



## **CASE STUDY ON COMMODITY PRICES DATASET**

**SUBMITTED BY ADITYA TIRAKAPADI**

**Problem statement :-** The problem is to analyze commodity prices for various commodities using the commodity prices dataset. The goal is to leverage Python, data science techniques, statistical analysis and data modeling. Perform all necessary steps to get the key insights from the data.

**Dataset Description :-** This dataset contains monthly commodity prices from 1960 to 2022. The commodity prices dataset includes the following attributes:

Attributes	Description
date	The date of the recorded commodity price
oil_brent	The price of Brent oil (\$/bbl)
Oil_Dubai	Dubai The price of Dubai oil (\$/bbl)
Coffee_Arabica	The price of Arabica coffee (\$/kg)
Coffee_Robustas	The price of Robusta coffee (\$/kg)
Tea_Columbo	The price of Columbo tea (\$/kg)
Tea_Kolkata	The price of Kolkata tea (\$/kg)
Tea_Mombasa	The price of Mombasa tea (\$/kg)
Sugar_EU	The price of US sugar (\$/kg)
Sugar_World	The price of global sugar (\$/kg)

# OBJECTIVE

## Main Goals of the Case Study:

- Analyze historical data to understand **commodity price trends** over time.
- Identify **regional and global variations** in commodity prices.
- Examine **relationships and correlations** between different commodities.
- Provide actionable insights for businesses, policymakers, or investors.

## Specific Questions Addressed:

- 1.How have commodity prices evolved over the decades?
- 2.What differences exist between regional prices?
- 3.Are there any notable correlations between commodities?
- 4.Can the trends suggest future price movements?

# Methodology

## 1.1 Data Loading:

- Dataset commodity\_prices.csv loaded into a pandas Data Frame.
- Reviewed file structure, column names, and data types.

```
1 #Importing necessary libraries.
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import seaborn as sns
```

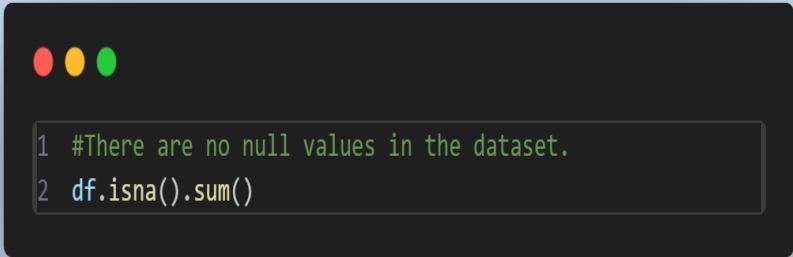
```
1 #Importing the dataset.
2 df = pd.read_csv(r"C:\Users\adity\Downloads\I
  nteelipat course\commodity_prices.csv")
3 df
```

Unnamed: 0		date	oil_brent	oil_dubai	coffee_arabica	coffee_robustas	tea_columbo	tea_kolkata	tea_mombasa	sugar_eu	sugar_us	sugar_world
0	1	01-01-1960	1.63	1.63	0.940900	0.696864	0.930301	1.121401	1.037400	0.122356	0.116845	0.066600
1	2	01-02-1960	1.63	1.63	0.946900	0.688707	0.930301	1.121401	1.037400	0.122356	0.119049	0.067900
2	3	01-03-1960	1.63	1.63	0.928100	0.688707	0.930301	1.121401	1.037400	0.122356	0.121254	0.068300
3	4	01-04-1960	1.63	1.63	0.930300	0.684519	0.930301	1.121401	1.037400	0.122356	0.123459	0.068100
4	5	01-05-1960	1.63	1.63	0.920000	0.690692	0.930301	1.121401	1.037400	0.122356	0.121254	0.068300
...	...	...	...	...	...	...	...	...	...	...	...	...
751	752	01-08-2022	98.60	97.75	5.917861	2.417366	4.210000	3.538154	2.360000	0.330773	0.782199	0.393525
752	753	01-09-2022	90.16	90.63	5.897138	2.455065	4.490000	3.153198	2.360000	0.323621	0.770956	0.390659
753	754	01-10-2022	93.13	90.59	5.292852	2.270979	4.135621	2.833112	2.457500	0.320943	0.762578	0.386911
754	755	01-11-2022	91.07	86.28	4.715462	2.041258	3.831528	2.849979	2.490000	0.332993	0.792340	0.407414
755	756	01-12-2022	80.90	76.78	4.629482	2.045446	3.994073	2.421516	2.386667	0.345729	0.805127	0.417335

756 rows × 12 columns

## 1.2 Data Cleaning and Preprocessing:

- Converted date column to a **datetime format** for time-series analysis.
- Checked for **missing values** and inconsistencies.



```
1 #There are no null values in the dataset.  
2 df.isna().sum()
```

```
Unnamed: 0      0  
date            0  
oil_brent       0  
oil_dubai       0  
coffee_arabica  0  
coffee_robustas 0  
tea_columbo     0  
tea_kolkata     0  
tea_mombasa     0  
sugar_eu        0  
sugar_us        0  
sugar_world     0  
dtype: int64
```

## 2.1 GETTING INFORMATION ABOUT THE DATASET

So basically, There is good thing which we can see in this is there are no null values in our dataset and we have all the information the entire information from column name the column count to datatypes which helps to get a rough idea like what a dataset comprises off.

```
1 #Dataset information.  
2 df.info()
```

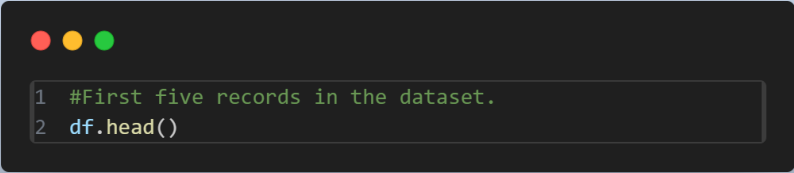
```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 756 entries, 0 to 755  
Data columns (total 12 columns):  
#      Column              Non-Null Count  Dtype  
---  -  
0     Unnamed: 0              756 non-null   int64  
1     date                    756 non-null   object  
2     oil_brent               756 non-null   float64  
3     oil_dubai               756 non-null   float64  
4     coffee_arabica          756 non-null   float64  
5     coffee_robustas         756 non-null   float64  
6     tea_columbo             756 non-null   float64  
7     tea_kolkata             756 non-null   float64  
8     tea_mombasa             756 non-null   float64  
9     sugar_eu                756 non-null   float64  
10    sugar_us                756 non-null   float64  
11    sugar_world             756 non-null   float64  
dtypes: float64(10), int64(1), object(1)  
memory usage: 71.0+ KB
```

## 2.2 SHAPE ,HEAD AND TAIL FUNCTIONS

Shape :- Used get the shape/count of number of rows and column present in the dataset.

