In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv('D:\Documents\College\DSBDA\Banglore Housing Prices.csv')
df
```

Out[2]:

	location	size	total_sqft	bath	price
0	Electronic City Phase II	2 BHK	1056	2.0	39.07
1	Chikka Tirupathi	4 Bedroom	2600	5.0	120.00
2	Uttarahalli	3 BHK	1440	2.0	62.00
3	Lingadheeranahalli	3 BHK	1521	3.0	95.00
4	Kothanur	2 BHK	1200	2.0	51.00
...	...	...	...	...	...
13315	Whitefield	5 Bedroom	3453	4.0	231.00
13316	Richards Town	4 BHK	3600	5.0	400.00
13317	Raja Rajeshwari Nagar	2 BHK	1141	2.0	60.00
13318	Padmanabhanagar	4 BHK	4689	4.0	488.00
13319	Doddathoguru	1 BHK	550	1.0	17.00

13320 rows × 5 columns

Replacing Null Values

In [3]:
```
df.isnull().sum()
```

Out[3]:
```
location       1
size          16
total_sqft     0
bath          73
price          0
dtype: int64
```

In [4]:
```
df.dropna(subset=["location"], inplace=True)
```

In [5]:
```
mode_size = df['size'].mode()[0]
df['size'].fillna(mode_size, inplace=True)
```

In [6]:
```
mode_bath = df['bath'].mode()[0]
df['bath'].fillna(mode_bath, inplace=True)
```

In [7]:
```
df['bath'] = df['bath'].astype(float)
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
location      0
size          0
total_sqft    0
bath          0
price         0
dtype: int64
```

Transforming size column to numeric value

In [9]:
```
df.dtypes
```

Out[9]:
```
location       object
size           object
total_sqft     object
bath          float64
price         float64
dtype: object
```

In [10]:
```
def transform(x):
    if isinstance(x, str):
        return x.split()[0]
    else:
        return x
```

In [11]:
```
df['size'] = df['size'].apply(transform)
```

In [12]:
```
df['size'] = df['size'].astype(float)
```

In [13]:
```
df.dtypes
```

Out[13]:
```
location       object
size          float64
total_sqft     object
bath          float64
price         float64
dtype: object
```

Transforming total_sqft column to Numeric value

In [14]:
```
def convert_to_numeric(value):
    if ' - ' in value:
        num = value.split('-')
        return (float(num[0]) + float(num[1]))/2
    try:
        return float(value)
    except:
        return 0
```

In [15]:
```
df['total_sqft'] = df['total_sqft'].apply(convert_to_numeric)
```

In [16]:
```
df.dtypes
```

Out[16]:
```
location       object
size          float64
total_sqft    float64
bath          float64
price         float64
dtype: object
```

Adding new column 'Price_Per_Sqft'

In [17]:
```
df['Price_Per_Sqft'] = df['price'] / df['total_sqft']
```

In [18]:
```
df
```

Out[18]:

	location	size	total_sqft	bath	price	Price_Per_Sqft
0	Electronic City Phase II	2.0	1056.0	2.0	39.07	0.036998
1	Chikka Tirupathi	4.0	2600.0	5.0	120.00	0.046154
2	Uttarahalli	3.0	1440.0	2.0	62.00	0.043056
3	Lingadheeranahalli	3.0	1521.0	3.0	95.00	0.062459
4	Kothanur	2.0	1200.0	2.0	51.00	0.042500
...	...	...	...	...	...	...
13315	Whitefield	5.0	3453.0	4.0	231.00	0.066898
13316	Richards Town	4.0	3600.0	5.0	400.00	0.111111
13317	Raja Rajeshwari Nagar	2.0	1141.0	2.0	60.00	0.052585
13318	Padmanabhanagar	4.0	4689.0	4.0	488.00	0.104073
13319	Doddathoguru	1.0	550.0	1.0	17.00	0.030909

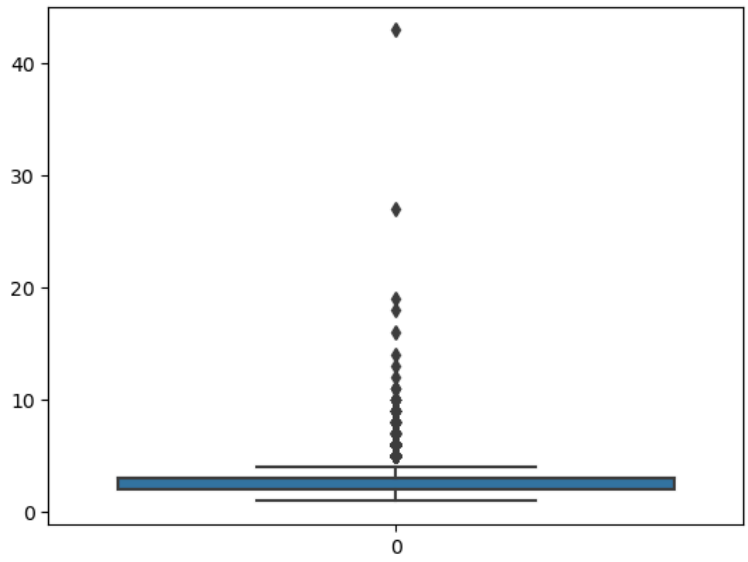
13319 rows × 6 columns

Removing Outliers

In [19]:
```
import seaborn as sns
```

In [20]:
```
sns.boxplot(df['size'])
```

Out[20]: <Axes: >



In [21]:
```
import numpy as np
```

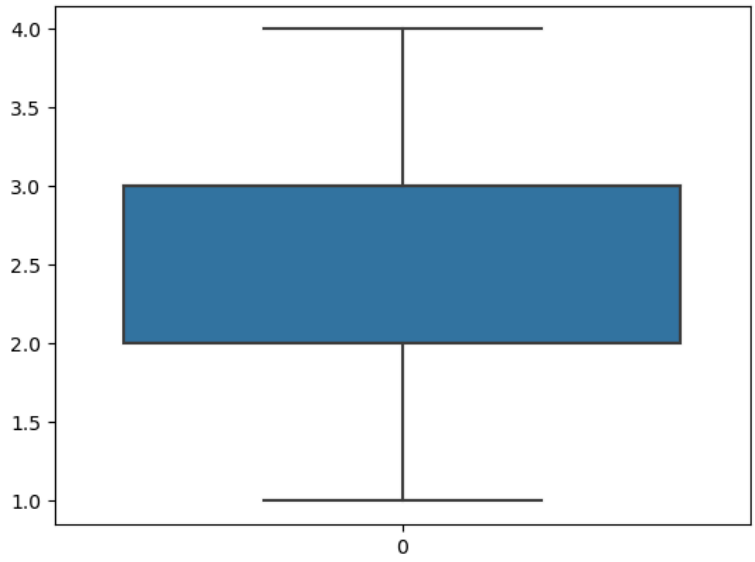
In [22]:
```
def outliers(data):
    data_item=sorted(data)
    q1 = np.percentile(data_item,25)
    q3 = np.percentile(data_item,75)
    iqr = q3-q1
    lower = q1-(1.5*iqr)
    upper = q3+(1.5*iqr)
    return lower,upper
```

In [23]:
```
lower_size,upper_size = outliers(df['size'])
```

In [24]:
```
df = df[df['size']>lower_size]
df = df[df['size']<upper_size]
```

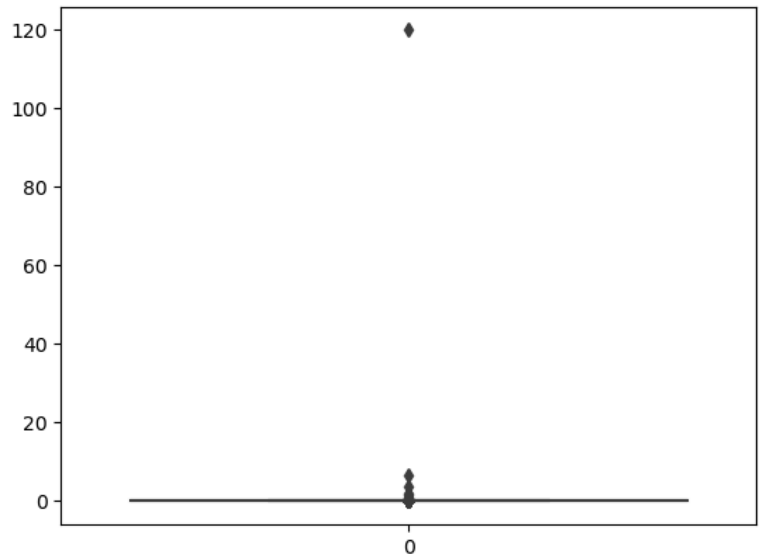
In [25]:
```
sns.boxplot(df['size'])
```

Out[25]: <Axes: >



In [26]:
```
sns.boxplot(df['Price_Per_Sqft'])
```

Out[26]: <Axes: >

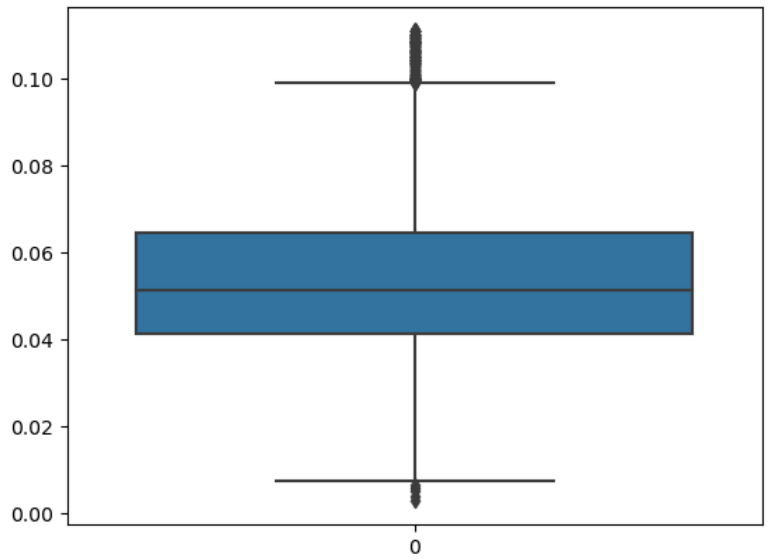


In [27]:
```
lower_pps,upper_pps = outliers(df['Price_Per_Sqft'])
```

In [28]:
```
df = df[df['Price_Per_Sqft']>lower_pps]
df = df[df['Price_Per_Sqft']<upper_pps]
```

In [29]:
```
sns.boxplot(df['Price_Per_Sqft'])
```

Out[29]: <Axes: >



In [30]:
```
df
```

Out[30]:

	location	size	total_sqft	bath	price	Price_Per_Sqft
0	Electronic City Phase II	2.0	1056.0	2.0	39.07	0.036998
1	Chikka Tirupathi	4.0	2600.0	5.0	120.00	0.046154
2	Uttarahalli	3.0	1440.0	2.0	62.00	0.043056
3	Lingadheeranahalli	3.0	1521.0	3.0	95.00	0.062459
4	Kothanur	2.0	1200.0	2.0	51.00	0.042500
...	...	...	...	...	...	...
13314	Green Glen Layout	3.0	1715.0	3.0	112.00	0.065306
13316	Richards Town	4.0	3600.0	5.0	400.00	0.111111
13317	Raja Rajeshwari Nagar	2.0	1141.0	2.0	60.00	0.052585
13318	Padmanabhanagar	4.0	4689.0	4.0	488.00	0.104073
13319	Doddathoguru	1.0	550.0	1.0	17.00	0.030909

11425 rows × 6 columns

Linear Regression

In [31]:
```
X = df[['total_sqft','size','bath']]
y = df['price']
```

In [32]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [33]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train,y_train)
```

Out[33]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [34]:
```
y_pred = model.predict(X_test)
```

In [35]:
```
from sklearn.metrics import mean_squared_error,r2_score
mse = mean_squared_error(y_test,y_pred)
```

In [36]:
```
r2 = r2_score(y_test,y_pred)
```

In [37]:
```
mse
```

Out[37]:
```
2644.363240704238
```

In [38]:
```
r2
```

Out[38]:
```
0.5409793884309391
```