Final Project Submission

Please fill out:

- Student name:
- Student pace: self paced / part time / full time
- Scheduled project review date/time:
- Instructor name:
- Blog post URL:

ANALYSIS ON EFFECT OF HOUSE FEATURES ON HOUSES PRICES

Research objectives

Main Objective

To determine the influence of house features on home pricing

Specific objectives

To assess the influence of number of floors on house pricing

To evaluate the influence of number of bedrooms on house pricing

To assess the influence of the views on house pricing

Data Understanding

The analysis used data from Kings County which are in the folder Research Data and in csv file format.We used the file 'kc_house_data.csv' for the analysis.

Loading the Dataset

In [4]:

```
# Loading the data set and displaying using pandas
import pandas as pd
data=pd.read_csv("Research Data/kc_house_data.csv")
data.head()
```

Out[4]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wa
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	
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 [5]:

```
data.tail()
```

Out[5]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors
21592	263000018	5/21/2014	360000.0	3	2.50	1530	1131	3.0
21593	6600060120	2/23/2015	400000.0	4	2.50	2310	5813	2.0
21594	1523300141	6/23/2014	402101.0	2	0.75	1020	1350	2.0
21595	291310100	1/16/2015	400000.0	3	2.50	1600	2388	2.0
21596	1523300157	10/15/2014	325000.0	2	0.75	1020	1076	2.0

5 rows × 21 columns

In [6]:

```
# checking data summary
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21597 non-null  int64  
 1   date           21597 non-null  object 
 2   price          21597 non-null  float64
 3   bedrooms       21597 non-null  int64  
 4   bathrooms      21597 non-null  float64
 5   sqft_living    21597 non-null  int64  
 6   sqft_lot       21597 non-null  int64  
 7   floors         21597 non-null  float64
 8   waterfront     19221 non-null  object 
 9   view           21534 non-null  object 
 10  condition      21597 non-null  object 
 11  grade          21597 non-null  object 
 12  sqft_above     21597 non-null  int64  
 13  sqft_basement  21597 non-null  object 
 14  yr_built       21597 non-null  int64  
 15  yr_renovated   17755 non-null  float64
 16  zipcode        21597 non-null  int64  
 17  lat            21597 non-null  float64
 18  long           21597 non-null  float64
 19  sqft_living15  21597 non-null  int64  
 20  sqft_lot15     21597 non-null  int64  
dtypes: float64(6), int64(9), object(6)
memory usage: 3.5+ MB
```

In [7]:

```
data.columns
```

Out[7]:

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

Data Cleaning

In [8]:

```
# checking null values
null=data.isna().sum()
```

In [9]:

```
# percentage of missing data
percentage_missing=null*100/len(data)
percentage_missing
```

Out[9]:

```
id                0.000000
date              0.000000
price             0.000000
bedrooms          0.000000
bathrooms         0.000000
sqft_living       0.000000
sqft_lot          0.000000
floors            0.000000
waterfront       11.001528
view              0.291707
condition         0.000000
grade             0.000000
sqft_above        0.000000
sqft_basement     0.000000
yr_built          0.000000
yr_renovated     17.789508
zipcode           0.000000
lat               0.000000
long              0.000000
sqft_living15     0.000000
sqft_lot15        0.000000
dtype: float64
```

From the results above one of the variables for our analysis 'view' has some missing data of 0.291707%. We will proceed and first clean that.

In [32]:

```
data["view"].unique()
```

Out[32]:

```
array(['NONE', 'GOOD', 'EXCELLENT', 'AVERAGE', 'FAIR'], dtype=object)
```

In [33]:

```
# dealing with missing data on 'view' column
# drop the null values for 'view' since it is a small percentage
data.dropna(axis=0, subset=['view'], inplace=True)
data["view"].isnull().sum()
```

Out[33]:

```
0
```

In [34]:

```
# replace null values in column 'waterfront' with place holder 'unknown'
data['waterfront'].fillna('Unknown', inplace=True)
data["waterfront"].isnull().sum()
```

Out[34]:

```
0
```

In [35]:
```
data["yr_renovated"].unique()
```

Out[35]:
```
array([1991.0, '0', 0.0, 2002.0, 2010.0, 1992.0, 2013.0, 1994.0, 1978.0,
       2005.0, 2003.0, 1984.0, 1954.0, 2014.0, 2011.0, 1983.0, 1990.0,
       1988.0, 1977.0, 1981.0, 1995.0, 2000.0, 1999.0, 1998.0, 1970.0,
       1989.0, 2004.0, 1986.0, 2007.0, 1987.0, 2006.0, 1985.0, 2001.0,
       1980.0, 1971.0, 1945.0, 1979.0, 1997.0, 1950.0, 1969.0, 1948.0,
       2009.0, 2015.0, 2008.0, 2012.0, 1968.0, 1963.0, 1951.0, 1962.0,
       1953.0, 1993.0, 1955.0, 1996.0, 1982.0, 1956.0, 1940.0, 1976.0,
       1946.0, 1975.0, 1964.0, 1973.0, 1957.0, 1959.0, 1960.0, 1965.0,
       1967.0, 1934.0, 1972.0, 1944.0, 1958.0, 1974.0], dtype=object)
```

In [38]:
```
# replace null values in column with place holder'0'
data['yr_renovated'].fillna('0', inplace=True)
data["yr_renovated"].isnull().sum()
```

Out[38]:
```
0
```

In [41]:
```
# checking if all missing data have been cleaned
data.isnull().sum()
```

Out[41]:
```
id               0
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode          0
lat              0
long             0
sqft_living15    0
sqft_lot15       0
dtype: int64
```

We see that all the measing values have been cleaned

Exploratory Data Analysis

In [42]:

```
data.shape
```

Out[42]:

```
(19164, 21)
```

In [43]:

```
data.dtypes
```

Out[43]:

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

In [44]:

```
# data descripyion
data.describe()
```

Out[44]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lo
count	1.916400e+04	1.916400e+04	19164.000000	19164.000000	19164.000000	1.916400e+0
mean	4.594087e+09	5.414490e+05	3.374452	2.117029	2082.038301	1.506174e+0
std	2.876912e+09	3.709009e+05	0.928676	0.769241	921.918226	4.077215e+0
min	1.000102e+06	7.800000e+04	1.000000	0.500000	370.000000	5.200000e+0
25%	2.124077e+09	3.220000e+05	3.000000	1.750000	1430.000000	5.040000e+0
50%	3.905082e+09	4.500000e+05	3.000000	2.250000	1920.000000	7.620000e+0
75%	7.334501e+09	6.439625e+05	4.000000	2.500000	2550.000000	1.072000e+0
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+0

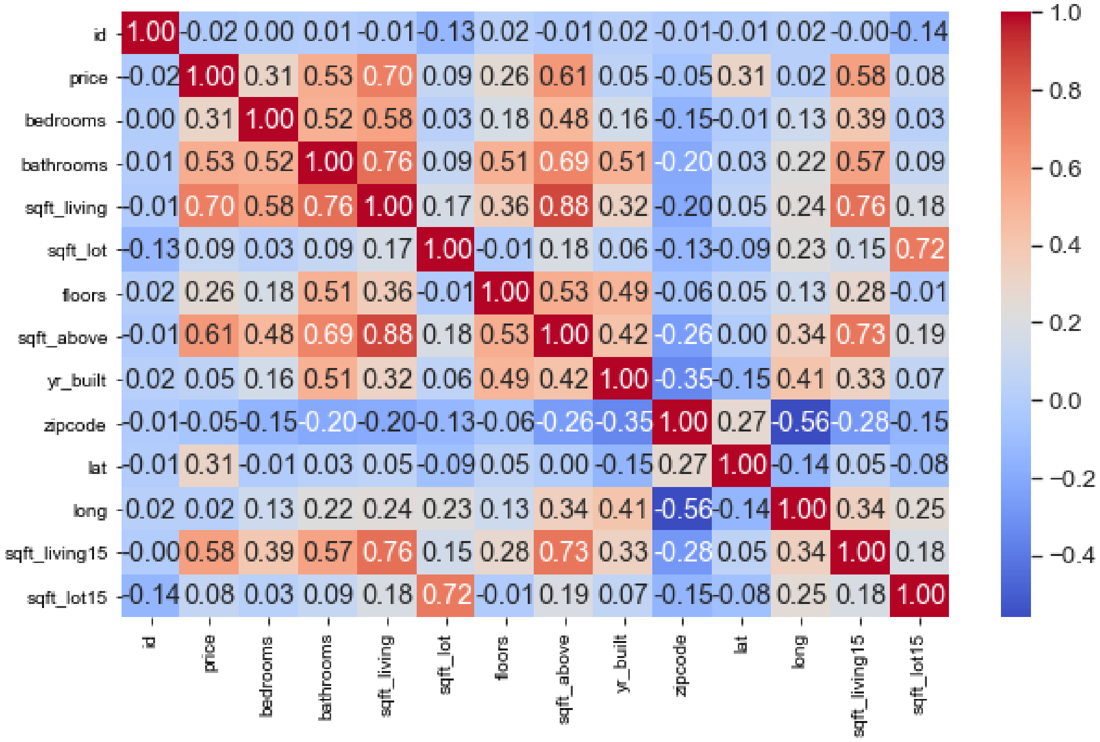
In [48]:

```
# Correlation matrix to see our variable correlations
correlation_matrix = data.corr()
correlation_matrix
```

bedrooms	0.003630	0.309057	1.000000	0.516137	0.577972	0.029685	0.181
bathrooms	0.006942	0.526609	0.516137	1.000000	0.755909	0.085666	0.506
sqft_living	-0.012064	0.704428	0.577972	0.755909	1.000000	0.173624	0.356
sqft_lot	-0.133577	0.087430	0.029685	0.085666	0.173624	1.000000	-0.007
floors	0.018187	0.258797	0.181909	0.506058	0.356938	-0.007519	1.000
sqft_above	-0.011740	0.609611	0.480400	0.687621	0.877669	0.184383	0.525
yr_built	0.023100	0.053433	0.157011	0.507069	0.317123	0.055560	0.490
zipcode	-0.007259	-0.050191	-0.151606	-0.201668	-0.196237	-0.130027	-0.058
lat	-0.005591	0.306372	-0.011111	0.026197	0.054211	-0.085350	0.051
long	0.019871	0.021714	0.131889	0.223675	0.239791	0.229887	0.127
sqft_living15	-0.001334	0.582450	0.392586	0.569443	0.755524	0.146576	0.280
sqft_lot15	-0.138848	0.081562	0.028005	0.086466	0.183177	0.721839	-0.011

In [52]:

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
plt.subplots(figsize=(10,6))
sns.set(font_scale=1.2)
sns.heatmap(correlation_matrix, annot=True, fmt="0.2f", cmap="coolwarm")
plt.show()
```



In [54]:

```
for col1 in correlation_matrix.columns:
    for col2 in correlation_matrix.columns:
        if high_correlation_pairs.loc[col1, col2]:
            correlation_coefficient = correlation_matrix.loc[col1, col2]
            print(f"{col1} and {col2} have a correlation coefficient of {c
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call la
st)
<ipython-input-54-b0d208f97638> in <module>
      1 for value in correlation_matrix:
----> 2     if value>0.70:
      3         print(value)

TypeError: '>' not supported between instances of 'str' and 'float'
```

In [55]:

```
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call la
st)
<ipython-input-55-98bec066f32c> in <module>
      1 for col1 in correlation_matrix.columns:
----> 2     if col1>0.70:
      3         print(col1)

TypeError: '>' not supported between instances of 'str' and 'float'
```

In [47]:

```
# checking the columns for our variables
data['floors']
```

Out[47]:

```
1        2.0
2        1.0
3        1.0
4        1.0
5        1.0
        ... 
21591    2.0
21592    3.0
21593    2.0
21594    2.0
21596    2.0
Name: floors, Length: 19164, dtype: float64
```

In [46]:

```
data['bedrooms']
```

Out[46]:

```
1        3
2        2
3        4
4        3
5        4
        ..
21591    3
21592    3
21593    4
21594    2
21596    2
Name: bedrooms, Length: 19164, dtype: int64
```

In [37]:

```
data['view']
```

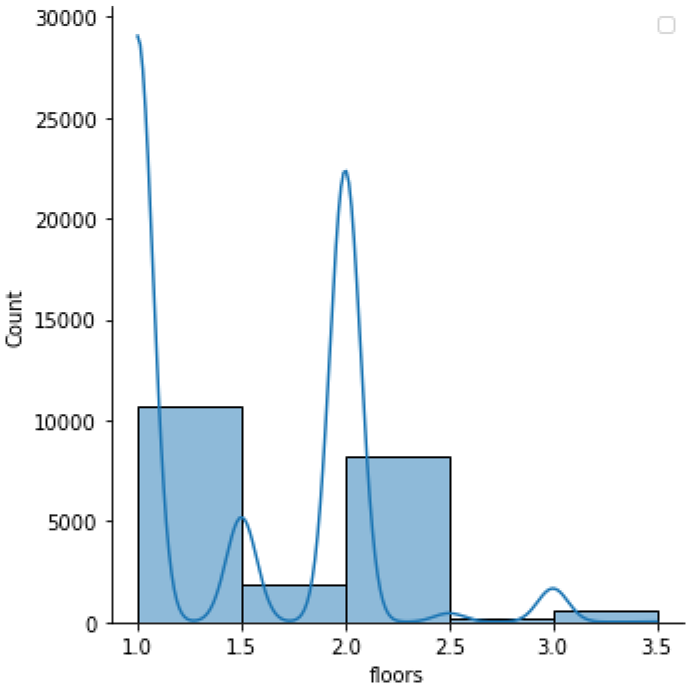
Out[37]:

```
0        NONE
1        NONE
2        NONE
3        NONE
4        NONE
         ...
21592    NONE
21593    NONE
21594    NONE
21595    NONE
21596    NONE
Name: view, Length: 21597, dtype: object
```

In [55]:

```
import seaborn as sns
plt.figure("Test Samples")
sns.displot(data['floors'],bins=5, kde=True);
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.

<Figure size 432x288 with 0 Axes>
```



In [42]:

```
floors=data['floors']
floors.describe()
```

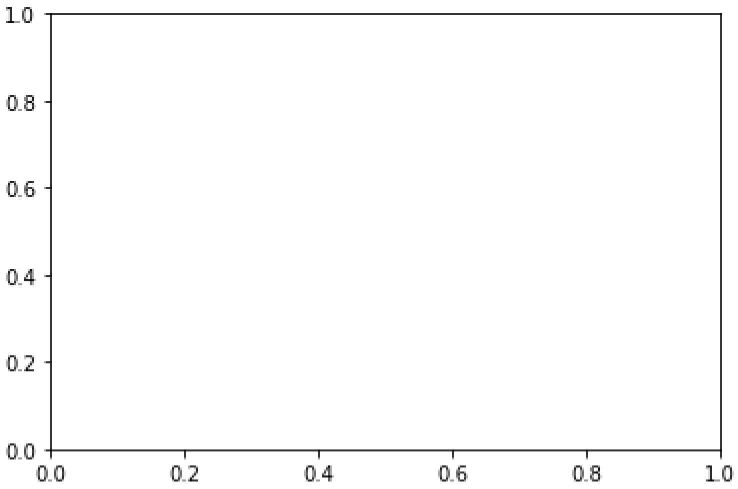
Out[42]:

```
count    21597.000000
mean         1.494096
std          0.539683
min          1.000000
25%          1.000000
50%          1.500000
75%          2.000000
max          3.500000
Name: floors, dtype: float64
```

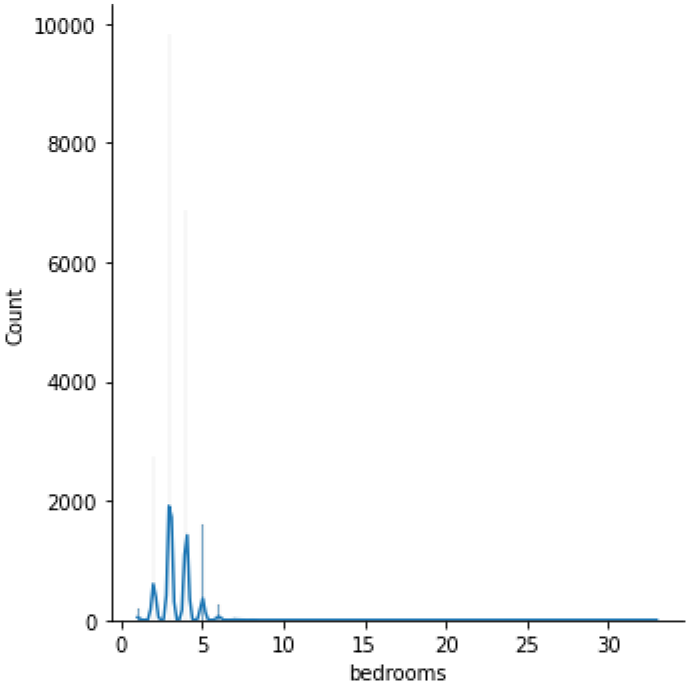
In [46]:

```
#mu = 1.494096
#std = 0.539683
from scipy.stats import norm
import matplotlib.pyplot as plt
mu, std = norm.fit(data['floors'])

# Plot the PDF of the fitted normal distribution
xmin, xmax = plt.xlim()
#x = np.linspace(xmin, xmax, 100)
p = norm.pdf( mu, std)
plt.plot(x, p, 'k', linewidth=2)
plt.show()
```

```
-----------------------------------------------------------------------
---
NameError                                 Traceback (most recent call la
st)
<ipython-input-46-1bb3e9349591> in <module>
      9 #x = np.linspace(xmin, xmax, 100)
     10 p = norm.pdf( mu, std)
---> 11 plt.plot(x, p, 'k', linewidth=2)
     12 plt.show()

NameError: name 'x' is not defined
```



In [53]:
```
sns.displot(data['bedrooms'],bins='auto', kde=True);
```



In [ ]:
```
```