Practical No.1

Write a python program to perform data preprocessing and cleaning tasks on a dataset using the NumPy library

Aim: Perform basic data preprocessing and cleaning tasks on a dataset using the NumPy library.

Code:

```
import numpy as np
# Example dataset: rows represent data points, columns are features
data = np.array([
 [np.nan, 7, np.nan],
 [8, np.nan, 6],
 [2, 9, 9],
 [4, np.nan, 6], # Duplicate row
 [np.nan, np.nan, 9],
])
print("Original Data:")
print(data)
# 1. Handle missing values (e.g., replace with mean of the column)
col_means = np.nanmean(data, axis=0)
indices = np.where(np.isnan(data))
data[indices] = np.take(col_means, indices[1])
print("\nData after handling missing values (replacing with column means):")
print(data)
# 2. Remove duplicate rows
data_unique = np.unique(data, axis=0)
print("\nData after removing duplicates:")
print(data_unique)
# 3. Normalize data (scale to 0-1 range)
data_min = np.min(data_unique, axis=0)
data_max = np.max(data_unique, axis=0)
normalized_data = (data_unique - data_min) / (data_max - data_min)
print("\nNormalized Data (scaled to 0-1):")
print(normalized_data)
```

Output:

```
Original Data:
[[nan  7. nan]
 [ 8. nan  6.]
 [ 2.  9.  9.]
 [ 4. nan  6.]
 [nan nan  9.]]

Data after handling missing values (replacing with column means):
[[4.66666667 7.         7.5       ]
 [8.         8.         6.        ]
 [2.         9.         9.        ]
 [4.         8.         6.        ]
 [4.66666667 8.         9.        ]]

Data after removing duplicates:
[[2.         9.         9.        ]
 [4.         8.         6.        ]
 [4.66666667 7.         7.5       ]
 [4.66666667 8.         9.        ]
 [8.         8.         6.        ]]

Normalized Data (scaled to 0-1):
[[0.         1.         1.        ]
 [0.33333333 0.5        0.        ]
 [0.44444444 0.         0.5       ]
 [0.44444444 0.5        1.        ]
 [1.         0.5        0.        ]]
```

Practical No.2

Write a python program to perform statistical analysis technique on a dataset using the Pandas library

Aim: Calculate central tendency measures such as mean, median, and mode

Code:

```
import pandas as pd
# Real-life example: Exam scores of 10 students in three subjects
data = {
 'Student': ['Deepak', 'Anish', 'Harsh', 'Sahil', 'Prathmesh', 'OM ', 'Sarthak ', 'Helen', 'Ian', 'Jack'],
 'Math': [85, 68, 77, 87, 90, 79, 80, 99, 70, 76],
 'Science': [90, 88, 83, 98, 94, 78, 83, 81, 89, 98],
 'English': [88, 79, 85, 86, 93, 77, 84, 91, 79, 88]
}
# Create a DataFrame
df = pd.DataFrame(data)
# Display the original dataset
print("Original Data:")
print(df)
# 1. Mean Calculation: The average score in each subject
mean_scores = df[['Math', 'Science', 'English']].mean()
print("\nMean Scores (Average):")
print(mean_scores)
# 2. Median Calculation: The middle value of scores in each subject
median_scores = df[['Math', 'Science', 'English']].median()
print("\nMedian Scores:")
print(median_scores)
# 3. Mode Calculation: The most frequent score in each subject
mode_scores = df[['Math', 'Science', 'English']].mode().iloc[0] # .iloc[0] to get the first mode if multiple modes exist
print("\nMode Scores:")
print(mode_scores)
```

Output:

```
Original Data:
     Student  Math  Science  English
0     Deepak    85       90       88
1      Anish    68       88       79
2      Harsh    77       83       85
3      Sahil    87       98       86
4  Prathmesh    90       94       93
5         OM    79       78       77
6   Sarthak     80       83       84
7      Helen    99       81       91
8        Ian    70       89       79
9       Jack    76       98       88

Mean Scores (Average):
Math       81.1
Science    88.2
English    85.0
dtype: float64

Median Scores:
Math       79.5
Science    88.5
English    85.5
dtype: float64

Mode Scores:
Math       68.0
Science    83.0
English    79.0
Name: 0, dtype: float64
```

