Final Project Submission

Please fill out:

- Student name: Sammy Warah
- Student pace: Partime
- Scheduled project review date/time:
- Instructor name: Noah Kandie
- Blog post URL:

Business Problem:

The goal of this project is to provide valuable insights to a real estate agency, which aims to offer data-driven recommendations to homeowners regarding home renovations and their potential impact on the estimated value of their properties. By leveraging statistical modeling techniques, the agency seeks to empower homeowners with actionable advice on how different home renovation projects may influence the resale value of their homes.

Objective:

Utilize multiple linear regression modeling to analyze the King County House Sales dataset and identify key factors influencing home prices. By doing so, the real estate agency can offer personalized recommendations to homeowners regarding the types of renovations or improvements that could potentially increase the estimated value of their homes. Additionally, the agency aims to quantify the potential increase in home value associated with different renovation projects to provide homeowners with a clear understanding of the expected Return On Investment.

Data Understanding:

The dataset used in this project is the King County House Sales dataset, comprising various features related to house sales in a northwestern county. It includes information such as the price of houses, number of bedrooms and bathrooms, square footage of living space and lot, condition, grade, year built, waterfront status, and dates of sale. Additionally, the dataset contains geographic information such as latitude and longitude, as well as details on any renovations.

1. Importing Relevant libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()

import warnings
warnings.filterwarnings('ignore')

import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
```

2. Loading our data set

```
raw_data =  pd.read_csv('data/kc_house_data.csv')
print(raw_data.columns)
raw_data.head()
```

```
Index(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',
       'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode',
       'lat', 'long', 'sqft_living15', 'sqft_lot15'],
      dtype='object')
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_abov
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	NaN	NONE	...	7 Average	118
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	NO	NONE	...	7 Average	217
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	NO	NONE	...	6 Low Average	77
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	NO	NONE	...	7 Average	105
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	NO	NONE	...	8 Good	168

5 rows × 21 columns

```
raw_data.describe()
```

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	sqft_above	yr_built
count	2.159700e+04	2.159700e+04	21597.000000	21597.000000	21597.000000	2.159700e+04	21597.000000	21597.000000	21597.000000
mean	4.580474e+09	5.402966e+05	3.373200	2.115826	2080.321850	1.509941e+04	1.494096	1788.596842	1970.999676
std	2.876736e+09	3.673681e+05	0.926299	0.768984	918.106125	4.141264e+04	0.539683	827.759761	29.375234
min	1.000102e+06	7.800000e+04	1.000000	0.500000	370.000000	5.200000e+02	1.000000	370.000000	1900.000000
25%	2.123049e+09	3.220000e+05	3.000000	1.750000	1430.000000	5.040000e+03	1.000000	1190.000000	1951.000000
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	1560.000000	1975.000000
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068500e+04	2.000000	2210.000000	1997.000000
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	9410.000000	2015.000000

3. Data Preperation.

```
# Creating a copy of our dataset
data =  raw_data.copy()
```

```
#Checking the correlation of the features and that of the target variable
data.corr()['price']
```

```
id              -0.016772
price            1.000000
bedrooms         0.308787
bathrooms        0.525906
sqft_living      0.701917
sqft_lot         0.089876
floors           0.256804
sqft_above       0.605368
yr_built         0.053953
yr_renovated     0.129599
zipcode         -0.053402
lat              0.306692
long             0.022036
sqft_living15    0.585241
sqft_lot15       0.082845
Name: price, dtype: float64
```

3.1 Data Cleaning.

```
# Checking for null values
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
```

```
# Cleaning the waterfront column
# We replace the missing values with 'No' to indicate that the property does not have a waterfront

data['waterfront'] = data['waterfront'].fillna('NO')

#droping the id column
data.drop('id', axis = 1, inplace = True)

#droping columns with the view missing
data.dropna(subset=['view'], inplace = True, axis = 0)

# converting the data column to a datetime datatype
data['date'] =  pd.to_datetime(data['date'])
```

The mean of the year of renovation column indicates that some houses were yet to renovated or the infomation on when the house was renovated was not captured. Hence we create an additional column to represent whether the house was ever renovated or not and drop the yr_renovated column since 16961 entries are missing or 0

```
# Creating a column to represent if a house has been renovated or not
data['renovated'] = data['yr_renovated'].apply(lambda year: 'No' if year == 0 else 'Yes')
```

```
# Dropping unecessary columns for our data
cols_to_drop = ['date','view','yr_renovated','zipcode','lat','long']
data = data.drop(columns=cols_to_drop)
```

```
data.isnull().sum()
```

```
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
sqft_living15    0
sqft_lot15       0
renovated        0
dtype: int64
```

```
#Checking our data types
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21534 entries, 0 to 21596
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   price          21534 non-null  float64
 1   bedrooms       21534 non-null  int64
 2   bathrooms      21534 non-null  float64
 3   sqft_living    21534 non-null  int64
 4   sqft_lot       21534 non-null  int64
 5   floors         21534 non-null  float64
 6   waterfront     21534 non-null  object
```

```
 7   condition      21534 non-null  object
 8   grade          21534 non-null  object
 9   sqft_above     21534 non-null  int64
 10  sqft_basement  21534 non-null  object
 11  yr_built       21534 non-null  int64
 12  sqft_living15  21534 non-null  int64
 13  sqft_lot15     21534 non-null  int64
 14  renovated      21534 non-null  object
dtypes: float64(3), int64(7), object(5)
memory usage: 2.6+ MB
```

Some features, such as waterfront and condition, are categorical, while others, like price and square footage, are numerical. The dataset contains a total of 21 columns and 21,597 entries. Initial exploration reveals missing values in certain columns, which require preprocessing before conducting statistical modeling.

```
# Checking the sqft_basement column
data['sqft_basement']
```

```
0          0.0
1        400.0
2          0.0
3        910.0
4          0.0
         ...
21592      0.0
21593      0.0
21594      0.0
21595      0.0
21596      0.0
Name: sqft_basement, Length: 21534, dtype: object
```

```
# Convertion the values in the sqft_basement to a numeric data type
data['sqft_basement'] = pd.to_numeric(data['sqft_basement'], errors='coerce')
```

```
data['sqft_basement'].describe()
```

```
count    21082.000000
mean       291.359975
std        442.007858
min          0.000000
25%          0.000000
50%          0.000000
75%        560.000000
max       4820.000000
Name: sqft_basement, dtype: float64
```

In this case, given that the mean is 291.85 and the median (50th percentile) is 0, Therefore we impute the missing values with the median. This way we preserve the distribution's integrity.

```
# Calculate the median of the column
median_value = data['sqft_basement'].median()

# Replace missing values with the median
data['sqft_basement'].fillna(median_value, inplace=True)

data['sqft_basement'].dtype
```

```
dtype('float64')
```

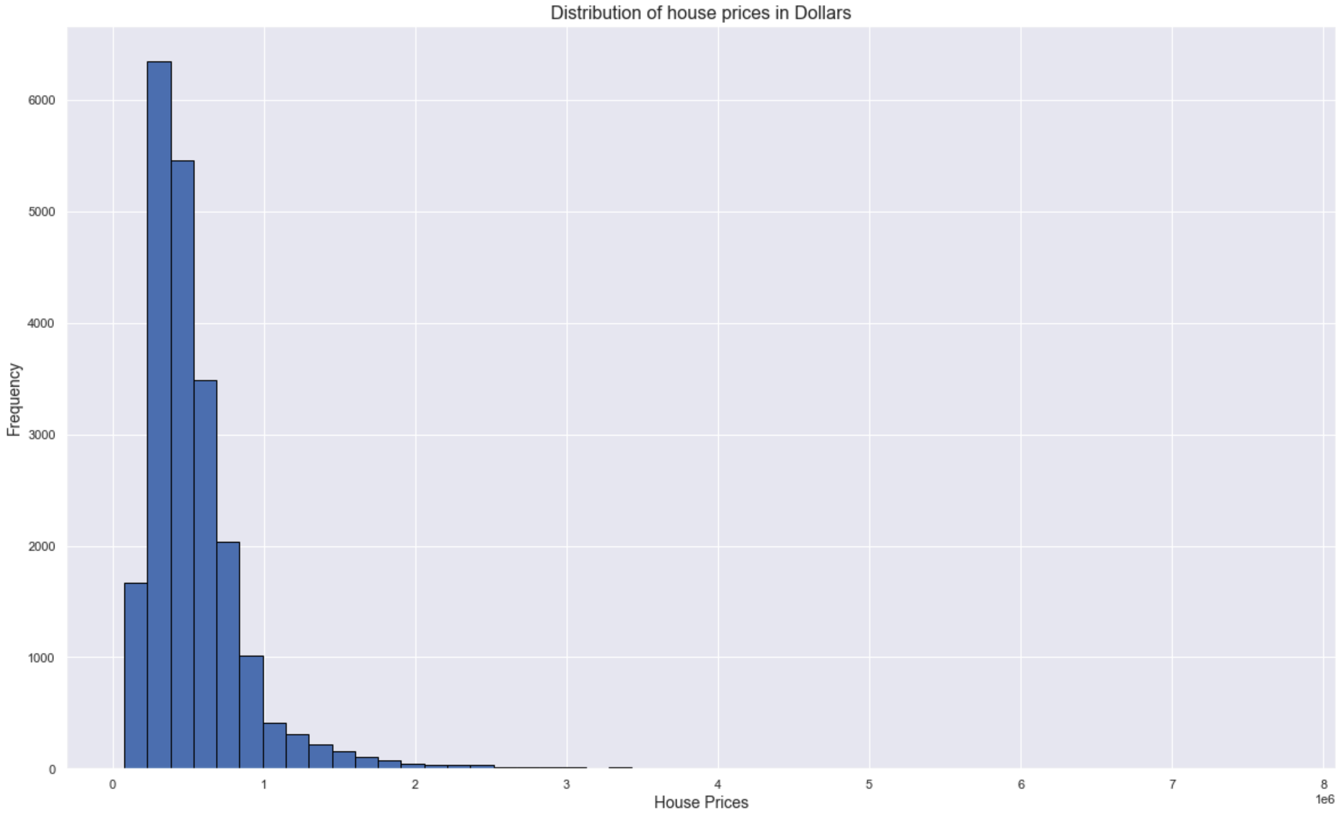
```
#Creating a copy of a data

cleaned_data =  data.copy()
```

4. Data Visualization

Visualizing the distribution of our target variable, Price

```
plt.figure(figsize=(20, 12))
plt.hist(data['price'], bins = 50, edgecolor =  'black')
plt.title('Distribution of house prices in Dollars', fontsize = 16)
plt.xlabel('House Prices', fontsize = 14)
plt.ylabel('Frequency', fontsize = 14)
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```




```
price_mean = data['price'].mean()
price_mode = data['price'].mode()
price_median = data['price'].median()

print(f'The Mean of Price is:{price_mean}')
print(f'The Median of Price is:{price_median}')
print(f'The Mode of Price is:{price_mode}')
```

```
The Mean of Price is:540057.663833937
The Median of Price is:450000.0
The Mode of Price is:0    350000.0
1    450000.0
dtype: float64
```

Based on our data and the Histogram above, House Prices are positively skewed, meaning that: Mode < Median < Mean

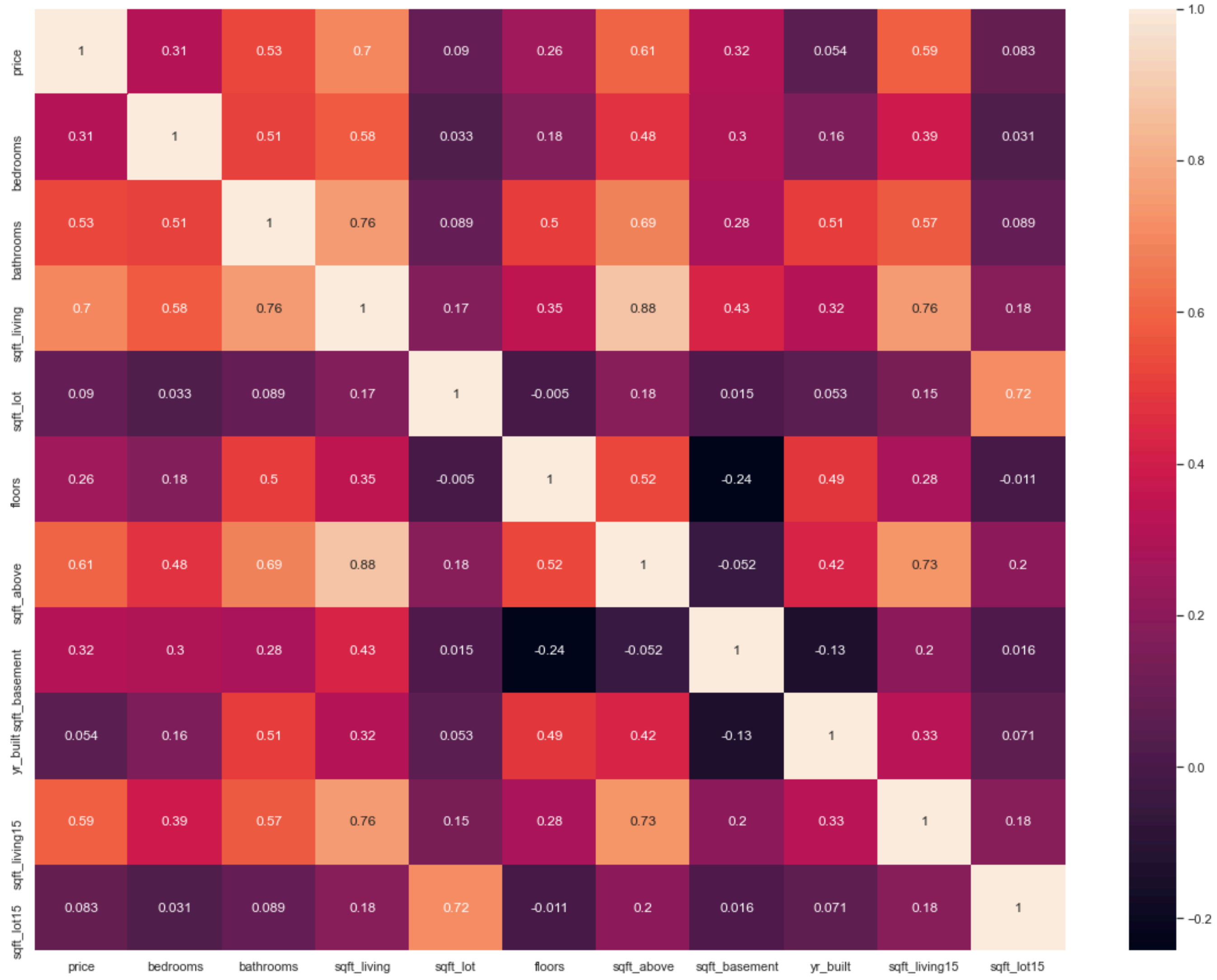
4.1 Correlation Matrix

```
#Checking the correlation of our numerial variables and target variable(price)
cor_price = data.corr()['price']
cor_price.sort_values(ascending = False)
```

```
price            1.000000
sqft_living      0.701587
sqft_above       0.605695
sqft_living15    0.585304
bathrooms        0.525053
sqft_basement    0.319082
bedrooms         0.308063
floors           0.257052
sqft_lot         0.090338
sqft_lot15       0.083189
yr_built         0.054273
Name: price, dtype: float64
```

```
plt.figure(figsize = (20,15))
sns.heatmap(cleaned_data.corr(), annot = True)
```

```
<AxesSubplot:>
```



Based on the above correlation matrix, it can be observed that the Square footage of living space other than the basement has the highest positive correlation to the price of 0.70. Which is a strong correlation. Additionally,the yr_build variable had the least correlation of 0.054

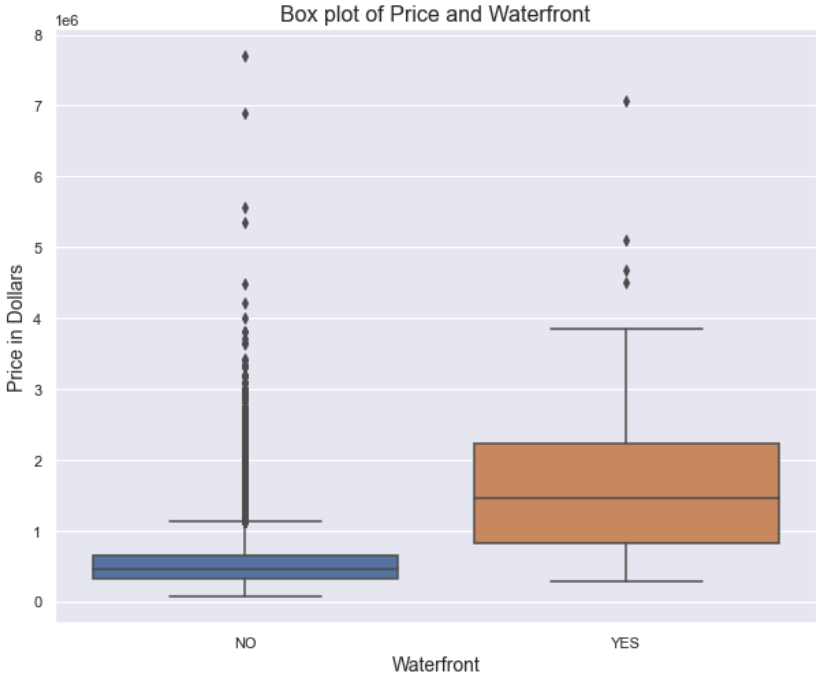
4.2 Visualizing the categorical Variables

1. Price vs Waterfront

```
data['waterfront'].unique()
```

```
array(['NO', 'YES'], dtype=object)
```

```
#Checking the distribution of price based on whether it has a waterfront or not

plt.figure(figsize=(10, 8))
sns.boxplot(x='waterfront', y='price', data=data)
plt.xlabel('Waterfront', fontsize = 14)
plt.ylabel('Price in Dollars', fontsize = 14)
plt.title('Box plot of Price and Waterfront', fontsize = 16)
plt.show()
```



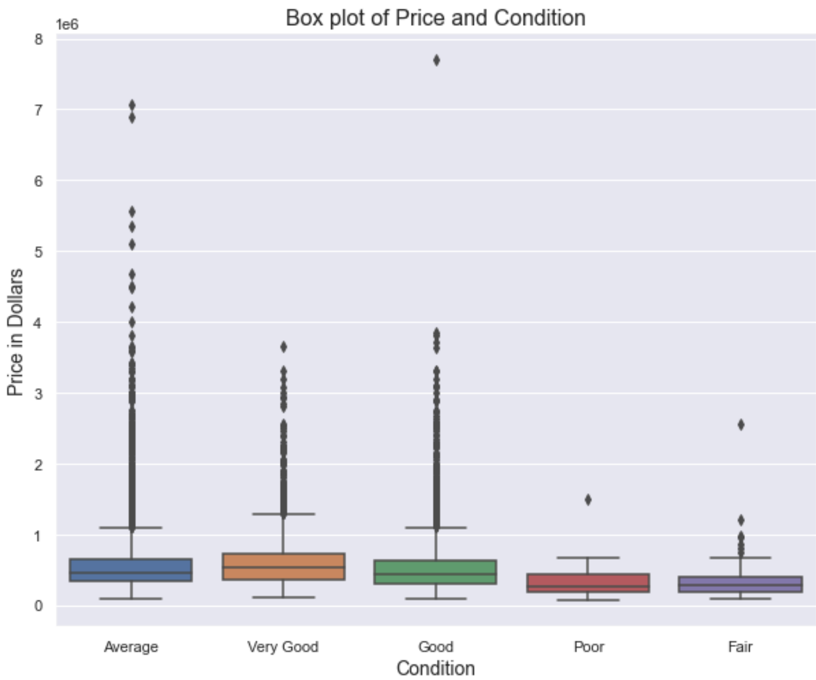
From the Box plot above, we can observe that the price of a house tends to be higher when the house has a waterfront. Additionally, houses that do not have a waterfront tend to have more outliers than those with waterfronts

2. Price Vs Condition

```
data['condition'].unique()
```

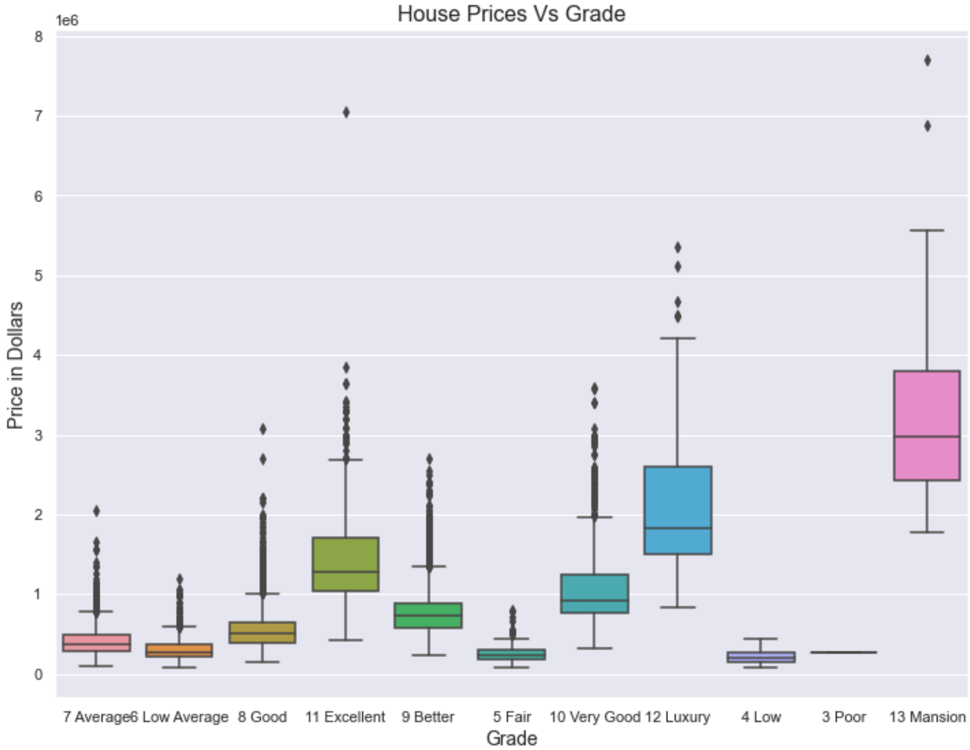
```
array(['Average', 'Very Good', 'Good', 'Poor', 'Fair'], dtype=object)
```

```
#Checking the distribution of price based on condition

plt.figure(figsize=(10, 8))
sns.boxplot(x='condition', y='price', data=data)
plt.xlabel('Condition', fontsize = 14)
plt.ylabel('Price in Dollars', fontsize = 14)
plt.title('Box plot of Price and Condition',fontsize = 16)
plt.show()
```



3. Price Vs Grade

```
#Checking the distribution of price and Grade
plt.figure(figsize=(12, 9))
sns.boxplot(x='grade', y='price', data=data)
plt.xlabel('Grade', fontsize = 14)
plt.ylabel('Price in Dollars', fontsize = 14)
plt.title('House Prices Vs Grade',fontsize = 16)
plt.show()
```

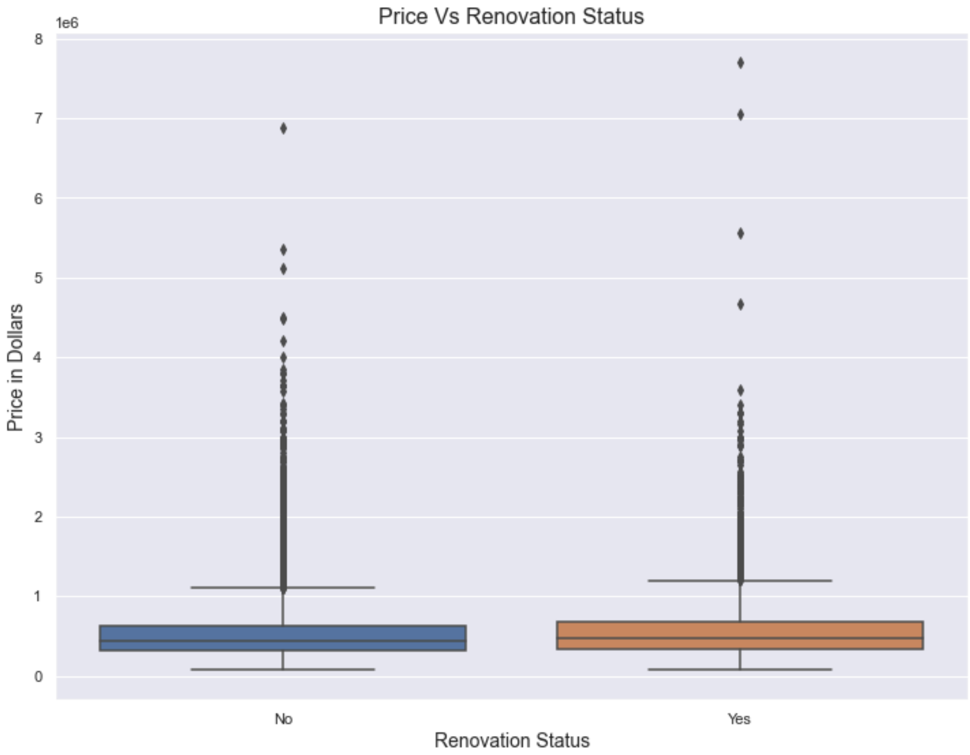



```
data['grade'].unique()
```

```
array(['7 Average', '6 Low Average', '8 Good', '11 Excellent', '9 Better',
       '5 Fair', '10 Very Good', '12 Luxury', '4 Low', '3 Poor',
       '13 Mansion'], dtype=object)
```

4. Price vs Renovated

```
#Checking the distribution of price and Renovation status
plt.figure(figsize=(12, 9))
sns.boxplot(x='renovated', y='price', data=data)
plt.xlabel('Renovation Status', fontsize = 14)
plt.ylabel('Price in Dollars', fontsize = 14)
plt.title('Price Vs Renovation Status', fontsize = 16)
plt.show()
```



Based on the renovation status, the price on houses that have been renovated and those that have not been renovated seem to be the same.

4.3 Visualizing Some of our numeric features

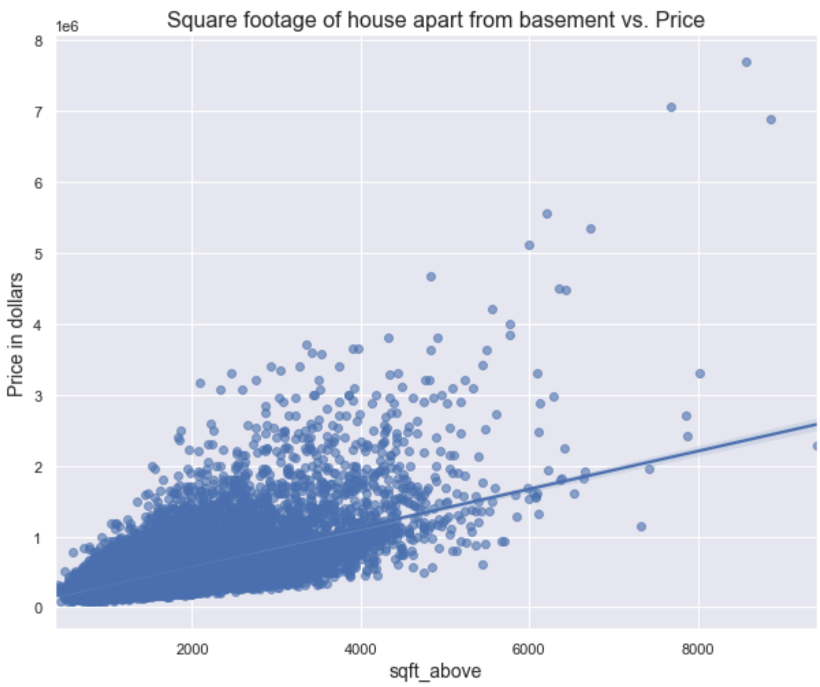
We visualize the relationship between price and some of the independent variables with the highest correlation

1. Price vs Square footage of living space in the home (sqft_living)

```
# Scatter plot of sqft_living vs. price
plt.figure(figsize=(10, 8))
sns.regplot(data=data, x='sqft_living', y='price', scatter_kws={'alpha':0.6})
plt.title('Square footage of living space in the home vs. Price', fontsize =16)
plt.xlabel('Square footage of living space in the home', fontsize = 14)
plt.ylabel('Price in Dollars',fontsize = 14)
plt.grid(True)
plt.tight_layout()
plt.show()
```

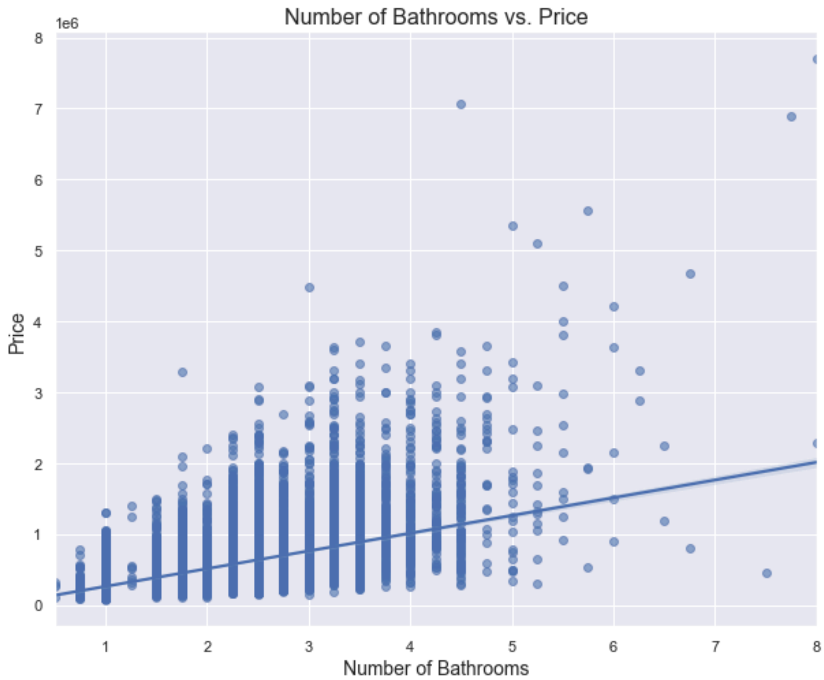
2. Price vs sqft_above

```
# Scatter plot of Square footage of house apart from basement(sqft_above) vs. price
plt.figure(figsize=(10, 8))
sns.regplot(data=data, x='sqft_above', y='price', scatter_kws={'alpha':0.6})
plt.title('Square footage of house apart from basement vs. Price', fontsize = 16)
plt.xlabel('sqft_above', fontsize = 14)
plt.ylabel('Price in dollars', fontsize = 14)
plt.grid(True)
plt.show()
```



3. Price vs Number of Bathrooms

```
# Scatter plot of Number of bathrooms vs. price
plt.figure(figsize=(10, 8))
sns.regplot(data=data, x='bathrooms', y='price', scatter_kws={'alpha':0.6})
plt.title('Number of Bathrooms vs. Price', fontsize = 16)
plt.xlabel('Number of Bathrooms', fontsize = 14)
plt.ylabel('Price', fontsize = 14)
plt.grid(True)
plt.show()
```



The three graphs above illustrate the presense of a positive relationship between Square footage of living space in the home, Number of Bathrooms, Square footage of house apart from basement and Price of the home.

---

5. Regression Modeling.

```
#Checking the correlation on our independent variable
cleaned_data.corr()['price'].sort_values(ascending =  False)
```

```
price            1.000000
sqft_living      0.701587
sqft_above       0.605695
sqft_living15    0.585304
bathrooms        0.525053
sqft_basement    0.319082
bedrooms         0.308063
floors           0.257052
sqft_lot         0.090338
sqft_lot15       0.083189
yr_built         0.054273
Name: price, dtype: float64
```

4.1 Simple Linear Model

```
# Creating our target and feature variables
y = cleaned_data['price']
X = cleaned_data['sqft_living']

# Create an OLS model
model = sm.OLS(endog = y, exog = sm.add_constant(X))
results = model.fit()


#Checking the model results
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
```

```
Dep. Variable:                  price   R-squared:                       0.492
Model:                            OLS   Adj. R-squared:                  0.492
Method:                 Least Squares   F-statistic:                 2.087e+04
Date:                Tue, 09 Apr 2024   Prob (F-statistic):               0.00
Time:                        20:27:52   Log-Likelihood:            -2.9912e+05
No. Observations:               21534   AIC:                         5.982e+05
Df Residuals:                   21532   BIC:                         5.983e+05
Df Model:                           1
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
const       -4.215e+04   4404.521     -9.570      0.000   -5.08e+04   -3.35e+04
sqft_living   279.9321      1.938    144.473      0.000     276.134     283.730
==============================================================================
Omnibus:                    14582.265   Durbin-Watson:                   1.981
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           516142.289
Skew:                           2.781   Prob(JB):                         0.00
Kurtosis:                      26.331   Cond. No.                     5.63e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.63e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Observations.

1. Our Model and coefficient are statistically significant, because the F-value is less than our assumed alpha of 0.05.
2. Our Adjusted R Squared is 0.493, hence the model explains 49.2% of the variance in price, target variable.
3. For a square-feet living of 0, our model would predict a price of -0.0004399 dollars. An increase of 1 square-feet living, would increase the price by 280.93.
4. The condition number is 5630. Since our condition number is above 100, there is evidence of a multicollinearity issue hence further modeling using additional feature variables

4.2 Multiple Regression Model

```
cleaned_data.columns
```

```
Index(['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors',
       'waterfront', 'condition', 'grade', 'sqft_above', 'sqft_basement',
       'yr_built', 'sqft_living15', 'sqft_lot15', 'renovated'],
      dtype='object')
```

```
cleaned_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21534 entries, 0 to 21596
Data columns (total 15 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   price          21534 non-null  float64
 1   bedrooms       21534 non-null  int64
 2   bathrooms      21534 non-null  float64
 3   sqft_living    21534 non-null  int64
 4   sqft_lot       21534 non-null  int64
 5   floors         21534 non-null  float64
 6   waterfront     21534 non-null  object
 7   condition      21534 non-null  object
 8   grade          21534 non-null  object
 9   sqft_above     21534 non-null  int64
 10  sqft_basement  21534 non-null  float64
 11  yr_built       21534 non-null  int64
 12  sqft_living15  21534 non-null  int64
 13  sqft_lot15     21534 non-null  int64
 14  renovated      21534 non-null  object
dtypes: float64(4), int64(7), object(4)
memory usage: 3.3+ MB
```

```
#Creating dummies for our categorical variables: condition, waterfront,renovated variables
# List of columns to create dummy variables for
columns_create_dummies = ['waterfront', 'condition', 'grade', 'renovated']

# Iterate over each column and create dummy variables
for column in columns_create_dummies:
    cleaned_data = pd.get_dummies(cleaned_data, columns=[column], drop_first=True)

# Display the DataFrame with dummy variables
print(cleaned_data)
```

```
          price  bedrooms  bathrooms  sqft_living  sqft_lot  floors  \
0      221900.0         3       1.00         1180      5650     1.0
1      538000.0         3       2.25         2570      7242     2.0
2      180000.0         2       1.00          770     10000     1.0
3      604000.0         4       3.00         1960      5000     1.0
4      510000.0         3       2.00         1680      8080     1.0
...         ...       ...        ...          ...       ...     ...
21592  360000.0         3       2.50         1530      1131     3.0
21593  400000.0         4       2.50         2310      5813     2.0
21594  402101.0         2       0.75         1020      1350     2.0
21595  400000.0         3       2.50         1600      2388     2.0
21596  325000.0         2       0.75         1020      1076     2.0

       sqft_above  sqft_basement  yr_built  sqft_living15  ...  \
0            1180            0.0      1955           1340  ...
1            2170          400.0      1951           1690  ...
2             770            0.0      1933           2720  ...
3            1050          910.0      1965           1360  ...
4            1680            0.0      1987           1800  ...
...           ...            ...       ...            ...  ...
21592        1530            0.0      2009           1530  ...
21593        2310            0.0      2014           1830  ...
21594        1020            0.0      2009           1020  ...
21595        1600            0.0      2004           1410  ...
21596        1020            0.0      2008           1020  ...

       grade_12 Luxury  grade_13 Mansion  grade_3 Poor  grade_4 Low  \
0                    0                 0             0            0
1                    0                 0             0            0
2                    0                 0             0            0
3                    0                 0             0            0
4                    0                 0             0            0
...                ...               ...           ...          ...
21592                0                 0             0            0
21593                0                 0             0            0
21594                0                 0             0            0
21595                0                 0             0            0
21596                0                 0             0            0

       grade_5 Fair  grade_6 Low Average  grade_7 Average  grade_8 Good  \
0                 0                    0                1             0
1                 0                    0                1             0
2                 0                    1                0             0
3                 0                    0                1             0
4                 0                    0                0             1
...             ...                  ...              ...           ...
21592             0                    0                0             1
21593             0                    0                0             1
21594             0                    0                1             0
21595             0                    0                0             1
21596             0                    0                1             0

       grade_9 Better  renovated_Yes
0                   0              0
1                   0              1
2                   0              1
3                   0              0
4                   0              0
```

```
cleaned_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21534 entries, 0 to 21596
Data columns (total 27 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   price               21534 non-null  float64
 1   bedrooms            21534 non-null  int64
 2   bathrooms           21534 non-null  float64
 3   sqft_living         21534 non-null  int64
 4   sqft_lot            21534 non-null  int64
 5   floors              21534 non-null  float64
 6   sqft_above          21534 non-null  int64
 7   sqft_basement       21534 non-null  float64
 8   yr_built            21534 non-null  int64
```

```
 9   sqft_living15        21534 non-null  int64
 10  sqft_lot15           21534 non-null  int64
 11  waterfront_YES       21534 non-null  uint8
 12  condition_Fair       21534 non-null  uint8
 13  condition_Good       21534 non-null  uint8
 14  condition_Poor       21534 non-null  uint8
 15  condition_Very Good  21534 non-null  uint8
 16  grade_11 Excellent   21534 non-null  uint8
 17  grade_12 Luxury      21534 non-null  uint8
 18  grade_13 Mansion     21534 non-null  uint8
 19  grade_3 Poor         21534 non-null  uint8
 20  grade_4 Low          21534 non-null  uint8
 21  grade_5 Fair         21534 non-null  uint8
 22  grade_6 Low Average  21534 non-null  uint8
 23  grade_7 Average      21534 non-null  uint8
 24  grade_8 Good         21534 non-null  uint8
 25  grade_9 Better       21534 non-null  uint8
 26  renovated_Yes        21534 non-null  uint8
dtypes: float64(4), int64(7), uint8(16)
memory usage: 2.9 MB
```

```
#Assigning the target variable and independent variables
y = cleaned_data['price']
X = cleaned_data.drop(['price'], axis = 1)
```

```
# Create an OLS model
model = sm.OLS(endog = y, exog = sm.add_constant(X))
results = model.fit()

#Printing a summary of our results
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.676
Model:                            OLS   Adj. R-squared:                  0.675
Method:                 Least Squares   F-statistic:                     1723.
Date:                Tue, 09 Apr 2024   Prob (F-statistic):               0.00
Time:                        20:27:52   Log-Likelihood:            -2.9430e+05
No. Observations:               21534   AIC:                         5.886e+05
Df Residuals:                   21507   BIC:                         5.889e+05
Df Model:                          26
Covariance Type:            nonrobust
=======================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------
const                7.136e+06   1.31e+05     54.630      0.000    6.88e+06    7.39e+06
bedrooms              -2.8e+04   2004.519    -13.968      0.000   -3.19e+04   -2.41e+04
bathrooms            5.019e+04   3387.154     14.818      0.000    4.36e+04    5.68e+04
sqft_living           118.3796     18.719      6.324      0.000      81.689     155.070
sqft_lot                0.0255      0.050      0.514      0.607      -0.072       0.123
floors               4.787e+04   3726.039     12.847      0.000    4.06e+04    5.52e+04
sqft_above             -4.7131     18.711     -0.252      0.801     -41.387      31.961
sqft_basement          44.0528     18.565      2.373      0.018       7.664      80.442
yr_built            -3393.7654     66.643    -50.924      0.000   -3524.391   -3263.140
sqft_living15          40.3764      3.491     11.565      0.000      33.533      47.220
sqft_lot15             -0.5482      0.076     -7.235      0.000      -0.697      -0.400
waterfront_YES       7.098e+05   1.76e+04     40.274      0.000    6.75e+05    7.44e+05
condition_Fair       -3.08e+04   1.63e+04     -1.890      0.059   -6.27e+04    1133.965
condition_Good       1.707e+04   3542.641      4.818      0.000    1.01e+04      2.4e+04
condition_Poor      -4.687e+04   3.91e+04     -1.198      0.231   -1.24e+05    2.98e+04
condition_Very Good  5.667e+04   5717.652      9.911      0.000    4.55e+04    6.79e+04
grade_11 Excellent   2.749e+05   1.24e+04     22.118      0.000    2.51e+05    2.99e+05
grade_12 Luxury      7.385e+05   2.38e+04     31.091      0.000    6.92e+05    7.85e+05
grade_13 Mansion     1.994e+06   5.93e+04     33.626      0.000    1.88e+06    2.11e+06
grade_3 Poor        -5.794e+05   2.09e+05     -2.773      0.006   -9.89e+05    -1.7e+05
grade_4 Low         -5.393e+05   4.15e+04    -12.997      0.000   -6.21e+05   -4.58e+05
grade_5 Fair        -5.464e+05   1.67e+04    -32.715      0.000   -5.79e+05   -5.14e+05
grade_6 Low Average -4.917e+05   1.06e+04    -46.564      0.000   -5.12e+05   -4.71e+05
grade_7 Average     -4.148e+05   8743.371    -47.442      0.000   -4.32e+05   -3.98e+05
grade_8 Good        -3.275e+05   7894.030    -41.493      0.000   -3.43e+05   -3.12e+05
grade_9 Better      -1.823e+05   7684.098    -23.730      0.000   -1.97e+05   -1.67e+05
renovated_Yes         1.18e+04   3516.767      3.357      0.001    4910.928    1.87e+04
==============================================================================
Omnibus:                    12587.272   Durbin-Watson:                   1.976
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           447468.241
Skew:                           2.226   Prob(JB):                         0.00
Kurtosis:                      24.884   Cond. No.                     7.43e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.43e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
coefficients_table = results.params.sort_values(ascending = True)
print(coefficients_table)
```

```
grade_3 Poor          -5.794356e+05
grade_5 Fair          -5.463648e+05
grade_4 Low           -5.393215e+05
grade_6 Low Average   -4.917013e+05
grade_7 Average       -4.148012e+05
grade_8 Good          -3.275485e+05
grade_9 Better        -1.823462e+05
condition_Poor        -4.686529e+04
condition_Fair        -3.079689e+04
bedrooms              -2.800013e+04
yr_built              -3.393765e+03
sqft_above            -4.713118e+00
sqft_lot15            -5.482174e-01
sqft_lot               2.546442e-02
sqft_living15          4.037643e+01
sqft_basement          4.405275e+01
sqft_living            1.183796e+02
renovated_Yes          1.180405e+04
condition_Good         1.706961e+04
floors                 4.786963e+04
bathrooms              5.019213e+04
condition_Very Good    5.666595e+04
grade_11 Excellent     2.748605e+05
waterfront_YES         7.098442e+05
grade_12 Luxury        7.385234e+05
grade_13 Mansion       1.993857e+06
const                  7.136450e+06
dtype: float64
```

Observations.

1. Our Model and coefficient are statistically significant, because the F-value is less than our assumed alpha of 0.05.
2. Our Adjusted R Squared is 0.675, hence the model eplains 67.5% of the variance in price, target variable.
3. Our intercept is 7136000, meaning that the price of a house will start from 7136000 dollars
4. The condition number is 7430000. Since our condition number is above 100, there is evidence of a multicollinearity issue hence further modeling using additional feature variables

From the above results, calculated t-value for sqft_lot and sqft_above is above the assumed alpha of 0.05, therefore we drop the variables and assess if our model performs better

```
#Assigning the target variable and independent variables, droping the sqft_lot and sqft_above columns to observe if our model is bet
y = cleaned_data['price']
X = cleaned_data.drop(['price','sqft_lot','sqft_above'], axis = 1)

# Create an OLS model
model = sm.OLS(endog = y, exog = sm.add_constant(X))
results = model.fit()

#Printing a summary of our results
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.676
Model:                            OLS   Adj. R-squared:                  0.675
Method:                 Least Squares   F-statistic:                     1867.
Date:                Tue, 09 Apr 2024   Prob (F-statistic):               0.00
Time:                        20:27:52   Log-Likelihood:            -2.9430e+05
No. Observations:               21534   AIC:                         5.886e+05
Df Residuals:                   21509   BIC:                         5.888e+05
Df Model:                          24
Covariance Type:            nonrobust
=======================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------
const                7.138e+06   1.31e+05     54.692      0.000    6.88e+06    7.39e+06
bedrooms            -2.803e+04   2003.856    -13.987      0.000    -3.2e+04   -2.41e+04
bathrooms            5.024e+04   3383.670     14.848      0.000    4.36e+04    5.69e+04
sqft_living           113.9033      3.916     29.089      0.000     106.228     121.578
floors               4.771e+04   3704.509     12.879      0.000    4.04e+04     5.5e+04
sqft_basement          48.5041      4.359     11.126      0.000      39.959      57.049
yr_built            -3394.7148     66.588    -50.981      0.000   -3525.233   -3264.196
sqft_living15          40.2369      3.481     11.559      0.000      33.414      47.060
sqft_lot15             -0.5212      0.054     -9.623      0.000      -0.627      -0.415
waterfront_YES       7.098e+05   1.76e+04     40.273      0.000    6.75e+05    7.44e+05
condition_Fair      -3.051e+04   1.63e+04     -1.874      0.061   -6.24e+04    1405.416
```

```
condition_Good       1.709e+04   3541.776      4.826      0.000    1.01e+04     2.4e+04
condition_Poor      -4.659e+04    3.91e+04    -1.191      0.234   -1.23e+05    3.01e+04
condition_Very Good  5.669e+04   5715.896      9.918      0.000    4.55e+04    6.79e+04
grade_11 Excellent   2.749e+05    1.24e+04    22.120      0.000     2.5e+05    2.99e+05
grade_12 Luxury      7.384e+05    2.37e+04    31.099      0.000    6.92e+05    7.85e+05
grade_13 Mansion     1.993e+06    5.93e+04    33.626      0.000    1.88e+06    2.11e+06
grade_3 Poor        -5.795e+05    2.09e+05    -2.773      0.006   -9.89e+05    -1.7e+05
grade_4 Low         -5.393e+05    4.15e+04   -12.998      0.000   -6.21e+05   -4.58e+05
grade_5 Fair        -5.463e+05    1.67e+04   -32.712      0.000   -5.79e+05   -5.14e+05
grade_6 Low Average -4.917e+05    1.06e+04   -46.574      0.000   -5.12e+05   -4.71e+05
grade_7 Average     -4.148e+05   8740.474    -47.460      0.000   -4.32e+05   -3.98e+05
grade_8 Good        -3.275e+05   7890.605    -41.510      0.000   -3.43e+05   -3.12e+05
grade_9 Better      -1.824e+05   7682.676    -23.738      0.000   -1.97e+05   -1.67e+05
renovated_Yes         1.18e+04   3516.495      3.355      0.001    4905.392    1.87e+04
==============================================================================
Omnibus:                    12581.221   Durbin-Watson:                   1.976
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           446759.374
Skew:                           2.224   Prob(JB):                         0.00
Kurtosis:                      24.866   Cond. No.                     4.43e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.43e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

From the above OLS model, The adjusted R remains the same at 0.675, but the condition number has reduced to 4.43e+06

4.3 Normalization of our model

```
from sklearn.preprocessing import StandardScaler


# Initialize the StandardScaler
scaler = StandardScaler()

# Fit and transform the feature matrix X
X_scaled = scaler.fit_transform(X)

# Fit and transform the target vector y
y_scaled = scaler.fit_transform(y.values.reshape(-1, 1)).flatten()

# Convert scaled feature matrix back to DataFrame if needed
X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns)

# Convert scaled target vector back to Series if needed
y_scaled_series = pd.Series(y_scaled, name=y.name)

# Now you can create your OLS model using the scaled features and target
model = sm.OLS(endog=y_scaled_series, exog=sm.add_constant(X_scaled_df))

#fiting our model
results = model.fit()

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

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.676
Model:                            OLS   Adj. R-squared:                  0.675
Method:                 Least Squares   F-statistic:                     1867.
Date:                Tue, 09 Apr 2024   Prob (F-statistic):               0.00
Time:                        20:27:52   Log-Likelihood:                -18434.
No. Observations:               21534   AIC:                         3.692e+04
Df Residuals:                   21509   BIC:                         3.712e+04
Df Model:                          24
Covariance Type:            nonrobust
=======================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------
const                1.267e-18      0.004   3.26e-16      1.000      -0.008       0.008
bedrooms               -0.0709      0.005    -13.987      0.000      -0.081      -0.061
bathrooms               0.1055      0.007     14.848      0.000       0.092       0.119
sqft_living             0.2855      0.010     29.089      0.000       0.266       0.305
floors                  0.0704      0.005     12.879      0.000       0.060       0.081
sqft_basement           0.0582      0.005     11.126      0.000       0.048       0.068
yr_built               -0.2724      0.005    -50.981      0.000      -0.283      -0.262
sqft_living15           0.0753      0.007     11.559      0.000       0.063       0.088
sqft_lot15             -0.0388      0.004     -9.623      0.000      -0.047      -0.031
```

```
waterfront_YES          0.1586      0.004     40.273      0.000       0.151       0.166
condition_Fair         -0.0074      0.004     -1.874      0.061      -0.015       0.000
condition_Good          0.0205      0.004      4.826      0.000       0.012       0.029
condition_Poor         -0.0047      0.004     -1.191      0.234      -0.012       0.003
condition_Very Good     0.0417      0.004      9.918      0.000       0.033       0.050
grade_11 Excellent      0.1010      0.005     22.120      0.000       0.092       0.110
grade_12 Luxury         0.1287      0.004     31.099      0.000       0.121       0.137
grade_13 Mansion        0.1337      0.004     33.626      0.000       0.126       0.142
grade_3 Poor           -0.0108      0.004     -2.773      0.006      -0.018      -0.003
grade_4 Low            -0.0521      0.004    -12.998      0.000      -0.060      -0.044
grade_5 Fair           -0.1573      0.005    -32.712      0.000      -0.167      -0.148
grade_6 Low Average    -0.3926      0.008    -46.574      0.000      -0.409      -0.376
grade_7 Average        -0.5585      0.012    -47.460      0.000      -0.582      -0.535
grade_8 Good           -0.4022      0.010    -41.510      0.000      -0.421      -0.383
grade_9 Better         -0.1624      0.007    -23.738      0.000      -0.176      -0.149
renovated_Yes           0.0132      0.004      3.355      0.001       0.005       0.021
==============================================================================
Omnibus:                    12581.221   Durbin-Watson:                   1.976
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           446759.374
Skew:                           2.224   Prob(JB):                         0.00
Kurtosis:                      24.866   Cond. No.                         9.56
==============================================================================
```