

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

```
In [2]: data=pd.read_csv("C:/Users/91740/OneDrive/Desktop/SLASH MARKS INTERNSHIP/house-price-prediction-master/house-pr
data.head()
```

Out[2]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_ab
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1

5 rows × 21 columns

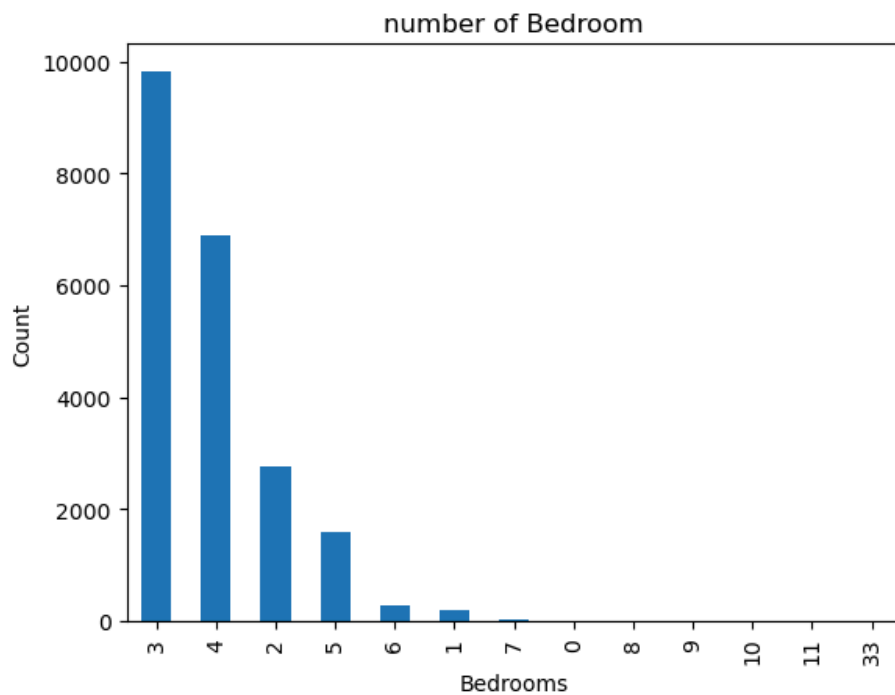
```
In [3]: data.describe()
```

Out[3]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	vi
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.0000
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494309	0.007542	0.2343
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	0.539989	0.086517	0.7663
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	1.000000	0.000000	0.0000
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	0.0000
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.0000
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	0.000000	0.0000
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.0000

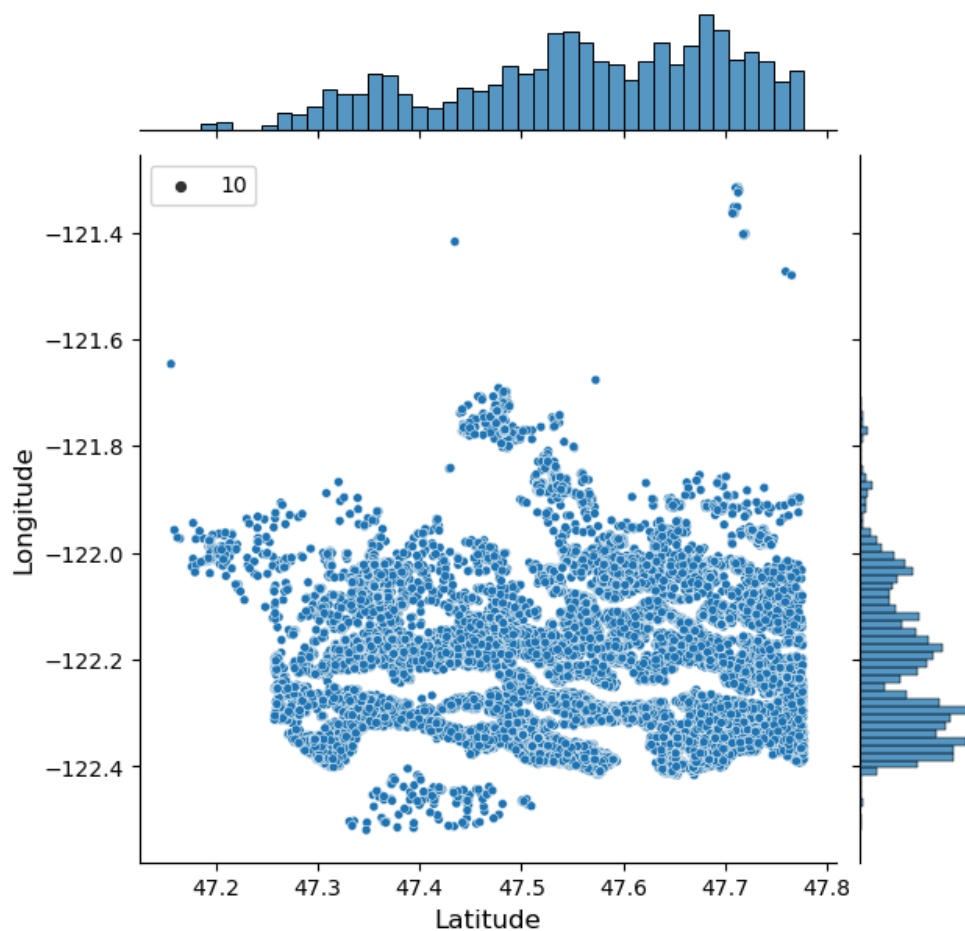
```
In [4]: data['bedrooms'].value_counts().plot(kind='bar')
plt.title('number of Bedroom')
plt.xlabel('Bedrooms')
plt.ylabel('Count')
sns.despine
```

```
Out[4]: <function seaborn.utils.despine(fig=None, ax=None, top=True, right=True, left=False, bottom=False, offset=None,
trim=False)>
```



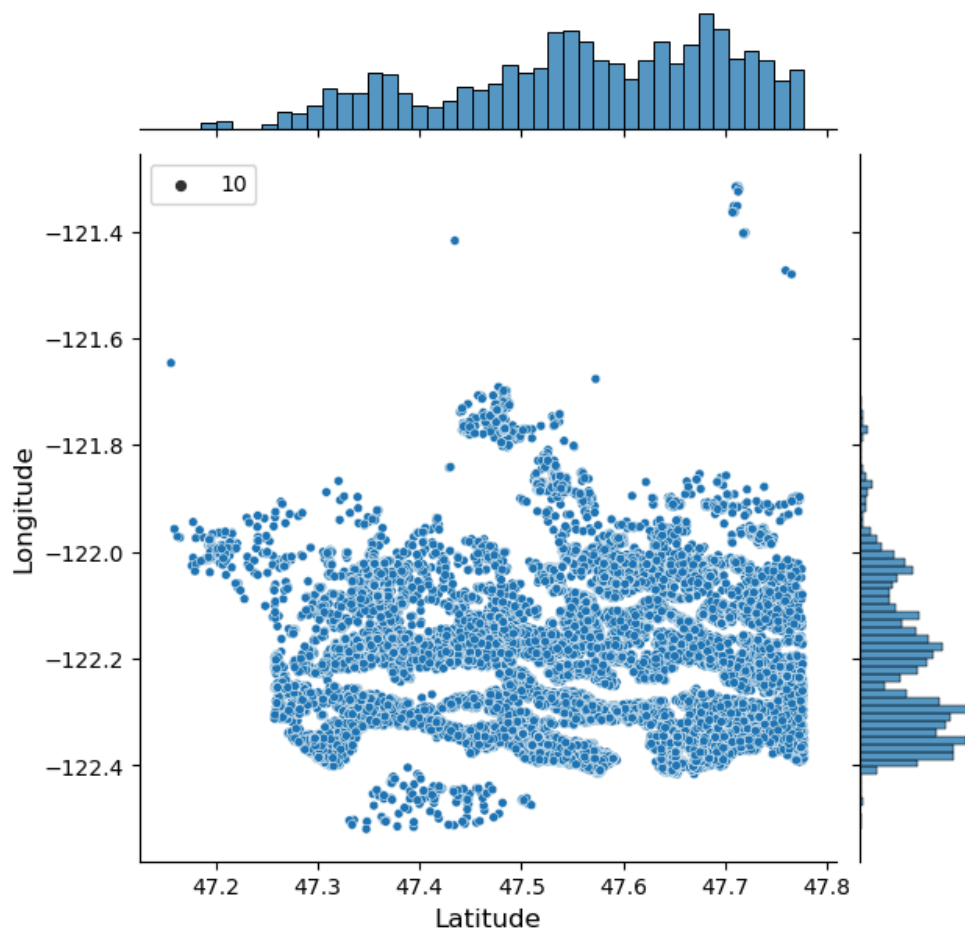
```
In [5]: plt.figure(figsize=(10,10))
sns.jointplot(x=data.lat.values, y=data.long.values, size=10)
plt.ylabel('Longitude', fontsize=12)
plt.xlabel('Latitude', fontsize=12)
plt.show()
```

<Figure size 1000x1000 with 0 Axes>



```
In [6]: plt.figure(figsize=(10,10))
sns.jointplot(x=data.lat.values, y=data.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>



```

-----
TypeError                                Traceback (most recent call last)
Cell In[6], line 6
      4 plt.xlabel('Latitude', fontsize=12)
      5 plt.show()
----> 6 plt1 = plt()
      7 sns.despine

TypeError: 'module' object is not callable

```

```

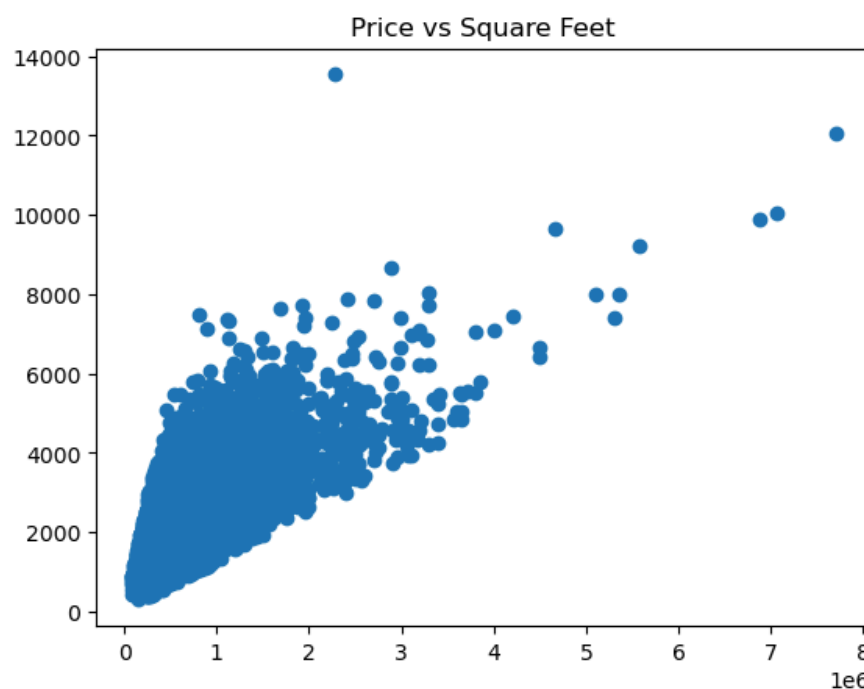
In [82]: plt.scatter(data.price, data.sqft_living)
         plt.title("Price vs Square Feet")

```

```

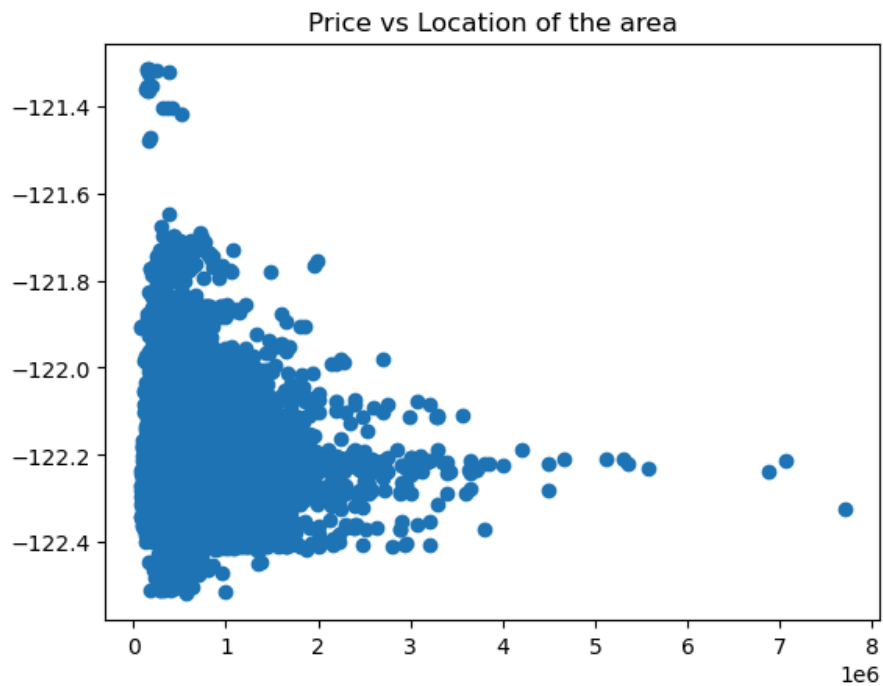
Out[82]: Text(0.5, 1.0, 'Price vs Square Feet')

```



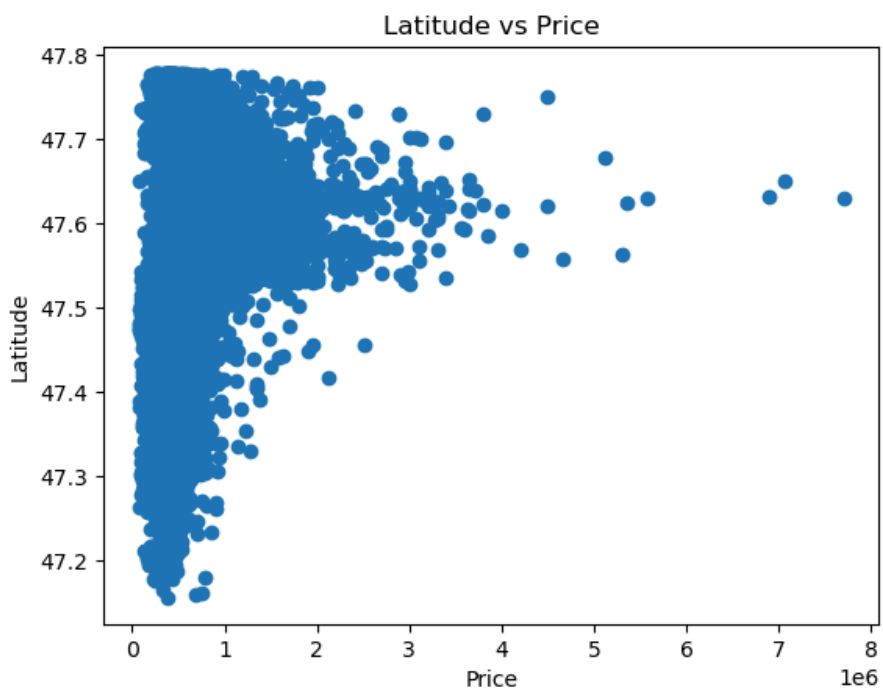
```
In [83]: plt.scatter(data.price,data.long)
plt.title("Price vs Location of the area")
```

```
Out[83]: Text(0.5, 1.0, 'Price vs Location of the area')
```

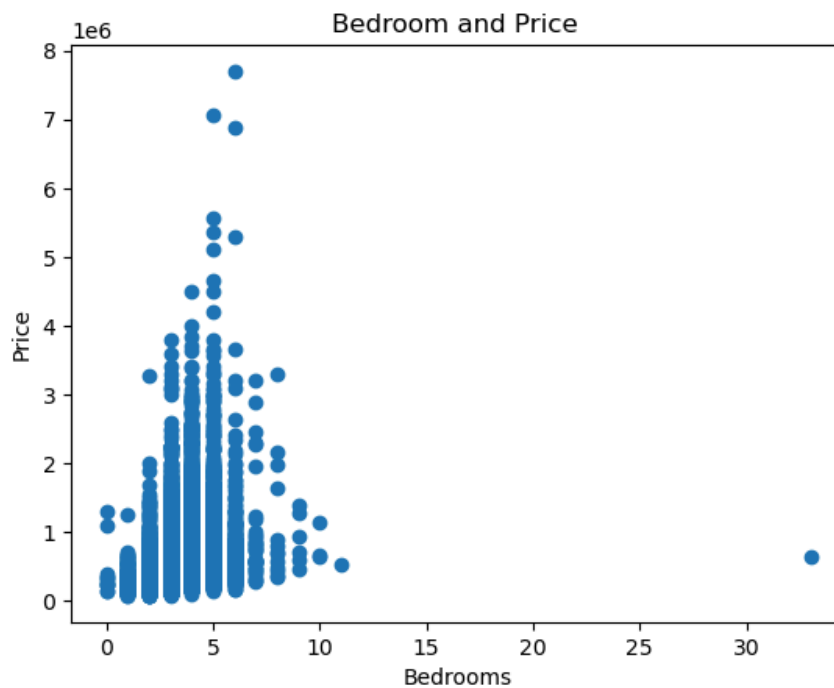


```
In [84]: plt.scatter(data.price,data.lat)
plt.xlabel("Price")
plt.ylabel('Latitude')
plt.title("Latitude vs Price")
```

```
Out[84]: Text(0.5, 1.0, 'Latitude vs Price')
```



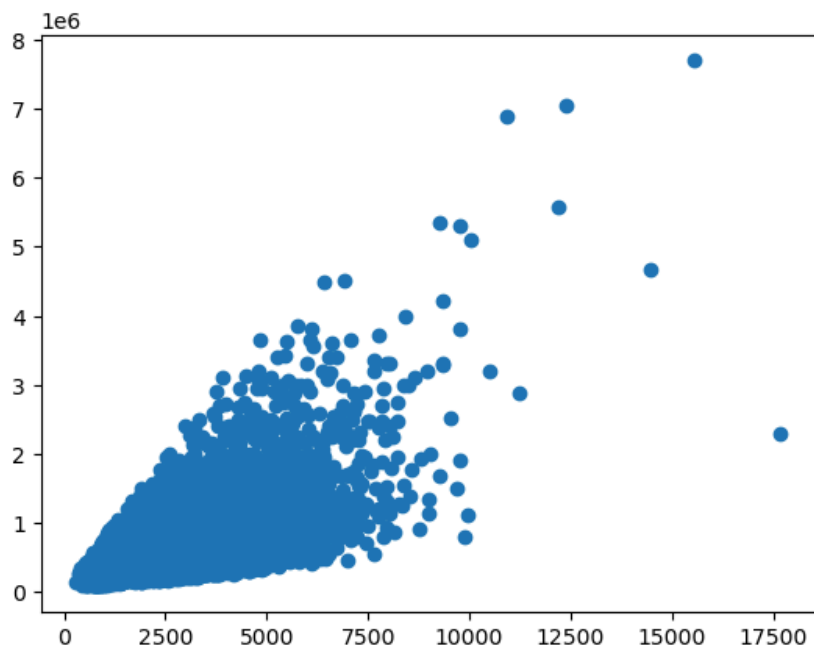
```
In [85]: plt.scatter(data.bedrooms,data.price)
plt.title("Bedroom and Price ")
plt.xlabel("Bedrooms")
plt.ylabel("Price")
plt.show()
sns.despine
```



Out[85]: <function seaborn.utils.despine(fig=None, ax=None, top=True, right=True, left=False, bottom=False, offset=None, trim=False)>

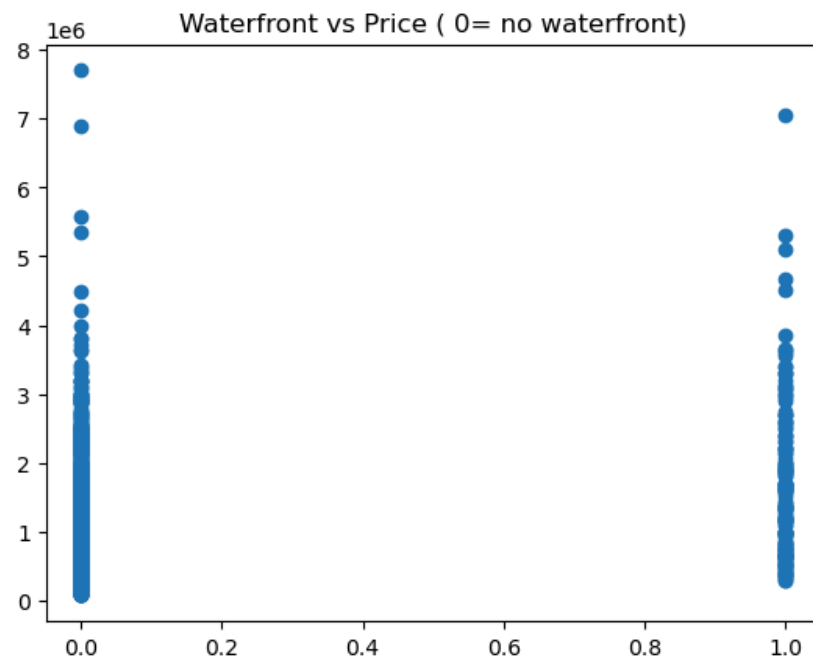
In [86]: `plt.scatter((data['sqft_living']+data['sqft_basement']),data['price'])`

Out[86]: <matplotlib.collections.PathCollection at 0x2dde06f8c90>



In [87]: `plt.scatter(data.waterfront,data.price)`
`plt.title("Waterfront vs Price (0= no waterfront)")`

Out[87]: Text(0.5, 1.0, 'Waterfront vs Price (0= no waterfront)')



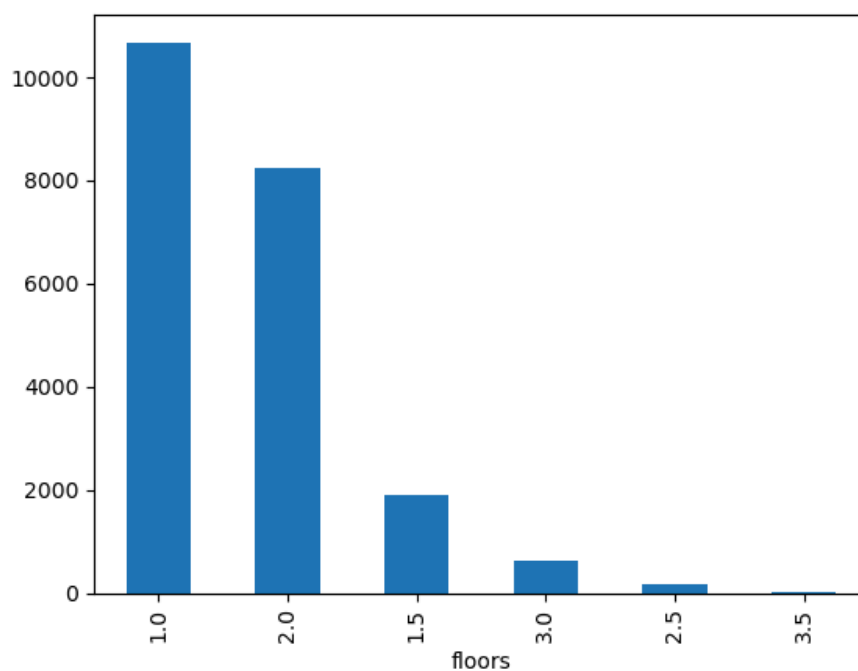
```
In [88]: train1=data.drop(['id','price'],axis=1)
         train1.head()
```

