1.Load the Dataset

#The first step is to load the dataset.

```
In [45]: import pandas as pd
```

```
In [46]: df=pd.read_csv(r'C:/Users/rushi/Downloads/aapl.csv')
```

2. Explore the Dataset

The next step is to explore the dataset. We can use the head() method to view the first few rows of the dataset.

```
In [47]: df.head()
```

Out[47]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2012-10-10	22.847857	23.035000	22.750000	22.889643	19.486397	510356000
1	2012-10-11	23.089287	23.114286	22.432142	22.432142	19.096912	546081200
2	2012-10-12	22.484285	22.692142	22.332144	22.489643	19.145868	460014800
3	2012-10-15	22.583929	22.683214	22.280357	22.670000	19.299404	432502000
4	2012-10-16	22.691786	23.225000	22.535713	23.206785	19.756378	549771600

```
In [48]: # View the data types and non-null values in the dataset
         df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1005 entries, 0 to 1004
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       1005 non-null   object
 1   Open       1005 non-null   float64
 2   High       1005 non-null   float64
 3   Low        1005 non-null   float64
 4   Close      1005 non-null   float64
 5   Adj Close  1005 non-null   float64
 6   Volume     1005 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 55.1+ KB
```

3. Data Cleaning

The next step is to clean the data.

```
In [49]: #drop nan rows
         df = df.dropna()
```

```
In [50]: # Convert the date column to datetime format
         df['Date'] = pd.to_datetime(df['Date'])
```

```
In [51]: # Check for duplicates
         print(df.duplicated().sum())
```

```
0
```

```
In [52]: # remove any duplicate rows
         df.drop_duplicates(keep=False, inplace=True)
```

```
In [53]: # Convert the date column to datetime format
         df['Date'] = pd.to_datetime(df['Date'])
```

```
In [55]: # Rename columns
         df.rename(columns={'open': 'Open', 'high': 'High', 'low': 'Low', 'close': 'C
lose', 'volume': 'Volume'},inplace=True)
```

```
In [56]: # Remove irrelevant columns
         df.drop(['High', 'Low', 'Volume'], axis=1, inplace=True)
```

```
In [17]: # Handle outliers
         q1 = df['Close'].quantile(0.25)
q3 = df['Close'].quantile(0.75)
iqr = q3 - q1
upper_bound = q3 + 1.5 * iqr
df = df[df['Close'] <= upper_bound]
```

```
In [18]: #standardize data
         from sklearn.preprocessing import StandardScaler
```

the next step in EDA

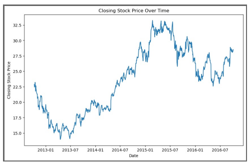
The next step is to visualize the data. We can use various types of plots to visualize the patterns and relationships in the data. Here, we will use the matplotlib and seaborn libraries to create plots.

```
In [57]: #import libraries first
         import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

Line Plot: We can create a line plot of the closing stock prices over time using the matplotlib library.

```
In [58]: # Line chart of closing stock price over time
         plt.figure(figsize=(10, 6))
sns.lineplot(x='Date', y='Close', data=df)
plt.title('Closing Stock Price Over Time')
plt.xlabel('Date')
```

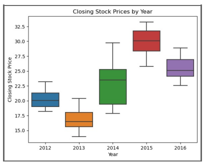
```
plt.ylabel('Closing Stock Price')
plt.show()
```



From the plot, we can see that the closing stock prices have increased over time, with some fluctuations

Box Plot: We can create a box plot of the closing stock prices by year using the seaborn library.

```
In [59]: df['year'] = df['Date'].dt.year
         sns.boxplot(x='year', y='Close', data=df)
plt.title('Closing Stock Prices by Year')
plt.xlabel('Year')
plt.ylabel('Closing Stock Price')
plt.show()
```

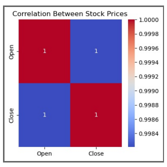


From the plot, we can see that the closing stock prices have generally increased over the years, with some outliers.

Heatmap:

We can create a heatmap to visualize the correlation between the stock prices using the seaborn library.

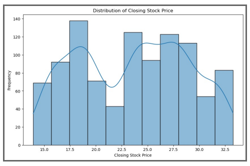
```
In [68]: # Create a heatmap of the correlation between stock prices
         corr = df[['Open','Close']].corr()
plt.figure(figsize=(4,4))
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Between Stock Prices')
plt.show()
```



We can start by visualizing the distribution of the target variable, which in this case is the closing stock price. We can use a histogram to visualize the distribution.

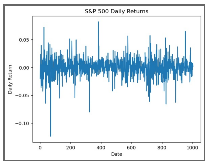
```
In [69]: plt.figure(figsize=(10, 6))
         sns.histplot(df['Close'], kde=True)
```

```
plt.title('Distribution of Closing Stock Price')
plt.xlabel('Closing Stock Price')
plt.ylabel('Frequency')
plt.show()
```



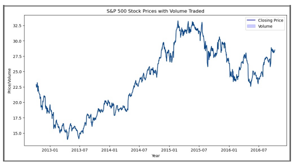
```
In [113]: daily_returns = df['Close'].pct_change()
```

```
In [114]: # Create a line chart of the daily returns
          plt.plot(daily_returns.index, daily_returns.values)
plt.title('S&P 500 Daily Returns')
plt.xlabel('Date')
plt.ylabel('Daily Return')
plt.show()
#This will create a line chart showing the daily returns over time.
```



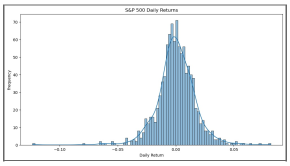
We can use a combination chart to visualize the stock prices with the volume traded.

```
In [121]: # Create a combination plot of stock prices and volume traded
          plt.figure(figsize=(12,6))
sns.lineplot(x='Date', y='Close', data=df, color='b')
sns.lineplot(x='Date', y='Close', data=df, color='g', alpha=0.5)
plt.title('S&P 500 Stock Prices with Volume Traded')
plt.xlabel('Year')
plt.ylabel('Price/Volume')
plt.legend(['Closing Price', 'Volume'])
plt.show()
```



We can also use a histogram to visualize the daily returns.

```
In [122]: # Create a histogram of the daily returns
          plt.figure(figsize=(12,6))
sns.histplot(df['Close'].pct_change().dropna(), bins=100, kde=True)
plt.title('S&P 500 Daily Returns')
plt.xlabel('Daily Return')
plt.ylabel('Frequency')
plt.show()
```


S&P 500 Daily Returns
Frequency
Daily Return
70
60
50
40
30
20
10
0
−0.10
−0.05
0.00
0.05

In [ ]: