Assignment - 3

Name: GAGAN SAI G B Reg No: 20MID0192 Campus: VIT Vellore

Building a Regression Model

1. Download the dataset: Dataset
2. Load the dataset into the tool.

In [1]:
```
#Importing the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

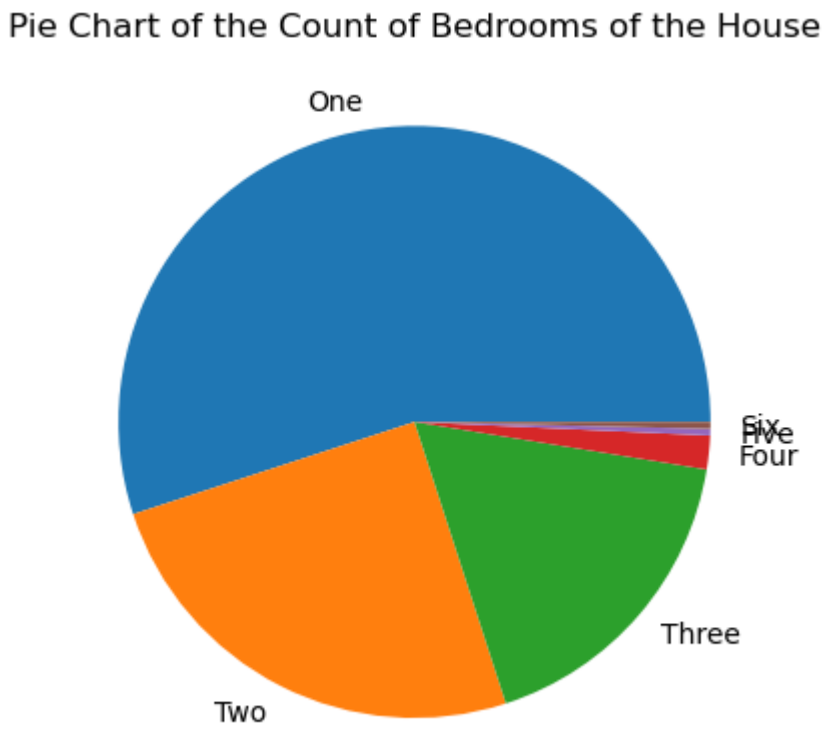
In [3]:
```
#Loading the dataset
data = pd.read_csv('C:/Users/gagan/Downloads/Housing.csv')
data.head()
```

Out[3]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating
0	13300000	7420	4	2	3	yes	no	no	no
1	12250000	8960	4	4	4	yes	no	no	no
2	12250000	9960	3	2	2	yes	no	yes	no
3	12215000	7500	4	2	2	yes	no	yes	no
4	11410000	7420	4	1	2	yes	yes	yes	no

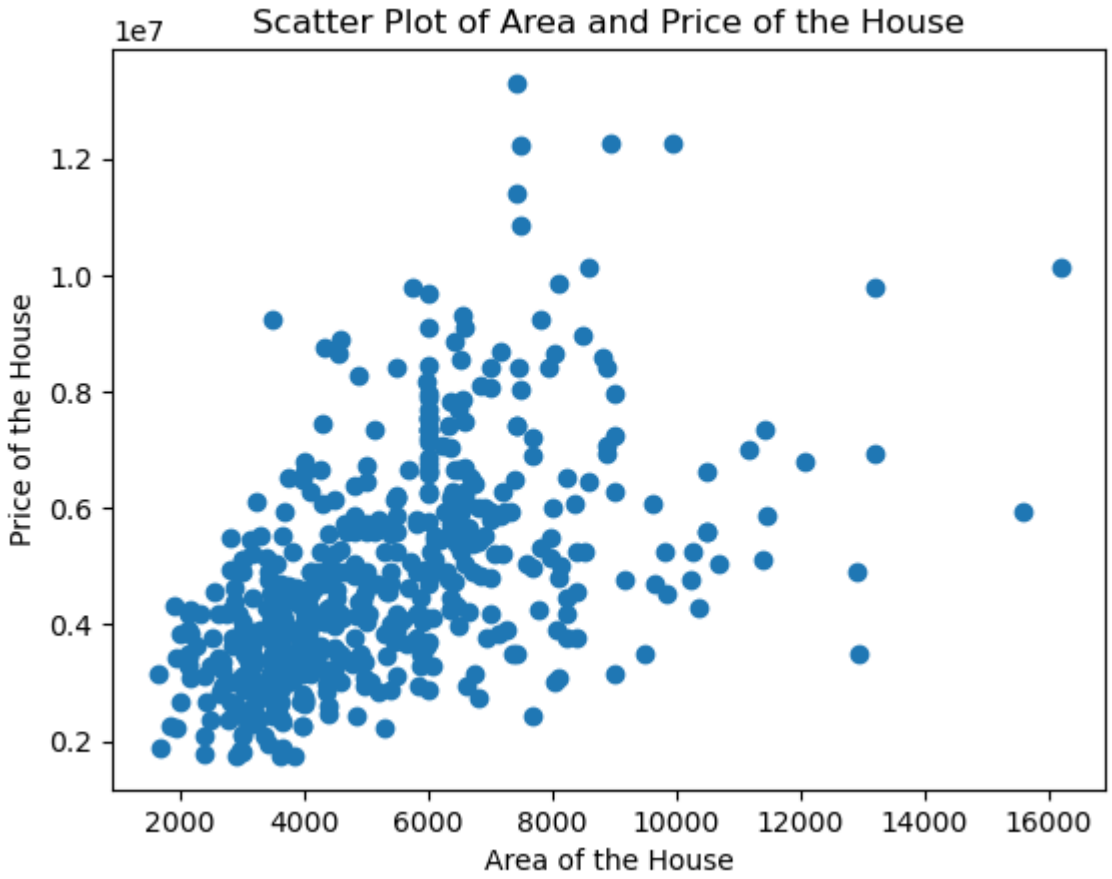
1. Perform Below Visualizations. □ Univariate Analysis

In [4]:
```
#Pie Chart
pie_chart=data['bedrooms'].value_counts()
label=['One','Two','Three', 'Four', 'Five', 'Six']
pie_chart
plt.pie(pie_chart,labels=label, radius=1)
plt.title('Pie Chart of the Count of Bedrooms of the House')
plt.show()
```



Bi-Variate Analysis

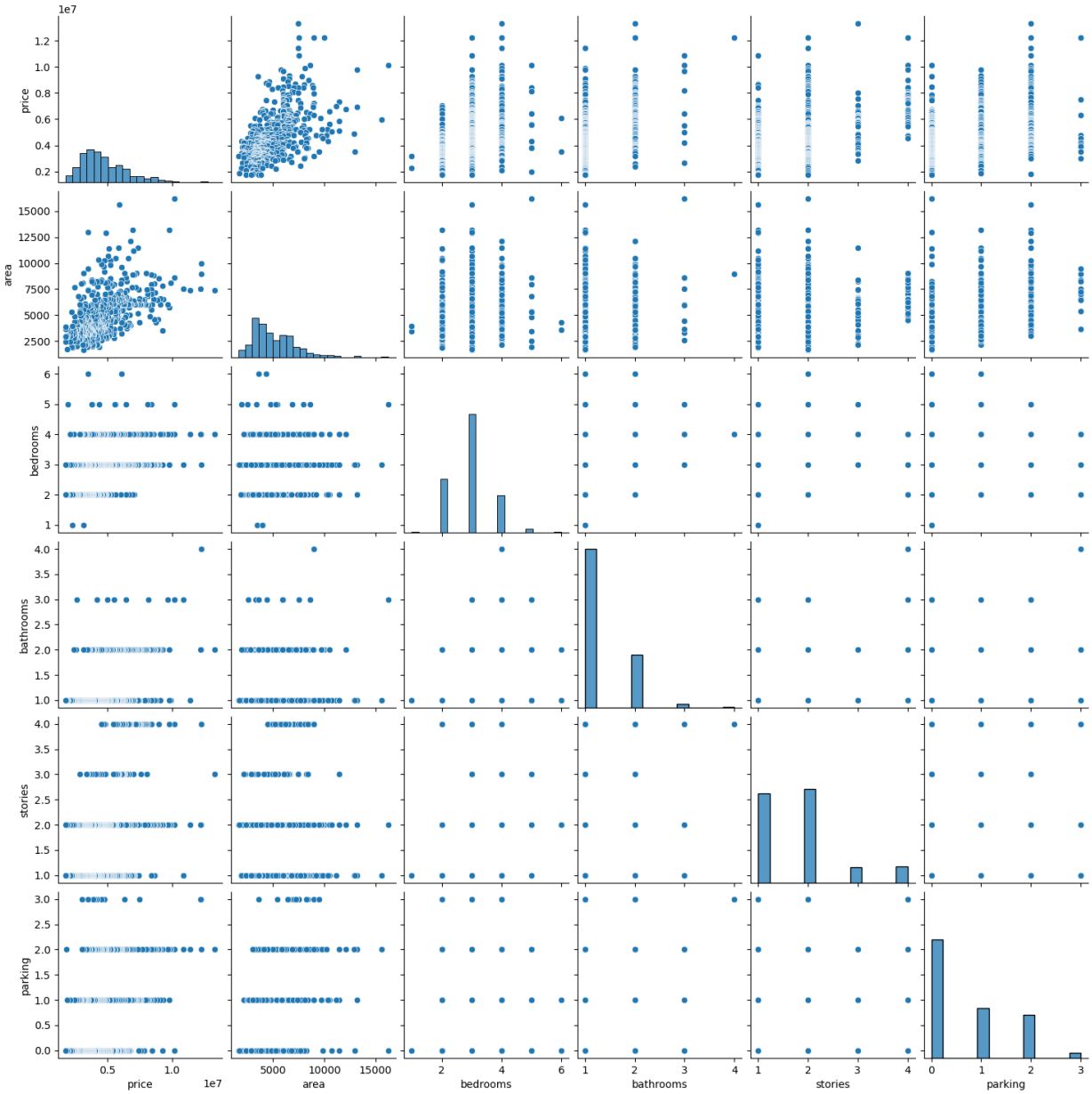
In [5]:
```
#Scatter Plot
plt.scatter(data['area'], data['price'])
plt.xlabel('Area of the House')
plt.ylabel('Price of the House')
plt.title('Scatter Plot of Area and Price of the House')
plt.show()
```



Multi-Variate Analysis

In [6]:
```
#Pair Plot
import seaborn as sns
sns.pairplot(data[['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']])
```

Out[6]:
```
<seaborn.axisgrid.PairGrid at 0x2b1d20981f0>
```



1. Perform descriptive statistics on the dataset.

In [7]:
```
data.describe()
```

Out[7]:

	price	area	bedrooms	bathrooms	stories	parking
count	5.450000e+02	545.000000	545.000000	545.000000	545.000000	545.000000
mean	4.766729e+06	5150.541284	2.965138	1.286239	1.805505	0.693578
std	1.870440e+06	2170.141023	0.738064	0.502470	0.867492	0.861586
min	1.750000e+06	1650.000000	1.000000	1.000000	1.000000	0.000000
25%	3.430000e+06	3600.000000	2.000000	1.000000	1.000000	0.000000
50%	4.340000e+06	4600.000000	3.000000	1.000000	2.000000	0.000000
75%	5.740000e+06	6360.000000	3.000000	2.000000	2.000000	1.000000
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.000000

1. Check for Missing values and deal with them.

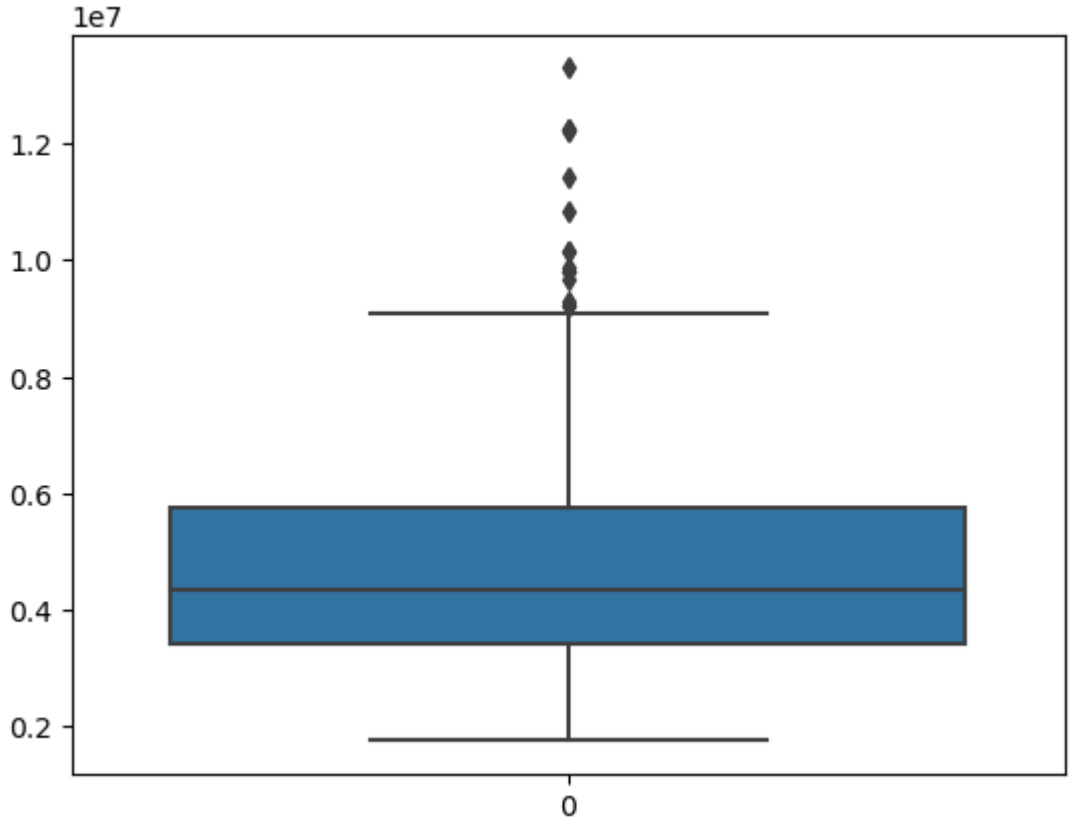
In [8]:
```
data.isnull().sum()
```

Out[8]:
```
price             0
area              0
bedrooms          0
bathrooms         0
stories           0
mainroad          0
guestroom         0
basement          0
hotwaterheating   0
airconditioning   0
parking           0
furnishingstatus  0
dtype: int64
```

1. Find the outliers and replace them outliers

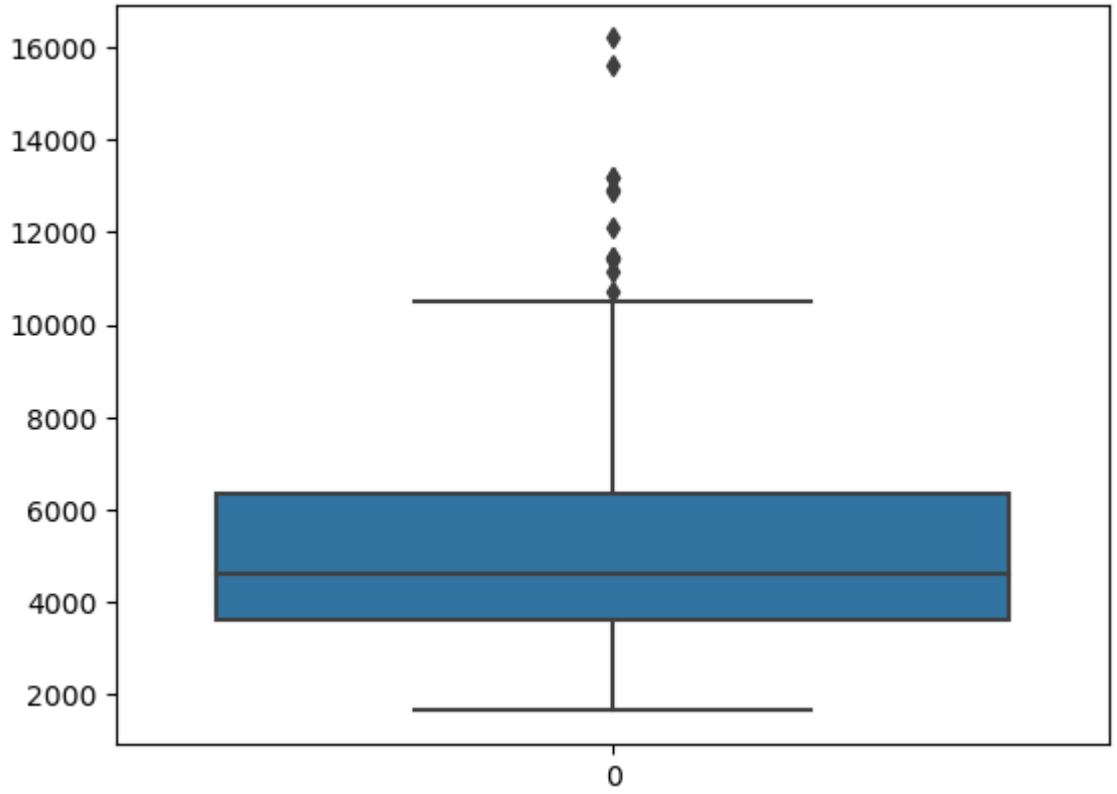
In [9]:
```
#Checking outliers of the Numerical attributes using a Box Plot
#Price Attribute
sns.boxplot(data['price'])
```

Out[9]:
```
<Axes: >
```



In [10]:
```
#Area Attribute
sns.boxplot(data['area'])
```

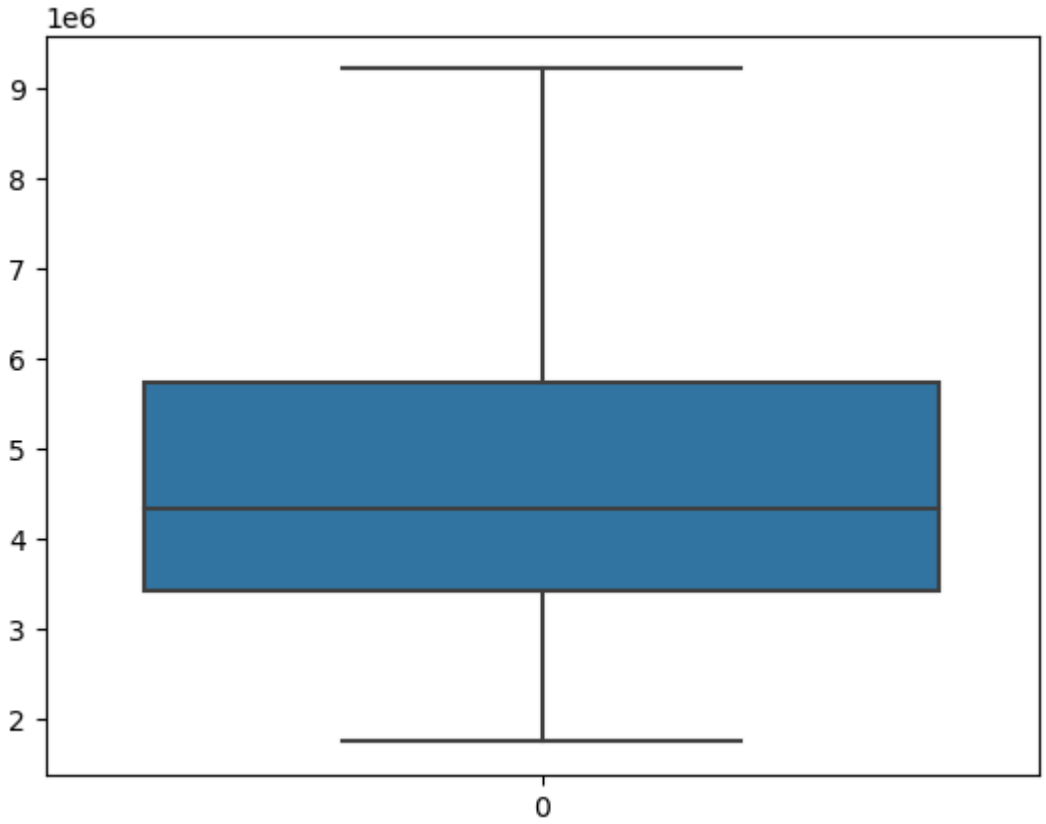
Out[10]: <Axes: >



In [11]:
```
#Outliers identified in both the attributes
#Handling Outliers
#Price Attribute
q1 = data['price'].quantile(0.25)
q2 = data['price'].quantile(0.75)
Inter_Quartile_Range = q2 - q1
whisker_width = 1.5
lower_whisker = q1 -(whisker_width*Inter_Quartile_Range)
upper_whisker = q2 + (whisker_width*Inter_Quartile_Range)
data['price']=np.where(data['price']>upper_whisker,upper_whisker,np.where(data['price'
```

In [12]:
```
#After removing Outliers in Price Attribute
sns.boxplot(data['price'])
```

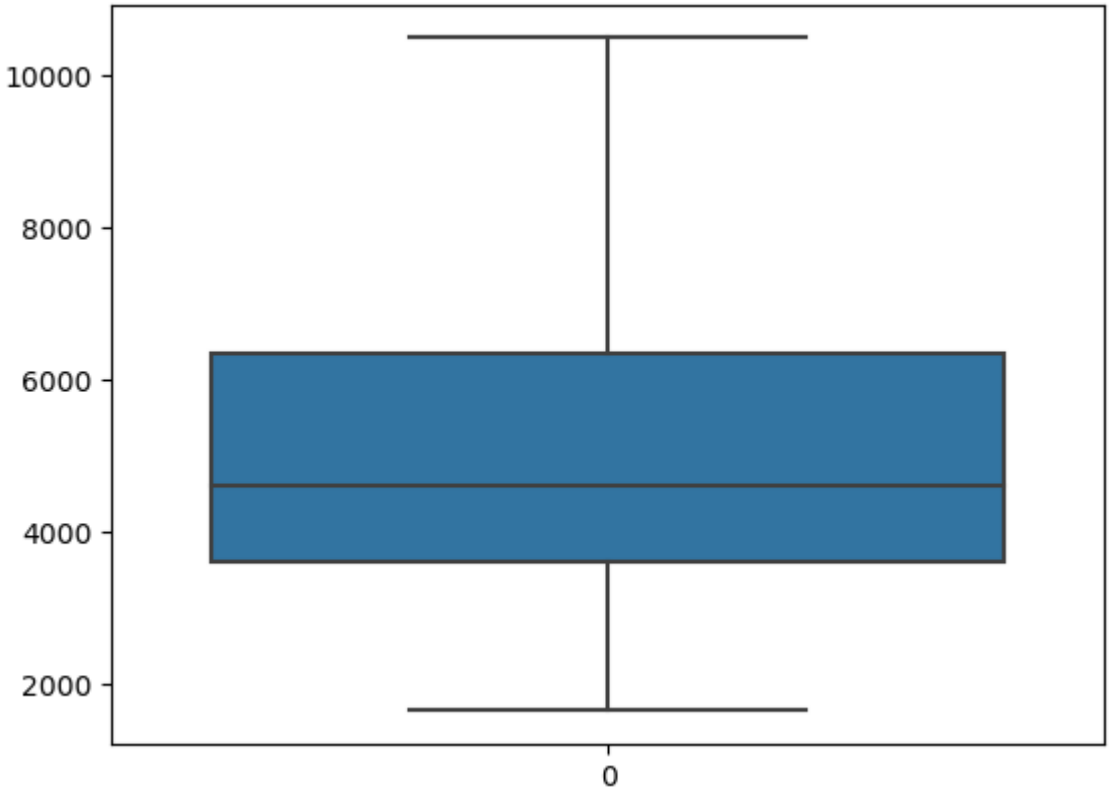
Out[12]:
```
<Axes: >
```



In [13]:
```
#Handling Outliers
#Area Attribute
q1 = data['area'].quantile(0.25)
q2 = data['area'].quantile(0.75)
Inter_Quartile_Range = q2 - q1
whisker_width = 1.5
lower_whisker = q1 -(whisker_width*Inter_Quartile_Range)
upper_whisker = q2 + (whisker_width*Inter_Quartile_Range)
data['area']=np.where(data['area']>upper_whisker,upper_whisker,np.where(data['area']<
```

In [14]:
```
#After removing outliers in Area Attribute
sns.boxplot(data['area'])
```

Out[14]:
```
<Axes: >
```



1. Check for Categorical columns and perform encoding.

In [15]:
```
#Identify Categorical columns
categ_cols=data.select_dtypes(include=['object']).columns
print('Categorical Columns: ', categ_cols)
```

```
Categorical Columns:  Index(['mainroad', 'guestroom', 'basement', 'hotwaterheating',
       'airconditioning', 'furnishingstatus'],
      dtype='object')
```

In [16]:
```
#Label Encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()

for col in categ_cols:
    data[col]=le.fit_transform(data[col])
```

In [17]:
```
data.head()
```

Out[17]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheati
0	9205000.0	7420.0	4	2	3	1	0	0	
1	9205000.0	8960.0	4	4	4	1	0	0	
2	9205000.0	9960.0	3	2	2	1	0	1	
3	9205000.0	7500.0	4	2	2	1	0	1	
4	9205000.0	7420.0	4	1	2	1	1	1	

1. Split the data into dependent and independent variables.

In [18]:
```
#Price Column is identified as the dependent variable
dep_var=data['price']
indep_var=data.drop('price', axis=1)
```

In [19]:
```
#Dependent Variables
print('Dependent Variables: \n',dep_var.head(0))
```

```
Dependent Variables:
 Series([], Name: price, dtype: float64)
```

In [20]:
```
#Independent Variables
print('Independent Variables: \n',indep_var.head(0))
```

```
Independent Variables:
 Empty DataFrame
Columns: [area, bedrooms, bathrooms, stories, mainroad, guestroom, basement, hotwater
heating, airconditioning, parking, furnishingstatus]
Index: []
```

1. Scale the independent variables

In [21]:
```
from sklearn.preprocessing import StandardScaler
scale=StandardScaler()
indep_scaled=scale.fit_transform(indep_var)
```

In [22]:
```
indep_scaled_data=pd.DataFrame(indep_scaled, columns=indep_var.columns)
indep_scaled_data.head()
```

Out[22]:

	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airc
0	1.156583	1.403419	1.421812	1.378217	0.405623	-0.465315	-0.734539	-0.219265	
1	1.925060	1.403419	5.405809	2.532024	0.405623	-0.465315	-0.734539	-0.219265	
2	2.424072	0.047278	1.421812	0.224410	0.405623	-0.465315	1.361397	-0.219265	
3	1.196504	1.403419	1.421812	0.224410	0.405623	-0.465315	1.361397	-0.219265	
4	1.156583	1.403419	-0.570187	0.224410	0.405623	2.149083	1.361397	-0.219265	

1. Split the data into training and testing

In [23]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(indep_scaled_data, dep_var, test_s
```

In [24]:
```
print("Shape of x_train:", x_train.shape)
print("Shape of x_test:", x_test.shape)
print("Shape of y_train:", y_train.shape)
print("Shape of y_test:", y_test.shape)
```

```
Shape of x_train: (436, 11)
Shape of x_test: (109, 11)
Shape of y_train: (436,)
Shape of y_test: (109,)
```

1. Build the Model Linear Regression

In [25]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr
```

Out[25]:
```
LinearRegression
LinearRegression()
```

1. Train the Model

In [26]:
```
#Training the model
lr.fit(x_train, y_train)
```

Out[26]:
```
LinearRegression
LinearRegression()
```

1. Test the Model

In [27]:
```
#Testing the model
y_pred = lr.predict(x_test)
y_pred
```

Out[27]:
```
array([5215154.81060095, 6675719.70641621, 3210585.63155314,
       4719348.1216706 , 3494112.11179584, 3844003.45060721,
       5837299.87684943, 5957697.38811311, 2745391.27972497,
       2658153.41177725, 8841079.55503709, 2979167.86492965,
       3173184.08483981, 3428370.56772898, 3955248.50258283,
       5067043.6022799 , 3052057.38643865, 4966513.49574211,
       4645490.66042896, 3746569.7059927 , 5321051.93476343,
       5554281.0446497 , 2901231.40512594, 4356163.12821942,
       5457165.80164233, 7092398.39430059, 3483565.40870114,
       5310536.96736101, 7242016.7003042 , 3527949.52170957,
       5854233.74056216, 3435010.62044492, 6810257.13103388,
       4417053.62117231, 3814426.53386575, 5730514.68900455,
       4966077.2134742 , 4586121.05305665, 3210525.89028568,
       4620063.37759807, 4743265.48292833, 3639970.08052502,
       6653842.80541868, 4135210.52951514, 3951665.65284458,
       4432760.21179709, 6735890.93270247, 4156075.12063828,
       4104734.42125273, 3539955.80803489, 7289085.39085094,
       2899890.03018756, 4609161.99776899, 4555150.7745718 ,
       3967978.79440647, 2715580.27689577, 6912497.78453977,
       3061983.52706302, 4474905.02130336, 2942524.05575878,
       4736025.7959986 , 3442534.87111875, 5139232.05407745,
       4467604.35305223, 4223698.83277996, 4681316.87977892,
       6719422.40860448, 3689991.07798774, 6132794.02417399,
       5801337.96598186, 4101990.46722743, 4746653.72956527,
       4781442.59312456, 7610965.07349499, 3557410.7106333 ,
       5513166.13856126, 3953041.72151006, 4166452.54935274,
       4972042.51102303, 3708225.38797955, 7092677.60691721,
       4076099.28962119, 5947127.36201953, 5092926.64157217,
       2870560.6433346 , 6835671.63098782, 2772035.30172809,
       3727683.70986151, 7382546.54513452, 8020873.64127878,
       3307529.37595526, 6267836.94838029, 3798596.88443642,
       3793147.20503719, 7603883.03624612, 4940519.91755316,
       5374745.63254499, 6478767.80740688, 4485181.15544413,
       5524938.36171186, 3846391.82914709, 6380348.60206266,
       3819620.0497948 , 5863292.50536525, 5168048.21544367,
       4493334.77518699, 7014998.42554504, 6384628.51231362,
       5839936.06364878])
```

1. Measure the performance using Metrics

In [28]:
```
from sklearn.metrics import mean_squared_error, r2_score
#Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)
print('MSE of the Model = ', mse)
```

```
MSE of the Model =  1340722973330.946
```

In [29]:
```
#R2 Score
r2 = r2_score(y_test, y_pred)
print('R2 Score of the Model = ', r2)
```

```
R2 Score of the Model =  0.6681289959808647
```