Phase 2 Project Notebook

In [1]:
```
# Project Introduction

# For this project, we were told a real estate agency was looking to discover which features or aspects of houses act as
# significant predictors of a house's selling price.  This real estate agency looks to seek out this information in order
# to gain a better understanding of which houses to purchase at certain prices in addition to what price certain owned houses
# should be sold at to maximize profits and minimize expenses.

# The real estate agency provided a data set containing house listings with various features of each listing
# which we then proceeded to analyze using descriptive analysis to provide various business insights.
# After obtaining such said business insights, using the data set a multiple linear regression model was created
# using the data set provided to identify features that would assist in predicting a house listings selling price.

# After confirming the model's accuracy through an iterative approach in its creation, several house listing features were
# indicated to be important to take into consideration when selling property and purchasing property.
```

In [2]:
```
# Import initial libraries to use for EDA.

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

In [3]:
```
# Import data set to be used.

house_df = pd.read_csv('C://Users//rychu//Desktop//2021//P2-Project//kc_house_data.csv')
```

Proceed with Initial Data Cleaning + EDA

In [5]:
```
house_df.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  float64
 9   view           21534 non-null  float64
 10  condition      21597 non-null  int64  
 11  grade          21597 non-null  int64  
 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(8), int64(11), object(2)
memory usage: 3.5+ MB
```

In [6]:
```
house_df.head(10)
```

Out[6]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_built
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	NaN	0.0	...	7	1180	0.0	1955
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	0.0	0.0	...	7	2170	400.0	1951
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	0.0	0.0	...	6	770	0.0	1933
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	0.0	0.0	...	7	1050	910.0	1965
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	0.0	0.0	...	8	1680	0.0	1987
5	7237550310	5/12/2014	1230000.0	4	4.50	5420	101930	1.0	0.0	0.0	...	11	3890	1530.0	2001
6	1321400060	6/27/2014	257500.0	3	2.25	1715	6819	2.0	0.0	0.0	...	7	1715	?	1995

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_built
7	2008000270	1/15/2015	291850.0	3	1.50	1060	9711	1.0	0.0	NaN	...	7	1060	0.0	1963
8	2414600126	4/15/2015	229500.0	3	1.00	1780	7470	1.0	0.0	0.0	...	7	1050	730.0	1960
9	3793500160	3/12/2015	323000.0	3	2.50	1890	6560	2.0	0.0	0.0	...	7	1890	0.0	2003

10 rows × 21 columns

In [7]:
```
house_df.describe()
```

Out[7]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	
count	2.159700e+04	2.159700e+04	21597.000000	21597.000000	21597.000000	2.159700e+04	21597.000000	19221.000000	21534.000000	21597.000000	21597.
mean	4.580474e+09	5.402966e+05	3.373200	2.115826	2080.321850	1.509941e+04	1.494096	0.007596	0.233863	3.409825	7.
std	2.876736e+09	3.673681e+05	0.926299	0.768984	918.106125	4.141264e+04	0.539683	0.086825	0.765686	0.650546	1.
min	1.000102e+06	7.800000e+04	1.000000	0.500000	370.000000	5.200000e+02	1.000000	0.000000	0.000000	1.000000	3.
25%	2.123049e+09	3.220000e+05	3.000000	1.750000	1430.000000	5.040000e+03	1.000000	0.000000	0.000000	3.000000	7.
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000	3.000000	7.
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068500e+04	2.000000	0.000000	0.000000	4.000000	8.
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000	5.000000	13.

In [8]:
```
house_df.isna().any()
```

Out[8]:
```
id               False
date             False
price            False
bedrooms         False
bathrooms        False
sqft_living      False
sqft_lot         False
floors           False
waterfront        True
view              True
condition        False
grade            False
sqft_above       False
sqft_basement    False
yr_built         False
yr_renovated      True
zipcode          False
lat              False
long             False
sqft_living15    False
sqft_lot15       False
dtype: bool
```

In [9]:
```
house_df.isna().sum()
```

Out[9]:
```
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 [10]:
```
# Filtering DF based off of missing values
```

In [11]:
```

```

```
house_df = house_df[house_df['waterfront'] >= 0]
house_df = house_df[house_df['view'] >= 0]
house_df = house_df[house_df['yr_renovated'] >= 0]
```

In [12]:
```
house_df.isna().any()
```

Out[12]:
```
id               False
date             False
price            False
bedrooms         False
bathrooms        False
sqft_living      False
sqft_lot         False
floors           False
waterfront       False
view             False
condition        False
grade            False
sqft_above       False
sqft_basement    False
yr_built         False
yr_renovated     False
zipcode          False
lat              False
long             False
sqft_living15    False
sqft_lot15       False
dtype: bool
```

In [13]:
```
house_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15762 entries, 1 to 21596
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             15762 non-null  int64
 1   date           15762 non-null  object
 2   price          15762 non-null  float64
 3   bedrooms       15762 non-null  int64
 4   bathrooms      15762 non-null  float64
 5   sqft_living    15762 non-null  int64
 6   sqft_lot       15762 non-null  int64
 7   floors         15762 non-null  float64
 8   waterfront     15762 non-null  float64
 9   view           15762 non-null  float64
 10  condition      15762 non-null  int64
 11  grade          15762 non-null  int64
 12  sqft_above     15762 non-null  int64
 13  sqft_basement  15762 non-null  object
 14  yr_built       15762 non-null  int64
 15  yr_renovated   15762 non-null  float64
 16  zipcode        15762 non-null  int64
 17  lat            15762 non-null  float64
 18  long           15762 non-null  float64
 19  sqft_living15  15762 non-null  int64
 20  sqft_lot15     15762 non-null  int64
dtypes: float64(8), int64(11), object(2)
memory usage: 2.6+ MB
```

In [14]:
```
house_df.describe()
```

Out[14]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	
count	1.576200e+04	1.576200e+04	15762.000000	15762.000000	15762.000000	1.576200e+04	15762.000000	15762.000000	15762.000000	15762.000000	15762.
mean	4.593364e+09	5.413172e+05	3.378949	2.120797	2084.512372	1.528082e+04	1.495147	0.007613	0.229984	3.410862	7.
std	2.876078e+09	3.722258e+05	0.935301	0.766772	918.617686	4.182288e+04	0.539352	0.086924	0.761324	0.651961	1.
min	1.000102e+06	8.200000e+04	1.000000	0.500000	370.000000	5.200000e+02	1.000000	0.000000	0.000000	1.000000	3.
25%	2.125159e+09	3.210000e+05	3.000000	1.750000	1430.000000	5.048500e+03	1.000000	0.000000	0.000000	3.000000	7.
50%	3.905081e+09	4.500000e+05	3.000000	2.250000	1920.000000	7.602000e+03	1.500000	0.000000	0.000000	3.000000	7.
75%	7.334501e+09	6.448750e+05	4.000000	2.500000	2550.000000	1.072000e+04	2.000000	0.000000	0.000000	4.000000	8.
max	9.895000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000	5.000000	13.

In [15]:
```
house_df.head(30)
```

Out[15]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_buil
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	0.0	0.0	...	7	2170	400.0	195

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_buil
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	0.0	0.0	...	7	1050	910.0	1965
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	0.0	0.0	...	8	1680	0.0	1987
5	7237550310	5/12/2014	1230000.0	4	4.50	5420	101930	1.0	0.0	0.0	...	11	3890	1530.0	2001
6	1321400060	6/27/2014	257500.0	3	2.25	1715	6819	2.0	0.0	0.0	...	7	1715	?	1995
8	2414600126	4/15/2015	229500.0	3	1.00	1780	7470	1.0	0.0	0.0	...	7	1050	730.0	1960
9	3793500160	3/12/2015	323000.0	3	2.50	1890	6560	2.0	0.0	0.0	...	7	1890	0.0	2003
11	9212900260	5/27/2014	468000.0	2	1.00	1160	6000	1.0	0.0	0.0	...	7	860	300.0	1942
13	6054650070	10/7/2014	400000.0	3	1.75	1370	9680	1.0	0.0	0.0	...	7	1370	0.0	1977
14	1175000570	3/12/2015	530000.0	5	2.00	1810	4850	1.5	0.0	0.0	...	7	1810	0.0	1900
15	9297300055	1/24/2015	650000.0	4	3.00	2950	5000	2.0	0.0	3.0	...	9	1980	970.0	1979
16	1875500060	7/31/2014	395000.0	3	2.00	1890	14040	2.0	0.0	0.0	...	7	1890	0.0	1994
17	6865200140	5/29/2014	485000.0	4	1.00	1600	4300	1.5	0.0	0.0	...	7	1600	0.0	1916
18	16000397	12/5/2014	189000.0	2	1.00	1200	9850	1.0	0.0	0.0	...	7	1200	?	1921
19	7983200060	4/24/2015	230000.0	3	1.00	1250	9774	1.0	0.0	0.0	...	7	1250	0.0	1969
20	6300500875	5/14/2014	385000.0	4	1.75	1620	4980	1.0	0.0	0.0	...	7	860	760.0	1947
21	2524049179	8/26/2014	2000000.0	3	2.75	3050	44867	1.0	0.0	4.0	...	9	2330	720.0	1968
22	7137970340	7/3/2014	285000.0	5	2.50	2270	6300	2.0	0.0	0.0	...	8	2270	0.0	1995
24	3814700200	11/20/2014	329000.0	3	2.25	2450	6500	2.0	0.0	0.0	...	8	2450	0.0	1985
25	1202000200	11/3/2014	233000.0	3	2.00	1710	4697	1.5	0.0	0.0	...	6	1710	0.0	1941
27	3303700376	12/1/2014	667000.0	3	1.00	1400	1581	1.5	0.0	0.0	...	8	1400	0.0	1909
29	1873100390	3/2/2015	719000.0	4	2.50	2570	7173	2.0	0.0	0.0	...	8	2570	0.0	2005
30	8562750320	11/10/2014	580500.0	3	2.50	2320	3980	2.0	0.0	0.0	...	8	2320	0.0	2003
31	2426039314	12/1/2014	280000.0	2	1.50	1190	1265	3.0	0.0	0.0	...	7	1190	0.0	2005
32	461000390	6/24/2014	687500.0	4	1.75	2330	5000	1.5	0.0	0.0	...	7	1510	820.0	1929
33	7589200193	11/10/2014	535000.0	3	1.00	1090	3000	1.5	0.0	0.0	...	8	1090	0.0	1929
34	7955080270	12/3/2014	322500.0	4	2.75	2060	6659	1.0	0.0	0.0	...	7	1280	780.0	1981
35	9547205180	6/13/2014	696000.0	3	2.50	2300	3060	1.5	0.0	0.0	...	8	1510	790.0	1930
36	9435300030	5/28/2014	550000.0	4	1.00	1660	34848	1.0	0.0	0.0	...	5	930	730.0	1933
37	2768000400	12/30/2014	640000.0	4	2.00	2360	6000	2.0	0.0	0.0	...	8	2360	0.0	1904

30 rows × 21 columns

In [16]:
```
house_df = house_df[house_df['sqft_basement'] != '?']
```

In [17]:
```
house_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15429 entries, 1 to 21596
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             15429 non-null  int64  
 1   date           15429 non-null  object 
 2   price          15429 non-null  float64
 3   bedrooms       15429 non-null  int64  
 4   bathrooms      15429 non-null  float64
 5   sqft_living    15429 non-null  int64  
 6   sqft_lot       15429 non-null  int64  
 7   floors         15429 non-null  float64
 8   waterfront     15429 non-null  float64
 9   view           15429 non-null  float64
 10  condition      15429 non-null  int64  
 11  grade          15429 non-null  int64  
 12  sqft_above     15429 non-null  int64  
 13  sqft_basement  15429 non-null  object 
 14  yr_built       15429 non-null  int64  
 15  yr_renovated   15429 non-null  float64
 16  zipcode        15429 non-null  int64  
 17  lat            15429 non-null  float64
 18  long           15429 non-null  float64
 19  sqft_living15  15429 non-null  int64  
```

```
 20  sqft_lot15     15429 non-null  int64
dtypes: float64(8), int64(11), object(2)
memory usage: 2.6+ MB
```

In [18]:
```
house_df.describe()
```

Out[18]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	
count	1.542900e+04	1.542900e+04	15429.000000	15429.000000	15429.00000	1.542900e+04	15429.000000	15429.000000	15429.000000	15429.000000	15429.0
mean	4.593825e+09	5.414978e+05	3.378767	2.121508	2085.51656	1.528616e+04	1.494556	0.007518	0.228855	3.410979	7.6
std	2.874791e+09	3.730219e+05	0.934200	0.767027	919.54924	4.199737e+04	0.538903	0.086384	0.759902	0.651825	1.1
min	1.000102e+06	8.200000e+04	1.000000	0.500000	370.00000	5.200000e+02	1.000000	0.000000	0.000000	1.000000	3.0
25%	2.126049e+09	3.200000e+05	3.000000	1.750000	1430.00000	5.050000e+03	1.000000	0.000000	0.000000	3.000000	7.0
50%	3.905082e+09	4.500000e+05	3.000000	2.250000	1920.00000	7.620000e+03	1.500000	0.000000	0.000000	3.000000	7.0
75%	7.334501e+09	6.435000e+05	4.000000	2.500000	2550.00000	1.072000e+04	2.000000	0.000000	0.000000	4.000000	8.0
max	9.895000e+09	7.700000e+06	33.000000	8.000000	13540.00000	1.651359e+06	3.500000	1.000000	4.000000	5.000000	13.0

In [19]:
```
house_df.head(30)
```

Out[19]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_buil
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	0.0	0.0	...	7	2170	400.0	195
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	0.0	0.0	...	7	1050	910.0	196
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	0.0	0.0	...	8	1680	0.0	198
5	7237550310	5/12/2014	1230000.0	4	4.50	5420	101930	1.0	0.0	0.0	...	11	3890	1530.0	200
8	2414600126	4/15/2015	229500.0	3	1.00	1780	7470	1.0	0.0	0.0	...	7	1050	730.0	196
9	3793500160	3/12/2015	323000.0	3	2.50	1890	6560	2.0	0.0	0.0	...	7	1890	0.0	200
11	9212900260	5/27/2014	468000.0	2	1.00	1160	6000	1.0	0.0	0.0	...	7	860	300.0	194
13	6054650070	10/7/2014	400000.0	3	1.75	1370	9680	1.0	0.0	0.0	...	7	1370	0.0	197
14	1175000570	3/12/2015	530000.0	5	2.00	1810	4850	1.5	0.0	0.0	...	7	1810	0.0	190
15	9297300055	1/24/2015	650000.0	4	3.00	2950	5000	2.0	0.0	3.0	...	9	1980	970.0	197
16	1875500060	7/31/2014	395000.0	3	2.00	1890	14040	2.0	0.0	0.0	...	7	1890	0.0	199
17	6865200140	5/29/2014	485000.0	4	1.00	1600	4300	1.5	0.0	0.0	...	7	1600	0.0	191
19	7983200060	4/24/2015	230000.0	3	1.00	1250	9774	1.0	0.0	0.0	...	7	1250	0.0	196
20	6300500875	5/14/2014	385000.0	4	1.75	1620	4980	1.0	0.0	0.0	...	7	860	760.0	194
21	2524049179	8/26/2014	2000000.0	3	2.75	3050	44867	1.0	0.0	4.0	...	9	2330	720.0	196
22	7137970340	7/3/2014	285000.0	5	2.50	2270	6300	2.0	0.0	0.0	...	8	2270	0.0	199
24	3814700200	11/20/2014	329000.0	3	2.25	2450	6500	2.0	0.0	0.0	...	8	2450	0.0	198
25	1202000200	11/3/2014	233000.0	3	2.00	1710	4697	1.5	0.0	0.0	...	6	1710	0.0	194
27	3303700376	12/1/2014	667000.0	3	1.00	1400	1581	1.5	0.0	0.0	...	8	1400	0.0	190
29	1873100390	3/2/2015	719000.0	4	2.50	2570	7173	2.0	0.0	0.0	...	8	2570	0.0	200
30	8562750320	11/10/2014	580500.0	3	2.50	2320	3980	2.0	0.0	0.0	...	8	2320	0.0	200
31	2426039314	12/1/2014	280000.0	2	1.50	1190	1265	3.0	0.0	0.0	...	7	1190	0.0	200
32	461000390	6/24/2014	687500.0	4	1.75	2330	5000	1.5	0.0	0.0	...	7	1510	820.0	192
33	7589200193	11/10/2014	535000.0	3	1.00	1090	3000	1.5	0.0	0.0	...	8	1090	0.0	192
34	7955080270	12/3/2014	322500.0	4	2.75	2060	6659	1.0	0.0	0.0	...	7	1280	780.0	198
35	9547205180	6/13/2014	696000.0	3	2.50	2300	3060	1.5	0.0	0.0	...	8	1510	790.0	193
36	9435300030	5/28/2014	550000.0	4	1.00	1660	34848	1.0	0.0	0.0	...	5	930	730.0	193
37	2768000400	12/30/2014	640000.0	4	2.00	2360	6000	2.0	0.0	0.0	...	8	2360	0.0	190
38	7895500070	2/13/2015	240000.0	4	1.00	1220	8075	1.0	0.0	0.0	...	7	890	330.0	196
39	2078500320	6/20/2014	605000.0	4	2.50	2620	7553	2.0	0.0	0.0	...	8	2620	0.0	199

30 rows × 21 columns

In [20]:
```
house_df = house_df.drop('yr_renovated', axis = 1)
```

In [21]:
```
house_df.head(30)
```

Out[21]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	0.0	0.0	3	7	2170	400.0
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	0.0	0.0	5	7	1050	910.0
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	0.0	0.0	3	8	1680	0.0
5	7237550310	5/12/2014	1230000.0	4	4.50	5420	101930	1.0	0.0	0.0	3	11	3890	1530.0
8	2414600126	4/15/2015	229500.0	3	1.00	1780	7470	1.0	0.0	0.0	3	7	1050	730.0
9	3793500160	3/12/2015	323000.0	3	2.50	1890	6560	2.0	0.0	0.0	3	7	1890	0.0
11	9212900260	5/27/2014	468000.0	2	1.00	1160	6000	1.0	0.0	0.0	4	7	860	300.0
13	6054650070	10/7/2014	400000.0	3	1.75	1370	9680	1.0	0.0	0.0	4	7	1370	0.0
14	1175000570	3/12/2015	530000.0	5	2.00	1810	4850	1.5	0.0	0.0	3	7	1810	0.0
15	9297300055	1/24/2015	650000.0	4	3.00	2950	5000	2.0	0.0	3.0	3	9	1980	970.0
16	1875500060	7/31/2014	395000.0	3	2.00	1890	14040	2.0	0.0	0.0	3	7	1890	0.0
17	6865200140	5/29/2014	485000.0	4	1.00	1600	4300	1.5	0.0	0.0	4	7	1600	0.0
19	7983200060	4/24/2015	230000.0	3	1.00	1250	9774	1.0	0.0	0.0	4	7	1250	0.0
20	6300500875	5/14/2014	385000.0	4	1.75	1620	4980	1.0	0.0	0.0	4	7	860	760.0
21	2524049179	8/26/2014	2000000.0	3	2.75	3050	44867	1.0	0.0	4.0	3	9	2330	720.0
22	7137970340	7/3/2014	285000.0	5	2.50	2270	6300	2.0	0.0	0.0	3	8	2270	0.0
24	3814700200	11/20/2014	329000.0	3	2.25	2450	6500	2.0	0.0	0.0	4	8	2450	0.0
25	1202000200	11/3/2014	233000.0	3	2.00	1710	4697	1.5	0.0	0.0	5	6	1710	0.0
27	3303700376	12/1/2014	667000.0	3	1.00	1400	1581	1.5	0.0	0.0	5	8	1400	0.0
29	1873100390	3/2/2015	719000.0	4	2.50	2570	7173	2.0	0.0	0.0	3	8	2570	0.0
30	8562750320	11/10/2014	580500.0	3	2.50	2320	3980	2.0	0.0	0.0	3	8	2320	0.0
31	2426039314	12/1/2014	280000.0	2	1.50	1190	1265	3.0	0.0	0.0	3	7	1190	0.0
32	461000390	6/24/2014	687500.0	4	1.75	2330	5000	1.5	0.0	0.0	4	7	1510	820.0
33	7589200193	11/10/2014	535000.0	3	1.00	1090	3000	1.5	0.0	0.0	4	8	1090	0.0
34	7955080270	12/3/2014	322500.0	4	2.75	2060	6659	1.0	0.0	0.0	3	7	1280	780.0
35	9547205180	6/13/2014	696000.0	3	2.50	2300	3060	1.5	0.0	0.0	3	8	1510	790.0
36	9435300030	5/28/2014	550000.0	4	1.00	1660	34848	1.0	0.0	0.0	1	5	930	730.0
37	2768000400	12/30/2014	640000.0	4	2.00	2360	6000	2.0	0.0	0.0	4	8	2360	0.0
38	7895500070	2/13/2015	240000.0	4	1.00	1220	8075	1.0	0.0	0.0	2	7	890	330.0
39	2078500320	6/20/2014	605000.0	4	2.50	2620	7553	2.0	0.0	0.0	3	8	2620	0.0

In [22]:
```
house_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15429 entries, 1 to 21596
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             15429 non-null  int64
 1   date           15429 non-null  object
 2   price          15429 non-null  float64
 3   bedrooms       15429 non-null  int64
 4   bathrooms      15429 non-null  float64
 5   sqft_living    15429 non-null  int64
 6   sqft_lot       15429 non-null  int64
 7   floors         15429 non-null  float64
 8   waterfront     15429 non-null  float64
 9   view           15429 non-null  float64
 10  condition      15429 non-null  int64
 11  grade          15429 non-null  int64
 12  sqft_above     15429 non-null  int64
 13  sqft_basement  15429 non-null  object
 14  yr_built       15429 non-null  int64
 15  zipcode        15429 non-null  int64
 16  lat            15429 non-null  float64
 17  long           15429 non-null  float64
```

```
 18  sqft_living15  15429 non-null  int64
 19  sqft_lot15     15429 non-null  int64
dtypes: float64(7), int64(11), object(2)
memory usage: 2.5+ MB
```

In [23]:
```
house_df = house_df.astype({'id': int, 'date': object, 'price': float, 'bedrooms': float, 'bathrooms': float, 'sqft_living': float,
```

In [24]:
```
house_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15429 entries, 1 to 21596
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             15429 non-null  int32
 1   date           15429 non-null  object
 2   price          15429 non-null  float64
 3   bedrooms       15429 non-null  float64
 4   bathrooms      15429 non-null  float64
 5   sqft_living    15429 non-null  float64
 6   sqft_lot       15429 non-null  float64
 7   floors         15429 non-null  float64
 8   waterfront     15429 non-null  float64
 9   view           15429 non-null  float64
 10  condition      15429 non-null  float64
 11  grade          15429 non-null  float64
 12  sqft_above     15429 non-null  float64
 13  sqft_basement  15429 non-null  float64
 14  yr_built       15429 non-null  float64
 15  zipcode        15429 non-null  float64
 16  lat            15429 non-null  float64
 17  long           15429 non-null  float64
 18  sqft_living15  15429 non-null  float64
 19  sqft_lot15     15429 non-null  float64
dtypes: float64(18), int32(1), object(1)
memory usage: 2.4+ MB
```

In [25]:
```
house_df.head(30)
```

Out[25]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement
1	2119132896	12/9/2014	538000.0	3.0	2.25	2570.0	7242.0	2.0	0.0	0.0	3.0	7.0	2170.0	400.0
3	-1807766421	12/9/2014	604000.0	4.0	3.00	1960.0	5000.0	1.0	0.0	0.0	5.0	7.0	1050.0	910.0
4	1954400510	2/18/2015	510000.0	3.0	2.00	1680.0	8080.0	1.0	0.0	0.0	3.0	8.0	1680.0	0.0
5	-1352384282	5/12/2014	1230000.0	4.0	4.50	5420.0	101930.0	1.0	0.0	0.0	3.0	11.0	3890.0	1530.0
8	-1880367170	4/15/2015	229500.0	3.0	1.00	1780.0	7470.0	1.0	0.0	0.0	3.0	7.0	1050.0	730.0
9	-501467136	3/12/2015	323000.0	3.0	2.50	1890.0	6560.0	2.0	0.0	0.0	3.0	7.0	1890.0	0.0
11	622965668	5/27/2014	468000.0	2.0	1.00	1160.0	6000.0	1.0	0.0	0.0	4.0	7.0	860.0	300.0
13	1759682774	10/7/2014	400000.0	3.0	1.75	1370.0	9680.0	1.0	0.0	0.0	4.0	7.0	1370.0	0.0
14	1175000570	3/12/2015	530000.0	5.0	2.00	1810.0	4850.0	1.5	0.0	0.0	3.0	7.0	1810.0	0.0
15	707365463	1/24/2015	650000.0	4.0	3.00	2950.0	5000.0	2.0	0.0	3.0	3.0	9.0	1980.0	970.0
16	1875500060	7/31/2014	395000.0	3.0	2.00	1890.0	14040.0	2.0	0.0	0.0	3.0	7.0	1890.0	0.0
17	-1724734452	5/29/2014	485000.0	4.0	1.00	1600.0	4300.0	1.5	0.0	0.0	4.0	7.0	1600.0	0.0
19	-606734532	4/24/2015	230000.0	3.0	1.00	1250.0	9774.0	1.0	0.0	0.0	4.0	7.0	1250.0	0.0
20	2005533579	5/14/2014	385000.0	4.0	1.75	1620.0	4980.0	1.0	0.0	0.0	4.0	7.0	860.0	760.0
21	-1770918117	8/26/2014	2000000.0	3.0	2.75	3050.0	44867.0	1.0	0.0	4.0	3.0	9.0	2330.0	720.0
22	-1451964252	7/3/2014	285000.0	5.0	2.50	2270.0	6300.0	2.0	0.0	0.0	3.0	8.0	2270.0	0.0
24	-480267096	11/20/2014	329000.0	3.0	2.25	2450.0	6500.0	2.0	0.0	0.0	4.0	8.0	2450.0	0.0
25	1202000200	11/3/2014	233000.0	3.0	2.00	1710.0	4697.0	1.5	0.0	0.0	5.0	6.0	1710.0	0.0
27	-991266920	12/1/2014	667000.0	3.0	1.00	1400.0	1581.0	1.5	0.0	0.0	5.0	8.0	1400.0	0.0
29	1873100390	3/2/2015	719000.0	4.0	2.50	2570.0	7173.0	2.0	0.0	0.0	3.0	8.0	2570.0	0.0
30	-27184272	11/10/2014	580500.0	3.0	2.50	2320.0	3980.0	2.0	0.0	0.0	3.0	8.0	2320.0	0.0
31	-1868927982	12/1/2014	280000.0	2.0	1.50	1190.0	1265.0	3.0	0.0	0.0	3.0	7.0	1190.0	0.0
32	461000390	6/24/2014	687500.0	4.0	1.75	2330.0	5000.0	1.5	0.0	0.0	4.0	7.0	1510.0	820.0
33	-1000734399	11/10/2014	535000.0	3.0	1.00	1090.0	3000.0	1.5	0.0	0.0	4.0	8.0	1090.0	0.0
34	-634854322	12/3/2014	322500.0	4.0	2.75	2060.0	6659.0	1.0	0.0	0.0	3.0	7.0	1280.0	780.0
35	957270588	6/13/2014	696000.0	3.0	2.50	2300.0	3060.0	1.5	0.0	0.0	3.0	8.0	1510.0	790.0

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement
36	845365438	5/28/2014	550000.0	4.0	1.00	1660.0	34848.0	1.0	0.0	0.0	1.0	5.0	930.0	730.0
37	-1526966896	12/30/2014	640000.0	4.0	2.00	2360.0	6000.0	2.0	0.0	0.0	4.0	8.0	2360.0	0.0
38	-694434522	2/13/2015	240000.0	4.0	1.00	1220.0	8075.0	1.0	0.0	0.0	2.0	7.0	890.0	330.0
39	2078500320	6/20/2014	605000.0	4.0	2.50	2620.0	7553.0	2.0	0.0	0.0	3.0	8.0	2620.0	0.0

Basic Visualizations

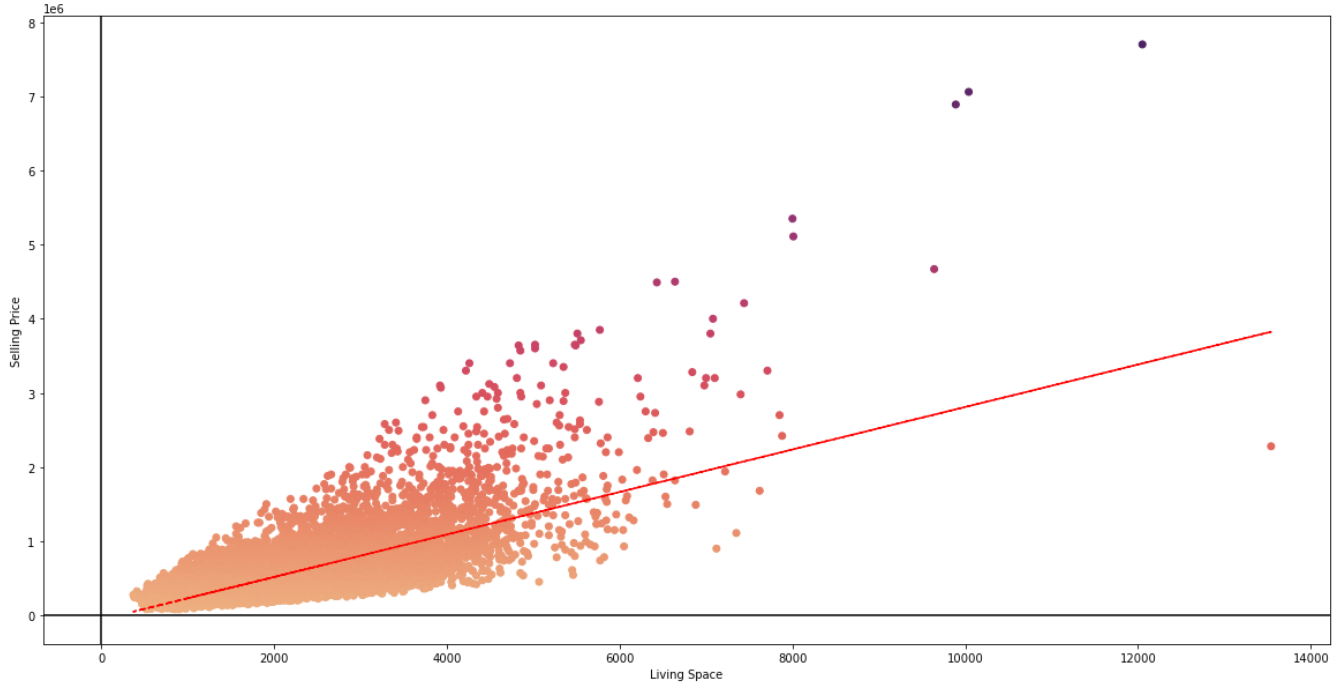
In [6]:
```
# With our dataframe cleaned, move forward with the creation of basic visualizations to figure out potential significant
# features of selling price predictors.
```

In [27]:
```
x = house_df['sqft_living']
y = house_df['price']

plt.figure(figsize = (20, 10))
plt.scatter (x, y, c=y, cmap = 'flare')
plt.axvline (0, c = 'black')
plt.axhline(0, c = 'black')
plt.xlabel("Living Space")
plt.ylabel("Selling Price")

z = np.polyfit (x, y, 1)
p = np.poly1d(z)
plt.plot(x,p(x), "r--")

plt.show()
```

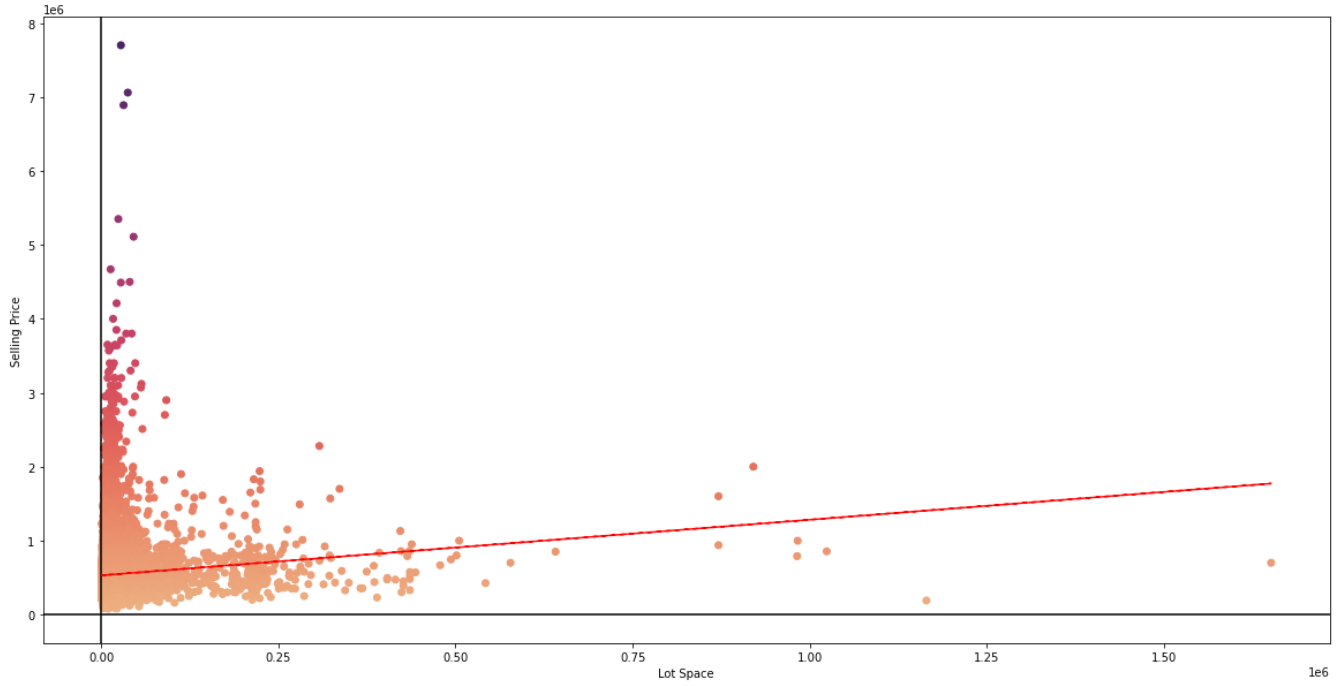


In [28]:
```
x = house_df['sqft_lot']
y = house_df['price']

plt.figure(figsize = (20, 10))
plt.scatter (x, y, c=y, cmap = 'flare')
plt.axvline (0, c = 'black')
plt.axhline(0, c = 'black')
plt.xlabel("Lot Space")
plt.ylabel("Selling Price")

z = np.polyfit (x, y, 1)
p = np.poly1d(z)
plt.plot(x,p(x), "r--")

plt.show()
```



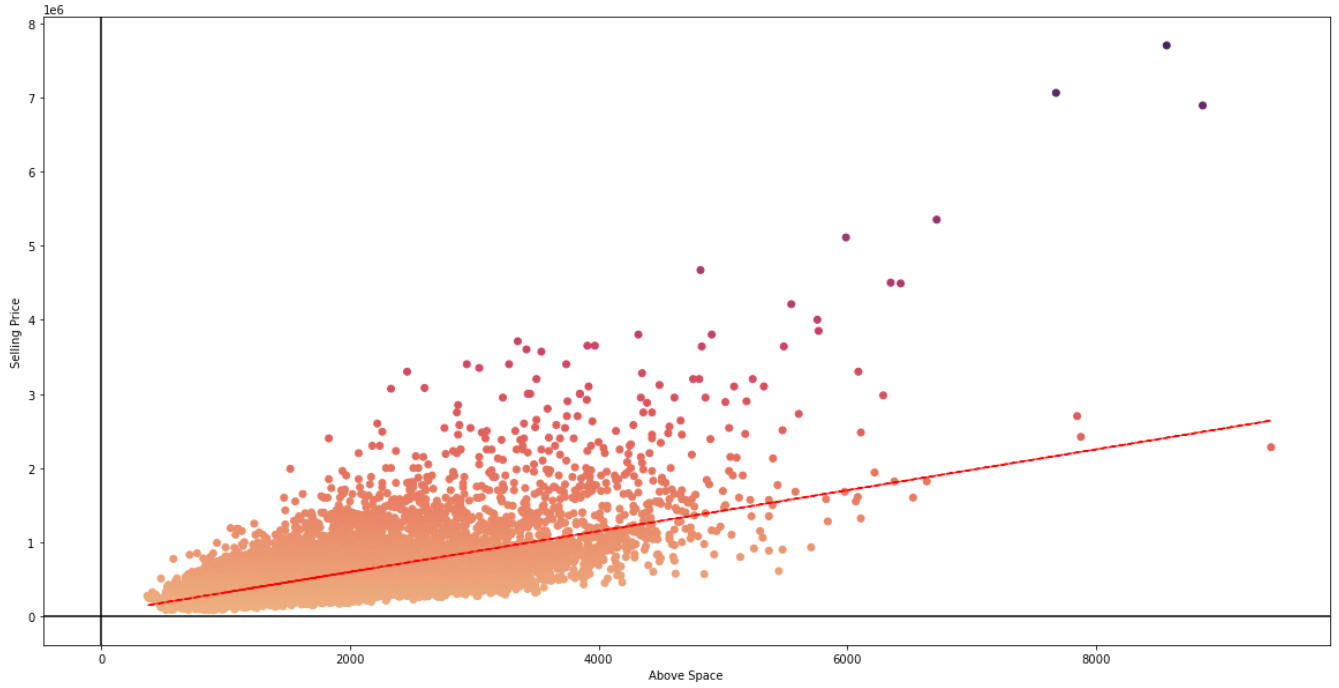
In [29]:

```
x = house_df['sqft_above']
y = house_df['price']

plt.figure(figsize = (20, 10))
plt.scatter (x, y, c=y, cmap = 'flare')
plt.axvline (0, c = 'black')
plt.axhline(0, c = 'black')
plt.xlabel("Above Space")
plt.ylabel("Selling Price")

z = np.polyfit (x, y, 1)
p = np.poly1d(z)
plt.plot(x,p(x), "r--")

plt.show()
```



In [30]:

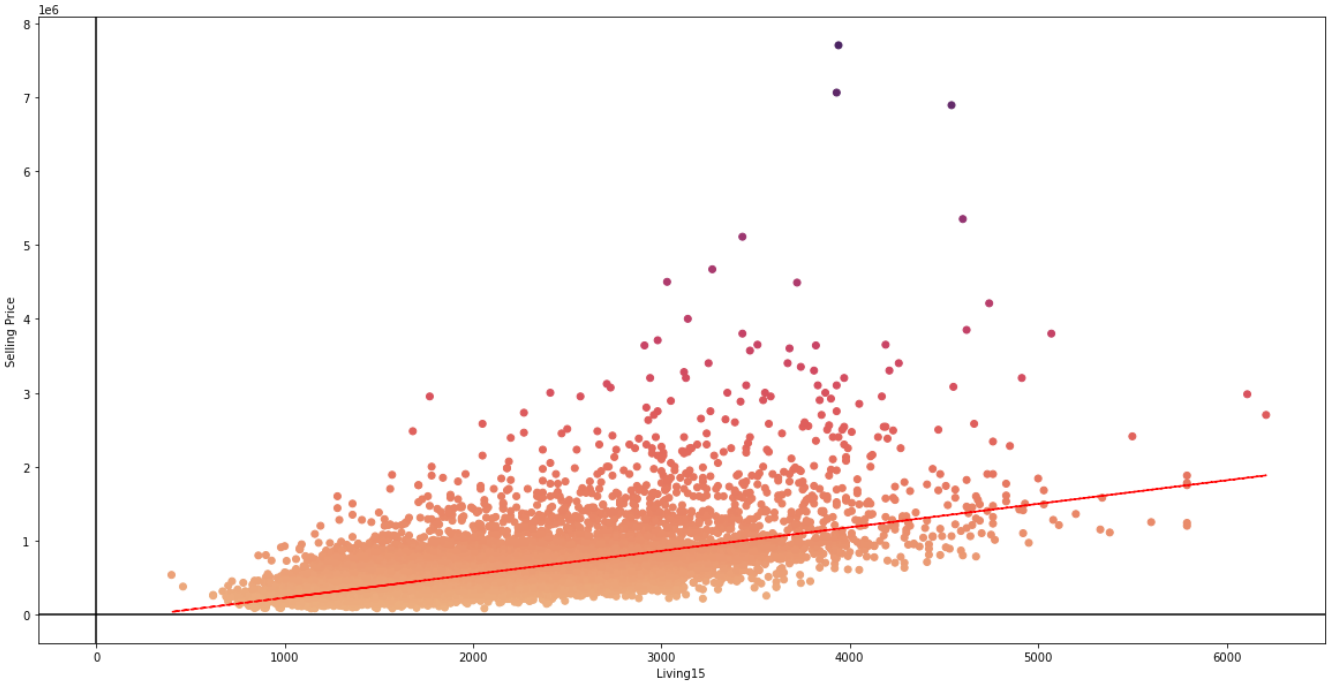
```
x = house_df['sqft_living15']
y = house_df['price']

plt.figure(figsize = (20, 10))
plt.scatter (x, y, c=y, cmap = 'flare')
plt.axvline (0, c = 'black')
```

```
plt.axhline(0, c = 'black')
plt.xlabel("Living15")
plt.ylabel("Selling Price")

z = np.polyfit (x, y, 1)
p = np.poly1d(z)
plt.plot(x,p(x), "r--")

plt.show()
```



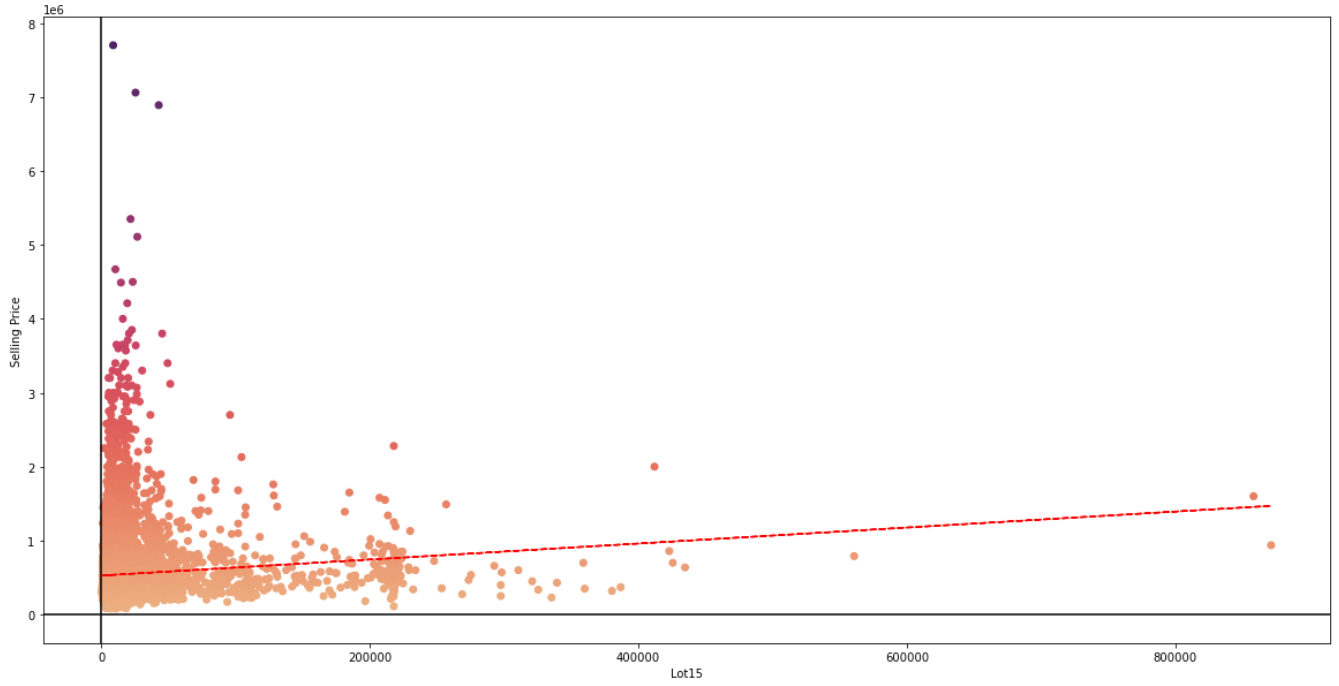
In [31]:

```
x = house_df['sqft_lot15']
y = house_df['price']

plt.figure(figsize = (20, 10))
plt.scatter (x, y, c=y, cmap = 'flare')
plt.axvline (0, c = 'black')
plt.axhline(0, c = 'black')
plt.xlabel("Lot15")
plt.ylabel("Selling Price")

z = np.polyfit (x, y, 1)
p = np.poly1d(z)
plt.plot(x,p(x), "r--")

plt.show()
```



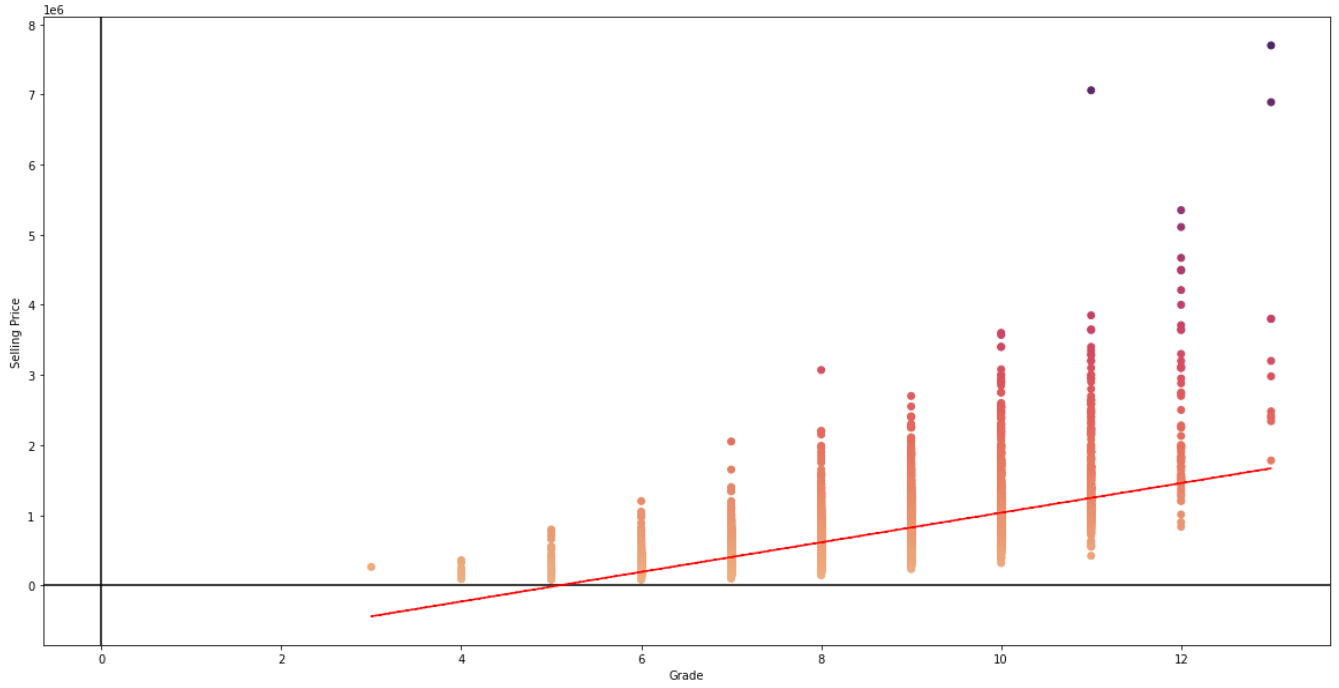
In [32]:

```
x = house_df['grade']
y = house_df['price']

plt.figure(figsize = (20, 10))
plt.scatter (x, y, c=y, cmap = 'flare')
plt.axvline (0, c = 'black')
plt.axhline(0, c = 'black')
plt.xlabel("Grade")
plt.ylabel("Selling Price")

z = np.polyfit (x, y, 1)
p = np.poly1d(z)
plt.plot(x,p(x), "r--")

plt.show()
```



In [33]:

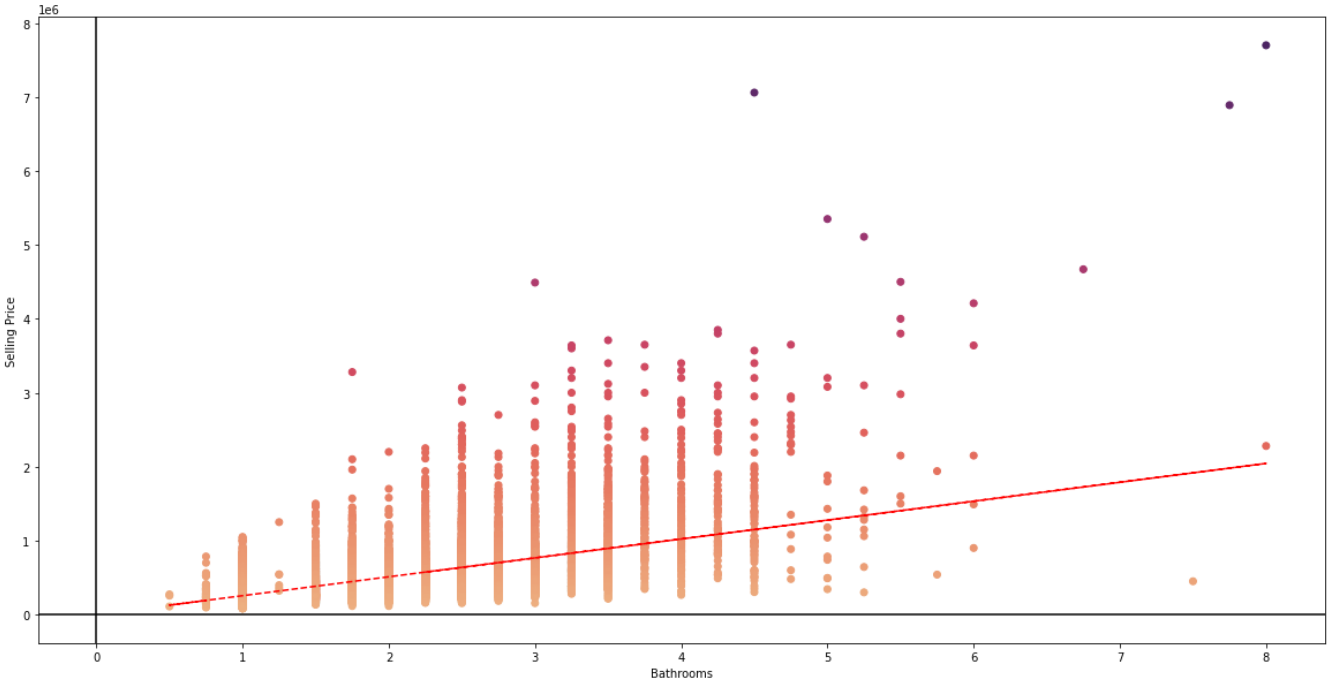
```
x = house_df['bathrooms']
y = house_df['price']

plt.figure(figsize = (20, 10))
plt.scatter (x, y, c=y, cmap = 'flare')
plt.axvline (0, c = 'black')
```

```
plt.axhline(0, c = 'black')
plt.xlabel("Bathrooms")
plt.ylabel("Selling Price")

z = np.polyfit (x, y, 1)
p = np.poly1d(z)
plt.plot(x,p(x), "r--")

plt.show()
```



Creation of Simple Linear Regression Models

In [7]:
```
# With the basic visualizations completed, move towards the creation of simple linear regression models
# which look at single potential significant predictors of a house's selling price.
```

In [35]:
```
import statsmodels.api as sm
import statsmodels.formula.api as smf
import scipy.stats as stats
```

In [36]:
```
f1 = 'price~sqft_living'
model_1 = smf.ols(formula=f1, data=house_df).fit()
model_1.summary()
```

Out[36]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.499
Model:	OLS	**Adj. R-squared:**	0.499
Method:	Least Squares	**F-statistic:**	1.537e+04
Date:	Mon, 09 Aug 2021	**Prob (F-statistic):**	0.00
Time:	20:34:53	**Log-Likelihood:**	-2.1450e+05
No. Observations:	15429	**AIC:**	4.290e+05
Df Residuals:	15427	**BIC:**	4.290e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-5.62e+04	5268.236	-10.668	0.000	-6.65e+04	-4.59e+04
sqft_living	286.5963	2.311	123.992	0.000	282.066	291.127

Omnibus:	10920.696	**Durbin-Watson:**	1.975
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	451159.757

Skew:	2.916	**Prob(JB):**	0.00
Kurtosis:	28.841	**Cond. No.**	5.65e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.65e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [37]:
```
f2 = 'price~sqft_above'
model_2 = smf.ols(formula=f2, data=house_df).fit()
model_2.summary()
```

Out[37]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.375
Model:	OLS	**Adj. R-squared:**	0.375
Method:	Least Squares	**F-statistic:**	9256.
Date:	Mon, 09 Aug 2021	**Prob (F-statistic):**	0.00
Time:	20:34:53	**Log-Likelihood:**	-2.1621e+05
No. Observations:	15429	**AIC:**	4.324e+05
Df Residuals:	15427	**BIC:**	4.324e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	4.735e+04	5658.405	8.367	0.000	3.63e+04	5.84e+04
sqft_above	275.4926	2.863	96.210	0.000	269.880	281.105

Omnibus:	12100.821	**Durbin-Watson:**	1.980
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	591840.592
Skew:	3.364	**Prob(JB):**	0.00
Kurtosis:	32.586	**Cond. No.**	4.71e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.71e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [38]:
```
f3 = 'price~sqft_living15'
model_3 = smf.ols(formula=f3, data=house_df).fit()
model_3.summary()
```

Out[38]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.340
Model:	OLS	**Adj. R-squared:**	0.340
Method:	Least Squares	**F-statistic:**	7960.
Date:	Mon, 09 Aug 2021	**Prob (F-statistic):**	0.00
Time:	20:34:53	**Log-Likelihood:**	-2.1663e+05
No. Observations:	15429	**AIC:**	4.333e+05
Df Residuals:	15427	**BIC:**	4.333e+05
Df Model:	1		
Covariance Type:	nonrobust		

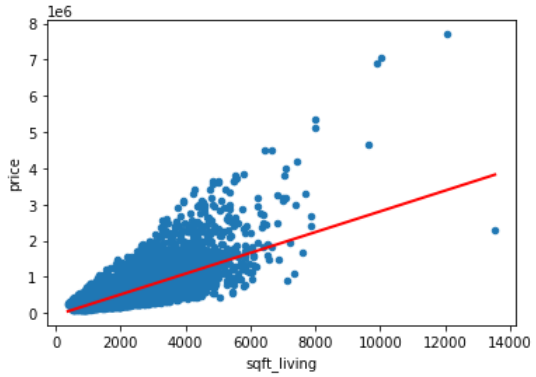
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-9.12e+04	7499.318	-12.161	0.000	-1.06e+05	-7.65e+04
sqft_living15	317.7893	3.562	89.218	0.000	310.807	324.771

Omnibus:	15050.272	**Durbin-Watson:**	1.981
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1693305.725
Skew:	4.475	**Prob(JB):**	0.00
Kurtosis:	53.536	**Cond. No.**	6.47e+03

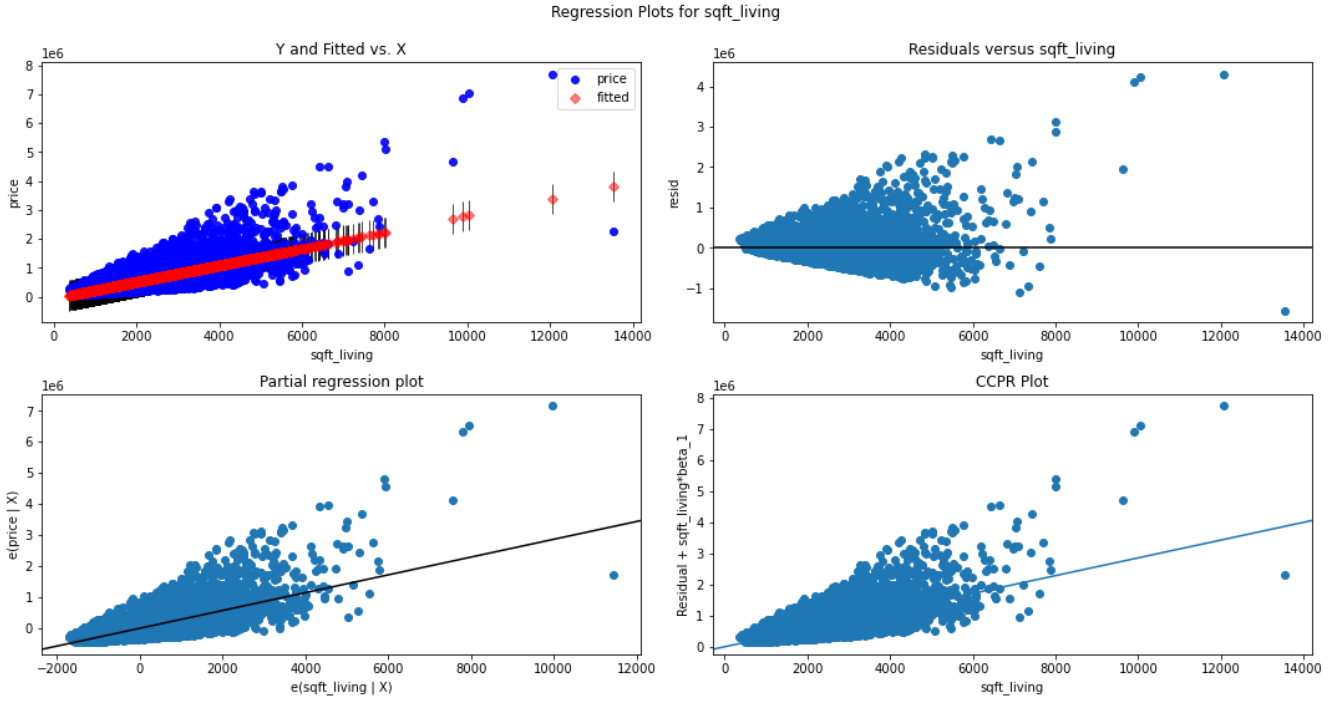
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.47e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [39]:
```
x_new = pd.DataFrame({'sqft_living': [house_df.sqft_living.min(), house_df.sqft_living.max()]})
preds = model_1.predict(x_new)
```

In [40]:
```
house_df.plot(kind = 'scatter', x = 'sqft_living', y = 'price')
plt.plot(x_new, preds, c = 'red', linewidth = 2)
plt.show()
```



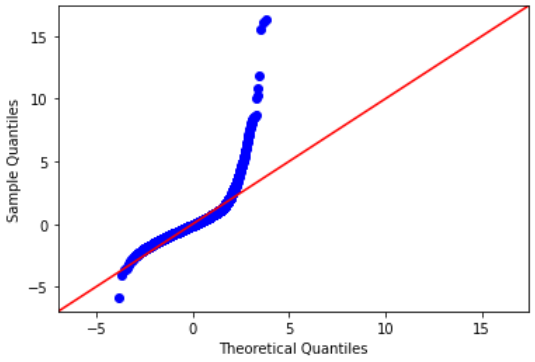
In [41]:
```
fig1 = plt.figure(figsize = (15, 8))
fig1 = sm.graphics.plot_regress_exog(model_1, "sqft_living", fig = fig1)
plt.show()
```



In [42]:
```
residuals = model_1.resid
fig = sm.graphics.qqplot(residuals, dist=stats.norm, line='45', fit=True)
fig.show()
```

```
<ipython-input-42-307a0785a08b>:3: UserWarning: Matplotlib is currently using module://ipykernel.pylab.backend_inline, which is a no
```

```
n-GUI backend, so cannot show the figure.
  fig.show()
```



Creation of Multiple Linear Regression Model

In [63]:
```
# With the initial investigation of housing data provided, move towards the creation of a multiple linear regression model
# with an interative approach, in order to consolidate and indicate statistically significant predictors of a house's
# selling price on the market.
```

In [70]:
```
house_df.head()
```

Out[70]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement
1	2119132896	12/9/2014	538000.0	3.0	2.25	2570.0	7242.0	2.0	0.0	0.0	3.0	7.0	2170.0	400.0
3	-1807766421	12/9/2014	604000.0	4.0	3.00	1960.0	5000.0	1.0	0.0	0.0	5.0	7.0	1050.0	910.0
4	1954400510	2/18/2015	510000.0	3.0	2.00	1680.0	8080.0	1.0	0.0	0.0	3.0	8.0	1680.0	0.0
5	-1352384282	5/12/2014	1230000.0	4.0	4.50	5420.0	101930.0	1.0	0.0	0.0	3.0	11.0	3890.0	1530.0
8	-1880367170	4/15/2015	229500.0	3.0	1.00	1780.0	7470.0	1.0	0.0	0.0	3.0	7.0	1050.0	730.0

In [71]:
```
house_df_2 = house_df[['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'gr
```

For the creation of our MLR model, split the data set provided in a 7 to 3 ratio to train and test the desired model.

In [72]:
```
from sklearn.model_selection import train_test_split
```

In [73]:
```
np.random.seed(0)
```

In [74]:
```
df_train, df_test = train_test_split(house_df_2, train_size = 0.7, test_size = 0.3, random_state = 100)
```

In [75]:
```
house_df_2.head()
```

Out[75]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	sqft_living15	sqft_lot15
1	538000.0	3.0	2.25	2570.0	7242.0	2.0	0.0	0.0	3.0	7.0	2170.0	400.0	1690.0	7639.0
3	604000.0	4.0	3.00	1960.0	5000.0	1.0	0.0	0.0	5.0	7.0	1050.0	910.0	1360.0	5000.0
4	510000.0	3.0	2.00	1680.0	8080.0	1.0	0.0	0.0	3.0	8.0	1680.0	0.0	1800.0	7503.0
5	1230000.0	4.0	4.50	5420.0	101930.0	1.0	0.0	0.0	3.0	11.0	3890.0	1530.0	4760.0	101930.0
8	229500.0	3.0	1.00	1780.0	7470.0	1.0	0.0	0.0	3.0	7.0	1050.0	730.0	1780.0	8113.0

Scale and normalize features using the MinMaxScaler fuction from the sklearn library.

In [76]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [77]:
```
scaler = MinMaxScaler()
```

In [78]:
```
num_vars = ['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade', 'sqft
```

In [79]:
```
df_train[num_vars] = scaler.fit_transform(df_train[num_vars])
```

```
<ipython-input-79-83f96a893732>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_train[num_vars] = scaler.fit_transform(df_train[num_vars])
C:\Users\rychu\anaconda3\lib\site-packages\pandas\core\indexing.py:1637: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
C:\Users\rychu\anaconda3\lib\site-packages\pandas\core\indexing.py:692: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  iloc._setitem_with_indexer(indexer, value, self.name)
```

In [80]:
```
df_train
```

Out[80]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	sqft_living15	sqft_lot
10855	0.087627	0.09375	0.133333	0.128322	0.002108	0.2	0.0	0.5	0.50	0.4	0.133850	0.116223	0.121739	0.0038
15562	0.049951	0.06250	0.166667	0.125285	0.006406	0.0	0.0	0.0	0.75	0.4	0.122788	0.130751	0.273043	0.0115
3862	0.047194	0.06250	0.133333	0.127563	0.003552	0.0	0.0	0.0	0.50	0.4	0.108407	0.169492	0.196522	0.0075
20426	0.038924	0.06250	0.266667	0.091875	0.001548	0.4	0.0	0.0	0.50	0.5	0.133850	0.000000	0.217391	0.0026
11375	0.031178	0.06250	0.200000	0.114655	0.006202	0.0	0.0	0.0	1.00	0.3	0.063053	0.227603	0.278261	0.0233
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
16938	0.006104	0.00000	0.066667	0.021260	0.008992	0.0	0.0	0.0	0.75	0.2	0.030973	0.000000	0.203478	0.0083
11219	0.027896	0.09375	0.333333	0.132878	0.004319	0.4	0.0	0.0	0.50	0.5	0.193584	0.000000	0.313043	0.0083
19761	0.335740	0.12500	0.500000	0.325740	0.009498	0.4	0.0	0.5	0.50	0.8	0.474558	0.000000	0.500870	0.0178
9695	0.023302	0.06250	0.200000	0.177677	0.002714	0.4	0.0	0.0	0.50	0.3	0.258850	0.000000	0.067826	0.0049
7893	0.037611	0.06250	0.166667	0.070615	0.002014	0.0	0.0	0.0	0.50	0.4	0.102876	0.000000	0.144348	0.0036

10800 rows × 14 columns

In [81]:
```
# Split the training data set into X and Y
```

In [82]:
```
y_train = df_train.pop('price')
```

In [83]:
```
X_train = df_train
```

In [84]:
```
# Build the linear Model
```

In [85]:
```
import statsmodels.api as sm
```

In [86]:
```
X_train_LM = sm.add_constant(X_train)
```

In [87]:
```
LR_1 = sm.OLS(y_train, X_train_LM.astype(float)).fit()
```

In [88]:
```
LR_1.summary()
```

Out[88]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.610
Model:	OLS	**Adj. R-squared:**	0.610
Method:	Least Squares	**F-statistic:**	1407.
Date:	Mon, 09 Aug 2021	**Prob (F-statistic):**	0.00

Time:	20:40:09	**Log-Likelihood:**	22072.
No. Observations:	10800	**AIC:**	-4.412e+04
Df Residuals:	10787	**BIC:**	-4.402e+04
Df Model:	12		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0495	0.002	-24.282	0.000	-0.054	-0.046
bedrooms	-0.1571	0.013	-12.159	0.000	-0.182	-0.132
bathrooms	-0.0098	0.005	-1.941	0.052	-0.020	9.6e-05
sqft_living	0.1354	0.003	38.861	0.000	0.129	0.142
sqft_lot	0.0107	0.017	0.615	0.538	-0.023	0.045
floors	-0.0038	0.002	-1.972	0.049	-0.008	-2.2e-05
waterfront	0.0838	0.004	21.758	0.000	0.076	0.091
view	0.0337	0.002	18.119	0.000	0.030	0.037
condition	0.0279	0.002	14.345	0.000	0.024	0.032
grade	0.1250	0.005	27.239	0.000	0.116	0.134
sqft_above	0.1613	0.005	34.035	0.000	0.152	0.171
sqft_basement	0.0786	0.003	24.002	0.000	0.072	0.085
sqft_living15	-0.0024	0.004	-0.576	0.564	-0.011	0.006
sqft_lot15	-0.0887	0.014	-6.516	0.000	-0.115	-0.062

Omnibus:	8500.808	**Durbin-Watson:**	2.006
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	661410.477
Skew:	3.207	**Prob(JB):**	0.00
Kurtosis:	40.798	**Cond. No.**	2.42e+16

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 3.33e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

Check for Multicollinearity

In [90]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [9]:
```
# Creation of dataframe that contains names of all featured variables + Variance inflation factor.
# VIF = quantifies severity of multicollinearity in an OLS regression analysis.
```

In [99]:
```
vif = pd.DataFrame()
```

In [100…
```
vif['Features'] = X_train.columns
```

In [101…
```
vif['VIF'] = [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape[1])]
```

In [102…
```
vif['VIF'] = round(vif['VIF'], 2)
```

In [103…
```
vif = vif.sort_values(by ="VIF", ascending = False)
```

In [104…
```
vif
```

Out[104…

	Features	VIF
2	sqft_living	inf

	Features	VIF
9	sqft_above	inf
10	sqft_basement	inf
8	grade	31.62
11	sqft_living15	16.67
1	bathrooms	15.90
0	bedrooms	10.62
7	condition	9.73
4	floors	3.45
12	sqft_lot15	2.70
3	sqft_lot	2.53
6	view	1.48
5	waterfront	1.20

In [145…
```
X = X_train.drop('sqft_living', 1,)
```

In [146…
```
X_train_LM = sm.add_constant(X)
```

In [147…
```
LR_2 = sm.OLS(y_train, X_train_LM).fit()
```

In [148…
```
LR_2.summary()
```

Out[148…

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.610
Model:	OLS	**Adj. R-squared:**	0.610
Method:	Least Squares	**F-statistic:**	1407.
Date:	Mon, 09 Aug 2021	**Prob (F-statistic):**	0.00
Time:	21:10:24	**Log-Likelihood:**	22072.
No. Observations:	10800	**AIC:**	-4.412e+04
Df Residuals:	10787	**BIC:**	-4.402e+04
Df Model:	12		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0495	0.002	-24.282	0.000	-0.054	-0.046
bedrooms	-0.1571	0.013	-12.159	0.000	-0.182	-0.132
bathrooms	-0.0098	0.005	-1.941	0.052	-0.020	9.6e-05
sqft_lot	0.0107	0.017	0.615	0.538	-0.023	0.045
floors	-0.0038	0.002	-1.972	0.049	-0.008	-2.2e-05
waterfront	0.0838	0.004	21.758	0.000	0.076	0.091
view	0.0337	0.002	18.119	0.000	0.030	0.037
condition	0.0279	0.002	14.345	0.000	0.024	0.032
grade	0.1250	0.005	27.239	0.000	0.116	0.134
sqft_above	0.2543	0.007	36.010	0.000	0.240	0.268
sqft_basement	0.1211	0.004	31.763	0.000	0.114	0.129
sqft_living15	-0.0024	0.004	-0.576	0.564	-0.011	0.006
sqft_lot15	-0.0887	0.014	-6.516	0.000	-0.115	-0.062

Omnibus:	8500.808	**Durbin-Watson:**	2.006
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	661410.477
Skew:	3.207	**Prob(JB):**	0.00
Kurtosis:	40.798	**Cond. No.**	91.3

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

In [149…
```
vif = pd.DataFrame()
vif['Features'] = X.columns
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by ="VIF", ascending = False)

vif
```

Out[149…

	Features	VIF
7	grade	31.62
10	sqft_living15	16.67
8	sqft_above	15.92
1	bathrooms	15.90
0	bedrooms	10.62
6	condition	9.73
3	floors	3.45
11	sqft_lot15	2.70
2	sqft_lot	2.53
9	sqft_basement	2.49
5	view	1.48
4	waterfront	1.20

In [150…
```
X2 = X.drop('grade', 1,)
X_train_LM = sm.add_constant(X2)
LR_3 = sm.OLS(y_train, X_train_LM).fit()
LR_3.summary()
```

Out[150…

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.583
Model:	OLS	**Adj. R-squared:**	0.583
Method:	Least Squares	**F-statistic:**	1373.
Date:	Mon, 09 Aug 2021	**Prob (F-statistic):**	0.00
Time:	21:10:36	**Log-Likelihood:**	21713.
No. Observations:	10800	**AIC:**	-4.340e+04
Df Residuals:	10788	**BIC:**	-4.331e+04
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0140	0.002	-8.635	0.000	-0.017	-0.011
bedrooms	-0.2084	0.013	-15.772	0.000	-0.234	-0.183
bathrooms	0.0114	0.005	2.227	0.026	0.001	0.022
sqft_lot	0.0062	0.018	0.346	0.729	-0.029	0.041
floors	0.0043	0.002	2.174	0.030	0.000	0.008
waterfront	0.0813	0.004	20.437	0.000	0.074	0.089
view	0.0375	0.002	19.554	0.000	0.034	0.041
condition	0.0257	0.002	12.809	0.000	0.022	0.030
sqft_above	0.3242	0.007	47.669	0.000	0.311	0.338
sqft_basement	0.1405	0.004	36.290	0.000	0.133	0.148
sqft_living15	0.0324	0.004	7.794	0.000	0.024	0.041
sqft_lot15	-0.0964	0.014	-6.853	0.000	-0.124	-0.069

Omnibus:	7614.125	**Durbin-Watson:**	2.031
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	404163.477
Skew:	2.804	**Prob(JB):**	0.00
Kurtosis:	32.440	**Cond. No.**	85.5

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

In [151...
```
vif = pd.DataFrame()
vif['Features'] = X2.columns
vif['VIF'] = [variance_inflation_factor(X2.values, i) for i in range(X2.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by ="VIF", ascending = False)

vif
```

Out[151...

	Features	VIF
7	sqft_above	15.36
1	bathrooms	15.07
9	sqft_living15	13.74
0	bedrooms	10.58
6	condition	6.15
3	floors	3.31
10	sqft_lot15	2.70
2	sqft_lot	2.53
8	sqft_basement	2.49
5	view	1.48
4	waterfront	1.20

In [152...
```
X3 = X2.drop('sqft_above', 1,)
X_train_LM = sm.add_constant(X3)
LR_4 = sm.OLS(y_train, X_train_LM).fit()
LR_4.summary()
```

Out[152...

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.496
Model:	OLS	**Adj. R-squared:**	0.495
Method:	Least Squares	**F-statistic:**	1060.
Date:	Mon, 09 Aug 2021	**Prob (F-statistic):**	0.00
Time:	21:10:52	**Log-Likelihood:**	20680.
No. Observations:	10800	**AIC:**	-4.134e+04
Df Residuals:	10789	**BIC:**	-4.126e+04
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0277	0.002	-15.767	0.000	-0.031	-0.024
bedrooms	-0.0437	0.014	-3.115	0.002	-0.071	-0.016
bathrooms	0.1072	0.005	20.616	0.000	0.097	0.117
sqft_lot	0.0583	0.020	2.959	0.003	0.020	0.097
floors	0.0232	0.002	10.951	0.000	0.019	0.027
waterfront	0.0904	0.004	20.677	0.000	0.082	0.099
view	0.0403	0.002	19.127	0.000	0.036	0.044
condition	0.0262	0.002	11.850	0.000	0.022	0.031
sqft_basement	0.0675	0.004	17.254	0.000	0.060	0.075

sqft_living15	0.1515	0.004	41.463	0.000	0.144	0.159
sqft_lot15	-0.0563	0.015	-3.644	0.000	-0.087	-0.026

Omnibus:	9914.425	**Durbin-Watson:**	2.027
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1104675.617
Skew:	4.016	**Prob(JB):**	0.00
Kurtosis:	51.891	**Cond. No.**	84.8

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

In [153...
```
vif = pd.DataFrame()
vif['Features'] = X3.columns
vif['VIF'] = [variance_inflation_factor(X3.values, i) for i in range(X3.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by ="VIF", ascending = False)

vif
```

Out[153...

	Features	VIF
1	bathrooms	12.99
0	bedrooms	10.04
8	sqft_living15	9.06
6	condition	5.85
3	floors	3.20
9	sqft_lot15	2.69
2	sqft_lot	2.52
7	sqft_basement	2.14
5	view	1.48
4	waterfront	1.20

In [154...
```
X4 = X3.drop('bathrooms', 1,)
X_train_LM = sm.add_constant(X4)
LR_5 = sm.OLS(y_train, X_train_LM).fit()
LR_5.summary()
```

Out[154...

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.476
Model:	OLS	**Adj. R-squared:**	0.475
Method:	Least Squares	**F-statistic:**	1088.
Date:	Mon, 09 Aug 2021	**Prob (F-statistic):**	0.00
Time:	21:10:57	**Log-Likelihood:**	20472.
No. Observations:	10800	**AIC:**	-4.092e+04
Df Residuals:	10790	**BIC:**	-4.085e+04
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0236	0.002	-13.271	0.000	-0.027	-0.020
bedrooms	0.0474	0.014	3.490	0.000	0.021	0.074
sqft_lot	0.0665	0.020	3.311	0.001	0.027	0.106
floors	0.0456	0.002	24.585	0.000	0.042	0.049
waterfront	0.0910	0.004	20.401	0.000	0.082	0.100
view	0.0401	0.002	18.639	0.000	0.036	0.044
condition	0.0243	0.002	10.778	0.000	0.020	0.029
sqft_basement	0.0936	0.004	24.784	0.000	0.086	0.101

sqft_living15	0.1782	0.003	51.171	0.000	0.171	0.185
sqft_lot15	-0.0506	0.016	-3.216	0.001	-0.082	-0.020

Omnibus:	10464.770	**Durbin-Watson:**	2.034
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1374451.917
Skew:	4.352	**Prob(JB):**	0.00
Kurtosis:	57.576	**Cond. No.**	83.4

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

In [155…
```
vif = pd.DataFrame()
vif['Features'] = X4.columns
vif['VIF'] = [variance_inflation_factor(X4.values, i) for i in range(X4.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by ="VIF", ascending = False)

vif
```

Out[155…

	Features	VIF
0	bedrooms	8.78
7	sqft_living15	7.62
5	condition	5.81
8	sqft_lot15	2.69
1	sqft_lot	2.52
2	floors	2.30
6	sqft_basement	1.93
4	view	1.48
3	waterfront	1.20

In [156…
```
X5 = X4.drop('bedrooms', 1,)
X_train_LM = sm.add_constant(X5)
LR_6 = sm.OLS(y_train, X_train_LM).fit()
LR_6.summary()
```

Out[156…

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.475
Model:	OLS	**Adj. R-squared:**	0.475
Method:	Least Squares	**F-statistic:**	1221.
Date:	Mon, 09 Aug 2021	**Prob (F-statistic):**	0.00
Time:	21:11:01	**Log-Likelihood:**	20466.
No. Observations:	10800	**AIC:**	-4.091e+04
Df Residuals:	10791	**BIC:**	-4.085e+04
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0218	0.002	-12.803	0.000	-0.025	-0.018
sqft_lot	0.0668	0.020	3.327	0.001	0.027	0.106
floors	0.0467	0.002	25.611	0.000	0.043	0.050
waterfront	0.0905	0.004	20.285	0.000	0.082	0.099
view	0.0395	0.002	18.425	0.000	0.035	0.044
condition	0.0247	0.002	11.004	0.000	0.020	0.029
sqft_basement	0.0974	0.004	26.986	0.000	0.090	0.105
sqft_living15	0.1818	0.003	54.661	0.000	0.175	0.188
sqft_lot15	-0.0518	0.016	-3.288	0.001	-0.083	-0.021

Omnibus:	10480.009	**Durbin-Watson:**	2.035
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1377783.096
Skew:	4.363	**Prob(JB):**	0.00
Kurtosis:	57.641	**Cond. No.**	83.3

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

In [157…
```
vif = pd.DataFrame()
vif['Features'] = X5.columns
vif['VIF'] = [variance_inflation_factor(X5.values, i) for i in range(X5.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by ="VIF", ascending = False)

vif
```

Out[157…

	Features	VIF
6	sqft_living15	6.21
4	condition	4.29
7	sqft_lot15	2.69
0	sqft_lot	2.52
1	floors	2.16
5	sqft_basement	1.78
3	view	1.46
2	waterfront	1.20

In [158…
```
X6 = X5.drop('sqft_living15', 1,)
X_train_LM = sm.add_constant(X6)
LR_7 = sm.OLS(y_train, X_train_LM).fit()
LR_7.summary()
```

Out[158…

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.330
Model:	OLS	**Adj. R-squared:**	0.329
Method:	Least Squares	**F-statistic:**	758.3
Date:	Mon, 09 Aug 2021	**Prob (F-statistic):**	0.00
Time:	21:11:07	**Log-Likelihood:**	19146.
No. Observations:	10800	**AIC:**	-3.828e+04
Df Residuals:	10792	**BIC:**	-3.822e+04
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0193	0.002	11.162	0.000	0.016	0.023
sqft_lot	0.0741	0.023	3.266	0.001	0.030	0.119
floors	0.0786	0.002	40.197	0.000	0.075	0.082
waterfront	0.0850	0.005	16.871	0.000	0.075	0.095
view	0.0640	0.002	26.988	0.000	0.059	0.069
condition	0.0165	0.003	6.509	0.000	0.012	0.021
sqft_basement	0.1438	0.004	36.260	0.000	0.136	0.152
sqft_lot15	0.0497	0.018	2.813	0.005	0.015	0.084

Omnibus:	9602.168	**Durbin-Watson:**	2.039
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	793601.084
Skew:	3.911	**Prob(JB):**	0.00
Kurtosis:	44.260	**Cond. No.**	81.2

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

In [159…
```
vif = pd.DataFrame()
vif['Features'] = X6.columns
vif['VIF'] = [variance_inflation_factor(X6.values, i) for i in range(X6.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by ="VIF", ascending = False)

vif
```

Out[159…

	Features	VIF
6	sqft_lot15	2.65
0	sqft_lot	2.52
4	condition	2.37
5	sqft_basement	1.66
1	floors	1.63
3	view	1.42
2	waterfront	1.20

In [160…
```
print(LR_7.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.330
Model:                            OLS   Adj. R-squared:                  0.329
Method:                 Least Squares   F-statistic:                     758.3
Date:                Mon, 09 Aug 2021   Prob (F-statistic):               0.00
Time:                        21:11:11   Log-Likelihood:                 19146.
No. Observations:               10800   AIC:                        -3.828e+04
Df Residuals:                   10792   BIC:                        -3.822e+04
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.0193      0.002     11.162      0.000       0.016       0.023
sqft_lot         0.0741      0.023      3.266      0.001       0.030       0.119
floors           0.0786      0.002     40.197      0.000       0.075       0.082
waterfront       0.0850      0.005     16.871      0.000       0.075       0.095
view             0.0640      0.002     26.988      0.000       0.059       0.069
condition        0.0165      0.003      6.509      0.000       0.012       0.021
sqft_basement    0.1438      0.004     36.260      0.000       0.136       0.152
sqft_lot15       0.0497      0.018      2.813      0.005       0.015       0.084
==============================================================================
Omnibus:                     9602.168   Durbin-Watson:                   2.039
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           793601.084
Skew:                           3.911   Prob(JB):                         0.00
Kurtosis:                      44.260   Cond. No.                         81.2
==============================================================================

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

Check if error terms are normally distributed

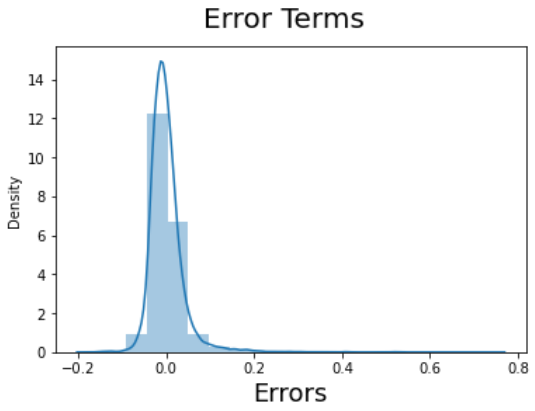
In [162…
```
import seaborn as sns
```

In [163…
```
y_train_price = LR_7.predict(X_train_LM)
```

In [164…
```
fig = plt.figure()
sns.distplot((y_train - y_train_price), bins = 20)
fig.suptitle('Error Terms', fontsize = 20)
plt.xlabel('Errors', fontsize = 18)
```

```
C:\Users\rychu\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and wil
l be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[164…
```
Text(0.5, 0, 'Errors')
```



In [10]:
```
# With a normal distribution of error terms, move to the creation of the final MLR model.
```

Predictions Using Final Model

In [166…
```
num_vars = ['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade', 'sqft
```

In [167…
```
df_test[num_vars] = scaler.transform(df_test[num_vars])
```

```
<ipython-input-167-f3a36cc29a9d>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy
  df_test[num_vars] = scaler.transform(df_test[num_vars])
C:\Users\rychu\anaconda3\lib\site-packages\pandas\core\indexing.py:1637: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy
  self._setitem_single_block(indexer, value, name)
C:\Users\rychu\anaconda3\lib\site-packages\pandas\core\indexing.py:692: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy
  iloc._setitem_with_indexer(indexer, value, self.name)
```

In [168…
```
df_test
```

Out[168…

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	sqft_living15	sqft_lot
4785	0.037873	0.09375	0.066667	0.120729	0.056944	0.2	0.0	0.00	0.50	0.3	0.175885	0.000000	0.342609	0.2158
13514	0.112570	0.09375	0.233333	0.184510	0.010996	0.0	0.0	0.00	1.00	0.6	0.141593	0.278450	0.406957	0.0168
17630	0.020670	0.03125	0.133333	0.052999	0.000565	0.4	0.0	0.00	0.50	0.4	0.077212	0.000000	0.107478	0.0008
20662	0.027634	0.03125	0.233333	0.094153	0.000877	0.4	0.0	0.00	0.75	0.4	0.137168	0.000000	0.259130	0.0015
6511	0.062685	0.09375	0.200000	0.178436	0.005027	0.4	0.0	0.75	0.50	0.5	0.206858	0.116223	0.434783	0.0103
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
13164	0.065967	0.06250	0.400000	0.100987	0.001016	0.4	0.0	0.00	0.50	0.5	0.098451	0.106538	0.156522	0.0008
13676	0.036792	0.06250	0.200000	0.082764	0.002320	0.4	0.0	0.00	0.50	0.4	0.120575	0.000000	0.222609	0.0045
9471	0.055464	0.06250	0.233333	0.082764	0.004052	0.0	0.0	0.00	0.50	0.4	0.120575	0.000000	0.241739	0.0078
5699	0.025468	0.09375	0.166667	0.078208	0.003713	0.2	0.0	0.00	0.75	0.3	0.113938	0.000000	0.114783	0.0091
7446	0.027502	0.06250	0.166667	0.097191	0.008549	0.0	0.0	0.00	0.75	0.4	0.141593	0.000000	0.158261	0.0135

4629 rows × 14 columns

In [169…
```
y_test = df_test.pop('price')
X_test = df_test

X_test_M7 = sm.add_constant(X_test)
X_test_M7 = X_test_M7.drop(['sqft_living', 'grade', 'sqft_above', 'bathrooms', 'bedrooms', 'sqft_living15'], axis =1)

y_pred_M7 = LR_7.predict(X_test_M7)
```

Check R2 values to confirm if final model is best fitted or not.

In [171…
```
from sklearn.metrics import r2_score
```

In [1]:
```
# Train OLS R2 = 0.330
```

In [172…
```
r2_score(y_true = y_test, y_pred = y_pred_M7)
```

Out[172… 0.33375961922542563

In [190…
```
# With R2 values almost equal the model is best fitted!!!
```

Check RMSE

In [191…
```
import sklearn
import math
```

In [192…
```
mse = sklearn.metrics.mean_squared_error(y_test, y_pred_M7)
```

In [193…
```
rmse = math.sqrt(mse)
```

In [194…
```
print(rmse)
```

```
0.03750663037047103
```

Final Model OLS Summary

In [195…
```
LR_7.summary()
```

Out[195…

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.330
Model:	OLS	**Adj. R-squared:**	0.329
Method:	Least Squares	**F-statistic:**	758.3
Date:	Mon, 09 Aug 2021	**Prob (F-statistic):**	0.00
Time:	22:00:53	**Log-Likelihood:**	19146.
No. Observations:	10800	**AIC:**	-3.828e+04
Df Residuals:	10792	**BIC:**	-3.822e+04
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0193	0.002	11.162	0.000	0.016	0.023
sqft_lot	0.0741	0.023	3.266	0.001	0.030	0.119
floors	0.0786	0.002	40.197	0.000	0.075	0.082
waterfront	0.0850	0.005	16.871	0.000	0.075	0.095
view	0.0640	0.002	26.988	0.000	0.059	0.069
condition	0.0165	0.003	6.509	0.000	0.012	0.021
sqft_basement	0.1438	0.004	36.260	0.000	0.136	0.152
sqft_lot15	0.0497	0.018	2.813	0.005	0.015	0.084

Omnibus:	9602.168	**Durbin-Watson:**	2.039
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	793601.084
Skew:	3.911	**Prob(JB):**	0.00
Kurtosis:	44.260	**Cond. No.**	81.2

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

In [ ]: