In [1]:
```
import pandas as pd
import numpy as np
df = pd.read_excel('BostonHousing.xls','Data')
```

Finding Missing Values

In [2]:
```
df.dtypes
```

Out[2]:
```
CRIM       float64
ZN         float64
INDUS       object
CHAS         int64
NOX         object
RM         float64
AGE        float64
DIS         object
RAD          int64
TAX          int64
PTRATIO     object
dtype: object
```

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 167 entries, 0 to 166
Data columns (total 11 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     167 non-null    float64
 1   ZN       167 non-null    float64
 2   INDUS    147 non-null    object 
 3   CHAS     167 non-null    int64  
 4   NOX      165 non-null    object 
 5   RM       167 non-null    float64
 6   AGE      167 non-null    float64
 7   DIS      164 non-null    object 
 8   RAD      167 non-null    int64  
 9   TAX      167 non-null    int64  
 10  PTRATIO  167 non-null    object 
dtypes: float64(4), int64(3), object(4)
memory usage: 14.5+ KB
```

In [4]:
```
df = df.replace(' ', np.NaN)
df = df.replace('nan', np.NaN)
df = df.replace('****', np.NaN)
df = df.replace('*****', np.NaN)
df = df.replace('&&&', np.NaN)
df['INDUS'] = df['INDUS'].replace('Sara',np.NAN)
df.isna().sum()
```

Out[4]:
```
CRIM        0
ZN          0
INDUS      28
CHAS        0
NOX         6
RM          0
AGE         0
DIS         4
RAD         0
TAX         0
PTRATIO     0
dtype: int64
```

Highlighting Missing Values

In [5]:
```
df.style.highlight_null('red')
```

121	0.071650	0.000000	25.650000	0	0.581000	6.004000	84.100000	2.197400	2	188	19.100000
122	0.092990	0.000000	25.650000	0	0.581000	5.961000	92.900000	2.086900	2	188	19.100000
123	0.150380	0.000000	25.650000	0	0.581000	5.856000	97.000000	1.944400	2	188	19.100000
124	0.098490	0.000000	nan	0	0.581000	5.879000	95.800000	2.006300	2	188	19.100000
125	0.169020	0.000000	nan	0	0.581000	5.986000	88.400000	1.992900	2	188	19.100000
126	0.387350	0.000000	25.650000	0	0.581000	5.613000	95.600000	1.757200	2	188	19.100000
127	0.259150	0.000000	21.890000	0	0.624000	5.693000	96.000000	1.788300	4	437	21.200000
128	0.325430	0.000000	21.890000	0	0.624000	6.431000	98.800000	1.812500	4	437	21.200000
129	0.881250	0.000000	nan	0	0.624000	5.637000	94.700000	1.979900	4	437	21.200000
130	0.340060	0.000000	21.890000	0	0.624000	6.458000	98.900000	nan	4	437	21.200000
131	1.192940	0.000000	21.890000	0	0.624000	6.326000	97.700000	2.271000	4	437	21.200000
132	0.590050	0.000000	21.890000	0	0.624000	6.372000	97.900000	2.327400	4	437	21.200000
133	0.329820	0.000000	21.890000	0	0.624000	5.822000	95.400000	2.469900	4	437	21.200000

Finding Outliers in PTRATIO

In [6]:
```
df['PTRATIO'] = df['PTRATIO'].replace(' ',np.NAN)
df['PTRATIO'] = df['PTRATIO'].replace('Alina',np.NAN)
df['PTRATIO'] = df['PTRATIO'].replace('Adam',np.NAN)
df['PTRATIO'] = df['PTRATIO'].replace('##',np.NAN)
df.isna().sum()
```

Out[6]:
```
CRIM        0
ZN          0
INDUS      28
CHAS        0
NOX         6
RM          0
AGE         0
DIS         4
RAD         0
TAX         0
PTRATIO     3
dtype: int64
```

In [7]:
```
df.isna().sum()
```

Out[7]:
```
CRIM        0
ZN          0
INDUS      28
CHAS        0
NOX         6
RM          0
AGE         0
DIS         4
RAD         0
TAX         0
PTRATIO     3
dtype: int64
```

In [8]:
```
df.style.highlight_null('red')
```

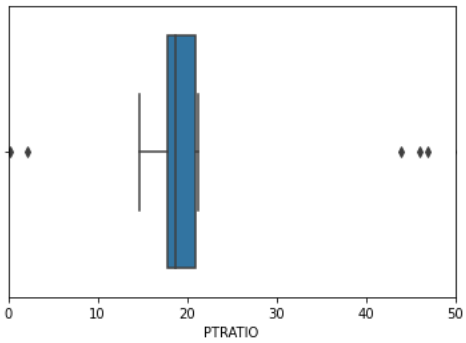
70	0.088260	0.000000	10.810000	0	0.413000	6.417000	6.600000	5.287300	4	305	19.200000
71	0.158760	0.000000	10.810000	0	0.413000	5.961000	17.500000	5.287300	4	305	19.200000
72	0.091640	0.000000	10.810000	0	0.413000	6.065000	7.800000	5.287300	4	305	19.200000
73	0.195390	0.000000	10.810000	0	0.413000	6.245000	6.200000	5.287300	4	305	19.200000
74	0.078960	0.000000	12.830000	0	0.437000	6.273000	6.000000	4.251500	5	398	177.000000
75	0.095120	0.000000	12.830000	0	0.437000	6.286000	45.000000	4.502600	5	398	18.700000
76	0.101530	0.000000	nan	0	0.437000	6.279000	74.500000	4.052200	5	398	18.700000
77	0.087070	0.000000	nan	0	0.437000	6.140000	45.800000	4.090500	5	398	18.700000
78	0.056460	0.000000	12.830000	0	0.437000	6.232000	53.700000	5.014100	5	398	18.700000
79	0.083870	0.000000	12.830000	0	0.437000	5.874000	36.600000	4.502600	5	398	18.700000
80	0.041130	25.000000	4.860000	0	0.426000	6.727000	33.500000	5.400700	4	281	19.000000
81	0.044620	25.000000	4.860000	0	0.426000	6.619000	70.400000	5.400700	4	281	19.000000
82	0.036590	25.000000	4.860000	0	0.426000	6.302000	32.200000	5.400700	4	281	19.000000

Creating a box plot for outliers

In [9]:
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.boxplot(df['PTRATIO'])
plt.xlim(0, 50)
```

```
C:\Users\deepa\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keywor
d arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an expli
cit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[9]: (0.0, 50.0)



In [10]:
```
q1 = df.quantile(0.25)
q3 = df.quantile(0.75)
iqr = q3 - q1
first_whisker = q1-1.5*iqr
last_whisker = q3+1.5*iqr
print(q1)
print(q3)
print(first_whisker)
print(last_whisker)
```

```
CRIM         0.06762
ZN           0.00000
INDUS        4.05000
CHAS         0.00000
NOX          0.44800
RM           5.88650
AGE         44.20000
DIS          2.70125
RAD          3.00000
TAX        263.00000
PTRATIO     17.80000
Name: 0.25, dtype: float64
CRIM         0.223505
ZN          12.500000
INDUS       10.010000
CHAS         0.000000
NOX          0.538000
RM           6.455000
AGE         92.050000
DIS          5.400900
RAD          5.000000
TAX        384.000000
PTRATIO     20.900000
Name: 0.75, dtype: float64
CRIM       -0.166208
ZN        -18.750000
INDUS      -4.890000
CHAS        0.000000
NOX         0.313000
RM          5.033750
AGE       -27.575000
DIS        -1.348225
RAD         0.000000
TAX        81.500000
PTRATIO    13.150000
dtype: float64
CRIM         0.457333
ZN          31.250000
INDUS       18.950000
CHAS         0.000000
NOX          0.673000
RM           7.307750
AGE        163.825000
DIS          9.450375
RAD          8.000000
TAX        565.500000
PTRATIO     25.550000
dtype: float64
```

Omission

In [13]:
```
reduced_df = df.dropna()
reduced_df.style.highlight_null('red')
```

16	0.053930	0.000000	8.140000	0	0.538000	5.935000	29.300000	4.498600	4	307	19.000000
17	0.784200	0.000000	8.140000	0	0.538000	5.990000	81.700000	4.257900	4	307	21.000000
18	0.802710	0.000000	8.140000	0	0.538000	5.456000	36.600000	3.796500	4	307	21.000000
19	0.725800	0.000000	8.140000	0	0.538000	5.727000	69.500000	3.796500	4	307	21.000000
20	1.251790	0.000000	8.140000	0	0.538000	5.570000	98.100000	3.797900	4	307	44.000000
21	0.852040	0.000000	8.140000	0	0.538000	5.965000	89.200000	4.012300	4	307	21.000000
22	0.232470	0.000000	8.140000	0	0.538000	6.142000	91.700000	3.976900	4	307	15.200000
23	0.988430	0.000000	8.140000	0	0.538000	5.813000	100.000000	4.095200	4	307	21.000000
25	0.840540	0.000000	8.140000	0	0.538000	5.599000	85.700000	4.454600	4	307	21.000000
26	0.671910	0.000000	8.140000	0	0.538000	5.813000	90.300000	4.682000	4	307	21.000000
27	0.955770	0.000000	8.140000	0	0.538000	6.047000	88.800000	4.453400	4	307	21.000000
28	0.772990	0.000000	8.140000	0	0.538000	6.495000	94.400000	4.454700	4	307	21.000000
30	0.130810	0.000000	8.140000	0	0.538000	5.713000	94.100000	4.233000	4	307	21.000000

Imputation

Imputation By Mean

In [14]:
```
df.fillna(df.mean(), inplace=True)
df.style.highlight_null('red')
```

Out[14]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO
0	0.006320	18.000000	2.310000	0	0.538000	6.575000	65.200000	4.090000	1	296	15.300000
1	0.027310	0.000000	7.070000	0	0.469000	6.421000	78.900000	4.967100	2	242	17.800000
2	0.027290	0.000000	7.070000	0	0.469000	7.185000	61.100000	4.967100	2	242	17.800000
3	0.032370	0.000000	2.180000	0	0.458000	6.998000	45.800000	6.062200	3	222	18.700000
4	0.069050	0.000000	7.070000	0	0.458000	7.147000	54.200000	4.169953	3	222	18.700000
5	0.029850	0.000000	9.122878	0	0.458000	6.430000	58.700000	6.062200	3	222	137.000000
6	0.088290	12.500000	7.070000	0	0.524000	6.012000	66.600000	5.560500	5	311	15.200000
7	0.144550	12.500000	9.122878	0	0.524000	6.172000	96.100000	5.950500	5	311	15.200000
8	0.211240	12.500000	7.870000	0	0.524000	5.631000	100.000000	6.082100	5	311	15.200000
9	0.170040	12.500000	9.122878	0	0.524000	6.004000	85.900000	6.592100	5	311	15.200000
10	0.224890	12.500000	7.870000	0	0.524000	6.377000	94.300000	6.346700	5	311	15.200000