python

MIT Academy of Engineering

EDS PROJECT

Presented by:

645 – Pratik Pachpute

Guide by:

Madhavi Nimkar

DETAIL OF DATASET

648 – Chaitanya Parab

657 – Ishwar Survase

651 – Vishnu Pawar

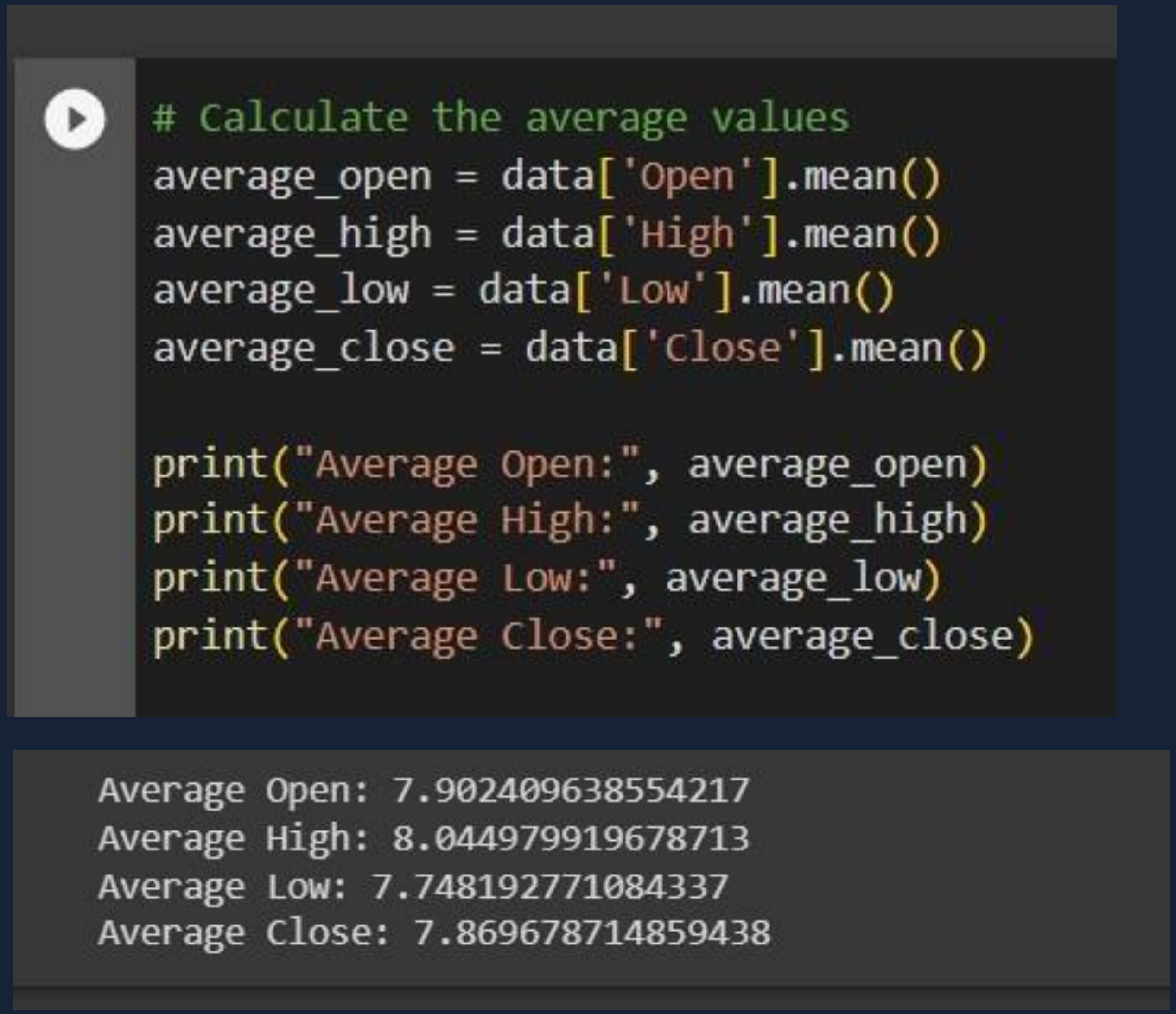
- Name: Stock Price
- Dataset:

  https://drive.google.com/file/d/1iBGkFAuRax1L_xHJgnqpq7kXy_BnfuE/view?usp=sharing
- Colab :

https://colab.research.google.com/drive/18hH5nRYSkEJD9GhpIrAIFDrek2L5OtRa?usp=sharing

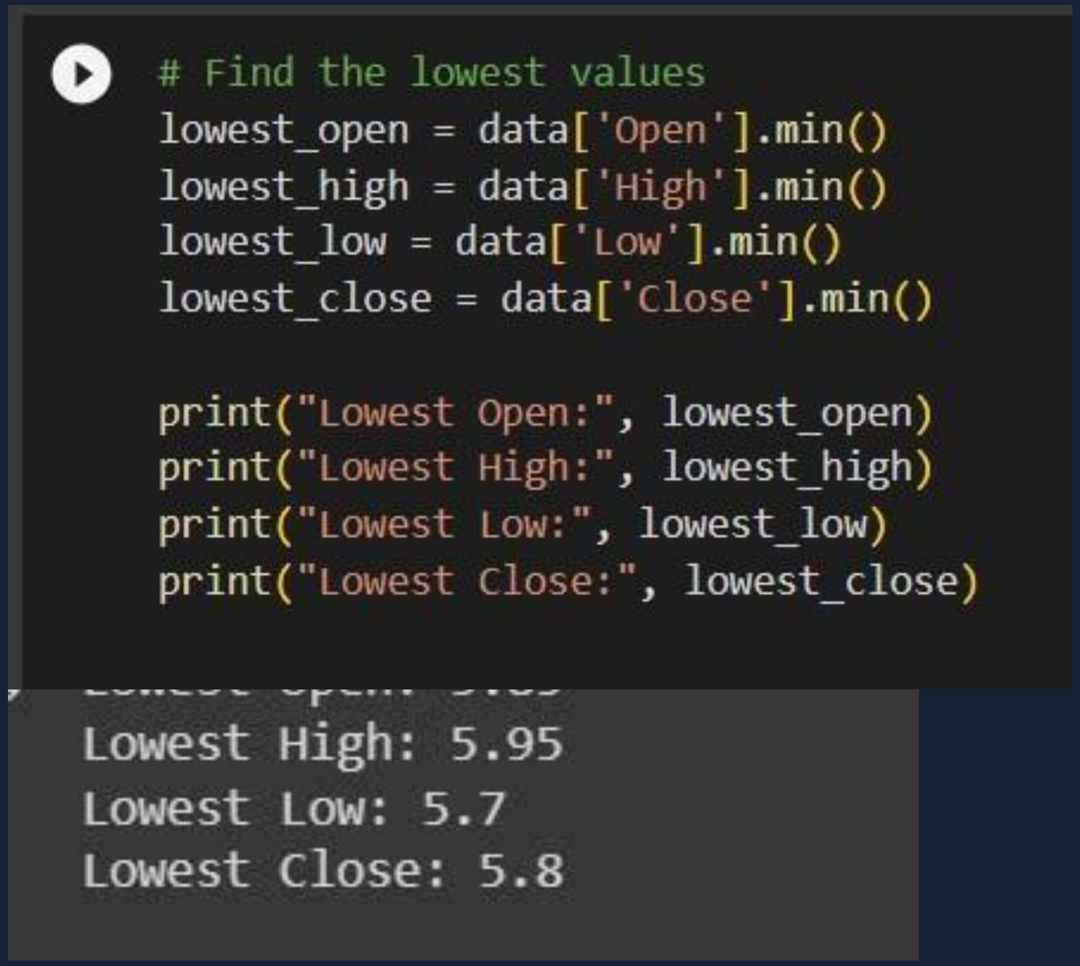
DATA MANIPULATION

Data manipulation is a fundamental process in data analysis that involves transforming and preparing raw data to make it suitable for further exploration and analysis. It encompasses a range of operations aimed at ensuring data quality, consistency, and usability. Missing values can be imputed or removed, while outliers can be addressed through various methods such as transformation



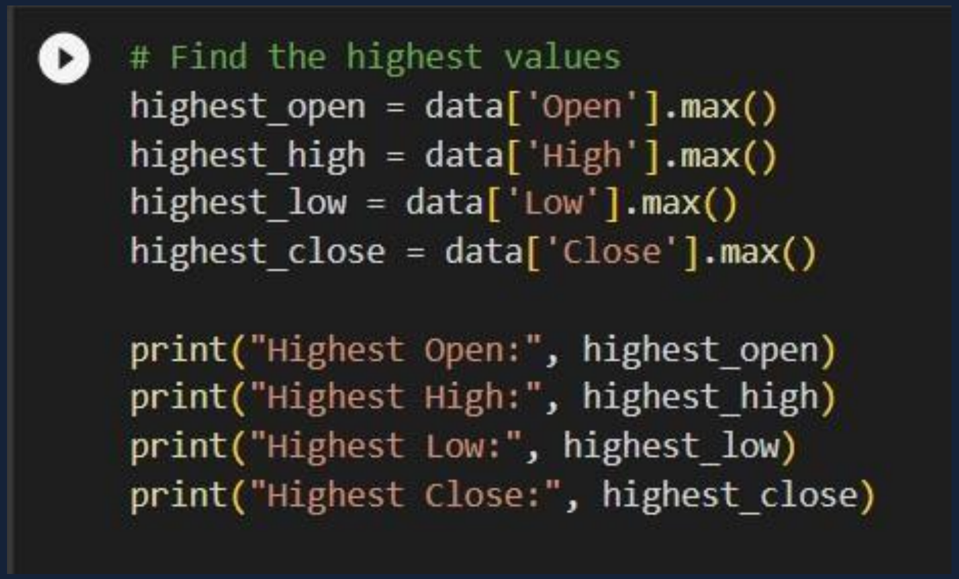
```python
# Calculate the average values
average_open = data['Open'].mean()
average_high = data['High'].mean()
average_low = data['Low'].mean()
average_close = data['Close'].mean()

print("Average Open:", average_open)
print("Average High:", average_high)
print("Average Low:", average_low)
print("Average Close:", average_close)
```

```
Average Open: 7.902409638554217
Average High: 8.044979919678713
Average Low: 7.748192771084337
Average Close: 7.869678714859438
```

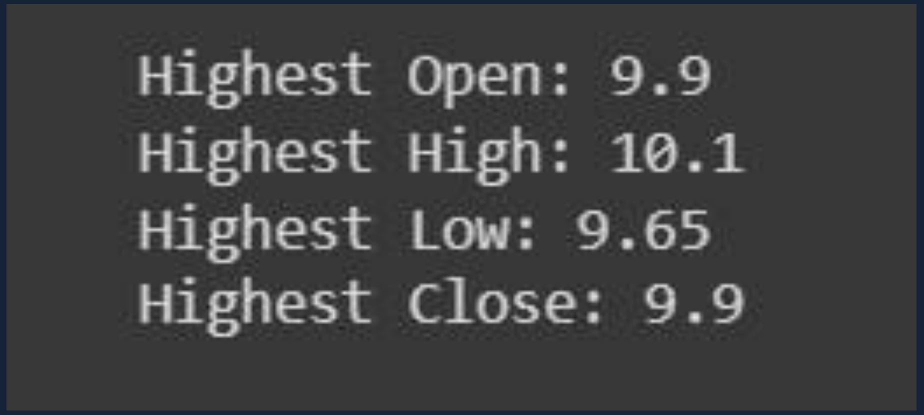


```python
# Find the lowest values
lowest_open = data['Open'].min()
lowest_high = data['High'].min()
lowest_low = data['Low'].min()
lowest_close = data['Close'].min()

print("Lowest Open:", lowest_open)
print("Lowest High:", lowest_high)
print("Lowest Low:", lowest_low)
print("Lowest Close:", lowest_close)
```

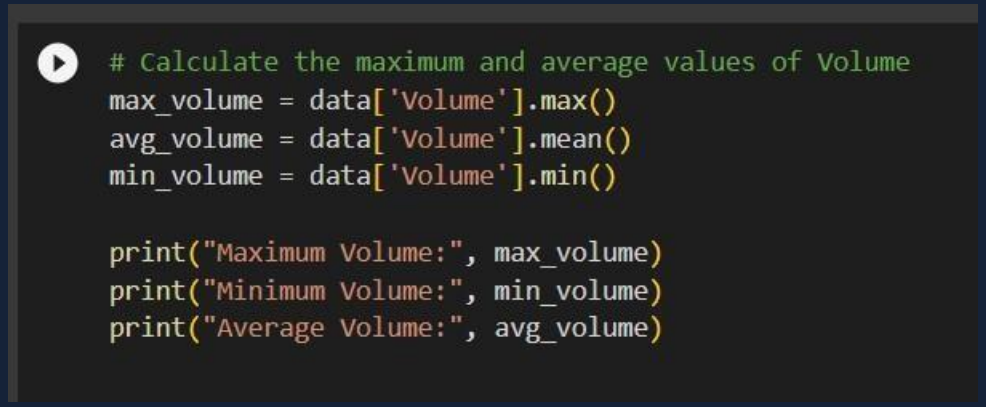
```
Lowest Open: 5.85
Lowest High: 5.95
Lowest Low: 5.7
Lowest Close: 5.8
```



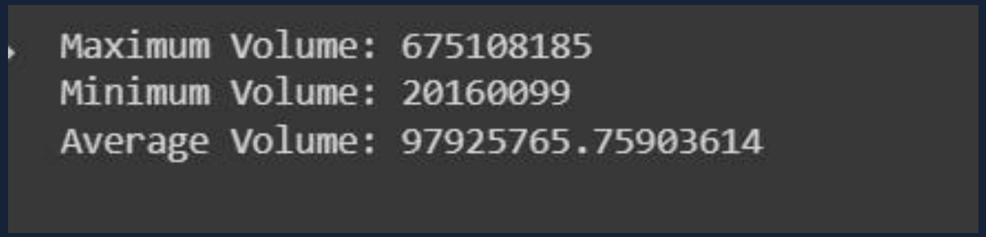
```
# Find the highest values
highest_open = data['Open'].max()
highest_high = data['High'].max()
highest_low = data['Low'].max()
highest_close = data['Close'].max()

print("Highest Open:", highest_open)
print("Highest High:", highest_high)
print("Highest Low:", highest_low)
print("Highest Close:", highest_close)
```



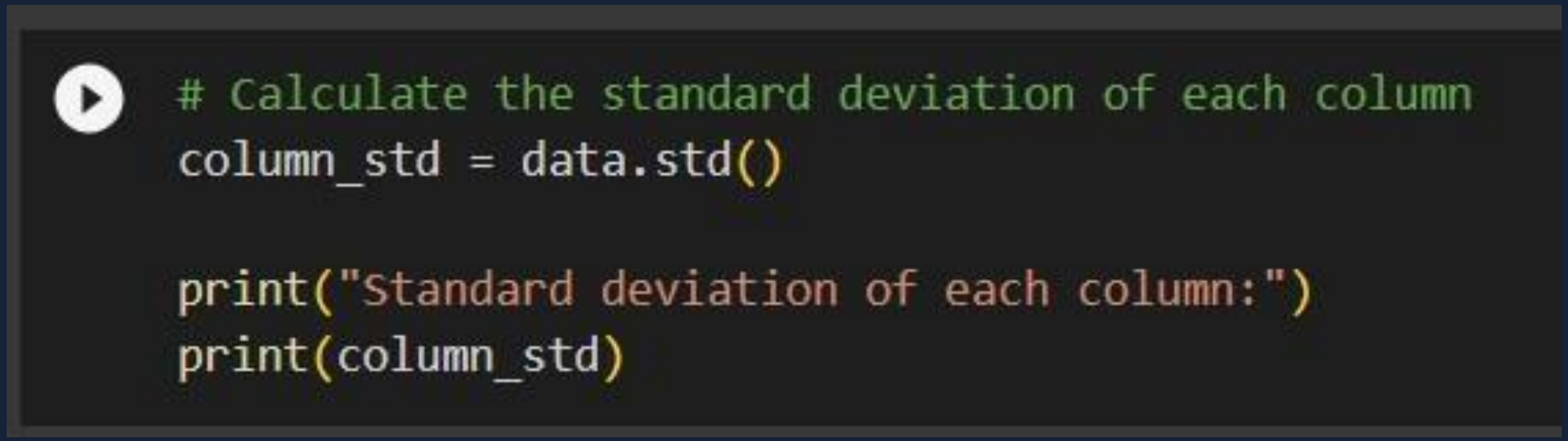
```
Highest Open: 9.9
Highest High: 10.1
Highest Low: 9.65
Highest Close: 9.9
```



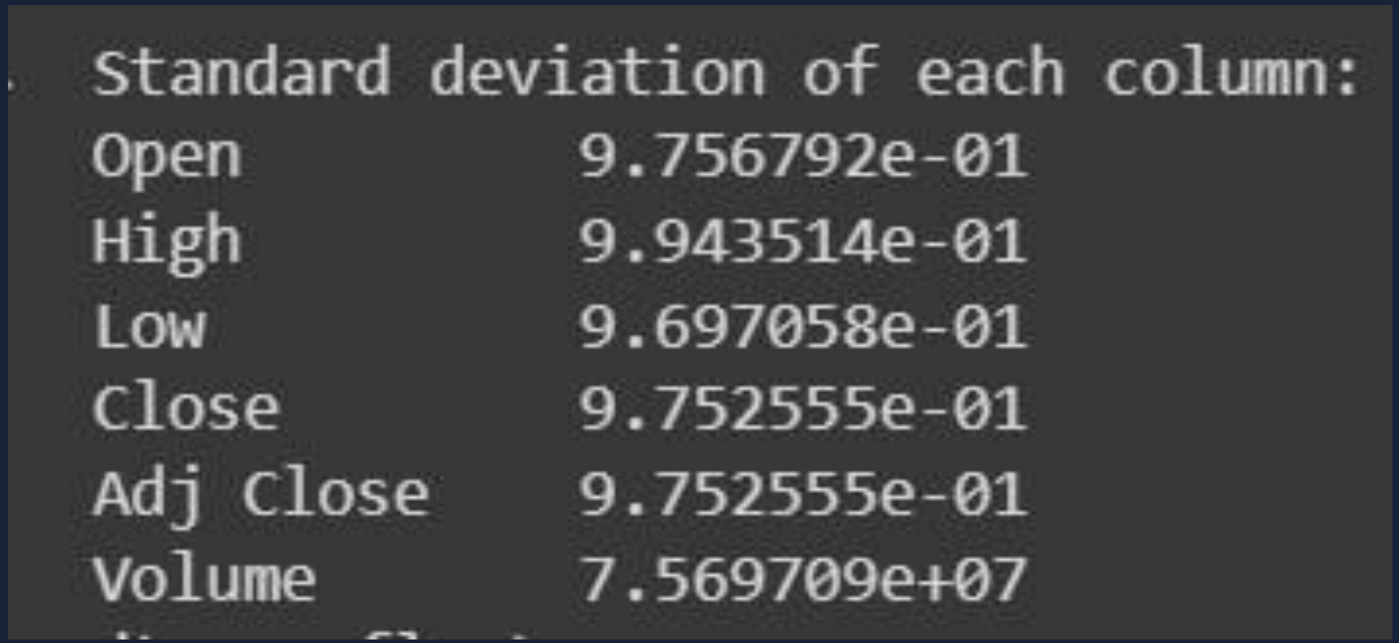
```python
# Calculate the maximum and average values of Volume
max_volume = data['Volume'].max()
avg_volume = data['Volume'].mean()
min_volume = data['Volume'].min()

print("Maximum Volume:", max_volume)
print("Minimum Volume:", min_volume)
print("Average Volume:", avg_volume)
```



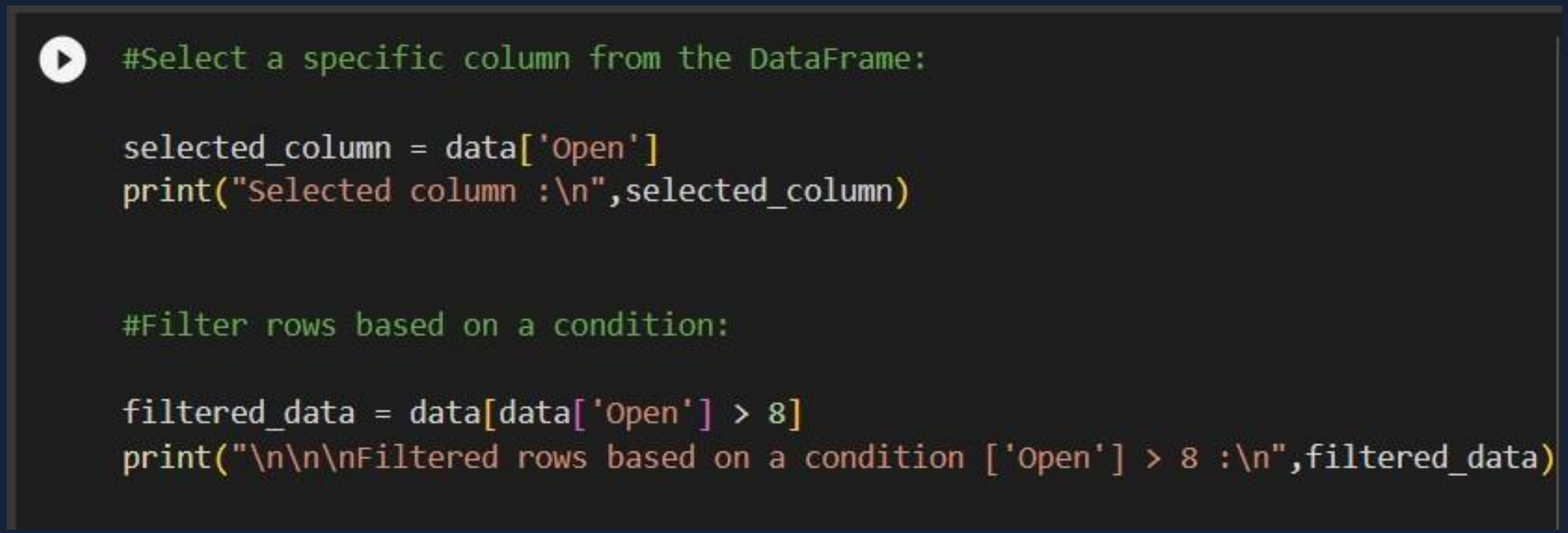
```
Maximum Volume: 675108185
Minimum Volume: 20160099
Average Volume: 97925765.75903614
```



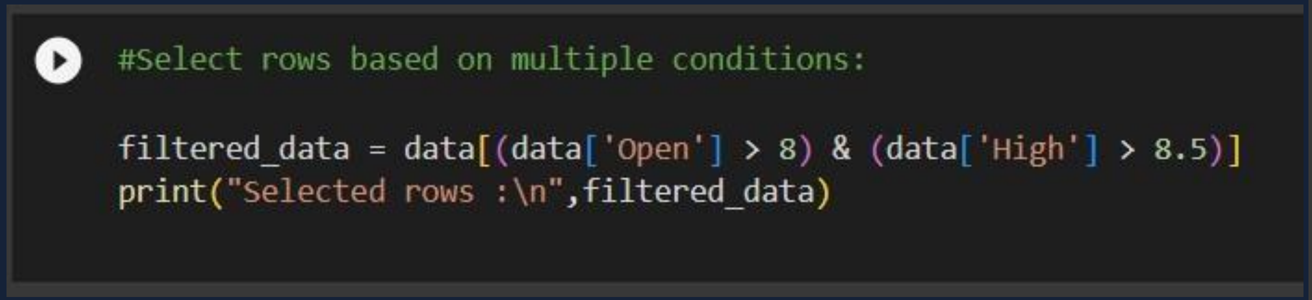
```python
# Calculate the standard deviation of each column
column_std = data.std()

print("Standard deviation of each column:")
print(column_std)
```



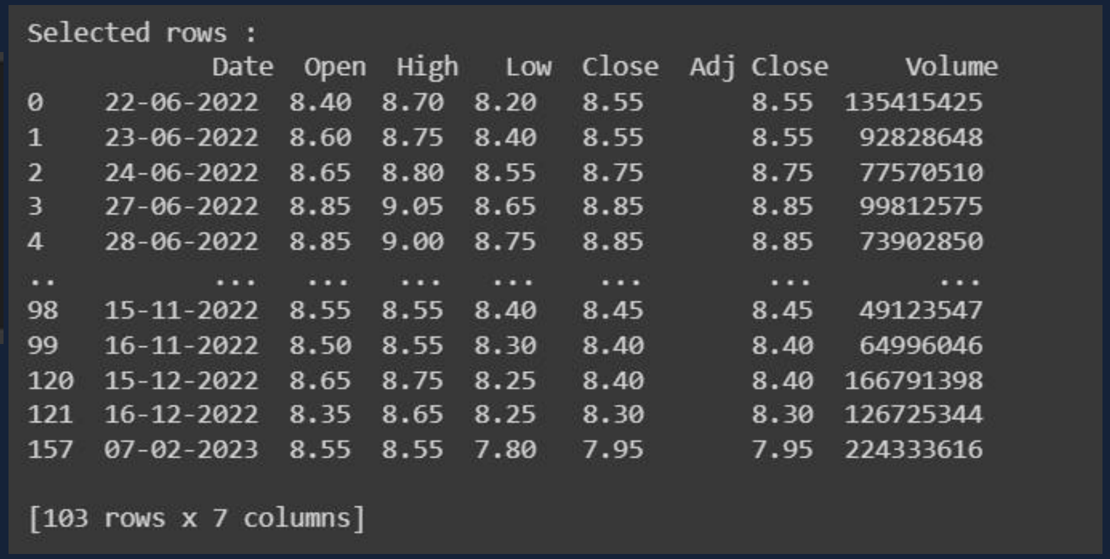
```
Standard deviation of each column:
Open          9.756792e-01
High          9.943514e-01
Low           9.697058e-01
Close         9.752555e-01
Adj Close     9.752555e-01
Volume        7.569709e+07
```



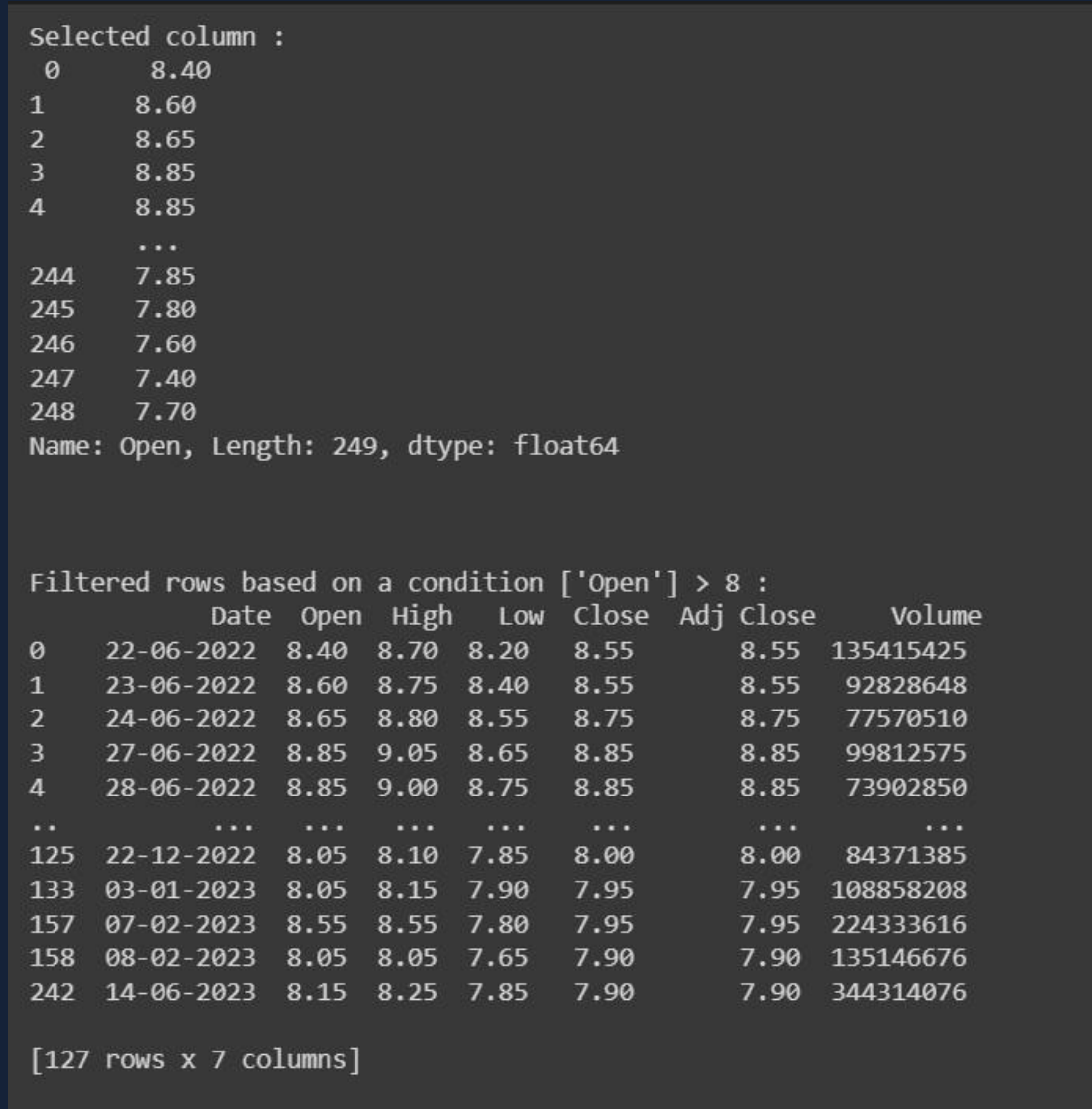
```python
#Select a specific column from the DataFrame:

selected_column = data['Open']
print("Selected column :\n",selected_column)


#Filter rows based on a condition:

filtered_data = data[data['Open'] > 8]
print("\n\n\nFiltered rows based on a condition ['Open'] > 8 :\n",filtered_data)
```



```
#Select rows based on multiple conditions:

filtered_data = data[(data['Open'] > 8) & (data['High'] > 8.5)]
print("Selected rows :\n",filtered_data)
```



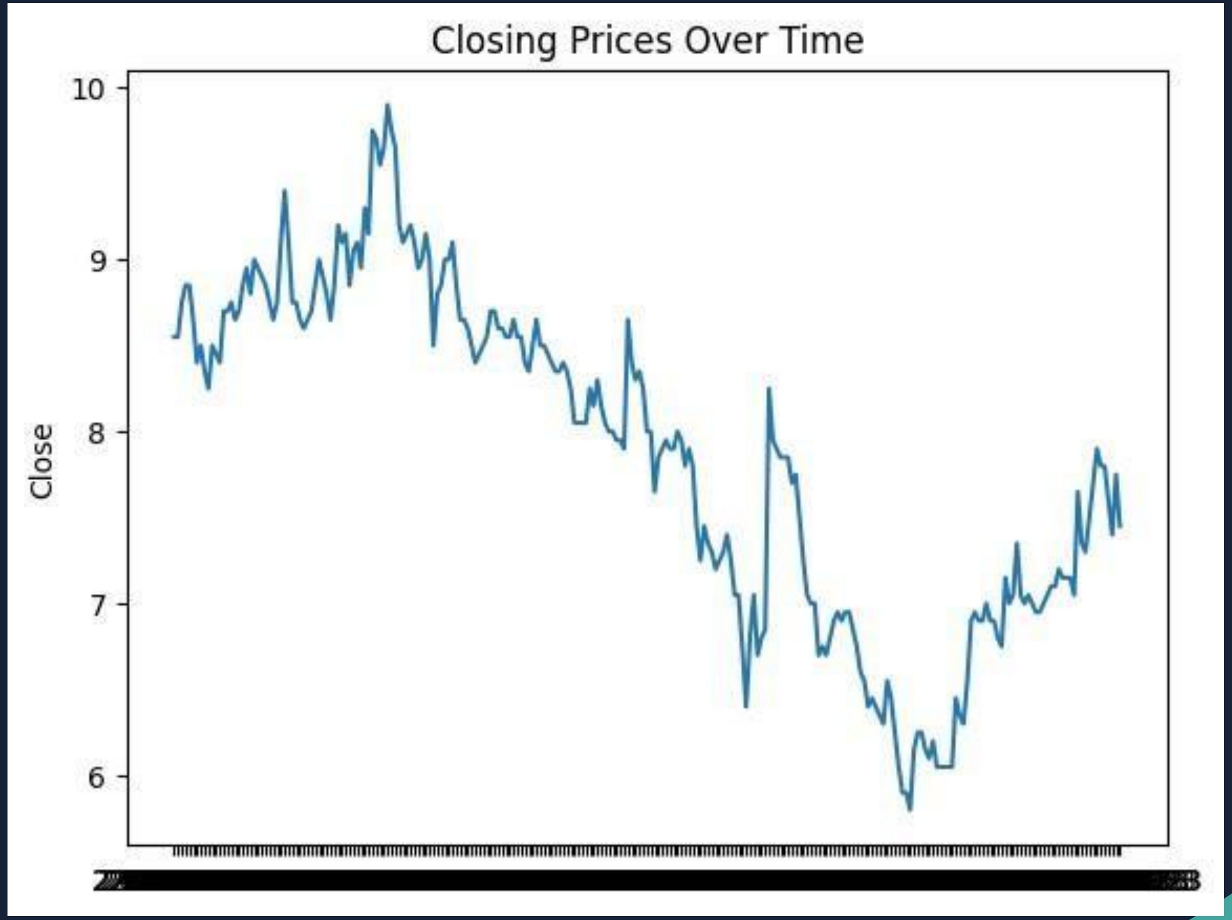
```
Selected rows :
            Date  Open  High   Low  Close  Adj Close     Volume
0    22-06-2022  8.40  8.70  8.20   8.55       8.55  135415425
1    23-06-2022  8.60  8.75  8.40   8.55       8.55   92828648
2    24-06-2022  8.65  8.80  8.55   8.75       8.75   77570510
3    27-06-2022  8.85  9.05  8.65   8.85       8.85   99812575
4    28-06-2022  8.85  9.00  8.75   8.85       8.85   73902850
..          ...   ...   ...   ...    ...        ...        ...
98   15-11-2022  8.55  8.55  8.40   8.45       8.45   49123547
99   16-11-2022  8.50  8.55  8.30   8.40       8.40   64996046
120  15-12-2022  8.65  8.75  8.25   8.40       8.40  166791398
121  16-12-2022  8.35  8.65  8.25   8.30       8.30  126725344
157  07-02-2023  8.55  8.55  7.80   7.95       7.95  224333616

[103 rows x 7 columns]
```

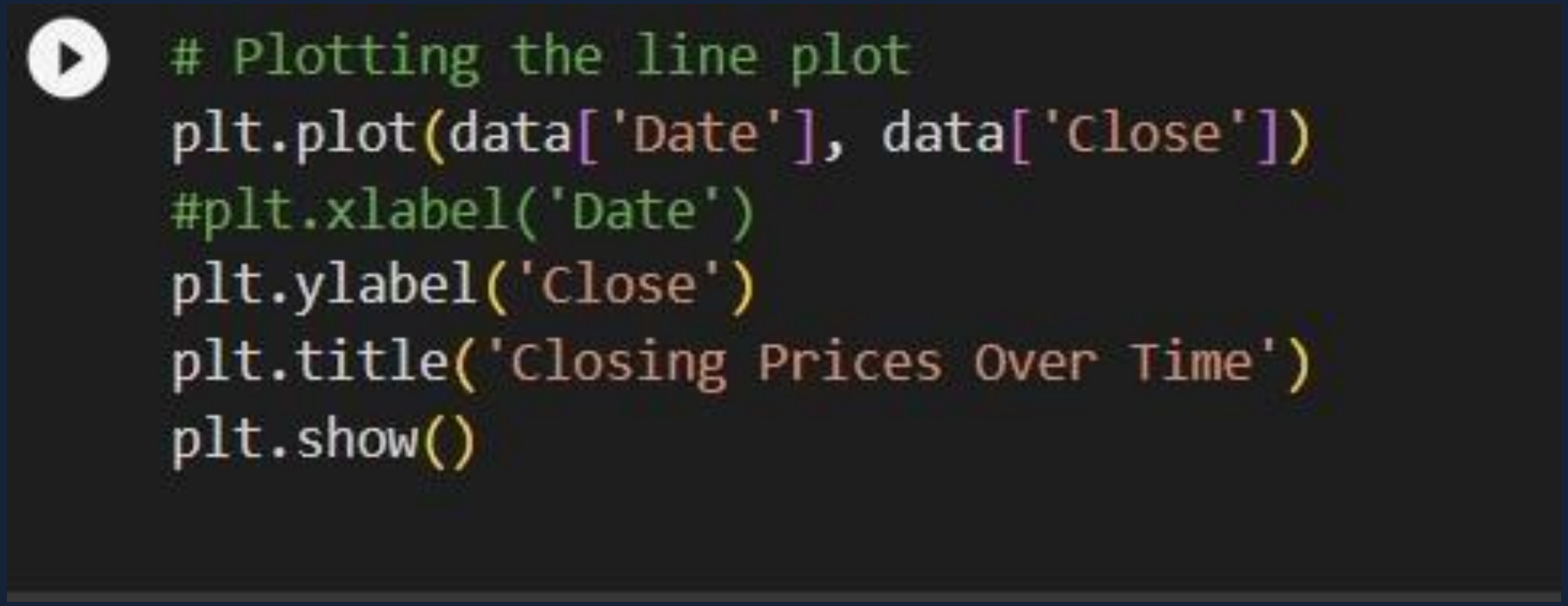


```
Selected column :
 0      8.40
1      8.60
2      8.65
3      8.85
4      8.85
       ...
244    7.85
245    7.80
246    7.60
247    7.40
248    7.70
Name: Open, Length: 249, dtype: float64



Filtered rows based on a condition ['Open'] > 8 :
            Date  Open  High   Low  Close  Adj Close     Volume
0    22-06-2022  8.40  8.70  8.20   8.55       8.55  135415425
1    23-06-2022  8.60  8.75  8.40   8.55       8.55   92828648
2    24-06-2022  8.65  8.80  8.55   8.75       8.75   77570510
3    27-06-2022  8.85  9.05  8.65   8.85       8.85   99812575
4    28-06-2022  8.85  9.00  8.75   8.85       8.85   73902850
..          ...   ...   ...   ...    ...        ...        ...
125  22-12-2022  8.05  8.10  7.85   8.00       8.00   84371385
133  03-01-2023  8.05  8.15  7.90   7.95       7.95  108858208
157  07-02-2023  8.55  8.55  7.80   7.95       7.95  224333616
158  08-02-2023  8.05  8.05  7.65   7.90       7.90  135146676
242  14-06-2023  8.15  8.25  7.85   7.90       7.90  344314076

[127 rows x 7 columns]
```

