In [4]:
```python
# IMPORT ALL THE NECESSARY LIBRARIES

import warnings
warnings.filterwarnings(action="ignore")

import pandas as pd
import numpy as np


import matplotlib.pyplot as plt
import seaborn as sns
from time import time


from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [5]:
```python
# DATA PREPARATION

pd.set_option('display.max_rows', 12)
pd.set_option('display.max_column', None)

#Loading the Red Wines dataset
df = pd.read_csv("winequality-red.csv" ,sep=';')

# Display first few records
print(df.head(5))

# Shape of the dataset
print(df.shape)
```

```
   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
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5
(1599, 12)
```

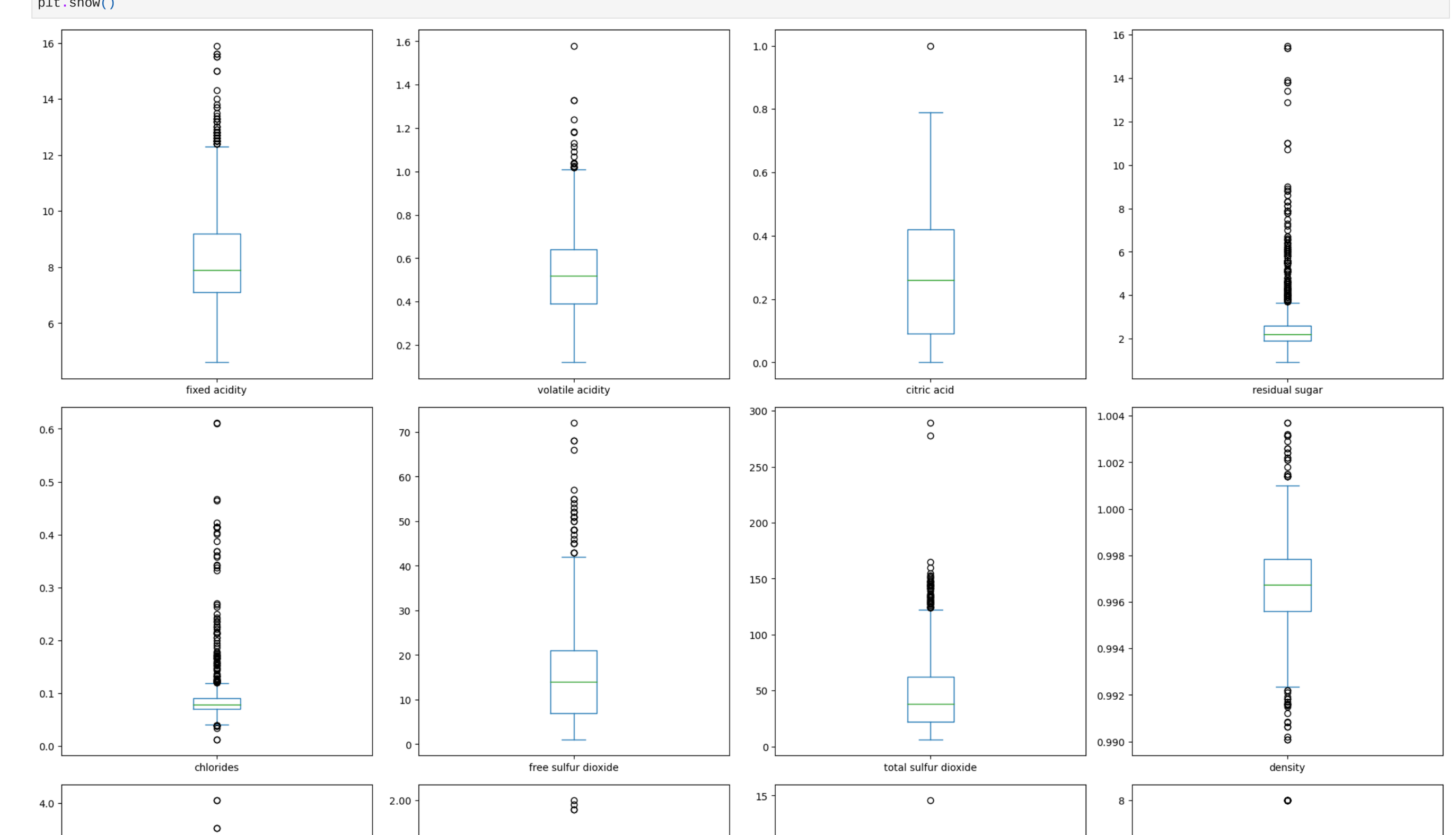
In [6]:
```python
#checking for the null values in the dataset


print(df.isnull().sum())
print("\n\n")
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

In [7]:
```python
# checking for the outlier

fig, axes = plt.subplots(figsize=(20, 16))
df.plot( kind='box', subplots=True,
                            layout=(3,4) ,
                            sharex=False,
                            sharey=False , ax=axes)
plt.tight_layout()
plt.show()
```



In [8]:
```python
#EXPLORATORY DATA ANALYSIS(EDA)


#DATA ANALYSIS AND VASULIZATION

print(df.describe())
print("\n\n")
```

```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    1599.000000       1599.000000  1599.000000     1599.000000
mean        8.319637          0.527821     0.270976        2.538806
std         1.741096          0.179060     0.194801        1.409928
min         4.600000          0.120000     0.000000        0.900000
25%         7.100000          0.390000     0.090000        1.900000
50%         7.900000          0.520000     0.260000        2.200000
75%         9.200000          0.640000     0.420000        2.600000
max        15.900000          1.580000     1.000000       15.500000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  1599.000000          1599.000000           1599.000000  1599.000000
mean      0.087467            15.874922             46.467792     0.996747
std       0.047065            10.460157             32.895324     0.001887
min       0.012000             1.000000              6.000000     0.990070
25%       0.070000             7.000000             22.000000     0.995600
50%       0.079000            14.000000             38.000000     0.996750
75%       0.090000            21.000000             62.000000     0.997835
max       0.611000            72.000000            289.000000     1.003690

                pH    sulphates      alcohol      quality
count  1599.000000  1599.000000  1599.000000  1599.000000
mean      3.311113     0.658149    10.422983     5.636023
std       0.154386     0.169507     1.065668     0.807569
min       2.740000     0.330000     8.400000     3.000000
25%       3.210000     0.550000     9.500000     5.000000
50%       3.310000     0.620000    10.200000     6.000000
75%       3.400000     0.730000    11.100000     6.000000
max       4.010000     2.000000    14.900000     8.000000
```

In [9]:
```python
#checking for duplicate records
duplicate=df.duplicated()
print(duplicate.sum())
df[duplicate]
```

240

Out[9]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
4	7.4	0.700	0.00	1.90	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
11	7.5	0.500	0.36	6.10	0.071	17.0	102.0	0.99780	3.35	0.80	10.5	5
27	7.9	0.430	0.21	1.60	0.106	10.0	37.0	0.99660	3.17	0.91	9.5	5
40	7.3	0.450	0.36	5.90	0.074	12.0	87.0	0.99780	3.33	0.83	10.5	5
65	7.2	0.725	0.05	4.65	0.086	4.0	11.0	0.99620	3.41	0.39	10.9	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1563	7.2	0.695	0.13	2.00	0.076	12.0	20.0	0.99546	3.29	0.54	10.1	5
1564	7.2	0.695	0.13	2.00	0.076	12.0	20.0	0.99546	3.29	0.54	10.1	5
1567	7.2	0.695	0.13	2.00	0.076	12.0	20.0	0.99546	3.29	0.54	10.1	5
1581	6.2	0.560	0.09	1.70	0.053	24.0	32.0	0.99402	3.54	0.60	11.3	5
1596	6.3	0.510	0.13	2.30	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6

240 rows × 12 columns

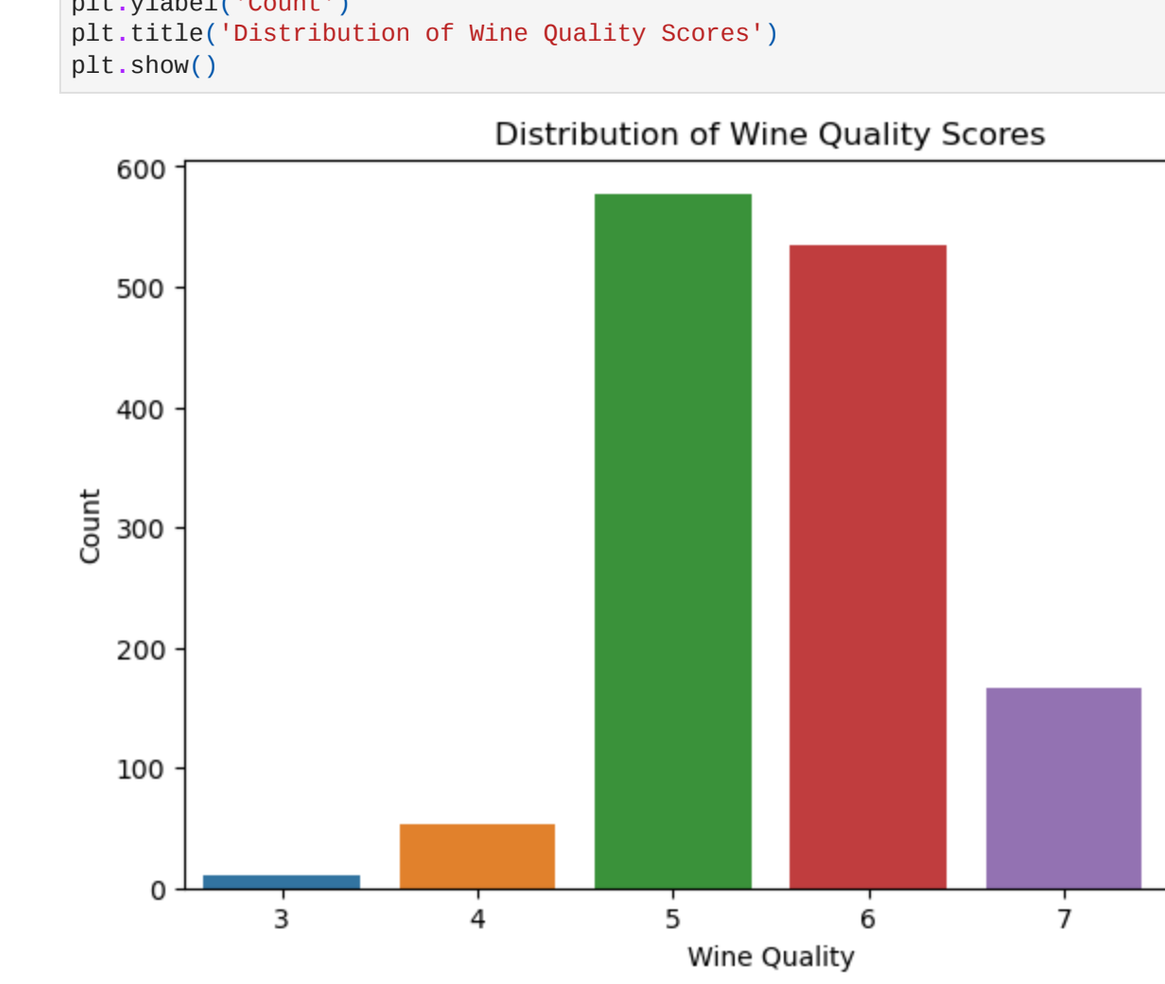
In [10]:
```python
#call drop_duplicate

df.drop_duplicates(inplace=True)
print(df.shape)
```

(1359, 12)

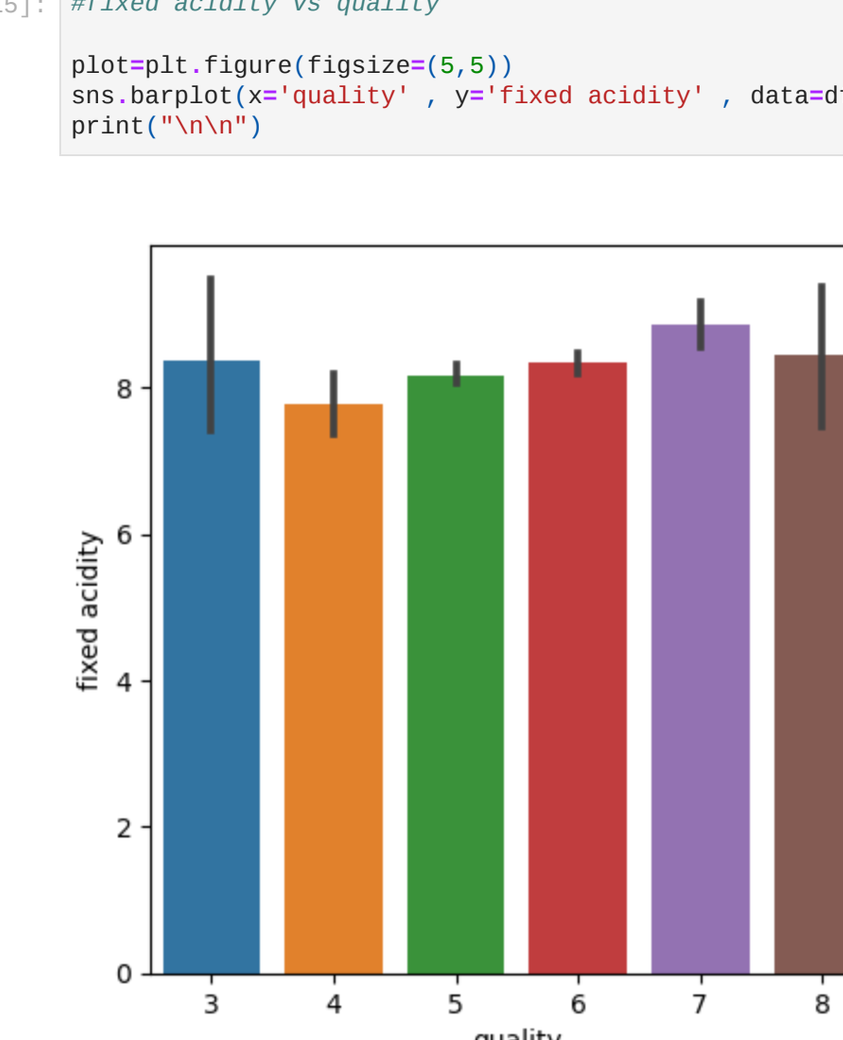
In [11]:
```python
# distribution of wine quality scores

plt.figure(figsize=(8, 5))
sns.countplot(x='quality', data=df)
plt.xlabel('Wine Quality')
plt.ylabel('Count')
plt.title('Distribution of Wine Quality Scores')
plt.show()
```



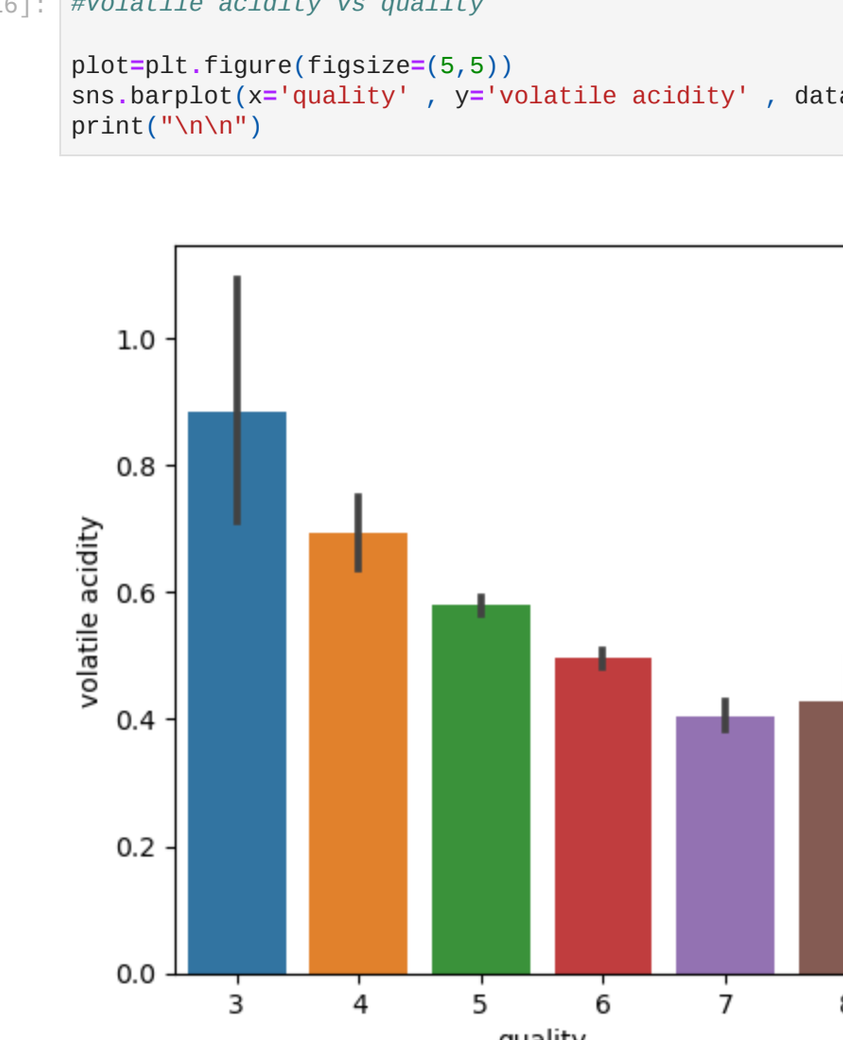
In [15]:
```python
#fixed acidity vs quality

plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality' , y='fixed acidity' , data=df)
print("\n\n")
```



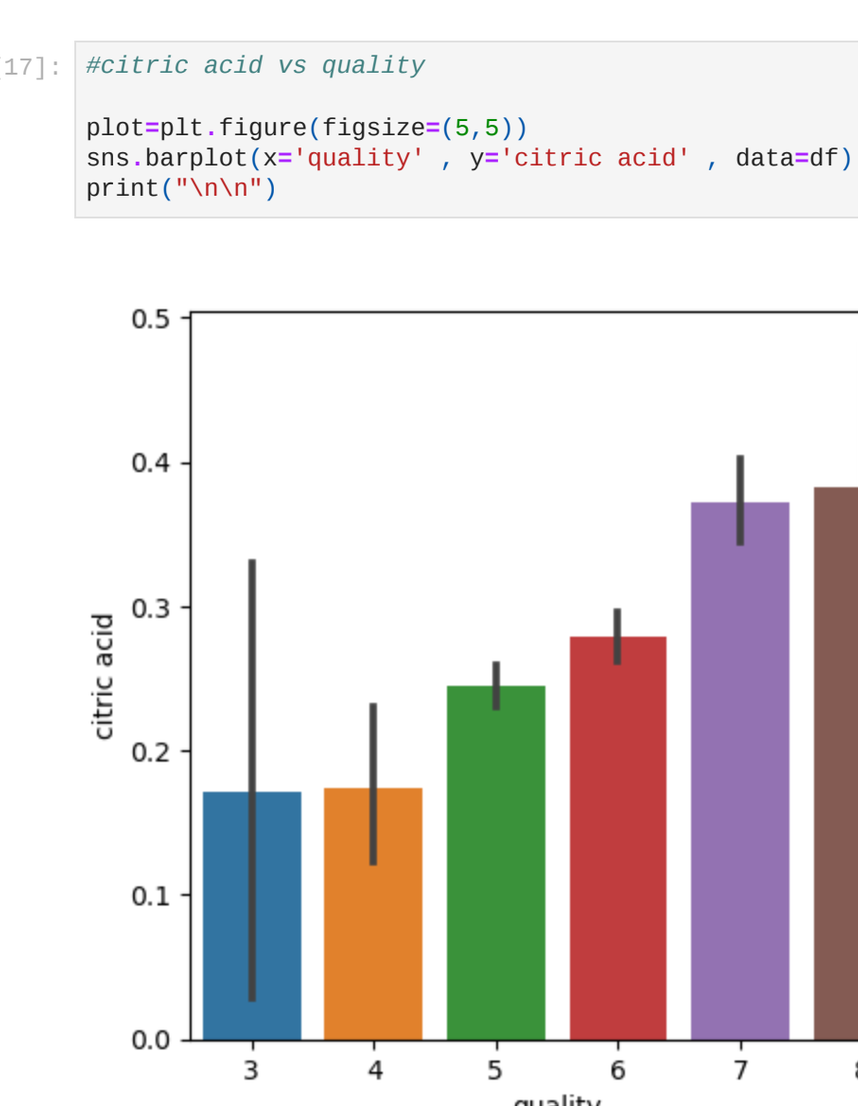
In [16]:
```python
#Volatile acidity vs quality

plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality' , y='volatile acidity' , data=df)
print("\n\n")
```



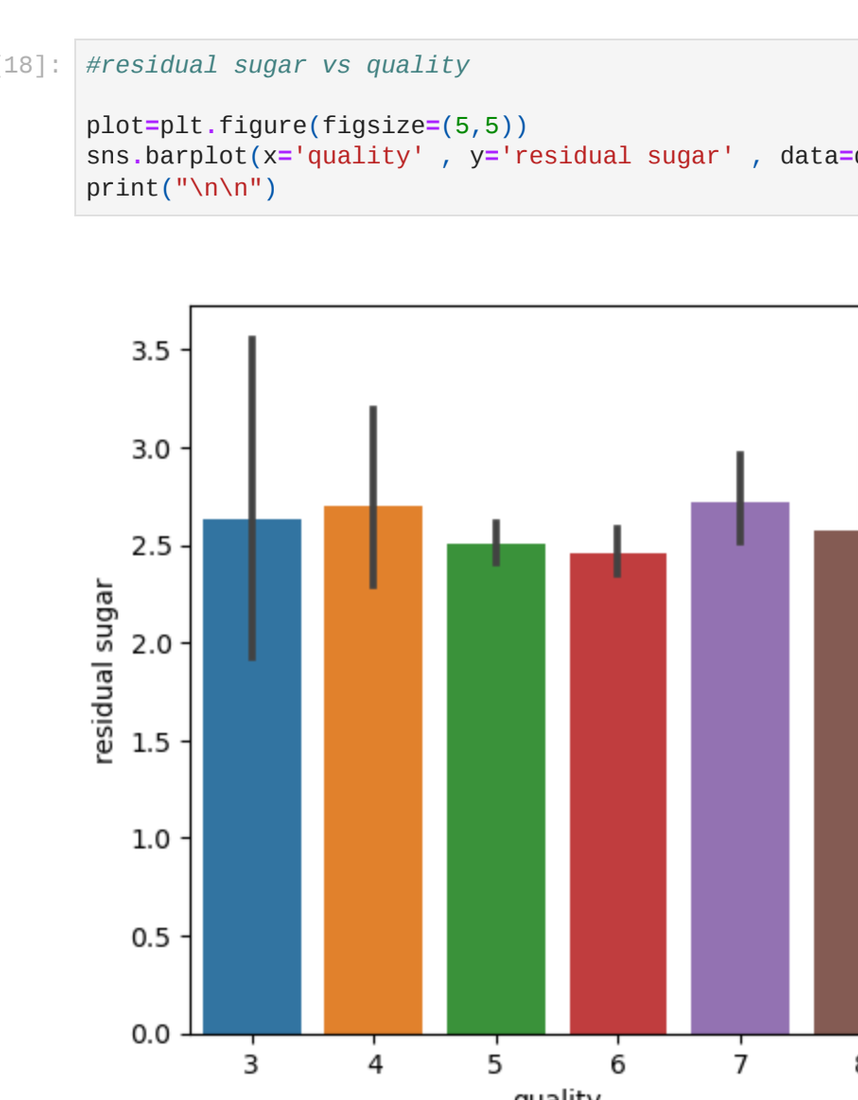
In [17]:
```python
#citric acid vs quality

plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality' , y='citric acid' , data=df)
print("\n\n")
```



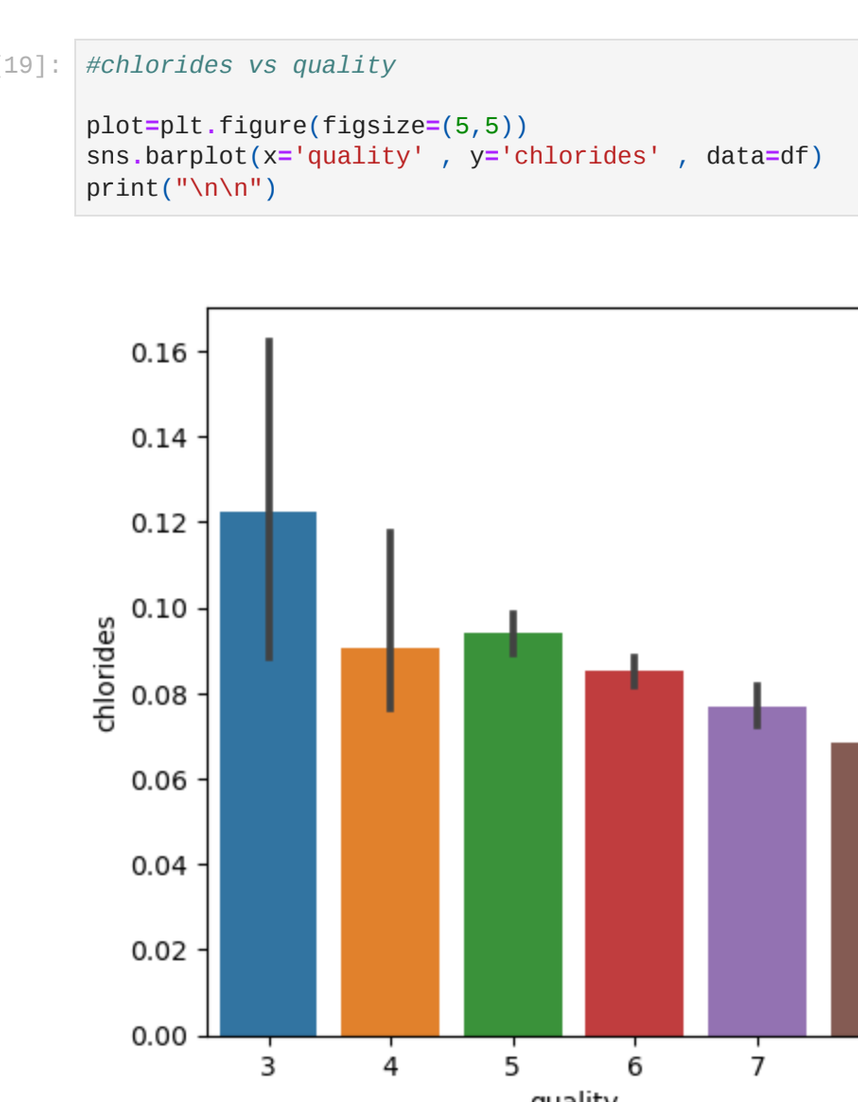
In [18]:
```python
#residual sugar vs quality

plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality' , y='residual sugar' , data=df)
print("\n\n")
```



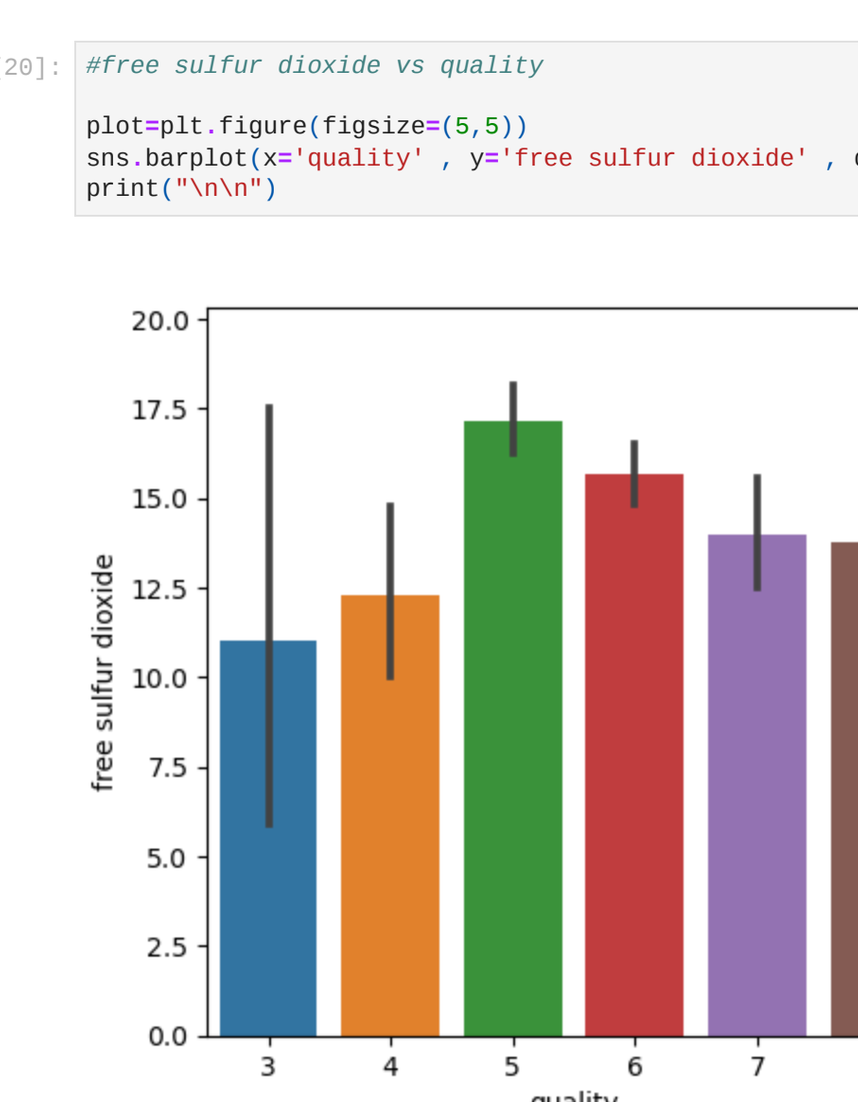
In [19]:
```python
#chlorides vs quality

plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality' , y='chlorides' , data=df)
print("\n\n")
```



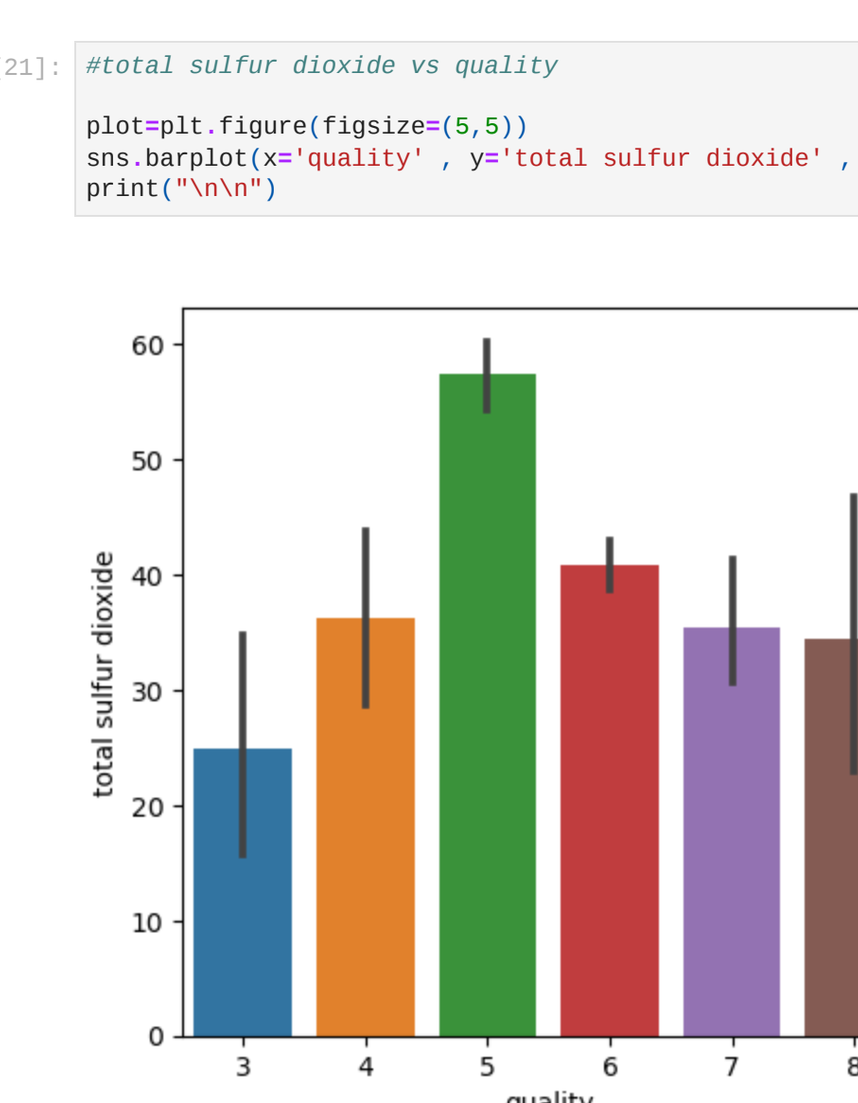
In [20]:
```python
#free sulfur dioxide vs quality

plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality' , y='free sulfur dioxide' , data=df)
print("\n\n")
```



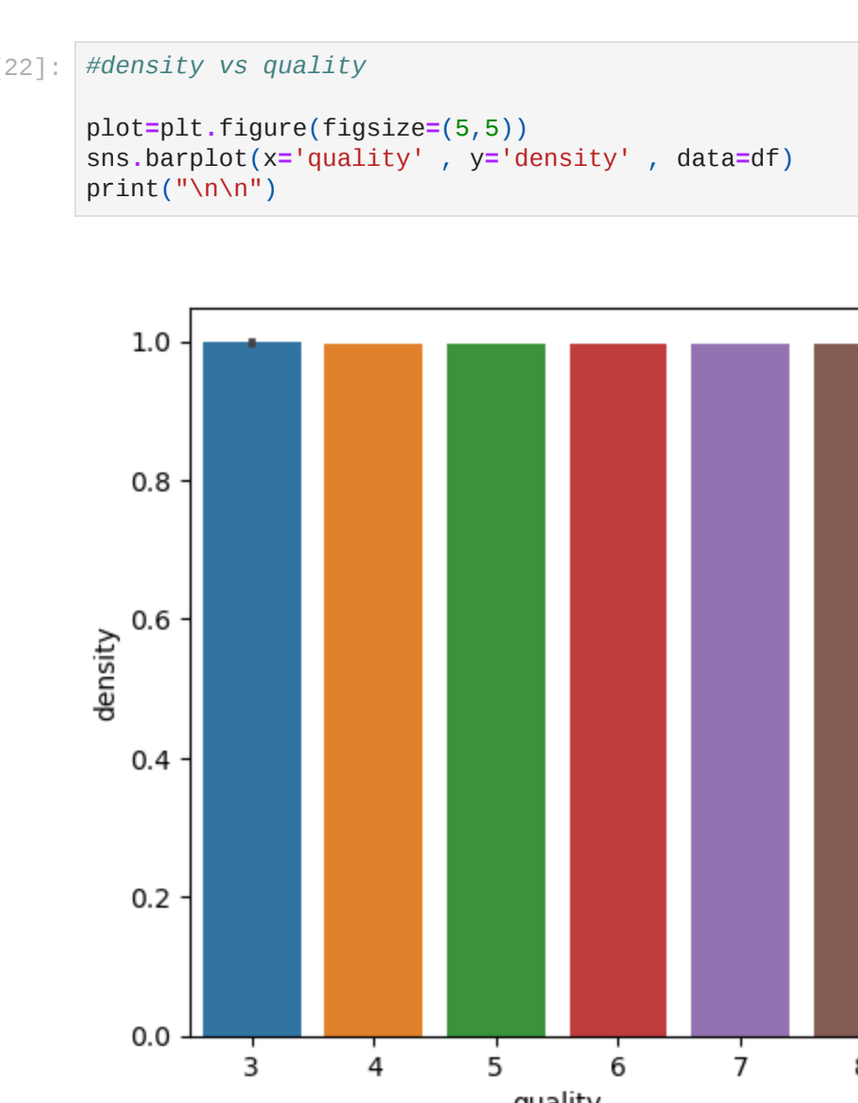
In [21]:
```python
#total sulfur dioxide vs quality

plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality' , y='total sulfur dioxide' , data=df)
print("\n\n")
```



In [22]:
```python
#density vs quality

plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality' , y='density' , data=df)
print("\n\n")
```

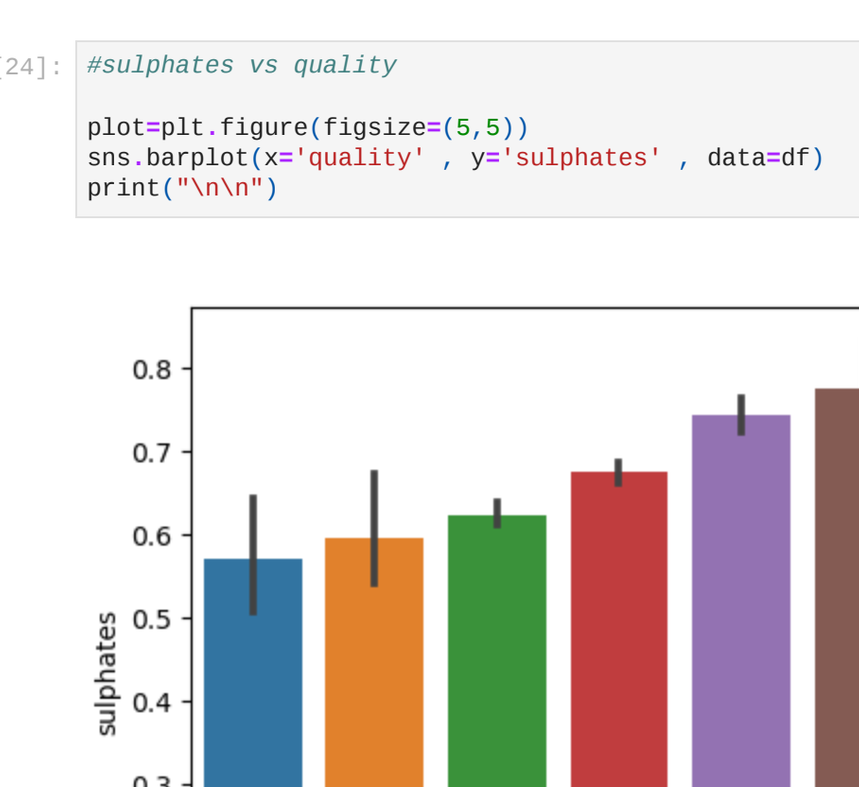


In [23]:
```python
#pH vs quality

plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality' , y='pH' , data=df)
print("\n\n")
```

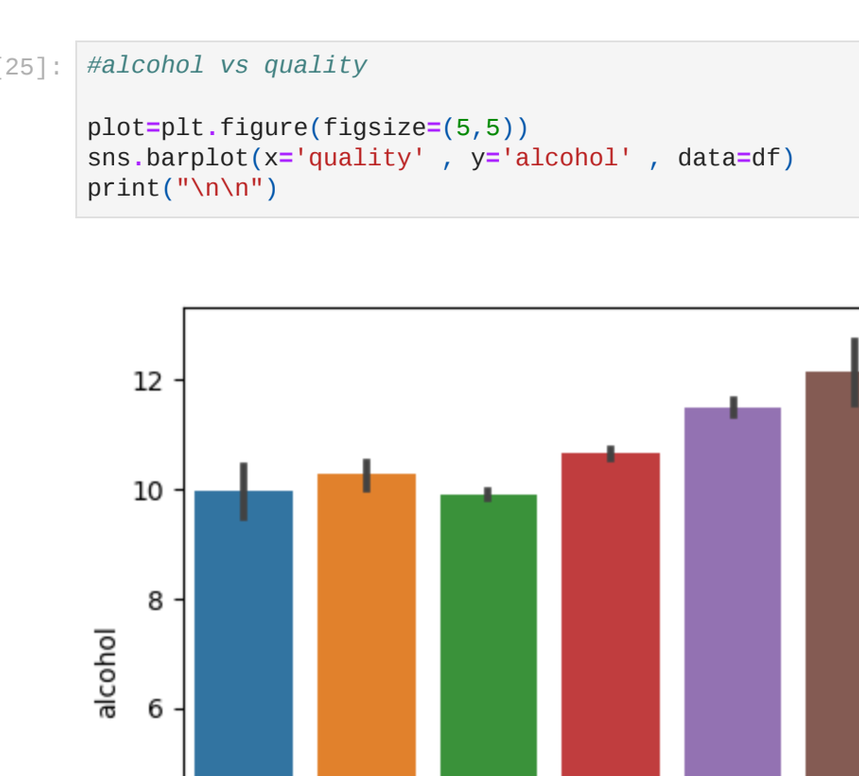
In [24]:
```python
#sulphates vs quality

plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality' , y='sulphates' , data=df)
print("\n\n")
```



In [25]:
```python
#alcohol vs quality

plot=plt.figure(figsize=(5,5))
sns.barplot(x='quality' , y='alcohol' , data=df)
print("\n\n")
```

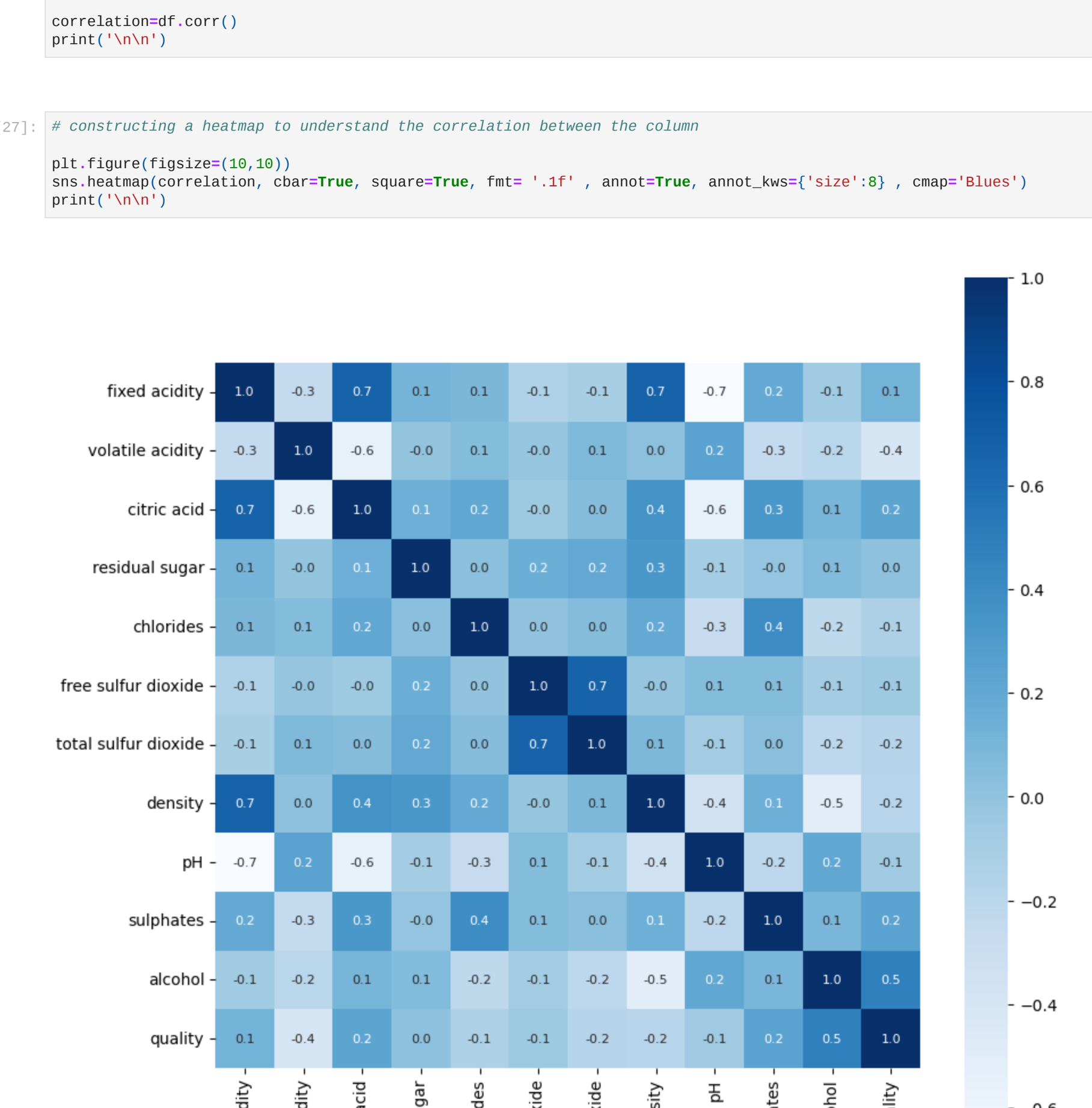


In [26]:
```python
# CORRELATION BETWEEN ALL THE COLUMN AND QUALITY

correlation=df.corr()
print('\n\n')
```

In [27]:
```python
# constructing a heatmap to understand the correlation between the column

plt.figure(figsize=(10,10))
sns.heatmap(correlation, cbar=True, square=True, fmt= '.1f' , annot=True, annot_kws={'size':8} , cmap='Blues')
print('\n\n')
```



In [28]:
```python
# data preprocessing

# separate the data and label

X=df.drop('quality',axis=1)
print(X)
print('\n\n')
```

```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0               7.4             0.700         0.00             1.9      0.076
1               7.8             0.880         0.00             2.6      0.098
2               7.8             0.760         0.04             2.3      0.092
3              11.2             0.280         0.56             1.9      0.075
5               7.4             0.660         0.00             1.8      0.075
...             ...               ...          ...             ...        ...
1593            6.8             0.620         0.08             1.9      0.068
1594            6.2             0.600         0.08             2.0      0.090
1595            5.9             0.550         0.10             2.2      0.062
1597            5.9             0.645         0.12             2.0      0.075
1598            6.0             0.310         0.47             3.6      0.067

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                    11.0                  34.0  0.99780  3.51       0.56
1                    25.0                  67.0  0.99680  3.20       0.68
2                    15.0                  54.0  0.99700  3.26       0.65
3                    17.0                  60.0  0.99800  3.16       0.58
5                    13.0                  40.0  0.99780  3.51       0.56
...                   ...                   ...      ...   ...        ...
1593                 28.0                  38.0  0.99651  3.42       0.82
1594                 32.0                  44.0  0.99490  3.45       0.58
1595                 39.0                  51.0  0.99512  3.52       0.76
1597                 32.0                  44.0  0.99547  3.57       0.71
1598                 18.0                  42.0  0.99549  3.39       0.66

      alcohol
0         9.4
1         9.8
2         9.8
3         9.8
5         9.4
...       ...
1593      9.5
1594     10.5
1595     11.2
1597     10.2
1598     11.0

[1359 rows x 11 columns]
```

In [29]:
```python
# LABEL BINARIZATION

Y=df['quality'].apply(lambda y_value:1 if y_value>=7 else 0)
print(Y)
print('\n\n')
```

```
0       0
1       0
2       0
3       0
5       0
       ..
1593    0
1594    0
1595    0
1597    0
1598    0
Name: quality, Length: 1359, dtype: int64
```

In [46]:
```python
# MODEL BUILDING

# TRAIN AND TEST SPLIT

X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=7)
# here i have taken test_size=0.2 as to split the data into 20:80 testing and training data
# here i have taken random_state = 7 as it gives maximum accuracy
print(Y.shape,Y_train.shape,Y_test.shape)
```

(1359,) (1087,) (272,)

In [42]:
```python
#training model

model=LogisticRegression()
model.fit(X_train,Y_train)
```

Out[42]:

▾ LogisticRegression

LogisticRegression()

In [47]:
```python
#Evaluation of the model

Y_pred=model.predict(X_test)
accuracy=accuracy_score(Y_test,Y_pred)
print(f"accuracy: {accuracy}")
print("\n\n")
print("confusion matrix:")
print(confusion_matrix(Y_test,Y_pred),'\n\n')
print("Classification Report:")
print(classification_report(Y_test,Y_pred))
```

```
accuracy: 0.9117647058823529



confusion matrix:
[[231   5]
 [ 19  17]]


Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.98      0.95       236
           1       0.77      0.47      0.59        36

    accuracy                           0.91       272
   macro avg       0.85      0.73      0.77       272
weighted avg       0.90      0.91      0.90       272
```

In [33]:
```python
# MODEL EVALUATION


from sklearn.metrics import mean_squared_error, r2_score


# Calculate MSE and RMSE
mse = mean_squared_error(Y_test, Y_pred)
rmse = np.sqrt(mse)

# Calculate R-squared
r_squared = r2_score(Y_test, Y_pred)

print(f"MSE: {mse}")
print(f"RMSE: {rmse}")
print(f"R-squared: {r_squared}")
```

```
MSE: 0.13602941176470587
RMSE: 0.36882165305836623
R-squared: -0.08448275862068955
```

In [ ]: