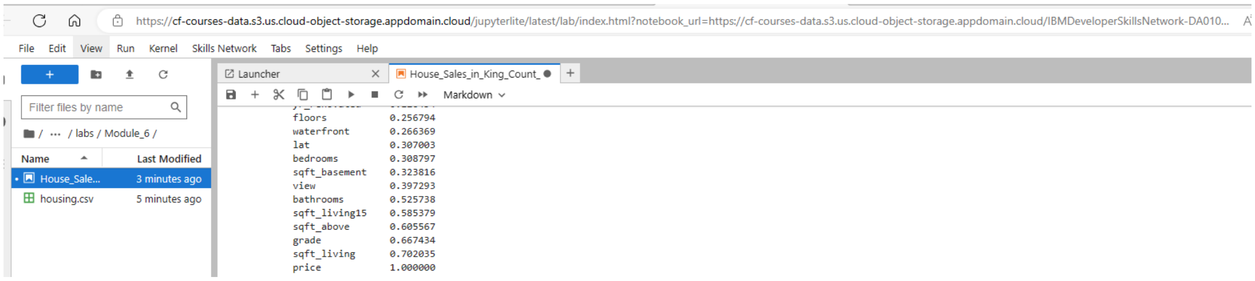
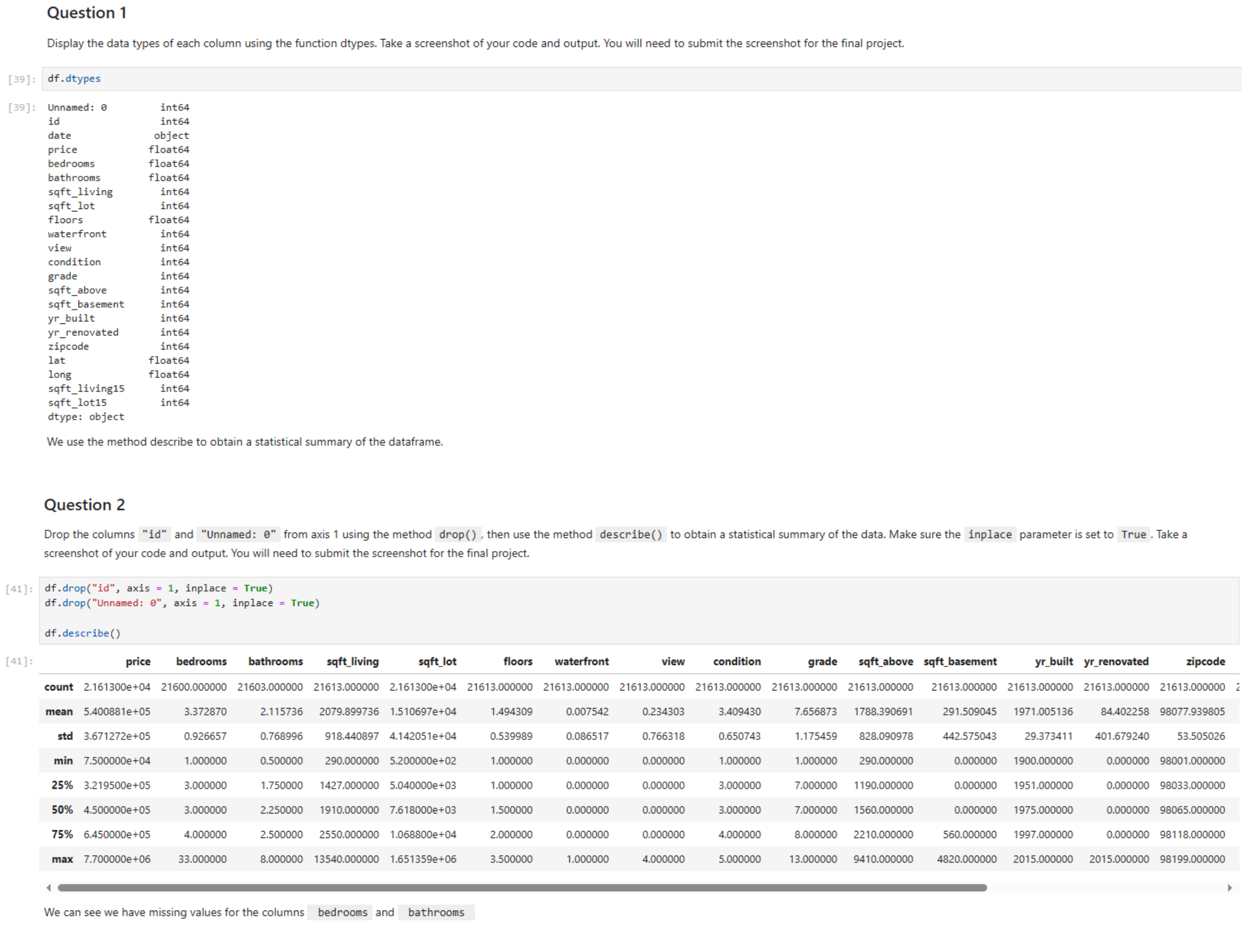
House sales King County – Final Lab Data Analysis for Python – Coursera (HS)

Path:


https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/jupyterlite/latest/lab/index.html?notebook_url=https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA010...
File
Edit
View
Run
Kernel
Skills Network
Tabs
Settings
Help
Filter files by name
/ ··· / labs / Module_6 /
Name
Last Modified
House_Sale...
3 minutes ago
housing.csv
5 minutes ago
Launcher
House_Sales_in_King_Count_
Markdown

```
floors           0.256794
waterfront       0.266369
lat              0.307003
bedrooms         0.308797
sqft_basement    0.323816
view             0.397293
bathrooms        0.525738
sqft_living15    0.585379
sqft_above       0.605567
grade            0.667434
sqft_living      0.702035
price            1.000000
```



Question 1

Display the data types of each column using the function dtypes. Take a screenshot of your code and output. You will need to submit the screenshot for the final project.

```
[39]: df.dtypes
```

```
[39]: Unnamed: 0         int64
      id                 int64
      date              object
      price            float64
      bedrooms         float64
      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
```

We use the method describe to obtain a statistical summary of the dataframe.

Question 2

Drop the columns `"id"` and `"Unnamed: 0"` from axis 1 using the method `drop()`, then use the method `describe()` to obtain a statistical summary of the data. Make sure the `inplace` parameter is set to `True`. Take a screenshot of your code and output. You will need to submit the screenshot for the final project.

```
[41]: df.drop("id", axis = 1, inplace = True)
      df.drop("Unnamed: 0", axis = 1, inplace = True)

      df.describe()
```

[41]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode
count	2.161300e+04	21600.000000	21603.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000
mean	5.400881e+05	3.372870	2.115736	2079.899736	1.510697e+04	1.494309	0.007542	0.234303	3.409430	7.656873	1788.390691	291.509045	1971.005136	84.402258	98077.939805
std	3.671272e+05	0.926657	0.768996	918.440897	4.142051e+04	0.539989	0.086517	0.766318	0.650743	1.175459	828.090978	442.575043	29.373411	401.679240	53.505026
min	7.500000e+04	1.000000	0.500000	290.000000	5.200000e+02	1.000000	0.000000	0.000000	1.000000	1.000000	290.000000	0.000000	1900.000000	0.000000	98001.000000
25%	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	0.000000	3.000000	7.000000	1190.000000	0.000000	1951.000000	0.000000	98033.000000
50%	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000	3.000000	7.000000	1560.000000	0.000000	1975.000000	0.000000	98065.000000
75%	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	0.000000	0.000000	4.000000	8.000000	2210.000000	560.000000	1997.000000	0.000000	98118.000000
max	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000	5.000000	13.000000	9410.000000	4820.000000	2015.000000	2015.000000	98199.000000

We can see we have missing values for the columns `bedrooms` and `bathrooms`

Module 3: Exploratory Data Analysis

Question 3

Use the method `value_counts` to count the number of houses with unique floor values, use the method `.to_frame()` to convert it to a data frame. Take a screenshot of your code and output. You will need to submit the screenshot for the final project.

```
[46]: df['floors'].value_counts().to_frame()
```

[46]:

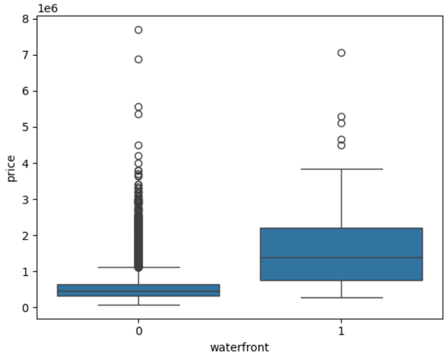
	floors
1.0	10680
2.0	8241
1.5	1910
3.0	613
2.5	161
3.5	8

Question 4

Use the function `boxplot` in the seaborn library to determine whether houses with a waterfront view or without a waterfront view have more price outliers. Take a screenshot of your code and boxplot. You will need to submit the screenshot for the final project.