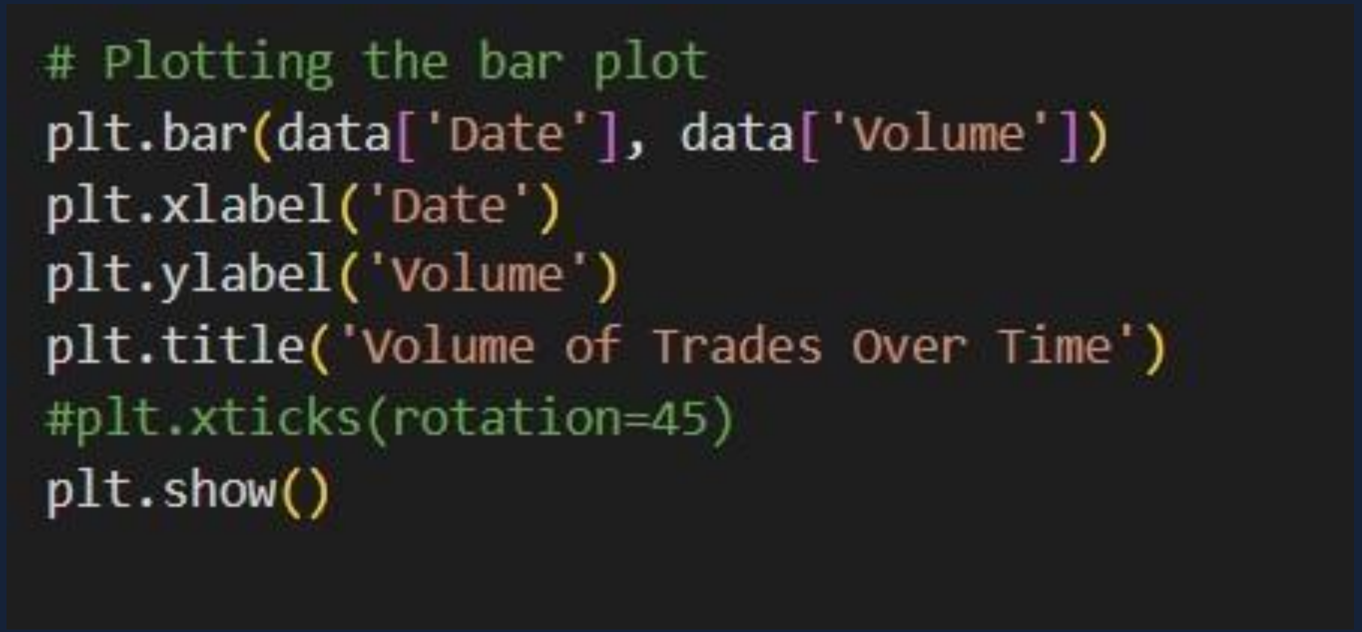
DATA VISUALIZATION

Data visualization is the process of representing data and information visually through charts, graphs, maps, and other graphical elements. It is a powerful technique that allows us to effectively communicate complex concepts, patterns, and trends in a visual format. Data visualization transforms complex data into visual representations that enhance understanding, reveal patterns, and support decision-making

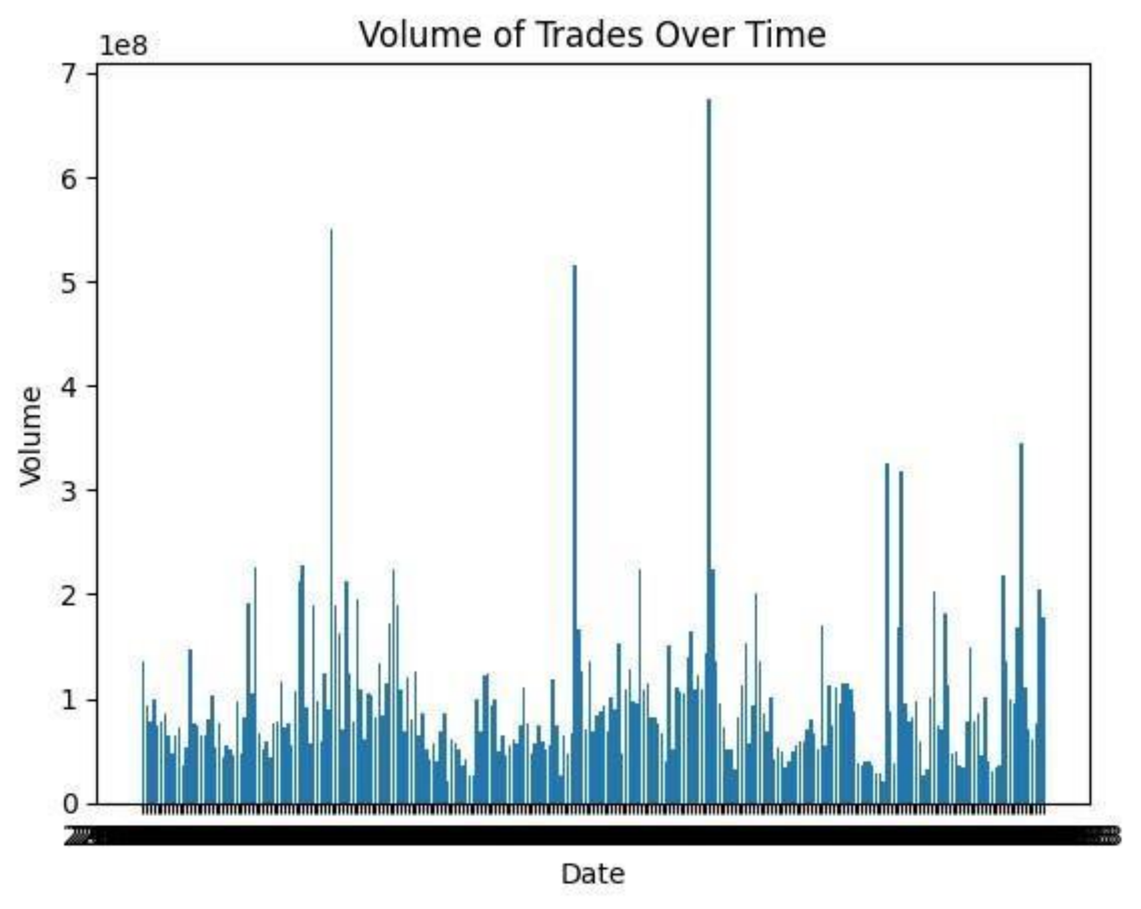


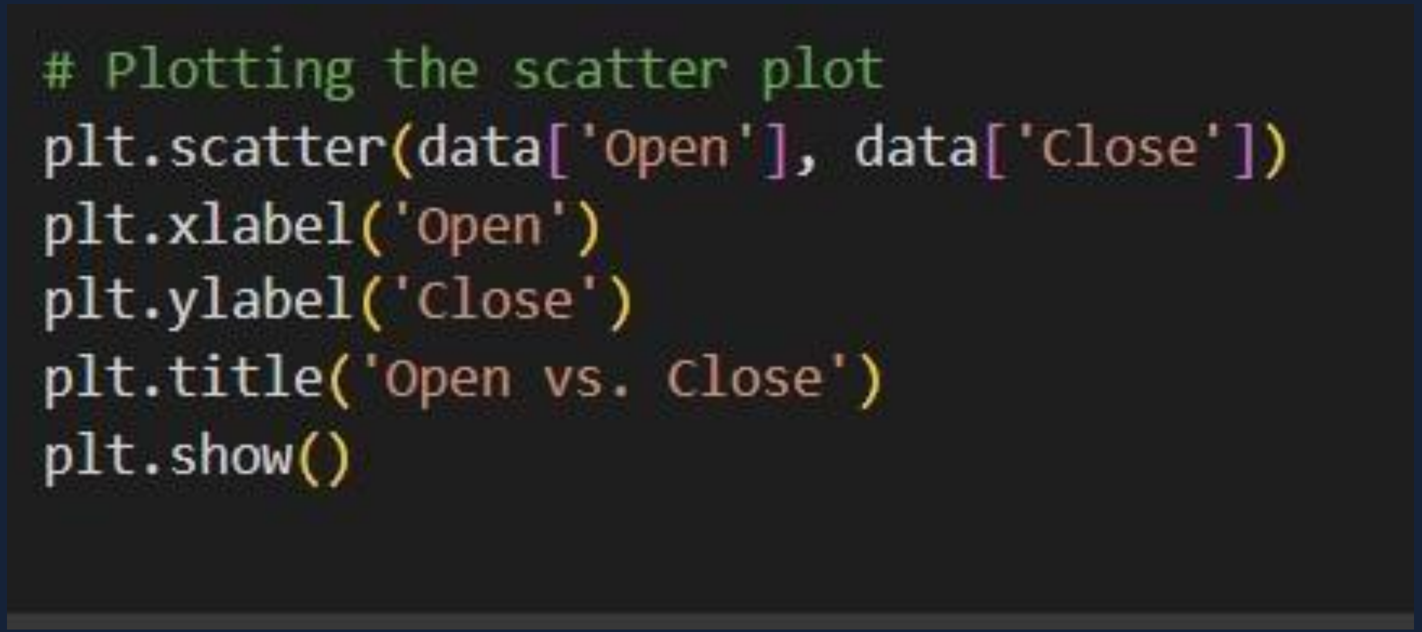


```
# Plotting the line plot
plt.plot(data['Date'], data['Close'])
#plt.xlabel('Date')
plt.ylabel('Close')
plt.title('Closing Prices Over Time')
plt.show()
```

