Exploratory Analysis & Cleaning

In [73]:

```
#Import required libraries
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
```

In [74]:

```
#Loading the dataset
housing=pd.read_csv('HousingDataSet.csv')
```

In [75]:

```
#To control the display of decimals. Printed dataframe was showing a lot of de
pd.set_option('float_format', '{:.2f}'.format)
```

In [76]:

```
# Print the first 5 rows to understand the dataset
housing.head()
```

Out[76]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors
0	7129300520	20141013T000000	221900.00	3	1.00	1180	5650	1.00
1	6414100192	20141209T000000	538000.00	3	2.25	2570	7242	2.00
2	5631500400	20150225T000000	180000.00	2	1.00	770	10000	1.00
3	2487200875	20141209T000000	604000.00	4	3.00	1960	5000	1.00
4	1954400510	20150218T000000	510000.00	3	2.00	1680	8080	1.00

5 rows × 21 columns

In [77]:

```
#Format the date to compute age of the hous and rennovation age
#Drop columns with date form and keep calculated age and rennovation age
# As previously done by Sharma (2021)
d =[]
for i in housing['date'].values:
    d.append(i[:4])

housing['date'] = d

# convert everything to same datatype
for i in housing.columns:
    housing[i]=housing[i].astype(float)

#make a new column age of the house
housing['age'] = housing['date'] - housing['yr_built']

#calculate the total years of renovation
housing['renov_age'] = np.abs(housing['yr_renovated'] - housing['yr_built'])
housing['renov_age'] = housing.renov_age.apply(lambda x: x if len(str(int(x)))

#remove unwanted columns like yr_built, date, id
housing.drop(['date', 'yr_built', 'yr_renovated'], axis=1, inplace=True)
housing.head()
```

Out[77]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	vie
0	7129300520.00	221900.00	3.00	1.00	1180.00	5650.00	1.00	0.00	0.C
1	6414100192.00	538000.00	3.00	2.25	2570.00	7242.00	2.00	0.00	0.C
2	5631500400.00	180000.00	2.00	1.00	770.00	10000.00	1.00	0.00	0.C
3	2487200875.00	604000.00	4.00	3.00	1960.00	5000.00	1.00	0.00	0.C
4	1954400510.00	510000.00	3.00	2.00	1680.00	8080.00	1.00	0.00	0.C

In [78]:

```
# Check dtypes and null values
housing.info()
# Dataframe contains 21 columns and 21.613 rows
# dtype for price,bathrooms and floors are float64.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21613 non-null  float64
 1   price          21613 non-null  float64
 2   bedrooms       21613 non-null  float64
 3   bathrooms      21613 non-null  float64
 4   sqft_living    21613 non-null  float64
 5   sqft_lot       21613 non-null  float64
 6   floors         21613 non-null  float64
 7   waterfront     21613 non-null  float64
 8   view           21613 non-null  float64
 9   condition      21613 non-null  float64
 10  grade          21613 non-null  float64
 11  sqft_above     21613 non-null  float64
 12  sqft_basement  21613 non-null  float64
 13  zipcode        21613 non-null  float64
 14  lat            21613 non-null  float64
 15  long           21613 non-null  float64
 16  sqft_living15  21613 non-null  float64
 17  sqft_lot15     21613 non-null  float64
 18  age            21613 non-null  float64
 19  renov_age      21613 non-null  float64
dtypes: float64(20)
memory usage: 3.3 MB
```

In [79]:
```
#Converting floats to integers
housing[['price','floors','bathrooms']] = housing[['price','floors','bathrooms
housing.dtypes
```

Out[79]:
```
id               float64
price              int32
bedrooms         float64
bathrooms          int32
sqft_living      float64
sqft_lot         float64
floors             int32
waterfront       float64
view             float64
condition        float64
grade            float64
sqft_above       float64
sqft_basement    float64
zipcode          float64
lat              float64
long             float64
sqft_living15    float64
sqft_lot15       float64
age              float64
renov_age        float64
dtype: object
```

Null Values

In [80]:
```
#Alternative method to confirm that there is not null values in the entire dat
#There is no null values
housing.isnull().values.any()
```

Out[80]:
```
False
```

Find Duplicates

In [81]:
```
# Find duplicates in pandas based on Id Column
duplicated=housing.duplicated(subset=['id'],keep='first')
duplicated.sum() # There are 177 duplicates
```

Out[81]:
```
177
```

In [82]:
```
## Printing duplicated rows
duplicated=housing[housing.duplicated(subset=['id'],keep='first')]
print(duplicated)
```

```
                  id    price  bedrooms  bathrooms  sqft_living  sqft_lot  \
94     6021501535.00   700000      3.00          1      1580.00   5000.00
314    4139480200.00  1400000      4.00          3      4290.00  12103.00
325    7520000520.00   240500      2.00          1      1240.00  12092.00
346    3969300030.00   239900      4.00          1      1000.00   7134.00
372    2231500030.00   530000      4.00          2      2180.00  10754.00
...              ...      ...       ...        ...          ...       ...
20181 7853400250.00   645000      4.00          3      2910.00   5260.00
20613 2724049222.00   220000      2.00          2      1000.00   1092.00
20670 8564860270.00   502000      4.00          2      2680.00   5539.00
20780 6300000226.00   380000      4.00          1      1200.00   2171.00
21581 7853420110.00   625000      3.00          3      2780.00   6000.00

       floors  waterfront  view  condition  grade  sqft_above  sqft_basement
\
94          1        0.00  0.00       3.00   8.00     1290.00         290.00
314         1        0.00  3.00       3.00  11.00     2690.00        1600.00
325         1        0.00  0.00       3.00   6.00      960.00         280.00
346         1        0.00  0.00       3.00   6.00     1000.00           0.00
```

In [83]:
```
# Remove outliers. We are keeping the last ocurrence based on the assumption t
housing.drop_duplicates(subset=['id'],keep='last',inplace=True)
```

In [84]:
```
# Dataset now contains 21.436 rows after removing the duplicates
housing.shape
```

Out[84]:
```
(21436, 20)
```

Outliers

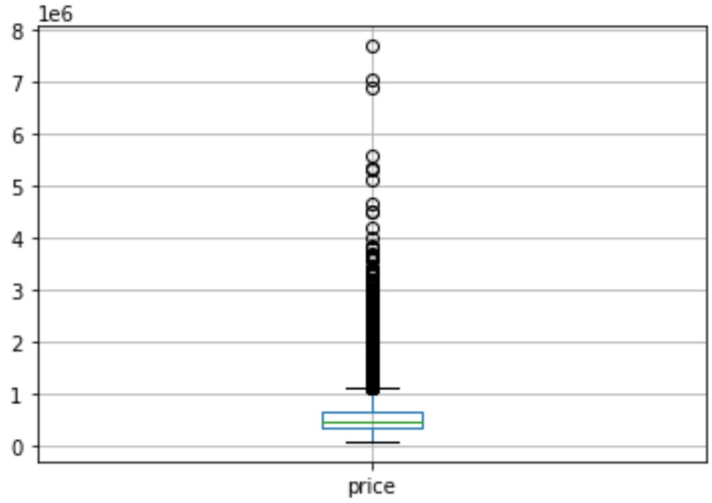
In [85]:
```
# Statistical Analysis
# Price,bedrooms,bathrooms, sqt_living, sqft_lot,sqft_above, sqft_basement var
housing.describe()
```

Out[85]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wate
count	21436.00	21436.00	21436.00	21436.00	21436.00	21436.00	21436.00	214
mean	4580765328.18	541649.96	3.37	1.75	2082.70	15135.64	1.45	
std	2876589633.67	367314.93	0.93	0.73	919.15	41538.62	0.55	
min	1000102.00	75000.00	0.00	0.00	290.00	520.00	1.00	
25%	2123700078.75	324866.00	3.00	1.00	1430.00	5040.00	1.00	
50%	3904921185.00	450000.00	3.00	2.00	1920.00	7614.00	1.00	
75%	7308675062.50	645000.00	4.00	2.00	2550.00	10696.25	2.00	
max	9900000190.00	7700000.00	33.00	8.00	13540.00	1651359.00	3.00	

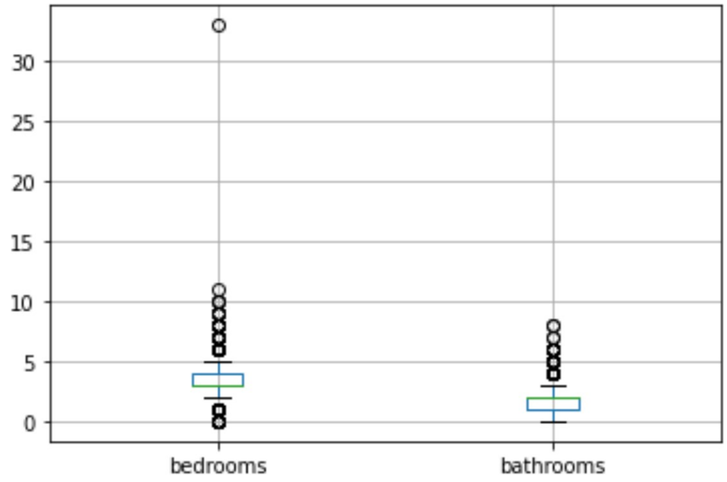
In [86]:
```
# box plot price
housing.boxplot(column='price')
```

Out[86]:
```
<AxesSubplot:>
```



In [87]:
```
# box plots bedrooms and bathrooms
housing.boxplot(column=['bedrooms','bathrooms'])
```

Out[87]:
```
<AxesSubplot:>
```

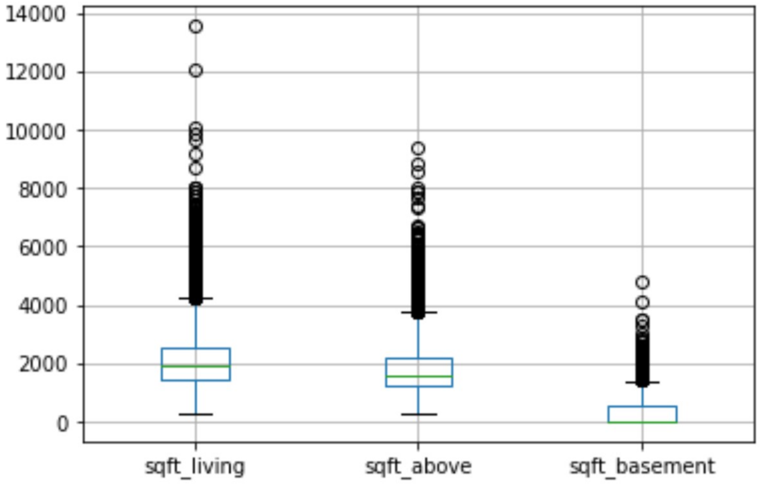


In [88]:

```
# box plots sqft_living, sqft_above and sqft_basement
housing.boxplot(column=['sqft_living','sqft_above','sqft_basement'])
```

Out[88]:

```
<AxesSubplot:>
```



REMOVING OUTLIERS

The following code is to remove outliers of the dataframe using the IQR alternative

[https://www.youtube.com/watch?v=Vc4cXIAa69Y (https://www.youtube.com/watch?v=Vc4cXIAa69Y)](https://www.youtube.com/watch?v=Vc4cXIAa69Y)

In [89]:

```
## define a function called outliers which returns a list of index of outliers
def outliers(df,ft):
    Q1 = df[ft].quantile(0.25)
    Q3 = df[ft].quantile(0.75)
    IQR = Q3 - Q1

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

    ls=df.index[ (df[ft]<lower_bound)|(df[ft]>upper_bound)]

    return ls
```

In [90]:

```
# create an empty list to store the output indices from multiple rows
index_list=[]
for feature in ['price', 'bedrooms','bathrooms','sqft_living','sqft_lot','sqft
    index_list.extend(outliers(housing, feature))
```

In [91]:
```
index_list
```

Out[91]:
```
[5,
 21,
 49,
 69,
 125,
 153,
 216,
 246,
 269,
 270,
 282,
 300,
 312,
 314,
 384,
 419,
 427,
 450,
 472,
```

In [92]:
```
# The number of items in the list, which represents the number of outiers foun
len(index_list)
```

Out[92]:
```
6148
```

In [93]:
```
#define a function called "remove" which returns a cleaned dataframe without o
def remove(df,ls):
    ls=sorted(set(ls))
    df= df.drop(ls)
    return df
```

In [94]:
```
# applying the "remove" function created above.
new_house=remove(housing,index_list)
```

In [95]:
```
new_house.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 17357 entries, 0 to 21612
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             17357 non-null  float64
 1   price          17357 non-null  int32  
 2   bedrooms       17357 non-null  float64
 3   bathrooms      17357 non-null  int32  
 4   sqft_living    17357 non-null  float64
 5   sqft_lot       17357 non-null  float64
 6   floors         17357 non-null  int32  
 7   waterfront     17357 non-null  float64
 8   view           17357 non-null  float64
 9   condition      17357 non-null  float64
 10  grade          17357 non-null  float64
 11  sqft_above     17357 non-null  float64
 12  sqft_basement  17357 non-null  float64
 13  zipcode        17357 non-null  float64
 14  lat            17357 non-null  float64
```

In [96]:
```
new_house.describe()
```

Out[96]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfr
count	17357.00	17357.00	17357.00	17357.00	17357.00	17357.00	17357.00	17357
mean	4755449747.46	460796.25	3.28	1.64	1869.30	7204.29	1.43	0
std	2869137858.73	196947.87	0.78	0.61	660.07	3521.80	0.56	0
min	2800031.00	78000.00	2.00	0.00	440.00	520.00	1.00	0
25%	2321300325.00	308900.00	3.00	1.00	1370.00	4800.00	1.00	0
50%	4068300280.00	425000.00	3.00	2.00	1788.00	7140.00	1.00	0
75%	7507500015.00	577500.00	4.00	2.00	2300.00	9176.00	2.00	0
max	9900000190.00	1125000.00	5.00	3.00	4220.00	19177.00	3.00	1

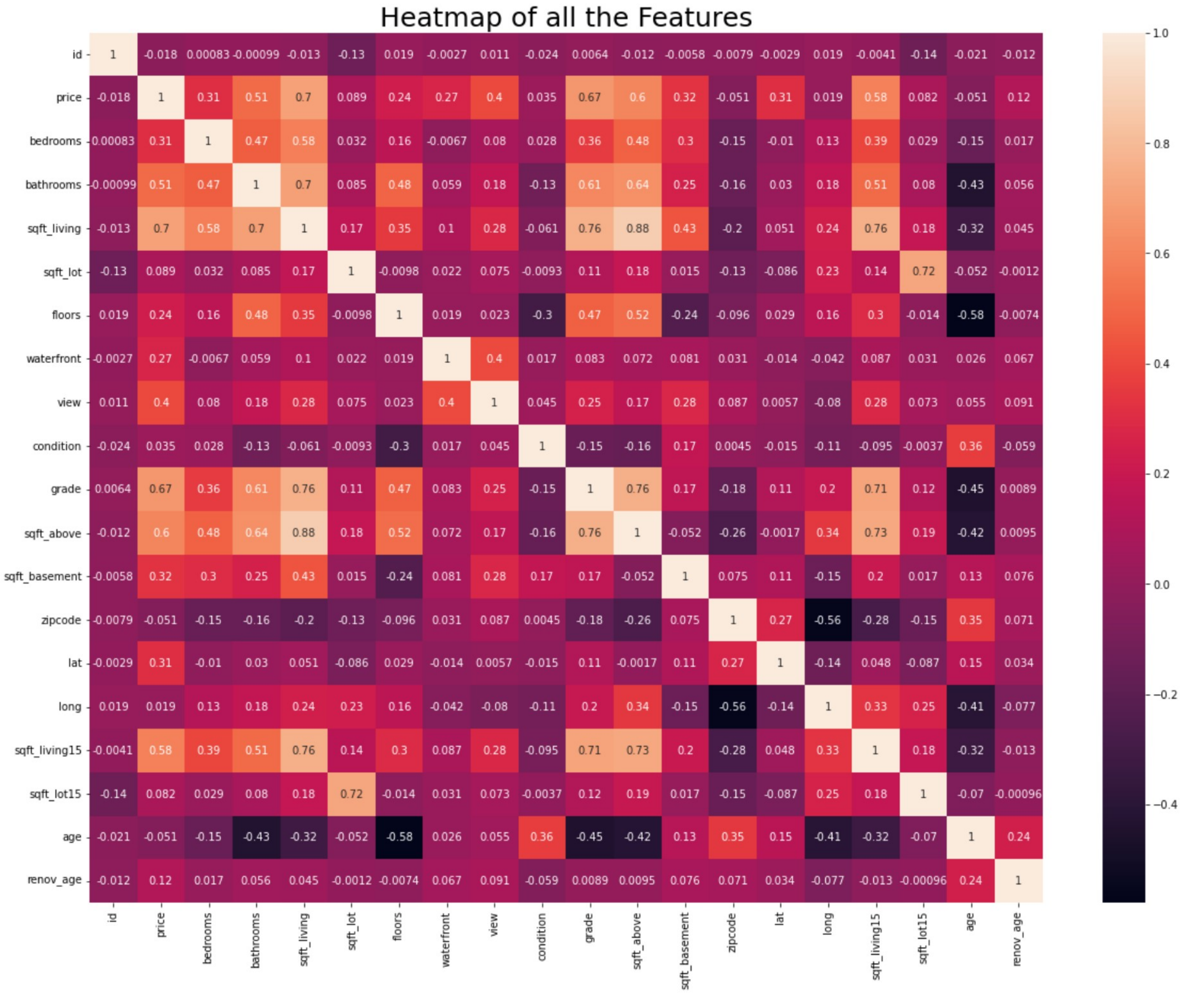
In [97]:
```
#Remove column id since it is not relevant for the following algorithms
#Remove sqft_living, sqft_lot since there is a more updated version sqft_livin
new_house.drop(['id','sqft_living','sqft_lot'], axis=1, inplace=True)
new_house.head()
```

Out[97]:

	price	bedrooms	bathrooms	floors	waterfront	view	condition	grade	sqft_above	sqft_ba
0	221900	3.00	1	1	0.00	0.00	3.00	7.00	1180.00	
1	538000	3.00	2	2	0.00	0.00	3.00	7.00	2170.00	
2	180000	2.00	1	1	0.00	0.00	3.00	6.00	770.00	
3	604000	4.00	3	1	0.00	0.00	5.00	7.00	1050.00	
4	510000	3.00	2	1	0.00	0.00	3.00	8.00	1680.00	

Correlation

In [98]:
```
plt.figure(figsize=(20,15))
sns.heatmap(housing.corr(), annot=True)
plt.title("Heatmap of all the Features", fontsize = 25);
plt.show()
```



In [99]:
```
#Finding out current data set dimensions
new_house.shape
```

Out[99]:
```
(17357, 17)
```

In [100]:

```
#Finding correlation coeficient among variables
corr_features =[]

for i , r in new_house.corr().iterrows():
    k=0
    for j in range(len(r)):
        if i!= r.index[k]:
            if r.values[k] >=0.5:
                corr_features.append([i, r.index[k], r.values[k]])
        k += 1
corr_features
```

Out[100]:

```
[['price', 'grade', 0.5849647065295798],
 ['price', 'sqft_living15', 0.5040975834167322],
 ['bathrooms', 'floors', 0.509518412623005],
 ['bathrooms', 'grade', 0.5088272418632837],
 ['bathrooms', 'sqft_above', 0.5320829257269674],
 ['floors', 'bathrooms', 0.509518412623005],
 ['floors', 'sqft_above', 0.5312115290185615],
 ['grade', 'price', 0.5849647065295798],
 ['grade', 'bathrooms', 0.5088272418632837],
 ['grade', 'sqft_above', 0.6754818964898076],
 ['grade', 'sqft_living15', 0.6404299564673358],
 ['sqft_above', 'bathrooms', 0.5320829257269674],
 ['sqft_above', 'floors', 0.5312115290185615],
 ['sqft_above', 'grade', 0.6754818964898076],
 ['sqft_above', 'sqft_living15', 0.7098650293884988],
 ['sqft_living15', 'price', 0.5040975834167322],
 ['sqft_living15', 'grade', 0.6404299564673358],
 ['sqft_living15', 'sqft_above', 0.7098650293884988]]
```

In [101]:

```
#Removing highly correlated variables with a coeficient above 0.8 and printing
feat =[]
for i in corr_features:
    if i[2] >= 0.8:
        feat.append(i[0])
        feat.append(i[1])

new_house.drop(list(set(feat)), axis=1, inplace=True)
new_house.shape
#Dimensions remain the same beacuse there are zero highly correlated variables
```

Out[101]:

```
(17357, 17)
```

In [103]:

```
# Export cleaned dataset as a csv
new_house.to_csv(r'C:\Users\may93\Downloads\FinalNew_house.csv', index=False)
```

In [ ]: