In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
data=pd.read_csv('WINE DATA\\winequality-red.csv',delimiter = ";")
```

In [3]:

```
data
```

Out[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

1599 rows × 12 columns

In [4]:

```
data.columns
```

Out[4]:

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

In [5]:

```
data.describe()
```

Out[5]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulph
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.00
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.65
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.16
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.33
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.55
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.62

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulph
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.73
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.00

In [6]:

```
print("Total quantity of wine quality 3 : ",len([a for a in data['quality'] if a==3]))
print("Total quantity of wine quality 4 : ",len([a for a in data['quality'] if a==4]))
print("Total quantity of wine quality 5 : ",len([a for a in data['quality'] if a==5]))
print("Total quantity of wine quality 6 : ",len([a for a in data['quality'] if a==6]))
print("Total quantity of wine quality 7 : ",len([a for a in data['quality'] if a==7]))
print("Total quantity of wine quality 8 : ",len([a for a in data['quality'] if a==8]))
```

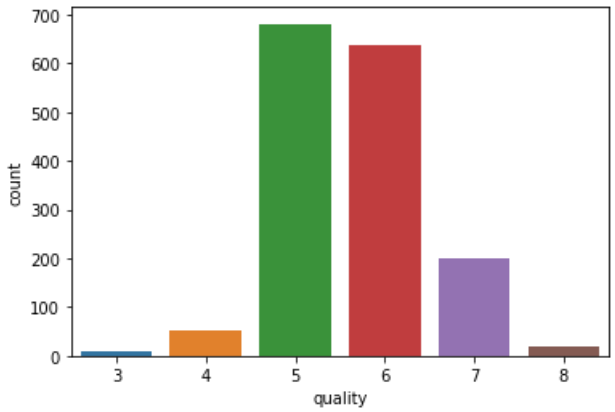
```
Total quantity of wine quality 3 :  10
Total quantity of wine quality 4 :  53
Total quantity of wine quality 5 :  681
Total quantity of wine quality 6 :  638
Total quantity of wine quality 7 :  199
Total quantity of wine quality 8 :  18
```

In [7]:

```
sns.countplot(x='quality', data=data)
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2d388c17eb8>
```



In [8]:

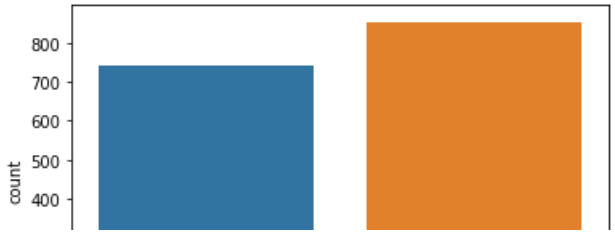
```
reviews = []
for i in data['quality']:
    if i <= 5:
        reviews.append(0)
    else:
        reviews.append(1)
data['Reviews'] = reviews
```

In [9]:

```
sns.countplot(x='Reviews', data=data)
```

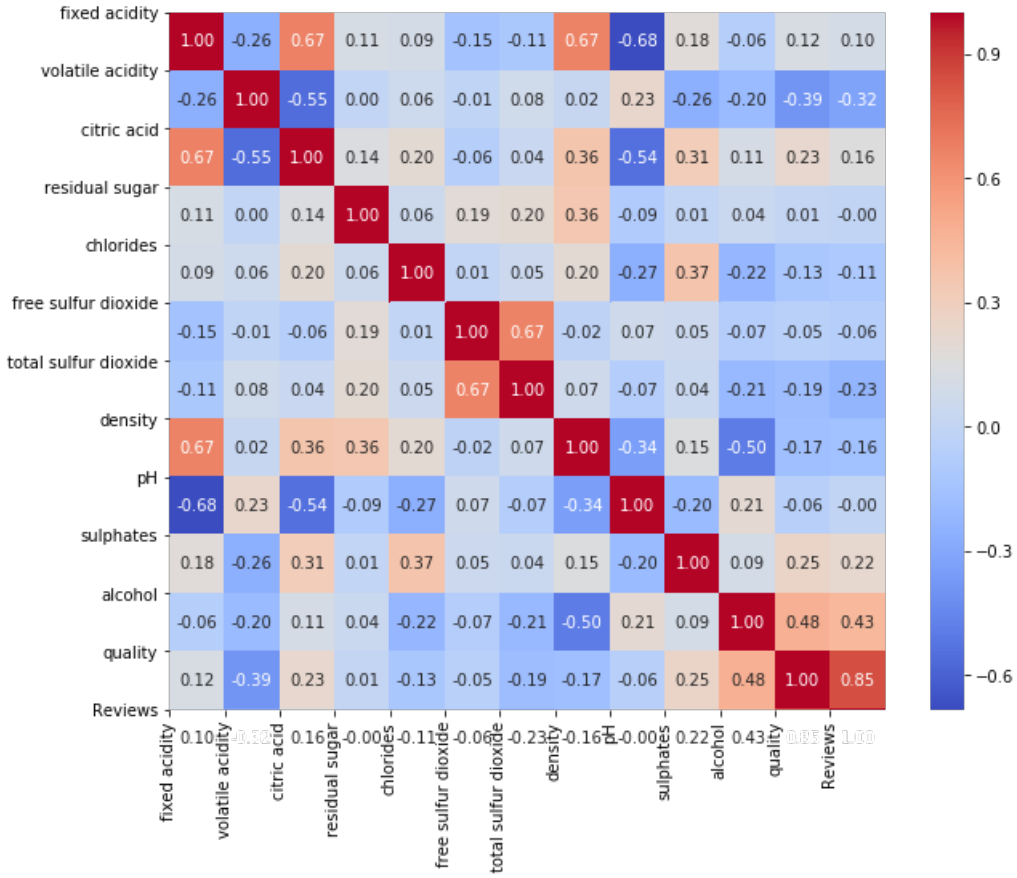
Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2d38a2315c0>
```



In [10]:

```
corr = data.corr()
#Plot figsize
fig, ax = plt.subplots(figsize=(10, 8))
#Generate Heat Map, allow annotations and place floats in map
sns.heatmap(corr, cmap='coolwarm', annot=True, fmt=".2f")
#Apply xticks
plt.xticks(range(len(corr.columns)), corr.columns);
#Apply yticks
plt.yticks(range(len(corr.columns)), corr.columns)
#show plot
plt.show()
```



In [11]:

```
X = data.iloc[:,0:11]
y = data.iloc[:,12]
y_mul = data.iloc[:,11]
```

In [14]:

```
from sklearn.preprocessing import MinMaxScaler
minmax = MinMaxScaler()
X_s = minmax.fit_transform(X)
```

In [15]:

```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(X_s,y,test_size = 0.3,random_state = 42)
```

In [16]:

```
print(X_train.shape)
print(X_test.shape)
```

```
(1119, 11)
(480, 11)
```

In [17]:

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```

In [18]:

```
svc_lin  = SVC(kernel = 'linear')
svc_lin.fit(X_train, y_train)
y_pred_svc_lin=svc_lin.predict(X_test)
lin_svc_conf_matrix = confusion_matrix(y_test, y_pred_svc_lin)
lin_svc_clf = classification_report(y_test, y_pred_svc_lin)
print(lin_svc_conf_matrix)
print(lin_svc_clf)
```

```
[[164  49]
 [ 87 180]]
              precision    recall  f1-score   support

           0       0.65      0.77      0.71       213
           1       0.79      0.67      0.73       267

    accuracy                           0.72       480
   macro avg       0.72      0.72      0.72       480
weighted avg       0.73      0.72      0.72       480
```

In [32]:

```
svc_rbf  = SVC(kernel = 'rbf')
svc_rbf.fit(X_train, y_train)
y_pred_svc_rbf=svc_rbf.predict(X_test)
rbf_svc_conf_matrix = confusion_matrix(y_test, y_pred_svc_rbf)
rbf_svc_clf = classification_report(y_test, y_pred_svc_rbf)
print(rbf_svc_conf_matrix)
print(rbf_svc_clf)
```

```
[[161  52]
 [ 67 200]]
              precision    recall  f1-score   support

           0       0.71      0.76      0.73       213
           1       0.79      0.75      0.77       267

    accuracy                           0.75       480
   macro avg       0.75      0.75      0.75       480
weighted avg       0.75      0.75      0.75       480
```

In [33]:

```
svc_poly  = SVC(kernel = 'poly')
svc_poly.fit(X_train, y_train)
y_pred_svc_poly =  svc_poly.predict(X_test)
poly_svc_conf_matrix = confusion_matrix(y_test, y_pred_svc_poly)
poly_svc_clf = classification_report(y_test, y_pred_svc_poly)
print(poly_svc_conf_matrix)
print(poly_svc_clf)
```

```
[[168  45]
```

```
 [ 72 195]]
              precision    recall  f1-score   support

           0       0.70      0.79      0.74       213
           1       0.81      0.73      0.77       267

    accuracy                           0.76       480
   macro avg       0.76      0.76      0.76       480
weighted avg       0.76      0.76      0.76       480
```

In [34]:

```
svc_sigmoid  = SVC(kernel = 'sigmoid')
svc_sigmoid.fit(X_train, y_train)
y_pred_svc_sigmoid=svc_sigmoid.predict(X_test)
sigmoid_svc_conf_matrix = confusion_matrix(y_test, y_pred_svc_sigmoid)
sigmoid_svc_clf = classification_report(y_test, y_pred_svc_sigmoid)
print(sigmoid_svc_conf_matrix)
print(sigmoid_svc_clf)
```

```
[[109 104]
 [110 157]]
              precision    recall  f1-score   support

           0       0.50      0.51      0.50       213
           1       0.60      0.59      0.59       267

    accuracy                           0.55       480
   macro avg       0.55      0.55      0.55       480
weighted avg       0.56      0.55      0.55       480
```

In [41]:

```
from sklearn.model_selection import cross_val_score
clf = SVC(kernel='linear', C=1)
scores = cross_val_score(clf, X_s, y, cv=5)
print(scores)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
[0.69375    0.690625   0.778125   0.75625    0.73981191]
Accuracy: 0.73 (+/- 0.07)
```

In [42]:

```
from sklearn.model_selection import cross_val_score
clf = SVC(kernel='rbf', C=1)
scores = cross_val_score(clf, X_s, y, cv=5)
print(scores)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
[0.690625   0.684375   0.790625   0.7375     0.74294671]
Accuracy: 0.73 (+/- 0.08)
```

In [43]:

```
from sklearn.model_selection import cross_val_score
clf = SVC(kernel='poly',C=1)
scores = cross_val_score(clf, X_s, y, cv=5)
print(scores)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
[0.7125     0.690625   0.8        0.76875    0.72413793]
Accuracy: 0.74 (+/- 0.08)
```

In [44]:

```
from sklearn.model_selection import cross_val_score
clf = SVC(kernel='sigmoid',C=1)
```

```
scores = cross_val_score(clf, X_s, y, cv=5)
print(scores)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
[0.66875    0.53125    0.671875   0.571875   0.57053292]
Accuracy: 0.60 (+/- 0.11)
```

In [ ]:

```
```