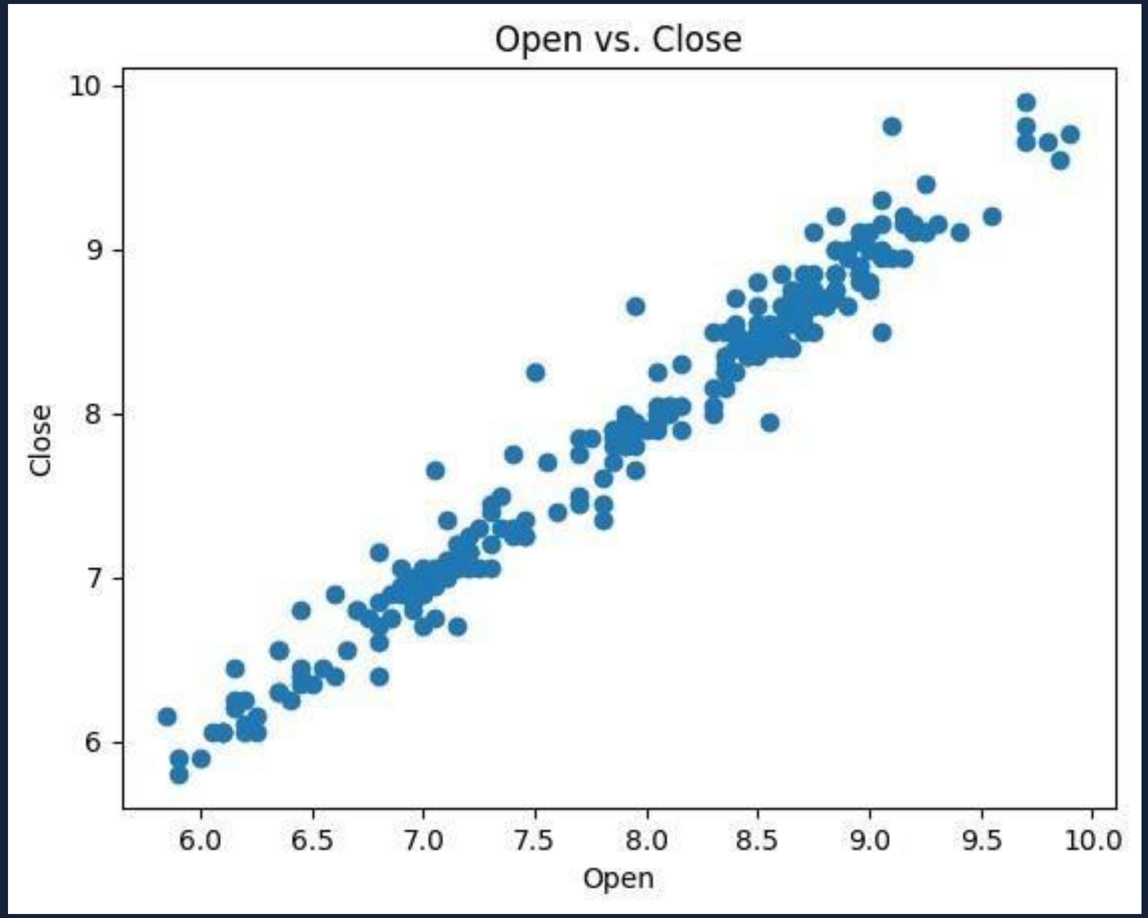


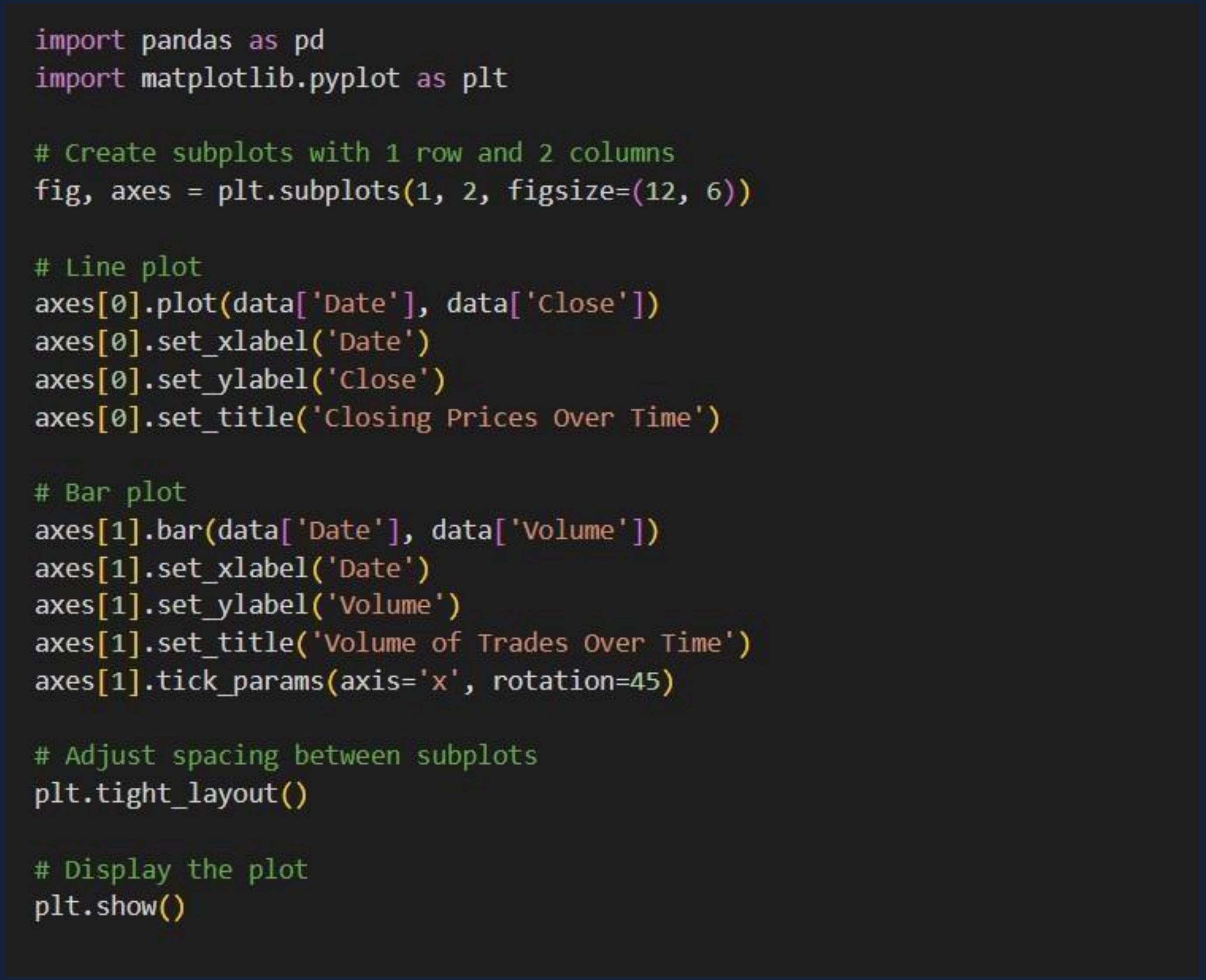
```python
# Plotting the bar plot
plt.bar(data['Date'], data['Volume'])
plt.xlabel('Date')
plt.ylabel('Volume')
plt.title('Volume of Trades Over Time')
#plt.xticks(rotation=45)
plt.show()
```



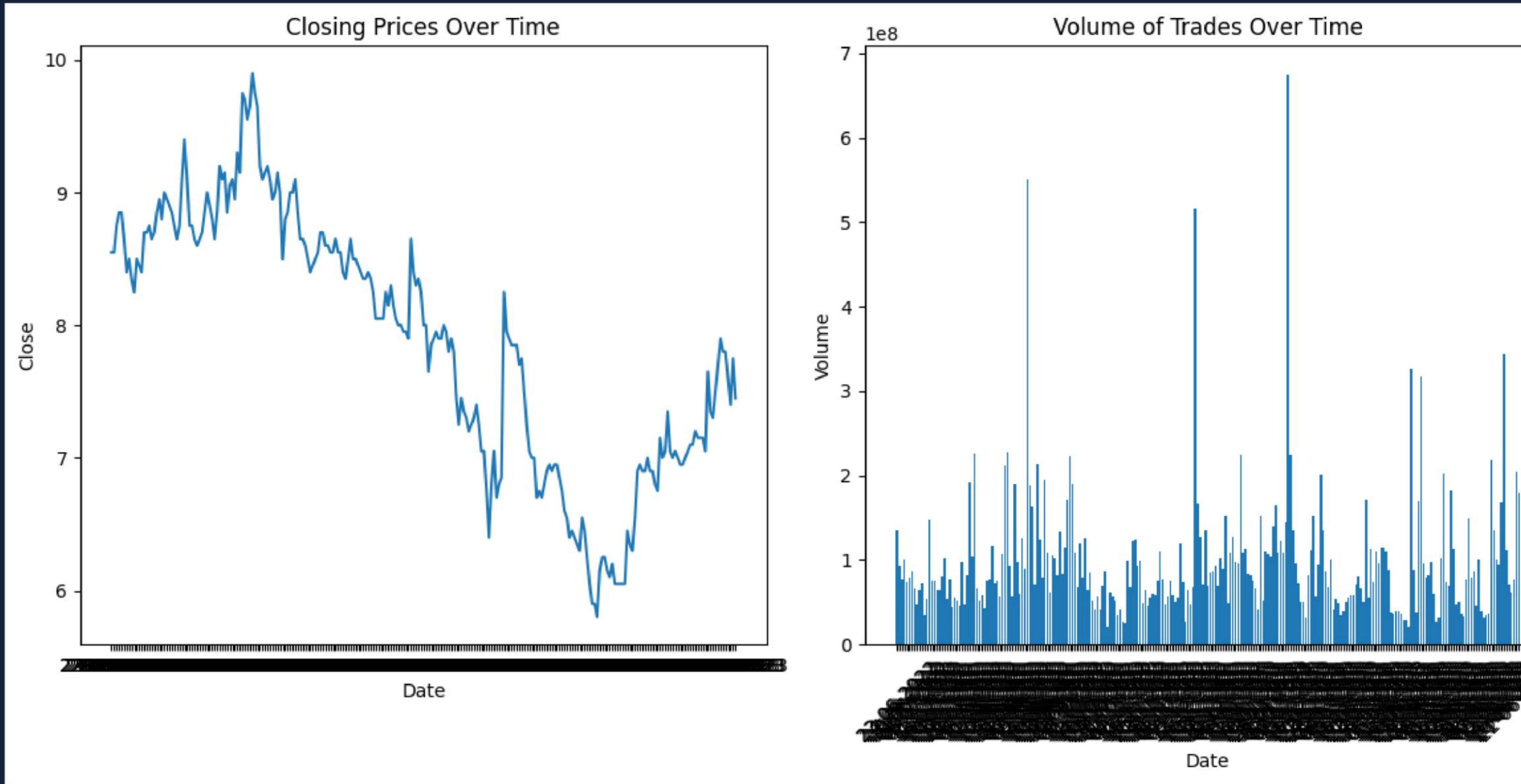


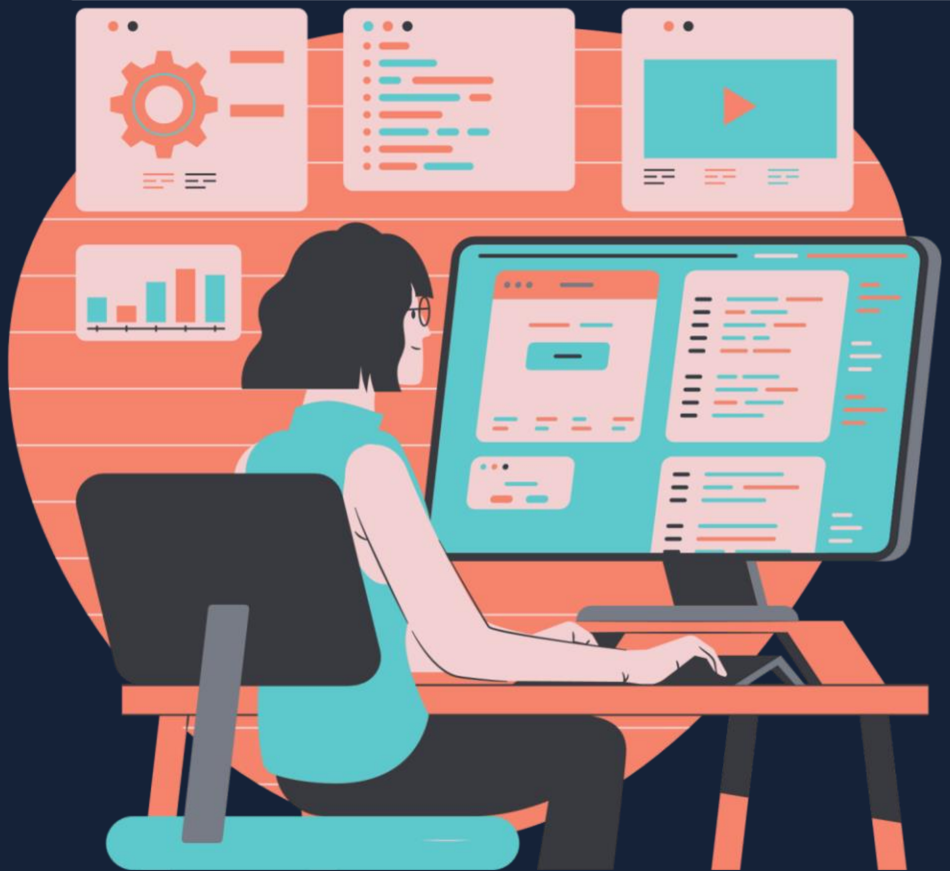
```python
# Plotting the scatter plot
plt.scatter(data['Open'], data['Close'])
plt.xlabel('Open')
plt.ylabel('Close')
plt.title('Open vs. Close')
plt.show()
```





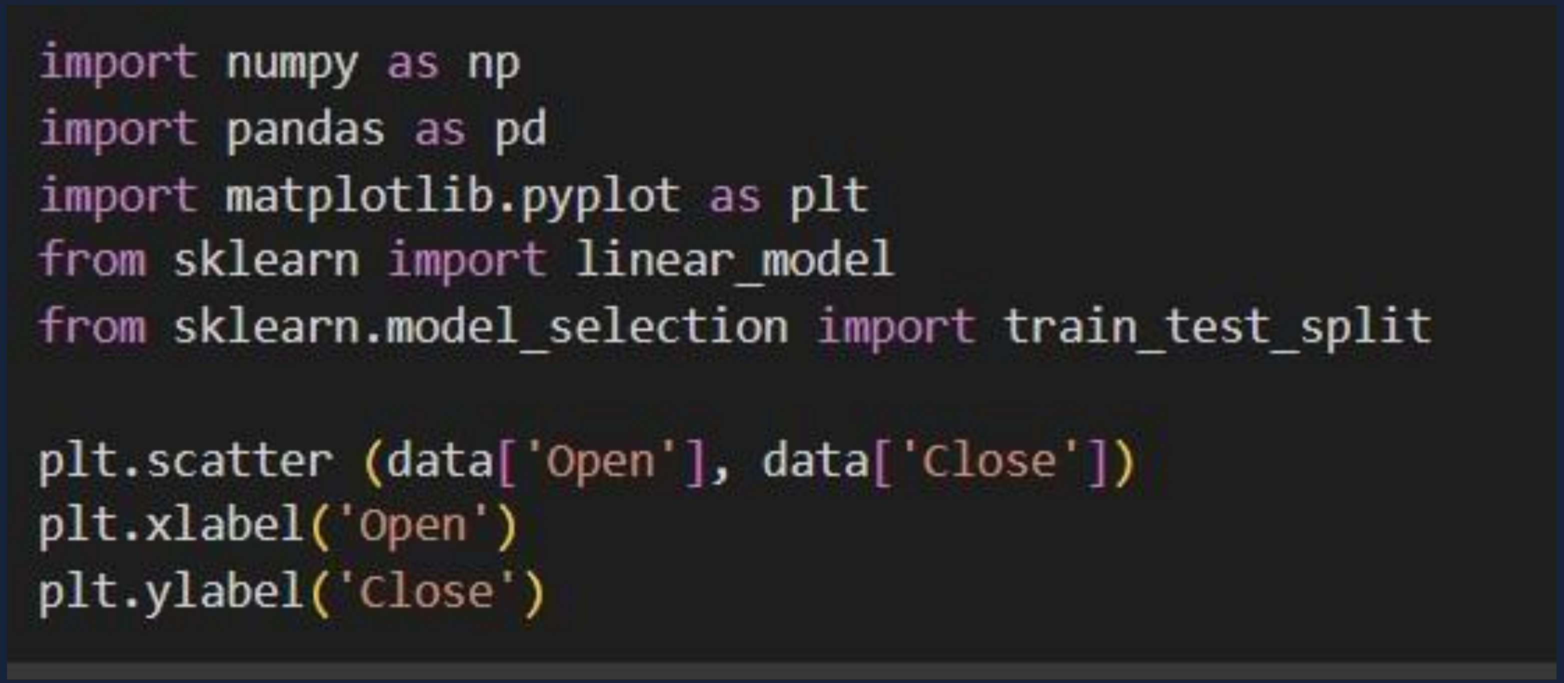
```python
import pandas as pd
import matplotlib.pyplot as plt

# Create subplots with 1 row and 2 columns
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

# Line plot
axes[0].plot(data['Date'], data['Close'])
axes[0].set_xlabel('Date')
axes[0].set_ylabel('Close')
axes[0].set_title('Closing Prices Over Time')

# Bar plot
axes[1].bar(data['Date'], data['Volume'])
axes[1].set_xlabel('Date')
axes[1].set_ylabel('Volume')
axes[1].set_title('Volume of Trades Over Time')
axes[1].tick_params(axis='x', rotation=45)

# Adjust spacing between subplots
plt.tight_layout()

# Display the plot
plt.show()
```


Closing Prices Over Time
Close
10
9
8
7
6
Date
Volume of Trades Over Time
1e8
7
6
5
4
3
2
1
0
Volume
Date

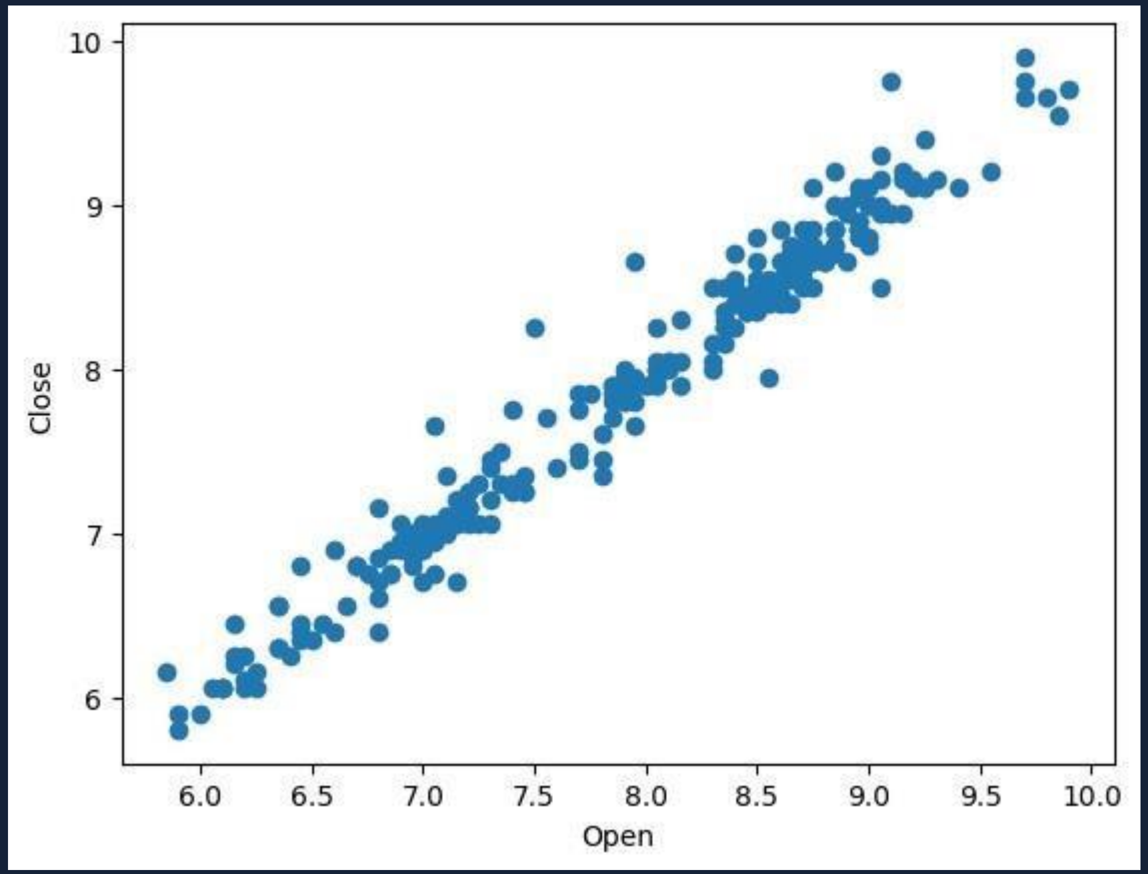


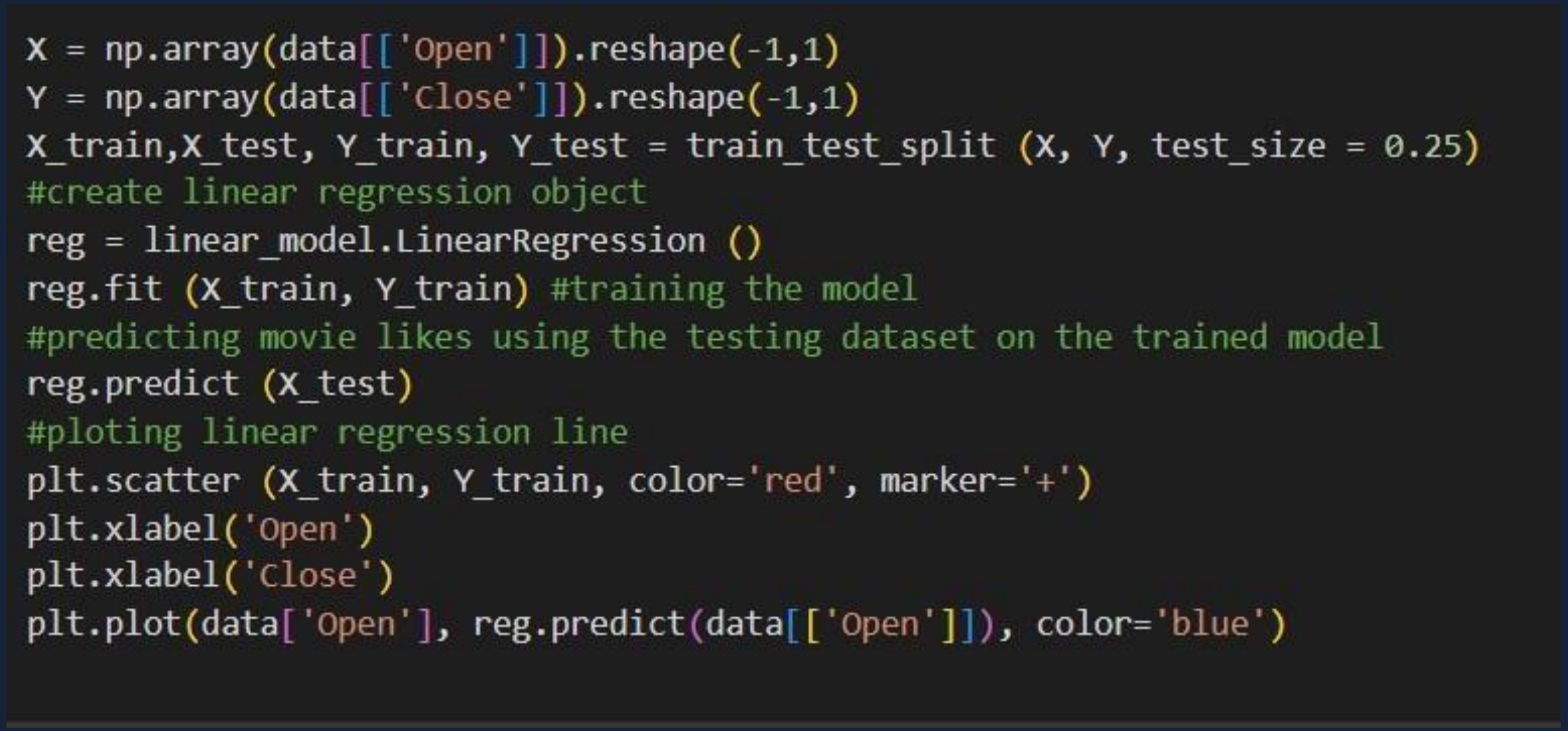
PREDICTIVE TECHNIQUE

(LINEAR RIGRESSION)

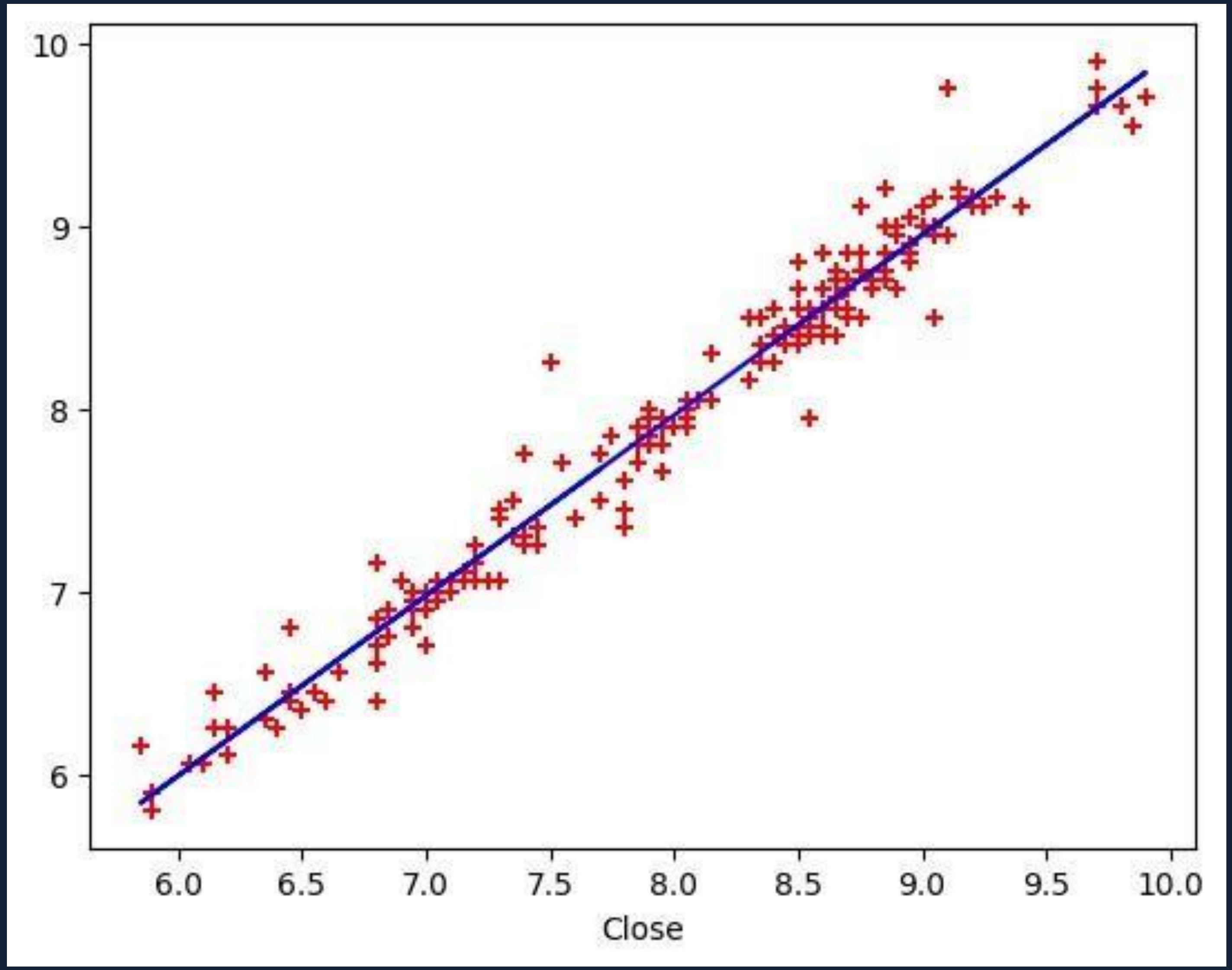


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.model_selection import train_test_split

plt.scatter (data['Open'], data['Close'])
plt.xlabel('Open')
plt.ylabel('Close')
```





```
X = np.array(data[['Open']]).reshape(-1,1)
Y = np.array(data[['Close']]).reshape(-1,1)
X_train,X_test, Y_train, Y_test = train_test_split (X, Y, test_size = 0.25)
#create linear regression object
reg = linear_model.LinearRegression ()
reg.fit (X_train, Y_train) #training the model
#predicting movie likes using the testing dataset on the trained model
reg.predict (X_test)
#ploting linear regression line
plt.scatter (X_train, Y_train, color='red', marker='+')
plt.xlabel('Open')
plt.xlabel('Close')
plt.plot(data['Open'], reg.predict(data[['Open']]), color='blue')
```



21:51

THANK YOU