Practical No.3

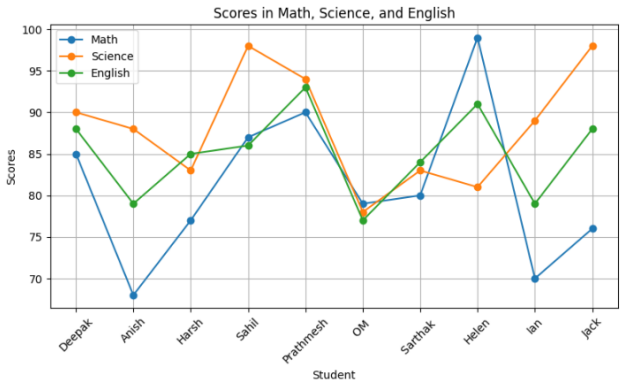
Write a python program to perform basic visualization using the matplotlib

Aim: To perform a basic visualization using the matplotib library

Code:

```
import matplotlib.pyplot as plt
import numpy as np
# Example dataset: Students' exam scores in Math, Science, and English
students = ['Alice', 'Bob', 'Charlie', 'David', 'Eva', 'Frank', 'Grace', 'Helen', 'Ian', 'Jack']
math_scores = [85, 78, 92, 88, 95, 73, 81, 89, 76, 91]
science_scores = [90, 85, 87, 91, 94, 78, 82, 88, 80, 92]
english_scores = [88, 79, 85, 86, 93, 77, 84, 91, 79, 88]
# Line Plot (Math, Science, English Scores Over Students)
plt.figure(figsize=(8, 5))
plt.plot(students, math_scores, label='Math', marker='o')
plt.plot(students, science_scores, label='Science', marker='o')
plt.plot(students, english_scores, label='English', marker='o')
plt.title('Scores in Math, Science, and English')
plt.xlabel('Students')
plt.ylabel('Scores')
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Output:



Practical No.4

Write a python program to perform visualization of frequency curve using the seaborn

Aim: To plot frequency curve using csv dataset.

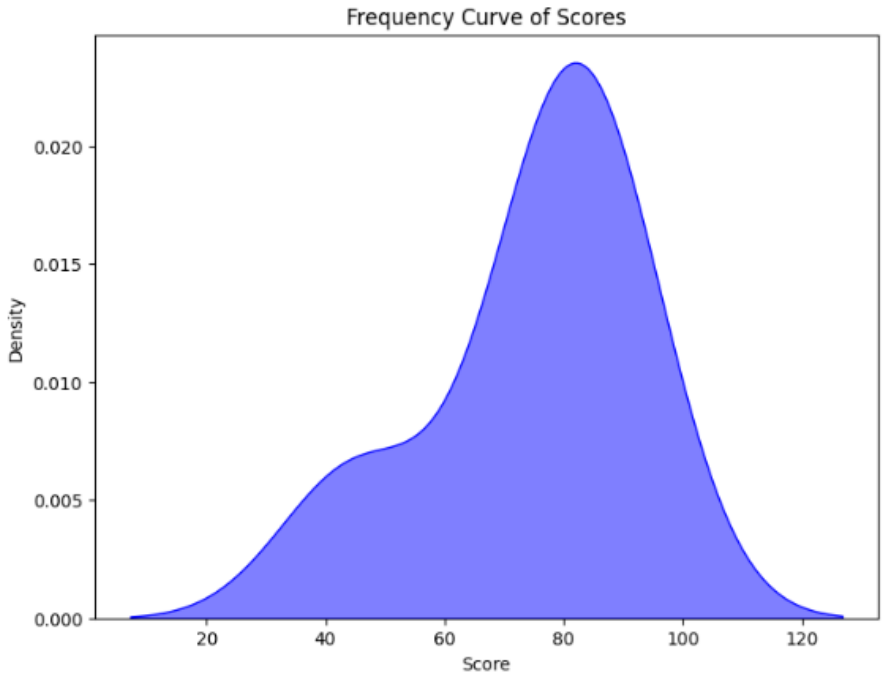
Code:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
# Create a sample dataset on education
data = {
 "Student_ID": [1, 2, 3, 4, 5],
 "Name": ["Deepak", "Anish", "Parthmesh", "Sahil", "Pooja"],
 "Age": [15, 16, 15, 17, 16],
 "Gender": ["Male", "Male", "Male", "Male", "Female"],
 "Grade_Level": ["10th", "12th", "10th", "11th", "11th"],
 "Attendance_Rate": [90, 78, 77, 70, 99],
 "Subject": ["Math", "Science", "Math", "History", "Science"],
 "Score": [89, 77, 45, 78, 85],
 "Socioeconomic_Status": ["Middle", "High", "Middle", "Low", "High"],
 "Hours_Studied": [8, 4, 6, 3, 5],
 "Parental_Involvement": ["Medium", "Low", "Medium", "Low", "Medium"]
}
# Create DataFrame
df = pd.DataFrame(data)
# Save DataFrame to CSV
csv_file_path = "education_dataset.csv"
df.to_csv(csv_file_path, index=False)
print(f"CSV file '{csv_file_path}' created successfully!")
# Load the dataset from the CSV file
df_loaded = pd.read_csv(csv_file_path)
# Create a KDE plot of scores
plt.figure(figsize=(8, 6))
sns.kdeplot(data=df_loaded, x='Score', fill=True, color='blue', alpha=0.5)
# Add labels and title
```

```
plt.xlabel("Score")
plt.ylabel("Density")
plt.title("Frequency Curve of Scores")
# Display the plot
plt.show()
```

Output:

```
CSV file 'education_dataset.csv' created successfully!
```



Practical No.5

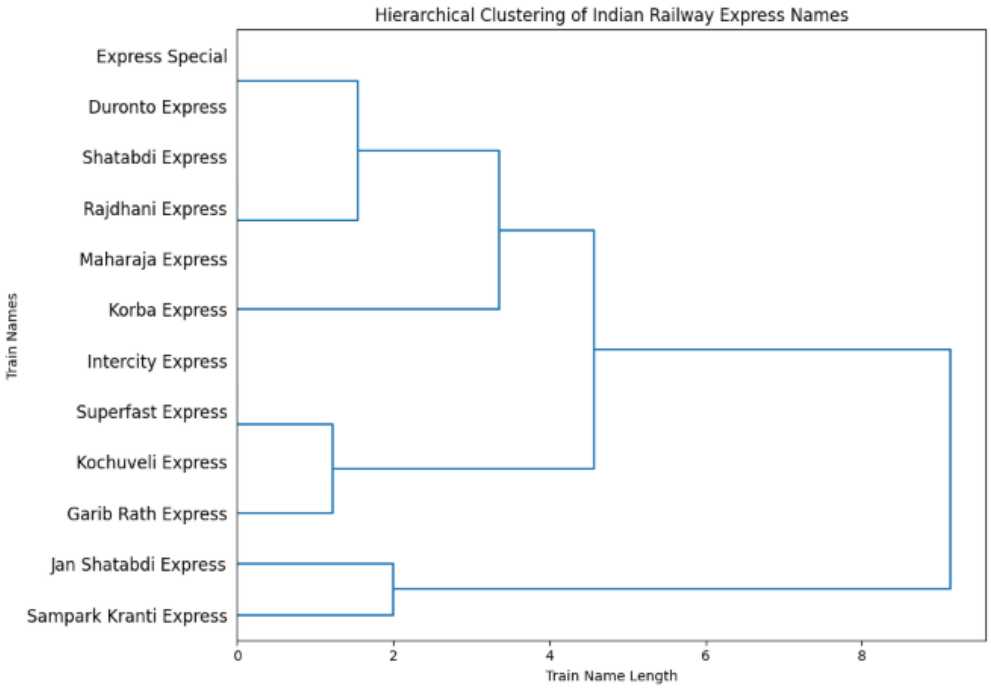
Write a python program to perform hierarchal clustering using the scipy

Aim: To use the SciPy library to analyze a dataset of Indian Railway Express names.

Code:

```
import numpy as np
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt
# Example dataset: Indian Railway Express names (fictional)
train_names = [
 "Rajdhani Express", "Shatabdi Express", "Duronto Express", "Sampark Kranti Express",
 "Jan Shatabdi Express", "Maharaja Express", "Garib Rath Express", "Express Special",
 "Superfast Express", "Intercity Express", "Korba Express", "Kochuveli Express"
]
# 1. Convert train names to a simple numeric feature representation
# We will use the length of each train name as a feature for simplicity
train_name_lengths = np.array([len(name) for name in train_names]).reshape(-1, 1)
# 2. Perform hierarchical clustering on the train name lengths
Z = linkage(train_name_lengths, method='ward')
# 3. Create a dendrogram to visualize the clustering
plt.figure(figsize=(10, 7))
dendrogram(Z, labels=train_names, orientation='right', color_threshold=0)
plt.title('Hierarchical Clustering of Indian Railway Express Names')
plt.xlabel('Train Name Length')
plt.ylabel('Train Names')
plt.tight_layout()
plt.show()
```

Output:


Hierarchical Clustering of Indian Railway Express Names
Express Special
Duronto Express
Shatabdi Express
Rajdhani Express
Maharaja Express
Korba Express
Intercity Express
Superfast Express
Kochuveli Express
Garib Rath Express
Jan Shatabdi Express
Sampark Kranti Express
Train Names
0
2
4
6
8
Train Name Length

Practical No.6

Write a python program to perform EDA on real time data

Aim: The aim of this program is to perform Exploratory Data Analysis (EDA) on a dataset related to agriculture in India.

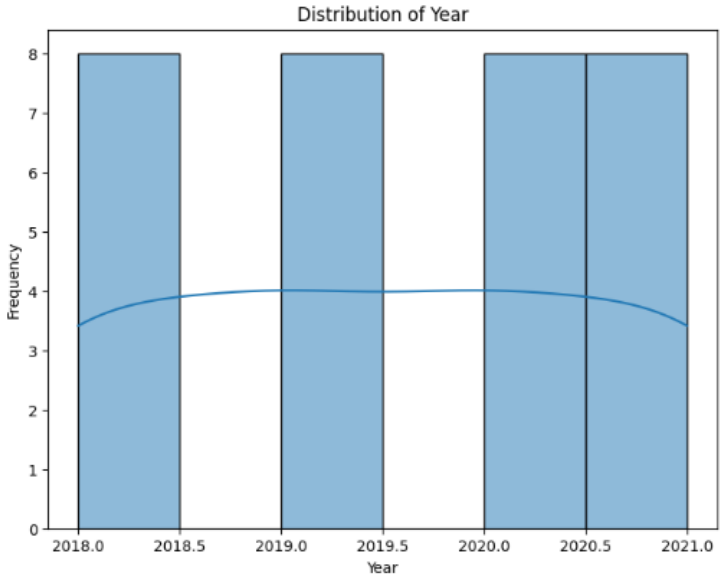
Code:

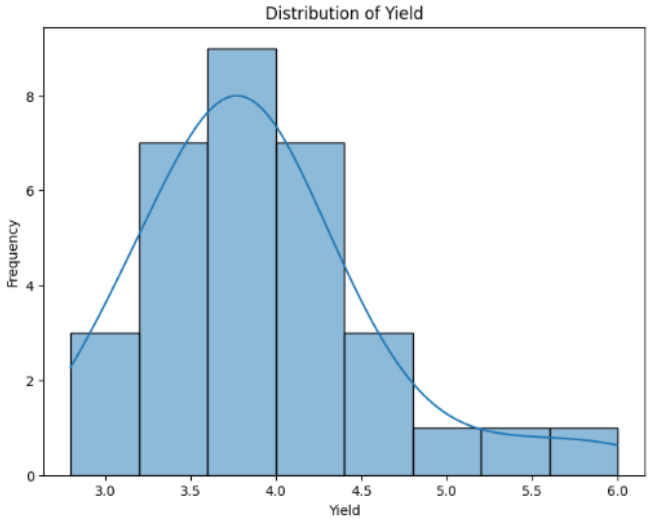
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Load the dataset (replace with actual dataset path)
file_path = r"C:\Users\Admin\Desktop\MSc Sem 3\Data Science Using Python\agriculture_yield_year.csv" # Update with the correct file path
data = pd.read_csv(file_path)
# 1. Data Overview
print("First few rows of the dataset:")
print(data.head())
# 2. Check for missing values
missing_values = data.isnull().sum()
print("\nMissing values in each column:")
print(missing_values)
# 3. Summary Statistics for Numeric Columns
print("\nSummary Statistics:")
print(data.describe())
# 4. Distribution of Numeric Columns
numeric_columns = data.select_dtypes(include=['number']).columns
for column in numeric_columns:
 plt.figure(figsize=(8, 6))
 sns.histplot(data[column], kde=True)
 plt.title(f'Distribution of {column}')
 plt.xlabel(column)
 plt.ylabel('Frequency')
 plt.show()
# 5. Correlation Heatmap for Numeric Columns (only numeric columns)
numeric_data = data.select_dtypes(include=['number']) # Selecting only numeric columns
correlation_matrix = numeric_data.corr()
```

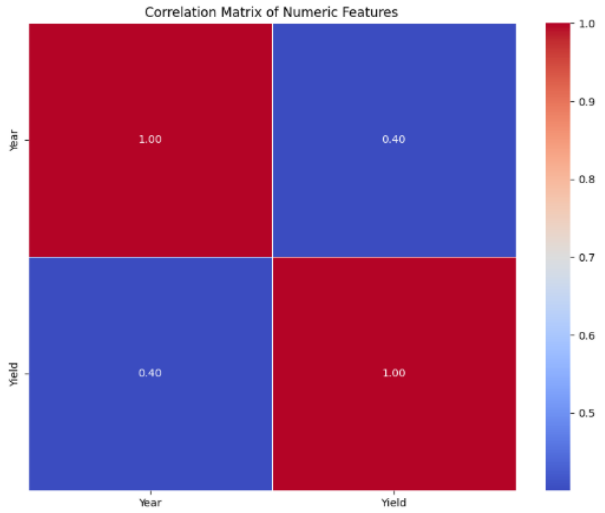
```
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title("Correlation Matrix of Numeric Features")
plt.show()
# 6. Visualizing Categorical Columns (If applicable)
categorical_columns = data.select_dtypes(include=['object']).columns
for column in categorical_columns:
 plt.figure(figsize=(10, 6))
 sns.countplot(data=data, x=column)
 plt.title(f'Count Plot of {column}')
 plt.xlabel(column)
 plt.ylabel('Count')
 plt.xticks(rotation=45)
 plt.show()
```

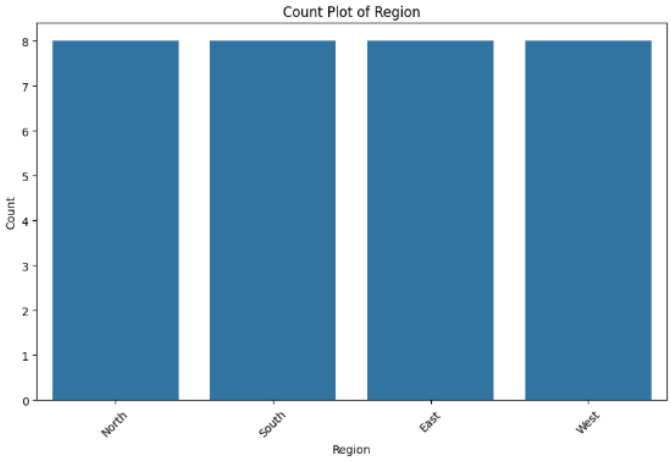
Output:

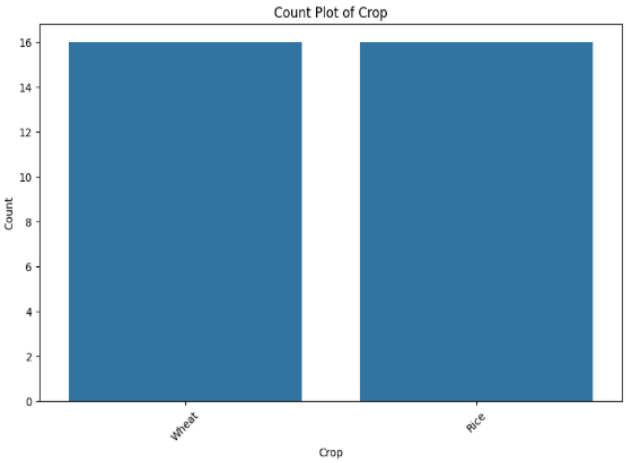
```
First few rows of the dataset:
   Year Region   Crop  Yield
0  2018  North  Wheat    4.5
1  2018  South  Wheat    3.2
2  2018   East  Wheat    4.0
3  2018   West  Wheat    3.8
4  2018  North   Rice    3.0

Missing values in each column:
Year      0
Region    0
Crop      0
Yield     0
dtype: int64

Summary Statistics:
              Year      Yield
count    32.000000  32.000000
mean   2019.500000   3.903125
std       1.135924   0.691648
min    2018.000000   2.800000
25%    2018.750000   3.500000
50%    2019.500000   3.800000
75%    2020.250000   4.200000
max    2021.000000   6.000000
```




Distribution of Yield
Frequency
Yield
0
2
4
6
8
3.0
3.5
4.0
4.5
5.0
5.5
6.0


Correlation Matrix of Numeric Features
Year
Yield
1.00
0.40
0.40
1.00
1.0
0.9
0.8
0.7
0.6
0.5


Count Plot of Region
Count
Region
North
South
East
West


Count Plot of Crop
Count
Crop
Wheat
Rice

Practical No.7

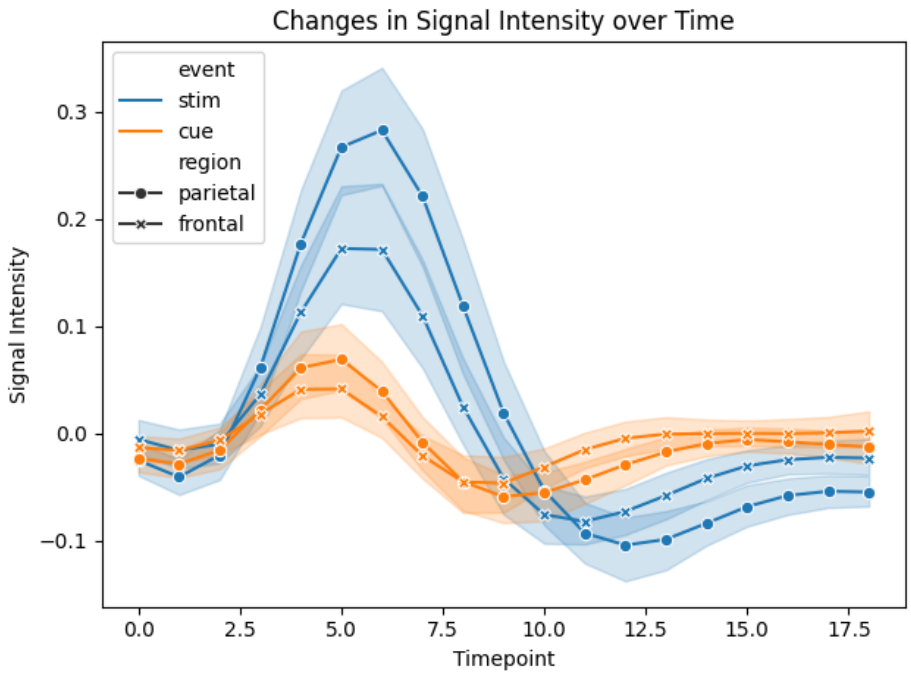
Write a python program to perform advance visualization on real time data

Aim: Performing a advance level plotting seaborn and matplotlib library.

Code:

```
import seaborn as sns
import matplotlib.pyplot as plt
fmri = sns.load_dataset("fmri")
# cusomize the line plot
sns.lineplot(x="timepoint", y="signal", hue="event", style="region", markers=True, dashes=False, data=fmri)
# add labels and title
plt.xlabel("Timepoint")
plt.ylabel("Signal Intensity")
plt.title("Changes in Signal Intensity over Time")
# display the plot
plt.show()
```

Output:



Practical No.8

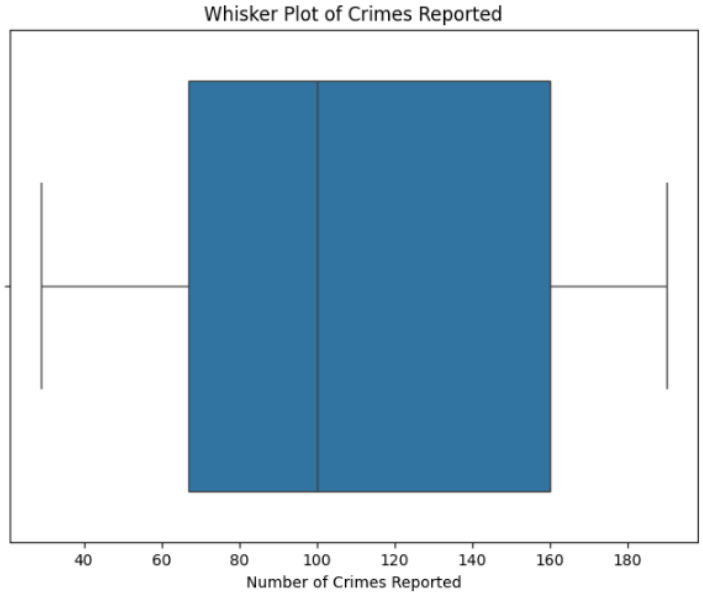
Write a python program to perform whiskers Plot.

Aim: To plot a simple whisker plot.

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Example Data (replace with your own dataset)
data = {
 'Crime_Reported': [29, 45, 160, 90, 56, 150, 70, 190, 120, 178, 189, 67, 100]
}
# Create DataFrame
df = pd.DataFrame(data)
# Plotting the Whisker Plot (Boxplot)
plt.figure(figsize=(8, 6))
sns.boxplot(x=df['Crime_Reported'])
# Add title and labels
plt.title('Whisker Plot of Crimes Reported')
plt.xlabel('Number of Crimes Reported')
# Show the plot
plt.show()
```

Output:



Practical No.9

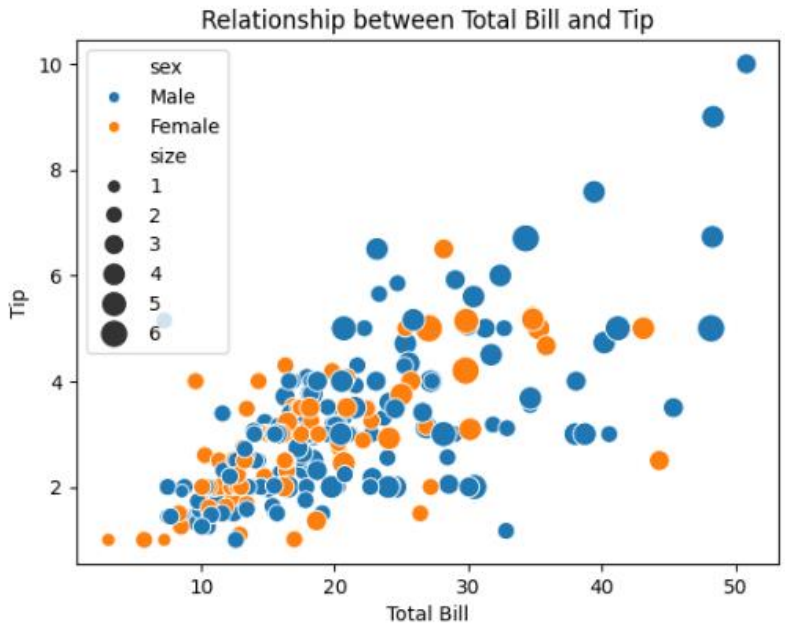
Write a python program to perform scatter plot using csv file.

Aim: To implement scatter plot using csv dataset

Code:

```
import seaborn as sns
import matplotlib.pyplot as plt
tips = sns.load_dataset("tips")
# customize the scatter plot
sns.scatterplot(x="total_bill", y="tip", hue="sex", size="size", sizes=(50, 200), data=tips)
# add labels and title
plt.xlabel("Total Bill")
plt.ylabel("Tip")
plt.title("Relationship between Total Bill and Tip")
# display the plot
plt.show()
```

Output:



Practical No.10

Write a python programs using seaborn library to plot line chart of a csv dataset.

Aim: To plot a line chart of csv dataset using seaborn.

Code:

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Load the dataset (make sure the CSV file is in the same directory or adjust the path accordingly)
df = pd.read_csv(r"C:\Users\Admin\Downloads\agriculture_yield_data.csv")
# Set the style for the plots
sns.set_style("whitegrid")
# Create a line plot showing yield trends over the years for each region
plt.figure(figsize=(10, 6))
sns.lineplot(data=df, x="Year", y="Yield", hue="Region", marker="o")
plt.title("Crop Yield Trends by Region (2018-2021)")
plt.ylabel("Yield (tons/ha)")
plt.xlabel("Year")
plt.legend(title="Region")
plt.show()
```

Output: