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

df = pd.read_csv('boston.csv')
df.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7

Next steps: View recommended plots

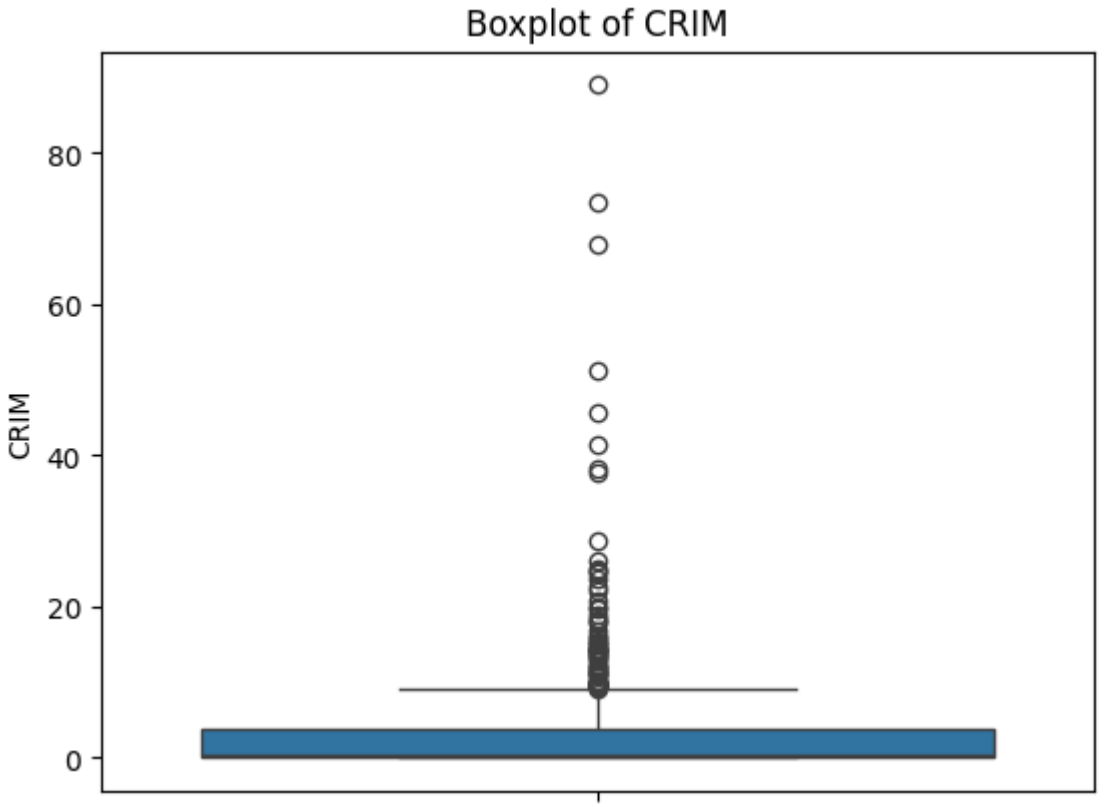
```
df.isnull().sum()
```

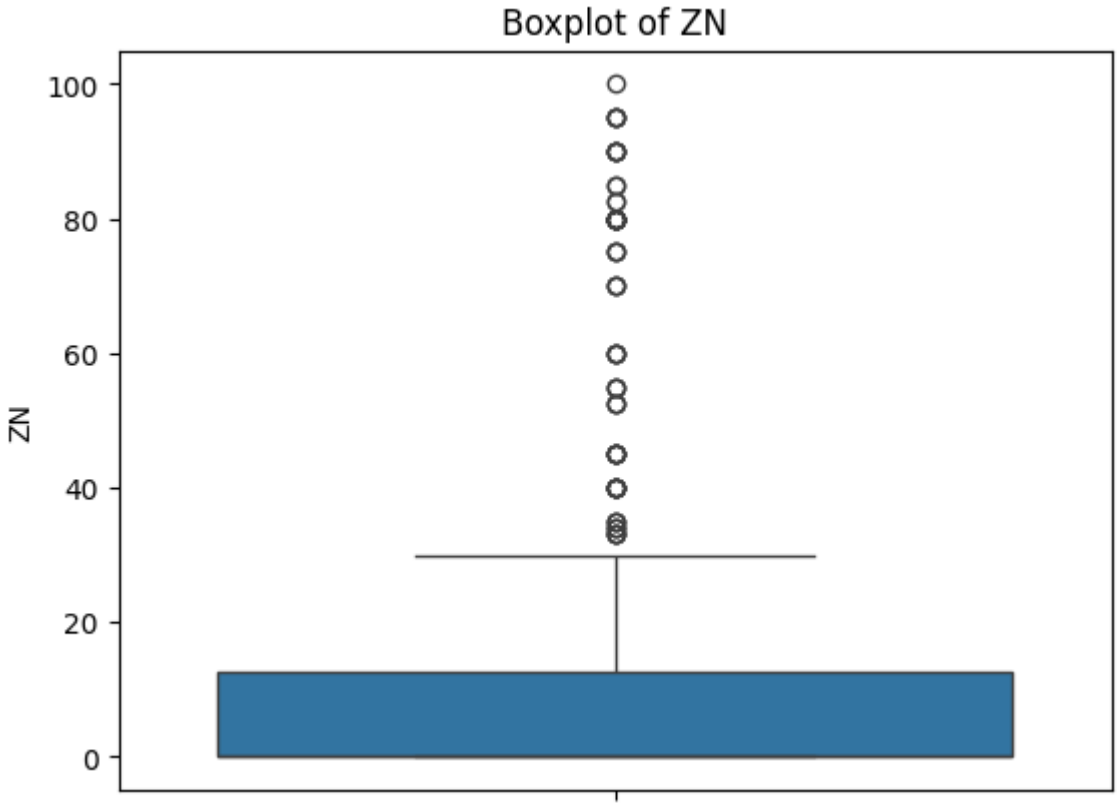
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

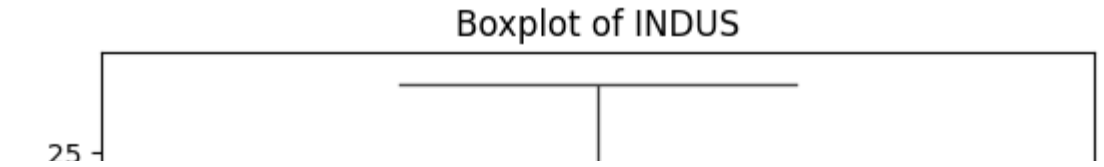
```
numerical_features = []
categorical_features = []

for f in df.columns:
  if df[f].dtype != 'O':
    numerical_features.append(f)
  else:
    categorical_features.append(f)

print(numerical_features)
print(categorical_features)
```

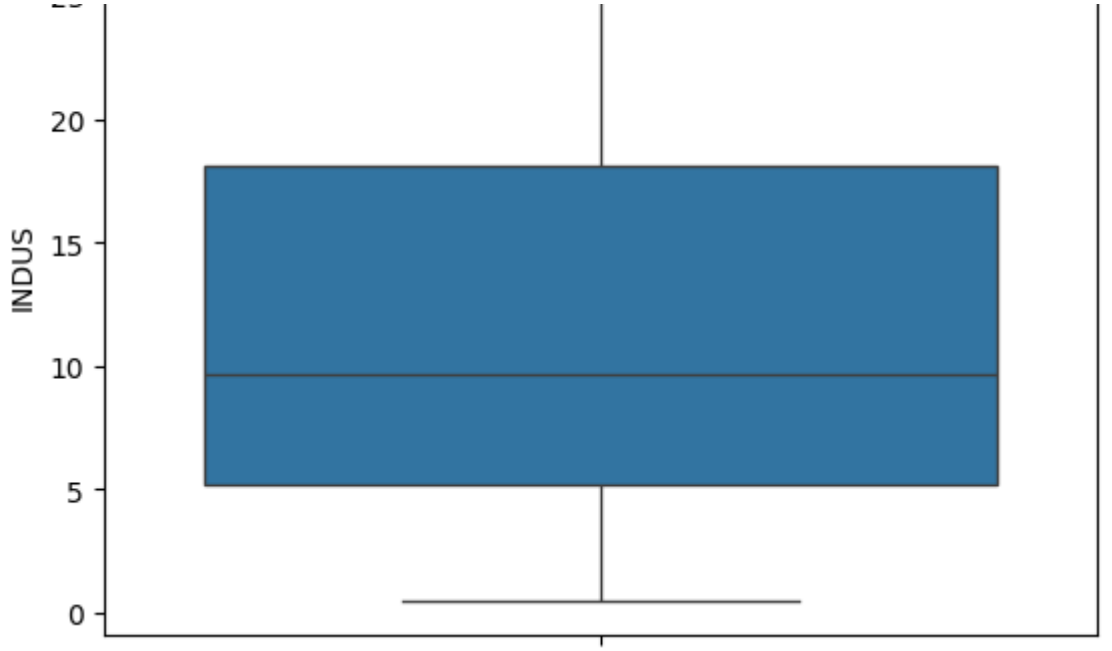
```
['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'I
[]
```

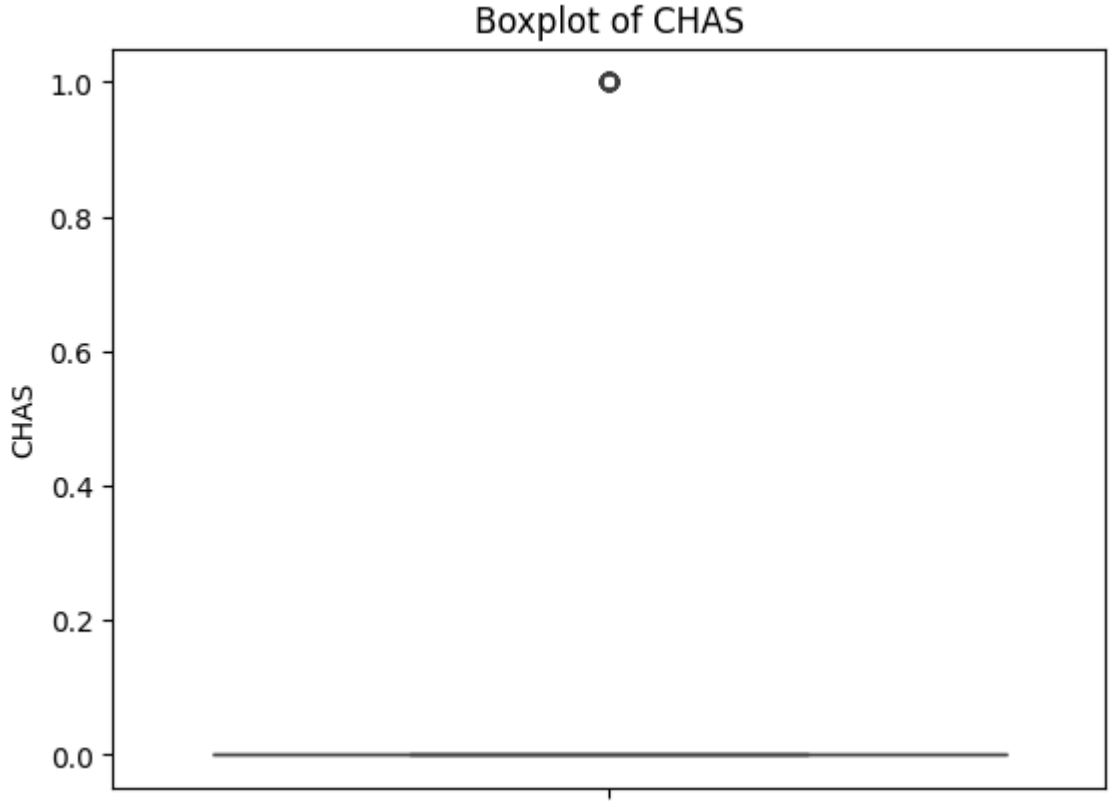
```
for f in df.columns:
  sns.boxplot(df[f])
  plt.title(f'Boxplot of {f}')
  plt.show()
```

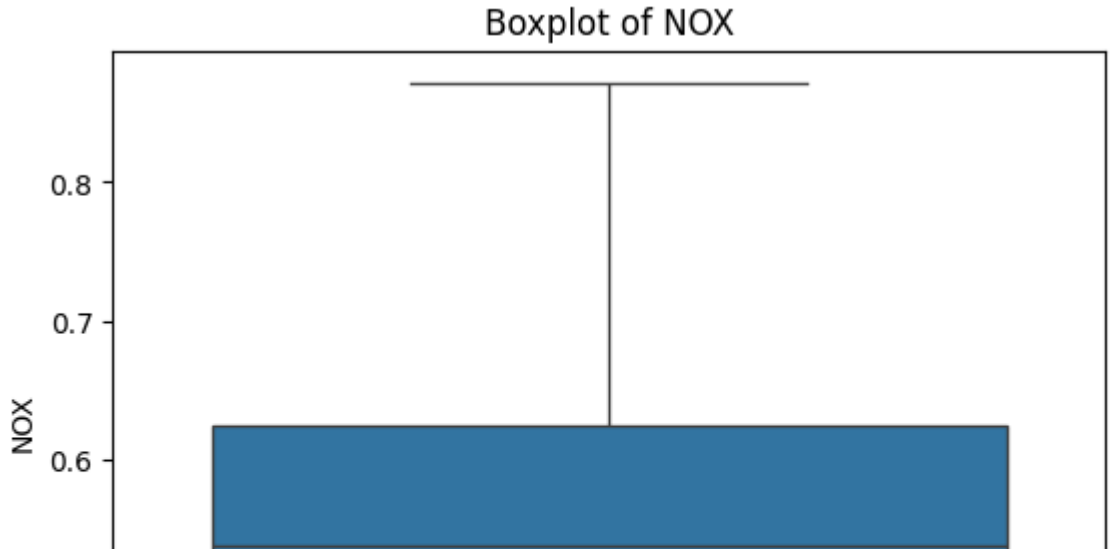


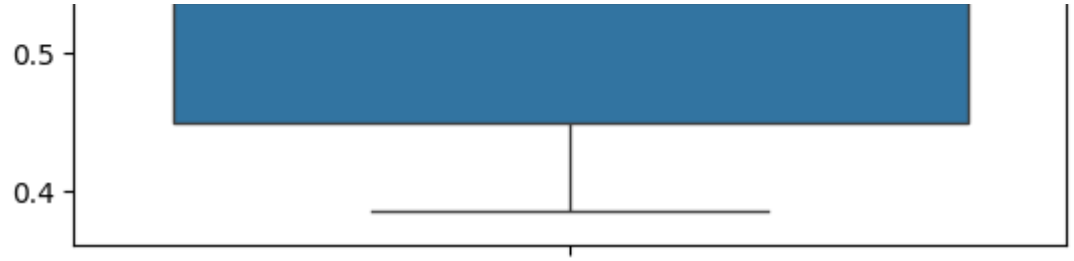


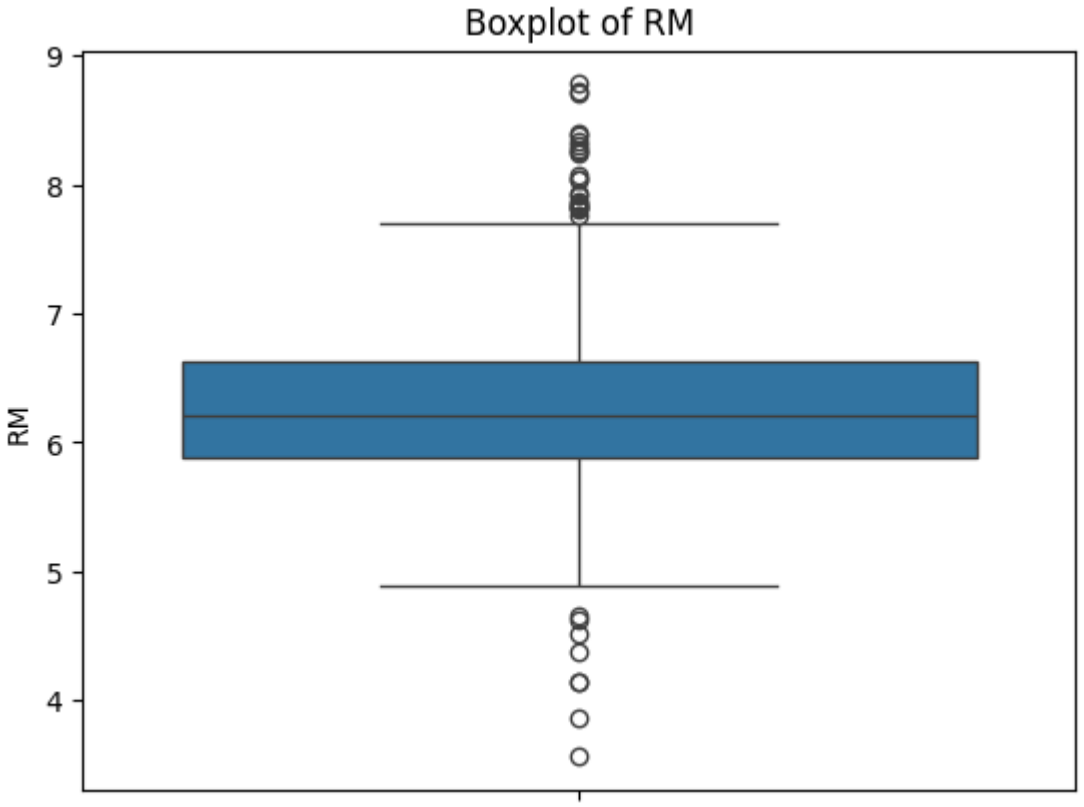


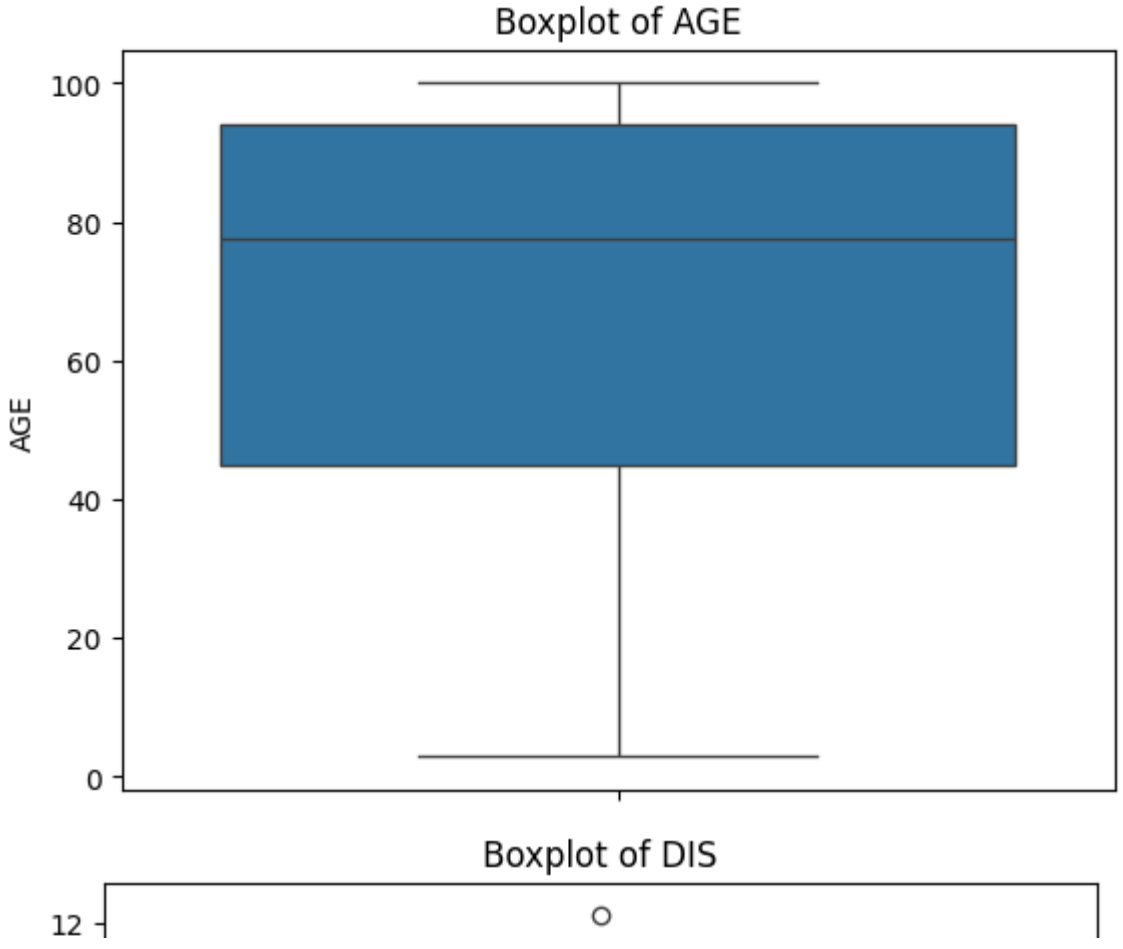

INDUS
20
15
10
5
0


Boxplot of CHAS
CHAS
1.0
0.8
0.6
0.4
0.2
0.0

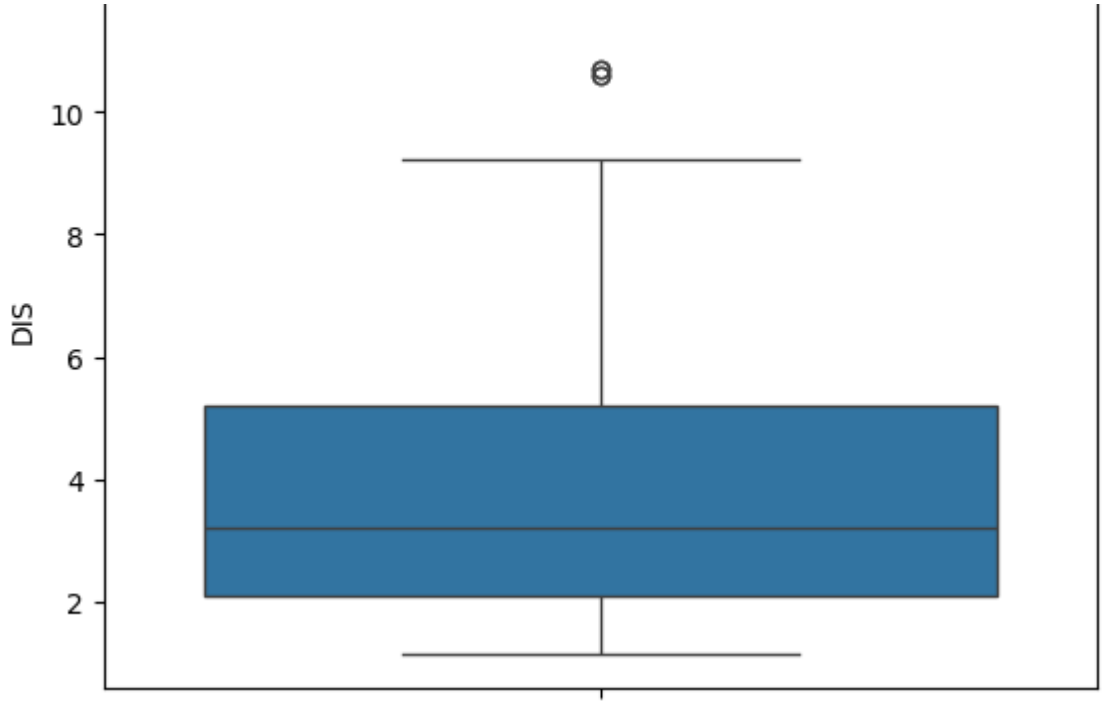

Boxplot of NOX
NOX
0.8
0.7
0.6

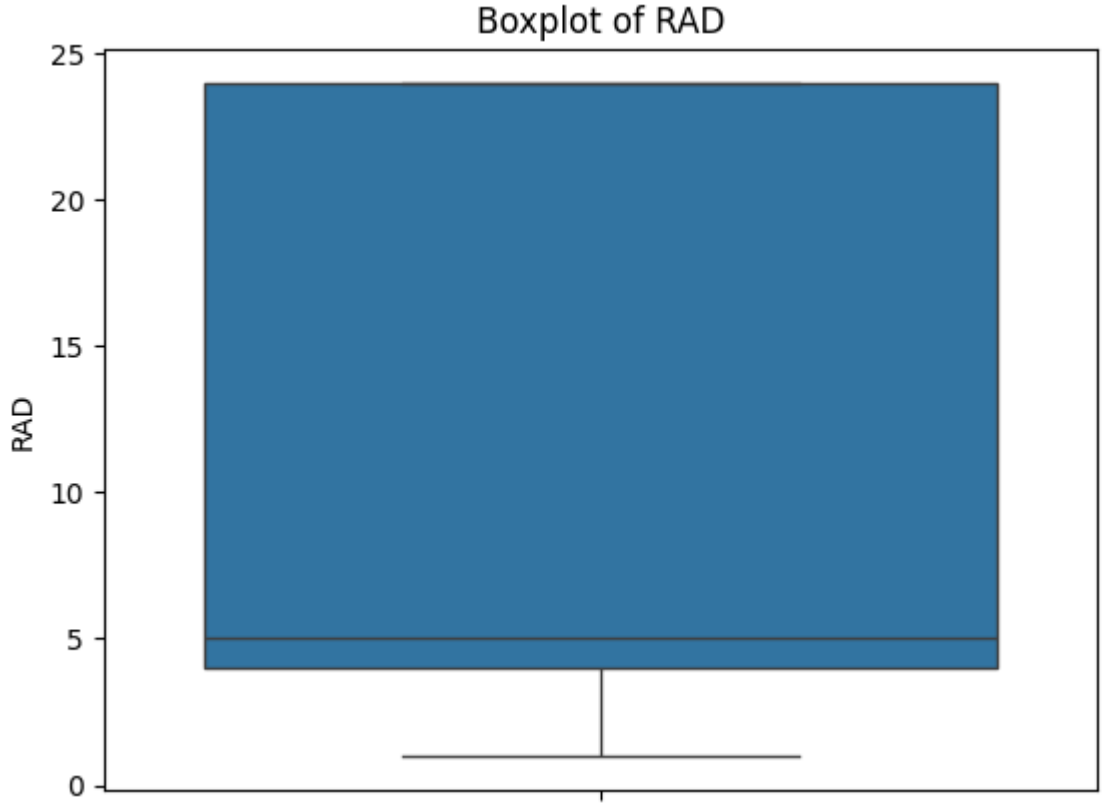

0.5
0.4

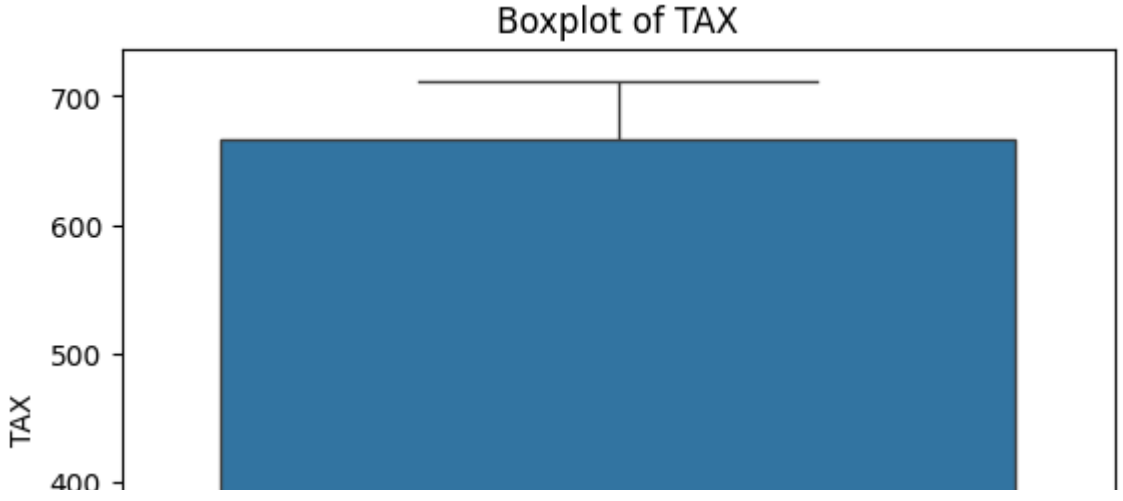

Boxplot of RM
RM
9
8
7
6
5
4

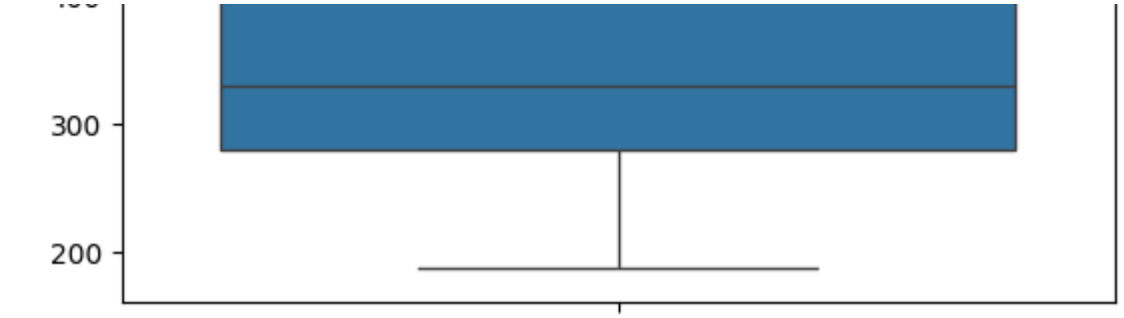

Boxplot of AGE
AGE
100
80
60
40
20
0

Boxplot of DIS
12

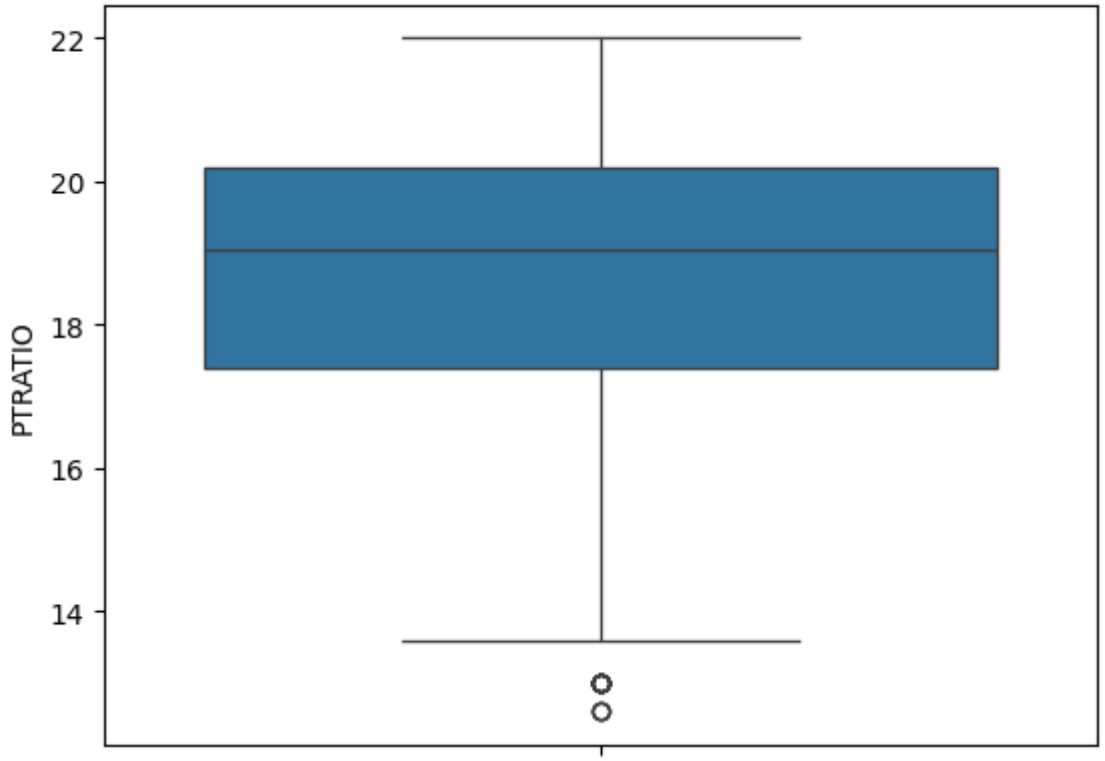

DIS
10
8
6
4
2


Boxplot of RAD
RAD
25
20
15
10
5
0

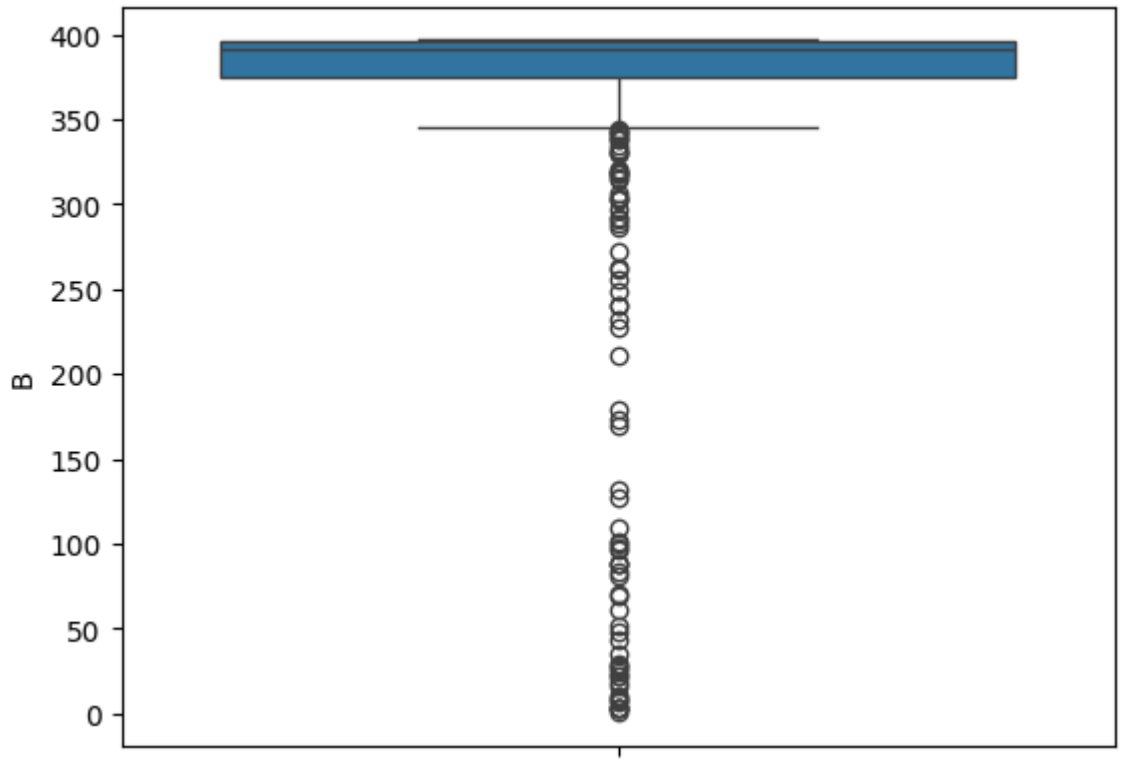

Boxplot of TAX
TAX
700
600
500
400



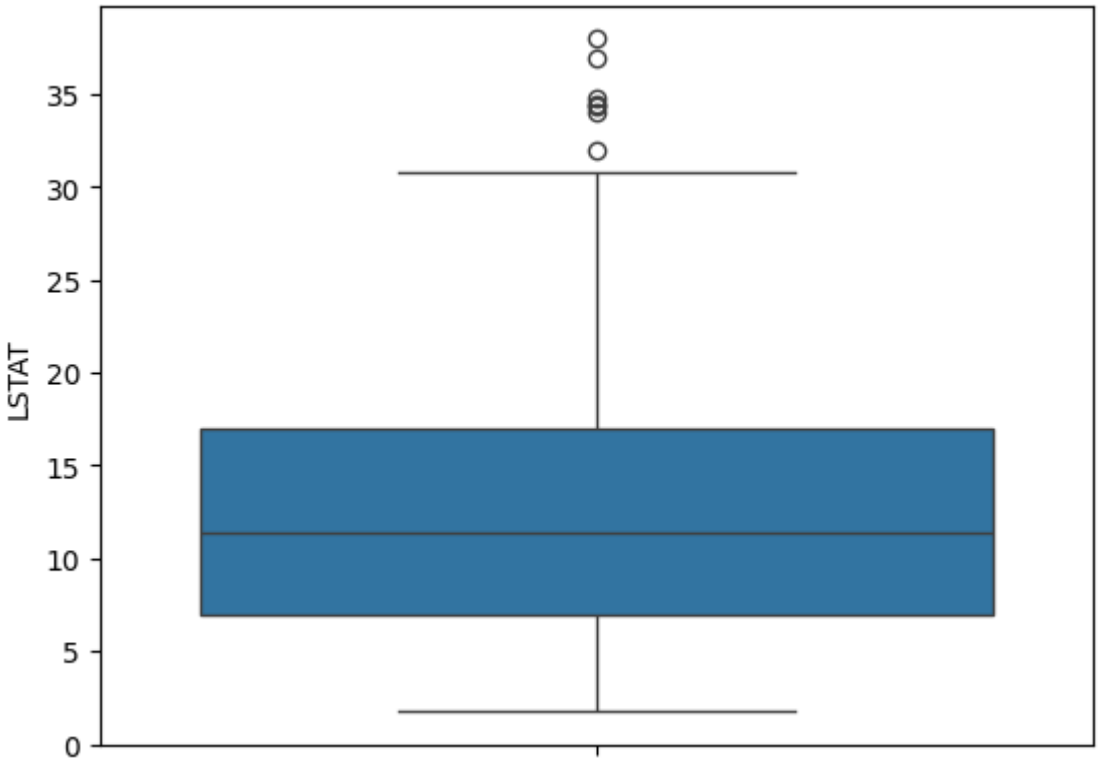
Boxplot of PTRATIO

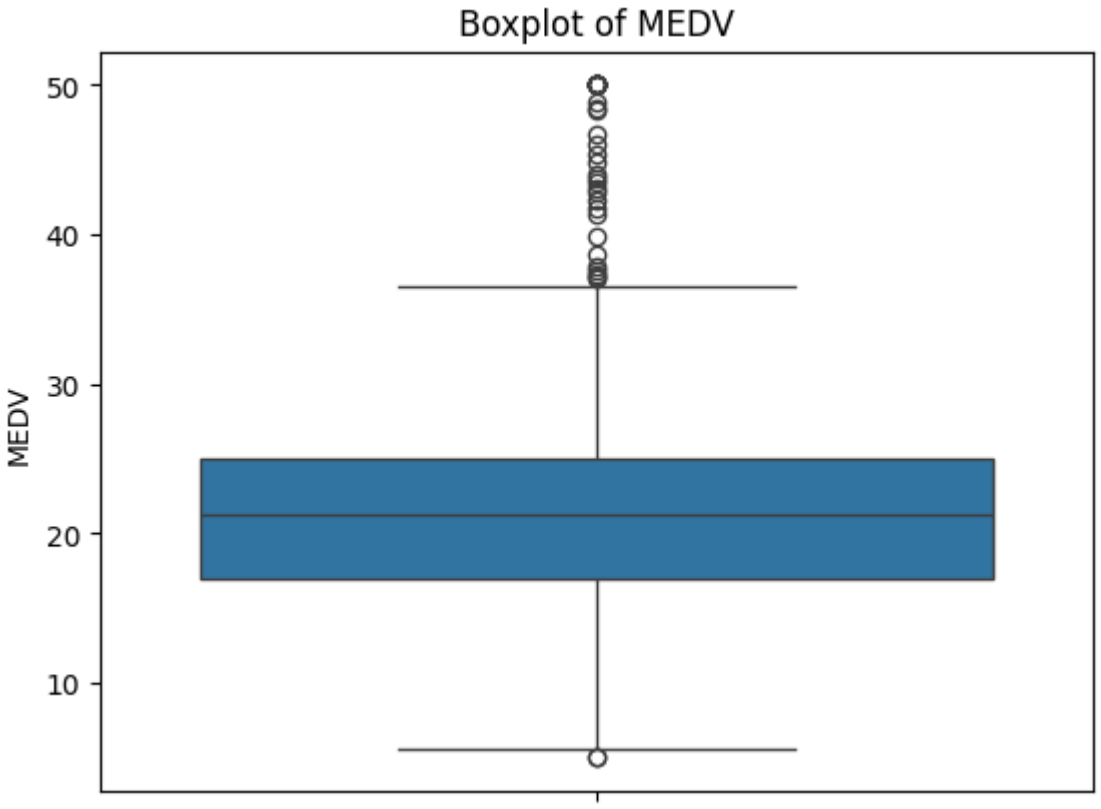


Boxplot of B



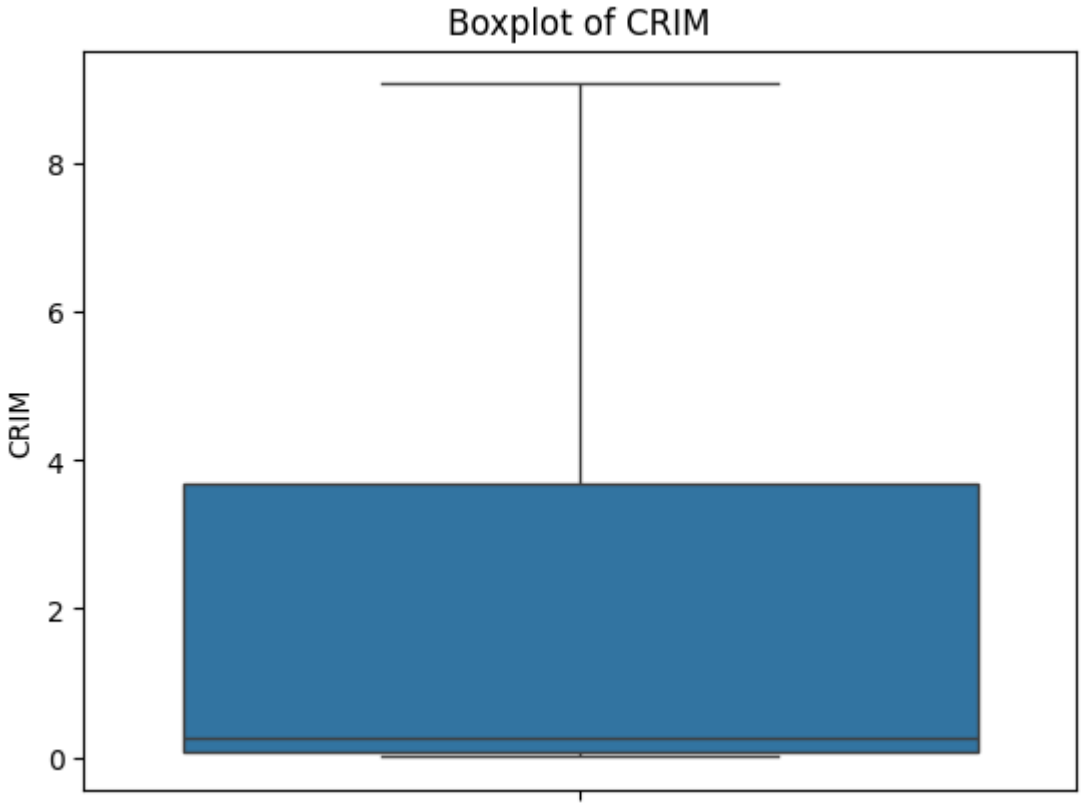
Boxplot of LSTAT

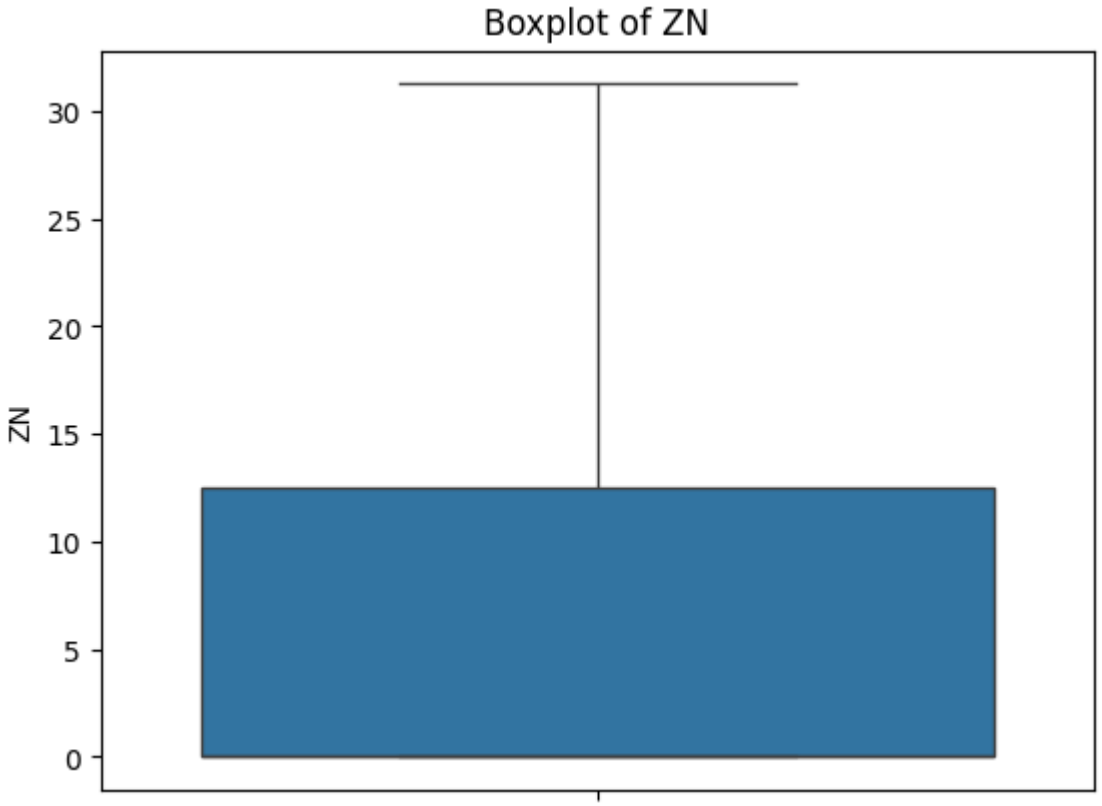


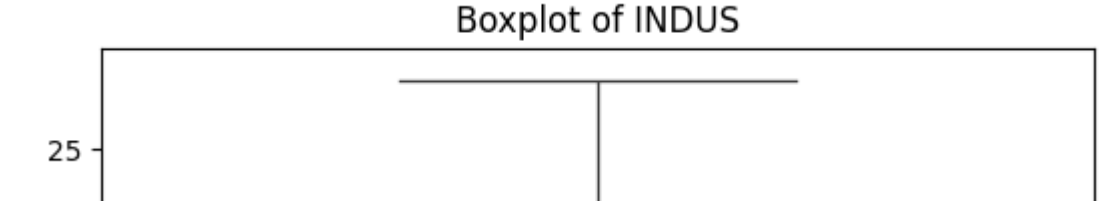


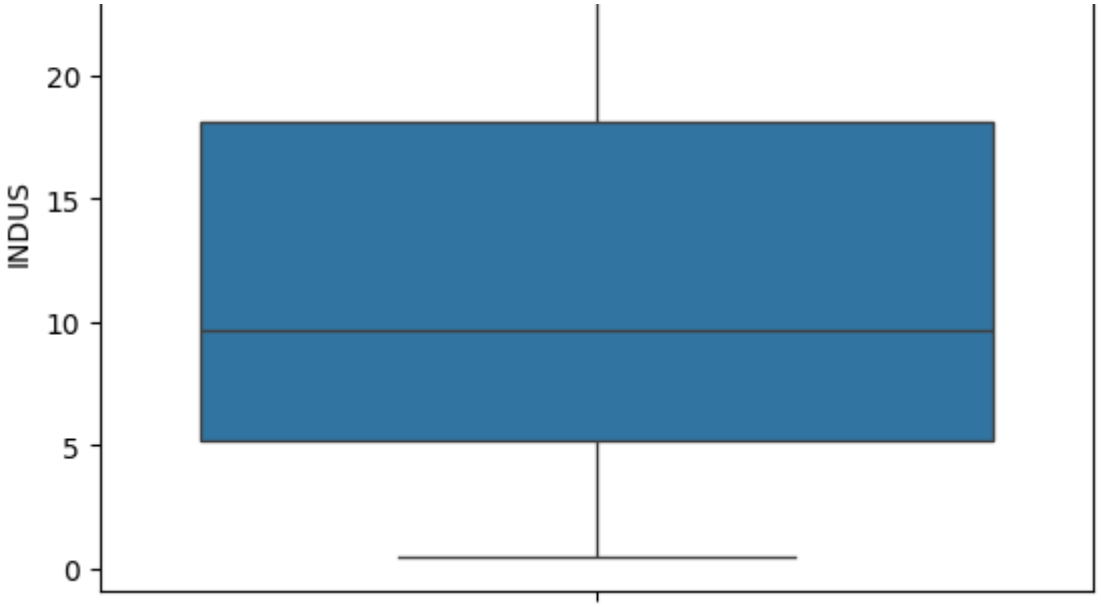

```
numericalCols = df.select_dtypes(include='number').columns
for f in numericalCols:
  q3 = df[f].quantile(0.75)
  q1 = df[f].quantile(0.25)
  iqr = q3 - q1
  upper_bound = q3 + (1.5 * iqr)
  lower_bound = q1 - (1.5 * iqr)
  df.loc[df[f] >= upper_bound, f] = upper_bound
  df.loc[df[f] <= lower_bound, f] = lower_bound
```

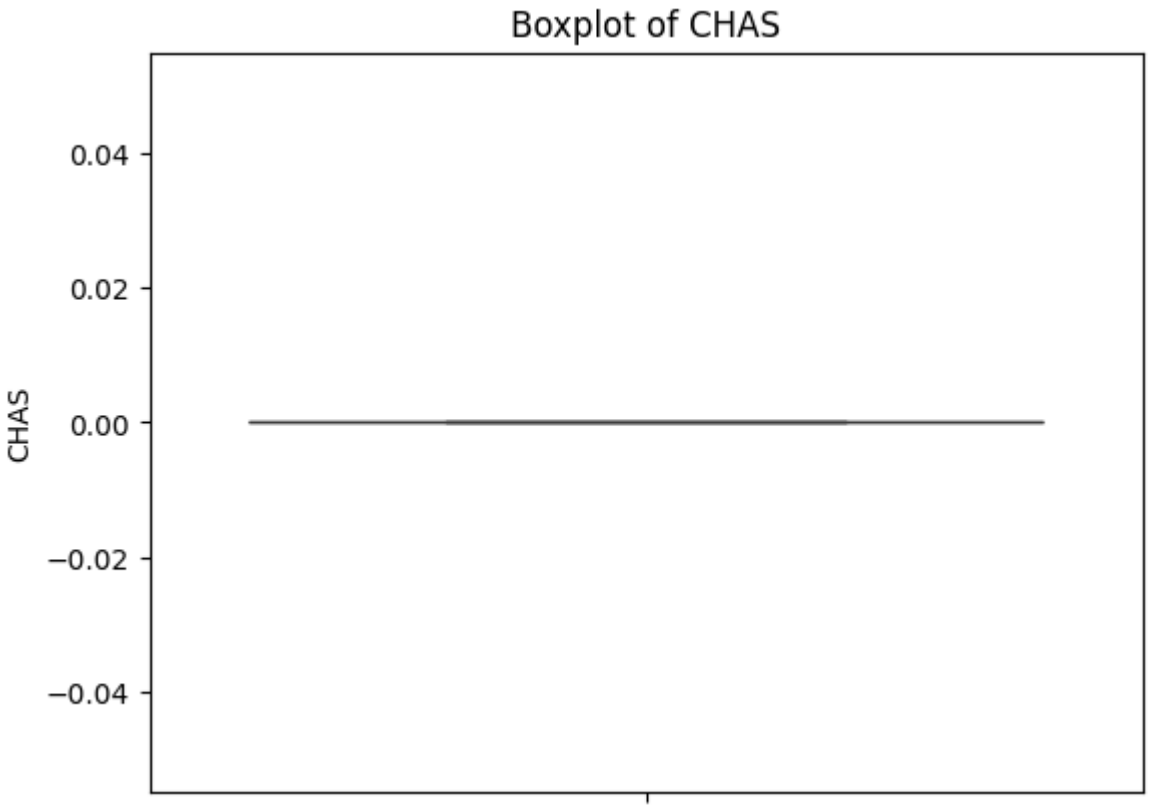
```
for f in df.columns:
  sns.boxplot(df[f])
  plt.title(f'Boxplot of {f}')
  plt.show()
```

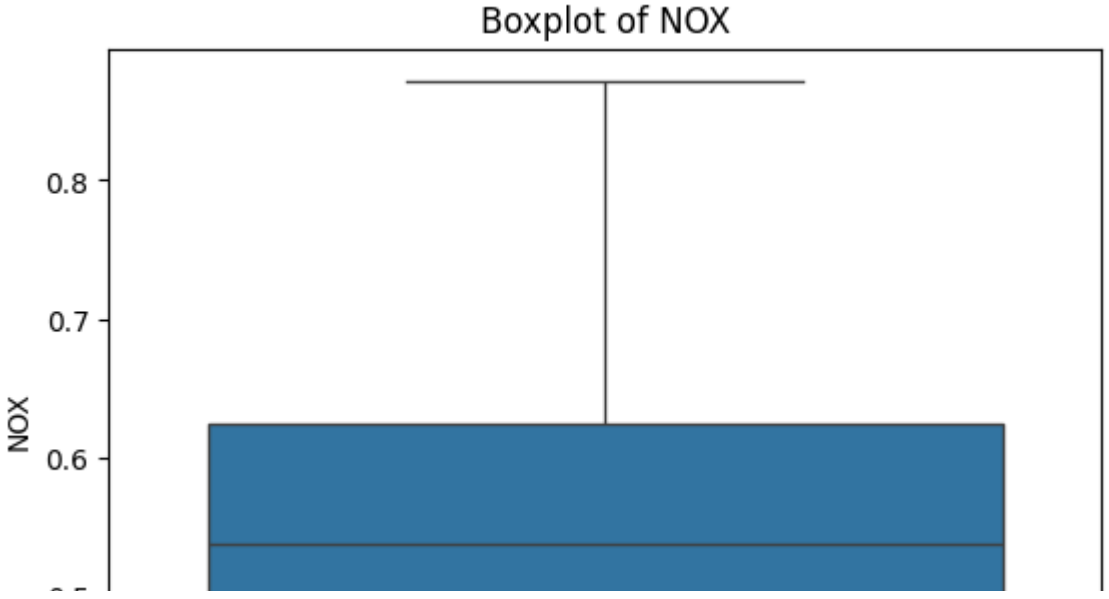


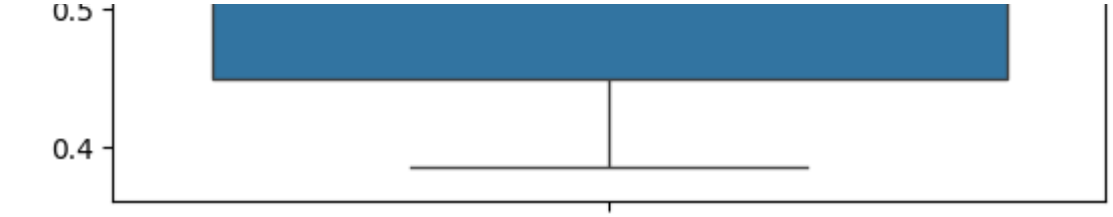


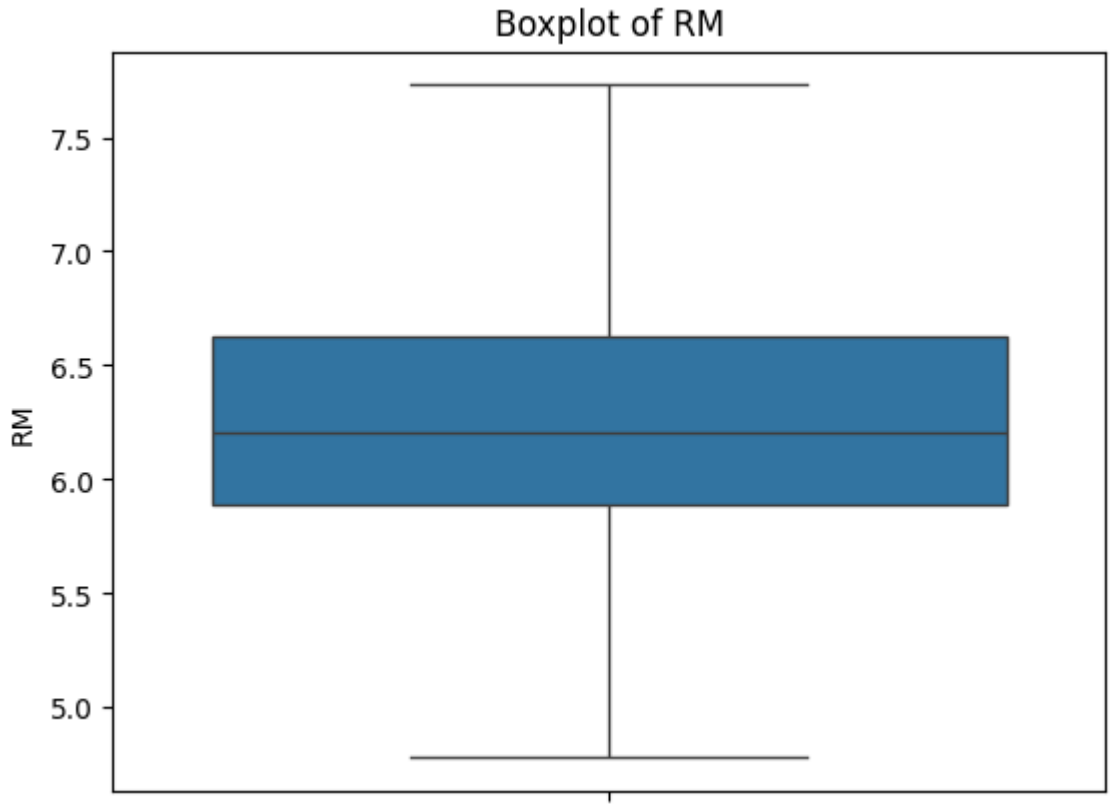


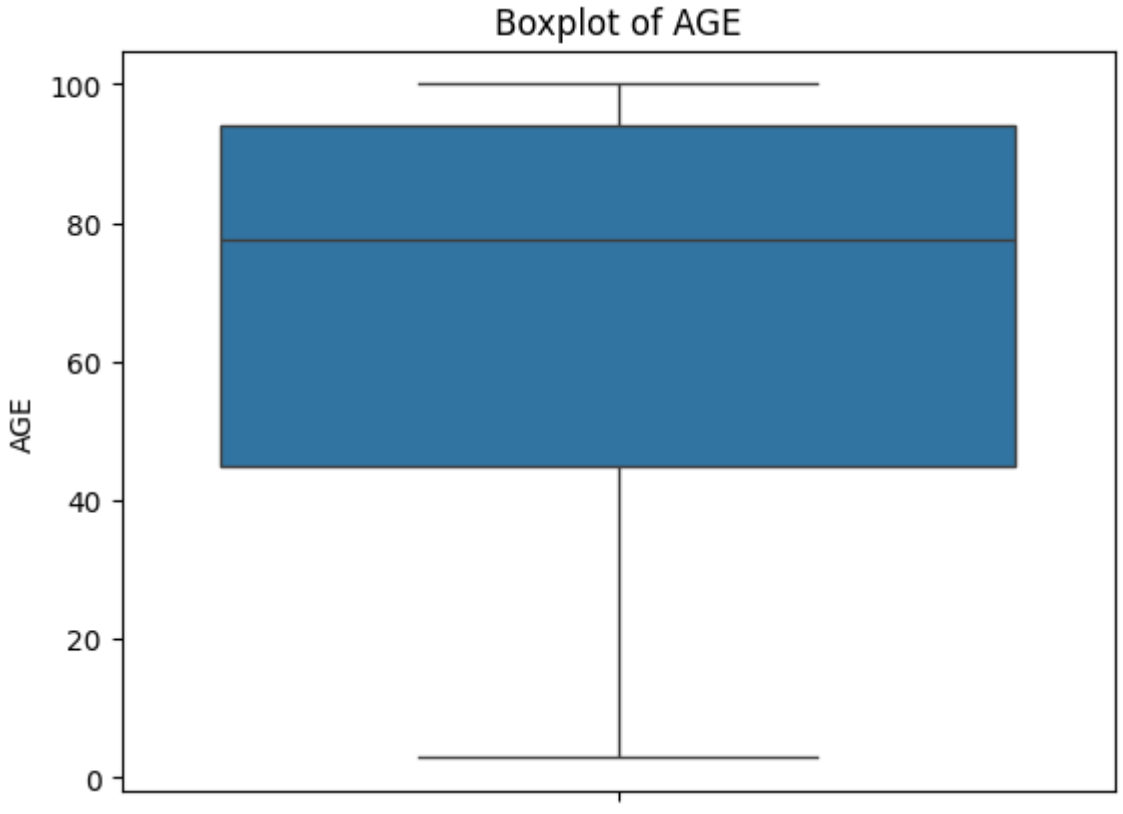

INDUS
20
15
10
5
0

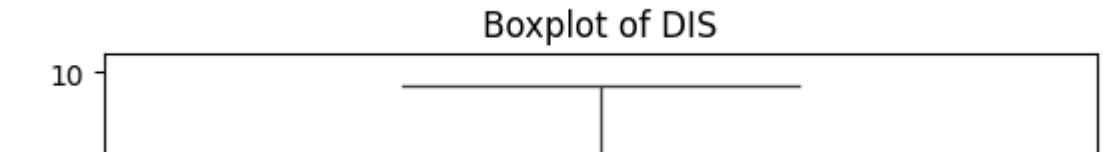

Boxplot of CHAS
CHAS
0.04
0.02
0.00
−0.02
−0.04

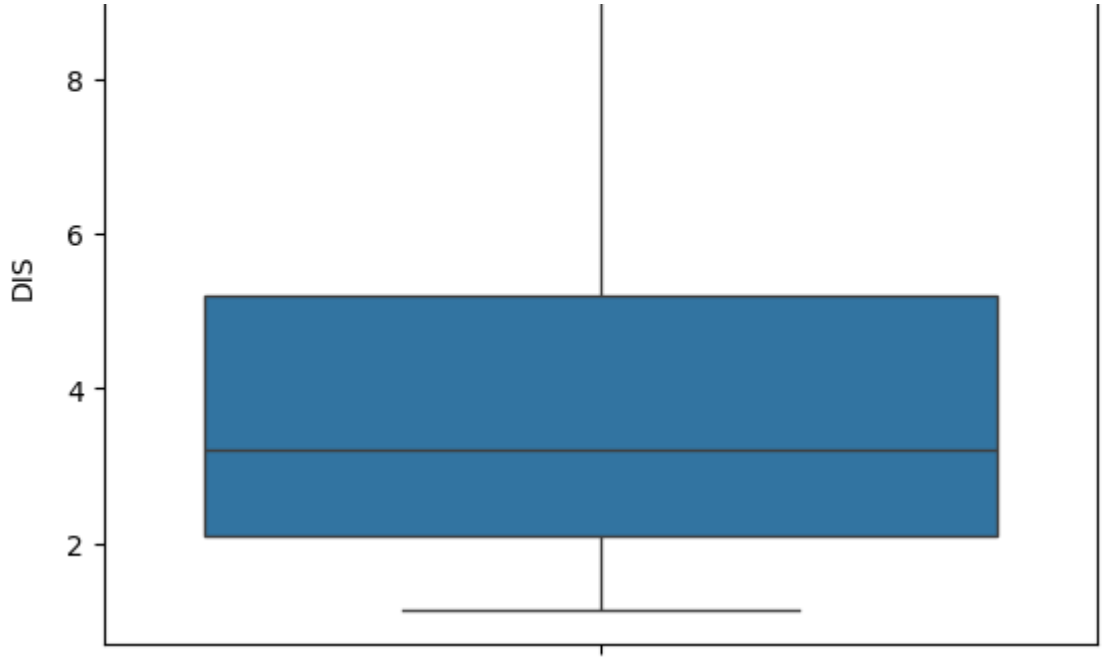

Boxplot of NOX
NOX
0.8
0.7
0.6

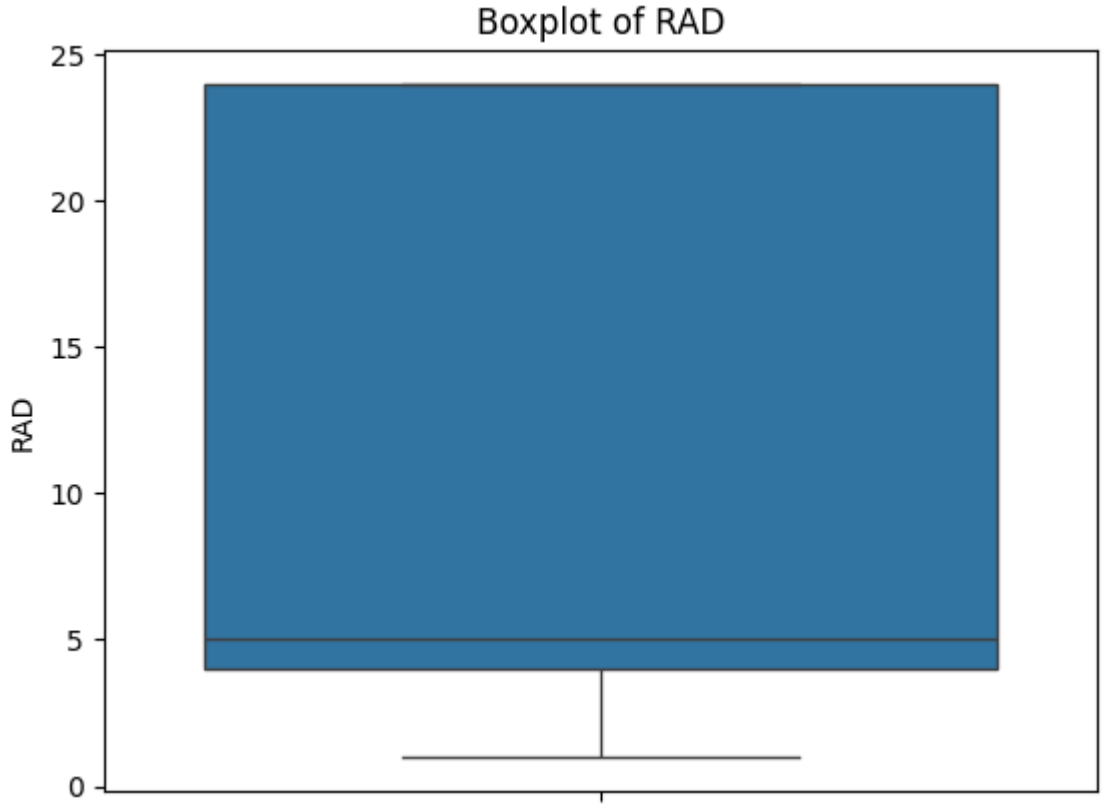

0.5
0.4

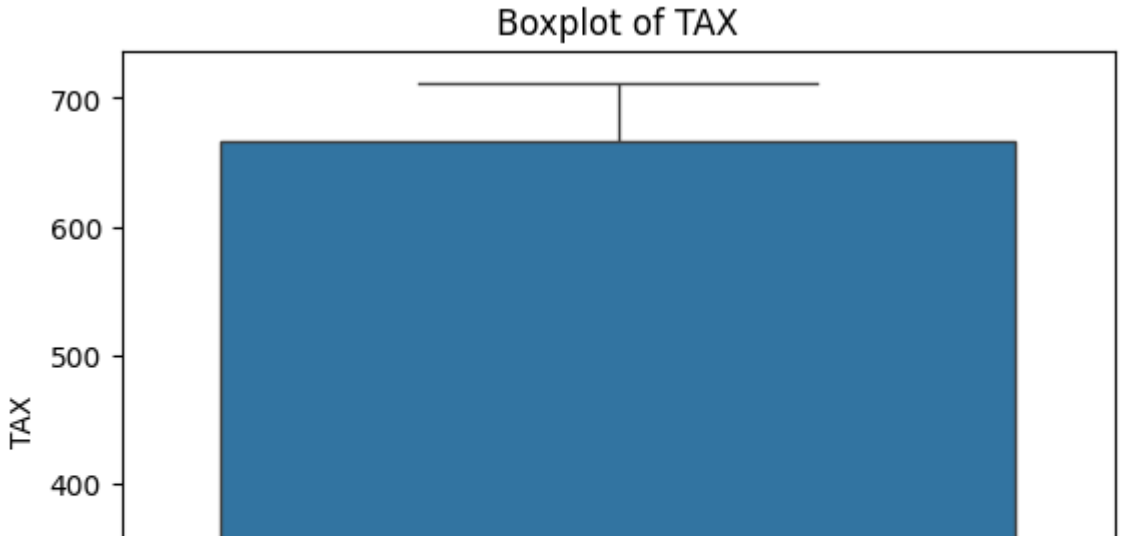

Boxplot of RM
RM
7.5
7.0
6.5
6.0
5.5
5.0

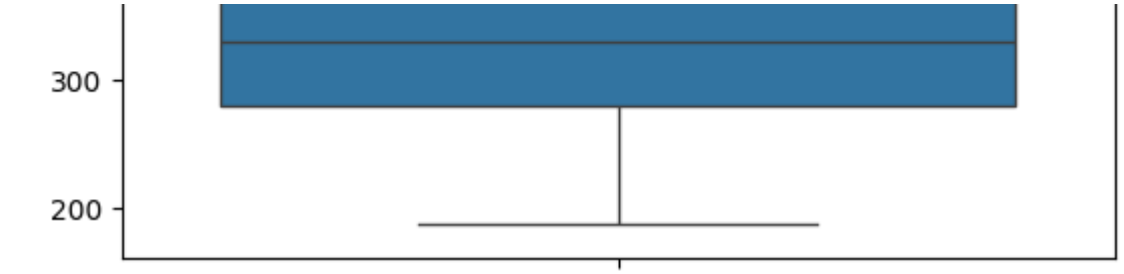

Boxplot of AGE
AGE
100
80
60
40
20
0


Boxplot of DIS
10

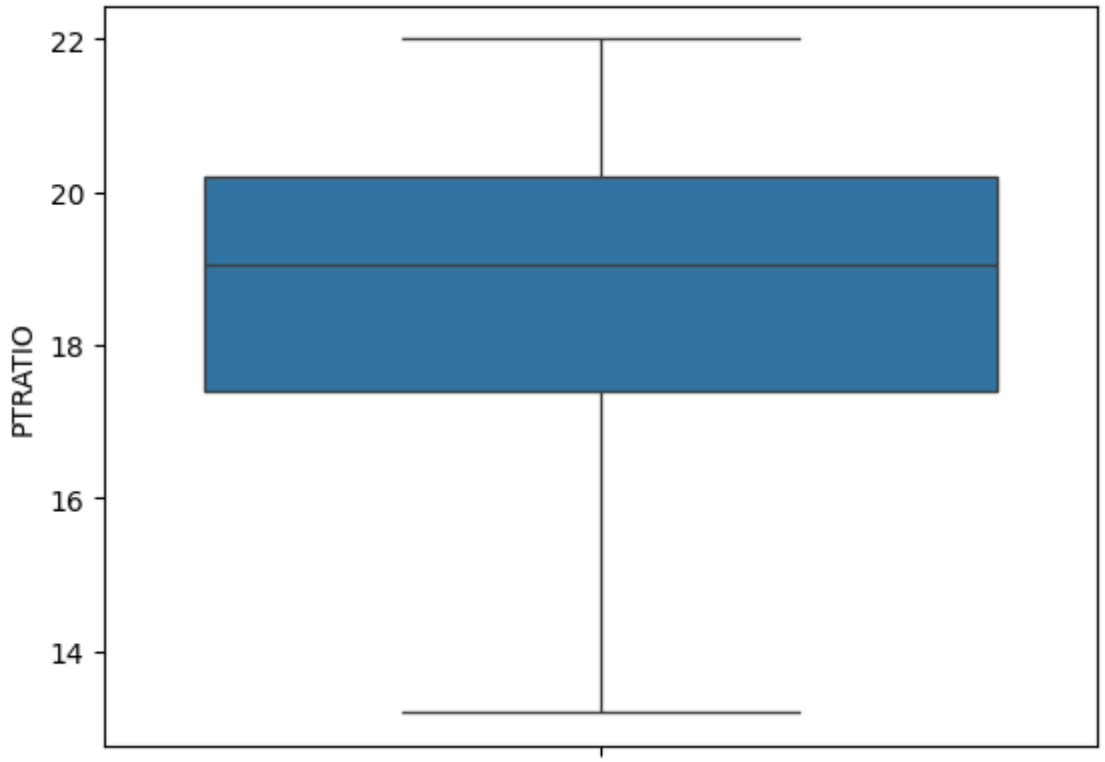

8
6
4
2
DIS


Boxplot of RAD
25
20
15
10
5
0
RAD

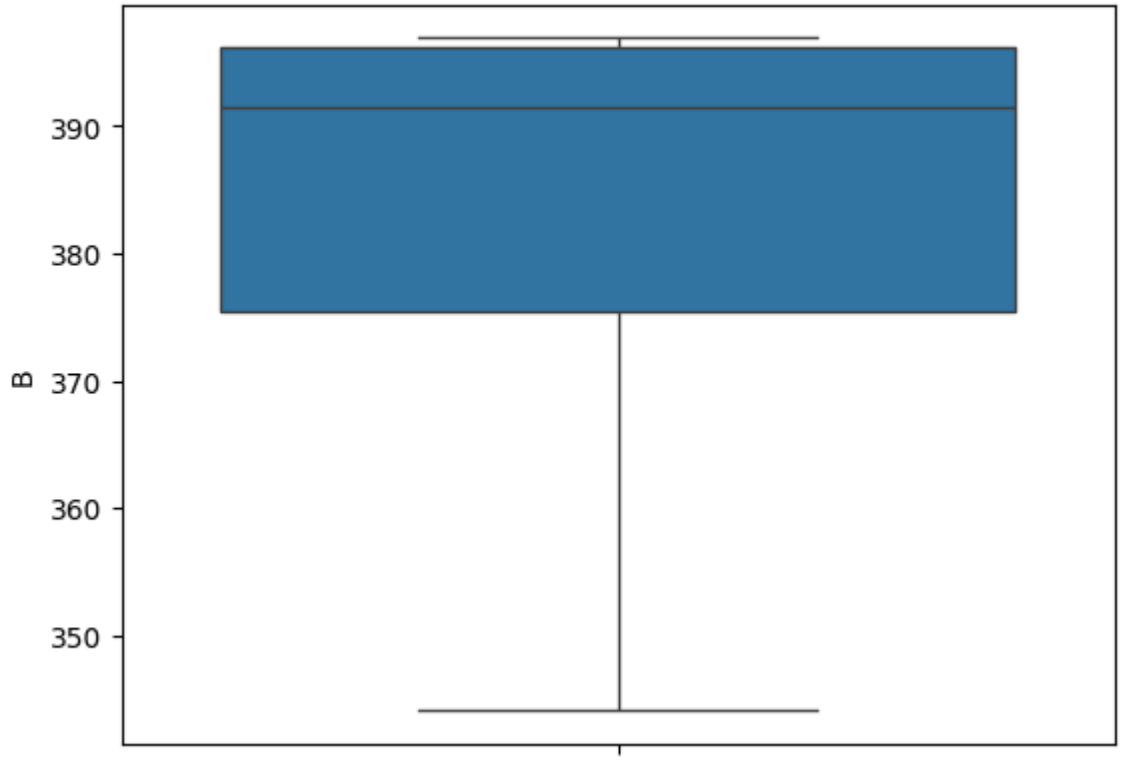

Boxplot of TAX
700
600
500
400
TAX



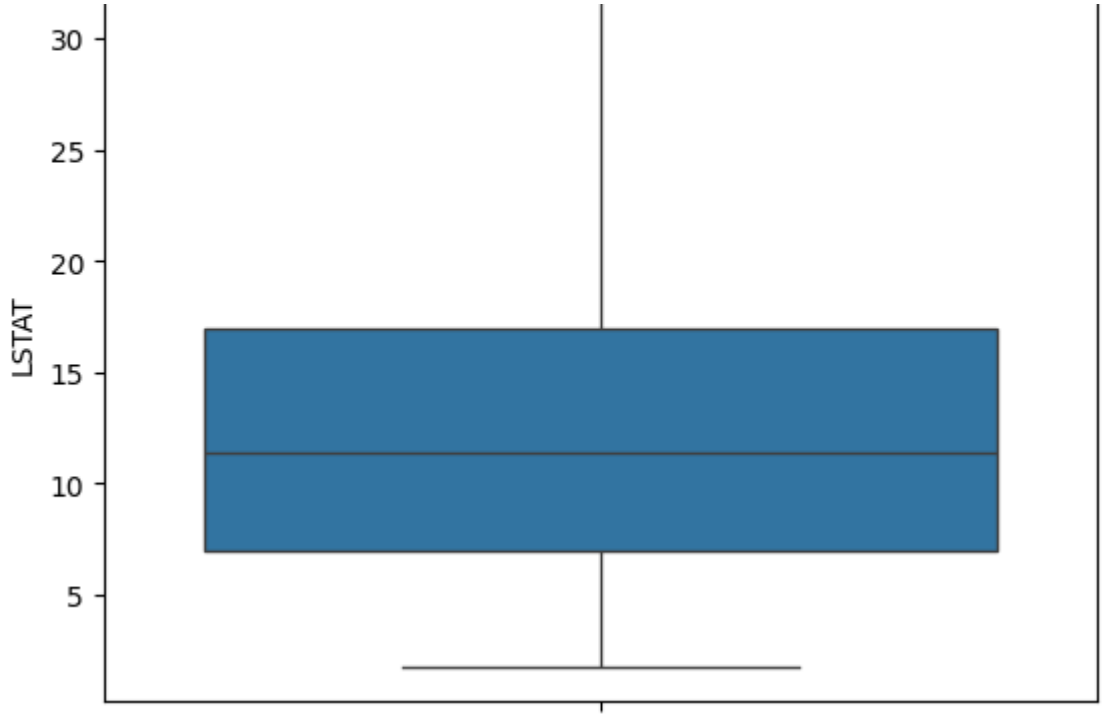
Boxplot of PTRATIO

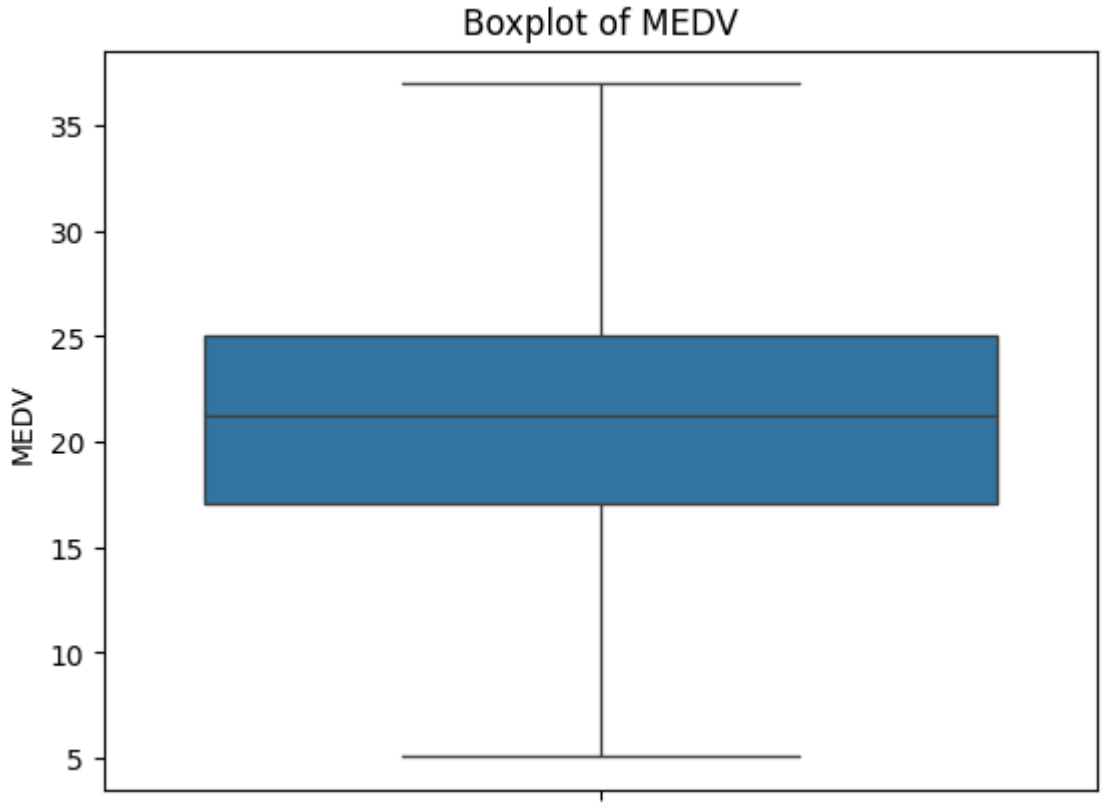


Boxplot of B



Boxplot of LSTAT

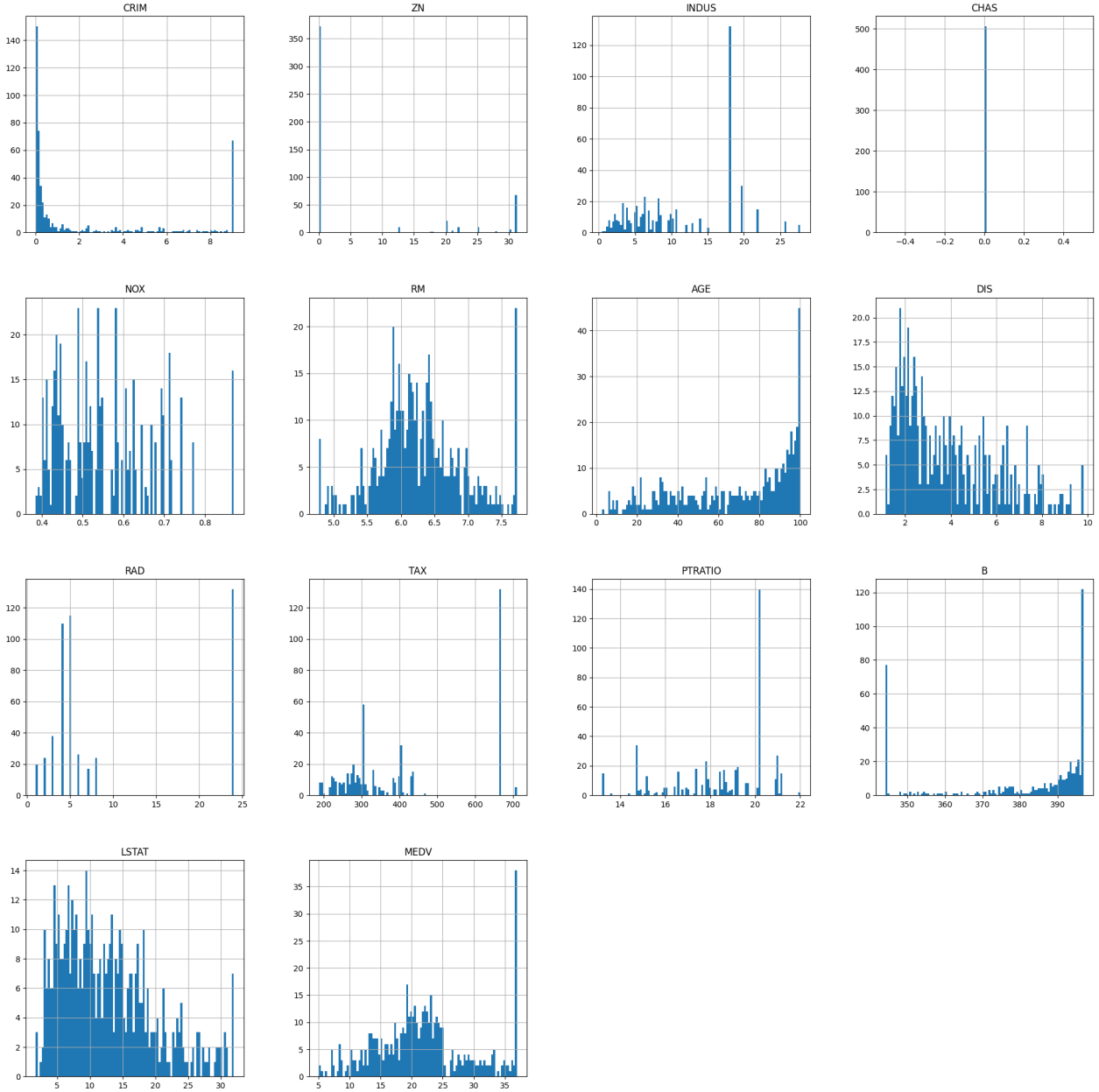





```
df.hist(figsize=(24,24), bins=100)
```

```
array([[<Axes: title={'center': 'CRIM'}>, <Axes: title={'center': 'ZN'}>,
        <Axes: title={'center': 'INDUS'}>,
        <Axes: title={'center': 'CHAS'}>],
       [<Axes: title={'center': 'NOX'}>, <Axes: title={'center': 'RM'}>,
        <Axes: title={'center': 'AGE'}>, <Axes: title={'center': 'DIS'}>],
       [<Axes: title={'center': 'RAD'}>, <Axes: title={'center': 'TAX'}>,
        <Axes: title={'center': 'PTRATIO'}>,
        <Axes: title={'center': 'B'}>],
       [<Axes: title={'center': 'LSTAT'}>,
        <Axes: title={'center': 'MEDV'}>, <Axes: >, <Axes: >]],
```

```
dtype=object)
```



```
df.skew()
```

```
CRIM       1.282313
ZN         1.261340
INDUS      0.295022
CHAS       0.000000
NOX        0.729308
RM         0.296640
AGE       -0.598963
DIS        0.908467
RAD        1.004815
TAX        0.669956
PTRATIO   -0.762495
B         -1.164208
LSTAT      0.808671
MEDV       0.353614
dtype: float64
```

```
# Model building
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error

X = df.drop('MEDV', axis=1)
y = df['MEDV']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
regr = LinearRegression()
regr.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
# predicting the model output
y_pred = regr.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 13.264821903384027
```

```
print("Tuned Weights:")
for i in range(len(X.columns)):
    print(f"{X.columns[i]}: {regr.coef_[i]}")
print(f"Intercept: {regr.intercept_}")
```

```
Tuned Weights:
CRIM: -0.47702088562571165
ZN: 0.04146279314123744
INDUS: -0.021440789997329568
CHAS: 8.340550472496489e-14
NOX: -14.041056617657127
RM: 2.7458448668745974
AGE: -0.002722590804800087
```

```
AGE: 0.0027229568048000372
DIS: -1.0342233144791397
RAD: 0.28211799925832115
TAX: -0.008092199582586938
PTRATIO: -0.8728299586706918
B: 0.006769390573664431
LSTAT: -0.4916902018584183
Intercept: 37.97381451281785
```