EDA 07 Data Wrangling -II (Handling Outliers)

1. Necessary Imports

In [4]:
```
import pandas as pd
import numpy as np
from sklearn import datasets
from matplotlib import pyplot as plt
```

2. Loading and undersatnding the Dataset

In [104]:
```
boston=datasets.load_boston();
```

In [105]:
```
boston;
```

In [106]:
```
boston.data;
```

In [107]:
```
boston.target;
```

In [108]:
```
boston.feature_names;
```

In [109]:
```
boston.DESCR;
```

In [20]:
```
X=boston.data
Y=boston.target
columns=boston.feature_names
desc=boston.DESCR
```

In [21]:
```
boston_df=pd.DataFrame(X)
```

In [115]:
```
boston_df.head()
```

Out[115]:

	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

In [23]:
```
target_df=pd.DataFrame(Y)
```

In [110]:
```
target_df;
```

In [25]:
```
boston_df.columns=columns
```

In [111]:
```
boston_df;
```

In [27]:
```
target_df.columns=['MDEV']
```

In [28]:
```
data=pd.concat([boston_df,target_df],axis=1)
```

In [116]:
```
data.head()
```

Out[116]:

	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

In [112]:
```
desc;
```

In [31]:
```
desc=desc.split('\n')
```

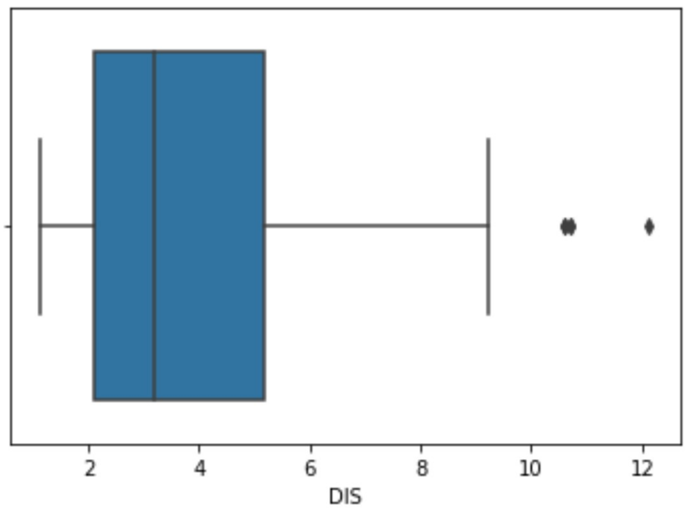
In [114]:
```
desc;
```

3. Outlier Detection

A) Univariate Analysis / Boxplots

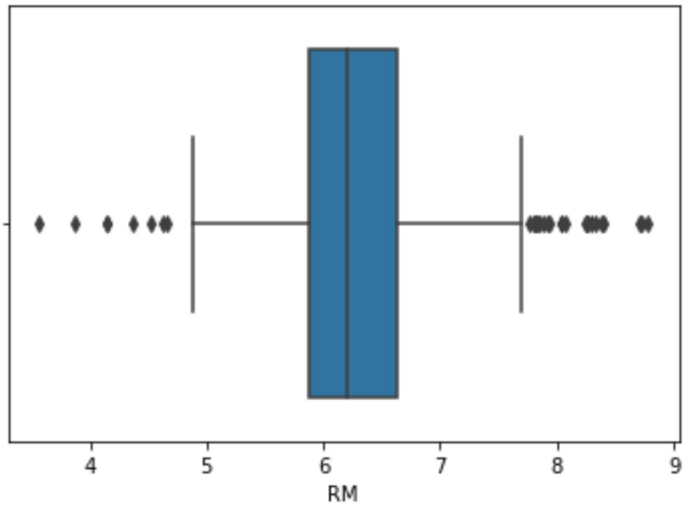
In [33]:
```
import seaborn as sns
sns.boxplot(x=data['DIS'])
```

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x22874c7d320>



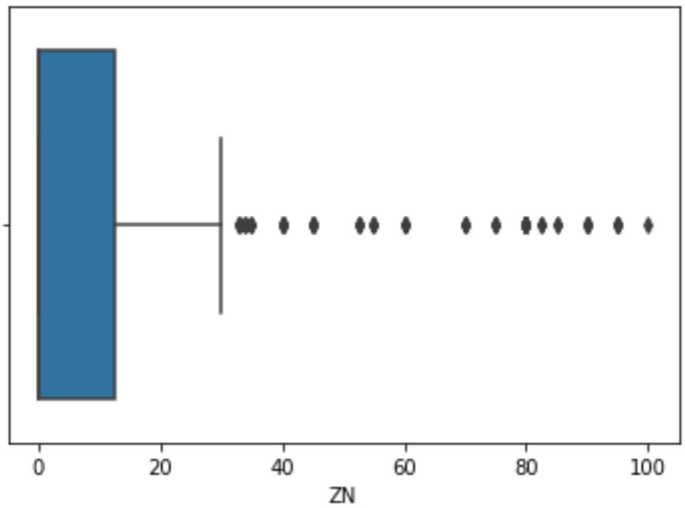
In [34]:
```
sns.boxplot(x=data['RM'])
```

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x22875376e80>



In [35]:
```
sns.boxplot(x=data['ZN'])
```

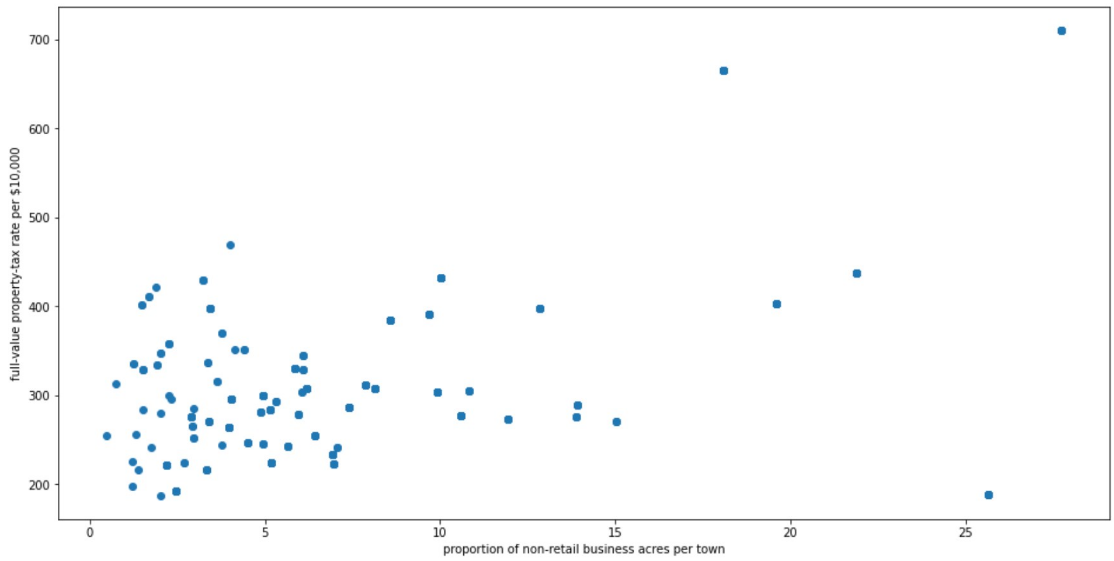
Out[35]: <matplotlib.axes._subplots.AxesSubplot at 0x2287543a208>



B). Bivariate Analysis Scatterplot

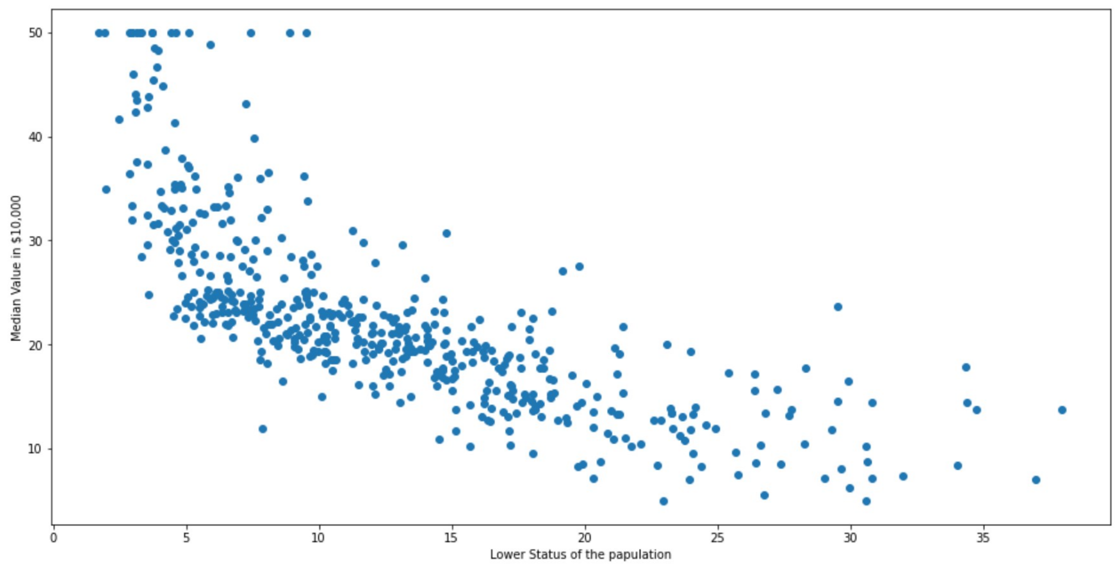
In [36]:
```
fig , ax=plt.subplots(figsize=(16,8))
ax.scatter(data['INDUS'],data['TAX'])
ax.set_xlabel("proportion of non-retail business acres per town")
ax.set_ylabel("full-value property-tax rate per $10,000")
```

Out[36]: Text(0, 0.5, 'full-value property-tax rate per $10,000')



In [38]:
```
fig , ax=plt.subplots(figsize=(16,8))
ax.scatter(data['LSTAT'],data['MDEV'])
ax.set_xlabel("Lower Status of the papulation")
ax.set_ylabel("Median Value in $10,000")
```

Out[38]: Text(0, 0.5, 'Median Value in $10,000')



C) Using Z-Score

In [53]:
```
from scipy import stats
z=np.abs(stats.zscore(data))
```

In [54]:
```
z
```

Out[54]:
```
array([[0.41978194, 0.28482986, 1.2879095 , ..., 0.44105193, 1.075
5623 ,
         0.15968566],
        [0.41733926, 0.48772236, 0.59338101, ..., 0.44105193, 0.492
43937,
         0.10152429],
        [0.41734159, 0.48772236, 0.59338101, ..., 0.39642699, 1.208
7274 ,
         1.32424667],
        ...,
        [0.41344658, 0.48772236, 0.11573841, ..., 0.44105193, 0.983
04761,
         0.14880191],
        [0.40776407, 0.48772236, 0.11573841, ..., 0.4032249 , 0.865
30163,
         0.0579893 ],
        [0.41500016, 0.48772236, 0.11573841, ..., 0.44105193, 0.669
05833,
         1.15724782]])
```

In [55]:
```
outliers=np.where(z>3)
```

In [56]:
```
outliers
```

Out[56]:
```
(array([ 55,  56,  57, 102, 141, 142, 152, 154, 155, 160, 162, 16
3, 199,
         200, 201, 202, 203, 204, 208, 209, 210, 211, 212, 216, 21
8, 219,
         220, 221, 222, 225, 234, 236, 256, 257, 262, 269, 273, 27
4, 276,
         277, 282, 283, 283, 284, 347, 351, 352, 353, 353, 354, 35
5, 356,
         357, 358, 363, 364, 364, 365, 367, 369, 370, 372, 373, 37
4, 374,
         380, 398, 404, 405, 406, 410, 410, 411, 412, 412, 414, 41
4, 415,
         416, 418, 418, 419, 423, 424, 425, 426, 427, 427, 429, 43
1, 436,
         437, 438, 445, 450, 454, 455, 456, 457, 466], dtype=int6
4),
 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, 1
1, 11, 11,
         11,  0, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 1
1],
       dtype=int64))
```

D) Using IQR

In [57]:
```
Q1=data.quantile(0.25)
Q3=data.quantile(0.75)
IQR=Q3-Q1
```

In [58]:
```
IQR
```

Out[58]:
```
CRIM          3.595038
ZN           12.500000
INDUS        12.910000
CHAS          0.000000
NOX           0.175000
RM            0.738000
AGE          49.050000
DIS           3.088250
RAD          20.000000
TAX         387.000000
PTRATIO       2.800000
B            20.847500
LSTAT        10.005000
MDEV          7.975000
dtype: float64
```

In [59]:
```
((data<(Q1-1.5*IQR))| (data>(Q3+1.5*IQR))).sum()
```

Out[59]:
```
CRIM       66
ZN         68
INDUS       0
CHAS       35
NOX         0
RM         30
AGE         0
DIS         5
RAD         0
TAX         0
PTRATIO    15
B          77
LSTAT       7
MDEV       40
dtype: int64
```

In [60]:
```
data2=data.copy()
```

In [61]:
```
data.shape
```

Out[61]:
```
(506, 14)
```

4. Handling Outliers

A) Removing Outliers

Using Z-Score

In [66]:
```
data2=data[(z<3).all(axis=1)]
```

All values in a record should be having z-values less than three. Not even a single column should have an outlier.

In [67]:
```
data2.shape
```

Out[67]:
```
(415, 14)
```

Using IQR

In [42]:
```
data3=data.copy()
```

In [68]:
```
data3=data[((data>=(Q1-1.5*IQR))& (data<=(Q3+1.5*IQR))).all(axis=1)]
```

All values in a records should be in between IQR threshold

In [46]:
```
data3.shape
```

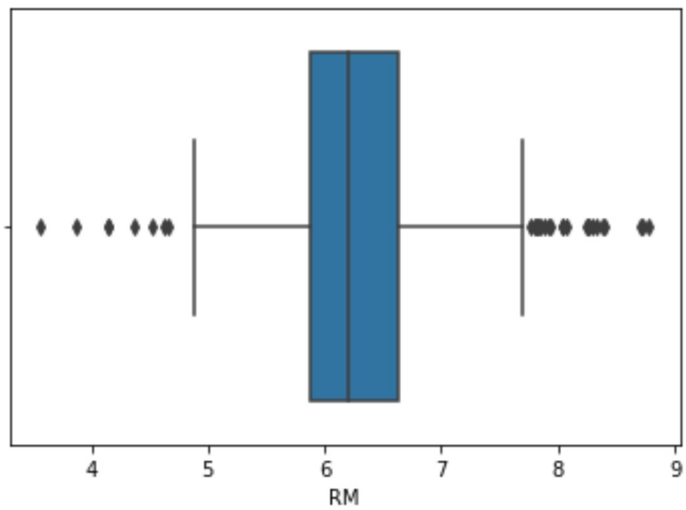
Out[46]:
```
(268, 14)
```

5. Replacing Outliers

Using IQR

In [69]:
```
sns.boxplot(x=data['RM'])
```

Out[69]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2287576f208>
```



In [70]:
```
Q1=data['RM'].quantile(0.25)
Q3=data['RM'].quantile(0.75)
IQR=Q3-Q1
```

In [71]:
```
IQR
```

Out[71]: 0.7379999999999995

In [72]:
```
(data['RM']<(Q1-1.5*IQR)).sum()
```

Out[72]: 8

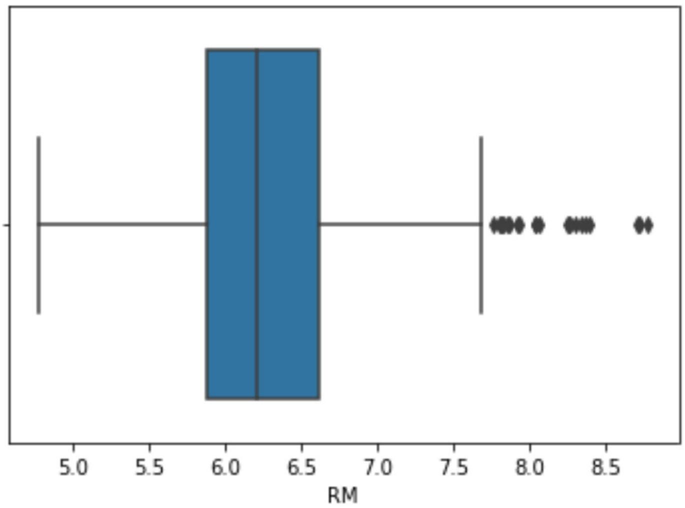
In [73]:
```
(data['RM']>(Q3+1.5*IQR)).sum()
```

Out[73]: 22

In [74]:
```
data['RM'] = np.where(data['RM'] <(Q1-1.5*IQR),Q1-1.5*IQR ,data['RM
'])
```

In [75]:
```
sns.boxplot(x=data['RM'])
```

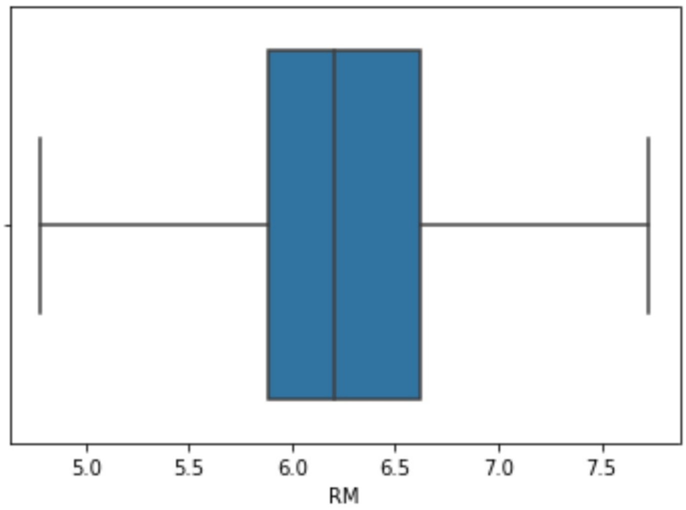
Out[75]: <matplotlib.axes._subplots.AxesSubplot at 0x22875a3c4e0>



In [76]:
```
data['RM'] = np.where(data['RM']>(Q3+1.5*IQR),Q3+1.5*IQR ,data['RM
'])
```

In [77]:
```
sns.boxplot(x=data['RM'])
```

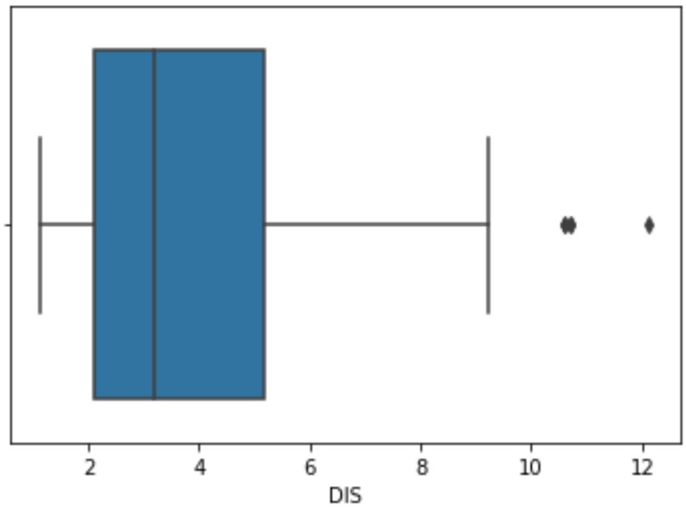
Out[77]: <matplotlib.axes._subplots.AxesSubplot at 0x22875420748>



Using Z Score

In [82]:
```
sns.boxplot(x=boston_df['DIS'])
```

Out[82]: <matplotlib.axes._subplots.AxesSubplot at 0x22875b55828>

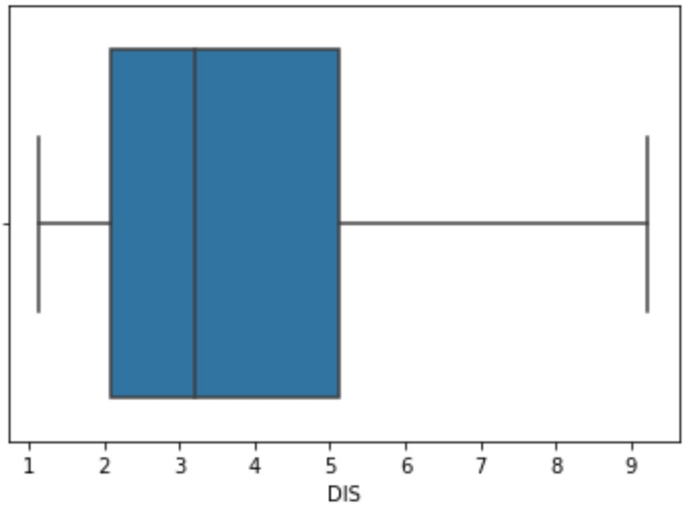


In [83]:
```
z = np.abs(stats.zscore(data['DIS']))
```

In [102]:
```
data['DIS'] = np.where(z>3,data['DIS'].mean(),data['DIS'])
```

In [103]:
```
sns.boxplot(x=data['DIS'])
```

Out[103]: <matplotlib.axes._subplots.AxesSubplot at 0x228761085f8>



In [ ]: