Importing Required Libraries

In [3]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

EDA

In [4]:
```
df = pd.read_csv('C:/Users/BALLA/Desktop/SERGE/JUPYTER/ANN/DATA/kc_house_data.csv')
```

In [5]:
```
#df.isnull().sum()
```

In [6]:
```
df.info()
```

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

In [7]:
```
df.describe().transpose()
```

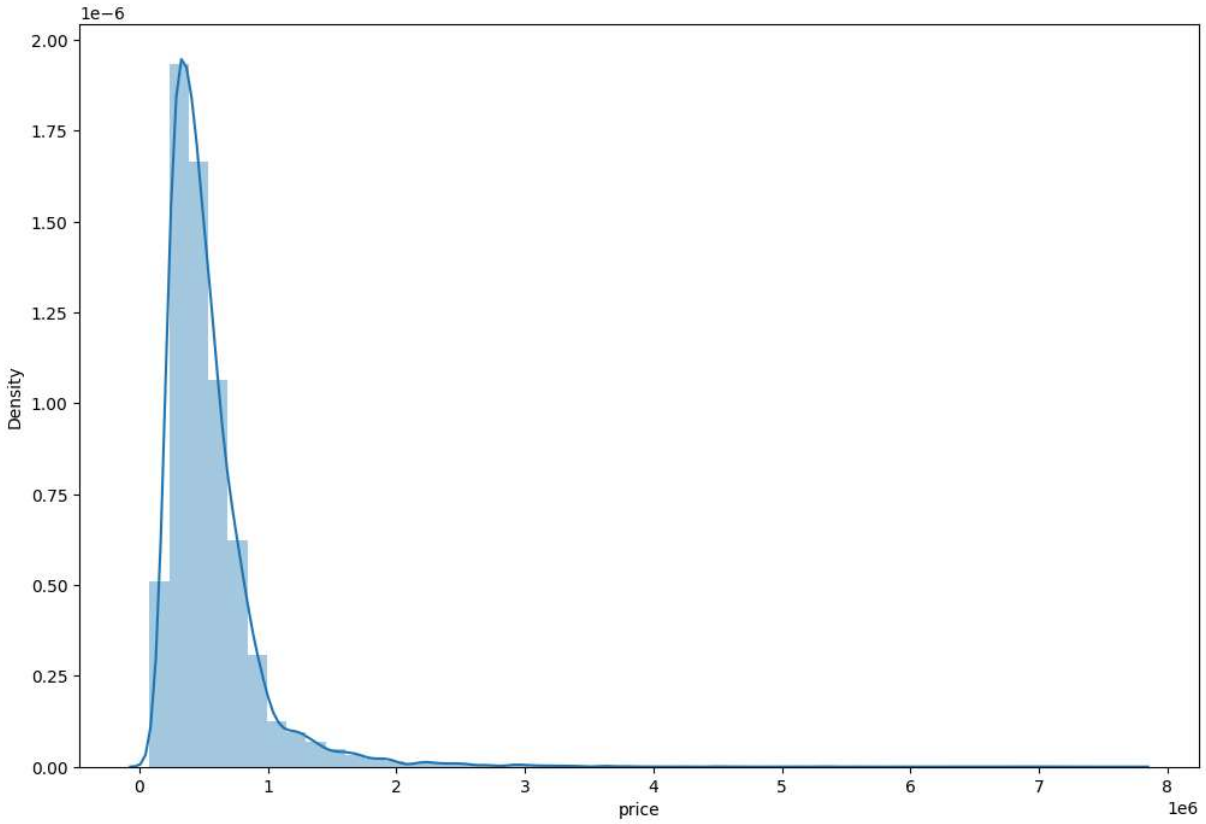
Out[7]:

	count	mean	std	min	25%	50%	75%
id	21597.0	4.580474e+09	2.876736e+09	1.000102e+06	2.123049e+09	3.904930e+09	7.308900e+0
price	21597.0	5.402966e+05	3.673681e+05	7.800000e+04	3.220000e+05	4.500000e+05	6.450000e+0
bedrooms	21597.0	3.373200e+00	9.262989e-01	1.000000e+00	3.000000e+00	3.000000e+00	4.000000e+0
bathrooms	21597.0	2.115826e+00	7.689843e-01	5.000000e-01	1.750000e+00	2.250000e+00	2.500000e+0
sqft_living	21597.0	2.080322e+03	9.181061e+02	3.700000e+02	1.430000e+03	1.910000e+03	2.550000e+0
sqft_lot	21597.0	1.509941e+04	4.141264e+04	5.200000e+02	5.040000e+03	7.618000e+03	1.068500e+0
floors	21597.0	1.494096e+00	5.396828e-01	1.000000e+00	1.000000e+00	1.500000e+00	2.000000e+0
waterfront	21597.0	7.547345e-03	8.654900e-02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+0
view	21597.0	2.342918e-01	7.663898e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+0
condition	21597.0	3.409825e+00	6.505456e-01	1.000000e+00	3.000000e+00	3.000000e+00	4.000000e+0
grade	21597.0	7.657915e+00	1.173200e+00	3.000000e+00	7.000000e+00	7.000000e+00	8.000000e+0
sqft_above	21597.0	1.788597e+03	8.277598e+02	3.700000e+02	1.190000e+03	1.560000e+03	2.210000e+0
sqft_basement	21597.0	2.917250e+02	4.426678e+02	0.000000e+00	0.000000e+00	0.000000e+00	5.600000e+0
yr_built	21597.0	1.971000e+03	2.937523e+01	1.900000e+03	1.951000e+03	1.975000e+03	1.997000e+0
yr_renovated	21597.0	8.446479e+01	4.018214e+02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+0
zipcode	21597.0	9.807795e+04	5.351307e+01	9.800100e+04	9.803300e+04	9.806500e+04	9.811800e+0
lat	21597.0	4.756009e+01	1.385518e-01	4.715590e+01	4.747110e+01	4.757180e+01	4.767800e+0
long	21597.0	-1.222140e+02	1.407235e-01	-1.225190e+02	-1.223280e+02	-1.222310e+02	-1.221250e+0
sqft_living15	21597.0	1.986620e+03	6.852305e+02	3.990000e+02	1.490000e+03	1.840000e+03	2.360000e+0
sqft_lot15	21597.0	1.275828e+04	2.727444e+04	6.510000e+02	5.100000e+03	7.620000e+03	1.008300e+0

In [8]:
```
plt.figure(figsize=(12,8))
sns.distplot(df['price'])
```

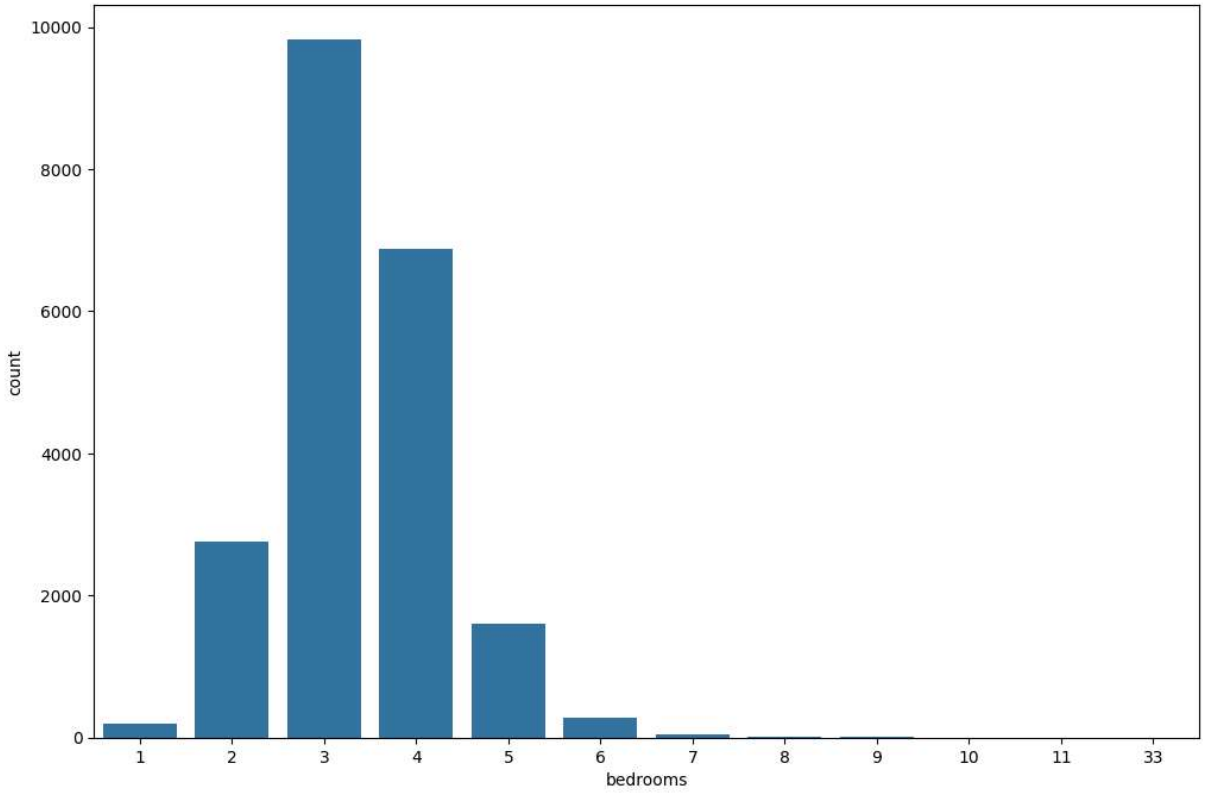
```
C:\Users\BALLA\AppData\Local\Temp\ipykernel_9836\4209215003.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df['price'])
```

