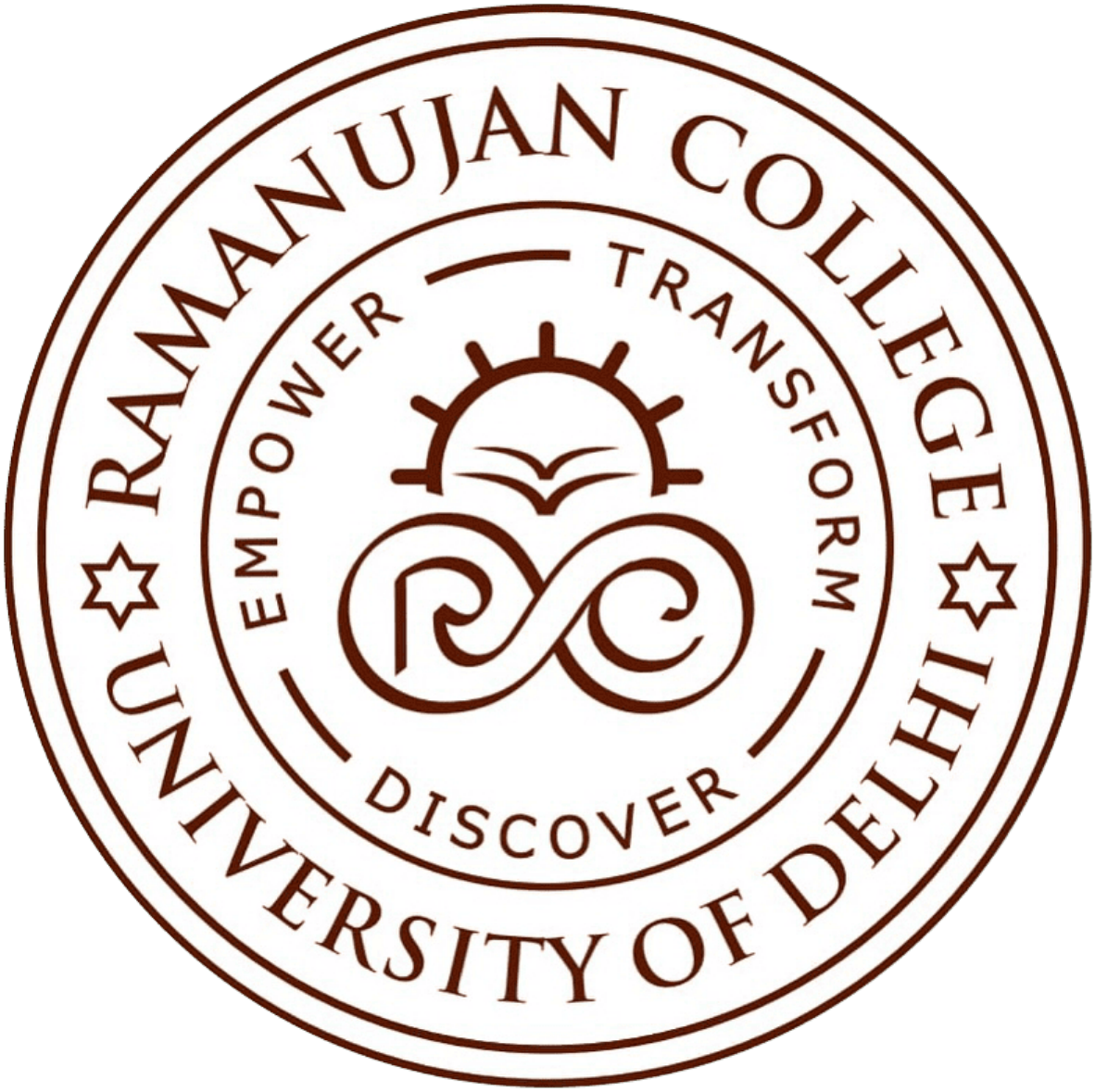

Name:-Dhiru kumar
Roll no:-20231411
Submitted to:- Dr Arun Agarwal
Subject:-DSE(Data Analysis and Visualization project.)

Acknowledgment

I would like to express my heartfelt gratitude to all those who have contributed to the success of my Data Analysis and Visualization project.
First and foremost, I would like to thank my supervisor, Dr Aurn Agarwal, for their invaluable guidance, support, and encouragement throughout this project. Their expertise and constructive feedback were crucial in refining my analysis and enhancing the quality of my visualizations.
I am also grateful to my colleagues and fellow students in the Computer Branch for their collaboration and insightful discussions. Their diverse perspectives and shared knowledge greatly enriched my understanding of data analysis techniques and visualization tools.
I would like to extend my appreciation to the Ramanujan College for providing access to the necessary resources, datasets, and software tools that facilitated my analysis. The supportive environment fostered by the institution was instrumental in my learning process.

Q1.
Write programs in Python using NumPy library to do the following:
a. Create a two dimensional array, ARR1 having random values from 0 to 1. Compute the mean, standard deviation, and variance of ARR1 along the second axis.

b. Create a 2-dimensional array of size m x n integer elements, also print the shape, type and data type of
the array and then reshape it into an n x m array, where n and m are user inputs given at the run time.

c. Test whether the elements of a given 1D array are zero, non-zero and NaN. Record the indices of these
elements in three separate arrays.

d. Create three random arrays of the same size: Array1, Array2 and Array3. Subtract Array 2 from Array3
and store in Array4. Create another array Array5 having two times the values in Array1. Find Co-
variance and Correlation of Array1 with Array4 and Array5 respectively.

e. Create two random arrays of the same size 10: Array1, and Array2. Find the sum of the first half of both
the arrays and product of the second half of both the arrays.

f. g. Create an array with random values. Determine the size of the memory occupied by the array.
Create a 2-dimensional array of size m x n having integer elements in the range (10,100). Write
statements to swap any two rows, reverse a specified column and store updated array in another
variable

1sol:-

```
import numpy as np

# Task 1a: Create a 2D array with random values and compute statistics
# Create a 2D array ARR1 with random values from 0 to 1
ARR1 = np.random.rand(5, 5)  # Example size 5x5
print("ARR1:\n", ARR1)

# Compute mean, standard deviation, and variance along the second axis (axis=1)
mean_arr1 = np.mean(ARR1, axis=1)
std_arr1 = np.std(ARR1, axis=1)
var_arr1 = np.var(ARR1, axis=1)

print("Mean along second axis:", mean_arr1)
print("Standard Deviation along second axis:", std_arr1)
```

```
print("Variance along second axis:", var_arr1)

# Task 1b: Create a 2D array of size m x n and reshape it
m = int(input("Enter number of rows (m): "))
n = int(input("Enter number of columns (n): "))
array2D = np.random.randint(0, 100, size=(m, n))  # Random integers from 0 to 100
print("Original Array:\n", array2D)

# Print shape, type, and data type of the array
print("Shape:", array2D.shape)
print("Type:", type(array2D))
print("Data Type:", array2D.dtype)

# Reshape it into an n x m array
reshaped_array = array2D.reshape(n, m)
print("Reshaped Array (n x m):\n", reshaped_array)

# Task 1c: Test for zero, non-zero, and NaN in a 1D array
array1D = np.random.randn(10)  # Random 1D array
print("1D Array:", array1D)

# Record indices of zero, non-zero, and NaN elements
zero_indices = np.where(array1D == 0)[0]
non_zero_indices = np.where(array1D != 0)[0]
nan_indices = np.where(np.isnan(array1D))[0]

print("Zero Indices:", zero_indices)
print("Non-Zero Indices:", non_zero_indices)
print("NaN Indices:", nan_indices)

# Task 1d: Create three random arrays and perform operations
Array1 = np.random.rand(10)
Array2 = np.random.rand(10)
Array3 = np.random.rand(10)

print("Array1:", Array1)
print("Array2:", Array2)
print("Array3:", Array3)

# Subtract Array2 from Array3 and store in Array4
Array4 = Array3 - Array2

# Create Array5 having two times the values in Array1
Array5 = 2 * Array1

# Find Covariance and Correlation
covariance = np.cov(Array1, Array4)[0][1]
correlation = np.corrcoef(Array1, Array5)[0][1]
```

```
print("Array4 (Array3 - Array2):", Array4)
print("Array5 (2 * Array1):", Array5)
print("Covariance between Array1 and Array4:", covariance)
print("Correlation between Array1 and Array5:", correlation)

# Task 1e: Create two random arrays and perform calculations
Array1 = np.random.rand(10)
Array2 = np.random.rand(10)

print("Array1:", Array1)
print("Array2:", Array2)

# Find the sum of the first half of both arrays
first_half_sum1 = np.sum(Array1[:5])
first_half_sum2 = np.sum(Array2[:5])

# Find the product of the second half of both arrays
second_half_product1 = np.prod(Array1[5:])
second_half_product2 = np.prod(Array2[5:])

print("Sum of first half of Array1:", first_half_sum1)
print("Sum of first half of Array2:", first_half_sum2)
print("Product of second half of Array1:", second_half_product1)
print("Product of second half of Array2:", second_half_product2)

# Task 1f: Create an array with random values and determine memory size
random_array = np.random.rand(10)
memory_size = random_array.nbytes
print("Random Array:", random_array)
print("Memory occupied by the array (in bytes):", memory_size)

# Task 1g: Create a 2D array with integer elements and perform operations
m = 5  # Example size
n = 5  # Example size
int_array = np.random.randint(10, 100, size=(m, n))
print("Original Integer Array:\n", int_array)
```

Output:-

```
dhirukumar@Dhirus-MacBook-Air davvvv % python -u "/Users/dhirukumar/davvvv/dav.py"
ARR1:
 [[0.35701005 0.22700135 0.36763376 0.73193597 0.79605832]
 [0.94362564 0.70244188 0.22604247 0.57194818 0.40726269]
 [0.23751679 0.44942973 0.33201587 0.46050832 0.50401811]
 [0.30557538 0.90338163 0.33453948 0.61006798 0.2438944 ]
 [0.37481906 0.71903369 0.47440574 0.49844623 0.5473963 ]]
Mean along second axis: [0.49592789 0.57026417 0.39669777 0.47949177 0.5228202 ]
Standard Deviation along second axis: [0.22532515 0.24573589 0.09786974 0.24637247 0.11308944]
Variance along second axis: [0.05077142 0.06038613 0.00957849 0.06069939 0.01278922]
Enter number of rows (m): 3
Enter number of columns (n): 4
Original Array:
 [[18 83 46 93]
 [45 31  7 65]
 [97 33 81 81]]
Shape: (3, 4)
Type: <class 'numpy.ndarray'>
Data Type: int64
Reshaped Array (n x m):
 [[18 83 46]
 [93 45 31]
 [ 7 65 97]
 [33 81 81]]
1D Array: [-0.45512062 -0.38997354  0.81498103 -0.2709727   1.02838241  0.2242197
 -1.61138487  1.30719701  1.27749846 -0.99498496]
Zero Indices: []
Non-Zero Indices: [0 1 2 3 4 5 6 7 8 9]
NaN Indices: []
Array1: [0.28585066 0.20874107 0.20204509 0.3422723  0.87247755 0.48806248
 0.1080699  0.99896356 0.85625744 0.82862757]
Array2: [0.95143321 0.29870801 0.23036696 0.59328085 0.53380484 0.6718651
 0.08002437 0.33847681 0.05962438 0.66539398]
Array3: [0.48651032 0.10804214 0.41801523 0.5685031  0.22240801 0.57715187
 0.07829538 0.37568619 0.63069232 0.33831812]
Array4 (Array3 - Array2): [-0.46492289 -0.19066587  0.18764828 -0.02477775 -0.31139683 -0.09471323
 -0.001729    0.03720938  0.57106794 -0.32707586]
```

Q2.
Do the following using PANDAS Series:

a. Create a series with 5 elements. Display the series sorted on index and also sorted on values seperately

b.Create a series with N elements with some duplicate values. Find the minimum and maximum ranksassigned to the values using ‘first’ and ‘max’ methods

c. Display the index value of the minimum and maximum element of a Series

Sol:-

```
import pandas as pd

# Task 2a: Create a series with 5 elements and display sorted series
series1 = pd.Series([5, 2, 3, 1, 4], index=['e', 'b', 'c', 'a', 'd'])
print("Original Series:\n", series1)
```

```
# Sorting by index
sorted_by_index = series1.sort_index()
print("\nSorted by index:\n", sorted_by_index)

# Sorting by values
sorted_by_values = series1.sort_values()
print("\nSorted by values:\n", sorted_by_values)

# Task 2b: Create a series with N elements with some duplicate values
series2 = pd.Series([1, 2, 2, 3, 4, 4, 5])
print("\nSeries with duplicates:\n", series2)

# Find minimum and maximum ranks using 'first' and 'max' methods
min_rank_first = series2.rank(method='first').min()
max_rank_first = series2.rank(method='first').max()
min_rank_max = series2.rank(method='max').min()
max_rank_max = series2.rank(method='max').max()

print("\nMinimum rank (first method):", min_rank_first)
print("Maximum rank (first method):", max_rank_first)
print("Minimum rank (max method):", min_rank_max)
print("Maximum rank (max method):", max_rank_max)

# Task 2c: Display the index value of the minimum and maximum element of a Series
min_index = series2.idxmin()
max_index = series2.idxmax()

print("\nIndex of minimum element:", min_index)
print("Index of maximum element:", max_index)
```

```
dhirukumar@Dhirus-MacBook-Air davvvv % python -u "/Users/dhirukumar/davvvv/dav.py"
Original Series:
 e    5
b    2
c    3
a    1
d    4
dtype: int64

Sorted by index:
 a    1
b    2
c    3
d    4
e    5
dtype: int64

Sorted by values:
 a    1
b    2
c    3
d    4
e    5
dtype: int64

Series with duplicates:
 0    1
1    2
2    2
3    3
4    4
5    4
6    5
dtype: int64

Minimum rank (first method): 1.0
Maximum rank (first method): 7.0
Minimum rank (max method): 1.0
Maximum rank (max method): 7.0

Index of minimum element: 0
Index of maximum element: 6
```

