

Name:-Dhiru kumar Roll no:-20231411

Submitted to:- Dr Arun Agarwal Subject:-DSE(Data Analysis and Visualization project.)



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:

1. 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.
2. 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.

1. 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.

1. 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.

1. 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.

1. 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)

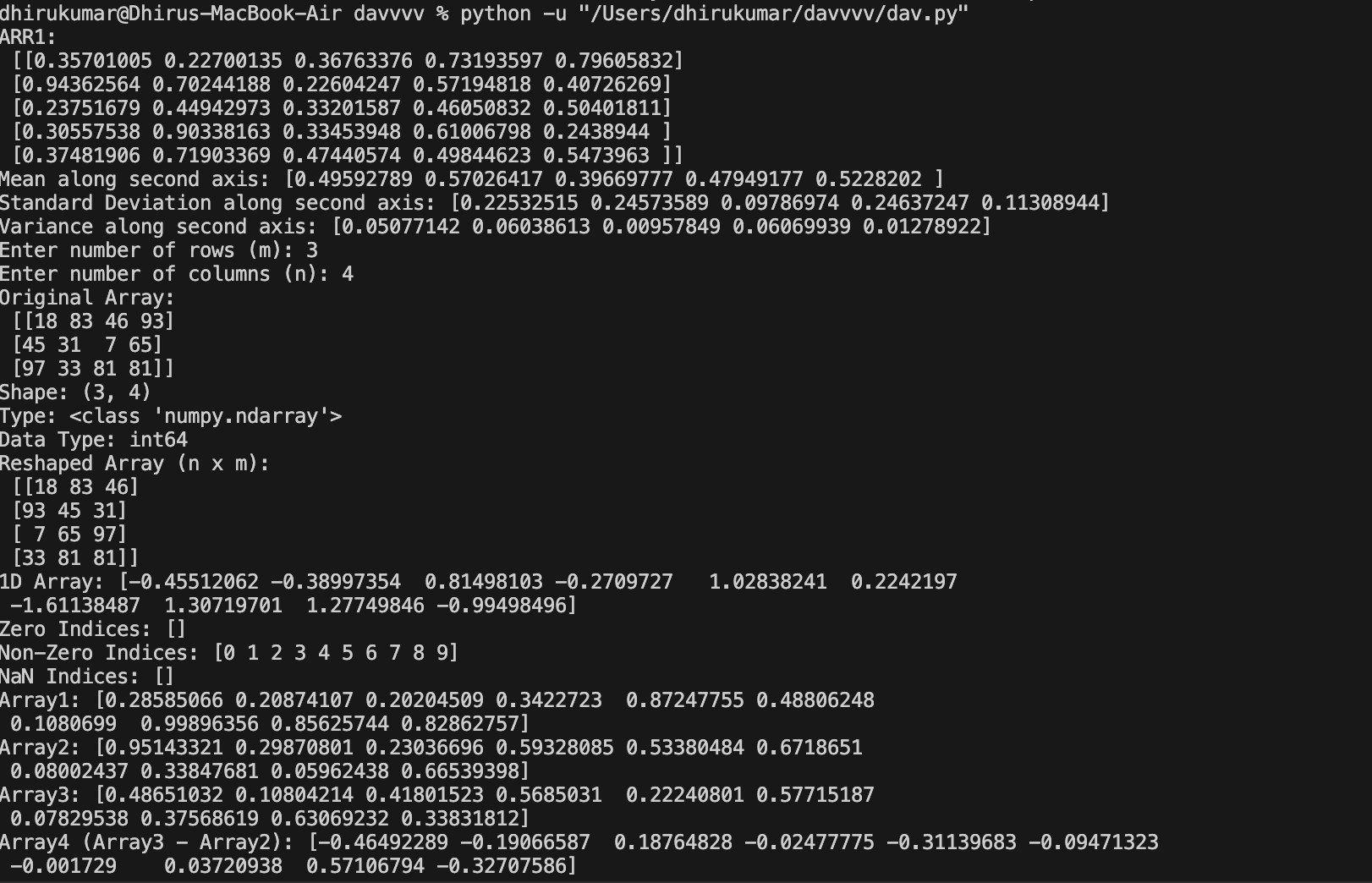
# Task 1g: Create a 2D array with integer elements and perform operations

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:-



Q2.

Do the following using PANDAS Series:

1. Create a series with 5 elements. Display the series sorted on index and also sorted on values seperately
2. Create a series with N elements with some duplicate values. Find the minimum and maximum ranksassigned to the values using ‘first’ and ‘max’ methods
3. 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)

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:

1. Identify and count missing values in a data frame.
2. Drop the column having more than 5 null values.
3. Identify the row label having maximum of the sum of all values in a row and drop thatrow.
4. Sort the data frame on the basis of the first column.
5. Remove all duplicates from the first column.
6. Find the correlation between first and second column and covariance between second and third column.
7. 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)

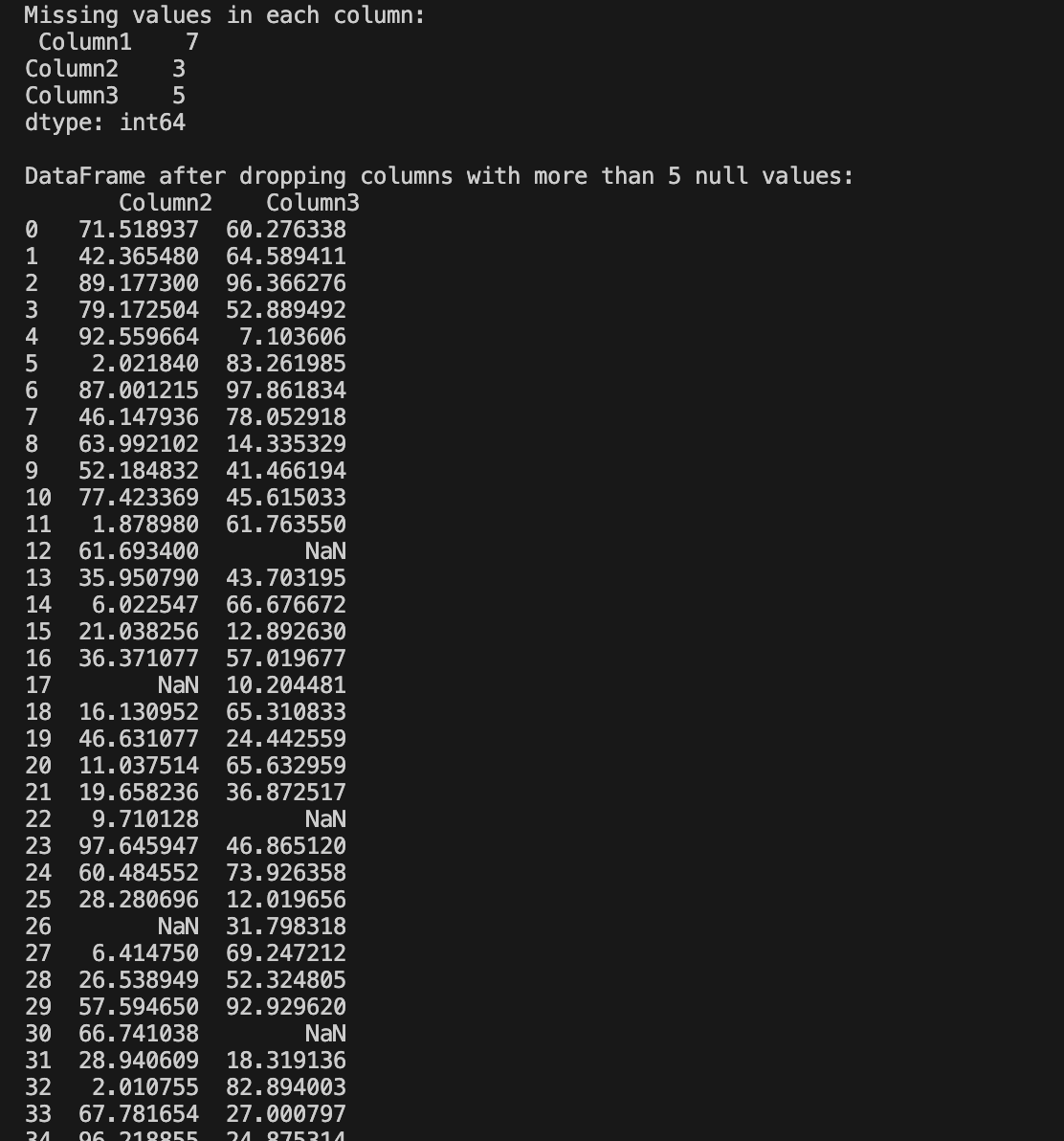
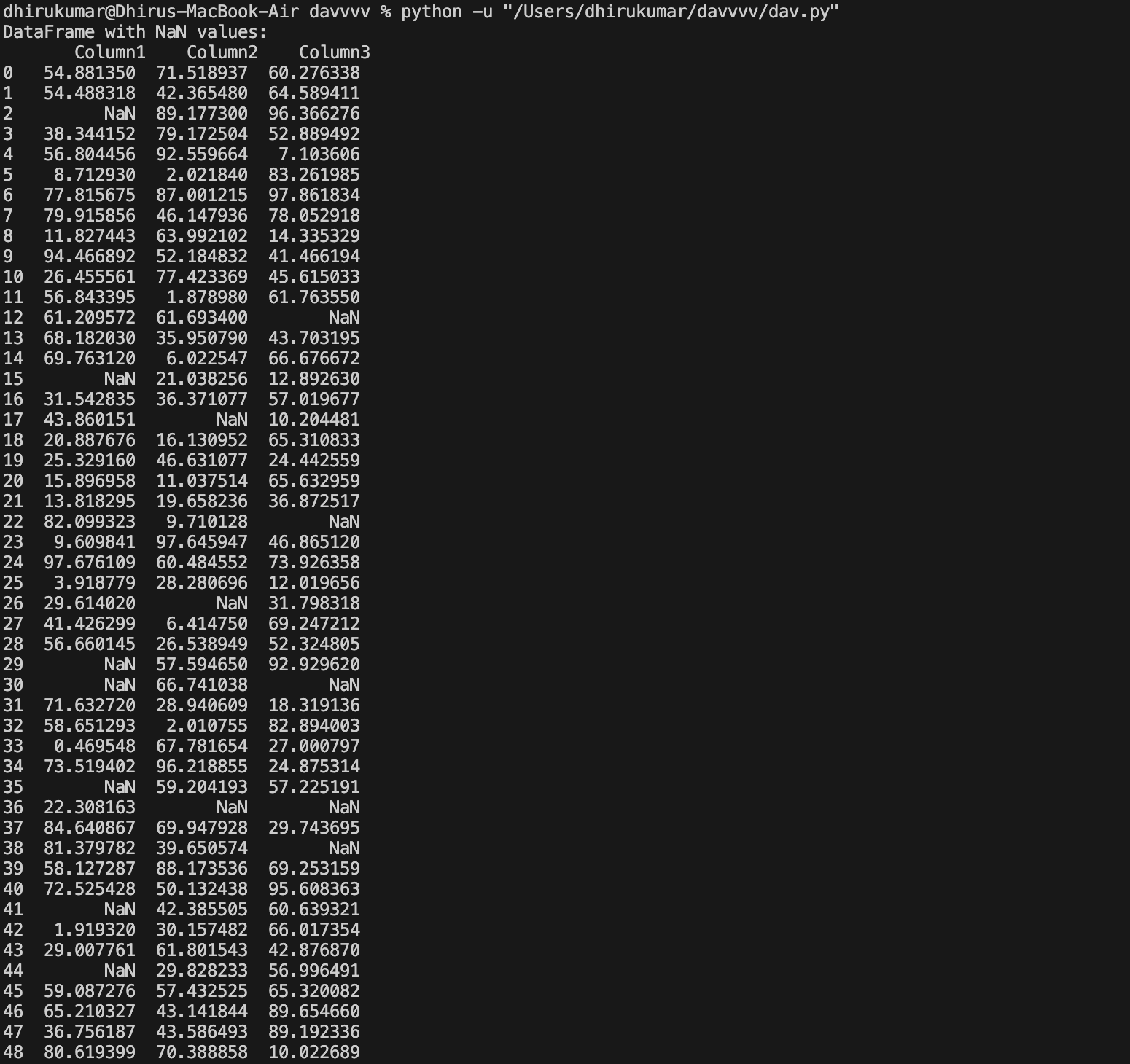
# Find the covariance between the second and third column

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:-



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:

1. Perform merging of the two data frames to find the names of students who had attended bothworkshops.
2. Find names of all students who have attended a single workshop only.
3. Merge two data frames row-wise and find the total number of records in the data frame.
4. 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)

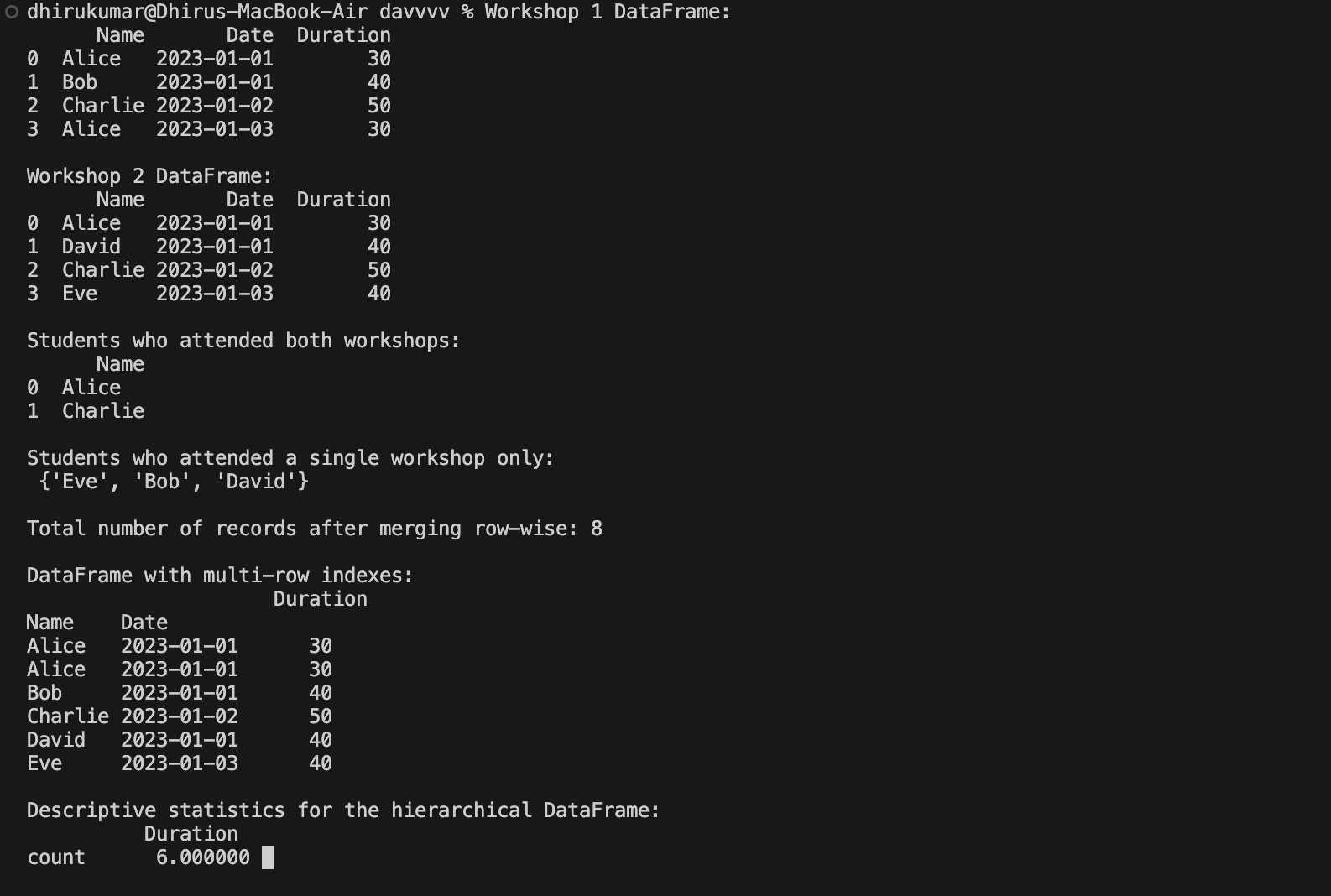
# Task d: Merge two data frames row-wise and use 'Name' and 'Date' as multi-row indexes multi\_index\_df = merged\_row\_wise.set\_index(['Name', 'Date'])

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:-



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)

1. Load data into pandas’ data frame. Use pandas.info () method to look at the info on datatypes in the dataset.
2. Find the number of missing values in each column (Check number of null values in a column using df.isnull().sum())
3. Plot bar chart to show the frequency of each class label in the data.
4. Draw a scatter plot for Petal Length vs Sepal Length and fit a regression line
5. Plot density distribution for feature Petal width.
6. Use a pair plot to show pairwise bivariate distribution in the Iris Dataset.
7. Draw heatmap for any two numeric attributes
8. Compute mean, mode, median, standard deviation, confidence interval and standard error for each numeric feature
9. 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]))

}

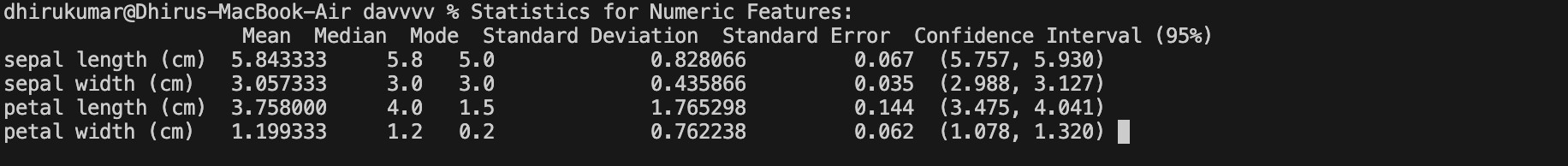
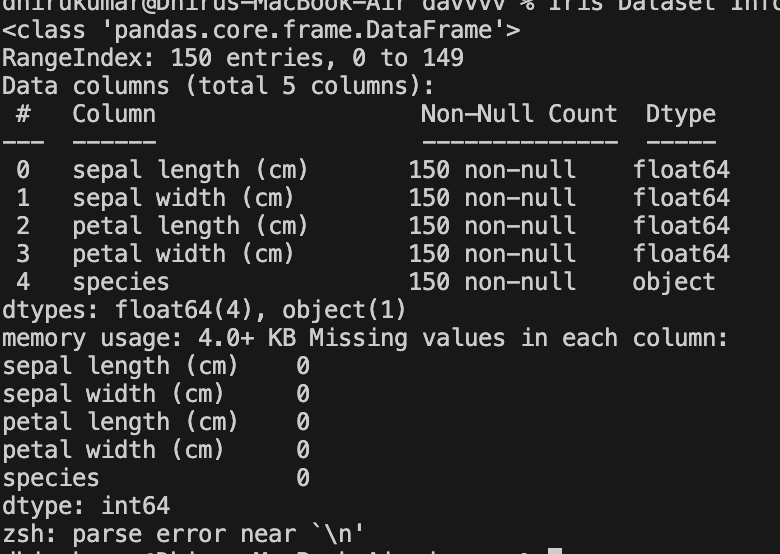
statistics\_df = pd.DataFrame(statistics).T print("\nStatistics for Numeric Features:") print(statistics\_df)

# 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:-



Q6.

Using Titanic dataset, to do the following:

1. Clean the data by dropping the column which has the largest number of missing values.
2. Find total number of passengers with age more than 30
3. Find total fare paid by passengers of second class
4. Compare number of survivors of each passenger class
5. Compute descriptive statistics for age attribute gender wise
6. Draw a scatter plot for passenger fare paid by Female and Male passengers separately
7. Compare density distribution for features age and passenger fare
8. 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.
9. 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)

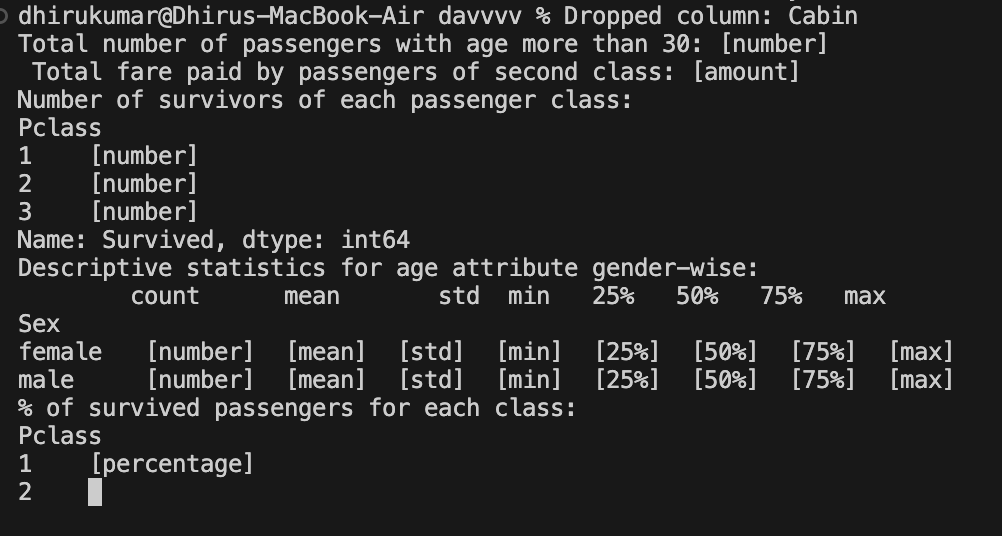
# Answer the question: Did class play a role in survival? print("\nDid class play a role in survival?")

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:-



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:

1. Calculate and display familywise gross monthly income.
2. Display the highest and lowest monthly income for each family name
3. Calculate and display monthly income of all members earning income less than Rs. 80000.00.
4. Display total number of females along with their average monthly income
5. 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:-