Out[8]: <Axes: xlabel='price', ylabel='Density'>



In [9]:
```
plt.figure(figsize=(12,8))
sns.countplot(x='bedrooms',data=df)
```

Out[9]:
```
<Axes: xlabel='bedrooms', ylabel='count'>
```



In [17]:
```
df1 = df.drop('date',axis=1)
df1
```

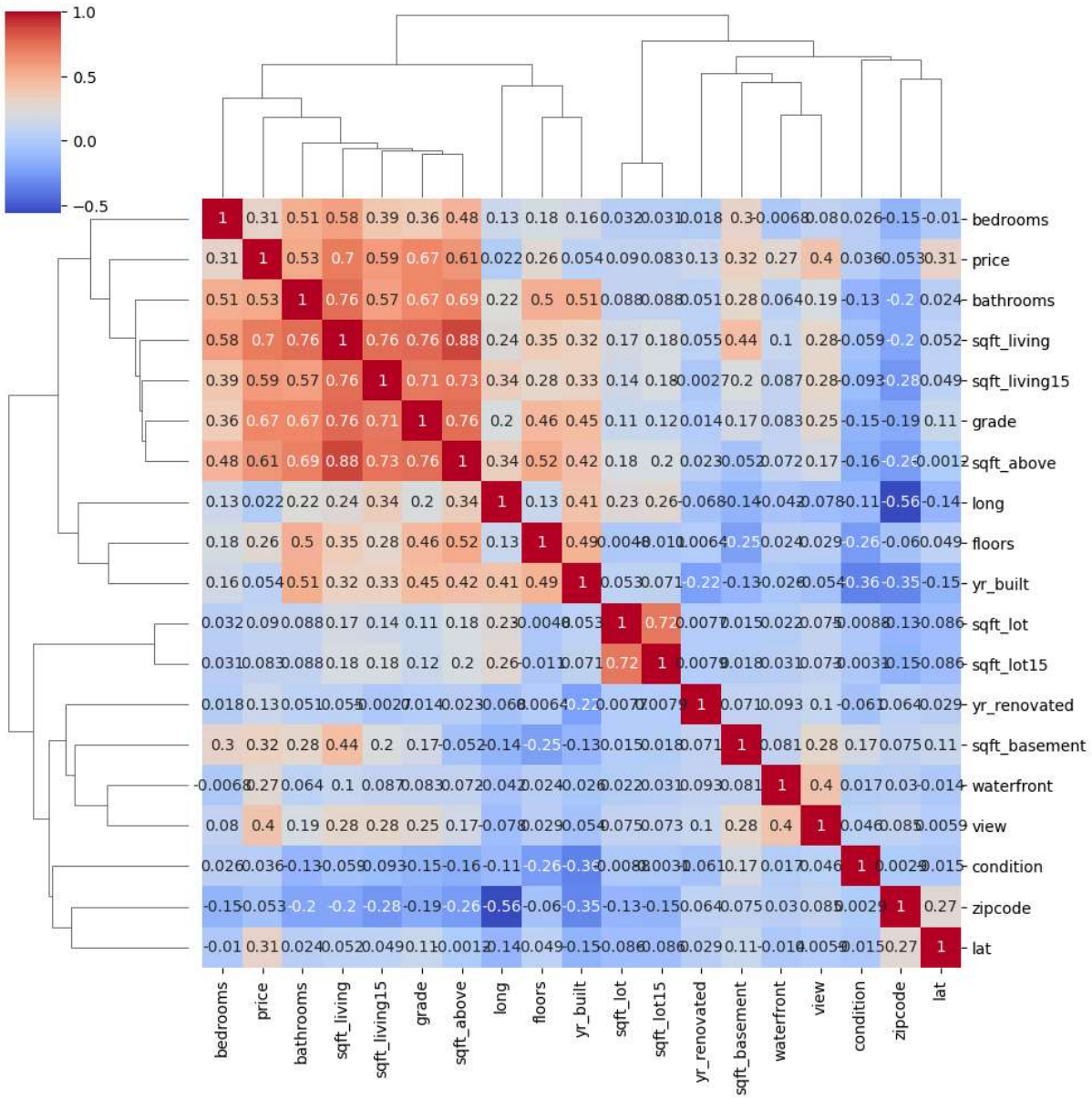
Out[17]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	gı
0	7129300520	221900.0	3	1.00	1180	5650	1.0	0	0	3	
1	6414100192	538000.0	3	2.25	2570	7242	2.0	0	0	3	
2	5631500400	180000.0	2	1.00	770	10000	1.0	0	0	3	
3	2487200875	604000.0	4	3.00	1960	5000	1.0	0	0	5	
4	1954400510	510000.0	3	2.00	1680	8080	1.0	0	0	3	
...	...	...	...	...	...	...	...	...	...	...	
21592	263000018	360000.0	3	2.50	1530	1131	3.0	0	0	3	
21593	6600060120	400000.0	4	2.50	2310	5813	2.0	0	0	3	
21594	1523300141	402101.0	2	0.75	1020	1350	2.0	0	0	3	
21595	291310100	400000.0	3	2.50	1600	2388	2.0	0	0	3	
21596	1523300157	325000.0	2	0.75	1020	1076	2.0	0	0	3	

21597 rows × 20 columns

In [18]:
```
df1=df1.drop('id',axis=1)
df1.corr()
plt.figure(figsize=(12,8))
sns.clustermap(df1.corr(), annot=True, cmap='coolwarm')
```

Out[18]:
```
<seaborn.matrix.ClusterGrid at 0x1b95ba0be10>

<Figure size 1200x800 with 0 Axes>
```

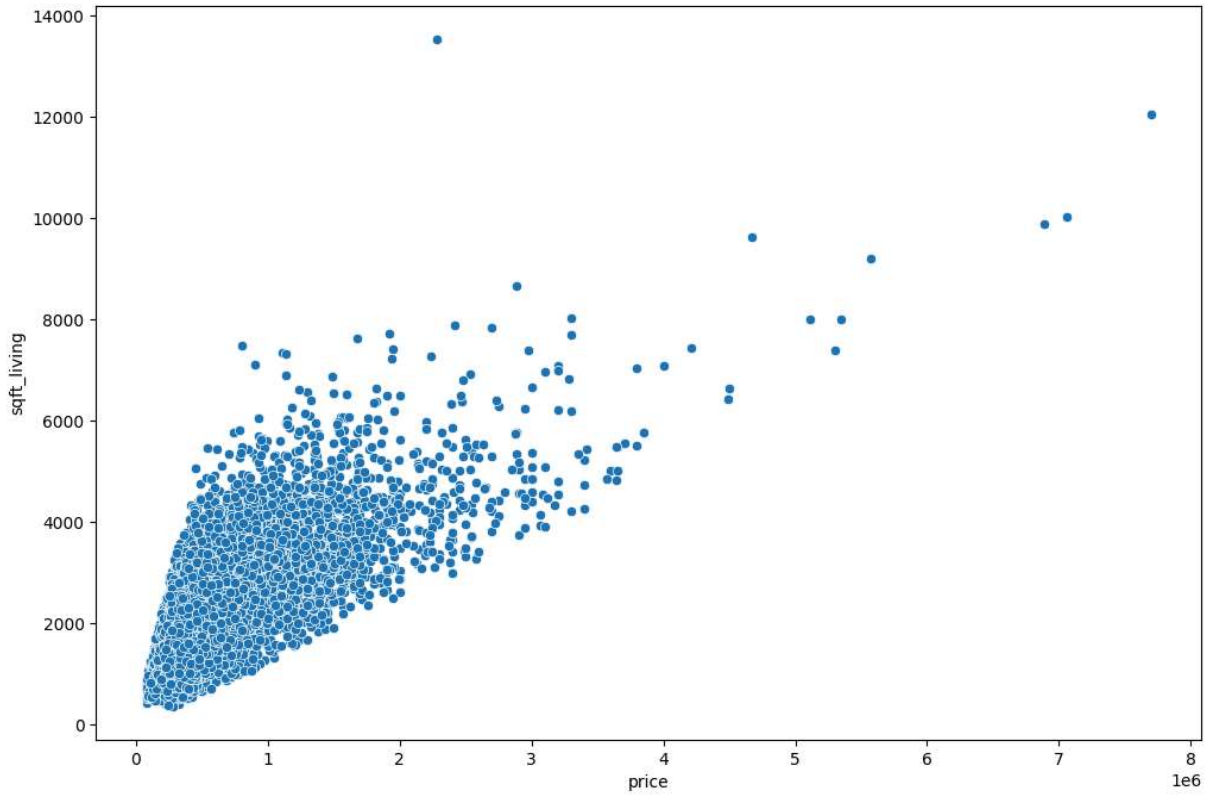


In [19]:
```
df1.corr()['price'].sort_values()
```

Out[19]:
```
zipcode         -0.053402
long             0.022036
condition        0.036056
yr_built         0.053953
sqft_lot15       0.082845
sqft_lot         0.089876
yr_renovated     0.126424
floors           0.256804
waterfront       0.266398
lat              0.306692
bedrooms         0.308787
sqft_basement    0.323799
view             0.397370
bathrooms        0.525906
sqft_living15    0.585241
sqft_above       0.605368
grade            0.667951
sqft_living      0.701917
price            1.000000
Name: price, dtype: float64
```

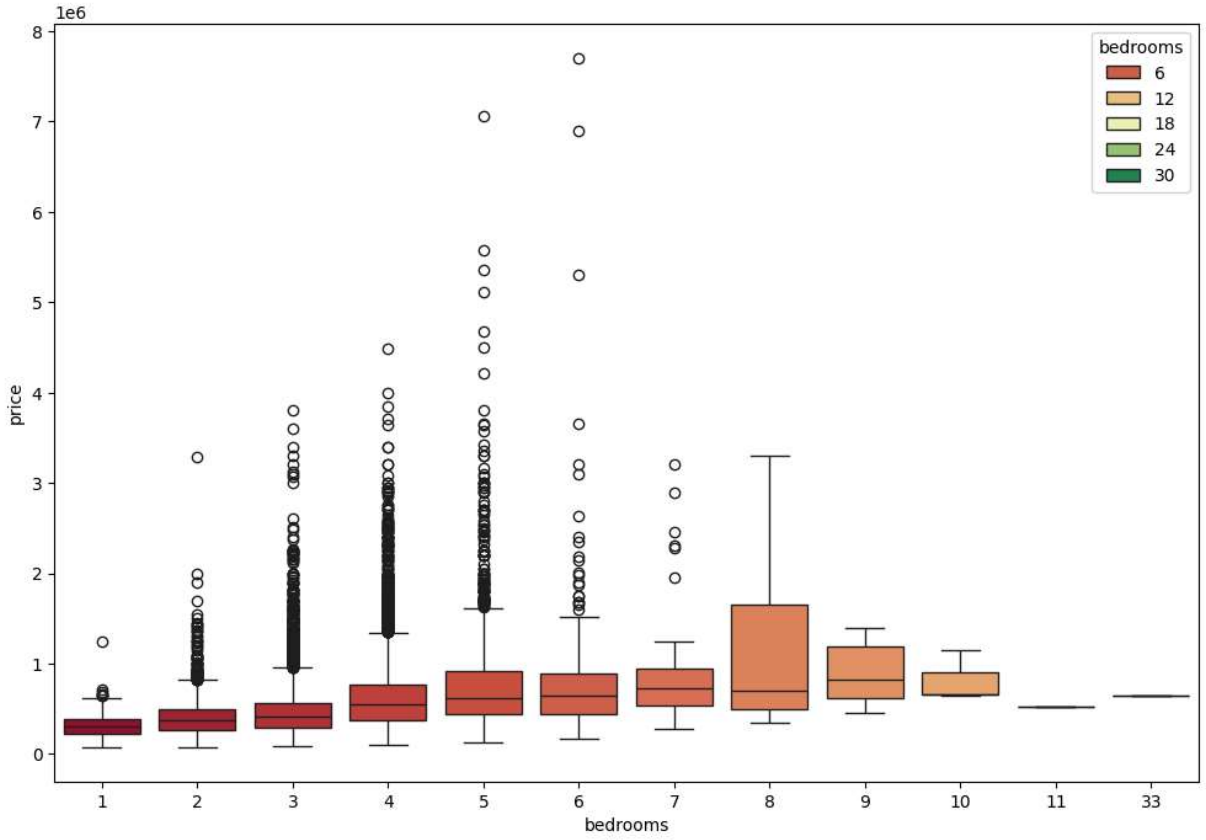
In [20]:
```
plt.figure(figsize=(12,8))
sns.scatterplot(x='price',y='sqft_living',data=df1)
```

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



In [21]:
```
plt.figure(figsize=(12,8))
sns.boxplot(x='bedrooms', y='price', data = df1, hue ='bedrooms', palette = 'RdYlGn')
```

