House Price Prediction

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use(['ggplot'])
from sklearn.linear_model import LinearRegression
```

Read the file

In [2]:
```
file_path='C:\\Users\\Dayakar\\Desktop\\DS  Assigments\\internship 27\\archive
house_data=pd.read_csv(file_path)
house_data
```

Out[2]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.(
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.(
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.(
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.(
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.(
...	...	...	...	...	...	...	...	..
21608	263000018	20140521T000000	360000.0	3	2.50	1530	1131	3.(
21609	6600060120	20150223T000000	400000.0	4	2.50	2310	5813	2.(
21610	1523300141	20140623T000000	402101.0	2	0.75	1020	1350	2.(
21611	291310100	20150116T000000	400000.0	3	2.50	1600	2388	2.(
21612	1523300157	20141015T000000	325000.0	2	0.75	1020	1076	2.(

21613 rows × 21 columns

In [3]:
```
house_data.size
```

Out[3]: 453873

In [4]:
```
house_data.shape
```

Out[4]: (21613, 21)

In [5]:
```
house_data.columns
```

Out[5]:
```
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 [6]:
```
house_data.dtypes
```

Out[6]:
```
id               int64
date            object
price          float64
bedrooms         int64
bathrooms      float64
sqft_living      int64
sqft_lot         int64
floors         float64
waterfront       int64
view             int64
condition        int64
grade            int64
sqft_above       int64
sqft_basement    int64
yr_built         int64
yr_renovated     int64
zipcode          int64
lat            float64
long           float64
sqft_living15    int64
sqft_lot15       int64
dtype: object
```

In [7]:
```
house_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21613 non-null  int64  
 1   date           21613 non-null  object 
 2   price          21613 non-null  float64
 3   bedrooms       21613 non-null  int64  
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64  
 6   sqft_lot       21613 non-null  int64  
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64  
 9   view           21613 non-null  int64  
 10  condition      21613 non-null  int64  
 11  grade          21613 non-null  int64  
 12  sqft_above     21613 non-null  int64  
 13  sqft_basement  21613 non-null  int64  
 14  yr_built       21613 non-null  int64  
 15  yr_renovated   21613 non-null  int64  
 16  zipcode        21613 non-null  int64  
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
 19  sqft_living15  21613 non-null  int64  
 20  sqft_lot15     21613 non-null  int64  
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
```

In [8]:
```
house_data.isnull().sum()
```

Out[8]:
```
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
```

In [9]:
```
house_data.describe()
```

Out[9]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	216
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	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	1.068800e+04	
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	

In [10]:
```
house_data.drop("id",axis=1,inplace=True)
```

In [11]:
```
house_data.head(5)
```

Out[11]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	vie
0	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	
1	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	
2	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	
3	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	
4	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	

In [12]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
house_data["date"]=le.fit_transform(house_data["date"])
house_data['date'].dtype
```

Out[12]:
```
dtype('int32')
```

In [13]:
```
house_data.head()
```

Out[13]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition
0	164	221900.0	3	1.00	1180	5650	1.0	0	0	3
1	220	538000.0	3	2.25	2570	7242	2.0	0	0	3
2	290	180000.0	2	1.00	770	10000	1.0	0	0	3
3	220	604000.0	4	3.00	1960	5000	1.0	0	0	5
4	283	510000.0	3	2.00	1680	8080	1.0	0	0	3

Exploratory Data Analysis

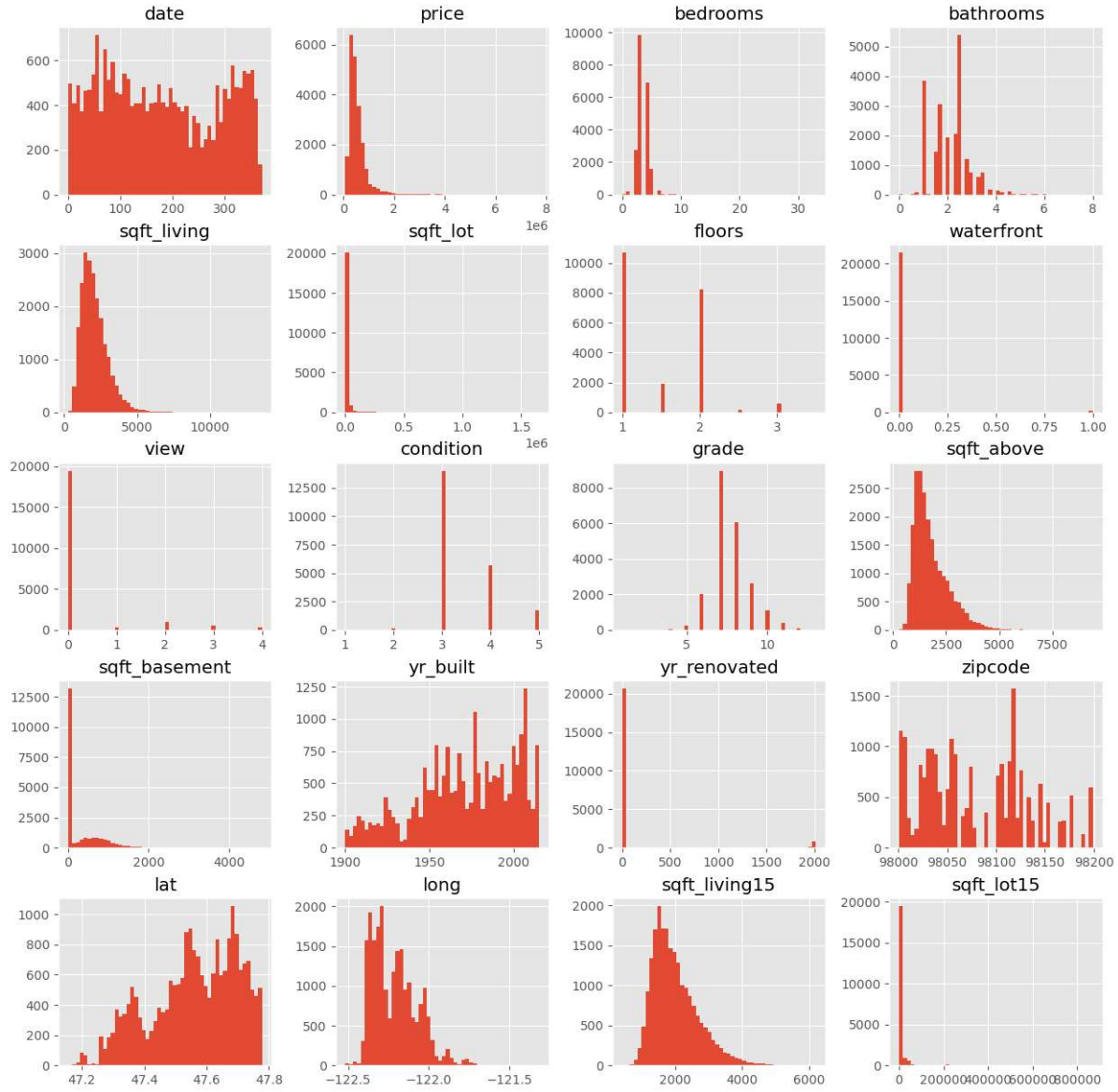
count the number of houses with unique floor values.

In [14]:
```
house_data['floors'].value_counts().to_frame()
```

Out[14]:

floors	count
1.0	10680
2.0	8241
1.5	1910
3.0	613
2.5	161
3.5	8

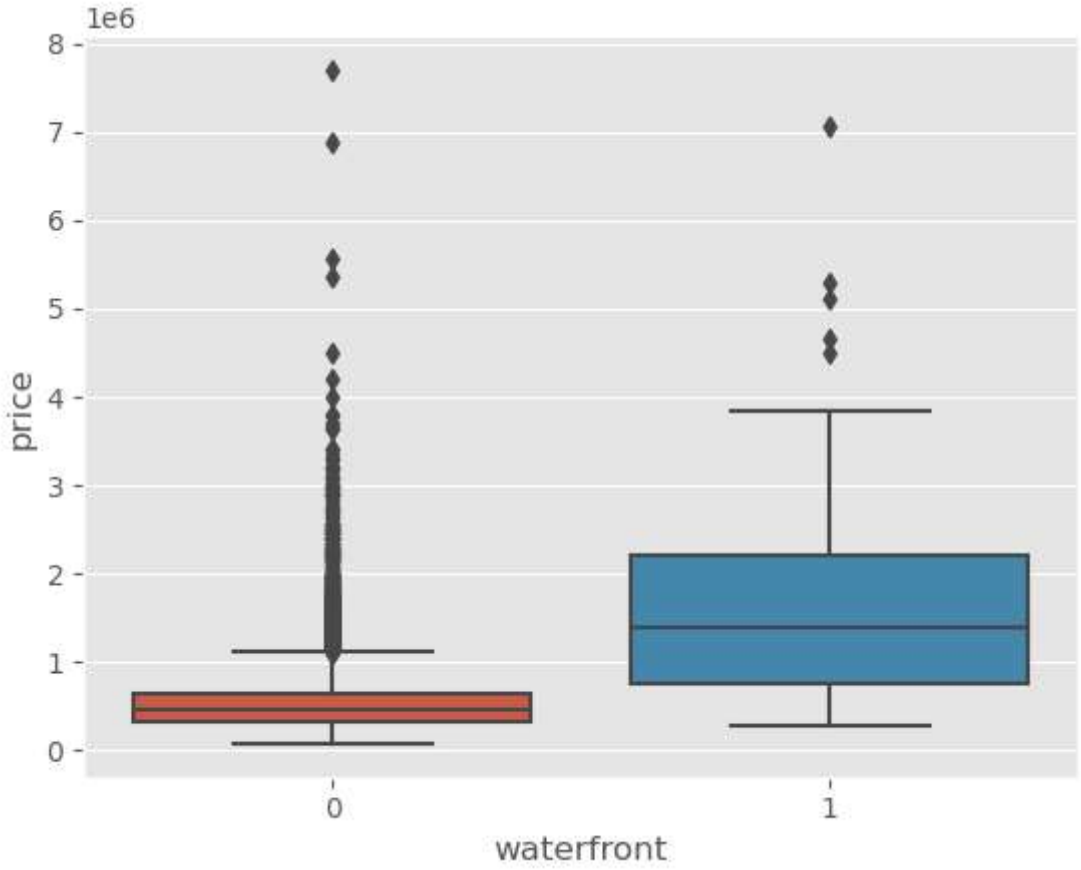
In [15]:
```
house_data.hist(bins=50,figsize=(15,15))
plt.show()
```



determine whether houses with a waterfront view or without a waterfront view have more price outliers.

