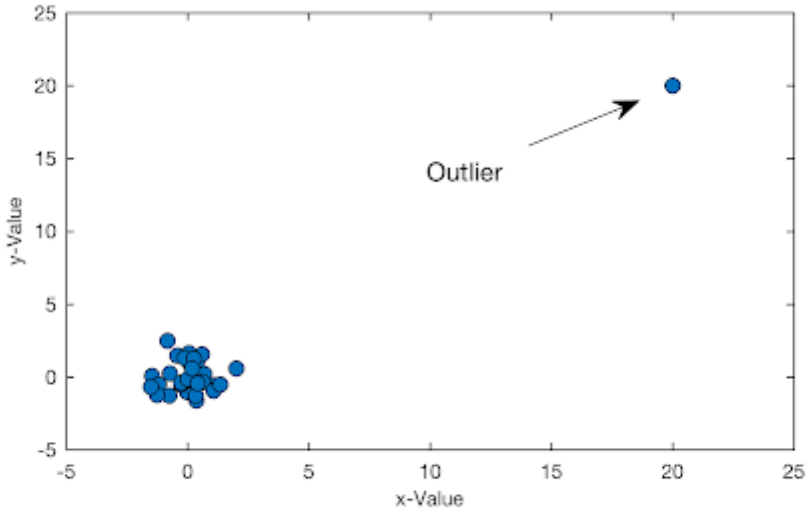
Outlier

- Outliers are extreme values that deviate from other observations on data , they may indicate a variability in a measurement, experimental errors or a novelty. In other words, an outlier is an observation that diverges from an overall pattern on a sample.



Nature of Outliers :

1. Genuine extreme high and low values in the dataset
2. Introduced due to human or mechanical error
3. Introduced by replacing missing values

Types of outliers :

Outliers can be of two kinds:

1. Univariate - can be found when looking at a distribution of values in a single feature space.
2. Multivariate - can be found in a n-dimensional space (of n-features).

Most common causes of outliers on a data set :

- Data entry errors (human errors)
- Measurement errors (instrument errors)
- Experimental errors (data extraction or experiment planning/executing errors)
- Intentional (dummy outliers made to test detection methods)
- Data processing errors (data manipulation or data set unintended mutations)
- Sampling errors (extracting or mixing data from wrong or various sources)
- Natural (not an error, novelties in data)

Outliers can also come in different variate :

```
- Point Outlier -  single data points that lay far from the rest of the distributi
on
- collective outliers - can be noise in data, such as punctuation symbols when rea
lizing text analysis or background noise signal when doing speech recognition.
- contextual outliers -  subsets of novelties in data such as a signal that may in
dicate the discovery of new phenomena
```

Detection And Prevention

- There are many method for do this function but here i show widly use method.

Outlier Detection

- Boxplot
- Scatterplot
- Z_score

Outlier Prevention

- Z-score method
- IQR method
- Log transform

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.simplefilter('ignore')
```

In [2]:

