In [21]:

```python
# Q1 Import Dataset and do the followings:
# a) Describing the dataset
# b) Shape of the dataset
# c) Display first 3 rows from dataset
import pandas as pd
df = pd.read_csv('data.csv')
print('DataFrame')
print('---------')
print(df)
print('\n\nStatistical Description')
print('------------------------')
print(df.describe())
print('\n\nNumber of Rows * Columns :',df.shape)
print("\n\nFirst 3 records")
print("------------------")
print(df.head(3))
```

```
DataFrame
---------
   Country   Age   Salary Purchased
0   France  44.0  72000.0        No
1    Spain  27.0  48000.0       Yes
2  Germany  30.0  54000.0        No
3    Spain  38.0  61000.0        No
4  Germany  40.0      NaN       Yes
5   France  35.0  58000.0       Yes
6    Spain   NaN  52000.0        No
7   France  48.0  79000.0       Yes
8  Germany  50.0  83000.0        No
9   France  37.0  67000.0       Yes


Statistical Description
------------------------
             Age        Salary
count   9.000000      9.000000
mean   38.777778  63777.777778
std     7.693793  12265.579662
min    27.000000  48000.000000
25%    35.000000  54000.000000
50%    38.000000  61000.000000
75%    44.000000  72000.000000
max    50.000000  83000.000000


Number of Rows * Columns : (10, 4)


First 3 records
------------------
   Country   Age   Salary Purchased
0   France  44.0  72000.0        No
1    Spain  27.0  48000.0       Yes
2  Germany  30.0  54000.0        No
```

In [22]:

```python
# Q2 Handling Missing Value:
# a) Replace missing value of salary,age column with mean of that column.
agemean = df['Age'].mean()
salarymean = df['Salary'].mean()
df['Age'].fillna(agemean, inplace=True)
df['Salary'].fillna(salarymean, inplace=True)
print("\n\n Missing Values for Age and Salary Replaced with MeanValue:")
df
```

```
 Missing Values for Age and Salary Replaced with MeanValue:
```

Out[22]:

	Country	Age	Salary	Purchased
0	France	44.000000	72000.000000	No
1	Spain	27.000000	48000.000000	Yes
2	Germany	30.000000	54000.000000	No
3	Spain	38.000000	61000.000000	No
4	Germany	40.000000	63777.777778	Yes
5	France	35.000000	58000.000000	Yes
6	Spain	38.777778	52000.000000	No
7	France	48.000000	79000.000000	Yes
8	Germany	50.000000	83000.000000	No
9	France	37.000000	67000.000000	Yes

In [23]:

```python
# Q3 Data.csv have two categorical column (the country column, and the purcha
# a. Apply OneHot coding on Country column.
# b. Apply Label encoding on purchased column
from sklearn.preprocessing import OneHotEncoder,LabelEncoder
hotencoder = OneHotEncoder()
enc_df = pd.DataFrame(hotencoder.fit_transform(df[['Country']]).toarray())
df = df.join(enc_df)
print('\n\nOneHot encoding on Country Column')
print('---------------------------')
print(df)
labelencoder = LabelEncoder()
df['Purchased'] = labelencoder.fit_transform(df['Purchased'])
print('\n\nLabel encoding on Purchased Column')
print(df)
```

```
OneHot encoding on Country Column
---------------------------
   Country        Age        Salary Purchased    0    1    2
0   France  44.000000  72000.000000        No  1.0  0.0  0.0
1    Spain  27.000000  48000.000000       Yes  0.0  0.0  1.0
2  Germany  30.000000  54000.000000        No  0.0  1.0  0.0
3    Spain  38.000000  61000.000000        No  0.0  0.0  1.0
4  Germany  40.000000  63777.777778       Yes  0.0  1.0  0.0
5   France  35.000000  58000.000000       Yes  1.0  0.0  0.0
6    Spain  38.777778  52000.000000        No  0.0  0.0  1.0
7   France  48.000000  79000.000000       Yes  1.0  0.0  0.0
8  Germany  50.000000  83000.000000        No  0.0  1.0  0.0
9   France  37.000000  67000.000000       Yes  1.0  0.0  0.0


Label encoding on Purchased Column
   Country        Age        Salary  Purchased    0    1    2
0   France  44.000000  72000.000000          0  1.0  0.0  0.0
1    Spain  27.000000  48000.000000          1  0.0  0.0  1.0
2  Germany  30.000000  54000.000000          0  0.0  1.0  0.0
3    Spain  38.000000  61000.000000          0  0.0  0.0  1.0
4  Germany  40.000000  63777.777778          1  0.0  1.0  0.0
5   France  35.000000  58000.000000          1  1.0  0.0  0.0
6    Spain  38.777778  52000.000000          0  0.0  0.0  1.0
7   France  48.000000  79000.000000          1  1.0  0.0  0.0
8  Germany  50.000000  83000.000000          0  0.0  1.0  0.0
9   France  37.000000  67000.000000          1  1.0  0.0  0.0
```

In [25]:

```
# Q1 Import Dataset and create DataFrame
import pandas as pd
import scipy.stats as s
from sklearn import preprocessing
df = pd.read_csv('winequality-red.csv')
df
```

Out[25]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	a
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	
...	...	...	...	...	...	...	...	...	...	...	
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	

1599 rows × 12 columns

In [26]:

```python
# Q2 Rescaling: Normalised the dataset using MinMaxScaler class
print("\n\n Data Scaled Between 0 to 1")
data_scaler = preprocessing.MinMaxScaler(feature_range = (0, 1))
data_scaled = data_scaler.fit_transform(df)
print("\n Min Max Scaled Data ")
print("-----------------------")
print(data_scaled.round(2))
```

```
 Data Scaled Between 0 to 1

 Min Max Scaled Data 
-----------------------
[[0.25 0.4  0.   ... 0.14 0.15 0.4 ]
 [0.28 0.52 0.   ... 0.21 0.22 0.4 ]
 [0.28 0.44 0.04 ... 0.19 0.22 0.4 ]
 ...
 [0.15 0.27 0.13 ... 0.25 0.4  0.6 ]
 [0.12 0.36 0.12 ... 0.23 0.28 0.4 ]
 [0.12 0.13 0.47 ... 0.2  0.4  0.6 ]]
```

In [28]:
```python
# Q3 Standardizing Data (transform them into a standard Gaussian distribution
# with a mean of 0 and a standard deviation of 1)
print(" Orginal Data \n")
print(df)
print("\n Initial Mean : ", s.tmean(df).round(2))
print("\n\n Initial Standard Deviation :")
print("--------------------------")
print(round(df.std(),2))
df_scaled = preprocessing.scale(df)
df_scaled.mean(axis=0)
df_scaled.std(axis=0)
print("\n Standardized Data \n", df_scaled.round(2))
print("\n Scaled Mean : ",s.tmean(df_scaled).round(2))
print(" Scaled Standard Deviation : ",round(df_scaled.std(),2))
```

```
 Orginal Data

      fixed acidity  volatile acidity  citric acid  residual sugar  chlorid
es  \
0               7.4             0.700         0.00             1.9       0.0
76
1               7.8             0.880         0.00             2.6       0.0
98
2               7.8             0.760         0.04             2.3       0.0
92
3              11.2             0.280         0.56             1.9       0.0
75
4               7.4             0.700         0.00             1.9       0.0
76
...             ...               ...          ...             ...
...
1594            6.2             0.600         0.08             2.0       0.0
90
1595            5.9             0.550         0.10             2.2       0.0
62
1596            6.3             0.510         0.13             2.3       0.0
76
1597            5.9             0.645         0.12             2.0       0.0
75
1598            6.0             0.310         0.47             3.6       0.0
67

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates
\
0                    11.0                  34.0  0.99780  3.51       0.56
1                    25.0                  67.0  0.99680  3.20       0.68
2                    15.0                  54.0  0.99700  3.26       0.65
3                    17.0                  60.0  0.99800  3.16       0.58
4                    11.0                  34.0  0.99780  3.51       0.56
...                   ...                   ...      ...   ...        ...
1594                 32.0                  44.0  0.99490  3.45       0.58
1595                 39.0                  51.0  0.99512  3.52       0.76
1596                 29.0                  40.0  0.99574  3.42       0.75
1597                 32.0                  44.0  0.99547  3.57       0.71
1598                 18.0                  42.0  0.99549  3.39       0.66
```

```
      alcohol  quality
0         9.4        5
1         9.8        5
2         9.8        5
3         9.8        6
4         9.4        5
...       ...      ...
1594     10.5        5
1595     11.2        6
1596     11.0        6
1597     10.2        5
1598     11.0        6

[1599 rows x 12 columns]

 Initial Mean :  7.93


 Initial Standard Deviation :
--------------------------
fixed acidity            1.74
volatile acidity         0.18
citric acid              0.19
residual sugar           1.41
chlorides                0.05
free sulfur dioxide     10.46
total sulfur dioxide    32.90
density                  0.00
pH                       0.15
sulphates                0.17
alcohol                  1.07
quality                  0.81
dtype: float64

 Standardized Data
 [[-0.53  0.96 -1.39 ... -0.58 -0.96 -0.79]
 [-0.3   1.97 -1.39 ...  0.13 -0.58 -0.79]
 [-0.3   1.3  -1.19 ... -0.05 -0.58 -0.79]
 ...
 [-1.16 -0.1  -0.72 ...  0.54  0.54  0.45]
 [-1.39  0.65 -0.78 ...  0.31 -0.21 -0.79]
 [-1.33 -1.22  1.02 ...  0.01  0.54  0.45]]

 Scaled Mean :  -0.0
 Scaled Standard Deviation :  1.0
```

In [30]:

```python
# Q4 Normalizing Data ( rescale each observation to a length of 1 (a unit nor
# For this, use the Normalizer class.)
dn = preprocessing.normalize(df, norm = 'l1')
print("\n L1 Normalized Data ")
print(" ----------------------")
print(dn.round(2))
```

```
 L1 Normalized Data
 ----------------------
[[0.1  0.01 0.   ... 0.01 0.13 0.07]
 [0.06 0.01 0.   ... 0.01 0.08 0.04]
 [0.08 0.01 0.   ... 0.01 0.1  0.05]
 ...
 [0.06 0.01 0.   ... 0.01 0.11 0.06]
 [0.06 0.01 0.   ... 0.01 0.1  0.05]
 [0.06 0.   0.01 ... 0.01 0.12 0.06]]
```

In [33]:

```python
# Q5 Binarizing Data using we use the Binarizer class (Using a binary
# threshold, it is possible to transform our data by marking the values
# above it 1 and those equal to or below it, 0)
data_binarized = preprocessing.Binarizer(threshold=5).transform(df)
print("\n Binarized data ")
print(" -----------------")
print(data_binarized)
```

```
 Binarized data
 -----------------
[[1. 0. 0. ... 0. 1. 0.]
 [1. 0. 0. ... 0. 1. 0.]
 [1. 0. 0. ... 0. 1. 0.]
 ...
 [1. 0. 0. ... 0. 1. 1.]
 [1. 0. 0. ... 0. 1. 0.]
 [1. 0. 0. ... 0. 1. 1.]]
```

In [37]:

```python
# Q1
import pandas as pd
import numpy as np
df = pd.read_csv('student_bucketing.csv')
print("\n ORIGINAL DATASET")
print(" ----------------")
print(df)
#Creating bins
m1=min(df["marks"])
m2=max(df["marks"])
# Bin labels must be one fewer than the number of bin edges
bins=np.linspace(m1,m2,6) # bin edges
names=["Poor", "Below_average", "Average","Above_Average","Excellent"]
df["marks_bin"]=pd.cut(df["marks"],bins,labels=names,include_lowest=True)
print("\n BINNED DATASET")
print(" ----------------")
df.sample(15)
```

```
 ORIGINAL DATASET
 ----------------
     Student_id  Age      Grade Employed  marks
0             1   19  1st Class      yes     29
1             2   20  2nd Class       no     41
2             3   18  1st Class       no     57
3             4   21  2nd Class       no     29
4             5   19  1st Class       no     57
..          ...  ...        ...      ...    ...
227         228   21  1st Class       no     42
228         229   20  2nd Class       no     47
229         230   20  3rd Class      yes     21
230         231   19  1st Class      yes     64
231         232   20  3rd Class      yes     30

[232 rows x 5 columns]

 BINNED DATASET
 ----------------
```

Out[37]:

	Student_id	Age	Grade	Employed	marks	marks_bin
103	104	20	2nd Class	yes	53	Average
104	105	18	3rd Class	no	71	Above_Average
214	215	20	1st Class	no	35	Poor
106	107	19	2nd Class	no	74	Above_Average
225	226	21	2nd Class	yes	70	Above_Average
68	69	20	3rd Class	yes	81	Above_Average
7	8	21	3rd Class	yes	70	Above_Average
63	64	20	2nd Class	yes	20	Poor
205	206	18	1st Class	no	68	Above_Average
89	90	18	1st Class	no	35	Poor

	Student_id	Age	Grade	Employed	marks	marks_bin
81	82	22	3rd Class	no	31	Poor
203	204	19	1st Class	yes	31	Poor
120	121	19	1st Class	no	25	Poor
142	143	20	3rd Class	no	48	Below_average
229	230	20	3rd Class	yes	21	Poor

In [ ]: