In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Load Dataset

In [23]:
```
df = pd.read_csv('House Price India.csv')
df.head()
```

Out[23]:

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condit of ho
0	6762810145	42491	5	2.50	3650	9050	2.0	0	4	
1	6762810635	42491	4	2.50	2920	4000	1.5	0	0	
2	6762810998	42491	5	2.75	2910	9480	1.5	0	0	
3	6762812605	42491	4	2.50	3310	42998	2.0	0	0	
4	6762812919	42491	3	2.00	2710	4500	1.5	0	0	

5 rows × 23 columns

In [3]:
```
df.info()
```

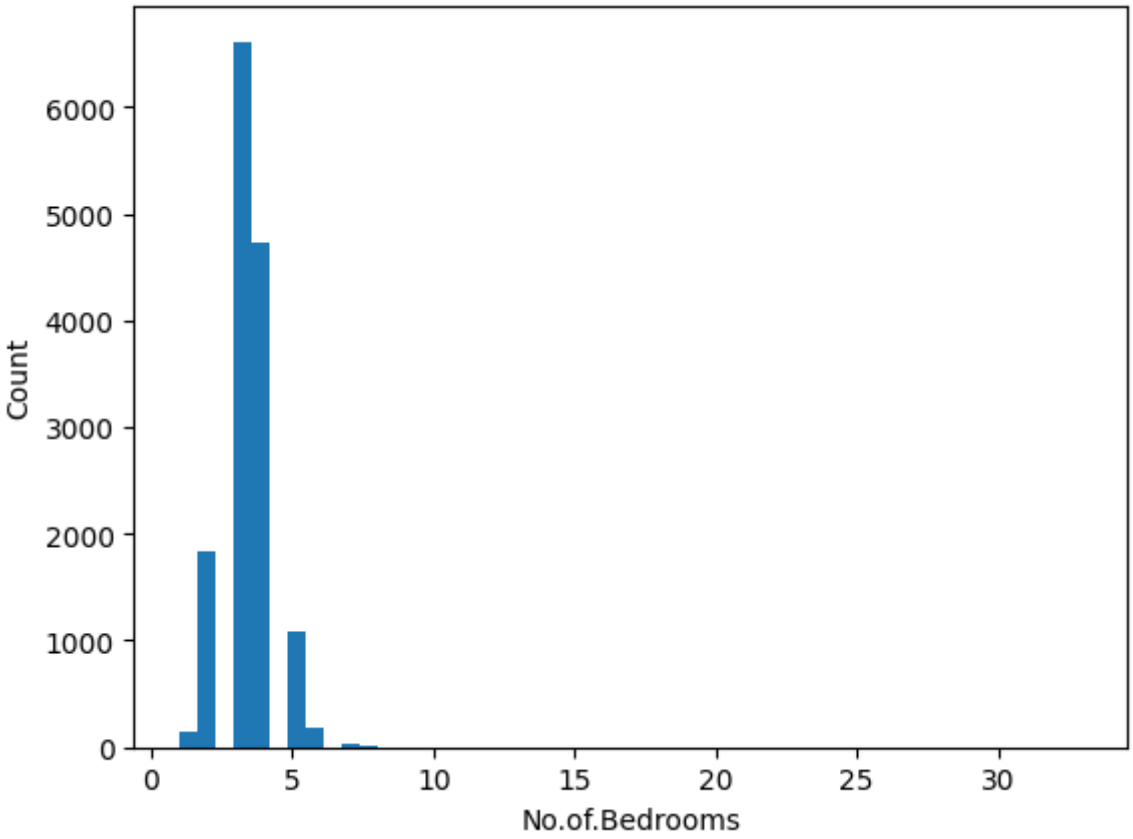
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14620 entries, 0 to 14619
Data columns (total 23 columns):
 #   Column                                 Non-Null Count  Dtype  
---  ------                                 --------------  -----  
 0   id                                     14620 non-null  int64  
 1   Date                                   14620 non-null  int64  
 2   number of bedrooms                     14620 non-null  int64  
 3   number of bathrooms                    14620 non-null  float64
 4   living area                            14620 non-null  int64  
 5   lot area                               14620 non-null  int64  
 6   number of floors                       14620 non-null  float64
 7   waterfront present                     14620 non-null  int64  
 8   number of views                        14620 non-null  int64  
 9   condition of the house                 14620 non-null  int64  
 10  grade of the house                     14620 non-null  int64  
 11  Area of the house(excluding basement)  14620 non-null  int64  
 12  Area of the basement                   14620 non-null  int64  
 13  Built Year                             14620 non-null  int64  
 14  Renovation Year                        14620 non-null  int64  
 15  Postal Code                            14620 non-null  int64  
 16  Lattitude                              14620 non-null  float64
 17  Longitude                              14620 non-null  float64
 18  living_area_renov                      14620 non-null  int64  
 19  lot_area_renov                         14620 non-null  int64  
 20  Number of schools nearby               14620 non-null  int64  
 21  Distance from the airport              14620 non-null  int64  
 22  Price                                  14620 non-null  int64  
dtypes: float64(4), int64(19)
memory usage: 2.6 MB
```

Univariate Analysis

Histogram

In [10]:
```
plt.hist(df['number of bedrooms'],bins=50)
plt.xlabel("No.of.Bedrooms")
plt.ylabel("Count")
```

Out[10]:
```
Text(0, 0.5, 'Count')
```



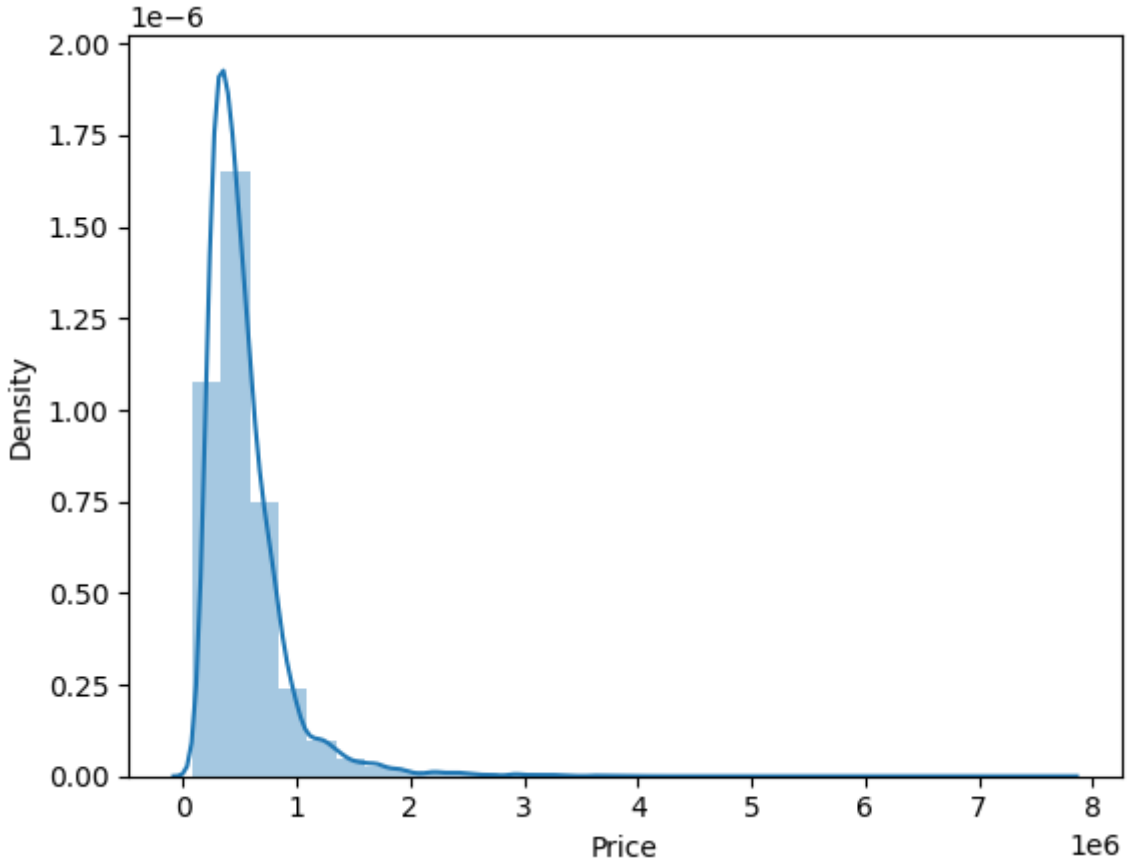
From the above graph we can clearly see that the peek count above 6000 is at range between 0 to 5. As the no.of.bedrooms increases after 5 the count values decreases tremoundously.

Distplot

In [11]:
```
sns.distplot(df['Price'],bins=30)
```

```
C:\Users\ELCOT\AppData\Local\Temp\ipykernel_70496\507312228.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Price'],bins=30)
```

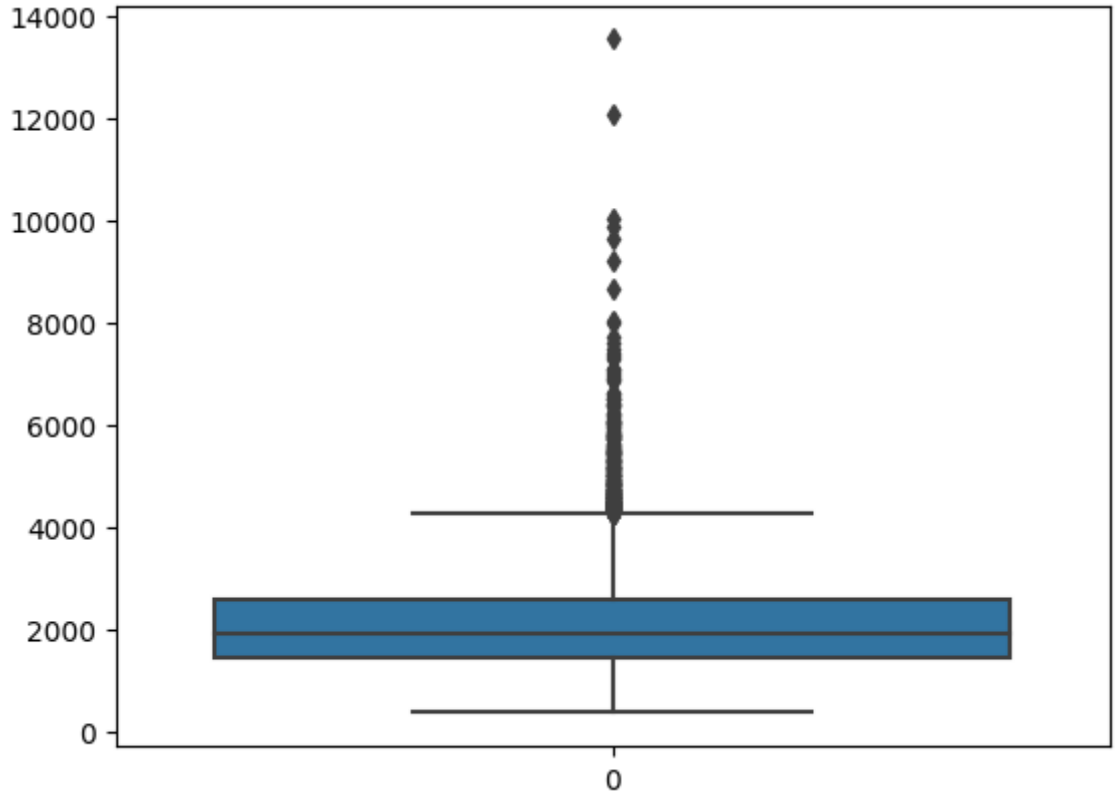
Out[11]:
```
<AxesSubplot: xlabel='Price', ylabel='Density'>
```



Boxplot

In [12]:
```
sns.boxplot(df['living area'])
```

Out[12]: <AxesSubplot: >



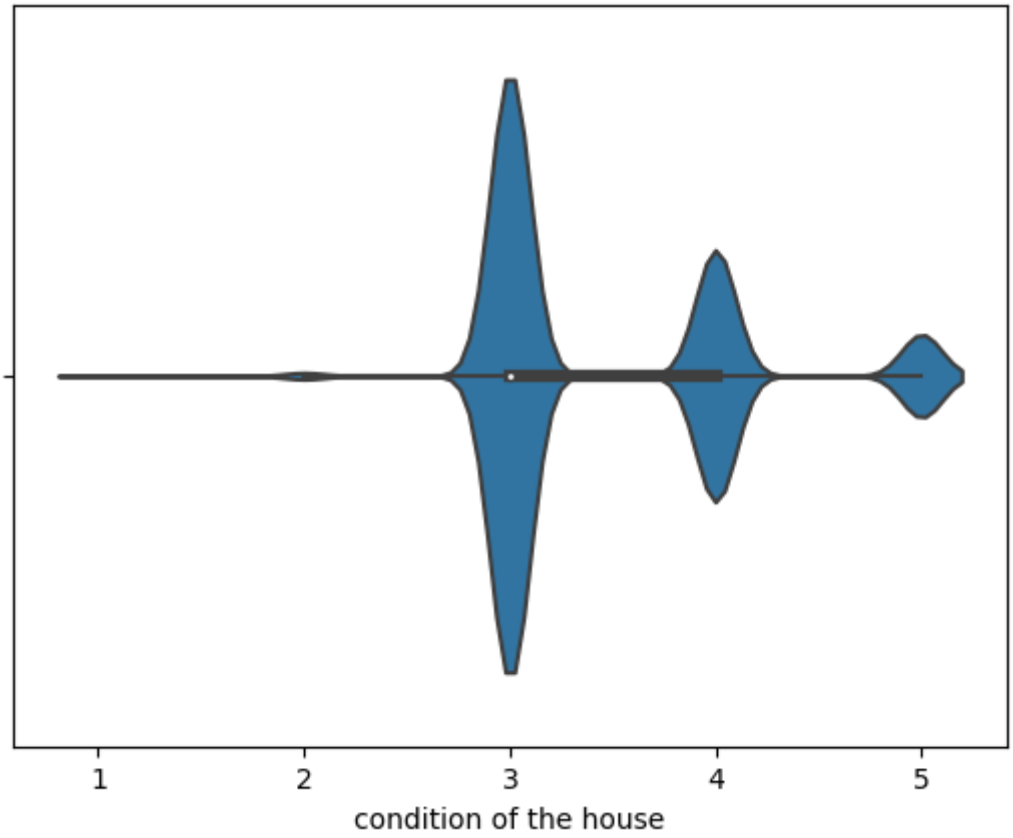
Boxplot is also used for detect the outlier in data set. It captures the summary of the data efficiently with a simple box and whiskers and allows

us to compare easily across groups. Boxplot for living area and it contains many outliers and many outliers present in the features. The above one is a sample for detecting outliers.

Violinplot

In [13]:
```
sns.violinplot(x=df['condition of the house'])
```

Out[13]:
```
<AxesSubplot: xlabel='condition of the house'>
```



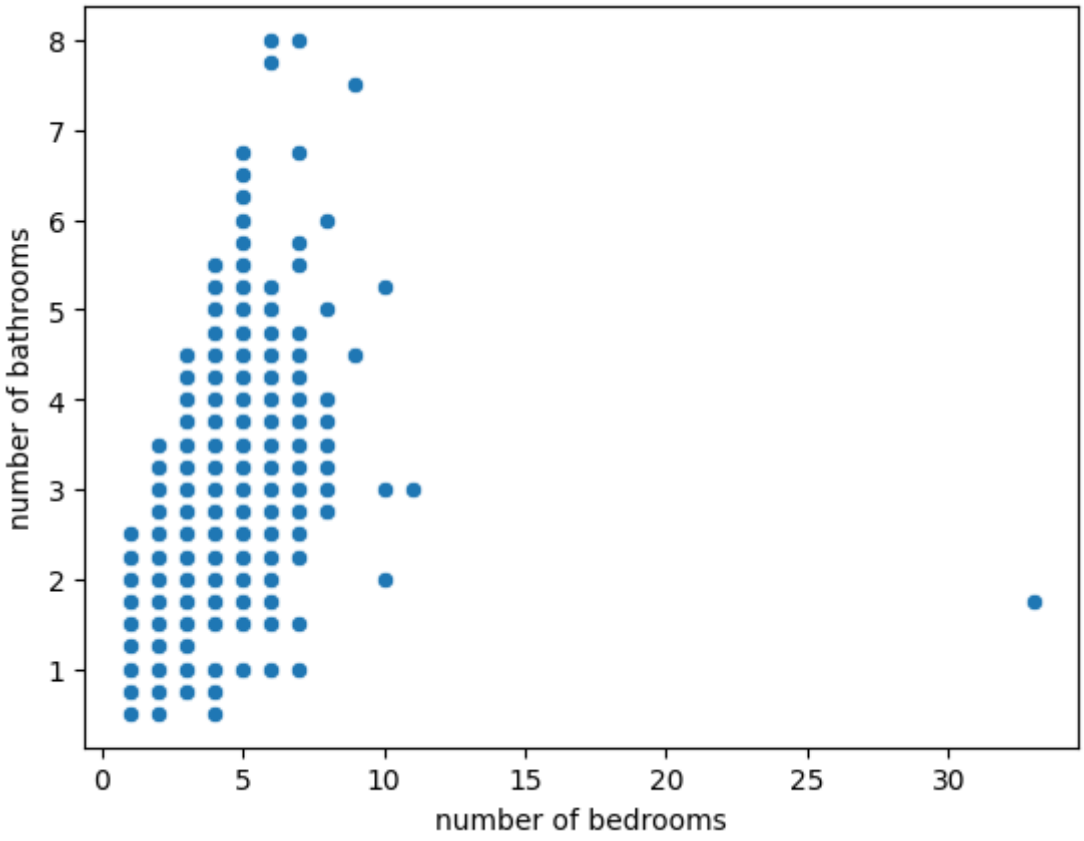
violinplot is used to vizualize the distribution numerical data and it shows the full distribution of data. The mean value of the variable "condition of the house" lies in 3 and the interquartile ranges between 3 to 4. The rest thin lines represents the rest distributions, except for the points that are determined to be the outliers. The higher probability lies in 3 and lowest probability lies above 5.

Bivariate Analysis

Scatterplot

In [14]:
```
sns.scatterplot(x=df['number of bedrooms'],y=df['number of bathrooms'])
```

Out[14]:
```
<AxesSubplot: xlabel='number of bedrooms', ylabel='number of bathrooms'>
```

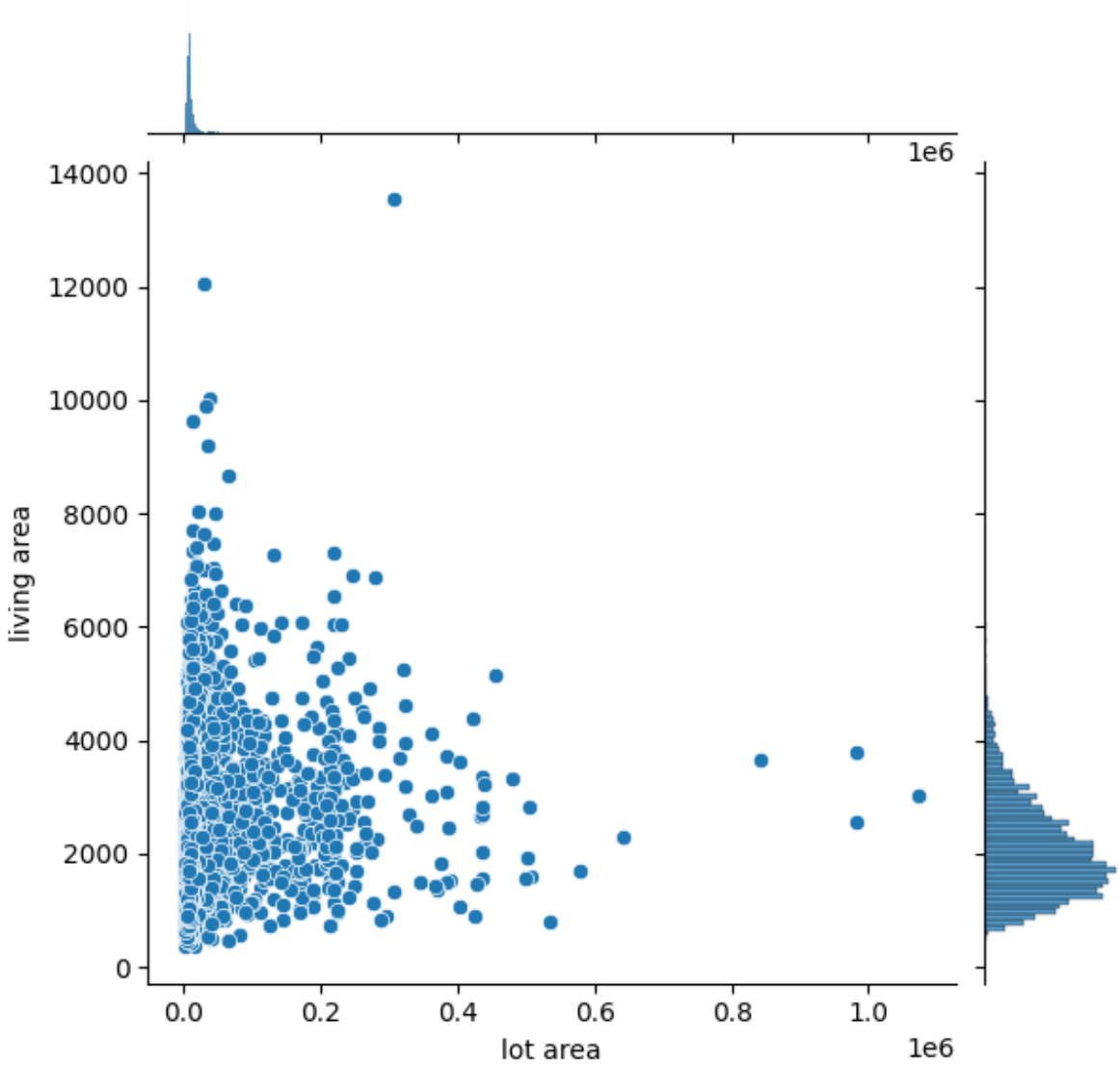


he scatterplot is used to show distributions between two variables. For no.of.bathrooms and no.of.bedrooms as far as the bathroom increases the bedroom number increases. And there are some outliers present in them.

Jointplot

In [15]:
```
sns.jointplot(data = df,x = 'lot area',y = 'living area')
```

Out[15]:
```
<seaborn.axisgrid.JointGrid at 0x21867173ca0>
```

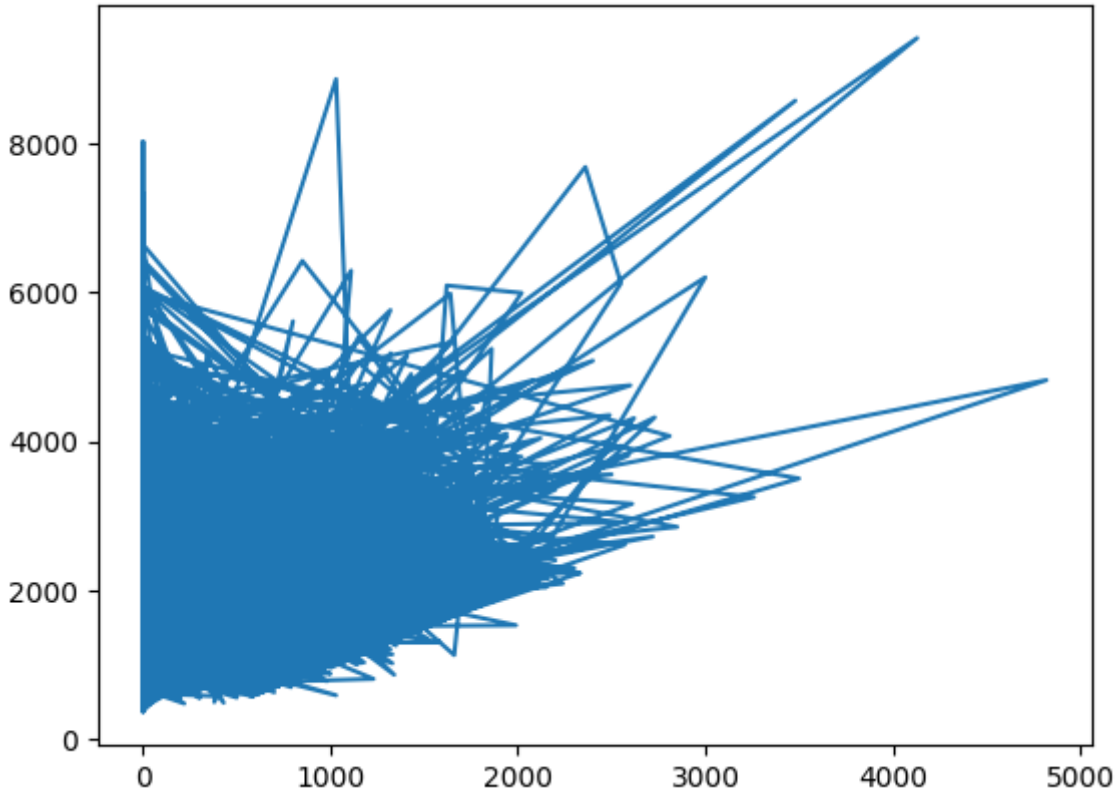


The relation between living area vs lot area and univariate of these has been shown. As far as the living area increases the lot area increases slighter and present many outliers between them. Univariate distribution of lot area remains same with slight increase in area but for living area the peak value is achieved at 2000 by gradual increase in it and then decreases until at a range of 5000.

Line plot

In [16]:
```
plt.plot(df['Area of the basement'],df['Area of the house(excluding basement)'])
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x218699cf2e0>]
```

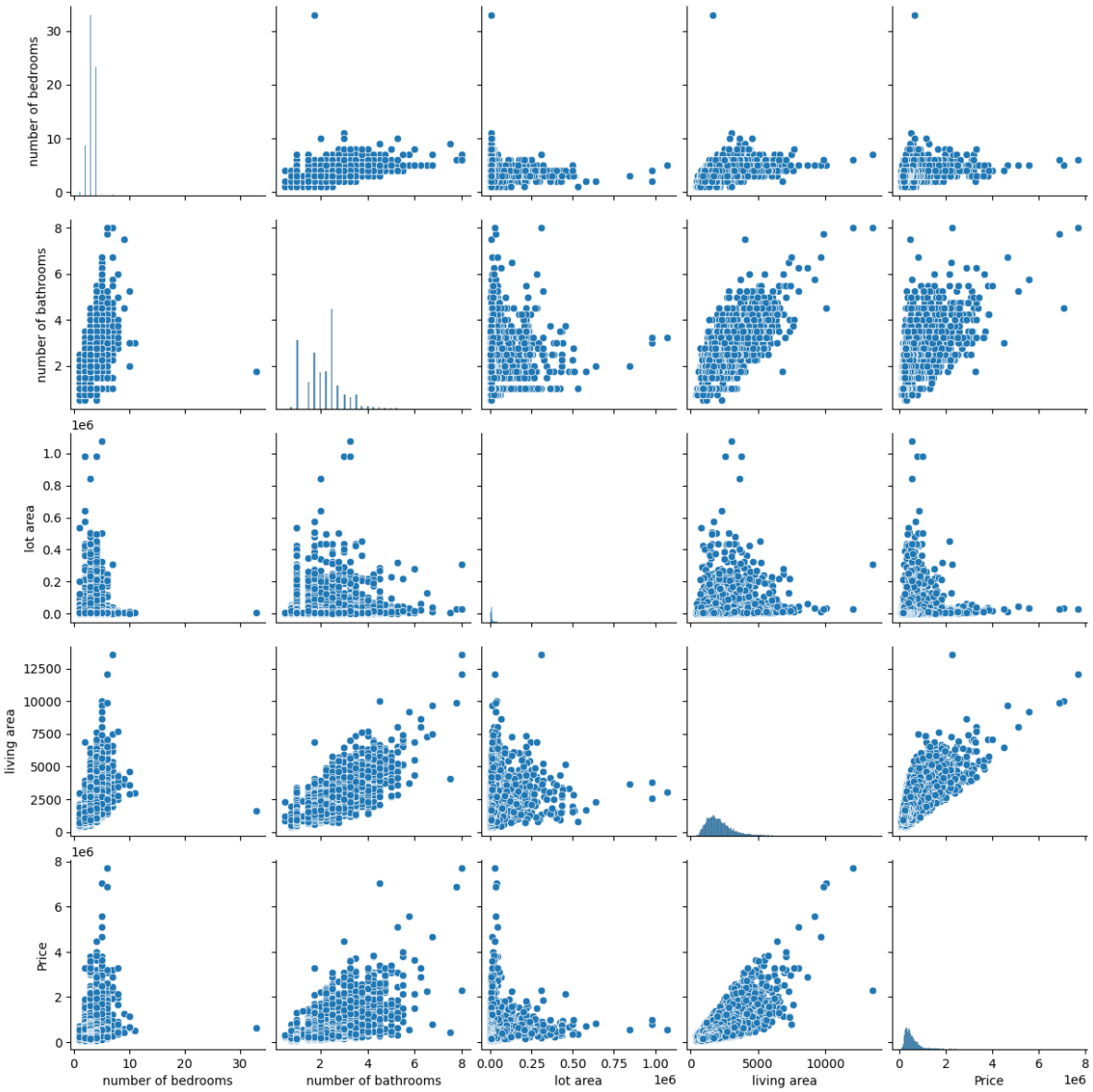


Multivariate Analysis

Pair plot

In [17]:
```
X = df[['number of bedrooms','number of bathrooms','lot area','living area','Price

sns.pairplot(X,dropna=True)
```

Out[17]:
```
<seaborn.axisgrid.PairGrid at 0x218672039a0>
```

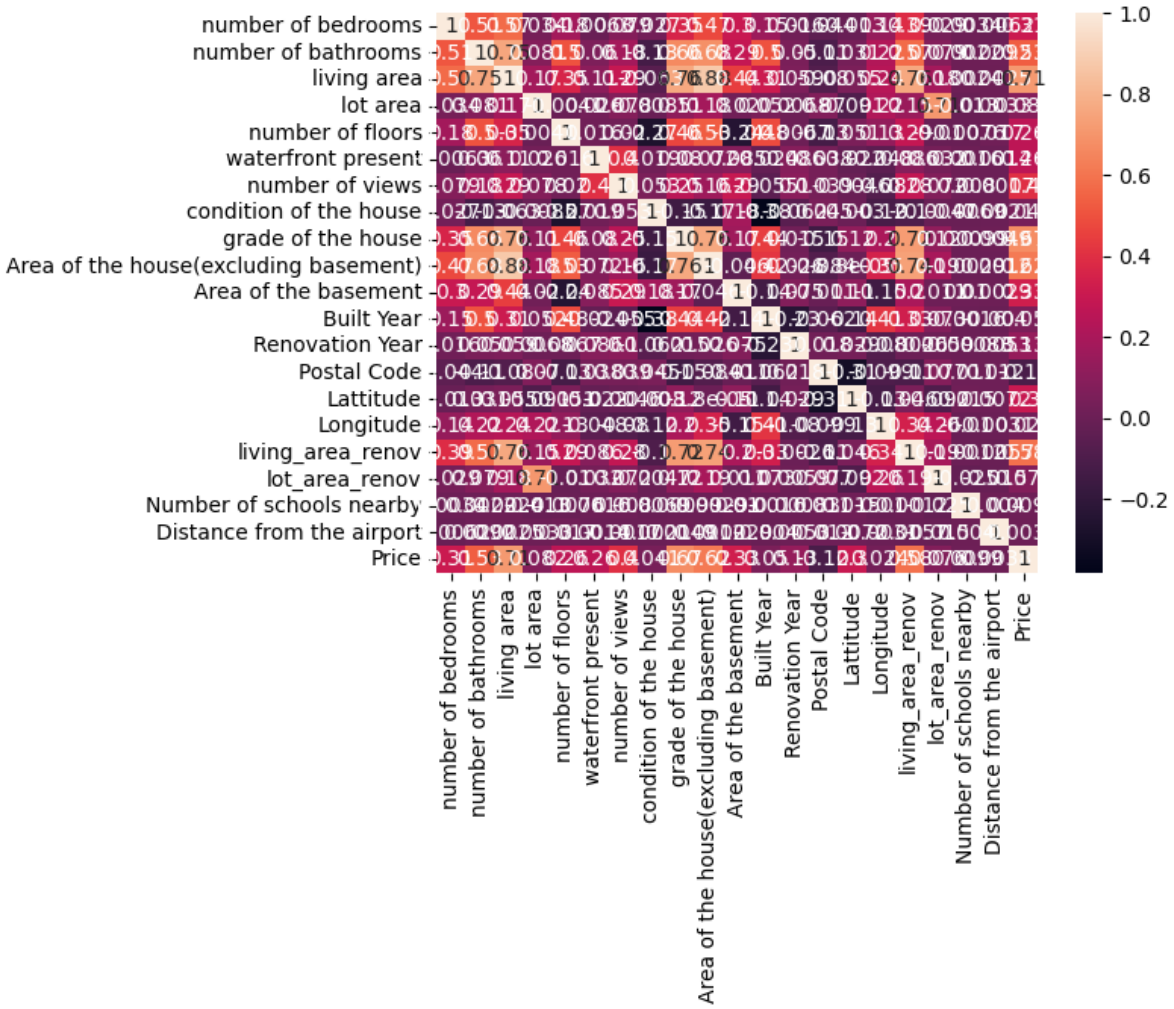


From pairplot we can clearly see that some variable are linear to some variable and logistic to some variables. Most of the variables are linear to other variables. But in all variables outliers present in it.

Heat map

In [18]:
```
df.drop(columns=['id','Date'],inplace=True)
sns.heatmap(df.corr(),annot=True)
```

Out[18]: <AxesSubplot: >



Statistical analysis

In [5]:
```
df.describe()
```

Out[5]:

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	nu
count	1.462000e+04	14620.000000	14620.000000	14620.000000	14620.000000	1.462000e+04	1462
mean	6.762821e+09	42604.538646	3.379343	2.129583	2098.262996	1.509328e+04	
std	6.237575e+03	67.347991	0.938719	0.769934	928.275721	3.791962e+04	
min	6.762810e+09	42491.000000	1.000000	0.500000	370.000000	5.200000e+02	
25%	6.762815e+09	42546.000000	3.000000	1.750000	1440.000000	5.010750e+03	
50%	6.762821e+09	42600.000000	3.000000	2.250000	1930.000000	7.620000e+03	
75%	6.762826e+09	42662.000000	4.000000	2.500000	2570.000000	1.080000e+04	
max	6.762832e+09	42734.000000	33.000000	8.000000	13540.000000	1.074218e+06	

8 rows × 23 columns

Descriptive statistics to summerize the data by computing mean, median, mode, standard derivation and likewise other informations of data.

Handling missing values

In [21]:
```
df.isnull().any()
```

Out[21]:
```
number of bedrooms                        False
number of bathrooms                       False
living area                               False
lot area                                  False
number of floors                          False
waterfront present                        False
number of views                           False
condition of the house                    False
grade of the house                        False
Area of the house(excluding basement)     False
Area of the basement                      False
Built Year                                False
Renovation Year                           False
Postal Code                               False
Lattitude                                 False
Longitude                                 False
living_area_renov                         False
lot_area_renov                            False
Number of schools nearby                  False
Distance from the airport                 False
Price                                     False
dtype: bool
```

In [22]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14620 entries, 0 to 14619
Data columns (total 21 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   number of bedrooms                    14620 non-null  int64
 1   number of bathrooms                   14620 non-null  float64
 2   living area                           14620 non-null  int64
 3   lot area                              14620 non-null  int64
 4   number of floors                      14620 non-null  float64
 5   waterfront present                    14620 non-null  int64
 6   number of views                       14620 non-null  int64
 7   condition of the house                14620 non-null  int64
 8   grade of the house                    14620 non-null  int64
 9   Area of the house(excluding basement) 14620 non-null  int64
 10  Area of the basement                  14620 non-null  int64
 11  Built Year                            14620 non-null  int64
 12  Renovation Year                       14620 non-null  int64
 13  Postal Code                           14620 non-null  int64
 14  Lattitude                             14620 non-null  float64
 15  Longitude                             14620 non-null  float64
 16  living_area_renov                     14620 non-null  int64
 17  lot_area_renov                        14620 non-null  int64
 18  Number of schools nearby              14620 non-null  int64
 19  Distance from the airport             14620 non-null  int64
 20  Price                                 14620 non-null  int64
dtypes: float64(4), int64(17)
memory usage: 2.3 MB
```

The above information shows that the none of the columns contains any null value in it. We don't need to perform any specific operations to handle the missing values.

In [ ]: