In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import mpl_toolkits
%matplotlib inline
```

In [2]:
```
df=pd.read_csv("kc_house_data (1).csv")
```

In [3]:
```
df.head()
```

Out[3]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	v
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	

5 rows × 21 columns

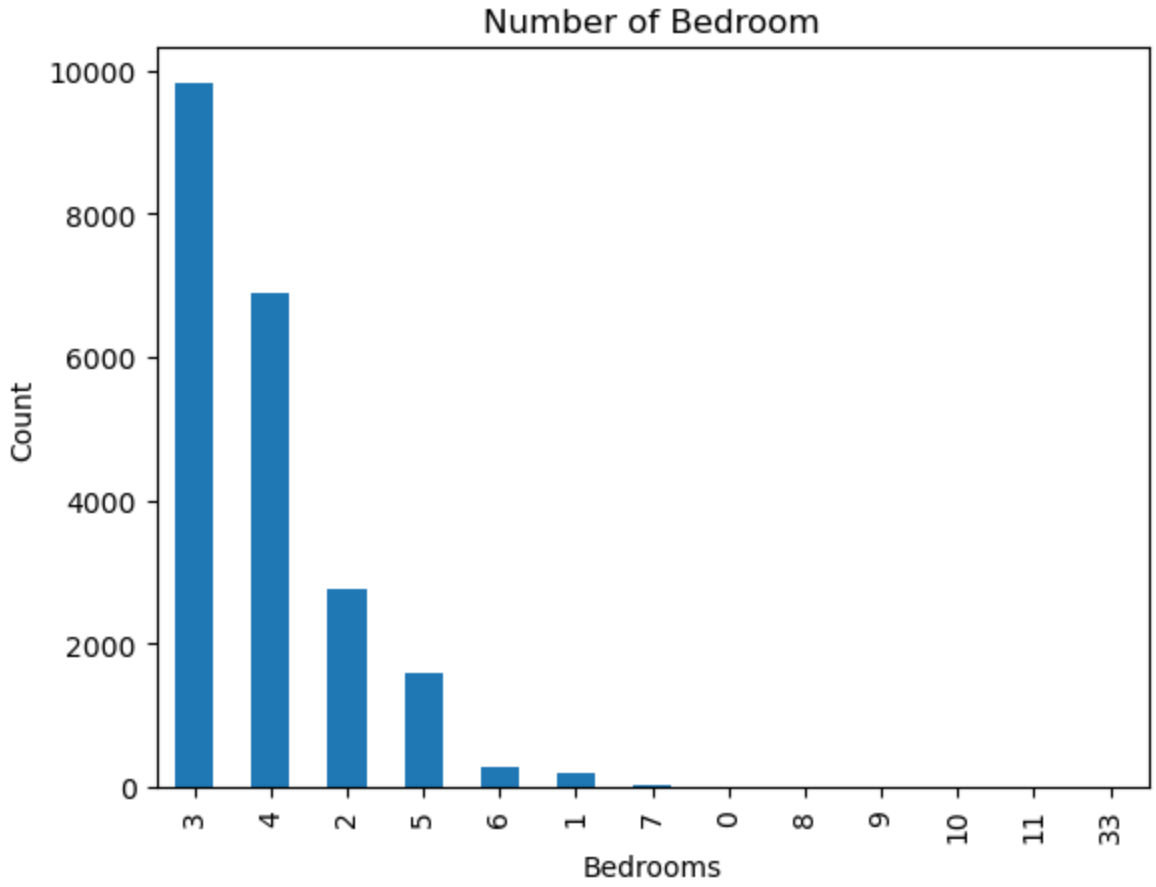
In [4]:
```
df.describe()
```

Out[4]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	21(
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	
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	

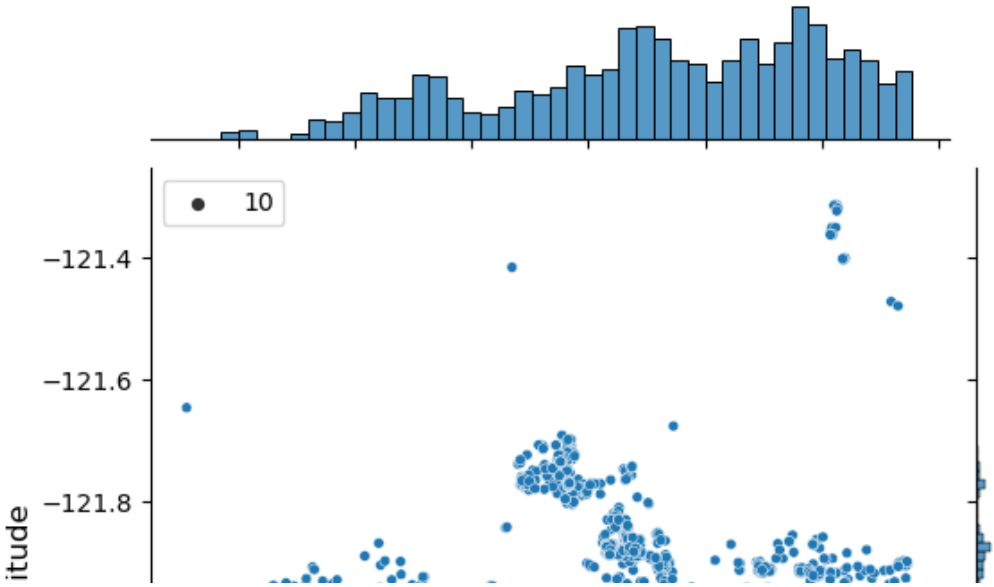
In [5]:
```
df['bedrooms'].value_counts().plot(kind='bar')
plt.title("Number of Bedroom")
plt.xlabel("Bedrooms")
plt.ylabel("Count")
sns.despine
```

Out[5]:
```
<function seaborn.utils.despine(fig=None, ax=None, top=True, right=True, left
=False, bottom=False, offset=None, trim=False)>
```



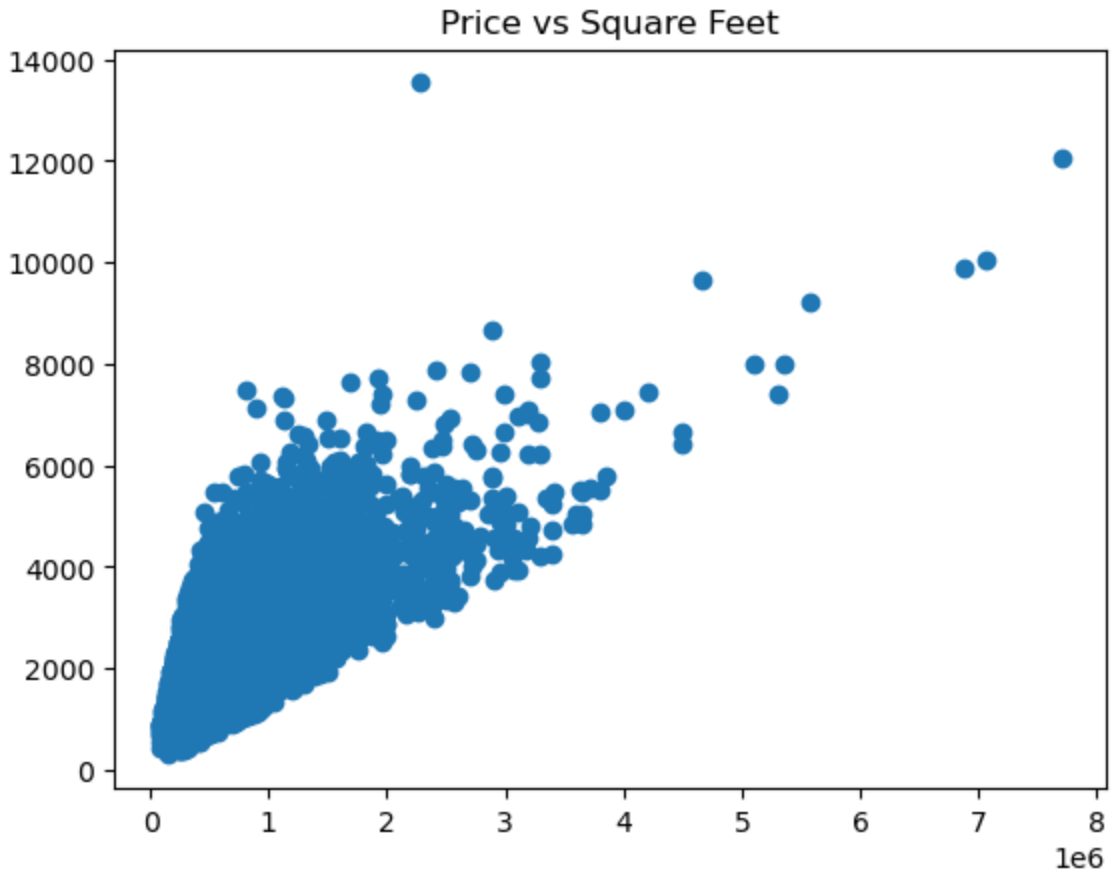
In [6]:
```
plt.figure(figsize=(10,10))
sns.jointplot(x=df.lat.values,y=df.long.values,size=10)
