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

import pandas as pd
df= pd.read_csv("Desktop/raw_house_data_raw_house_data.csv")

print(f"df")

df

df.head()
```

```
        MLS  sold_price  zipcode   longitude   latitude  lot_acres
taxes  \
0  21530491   5300000.0    85637 -110.378200  31.356362    2154.00
5272.00
1  21529082   4200000.0    85646 -111.045371  31.594213    1707.00
10422.36
2   3054672   4200000.0    85646 -111.040707  31.594844    1707.00
10482.00
3  21919321   4500000.0    85646 -111.035925  31.645878     636.67
8418.58
4  21306357   3411450.0    85750 -110.813768  32.285162       3.21
15393.00

   year_built  bedrooms  bathrooms  sqrt_ft  garage  \
0        1941        13       10.0  10500.0     0.0
1        1997         2        2.0   7300.0     0.0
2        1997         2        3.0      NaN     NaN
3        1930         7        5.0   9019.0     4.0
4        1995         4        6.0   6396.0     3.0

                                    kitchen_features  fireplaces  \
0           Dishwasher, Freezer, Refrigerator, Oven         6.0
1                       Dishwasher, Garbage Disposal         5.0
2        Dishwasher, Garbage Disposal, Refrigerator         5.0
3  Dishwasher, Double Sink, Pantry: Butler, Refri...         4.0
4  Dishwasher, Garbage Disposal, Refrigerator, Mi...         5.0

                 floor_covering  HOA
0           Mexican Tile, Wood    0
1         Natural Stone, Other    0
2    Natural Stone, Other: Rock  NaN
3  Ceramic Tile, Laminate, Wood  NaN
4             Carpet, Concrete   55
```

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

```
MLS                   0
sold_price            0
zipcode               0
longitude             0
latitude              0
lot_acres            10
taxes                 0
year_built            0
bedrooms              0
bathrooms             6
sqrt_ft              56
garage                7
kitchen_features     33
fireplaces           25
floor_covering        1
HOA                 562
dtype: int64
```

```
print(df.duplicated().sum())
```

```
0
```

```
print(df.isnull().values.any())
```

```
True
```

```
missing_percentage = (df.isnull().sum() / len(df)) * 100
```

```
print(missing_percentage)
```

```
MLS                  0.00
sold_price           0.00
zipcode              0.00
longitude            0.00
latitude             0.00
lot_acres            0.20
taxes                0.00
year_built           0.00
bedrooms             0.00
bathrooms            0.12
sqrt_ft              1.12
garage               0.14
kitchen_features     0.66
fireplaces           0.50
floor_covering       0.02
HOA                 11.24
dtype: float64
```

```
import seaborn as sns
import missingno as msno
import matplotlib.pyplot as plt
```

```
# Heatmap to show missing values
sns.heatmap(df.isnull(), cmap='viridis', cbar=False)
plt.show()

# Bar chart using Missingno
msno.bar(df)
plt.show()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[17], line 1
----> 1 import seaborn as sns
      2 import missingno as msno
      3 import matplotlib.pyplot as plt

ModuleNotFoundError: No module named 'seaborn'
```

```
pip install matplotlib.pyplot
```

```
Note: you may need to restart the kernel to use updated packages.
```

```
ERROR: Could not find a version that satisfies the requirement matplotlib.pyplot (from versions: none)
ERROR: No matching distribution found for matplotlib.pyplot
```

```
zero_counts = (df == 0).sum()
print(zero_counts)
```

```
MLS                   0
sold_price            0
zipcode               0
longitude             0
latitude              0
lot_acres            35
taxes                22
year_built            5
bedrooms              0
bathrooms             0
sqrt_ft               0
garage              184
kitchen_features      0
fireplaces          303
floor_covering        0
HOA                   0
dtype: int64
```

```
# Define a function to detect outliers using IQR
def find_outliers_iqr(data, column):
    Q1 = data[column].quantile(0.25)  # First quartile (25%)
```

```
    Q3 = data[column].quantile(0.75)  # Third quartile (75%)
    IQR = Q3 - Q1  # Interquartile range

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = data[(data[column] < lower_bound) | (data[column] >
upper_bound)]
    return outliers

# Find outliers in a specific column (e.g., 'sold_price')
outliers = find_outliers_iqr(df, 'sold_price')
print(outliers)
```

```
          MLS  sold_price  zipcode   longitude   latitude
lot_acres  \
0    21530491   5300000.0    85637 -110.378200  31.356362    2154.00

1    21529082   4200000.0    85646 -111.045371  31.594213    1707.00

2     3054672   4200000.0    85646 -111.040707  31.594844    1707.00

3    21919321   4500000.0    85646 -111.035925  31.645878     636.67

4    21306357   3411450.0    85750 -110.813768  32.285162       3.21

..        ...         ...      ...         ...        ...        ...

460  21700941   1225000.0    85718 -110.948133  32.344923       1.14

462  21205529   1225000.0    85750 -110.865779  32.325449       0.44

466  21901423   1249000.0    85718 -110.912026  32.290850       2.27

468  21603062   1220000.0    85718 -110.890353  32.291517       0.86

554  21422822   1225000.0    85750 -110.832363  32.281855       0.96


        taxes  year_built  bedrooms  bathrooms  sqrt_ft  garage  \
0     5272.00        1941        13       10.0  10500.0     0.0
1    10422.36        1997         2        2.0   7300.0     0.0
2    10482.00        1997         2        3.0      NaN     NaN
3     8418.58        1930         7        5.0   9019.0     4.0
4    15393.00        1995         4        6.0   6396.0     3.0
..        ...         ...       ...        ...      ...     ...
460  13918.23        2001         4        4.0   4588.0     3.0
462   4016.00        2012         4        5.0   4038.0     3.0
466  10741.76        1994         4        6.0   5462.0     3.0
468  10091.36        2010         3        5.0   3810.0     3.0
554  14936.78        2003         4        5.0   6569.0     3.0
```

```
                                     kitchen_features  fireplaces  \
0              Dishwasher, Freezer, Refrigerator, Oven         6.0
1                        Dishwasher, Garbage Disposal         5.0
2           Dishwasher, Garbage Disposal, Refrigerator         5.0
3    Dishwasher, Double Sink, Pantry: Butler, Refri...         4.0
4    Dishwasher, Garbage Disposal, Refrigerator, Mi...         5.0
..                                                 ...         ...
460  Dishwasher, Garbage Disposal, Refrigerator, Mi...         3.0
462  Dishwasher, Garbage Disposal, Refrigerator, Mi...         2.0
466  Desk, Dishwasher, Double Sink, Garbage Disposa...         2.0
468  Dishwasher, Garbage Disposal, Refrigerator, Mi...         1.0
554  Compactor, Dishwasher, Freezer, Garbage Dispos...         2.0

                           floor_covering  HOA
0                     Mexican Tile, Wood    0
1                   Natural Stone, Other    0
2              Natural Stone, Other: Rock  NaN
3            Ceramic Tile, Laminate, Wood  NaN
4                         Carpet, Concrete   55
..                                     ...  ...
460                 Carpet, Natural Stone  157
462                   Natural Stone, Wood   35
466  Carpet, Wood, Other: Travertine Tile  208
468           Carpet, Natural Stone, Wood  175
554           Carpet, Natural Stone, Wood  110

[395 rows x 16 columns]
```

```python
def find_outliers_iqr(data):
    outlier_sums = {}  # Store sum of outliers for each column

    for column in data.select_dtypes(include=[np.number]):  # Only
numeric columns
        Q1 = data[column].quantile(0.25)  # First quartile
        Q3 = data[column].quantile(0.75)  # Third quartile
        IQR = Q3 - Q1  # Interquartile range

        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Find outliers
        outliers = data[(data[column] < lower_bound) | (data[column] >
upper_bound)]

        # Sum of outliers
        outlier_sums[column] = outliers[column].sum()

    return outlier_sums
```

```
# Get sum of outliers for each column
outlier_sums = find_outliers_iqr(df)

# Print results
print(outlier_sums)
```

```
{'MLS': np.int64(302759631), 'sold_price': np.float64(635496439.0),
'zipcode': np.int64(55668806), 'longitude': np.float64(-
12054.029528000001), 'latitude': np.float64(10912.364324),
'lot_acres': np.float64(18419.039999999997), 'taxes':
np.float64(17555833.86), 'year_built': np.int64(467872), 'bedrooms':
np.int64(1226), 'bathrooms': np.float64(2095.0), 'sqrt_ft':
np.float64(1596961.0), 'garage': np.float64(1240.0), 'fireplaces':
np.float64(89.0)}
```

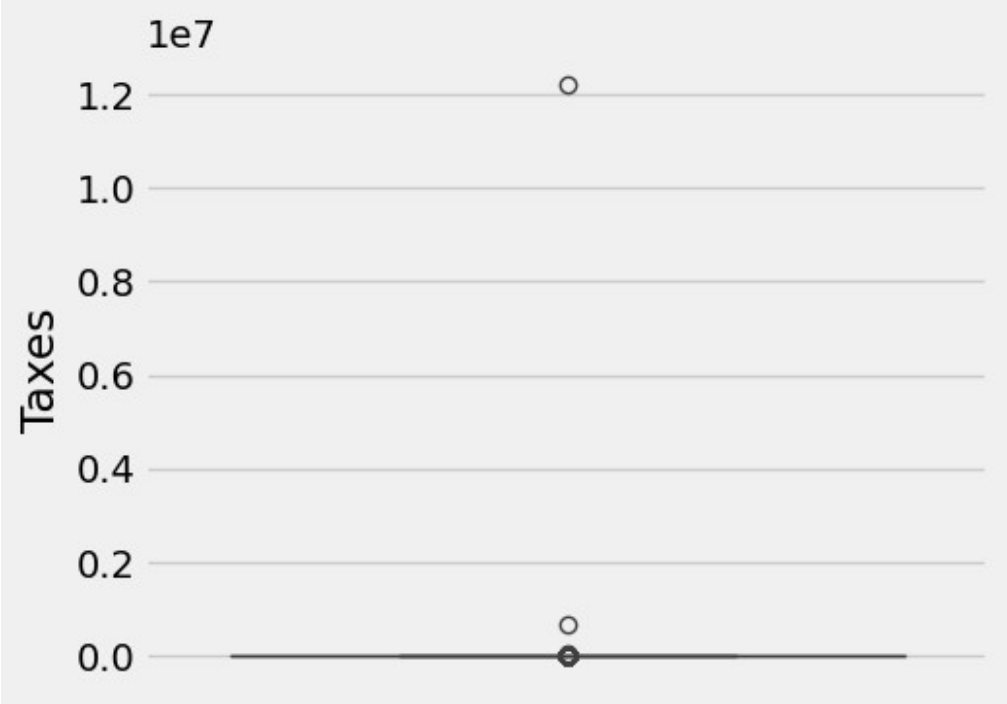
```
def count_outliers_iqr(data):
    outlier_counts = {}  # Dictionary to store counts

    for column in data.select_dtypes(include=[np.number]):  # Only
numeric columns
        Q1 = data[column].quantile(0.25)  # First quartile
        Q3 = data[column].quantile(0.75)  # Third quartile
        IQR = Q3 - Q1  # Interquartile range

        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Find outliers
        outliers = data[(data[column] < lower_bound) | (data[column] >
upper_bound)]

        # Count of outliers
        outlier_counts[column] = outliers.shape[0]

    return outlier_counts

# Get count of outliers for each column
outlier_counts = count_outliers_iqr(df)

# Print results
print(outlier_counts)
```

```
{'MLS': 88, 'sold_price': 395, 'zipcode': 650, 'longitude': 109,
'latitude': 342, 'lot_acres': 578, 'taxes': 277, 'year_built': 246,
'bedrooms': 173, 'bathrooms': 305, 'sqrt_ft': 230, 'garage': 378,
'fireplaces': 12}
```

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

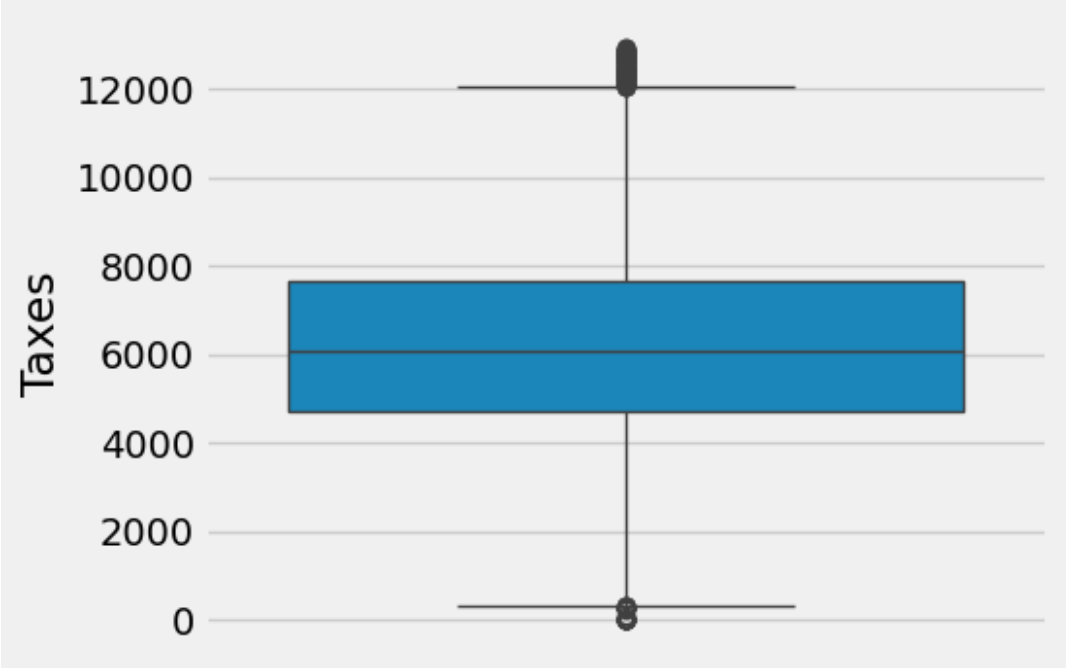
```
plt.figure(figsize=(5,4))
sns.boxplot(y=df['taxes'])

plt.ylabel('Taxes')
```

```
Text(0, 0.5, 'Taxes')
```



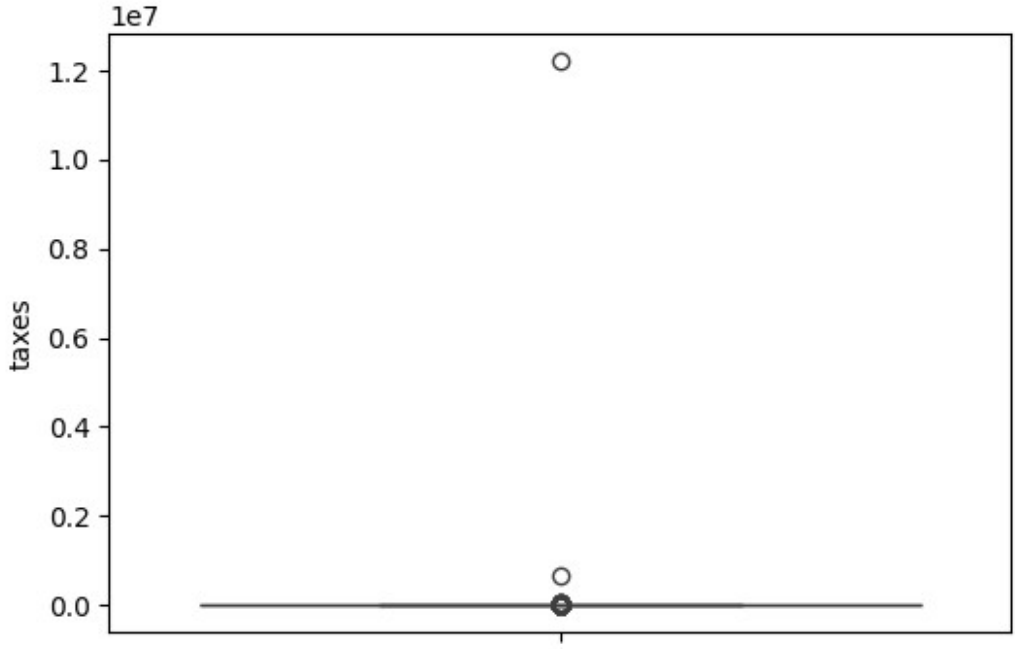

```
Q1 = df['taxes'].quantile(0.25)
Q3 = df['taxes'].quantile(0.75)
IQR = Q3 - Q1  # Interquartile Range

# Define lower and upper bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Remove outliers
df_filtered = df[(df['taxes'] >= lower_bound) & (df['taxes'] <=
upper_bound)]

plt.figure(figsize=(5, 4))
sns.boxplot(y=df_filtered['taxes'])


plt.ylabel('Taxes')
```

```
Text(0, 0.5, 'Taxes')
```




```
plt.figure(figsize=(6, 4))
sns.boxplot(y=df['taxes'])
```

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



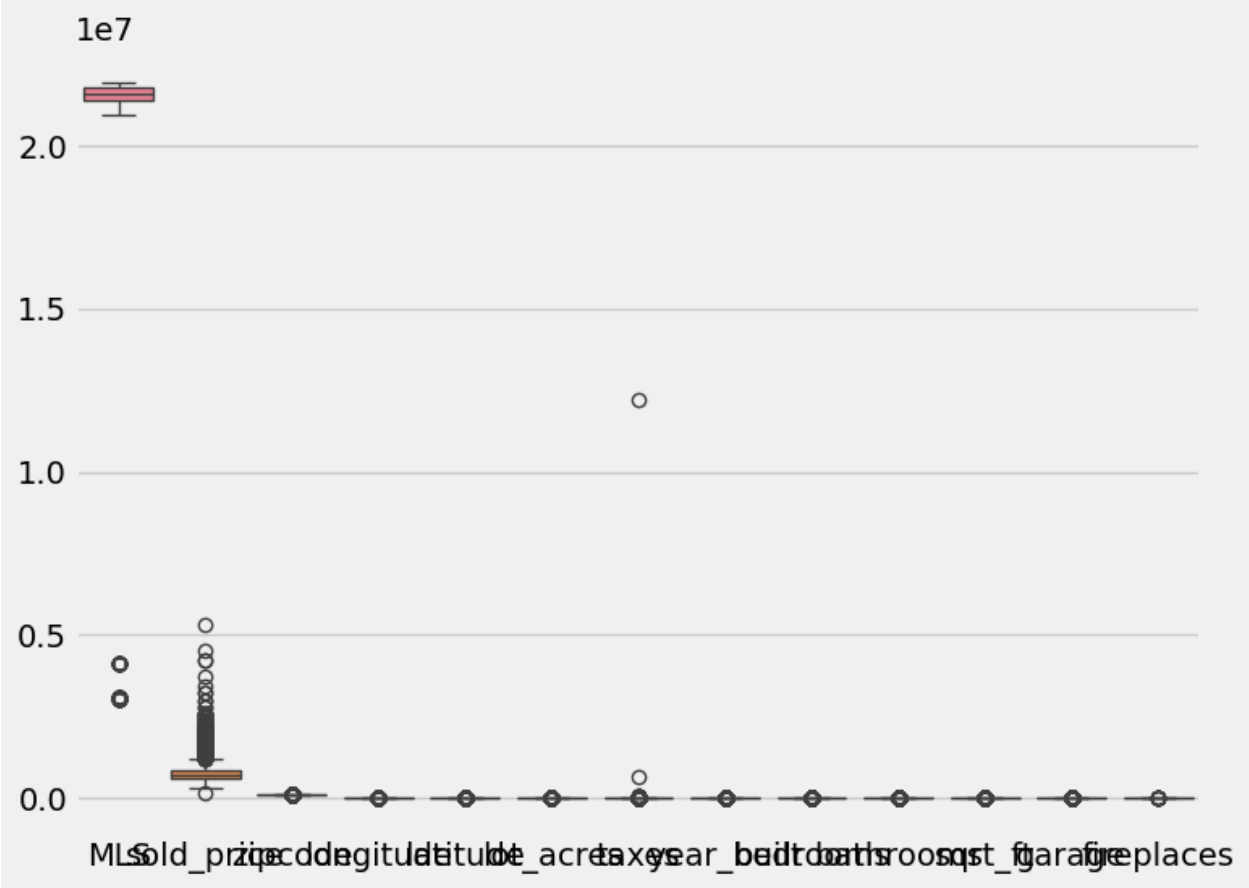

```
import pandas as pd

# Sample DataFrame
```

```
data = df  # Example with an outlier (400)
df = pd.DataFrame(data)

# Calculate Q1 (25th percentile) and Q3 (75th percentile)
Q1 = df['ColumnName'].quantile(0.25)
Q3 = df['ColumnName'].quantile(0.75)
IQR = Q3 - Q1  # Interquartile Range

# Define lower and upper bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Remove outliers
df_filtered = df[(df['ColumnName'] >= lower_bound) & (df['ColumnName']
<= upper_bound)]

print("Original DataFrame:")
print(df)
print("\nDataFrame after removing outliers:")
print(df_filtered)
```

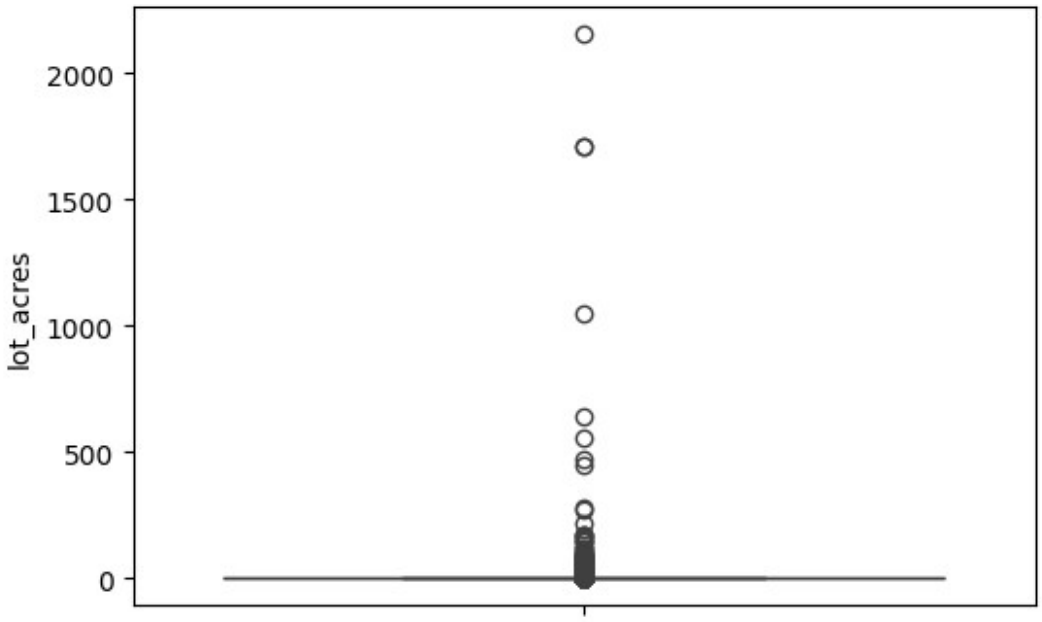
```
Original DataFrame:
   ColumnName
0          10
1          20
2          30
3          25
4          15
5         400
6          35
7          50
8          45
9          30

DataFrame after removing outliers:
   ColumnName
0          10
1          20
2          30
3          25
4          15
6          35
7          50
8          45
9          30
```

```
data = pd.DataFrame(df)

plt.figure(figsize=(8, 6))
sns.boxplot(data=data)
```

```
<Axes: >
```

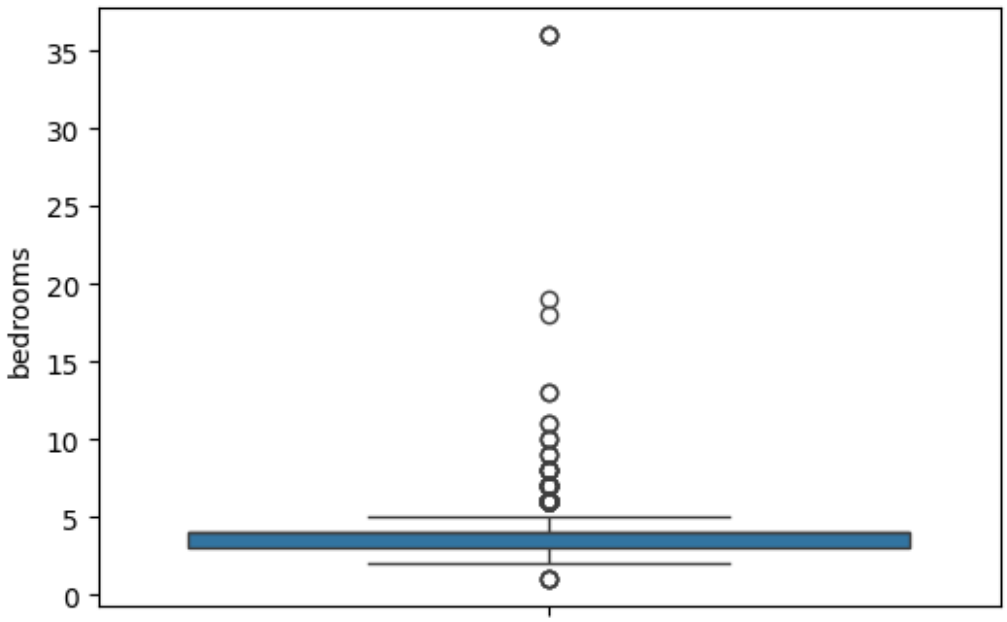



```
lot acres , bedrooms , bathrooms taxes ,hoa

plt.figure(figsize=(6, 4))
sns.boxplot(y=df['lot_acres'])

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




```
plt.figure(figsize=(6, 4))
sns.boxplot(y=df['bedrooms'])
```

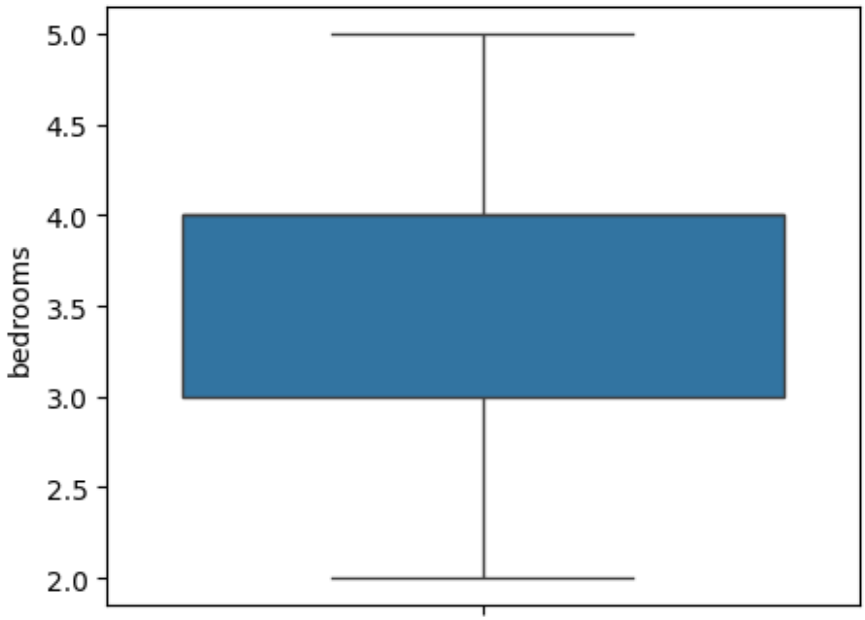
```
<Axes: ylabel='bedrooms'>
```




```
Q1 = df['bedrooms'].quantile(0.25)
Q3 = df['bedrooms'].quantile(0.75)
IQR = Q3 - Q1  # Interquartile Range

# Define lower and upper bounds
```

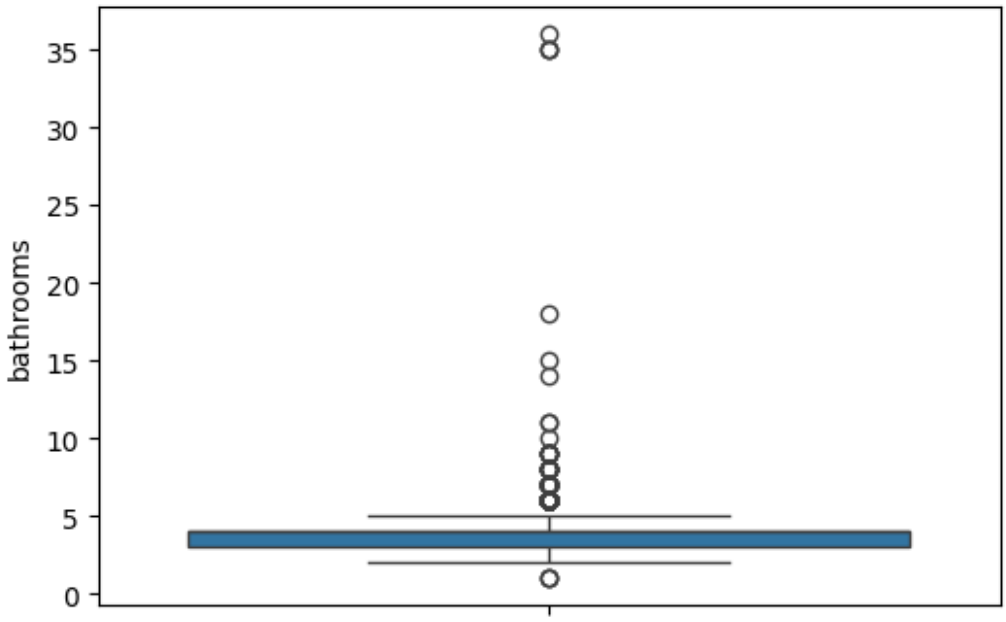
```
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Remove outliers
df_filtered = df[(df['bedrooms'] >= lower_bound) & (df['bedrooms'] <=
upper_bound)]

plt.figure(figsize=(5, 4))
sns.boxplot(y=df_filtered['bedrooms'])


plt.ylabel('bedrooms')
```

```
Text(0, 0.5, 'bedrooms')
```



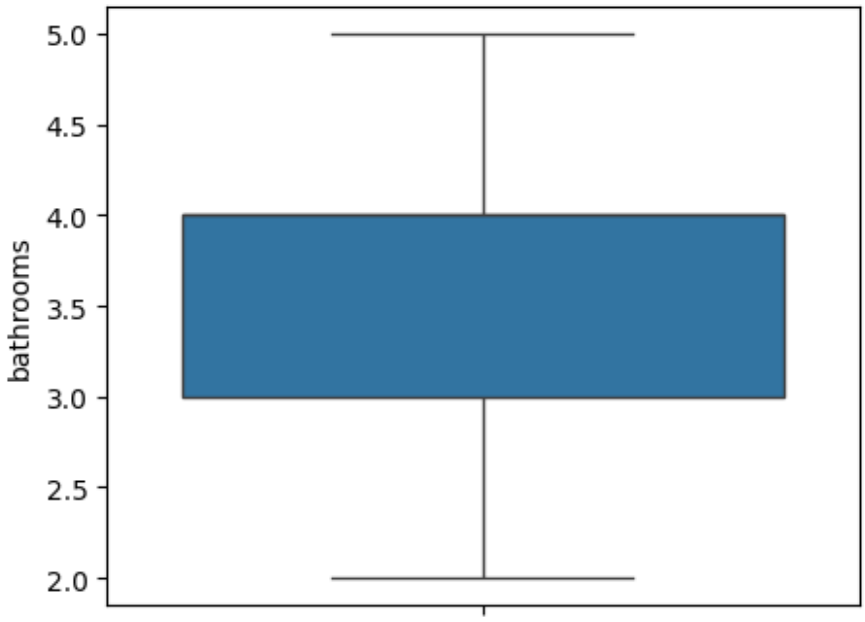

```
plt.figure(figsize=(6, 4))
sns.boxplot(y=df['bathrooms'])
```

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



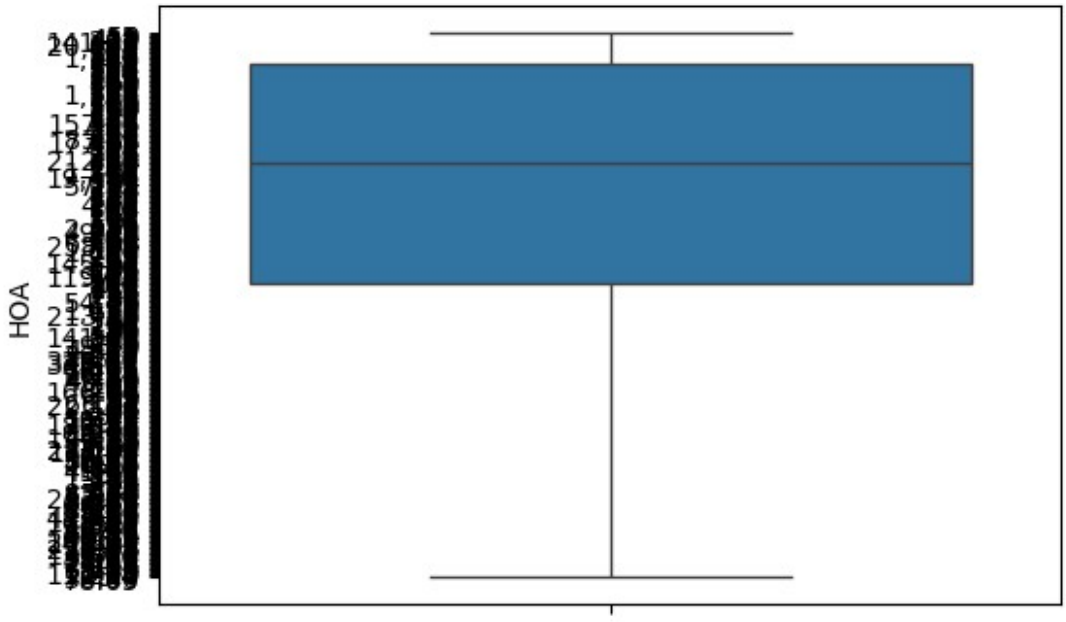

```
name  = 'bathrooms'
Q1 = df[name].quantile(0.25)
Q3 = df[name].quantile(0.75)
IQR = Q3 - Q1  # Interquartile Range

# Define lower and upper bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Remove outliers
df_filtered = df[(df[name] >= lower_bound) & (df[name] <=
upper_bound)]

plt.figure(figsize=(5, 4))
sns.boxplot(y=df_filtered[name])


plt.ylabel(name)
```

```
Text(0, 0.5, 'bathrooms')
```



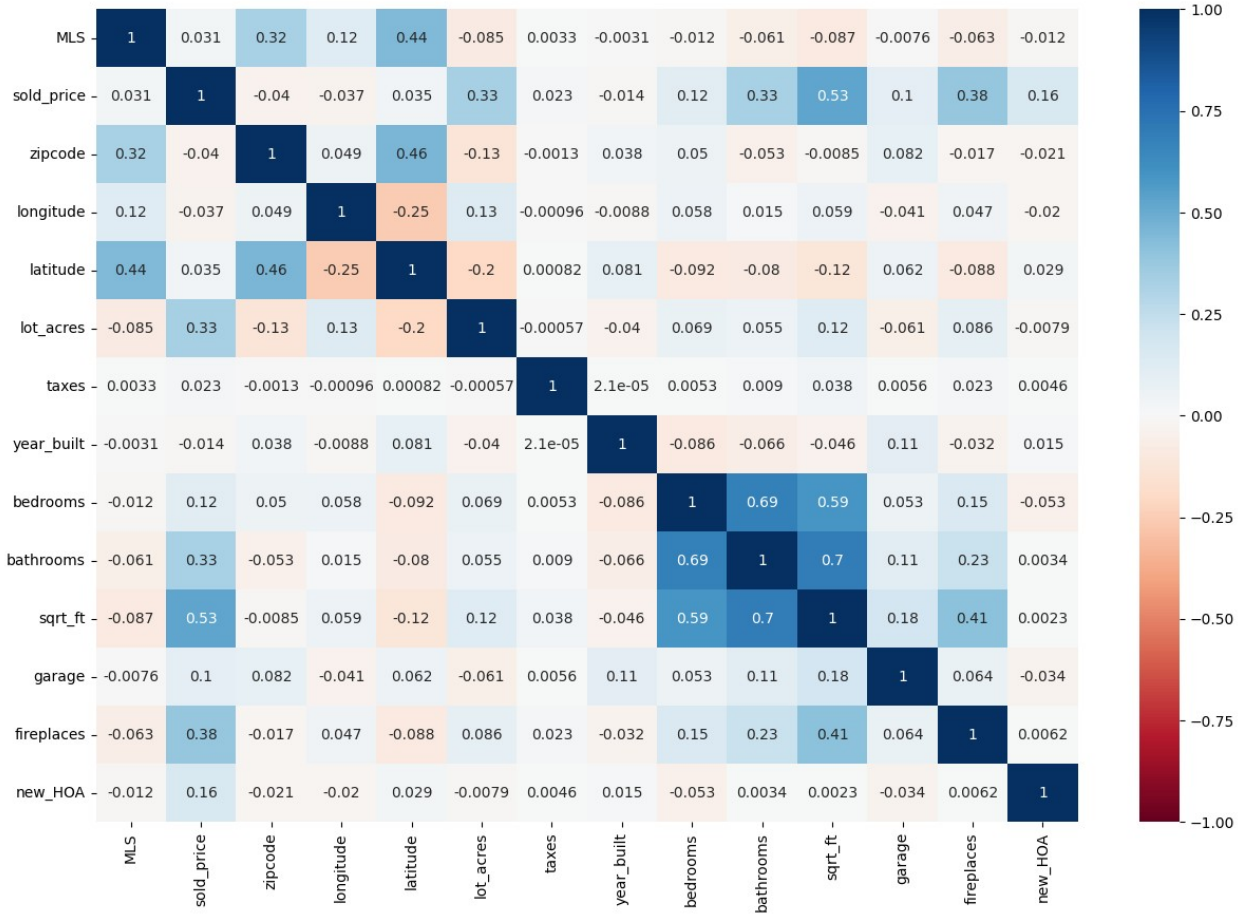

```
name = 'HOA'
plt.figure(figsize=(6, 4))
sns.boxplot(y=df[name])
```

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



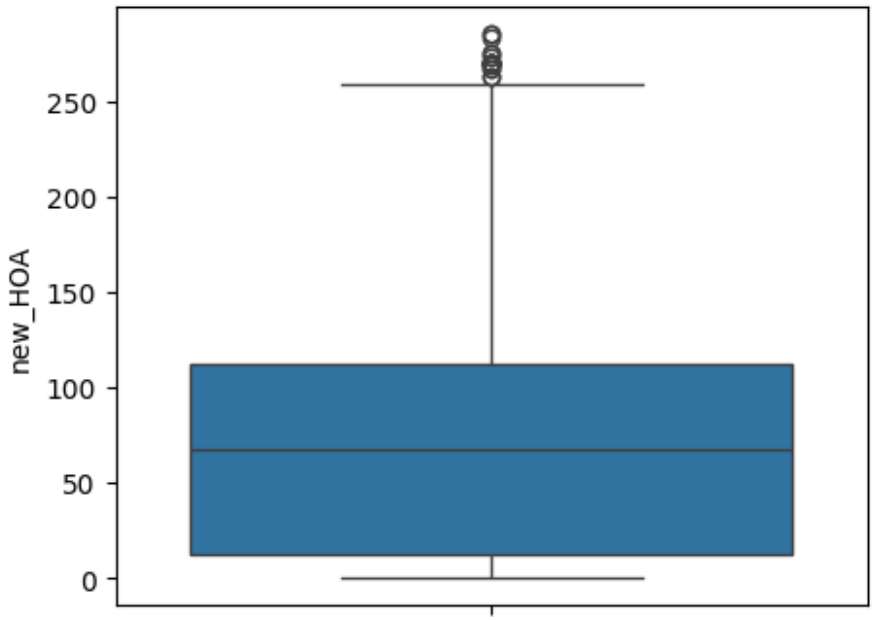

```
def outliremove(name: str):
    Q1 = df[name].quantile(0.25)
    Q3 = df[name].quantile(0.75)
    IQR = Q3 - Q1  # Interquartile Range
```

```
    # Define lower and upper bounds
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Remove outliers
    df_filtered = df[(df[name] >= lower_bound) & (df[name] <=
upper_bound)]

    plt.figure(figsize=(5, 4))
    sns.boxplot(y=df_filtered[name])
    plt.ylabel(name)

Total_Cols = df.select_dtypes(include=['number']).columns

plt.figure(figsize=(15,10))
sns.heatmap(df[Total_Cols].corr(),annot=True,vmin=-1,cmap='RdBu');
```

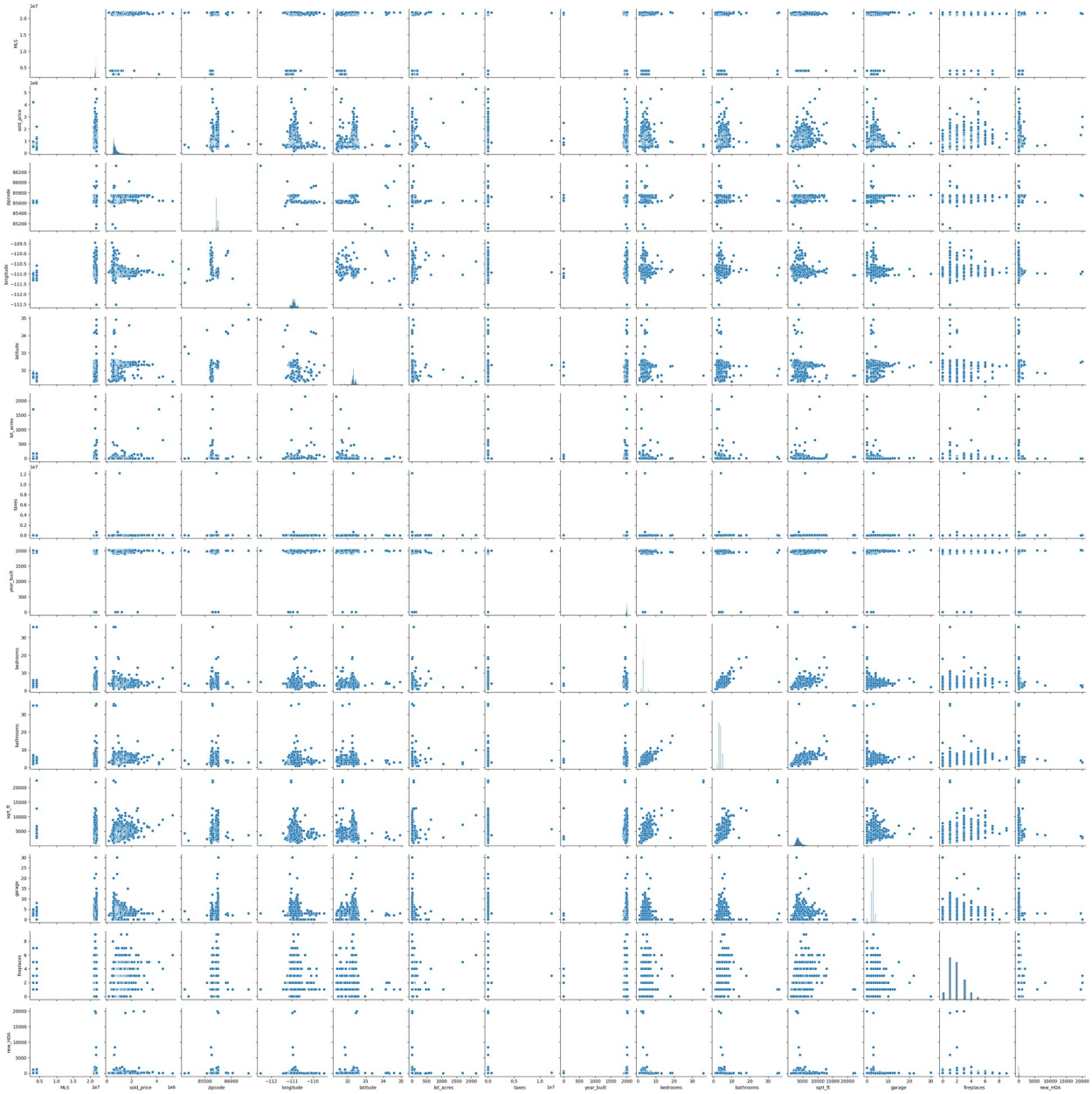



```
import re
df['new_HOA'] = df['HOA'].apply(lambda x: re.sub(r'[^0-9.]', '',
str(x)) if pd.notna(x) else x)
```

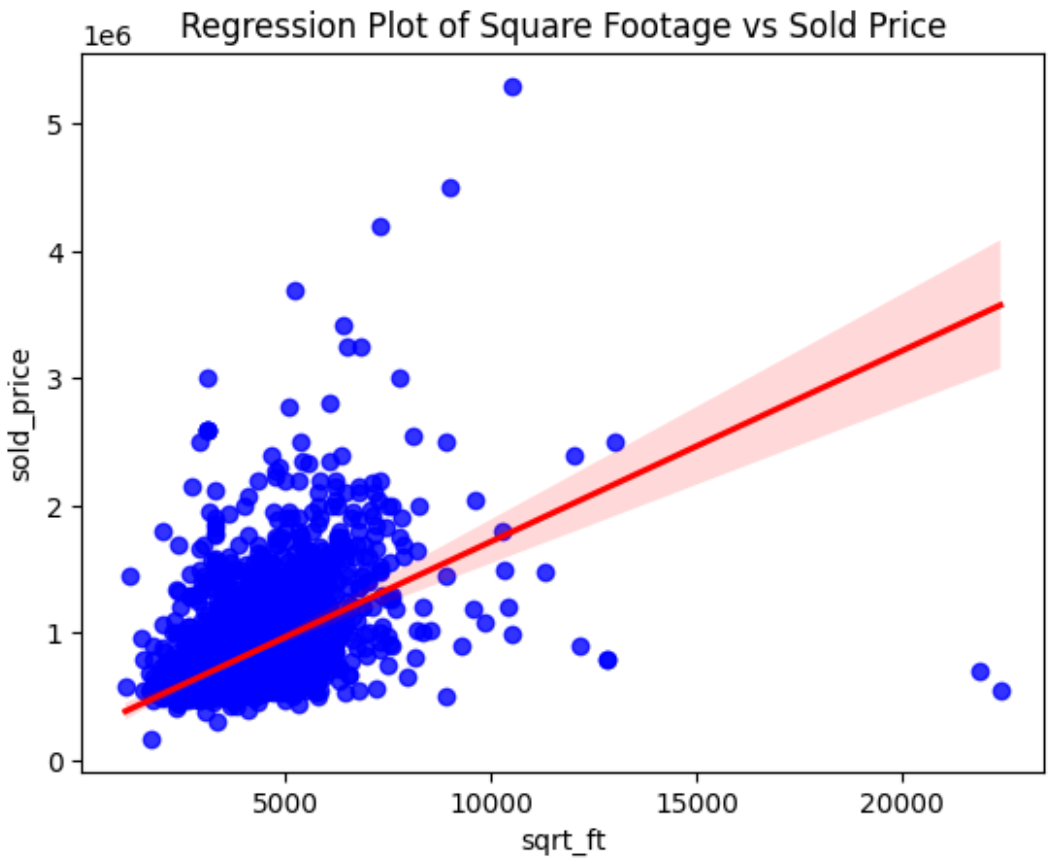
```
df.new_HOA = pd.to_numeric(df.new_HOA, errors='coerce')

quantile_99 = df['new_HOA'].quantile(0.99)
data_sorted = df[df['new_HOA'] < quantile_99]
df.new_HOA = df.new_HOA.fillna(data_sorted.new_HOA.mean())

outliremove('new_HOA')
```



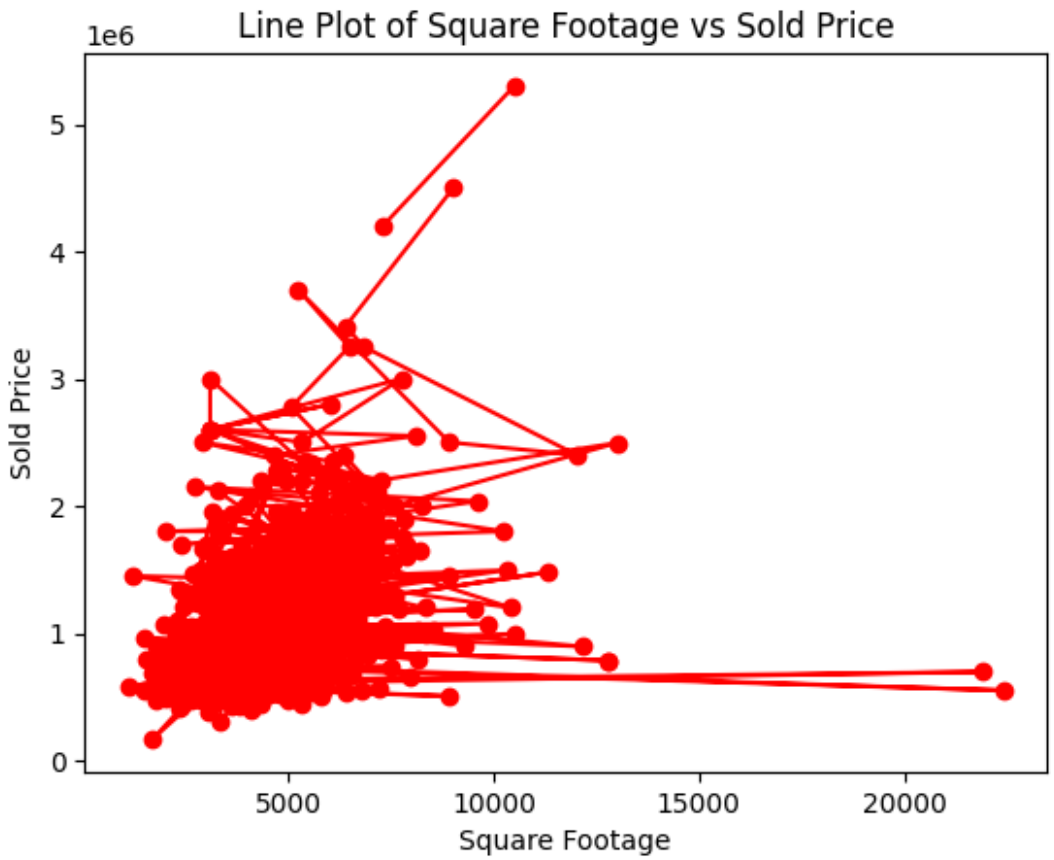

```
sns.histplot(df.sold_price, df.sqrt_ft,kde=True, bins=20,
color='blue')
plt.tight_layout()
plt.show()

---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[126], line 1
----> 1 sns.histplot(df.sold_price, df.sqrt_ft,kde=True, bins=20,
color='blue')
      2 plt.tight_layout()
      3 plt.show()

TypeError: histplot() takes from 0 to 1 positional arguments but 2
positional arguments (and 3 keyword-only arguments) were given

sns.pairplot(df)

<seaborn.axisgrid.PairGrid at 0x11f0fdc1bb0>
```




```
sns.regplot(x=df['sqrt_ft'], y=df['sold_price'], scatter_kws={"color": "blue"}, line_kws={"color": "red"})
plt.title("Regression Plot of Square Footage vs Sold Price")
plt.show()
```



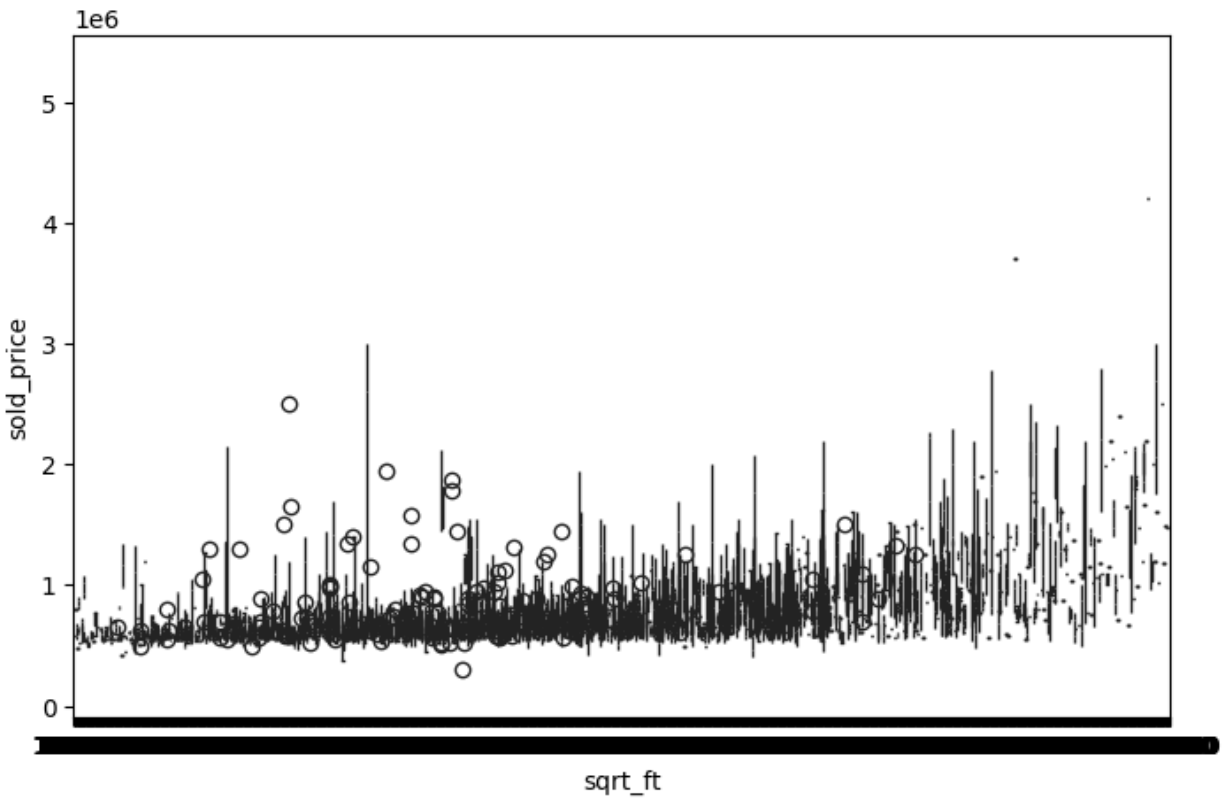

```
plt.plot(df['sqrt_ft'], df['sold_price'], marker='o', linestyle='-',
color='red')
plt.xlabel("Square Footage")
plt.ylabel("Sold Price")
plt.title("Line Plot of Square Footage vs Sold Price")
plt.show()
```




```
plt.figure(figsize=(8,5))
sns.boxplot(x=df['sqrt_ft'], y=df['sold_price'],data =
df,palette='Blues')
```

```
C:\Users\Shravan Gumudavelli\AppData\Local\Temp\
ipykernel_16236\2066491374.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x=df['sqrt_ft'], y=df['sold_price'],palette='Blues')
```

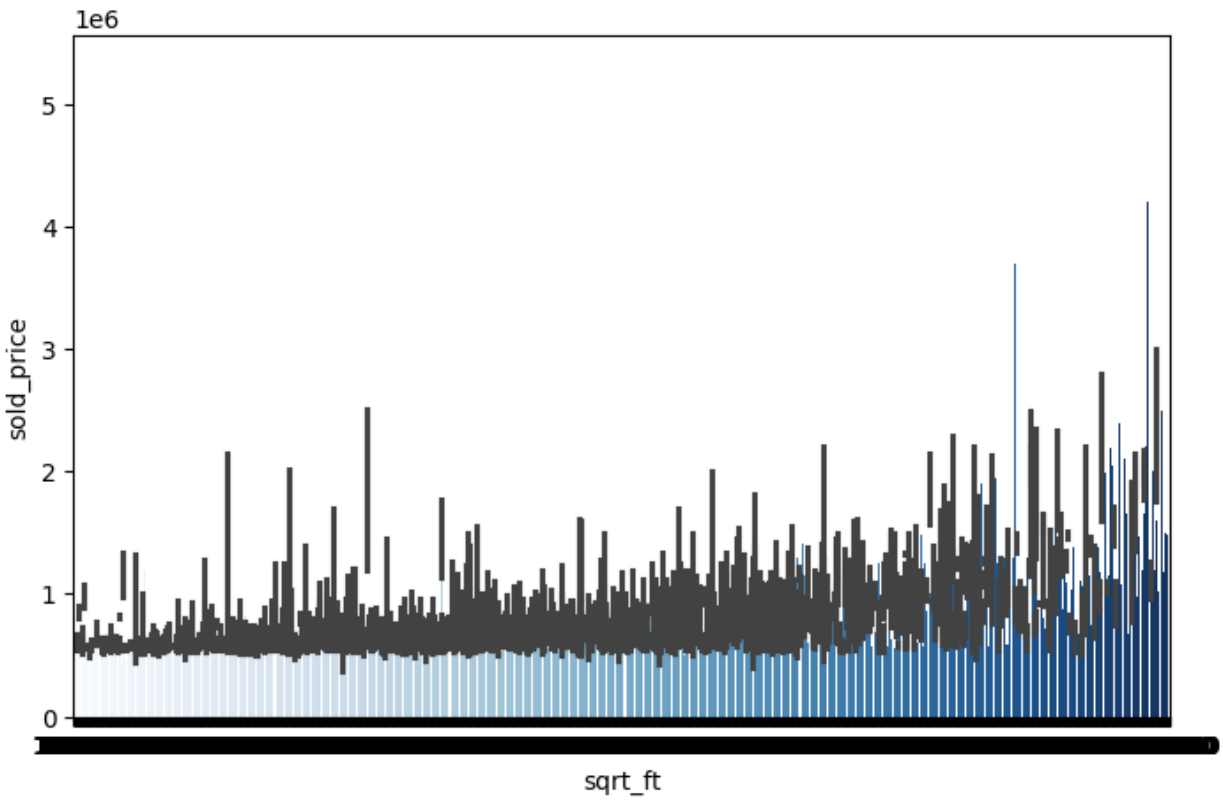
```
<Axes: xlabel='sqrt_ft', ylabel='sold_price'>
```



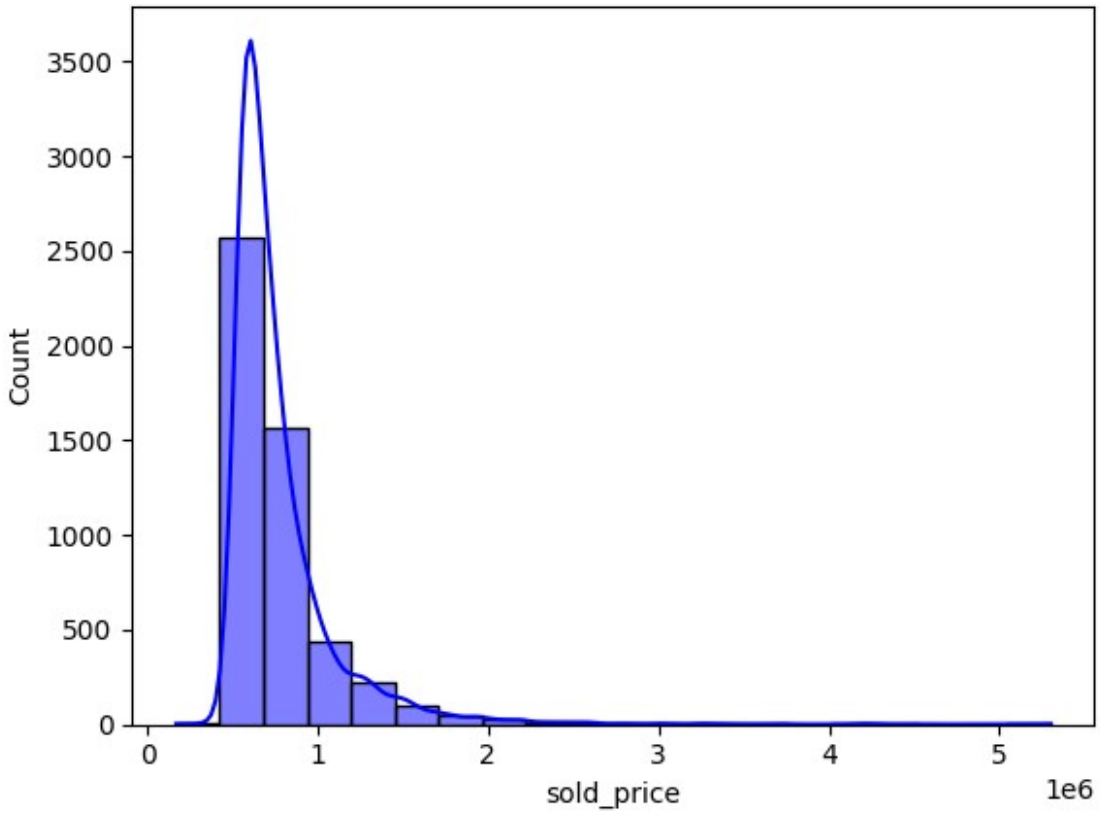

```
plt.figure(figsize=(8,5))
sns.barplot(x='sqrt_ft', y='sold_price', data=df, palette='Blues')
```

```
C:\Users\Shravan Gumudavelli\AppData\Local\Temp\
ipykernel_16236\3275738729.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='sqrt_ft', y='sold_price', data=df, palette='Blues')
```

```
<Axes: xlabel='sqrt_ft', ylabel='sold_price'>
```




```
sns.histplot(df['sold_price'], kde=True, bins=20, color='blue')
plt.show()
```


3500
3000
2500
2000
1500
1000
500
0
Count
0
1
2
3
4
5
sold_price
1e6