Dataset

Loading the Dataset:

```
import pandas as pd

# Read the individual CSV files
btc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Bitcoin.csv")
eth_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Ethereum.csv")
xrp_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_XRP.csv")
ltc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Litecoin.csv")
usdc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_USDCoin.csv")

# Concatenate the datasets vertically
df = pd.concat([btc_df, eth_df, xrp_df, ltc_df, usdc_df])

# Save the merged dataset to a new CSV file
df.to_csv("cryptocurrency.csv", index=False)


```

EDA

Summarizing the Dataset:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn import tree
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from sklearn import metrics
import seaborn as sns
%matplotlib inline
```

```
import sys
!{sys.executable} -m pip install -U pandas-profiling
!jupyter nbextension enable --py widgetsnbextension
!pip install matplotlib
!pip install graphviz
```

```
Requirement already satisfied: jinja2<3.2,>=2.11.1 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling
Requirement already satisfied: visions[type_image_path]==0.7.5 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pand
Requirement already satisfied: numpy<1.24,>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling
Requirement already satisfied: htmlmin==0.1.12 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling) (0
Requirement already satisfied: phik<0.13,>=0.11.1 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling)
Requirement already satisfied: requests<3,>=2.24.0 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling
Requirement already satisfied: tqdm<5,>=4.48.2 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling) (4
Requirement already satisfied: seaborn<0.13,>=0.10.1 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profili
Requirement already satisfied: multimethod<2,>=1.4 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling
Requirement already satisfied: statsmodels<1,>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profil
Requirement already satisfied: typeguard<3,>=2.13.2 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profilin
Requirement already satisfied: imagehash==4.3.1 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling) (
Requirement already satisfied: wordcloud>=1.9.1 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling) (
Requirement already satisfied: dacite>=1.8 in /usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling) (1.8.1
Requirement already satisfied: PyWavelets in /usr/local/lib/python3.10/dist-packages (from imagehash==4.3.1->ydata-profiling->pandas-
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (from imagehash==4.3.1->ydata-profiling->pandas-prof
Requirement already satisfied: attrs>=19.3.0 in /usr/local/lib/python3.10/dist-packages (from visions[type_image_path]==0.7.5->ydata-
```

```
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4,>=3.2->ydata-profiling-
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4,>=3.2->ydata-profil
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas!=1.4.0,<2,>1.1->ydata-profiling->
Requirement already satisfied: joblib>=0.14.1 in /usr/local/lib/python3.10/dist-packages (from phik<0.13,>=0.11.1->ydata-profiling->p
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from pydantic<2,>=1.8.1->ydata-pr
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.24.0->ydata-prof
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.24.0->ydata-profili
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.24.0->ydata-
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.24.0->ydata-profiling->pa
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-packages (from statsmodels<1,>=0.13.2->ydata-profiling-
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.2->statsmodels<1,>=0.13.2->ydata-profi
Enabling notebook extension jupyter-js-widgets/extension...
Paths used for configuration of notebook:
        /root/.jupyter/nbconfig/notebook.json
Paths used for configuration of notebook:

      - Validating: OK
Paths used for configuration of notebook:
        /root/.jupyter/nbconfig/notebook.json
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.0.7)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.39.3)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.4)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.22.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (8.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (0.20.1)
```

```
1
2 # Read the individual CSV files
3 btc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Bitcoin.csv")
4 eth_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Ethereum.csv")
5 xrp_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_XRP.csv")
6 ltc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_Litecoin.csv")
7 usdc_df = pd.read_csv("https://raw.githubusercontent.com/Amarpreet3/CIND-820-CAPSTONE/main/Dataset/coin_USDCoin.csv")
8
9 # Concatenate the datasets vertically
10 df = pd.concat([btc_df, eth_df, xrp_df, ltc_df, usdc_df])
11
12 # Save the merged dataset to a new CSV file
13 df.to_csv("cryptocurrency.csv", index=False)
14
15
```

```
1 print(type(df))
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
1 df.head()
```

	SNo	Name	Symbol	Date	High	Low	Open	Close	Vol
0	1	Bitcoin	BTC	2013-04-29 23:59:59	147.488007	134.000000	134.444000	144.539993	
1	2	Bitcoin	BTC	2013-04-30 23:59:59	146.929993	134.050003	144.000000	139.000000	
				2013-					

```
1 df.tail()
```

	SNo	Name	Symbol	Date	High	Low	Open	Close	Vol

```
1 df.Name.unique()
```

```
array(['Bitcoin', 'Ethereum', 'XRP', 'Litecoin', 'USD Coin'], dtype=object)
```

2021-

```
1 df.Symbol.unique()
```

```
array(['BTC', 'ETH', 'XRP', 'LTC', 'USDC'], dtype=object)
```

```
1 df.columns
```

```
Index(['SNo', 'Name', 'Symbol', 'Date', 'High', 'Low', 'Open', 'Close',
       'Volume', 'Marketcap'],
      dtype='object')
```

```
1 df['Name'].value_counts()
```

```
Bitcoin     2991
Litecoin    2991
XRP         2893
Ethereum    2160
USD Coin    1002
Name: Name, dtype: int64
```

```
1 df.isnull().sum()
```

```
SNo          0
Name         0
Symbol       0
Date         0
High         0
Low          0
Open         0
Close        0
Volume       0
Marketcap    0
dtype: int64
```

```
1
2 print(df.dtypes)
```

```
SNo                  int64
Name                object
Symbol              object
Date         datetime64[ns]
High                float64
Low                 float64
Open                float64
Close               float64
Volume              float64
Marketcap           float64
dtype: object
```

```
1 df.isnull().count()
```

```
SNo          12037
Name         12037
Symbol       12037
Date         12037
High         12037
Low          12037
Open         12037
Close        12037
Volume       12037
Marketcap    12037
dtype: int64
```

```
1
2 # Get the shape of the merged dataset
3 rows, columns = df.shape
4 print("Number of rows:", rows)
5 print("Number of columns:", columns)
```

```
Number of rows: 12037
Number of columns: 10
```

```
1 # look at meta information about data, such as null values
2 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12037 entries, 0 to 1001
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   SNo        12037 non-null  int64  
 1   Name       12037 non-null  object 
 2   Symbol     12037 non-null  object 
 3   Date       12037 non-null  object 
 4   High       12037 non-null  float64
 5   Low        12037 non-null  float64
 6   Open       12037 non-null  float64
 7   Close      12037 non-null  float64
 8   Volume     12037 non-null  float64
 9   Marketcap  12037 non-null  float64
dtypes: float64(6), int64(1), object(3)
memory usage: 1.0+ MB
```

```
1 # Display summary statistics of numeric columns
2 numeric_columns = df.select_dtypes(include=[int, float])
3 summary_statistics = numeric_columns.describe()
4 print("Summary Statistics:")
5 print(summary_statistics)
```

```
Summary Statistics:
                SNo          High           Low          Open         Close  \
count  12037.000000  12037.000000  12037.000000  12037.000000  12037.000000   
mean    1326.877627   1797.247945   1689.071077   1745.961716   1748.923766   
std      836.777590   6507.990175   6085.346811   6312.961050   6319.601286   
min        1.000000      0.003082      0.002802      0.002809      0.002810   
25%      602.000000      1.000692      0.990570      0.999090      0.999228   
50%     1255.000000     33.345750     30.523111     32.016525     32.001958   
75%     2007.000000    372.767424    350.941986    360.312012    361.045990   
max     2991.000000  64863.098908  62208.964366  63523.754869  63503.457930   

             Volume     Marketcap  
count  1.203700e+04  1.203700e+04  
mean   4.652802e+09  4.082986e+10  
std    1.141572e+10  1.194340e+11  
min    0.000000e+00  0.000000e+00  
25%    5.895540e+06  3.682208e+08  
50%    3.204215e+08  5.655924e+09  
75%    3.146472e+09  2.090613e+10  
max    3.509679e+11  1.186364e+12  
```

```
1 # Let's see meta information about numeric data, we can also see if there any extreme values
2 df.describe()
```

	SNo	High	Low	Open	Close	V
count	12037.000000	12037.000000	12037.000000	12037.000000	12037.000000	1.20370
mean	1326.877627	1797.247945	1689.071077	1745.961716	1748.923766	4.65280
std	836.777590	6507.990175	6085.346811	6312.961050	6319.601286	1.14157
min	1.000000	0.003082	0.002802	0.002809	0.002810	0.00000
25%	602.000000	1.000692	0.990570	0.999090	0.999228	5.89554
50%	1255.000000	33.345750	30.523111	32.016525	32.001958	3.20421
75%	2007.000000	372.767424	350.941986	360.312012	361.045990	3.14647
max	2991.000000	64863.098908	62208.964366	63523.754869	63503.457930	3.50967

```
1 df.describe().transpose()
```

	count	mean	std	min	25%	50%
SNo	12037.0	1.326878e+03	8.367776e+02	1.000000	6.020000e+02	1.255000e+03
High	12037.0	1.797248e+03	6.507990e+03	0.003082	1.000692e+00	3.334575e+01
Low	12037.0	1.689071e+03	6.085347e+03	0.002802	9.905703e-01	3.052311e+01

```
# Convert the 'Date' column to datetime
df['Date'] = pd.to_datetime(df['Date'])

# Group the data by each coin
grouped_df = df.groupby('Symbol')

# Calculate the date duration for each coin and describtion
for symbol, group in grouped_df:
    min_date = group['Date'].min()
    max_date = group['Date'].max()
    duration = max_date - min_date

    print("Coin:", symbol)
    print("Start Date:", min_date)
    print("End Date:", max_date)
    print("Date Duration:", duration)
    print(group.describe().transpose())
    print()

```

```
Open       2991.0  4.924736e+01  6.325627e+01  1.153240e+00  3.792325e+00
Close      2991.0  4.927901e+01  6.324046e+01  1.157010e+00  3.794135e+00
```

```
Close       1.826080e-01  3.124647e-01  3.377810e+00
Volume      1.119370e+08  1.238721e+09  3.695518e+10
Marketcap   7.523251e+09  1.306444e+10  1.308535e+11
```

Categorical to One-Hot (numeric) Encoding

```
1 df.head()
```

	SNo	Name	Symbol	Date	High	Low	Open	Close	Volu
0	1	Bitcoin	BTC	2013-04-29 23:59:59	147.488007	134.000000	134.444000	144.539993	
1	2	Bitcoin	BTC	2013-04-30 23:59:59	146.929993	134.050003	144.000000	139.000000	
				2013-					

```
1 #Let's create a list for our categorical columns
2 cat_cols=["Name", "Symbol", "Date"]
```

```
1 # Create a copy of the data frame in memory with a different name
2 df_onehot=df.copy()
3 #convert only categorical variables/features to dummy/one-hot features
4 df_onehot = pd.get_dummies(df, columns=cat_cols, prefix = cat_cols)
5 #print the dataset
6 df_onehot
```

	SNo	High	Low	Open	Close	Volume	Marketc
0	1	147.488007	134.000000	134.444000	144.539993	0.000000e+00	1.603769e+
1	2	146.929993	134.050003	144.000000	139.000000	0.000000e+00	1.542813e+
2	3	139.889999	107.720001	139.000000	116.989998	0.000000e+00	1.298955e+
3	4	125.599998	92.281898	116.379997	105.209999	0.000000e+00	1.168517e+
4	5	108.127998	79.099998	106.250000	97.750000	0.000000e+00	1.085995e+
...	...	...	...	...	...	...	
997	998	1.000916	0.999966	1.000177	1.000035	1.787896e+09	2.539775e+
998	999	1.000670	0.999925	1.000048	0.999984	1.491017e+09	2.550437e+
999	1000	1.000187	0.998901	0.999956	0.999500	1.578667e+09	2.551172e+
1000	1001	1.000839	0.999459	0.999565	1.000528	1.887496e+09	2.554724e+
1001	1002	1.000731	0.999662	1.000501	1.000059	2.312602e+09	2.567322e+

12037 rows × 3008 columns

Pandas Profiling

```
1 from pandas_profiling import ProfileReport
2 profile = ProfileReport(df)
```

```
<ipython-input-51-ce1e41813c97>:1: DeprecationWarning: `import pandas_profiling` is going to be deprecated by April 1st. Please use `im
  from pandas_profiling import ProfileReport
```

```
1 profile.to_notebook_iframe()
```

Summarize dataset: 68/68 [00:17<00:00, 3.04it/s, 100% Completed]

Generate report structure: 1/1 [00:07<00:00, 7.13s/it] 100%

Render HTML: 100% 1/1 [00:01<00:00, 1.80s/it]

The Unicode Standard assigns character properties to each code point, which can be used to analyse textual variables.

Unique

Unique	0	**?**
Unique (%)	0.0%	

Sample

1st row	Bitcoin
2nd row	Bitcoin
3rd row	Bitcoin
4th row	Bitcoin
5th row	Bitcoin

Common Values

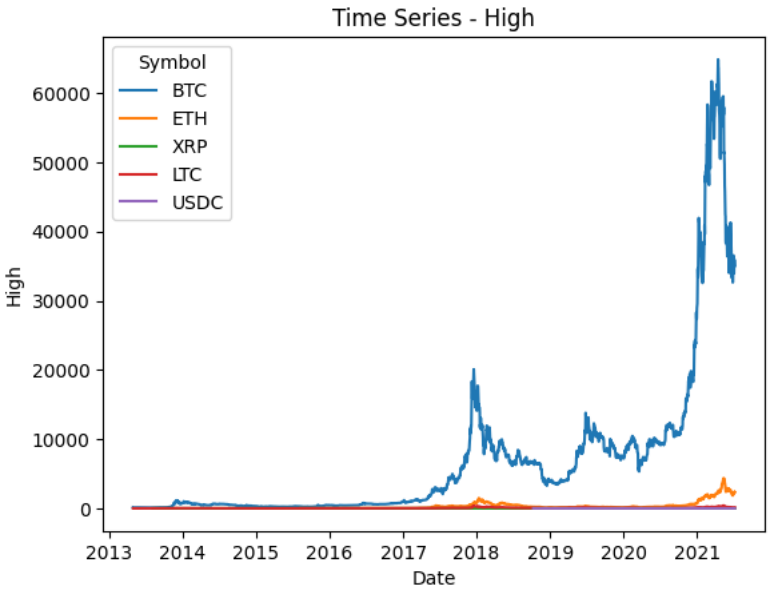
Value	Count	Frequency (%)
Bitcoin	2991	24.8%
Litecoin	2991	24.8%
XRP	2893	24.0%
Ethereum	2160	17.9%
USD Coin	1002	8.3%

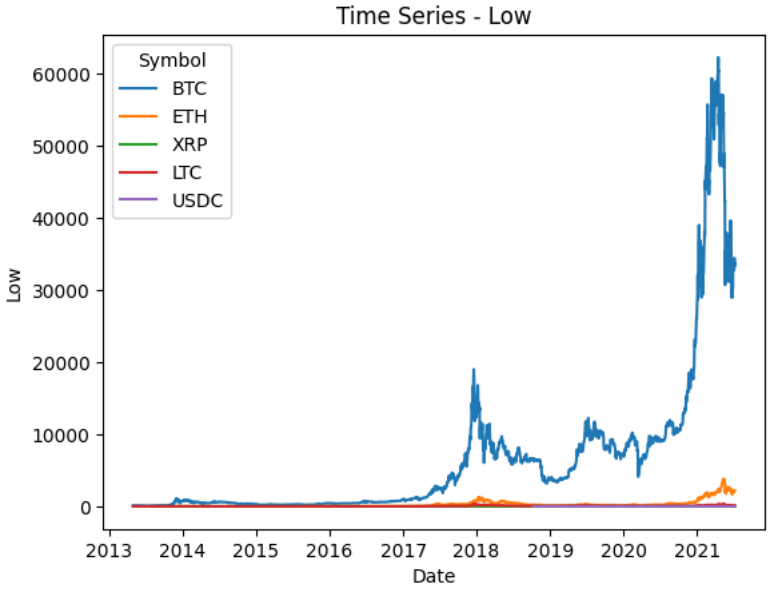
Visualizing the Data:

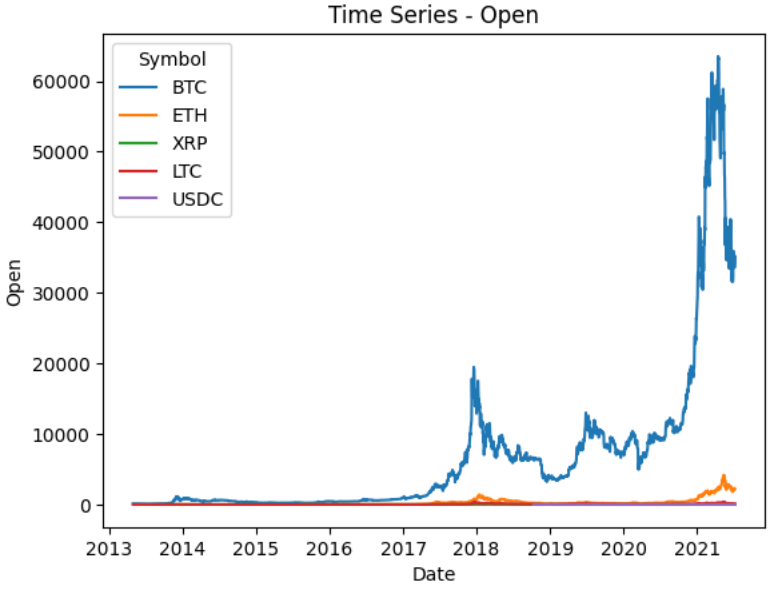
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
```

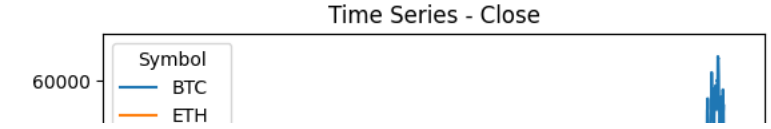
```
1 #DataFrame copied to keep the original unaltered
2 df_visual = df.copy()
```

```
1
2 # List of columns to visualize
3 columns_to_visualize = ['High', 'Low', 'Open', 'Close', 'Volume', 'Marketcap']
4
5 # Loop through each column
6 for column in columns_to_visualize:
7     # Set up the plot for the current column
8     sns.lineplot(data=df_visual, x='Date', y=column, hue='Symbol')
9
10    # Add title and labels
11    plt.title(f'Time Series - {column}')
12    plt.xlabel('Date')
13    plt.ylabel(column)
14
15    # Display the plot
16    plt.show()
17
```

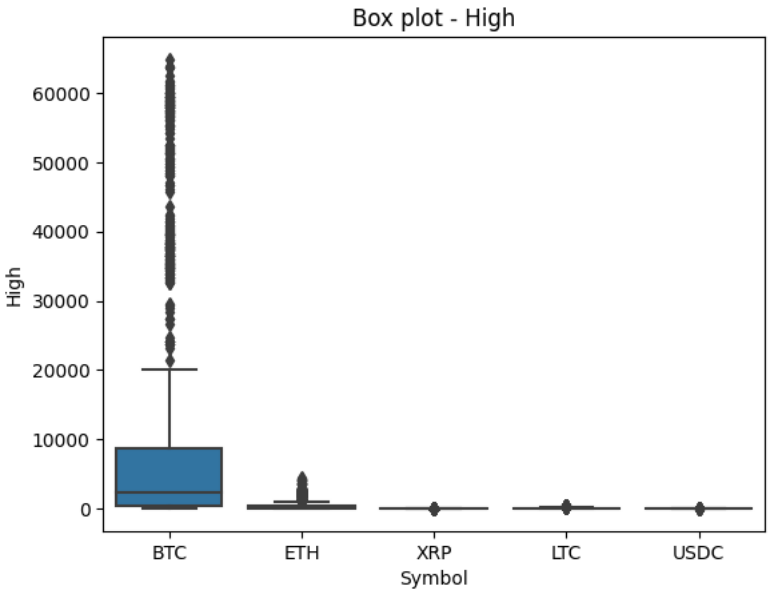

Time Series - High
Symbol
BTC
ETH
XRP
LTC
USDC
High
60000
50000
40000
30000
20000
10000
0
2013
2014
2015
2016
2017
2018
2019
2020
2021
Date

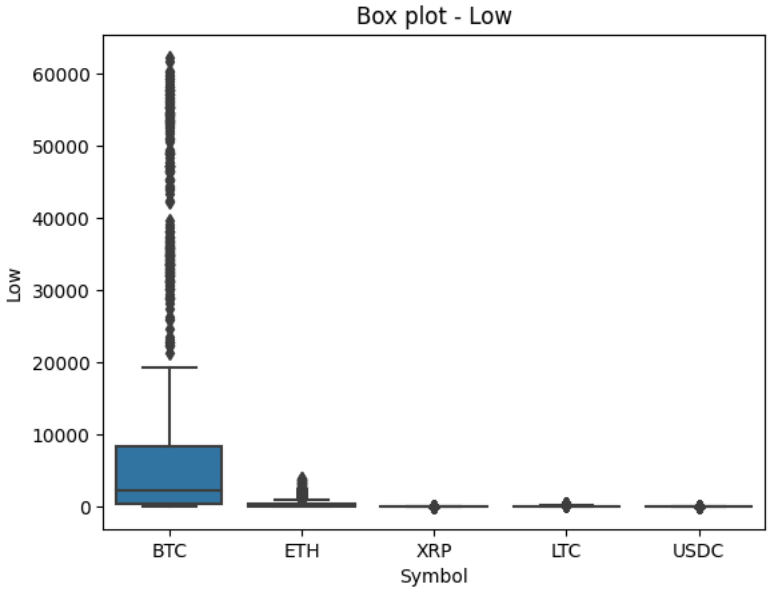

Time Series - Low
Symbol
BTC
ETH
XRP
LTC
USDC
Low
60000
50000
40000
30000
20000
10000
0
2013
2014
2015
2016
2017
2018
2019
2020
2021
Date

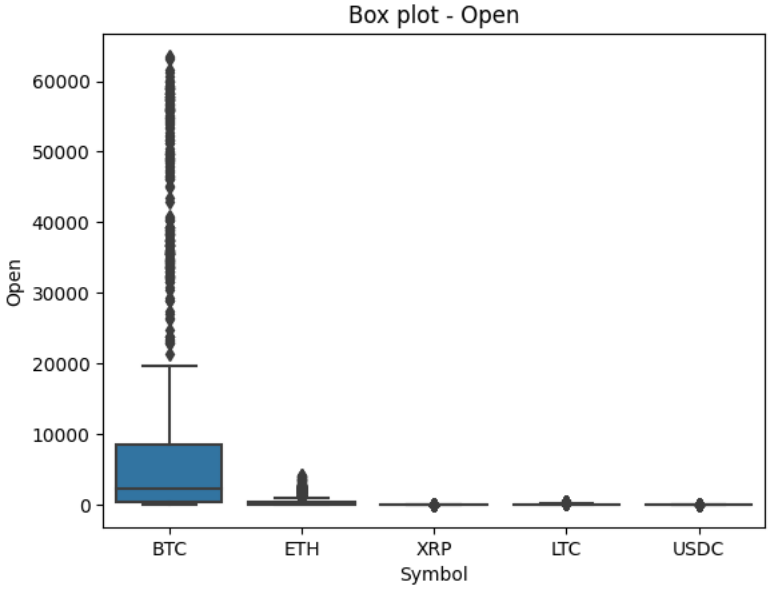

Time Series - Open
Symbol
BTC
ETH
XRP
LTC
USDC
Open
60000
50000
40000
30000
20000
10000
0
2013
2014
2015
2016
2017
2018
2019
2020
2021
Date

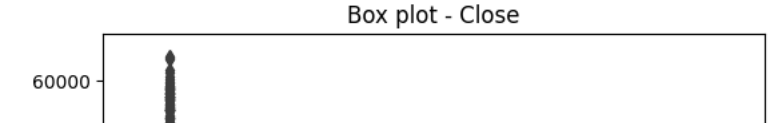

Time Series - Close
Symbol
BTC
ETH
60000

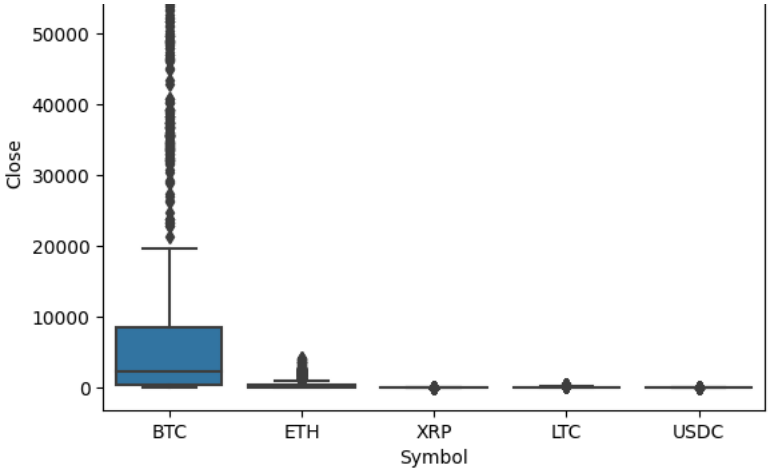
```
1
2 # Loop through each column
3 for column in columns_to_visualize:
4     # Set up the boxplot for the current column
5     sns.boxplot(x='Symbol', y=column, data=df_visual)
6
7     # Add title and labels
8     plt.title(f'Box plot - {column}')
9     plt.xlabel('Symbol')
10    plt.ylabel(column)
11
12    # Display the plot
13    plt.show()
14
```

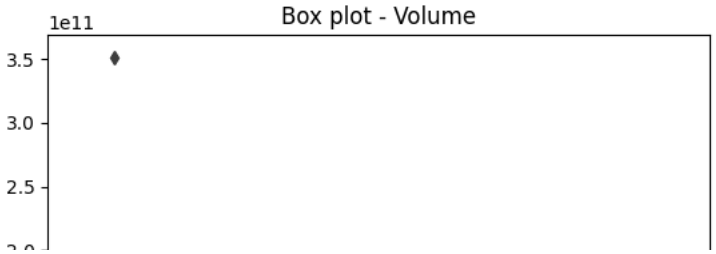

Box plot - High
High
60000
50000
40000
30000
20000
10000
0
BTC
ETH
XRP
LTC
USDC
Symbol


Box plot - Low
Low
60000
50000
40000
30000
20000
10000
0
BTC
ETH
XRP
LTC
USDC
Symbol


Box plot - Open
Open
60000
50000
40000
30000
20000
10000
0
BTC
ETH
XRP
LTC
USDC
Symbol


Box plot - Close
60000


50000
40000
30000
20000
10000
0
Close
BTC
ETH
XRP
LTC
USDC
Symbol


Box plot - Volume
1e11
3.5
3.0
2.5

racts here