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

In [2]:
```python
# Q(1) Load the pandas library and create two different data frames, namely, df_red for
#the white wine dataset
df_red = pd.read_csv(r"C:\Users\hp\Downloads\winequality-red.csv")
df_white = pd.read_csv(r"C:\Users\hp\Downloads\winequality-white.csv")
```

In [3]:
```python
#Q(2) Find the name of the available columns.
print(df_red.columns)
print(df_white.columns)
```

```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

In [4]:
```python
#Q(3) Let's see some sample data from the red wine data frame. check the entries between
df_red.iloc[101:110]
```

Out[4]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
101	7.8	0.500	0.30	1.9	0.075	8.0	22.0	0.9959	3.31	0.56	10.4	6
102	8.1	0.545	0.18	1.9	0.080	13.0	35.0	0.9972	3.30	0.59	9.0	6
103	8.1	0.575	0.22	2.1	0.077	12.0	65.0	0.9967	3.29	0.51	9.2	5
104	7.2	0.490	0.24	2.2	0.070	5.0	36.0	0.9960	3.33	0.48	9.4	5
105	8.1	0.575	0.22	2.1	0.077	12.0	65.0	0.9967	3.29	0.51	9.2	5
106	7.8	0.410	0.68	1.7	0.467	18.0	69.0	0.9973	3.08	1.31	9.3	5
107	6.2	0.630	0.31	1.7	0.088	15.0	64.0	0.9969	3.46	0.79	9.3	5
108	8.0	0.330	0.53	2.5	0.091	18.0	80.0	0.9976	3.37	0.80	9.6	6
109	8.1	0.785	0.52	2.0	0.122	37.0	153.0	0.9969	3.21	0.69	9.3	5

In [5]:
```python
# Q(4) Check the data types for each column.
df_red.info()
df_white.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
```

```
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4898 entries, 0 to 4897
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         4898 non-null   float64
 1   volatile acidity      4898 non-null   float64
 2   citric acid           4898 non-null   float64
 3   residual sugar        4898 non-null   float64
 4   chlorides             4898 non-null   float64
 5   free sulfur dioxide   4898 non-null   float64
 6   total sulfur dioxide  4898 non-null   float64
 7   density               4898 non-null   float64
 8   pH                    4898 non-null   float64
 9   sulphates             4898 non-null   float64
 10  alcohol               4898 non-null   float64
 11  quality               4898 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 459.3 KB
```

In [6]:
```
# Q(5)Describe the data frame to get more descriptive information.
df_red.describe()
```

Out[6]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690

In [7]:
```
# Q6. Find which of the columns have null values.
df_red.isna().any()
```

Out[7]:
```
fixed acidity           False
volatile acidity        False
citric acid             False
residual sugar          False
chlorides               False
free sulfur dioxide     False
total sulfur dioxide    False
density                 False
pH                      False
sulphates               False
alcohol                 False
quality                 False
dtype: bool
```

