In [45]:

```
# Assignment - 3
# Name: Selva Manooj M Reg No: 20MID0189 Campus: VIT Vellore
```

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
data = pd.read_csv('C:/Users/imsel/Downloads/archive/Housing.csv')
data.head()
```

Out[2]:

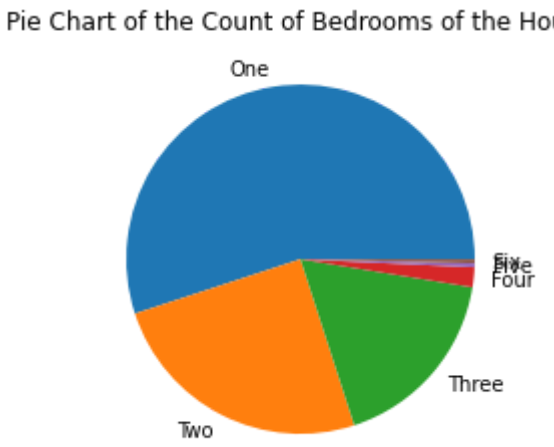
	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwater
0	13300000	7420	4	2	3	yes	no	no	
1	12250000	8960	4	4	4	yes	no	no	
2	12250000	9960	3	2	2	yes	no	yes	
3	12215000	7500	4	2	2	yes	no	yes	
4	11410000	7420	4	1	2	yes	yes	yes	

In [3]:

```
#1. Perform Below Visualizations.Univariate Analysis
```

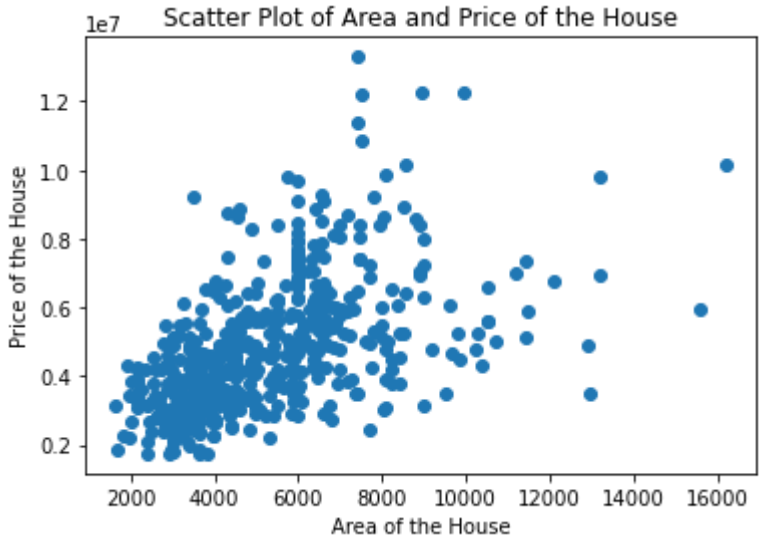
In [5]:

```
#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()
```



In [6]:

```
#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()
```

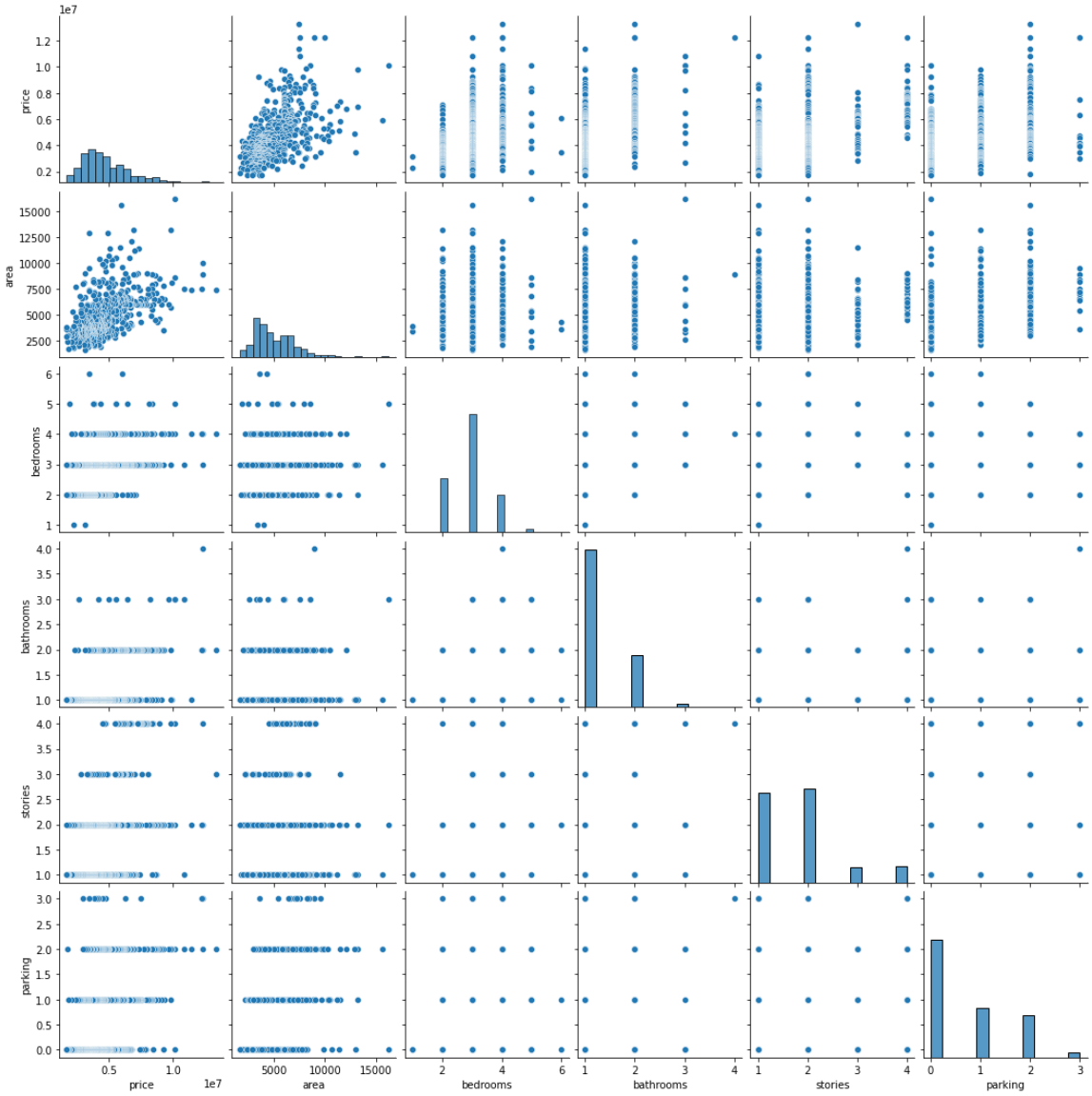


In [7]:

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

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x1afe734f730>
```



In [8]:

```
# Perform descriptive statistics on the dataset.
```

In [9]:

```
data.describe()
```

Out[9]:

	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

In [10]:

```
data.isnull().sum()
```

Out[10]:

```
price             0
area              0
bedrooms          0
bathrooms         0
stories           0
mainroad          0
guestroom         0
basement          0
hotwaterheating   0
airconditioning   0
parking           0
prefarea          0
furnishingstatus  0
dtype: int64
```

In [11]:

```
#Find the outliers and replace them outliers
```

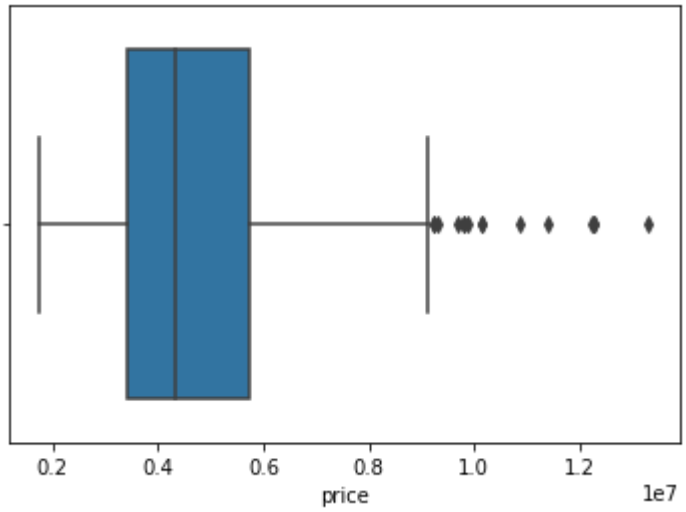
In [12]:

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

```
C:\Users\imsel\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

Out[12]:

```
<AxesSubplot:xlabel='price'>
```



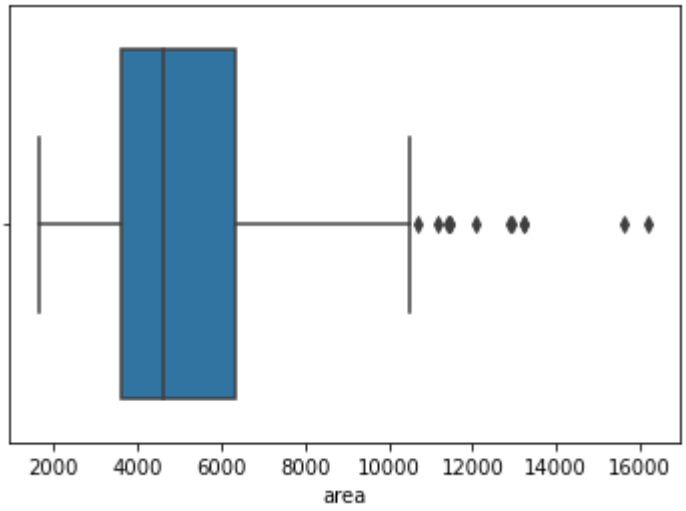
In [13]:

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

```
C:\Users\imsel\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

Out[13]:

```
<AxesSubplot:xlabel='area'>
```



In [14]:

```
#Check for Categorical columns and perform encoding.
```

In [29]:

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

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

In [30]:

```
#Label Encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for col in categ_cols:
 data[col]=le.fit_transform(data[col])
```

In [17]:

```
#. Split the data into dependent and independent variables.
```

In [31]:

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

In [32]:

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

```
Dependent Variables:
 Series([], Name: price, dtype: int64)
```

In [33]:

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

```
Independent Variables:
 Empty DataFrame
Columns: [area, bedrooms, bathrooms, stories, mainroad, guestroom, basemen
t, hotwaterheating, airconditioning, parking, prefarea, furnishingstatus]
Index: []
```

In [34]:

```
from sklearn.preprocessing import StandardScaler
scale=StandardScaler()
indep_scaled=scale.fit_transform(indep_var)
```

In [35]:

```
indep_scaled_data=pd.DataFrame(indep_scaled, columns=indep_var.columns)
indep_scaled_data.head()
```

Out[35]:

	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheati
0	1.046726	1.403419	1.421812	1.378217	0.405623	-0.465315	-0.734539	-0.2192
1	1.757010	1.403419	5.405809	2.532024	0.405623	-0.465315	-0.734539	-0.2192
2	2.218232	0.047278	1.421812	0.224410	0.405623	-0.465315	1.361397	-0.2192
3	1.083624	1.403419	1.421812	0.224410	0.405623	-0.465315	1.361397	-0.2192
4	1.046726	1.403419	-0.570187	0.224410	0.405623	2.149083	1.361397	-0.2192

In [36]:

```
#Split the data into training and testing
```

In [38]:

```
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_size
```

In [39]:

```
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, 12)
Shape of x_test: (109, 12)
Shape of y_train: (436,)
Shape of y_test: (109,)
```

In [40]:

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr
```

Out[40]:

```
LinearRegression()
```

In [41]:

```
#Training the model
lr.fit(x_train, y_train)
```

Out[41]:

```
LinearRegression()
```

In [42]:

```
#Testing the model
y_pred = lr.predict(x_test)
y_pred
```

Out[42]:

```
array([3586317.24179681, 3799849.7055172 , 4359072.24016124,
       3269465.51506229, 2655380.44547121, 2852713.06146378,
       6321375.03723823, 3758047.67079753, 2124563.7664246 ,
       2672977.15588909, 2991704.17876141, 4271583.7028644 ,
       5110385.15488699, 2903360.71145828, 5403674.67182258,
       4495831.67876516, 4517381.69793773, 2893684.45430232,
       5571253.33521514, 6665891.64992053, 3990859.78005518,
       4543980.86607098, 3427443.91813207, 4786525.8181594 ,
       4675756.80222087, 9250064.657601  , 4476108.23812346,
       4348222.26856081, 3757770.58529906, 4452731.10595832,
       7653332.86048556, 5771619.81244554, 4140060.36569699,
       4523001.49418628, 2601595.15632729, 8172804.79921605,
       4882953.67190902, 5048163.56793925, 3535814.54870916,
       3155635.84909793, 3674704.60715913, 2689457.02257614,
       3990494.87854   , 6240302.34723513, 8244716.95574814,
       3190777.9638743 , 7153940.88592607, 3202934.27497815,
       3705034.00207261, 2736162.60644494, 2946373.33672446,
       8057079.35374191, 8030146.51886616, 4771371.21890119,
       4773882.91655414, 4179143.30950497, 2902427.20641024,
       4279559.82515303, 5872034.37412819, 3926602.12757583,
       4617983.64676565, 3100905.69722595, 3104088.00775066,
       3981217.36957227, 4225257.73324877, 4698008.20874824,
       5391507.75171237, 4307865.33970946, 3587956.41703456,
       4648248.78170148, 4929850.88105021, 4944779.15559866,
       3147126.18846106, 3293685.17530581, 7024563.09499263,
       6710407.79512557, 3060838.76430617, 5782614.4099443 ,
       3956576.58953185, 6653863.83038243, 5063765.64096998,
       6593797.64590187, 7555757.06319343, 6654294.95568683,
       3373067.70459894, 3571984.54234265, 3569653.05394013,
       6517767.14412933, 6630416.43936654, 4752075.32117065,
       4072163.51692717, 3536915.10552433, 7786902.75634788,
       5779896.55991045, 4556760.02298584, 5780345.10672466,
       4966359.3945906 , 3217881.19241319, 2701688.69061539,
       5940683.51169699, 4637846.09453601, 2186127.92220728,
       5120659.01029526, 6234181.9232956 , 5225140.69740647,
       2684312.78266897, 3706193.6330444 , 4457360.81524439,
       5704388.26456893])
```

In [43]:

```
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 =  1506750239572.6777
```

In [44]:

```
#R2 Score
r2 = r2_score(y_test, y_pred)
print('R2 Score of the Model = ', r2)
```

```
R2 Score of the Model =  0.572197434032998
```