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

In [2]:
```
df=pd.read_csv("/home/aparna/Downloads/winequality-red.csv")
df.head(10)
```

Out[2]:

	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
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4	5
6	7.9	0.60	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4	5
7	7.3	0.65	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0	7
8	7.8	0.58	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5	7
9	7.5	0.50	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5	5

In [3]:
```
df.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
```

In [4]:
```
df.shape
```

Out[4]:
```
(1599, 12)
```

In [5]:
```
df.describe()   #find the statistical measures
```

Out[5]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	densit
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.00000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.99674

std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.0018
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.99007
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.9956
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.99675
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.9978
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.0036

In [6]:
```
df.isna().sum()
```

Out[6]:
```
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
```

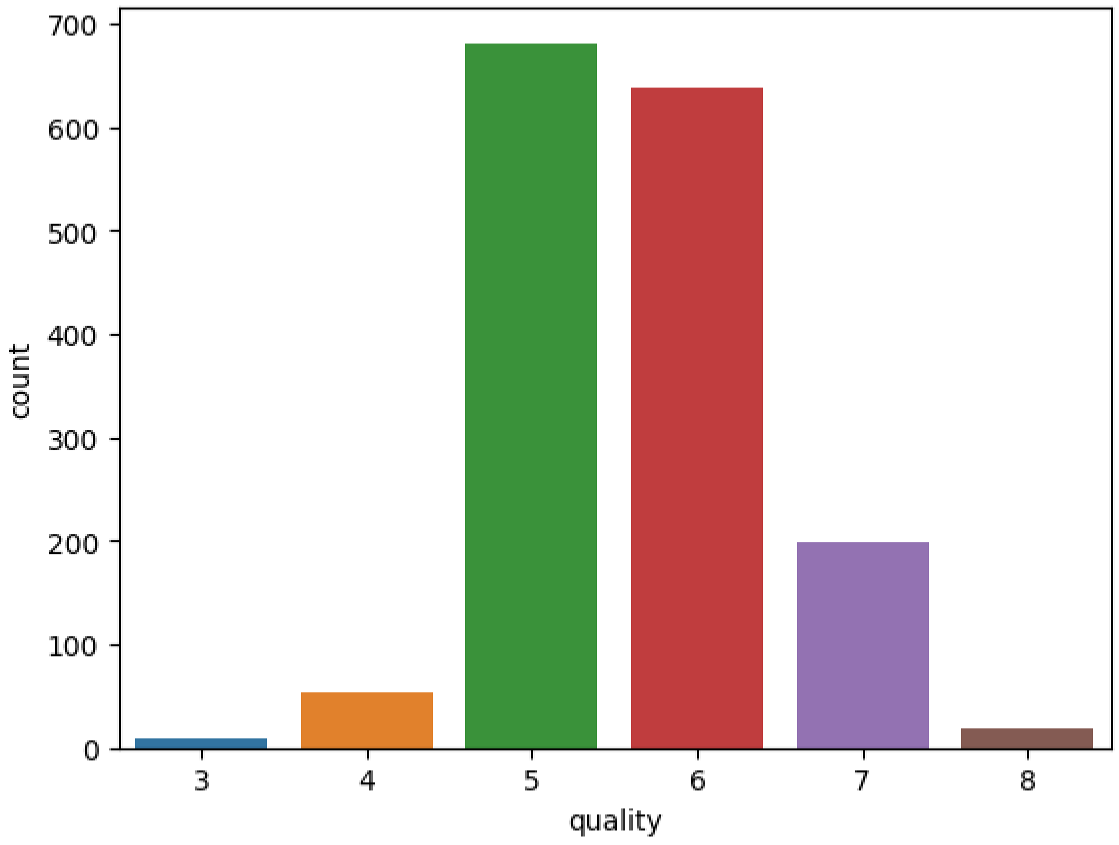
In [7]:
```
df['quality'].value_counts()
```

Out[7]:
```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64
```

In [8]:
```
#number of values for each quality
sns.countplot('quality',data=df)
```

