INDEX

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

	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.

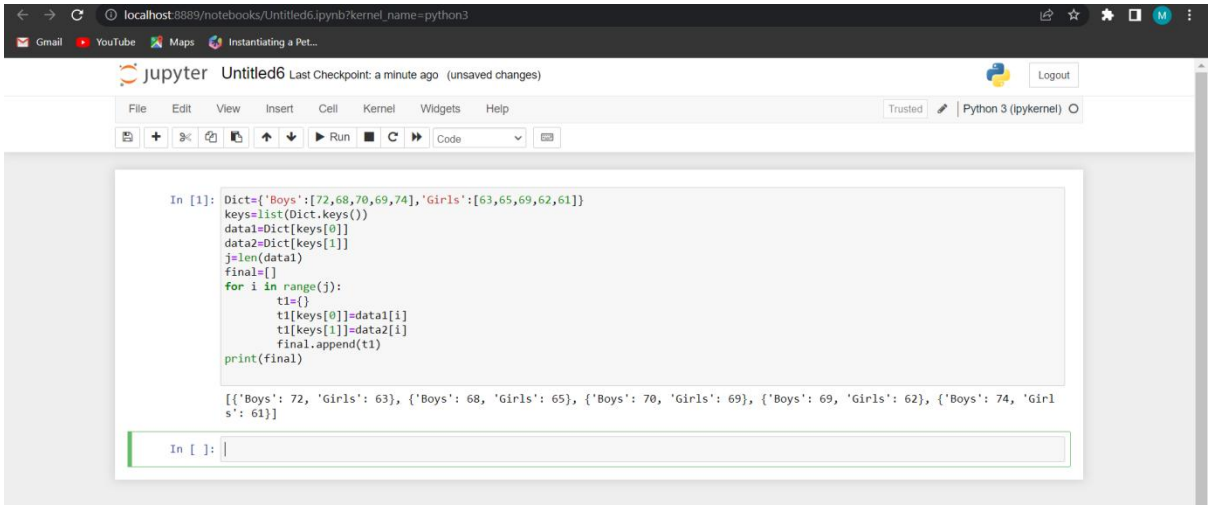
QUESTION1: Given below is a dictionary having two keys 'Boys' and 'Girls' and having two lists of heights of five Boys and Five Girls respectively as values associated with these keys
Original dictionary of lists: {'Boys': [72, 68, 70, 69, 74], 'Girls': [63, 65, 69, 62, 61]}
From the given dictionary 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)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykernel)
Run
Code
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.

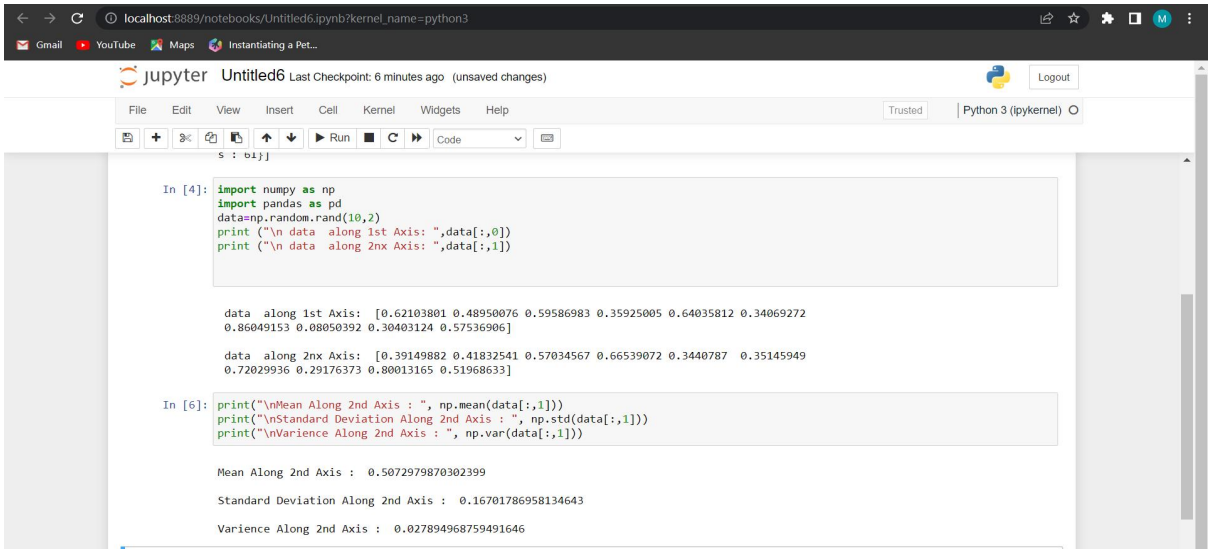
d) Test whether the elements of a given array are zero, non-zero and NaN. Record the indices of these elements in three separate arrays.

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



(b)

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

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

OUTPUT

(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]))

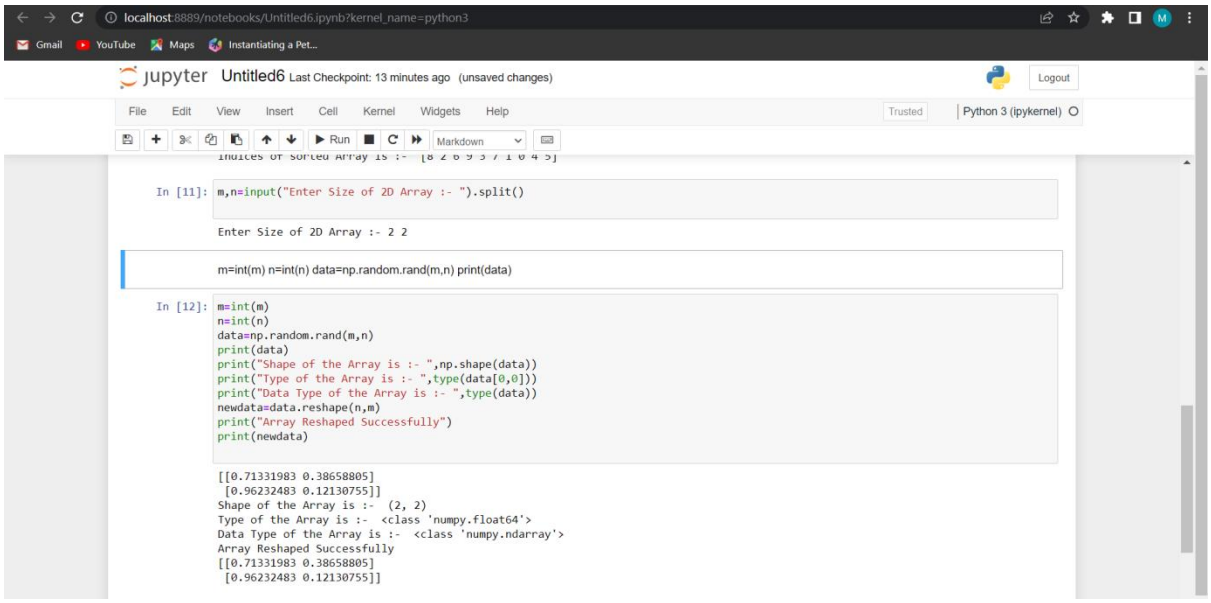
print("Data Type of the Array is :- ",type(data))

newdata=data.reshape(n,m)

print("Array Reshaped Successfully")

print(newdata)

OUTPUT



(d)

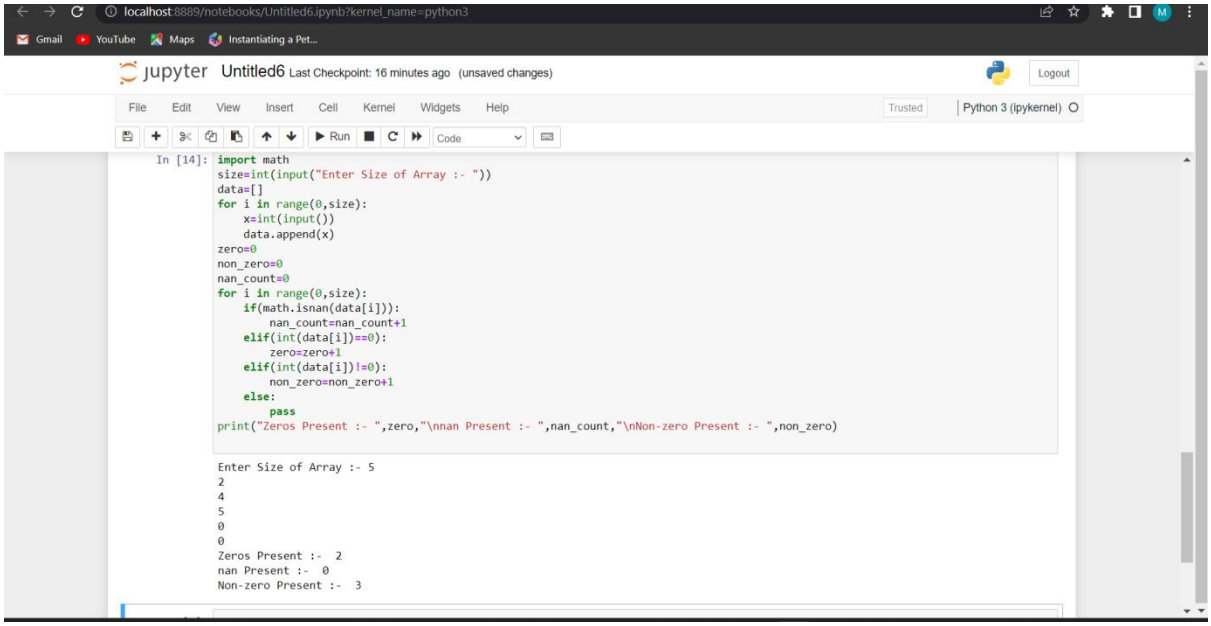
import math

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)
Run
Code
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 2nd & 3rd 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,),1]=float("nan")
count_nan=0
for i in range(0,len(df.columns)):
```

```
    for j in range(0,len(df)):
        if(math.isnan(df.iat[j,i])):
            count_nan=count_nan+1
print("Number  of Missing Values :- ",count_nan)
```

OUTPUT

```
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)
```

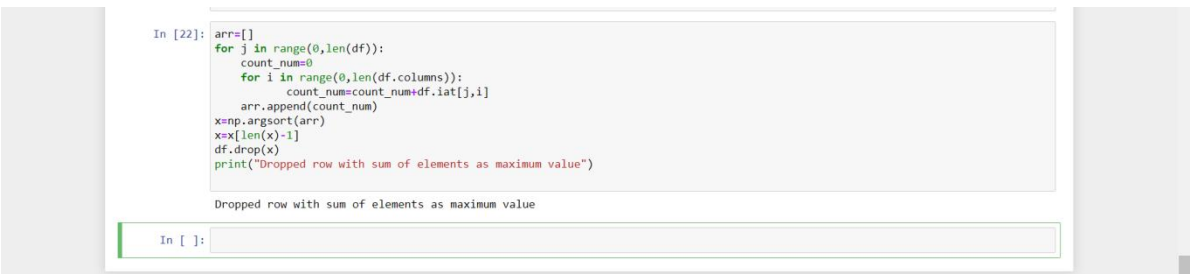
```
    print("Dropped Column ",i)
```

OUTPUT

(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")
```

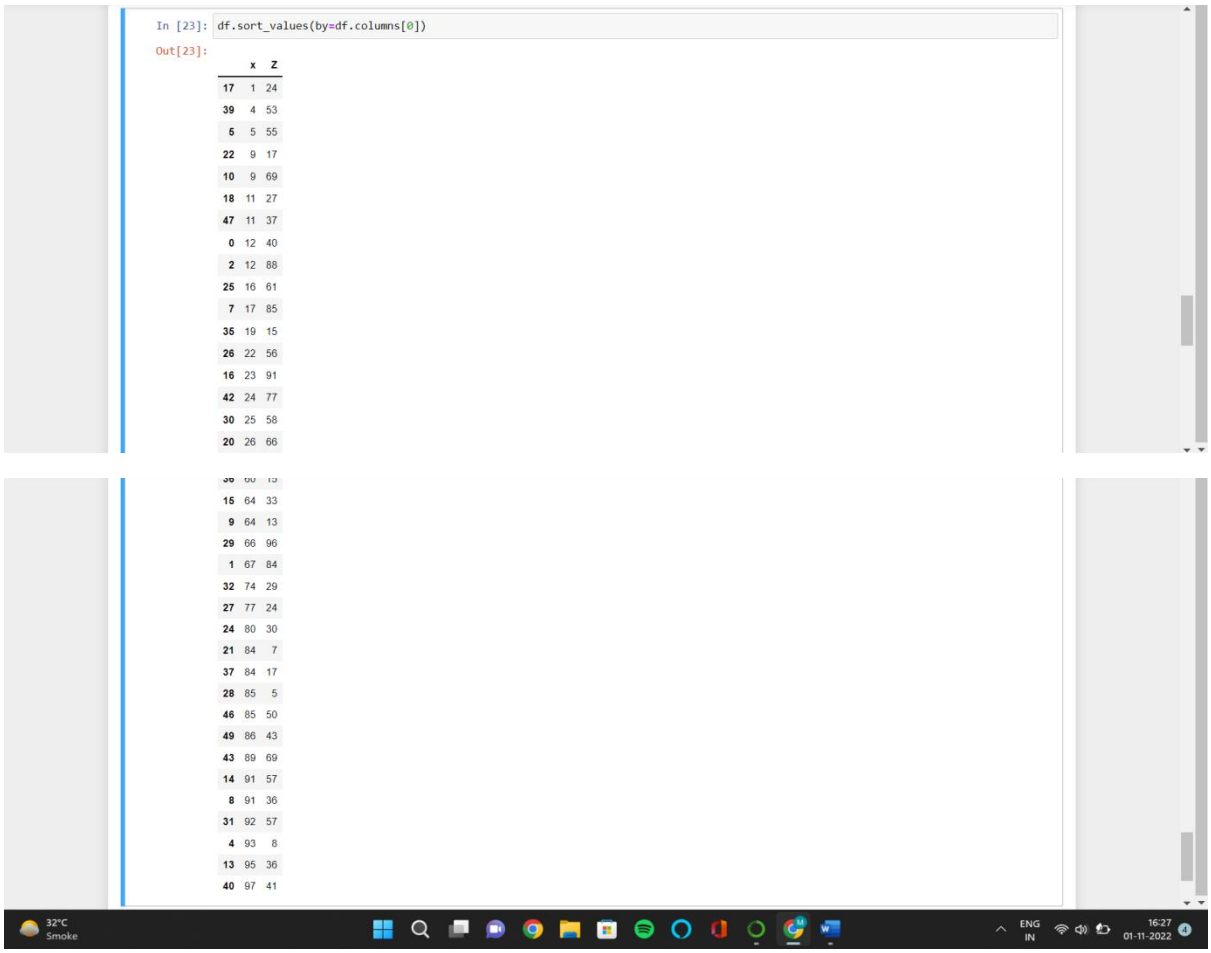
OUTPUT



(d)

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

OUTPUT



(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)

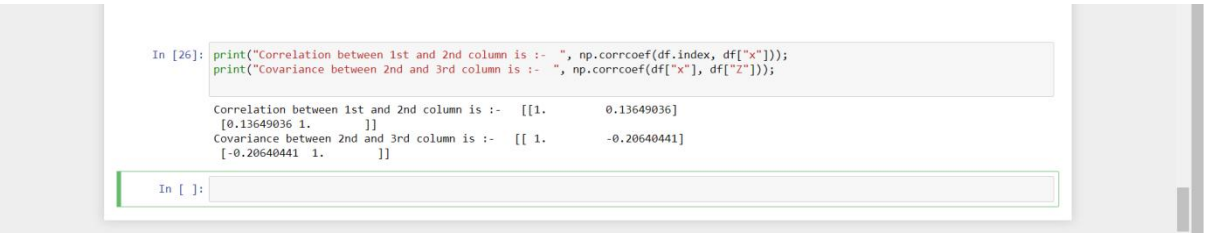
print("Dropped Column ",i)

(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



(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])

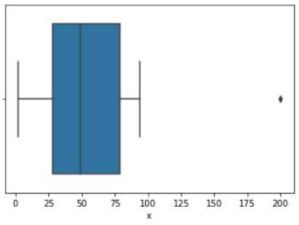
OUTPUT

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]
```

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.

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					

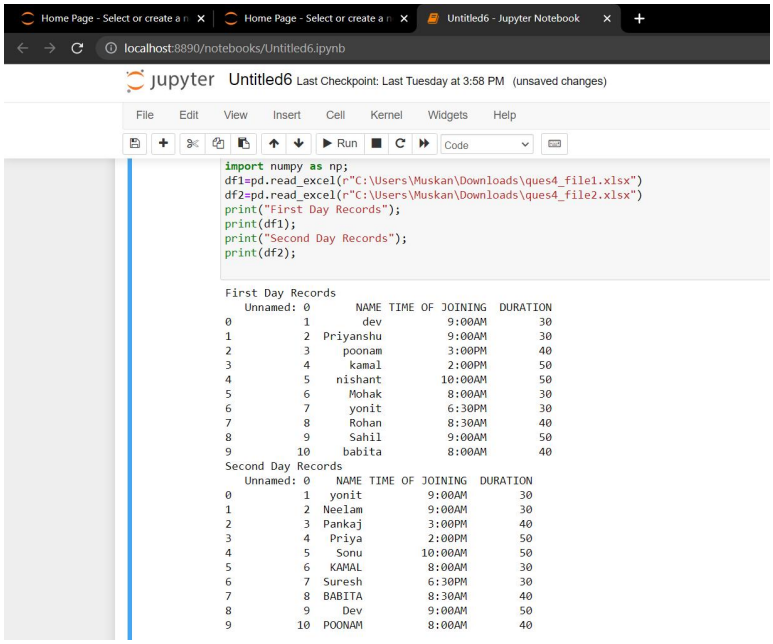
File2

	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

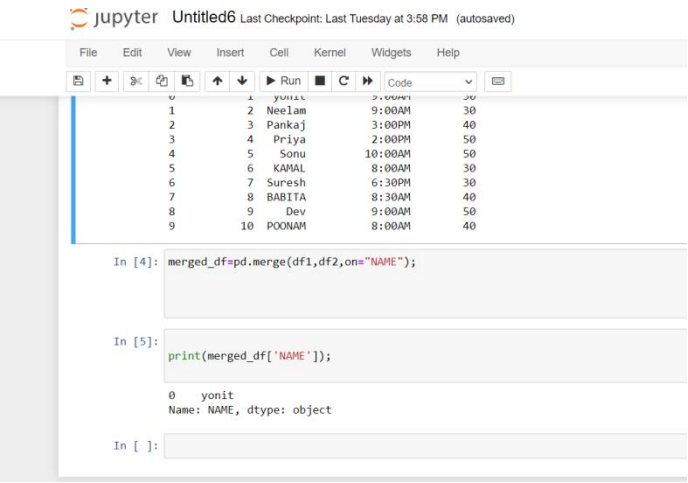
```
import pandas as pd;
import numpy as np;
df1=pd.read_excel(r"C:\Users\Muskan\Downloads\ques4_file1.xlsx")
df2=pd.read_excel(r"C:\Users\Muskan\Downloads\ques4_file2.xlsx")
print("First Day Records");
print(df1);
print("Second Day Records");
print(df2);
```

OUTPUT



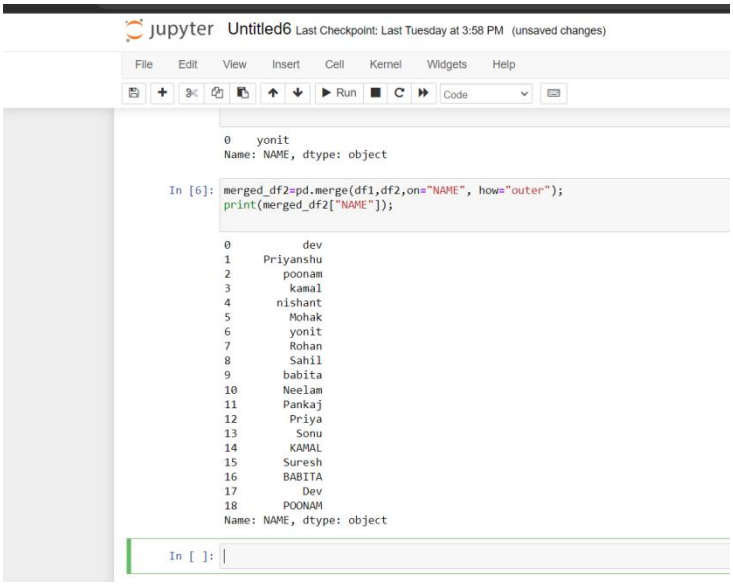
(a)

```
merged_df=pd.merge(df1,df2,on="Name");
print(merged_df['Name']);
```



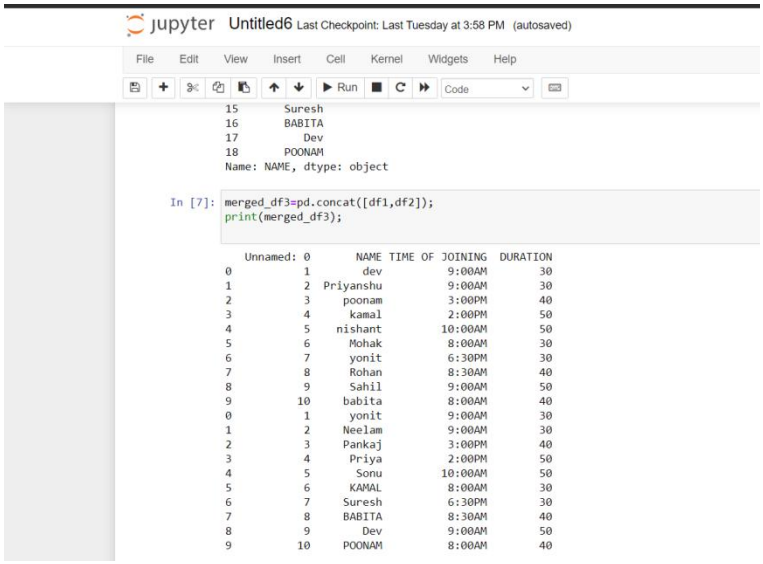
(b) merged_df2=pd.merge(df1,df2,on="NAME", how="outer");

print(merged_df2["NAME"]);



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

print(merged_df3);

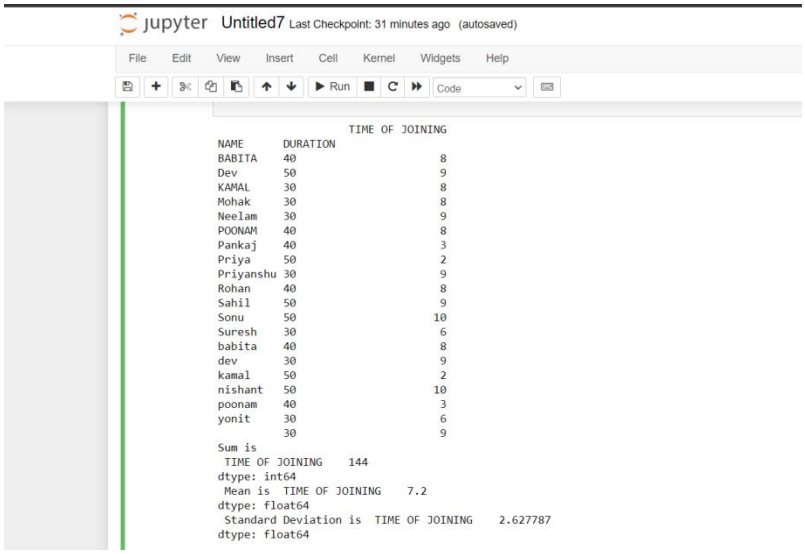


(d)

```
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),);



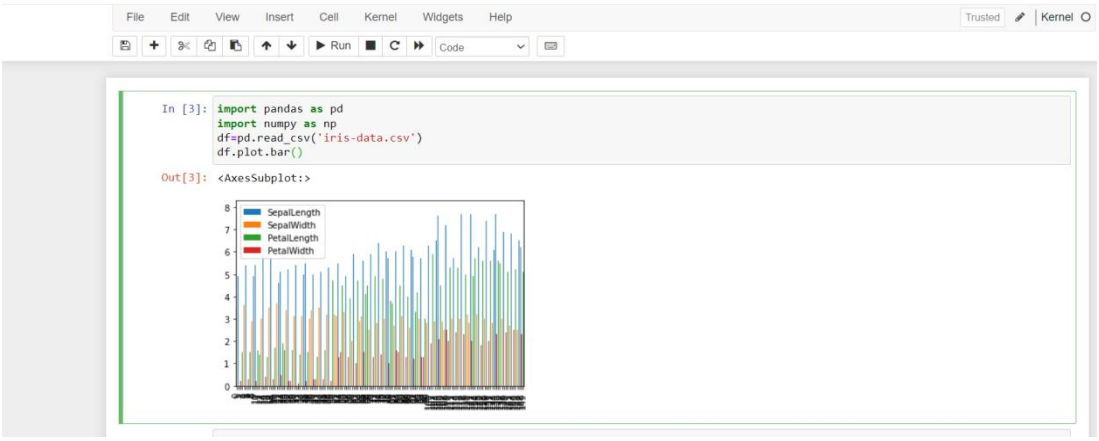
Ques 5.

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.

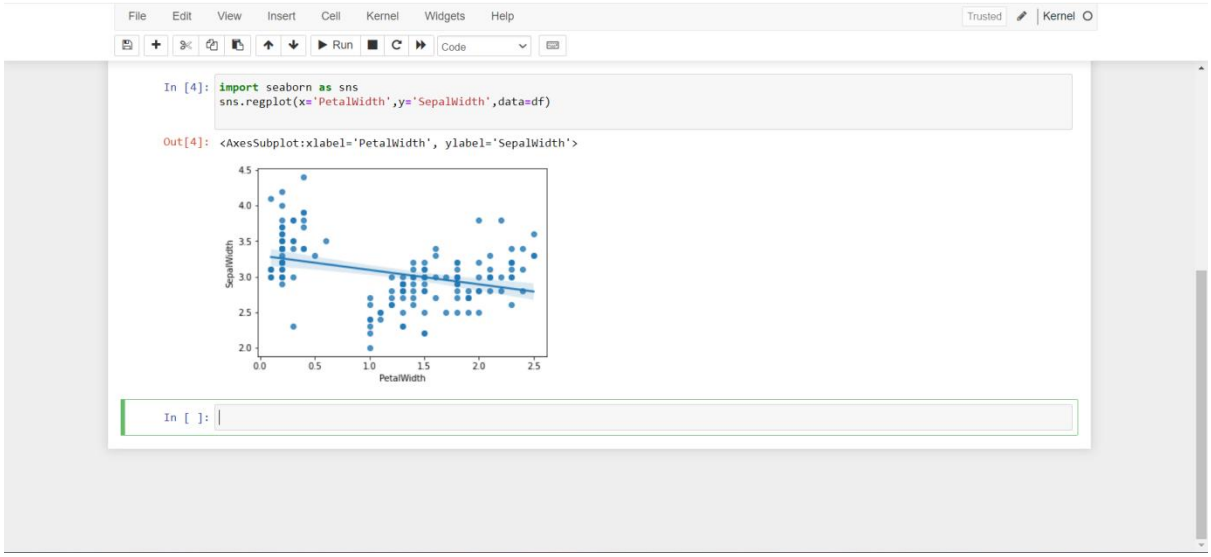
Code (a)

```
import pandas as pd

import numpy as np

df=pd.read_csv('iris-data.csv')

df.plot.bar()
```



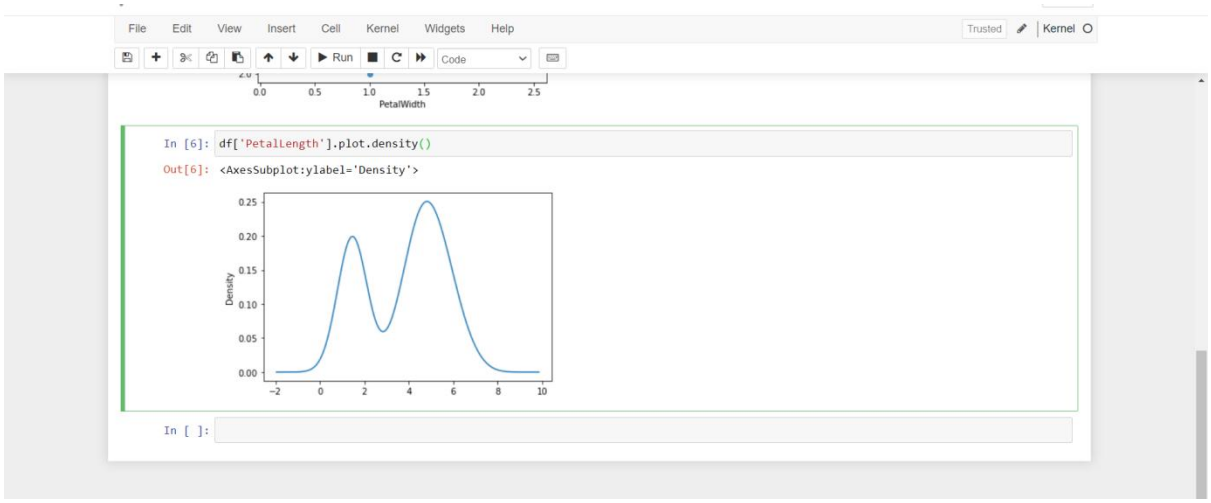
Code(b):

```
import seaborn as sns

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



Code (c):

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



Code d):

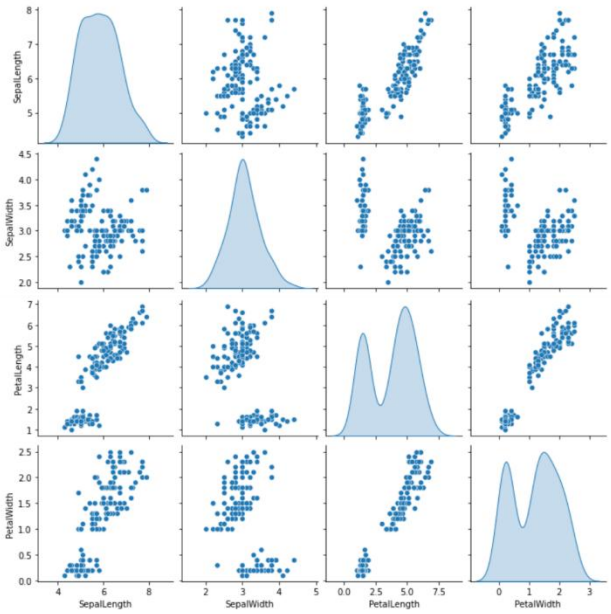
sns.pairplot(df,diag_kind='kde')

File Edit View Insert Cell Kernel Widgets Help Trusted Kernel

Run Code

In [7]:
```
sns.pairplot(df,diag_kind='kde')
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x1f8a56919a0>



In [ ]:

Ques 6.

Consider any sales training/ weather forecasting dataset

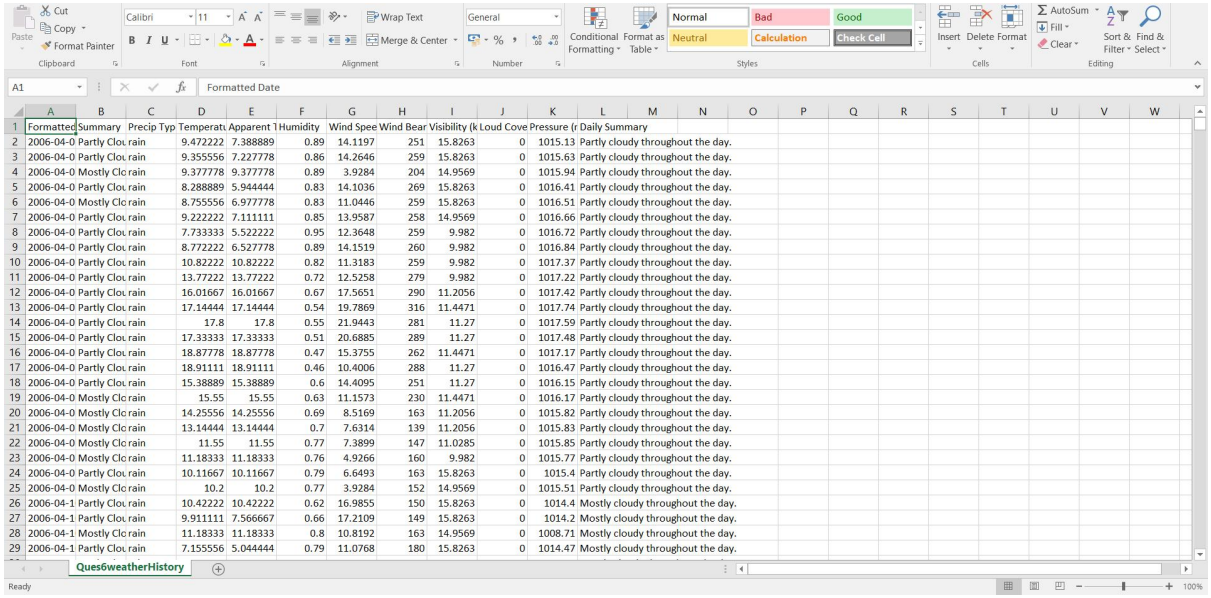
a. Compute mean of a series grouped by another series

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



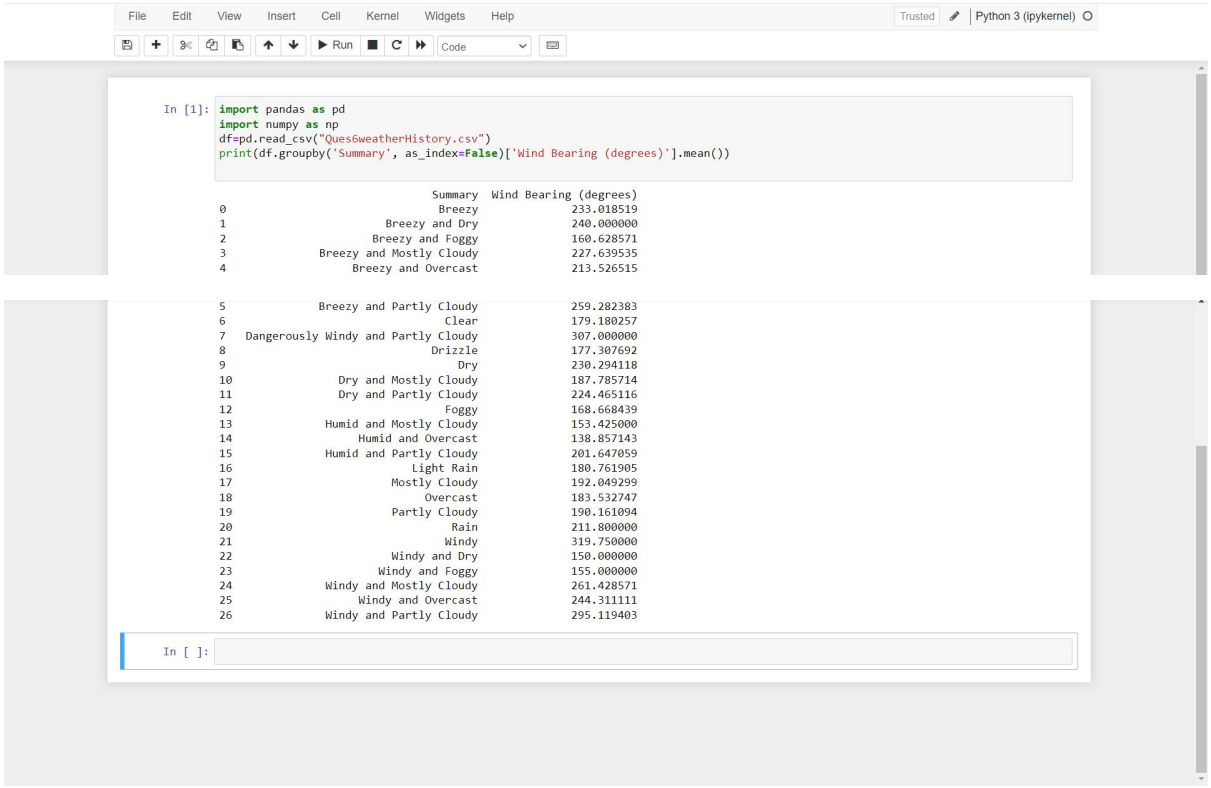
Formatted	Summary	Precip Typ	Temperati	Apparent T	Humidity	Wind Spee	Wind Bear	Visibility (k	Loud Cove	Pressure (r	Daily Summary
2006-04-0	Partly Clou	rain	9.472222	7.388889	0.89	14.1197	251	15.8263	0	1015.13	Partly cloudy throughout the day.
2006-04-0	Partly Clou	rain	9.355556	7.227778	0.86	14.2646	259	15.8263	0	1015.63	Partly cloudy throughout the day.
2006-04-0	Mostly Clo	rain	9.377778	9.377778	0.89	3.9284	204	14.9569	0	1015.94	Partly cloudy throughout the day.
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.
2006-04-0	Mostly Clo	rain	8.755556	6.977778	0.83	11.0446	259	15.8263	0	1016.51	Partly cloudy throughout the day.
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.
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.
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.
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.
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.
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.
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.
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.
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.
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.
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.
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.
2006-04-0	Mostly Clo	rain	15.55	15.55	0.63	11.1573	230	11.4471	0	1016.17	Partly cloudy throughout the day.
2006-04-0	Mostly Clo	rain	14.25556	14.25556	0.69	8.5169	163	11.2056	0	1015.82	Partly cloudy throughout the day.
2006-04-0	Mostly Clo	rain	13.14444	13.14444	0.7	7.6314	139	11.2056	0	1015.83	Partly cloudy throughout the day.
2006-04-0	Mostly Clo	rain	11.55	11.55	0.77	7.3899	147	11.0285	0	1015.85	Partly cloudy throughout the day.
2006-04-0	Mostly Clo	rain	11.18333	11.18333	0.76	4.9266	160	9.982	0	1015.77	Partly cloudy throughout the day.
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.
2006-04-0	Mostly Clo	rain	10.2	10.2	0.77	3.9284	152	14.9569	0	1015.51	Partly cloudy throughout the day.
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.
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.
2006-04-1	Mostly Clo	rain	11.18333	11.18333	0.8	10.8192	163	14.9569	0	1008.71	Mostly cloudy throughout the day.
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



```
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())
```

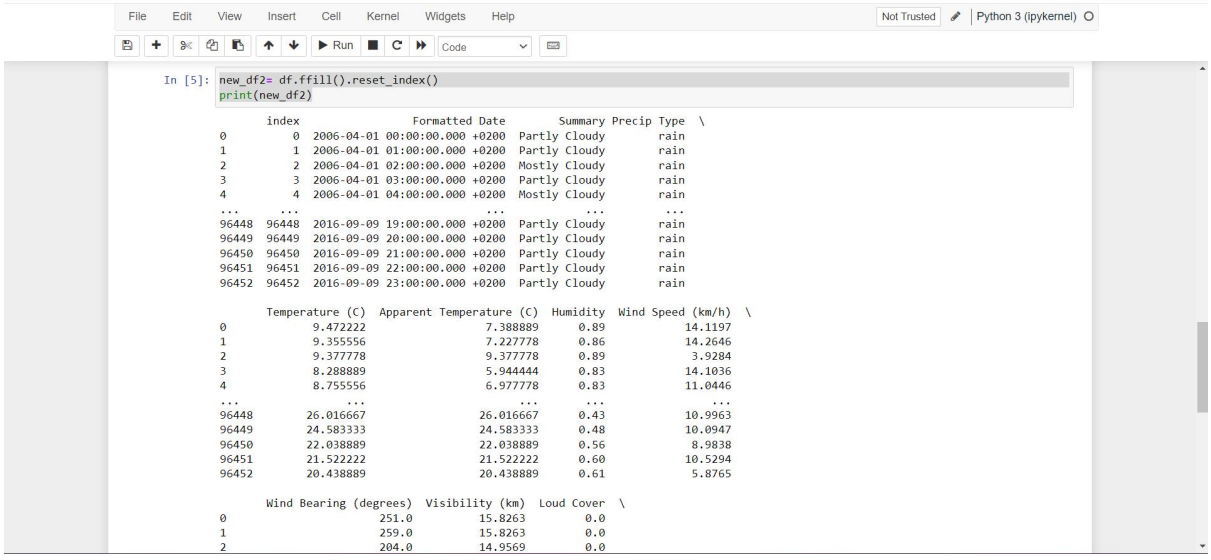
```
                                Summary  Wind Bearing (degrees)
0                                Breezy              233.018519
1                          Breezy and Dry              240.000000
2                        Breezy and Foggy              160.628571
3                 Breezy and Mostly Cloudy              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)

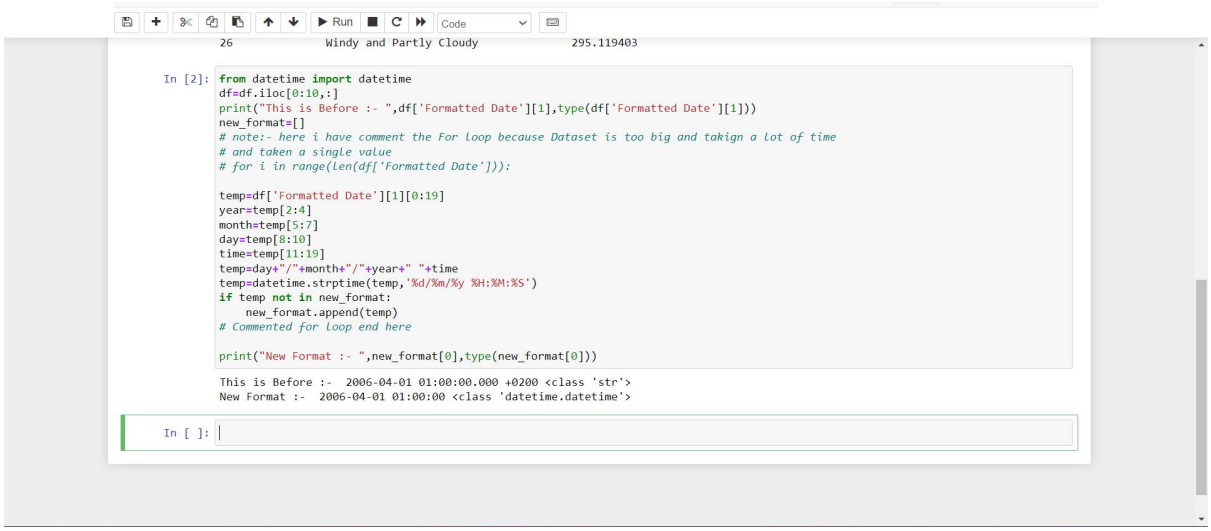


```
In [5]: new_df2= df.ffill().reset_index()
        print(new_df2)
```

```
       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

       Temperature (C)  Apparent Temperature (C)  Humidity  Wind Speed (km/h)  \
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

       Wind Bearing (degrees)  Visibility (km)  Loud Cover  \
0                       251.0          15.8263         0.0
1                       259.0          15.8263         0.0
2                       204.0          14.9569         0.0
```

Code c):

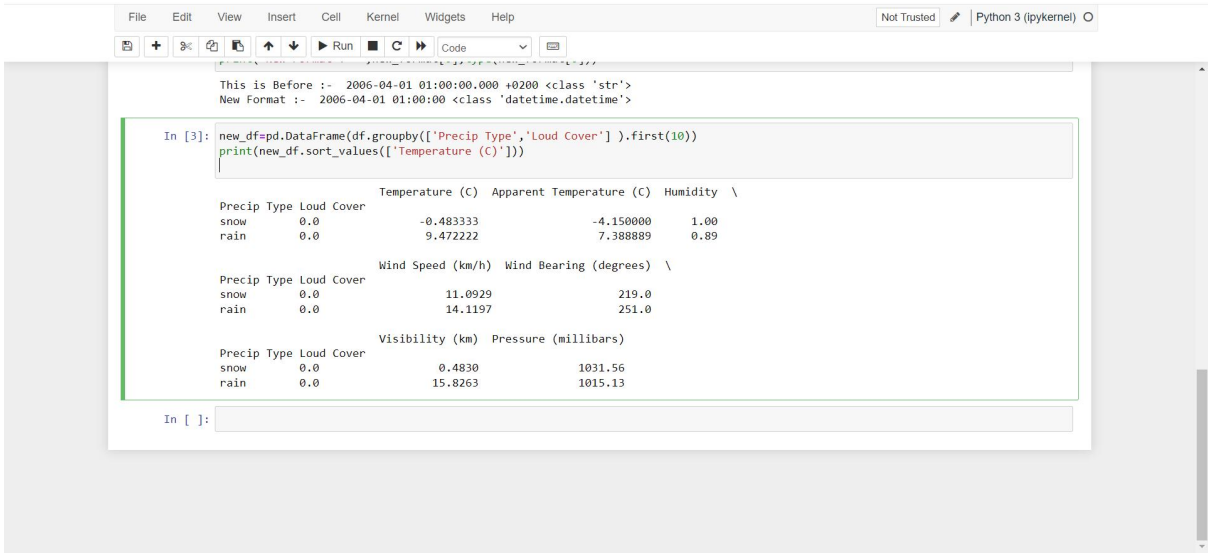
```
from datetime import datetime
df=df.iloc[0:10,:]
print("This is Before :- ",df['Formatted Date'][1],type(df['Formatted Date'][1]))
new_format=[]
# note:- here i have comment the For loop because Dataset is too big and takign a lot of time
# and taken a single value
# for i in range(len(df['Formatted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

temp=df['Formatted Date'][1][0:19]
year=temp[2:4]
month=temp[5:7]
day=temp[8:10]
time=temp[11:19]
temp=day+"/"+month+"/"+year+" "+time
temp=datetime.strptime(temp,'%d/%m/%y %H:%M:%S')
if temp not in new_format:
    new_format.append(temp)
# Commented for loop end here

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

This is Before :-  2006-04-01 01:00:00.000 +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)

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 [3]:
new_df=pd.DataFrame(df.groupby(['Precip Type','Loud Cover'] ).first(10))
print(new_df.sort_values(['Temperature (C)']))

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

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

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

In [ ]:
```

Code e):

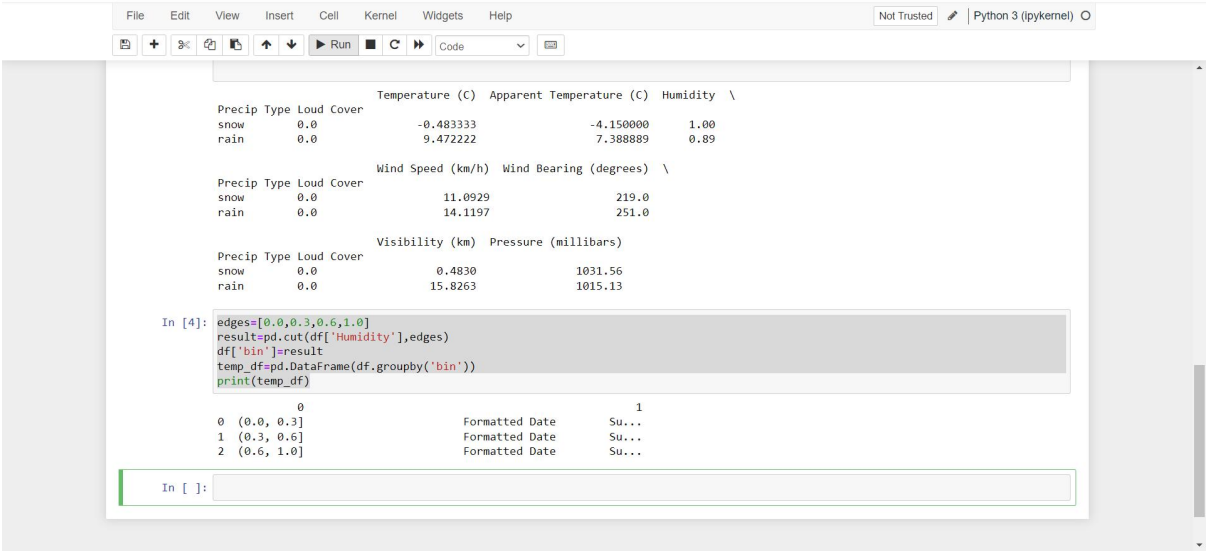
```
edges=[0.0,0.3,0.6,1.0]
result=pd.cut(df['Humidity'],edges)
```

df['bin']=result

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

print(temp_df)

OUTPUT



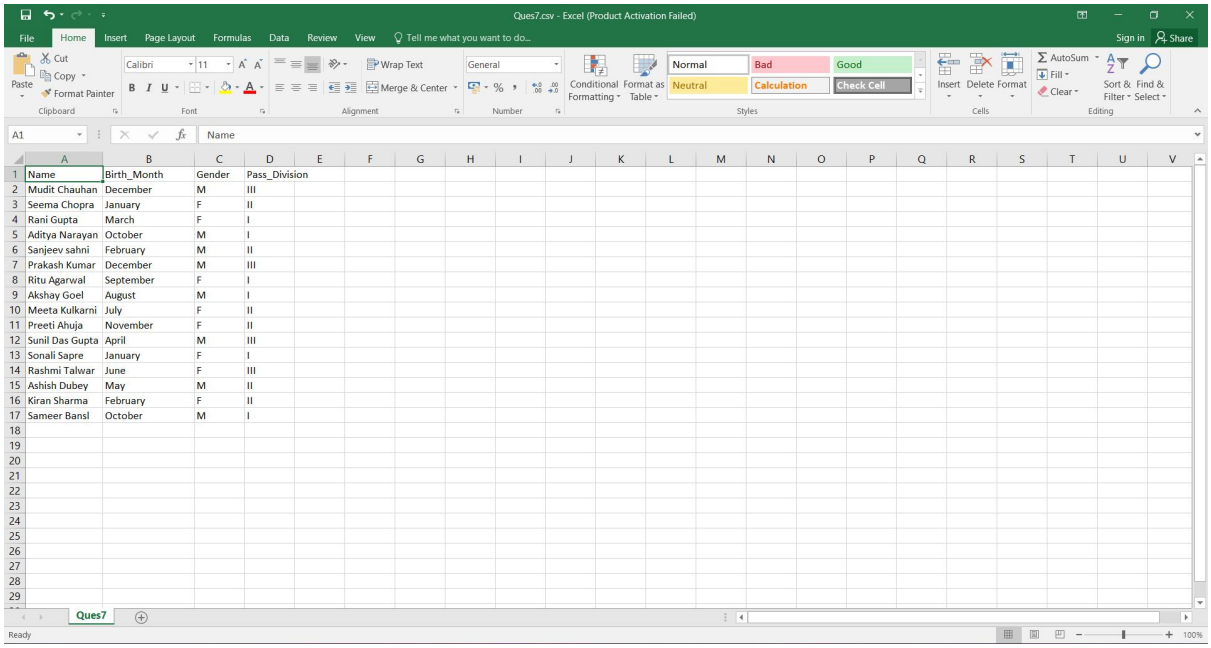
Ques 7.

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

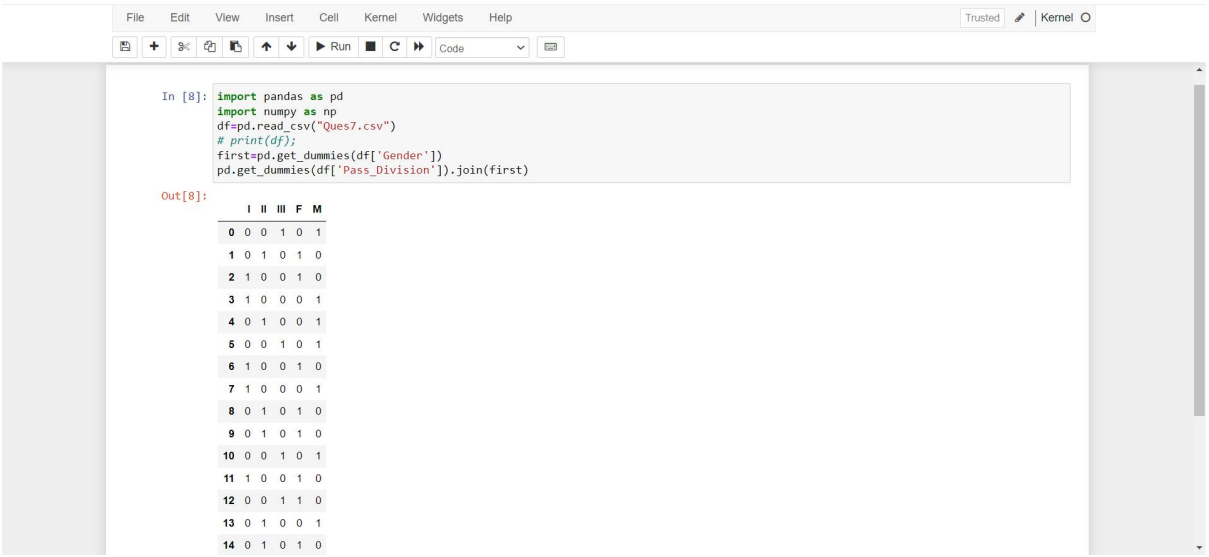
a. Perform one hot encoding of the last two columns of categorical data using the get_dummies() function.

b. Sort this data frame on the "Birth Month" column (i.e. January to December). Hint: Convert Month to Categorical.



Code 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')
```

File Edit View Insert Cell Kernel Widgets Help Trusted Kernel

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:

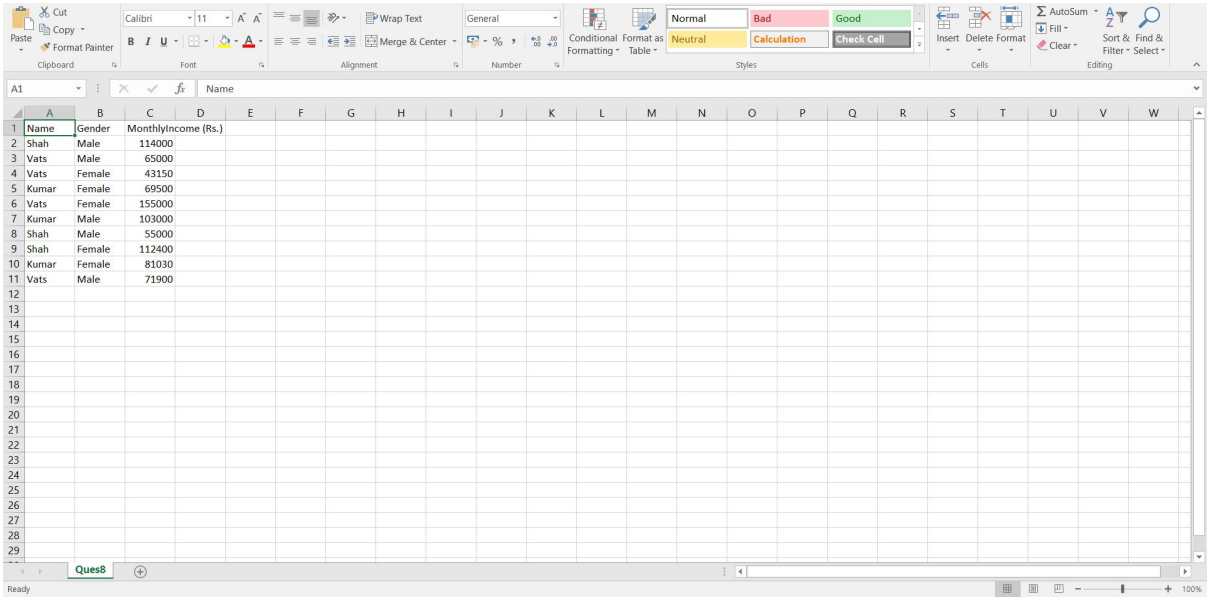
a. Calculate and display familywise gross monthly income.

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

OUTPUT

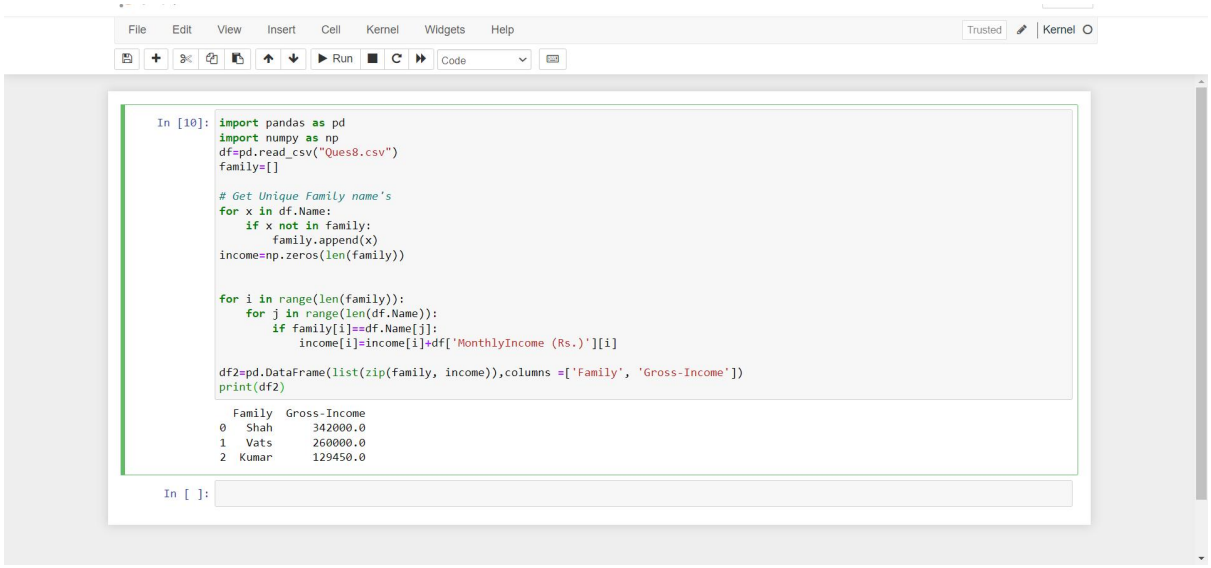


Name	Gender	MonthlyIncome (Rs.)
Shah	Male	114000
Vats	Male	65000
Vats	Female	43150
Kumar	Female	69500
Vats	Female	155000
Kumar	Male	103000
Shah	Male	55000
Shah	Female	112400
Kumar	Female	81030
Vats	Male	71900

Code a):

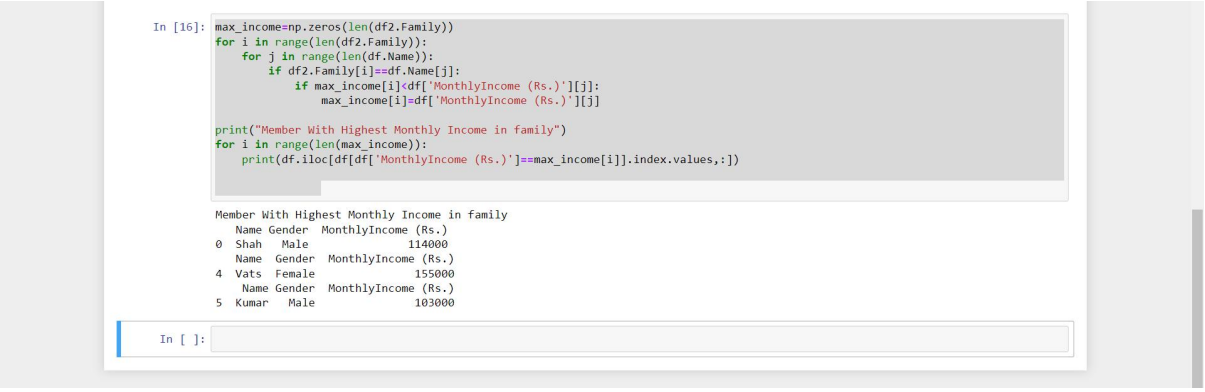
```
import pandas as pd
import numpy as np
df=pd.read_csv("Ques8.csv")
family=[]

# 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)
```



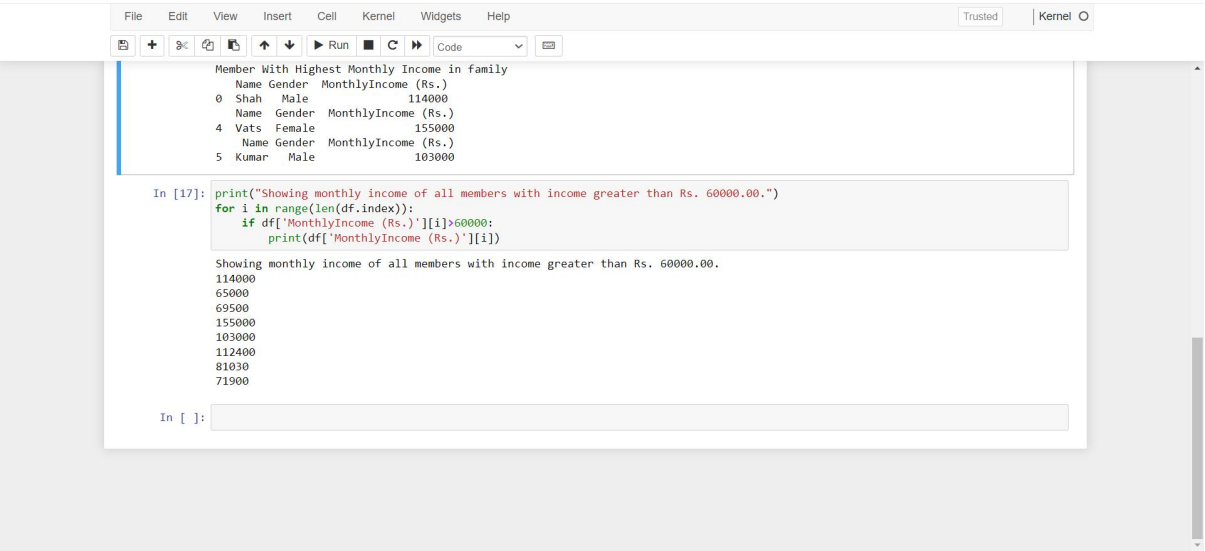
Code b):

```
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,:])
```



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])
```



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))
```