

# INDEX

SNO	PRACTICALS
1	<p><b>Ques1.</b> 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}]</p>
2	<p><b>Ques2.</b> Write programs in Python using NumPy library to do the following:</p> <ol style="list-style-type: none"> <li>Compute the mean, standard deviation, and variance of a two dimensional random integer array along the second axis.</li> <li>Get the indices of the sorted elements of a given array.  a. B = [56, 48, 22, 41, 78, 91, 24, 46, 8, 33]</li> <li>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.</li> <li>Test whether the elements of a given array are zero, non-zero and NaN. Record the indices of these elements in three separate arrays.</li> </ol>
3	<p><b>Ques3.</b> 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:</p> <ol style="list-style-type: none"> <li>Identify and count missing values in a dataframe.</li> <li>Drop the column having more than 5 null values.</li> <li>Identify the row label having maximum of the sum of all values in a row and drop that row.</li> <li>Sort the dataframe on the basis of the first column.</li> <li>Remove all duplicates from the first column.</li> <li>Find the correlation between first and second column and covariance between second and third column.</li> <li>Detect the outliers and remove the rows having outliers.</li> <li>Discretize second column and create 5 bins</li> </ol>
4	<p><b>Ques4.</b> 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> <ol style="list-style-type: none"> <li>Perform merging of the two dataframes to find the names of students who had attended the workshop on both days.</li> <li>Find names of all students who have attended workshop on either of the days.</li> <li>Merge two data frames row-wise and find the total number of records in the data frame</li> <li>Merge two data frames and use two columns names and duration as multi-row indexes. Generate descriptive statistics for this multi-index.</li> </ol>
5	<p><b>Ques5.</b> Taking Iris data, plot the following with proper legend and axis labels: (Download IRIS data from: <a href="https://archive.ics.uci.edu/ml/datasets/iris">https://archive.ics.uci.edu/ml/datasets/iris</a> or import it from sklearn.datasets</p>

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.

6

**Ques6.** 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

7

**Ques7.** 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.

8

**Ques8.** 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

	<p>Write a program in Python using Pandas to perform the following:</p> <ul style="list-style-type: none"><li>a. Calculate and display familywise gross monthly income.</li><li>b. Calculate and display the member with the highest monthly income in a family.</li><li>c. Calculate and display monthly income of all members with income greater than Rs. 60000.00.</li><li>d. Calculate and display the average monthly income of the female members in the Shah family.</li></ul>
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**QUESTION1:** 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}].

Solution:

Code

```
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):
```

```
    tl={}
```

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

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

```
    final.append(tl)
```

```
print(final)
```

**OUTPUT**

localhost:8889/notebooks/Untitled6.ipynb?kernel\_name=python3

Gmail YouTube Maps Instantiating a Pet...

jupyter Untitled6 Last Checkpoint: a minute ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
In [1]: 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 [ ]:

**QUESTION2:** 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.

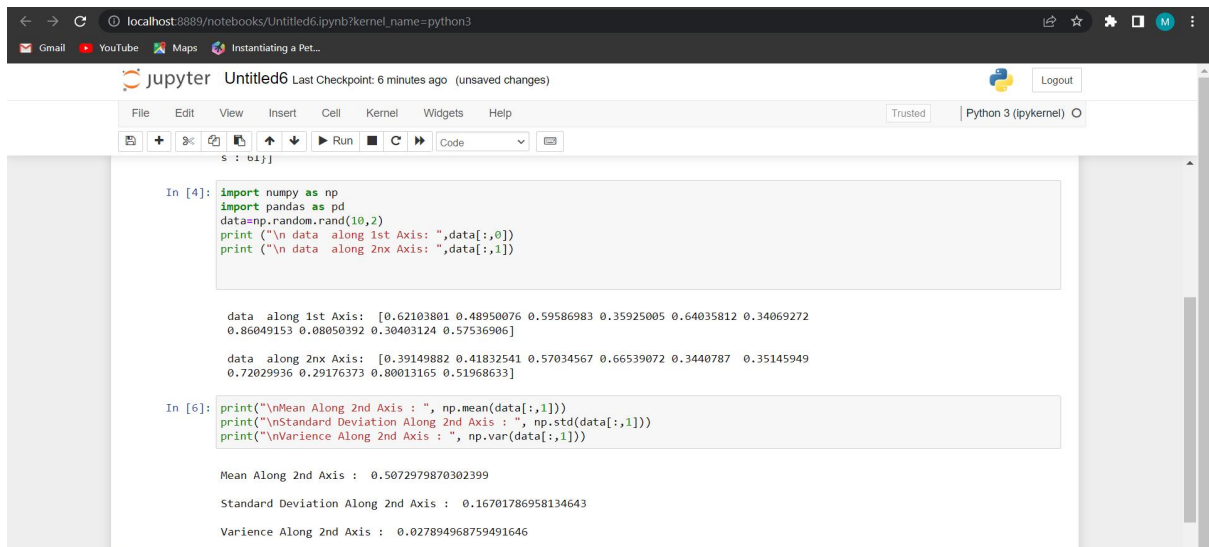
d) Test whether the elements of a given array are zero, non-zero and NaN. Record the indices of these elements in three separate arrays.

## SOLUTION

### CODE: (a)

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

### OUTPUT



```
In [4]: 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])

data along 1st Axis: [0.62103801 0.48950076 0.59586983 0.35925005 0.64035812 0.34069272
0.86049153 0.08050392 0.30403124 0.57536906]

data along 2nx Axis: [0.39149882 0.41832541 0.57034567 0.66539072 0.3440787 0.35145949
0.72029936 0.29176373 0.80013165 0.51968633]

In [6]: 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]))

Mean Along 2nd Axis : 0.5072979870302399
Standard Deviation Along 2nd Axis : 0.16701786958134643
Variance Along 2nd Axis : 0.027894968759491646
```

(b)

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

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

OUTPUT



```
In [7]: B = [56, 48, 22, 41, 78, 91, 24, 46, 8, 33]
print("Indices of sorted Array is :- ",np.argsort(B))

Indices of sorted Array is :- [8 2 6 9 3 7 1 0 4 5]
```

(c)

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

Enter Size of 2D Array :- 2 2

m=int(m) n=int(n) data=np.random.rand(m,n) print(data)

In [12]: 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)

[[0.71331983 0.38658805]
 [0.96232483 0.12130755]]
(2, 2)
Type of the Array is :- <class 'numpy.float64'>
Data Type of the Array is :- <class 'numpy.ndarray'>
Array Reshaped Successfully
[[0.71331983 0.38658805]
 [0.96232483 0.12130755]]

```

(d)

```

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

localhost:8889/notebooks/Untitled6.ipynb?kernel\_name=python3

Gmail YouTube Maps Instantiating a Pet...

Jupyter Untitled6 Last Checkpoint: 16 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [14]:

```
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 :- 5  
2  
4  
5  
0  
0  
Zeros Present :- 2  
nan Present :- 0  
Non-zero Present :- 3

### **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 2<sup>nd</sup> & 3<sup>rd</sup> column.
- g. Detect the outliers and remove the rows having outliers.**
- h. Discretize second column and create 5 bins**

Solution:

Code:

(a)

```
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,),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)

```

## OUTPUT

```

In [15]: 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)

```

Number of Missing Values :- 15

(b)

```

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

```
In [21]: 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)
```

(c)

```
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

```
In [22]: 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")
```

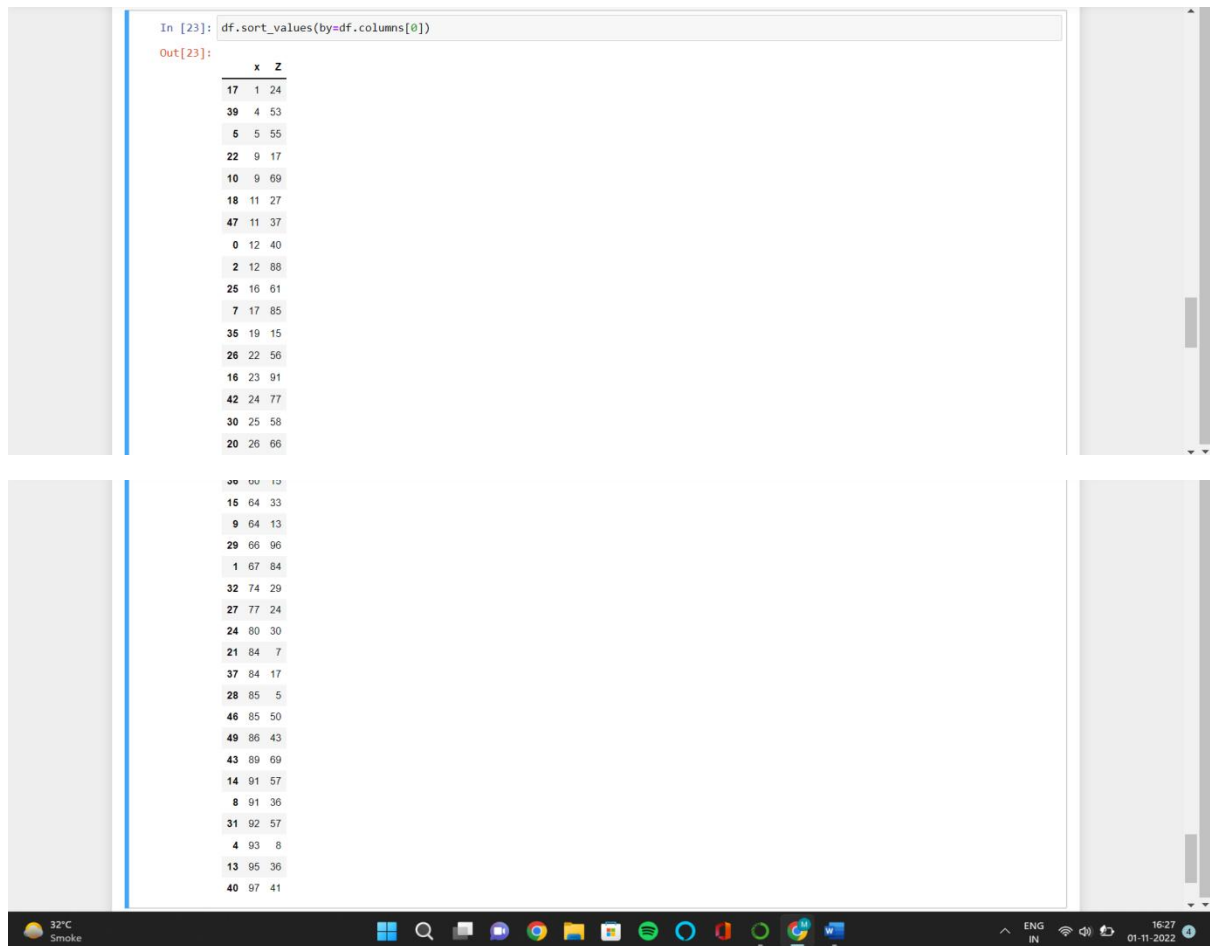
Dropped row with sum of elements as maximum value

```
In [ ]:
```

(d)

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

## OUTPUT



In [23]: df.sort\_values(by=df.columns[0])

Out[23]:

	x	z
17	1	24
39	4	53
5	5	55
22	9	17
10	9	69
18	11	27
47	11	37
0	12	40
2	12	88
25	16	61
7	17	85
36	19	15
26	22	56
16	23	91
42	24	77
30	25	58
20	26	66
30	66	12
15	64	33
9	64	13
29	66	96
1	67	84
32	74	29
27	77	24
24	80	30
21	84	7
37	84	17
28	85	5
46	85	50
49	86	43
43	89	69
14	91	57
8	91	36
31	92	57
4	93	8
13	95	36
40	97	41

(e)

```
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)
```

```

13  95  35
40  97  41

In [24]: 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 [ ]:
```

(f)

```
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**

```

In [26]: 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"]));

Correlation between 1st and 2nd column is :-  [[1.          0.13649036]
 [0.13649036  1.          ]]
Covariance between 2nd and 3rd column is :-  [[ 1.          -0.20640441]
 [-0.20640441  1.          ]]

In [ ]:
```

(g) 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**

```
In [8]: import seaborn as sns
df['x'][22]=200
sns.boxplot(x=df['x'])

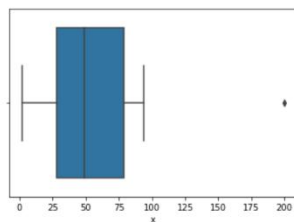
ol=np.where(df['x']>100)
print(ol)
df.drop(ol[0])

(array([22], dtype=int64),)
```

```
Out[8]:
```

	x	Y	Z
0	64	88	65
1	4	30	70
2	92	75	33
3	63	33	68
4	3	46	75
5	79	46	62
6	2	40	92
7	17	59	28
8	27	98	35
9	45	2	34
10	84	4	41
11	77	14	57
12	21	29	51
13	39	1	16
14	79	95	32
15	33	38	95
16	43	24	8
17	33	39	17
18	65	34	58
19	85	36	57
20	85	76	40
21	38	91	37
23	14	73	78
24	68	49	21
25	94	87	6

```
46 73 27 65
47 89 50 75
48 67 58 8
49 9 28 6
```



```
In [ ]:
```

**Code h) :**

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

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

```
print(temp)
```



## OUTPUT

```
In [9]: edges=[0,20,40,60,80,100]
temp=pd.cut(df.iloc[:,2],edges)
print(temp)
```

```
0    (60, 80]
1    (60, 80]
2    (20, 40]
3    (60, 80]
4    (60, 80]
5    (60, 80]
6    (80, 100]
7    (20, 40]
8    (20, 40]
9    (20, 40]
10   (40, 60]
11   (40, 60]
12   (40, 60]
13   (0, 20]
14   (20, 40]
15   (80, 100]
16   (0, 20]
17   (0, 20]
18   (40, 60]
19   (40, 60]
20   (20, 40]
21   (20, 40]
22   (40, 60]
23   (60, 80]
24   (20, 40]
25   (0, 20]
26   (40, 60]
27   (80, 100]
28   (60, 80]
```

#### 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.

#### Solution

##### FILE1

	A	B	C	D	E
1		NAME	TIME OF JOINING	DURATION	
2	1	dev	9:00AM	30	
3	2	Priyanshu	9:00AM	30	
4	3	poonam	3:00PM	40	
5	4	kamal	2:00PM	50	
6	5	nishant	10:00AM	50	
7	6	Mohak	8:00AM	30	
8	7	yonit	6:30PM	30	
9	8	Rohan	8:30AM	40	
10	9	Sahil	9:00AM	50	
11	10	babita	8:00AM	40	
12					
13					
14					
15					
16					

##### File2

	A	B	C	D	E
1		NAME	TIME OF JOINING	DURATION	
2	1	yonit	9:00AM	30	
3	2	Neelam	9:00AM	30	
4	3	Pankaj	3:00PM	40	
5	4	Priya	2:00PM	50	
6	5	Sonu	10:00AM	50	
7	6	KAMAL	8:00AM	30	
8	7	Suresh	6:30PM	30	
9	8	BABITA	8:30AM	40	
10	9	Dev	9:00AM	50	
11	10	POONAM	8:00AM	40	
12					

## CODE

```
import pandas as pd;

import numpy as np;

df1=pd.read_excel(r"C:\Users\Muskan\Downloads\ques4_file1.xlsx")

df2=pd.read_excel(r"C:\Users\Muskan\Downloads\ques4_file2.xlsx")

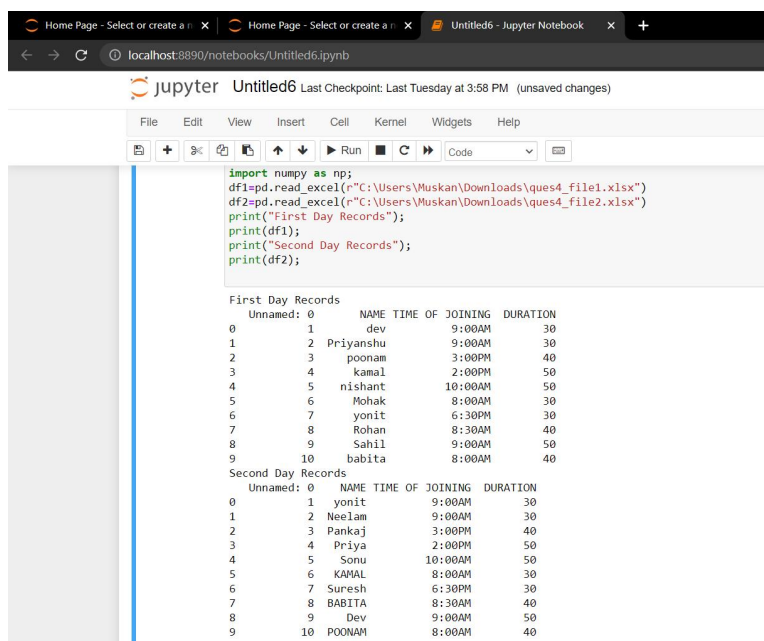
print("First Day Records");

print(df1);

print("Second Day Records");

print(df2);
```

## OUTPUT



```
import numpy as np;
df1=pd.read_excel(r"C:\Users\Muskan\Downloads\ques4_file1.xlsx")
df2=pd.read_excel(r"C:\Users\Muskan\Downloads\ques4_file2.xlsx")
print("First Day Records");
print(df1);
print("Second Day Records");
print(df2);
```

First Day Records

Unnamed: 0	NAME	TIME OF JOINING	DURATION
0	1	dev	9:00AM
1	2	Priyanshu	9:00AM
2	3	poonam	3:00PM
3	4	kamal	2:00PM
4	5	nishant	10:00AM
5	6	Mohak	8:00AM
6	7	yonit	6:30PM
7	8	Rohan	8:30AM
8	9	Sahil	9:00AM
9	10	babita	8:00AM

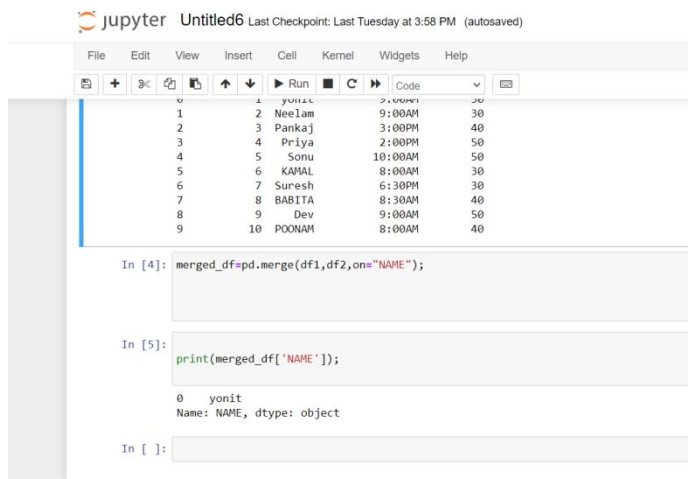
Second Day Records

Unnamed: 0	NAME	TIME OF JOINING	DURATION
0	1	yonit	9:00AM
1	2	Neelam	9:00AM
2	3	Pankaj	3:00PM
3	4	Priya	2:00PM
4	5	Sonu	10:00AM
5	6	KAMAL	8:00AM
6	7	Suresh	6:30PM
7	8	BABITA	8:30AM
8	9	Dev	9:00AM
9	10	POONAM	8:00AM

(a)

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

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

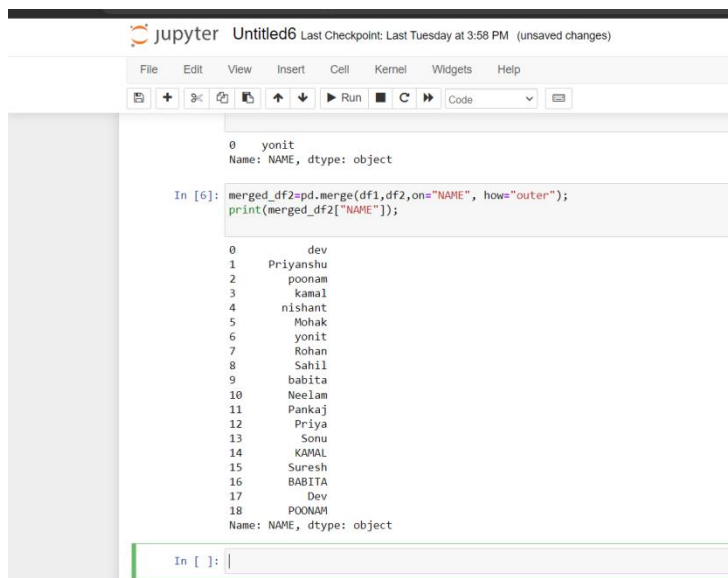


The screenshot shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar. A DataFrame is displayed with 10 rows and 5 columns. Below the DataFrame, two code cells are shown. The first cell contains the command `merged_df=pd.merge(df1,df2,on="NAME")`. The second cell contains `print(merged_df['NAME'])`, which outputs the following text:

```
0 yonit
Name: NAME, dtype: object
```

0	yonit	3:00PM	30	
1	Neelam	9:00AM	30	
2	Pankaj	3:00PM	40	
3	Priya	2:00PM	50	
4	Sonu	10:00AM	50	
5	KAMAL	8:00AM	30	
6	Suresh	6:30PM	30	
7	BABITA	8:30AM	40	
8	Dev	9:00AM	50	
9	POONAM	8:00AM	40	

(b) `merged_df2=pd.merge(df1,df2,on="NAME", how="outer");`  
`print(merged_df2["NAME"]);`



The screenshot shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar. Below the toolbar, the output from the previous cell is visible: `0 yonit` and `Name: NAME, dtype: object`. A new code cell contains the command `merged_df2=pd.merge(df1,df2,on="NAME", how="outer");` followed by `print(merged_df2["NAME"]);`. The output of this cell is a list of names:

```
0 dev
1 Priyanshu
2 poonam
3 kamal
4 nishant
5 Mohak
6 yonit
7 Rohan
8 Sahil
9 babita
10 Neelam
11 Pankaj
12 Priya
13 Sonu
14 KAMAL
15 Suresh
16 BABITA
17 Dev
18 POONAM
Name: NAME, dtype: object
```

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

The screenshot shows a Jupyter Notebook titled 'Untitled6' with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar. The code cell contains the following Python code:

```

15 Suresh
16 BABITA
17 Dev
18 POONAM
Name: NAME, dtype: object

In [7]: merged_df3=pd.concat([df1,df2]);
        print(merged_df3);

```

The output of the code is a pandas DataFrame with the following structure:

Unnamed: 0		NAME	TIME OF JOINING	DURATION
0	1	dev	9:00AM	30
1	2	Priyanshu	9:00AM	30
2	3	poonam	3:00PM	40
3	4	kamal	2:00PM	50
4	5	nishant	10:00AM	50
5	6	Mohak	8:00AM	30
6	7	yonit	6:30PM	30
7	8	Rohan	8:30AM	40
8	9	Sahil	9:00AM	50
9	10	babita	8:00AM	40
0	1	yonit	9:00AM	30
1	2	Neelam	9:00AM	30
2	3	Pankaj	3:00PM	40
3	4	Priya	2:00PM	50
4	5	Sonu	10:00AM	50
5	6	KANAL	8:00AM	30
6	7	Suresh	6:30PM	30
7	8	BABITA	8:30AM	40
8	9	Dev	9:00AM	50
9	10	POONAM	8:00AM	40

(d)

```

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');

# print(merged_df4);
# for i in range(len(merged_df4)):
#     print(i);
for i in range(len(merged_df4)):
    merged_df4['TIME OF JOINING'][i]=merged_df4['TIME OF JOINING'][i].replace('AM','');
    merged_df4['TIME OF JOINING'][i]=merged_df4['TIME OF JOINING'][i].replace('PM','');

merged_df4['TIME OF JOINING']=merged_df4['TIME OF JOINING'].astype(int);

```

```
print(merged_df4);
```

```
print ("Sum is \n",merged_df4.sum(0),"\\n Mean is ",merged_df4.mean(0),"\\n Standard  
Deviation is ",merged_df4.std(0),);
```

Jupyter Untitled7 Last Checkpoint: 31 minutes ago (autosaved)

NAME	DURATION	TIME OF JOINING
BABITA	40	8
Dev	50	9
KAMAL	30	8
Mohak	30	8
Neelam	30	9
POONAM	40	8
Pankaj	40	3
Priya	50	2
Priyanshu	30	9
Rohan	40	8
Sahil	50	9
Sonu	50	10
Suresh	30	6
babita	40	8
dev	30	9
kamal	50	2
nishant	50	10
poonam	40	3
yonit	30	6
yonit	30	9
Sum is		
TIME OF JOINING	144	
dtype: int64		
Mean is	TIME OF JOINING	7.2
dtype: float64		
Standard Deviation is	TIME OF JOINING	2.627787
dtype: float64		

### Ques 5.

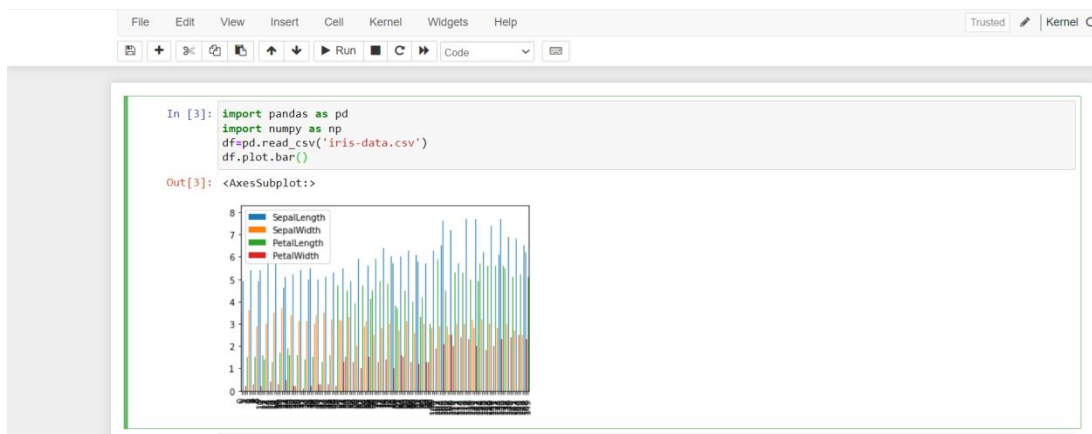
Taking Iris data, plot the following with proper legend and axis labels: (Download IRIS data from:

<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.

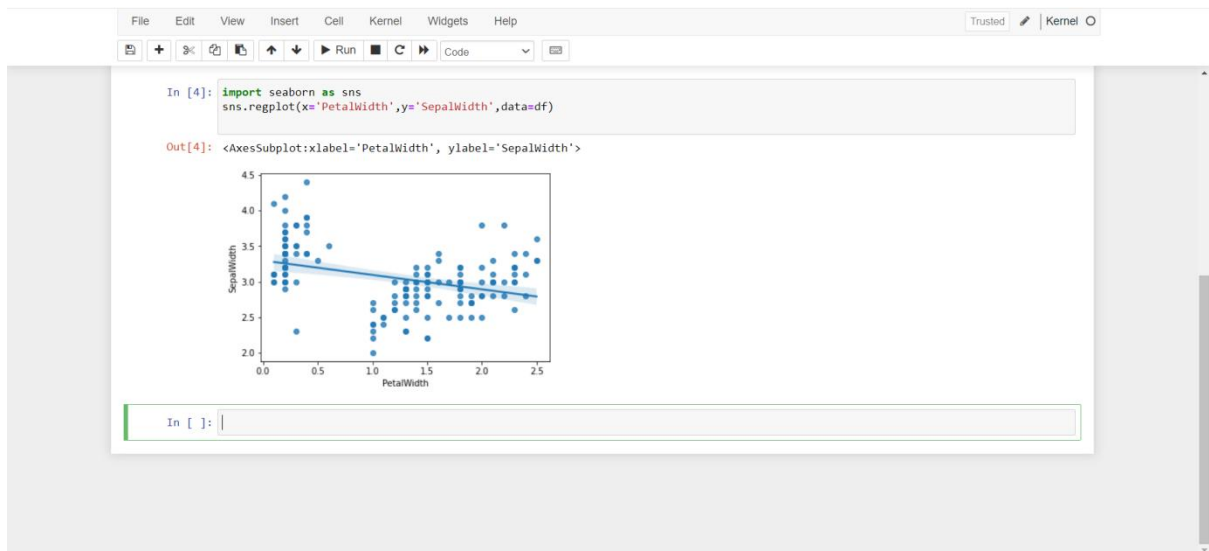
Code (a)

```
import pandas as pd
import numpy as np
df=pd.read_csv('iris-data.csv')
df.plot.bar()
```



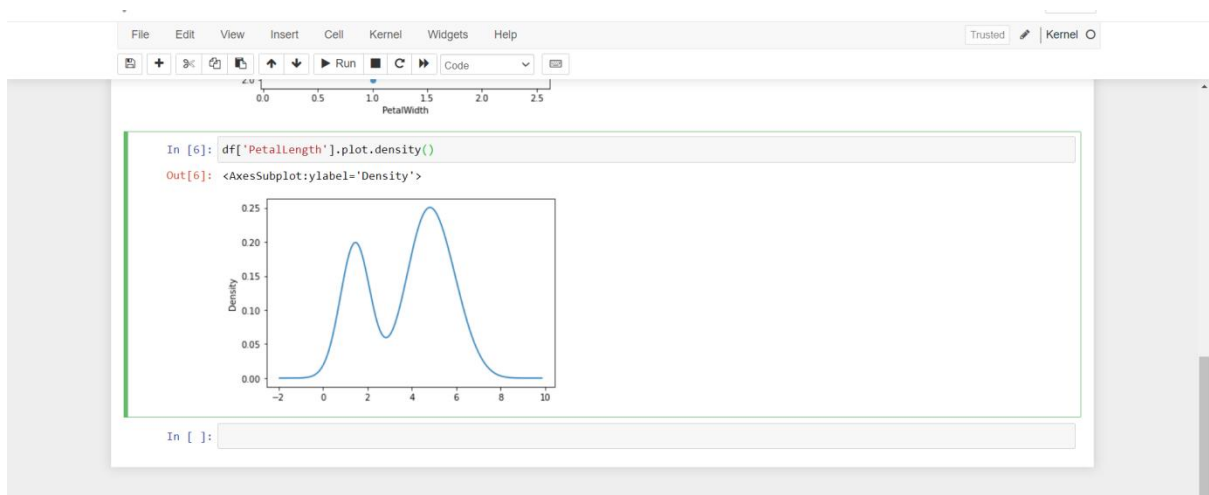
**Code( b):**

```
import seaborn as sns
sns.regplot(x='PetalWidth',y='SepalWidth',data=df)
```



**Code (c):**

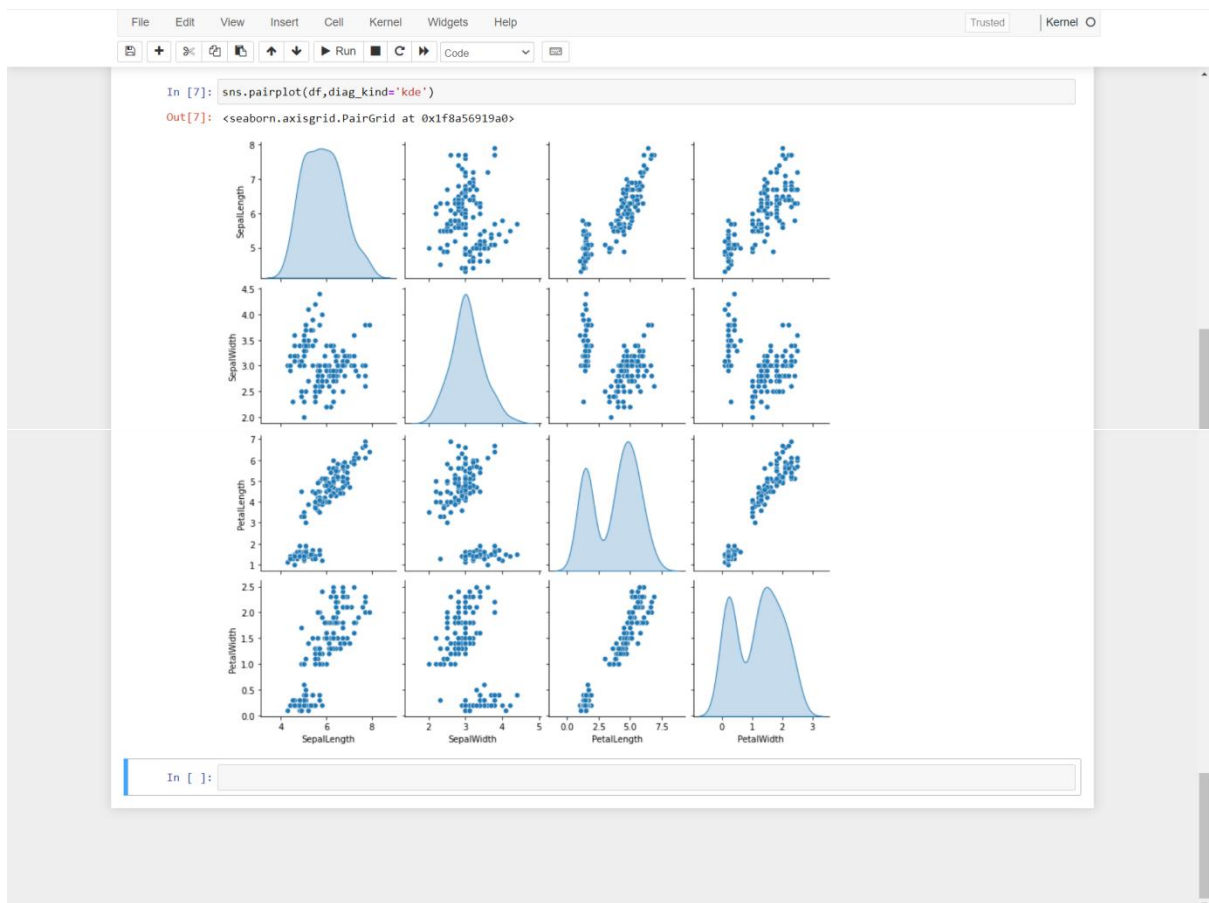
```
df['PetalLength'].plot.density()
```



**Code d):**

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





## 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

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	Formatted Summary	Precip Type	Temperat	Apparent T	Humidity	Wind Spee	Wind Bear	Visibility (K	Loud Cove	Pressure (i	Daily Summary											
2	2006-04-0	Partly Clou,rain	9.472222	7.388889	0.89	14.1197	251	15.8263	0	1015.13	Partly cloudy throughout the day.											
3	2006-04-0	Partly Clou,rain	9.355556	7.227778	0.86	14.2646	259	15.8263	0	1015.63	Partly cloudy throughout the day.											
4	2006-04-0	Mostly Clou,rain	9.377778	9.377778	0.89	3.9284	204	14.9569	0	1015.94	Partly cloudy throughout the day.											
5	2006-04-0	Partly Clou,rain	8.288889	5.944444	0.83	14.1036	269	15.8263	0	1016.41	Partly cloudy throughout the day.											
6	2006-04-0	Mostly Clou,rain	8.755556	6.977778	0.83	11.0446	259	15.8263	0	1016.51	Partly cloudy throughout the day.											
7	2006-04-0	Partly Clou,rain	9.222222	7.111111	0.85	13.9587	258	14.9569	0	1016.66	Partly cloudy throughout the day.											
8	2006-04-0	Partly Clou,rain	7.733333	5.522222	0.95	12.3648	259	9.982	0	1016.72	Partly cloudy throughout the day.											
9	2006-04-0	Partly Clou,rain	8.772222	6.527778	0.89	14.1519	260	9.982	0	1016.84	Partly cloudy throughout the day.											
10	2006-04-0	Partly Clou,rain	10.82222	10.82222	0.82	11.3183	259	9.982	0	1017.37	Partly cloudy throughout the day.											
11	2006-04-0	Partly Clou,rain	13.77222	13.77222	0.72	12.5258	279	9.982	0	1017.22	Partly cloudy throughout the day.											
12	2006-04-0	Partly Clou,rain	16.01667	16.01667	0.67	17.5651	290	11.2056	0	1017.42	Partly cloudy throughout the day.											
13	2006-04-0	Partly Clou,rain	17.14444	17.14444	0.54	19.7869	316	11.4471	0	1017.74	Partly cloudy throughout the day.											
14	2006-04-0	Partly Clou,rain	17.8	17.8	0.55	21.9443	281	11.27	0	1017.59	Partly cloudy throughout the day.											
15	2006-04-0	Partly Clou,rain	17.33333	17.33333	0.51	20.6885	289	11.27	0	1017.48	Partly cloudy throughout the day.											
16	2006-04-0	Partly Clou,rain	18.87778	18.87778	0.47	15.3755	262	11.4471	0	1017.17	Partly cloudy throughout the day.											
17	2006-04-0	Partly Clou,rain	18.91111	18.91111	0.46	10.4006	288	11.27	0	1016.47	Partly cloudy throughout the day.											
18	2006-04-0	Partly Clou,rain	15.38889	15.38889	0.6	14.4095	251	11.27	0	1016.15	Partly cloudy throughout the day.											
19	2006-04-0	Mostly Clou,rain	15.55	15.55	0.63	11.1573	230	11.4471	0	1016.17	Partly cloudy throughout the day.											
20	2006-04-0	Mostly Clou,rain	14.25556	14.25556	0.69	8.5169	163	11.2056	0	1015.82	Partly cloudy throughout the day.											
21	2006-04-0	Mostly Clou,rain	13.14444	13.14444	0.7	7.6314	139	11.2056	0	1015.83	Partly cloudy throughout the day.											
22	2006-04-0	Mostly Clou,rain	11.55	11.55	0.77	7.3899	147	11.0285	0	1015.85	Partly cloudy throughout the day.											
23	2006-04-0	Mostly Clou,rain	11.18333	11.18333	0.76	4.9266	160	9.982	0	1015.77	Partly cloudy throughout the day.											
24	2006-04-0	Partly Clou,rain	10.11667	10.11667	0.79	6.6493	163	15.8263	0	1015.4	Partly cloudy throughout the day.											
25	2006-04-0	Mostly Clou,rain	10.2	10.2	0.77	3.9284	152	14.9569	0	1015.51	Partly cloudy throughout the day.											
26	2006-04-1	Partly Clou,rain	10.42222	10.42222	0.62	16.9855	150	15.8263	0	1014.4	Mostly cloudy throughout the day.											
27	2006-04-1	Partly Clou,rain	9.911111	7.566667	0.66	17.2109	149	15.8263	0	1014.2	Mostly cloudy throughout the day.											
28	2006-04-1	Mostly Clou,rain	11.18333	11.18333	0.8	10.8192	163	14.9569	0	1008.71	Mostly cloudy throughout the day.											
29	2006-04-1	Partly Clou,rain	7.155556	5.044444	0.79	11.0768	180	15.8263	0	1014.47	Mostly cloudy throughout the day.											

## Code a):

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

```
import pandas as pd
```

```
import numpy as np
```

```
df=pd.read_csv("Ques6weatherHistory.csv")
```

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

# OUTPUT

```
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (pykernel)

In [1]: import pandas as pd
import numpy as np
df=pd.read_csv("Ques6weatherHistory.csv")
print(df.groupby('Summary', as_index=False)['Wind Bearing (degrees)'].mean())
```

	Summary	Wind Bearing (degrees)
0	Breezy	233.018519
1	Breezy and Dry	240.000000
2	Breezy and Foggy	160.628571
3	Breezy and Mostly Cloudy	227.639535
4	Breezy and Overcast	213.526515
5	Breezy and Partly Cloudy	259.282383
6	Clear	179.180257
7	Dangerously Windy and Partly Cloudy	307.000000
8	Drizzle	177.307692
9	Dry	230.294118
10	Dry and Mostly Cloudy	187.785714
11	Dry and Partly Cloudy	224.465116
12	Foggy	168.668439
13	Humid and Mostly Cloudy	153.425000
14	Humid and Overcast	138.857143
15	Humid and Partly Cloudy	201.647059
16	Light Rain	180.761905
17	Mostly Cloudy	192.049299
18	Overcast	183.532747
19	Partly Cloudy	190.161094
20	Rain	211.800000
21	Windy	319.750000
22	Windy and Dry	150.000000
23	Windy and Foggy	155.000000
24	Windy and Mostly Cloudy	261.428571
25	Windy and Overcast	244.311111
26	Windy and Partly Cloudy	295.119403

```
In [ ]:
```

## Code b):

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

```
print(new_df2)
```

```
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (pykernel)

In [5]: new_df2= df.ffill().reset_index()
print(new_df2)
```

	index	Formatted Date	Summary	Precip	Type \
0	0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	
1	1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	
2	2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	
3	3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	
4	4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	
...	...	...	...	...	...
96448	96448	2016-09-09 19:00:00.000 +0200	Partly Cloudy	rain	
96449	96449	2016-09-09 20:00:00.000 +0200	Partly Cloudy	rain	
96450	96450	2016-09-09 21:00:00.000 +0200	Partly Cloudy	rain	
96451	96451	2016-09-09 22:00:00.000 +0200	Partly Cloudy	rain	
96452	96452	2016-09-09 23:00:00.000 +0200	Partly Cloudy	rain	
...	...	...	...	...	...
0		9.472222	7.388889	0.89	14.1197
1		9.355556	7.227778	0.86	14.2646
2		9.377778	9.377778	0.89	3.9284
3		8.288889	5.944444	0.83	14.1036
4		8.755556	6.977778	0.83	11.0446
...	...	...	...	...	...
96448		26.016667	26.016667	0.43	10.9963
96449		24.583333	24.583333	0.48	10.0947
96450		22.038889	22.038889	0.56	8.9838
96451		21.522222	21.522222	0.60	10.5294
96452		20.438889	20.438889	0.61	5.8765
...	...	...	...	...	...
0		Wind Bearing (degrees)	Visibility (km)	Loud Cover	\
1		251.0	15.8263	0.0	
2		259.0	15.8263	0.0	
		204.0	14.9569	0.0	

**Code c):**

```
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]))
```

```

26 Windy and Partly Cloudy 295.119403

In [2]: 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 taken 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 01:00:00.000 +0200 <class 'str'>
New Format :- 2006-04-01 01:00:00 <class 'datetime.datetime'>

In [ ]:

```

**Code d):**

```

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

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

```

```

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

In [3]: 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 -0.483333 -4.150000 1.00
rain 0.0 9.472222 7.388889 0.89

Precip Type Loud Cover Wind Speed (km/h) Wind Bearing (degrees) \
snow 0.0 11.0929 219.0
rain 0.0 14.1197 251.0

Precip Type Loud Cover Visibility (km) Pressure (millibars)
snow 0.0 0.4830 1031.56
rain 0.0 15.8263 1015.13

In [ ]:

```

**Code e):**

```

edges=[0.0,0.3,0.6,1.0]

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

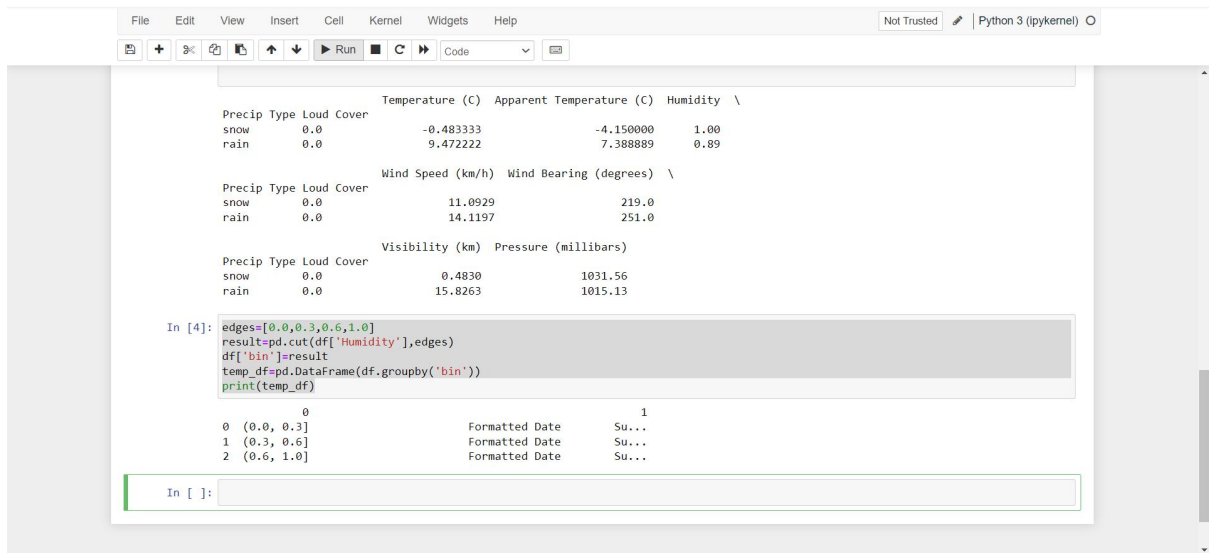
```

```
df['bin']=result
```

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

```
print(temp_df)
```

## OUTPUT



```
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)
```

```
Precip Type Cloud Cover Temperature (C) Apparent Temperature (C) Humidity \
snow 0.0 -0.483333 -4.150000 1.00
rain 0.0 9.472222 7.388889 0.89

Precip Type Cloud Cover Wind Speed (km/h) Wind Bearing (degrees) \
snow 0.0 11.0929 219.0
rain 0.0 14.1197 251.0

Precip Type Cloud Cover Visibility (km) Pressure (millibars)
snow 0.0 0.4830 1031.56
rain 0.0 15.8263 1015.13

In [4]: 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 Su... 1
1 (0.3, 0.6] Formatted Date Su...
2 (0.6, 1.0] Formatted Date Su...
```

```
In [ ]:
```

## 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.

	Name	Birth_Month	Gender	Pass_Division
1	Mudit Chauhan	December	M	III
2	Seema Chopra	January	F	II
3	Rani Gupta	March	F	I
4	Aditya Narayan	October	M	I
5	Sanjeev Sahni	February	M	II
6	Prakash Kumar	December	M	III
7	Ritu Agarwal	September	F	I
8	Akshay Goel	August	M	I
9	Meeta Kulkarni	July	F	II
10	Preeti Ahuja	November	F	II
11	Sunil Das Gupta	April	M	III
12	Sonali Sapre	January	F	I
13	Rashmi Talwar	June	F	III
14	Ashish Dubey	May	M	II
15	Kiran Sharma	February	F	II
16	Sameer Bansal	October	M	I
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				

### Code a):

```
import pandas as pd

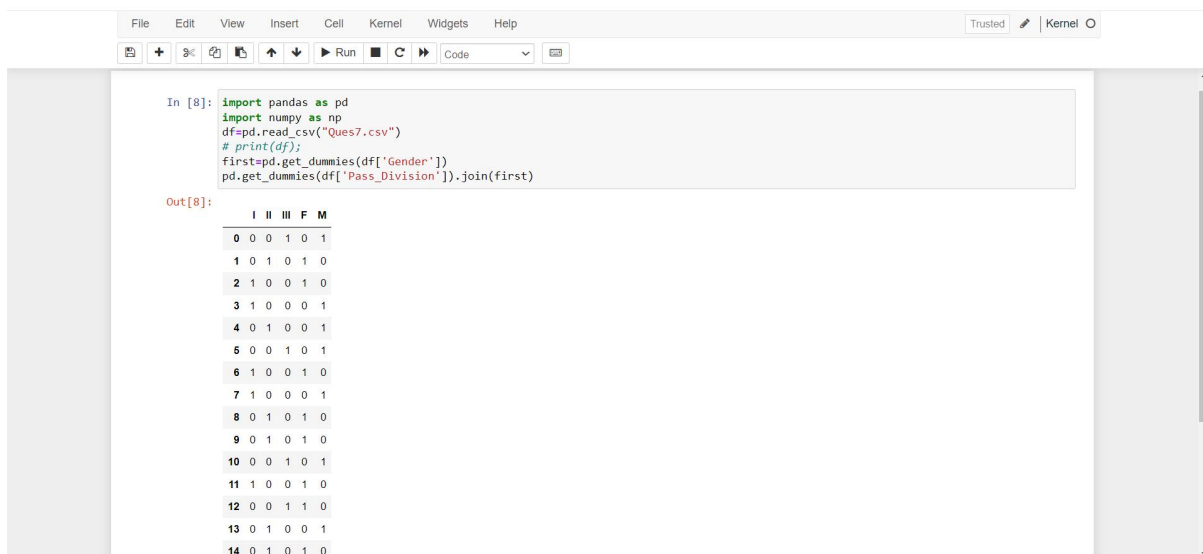
import numpy as np

df=pd.read_csv("Ques7.csv")

# print(df);

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

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



```
In [8]: import pandas as pd
import numpy as np
df=pd.read_csv("Ques7.csv")
# print(df);
first=pd.get_dummies(df['Gender'])
pd.get_dummies(df['Pass_Division']).join(first)
```

Out[8]:

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

### Code b):

```
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')
```



FileEditViewInsertCellWidgetsHelp

TrustedKernel

+

⌕

📄

📁

⬆️

⬇️

▶️

Run

⏮️

⏪

Code

⌵

🗨️

In [9]:

```
myorderis=pd.CategoricalDtype(['January','February','March','April','May','June','July','August','September','October','November'])
df['Birth_Month']=df['Birth_Month'].astype(myorderis)
df.sort_values(by='Birth_Month')
```

Out[9]:

	Name	Birth_Month	Gender	Pass_Division
1	Seema Chopra	January	F	II
11	Sonali Sapre	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 Bansl	October	M	I
9	Preeti Ahuja	November	F	II
0	Mudit Chauhan	December	M	III
5	Prakash Kumar	December	M	III

### **Ques 8.**

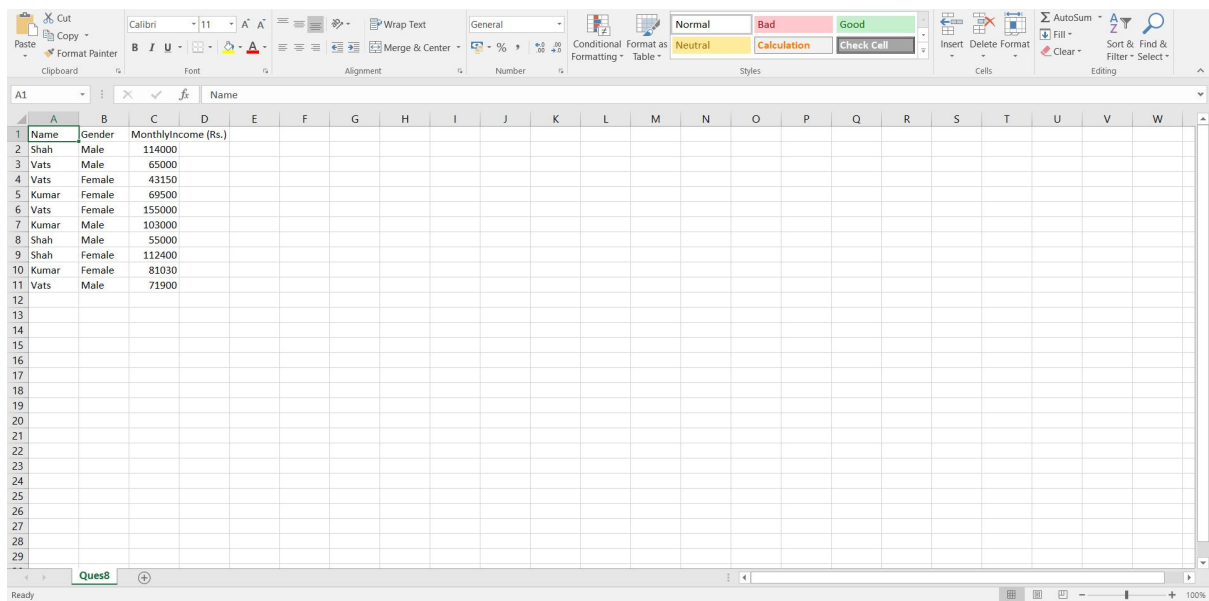
Consider the following data frame containing a family name, gender of the family member and her/his monthly income in each record.

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

## OUTPUT



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	Name	Gender	MonthlyIncome (Rs.)																				
2	Shah	Male	114000																				
3	Vats	Male	65000																				
4	Vats	Female	43150																				
5	Kumar	Female	69500																				
6	Vats	Female	155000																				
7	Kumar	Male	103000																				
8	Shah	Male	55000																				
9	Shah	Female	112400																				
10	Kumar	Female	81030																				
11	Vats	Male	71900																				
12																							
13																							
14																							
15																							
16																							
17																							
18																							
19																							
20																							
21																							
22																							
23																							
24																							
25																							
26																							
27																							
28																							
29																							

### Code a):

```
import pandas as pd

import numpy as np

df=pd.read_csv("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)):
```

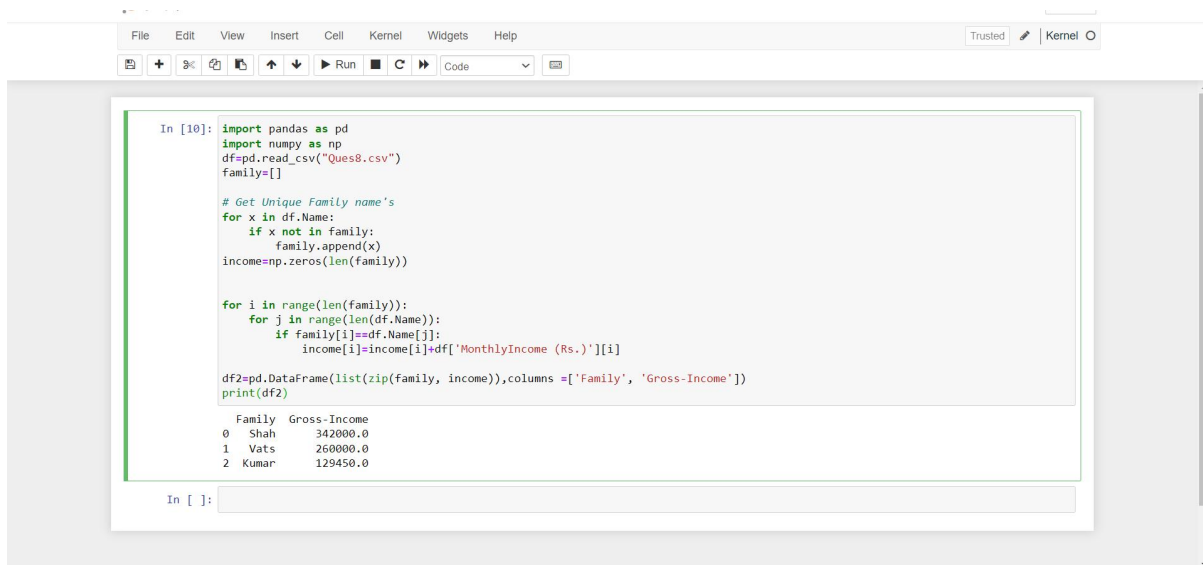
```
for j in range(len(df.Name)):
```

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

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

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

```
print(df2)
```



The screenshot shows a Jupyter Notebook interface. The top bar includes menus for File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Below the menus is a toolbar with icons for adding, deleting, and running code cells. The main area contains a code cell with the following Python code:

```
In [10]: import pandas as pd
import numpy as np
df=pd.read_csv("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'. The data is as follows:

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

Below the code cell, there is an input prompt 'In [ ]:' followed by a text box for entering the next command.

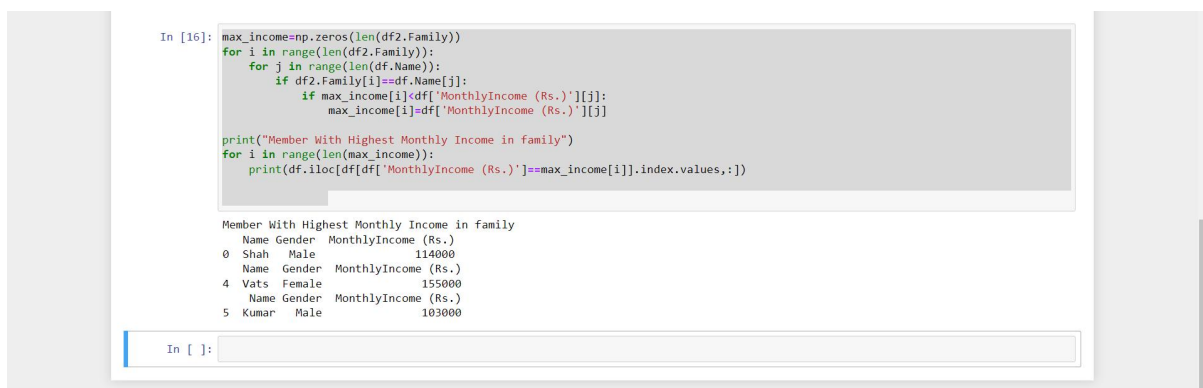
### Code b):

```
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,:])
```



```
In [16]: 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
4  Vats  Female        155000
5  Kumar  Male         103000
```

### Code c):

```
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])
```

```
File Edit View Insert Cell Kernel Widgets Help Trusted Kernel O
+ %< [Run] [Code]
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 [17]: 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 [ ]:
```

## Code d):

```
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))
```

```
+ %< [Run] [Code]

In [17]: 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 [18]: 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 [ ]:
```