In [18]:
```
sns.boxplot(data=house_data,x=house_data['waterfront'],y=house_data['price'])
```

Out[18]:
```
<Axes: xlabel='waterfront', ylabel='price'>
```

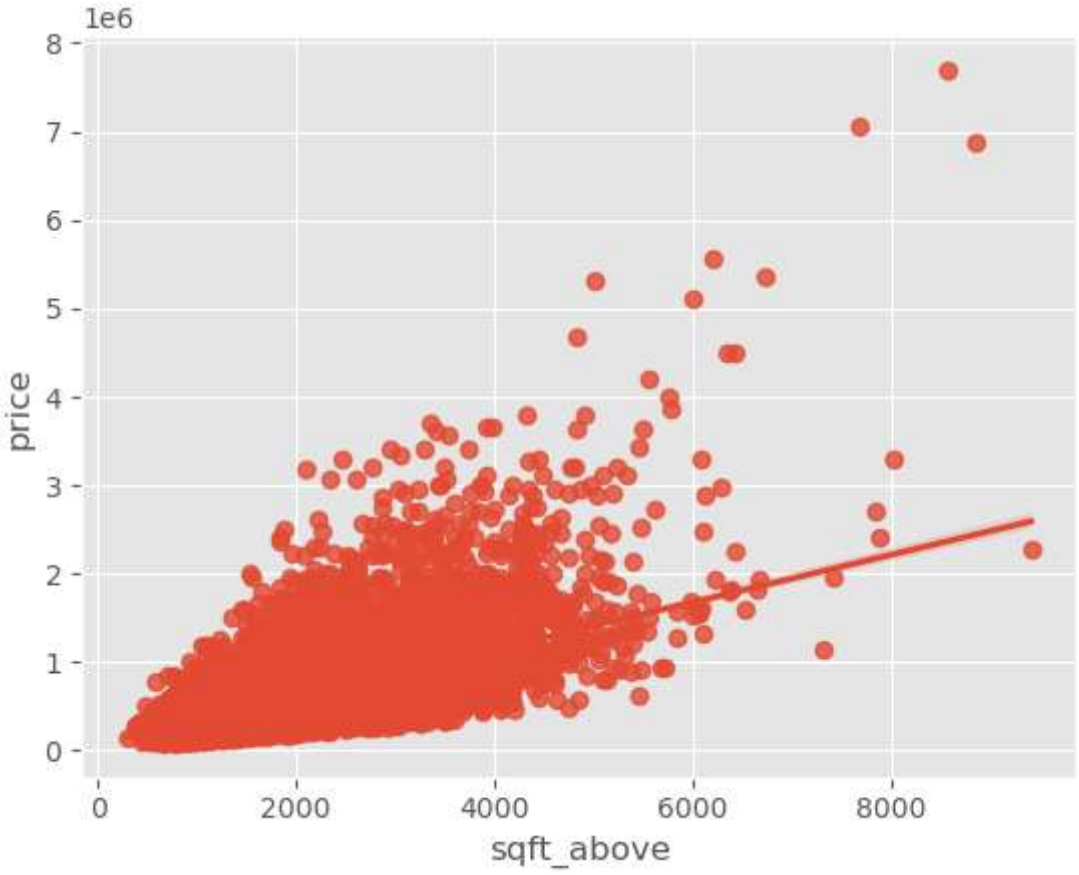


determine if the feature sqft_above is negatively or positively correlated␣

with price.

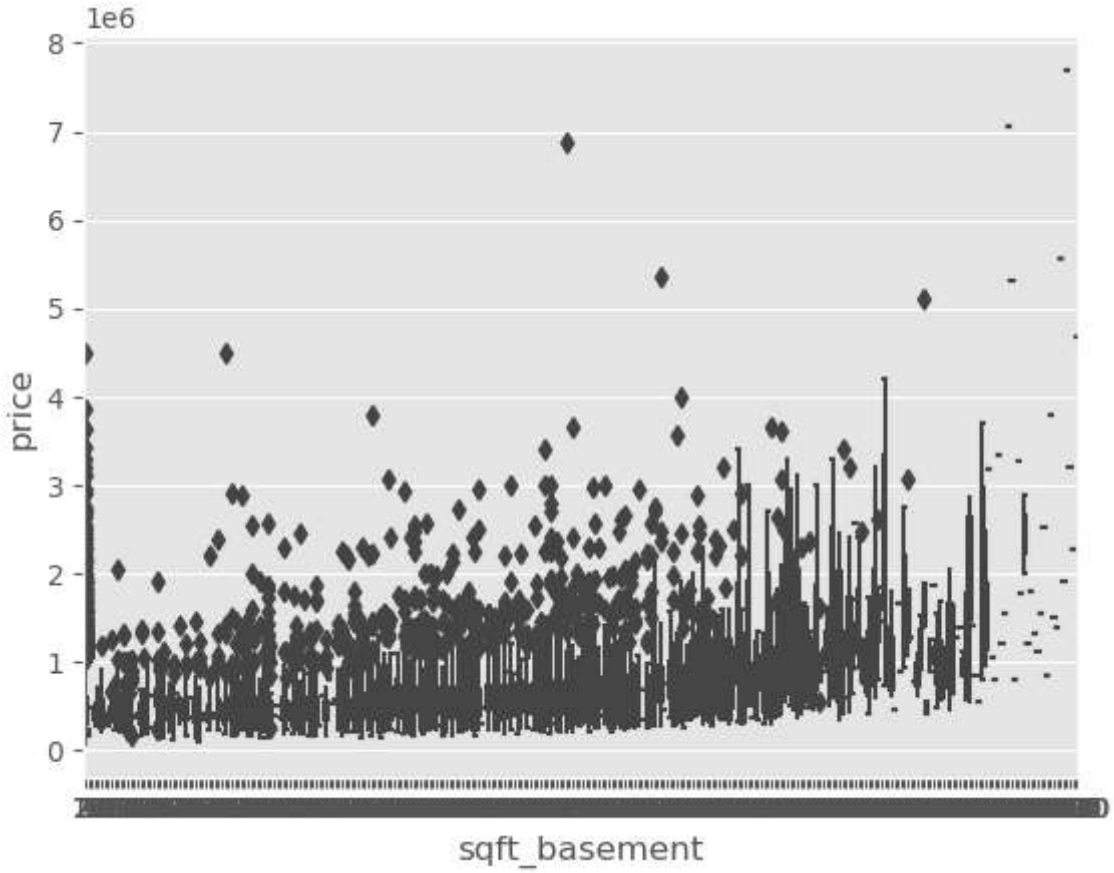
In [19]:
```
sns.regplot(data=house_data,x=house_data['sqft_above'],y=house_data['price'])
```

Out[19]:
```
<Axes: xlabel='sqft_above', ylabel='price'>
```



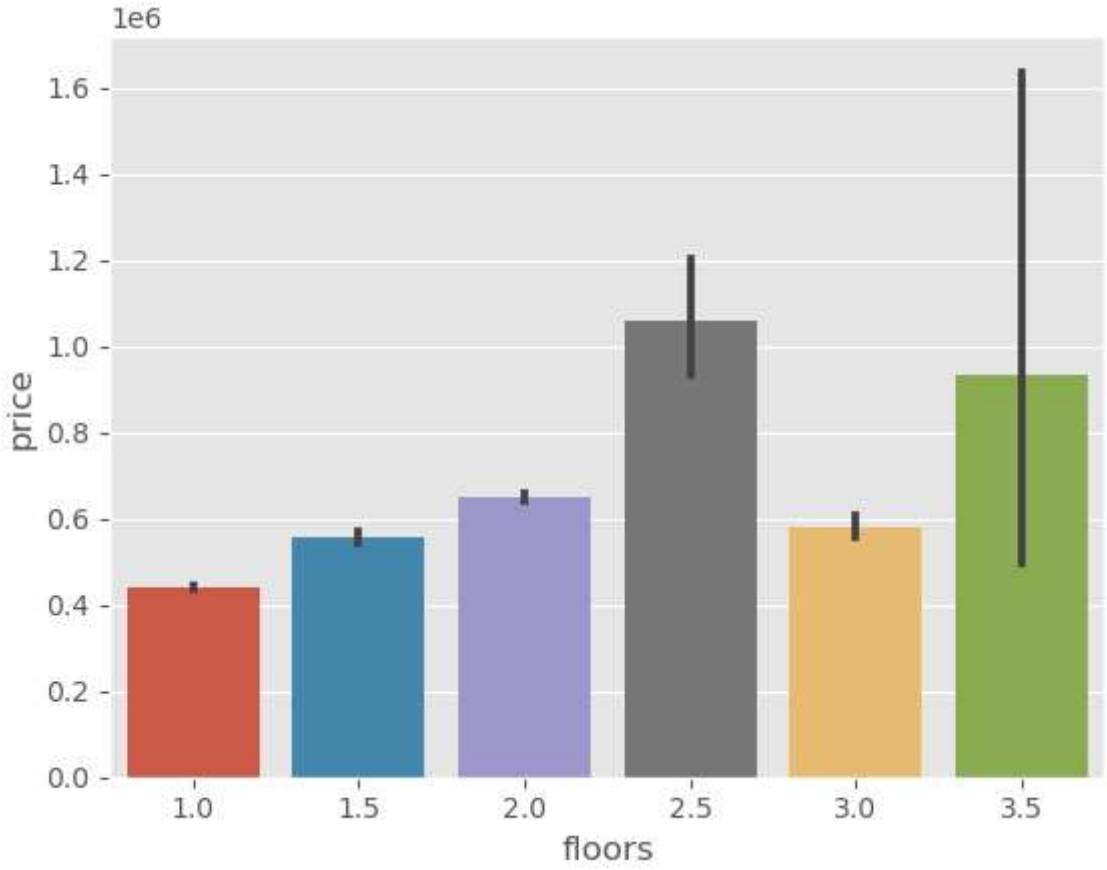
In [20]:
```
sns.boxplot(data=house_data,x=house_data['sqft_basement'],y=house_data['price'])
```

Out[20]:
```
<Axes: xlabel='sqft_basement', ylabel='price'>
```



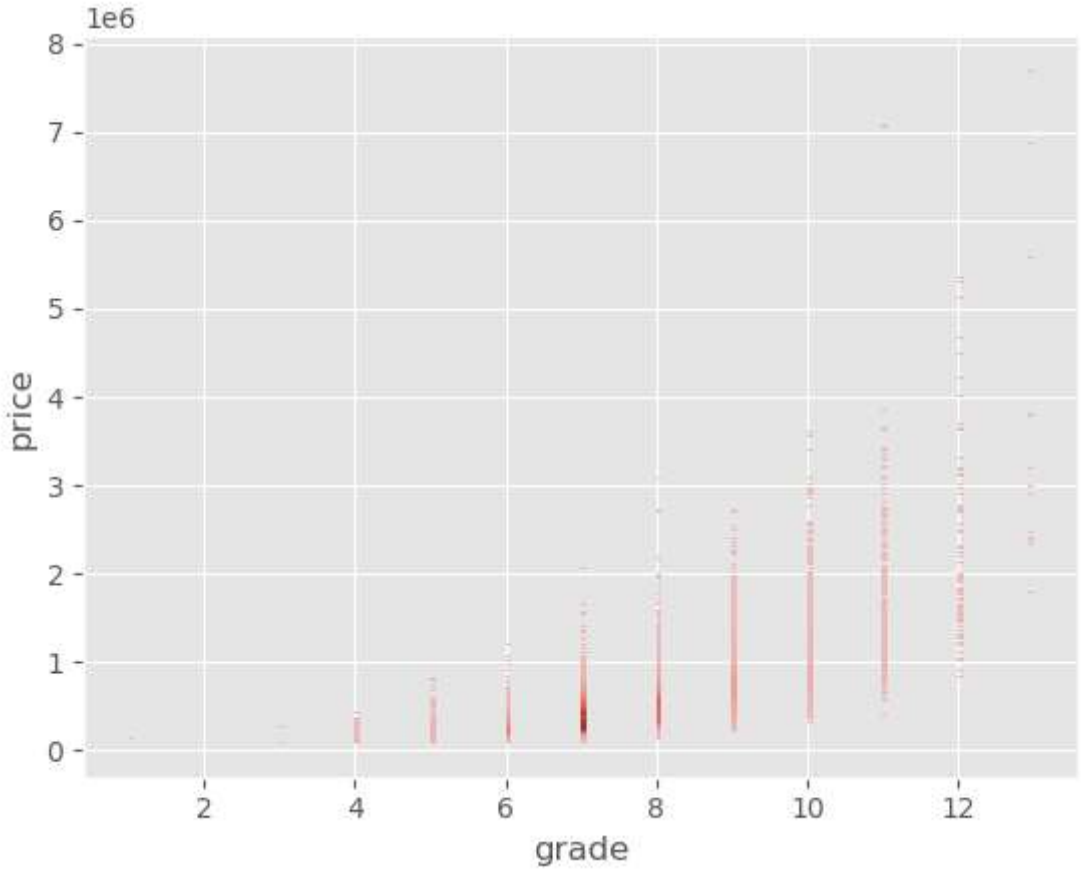
In [21]:
```
sns.barplot(data=house_data,x=house_data['floors'],y=house_data['price'])
```

Out[21]:
```
<Axes: xlabel='floors', ylabel='price'>
```



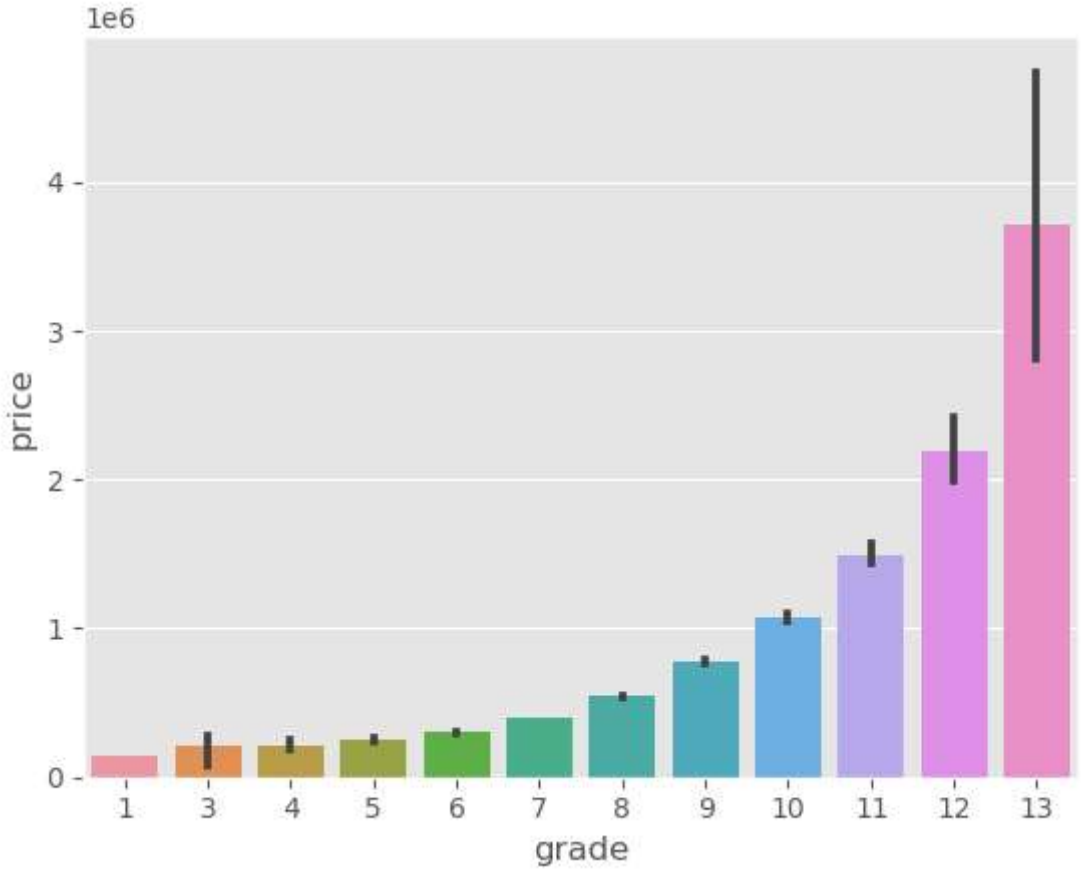
In [22]:
```
sns.histplot(data=house_data, x=house_data['grade'],y=house_data['price'] )
```

Out[22]:
```
<Axes: xlabel='grade', ylabel='price'>
```



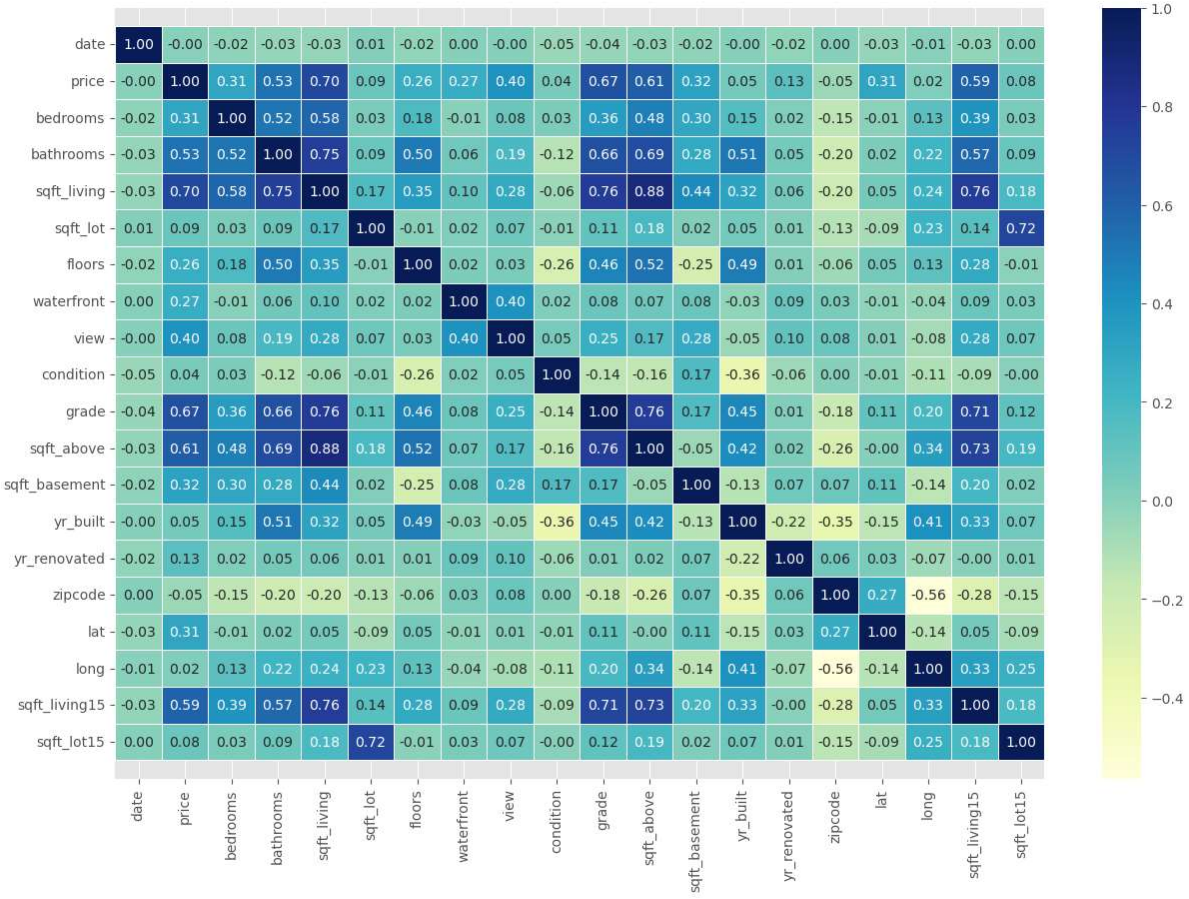
In [23]:
```
sns.barplot(data=house_data,x=house_data['grade'],y=house_data['price'])
```

Out[23]:
```
<Axes: xlabel='grade', ylabel='price'>
```

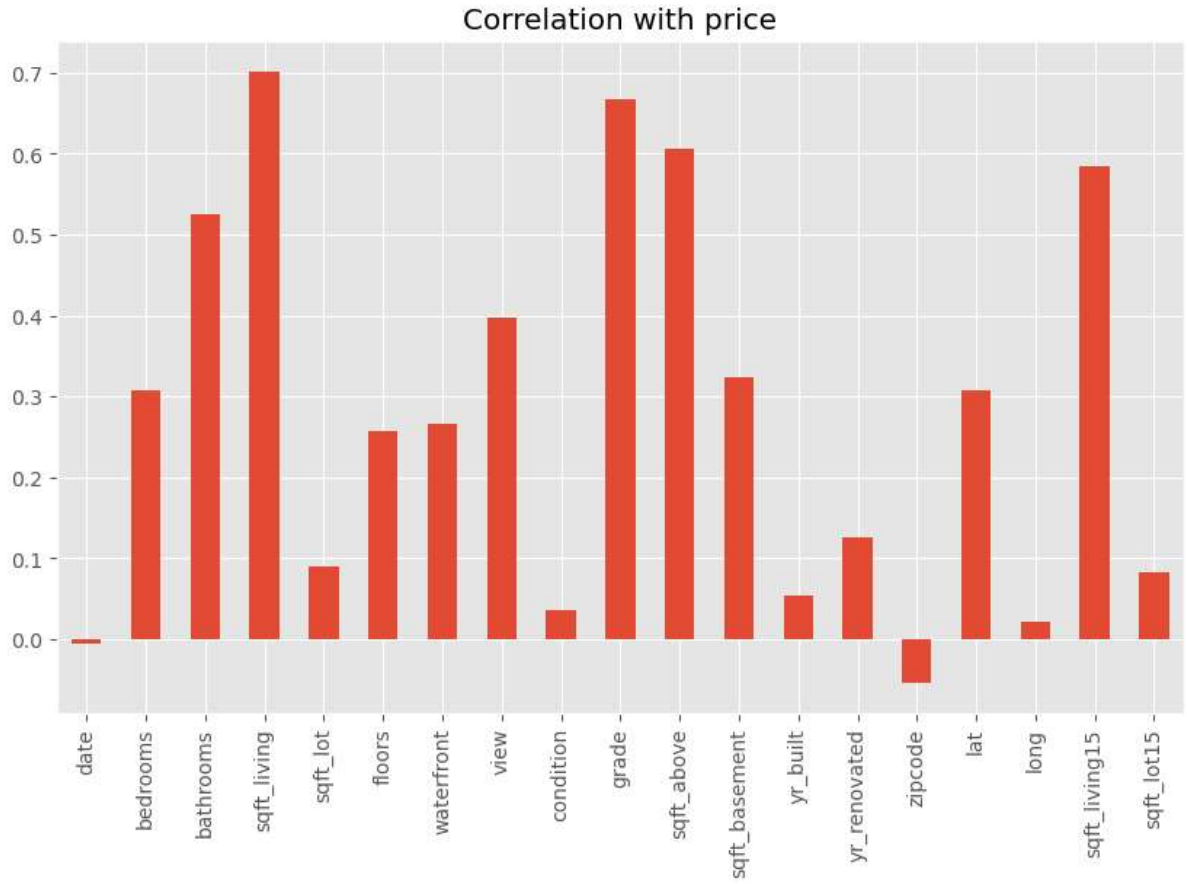


In [24]:
```
corr_matrix = house_data.corr()
fig, ax = plt.subplots(figsize=(15, 10))
ax = sns.heatmap(corr_matrix,
annot=True,
linewidths=0.5,
fmt=".2f",
cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top- 0.5)
```

Out[24]: (20.5, -0.5)



In [26]:
```
correlation_values = house_data.drop('price', axis=1).corrwith(house_data['pric
correlation_values.plot(kind='bar', grid=True, figsize=(10, 6), title="Correlat
plt.show()
```



In [27]:
```
house_data.skew()
```

Out[27]:
```
date             0.147286
price            4.024069
bedrooms         1.974300
bathrooms        0.511108
sqft_living      1.471555
sqft_lot        13.060019
floors           0.616177
waterfront      11.385108
view             3.395750
condition        1.032805
grade            0.771103
sqft_above       1.446664
sqft_basement    1.577965
yr_built        -0.469805
yr_renovated     4.549493
zipcode          0.405661
lat             -0.485270
long             0.885053
sqft_living15    1.108181
sqft_lot15       9.506743
dtype: float64
```

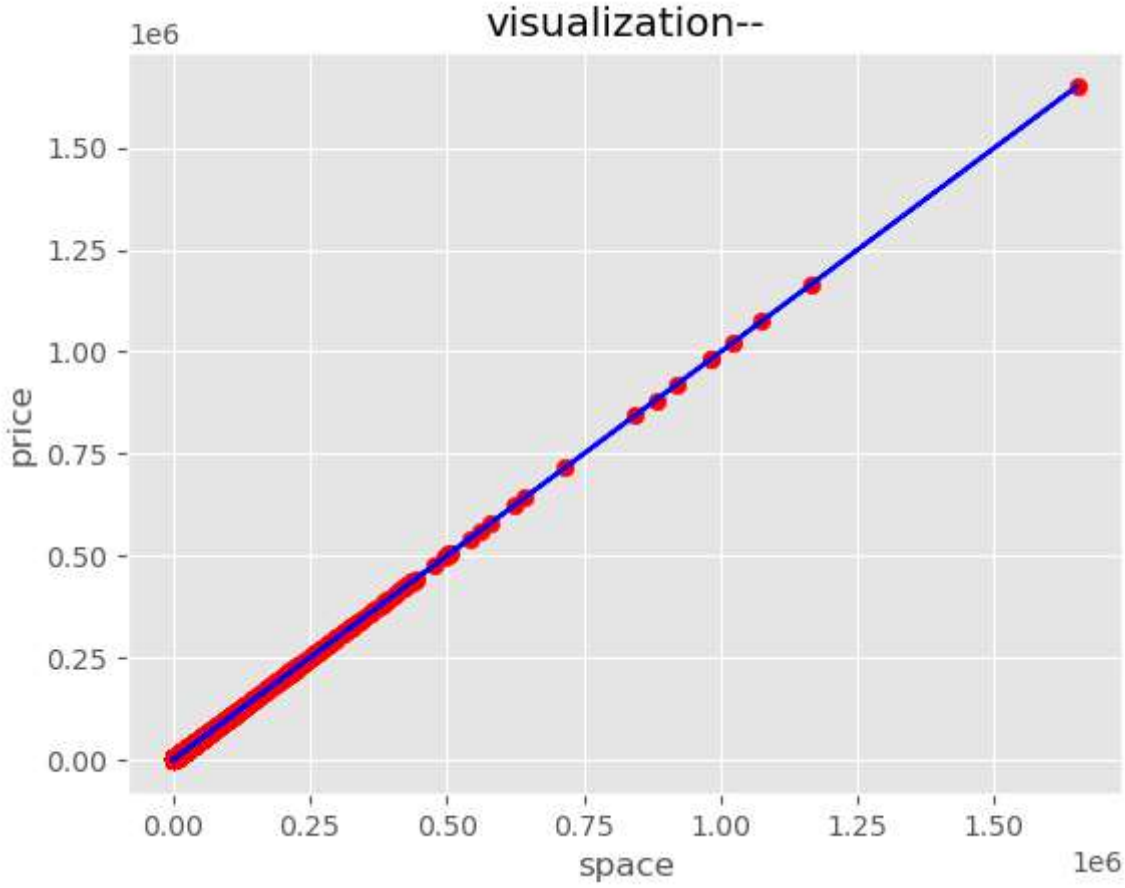
Divide the Data into Train and Test

In [28]:
```
from sklearn.model_selection import train_test_split
X=np.array(house_data.drop(columns="price"))
y=np.array(house_data.drop(columns='price'))
space=house_data["sqft_living"]
price=house_data["price"]
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=
```

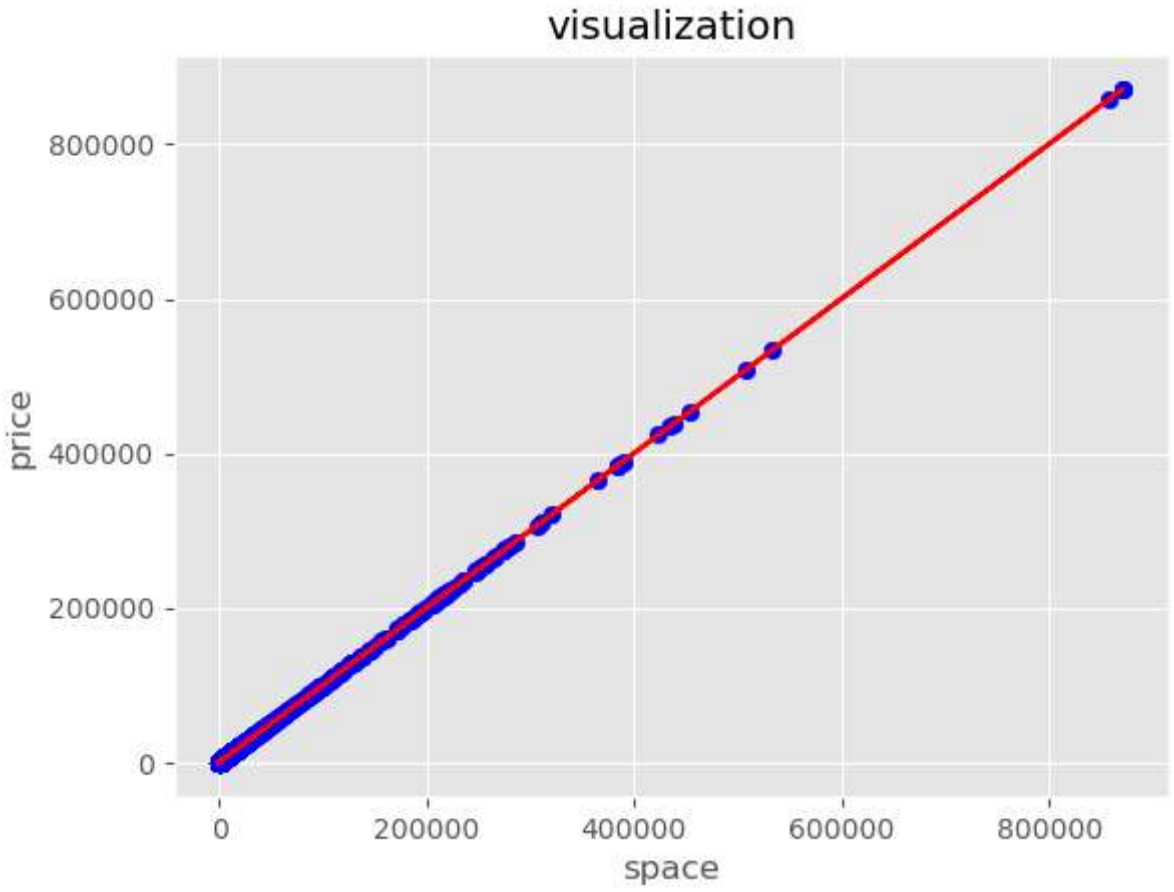
In [29]:
```
from sklearn.metrics import r2_score,mean_absolute_error
model3=LinearRegression()
model3.fit(X_train,y_train)
y_pred3=model3.predict(X_test)

print(f'R2 Score is : {r2_score(y_test,y_pred3)}')
print(f'Mae is : {mean_absolute_error(y_test,y_pred3)}')
```

```
R2 Score is : 1.0
Mae is : 1.145764819987371e-12
```

In [30]:
```
plt.scatter(X_train, y_train, color='red')
plt.plot(X_train, y_train, color='blue')
plt.title("visualization--")
plt.xlabel('space')
plt.ylabel('price')
plt.show()
```



In [31]:
```
plt.scatter(X_test, y_test, label='Actual data',color='blue')
plt.plot(X_test, y_test, color='red')
plt.title("visualization")
plt.xlabel('space')
plt.ylabel('price')
plt.show()
```



In [ ]:
```
```