In [1]:
```
#Import relevant libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#Set style
plt.style.use('seaborn-whitegrid')
```

# Load Datasets

## Dataset 1 -- King County House Prices (2014 - 2015)

### a) Load and Preview the Data

In [2]:
```
#Load dataset into DataFrame and preview
kc = pd.read_csv("Housing_Prices_Modeling/Data/kc_house_data.csv")
kc.head()
```

Out[2]:

| | id | date | price | bedrooms | bathrooms | sqft_living | sqft_lot | floors | waterfro |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 7129300520 | 10/13/2014 | 221900.0 | 3 | 1.00 | 1180 | 5650 | 1.0 | N |
| **1** | 6414100192 | 12/9/2014 | 538000.0 | 3 | 2.25 | 2570 | 7242 | 2.0 | ( |
| **2** | 5631500400 | 2/25/2015 | 180000.0 | 2 | 1.00 | 770 | 10000 | 1.0 | ( |
| **3** | 2487200875 | 12/9/2014 | 604000.0 | 4 | 3.00 | 1960 | 5000 | 1.0 | ( |
| **4** | 1954400510 | 2/18/2015 | 510000.0 | 3 | 2.00 | 1680 | 8080 | 1.0 | ( |

5 rows × 21 columns

In [3]:
```
#Check DataFrame summaries
kc.info()
kc.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 21 columns):
id               21597 non-null int64
date             21597 non-null object
price            21597 non-null float64
bedrooms         21597 non-null int64
bathrooms        21597 non-null float64
sqft_living      21597 non-null int64
sqft_lot         21597 non-null int64
floors           21597 non-null float64
waterfront       19221 non-null float64
view             21534 non-null float64
condition        21597 non-null int64
grade            21597 non-null int64
sqft_above       21597 non-null int64
sqft_basement    21597 non-null object
yr_built         21597 non-null int64
yr_renovated     17755 non-null float64
zipcode          21597 non-null int64
lat              21597 non-null float64
long             21597 non-null float64
sqft_living15    21597 non-null int64
sqft_lot15       21597 non-null int64
dtypes: float64(8), int64(11), object(2)
memory usage: 3.5+ MB
```

Out[3]:

| | id | price | bedrooms | bathrooms | sqft_living | sqft_lot | |
|---|---|---|---|---|---|---|---|
| **count** | 2.159700e+04 | 2.159700e+04 | 21597.000000 | 21597.000000 | 21597.000000 | 2.159700e+04 | 21 |
| **mean** | 4.580474e+09 | 5.402966e+05 | 3.373200 | 2.115826 | 2080.321850 | 1.509941e+04 | |
| **std** | 2.876736e+09 | 3.673681e+05 | 0.926299 | 0.768984 | 918.106125 | 4.141264e+04 | |
| **min** | 1.000102e+06 | 7.800000e+04 | 1.000000 | 0.500000 | 370.000000 | 5.200000e+02 | |
| **25%** | 2.123049e+09 | 3.220000e+05 | 3.000000 | 1.750000 | 1430.000000 | 5.040000e+03 | |
| **50%** | 3.904930e+09 | 4.500000e+05 | 3.000000 | 2.250000 | 1910.000000 | 7.618000e+03 | |
| **75%** | 7.308900e+09 | 6.450000e+05 | 4.000000 | 2.500000 | 2550.000000 | 1.068500e+04 | |
| **max** | 9.900000e+09 | 7.700000e+06 | 33.000000 | 8.000000 | 13540.000000 | 1.651359e+06 | |

### b) Clean Data

In [4]:
```
#Check columns names
print(kc.columns)
```

```
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')
```

In [5]:
```
#Examine sqft_basement values
print(kc.sqft_basement.value_counts())
```

```
0.0       12826
?           454
600.0       217
500.0       209
700.0       208
          ...
1481.0        1
704.0         1
172.0         1
1852.0        1
2500.0        1
Name: sqft_basement, Length: 304, dtype: int64
```

In [6]:
```
#Rows to drop
to_drop = kc.loc[kc['sqft_basement'] == '?'].index

#Drop rows
kc.drop(to_drop, inplace = True)

#Change the Series type to float
kc['sqft_basement'] = kc['sqft_basement'].astype(str).astype(float)

#Check DataFrame
kc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21143 entries, 0 to 21596
Data columns (total 21 columns):
id                 21143 non-null int64
date               21143 non-null object
price              21143 non-null float64
bedrooms           21143 non-null int64
bathrooms          21143 non-null float64
sqft_living        21143 non-null int64
sqft_lot           21143 non-null int64
floors             21143 non-null float64
waterfront         18804 non-null float64
view               21082 non-null float64
condition          21143 non-null int64
grade              21143 non-null int64
sqft_above         21143 non-null int64
sqft_basement      21143 non-null float64
yr_built           21143 non-null int64
yr_renovated       17389 non-null float64
zipcode            21143 non-null int64
lat                21143 non-null float64
long               21143 non-null float64
sqft_living15      21143 non-null int64
sqft_lot15         21143 non-null int64
dtypes: float64(9), int64(11), object(1)
memory usage: 3.5+ MB
```

In [7]:
```
#Check for duplicate rows
duplicate_rows = kc[kc.duplicated()]
print(duplicate_rows.shape)
```

```
(0, 21)
```

In [8]:
```
#Check for null values
print(kc.isna().sum())
```

```
id                  0
date                0
price               0
bedrooms            0
bathrooms           0
sqft_living         0
sqft_lot            0
floors              0
waterfront       2339
view               61
condition           0
grade               0
sqft_above          0
sqft_basement       0
yr_built            0
yr_renovated     3754
zipcode             0
lat                 0
long                0
sqft_living15       0
sqft_lot15          0
dtype: int64
```

In [9]:
```
#Examine the values in the columns with null values
print('View of the Waterfront values: ', kc['waterfront'].value_counts())
print('Year Renovated values: ', kc['yr_renovated'].value_counts())
print('View values: ', kc['view'].value_counts())
```

```
View of the Waterfront values:  0.0    18662
1.0      142
Name: waterfront, dtype: int64
Year Renovated values:  0.0       16666
2014.0       69
2003.0       31
2013.0       31
2007.0       30
          ...
1953.0        1
1944.0        1
1934.0        1
1971.0        1
1959.0        1
Name: yr_renovated, Length: 69, dtype: int64
View values:  0.0    19018
2.0      930
3.0      496
1.0      327
4.0      311
Name: view, dtype: int64
```

In [10]:
```
#Drop the unnecessary and/or uninteresting columns / columns with null values
kc.drop(['id', 'date', 'view', 'yr_renovated'], axis = 1, inplace = True)

#Check DataFrame
kc.head(2)
```

Out[10]:

| | price | bedrooms | bathrooms | sqft_living | sqft_lot | floors | waterfront | condition | grade | sq |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 221900.0 | 3 | 1.00 | 1180 | 5650 | 1.0 | NaN | 3 | 7 | |
| **1** | 538000.0 | 3 | 2.25 | 2570 | 7242 | 2.0 | 0.0 | 3 | 7 | |

In [11]:
```
#Drop the remaining null values by the row
kc.dropna(inplace = True)

#Re-Check for null values
print(kc.isna().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
zipcode          0
lat              0
long             0
sqft_living15    0
sqft_lot15       0
dtype: int64
```

## Dataset 1 -- King County House Prices (2014 - 2015)

### a) Load and Preview the Data

In [12]:
```
#Load dataset into DataFrame and preview
cpi = pd.read_csv("Housing_Prices_Modeling/Data/Common_Points_of_Interest_for_King_County____common_interest_point.csv")
cpi.head()
```

Out[12]:

| | X | Y | OBJECTID | FEATURE_ID | ESITE | CODE | NAME | ABB_NAME | AD |
|---|---|---|---|---|---|---|---|---|---|
| **0** | -122.286944 | 47.499985 | 1 | 6002948 | 0.0 | 600 | Green River Trail Site - Tukwila | Green River Trail Site - Tukwila | 27 11 69 |
| **1** | -122.305465 | 47.635532 | 2 | 828 | 0.0 | 600 | Interlaken Park | Interlaken Park | In D |
| **2** | -122.211064 | 47.405961 | 3 | 374 | 0.0 | 600 | Garrison Creek Park | Garrison Creek Park | S & |
| **3** | -121.912156 | 47.650466 | 4 | 1891 | 124849.0 | 390 | Carnation Library | Carnation Lib | 4 |
| **4** | -122.295038 | 47.441348 | 5 | 1817 | 401027.0 | 60 | Sea-Tac Office Center | Sea-Tac Office Center | |

In [13]:
```
#Check the DataFrame summary
cpi.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6491 entries, 0 to 6490
Data columns (total 10 columns):
X             6491 non-null float64
Y             6491 non-null float64
OBJECTID      6491 non-null int64
FEATURE_ID    6491 non-null int64
ESITE         6484 non-null float64
CODE          6491 non-null int64
NAME          6491 non-null object
ABB_NAME      6491 non-null object
ADDRESS       6489 non-null object
ZIPCODE       6491 non-null int64
dtypes: float64(3), int64(4), object(3)
memory usage: 507.2+ KB
```

### b) Clean Data

In [14]:
```
#Rename all column names to lowercase
cpi.columns = [x.lower() for x in cpi.columns]

#Check column names
print(cpi.columns)
```

```
Index(['x', 'y', 'objectid', 'feature_id', 'esite', 'code', 'name', 'abb_nam
e',
       'address', 'zipcode'],
      dtype='object')
```

In [15]:
```
#Remove uninteresting columns
cpi.drop(['x', 'y', 'objectid', 'feature_id', 'esite'], axis = 1, inplace = Tr
ue)
```

In [16]:
```
#Check for duplicate rows in the dataset
duplicate_rows = cpi[cpi.duplicated()]
print(duplicate_rows.shape)

#Checkk for null values
print(cpi.isna().sum())
```

```
(1, 5)
code        0
name        0
abb_name    0
address     2
zipcode     0
dtype: int64
```

In [17]:
```
#Drop duplicate rows
cpi.drop_duplicates(inplace = True)

#Drop the 2 rows with null values
cpi.dropna(inplace = True)
```

In [18]:
```
#Re-Check the values
print(cpi.isna().sum())
```

```
code        0
name        0
abb_name    0
address     0
zipcode     0
dtype: int64
```

In [19]:
```
#Check Zipcode values
zipcode = list(cpi.zipcode.value_counts().index)
print(sorted(zipcode))
```

```
[0, 9827, 91855, 98001, 98002, 98003, 98004, 98005, 98006, 98007, 98008, 9800
9, 98010, 98011, 98012, 98014, 98015, 98019, 98020, 98021, 98022, 98023, 9802
4, 98027, 98028, 98029, 98030, 98031, 98032, 98033, 98034, 98035, 98036, 9803
8, 98039, 98040, 98042, 98043, 98045, 98047, 98050, 98051, 98052, 98053, 9805
4, 98055, 98056, 98057, 98058, 98059, 98063, 98065, 98068, 98070, 98072, 9807
4, 98075, 98077, 98083, 98092, 98101, 98102, 98103, 98104, 98105, 98106, 9810
7, 98108, 98109, 98110, 98112, 98115, 98116, 98117, 98118, 98119, 98121, 9812
2, 98124, 98125, 98126, 98127, 98133, 98134, 98136, 98138, 98144, 98145, 9814
6, 98148, 98154, 98155, 98158, 98160, 98164, 98165, 98166, 98168, 98174, 9817
7, 98178, 98185, 98187, 98188, 98191, 98195, 98198, 98199, 98203, 98204, 9822
4, 98288, 98366, 98391]
```

In [20]:
```
#Select rows to drop
rows_to_drop = cpi.loc[(cpi['zipcode'] == 0) | (cpi['zipcode'] == 9827)].index

#Drop the rows
cpi.drop(rows_to_drop, inplace = True)
```

In [21]:
```
#Check DataFrame information
cpi.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6178 entries, 0 to 6490
Data columns (total 5 columns):
code        6178 non-null int64
name        6178 non-null object
abb_name    6178 non-null object
address     6178 non-null object
zipcode     6178 non-null int64
dtypes: int64(2), object(3)
memory usage: 289.6+ KB
```

#### c) Webscrape Dataset metadata

The 'code' Series in the Common Points of Interest (cpi) DataFrame corresponds to points of interest along multiple different domain classes (e.g. Schools or Parks and Recreation).

Check Metadata:
https://www.arcgis.com/sharing/rest/content/items/6f6dfde35681494f92da924faf7ee47c/info/metadata/metadata.xr
format=default&output=html
(https://www.arcgis.com/sharing/rest/content/items/6f6dfde35681494f92da924faf7ee47c/info/metadata/metadata.x
format=default&output=html)

In [22]:
```
import requests
from bs4 import BeautifulSoup
```

In [23]:
```
html_page = requests.get('https://www.arcgis.com/sharing/rest/content/items/6f6dfde35681494f92da924faf7ee47c/info/metadata/metadata.xml?format=default&output=html')
soup = BeautifulSoup(html_page.content, 'html.parser')
container = soup.find('body', class_ = 'bodyText')
```

In [24]:
```
#Get the text from the html source
page_text = container.get_text()

#Clean the text
#Split into a list
page_text = page_text.split('\n')

#Remove empty elements in the list
for item in page_text:
    if item == '':
        page_text.remove(item)
```

In [25]:
```
#Create an empty list to append the relevant reference information on the CODE
Series in the DataFrame
temp_list = []

#Iterate over the page_text list to find the necessary information and append
 to temp_list
for item in page_text:
    if 'Enumerated Domain Value: ' in item and item not in temp_list:
        temp_list.append(item)
    if 'Enumerated Domain Value Definition: ' in item and item not in temp_lis
t:
        temp_list.append(item)

#Clean the entries in the temp_list
#Remove duplicates & create final lists of keys and values to create a referen
ce dictionnary
#Create two empty keys and values lists
keys = []
values = []

#Assemble the keys and values list
for item in temp_list:
    if temp_list.index(item) % 2 == 0:
        item = item.replace('Enumerated Domain Value:', '')
        keys.append(int(item))
    else:
        item = item.replace('Enumerated Domain Value Definition: ', '')
        values.append(item)

#Check the lenghts of both lists match
print(len(keys), len(values))

# Create and assemble the reference dictionnary
ref = {}
for index in range(0, len(keys)):
    ref[keys[index]] = values[index]

#Check the new reference dictionary
ref
```

```
49 49
```

Out[25]:

```
{30: 'Airport',
 60: 'Non-Government Building',
 61: 'Building - City Government',
 62: 'Building - County Government',
 63: 'Building - State Government',
 64: 'Building - Federal Government',
 65: 'Police Station',
 66: 'Fire Station',
 67: 'Medic Units',
 90: 'Community area, Business Center or neighborhood',
 120: 'Cemetery',
 150: 'Chamber of Commerce',
 180: 'City Hall',
 210: 'Department of Motor Vehicles',
 240: 'Entertainment and Sport facility',
 270: 'Fairground',
 300: 'Golf Course',
 330: 'Hospital or Medical Center',
 340: 'Public Health Clinic',
 360: 'Hotel or Motel',
 390: 'Library',
 420: 'Major employment center or large business',
 450: 'Military installation',
 480: 'Museum',
 500: 'Transit center',
 510: 'Other transportation center',
 520: 'Fare outlet - All type',
 530: 'Fare outlet - Limited type',
 540: 'Park and Pool',
 570: 'Park and Ride',
 580: 'Bike Lockers',
 581: 'Electrical outlets',
 600: 'Parks and Recreation',
 630: 'Pier or Terminal',
 660: 'School - Elementary',
 661: 'School - Junior High or Middle',
 662: 'School - High',
 663: 'School - College or University',
 664: 'School - Alternative',
 665: 'School - Other facility',
 666: 'Schools - K thru 12',
 690: 'Shopping center',
 720: 'Winery',
 902: 'Bike shop',
 903: 'Farmers Market',
 904: 'Public Access farm',
 999: 'General reference feature',
 350: 'Food Facility',
 68: 'Public Safety Answering Point (PSAP)'}
```

In [26]:
```
#Bin the reference points of interest into categories with the list of corresp
onding codes
domains_of_interest = {'public_safety' : [65, 66, 67, 68],
                      'government_building' : [61, 62, 63, 64, 150, 180, 450],
                      'schools' : [660, 661, 662, 663, 664, 665, 666],
                      'parks_and_recreation' : [540, 570, 600],
                      'food_and_restaurants' : [720, 903, 350],
                      'shopping_and_entertainment' : [90, 240, 270, 300, 390,
480, 690, 902],
                      'hubs_of_transport' : [30, 500, 510, 520, 530, 630],
                      'medical' : [67, 330, 340],
                      'other' : [60, 120, 210, 360, 580, 581, 630, 904, 994]}
```

In [27]:
```
zipcodes = sorted(list(cpi.zipcode.value_counts().index))
cpi_per_area = {}

for zipcode in zipcodes:
    #Create count dictionnary
    temp_dict = {'public_safety' : 0,
                  'government_building' : 0,
                  'schools' : 0,
                  'parks_and_recreation' : 0,
                  'food_and_restaurants' : 0,
                  'shopping_and_entertainment' : 0,
                  'hubs_of_transport' : 0,
                  'medical' : 0,
                  'other' : 0}

    #Create list of codes occuring in the zipcode
    codes_list = list(cpi.loc[cpi['zipcode'] == zipcode, 'code'].values)

    #Iterate through codes and the domain_of_interest reference list to count
    #the number of points of interest per zipcode
    for code in codes_list:
        for domain in domains_of_interest:
            if code in domains_of_interest[domain]:
                temp_dict[domain] += 1

    #Append the zipcode specific dictionnary to the overall dict
    cpi_per_area[zipcode] = temp_dict
```

In [28]:
```
#Create an empty (full zeros) DataFrame to concat with kc
a = np.zeros(shape =(kc.shape[0],9))
tempdf = pd.DataFrame(a, columns = list(domains_of_interest.keys()))
```

In [29]:
```
#Reset the index of the 1st DataFrame kc
kc.reset_index(inplace = True)
kc.drop('index', axis = 1, inplace = True)
```

In [30]:
```
#Create new, final DataFrame with the new Series of interest
kcdf = pd.concat([kc, tempdf], axis = 1)
kcdf.head()
```

Out[30]:

| | price | bedrooms | bathrooms | sqft_living | sqft_lot | floors | waterfront | condition | grade | s |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 538000.0 | 3 | 2.25 | 2570 | 7242 | 2.0 | 0.0 | 3 | 7 | |
| **1** | 180000.0 | 2 | 1.00 | 770 | 10000 | 1.0 | 0.0 | 3 | 6 | |
| **2** | 604000.0 | 4 | 3.00 | 1960 | 5000 | 1.0 | 0.0 | 5 | 7 | |
| **3** | 510000.0 | 3 | 2.00 | 1680 | 8080 | 1.0 | 0.0 | 3 | 8 | |
| **4** | 1230000.0 | 4 | 4.50 | 5420 | 101930 | 1.0 | 0.0 | 3 | 11 | |

5 rows × 26 columns

In [31]:
```
#Populate the new Series in the King County DataFrame with the reference zipdf
for index in range(len(kcdf)):

    #Check if the zipcode is present in the cpi_per_area reference
    if kcdf.zipcode[index] in cpi_per_area:

        #Access the reference dictionary for the zipcode
        reference_dict = cpi_per_area[kcdf.zipcode[index]]

        #Change the value to 1 if the cpi is present, 0 if not present in the
 area
        for key in reference_dict.keys():
            if reference_dict[key] == 0:
                kcdf.loc[index, key] = 0
            else:
                kcdf.loc[index, key] = 1

    #If not, replace with a null value
    else:
        for key in reference_dict.keys():
            kcdf.loc[index, key] = np.nan
```

In [32]:
```
#Check for null values
kcdf.isna().sum()
```

Out[32]:
```
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
zipcode                        0
lat                            0
long                           0
sqft_living15                  0
sqft_lot15                     0
public_safety                  0
government_building            0
schools                        0
parks_and_recreation           0
food_and_restaurants           0
shopping_and_entertainment     0
hubs_of_transport              0
medical                        0
other                          0
dtype: int64
```

# Explore Finalized DataFrame

In [33]:
```
#Preview the DataFrame
kcdf.head()
```

Out[33]:

| | price | bedrooms | bathrooms | sqft_living | sqft_lot | floors | waterfront | condition | grade | s |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 538000.0 | 3 | 2.25 | 2570 | 7242 | 2.0 | 0.0 | 3 | 7 | |
| **1** | 180000.0 | 2 | 1.00 | 770 | 10000 | 1.0 | 0.0 | 3 | 6 | |
| **2** | 604000.0 | 4 | 3.00 | 1960 | 5000 | 1.0 | 0.0 | 5 | 7 | |
| **3** | 510000.0 | 3 | 2.00 | 1680 | 8080 | 1.0 | 0.0 | 3 | 8 | |
| **4** | 1230000.0 | 4 | 4.50 | 5420 | 101930 | 1.0 | 0.0 | 3 | 11 | |

5 rows × 26 columns

In [34]:
```
#Check the Summaries
kcdf.info()
kcdf.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18804 entries, 0 to 18803
Data columns (total 26 columns):
price                       18804 non-null float64
bedrooms                    18804 non-null int64
bathrooms                   18804 non-null float64
sqft_living                 18804 non-null int64
sqft_lot                    18804 non-null int64
floors                      18804 non-null float64
waterfront                  18804 non-null float64
condition                   18804 non-null int64
grade                       18804 non-null int64
sqft_above                  18804 non-null int64
sqft_basement               18804 non-null float64
yr_built                    18804 non-null int64
zipcode                     18804 non-null int64
lat                         18804 non-null float64
long                        18804 non-null float64
sqft_living15               18804 non-null int64
sqft_lot15                  18804 non-null int64
public_safety               18804 non-null float64
government_building         18804 non-null float64
schools                     18804 non-null float64
parks_and_recreation        18804 non-null float64
food_and_restaurants        18804 non-null float64
shopping_and_entertainment  18804 non-null float64
hubs_of_transport           18804 non-null float64
medical                     18804 non-null float64
other                       18804 non-null float64
dtypes: float64(16), int64(10)
memory usage: 3.7 MB
```

Out[34]:

| | price | bedrooms | bathrooms | sqft_living | sqft_lot | floors | |
|---|---|---|---|---|---|---|---|
| **count** | 1.880400e+04 | 18804.000000 | 18804.000000 | 18804.000000 | 1.880400e+04 | 18804.000000 | 188 |
| **mean** | 5.418399e+05 | 3.374388 | 2.117541 | 2083.155499 | 1.509805e+04 | 1.494522 | |
| **std** | 3.730331e+05 | 0.927297 | 0.769623 | 923.070881 | 4.102504e+04 | 0.539777 | |
| **min** | 7.800000e+04 | 1.000000 | 0.500000 | 370.000000 | 5.200000e+02 | 1.000000 | |
| **25%** | 3.215000e+05 | 3.000000 | 1.750000 | 1430.000000 | 5.048000e+03 | 1.000000 | |
| **50%** | 4.500000e+05 | 3.000000 | 2.250000 | 1920.000000 | 7.629500e+03 | 1.500000 | |
| **75%** | 6.436125e+05 | 4.000000 | 2.500000 | 2550.000000 | 1.072075e+04 | 2.000000 | |
| **max** | 7.700000e+06 | 33.000000 | 8.000000 | 13540.000000 | 1.651359e+06 | 3.500000 | |

8 rows × 26 columns

In [35]:
```
continuous = ['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'sq
ft_above', 'sqft_basement', 'lat', 'long',
             'sqft_living15', 'sqft_lot15']
categorical = ['floors', 'waterfront', 'condition', 'grade', 'yr_built', 'publ
ic_safety', 'government_building',
              'food_and_restaurants', 'shopping_and_entertainment', 'hubs_of_t
ransport', 'medical', 'other']
```

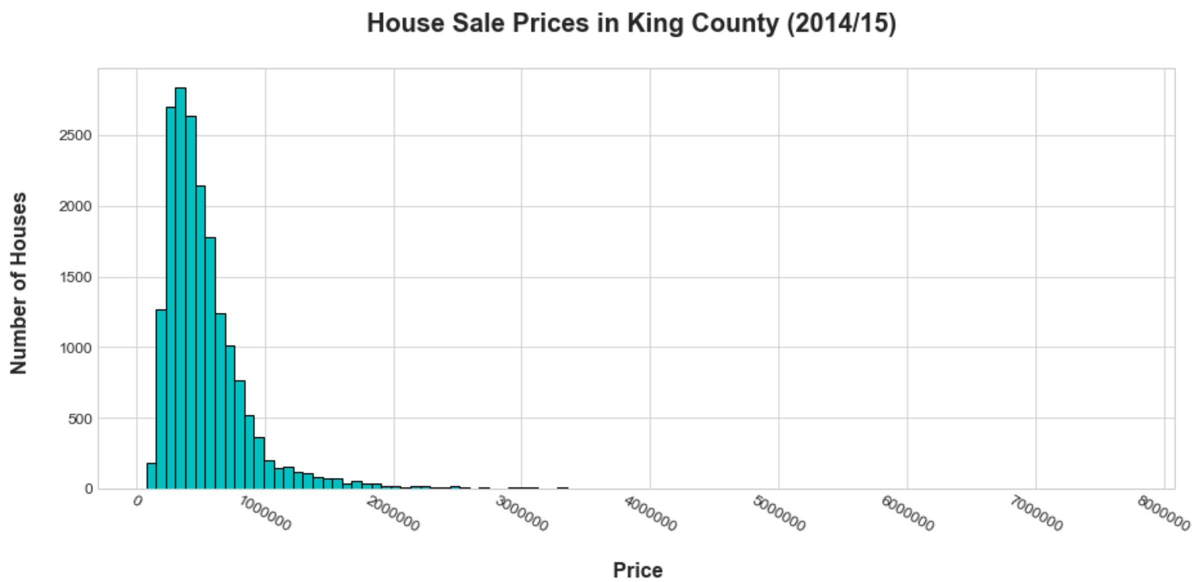
In [36]:
```
#Normalize/standardize the data
for column in continuous:
    if column != 'price':
        kcdf[column] = (kcdf[column] - kcdf[column].mean()) / kcdf[column].std
()
```

## Exploratory Data Analysis & Visualization

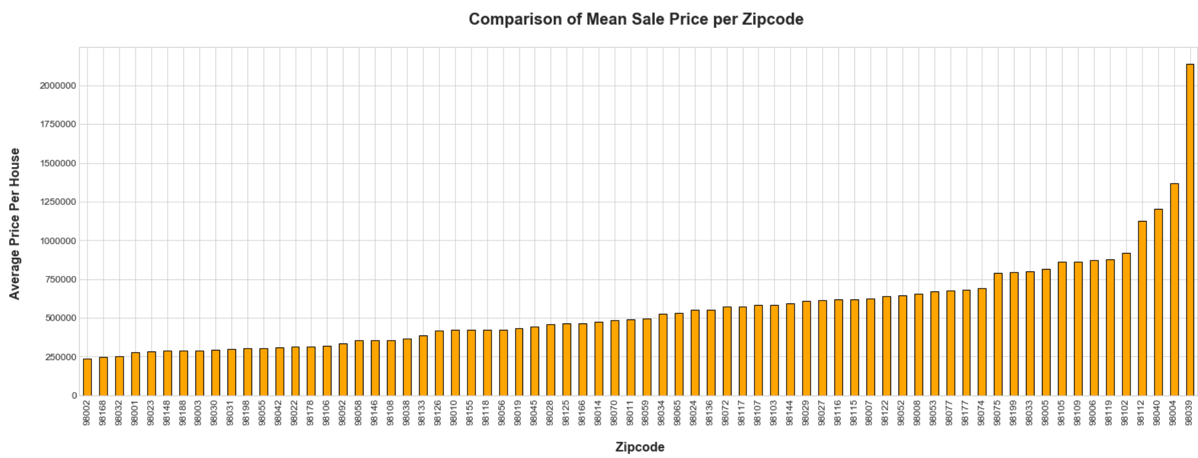
In [37]:
```
kcdf.hist(figsize = (20, 20), color = 'mediumaquamarine', edgecolor = 'black'
);
```


## Zooming in on House Sale Prices

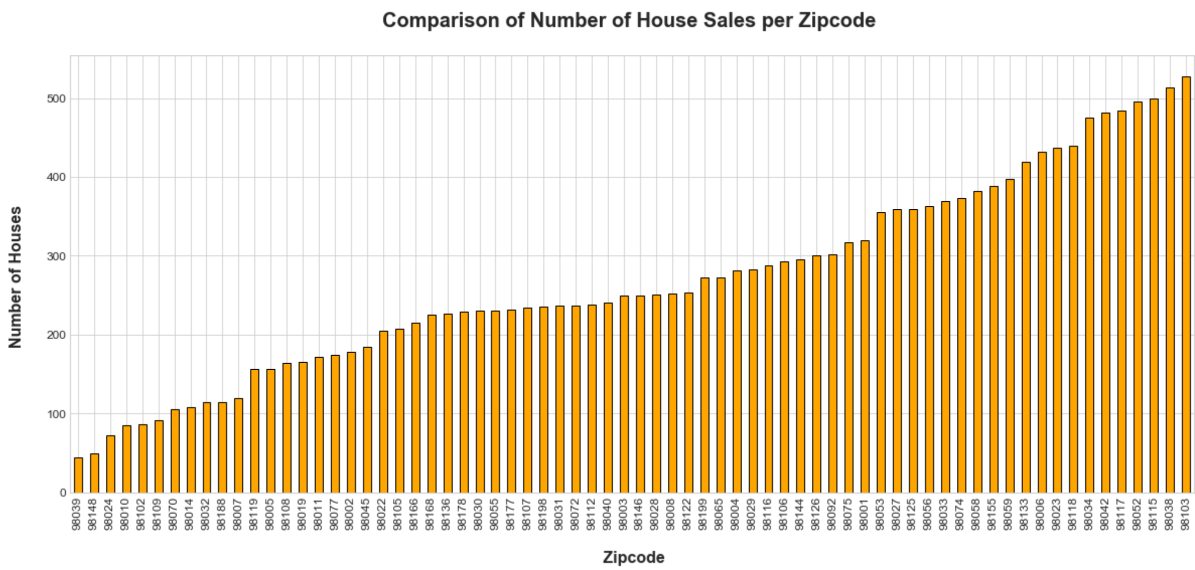
In [38]:
```
#Visualize house sales in King County in the dataset
plt.figure(figsize = (15, 6))
kc.price.plot.hist(bins = 100, color = 'c',  edgecolor = 'black')

#Format the x and y axis
plt.xticks(fontsize = 12, rotation = -30)
plt.yticks(fontsize = 12)

#Format the titles
plt.title('House Sale Prices in King County (2014/15) \n', fontsize = 20, font
weight = 'bold')
plt.xlabel('\n Price', fontsize = 16, fontweight = 'bold')
plt.xticks()
plt.ylabel('Number of Houses \n', fontsize = 16, fontweight = 'bold')


plt.show();
```



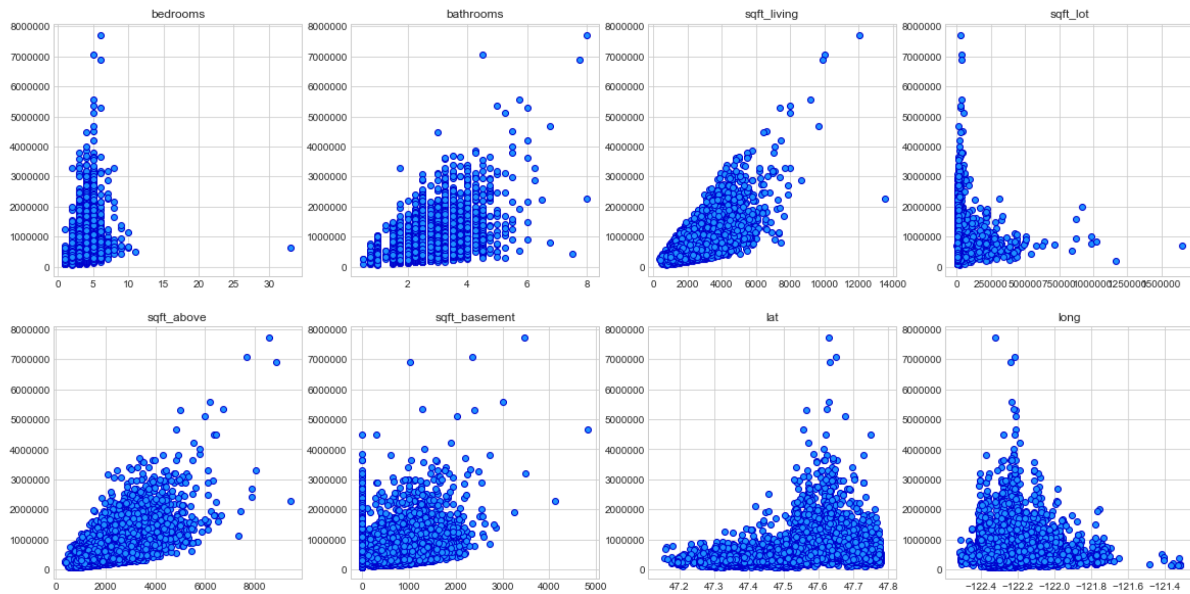
In [82]:
```
#Looking at the Zipcode effect on price
plt.figure(figsize = (25, 8))

#Create plot
kc.groupby('zipcode').mean()['price'].sort_values().plot(kind = 'bar',
                                                          color = 'orange',
                                                          edgecolor = 'black')

#Format plot
plt.title('Comparison of Mean Sale Price per Zipcode \n', fontweight = 'bold',
fontsize = 20)
plt.xlabel('\n Zipcode', fontsize = 16, fontweight = 'bold')
plt.xticks(rotation = 90, fontsize = 12)
plt.ylabel('Average Price Per House \n', fontsize = 16, fontweight = 'bold')
plt.yticks(fontsize = 12)

plt.show();
```



In [41]:
```
#Looking at the Sales numbers by zipcode
plt.figure(figsize = (20, 8))

#Create plot
kc.zipcode.value_counts().sort_values().plot(kind = 'bar', color = 'orange', e
dgecolor = 'black')

#Format plot
plt.title('Comparison of Number of House Sales per Zipcode \n', fontweight =
'bold', fontsize = 20)
plt.xlabel('\n Zipcode', fontsize = 16, fontweight = 'bold')
plt.xticks(rotation = 90, fontsize = 12)
plt.ylabel('Number of Houses \n', fontsize = 16, fontweight = 'bold')
plt.yticks(fontsize = 12)

plt.show();
```



In [42]:
```python
#Visualize the relationships of continuous variables vis a vis price
fig, axes = plt.subplots(nrows = 2, ncols = 4, figsize = (20, 10))

#List of variables to visualize
variables = continuous
variables.remove('price')

for col, ax in zip(continuous, axes.flatten()):
    ax.scatter(kc[col], kc.price, color = 'dodgerblue', edgecolor = 'mediumblue')
    ax.set_title(col)
```




## Prepare Data for Modeling

In [43]:
```
kcdf.nunique()
```

Out[43]:
```
price                       3354
bedrooms                      12
bathrooms                     29
sqft_living                  983
sqft_lot                    8935
floors                         6
waterfront                     2
condition                      5
grade                         11
sqft_above                   896
sqft_basement                299
yr_built                     116
zipcode                       70
lat                         4924
long                         739
sqft_living15                744
sqft_lot15                  7976
public_safety                  2
government_building            2
schools                        1
parks_and_recreation           1
food_and_restaurants           2
shopping_and_entertainment     2
hubs_of_transport              2
medical                        2
other                          2
dtype: int64
```

In [44]:
```
#Drop unsuitable columns
kcdf.drop(['yr_built', 'zipcode', 'schools', 'parks_and_recreation'], axis = 1
, inplace = True)
categorical.remove('yr_built')
```

In [45]:
```
#Transform categorical variables
for col in categorical:
    kcdf[col] = kcdf[col].astype('category')
    kcdf[col].cat.codes
```

In [46]:
```
#Create dummy variables for categorical data
dummy = pd.get_dummies(kcdf[categorical], drop_first = True)
```

In [47]:
```
#Remove original columns from dataset and add the dummy columns
kcdf.drop(categorical, axis = 1, inplace = True)

kcdf = pd.concat([kcdf, dummy], axis = 1)
kcdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18804 entries, 0 to 18803
Data columns (total 38 columns):
price                          18804 non-null float64
bedrooms                       18804 non-null float64
bathrooms                      18804 non-null float64
sqft_living                    18804 non-null float64
sqft_lot                       18804 non-null float64
sqft_above                     18804 non-null float64
sqft_basement                  18804 non-null float64
lat                            18804 non-null float64
long                           18804 non-null float64
sqft_living15                  18804 non-null float64
sqft_lot15                     18804 non-null float64
floors_1.5                     18804 non-null uint8
floors_2.0                     18804 non-null uint8
floors_2.5                     18804 non-null uint8
floors_3.0                     18804 non-null uint8
floors_3.5                     18804 non-null uint8
waterfront_1.0                 18804 non-null uint8
condition_2                    18804 non-null uint8
condition_3                    18804 non-null uint8
condition_4                    18804 non-null uint8
condition_5                    18804 non-null uint8
grade_4                        18804 non-null uint8
grade_5                        18804 non-null uint8
grade_6                        18804 non-null uint8
grade_7                        18804 non-null uint8
grade_8                        18804 non-null uint8
grade_9                        18804 non-null uint8
grade_10                       18804 non-null uint8
grade_11                       18804 non-null uint8
grade_12                       18804 non-null uint8
grade_13                       18804 non-null uint8
public_safety_1.0              18804 non-null uint8
government_building_1.0        18804 non-null uint8
food_and_restaurants_1.0       18804 non-null uint8
shopping_and_entertainment_1.0 18804 non-null uint8
hubs_of_transport_1.0          18804 non-null uint8
medical_1.0                    18804 non-null uint8
other_1.0                      18804 non-null uint8
dtypes: float64(11), uint8(27)
memory usage: 2.1 MB
```

In [48]:
```
#Rename Columns
kcdf.columns = kcdf.columns.str.strip().str.replace('_', '').str.replace('0',
'').str.replace('.', '')

kcdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18804 entries, 0 to 18803
Data columns (total 38 columns):
price                        18804 non-null float64
bedrooms                     18804 non-null float64
bathrooms                    18804 non-null float64
sqftliving                   18804 non-null float64
sqftlot                      18804 non-null float64
sqftabove                    18804 non-null float64
sqftbasement                 18804 non-null float64
lat                          18804 non-null float64
long                         18804 non-null float64
sqftliving15                 18804 non-null float64
sqftlot15                    18804 non-null float64
floors15                     18804 non-null uint8
floors2                      18804 non-null uint8
floors25                     18804 non-null uint8
floors3                      18804 non-null uint8
floors35                     18804 non-null uint8
waterfront1                  18804 non-null uint8
condition2                   18804 non-null uint8
condition3                   18804 non-null uint8
condition4                   18804 non-null uint8
condition5                   18804 non-null uint8
grade4                       18804 non-null uint8
grade5                       18804 non-null uint8
grade6                       18804 non-null uint8
grade7                       18804 non-null uint8
grade8                       18804 non-null uint8
grade9                       18804 non-null uint8
grade1                       18804 non-null uint8
grade11                      18804 non-null uint8
grade12                      18804 non-null uint8
grade13                      18804 non-null uint8
publicsafety1                18804 non-null uint8
governmentbuilding1          18804 non-null uint8
foodandrestaurants1          18804 non-null uint8
shoppingandentertainment1    18804 non-null uint8
hubsoftransport1             18804 non-null uint8
medical1                     18804 non-null uint8
other1                       18804 non-null uint8
dtypes: float64(11), uint8(27)
memory usage: 2.1 MB
```

# Modeling

In [49]:
```
from statsmodels.formula.api import ols
import statsmodels.api as sm
import scipy.stats as stats
from sklearn.model_selection import train_test_split

#Create training and testing set
train, test = train_test_split(kcdf)
```

In [50]:
```
#Check the train and testing sets
print(len(train), len(test))
```

```
14103 4701
```

In [51]:
```
#Defining the problem
outcome = 'price'
x_cols = list(kcdf.columns)
x_cols.remove('price')
```

In [52]:
```
#Fitting the model
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model = ols(formula = formula, data = train).fit()
model.summary()
```

Out[52]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | price | **R-squared:** | 0.726 |
| **Model:** | OLS | **Adj. R-squared:** | 0.725 |
| **Method:** | Least Squares | **F-statistic:** | 1036. |
| **Date:** | Tue, 22 Dec 2020 | **Prob (F-statistic):** | 0.00 |
| **Time:** | 12:07:00 | **Log-Likelihood:** | -1.9167e+05 |
| **No. Observations:** | 14103 | **AIC:** | 3.834e+05 |
| **Df Residuals:** | 14066 | **BIC:** | 3.837e+05 |
| **Df Model:** | 36 | | |
| **Covariance Type:** | nonrobust | | |

| | **coef** | **std err** | **t** | **P>\|t\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|
| **Intercept** | 1.565e+06 | 2.03e+05 | 7.729 | 0.000 | 1.17e+06 | 1.96e+06 |
| **bedrooms** | -2.046e+04 | 2242.097 | -9.126 | 0.000 | -2.49e+04 | -1.61e+04 |
| **bathrooms** | 1.919e+04 | 2892.165 | 6.636 | 0.000 | 1.35e+04 | 2.49e+04 |
| **sqftliving** | 7.071e+04 | 1999.173 | 35.368 | 0.000 | 6.68e+04 | 7.46e+04 |
| **sqftlot** | 9520.0360 | 2385.760 | 3.990 | 0.000 | 4843.630 | 1.42e+04 |
| **sqftabove** | 5.758e+04 | 2175.561 | 26.469 | 0.000 | 5.33e+04 | 6.18e+04 |
| **sqftbasement** | 3.915e+04 | 1848.419 | 21.182 | 0.000 | 3.55e+04 | 4.28e+04 |
| **lat** | 8.07e+04 | 1754.975 | 45.981 | 0.000 | 7.73e+04 | 8.41e+04 |
| **long** | -3.129e+04 | 1946.671 | -16.071 | 0.000 | -3.51e+04 | -2.75e+04 |
| **sqftliving15** | 3.01e+04 | 2803.615 | 10.737 | 0.000 | 2.46e+04 | 3.56e+04 |
| **sqftlot15** | -1.349e+04 | 2419.915 | -5.575 | 0.000 | -1.82e+04 | -8746.860 |
| **floors15** | 5.755e+04 | 6230.414 | 9.236 | 0.000 | 4.53e+04 | 6.98e+04 |
| **floors2** | -1.151e+04 | 5196.402 | -2.215 | 0.027 | -2.17e+04 | -1325.757 |
| **floors25** | 1.463e+05 | 1.92e+04 | 7.615 | 0.000 | 1.09e+05 | 1.84e+05 |
| **floors3** | -4697.1980 | 1.11e+04 | -0.422 | 0.673 | -2.65e+04 | 1.71e+04 |
| **floors35** | 41.8800 | 9.7e+04 | 0.000 | 1.000 | -1.9e+05 | 1.9e+05 |
| **waterfront1** | 7.838e+05 | 1.9e+04 | 41.186 | 0.000 | 7.47e+05 | 8.21e+05 |
| **condition2** | 5.396e+04 | 4.96e+04 | 1.088 | 0.277 | -4.32e+04 | 1.51e+05 |
| **condition3** | 5.405e+04 | 4.61e+04 | 1.173 | 0.241 | -3.63e+04 | 1.44e+05 |
| **condition4** | 1.066e+05 | 4.61e+04 | 2.311 | 0.021 | 1.62e+04 | 1.97e+05 |
| **condition5** | 1.546e+05 | 4.64e+04 | 3.333 | 0.001 | 6.37e+04 | 2.45e+05 |
| **grade4** | -1.614e+05 | 1.98e+05 | -0.813 | 0.416 | -5.5e+05 | 2.28e+05 |
| **grade5** | -2.19e+05 | 1.94e+05 | -1.127 | 0.260 | -6e+05 | 1.62e+05 |
| **grade6** | -2.217e+05 | 1.94e+05 | -1.144 | 0.253 | -6.01e+05 | 1.58e+05 |
| **grade7** | -2.091e+05 | 1.94e+05 | -1.079 | 0.281 | -5.89e+05 | 1.71e+05 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **grade8** | -1.647e+05 | 1.94e+05 | -0.850 | 0.396 | -5.45e+05 | 2.15e+05 |
| **grade9** | -4.494e+04 | 1.94e+05 | -0.232 | 0.817 | -4.25e+05 | 3.35e+05 |
| **grade1** | 1.102e+05 | 1.94e+05 | 0.568 | 0.570 | -2.7e+05 | 4.91e+05 |
| **grade11** | 3.453e+05 | 1.94e+05 | 1.775 | 0.076 | -3.59e+04 | 7.27e+05 |
| **grade12** | 7.624e+05 | 1.96e+05 | 3.889 | 0.000 | 3.78e+05 | 1.15e+06 |
| **grade13** | 1.433e+06 | 2.05e+05 | 6.976 | 0.000 | 1.03e+06 | 1.84e+06 |
| **publicsafety1** | 1.299e+04 | 1.17e+04 | 1.112 | 0.266 | -9910.973 | 3.59e+04 |
| **governmentbuilding1** | -1.638e+04 | 4790.196 | -3.420 | 0.001 | -2.58e+04 | -6993.779 |
| **foodandrestaurants1** | -9.8e+05 | 3.45e+04 | -28.390 | 0.000 | -1.05e+06 | -9.12e+05 |
| **shoppingandentertainment1** | -5.447e+04 | 1.63e+04 | -3.347 | 0.001 | -8.64e+04 | -2.26e+04 |
| **hubsoftransport1** | 9.275e+04 | 6429.997 | 14.425 | 0.000 | 8.01e+04 | 1.05e+05 |
| **medical1** | -1.227e+04 | 3846.647 | -3.190 | 0.001 | -1.98e+04 | -4730.965 |
| **other1** | -3394.5891 | 5194.045 | -0.654 | 0.513 | -1.36e+04 | 6786.428 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 9358.198 | **Durbin-Watson:** | 1.995 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 435402.193 |
| **Skew:** | 2.597 | **Prob(JB):** | 0.00 |
| **Kurtosis:** | 29.720 | **Cond. No.** | 1.08e+16 |

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 9.19e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [53]:
```
#The p-values of some of the variables are above the alpha of 0.05, therefore
#these variables should be removed

#Backward elimination
y = kcdf['price']
X = kcdf[x_cols]
pmax = 1
while (len(x_cols) > 0 ):
    p = []
    X_1 = X[x_cols]
    X_1 = sm.add_constant(X_1)
    model = sm.OLS(y, X_1).fit()
    p = pd.Series(model.pvalues.values[1:], index = x_cols)
    pmax = max(p)
    feature_with_p_max = p.idxmax()
    if(pmax > 0.05):
        x_cols.remove(feature_with_p_max)
    else:
        break
selected_features_BE = x_cols
print(selected_features_BE)
```

```
C:\Users\u-ana\Anaconda3\envs\learn-env\lib\site-packages\numpy\core\fromnume
ric.py:2580: FutureWarning: Method .ptp is deprecated and will be removed in
a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)

['bedrooms', 'bathrooms', 'sqftliving', 'sqftlot', 'sqftabove', 'sqftbasemen
t', 'lat', 'long', 'sqftliving15', 'sqftlot15', 'floors15', 'floors2', 'floor
s25', 'waterfront1', 'condition4', 'condition5', 'grade4', 'grade5', 'grade
6', 'grade7', 'grade8', 'grade1', 'grade11', 'grade12', 'grade13', 'governmen
tbuilding1', 'foodandrestaurants1', 'shoppingandentertainment1', 'hubsoftrans
port1', 'medical1']
```

In [54]:
```
#Re-testing the model after removing initial variables
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model = ols(formula = formula, data = train).fit()
model.summary()
```

Out[54]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | price | **R-squared:** | 0.726 |
| **Model:** | OLS | **Adj. R-squared:** | 0.726 |
| **Method:** | Least Squares | **F-statistic:** | 1287. |
| **Date:** | Tue, 22 Dec 2020 | **Prob (F-statistic):** | 0.00 |
| **Time:** | 12:07:01 | **Log-Likelihood:** | -1.9167e+05 |
| **No. Observations:** | 14103 | **AIC:** | 3.834e+05 |
| **Df Residuals:** | 14073 | **BIC:** | 3.836e+05 |
| **Df Model:** | 29 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 1.576e+06 | 3.74e+04 | 42.187 | 0.000 | 1.5e+06 | 1.65e+06 |
| **bedrooms** | -2.042e+04 | 2235.915 | -9.132 | 0.000 | -2.48e+04 | -1.6e+04 |
| **bathrooms** | 1.892e+04 | 2804.867 | 6.745 | 0.000 | 1.34e+04 | 2.44e+04 |
| **sqftliving** | 7.083e+04 | 1995.764 | 35.489 | 0.000 | 6.69e+04 | 7.47e+04 |
| **sqftlot** | 9438.2857 | 2383.759 | 3.959 | 0.000 | 4765.802 | 1.41e+04 |
| **sqftabove** | 5.761e+04 | 2166.574 | 26.592 | 0.000 | 5.34e+04 | 6.19e+04 |
| **sqftbasement** | 3.935e+04 | 1790.234 | 21.980 | 0.000 | 3.58e+04 | 4.29e+04 |
| **lat** | 8.038e+04 | 1737.519 | 46.262 | 0.000 | 7.7e+04 | 8.38e+04 |
| **long** | -3.122e+04 | 1921.655 | -16.248 | 0.000 | -3.5e+04 | -2.75e+04 |
| **sqftliving15** | 3.005e+04 | 2775.993 | 10.826 | 0.000 | 2.46e+04 | 3.55e+04 |
| **sqftlot15** | -1.353e+04 | 2416.348 | -5.600 | 0.000 | -1.83e+04 | -8794.300 |
| **floors15** | 5.782e+04 | 6183.716 | 9.350 | 0.000 | 4.57e+04 | 6.99e+04 |
| **floors2** | -1.096e+04 | 4889.314 | -2.241 | 0.025 | -2.05e+04 | -1374.130 |
| **floors25** | 1.469e+05 | 1.91e+04 | 7.688 | 0.000 | 1.09e+05 | 1.84e+05 |
| **waterfront1** | 7.838e+05 | 1.9e+04 | 41.195 | 0.000 | 7.46e+05 | 8.21e+05 |
| **condition4** | 5.27e+04 | 3963.946 | 13.294 | 0.000 | 4.49e+04 | 6.05e+04 |
| **condition5** | 1.008e+05 | 6330.739 | 15.922 | 0.000 | 8.84e+04 | 1.13e+05 |
| **grade4** | -1.184e+05 | 4.41e+04 | -2.681 | 0.007 | -2.05e+05 | -3.18e+04 |
| **grade5** | -1.755e+05 | 1.77e+04 | -9.936 | 0.000 | -2.1e+05 | -1.41e+05 |
| **grade6** | -1.769e+05 | 9365.695 | -18.891 | 0.000 | -1.95e+05 | -1.59e+05 |
| **grade7** | -1.642e+05 | 6972.468 | -23.543 | 0.000 | -1.78e+05 | -1.5e+05 |
| **grade8** | -1.199e+05 | 6164.766 | -19.450 | 0.000 | -1.32e+05 | -1.08e+05 |
| **grade1** | 1.55e+05 | 8845.966 | 17.521 | 0.000 | 1.38e+05 | 1.72e+05 |
| **grade11** | 3.897e+05 | 1.39e+04 | 28.029 | 0.000 | 3.62e+05 | 4.17e+05 |
| **grade12** | 8.068e+05 | 2.78e+04 | 29.063 | 0.000 | 7.52e+05 | 8.61e+05 |

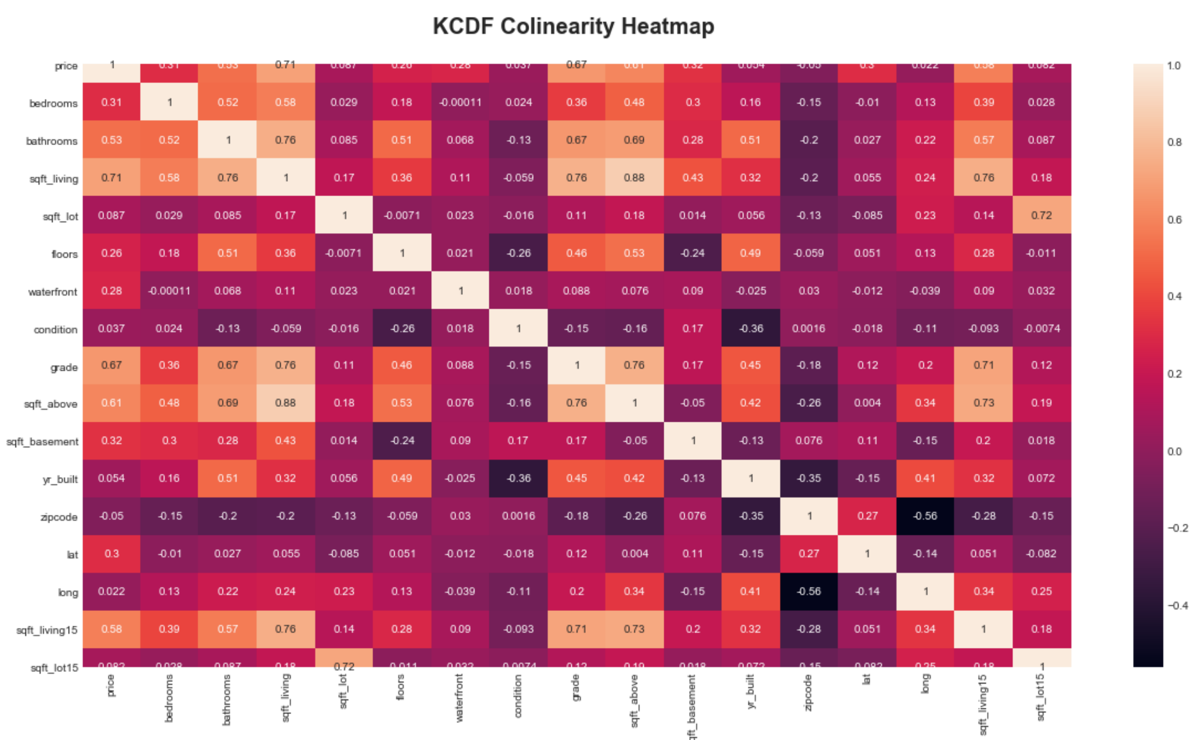
| | | | | | | |
|---|---|---|---|---|---|---|
| **grade13** | 1.479e+06 | 6.66e+04 | 22.194 | 0.000 | 1.35e+06 | 1.61e+06 |
| **governmentbuilding1** | -1.513e+04 | 4650.889 | -3.253 | 0.001 | -2.42e+04 | -6013.598 |
| **foodandrestaurants1** | -9.838e+05 | 3.42e+04 | -28.775 | 0.000 | -1.05e+06 | -9.17e+05 |
| **shoppingandentertainment1** | -4.259e+04 | 1.26e+04 | -3.387 | 0.001 | -6.72e+04 | -1.79e+04 |
| **hubsoftransport1** | 9.13e+04 | 5577.495 | 16.369 | 0.000 | 8.04e+04 | 1.02e+05 |
| **medical1** | -1.182e+04 | 3817.424 | -3.097 | 0.002 | -1.93e+04 | -4338.644 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 9356.496 | **Durbin-Watson:** | 1.995 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 434867.838 |
| **Skew:** | 2.596 | **Prob(JB):** | 0.00 |
| **Kurtosis:** | 29.704 | **Cond. No.** | 1.12e+16 |

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 6.29e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

# Check Multicolinearity Assumption

In [55]:
```
#Checking for multicollinearity
plt.figure(figsize = [20, 10])
sns.heatmap(kc.corr(), annot = True).set_title('KCDF Colinearity Heatmap \n',
fontsize = 20, fontweight = 'bold');
```



In [56]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [57]:
```
#Use the variance inflation factor to check for multicolinearity
X = kcdf[x_cols]
vif = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
list(zip(x_cols, vif))
```

```
C:\Users\u-ana\Anaconda3\envs\learn-env\lib\site-packages\statsmodels\stats\o
utliers_influence.py:193: RuntimeWarning: divide by zero encountered in doubl
e_scalars
  vif = 1. / (1. - r_squared_i)
```

Out[57]:
```
[('bedrooms', 1.7166227832652403),
 ('bathrooms', 2.9985776738318406),
 ('sqftliving', inf),
 ('sqftlot', 2.118808489483488),
 ('sqftabove', inf),
 ('sqftbasement', inf),
 ('lat', 1.1351959926540611),
 ('long', 1.3711379122874103),
 ('sqftliving15', 2.9017866221437223),
 ('sqftlot15', 2.1698167429792505),
 ('floors15', 1.2562869969373414),
 ('floors2', 3.4314395063081924),
 ('floors25', 1.0691784617657754),
 ('waterfront1', 1.0441222124176783),
 ('condition4', 1.5524025500582985),
 ('condition5', 1.1984172908282784),
 ('grade4', 1.0378914843955527),
 ('grade5', 1.274195467114227),
 ('grade6', 3.0903692457906584),
 ('grade7', 7.488973294056128),
 ('grade8', 3.9678980193278246),
 ('grade1', 1.5458974162235335),
 ('grade11', 1.356705416423),
 ('grade12', 1.1753421790812582),
 ('grade13', 1.0740266980649464),
 ('governmentbuilding1', 6.701529175565841),
 ('foodandrestaurants1', 73.66546234409783),
 ('shoppingandentertainment1', 52.57426767791397),
 ('hubsoftransport1', 10.020334789989056),
 ('medical1', 3.882635544418143)]
```

In [58]:
```
#Remove highly colinear columns
cols_to_remove = ['sqftliving', 'sqftabove', 'sqftbasement', 'foodandrestaurants1', 'shoppingandentertainment1',
                 'hubsoftransport1']
for col in cols_to_remove:
    x_cols.remove(col)
print(x_cols)
```

```
['bedrooms', 'bathrooms', 'sqftlot', 'lat', 'long', 'sqftliving15', 'sqftlot15', 'floors15', 'floors2', 'floors25', 'waterfront1', 'condition4', 'condition5', 'grade4', 'grade5', 'grade6', 'grade7', 'grade8', 'grade1', 'grade11', 'grade12', 'grade13', 'governmentbuilding1', 'medical1']
```

In [59]:
```
#Re-testing the model after addressing the colinearity issue
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model = ols(formula = formula, data = train).fit()
model.summary()
```

Out[59]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | price | **R-squared:** | 0.676 |
| **Model:** | OLS | **Adj. R-squared:** | 0.676 |
| **Method:** | Least Squares | **F-statistic:** | 1226. |
| **Date:** | Tue, 22 Dec 2020 | **Prob (F-statistic):** | 0.00 |
| **Time:** | 12:07:03 | **Log-Likelihood:** | -1.9285e+05 |
| **No. Observations:** | 14103 | **AIC:** | 3.857e+05 |
| **Df Residuals:** | 14078 | **BIC:** | 3.859e+05 |
| **Df Model:** | 24 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 6.641e+05 | 7974.708 | 83.276 | 0.000 | 6.48e+05 | 6.8e+05 |
| **bedrooms** | 1.139e+04 | 2232.537 | 5.101 | 0.000 | 7011.098 | 1.58e+04 |
| **bathrooms** | 6.201e+04 | 2801.278 | 22.137 | 0.000 | 5.65e+04 | 6.75e+04 |
| **sqftlot** | 1.411e+04 | 2585.536 | 5.456 | 0.000 | 9039.868 | 1.92e+04 |
| **lat** | 8.703e+04 | 1852.804 | 46.970 | 0.000 | 8.34e+04 | 9.07e+04 |
| **long** | -3.626e+04 | 2016.413 | -17.983 | 0.000 | -4.02e+04 | -3.23e+04 |
| **sqftliving15** | 6.98e+04 | 2771.686 | 25.183 | 0.000 | 6.44e+04 | 7.52e+04 |
| **sqftlot15** | -1.042e+04 | 2623.158 | -3.971 | 0.000 | -1.56e+04 | -5273.654 |
| **floors15** | 6.595e+04 | 6592.522 | 10.003 | 0.000 | 5.3e+04 | 7.89e+04 |
| **floors2** | -1.768e+04 | 4750.334 | -3.722 | 0.000 | -2.7e+04 | -8368.937 |
| **floors25** | 1.677e+05 | 2.05e+04 | 8.160 | 0.000 | 1.27e+05 | 2.08e+05 |
| **waterfront1** | 8.337e+05 | 2.06e+04 | 40.472 | 0.000 | 7.93e+05 | 8.74e+05 |
| **condition4** | 6.828e+04 | 4287.447 | 15.925 | 0.000 | 5.99e+04 | 7.67e+04 |
| **condition5** | 1.234e+05 | 6832.983 | 18.059 | 0.000 | 1.1e+05 | 1.37e+05 |
| **grade4** | -2.344e+05 | 4.78e+04 | -4.901 | 0.000 | -3.28e+05 | -1.41e+05 |
| **grade5** | -2.677e+05 | 1.9e+04 | -14.120 | 0.000 | -3.05e+05 | -2.31e+05 |
| **grade6** | -2.592e+05 | 9853.860 | -26.304 | 0.000 | -2.79e+05 | -2.4e+05 |
| **grade7** | -2.312e+05 | 7249.231 | -31.899 | 0.000 | -2.45e+05 | -2.17e+05 |
| **grade8** | -1.659e+05 | 6526.507 | -25.422 | 0.000 | -1.79e+05 | -1.53e+05 |
| **grade1** | 2.065e+05 | 9454.307 | 21.845 | 0.000 | 1.88e+05 | 2.25e+05 |
| **grade11** | 5.274e+05 | 1.45e+04 | 36.380 | 0.000 | 4.99e+05 | 5.56e+05 |
| **grade12** | 1.057e+06 | 2.93e+04 | 36.022 | 0.000 | 9.99e+05 | 1.11e+06 |
| **grade13** | 1.988e+06 | 7.1e+04 | 27.987 | 0.000 | 1.85e+06 | 2.13e+06 |
| **governmentbuilding1** | -2845.8584 | 5001.200 | -0.569 | 0.569 | -1.26e+04 | 6957.156 |
| **medical1** | -1.4e+04 | 4115.148 | -3.402 | 0.001 | -2.21e+04 | -5932.964 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 10441.973 | **Durbin-Watson:** | 1.983 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 703850.588 |
| **Skew:** | 2.953 | **Prob(JB):** | 0.00 |
| **Kurtosis:** | 37.101 | **Cond. No.** | 67.5 |

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

## Re-Checking the features of the model

In [60]:
```
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE
```

In [61]:
```
#Re-checking for feature elemination with RFE
model = LinearRegression()

#Initializing the RFE model
rfe = RFE(model)

#Transforming data using RFE
X = kcdf[x_cols]
X_rfe = rfe.fit_transform(X, y)

#Fitting the data to the model
model.fit(X_rfe, y)

print(x_cols)
print(rfe.support_)
print(rfe.ranking_)
```

```
['bedrooms', 'bathrooms', 'sqftlot', 'lat', 'long', 'sqftliving15', 'sqftlot1
5', 'floors15', 'floors2', 'floors25', 'waterfront1', 'condition4', 'conditio
n5', 'grade4', 'grade5', 'grade6', 'grade7', 'grade8', 'grade1', 'grade11',
'grade12', 'grade13', 'governmentbuilding1', 'medical1']
[False False False False False False False False False  True  True False
  True  True  True  True  True  True  True  True  True  True False False]
[10  5 11  2  7  6 12  3  8  1  1  4  1  1  1  1  1  1  1  1  1  1 13  9]
```

In [62]:
```
#Remove columns
cols_to_remove = ['bedrooms', 'bathrooms', 'sqftlot', 'lat', 'long', 'sqftlivi
ng', 'sqftlot15', 'floors15', 'floors2',
                  'condition4', 'governmentbuilding1', 'medical1']
for col in cols_to_remove:
    if col in x_cols:
        x_cols.remove(col)

print(x_cols)
```

```
['sqftliving15', 'floors25', 'waterfront1', 'condition5', 'grade4', 'grade5',
'grade6', 'grade7', 'grade8', 'grade1', 'grade11', 'grade12', 'grade13']
```

In [63]:
```
#Re-testing the model
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model = ols(formula = formula, data = train).fit()
model.summary()
```

Out[63]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | price | **R-squared:** | 0.583 |
| **Model:** | OLS | **Adj. R-squared:** | 0.583 |
| **Method:** | Least Squares | **F-statistic:** | 1516. |
| **Date:** | Tue, 22 Dec 2020 | **Prob (F-statistic):** | 0.00 |
| **Time:** | 12:07:04 | **Log-Likelihood:** | -1.9463e+05 |
| **No. Observations:** | 14103 | **AIC:** | 3.893e+05 |
| **Df Residuals:** | 14089 | **BIC:** | 3.894e+05 |
| **Df Model:** | 13 | | |
| **Covariance Type:** | nonrobust | | |

| | **coef** | **std err** | **t** | **P>\|t\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|
| **Intercept** | 7.031e+05 | 6394.776 | 109.946 | 0.000 | 6.91e+05 | 7.16e+05 |
| **sqftliving15** | 6.683e+04 | 2907.223 | 22.987 | 0.000 | 6.11e+04 | 7.25e+04 |
| **floors25** | 2.074e+05 | 2.3e+04 | 9.012 | 0.000 | 1.62e+05 | 2.53e+05 |
| **waterfront1** | 8.567e+05 | 2.32e+04 | 36.877 | 0.000 | 8.11e+05 | 9.02e+05 |
| **condition5** | 1.359e+05 | 7463.585 | 18.211 | 0.000 | 1.21e+05 | 1.51e+05 |
| **grade4** | -4.33e+05 | 5.38e+04 | -8.041 | 0.000 | -5.38e+05 | -3.27e+05 |
| **grade5** | -4.27e+05 | 2.08e+04 | -20.547 | 0.000 | -4.68e+05 | -3.86e+05 |
| **grade6** | -3.675e+05 | 1.02e+04 | -36.155 | 0.000 | -3.87e+05 | -3.48e+05 |
| **grade7** | -2.86e+05 | 7688.398 | -37.202 | 0.000 | -3.01e+05 | -2.71e+05 |
| **grade8** | -1.842e+05 | 7312.056 | -25.195 | 0.000 | -1.99e+05 | -1.7e+05 |
| **grade1** | 2.393e+05 | 1.07e+04 | 22.455 | 0.000 | 2.18e+05 | 2.6e+05 |
| **grade11** | 6.114e+05 | 1.62e+04 | 37.716 | 0.000 | 5.8e+05 | 6.43e+05 |
| **grade12** | 1.165e+06 | 3.3e+04 | 35.330 | 0.000 | 1.1e+06 | 1.23e+06 |
| **grade13** | 2.254e+06 | 8.01e+04 | 28.136 | 0.000 | 2.1e+06 | 2.41e+06 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 9040.504 | **Durbin-Watson:** | 1.980 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 368023.165 |
| **Skew:** | 2.506 | **Prob(JB):** | 0.00 |
| **Kurtosis:** | 27.519 | **Cond. No.** | 45.8 |

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

In [64]:
```
#Let's re-add some features
x_cols.append('lat')
```

In [65]:
```
#Re-testing the model
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model = ols(formula = formula, data = train).fit()
model.summary()
```

Out[65]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | price | **R-squared:** | 0.642 |
| **Model:** | OLS | **Adj. R-squared:** | 0.642 |
| **Method:** | Least Squares | **F-statistic:** | 1806. |
| **Date:** | Tue, 22 Dec 2020 | **Prob (F-statistic):** | 0.00 |
| **Time:** | 12:07:04 | **Log-Likelihood:** | -1.9356e+05 |
| **No. Observations:** | 14103 | **AIC:** | 3.871e+05 |
| **Df Residuals:** | 14088 | **BIC:** | 3.873e+05 |
| **Df Model:** | 14 | | |
| **Covariance Type:** | nonrobust | | |

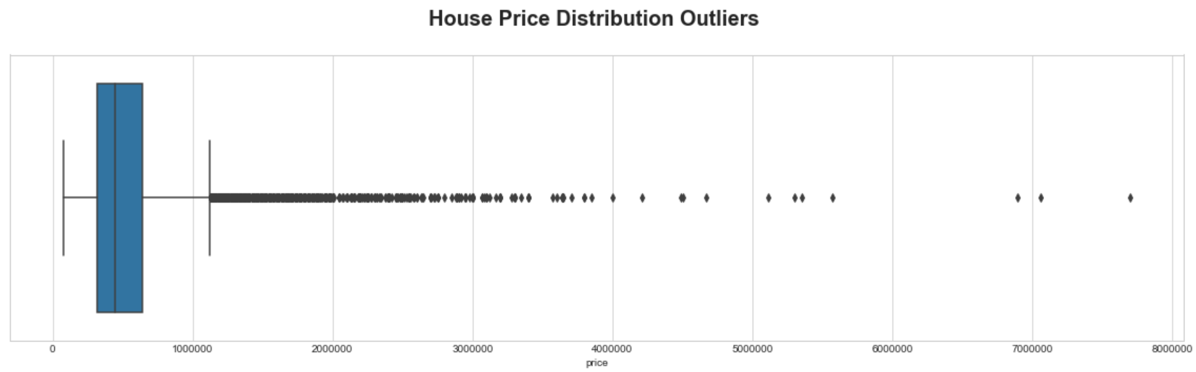
| | **coef** | **std err** | **t** | **P>\|t\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|
| **Intercept** | 6.876e+05 | 5933.212 | 115.898 | 0.000 | 6.76e+05 | 6.99e+05 |
| **sqftliving15** | 7.19e+04 | 2695.517 | 26.676 | 0.000 | 6.66e+04 | 7.72e+04 |
| **floors25** | 2.031e+05 | 2.13e+04 | 9.522 | 0.000 | 1.61e+05 | 2.45e+05 |
| **waterfront1** | 8.794e+05 | 2.15e+04 | 40.846 | 0.000 | 8.37e+05 | 9.22e+05 |
| **condition5** | 1.258e+05 | 6917.966 | 18.189 | 0.000 | 1.12e+05 | 1.39e+05 |
| **grade4** | -3.621e+05 | 4.99e+04 | -7.255 | 0.000 | -4.6e+05 | -2.64e+05 |
| **grade5** | -3.668e+05 | 1.93e+04 | -19.007 | 0.000 | -4.05e+05 | -3.29e+05 |
| **grade6** | -3.261e+05 | 9457.316 | -34.477 | 0.000 | -3.45e+05 | -3.08e+05 |
| **grade7** | -2.619e+05 | 7140.673 | -36.673 | 0.000 | -2.76e+05 | -2.48e+05 |
| **grade8** | -1.73e+05 | 6778.394 | -25.527 | 0.000 | -1.86e+05 | -1.6e+05 |
| **grade1** | 2.267e+05 | 9875.356 | 22.961 | 0.000 | 2.07e+05 | 2.46e+05 |
| **grade11** | 5.891e+05 | 1.5e+04 | 39.203 | 0.000 | 5.6e+05 | 6.19e+05 |
| **grade12** | 1.146e+06 | 3.06e+04 | 37.485 | 0.000 | 1.09e+06 | 1.21e+06 |
| **grade13** | 2.206e+06 | 7.42e+04 | 29.728 | 0.000 | 2.06e+06 | 2.35e+06 |
| **lat** | 9.052e+04 | 1876.897 | 48.229 | 0.000 | 8.68e+04 | 9.42e+04 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 10404.395 | **Durbin-Watson:** | 1.992 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 622079.394 |
| **Skew:** | 2.977 | **Prob(JB):** | 0.00 |
| **Kurtosis:** | 34.987 | **Cond. No.** | 45.8 |

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

# Checking for Outliers

In [66]:
```
#Checking for outliers
plt.figure(figsize = (20, 5))
sns.boxplot(kc['price'])

plt.title('House Price Distribution Outliers \n', fontsize = 20, fontweight = 
'bold');
```




In [67]:
```
#Calculate Summary Statistics to eliminate outliers
price_mean, price_std = np.mean(kc.price), np.std(kc.price)

cut_off = price_std * 3
lower, upper = price_mean - cut_off, price_mean + cut_off
```

In [68]:
```
#identify outliers
outliers = [x for x in kc.price if x < lower or x > upper]

print(len(outliers))
print(sorted(outliers)[:5])
```

```
344
[1670000.0, 1670000.0, 1680000.0, 1680000.0, 1680000.0]
```

In [69]:
```
#Remove rows with outlier values
kcdf = kcdf[kcdf.price < 1670000]
```

In [70]:
```
#Check DataFrame information
kcdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 18460 entries, 0 to 18803
Data columns (total 38 columns):
price                        18460 non-null float64
bedrooms                     18460 non-null float64
bathrooms                    18460 non-null float64
sqftliving                   18460 non-null float64
sqftlot                      18460 non-null float64
sqftabove                    18460 non-null float64
sqftbasement                 18460 non-null float64
lat                          18460 non-null float64
long                         18460 non-null float64
sqftliving15                 18460 non-null float64
sqftlot15                    18460 non-null float64
floors15                     18460 non-null uint8
floors2                      18460 non-null uint8
floors25                     18460 non-null uint8
floors3                      18460 non-null uint8
floors35                     18460 non-null uint8
waterfront1                  18460 non-null uint8
condition2                   18460 non-null uint8
condition3                   18460 non-null uint8
condition4                   18460 non-null uint8
condition5                   18460 non-null uint8
grade4                       18460 non-null uint8
grade5                       18460 non-null uint8
grade6                       18460 non-null uint8
grade7                       18460 non-null uint8
grade8                       18460 non-null uint8
grade9                       18460 non-null uint8
grade1                       18460 non-null uint8
grade11                      18460 non-null uint8
grade12                      18460 non-null uint8
grade13                      18460 non-null uint8
publicsafety1                18460 non-null uint8
governmentbuilding1          18460 non-null uint8
foodandrestaurants1          18460 non-null uint8
shoppingandentertainment1    18460 non-null uint8
hubsoftransport1             18460 non-null uint8
medical1                     18460 non-null uint8
other1                       18460 non-null uint8
dtypes: float64(11), uint8(27)
memory usage: 2.2 MB
```

In [71]:
```
#Re-testing the model
#Create new training and testing set
train, test = train_test_split(kcdf)

#Fit
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model = ols(formula = formula, data = train).fit()
model.summary()
```

```
C:\Users\u-ana\Anaconda3\envs\learn-env\lib\site-packages\statsmodels\base\mo
del.py:1362: RuntimeWarning: invalid value encountered in true_divide
  return self.params / self.bse
```

Out[71]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | price | **R-squared:** | 0.624 |
| **Model:** | OLS | **Adj. R-squared:** | 0.624 |
| **Method:** | Least Squares | **F-statistic:** | 1766. |
| **Date:** | Tue, 22 Dec 2020 | **Prob (F-statistic):** | 0.00 |
| **Time:** | 12:07:04 | **Log-Likelihood:** | -1.8565e+05 |
| **No. Observations:** | 13845 | **AIC:** | 3.713e+05 |
| **Df Residuals:** | 13831 | **BIC:** | 3.714e+05 |
| **Df Model:** | 13 | | |
| **Covariance Type:** | nonrobust | | |

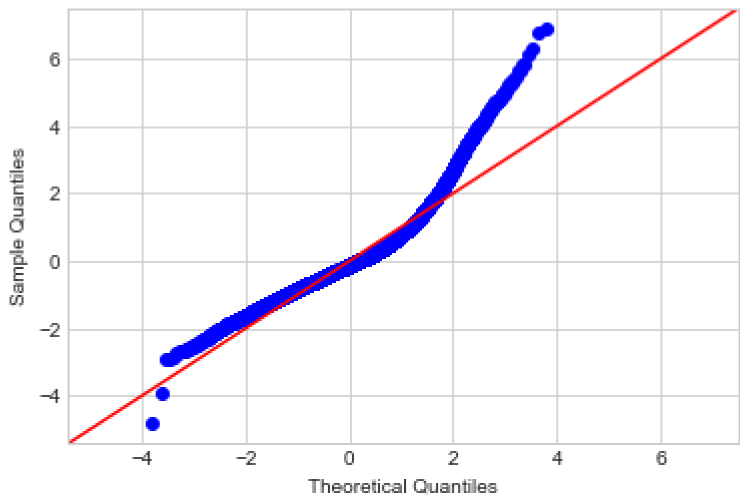
| | **coef** | **std err** | **t** | **P>\|t\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|
| **Intercept** | 6.714e+05 | 4372.187 | 153.563 | 0.000 | 6.63e+05 | 6.8e+05 |
| **sqftliving15** | 6.54e+04 | 2047.908 | 31.934 | 0.000 | 6.14e+04 | 6.94e+04 |
| **floors25** | 1.339e+05 | 1.7e+04 | 7.862 | 0.000 | 1.01e+05 | 1.67e+05 |
| **waterfront1** | 4.09e+05 | 2.2e+04 | 18.567 | 0.000 | 3.66e+05 | 4.52e+05 |
| **condition5** | 1.172e+05 | 5161.453 | 22.705 | 0.000 | 1.07e+05 | 1.27e+05 |
| **grade4** | -3.636e+05 | 3.56e+04 | -10.218 | 0.000 | -4.33e+05 | -2.94e+05 |
| **grade5** | -3.506e+05 | 1.44e+04 | -24.325 | 0.000 | -3.79e+05 | -3.22e+05 |
| **grade6** | -3.116e+05 | 7018.018 | -44.394 | 0.000 | -3.25e+05 | -2.98e+05 |
| **grade7** | -2.474e+05 | 5275.811 | -46.893 | 0.000 | -2.58e+05 | -2.37e+05 |
| **grade8** | -1.518e+05 | 4972.339 | -30.521 | 0.000 | -1.62e+05 | -1.42e+05 |
| **grade1** | 1.36e+05 | 7540.894 | 18.040 | 0.000 | 1.21e+05 | 1.51e+05 |
| **grade11** | 3.093e+05 | 1.26e+04 | 24.614 | 0.000 | 2.85e+05 | 3.34e+05 |
| **grade12** | 4.564e+05 | 3.85e+04 | 11.846 | 0.000 | 3.81e+05 | 5.32e+05 |
| **grade13** | 0 | 0 | nan | nan | 0 | 0 |
| **lat** | 8.614e+04 | 1376.766 | 62.570 | 0.000 | 8.34e+04 | 8.88e+04 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 3746.133 | **Durbin-Watson:** | 2.021 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 13329.345 |
| **Skew:** | 1.336 | **Prob(JB):** | 0.00 |
| **Kurtosis:** | 6.996 | **Cond. No.** | 2.63e+19 |

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.68e-35. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

# Check Normality Assumption

In [72]:
```
#Check for normality assumption with qqplot
residuals = model.resid
fig = sm.graphics.qqplot(residuals, dist = stats.norm, line = '45', fit = True
)
```




In [73]:
```
#The right tail seems to still contain outlier values
#Drop outliers

#Find percentile cutoff point
for i in range(90, 100):
    q = i / 100
    print('{} percentile: {}'.format(q, kcdf['price'].quantile(q=q)))
```

```
0.9 percentile: 850000.0
0.91 percentile: 870000.0
0.92 percentile: 900000.0
0.93 percentile: 930000.0
0.94 percentile: 969229.9999999995
0.95 percentile: 1010000.0
0.96 percentile: 1100000.0
0.97 percentile: 1200000.0
0.98 percentile: 1300000.0
0.99 percentile: 1430000.0
```

In [74]:
```
#Remove rows with outlier values from 98th percentile up
#kcdf = kcdf[kcdf.price < 1300000]
```

In [75]:
```
#Re-testing the model
#Create new training and testing set
train, test = train_test_split(kcdf)

#Fit
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model = ols(formula = formula, data = train).fit()
model.summary()
```

```
C:\Users\u-ana\Anaconda3\envs\learn-env\lib\site-packages\statsmodels\regress
ion\linear_model.py:1827: RuntimeWarning: divide by zero encountered in doubl
e_scalars
  return np.sqrt(eigvals[0]/eigvals[-1])
```

Out[75]:

OLS Regression Results

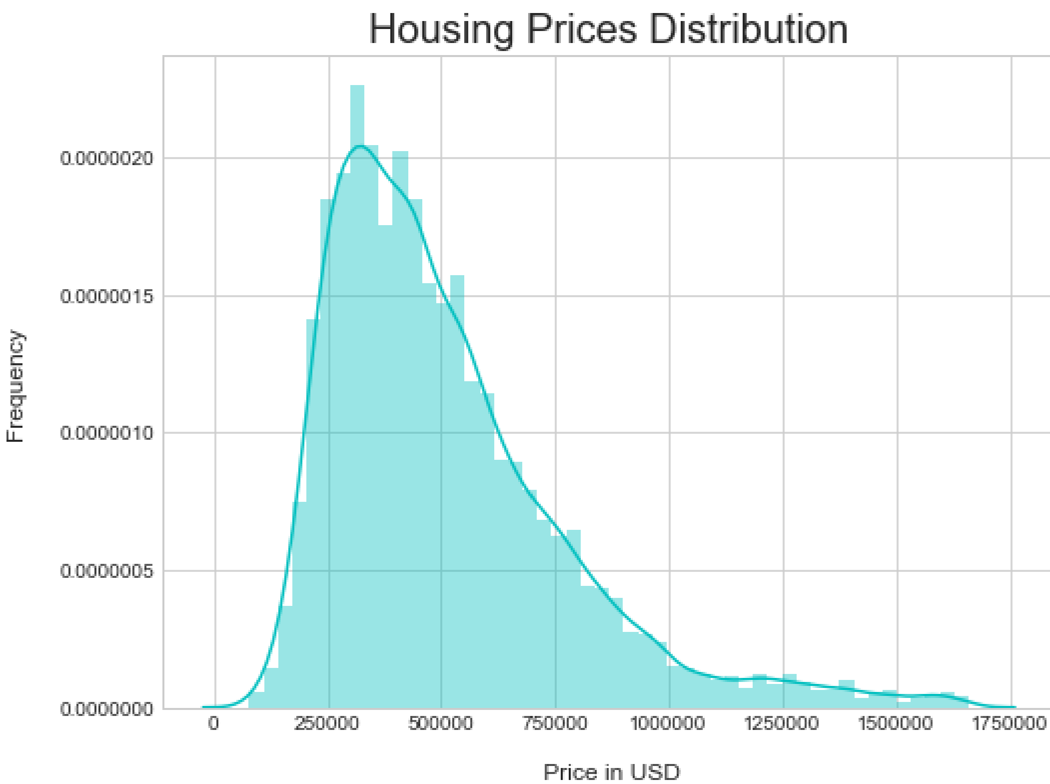
| | | | |
|---|---|---|---|
| **Dep. Variable:** | price | **R-squared:** | 0.627 |
| **Model:** | OLS | **Adj. R-squared:** | 0.627 |
| **Method:** | Least Squares | **F-statistic:** | 1789. |
| **Date:** | Tue, 22 Dec 2020 | **Prob (F-statistic):** | 0.00 |
| **Time:** | 12:07:05 | **Log-Likelihood:** | -1.8555e+05 |
| **No. Observations:** | 13845 | **AIC:** | 3.711e+05 |
| **Df Residuals:** | 13831 | **BIC:** | 3.712e+05 |
| **Df Model:** | 13 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 6.723e+05 | 4346.193 | 154.680 | 0.000 | 6.64e+05 | 6.81e+05 |
| **sqftliving15** | 6.693e+04 | 2026.022 | 33.034 | 0.000 | 6.3e+04 | 7.09e+04 |
| **floors25** | 1.342e+05 | 1.71e+04 | 7.851 | 0.000 | 1.01e+05 | 1.68e+05 |
| **waterfront1** | 4.087e+05 | 2.13e+04 | 19.200 | 0.000 | 3.67e+05 | 4.5e+05 |
| **condition5** | 1.124e+05 | 5103.373 | 22.018 | 0.000 | 1.02e+05 | 1.22e+05 |
| **grade4** | -3.668e+05 | 3.31e+04 | -11.088 | 0.000 | -4.32e+05 | -3.02e+05 |
| **grade5** | -3.409e+05 | 1.39e+04 | -24.519 | 0.000 | -3.68e+05 | -3.14e+05 |
| **grade6** | -3.12e+05 | 6878.522 | -45.357 | 0.000 | -3.25e+05 | -2.99e+05 |
| **grade7** | -2.483e+05 | 5231.052 | -47.458 | 0.000 | -2.59e+05 | -2.38e+05 |
| **grade8** | -1.536e+05 | 4958.853 | -30.973 | 0.000 | -1.63e+05 | -1.44e+05 |
| **grade1** | 1.33e+05 | 7570.333 | 17.573 | 0.000 | 1.18e+05 | 1.48e+05 |
| **grade11** | 2.904e+05 | 1.25e+04 | 23.153 | 0.000 | 2.66e+05 | 3.15e+05 |
| **grade12** | 4.833e+05 | 3.4e+04 | 14.218 | 0.000 | 4.17e+05 | 5.5e+05 |
| **grade13** | 0 | 0 | nan | nan | 0 | 0 |
| **lat** | 8.614e+04 | 1358.564 | 63.403 | 0.000 | 8.35e+04 | 8.88e+04 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 3641.138 | **Durbin-Watson:** | 1.988 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 12682.470 |
| **Skew:** | 1.305 | **Prob(JB):** | 0.00 |
| **Kurtosis:** | 6.896 | **Cond. No.** | inf |

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 0. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [76]:
```
plt.figure(figsize = [8, 6])

sns.distplot(kcdf['price'], color = 'c').set_title('Housing Prices Distributio
n', fontsize = 20)
plt.xlabel('\n Price in USD', fontsize = 12)
plt.ylabel('Frequency \n', fontsize = 12);
```




Removing some rows of data has decreased the r_squared, therefore we revert to the previous model R_2 = .666

In [77]:
```
#The p-values of some of the variables are above the alpha of 0.05, therefore
#we tried removing these variable

#Backward elimination
y = kcdf['price']
X = kcdf[x_cols]
pmax = 1
while (len(x_cols) > 0 ):
    p = []
    X_1 = X[x_cols]
    X_1 = sm.add_constant(X_1)
    model = sm.OLS(y, X_1).fit()
    p = pd.Series(model.pvalues.values[1:], index = x_cols)
    pmax = max(p)
    feature_with_p_max = p.idxmax()
    if(pmax > 0.05):
        x_cols.remove(feature_with_p_max)
    else:
        break
selected_features_BE = x_cols
print(selected_features_BE)
```

```
['sqftliving15', 'floors25', 'waterfront1', 'condition5', 'grade4', 'grade5',
'grade6', 'grade7', 'grade8', 'grade1', 'grade11', 'grade12', 'grade13', 'la
t']

C:\Users\u-ana\Anaconda3\envs\learn-env\lib\site-packages\numpy\core\fromnume
ric.py:2580: FutureWarning: Method .ptp is deprecated and will be removed in
a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

In [78]:
```
#Re-Check model
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model = ols(formula = formula, data = train).fit()
model.summary()
```

Out[78]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | price | **R-squared:** | 0.627 |
| **Model:** | OLS | **Adj. R-squared:** | 0.627 |
| **Method:** | Least Squares | **F-statistic:** | 1789. |
| **Date:** | Tue, 22 Dec 2020 | **Prob (F-statistic):** | 0.00 |
| **Time:** | 12:07:05 | **Log-Likelihood:** | -1.8555e+05 |
| **No. Observations:** | 13845 | **AIC:** | 3.711e+05 |
| **Df Residuals:** | 13831 | **BIC:** | 3.712e+05 |
| **Df Model:** | 13 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 6.723e+05 | 4346.193 | 154.680 | 0.000 | 6.64e+05 | 6.81e+05 |
| **sqftliving15** | 6.693e+04 | 2026.022 | 33.034 | 0.000 | 6.3e+04 | 7.09e+04 |
| **floors25** | 1.342e+05 | 1.71e+04 | 7.851 | 0.000 | 1.01e+05 | 1.68e+05 |
| **waterfront1** | 4.087e+05 | 2.13e+04 | 19.200 | 0.000 | 3.67e+05 | 4.5e+05 |
| **condition5** | 1.124e+05 | 5103.373 | 22.018 | 0.000 | 1.02e+05 | 1.22e+05 |
| **grade4** | -3.668e+05 | 3.31e+04 | -11.088 | 0.000 | -4.32e+05 | -3.02e+05 |
| **grade5** | -3.409e+05 | 1.39e+04 | -24.519 | 0.000 | -3.68e+05 | -3.14e+05 |
| **grade6** | -3.12e+05 | 6878.522 | -45.357 | 0.000 | -3.25e+05 | -2.99e+05 |
| **grade7** | -2.483e+05 | 5231.052 | -47.458 | 0.000 | -2.59e+05 | -2.38e+05 |
| **grade8** | -1.536e+05 | 4958.853 | -30.973 | 0.000 | -1.63e+05 | -1.44e+05 |
| **grade1** | 1.33e+05 | 7570.333 | 17.573 | 0.000 | 1.18e+05 | 1.48e+05 |
| **grade11** | 2.904e+05 | 1.25e+04 | 23.153 | 0.000 | 2.66e+05 | 3.15e+05 |
| **grade12** | 4.833e+05 | 3.4e+04 | 14.218 | 0.000 | 4.17e+05 | 5.5e+05 |
| **grade13** | 0 | 0 | nan | nan | 0 | 0 |
| **lat** | 8.614e+04 | 1358.564 | 63.403 | 0.000 | 8.35e+04 | 8.88e+04 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 3641.138 | **Durbin-Watson:** | 1.988 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 12682.470 |
| **Skew:** | 1.305 | **Prob(JB):** | 0.00 |
| **Kurtosis:** | 6.896 | **Cond. No.** | inf |

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 0. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [79]:
```
x_cols.remove('grade13')
```

In [80]:
```
#Re-Check model
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model = ols(formula = formula, data = train).fit()
model.summary()
```

Out[80]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | price | **R-squared:** | 0.627 |
| **Model:** | OLS | **Adj. R-squared:** | 0.627 |
| **Method:** | Least Squares | **F-statistic:** | 1789. |
| **Date:** | Tue, 22 Dec 2020 | **Prob (F-statistic):** | 0.00 |
| **Time:** | 12:07:05 | **Log-Likelihood:** | -1.8555e+05 |
| **No. Observations:** | 13845 | **AIC:** | 3.711e+05 |
| **Df Residuals:** | 13831 | **BIC:** | 3.712e+05 |
| **Df Model:** | 13 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 6.723e+05 | 4346.193 | 154.680 | 0.000 | 6.64e+05 | 6.81e+05 |
| **sqftliving15** | 6.693e+04 | 2026.022 | 33.034 | 0.000 | 6.3e+04 | 7.09e+04 |
| **floors25** | 1.342e+05 | 1.71e+04 | 7.851 | 0.000 | 1.01e+05 | 1.68e+05 |
| **waterfront1** | 4.087e+05 | 2.13e+04 | 19.200 | 0.000 | 3.67e+05 | 4.5e+05 |
| **condition5** | 1.124e+05 | 5103.373 | 22.018 | 0.000 | 1.02e+05 | 1.22e+05 |
| **grade4** | -3.668e+05 | 3.31e+04 | -11.088 | 0.000 | -4.32e+05 | -3.02e+05 |
| **grade5** | -3.409e+05 | 1.39e+04 | -24.519 | 0.000 | -3.68e+05 | -3.14e+05 |
| **grade6** | -3.12e+05 | 6878.522 | -45.357 | 0.000 | -3.25e+05 | -2.99e+05 |
| **grade7** | -2.483e+05 | 5231.052 | -47.458 | 0.000 | -2.59e+05 | -2.38e+05 |
| **grade8** | -1.536e+05 | 4958.853 | -30.973 | 0.000 | -1.63e+05 | -1.44e+05 |
| **grade1** | 1.33e+05 | 7570.333 | 17.573 | 0.000 | 1.18e+05 | 1.48e+05 |
| **grade11** | 2.904e+05 | 1.25e+04 | 23.153 | 0.000 | 2.66e+05 | 3.15e+05 |
| **grade12** | 4.833e+05 | 3.4e+04 | 14.218 | 0.000 | 4.17e+05 | 5.5e+05 |
| **lat** | 8.614e+04 | 1358.564 | 63.403 | 0.000 | 8.35e+04 | 8.88e+04 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 3641.138 | **Durbin-Watson:** | 1.988 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 12682.470 |
| **Skew:** | 1.305 | **Prob(JB):** | 0.00 |
| **Kurtosis:** | 6.896 | **Cond. No.** | 29.0 |

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

#### The removal of outliers have reduced the R-squared values, therefore we will return to the prior model as the final one at present