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

```
[47]: <AxesSubplot:xlabel='waterfront', ylabel='price'>
```

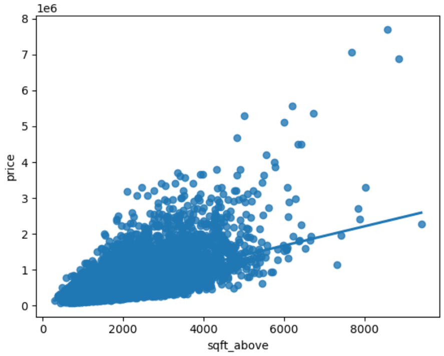


Question 5

Use the function `regplot` in the seaborn library to determine if the feature `sqft_above` is negatively or positively correlated with price. Take a screenshot of your code and scatterplot. You will need to submit the screenshot for the final project.

```
[48]: sns.regplot(x="sqft_above", y="price", data=df, ci = None)
```

```
[48]: <AxesSubplot:xlabel='sqft_above', ylabel='price'>
```



Module 4: Model Development

We can Fit a linear regression model using the longitude feature `'long'` and caculate the R^2.

```
[51]: X = df[['long']]
      Y = df['price']
      lm = LinearRegression()
      lm.fit(X,Y)
      lm.score(X, Y)
```

```
[51]: 0.00046769430149007363
```

Question 6

Fit a linear regression model to predict the `'price'` using the feature `'sqft_living'` then calculate the R^2. Take a screenshot of your code and the value of the R^2. You will need to submit it for the final project.

```
[52]: X1 = df[['sqft_living']]
      Y1 = df['price']
      lm = LinearRegression()
      lm
      lm.fit(X1,Y1)
      lm.score(X1, Y1)
```

```
[52]: 0.4928532179037931
```

Question 7

Fit a linear regression model to predict the `'price'` using the list of features:

```
[53]: features =["floors", "waterfront","lat" ,"bedrooms" ,"sqft_basement" ,"view" ,"bathrooms","sqft_living15","sqft_above","grade","sqft_living"]
```

Then calculate the R^2. Take a screenshot of your code and the value of the R^2. You will need to submit it for the final project.

```
[54]: X2 = df[features]
      Y2 = df['price']
      lm.fit(X2,Y2)
      lm.score(X2,Y2)
```

```
[54]: 0.6576890354915759
```

Question 8

Use the list to create a pipeline object to predict the 'price', fit the object using the features in the list `features`, and calculate the R^2. Take a screenshot of your code and the value of the R^2. You will need to submit it for the final project.

```
[56]: pipe=Pipeline(Input)
      pipe
      X = df[features]
      Y = df['price']
      pipe.fit(X,Y)
      pipe.score(X,Y)
```

```
[56]: 0.7512051345272872
```

Question 9

Create and fit a Ridge regression object using the training data, set the regularization parameter to 0.1, and calculate the R^2 using the test data. Take a screenshot of your code and the value of the R^2. You will need to submit it for the final project.

```
[59]: from sklearn.linear_model import Ridge
```

```
[60]: RidgeModel = Ridge(alpha=0.1)
      RidgeModel.fit(x_train, y_train)
      RidgeModel.score(x_test, y_test)
```

```
[60]: 0.647875916393907
```

Question 10

Perform a second order polynomial transform on both the training data and testing data. Create and fit a Ridge regression object using the training data, set the regularisation parameter to 0.1, and calculate the R^2 utilising the test data provided. Take a screenshot of your code and the R^2. You will need to submit it for the final project.

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

      RidgeModel1 = Ridge(alpha = 0.1)
      RidgeModel1.fit(x_train_pr, y_train)
      RidgeModel1.score(x_test_pr, y_test)
```

```
[61]: 0.7002744263583341
```