Group 8 DSF PT6-Phase 2 Project

`Members`

- *steve.abonyo@student.moringaschool.com*
- *brian.kariithi@student.moringaschool.com*
- *rosaline.mungai@student.moringaschool.com*
- *ruth.nyakio@student.moringaschool.com*
- *lynette.mwiti@student.moringaschool.com*
- *ivan.wawire@student.moringaschool.com*

1. Business Understanding

Overview :
Housing sales in a northwestern county between the year `2014` and `2015`

Problem Statement :
Finsco Limited, a real estate group investing in USA real estate has opened a consultancy arm. for their first project, they would like to understand how home renovations might increase the estimated value of homes, and by what amount.

The goal is to get insights to provide advice to homeowners, real estate investors and clients who do house-flipping

They have tasked the Hepta Group to conduct multiple linear regression modelling to analyze house sales in a Northwestern County they have been provided with.

Stakeholders :
-Homeowners : *These are the people who want to increase the value of their homes and want to know the kind of renovations to do.*
-Real estate agency : *The company that is conducting this model to help homeowners know what renovations to do to increase the value of their homes.*

Understanding :
We have several parameters/variables in our data that when adjusted/improved, may positively affect the value of homes in this county. `We need to find and observe which parameter greatly influences the value of the homes and perhaps, what parameters might give the best value of the homes.`
Our dependent variable is the value of homes; what we are trying to predict.
Our independent variables are the home renovations, and there are several, we will use this to help us find the best possible values of the homes.

The above two concepts (`dependent` and `independent`) lead us to the concept we are going to use which is the multiple linear regression. `Multiple linear` `regression` is used when we want to predict a dependent variable (value of homes) using two or more independent variables (the several parameters present in our data).

Key items to check before we build the model;

- Have a basic understanding of the data
- Which parameters greatly/least influence the value of homes for the homeowners?
- Which parameters are irrelevant to our model (*through observation after understanding our problem*)

Implications

- With the insights provided, Finsco Limited and its clients can strategically invest in real estate by choosing renovations that significantly increase home values.
- Homeowners working with Finsco's consultancy learn which renovations offer the best returns, helping them wisely enhance their properties.
- House flippers working with FINSCO consultancy can tailor their strategies to target high-ROI renovations.
- The consultancy's success not only benefits Finsco and its clients but also stimulates economic growth through increased renovation and property transaction activities.

2. Data understanding

Data Source & Size
The data we are going to use is called `kc_house_data.csv` from the `King County House Sales` and `it has 21597 records`.

Variables description
Below is the description of our variables;

- `id` - Unique identifier for a house
- `date` - Date house was sold
- `price` - Sale price (prediction target)
- `bedrooms` - Number of bedrooms
- `bathrooms` - Number of bathrooms
- `sqft_living` - Square footage of living space in the home
- `sqft_lot` - Square footage of the lot
- `floors` - Number of floors (levels) in house
- `waterfront` - Whether the house is on a waterfront
  - Includes Duwamish, Elliott Bay, Puget Sound, Lake Union, Ship Canal, Lake Washington, Lake Sammamish, other lake, and river/slough waterfronts
- `view` - Quality of view from house
  - Includes views of Mt. Rainier, Olympics, Cascades, Territorial, Seattle Skyline, Puget Sound, Lake Washington, Lake Sammamish, small lake / river / creek, and other
- `condition` - How good the overall condition of the house is. Related to maintenance of house.

    - See the King County Assessor Website for further explanation of each condition code
- `grade` - Overall grade of the house. Related to the construction and design of the house.
    - See the King County Assessor Website for further explanation of each building grade code
- `sqft_above` - Square footage of house apart from basement
- `sqft_basement` - Square footage of the basement
- `yr_built` - Year when house was built
- `yr_renovated` - Year when house was renovated
- `zipcode` - ZIP Code used by the United States Postal Service
- `lat` - Latitude coordinate
- `long` - Longitude coordinate
- `sqft_living15` - The square footage of interior housing living space for the nearest 15 neighbors
- `sqft_lot15` - The square footage of the land lots of .the nearest 15 neighbors

The `target variable from the above dataset is the price`, where as the `others form the predictor/independent variables`. From these datasets we can already start seeing some predictor variables that may have an impact on the target variable (*ofcourse this is by observation*), for example how will changing the number of bedrooms affect the house prices for the home owners.

Limitations :
From a quick observation of the data, we have noticed the `presence of missing values` in some of the predictor variables like waterfront and view. We will first have to check if the variables having missing values significantly affect the value of homes or not. Also we have noticed that `there are non-numerical variables`. Linear regression only uses numerical columns. We will have to adjust this columns to numbers if at all we are going to use them in building our model (NB: it is a requirement that we use atleast one non-numeric column)

In [1]:
```
# Importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv('kc_house_data.csv')

# Displaying the first few rows of the data frame
data.head()
```

Out[1]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floor:
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.(
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.(
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.(
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.(
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.(

5 rows × 21 columns

In [2]:

```
# Getting basic information about the datatype
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21597 non-null  int64  
 1   date           21597 non-null  object 
 2   price          21597 non-null  float64
 3   bedrooms       21597 non-null  int64  
 4   bathrooms      21597 non-null  float64
 5   sqft_living    21597 non-null  int64  
 6   sqft_lot       21597 non-null  int64  
 7   floors         21597 non-null  float64
 8   waterfront     19221 non-null  object 
 9   view           21534 non-null  object 
 10  condition      21597 non-null  object 
 11  grade          21597 non-null  object 
 12  sqft_above     21597 non-null  int64  
 13  sqft_basement  21597 non-null  object 
 14  yr_built       21597 non-null  int64  
 15  yr_renovated   17755 non-null  float64
 16  zipcode        21597 non-null  int64  
 17  lat            21597 non-null  float64
 18  long           21597 non-null  float64
 19  sqft_living15  21597 non-null  int64  
 20  sqft_lot15     21597 non-null  int64  
dtypes: float64(6), int64(9), object(6)
memory usage: 3.5+ MB
None
```

In [3]:

```
# Descriptive statistics of our dataset
data.describe()
```

Out[3]:

	id	price	bedrooms	bathrooms	sqft_living	sqf
count	2.159700e+04	2.159700e+04	21597.000000	21597.000000	21597.000000	2.159700
mean	4.580474e+09	5.402966e+05	3.373200	2.115826	2080.321850	1.509941
std	2.876736e+09	3.673681e+05	0.926299	0.768984	918.106125	4.141264
min	1.000102e+06	7.800000e+04	1.000000	0.500000	370.000000	5.200000
25%	2.123049e+09	3.220000e+05	3.000000	1.750000	1430.000000	5.040000
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068500
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359

In [4]:
```
# Check for missing values
print(data.isnull().sum())
```

```
id                  0
date                0
price               0
bedrooms            0
bathrooms           0
sqft_living         0
sqft_lot            0
floors              0
waterfront       2376
view               63
condition           0
grade               0
sqft_above          0
sqft_basement       0
yr_built            0
yr_renovated     3842
zipcode             0
lat                 0
long                0
sqft_living15       0
sqft_lot15          0
dtype: int64
```

In [5]:
```
# Sampling 5 random rows of our dataset
data.sample(5)
```

Out[5]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	fl
21437	2254502071	5/23/2014	375000.0	2	2.50	750	1430	
8905	822059038	7/31/2014	290000.0	6	4.50	2810	11214	
10235	3876200060	5/2/2014	382500.0	4	1.75	1560	8700	
13619	8078390150	6/26/2014	675750.0	4	2.50	2770	10274	
955	510002065	3/23/2015	700000.0	4	1.00	1980	4560	

5 rows × 21 columns

In [6]:

```
# Getting data types of our dataset
data.dtypes
```

Out[6]:

```
id                int64
date             object
price           float64
bedrooms          int64
bathrooms       float64
sqft_living       int64
sqft_lot          int64
floors          float64
waterfront       object
view             object
condition        object
grade            object
sqft_above        int64
sqft_basement    object
yr_built          int64
yr_renovated    float64
zipcode           int64
lat             float64
long            float64
sqft_living15     int64
sqft_lot15        int64
dtype: object
```

In [7]:

```
# Explore unique values and frequency counts for categorical variables
categorical_cols = ['waterfront', 'view', 'condition', 'grade', 'zipcode']
for col in categorical_cols:
    print(data[col].value_counts())
```

```
waterfront
NO     19075
YES      146
Name: count, dtype: int64
view
NONE         19422
AVERAGE        957
GOOD           508
FAIR           330
EXCELLENT      317
Name: count, dtype: int64
condition
Average      14020
Good          5677
Very Good     1701
Fair           170
Poor            29
Name: count, dtype: int64
grade
7 Average        8974
8 Good           6065
9 Better         2615
6 Low Average    2038
10 Very Good     1134
11 Excellent      399
5 Fair            242
12 Luxury          89
4 Low              27
13 Mansion         13
3 Poor              1
Name: count, dtype: int64
zipcode
98103    602
98038    589
98115    583
98052    574
98117    553
        ...
98102    104
98010    100
98024     80
98148     57
98039     50
Name: count, Length: 70, dtype: int64
```

In [8]:
```
# Visualising how data using histogram
# Histograms for only numerical variables

data.hist(bins=20, figsize=(15,10))
plt.show()
```

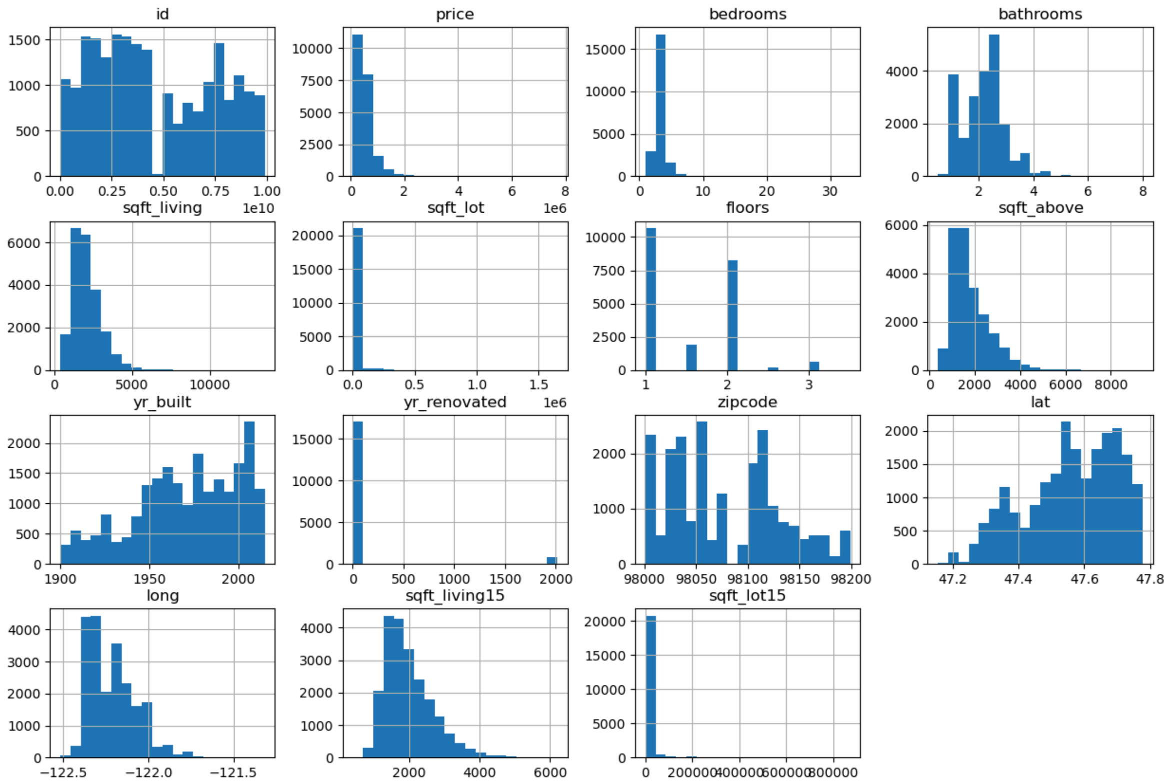


Tableau dashboard

In order to have better understanding of the data, a tableaue dashboard was produced. Below is the link to the dashboard

https://public.tableau.com/views/KingCountySalesDashboard-Group8/KingCountySalesdashboard?:language=en-US&publish=yes&:sid=&:display_count=n&:origin=viz_share_link

A. Data Cleaning

In [9]:
```
data['sqft_basement'].value_counts()
## 1st we can see there are a lot of zeros
## the missing values are 454(indicated as ?)
```

Out[9]:
```
sqft_basement
0.0       12826
?           454
600.0       217
500.0       209
700.0       208
          ...
1920.0        1
3480.0        1
2730.0        1
2720.0        1
248.0         1
Name: count, Length: 304, dtype: int64
```

In [10]:
```
## lets strip the ? to be an empty space then we impute the blanks
data['sqft_basement'] = data['sqft_basement'].replace('?',None).astype("float")
```

```
data['sqft_basement'].isna().sum()
```

Out[10]: 454

In [11]:
```
print("Mean:",data['sqft_basement'].mean())
print("Median:",data['sqft_basement'].median())

## let's check the skewness of this variable so that we know how we will impute
print("Skewness:",data['sqft_basement'].skew())
## this data is positively skewed/highly skewed/right skewed, the tail is on the
```

```
Mean: 291.851723974838
Median: 0.0
Skewness: 1.574329769495408
```

In [12]:
```
## imputing the missing value using the median
data['sqft_basement'].fillna(data['sqft_basement'].median(), inplace=True)
data['sqft_basement'].isna().sum() #now there are no missing values
```

Out[12]: 0

Dropping Columns

In [13]:
```
## We have decided to drop these columns

dropped_columns = ['date', 'view', 'sqft_above', 'sqft_basement', 'yr_renovated'
data1 = data.drop(columns = dropped_columns)
data1.head(2)
```

Out[13]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	condition
0	221900.0	3	1.00	1180	5650	1.0	NaN	Average
1	538000.0	3	2.25	2570	7242	2.0	NO	Average

Missing Data

Column `waterfront` has missing values, we will fill the missing values with `NA` as they are empty spaces. We don't want to manipulate the analysis by increasing the number of `YES's/ NO's`

In [14]:
```
print("Missing values:", data1['waterfront'].isna().sum())
print("Table BEFORE dealing with missing values")
data1['waterfront'].value_counts()
```

```
Missing values: 2376
Table BEFORE dealing with missing values
```

Out[14]:
```
waterfront
NO     19075
YES      146
Name: count, dtype: int64
```

In [15]:
```
data1['waterfront'].fillna("NA",inplace=True)
print("Missing values:", data1['waterfront'].isna().sum())
```

```
print("Table AFTER dealing with missing values")
data1['waterfront'].value_counts()
```

```
Missing values: 0
Table AFTER dealing with missing values
```

Out[15]:
```
waterfront
NO     19075
NA      2376
YES      146
Name: count, dtype: int64
```

Encoding Data

In [16]:
```
# Checking the data types of our columns
data1.dtypes
```

Out[16]:
```
price          float64
bedrooms         int64
bathrooms      float64
sqft_living      int64
sqft_lot         int64
floors         float64
waterfront      object
condition       object
grade           object
yr_built         int64
dtype: object
```

They are 3 i.e., `waterfront` , `condition` and `grade`
Based on our business problem which is ***Advice to homeowners by a real estate agency on how `home renovations` might increase the estimated `values of their homes` and by what amount using multiple linear regression***, we are going to choose `condition` variable, encode it and use it in our model.
The reason is because if we improve the condition of the house e.g from average to good, then the value of the home might increase.
Following the above decision, we will drop the other two categorical variables i.e., waterfront and grade

We are going to use ordinal encoding to transform our selected categorical variable to numeric variable. Why we have selected ordinal encoding is because the choices observe some sequence/hierachy

In [17]:
```
# Summary and count of the condition column values
data1['condition'].value_counts()
```

Out[17]:
```
condition
Average      14020
Good          5677
Very Good     1701
Fair           170
Poor            29
Name: count, dtype: int64
```

In [18]:
```
## Create the codes
condition_codes = {'Poor':1, 'Fair':2, 'Average':3, 'Good':4, 'Very Good':5}
```

```
## Inputting our codes back into our data frame
data1['condition_coded'] = data1['condition'].replace(condition_codes)
data1.head(2)
```

Out[18]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	condition
0	221900.0	3	1.00	1180	5650	1.0	NA	Average
1	538000.0	3	2.25	2570	7242	2.0	NO	Average

In [19]:

```
# Dropping more columns that contain objects

drop_columns = ['waterfront', 'condition', 'grade']
data2 = data1.drop(columns = drop_columns)
data2.head(2)
```

Out[19]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	yr_built	condition_cod
0	221900.0	3	1.00	1180	5650	1.0	1955	
1	538000.0	3	2.25	2570	7242	2.0	1951	

Correlation

In [20]:

```
## Correlation of our selected variables to the price
data2.corr()['price'].sort_values(ascending = False)
```

Out[20]:

```
price             1.000000
sqft_living       0.701917
bathrooms         0.525906
bedrooms          0.308787
floors            0.256804
sqft_lot          0.089876
yr_built          0.053953
condition_coded   0.036056
Name: price, dtype: float64
```

From the correlation results above we can already start to see that variables like sqft_living, bathrooms, bedrooms and floors will produce a better model as compared to their other counterparts.
Our newly coded condition variable is the least on the correlation table, implying it will not be very significant to our model when compared to the rest.

In [21]:

```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   price           21597 non-null  float64
 1   bedrooms        21597 non-null  int64  
 2   bathrooms       21597 non-null  float64
 3   sqft_living     21597 non-null  int64  
 4   sqft_lot        21597 non-null  int64  
 5   floors          21597 non-null  float64
 6   yr_built        21597 non-null  int64  
 7   condition_coded 21597 non-null  int64  
dtypes: float64(3), int64(5)
memory usage: 1.3 MB
```

In [22]:
```
data2.isna().sum()
##by now we dont have any missing value
```

Out[22]:
```
price              0
bedrooms           0
bathrooms          0
sqft_living        0
sqft_lot           0
floors             0
yr_built           0
condition_coded    0
dtype: int64
```

In [23]:
```
## The final data to proceed to the modelling step
df = data2.copy(deep=True)
df.head()
```

Out[23]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	yr_built	condition_cod
0	221900.0	3	1.00	1180	5650	1.0	1955	
1	538000.0	3	2.25	2570	7242	2.0	1951	
2	180000.0	2	1.00	770	10000	1.0	1933	
3	604000.0	4	3.00	1960	5000	1.0	1965	
4	510000.0	3	2.00	1680	8080	1.0	1987	

3. Modelling

A. Data Splitting

- Variable X contains all the independent features except the 'price' column.
- y contains the 'price' column, the target variable.
- We will use 20% of the data for testing and 80% for training.

In [24]:
```
from sklearn.model_selection import train_test_split
```

```
# Getting out independent features. We will exclude the price
X = df.drop('price', axis=1)

# Dependent variable price which is our target
y = df['price']

# Split the data into training and testing sets
# We will use 20% of the data for testing and 80% for training.

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

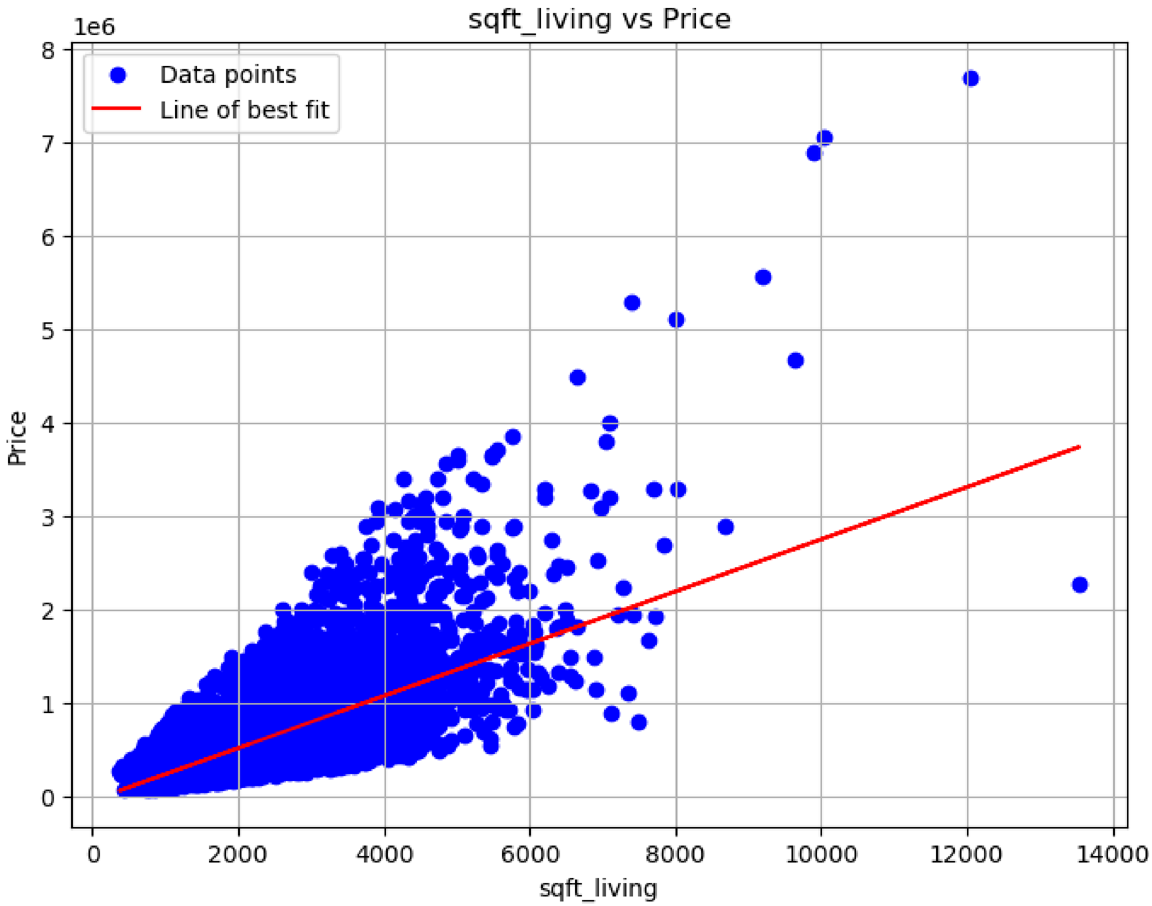
B. Simple Linear Regression for each Independent Variable

In [25]:
```
# We are going to create a function for plotting a simple linear regression mode
# This will make the plotting of linear regression models for all our features e

# Importing necessary libraries
import statsmodels.api as sm
import seaborn as sns

# Creating a function for simple linear regression
def simple_linear_regression(X, y):
    # Add constant to X
    X = sm.add_constant(X)

    # Fit OLS regression model
    model = sm.OLS(y, X).fit()

    # Get the coefficients
    intercept = model.params.iloc[0]
    slope = model.params.iloc[1]

    # Predictions
    y_pred = model.predict(X)

    # Plotting
    # plt.figure(figsize=(6, 4))
    # sns.scatterplot(x=X.iloc[:, 1], y=y, alpha=0.5)
    # sns.lineplot(x=X.iloc[:, 1], y=y_pred, color='red')

    # plt.title(f'{X.columns[1]} vs Price')
    # plt.xlabel(X.columns[1])
    # plt.ylabel('Price')
    # plt.show()

    plt.figure(figsize=(8, 6))  #Set figure size
    plt.scatter(X.iloc[:, 1], y ,color="blue", label="Data points") #original da
    plt.plot(X.iloc[:, 1], y_pred, color="red", label="Line of best fit") ## not
    plt.title(f'{X.columns[1]} vs Price')
    plt.xlabel(X.columns[1])
    plt.ylabel('Price')
    plt.legend()
    plt.grid(True)
    plt.show()

    # Print the summary of the regression results
    print(model.summary())
    print('')
    print(f"price = {slope}*({X.columns[1]}) + {intercept}, This is in the form
```

```
    print("f_value:", model.fvalue)
    print("p_value:", model.f_pvalue)
    print('')
```

i. sqft_living

In [26]:
```
X_sqft_living = X_train[['sqft_living']]
y = y_train
simple_linear_regression(X_sqft_living, y)
```



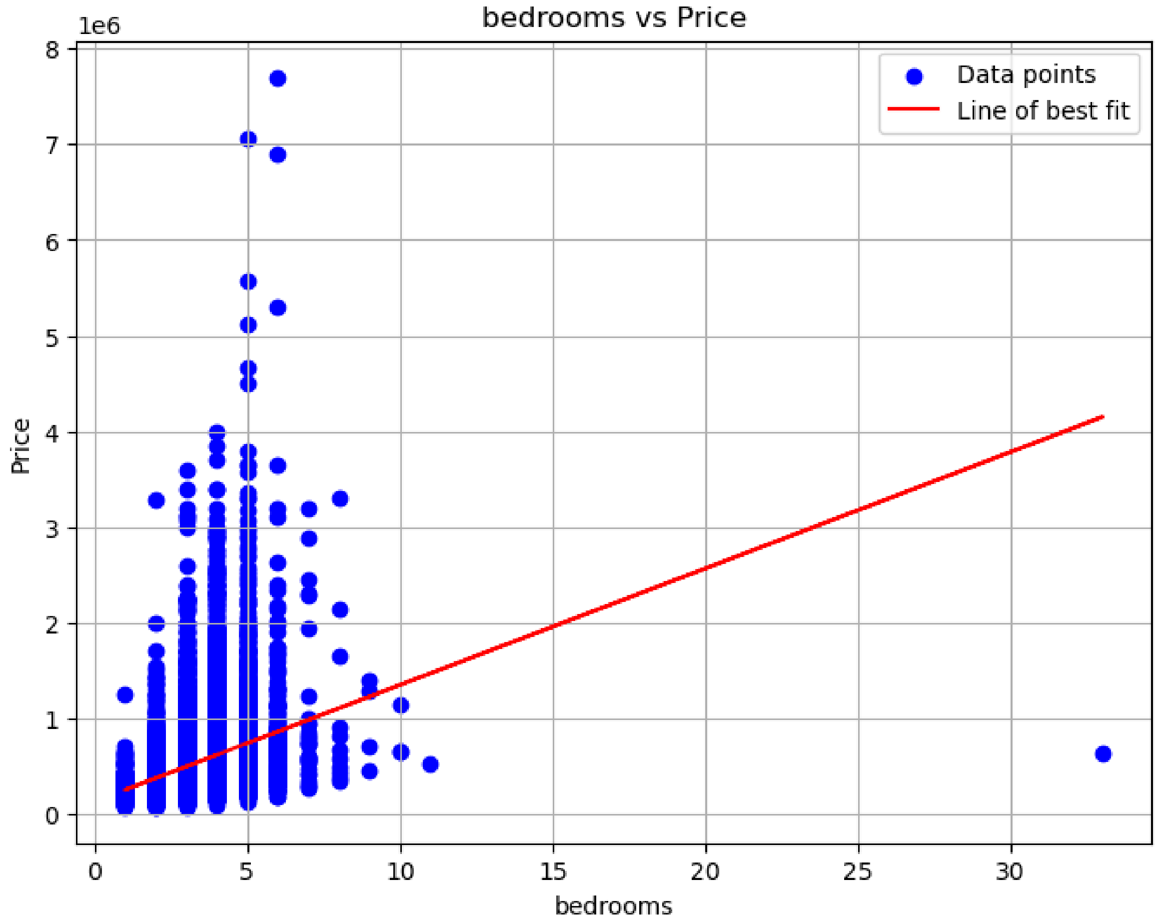
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.489
Model:                            OLS   Adj. R-squared:                  0.489
Method:                 Least Squares   F-statistic:                 1.656e+04
Date:                Tue, 09 Apr 2024   Prob (F-statistic):               0.00
Time:                        18:37:11   Log-Likelihood:            -2.4012e+05
No. Observations:               17277   AIC:                         4.802e+05
Df Residuals:                   17275   BIC:                         4.803e+05
Df Model:                           1
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
const       -4.074e+04   4945.683     -8.238      0.000   -5.04e+04   -3.11e+04
sqft_living   279.2624      2.170    128.675      0.000     275.008     283.516
==============================================================================
Omnibus:                    12143.926   Durbin-Watson:                   2.010
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           489855.060
Skew:                           2.894   Prob(JB):                         0.00
Kurtosis:                      28.435   Cond. No.                     5.64e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 5.64e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

price = 279.262427562963*(sqft_living) + -40744.59249028359, This is in the form
y = mx+c
f_value: 16557.204671729098
p_value: 0.0
```

- The coefficient for sqft_living is approximately 279. For every 1 unit increase in square footage of living space (sqft_living), the house price is estimated to increase by $279, whereas, while other sqft_living is constant the house price is expected to decrease by 40744.
- Given a p-value of 0.000, the coefficient for sqft_living is statistically significant. This means that the sqft_living space has a significant impact on the house price.
- The R-squared value is 0.489, indicating that ~ 49.2% of the variance in house prices is explained by the sqft_living space. This suggests that the model provides a moderate fit to the data.
- The F-statistic is 1.656e+04, with a p-value of 0.000, which means that our regression model is statistically significant.

ii. Bedrooms

In [27]:
```
# Call the function for bedrooms
X_bedrooms = X_train[['bedrooms']]
print("Regression results for bedrooms:")
simple_linear_regression(X_bedrooms, y)
```

```
Regression results for bedrooms:
```




```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.095
Model:                            OLS   Adj. R-squared:                  0.095
Method:                 Least Squares   F-statistic:                     1820.
Date:                Tue, 09 Apr 2024   Prob (F-statistic):               0.00
Time:                        18:37:12   Log-Likelihood:            -2.4506e+05
No. Observations:               17277   AIC:                         4.901e+05
Df Residuals:                   17275   BIC:                         4.901e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       1.301e+05      1e+04     13.006      0.000     1.1e+05     1.5e+05
bedrooms    1.218e+05   2855.071     42.659      0.000    1.16e+05    1.27e+05
==============================================================================
Omnibus:                    15325.768   Durbin-Watson:                   2.012
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1091222.015
Skew:                           3.965   Prob(JB):                         0.00
Kurtosis:                      41.118   Cond. No.                         14.2
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.

price = 121794.12831776463*(bedrooms) + 130065.79490555076, This is in the form y
= mx+c
f_value: 1819.779672482757
p_value: 0.0
```

- The coefficient for bedrooms is around 121,794. Meaning additional bedroom in a house, the price of the house might increase by USD 121,794, whereas if there are no other changes in bedrooms, the the value of the house will be 130065.
- The p-value of 0.000 means the coefficient for bedrooms is statistically significant
- The R-squared value is 0.095, meaning ~ 9.5% of the variance in house prices is explained by the number of bedrooms.
- The F-statistic is 1820.0, with a p-value of 0.000, meaning the regression model is statistically significant.

iii. Bathrooms

In [28]:
```
# Call the function for bathrooms
X_bathrooms = X_train[['bathrooms']]
print("Regression results for bathrooms:")
simple_linear_regression(X_bathrooms, y)
```

```
Regression results for bathrooms:
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.276
Model:                            OLS   Adj. R-squared:                  0.275
Method:                 Least Squares   F-statistic:                     6569.
Date:                Tue, 09 Apr 2024   Prob (F-statistic):               0.00
Time:                        18:37:12   Log-Likelihood:            -2.4314e+05
No. Observations:               17277   AIC:                         4.863e+05
Df Residuals:                   17275   BIC:                         4.863e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       1.141e+04   6958.084      1.640      0.101   -2228.620     2.5e+04
bathrooms     2.5e+05   3084.344     81.052      0.000    2.44e+05    2.56e+05
==============================================================================
Omnibus:                    14070.987   Durbin-Watson:                   2.027
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           794843.170
Skew:                           3.522   Prob(JB):                         0.00
Kurtosis:                      35.473   Cond. No.                         7.76
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.

price = 249991.5449970878*(bathrooms) + 11409.928954557778, This is in the form y
= mx+c
f_value: 6569.388742409433
p_value: 0.0
```

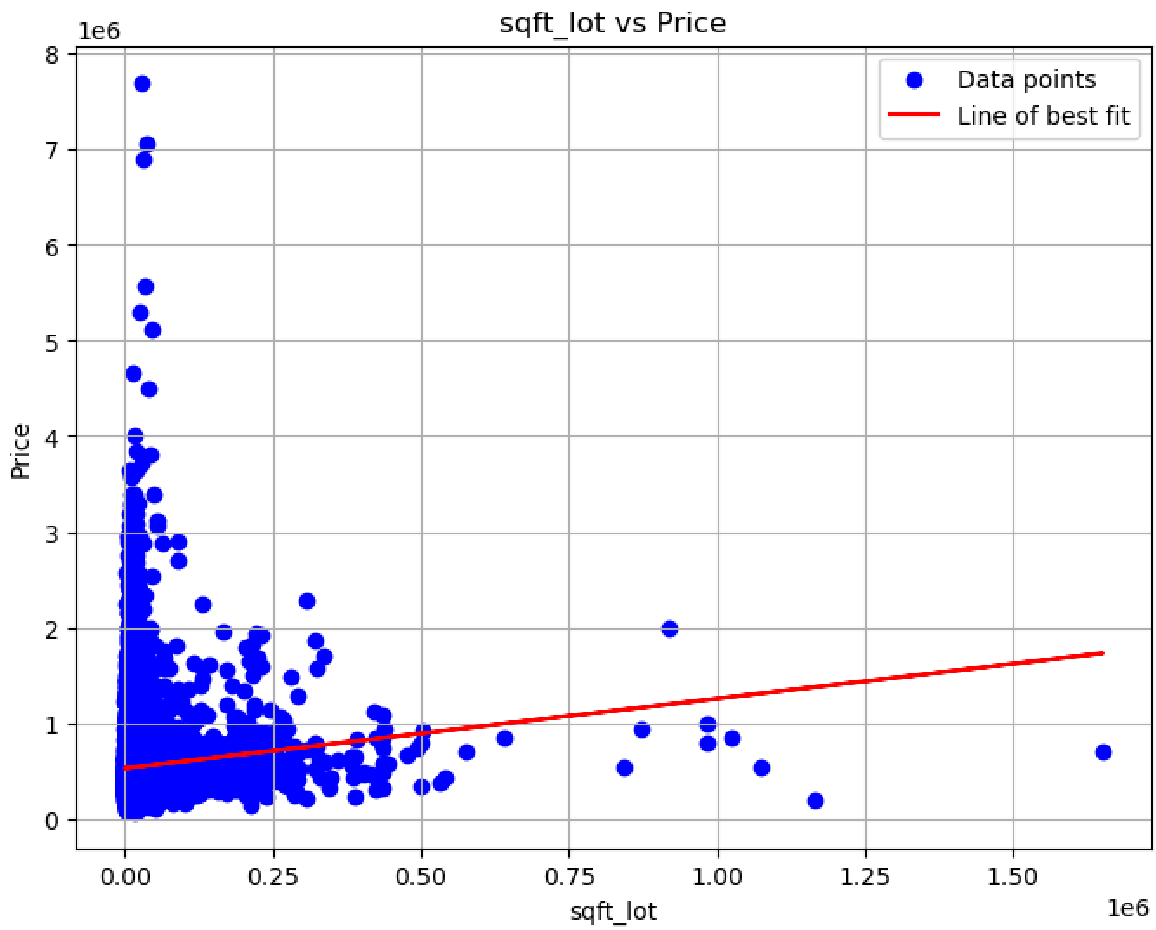
- The coefficient for bathrooms is approximately 250,000. This means that for an additional bathroom in a house, the price of the house will increase by USD 250,000 whereas if no renovation is done in bathrooms, then the value of the house is expected to be 11409.
- The coefficient for bathrooms is statistically significant, as indicated by the p-value of 0.000.
- This suggests that the number of bathrooms has a significant impact on the house price.

iv. Sqft Lot

In [29]:

```
# Call the function for sqft_lot
X_sqft_lot = X_train[['sqft_lot']]
print("Regression results for sqft_lot:")
simple_linear_regression(X_sqft_lot, y)
```

```
Regression results for sqft_lot:
```


sqft_lot vs Price
1e6
8
7
6
5
4
3
2
1
0
Price
Data points
Line of best fit
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1e6
sqft_lot

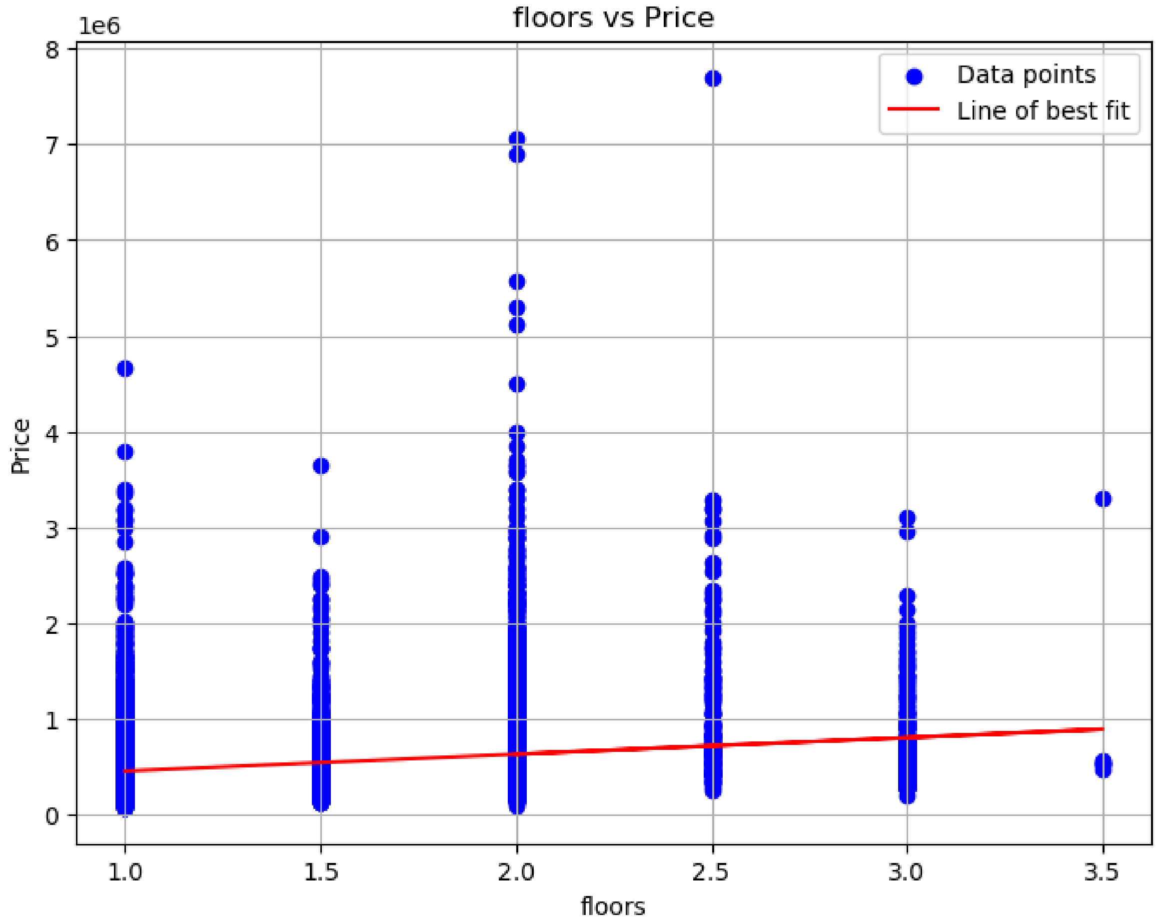
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.007
Model:                            OLS   Adj. R-squared:                  0.007
Method:                 Least Squares   F-statistic:                     119.0
Date:                Tue, 09 Apr 2024   Prob (F-statistic):           1.26e-27
Time:                        18:37:12   Log-Likelihood:            -2.4587e+05
No. Observations:               17277   AIC:                         4.917e+05
Df Residuals:                   17275   BIC:                         4.918e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       5.303e+05   2966.401    178.759      0.000    5.24e+05    5.36e+05
sqft_lot       0.7275      0.067     10.911      0.000       0.597       0.858
==============================================================================
Omnibus:                    15595.690   Durbin-Watson:                   2.006
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1044159.618
Skew:                           4.109   Prob(JB):                         0.00
Kurtosis:                      40.188   Cond. No.                     4.73e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 4.73e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

price = 0.7274845493935727*(sqft_lot) + 530271.7004753138, This is in the form y
= mx+c
f_value: 119.0405586196337
p_value: 1.2627869637889872e-27
```

- The coefficient for sqft_lot is approximately 0.727, meaning a one-unit increase in the square footage of the lot, the price of the house will increase by USD 0.727, if no other change is sqft_lot is done in the house, then the value of the home is expected to be 530271.
- Given the p-value of 0.000, This suggests that the square footage of the lot has a significant impact on the house price.
- Approximately 0.8% of the variance in house prices is explained by the square footage of the lot (sqft_lot).
- The overall regression model is statistically significant.

v. Floors

In [30]:
```
# Call the function for floors
X_floors = X_train[['floors']]
print("Regression results for floors:")
simple_linear_regression(X_floors, y)
```

```
Regression results for floors:
```



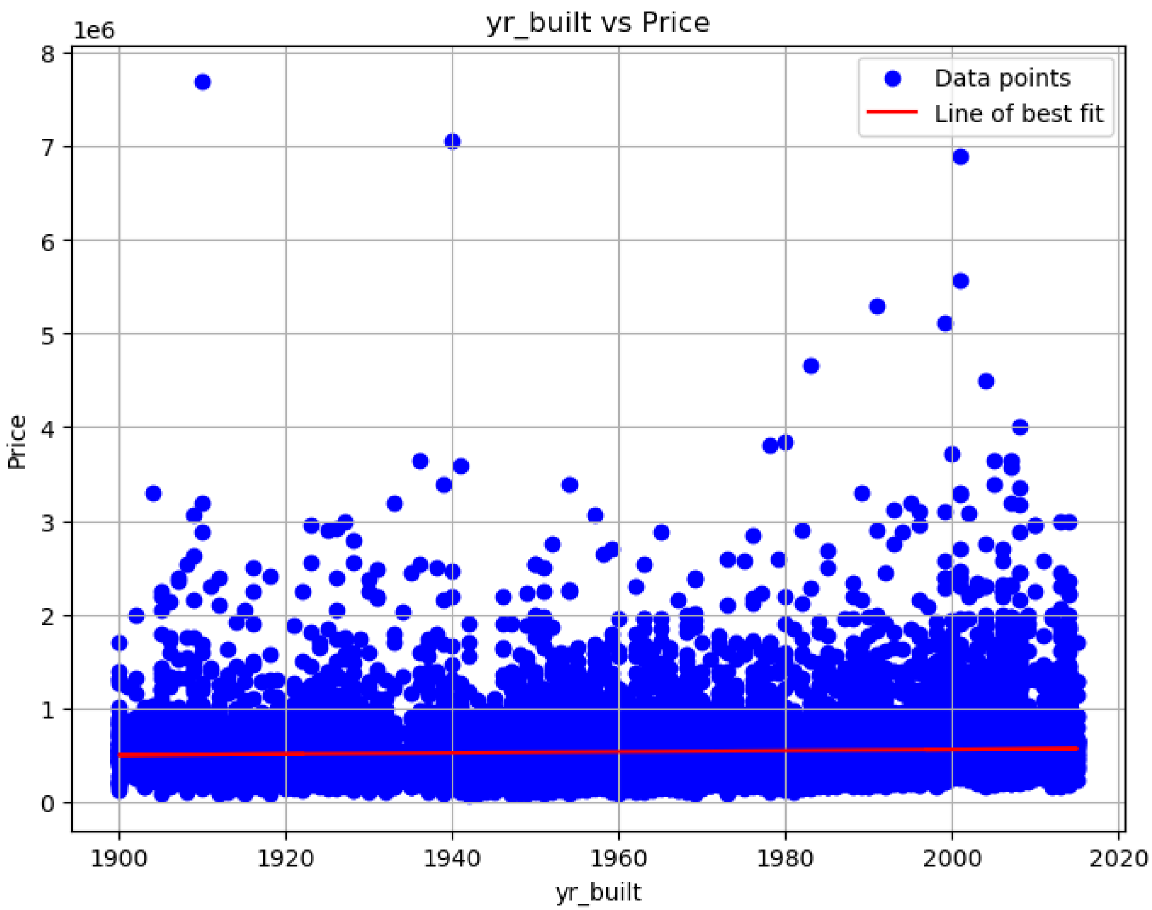

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.066
Model:                            OLS   Adj. R-squared:                  0.066
Method:                 Least Squares   F-statistic:                     1213.
Date:                Tue, 09 Apr 2024   Prob (F-statistic):          5.31e-257
Time:                        18:37:13   Log-Likelihood:            -2.4534e+05
No. Observations:               17277   AIC:                         4.907e+05
Df Residuals:                   17275   BIC:                         4.907e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       2.801e+05   7971.267     35.139      0.000    2.64e+05    2.96e+05
floors      1.745e+05   5008.961     34.834      0.000    1.65e+05    1.84e+05
==============================================================================
Omnibus:                    15790.117   Durbin-Watson:                   2.008
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1148912.256
Skew:                           4.163   Prob(JB):                         0.00
Kurtosis:                      42.073   Cond. No.                         6.38
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.

price = 174483.3344052792*(floors) + 280103.3012835714, This is in the form y = m
x+c
f_value: 1213.423949281076
p_value: 5.308358683083986e-257
```

- The coefficient for floors is approximately 174,483. This means that for every additional floor in a house, the price of the house will increase by USD 174,483, if floor(floor modification) is not changed, then the value of the house is expected to be 280103.
- As indicated by the p-value of 0.000, the number of floors has a significant impact on the house price.
- Around 6.7% of the variance in house prices is explained by the number of floors (floors)
- The overall regression model is statistically significant given the F-statistic

vi. Year Built

In [31]:
```
# Call the function for yr_built
X_yr_built = X_train[['yr_built']]
print("Regression results for yr_built:")
simple_linear_regression(X_yr_built, y)
```

```
Regression results for yr_built:
```



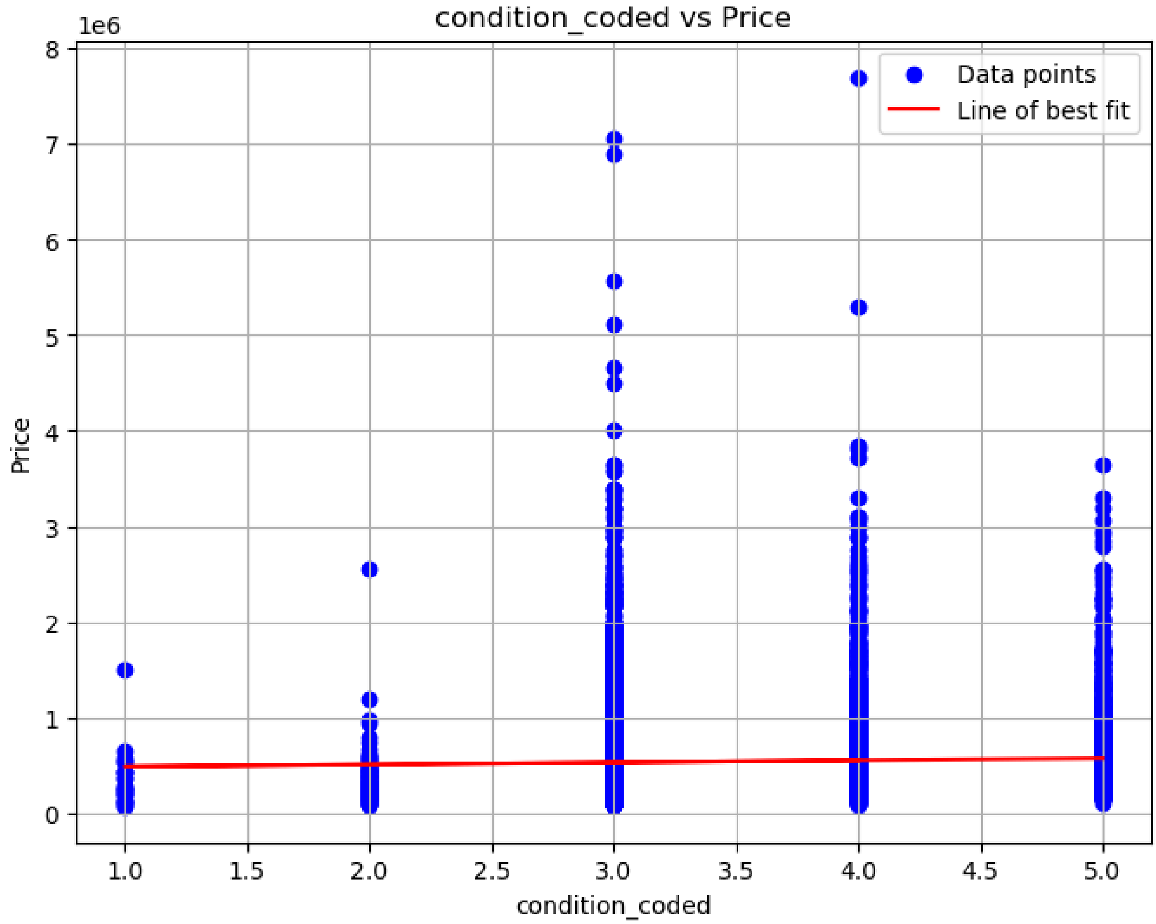
```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                  price   R-squared:                       0.002
Model:                            OLS   Adj. R-squared:                  0.002
Method:                 Least Squares   F-statistic:                     40.19
Date:                Tue, 09 Apr 2024   Prob (F-statistic):           2.36e-10
Time:                        18:37:13   Log-Likelihood:            -2.4591e+05
No. Observations:               17277   AIC:                         4.918e+05
Df Residuals:                   17275   BIC:                         4.918e+05
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -6.465e+05   1.87e+05     -3.450      0.001   -1.01e+06   -2.79e+05
yr_built     602.6068     95.054      6.340      0.000     416.292     788.922
==============================================================================
Omnibus:                    15607.659   Durbin-Watson:                   2.005
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1044645.468
Skew:                           4.115   Prob(JB):                         0.00
Kurtosis:                      40.194   Cond. No.                     1.32e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.32e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

price = 602.6068272638458*(yr_built) + -646471.1768189413, This is in the form y = mx+c
f_value: 40.19104577848836
p_value: 2.3601119184005135e-10
```

- The coefficient for yr_built is approximately 602.61. For every additional year since the year built, the price of the house is estimated to increase by USD 602.61, if year is not considered, then still the value of the house is expected to decrease by 646471.
- The year the house was built has a significant influence on the house price.
- 0.2% of the variance in house prices is explained by the year the house was built
- The regression model is statistically significant given the F-statistic

vii. Condition Coded

In [32]:
```
# Call the function for condition_coded
X_condition_coded = X_train[['condition_coded']]
print("Regression results for condition_coded:")
simple_linear_regression(X_condition_coded, y)
```

```
Regression results for condition_coded:
```


condition_coded vs Price
1e6
8
7
6
5
4
3
2
1
0
Price
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
condition_coded
Data points
Line of best fit

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.001
Model:                            OLS   Adj. R-squared:                  0.001
Method:                 Least Squares   F-statistic:                     24.91
Date:                Tue, 09 Apr 2024   Prob (F-statistic):           6.08e-07
Time:                        18:37:13   Log-Likelihood:            -2.4591e+05
No. Observations:               17277   AIC:                         4.918e+05
Df Residuals:                   17275   BIC:                         4.918e+05
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
==
                     coef    std err          t      P>|t|      [0.025      0.97
5]
--------------------------------------------------------------------------------
--
const           4.681e+05   1.49e+04     31.367      0.000    4.39e+05    4.97e+
05
condition_coded 2.146e+04   4300.180      4.991      0.000     1.3e+04    2.99e+
04
==============================================================================
Omnibus:                    15577.833   Durbin-Watson:                   2.003
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1036374.512
Skew:                           4.104   Prob(JB):                         0.00
Kurtosis:                      40.045   Cond. No.                         20.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.

price = 21460.518165208247*(condition_coded) + 468134.58549233497, This is in the
form y = mx+c
f_value: 24.906181000021636
p_value: 6.077447064239542e-07
```

- The coefficient for condition_coded is approximately USD 21,460. If the condition improves for example from good to very good, the price of the house might increase by USD 21,460, if no change in the condition of the house, then the value of the house will be 468134
- This suggests that the condition code of the house has an impact on the house price.
- ~ 0.2% of the variance in house prices is explained by the condition code of the house
- The regression model is statistically significant.

C. Multiple Linear Regression

We use training data to model our multiple linear regression

In [33]:
```
import statsmodels.api as sm

# Add a constant term to the independent variables (required for OLS regression)
X_train_ols = sm.add_constant(X_train)
```

```
# Create and fit the OLS model
ols_model = sm.OLS(y_train, X_train_ols)
ols_results = ols_model.fit()

# Print the summary of the OLS regression results
print(ols_results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.556
Model:                            OLS   Adj. R-squared:                  0.555
Method:                 Least Squares   F-statistic:                     3084.
Date:                Tue, 09 Apr 2024   Prob (F-statistic):               0.00
Time:                        18:37:13   Log-Likelihood:            -2.3892e+05
No. Observations:               17277   AIC:                         4.779e+05
Df Residuals:                   17269   BIC:                         4.779e+05
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
==
                     coef    std err          t      P>|t|      [0.025      0.97
5]
--------------------------------------------------------------------------------
--
const           6.342e+06   1.62e+05     39.263      0.000    6.03e+06    6.66e+
06
bedrooms       -6.779e+04   2503.940    -27.072      0.000   -7.27e+04   -6.29e+
04
bathrooms       6.513e+04   4315.497     15.093      0.000    5.67e+04    7.36e+
04
sqft_living      302.1749      3.368     89.723      0.000     295.574     308.7
76
sqft_lot          -0.3332      0.046     -7.310      0.000      -0.423      -0.2
44
floors          5.817e+04   4264.216     13.642      0.000    4.98e+04    6.65e+
04
yr_built       -3290.3978     81.902    -40.175      0.000   -3450.934   -3129.8
62
condition_coded 1.869e+04   3122.388      5.987      0.000    1.26e+04    2.48e+
04
==============================================================================
Omnibus:                    11729.996   Durbin-Watson:                   2.002
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           466829.361
Skew:                           2.748   Prob(JB):                         0.00
Kurtosis:                      27.865   Cond. No.                     3.85e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.85e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Summary of the Multiple Regression Model

Model Performance

- The R-squared value of 0.556 indicates that ~ 55.6% of the variance in the price is explained by the independent variables included in the model.
- A high F-statistic of 3084 and a low Prob(F-statistic)) means that our model is statistically significant.

- Intercept (const): The intercept is the estimated value of the dependent variable when all independent variables are set to zero. In this case, it's approximately USD 6.35 million.

Coefficients for independent variables:

- bedrooms: For each additional bedroom, there is an estimated decrease of approximately USD 67,790 in the price.
- bathrooms: For each additional bathroom, there is an estimated increase of approximately USD 65,1300 in the price.
- sqft_living: For each additional square foot of living space, there is an estimated increase of approximately USD 302.17 in the price.
- sqft_lot: For each additional square foot of lot size, there is an estimated decrease of approximately USD 0.333 in the price.
- floors: For each additional floor, there is an estimated increase of approximately USD 58,170 in the price.
- yr_built: For each additional year of the house's age, there is an estimated decrease of approximately USD 3,290 in the price.
- condition_coded: Improvement of the overall condition of the house from 1 rating to another (e.g. good to very good) will increase the price of the house by approximately USD 19,610

Summary

- The high condition number suggests potential multicollinearity or numerical problems in the model.

D. Multicollinearity

i. Exploring Data for Multicollinearity

In [34]:
```
# Previewing our data - independent variables
X_train.head(2)
```

Out[34]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	yr_built	condition_coded
2093	4	2.0	2130	2800	1.0	1922	5
9738	3	1.0	1160	3700	1.5	1909	3

In [35]:
```
# Training data preview - target variable
y_train.head(2)
```

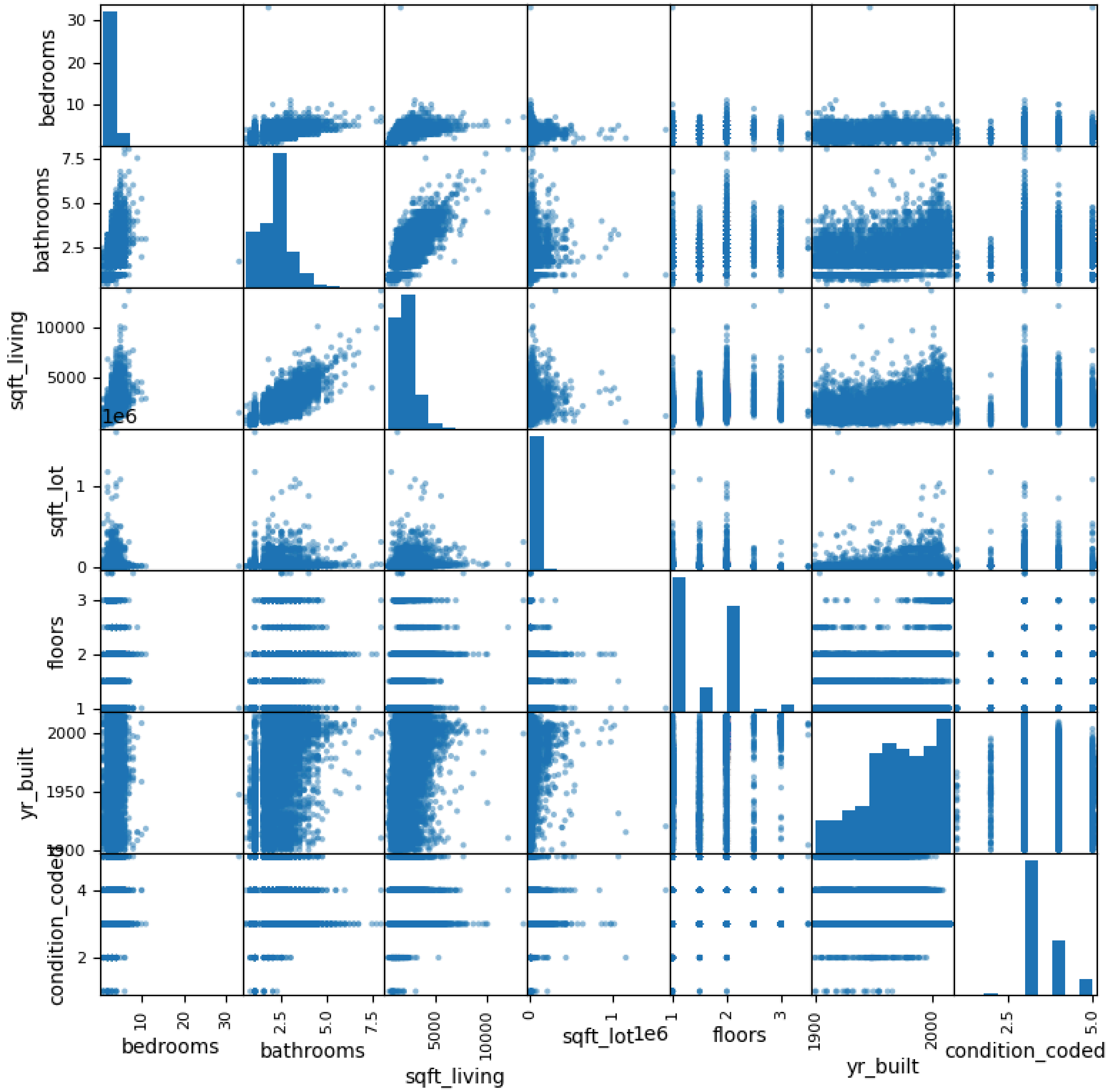
Out[35]:
```
2093    800000.0
9738    315000.0
Name: price, dtype: float64
```

In [36]:
```
X = X_train
X.corr()
```

Out[36]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	yr_built	co
bedrooms	1.000000	0.511014	0.577830	0.034186	0.180796	0.158268	
bathrooms	0.511014	1.000000	0.758566	0.085941	0.503578	0.507776	
sqft_living	0.577830	0.758566	1.000000	0.169754	0.356850	0.318708	
sqft_lot	0.034186	0.085941	0.169754	1.000000	-0.002811	0.050062	
floors	0.180796	0.503578	0.356850	-0.002811	1.000000	0.488767	
yr_built	0.158268	0.507776	0.318708	0.050062	0.488767	1.000000	
condition_coded	0.022288	-0.134546	-0.062333	-0.010503	-0.262695	-0.366237	

In [37]:
```
# Scatterplot for our independent variables
pd.plotting.scatter_matrix(X,figsize  = [9, 9]);
plt.show()
```



In [38]:
```
## Getting the correlation of the independent variables
X.corr()
```

Out[38]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	yr_built	co
bedrooms	1.000000	0.511014	0.577830	0.034186	0.180796	0.158268	
bathrooms	0.511014	1.000000	0.758566	0.085941	0.503578	0.507776	
sqft_living	0.577830	0.758566	1.000000	0.169754	0.356850	0.318708	
sqft_lot	0.034186	0.085941	0.169754	1.000000	-0.002811	0.050062	
floors	0.180796	0.503578	0.356850	-0.002811	1.000000	0.488767	
yr_built	0.158268	0.507776	0.318708	0.050062	0.488767	1.000000	
condition_coded	0.022288	-0.134546	-0.062333	-0.010503	-0.262695	-0.366237	

Generally, a correlation with an absolute value around 0.7-0.8 or higher is considered a high correlation. We will use 0.75 as our cut-off

In [39]:
```
# Checking how many correlations have is more than 0.75
abs(X.corr()) > 0.75
```

Out[39]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	yr_built	condition
bedrooms	True	False	False	False	False	False	
bathrooms	False	True	True	False	False	False	
sqft_living	False	True	True	False	False	False	
sqft_lot	False	False	False	True	False	False	
floors	False	False	False	False	True	False	
yr_built	False	False	False	False	False	True	
condition_coded	False	False	False	False	False	False	

"bathrooms" and "sqft_living" are highly correlated. Also, This relationship may influence regression model stability and interpretation.

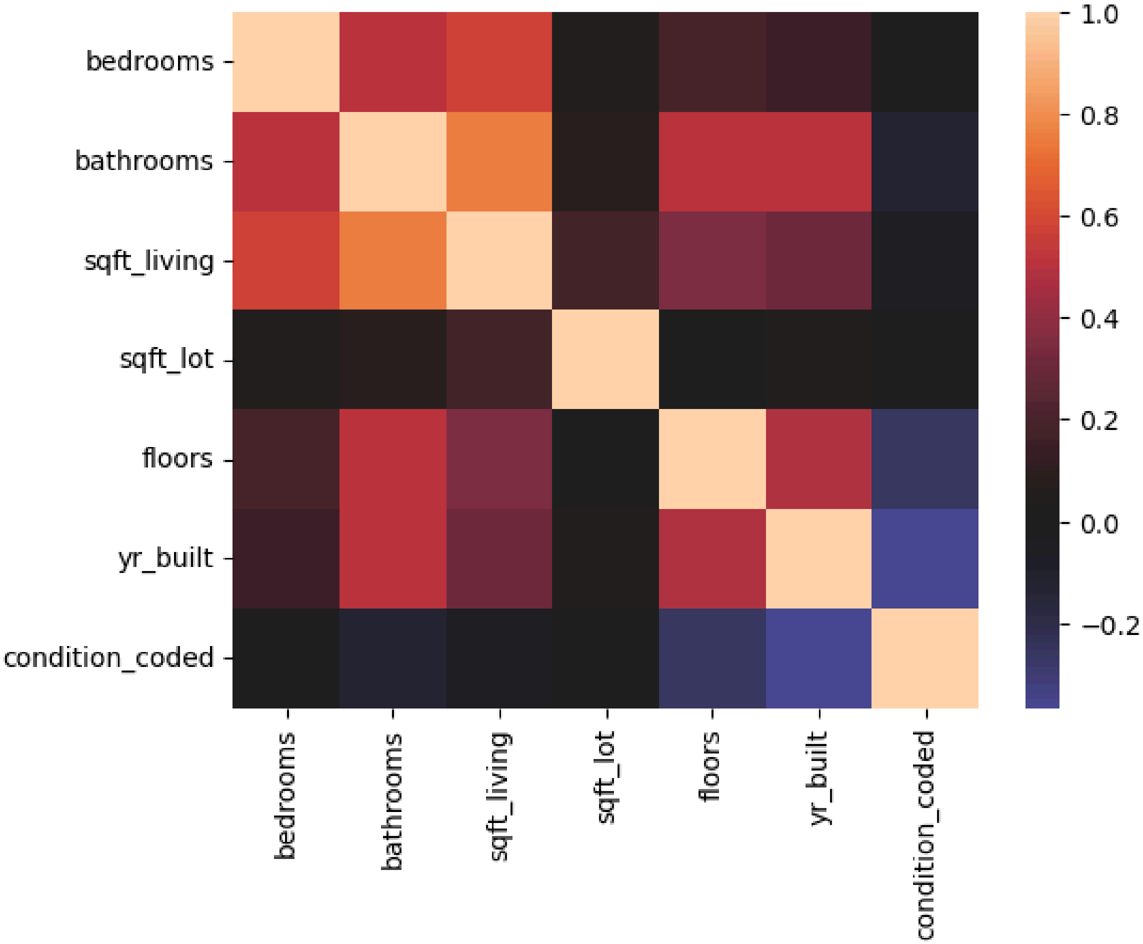
In [40]:
```
df2=X.corr().abs().stack().reset_index().sort_values(0, ascending=False)

# zip the variable name columns (Which were only named level_0 and level_1 by de
df2['pairs'] = list(zip(df2.level_0, df2.level_1))

# set index to pairs
df2.set_index(['pairs'], inplace = True)

#d rop level columns
df2.drop(columns=['level_1', 'level_0'], inplace = True)

# rename correlation column as cc rather than 0
df2.columns = ['cc']

df2.drop_duplicates(inplace=True)
```

In [41]:
```
# Returning pairs that are highly correlated

df2[(df2.cc>.75) & (df2.cc <1)]
```

Out[41]:

	cc
pairs	
(bathrooms, sqft_living)	0.758566

In [42]:
```
## Lets use heatmap to check the correlation

import seaborn as sns
sns.heatmap(X.corr(), center=0);
```



In [43]:
```
# Preview the new df
X.head()
```

Out[43]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	yr_built	condition_coded
2093	4	2.00	2130	2800	1.0	1922	5
9738	3	1.00	1160	3700	1.5	1909	3
4382	3	1.75	1820	15570	1.0	1948	3
11641	3	1.75	1660	8301	1.0	1955	5
13114	2	2.25	1390	1222	3.0	2009	3

In [44]:
```
# Create new df
data = X.copy()
data.head()
```

Out[44]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	yr_built	condition_coded
2093	4	2.00	2130	2800	1.0	1922	5
9738	3	1.00	1160	3700	1.5	1909	3
4382	3	1.75	1820	15570	1.0	1948	3
11641	3	1.75	1660	8301	1.0	1955	5
13114	2	2.25	1390	1222	3.0	2009	3

We will create a new column called bathroom_density to hold the ratio of bathrooms to the number of bedrooms

In [45]:
```
# We will only address the 2 highly correlated columns bathrooms and sqft_living
# The ratio of bathrooms to the number of bedrooms

data['bathroom_density'] = data['bathrooms'] / data['bedrooms']
```

In [46]:
```
# Preview our new data frame
data.head()
```

Out[46]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	yr_built	condition_coded	b
2093	4	2.00	2130	2800	1.0	1922	5	
9738	3	1.00	1160	3700	1.5	1909	3	
4382	3	1.75	1820	15570	1.0	1948	3	
11641	3	1.75	1660	8301	1.0	1955	5	
13114	2	2.25	1390	1222	3.0	2009	3	

ii. Dropping column bathrooms

"Bathrooms" has a correlation coefficient of 0.76 with "sqft_living", indicating a high positive correlation. "Bathrooms" has a correlation coefficient of 0.51 with "bedrooms", indicating a moderate positive correlation

Also

"Bathrooms" has a correlation coefficient of 0.525906 with "price", indicating a moderate positive correlation. "Sqft_living" has a higher correlation coefficient of 0.701917 with "price", indicating a stronger positive correlation.

So we will drop the bathrooms column

In [47]:
```
# Drop the 'bathrooms' column from the DataFrame
data.drop('bathrooms', axis=1, inplace=True)
data.head()
```

Out[47]:

	bedrooms	sqft_living	sqft_lot	floors	yr_built	condition_coded	bathroom_den
2093	4	2130	2800	1.0	1922	5	0.500
9738	3	1160	3700	1.5	1909	3	0.333
4382	3	1820	15570	1.0	1948	3	0.583
11641	3	1660	8301	1.0	1955	5	0.583
13114	2	1390	1222	3.0	2009	3	1.125

In [48]:
```
data.corr()
```

Out[48]:

	bedrooms	sqft_living	sqft_lot	floors	yr_built	condition_cod
bedrooms	1.000000	0.577830	0.034186	0.180796	0.158268	0.0222
sqft_living	0.577830	1.000000	0.169754	0.356850	0.318708	-0.0623
sqft_lot	0.034186	0.169754	1.000000	-0.002811	0.050062	-0.0105
floors	0.180796	0.356850	-0.002811	1.000000	0.488767	-0.2626
yr_built	0.158268	0.318708	0.050062	0.488767	1.000000	-0.3662
condition_coded	0.022288	-0.062333	-0.010503	-0.262695	-0.366237	1.0000
bathroom_density	-0.234908	0.311833	0.062219	0.418674	0.425651	-0.1629

In [49]:

```
## Rechecking the correlation between the features
abs(data.corr()) > 0.75
```

Out[49]:

	bedrooms	sqft_living	sqft_lot	floors	yr_built	condition_coded	ba
bedrooms	True	False	False	False	False	False	
sqft_living	False	True	False	False	False	False	
sqft_lot	False	False	True	False	False	False	
floors	False	False	False	True	False	False	
yr_built	False	False	False	False	True	False	
condition_coded	False	False	False	False	False	True	
bathroom_density	False	False	False	False	False	False	

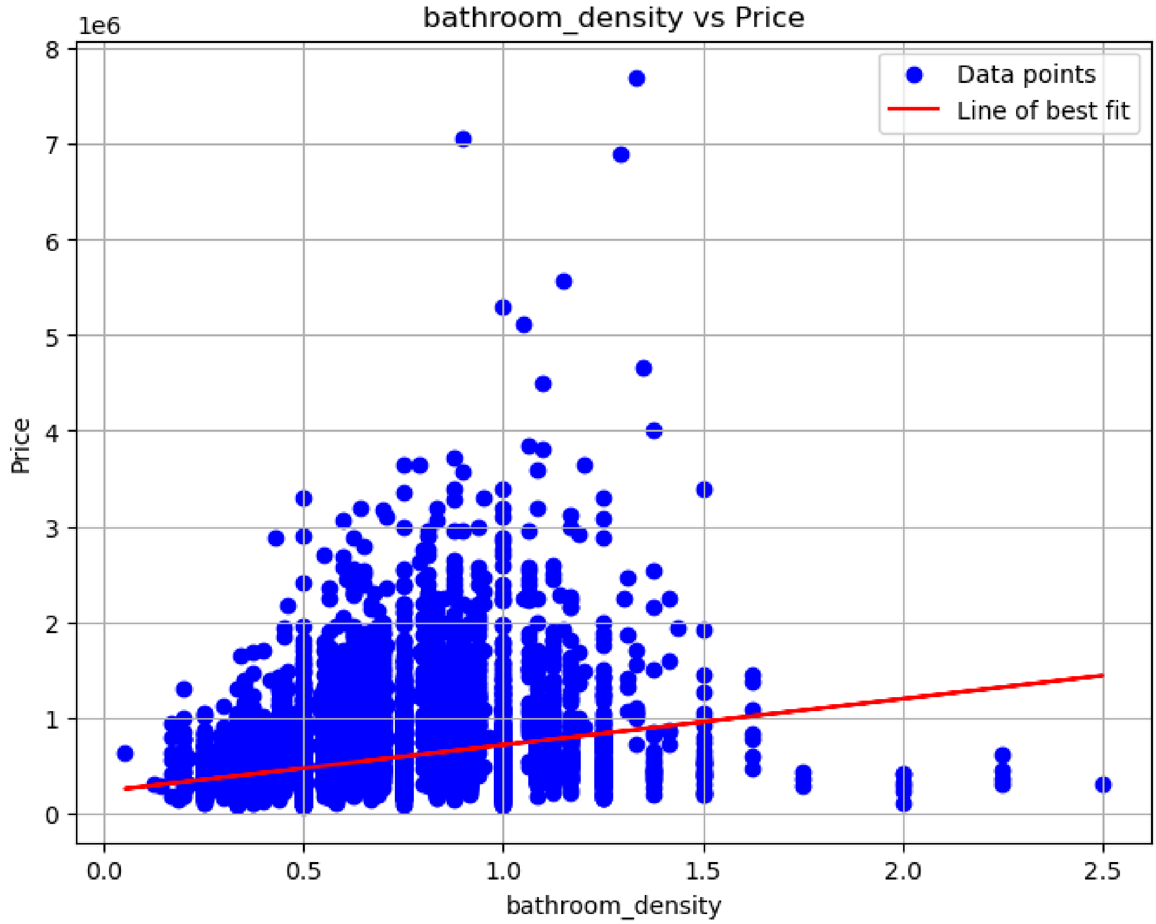
In [50]:

```
# creating another copy of the data as X-Train
X_train = data.copy()
```

In [51]:

```
#### Exploring the newly created column
X_bathdensity = X_train[['bathroom_density']]
print("Regression results for yr_built:")
simple_linear_regression(X_bathdensity, y)
```

```
Regression results for yr_built:
```


bathroom_density vs Price
1e6
Price
bathroom_density
Data points
Line of best fit
0
1
2
3
4
5
6
7
8
0.0
0.5
1.0
1.5
2.0
2.5

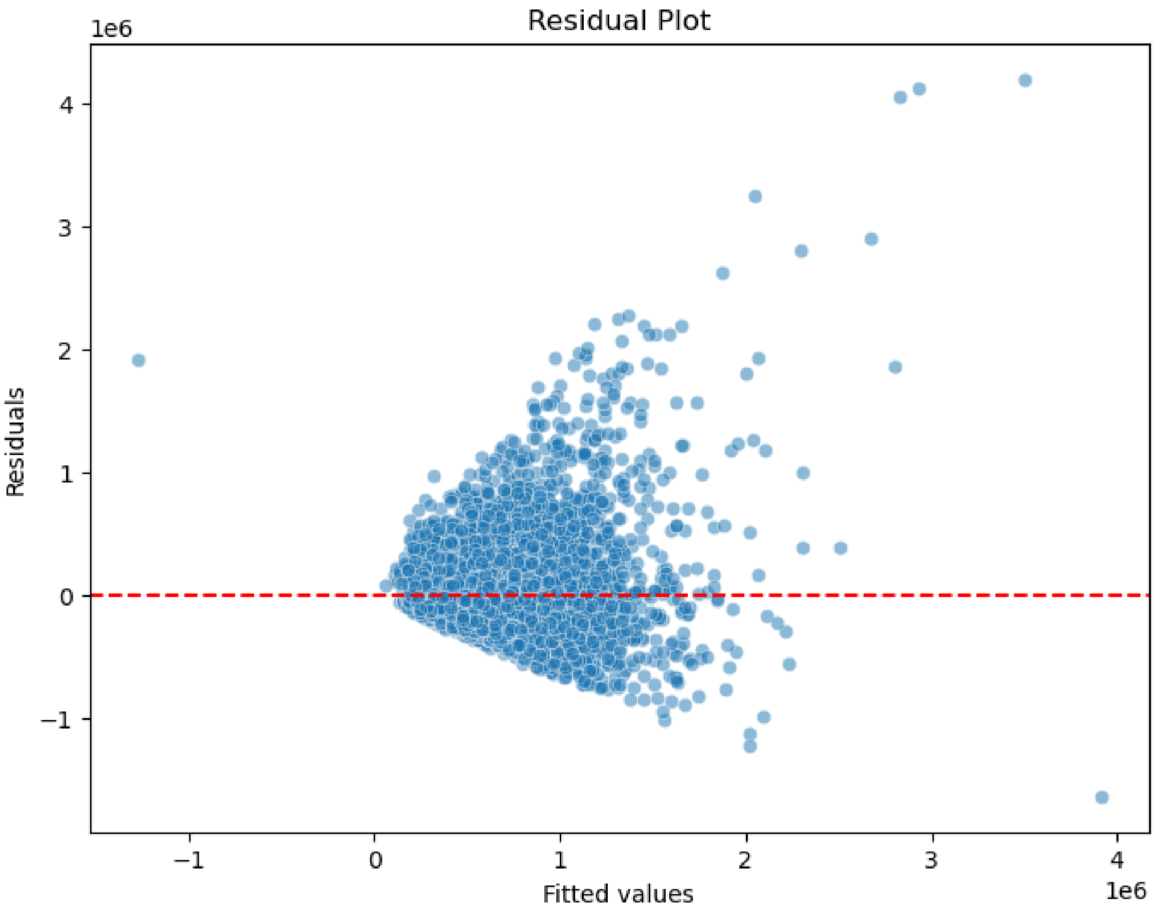
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.078
Model:                            OLS   Adj. R-squared:                  0.078
Method:                 Least Squares   F-statistic:                     1460.
Date:                Tue, 09 Apr 2024   Prob (F-statistic):          1.18e-306
Time:                        18:37:17   Log-Likelihood:            -2.4523e+05
No. Observations:               17277   AIC:                         4.905e+05
Df Residuals:                   17275   BIC:                         4.905e+05
Df Model:                           1
Covariance Type:            nonrobust
=================================================================================
===
                       coef    std err          t      P>|t|      [0.025      0.9
75]
---------------------------------------------------------------------------------
---
const             2.311e+05   8551.700     27.028      0.000    2.14e+05    2.48e
+05
bathroom_density  4.835e+05   1.27e+04     38.204      0.000    4.59e+05    5.08e
+05
==============================================================================
Omnibus:                    15134.349   Durbin-Watson:                   2.007
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           982780.240
Skew:                           3.918   Prob(JB):                         0.00
Kurtosis:                      39.108   Cond. No.                         6.71
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.

price = 483465.1676656219*(bathroom_density) + 231135.55331064545, This is in the
form y = mx+c
f_value: 1459.5795917258044
p_value: 1.1842970305104985e-306
```

iii. Feature Selections and modelling

In [52]:
```python
# Concatenate X_train and y_train into a single data frame (We need both price a

train_data = pd.concat([X_train, y_train], axis=1)

# Calculate the p correlation coefficient
correlation_matrix = train_data.corr()

# Extract the correlations
correlation_with_y = correlation_matrix['price'].drop('price')
correlation = correlation_with_y.sort_values(ascending=False)

correlation
```

Out[52]:
```
sqft_living          0.699565
bedrooms             0.308711
bathroom_density     0.279121
floors               0.256187
sqft_lot             0.082727
yr_built             0.048178
condition_coded      0.037943
Name: price, dtype: float64
```

In [53]:
```
# Multiple Regression (top 4 most corr)
# Extract the feature variables and target variable

X = X_train[['sqft_living', 'bedrooms', 'bathroom_density','floors']]
y = y_train

X = sm.add_constant(X)

# Fit OLS regression model
select_model = sm.OLS(y, X).fit()

# Scatter plot
plt.figure(figsize=(8, 6))
sns.scatterplot(x=select_model.fittedvalues, y=select_model.resid, alpha=0.5)
plt.axhline(y=0, color='red', linestyle='--')
plt.title('Residual Plot')
plt.xlabel('Fitted values')
plt.ylabel('Residuals')
plt.show()

# Print the summary of the regression results
print(select_model.summary())
```



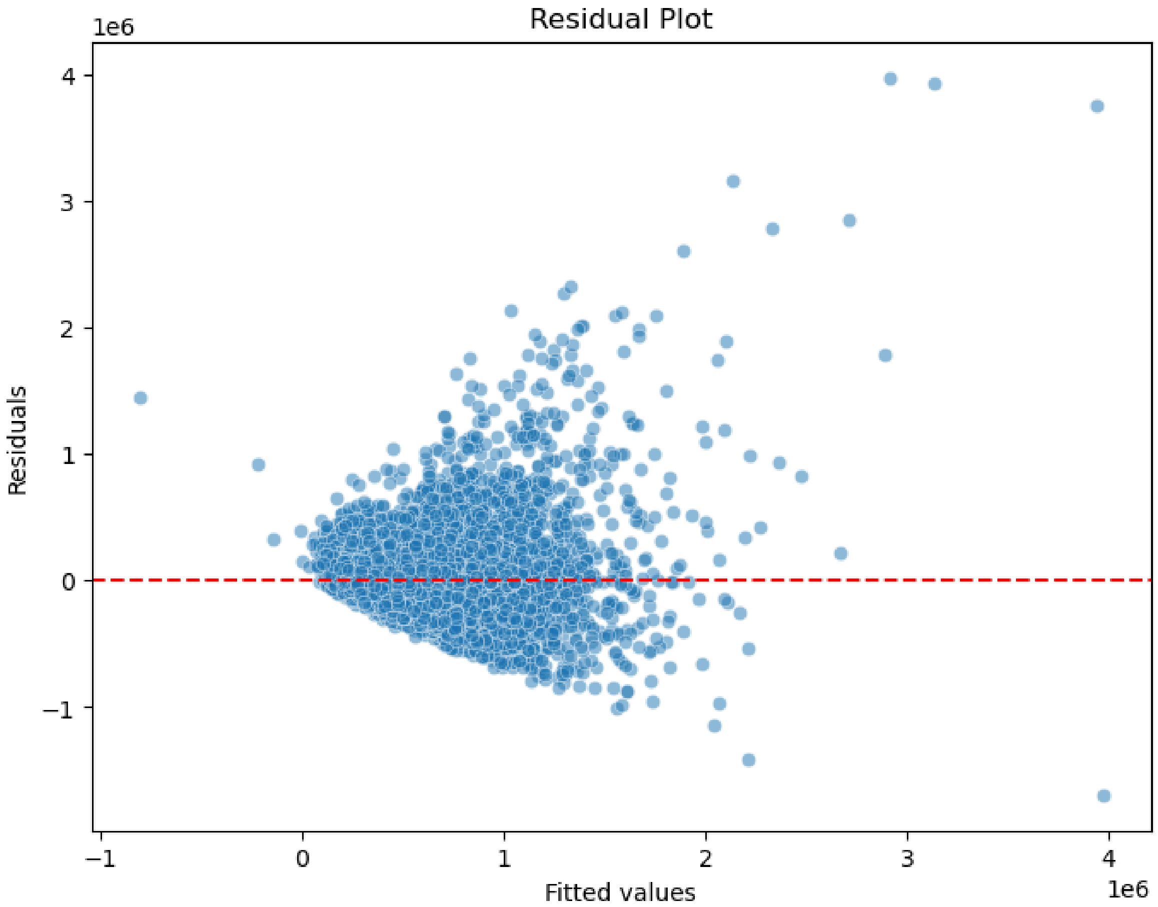
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.503
Model:                            OLS   Adj. R-squared:                  0.503
Method:                 Least Squares   F-statistic:                     4372.
Date:                Tue, 09 Apr 2024   Prob (F-statistic):               0.00
Time:                        18:37:17   Log-Likelihood:            -2.3988e+05
No. Observations:               17277   AIC:                         4.798e+05
Df Residuals:                   17272   BIC:                         4.798e+05
Df Model:                           4
Covariance Type:            nonrobust
=================================================================================
===
                       coef    std err          t      P>|t|      [0.025      0.9
75]
---------------------------------------------------------------------------------
---
const             7.783e+04   1.19e+04      6.552      0.000    5.45e+04    1.01e
+05
sqft_living        311.6452      3.183     97.914      0.000     305.407     317.
884
bedrooms         -5.631e+04   3127.702    -18.005      0.000   -6.24e+04   -5.02e
+04
bathroom_density  1515.7304   1.26e+04      0.121      0.904   -2.31e+04    2.61e
+04
floors            2069.2793   4247.491      0.487      0.626   -6256.234    1.04e
+04
==============================================================================
Omnibus:                    11850.922   Durbin-Watson:                   2.005
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           444443.297
Skew:                           2.813   Prob(JB):                         0.00
Kurtosis:                      27.202   Cond. No.                     1.86e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.86e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

iv. All Features Modelling

In [54]:

```
# Building a model with all the features

X_train_c = sm.add_constant(X_train)
final_multiple_model = sm.OLS(y_train, X_train_c)
final_multiple_model = final_multiple_model.fit()

# Print a summary of the regression results


# Scatter plot
plt.figure(figsize=(8, 6))
sns.scatterplot(x=final_multiple_model.fittedvalues, y=final_multiple_model.resi
plt.axhline(y=0, color='red', linestyle='--')
plt.title('Residual Plot')
plt.xlabel('Fitted values')
plt.ylabel('Residuals')
plt.show()
```

```
# Print the summary of the regression results
print(final_multiple_model.summary())
```



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.554
Model:                            OLS   Adj. R-squared:                  0.553
Method:                 Least Squares   F-statistic:                     3059.
Date:                Tue, 09 Apr 2024   Prob (F-statistic):               0.00
Time:                        18:37:17   Log-Likelihood:            -2.3896e+05
No. Observations:               17277   AIC:                         4.779e+05
Df Residuals:                   17269   BIC:                         4.780e+05
Df Model:                           7
Covariance Type:            nonrobust
=================================================================================
===
                       coef    std err          t      P>|t|      [0.025      0.9
75]
---------------------------------------------------------------------------------
---
const             6.027e+06   1.58e+05     38.135      0.000    5.72e+06    6.34e
+06
bedrooms         -4.021e+04   3017.400    -13.325      0.000   -4.61e+04   -3.43e
+04
sqft_living        314.9737      3.066    102.731      0.000     308.964     320.
983
sqft_lot            -0.3508      0.046     -7.683      0.000      -0.440      -0.
261
floors            6.064e+04   4287.097     14.144      0.000    5.22e+04     6.9e
+04
yr_built         -3173.2507     81.072    -39.141      0.000   -3332.159   -3014.
342
condition_coded    1.91e+04   3129.629      6.103      0.000      1.3e+04    2.52e
+04
bathroom_density  1.521e+05   1.25e+04     12.199      0.000    1.28e+05    1.77e
+05
==============================================================================
Omnibus:                    11767.200   Durbin-Watson:                   2.000
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           478168.608
Skew:                           2.754   Prob(JB):                         0.00
Kurtosis:                      28.177   Cond. No.                     3.76e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.76e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

The Final equation for our model

price=(-40,210×bedrooms)+(314.97×sqft_living)−(0.35×sqft_lot)+(60,640×floors)−(-3173.25×yr_built)+(19,100×condition_coded)+(152,100×bathroom_density)+ 6,027,000

Model Performance

- R-squared: 55.8% of the variance in the price is explained by the independent variables included in the model.
- F-statistic: We have a high F-statistic of 3059 and a low F-statistic probability which suggests that the overall model is statistically significant.

v. Further Adressing the multicollineary using VIFs

In [55]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant

# Add constant term to the independent variables
X_train_with_const = add_constant(X_train[['bedrooms', 'sqft_living', 'sqft_lot'

# Calculate VIFs for each independent variable
vif_data = pd.DataFrame()
vif_data["feature"] = X_train_with_const.columns
vif_data["VIF"] = [variance_inflation_factor(X_train_with_const.values, i) for i

# Print VIFs
print(vif_data)
```

```
           feature          VIF
0            const  7142.086783
1         bedrooms     2.262858
2      sqft_living     2.281996
3         sqft_lot     1.042720
4           floors     1.532751
5         yr_built     1.625214
6   condition_coded     1.184428
7  bathroom_density     2.005565
```

Conclusion on multicollinearity

The VIF values for all independent features are below the commonly used threshold of 5. This means that there is low multicollinearity among the features. We will conclude based on this that our model is stable and reliable for house price prediction given our features.

E. Model Evaluation

In [56]:
```
# Previewing the first 5 rows
X_test.head()
```

Out[56]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	yr_built	condition_coded
2398	3	1.00	950	4500	1.0	1943	4
14724	2	1.00	1190	6200	1.0	1948	3
20980	4	3.00	5520	8313	2.0	2008	3
12156	3	2.00	1980	12150	1.0	1994	3
19485	2	1.75	1870	6625	1.0	1948	3

In [57]:
```
## Aligning test data with training data
## Deleting the bathrooms column
# Adding the bathroom_density
X_test["bathroom_density"] = X_test["bathrooms"] / X_test["bedrooms"]
X_test.drop('bathrooms', axis=1, inplace=True)
X_test.head()
```

Out[57]:

	bedrooms	sqft_living	sqft_lot	floors	yr_built	condition_coded	bathroom_den
2398	3	950	4500	1.0	1943	4	0.333
14724	2	1190	6200	1.0	1948	3	0.500
20980	4	5520	8313	2.0	2008	3	0.750
12156	3	1980	12150	1.0	1994	3	0.666
19485	2	1870	6625	1.0	1948	3	0.875

In [58]:

```
# Additional imports
from sklearn.metrics import mean_squared_error, r2_score
```

In [59]:

```
# Make predictions using the model with selected features as used above on our s
X_test_select = sm.add_constant(X_test[['sqft_living', 'bedrooms', 'bathroom_den
y_pred_select = select_model.predict(X_test_select)

# Calculate the evaluation metrics for the model with selected features
# We will use MSE, RSME and R-squared to evaluate our model
mse_select_model = mean_squared_error(y_test, y_pred_select)
rmse_select_model = np.sqrt(mse_select_model)
r_squared_select_model = r2_score(y_test, y_pred_select)

# Print evaluation metrics for the model with selected features
print("Evaluation metrics for model with selected features:")
print("Mean Squared Error (MSE):", mse_select_model)
print("Root Mean Squared Error (RMSE):", rmse_select_model)
print("R-squared:", r_squared_select_model)
print()
```

```
Evaluation metrics for model with selected features:
Mean Squared Error (MSE): 63854708809.887505
Root Mean Squared Error (RMSE): 252694.89272616393
R-squared: 0.5219348196988434
```

Conclusion of the select features model

- The Mean Squared Error (MSE) indicates that, on average, the model's predictions are off by approximately USD 63,854,708,809.
- The Root Mean Squared Error (RMSE) suggests that, on average, the model's predictions are off by approximately USD 252,694.89.
- The R-squared value of ~0.5004 suggests that around 52.19% of the variance in the price is explained by the selected features.

In [60]:

```
# Make predictions using the model with all features
X_test_c = sm.add_constant(X_test)  # Add constant term for intercept
y_pred_all = final_multiple_model.predict(X_test_c)

# Calculate evaluation metrics for the model with all features
mse_final_model = mean_squared_error(y_test, y_pred_all)
rmse_final_model = np.sqrt(mse_final_model)
r_squared_final_model = r2_score(y_test, y_pred_all)
```

```
# Print evaluation metrics for the model with all features
print("Evaluation metrics for model with all features:")
print("Mean Squared Error (MSE):", mse_final_model)
print("Root Mean Squared Error (RMSE):", rmse_final_model)
print("R-squared:", r_squared_final_model)
```

```
Evaluation metrics for model with all features:
Mean Squared Error (MSE): 58831328055.65425
Root Mean Squared Error (RMSE): 242551.7018197445
R-squared: 0.5595436894400523
```

Conclusion of all features model

Evaluation

- Mean Squared Error (MSE): Approximately USD 58,831,328,055
- Root Mean Squared Error (RMSE): Approximately USD 242,551.70
- R-squared: Approximately 0.60

Comparison of the select feature model

- The R-squared value is higher, meaning 55.6% of the variance in the price is explained by all the features.

Conclusion

- The model using all features provides better predictive performance compared to the model with selected features.

4. Results

- The size of the living area has a significant positive effect on the home price. For every additional square foot of living space, the price tends to increase by approximately $282.20 on average.

- The number of bedrooms in a property also positively impacts its price. Each additional bedroom contributes to an average increase of about $121,700 in the property price.

- The number of bathrooms in a property is positively correlated with its price. On average, each additional bathroom adds approximately $254,400 to the property price.

- The size of the lot (in square feet) has a relatively minor impact on the property price. For every unit increase in the square footage of the lot, the price tends to increase by about $0.82 on average.

- The number of floors in a property is also a significant factor in determining its price. On average, each additional floor contributes to an increase of around $176,400 in the property price.

- The age of the property (year built) has a relatively minor impact on its price. For every additional year since the property was built, the price tends to increase by approximately $600.43 on average.
- The coded condition of the property has a moderate effect on its price with the value of the house increasing by $24,330 on average if the condition of the house improves from one rating to another (e.g 1 to 2)
- We proceeded and found out that the bathroom was highly correlated to other independent variables ie: bedrooms,sqft_living, floors, and yr_built where the correlation was above 0.75 which is a high positive correlation.
- We created a new column representing the ratio of bathrooms to the number of bedrooms
- Because of this high multicollinearity effect of the bathroom we dropped it
- The multiple regression model with select features shows an MSE of approximately 63 billion, an RMSE of around 25k, and an R-squared value of approximately 0.52.
- The model with all features yields an MSE of roughly 58.8 billion, an RMSE of about 242,551, and a higher R-squared value of around 0.6.
- This suggests that the model with all features performs slightly better, meaning that including additional features improves the model's predictive accuracy.

5. Recommendations to Finsco Limited

1. Finsco Limited can advise its client to focus on increasing the size of the living room: Renovations that increase the square footage of living space have a significant impact on home value. For every additional square foot of living space added, the estimated home value can increase by approximately USD 320 to USD 322.
2. Upgrading bedrooms can also increase clients' home value. Investing in bedroom renovations, such as improving the layout, fixtures, colours and furnishing, could increase the value of the property by around USD 48,290 to USD 65,110 per bedroom, based on our regression results.
3. Improve Overall Condition will increase the value of the property. Renovations focused on improving the condition of the home, such as repairing structural issues, outside colouring, updating fixtures, and mowing may result in an estimated value increase of approximately USD 20,530, according to our analysis.
4. Bathroom is a minimum requirement for a house, from our model and correlation test, we found out that it is highly correlated to most of the other independent variables, implying it impacts the value of homes a lot. Assumption, improving the quality of the bathroom will increase the value of the home.
5. Overall, making renovations in the entire house to improve the house's overall condition will be better than doing partial renovations in the bathrooms, and

bedrooms or increasing the living room space. Overall house improvements will have a better impact on the value of the property than just renovating part of the house

6. Next Steps

1. The Hepta team will gather feedback from the Finsco team after this first iteration and gather any additional requirements and feedback.
2. Market Research: Hepta group will conduct market research to identify trends and patterns in the real estate market such as renovations and, property features demands. This will help us make informed conclusions while we continue to improve on the model.
3. Data Enrichment: The Hepta group will also work with the Finsco team to ensure we have enough additional data on other features that might help us improve our model. This may include users' preferences, renovations past data and other features that might have property prices.
4. Model improvement: Our Data scientists will try other regression algorithms to compare and see if we can use a better model for this project.