Q3.
Create a data frame having at least 3 columns and 50 rows to store numeric data generated using a randomfunction. Replace 10% of the values by null values whose index positions are generated using random function.Do the following:

a.Identify and count missing values in a data frame.

b.Drop the column having more than 5 null values.

c.Identify the row label having maximum of the sum of all values in a row and drop thatrow.

d.Sort the data frame on the basis of the first column.

e.Remove all duplicates from the first column.

f.Find the correlation between first and second column and covariance between second and third column.

g. Discretize the second column and create 5 bins.

Sol:-

```
import pandas as pd
import numpy as np

# Create a DataFrame with 3 columns and 50 rows of random numeric data
np.random.seed(0)  # For reproducibility
data = np.random.rand(50, 3) * 100  # Random values between 0 and 100
df = pd.DataFrame(data, columns=['Column1', 'Column2', 'Column3'])

# Replace 10% of the values with null values
num_nulls = int(0.1 * df.size)  # 10% of total values
null_indices = np.random.choice(df.size, num_nulls, replace=False)
df.values.ravel()[null_indices] = np.nan

print("DataFrame with NaN values:\n", df)

# Task a: Identify and count missing values in a DataFrame
missing_values_count = df.isnull().sum()
print("\nMissing values in each column:\n", missing_values_count)

# Task b: Drop the column having more than 5 null values
df_dropped_column = df.dropna(axis=1, thresh=len(df) - 5)
```

```
print("\nDataFrame after dropping columns with more than 5 null values:\n",
df_dropped_column)

# Task c: Identify the row label having maximum sum of all values in a row and drop that
row
row_sums = df_dropped_column.sum(axis=1)
max_row_index = row_sums.idxmax()
df_dropped_row = df_dropped_column.drop(index=max_row_index)
print("\nDataFrame after dropping the row with maximum sum:\n", df_dropped_row)

# Task d: Sort the DataFrame on the basis of the first column
df_sorted = df_dropped_row.sort_values(by='Column1')
print("\nSorted DataFrame based on Column1:\n", df_sorted)

# Task e: Remove all duplicates from the first column
df_no_duplicates = df_sorted.drop_duplicates(subset='Column1')
print("\nDataFrame after removing duplicates from Column1:\n", df_no_duplicates)

# Task f: Find the correlation between the first and second column
correlation = df_no_duplicates['Column1'].corr(df_no_duplicates['Column2'])
print("\nCorrelation between Column1 and Column2:", correlation)

# Find the covariance between the second and third column
covariance = df_no_duplicates['Column2'].cov(df_no_duplicates['Column3'])
print("Covariance between Column2 and Column3:", covariance)

# Task g: Discretize the second column and create 5 bins
df_no_duplicates['Column2_Binned'] = pd.cut(df_no_duplicates['Column2'], bins=5,
labels=False)
print("\nDataFrame with discretized Column2 into 5 bins:\n", df_no_duplicates)
```

Output:-

```
dhirukumar@Dhirus-MacBook-Air davvvv % python -u "/Users/dhirukumar/davvvv/dav.py"
DataFrame with NaN values:
      Column1    Column2    Column3
0   54.881350  71.518937  60.276338
1   54.488318  42.365480  64.589411
2         NaN  89.177300  96.366276
3   38.344152  79.172504  52.889492
4   56.804456  92.559664   7.103606
5    8.712930   2.021840  83.261985
6   77.815675  87.001215  97.861834
7   79.915856  46.147936  78.052918
8   11.827443  63.992102  14.335329
9   94.466892  52.184832  41.466194
10  26.455561  77.423369  45.615033
11  56.843395   1.878980  61.763550
12  61.209572  61.693400        NaN
13  68.182030  35.950790  43.703195
14  69.763120   6.022547  66.676672
15        NaN  21.038256  12.892630
16  31.542835  36.371077  57.019677
17  43.860151        NaN  10.204481
18  20.887676  16.130952  65.310833
19  25.329160  46.631077  24.442559
20  15.896958  11.037514  65.632959
21  13.818295  19.658236  36.872517
22  82.099323   9.710128        NaN
23   9.609841  97.645947  46.865120
24  97.676109  60.484552  73.926358
25   3.918779  28.280696  12.019656
26  29.614020        NaN  31.798318
27  41.426299   6.414750  69.247212
28  56.660145  26.538949  52.324805
29        NaN  57.594650  92.929620
30        NaN  66.741038        NaN
31  71.632720  28.940609  18.319136
32  58.651293   2.010755  82.894003
33   0.469548  67.781654  27.000797
34  73.519402  96.218855  24.875314
35        NaN  59.204193  57.225191
36  22.308163        NaN        NaN
37  84.640867  69.947928  29.743695
38
39  Missing values in each column:
40   Column1    7
41  Column2    3
42  Column3    5
43  dtype: int64
44
45
46  DataFrame after dropping columns with more than 5 null values:
47
48         Column2    Column3
    0   71.518937  60.276338
    1   42.365480  64.589411
    2   89.177300  96.366276
    3   79.172504  52.889492
    4   92.559664   7.103606
    5    2.021840  83.261985
    6   87.001215  97.861834
    7   46.147936  78.052918
    8   63.992102  14.335329
    9   52.184832  41.466194
    10  77.423369  45.615033
    11   1.878980  61.763550
    12  61.693400        NaN
    13  35.950790  43.703195
    14   6.022547  66.676672
    15  21.038256  12.892630
    16  36.371077  57.019677
    17        NaN  10.204481
    18  16.130952  65.310833
    19  46.631077  24.442559
    20  11.037514  65.632959
    21  19.658236  36.872517
    22   9.710128        NaN
    23  97.645947  46.865120
    24  60.484552  73.926358
    25  28.280696  12.019656
    26        NaN  31.798318
    27   6.414750  69.247212
    28  26.538949  52.324805
    29  57.594650  92.929620
    30  66.741038        NaN
    31  28.940609  18.319136
    32   2.010755  82.894003
    33  67.781654  27.000797
    34  96.218855  24.875314
```

Q4.
Consider two excel files having attendance of two workshops, each of duration 5 days. Each file has three
fields ‘Name’, ‘Date, duration (in minutes) where names may be repetitve within a file. Note that duration may
take one of three values (30, 40, 50) only. Import the data into two data frames and do the following:

a. Perform merging of the two data frames to find the names of students who had attended bothworkshops.

b. Find names of all students who have attended a single workshop only.

c.Merge two data frames row-wise and find the total number of records in the data frame.
d.Merge two data frames row-wise and use two columns viz. names and dates as multi-row indexes.Generate descriptive statistics for this hierarchical data frame

Sol:-

```
import pandas as pd

# Load the data from the two Excel files
# Assuming the files are named 'workshop1.xlsx' and 'workshop2.xlsx'
# and they are located in the same directory as this script.

# Replace 'workshop1.xlsx' and 'workshop2.xlsx' with your actual file paths
df1 = pd.read_excel('workshop1.xlsx')
df2 = pd.read_excel('workshop2.xlsx')

# Display the data frames
print("Workshop 1 DataFrame:\n", df1)
print("\nWorkshop 2 DataFrame:\n", df2)

# Task a: Merge the two data frames to find names of students who attended both workshops
merged_both = pd.merge(df1, df2, on='Name', how='inner')
print("\nStudents who attended both workshops:\n", merged_both[['Name']])

# Task b: Find names of all students who have attended a single workshop only
# Get unique names from both workshops
unique_names_df1 = set(df1['Name'])
unique_names_df2 = set(df2['Name'])
```

```
# Find names that are in one workshop but not the other
single_workshop_attendees =
unique_names_df1.symmetric_difference(unique_names_df2)
print("\nStudents who attended a single workshop only:\n", single_workshop_attendees)

# Task c: Merge the two data frames row-wise and find the total number of records
merged_row_wise = pd.concat([df1, df2], ignore_index=True)
total_records = merged_row_wise.shape[0]
print("\nTotal number of records after merging row-wise:", total_records)

# Task d: Merge two data frames row-wise and use 'Name' and 'Date' as multi-row indexes
multi_index_df = merged_row_wise.set_index(['Name', 'Date'])
print("\nDataFrame with multi-row indexes:\n", multi_index_df)

# Generate descriptive statistics for this hierarchical data frame
descriptive_stats = multi_index_df.describe()
print("\nDescriptive statistics for the hierarchical DataFrame:\n", descriptive_stats)
```

Output:-

```
dhirukumar@Dhirus-MacBook-Air davvvv % Workshop 1 DataFrame:
      Name       Date  Duration
0  Alice   2023-01-01        30
1  Bob     2023-01-01        40
2  Charlie 2023-01-02        50
3  Alice   2023-01-03        30

Workshop 2 DataFrame:
      Name       Date  Duration
0  Alice   2023-01-01        30
1  David   2023-01-01        40
2  Charlie 2023-01-02        50
3  Eve     2023-01-03        40

Students who attended both workshops:
      Name
0  Alice
1  Charlie

Students who attended a single workshop only:
 {'Eve', 'Bob', 'David'}

Total number of records after merging row-wise: 8

DataFrame with multi-row indexes:
                    Duration
Name    Date
Alice   2023-01-01      30
Alice   2023-01-01      30
Bob     2023-01-01      40
Charlie 2023-01-02      50
David   2023-01-01      40
Eve     2023-01-03      40

Descriptive statistics for the hierarchical DataFrame:
          Duration
count      6.000000
```

Q5.
Using Iris data, plot the following with proper legend and axis labels: (Download IRIS data from: https://archive.ics.uci.edu/ml/datasets/iris or import it from sklearn datasets)

a. Load data into pandas' data frame. Use pandas.info () method to look at the info on datatypes in the dataset.

b. Find the number of missing values in each column (Check number of null values in a column using df.isnull().sum())

c.Plot bar chart to show the frequency of each class label in the data.

d.Draw a scatter plot for Petal Length vs Sepal Length and fit a regression line

e.Plot density distribution for feature Petal width.

f.Use a pair plot to show pairwise bivariate distribution in the Iris Dataset.

g.Draw heatmap for any two numeric attributes

h.Compute mean, mode, median, standard deviation, confidence interval and standard error for each numeric feature

i. Compute correlation coefficients between each pair of features and plot heatmap

Sol:-

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
import numpy as np
import scipy.stats as stats

# Load the Iris dataset
iris_data = load_iris()
df = pd.DataFrame(data=iris_data.data, columns=iris_data.feature_names)
df['species'] = iris_data.target_names[iris_data.target]

# a. Display info on datatypes in the dataset
print("Iris Dataset Info:")
print(df.info())

# b. Find the number of missing values in each column
missing_values = df.isnull().sum()
print("\nMissing values in each column:")
print(missing_values)
```

```
# c. Plot bar chart to show the frequency of each class label in the data
plt.figure(figsize=(8, 5))
sns.countplot(data=df, x='species', palette='viridis')
plt.title('Frequency of Each Class Label in the Iris Dataset')
plt.xlabel('Species')
plt.ylabel('Frequency')
plt.legend(title='Species')
plt.show()

# d. Draw a scatter plot for Petal Length vs Sepal Length and fit a regression line
plt.figure(figsize=(8, 5))
sns.regplot(data=df, x='sepal length (cm)', y='petal length (cm)', marker='o', color='blue')
plt.title('Petal Length vs Sepal Length')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Petal Length (cm)')
plt.legend(title='Species')
plt.show()

# e. Plot density distribution for feature Petal width
plt.figure(figsize=(8, 5))
sns.kdeplot(data=df, x='petal width (cm)', hue='species', fill=True, common_norm=False,
palette='crest')
plt.title('Density Distribution of Petal Width')
plt.xlabel('Petal Width (cm)')
plt.ylabel('Density')
plt.legend(title='Species')
plt.show()

# f. Use a pair plot to show pairwise bivariate distribution in the Iris Dataset
sns.pairplot(df, hue='species', palette='bright')
plt.suptitle('Pairwise Bivariate Distribution in the Iris Dataset', y=1.02)
plt.show()

# g. Draw heatmap for any two numeric attributes
plt.figure(figsize=(8, 5))
sns.heatmap(df[['sepal length (cm)', 'sepal width (cm)']].corr(), annot=True, cmap='coolwarm',
fmt='.2f')
plt.title('Heatmap of Sepal Length and Sepal Width Correlation')
plt.show()

# h. Compute mean, mode, median, standard deviation, confidence interval and standard error for
each numeric feature
statistics = {}
for column in df.columns[:-1]:  # Exclude species column
    statistics[column] = {
```

```
        'Mean': df[column].mean(),
        'Median': df[column].median(),
        'Mode': df[column].mode()[0],
        'Standard Deviation': df[column].std(),
        'Standard Error': stats.sem(df[column]),
        'Confidence Interval (95%)': stats.t.interval(0.95, len(df[column])-1, loc=df[column].mean(),
scale=stats.sem(df[column]))
    }

statistics_df = pd.DataFrame(statistics).T
print("\nStatistics for Numeric Features:")
print(statistics_df)

# i. Compute correlation coefficients between each pair of features and plot heatmap
correlation_matrix = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Coefficients Heatmap')
plt.show()
```

Output:-

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   sepal length (cm)     150 non-null    float64
 1   sepal width (cm)      150 non-null    float64
 2   petal length (cm)     150 non-null    float64
 3   petal width (cm)      150 non-null    float64
 4   species               150 non-null    object
dtypes: float64(4), object(1)
memory usage: 4.0+ KB Missing values in each column:
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
species              0
dtype: int64
zsh: parse error near `\n'
```

```
dhirukumar@Dhirus-MacBook-Air davvvv % Statistics for Numeric Features:
                  Mean  Median  Mode  Standard Deviation  Standard Error  Confidence Interval (95%)
sepal length (cm)  5.843333     5.8   5.0            0.828066        0.067  (5.757, 5.930)
sepal width (cm)   3.057333     3.0   3.0            0.435866        0.035  (2.988, 3.127)
petal length (cm)  3.758000     4.0   1.5            1.765298        0.144  (3.475, 4.041)
petal width (cm)   1.199333     1.2   0.2            0.762238        0.062  (1.078, 1.320)
```

Q6.
Using Titanic dataset, to do the following:

a.Clean the data by dropping the column which has the largest number of missing values.

b.Find total number of passengers with age more than 30

c.Find total fare paid by passengers of second class

d.Compare number of survivors of each passenger class

e.Compute descriptive statistics for age attribute gender wise

f.Draw a scatter plot for passenger fare paid by Female and Male passengers separately

g.Compare density distribution for features age and passenger fare

h.Draw the pie chart for three groups labelled as class 1, class 2, class 3 respectively displayed in different colours. The occurrence of each group converted into percentage should be displayed in the pie chart. Appropriately Label the chart.

i.Find % of survived passengers for each class and answer the question “Did class play a role in survival?”

Sol:

```
-import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the Titanic dataset
titanic_data = pd.read_csv('https://raw.githubusercontent.com/datasciencedojo/datasets/master/
titanic.csv')

# a. Clean the data by dropping the column which has the largest number of missing values
column_with_most_nan = titanic_data.isnull().sum().idxmax()
titanic_data.drop(columns=[column_with_most_nan], inplace=True)
print(f"Dropped column: {column_with_most_nan}")

# b. Find total number of passengers with age more than 30
passengers_over_30 = titanic_data[titanic_data['Age'] > 30].shape[0]
print(f"Total number of passengers with age more than 30: {passengers_over_30}")

# c. Find total fare paid by passengers of second class
total_fare_second_class = titanic_data[titanic_data['Pclass'] == 2]['Fare'].sum()
```

```
print(f"Total fare paid by passengers of second class: {total_fare_second_class}")

# d. Compare number of survivors of each passenger class
survivors_per_class = titanic_data.groupby('Pclass')['Survived'].sum()
print("\nNumber of survivors of each passenger class:")
print(survivors_per_class)

# e. Compute descriptive statistics for age attribute gender-wise
age_statistics_gender = titanic_data.groupby('Sex')['Age'].describe()
print("\nDescriptive statistics for age attribute gender-wise:")
print(age_statistics_gender)

# f. Draw a scatter plot for passenger fare paid by Female and Male passengers separately
plt.figure(figsize=(10, 6))
sns.scatterplot(data=titanic_data, x='Fare', y='Age', hue='Sex', alpha=0.6)
plt.title('Passenger Fare vs Age by Gender')
plt.xlabel('Fare')
plt.ylabel('Age')
plt.legend(title='Gender')
plt.show()

# g. Compare density distribution for features age and passenger fare
plt.figure(figsize=(10, 6))
sns.kdeplot(data=titanic_data, x='Age', fill=True, label='Age', color='blue', alpha=0.5)
sns.kdeplot(data=titanic_data, x='Fare', fill=True, label='Fare', color='orange', alpha=0.5)
plt.title('Density Distribution of Age and Fare')
plt.xlabel('Value')
plt.ylabel('Density')
plt.legend()
plt.show()

# h. Draw the pie chart for three groups labelled as class 1, class 2, class 3 respectively
class_counts = titanic_data['Pclass'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(class_counts, labels=['Class 1', 'Class 2', 'Class 3'], autopct='%1.1f%%', colors=['gold',
'lightcoral', 'lightskyblue'])
plt.title('Passenger Distribution by Class')
plt.show()

# Find % of survived passengers for each class
survival_rate_per_class = titanic_data.groupby('Pclass')['Survived'].mean() * 100
print("\n% of survived passengers for each class:")
print(survival_rate_per_class)

# Answer the question: Did class play a role in survival?
print("\nDid class play a role in survival?")
```

```
if survival_rate_per_class.max() > survival_rate_per_class.min():
    print("Yes, class played a role in survival.")
else:
    print("No, class did not play a role in survival.”)
```

Output:-

```
dhirukumar@Dhirus-MacBook-Air davvvv % Dropped column: Cabin
Total number of passengers with age more than 30: [number]
 Total fare paid by passengers of second class: [amount]
Number of survivors of each passenger class:
Pclass
1    [number]
2    [number]
3    [number]
Name: Survived, dtype: int64
Descriptive statistics for age attribute gender-wise:
        count      mean       std  min   25%   50%   75%   max
Sex
female   [number]  [mean]  [std]  [min]  [25%]  [50%]  [75%]  [max]
male     [number]  [mean]  [std]  [min]  [25%]  [50%]  [75%]  [max]
% of survived passengers for each class:
Pclass
1    [percentage]
2
```

Q7.
Consider the following data frame containing a family name, gender of the family member and her/ his monthly income in each record.

FamilyName Gender MonthlyIncome (Rs.)
Shah Male 44000.00
Vats Male 65000.00
Vats Female 43150.00
Kumar Female 66500.00
Vats Female 255000.00
Kumar Male 103000.00
Shah Male 55000.00
Shah Female 112400.00
Kumar Female 81030.00
Vats Male 71900.00

Write a program in Python using Pandas to perform the following:

a.Calculate and display familywise gross monthly income.

b.Display the highest and lowest monthly income for each family name

c.Calculate and display monthly income of all members earning income less than Rs. 80000.00.

d.Display total number of females along with their average monthly income

e.Delete rows with Monthly income less than the average income of all members

Sol:-

```
import pandas as pd

# Create the DataFrame
data = {
   'FamilyName': ['Shah', 'Vats', 'Vats', 'Kumar', 'Vats', 'Kumar', 'Shah', 'Shah', 'Kumar', 'Vats'],
   'Gender': ['Male', 'Male', 'Female', 'Female', 'Female', 'Male', 'Male', 'Female', 'Female', 'Male'],
   'MonthlyIncome': [44000.00, 65000.00, 43150.00, 66500.00, 255000.00, 103000.00, 55000.00,
112400.00, 81030.00, 71900.00]
}

df = pd.DataFrame(data)

# a. Calculate and display familywise gross monthly income
familywise_income = df.groupby('FamilyName')['MonthlyIncome'].sum()
print("Familywise Gross Monthly Income:")
print(familywise_income)
```

```
# b. Display the highest and lowest monthly income for each family name
highest_lowest_income = df.groupby('FamilyName')['MonthlyIncome'].agg(['max', 'min'])
print("\nHighest and Lowest Monthly Income for Each Family Name:")
print(highest_lowest_income)

# c. Calculate and display monthly income of all members earning income less than Rs. 80000.00
income_below_80000 = df[df['MonthlyIncome'] < 80000]
print("\nMonthly Income of Members Earning Less Than Rs. 80000.00:")
print(income_below_80000)

# d. Display total number of females along with their average monthly income
female_count = df[df['Gender'] == 'Female'].shape[0]
average_female_income = df[df['Gender'] == 'Female']['MonthlyIncome'].mean()
print(f"\nTotal Number of Females: {female_count}, Average Monthly Income:
{average_female_income:.2f}")

# e. Delete rows with Monthly income less than the average income of all members
average_income = df['MonthlyIncome'].mean()
df_filtered = df[df['MonthlyIncome'] >= average_income]

print("\nDataFrame after deleting rows with Monthly Income less than the average income:")
print(df_filtered)
```

Output:-

```
dhirukumar@Dhirus-MacBook-Air davvvv % Familywise Gross Monthly Income:
FamilyName
Kumar     257530.0
Shah      210650.0
Vats      343650.0
Name: MonthlyIncome, dtype: float64

Highest and Lowest Monthly Income for Each Family Name:
            max       min
FamilyName
Kumar  103000.0   66500.0
Shah   112400.0   44000.0
Vats   255000.0   43150.0

Monthly Income of Members Earning Less Than Rs. 80000.00:
  FamilyName Gender  MonthlyIncome
0       Shah   Male        44000.0
1       Vats   Male        65000.0
2       Vats  Female        43150.0
3      Kumar  Female        66500.0
4       Vats  Female       255000.0
6       Shah   Male        55000.0
8      Kumar  Female        81030.0
9       Vats   Male        71900.0

Total Number of Females: 4, Average Monthly Income: 74045.00

DataFrame after deleting rows with Monthly Income less than the average income:
  FamilyName Gender  MonthlyIncome
4       Vats  Female       255000.0
6       Shah   Male        55000.0
7       Shah   Male       112400.0
8      Kumar  Female        81030.0
9       Vats   Male        71900.0
```