |
| **259** | 2024-03-31 | 96 | 8707806 |
| **260** | 2024-04-07 | 97 | 8798512 |
| **261** | 2024-04-14 | 97 | 8798512 |

262 rows × 3 columns

| | Time (week of) | Normalized Value (0-100) | Absolute Google Search Volume |
|---|---|---|---|
| **0** | 2019-04-14 | 15 | 14.123432 |
| **1** | 2019-04-21 | 14 | 14.054439 |
| **2** | 2019-04-28 | 15 | 14.123432 |
| **3** | 2019-05-05 | 15 | 14.123432 |
| **4** | 2019-05-12 | 15 | 14.123432 |
| **...** | ... | ... | ... |
| **257** | 2024-03-17 | 98 | 16.000350 |
| **258** | 2024-03-24 | 100 | 16.020552 |
| **259** | 2024-03-31 | 96 | 15.979730 |
| **260** | 2024-04-07 | 97 | 15.990093 |
| **261** | 2024-04-14 | 97 | 15.990093 |

262 rows × 3 columns

In [5]:
```
#cleaning data
nvda = nvda.drop(columns=['Open','High','Low','Adj Close'])
display(nvda)
```

| | Date | Close | Volume |
|---|---|---|---|
| **0** | 2019-04-15 | 46.575001 | 156704800 |
| **1** | 2019-04-22 | 44.522499 | 231328000 |
| **2** | 2019-04-29 | 45.752499 | 184062800 |
| **3** | 2019-05-06 | 42.205002 | 262631200 |
| **4** | 2019-05-13 | 39.132500 | 348726800 |
| **...** | ... | ... | ... |
| **257** | 2024-03-18 | 942.890015 | 288579700 |
| **258** | 2024-03-25 | 903.559998 | 208706300 |
| **259** | 2024-04-01 | 880.080017 | 208939400 |
| **260** | 2024-04-08 | 881.859985 | 207522200 |
| **261** | 2024-04-15 | 860.010010 | 44307700 |

262 rows × 3 columns

In [6]:
```
searchvol = searchvol.rename(columns = {'Time (week of)': 'Date',"Absolute
lnsearchvol = lnsearchvol.rename(columns = {'Time (week of)': 'Date',"Absol
display(searchvol)
display(lnsearchvol)
```

| | Date | Normalized Value (0-100) | Search Volume |
|---|---|---|---|
| **0** | 2019-04-14 | 15 | 1360594 |
| **1** | 2019-04-21 | 14 | 1269888 |
| **2** | 2019-04-28 | 15 | 1360594 |
| **3** | 2019-05-05 | 15 | 1360594 |
| **4** | 2019-05-12 | 15 | 1360594 |
| **...** | ... | ... | ... |
| **257** | 2024-03-17 | 98 | 8889219 |
| **258** | 2024-03-24 | 100 | 9070631 |
| **259** | 2024-03-31 | 96 | 8707806 |
| **260** | 2024-04-07 | 97 | 8798512 |
| **261** | 2024-04-14 | 97 | 8798512 |

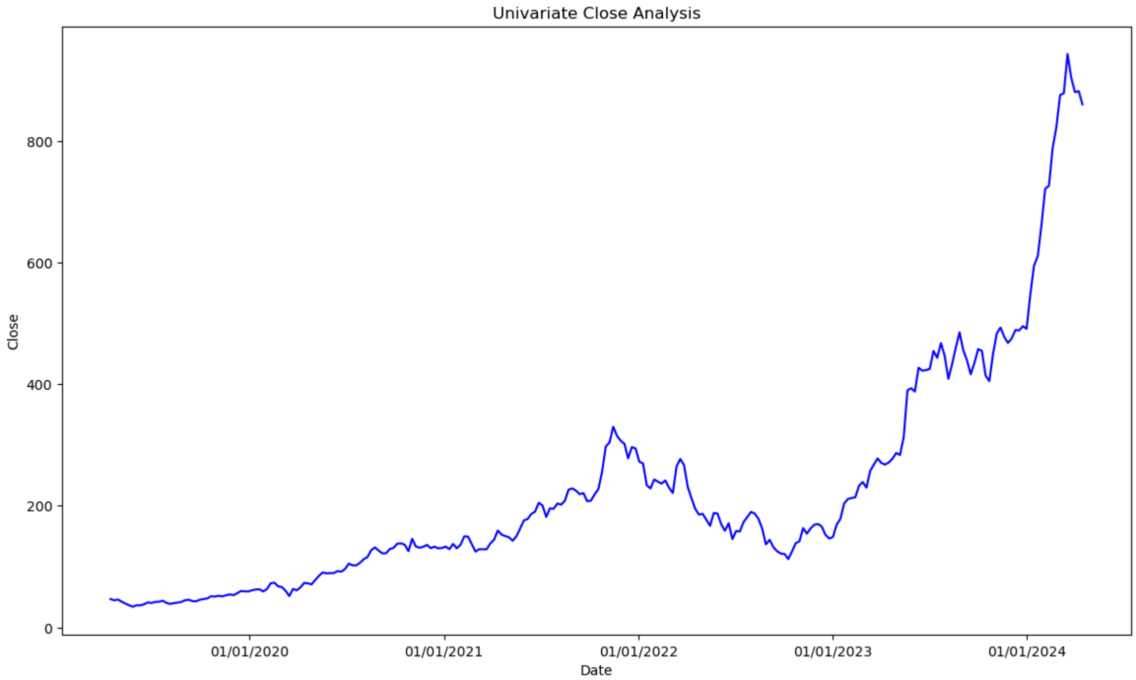
262 rows × 3 columns

| | Date | Normalized Value (0-100) | Search Volume |
|---|---|---|---|
| **0** | 2019-04-14 | 15 | 14.123432 |
| **1** | 2019-04-21 | 14 | 14.054439 |
| **2** | 2019-04-28 | 15 | 14.123432 |
| **3** | 2019-05-05 | 15 | 14.123432 |
| **4** | 2019-05-12 | 15 | 14.123432 |
| **...** | ... | ... | ... |
| **257** | 2024-03-17 | 98 | 16.000350 |
| **258** | 2024-03-24 | 100 | 16.020552 |
| **259** | 2024-03-31 | 96 | 15.979730 |
| **260** | 2024-04-07 | 97 | 15.990093 |
| **261** | 2024-04-14 | 97 | 15.990093 |

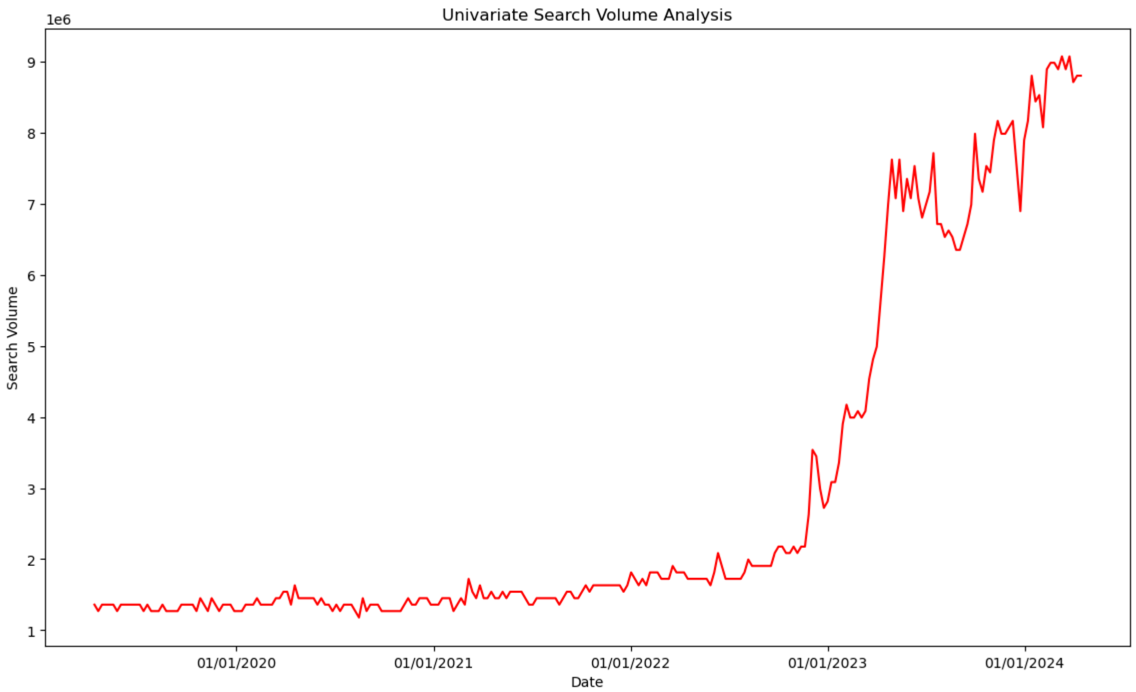
262 rows × 3 columns

In [7]:
```
def plot_uni(data,variable,colour):
  plt.figure(figsize=(14, 8))
  plt.xlabel('Date')
  plt.ylabel(variable)
  plt.title('Univariate '+variable+' Analysis')
    # Format x-axis dates
  date_form = DateFormatter("%d/%m/%Y")
  plt.gca().xaxis.set_major_formatter(date_form)
  plt.plot(data['Date'], data[variable], color=colour)
```

In [8]:
```
plot_uni(nvda,'Close','blue')
plt.savefig("outputs/figure1.png")
plt.show()
```



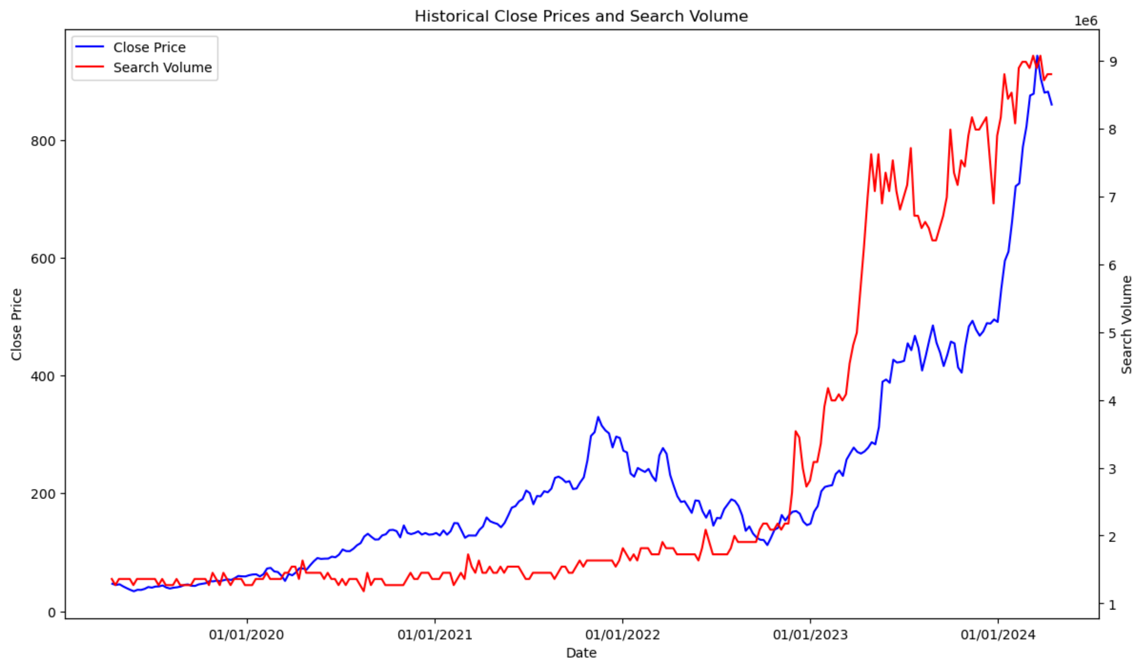

In [9]:
```
plot_uni(searchvol,'Search Volume','red')
plt.savefig("outputs/figure2")
plt.show()
```



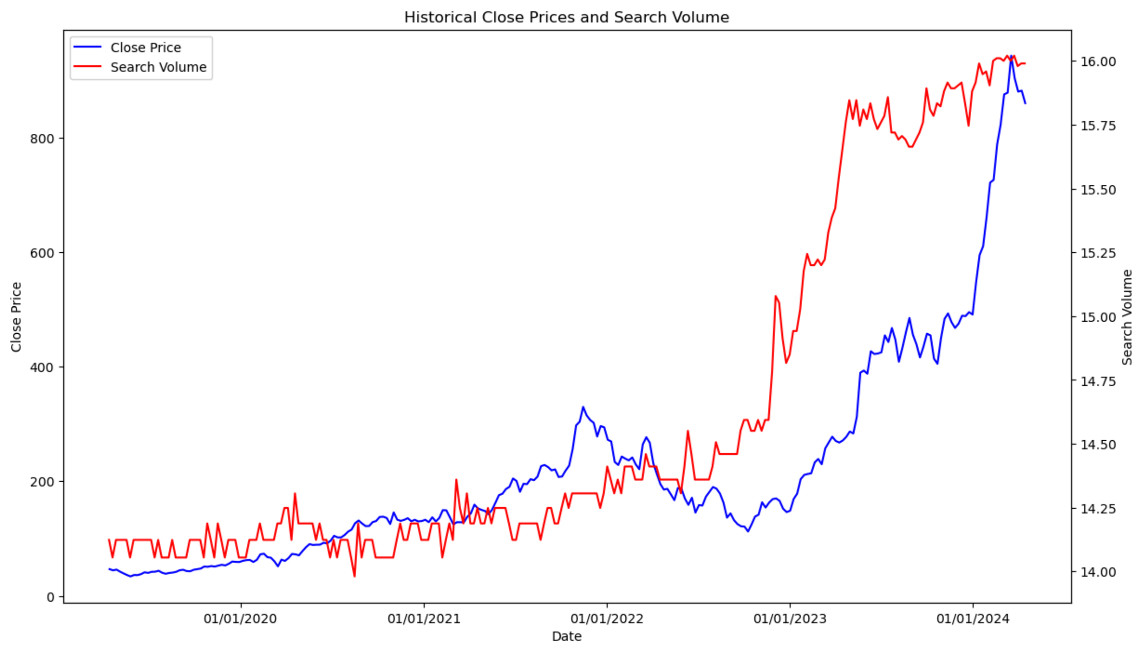
In [10]:
```
def scientific_formatter(x, pos):
    # Format the tick label in scientific notation with exponent next to th
    return "{:.0e}".format(x)
```

In [11]:
```
def plot_comparison(nvda, searchvol, important_dates=None):
    # Plotting historical close prices and search volume
    plt.figure(figsize=(14, 8))
    plt.xlabel('Date')
    plt.ylabel('Close Price')
    plt.title('Historical Close Prices and Search Volume')
    # Format x-axis dates
    date_form = DateFormatter("%d/%m/%Y")
    plt.gca().xaxis.set_major_formatter(date_form)

    # Plot Close Price on primary y-axis
    stock = plt.plot(nvda['Date'], nvda['Close'], label='Close Price', colo
    # Create secondary y-axis for Search Volume
    ax2 = plt.twinx()
    search = ax2.plot(searchvol['Date'], searchvol['Search Volume'], label=
    ax2.set_ylabel('Search Volume')

    # Combine legend for both primary and secondary plots
    lns = stock + search
    labs = [l.get_label() for l in lns]
    plt.legend(lns, labs, loc='upper left')

    if important_dates != None:
        for item in important_dates:
            plt.axvline(x=pd.Timestamp(item['date']), color='gray', linesty
            y = plt.ylim()[0] + (.5*plt.ylim()[1]-6.75) if item['top'] == T
            plt.text(pd.Timestamp(item['date']), y, item['name'], rotation=
```

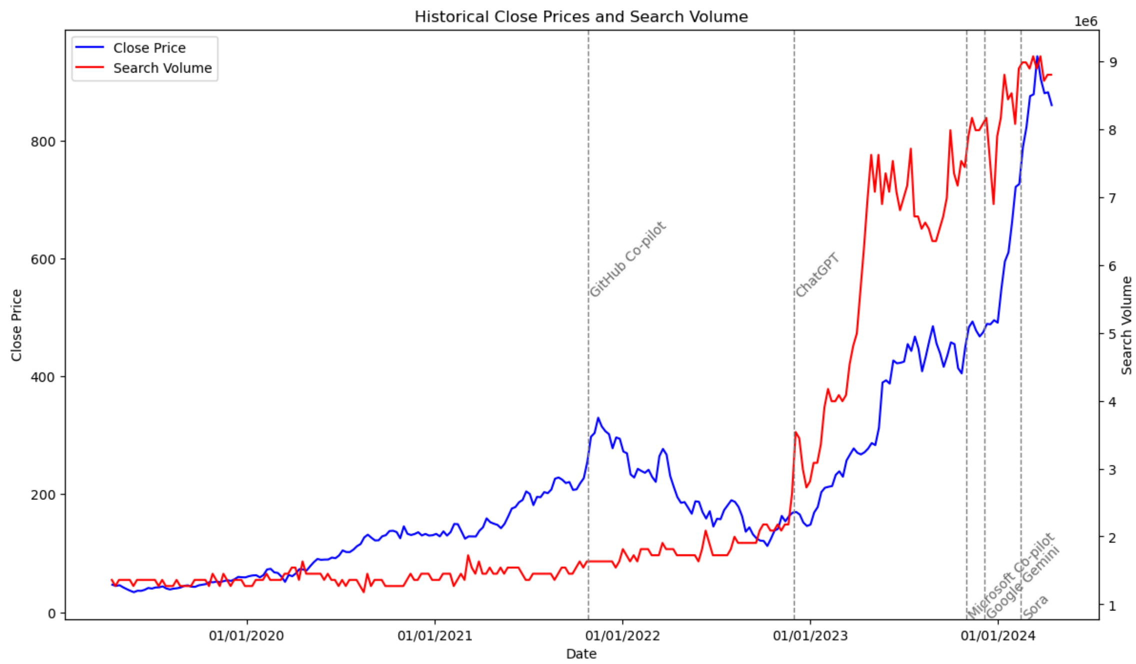
In [12]:
```
plot_comparison(nvda, searchvol)
plt.savefig("outputs/figure3")
plt.show()
```



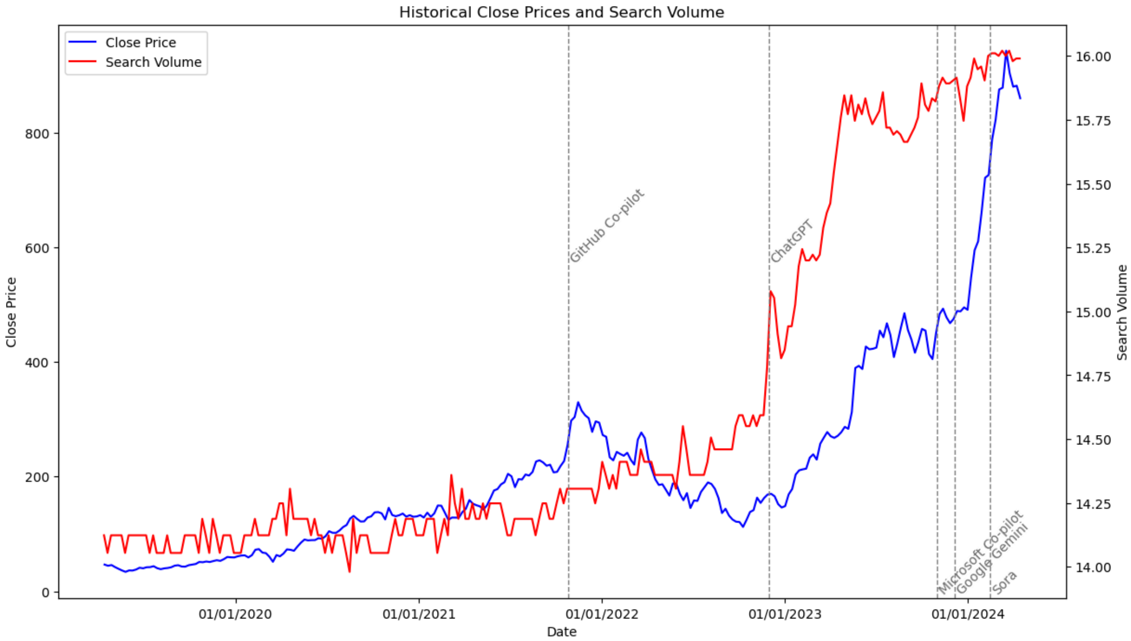
In [13]:
```
plot_comparison(nvda, lnsearchvol)
plt.show()
```




In [14]:
```
# Dates of major AI releases
important_dates = [
    {'date':'2024-02-15', 'name':'Sora', 'top':False},
    {'date':'2023-12-6', 'name':'Google Gemini', 'top':False},
    {'date':'2023-11-1', 'name':'Microsoft Co-pilot', 'top':False},
    {'date':'2022-11-30', 'name':'ChatGPT', 'top':True},
    {'date':'2021-10-27', 'name':'GitHub Co-pilot', 'top':True}
    ]  # Example dates and corresponding titles

plot_comparison(nvda, searchvol, important_dates)
plt.savefig("outputs/figure4")
plt.show()
```



In [15]:
```
plot_comparison(nvda, lnsearchvol, important_dates)
plt.savefig("outputs/figure7")
plt.show()
```



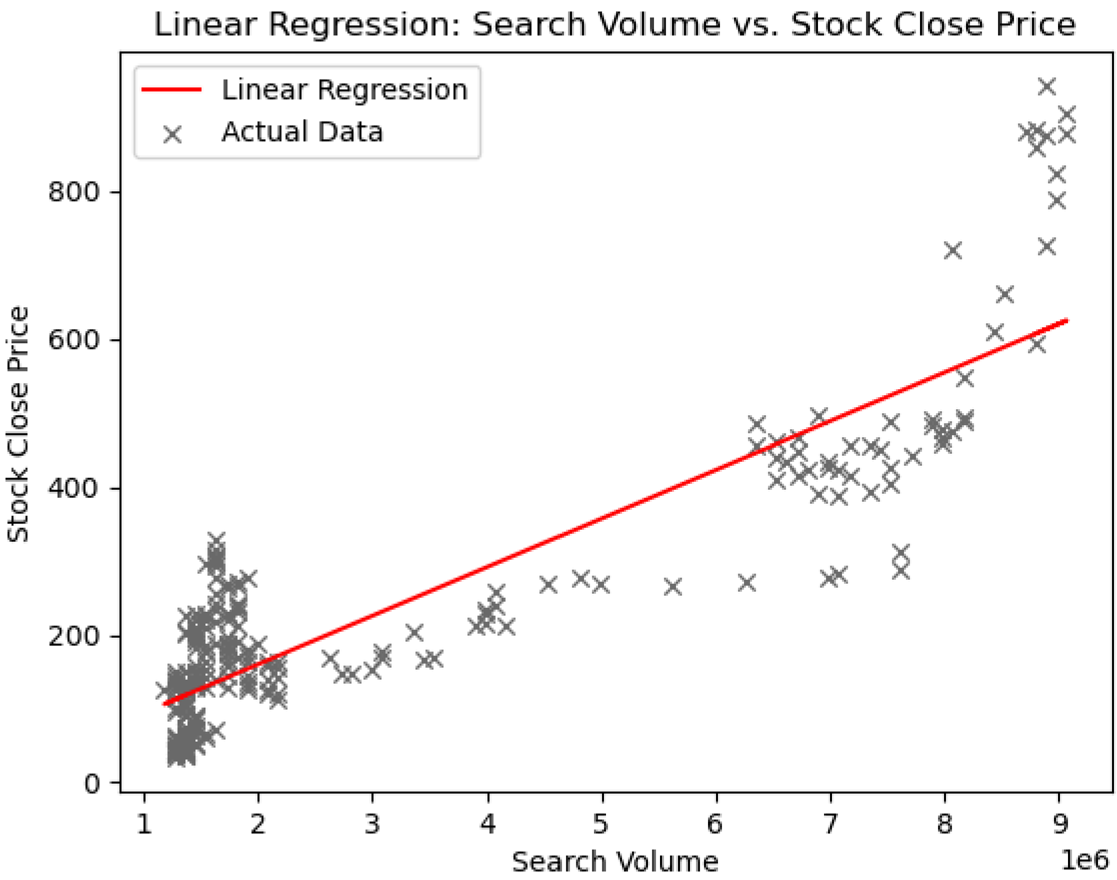

In [16]:
```
# x_train, x_test, y_train, y_test = train_test_split(joined['Search Volume
# plt.scatter(x_train, y_train, label='Trainng dtaa',color='r',alpha=.7)
# plt.scatter(x_test, y_test, label='Testing dtaa',color='b',alpha=.4)
# plt.xlabel('search volume')
# plt.ylabel('stock close price')
# plt.legend()
# plt.title('test traiin split')
# plt.show()

# LR = LinearRegression()
# LR.fit(x.values.reshape(-1,1),y)
# predict = LR.predict(x.values.reshape(-1,1))

# plt.plot(x,predict,label='linear',color='b')
# plt.scatter(x,y,label='actual test data',color='g',alpha=.7)
# plt.xlabel('search volume')
# plt.ylabel('stock close price')
# plt.legend()
# plt.show()
# b= LR.coef_
# print(b)
# #R-squared of model
# LR.score(x.values.reshape(-1,1),y.values)
```

In [17]:
```
def calc_regression(x, y):
    x_reshaped = x.values.reshape(-1, 1)

    # Create and fit the linear regression model
    LR = LinearRegression()
    LR.fit(x_reshaped, y)

    # Predict using the fitted model
    predict = LR.predict(x_reshaped)
    # Display the coefficient (slope) of the linear regression model
    coef = LR.coef_
    print("Coefficient (slope) of the linear regression model:", coef)

    # Calculate R-squared (coefficient of determination) of the model
    r_squared = LR.score(x_reshaped, y)
    print("R-squared of the linear regression model:", r_squared)

    return predict
```

In [18]:
```
def plot_regression(x, y, predict):
    # Plotting the linear regression line and actual data
    plt.plot(x, predict, label='Linear Regression', color='r')
    plt.scatter(x, y, label='Actual Data', color='dimgray', alpha=1, linewi
    plt.xlabel('Search Volume')
    plt.ylabel('Stock Close Price')
    plt.legend()
    plt.title('Linear Regression: Search Volume vs. Stock Close Price')
```

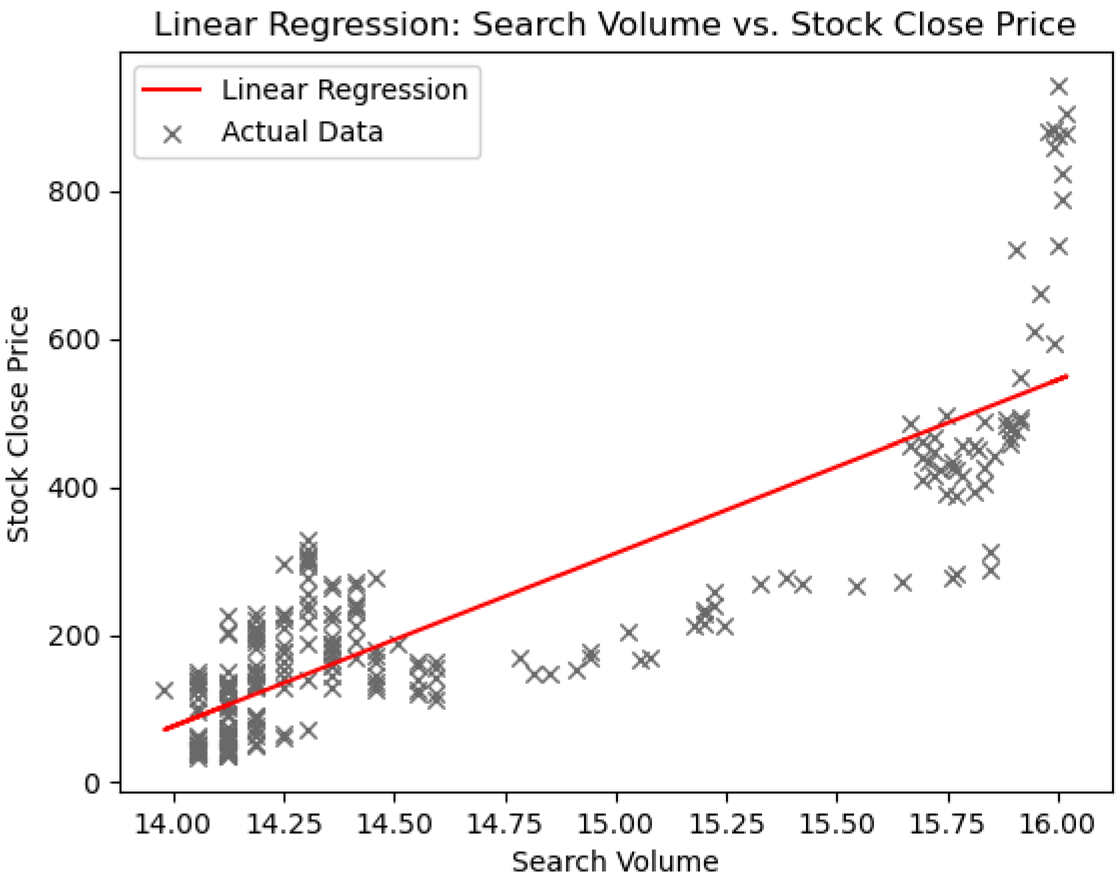
In [19]:
```
predict1 = calc_regression(searchvol['Search Volume'], nvda['Close'])
plot_regression(searchvol['Search Volume'], nvda['Close'], predict1)
plt.savefig("outputs/figure5")
plt.show()
```

```
Coefficient (slope) of the linear regression model: [6.57219112e-05]
R-squared of the linear regression model: 0.7786908923901182
```



In [20]:
```
predict2 = calc_regression(lnsearchvol['Search Volume'], nvda['Close'])
plot_regression(lnsearchvol['Search Volume'], nvda['Close'], predict2)
plt.savefig("outputs/figure8")
plt.show()
```

```
Coefficient (slope) of the linear regression model: [234.11895979]
R-squared of the linear regression model: 0.7247326809795085
```




In [21]:
```
def save_regression(figure):
    # Get the summary as a string
    summary_str = results.summary().as_text()
    # Define the file path where you want to save the summary
    output_file_path = "regression_outputs/"+figure+".txt"
    # Write the summary string to a text file
    with open(output_file_path, 'w') as f:
        f.write(summary_str)
```

In [22]:
```
results = sm.OLS(nvda['Close'].values, sm.add_constant(searchvol['Search Vo
print(results.summary())
save_regression("figure6")
```

```
                            OLS Regression Results
==========================================================================
====
Dep. Variable:                      y   R-squared:
0.779
Model:                            OLS   Adj. R-squared:
0.778
Method:                 Least Squares   F-statistic:                     9
14.8
Date:                Wed, 24 Apr 2024   Prob (F-statistic):           3.96
e-87
Time:                        21:46:22   Log-Likelihood:                -15
43.0
No. Observations:                 262   AIC:                             3
090.
Df Residuals:                     260   BIC:                             3
097.
Df Model:                           1
Covariance Type:            nonrobust
==========================================================================
====
                 coef    std err          t      P>|t|      [0.025      0.
975]
--------------------------------------------------------------------------
----
const         28.4688      8.366      3.403      0.001      11.995      4
4.943
x1          6.572e-05   2.17e-06     30.246      0.000    6.14e-05       7
e-05
==========================================================================
====
Omnibus:                       40.594   Durbin-Watson:
0.064
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               6
3.567
Skew:                           0.903   Prob(JB):                     1.57
e-14
Kurtosis:                       4.600   Cond. No.                     5.94
e+06
==========================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
[2] The condition number is large, 5.94e+06. This might indicate that ther
e are
strong multicollinearity or other numerical problems.
```

In [23]:
```
results = sm.OLS(nvda['Close'].values, sm.add_constant(lnsearchvol['Search
print(results.summary())
save_regression("figure9")
```

```
                            OLS Regression Results
============================================================================
====
Dep. Variable:                      y   R-squared:
0.725
Model:                            OLS   Adj. R-squared:
0.724
Method:                 Least Squares   F-statistic:                       6
84.5
Date:                Wed, 24 Apr 2024   Prob (F-statistic):             8.54
e-75
Time:                        21:46:22   Log-Likelihood:                  -15
71.6
No. Observations:                 262   AIC:                               3
147.
Df Residuals:                     260   BIC:                               3
154.
Df Model:                           1
Covariance Type:            nonrobust
============================================================================
====
                 coef    std err          t      P>|t|      [0.025      0.
975]
----------------------------------------------------------------------------
----
const      -3201.6927    130.967    -24.447      0.000   -3459.584    -294
3.801
x1           234.1190      8.948     26.164      0.000     216.499      25
1.739
============================================================================
====
Omnibus:                       66.055   Durbin-Watson:
0.052
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                 15
3.230
Skew:                           1.194   Prob(JB):                       5.33
e-34
Kurtosis:                       5.887   Cond. No.
319.
============================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
```

In [24]:
```
SnP = read_data("data/SPY.csv")
#cleaning data
SnP = SnP.drop(columns=['Open','High','Low','Adj Close'])
display(SnP)
```

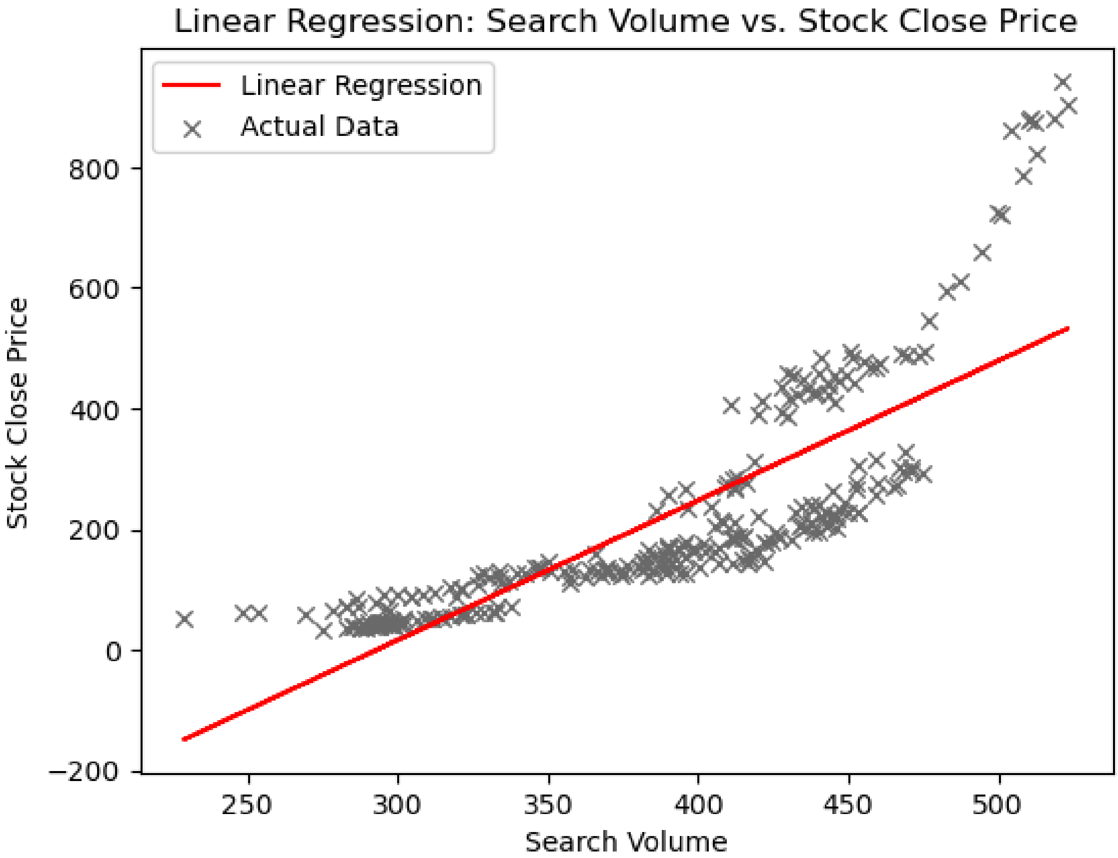
| | Date | Close | Volume |
|---|---|---|---|
| **0** | 2019-04-15 | 290.019989 | 228726700 |
| **1** | 2019-04-22 | 293.410004 | 251486900 |
| **2** | 2019-04-29 | 294.029999 | 331555200 |
| **3** | 2019-05-06 | 288.100006 | 559396700 |
| **4** | 2019-05-13 | 285.839996 | 455352700 |
| **...** | ... | ... | ... |
| **257** | 2024-03-18 | 521.210022 | 358522300 |
| **258** | 2024-03-25 | 523.070007 | 293270500 |
| **259** | 2024-04-01 | 518.429993 | 367203800 |
| **260** | 2024-04-08 | 510.850006 | 361747100 |
| **261** | 2024-04-15 | 504.450012 | 92101400 |

262 rows × 3 columns

In [25]:
```
multi= pd.concat([lnsearchvol['Search Volume'],SnP['Close']],axis=1)
```

In [26]:
```
predict3 = calc_regression(SnP['Close'], nvda['Close'])
plot_regression(SnP['Close'], nvda['Close'], predict3)
```

```
Coefficient (slope) of the linear regression model: [2.31572742]
R-squared of the linear regression model: 0.651732842120399
```

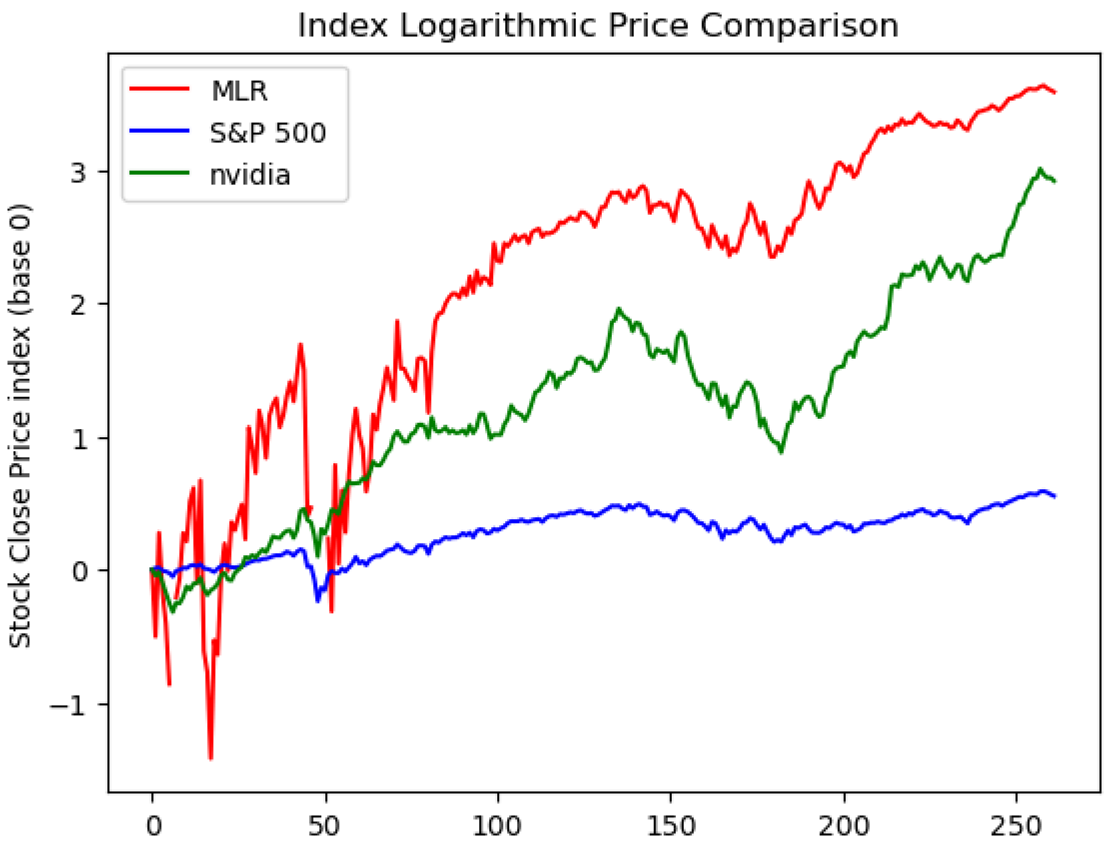



In [27]:
```

multi_reshaped = multi.values
multi_reshaped
# Create and fit the linear regression model
LR = LinearRegression()
LR.fit(multi_reshaped, nvda['Close'])

# Predict nvidia price based on ai search volume and s&p 500 regression
predict4 = LR.predict(multi_reshaped)
```

In [28]:

```
# Plotting the linear regression lines adn
plt.plot(np.log(predict4/predict4[0]), label='MLR', color='r')
plt.plot(np.log(SnP['Close']/SnP['Close'][0]),label='S&P 500 ',color='b')
plt.plot(np.log(nvda['Close']/nvda['Close'][0]),label= 'nvidia',color='g')
#plt.xlabel('time')
plt.ylabel('Stock Close Price index (base 0)')
plt.legend()
plt.title('Index Logarithmic Price Comparison ')
plt.savefig("outputs/figure11")
plt.show()
```

```
C:\Users\jarvit\AppData\Local\Temp\ipykernel_20272\2289097429.py:2: Runtim
eWarning: invalid value encountered in log
  plt.plot(np.log(predict4/predict4[0]), label='MLR', color='r')
```



In [29]:
```
results= sm.OLS(nvda['Close'].values, sm.add_constant(multi).values).fit()
print(results.summary())
save_regression("figure10")
```

```
                            OLS Regression Results
==============================================================================
====
Dep. Variable:                      y   R-squared:
0.848
Model:                            OLS   Adj. R-squared:
0.846
Method:                 Least Squares   F-statistic:                     7
20.5
Date:                Wed, 24 Apr 2024   Prob (F-statistic):          1.50e
-106
Time:                        21:46:22   Log-Likelihood:                  -14
94.1
No. Observations:                 262   AIC:                             2
994.
Df Residuals:                     259   BIC:                             3
005.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
====
                 coef    std err          t      P>|t|      [0.025      0.
975]
------------------------------------------------------------------------------
----
const      -2566.9480    107.041    -23.981      0.000   -2777.730   -235
6.166
x1           156.3712      8.568     18.250      0.000     139.499     17
3.243
x2             1.2919      0.089     14.456      0.000       1.116
1.468
==============================================================================
====
Omnibus:                      103.416   Durbin-Watson:
0.057
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               44
6.523
Skew:                           1.590   Prob(JB):                       1.09
e-97
Kurtosis:                       8.549   Cond. No.                       9.39
e+03
==============================================================================
====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
[2] The condition number is large, 9.39e+03. This might indicate that ther
e are
strong multicollinearity or other numerical problems.
```

In [ ]: