In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
```

In [2]:

```
df = pd.read_csv("Banglore Housing Prices.csv")
```

In [4]:

```
df.head()
```

Out[4]:

	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

In [5]:

```
df.shape
```

Out[5]:

```
(13320, 5)
```

In [6]:

```
df.describe
```

Out[6]:

```
<bound method NDFrame.describe of                        location       size total_sqft  bath    pr
ice
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 x 5 columns]>
```

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13320 entries, 0 to 13319
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   location    13319 non-null  object
 1   size        13304 non-null  object
 2   total_sqft  13320 non-null  object
 3   bath        13247 non-null  float64
 4   price       13320 non-null  float64
dtypes: float64(2), object(3)
memory usage: 520.4+ KB
```

In [8]:

```
df.isnull().sum()
```

Out[8]:

```
location       1
size          16
total_sqft     0
bath          73
price          0
dtype: int64
```

In [9]:

```
df.dropna(inplace = True)
```

In [12]:

```
df.isnull().sum()
```

Out[12]:

```
location      0
size          0
total_sqft    0
bath          0
price         0
dtype: int64
```

In [13]:

```
df['size'] = [int(value.split(' ')[0]) for value in df['size']]
```

In [14]:

```
df.head()
```

Out[14]:

	location	size	total_sqft	bath	price
0	Electronic City Phase II	2	1056	2.0	39.07
1	Chikka Tirupathi	4	2600	5.0	120.00
2	Uttarahalli	3	1440	2.0	62.00
3	Lingadheeranahalli	3	1521	3.0	95.00
4	Kothanur	2	1200	2.0	51.00

In [22]:

```
df['total_sqft'].describe
```

Out[22]:

```
<bound method NDFrame.describe of 0        1056
1        2600
2        1440
3        1521
4        1200
         ... 
13315    3453
13316    3600
13317    1141
13318    4689
13319     550
Name: total_sqft, Length: 13246, dtype: object>
```

In [23]:

```
def convert_sqft(value):
    try:
        if '-' in value:
            start, end = map(float, value.split('-'))
            return (start + end) / 2
        else:
            return float(value)
    except ValueError:
        return float('nan')
```

In [24]:

```
df['total_sqft'] = [convert_sqft(value) for value in df['total_sqft']]
```

In [25]:

```
df['total_sqft']
```

Out[25]:

```
0        1056.0
1        2600.0
2        1440.0
3        1521.0
4        1200.0
          ...
13315    3453.0
13316    3600.0
13317    1141.0
13318    4689.0
13319     550.0
Name: total_sqft, Length: 13246, dtype: float64
```

In [26]:

```
df['total_sqft'].isnull().sum()
```

Out[26]:

```
46
```

In [27]:

```
df.isnull().sum()
```

Out[27]:

```
location       0
size           0
total_sqft    46
bath           0
price          0
dtype: int64
```

In [28]:

```
df.dropna(inplace = True)
```

In [29]:

```
df.isnull().sum()
```

Out[29]:

```
location      0
size          0
total_sqft    0
bath          0
price         0
dtype: int64
```

In [30]:

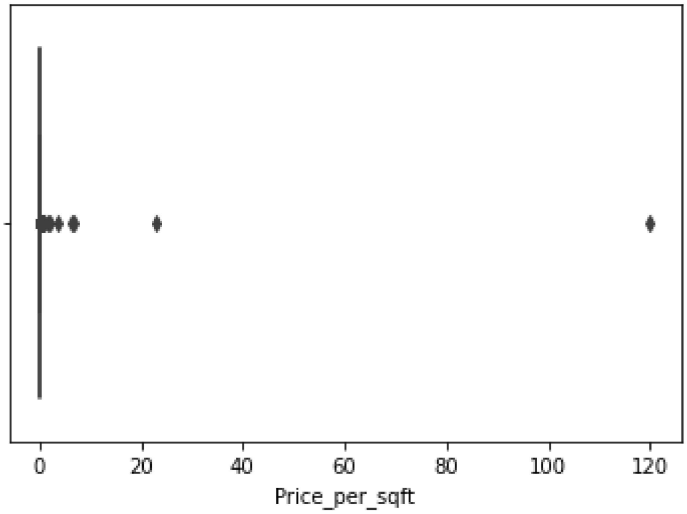
```
df['Price_per_sqft'] = df['price']/df['total_sqft']
```

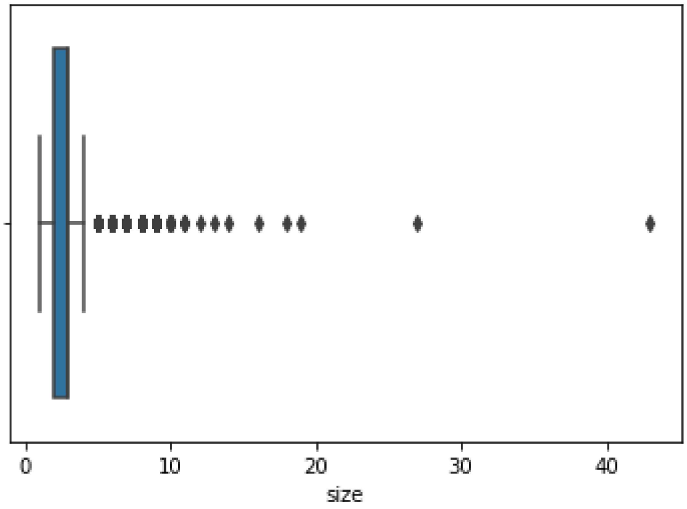
In [31]:

```
selected_columns = ['Price_per_sqft', 'size']
outliers = df[selected_columns]
```

In [33]:

```
for i in outliers:
    sns.boxplot(x=df[i])
    plt.show()
```





In [34]:

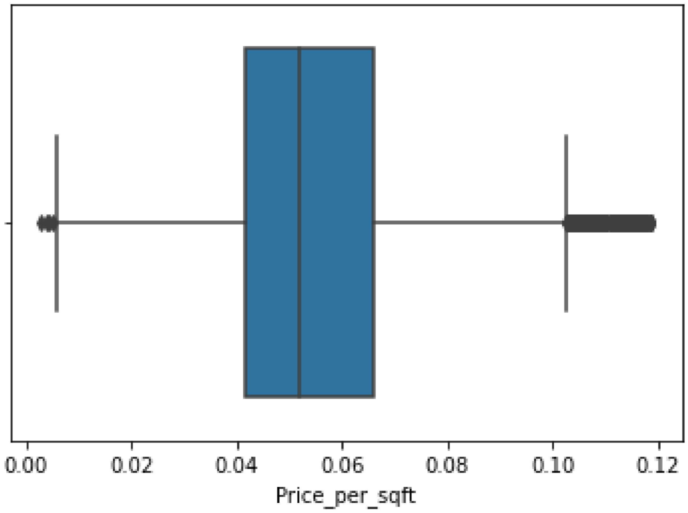
```
def remove_outliers(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return column[(column >= lower_bound) & (column <= upper_bound)]
```

In [35]:

```
df['Price_per_sqft'] = remove_outliers(df['Price_per_sqft'])
```

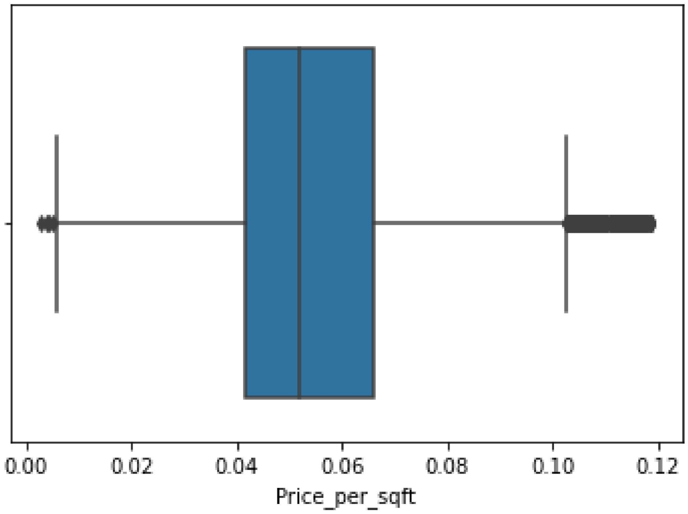
In [36]:

```
sns.boxplot(x=df['Price_per_sqft'])
plt.show()
```



In [37]:

```
sns.boxplot(x=df['Price_per_sqft'])
plt.show()
```



In [39]:

```
df.isnull().sum()
```

Out[39]:

```
location            0
size                0
total_sqft          0
bath                0
price               0
Price_per_sqft   1265
dtype: int64
```

In [40]:

```
df.dropna(inplace = True)
```

In [41]:

```
df.isnull().sum()
```

Out[41]:

```
location          0
size              0
total_sqft        0
bath              0
price             0
Price_per_sqft    0
dtype: int64
```

In [42]:

```
df
```

Out[42]:

	location	size	total_sqft	bath	price	Price_per_sqft
0	Electronic City Phase II	2	1056.0	2.0	39.07	0.036998
1	Chikka Tirupathi	4	2600.0	5.0	120.00	0.046154
2	Uttarahalli	3	1440.0	2.0	62.00	0.043056
3	Lingadheeranahalli	3	1521.0	3.0	95.00	0.062459
4	Kothanur	2	1200.0	2.0	51.00	0.042500
...	...	...	...	...	...	...
13315	Whitefield	5	3453.0	4.0	231.00	0.066898
13316	Richards Town	4	3600.0	5.0	400.00	0.111111
13317	Raja Rajeshwari Nagar	2	1141.0	2.0	60.00	0.052585
13318	Padmanabhanagar	4	4689.0	4.0	488.00	0.104073
13319	Doddathoguru	1	550.0	1.0	17.00	0.030909

11935 rows × 6 columns

In [43]:

```
X = df[['size','total_sqft', 'bath','Price_per_sqft' ]]
y = df['price']
```

In [44]:

```
X
```

Out[44]:

	size	total_sqft	bath	Price_per_sqft
0	2	1056.0	2.0	0.036998
1	4	2600.0	5.0	0.046154
2	3	1440.0	2.0	0.043056
3	3	1521.0	3.0	0.062459
4	2	1200.0	2.0	0.042500
...	...	...	...	...
13315	5	3453.0	4.0	0.066898
13316	4	3600.0	5.0	0.111111
13317	2	1141.0	2.0	0.052585
13318	4	4689.0	4.0	0.104073
13319	1	550.0	1.0	0.030909

11935 rows × 4 columns

In [45]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=42 )
```

In [46]:

```
model = LinearRegression()
```

In [47]:

```
model.fit(X_train, y_train)
```

Out[47]:

```
▾ LinearRegression
LinearRegression()
```

In [48]:

```
y_pred = model.predict(X_test)
```

In [49]:

```
mse = mean_squared_error(y_test, y_pred)
```

In [50]:

```
r_squared = r2_score(y_test, y_pred)
```

In [51]:

```
cv = np.mean(cross_val_score(model, X, y, cv=5))
```

In [52]:

```
print(f'Mean Squared Error (MSE): {mse}')
print(f'R-squared: {r_squared}')
print(f'Cross Validation Score: {cv}')
```

```
Mean Squared Error (MSE): 1131.4300760113033
R-squared: 0.7793564127656505
Cross Validation Score: 0.6983224679434117
```

In [ ]:

```

```