```
# Q7. We will continue analyzing the red wine dataset. First, we will start by exploring
```

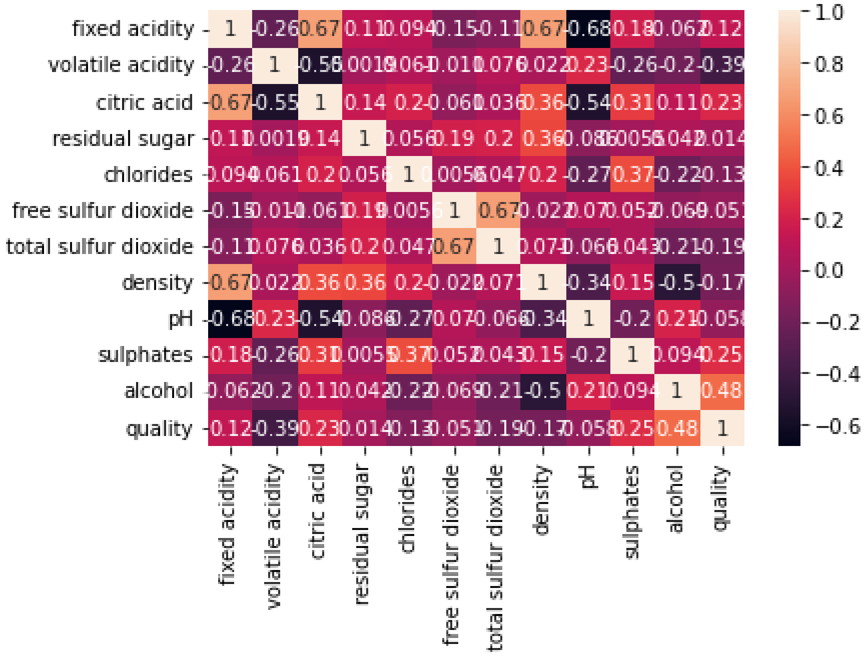
In [8]:
```
#different columns and observe their columns.
#First, start with the quality column of red wine. Plot a graph of your choice to do the
df_red.corr()
```

Out[8]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulpha
fixed acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153794	-0.113181	0.668047	-0.682978	0.1830
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470	0.022026	0.234937	-0.2609
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533	0.364947	-0.541904	0.3127
residual sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187049	0.203028	0.355283	-0.085652	0.0055
chlorides	0.093705	0.061298	0.203823	0.055610	1.000000	0.005562	0.047400	0.200632	-0.265026	0.3712
free sulfur dioxide	-0.153794	-0.010504	-0.060978	0.187049	0.005562	1.000000	0.667666	-0.021946	0.070377	0.0516
total sulfur dioxide	-0.113181	0.076470	0.035533	0.203028	0.047400	0.667666	1.000000	0.071269	-0.066495	0.0429
density	0.668047	0.022026	0.364947	0.355283	0.200632	-0.021946	0.071269	1.000000	-0.341699	0.1485
pH	-0.682978	0.234937	-0.541904	-0.085652	-0.265026	0.070377	-0.066495	-0.341699	1.000000	-0.1966
sulphates	0.183006	-0.260987	0.312770	0.005527	0.371260	0.051658	0.042947	0.148506	-0.196648	1.0000
alcohol	-0.061668	-0.202288	0.109903	0.042075	-0.221141	-0.069408	-0.205654	-0.496180	0.205633	0.0935
quality	0.124052	-0.390558	0.226373	0.013732	-0.128907	-0.050656	-0.185100	-0.174919	-0.057731	0.2513

In [9]:
```
# Q8. Let's next find out which of the columns from the red wine database are highly cor
#You need to confirm whether the following is true
#- Alcohol is positively correlated with the quality of red wine.
#- Alcohol has a weak positive correlation with the pH value.
#- Citric acid and density have a strong positive correlation with fixed acidity.
#- pH has a negative correlation with density, fixed acidity, citric acid, and sulfates.
#Use pair plot, heatmap, etc to draw these correlations.
sns.heatmap(df_red.corr(),annot=True)
```

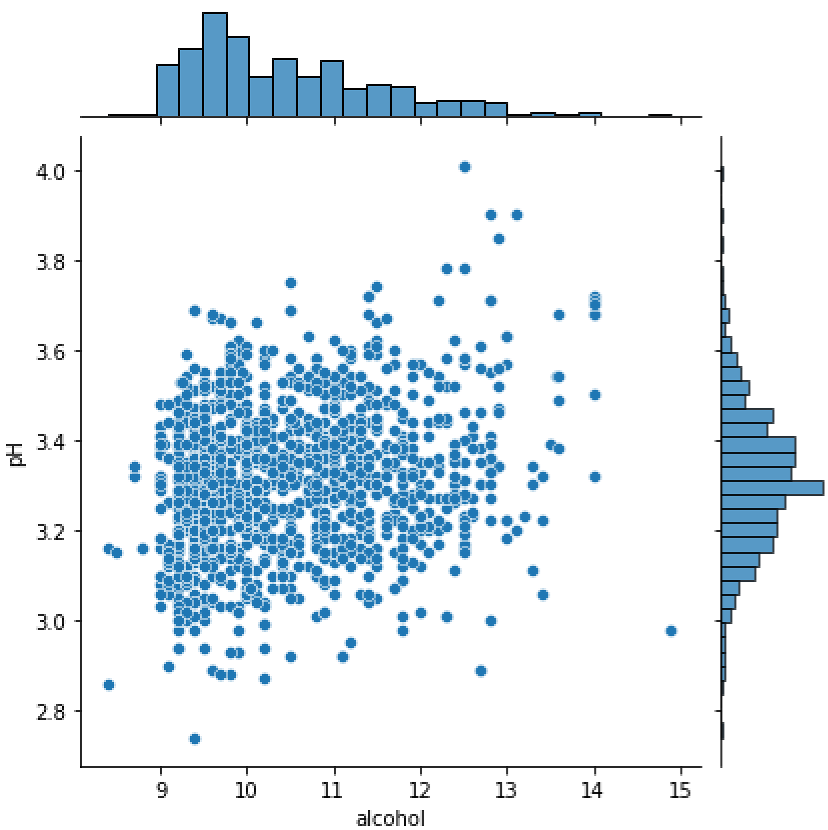
Out[9]: <AxesSubplot:>



# Q9. Let's see how the quality of wine varies with respect to alcohol concentration. Use a boxplot for the same. #Can you draw this conclusion: The higher the alcohol concentration is, the higher the quality of the wine.
sns.boxplot(x=df_red['quality'], y=df_red['alcohol'])

In [11]:
```
# Q10. Let's also see the correlation between the alcohol column and pH values. Use join
sns.jointplot(x= df_red['alcohol'], y =df_red['pH'])
from scipy.stats import pearsonr
corr, _ = pearsonr(df_red['alcohol'], df_red['pH'])
print('Pearsons correlation: %.3f' % corr)
```

```
Pearsons correlation: 0.206
```



In [12]:
```
# Q11. Create a correlation function that can give a correlation between any two columns
#alcohol and PH using this function.

def get_correlation(x,y):
    corr =x.corr(y)
```

```
    print(corr)

get_correlation(df_red['alcohol'], df_red['pH'])
```

0.20563250851242015

In [13]:
```
# Q12. Let's start with white wine analysis now. Load the white wine data frame.
df_white = pd.read_csv(r"C:\Users\hp\Downloads\winequality-white.csv")
```

In [14]:
```
# Q13. Find the average quality of both red wine and white wine.

print("red_wine average quality:",df_red['quality'].mean())
print("white_wine average quality:",df_white['quality'].mean())
```

```
red_wine average quality: 5.6360225140712945
white_wine average quality: 5.87790935075541
```

In [15]:
```
# Q14. Let's add a new attribute, wine_category, to both data frames.

df_red['wine_catagory']= "red wine"
df_white['wine_catagory'] = "white wine"
```

In [16]:
```
# Q15. let's find out what are the unique values of the column quality in both types of

print("red wine unique qualities:",df_red.quality.unique())
print("white wine unique qualities:",df_white.quality.unique())
```

```
red wine unique qualities: [5 6 7 4 8 3]
white wine unique qualities: [6 5 7 8 4 3 9]
```

In [17]:
```
# Q16. Although the quality column is numerical, here, we are interested in taking quali
# values into categorical values.
#Q17. To do so, we need a set of rules. Let's define a set of rules:
# quality_label- low if value <= 5, medium if value>5 and value <=7 and high if value >
# Write the code to add this quality_label column using the rule described above.
values=[]
for i in df_red['quality']:
    if i <= 5 :
        values.append("low")
    elif i>5 and i<=7 :
        values.append("medium")
    else :
        values.append("high")
df_red.insert(13, "quality_label", values)
```

In [18]:
```
# Q16 & 17 continue...
values_2=[]
for j in df_white['quality']:
    if j <= 5 :
        values_2.append("low")
    elif j>5 and j<=7 :
        values_2.append("medium")
    else :
        values_2.append("high")
df_white.insert(13, "quality_label", values_2)
df_white.head()
```

Out[18]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	wine_c
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6	wh
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6	wh
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6	wh

3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6	wh
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6	wh

In [19]:
```
# Q18. Count the number of values in each category of wine for column quality_label.
X = df_red.groupby(by="quality_label")
X.quality_label.count()
```

Out[19]:
```
quality_label
high        18
low        744
medium     837
Name: quality_label, dtype: int64
```

In [30]:
```
# Q19. Concatenate both red and white wine data frames now.
wine_comb = pd.concat([df_red,df_white], axis=0)
```

In [29]:
```
# Q20. Reshuffle the rows for randomization.
wine_comb.sample(frac=1)
```

Out[29]:

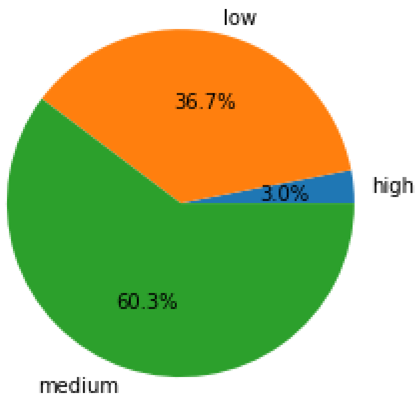
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	wi
4675	5.7	0.210	0.37	4.50	0.040	58.0	140.0	0.99332	3.29	0.62	10.6	6	
4673	6.0	0.280	0.52	6.20	0.028	37.0	104.0	0.99161	3.28	0.51	11.8	7	
224	8.4	0.635	0.36	2.00	0.089	15.0	55.0	0.99745	3.31	0.57	10.4	4	
2846	6.9	0.150	0.29	2.30	0.033	14.0	82.0	0.99132	3.10	0.58	11.2	7	
420	9.5	0.560	0.33	2.40	0.089	35.0	67.0	0.99720	3.28	0.73	11.8	7	
...	...	...	...	...	...	...	...	...	...	...	...	...	
757	6.8	0.220	0.37	15.20	0.051	68.0	178.0	0.99935	3.40	0.85	9.3	6	
401	6.8	0.370	0.51	11.80	0.044	62.0	163.0	0.99760	3.19	0.44	8.8	5	
4250	6.7	0.110	0.26	14.80	0.053	44.0	95.0	0.99676	3.20	0.35	9.8	6	
265	6.9	0.290	0.40	19.45	0.043	36.0	156.0	0.99960	2.93	0.47	8.9	5	
318	5.9	0.300	0.47	7.85	0.030	19.0	133.0	0.99330	3.52	0.43	11.5	7	

6497 rows × 14 columns

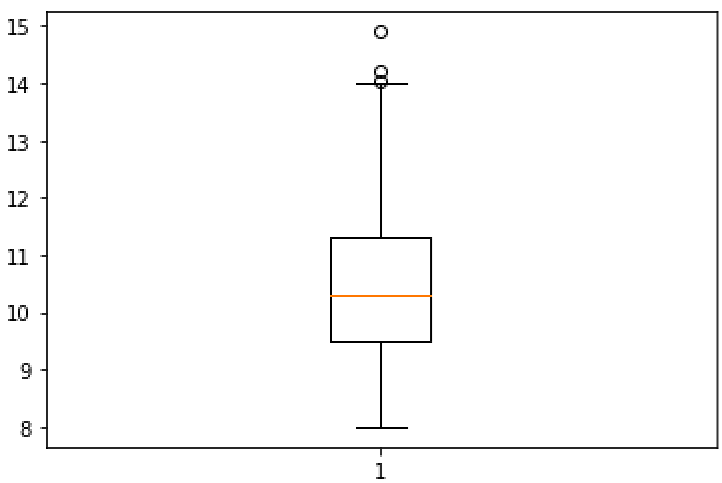
In [39]:
```
#Q21. Let's use the combined data frame and group them using the columns, alcohol, densi
#Hint: - create a subset of attributes that we are interested in. Then, create three dif
#and high-quality wine. Finally, concatenate them.

# creating 3 diff dataframe:
low_quality_wine= wine_comb.loc[wine_comb.quality_label=="low"]
medium_quality_wine= wine_comb.loc[wine_comb.quality_label=="medium"]
high_quality_wine= wine_comb.loc[wine_comb.quality_label=="high"]

#concatenating:
All_wines= pd.concat([low_quality_wine,medium_quality_wine,high_quality_wine], axis=0)
```

In [45]:
```
# Q22. How will you do a univariate analysis for numerical data . Apply the logic to do

# analysis by quality
qualities = dict(All_wines.groupby(['quality_label']).size())
qualities
```

```
quality_list =[]
quality_count=[]
for i in qualities:
    quality_count.append(i),
    quality_list.append(qualities[i])
plt.pie(quality_list, labels=quality_count, autopct='%0.1f%%')
plt.show()
```

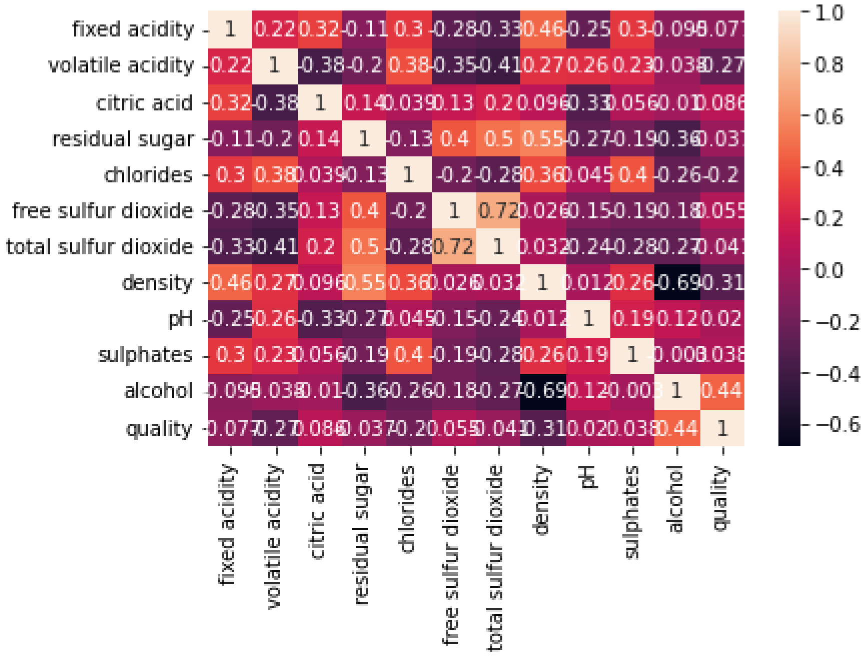


In [49]:
```
# continue univariate analysis
# alcohol Perecntage wise analysis:
plt.boxplot(wine_comb['alcohol'])
plt.show()
```



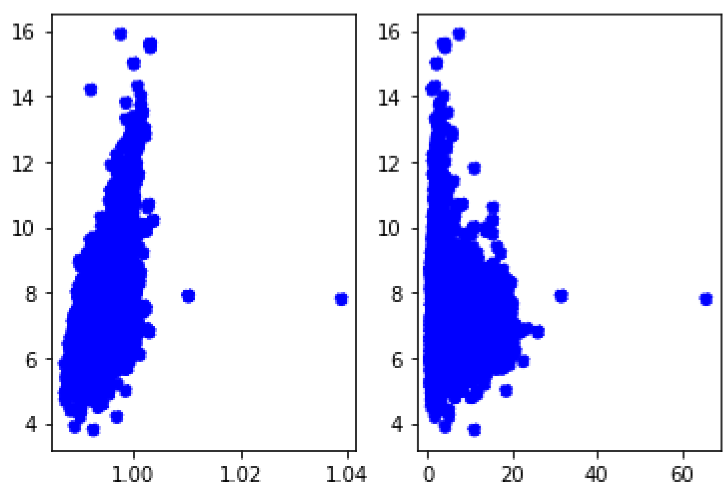
In [50]:
```
# Q23. Do a multivariate analysis for all the columns using heatmap.

sns.heatmap(All_wines.corr(),annot=True)

# we can see there not much columns are correlated.
```

Out[50]:
```
<AxesSubplot:>
```

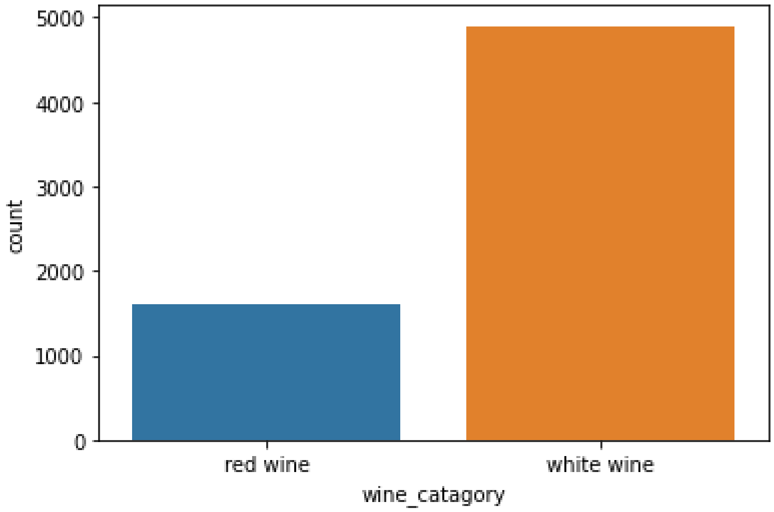


In [53]:
```
# multivariate analysis continue

plt.subplot(1,2,1)
plt.scatter(All_wines['density'], All_wines['fixed acidity'],linestyle='--',color='b')
plt.subplot(1,2,2)
plt.scatter(All_wines['residual sugar'], All_wines['fixed acidity'],linestyle='--',color
```

Out[53]:
```
<matplotlib.collections.PathCollection at 0x1b6631526e0>
```



In [56]:
```
# Q24. Draw a count plot for wine_category.

sns.countplot(x = 'wine_catagory', data = All_wines)
plt.show()
```


5000
4000
3000
2000
1000
0
count
red wine
white wine
wine_catagory

In [ ]:
```
## thank you.
```