



Course : B. Sc. (h) Computer Science

Year : III

Semester : V

Name : Raman

College Rollno : CSC/20/26

University Rollno : 20059570019

Data Analysis and Visualisation Practical File

Submitted to :- Mr. Rajeev Rai

INDEX

Sno	Practical Questions	Page No
1	<p>Ques1. Given below is a dictionary having two keys 'Boys' and 'Girls' and having two lists of heights of five Boys and Five Girls respectively as values associated with these keys Original dictionary of lists: {'Boys': [72, 68, 70, 69, 74], 'Girls': [63, 65, 69, 62, 61]}</p> <p>From the given dictionary of lists create the following list of dictionaries: [{'Boys': 72, 'Girls': 63}, {'Boys': 68, 'Girls': 65}, {'Boys': 70, 'Girls': 69}, {'Boys': 69, 'Girls': 62}, {'Boys': 74, 'Girls': 61}]</p>	5
2	<p>Ques2. Write programs in Python using NumPy library to do the following:</p> <p>a) Compute the mean, standard deviation, and variance of a two dimensional random integer array along the second axis.</p> <p>b) Get the indices of the sorted elements of a given array. a. B = [56, 48, 22, 41, 78, 91, 24, 46, 8, 33]</p> <p>c) 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 nx m array, n and m are user inputs given at the run time.</p> <p>d) Test whether the elements of a given array are zero, non-zero and NaN. Record the indices of these elements in three separate arrays.</p>	6
3	<p>Ques3. Create a dataframe 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.</p> <p>Do the following:</p> <p>a. Identify and count missing values in a dataframe.</p> <p>b. Drop the column having more than 5 null values.</p> <p>c. Identify the row label having maximum of the sum of all values in a row and drop that row.</p> <p>d. Sort the dataframe on the basis of the first column.</p> <p>e. Remove all duplicates from the first column.</p> <p>f. Find the correlation between first and second column and covariance between second and third column.</p> <p>g. Detect the outliers and remove the rows having outliers.</p> <p>h. Discretize second column and create 5 bins</p>	10
4	<p>Ques4. Consider two excel files having attendance of a workshop's participants for two days. Each file has three fields 'Name', 'Time of joining', duration (in minutes) where names are unique within a file. Note that duration may take one of three values (30, 40, 50) only. Import the data into two dataframes and do the following:</p> <p>a. Perform merging of the two dataframes to find the names of</p>	19

	<p>students who had attended the workshop on both days.</p> <p>b. Find names of all students who have attended workshop on either of the days.</p> <p>c. Merge two data frames row-wise and find the total number of records in the data frame</p> <p>d. Merge two data frames and use two columns names and duration as multi-row indexes. Generate descriptive statistics for this multi-index.</p>																																																																																						
5	<p>Ques5. Taking 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 <code>sklearn.datasets</code>)</p> <p>a. Plot bar chart to show the frequency of each class label in the data.</p> <p>b. Draw a scatter plot for Petal width vs sepal width.</p> <p>c. Plot density distribution for feature petal length.</p> <p>d. Use a pair plot to show pairwise bivariate distribution in the Iris Dataset.</p>	25																																																																																					
6	<p>Ques6. Consider any sales training/ weather forecasting dataset</p> <p>a. Compute mean of a series grouped by another series</p> <p>b. Fill an intermittent time series to replace all missing dates with values of previous non-missing date.</p> <p>c. Perform appropriate year-month string to dates conversion.</p> <p>d. Split a dataset to group by two columns and then sort the aggregated results within the groups.</p> <p>e. Split a given dataframe into groups with bin counts</p>	31																																																																																					
7	<p>Ques7. Consider a data frame containing data about students i.e. name, gender and passing division:</p> <table border="1"><thead><tr><th></th><th>Name</th><th>Birth_Month</th><th>Gender</th><th>Pass_Division</th></tr></thead><tbody><tr><td>0</td><td>Mudit Chauhan</td><td>December</td><td>M</td><td>III</td></tr><tr><td>1</td><td>Seema Chopra</td><td>January</td><td>F</td><td>II</td></tr><tr><td>2</td><td>Rani Gupta</td><td>March</td><td>F</td><td>I</td></tr><tr><td>3</td><td>Aditya Narayan</td><td>October</td><td>M</td><td>I</td></tr><tr><td>4</td><td>Sanjeev Sahni</td><td>February</td><td>M</td><td>II</td></tr><tr><td>5</td><td>Prakash Kumar</td><td>December</td><td>M</td><td>III</td></tr><tr><td>6</td><td>Ritu Agarwal</td><td>September</td><td>F</td><td>I</td></tr><tr><td>7</td><td>Akshay Goel</td><td>August</td><td>M</td><td>I</td></tr><tr><td>8</td><td>Meeta Kulkarni</td><td>July</td><td>F</td><td>II</td></tr><tr><td>9</td><td>Preeti Ahuja</td><td>November</td><td>F</td><td>II</td></tr><tr><td>10</td><td>Sunil Das Gupta</td><td>April</td><td>M</td><td>III</td></tr><tr><td>11</td><td>Sonali Sapre</td><td>January</td><td>F</td><td>I</td></tr><tr><td>12</td><td>Rashmi Talwar</td><td>June</td><td>F</td><td>III</td></tr><tr><td>13</td><td>Ashish Dubey</td><td>May</td><td>M</td><td>II</td></tr><tr><td>14</td><td>Kiran Sharma</td><td>February</td><td>F</td><td>II</td></tr><tr><td>15</td><td>Sameer Bansal</td><td>October</td><td>M</td><td>I</td></tr></tbody></table> <p>a. Perform one hot encoding of the last two columns of categorical data using the <code>get_dummies()</code> function.</p> <p>b. Sort this data frame on the "Birth Month" column (i.e. January to December). Hint: Convert Month to Categorical.</p>		Name	Birth_Month	Gender	Pass_Division	0	Mudit Chauhan	December	M	III	1	Seema Chopra	January	F	II	2	Rani Gupta	March	F	I	3	Aditya Narayan	October	M	I	4	Sanjeev Sahni	February	M	II	5	Prakash Kumar	December	M	III	6	Ritu Agarwal	September	F	I	7	Akshay Goel	August	M	I	8	Meeta Kulkarni	July	F	II	9	Preeti Ahuja	November	F	II	10	Sunil Das Gupta	April	M	III	11	Sonali Sapre	January	F	I	12	Rashmi Talwar	June	F	III	13	Ashish Dubey	May	M	II	14	Kiran Sharma	February	F	II	15	Sameer Bansal	October	M	I	35
	Name	Birth_Month	Gender	Pass_Division																																																																																			
0	Mudit Chauhan	December	M	III																																																																																			
1	Seema Chopra	January	F	II																																																																																			
2	Rani Gupta	March	F	I																																																																																			
3	Aditya Narayan	October	M	I																																																																																			
4	Sanjeev Sahni	February	M	II																																																																																			
5	Prakash Kumar	December	M	III																																																																																			
6	Ritu Agarwal	September	F	I																																																																																			
7	Akshay Goel	August	M	I																																																																																			
8	Meeta Kulkarni	July	F	II																																																																																			
9	Preeti Ahuja	November	F	II																																																																																			
10	Sunil Das Gupta	April	M	III																																																																																			
11	Sonali Sapre	January	F	I																																																																																			
12	Rashmi Talwar	June	F	III																																																																																			
13	Ashish Dubey	May	M	II																																																																																			
14	Kiran Sharma	February	F	II																																																																																			
15	Sameer Bansal	October	M	I																																																																																			

8	<p>Ques8. Consider the following data frame containing a family name, gender of the family member and her/his monthly income in each record.</p> <table> <tr> <th>Name</th><th>Gender</th><th>MonthlyIncome (Rs.)</th></tr> <tr> <td>Shah</td><td>Male</td><td>114000.00</td></tr> <tr> <td>Vats</td><td>Male</td><td>65000.00</td></tr> <tr> <td>Vats</td><td>Female</td><td>43150.00</td></tr> <tr> <td>Kumar</td><td>Female</td><td>69500.00</td></tr> <tr> <td>Vats</td><td>Female</td><td>155000.00</td></tr> <tr> <td>Kumar</td><td>Male</td><td>103000.00</td></tr> <tr> <td>Shah</td><td>Male</td><td>55000.00</td></tr> <tr> <td>Shah</td><td>Female</td><td>112400.00</td></tr> <tr> <td>Kumar</td><td>Female</td><td>81030.00</td></tr> <tr> <td>Vats</td><td>Male</td><td>71900.00</td></tr> </table> <p>Write a program in Python using Pandas to perform the following:</p> <ol style="list-style-type: none"> Calculate and display familywise gross monthly income. Calculate and display the member with the highest monthly income in a family. Calculate and display monthly income of all members with income greater than Rs. 60000.00. Calculate and display the average monthly income of the female members in the Shah family. 	Name	Gender	MonthlyIncome (Rs.)	Shah	Male	114000.00	Vats	Male	65000.00	Vats	Female	43150.00	Kumar	Female	69500.00	Vats	Female	155000.00	Kumar	Male	103000.00	Shah	Male	55000.00	Shah	Female	112400.00	Kumar	Female	81030.00	Vats	Male	71900.00	38
Name	Gender	MonthlyIncome (Rs.)																																	
Shah	Male	114000.00																																	
Vats	Male	65000.00																																	
Vats	Female	43150.00																																	
Kumar	Female	69500.00																																	
Vats	Female	155000.00																																	
Kumar	Male	103000.00																																	
Shah	Male	55000.00																																	
Shah	Female	112400.00																																	
Kumar	Female	81030.00																																	
Vats	Male	71900.00																																	

Ques 1.

Given below is a dictionary having two keys 'Boys' and 'Girls' and having two lists of heights of five Boys and Five Girls respectively as values associated with these keys

Original dictionary of lists: {'Boys': [72, 68, 70, 69, 74], 'Girls': [63, 65, 69, 62, 61]}

From the given dictionary of lists create the following list of dictionaries: [{'Boys': 72, 'Girls': 63}, {'Boys': 68, 'Girls': 65}, {'Boys': 70, 'Girls': 69}, {'Boys': 69, 'Girls': 62}, {'Boys': 74, 'Girls': 61}].

Code :

```
#RAMAN CSC/20/26 Univ_Roll_No: 20059570019
```

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

```
keys=list(Dict.keys())
```

```
data1=Dict[keys[0]]
```

```
data2=Dict[keys[1]]
```

```
j=len(data1)
```

```
final=[]
```

```
for i in range(j):
```

```
    t1={}
```

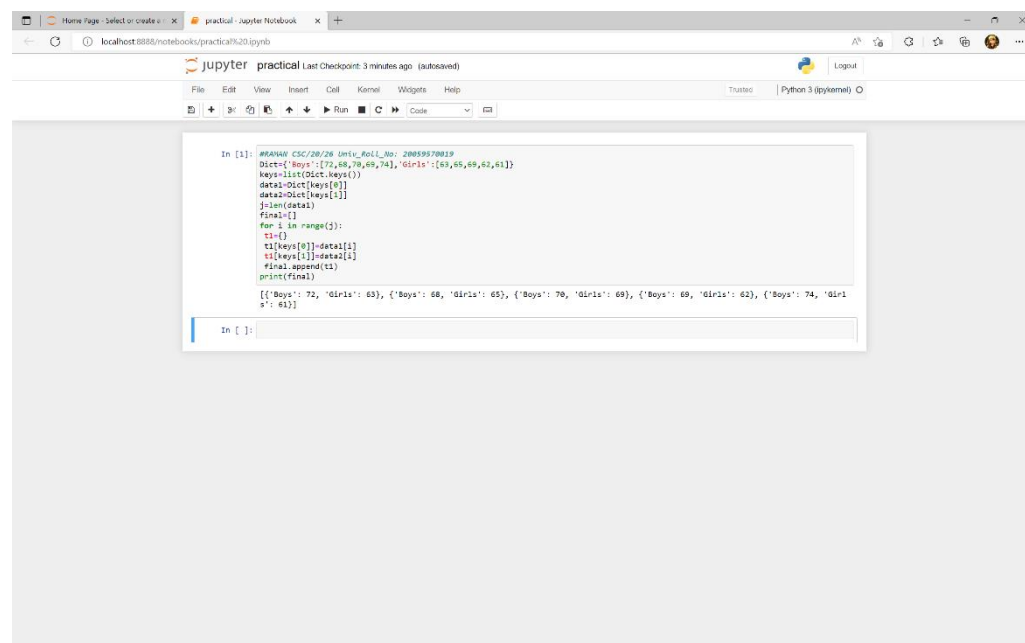
```
    t1[keys[0]]=data1[i]
```

```
    t1[keys[1]]=data2[i]
```

```
    final.append(t1)
```

```
print(final)
```

OUTPUT :-



```
In [1]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
Dict={'Boys':[72,68,70,69,74],'Girls':[63,65,69,62,61]}
keys=list(Dict.keys())
data1=Dict[keys[0]]
data2=Dict[keys[1]]
j=len(data1)
final=[]
for i in range(j):
    t1={}
    t1[keys[0]]=data1[i]
    t1[keys[1]]=data2[i]
    final.append(t1)
print(final)

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

In [ ]:
```

Ques 2.

Write programs in Python using NumPy library to do the following:

a) Compute the mean, standard deviation, and variance of a two dimensional random integer array along the second axis.

b) Get the indices of the sorted elements of a given array.

a. B = [56, 48, 22, 41, 78, 91, 24, 46, 8, 33]

c) 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 nx m array, n and m are user inputs given at the run time.

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

Code a) :

```
#RAMAN CSC/20/26 Univ_Roll_No: 20059570019
```

```
import numpy as np
```

```
import pandas as pd
```

```
data=np.random.rand(10,2)
```

```
print("\n data along 1st Axis: ",data[:,0])
```

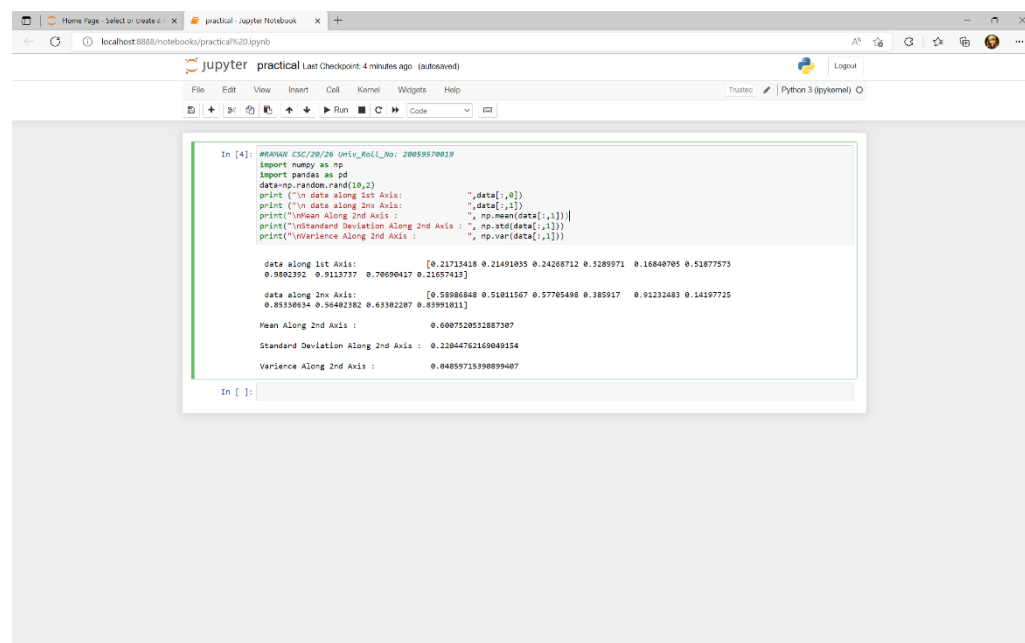
```
print("\n data along 2nx Axis: ",data[:,1])
```

```
print("\nMean Along 2nd Axis : ", np.mean(data[:,1]))
```

```
print("\nStandard Deviation Along 2nd Axis : ", np.std(data[:,1]))
```

```
print("\nVariance Along 2nd Axis : ", np.var(data[:,1]))
```

OUTPUT :-



```
In [4]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
import numpy as np
import pandas as pd
data=np.random.rand(10,2)
print("\n data along 1st Axis: ",data[:,0])
print("\n data along 2nx Axis: ",data[:,1])
print("\nMean Along 2nd Axis : ", np.mean(data[:,1]))
print("\nStandard Deviation Along 2nd Axis : ", np.std(data[:,1]))
print("\nVariance Along 2nd Axis : ", np.var(data[:,1]))

data along 1st Axis:      [0.21713418 0.21491835 0.24268712 0.31289971 0.16848705 0.51877573
0.9802392  0.9113737  0.70690417 0.21657413]
data along 2nx Axis:      [0.58086848 0.51811567 0.57705498 0.385917  0.91232483 0.54197725
0.83336834 0.56482382 0.63302287 0.83991811]
Mean Along 2nd Axis :      0.6087528532887307
Standard Deviation Along 2nd Axis :  0.22844702169049154
Variance Along 2nd Axis :      0.04859715398899487

In [ ]:
```

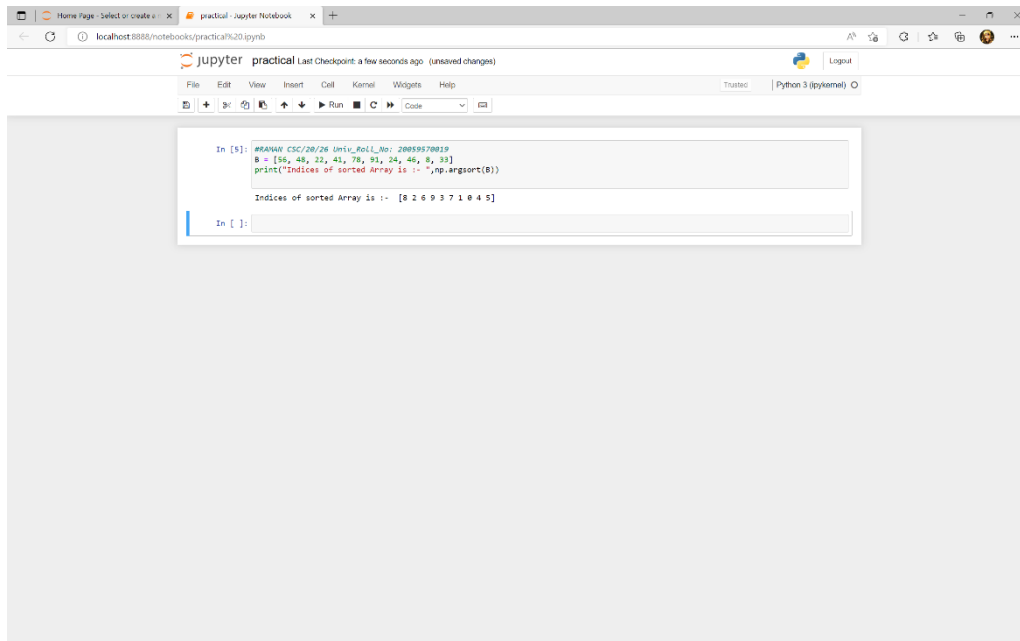
Code b) :

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

B = [56, 48, 22, 41, 78, 91, 24, 46, 8, 33]

print("Indices of sorted Array is :- ",np.argsort(B))

OUTPUT :-



Code c) :

```
#RAMAN CSC/20/26 Univ_Roll_No: 20059570019
```

```
m,n=input("Enter Size of 2D Array :- ").split()
```

```
m=int(m)
```

```
n=int(n)
```

```
data=np.random.rand(m,n)
```

```
print(data)
```

```
print("Shape of the Array is :- ",np.shape(data))
```

```
print("Type of the Array is :- ",type(data[0,0]))
```

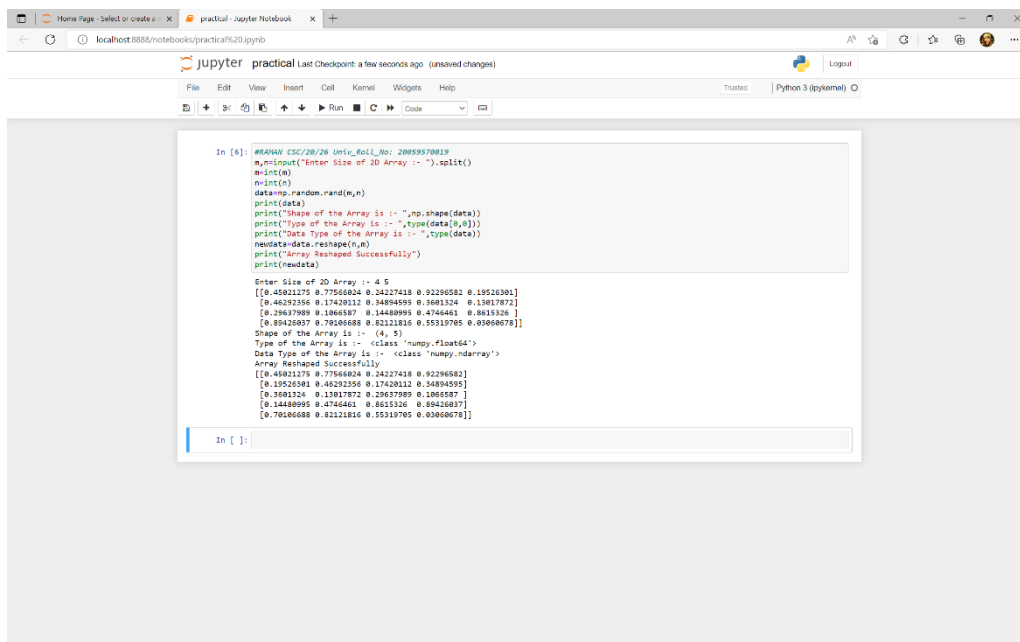
```
print("Data Type of the Array is :- ",type(data))
```

```
newdata=data.reshape(n,m)
```

```
print("Array Reshaped Successfully")
```

```
print(newdata)
```

OUTPUT :-



```
In [6]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
m,n=input("Enter Size of 2D Array :- ").split()
m=int(m)
n=int(n)
data=np.random.rand(m,n)
print(data)
print("Shape of the Array is :- ",np.shape(data))
print("Type of the Array is :- ",type(data[0,0]))
print("Data Type of the Array is :- ",type(data))
newdata=data.reshape(n,m)
print("Array Reshaped Successfully")
print(newdata)

Enter Size of 2D Array :- 4 5
[[0.45021275 0.77566824 0.24227418 0.92296582 0.19526381]
 [0.46293356 0.17420212 0.34894595 0.3681324 0.13017872]
 [0.29637989 0.1066587 0.14480995 0.4746461 0.8615326 ]
 [0.89426857 0.70206688 0.82121816 0.55319705 0.03060678]]
Shape of the Array is :- (4, 5)
Type of the Array is :- <class 'numpy.float64'>
Data Type of the Array is :- <class 'numpy.ndarray'>
Array Reshaped Successfully
[[0.45021275 0.77566824 0.24227418 0.92296582]
 [0.19526381 0.46293356 0.17420212 0.34894595]
 [0.3681324 0.13017872 0.29637989 0.1066587 ]
 [0.14480995 0.4746461 0.8615326 0.89426857]
 [0.70206688 0.82121816 0.55319705 0.03060678]]
```


Code d) :

```
#RAMAN CSC/20/26 Univ_Roll_No: 20059570019
```

```
import math
```

```
size=int(input("Enter Size of Array :- "))
```

```
data=[]
```

```
for i in range(0,size):
```

```
    x=int(input())
```

```
    data.append(x)
```

```
zero=0
```

```
non_zero=0
```

```
nan_count=0
```

```
for i in range(0,size):
```

```
    if(math.isnan(data[i])):
```

```
        nan_count=nan_count+1
```

```
    elif(int(data[i])==0):
```

```
        zero=zero+1
```

```
    elif(int(data[i])!=0):
```

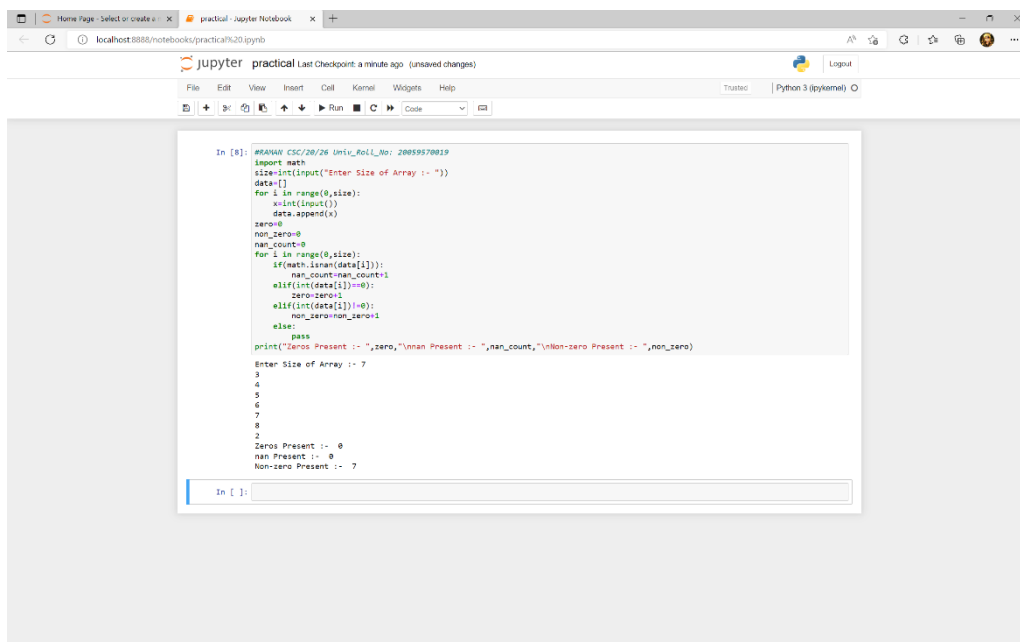
```
        non_zero=non_zero+1
```

```
    else:
```

```
        pass
```

```
print("Zeros Present :- ",zero,"\nnan Present :- ",nan_count,"\nNon-zero Present :- ",non_zero)
```

OUTPUT :-



```
In [8]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
import math
size=int(input("Enter Size of Array :- "))
data=[]
for i in range(0,size):
    x=int(input())
    data.append(x)
zero=0
non_zero=0
nan_count=0
for i in range(0,size):
    if(math.isnan(data[i])):
        nan_count=nan_count+1
    elif(int(data[i])==0):
        zero=zero+1
    elif(int(data[i])!=0):
        non_zero=non_zero+1
    else:
        pass
print("Zeros Present :- ",zero,"\nnan Present :- ",nan_count,"\nNon-zero Present :- ",non_zero)

Enter Size of Array :- 7
3
4
5
6
7
8
2
Zeros Present :- 0
nan Present :- 0
Non-zero Present :- 2
```

Ques 3.

Create a dataframe 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 dataframe.
- 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 dataframe 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 2nd & 3rd column.
- g. Detect the outliers and remove the rows having outliers.
- h. Discretize second column and create 5 bins

Code a) :

```
#RAMAN CSC/20/26 Univ_Roll_No: 20059570019
```

```
import numpy as np
```

```
import pandas as pd
```

```
import math
```

```
firstdata=np.random.randint(1,100,size=(50,3))
```

```
df=pd.DataFrame(firstdata,columns=['x','Y','Z'])
```

```
size2=len(df)*0.1
```

```
size2=int(size2)
```

```
for i in range(0,(size2*3)):
```

```
    df.iat[np.random.randint(1,50,),1]=float("nan")
```

```
count_nan=0
```

```
for i in range(0,len(df.columns)):
```

```
    for j in range(0,len(df)):
```

```
        if(math.isnan(df.iat[j,i])):
```

```
            count_nan=count_nan+1
```

```
print("Number of Missing Values :- ",count_nan)
```

OUTPUT :-

Run

```
In [9]: #RAHMAN CSC/20/26 Univ_Roll_No: 20059570019
import numpy as np
import pandas as pd
import math
firstdata=np.random.randint(1,100,size=(50,3))
df=pd.DataFrame(firstdata,columns=['X','Y','Z'])
size2=len(df)*0.1
size2=int(size2)
for i in range(0,(size2*3)):
    df.iat[np.random.randint(1,50,1),1]=float("nan")
count_nan=0
for i in range(0,len(df.columns)):
    for j in range(0,len(df)):
        if(math.isnan(df.iat[j,i])):
            count_nan=count_nan+1
print("Number of Missing Values :- ",count_nan)
```

Number of Missing Values :- 14

In []:

Code b) :

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

```
for i in range(0,len(df.columns)):

    count_num=0

    for j in range(0,len(df)):

        if(math.isnan(df.iat[j,i])):

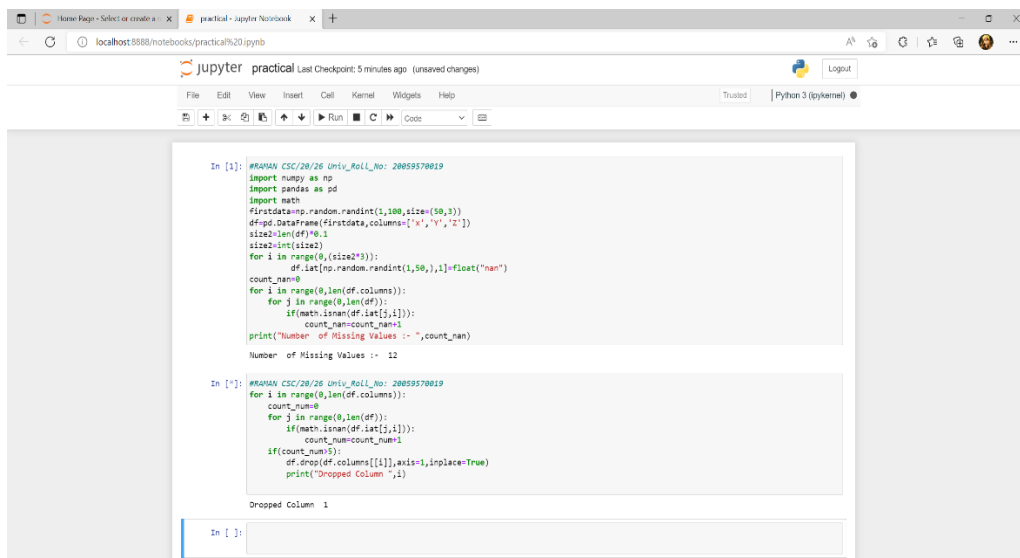
            count_num=count_num+1

    if(count_num>5):

        df.drop(df.columns[[i]],axis=1,inplace=True)

    print("Dropped Column ",i)
```

OUTPUT :-



The screenshot shows a Jupyter Notebook window titled 'practical - Jupyter Notebook'. The interface includes a top bar with navigation icons and a 'Logout' button. Below the top bar is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A toolbar contains icons for running, saving, and other actions. The notebook area displays two code cells. The first cell, labeled 'In [1]:', contains code that imports numpy and pandas, generates random data, and prints the number of missing values, which is 12. The second cell, labeled 'In [2]:', contains code that iterates through the columns of the DataFrame, counts missing values, and drops columns with more than 5 missing values. The output of the second cell is 'Dropped Column 1'.

```
In [1]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
import numpy as np
import pandas as pd
import math
firstdata=np.random.randint(1,100,size=(50,3))
df=pd.DataFrame(firstdata,columns=['x','y','z'])
size=len(df)*0.1
size2=int(size2)
for i in range(0,(size2*3)):
    df.iat[np.random.randint(1,50),i]=float("nan")
count_nan=0
for i in range(0,len(df.columns)):
    for j in range(0,len(df)):
        if(math.isnan(df.iat[j,i])):
            count_nan=count_nan+1
print("Number of Missing Values :- ",count_nan)
Number of Missing Values :- 12

In [2]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
for i in range(0,len(df.columns)):
    count_num=0
    for j in range(0,len(df)):
        if(math.isnan(df.iat[j,i])):
            count_num=count_num+1
    if(count_num>5):
        df.drop(df.columns[[i]],axis=1,inplace=True)
    print("Dropped Column ",i)

Dropped Column 1

In [ ]:
```

Code c) :

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

```
arr=[]
```

```
for j in range(0,len(df)):
```

```
    count_num=0
```

```
    for i in range(0,len(df.columns)):
```

```
        count_num=count_num+df.iat[j,i]
```

```
    arr.append(count_num)
```

```
x=np.argsort(arr)
```

```
x=x[len(x)-1]
```

```
df.drop(x)
```

```
print("Dropped row with sum of elements as maximum value")
```

OUTPUT :-

```
size=int(size2)
for i in range(0,size2*3):
    df.iat[np.random.randint(1,50,,1)]=float("nan")
count_num=0
for i in range(0,len(df.columns)):
    for j in range(0,len(df)):
        if(math.isnan(df.iat[j,i])):
            count_num=count_num+1
print("Number of Missing Values :- ",count_num)
Number of Missing Values :- 12

In [ ]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
for i in range(0,len(df.columns)):
    count_num=0
    for j in range(0,len(df)):
        if(math.isnan(df.iat[j,i])):
            count_num=count_num+1
    if(count_num>5):
        df.drop(df.columns[[i]],axis=1,inplace=True)
        print("Dropped Column ",i)

In [ ]:

In [3]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
arr=[]
for j in range(0,len(df)):
    count_num=0
    for i in range(0,len(df.columns)):
        count_num=count_num+df.iat[j,i]
    arr.append(count_num)
x=np.argsort(arr)
x=x[len(x)-1]
df.drop(x)
print("Dropped row with sum of elements as maximum value")

Dropped row with sum of elements as maximum value

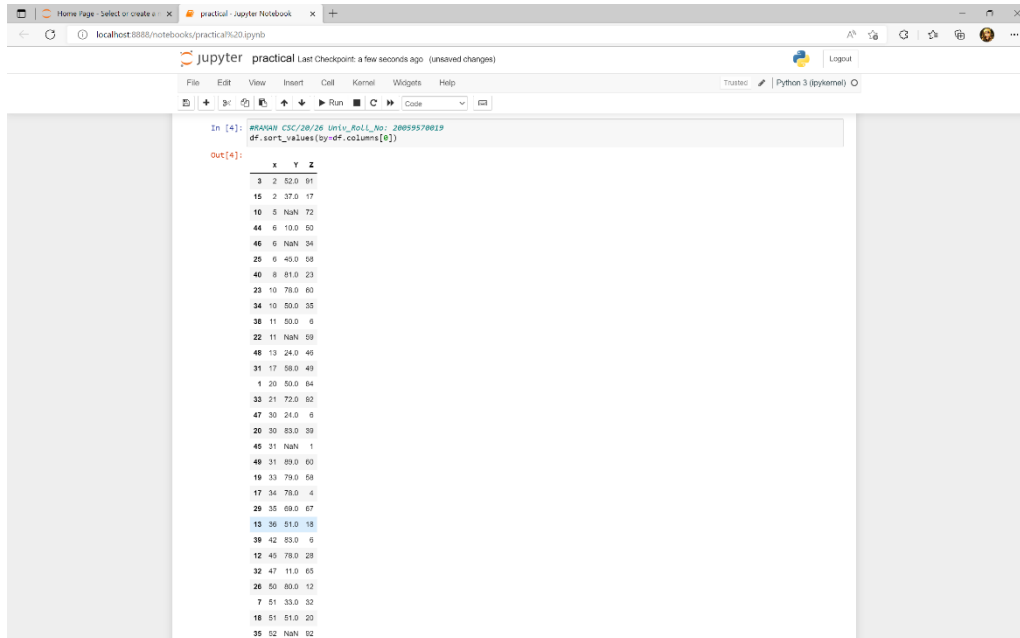
In [ ]:
```

Code d) :

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

```
df.sort_values(by=df.columns[0])
```

OUTPUT :-



```
In [4]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
df.sort_values(by=df.columns[0])
```

Out[4]:

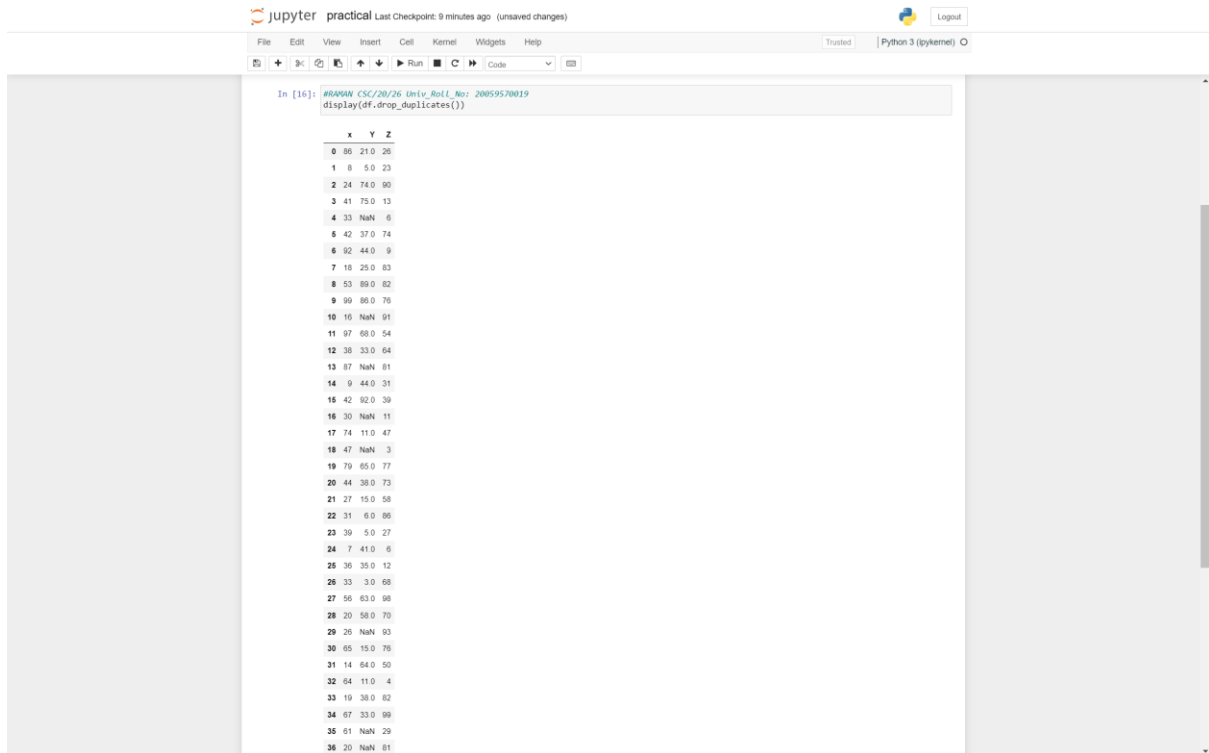
	X	Y	Z
3	2	52.0	91
15	2	37.0	17
10	5	NaN	72
44	6	10.0	50
46	6	NaN	54
25	6	40.0	08
40	8	81.0	23
23	10	79.0	60
34	10	50.0	55
38	11	50.0	6
22	11	NaN	59
48	13	24.0	46
31	17	58.0	49
1	20	50.0	84
33	21	72.0	82
47	30	24.0	6
20	30	83.0	39
45	31	NaN	1
49	31	89.0	80
19	33	79.0	58
17	34	78.0	4
29	35	69.0	67
13	36	51.0	18
39	42	83.0	6
12	45	79.0	28
32	47	11.0	65
28	50	85.0	12
7	51	33.0	52
18	51	51.0	20
35	52	NaN	62

Code e) :

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

```
display(df.drop_duplicates())
```

OUTPUT :-



The screenshot shows a Jupyter Notebook interface with the following components:

- Header:** jupyter practical Last Checkpoint: 9 minutes ago (unsaved changes) | Python 3 (ipykernel) | Logout
- Menu Bar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help
- Toolbar:** Includes icons for file operations, cell execution, and code management.
- Code Cell:** Contains the code:

```
#RAMAN CSC/20/26 Univ_Roll_No: 20059570019  
display(df.drop_duplicates())
```
- Output:** A table with 36 rows and 3 columns (x, Y, Z). The data is as follows:

	x	Y	Z
0	86	21.0	26
1	8	5.0	23
2	24	74.0	90
3	41	75.0	13
4	33	NaN	6
5	42	37.0	74
6	92	44.0	9
7	18	25.0	83
8	53	89.0	82
9	99	86.0	76
10	16	NaN	91
11	97	68.0	54
12	38	33.0	64
13	87	NaN	81
14	9	44.0	31
15	42	92.0	39
16	30	NaN	11
17	74	11.0	47
18	47	NaN	3
19	79	65.0	77
20	44	38.0	73
21	27	15.0	58
22	31	6.0	86
23	39	5.0	27
24	7	41.0	6
25	36	35.0	12
26	33	3.0	68
27	56	63.0	98
28	20	58.0	70
29	26	NaN	93
30	65	15.0	76
31	14	64.0	50
32	64	11.0	4
33	19	38.0	82
34	67	33.0	99
35	61	NaN	29
36	20	NaN	81

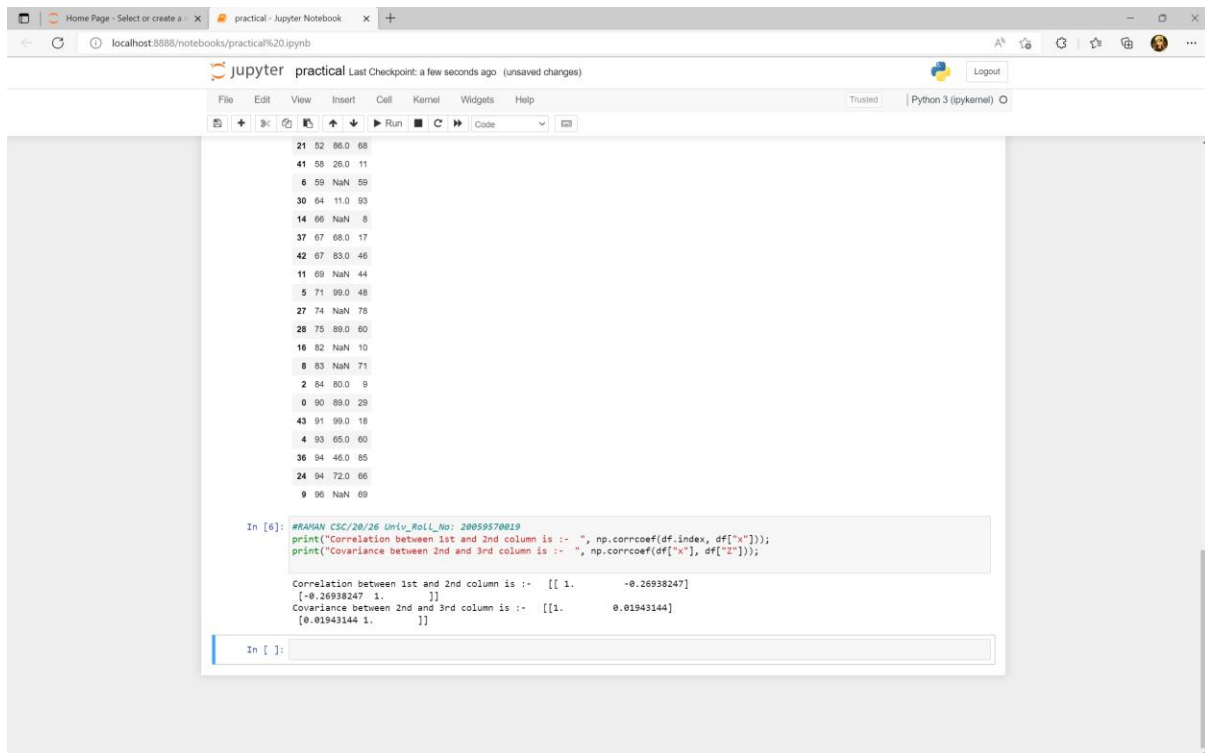
Code f) :

```
#RAMAN CSC/20/26 Univ_Roll_No: 20059570019
```

```
print("Correlation between 1st and 2nd column is :- ", np.corrcoef(df.index, df["x"]));
```

```
print("Covariance between 2nd and 3rd column is :- ", np.corrcoef(df["x"], df["Z"]));
```

OUTPUT :-



The screenshot shows a Jupyter Notebook window with a browser address bar at `localhost:8888/notebooks/practical%20.ipynb`. The notebook contains a data frame with 30 rows and 3 columns. The first column contains indices from 21 to 30, the second column contains values from 52 to 96, and the third column contains values from 68 to 69. Below the data frame, a code cell (In [6]) contains the following Python code:

```
#RAMAN CSC/20/26 Univ_Roll_No: 20059570019
print("Correlation between 1st and 2nd column is :- ", np.corrcoef(df.index, df["x"]));
print("Covariance between 2nd and 3rd column is :- ", np.corrcoef(df["x"], df["Z"]));
```

The output of the code is displayed below the code cell:

```
Correlation between 1st and 2nd column is :- [[ 1.          -0.26938247]
 [-0.26938247  1.          ]]
Covariance between 2nd and 3rd column is :- [[1.          0.01943144]
 [0.01943144  1.          ]]
```


Code g) :

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

```
import seaborn as sns
```

```
df['x'][22]=200
```

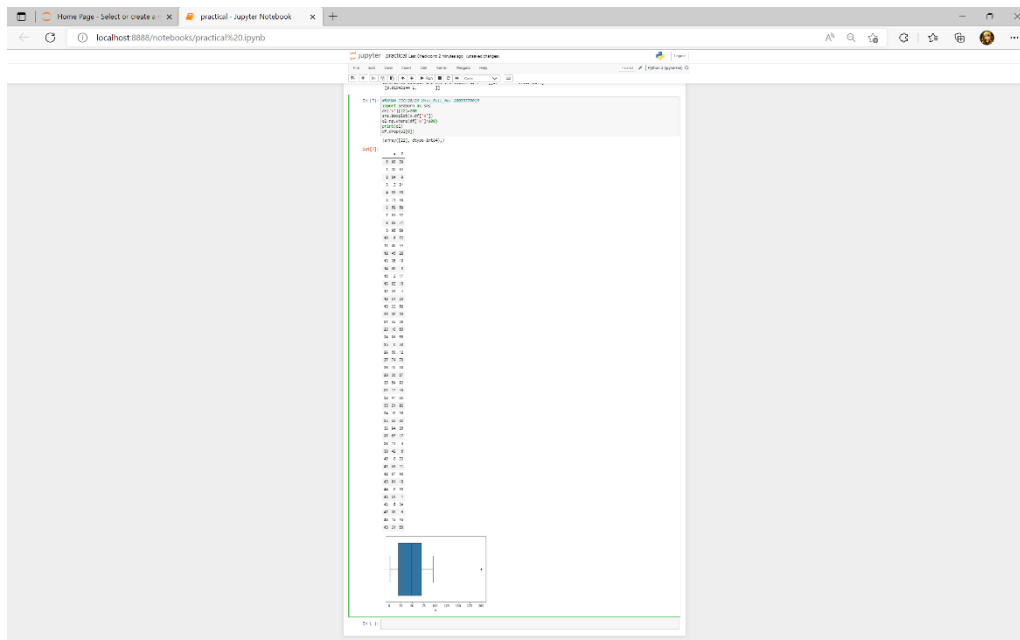
```
sns.boxplot(x=df['x'])
```

```
ol=np.where(df['x']>100)
```

```
print(ol)
```

```
df.drop(ol[0])
```

OUTPUT :-



Code h) :

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

edges=[0,20,40,60,80,100]

temp=pd.cut(df.iloc[:,2],edges)

print(temp)

OUTPUT :-

```
in [14]: answer = raman_csc.univ_roll_no : 20059570019
edges=[0,20,40,60,80,100]
temp=pd.cut(df.iloc[:,2],edges)
print(temp)
0      (0, 40]
1      (0, 100]
2      (0, 20]
3      (0, 100]
4      (0, 60]
5      (40, 60]
6      (0, 60]
7      (0, 40]
8      (0, 80]
9      (0, 80]
10     (0, 80]
11     (0, 80]
12     (20, 40]
13     (0, 20]
14     (0, 20]
15     (0, 20]
16     (0, 20]
17     (0, 20]
18     (0, 20]
19     (0, 60]
20     (20, 40]
21     (0, 80]
22     (0, 60]
23     (40, 60]
24     (0, 80]
25     (40, 60]
26     (0, 20]
27     (0, 80]
28     (0, 60]
29     (0, 80]
30     (0, 100]
31     (0, 100]
32     (0, 80]
33     (0, 100]
34     (20, 40]
35     (0, 100]
36     (0, 100]
37     (0, 20]
38     (0, 20]
39     (0, 20]
40     (20, 40]
41     (0, 20]
42     (0, 80]
43     (0, 20]
44     (0, 60]
45     (0, 20]
46     (20, 40]
47     (0, 20]
48     (0, 60]
49     (40, 60]
Name: 2, dtype: category
categories [0, interval[["left", "right"]]: [(0, 20] < (20, 40] < (40, 60] < (60, 80] < (80, 100]]
```

Ques 4.

Consider two excel files having attendance of a workshop's participants for two days. Each file has three fields 'Name', 'Time of joining', duration (in minutes) where names are unique within a file. Note that duration may take one of three values (30, 40, 50) only. Import the data into two dataframes and do the following:

- Perform merging of the two dataframes to find the names of students who had attended the workshop on both days.
- Find names of all students who have attended workshop on either of the days.
- Merge two data frames row-wise and find the total number of records in the data frame.
- Merge two data frames and use two columns names and duration as multi-row indexes. Generate descriptive statistics for this multi-index.

1ST File

	Name	Time of joining	Duration
1	ramani	9AM	30
2	Nishant	10AM	50
3	Khushi	9AM	30
4	Prateek	9AM	70
5	nemo	9AM	40
6	Deepanshu	9AM	30
7	Shruti	10AM	30
8	Khushi	9AM	40
9	vasu	9AM	50
10	Apni	9AM	40
11	annavi	9AM	30
12	Naveen	10AM	30
13	Kunal	9AM	40
14	Namika	9AM	50
15	Tanisha	9AM	40

2ND File

	Name	Time of Joining	Duration
1	Naveen	8AM	40
2	Nishant	10AM	50
3	Tanishia	8AM	30
4	Prateek	8AM	30
5	Neesha	8AM	40
6	Bharat	8AM	30
7	Shruti	10AM	30

Code :

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

```
import pandas as pd;
```

```
import numpy as np;
```

```
df1=pd.read_excel("D:/data/Ques4_Day1.xlsx",index_col=0);
```

```
df2=pd.read_excel("D:/data/Ques4_Day2.xlsx",index_col=0);
```

```
print("First Day Records");
```

```
print(df1);
```

```
print("Second Day Records");
```

```
print(df2);
```

OUTPUT :-

```
In [12]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
import pandas as pd;
import numpy as np;
df1=pd.read_excel("D:/data/Ques4_Day1.xlsx",index_col=0);
df2=pd.read_excel("D:/data/Ques4_Day2.xlsx",index_col=0);
print("First Day Records");
print(df1);
print("Second Day Records");
print(df2);
```

First Day Records

	Name	Time of Joining	Duration
1	raman	8AM	30
2	Nishant	10AM	50
3	khayti	8AM	40
4	Prateek	8AM	70
5	nemo	8AM	40
6	Deepanshu	8AM	30
7	shruti	10AM	30
8	Khushit	8AM	40
9	vasu	8AM	80
10	Ajay	8AM	40
11	anharvi	8AM	30
12	Naveen	10AM	30
13	Kunal	8AM	40
14	Neesha	8AM	50
15	Tanishia	8AM	40

Second Day Records

	Name	Time of Joining	Duration
1	Naveen	8AM	40
2	Nishant	10AM	50
3	Tanishia	8AM	30
4	Prateek	8AM	30
5	Neesha	8AM	40
6	Bharat	8AM	30
7	Shruti	10AM	30

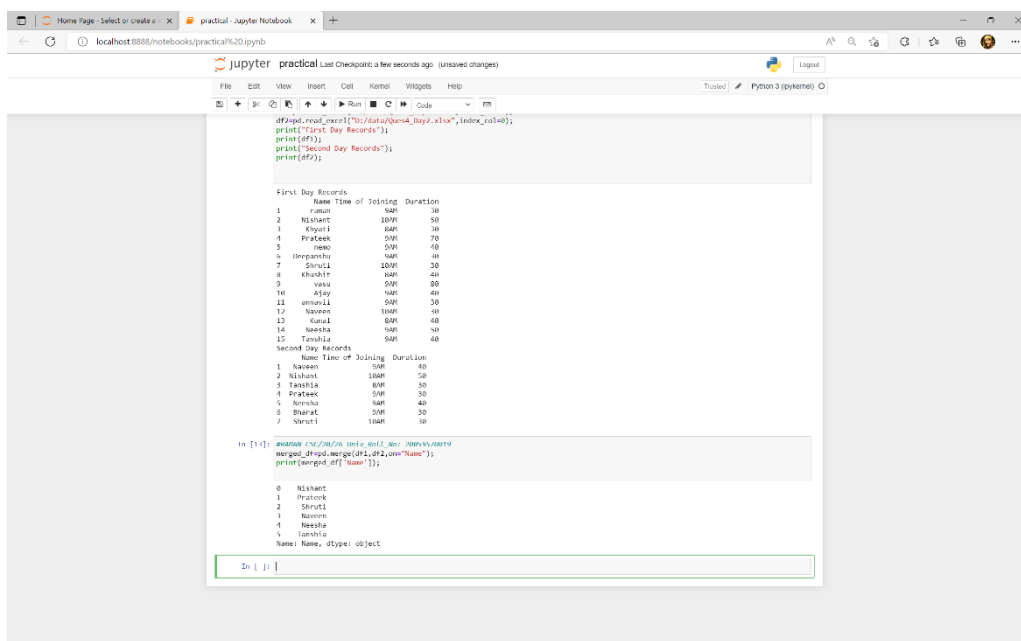
Code a) :

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

```
merged_df=pd.merge(df1,df2,on="Name");
```

```
print(merged_df['Name']);
```

OUTPUT :-



```
df1=pd.read_excel('D:/data/Gen4/day1.xlsx',index_col=0);
print("First Day Records");
print(df1);
df2=pd.read_excel('D:/data/Gen4/day2.xlsx',index_col=0);
print("Second Day Records");
print(df2);
```

First Day Records

	Name	Time of Joining	Duration
1	Ram	5AM	30
2	Nishant	10AM	50
3	Kyush	6AM	30
4	Prateek	5AM	70
5	neha	5PM	40
6	Dhyanesh	NAH	40
7	Shrut	10PM	30
8	Kushit	6AM	40
9	Vasu	5PM	80
10	Ajay	NAH	40
11	omavili	5AM	30
12	Naveen	10AM	40
13	Kunal	6PM	40
14	Nashia	5PM	NA
15	Tansha	5PM	40

Second Day Records

	Name	Time of Joining	Duration
1	Naveen	5AM	40
2	Nishant	10AM	50
3	Tansha	6PM	50
4	Prateek	5PM	50
5	Komcha	NAH	40
6	Bharat	5PM	30
7	Shrut	10AM	30

```
in [1]: ramana C:/Users/raman/OneDrive/Desktop/20059570019
merged_df=pd.merge(df1,df2,on="Name");
print(merged_df['Name']);
```

0 Nishant
1 Prateek
2 Shrut
3 Naveen
4 Neesha
5 Tansha
Name: Name, dtype: object

Code b) :

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

```
merged_df2=pd.merge(df1,df2,on="Name", how="outer");
```

```
print(merged_df2["Name"]);
```

OUTPUT :-

The screenshot shows a Jupyter Notebook interface with a single cell containing the following code and output:

```
In [13]: merges = CSG(30/26, title='Roll No: 20055790019')
merges_df = pd.merge(merges, df2, on='name')
print(merges_df[['name']])
```

Output:

```
0    Nishant
1    Prateek
2    Shrut
3    Raveen
4    Anusha
5    Tanisha
Name: name, dtype: object
```

```
In [14]: merges = CSG(30/26, title='Roll No: 20055790019')
merges_01 = pd.merge(merges, df1, on='name', how='outer')
print(merges_01[['name']])
```

Output:

```
0    ramon
1    Nishant
2    Shrut
3    Prateek
4    ramon
5    Deepanshu
6    Shrut
7    Shrut
8    ramon
9    Shrut
10    Anusha
11    Raveen
12    Ramal
13    Anusha
14    Tanisha
15    Shrut
Name: name, dtype: object
```

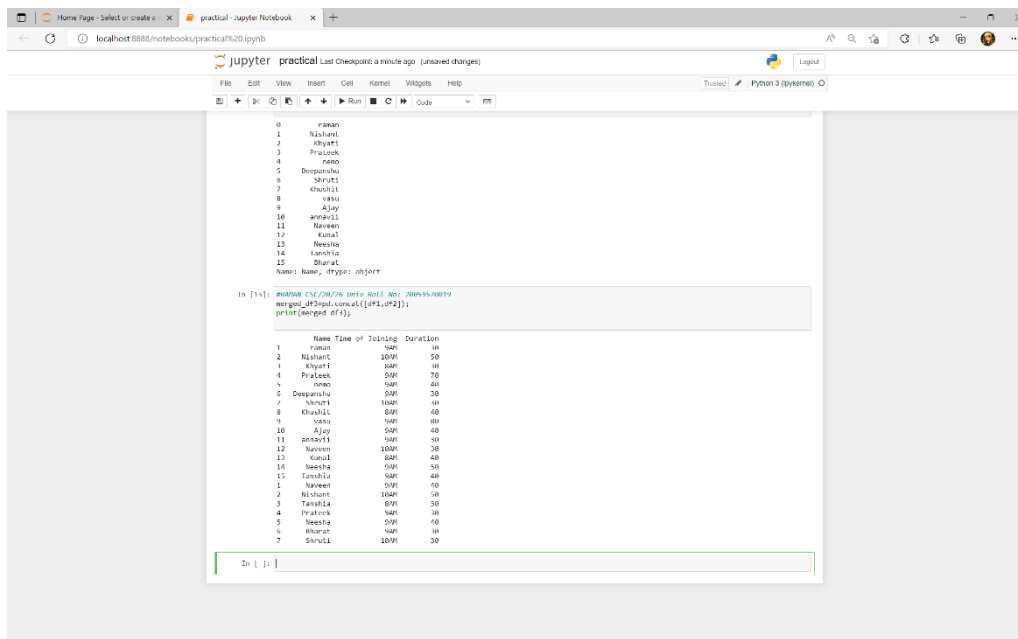
Code c) :

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

```
merged_df3=pd.concat([df1,df2]);
```

```
print(merged_df3);
```

OUTPUT :-



The screenshot shows a Jupyter Notebook interface with a code cell containing the following Python code:

```
0 raman  
1 Nishant  
2 Nishant  
3 Prateek  
4 rano  
5 Deepanshu  
6 Shrutti  
7 Khushil  
8 vasu  
9 Ajay  
10 anant  
11 Naveen  
12 Kumar  
13 Neesha  
14 Tanishia  
15 Shrut  
Name: Name, dtype: object
```

The output of the code is displayed below the cell:

```
In [1]: raman csc/20/26 Univ_Roll_No: 20059570019  
merged_df3=pd.concat([df1,df2]);  
print(merged_df3);
```

	Name	Time of Joining	Duration
1	raman	NaN	90
2	Nishant	10PM	50
3	Nishant	NaN	90
4	Prateek	5PM	70
5	Deepanshu	NaN	40
6	Shrutti	5PM	30
7	Khushil	6PM	40
8	vasu	6PM	40
9	Ajay	5PM	40
10	anant	5PM	40
11	Naveen	10PM	20
12	Kumar	6PM	40
13	Neesha	5PM	50
14	Tanishia	NaN	40
15	Shrut	5PM	40
16	Naveen	5PM	40
17	Nishant	NaN	50
18	Tanishia	6PM	30
19	Prateek	NaN	30
20	Neesha	5PM	40
21	Shrut	NaN	30
22	Shrut	10PM	30

Code d) :

```
#RAMAN CSC/20/26 Univ_Roll_No: 20059570019
merged_df4=pd.merge(df1,df2,how="outer");
merged_df4=merged_df4.set_index(['Name','Duration']);
merged_df4=merged_df4.sort_values(by=['Name']);
# For Changing "Time of Joining " to integer from String
merged_df4['Time of Joining']=merged_df4['Time of Joining'].astype('string');
for i in range(len(x)):
    merged_df4['Time of Joining'][i]=merged_df4['Time of Joining'][i].replace('AM','');
merged_df4['Time of Joining']=merged_df4['Time of Joining'].astype(int);
print ("Sum is \n",merged_df4.sum(0),"\n Mean is ",merged_df4.mean(0),"\n Standard Deviation is ",merged_df4.std(0),);
```

OUTPUT :-


```

merged_df3=pd.concat([df1,df2]);
print(merged_df3);

   Name Time of Joining Duration
1  raman              500      50
2  Nishant            1000      20
3  Khyati              800      30
4  Prateek              900      70
5    nico              700      60
6  Deepansha           900      30
7  Shivu              1000      30
8  Khushi              900      40
9    vasu              500      80
10  Arij              900      40
11  anavika            500      30
12  Naveen              1000      10
13  Guna              900      40
14  Neeha              500      50
15  Ransha              900      40
1  Naveen              500      40
2  Nishant            1000      10
3  Tanisha              800      30
4  Prateek              900      10
5  Neeha              500      40
6  Bharat              500      10
7  Shivu              1000      30

In [17]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
merged_df3=pd.merge(df1,df2,how="outer");
merged_df4=merged_df3.set_index(['Name','Duration']);
merged_df5=merged_df3.sort_values(['Name']);
# for changing 'Time of Joining' to Integer from String
merged_df6['Time of Joining']=merged_df6['Time of Joining'].astype('int');
for i in range(len(merged_df6)):
    merged_df6['Time of Joining'][i]=merged_df6['Time of Joining'][i].replace('AM','');
merged_df6['Time of Joining']=merged_df6['Time of Joining'].astype(int);
print ("Sum is :",merged_df6.sum(0),"N Mean is :",merged_df6.mean(0),"N Standard Deviation is :",merged_df6.std(0));

Sum is
Time of Joining    179
Duration          1050
Mean is :Sum of Joining    8.5%
dtype: float64
Standard Deviation is :Time of Joining    0.64488%
dtype: float64

```

Ques 5.

Taking 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

- Plot bar chart to show the frequency of each class label in the data.
- Draw a scatter plot for Petal width vs sepal width.
- Plot density distribution for feature petal length.
- Use a pair plot to show pairwise bivariate distribution in the Iris Dataset.

Code a):

```
#RAMAN CSC/20/26 Univ_Roll_No: 20059570019
```

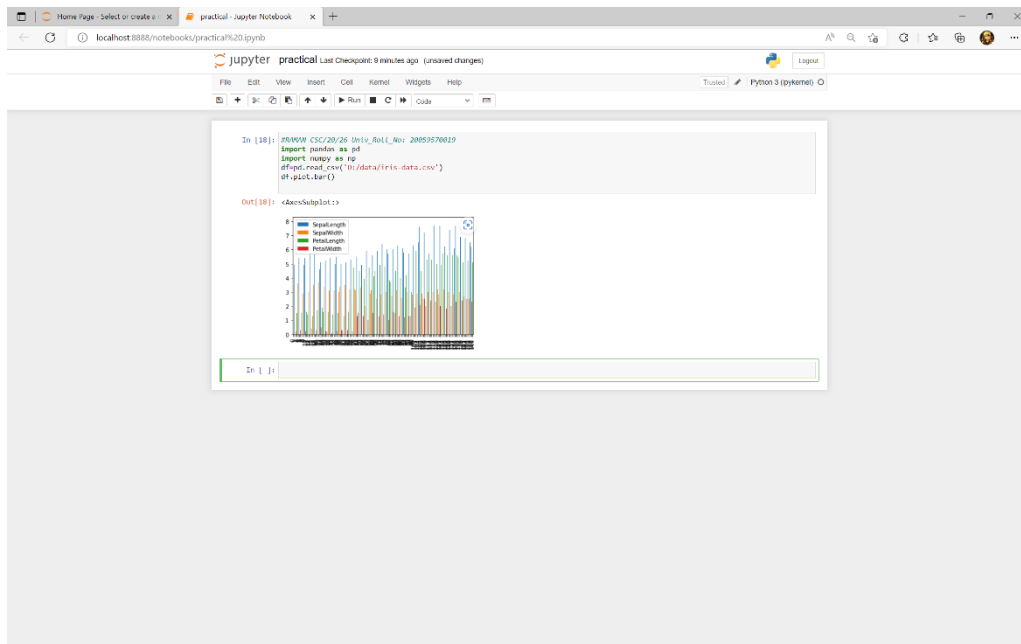
```
import pandas as pd
```

```
import numpy as np
```

```
df=pd.read_csv('D:/data/iris-data.csv')
```

```
df.plot.bar()
```

OUTPUT :-



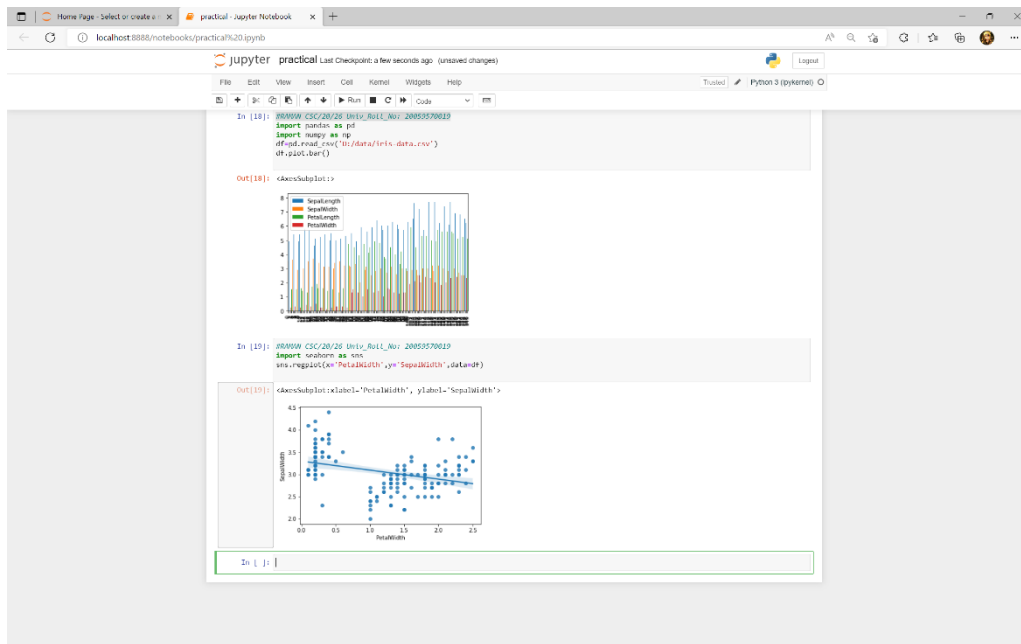
Code b):

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

import seaborn as sns

sns.regplot(x='PetalWidth',y='SepalWidth',data=df)

OUTPUT :-

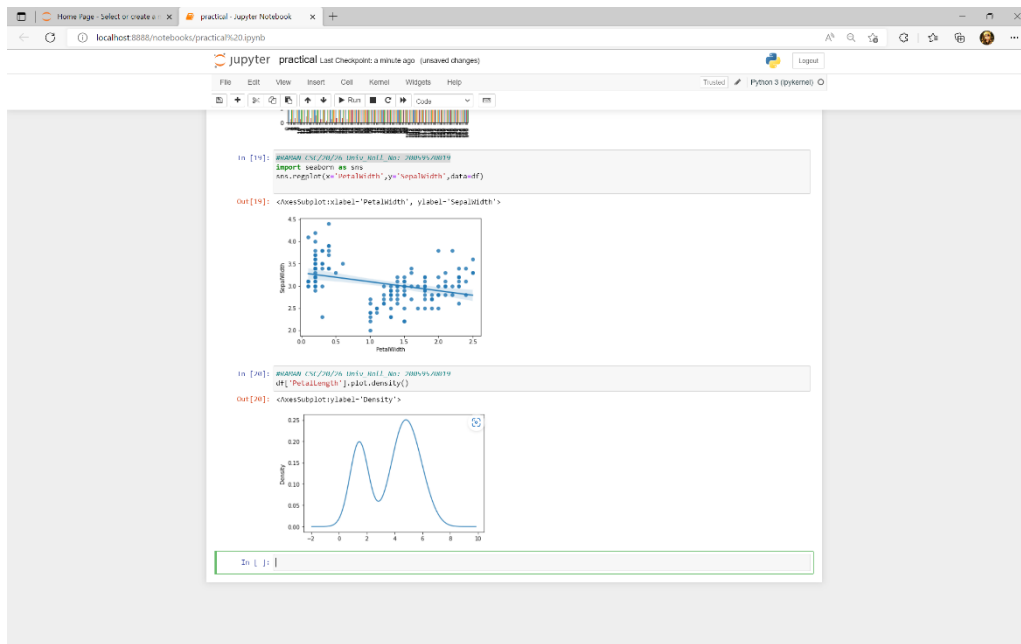


Code c):

```
#RAMAN CSC/20/26 Univ_Roll_No: 20059570019
```

```
df['Petal.Length'].plot.density()
```

OUTPUT :-

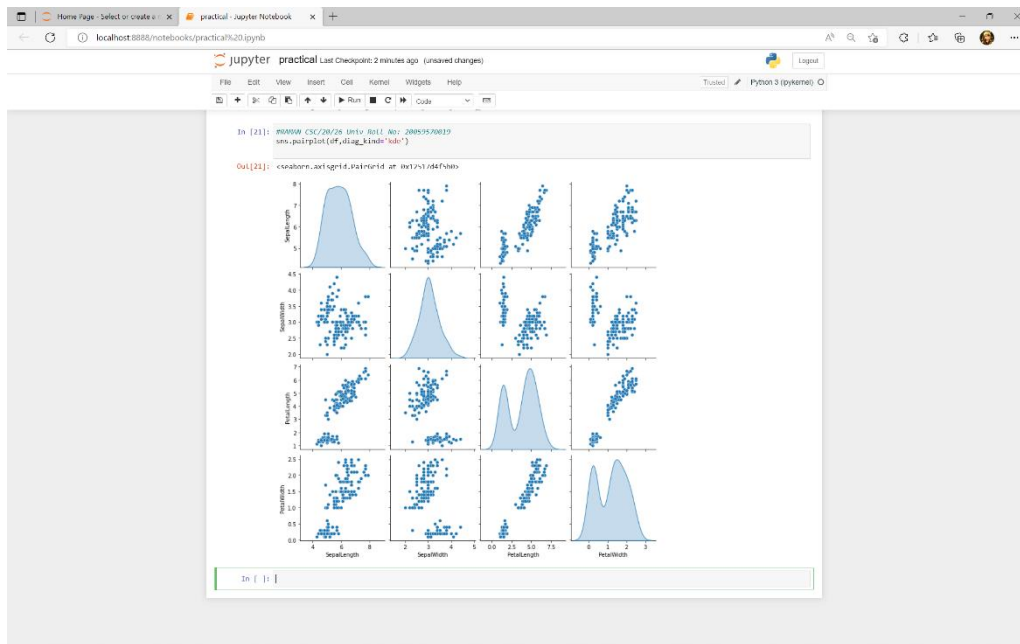


Code d):

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

sns.pairplot(df,diag_kind='kde')

OUTPUT :-



Ques 6.

Consider any sales training/ weather forecasting dataset

- Compute mean of a series grouped by another series
- Fill an intermittent time series to replace all missing dates with values of previous non-missing date.
- Perform appropriate year-month string to dates conversion.
- Split a dataset to group by two columns and then sort the aggregated results within the groups.
- Split a given dataframe into groups with bin counts

File taken :- weatherHistory.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	Formatted Summary	Precip	Temp	Apparent	Humidity	Wind Spd	Wind Dir	Wave Hgt	Wave Dir	Wave Per	Wave Len	Wave P	Wave S	Wave T	Wave U	Wave V	Wave W	Wave X	Wave Y	Wave Z						
2	2006-04-0 Partly Cloudy	9.472222	7.888889	0.86	14.1157	251	15.8263	0	1015.13	Partly cloudy throughout the day.																
3	2006-04-0 Partly Cloudy	9.355556	7.227778	0.86	14.2646	259	15.8263	0	1015.83	Partly cloudy throughout the day.																
4	2006-04-0 Mostly Clear	9.377778	9.377778	0.89	3.9284	204	14.9569	0	1015.94	Partly cloudy throughout the day.																
5	2006-04-0 Partly Cloudy	8.388889	5.944444	0.83	14.0236	209	15.8263	0	1016.41	Partly cloudy throughout the day.																
6	2006-04-0 Mostly Clear	8.755556	6.977778	0.83	11.0446	259	15.8263	0	1016.51	Partly cloudy throughout the day.																
7	2006-04-0 Partly Cloudy	9.222222	7.111111	0.85	13.9387	258	14.9569	0	1016.66	Partly cloudy throughout the day.																
8	2006-04-0 Partly Cloudy	7.733333	5.222222	0.95	12.3648	259	9.982	0	1016.72	Partly cloudy throughout the day.																
9	2006-04-0 Partly Cloudy	8.772222	6.527778	0.89	14.1510	260	9.982	0	1016.84	Partly cloudy throughout the day.																
10	2006-04-0 Partly Cloudy	10.822222	10.822222	0.82	11.3183	259	9.982	0	1017.37	Partly cloudy throughout the day.																
11	2006-04-0 Partly Cloudy	13.772222	13.772222	0.72	12.3258	279	9.982	0	1017.22	Partly cloudy throughout the day.																
12	2006-04-0 Partly Cloudy	16.01667	16.01667	0.67	17.3651	290	11.2056	0	1017.42	Partly cloudy throughout the day.																
13	2006-04-0 Partly Cloudy	17.144444	17.144444	0.54	19.7869	316	11.4471	0	1017.76	Partly cloudy throughout the day.																
14	2006-04-0 Partly Cloudy	17.8	17.8	0.55	21.9443	281	11.27	0	1017.59	Partly cloudy throughout the day.																
15	2006-04-0 Partly Cloudy	17.333333	17.333333	0.51	20.6885	289	11.27	0	1017.48	Partly cloudy throughout the day.																
16	2006-04-0 Partly Cloudy	18.87778	18.87778	0.47	15.3755	262	11.4471	0	1017.17	Partly cloudy throughout the day.																
17	2006-04-0 Partly Cloudy	18.91111	18.91111	0.46	10.4006	288	11.27	0	1016.47	Partly cloudy throughout the day.																
18	2006-04-0 Partly Cloudy	15.38889	15.38889	0.6	14.4095	251	11.27	0	1016.15	Partly cloudy throughout the day.																
19	2006-04-0 Mostly Clear	15.55	15.55	0.63	11.1573	230	11.4471	0	1016.17	Partly cloudy throughout the day.																
20	2006-04-0 Mostly Clear	14.25556	14.25556	0.69	8.5169	163	11.2056	0	1015.82	Partly cloudy throughout the day.																
21	2006-04-0 Mostly Clear	13.144444	13.144444	0.7	7.6314	139	11.2056	0	1015.83	Partly cloudy throughout the day.																
22	2006-04-0 Mostly Clear	11.55	11.55	0.77	7.3899	147	11.6285	0	1015.85	Partly cloudy throughout the day.																
23	2006-04-0 Mostly Clear	11.18333	11.18333	0.76	4.9266	160	9.982	0	1015.77	Partly cloudy throughout the day.																
24	2006-04-0 Partly Cloudy	10.11667	10.11667	0.79	6.6493	163	15.8263	0	1015.4	Partly cloudy throughout the day.																
25	2006-04-0 Mostly Clear	10.2	10.2	0.77	3.9284	152	14.9569	0	1015.51	Partly cloudy throughout the day.																
26	2006-04-1 Partly Cloudy	10.42222	10.42222	0.62	16.9835	150	15.8263	0	1014.4	Mostly cloudy throughout the day.																
27	2006-04-1 Partly Cloudy	9.911111	7.566667	0.66	17.2109	149	15.8263	0	1014.2	Mostly cloudy throughout the day.																
28	2006-04-1 Mostly Clear	11.18333	11.18333	0.8	10.8192	163	14.9569	0	1008.71	Mostly cloudy throughout the day.																
29	2006-04-1 Partly Cloudy	7.155556	5.044444	0.79	11.0768	180	15.8263	0	1014.47	Mostly cloudy throughout the day.																
30	2006-04-1 Partly Cloudy	8.111111	4.816667	0.82	6.4493	161	15.8263	0	1014.45	Mostly cloudy throughout the day.																
31	2006-04-1 Partly Cloudy	6.788889	4.272222	0.83	13.0088	155	14.9569	0	1014.49	Mostly cloudy throughout the day.																
32	2006-04-1 Mostly Clear	7.261111	5.155556	0.85	11.1734	141	6.1985	0	1014.52	Mostly cloudy throughout the day.																
33	2006-04-1 Mostly Clear	7.8	5.527778	0.83	12.8156	150	8.05	0	1014.16	Mostly cloudy throughout the day.																
34	2006-04-1 Mostly Clear	9.872222	7.933333	0.76	13.7404	160	9.982	0	1014.24	Mostly cloudy throughout the day.																
35	2006-04-1 Mostly Clear	12.222222	12.222222	0.72	15.6331	150	9.982	0	1014.25	Mostly cloudy throughout the day.																
36	2006-04-1 Mostly Clear	15.09444	15.09444	0.61	17.549	151	9.982	0	1013.96	Mostly cloudy throughout the day.																
37	2006-04-1 Mostly Clear	17.35556	17.35556	0.52	22.7815	169	9.982	0	1013.85	Mostly cloudy throughout the day.																
38	2006-04-1 Mostly Clear	15.00556	15.00556	0.46	28.8624	169	9.982	0	1013.04	Mostly cloudy throughout the day.																
39	2006-04-1 Mostly Clear	20.04444	20.04444	0.4	28.3682	170	9.982	0	1012.22	Mostly cloudy throughout the day.																
40	2006-04-1 Mostly Clear	21.05	21.05	0.4	26.9031	187	10.3523	0	1011.44	Mostly cloudy throughout the day.																
41	2006-04-1 Mostly Clear	21.18333	21.18333	0.47	25.8926	174	9.982	0	1010.57	Mostly cloudy throughout the day.																

Code a):

Note:- I have calculated Mean value of “Wind Bearing (degrees)” grouped by “Summary”

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

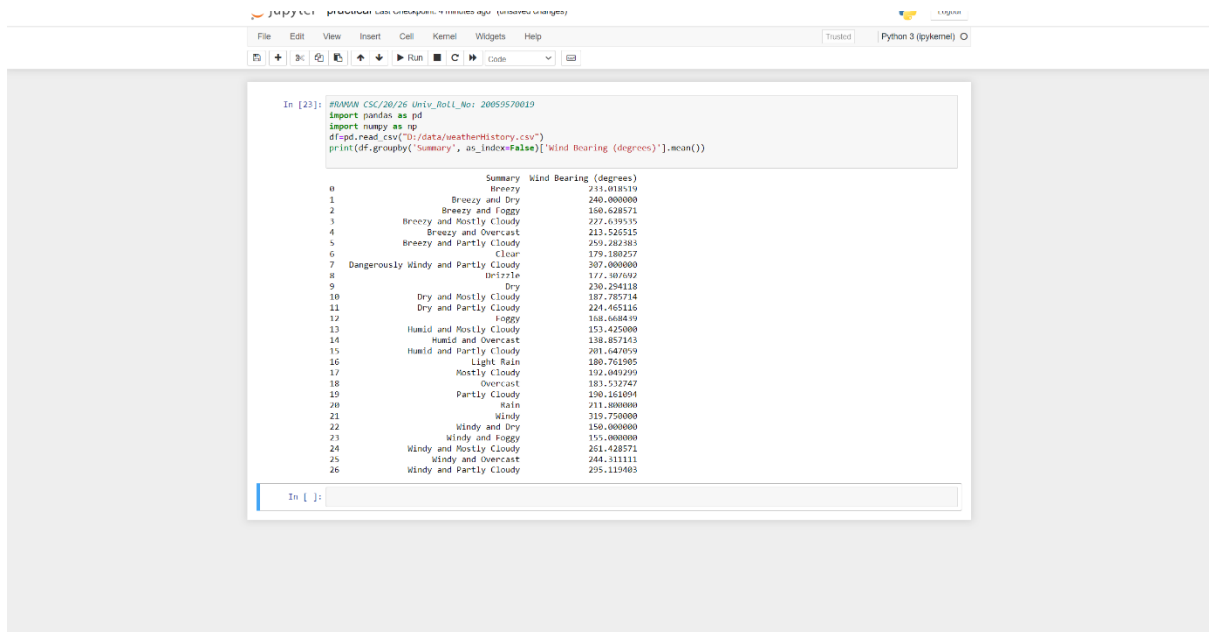
```
import pandas as pd
```

```
import numpy as np
```

```
df=pd.read_csv("D:/data/weatherHistory.csv")
```

```
print(df.groupby('Summary', as_index=False)['Wind Bearing (degrees)'].mean())
```

OUTPUT :-



```
In [23]: #RAWAN CSC/20/26 Univ_Roll_No: 20050570019
import pandas as pd
import numpy as np
df=pd.read_csv("D:/data/weatherHistory.csv")
print(df.groupby('Summary', as_index=False)['Wind Bearing (degrees)'].mean())
```

	Summary	Wind Bearing (degrees)
0	Breezy	211.018519
1	Breezy and Dry	240.000000
2	Breezy and Foggy	180.628571
3	Breezy and Mostly Cloudy	277.619535
4	Breezy and Overcast	213.525515
5	Breezy and Partly Cloudy	259.282383
6	Clear	179.180377
7	Dangerously Windy and Partly Cloudy	307.000000
8	Drizzle	177.107692
9	Dry	230.294118
10	Dry and Mostly Cloudy	187.785714
11	Dry and Partly Cloudy	221.455116
12	Foggy	168.604819
13	Humid and Mostly Cloudy	153.425000
14	Humid and Overcast	130.857343
15	Humid and Partly Cloudy	201.647099
16	Light Rain	180.761905
17	Mostly Cloudy	192.049299
18	Overcast	183.532747
19	Partly Cloudy	190.161004
20	Rain	211.808000
21	Windy	315.750000
22	Windy and Dry	150.000000
23	Windy and Foggy	155.000000
24	Windy and Mostly Cloudy	251.425771
25	Windy and Overcast	244.311111
26	Windy and Partly Cloudy	295.119403

Code b):

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

```
new_df2= df.ffill().reset_index()
```

```
print(new_df2)
```

OUTPUT :-

```
Home Page - Select or create a x practical - Jupyter Notebook x +
localhost8888/notebooks/practical%20jybn
jupyter practical Last checkpoint: 5 minutes ago (unsaved changes)
Python 3 (ipykernel)
File Edit View Insert Cell Kernel Help Trusted Python 3 (ipykernel)
In [24]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
new_df2= df.ffill().reset_index()
print(new_df2)

Index Formatted Date Summary Precip Type \
0 2006-04-01 00:00:00.000 +0000 Partly Cloudy rain
1 2006-04-01 01:00:00.000 +0000 Partly Cloudy rain
2 2006-04-01 02:00:00.000 +0000 Partly Cloudy rain
3 2006-04-01 03:00:00.000 +0000 Partly Cloudy rain
4 2006-04-01 04:00:00.000 +0000 Partly Cloudy rain
...
96448 2016-09-09 10:00:00.000 +0000 Partly Cloudy rain
96449 2016-09-09 11:00:00.000 +0000 Partly Cloudy rain
96450 2016-09-09 12:00:00.000 +0000 Partly Cloudy rain
96451 2016-09-09 13:00:00.000 +0000 Partly Cloudy rain
96452 2016-09-09 14:00:00.000 +0000 Partly Cloudy rain

Temperature (C) Apparent Temperature (C) Humidity Wind Speed (km/h) \
0 9.472722 7.288000 0.89 14.1157
1 9.755556 7.227778 0.86 14.2046
2 9.777778 6.977778 0.89 13.9284
3 8.388889 5.544444 0.83 14.1035
4 8.755556 6.977778 0.83 11.0446
...
96448 26.816667 26.816667 0.43 10.8903
96449 24.583333 24.583333 0.58 10.8947
96450 22.838889 22.838889 0.56 8.1618
96451 21.522222 21.522222 0.60 10.5291
96452 19.438889 19.438889 0.51 5.8766

Wind bearing (degrees) Visibility (km) Cloud Cover \
0 251.0 15.8205 0.0
1 279.0 15.8204 0.0
2 201.0 14.9550 0.0
3 269.0 15.8204 0.0
4 255.0 15.8203 0.0
...
96448 31.0 16.1000 0.0
96449 30.0 15.5520 0.0
96450 30.0 16.1000 0.0
96451 30.0 16.1000 0.0
96452 39.0 15.5204 0.0

Pressure (millibars) Daily Summary
0 1015.13 Partly cloudy throughout the day.
1 1015.63 Partly cloudy throughout the day.
2 1015.94 Partly cloudy throughout the day.
3 1015.51 Partly cloudy throughout the day.
4 1016.51 Partly cloudy throughout the day.
...
96448 1014.46 Partly cloudy starting in the morning.
96449 1021.10 Partly cloudy starting in the morning.
96450 1015.64 Partly cloudy starting in the morning.
96451 1022.05 Partly cloudy starting in the morning.
96452 1016.46 Partly cloudy starting in the morning.

[96453 rows x 13 columns]
```

Code c):

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019


```

from datetime import datetime

df=df.iloc[0:10,: ]

print("This is Before :- ",df['Formatted Date'][1],type(df['Formatted Date'][1]))

new_format=[]

# note:- here i have comment the For loop because Dataset is too big and takign a lot of time
# and taken a single value
# for i in range(len(df['Formatted Date'])):

temp=df['Formatted Date'][1][0:19]

year=temp[2:4]

month=temp[5:7]

day=temp[8:10]

time=temp[11:19]

temp=day+"/"+month+"/"+year+" "+time

temp=datetime.strptime(temp,'%d/%m/%y %H:%M:%S')

if temp not in new_format:

    new_format.append(temp)

# Commented for loop end here

print("New Format :- ",new_format[0],type(new_format[0]))

```

OUTPUT :-

```

...
...
...
96448      31.0      15.1000      0.0
96449      20.0      15.5520      0.0
96450      30.0      15.1000      0.0
96451      20.0      15.1000      0.0
96452      20.0      15.5500      0.0
...
...
...
Pressure (millibars)      Daily Summary
0      1015.13      Partly cloudy throughout the day.
1      1015.63      Partly cloudy throughout the day.
2      1015.94      Partly cloudy throughout the day.
3      1016.41      Partly cloudy throughout the day.
4      1016.53      Partly cloudy throughout the day.
...
...
...
96448      1014.36      Partly cloudy starting in the morning.
96449      1015.18      Partly cloudy starting in the morning.
96450      1015.66      Partly cloudy starting in the morning.
96451      1015.95      Partly cloudy starting in the morning.
96452      1016.16      Partly cloudy starting in the morning.
...
[96453 rows x 12 columns]

In [25]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
from datetime import datetime
df=df.iloc[0:10,: ]
print("This is Before :- ",df['Formatted Date'][1],type(df['Formatted Date'][1]))
new_format=[]
# note:- here i have comment the For loop because Dataset is too big and takign a lot of time
# and taken a single value
# for i in range(len(df['Formatted Date'])):

temp=df['Formatted Date'][1][0:19]

year=temp[2:4]

month=temp[5:7]

day=temp[8:10]

time=temp[11:19]

temp=day+"/"+month+"/"+year+" "+time

temp=datetime.strptime(temp,'%d/%m/%y %H:%M:%S')

if temp not in new_format:

    new_format.append(temp)

# Commented for loop end here

print("New Format :- ",new_format[0],type(new_format[0]))

This is Before :-  2006-04-01 01:00:00.0000 <class 'str'>
New Format :-  2006-04-01 01:00:00 <class 'datetime.datetime'>

In [ ]:

```

Code d):

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

```
new_df=pd.DataFrame(df.groupby(['Precip Type','Loud Cover']).first(10))

print(new_df.sort_values(['Temperature (C)']))
```

OUTPUT :-

```

In [1]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
from datetime import datetime
df=pd.read_csv('data.csv')
print("This is before : ",df['formatted Date'][1],type(df['formatted Date'][1]))
new_format=[]
# note: here i have comment the for loop because Dataset is too big and taking a lot of time
# and taken a single value
# for i in range(len(df['formatted Date'])):
#     temp=df['formatted Date'][i][0:19]
#     year=temp[2:4]
#     month=temp[5:7]
#     day=temp[8:10]
#     time=temp[11:13]
#     tempday="."+month+"/"+year+" "+time
#     tempdate=datetime.strptime(temp,".%m/%y %H:%M:%S")
#     if temp not in new_format:
#         new_format.append(temp)
#     # commented for loop end here
print("New format :- ",new_format[0],type(new_format[0]))

In [3]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
new_df=pd.DataFrame(df.groupby(['Precip Type','Loud Cover']).first(10))
print(new_df.sort_values(['Temperature (C)']))

Precip Type Loud Cover    temperature (C)    Apparent temperature (C)    Humidity \
snow      0.0             -0.481555          -6.134000          1.00
rain      0.0             5.472222           7.388889          0.89

Precip Type Loud Cover    Wind Speed (km/h)    Wind Bearing (degrees) \
snow      0.0             11.0923            210.0
rain      0.0             14.1187            253.0

Precip Type Loud Cover    Visibility (km)    Pressure (millibars)
snow      0.0             9.4838            1011.56
rain      0.0             15.8265            1010.13

```

Code e):

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

```
edges=[0.0,0.3,0.6,1.0]

result=pd.cut(df['Humidity'],edges)

df['bin']=result

temp_df=pd.DataFrame(df.groupby('bin'))

print(temp_df)
```

OUTPUT :-

```
# for x in range(len(df['Formatted Date'])):
#     temp_df['Formatted Date'] = df['Formatted Date']
#     year=temp_df[0]
#     month=temp_df[1]
#     day=temp_df[2]
#     time=temp_df[3]
#     temp_df['month'] = year + "-" + month + "-" + day
#     temp_df['time'] = month + "-" + day + "-" + time
#     temp_df['temp'] = temp_df['temp'].astype(float)
#     if temp not in new_format:
#         new_format.append(temp)
#     # commented for loop end here
# print("New format :- ",new_format,type(new_format))
```

```
In [1]: m0000 CSC/20/20 Univ Roll No: 20003570019
new_df=pd.DataFrame(df.groupby(['Precip Type','Loud Cover']).first(10))
print(new_df.sort_values(['Temperature (C)']))
```

	Temperature (C)	Apparent temperature (C)	Humidity \
Precip Type Loud Cover			
snow	0.0	-0.401444	-0.140000 1.00
rain	0.0	9.472222	7.380000 0.89

	Wind Speed (km/h)	Wind Bearing (degrees) \
Precip Type Loud Cover		
snow	0.0	11.0929 219.0
rain	0.0	14.1167 251.0

	Visibility (km)	Pressure (millibars)
Precip Type Loud Cover		
snow	0.0	0.4838 1011.56
rain	0.0	15.0263 1015.13


```
In [4]: m0000 CSC/20/20 Univ Roll No: 20003570019
edges=[0.0,0.3,0.6,1.0]
result=pd.cut(df['Humidity'],edges)
df['bin']=result
temp_df=pd.DataFrame(df.groupby('bin'))
print(temp_df)
```

0 (0.0, 0.3]	Formatted Date	Se...
1 (0.3, 0.6]	Formatted Date	Se...
2 (0.6, 1.0]	Formatted Date	Se...

Ques 7.

Consider a data frame containing data about students i.e. name, gender and passing division:

	Name	Birth_Month	Gender	Pass_Division
0	Mudit Chauhan	December	M	III
1	Seema Chopra	January	F	II
2	Rani Gupta	March	F	I
3	Aditya Narayan	October	M	I
4	Sanjeev Sahni	February	M	II
5	Prakash Kumar	December	M	III
6	Ritu Agarwal	September	F	I
7	Akshay Goel	August	M	I
8	Meeta Kulkarni	July	F	II
9	Preeti Ahuja	November	F	II
10	Sunil Das Gupta	April	M	III
11	Sonali Sapre	January	F	I
12	Rashmi Talwar	June	F	III
13	Ashish Dubey	May	M	II
14	Kiran Sharma	February	F	II
15	Sameer Bansal	October	M	I

- Perform one hot encoding of the last two columns of categorical data using the `get_dummies()` function.
- Sort this data frame on the "Birth Month" column (i.e. January to December). Hint: Convert Month to Categorical.

Note: I have used a CSV File for storing data

Code a):

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

```
import pandas as pd
```

```
import numpy as np

df=pd.read_csv("D:/data/Ques7.csv")

# print(df);

first=pd.get_dummies(df['Gender'])

pd.get_dummies(df['Pass_Division']).join(first)
```

OUTPUT :-

The screenshot shows a Jupyter Notebook interface with a code cell and its output. The code cell contains the following Python code:

```
In [5]: #SAGAN CSC/20/26 Univ_Roll_No: 20059570019
import pandas as pd
import numpy as np
df=pd.read_csv("D:/data/Ques7.csv")
# print(df);
first=pd.get_dummies(df['Gender'])
pd.get_dummies(df['Pass_Division']).join(first)
```

The output cell shows the result of the code execution, which is a 15x6 matrix of dummy variables. The columns are labeled I, III, F, M, and the rows are indexed from 0 to 15.

```
Out[5]:
```

	I	III	F	M
0	0	0	1	0
1	0	1	0	1
2	1	0	0	1
3	1	0	0	1
4	0	1	0	1
5	0	0	1	0
6	1	0	0	1
7	1	0	0	1
8	0	1	0	1
9	0	1	0	1
10	0	0	1	0
11	1	0	0	1
12	0	0	1	1
13	0	1	0	1
14	0	1	0	1
15	1	0	0	1

Code b):

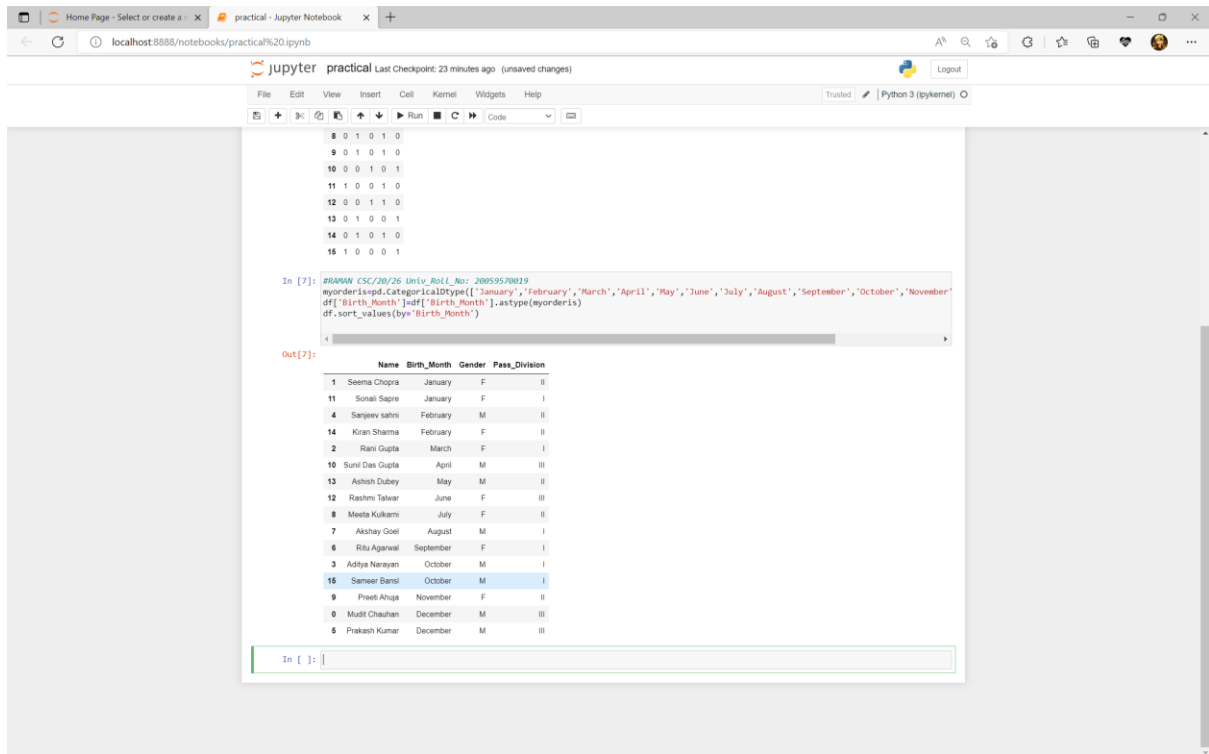
```
#RAMAN CSC/20/26 Univ_Roll_No: 20059570019
```

```
myorderis=pd.CategoricalDtype(['January','February','March','April','May','June','July','August','September','October','November',  
'December'],ordered=True)
```

```
df['Birth_Month']=df['Birth_Month'].astype(myorderis)
```

```
df.sort_values(by='Birth_Month')
```

OUTPUT :-



```
#RAMAN CSC/20/26 Univ_Roll_No: 20059570019
myorderis=pd.CategoricalDtype(['January','February','March','April','May','June','July','August','September','October','November',
'December'],ordered=True)
df['Birth_Month']=df['Birth_Month'].astype(myorderis)
df.sort_values(by='Birth_Month')
```

	Name	Birth_Month	Gender	Pass_Division
1	Seema Chopra	January	F	II
11	Sonali Sagre	January	F	I
4	Sanjeev sahni	February	M	II
14	Kiran Sharma	February	F	II
2	Rani Gupta	March	F	I
10	Sunil Das Gupta	April	M	III
13	Ashish Dubey	May	M	II
12	Rashmi Talwar	June	F	III
8	Meeta Kulkarni	July	F	II
7	Akshay Goel	August	M	I
6	Ritu Agarwal	September	F	I
3	Aditya Narayan	October	M	I
15	Sameer Bansal	October	M	I
9	Preeti Ahuja	November	F	II
0	Mudit Chauhan	December	M	III
5	Prakash Kumar	December	M	III

Ques 8.

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

Name	Gender	MonthlyIncome (Rs.)
Shah	Male	114000.00
Vats	Male	65000.00
Vats	Female	43150.00
Kumar	Female	69500.00
Vats	Female	155000.00
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:

- Calculate and display familywise gross monthly income.
- Calculate and display the member with the highest monthly income in a family.
- Calculate and display monthly income of all members with income greater than Rs. 60000.00.
- Calculate and display the average monthly income of the female members in the Shah family.

Note:- I have created a CSV File using the Provided Dataset and load the Data into a Dataframe from the Created CSV File.

Name	Gender	MonthlyIncome (Rs.)
Shah	Male	114000
Vats	Male	65000
Vats	Female	43150
Kumar	Female	69500
Vats	Female	155000
Kumar	Male	103000
Shah	Male	55000
Shah	Female	112400
Kumar	Female	81030
Vats	Male	71900

Code a):

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

import pandas as pd

import numpy as np

df=pd.read_csv("D:/data/Ques8.csv")

family=[]

Get Unique Family name's

for x in df.Name:

if x not in family:

family.append(x)

income=np.zeros(len(family))

for i in range(len(family)):

for j in range(len(df.Name)):

if family[i]==df.Name[j]:

income[i]=income[i]+df['MonthlyIncome (Rs.)'][j]


```
df2=pd.DataFrame(list(zip(family, income)),columns =['Family', 'Gross-Income'])

print(df2)
```

OUTPUT :-

The screenshot shows a Jupyter Notebook window titled 'practical' with a 'Python 3 (ipykernel)' kernel. The code in the cell is as follows:

```
In [9]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
import pandas as pd
import numpy as np
df=pd.read_csv("D:/data/ques8.csv")
family=[]

# Get Unique Family name's
for x in df.Name:
    if x not in family:
        family.append(x)
income=np.zeros(len(family))

for i in range(len(family)):
    for j in range(len(df.Name)):
        if family[i]==df.Name[j]:
            income[i]=income[i]+df['MonthlyIncome (Rs.)'][j]

df2=pd.DataFrame(list(zip(family, income)),columns =['Family', 'Gross-Income'])
print(df2)
```

The output of the code is a DataFrame with two columns: 'Family' and 'Gross-Income'.

	Family	Gross-Income
0	Shah	342000.0
1	Vats	260000.0
2	Kumar	129450.0

Code b):

```
#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

max_income=np.zeros(len(df2.Family))

for i in range(len(df2.Family)):
    for j in range(len(df.Name)):
        if df2.Family[i]==df.Name[j]:
            if max_income[i]<df['MonthlyIncome (Rs.)'][j]:
                max_income[i]=df['MonthlyIncome (Rs.)'][j]

print("Member With Highest Monthly Income in family")

for i in range(len(max_income)):
    print(df.iloc[df[df['MonthlyIncome (Rs.)']==max_income[i]].index.values,:])
```

OUTPUT :-

```

In [9]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
import pandas as pd
import numpy as np
df=pd.read_csv("D:/data/Ques8.csv")
family=[]

# Get Unique Family name's
for x in df.Name:
    if x not in family:
        family.append(x)
income=np.zeros(len(family))

for i in range(len(family)):
    for j in range(len(df.Name)):
        if family[i]==df.Name[j]:
            income[i]=income[i]+df['MonthlyIncome (Rs.)'][j]

df2=pd.DataFrame(list(zip(family, income)),columns=['Family', 'Gross-Income'])
print(df2)

Family Gross-Income
0 Shah 342000.0
1 Vats 260000.0
2 Kumar 129450.0

In [10]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
max_income=np.zeros(len(df2.Family))
for i in range(len(df2.Family)):
    for j in range(len(df2.Name)):
        if df2.Family[i]==df2.Name[j]:
            if max_income[i]<df2['MonthlyIncome (Rs.)'][j]:
                max_income[i]=df2['MonthlyIncome (Rs.)'][j]

print("Member With Highest Monthly Income in family")
for i in range(len(max_income)):
    print(df2.iloc[df2['MonthlyIncome (Rs.)']==max_income[i].index.values,:])

Member With Highest Monthly Income in family
Name Gender MonthlyIncome (Rs.)
0 Shah Male 114000
Name Gender MonthlyIncome (Rs.)
4 Vats Female 155000
Name Gender MonthlyIncome (Rs.)
5 Kumar Male 103000

```

Code c):

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

print("Showing monthly income of all members with income greater than Rs. 60000.00.")

for i in range(len(df.index)):

 if df['MonthlyIncome (Rs.)'][i]>60000:

 print(df['MonthlyIncome (Rs.)'][i])

OUTPUT :-

```

df2=pd.DataFrame(list(zip(Family, income)),columns=['Family', 'Gross-Income'])
print(df2)

Family Gross-Income
0 Shah 342000.0
1 Vats 260000.0
2 Kumar 129450.0

In [10]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
max_income=np.zeros(len(df2.Family))
for i in range(len(df2.Family)):
    for j in range(len(df.Name)):
        if df2.Family[i]==df.Name[j]:
            if max_income[i]<df['MonthlyIncome (Rs.)'][j]:
                max_income[i]=df['MonthlyIncome (Rs.)'][j]

print("Member With Highest Monthly Income in family")
for i in range(len(max_income)):
    print(df.iloc[df[df['MonthlyIncome (Rs.)']==max_income[i]].index.values,:])

Member With Highest Monthly Income in family
Name Gender MonthlyIncome (Rs.)
0 Shah Male 114000
Name Gender MonthlyIncome (Rs.)
4 Vats Female 155000
Name Gender MonthlyIncome (Rs.)
5 Kumar Male 103000

In [11]: #RAMAN CSC/20/26 Univ_Roll_No: 20059570019
print("Showing monthly income of all members with income greater than Rs. 60000.00.")
for i in range(len(df.index)):
    if df['MonthlyIncome (Rs.)'][i]>60000:
        print(df['MonthlyIncome (Rs.)'][i])

Showing monthly income of all members with income greater than Rs. 60000.00.
114000
65000
69500
155000
103000
112400
81030
71900

```

Code d):

#RAMAN CSC/20/26 Univ_Roll_No: 20059570019

import statistics as stats

female_Shah=[]

for i in range(len(df.index)):

if (df.Name[i]=='Shah') and (df.Gender[i]=='Female'):

```
female_Shah.append(df['MonthlyIncome (Rs.)'][i])

print("The average monthly income of the female members in the Shah family :- ",stats.mean(female_Shah))
```

OUTPUT :-

```

In [10]: #RAVAN CSC/20/26 Univ.Roll.No: 20059570019
max_income=np.zeros(len(df2.Family))
for i in range(len(df2.Family)):
    for j in range(len(df.Name)):
        if df2.Family[i]==df.Name[j]:
            if max_income[i]<df['MonthlyIncome (Rs.)'][j]:
                max_income[i]=df['MonthlyIncome (Rs.)'][j]
print("Member With Highest Monthly Income in family")
for i in range(len(max_income)):
    print(df.iloc[df['MonthlyIncome (Rs.)']==max_income[i]].index.values,:))

Member With Highest Monthly Income in family
Name Gender MonthlyIncome (Rs.)
0 Shah Male 114000
Name Gender MonthlyIncome (Rs.)
4 Vats Female 155000
Name Gender MonthlyIncome (Rs.)
5 Kumar Male 103000

In [11]: #RAVAN CSC/20/26 Univ.Roll.No: 20059570019
print("Showing monthly income of all members with income greater than Rs. 60000.00.")
for i in range(len(df.index)):
    if df['MonthlyIncome (Rs.)'][i]>60000:
        print(df['MonthlyIncome (Rs.)'][i])

Showing monthly income of all members with income greater than Rs. 60000.00.
114000
65000
69500
155000
103000
112400
81030
71900

In [12]: #RAVAN CSC/20/26 Univ.Roll.No: 20059570019
import statistics as stats
female_Shah=[]
for i in range(len(df.index)):
    if (df.Name[i]=='Shah') and (df.Gender[i]=='Female'):
        female_Shah.append(df['MonthlyIncome (Rs.)'][i])
print("The average monthly income of the female members in the Shah family :- ",stats.mean(female_Shah))

The average monthly income of the female members in the Shah family :- 112400

In [ ]:

```