Boston Housing Dataset Cleaning Assignment

Name: Tanzeel Ahmad

```
from google.colab import files
uploaded = files.upload()
import pandas as pd
import numpy as np
#load Data set
df = pd.read_csv(list(uploaded.keys())[0])
# Show first rows
df.head() # showing first five rows known as headers.

```

Choose files boston_housing.csv

- **boston_housing.csv**(text/csv) - 35200 bytes, last modified: 30/08/2025 - 100% done

```
Saving boston_housing.csv to boston_housing.csv
```

	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

Next steps: Generate code with df View recommended plots New interactive sheet

CRIM ZN INDUS CHAS CRIM vs ZN

ZN vs INDUS INDUS vs CHAS CHAS vs NOX CRIM ZN

INDUS CHAS

```
# Step 2: Inspect data
print("Data Types:\n", df.dtypes)#Analysing DataTypes
print("\nMissing Values:\n", df.isnull().sum())#Checking for Missing Values
print("\nSummary Statistics:\n", df.describe())#Printing Statics of the Data set.

```

```
NOX        float64
RM         float64
AGE        float64
DIS        float64
RAD          int64
TAX          int64
PTRATIO    float64
B          float64



CRIM       0
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

Summary Statistics:
             CRIM          ZN       INDUS        CHAS         NOX          RM  \
count  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   
std      8.601545   23.322453    6.860353    0.253994    0.115878    0.702617   
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000   
25%      0.082045    0.000000    5.190000    0.000000    0.449000    5.885500   
50%      0.256510    0.000000    9.690000    0.000000    0.538000    6.208500   
75%      3.677083   12.500000   18.100000    0.000000    0.624000    6.623500   
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780000   

              AGE         DIS         RAD         TAX     PTRATIO           B  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000   
mean    68.574901    3.795043    9.549407  408.237154   18.455534  356.674032   
std     28.148861    2.105710    8.707259  168.537116    2.164946   91.294864   
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000   
25%     45.025000    2.100175    4.000000  279.000000   17.400000  375.377500   
50%     77.500000    3.207450    5.000000  330.000000   19.050000  391.440000   
75%     94.075000    5.188425   24.000000  666.000000   20.200000  396.225000   
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000   

            LSTAT        MEDV  
count  506.000000  506.000000  
mean    12.653063   22.532806  
std      7.141062    9.197104  
min      1.730000    5.000000  
25%      6.950000   17.025000  
50%     11.360000   21.200000  
75%     16.955000   25.000000  
max     37.970000   50.000000  
```

```
# Step 3: Handle missing values
df = df.fillna(df.mean(numeric_only=True))   # numeric → mean
for col in df.select_dtypes(include='object').columns:  # categorical → mode
    df[col] = df[col].fillna(df[col].mode()[0])

```

```
# Step 4: Detect outliers before handling
for col in df.select_dtypes(include=np.number).columns:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    outliers = df[(df[col] < lower) | (df[col] > upper)]
    print(f"{col}: {len(outliers)} outliers detected")

# Step 4b: Handle outliers (capping)
for col in df.select_dtypes(include=np.number).columns:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    df[col] = np.where(df[col] < lower, lower,
                       np.where(df[col] > upper, upper, df[col]))

```

```
CRIM: 0 outliers detected
ZN: 0 outliers detected
INDUS: 0 outliers detected
CHAS: 0 outliers detected
NOX: 0 outliers detected
RM: 0 outliers detected
AGE: 0 outliers detected
DIS: 0 outliers detected
```

```
RAD: 0 outliers detected
TAX: 0 outliers detected
PTRATIO: 0 outliers detected
B: 0 outliers detected
LSTAT: 0 outliers detected
MEDV: 0 outliers detected
```

```
# Step 5: Save cleaned dataset
df.to_csv("boston_cleaned.csv", index=False)

files.download("boston_cleaned.csv")

```