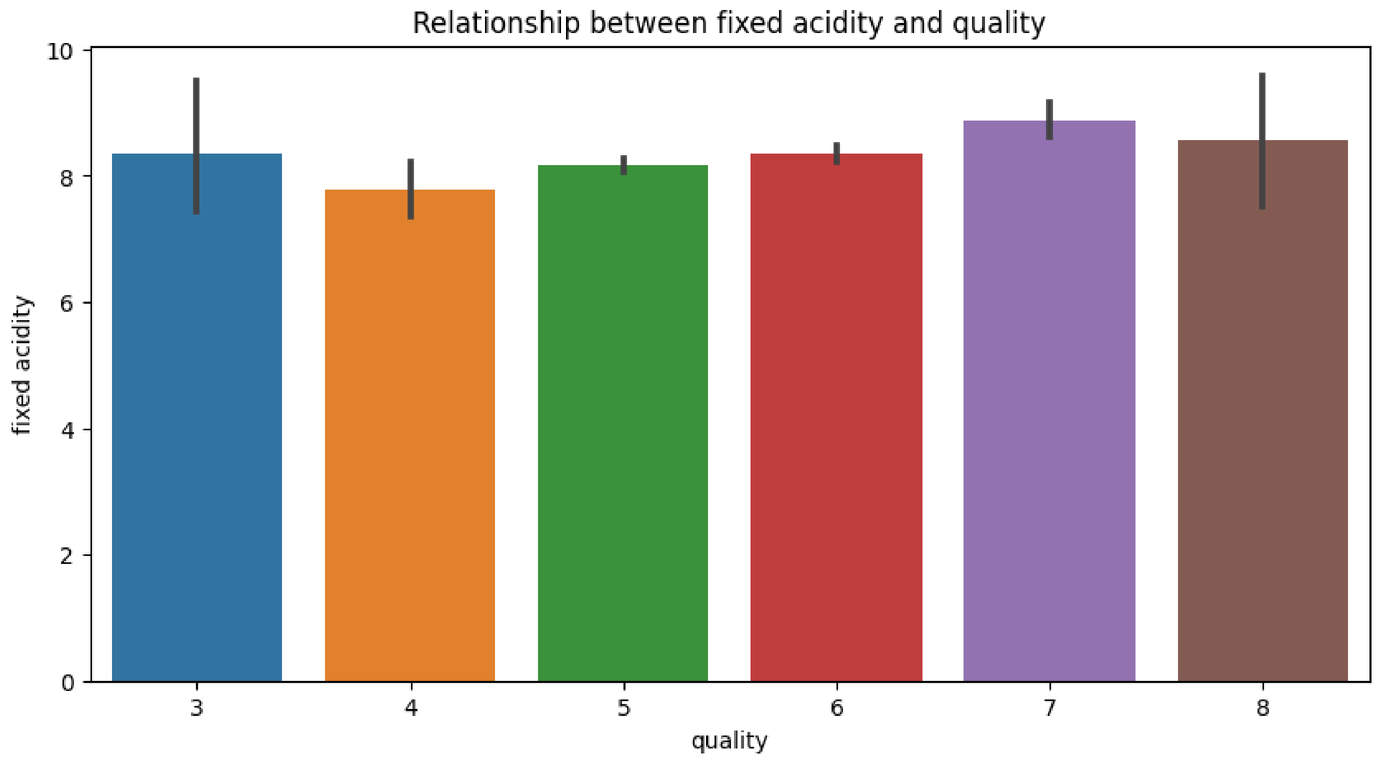
```
/home/aparna/.local/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

Out[8]:
```
<AxesSubplot:xlabel='quality', ylabel='count'>
```



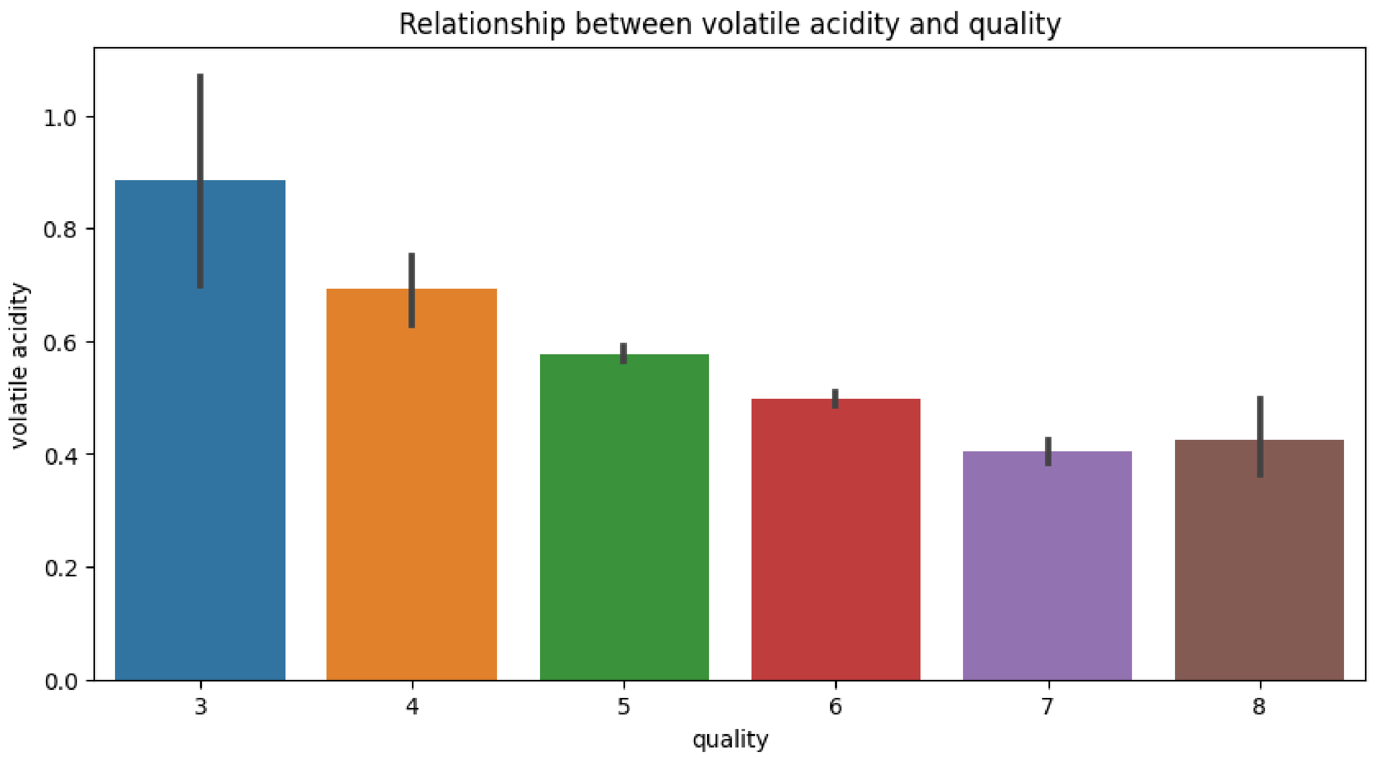
In [9]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='quality',y='fixed acidity',data=df)
plt.title('Relationship between fixed acidity and quality')
```

Out[9]:
```
Text(0.5, 1.0, 'Relationship between fixed acidity and quality')
```



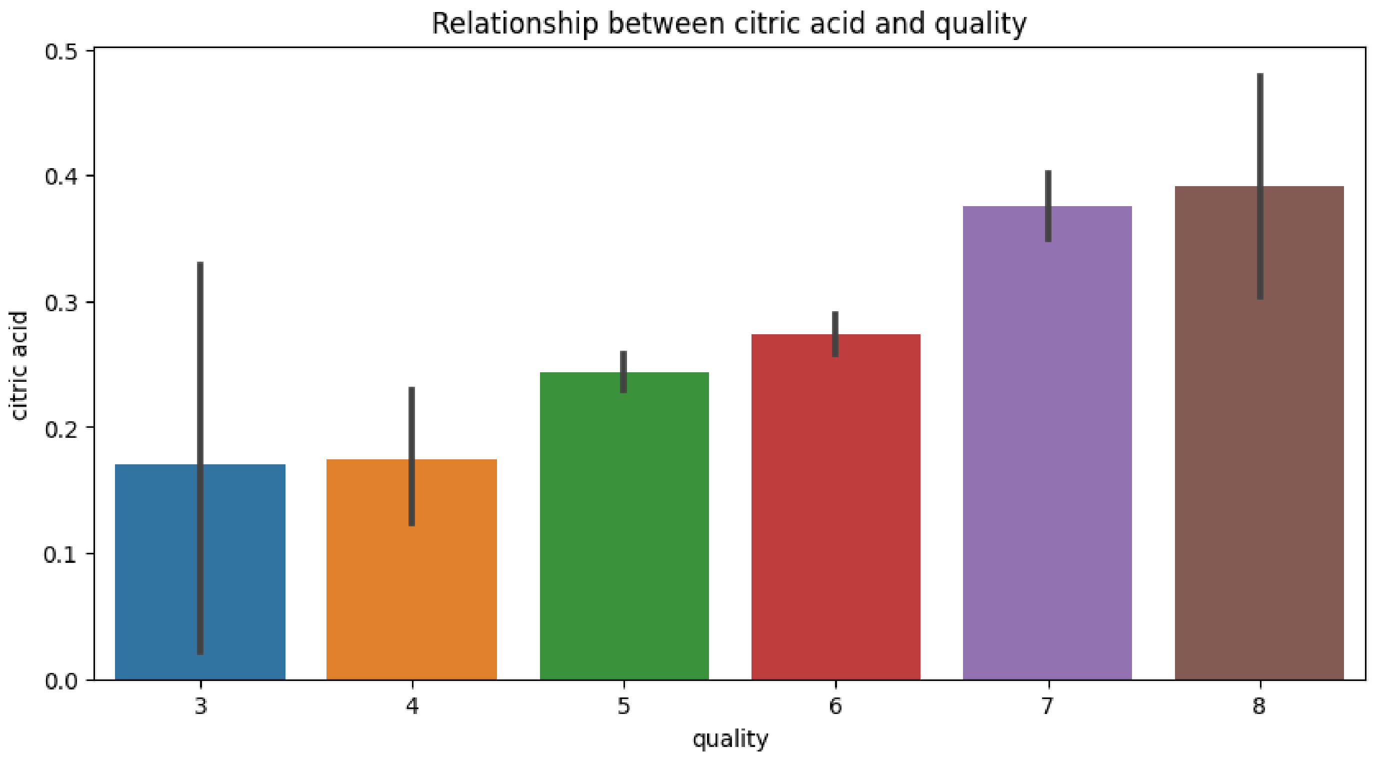
In [10]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='quality',y='volatile acidity',data=df)
plt.title('Relationship between volatile acidity and quality')
```

Out[10]:
```
Text(0.5, 1.0, 'Relationship between volatile acidity and quality')
```



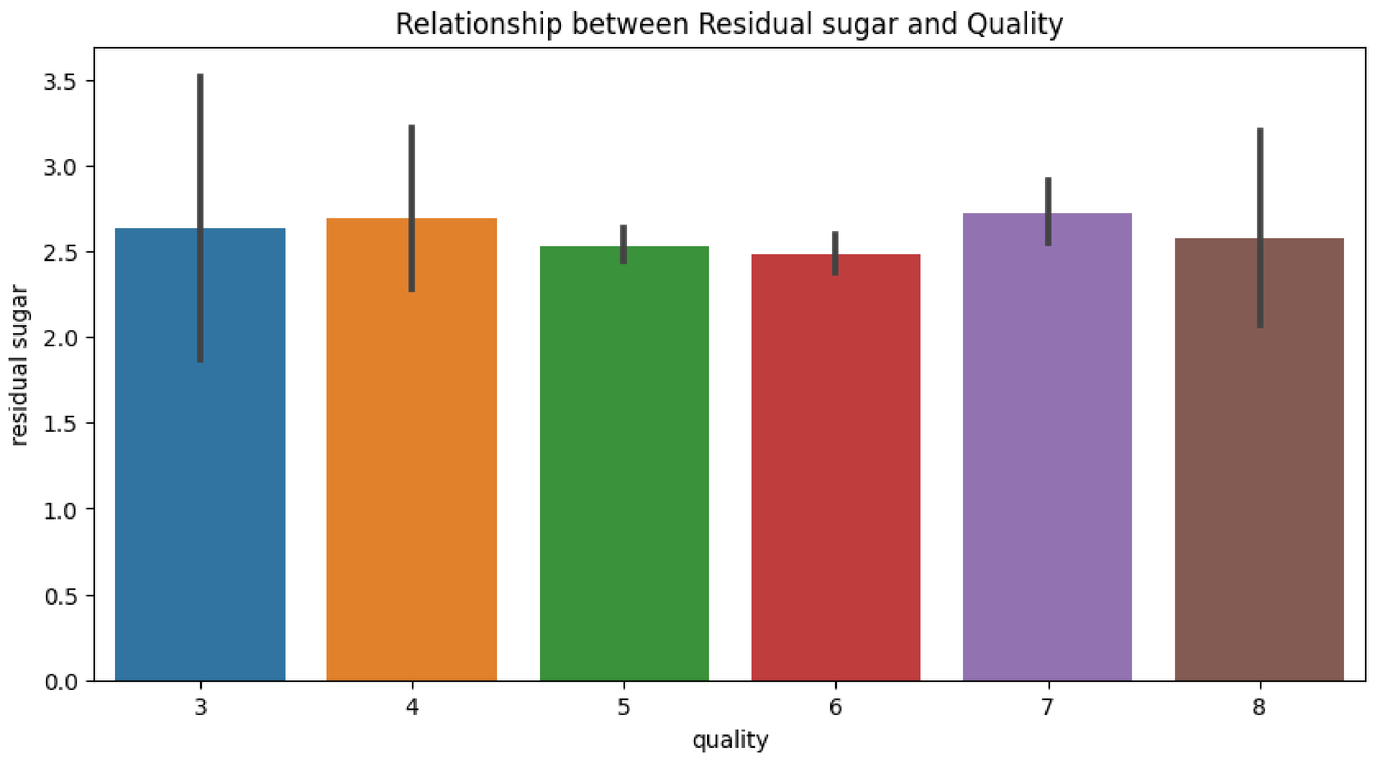
In [11]:
```python
plt.figure(figsize=(10,5))
sns.barplot(x='quality',y='citric acid',data=df)
plt.title('Relationship between citric acid and quality')
```

Out[11]:
```
Text(0.5, 1.0, 'Relationship between citric acid and quality')
```



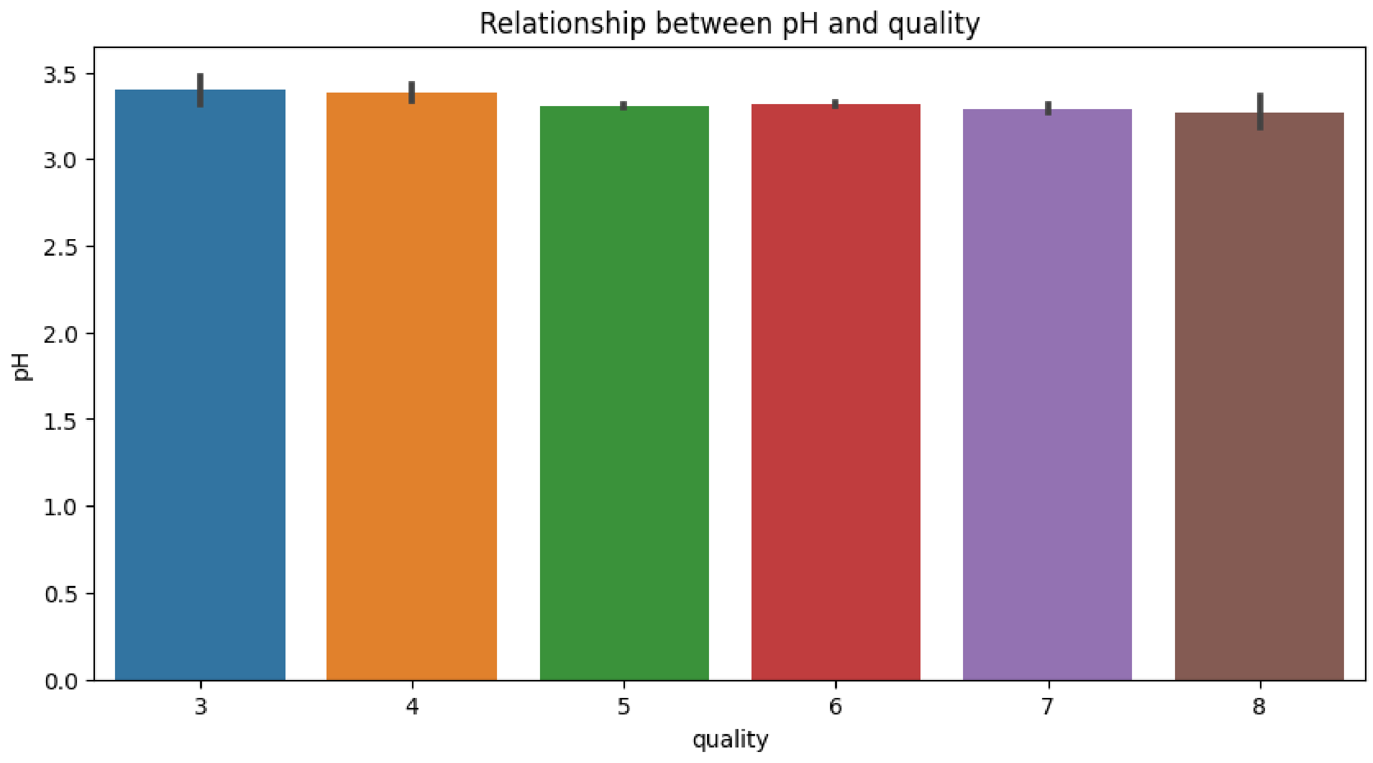
In [18]:
```python
plt.figure(figsize=(10,5))
sns.barplot(x='quality',y='residual sugar',data=df)
plt.title('Relationship between Residual sugar and Quality')
```

Out[18]:
```
Text(0.5, 1.0, 'Relationship between Residual sugar and Quality')
```



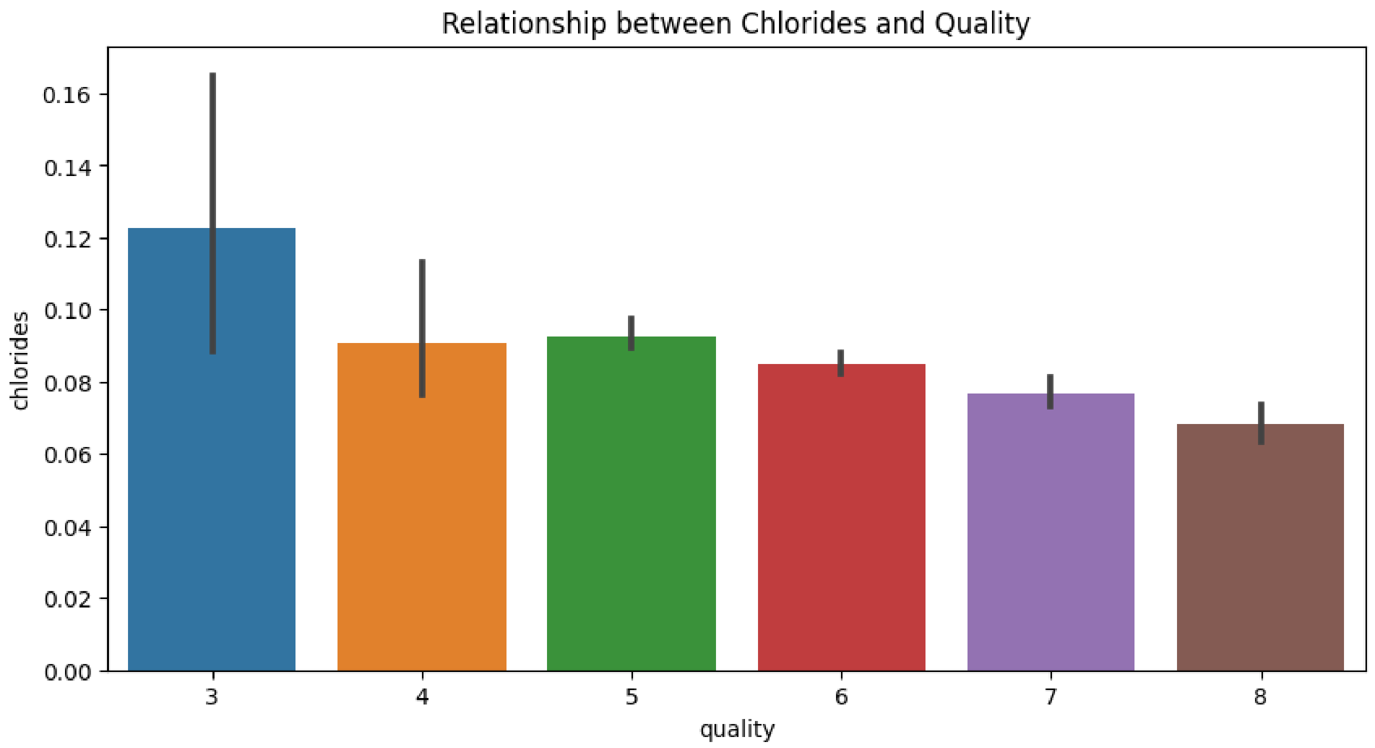
```
In [12]: plt.figure(figsize=(10,5))
         sns.barplot(x='quality',y='pH',data=df)
         plt.title('Relationship between pH and quality')
```

```
Out[12]: Text(0.5, 1.0, 'Relationship between pH and quality')
```

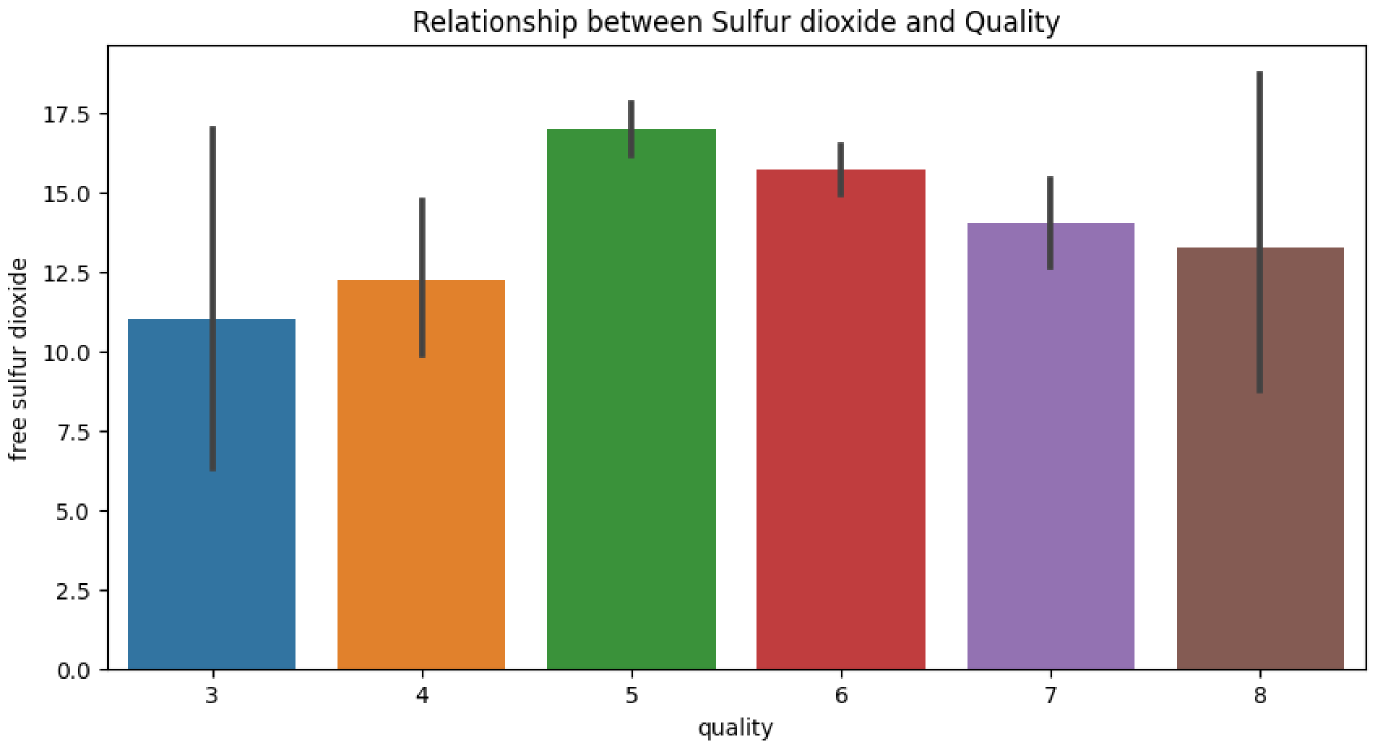


```
In [19]: plt.figure(figsize=(10,5))
         sns.barplot(x='quality',y='chlorides',data=df)
         plt.title('Relationship between Chlorides and Quality')
```

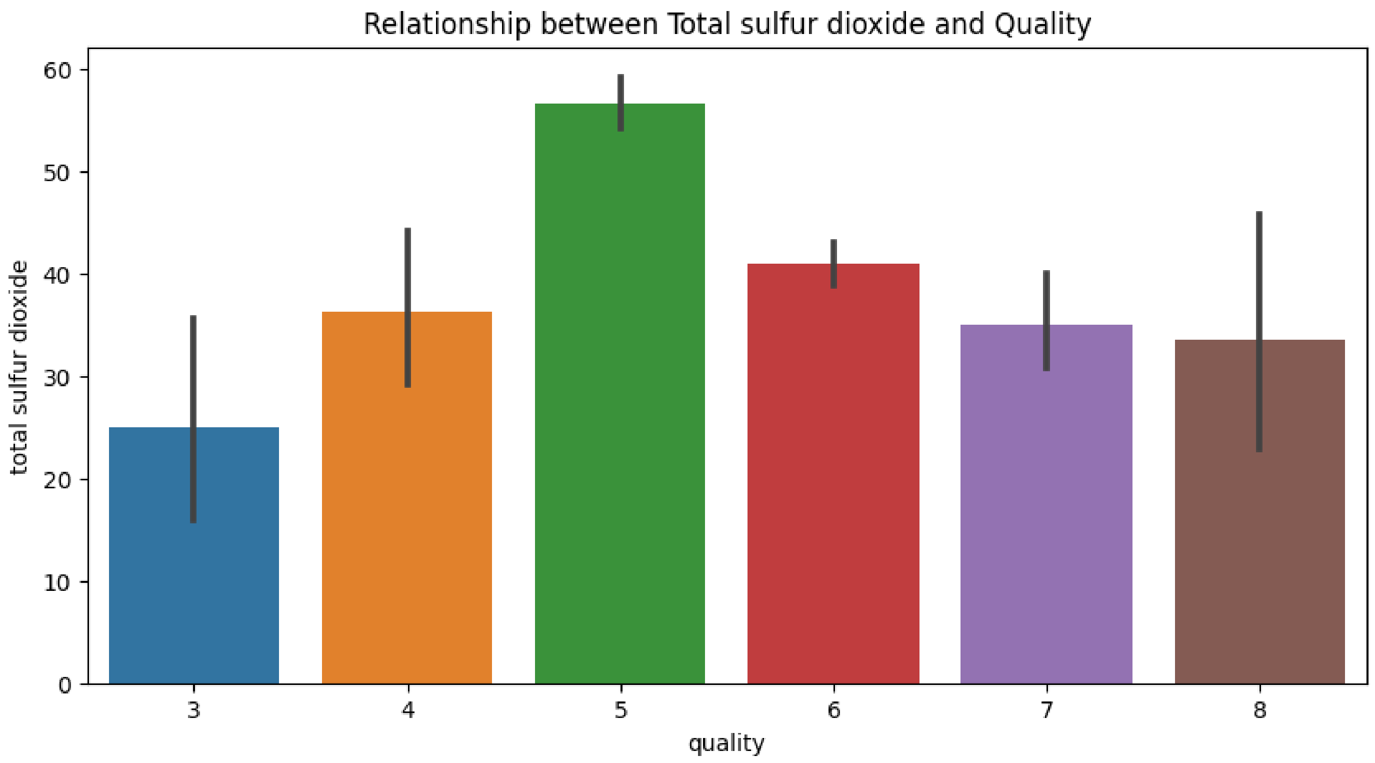
```
Out[19]: Text(0.5, 1.0, 'Relationship between Chlorides and Quality')
```



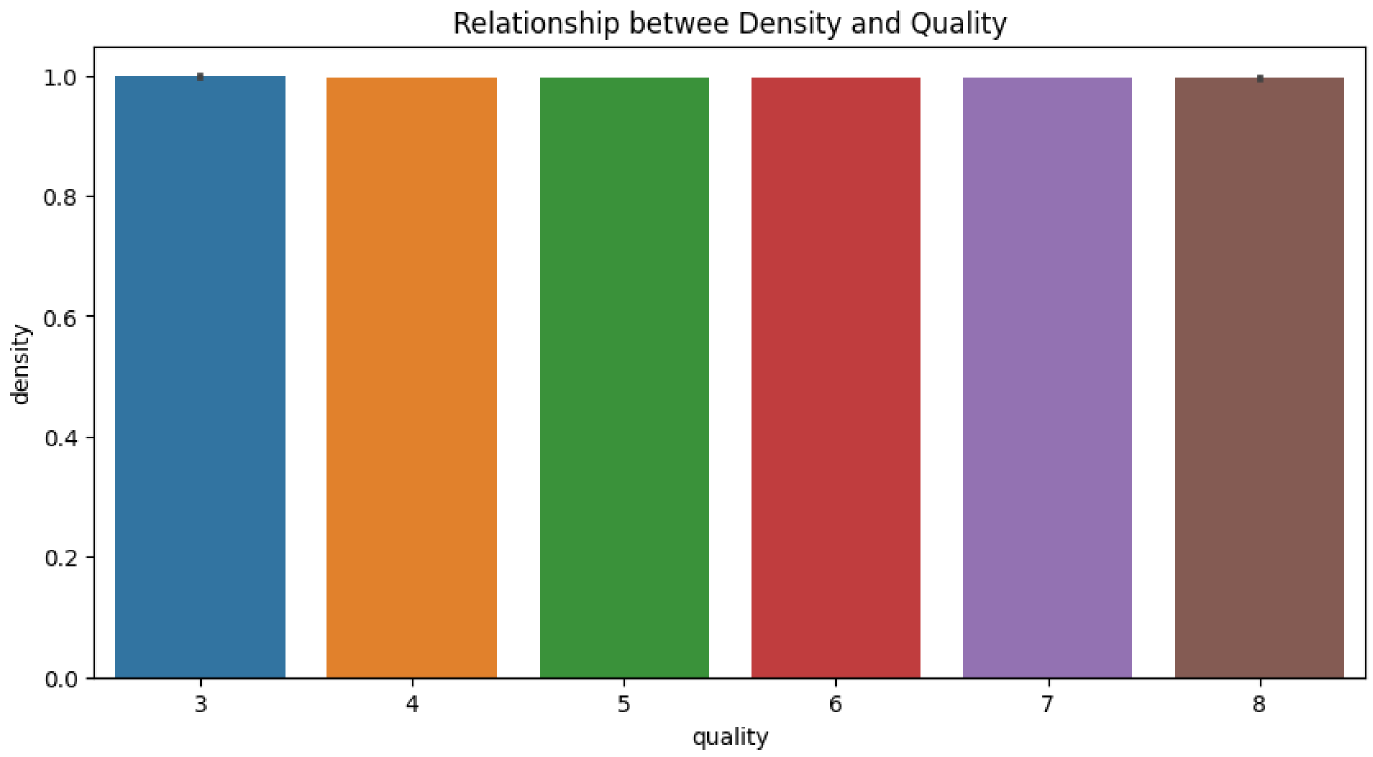
```
In [20]: plt.figure(figsize=(10,5))
         sns.barplot(x='quality',y='free sulfur dioxide',data=df)
         plt.title('Relationship between Sulfur dioxide and Quality')