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

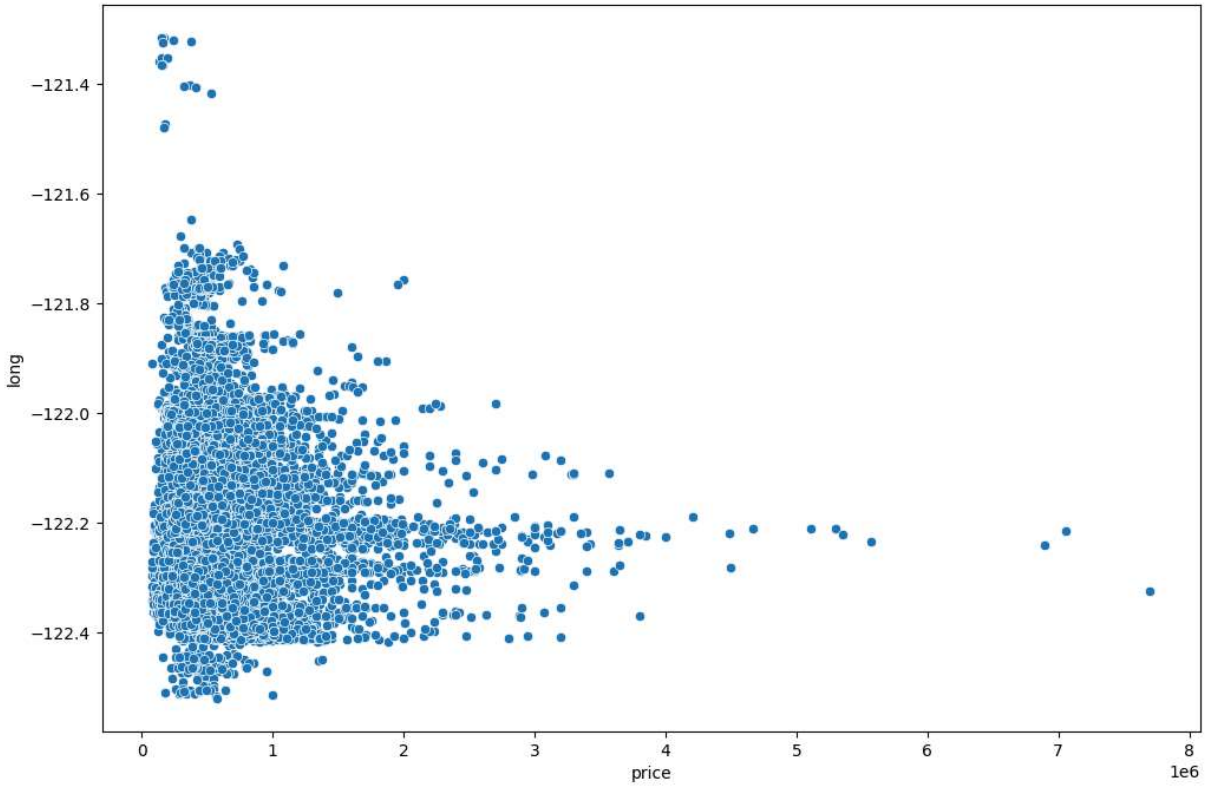


In [22]:
```
df1.columns
```

Out[22]:
```
Index(['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')
```

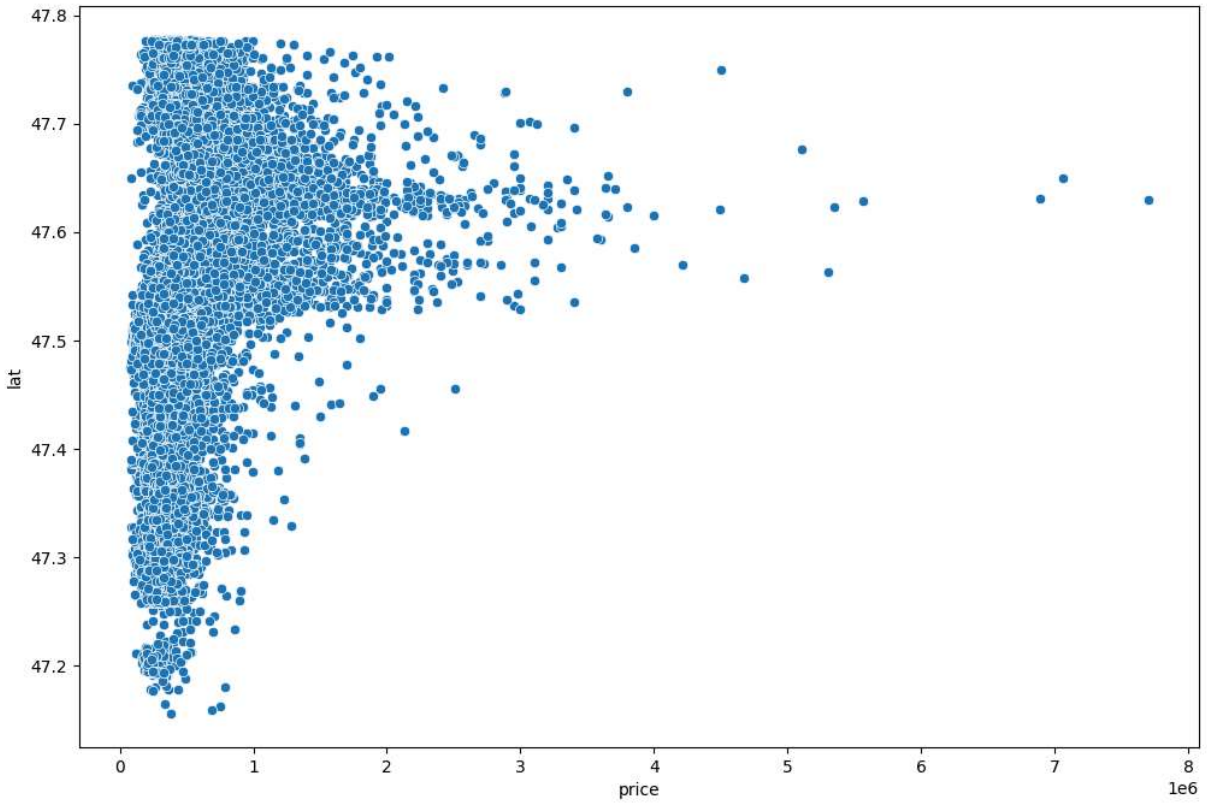
In [23]:
```
plt.figure(figsize=(12,8))
sns.scatterplot(x='price',y='long',data=df1)
```

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



In [24]:
```
plt.figure(figsize=(12,8))
sns.scatterplot(x='price',y='lat',data=df1)
```

Out[24]: <Axes: xlabel='price', ylabel='lat'>



In [113]:
```
df1.sort_values('price', ascending=False).head(20)
```

Out[113]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_al
7245	7700000.0	6	8.00	12050	27600	2.5	0	3	4	13	
3910	7060000.0	5	4.50	10040	37325	2.0	1	2	3	11	
9245	6890000.0	6	7.75	9890	31374	2.0	0	4	3	13	
4407	5570000.0	5	5.75	9200	35069	2.0	0	0	3	13	
1446	5350000.0	5	5.00	8000	23985	2.0	0	4	3	12	
1313	5300000.0	6	6.00	7390	24829	2.0	1	4	4	12	
1162	5110000.0	5	5.25	8010	45517	2.0	1	4	3	12	
8085	4670000.0	5	6.75	9640	13068	1.0	1	4	3	12	
2624	4500000.0	5	5.50	6640	40014	2.0	1	4	3	12	
8629	4490000.0	4	3.00	6430	27517	2.0	0	0	3	12	
12358	4210000.0	5	6.00	7440	21540	2.0	0	0	3	12	
4145	4000000.0	4	5.50	7080	16573	2.0	0	0	3	12	
2083	3850000.0	4	4.25	5770	21300	2.0	1	4	4	11	
7028	3800000.0	5	5.50	7050	42840	1.0	0	2	4	13	
19002	3800000.0	3	4.25	5510	35000	2.0	0	4	3	13	
16288	3710000.0	4	3.50	5550	28078	2.0	0	2	4	12	
18467	3650000.0	5	3.75	5020	8694	2.0	0	1	3	12	
6502	3650000.0	6	4.75	5480	19401	1.5	1	4	5	11	
15241	3640000.0	4	3.25	4830	22257	2.0	1	4	4	11	
19133	3640000.0	5	6.00	5490	19897	2.0	0	0	3	12	

In [110]:
```
#above_3m=df1[df1['price']>3000000]
#above_3m.sort_values('price',ascending=False)
```

In [111]:
```
#above_3m.count()
```

In [114]:
```
bott_99_per=df1.sort_values('price', ascending=False).iloc[216:]
bott_99_per
```

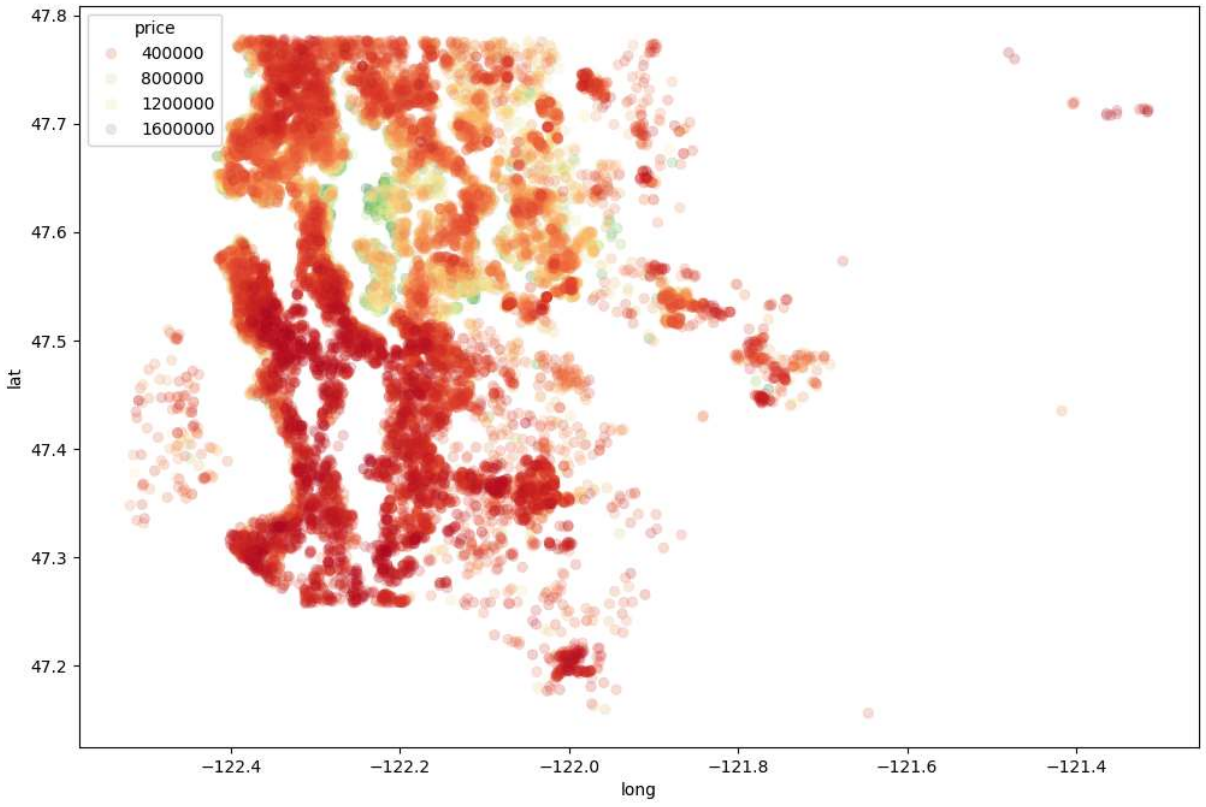
Out[114]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_al
6329	1970000.0	4	3.50	4370	8510	2.0	0	1	3	10	
12271	1960000.0	5	4.50	6200	23373	3.0	0	1	4	11	
9166	1960000.0	3	1.75	3330	12566	1.0	1	4	4	8	
12565	1960000.0	4	4.00	4430	31353	2.0	0	0	3	12	
1150	1960000.0	4	2.75	3120	7898	1.0	1	4	4	8	
...	...	...	...	...	...	...	...	...	...	...	...
2139	82500.0	2	1.00	520	22334	1.0	0	0	2	5	
8267	82000.0	3	1.00	860	10426	1.0	0	0	3	6	
16184	81000.0	2	1.00	730	9975	1.0	0	0	1	5	
465	80000.0	1	0.75	430	5050	1.0	0	0	2	4	
15279	78000.0	2	1.00	780	16344	1.0	0	0	1	5	

21381 rows × 19 columns

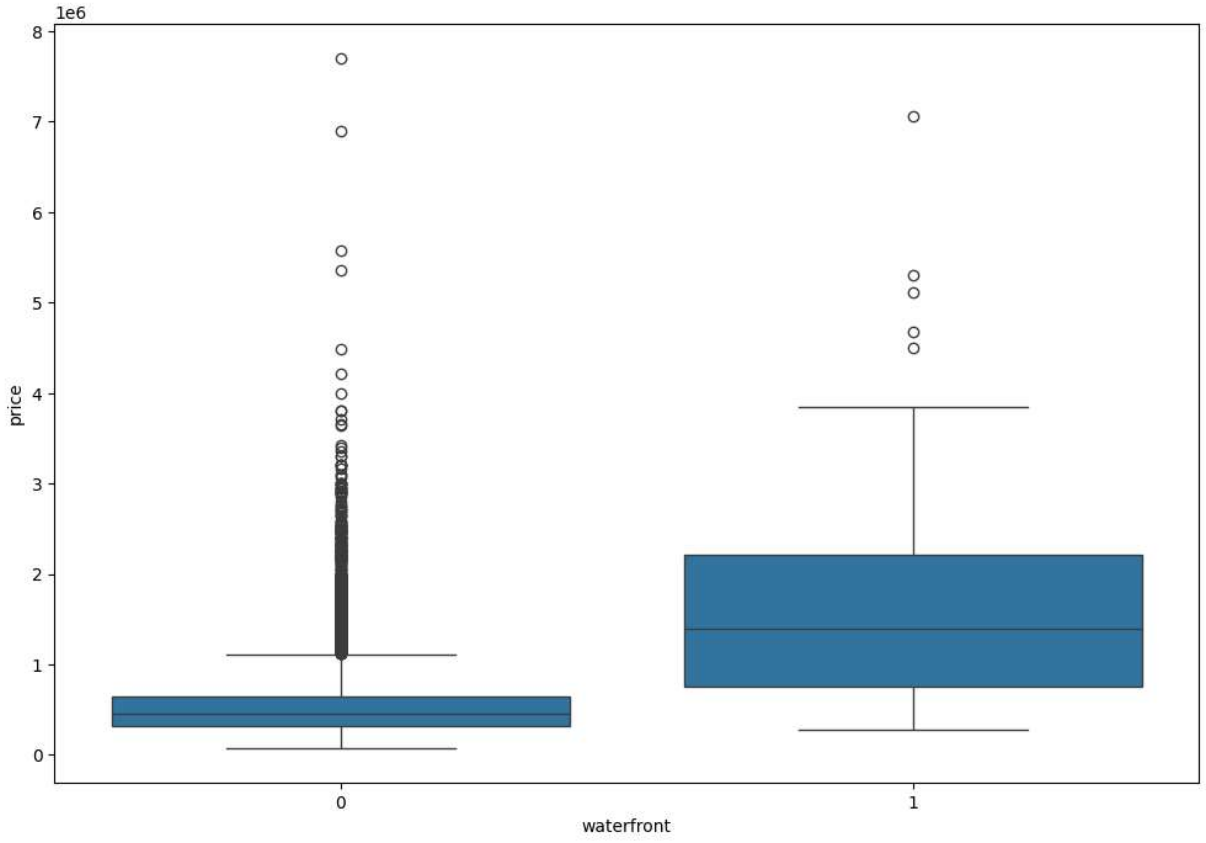
In [30]:
```
plt.figure(figsize=(12,8))
sns.scatterplot(y='lat',x='long',data=bott_99_per,
                hue = 'price', edgecolor = None,alpha=0.2, palette = 'RdYlGn')
```

Out[30]: <Axes: xlabel='long', ylabel='lat'>



In [31]:
```
plt.figure(figsize=(12,8))
sns.boxplot(x='waterfront', y='price', data = df1)
```

Out[31]: <Axes: xlabel='waterfront', ylabel='price'>



Feature Engineering

In [32]:
```
#removing the id column since useless
df2 = df.drop('id', axis=1)

df2.head()
```

Out[32]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade
0	10/13/2014	221900.0	3	1.00	1180	5650	1.0	0	0	3	7
1	12/9/2014	538000.0	3	2.25	2570	7242	2.0	0	0	3	7
2	2/25/2015	180000.0	2	1.00	770	10000	1.0	0	0	3	6
3	12/9/2014	604000.0	4	3.00	1960	5000	1.0	0	0	5	7
4	2/18/2015	510000.0	3	2.00	1680	8080	1.0	0	0	3	8

In [33]:
```
df2['date']
```

Out[33]:
```
0        10/13/2014
1         12/9/2014
2         2/25/2015
3         12/9/2014
4         2/18/2015
            ...
21592     5/21/2014
21593     2/23/2015
21594     6/23/2014
21595     1/16/2015
21596    10/15/2014
Name: date, Length: 21597, dtype: object
```

In [34]:
```
#converting date objects from object type to date type

df2['date']=pd.to_datetime(df2['date'])
df2['date']
```

Out[34]:
```
0       2014-10-13
1       2014-12-09
2       2015-02-25
3       2014-12-09
4       2015-02-18
           ...
21592   2014-05-21
21593   2015-02-23
21594   2014-06-23
21595   2015-01-16
21596   2014-10-15
Name: date, Length: 21597, dtype: datetime64[ns]
```

In [35]:
```
#Extracting year component from date column

df2['year'] = df2['date'].apply(lambda date:date.year )
df2['year']
```

Out[35]:
```
0        2014
1        2014
2        2015
3        2014
4        2015
         ...
21592    2014
21593    2015
21594    2014
21595    2015
21596    2014
Name: year, Length: 21597, dtype: int64
```

In [36]:
```
#Extracting month component from date column

df2['month'] = df2['date'].apply(lambda date:date.month )
df2['month']
```

Out[36]:
```
0        10
1        12
2         2
3        12
4         2
         ..
21592     5
21593     2
21594     6
21595     1
21596    10
Name: month, Length: 21597, dtype: int64
```

In [37]:
```
df2.head()
```

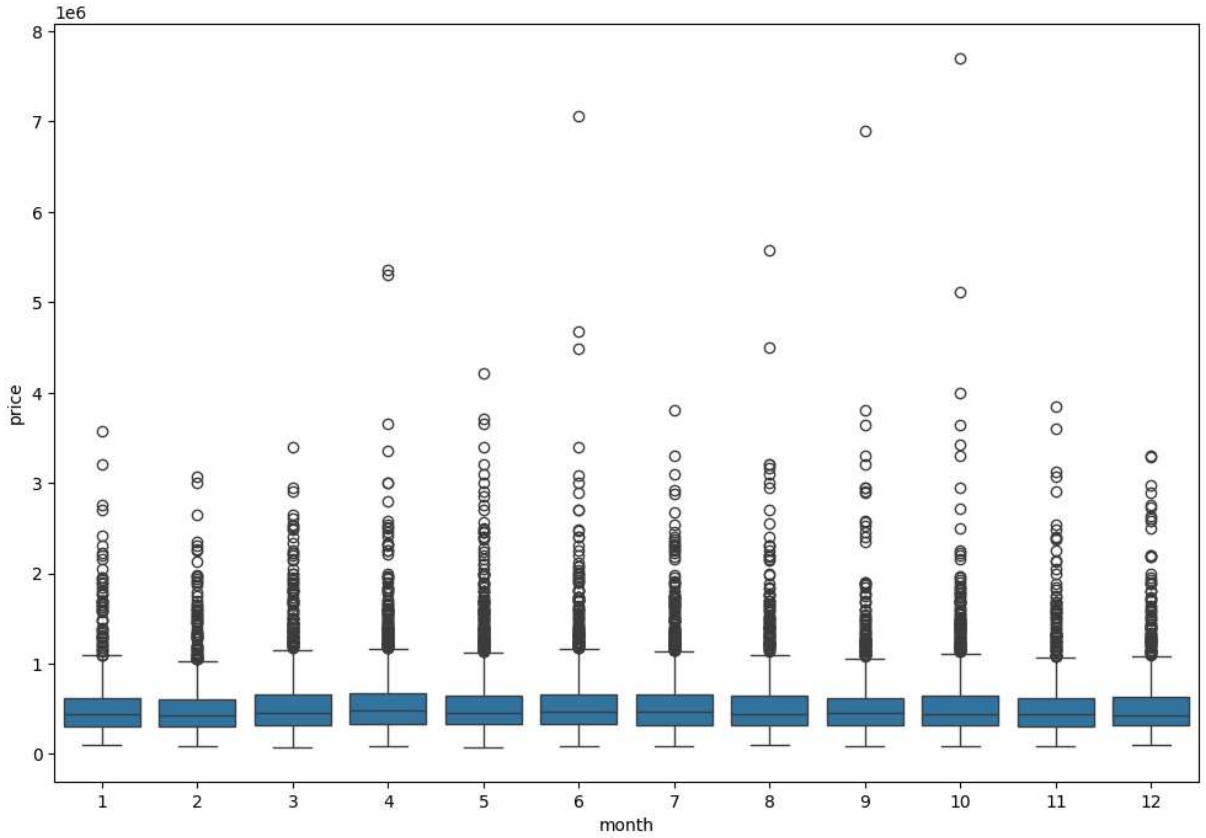
Out[37]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	...	sqft_bas
0	2014-10-13	221900.0	3	1.00	1180	5650	1.0	0	0	3	...	
1	2014-12-09	538000.0	3	2.25	2570	7242	2.0	0	0	3	...	
2	2015-02-25	180000.0	2	1.00	770	10000	1.0	0	0	3	...	
3	2014-12-09	604000.0	4	3.00	1960	5000	1.0	0	0	5	...	
4	2015-02-18	510000.0	3	2.00	1680	8080	1.0	0	0	3	...	

5 rows × 22 columns

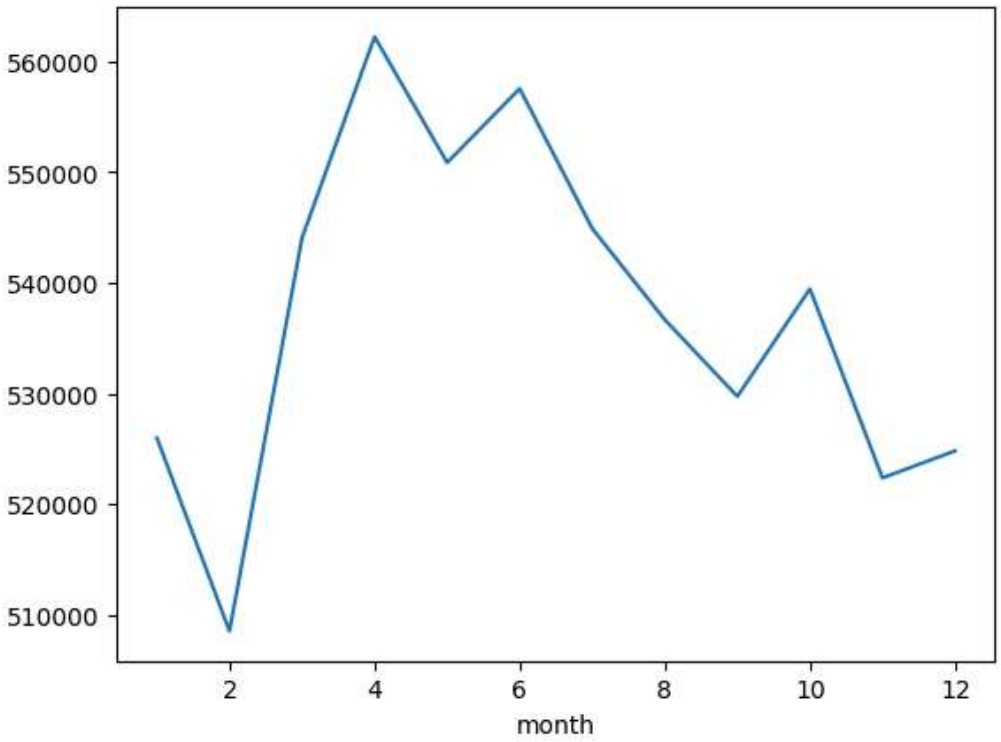
In [38]:
```
plt.figure(figsize=(12,8))
sns.boxplot(x='month', y='price', data = df2)
```

Out[38]: <Axes: xlabel='month', ylabel='price'>



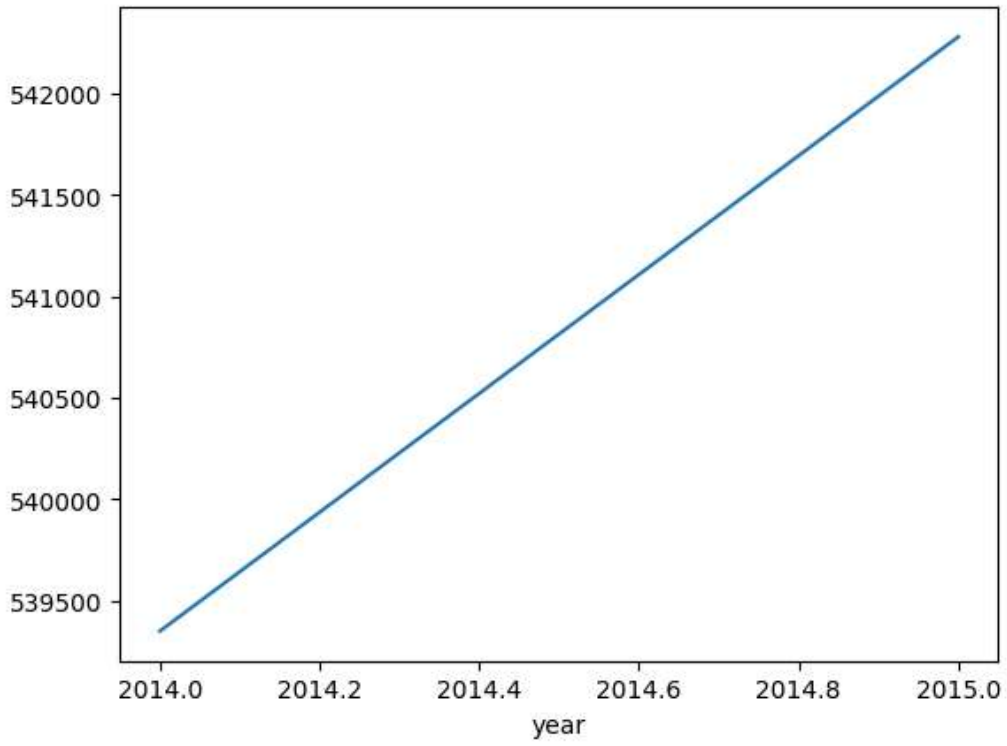
In [39]:
```
#visualising average prices per month
df2.groupby('month').mean()['price'].plot()
```

Out[39]: <Axes: xlabel='month'>



In [40]:
```
#visualising average prices per year
df2.groupby('year').mean()['price'].plot()
```

Out[40]: <Axes: xlabel='year'>



In [41]:
```
df2 = df2.drop('date', axis=1)
df2.head()
```

Out[41]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	...	sqft_bas
0	221900.0	3	1.00	1180	5650	1.0	0	0	3	7	...	
1	538000.0	3	2.25	2570	7242	2.0	0	0	3	7	...	
2	180000.0	2	1.00	770	10000	1.0	0	0	3	6	...	
3	604000.0	4	3.00	1960	5000	1.0	0	0	5	7	...	
4	510000.0	3	2.00	1680	8080	1.0	0	0	3	8	...	

5 rows × 21 columns

In [42]:
```
df2.columns
```

Out[42]:
```
Index(['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', 'year', 'month'],
      dtype='object')
```

In [43]:
```
# could make sense due to scaling, higher should correlate to more value
df['yr_renovated'].value_counts()
```

Out[43]:
```
yr_renovated
0       20683
2014       91
2013       37
2003       36
2005       35
          ...
1951        1
1959        1
1948        1
1954        1
1944        1
Name: count, Length: 70, dtype: int64
```

In [44]:
```
df['sqft_basement'].value_counts()
```

Out[44]:
```
sqft_basement
0      13110
600      221
700      218
500      214
800      206
         ...
518        1
374        1
784        1
906        1
248        1
Name: count, Length: 306, dtype: int64
```

Train-Test Split and Scaling

In [45]:
```
X = df2.drop('price',axis=1)
y = df2['price']
```

In [46]:
```
from sklearn.model_selection import train_test_split
```

In [47]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=101)
```

Scaling

In [48]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [49]:
```
scaler = MinMaxScaler()
```

In [50]:
```
X_train= scaler.fit_transform(X_train)
```

In [51]:
```
X_test = scaler.transform(X_test)
```

In [52]:
```
X_train.shape
```

Out[52]:
```
(15117, 20)
```

In [53]:
```
X_test.shape
```

Out[53]:
```
(6480, 20)
```

Creating a Model

In [54]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import Adam
```

In [55]:
```
model = Sequential()

model.add(Dense(20,activation='relu'))
model.add(Dense(20,activation='relu'))
model.add(Dense(20,activation='relu'))
model.add(Dense(20,activation='relu'))
model.add(Dense(1))

model.compile(optimizer='adam',loss='mse')
```

In [56]:
```
model.fit(x=X_train,y=y_train.values,
          validation_data=(X_test,y_test.values),
          batch_size=128,epochs=400)
```

```
Epoch 248/400
119/119 [==============================] - 0s 3ms/step - loss: 27835516928.0000 - val_l
oss: 27622483968.0000
Epoch 249/400
119/119 [==============================] - 0s 3ms/step - loss: 27729954816.0000 - val_l
oss: 27623516160.0000
Epoch 250/400
119/119 [==============================] - 0s 3ms/step - loss: 27766931456.0000 - val_l
oss: 27733452800.0000
Epoch 251/400
119/119 [==============================] - 0s 3ms/step - loss: 27715856384.0000 - val_l
oss: 27584571392.0000
Epoch 252/400
119/119 [==============================] - 0s 3ms/step - loss: 27715520512.0000 - val_l
oss: 27606095872.0000
Epoch 253/400
119/119 [==============================] - 0s 3ms/step - loss: 27706222592.0000 - val_l
oss: 27835418624.0000
Epoch 254/400
119/119 [==============================] - 0s 3ms/step - loss: 27726790656.0000 - val_l
```

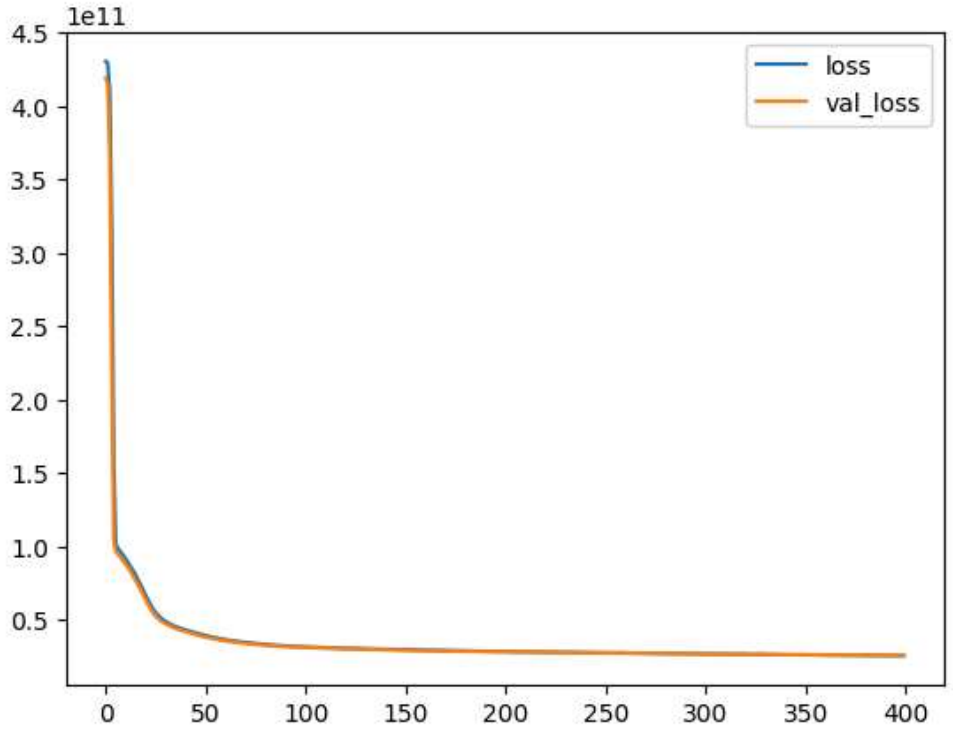
In [57]:
```
model.summary
```

Out[57]:
```
<bound method Model.summary of <keras.src.engine.sequential.Sequential object at 0x000001
B969933E90>>
```

In [58]:
```
losses = pd.DataFrame(model.history.history)
```

In [59]:
```
losses.plot()
```

Out[59]: <Axes: >



Evaluation on Test Data

In [60]:
```
from sklearn.metrics import mean_squared_error,mean_absolute_error,explained_variance_scor
```

In [61]:
```
X_test
```

Out[61]:
```
array([[0.1       , 0.08      , 0.04239917, ..., 0.00887725, 0.        ,
        0.63636364],
       [0.3       , 0.36      , 0.17269907, ..., 0.00993734, 0.        ,
        0.81818182],
       [0.2       , 0.24      , 0.12512927, ..., 0.00547073, 0.        ,
        0.90909091],
       ...,
       [0.1       , 0.08      , 0.05584281, ..., 0.00506255, 0.        ,
        1.        ],
       [0.3       , 0.2       , 0.22233713, ..., 0.00774485, 1.        ,
        0.09090909],
       [0.3       , 0.32      , 0.27611169, ..., 0.0196531 , 0.        ,
        0.45454545]])
```

In [62]:
```
predictions = model.predict(X_test)
```

```
203/203 [==============================] - 1s 2ms/step
```

In [63]:
```
mean_absolute_error(y_test,predictions)
```

Out[63]:
```
99501.33434365355
```

In [64]:
```
np.sqrt(mean_squared_error(y_test,predictions))
```

Out[64]:
```
160953.89477378476
```

In [65]:
```
explained_variance_score(y_test,predictions)
```

Out[65]:
```
0.8051231447876364
```

In [66]:
```
df['price'].mean()
```

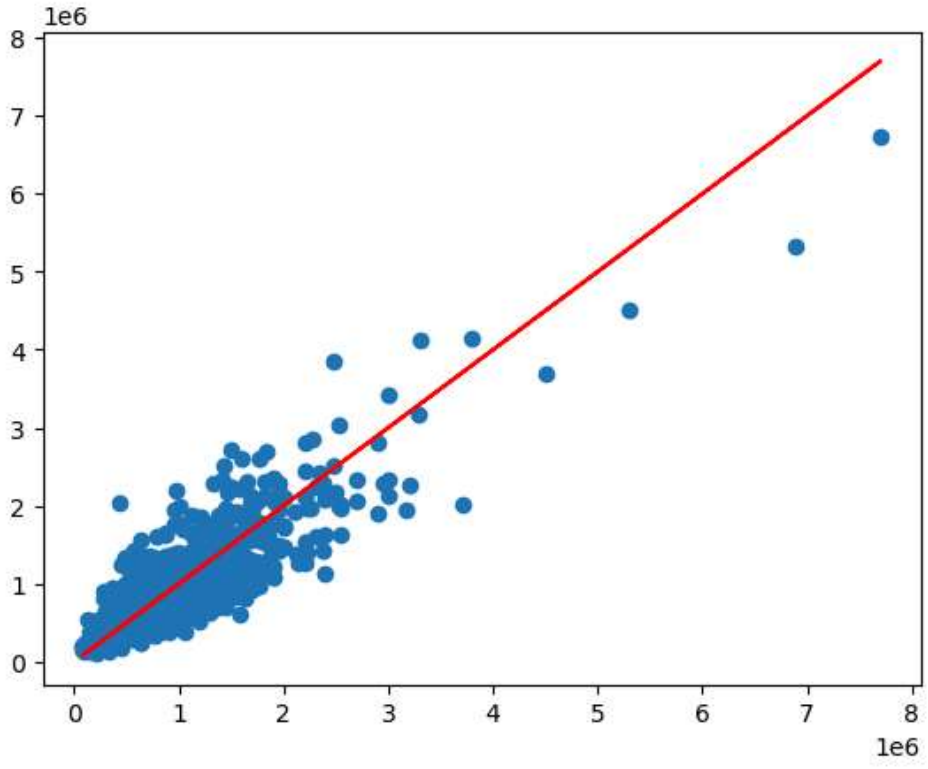
Out[66]:
```
540296.5735055795
```

In [67]:
```
df['price'].median()
```

Out[67]:
```
450000.0
```

In [68]:
```
# predictions
plt.scatter(y_test,predictions)

# perfect predictions
plt.plot(y_test,y_test,'r')
```

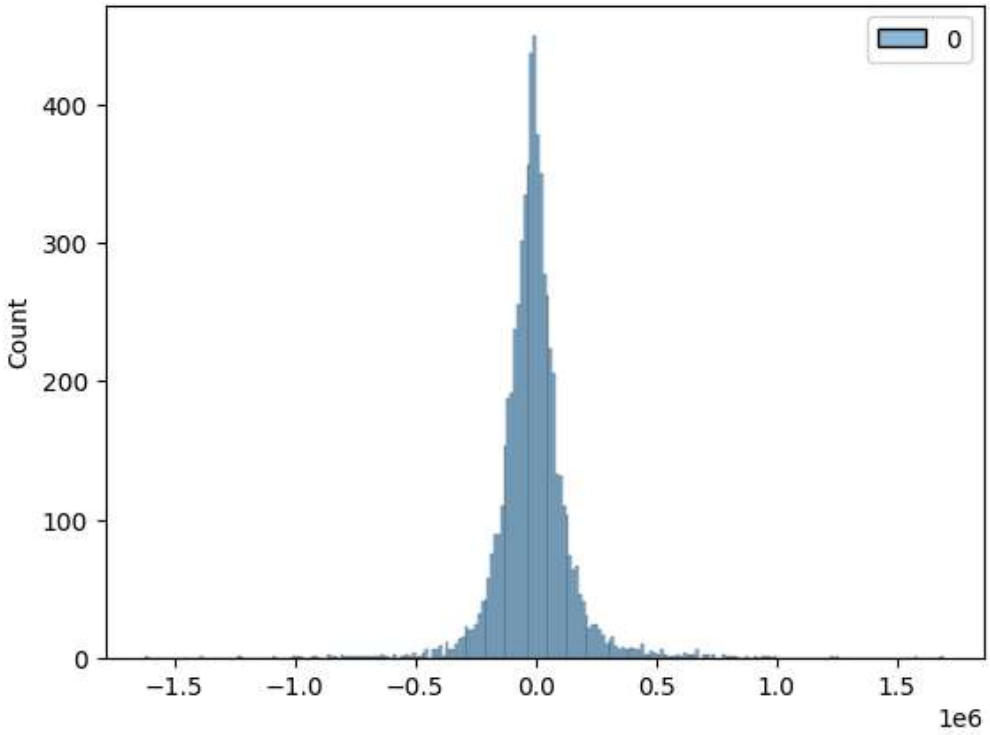
Out[68]:
```
[<matplotlib.lines.Line2D at 0x1b96cd6c050>]
```



In [69]:
```
errors = y_test.values.reshape(6480, 1) - predictions
```

In [72]:
```
sns.histplot(errors)
```

Out[72]: <Axes: ylabel='Count'>



Predicting on a new house

In [88]:
```
single_house = df2.drop('price',axis=1).iloc[0]
```

In [91]:
```
single_house = scaler.transform(single_house.values.reshape(-1, 20))
```

```
C:\Users\BALLA\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0
\LocalCache\local-packages\Python311\site-packages\sklearn\base.py:464: UserWarning: X do
es not have valid feature names, but MinMaxScaler was fitted with feature names
  warnings.warn(
```

In [92]:
```
single_house
```

Out[92]:
```
array([[0.2       , 0.08      , 0.08376422, 0.00310751, 0.        ,
        0.        , 0.        , 0.5       , 0.4       , 0.10785619,
        0.        , 0.47826087, 0.        , 0.89393939, 0.57149751,
        0.21760797, 0.16193426, 0.00582059, 0.        , 0.81818182]])
```

In [93]:
```
model.predict(single_house)
```

```
1/1 [==============================] - 0s 112ms/step
```

Out[93]:
```
array([[269511.16]], dtype=float32)
```

In [94]:
```
df2.iloc[0]
```

Out[94]:
```
price            221900.0000
bedrooms              3.0000
bathrooms             1.0000
sqft_living        1180.0000
sqft_lot           5650.0000
floors                1.0000
waterfront            0.0000
view                  0.0000
condition             3.0000
grade                 7.0000
sqft_above         1180.0000
sqft_basement         0.0000
yr_built           1955.0000
yr_renovated          0.0000
zipcode           98178.0000
lat                  47.5112
long               -122.2570
sqft_living15      1340.0000
sqft_lot15         5650.0000
year               2014.0000
month                10.0000
Name: 0, dtype: float64
```

THANKS