

RAMANUJAN COLLEGE

UNIVERSITY OF DELHI

SOUTH CAMPUS IN KALKAJI 110019

PRACTICAL FILE

Data Analysis and Visualization using Python

Submitted By: Yuvraj Singh

Examination Roll Number: 23020570056

Class Roll Number: 20231471

Submitted to: Dr. Arun Agarwal

Index

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO. | Practical | Page No. | Remarks |
| 1 | Write programs in Python using NumPy library. | 4 |  |
| 2 | Do the following using PANDAS Series: | 12 |  |
| 3 | Create a data frame having at least 3 columns and 50 rows to store numeric data generated using a random function. Replace 10% of the values by null values whose index positions are generated using random function. | 15 |  |
| 4 | Using Iris data, plot the following with proper legend and axis labels | 18 |  |
| 5 | Using Titanic dataset, to Clean the data by dropping the column which has the largest number of missing values. | 26 |  |
| 6 | Consider the following data frame containing a family name, gender of the family member and her/his monthly income in each record. Write a program in Python using Pandas. | 29 |  |

# ***Acknowledgment***

**I would like to express my sincere gratitude to my instructor, Dr. Arun Agarwal, for providing me with the opportunity to work on this series of practical. Their invaluable guidance and support have been instrumental in understanding the core concepts of computer graphics.**

**I would also like to thank Ramanujan College, University of Delhi, for offering such a rich curriculum that fosters technical learning and hands-on experience. The practical applications of various algorithms and transformation techniques have deepened my knowledge in the field.**

**Furthermore, I would like to thank my peers for their encouragement and collaboration during the course of this project. Their inputs have enhanced my learning experience and contributed to a stimulating environment for problem-solving.**

1. 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.**

**Code:**

import numpy as np

ARR1 = np.random.rand(5, 4)

mean\_along\_axis1 = np.mean(ARR1, axis=1)

std\_dev\_along\_axis1 = np.std(ARR1, axis=1)

variance\_along\_axis1 = np.var(ARR1, axis=1)

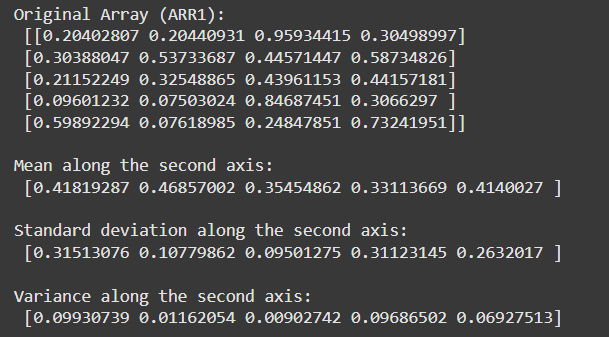
print("Original Array (ARR1):\n", ARR1)

print("\nMean along the second axis:\n", mean\_along\_axis1)

print("\nStandard deviation along the second axis:\n", std\_dev\_along\_axis1)

print("\nVariance along the second axis:\n", variance\_along\_axis1)

**Output:**

****

**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.**

**Code:**

import numpy as np

m = int(input("Enter the number of rows (m): "))

n = int(input("Enter the number of columns (n): "))

arr = np.random.randint(0, 100, size=(m, n))

print("Original Array:\n", arr)

print("\nShape:", arr.shape)

print("Type:", type(arr))

print("Data Type:", arr.dtype)

reshaped\_arr = arr.reshape(n, m)

print("\nReshaped Array:\n", reshaped\_arr)

ARR1 = np.random.rand(5, 4)

mean\_along\_axis1 = np.mean(ARR1, axis=1)

std\_dev\_along\_axis1 = np.std(ARR1, axis=1)

variance\_along\_axis1 = np.var(ARR1, axis=1)

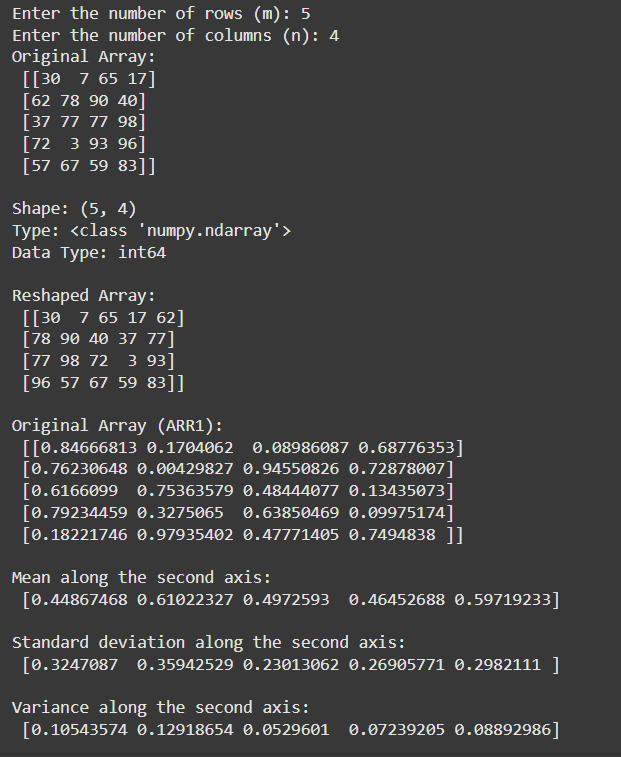
print("\nOriginal Array (ARR1):\n", ARR1)

print("\nMean along the second axis:\n", mean\_along\_axis1)

print("\nStandard deviation along the second axis:\n", std\_dev\_along\_axis1)

print("\nVariance along the second axis:\n", variance\_along\_axis1)

**Output:**

****

**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.**

**Code:**

import numpy as np

arr = np.array([0, 1, 2, 0, np.nan, 4, 0, np.nan, 9])

zero\_indices = []

non\_zero\_indices = []

nan\_indices = []

for i in range(len(arr)):

    if arr[i] == 0:

        zero\_indices.append(i)

    elif arr[i] != 0 and not np.isnan(arr[i]):

        non\_zero\_indices.append(i)

    elif np.isnan(arr[i]):

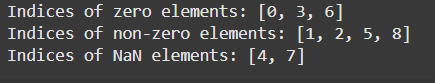
        nan\_indices.append(i)

print("Indices of zero elements:", zero\_indices)

print("Indices of non-zero elements:", non\_zero\_indices)

print("Indices of NaN elements:", nan\_indices)

**Output:**



**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.**

**Code:**

import numpy as np

Array1 = np.random.rand(5, 4)

Array2 = np.random.rand(5, 4)

Array3 = np.random.rand(5, 4)

Array4 = Array3 - Array2

Array5 = 2 \* Array1

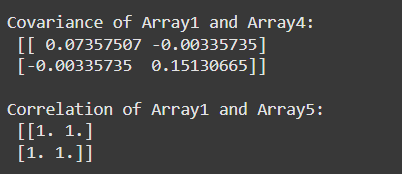
covariance\_1\_4 = np.cov(Array1.flatten(), Array4.flatten())

print("\nCovariance of Array1 and Array4:\n", covariance\_1\_4)

correlation\_1\_5 = np.corrcoef(Array1.flatten(), Array5.flatten())

print("\nCorrelation of Array1 and Array5:\n", correlation\_1\_5)

**Output:**

****

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.

**Code:**

import numpy as np

Array1 = np.random.rand(10)

Array2 = np.random.rand(10)

sum\_first\_half = np.sum(Array1[:5]) + np.sum(Array2[:5])

product\_second\_half = np.prod(Array1[5:]) \* np.prod(Array2[5:])

print("Sum of the first half of both arrays:", sum\_first\_half)

print("Product of the second half of both arrays:", product\_second\_half)

**Output:**



f. Create an array with random values. Determine the size of the memory occupied by the array.

**Code:**

import numpy as np

my\_array = np.random.rand(5, 4)

size\_in\_bytes = my\_array.nbytes

print("Size of the array in bytes:", size\_in\_bytes)

**Output:**



g. 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

**Code:**

import numpy as np

m = int(input("Enter the number of rows (m): "))

n = int(input("Enter the number of columns (n): "))

arr = np.random.randint(10, 100, size=(m, n))

print("Original Array:\n", arr)

row1 = int(input("Enter the first row index to swap: "))

row2 = int(input("Enter the second row index to swap: "))

if 0 <= row1 < m and 0 <= row2 < m:

    new\_arr = arr.copy()

    new\_arr[[row1, row2]] = new\_arr[[row2, row1]]

    print("\nArray after swapping rows", row1, "and", row2, ":\n", new\_arr)

else:

    print("\nInvalid row indices.")

col\_index = int(input("Enter the column index to reverse: "))

if 0 <= col\_index < n:

    new\_arr = new\_arr.copy()

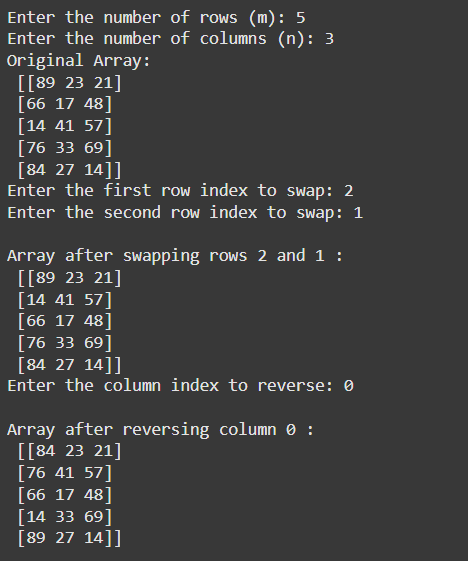
    new\_arr[:, col\_index] = new\_arr[::-1, col\_index]

    print("\nArray after reversing column", col\_index, ":\n", new\_arr)

else:

    print("\nInvalid column index.")

**Output:**



2. Do the following using PANDAS Series:

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

**Code:**

import pandas as pd

data = {'a': 10, 'b': 20, 'c': 30, 'd': 40, 'e': 50}

series = pd.Series(data)

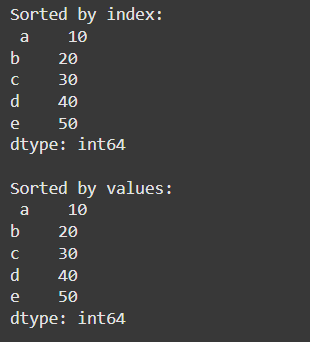
sorted\_by\_index = series.sort\_index()

print("Sorted by index:\n", sorted\_by\_index)

sorted\_by\_values = series.sort\_values()

print("\nSorted by values:\n", sorted\_by\_values)

Output:



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

**Code:**

import pandas as pd

import numpy as np

data = {'a': 10, 'b': 20, 'c': 10, 'd': 40, 'e': 20}

series = pd.Series(data)

ranks\_first = series.rank(method='first')

print("Ranks using 'first' method:\n", ranks\_first)

min\_rank\_first = ranks\_first.min()

max\_rank\_first = ranks\_first.max()

print(f"\nMinimum rank (first): {min\_rank\_first}")

print(f"Maximum rank (first): {max\_rank\_first}")

ranks\_max = series.rank(method='max')

print("\nRanks using 'max' method:\n", ranks\_max)

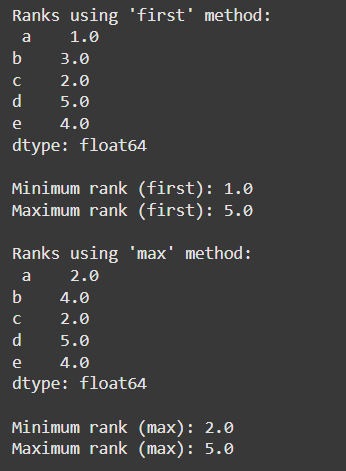
min\_rank\_max = ranks\_max.min()

max\_rank\_max = ranks\_max.max()

print(f"\nMinimum rank (max): {min\_rank\_max}")

print(f"Maximum rank (max): {max\_rank\_max}")

**Output:**



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

**Code:**

min\_index = series.idxmin()

max\_index = series.idxmax()

print(f"\nIndex of minimum element: {min\_index}")

print(f"Index of maximum element: {max\_index}")

**Output:**

****

**3. Create a data frame having at least 3 columns and 50 rows to store numeric data generated using a random function. 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 that row.

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.

**Code:**

import pandas as pd

import numpy as np

data = {'col1': np.random.rand(50),

        'col2': np.random.rand(50),

        'col3': np.random.rand(50)}

df = pd.DataFrame(data)

num\_nulls = int(0.1 \* df.size)

null\_indices = np.random.choice(df.size, num\_nulls, replace=False)

df.values.ravel()[null\_indices] = np.nan

# a. Identify and count missing values

missing\_values = df.isnull().sum()

print("Missing values:\n", missing\_values)

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

df = df.dropna(axis=1, thresh=df.shape[0]-5)

# c. Identify the row with the maximum sum and drop that row

row\_sums = df.sum(axis=1)

max\_sum\_row\_index = row\_sums.idxmax()

df = df.drop(index=max\_sum\_row\_index)

# d. Sort the DataFrame based on the first column

df = df.sort\_values(by='col1')

# e. Remove all duplicates from the first column

df = df.drop\_duplicates(subset='col1')

# f. Find the correlation and covariance

correlation = df['col1'].corr(df['col2'])

covariance = df['col2'].cov(df['col3'])

print("\nCorrelation between col1 and col2:", correlation)

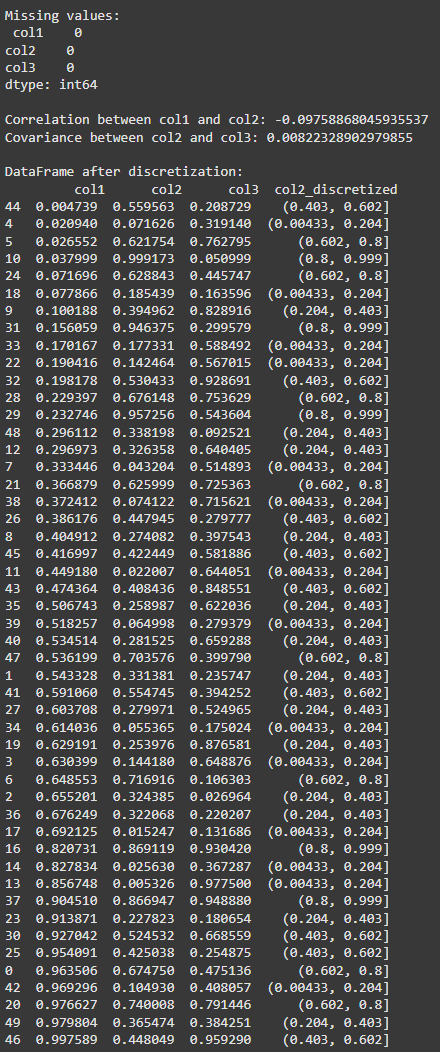
print("Covariance between col2 and col3:", covariance)

# g. Discretize the second column into 5 bins

df['col2\_discretized'] = pd.cut(df['col2'], bins=5)

print("\nDataFrame after discretization:\n", df)

**Output:**



**4. 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**

**Code:**

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np

from scipy import stats

from sklearn.datasets import load\_iris

# a. Load data into pandas DataFrame

iris = load\_iris()

df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

df['target'] = iris.target

df['species'] = df['target'].apply(lambda x: iris.target\_names[x])

print(df.info())

# b. Find the number of missing values in each column

print(df.isnull().sum())

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

plt.figure(figsize=(10, 6))

sns.countplot(data=df, x='species')

plt.title('Frequency of Each Class Label in the Iris Dataset')

plt.xlabel('Species')

plt.ylabel('Frequency')

plt.show()

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

plt.figure(figsize=(10, 6))

sns.regplot(x='sepal length (cm)', y='petal length (cm)', data=df)

plt.title('Scatter Plot with Regression Line: Petal Length vs Sepal Length')

plt.xlabel('Sepal Length (cm)')

plt.ylabel('Petal Length (cm)')

plt.show()

# e. Plot density distribution for feature Petal Width

plt.figure(figsize=(10, 6))

sns.kdeplot(data=df, x='petal width (cm)', fill=True)

plt.title('Density Distribution for Petal Width')

plt.xlabel('Petal Width (cm)')

plt.ylabel('Density')

plt.show()

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

sns.pairplot(df, hue='species')

plt.show()

# g. Draw heatmap for any two numeric attributes

plt.figure(figsize=(10, 6))

sns.heatmap(df[['sepal length (cm)', 'petal length (cm)']].corr(), annot=True, cmap='coolwarm')

plt.title('Heatmap for Sepal Length and Petal Length')

plt.show()

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

numeric\_cols = df.select\_dtypes(include=[np.number])

descriptive\_stats = numeric\_cols.describe()

mode = numeric\_cols.mode().iloc[0]

std\_error = numeric\_cols.sem()

print("Descriptive Statistics:")

print(descriptive\_stats)

print("\nMode:")

print(mode)

print("\nStandard Error:")

print(std\_error)

# Confidence Interval (95%) for each numeric feature

conf\_intervals = {}

for column in numeric\_cols.columns:

    mean = numeric\_cols[column].mean()

    std\_err = stats.sem(numeric\_cols[column])

    conf\_interval = stats.t.interval(0.95, len(numeric\_cols[column])-1, loc=mean, scale=std\_err)

    conf\_intervals[column] = conf\_interval

print("\nConfidence Intervals (95%):")

print(conf\_intervals)

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

plt.figure(figsize=(10, 6))

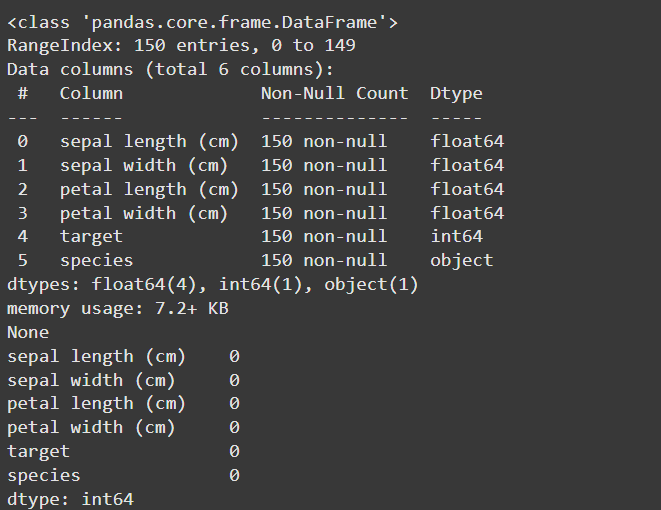
correlation\_matrix = numeric\_cols.corr()

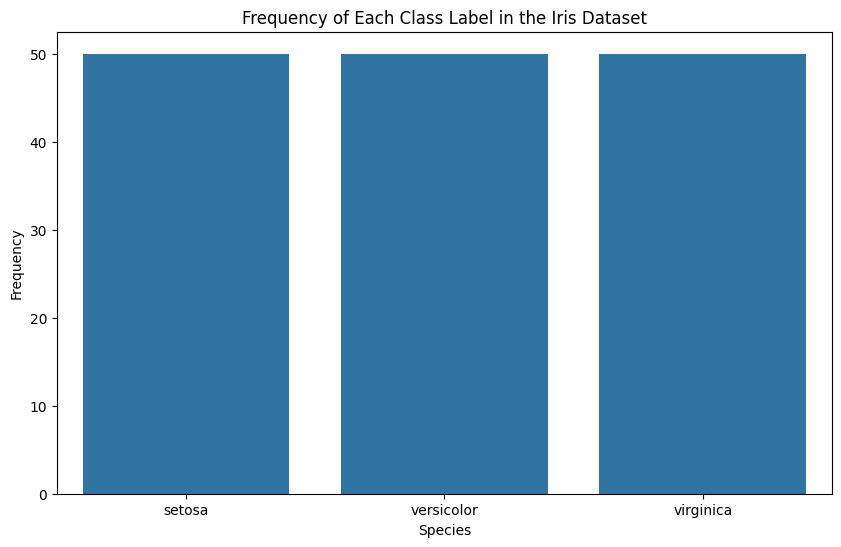
sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm')

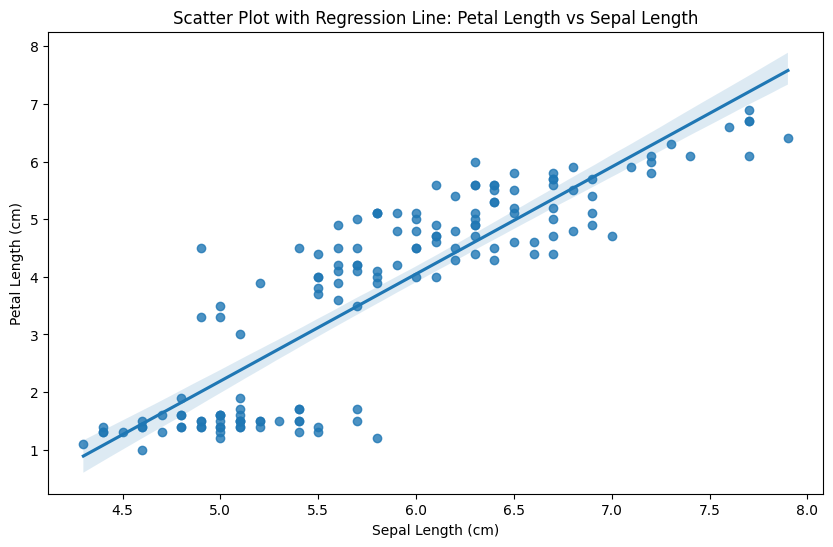
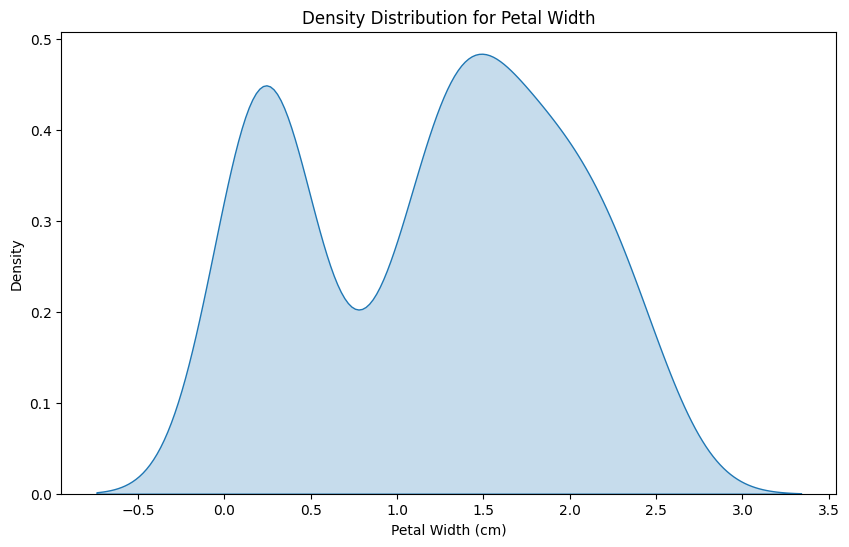
plt.title('Correlation Heatmap')

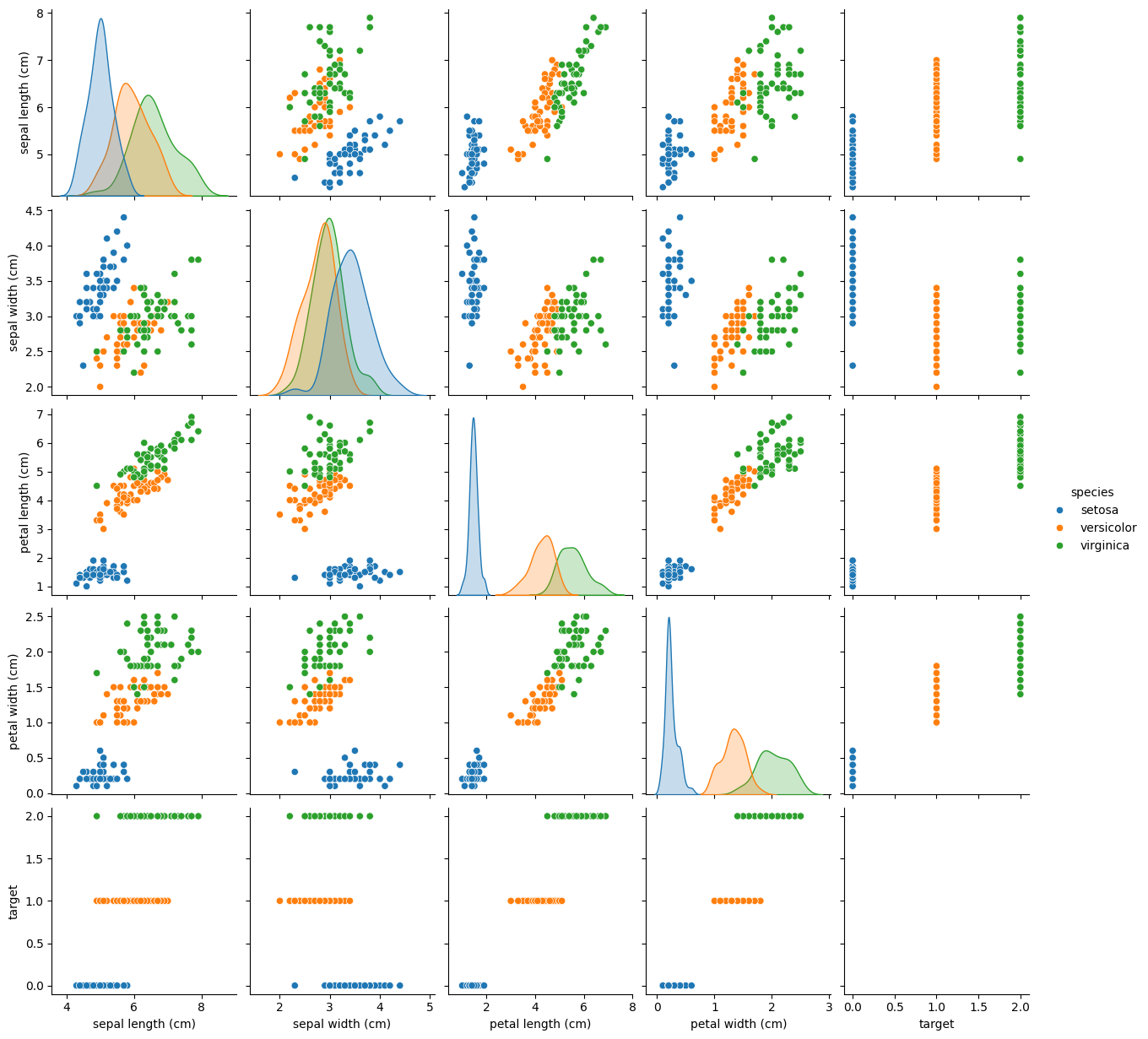
plt.show()

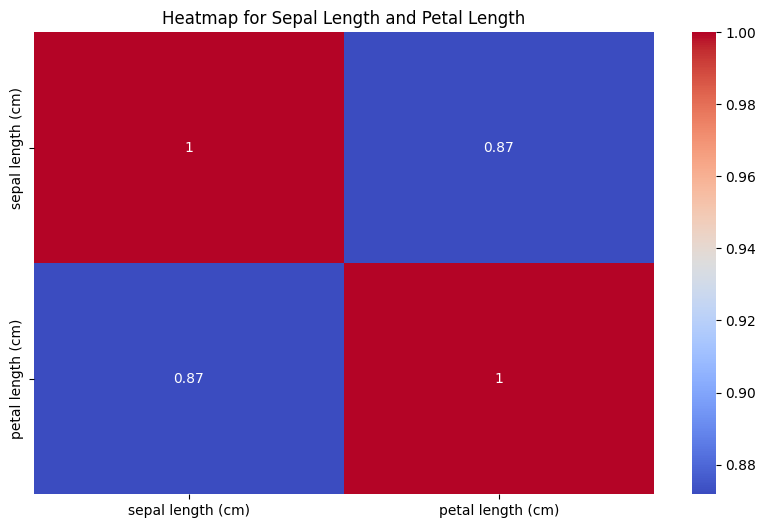
**Output:**

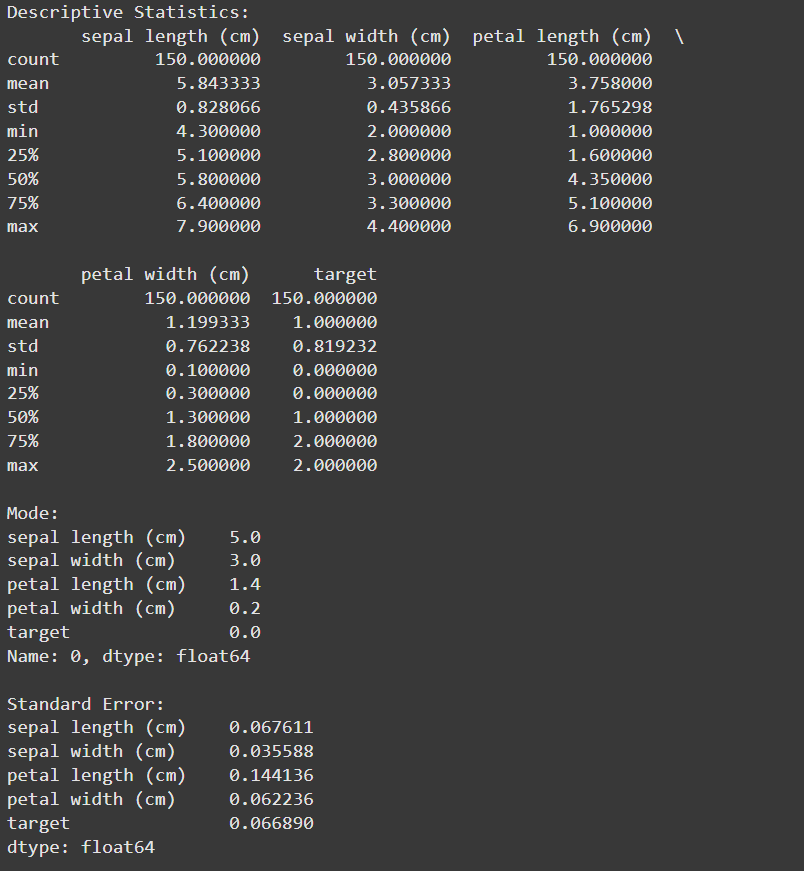


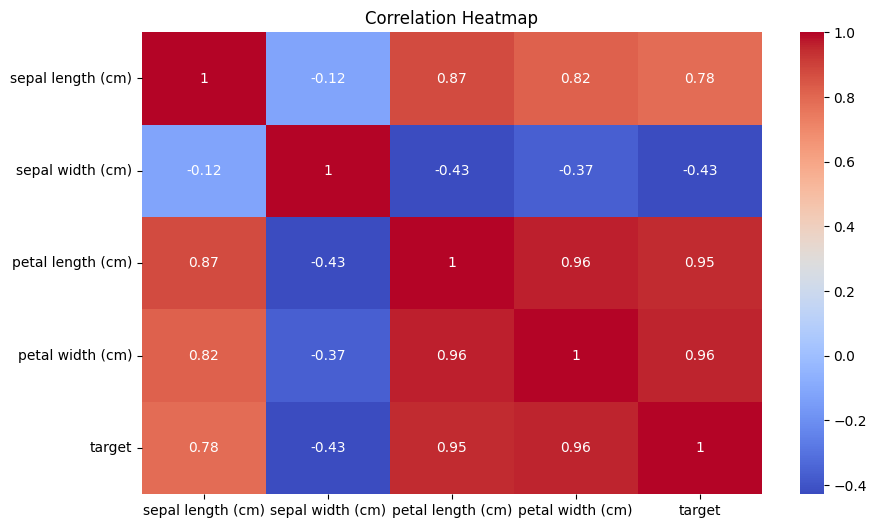










**5. Using Titanic dataset, to do the following:**

**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**

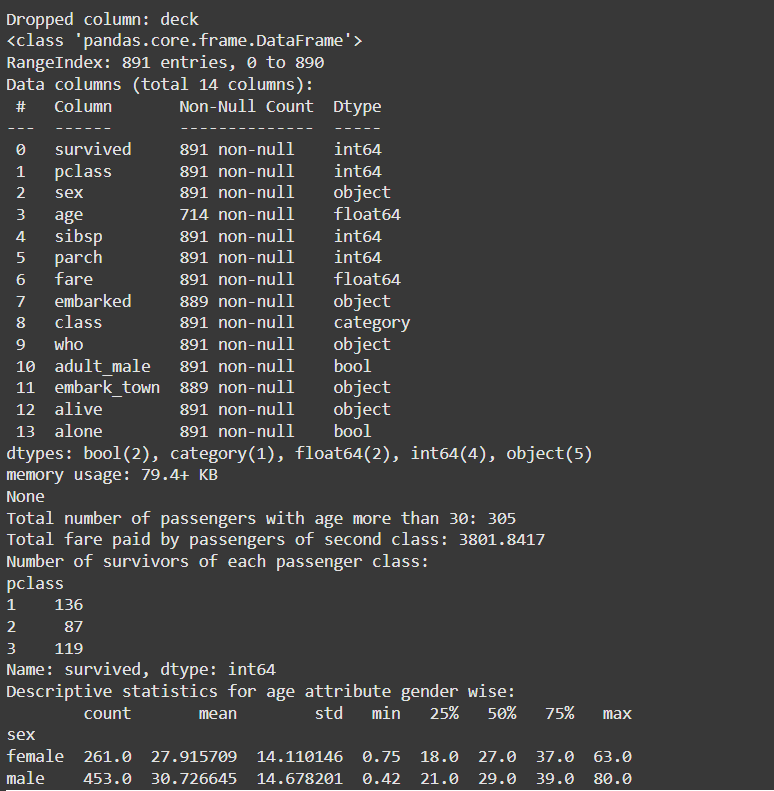
**f. Draw a scatter plot for passenger fare paid by Female and Male passengers separately**

**g. Compare density distribution for features age and passenger fare**

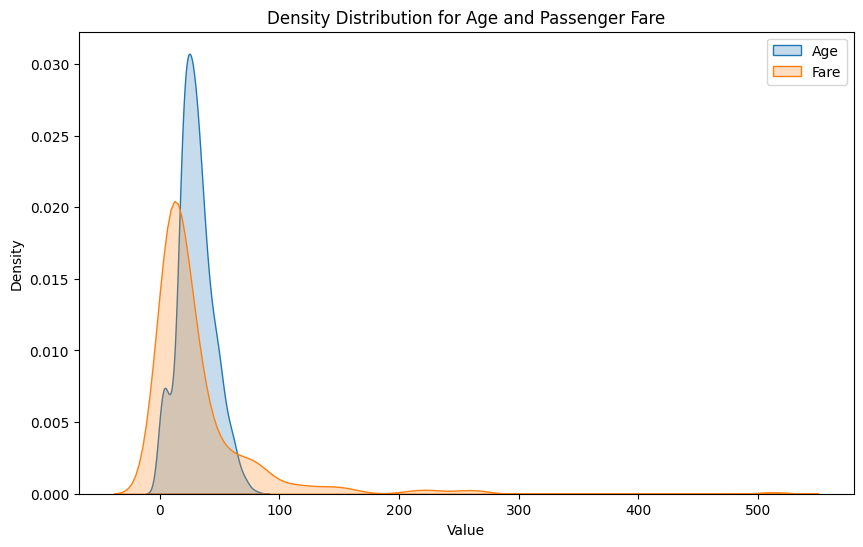
**h. Draw the pie chart for three groups labelled as class 1, class 2, class 3 respectively displayed in different colours. The occurrence of each group converted into percentage should be displayed in the pie chart. Appropriately Label the chart.**

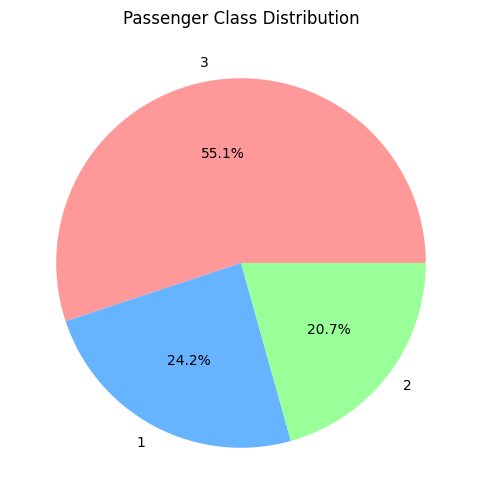
**i. Find % of survived passengers for each class and answer the question “Did class play a role in survival?”.**

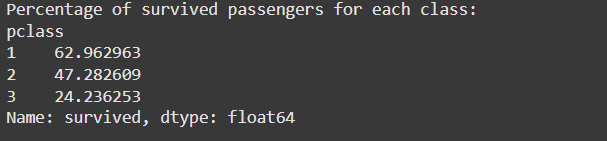
**Code:**

****







****

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

**Code:**

import pandas as pd

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\_income = df.groupby('FamilyName')['MonthlyIncome'].max()

lowest\_income = df.groupby('FamilyName')['MonthlyIncome'].min()

print("\nHighest Monthly Income for each family:")

print(highest\_income)

print("\nLowest Monthly Income for each family:")

print(lowest\_income)

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

low\_income\_members = df[df['MonthlyIncome'] < 80000.00]

print("\nMonthly Income of members earning less than Rs. 80000.00:")

print(low\_income\_members)

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

female\_data = df[df['Gender'] == 'Female']

total\_females = female\_data['Gender'].count()

average\_female\_income = female\_data['MonthlyIncome'].mean()

print("\nTotal number of females:", total\_females)

print("Average monthly income of females: Rs.", average\_female\_income)

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

average\_income\_all = df['MonthlyIncome'].mean()

filtered\_df = df[df['MonthlyIncome'] >= average\_income\_all]

print("\nDataFrame after deleting rows with monthly income less than average income:")

print(filtered\_df)

**Output:**

