DSBDA Assg 4 Banglore Housing prices

```
import pandas as pd
import numpy as np

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

```
                      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 x 5 columns]
```

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

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

```
df['bath']=df['bath'].replace(np.NaN,df['bath'].mean())
df.isnull().sum()
```

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

```
df['location'].fillna('Not known' , inplace = True)
df.isnull().sum()
df['size'].fillna('1 BHK' , inplace = True)
df.isnull().sum()
```

```
C:\Users\anuja\AppData\Local\Temp\ipykernel_14144\4046444482.py:1:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
```

```
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df['location'].fillna('Not known' , inplace = True)
C:\Users\anuja\AppData\Local\Temp\ipykernel_14144\4046444482.py:3:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df['size'].fillna('1 BHK' , inplace = True)

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

df

                      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 x 5 columns]

df['size'] = df['size'].str.extract(r'(\d+)').astype(float)
```

```
---------------------------------------------------------------------
-----
AttributeError                            Traceback (most recent call
last)
Cell In[17], line 1
----> 1 df['size'] = df['size'].str.extract(r'(\d+)').astype(float)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\
generic.py:6299, in NDFrame.__getattr__(self, name)
   6292 if (
   6293     name not in self._internal_names_set
   6294     and name not in self._metadata
   6295     and name not in self._accessors
   6296     and
self._info_axis._can_hold_identifiers_and_holds_name(name)
   6297 ):
   6298     return self[name]
-> 6299 return object.__getattribute__(self, name)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\
accessor.py:224, in CachedAccessor.__get__(self, obj, cls)
    221 if obj is None:
    222     # we're accessing the attribute of the class, i.e.,
Dataset.geo
    223     return self._accessor
--> 224 accessor_obj = self._accessor(obj)
    225 # Replace the property with the accessor object. Inspired by:
    226 # https://www.pydanny.com/cached-property.html
    227 # We need to use object.__setattr__ because we overwrite
__setattr__ on
    228 # NDFrame
    229 object.__setattr__(obj, self._name, accessor_obj)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\strings\
accessor.py:191, in StringMethods.__init__(self, data)
    188 def __init__(self, data) -> None:
    189     from pandas.core.arrays.string_ import StringDtype
--> 191     self._inferred_dtype = self._validate(data)
    192     self._is_categorical = isinstance(data.dtype,
CategoricalDtype)
    193     self._is_string = isinstance(data.dtype, StringDtype)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\strings\
accessor.py:245, in StringMethods._validate(data)
    242 inferred_dtype = lib.infer_dtype(values, skipna=True)
    244 if inferred_dtype not in allowed_types:
--> 245     raise AttributeError("Can only use .str accessor with
string values!")
    246 return inferred_dtype
```

```
AttributeError: Can only use .str accessor with string values!
```

```
df[['size']].head()
```

```
   size
0   2.0
1   4.0
2   3.0
3   3.0
4   2.0
```

```
df['size']
```

```
0        2.0
1        4.0
2        3.0
3        3.0
4        2.0
        ...
13315    5.0
13316    4.0
13317    2.0
13318    4.0
13319    1.0
Name: size, Length: 13320, dtype: float64
```

c) Transform the 'total_sqft' column to contain numerical values on same scale. If the range is given average value of the range to be taken.

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

```
0        1056
1        2600
2        1440
3        1521
4        1200
        ...
13315    3453
13316    3600
13317    1141
13318    4689
13319     550
Name: total_sqft, Length: 13320, dtype: object
```

```
def convertSqu(sqft):
    try:
        if '-' in sqft:
            total = list(map(float, sqft.split('-')))
            return (total[0]+total[1])/2
```

```
        return float(sqft)
    except:
        return None

df['total_sqft'] = df['total_sqft'].apply(convertSqu)

df.dropna(subset=['total_sqft'], inplace = True)

df['total_sqft']
```

```
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: 13274, dtype: float64
```

d) Calculate and add one more column as 'Price_Per_Sqft'

```
def pricePerSqu():
    df['Price_per_square'] = df['price']*100000 / df['total_sqft']
    print(df['Price_per_square'])
pricePerSqu()
```

```
0         3699.810606
1         4615.384615
2         4305.555556
3         6245.890861
4         4250.000000
             ...
13315     6689.834926
13316    11111.111111
13317     5258.545136
13318    10407.336319
13319     3090.909091
Name: Price_per_square, Length: 13274, dtype: float64
```

```
df.head()
```

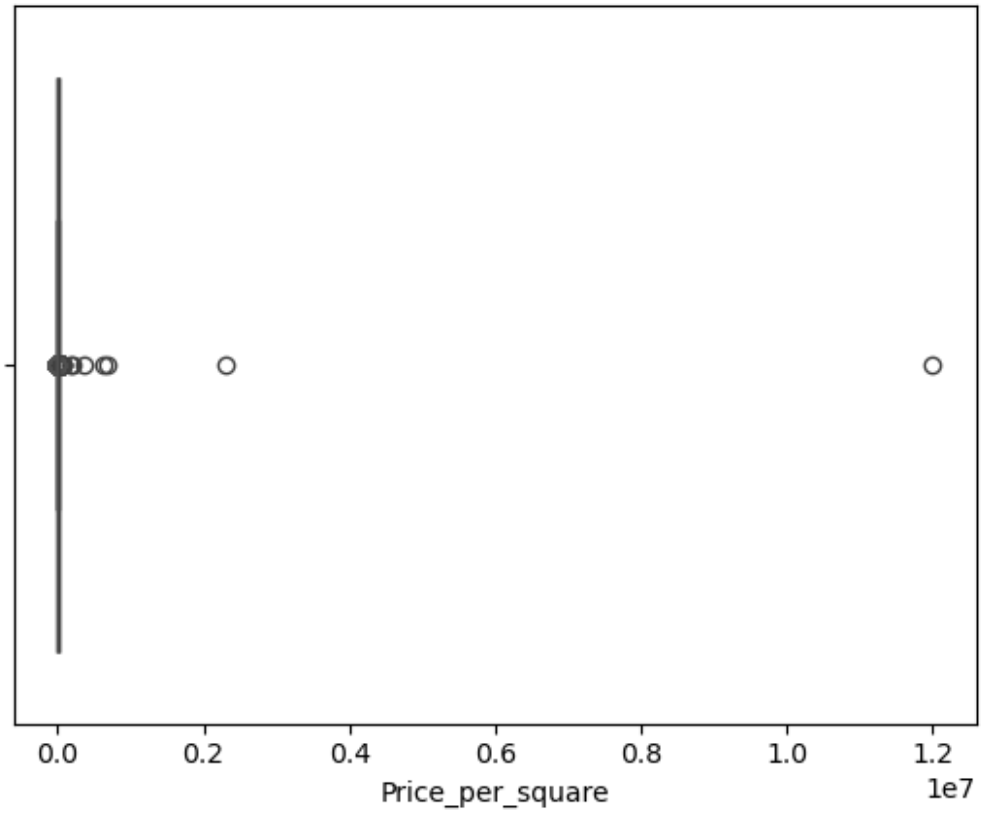
```
                  location  size  total_sqft  bath   price  BHK  \
0  Electronic City Phase II   2.0      1056.0   2.0   39.07    2
1          Chikka Tirupathi   4.0      2600.0   5.0  120.00    4
2               Uttarahalli   3.0      1440.0   2.0   62.00    3
3        Lingadheeranahalli   3.0      1521.0   3.0   95.00    3
4                  Kothanur   2.0      1200.0   2.0   51.00    2
```

```
   Price_per_square
0       3699.810606
1       4615.384615
2       4305.555556
3       6245.890861
4       4250.000000
```

e) Remove the outliers from Price_Per_Sqft and BHK Size column if any.

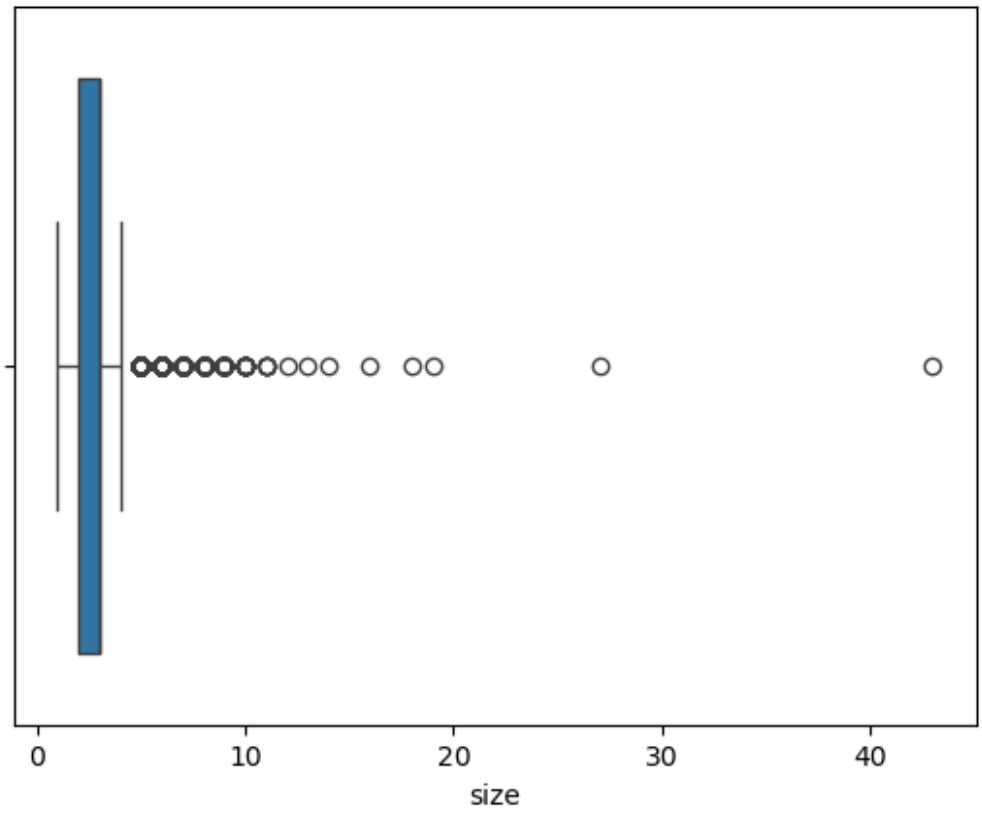
```
import seaborn as sn
sn.boxplot( x=df['Price_per_square'])
```

```
<Axes: xlabel='Price_per_square'>
```




```
sn.boxplot( x=df['size'])
```

```
<Axes: xlabel='size'>
```



Removing outliers using IQR

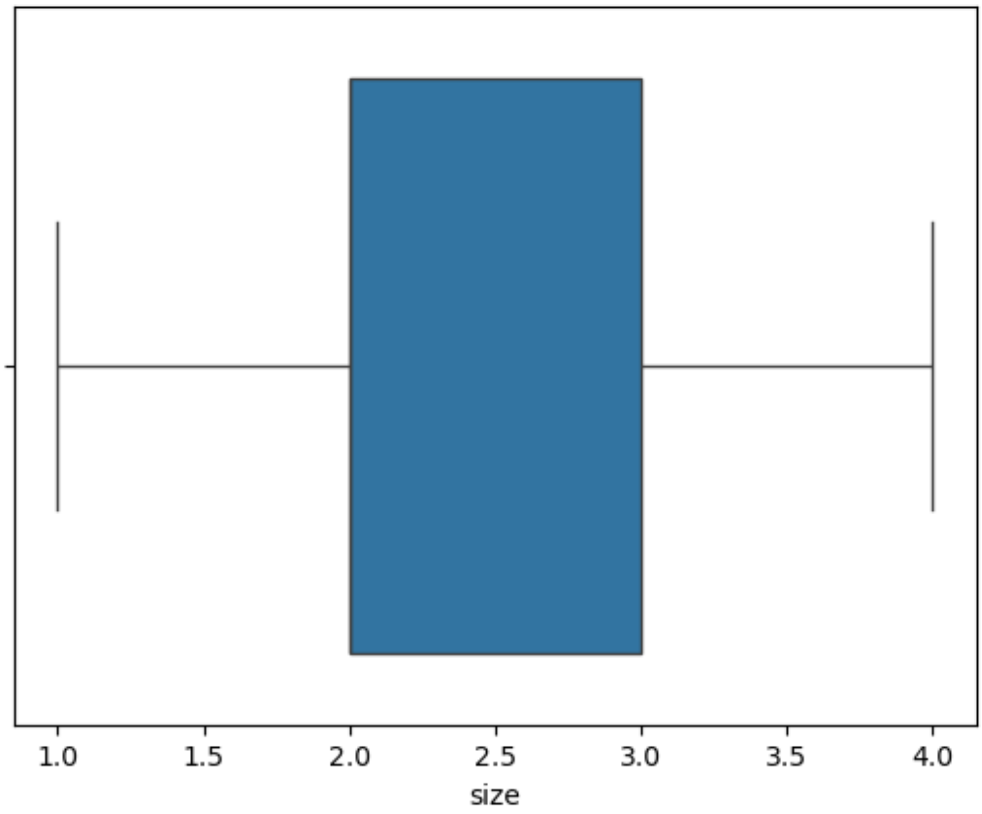
```
def outliersRemove(df,col):
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    IQR = q3-q1
    low_bound = q1-1.5 * IQR
    up_bound = q3 +1.5 * IQR
    return df[(df[col] >= low_bound) & (df[col] <= up_bound)]
df=outliersRemove(df,'Price_per_square')
df=outliersRemove(df,'size')

df.shape
```

```
(11559, 7)
```

```
sn.boxplot( x=df['size'])
```

```
<Axes: xlabel='size'>
```



f) Apply the Linear Regression model to the data and display the training and testing performance measures as Mean Squared Error and Accuracy.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

X=df[['total_sqft', 'size','bath','Price_per_square']]
y=df['price']*10000

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

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

mse, r2
```

```
(77036993926.93571, 0.8210632793372097)
```