HOUSE PRICE PREDICTION PROJECT:

In [ ]:

In [102]:

```
#Import required libraries:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#Ignore all warnings:
import warnings
warnings.filterwarnings("ignore")
```

In [103]:

```
#Import the dataset:
data = pd.read_csv("dataset.csv")
```

get a general information about the dataset:

In [104]:

```
print("No. of rows X columns = ",data.shape)
data.info()
round(data.describe(),2)
```

```
No. of rows X columns =  (14619, 23)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14619 entries, 0 to 14618
Data columns (total 23 columns):
 #   Column                                 Non-Null Count  Dtype  
---  ------                                 --------------  -----  
 0   id                                     14619 non-null  int64  
 1   Date                                   14619 non-null  int64  
 2   number of bedrooms                     14619 non-null  int64  
 3   number of bathrooms                    14619 non-null  float64
 4   living area                            14619 non-null  int64  
 5   lot area                               14619 non-null  int64  
 6   number of floors                       14619 non-null  float64
 7   waterfront present                     14619 non-null  int64  
 8   number of views                        14619 non-null  int64  
 9   condition of the house                 14619 non-null  int64  
 10  grade of the house                     14619 non-null  int64  
 11  Area of the house(excluding basement)  14619 non-null  int64  
 12  Area of the basement                   14619 non-null  int64  
 13  Built Year                             14619 non-null  int64  
 14  Renovation Year                        14619 non-null  int64  
 15  Postal Code                            14619 non-null  int64  
 16  Lattitude                              14619 non-null  float64
 17  Longitude                              14619 non-null  float64
 18  living_area_renov                      14619 non-null  int64  
 19  lot_area_renov                         14619 non-null  int64  
 20  Number of schools nearby               14619 non-null  int64  
 21  Distance from the airport              14619 non-null  int64  
 22  Price                                  14619 non-null  int64  
dtypes: float64(4), int64(19)
memory usage: 2.6 MB
```

Out[104]:

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	...	
count	1.461900e+04	14619.00	14619.00	14619.00	14619.00	14619.00	14619.00	14619.00	14619.00	14619.00	...	14
mean	6.762821e+09	42604.55	3.38	2.13	2098.16	15093.69	1.50	0.01	0.23	3.43	...	1
std	6.237160e+03	67.34	0.94	0.77	928.22	37920.89	0.54	0.09	0.77	0.66	...	
min	6.762810e+09	42491.00	1.00	0.50	370.00	520.00	1.00	0.00	0.00	1.00	...	1
25%	6.762815e+09	42546.00	3.00	1.75	1440.00	5010.50	1.00	0.00	0.00	3.00	...	1
50%	6.762821e+09	42600.00	3.00	2.25	1930.00	7620.00	1.50	0.00	0.00	3.00	...	1
75%	6.762826e+09	42662.00	4.00	2.50	2570.00	10800.00	2.00	0.00	0.00	4.00	...	1
max	6.762832e+09	42734.00	33.00	8.00	13540.00	1074218.00	3.50	1.00	4.00	5.00	...	2

8 rows × 23 columns

In [105]:

```python
print("Sample 5 entries of data: ")
data.sample(5)
```

```
Sample 5 entries of data:
```

Out[105]:

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	...	Built Year	Renovation Year
3338	6762826256	42543	4	1.50	1890	43560	1.0	0	0	4	...	1974	0
7456	6762818993	42602	3	2.00	1500	2500	2.0	0	0	3	...	2002	0
10164	6762830891	42648	3	1.00	1090	17630	1.0	0	0	4	...	1962	0
12782	6762813696	42693	4	2.50	2370	2971	2.0	0	2	3	...	2008	0
10668	6762813432	42657	4	2.75	2660	4500	1.5	0	0	5	...	1909	0

5 rows × 23 columns

In [106]:

```python
print("First 5 entries of data: ")
data.head()
```

```
First 5 entries of data:
```

Out[106]:

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	...	Built Year	Renovation Year	Po C
0	6762810635	42491	4	2.50	2920	4000	1.5	0	0	5	...	1909	0	122
1	6762810998	42491	5	2.75	2910	9480	1.5	0	0	3	...	1939	0	122
2	6762812605	42491	4	2.50	3310	42998	2.0	0	0	3	...	2001	0	122
3	6762812919	42491	3	2.00	2710	4500	1.5	0	0	4	...	1929	0	122
4	6762813105	42491	3	2.50	2600	4750	1.0	0	0	4	...	1951	0	122

5 rows × 23 columns

In [107]:

```python
print("Last 5 entries of data: ")
data.tail()
```

```
Last 5 entries of data:
```

Out[107]:

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	...	Built Year	Renovation Year
14614	6762830250	42734	2	1.5	1556	20000	1.0	0	0	4	...	1957	0
14615	6762830339	42734	3	2.0	1680	7000	1.5	0	0	4	...	1968	0
14616	6762830618	42734	2	1.0	1070	6120	1.0	0	0	3	...	1962	0
14617	6762830709	42734	4	1.0	1030	6621	1.0	0	0	4	...	1955	0
14618	6762831463	42734	3	1.0	900	4770	1.0	0	0	3	...	1969	2009

5 rows × 23 columns

In [108]:

```python
#Check for any null-values in each column:
result = data.isna().sum().reset_index()
result.columns=["Column-name","No. of null values"]
print(result)

print("\n\n\n")

#check for any duplicated values:
result = data.duplicated().sum()
print("No. of duplicated values = ",result)
```

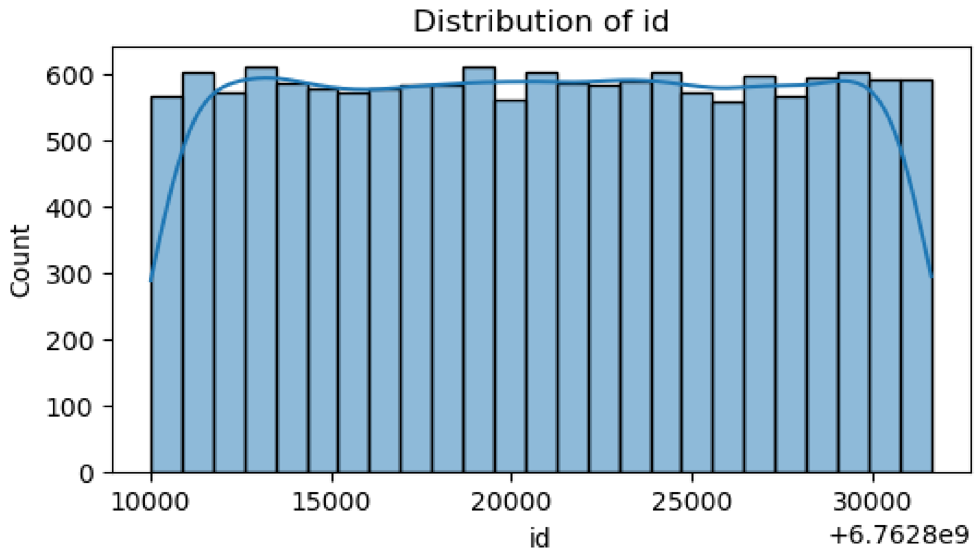
```
                           Column-name  No. of null values
0                                   id                   0
1                                 Date                   0
2                   number of bedrooms                   0
3                  number of bathrooms                   0
4                          living area                   0
5                             lot area                   0
6                     number of floors                   0
7                   waterfront present                   0
8                      number of views                   0
9               condition of the house                   0
10                  grade of the house                   0
11  Area of the house(excluding basement)                   0
12                 Area of the basement                   0
13                          Built Year                   0
14                     Renovation Year                   0
15                         Postal Code                   0
16                           Lattitude                   0
17                           Longitude                   0
18                   living_area_renov                   0
19                      lot_area_renov                   0
20             Number of schools nearby                   0
21            Distance from the airport                   0
22                               Price                   0




No. of duplicated values =  0
```

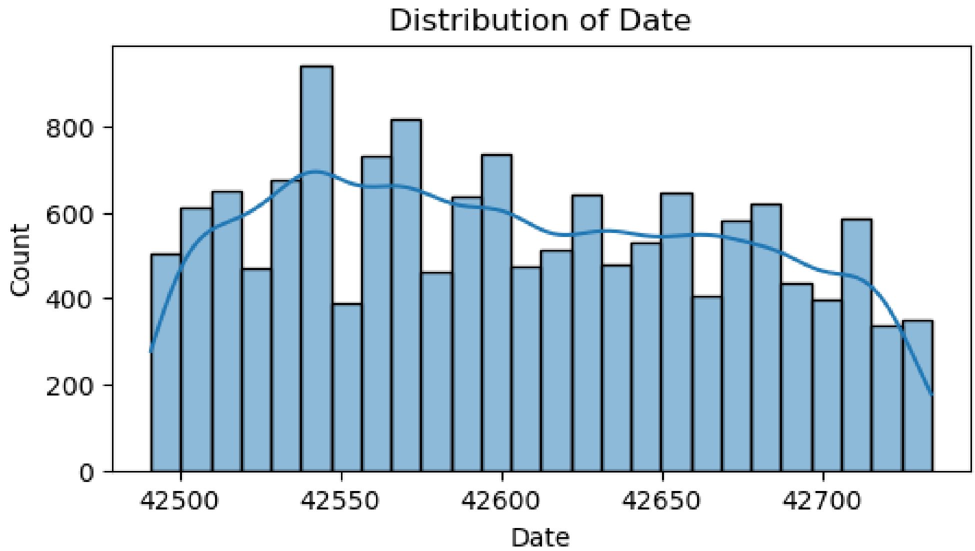
In [ ]:

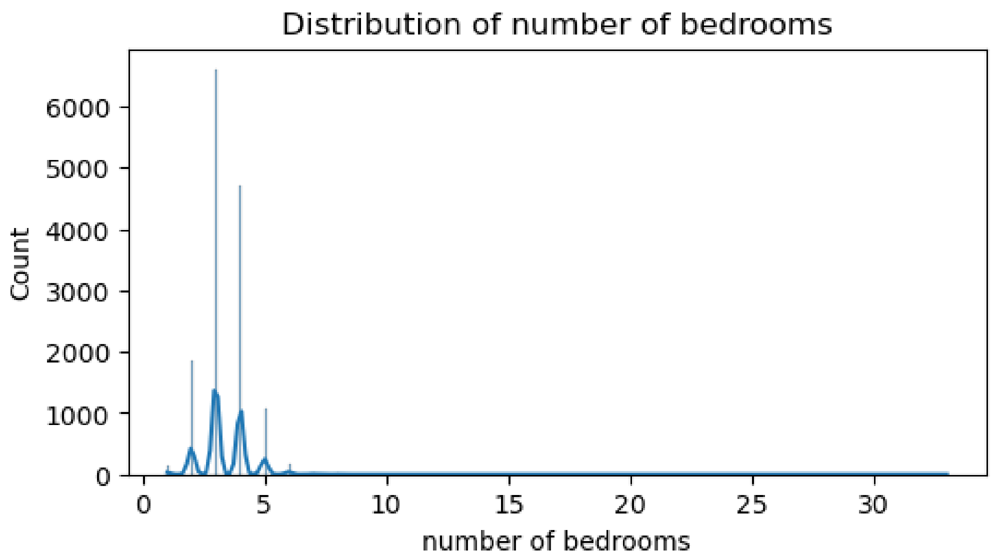
EDA:

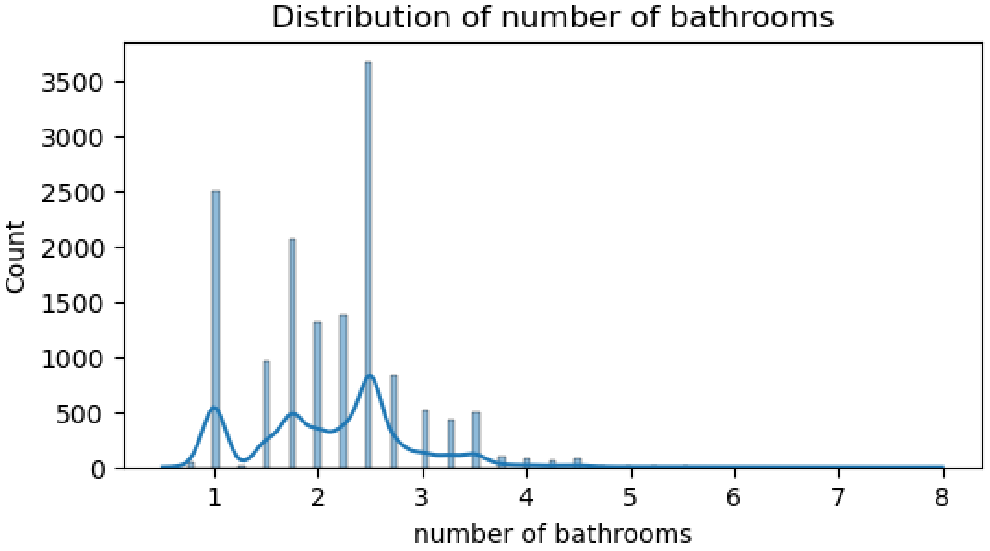
In [109]:

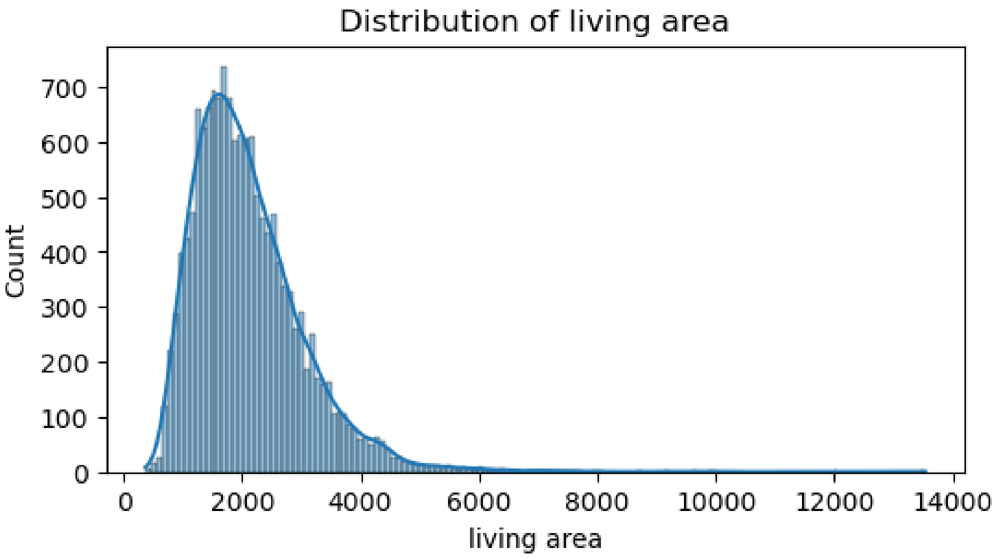
```
numerical_cols = data.select_dtypes(include=['int64', 'float64']).columns

for col in numerical_cols:
    plt.figure(figsize=(6,3))
    sns.histplot(data[col], kde=True)
    plt.title(f"Distribution of {col}")
    plt.show()
```

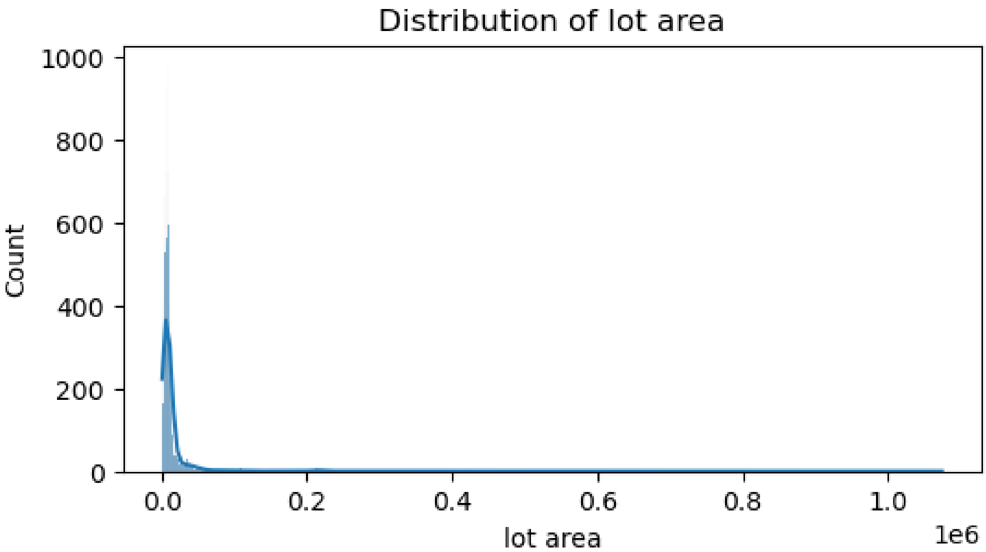


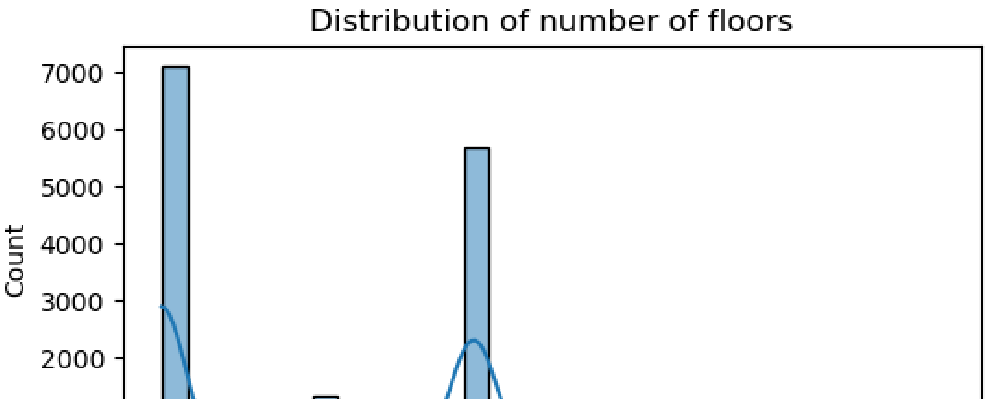


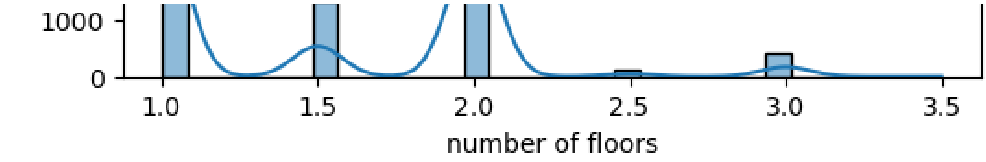


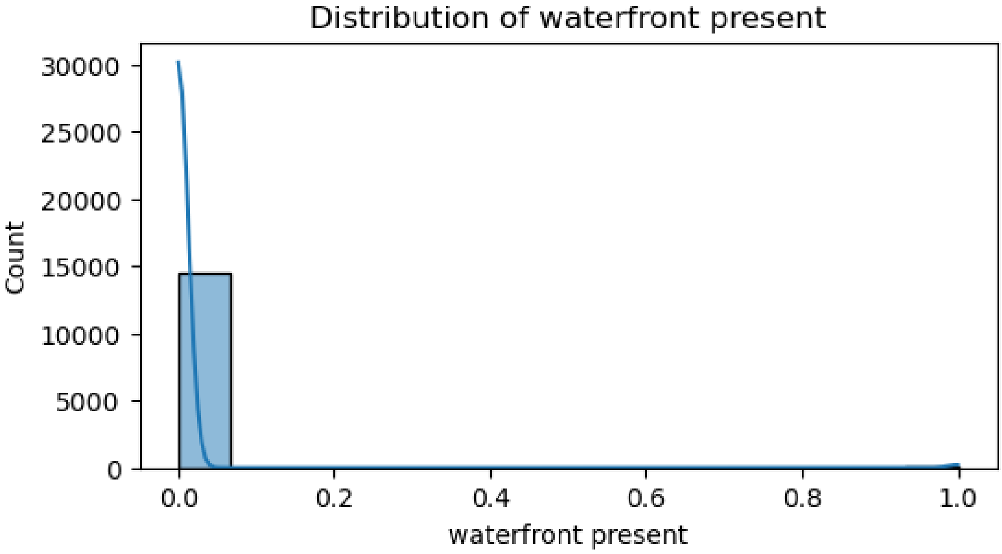

Distribution of number of bathrooms
Count
0
500
1000
1500
2000
2500
3000
3500
1
2
3
4
5
6
7
8
number of bathrooms

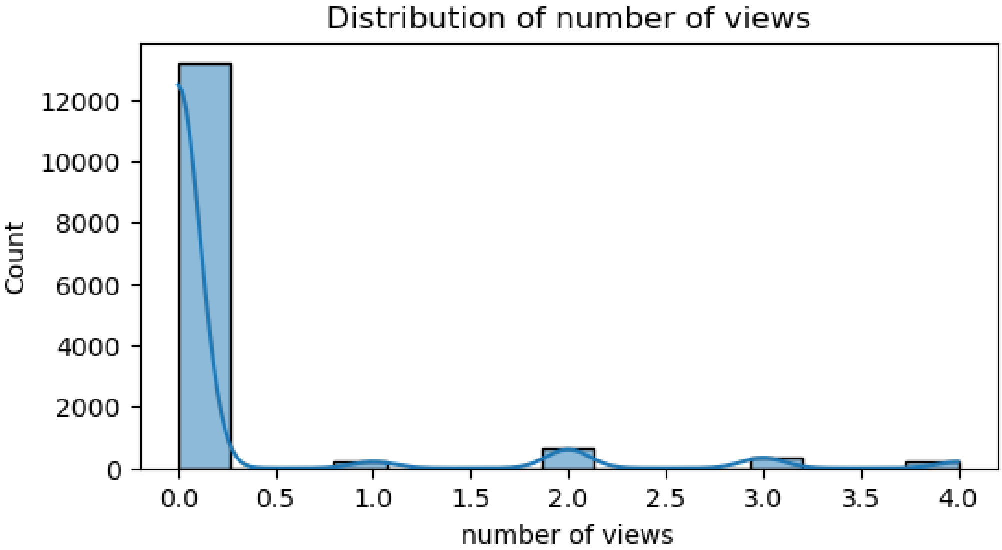

Distribution of living area
Count
0
100
200
300
400
500
600
700
0
2000
4000
6000
8000
10000
12000
14000
living area

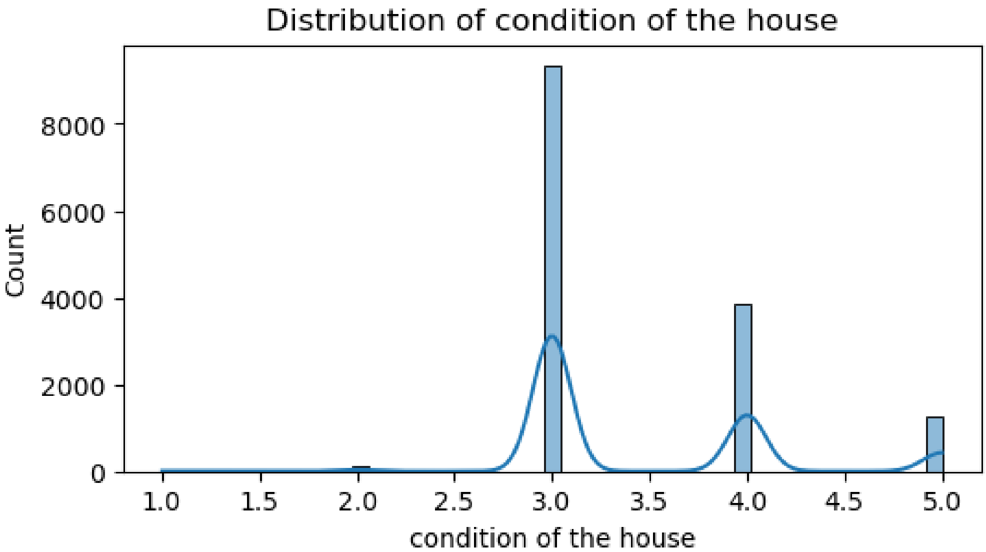

Distribution of lot area
Count
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0
1e6
lot area

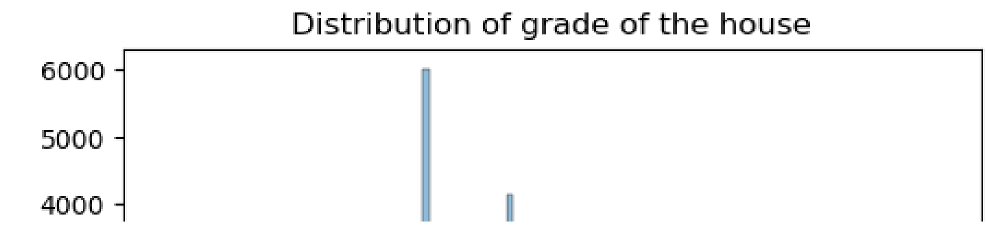

Distribution of number of floors
Count
2000
3000
4000
5000
6000
7000

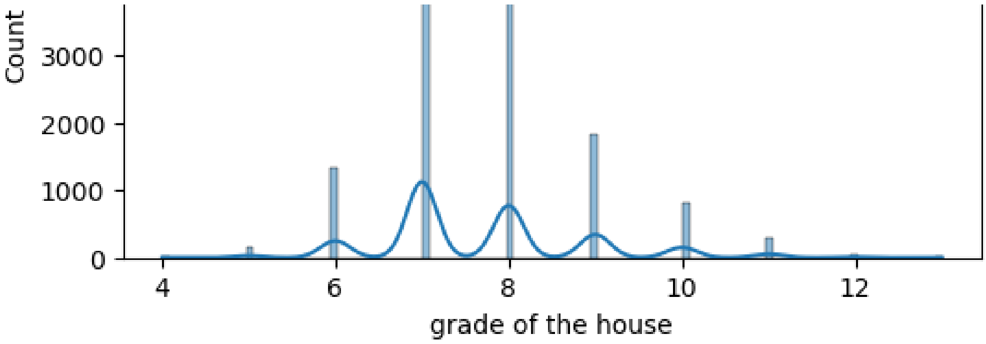

1000
0
1.0
1.5
2.0
2.5
3.0
3.5
number of floors


Distribution of waterfront present
Count
30000
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0
waterfront present

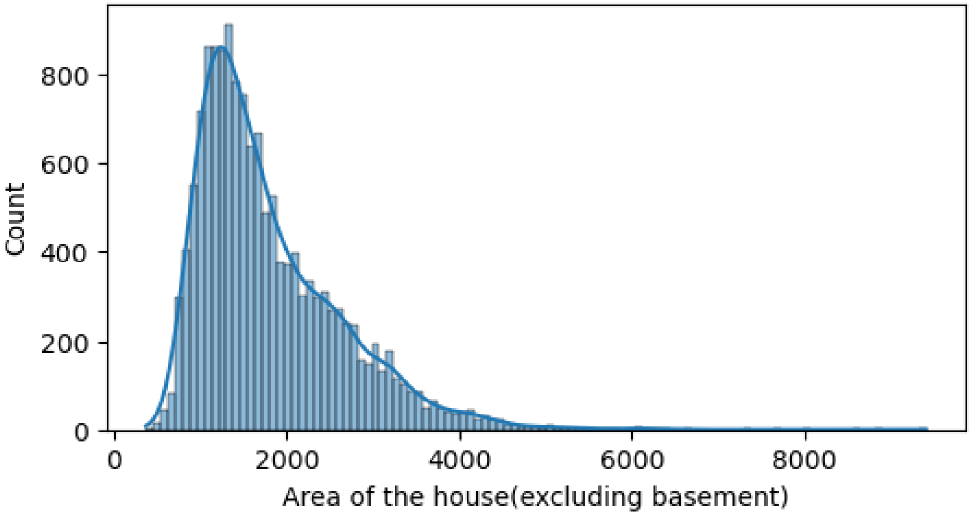

Distribution of number of views
Count
12000
10000
8000
6000
4000
2000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
number of views


Distribution of condition of the house
Count
8000
6000
4000
2000
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
condition of the house

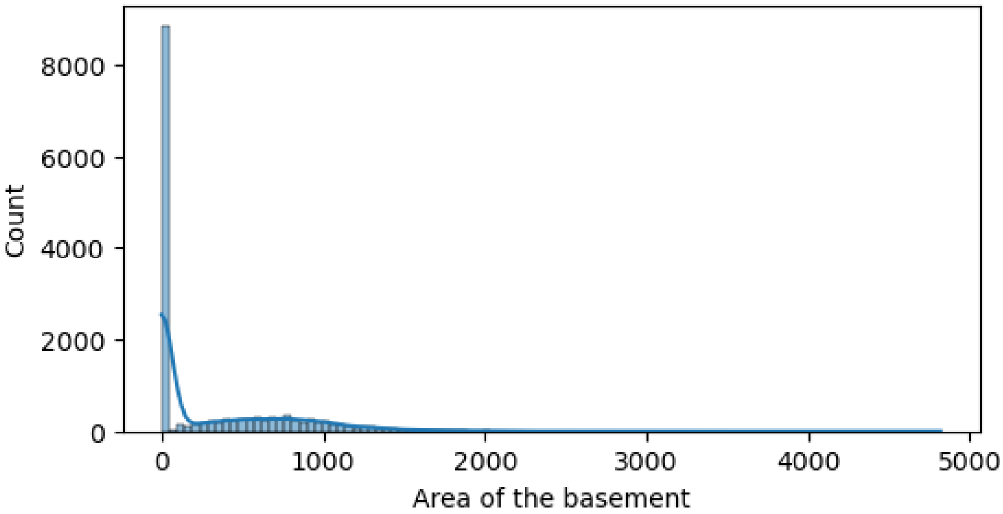

Distribution of grade of the house
6000
5000
4000



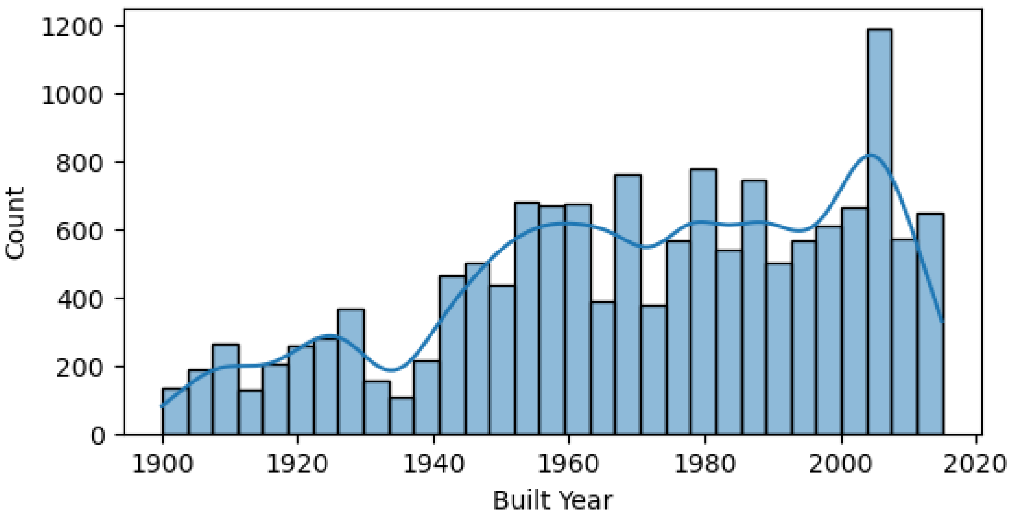
Distribution of Area of the house(excluding basement)



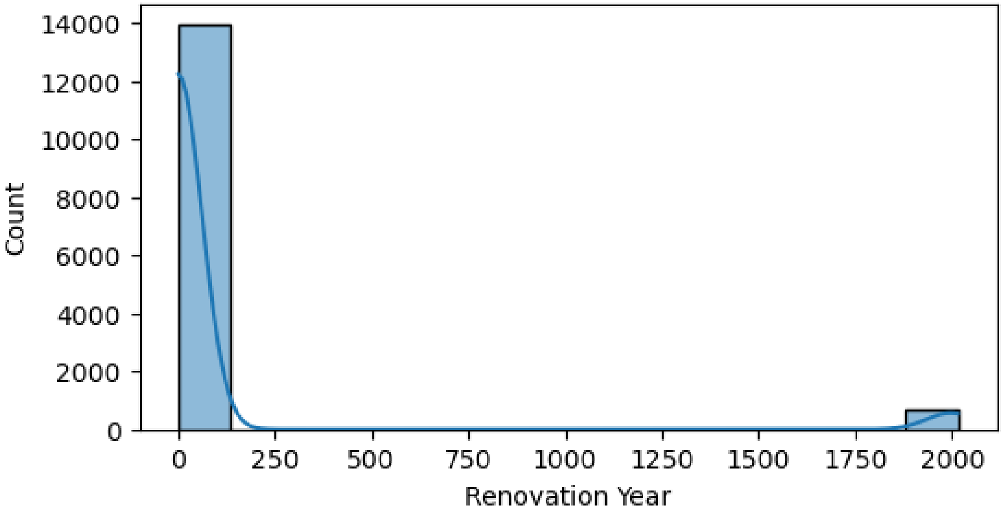
Distribution of Area of the basement



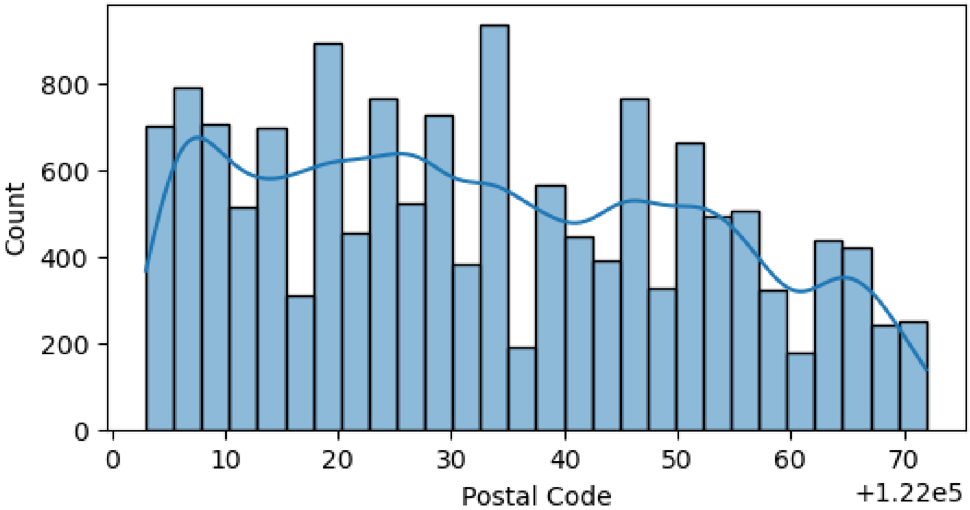
Distribution of Built Year



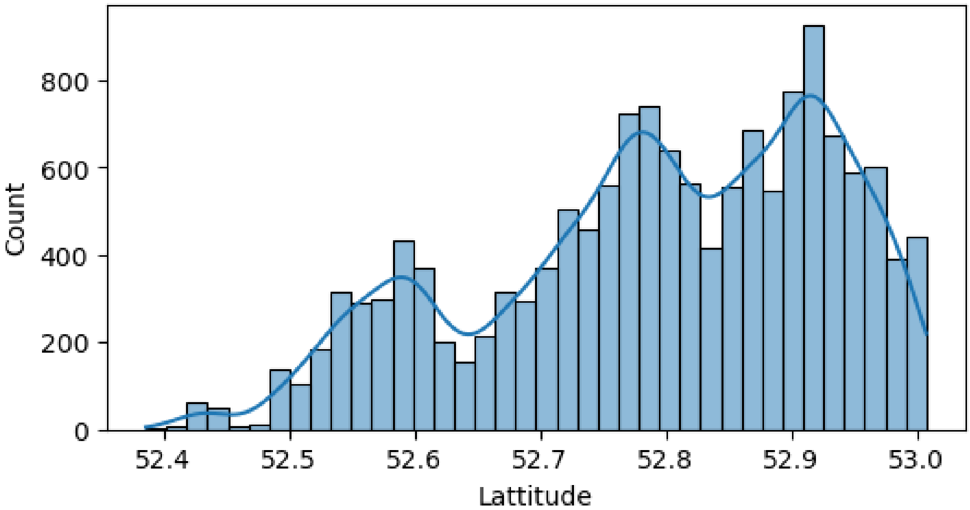
Distribution of Renovation Year



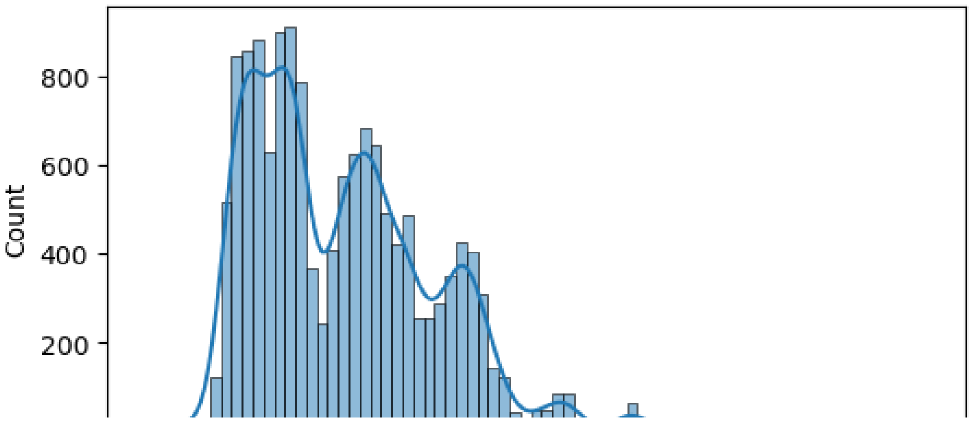
Distribution of Postal Code

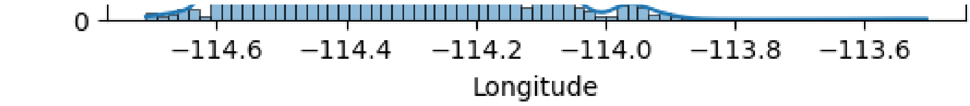


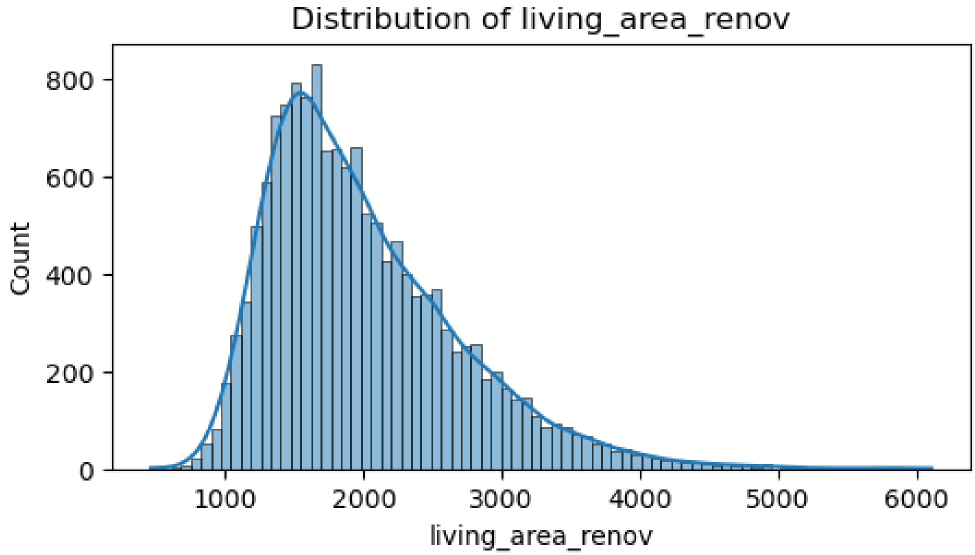
Distribution of Lattitude

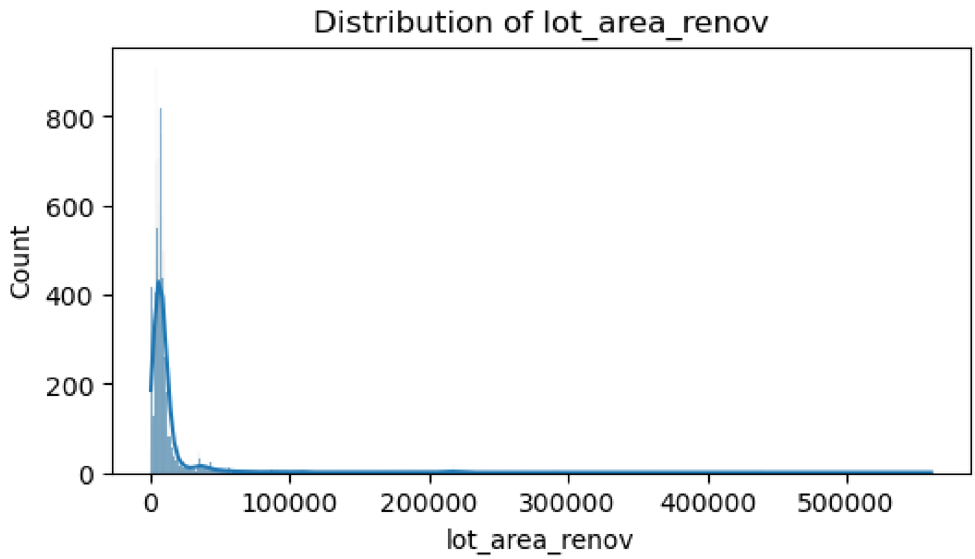


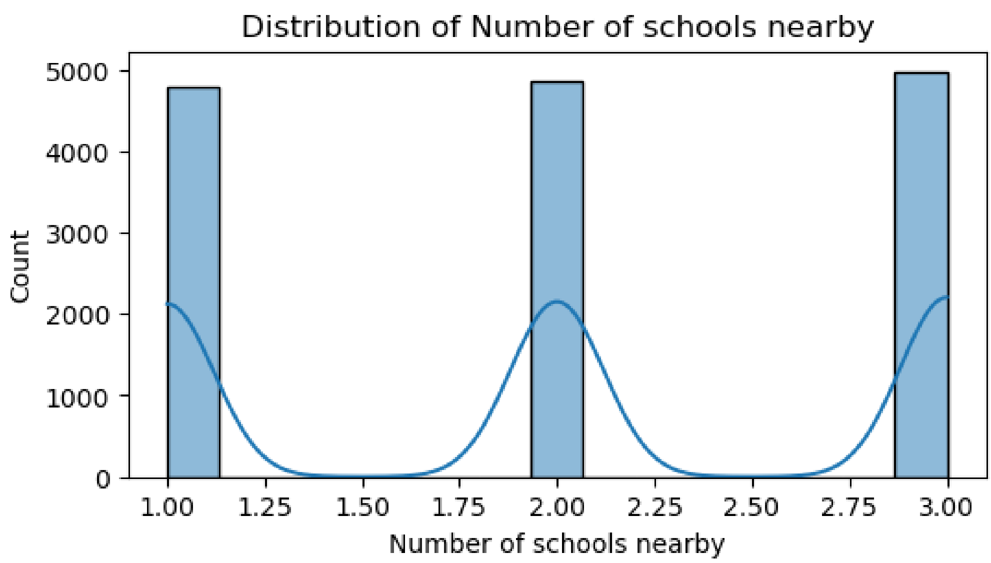
Distribution of Longitude

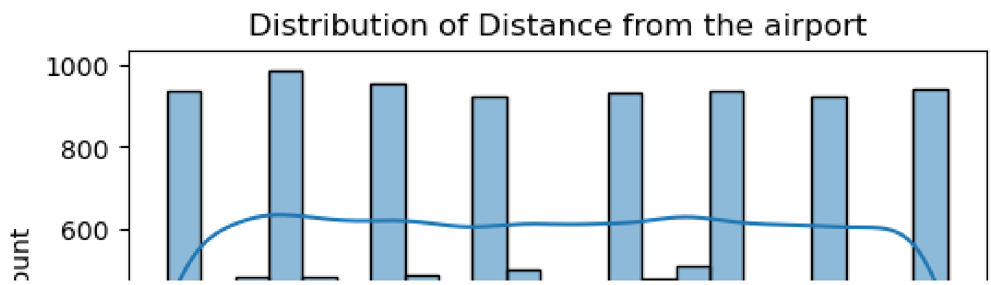


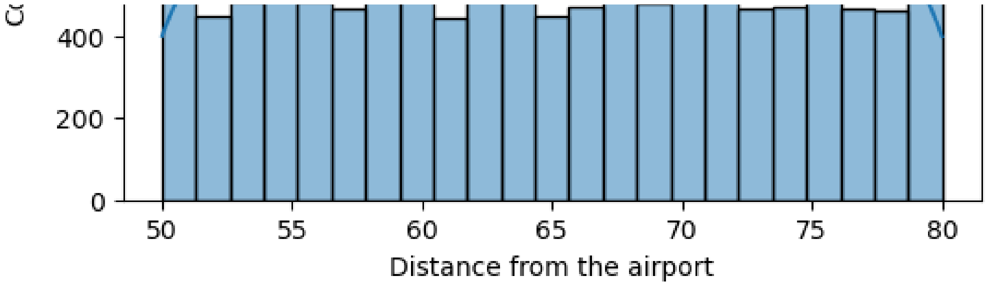

0
−114.6
−114.4
−114.2
−114.0
−113.8
−113.6
Longitude

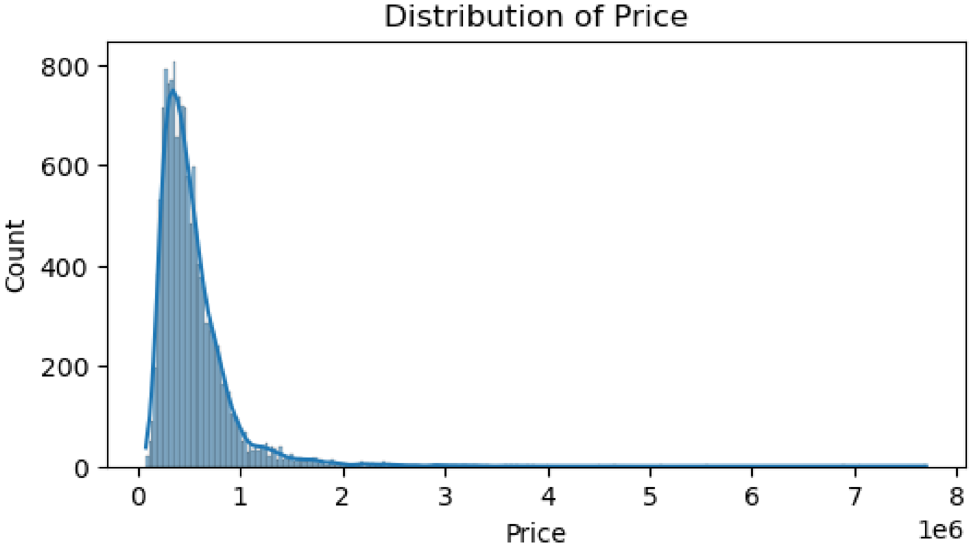

Distribution of living_area_renov
Count
0
200
400
600
800
1000
2000
3000
4000
5000
6000
living_area_renov


Distribution of lot_area_renov
Count
0
200
400
600
800
0
100000
200000
300000
400000
500000
lot_area_renov


Distribution of Number of schools nearby
Count
0
1000
2000
3000
4000
5000
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
Number of schools nearby

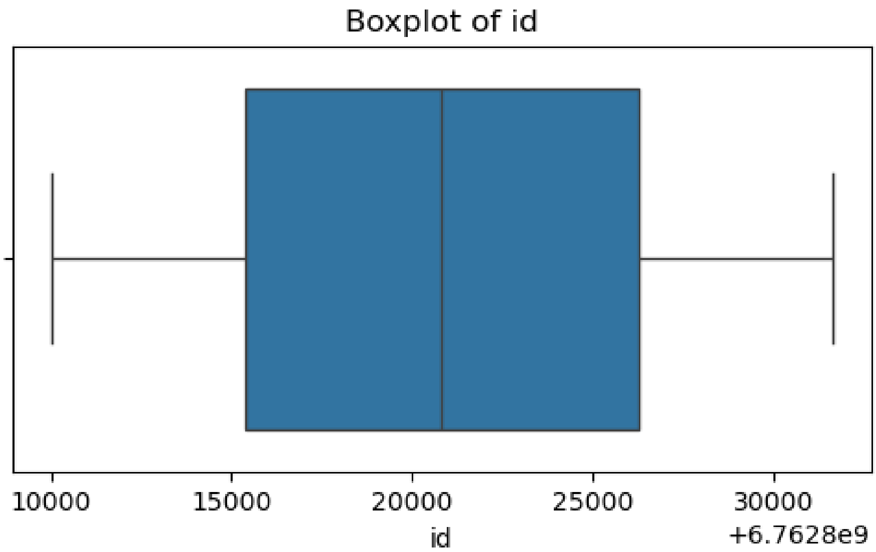

Distribution of Distance from the airport
600
800
1000

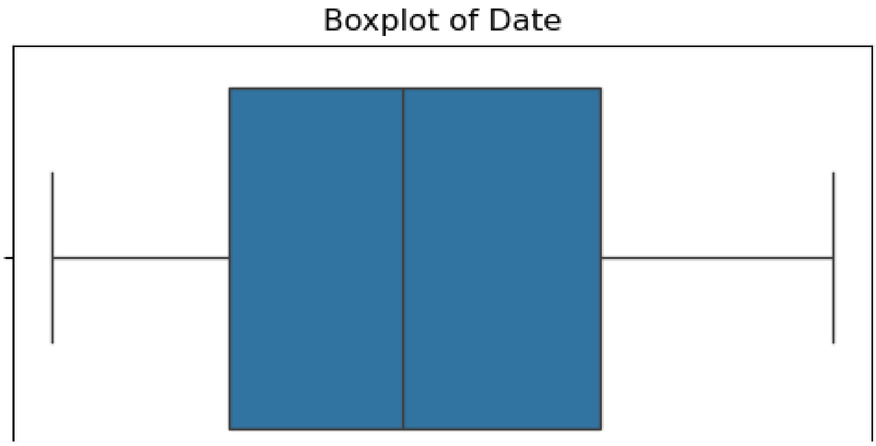




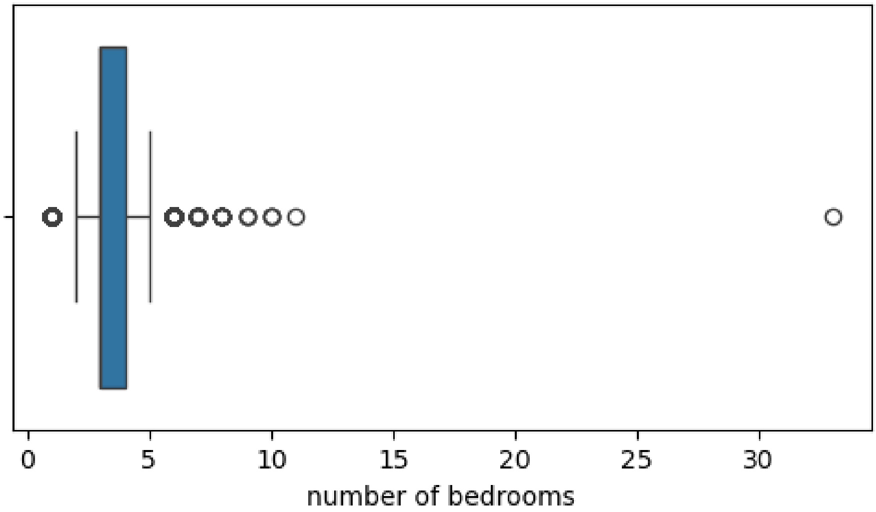
In [110]:

```
for col in numerical_cols:
    plt.figure(figsize=(6,3))
    sns.boxplot(x=data[col])
    plt.title(f"Boxplot of {col}")
    plt.show()
```

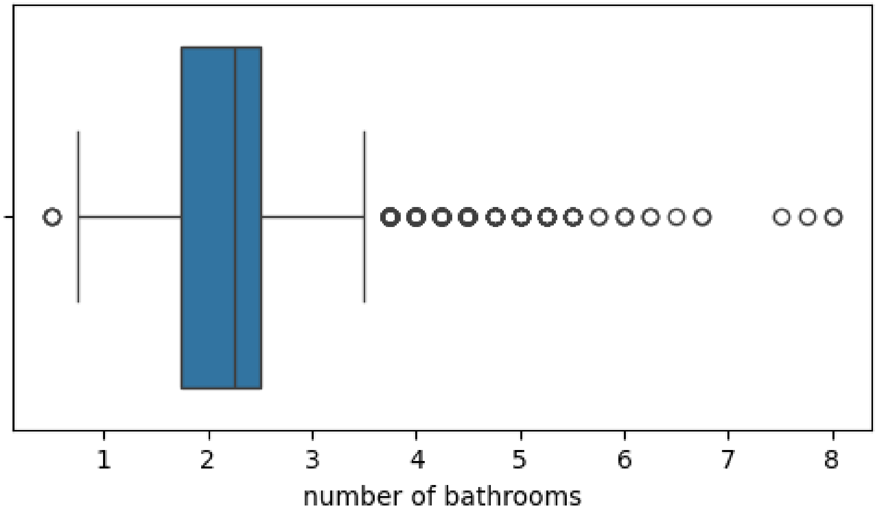




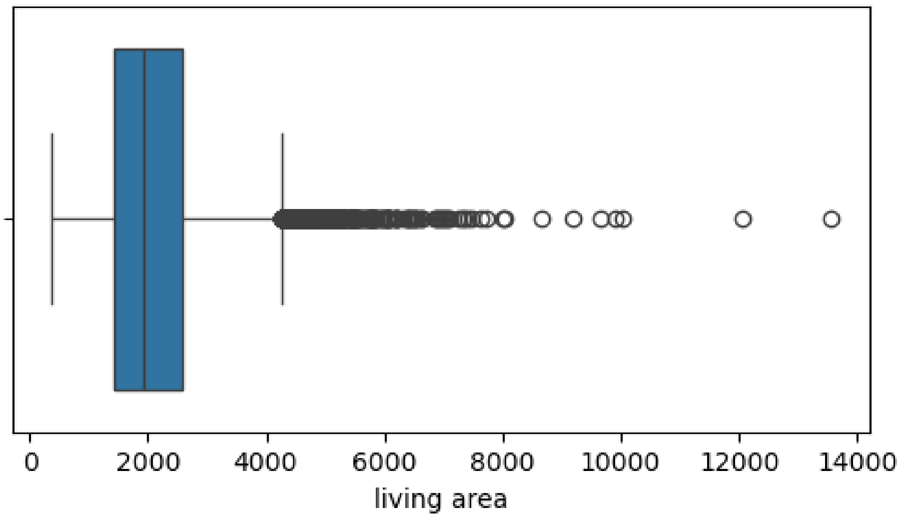
Boxplot of number of bedrooms



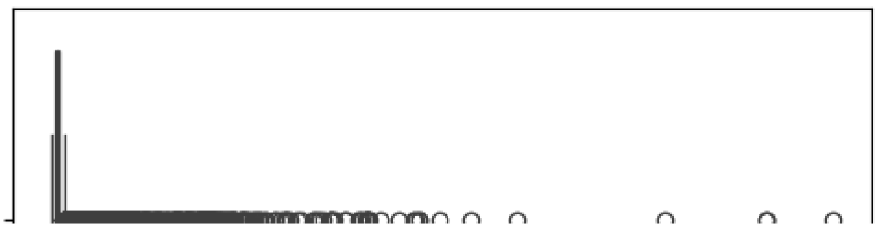
Boxplot of number of bathrooms

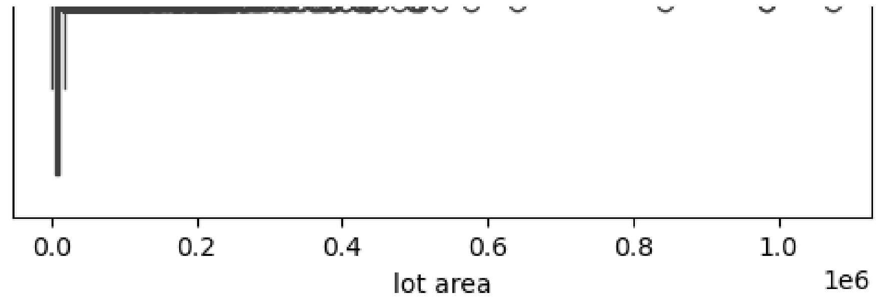


Boxplot of living area

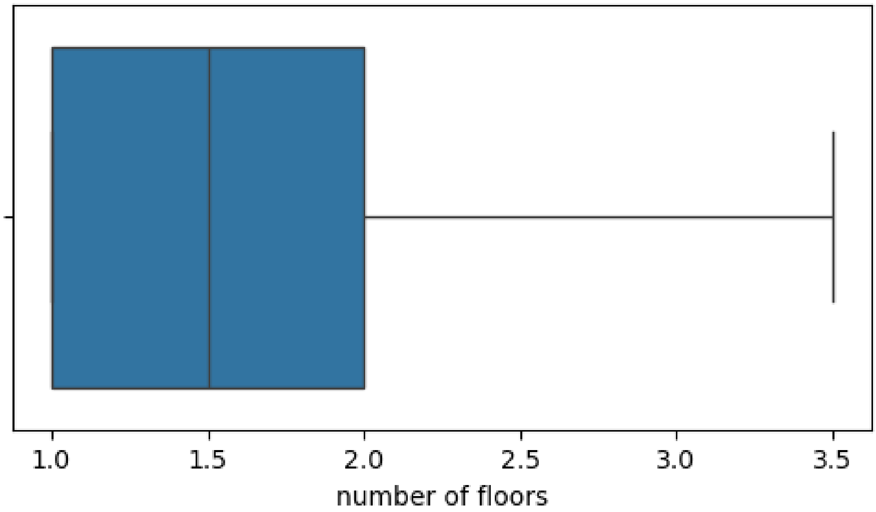


Boxplot of lot area

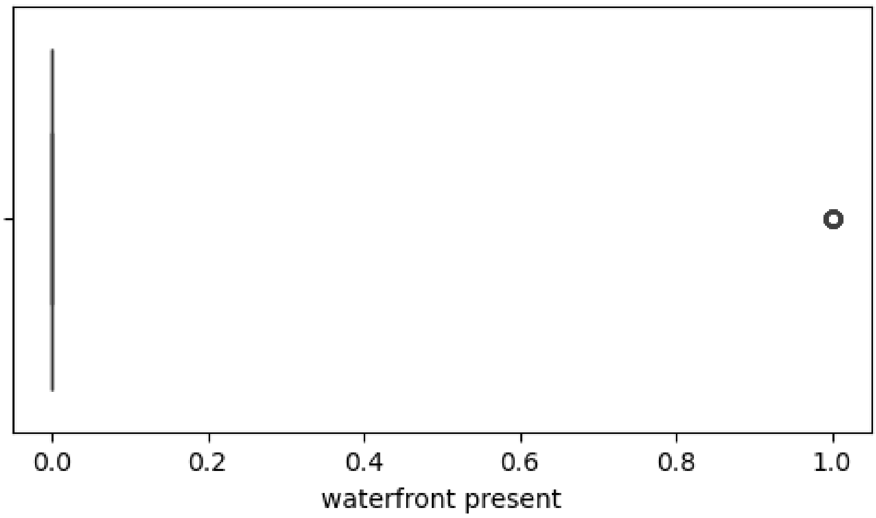




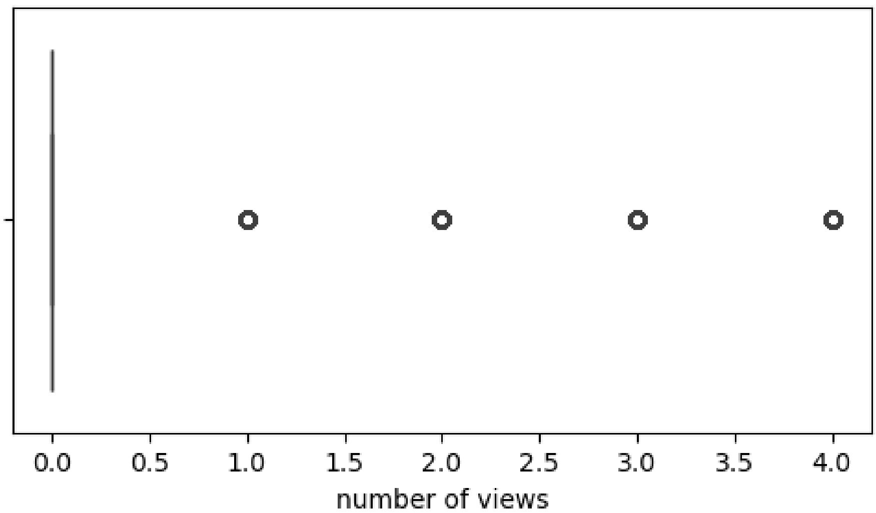
Boxplot of number of floors



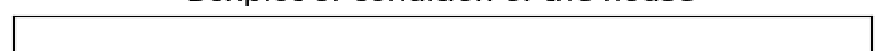
Boxplot of waterfront present

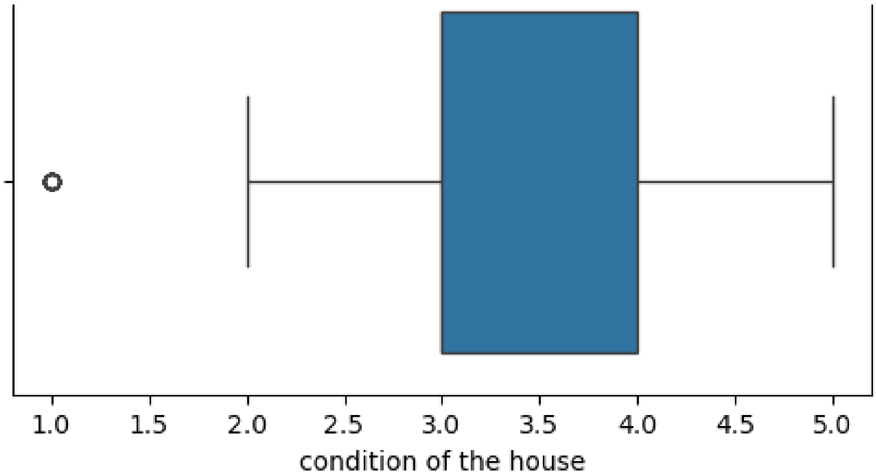


Boxplot of number of views

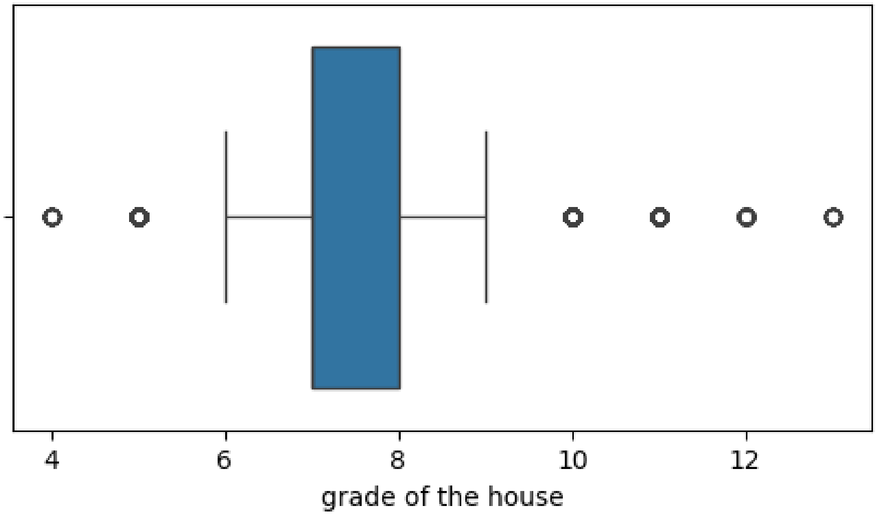


Boxplot of condition of the house

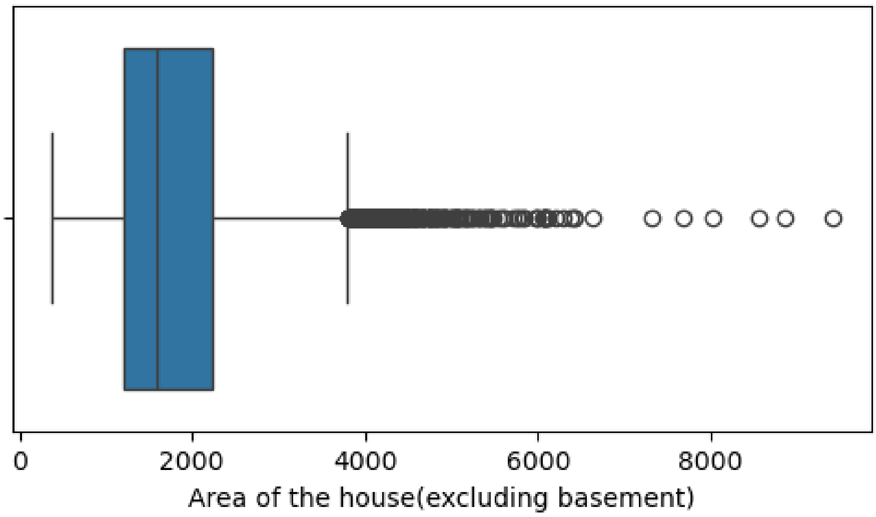




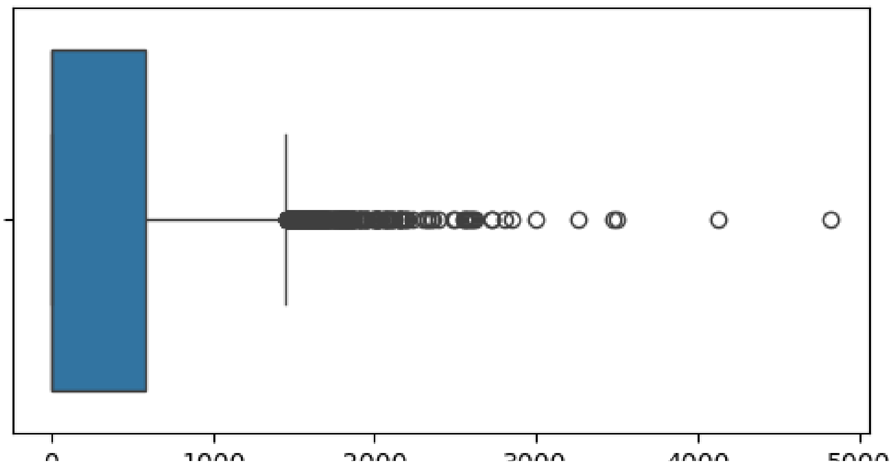
Boxplot of grade of the house

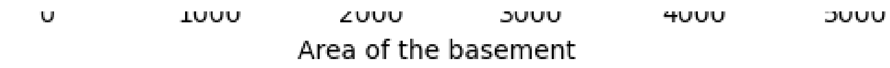


Boxplot of Area of the house(excluding basement)

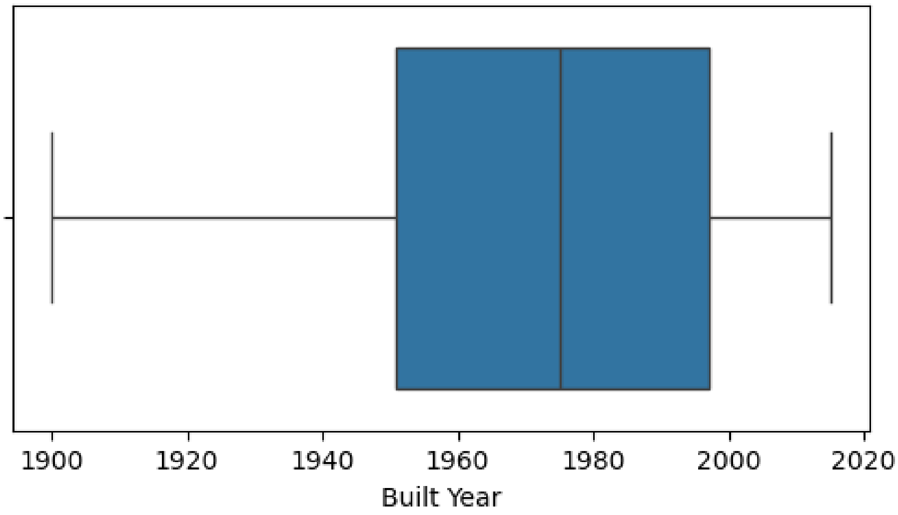


Boxplot of Area of the basement

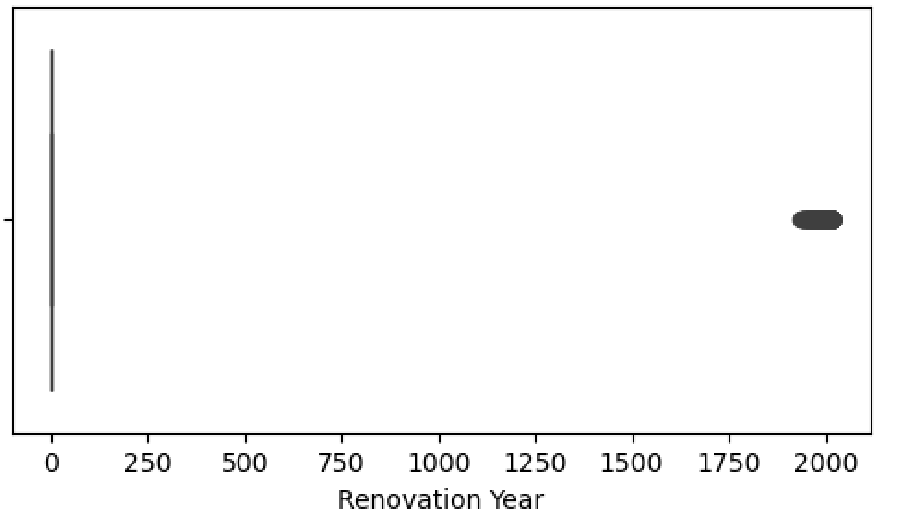




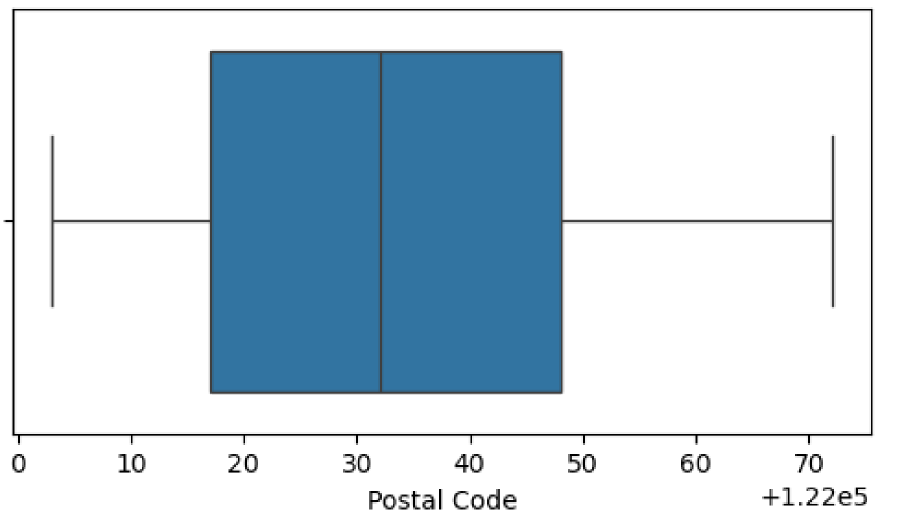
Boxplot of Built Year



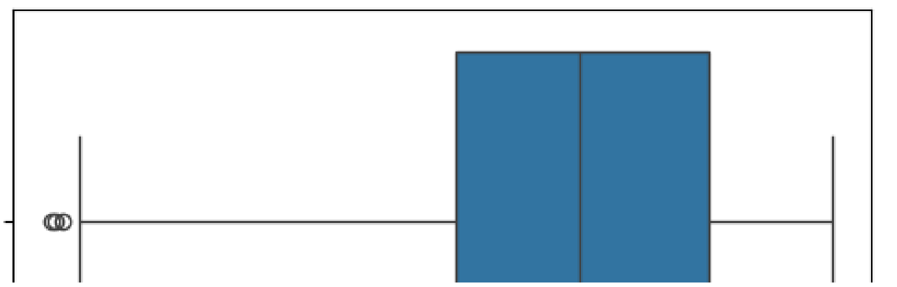
Boxplot of Renovation Year

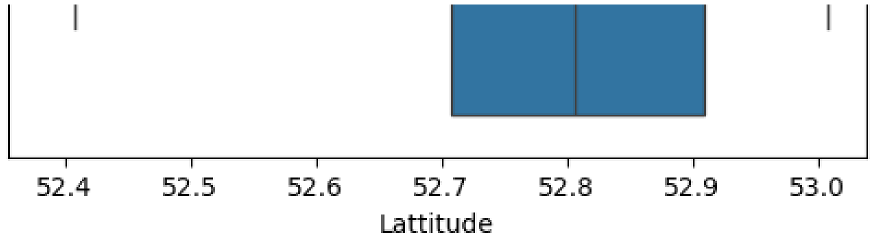


Boxplot of Postal Code

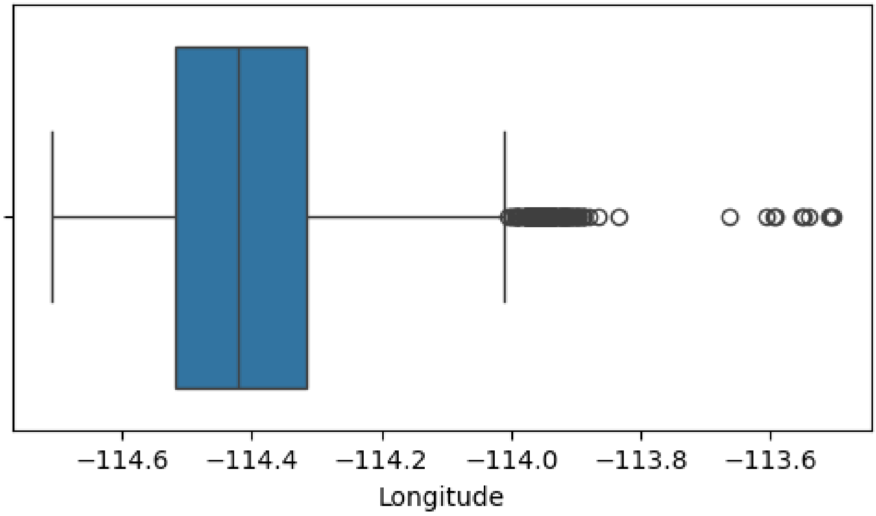


Boxplot of Lattitude

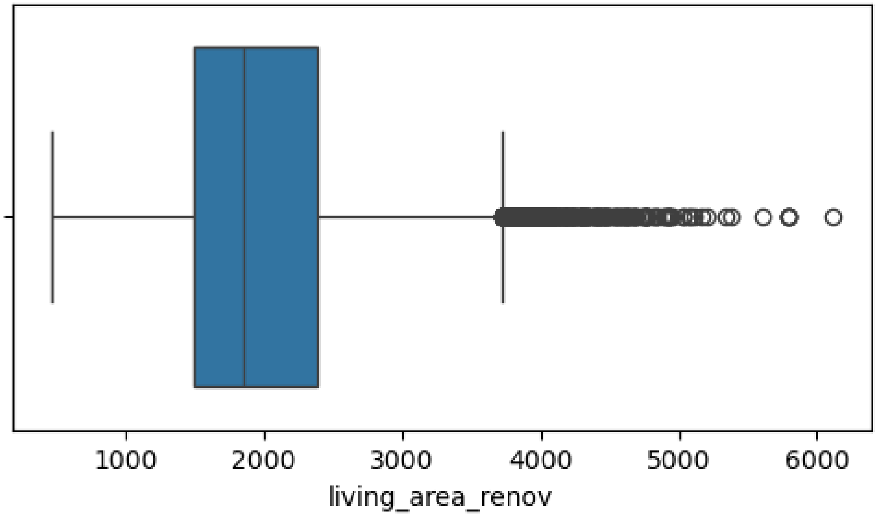




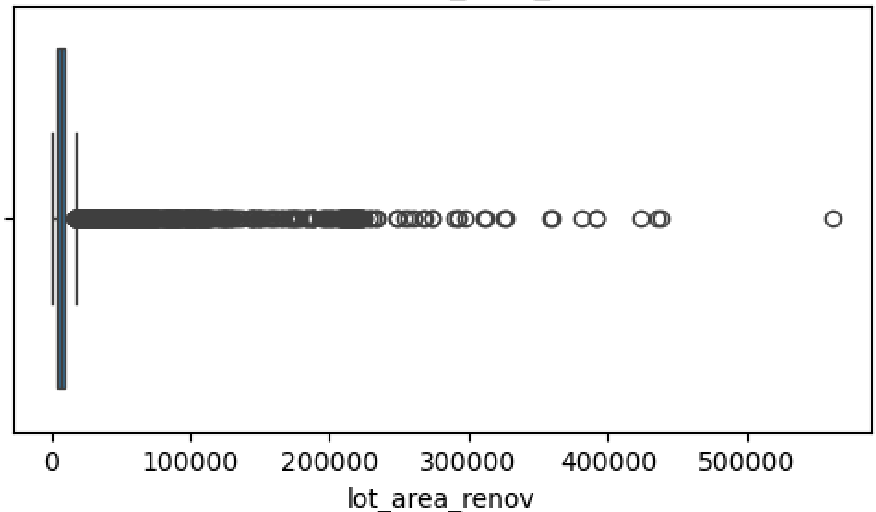
Boxplot of Longitude



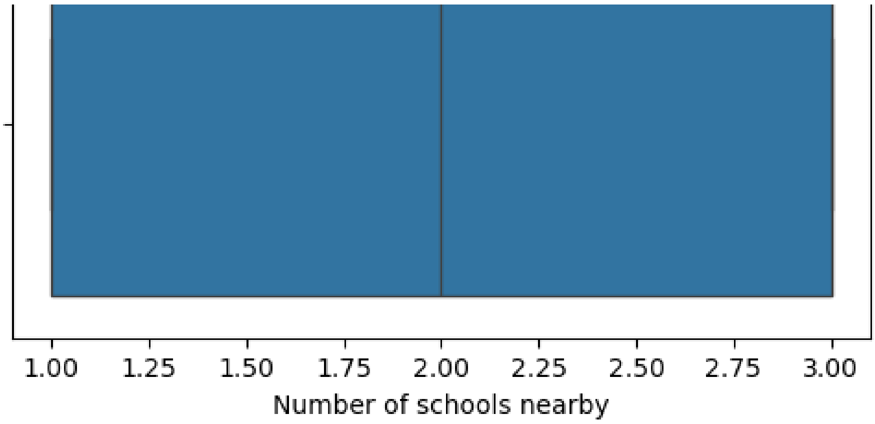
Boxplot of living_area_renov



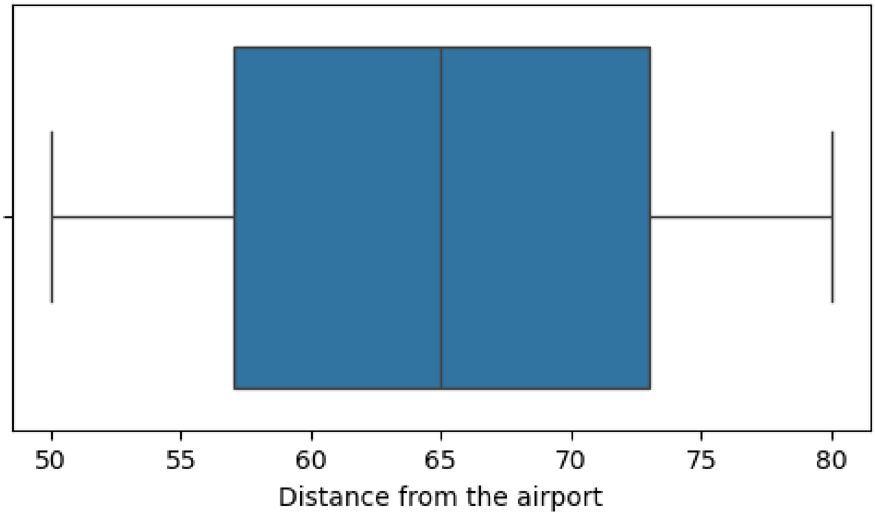
Boxplot of lot_area_renov



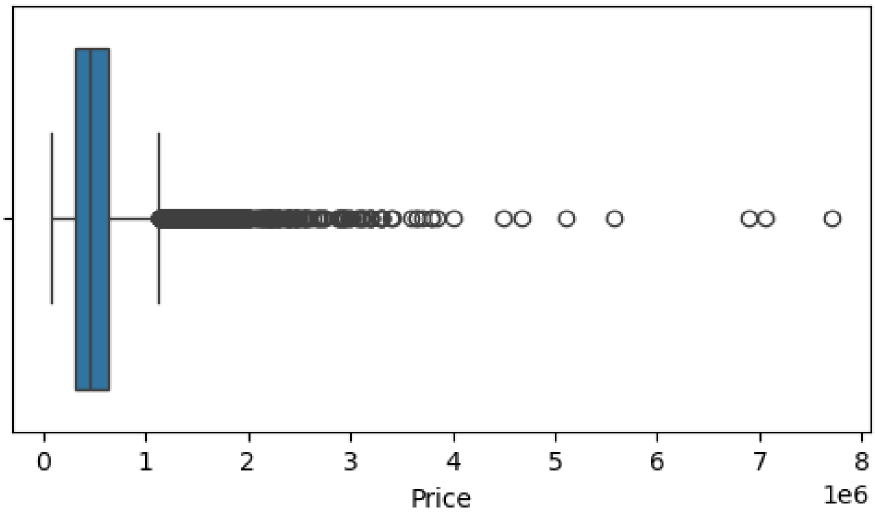
Boxplot of Number of schools nearby



Boxplot of Distance from the airport



Boxplot of Price



Now check the relationship between:

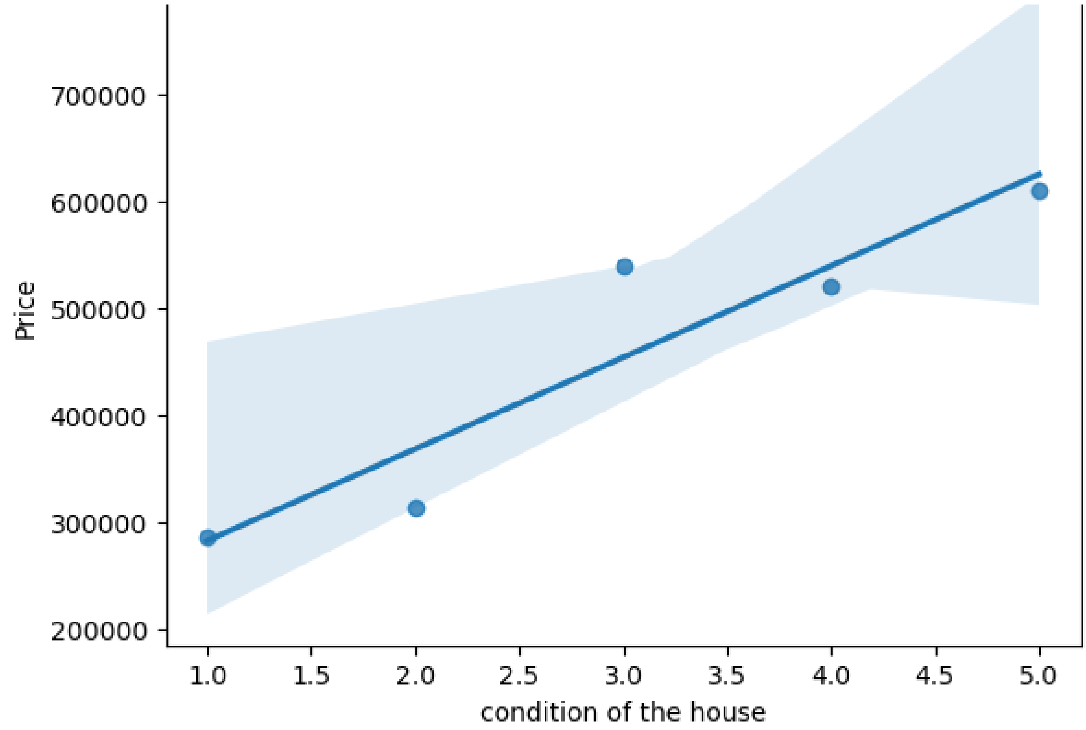
- **Condition of the house**
- **Price of the house (on average)**

In [111]:

```
df = data.groupby("condition of the house")["Price"].mean().reset_index().sort_values(by
="Price")
sns.regplot(x="condition of the house",y="Price",data=df)
```

Out[111]:

<Axes: xlabel='condition of the house', ylabel='Price'>



_conclusion at this point:

- **Houses in proper condition are costly.**
- **Whereas; houses (not in proper condition) are cheap.**

In [112]:

```
data.head()
```

Out[112]:

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	...	Built Year	Renovation Year	Po C
0	6762810635	42491	4	2.50	2920	4000	1.5	0	0	5	...	1909	0	122
1	6762810998	42491	5	2.75	2910	9480	1.5	0	0	3	...	1939	0	122
2	6762812605	42491	4	2.50	3310	42998	2.0	0	0	3	...	2001	0	122
3	6762812919	42491	3	2.00	2710	4500	1.5	0	0	4	...	1929	0	122
4	6762813105	42491	3	2.50	2600	4750	1.0	0	0	4	...	1951	0	122

5 rows × 23 columns

ML QUESTIONS:

Q1] Predict house-prices based on:

- **Number of bedrooms**
- **Numbers of bathrooms**
- **Living area**
- **Condition of the house**

In [113]:

```
X = data[["number of bedrooms","number of bathrooms","living area","condition of the hous
e"]]
```

```
y = data[["Price"]]
```

In [114]:

```
from sklearn.model_selection import train_test_split, GridSearchCV
X_train , X_test , y_train, y_test = train_test_split(X,y,test_size=0.2)

from sklearn.tree import DecisionTreeRegressor
param_grid = {
    "criterion" : ["squared_error","friedman_mse","absolute_error"],
    "splitter" : ["best","random"],
    "max_depth": [None, 10,20,30,40,50],
    "min_samples_split": [2,5,10],
    "min_samples_leaf": [1,2,4]
}

tree_model = DecisionTreeRegressor()
grid_tree= GridSearchCV(estimator = tree_model, param_grid = param_grid)
grid_tree.fit(X_train, y_train)
```

Out[114]:

```
▸ GridSearchCV                     i ?

▸ best_estimator_:
  DecisionTreeRegressor

  ▸ DecisionTreeRegressor          ?
```

In [115]:

```
#Check the performance of this model:

##but first make the predictions for the testing-dataset:
tree_preds = grid_tree.predict(X_test)
print(tree_preds)

from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, tree_preds)
```

```
[355495.67315175 522026.90358744 238611.75555556 ... 353295.83333333
 539271.16622691 421135.27925532]
```

Out[115]:

```
61623684399.663
```

In [116]:

```
###Now make the predictions and check the performance of the linear-regression model:
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)


predslr = lr.predict(X_test) #predictions for the testing data using linear-regression
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, predslr)
```

Out[116]:

```
60576794781.62602
```

In [117]:

```
### Now for both the models check the mean_absolute_error values:
from sklearn.metrics import mean_absolute_error
print("For GridSearchCV: ",mean_absolute_error(y_test, tree_preds))
print("For LinearRegression: ",mean_absolute_error(y_test, predslr))
```

```
For GridSearchCV:  161548.44850459037
For LinearRegression:  165031.71973815977
```

In [118]:

```
#Now make the predictions using Random-forest model:
from sklearn.ensemble import RandomForestRegressor
rfrmodel = RandomForestRegressor()
param_gridrfr = {
    "max_depth": [5,10,15],
    "n_estimators":[2,3,4,5,6,7,8,9,10]
}

gridrfr = GridSearchCV(rfrmodel,param_gridrfr)
gridrfr.fit(X_train, y_train)
```

Out[118]:

```
▸ GridSearchCV
                                        i ?
▸ best_estimator_:
  RandomForestRegressor

  ▸ RandomForestRegressor
                                        ?
```

In [119]:

```
rfrpredictions = gridrfr.predict(X_test)
mean_absolute_error(y_test,rfrpredictions)
```

Out[119]:

```
157897.00309501652
```

In [121]:

```
### Now for all the models check the mean_absolute_error values:
from sklearn.metrics import mean_absolute_error
print("For GridSearchCV: ",mean_absolute_error(y_test, tree_preds))
print("For LinearRegression: ",mean_absolute_error(y_test, predslr))
print("For Random-Forest: ",mean_absolute_error(y_test, rfrpredictions))
```

```
For GridSearchCV:  161548.44850459037
For LinearRegression:  165031.71973815977
For Random-Forest:  157897.00309501652
```

In [122]:

```
#Save your random-forest model on your system:
import joblib
joblib.dump(gridrfr,"model.pkl")
```

Out[122]:

```
['model.pkl']
```

In [ ]:

```

```