```
Out[88]:
```

	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basen
0	20141013T000000	3	1.00	1180	5650	1.0	0	0	3	7	1180	
1	20141209T000000	3	2.25	2570	7242	2.0	0	0	3	7	2170	
2	20150225T000000	2	1.00	770	10000	1.0	0	0	3	6	770	
3	20141209T000000	4	3.00	1960	5000	1.0	0	0	5	7	1050	
4	20150218T000000	3	2.00	1680	8080	1.0	0	0	3	8	1680	

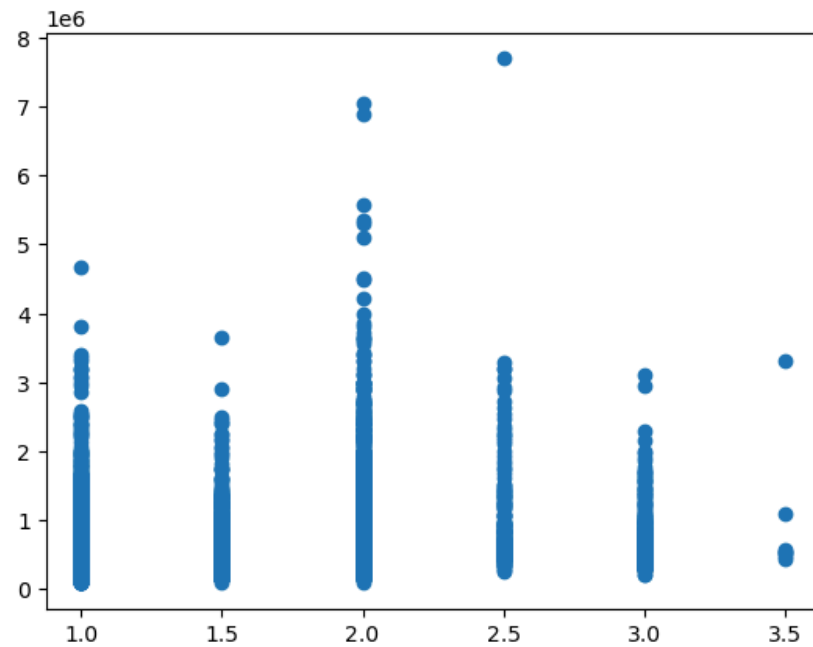
```
In [89]: data.floors.value_counts().plot(kind='bar')
```

```
Out[89]: <Axes: xlabel='floors'>
```



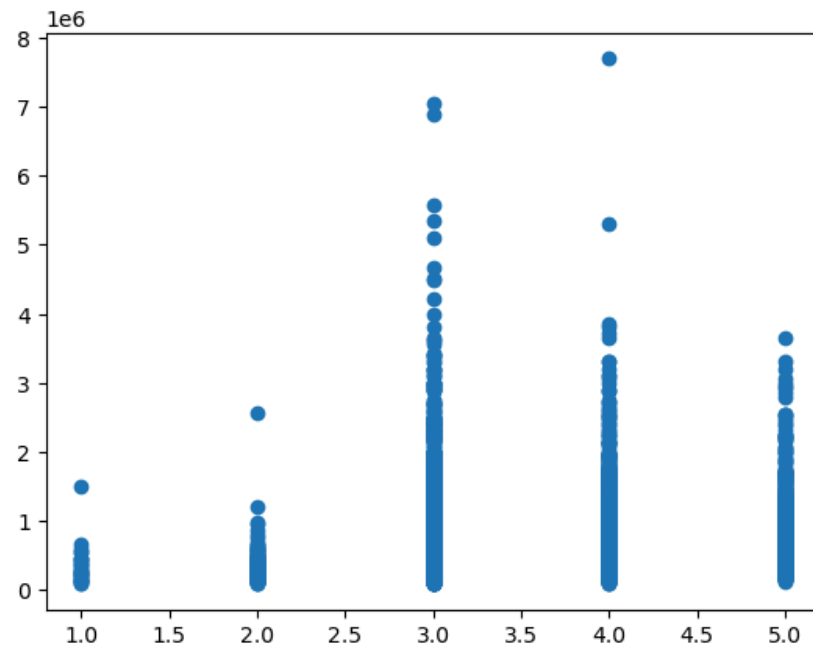
```
In [90]: plt.scatter(data.floors,data.price)
```

```
Out[90]: <matplotlib.collections.PathCollection at 0x2dde10b6190>
```



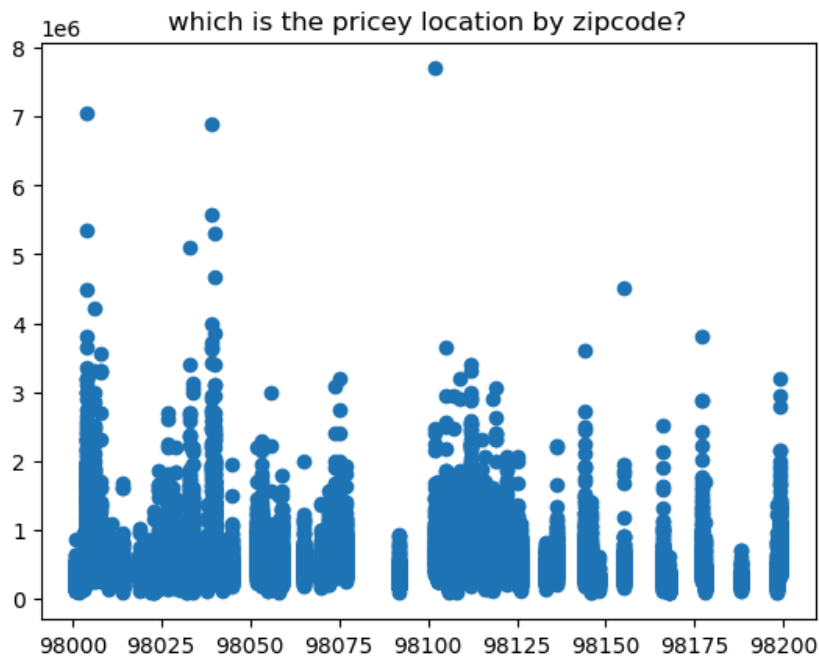
```
In [91]: plt.scatter(data.condition, data.price)
```

```
Out[91]: <matplotlib.collections.PathCollection at 0x2dde112c810>
```



```
In [92]: plt.scatter(data.zipcode, data.price)
plt.title("which is the pricey location by zipcode?")
```

```
Out[92]: Text(0.5, 1.0, 'which is the pricey location by zipcode?')
```



```
In [93]: from sklearn.linear_model import LinearRegression
reg=LinearRegression
```

```
In [94]: labels = data['price']
conv_dates = [1 if values == 2014 else 0 for values in data.date ]
data['date'] = conv_dates
train1 = data.drop(['id', 'price'],axis=1)
```

```
In [95]: from sklearn.model_selection import train_test_split
```

```
In [96]: X_train , X_test , y_train , y_test = train_test_split(train1 , labels , test_size = 0.10,random_state =2)
```

```
In [97]: from sklearn.linear_model import LinearRegression

# Assuming you have your training and testing data split
X_train, X_test, y_train, y_test = train_test_split(train1 , labels , test_size = 0.10,random_state =2)

# Create the regressor object
reg = LinearRegression()

# Train the regressor
reg.fit(X_train, y_train)
```

```
Out[97]: LinearRegression
LinearRegression()
```

```
In [98]: from sklearn.linear_model import LinearRegression

# Assuming you have your training and testing data split
X_train, X_test, y_train, y_test = train_test_split(train1 , labels , test_size = 0.10,random_state =2)

# Create the regressor object
reg = LinearRegression()

# Train the regressor
reg.fit(X_train, y_train)

# Make predictions on the test set
y_pred = reg.predict(X_test)

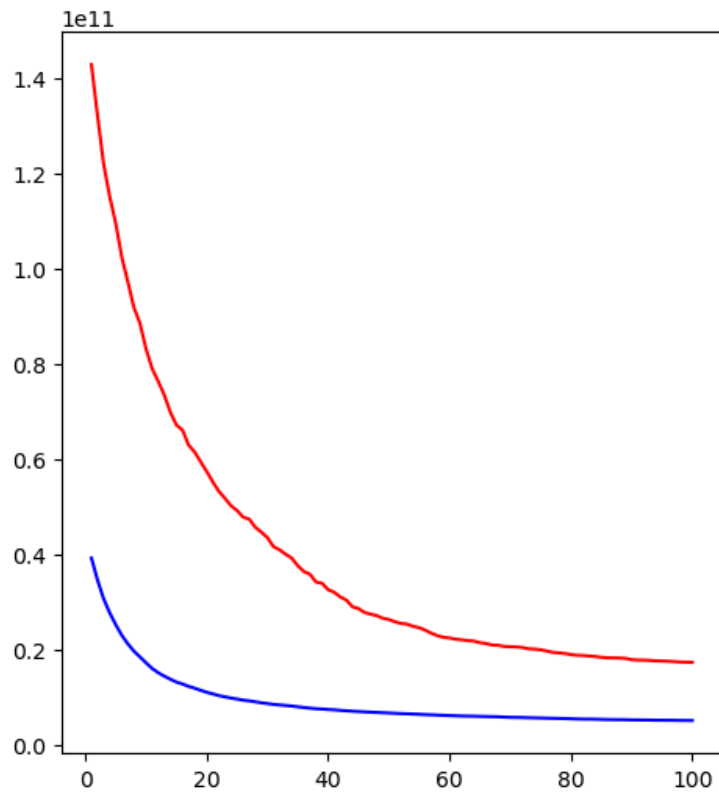
# Evaluate the performance using score (higher is better for regression)
score = reg.score(X_test, y_test)
print("R-squared score:", score)
```

R-squared score: 0.7320342760357297

```
In [99]: from sklearn import ensemble
clf = ensemble.GradientBoostingRegressor(n_estimators = 400, max_depth = 5, min_samples_split = 2,
learning_rate = 0.1, loss = 'huber')
```



```
Out[110]: []
```



```
In [111]: from sklearn.preprocessing import scale
          from sklearn.decomposition import PCA
```

```
In [112]: pca = PCA()
```

```
In [113]: pca
```

```
Out[113]: PCA
PCA()
```

```
In [114]: pca.fit_transform(scale(train1))
```

```
Out[114]: array([[ -2.64785461e+00,  -4.54699955e-02,  -3.16665762e-01,  ...,
        -7.94687728e-02,   2.59674388e-17,   0.00000000e+00],
        [ -2.34485164e-01,   1.68297114e+00,  -7.61521725e-01,  ...,
         9.81487761e-01,   2.80892751e-14,  -0.00000000e+00],
        [ -2.57007792e+00,  -6.14344122e-01,   3.49292423e-01,  ...,
        -1.38570764e-01,   3.09103597e-14,  -0.00000000e+00],
        ...,
        [ -2.41985641e+00,  -1.10027662e+00,  -1.46293798e+00,  ...,
         9.66785881e-01,   1.09776500e-16,  -0.00000000e+00],
        [  3.32183025e-01,  -1.88043103e+00,  -1.04412760e+00,  ...,
        -3.97449542e-01,   2.89754352e-17,   0.00000000e+00],
        [ -2.43180432e+00,  -1.08505981e+00,  -1.47248379e+00,  ...,
         9.53674385e-01,   2.73549743e-17,  -0.00000000e+00]])
```

```
In [ ]:
```