```
import numpy as np
import matplotlib.pyplot as pt
import pandas as pd
import seaborn as sns
df=pd.read_csv('/content/House Price India.csv')
da=pd.DataFrame(df)
```

Data frame

```
df.head()
```

| | id | Date | number of bedrooms | number of bathrooms | living area | lot area | number of floors | waterfront present | number of views |
|---|---|---|---|---|---|---|---|---|---|
| **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

```
df.shape
```

```
(14620, 23)
```

```
#describing the dataset
df.describe()
```

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

8 rows × 23 columns

```
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
```

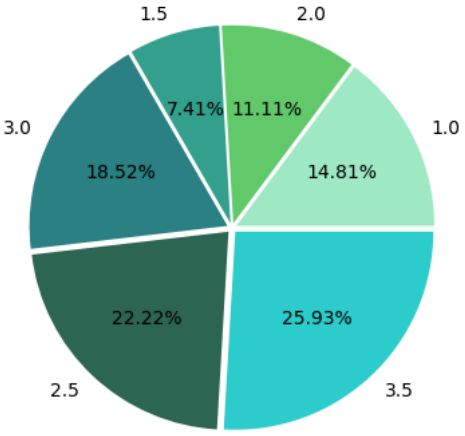
```
 18  living_area_renov                  14620 non-null  int64
 19  lot_area_renov                     14620 non-null  int64
 20  Number of schools nearby           14620 non-null  int64
 21  Distance from the airport          14620 non-null  int64
 22  Price                              14620 non-null  int64
dtypes: float64(4), int64(19)
memory usage: 2.6 MB
```

## Univariant analysis

```
p=df['number of floors'].value_counts()
p
```

```
1.0    7103
2.0    5666
1.5    1311
3.0     418
2.5     118
3.5       4
Name: number of floors, dtype: int64
```

```
pt.pie(df['number of floors'].unique(),labels=p.keys(),autopct='%0.2f%%',colors=['#9ee9c4','#61c969',
                '#339f8d','#2a8082','#2c6552','#2ccccc'],explode=[0.025,0.025,0.025,0.025,0.025,0.025])
pt.show()
```




```
p=df['condition of the house'].value_counts()
p
```

```
3    9350
4    3874
5    1278
2     100
1      18
Name: condition of the house, dtype: int64
```

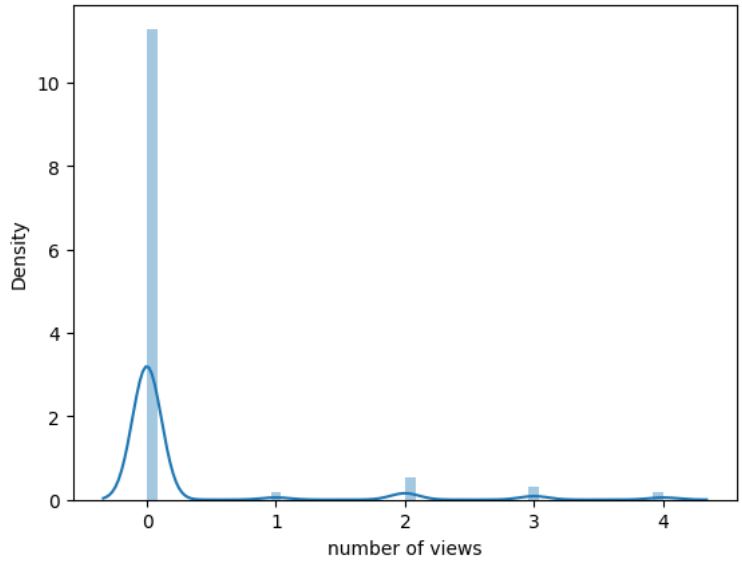
```
pt.pie(df['condition of the house'].unique(),labels=p.keys(),autopct='%0.2f%%',colors=['#f47d70','#f58c80',
                '#f79a90','#f8a5a0','#f9b7b0'],explode=[0.025,0.025,0.025,0.025,0.025])
pt.show()
```

```
p=df['number of views'].value_counts()
p
```

```
0    13198
2      636
3      351
1      219
4      216
Name: number of views, dtype: int64
```

```
sns.distplot(df['number of views'])
```

```
<ipython-input-89-edd14fddb5fd>: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['number of views'])
<Axes: xlabel='number of views', ylabel='Density'>
```




```
p=df['Renovation Year'].value_counts()
p
```

```
0       13954
2014       76
2013       30
2003       27
2005       23
          ...
1948        1
1967        1
1944        1
1959        1
1962        1
Name: Renovation Year, Length: 68, dtype: int64
```

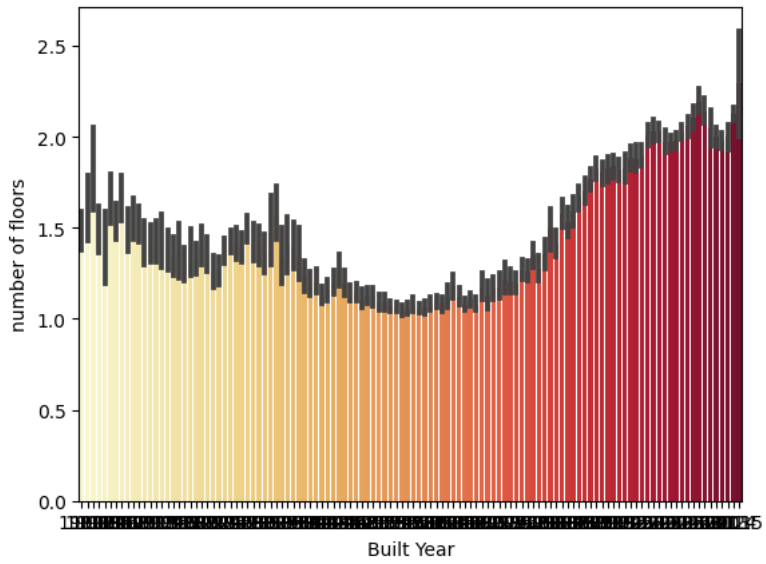
```
sns.distplot(df['Renovation Year'],color="#9e9ee8")
```

```
<ipython-input-96-c102ebe1bcde>: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['Renovation Year'],color="#9e9ee8",rug=True)
<Axes: xlabel='Renovation Year', ylabel='Density'>
```


Bivariat Analysis

```
sns.barplot(x='Built Year',y='number of floors',data=df,palette='YlOrRd',)
```

```
<Axes: xlabel='Built Year', ylabel='number of floors'>
```

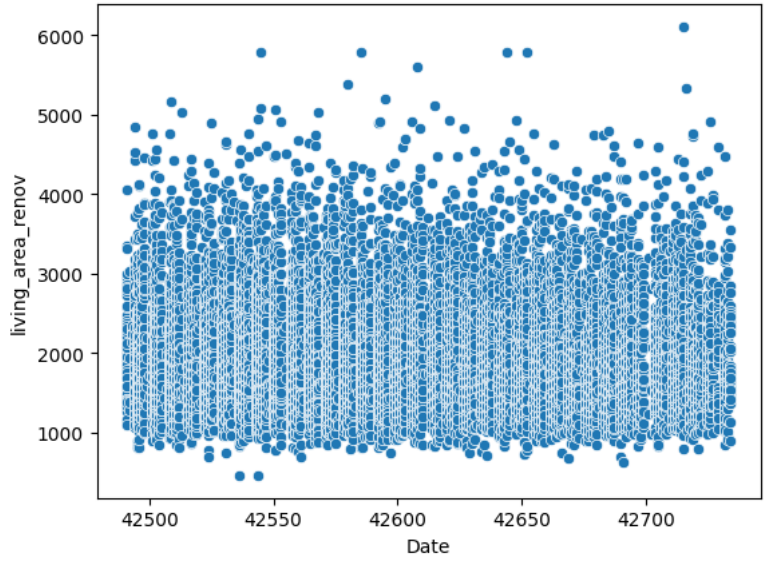



```
pt.figure(figsize=(8,8))
sns.barplot(x='Postal Code',y='Number of schools nearby',data=df,palette='Greys',hue='waterfront present')
```

```
<Axes: xlabel='Postal Code', ylabel='Number of schools nearby'>
```

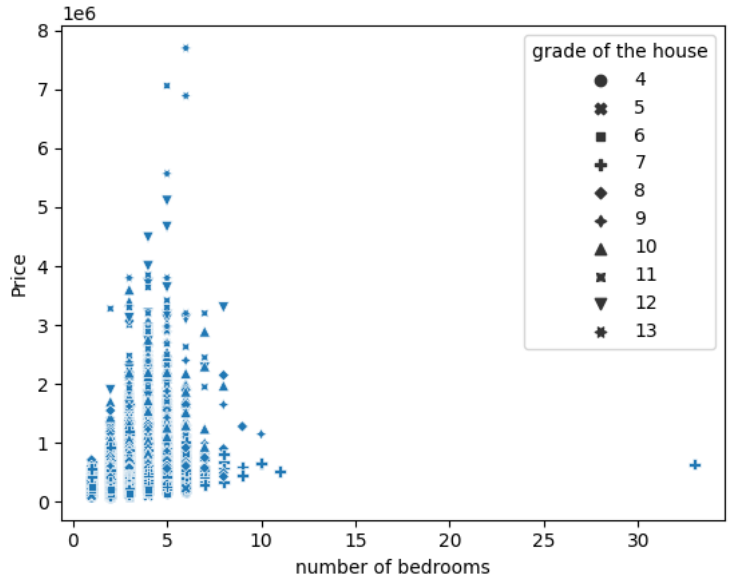

```
sns.scatterplot(x='Date',y='living_area_renov',data=df)
```

```
<Axes: xlabel='Date', ylabel='living_area_renov'>
```



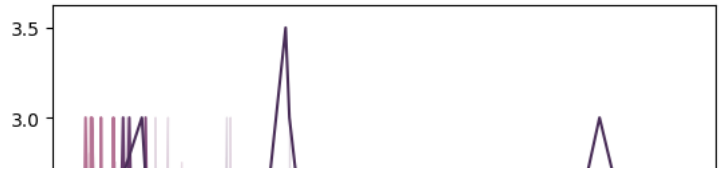

```
sns.scatterplot(x='number of bedrooms',y='Price',data=df,style='grade of the house',)
```

```
<Axes: xlabel='number of bedrooms', ylabel='Price'>
```



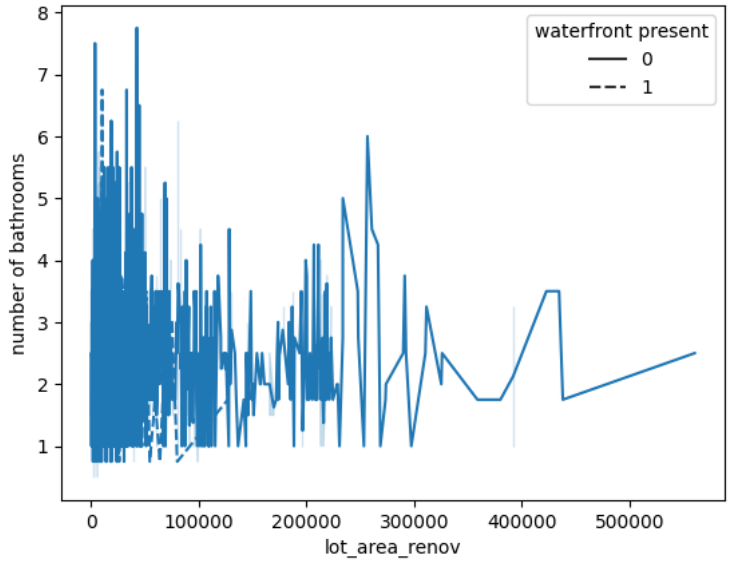

```
sns.lineplot(x='Area of the basement',y='number of floors',data=df,hue='grade of the house')
```

```
<Axes: xlabel='Area of the basement', ylabel='number of floors'>
```




```
sns.lineplot(x='lot_area_renov',y='number of bathrooms',data=df,style='waterfront present')
```

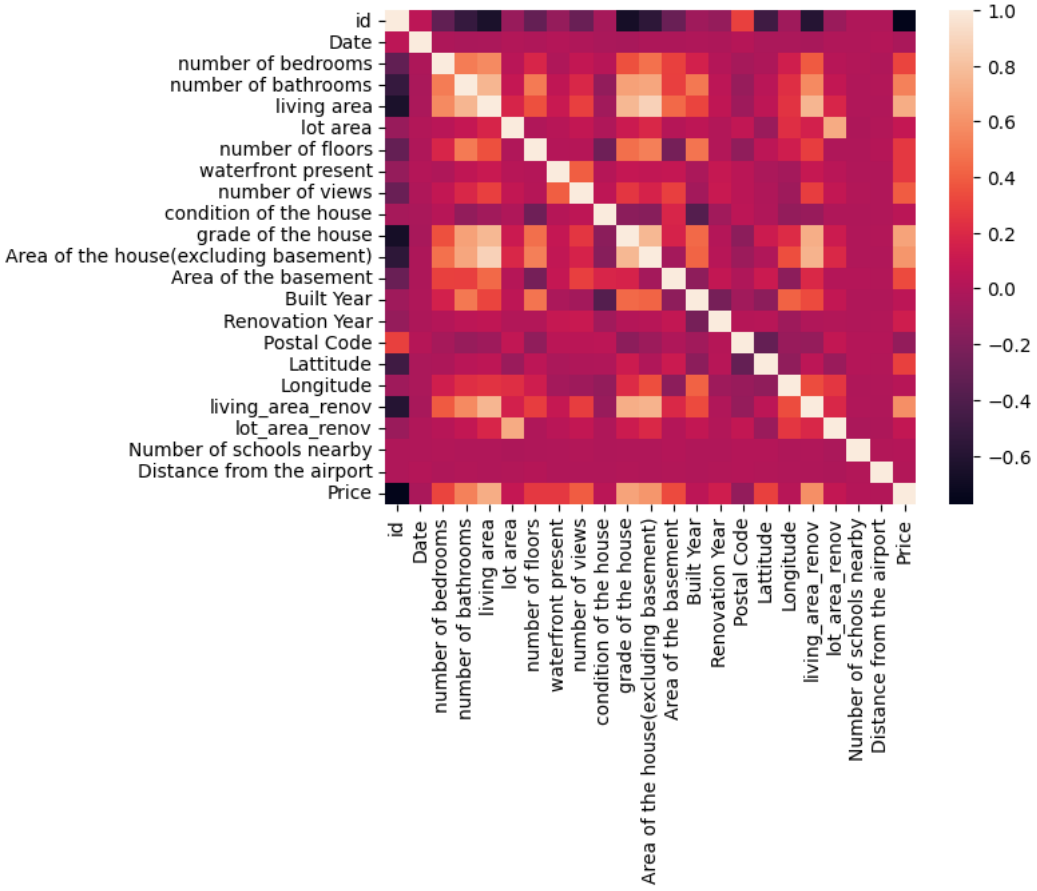
```
<Axes: xlabel='lot_area_renov', ylabel='number of bathrooms'>
```




Multivariant Analysis

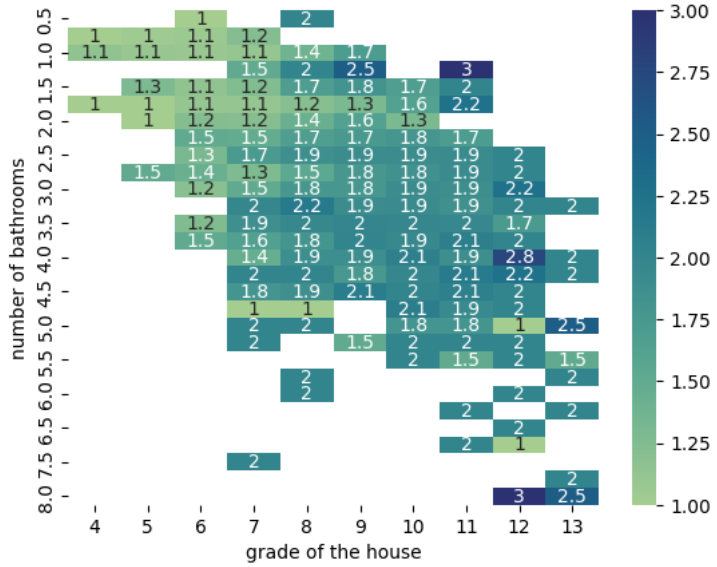
```
sns.heatmap(df.corr())
```

```
<Axes: >
```



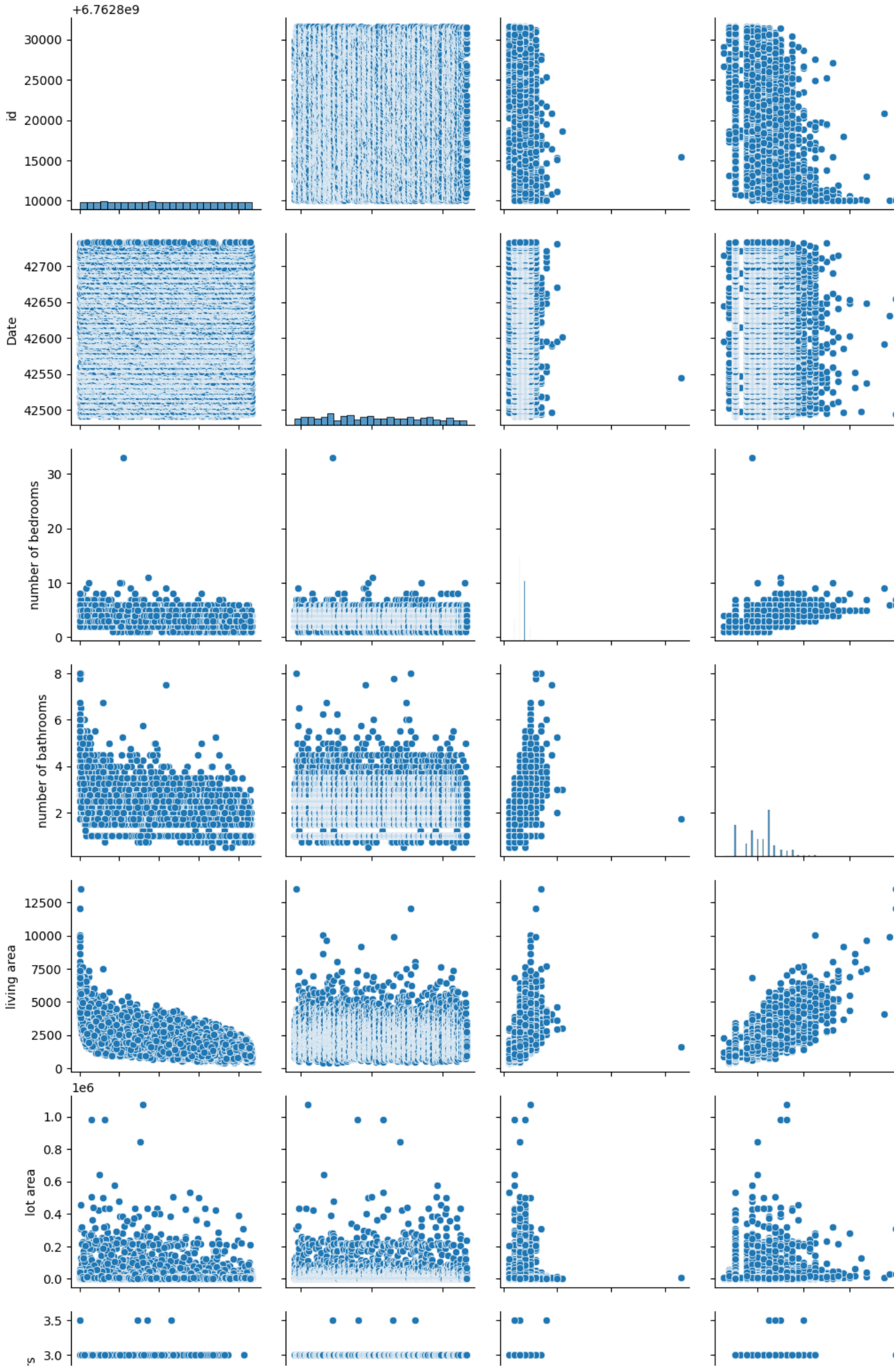

```
a=df.pivot_table(index="number of bathrooms",values="number of floors",columns="grade of the house")
sns.heatmap(a,cmap='crest',annot=True)
```

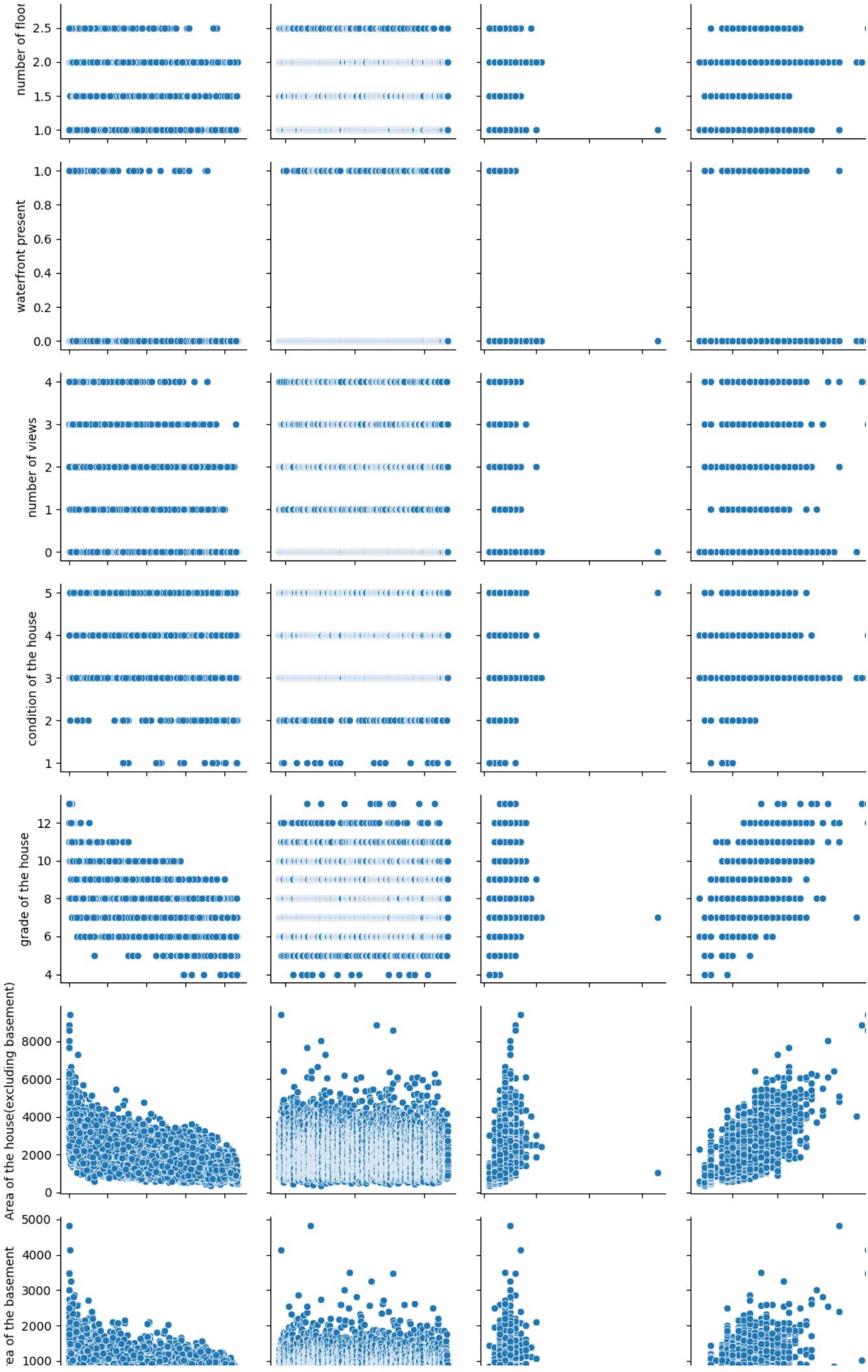
```
<Axes: xlabel='grade of the house', ylabel='number of bathrooms'>
```

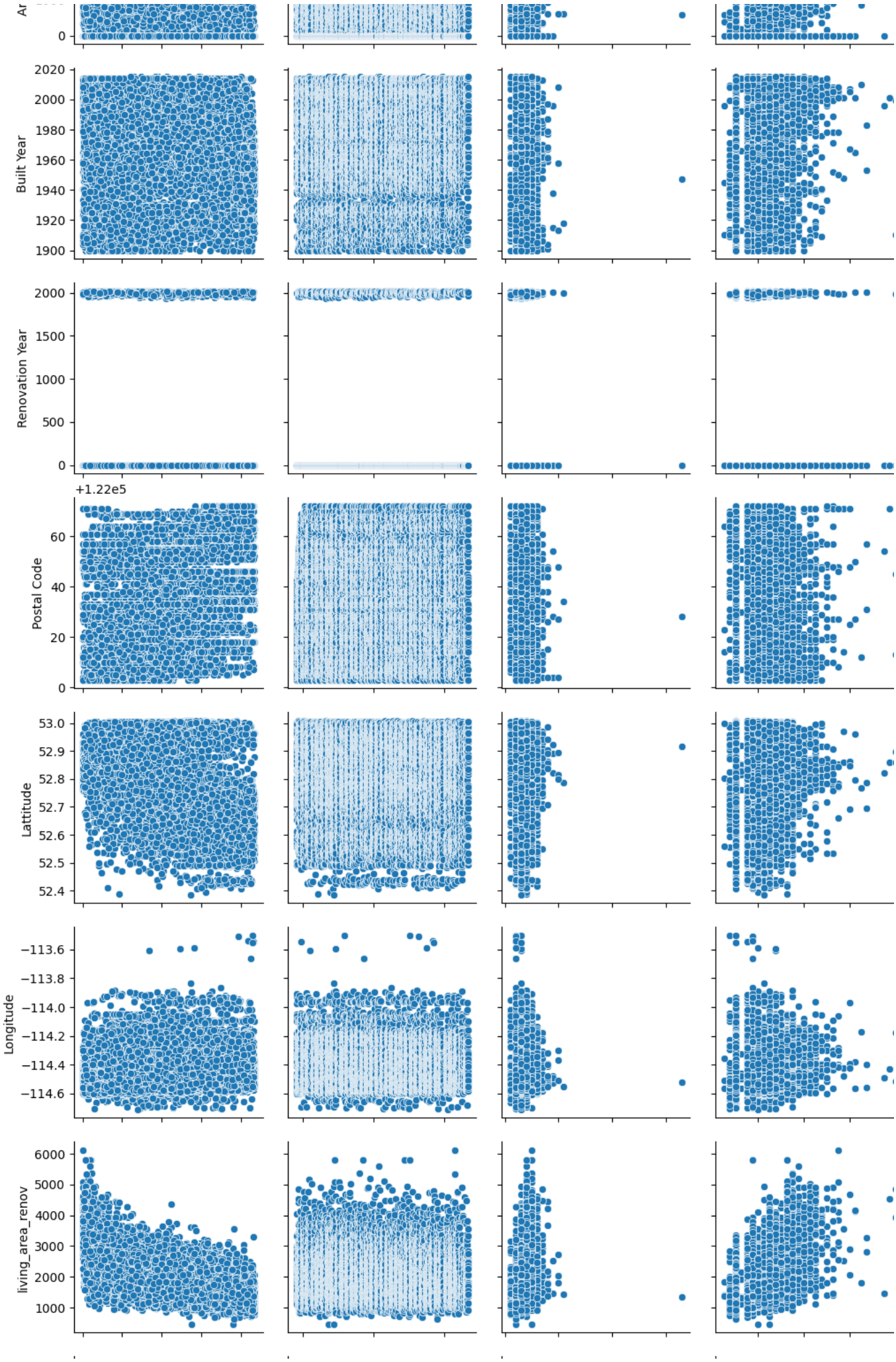



```
#all 23 columns are used in this plot
sns.pairplot(df)
```

```
seaborn.axisgrid.PairGrid at 0x79d6a9d430d0>
```

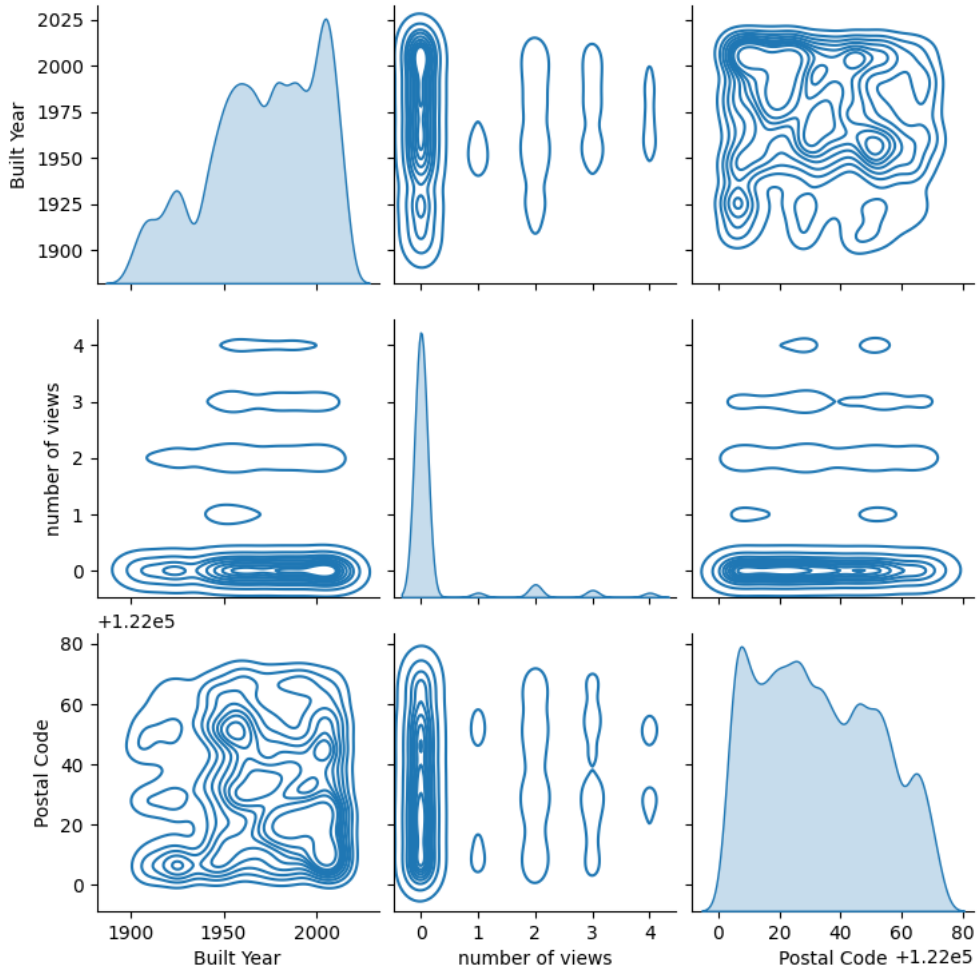



number of floor
waterfront present
number of views
condition of the house
grade of the house
Area of the house(excluding basement)
ea of the basement


Ar
0
Built Year
2020
2000
1980
1960
1940
1920
1900
Renovation Year
2000
1500
1000
500
0
+1.22e5
Postal Code
60
40
20
0
Lattitude
53.0
52.9
52.8
52.7
52.6
52.5
52.4
Longitude
−113.6
−113.8
−114.0
−114.2
−114.4
−114.6
living_area_renov
6000
5000
4000
3000
2000
1000

```
#taking only 3 columns from the dataset and printing the graphs in "kde" form
b=df[["Built Year","number of views","Postal Code"]]
sns.pairplot(b,kind='kde')
```

```
<seaborn.axisgrid.PairGrid at 0x79d6ac527040>
```




## Handling null values

```
df.isnull().sum()
```

```
id                                    0
Date                                  0
number of bedrooms                    0
number of bathrooms                   0
living area                           0
lot area                              0
number of floors                      0
waterfront present                    0
number of views                       0
condition of the house                0
grade of the house                    0
Area of the house(excluding basement)    0
Area of the basement                  0
Built Year                            0
Renovation Year                       0
Postal Code                           0
Lattitude                             0
Longitude                             0
living_area_renov                     0
lot_area_renov                        0
Number of schools nearby              0
Distance from the airport             0
Price                                 0
dtype: int64
```

```
# As there are no null values in the dataset there is no need to handle null values
#but if we want to fill null values we can ues the below method
df['Postal Code'].fillna(value=df['Postal Code'].mean())
```

```
0        122003
1        122004
2        122004
3        122005
```

```
4        122006
          ...
14615    122066
14616    122072
14617    122056
14618    122042
14619    122018
Name: Postal Code, Length: 14620, dtype: int64
```

✓ 0s completed at 5:17 PM