Practical File

Name – Poonam

Roll No. - 10858

Course – B. Sc (Hons) Computer Science

Section – B

Semester - V

Subject – Data Analysis and Visualization

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

```
import numpy as np
Dict= {                          #generation of Dictionary
   "Boys":[72,68,70,69,74],
   "Girls":[63,65,69,62,61] }
print("Original Dictionary : ",Dict)


result = [{'Boys': boys_height, 'Girls': girls_height} for boys_height, girls_height in zip(Dict['Boys'], Dict['Girls'])]
print(result)
l = []
for i in range(len(Dict["Boys"])):
  l.append({"Boys": Dict["Boys"][i], "Girls": Dict["Girls"][i]})
```

Output

In [2]:
```
                                                  #Question 1
import numpy as np
Dict= {                                  #generation of Dictionary
    "Boys":[72,68,70,69,74],
    "Girls":[63,65,69,62,61] }
print("Original Dictionary : ",Dict)
```

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

In [5]:
```
result = [{'Boys': boys_height, 'Girls': girls_height} for boys_height, girls_height in zip(Dict['Boys'], Dict['Girls'])]
print(result)
l = []
for i in range(len(Dict["Boys"])):
  l.append({"Boys": Dict["Boys"][i], "Girls": Dict["Girls"][i]})
```

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

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.

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 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

```
import numpy as np
arr1 = np.random.randint(0, 200, size=(3,4))     #generation of randon array
print(arr1)

#Part - a
#mean
mean_arr = np.mean(arr1, axis=1)
print(mean_arr)

#standard Deviation
std_arr= np.std(arr1, axis=1)
print(std_arr)

#Variance
variance_arr = np.var(arr1, axis=1)
print(variance_arr)

#Part - b
B = np.array([56, 48, 22, 41, 78, 91, 24, 46, 8, 33])
indices = np.argsort(B)
print(indices)

#Part - c
m = int(input("Enter the number of rows(m) : "))
n = int(input("Enter the number of columns(n) : "))
arr2 = np.random.randint(0, 100, size=(m, n))
print(arr2)
print("Shape of array  :", arr2.shape)
print("Type of array  :", type(arr2))
print("Data type :", arr2.dtype)
reshaped_array = arr2.reshape(n, m)
print("Reshaped array :", reshaped_array)

#Part - d
arr2 = np.array([10, 3, 25, 52, 0, np.nan, 0, np.nan])
zero_indices = np.where(arr2 == 0)[0]
non_zero_indices = np.where(arr2 != 0)[0]
nan_indices = np.where(np.isnan(arr2))[0]
print("Array:", arr2)
print("Indices of Zero Elements are ", zero_indices)
print("Indices of Non-Zero Elements are", non_zero_indices)
print("Indices of NaN Elements are", nan_indices)
```

Output

```
                                                #Question 2

import numpy as np
arr1 = np.random.randint(0, 200, size=(3,4))     #generation of randon array
print(arr1)
```

```
[[ 23 147  68  46]
 [ 48 184   4 157]
 [197 116  83 182]]
```

```
[10]:                                           #Part - a
#mean
mean_arr = np.mean(arr1, axis=1)
print(mean_arr)

#standard Deviation
std_arr= np.std(arr1, axis=1)
print(std_arr)

#Variance
variance_arr = np.var(arr1, axis=1)
print(variance_arr)
```

```
[ 71.    98.25 144.5 ]
[46.67440412 74.51971216 46.78942188]
[2178.5    5553.1875 2189.25  ]
```

```
[11]:                                           #Part - b
B = np.array([56, 48, 22, 41, 78, 91, 24, 46, 8, 33])
indices = np.argsort(B)
print(indices)
```

```
[8 2 6 9 3 7 1 0 4 5]
```

```
[12]:                                           #Part - c
m = int(input("Enter the number of rows(m) : "))
n = int(input("Enter the number of columns(n) : "))
arr2 = np.random.randint(0, 100, size=(m, n))
print(arr2)

print("Shape of array  :", arr2.shape)
print("Type of array  :", type(arr2))
print("Data type :", arr2.dtype)

reshaped_array = arr2.reshape(n, m)
print("Reshaped array :", reshaped_array)
```

```
Enter the number of rows(m) : 3
Enter the number of columns(n) : 2
[[94 37]
 [20  5]
 [46 30]]
Shape of array  : (3, 2)
Type of array  : <class 'numpy.ndarray'>
Data type : int32
Reshaped array : [[94 37 20]
 [ 5 46 30]]
```

```
3]:                                             #Part - d
arr2 = np.array([10, 3, 25, 52, 0, np.nan, 0, np.nan])

zero_indices = np.where(arr2 == 0)[0]
non_zero_indices = np.where(arr2 != 0)[0]
nan_indices = np.where(np.isnan(arr2))[0]

print("Array:", arr2)
print("Indices of Zero Elements are ", zero_indices)
print("Indices of Non-Zero Elements are", non_zero_indices)
print("Indices of NaN Elements are", nan_indices)
```

```
Array: [10.  3. 25. 52.  0. nan  0. nan]
Indices of Zero Elements are  [4 6]
Indices of Non-Zero Elements are [0 1 2 3 5 7]
Indices of NaN Elements are [5 7]
```

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 second and third column.
g. Detect the outliers and remove the rows having outliers.
h. Discretize second column and create 5 bins.

Code

```
import pandas as pd
import numpy as np
r1= 50
c1= 3

data=np.random.randn(r1,c1)
null_index= np.random.choice([True,False],size=(r1,c1),p=[0.10,0.90])
data[null_index]=np.nan
df=pd.DataFrame(data,columns=["First column","Second column","Third column"])
print(abs(df))

#part -A
df1=df.isnull().sum()
df1

#part -B
df.dropna(axis=1,thresh=45)

#part -C
a=df.sum(axis=1).idxmax()
df.drop(index=a)

#part -D
df.sort_values('First column')

#part -E
df.drop_duplicates('First column')

#part - F
df['First column'].corr(df['Second column'])

df['Second column'].cov(df['Third column'])

#part - G
outlier=pd.Series(data=False,index=df.index)
for col in df.columns:

    Q1= df[col].quantile(0.25)
    Q3= df[col].quantile(0.75)
```

```
    IQR=Q3-Q1

    lower_bound = Q1-(1.5 * IQR)
    upper_bound = Q3+(1.5 * IQR)
    outlier |= (df[col] < lower_bound) | (df[col] > upper_bound)
df=df[~outlier]
print(df)

 #part -H
df['Second column']= pd.cut(df['Second column'],bins=5)
```

Output

```
:                                       #Question-3
import pandas as pd
import numpy as np
r1= 50
c1= 3

data=np.random.randn(r1,c1)
null_index= np.random.choice([True,False],size=(r1,c1),p=[0.10,0.90])
data[null_index]=np.nan
df=pd.DataFrame(data,columns=["First column","Second column","Third column"])
print(abs(df))
```

	First column	Second column	Third column
0	0.228111	NaN	1.726669
1	0.613255	0.994162	0.129132
2	0.884030	0.078211	NaN
3	NaN	NaN	0.422133
4	0.756703	0.782885	0.891896
5	1.275814	0.083754	0.367872
6	1.285298	0.103647	0.627299
7	NaN	0.036455	1.245829
8	0.782837	1.496655	0.149763
9	1.323033	0.500998	0.317684
10	1.171566	0.429270	1.224706
11	0.706385	0.594991	0.132243
12	0.765036	0.102585	0.049464
13	0.559781	0.961744	0.620749
14	NaN	0.178661	1.602048
15	1.103450	1.380712	0.665275
16	1.902848	0.670232	1.831214
17	0.651222	0.954976	0.072868
18	0.497239	0.437618	0.453964
19	0.035528	0.318433	0.365497
20	0.032423	0.248372	1.934635
21	NaN	0.883031	NaN
22	1.386056	0.273582	1.223784
23	0.515873	0.479048	0.626410
24	0.497163	NaN	0.669067
25	0.006325	1.207553	1.726786
26	0.491003	0.430826	1.638820
27	0.329114	1.122936	1.448619
28	0.506823	1.447161	1.176364
29	1.796743	0.318859	0.229446
30	0.303093	0.137800	0.168453
31	1.298823	1.048775	0.675485
32	0.105765	0.736271	1.386905
33	1.748889	0.822482	1.432422
34	0.539684	0.761647	0.764708
35	2.009205	1.008071	NaN
36	0.916409	0.494376	1.442296
37	1.089841	0.874339	0.295120
38	NaN	0.870870	0.757583

```
38          NaN       0.870870      0.757583
39     0.880547       0.335459      1.325049
40     0.267593       1.172731      1.397460
41     0.211578       2.433989      0.444625
42     0.675851       0.731638      0.143735
43          NaN       0.936717      0.042491
44     1.714430       0.884834      1.222605
45     1.406542       0.959802      0.085730
46     1.022574       0.566003      0.612659
47          NaN       1.194853      0.436772
48     0.846133       0.080953           NaN
49     0.416000       0.832899      0.067532
```

12]:
```
#part -A
df1=df.isnull().sum()
df1
```

12]:
```
First column     11
Second column     6
Third column      6
dtype: int64
```

```
#part -B
df.dropna(axis=1,thresh=45)
```

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14

```
#part -C
a=df.sum(axis=1).idxmax()
df.drop(index=a)
```

	First column	Second column	Third column
0	0.142946	1.666602	1.765267
1	NaN	2.241660	0.889981
2	-0.693036	-0.055222	0.428419
3	0.046681	0.552210	-1.767049
4	-0.309975	0.615354	1.618406
5	0.237075	-0.699066	-0.756225
6	-1.284017	1.038479	1.897157
7	1.478610	-0.167375	0.879906
8	-0.504064	NaN	-0.314312
9	0.640795	-0.707162	0.919359
10	0.858632	1.748261	-0.093536
11	-0.256727	-0.892532	-1.070173
12	-0.332279	NaN	-0.236813
13	-0.240332	0.746446	0.060218

]:
```
#part -D
df.sort_values('First column')
```

]:

	First column	Second column	Third column
28	-2.497323	-1.263930	-0.814487
40	-2.418404	NaN	0.853587
41	-1.705613	0.291684	1.370186
6	-1.284017	1.038479	1.897157
38	-1.029464	0.716691	-0.914037
49	-1.024168	-0.211686	1.533853
21	-0.928350	NaN	-0.094555
17	-0.871283	2.125944	-1.156573
29	-0.757035	-1.056147	-0.202384
2	-0.693036	-0.055222	0.428419
8	-0.504064	NaN	-0.314312
43	-0.472520	-0.043631	0.647330
26	-0.398626	2.040004	0.021939
19	-0.369312	-0.811645	0.791040
12	-0.332279	NaN	-0.236813
4	-0.309975	0.615354	1.618406
11	-0.256727	-0.892532	-1.070173
13	-0.240332	0.746446	0.060218
44	-0.229164	0.759227	0.235136
32	-0.214770	0.832799	-0.395481
16	0.033418	1.219102	-0.565634
3	0.046681	0.552210	-1.767049
0	0.142946	1.666602	1.765267
20	0.182063	NaN	0.043177
48	0.189471	-0.093777	1.983992
25	0.200243	NaN	-0.263076
5	0.237075	-0.699066	-0.756225
45	0.466376	0.474657	-1.970707
23	0.484497	0.434143	-0.505100
15	0.529815	NaN	-1.004072
24	0.530868	1.408826	0.085766
39	0.544254	-1.275421	NaN

:
```
#part -E
df.drop_duplicates('First column')
```

:

	First column	Second column	Third column
0	-0.422677	NaN	1.427797
1	NaN	-0.866924	0.032074
2	0.612307	1.990506	-0.551327
3	0.067600	0.955814	1.828941
4	0.659705	0.654610	-0.385438
5	-0.261606	0.184553	0.492171
6	-2.167295	-0.510525	0.441451
7	-0.154179	1.907968	NaN
8	0.662775	0.310312	0.092452
9	0.420816	-0.219309	-0.075802
10	0.165566	0.061488	0.725821
11	-0.572211	-0.285222	-1.273520
12	-0.143635	-2.150827	0.721438
13	1.868457	0.255567	0.865023

```
                                                    #part - F
df['First column'].corr(df['Second column'])
```

```
0.1307157916606341
```

```
df['Second column'].cov(df['Third column'])
```

```
0.017426545355214214
```

```
:                                                   #part - G
outlier=pd.Series(data=False,index=df.index)
for col in df.columns:

        Q1= df[col].quantile(0.25)
        Q3= df[col].quantile(0.75)
        IQR=Q3-Q1

        lower_bound = Q1-(1.5 * IQR)
        upper_bound = Q3+(1.5 * IQR)
        outlier |= (df[col] < lower_bound) | (df[col] > upper_bound)
df=df[~outlier]
print(df)
```

```
    First column  Second column  Third column
0      -0.228111            NaN     -1.726669
1      -0.613255      -0.994162     -0.129132
2      -0.884030       0.078211           NaN
3            NaN            NaN      0.422133
4       0.756703       0.782885     -0.891896
5       1.275814      -0.083754     -0.367872
6       1.285298      -0.103647      0.627299
7            NaN      -0.036455     -1.245829
8      -0.782837      -1.496655     -0.149763
9      -1.323033      -0.500998      0.317684
10      1.171566       0.429270      1.224706
11      0.706385      -0.594991      0.132243
12     -0.765036       0.102585      0.049464
13      0.559781      -0.961744      0.620749
14           NaN       0.178661     -1.602048
15     -1.103450      -1.380712     -0.665275
16     -1.902848      -0.670232     -1.831214
17     -0.651222      -0.954976      0.072868
18     -0.497239       0.437618      0.453964
19     -0.035528       0.318433     -0.365497
20      0.032423      -0.248372      1.934635
21           NaN       0.883031           NaN
22      1.386056       0.273582     -1.223784
23      0.515873      -0.479048     -0.626410
24      0.497163            NaN     -0.669067
25      0.006325      -1.207553      1.726786
26      0.491003       0.430826     -1.638820
27      0.329114       1.122936     -1.448619
28      0.506823      -1.447161      1.176364
29     -1.796743       0.318859      0.229446
30     -0.303093       0.137800      0.168453
31      1.298823      -1.048775     -0.675485
32     -0.105765       0.736271      1.386905
33      1.748889       0.822482     -1.432422
34      0.539684       0.761647     -0.764708
35     -2.009205       1.008071           NaN
36      0.916409      -0.494376      1.442296
37      1.089841       0.874339      0.295120
38           NaN      -0.870870     -0.757583
```

```
                                                    #part -H
df['Second column']= pd.cut(df['Second column'],bins=5)
```

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:

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.

Code

```
#Question -4
File1=r"C:\Users\CSLab\Documents\Poonam HTMl\Attendence1.xlsx"
File2 = r"C:\Users\CSLab\Documents\Poonam HTMl\Attendence2.xlsx"
import pandas as pd
df=pd.read_excel(File1)
df1=pd.read_excel(File2)

df.parse_dates = ("Time of joining")
df

df1.parse_dates=("Time of joining")
df1

#part -a
df.merge(df1,how="inner",on="Name")

#part - b
df.merge(df1,how="outer")

#part -c
a=pd.concat([df,df1],ignore_index=True)
a
len(a)

#part -d
b=df.merge(df1,how="outer")
b
c=b.set_index(keys=["Name","Duration"])
c
c.describe()
```

Output

```
#Question -4
```

```
File1=r"C:\Users\CSLab\Documents\Poonam HTMl\Attendence1.xlsx"
File2 = r"C:\Users\CSLab\Documents\Poonam HTMl\Attendence2.xlsx"
```

```
import pandas as pd
df=pd.read_excel(File1)
df1=pd.read_excel(File2)
```

```
df.parse_dates = ("Time of joining")
df
```

	Name	Time of joining	Duration
0	Riya	17:00:00	30
1	Dhruv	17:50:00	50
2	Kiran	12:00:00	40
3	Ayush	22:45:00	50
4	Priyanjali	09:22:00	30
5	Sanskriti	15:32:00	40
6	Aditi	20:33:00	50
7	Neeraj	22:23:00	30
8	Saurav	14:00:00	50
9	Gaurav	13:55:00	50

```
df1.parse_dates=("Time of joining")
df1
```

```
df1.parse_dates=("Time of joining")
df1
```

	Name	Time of joining	Duration
0	Prathvi	02:08:08	30
1	Rishika	03:56:04	40
2	samyak	15:04:05	30
3	Riya	04:04:06	50
4	Sanskriti	17:05:05	30
5	Aditya	01:04:05	40
6	Anshuman	16:05:06	40
7	Geet	02:09:09	50
8	Bebika	16:09:09	30
9	Dhruv	20:08:05	50

```
                    #part -a
df.merge(df1,how="inner",on="Name")
```

	Name	Time of joining_x	Duration_x	Time of joining_y	Duration_y
0	Riya	17:00:00	30	04:04:06	50
1	Dhruv	17:50:00	50	20:08:05	50
2	Sanskriti	15:32:00	40	17:05:05	30

```
#part - b
df.merge(df1,how="outer")
```

	Name	Time of joining	Duration
0	Riya	17:00:00	30
1	Dhruv	17:50:00	50
2	Kiran	12:00:00	40
3	Ayush	22:45:00	50
4	Priyanjali	09:22:00	30
5	Sanskriti	15:32:00	40
6	Aditi	20:33:00	50
7	Neeraj	22:23:00	30
8	Saurav	14:00:00	50
9	Gaurav	13:55:00	50
10	Prathvi	02:08:08	30
11	Rishika	03:56:04	40
12	samyak	15:04:05	30
13	Riya	04:04:06	50
14	Sanskriti	17:05:05	30
15	Aditya	01:04:05	40
16	Anshuman	16:05:06	40
17	Geet	02:09:09	50
18	Bebika	16:09:09	30
19	Dhruv	20:08:05	50

```
#part -c
a=pd.concat([df,df1],ignore_index=True)
a
```

```
len(a)
```

```
20
```

```
#part -d
b=df.merge(df1,how="outer")
b
```

	Name	Time of joining	Duration
0	Riya	17:00:00	30
1	Dhruv	17:50:00	50
2	Kiran	12:00:00	40
3	Ayush	22:45:00	50
4	Priyanjali	09:22:00	30
5	Sanskriti	15:32:00	40
6	Aditi	20:33:00	50
7	Neeraj	22:23:00	30
8	Saurav	14:00:00	50
9	Gaurav	13:55:00	50
10	Prathvi	02:08:08	30

11	Rishika	03:56:04	40
12	samyak	15:04:05	30
13	Riya	04:04:06	50
14	Sanskriti	17:05:05	30
15	Aditya	01:04:05	40
16	Anshuman	16:05:06	40
17	Geet	02:09:09	50
18	Bebika	16:09:09	30
19	Dhruv	20:08:05	50

```
c=b.set_index(keys=["Name","Duration"])
c
```

```
c.describe()
```

	Time of joining
count	20
unique	20
top	17:00:00
freq	1

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)

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.

Code

```
#Question -5
from matplotlib import pyplot as plt
import seaborn
import pandas as pd
import numpy as np
file1 = pd.read_csv(r"C:\Users\CSLab\Downloads\iris\iris.data",header=None,)
file1
file1.columns=["SepalLengthCm","SepalWidthCm" ,"PetalLengthCm","PetalWidthCm","Species"]
file1

 #part - A
frequency=file1["Species"].value_counts()
frequency
plt.hist(file1["Species"])

#part - B
a=file1["PetalWidthCm"]
b=file1["SepalWidthCm"]
plt.scatter(a,b)

#part - C
x=file1["PetalLengthCm"]
x.plot.density()

#part - D
seaborn.pairplot(file1)
```

Output

```
#                                    Question -5
```

```
from matplotlib import pyplot as plt
import seaborn
import pandas as pd
import numpy as np
```

```
file1 = pd.read_csv(r"C:\Users\CSLab\Downloads\iris\iris.data",header=None,)
file1
```

	0	1	2	3	4
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

```
file1.columns=["SepalLengthCm","SepalWidthCm" ,"PetalLengthCm","PetalWidthCm","Species"]
file1
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

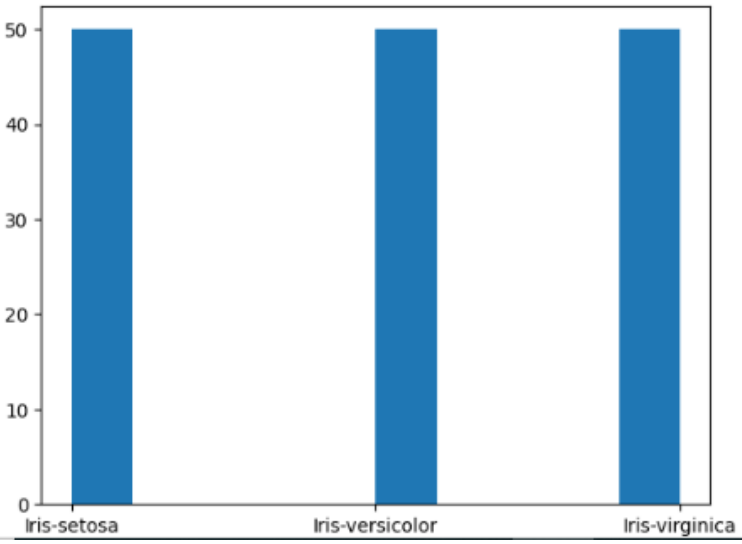
150 rows × 5 columns

47]:
```
#part - A
frequency=file1["Species"].value_counts()
frequency
```

47]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

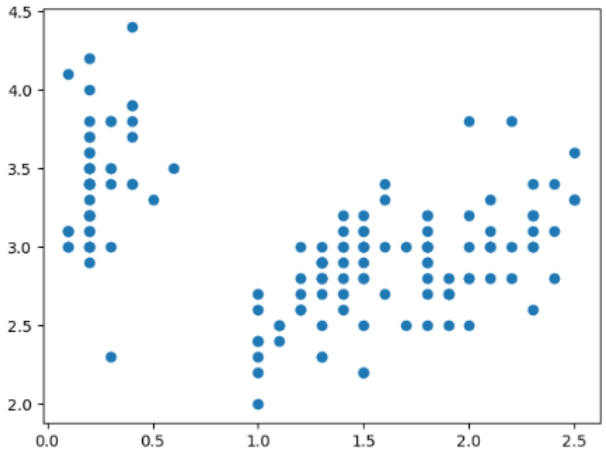
72]:
```
plt.hist(file1["Species"])
```

72]:
```
(array([50.,  0.,  0.,  0.,  0., 50.,  0.,  0.,  0., 50.]),
 array([0. , 0.2, 0.4, 0.6, 0.8, 1. , 1.2, 1.4, 1.6, 1.8, 2. ]),
 <BarContainer object of 10 artists>)
```



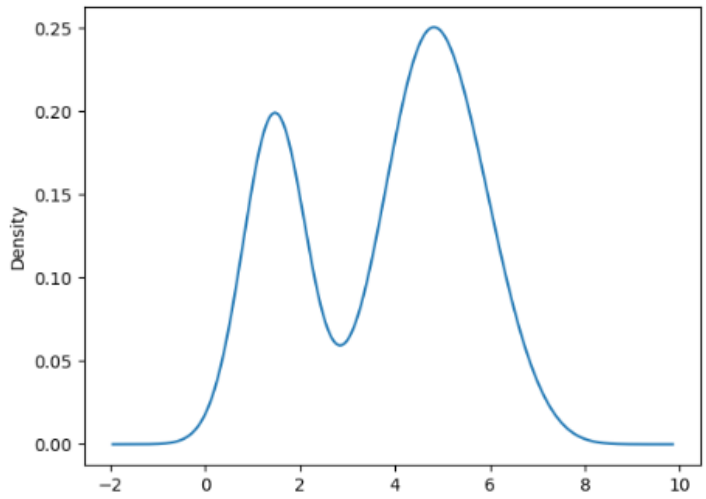
:
```
#part - B
a=file1["PetalWidthCm"]
b=file1["SepalWidthCm"]
plt.scatter(a,b)
```

:
```
<matplotlib.collections.PathCollection at 0x1f4aa03b760>
```



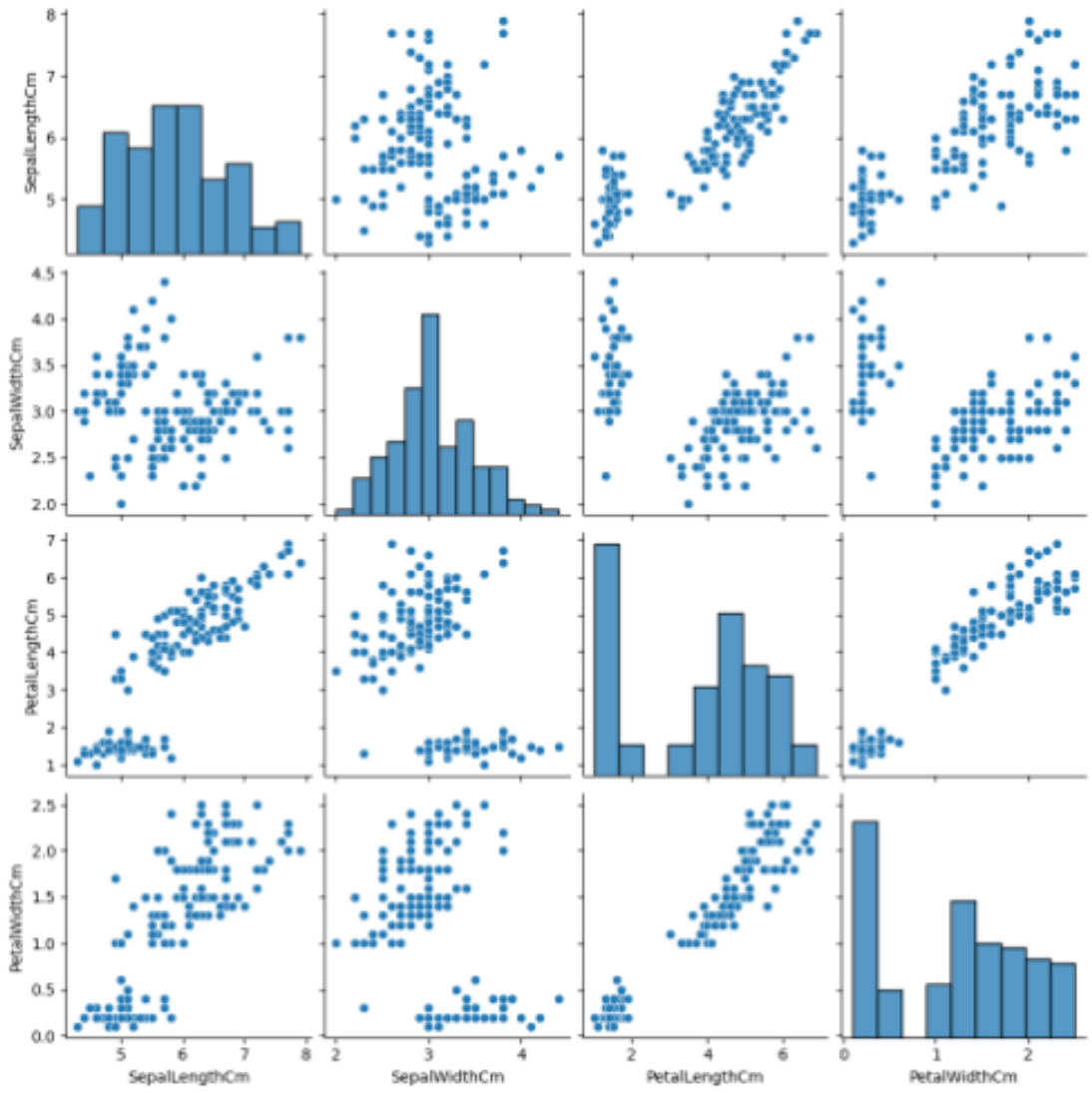
```
                                        #part - C
x=file1["PetalLengthCm"]
x.plot.density()
```

```
<AxesSubplot:ylabel='Density'>
```



81]:
```
                                        #part - 4
seaborn.pairplot(file1)
```

81]:
```
<seaborn.axisgrid.PairGrid at 0x1f4ae5b3430>
```



Ques.6 Consider any sales training/ weather forecasting dataset

a. Compute mean of a series grouped by another series

b. Fill an intermittent time series to replace all missing dates with values of previous non-missing date.

c. Perform appropriate year-month string to dates conversion.

d. Split a dataset to group by two columns and then sort the aggregated results within the groups.

e. Split a given dataframe into groups with bin counts.

Code

```
#Question - 6
import pandas as pd
import numpy as np
from datetime import datetime
df=pd.read_csv(r"C:\Users\CSLab\Documents\Poonam HTMl\Dav\transactions.csv")
df

#part -A
df1=df.groupby('date')['transactions'].mean()
df1
#part -B
df['date'].fillna(method="ffill" , inplace=True)
df
df.isnull().sum()

#part -C
df['date']
dt=list()
from datetime import datetime
for i in df['date']:
   dt.append(datetime.strptime(str(i),'%d-%m-%Y'))
dt
#part -D
df3=df.groupby(['date','store_nbr']).agg({'transactions' :sum })
res=df3['transactions'].groupby(level=1)
print(res.nlargest())

#part -E
df5=pd.cut(df['store_nbr'],bins=5)
df5
```

Output

```
#Question - 6
import pandas as pd
import numpy as np
from datetime import datetime
```

```
df=pd.read_csv(r"C:\Users\CSLab\Documents\Poonam HTMl\Dav\transactions.csv")
df
```

	date	store_nbr	transactions
0	01-01-2013	25	770
1	02-01-2013	1	2111
2	02-01-2013	2	2358
3	02-01-2013	3	3487
4	02-01-2013	4	1922
...	...	...	...
83483	15-08-2017	50	2804
83484	15-08-2017	51	1573
83485	15-08-2017	52	2255
83486	15-08-2017	53	932
83487	15-08-2017	54	802

```
                                                #part -A
df1=df.groupby('date')['transactions'].mean()
df1
```

```
date
01-01-2013     770.000000
01-01-2014     663.500000
01-01-2015    2202.000000
01-01-2017    1642.000000
01-02-2013    1702.217391
                  ...
31-10-2016    1461.307692
31-12-2013    2493.914894
31-12-2014    2644.833333
31-12-2015    2487.283019
31-12-2016    2557.886792
Name: transactions, Length: 1682, dtype: float64
```

```
                                                #part -B
df['date'].fillna(method="ffill" , inplace=True)
df
```

	date	store_nbr	transactions
0	01-01-2013	25	770
1	02-01-2013	1	2111
2	02-01-2013	2	2358
3	02-01-2013	3	3487
4	02-01-2013	4	1922
...	...	...	...
83483	15-08-2017	50	2804
83484	15-08-2017	51	1573
83485	15-08-2017	52	2255
83486	15-08-2017	53	932
83487	15-08-2017	54	802

83488 rows × 3 columns

```
df.isnull().sum()
```

```
date            0
store_nbr       0
transactions    0
dtype: int64
```

```
                                                #part -D
df3=df.groupby(['date','store_nbr']).agg({'transactions' :sum })
res=df3['transactions'].groupby(level=1)
print(res.nlargest())
```

```
store_nbr  date        store_nbr
1          23-12-2016  1            3023
           23-12-2014  1            2861
           23-12-2013  1            2848
           24-12-2013  1            2844
           23-12-2015  1            2833
                                    ...
54         24-12-2014  54           1811
           24-12-2016  54           1807
           24-12-2013  54           1756
           24-12-2015  54           1726
           31-12-2016  54           1647
Name: transactions, Length: 270, dtype: int64
```

n [21]:
```
                                    #part -C
df['date']
```

ut[21]:
```
0        01-01-2013
1        02-01-2013
2        02-01-2013
3        02-01-2013
4        02-01-2013
            ...
83483    15-08-2017
83484    15-08-2017
83485    15-08-2017
83486    15-08-2017
83487    15-08-2017
Name: date, Length: 83488, dtype: object
```

n [22]:
```
dt=list()
from datetime import datetime
for i in df['date']:
    dt.append(datetime.strptime(str(i),'%d-%m-%Y'))
dt
```

ut[22]:
```
[datetime.datetime(2013, 1, 1, 0, 0),
 datetime.datetime(2013, 1, 2, 0, 0),
 datetime.datetime(2013, 1, 2, 0, 0),
 datetime.datetime(2013, 1, 2, 0, 0),
 datetime.datetime(2013, 1, 2, 0, 0),
 datetime.datetime(2013, 1, 2, 0, 0),
 datetime.datetime(2013, 1, 2, 0, 0),
 datetime.datetime(2013, 1, 2, 0, 0),
 datetime.datetime(2013, 1, 2, 0, 0),
 datetime.datetime(2013, 1, 2, 0, 0),
 datetime.datetime(2013, 1, 2, 0, 0),
 datetime.datetime(2013, 1, 2, 0, 0),
 datetime.datetime(2013, 1, 2, 0, 0),
 datetime.datetime(2013, 1, 2, 0, 0),
 datetime.datetime(2013, 1, 2, 0, 0),
 datetime.datetime(2013, 1, 2, 0, 0),
 datetime.datetime(2013, 1, 2, 0, 0),
 datetime.datetime(2013, 1, 2, 0, 0),
 datetime.datetime(2013, 1, 2, 0, 0),
```

4]:
```
                                    #part -E
df5=pd.cut(df['store_nbr'],bins=5)
df5
```

4]:
```
0         (22.2, 32.8]
1        (0.947, 11.6]
2        (0.947, 11.6]
3        (0.947, 11.6]
4        (0.947, 11.6]
             ...
83483     (43.4, 54.0]
83484     (43.4, 54.0]
83485     (43.4, 54.0]
83486     (43.4, 54.0]
83487     (43.4, 54.0]
Name: store_nbr, Length: 83488, dtype: category
Categories (5, interval[float64, right]): [(0.947, 11.6] < (11.6, 22.2] < (22.2, 32.8] < (32.8, 43.4] < (43.4, 54.0]]
```

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

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

Code

```
import pandas as pd
import numpy as np
data={
   'Name' : ['Mudit Chauhan','Seema Chopra','Rani Gupta','Aditya Narayan','Sanjeev Sahni','Prakash Kumar','Ritu Aggarwal','Akshay Goel','Meeta Kulkarni','Preeti Ahuja','Sunil Das Gupta','Sonali Sapre','Rashmi Talwar','Ashish Dubey','Kiran Sharma','Sameer Bansal'],
   'Birth_Month' : ['Dec','Jan','Mar','Oct','Feb','Dec','Sep','Aug','Jul','Nov','Apr','Jan','Jun','May','Feb','Oct'],
   'Gender' : ['M','F','F','M','M','M','F','M','F','F','M','F','F','M','F','M'],
   'Pass_Division' : ['III','II','I','I','II','III','I','I','II','II','III','I','III','II','II','I']
}
df=pd.DataFrame(data)
df

#Part - A
df=pd.get_dummies(df,columns=['Gender','Pass_Division'])
df

#Part-B
df["Birth_Month"]=df.Birth_Month.astype("category")
```

df
df.dtypes

month=['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
df['Birth_Month']=pd.Categorical(df['Birth_Month'] , categories = month) #used for passing the list for sort values
df.sort_values(by='Birth_Month' ,inplace=True)
df

Output

```
import pandas as pd
import numpy as np
```

```
                                    #Question - 7
data={
    'Name' : ['Mudit Chauhan','Seema Chopra','Rani Gupta','Aditya Narayan','Sanjeev Sahni','Prakash Kumar','Ritu Aggarwal','Aksha
    'Birth_Month' : ['Dec','Jan','Mar','Oct','Feb','Dec','Sep','Aug','Jul','Nov','Apr','Jan','Jun','May','Feb','Oct'],
    'Gender' : ['M','F','F','M','M','M','F','M','F','F','M','F','F','M','F','M'],
    'Pass_Division' : ['III','II','I','I','II','III','I','I','II','II','III','I','III','II','II','I']
}
df=pd.DataFrame(data)
df
```

	Name	Birth_Month	Gender	Pass_Division
0	Mudit Chauhan	Dec	M	III
1	Seema Chopra	Jan	F	II
2	Rani Gupta	Mar	F	I
3	Aditya Narayan	Oct	M	I
4	Sanjeev Sahni	Feb	M	II
5	Prakash Kumar	Dec	M	III
6	Ritu Aggarwal	Sep	F	I
7	Akshay Goel	Aug	M	I
8	Meeta Kulkarni	Jul	F	II
9	Preeti Ahuja	Nov	F	II
10	Sunil Das Gupta	Apr	M	III
11	Sonali Sapre	Jan	F	I
12	Rashmi Talwar	Jun	F	III
13	Ashish Dubey	May	M	II
14	Kiran Sharma	Feb	F	II
15	Sameer Bansal	Oct	M	I

```
                                    #Part - A
df=pd.get_dummies(df,columns=['Gender','Pass_Division'])
```

```
df
```

	Name	Birth_Month	Gender_F	Gender_M	Pass_Division_I	Pass_Division_II	Pass_Division_III
0	Mudit Chauhan	Dec	0	1	0	0	1
1	Seema Chopra	Jan	1	0	0	1	0
2	Rani Gupta	Mar	1	0	1	0	0
3	Aditya Narayan	Oct	0	1	1	0	0
4	Sanjeev Sahni	Feb	0	1	0	1	0
5	Prakash Kumar	Dec	0	1	0	0	1
6	Ritu Aggarwal	Sep	1	0	1	0	0
7	Akshay Goel	Aug	0	1	1	0	0
8	Meeta Kulkarni	Jul	1	0	0	1	0
9	Preeti Ahuja	Nov	1	0	0	1	0
10	Sunil Das Gupta	Apr	0	1	0	0	1
11	Sonali Sapre	Jan	1	0	1	0	0
12	Rashmi Talwar	Jun	1	0	0	0	1
13	Ashish Dubey	May	0	1	0	1	0
14	Kiran Sharma	Feb	1	0	0	1	0
15	Sameer Bansal	Oct	0	1	1	0	0

```
                                                    #Part-B
df["Birth_Month"]=df.Birth_Month.astype("category")
df
```

	Name	Birth_Month	Gender_F	Gender_M	Pass_Division_I	Pass_Division_II	Pass_Division_III
0	Mudit Chauhan	Dec	0	1	0	0	1
1	Seema Chopra	Jan	1	0	0	1	0
2	Rani Gupta	Mar	1	0	1	0	0
3	Aditya Narayan	Oct	0	1	1	0	0
4	Sanjeev Sahni	Feb	0	1	0	1	0
5	Prakash Kumar	Dec	0	1	0	0	1
6	Ritu Aggarwal	Sep	1	0	1	0	0
7	Akshay Goel	Aug	0	1	1	0	0
8	Meeta Kulkarni	Jul	1	0	0	1	0
9	Preeti Ahuja	Nov	1	0	0	1	0
10	Sunil Das Gupta	Apr	0	1	0	0	1
11	Sonali Sapre	Jan	1	0	1	0	0
12	Rashmi Talwar	Jun	1	0	0	0	1
13	Ashish Dubey	May	0	1	0	1	0
14	Kiran Sharma	Feb	1	0	0	1	0
15	Sameer Bansal	Oct	0	1	1	0	0

```
df.dtypes
```

```
Name                object
Birth_Month       category
Gender_F             uint8
Gender_M             uint8
Pass_Division_I      uint8
Pass_Division_II     uint8
Pass_Division_III    uint8
dtype: object
```

```
month=['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec']
df['Birth_Month']=pd.Categorical(df['Birth_Month'] , categories = month)     #used for passing the list for sort values
df.sort_values(by='Birth_Month' ,inplace=True)
df
```

	Name	Birth_Month	Gender_F	Gender_M	Pass_Division_I	Pass_Division_II	Pass_Division_III
1	Seema Chopra	Jan	1	0	0	1	0
11	Sonali Sapre	Jan	1	0	1	0	0
4	Sanjeev Sahni	Feb	0	1	0	1	0
14	Kiran Sharma	Feb	1	0	0	1	0
2	Rani Gupta	Mar	1	0	1	0	0
10	Sunil Das Gupta	Apr	0	1	0	0	1
13	Ashish Dubey	May	0	1	0	1	0
12	Rashmi Talwar	Jun	1	0	0	0	1
8	Meeta Kulkarni	Jul	1	0	0	1	0
7	Akshay Goel	Aug	0	1	1	0	0
6	Ritu Aggarwal	Sep	1	0	1	0	0
3	Aditya Narayan	Oct	0	1	1	0	0
15	Sameer Bansal	Oct	0	1	1	0	0
9	Preeti Ahuja	Nov	1	0	0	1	0
0	Mudit Chauhan	Dec	0	1	0	0	1

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

Code

```
#Question- 8
data=pd.read_excel(r"C:\Users\CSLab\Documents\Poonam HTMl\Dav\Ques_8.xlsx")
data

#part - A
data.groupby(['Name'])['MonthlyIncome(Rs.)'].sum()

#part-B
data.groupby(['Name'])['MonthlyIncome(Rs.)'].max()

#part-C
df=data[data['MonthlyIncome(Rs.)'] >60000.00]
df
df2=data.groupby(['Name','Gender'])['MonthlyIncome(Rs.)'].mean()
df2

#part-D
df1=data.groupby(['Name','Gender']).mean().query("Name=='Shah' & Gender=='Female'")
df1

#part-D(Another Method)
df4=data[(data['Name']=='Shah') & (data['Gender']=='Female')]['MonthlyIncome(Rs.)'].mean()
df4
```

Output

```
#Question- 8
data=pd.read_excel(r"C:\Users\CSLab\Documents\Poonam HTML\Dav\Ques_8.xlsx")
data
```

	Name	Gender	MonthlyIncome(Rs.)
0	Shah	Male	114000
1	Vats	Male	65000
2	Vats	Female	43150
3	Kumar	Female	69500
4	Vats	Female	155000
5	Kumar	Male	103000
6	Shah	Male	55000
7	Shah	Female	112400
8	Kumar	Female	81030
9	Vats	Male	71900

```
#part - A
data.groupby(['Name'])['MonthlyIncome(Rs.)'].sum()
```

```
Name
Kumar    253530
Shah     281400
Vats     335050
Name: MonthlyIncome(Rs.), dtype: int64
```

```
#part-B
data.groupby(['Name'])['MonthlyIncome(Rs.)'].max()
```

```
Name
Kumar    103000
Shah     114000
Vats     155000
Name: MonthlyIncome(Rs.), dtype: int64
```

```
#part-C
df=data[data['MonthlyIncome(Rs.)'] >60000.00]
df
```

	Name	Gender	MonthlyIncome(Rs.)
0	Shah	Male	114000
1	Vats	Male	65000
3	Kumar	Female	69500
4	Vats	Female	155000
5	Kumar	Male	103000
7	Shah	Female	112400
8	Kumar	Female	81030
9	Vats	Male	71900

```
df2=data.groupby(['Name','Gender'])['MonthlyIncome(Rs.)'].mean()
df2
```

```
Name   Gender
Kumar  Female     75265.0
       Male      103000.0
Shah   Female    112400.0
       Male       84500.0
```

```
       Male       84500.0
Vats   Female     99075.0
       Male       68450.0
Name: MonthlyIncome(Rs.), dtype: float64
```

```
                                        #part-D
df1=data.groupby(['Name','Gender']).mean().query("Name=='Shah'")              #just for checking
df1
```

		MonthlyIncome(Rs.)
Name	**Gender**	
Shah	**Female**	112400.0
	Male	84500.0

```
df1=data.groupby(['Name','Gender']).mean().query("Name=='Shah' & Gender=='Female'")
df1
```

		MonthlyIncome(Rs.)
Name	**Gender**	
Shah	**Female**	112400.0

```
                                                #part-D(Another Method)
df4=data[(data['Name']=='Shah') & (data['Gender']=='Female')]['MonthlyIncome(Rs.)'].mean()
df4
```

```
112400.0
```

**********End Of The Programs**********

Poonam

(10858)