AMAZON STOCK ANALYSIS USING PYTHON AND POWER BI

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

importing data

In [2]:
```
stocks=pd.read_csv("C:\\Users\\Dell\\OneDrive\\Desktop\\excel books\\AMZN.csv")
```

In [3]:
```
stocks
```

Out[3]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	1997-05-15	0.121875	0.125000	0.096354	0.097917	0.097917	1443120000
1	1997-05-16	0.098438	0.098958	0.085417	0.086458	0.086458	294000000
2	1997-05-19	0.088021	0.088542	0.081250	0.085417	0.085417	122136000
3	1997-05-20	0.086458	0.087500	0.081771	0.081771	0.081771	109344000
4	1997-05-21	0.081771	0.082292	0.068750	0.071354	0.071354	377064000
...	...	...	...	...	...	...	...
6511	2023-03-30	101.550003	103.040001	101.010002	102.000000	102.000000	53633400
6512	2023-03-31	102.160004	103.489998	101.949997	103.290001	103.290001	56704300
6513	2023-04-03	102.300003	103.290001	101.430000	102.410004	102.410004	41135700
6514	2023-04-04	102.750000	104.199997	102.110001	103.949997	103.949997	48662500
6515	2023-04-05	103.910004	103.910004	100.750000	101.099998	101.099998	45103000

6516 rows × 7 columns

Checking for missing values

In [4]:
```
stocks.isnull().sum()
```

Out[4]:
```
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

no missing values

Removing duplicates

In [5]:
```
stocks
```

Out[5]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	1997-05-15	0.121875	0.125000	0.096354	0.097917	0.097917	1443120000
1	1997-05-16	0.098438	0.098958	0.085417	0.086458	0.086458	294000000
2	1997-05-19	0.088021	0.088542	0.081250	0.085417	0.085417	122136000
3	1997-05-20	0.086458	0.087500	0.081771	0.081771	0.081771	109344000
4	1997-05-21	0.081771	0.082292	0.068750	0.071354	0.071354	377064000
...	...	...	...	...	...	...	...
6511	2023-03-30	101.550003	103.040001	101.010002	102.000000	102.000000	53633400
6512	2023-03-31	102.160004	103.489998	101.949997	103.290001	103.290001	56704300
6513	2023-04-03	102.300003	103.290001	101.430000	102.410004	102.410004	41135700
6514	2023-04-04	102.750000	104.199997	102.110001	103.949997	103.949997	48662500
6515	2023-04-05	103.910004	103.910004	100.750000	101.099998	101.099998	45103000

6516 rows × 7 columns

In [6]:
```
stocks=stocks.drop_duplicates()
```

In [7]:
```
stocks
```

Out[7]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	1997-05-15	0.121875	0.125000	0.096354	0.097917	0.097917	1443120000
1	1997-05-16	0.098438	0.098958	0.085417	0.086458	0.086458	294000000
2	1997-05-19	0.088021	0.088542	0.081250	0.085417	0.085417	122136000
3	1997-05-20	0.086458	0.087500	0.081771	0.081771	0.081771	109344000
4	1997-05-21	0.081771	0.082292	0.068750	0.071354	0.071354	377064000
...	...	...	...	...	...	...	...
6511	2023-03-30	101.550003	103.040001	101.010002	102.000000	102.000000	53633400
6512	2023-03-31	102.160004	103.489998	101.949997	103.290001	103.290001	56704300
6513	2023-04-03	102.300003	103.290001	101.430000	102.410004	102.410004	41135700
6514	2023-04-04	102.750000	104.199997	102.110001	103.949997	103.949997	48662500
6515	2023-04-05	103.910004	103.910004	100.750000	101.099998	101.099998	45103000

6516 rows × 7 columns

Checking for data types

In [8]:
```
stocks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6516 entries, 0 to 6515
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       6516 non-null   object 
 1   Open       6516 non-null   float64
 2   High       6516 non-null   float64
 3   Low        6516 non-null   float64
 4   Close      6516 non-null   float64
 5   Adj Close  6516 non-null   float64
 6   Volume     6516 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 407.2+ KB
```

In [9]:
```
stocks['Date']= pd.to_datetime(stocks['Date'])
```

In [10]:
```
stocks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6516 entries, 0 to 6515
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype         
---  ------     --------------  -----         
 0   Date       6516 non-null   datetime64[ns]
 1   Open       6516 non-null   float64       
 2   High       6516 non-null   float64       
 3   Low        6516 non-null   float64       
 4   Close      6516 non-null   float64       
 5   Adj Close  6516 non-null   float64       
 6   Volume     6516 non-null   int64         
dtypes: datetime64[ns](1), float64(5), int64(1)
memory usage: 407.2 KB
```

Exploring the data

In [11]:
```
stocks.describe()
```

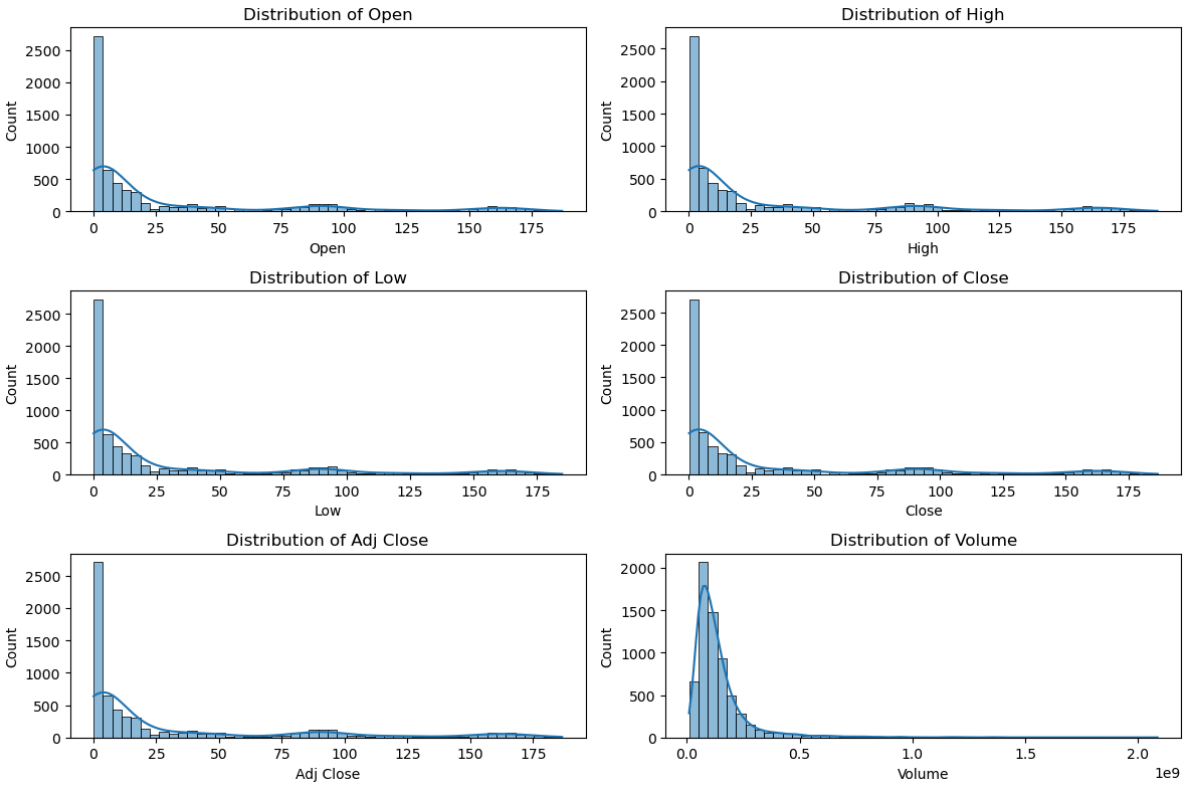
Out[11]:

	Open	High	Low	Close	Adj Close	Volume
count	6516.000000	6516.000000	6516.000000	6516.000000	6516.000000	6.516000e+03
mean	31.611626	31.991995	31.193432	31.599740	31.599740	1.425338e+08
std	48.095343	48.659651	47.464476	48.060258	48.060258	1.401619e+08
min	0.070313	0.072396	0.065625	0.069792	0.069792	9.744000e+06
25%	1.998875	2.028500	1.964750	2.001250	2.001250	6.888182e+07
50%	6.456750	6.535500	6.353250	6.444250	6.444250	1.059050e+08
75%	38.451375	38.688000	38.203001	38.464625	38.464625	1.607700e+08
max	187.199997	188.654007	184.839493	186.570496	186.570496	2.086584e+09

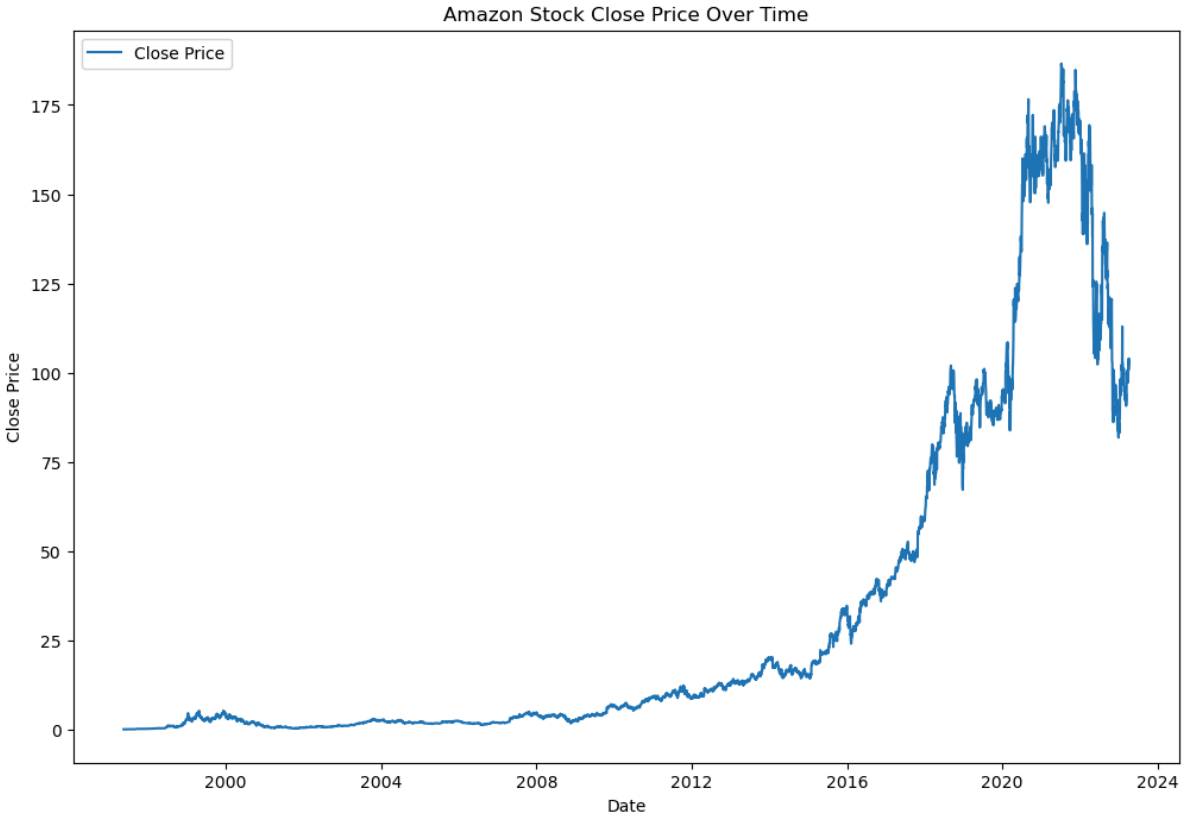
In [12]:
```
##pip install --upgrade seaborn
```

In [13]:
```
plt.figure(figsize=(12, 8))
for i, column in enumerate(['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'],
    plt.subplot(3, 2, i)
    sns.histplot(stocks[column], bins=50, kde=True)
    plt.title(f'Distribution of {column}')
```

```
plt.tight_layout()
plt.show()
```



In [14]:
```
plt.figure(figsize=(12, 8))
plt.plot(stocks['Date'], stocks['Close'], label='Close Price')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.title('Amazon Stock Close Price Over Time')
plt.legend()
plt.show()
```



Average Closing price

In [15]:
```
average_closing_price=stocks['Close'].mean()
print(f"Average Closing Price Of Stocks is {average_closing_price:.2f}")
```

```
Average Closing Price Of Stocks is 31.60
```

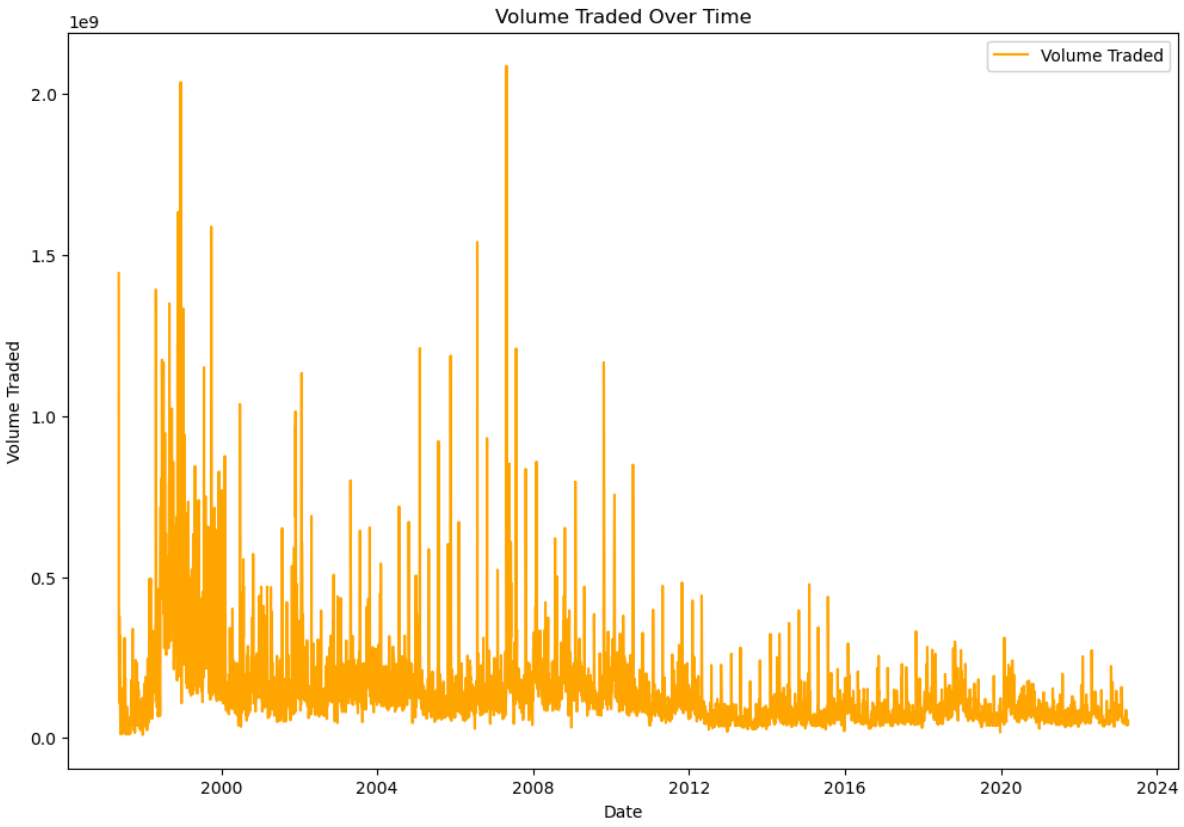
Highest and Lowest closing Prices

In [16]:
```
h_close=stocks['Close'].max()
l_close=stocks['Close'].min()
print(f"Highest closing Price: {h_close:.2f} \nLowest closing price is : {l_close:.
```

```
Highest closing Price: 186.57
Lowest closing price is : 0.07
```

How the volume of stocks traded vary over time?

In [17]:
```
plt.figure(figsize=(12, 8))
plt.plot(stocks['Date'],stocks['Volume'],label='Volume Traded',color='orange')
plt.xlabel('Date')
plt.ylabel('Volume Traded')
plt.title('Volume Traded Over Time')
plt.legend()
plt.show()
```

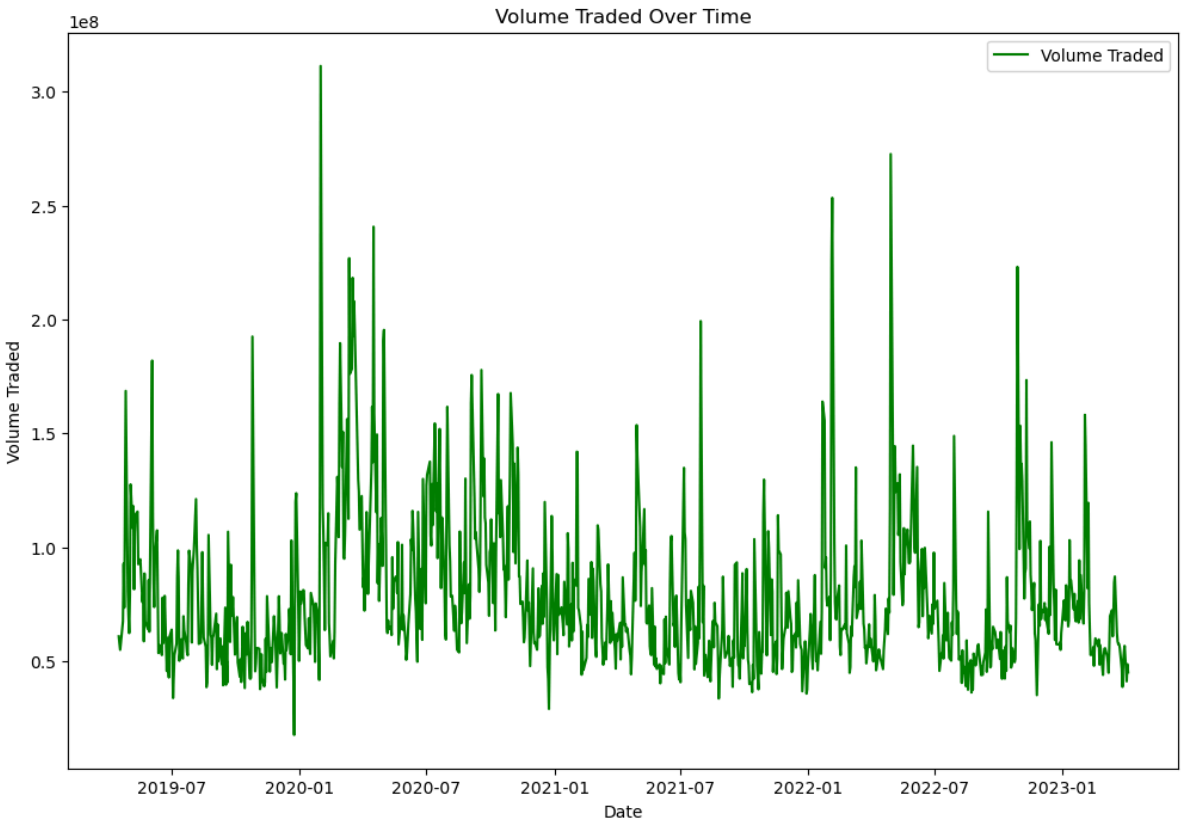


Recent Trend

In [18]:
```
Recent_volume=stocks['Volume'].iloc[5515:]
Recent_dates = stocks['Date'].iloc[5515:]
```

In [19]:
```
plt.figure(figsize=(12, 8))
plt.plot(Recent_dates,Recent_volume,label='Volume Traded',color='green')
plt.xlabel('Date')
plt.ylabel('Volume Traded')
```

```
plt.title('Volume Traded Over Time')
plt.legend()
plt.show()
```



Daily Returns For a Stock

In [20]:
```
stocks['Daily_Return']=stocks['Close'].pct_change()
```

In [21]:
```
stocks
```

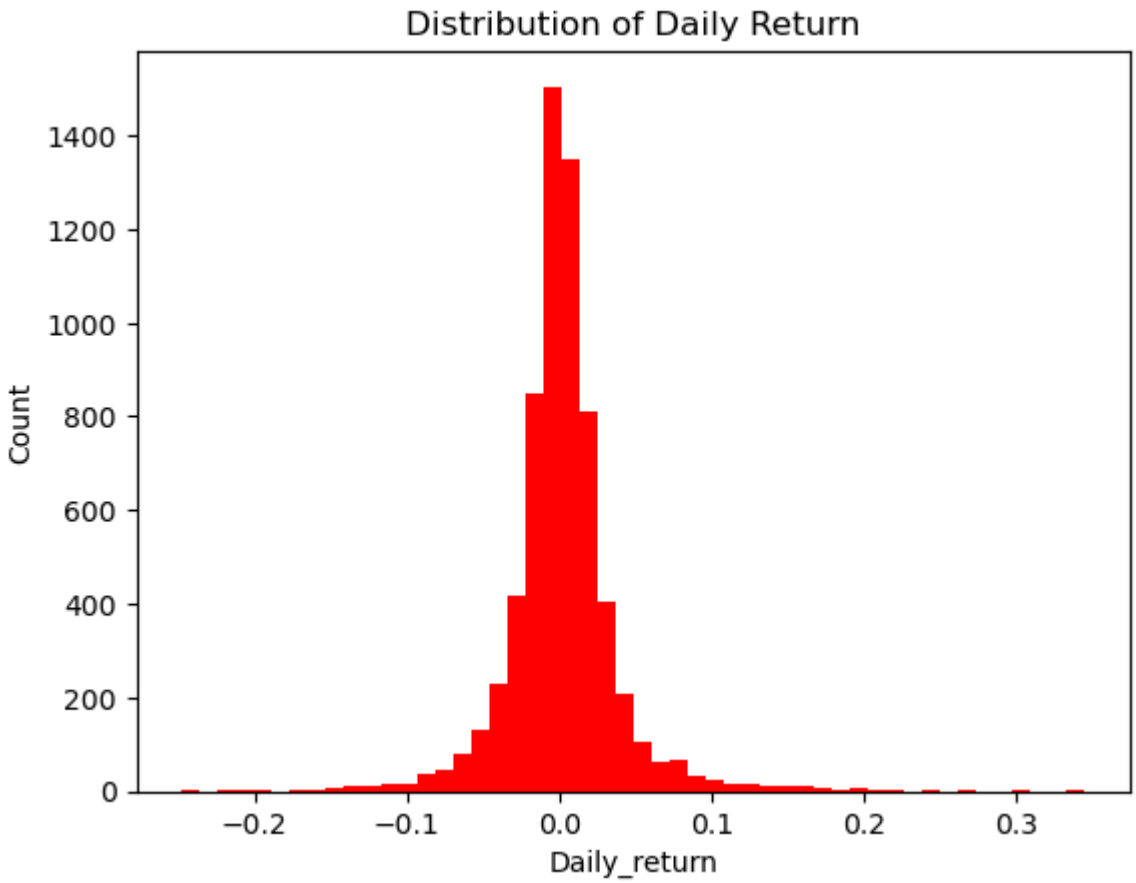
Out[21]:

	Date	Open	High	Low	Close	Adj Close	Volume	Daily_Returr
0	1997-05-15	0.121875	0.125000	0.096354	0.097917	0.097917	1443120000	NaN
1	1997-05-16	0.098438	0.098958	0.085417	0.086458	0.086458	294000000	-0.11702
2	1997-05-19	0.088021	0.088542	0.081250	0.085417	0.085417	122136000	-0.01204
3	1997-05-20	0.086458	0.087500	0.081771	0.081771	0.081771	109344000	-0.04268
4	1997-05-21	0.081771	0.082292	0.068750	0.071354	0.071354	377064000	-0.12739
...	...	...	...	...	...	...	...	...
6511	2023-03-30	101.550003	103.040001	101.010002	102.000000	102.000000	53633400	0.01745
6512	2023-03-31	102.160004	103.489998	101.949997	103.290001	103.290001	56704300	0.01264
6513	2023-04-03	102.300003	103.290001	101.430000	102.410004	102.410004	41135700	-0.00852
6514	2023-04-04	102.750000	104.199997	102.110001	103.949997	103.949997	48662500	0.01503
6515	2023-04-05	103.910004	103.910004	100.750000	101.099998	101.099998	45103000	-0.02741

6516 rows × 8 columns

In [22]:

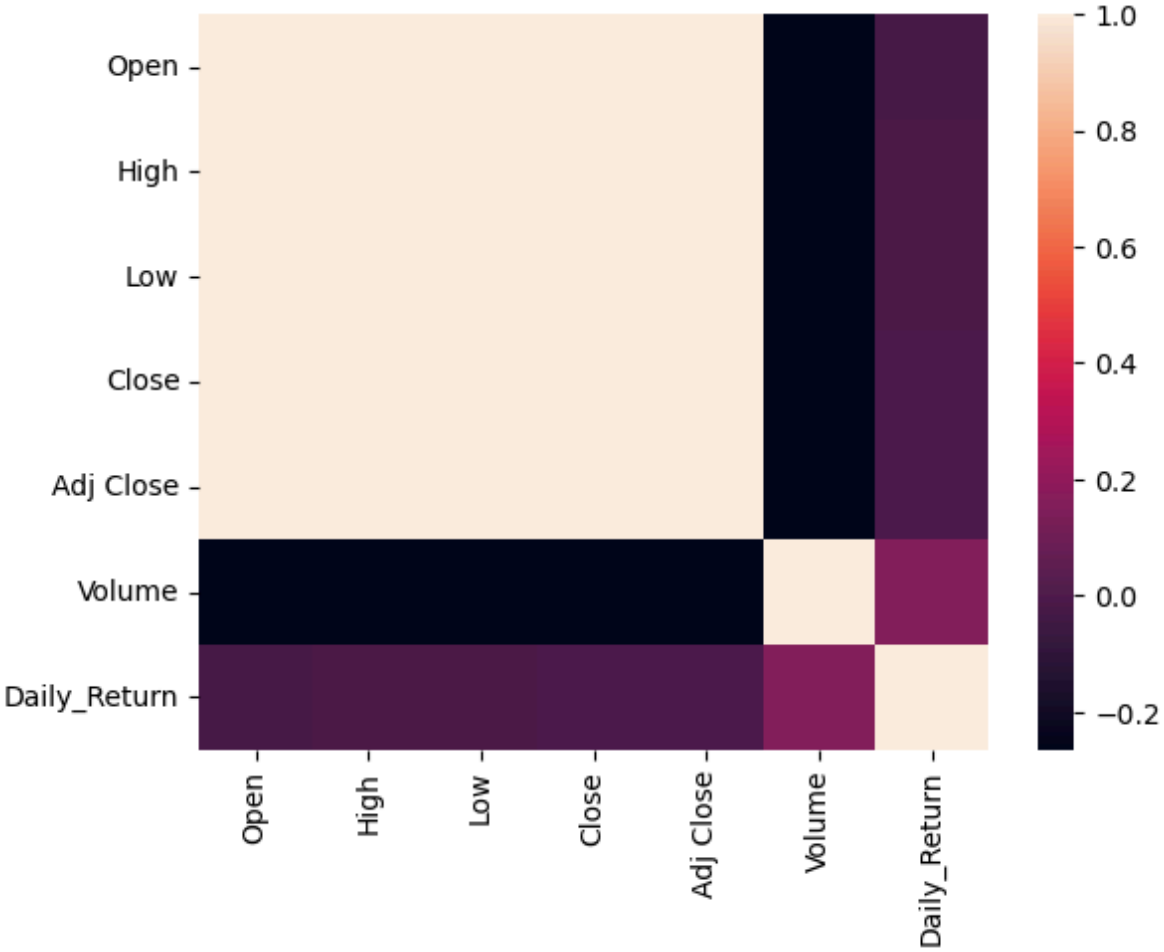
```
plt.hist(stocks['Daily_Return'],bins=50,color='red',label='Daily return')
plt.xlabel("Daily_return")
plt.ylabel("Count")
plt.title('Distribution of Daily Return')
plt.show()
```



Identifying Corelation betweening variables

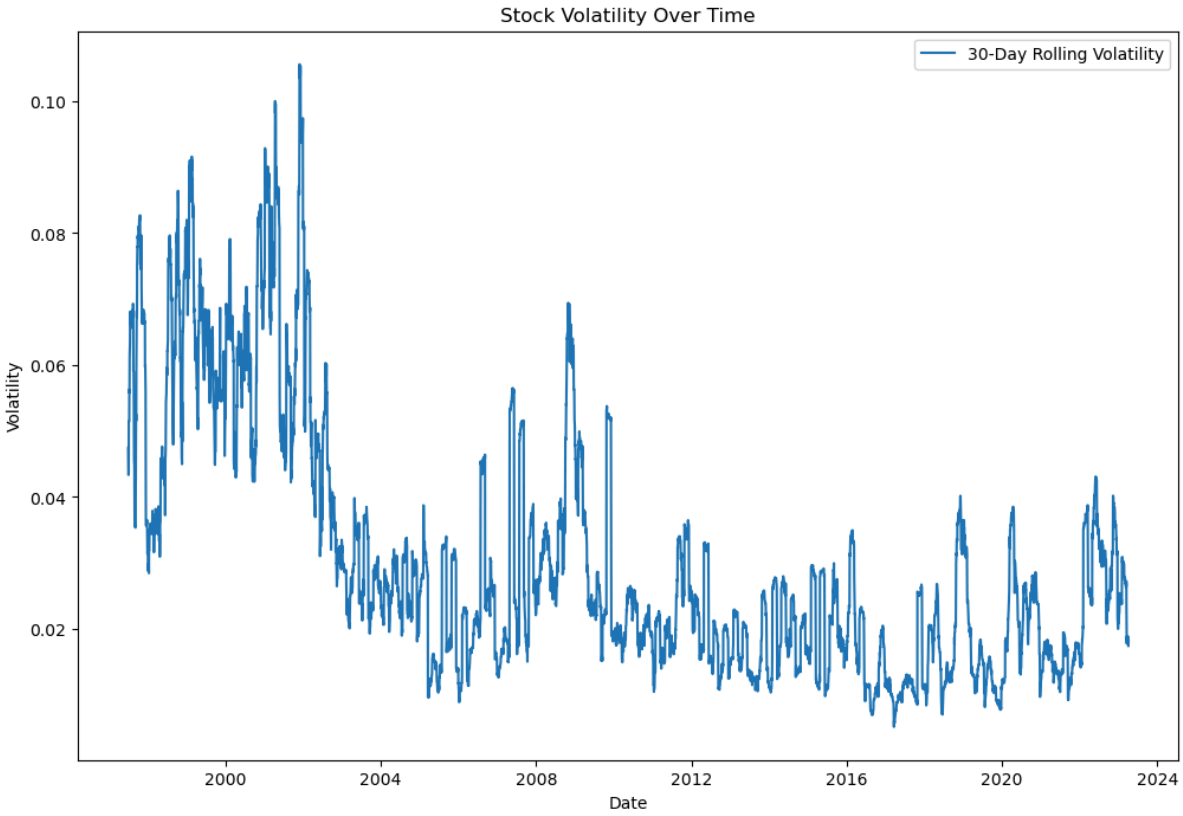
In [23]:
```
sns.heatmap(stocks.corr())
```

Out[23]:
```
<AxesSubplot:>
```



Periods with Highest Volatility

In [24]:
```
rolling_volatility = stocks['Daily_Return'].rolling(window=30).std()
plt.figure(figsize=(12, 8))
plt.plot(stocks['Date'], rolling_volatility, label='30-Day Rolling Volatility')
plt.title('Stock Volatility Over Time')
plt.xlabel('Date')
plt.ylabel('Volatility')
plt.legend()
plt.show()
```



In [ ]:

Average Volatility

In [25]:
```
rolling_volatility=stocks['Daily_Return'].std()
print(f"Average Volatility : {rolling_volatility}")
```

```
Average Volatility : 0.03609239649464962
```

In [26]:
```
stocks.set_index(stocks["Date"])
```

Out[26]:

	Date	Open	High	Low	Close	Adj Close	Volume	Daily_Retur
Date								
1997-05-15	1997-05-15	0.121875	0.125000	0.096354	0.097917	0.097917	1443120000	Na
1997-05-16	1997-05-16	0.098438	0.098958	0.085417	0.086458	0.086458	294000000	-0.11702
1997-05-19	1997-05-19	0.088021	0.088542	0.081250	0.085417	0.085417	122136000	-0.01204
1997-05-20	1997-05-20	0.086458	0.087500	0.081771	0.081771	0.081771	109344000	-0.04268
1997-05-21	1997-05-21	0.081771	0.082292	0.068750	0.071354	0.071354	377064000	-0.12739
...	...	...	...	...	...	...	...	
2023-03-30	2023-03-30	101.550003	103.040001	101.010002	102.000000	102.000000	53633400	0.01745
2023-03-31	2023-03-31	102.160004	103.489998	101.949997	103.290001	103.290001	56704300	0.01264
2023-04-03	2023-04-03	102.300003	103.290001	101.430000	102.410004	102.410004	41135700	-0.00852
2023-04-04	2023-04-04	102.750000	104.199997	102.110001	103.949997	103.949997	48662500	0.01503
2023-04-05	2023-04-05	103.910004	103.910004	100.750000	101.099998	101.099998	45103000	-0.02741

6516 rows × 8 columns

In [27]:
```
stocks['year']=stocks['Date'].dt.year
```

In [28]:
```
stocks
```

Out[28]:

	Date	Open	High	Low	Close	Adj Close	Volume	Daily_Returr
0	1997-05-15	0.121875	0.125000	0.096354	0.097917	0.097917	1443120000	NaN
1	1997-05-16	0.098438	0.098958	0.085417	0.086458	0.086458	294000000	-0.11702
2	1997-05-19	0.088021	0.088542	0.081250	0.085417	0.085417	122136000	-0.01204
3	1997-05-20	0.086458	0.087500	0.081771	0.081771	0.081771	109344000	-0.04268
4	1997-05-21	0.081771	0.082292	0.068750	0.071354	0.071354	377064000	-0.12739
...	...	...	...	...	...	...	...	...
6511	2023-03-30	101.550003	103.040001	101.010002	102.000000	102.000000	53633400	0.01745
6512	2023-03-31	102.160004	103.489998	101.949997	103.290001	103.290001	56704300	0.01264
6513	2023-04-03	102.300003	103.290001	101.430000	102.410004	102.410004	41135700	-0.00852
6514	2023-04-04	102.750000	104.199997	102.110001	103.949997	103.949997	48662500	0.01503
6515	2023-04-05	103.910004	103.910004	100.750000	101.099998	101.099998	45103000	-0.02741

6516 rows × 9 columns

In [29]:
```
stocks['month']=stocks['Date'].dt.month
```

In [30]:
```
stocks['day']=stocks['Date'].dt.month
```

In [31]:
```
stocks['Quarter']=stocks['Date'].dt.quarter
```

In [32]:
```
stocks
```

Out[32]:

	Date	Open	High	Low	Close	Adj Close	Volume	Daily_Returr
0	1997-05-15	0.121875	0.125000	0.096354	0.097917	0.097917	1443120000	NaN
1	1997-05-16	0.098438	0.098958	0.085417	0.086458	0.086458	294000000	-0.11702
2	1997-05-19	0.088021	0.088542	0.081250	0.085417	0.085417	122136000	-0.01204
3	1997-05-20	0.086458	0.087500	0.081771	0.081771	0.081771	109344000	-0.04268
4	1997-05-21	0.081771	0.082292	0.068750	0.071354	0.071354	377064000	-0.12739
...	...	...	...	...	...	...	...	...
6511	2023-03-30	101.550003	103.040001	101.010002	102.000000	102.000000	53633400	0.01745
6512	2023-03-31	102.160004	103.489998	101.949997	103.290001	103.290001	56704300	0.01264
6513	2023-04-03	102.300003	103.290001	101.430000	102.410004	102.410004	41135700	-0.00852
6514	2023-04-04	102.750000	104.199997	102.110001	103.949997	103.949997	48662500	0.01503
6515	2023-04-05	103.910004	103.910004	100.750000	101.099998	101.099998	45103000	-0.02741

6516 rows × 12 columns

In [33]:
```
stocks['month'] = stocks['Date'].dt.strftime('%B')

stocks
```

Out[33]:

	Date	Open	High	Low	Close	Adj Close	Volume	Daily_Returr
0	1997-05-15	0.121875	0.125000	0.096354	0.097917	0.097917	1443120000	NaN
1	1997-05-16	0.098438	0.098958	0.085417	0.086458	0.086458	294000000	-0.11702
2	1997-05-19	0.088021	0.088542	0.081250	0.085417	0.085417	122136000	-0.01204
3	1997-05-20	0.086458	0.087500	0.081771	0.081771	0.081771	109344000	-0.04268
4	1997-05-21	0.081771	0.082292	0.068750	0.071354	0.071354	377064000	-0.12739
...	...	...	...	...	...	...	...	...
6511	2023-03-30	101.550003	103.040001	101.010002	102.000000	102.000000	53633400	0.01745
6512	2023-03-31	102.160004	103.489998	101.949997	103.290001	103.290001	56704300	0.01264
6513	2023-04-03	102.300003	103.290001	101.430000	102.410004	102.410004	41135700	-0.00852
6514	2023-04-04	102.750000	104.199997	102.110001	103.949997	103.949997	48662500	0.01503
6515	2023-04-05	103.910004	103.910004	100.750000	101.099998	101.099998	45103000	-0.02741

6516 rows × 12 columns

seasonal Trends

In [34]:
```
seasonal_trends = stocks.groupby(['year', 'month'])['Close'].mean()
```

In [35]:
```
seasonal_trends
```

Out[35]:
```
year  month
1997  August         0.111570
      December       0.228161
      July           0.110275
      June           0.076885
      May            0.079427
                        ...
2022  September    123.213811
2023  April        102.486666
      February      99.214211
      January       94.223500
      March         96.546956
Name: Close, Length: 312, dtype: float64
```

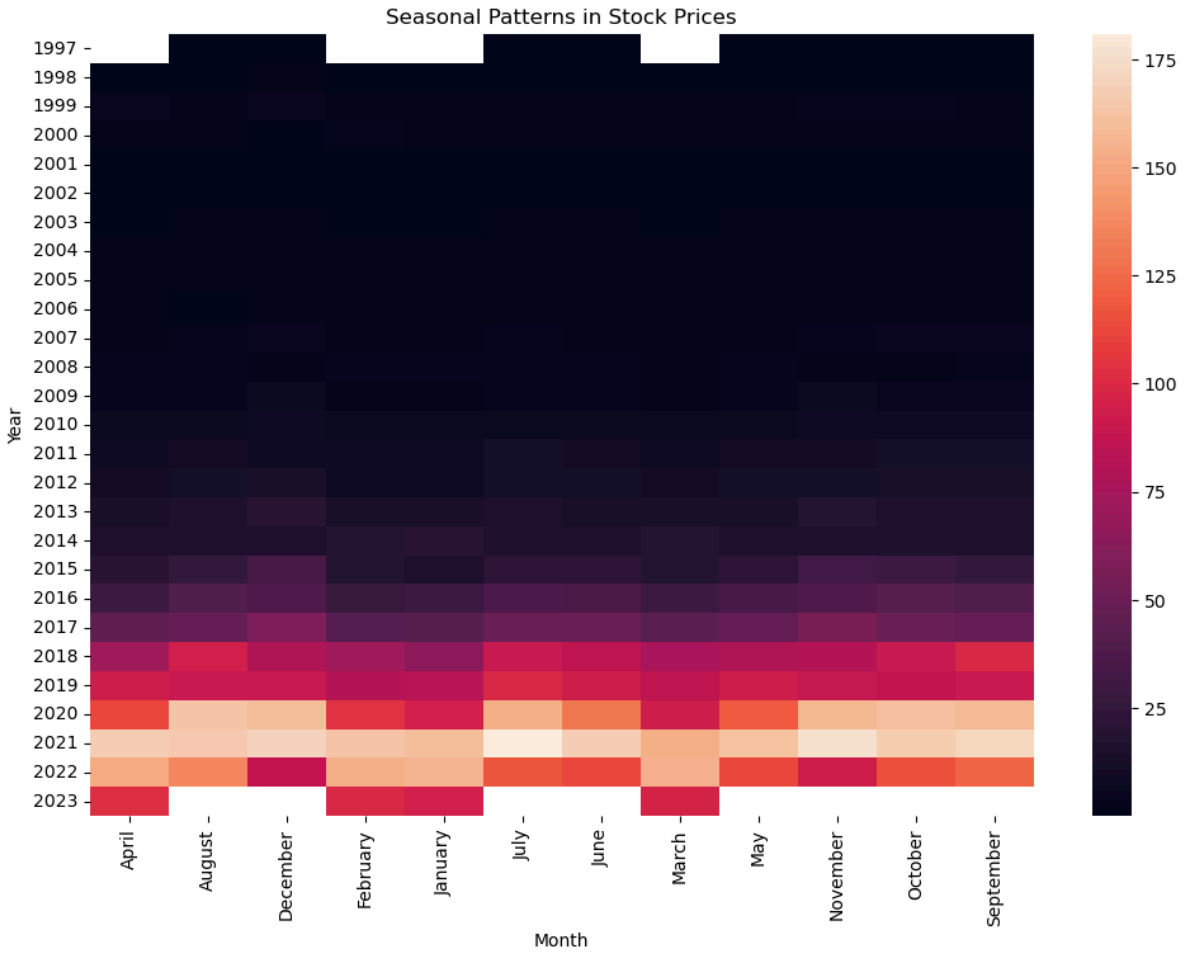
In [36]:
```
seasonal_trends.unstack()
```

Out[36]:

month	April	August	December	February	January	July	June	M
year								
1997	NaN	0.111570	0.228161	NaN	NaN	0.110275	0.076885	
1998	0.381697	0.997725	2.185440	0.264871	0.240195	0.989607	0.577273	0.33
1999	4.551637	2.651137	4.553551	2.731497	3.353989	2.993676	2.789773	3.36
2000	2.818750	1.805299	1.044531	3.623125	3.402500	1.839219	2.273438	3.33
2001	0.665825	0.521000	0.553650	0.675164	0.890662	0.739143	0.707929	0.54
2002	0.727523	0.727386	1.092119	0.653579	0.603952	0.739000	0.869550	0.76
2003	1.312381	2.109595	2.564591	1.073658	1.075429	1.942386	1.762857	1.25
2004	2.326643	1.896705	2.033182	2.260868	2.697350	2.300143	2.542881	2.11
2005	1.685167	2.211174	2.439095	1.810053	2.118000	1.880250	1.754614	1.72
2006	1.816211	1.389848	1.965575	1.941105	2.261225	1.662875	1.763409	1.81
2007	2.331475	3.841065	4.579900	1.974763	1.878150	3.714214	3.517952	1.93
2008	3.854091	4.114762	2.473159	3.613875	4.095833	3.621614	3.991190	3.42
2009	3.930571	4.204000	6.744773	3.171316	2.615850	4.180614	4.177159	3.42
2010	7.053500	6.318636	8.941955	5.901053	6.371158	5.875952	6.083364	6.54
2011	9.228825	9.972935	9.196690	9.055895	9.117625	10.759750	9.583364	8.44
2012	9.764575	11.949696	12.640900	9.162150	9.251400	11.203143	10.963738	9.47
2013	13.153614	14.557659	19.574833	13.184868	13.418976	14.948841	13.705100	13.28
2014	16.082072	16.366595	15.439341	17.716842	19.743167	16.995955	16.228667	18.13
2015	19.710643	25.923238	33.463114	18.787263	15.137375	23.935455	21.629977	18.76
2016	30.679714	38.241956	38.166262	26.531000	30.053079	37.073375	35.819523	28.61
2017	45.169289	48.571826	58.442075	41.787263	40.375250	50.424200	49.522205	42.71
2018	73.411023	94.892565	77.972157	72.118158	65.450547	89.232452	84.941190	77.01
2019	93.310119	89.680137	89.288643	81.346868	82.001619	98.294113	92.630875	86.12
2020	111.435262	162.462737	159.887500	103.308763	94.211880	152.692657	130.677273	93.61
2021	167.608714	165.645887	170.862479	163.193421	160.002421	180.800310	168.386271	153.41
2022	151.308700	137.439566	87.937142	153.783815	155.480849	117.037500	112.863095	154.28
2023	102.486666	NaN	NaN	99.214211	94.223500	NaN	NaN	96.54

In [37]:

```
plt.figure(figsize=(12, 8))
sns.heatmap(seasonal_trends.unstack())
plt.title('Seasonal Patterns in Stock Prices')
plt.xlabel('Month')
plt.ylabel('Year')
plt.show()
```



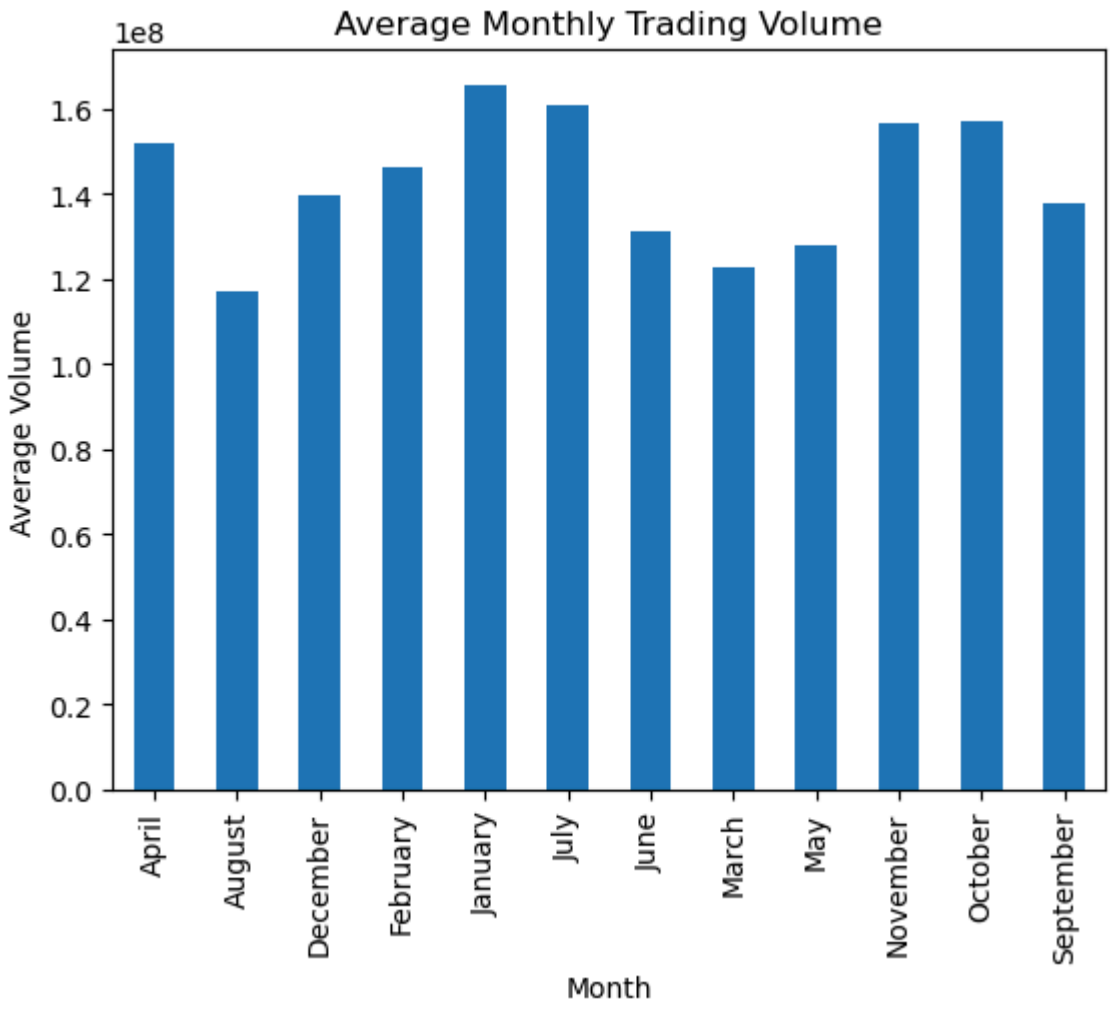
Trading Volume Variations Across Months

In [38]:
```
seasonal_trends = stocks.groupby(['month'])['Volume'].mean()
```

In [39]:
```
seasonal_trends
```

Out[39]:
```
month
April        1.520816e+08
August       1.168715e+08
December     1.398213e+08
February     1.460396e+08
January      1.655362e+08
July         1.606618e+08
June         1.309893e+08
March        1.225897e+08
May          1.277098e+08
November     1.563787e+08
October      1.572784e+08
September    1.377840e+08
Name: Volume, dtype: float64
```

In [40]:
```
seasonal_trends.plot(kind='bar')
plt.title('Average Monthly Trading Volume')
plt.xlabel('Month')
plt.ylabel('Average Volume')
plt.show()
```



Short-term and Long-term Moving Averages

In [41]:
```
stocks
```

Out[41]:

	Date	Open	High	Low	Close	Adj Close	Volume	Daily_Returr
0	1997-05-15	0.121875	0.125000	0.096354	0.097917	0.097917	1443120000	NaN
1	1997-05-16	0.098438	0.098958	0.085417	0.086458	0.086458	294000000	-0.11702
2	1997-05-19	0.088021	0.088542	0.081250	0.085417	0.085417	122136000	-0.01204
3	1997-05-20	0.086458	0.087500	0.081771	0.081771	0.081771	109344000	-0.04268
4	1997-05-21	0.081771	0.082292	0.068750	0.071354	0.071354	377064000	-0.12739
...	...	...	...	...	...	...	...	...
6511	2023-03-30	101.550003	103.040001	101.010002	102.000000	102.000000	53633400	0.01745
6512	2023-03-31	102.160004	103.489998	101.949997	103.290001	103.290001	56704300	0.01264
6513	2023-04-03	102.300003	103.290001	101.430000	102.410004	102.410004	41135700	-0.00852
6514	2023-04-04	102.750000	104.199997	102.110001	103.949997	103.949997	48662500	0.01503
6515	2023-04-05	103.910004	103.910004	100.750000	101.099998	101.099998	45103000	-0.02741

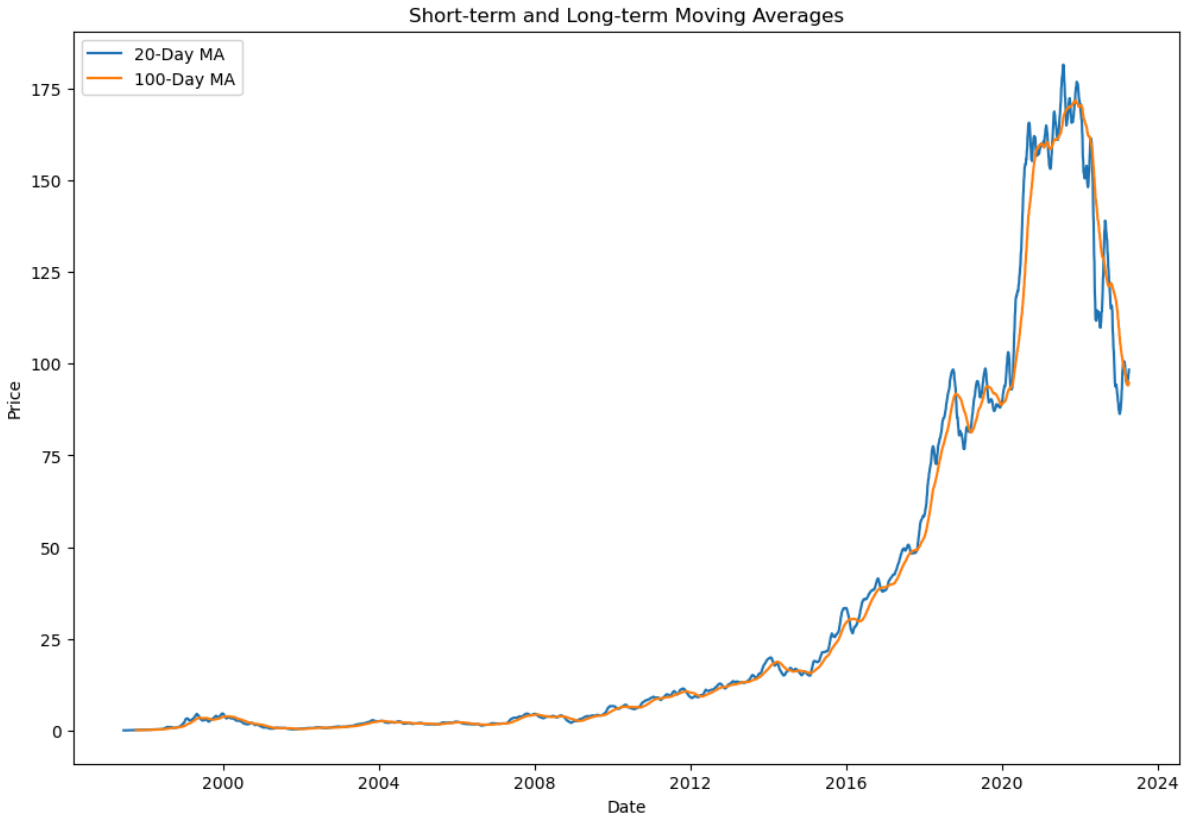
6516 rows × 12 columns

In [42]:
```
stocks['short_MA']=stocks['Close'].rolling(20).mean()
```

In [43]:
```
stocks['long_MA']=stocks['Close'].rolling(100).mean()
```

In [44]:
```
plt.figure(figsize=(12, 8))

plt.plot(stocks['Date'], stocks['short_MA'], label='20-Day MA')
plt.plot(stocks['Date'], stocks['long_MA'], label='100-Day MA')
plt.title('Short-term and Long-term Moving Averages')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()
```



Specific Days or Months with High Trading Volumes

In [51]:
```
high_volume_days = stocks[stocks['Volume'] > stocks['Volume'].quantile(0.95)]
high_volume_days[['day','month']]
```

Out[51]:

	day	month
0	5	May
4	5	May
198	2	February
204	3	March
205	3	March
...	...	...
3703	2	February
3763	4	April
4390	10	October
4456	1	January
4577	7	July

326 rows × 2 columns

Outlier detection

In [60]:
```
price_outliers = stocks[(np.abs(stocks['Close'] - stocks['Close'].mean()) > (3 * st
volume_outliers = stocks[(np.abs(stocks['Volume'] - stocks['Volume'].mean()) > (3 *
```

In [61]:
```
price_outliers
```

Out[61]:

	Date	Open	High	Low	Close	Adj Close	Volume	Daily_Return
5864	2020-09-02	177.350006	177.612503	174.334503	176.572495	176.572495	78630000	0.009239
6074	2021-07-06	176.505493	184.274002	176.449997	183.787003	183.787003	134896000	0.046927
6075	2021-07-07	185.869003	186.710007	183.945496	184.828995	184.828995	106562000	0.005670
6076	2021-07-08	182.177994	187.999496	181.056000	186.570496	186.570496	103612000	0.009422
6077	2021-07-09	186.126007	187.399994	184.669998	185.966995	185.966995	74964000	-0.003235
6078	2021-07-12	187.199997	187.864502	184.839493	185.927505	185.927505	51432000	-0.000212
6079	2021-07-13	185.104996	188.654007	183.565994	183.867996	183.867996	76918000	-0.011077
6080	2021-07-14	185.442505	185.882996	183.041504	184.084000	184.084000	65932000	0.001175
6081	2021-07-15	184.710007	184.770004	181.046005	181.559998	181.559998	63706000	-0.013711
6082	2021-07-16	181.665497	182.302994	178.522995	178.681503	178.681503	80874000	-0.015854
6083	2021-07-19	176.628998	177.510498	174.957993	177.479507	177.479507	75692000	-0.006727
6084	2021-07-20	178.365997	179.600006	175.899994	178.659500	178.659500	65114000	0.006649
6085	2021-07-21	178.819000	179.322495	177.182007	179.259995	179.259995	46380000	0.003361
6086	2021-07-22	179.361496	182.001007	179.113495	181.901505	181.901505	65308000	0.014736
6087	2021-07-23	182.000000	183.305496	181.102005	182.832001	182.832001	48726000	0.005115
6088	2021-07-26	183.658493	185.604004	182.362503	184.990997	184.990997	58002000	0.011809
6089	2021-07-27	184.925003	184.925003	179.307495	181.319504	181.319504	82638000	-0.019847
6090	2021-07-28	181.688995	182.921005	180.050003	181.516006	181.516006	59988000	0.001084
6091	2021-07-29	181.387497	181.897507	179.000504	179.996002	179.996002	110400000	-0.008374
6119	2021-09-08	175.582504	177.281494	174.783493	176.274994	176.274994	61068000	0.004619
6161	2021-11-05	173.850006	178.312500	173.848999	175.949493	175.949493	99940000	0.012076
6163	2021-11-09	175.762497	179.688507	175.071503	178.811493	178.811493	85898000	0.025007
6166	2021-11-12	174.250000	177.036499	172.352493	176.257507	176.257507	53788000	0.015162

	Date	Open	High	Low	Close	Adj Close	Volume	Daily_Return
6167	2021-11-15	176.850006	179.694000	176.290497	177.283997	177.283997	58594000	0.005824
6168	2021-11-16	176.949997	178.824997	176.257507	177.035004	177.035004	44342000	-0.001404
6169	2021-11-17	178.235992	179.362503	177.267502	177.449997	177.449997	51206000	0.002344
6170	2021-11-18	178.317505	185.210007	178.050003	184.802994	184.802994	114070000	0.041437
6171	2021-11-19	185.634506	188.107498	183.785995	183.828506	183.828506	98734000	-0.005273
6172	2021-11-22	183.819000	185.673004	178.375000	178.628494	178.628494	96844000	-0.028287
6173	2021-11-23	179.251999	181.052505	176.385498	179.001999	179.001999	73804000	0.002091
6174	2021-11-24	178.133499	180.682007	176.842499	179.020493	179.020493	46560000	0.000103
6176	2021-11-29	177.382004	179.800003	176.574997	178.078506	178.078506	65312000	0.016267
6182	2021-12-07	174.600006	177.499496	173.334503	176.164505	176.164505	66410000	0.027987
6183	2021-12-08	176.150497	177.179993	174.750504	176.158005	176.158005	45254000	-0.000037

In [62]:
```
volume_outliers
```

Out[62]:

	Date	Open	High	Low	Close	Adj Close	Volume	Daily_Return	year	
0	1997-05-15	0.121875	0.125000	0.096354	0.097917	0.097917	1443120000	NaN	1997	
239	1998-04-28	0.386979	0.414583	0.371094	0.398438	0.398438	1391880000	0.155589	1998	
240	1998-04-29	0.405208	0.410938	0.390104	0.397917	0.397917	633144000	-0.001308	1998	
270	1998-06-11	0.460417	0.521875	0.458333	0.520833	0.520833	717024000	0.154733	1998	
272	1998-06-15	0.495833	0.570833	0.492708	0.547917	0.547917	674028000	0.077870	1998	
...	...	...	...	...	...	...	...	...	...	
3131	2009-10-23	5.552500	5.982500	5.531000	5.924500	5.924500	1166116000	0.267951	2009	O
3132	2009-10-26	5.960500	6.284000	5.924500	6.232000	6.232000	645424000	0.051903	2009	O
3197	2010-01-29	6.488500	6.592500	6.207000	6.270500	6.270500	589426000	-0.004919	2010	J
3198	2010-02-01	6.159000	6.243000	5.691000	5.943500	5.943500	755488000	-0.052149	2010	Fe
3318	2010-07-23	5.296500	5.964000	5.290000	5.943500	5.943500	848422000	-0.009994	2010	

140 rows × 14 columns

In [ ]:

AMAZON STOCKS ANALYSIS