[75]
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import
StandardScaler,PolynomialFeatures

# Read the csv file in
df = pd.read_csv(r'C:\Users\maniv\Documents\housing data.csv')

# Save to file
df.to_html('myTable.htm')

# Assign to string
htmTable = df.to_html()

print(df.to_html)
```

```
<bound method DataFrame.to_html of                 id             date
price  bedrooms  bathrooms  \
0      7129300520  20141013T000000  221900.0         3       1.00
1      6414100192  20141209T000000  538000.0         3       2.25
2      5631500400  20150225T000000  180000.0         2       1.00
3      2487200875  20141209T000000  604000.0         4       3.00
4      1954400510  20150218T000000  510000.0         3       2.00
...           ...              ...       ...       ...        ...
21608   263000018  20140521T000000  360000.0         3       2.50
21609  6600060120  20150223T000000  400000.0         4       2.50
21610  1523300141  20140623T000000  402101.0         2       0.75

21611   291310100  20150116T000000  400000.0         3       2.50
21612  1523300157  20141015T000000  325000.0         2       0.75

       sqft_living  sqft_lot  floors  waterfront  view  ...  Unnamed: 21
\
0             1180      5650     1.0           0     0  ...          N...
1             2570      7242     2.0           0     0  ...          N...
2              770     10000     1.0           0     0  ...          N...
3             1960      5000     1.0           0     0  ...          N...
4             1680      8080     1.0           0     0  ...          N...
...            ...       ...     ...         ...   ...  ...          ....
21608         1530      1131     3.0           0     0  ...          N...
21609         2310      5813     2.0           0     0  ...          N...
21610         1020      1350     2.0           0     0  ...          N...
21611         1600      2388     2.0           0     0  ...          N...
21612         1020      1076     2.0           0     0  ...          NaN

       Unnamed: 22  Unnamed: 23  Unnamed: 24  Unnamed: 25  Unnamed: 26
\
```

```
0              NaN           NaN           NaN           NaN           NaN...
1              NaN           NaN           NaN           NaN           NaN...
2              NaN           NaN           NaN           NaN           NaN...
3              NaN           NaN           NaN           NaN           NaN...
4              NaN           NaN           NaN           NaN           NaN...
...            ...           ...           ...           ...           .....
21608          NaN           NaN           NaN           NaN           NaN...
21609          NaN           NaN           NaN           NaN           NaN...
21610          NaN           NaN           NaN           NaN           NaN...
21611          NaN           NaN           NaN           NaN           NaN...
21612          NaN           NaN           NaN           NaN           NaN...

       Unnamed: 27  Unnamed: 28  Unnamed: 29  Unnamed: 30
0              NaN           NaN           NaN           NaN
1              NaN           NaN           NaN           NaN
2              NaN           NaN           NaN           NaN
3              NaN           NaN           NaN           NaN
4              NaN           NaN           NaN           NaN
...            ...           ...           ...           ...
21608          NaN           NaN           NaN           NaN
21609          NaN           NaN           NaN           NaN
21610          NaN           NaN           NaN           NaN
21611          NaN           NaN           NaN           NaN
21612          NaN           NaN           NaN           NaN

[21613 rows x 31 columns]>
```

[76]
```
df.head()
```

	id	date	price	bedrooms	bathrooms
0	7129300520	20141013T000000	221900.0	3	1.00
1	6414100192	20141209T000000	538000.0	3	2.25
2	5631500400	20150225T000000	180000.0	2	1.00
3	2487200875	20141209T000000	604000.0	4	3.00
4	1954400510	20150218T000000	510000.0	3	2.00

5 rows × 31 columns

[77]
```
df.tail()
```

	id	date	price	bedrooms	bathroom
21608	263000018	20140521T000000	360000.0	3	2.50
21609	6600060120	20150223T000000	400000.0	4	2.50
21610	1523300141	20140623T000000	402101.0	2	0.75
21611	291310100	20150116T000000	400000.0	3	2.50
21612	1523300157	20141015T000000	325000.0	2	0.75

5 rows × 31 columns

[78]
```
df.dtypes
```

```
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
Unnamed: 21     float64
Unnamed: 22     float64
Unnamed: 23     float64
Unnamed: 24     float64
Unnamed: 25     float64
Unnamed: 26     float64
Unnamed: 27     float64
Unnamed: 28     float64
Unnamed: 29     float64
Unnamed: 30     float64
dtype: object
```

[91]
```
df.describe()
```

	price	bedrooms	bathrooms	sqft_living	
count	2.161300e+04	21613.000000	21613.000000	21613.000000	2.16
mean	5.401822e+05	3.370842	2.114757	2079.899736	1.51
std	3.673622e+05	0.930062	0.770163	918.440897	4.14
min	7.500000e+04	0.000000	0.000000	290.000000	5.20
25%	3.219500e+05	3.000000	1.750000	1427.000000	5.04
50%	4.500000e+05	3.000000	2.250000	1910.000000	7.61
75%	6.450000e+05	4.000000	2.500000	2550.000000	1.06
max	7.700000e+06	33.000000	8.000000	13540.000000	1.65

8 rows × 29 columns

[100]
```
df.set_index('id', drop=False, inplace=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-100-f44fec969864> in <module>
----> 1 df.set_index('id', drop=False, inplace=True)

~\Anaconda3\lib\site-packages\pandas\core\frame.py in set_index(self, keys, drop, append, inplace, verify_integrity)
   4394
   4395         if missing:
-> 4396             raise KeyError("None of {} are in the columns".format(missing))
   4397
   4398         if inplace:

KeyError: "None of ['id'] are in the columns"
```

[101]
```
df
```

	date	price	bedrooms	bathrooms	sqft_livin

	date	price	bedrooms	bathrooms	sqft_livin
0	20141013T000000	221900.0	3	1.00	1180
1	20141209T000000	538000.0	3	2.25	2570
2	20150225T000000	180000.0	2	1.00	770
3	20141209T000000	604000.0	4	3.00	1960
4	20150218T000000	510000.0	3	2.00	1680
...	...	...	...	...	...
21608	20140521T000000	360000.0	3	2.50	1530
21609	20150223T000000	400000.0	4	2.50	2310
21610	20140623T000000	402101.0	2	0.75	1020
21611	20150116T000000	400000.0	3	2.50	1600
21612	20141015T000000	325000.0	2	0.75	1020

21613 rows × 30 columns

[102]
```
df.set_index('Unnamed', drop=False, inplace=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-102-ca4382f69c04> in <module>
----> 1 df.set_index('Unnamed', drop=False, inplace=True)

~\Anaconda3\lib\site-packages\pandas\core\frame.py in set_index(self, keys, drop, append, inplace, verify_integrity)
   4394
   4395         if missing:
-> 4396             raise KeyError("None of {} are in the columns".format(missing))
   4397
   4398         if inplace:

KeyError: "None of ['Unnamed'] are in the columns"
```

[103]
```
df
```

	date	price	bedrooms	bathrooms	sqft_livin
0	20141013T000000	221900.0	3	1.00	1180
1	20141209T000000	538000.0	3	2.25	2570
2	20150225T000000	180000.0	2	1.00	770
3	20141209T000000	604000.0	4	3.00	1960
4	20150218T000000	510000.0	3	2.00	1680
...	...	...	...	...	...
21608	20140521T000000	360000.0	3	2.50	1530
21609	20150223T000000	400000.0	4	2.50	2310
21610	20140623T000000	402101.0	2	0.75	1020
21611	20150116T000000	400000.0	3	2.50	1600
21612	20141015T000000	325000.0	2	0.75	1020

21613 rows × 30 columns

[137]
```
# pandas drop columns using list of column names
data = df.drop(['Unnamed: 21', 'Unnamed: 22', 'Unnamed: 23',
'Unnamed: 24', 'Unnamed: 25', 'Unnamed: 26', 'Unnamed:
27','Unnamed: 28', 'Unnamed: 29','Unnamed: 30', 'ColumnA'],
axis=1)
```

[139]
```
data.describe()
```

	price	bedrooms	bathrooms	sqft_living	
count	2.161300e+04	21613.000000	21613.000000	21613.000000	2.16
mean	5.401822e+05	3.370842	2.114757	2079.899736	1.51
std	3.673622e+05	0.930062	0.770163	918.440897	4.14
min	7.500000e+04	0.000000	0.000000	290.000000	5.20
25%	3.219500e+05	3.000000	1.750000	1427.000000	5.04
50%	4.500000e+05	3.000000	2.250000	1910.000000	7.61
75%	6.450000e+05	4.000000	2.500000	2550.000000	1.06

	price	bedrooms	bathrooms	sqft_living	
max	7.700000e+06	33.000000	8.000000	13540.000000	1.65

[140]
```
print("number of NaN values for the column bedrooms :",
df['bedrooms'].isnull().sum())
print("number of NaN values for the column bathrooms :",
df['bathrooms'].isnull().sum())
```

```
number of NaN values for the column bedrooms : 0
number of NaN values for the column bathrooms : 0
```

[141]
```
print("number of NaN values for the column yr_built :",
df['yr_built'].isnull().sum())
print("number of NaN values for the column yr_renovated :",
df['yr_renovated'].isnull().sum())
```

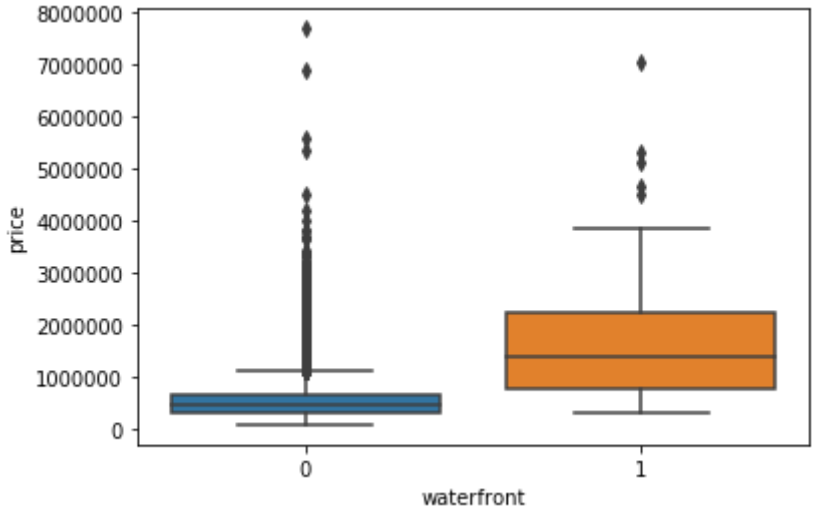
```
number of NaN values for the column yr_built : 0
number of NaN values for the column yr_renovated : 0
```

[142]
```
df['floors'].value_counts().to_frame()
```

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

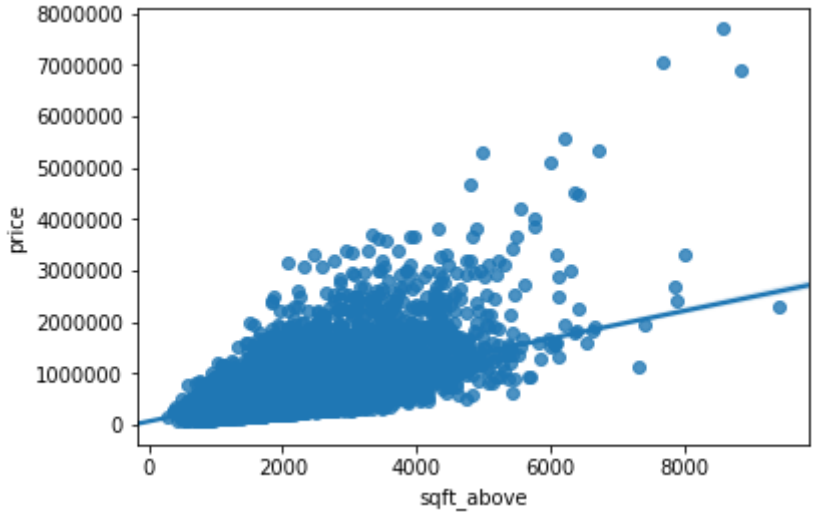
[144]
```
sns.boxplot(x="waterfront", y="price", data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x13bb0740278>
```



[146]
```
sns.regplot(x="sqft_above", y="price", data=df, x_jitter=.05)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x13baf1e6438>
```



[148]
```
X1 = df[['sqft_living']]
Y1 = df['price']
```

[149]
```
print(X1)
```

```
       sqft_living
0             1180
1             2570
2              770
3             1960
4             1680
...            ...
21608         1530
```

```
21609         2310
21610         1020
21611         1600
21612         1020

[21613 rows x 1 columns]
```

[150]
```
print(Y1)
```

```
0        221900.0
1        538000.0
2        180000.0
3        604000.0
4        510000.0
           ...
21608    360000.0
21609    400000.0
21610    402101.0
21611    400000.0
21612    325000.0
Name: price, Length: 21613, dtype: float64
```

[151]
```
from sklearn.linear_model import LinearRegression
```

[152]
```
model = LinearRegression()
```

[153]
```
model.fit(X1,Y1)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
normalize=False)
```

[154]
```
r_sq= model.score(X1,Y1)
```

[155]
```
print("Coefficient of determination", r_sq)
```

```
Coefficient of determination 0.49286538652201417
```

[156]
```
features= ["sqft_above", "sqft_basement", "sqft_living15",
"waterfront", "sqft_living", "view", "floors", "grade", "lat",
```

```
"bathrooms", "bedrooms"]
```

[158]
```
X2= df[features]
Y2= df['price']
```

[159]
```
print(X2)
```

```
       sqft_above  sqft_basement  sqft_living15  waterfront  sqft_living
\
0            1180              0           1340           0         11...
1            2170            400           1690           0         25...
2             770              0           2720           0          7...
3            1050            910           1360           0         19...
4            1680              0           1800           0         16...
...           ...            ...            ...         ...          ....
21608        1530              0           1530           0         15...
21609        2310              0           1830           0         23...
21610        1020              0           1020           0         10...
21611        1600              0           1410           0         16...
21612        1020              0           1020           0         1020

       view  floors  grade      lat  bathrooms  bedrooms
0         0     1.0      7  47.5112       1.00         3
1         0     2.0      7  47.7210       2.25         3
2         0     1.0      6  47.7379       1.00         2
3         0     1.0      7  47.5208       3.00         4
4         0     1.0      8  47.6168       2.00         3
...     ...     ...    ...      ...        ...       ...
21608     0     3.0      8  47.6993       2.50         3
21609     0     2.0      8  47.5107       2.50         4
21610     0     2.0      7  47.5944       0.75         2
21611     0     2.0      8  47.5345       2.50         3
21612     0     2.0      7  47.5941       0.75         2

[21613 rows x 11 columns]
```

[160]
```
print(Y2)
```

```
0        221900.0
1        538000.0
2        180000.0
3        604000.0
4        510000.0
           ...
21608    360000.0
```

```
21609    400000.0
21610    402101.0
21611    400000.0
21612    325000.0
Name: price, Length: 21613, dtype: float64
```

[161]
```
model = LinearRegression()
```

[162]
```
model.fit(X2,Y2)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
normalize=False)
```

[163]
```
r_sq= model.score(X2,Y2)
```

[164]
```
print("Coefficient of determination", r_sq)
```

```
Coefficient of determination 0.657699280616376
```

[165]
```
#Create a pipeline object that scales the data performs a
polynomial transform and fits a linear regression model. Fit the
object using the features in the question above, then fit the
model and calculate the R^2. Take a screenshot of your code and
the R^2.

Input=[('scale',StandardScaler()),('polynomial',
PolynomialFeatures(include_bias=False)),
('model',LinearRegression())]
```

[169]
```
pipe=Pipeline(Input)
pipe
```

```
Pipeline(memory=None,
         steps=[('scale',
                 StandardScaler(copy=True, with_mean=True,
with_std=True)),
                ('polynomial',
                 PolynomialFeatures(degree=2, include_bias=False,
                                    interaction_only=False, order='C')),
                ('model',
                 LinearRegression(copy_X=True, fit_intercept=True,
n_jobs=None,
```

```
                                       normalize=False))],
         verbose=False)
```

[171]
```
pipe.fit(df[features],df['price'])
```

```
Pipeline(memory=None,
         steps=[('scale',
                 StandardScaler(copy=True, with_mean=True,
with_std=True)),
                ('polynomial',
                 PolynomialFeatures(degree=2, include_bias=False,
                                    interaction_only=False, order='C')),
                ('model',
                 LinearRegression(copy_X=True, fit_intercept=True,
n_jobs=None,
                                  normalize=False))],
         verbose=False)
```

[172]
```
pipe.score(df[features],df['price'])
```

```
0.7513366125640563
```

[173]
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
```

[174]
```
X = df[features ]
Y = df['price']

x_train, x_test, y_train, y_test = train_test_split(X, Y,
test_size=0.15, random_state=1)


print("number of test samples :", x_test.shape[0])
print("number of training samples:",x_train.shape[0])
```

```
number of test samples : 3242
number of training samples: 18371
```

[178]
```
from sklearn.linear_model import Ridge
```

[179]
```
RigeModel = Ridge(alpha=0.1)
RigeModel.fit(x_train, y_train)
RigeModel.score(x_test, y_test)
```

```
0.6481004568444418
```

[181]
```
pr=PolynomialFeatures(degree=2)
x_train_pr=pr.fit_transform(x_train[features])
x_test_pr=pr.fit_transform(x_test[features])

RigeModel = Ridge(alpha=0.1)
RigeModel.fit(x_train_pr, y_train)
RigeModel.score(x_test_pr, y_test)
```

```
0.7005139891083106
```