Department of Computer Science and Engineering (Data Science)
S.Y. B.Tech. Sem: IV Subject: Statistics for Data Science
Experiment 0

Name: Yati Rathod **SAP ID: 60009230026**

Date:	**Experiment Title: Visualizing descriptive statistics**
Aim	To visualize descriptive statistics on data
Software	Google Colab
Implementation	1. Load the Dataset for Visualization (Kaggle Retail Dataset for Data visualization)

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

```
df = pd.read_excel("/content/drive/MyDrive/SDS/online_retail_II.xlsx")
```

2. Understand the dataset using methods like head, tail, describe, etc.

```
df.head()
#first 5 rows of dataset
```

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45:00	2.10	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom

```
df.describe()
#numerical information abt dataset
```

	Quantity	InvoiceDate	Price	Customer ID
count	525461.000000	525461	525461.000000	417534.000000
mean	10.337667	2010-06-28 11:37:36.845017856	4.688834	15360.645478
min	-9600.000000	2009-12-01 07:45:00	-53594.360000	12346.000000
25%	1.000000	2010-03-21 12:20:00	1.250000	13983.000000
50%	3.000000	2010-07-06 09:51:00	2.100000	15311.000000
75%	10.000000	2010-10-15 12:45:00	4.210000	16799.000000
max	19152.000000	2010-12-09 20:01:00	25111.090000	18287.000000
std	107.424110	NaN	146.126914	1680.811316

```
df.info()
#information abt dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 525461 entries, 0 to 525460
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Invoice      525461 non-null  object
 1   StockCode    525461 non-null  object
 2   Description  522533 non-null  object
 3   Quantity     525461 non-null  int64
 4   InvoiceDate  525461 non-null  datetime64[ns]
 5   Price        525461 non-null  float64
 6   Customer ID  417534 non-null  float64
 7   Country      525461 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 32.1+ MB
```

```
df.tail()
#last 5 rows of dataset
```

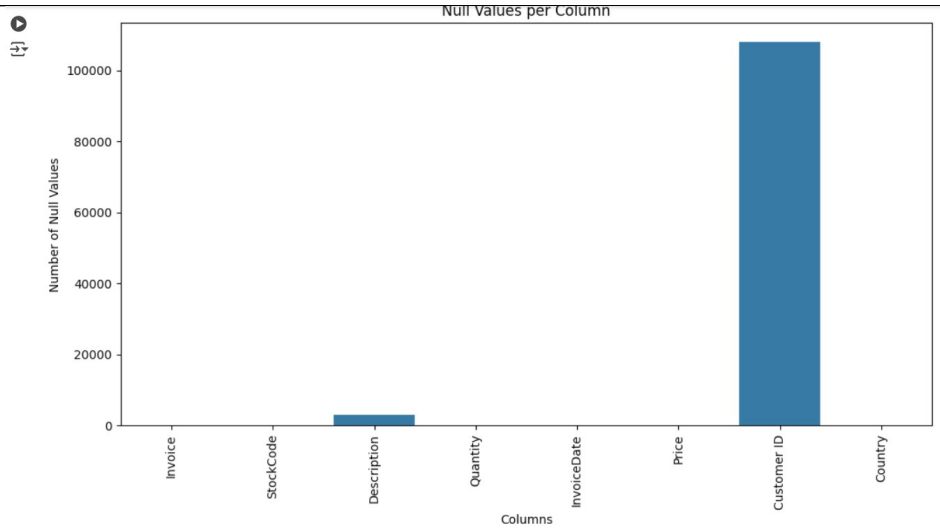
	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
525456	538171	22271	FELTCRAFT DOLL ROSIE	2	2010-12-09 20:01:00	2.95	17530.0	United Kingdom
525457	538171	22750	FELTCRAFT PRINCESS LOLA DOLL	1	2010-12-09 20:01:00	3.75	17530.0	United Kingdom
525458	538171	22751	FELTCRAFT PRINCESS OLIVIA DOLL	1	2010-12-09 20:01:00	3.75	17530.0	United Kingdom
525459	538171	20970	PINK FLORAL FELTCRAFT SHOULDER BAG	2	2010-12-09 20:01:00	3.75	17530.0	United Kingdom
525460	538171	21931	JUMBO STORAGE BAG SUKI	2	2010-12-09 20:01:00	1.95	17530.0	United Kingdom

3. Find out and plot Null values in the dataset

```
null_values = df.isnull().sum()
print(null_values)
```

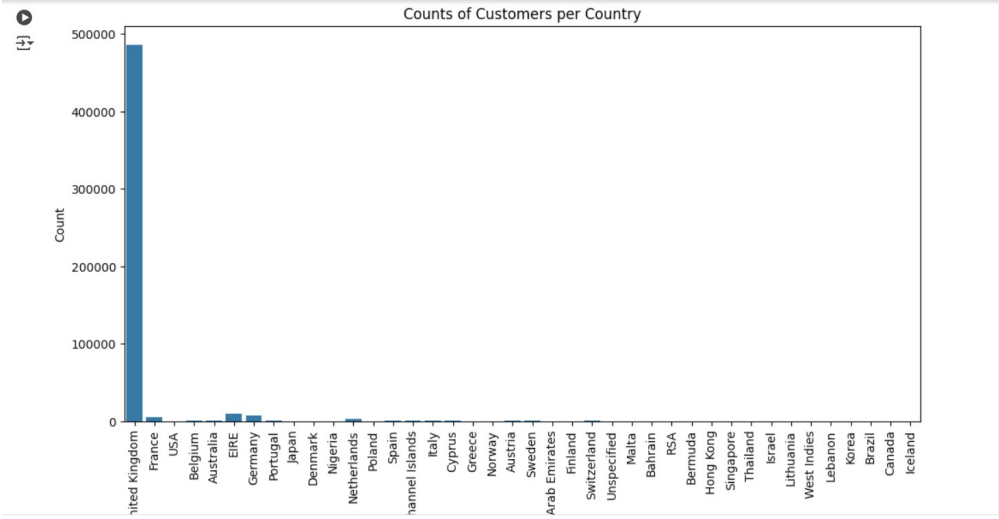
```
Invoice            0
StockCode          0
Description     2928
Quantity           0
InvoiceDate        0
Price              0
Customer ID    107927
Country            0
dtype: int64
```

```
plt.figure(figsize=(12, 6))
sns.barplot(x=null_values.index, y=null_values.values)
#.index for column name, .values for no. of null values
plt.xlabel("Columns")
plt.ylabel("Number of Null Values")
plt.title("Null Values per Column")
plt.show()
```



4. Plot counts (Bar plots) for categorical variables

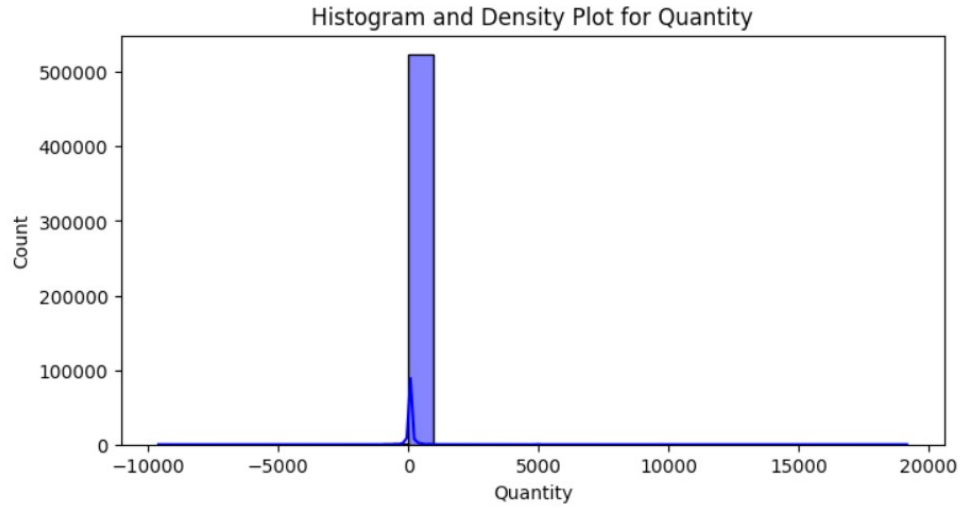
```
plt.figure(figsize=(12, 6))
sns.countplot(x='Country', data=df)
plt.xlabel("Country")
plt.ylabel("Count")
plt.title("Counts of Customers per Country")
plt.xticks(rotation=90) # Rotate x-axis labels for better readability
plt.show()
```

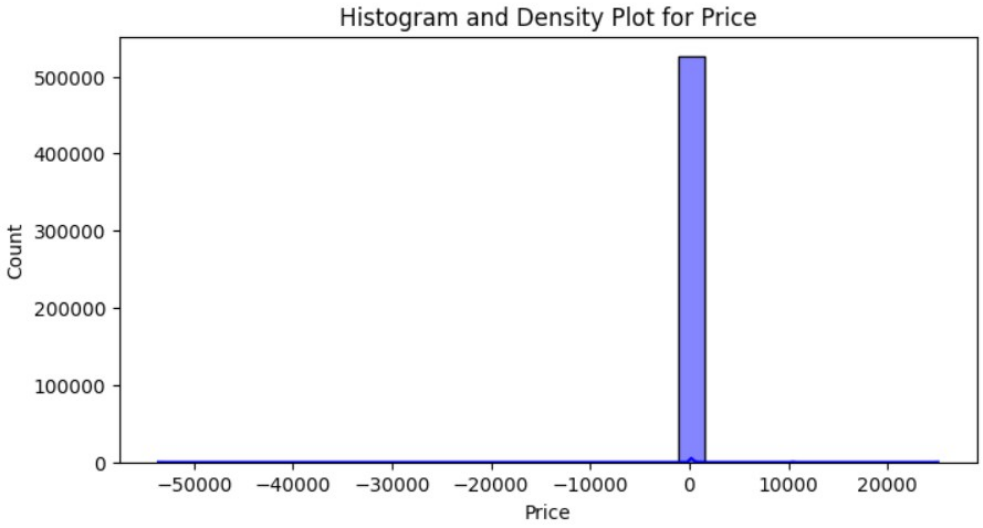


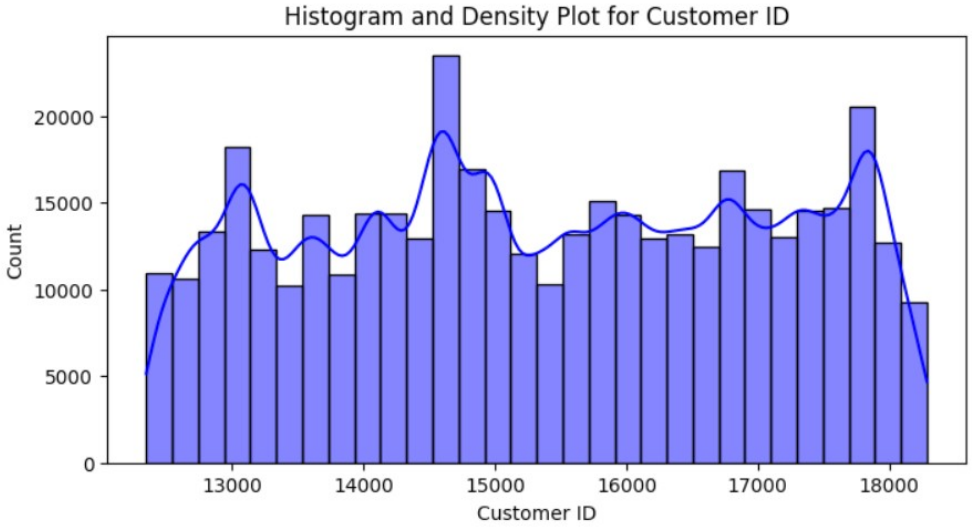
5. Plot Histograms or Density Plots

For numerical column, histograms or density plots would help to visualize the distribution of data. This gives an idea of the data spread, central tendency, and skewness.

```
numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns

for col in numerical_cols:
    plt.figure(figsize=(8, 4))
    sns.histplot(df[col], kde=True, bins=30, color='blue')
    plt.title(f"Histogram and Density Plot for {col}")
    plt.show()
```

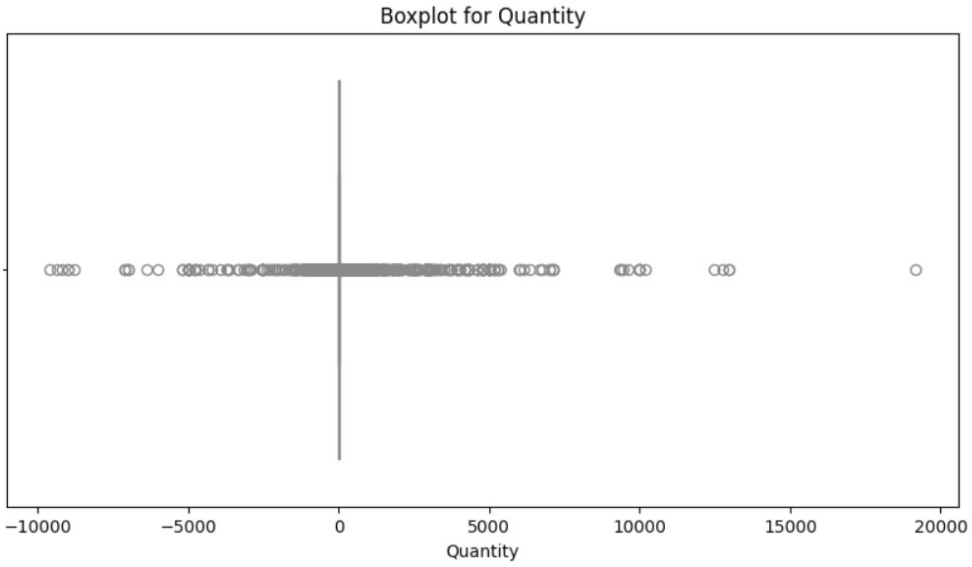


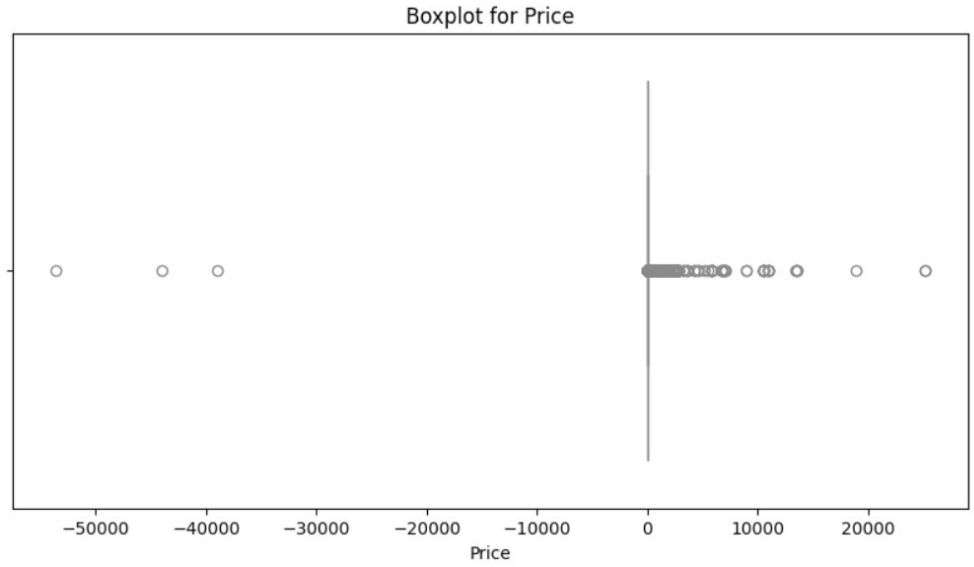


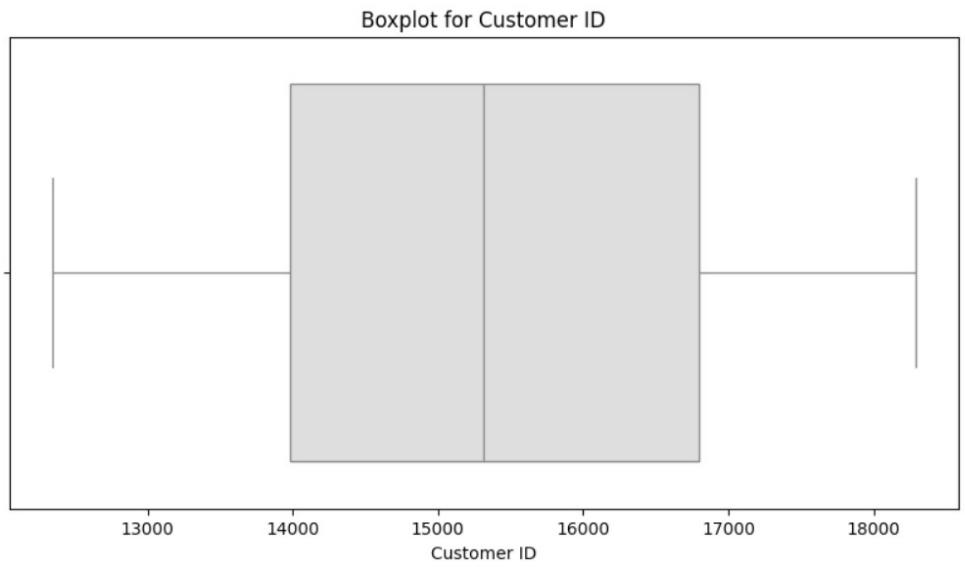


6. Boxplots- for visualizing the distribution of data in terms of quartiles. To identify outliers and compare distributions across different groups.

```
[ ]  numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns

     for col in numerical_cols:
         plt.figure(figsize=(10, 5))
         sns.boxplot(x=df[col], palette="coolwarm")
         plt.title(f"Boxplot for {col}")
         plt.show()
```





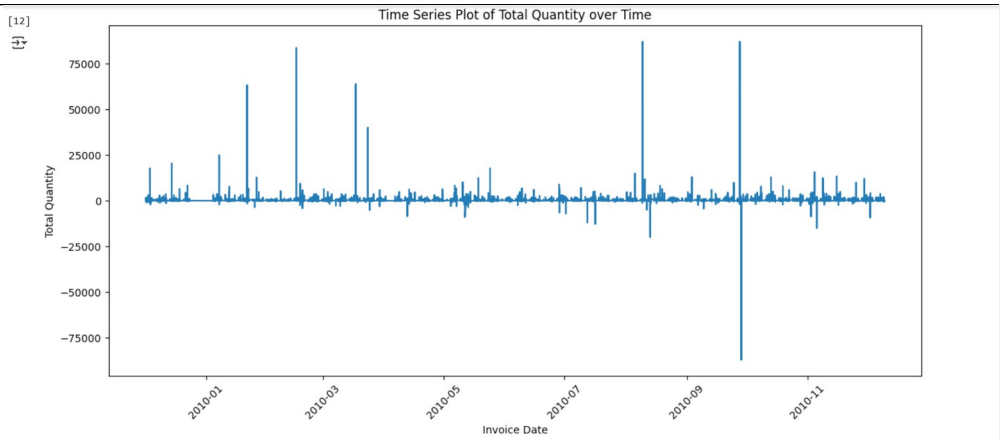


7. Time Series Plots: If your Data contains time series data, plot the time series and observe any trend.

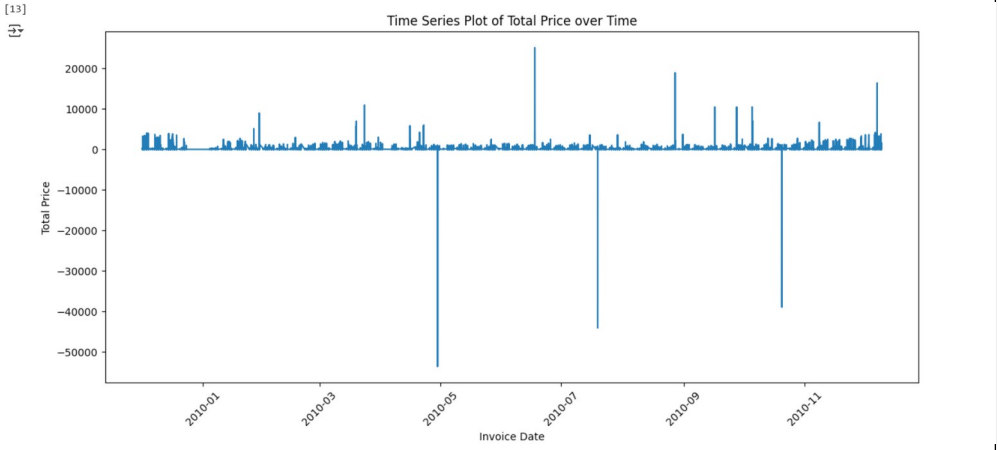
```
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
time_series_data = df.groupby('InvoiceDate')['Quantity'].sum()

plt.figure(figsize=(12, 6))
plt.plot(time_series_data.index, time_series_data.values)
plt.xlabel('Invoice Date')
plt.ylabel('Total Quantity')
plt.title('Time Series Plot of Total Quantity over Time')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
[13] df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])

     time_series_quantity = df.groupby('InvoiceDate')['Quantity'].sum()
     plt.figure(figsize=(12, 6))
     plt.plot(time_series_quantity.index, time_series_quantity.values)
     plt.xlabel('Invoice Date')
     plt.ylabel('Total Quantity')
     plt.title('Time Series Plot of Total Quantity over Time')
     plt.xticks(rotation=45)
     plt.tight_layout()
     plt.show()

     if 'Price' in df.columns and pd.api.types.is_numeric_dtype(df['Price']):
         time_series_price = df.groupby('InvoiceDate')['Price'].sum()
         plt.figure(figsize=(12, 6))
         plt.plot(time_series_price.index, time_series_price.values)
         plt.xlabel('Invoice Date')
         plt.ylabel('Total Price')
         plt.title('Time Series Plot of Total Price over Time')
         plt.xticks(rotation=45)
         plt.tight_layout()
         plt.show()
     else:
         print("'Price' column not found or not numeric. Cannot create the time series plot.")
```

[12]

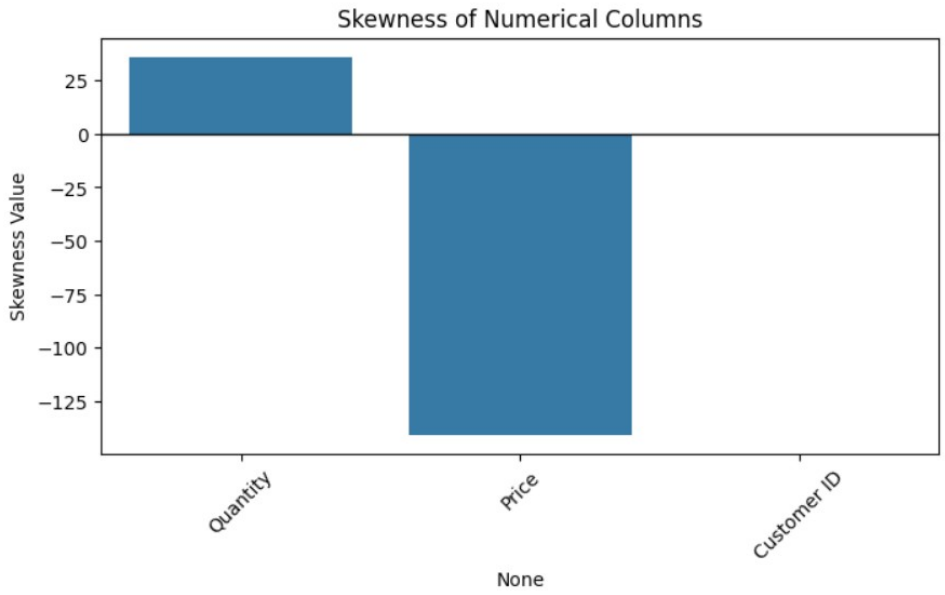


[13]



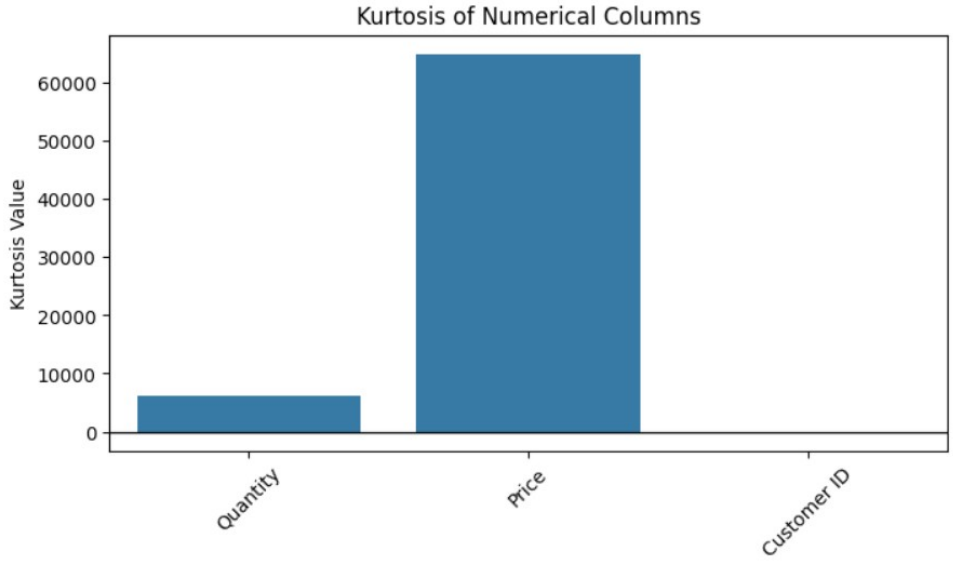
8. Plot skewness values for numerical columns using skew() function.
The skew() function can be used to calculate skewness in data. It represents the shape of the distribution. Skewness can be quantified to define the extent to which a distribution differs from a normal distribution.

```
skew_values = df.select_dtypes(include=['int64', 'float64']).skew()

plt.figure(figsize=(8, 4))
sns.barplot(x=skew_values.index, y=skew_values.values)
plt.xticks(rotation=45)
plt.title("Skewness of Numerical Columns")
plt.ylabel("Skewness Value")
plt.axhline(0, color='black', linewidth=1)  # Reference line for normal distribution
plt.show()
```



9. Plot kurtosis values for numerical columns using kurt() function.
The kurt() function can be used to calculate kurtosis in data. Kurtosis is the measure of thickness or heaviness of the distribution. It represents the height of the distribution.

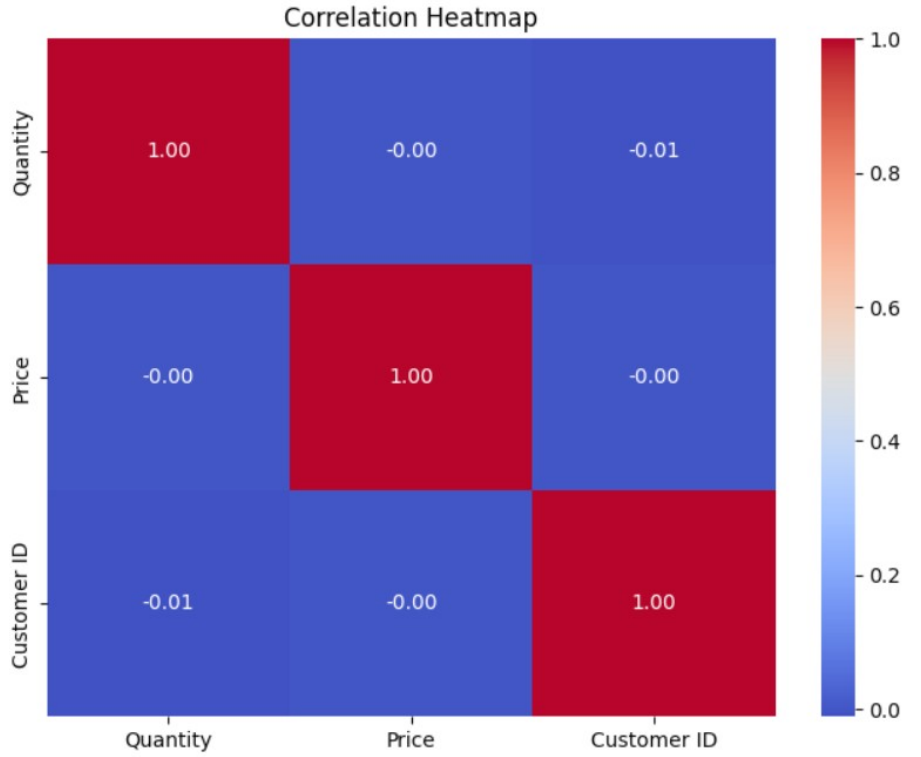
```
kurt_values = df.select_dtypes(include=['int64', 'float64']).kurt()

plt.figure(figsize=(8, 4))
sns.barplot(x=kurt_values.index, y=kurt_values.values)
plt.xticks(rotation=45)
plt.title("Kurtosis of Numerical Columns")
plt.ylabel("Kurtosis Value")
plt.axhline(0, color='black', linewidth=1)  # Reference line for normal distribution
plt.show()
```



10. Plot the corr() function using heatmap(): Perform on glue Dataset,l
The corr() used to find the pairwise correlation of all columns in the dataframe. Missing values excluded in the calculation. Correlation uncovers the complex and unknown relationships between the variables in the

dataset. The most common and default correlation coefficient is Pearson's correlation coefficient.

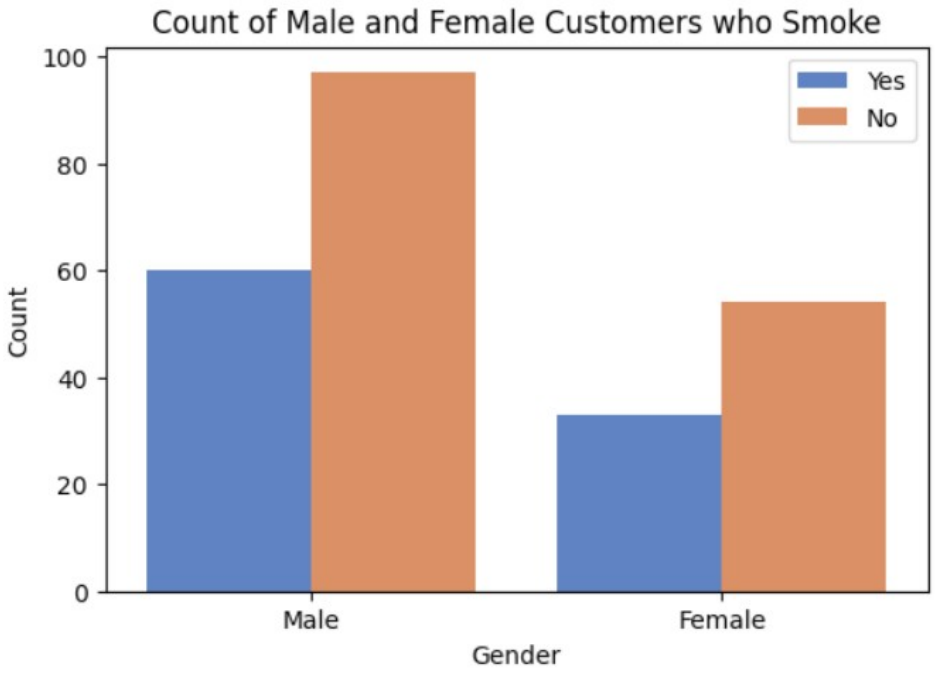
```
plt.figure(figsize=(8, 6))
correlation_matrix = numerical_cols = df.select_dtypes(include=['int64', 'float64']).corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```



11.factorplot():Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive.This method returns the FacetGrid object with the plot on it for further tweaking.

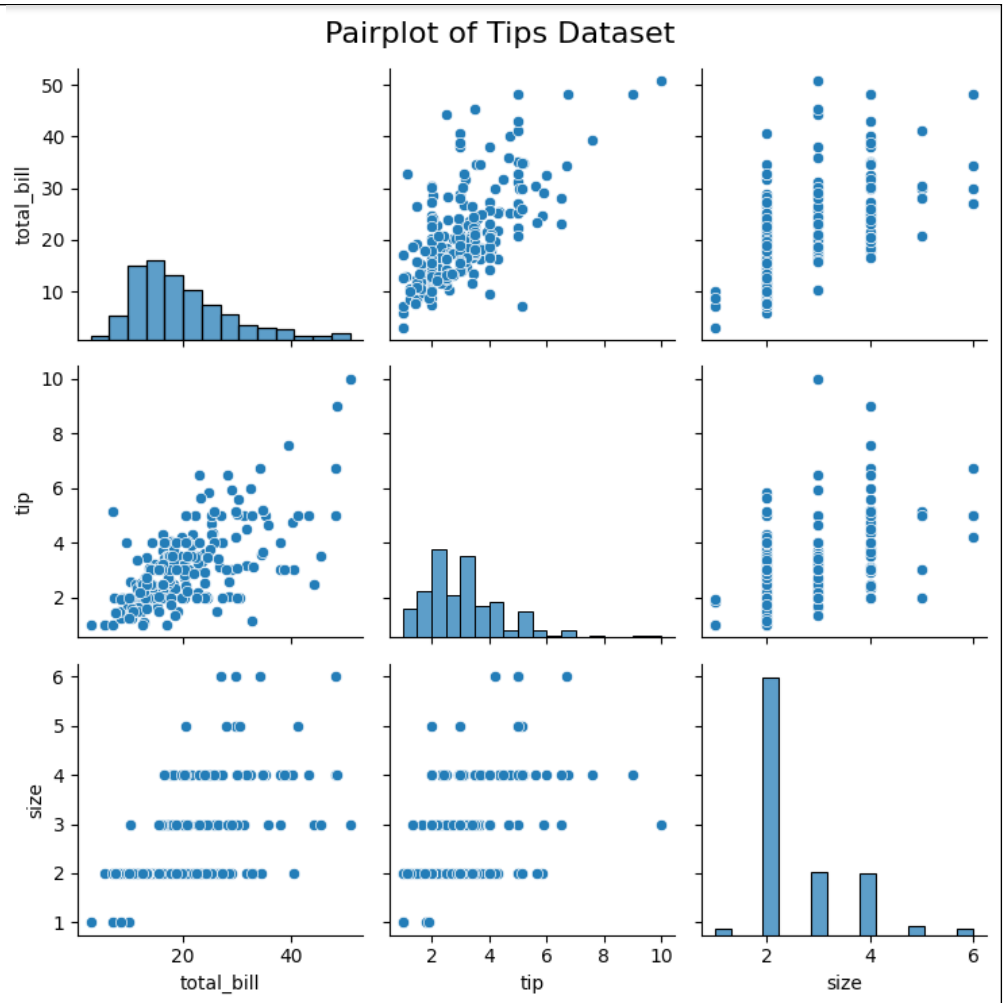
12. Countplot(): Perform operation on tips.csv file from seaborn library

```
plt.figure(figsize=(6,4))
df = sns.load_dataset('tips')
sns.countplot(x='sex', data=df, hue='smoker', palette='muted')
plt.title('Count of Male and Female Customers who Smoke')
plt.legend()
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()
```



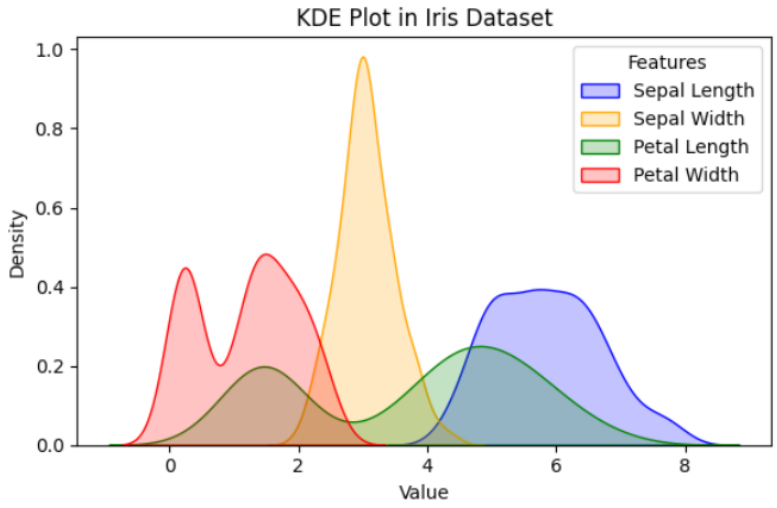
13 pairplot():This function helps you make a grid of plots where each row shares the same y-axis and each column shares the same x-axis. The plots on the diagonal (where the row and column are the same) show the distribution of just one variable. It is also possible to show a subset of variables or plot different variables on the rows and columns.

```
plt.figure(figsize=(6,4))
df = sns.load_dataset('tips')
sns.pairplot(df)
plt.suptitle('Pairplot of Tips Dataset', fontsize=16)
plt.tight_layout()
plt.show()
```

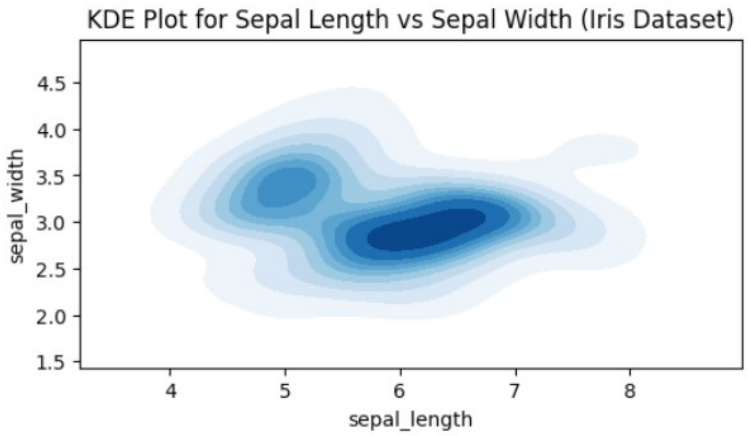


14. Kdeplot(): perform on Iris Dataset and geyser dataset

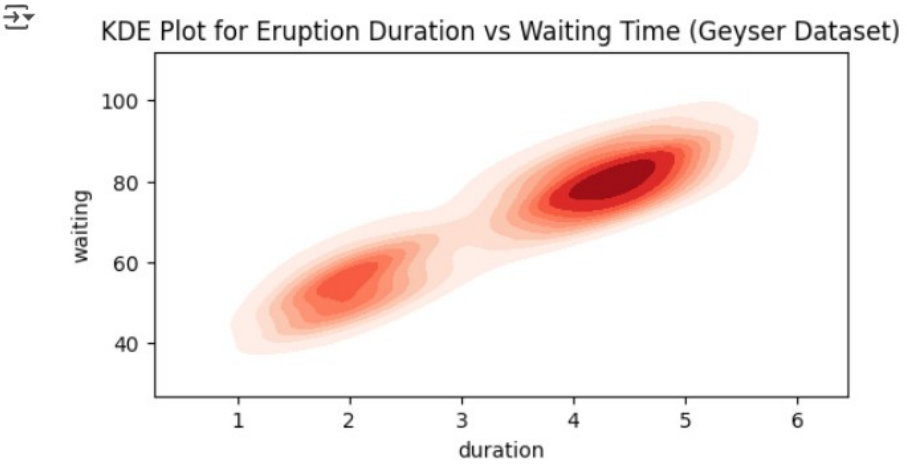
```
plt.figure(figsize=(6,4))
iris = sns.load_dataset('iris')
sns.kdeplot(data=iris, x='sepal_length', fill=True, color='blue', label='Sepal Length')
sns.kdeplot(data=iris, x='sepal_width', fill=True, color='orange', label='Sepal Width')
sns.kdeplot(data=iris, x='petal_length', fill=True, color='green', label='Petal Length')
sns.kdeplot(data=iris, x='petal_width', fill=True, color='red', label='Petal Width')
plt.title('KDE Plot in Iris Dataset')
plt.xlabel('Value')
plt.ylabel('Density')
plt.legend(title='Features')
plt.tight_layout()
plt.show()
```



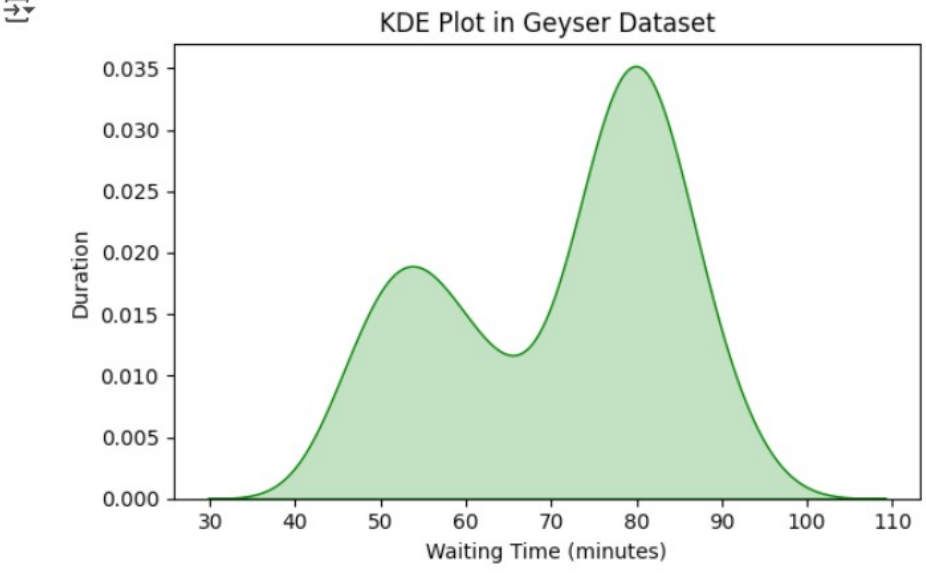

```
plt.figure(figsize=(6, 3))
sns.kdeplot(data=iris, x="sepal_length", y="sepal_width", cmap="Blues", fill=True)
plt.title("KDE Plot for Sepal Length vs Sepal Width (Iris Dataset)")
plt.show()
```



```
geyser = sns.load_dataset("geyser")

plt.figure(figsize=(6, 3))
sns.kdeplot(data=geyser, x="duration", y="waiting", cmap="Reds", fill=True)
plt.title("KDE Plot for Eruption Duration vs Waiting Time (Geyser Dataset)")
plt.show()
```



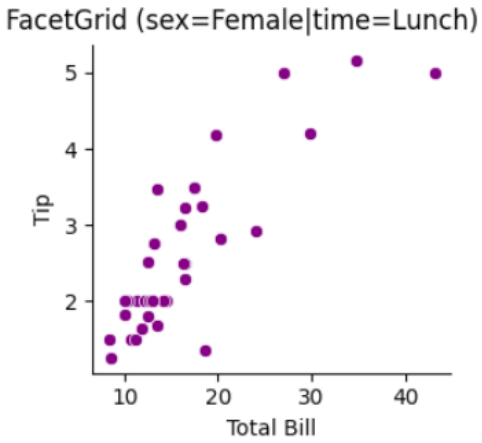

```
plt.figure(figsize=(6,4))
geyser = sns.load_dataset('geyser')
sns.kdeplot(data=geyser, x='waiting', fill=True, color='green')
plt.title('KDE Plot in Geyser Dataset')
plt.xlabel('Waiting Time (minutes)')
plt.ylabel('Duration')
plt.tight_layout()
plt.show()
```



15. FacetGrid():This class maps a dataset onto multiple axes arrayed in a grid of rows and columns that correspond to levels of variables in the dataset. The plots it produces are often called “lattice”, “trellis”, or “small-multiple” graphics.

```
[73] plt.figure(figsize=(6,4))
     tips = sns.load_dataset('tips')
     filtered_tips = tips[(tips['sex'] == 'Female') & (tips['time'] == 'Lunch')]
     g = sns.FacetGrid(filtered_tips)
     g.map(sns.scatterplot, 'total_bill', 'tip', color='purple')
     g.set_axis_labels('Total Bill', 'Tip')
     g.fig.suptitle('FacetGrid (sex=Female|time=Lunch)')
     g.tight_layout()
     g.fig.subplots_adjust(top=0.9)
     plt.show()
```

```
<Figure size 600x400 with 0 Axes>
```



Conclusion	Descriptive Statistics, combined with Python's powerful libraries, provides a foundational understanding of data. By mastering these measures and tools, you can effectively summarize, visualize, and draw valuable insights from your datasets.

Colab link:

https://colab.research.google.com/github/SmayanKulkarni/AI-and-ML-Course/blob/master/SDS/Descriptive.ipynb

Signature of Faculty