In [1]:
```
import pandas as pd
import matplotlib .pyplot as plt
```

In [2]:
```
pd.read_csv(r"C:\Users\HP\Documents\NVIDIA_STOCK.csv")
```

Out[2]:

	Price	Adj Close	Close	High	Low	Open	Volume
0	Ticker	NVDA	NVDA	NVDA	NVDA	NVDA	NVDA
1	Date	NaN	NaN	NaN	NaN	NaN	NaN
2	2018-01-02	4.929879665374756	4.983749866485596	4.987500190734863	4.862500190734863	4.894499778747559	355616000
3	2018-01-03	5.254334926605225	5.3117499351501465	5.34250020980835	5.09375	5.102499961853027	914704000
4	2018-01-04	5.2820329666137695	5.339749813079834	5.451250076293945	5.317249774932861	5.394000053405762	583268000
...	...	...	...	...	...	...	...
1694	2024-09-24	120.8616714477539	120.87000274658203	121.80000305175781	115.37999725341797	116.5199966430664	354966800
1695	2024-09-25	123.50149536132812	123.51000213623047	124.94000244140625	121.61000061035156	122.0199966430664	284692900
1696	2024-09-26	124.03145599365234	124.04000091552734	127.66999816894531	121.80000305175781	126.80000305175781	302582900
1697	2024-09-27	121.39163970947266	121.4000015258789	124.02999877929688	119.26000213623047	123.97000122070312	271009200
1698	2024-09-30	121.43163299560547	121.44000244140625	121.5	118.1500015258789	118.30999755859375	226553700

1699 rows × 7 columns

In [3]:
```
NVIDIA_df= pd.read_csv(r"C:\Users\HP\Documents\NVIDIA_STOCK.csv")
```

In [4]:
```
NVIDIA_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1699 entries, 0 to 1698
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   Price      1699 non-null   object
 1   Adj Close  1698 non-null   object
 2   Close      1698 non-null   object
 3   High       1698 non-null   object
 4   Low        1698 non-null   object
 5   Open       1698 non-null   object
 6   Volume     1698 non-null   object
dtypes: object(7)
memory usage: 93.0+ KB
```

In [5]:
```
NVIDIA_df.shape
```

Out[5]:
```
(1699, 7)
```

In [6]:
```
NVIDIA_df.isnull().sum()
```

Out[6]:
```
Price        0
Adj Close    1
Close        1
High         1
Low          1
Open         1
Volume       1
dtype: int64
```

In [7]:
```
correct_columns = ['Date', 'Adj Close', 'Close', 'High', 'Low', 'Open', 'Volume']
NVIDIA_df = NVIDIA_df.iloc[2:].copy()  # Skip the first two junk rows
NVIDIA_df.columns = correct_columns  # Assign correct column names
```

In [8]:
```
NVIDIA_df.columns.tolist()
```

Out[8]:
```
['Date', 'Adj Close', 'Close', 'High', 'Low', 'Open', 'Volume']
```

In [9]:
```
# Reset index
NVIDIA_df.reset_index(drop=True, inplace=True)
```

In [10]:
```
print(NVIDIA_df)
```

```
           Date           Adj Close               Close                High  \
0     2018-01-02   4.929879665374756   4.983749866485596   4.987500190734863
1     2018-01-03   5.254334926605225  5.3117499351501465    5.34250020980835
2     2018-01-04  5.2820329666137695   5.339749813079834   5.451250076293945
3     2018-01-05   5.326793670654297   5.385000228881836   5.422749996185303
4     2018-01-08   5.490012168884277   5.550000190734863               5.625
...          ...                 ...                 ...                 ...
1692  2024-09-24   120.8616714477539  120.87000274658203  121.80000305175781
1693  2024-09-25  123.50149536132812  123.51000213623047  124.94000244140625
1694  2024-09-26  124.03145599365234  124.04000091552734  127.66999816894531
1695  2024-09-27  121.39163970947266   121.4000015258789  124.02999877929688
1696  2024-09-30  121.43163299560547  121.44000244140625               121.5

                    Low                 Open     Volume
0      4.862500190734863   4.894499778747559  355616000
1                5.09375   5.102499961853027  914704000
2      5.317249774932861   5.394000053405762  583268000
3     5.2769999504089355   5.354750156402588  580124000
4     5.4644999504089355   5.510000228881836  881216000
...                  ...                 ...        ...
1692  115.37999725341797   116.5199966430664  354966800
1693  121.61000061035156   122.0199966430664  284692900
1694  121.80000305175781  126.80000305175781  302582900
1695  119.26000213623047  123.97000122070312  271009200
1696   118.1500015258789  118.30999755859375  226553700

[1697 rows x 7 columns]
```

In [11]:
```
NVIDIA_df['Date'] = pd.to_datetime(NVIDIA_df['Date'], errors='coerce')
for col in correct_columns[1:]:
  NVIDIA_df[col] = pd.to_numeric(NVIDIA_df[col], errors='coerce')
```

In [12]:
```
print(NVIDIA_df.head())
```

```
        Date  Adj Close     Close      High       Low      Open     Volume
0 2018-01-02   4.929880  4.98375  4.98750  4.86250  4.89450  355616000
1 2018-01-03   5.254335  5.31175  5.34250  5.09375  5.10250  914704000
2 2018-01-04   5.282033  5.33975  5.45125  5.31725  5.39400  583268000
3 2018-01-05   5.326794  5.38500  5.42275  5.27700  5.35475  580124000
4 2018-01-08   5.490012  5.55000  5.62500  5.46450  5.51000  881216000
```

In [13]:
```
NVIDIA_df.corr(numeric_only=True)
```

Out[13]:

	Adj Close	Close	High	Low	Open	Volume
Adj Close	1.000000	1.000000	0.999662	0.999763	0.999329	-0.113866
Close	1.000000	1.000000	0.999662	0.999763	0.999330	-0.113864
High	0.999662	0.999662	1.000000	0.999673	0.999780	-0.109508
Low	0.999763	0.999763	0.999673	1.000000	0.999668	-0.117262
Open	0.999329	0.999330	0.999780	0.999668	1.000000	-0.112821
Volume	-0.113866	-0.113864	-0.109508	-0.117262	-0.112821	1.000000

