## INTRODUCTION:

House price prediction can help the analyst determine the selling price of a house and can help the customer to arrange the right time to purchase a house. Housing prices keep changing day in and day out and sometimes are hyped rather than being based on valuation.There are three factors that influence the price of a house which include **physical conditions**, **concept** and **location**.

## About Dataset

This dataset provides comprehensive information for house price prediction, with 13 column names:

**Price**: bold text The price of the house. **Area:** The total area of the house in square feet. **Bedrooms:** The number of bedrooms in the house. **Bathrooms:** The number of bathrooms in the house. **Stories:** The number of stories in the house. **Mainroad:** Whether the house is connected to the main road (Yes/No). **Guestroom:** Whether the house has a guest room (Yes/No). **Basement:** Whether the house has a basement (Yes/No). **Hot water heating:** Whether the house has a hot water heating system (Yes/No). **Airconditioning:** Whether the house has an air conditioning system (Yes/No). **Parking:** The number of parking spaces available within the house. **Prefarea:** Whether the house is located in a preferred area (Yes/No). **Furnishing status:** The furnishing status of the house (Fully Furnished, Semi-Furnished, Unfurnished).

## Importing libraries

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

## Reading Dataset

```
housing_price = pd.read_csv("Housing.csv")
housing_price.head()
```

| | price | area | bedrooms | bathrooms | stories | mainroad | guestroom | basement | hotwaterheating | airconditioning | parking | prefarea | furn |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 13300000 | 7420 | 4 | 2 | 3 | yes | no | no | no | yes | 2 | yes | |
| **1** | 12250000 | 8960 | 4 | 4 | 4 | yes | no | no | no | yes | 3 | no | |
| **2** | 12250000 | 9960 | 3 | 2 | 2 | yes | no | yes | no | no | 2 | yes | |
| **3** | 12215000 | 7500 | 4 | 2 | 2 | yes | no | yes | no | yes | 3 | yes | |
| **4** | 11410000 | 7420 | 4 | 1 | 2 | yes | yes | yes | no | yes | 2 | no | |

```
housing_price.shape
```

```
(545, 13)
```

```
housing_price.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
```

```
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             545 non-null    int64
 1   area              545 non-null    int64
 2   bedrooms          545 non-null    int64
 3   bathrooms         545 non-null    int64
 4   stories           545 non-null    int64
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

```
housing_price.columns
```

```
Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'mainroad',
       'guestroom', 'basement', 'hotwaterheating', 'airconditioning',
       'parking', 'prefarea', 'furnishingstatus'],
      dtype='object')
```

```
housing_price.describe()
```

| | 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 |

# Data Preprocessing

## checking for duplicates

```
housing_price.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
540    False
541    False
542    False
543    False
544    False
Length: 545, dtype: bool
```

```
housing_price.duplicated().sum()
```

```
0
```

There are no duplicates in this dataset

## checking for missing values:

```
housing_price.isnull().sum()
```

```
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
```

There are no missing values in the dataset

## checking for value counts

```
airconditioning_count = housing_price['airconditioning'].value_counts()
basement_count = housing_price['basement'].value_counts()
bathroom_count = housing_price['bathrooms'].value_counts()
bedroom_count = housing_price['bedrooms'].value_counts()
furnishing_count = housing_price['furnishingstatus'].value_counts()
guestroom_status = housing_price['guestroom'].value_counts()
water_heating_status = housing_price['hotwaterheating'].value_counts()
stories_count = housing_price['stories'].value_counts()
mainroad_count = housing_price['mainroad'].value_counts()
parking_count = housing_price['parking'].value_counts()
prearea_count = housing_price['prefarea'].value_counts()
```

```
print(airconditioning_count)
print(basement_count)
print(bathroom_count)
print(bedroom_count)
print(furnishing_count)
```

```
no     373
yes    172
Name: airconditioning, dtype: int64
no     354
yes    191
Name: basement, dtype: int64
1    401
2    133
3     10
4      1
Name: bathrooms, dtype: int64
3    300
2    136
4     95
5     10
6      2
1      2
Name: bedrooms, dtype: int64
semi-furnished    227
unfurnished       178
furnished         140
Name: furnishingstatus, dtype: int64
```

```
print(guestroom_status)
print(water_heating_status)
print(stories_count)
print(mainroad_count)
print(parking_count)
print(prearea_count)
```

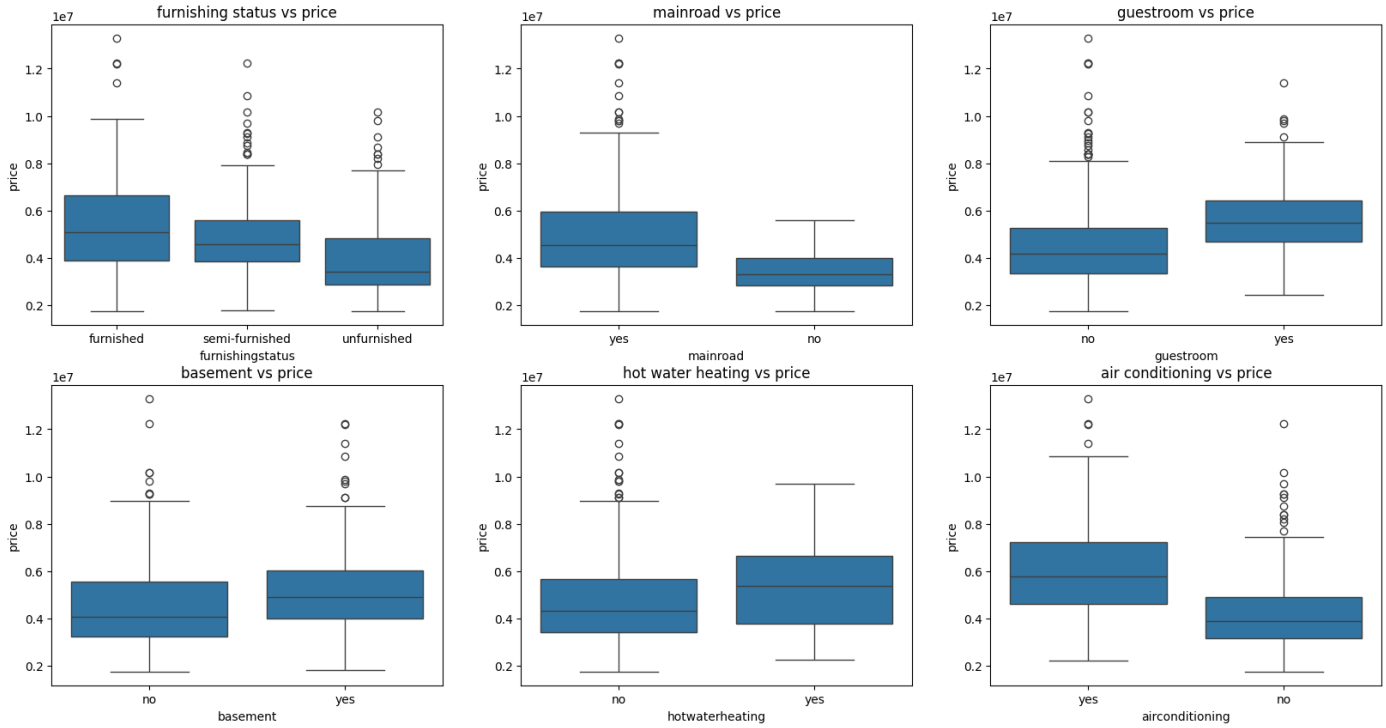
```
no     448
yes     97
```

```
Name: guestroom, dtype: int64
no     520
yes     25
Name: hotwaterheating, dtype: int64
2    238
1    227
4     41
3     39
Name: stories, dtype: int64
yes    468
no      77
Name: mainroad, dtype: int64
0    299
1    126
2    108
3     12
Name: parking, dtype: int64
no     417
yes    128
Name: prefarea, dtype: int64
```

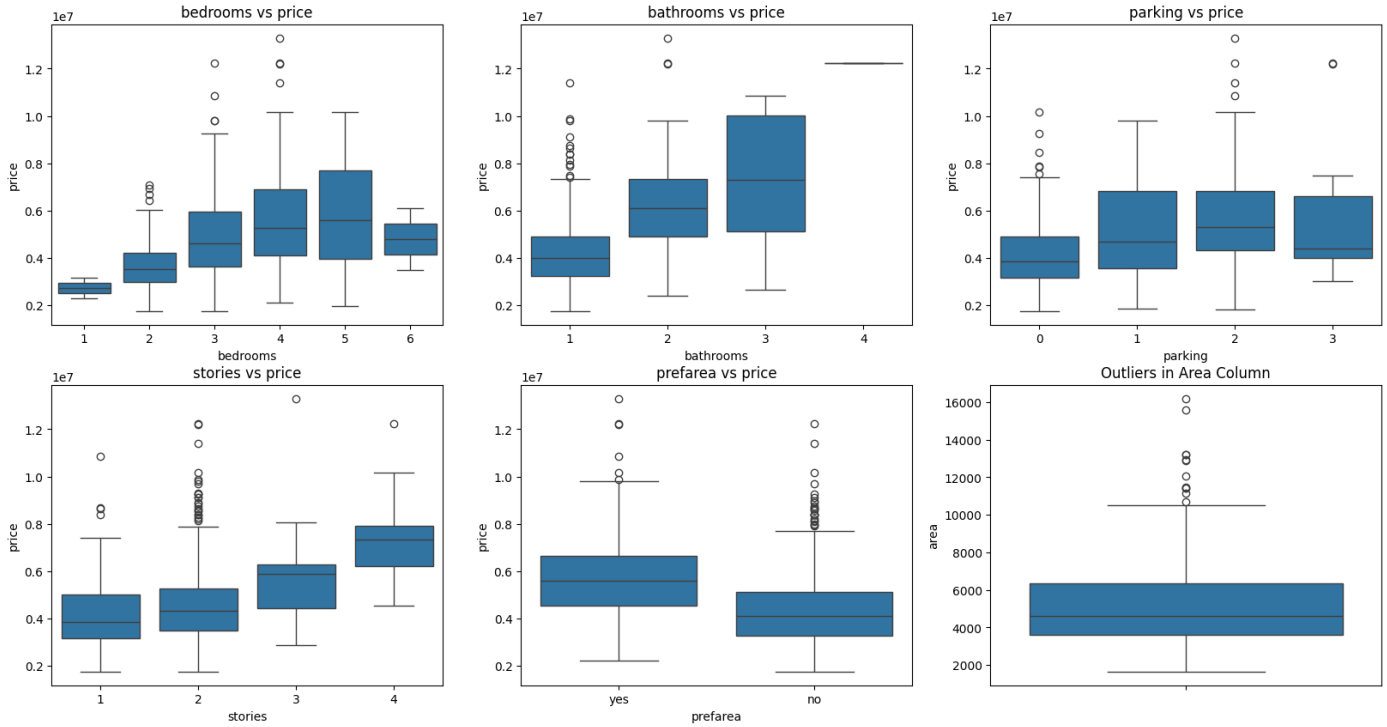
## checking for presence of outliers

Visualizing the outliers

```
plt.figure(figsize=(20,10))

plt.subplot(2, 3, 1)
sns.boxplot(data=housing_price, x='furnishingstatus', y='price')
plt.title("furnishing status vs price")

plt.subplot(2, 3, 2)
sns.boxplot(data=housing_price, x='mainroad', y='price')
plt.title("mainroad vs price")

plt.subplot(2, 3, 3)
sns.boxplot(data=housing_price, x='guestroom', y='price')
plt.title("guestroom vs price")

plt.subplot(2, 3, 4)
sns.boxplot(data=housing_price, x='basement', y='price')
plt.title("basement vs price")

plt.subplot(2, 3, 5)
sns.boxplot(data=housing_price, x='hotwaterheating', y='price')
plt.title("hot water heating vs price")

plt.subplot(2, 3, 6)
sns.boxplot(data=housing_price, x='airconditioning', y='price')
plt.title("air conditioning vs price")

plt.show()
```



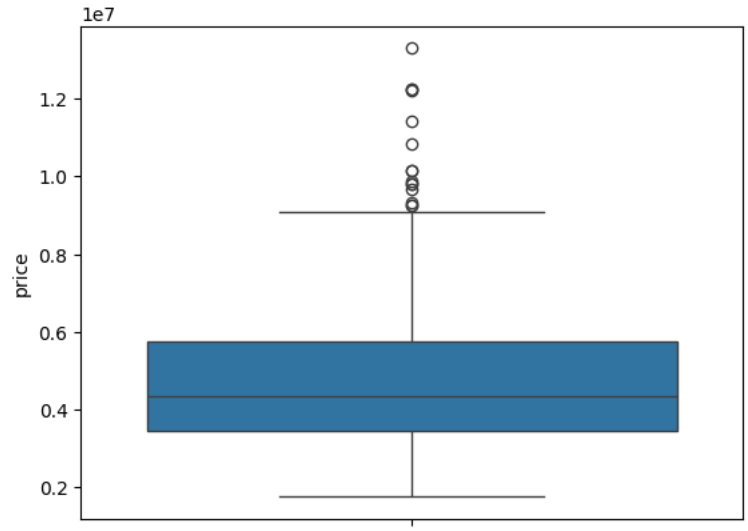

```
plt.figure(figsize=(20,10))

plt.subplot(2, 3, 1)
sns.boxplot(data=housing_price, x='bedrooms', y='price')
plt.title("bedrooms vs price")

plt.subplot(2, 3, 2)
sns.boxplot(data=housing_price, x='bathrooms', y='price')
plt.title("bathrooms vs price")

plt.subplot(2, 3, 3)
sns.boxplot(data=housing_price, x='parking', y='price')
plt.title("parking vs price")

plt.subplot(2, 3, 4)
sns.boxplot(data=housing_price, x='stories', y='price')
plt.title("stories vs price")

plt.subplot(2, 3, 5)
sns.boxplot(data=housing_price, x='prefarea', y='price')
plt.title("prefarea vs price")

plt.subplot(2, 3, 6)
sns.boxplot(housing_price['area'])
plt.title("Outliers in Area Column")
plt.show()
```




There are outliers in the price and area column. When price is plotted with other descrete features, we can see that each categories' prices have outliers.

```
sns.boxplot(housing_price['price'])
```

```
<Axes: ylabel='price'>
```



## getting the Q1, Q3 and IQR

```
price_Q1 = housing_price['price'].quantile(0.25)
price_Q3 = housing_price['price'].quantile(0.75)

price_IQR = price_Q3 - price_Q1

print(f'The Q1 is {price_Q1} and the Q3 is {price_Q3}')
print(f'The IQR is {price_IQR}')
```

```
The Q1 is 3430000.0 and the Q3 is 5740000.0
The IQR is 2310000.0
```

calculate the upper and lower bound of price column

```
lower_threshold_price = price_Q1 - 1.5*price_IQR
upper_threshold_price = price_Q3 + 1.5*price_IQR

print('The lower bound is:', lower_threshold_price)
print('The upper bound is:', upper_threshold_price)
```

```
The lower bound is: -35000.0
The upper bound is: 9205000.0
```

```
upper_array = np.array(housing_price['price'] >= upper_threshold_price)
print('The count of data higher than the upper bound is:', upper_array.sum())
```

```
The count of data higher than the upper bound is: 15
```

```
lower_array = np.array(housing_price['price'] <= lower_threshold_price)
print('The count of data lower than the lower bound is:', lower_array.sum())
```

```
The count of data lower than the lower bound is: 0
```

```
outliers_price = housing_price[(housing_price['price'] < lower_threshold_price) | (housing_price["price"]> upper_threshold_price)]
outliers_price
```

| | price | area | bedrooms | bathrooms | stories | mainroad | guestroom | basement | hotwaterheating | airconditioning | parking | prefarea | fu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 13300000 | 7420 | 4 | 2 | 3 | yes | no | no | no | yes | 2 | yes | |
| **1** | 12250000 | 8960 | 4 | 4 | 4 | yes | no | no | no | yes | 3 | no | |
| **2** | 12250000 | 9960 | 3 | 2 | 2 | yes | no | yes | no | no | 2 | yes | |
| **3** | 12215000 | 7500 | 4 | 2 | 2 | yes | no | yes | no | yes | 3 | yes | |
| **4** | 11410000 | 7420 | 4 | 1 | 2 | yes | yes | yes | no | yes | 2 | no | |
| **5** | 10850000 | 7500 | 3 | 3 | 1 | yes | no | yes | no | yes | 2 | yes | |
| **6** | 10150000 | 8580 | 4 | 3 | 4 | yes | no | no | no | yes | 2 | yes | |
| **7** | 10150000 | 16200 | 5 | 3 | 2 | yes | no | no | no | no | 0 | no | |
| **8** | 9870000 | 8100 | 4 | 1 | 2 | yes | yes | yes | no | yes | 2 | yes | |
| **9** | 9800000 | 5750 | 3 | 2 | 4 | yes | yes | no | no | yes | 1 | yes | |
| **10** | 9800000 | 13200 | 3 | 1 | 2 | yes | no | yes | no | yes | 2 | yes | |
| **11** | 9681000 | 6000 | 4 | 3 | 2 | yes | yes | yes | yes | no | 2 | no | |
| **12** | 9310000 | 6550 | 4 | 2 | 2 | yes | no | no | no | yes | 1 | yes | |
| **13** | 9240000 | 3500 | 4 | 2 | 2 | yes | no | no | yes | no | 2 | no | |
| **14** | 9240000 | 7800 | 3 | 2 | 2 | yes | no | no | no | no | 0 | yes | |

These are the 15 columns that are outliers under price column

```
# drop the outliers in price column
housing_price = housing_price[(housing_price['price'] > lower_threshold_price) & (housing_price['price'] < upper_threshold_price)]
```

```
area_Q1 = housing_price['area'].quantile(0.25)
area_Q3 = housing_price['area'].quantile(0.75)

area_IQR = area_Q3 - area_Q1

print(f'The Q1 is {area_Q1} and the Q3 is {area_Q3}')
print(area_IQR)
```

```
The Q1 is 3547.5 and the Q3 is 6315.75
2768.25
```

```
lower_threshold_area = area_Q1 - 1.5*area_IQR
upper_threshold_area = area_Q3 + 1.5*area_IQR

print('The lower bound is:', lower_threshold_area)
print('The upper bound is:', upper_threshold_area)
```

```
The lower bound is: -604.875
The upper bound is: 10468.125
```

```
upper_array_area = np.array(housing_price['area'] >= upper_threshold_area)
print('The count of data higher than the upper bound is:', upper_array_area.sum())
```

```
The count of data higher than the upper bound is: 13
```

```
lower_array_area = np.array(housing_price['price'] <= lower_threshold_area)
print('The count of data higher than the upper bound is:', lower_array_area.sum())
```

```
The count of data higher than the upper bound is: 0
```

dropping the outliers

```
outliers_area = housing_price[(housing_price['area'] < lower_threshold_area) | (housing_price["area"]> upper_threshold_area)]
outliers_area
```

| | price | area | bedrooms | bathrooms | stories | mainroad | guestroom | basement | hotwaterheating | airconditioning | parking | prefarea | fu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **56** | 7343000 | 11440 | 4 | 1 | 2 | yes | no | yes | no | no | 1 | yes | |
| **64** | 7000000 | 11175 | 3 | 1 | 1 | yes | no | yes | no | yes | 1 | yes | |
| **66** | 6930000 | 13200 | 2 | 1 | 1 | yes | no | yes | yes | no | 1 | no | |
| **69** | 6790000 | 12090 | 4 | 2 | 2 | yes | no | no | no | no | 2 | yes | |
| **82** | 6615000 | 10500 | 3 | 2 | 1 | yes | no | yes | no | yes | 1 | yes | |
| **125** | 5943000 | 15600 | 3 | 1 | 1 | yes | no | no | no | yes | 2 | no | |
| **129** | 5873000 | 11460 | 3 | 1 | 3 | yes | no | no | no | no | 2 | yes | |
| **142** | 5600000 | 10500 | 4 | 2 | 2 | yes | no | no | no | no | 1 | no | |
| **146** | 5600000 | 10500 | 2 | 1 | 1 | yes | no | no | no | no | 1 | no | |
| **186** | 5110000 | 11410 | 2 | 1 | 2 | yes | no | no | no | no | 0 | yes | |
| **191** | 5040000 | 10700 | 3 | 1 | 2 | yes | yes | yes | no | no | 0 | no | |
| **211** | 4900000 | 12900 | 3 | 1 | 1 | yes | no | no | no | no | 2 | no | |
| **403** | 3500000 | 12944 | 3 | 1 | 1 | yes | no | no | no | no | 0 | no | |

```
housing_price = housing_price[(housing_price['area'] > lower_threshold_area) & (housing_price['area'] < upper_threshold_area)]
```

```
# The new shape of the dataset after dropping the outliers
housing_price.shape
```

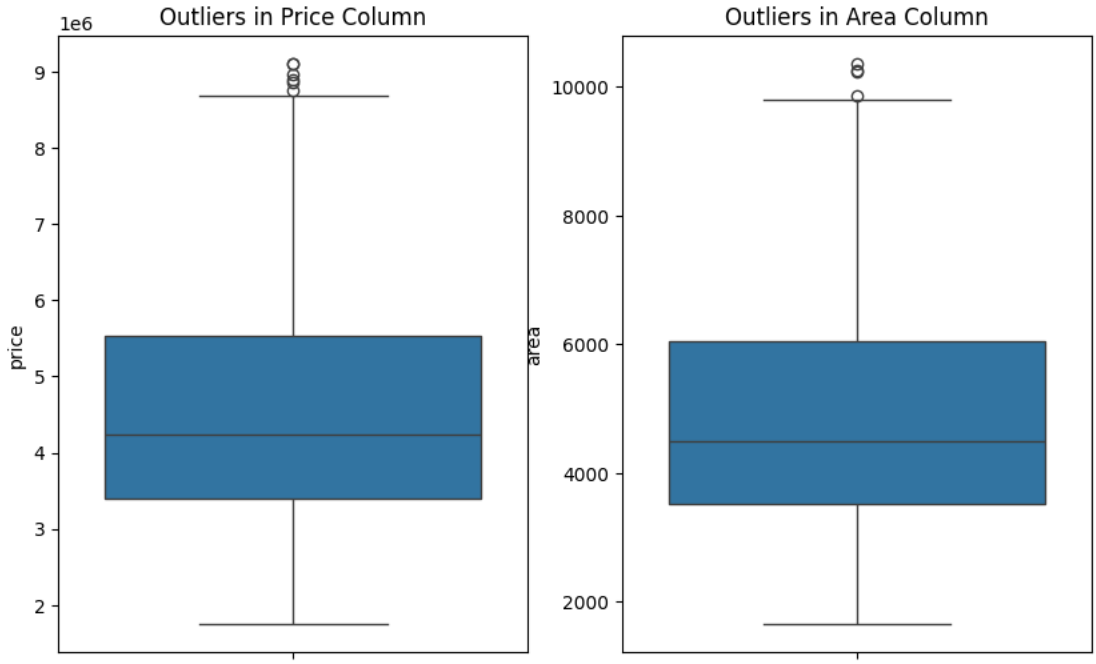
```
(517, 13)
```

```
housing_price.head()
```

| | price | area | bedrooms | bathrooms | stories | mainroad | guestroom | basement | hotwaterheating | airconditioning | parking | prefarea | furn |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **15** | 9100000 | 6000 | 4 | 1 | 2 | yes | no | yes | no | no | 2 | no | |
| **16** | 9100000 | 6600 | 4 | 2 | 2 | yes | yes | yes | no | yes | 1 | yes | |
| **17** | 8960000 | 8500 | 3 | 2 | 4 | yes | no | no | no | yes | 2 | no | |
| **18** | 8890000 | 4600 | 3 | 2 | 2 | yes | yes | no | no | yes | 2 | no | |
| **19** | 8855000 | 6420 | 3 | 2 | 2 | yes | no | no | no | yes | 1 | yes | |

visualizing after dropping outliers

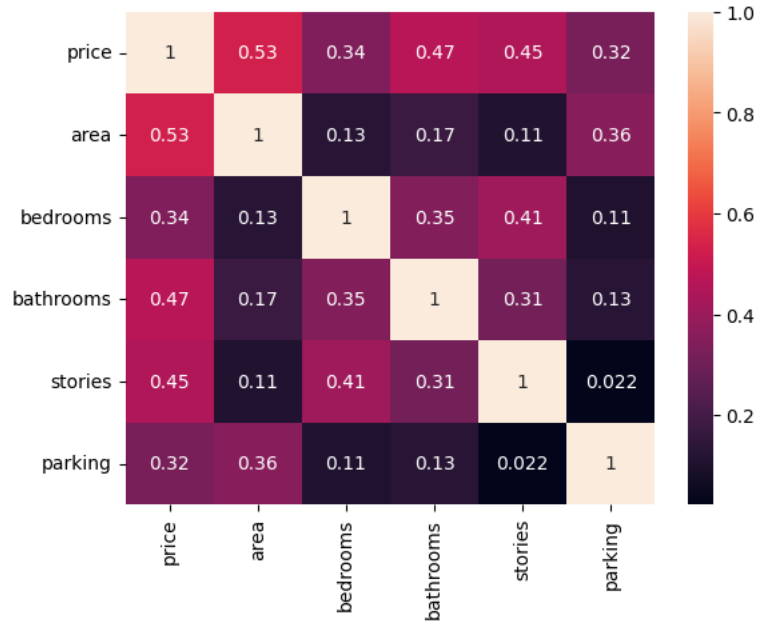
```
plt.figure(figsize=(10,6))

plt.subplot(1, 2, 1)
sns.boxplot(housing_price['price'])
plt.title("Outliers in Price Column")

plt.subplot(1, 2, 2)
sns.boxplot(housing_price['area'])
plt.title("Outliers in Area Column")
plt.show()
```




## EDA: Exploratory Data Analysis and Visualisation

```
housing_price_corr = housing_price[["price","area","bedrooms","bathrooms","stories","parking"]]

sns.heatmap(housing_price_corr.corr(), annot = True)
```

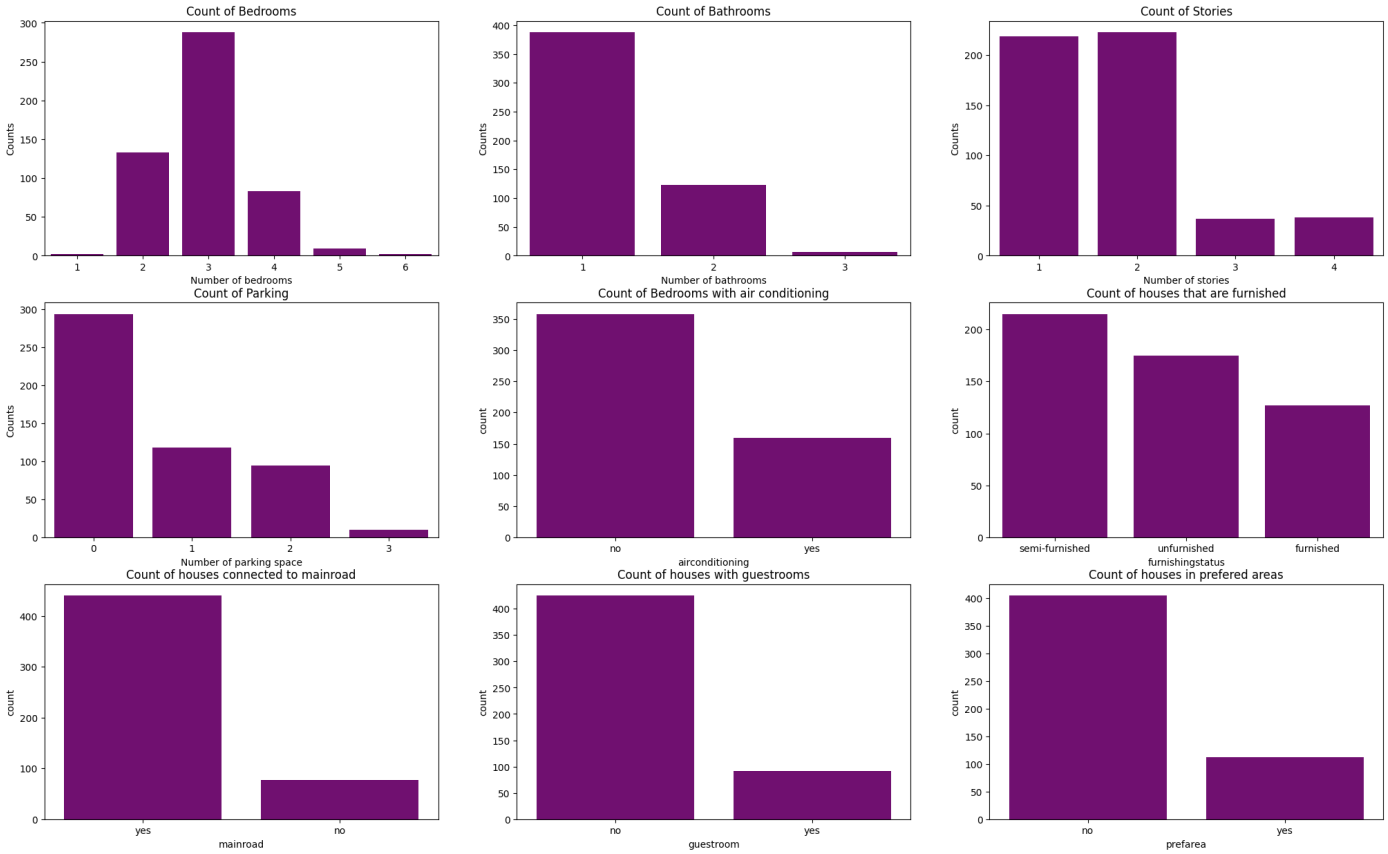
<Axes: >




Insight: There is no strong correlation between the features

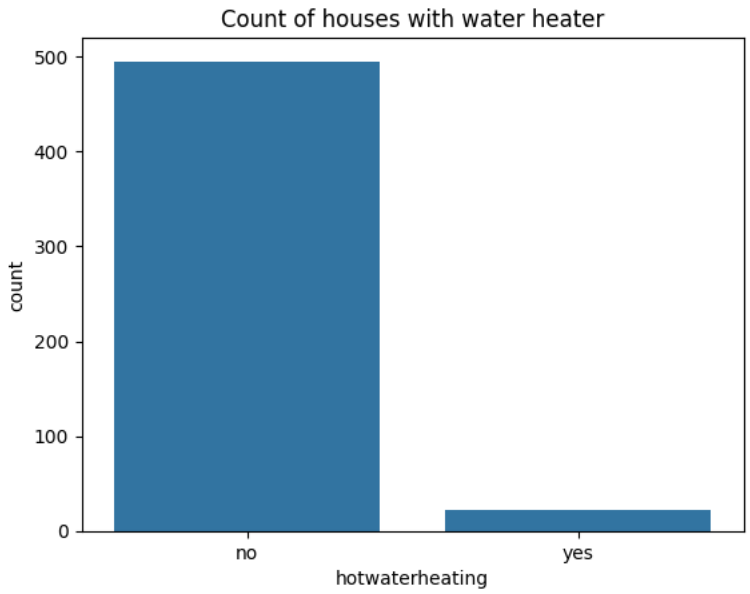
## Counts of each of the descrete values visualized

```
plt.figure(figsize=(25,15))

plt.subplot(3, 3, 1)
sns.countplot(housing_price, x= housing_price['bedrooms'])
plt.title('Count of Bedrooms')
plt.xlabel('Number of bedrooms')
plt.ylabel('Counts')

plt.subplot(3, 3, 2)
sns.countplot(housing_price, x= housing_price['bathrooms'])
plt.title('Count of Bathrooms')
plt.xlabel('Number of bathrooms')
plt.ylabel('Counts')


plt.subplot(3, 3, 3)
sns.countplot(housing_price, x= housing_price['stories'])
plt.title('Count of Stories')
plt.xlabel('Number of stories')
plt.ylabel('Counts')


plt.subplot(3, 3, 4)
sns.countplot(housing_price, x= housing_price['parking'])
plt.title('Count of Parking')
plt.xlabel('Number of parking space')
plt.ylabel('Counts')


plt.subplot(3, 3, 5)
sns.countplot(housing_price, x= housing_price['airconditioning'])
plt.title('Count of Bedrooms with air conditioning')


plt.subplot(3, 3, 6)
sns.countplot(housing_price, x= housing_price['furnishingstatus'])
plt.title('Count of houses that are furnished')


plt.subplot(3, 3, 7)
sns.countplot(housing_price, x= housing_price['mainroad'])
plt.title('Count of houses connected to mainroad')


plt.subplot(3, 3, 8)
sns.countplot(housing_price, x= housing_price['guestroom'])
plt.title('Count of houses with guestrooms')


plt.subplot(3, 3, 9)
sns.countplot(housing_price, x= housing_price['prefarea'])
plt.title('Count of houses in prefered areas')


plt.show()
```




```
sns.countplot(housing_price, x= housing_price['hotwaterheating'])
plt.title('Count of houses with water heater')
```

```
Text(0.5, 1.0, 'Count of houses with water heater')
```

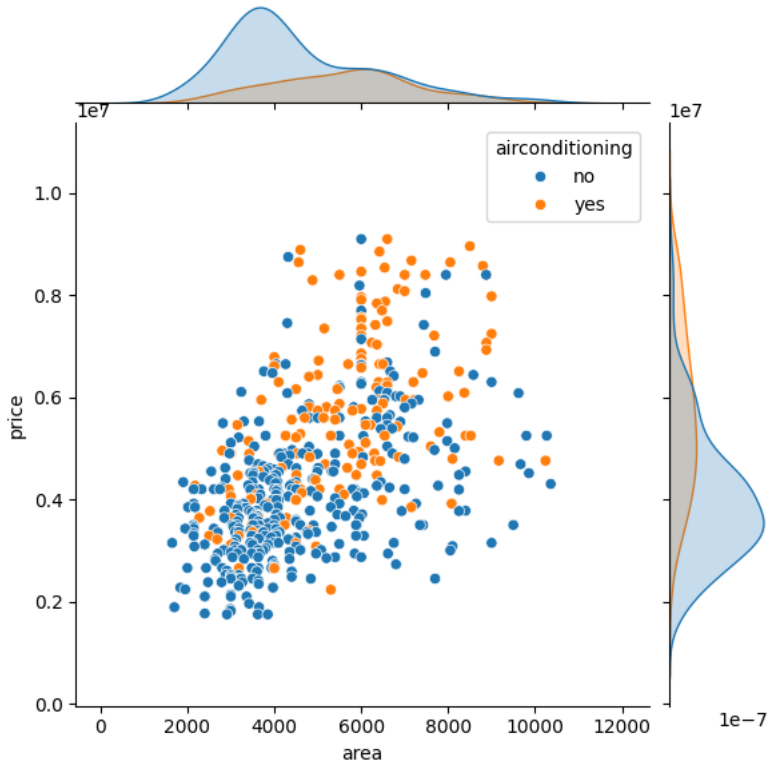



```
plt.plot('area','price', data =housing_price, linestyle ='none', marker ='o', color = "purple")
plt.xlabel('Area')
plt.ylabel("Price")
plt.title("Price vs Area")
plt.show()
```


insight: there is a strong relationship between price and area. This shows that the larger the property area, the higher the price

```
sns.jointplot(data=housing_price, x="area", y="price", hue="airconditioning")
```

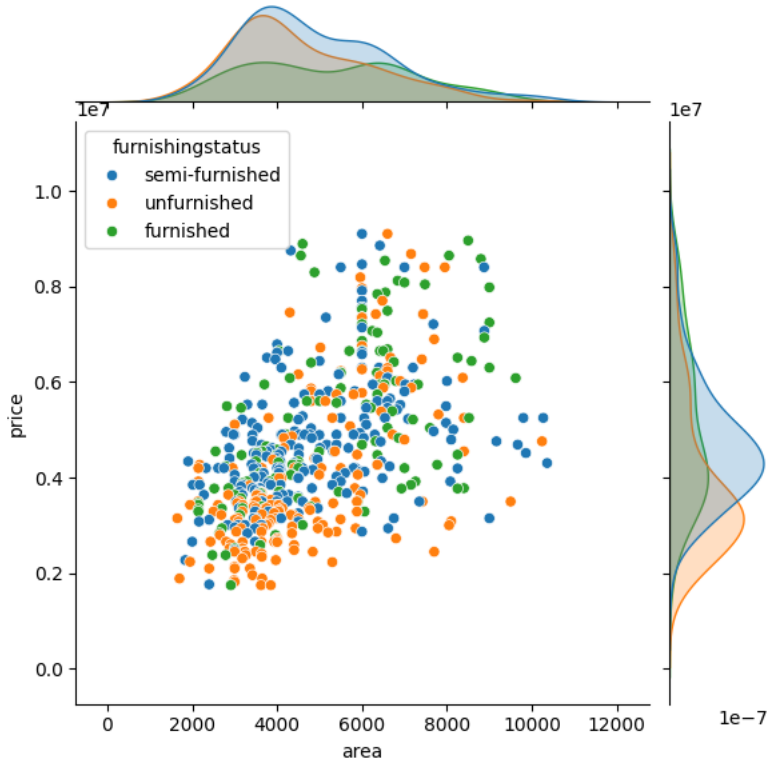
```
<seaborn.axisgrid.JointGrid at 0x78bc32e7e1d0>
```




insight: Houses with air conditioning have higher prices compared to houses with no air conditioning.

```
sns.jointplot(data=housing_price, x="area", y="price", hue="furnishingstatus")
```

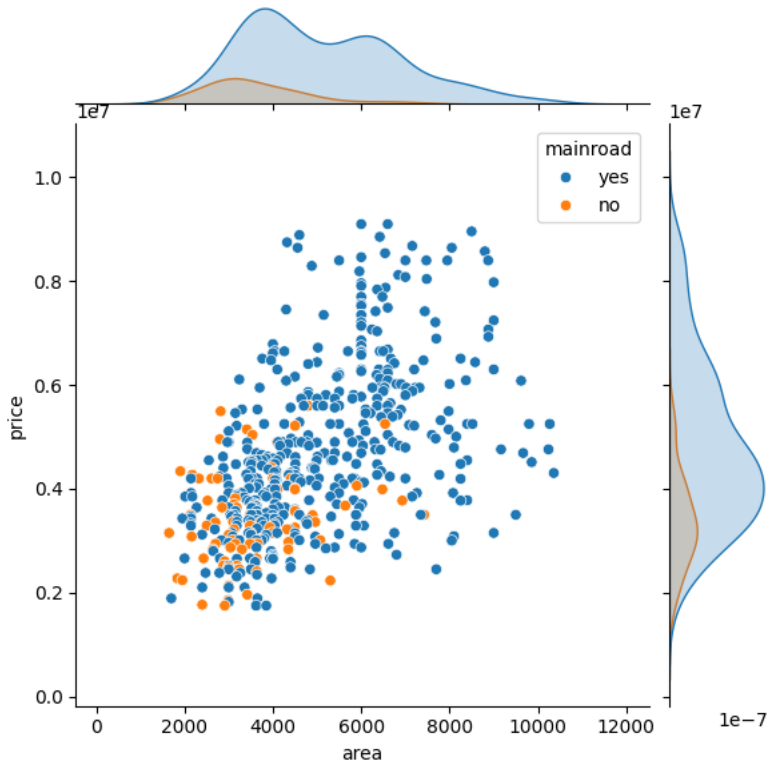
```
<seaborn.axisgrid.JointGrid at 0x78bc72e596c0>
```




insight: Houses that are not furnished are cheaper compared to houses that are semi-furnished or fully furnished.

```
sns.jointplot(data=housing_price, x="area", y="price", hue="mainroad")
```

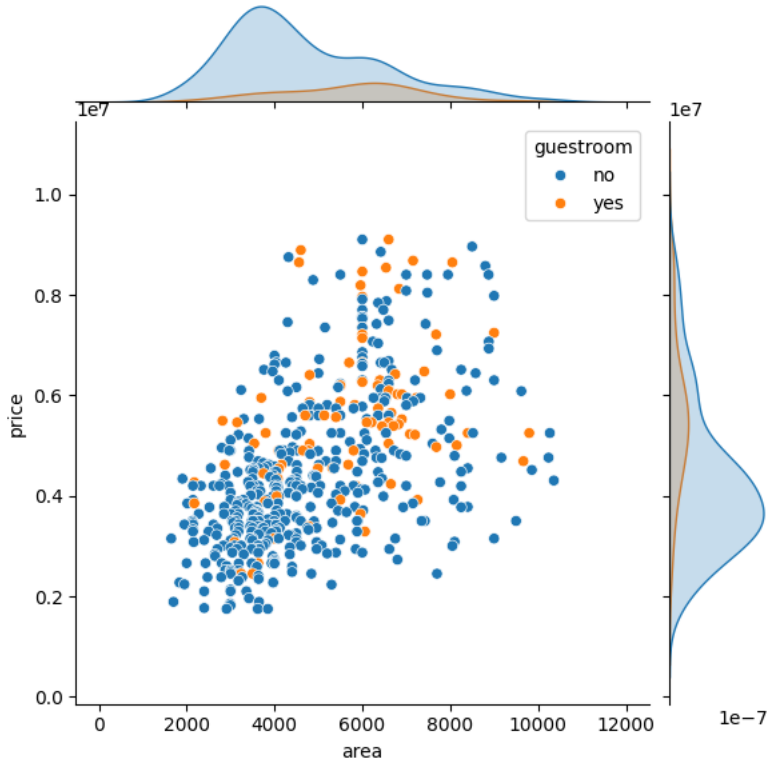
```
<seaborn.axisgrid.JointGrid at 0x78bc2f7426b0>
```




insight: Houses connected to main roads are more expensive compared to houses that are not connected to main roads.

```
sns.jointplot(data=housing_price, x="area", y="price", hue="guestroom")
```

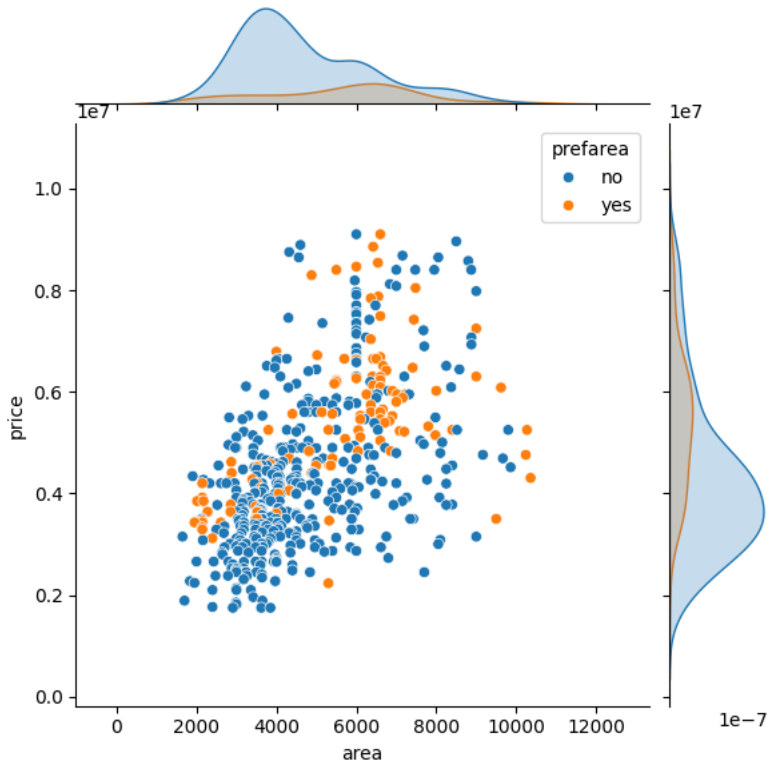
```
<seaborn.axisgrid.JointGrid at 0x78bc32cd15d0>
```




insight: Houses with no guestrooms are cheaper compared to houses with guestrooms

```
sns.jointplot(data=housing_price, x="area", y="price", hue="prefarea")
```

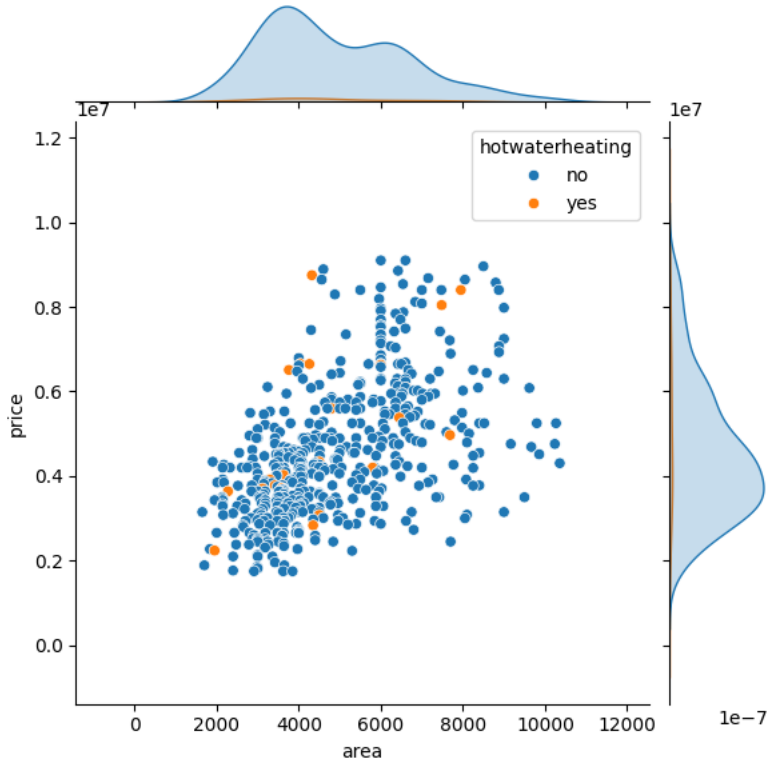
```
<seaborn.axisgrid.JointGrid at 0x78bc332a6f50>
```




insight: Houses in the prefered areas are more expensive than houses not in the prefered area

```
sns.jointplot(data=housing_price, x="area", y="price", hue="hotwaterheating")
```

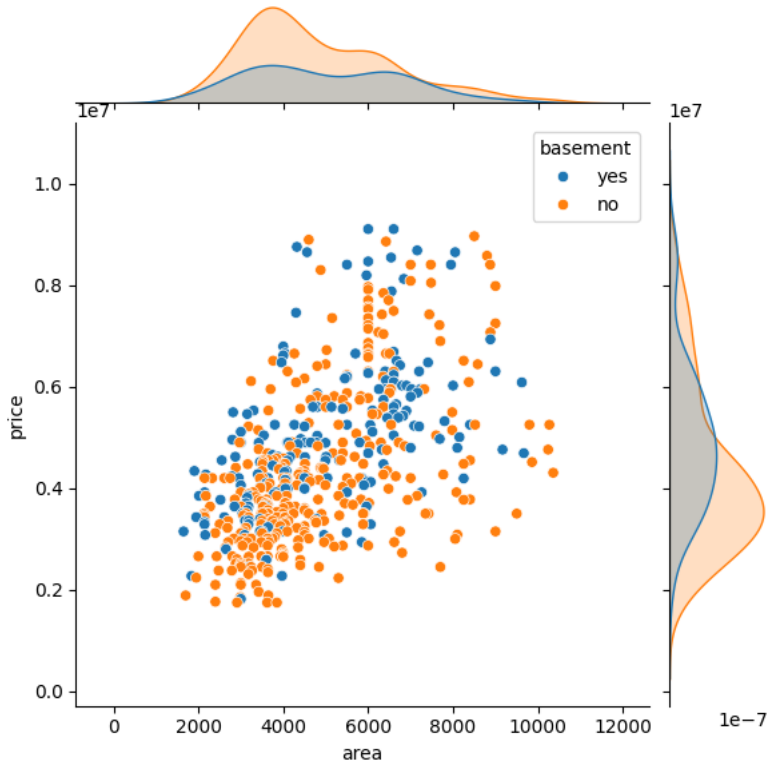
```
<seaborn.axisgrid.JointGrid at 0x78bc2fc8b820>
```




insight: majority of the houses do not have hot water heating

```
sns.jointplot(data=housing_price, x="area", y="price", hue="basement")
```

```
<seaborn.axisgrid.JointGrid at 0x78bc356df880>
```



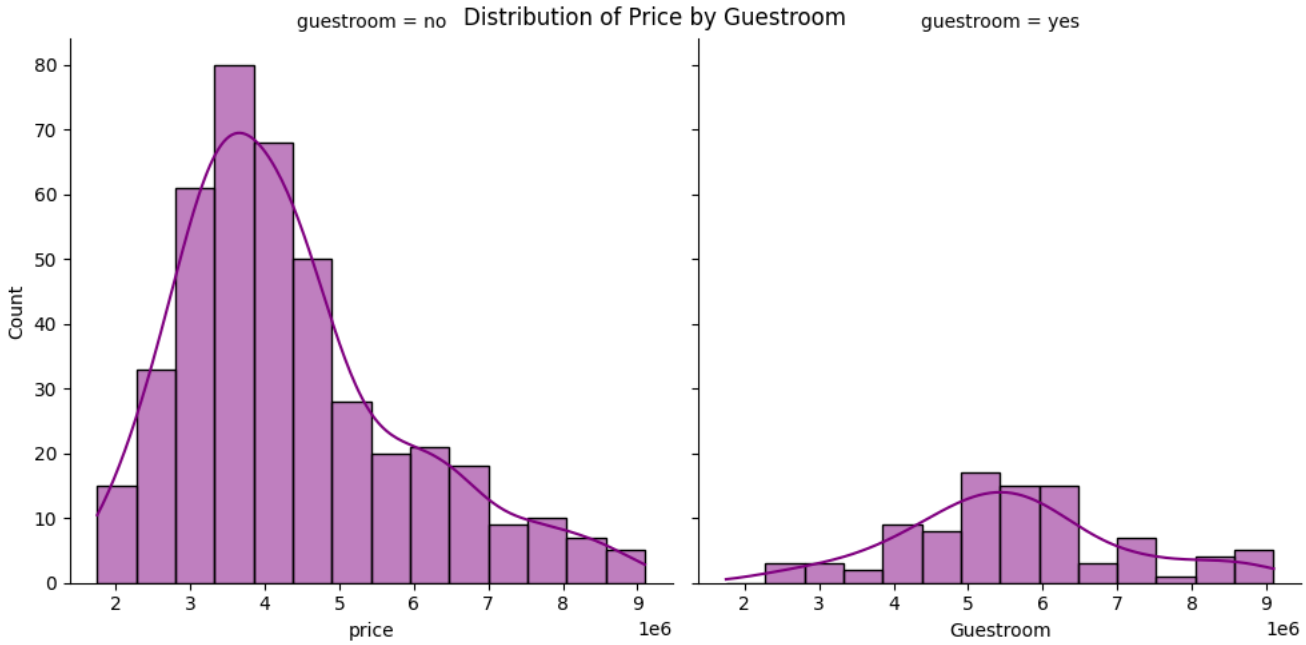

insight: majority of the houses with no basement are cheaper compared to houses with basements

```
plt.hist(housing_price["price"], color = "purple")
plt.xlabel("Price")
plt.ylabel("Count")
plt.title("Distribution of Price")
```

```
Text(0.5, 1.0, 'Distribution of Price')
```

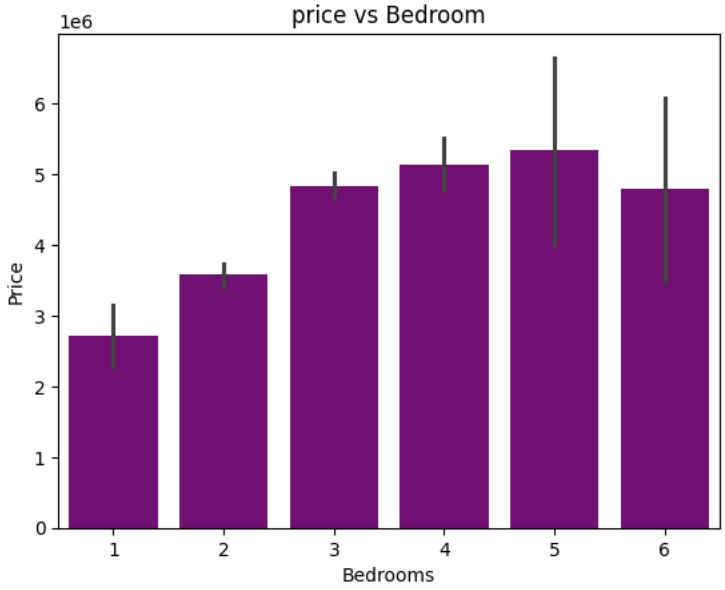

The price of most of the houses fall between 3.5 million to 4.5 million

```
sns.displot(data = housing_price, x="price", col ="guestroom", color = "purple", kde = True)
plt.xlabel("Guestroom")
plt.suptitle("Distribution of Price by Guestroom")
plt.show()
```




Insight: The house price of majority of the houses with guestrooms fall between 3 million and 5 million while the price of the houses with no guest room fall between 5 million and 6 million. This shows that houses with guest room are more expensive than houses with no guest room

```
sns.barplot(x ='bedrooms', y="price", data =housing_price, color = "purple")
plt.xlabel('Bedrooms')
plt.ylabel("Price")
plt.title("price vs Bedroom")
plt.show()
```




Houses with 5 bedrooms have the highest price, followed by houses with 4 bedrooms. Houses with 1 bedroom have the least price

```
sns.barplot(x='bathrooms', y='price', color = "purple", data=housing_price)
plt.xlabel('bathrooms')
plt.ylabel('Price')
plt.title('Price Vs Number of bathrooms')
```

```
Text(0.5, 1.0, 'Price Vs Number of bathrooms')
```