Head :- Used to fetch the first 5 columns of the dataset.

Tail :- Used to fetch last 5 columns of the dataset.



```
1 #First five records in the dataset.  
2 df.head()
```

Unnamed: 0		date	oil_brent	oil_dubai	coffee_arabica	coffee_robustas	tea_columbo	tea_kolkata	tea_mombasa	sugar_eu	sugar_us	sugar_world
0	1	01-01-1960	1.63	1.63	0.9409	0.696864	0.930301	1.121401	1.0374	0.122356	0.116845	0.0666
1	2	01-02-1960	1.63	1.63	0.9469	0.688707	0.930301	1.121401	1.0374	0.122356	0.119049	0.0679
2	3	01-03-1960	1.63	1.63	0.9281	0.688707	0.930301	1.121401	1.0374	0.122356	0.121254	0.0683
3	4	01-04-1960	1.63	1.63	0.9303	0.684519	0.930301	1.121401	1.0374	0.122356	0.123459	0.0681
4	5	01-05-1960	1.63	1.63	0.9200	0.690692	0.930301	1.121401	1.0374	0.122356	0.121254	0.0683



```
1 #For checking the number of rows and columns.  
2 df.shape
```

(756, 12)



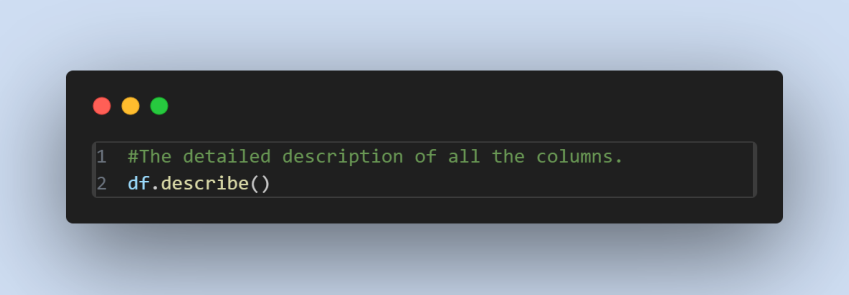
```
1 #Last 5 record in the data set.  
2 df.tail()
```

Unnamed: 0		date	oil_brent	oil_dubai	coffee_arabica	coffee_robustas	tea_columbo	tea_kolkata	tea_mombasa	sugar_eu	sugar_us	sugar_world
751	752	01-08-2022	98.60	97.75	5.917861	2.417366	4.210000	3.538154	2.360000	0.330773	0.782199	0.393525
752	753	01-09-2022	90.16	90.63	5.897138	2.455065	4.490000	3.153198	2.360000	0.323621	0.770956	0.390659
753	754	01-10-2022	93.13	90.59	5.292852	2.270979	4.135621	2.833112	2.457500	0.320943	0.762578	0.386911
754	755	01-11-2022	91.07	86.28	4.715462	2.041258	3.831528	2.849979	2.490000	0.332993	0.792340	0.407414
755	756	01-12-2022	80.90	76.78	4.629482	2.045446	3.994073	2.421516	2.386667	0.345729	0.805127	0.417335



## 2.3 DESCRIBE FUNCTION

- The describe() function is a powerful tool used to quickly summarize the main statistical characteristics of a dataset.
- It provides a fast and concise summary of the data's central tendency, spread, and distribution.
- By looking at the min, max, and quartiles, potential outliers can be identified.
- The percentiles give a sense of the shape of the distribution, whether it is skewed or symmetric.
- Comparing the statistics across different columns can reveal relationships and differences in their characteristics.



```
1 #The detailed description of all the columns.  
2 df.describe()
```

	Unnamed: 0	oil_brent	oil_dubai	coffee_arabica	coffee_robustas	tea_columbo	tea_kolkata	tea_mombasa	sugar_eu	sugar_us	sugar_world
count	756.000000	756.000000	756.000000	756.000000	756.000000	756.000000	756.000000	756.000000	756.000000	756.000000	756.000000
mean	378.500000	32.724944	31.238130	2.576555	1.727478	1.777962	1.870308	1.671222	0.405158	0.432462	0.240263
std	218.382692	31.885368	30.936611	1.342454	0.940748	1.008679	0.697867	0.615357	0.187741	0.188589	0.151947
min	1.000000	1.210000	1.210000	0.777600	0.487210	0.434198	0.664799	0.719600	0.112215	0.116845	0.028700
25%	189.750000	10.564999	10.452500	1.351625	0.923053	0.892501	1.297369	1.136800	0.298120	0.297624	0.139705
50%	378.500000	20.489130	18.550000	2.697794	1.632172	1.504001	1.850612	1.598257	0.402343	0.471119	0.215285
75%	567.250000	47.157500	45.576023	3.312950	2.282200	2.515204	2.376899	2.083830	0.569519	0.512188	0.309325
max	756.000000	133.873043	131.224783	7.003600	6.883547	4.490000	4.073011	3.392500	0.783171	1.263247	1.237700

## 2.4 CHECKING FOR DUPLICATES IN THE DATASET

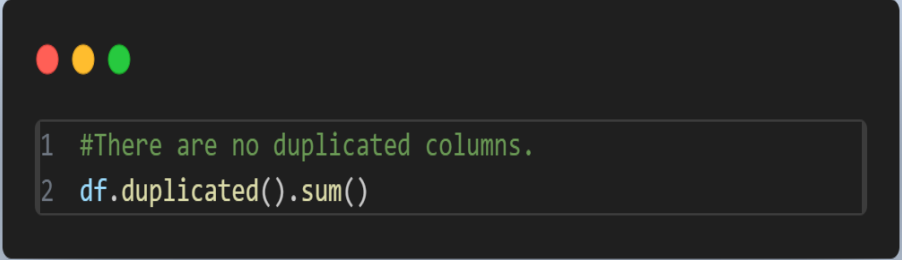
Checking for duplicates is important because it affects :

➤ **Data quality:**

Duplicate data can skew your analysis results, leading to inaccurate insights.

➤ **Data cleaning:**

Identifying and removing duplicates is a crucial step in data preprocessing before further analysis.



```
1 #There are no duplicated columns.  
2 df.duplicated().sum()
```

```
np.int64(0)
```

OUR DATASET DOESN'T CONTAIN ANY DUPLICATES

## 2.6 ACCESSING THE MAXIMUM AND MINIMUM PRICES OF ALL THE COLUMNS

- The "max" and "min" functions are used to identify the highest and lowest values respectively within a given data column.
- The "max" function returns the largest value in a dataset, while "min" returns the smallest value.
- By calculating the maximum and minimum values, you can quickly assess the range of data within a variable, which is useful for identifying potential outliers or understanding the overall variability.

```
1 #Minimum prices of all the columns.  
2 min = df.min()  
3 min
```

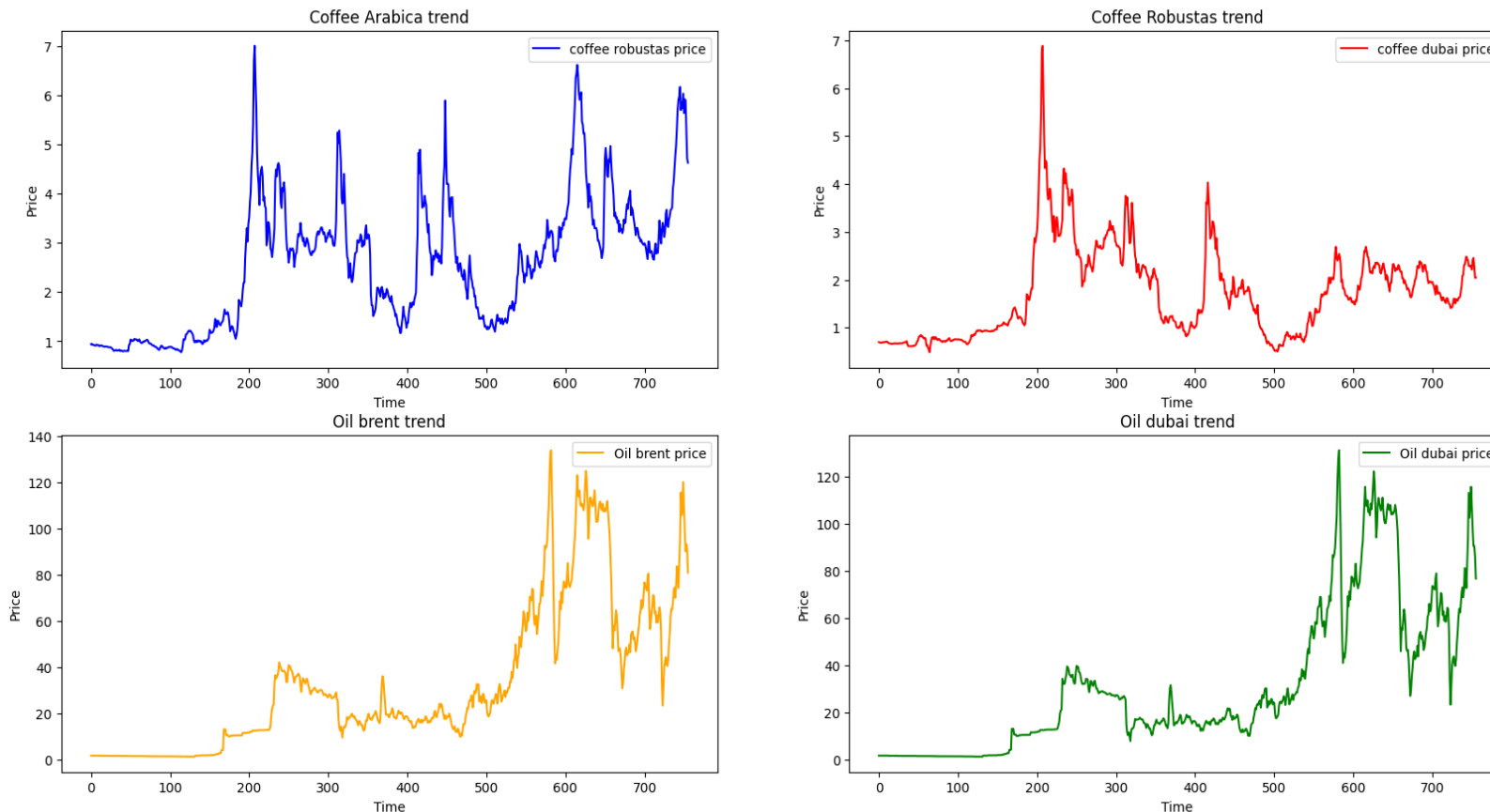
```
1 #Maximum prices of all the columns.  
2 max = df.max()  
3 max
```

Unnamed: 0	MIN	1	Unnamed: 0	MAX	756
date	01-01-1960		date	01-12-2022	
oil_brent	1.21		oil_brent	133.873043	
oil_dubai	1.21		oil_dubai	131.224783	
coffee_arabica	0.7776		coffee_arabica	7.0036	
coffee_robustas	0.48721		coffee_robustas	6.883547	
tea_columbo	0.434198		tea_columbo	4.49	
tea_kolkata	0.664799		tea_kolkata	4.073011	
tea_mombasa	0.7196		tea_mombasa	3.3925	
sugar_eu	0.112215		sugar_eu	0.783171	
sugar_us	0.116845		sugar_us	1.263247	
sugar_world	0.0287		sugar_world	1.2377	
dtype: object			dtype: object		

### 3. Exploratory Data Analysis (EDA):

- Used statistical summaries to understand data distribution.
- Identified outliers and anomalies in commodity prices.

#### 3.1. Line chart's of coffee and oil

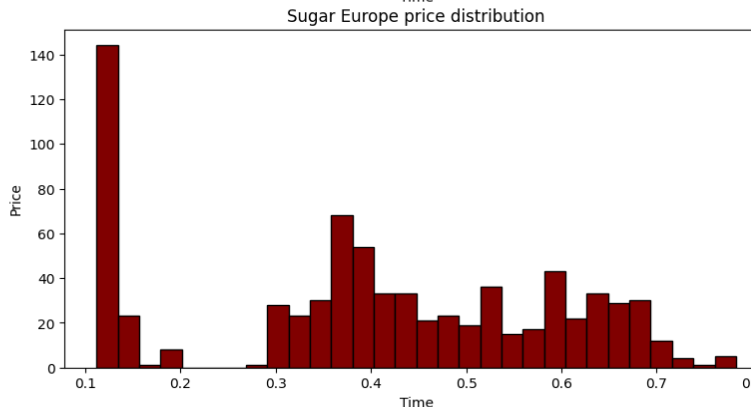
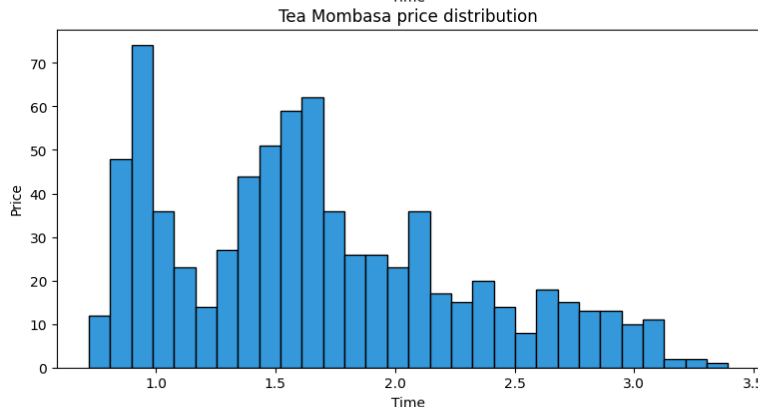
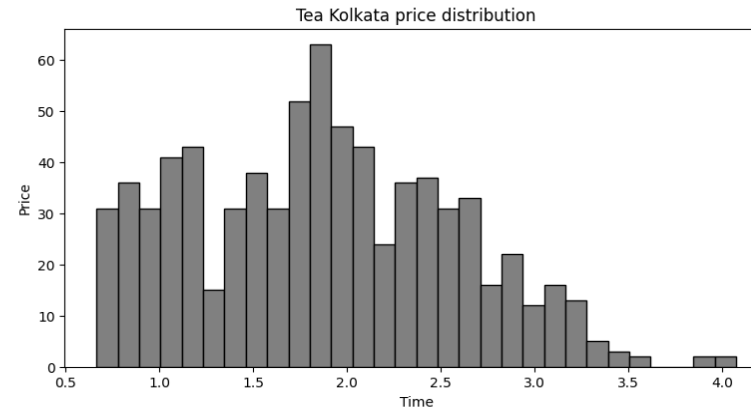
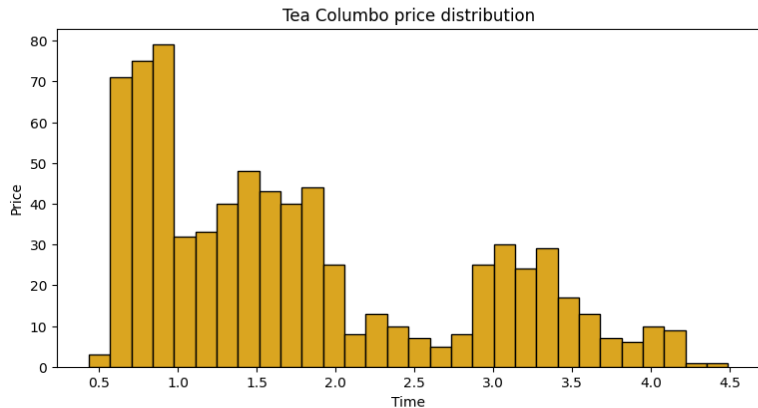


```
1 #What is the overall price trend for each commodity ?
2 # Line chart to check the trends over time.
3 fig = plt.subplots(2,2,figsize = (20,10))
4 plt.subplot(2,2,1)
5 plt.plot(df['coffee_arabica'],label =
6         'coffee robustas price',color = 'Blue')
7 plt.title("Coffee Arabica trend")
8 plt.ylabel("Price")
9 plt.xlabel("Time")
10 plt.legend()
11
12 plt.subplot(2,2,2)
13 plt.plot(df['coffee_robustas'],label =
14         'coffee dubai price',color = 'Red')
15 plt.title("Coffee Robustas trend")
16 plt.ylabel("Price")
17 plt.xlabel("Time")
18 plt.legend()
19
20 plt.subplot(2,2,3)
21 plt.plot(df['oil_brent'],label = 'Oil brent price',
22         color = 'Orange')
23 plt.title("Oil brent trend")
24 plt.ylabel("Price")
25 plt.xlabel("Time")
26 plt.legend()
27
28 plt.subplot(2,2,4)
29 plt.plot(df['oil_dubai'],label = 'Oil dubai price',
30         color = 'Green')
31 plt.title("Oil dubai trend")
32 plt.ylabel("Price")
33 plt.xlabel("Time")
34 plt.legend()
35 plt.show()
```

# HISTOGRAM

- Histograms are used to summarize and illustrate the distribution of data.
- They can help you understand the general features of a dataset, such as where the peaks are, if the distribution is skewed or symmetric, and if there are any outliers.

## 3.2. Histogram for checking price distributions

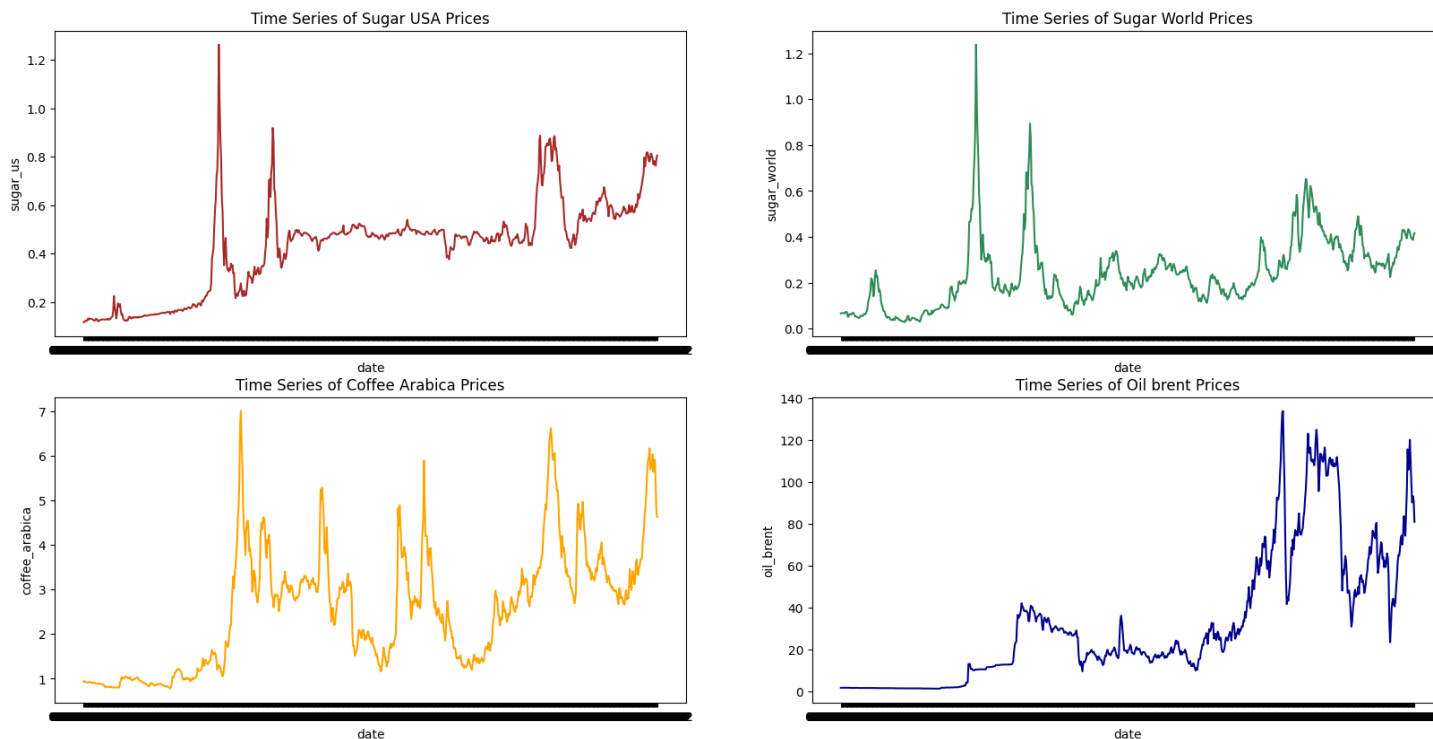


```
1 #Hist plots for checking price distribution.
2 fig = plt.subplots(2,2,figsize = (20,10))
3 plt.subplot(2,2,1)
4 plt.hist(df['tea_columbo'],color = 'goldenrod',bins=
5 30,edgecolor = 'black')
6 plt.title("Tea Columbo price distribution")
7 plt.ylabel("Price")
8 plt.xlabel("Time")
9
10 plt.subplot(2,2,2)
11 plt.hist(df['tea_kolkata'],color = 'Grey',bins=30,
12 edgecolor = 'black')
13 plt.title("Tea Kolkata price distribution")
14 plt.ylabel("Price")
15 plt.xlabel("Time")
16
17 plt.subplot(2,2,3)
18 plt.hist(df['tea_mombasa'],color = '#3498DB',bins=30,
19 edgecolor = 'black')
20 plt.title("Tea Mombasa price distribution")
21 plt.ylabel("Price")
22 plt.xlabel("Time")
23
24 plt.subplot(2,2,4)
25 plt.hist(df['sugar_eu'],color = 'maroon',bins=30,
26 edgecolor = 'black')
27 plt.title("Sugar Europe price distribution")
28 plt.ylabel("Price")
29 plt.xlabel("Time")
30 plt.show()
```

# Line plots

- A line plot function is a tool that creates a graph that connects data points with lines to show how values change over time:
- Line plots are used to visualize and analyze data trends and patterns. They are useful for comparing sets of data or tracking changes over time.

## 3.3. Line plot for checking time series and pricing



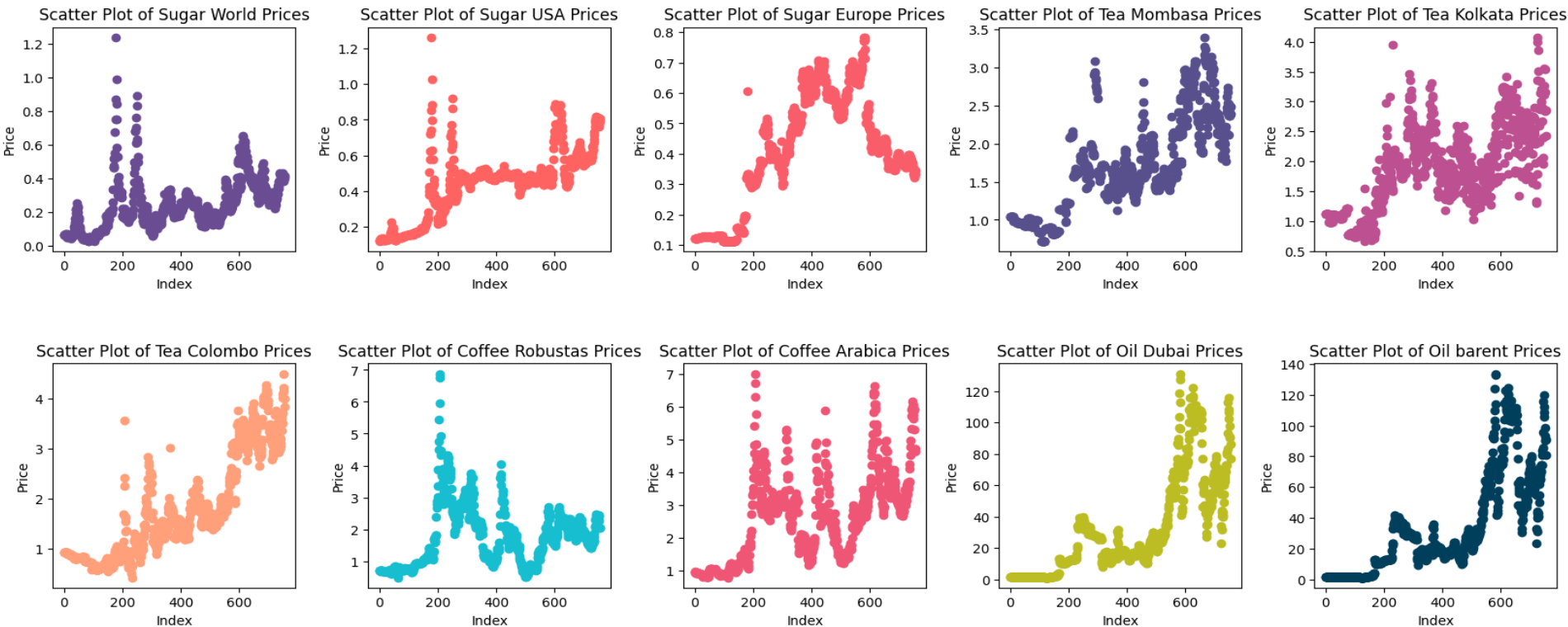
```
1 #Line plots to check the time series of USA prices
2 fig = plt.subplots(2,2,figsize = (20,10))
3 plt.subplot(2,2,1)
4 sns.lineplot(data=df,x='date', y='sugar_us', color=
  '#a52a2a')
5 plt.title('Time Series of Sugar USA Prices')
6
7 plt.subplot(2,2,2)
8 sns.lineplot(data=df,x='date', y='sugar_world', color=
  '#2e8b57')
9 plt.title('Time Series of Sugar World Prices')
10
11 plt.subplot(2,2,3)
12 sns.lineplot(data=df,x='date', y='coffee_arabica',
  color='#ffa500')
13 plt.title('Time Series of Coffee Arabica Prices')
14
15 plt.subplot(2,2,4)
16 sns.lineplot(data=df,x='date', y='oil_brent', color=
  '#00008b')
17 plt.title('Time Series of Oil brent Prices')
18 plt.show()
```



# SCATTER PLOTS

A "scatter plot" is used to visualize the relationship between two numerical variables within a dataset, displaying each data point as a dot on a graph to identify potential correlations or patterns between them; essentially, it helps you see how one variable changes with respect to another. It's used to visually examine the correlation between two variables, including the strength and direction of the relationship.

## 3.4. SCATTER PLOTS FOR ALL COLUMNS



```
#Scatter plots for comparing all the columns
fig = plt.subplots(2,5,figsize=(20,8))
plt.subplot(2,5,1)
plt.scatter(df.index, df['sugar_world'], color='#6a4c93')
plt.title('Scatter Plot of Sugar World Prices')
plt.xlabel('Index')
plt.ylabel('Price')

plt.subplot(2,5,2)
plt.scatter(df.index, df['sugar_us'], color='#ff6361')
plt.title('Scatter Plot of Sugar USA Prices')
plt.xlabel('Index')
plt.ylabel('Price')

plt.subplot(2,5,3)
plt.scatter(df.index, df['sugar_eu'], color='#f95d6a')
plt.title('Scatter Plot of Sugar Europe Prices')
plt.xlabel('Index')
plt.ylabel('Price')

plt.subplot(2,5,4)
plt.scatter(df.index, df['tea_mombasa'], color='#58508d')
plt.title('Scatter Plot of Tea Mombasa Prices')
plt.xlabel('Index')
plt.ylabel('Price')

plt.subplot(2,5,5)
plt.scatter(df.index, df['tea_kolkata'], color='#bc5090')
plt.title('Scatter Plot of Tea Kolkata Prices')
plt.xlabel('Index')
plt.ylabel('Price')

plt.subplot(2,5,6)
plt.scatter(df.index, df['tea_colombo'], color='#ffa07a')
plt.title('Scatter Plot of Tea Colombo Prices')
plt.xlabel('Index')
plt.ylabel('Price')

plt.subplot(2,5,7)
plt.scatter(df.index, df['coffee_robustas'], color=
'#17becf')
plt.title('Scatter Plot of Coffee Robustas Prices')
plt.xlabel('Index')
plt.ylabel('Price')

plt.subplot(2,5,8)
plt.scatter(df.index, df['coffee_arabica'], color=
'#ef5675')
plt.title('Scatter Plot of Coffee Arabica Prices')
plt.xlabel('Index')
plt.ylabel('Price')

plt.subplot(2,5,9)
plt.scatter(df.index, df['oil_dubai'], color='#bcbd22')
plt.title('Scatter Plot of Oil Dubai Prices')
plt.xlabel('Index')
plt.ylabel('Price')

plt.subplot(2,5,10)
plt.scatter(df.index, df['oil_brent'], color='#003f5c')
plt.title('Scatter Plot of Oil Barent Prices')
plt.xlabel('Index')
plt.ylabel('Price')

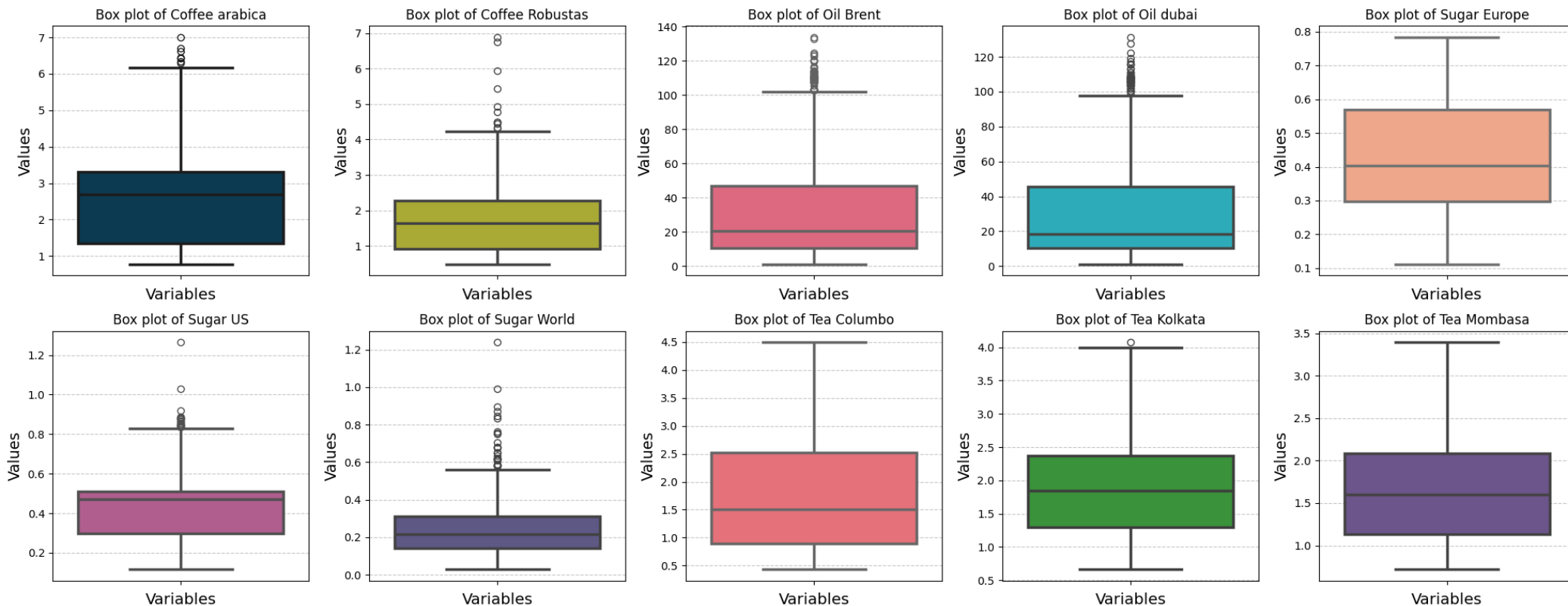
plt.subplots_adjust(
    wspace=0.3,
    hspace=0.5
)
plt.show()
```

# BOX PLOT

A "box plot" is used to visually represent the distribution of a numerical variable, highlighting key statistics like the median, quartiles, and potential outliers, allowing for quick comparison between different groups within the data.

The box represents the interquartile range (IQR) with the median line inside, while the "whiskers" extend to the minimum and maximum values (excluding outliers) majorly helps to find outliers.

## 3.5. BOX PLOTS FOR CHECKING OUTLIERS



```
#Checking for outliers
fig = plt.subplots(2,5,figsize=(20,8))
plt.subplot(2,5,1)
sns.boxplot(data=df['coffee_arabica'], color='#003f5c',
showfliers=True, linewidth=2.5)
plt.title("Box plot of Coffee arabica")
plt.xlabel("Variables", fontsize=14)
plt.ylabel("Values", fontsize=14)
plt.grid(axis="y", linestyle="--", alpha=0.7)

plt.subplot(2,5,2)
sns.boxplot(data=df['coffee_robustas'], color='#bcbdd2',
showfliers=True, linewidth=2.5)
plt.title("Box plot of Coffee Robustas")
plt.xlabel("Variables", fontsize=14)
plt.ylabel("Values", fontsize=14)
plt.grid(axis="y", linestyle="--", alpha=0.7)

plt.subplot(2,5,3)
sns.boxplot(data=df['oil_brent'], color='#ef5675',
showfliers=True, linewidth=2.5)
plt.title("Box plot of Oil Brent")
plt.xlabel("Variables", fontsize=14)
plt.ylabel("Values", fontsize=14)
plt.grid(axis="y", linestyle="--", alpha=0.7)

plt.subplot(2,5,4)
sns.boxplot(data=df['oil_dubai'],color='#17becf',
showfliers=True, linewidth=2.5)
plt.title("Box plot of Oil dubai")
plt.xlabel("Variables", fontsize=14)
plt.ylabel("Values", fontsize=14)
plt.grid(axis="y", linestyle="--", alpha=0.7)

plt.subplot(2,5,5)
sns.boxplot(data=df['sugar_eu'], color='#ffa07a',
showfliers=True, linewidth=2.5)
plt.title("Box plot of Sugar Europe")
plt.xlabel("Variables", fontsize=14)
plt.ylabel("Values", fontsize=14)
plt.grid(axis="y", linestyle="--", alpha=0.7)

plt.subplot(2,5,6)
sns.boxplot(data=df['sugar_us'], color='#bc5090',
showfliers=True, linewidth=2.5)
plt.title("Box plot of Sugar US")
plt.xlabel("Variables", fontsize=14)
plt.ylabel("Values", fontsize=14)
plt.grid(axis="y", linestyle="--", alpha=0.7)

plt.subplot(2,5,7)
sns.boxplot(data=df['sugar_world'], color='#58508d',
showfliers=True, linewidth=2.5)
plt.title("Box plot of Sugar World")
plt.xlabel("Variables", fontsize=14)
plt.ylabel("Values", fontsize=14)
plt.grid(axis="y", linestyle="--", alpha=0.7)

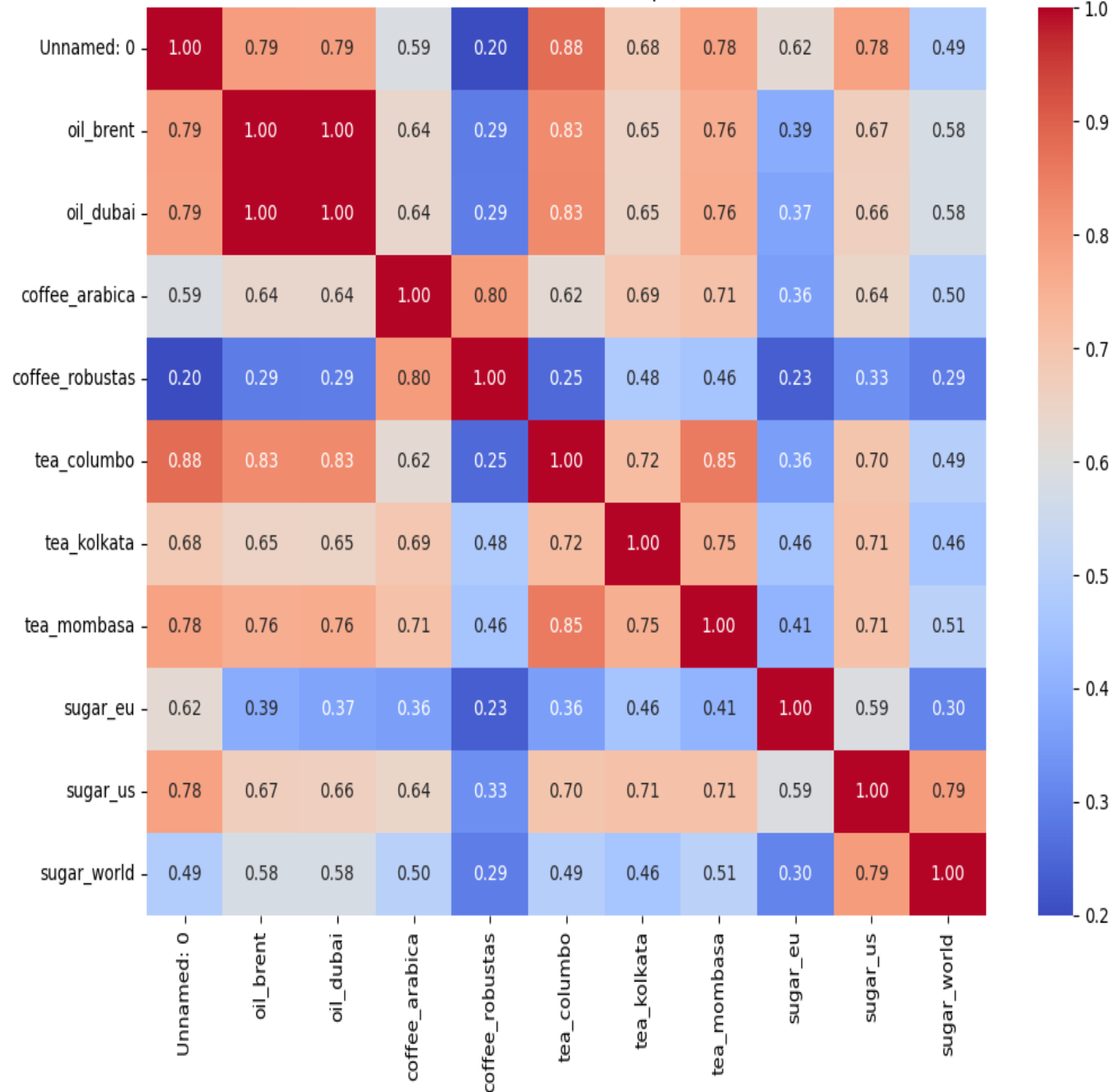
plt.subplot(2,5,8)
sns.boxplot(data=df['tea_columbo'], color='#f95d6a',
showfliers=True, linewidth=2.5)
plt.title("Box plot of Tea Columbo")
plt.xlabel("Variables", fontsize=14)
plt.ylabel("Values", fontsize=14)
plt.grid(axis="y", linestyle="--", alpha=0.7)

plt.subplot(2,5,9)
sns.boxplot(data=df['tea_kolkata'], color='#2ca02c',
showfliers=True, linewidth=2.5)
plt.title("Box plot of Tea Kolkata")
plt.xlabel("Variables", fontsize=14)
plt.ylabel("Values", fontsize=14)
plt.grid(axis="y", linestyle="--", alpha=0.7)

plt.subplot(2,5,10)
sns.boxplot(data=df['tea_mombasa'], color='#6a4c93',
showfliers=True, linewidth=2.5)
plt.title("Box plot of Tea Mombasa")
plt.xlabel("Variables", fontsize=14)
plt.ylabel("Values", fontsize=14)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.tight_layout()
plt.show()
```



Correlation Heatmap



HEAT MAP

- A correlation heatmap is a visual graphic that shows how each variable in the dataset are correlated to one another.
- A data visualization technique that displays the correlation coefficients between all pairs of variables in a dataset using a color-coded grid.
- To identify potential relationships between multiple variables in a dataset, particularly highlighting strong positive or negative correlations which can be useful for feature selection in machine learning models.

```

1 #HeatMap
2 plt.figure(figsize=(12, 8))
3 sns.heatmap(numeric_df.corr(), annot=True, cmap=
  "coolwarm", fmt=".2f")
4 plt.title("Correlation Heatmap")
5 plt.show()

```

# CONCLUSION

## Summary of Findings:

This analysis provided valuable insights into the trends, relationships, and regional variations in commodity prices over the decades. By leveraging a robust dataset and employing exploratory data analysis techniques, we were able to uncover meaningful patterns that offer a better understanding of market dynamics:

### 1.Oil Prices:

- Brent and Dubai oil prices showed remarkable stability for much of the 20th century but experienced significant volatility post-2000.
- This shift aligns with major global events, including geopolitical conflicts, economic booms, and downturns.
- The high correlation between Brent and Dubai oil prices reflects the global interconnectedness of energy markets.

### 2.Coffee Prices:

- Arabica coffee consistently commands a higher price than Robusta due to its superior quality and consumer preference.
- Both coffee varieties exhibit periodic price spikes, possibly driven by environmental factors like droughts or diseases in key coffee-producing regions.
- The interplay between supply chain disruptions and global demand significantly impacts coffee pricing.

### **3. Tea Prices:**

- Regional variations in tea prices highlight the influence of local market dynamics and production costs.
- Colombo and Kolkata maintain higher price points compared to Mombasa, indicating differences in quality, demand, or export strategies.
- Seasonal fluctuations suggest the impact of harvest cycles and weather patterns.

### **4. Sugar Prices:**

- The disparity between sugar prices in the EU, US, and global markets underscores the role of trade policies, subsidies, and regional demand.
- While the global average exhibits relative stability, EU and US prices show greater variance, likely reflecting localized economic conditions.