NAME : Chirag Sharma

ROLL NO. : 20232762

COURSE : B.VOC SOFTWARE DEVELOPMENT

SUBJECT : DATA ANALYSIS AND VISUALIZATION USING PYTHON

Submited to :DR.ARUN AGARWAL SIR

PRACTICAL 1

```
#1. 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
import numpy as np
x=np.random.randint(10,50,5)
y=np.random.randint(5,80,5)
arr=np.array((x))
arr1=np.array((y))
z=np.array([[arr],[arr1]])
print(z)
print(z.shape)
print("Mean\n",z.mean(axis=1))
print("standard deviation\n",z.std(axis=1))
print("variance\n",z.var(axis=1))
```

```
[[[18 16 13 12 23]]

 [[21 41  9 60 69]]]
(2, 1, 5)
Mean
 [[18. 16. 13. 12. 23.]
 [21. 41.  9. 60. 69.]]
standard deviation
 [[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
variance
 [[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
```

```
#b. 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 an n x m array, where n and m are user inputs given at the run time.
import numpy as np
m=eval(input("enter a row number"))
n=eval(input("enter a column number"))
x=np.random.randint(4,50,m)
y=np.random.randint(5,80,n)
arr=np.array((x))
arr1=np.array((y))
z=np.array([[arr],[arr1]])
print(z)
print(z.shape)
print(z.dtype)
```

```
enter a row number5
enter a column number5
[[[35  5 31 20 25]]

 [[58 18 24 35 60]]]
(2, 1, 5)
int64
```

```python
#c. Test whether the elements of a given 1D array are zero, non-zero and NaN. Record the indices of these
#elements in three separate arrays.

import numpy as np
arr = np.array([5,69,8,np.nan,89,0,5,0,56,0,np.nan,np.nan,89,569])
l1 = []
l2 = []
l3 = []
for i in range(len(arr)):
  if (np.isnan(arr[i])):
    l1.append(i)
  elif (i == 0):
    l2.append(i)
  else:
    l3.append(i)

print("null value index",l1)
print("zero value index",l2)
print("not null value index",l3)
```

```
null value index [3, 10, 11]
zero value index [0]
not null value index [1, 2, 4, 5, 6, 7, 8, 9, 12, 13]
```

```python
#d. Create three random arrays of the same size: Array1, Array2 and Array3. Subtract Array 2 from Array3
#and store in Array4. Create another array Array5 having two times the values in Array1.
# Find Covariance and Correlation of Array1 with Array4 and Array5 resp
import numpy as np
x=np.random.randint(0,100,5)
y=np.random.randint(0,100,5)
z=np.random.randint(0,100,5)
array_1=np.array((x))
array_2=np.array((y))
array_3=np.array((z))
array_4=np.array((array_2 - array_3))
print("array_4",array_4)
array_5=np.array((2*array_1))
print ("array_5",array_5)

print("correlation",np.corrcoef([array_1],[array_4]))
print("correlation",np.corrcoef([array_1],[array_5]))
print("covariance",np.cov(array_1,array_4))
print("covariance",np.cov(array_1,array_5))
```

```
array_4 [ 31  49  83  15 -16]
Code cell output actions
 190 152  16]
correlation [[1.          0.51730282]
 [0.51730282 1.        ]]
correlation [[1. 1.]
 [1. 1.]]
covariance [[1630.8   773.45]
 [ 773.45 1370.8 ]]
covariance [[1630.8 3261.6]
 [3261.6 6523.2]]
```

```python
#e. Create two random arrays of the same size 10: Array1, and Array2. Find the sum of the first half of both
#the arrays and product of the second half of both the arrays.
import numpy as np
x=np.random.randint(0,50,10)
y=np.random.randint(0,100,10)
arr=np.array((x))
arr1=np.array((y))
z=arr[:len(x)//2]
#print(arr[:z])
z1=arr1[:len(y)//2]
#print(arr1[:z1])
print("sum of z and z1",z+z1)
m = arr[len(x)//2:]
m1 = arr1[len(x)//2:]
print("multiple of m and m1",m*m1)
```

```
sum of z and z1 [121 117  77  87  66]
multiple of m and m1 [3040 1599 1092 1887  611]
```

PRACTICAL 2

```
#2. Do the following using PANDAS Series:
import pandas as pd
series = pd.Series((12,23,45,67,876,87,6,5445,43,45,5,66,7,7,6,6,6,56,0,78,8,89,6664,865,78,65))

#a. Create a series with 5 elements. Display the series sorted on index and also sorted on values seperately
sorted = series.sort_index()
sorted_values = series.sort_values()
print(sorted)
print("sorted_values",sorted_values)
```

```
0       12
1       23
2       45
3       67
4      876
5       87
6        6
7     5445
8       43
9       45
10       5
11      66
12       7
13       7
14       6
15       6
16       6
17      56
18       0
19      78
20       8
21      89
22    6664
23     865
24      78
25      65
dtype: int64
```

```
sorted_values 18       0
10       5
16       6
15       6
14       6
6        6
12       7
13       7
20       8
0       12
1       23
8       43
2       45
9       45
17      56
25      65
11      66
3       67
24      78
19      78
5       87
21      89
23     865
4      876
7     5445
22    6664
dtype: int64
```

```
#b. Create a series with N elements with some duplicate values. Find the minimum and maximum ranks
#assigned to the values using ‘first’ and ‘max’ methods
print("According to first method\n",series.rank(method = "first"))
print("\n According to max method\n",series.rank(method = "max"))
```

```
According to first method
 0     10.0
1     11.0
2     13.0
3     18.0
4     24.0
5     21.0
6      3.0
7     25.0
8     12.0
9     14.0
10     2.0
11    17.0
12     7.0
13     8.0
14     4.0
15     5.0
16     6.0
17    15.0
18     1.0
19    19.0
20     9.0
21    22.0
22    26.0
23    23.0
24    20.0
25    16.0
dtype: float64
```

```
According to max method
0     10.0
1     11.0
2     14.0
3     18.0
4     24.0
5     21.0
6      6.0
7     25.0
8     12.0
9     14.0
10     2.0
11    17.0
12     8.0
13     8.0
14     6.0
15     6.0
16     6.0
17    15.0
18     1.0
19    20.0
20     9.0
21    22.0
22    26.0
23    23.0
24    20.0
25    16.0
dtype: float64
```

```
#c. Display the index value of the minimum and maximum element of a Series
x = series.max()
y = series.min()
print("Max value",series.loc[series.idxmax()])
print("Max values index",series.idxmax())
print("Min value",series.loc[series.idxmin()])
print("Min values index",series.idxmin())
```

```
dtype: float64
Max value 6664
Max values index 22
Min value 0
Min values index 18
```

PRACTICAL 3

```python
#3. Create a data frame 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:
import pandas as pd
import numpy as np
series = np.random.rand(10,3)
data = pd.DataFrame(series,columns = ['column1' ,'column2','column3'])
#print(data)
null_values = np.random.choice(data.size , size = int(0.1 * data.size))
data.values.ravel()[null_values] = np.nan
#print(data)

#a. Identify and count missing values in a data frame.
missing = data.isnull().sum()
print(missing)

#b. Drop the column having more than 5 null values.
#data = data.dropna(axis = 1 ,thresh = len(data) - 5)

#c. Identify the row label having maximum of the sum of all values in a row and drop that row.
#max_row = data.sum(axis = 1).idxmax()
#data= data.drop(index = max_row)
```

```python
#d. Sort the data frame on the basis of the first column.
#data = data.sort_values("column1")

#e. Remove all duplicates from the first column.
#data = data.drop_duplicates("column1")

#f. Find the correlation between first and second column and covariance between second and third
#column.
correlation = data["column1"].corr(data["column2"])
covariance = data["column2"].cov(data["column3"])
print(correlation)
print(covariance)

#g. Discretize the second column and create 5 bins.
data["column2_bins"] = pd.cut(data["column2"] , bins = 5)
print(data)
```

```
column1    2
column2    0
column3    1
dtype: int64
-0.236751857037486
-0.04964966980899943
    column1   column2   column3    column2_bins
0  0.003349  0.714170  0.467341  (0.642, 0.817]
1  0.931825  0.423493  0.823253  (0.292, 0.467]
2  0.690441  0.992039  0.058476  (0.817, 0.992]
3       NaN  0.615732  0.612446  (0.467, 0.642]
4  0.876113  0.117184  0.969109  (0.116, 0.292]
5  0.570485  0.916646  0.759010  (0.817, 0.992]
6  0.492786  0.404287  0.622129  (0.292, 0.467]
7  0.063221  0.453420  0.674204  (0.292, 0.467]
8  0.612525  0.185338       NaN  (0.116, 0.292]
9       NaN  0.674480  0.726861  (0.642, 0.817]
```

PRACTICAL 4

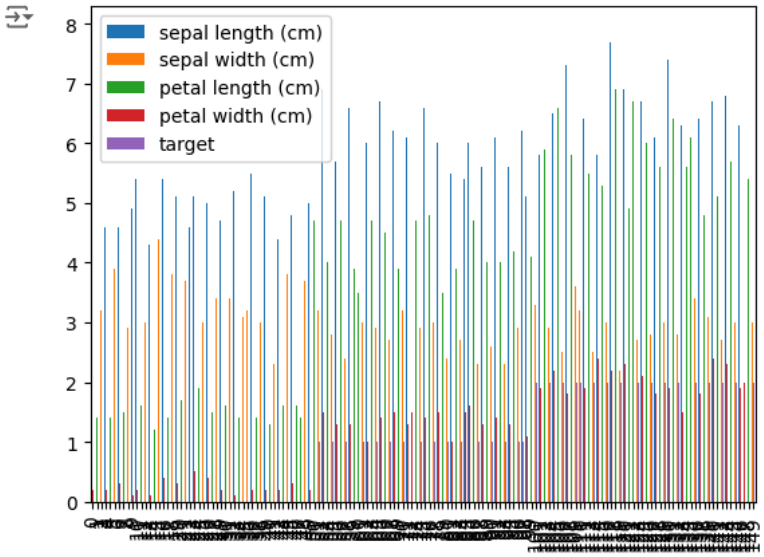
```python
#4. Consider two excel files having attendance of two workshops. Each file has three fields ‘Name’, ‘Date, duration
#(in minutes) where names are unique within a file. Note that duration may take one of three values (30, 40, 50)
#only. Import the data into two data frames and do the following:

import pandas as pd
df1 = pd.read_excel('workshop1.xlsx')
df2 = pd.read_excel('workshop2.xlsx')

# a. Perform merging of the two dataframes to find the names of students who attended both workshops
attended_both = pd.merge(df1, df2, on='Name', how='inner')
print("names of students who attended both workshops\n",attended_both)

# b. Find names of all students who attended a single workshop only
attended_one = pd.concat([df1, df2]).drop_duplicates(keep=False)['Name']
print("names of all students who attended a single workshop\n",attended_one)

# c. Merge two dataframes row-wise and find the total number of records
rowwise = pd.concat([df1, df2])
print(" the total number of records\n",len(rowwise))

# d. Merge two dataframes row-wise and use two columns (names and dates) as multi-row indexes
rowwise_indexes = pd.concat([df1.set_index(['Name', 'Date']), df2.set_index(['Name', 'Date'])])

# e. Generate descriptive statistics for this hierarchical dataframe
statistics = rowwise_indexes.describe()
print(statistics)
```

```
names of students who attended both workshops
     Name     Date_x  Duration_x     Date_y  Duration_y
0     RAM 2024-02-03          45 2024-05-03          45
1   NAREN 2024-02-04          45 2024-03-04          45
2   DHRUV 2024-02-06          45 2024-03-06          45
3    AMAN 2024-02-08          45 2024-03-08          45
4   MOHAN 2024-02-10          45 2024-03-10          45
names of all students who attended a single workshop
 0     SHIVAM
1        RAM
2      NAREN
3    ASHWANI
4      DHRUV
5      VIKAS
6       AMAN
7     DEEPAK
8      MOHAN
0     RUPESH
1        RAM
2      NAREN
3        DEV
4      DHRUV
5     ASHISH
6       AMAN
7      SHLOK
8      MOHAN
Name: Name, dtype: object
```

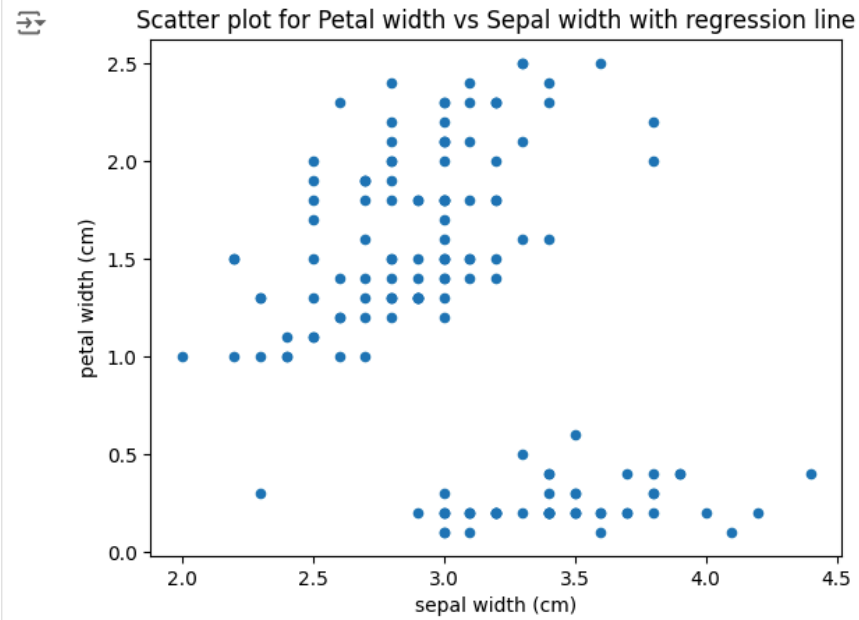
```
 the total number of records
 18
       Duration
count      18.0
mean       45.0
std         0.0
min        45.0
25%        45.0
50%        45.0
75%        45.0
max        45.0
```

PRACTICAL 5

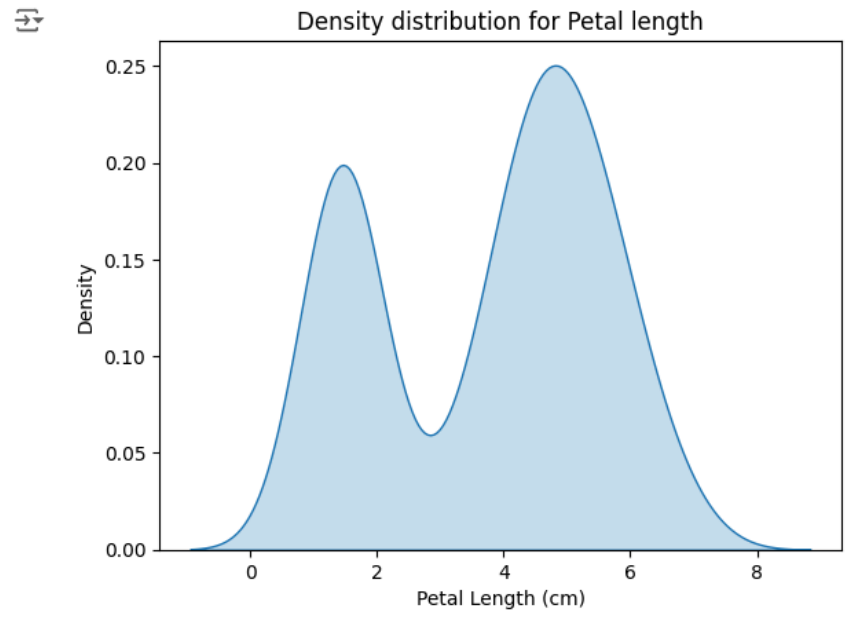
```
#5. Using Iris data, plot the following with proper legend and axis labels: (Download IRIS data from:
#https://archive.ics.uci.edu/ml/datasets/iris or import it from sklearn datasets
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris


iris = load_iris()
iris_df = pd.DataFrame(data=np.c_[iris['data'], iris['target']],
                       columns=iris['feature_names'] + ['target'])
iris_df['species'] = iris_df['target'].map({0: 'setosa', 1: 'versicolor', 2: 'virginica'})

#a. Plot bar chart to show the frequency of each class label in the data.
iris_df.plot.bar()
plt.show()
```



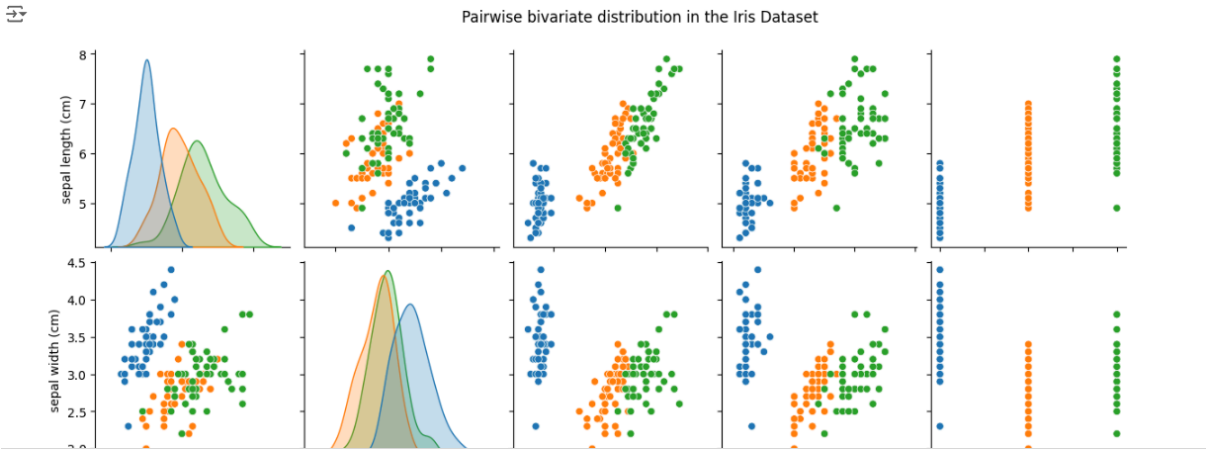

```
#b. Draw a scatter plot for Petal width vs sepal width and fit a regression line
iris_df.plot.scatter(x='sepal width (cm)', y='petal width (cm)')
plt.title('Scatter plot for Petal width vs Sepal width with regression line')
plt.show()
```



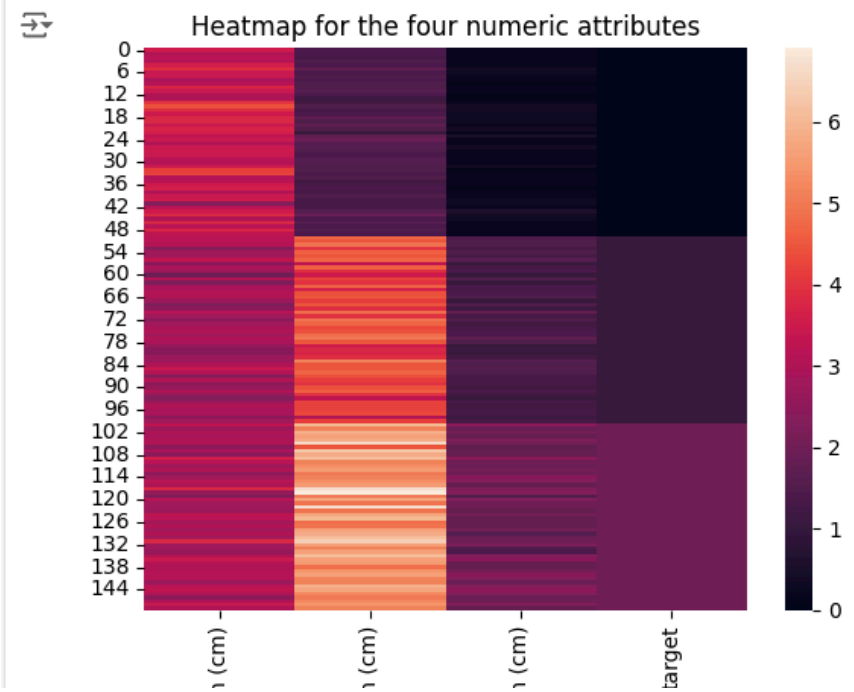

```
#c. Plot density distribution for feature petal length.
sns.kdeplot(data=iris_df['petal length (cm)'], shade=True)
plt.title('Density distribution for Petal length')
plt.xlabel('Petal Length (cm)')
plt.ylabel('Density')
plt.show()
```




```
#d. Use a pair plot to show pairwise bivariate distribution in the Iris Dataset.
sns.pairplot(iris_df, hue='species')
plt.suptitle('Pairwise bivariate distribution in the Iris Dataset', y=1.02)
plt.show()
```



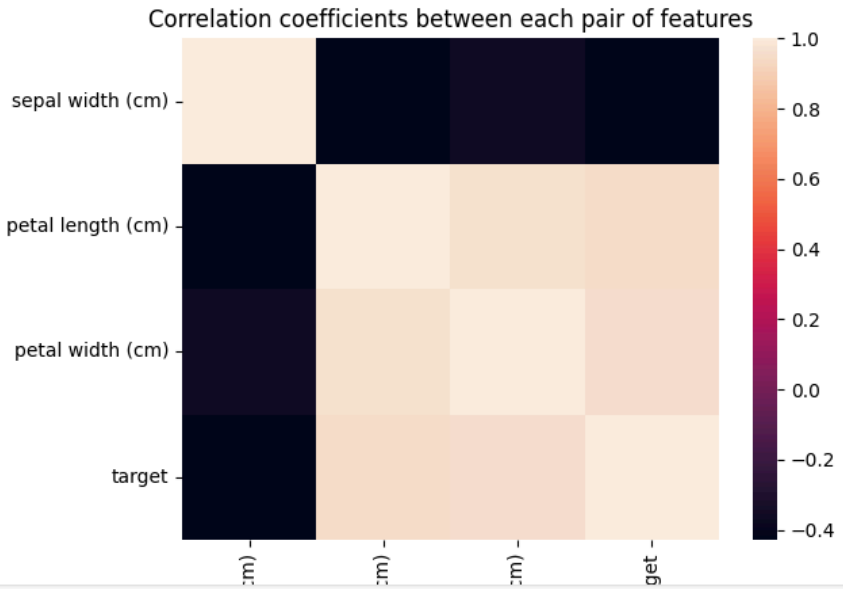

```
#e. Draw heatmap for the four numeric attributes
sns.heatmap(iris_df[iris_df.columns[1:5]])
plt.title('Heatmap for the four numeric attributes')
plt.show()
```



```
#f. Compute mean, mode, median, standard deviation, confidence interval and standard error for each
#feature
print(iris_df.describe())
print(iris_df[iris_df.columns[1:-2]].mode(axis=1))
```

```
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)      target
count        150.000000  150.000000
mean           1.199333    1.000000
std            0.762238    0.819232
min            0.100000    0.000000
25%            0.300000    0.000000
50%            1.300000    1.000000
75%            1.800000    2.000000
max            2.500000    2.000000
```

```
#g. Compute correlation coefficients between each pair of features and plot heatmap
correlation_matrix = (iris_df[iris_df.columns[1:5]].corr())
sns.heatmap(correlation_matrix)
plt.title('Correlation coefficients between each pair of features')
plt.show()
```



PRACTICAL 6

```
#6. Consider the following data frame containing a family name, gender of the family member and her/his monthly
#income in each record.
import pandas as pd
import numpy as np
data = {
    "name":['shah','vats','vats','kumar','vats','kumar','shah','shah','kumar','vats'],
    "Gender":['male','male','female','female','female','male','ma                e','female','male'],
    "MonthlyIncome":[114000.0,65000.0,43150.0,69500.0,155000.0,103000,55000,112400,81030,71900]
}
data1 = pd.DataFrame(data)
#print(data1)

#a. Calculate and display familywise gross monthly income.
familywise = data1.groupby("name")["MonthlyIncome"].sum()
print(familywise)
```

Loading...

```
name
kumar    253530.0
shah     281400.0
vats     335050.0
Name: MonthlyIncome, dtype: float64
```

```
#b. Calculate and display the member with the highest monthly income.
member = data1.loc[data1["MonthlyIncome"].idxmax()]
print(member)
```

```
name                 vats
Gender             female
MonthlyIncome    155000.0
Name: 4, dtype: object
```

```
#c. Calculate and display monthly income of all members with income greater than Rs. 60000.00.
monthly = data1[data1["MonthlyIncome"]>60000]
print(monthly)
```

```
Name: 4, dtype: object
    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
avg_income = data1[data1["Gender"]=="female"]["MonthlyIncome"].mean()
print("avg_income",avg_income)
```

```
avg_income 92216.0
```

PRACTICAL 7

```
#7. Using Titanic dataset, to do the following:
import pandas as pd
import numpy as np

data = pd.read_csv("titanic.csv")
#print(data)

#a. Find total number of passengers with age less than 30
age = data[data["age"]<30]
print("total number of passengers with age less than 30:",len(age))

#b. Find total fare paid by passengers of first class
first = data[data["pclass"]== 1]["fare"].sum()
print(" total fare paid by passengers of first class",first)

#c. Compare number of survivors of each passenger class
survivors  = data.groupby('pclass')['survived'].sum()
print("number of survivors of each passenger class",survivors)

#d. Compute descriptive statistics for any numeric attribute genderwise
statistics = data.groupby('sex')['age'].describe()
print(statistics)
```

```
total number of passengers with age less than 30: 384
 total fare paid by passengers of first class 18177.4125
number of survivors of each passenger class pclass
1    136
2     87
3    119
Name: survived, dtype: int64
        count       mean        std   min   25%   50%   75%   max
sex
female  261.0  27.915709  14.110146  0.75  18.0  27.0  37.0  63.0
male    453.0  30.726645  14.678201  0.42  21.0  29.0  39.0  80.0
```