Assignment 1- Group 6

Importing and review the data frame

In [1]:
```
import pandas as pd
import numpy as np
import re
#Load data (bostonhousing csv file)
df1 = pd.read_excel ("BostonHousing.xls",sheet_name="Data")
df1.sample(10)
```

Out[1]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO
30	0.13081	0.0	8.14	0	0.538	5.713	94.1	4.233	4	307	21
156	0.09103	0.0	2.46	0	0.488	7.155	92.2	2.7006	3	193	17.8
39	0.02763	75.0	2.95	0	0.428	6.595	21.8	5.4011	3	252	18.3
56	0.02055	85.0	0.74	0	0.41	6.383	35.7	9.1876	2	313	17.3
141	1.62864	0.0	21.89	0	0.624	5.019	100.0	1.4394	4	437	21.2
93	0.02875	28.0	15.04	0	0.464	6.211	28.9	3.6659	4	270	18.2
154	0.06588	0.0	2.46	0	0.488	7.765	83.3	2.741	3	193	17.8
100	0.14866	0.0	8.56	0	0.52	6.727	79.9	2.7778	5	384	Alina
22	0.23247	0.0	8.14	0	0.538	6.142	91.7	3.9769	4	307	15.2
53	0.04981	21.0	5.64	0	0.439	5.998	21.4	7.3197	4	243	16.8

In [2]:
```
df1["PTRATIO"].describe()
```

Out[2]:
```
count     167.0
unique     36.0
top        17.8
freq       21.0
Name: PTRATIO, dtype: float64
```

In [3]:
```
df1['INDUS'] = df1['INDUS'].astype(str)
df1['PTRATIO'] = df1['PTRATIO'].astype(str)
```

In [4]:
```
df1.dtypes
```

Out[4]:
```
CRIM        float64
ZN          float64
INDUS        object
CHAS          int64
NOX          object
RM          float64
AGE         float64
DIS          object
RAD           int64
TAX           int64
PTRATIO      object
dtype: object
```

Highlight Missing cells In Yellow:

Part B.1 - Highlight Missing Cells In yellow

In [5]:
```
#Making a new data frame object with data type as string

df = df1.applymap(str)
```

In [6]:
```
# Defining a function to Highlight Non numeric Values in Yellow.

def highlight_cells(data):
 result = re.findall('\d+\.\d+|\d+', data)
 color = 'yellow' if not result else ''
 return 'background-color: {}'.format(color)
```

In [7]:
```
result = df.drop(['PTRATIO'], axis=1).style.applymap(highlight_cells)
```

In [8]:
```
result
```

Out[8]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.09	1	296
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222
4	0.06905	0.0	7.07	0	0.458	7.147	54.2		3	222
5	0.02985	0.0	****	0	0.458	6.43	58.7	6.0622	3	222
6	0.08829	12.5	7.07	0	0.524	6.012	66.6	5.5605	5	311
7	0.14455	12.5	****	0	0.524	6.172	96.1	5.9505	5	311
8	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311
9	0.17004	12.5	****	0	0.524	6.004	85.9	6.5921	5	311
10	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311
11	0.11747	12.5	nan	0	0.524	6.009	82.9	6.2267	5	311
12	0.09378	12.5	7.87	0	0.524	5.889	39.0	5.4509	5	311
13	0.62976	0.0	8.14	0	nan	5.949	61.8	4.7075	4	307
14	0.63796	0.0	8.14	0	0.538	6.096	84.5	4.4619	4	307
15	0.62739	0.0	8.14	0	0.538	5.834	56.5	4.4986	4	307
16	0.05393	0.0	8.14	0	0.538	5.935	29.3	4.4986	4	307
17	0.7842	0.0	8.14	0	0.538	5.99	81.7	4.2579	4	307
18	0.80271	0.0	8.14	0	0.538	5.456	36.6	3.7965	4	307
19	0.7258	0.0	8.14	0	0.538	5.727	69.5	3.7965	4	307
20	1.25179	0.0	8.14	0	0.538	5.57	98.1	3.7979	4	307
21	0.85204	0.0	8.14	0	0.538	5.965	89.2	4.0123	4	307
22	0.23247	0.0	8.14	0	0.538	6.142	91.7	3.9769	4	307
23	0.98843	0.0	8.14	0	0.538	5.813	100.0	4.0952	4	307
24	0.75026	0.0	nan	0	0.538	5.924	94.1	4.3996	4	307
25	0.84054	0.0	8.14	0	0.538	5.599	85.7	4.4546	4	307
26	0.67191	0.0	8.14	0	0.538	5.813	90.3	4.682	4	307
27	0.95577	0.0	8.14	0	0.538	6.047	88.8	4.4534	4	307
28	0.77299	0.0	8.14	0	0.538	6.495	94.4	4.4547	4	307
29	0.10245	0.0	Sara	0	0.538	6.674	87.3	4.239	4	307
30	0.13081	0.0	8.14	0	0.538	5.713	94.1	4.233	4	307
31	1.35472	0.0	8.14	0	0.538	6.072	100.0	4.175	4	307
32	0.138799	0.0	8.14	0	0.538	5.95	82.0	3.99	4	307

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX
33	0.15172	0.0	8.14	0	0.538	5.701	95.0	3.7872	4	307
34	1.61282	0.0	8.14	0	0.538	6.096	96.9	3.7598	4	307
35	0.06417	0.0	5.96	0	0.499	5.933	68.2	3.3603	5	279
36	0.09744	0.0	5.96	0	0.499	5.841	61.4	3.3779	5	279
37	0.08014	0.0	nan	0	0.499	5.85	41.5	3.9342	5	279
38	0.17505	0.0	nan	0	0.499	5.966	30.2	3.8473	5	279
39	0.02763	75.0	2.95	0	0.428	6.595	21.8	5.4011	3	252
40	0.03359	75.0	nan	0	0.428	7.024	15.8	5.4011	3	252
41	0.12744	0.0	2.95	0	0.448	6.77	2.9	5.7209	3	233
42	0.1415	0.0	6.91	0	0.448	6.169	6.6	5.7209	3	233
43	0.15936	0.0	6.91	0	0.448	6.211	6.5	5.7209	3	233
44	0.12269	0.0	6.91	0	0.448	6.069	40.0	5.7209	3	233
45	0.17142	0.0	6.91	0	0.448	5.682	33.8	5.1004	3	233
46	0.18836	0.0	6.91	0	0.448	5.786	33.3	5.1004	3	233
47	0.22927	0.0	6.91	0	0.448	6.03	85.5	5.6894	3	233
48	0.25387	0.0	6.91	0	0.448	5.399	95.3	5.87	3	233
49	0.21977	0.0	6.91	0	0.448	5.602	62.0	6.0877	3	233
50	0.08873	21.0	nan	0	0.439	5.963	45.7	6.8147	4	243
51	0.04337	21.0	5.64	0	0.439	6.115	63.0	6.8147	4	243
52	0.0536	21.0	5.64	0	0.439	6.511	21.1	6.8147	4	243
53	0.04981	21.0	5.64	0	0.439	5.998	21.4	7.3197	4	243
54	0.0136	75.0	4	0	0.41	5.888	47.6	7.3197	3	469
55	0.01311	90.0	1.22	0	0.403	7.249	21.9	8.6966	5	226
56	0.02055	85.0	0.74	0	0.41	6.383	35.7	9.1876	2	313
57	0.01432	100.0	1.32	0	0.411	6.816	40.5	8.3248	5	256
58	0.15445	25.0	5.13	0	0.453	6.145	29.2	7.8148	8	284
59	0.10328	25.0	nan	0	0.453	5.927	47.2	6.932	8	284
60	0.14932	25.0	****	0	0.453	5.741	66.2	7.2254	8	284
61	0.17171	25.0	nan	0	0.453	5.966	93.4	6.8185	8	284
62	0.11027	25.0	5.13	0	0.453	6.456	67.8	7.2255	8	284
63	0.1265	25.0	5.13	0	0.453	6.762	43.4	7.9809	8	284
64	0.01951	17.5	1.38	0	0.4161	7.104	59.5	9.2229	3	216
65	0.03584	80.0	3.37	0	0.398	6.29	17.8	6.6115	4	337
66	0.04379	80.0	3.37	0	0.398	5.787	31.1	6.6115	4	337

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX
67	0.05789	12.5	3.4	0	0.409	5.878	21.4	6.498	4	345
68	0.13554	12.5	****	0	0.409	5.594	36.8	6.498	4	345
69	0.12816	12.5	****	0	0.409	5.885	33.0	6.498	4	345
70	0.08826	0.0	10.81	0	0.413	6.417	6.6	5.2873	4	305
71	0.15876	0.0	10.81	0	0.413	5.961	17.5	5.2873	4	305
72	0.09164	0.0	10.81	0	0.413	6.065	7.8	5.2873	4	305
73	0.19539	0.0	10.81	0	0.413	6.245	6.2	5.2873	4	305
74	0.07896	0.0	12.83	0	0.437	6.273	6.0	4.2515	5	398
75	0.09512	0.0	12.83	0	0.437	6.286	45.0	4.5026	5	398
76	0.10153	0.0	nan	0	0.437	6.279	74.5	4.0522	5	398
77	0.08707	0.0	nan	0	0.437	6.14	45.8	4.0905	5	398
78	0.05646	0.0	12.83	0	0.437	6.232	53.7	5.0141	5	398
79	0.08387	0.0	12.83	0	0.437	5.874	36.6	4.5026	5	398
80	0.04113	25.0	4.86	0	0.426	6.727	33.5	5.4007	4	281
81	0.04462	25.0	4.86	0	0.426	6.619	70.4	5.4007	4	281
82	0.03659	25.0	4.86	0	0.426	6.302	32.2	5.4007	4	281
83	0.03551	25.0	nan	0	0.426	6.167	46.7	5.4007	4	281
84	0.05059	0.0	nan	0	*****	6.389	48.0	4.7794	3	247
85	0.05735	0.0	4.49	0	0.449	6.63	56.1	4.4377	3	247
86	0.05188	0.0	4.49	0	0.449	6.015	45.1	4.4272	3	247
87	0.07151	0.0	4.49	0	0.449	6.121	56.8	3.7476	3	247
88	0.0566	0.0	3.41	0	0.489	7.007	86.3	nan	2	270
89	0.05302	0.0	3.41	0	0.489	7.079	63.1	3.4145	2	270
90	0.04684	0.0	3.41	0	0.489	6.417	66.1	3.0923	2	270
91	0.03932	0.0	3.41	0	0.489	6.405	73.9	3.0921	2	270
92	0.04203	28.0	15.04	0	0.464	6.442	53.6	3.6659	4	270
93	0.02875	28.0	15.04	0	0.464	6.211	28.9	3.6659	4	270
94	0.04294	28.0	15.04	0	0.464	6.249	77.3	3.615	4	270
95	0.12204	0.0	2.89	0	0.445	6.625	57.8	3.4952	2	276
96	0.11504	0.0	2.89	0	*****	6.163	69.6	3.4952	2	276
97	0.12083	0.0	2.89	0	0.445	8.069	76.0	3.4952	2	276
98	0.08187	0.0	2.89	0	0.445	7.82	36.9	3.4952	2	276
99	0.0686	0.0	2.89	0	0.445	7.416	62.5	3.4952	2	276
100	0.14866	0.0	8.56	0	0.52	6.727	79.9	2.7778	5	384

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX
101	0.11432	0.0	8.56	0	0.52	6.781	71.3	2.8561	5	384
102	0.22876	0.0	nan	0	0.52	6.405	85.4	2.7147	5	384
103	0.21161	0.0	8.56	0	0.52	6.137	87.4	2.7147	5	384
104	0.1396	0.0	8.56	0	0.52	6.167	90.0	2.421	5	384
105	0.13262	0.0	8.56	0	0.52	5.851	96.7	2.1069	5	384
106	0.1712	0.0	8.56	0	0.52	5.836	91.9	2.211	5	384
107	0.13117	0.0	8.56	0	0.52	6.127	85.2	2.1224	5	384
108	0.12802	0.0	8.56	0	0.52	6.474	97.1	2.4329	5	384
109	0.26363	0.0	8.56	0	0.52	6.229	91.2	2.5451	5	384
110	0.10793	0.0	nan	0	0.52	6.195	54.4	2.7778	5	384
111	0.10084	0.0	10.01	0	*****	6.715	81.6	2.6775	6	432
112	0.12329	0.0	10.01	0	0.547	5.913	92.9	2.3534	6	432
113	0.22212	0.0	10.01	0	0.547	6.092	95.4	2.548	6	432
114	0.14231	0.0	10.01	0	0.547	6.254	84.2	2.2565	6	432
115	0.17134	0.0	10.01	0	0.547	5.928	88.2	2.4631	6	432
116	0.13158	0.0	10.01	0	0.547	6.176	72.5	2.7301	6	432
117	0.15098	0.0	10.01	0	0.547	6.021	82.6	2.7474	6	432
118	0.13058	0.0	*****	0	0.547	5.872	73.1	2.4775	6	432
119	0.14476	0.0	10.01	0	0.547	5.731	65.2	2.7592	6	432
120	0.06899	0.0	25.65	0	0.581	5.87	69.7	2.2577	2	188
121	0.07165	0.0	25.65	0	0.581	6.004	84.1	2.1974	2	188
122	0.09299	0.0	25.65	0	0.581	5.961	92.9	2.0869	2	188
123	0.15038	0.0	25.65	0	0.581	5.856	97.0	1.9444	2	188
124	0.09849	0.0	nan	0	0.581	5.879	95.8	2.0063	2	188
125	0.16902	0.0	nan	0	0.581	5.986	88.4	1.9929	2	188
126	0.38735	0.0	25.65	0	0.581	5.613	95.6	1.7572	2	188
127	0.25915	0.0	21.89	0	0.624	5.693	96.0	1.7883	4	437
128	0.32543	0.0	21.89	0	0.624	6.431	98.8	1.8125	4	437
129	0.88125	0.0	nan	0	0.624	5.637	94.7	1.9799	4	437
130	0.34006	0.0	21.89	0	0.624	6.458	98.9	nan	4	437
131	1.19294	0.0	21.89	0	0.624	6.326	97.7	2.271	4	437
132	0.59005	0.0	21.89	0	0.624	6.372	97.9	2.3274	4	437
133	0.32982	0.0	21.89	0	0.624	5.822	95.4	2.4699	4	437
134	0.97617	0.0	21.89	0	0.624	5.757	98.4	2.346	4	437

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX
135	0.55778	0.0	21.89	0	0.624	6.335	98.2	2.1107	4	437
136	0.32264	0.0	21.89	0	0.624	5.942	93.5	1.9669	4	437
137	0.35233	0.0	21.89	0	0.624	6.454	98.4	1.8498	4	437
138	0.2498	0.0	21.89	0	0.624	5.857	98.2	1.6686	4	437
139	0.54452	0.0	21.89	0	0.624	6.151	97.9	1.6687	4	437
140	0.2909	0.0	21.89	0	0.624	6.174	93.6	1.6119	4	437
141	1.62864	0.0	21.89	0	0.624	5.019	100.0	1.4394	4	437
142	3.32105	0.0	19.58	1	&&&	5.403	100.0	1.3216	5	403
143	4.0974	0.0	19.58	0	0.871	5.468	100.0	1.4118	5	403
144	2.77974	0.0	19.58	0	0.871	4.903	97.8	1.3459	5	403
145	2.37934	0.0	19.58	0	0.871	6.13	100.0	1.4191	5	403
146	0.13914	0.0	4.05	0	0.51	5.572	88.5	2.5961	5	296
147	0.09178	0.0	nan	0	0.51	6.416	84.1	2.6463	5	296
148	0.08447	0.0	4.05	0	0.51	5.859	68.7	2.7019	5	296
149	0.06664	0.0	4.05	0	0.51	6.546	33.1	3.1323	5	296
150	0.07022	0.0	4.05	0	0.51	6.02	47.2	3.5549	5	296
151	0.05425	0.0	4.05	0	0.51	6.315	73.4	3.3175	5	296
152	0.06642	0.0	4.05	0	0.51	6.86	74.4	2.9153	5	296
153	0.0578	0.0	2.46	0	0.488	6.98	58.4	2.829	3	193
154	0.06588	0.0	2.46	0	0.488	7.765	83.3	2.741	3	193
155	0.06888	0.0	nan	0	0.488	6.144	62.2	2.5979	3	193
156	0.09103	0.0	2.46	0	0.488	7.155	92.2	2.7006	3	193
157	0.10008	0.0	2.46	0	0.488	6.563	95.6	2.847	3	193
158	0.08308	0.0	2.46	0	0.488	5.604	89.8	2.9879	3	193
159	0.06047	0.0	2.46	0	0.488	6.153	68.8	3.2797	3	193
160	0.05602	0.0	2.46	0	nan	7.831	53.6	nan	3	193
161	0.07875	45.0	3.44	0	0.437	6.782	41.1	3.7886	5	398
162	0.12579	45.0	3.44	0	0.437	6.556	29.1	4.5667	5	398
163	0.0837	45.0	3.44	0	0.437	7.185	38.9	4.5667	5	398
164	0.09068	45.0	nan	0	0.437	6.951	21.5	6.4798	5	398
165	0.06911	45.0	3.44	0	0.437	6.739	30.8	6.4798	5	398
166	0.08664	45.0	3.44	0	0.437	7.178	26.3	6.4798	5	398

In [9]:
```
df1.drop(['PTRATIO'], axis=1)
```

Out[9]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.09	1	296
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222
4	0.06905	0.0	7.07	0	0.458	7.147	54.2		3	222
...	...	...	...	...	...	...	...	...	...	...
162	0.12579	45.0	3.44	0	0.437	6.556	29.1	4.5667	5	398
163	0.08370	45.0	3.44	0	0.437	7.185	38.9	4.5667	5	398
164	0.09068	45.0	nan	0	0.437	6.951	21.5	6.4798	5	398
165	0.06911	45.0	3.44	0	0.437	6.739	30.8	6.4798	5	398
166	0.08664	45.0	3.44	0	0.437	7.178	26.3	6.4798	5	398

167 rows × 10 columns

Part B.2.2) : Outliers in PTRATIO:

In [10]:

```
    display (df1['PTRATIO'][pd.to_numeric(df1['PTRATIO'], errors='coerce').isnull()])
```

```
100    Alina
110       ##
117     Adam
Name: PTRATIO, dtype: object
```

In [11]:

```
#substituting cells with numeric value:
df1.at[110,'PTRATIO']=0
df1.at[100,'PTRATIO']=0
df1.at[117,'PTRATIO']=0

df1['PTRATIO'].sample(10)
```

Out[11]:

```
89     17.8
101    20.9
110       0
151    16.6
132    21.2
118    17.8
145    50.3
27       21
63     19.7
134    21.2
Name: PTRATIO, dtype: object
```

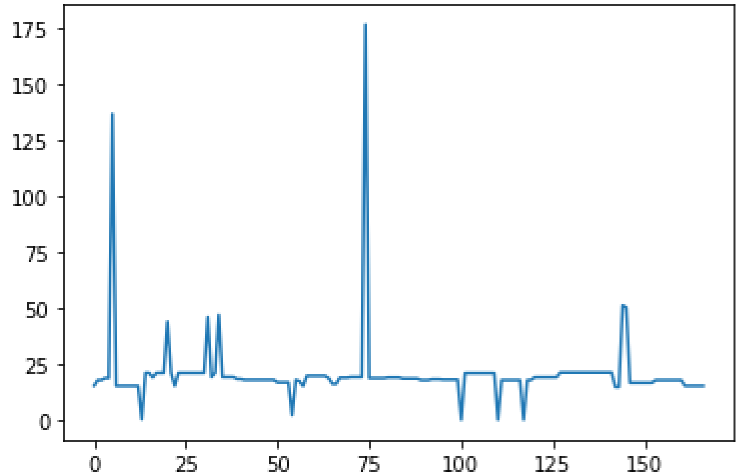
In [12]:

```
#finding more outliers using plots :

df1['PTRATIO'] = df1['PTRATIO'].astype(float)
df1['PTRATIO'].plot()
```

Out[12]:

```
<AxesSubplot:>
```



As we can see from the graph above, there are a 2 outlier values in the PTRATIO Column

In [13]:
```
df1['PTRATIO'].describe()
```

Out[13]:
```
count    167.000000
mean      20.562515
std       16.451836
min        0.000000
25%       17.800000
50%       18.600000
75%       20.900000
max      177.000000
Name: PTRATIO, dtype: float64
```

In [14]:
```
sorted(df1['PTRATIO'], reverse=True)
```

Out[14]:
```
[177.0,
 137.0,
 51.3,
 50.3,
 47.0,
 46.0,
 44.0,
 21.2,
 21.2,
 21.2,
 21.2,
 21.2,
 21.2,
 21.2,
 21.2,
 21.2,
 21.2,
 21.2,
 21.2,
 21.2,
 21.2,
 21.2,
 21.0,
 21.0,
 21.0,
 21.0,
 21.0,
 21.0,
 21.0,
 21.0,
 21.0,
 21.0,
 21.0,
 21.0,
 21.0,
 21.0,
 21.0,
 20.9,
 20.9,
 20.9,
 20.9,
 20.9,
 20.9,
 20.9,
 20.9,
 20.9,
 19.7,
 19.7,
 19.7,
 19.7,
 19.7,
 19.7,
 19.2,
 19.2,
 19.2,
 19.2,
 19.2,
 19.2,
 19.2,
 19.2,
```

```
19.2,
19.1,
19.1,
19.1,
19.1,
19.1,
19.1,
19.1,
19.0,
19.0,
19.0,
19.0,
19.0,
18.9,
18.9,
18.9,
18.7,
18.7,
18.7,
18.7,
18.7,
18.7,
18.7,
18.6,
18.5,
18.5,
18.5,
18.5,
18.5,
18.3,
18.3,
18.2,
18.2,
18.2,
18.0,
18.0,
18.0,
18.0,
18.0,
17.9,
17.9,
17.9,
17.9,
17.9,
17.9,
17.9,
17.9,
17.9,
17.9,
17.8,
17.8,
17.8,
17.8,
17.8,
17.8,
17.8,
17.8,
17.8,
17.8,
17.8,
```

```
 17.8,
 17.8,
 17.8,
 17.8,
 17.8,
 17.8,
 17.8,
 17.8,
 17.8,
 17.8,
 17.3,
 16.8,
 16.8,
 16.8,
 16.8,
 16.6,
 16.6,
 16.6,
 16.6,
 16.6,
 16.6,
 16.6,
 16.1,
 16.1,
 15.3,
 15.2,
 15.2,
 15.2,
 15.2,
 15.2,
 15.2,
 15.2,
 15.2,
 15.2,
 15.2,
 15.2,
 15.2,
 15.2,
 15.2,
 15.1,
 14.7,
 14.7,
 2.11,
 0.23,
 0.0,
 0.0,
 0.0]
```

values 177 and 137 appear to be outliers from the PTRATIO column these might be because of a decimal error or a genuine one

Python Code to Implement Omission and Imputation:

Part C.1

In [15]:
```
df1.replace(" ", np.nan, inplace=True)
df1.head()
```

Out[15]:

	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	15.3
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7
4	0.06905	0.0	7.07	0	0.458	7.147	54.2	NaN	3	222	18.7

In [16]:
```
print('Number of cells after replacing blanks with Nan: ')
df1.isna().sum()
```

```
Number of cells after replacing blanks with Nan:
```

Out[16]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        2
RM         0
AGE        0
DIS        4
RAD        0
TAX        0
PTRATIO    0
dtype: int64
```

Python code for Omission :

Part c.2

In [17]:
```
reduced_df = df1.dropna()
print('Number of rows after removing rows with missing values: ', len(reduced_df))
```

```
Number of rows after removing rows with missing values:  162
```

Python Code for Imputation

In [18]:
```
    # displaying the null values for columns where they exist:
    display (df1['INDUS'][pd.to_numeric(df1['INDUS'], errors='coerce').isnull()])
    display (df1['DIS'][pd.to_numeric(df1['DIS'], errors='coerce').isnull()])
    display (df1['NOX'][pd.to_numeric(df1['NOX'], errors='coerce').isnull()])
```

```
5       ****
7       ****
9       ****
11       nan
24       nan
29      Sara
37       nan
38       nan
40       nan
50       nan
59       nan
60      ****
61       nan
68      ****
69      ****
76       nan
77       nan
83       nan
84       nan
102      nan
110      nan
118    *****
124      nan
125      nan
129      nan
147      nan
155      nan
164      nan
Name: INDUS, dtype: object
4     NaN
88    NaN
130   NaN
160   NaN
Name: DIS, dtype: float64
13       NaN
84     *****
96     *****
111    *****
142      &&&
160      NaN
Name: NOX, dtype: object
```

In [19]:
```
df1['INDUS'].replace("*****", np.NaN, inplace=True)
df1['INDUS'].replace("****", np.NaN, inplace=True)
df1['INDUS'].replace("Sara", np.NaN, inplace=True)
df1['INDUS'].replace("nan", np.NaN, inplace=True)


display (df1['INDUS'][pd.to_numeric(df1['INDUS'], errors='coerce').isnull()])
```

```
5      NaN
7      NaN
9      NaN
11     NaN
24     NaN
29     NaN
37     NaN
38     NaN
40     NaN
50     NaN
59     NaN
60     NaN
61     NaN
68     NaN
69     NaN
76     NaN
77     NaN
83     NaN
84     NaN
102    NaN
110    NaN
118    NaN
124    NaN
125    NaN
129    NaN
147    NaN
155    NaN
164    NaN
Name: INDUS, dtype: object
```

In [20]:

```
df1['INDUS'] = df1['INDUS'].astype(float)
```

In [21]:

```
# imputation of NaN cells with mean Values:

df1['INDUS'].mean()
df1['INDUS']=df1['INDUS'].fillna(df1['INDUS'].mean())
df1['INDUS'].sample(20)
```

Out[21]:

```
40      9.122878
54      4.000000
129     9.122878
97      2.890000
123    25.650000
30      8.140000
59      9.122878
19      8.140000
26      8.140000
113    10.010000
53      5.640000
115    10.010000
9       9.122878
116    10.010000
7       9.122878
80      4.860000
72     10.810000
106     8.560000
149     4.050000
2       7.070000
Name: INDUS, dtype: float64
```