Untitled

June 9, 2021

1 Case study 1 from Udacity

Our data set is about red and white wine, we have diferrent values such that:acidity,citric_acid ,residual_sugar,chlorides,sulfur-dioxide,density,pH,sulphates,alcohol, quality. We made different operation in Jupyter Notebook: analyzing data, cleaning data and making visualizations.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
% matplotlib inline
import seaborn as sns
# At first, we need to read the csv file in Jupyter, we have two separate data which are
# for differentiating of data we named df_red and df_white, and we used the sep to add a
df_red=pd.read_csv('winequality-red.csv', sep=';')
df_white=pd.read_csv('winequality-white.csv',sep=';')
# We utilized head() function to check first five index
df_red.head()
```

Out[1]:

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur-dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

In [3]: df_white.head()

Out[3]:

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	\
0	7.0	0.27	0.36	20.7	0.045	
1	6.3	0.30	0.34	1.6	0.049	
2	8.1	0.28	0.40	6.9	0.050	
3	7.2	0.23	0.32	8.5	0.058	
4	7.2	0.23	0.32	8.5	0.058	

	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	\
0	45.0	170.0	1.0010	3.00	0.45	
1	14.0	132.0	0.9940	3.30	0.49	
2	30.0	97.0	0.9951	3.26	0.44	
3	47.0	186.0	0.9956	3.19	0.40	
4	47.0	186.0	0.9956	3.19	0.40	

	alcohol	quality
0	8.8	6
1	9.5	6
2	10.1	6
3	9.9	6
4	9.9	6

In [5]:
```
# to get brief information about the data we used info function,
# here we can see the number of samples, rows, the size of data and the datatypes etc.
df_red.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
fixed_acidity           1599 non-null float64
volatile_acidity        1599 non-null float64
citric_acid             1599 non-null float64
residual_sugar          1599 non-null float64
chlorides               1599 non-null float64
free_sulfur_dioxide     1599 non-null float64
total_sulfur-dioxide    1599 non-null float64
density                 1599 non-null float64
pH                      1599 non-null float64
sulphates               1599 non-null float64
alcohol                 1599 non-null float64
quality                 1599 non-null int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

In [7]: df_white.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4898 entries, 0 to 4897
```

```
Data columns (total 12 columns):
fixed_acidity           4898 non-null float64
volatile_acidity        4898 non-null float64
citric_acid             4898 non-null float64
residual_sugar          4898 non-null float64
chlorides               4898 non-null float64
free_sulfur_dioxide     4898 non-null float64
total_sulfur_dioxide    4898 non-null float64
density                 4898 non-null float64
pH                      4898 non-null float64
sulphates               4898 non-null float64
alcohol                 4898 non-null float64
quality                 4898 non-null int64
dtypes: float64(11), int64(1)
memory usage: 459.3 KB
```

```
In [9]: # it was asked the number of duplicated values from the data of white wine, we used sum(
        # if it needs to delete the duplicated values, we will use drop() function.
        sum(df_white.duplicated())
```

```
Out[9]: 937
```

```
In [11]: # After all next step is that how many unique value in each column there are,
         # we found data below with using nunique() function.
         df_red.nunique()
```

```
Out[11]: fixed_acidity            96
         volatile_acidity        143
         citric_acid              80
         residual_sugar           91
         chlorides               153
         free_sulfur_dioxide      60
         total_sulfur-dioxide    144
         density                 436
         pH                       89
         sulphates                96
         alcohol                  65
         quality                   6
         dtype: int64
```

```
In [13]: df_white.nunique()
```

```
Out[13]: fixed_acidity            68
         volatile_acidity        125
         citric_acid              87
         residual_sugar          310
         chlorides               160
         free_sulfur_dioxide     132
```

```
total_sulfur_dioxide    251
density                 890
pH                      103
sulphates                79
alcohol                 103
quality                   7
dtype: int64
```

In [15]:
```
# Next question is that what are averages of each columns in the data of red wine.
# We used mean() function to get this result.
df_red.mean()
```

Out[15]:
```
fixed_acidity            8.319637
volatile_acidity         0.527821
citric_acid              0.270976
residual_sugar           2.538806
chlorides                0.087467
free_sulfur_dioxide     15.874922
total_sulfur-dioxide    46.467792
density                  0.996747
pH                       3.311113
sulphates                0.658149
alcohol                 10.422983
quality                  5.636023
dtype: float64
```

In [17]:
```
# we need to use rename function to turn into the same name,
# because when we append the data then we will see the incorrect values.
df_red.rename(columns={'total_sulfur-dioxide':'total_sulfur_dioxide'}, inplace=True)
```

In [20]:
```
# create color array for red and white  dataframe
color_red = np.repeat('red', df_red.shape[0])

color_white = np.repeat('white', df_white.shape[0])
```

In [21]:
```
# we want to add column to the table, which called color in both data.
df_red['color'] = color_red
df_red.head()
```

Out[21]:

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	\
0	7.4	0.70	0.00	1.9	0.076	
1	7.8	0.88	0.00	2.6	0.098	
2	7.8	0.76	0.04	2.3	0.092	
3	11.2	0.28	0.56	1.9	0.075	
4	7.4	0.70	0.00	1.9	0.076	

	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	\
0	11.0	34.0	0.9978	3.51	0.56	
1	25.0	67.0	0.9968	3.20	0.68	

```
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol  quality color
0      9.4        5   red
1      9.8        5   red
2      9.8        5   red
3      9.8        6   red
4      9.4        5   red
```

In [23]:
```
df_white['color'] = color_white
df_white.head()
```

Out[23]:
```
   fixed_acidity  volatile_acidity  citric_acid  residual_sugar  chlorides  \
0            7.0              0.27         0.36            20.7      0.045
1            6.3              0.30         0.34             1.6      0.049
2            8.1              0.28         0.40             6.9      0.050
3            7.2              0.23         0.32             8.5      0.058
4            7.2              0.23         0.32             8.5      0.058

   free_sulfur_dioxide  total_sulfur_dioxide  density    pH  sulphates  \
0                 45.0                 170.0   1.0010  3.00       0.45
1                 14.0                 132.0   0.9940  3.30       0.49
2                 30.0                  97.0   0.9951  3.26       0.44
3                 47.0                 186.0   0.9956  3.19       0.40
4                 47.0                 186.0   0.9956  3.19       0.40

   alcohol  quality  color
0      8.8        6  white
1      9.5        6  white
2     10.1        6  white
3      9.9        6  white
4      9.9        6  white
```

In [25]:
```
# after all we append dataframes
wine_df = df_red.append(df_white)

# we check the dataframe to control our data
wine_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6497 entries, 0 to 4897
Data columns (total 13 columns):
fixed_acidity           6497 non-null float64
volatile_acidity        6497 non-null float64
citric_acid             6497 non-null float64
residual_sugar          6497 non-null float64
chlorides               6497 non-null float64
```

```
free_sulfur_dioxide     6497 non-null float64
total_sulfur_dioxide    6497 non-null float64
density                 6497 non-null float64
pH                      6497 non-null float64
sulphates               6497 non-null float64
alcohol                 6497 non-null float64
quality                 6497 non-null int64
color                   6497 non-null object
dtypes: float64(11), int64(1), object(1)
memory usage: 710.6+ KB
```

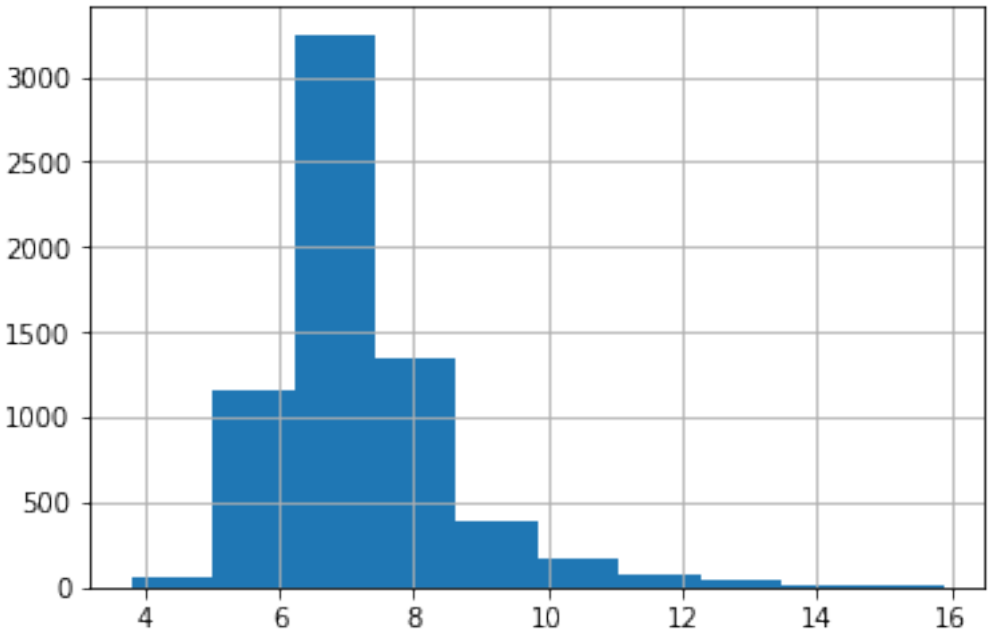
```
In [27]: wine_df.head()
```

Out[27]:

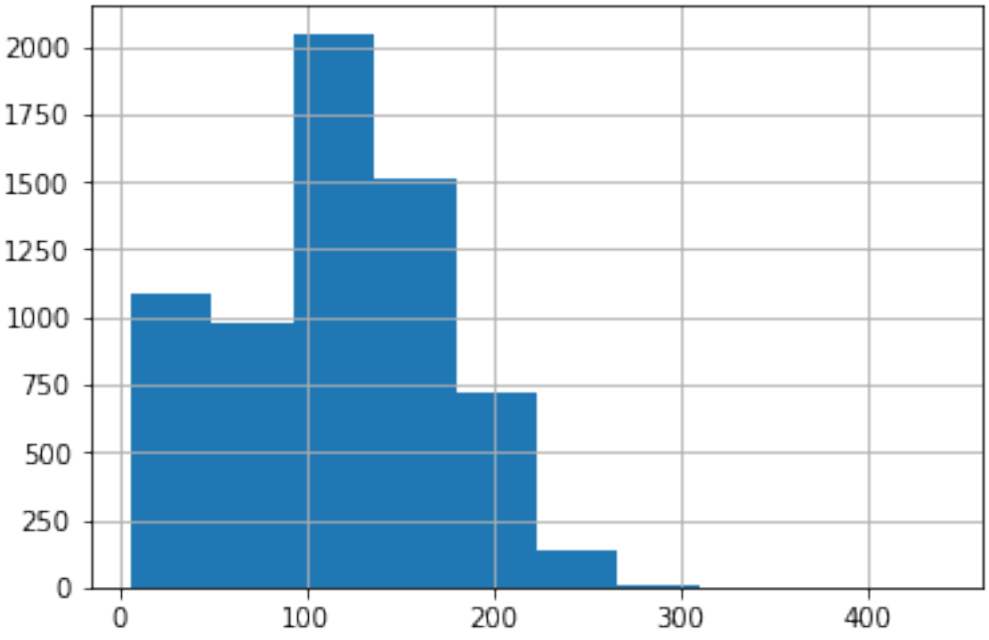
	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alcohol	quality	color
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	red
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	red
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	red
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	red
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	red

2 After all, we make visualization to understand data, it helps us to make predictions easily

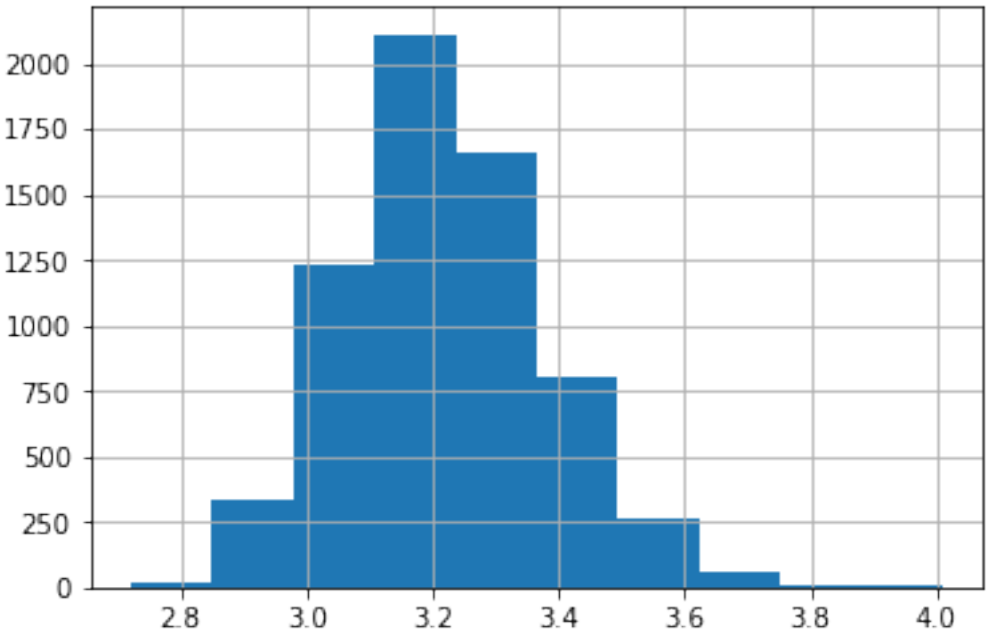
```
In [29]: ## we used hist() function to show different features of data
         wine_df.fixed_acidity.hist();
```



In [31]: `wine_df.total_sulfur_dioxide.hist();`

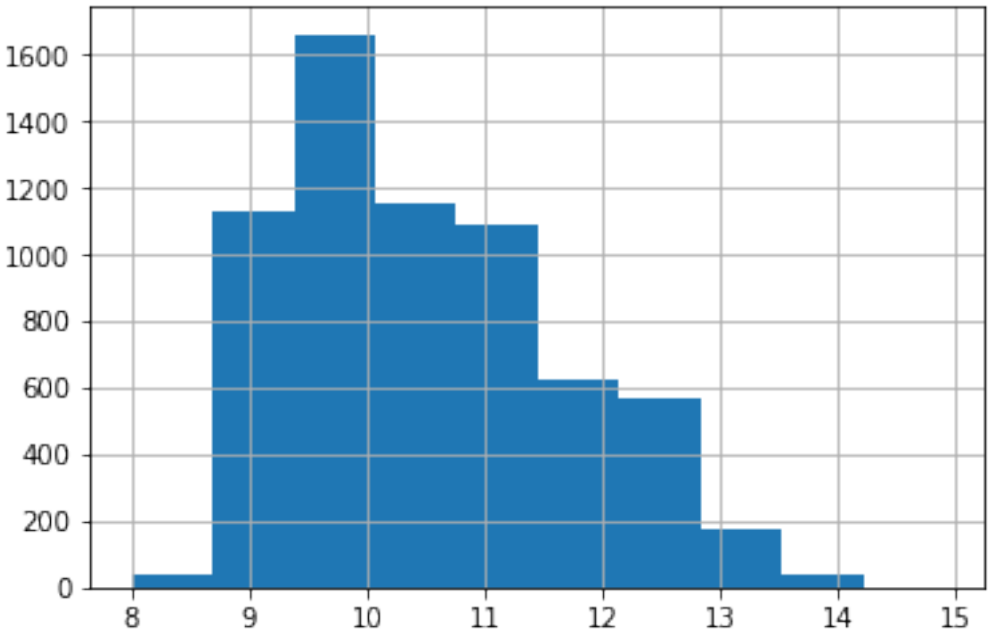


In [32]: `wine_df.pH.hist();`

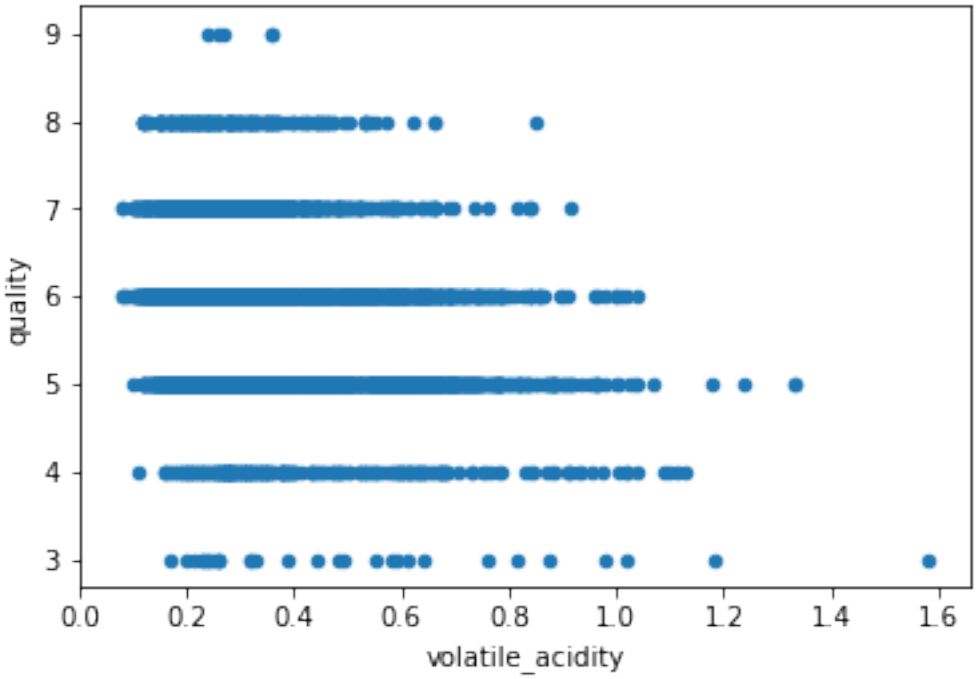



```
In [33]: wine_df.alcohol.hist()
```

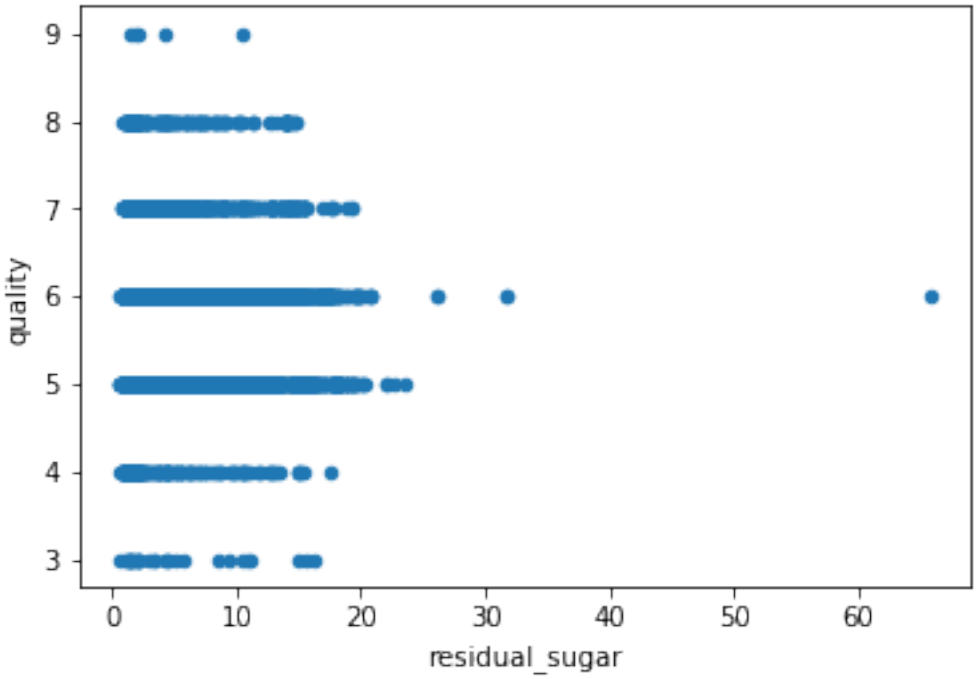
```
Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x7fddef8daf28>
```



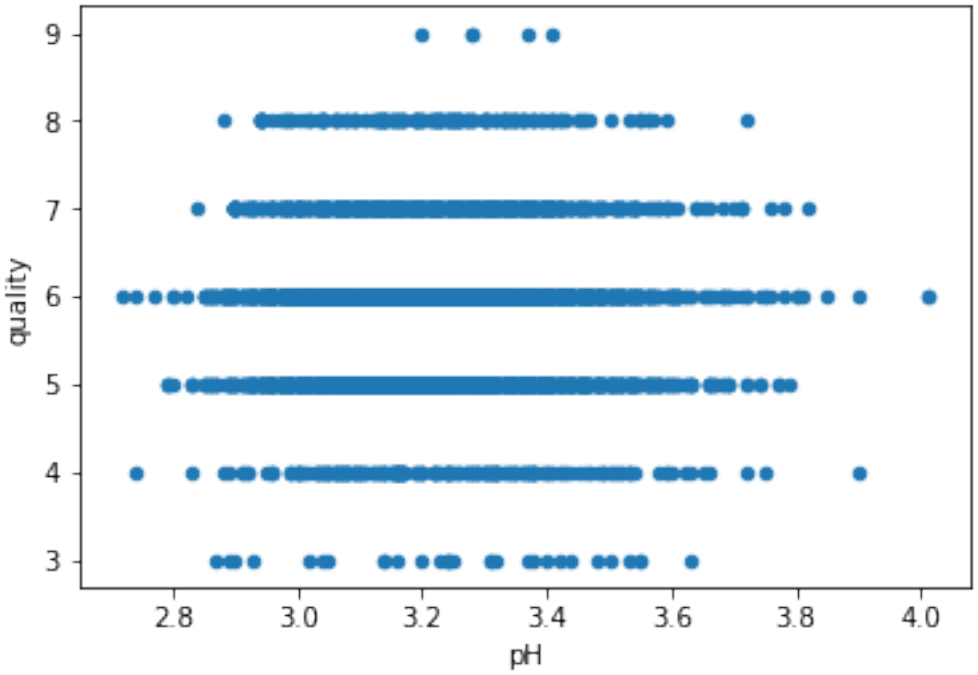
In [34]:
```
# after that, our next mission, to create scatter plot o compare different features wit
wine_df.plot(x="volatile_acidity", y="quality", kind="scatter");
```



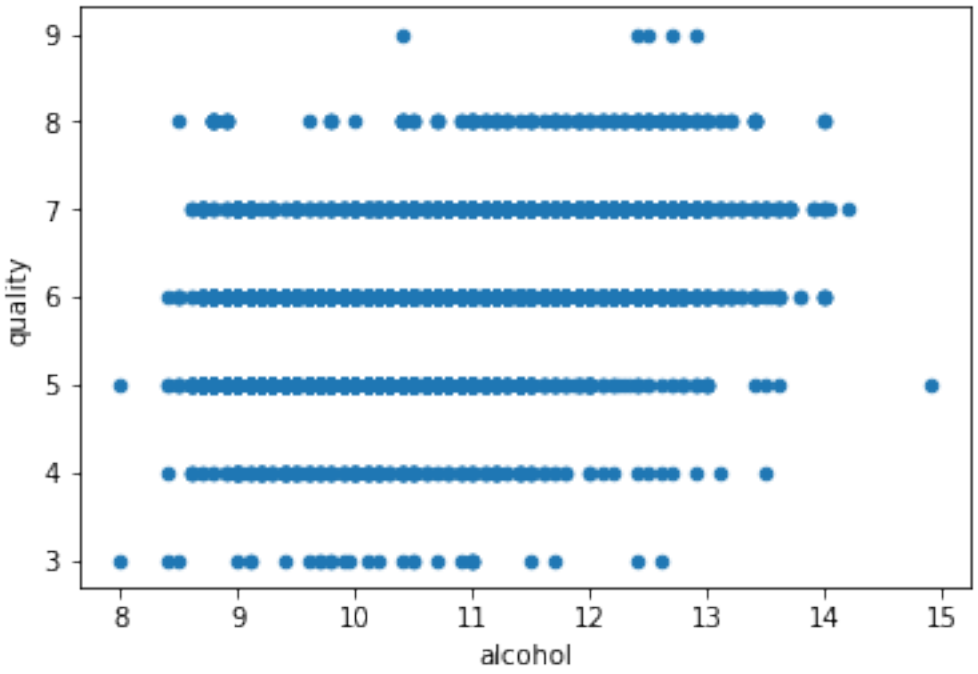
In [35]:
```
wine_df.plot(x="residual_sugar", y="quality", kind="scatter");
```



In [36]: `wine_df.plot(x="pH", y="quality", kind="scatter");`



In [37]: `wine_df.plot(x="alcohol", y="quality", kind="scatter");`



3 At last we make drawing conclusion

```
In [38]: # Find the mean quality of each wine type (red and white) with groupby
         wine_df.groupby('color').mean().quality
```

```
Out[38]: color
         red      5.636023
         white    5.877909
         Name: quality, dtype: float64
```

```
In [41]: # View the min, 25%, 50%, 75%, max pH values with Pandas describe
         wine_df.describe().pH
```

```
Out[41]: count    6497.000000
         mean        3.218501
         std         0.160787
         min         2.720000
         25%         3.110000
         50%         3.210000
         75%         3.320000
         max         4.010000
         Name: pH, dtype: float64
```

```
In [42]: # Bin edges that will be used to "cut" the data into groups
         bin_edges = [2, 3, 3.5, 4, 5] # Fill in this list with five values you just found
         # Labels for the four acidity level groups
         # Name each acidity level category
         bin_names = ['high', 'mod_high', 'medium', 'low']
         # Creates acidity_levels column
         wine_df['acidity_levels'] = pd.cut(wine_df['pH'], bin_edges, labels=bin_names)

         # Checks for successful creation of this column
         wine_df.head()
```

```
Out[42]:    fixed_acidity  volatile_acidity  citric_acid  residual_sugar  chlorides  \
         0            7.4              0.70         0.00             1.9      0.076
         1            7.8              0.88         0.00             2.6      0.098
         2            7.8              0.76         0.04             2.3      0.092
         3           11.2              0.28         0.56             1.9      0.075
         4            7.4              0.70         0.00             1.9      0.076

            free_sulfur_dioxide  total_sulfur_dioxide  density    pH  sulphates  \
         0                 11.0                  34.0   0.9978  3.51       0.56
         1                 25.0                  67.0   0.9968  3.20       0.68
         2                 15.0                  54.0   0.9970  3.26       0.65
```

```
3                17.0                  60.0   0.9980  3.16       0.58
4                11.0                  34.0   0.9978  3.51       0.56

   alcohol  quality color acidity_levels
0      9.4        5   red         medium
1      9.8        5   red       mod_high
2      9.8        5   red       mod_high
3      9.8        6   red       mod_high
4      9.4        5   red         medium
```

In [43]:
```
# Find the mean quality of each acidity level with groupby
wine_df.groupby('acidity_levels').mean().quality
```

Out[43]:
```
acidity_levels
high        5.836996
mod_high    5.820631
medium      5.742671
low         6.000000
Name: quality, dtype: float64
```

In [44]:
```
# get the median amount of alcohol content

wine_df.median().alcohol
```

Out[44]:
```
10.300000000000001
```

In [46]:
```
# select samples with alcohol content less than the median
low_alcohol = wine_df.query('alcohol < 10.3')

# select samples with alcohol content greater than or equal to the median
high_alcohol =  wine_df.query('alcohol >= 10.3')

# ensure these queries included each sample exactly once
num_samples = wine_df.shape[0]
num_samples == low_alcohol['quality'].count() + high_alcohol['quality'].count() # shoul
```

Out[46]:
```
True
```

In [47]:
```
# get mean quality rating for the low alcohol and high alcohol groups
low_alcohol.quality.mean(), high_alcohol.quality.mean()
```

Out[47]:
```
(5.475920679886686, 6.1460843373493974)
```

In [48]:
```
# get the median amount of residual sugar
wine_df.residual_sugar.median()
```

Out[48]:
```
3.0
```

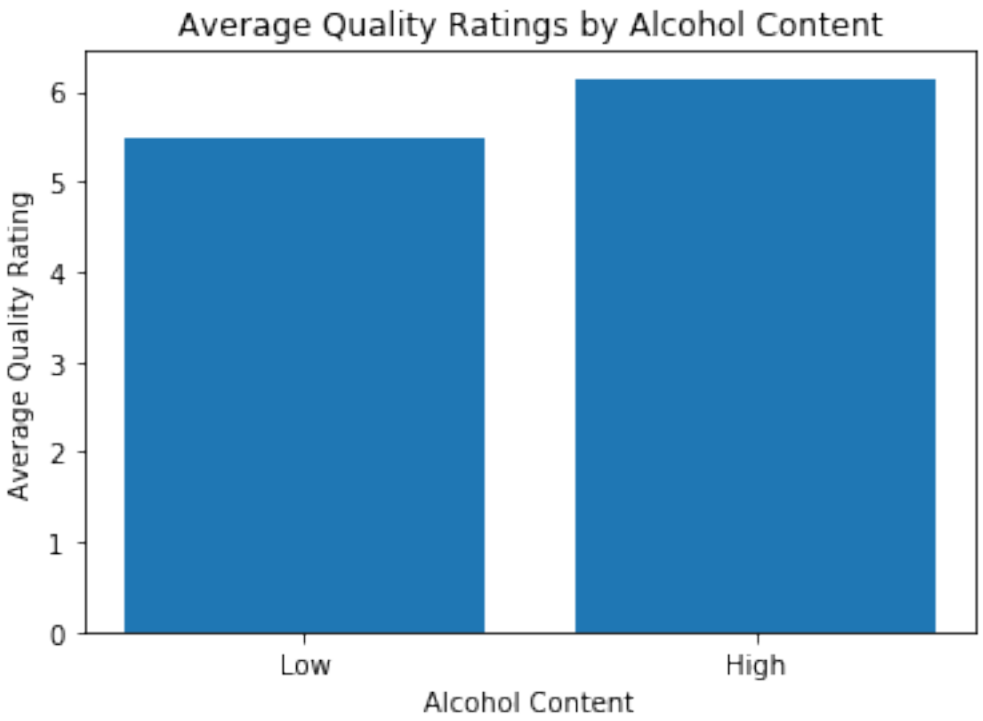
In [49]:
```
# select samples with residual sugar less than the median
low_sugar = wine_df.query('residual_sugar < 3.0')

# select samples with residual sugar greater than or equal to the median
high_sugar = wine_df.query('residual_sugar >= 3.0')

# ensure these queries included each sample exactly once
num_samples == low_sugar['quality'].count() + high_sugar['quality'].count() # should be
```

Out[49]:
```
True
```

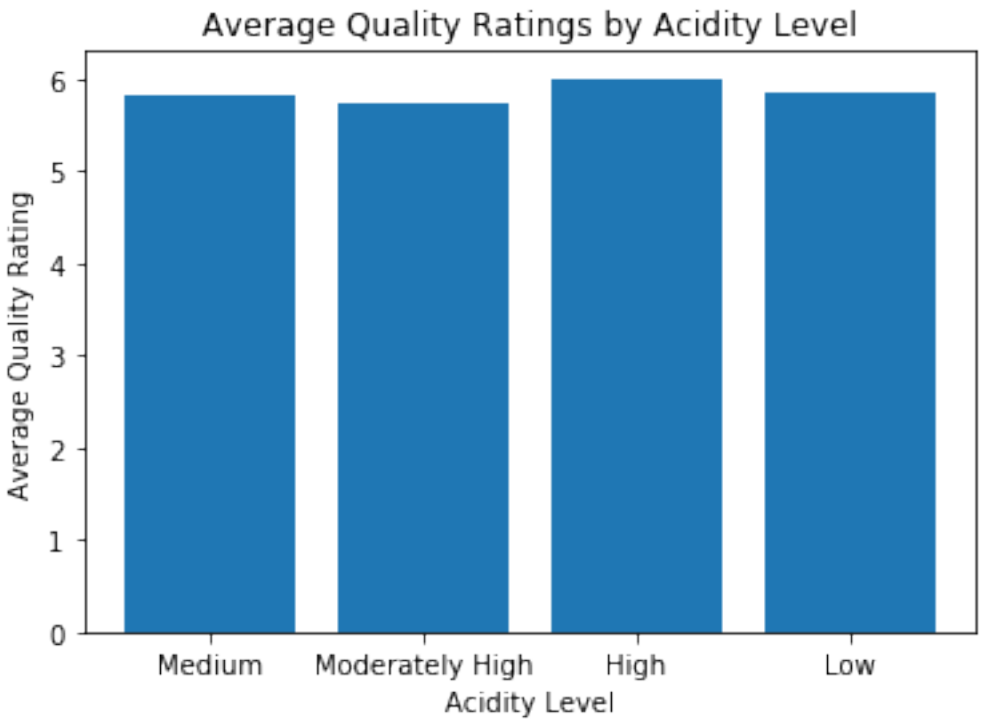
In [50]:
```
# get mean quality rating for the low sugar and high sugar groups
low_sugar.quality.mean(), high_sugar.quality.mean()
```

Out[50]:
```
(5.8088007437248219, 5.8278287461773699)
```

In [51]:
```
# Use query to select each group and get its mean quality
median = wine_df['alcohol'].median()
low = wine_df.query('alcohol < {}'.format(median))
high = wine_df.query('alcohol >= {}'.format(median))

mean_quality_low = low['quality'].mean()
mean_quality_high = high['quality'].mean()
# Create a bar chart with proper labels
locations = [1, 2]
heights = [mean_quality_low, mean_quality_high]
labels = ['Low', 'High']
plt.bar(locations, heights, tick_label=labels)
plt.title('Average Quality Ratings by Alcohol Content')
plt.xlabel('Alcohol Content')
plt.ylabel('Average Quality Rating');
```




```
In [60]: # Use groupby to get the mean quality for each acidity level
         acdf=wine_df.groupby('acidity_levels').mean().quality
         acdf

Out[60]: acidity_levels
         high        5.836996
         mod_high    5.820631
         medium      5.742671
         low         6.000000
         Name: quality, dtype: float64

In [63]: locations = [4, 1, 2, 3]
         heights = acdf
         labels = ['Low', 'Medium', 'Moderately High', 'High']

         plt.bar(locations, heights, tick_label=labels)
         plt.title('Average Quality Ratings by Acidity Level'),
         plt.xlabel('Acidity Level')
         plt.ylabel('Average Quality Rating');
```



4 Create arrays for red bar heights white bar heights

```
In [64]: import seaborn as sns
         sns.set_style('darkgrid')
```

```
In [65]: # get counts for each rating and color
         color_counts = wine_df.groupby(['color', 'quality']).count()['pH']
         color_counts
```

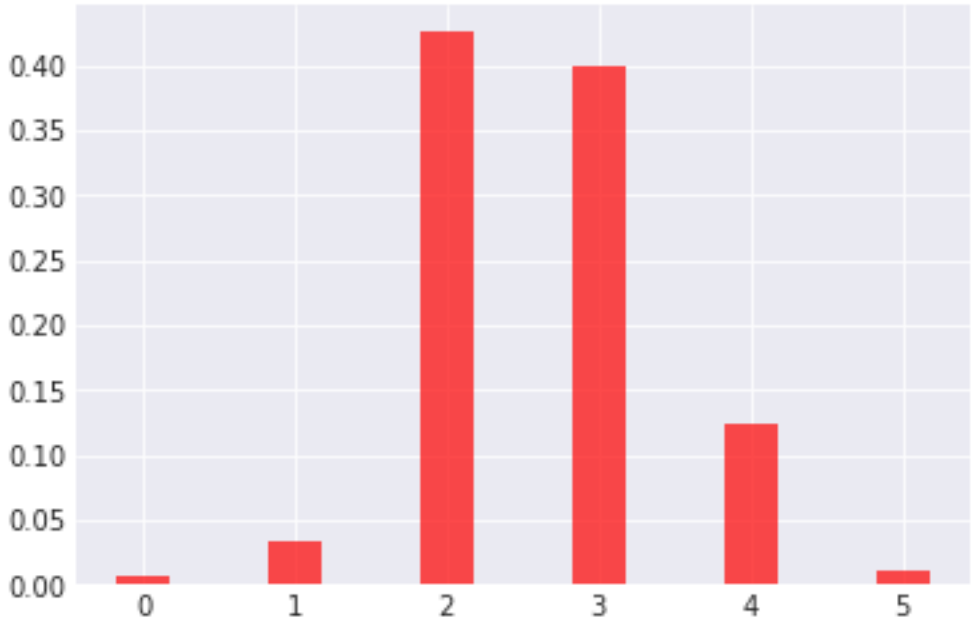
```
Out[65]: color  quality
         red    3            10
                4            53
                5           681
                6           638
                7           199
                8            18
         white  3            20
                4           163
                5          1457
                6          2198
                7           880
                8           175
                9             5
         Name: pH, dtype: int64
```

```
In [66]: # get total counts for each color
         color_totals = wine_df.groupby('color').count()['pH']
         color_totals

Out[66]: color
         red      1599
         white    4898
         Name: pH, dtype: int64

In [67]: # get proportions by dividing red rating counts by total # of red samples
         red_proportions = color_counts['red'] / color_totals['red']
         red_proportions

Out[67]: quality
         3    0.006254
         4    0.033146
         5    0.425891
         6    0.398999
         7    0.124453
         8    0.011257
         Name: pH, dtype: float64

In [68]: # get proportions by dividing white rating counts by total # of white samples
         white_proportions = color_counts['white'] / color_totals['white']
         white_proportions

Out[68]: quality
         3    0.004083
         4    0.033279
         5    0.297468
         6    0.448755
         7    0.179665
         8    0.035729
         9    0.001021
         Name: pH, dtype: float64

In [69]: #Plot proportions on a bar chart
         #Set the x coordinate location for each rating group and and width of each bar.
         ind = np.arange(len(red_proportions))  # the x locations for the groups
         width = 0.35       # the width of the bars

In [73]: # plot bars
         red_bars = plt.bar(ind, red_proportions, width, color='r', alpha=.7, label='Red Wine')
         white_bars = plt.bar(ind+ width, white_proportions, width, color='w', alpha=.7, label='

         # title and labels
         plt.ylabel('Proportion')
         plt.xlabel('Quality')
         plt.title('Proportion by Wine Color and Quality')
```

```
    locations = ind + width / 2  # xtick locations
    labels = ['3', '4', '5', '6', '7', '8', '9']  # xtick labels
    plt.xticks(locations, labels)

    # legend
    plt.legend()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)

<ipython-input-73-b5179befdc5a> in <module>()
      1 # plot bars
      2 red_bars = plt.bar(ind, red_proportions, width, color='r', alpha=.7, label='Red Wine
----> 3 white_bars = plt.bar(ind, width, white_proportions, width, color='w', alpha=.7, labe
      4
      5 # title and labels


/opt/conda/lib/python3.6/site-packages/matplotlib/pyplot.py in bar(*args, **kwargs)
   2625                         mplDeprecation)
   2626     try:
-> 2627         ret = ax.bar(*args, **kwargs)
   2628     finally:
   2629         ax._hold = washold


/opt/conda/lib/python3.6/site-packages/matplotlib/__init__.py in inner(ax, *args, **kwar
   1708                     warnings.warn(msg % (label_namer, func.__name__),
   1709                                   RuntimeWarning, stacklevel=2)
-> 1710             return func(ax, *args, **kwargs)
   1711         pre_doc = inner.__doc__
   1712         if pre_doc is None:


/opt/conda/lib/python3.6/site-packages/matplotlib/axes/_axes.py in bar(self, *args, **kw
   2079         x, height, width, y, linewidth = np.broadcast_arrays(
   2080             # Make args iterable too.
-> 2081             np.atleast_1d(x), height, width, y, linewidth)
   2082
   2083         if orientation == 'vertical':


/opt/conda/lib/python3.6/site-packages/numpy/lib/stride_tricks.py in broadcast_arrays(*a
    248     args = [np.array(_m, copy=False, subok=subok) for _m in args]
    249
--> 250     shape = _broadcast_shape(*args)
```

```
    251
    252     if all(array.shape == shape for array in args):


/opt/conda/lib/python3.6/site-packages/numpy/lib/stride_tricks.py in _broadcast_shape(*a
    183     # use the old-iterator because np.nditer does not handle size 0 arrays
    184     # consistently
--> 185     b = np.broadcast(*args[:32])
    186     # unfortunately, it cannot handle 32 or more arguments directly
    187     for pos in range(32, len(args), 31):


ValueError: shape mismatch: objects cannot be broadcast to a single shape
```



In [76]:
```
#Oh, that didn't work because we're missing a red wine value for a the 9 rating. Even t
#we need it for our plot. Run the last two cells after running the cell below.
red_proportions['9'] = 0
red_proportions
```

Out[76]:
```
quality
3    0.006254
4    0.033146
5    0.425891
6    0.398999
7    0.124453
```

```
8    0.011257
9    0.000000
Name: pH, dtype: float64
```

In [ ]: