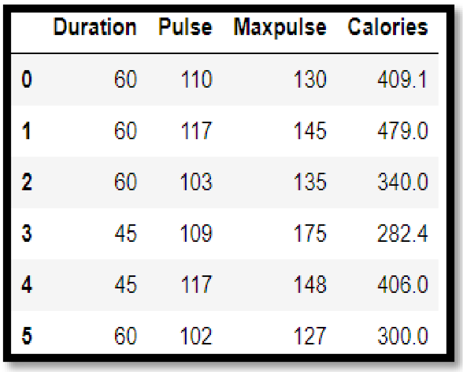
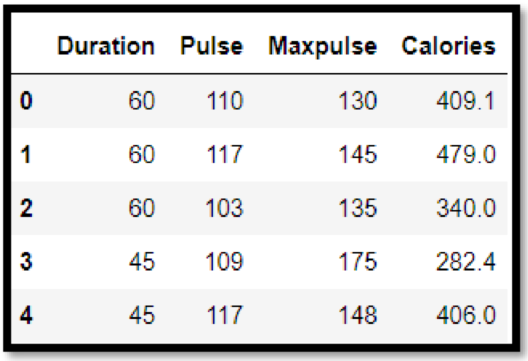
PROGRAM NO. – 1

Write a program in python to read the contents of .csv file using panda module.

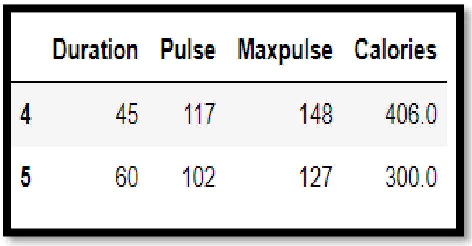
OUTPUT –



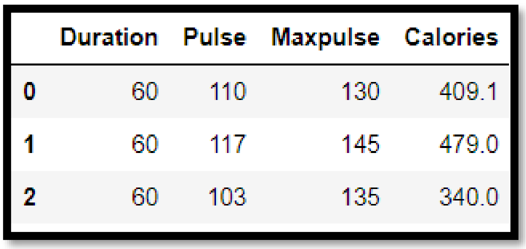
	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0
5	60	102	127	300.0



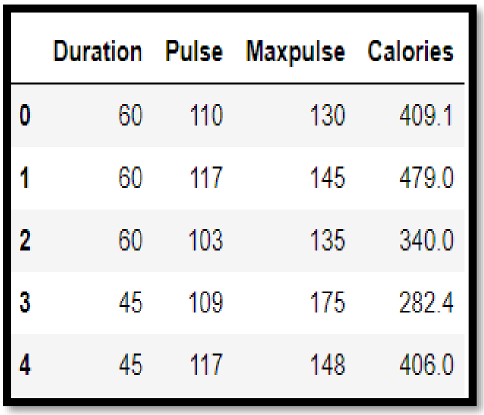
	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0



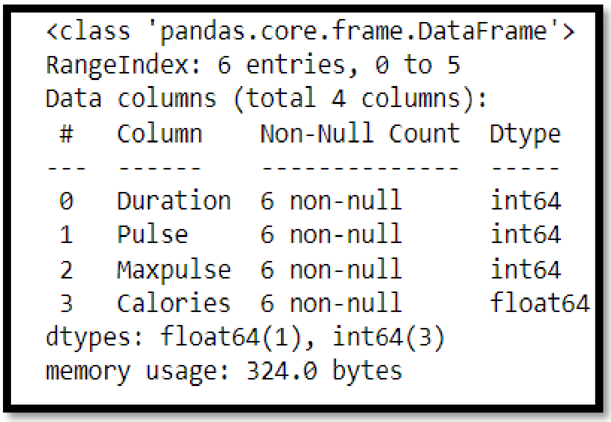
	Duration	Pulse	Maxpulse	Calories
4	45	117	148	406.0
5	60	102	127	300.0



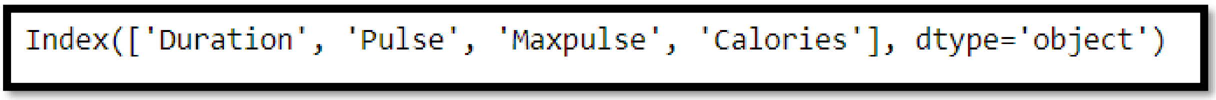
	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0



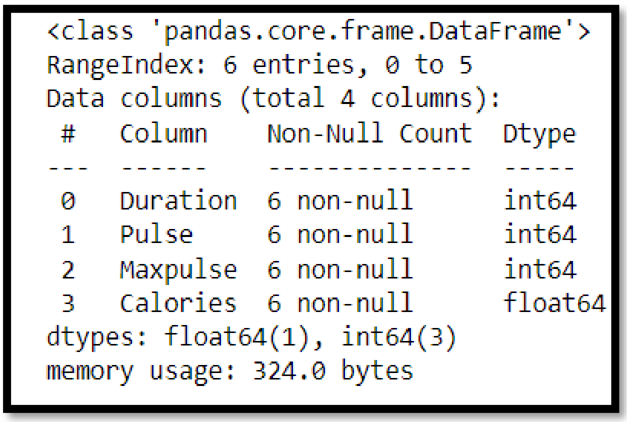
	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0



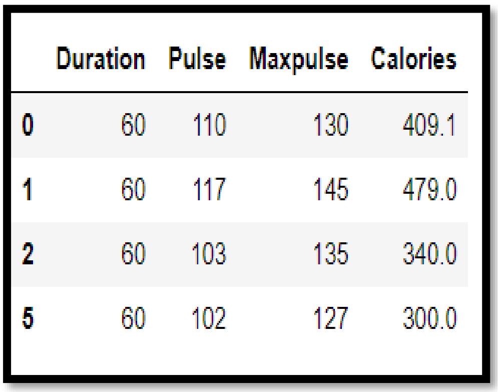
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Duration  6 non-null      int64
 1   Pulse     6 non-null      int64
 2   Maxpulse  6 non-null      int64
 3   Calories  6 non-null      float64
dtypes: float64(1), int64(3)
memory usage: 324.0 bytes
```



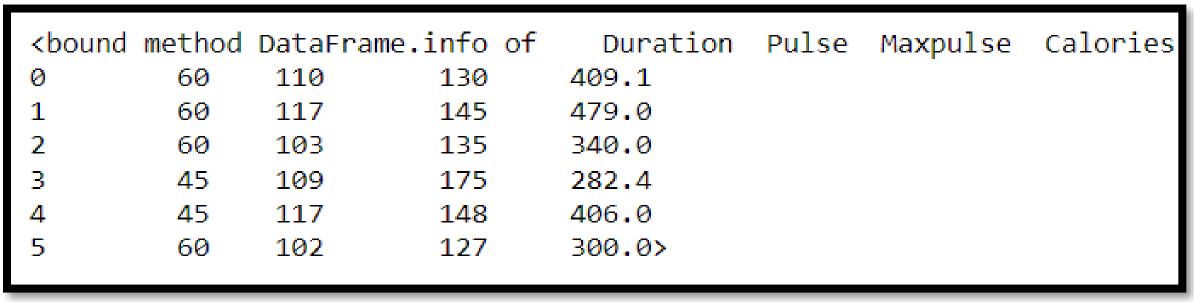
```
Index(['Duration', 'Pulse', 'Maxpulse', 'Calories'], dtype='object')
```



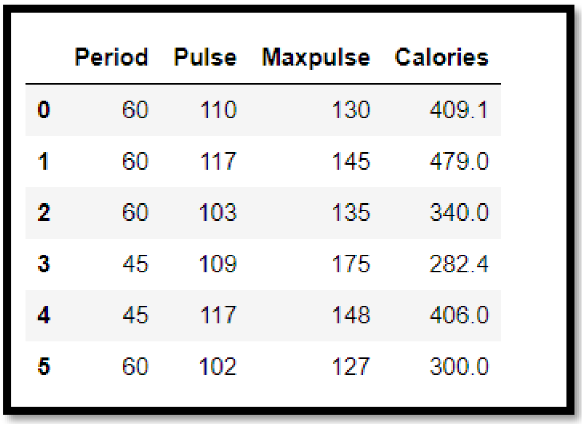
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Duration  6 non-null      int64
 1   Pulse     6 non-null      int64
 2   Maxpulse  6 non-null      int64
 3   Calories  6 non-null      float64
dtypes: float64(1), int64(3)
memory usage: 324.0 bytes
```



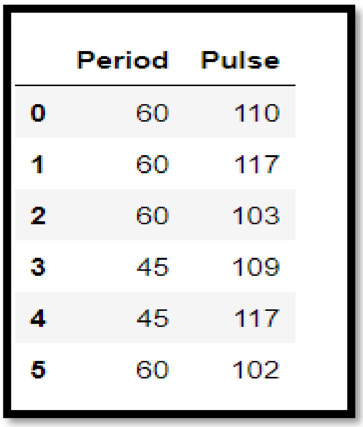
	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
5	60	102	127	300.0



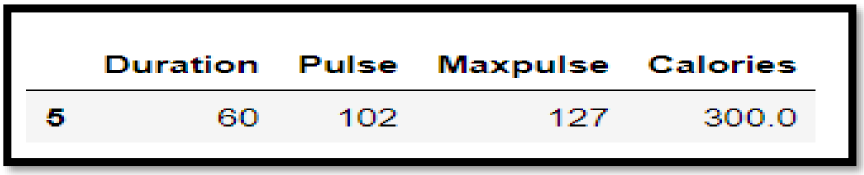
```
<bound method DataFrame.info of    Duration  Pulse  Maxpulse  Calories
0         60    110       130     409.1
1         60    117       145     479.0
2         60    103       135     340.0
3         45    109       175     282.4
4         45    117       148     406.0
5         60    102       127     300.0>
```



	Period	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0
5	60	102	127	300.0



	Period	Pulse
0	60	110
1	60	117
2	60	103
3	45	109
4	45	117
5	60	102



	Duration	Pulse	Maxpulse	Calories
5	60	102	127	300.0

PROGRAM NO. – 1

Write a program in python to read the contents of .csv file using panda module.

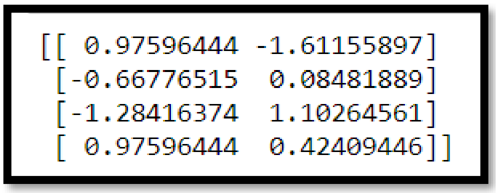
PROGRAM –

```
#prints the dataframe
import pandas as pd
import numpy as np
df = pd.read_csv('data.csv')
#returns first 3 rows of dataframe
df.head(3)

#returns last 2 rows of dataframe
df.tail(2)

#returns first 5 rows of dataframe
df.head()

#returns the columns
df.columns

#prints information about the data
df.info()

#primts the information about datafrmae
df.info

#prints rows with values greater than 50
df[df.Duration>50]

#prints the row with min Pulse value
df[df.Pulse==df.Pulse.min()]

#removes columns
df1 = df.drop(columns=['Calories', 'Maxpulse'])
df1

#renames columns
df = df.rename(columns={'Duration':'Period'})
df
```

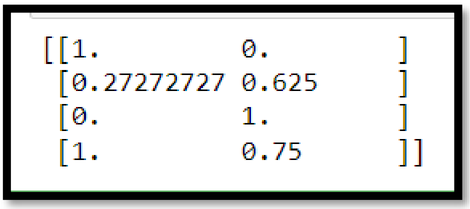
PROGRAM NO. – 2

Write a program for scaling the data using min-max scaling and standard scaling method

OUTPUT –



```
[[ 0.97596444 -1.61155897]
 [-0.66776515  0.08481889]
 [-1.28416374  1.10264561]
 [ 0.97596444  0.42409446]]
```



```
[[1.         0.        ]
 [0.27272727 0.625     ]
 [0.         1.        ]
 [1.         0.75      ]]
```

PROGRAM NO. – 2

Write a program for scaling the data using min-max scaling and standard scaling method

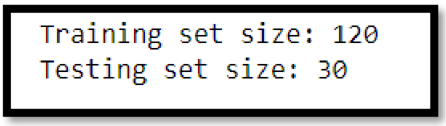
PROGRAM –

```
 # STANDARD SCALING METHOD

# import module
from sklearn.preprocessing import StandardScaler

# create data
data = [[11, 2], [3, 7], [0, 10], [11, 8]]

# compute required values
scaler = StandardScaler()
model = scaler.fit(data)
scaled_data = model.transform(data)

# print scaled data
print(scaled_data)

# MIN-MAX SCALING

# import module
from sklearn.preprocessing import MinMaxScaler

# create data
data = [[11, 2], [3, 7], [0, 10], [11, 8]]

# scale features
scaler = MinMaxScaler()
model=scaler.fit(data)
scaled_data=model.transform(data)

# print scaled features
print(scaled_data)
```

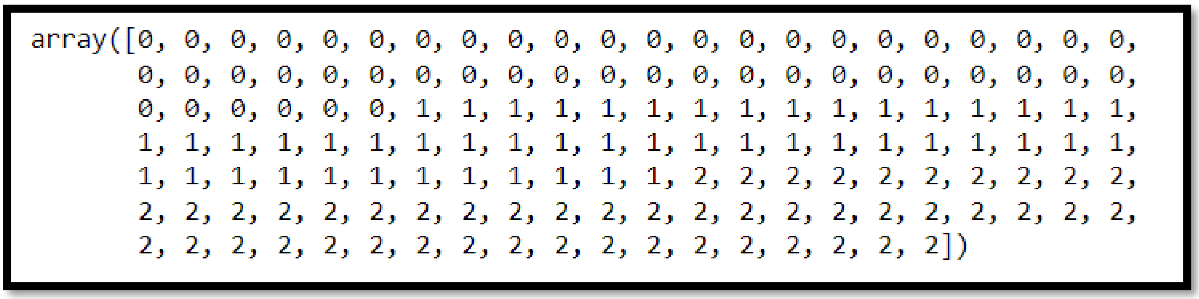
PROGRAM NO. – 3

Using train test split method of module model_selection split the data into training and testing.

OUTPUT –



```
Training set size: 120
Testing set size: 30
```



```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

PROGRAM NO. – 3

Using train test split method of module model_selection split the data into training and testing.

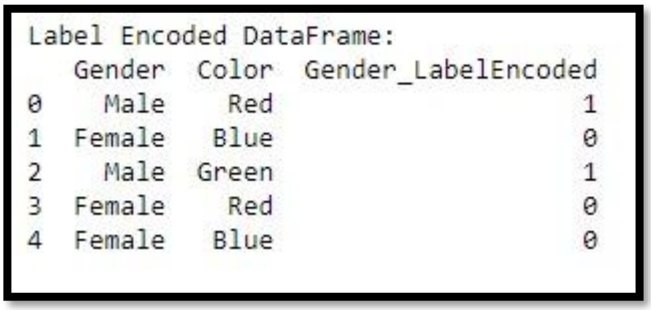
PROGRAM –

```
 from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

#load the iris datesheet
iris = load_iris()

#split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target,
test_size = 0.2, random_state=2)

#print the size of the training and testing sets
print('Training set size:', len(X_train))
print('Testing set size:', len(X_test))


iris.target
```

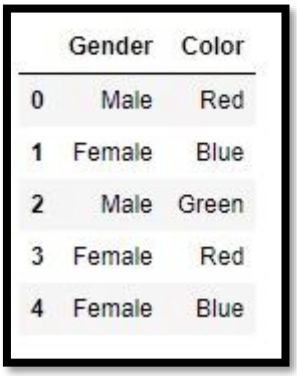
PROGRAM NO. – 4

Write a program to convert the categorical data into numerical values using label encoder, 1-hot encoder.

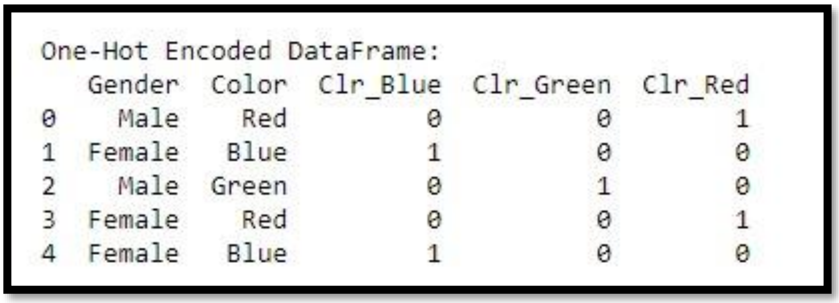
OUTPUT –



```
Label Encoded DataFrame:
   Gender  Color  Gender_LabelEncoded
0    Male    Red                    1
1  Female   Blue                    0
2    Male  Green                    1
3  Female    Red                    0
4  Female   Blue                    0
```



	Gender	Color
0	Male	Red
1	Female	Blue
2	Male	Green
3	Female	Red
4	Female	Blue



```
One-Hot Encoded DataFrame:
   Gender  Color  Clr_Blue  Clr_Green  Clr_Red
0    Male    Red         0          0        1
1  Female   Blue         1          0        0
2    Male  Green         0          1        0
3  Female    Red         0          0        1
4  Female   Blue         1          0        0
```

PROGRAM NO. – 4

Write a program to convert the categorical data into numerical values using label encoder, 1-hot encoder.

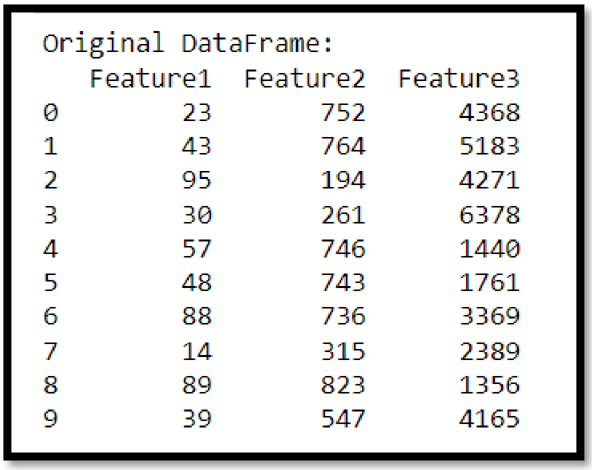
PROGRAM –

```
#LABEL ENCODER

from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
df['Gender_LabelEncoded'] =
label_encoder.fit_transform(df['Gender'])
print("Label Encoded DataFrame:")
print(df)

df2 = df.drop(columns=['Gender_LabelEncoded'])

df2


#ONE-HOT ENCODER

one_hot_encoded = pd.get_dummies(df['Color'], prefix='Clr')
df2 = pd.concat([df2, one_hot_encoded], axis=1)
print("\nOne-Hot Encoded DataFrame:")
print(df2)

df2
```

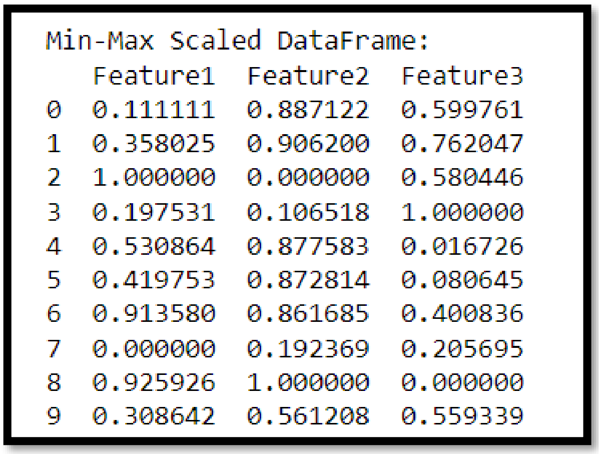
PROGRAM NO. – 5

Write a program to scale the data using robust scaling, max absolute scaling and minmax scaling within range.

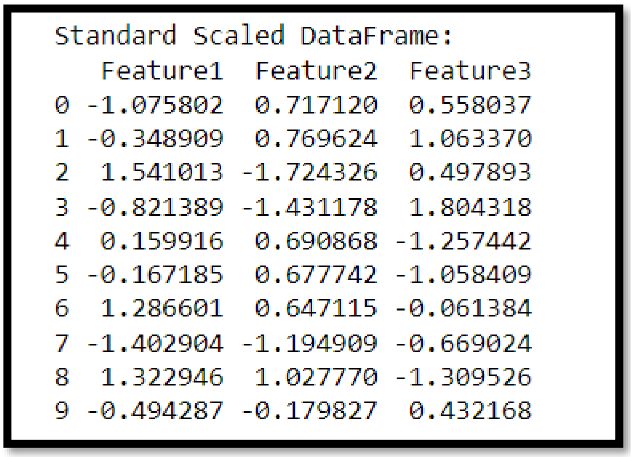
OUTPUT –



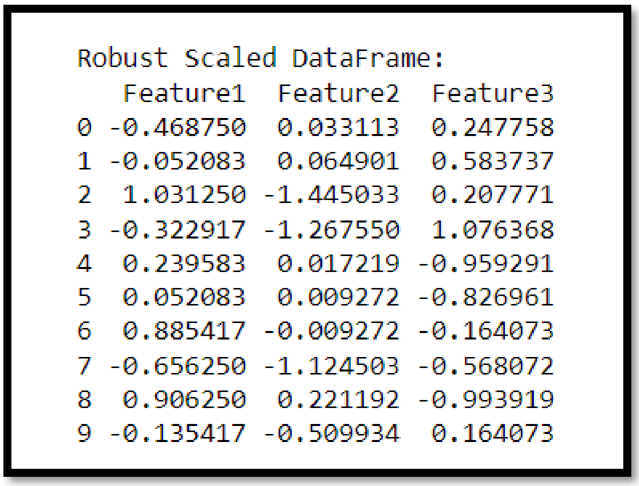
```
Original DataFrame:
   Feature1  Feature2  Feature3
0        23       752      4368
1        43       764      5183
2        95       194      4271
3        30       261      6378
4        57       746      1440
5        48       743      1761
6        88       736      3369
7        14       315      2389
8        89       823      1356
9        39       547      4165
```



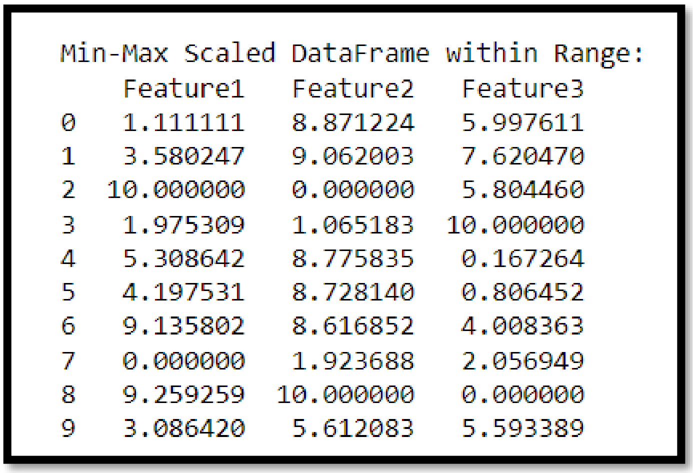
```
Min-Max Scaled DataFrame:
   Feature1  Feature2  Feature3
0  0.111111  0.887122  0.599761
1  0.358025  0.906200  0.762047
2  1.000000  0.000000  0.580446
3  0.197531  0.106518  1.000000
4  0.530864  0.877583  0.016726
5  0.419753  0.872814  0.080645
6  0.913580  0.861685  0.400836
7  0.000000  0.192369  0.205695
8  0.925926  1.000000  0.000000
9  0.308642  0.561208  0.559339
```



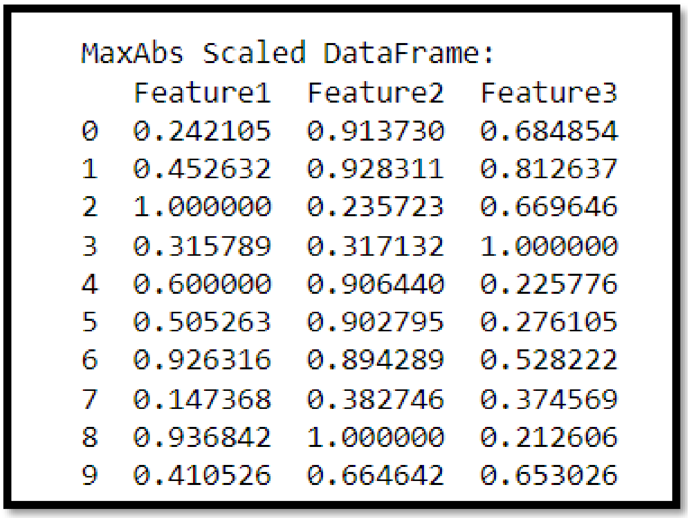
```
Standard Scaled DataFrame:
   Feature1  Feature2  Feature3
0 -1.075802  0.717120  0.558037
1 -0.348909  0.769624  1.063370
2  1.541013 -1.724326  0.497893
3 -0.821389 -1.431178  1.804318
4  0.159916  0.690868 -1.257442
5 -0.167185  0.677742 -1.058409
6  1.286601  0.647115 -0.061384
7 -1.402904 -1.194909 -0.669024
8  1.322946  1.027770 -1.309526
9 -0.494287 -0.179827  0.432168
```



```
Robust Scaled DataFrame:
   Feature1  Feature2  Feature3
0 -0.468750  0.033113  0.247758
1 -0.052083  0.064901  0.583737
2  1.031250 -1.445033  0.207771
3 -0.322917 -1.267550  1.076368
4  0.239583  0.017219 -0.959291
5  0.052083  0.009272 -0.826961
6  0.885417 -0.009272 -0.164073
7 -0.656250 -1.124503 -0.568072
8  0.906250  0.221192 -0.993919
9 -0.135417 -0.509934  0.164073
```



```
Min-Max Scaled DataFrame within Range:
    Feature1   Feature2   Feature3
0   1.111111   8.871224   5.997611
1   3.580247   9.062003   7.620470
2  10.000000   0.000000   5.804460
3   1.975309   1.065183  10.000000
4   5.308642   8.775835   0.167264
5   4.197531   8.728140   0.806452
6   9.135802   8.616852   4.008363
7   0.000000   1.923688   2.056949
8   9.259259  10.000000   0.000000
9   3.086420   5.612083   5.593389
```



```
MaxAbs Scaled DataFrame:
   Feature1  Feature2  Feature3
0  0.242105  0.913730  0.684854
1  0.452632  0.928311  0.812637
2  1.000000  0.235723  0.669646
3  0.315789  0.317132  1.000000
4  0.600000  0.906440  0.225776
5  0.505263  0.902795  0.276105
6  0.926316  0.894289  0.528222
7  0.147368  0.382746  0.374569
8  0.936842  1.000000  0.212606
9  0.410526  0.664642  0.653026
```

PROGRAM NO. – 5

Write a program to scale the data using robust scaling, max absolute scaling and minmax scaling within range.

PROGRAM –

```
#ORIGINAL DATAFRAME

import pandas as pd
import numpy as np

data = {
   'Feature1' : np.random.randint(1,100,10),
   'Feature2' : np.random.randint(100,1000,10),
   'Feature3' : np.random.randint(1000,10000,10)
}

df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)


#MinMaxScaler

from sklearn.preprocessing import MinMaxScaler
minmax_scaler = MinMaxScaler()
minmax_scaled = minmax_scaler.fit_transform(df)
minmax_df = pd.DataFrame(minmax_scaled, columns=df.columns)
print("\nMin-Max Scaled DataFrame:")
print(minmax_df)
```

```
#Standardisation or z-scrore

from sklearn.preprocessing import StandardScaler
standard_scaler = StandardScaler()
standard_scaled = standard_scaler.fit_transform(df)
standard_df = pd.DataFrame(standard_scaled, columns=df.columns)
print("\nStandard Scaled DataFrame:")
print(standard_df)


#Robust Scaling

from sklearn.preprocessing import RobustScaler
robust_scaler = RobustScaler()
robust_scaled = robust_scaler.fit_transform(df)
robust_df = pd.DataFrame(robust_scaled, columns=df.columns)
print("\nRobust Scaled DataFrame:")
print(robust_df)


#Max Absolute Scaling

from sklearn.preprocessing import MaxAbsScaler
maxabs_scaler = MaxAbsScaler()
maxabs_scaled = maxabs_scaler.fit_transform(df)
maxabs_df = pd.DataFrame(maxabs_scaled, columns=df.columns)
print("\nMaxAbs Scaled DataFrame:")
print(maxabs_df)
```

```
#MinMax Scaler within range

minmax_range = (0,10)
minmax_range_scaler =
MinMaxScaler(feature_range=minmax_range)
minmax_range_scaled = minmax_range_scaler.fit_transform(df)
minmax_range_df = pd.DataFrame(minmax_range_scaled,
columns=df.columns)
print("\nMin-Max Scaled DataFrame within Range:")
print(minmax_range_df)
```