```
from sklearn.datasets import load_boston
boston = load_boston()
df = pd.DataFrame(boston.data,columns=boston.feature_names)
df
```

Out[2]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90
...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90

506 rows × 13 columns

Outlier Detection

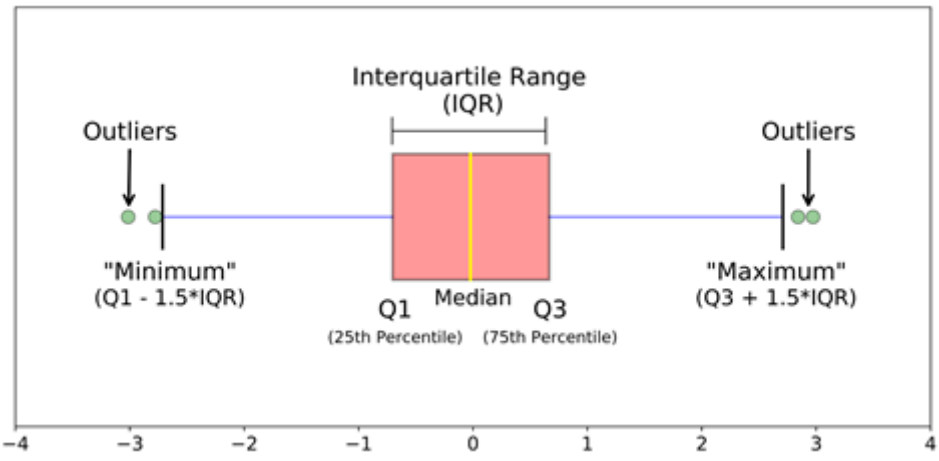
1. Boxplot

2. Scatter Plot

3. Z_score

1. BoxPlot

- The very purpose of box plots is to identify outliers in the data series before making any further analysis so that the conclusion made from the study gives more accurate results not influenced by any extremes or abnormal values.



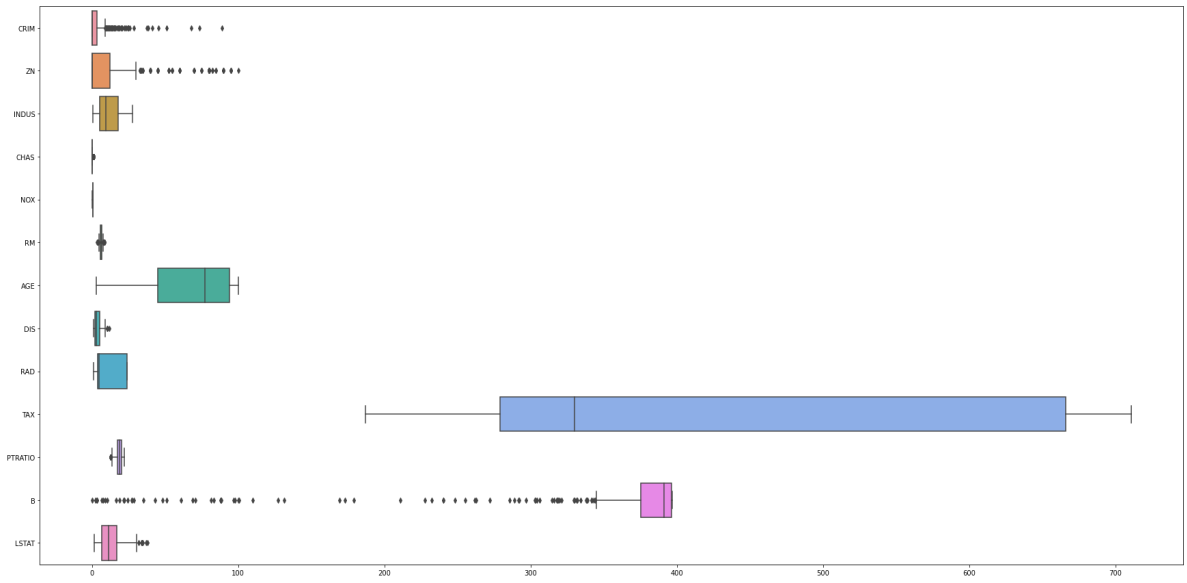
Implementation Boxplot

In [3]:

```
plt.figure(figsize=(30,15))
sns.boxplot(data=df, orient="h")
```

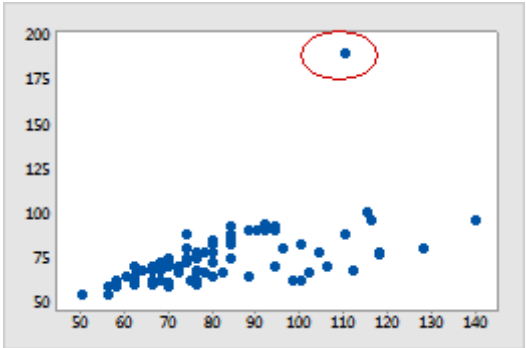
Out[3]:

<AxesSubplot:>



2. Scatter Plot

- Here we must put 2 variable as it is bivarient plot

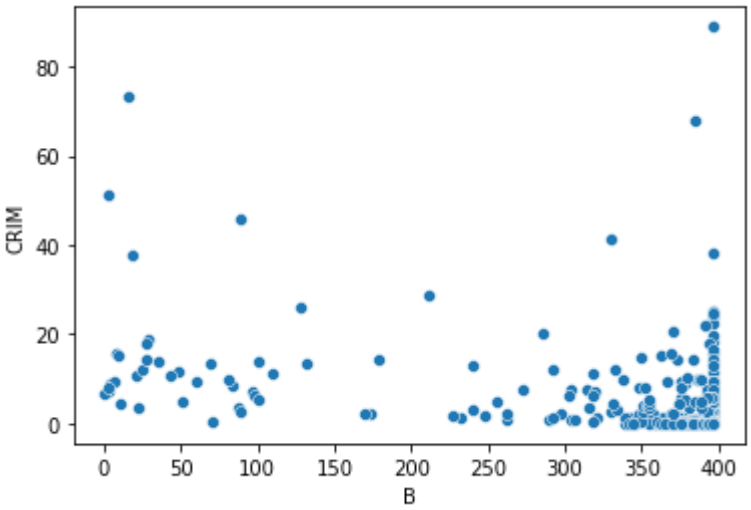


In [4]:

```
sns.scatterplot(df['B'],df['CRIM'])
```

Out[4]:

```
<AxesSubplot:xlabel='B', ylabel='CRIM'>
```



3. Z_Score

- Z score is an important concept in statistics. Z score is also called standard score. This score helps to understand if a data value is greater or smaller than mean and how far away it is from the mean. More specifically, Z score tells how many standard deviations away a data point is from the mean.

$$Z = \frac{X - \mu}{\sigma}$$

Z → Standard (Normal) or Z score

X → member element of group

μ → mean of expectation

σ → standard deviation

Implement

In [5]:

```
from scipy import stats
zscore = np.abs(stats.zscore(df))
print(zscore)
```

```
[[0.41978194 0.28482986 1.2879095  ... 1.45900038 0.44105193 1.0755623 ]
 [0.41733926 0.48772236 0.59338101 ... 0.30309415 0.44105193 0.49243937]
 [0.41734159 0.48772236 0.59338101 ... 0.30309415 0.39642699 1.2087274 ]
 ...
 [0.41344658 0.48772236 0.11573841 ... 1.17646583 0.44105193 0.98304761]
 [0.40776407 0.48772236 0.11573841 ... 1.17646583 0.4032249  0.86530163]
 [0.41500016 0.48772236 0.11573841 ... 1.17646583 0.44105193 0.66905833]]
```

- By seeing these value we can't say where outlier exist so here we need to define a threshload.
- Let take Threshold > 3

In [6]:

```
# here we fliter out all the zscore value which are greater than 3 so these are outlier
threshold = 3
print(np.where(zscore > 3))
```

```
(array([ 55,  56,  57, 102, 141, 142, 152, 154, 155, 160, 162, 163, 199,
       200, 201, 202, 203, 204, 208, 209, 210, 211, 212, 216, 218, 219,
       220, 221, 222, 225, 234, 236, 256, 257, 262, 269, 273, 274, 276,
       277, 282, 283, 283, 284, 347, 351, 352, 353, 353, 354, 355, 356,
       357, 358, 363, 364, 364, 365, 367, 369, 370, 372, 373, 374, 374,
       380, 398, 404, 405, 406, 410, 410, 411, 412, 412, 414, 414, 415,
       416, 418, 418, 419, 423, 424, 425, 426, 427, 427, 429, 431, 436,
       437, 438, 445, 450, 454, 455, 456, 457, 466], dtype=int64), array([
1,  1,  1, 11, 12,  3,  3,  3,  3,  3,  3,  3,  1,  1,  1,  1,  1,
        1,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  5,  3,  3,  1,  5,
        5,  3,  3,  3,  3,  3,  3,  1,  3,  1,  1,  7,  7,  1,  7,  7,  7,
        3,  3,  3,  3,  3,  5,  5,  5,  3,  3,  3, 12,  5, 12,  0,  0,  0,
        0,  5,  0, 11, 11, 11, 12,  0, 12, 11, 11,  0, 11, 11, 11, 11, 11,
       11,  0, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11],
      dtype=int64))
```

- you see there are two array , the first array contains the list of row numbers and second array respective column numbers.
- These all are outlier

Outlier Prevention :

1. Drop Using Z_score Threshold

2. Using IQR

3. Log Transform

Method 1 : Drop

1st Drop the value which are greater than the threshlod of z_score

In [7]:

```
df_clean=df
df_clean = df_clean[(zscore<3).all(axis=1)]
```

In [8]:

```
df.shape,df_clean.shape
```

Out[8]:

```
((506, 13), (415, 13))
```

See here i drop the outlier which are greater than 3 so the data value reduce 506 to 415

Method 2 : Use IQR

- The interquartile range IQR tells us the range .where the bulk of the values lie. The interquartile range is calculated by subtracting the first quartile from the third quartile. IQR = Q3 - Q1

In [9]:

```
df_iqr = df
Q1 = df_iqr.quantile(0.25)
Q3 = df_iqr.quantile(0.75)
IQR = Q3-Q1
```

In [10]:

```
df_iqr = df_iqr[~(df_iqr < (Q1-1.5*IQR)) | (df_iqr > (Q3+1.5*IQR))]
```

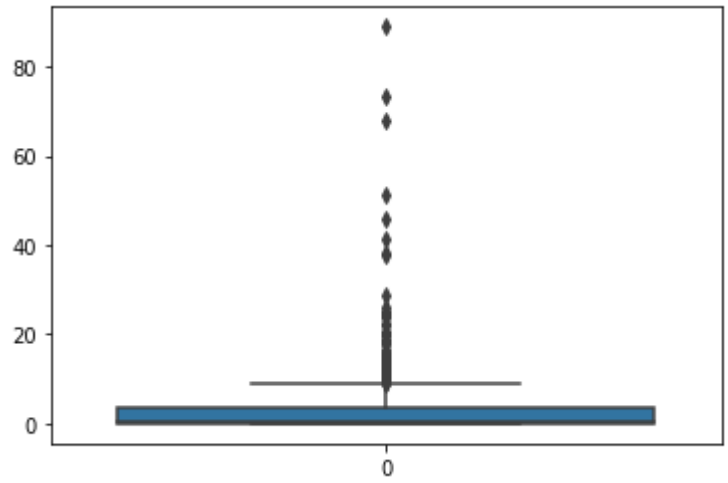
Method 3 :Log Transform

In [11]:

```
## before log transform
sns.boxplot(data=df['CRIM'])
```

Out[11]:

<AxesSubplot:>

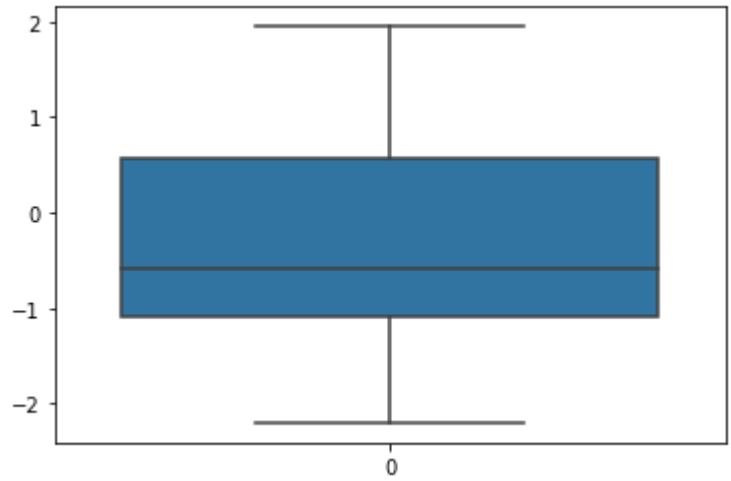


In [12]:

```
# after log transform
sns.boxplot(data=np.log10(df['CRIM']))
```

Out[12]:

<AxesSubplot:>



see after log transform all outlier gone.