Name - Prateek

Roll No - 20HCS4144

Course - Bsc. Hons. Computer Science

Year - 3

Semester - 5

Data Analysis and Visualization Assignment

Q #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]

Ans #1.

```
dic1= {'boys':[72,68,70,69,74],'girls':[63,65,69,62,61]}
print("Original dictionary of lists:",dic1)
list = []
for j in range(0,5):
    dic = {}
    dic['boys'] = dic1['boys'][j]
    dic['girls'] = dic1['girls'][j]
    list.append(dic)
print("From the given dictionary of lists the following is List of Dictionaries:",list)
```

Output:

```
Original dictionary of lists: {'boys': [72, 68, 70, 69, 74], 'girls': [63, 65, 69, 62, 61]}
From the given dictionary of lists the following is List of Dictionaries: [{'boys': 72, 'girls': 63}, {'boys': 68, 'girls': 6
5}, {'boys': 70, 'girls': 69}, {'boys': 69, 'girls': 62}, {'boys': 74, 'girls': 61}]
```

Q #2.

Given are three lists where L1 has names of n students, L2 has marks and L3 has hobbies of n students. Using three lists, create the following dictionary with hobbies as keys and (names, marks) as values. In case more than one student has same hobby then values must be appended for the same key instead of overwriting. E.g.
L1=[‘A’,’B’,’C’] L2=[100,40,50] L3=[‘painting’,’music’,’painting’]
output should be as follows:
dict1={painting:((‘A’,100),(’C’,50)), music(’B’,40)}

Q #3.

Write two lambda functions

a. One to arrange a list of names on the last letter of the name i.e. names=[axc, bxbb, xxb, zzxy, zzc] New sorted list on last letter is [bxbb,xxb,axc,zzc,zzxy]

b. Second lambda function arranges list on the length of names and store results in dictionary with length as key and value as names.

Ans #1.

a.

```
In [3]: elements= ['axc', 'bxbb','xxb','zzxy','zzc']
        new_lst=sorted(elements, key=lambda x: x[-1])
        print(new_lst)

        ['bxbb', 'xxb', 'axc', 'zzc', 'zzxy']
```

b.

```
Anydict = {}
for i in new_lst:
    j = len(i)
    Anydict.setdefault(j,[]).append(i)
print(Anydict)
```

```
{4: ['bxbb', 'zzxy'], 3: ['xxb', 'axc', 'zzc']}
```

Q #4.

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. 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 n x m array, n, m and array elements 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.

```
#4
#a
import numpy as np
arr = np.random.randint(1,25,(3,4))
print("Array : ")
print(arr)
# along the second axis
# Mean
print('Mean of the array: ',arr.mean(axis=1))
# Standard deviation
print('Standard Deviation of the array: ',arr.std(axis=1))
# Variance
print('Variance of the array: ',arr.var(axis=1))
```

```
Array :
[[24 23 18 19]
 [ 9  9  9 12]
 [17  6  2  5]]
Mean of the array:  [21.    9.75  7.5 ]
Standard Deviation of the array:  [2.54950976 1.29903811 5.67890835]
Variance of the array:  [ 6.5     1.6875 32.25  ]
```

```
#b
B = [56, 48, 22, 41, 78, 91, 24, 46, 8, 33]
arr1 = np.array(B)
#arr1
print("Sorted array: ",np.sort(arr1))
print("Indices of the sorted elements of a given array: ",np.argsort(arr1))
```

```
Sorted array:  [ 8 22 24 33 41 46 48 56 78 91]
Indices of the sorted elements of a given array:  [8 2 6 9 3 7 1 0 4 5]
```

```
#c
m = int(input('Enter the number of rows(m): '))
n = int(input('Enter the number of columns(m): '))
ar = np.random.randint(1,100,(m,n))
print("Shape of array: ",ar.shape)
print("Type of array: ",type(ar))
print("Data type: ",ar.dtype)
ar1 = ar.reshape(n,m)
print("After reshaping: \n", ar1)
print("New shape after reshaping: ",ar1.shape)
```

```
Enter the number of rows(m): 4
Enter the number of columns(m): 5
Shape of array:  (4, 5)
Type of array:  <class 'numpy.ndarray'>
Data type:  int32
After reshaping:
 [[27 86 22 97]
 [44 34  2 86]
 [69 97 75 52]
 [99 14 76 68]
 [20 84 15  3]]
New shape after reshaping:  (5, 4)
```

```
#d
ar2 = np.array([1,np.nan,3,4,np.nan,0,0,0,6,5,4,0])
print("Original array: ",ar2)
print("Test whether none of the elements of an array is zero: ",np.all(ar2))
print("The index of the zero elements is: ",np.where(ar2==0)[0])
print("Check if it contains nan: ",np.isnan(ar2))
print("The index of nan elements: ",np.where(np.isnan(ar2)==True)[0])
```

```
Original array:  [ 1. nan  3.  4. nan  0.  0.  0.  6.  5.  4.  0.]
Test whether none of the elements of an array is zero:  False
The index of the zero elements is:  [ 5  6  7 11]
Check if it contains nan:  [False  True False False  True False False False False False False False]
The index of nan elements:  [1 4]
```

Q #5.

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:

i) Identify and count missing values in a dataframe.

ii) Drop the column having more than 5 null values.

iii) Identify the row label having maximum of the sum of all values in a row and drop that row.

iv) Sort the dataframe on the basis of the first column.

v) Remove all duplicates from the first column.

vi) Find the correlation between first and second column and covariance between second and third column.

vii) Detect the outliers and remove the rows having outliers.

viii) Discretize second column and create 5 bins

Ans #5.

```
#5
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randint(1,20,(50,4)))
print("Dataframe having 50 rows and atleast 3 columns is: ",df)
#a
rows = len(df)
columns = len(df.columns)
n = 0.10*rows*columns
while n != 0:
    i = np.random.randint(rows)
    j = np.random.randint(columns)
    if df.iloc[i,j] != np.nan:
        df.iloc[i,j] = np.nan
    n-=1
print("Dataframe having 10 percent null values is: ",df)
```

```
Dataframe having 50 rows and atleast 3 columns is:       0   1   2   3
0    10  17  19  17
1    12   8   4   9
2    18   3   8  18
3    15   8   5  17
4    15  17  12  13
5    18  17   4  12
```

…….

```
Dataframe having 10 percent null values is:         0     1     2     3
0     NaN  17.0  19.0  17.0
1     NaN   8.0   4.0   9.0
2    18.0   3.0   8.0  18.0
3    15.0   8.0   5.0  17.0
4    15.0  17.0  12.0  13.0
5    18.0  17.0   4.0  12.0
6     3.0  18.0   1.0   9.0
7     9.0  13.0  13.0  15.0
8    13.0   2.0  10.0   8.0
9     9.0  17.0  12.0   NaN
10   11.0   2.0   NaN   1.0
11    7.0  17.0   9.0   5.0
12   14.0  18.0   NaN   9.0
13   17.0   NaN   5.0  18.0
14    8.0  19.0   2.0   9.0
15    3.0  12.0   7.0  13.0
16   17.0  10.0   7.0  12.0
17   14.0  11.0  15.0  19.0
18   18.0   9.0  18.0  16.0
19   12.0   NaN  13.0   9.0
20    NaN  10.0  13.0   4.0
21    1.0   4.0   9.0  10.0
22   10.0  19.0  16.0  13.0
23    NaN   NaN   7.0   NaN
24    9.0   2.0  13.0  19.0
25   15.0   8.0  15.0  16.0
26    NaN   NaN  12.0  13.0
27   17.0  15.0   NaN   1.0
28   19.0   1.0  13.0  17.0
29   10.0  18.0   1.0   8.0
30    2.0  11.0   1.0   3.0
31    2.0  15.0  10.0   9.0
32    9.0  18.0  10.0   5.0
33    NaN  13.0  10.0  19.0
34   16.0  12.0  12.0   4.0
35    7.0   9.0   4.0  18.0
36   16.0   9.0   3.0   7.0
37    8.0  12.0   8.0  11.0
38    9.0   4.0  13.0   9.0
39    1.0  10.0  15.0   6.0
40   16.0  12.0  16.0  11.0
41    4.0   6.0   9.0   9.0
42   18.0  15.0   4.0   2.0
43   12.0  16.0   NaN   6.0
44   13.0   1.0  12.0   NaN
45    6.0   5.0   1.0   NaN
46   10.0   1.0   5.0   5.0
47    9.0  14.0  18.0  16.0
48   15.0   5.0   1.0  17.0
49    7.0  16.0  14.0  18.0
```

```
#a
print("Count of total missing values in dataframe: ",df.isnull().sum().sum())
```

```
Count of total missing values in dataframe:  18
```

```
#b
print(df.dropna(axis = 1, how = 'any', thresh =46))
```

```
       1     2     3
0   17.0  19.0  17.0
1    8.0   4.0   9.0
2    3.0   8.0  18.0
3    8.0   5.0  17.0
4   17.0  12.0  13.0
5   17.0   4.0  12.0
6   18.0   1.0   9.0
7   13.0  13.0  15.0
8    2.0  10.0   8.0
9   17.0  12.0   NaN
10   2.0   NaN   1.0
11  17.0   9.0   5.0
12  18.0   NaN   9.0
13   NaN   5.0  18.0
14  19.0   2.0   9.0
15  12.0   7.0  13.0
16  10.0   7.0  12.0
17  11.0  15.0  19.0
18   9.0  18.0  16.0
19   NaN  13.0   9.0
20  10.0  13.0   4.0
21   4.0   9.0  10.0
22  19.0  16.0  13.0
23   NaN   7.0   NaN
24   2.0  13.0  19.0
25   8.0  15.0  16.0
26   NaN  12.0  13.0
27  15.0   NaN   1.0
28   1.0  13.0  17.0
29  18.0   1.0   8.0
30  11.0   1.0   3.0
31  15.0  10.0   9.0
32  18.0  10.0   5.0
33  13.0  10.0  19.0
34  12.0  12.0   4.0
35   9.0   4.0  18.0
36   9.0   3.0   7.0
37  12.0   8.0  11.0
38   4.0  13.0   9.0
39  10.0  15.0   6.0
```

```
40  12.0  16.0  11.0
41   6.0   9.0   9.0
42  15.0   4.0   2.0
43  16.0   NaN   6.0
44   1.0  12.0   NaN
45   5.0   1.0   NaN
46   1.0   5.0   5.0
47  14.0  18.0  16.0
48   5.0   1.0  17.0
49  16.0  14.0  18.0
```

```
#c
print(df.head())
print("The row having maximum sum: ",df.sum(axis=1).idxmax())
a=df.sum(axis=1).idxmax()
df = df.drop(a,axis=0)
df
```

```
      0     1     2     3
0   NaN  17.0  19.0  17.0
1   NaN   8.0   4.0   9.0
2  18.0   3.0   8.0  18.0
3  15.0   8.0   5.0  17.0
4  15.0  17.0  12.0  13.0
```

```
The row having maximum sum:  18
```

	0	1	2	3
0	NaN	17.0	19.0	17.0
1	NaN	8.0	4.0	9.0
2	18.0	3.0	8.0	18.0
3	15.0	8.0	5.0	17.0
4	15.0	17.0	12.0	13.0
5	18.0	17.0	4.0	12.0
6	3.0	18.0	1.0	9.0
7	9.0	13.0	13.0	15.0
8	13.0	2.0	10.0	8.0
9	9.0	17.0	12.0	NaN
10	11.0	2.0	NaN	1.0
11	7.0	17.0	9.0	5.0
12	14.0	18.0	NaN	9.0
13	17.0	NaN	5.0	18.0

...

35	7.0	9.0	4.0	18.0
36	16.0	9.0	3.0	7.0
37	8.0	12.0	8.0	11.0
38	9.0	4.0	13.0	9.0
39	1.0	10.0	15.0	6.0
40	16.0	12.0	16.0	11.0
41	4.0	6.0	9.0	9.0
42	18.0	15.0	4.0	2.0
43	12.0	16.0	NaN	6.0
44	13.0	1.0	12.0	NaN
45	6.0	5.0	1.0	NaN
46	10.0	1.0	5.0	5.0
47	9.0	14.0	18.0	16.0
48	15.0	5.0	1.0	17.0
49	7.0	16.0	14.0	18.0

```
#d
print("Sort the dataframe based on the 1st column: ",df.sort_values(by=1))
```

```
Sort the dataframe based on the 1st column:          0     1     2     3
28  19.0   1.0  13.0  17.0
46  10.0   1.0   5.0   5.0
44  13.0   1.0  12.0   NaN
24   9.0   2.0  13.0  19.0
10  11.0   2.0   NaN   1.0
8   13.0   2.0  10.0   8.0
2   18.0   3.0   8.0  18.0
38   9.0   4.0  13.0   9.0
21   1.0   4.0   9.0  10.0
48  15.0   5.0   1.0  17.0
45   6.0   5.0   1.0   NaN
41   4.0   6.0   9.0   9.0
25  15.0   8.0  15.0  16.0
3   15.0   8.0   5.0  17.0
1    NaN   8.0   4.0   9.0
36  16.0   9.0   3.0   7.0
35   7.0   9.0   4.0  18.0
39   1.0  10.0  15.0   6.0
16  17.0  10.0   7.0  12.0
20   NaN  10.0  13.0   4.0
30   2.0  11.0   1.0   3.0
17  14.0  11.0  15.0  19.0
34  16.0  12.0  12.0   4.0
37   8.0  12.0   8.0  11.0
15   3.0  12.0   7.0  13.0
40  16.0  12.0  16.0  11.0
33   NaN  13.0  10.0  19.0
7    9.0  13.0  13.0  15.0
47   9.0  14.0  18.0  16.0
42  18.0  15.0   4.0   2.0
31   2.0  15.0  10.0   9.0
27  17.0  15.0   NaN   1.0
43  12.0  16.0   NaN   6.0
49   7.0  16.0  14.0  18.0
11   7.0  17.0   9.0   5.0
9    9.0  17.0  12.0   NaN
5   18.0  17.0   4.0  12.0
4   15.0  17.0  12.0  13.0
0    NaN  17.0  19.0  17.0
29  10.0  18.0   1.0   8.0
12  14.0  18.0   NaN   9.0
6    3.0  18.0   1.0   9.0
32   9.0  18.0  10.0   5.0
22  10.0  19.0  16.0  13.0
14   8.0  19.0   2.0   9.0
```

```
13  17.0   NaN   5.0  18.0
19  12.0   NaN  13.0   9.0
23   NaN   NaN   7.0   NaN
26   NaN   NaN  12.0  13.0
```

```
#e
df = df.drop_duplicates(subset = 0, keep = 'first')
df
```

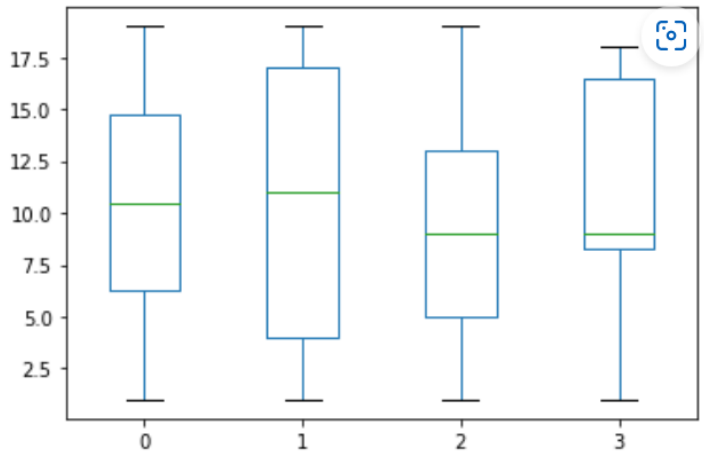
	0	1	2	3
0	NaN	17.0	19.0	17.0
2	18.0	3.0	8.0	18.0
3	15.0	8.0	5.0	17.0
6	3.0	18.0	1.0	9.0
7	9.0	13.0	13.0	15.0
8	13.0	2.0	10.0	8.0
10	11.0	2.0	NaN	1.0
11	7.0	17.0	9.0	5.0
11	7.0	17.0	9.0	5.0
12	14.0	18.0	NaN	9.0
13	17.0	NaN	5.0	18.0
14	8.0	19.0	2.0	9.0
19	12.0	NaN	13.0	9.0
21	1.0	4.0	9.0	10.0
22	10.0	19.0	16.0	13.0
28	19.0	1.0	13.0	17.0
30	2.0	11.0	1.0	3.0
34	16.0	12.0	12.0	4.0
41	4.0	6.0	9.0	9.0
45	6.0	5.0	1.0	NaN

```
#f
correlation = df[0].corr(df[1])
print("CORRELATION between column 1 and 2 : ", correlation)
covariance = df[1].cov(df[2])
print("COVARIANCE between column 2 and 3 :",covariance)
```

```
CORRELATION between column 1 and 2 :  -0.2462275087735184
COVARIANCE between column 2 and 3 : 2.095238095238095
```

```
#g
df.plot.box()
```

```
<AxesSubplot:>
```




```
#h
print(pd.cut(df[1],bins=5))
df
```

```
0     (15.4, 19.0]
2     (0.982, 4.6]
3       (4.6, 8.2]
6     (15.4, 19.0]
7     (11.8, 15.4]
8     (0.982, 4.6]
10    (0.982, 4.6]
11    (15.4, 19.0]
12    (15.4, 19.0]
13             NaN
14    (15.4, 19.0]
19             NaN
21    (0.982, 4.6]
22    (15.4, 19.0]
28    (0.982, 4.6]
30     (8.2, 11.8]
34    (11.8, 15.4]
41      (4.6, 8.2]
45      (4.6, 8.2]
Name: 1, dtype: category
Categories (5, interval[float64]): [(0.982, 4.6] < (4.6, 8.2] < (8.2, 11.8] < (11.8, 15.4] < (15.4, 1
9.0]]
```

Q #6.

Create a data frame to store marks of m students for n subjects and do the following:

i) Find average marks for each student and add as a column

ii) Display average marks of each subject and add as a new row

iii) Compute descriptive statistics subject-wise

iv) Compute grade obtained by each student as per the examination policy of your course (use lambda function) v) Find frequency of each grade for your class

vi) Find frequency of each grade obtained by each student and create a new DF as the following and set Rollno as the row index of the DF

Ans #6.

```
#6
subjects = {'Genetics': [87.0, 77.0, 79.0, 81.0], 'Healthcare': [98.0, 99.0, 91.0, 89.0], 'Maths': [87.
74.0]}
df1 = pd.DataFrame(subjects,index = ['Roger','Natalie','Meera','Rickie'])
df2 = df1.transpose()
df1
```

	Genetics	Healthcare	Maths	Physics	Literature
Roger	87.0	98.0	87.0	67.0	55.0
Natalie	77.0	99.0	65.0	68.0	80.0
Meera	79.0	91.0	69.0	79.0	71.0
Rickie	81.0	89.0	74.0	71.0	74.0

```
#a & b
df1['Avg'] = df1.mean(axis=1)
df1.loc['Sub_avg'] = df2.mean(axis=1)
print(df1)
```

```
Average of students is:
         Genetics  Healthcare  Maths  Physics  Literature   Avg
Roger        87.0       98.00  87.00    67.00        55.0  78.8
Natalie      77.0       99.00  65.00    68.00        80.0  77.8
Meera        79.0       91.00  69.00    79.00        71.0  77.8
Rickie       81.0       89.00  74.00    71.00        74.0  77.8
Sub_avg      81.0       94.25  73.75    71.25        70.0   NaN
```

```
#c
print("Descriptive statistics: ")
df1.describe()
```

```
Descriptive statistics:
```

	Genetics	Healthcare	Maths	Physics	Literature	Avg
count	5.000000	5.000000	5.000000	5.000000	5.000000	4.00
mean	81.000000	94.250000	73.750000	71.250000	70.000000	78.05
std	3.741657	4.322904	8.287792	4.710361	9.246621	0.50
min	77.000000	89.000000	65.000000	67.000000	55.000000	77.80
25%	79.000000	91.000000	69.000000	68.000000	70.000000	77.80
50%	81.000000	94.250000	73.750000	71.000000	71.000000	77.80
75%	81.000000	98.000000	74.000000	71.250000	74.000000	78.05
max	87.000000	99.000000	87.000000	79.000000	80.000000	78.80

```
#d
df1['Grade'] = df1.apply(lambda x: ("A" if x['Avg']>80 else ("B" if x['Avg']>70 else "C")),axis=1)
df1
```

	Genetics	Healthcare	Maths	Physics	Literature	Avg	Grade
Roger	87.0	98.00	87.00	67.00	55.0	78.8	B
Natalie	77.0	99.00	65.00	68.00	80.0	77.8	B
Meera	79.0	91.00	69.00	79.00	71.0	77.8	B
Rickie	81.0	89.00	74.00	71.00	74.0	77.8	B
Sub_avg	81.0	94.25	73.75	71.25	70.0	NaN	C

```
#e
df1.groupby('Grade').size()
```

```
Grade
B    4
C    1
dtype: int64
```

Q #7.

Consider two csv files of students of years 2019 and 2020 having following details (student name, hobby, course) where coruses are (CShons, BComhons, PSCS) and hobbies are (writing, painting, music, dancing). Answer the following

i) Find all hobbies types for each course in both years

ii) Find hobbies which are there in 2019 but not in 2020 for each course

iii) Find common hobbies in both year

iv) Find course name in which students are exploring all hobbies

v) Find count of students exploring each hobby in both year

```
import pandas as pd
frame1 = pd.read_csv(r'C:\Users\prateek\prac4\2019.csv')
df1 = pd.DataFrame(frame1)
print("2019 file : \n",df1)
frame2 = pd.read_csv(r'C:\Users\prateek\prac4\2020.csv)
df2 = pd.DataFrame(frame2)
print("\n2020 file : \n",df2)
```

```
(venv) PS C:\Users\hp> python -u "c:\Users\hp\Desktop\prac4\prac4.py"
2019 file :
     Name     Hobby     Course
0  Sohan   Writing    CS Hons
1   Riya  Painting       PSCS
2  Shyam   Writing  Bcom Hons
3   Siya   Dancing       PSCS

2020 file :
      Name    Hobby     Course
0   Priya  Singing       PSCS
1  Pushpa  Drawing       PSCS
2    Raju  Writing  Bcom Hons
3  Hemant  Dancing    CS Hons
(venv) PS C:\Users\hp>
```

```
df = pd.concat([df1,df2], axis=0, ignore_index=True)
df = df.groupby(['Course','Hobby']).size()
print(df)
```

```
(venv) PS C:\Users\hp> python -u "c:\Users\hp\Desktop\prac4\prac4.py"
Course      Hobby
Bcom Hons   Writing     2
CS Hons     Dancing     1
            Writing     1
PSCS        Dancing     1
            Drawing     1
            Painting    1
            Singing     1
dtype: int64
(venv) PS C:\Users\hp>
```

```
hobbies1 = df1.groupby(['Hobby']).groups
print("2019")
print(list(hobbies1))
hobbies2 = df2.groupby(['Hobby']).groups
print("\n2020")
print(list(hobbies2))
temp3 = []
temp3 += ([x for x in hobbies1 if x not in hobbies2])
print("\nhobbies in 2019 but not in 2020: \n",temp3)
```

```
(venv) PS C:\Users\hp> python -u "c:\Users\hp\Desktop\prac4\prac4.py"
2019
['Dancing', 'Painting', 'Writing']

2020
['Dancing', 'Drawing', 'Singing', 'Writing']

Hobbies in 2019 but not in 2020 :
 ['Painting']
(venv) PS C:\Users\hp>
```

```
hobbies1 = df1.groupby(['Hobby']).groups
print("2019")
print(list(hobbies))
hobbies = df2.groupby(['Hobby']).groups
print("\n2020")
print(list(hobbies2))
common = [value for value in hobbies1 if value in hobbies2]
print("\nCommon hobbies in both year are: \n",common)
```

```
(venv) PS C:\Users\hp> python -u "c:\Users\hp\Desktop\prac4\prac4.py"
2019
['Dancing', 'Painting', 'Writing']

2020
['Dancing', 'Drawing', 'Singing', 'Writing']

Common hobbies in both year are :
 ['Dancing', 'Writing']
(venv) PS C:\Users\hp>
```

```
grouped1 = df1.groupby(['Course', 'Hobby']) 11 courses = ["BcomHons", "Cshons", "PSCS"]
hobby= ["music","painting", "dancing", "writing"]
a = grouped1. first()
dic ={} 15 result =[]
for ind in filel.index:
    if filel["Course"][ind] in dic.keys():
dic[filel["Course"][ind]] (filel["Hobby"][inds
if(set(dic[filel["Course"][ind]]) == hobby):
    result.append(filel["Course"][ind])
else:
    dic[filel["Course"][ind]] = (filel["Hobby"][ind],)
print(dic)
```

```
(venv) PS C:\Users\hp> python -u "c:\Users\hp\Desktop\prac4\prac4.py"
{'CS Hons': ('Writing',), 'PSCS': ('Painting', 'Dancing'), 'Bcom Hons': ('Writing',)}
(venv) PS C:\Users\hp>
```

```
df = pd.concat([df1, df2], axis=e, ignore_index=True)
df df.groupby("Hobby').count()
print(df Course'])
```

```
(venv) PS C:\Users\hp> python -u "c:\Users\hp\Desktop\prac4\prac4.py"
Hobby
Dancing     2
Drawing     1
Painting    1
Singing     1
Writing     3
Name: Course, dtype: int64
(venv) PS C:\Users\hp>
```

Q #11.

```
#11
import requests
import json
response = requests.get('https://api.covid19api.com/summary').text
response_info = json.loads(response)
data = response_info['Countries']
df3 = pd.DataFrame(data)
df3
```

	ID	Country	CountryCode	Slug	NewConfirmed	TotalConfirmed	NewDeaths	TotalDeaths	NewRecover
0	b6389ddd-cef0-465a-9624-bfd19fb49908	Afghanistan	AF	afghanistan	102	203497	0	7825	
1	c0a68d82-5f2b-4d2b-98b8-4575fc32ee42	Albania	AL	albania	31	333027	0	3593	
2	7869cbfc-724b-4e05-8763-fe0dc6e30dda	Algeria	DZ	algeria	9	270856	0	6881	
3	45b4206e-5c6b-4813-8137-f5c2fc9a36ae	Andorra	AD	andorra	0	46588	0	155	
4	5629b1ca-50b3-4463-9299-a3d4c297970f	Angola	AO	angola	0	103131	0	1917	
...	...	...	...	...	...	...	...	...	
192	bc386ba3-7cd7-450c-9213-353619b37022	Venezuela (Bolivarian Republic)	VE	venezuela	12	545963	0	5820	
193	85ac2944-4a66-4356-ae48-d1a01ae02e1c	Viet Nam	VN	vietnam	339	11505249	0	43165	
194	86c4dcaf-3061-46b6-bf39-eeba9e5988bf	Yemen	YE	yemen	1	11945	1	2159	

Assignment 4 (Unit 3)

Q #1

Consider two excel files (construct yourself having minimum 20 records) having attendance of a workshop's participants for two days. Each file has three fields 'Name', 'Time of joining' (hh:mm:ss), 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:

a) Perform merging of the two dataframes to find the names of students who had attended the workshop on both days.

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

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

d) Merge two data frames and use two columns names and duration as multi-row indexes. Generate descriptive statistics for this multi-index where feature is categorical attribute.

e) Count number of rows with more than one NA values

Ans #1.

```
#1
import numpy as np
import pandas as pd
Day1 = pd.read_excel('attend1.xlsx')
Day2 = pd.read_excel('attend2.xlsx')
Day1
```

	Name	Time_of_joining	duration
0	Roger	12:23:34	30
1	Liz	23:34:56	40
2	Lery	01:23:56	50
3	Moraine	03:06:15	30
4	Ravi	23:04:42	40
5	Shubhash	14:34:23	50
6	Lann	12:23:16	40
7	Banff	10:17:15	30
8	Karan	23:20:29	40

```python
#a
both_days = pd.merge(Day1,Day2,on='Name',how='inner')
print("Students who attended both workshops: ")
print(both_days)
```

```
Students who attended both workshops:
    Name Time_of_joining_x  duration_x Time_of_joining_y  duration_y
0  Banff          10:17:15          30          10:17:15          50
1  Mandi          19:07:08          30          19:07:08          40
2   Watt          09:09:09          30          20:20:20          40
```

```python
#b
either_days = pd.merge(Day1,Day2,on='Name',how='outer')
print(either_days)
```

```
         Name Time_of_joining_x  duration_x Time_of_joining_y  duration_y
0       Roger          12:23:34        30.0                NaN         NaN
1         Liz          23:34:56        40.0                NaN         NaN
2        Lery          01:23:56        50.0                NaN         NaN
3     Moraine          03:06:15        30.0                NaN         NaN
4        Ravi          23:04:42        40.0                NaN         NaN
5    Shubhash          14:34:23        50.0                NaN         NaN
6        Lann          12:23:16        40.0                NaN         NaN
7       Banff          10:17:15        30.0           10:17:15        50.0
8       Karan          23:20:29        40.0                NaN         NaN
9      Kaveri          10:04:49        50.0                NaN         NaN
10    Pavitra          12:34:34        40.0                NaN         NaN
11    Bhupesh          12:34:23        50.0                NaN         NaN
12      Mandi          19:07:08        30.0           19:07:08        40.0
13    Natalie          23:45:13        30.0                NaN         NaN
14        Mac          20:45:06        30.0                NaN         NaN
15       Jack          23:34:56        40.0                NaN         NaN
16      Kilen          12:08:09        30.0                NaN         NaN
17       Watt          09:09:09        30.0           20:20:20        40.0
18      Aaron          08:08:03        50.0                NaN         NaN
19      Parul          02:34:09        50.0                NaN         NaN
20     Roger2               NaN         NaN           12:15:34        50.0
21       Liz2               NaN         NaN           15:34:56        40.0
22      Lery2               NaN         NaN           01:15:56        30.0
23   Moraine2               NaN         NaN           03:06:15        40.0
24      Ravi2               NaN         NaN           15:04:42        50.0
25  Shubhash2               NaN         NaN           14:34:15        30.0
26      Lann2               NaN         NaN           12:15:16        40.0
27     Karan2               NaN         NaN           15:20:29        40.0
```

```
#c
either_days['Name'].count()
```

```
37
```

```
both_days = pd.merge(Day1,Day2,how='outer',on=['Name','duration']).copy()
both_days.fillna(value='-',inplace=True)
both_days.set_index(['Name','duration'])
```

		Time_of_joining_x	Time_of_joining_y
Name	**duration**		
Roger	**30**	12:23:34	-
Liz	**40**	23:34:56	-
Lery	**50**	01:23:56	-
Moraine	**30**	03:06:15	-
Ravi	**40**	23:04:42	-
Shubhash	**50**	14:34:23	-
Lann	**40**	12:23:16	-
Banff	**30**	10:17:15	-

```
either_days.shape[1] - either_days.count(axis=1)
```

```
0     2
1     2
2     2
3     2
4     2
5     2
6     2
7     0
8     2
9     2
10    2
11    2
12    0
13    2
14    2
15    2
16    2
17    0
```

```
#e
either_days.isna().any(axis=1).sum()
```

```
34
```

Q #2.

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

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

b) Draw a scatter plot for Petal width vs sepal width.

c) Plot density distribution for feature petal length.

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

e) Compare five summary distribution information of two features petal width and sepal width using boxplots

f) Compare five point statistical summary of two features petal width and sepal width using appropriate graph

g) Draw a piechart showing distribution of three classes

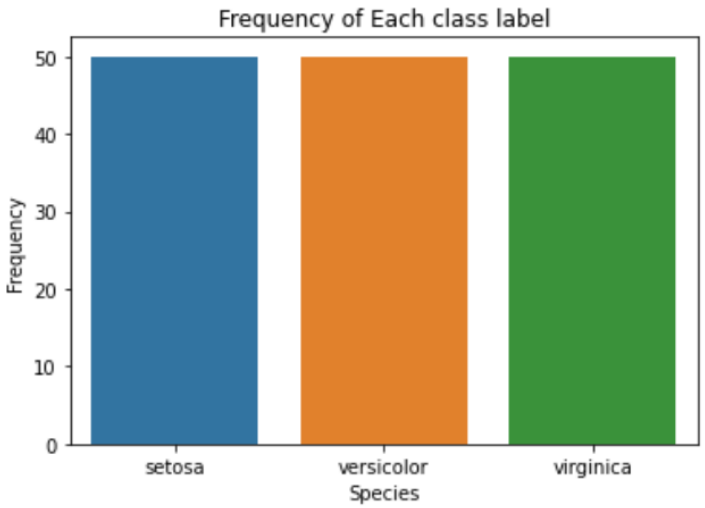
Ans #2.

```
#2
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
iris = sns.load_dataset('iris')
iris
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica

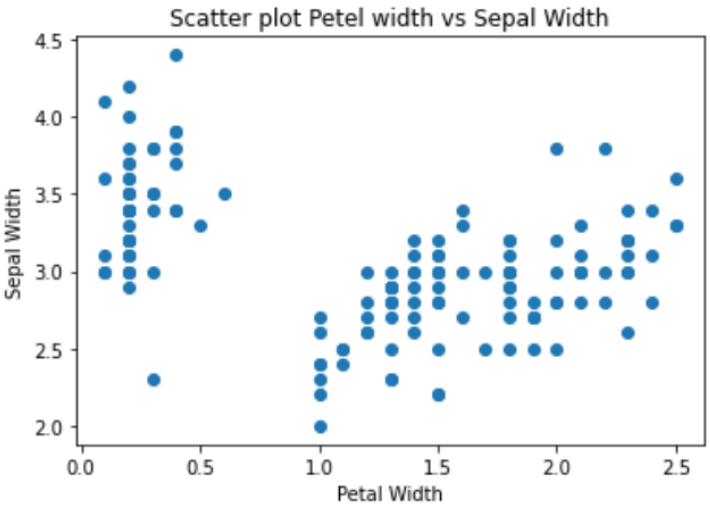
```
#a
sns.countplot(x='species',data=iris)
plt.xlabel('Species')
plt.ylabel('Frequency')
plt.title('Frequency of Each class label')
```

```
Text(0.5, 1.0, 'Frequency of Each class label')
```



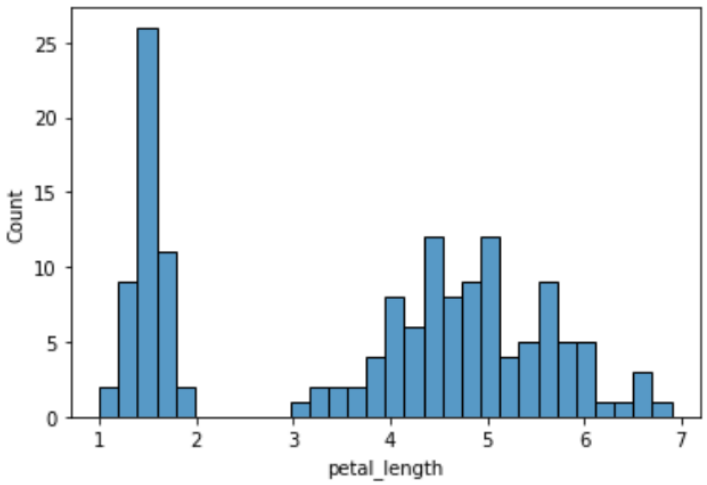

```
#b
plt.scatter(x='petal_width',y='sepal_width',data=iris)
plt.xlabel('Petal Width')
plt.ylabel('Sepal Width')
plt.title("Scatter plot Petel width vs Sepal Width")
```

```
Text(0.5, 1.0, 'Scatter plot Petel width vs Sepal Width')
```



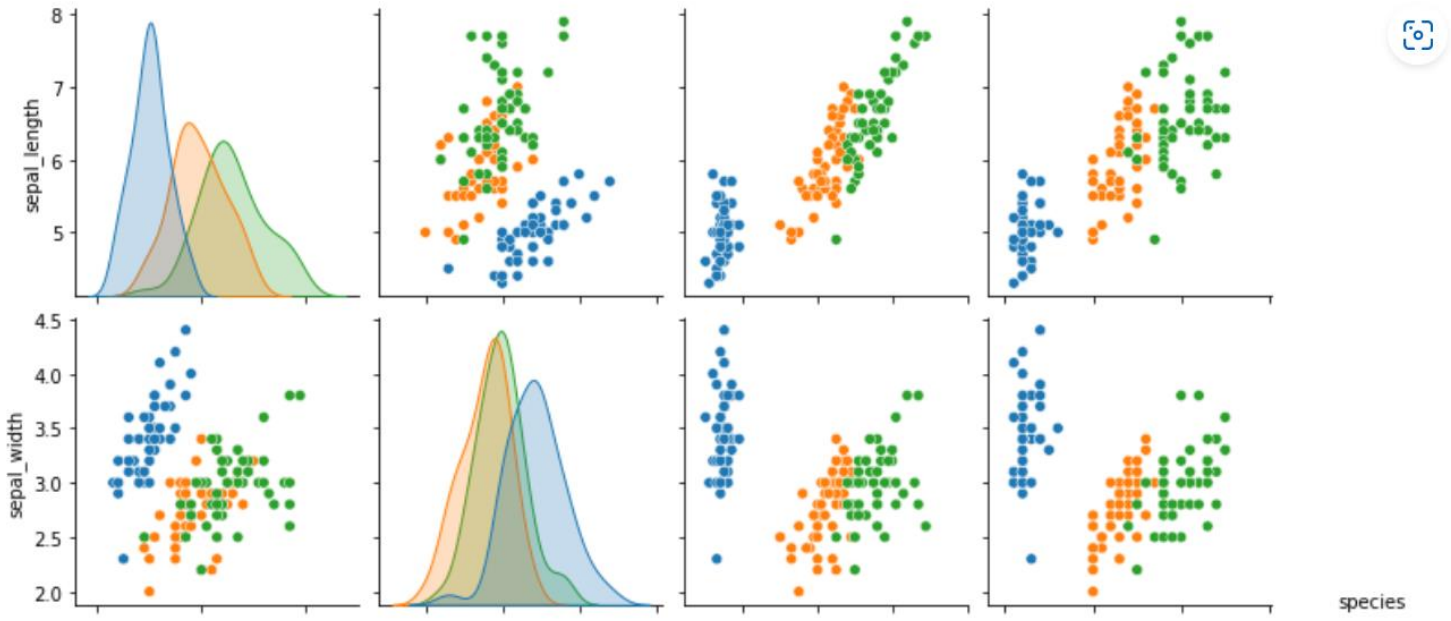
```
#c
sns.histplot(iris['petal_length'],kde=False,bins=30)
```

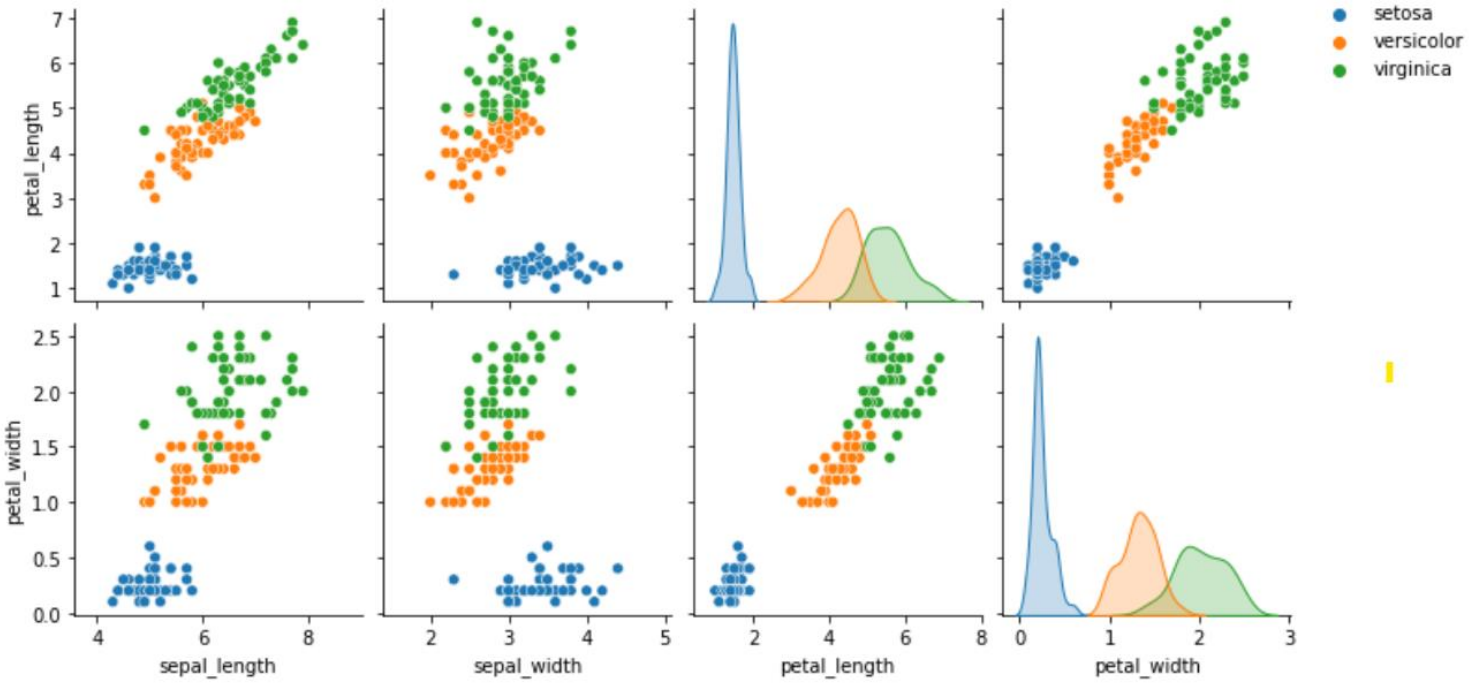
```
<AxesSubplot:xlabel='petal_length', ylabel='Count'>
```




```
#d
sns.pairplot(iris,hue='species')
```

```
<seaborn.axisgrid.PairGrid at 0x26bffdde100>
```




setosa
versicolor
virginica
petal_length
petal_width
sepal_length
sepal_width
petal_length
petal_width