1.

dict = {'Boys':[72,68,70,69,74],'Girls':[63,65,69,62,61]}

print(dict)

zipped = list({'Boys':b,'Girls':g}for b,g in zip(dict['Boys'],dict['Girls']))

print(zipped)

2.

a.

import numpy as np

ra = np.random.randint(1,10 ,size = (3,5))

print(ra)

amean = np.mean(ra,axis=1)

astd=np.std(ra,axis=1)

avar=np.var(ra,axis=1)

print("Mean:",amean)

print("Standard Deviation:",astd)

print("Variance:",avar)

b.

import numpy as np

B = np.array([56,48,22,41,78,91,24,46,8,33])

print(B)

print(np.sort(B))

print(np.argsort(B))

c.

import numpy as np

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

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

arr=np.random.randint(1 , 10 , size=(m,n))

print(arr)

print(arr.shape)

print(type(arr))

print(arr.dtype)

newarr = arr.reshape(n,m);

print(newarr)

d.

import numpy as np

# Given array

given\_array = np.array([1.0, 0.0, 3.5, 0.0, np.nan, 7.2, np.nan])

# Test for zero, non-zero, and NaN

zero\_indices = np.where(given\_array == 0)[0]

nonzero\_indices = np.where(given\_array != 0)[0]

nan\_indices = np.where(np.isnan(given\_array))[0]

print("Given Array:")

print(given\_array)

print("\nIndices of Zero Elements:")

print(zero\_indices)

print("\nIndices of Non-Zero Elements:")

print(nonzero\_indices)

print("\nIndices of NaN Elements:")

print(nan\_indices)

3.

import pandas as pd

import numpy as np

data = {

'Column1' : np.random.rand(50),

'Column2' : np.random.rand(50),

'Column3' : np.random.randn(50)

}

df=pd.DataFrame(data)

print(df)

null\_indices = np.random.choice(df.size , size=int(0.1\*df.size) , replace=False)

df.values.flat[null\_indices]=np.nan

print("New DataFrame - ")

print(df)

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

max\_row\_index=df.sum(axis=1).idxmax()

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

print("Dataframe after dropping row with max sum - ")

print(df)

df=df.sort\_values(by='Column1')

print("Dataframe after sorting on column 1 - ")

print(df)

df=df.drop\_duplicates(subset='Column1')

print("DataFrame after removing duplicates from column 1 - ")

print(df)

correlation = df['Column1'].corr(df['Column2'])

covariance = df['Column2'].cov(df['Column3'])

print("Correlation - ",correlation)

print("Covariance - ",covariance)

# g. Detect outliers and remove rows having outliers

z\_scores = (df - df.mean()) / df.std()

outliers\_indices = (z\_scores > 3) | (z\_scores < -3)

df = df[~outliers\_indices.any(axis=1)]

print("\nDataFrame after removing rows with outliers:")

print(df)

df['Column2'] = pd.to\_numeric(df['Column2'], errors='coerce')

df = df.dropna(subset=['Column2'])

# h. Discretize the second column and create 5 bins

df['Column2\_bins'] = pd.cut(df['Column2'], bins=5)

print("\nDataFrame with second column discretized into 5 bins:")

print(df)

4.

import pandas as pd

# Read data from two Excel files into dataframes

file1\_path = 'D:\DAV\workshop1.xlsx' # Replace with the actual file path

file2\_path = 'D:\DAV\workshop2.xlsx' # Replace with the actual file path

df\_day1 = pd.read\_excel(file1\_path)

df\_day2 = pd.read\_excel(file2\_path)

# a. Perform merging to find the names of students who attended the workshop on both days

common\_attendees = pd.merge(df\_day1, df\_day2, on='Name', how='inner')

print("Names of students who attended the workshop on both days:")

print(common\_attendees['Name'])

# b. Find names of all students who attended the workshop on either of the days

all\_attendees = pd.merge(df\_day1, df\_day2, on='Name', how='outer')

print("\nNames of all students who attended the workshop on either of the days:")

print(all\_attendees['Name'])

# c. Merge two data frames row-wise and find the total number of records in the data frame

merged\_rowwise = pd.concat([df\_day1, df\_day2], ignore\_index=True)

print("\nTotal number of records after merging row-wise:")

print(len(merged\_rowwise))

# d. Merge two data frames and use two columns 'Name' and 'Duration' as multi-row indexes.

# Generate descriptive statistics for this multi-index.

merged\_multi\_index = pd.merge(df\_day1, df\_day2, on=['Name', 'Duration'], how='outer')

multi\_index\_stats = merged\_multi\_index.groupby(['Name', 'Duration']).describe()

print("\nDescriptive statistics for multi-index (Name and Duration):")

print(multi\_index\_stats)

5.

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import load\_iris

# Load Iris dataset

iris = load\_iris()

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

iris\_df['Species'] = iris.target\_names[iris.target]

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

plt.figure(figsize=(8, 6))

sns.countplot(x='Species', data=iris\_df, palette='viridis')

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

plt.xlabel('Species')

plt.ylabel('Count')

plt.show()

# b. Draw a scatter plot for Petal width vs Sepal width

plt.figure(figsize=(8, 6))

sns.scatterplot(x='petal width (cm)', y='sepal width (cm)', hue='Species', data=iris\_df, palette='viridis')

plt.title('Scatter Plot: Petal width vs Sepal width')

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

plt.ylabel('Sepal Width (cm)')

plt.legend()

plt.show()

# c. Plot density distribution for feature petal length

plt.figure(figsize=(8, 6))

sns.kdeplot(iris\_df['petal length (cm)'], shade=True, color='skyblue')

plt.title('Density Distribution: Petal Length')

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

plt.ylabel('Density')

plt.show()

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

plt.figure(figsize=(12, 10))

sns.pairplot(iris\_df, hue='Species', palette='viridis')

plt.suptitle('Pairwise Bivariate Distribution in Iris Dataset', y=1.02)

plt.show()

6.

import pandas as pd

# Assuming you have a sales training dataset with columns 'sales' and 'region'

# df = pd.read\_csv('sales\_training.csv')

# Example:

df = pd.DataFrame({

'sales': [100, 150, 200, 120, 180],

'region': ['North', 'South', 'North', 'South', 'North']

})

# Compute mean of 'sales' grouped by 'region'

mean\_sales\_by\_region = df.groupby('region')['sales'].mean()

print("Mean sales by region:")

print(mean\_sales\_by\_region)

# Assuming you have a weather forecasting dataset with columns 'date' and 'temperature'

# df\_weather = pd.read\_csv('weather\_forecast.csv')

# Example:

df\_weather = pd.DataFrame({

'date': ['2022-01-01', '2022-01-02', '2022-01-04'],

'temperature': [20, 22, 25]

})

# Convert 'date' to datetime

df\_weather['date'] = pd.to\_datetime(df\_weather['date'])

df\_weather = df\_weather.set\_index('date')

# Fill missing dates with values of previous non-missing date

df\_weather = df\_weather.resample('D').ffill()

print("DataFrame with filled intermittent time series:")

print(df\_weather)

# Assuming you have a dataset with a column 'year\_month' in string format

# df\_dates = pd.read\_csv('dates\_dataset.csv')

# Example:

df\_dates = pd.DataFrame({

'year\_month': ['2022-01', '2022-02', '2022-03']

})

# Convert 'year\_month' to datetime

df\_dates['year\_month'] = pd.to\_datetime(df\_dates['year\_month'])

print("DataFrame with converted dates:")

print(df\_dates)

# Assuming you have a dataset with columns 'product', 'region', and 'sales'

# df\_sales = pd.read\_csv('sales\_dataset.csv')

# Example:

df\_sales = pd.DataFrame({

'product': ['A', 'B', 'A', 'B', 'A'],

'region': ['North', 'South', 'North', 'South', 'North'],

'sales': [100, 150, 200, 120, 180]

})

# Group by 'product' and 'region', then sort the aggregated results within the groups

sorted\_sales = df\_sales.groupby(['product', 'region'])['sales'].sum().sort\_values()

print("Sorted aggregated sales within groups:")

print(sorted\_sales)

# Assuming you have a dataset with a column 'value'

# df\_values = pd.read\_csv('values\_dataset.csv')

# Example:

df\_values = pd.DataFrame({

'value': [5, 12, 18, 22, 8, 15, 30, 40, 10]

})

# Split the dataframe into groups with bin counts

bin\_counts = pd.cut(df\_values['value'], bins=[0, 10, 20, 30, 40])

grouped\_bins = df\_values.groupby(bin\_counts).size()

print("Groups with bin counts:")

print(grouped\_bins)

7.

import numpy as np

import pandas as pd

df = pd.read\_csv('D:\DAV\Students-info.csv')

df

#a

df1 = pd.get\_dummies(df, columns=['Gender', 'Pass\_Division'])

df1

#b

months = ["January", "February", "March", "April", "May", "June", "July",

"August", "September", "October", "November", "December"]

df1['Birth\_Month'] = pd.Categorical(df1['Birth\_Month'], categories=months, ordered=True)

df1.sort\_values(by='Birth\_Month', inplace=True)

df1

8.

import numpy as np

import pandas as pd

df = pd.read\_csv('F:\CS\sem 5\Data Analysis and Visualisation\Practicals/family\_data8.csv')

df

#a

df.groupby('Name')['Monthly Income (Rs.)'].sum()

#b

df1 = pd.DataFrame(df.groupby('Name')['Monthly Income (Rs.)'].max())

df1

#c

df[df['Monthly Income (Rs.)'] > 60000]

#d

df[(df['Name'] == 'Shah') & (df['Gender'] == 'Female')]['Monthly Income (Rs.)'].mean()