In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
df=pd.read_csv('MagicBricks.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	Area	BHK	Bathroom	Furnishing	Locality	Parking	Price	Status	Transa
0	800.0	3	2.0	Semi-Furnished	Rohini Sector 25	1.0	6500000	Ready_to_move	New_Pr
1	750.0	2	2.0	Semi-Furnished	J R Designers Floors, Rohini Sector 24	1.0	5000000	Ready_to_move	New_Pr
2	950.0	2	2.0	Furnished	Citizen Apartment, Rohini Sector 13	1.0	15500000	Ready_to_move	F
3	600.0	2	2.0	Semi-Furnished	Rohini Sector 24	1.0	4200000	Ready_to_move	F
4	650.0	2	2.0	Semi-Furnished	Rohini Sector 24 carpet area 650 sqft status R...	1.0	6200000	Ready_to_move	New_Pr

In [4]:

```
# Let's look at the shape of the dataset
df.shape
```

Out[4]:

```
(1259, 11)
```

In [5]:

```
# let's check if any of the rows are duplicated, they must be removed immediately
df.duplicated().sum()
```

Out[5]:

```
83
```

In [6]:

```
# Removing duplicate rows
df.drop_duplicates(inplace=True)
```

In [7]:

```
df.shape
```

Out[7]:

```
(1176, 11)
```

In [8]:

```
#let's see if any column has any null values
df.isna().sum()
```

Out[8]:

```
Area             0
BHK              0
Bathroom         1
Furnishing       5
Locality         0
Parking         31
Price            0
Status           0
Transaction      0
Type             5
Per_Sqft       227
dtype: int64
```

The dataset has five columns with missing values - Parking, Bathroom, Furnishing, Type and Per_Sqft. Finding value for Per_Sqft is quite easy. We have to divide Price by Area to get Per_Sqft. To find the missing values in Parking, Bathroom, Furnishing and Type, I will replace the missing values with the mode of them.

In [9]:

```
df['Per_Sqft'].fillna((df['Price']/df['Area']),inplace=True)
```

In [10]:

```
df['Bathroom'].fillna(df['Bathroom'].mode()[0],inplace=True)
df['Furnishing'].fillna(df['Furnishing'].mode()[0],inplace=True)
df['Parking'].fillna(df['Parking'].mode()[0],inplace=True)
df['Type'].fillna(df['Type'].mode()[0],inplace=True)
```

In [11]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1176 entries, 0 to 1258
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Area         1176 non-null   float64
 1   BHK          1176 non-null   int64  
 2   Bathroom     1176 non-null   float64
 3   Furnishing   1176 non-null   object 
 4   Locality     1176 non-null   object 
 5   Parking      1176 non-null   float64
 6   Price        1176 non-null   int64  
 7   Status       1176 non-null   object 
 8   Transaction  1176 non-null   object 
 9   Type         1176 non-null   object 
 10  Per_Sqft     1176 non-null   float64
dtypes: float64(4), int64(2), object(5)
memory usage: 110.2+ KB
```

In [12]:

```
df.dtypes
```

Out[12]:

```
Area           float64
BHK              int64
Bathroom       float64
Furnishing      object
Locality        object
Parking        float64
Price            int64
Status          object
Transaction     object
Type            object
Per_Sqft       float64
dtype: object
```

Number of Parking and Bathroom cannot be float, so we can convert float to int to save memory.

In [13]:

```
df[['Parking','Bathroom']]=df[['Parking','Bathroom']].astype('int64')
```

In [14]:

```
df.dtypes
```

Out[14]:

```
Area           float64
BHK              int64
Bathroom         int64
Furnishing      object
Locality        object
Parking          int64
Price            int64
Status          object
Transaction     object
Type            object
Per_Sqft       float64
dtype: object
```

In [15]:

```
df.nunique()
```

Out[15]:

```
Area           315
BHK              8
Bathroom         7
Furnishing       3
Locality       365
Parking          9
Price          284
Status           2
Transaction      2
Type             2
Per_Sqft       433
dtype: int64
```

In [16]:

```
df['Locality'].unique()
```

Out[16]:

```
array(['Rohini Sector 25', 'J R Designers Floors, Rohini Sector 24',
       'Citizen Apartment, Rohini Sector 13', 'Rohini Sector 24',
       'Rohini Sector 24 carpet area 650 sqft status Ready to Move floor
4 out of 4 floors transaction New Property furnishing Semi-Furnished faci
ng East overlooking Garden/Park, Main Road car parking 1 Open bathroom 2
balcony 1 ownership Freehold Newly Constructed Property Newly Constructed
Property East Facing Property 2BHK Newly build property for Sale. A House
is waiting for a Friendly Family to make it a lovely home.So please come
and make his house feel alive once again. read more Contact Agent View Ph
one No. Share Feedback Garima properties Certified Agent Trusted by Users
Genuine Listings Market Knowledge',
       'Delhi Homes, Rohini Sector 24', 'Rohini Sector 21',
       'Rohini Sector 22', 'Rohini Sector 20',
       'Rohini Sector 8 How Auctions work? The borrower has the physical
possession of the Property. However the lender (Bank) can legally sell th
e Property. super area 32 sqm status Ready to Move floor 1 transaction Re
sale furnishing Semi-Furnished Contact Now Enquire Now Auction By',
       'Rohini Sector 25 carpet area 660 sqft status Ready to Move floor
```

As we can see that there are so many localities, dealing with these is challenging. I have decided to take only top 12 localities and list the remaining localities as 'other' in the dataset. It will help in analysing the locality of the house in a better way.

In [17]:

```
def grp_loc(locality):
    locality=locality.lower()
    if 'rohini' in locality:
        return 'Rohini Sector'
    elif 'dwarka' in locality:
        return 'Dwarka Sector'
    elif 'shahdara' in locality:
        return 'Shahdara'
    elif 'vasant' in locality:
        return 'Vasant Kunj'
    elif 'paschim' in locality:
        return 'Paschim Vihar'
    elif 'vasundhara' in locality:
        return 'Vasundhara Enclave'
    elif 'punjabi' in locality:
        return 'Punjabi Bagh'
    elif 'kalkaji' in locality:
        return 'Kalkaji'
    elif 'lajpat' in locality:
        return 'Lajpat Nagar'
    elif 'laxmi' in locality:
        return 'Laxmi Nagar'
    elif 'friends' in locality:
        return 'New Friends Colony'
    elif 'kailash' in locality:
        return 'Kailash Colony'
    else:
        return 'Other'
df['Locality']=df['Locality'].apply(grp_loc)
```

In [18]:

```
df['Locality'].value_counts()
```

Out[18]:

```
Other                616
Lajpat Nagar          85
Shahdara              75
Dwarka Sector         72
Rohini Sector         71
Kailash Colony        37
Laxmi Nagar           33
Vasant Kunj           33
Kalkaji               32
New Friends Colony    31
Punjabi Bagh          31
Paschim Vihar         30
Vasundhara Enclave    30
Name: Locality, dtype: int64
```

In [19]:

```
# Let's look at the descriptive statistics
df.describe()
```

Out[19]:

	Area	BHK	Bathroom	Parking	Price	Per_Sqft
count	1176.000000	1176.000000	1176.000000	1176.000000	1.176000e+03	1176.000000
mean	1447.542711	2.789966	2.551871	1.953231	2.109173e+07	15108.514563
std	1487.658687	0.960993	1.052994	6.409197	2.523174e+07	19767.263465
min	28.000000	1.000000	1.000000	1.000000	1.000000e+06	1250.000000
25%	800.000000	2.000000	2.000000	1.000000	5.800000e+06	6584.000000
50%	1172.500000	3.000000	2.000000	1.000000	1.400000e+07	11111.000000
75%	1700.000000	3.000000	3.000000	2.000000	2.600000e+07	17231.083333
max	24300.000000	10.000000	7.000000	114.000000	2.400000e+08	183333.000000

EDA

In [20]:

```
color_palette=sns.color_palette(['#797D62','#9B9B7A','#D9AE94','#E5C59E','#F1DCA7','#F8D4
sns.set_palette(color_palette)
```

In [21]:

```
df.columns
```

Out[21]:

```
Index(['Area', 'BHK', 'Bathroom', 'Furnishing', 'Locality', 'Parking', 'Pr
ice',
       'Status', 'Transaction', 'Type', 'Per_Sqft'],
      dtype='object')
```

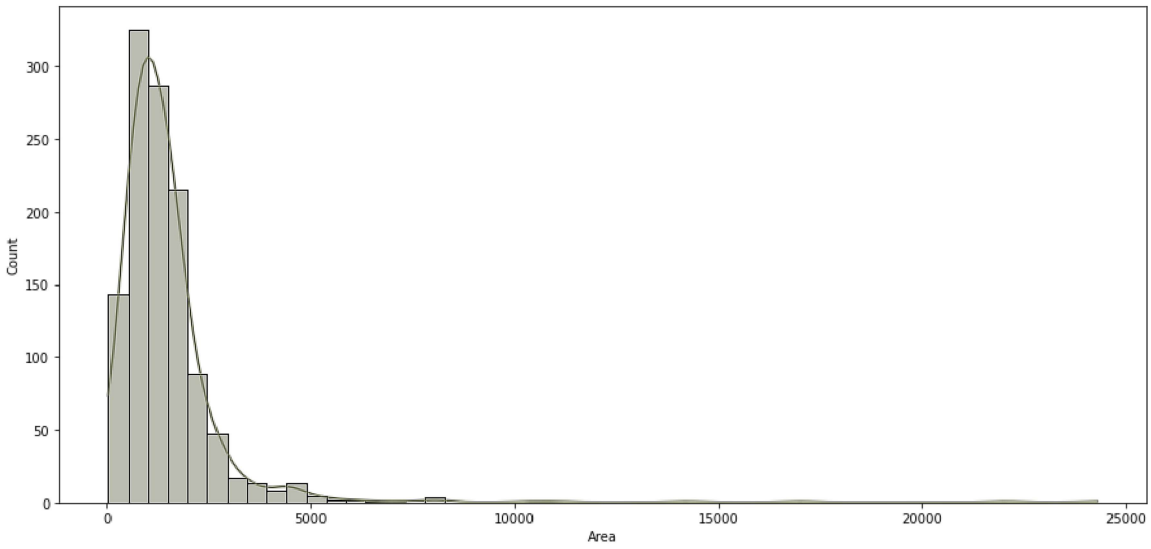
Area

In [22]:

```
plt.figure(figsize=(15,7))
sns.histplot(x=df['Area'],kde=True,bins=50)
```

Out[22]:

```
<AxesSubplot:xlabel='Area', ylabel='Count'>
```

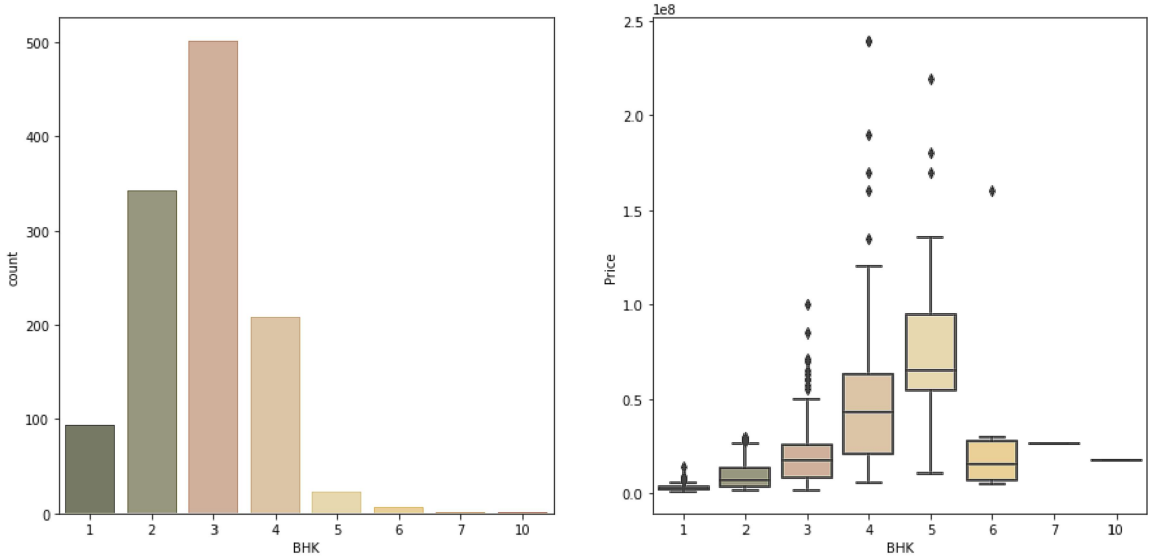


BHK and Price

In [23]:

```
plt.figure(figsize=(15,7))
plt.subplot(1,2,1)
sns.countplot(x=df['BHK'])
plt.subplot(1,2,2)
sns.boxplot(x=df['BHK'],y=df['Price'])
print('Correlation between BHK and Price is',df['BHK'].corr(df['Price']))
print('Skewness of the BHK is',df['BHK'].skew())
```

```
Correlation between BHK and Price is 0.566038856358519
Skewness of the BHK is 0.5395026710389439
```



1. Most of the properties are less than 5 bhk.
2. Very less properties are haing more than 5 bhk that means 2,3,4 and 5 bhk are more popular than 6,7 and 10 bhk.
3. Prices of 6,7 and 10 bhk are less than 4 or 5 bhk property, that feels strange.

Q. Why the prices are low for 6,7 and 10 bhk.

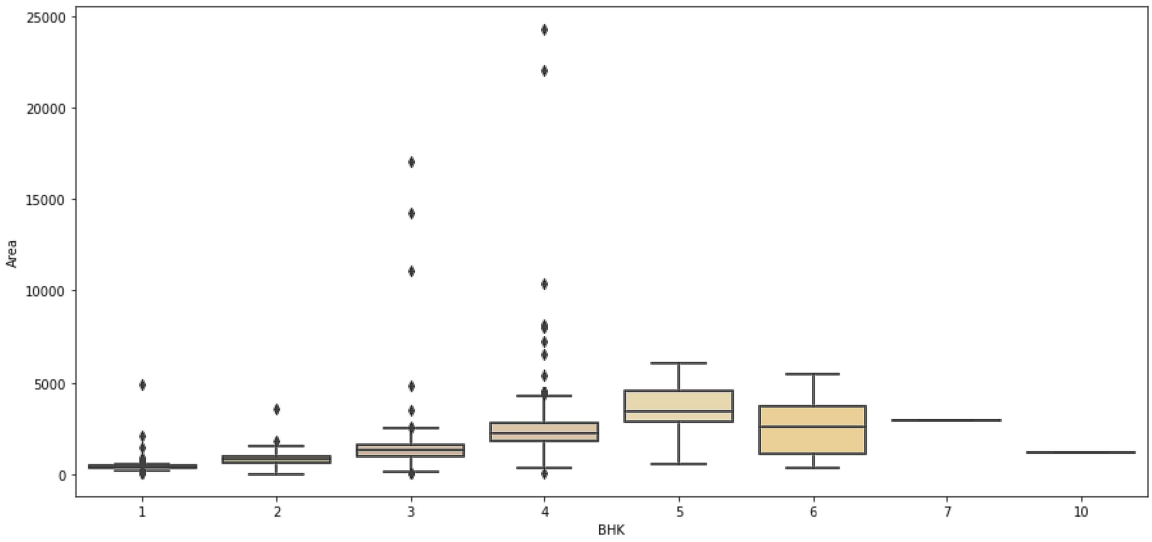
BHK and Area

In [24]:

```
plt.figure(figsize=(15,7))
sns.boxplot(x=df['BHK'],y=df['Area'])
```

Out[24]:

```
<AxesSubplot:xlabel='BHK', ylabel='Area'>
```



The reason for the lower prices of 6,7 and 10 bhk properties is that the area occpied by these properties are less than or equal to the properties with 3,4 and 5 bhk. So people prefer more space in rooms than more number of rooms.

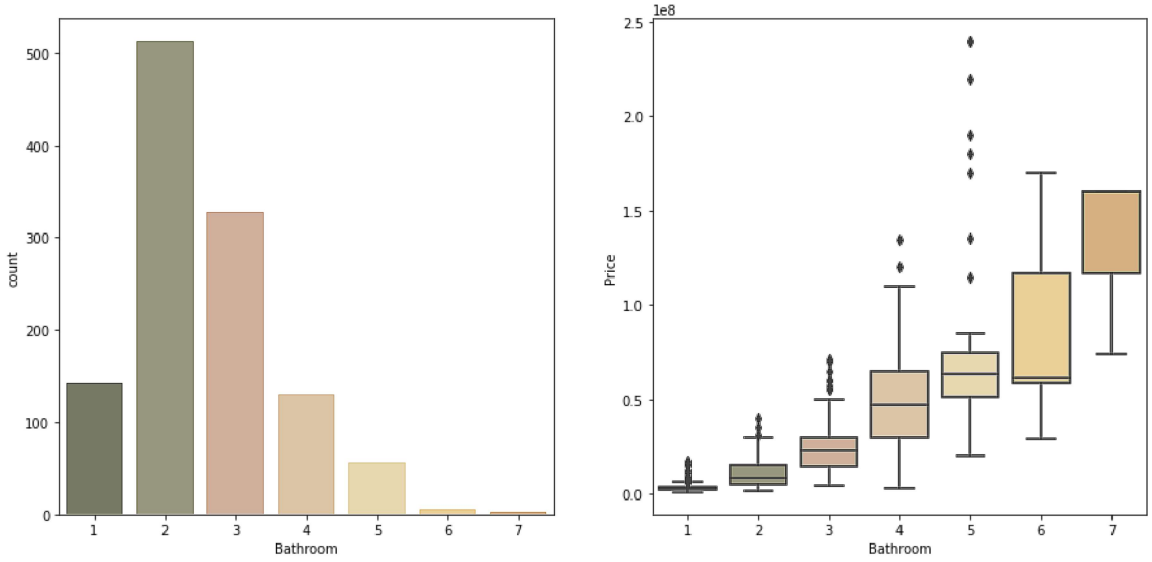
Interesting!.

Bathroom and Price

In [25]:

```
plt.figure(figsize=(15,7))
plt.subplot(1,2,1)
sns.countplot(x=df['Bathroom'])
plt.subplot(1,2,2)
sns.boxplot(x=df['Bathroom'],y=df['Price'])
print('Correlation between Bathroom and Price is',df['Bathroom'].corr(df['Price']))
print('Skewness of the Bathroom is',df['Bathroom'].skew())
```

```
Correlation between Bathroom and Price is 0.7316835672458932
Skewness of the Bathroom is 0.8268062674140753
```



More the number of bathrooms in property, more the price.

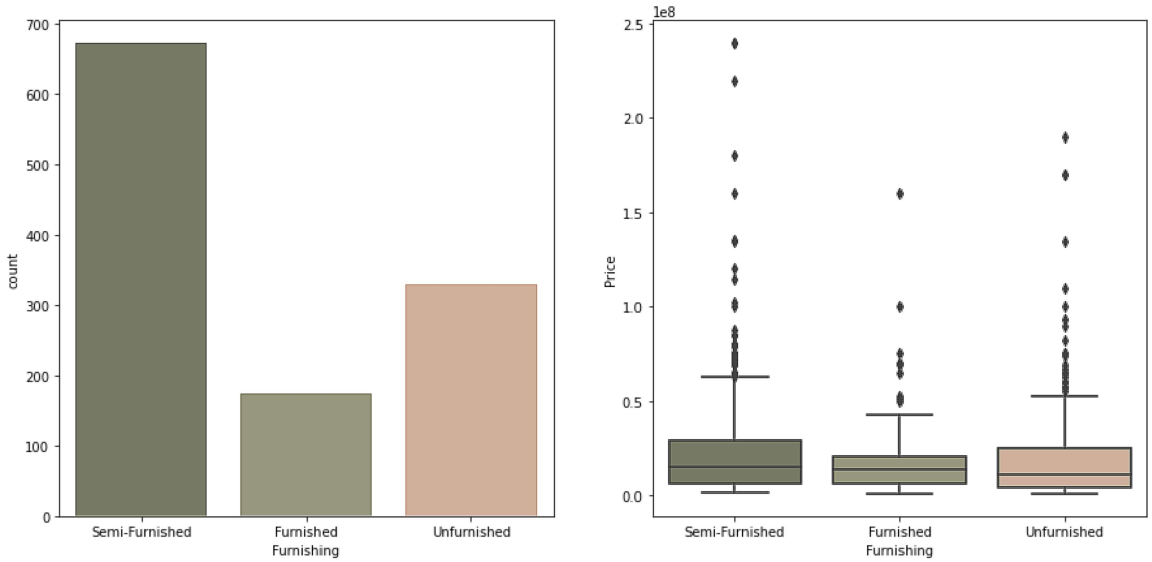
Furnishing and Price

In [26]:

```
plt.figure(figsize=(15,7))
plt.subplot(1,2,1)
sns.countplot(x=df['Furnishing'])
plt.subplot(1,2,2)
sns.boxplot(x=df['Furnishing'],y=df['Price'])
```

Out[26]:

```
<AxesSubplot:xlabel='Furnishing', ylabel='Price'>
```



Q. Why the price for the furnished, semi-furnished and unfurnished properties are almost in same range.

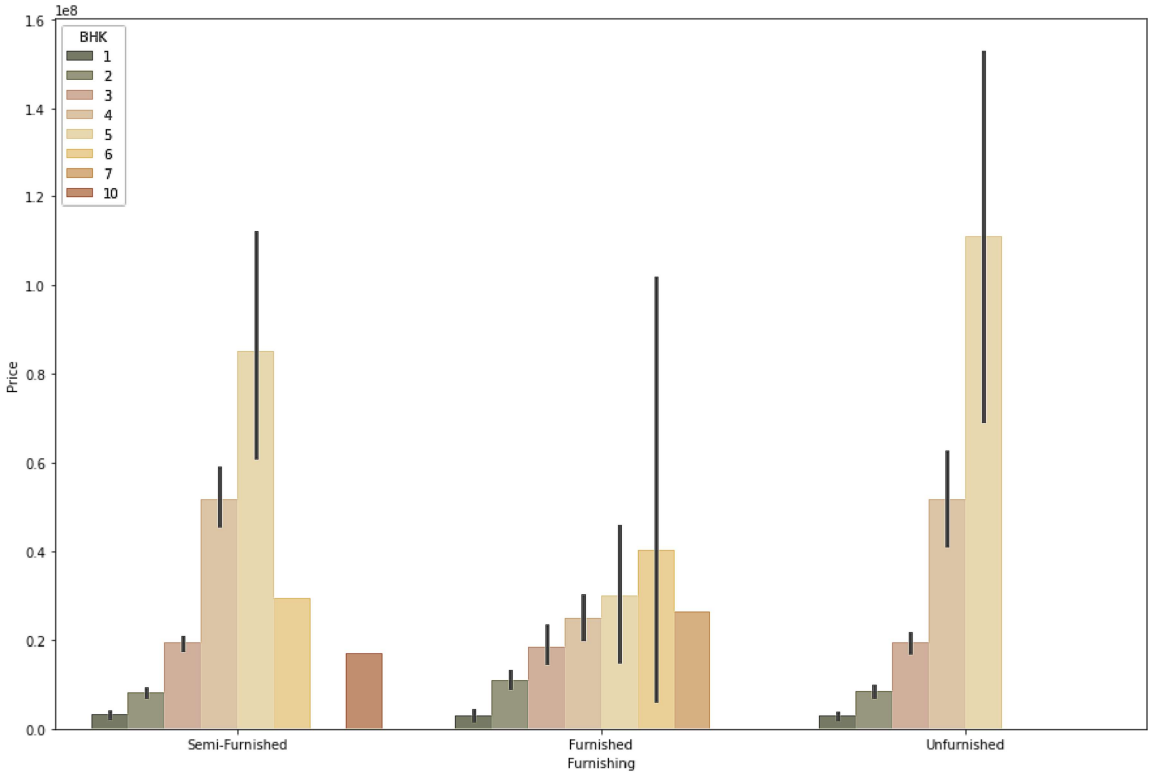
Furnishing ,Price and BHK

In [27]:

```
plt.figure(figsize=(15,10))
sns.barplot(x=df['Furnishing'],y=df['Price'],hue=df['BHK'])
```

Out[27]:

```
<AxesSubplot:xlabel='Furnishing', ylabel='Price'>
```



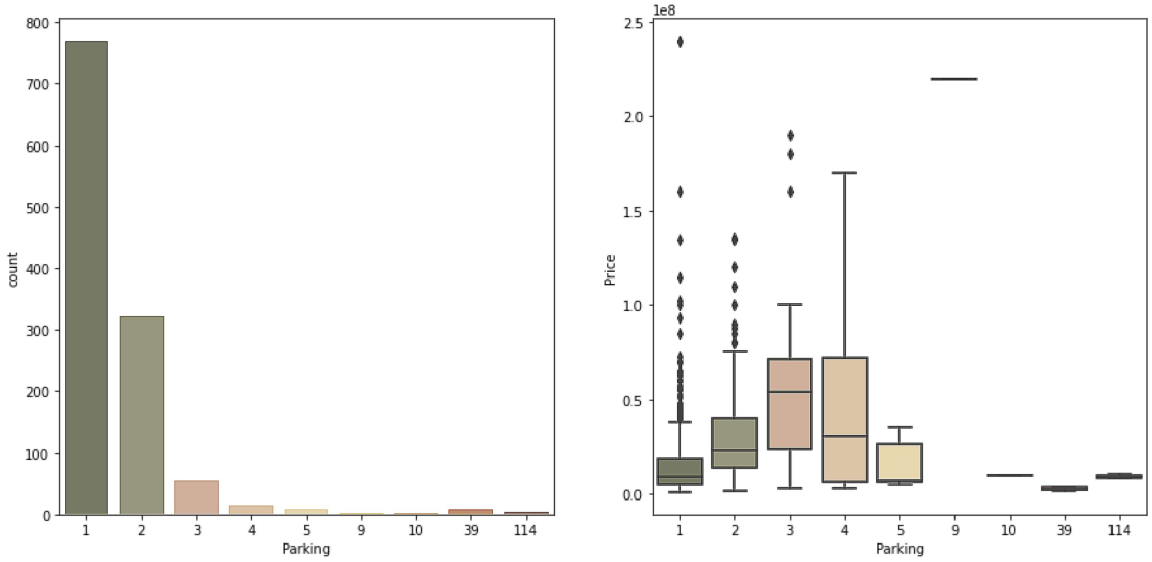
Ultimately, choosing to rent out a furnished or unfurnished property really depends on the target market. Since students and young professionals tend to rent furnished properties, while couples and families prefer unfurnished units. According to the this statement, people tend to buy unfurnished or semi-funished property because most are the customers buy property for their families to live.

Parking and Price

In [28]:

```
plt.figure(figsize=(15,7))
plt.subplot(1,2,1)
sns.countplot(x=df['Parking'])
plt.subplot(1,2,2)
sns.boxplot(x=df['Parking'],y=df['Price'])
print('Correlation between Parking and Price is',df['Parking'].corr(df['Price']))
print('Skewness of the Parking is',df['Parking'].skew())
```

```
Correlation between Parking and Price is -0.0012735025532615509
Skewness of the Parking is 14.819394368883357
```



1. There is some suspicion in the data as no property can have such huge number of parkings 39 or 114, there is something wrong in the data.
2. Let's remove these properties from our data.

In [29]:

```
df.drop(df.index[(df["Parking"] == 39)],axis=0,inplace=True)
df.drop(df.index[(df["Parking"] == 114)],axis=0,inplace=True)
```

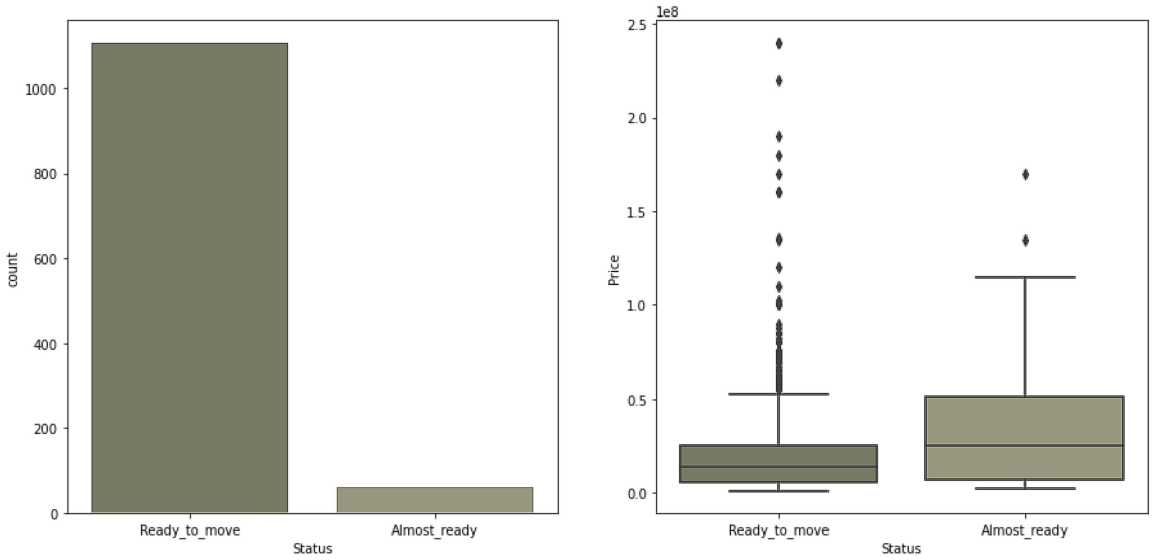
Status and Price

In [30]:

```
plt.figure(figsize=(15,7))
plt.subplot(1,2,1)
sns.countplot(x=df['Status'])
plt.subplot(1,2,2)
sns.boxplot(x=df['Status'],y=df['Price'])
```

Out[30]:

```
<AxesSubplot:xlabel='Status', ylabel='Price'>
```



Almost all the properties are ready to move in.

Q. Why the prices of the properties high which are not ready to move in yet.

In [31]:

```
temp=df.index[df['Status']=='Almost_ready']
```

In [32]:

```
dtemp=df.index[df['Status']=='Ready_to_move']
```

In [33]:

```
df.loc[temp].head(20)
```

Out[33]:

	Area	BHK	Bathroom	Furnishing	Locality	Parking	Price	Status	Tra
8	985.0	3	3	Unfurnished	Rohini Sector	1	6800000	Almost_ready	New_
67	700.0	2	1	Semi-Furnished	Dwarka Sector	1	2400000	Almost_ready	New_
131	1800.0	3	3	Semi-Furnished	Lajpat Nagar	2	35000000	Almost_ready	New_
209	4688.0	5	4	Unfurnished	Other	1	135000000	Almost_ready	New_
211	3901.0	4	4	Unfurnished	Other	1	93000000	Almost_ready	New_
224	2900.0	5	5	Furnished	Other	1	51000000	Almost_ready	New_
225	5025.0	5	6	Unfurnished	Kailash Colony	4	170000000	Almost_ready	New_
226	1000.0	3	2	Semi-Furnished	Other	1	5510000	Almost_ready	New_
227	1135.0	3	3	Furnished	Other	2	25000000	Almost_ready	
228	2306.0	3	3	Unfurnished	Other	1	57000000	Almost_ready	New_
229	850.0	2	2	Semi-Furnished	Other	1	2940000	Almost_ready	New_
234	22050.0	4	4	Semi-Furnished	Kailash Colony	2	51000000	Almost_ready	New_
279	3690.0	4	5	Semi-Furnished	Kailash Colony	3	79000000	Almost_ready	New_
322	900.0	3	2	Semi-Furnished	Shahdara	1	6000000	Almost_ready	New_
325	1650.0	4	3	Furnished	Shahdara	1	14000000	Almost_ready	New_
394	1000.0	3	2	Semi-Furnished	Other	1	5510000	Almost_ready	New_
428	1900.0	3	3	Semi-Furnished	Other	1	55300000	Almost_ready	New_
532	1650.0	4	3	Furnished	Shahdara	1	14000000	Almost_ready	New_
544	1450.0	3	3	Semi-Furnished	Other	1	34000000	Almost_ready	New_
762	800.0	2	2	Semi-Furnished	Other	1	5000000	Almost_ready	

In [34]:

```
df.loc[dtemp].head(20)
```

Out[34]:

	Area	BHK	Bathroom	Furnishing	Locality	Parking	Price	Status	Tra
0	800.0000	3	2	Semi-Furnished	Rohini Sector	1	6500000	Ready_to_move	New
1	750.0000	2	2	Semi-Furnished	Rohini Sector	1	5000000	Ready_to_move	New
2	950.0000	2	2	Furnished	Rohini Sector	1	15500000	Ready_to_move	
3	600.0000	2	2	Semi-Furnished	Rohini Sector	1	4200000	Ready_to_move	
4	650.0000	2	2	Semi-Furnished	Rohini Sector	1	6200000	Ready_to_move	New
5	1300.0000	4	3	Semi-Furnished	Rohini Sector	1	15500000	Ready_to_move	New
6	1350.0000	4	3	Semi-Furnished	Rohini Sector	1	10000000	Ready_to_move	
7	650.0000	2	2	Semi-Furnished	Rohini Sector	1	4000000	Ready_to_move	New
9	1300.0000	4	4	Semi-Furnished	Rohini Sector	1	15000000	Ready_to_move	New
10	1100.0000	3	2	Semi-Furnished	Rohini Sector	1	6200000	Ready_to_move	New
11	870.0000	3	2	Semi-Furnished	Rohini Sector	1	7700000	Ready_to_move	New
12	630.0000	2	2	Semi-Furnished	Rohini Sector	1	5500000	Ready_to_move	New
13	660.0000	2	2	Semi-Furnished	Rohini Sector	1	5000000	Ready_to_move	
14	344.4448	2	2	Semi-Furnished	Rohini Sector	1	3310000	Ready_to_move	
15	660.0000	2	2	Semi-Furnished	Rohini Sector	1	4700000	Ready_to_move	New
16	550.0000	2	2	Semi-Furnished	Rohini Sector	1	4500000	Ready_to_move	New
17	1100.0000	4	3	Semi-Furnished	Rohini Sector	1	17000000	Ready_to_move	New
18	1150.0000	3	3	Semi-Furnished	Rohini Sector	1	25000000	Ready_to_move	
19	650.0000	2	2	Semi-Furnished	Rohini Sector	1	6000000	Ready_to_move	
20	850.0000	2	2	Semi-Furnished	Rohini Sector	1	11000000	Ready_to_move	

There can be some possible reasons for the higher prices of properties which are not ready yet-

1. The property is newly built.

2. The area and bhk of these properties are more than 'almost ready' properties.

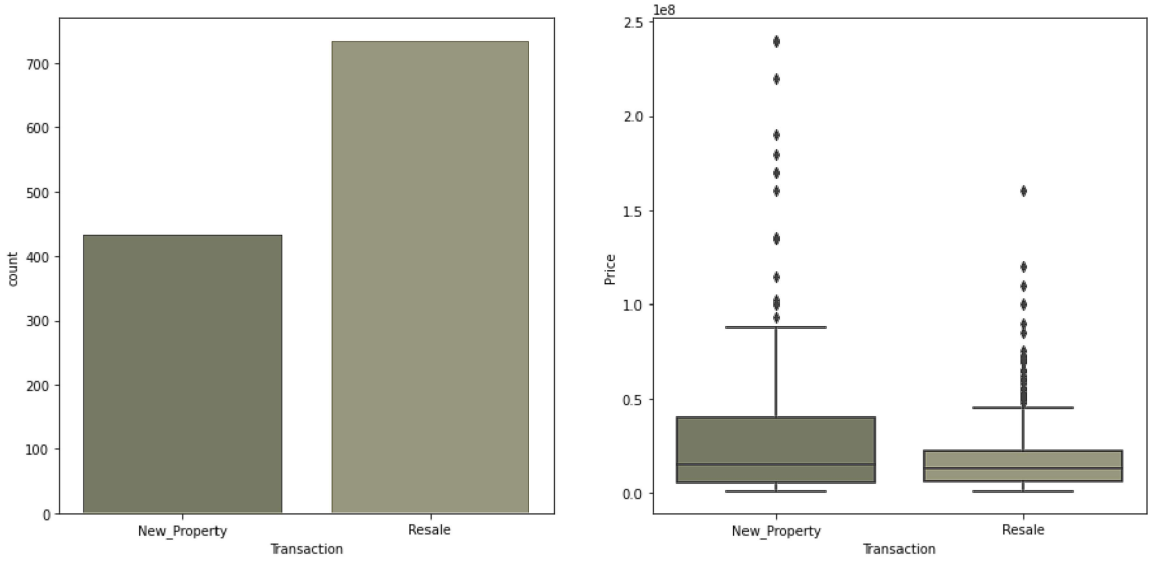
Transaction and Price

In [35]:

```
plt.figure(figsize=(15,7))
plt.subplot(1,2,1)
sns.countplot(x=df['Transaction'])
plt.subplot(1,2,2)
sns.boxplot(x=df['Transaction'],y=df['Price'])
```

Out[35]:

```
<AxesSubplot:xlabel='Transaction', ylabel='Price'>
```



1. As expected, the prices of newly constructed properties are more than the properties going for the resale.
2. That is the possible reason for the higher prices of almost ready properties than the properties that are ready to move in.

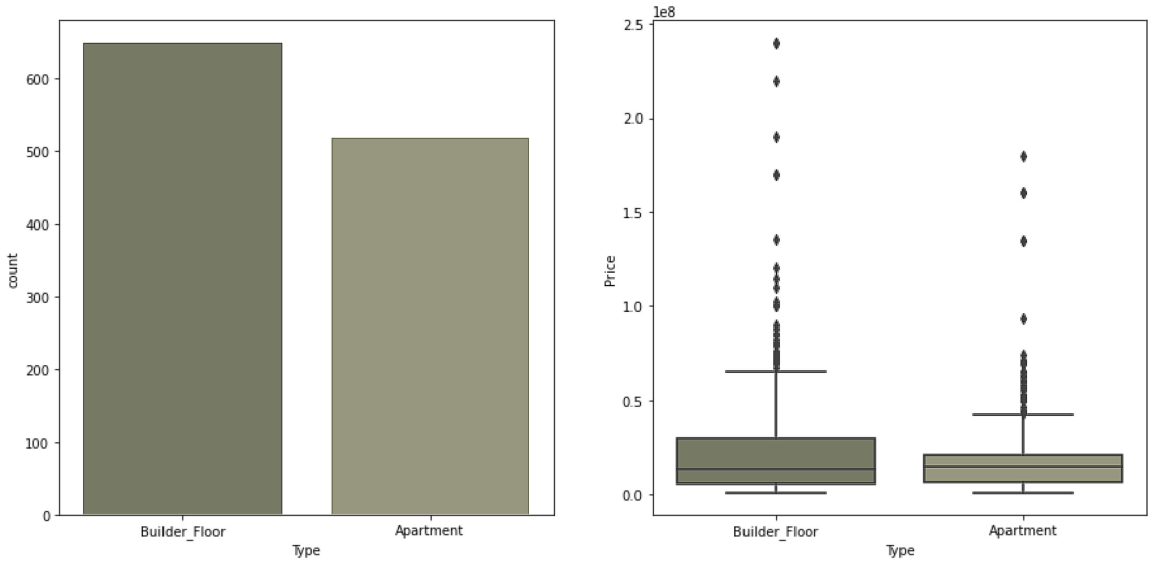
Type and Price

In [36]:

```
plt.figure(figsize=(15,7))
plt.subplot(1,2,1)
sns.countplot(x=df['Type'])
plt.subplot(1,2,2)
sns.boxplot(x=df['Type'],y=df['Price'])
```

Out[36]:

```
<AxesSubplot:xlabel='Type', ylabel='Price'>
```



1. There is slight difference in demand of Builder_floor than apartment.
2. The prices of Builder_floor is more than Apartment.
3. May be Builder_floor is the choice of the most people.

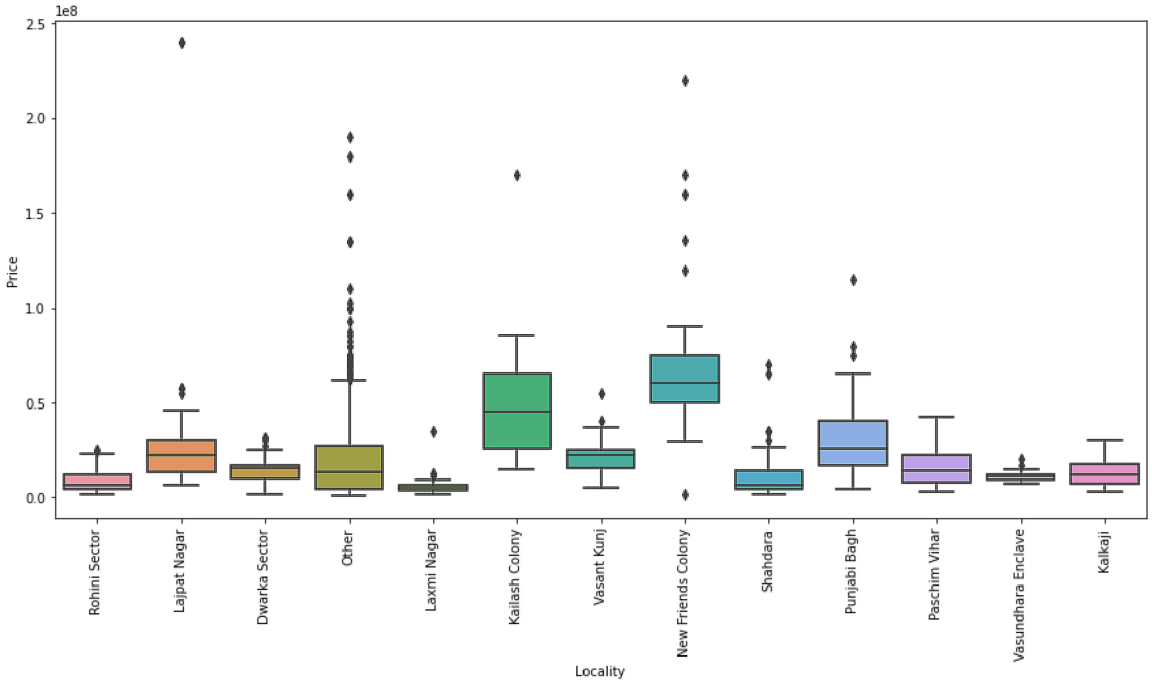
Locality and Price

In [37]:

```
plt.figure(figsize=(15,7))
sns.boxplot(x=df['Locality'],y=df['Price'])
plt.xticks(rotation=90)
```

Out[37]:

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12]),
 [Text(0, 0, 'Rohini Sector'),
  Text(1, 0, 'Lajpat Nagar'),
  Text(2, 0, 'Dwarka Sector'),
  Text(3, 0, 'Other'),
  Text(4, 0, 'Laxmi Nagar'),
  Text(5, 0, 'Kailash Colony'),
  Text(6, 0, 'Vasant Kunj'),
  Text(7, 0, 'New Friends Colony'),
  Text(8, 0, 'Shahdara'),
  Text(9, 0, 'Punjabi Bagh'),
  Text(10, 0, 'Paschim Vihar'),
  Text(11, 0, 'Vasundhara Enclave'),
  Text(12, 0, 'Kalkaji')])
```



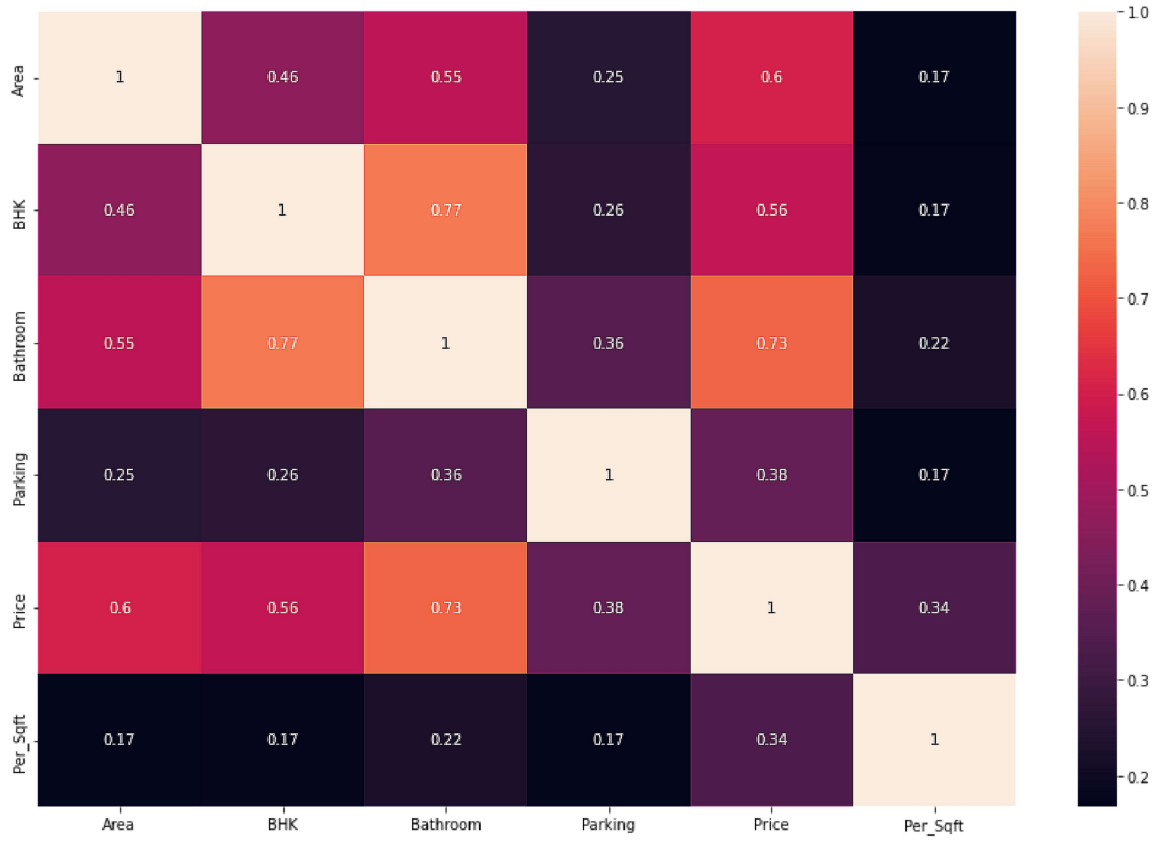
Correlation Heatmap

In [38]:

```
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(),annot=True)
```

Out[38]:

<AxesSubplot:>



1. There is a direct correlation between BHK and Area.
2. There is a direct correlation between Bathroom Area and BHK.
3. There is a direct correlation between Price, BHK ,Bathroom and Area.
4. There is a slight correlation between Price and Parking.
5. There is a correlation between Price and Per_sqft as Per_sqft is derived from Price ans Area itself.

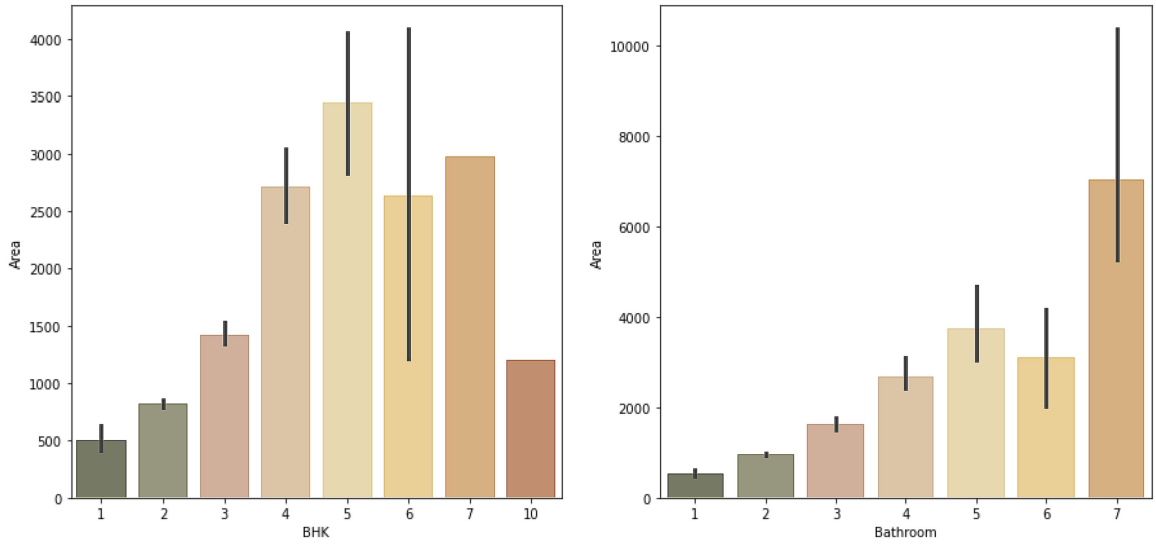
BHK, Bathroom and Area

In [39]:

```
plt.figure(figsize=(15,7))
plt.subplot(121)
sns.barplot(x=df['BHK'],y=df['Area'])
plt.subplot(122)
sns.barplot(x=df['Bathroom'],y=df['Area'])
```

Out[39]:

```
<AxesSubplot:xlabel='Bathroom', ylabel='Area'>
```



1. The area for 6,7 and 10 BHK is less than 5 BHK that explains the price difference of 5 BHK and 6,7,10 BHK properties.

Date Preprocessing

Let's remove the outliers .

In [40]:

```
from scipy import stats
z = np.abs(stats.zscore(df[df.dtypes[df.dtypes != 'object'].index]))
df = df[(z < 3).all(axis=1)]
```

In [41]:

```
df.shape
```

Out[41]:

```
(1090, 11)
```

In [42]:

```
df.dtypes
```

Out[42]:

```
Area           float64
BHK              int64
Bathroom         int64
Furnishing      object
Locality        object
Parking          int64
Price            int64
Status          object
Transaction     object
Type            object
Per_Sqft       float64
dtype: object
```

Label Encoding

In [43]:

```
from sklearn.preprocessing import LabelEncoder
```

In [44]:

```
le=LabelEncoder()
col=['Furnishing','Locality','Status','Transaction','Type']
for i in col:
    le.fit(df[i])
    df[i]=le.transform(df[i])
    print(i,df[i].unique())
```

```
Furnishing [1 0 2]
Locality [ 9  3  0  6  4  1 11  5 10  7 12  8  2]
Status [1 0]
Transaction [0 1]
Type [1 0]
```

Train_Test_Split

In [45]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(df.drop('Price',axis=1),df['Price'],test_s
```

Decision Tree Regressor

In [46]:

```
from sklearn.tree import DecisionTreeRegressor
dtr=DecisionTreeRegressor(max_depth=6)
```

In [47]:

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

Out[47]:

```
DecisionTreeRegressor
DecisionTreeRegressor(max_depth=6)
```

In [48]:

```
dtr_pred=dtr.predict(X_test)
```

Evaluting Decision Tree Regressor Model

In [49]:

```
dft=pd.DataFrame({'Actual_Price':y_test,'Predicted_Price':dtr_pred})
dft.reset_index(drop=True,inplace=True)
dft.head()
```

Out[49]:

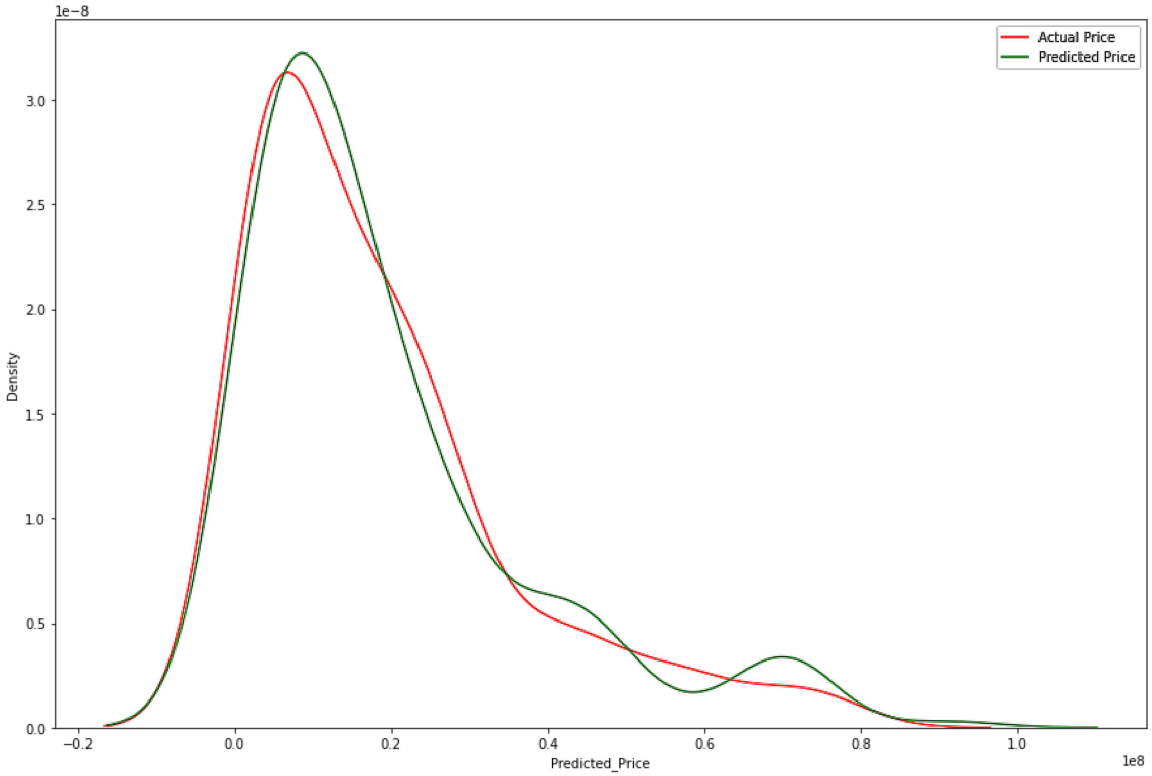
	Actual_Price	Predicted_Price
0	42500000	4.666667e+07
1	17500000	1.400689e+07
2	3300000	6.410143e+06
3	13100000	1.400689e+07
4	14500000	1.565505e+07

In [50]:

```
plt.figure(figsize=(15,10))
sns.distplot(dft['Actual_Price'],color='red',label='Actual Price',hist=False)
sns.distplot(dft['Predicted_Price'],color='green',label='Predicted Price',hist=False)
plt.legend()
```

Out[50]:

```
<matplotlib.legend.Legend at 0x273212af790>
```



In [51]:

```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
print('R2_Score:',r2_score(y_test,dtr_pred))
#print('Mean_Squared_Error:',mean_squared_error(y_test,dtr_pred))
#print('Mean_absolute_error:',mean_absolute_error(y_test,dtr_pred))
#print("Root Mean Squared Error: ", np.sqrt(mean_squared_error(y_test, dtr_pred)))
```

```
R2_Score: 0.7574758346730102
```

Random Forest Regressor

In [52]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [53]:

```
rfr=RandomForestRegressor()
```

In [54]:

```
# for training the random forest without passing headers
# for pickel file
# df_wout_col=X_train
# df_wout_col.columns = [None] * len(df_wout_col.columns)
```

In [55]:

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

Out[55]:

RandomForestRegressor

RandomForestRegressor()

In [56]:

```
#import joblib
#joblib.dump(rfr, 'rfr.pkl')
```

In [57]:

```
r_pred=rfr.predict(X_test)
```

Evaluating Random Forest

In [58]:

```
dfr=pd.DataFrame({'Actual':y_test,'Predicted':r_pred})
dfr.reset_index(drop=True,inplace=True)
dfr.head()
```

Out[58]:

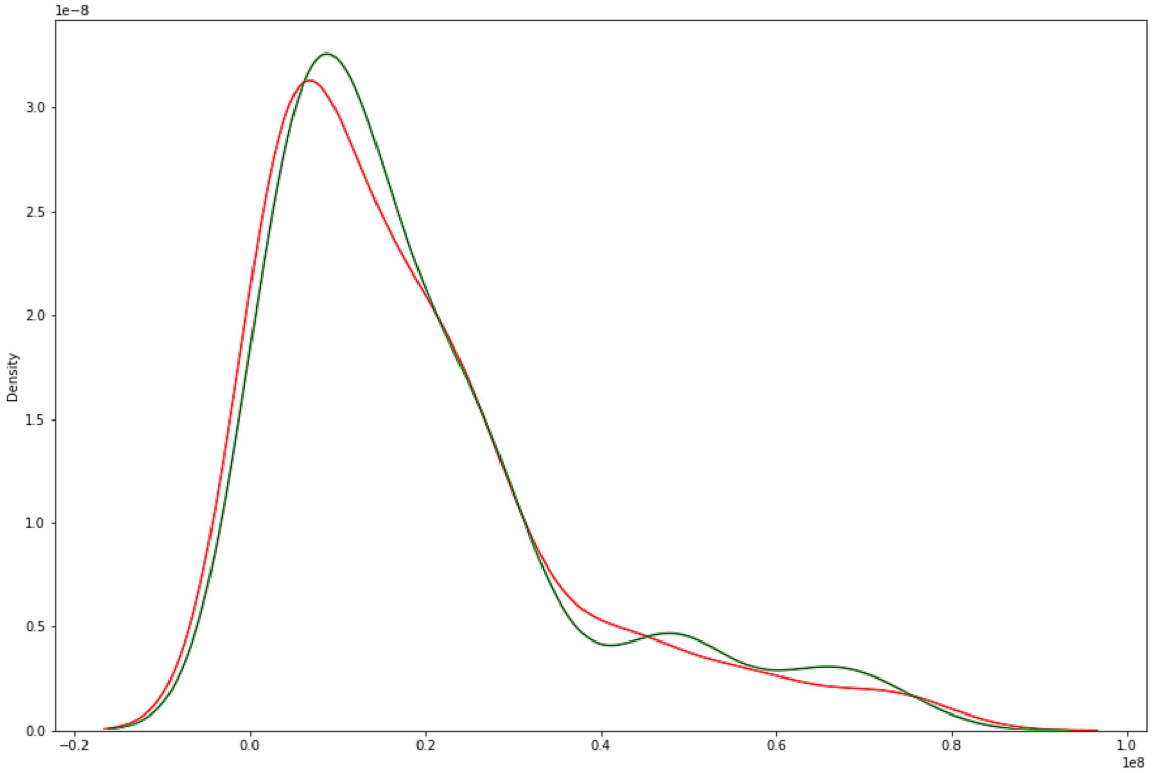
	Actual	Predicted
0	42500000	47057000.0
1	17500000	10544700.0
2	3300000	4689000.0
3	13100000	15858200.0
4	14500000	10649400.0

In [59]:

```
plt.figure(figsize=(15,10))
sns.distplot(x=dfr['Actual'],color='red',hist=False,label='Actual_Price')
sns.distplot(x=dfr['Predicted'],color='green',hist=False,label='Predicted_Price')
```

Out[59]:

```
<AxesSubplot:ylabel='Density'>
```



In [60]:

```
print('R2_Score:',r2_score(y_test,r_pred))
#print('Mean_Squared_Error:',mean_squared_error(y_test,r_pred))
#print('Mean_absolute_error:',mean_absolute_error(y_test,r_pred))
#print("Root Mean Squared Error: ", np.sqrt(mean_squared_error(y_test, r_pred)))
```

```
R2_Score: 0.868625153730545
```

XGBoost Regressor

In [61]:

```
from xgboost import XGBRegressor
RegModel=XGBRegressor(max_depth=3, learning_rate=0.1, n_estimators=500, objective='reg:sq
```

In [62]:

```
XGB=RegModel.fit(X_train,y_train)
xgb_prediction=XGB.predict(X_test)
```

Evaluating XGBoost

In [63]:

```
xgb=pd.DataFrame({'Actual':y_test,'Predicted':xgb_prediction})
xgb.reset_index(drop=True,inplace=True)
xgb.head()
```

Out[63]:

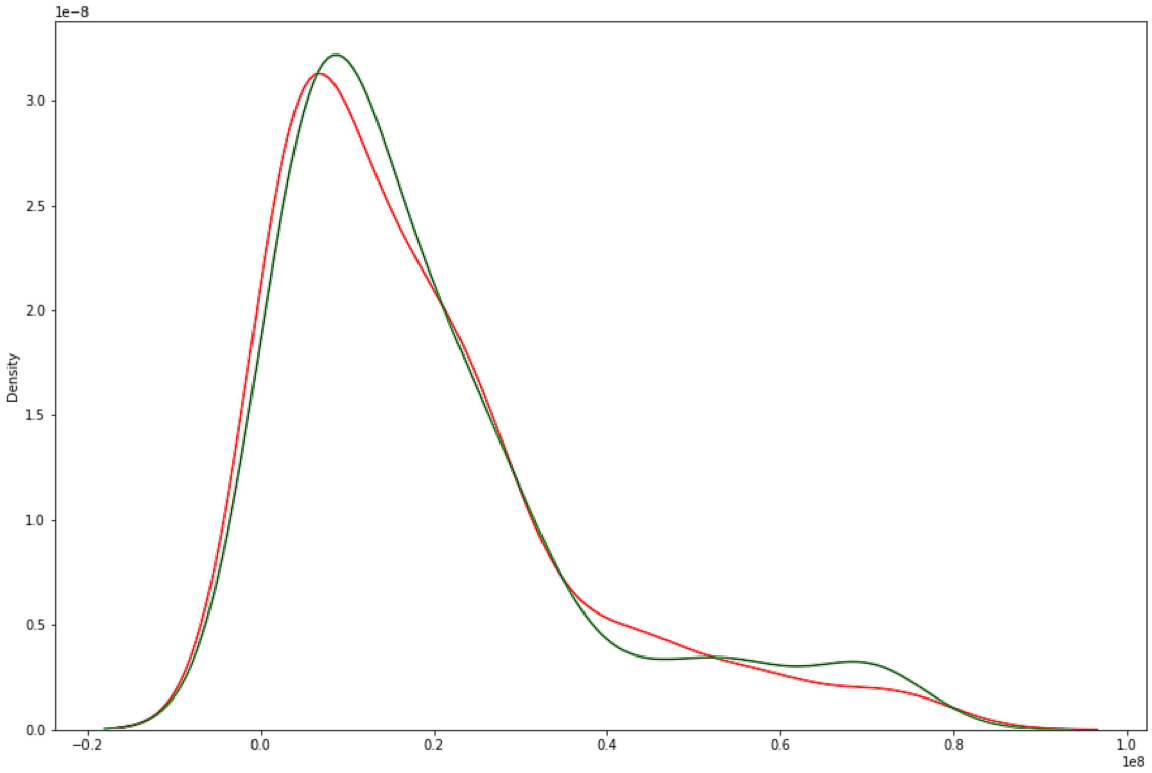
	Actual	Predicted
0	42500000	43905644.00
1	17500000	11755837.00
2	3300000	3928304.75
3	13100000	14460811.00
4	14500000	10682964.00

In [64]:

```
plt.figure(figsize=(15,10))
sns.distplot(x=xgb['Actual'],color='red',hist=False,label='Actual_Price')
sns.distplot(x=xgb['Predicted'],color='green',hist=False,label='Predicted_Price')
```

Out[64]:

```
<AxesSubplot:ylabel='Density'>
```



In [65]:

```
print('R2_Score:',r2_score(y_test,xgb_prediction))
#print('Mean_Squared_Error:',mean_squared_error(y_test,xgb_prediction))
#print('Mean_absolute_error:',mean_absolute_error(y_test,xgb_prediction))
#print("Root Mean Squared Error: ", np.sqrt(mean_squared_error(y_test, xgb_prediction)))
```

```
R2_Score: 0.8425106209651492
```

```python
from flask import Flask, render_template, url_for, flash, redirect
import joblib
from flask import request
import numpy as np

app = Flask(__name__)

@app.route('/')

@app.route("/House")
def home():
    return render_template("index.html")

@app.route('/predict', methods=["POST"])
def predict():
    if request.method == "POST":
        Area = float(request.form['Area'])
        BHK = int(request.form['BHK'])
        Bathroom = int(request.form['Bathroom'])
        Furnishing = request.form['Furnishing']
        Furnishing = 1 if Furnishing == 'Semi-Furnished' else 0
        Locality=request.form['Locality']
        if Locality=='Rohini Sector':
            Locality=9
        elif Locality=='Dwarka Sector':
            Locality=0
        elif Locality=='Shahdara':
            Locality=10
        elif Locality=='Vasant Kunj':
            Locality=11
        elif Locality=='Paschim Vihar':
            Locality=7
        elif Locality=='Vasundhara Enclave':
            Locality=12
        elif Locality=='Punjabi Bagh':
            Locality=8
```

```
        elif Locality== Lajpat Nagar :
            Locality=3
        elif Locality=='Laxmi Nagar':
            Locality=4
        elif Locality=='New Friends Colony':
            Locality=5
        elif Locality=='Kailash Colony':
            Locality=1
        else:
            Locality=6

        Parking = int(request.form['Parking'])
        Status = request.form['Status']
        Status = 1 if Status == 'Ready_to_move' else 0

        Transaction = request.form['Transaction']
        Transaction = 0 if Transaction == 'New_Property' else 1

        Type = request.form['Type']
        Type = 1 if Type == 'Builder_Floor' else 0

        Per_Sqft = float(request.form['Per_Sqft'])

        loaded_model = joblib.load(r'C:\Users\guptanuj890\OneDrive\Desktop\pro\Projects\Delhi House Price Prediction\House_Price_Website\models\rfr.pkl')

        # Create a list of features in the correct order and format
        input_data = [Area, BHK, Bathroom, Furnishing, Locality, Parking, Status, Transaction, Type, Per_Sqft]

        # Predict using the model
        prediction_result = loaded_model.predict([input_data])
        result = prediction_result[0]

    return render_template("index.html", prediction_text=result)
if __name__ == '__main__':

    app.run(debug=True)
```

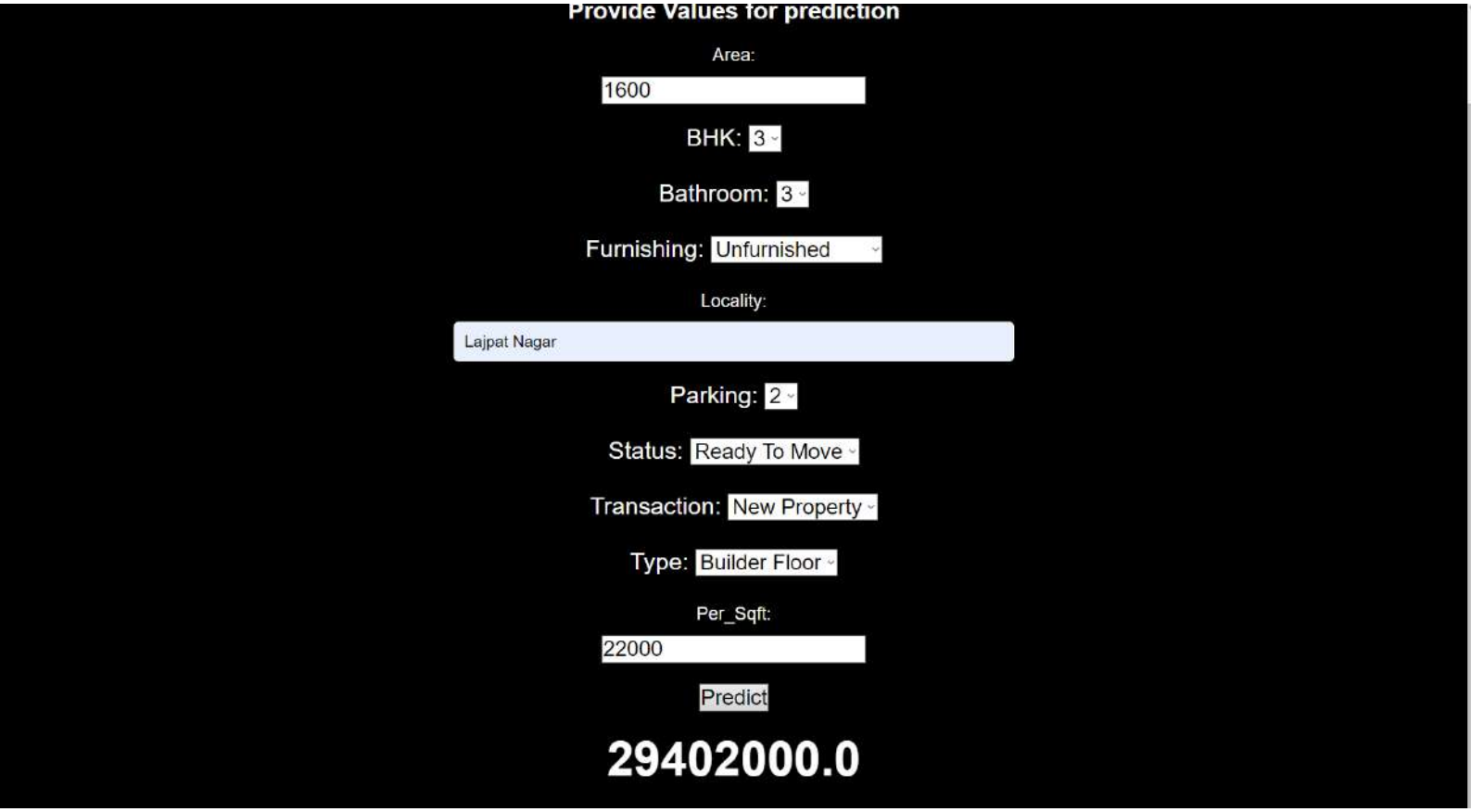
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Delhi House Price Prediction</title>
    <link rel="stylesheet" href="static/styles.css">
</head>
<body>
    <header>
        <!-- <img src="images/wine.png" alt="Your Logo"> -->
        <div class="logo">
            <img src="static/images/House.png" alt="Your Logo">
        </div>
        <h1>House Price Prediction</h1>

    </header>
    <main>
        <div class="container">
            <h2>Provide Values for prediction</h2>
            <form action="{{url_for('predict')}}" method="post">
                <div class="form-group">
                    <label for="Area">Area:</label>
                    <input type="number" id="Area" name="Area" required>
                </div>
                <label for="BHK">BHK:</label>
                <select id="BHK" name="BHK">
                    <option value="1">1</option>
                    <option value="2">2</option>
                    <option value="3">3</option>
                    <option value="4">4</option>
                    <option value="5">5</option>
                </select>
                <br>
                <br>
                <label for="Bathroom">Bathroom:</label>
                <select id="Bathroom" name="Bathroom">
```

```
<select id="Bathroom" name="Bathroom">
    <option value="1">1</option>
    <option value="2">2</option>
    <option value="3">3</option>
    <option value="4">4</option>
    <option value="5">5</option>
</select>
<br>
<br>
<label for="Furnishing">Furnishing:</label>
<select id="Furnishing" name="Furnishing">
    <option value="Semi-Furnished">Semi-Furnished</option>
    <option value="Furnished">Furnished</option>
    <option value="Unfurnished">Unfurnished</option>
</select>
<br>
<br>
<div class="form-group">
    <label for="Locality">Locality:</label>
    <input type="text" id="Locality" name="Locality" required>
</div>
<label for="Parking">Parking:</label>
<select id="Parking" name="Parking">
    <option value="1">1</option>
    <option value="2">2</option>
    <option value="3">3</option>
</select>
<br>
<br>
<label for="Status">Status:</label>
<select id="Status" name="Status">
    <option value="Ready To Move">Ready To Move</option>
    <option value="Almost Ready">Almost Ready</option>
</select>
<br>
<br>
```

```
            <label for="Transaction">Transaction:</label>
            <select id="Transaction" name="Transaction">
                <option value="New Property">New Property</option>
                <option value="Resale">Resale</option>
            </select>
            <br>
            <br>
            <label for="Type">Type:</label>
            <select id="Type" name="Type">
                <option value="Apartment">Apartment</option>
                <option value="Builder Floor">Builder Floor</option>
            </select>
            <br>
            <br>
            <div class="form-group">
                <label for="Per_Sqft">Per_Sqft:</label>
                <input type="number" id="Per_Sqft" name="Per_Sqft" required>
            </div>

            <div class="form-group">
                <input type="submit" value="Predict">
            </div>
        </form>
        <h1> {{ prediction_text }}</h1>
    </div>
</main>
<footer>
    <p>&copy; 2023 Harry's Homes</p>
</footer>
</body>
</html>
```


Provide Values for prediction
Area:
1600
BHK: 3
Bathroom: 3
Furnishing: Unfurnished
Locality:
Lajpat Nagar
Parking: 2
Status: Ready To Move
Transaction: New Property
Type: Builder Floor
Per_Sqft:
22000
Predict
29402000.0