RAMANUJAN COLLEGE

UNIVERSITY OF DELHI

DSE – DATA ANALYSIS AND VISUALIZATION (PRACTICAL FILE)

SUBMITTED TO: Mrs. Sheetal Singh Ma'am

SUBMITTED BY: Sukaina Inam Naqvi

ROLL NO – 20201426

Examination Roll No.- 20020570033

B.Sc. (H) Computer Science | V Semester

PRACTICAL – QUESTIONS

Program No.	Question
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
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. ______________
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:____
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:_____
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)___
6.	Consider any sales training/ weather forecasting dataset_____-
7.	Consider a data frame containing data about students i.e. name, gender and passing division:______
8.	Consider the following data frame containing a family name, gender of the family member and her/his monthly income in each record.

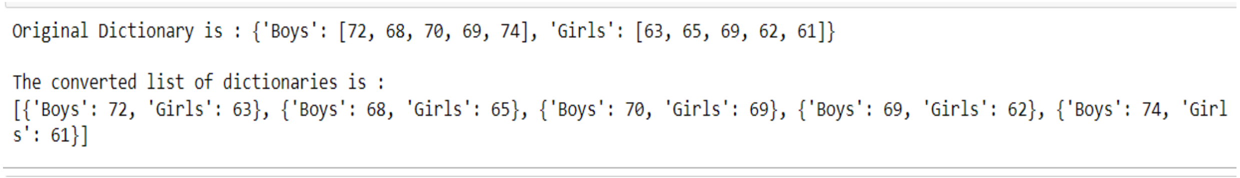
1.Given below is a dictionary having two keys ‘Boys’ and ‘Girls’ and having two lists of heights of five Boys and Five Girls respectively as values associated with these keys Original dictionary of lists:

{'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]

Solution :

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

print("Original Dictionary is : " + str(test_dict))

res = [{key:value[i] for key,value in test_dict.items()} for i in range(5)]

print("The converted list of dictionaries is : "  + str(res))
```

Output :



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

The converted list of dictionaries is :
[{'Boys': 72, 'Girls': 63}, {'Boys': 68, 'Girls': 65}, {'Boys': 70, 'Girls': 69}, {'Boys': 69, 'Girls': 62}, {'Boys': 74, 'Girl
s': 61}]
```

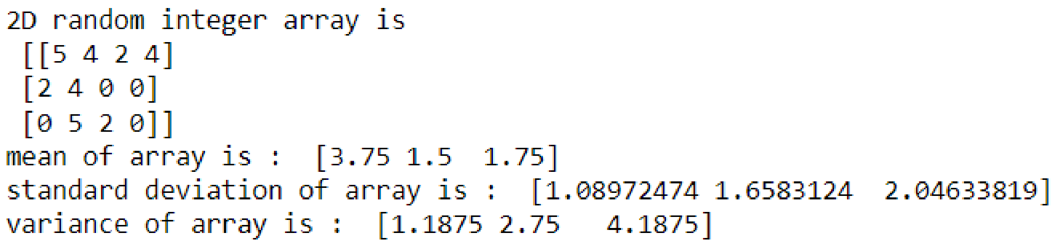
2. Write programs in Python using NumPy library to do the following:

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

Solution :

```
import numpy as np

from numpy import random

arr = np.random.randint(6, size=(3,4))

print("2D random integer array is  \n", arr)

print("mean of array is : ", arr.mean(axis= 1) )

print("standard deviation of array is : ", arr.std(axis= 1) )

print("variance of array is : ", arr.var(axis= 1) )
```

Output :



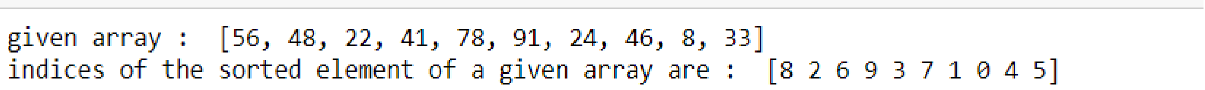
```
2D random integer array is
 [[5 4 2 4]
 [2 4 0 0]
 [0 5 2 0]]
mean of array is :  [3.75 1.5  1.75]
standard deviation of array is :  [1.08972474 1.6583124  2.04633819]
variance of array is :  [1.1875 2.75   4.1875]
```

b. Get the indices of the sorted elements of a given array. a. B = [56, 48, 22, 41, 78, 91, 24, 46, 8, 33]

Solution :

```
import numpy as np

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

print("given array : ", B)

indices = np.argsort(B)

print("indices of the sorted element of a given array are : ", indices)
```

Output :



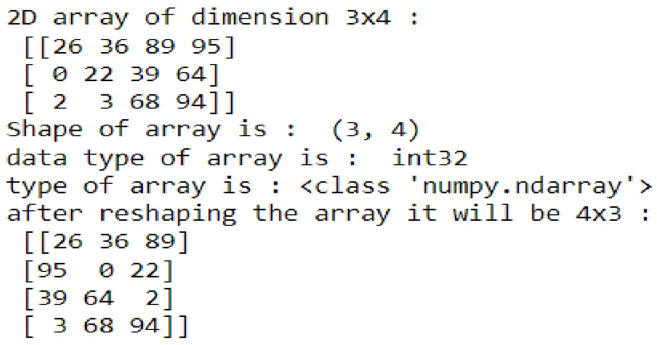
```
given array :  [56, 48, 22, 41, 78, 91, 24, 46, 8, 33]
indices of the sorted element of a given array are :  [8 2 6 9 3 7 1 0 4 5]
```

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.

Solution :

```
import numpy as np

from numpy import random

arr = np.random.randint(100,size=(3,4))

print("2D array of dimension 3x4 : \n",arr)

print("Shape of array is : ", np.shape(arr))

print("data type of array is : ", arr.dtype)

print("type of array is :", type(arr))

print("after reshaping the array it will be 4x3:\n", arr.reshape(4,3))
```

Output :



```
2D array of dimension 3x4 :
 [[26 36 89 95]
 [ 0 22 39 64]
 [ 2  3 68 94]]
Shape of array is :  (3, 4)
data type of array is :  int32
type of array is : <class 'numpy.ndarray'>
after reshaping the array it will be 4x3 :
 [[26 36 89]
 [95  0 22]
 [39 64  2]
 [ 3 68 94]]
```

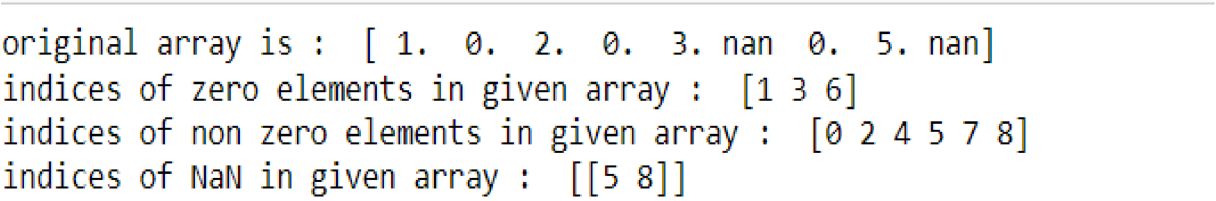
d. Test whether the elements of a given array are zero, non-zero and NaN. Record the indices of these elements in three separate arrays.

Solution :

```
import numpy as np

arr = np.array([1, 0, 2, 0, 3, np.nan, 0, 5 ,np.nan])
```

```
print("original array is : ", arr)

res = np.where(arr== 0)[0]

array1 = np.array(res)

print("indices of zero elements in given array : ", array1)

res1 = np.where(arr !=0)[0]

array2 = np.array(res1)

print("indices of non zero elements in given array : ",array2)

res2 = np.where(np.isnan(arr))

array3 = np.array(res2)

print("indices of NaN in given array : ",array3)
```

Output



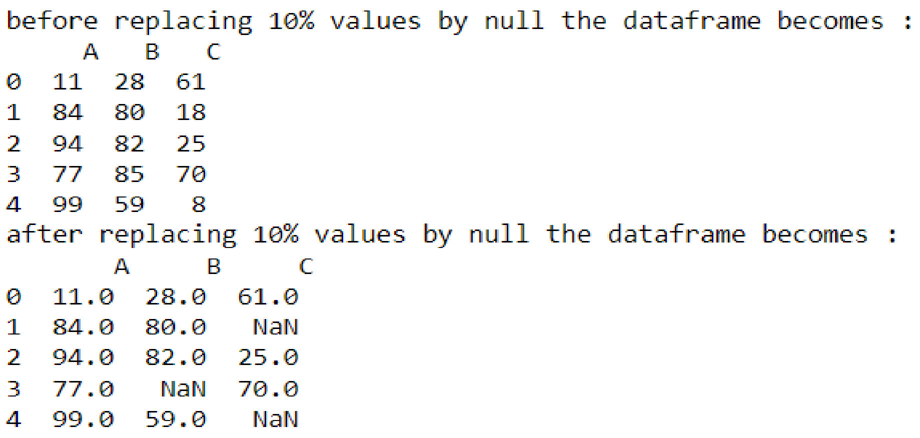
```
original array is :  [ 1.  0.  2.  0.  3. nan  0.  5. nan]
indices of zero elements in given array :  [1 3 6]
indices of non zero elements in given array :  [0 2 4 5 7 8]
indices of NaN in given array :  [[5 8]]
```

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:

Solution

```
import pandas as pd

import numpy as np

from numpy import random

df = pd.DataFrame(np.random.randint(0,100,size=(50, 3)),columns=list('ABC'))

ncols = len(df.columns)

print("before replacing 10% values by null the dataframe becomes :\n", df.head())

while df.isnull().sum().sum() != (ncols*50//10):

        df.iloc[random.randint(50),random.randint(ncols)] = None

print("after replacing 10% values by null the dataframe becomes :\n", df.head())
```

Output :



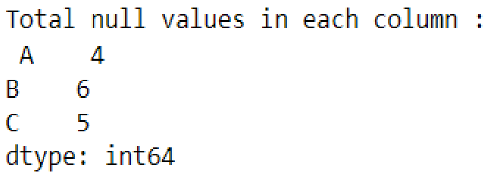
```
before replacing 10% values by null the dataframe becomes :
    A   B   C
0  11  28  61
1  84  80  18
2  94  82  25
3  77  85  70
4  99  59   8
after replacing 10% values by null the dataframe becomes :
      A     B     C
0  11.0  28.0  61.0
1  84.0  80.0   NaN
2  94.0  82.0  25.0
3  77.0   NaN  70.0
4  99.0  59.0   NaN
```

a) Identify and count missing values in a dataframe.

Solution:

print(" \nTotal null values in each column :\n",df.isnull().sum())

Output :



```
Total null values in each column :
 A    4
B    6
C    5
dtype: int64
```

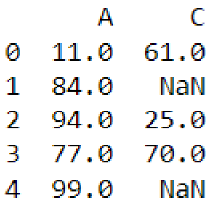
b) Drop the column having more than 5 null values.

Solution :

df1 = df.dropna(thresh=45,axis=1)

print(df1)

Output :



```
      A     C
0  11.0  61.0
1  84.0   NaN
2  94.0  25.0
3  77.0  70.0
4  99.0   NaN
```

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

Solution :

sum=df.sum(axis=1)

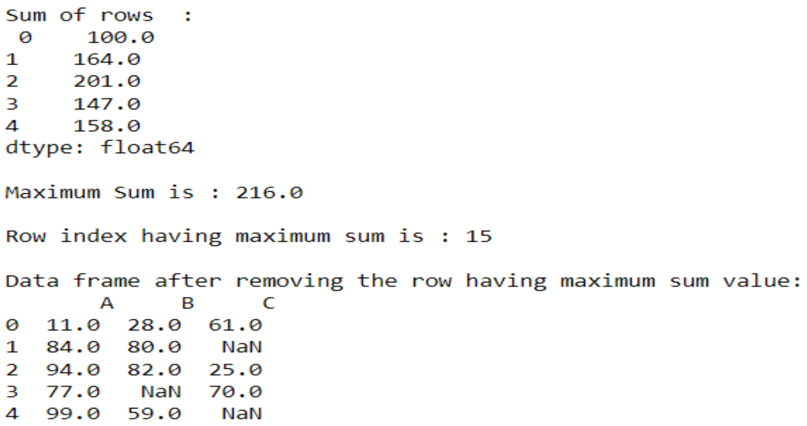
print("Sum of rows :\n",sum.head())

print("\nMaximum Sum is :",sum.max())

max_sum_row = df.sum(axis=1).idxmax()

```
print("\nRow index having maximum sum is :" ,max_sum_row)

df = df.drop(max_sum_row ,axis =0)

print("\nData frame after removing the row having maximum sum value: \n",df.head())
```

Output :



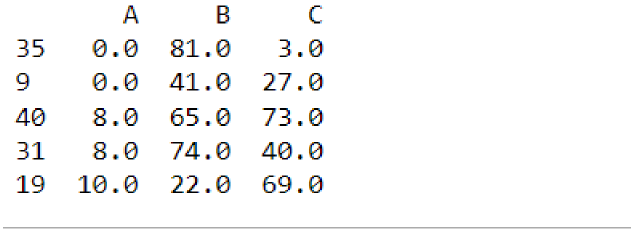
```
Sum of rows   :
 0     100.0
1     164.0
2     201.0
3     147.0
4     158.0
dtype: float64

Maximum Sum is : 216.0

Row index having maximum sum is : 15

Data frame after removing the row having maximum sum value:
       A     B     C
0   11.0  28.0  61.0
1   84.0  80.0   NaN
2   94.0  82.0  25.0
3   77.0   NaN  70.0
4   99.0  59.0   NaN
```

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

Solution:

```
sorted_df = df.sort_values(by='A')

print(sorted_df.head())
```

Output :



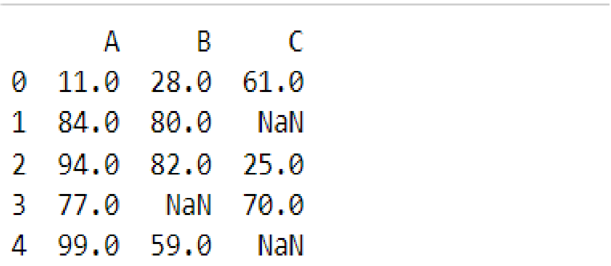
```
       A     B     C
35   0.0  81.0   3.0
9    0.0  41.0  27.0
40   8.0  65.0  73.0
31   8.0  74.0  40.0
19  10.0  22.0  69.0
```

e.) Remove all duplicates from the first column.

Solution :

```
df.drop_duplicates(subset="A")

print(df.head())
```

Output :



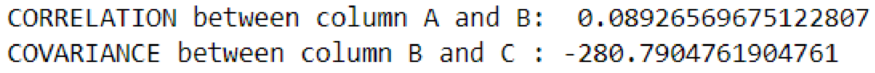
```
      A     B     C
0  11.0  28.0  61.0
1  84.0  80.0   NaN
2  94.0  82.0  25.0
3  77.0   NaN  70.0
4  99.0  59.0   NaN
```

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

Solution :

```
correlation = df['A'].corr(df['B'])
print("CORRELATION between column A and B: ", correlation)
covariance = df['B'].cov(df['C'])
print("COVARIANCE between column B and C :",covariance)
```

Output :



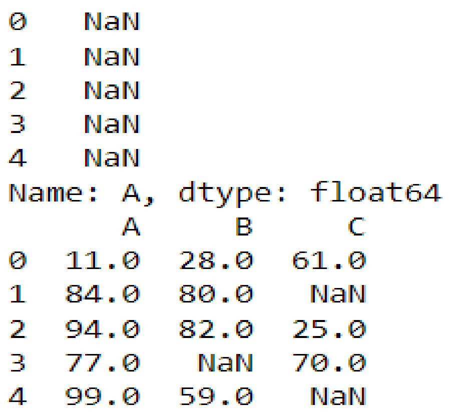
```
CORRELATION between column A and B:  0.08926569675122807
COVARIANCE between column B and C : -280.7904761904761
```

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

Solution:

```
from scipy import stats
import numpy as np

z = np.abs(stats.zscore(df['A']))
print(z)
threshold = 1
outlier_position = np.where(z > 1)
df.drop(outlier_position[0])
print(df)
```

Output :



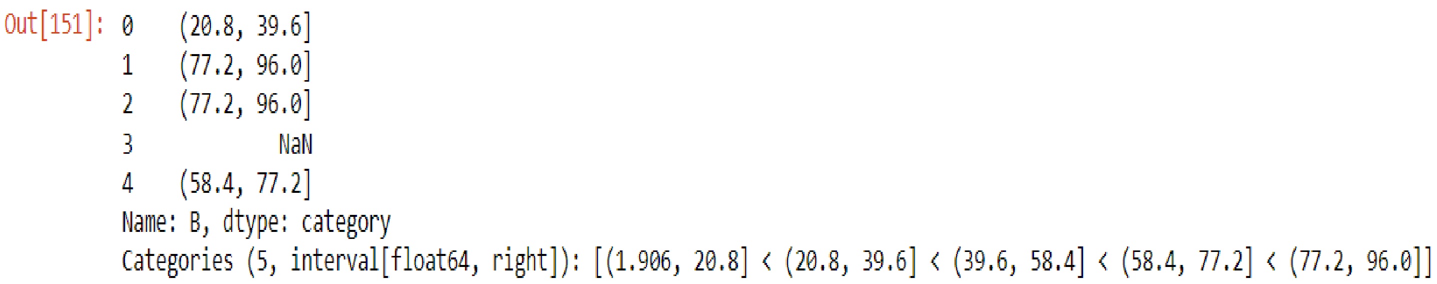
```
0    NaN
1    NaN
2    NaN
3    NaN
4    NaN
Name: A, dtype: float64
      A     B     C
0  11.0  28.0  61.0
1  84.0  80.0   NaN
2  94.0  82.0  25.0
3  77.0   NaN  70.0
4  99.0  59.0   NaN
```

h.) Discretize second column and create 5 bins

Solution :

```
df1 =  pd.cut(df['B'],bins=5).head()
df1
```

Output :



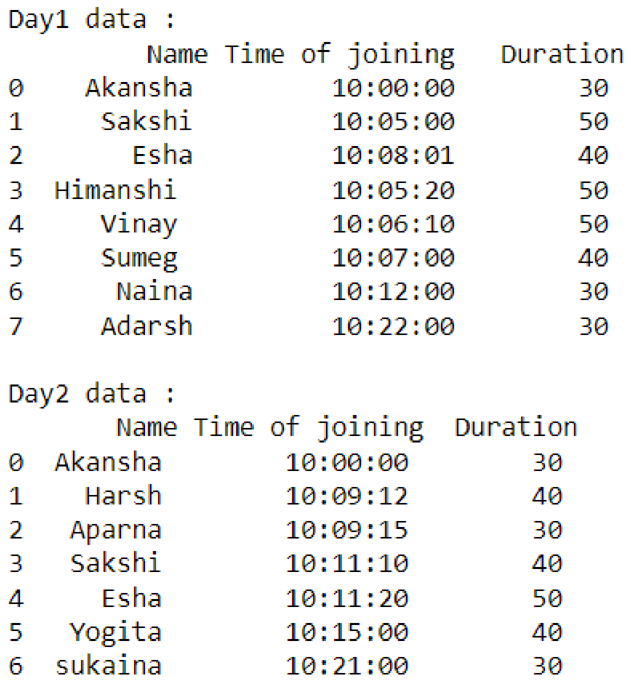
```
Out[151]: 0    (20.8, 39.6]
          1    (77.2, 96.0]
          2    (77.2, 96.0]
          3             NaN
          4    (58.4, 77.2]
          Name: B, dtype: category
          Categories (5, interval[float64, right]): [(1.906, 20.8] < (20.8, 39.6] < (39.6, 58.4] < (58.4, 77.2] < (77.2, 96.0]]
```

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:

Solution :

```
import numpy as np

import pandas as pd

d1_df = pd.read_excel(r'C:\Users\SUKAINA\Downloads\day1.xlsx')

d2_df = pd.read_excel(r'C:\Users\SUKAINA\Downloads\day2.xlsx')

print("Day1 data : \n",d1_df)

print("\nDay2 data : \n",d2_df)
```

Output :



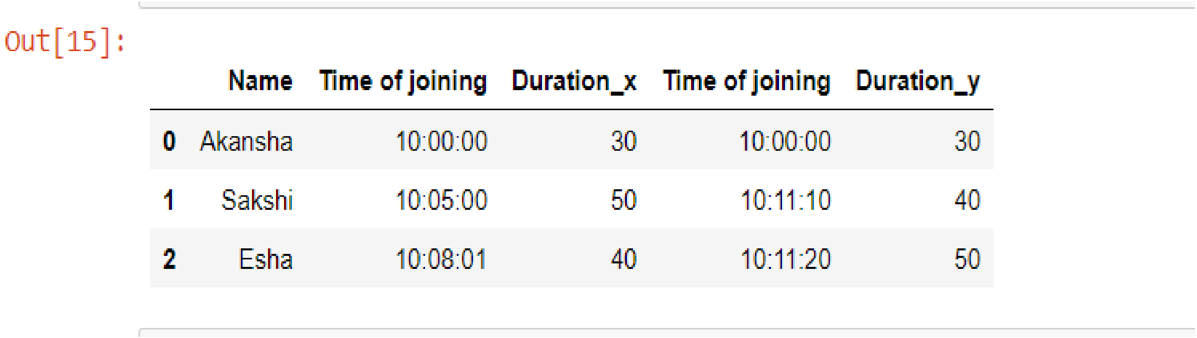
```
Day1 data : 
         Name Time of joining  Duration
0    Akansha        10:00:00        30
1     Sakshi        10:05:00        50
2       Esha        10:08:01        40
3   Himanshi        10:05:20        50
4      Vinay        10:06:10        50
5      Sumeg        10:07:00        40
6      Naina        10:12:00        30
7     Adarsh        10:22:00        30

Day2 data : 
       Name Time of joining  Duration
0  Akansha        10:00:00        30
1    Harsh        10:09:12        40
2   Aparna        10:09:15        30
3   Sakshi        10:11:10        40
4     Esha        10:11:20        50
5   Yogita        10:15:00        40
6  sukaina        10:21:00        30
```

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

Solution :

```
pd.merge(d1_df,d2_df,how='inner',on='Name')
```

Output :



Out[15]:

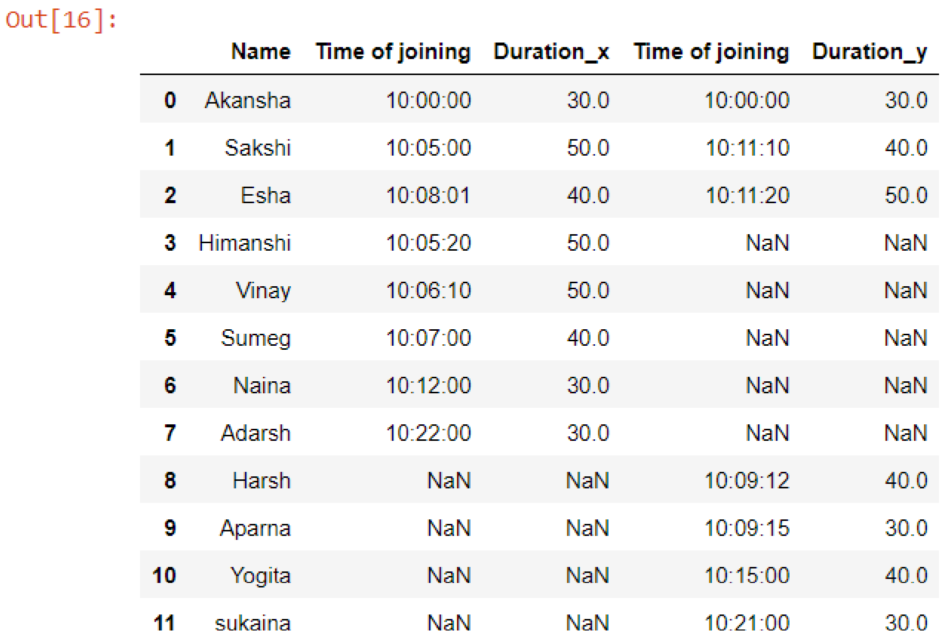
	Name	Time of joining	Duration_x	Time of joining	Duration_y
0	Akansha	10:00:00	30	10:00:00	30
1	Sakshi	10:05:00	50	10:11:10	40
2	Esha	10:08:01	40	10:11:20	50

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

Solution :

```
either_day = pd.merge(d1_df,d2_df,how='outer',on='Name')
either_day
```

Output :



Out[16]:

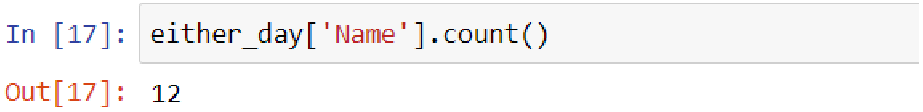
	Name	Time of joining	Duration_x	Time of joining	Duration_y
0	Akansha	10:00:00	30.0	10:00:00	30.0
1	Sakshi	10:05:00	50.0	10:11:10	40.0
2	Esha	10:08:01	40.0	10:11:20	50.0
3	Himanshi	10:05:20	50.0	NaN	NaN
4	Vinay	10:06:10	50.0	NaN	NaN
5	Sumeg	10:07:00	40.0	NaN	NaN
6	Naina	10:12:00	30.0	NaN	NaN
7	Adarsh	10:22:00	30.0	NaN	NaN
8	Harsh	NaN	NaN	10:09:12	40.0
9	Aparna	NaN	NaN	10:09:15	30.0
10	Yogita	NaN	NaN	10:15:00	40.0
11	sukaina	NaN	NaN	10:21:00	30.0

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

Solution :

```
either_day['Name'].count()
```

Output :



```
In [17]: either_day['Name'].count()

Out[17]: 12
```

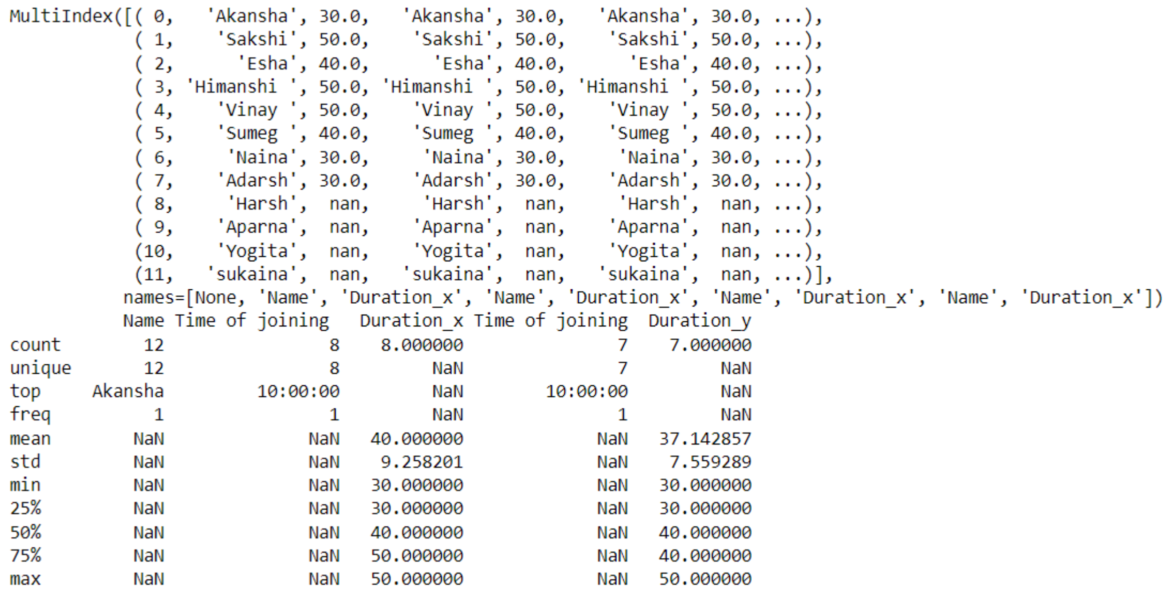
d.) Merge two data frames and use two columns names and duration as multi-row indexes. Generate descriptive statistics for this multi-index.

Solution :

```
either_day.set_index(['Name','Duration_x'],inplace = True,append = True,drop = False)
```

```
print(either_day.index)

print(either_day.describe(include='all')) #descriptive statistic of multi index multi index
```

Output :



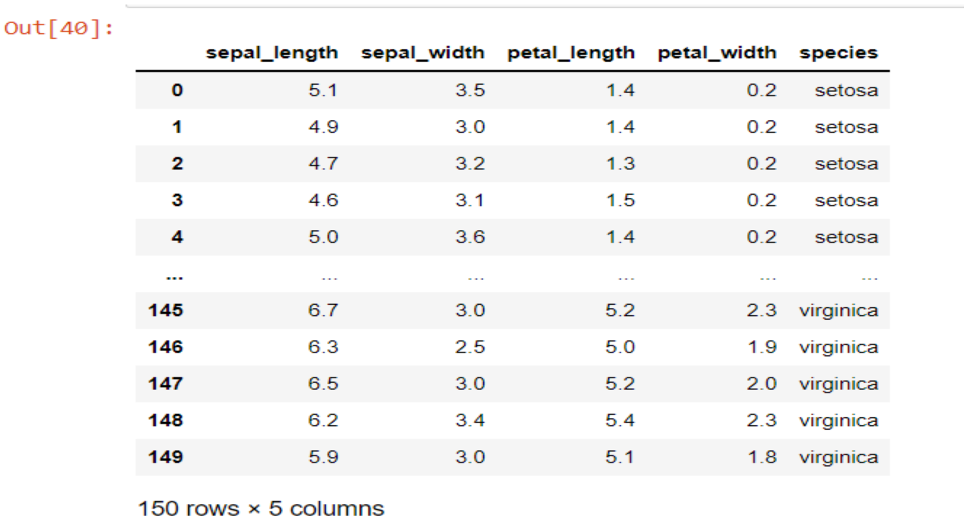
```
MultiIndex([( 0,   'Akansha', 30.0,   'Akansha', 30.0,   'Akansha', 30.0, ...),
            ( 1,    'Sakshi', 50.0,    'Sakshi', 50.0,    'Sakshi', 50.0, ...),
            ( 2,      'Esha', 40.0,      'Esha', 40.0,      'Esha', 40.0, ...),
            ( 3, 'Himanshi ', 50.0, 'Himanshi ', 50.0, 'Himanshi ', 50.0, ...),
            ( 4,    'Vinay ', 50.0,    'Vinay ', 50.0,    'Vinay ', 50.0, ...),
            ( 5,    'Sumeg ', 40.0,    'Sumeg ', 40.0,    'Sumeg ', 40.0, ...),
            ( 6,     'Naina', 30.0,     'Naina', 30.0,     'Naina', 30.0, ...),
            ( 7,    'Adarsh', 30.0,    'Adarsh', 30.0,    'Adarsh', 30.0, ...),
            ( 8,     'Harsh',  nan,     'Harsh',  nan,     'Harsh',  nan, ...),
            ( 9,    'Aparna',  nan,    'Aparna',  nan,    'Aparna',  nan, ...),
            (10,    'Yogita',  nan,    'Yogita',  nan,    'Yogita',  nan, ...),
            (11,   'sukaina',  nan,   'sukaina',  nan,   'sukaina',  nan, ...)],
           names=[None, 'Name', 'Duration_x', 'Name', 'Duration_x', 'Name', 'Duration_x', 'Name', 'Duration_x'])
           Name Time of joining  Duration_x Time of joining  Duration_y
count        12               8    8.000000               7    7.000000
unique       12               8         NaN               7         NaN
top     Akansha        10:00:00         NaN        10:00:00         NaN
freq          1               1         NaN               1         NaN
mean        NaN             NaN   40.000000             NaN   37.142857
std         NaN             NaN    9.258201             NaN    7.559289
min         NaN             NaN   30.000000             NaN   30.000000
25%         NaN             NaN   30.000000             NaN   30.000000
50%         NaN             NaN   40.000000             NaN   40.000000
75%         NaN             NaN   50.000000             NaN   40.000000
max         NaN             NaN   50.000000             NaN   50.000000
```

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)

Solution :

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

iris = sns.load_dataset('iris')

iris
```

Output :



Out[40]:

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

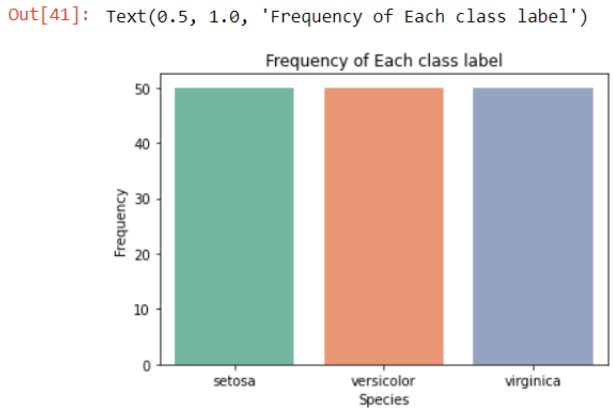
150 rows × 5 columns

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

Solution :

```
sns.countplot(x='species',data=iris,palette='Set2')
plt.xlabel('Species')
plt.ylabel('Frequency')
plt.title('Frequency of Each class label')
```

Output :



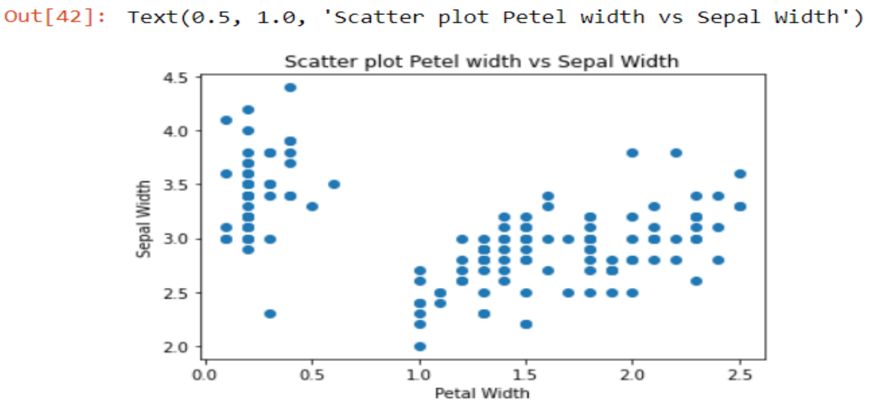
```
Out[41]: Text(0.5, 1.0, 'Frequency of Each class label')
```

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

Solution :

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

Output :



```
Out[42]: Text(0.5, 1.0, 'Scatter plot Petel width vs Sepal Width')
```

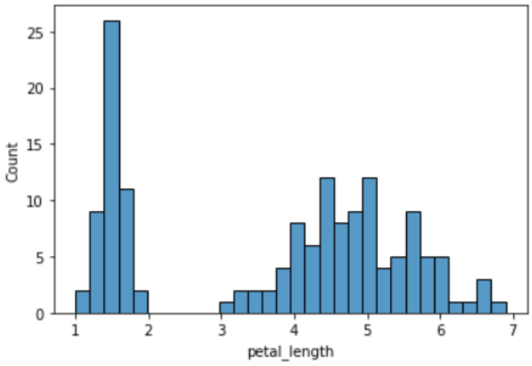
c.) Plot density distribution for feature petal length.

Solution :

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

Output :

```
Out[43]: <AxesSubplot:xlabel='petal_length', ylabel='Count'>
```



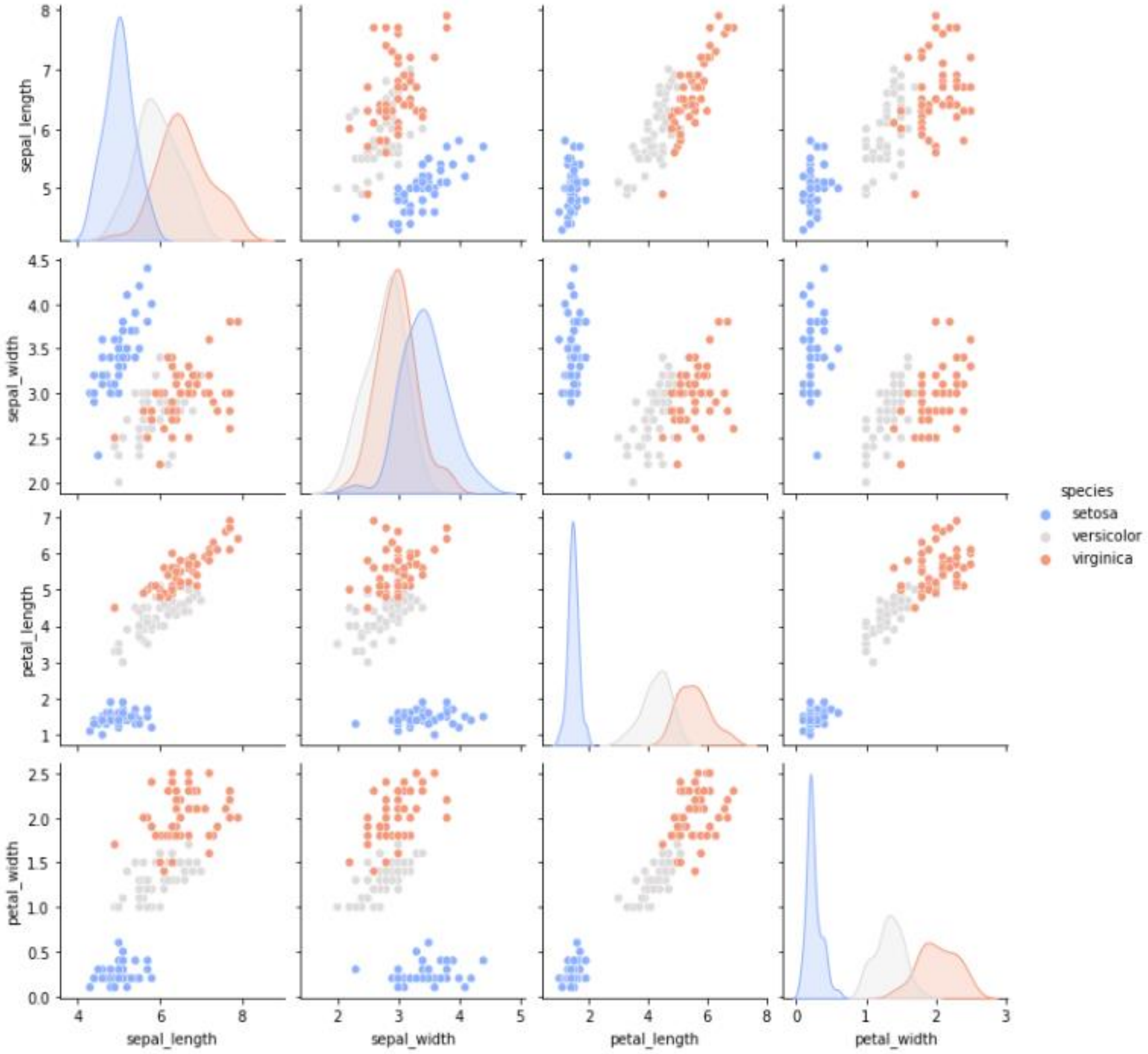
d.) Use a pair plot to show pairwise bivariate distribution in the Iris Dataset.

Solution :

```
sns.pairplot(iris,hue='species',palette='coolwarm')
```

Output :

```
Out[44]: <seaborn.axisgrid.PairGrid at 0x1dd48254bb0>
```



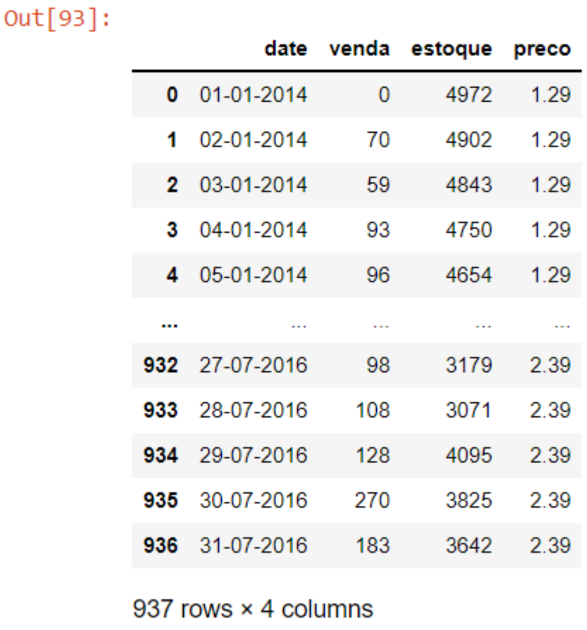
6. Consider any sales training/ weather forecasting dataset

- > I take retail sales forecasting dataset from Kaggle Retail Sales Forecasting | Kaggle ...

Solution :

```
import pandas as pd

import numpy as np

data = pd.read_csv(r"C:\Users\SUKAINA\Downloads\archive (3)\mock_kaggle.csv")

df = pd.DataFrame(data)

df
```

Output :



Out[93]:

	date	venda	estoque	preco
0	01-01-2014	0	4972	1.29
1	02-01-2014	70	4902	1.29
2	03-01-2014	59	4843	1.29
3	04-01-2014	93	4750	1.29
4	05-01-2014	96	4654	1.29
...	...	...	...	...
932	27-07-2016	98	3179	2.39
933	28-07-2016	108	3071	2.39
934	29-07-2016	128	4095	2.39
935	30-07-2016	270	3825	2.39
936	31-07-2016	183	3642	2.39

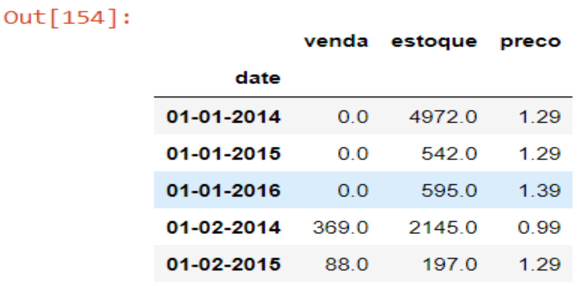
937 rows × 4 columns

a.) **Compute mean of a series grouped by another series**

Solution :

```
mean_series = df.groupby(['date']).mean()

mean_series.head()
```

Output :



Out[154]:

	venda	estoque	preco
date			
01-01-2014	0.0	4972.0	1.29
01-01-2015	0.0	542.0	1.29
01-01-2016	0.0	595.0	1.39
01-02-2014	369.0	2145.0	0.99
01-02-2015	88.0	197.0	1.29

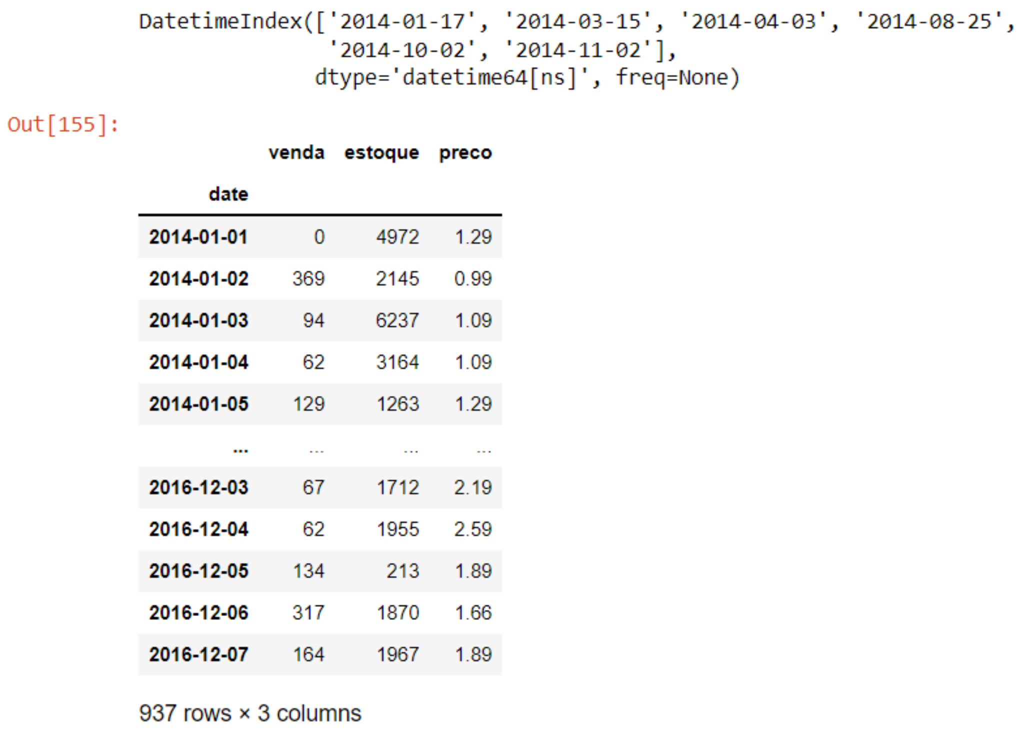
b.) Fill an intermittent time series to replace all missing dates with values of previous non-missing date.

Solution :

```
df = df.set_index('date')
```

```
# to_datetime() method converts string

# format to a DateTime object

df.index = pd.to_datetime(df.index)

# dates which are not in the sequence

# are returned

print(pd.date_range (start="2014-1-1", end="2015-12-31").difference(df.index))

df.sort_values(['date','venda','estoque','preco']).groupby('date').ffill()
```

Output:



```
DatetimeIndex(['2014-01-17', '2014-03-15', '2014-04-03', '2014-08-25',
               '2014-10-02', '2014-11-02'],
              dtype='datetime64[ns]', freq=None)
```

Out[155]:

	venda	estoque	preco
date			
2014-01-01	0	4972	1.29
2014-01-02	369	2145	0.99
2014-01-03	94	6237	1.09
2014-01-04	62	3164	1.09
2014-01-05	129	1263	1.29
...	...	...	...
2016-12-03	67	1712	2.19
2016-12-04	62	1955	2.59
2016-12-05	134	213	1.89
2016-12-06	317	1870	1.66
2016-12-07	164	1967	1.89

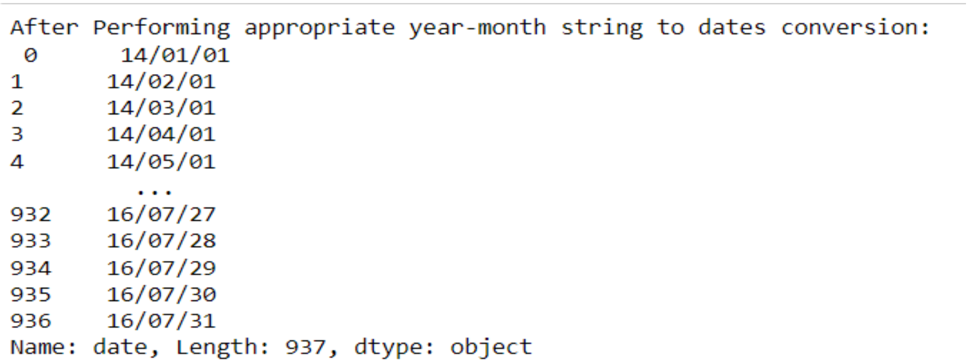
937 rows × 3 columns

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

Solution:

```
from datetime import datetime

df['date'] = pd.to_datetime(df['date'])

change_format = df['date'].dt.strftime('%y/%m/%d')

print("After Performing appropriate year-month string to dates conversion: \n",change_format)
```

Output:



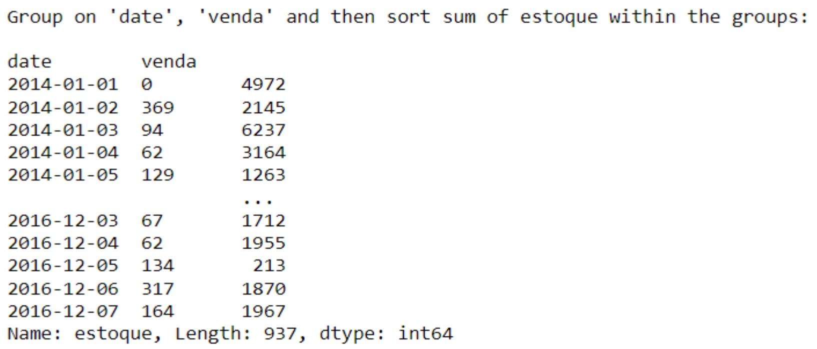
```
After Performing appropriate year-month string to dates conversion:
 0      14/01/01
1       14/02/01
2       14/03/01
3       14/04/01
4       14/05/01
          ...
932     16/07/27
933     16/07/28
934     16/07/29
935     16/07/30
936     16/07/31
Name: date, Length: 937, dtype: object
```

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

Solution :

```
df_agg = df.groupby(['date','venda']).agg({'estoque':sum})

result = df_agg['estoque'].groupby(level=0, group_keys=False)

print("\nGroup on 'date', 'venda' and then sort sum of estoque within the groups:\n")

print(result.nlargest())
```

Output :



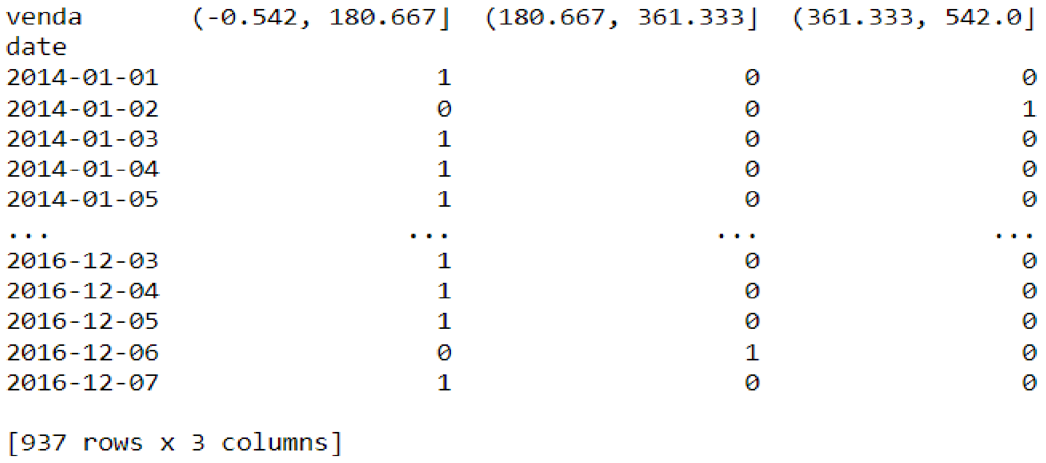
```
Group on 'date', 'venda' and then sort sum of estoque within the groups:

date        venda
2014-01-01  0        4972
2014-01-02  369      2145
2014-01-03  94       6237
2014-01-04  62       3164
2014-01-05  129      1263
                     ...
2016-12-03  67       1712
2016-12-04  62       1955
2016-12-05  134       213
2016-12-06  317      1870
2016-12-07  164      1967
Name: estoque, Length: 937, dtype: int64
```

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

Solution:

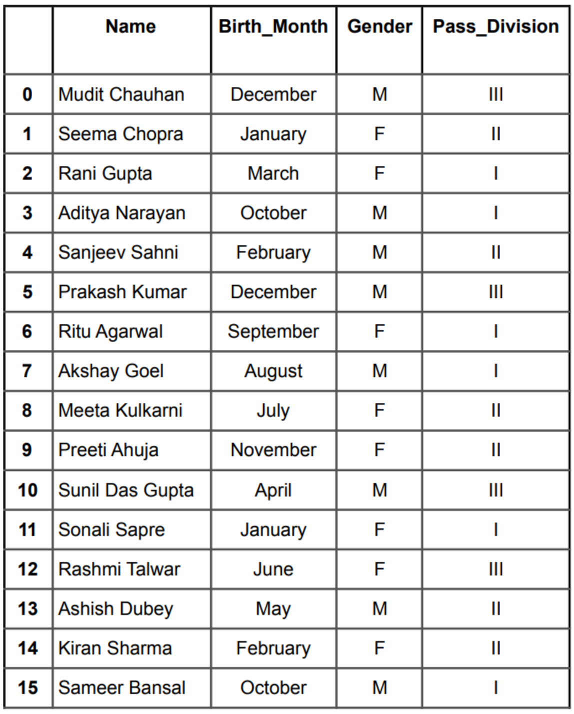
```
groups = df.groupby(['date', pd.cut(df.venda, 3)])

result = groups.size().unstack()

print(result)
```

Output:



```
venda       (-0.542, 180.667]  (180.667, 361.333]  (361.333, 542.0]
date
2014-01-01                  1                   0                 0
2014-01-02                  0                   0                 1
2014-01-03                  1                   0                 0
2014-01-04                  1                   0                 0
2014-01-05                  1                   0                 0
...                       ...                 ...               ...
2016-12-03                  1                   0                 0
2016-12-04                  1                   0                 0
2016-12-05                  1                   0                 0
2016-12-06                  0                   1                 0
2016-12-07                  1                   0                 0

[937 rows x 3 columns]
```

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.

Solution :

```
import pandas as pd

#categorical data

categorical_cols = ['Gender','pass_division']

df = pd.get_dummies(data, columns = categorical_cols)
```

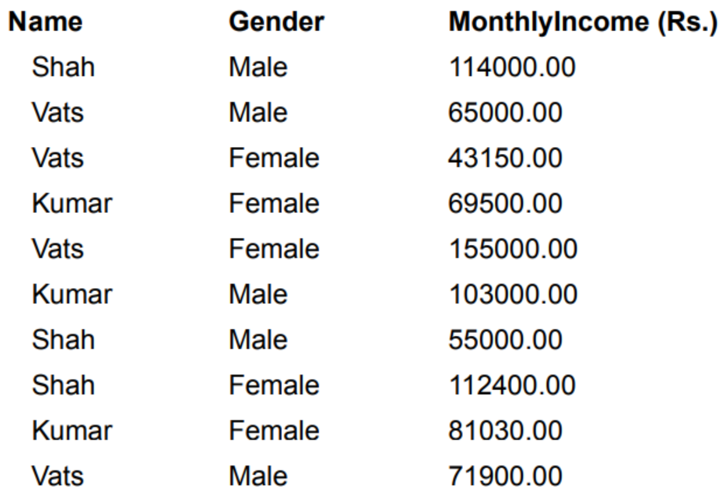
b.) Sort this data frame on the “Birth Month” column (i.e. January to December). Hint: Convert Month to Categorical.

Solution:

```
 import pandas as pd

dates_in_order = pd.date_range(start='2022-01-01',end='2022-12-01',freq='MS')

months_in_order= dates_in_order.map(lambda x : x.month_name()).to_list()

df['month'] = pd.Categorical(df['month'],

        categories=months_in_order,

        ordered=True)

df.sort_values('month')
```

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

Solution :



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

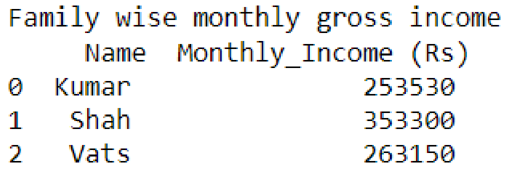
Write a program in Python using Pandas to perform the following:

a.) Calculate and display familywise gross monthly income

Solution :

```
import pandas as pd

df =  pd.DataFrame({

'Name': ['Shah','Vats','Vats','Kumar','Vats','Kumar','Shah','Shah','Kumar','Shah'],

'Gender': ['Male','Male' ,'Female','Female','Female','Male','Male','Female','Fem ale','Male'],

'Monthly_Income ': [114000,65000,43150,69500,155000,103000,55000,112400,81030,71900]})

df

gross_salary = df.groupby(by=['Name'], as_index=False)['Monthly_Income (Rs)'].sum()

print (gross_salary)
```

Output :



```
Family wise monthly gross income
     Name  Monthly_Income (Rs)
0  Kumar                253530
1   Shah                353300
2   Vats                263150
```

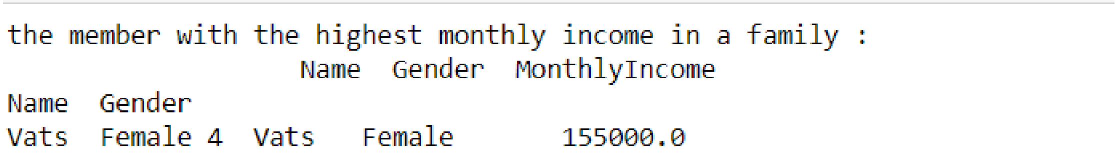
b.) Calculate and display the member with the highest monthly income in a family.

Solution :

```
max_salary = data.groupby(by=['Name','Gender']).apply(lambda x : x[x['MonthlyIncome'] ==
x['MonthlyIncome'].max()])

max_salary
```

```
s = max(max_salary['MonthlyIncome'])

res = max_salary[max_salary['MonthlyIncome'] == s ]

print("the member with the highest monthly income in a family :\n ",res)
```

Output :



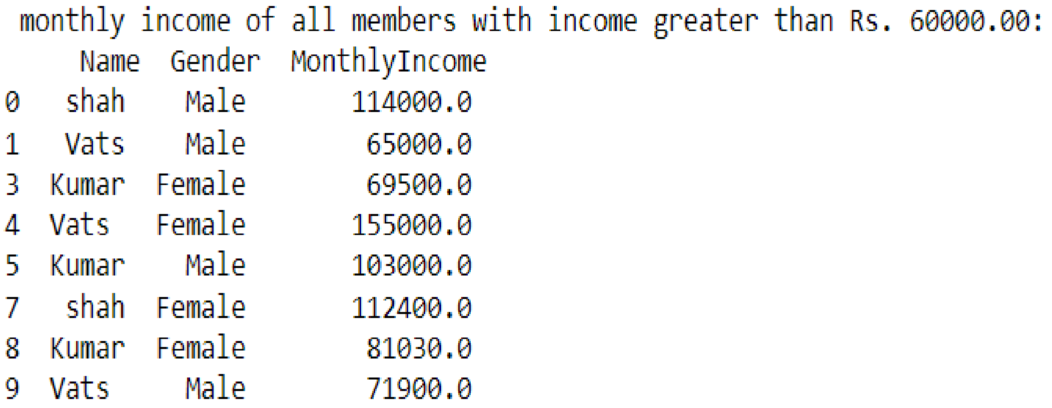
```
the member with the highest monthly income in a family :
                       Name   Gender  MonthlyIncome
Name  Gender
Vats  Female 4  Vats    Female       155000.0
```

c.) Calculate and display monthly income of all members with income greater than Rs. 60000.00.

Solution :

```
greater_income = data[data['MonthlyIncome'] > 60000.00]

print(" monthly income of all members with income greater than Rs. 60000.00:    \n",greater_income)
```

Output:



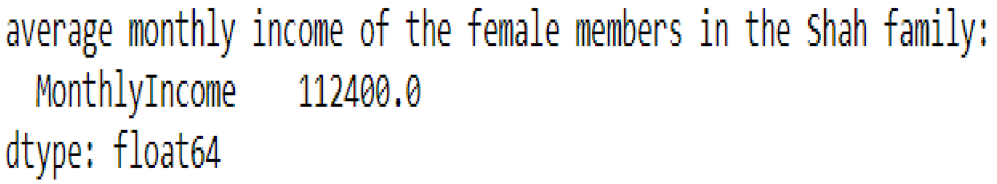
```
 monthly income of all members with income greater than Rs. 60000.00:
     Name  Gender  MonthlyIncome
0    shah    Male       114000.0
1    Vats    Male        65000.0
3   Kumar  Female        69500.0
4    Vats  Female       155000.0
5   Kumar    Male       103000.0
7    shah  Female       112400.0
8   Kumar  Female        81030.0
9    Vats    Male        71900.0
```

d.) Calculate and display the average monthly income of the female members in the Shah family.

Solution :

```
average = data[(data['Name']== 'shah') & (data['Gender']=='Female')].mean()

print("average monthly income of the female members in the Shah family: \n ",average)
```

Output :



```
average monthly income of the female members in the Shah family:
  MonthlyIncome    112400.0
dtype: float64
```