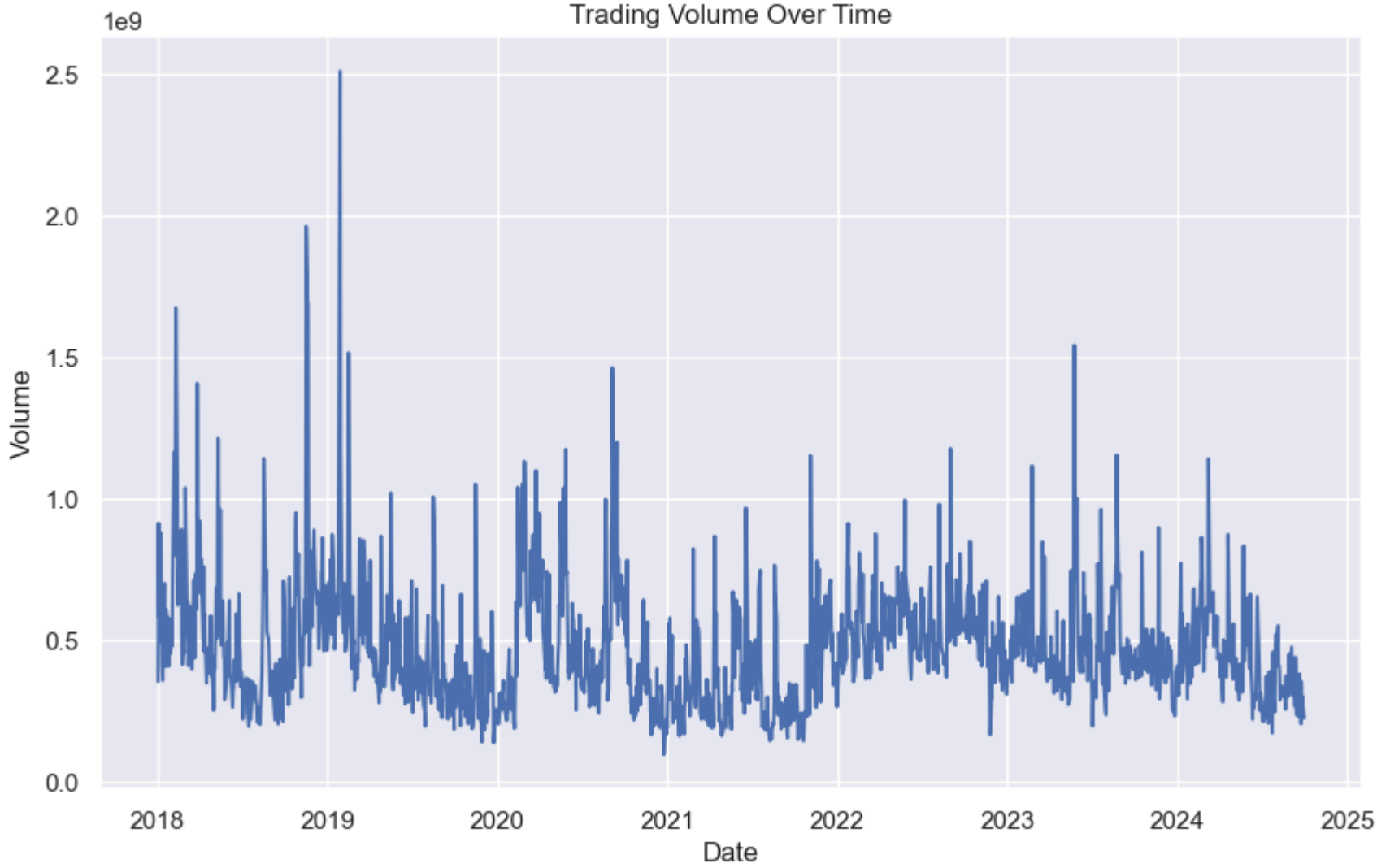
In [15]:
```
import seaborn as  sns
```

In [16]:
```
sns.set()
plt.figure(figsize=(10,6))
sns.lineplot(x='Date', y='Adj Close', data=NVIDIA_df)
plt.title('Adjusted Close Price Over Time')
plt.show()
```

```
C:\Users\HP\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\HP\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

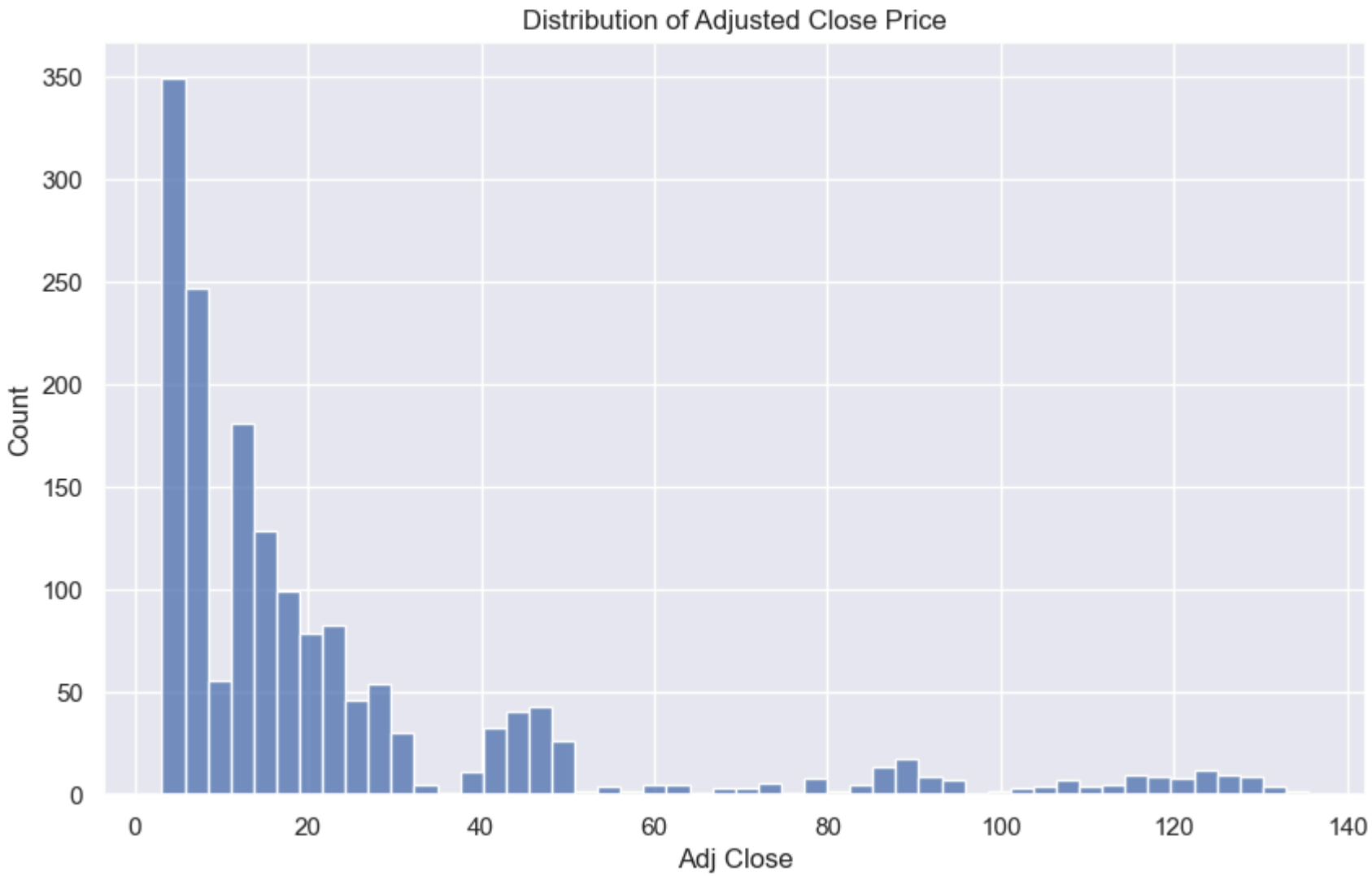
In [17]:
```
plt.figure(figsize=(10,6))
sns.lineplot(x='Date', y='Volume', data=NVIDIA_df)
plt.title('Trading Volume Over Time')
plt.show()
```

```
C:\Users\HP\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\HP\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



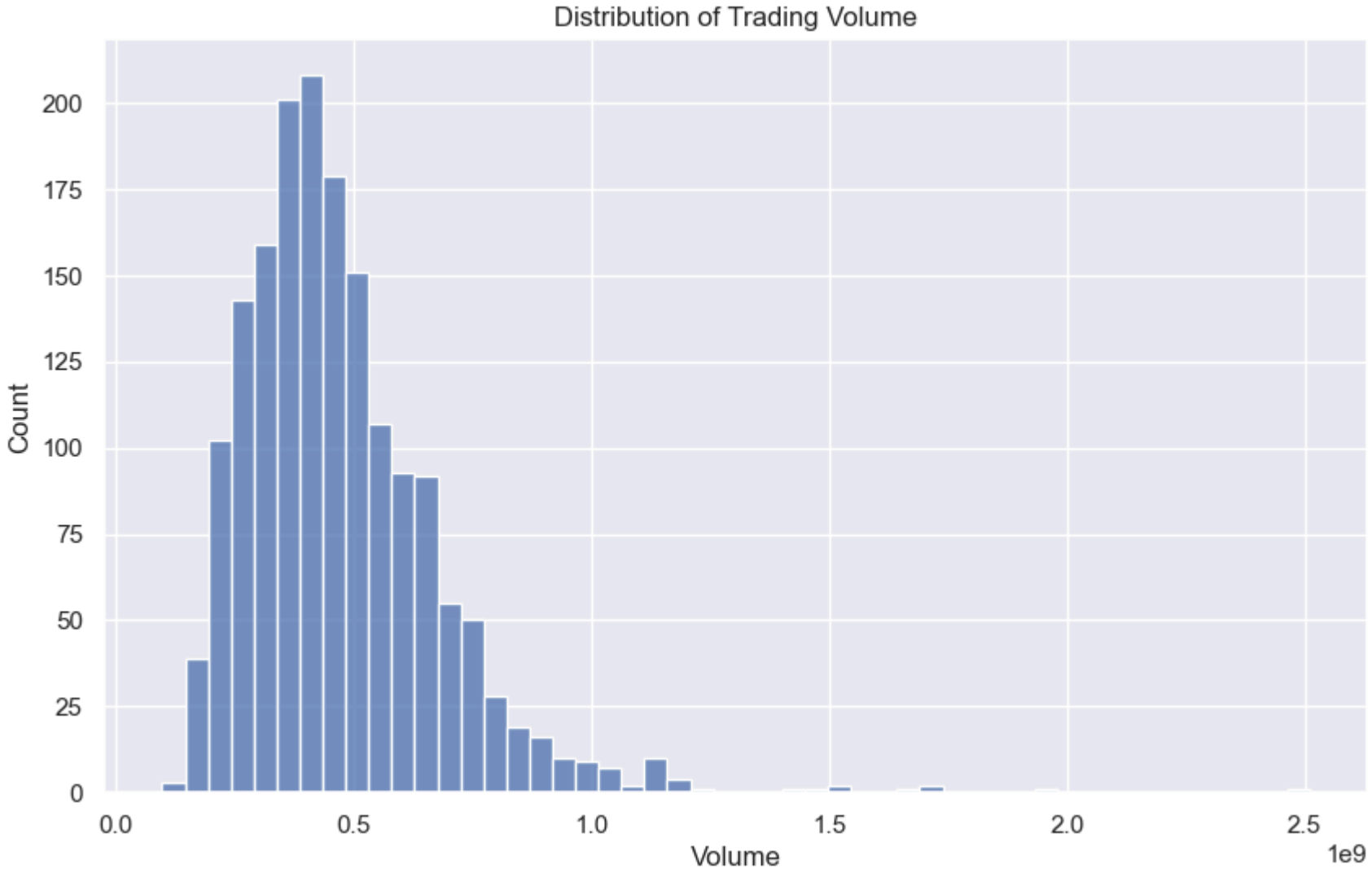
In [18]:
```
sns.set()
plt.figure(figsize=(10,6))
sns.histplot(NVIDIA_df['Adj Close'], bins=50)
plt.title('Distribution of Adjusted Close Price')
plt.show()
```

```
C:\Users\HP\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [19]:
```
plt.figure(figsize=(10,6))
sns.histplot(NVIDIA_df['Volume'], bins=50)
plt.title('Distribution of Trading Volume')
plt.show()
```

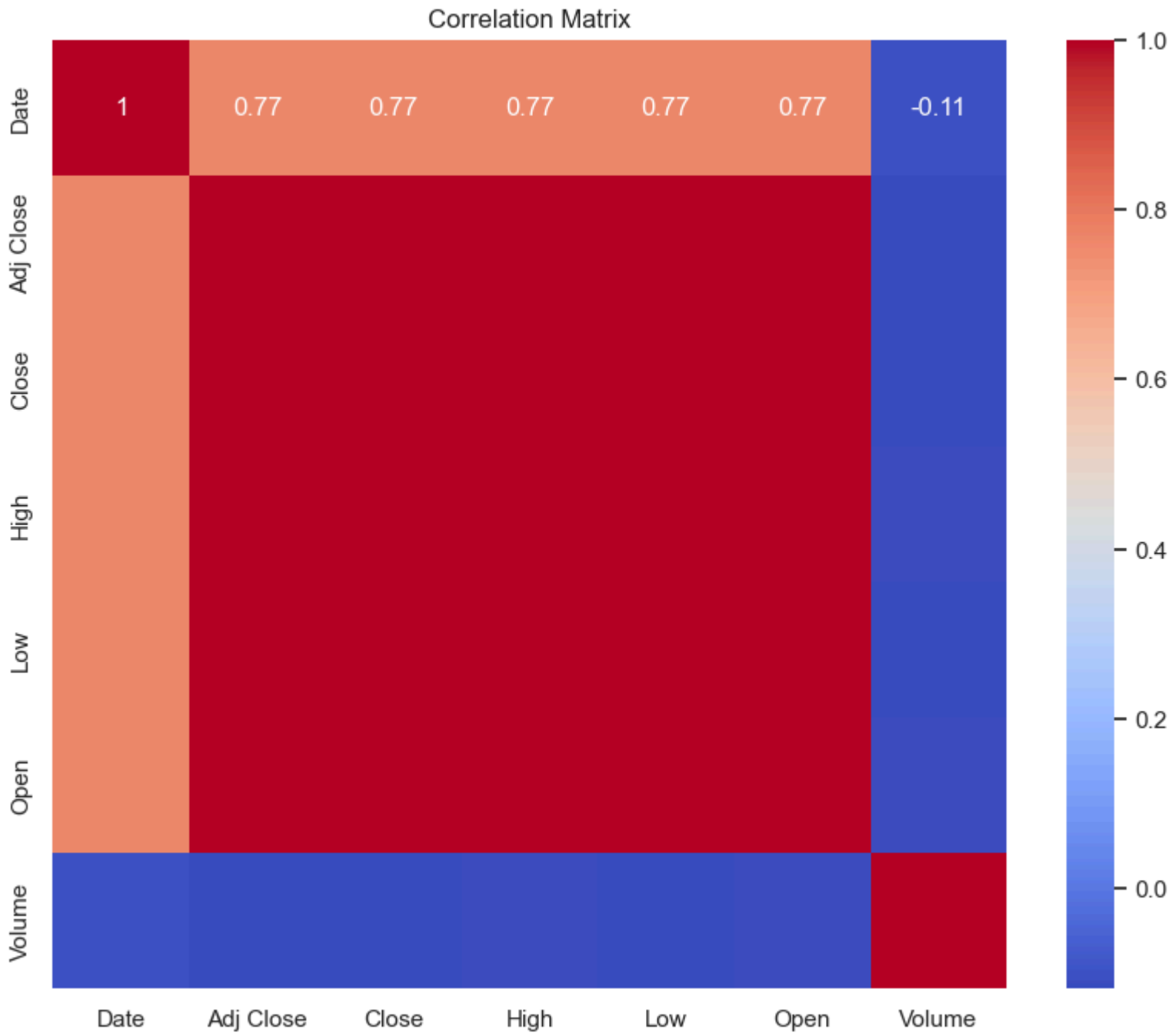
```
C:\Users\HP\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [20]:
```
corr_matrix = NVIDIA_df.corr()
print(corr_matrix)
```

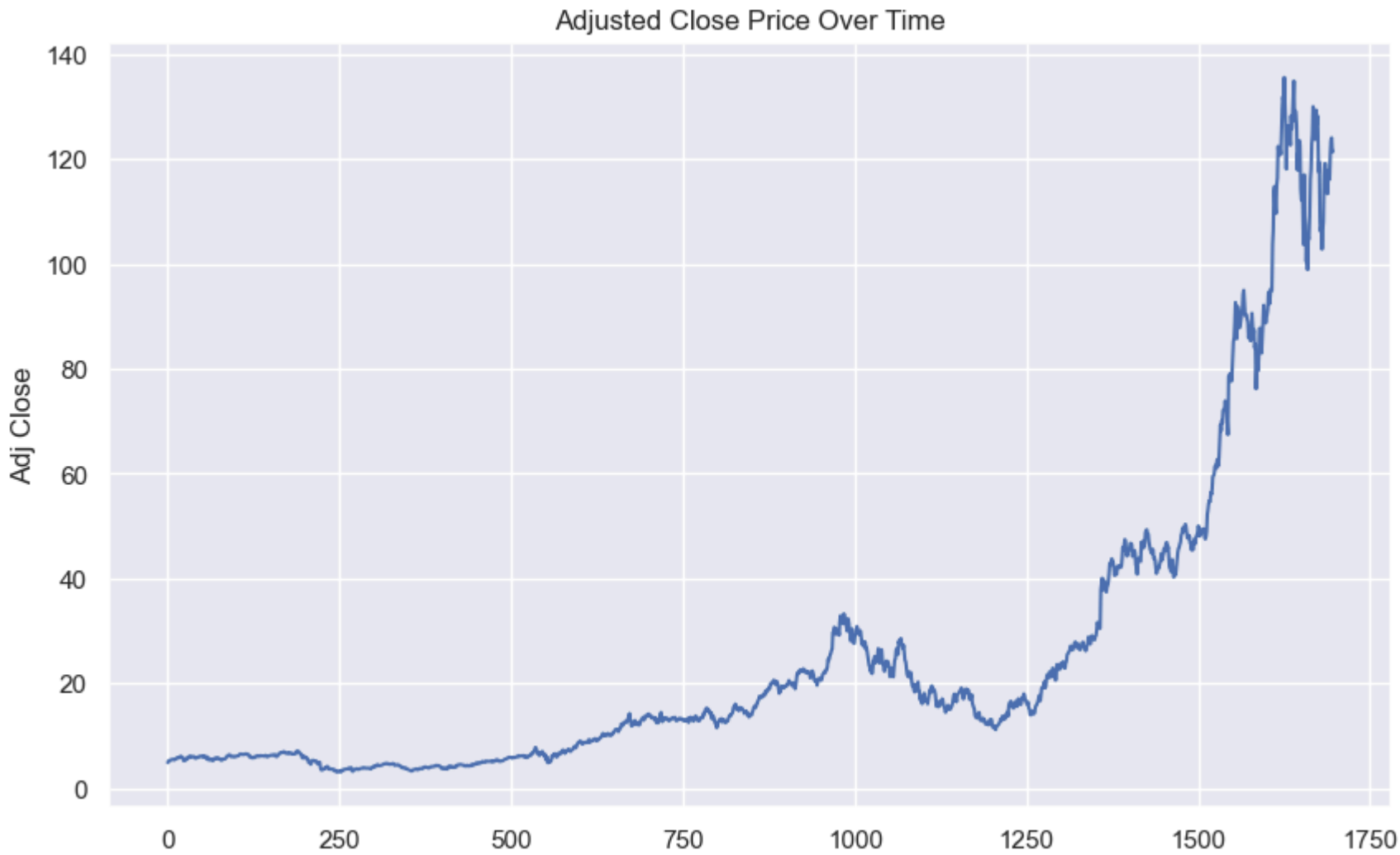
```
                Date  Adj Close     Close      High       Low      Open  \
Date        1.000000   0.766104  0.765909  0.765275  0.766219  0.765174
Adj Close   0.766104   1.000000  1.000000  0.999662  0.999763  0.999329
Close       0.765909   1.000000  1.000000  0.999662  0.999763  0.999330
High        0.765275   0.999662  0.999662  1.000000  0.999673  0.999780
Low         0.766219   0.999763  0.999763  0.999673  1.000000  0.999668
Open        0.765174   0.999329  0.999330  0.999780  0.999668  1.000000
Volume     -0.105869  -0.113866 -0.113864 -0.109508 -0.117262 -0.112821

              Volume
Date       -0.105869
Adj Close  -0.113866
Close      -0.113864
High       -0.109508
Low        -0.117262
Open       -0.112821
Volume      1.000000
```

In [21]:
```
plt.figure(figsize=(10,8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```


Correlation Matrix
Date
Adj Close
Close
High
Low
Open
Volume
1
0.77
0.77
0.77
0.77
0.77
-0.11
Date
Adj Close
Close
High
Low
Open
Volume
1.0
0.8
0.6
0.4
0.2
0.0

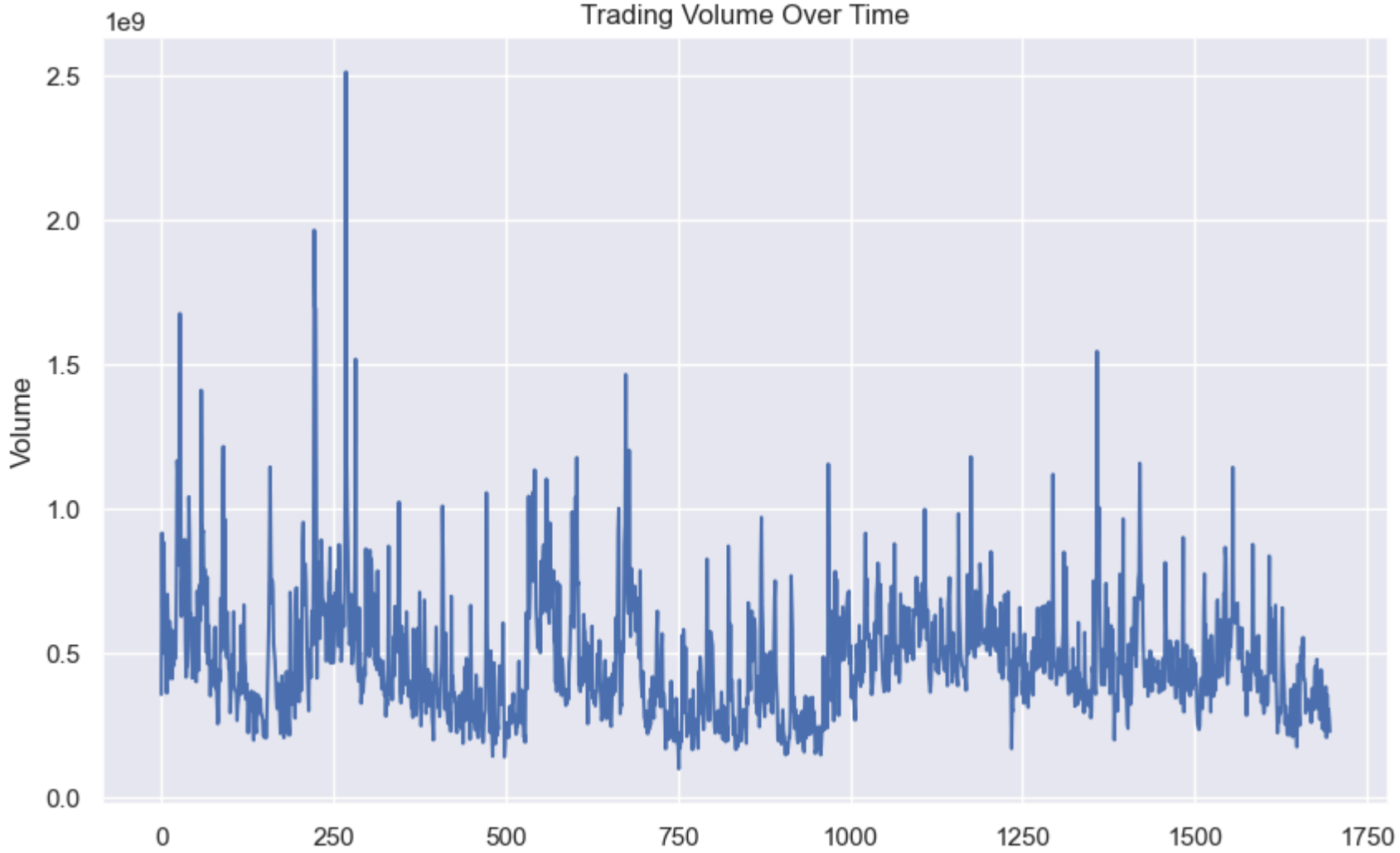
In [22]:
```
plt.figure(figsize=(10,6))
sns.lineplot(x=NVIDIA_df.index, y=NVIDIA_df['Adj Close'])
plt.title('Adjusted Close Price Over Time')
plt.show()
```

```
C:\Users\HP\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\HP\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [23]:
```
plt.figure(figsize=(10,6))
sns.lineplot(x=NVIDIA_df.index, y=NVIDIA_df['Volume'])
plt.title('Trading Volume Over Time')
plt.show()
```

```
C:\Users\HP\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\HP\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [24]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

In [25]:
```
target = "Close"
features = ["Adj Close", "High", "Low", "Open", "Volume"]
X = NVIDIA_df[features]
y = NVIDIA_df[target]
```

In [26]:
```
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
```

In [27]:
```
# Train Random Forest Classifier
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Out[27]:
```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [28]:
```
y_pred = model.predict(X_test)
```

In [29]:
```
print("MSE:", mean_squared_error(y_test, y_pred))
print("MAE:", mean_absolute_error(y_test, y_pred))
print("R^2 Score:", r2_score(y_test, y_pred))
```

```
MSE: 2664.382655144837
MAE: 40.819758959377516
R^2 Score: -1.6669254703438692
```

LINEAR REGRESSION

In [30]:
```
from sklearn.linear_model import LinearRegression
```

In [31]:
```
target = "Close"
features = ["Adj Close", "High", "Low", "Open", "Volume"]
X = NVIDIA_df[features]
y = NVIDIA_df[target]
```

In [32]:
```
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
```

In [33]:
```
# Train Random Forest Classifier
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[33]:
```
▾ LinearRegression

LinearRegression()
```

In [34]:
```
y_pred = model.predict(X_test)
```

In [35]:
```
print("MSE:", mean_squared_error(y_test, y_pred))
print("MAE:", mean_absolute_error(y_test, y_pred))
print("R^2 Score:", r2_score(y_test, y_pred))
```

```
MSE: 0.00010964789972121377
MAE: 0.008906863779295674
R^2 Score: 0.9999998902474553
```

In [36]:
```
# Predict the entire dataset
y_pred_entire_dataset = model.predict(X)
```

In [37]:
```
# Add predictions to the original dataset
NVIDIA_df['Predicted_Close'] = y_pred_entire_dataset
```

In [38]:
```
print(NVIDIA_df['Predicted_Close'])
```

```
0         4.965087
1         5.289988
2         5.317984
3         5.362332
4         5.526071
           ...
1692    120.861345
1693    123.512843
1694    124.044186
1695    121.403287
1696    121.436479
Name: Predicted_Close, Length: 1697, dtype: float64
```

In [ ]: