IMPORTING LIBRARIES

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

PRESENTED BY SANJU DEBNATH

DATA LOADING

```
path = ('/content/BostonHousing.csv')
```

```
df = pd.read_csv(path)
df = df.rename(columns={'crim':'CRIM', 'zn':'ZN', 'indus':'INDUS', 'chas':'CHAS', 'nox':'NOX',
                        'rm':'RM', 'age':'AGE','dis': 'DIS', 'rad':'RAD', 'tax':'TAX', 'ptratio':'PTRATIO', 'b':'B', 'lstat':'LSTAT', 'medv':
df.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

DESCRIPTIONS

CRIM: per capita crime rate by town

ZN: proportion of residential land zoned for lots over 25,000 sq.ft.

INDUS: proportion of non-retail business acres per town

CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

NOX: nitric oxides concentration (parts per 10 million)

RM: average number of rooms per dwelling

AGE: proportion of owner-occupied units built prior to 1940

DIS: weighted distances to five Boston employment centers

RAD: index of accessibility to radial highways

TAX: full-value property-tax rate per $10,000

PTRATIO: pupil-teacher ratio by town 12.

B: 1000(Bk−0.63)2 where Bk is the proportion of blacks by town 13.

LSTAT: % lower status of the population

MEDV: Median value of owner-occupied homes in $1000s

DATA EXPLORATION

NO OF ROWS AND COLUMNS

```
df.shape
```

```
(506, 14)
```

COLUMN NAMES

```
df.columns
```

```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

DATA TYPES

```
df.dtypes
```

```
CRIM       float64
ZN         float64
INDUS      float64
CHAS         int64
NOX        float64
RM         float64
AGE        float64
DIS        float64
RAD          int64
TAX          int64
PTRATIO    float64
B          float64
LSTAT      float64
MEDV       float64
dtype: object
```

OVERVIEW

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    int64  
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64  
 9   TAX      506 non-null    int64  
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

SUMMARY STATISTICS

```
df.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000

OBSERVATION

- In 50% data age is 77 it mean data is in stracture from

CHEKING NULL VALUES

```
df.isnull().sum()
```

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

OBSERVATION

- We can see that their is no null value.

CHECKING DUPLICATES

```
df.duplicated().sum()
```

```
0
```

OBSERVATION

- Their is no duplicate values

FINDING THE CO-RELATIONS

```
df.corr()['AGE']
```

```
CRIM       0.352734
ZN        -0.569537
INDUS      0.644779
CHAS       0.086518
NOX        0.731470
RM        -0.240265
AGE        1.000000
DIS       -0.747881
RAD        0.456022
TAX        0.506456
PTRATIO    0.261515
B         -0.273534
LSTAT      0.602339
MEDV      -0.376955
Name: AGE, dtype: float64
```

DATA MANIPULATION

1. Add a new column to the DataFrame that categorizes the ‘medv’ (median value of owner-occupied homes) column into ‘Low’, ‘Medium’, and ‘High’.

```
df["MEDV STATUS"]=pd.cut(df["MEDV"],bins=[0,20,35,50],labels=["Low","Medium","High"])
df.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV	MEDV STATUS
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0	Medium
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6	Medium
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7	Medium
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4	Medium
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2	High

2. Replace the ‘chas’ column (Charles River dummy variable) with ‘Yes’ if chas = 1 and ‘No’ if chas = 0.

```
df['CHAS']=df['CHAS'].map({1:"YES",0:"NO"})
df.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV	MEDV STATUS
0	0.00632	18.0	2.31	NO	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0	Medium
1	0.02731	0.0	7.07	NO	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6	Medium
2	0.02729	0.0	7.07	NO	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7	Medium
3	0.03237	0.0	2.18	NO	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4	Medium
4	0.06905	0.0	2.18	NO	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2	High

3. Rename the column ‘rm’ to ‘rooms’.