plt.ylabel('Longitude',fontsize=12)
plt.xlabel('Latitude', fontsize=12)
plt.show()
# plt1 = plt()
sns.despine
```

```
<Figure size 1000x1000 with 0 Axes>
```



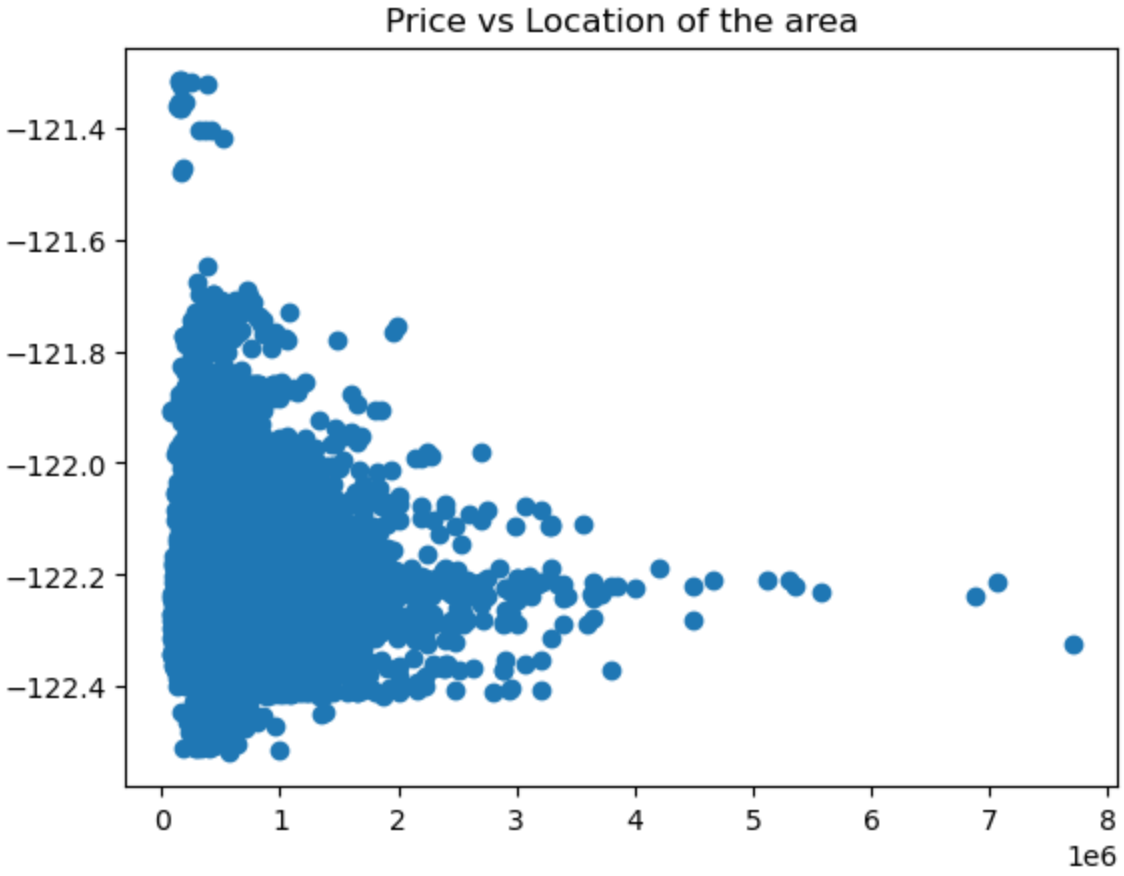
In [7]:
```
plt.scatter(df.price,df.sqft_living)
plt.title("Price vs Square Feet")
```

Out[7]:
```
Text(0.5, 1.0, 'Price vs Square Feet')
```



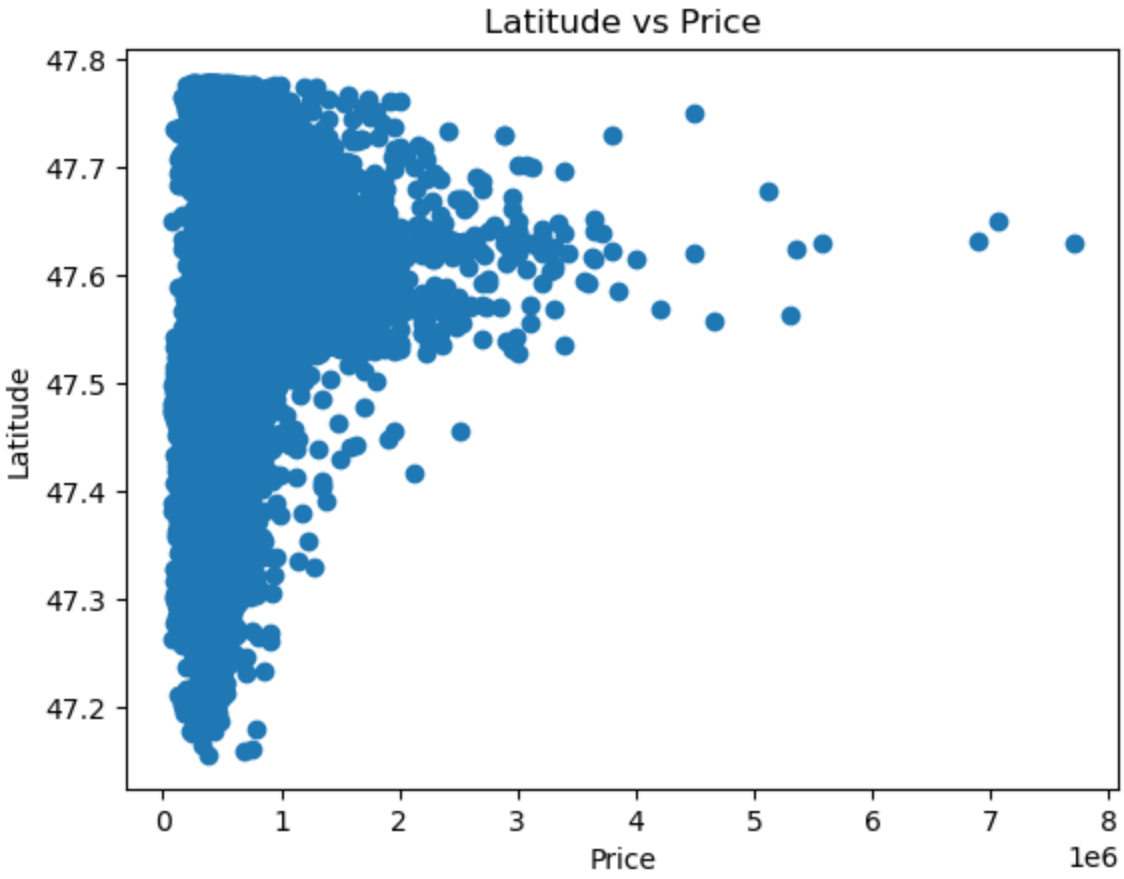
In [8]:
```
plt.scatter(df.price,df.long)
plt.title("Price vs Location of the area")
```

Out[8]:
```
Text(0.5, 1.0, 'Price vs Location of the area')
```

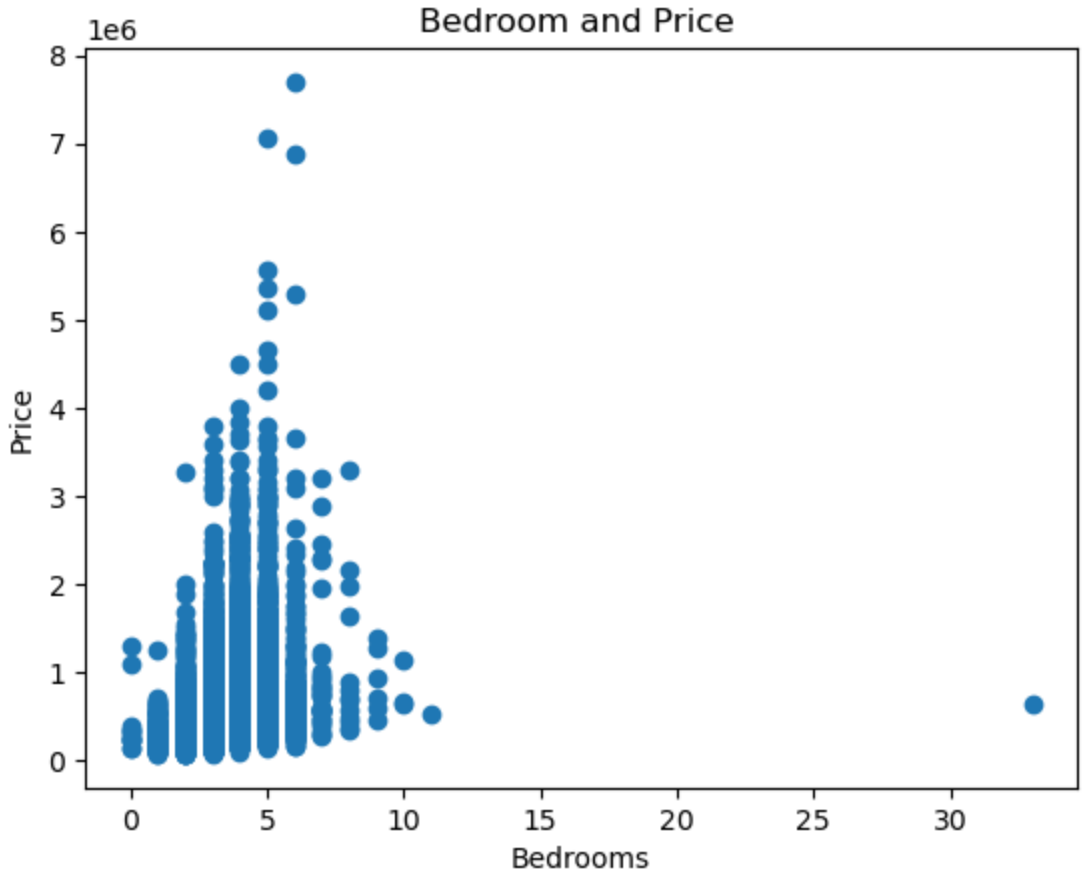


In [9]:
```
plt.scatter(df.price,df.lat)
plt.xlabel("Price")
plt.ylabel('Latitude')
plt.title("Latitude vs Price")
```

Out[9]:
```
Text(0.5, 1.0, 'Latitude vs Price')
```



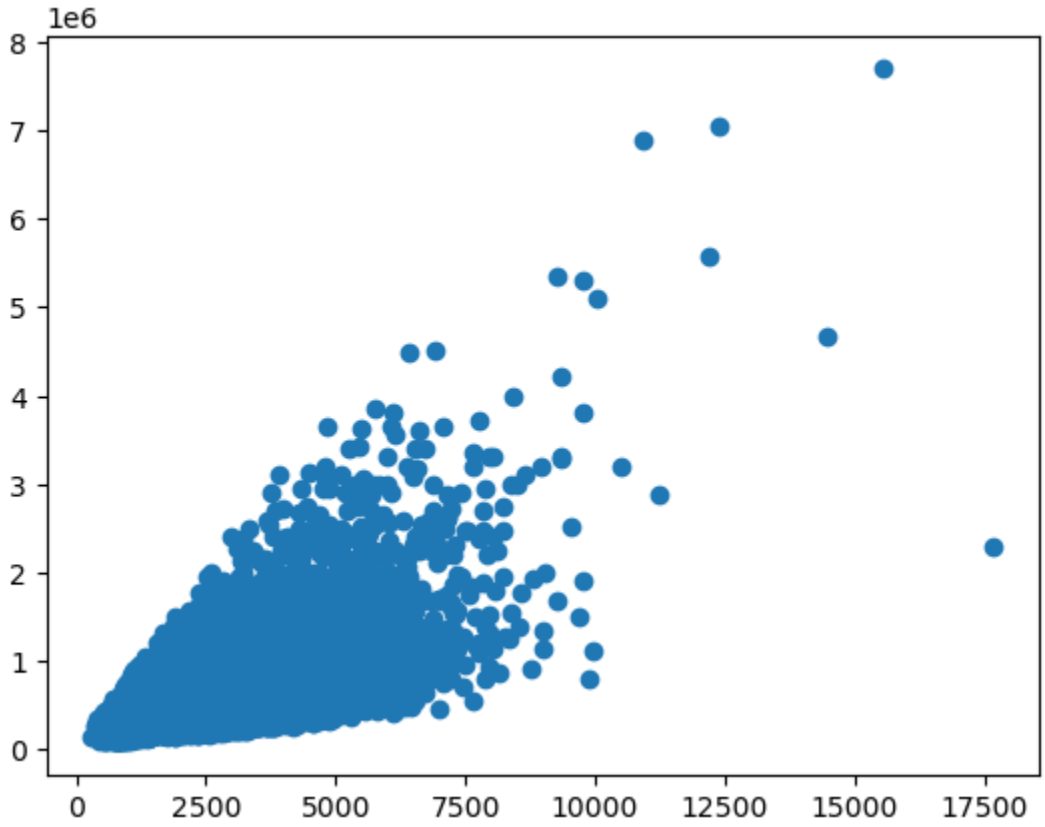
In [10]:
```
plt.scatter(df.bedrooms,df.price)
plt.title("Bedroom and Price ")
plt.xlabel("Bedrooms")
plt.ylabel("Price")
plt.show()
sns.despine
```



Out[10]:
```
<function seaborn.utils.despine(fig=None, ax=None, top=True, right=True, left
=False, bottom=False, offset=None, trim=False)>
```

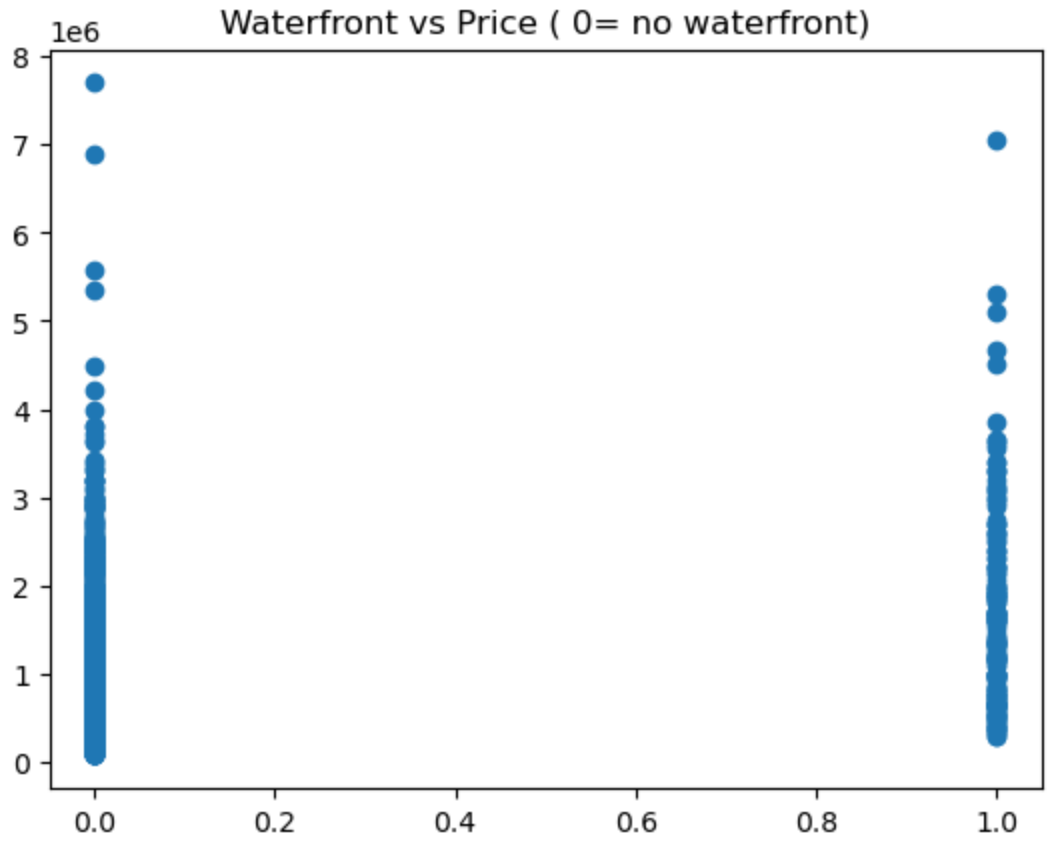
In [11]:
```
plt.scatter((df['sqft_living']+df['sqft_basement']),df['price'])
```

Out[11]:
```
<matplotlib.collections.PathCollection at 0x1f3dd2c6010>
```



In [12]:
```
plt.scatter(df.waterfront,df.price)
plt.title("Waterfront vs Price ( 0= no waterfront)")
```

Out[12]:
```
Text(0.5, 1.0, 'Waterfront vs Price ( 0= no waterfront)')
```



In [13]:
```
train1=df.drop(['id','price'],axis=1)
```

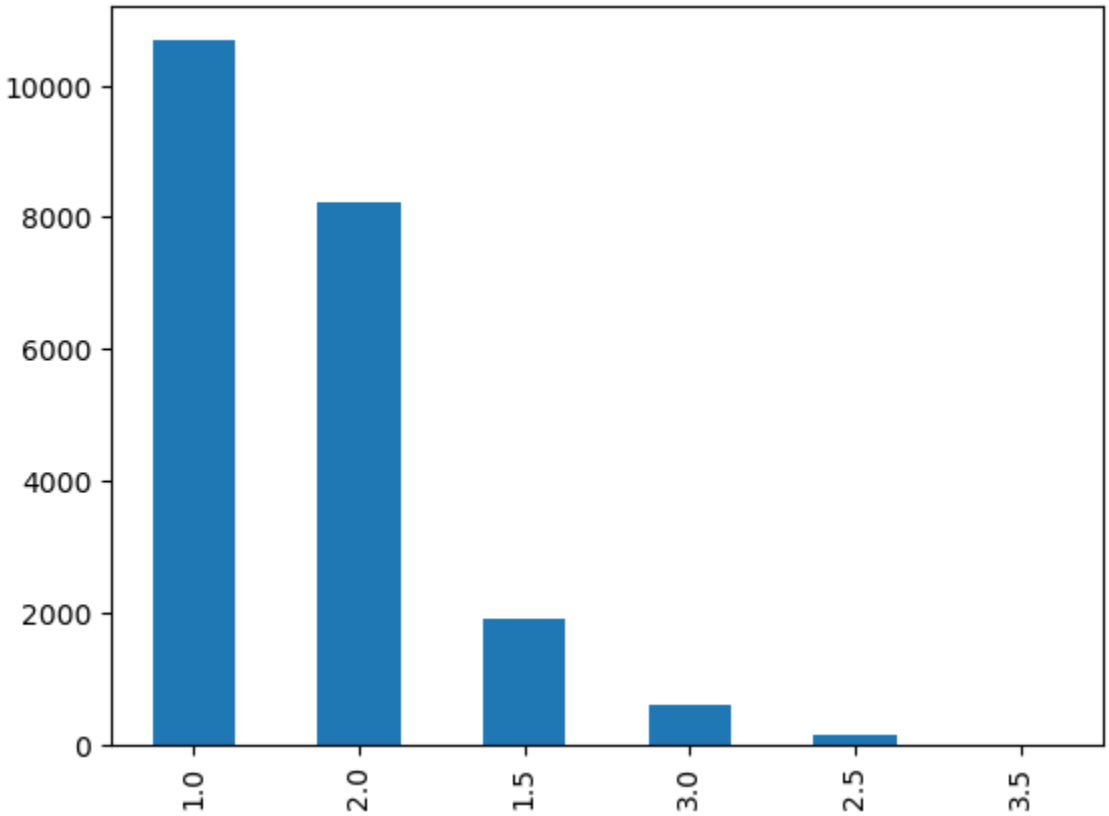
In [14]:
```
train1.head()
```

Out[14]:

	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condit
0	20141013T000000	3	1.00	1180	5650	1.0	0	0	
1	20141209T000000	3	2.25	2570	7242	2.0	0	0	
2	20150225T000000	2	1.00	770	10000	1.0	0	0	
3	20141209T000000	4	3.00	1960	5000	1.0	0	0	
4	20150218T000000	3	2.00	1680	8080	1.0	0	0	

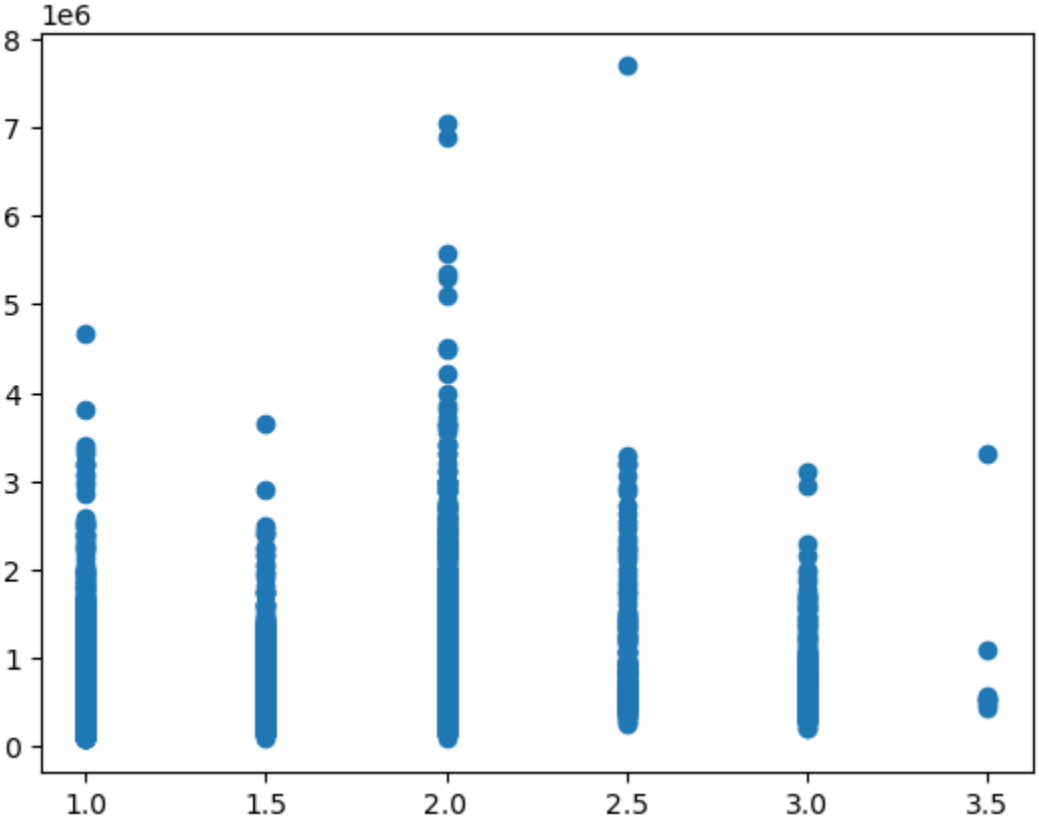
In [15]:
```
df.floors.value_counts().plot(kind='bar')
```

Out[15]: <Axes: >



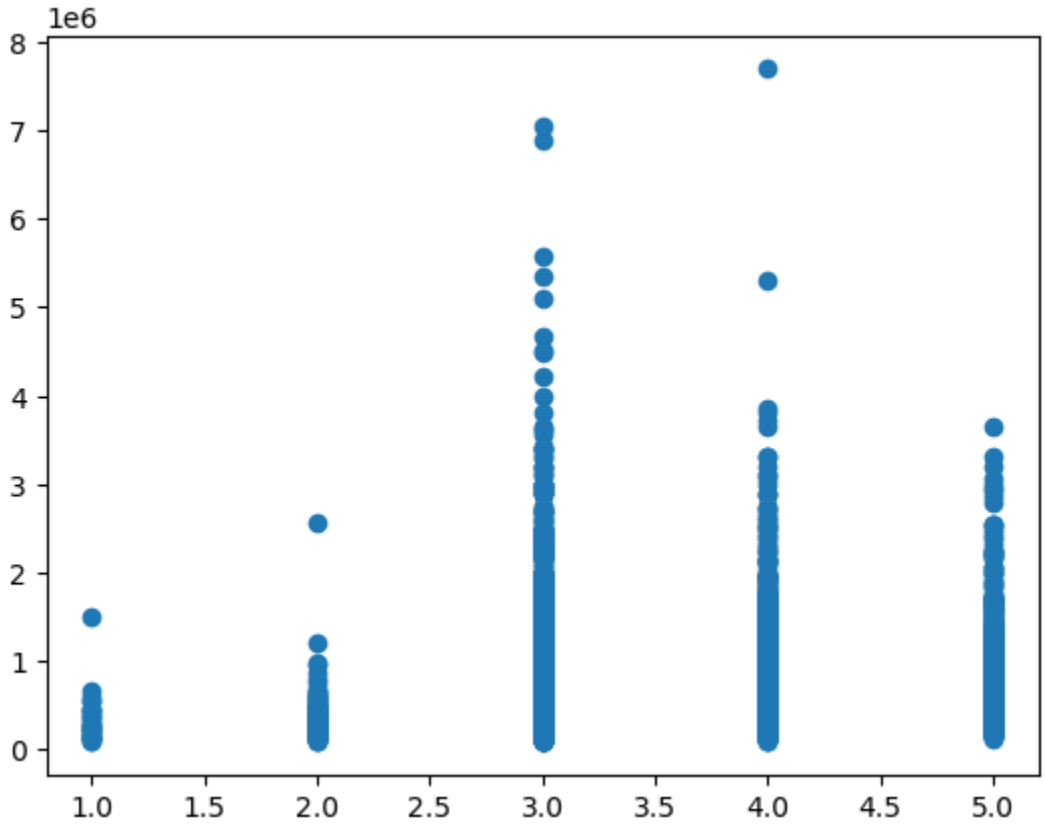
In [16]:
```
plt.scatter(df.floors,df.price)
```

Out[16]: <matplotlib.collections.PathCollection at 0x1f3dd6bdfd0>



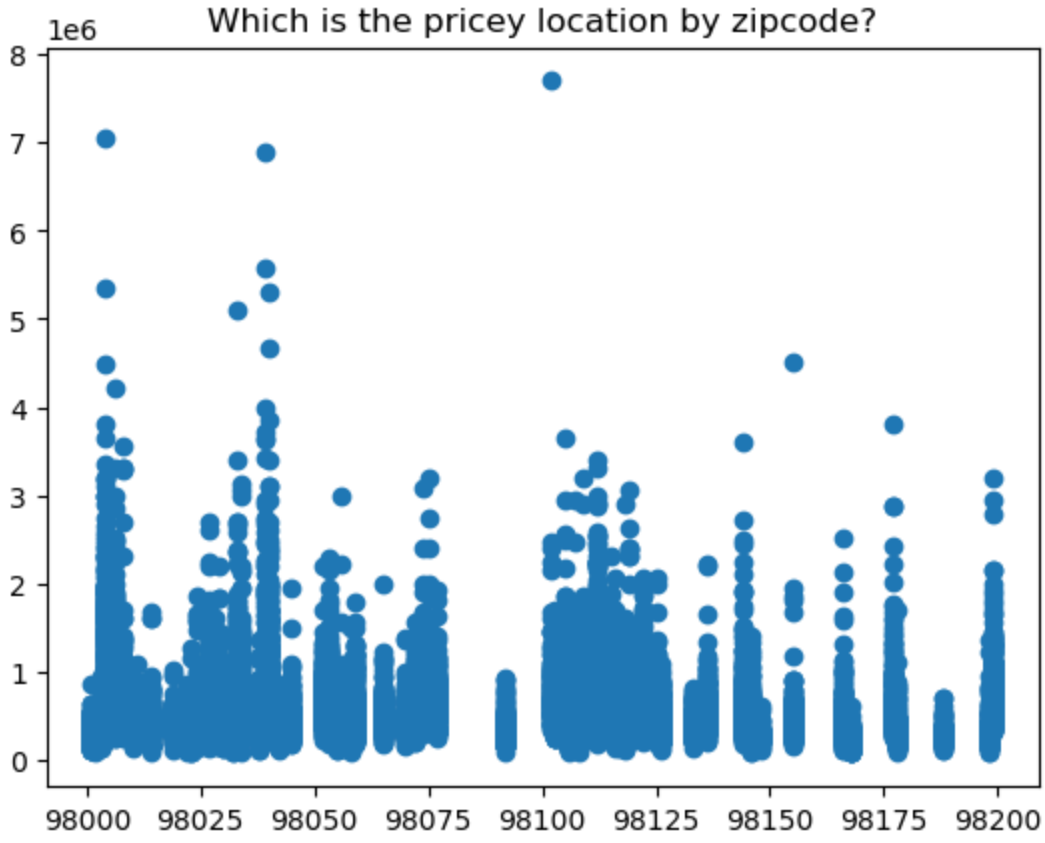
In [17]:
```
plt.scatter(df.condition,df.price)
```

Out[17]:
```
<matplotlib.collections.PathCollection at 0x1f3dd73d750>
```



In [18]:
```
plt.scatter(df.zipcode,df.price)
plt.title("Which is the pricey location by zipcode?")
```

Out[18]:
```
Text(0.5, 1.0, 'Which is the pricey location by zipcode?')
```



In [19]:
```
from sklearn.linear_model import LinearRegression
```

In [20]:
```
reg = LinearRegression()
```

In [21]:
```
labels=df['price']
conv_dates = [1 if values == 2014 else 0 for values in df.date ]
df['date'] = conv_dates
train1 = df.drop(['id', 'price'],axis=1)
```

In [22]:
```
from sklearn.model_selection import train_test_split
```

In [23]:
```
x_train , x_test , y_train , y_test = train_test_split(train1 , labels , test_
```

In [24]:
```
reg.fit(x_train,y_train)
```

Out[24]:
```
LinearRegression
LinearRegression()
```

In [25]:
```
from sklearn import ensemble
clf = ensemble.GradientBoostingRegressor(n_estimators = 400, max_depth = 5, mi
          learning_rate = 0.1)
```

In [26]:
```
clf.fit(x_train, y_train)
```

Out[26]:
```
GradientBoostingRegressor

GradientBoostingRegressor(max_depth=5, n_estimators=400)
```

In [27]:
```
clf.score(x_test,y_test)
```

Out[27]:
```
0.9193423659435747
```

In [28]:
```
params = {'n_estimators': 400, 'max_depth':5, 'min_samples_split':2, 'learning
```

In [29]:
```
t_sc = np.zeros((params['n_estimators']),dtype=np.float64)
```

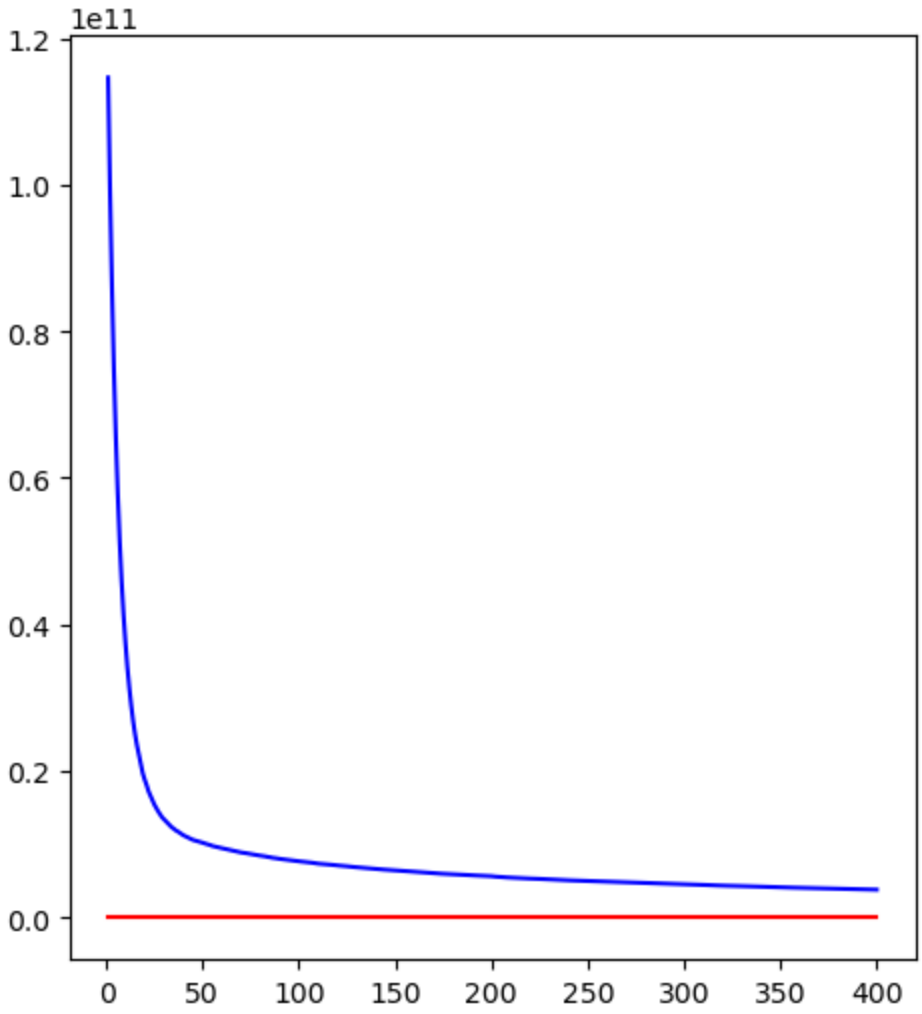
In [30]:
```
y_pred = reg.predict(x_test)
```

In [39]:
```
for i, y_pred in enumerate(clf.staged_predict(x_test)):
    t_sc[i] = clf.score(x_test, y_test)
```

In [40]:
```
testsc = np.arange((params['n_estimators']))+1
```

In [41]:
```
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(testsc,clf.train_score_,'b-',label= 'Set dev train')
plt.plot(testsc,t_sc,'r-',label = 'set dev test')
```

Out[41]:
```
[<matplotlib.lines.Line2D at 0x1f3e5ec40d0>]
```



In [42]:
```
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA
```

In [43]:
```
pca = PCA()
```

In [44]:
```
pca.fit_transform(scale(train1))
```

Out[44]:
```
array([[-2.64785461e+00, -4.54699955e-02, -3.16665762e-01, ...,
        -7.94687728e-02, -2.91214168e-16,  0.00000000e+00],
       [-2.34485164e-01,  1.68297114e+00, -7.61521725e-01, ...,
         9.81487761e-01,  2.63725032e-14,  0.00000000e+00],
       [-2.57007792e+00, -6.14344122e-01,  3.49292423e-01, ...,
        -1.38570764e-01,  3.28875263e-14,  0.00000000e+00],
       ...,
       [-2.41985641e+00, -1.10027662e+00, -1.46293798e+00, ...,
         9.66785881e-01,  8.38750844e-17, -0.00000000e+00],
       [ 3.32183025e-01, -1.88043103e+00, -1.04412760e+00, ...,
        -3.97449542e-01, -5.32337093e-17,  0.00000000e+00],
       [-2.43180432e+00, -1.08505981e+00, -1.47248379e+00, ...,
         9.53674385e-01,  8.34001402e-17, -0.00000000e+00]])
```

In [ ]: