Name:Arpit Aditya

Roll No:21BCE8700

Section:Smart Internez Assignment 2

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

In [ ]:
```
path='/home/godslayer/Desktop/Smart_Internz_Assignment/Practice/Data_set/Hou
df=pd.read_csv(path)
print("The shape of the data set is: ",df.shape)
df=df.head(100) # Taking only 100 rows
col=df.columns
# print(col)
```

```
The shape of the data set is:  (14620, 23)
```

In [ ]:
```
df.info()
# No null values are present in the data set
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 23 columns):
 #   Column                                  Non-Null Count  Dtype  
---  ------                                  --------------  -----  
 0   id                                      100 non-null    int64  
 1   Date                                    100 non-null    int64  
 2   number of bedrooms                      100 non-null    int64  
 3   number of bathrooms                     100 non-null    float64
 4   living area                             100 non-null    int64  
 5   lot area                                100 non-null    int64  
 6   number of floors                        100 non-null    float64
 7   waterfront present                      100 non-null    int64  
 8   number of views                         100 non-null    int64  
 9   condition of the house                  100 non-null    int64  
 10  grade of the house                      100 non-null    int64  
 11  Area of the house(excluding basement)   100 non-null    int64  
 12  Area of the basement                    100 non-null    int64  
 13  Built Year                              100 non-null    int64  
 14  Renovation Year                         100 non-null    int64  
 15  Postal Code                             100 non-null    int64  
 16  Lattitude                               100 non-null    float64
 17  Longitude                               100 non-null    float64
 18  living_area_renov                       100 non-null    int64  
 19  lot_area_renov                          100 non-null    int64  
 20  Number of schools nearby                100 non-null    int64  
 21  Distance from the airport               100 non-null    int64  
 22  Price                                   100 non-null    int64  
dtypes: float64(4), int64(19)
memory usage: 18.1 KB
```

In [ ]:
```
df.isnull().any()
# We can see that there are no null values in the data set
# Would have returned True if there were any null values
display(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
```

In [ ]:
```
# null values are in the numerical parameters, replace it with median or mea
# null values are in the categorical parameters, replace it with mode
```

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

Out[ ]:

	id	Date	number of bedrooms	number of bathrooms	living area	lot area
count	1.000000e+02	100.000000	100.000000	100.000000	100.00000	100.000000
mean	6.762819e+09	42491.860000	3.530000	2.290000	2389.90000	18152.190000
std	5.727200e+03	1.295057	0.892788	0.848885	1397.83707	37055.712743
min	6.762810e+09	42491.000000	2.000000	1.000000	800.00000	1180.000000
25%	6.762813e+09	42491.000000	3.000000	1.750000	1645.00000	5737.500000
50%	6.762817e+09	42491.000000	3.000000	2.500000	2205.00000	8459.000000
75%	6.762823e+09	42493.000000	4.000000	2.500000	2827.50000	11959.000000
max	6.762830e+09	42494.000000	7.000000	8.000000	13540.00000	307752.000000

8 rows × 23 columns

In [ ]:
```
df.head()
```

Out[ ]:

	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

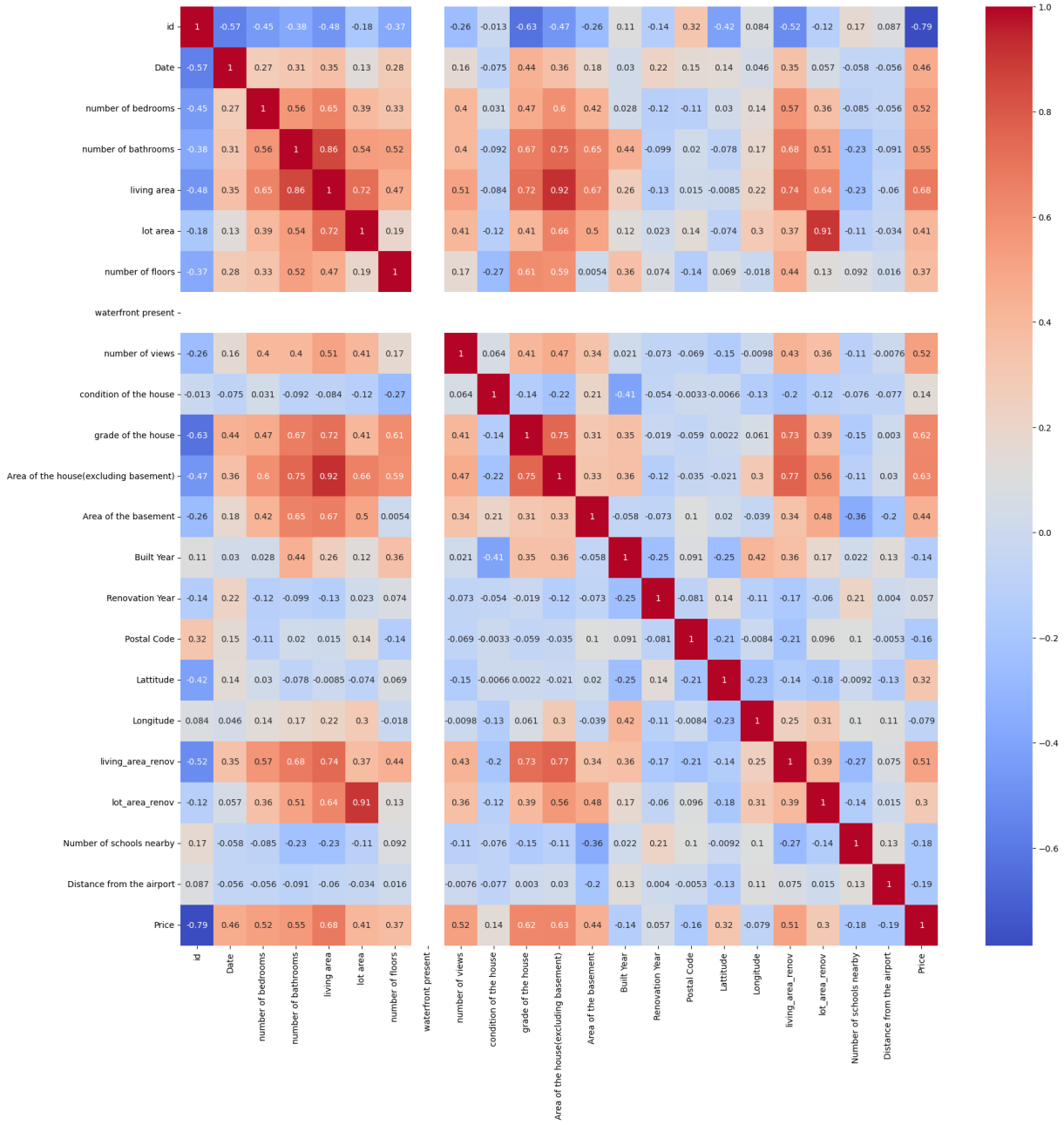
5 rows × 23 columns

In [ ]:
```
# Correlation for printing the heatmap
# Correlation is a statistical measure used to assess the strength and direc
# Correlation coefficients are values between -1 and 1 where:
# 1: Perfect positive linear correlation
# 0: No linear correlation, the two variables most likely do not affect each
# -1: Perfect negative linear correlation
display(df.corr(numeric_only=True).style.background_gradient(cmap='coolwarm'
plt.figure(figsize=(20,20))
sns.heatmap(df.corr(),annot=True,cmap='coolwarm')
# 23X23 matrix
```

```
/home/godslayer/.local/lib/python3.11/site-packages/pandas/io/formats/style.
py:3618: RuntimeWarning: All-NaN slice encountered
  smin = np.nanmin(gmap) if vmin is None else vmin
/home/godslayer/.local/lib/python3.11/site-packages/pandas/io/formats/style.
py:3619: RuntimeWarning: All-NaN slice encountered
  smax = np.nanmax(gmap) if vmax is None else vmax
```

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	numb of floc
id	1.000000	-0.567509	-0.450299	-0.379802	-0.477011	-0.182393	-0.3682
Date	-0.567509	1.000000	0.274495	0.310651	0.353027	0.132986	0.2828
number of bedrooms	-0.450299	0.274495	1.000000	0.561512	0.645049	0.386226	0.3340
number of bathrooms	-0.379802	0.310651	0.561512	1.000000	0.860409	0.539476	0.5222
living area	-0.477011	0.353027	0.645049	0.860409	1.000000	0.722268	0.4675
lot area	-0.182393	0.132986	0.386226	0.539476	0.722268	1.000000	0.1945
number of floors	-0.368286	0.282878	0.334017	0.522247	0.467570	0.194519	1.0000
waterfront present	nan	nan	nan	nan	nan	nan	n
number of views	-0.261100	0.158414	0.402134	0.401153	0.506880	0.409934	0.1671
condition of the house	-0.012986	-0.074760	0.030832	-0.091877	-0.084381	-0.120353	-0.2660
grade of the house	-0.625553	0.436223	0.467144	0.668374	0.715829	0.412004	0.6085
Area of the house(excluding basement)	-0.468893	0.355131	0.598945	0.754409	0.921148	0.656905	0.5928
Area of the basement	-0.264317	0.180178	0.424281	0.650631	0.671792	0.501222	0.0054
Built Year	0.112705	0.030275	0.027767	0.442844	0.257164	0.121616	0.3579
Renovation Year	-0.135048	0.219896	-0.121290	-0.098569	-0.128042	0.022738	0.0736
Postal Code	0.317410	0.151507	-0.106761	0.019730	0.015474	0.144058	-0.1352
Lattitude	-0.417184	0.144860	0.029929	-0.077842	-0.008523	-0.074424	0.0686
Longitude	0.084126	0.046485	0.137682	0.174492	0.218497	0.295094	-0.0176
living_area_renov	-0.518796	0.351144	0.573401	0.677203	0.742376	0.374970	0.4435
lot_area_renov	-0.119667	0.057215	0.357600	0.507096	0.638851	0.906169	0.1262
Number of schools nearby	0.165070	-0.058389	-0.085234	-0.228756	-0.232208	-0.107929	0.0922
Distance from the airport	0.087204	-0.056133	-0.056418	-0.090655	-0.059846	-0.033921	0.0160
Price	-0.785575	0.464336	0.523151	0.547501	0.680593	0.408524	0.3730

Out[ ]: <Axes: >



In [ ]:
```
# We want to predict price so we need correlation of price with other parame

cor=df.corr()['Price'].sort_values(ascending=False)
display(cor)
```

```
Price                                    1.000000
living area                              0.680593
Area of the house(excluding basement)    0.634136
grade of the house                       0.623934
number of bathrooms                      0.547501
number of views                          0.523807
number of bedrooms                       0.523151
living_area_renov                        0.507034
Date                                     0.464336
Area of the basement                     0.443498
lot area                                 0.408524
number of floors                         0.373094
Lattitude                                0.316863
lot_area_renov                           0.300415
condition of the house                   0.144376
Renovation Year                          0.056882
Longitude                               -0.078911
Built Year                              -0.135110
Postal Code                             -0.161325
Number of schools nearby                -0.181650
Distance from the airport               -0.185089
id                                      -0.785575
waterfront present                            NaN
Name: Price, dtype: float64
```

In [ ]:
```
# We can see the price is highly correlated with
# 1. Area
# 2. Grade of the house
# 3. Area of house
# 4. living_area_renov
# 5. Number of bathrooms
# 6. Number of views
# 7. Number of bedrooms
col=['Price','living area','Area of the house(excluding basement)','living_a
print(len(col))
# We will perform univariate,bivariate and multivariate analysis on the abov
```

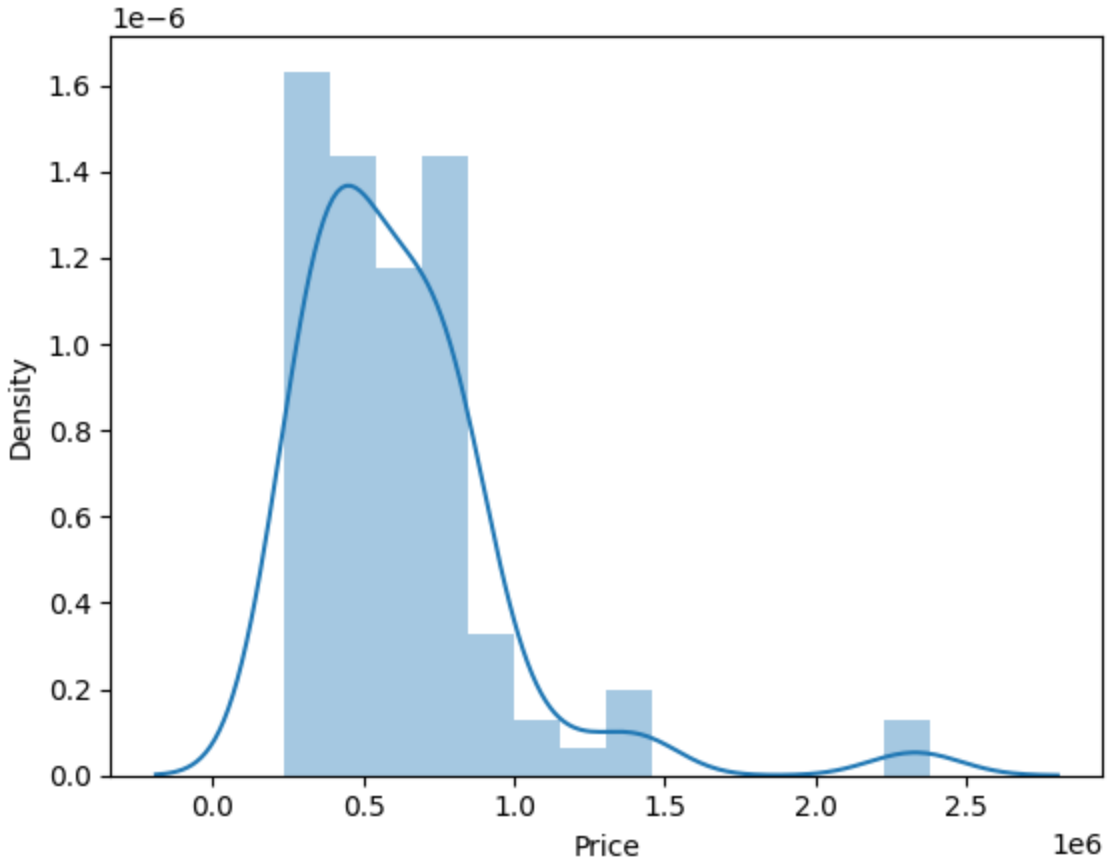
```
7
```

Univariate Analysis

In [ ]:
```
sns.distplot(df[col[0]])
```

```
/tmp/ipykernel_37638/3210613974.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[col[0]])
```

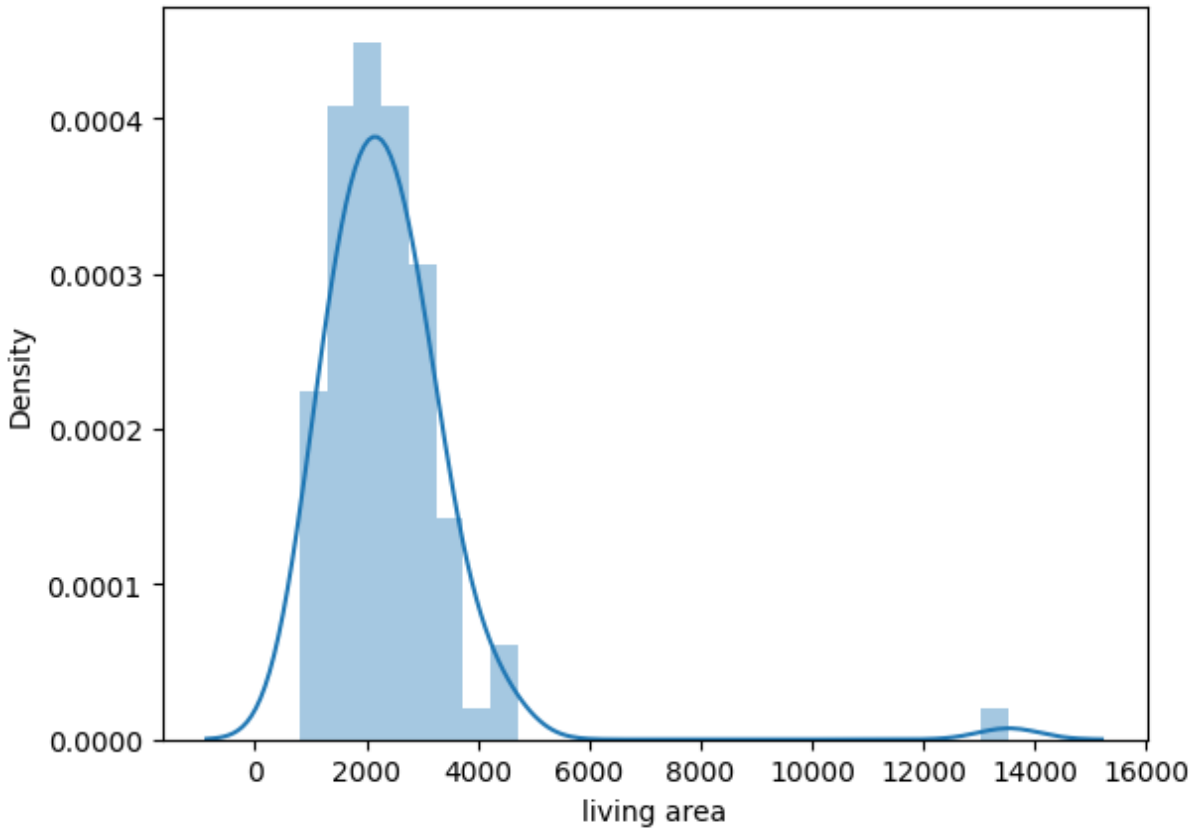
Out[ ]: <Axes: xlabel='Price', ylabel='Density'>



In [ ]:
```
sns.distplot(df[col[1]])
```

```
/tmp/ipykernel_37638/1526242776.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[col[1]])
```

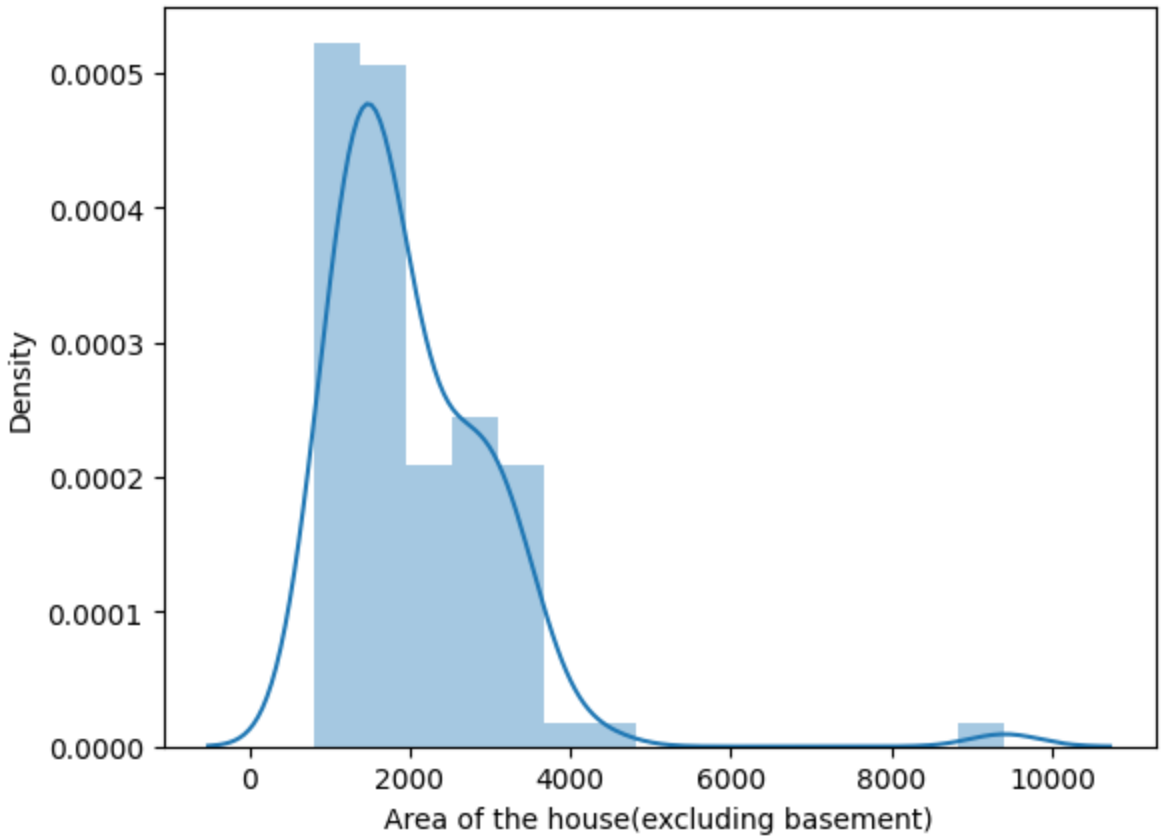
Out[ ]: <Axes: xlabel='living area', ylabel='Density'>



In [ ]:
```
sns.distplot(df[col[2]])
```

```
/tmp/ipykernel_37638/4253019484.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[col[2]])
```

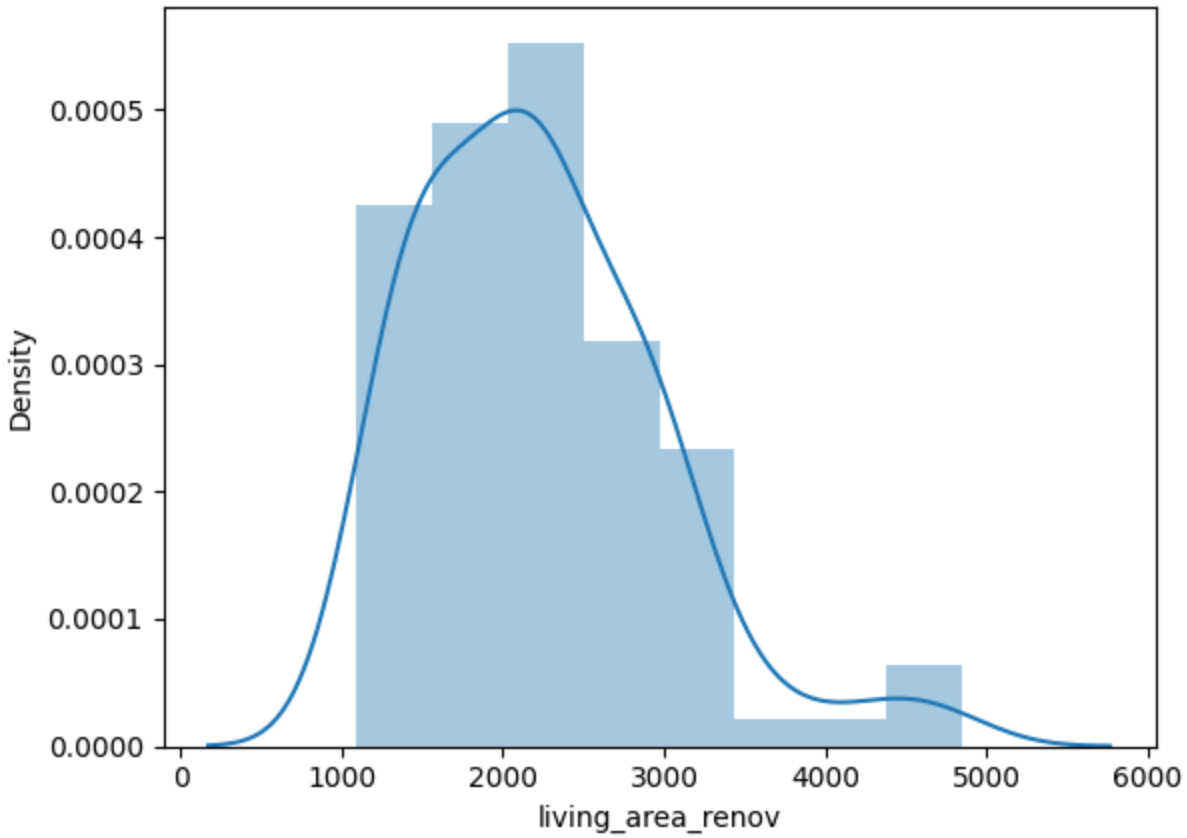
Out[ ]: <Axes: xlabel='Area of the house(excluding basement)', ylabel='Density'>



In [ ]:
```
sns.distplot(df[col[3]])
```

```
/tmp/ipykernel_37638/2692862890.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[col[3]])
```

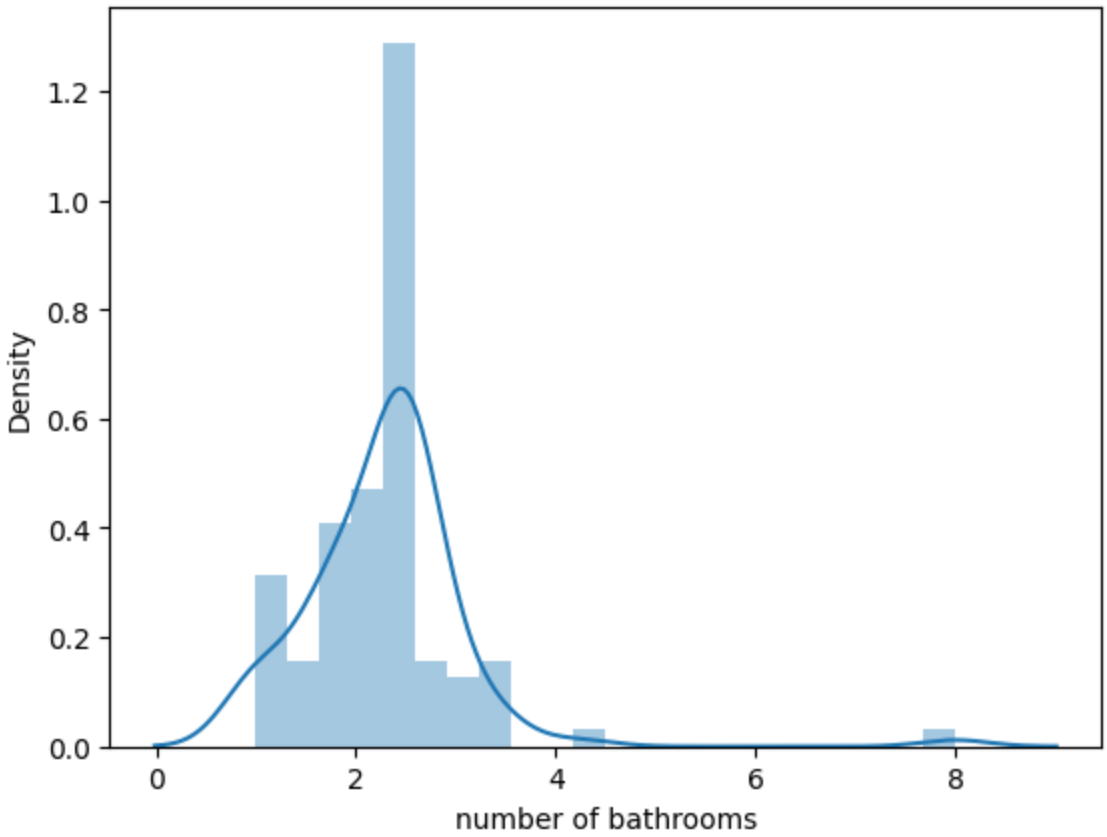
Out[ ]:
```
<Axes: xlabel='living_area_renov', ylabel='Density'>
```



In [ ]:
```
sns.distplot(df[col[4]])
```

```
/tmp/ipykernel_37638/1077085201.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[col[4]])
```

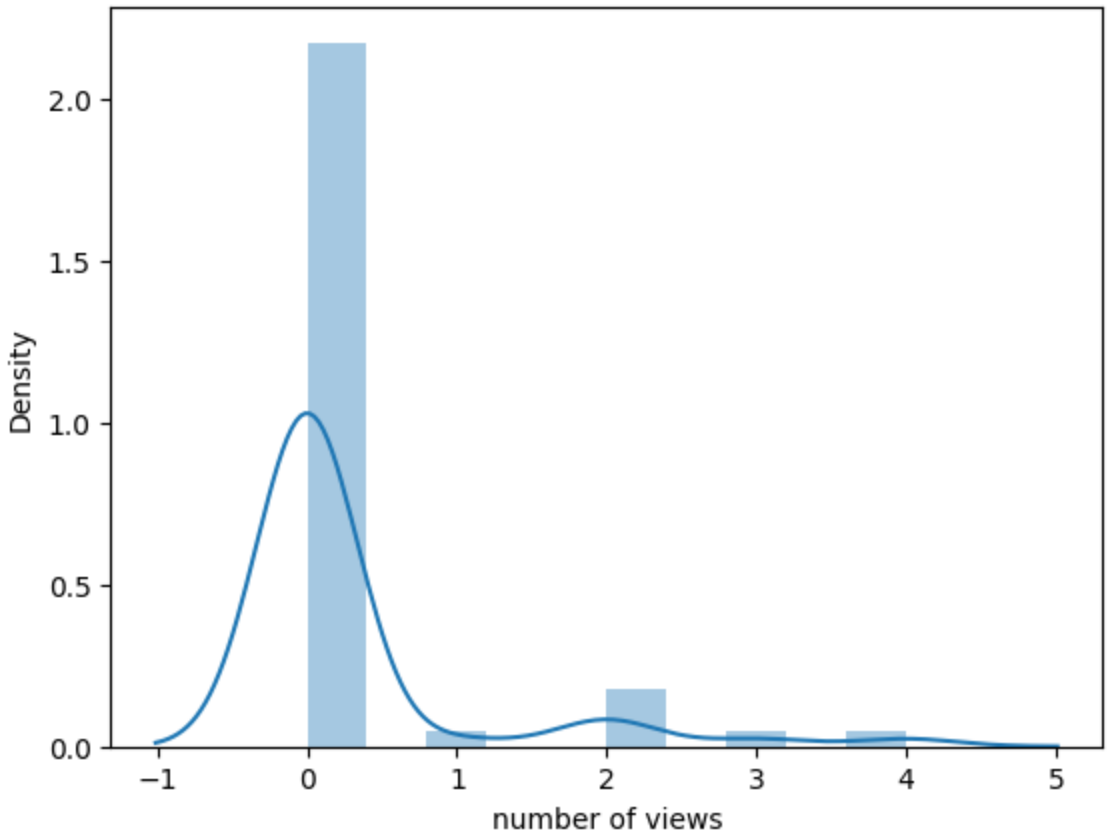
Out[ ]:
```
<Axes: xlabel='number of bathrooms', ylabel='Density'>
```



In [ ]:
```
sns.distplot(df[col[5]])
```

```
/tmp/ipykernel_37638/1614590936.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[col[5]])
```

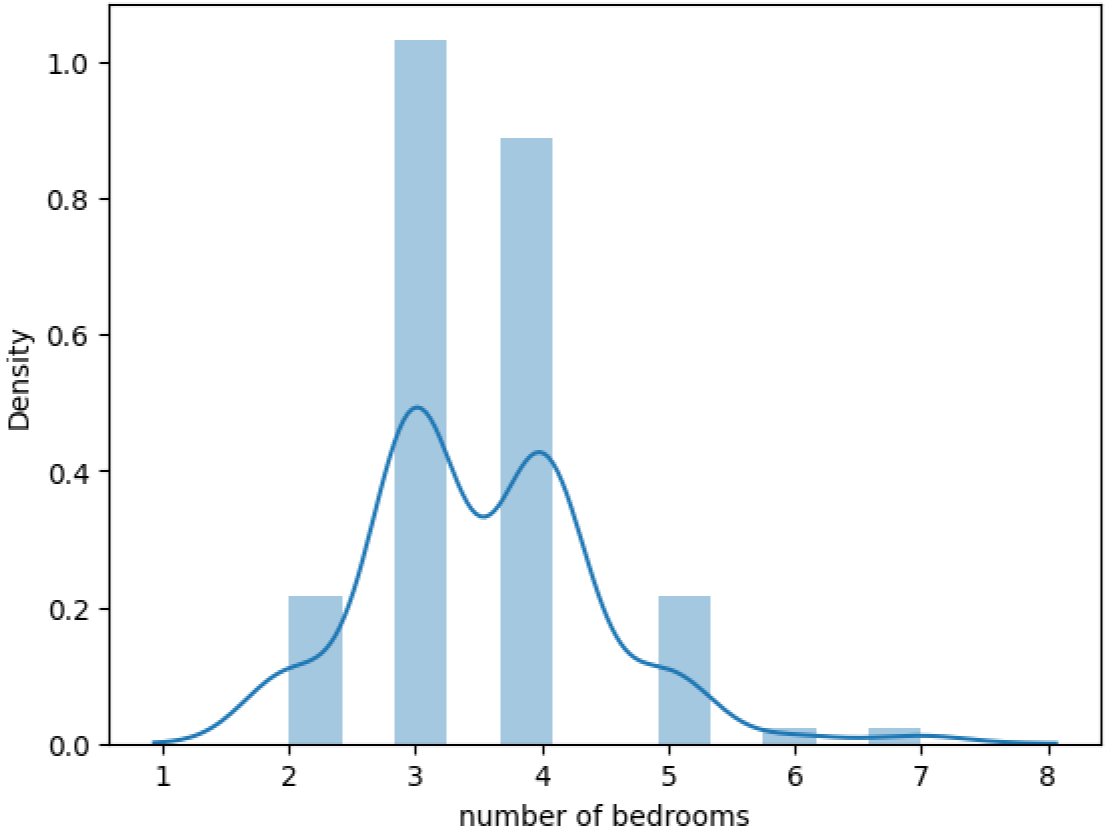
Out[ ]: <Axes: xlabel='number of views', ylabel='Density'>



In [ ]:
```
sns.distplot(df[col[6]])
```

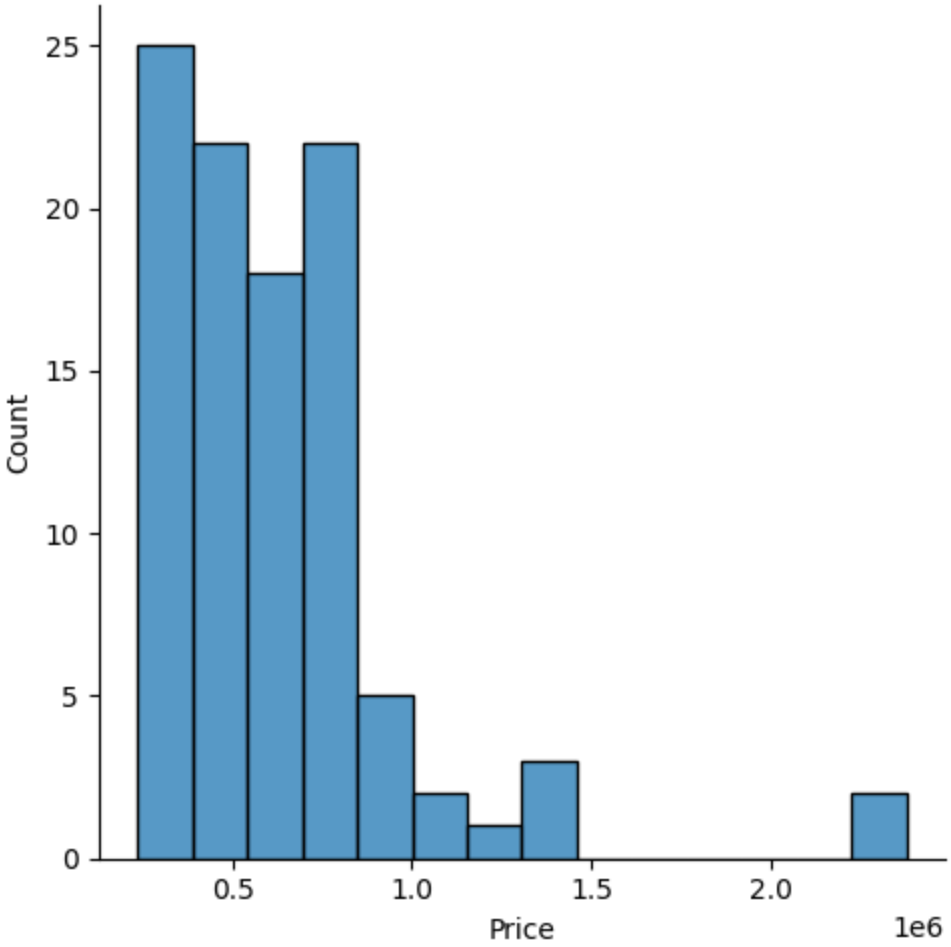
```
/tmp/ipykernel_37638/3258826001.py:1: UserWarning:

`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[col[6]])
```

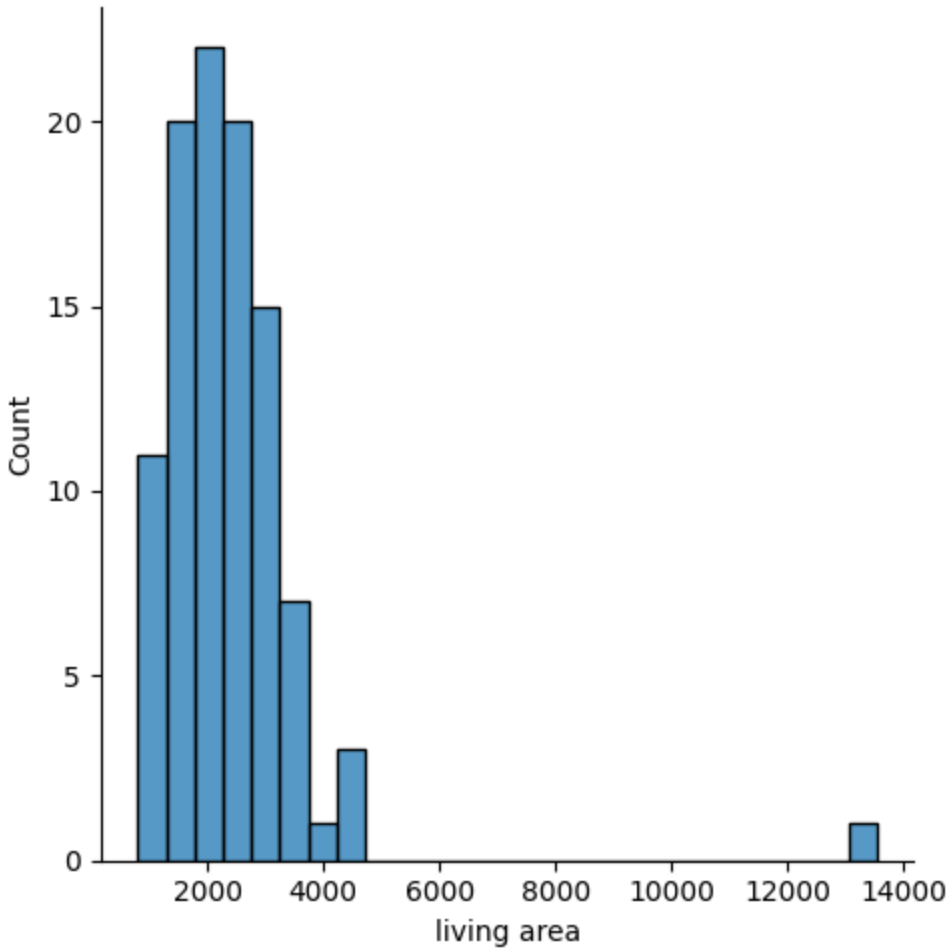
Out[ ]:
```
<Axes: xlabel='number of bedrooms', ylabel='Density'>
```

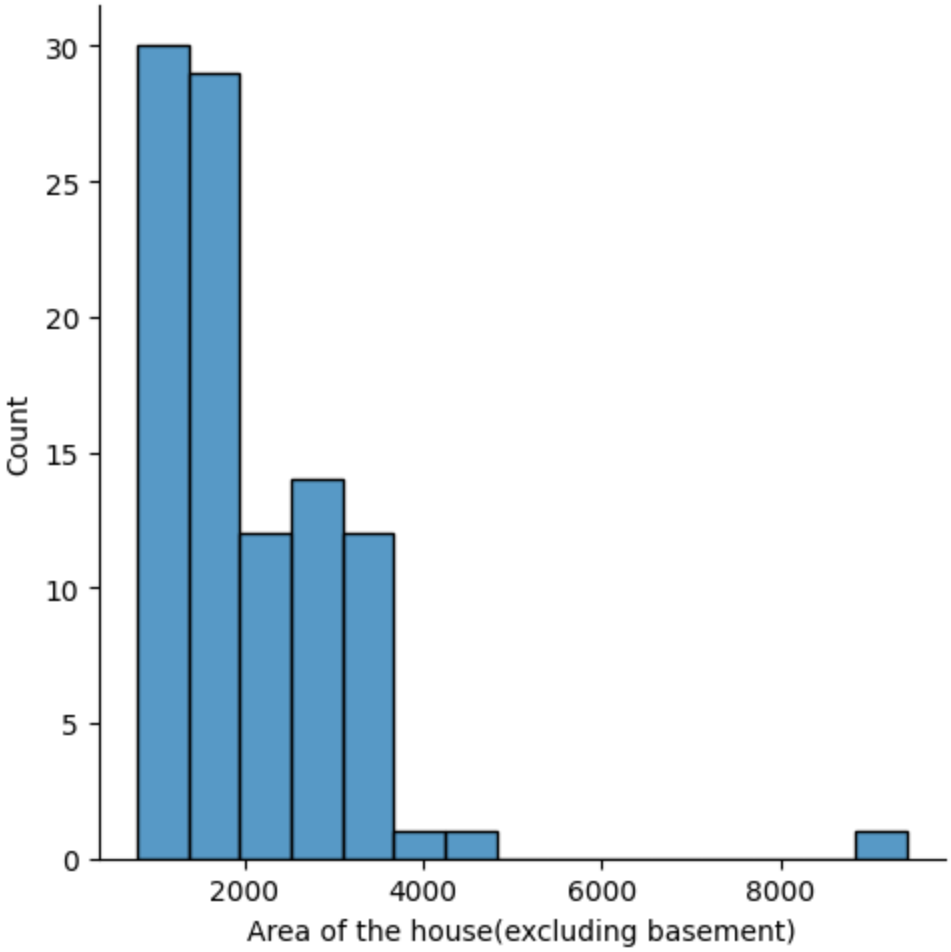


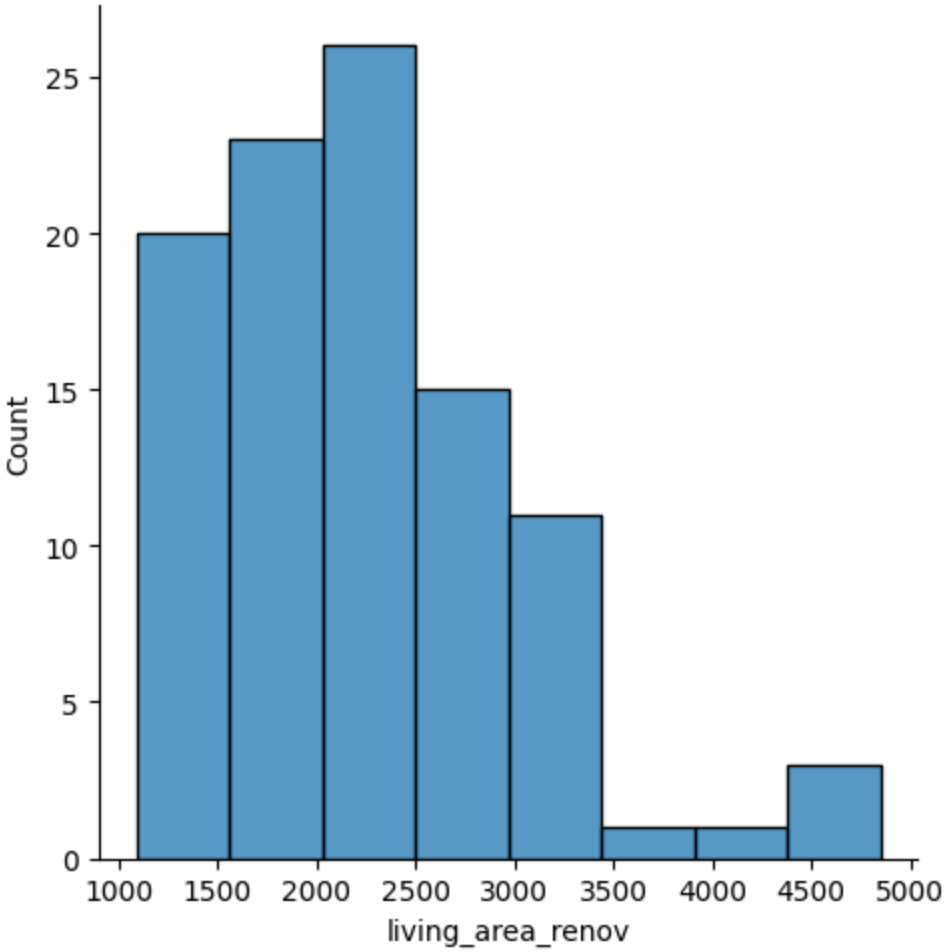
In [ ]:
```
for i in col:
    sns.displot(df[i])
```

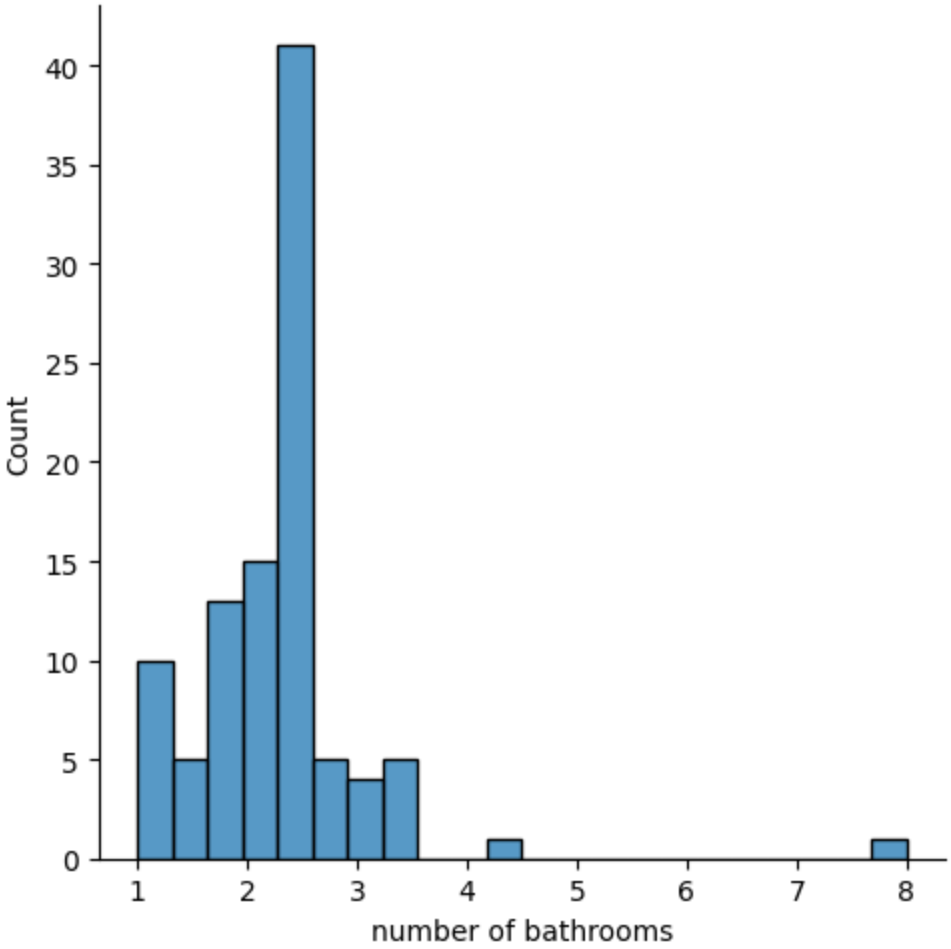
```
/home/godslayer/.local/lib/python3.11/site-packages/seaborn/axisgrid.py:118:
UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
/home/godslayer/.local/lib/python3.11/site-packages/seaborn/axisgrid.py:118:
UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
/home/godslayer/.local/lib/python3.11/site-packages/seaborn/axisgrid.py:118:
UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
/home/godslayer/.local/lib/python3.11/site-packages/seaborn/axisgrid.py:118:
UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
/home/godslayer/.local/lib/python3.11/site-packages/seaborn/axisgrid.py:118:
UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
/home/godslayer/.local/lib/python3.11/site-packages/seaborn/axisgrid.py:118:
UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
/home/godslayer/.local/lib/python3.11/site-packages/seaborn/axisgrid.py:118:
UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

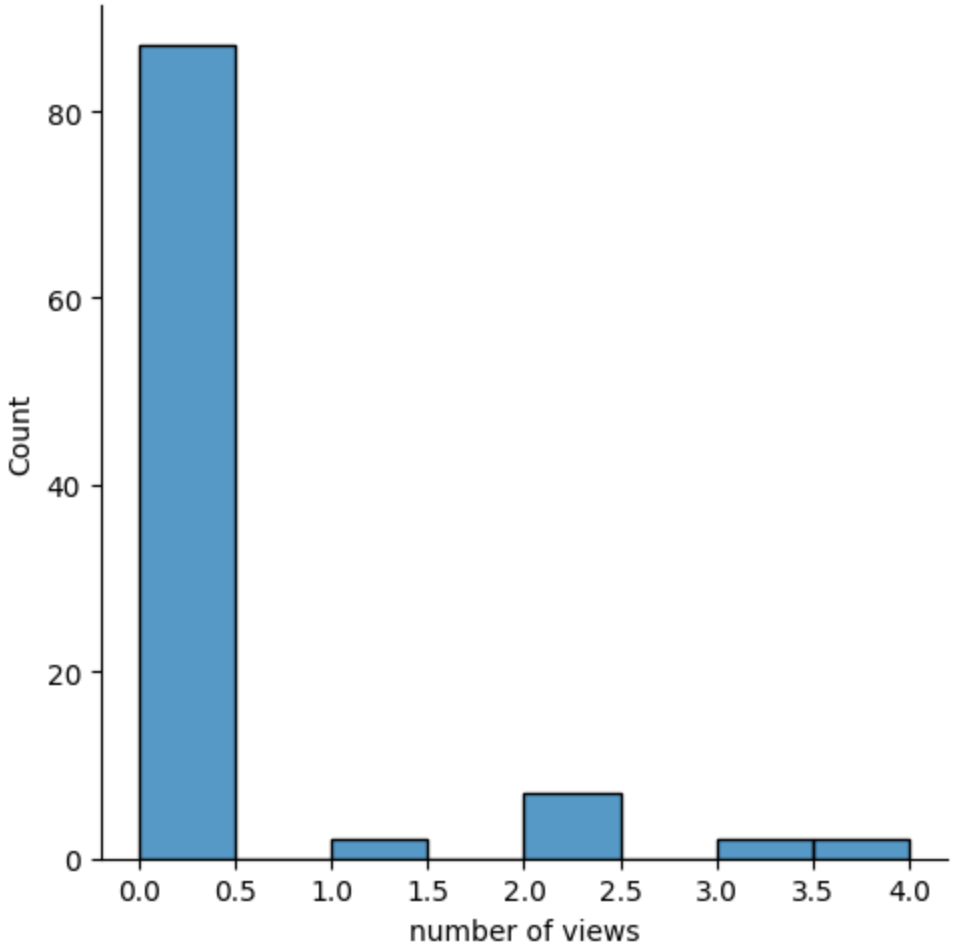

25
20
15
10
5
0
Count
0.5
1.0
1.5
2.0
Price
1e6

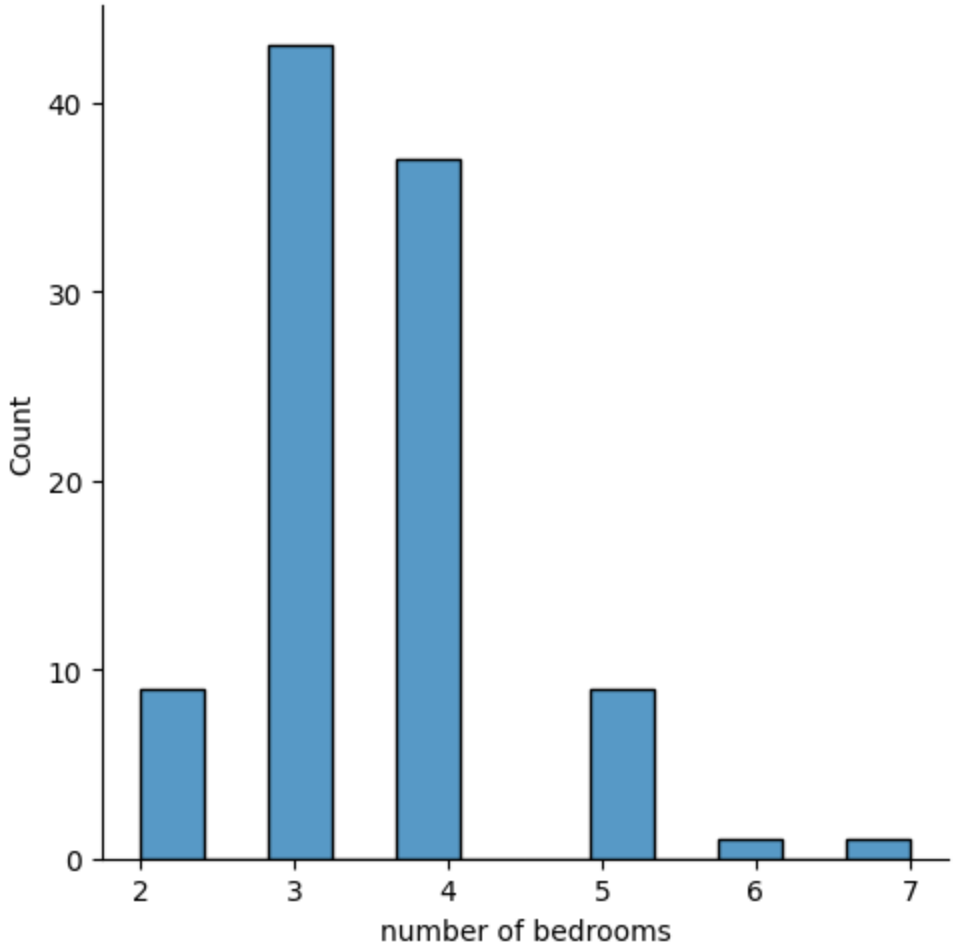

Count
0
5
10
15
20
2000
4000
6000
8000
10000
12000
14000
living area


30
25
20
15
10
5
0
Count
2000
4000
6000
8000
Area of the house(excluding basement)


25
20
15
10
5
0
Count
1000
1500
2000
2500
3000
3500
4000
4500
5000
living_area_renov

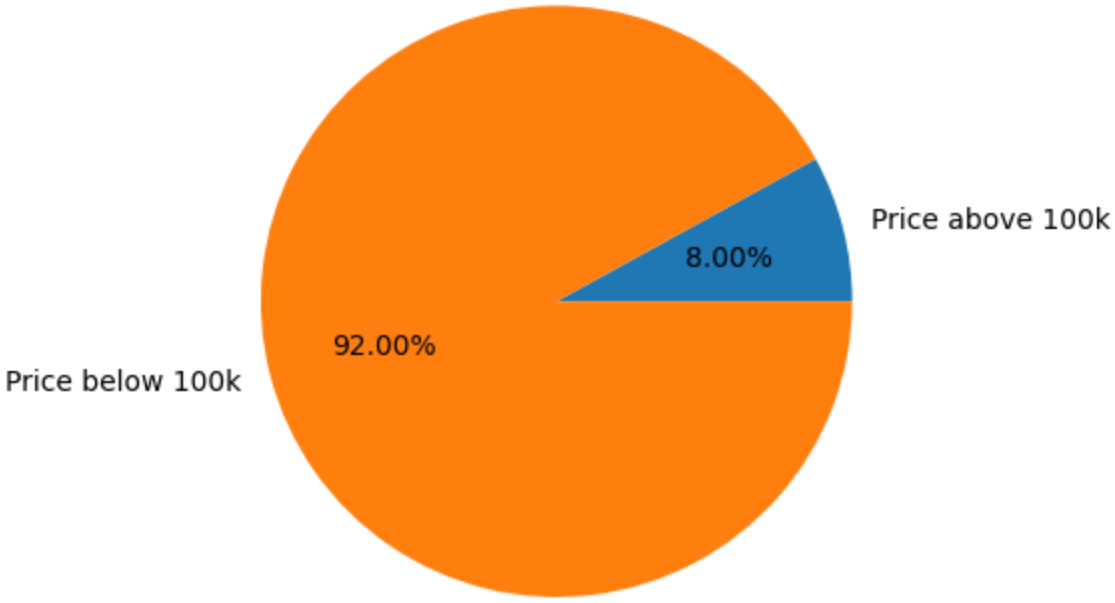

Count
number of bathrooms
0
5
10
15
20
25
30
35
40
1
2
3
4
5
6
7
8


Count
80
60
40
20
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
number of views



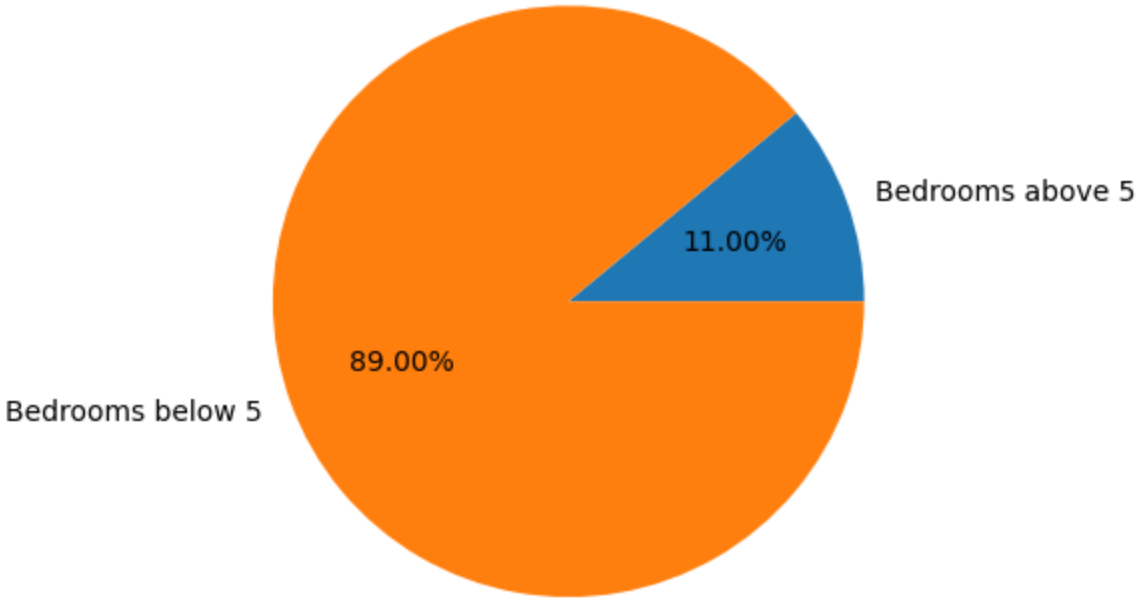
In [ ]:
```
# pie chart
# 1. Price
price_above_100k=df[df['Price']>=1000000]
price_below_100k=df[df['Price']<1000000]
plt.pie([len(price_above_100k),len(price_below_100k)],labels=['Price above 1
```

Out[ ]:
```
([<matplotlib.patches.Wedge at 0x7f030a8dae90>,
  <matplotlib.patches.Wedge at 0x7f030a8dbc50>],
 [Text(1.0654414787782402, 0.27355886989611045, 'Price above 100k'),
  Text(-1.0654414659720242, -0.2735589197730251, 'Price below 100k')],
 [Text(0.5811498975154037, 0.14921392903424205, '8.00%'),
  Text(-0.581149890530195, -0.14921395623983186, '92.00%')])
```



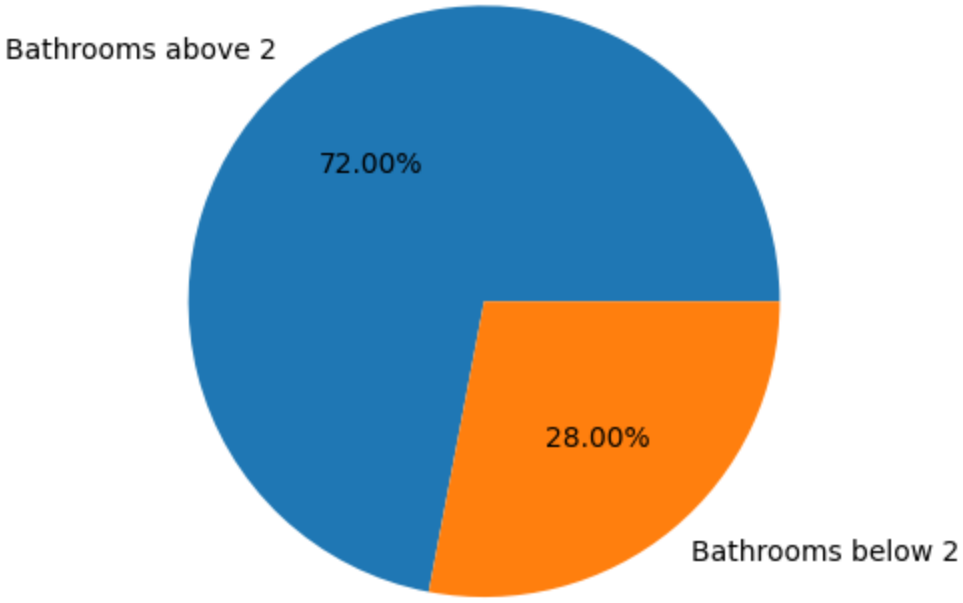
In [ ]:
```
# rooms with more than 5 bedrooms
bedroom_above_5=df[df['number of bedrooms']>=5]
bedroom_below_5=df[df['number of bedrooms']<5]
plt.pie([len(bedroom_above_5),len(bedroom_below_5)],labels=['Bedrooms above
```

Out[ ]:
```
([<matplotlib.patches.Wedge at 0x7f030a917890>,
  <matplotlib.patches.Wedge at 0x7f030a924a50>],
 [Text(1.0349688465473768, 0.372611710331805, 'Bedrooms above 5'),
  Text(-1.0349688639905885, -0.37261166188141603, 'Bedrooms below 5')],
 [Text(0.5645284617531144, 0.20324275109007545, '11.00%'),
  Text(-0.5645284712675936, -0.20324272466259055, '89.00%')])
```



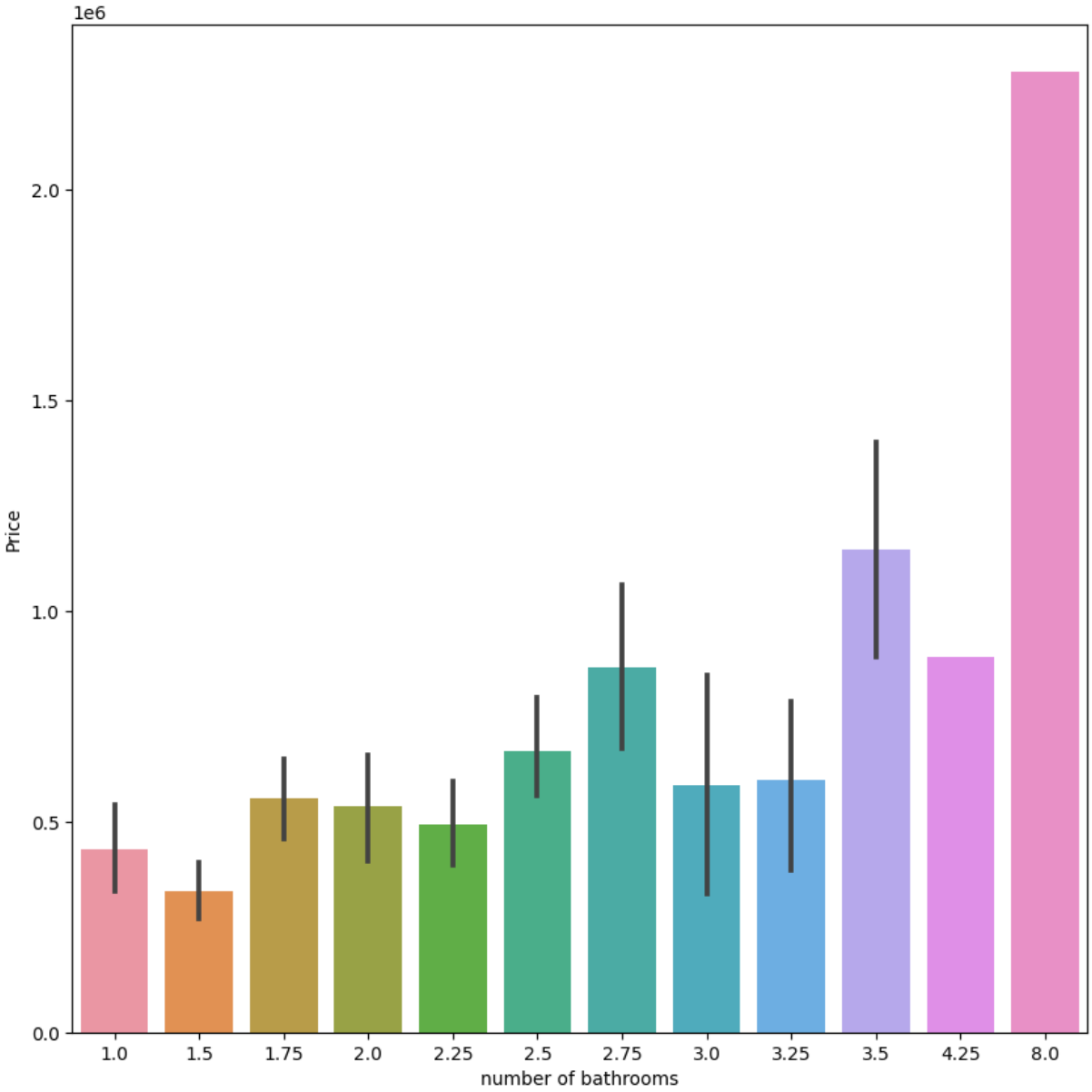
In [ ]:
```
# houses with more than 2 bathroom
bathroom_above_2=df[df['number of bathrooms']>=2]
bathroom_below_2=df[df['number of bathrooms']<2]
plt.pie([len(bathroom_above_2),len(bathroom_below_2)],labels=['Bathrooms abo
```

Out[ ]:
```
([<matplotlib.patches.Wedge at 0x7f030a965750>,
  <matplotlib.patches.Wedge at 0x7f030a95a3d0>],
 [Text(-0.7011664649040886, 0.8475645040313473, 'Bathrooms above 2'),
  Text(0.7011665442588013, -0.8475644383834013, 'Bathrooms below 2')],
 [Text(-0.3824544354022301, 0.46230791128982573, '72.00%'),
  Text(0.38245447868661886, -0.4623078754818552, '28.00%')])
```



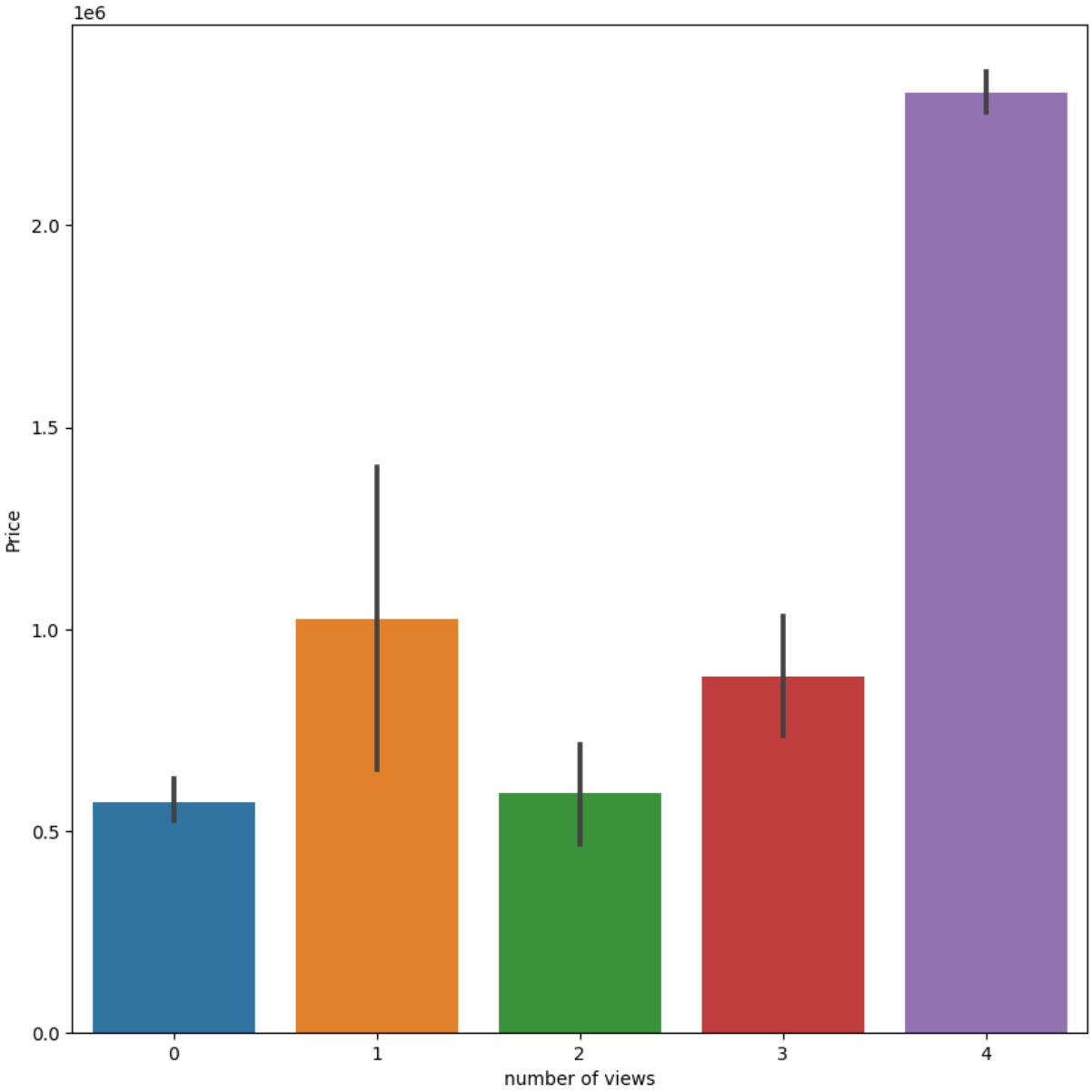
In [ ]:
```
# barplot for number of bedrooms
plt.figure(figsize=(10,10))
sns.barplot(x=df['number of bathrooms'],y=df['Price'])
```

Out[ ]:
```
<Axes: xlabel='number of bathrooms', ylabel='Price'>
```



In [ ]:
```
# barplot for number of views
plt.figure(figsize=(10,10))
sns.barplot(x=df['number of views'],y=df['Price'])
```

Out[ ]:
```
<Axes: xlabel='number of views', ylabel='Price'>
```

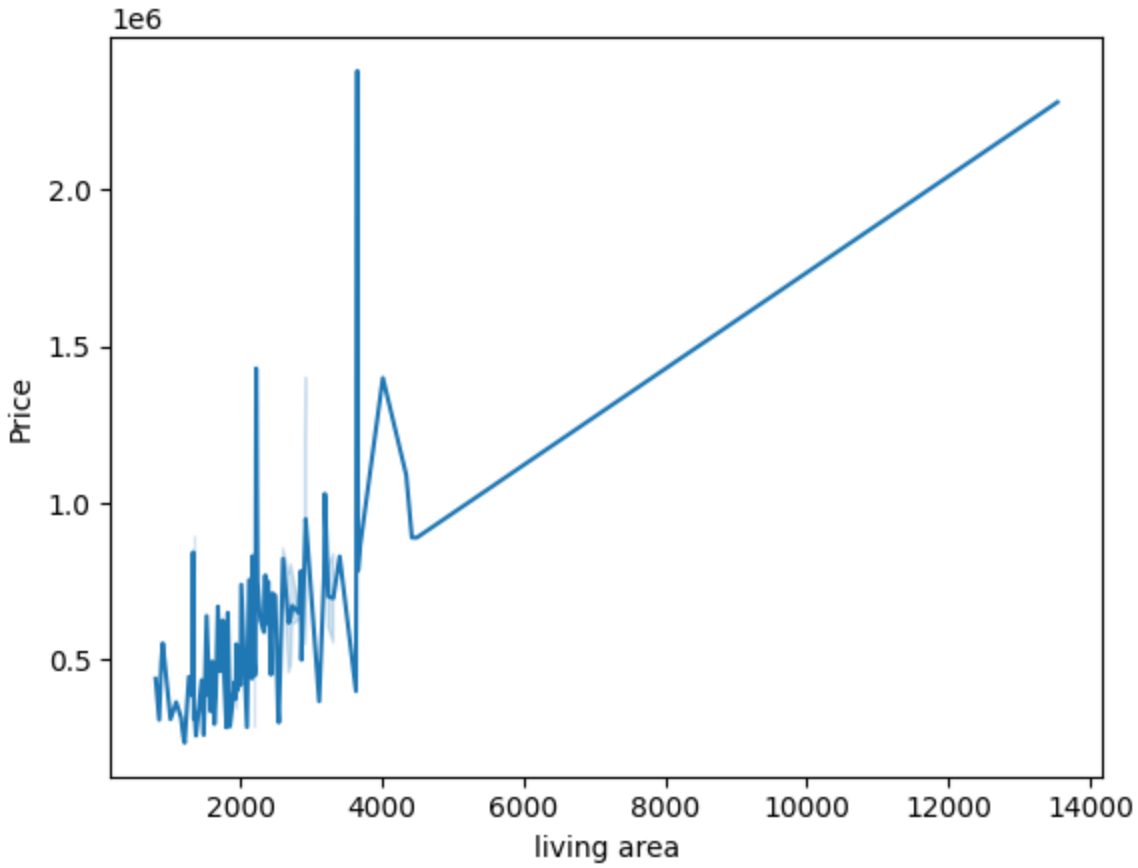


Bivariate Analysis

In [ ]:
```
print(col)
sns.lineplot(x=df[col[1]],y=df[col[0]])
```

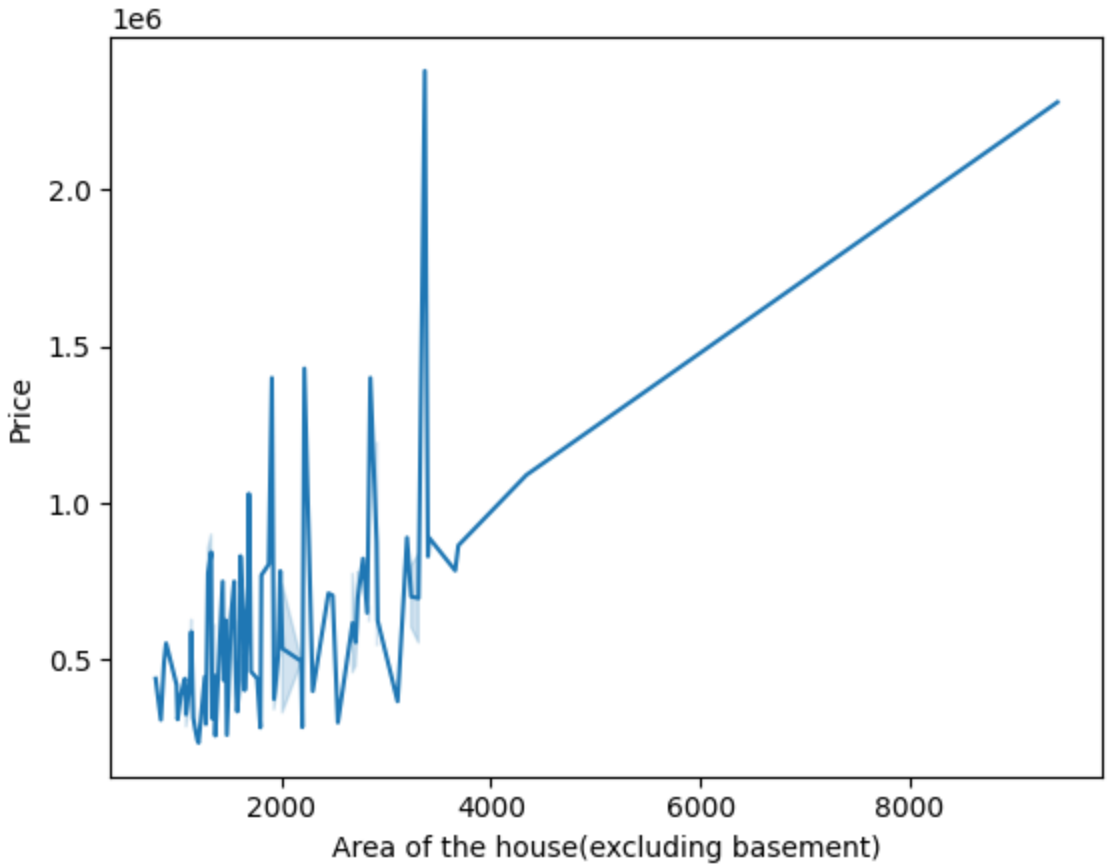
```
['Price', 'living area', 'Area of the house(excluding basement)', 'living_ar
ea_renov', 'number of bathrooms', 'number of views', 'number of bedrooms']
```

Out[ ]: <Axes: xlabel='living area', ylabel='Price'>



In [ ]:
```
sns.lineplot(x=df[col[2]],y=df[col[0]])
```

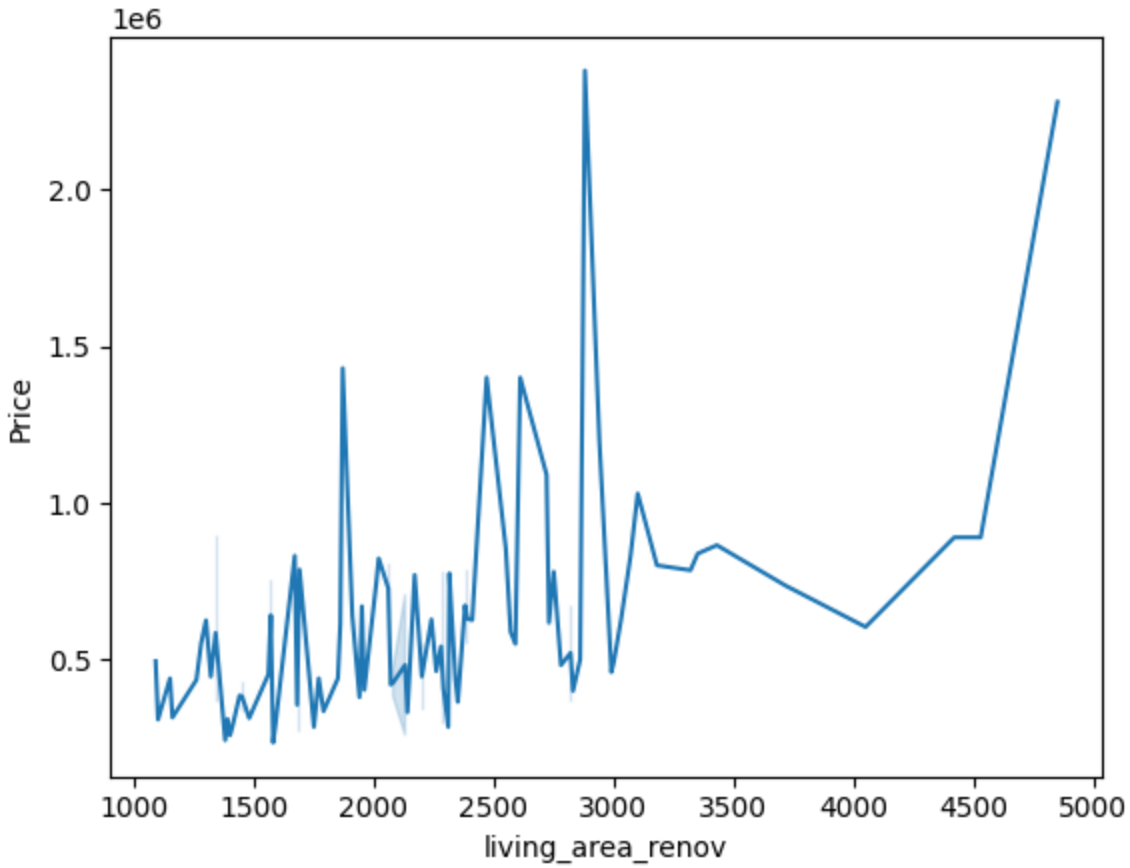
Out[ ]:
```
<Axes: xlabel='Area of the house(excluding basement)', ylabel='Price'>
```



In [ ]:
```
print(col)
sns.lineplot(x=df[col[3]],y=df[col[0]])
```

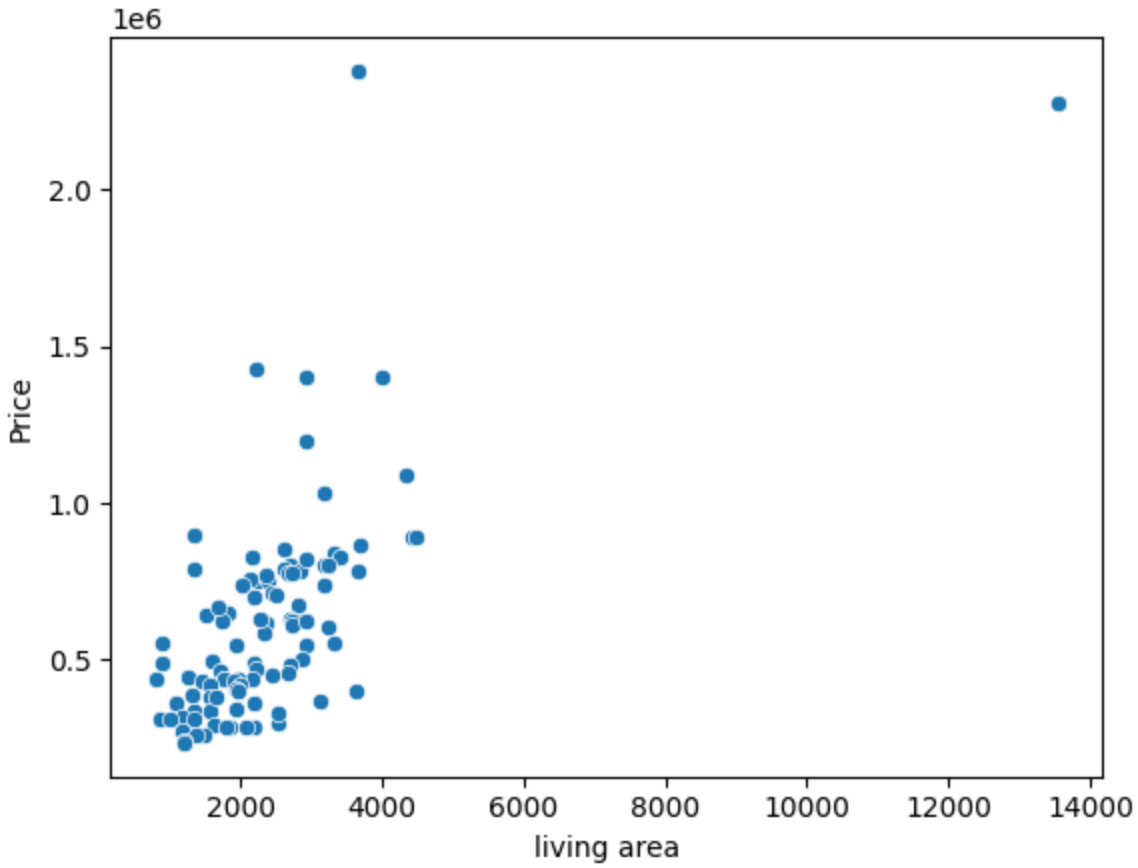
```
['Price', 'living area', 'Area of the house(excluding basement)', 'living_ar
ea_renov', 'number of bathrooms', 'number of views', 'number of bedrooms']
```

Out[ ]: <Axes: xlabel='living_area_renov', ylabel='Price'>



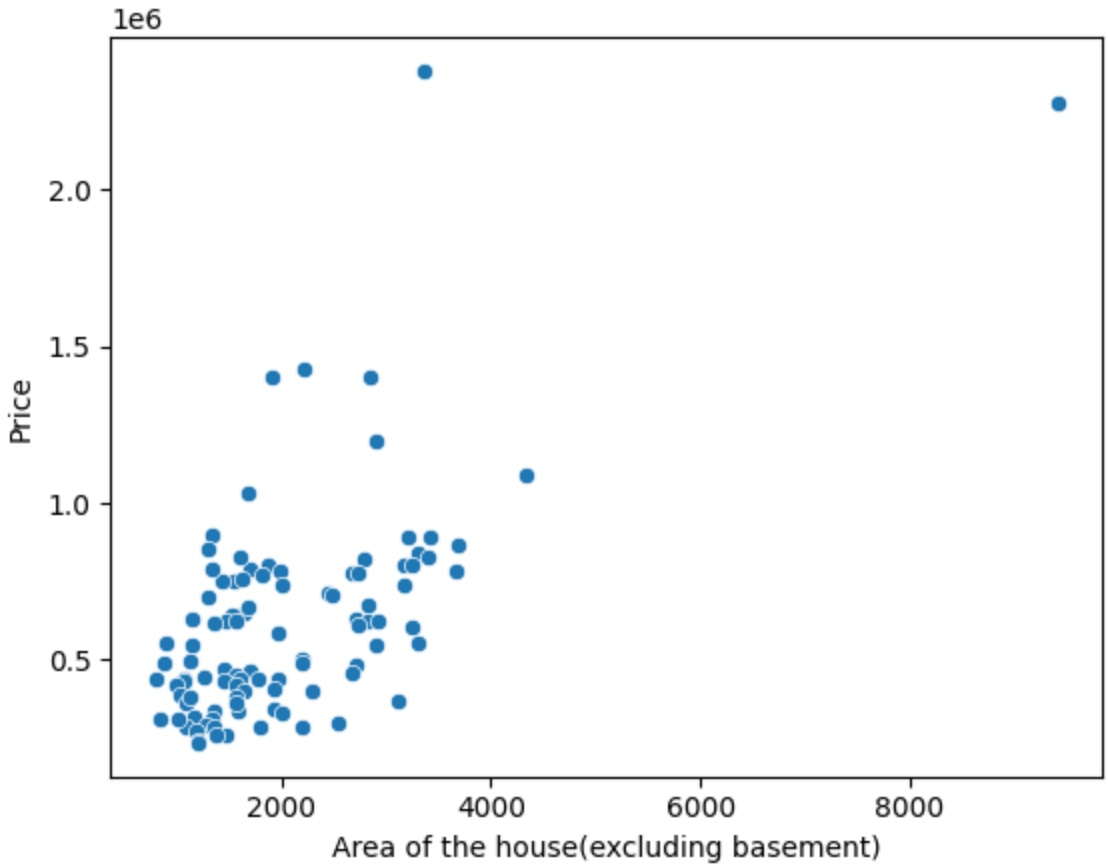
In [ ]:
```
sns.scatterplot(x = df[col[1]],y=df[col[0]])
```

Out[ ]:
```
<Axes: xlabel='living area', ylabel='Price'>
```



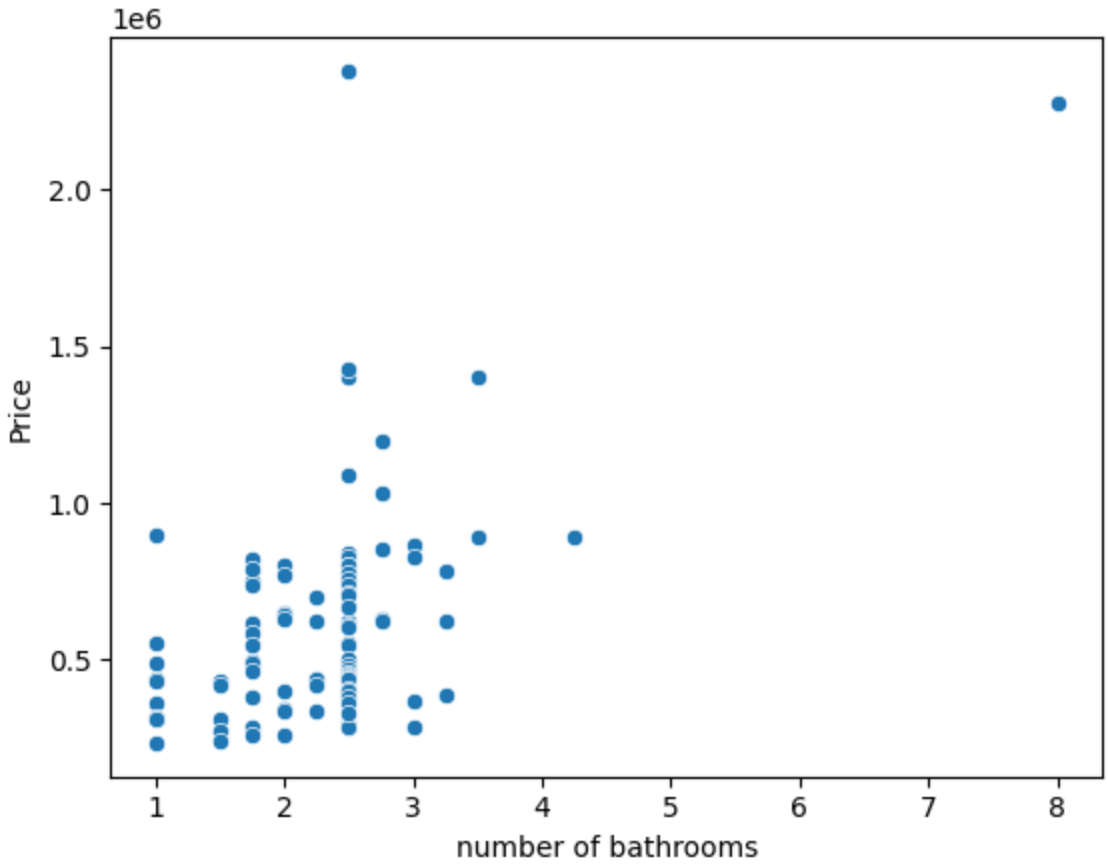
In [ ]:
```
sns.scatterplot(x = df[col[2]],y=df[col[0]])
```

Out[ ]:
```
<Axes: xlabel='Area of the house(excluding basement)', ylabel='Price'>
```



In [ ]:
```
sns.scatterplot(x = df[col[4]],y=df[col[0]])
```

Out[ ]:
```
<Axes: xlabel='number of bathrooms', ylabel='Price'>
```

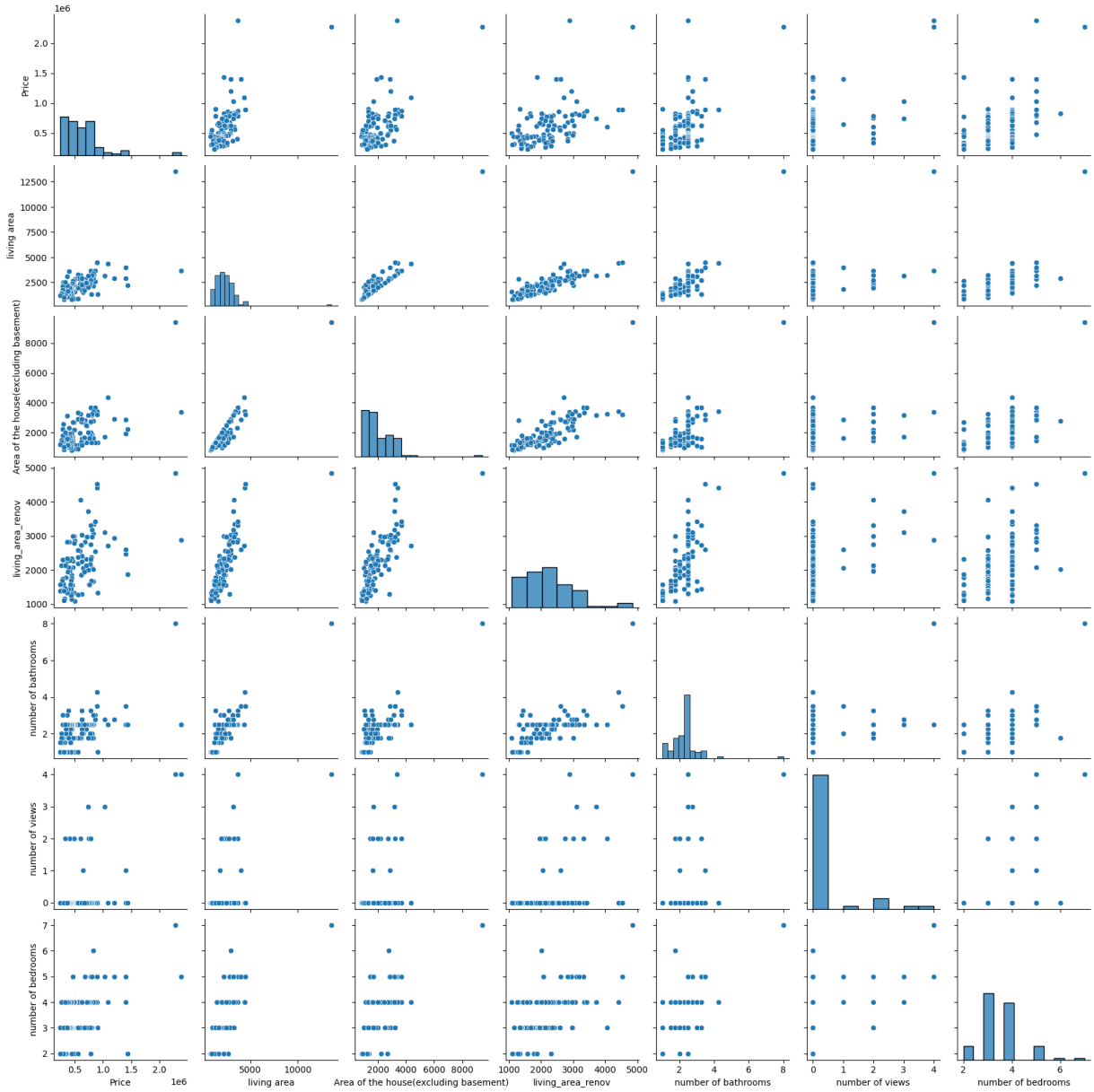


Multivariate Analysis

In [ ]:
```
sns.pairplot(df[col])
```

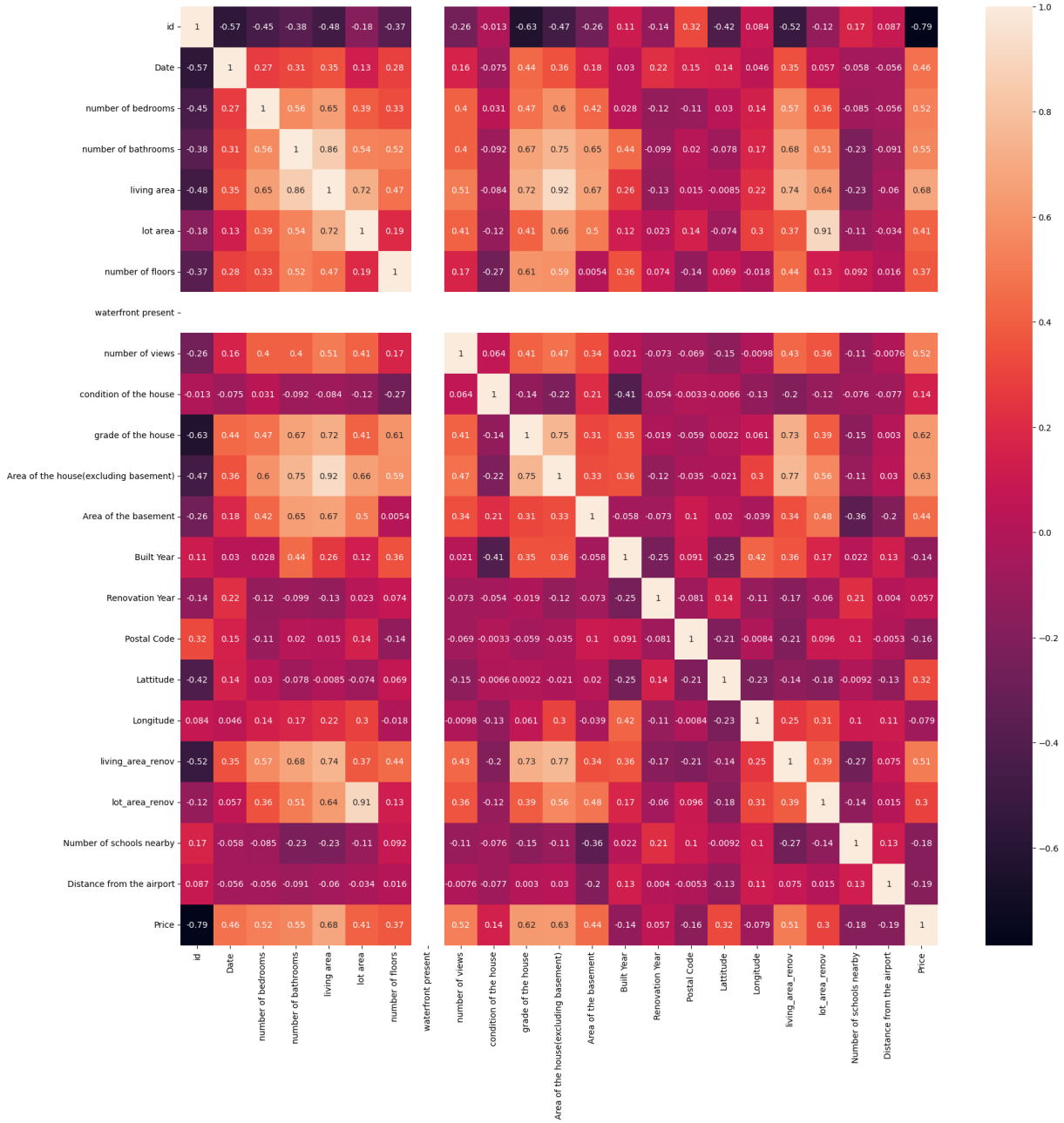
```
/home/godslayer/.local/lib/python3.11/site-packages/seaborn/axisgrid.py:118:
UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7f02fe0d7590>
```



In [ ]:
```
plt.figure(figsize=(20,20))
sns.heatmap(df.corr(numeric_only=True),annot=True)
```

Out[ ]: <Axes: >



Descriptive Statistics

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 23 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   id                                      100 non-null    int64
 1   Date                                    100 non-null    int64
 2   number of bedrooms                      100 non-null    int64
 3   number of bathrooms                     100 non-null    float64
 4   living area                             100 non-null    int64
 5   lot area                                100 non-null    int64
 6   number of floors                        100 non-null    float64
 7   waterfront present                      100 non-null    int64
 8   number of views                         100 non-null    int64
 9   condition of the house                  100 non-null    int64
 10  grade of the house                      100 non-null    int64
 11  Area of the house(excluding basement)   100 non-null    int64
 12  Area of the basement                    100 non-null    int64
 13  Built Year                              100 non-null    int64
 14  Renovation Year                         100 non-null    int64
 15  Postal Code                             100 non-null    int64
 16  Lattitude                               100 non-null    float64
 17  Longitude                               100 non-null    float64
 18  living_area_renov                       100 non-null    int64
 19  lot_area_renov                          100 non-null    int64
 20  Number of schools nearby                100 non-null    int64
 21  Distance from the airport               100 non-null    int64
 22  Price                                   100 non-null    int64
dtypes: float64(4), int64(19)
memory usage: 18.1 KB
```

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

Out[ ]:

	id	Date	number of bedrooms	number of bathrooms	living area	lot area
count	1.000000e+02	100.000000	100.000000	100.000000	100.00000	100.000000
mean	6.762819e+09	42491.860000	3.530000	2.290000	2389.90000	18152.190000
std	5.727200e+03	1.295057	0.892788	0.848885	1397.83707	37055.712743
min	6.762810e+09	42491.000000	2.000000	1.000000	800.00000	1180.000000
25%	6.762813e+09	42491.000000	3.000000	1.750000	1645.00000	5737.500000
50%	6.762817e+09	42491.000000	3.000000	2.500000	2205.00000	8459.000000
75%	6.762823e+09	42493.000000	4.000000	2.500000	2827.50000	11959.000000
max	6.762830e+09	42494.000000	7.000000	8.000000	13540.00000	307752.000000

8 rows × 23 columns

There is no null values in the data