```
df = df.rename(columns={'RM':'ROOMS'})
df.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	ROOMS	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV	MEDV STATUS
0	0.00632	18.0	2.31	NO	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0	Medium
1	0.02731	0.0	7.07	NO	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6	Medium
2	0.02729	0.0	7.07	NO	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7	Medium
3	0.03237	0.0	2.18	NO	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4	Medium
4	0.06905	0.0	2.18	NO	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2	High

DATA ANALYSIS

1. Find out the average number of rooms (‘rooms’ column) per dwelling for each category of ‘medv’ (Low, Medium, High).

```
avg = df.groupby('MEDV STATUS')
avg = avg[['ROOMS']].mean()
avg = avg.sort_values('ROOMS',ascending=False)
avg
```

MEDV STATUS	ROOMS
High	7.503000
Medium	6.371259
Low	5.914721

2. Find out the percentage of houses that bound the Charles River.

```
per = df['CHAS'].value_counts(normalize=True)*100
per
```

```
NO     93.083004
YES     6.916996
Name: CHAS, dtype: float64
```

NumPy Operations

```
arr = df['AGE']
np.array(arr)
```

```
array([ 65.2,  78.9,  61.1,  45.8,  54.2,  58.7,  66.6,  96.1, 100. ,
        85.9,  94.3,  82.9,  39. ,  61.8,  84.5,  56.5,  29.3,  81.7,
        36.6,  69.5,  98.1,  89.2,  91.7, 100. ,  94.1,  85.7,  90.3,
        88.8,  94.4,  87.3,  94.1, 100. ,  82. ,  95. ,  96.9,  68.2,
        61.4,  41.5,  30.2,  21.8,  15.8,   2.9,   6.6,   6.5,  40. ,
        33.8,  33.3,  85.5,  95.3,  62. ,  45.7,  63. ,  21.1,  21.4,
        47.6,  21.9,  35.7,  40.5,  29.2,  47.2,  66.2,  93.4,  67.8,
        43.4,  59.5,  17.8,  31.1,  21.4,  36.8,  33. ,   6.6,  17.5,
         7.8,   6.2,   6. ,  45. ,  74.5,  45.8,  53.7,  36.6,  33.5,
        70.4,  32.2,  46.7,  48. ,  56.1,  45.1,  56.8,  86.3,  63.1,
        66.1,  73.9,  53.6,  28.9,  77.3,  57.8,  69.6,  76. ,  36.9,
        62.5,  79.9,  71.3,  85.4,  87.4,  90. ,  96.7,  91.9,  85.2,
        97.1,  91.2,  54.4,  81.6,  92.9,  95.4,  84.2,  88.2,  72.5,
        82.6,  73.1,  65.2,  69.7,  84.1,  92.9,  97. ,  95.8,  88.4,
        95.6,  96. ,  98.8,  94.7,  98.9,  97.7,  97.9,  95.4,  98.4,
        98.2,  93.5,  98.4,  98.2,  97.9,  93.6, 100. , 100. , 100. ,
        97.8, 100. , 100. ,  95.7,  93.8,  94.9,  97.3, 100. ,  88. ,
        98.5,  96. ,  82.6,  94. ,  97.4, 100. , 100. ,  92.6,  90.8,
        98.2,  93.9,  91.8,  93. ,  96.2,  79.2,  96.1,  95.2,  94.6,
        97.3,  88.5,  84.1,  68.7,  33.1,  47.2,  73.4,  74.4,  58.4,
        83.3,  62.2,  92.2,  95.6,  89.8,  68.8,  53.6,  41.1,  29.1,
        38.9,  21.5,  30.8,  26.3,   9.9,  18.8,  32. ,  34.1,  36.6,
        38.3,  15.3,  13.9,  38.4,  15.7,  33.2,  31.9,  22.3,  52.5,
        72.7,  59.1, 100. ,  92.1,  88.6,  53.8,  32.3,   9.8,  42.4,
        56. ,  85.1,  93.8,  92.4,  88.5,  91.3,  77.7,  80.8,  78.3,
        83. ,  86.5,  79.9,  17. ,  21.4,  68.1,  76.9,  73.3,  70.4,
        66.5,  61.5,  76.5,  71.6,  18.5,  42.2,  54.3,  65.1,  52.9,
         7.8,  76.5,  70.2,  34.9,  79.2,  49.1,  17.5,  13. ,   8.9,
         6.8,   8.4,  32. ,  19.1,  34.2,  86.9, 100. , 100. ,  81.8,
        89.4,  91.5,  94.5,  91.6,  62.8,  84.6,  67. ,  52.6,  61.5,
        42.1,  16.3,  58.7,  51.8,  32.9,  42.8,  49. ,  27.6,  32.1,
        32.2,  64.5,  37.2,  49.7,  24.8,  20.8,  31.9,  31.5,  31.3,
        45.6,  22.9,  27.9,  27.7,  23.4,  18.4,  42.3,  31.1,  51. ,
        58. ,  20.1,  10. ,  47.4,  40.4,  18.4,  17.7,  41.1,  58.1,
        71.9,  70.3,  82.5,  76.7,  37.8,  52.8,  90.4,  82.8,  87.3,
        77.7,  83.2,  71.7,  67.2,  58.8,  52.3,  54.3,  49.9,  74.3,
        40.1,  14.7,  28.9,  43.7,  25.8,  17.2,  32.2,  28.4,  23.3,
        38.1,  38.5,  34.5,  46.3,  59.6,  37.3,  45.4,  58.5,  49.3,
        59.7,  56.4,  28.1,  48.5,  52.3,  27.7,  29.7,  34.5,  44.4,
        35.9,  18.5,  36.1,  21.9,  19.5,  97.4,  91. ,  83.4,  81.3,
        88. ,  91.1,  96.2,  89. ,  82.9,  87.9,  91.4, 100. , 100. ,
        96.8,  97.5, 100. ,  89.6, 100. , 100. ,  97.9,  93.3,  98.8,
        96.2, 100. ,  91.9,  99.1, 100. , 100. ,  91.2,  98.1, 100. ,
        89.5, 100. ,  98.9,  97. ,  82.5,  97. ,  92.6,  94.7,  98.8,
        96. ,  98.9, 100. ,  77.8, 100. , 100. , 100. ,  96. ,  85.4,
       100. , 100. , 100. ,  97.9, 100. , 100. , 100. , 100. , 100. ,
       100. , 100. ,  90.8,  89.1, 100. ,  76.5, 100. ,  95.3,  87.6,
        85.1,  70.6,  95.4,  59.7,  78.7,  78.1,  95.6,  86.1,  94.3,
        74.8,  87.9,  95. ,  94.6,  93.3, 100. ,  87.9,  93.9,  92.4,
        97.2, 100. , 100. ,  96.6,  94.8,  96.4,  96.6,  98.7,  98.3,
        92.6,  98.2,  91.8,  99.3,  94.1,  86.5,  87.9,  80.3,  83.7,
        84.4,  90. ,  88.4,  83. ,  89.9,  65.4,  48.2,  84.7,  94.5,
        71. ,  56.7,  84. ,  90.7,  75. ,  67.6,  95.4,  97.4,  93.6,
        97.3,  96.7,  88. ,  64.7,  74.9,  77. ,  40.3,  41.9,  51.9,
        79.8,  53.2,  92.7,  98.3,  98. ,  98.8,  83.5,  54. ,  42.6,
        28.8,  72.9,  70.6,  65.3,  73.5,  79.7,  69.1,  76.7,  91. ,
        89.3,  80.8])
```

Compute the mean and standard deviation.

```
print('MEAN OF THE AGE ARRAY : ',np.mean(arr))
print('STANDARD DEVIATION OF THE AGE ARRAY : ',np.std(arr))
```

```
MEAN OF THE AGE ARRAY :  68.57490118577076
STANDARD DEVIATION OF THE AGE ARRAY :  28.121032570236867
```

Normalize the array.

```
normalized_age = (arr - np.min(arr)) / (np.max(arr) - np.min(arr))
normalized_age.head(10)
```

```
0    0.641607
1    0.782698
2    0.599382
3    0.441813
4    0.528321
5    0.574665
6    0.656025
7    0.959835
8    1.000000
9    0.854789
Name: AGE, dtype: float64
```