Out[20]: Text(0.5, 1.0, 'Relationship between Sulfur dioxide and Quality')
```



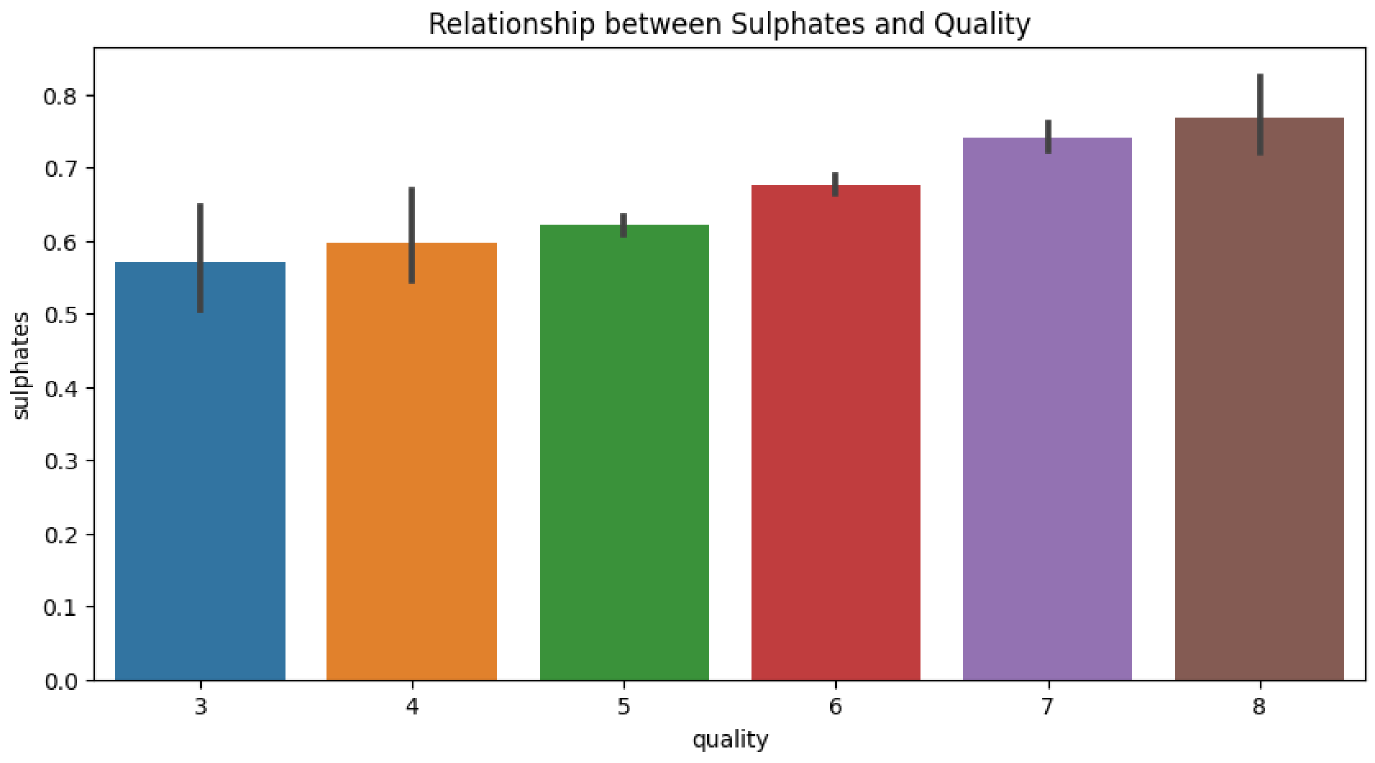
```
In [21]: plt.figure(figsize=(10,5))
         sns.barplot(x='quality',y='total sulfur dioxide',data=df)
         plt.title('Relationship between Total sulfur dioxide and Quality')

Out[21]: Text(0.5, 1.0, 'Relationship between Total sulfur dioxide and Quality')
```



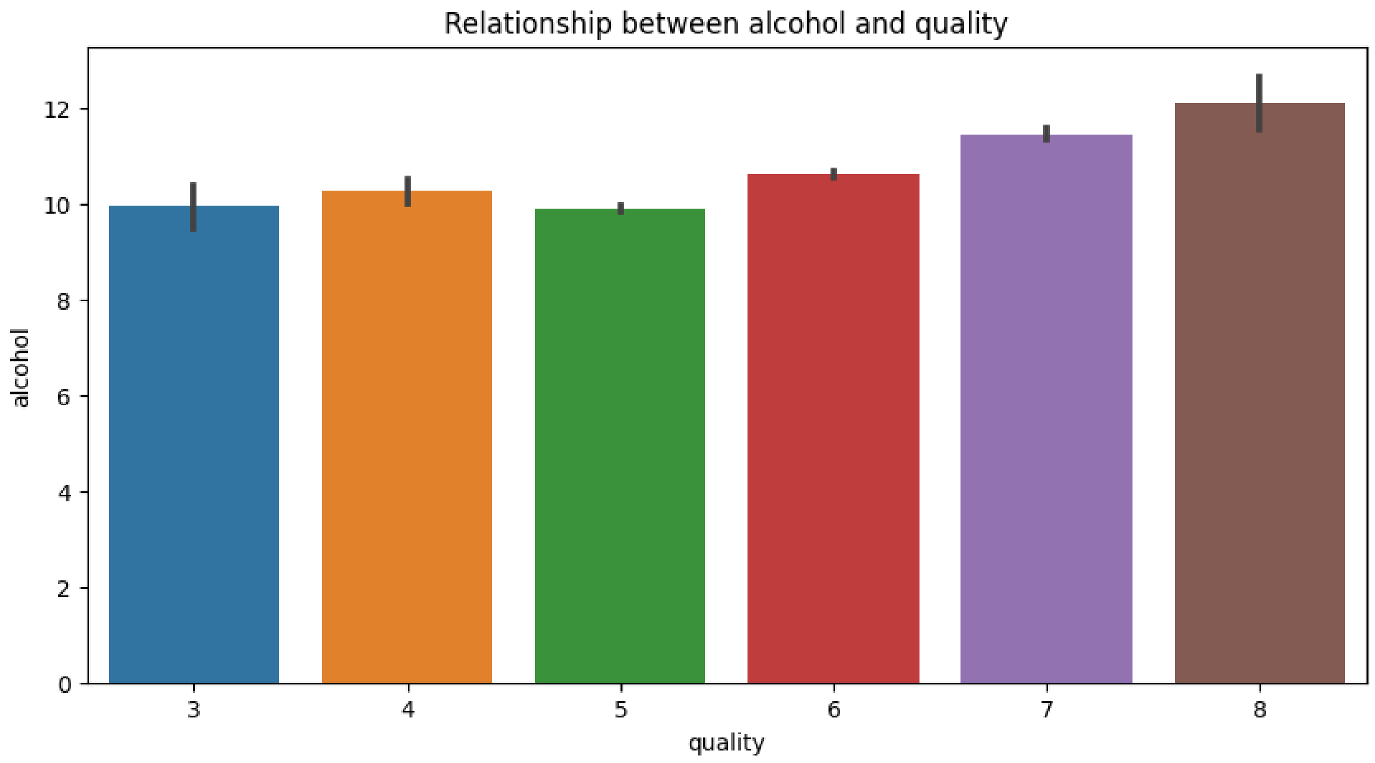
```
In [22]: plt.figure(figsize=(10,5))
         sns.barplot(x='quality',y='density',data=df)
         plt.title('Relationship betwee Density and Quality')
Out[22]: Text(0.5, 1.0, 'Relationship betwee Density and Quality')
```



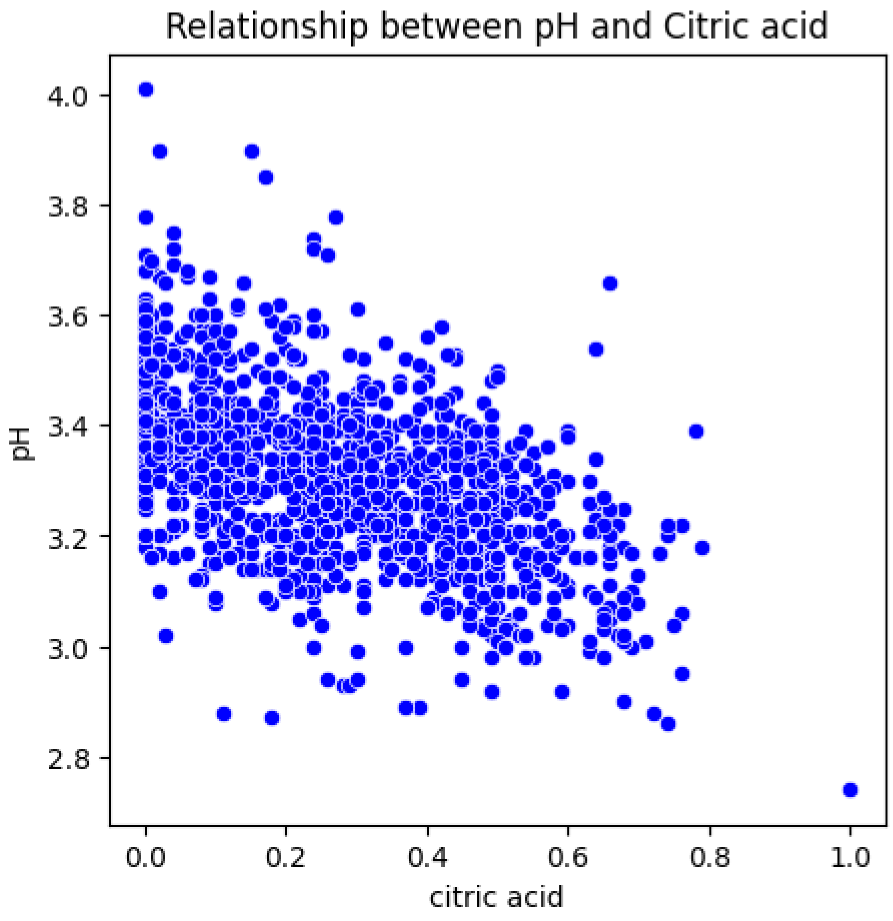
```
In [23]: plt.figure(figsize=(10,5))
         sns.barplot(x='quality',y='sulphates',data=df)
         plt.title('Relationship between Sulphates and Quality')
Out[23]: Text(0.5, 1.0, 'Relationship between Sulphates and Quality')
```



```
In [13]: plt.figure(figsize=(10,5))
         sns.barplot(x='quality',y='alcohol',data=df)
         plt.title('Relationship between alcohol and quality')

Out[13]: Text(0.5, 1.0, 'Relationship between alcohol and quality')
```

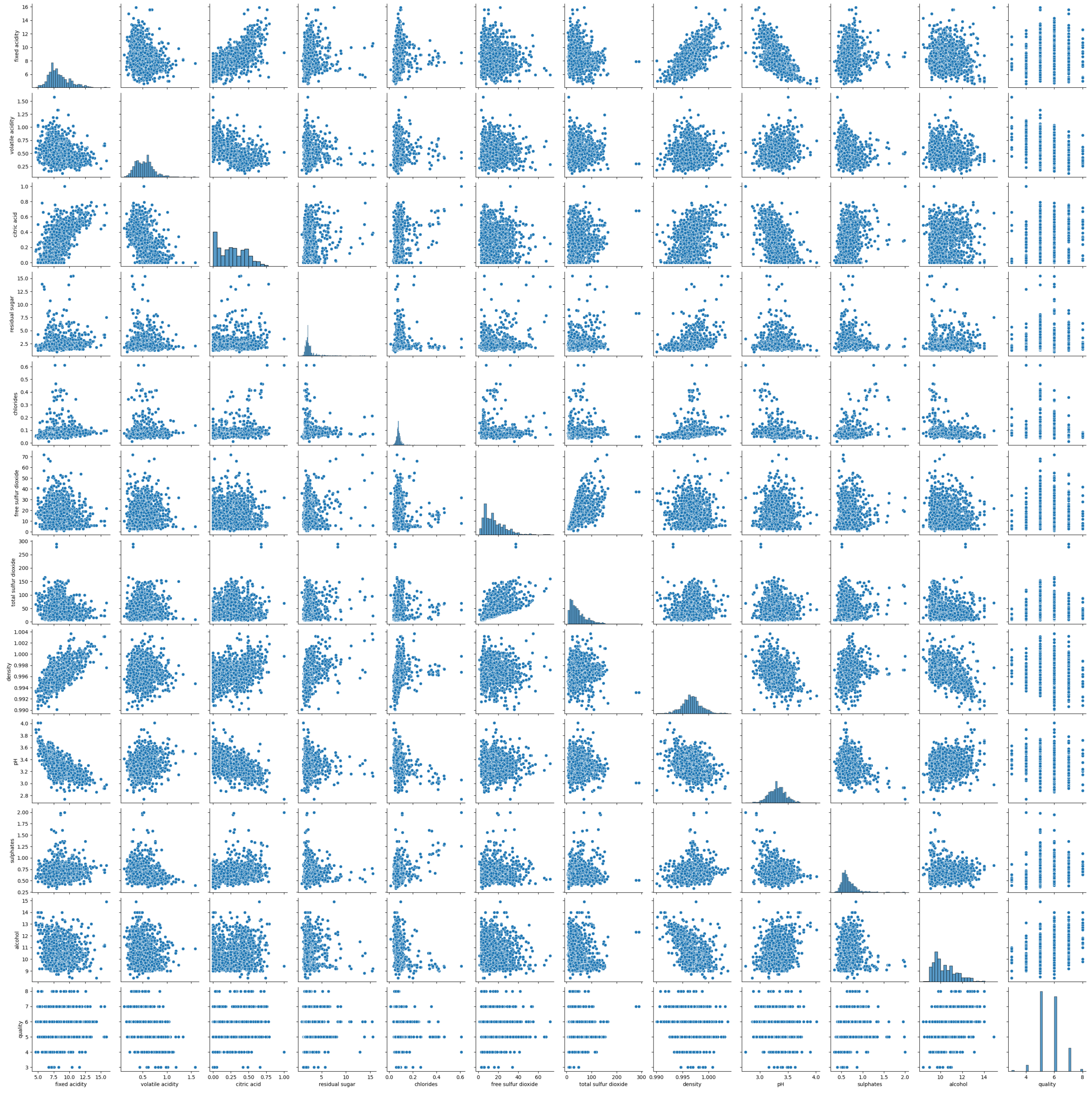


```
In [14]: plt.figure(figsize=(5,5))
         sns.scatterplot(x='citric acid',y='pH',data=df,color='b')
         plt.title("Relationship between pH and Citric acid")

Out[14]: Text(0.5, 1.0, 'Relationship between pH and Citric acid')
```



In [15]:
```
plt.figure(figsize=(10,10))
sns.pairplot(df)
```

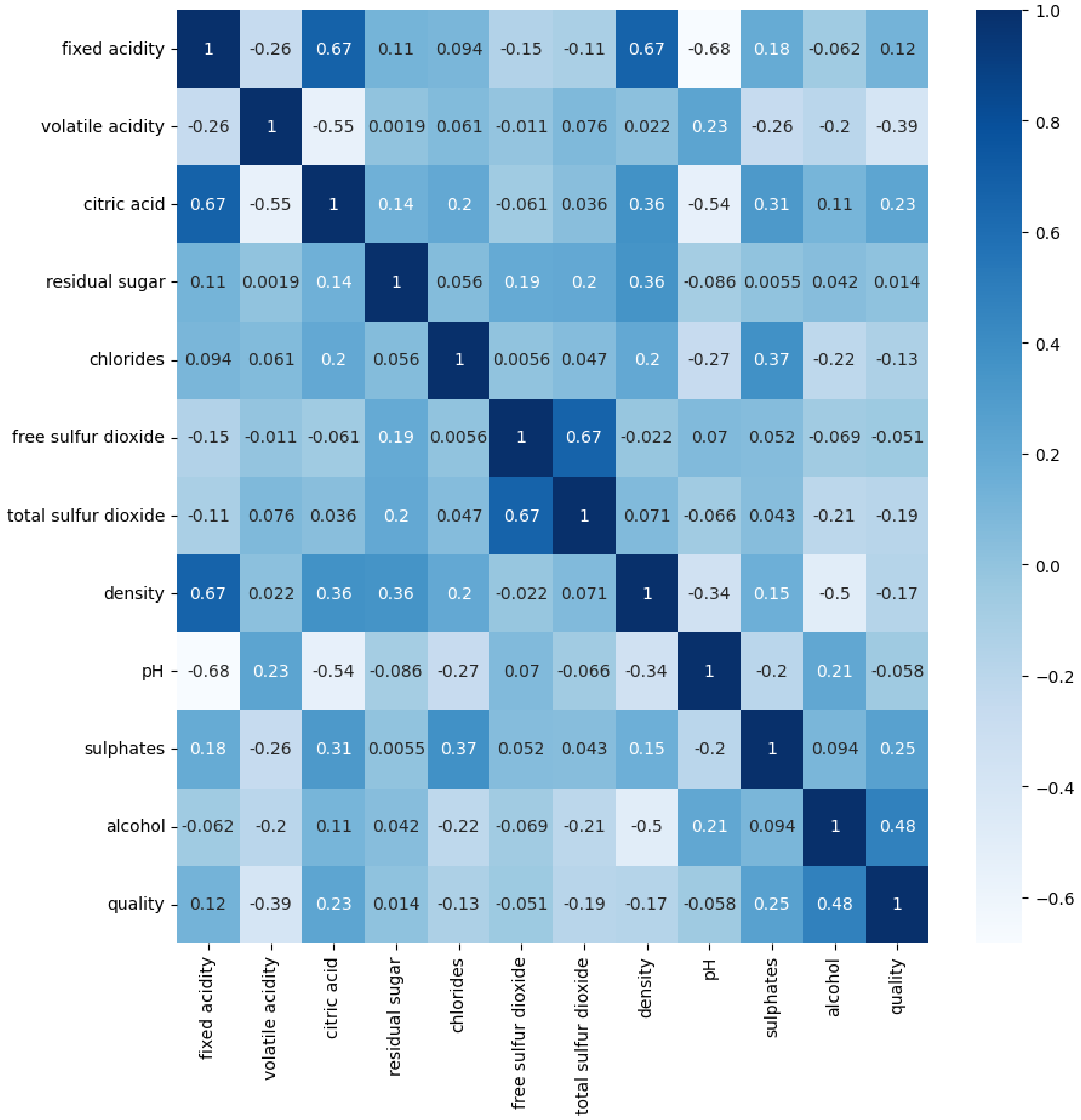
Out[15]:
```
<seaborn.axisgrid.PairGrid at 0x7f03aa19b190>

<Figure size 1000x1000 with 0 Axes>
```



In [ ]:

In [16]:
```python
#correlation between all the columns
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True,cmap='Blues')
```

Out[16]:
```
<AxesSubplot:>
```



In [24]:
```
x=df.drop(['quality'],axis=1)
x
```

Out[24]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2

1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0

1599 rows × 11 columns

In [25]:
```
#label binarization
y=df['quality'].apply(lambda x:1 if x>=7 else 0)
y
```

Out[25]:
```
0       0
1       0
2       0
3       0
4       0
       ..
1594    0
1595    0
1596    0
1597    0
1598    0
Name: quality, Length: 1599, dtype: int64
```

In [26]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
x_test
```

Out[26]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
75	8.8	0.41	0.64	2.2	0.093	9.0	42.0	0.99860	3.54	0.66	10.5
1283	8.7	0.63	0.28	2.7	0.096	17.0	69.0	0.99734	3.26	0.63	10.2
408	10.4	0.34	0.58	3.7	0.174	6.0	16.0	0.99700	3.19	0.70	11.3
1281	7.1	0.46	0.20	1.9	0.077	28.0	54.0	0.99560	3.37	0.64	10.4
1118	7.1	0.39	0.12	2.1	0.065	14.0	24.0	0.99252	3.30	0.53	13.3
...	...	...	...	...	...	...	...	...	...	...	...
1596	6.3	0.51	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0
985	7.4	0.58	0.00	2.0	0.064	7.0	11.0	0.99562	3.45	0.58	11.3
671	8.2	0.73	0.21	1.7	0.074	5.0	13.0	0.99680	3.20	0.52	9.5
1379	7.5	0.57	0.02	2.6	0.077	11.0	35.0	0.99557	3.36	0.62	10.8
1169	7.6	0.50	0.29	2.3	0.086	5.0	14.0	0.99502	3.32	0.62	11.5

480 rows × 11 columns

In [27]:
```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
y_pred
```

Out[27]:
```
array([0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [28]:
```
model.predict([[11.2,0.280,0.56 ,1.9 ,0.075 ,17.0,60.0,0.99800,3.16 ,0.58 ,9.8]])
```

```
/home/aparna/.local/lib/python3.8/site-packages/sklearn/base.py:450: UserWarning: X does
not have valid feature names, but KNeighborsClassifier was fitted with feature names
  warnings.warn(
```

Out[28]:
```
array([0])
```

In [29]:
```
from sklearn.metrics import accuracy_score,classification_report,ConfusionMatrixDisplay
print(accuracy_score(y_test,y_pred))
```

```
0.8854166666666666
```

In [30]:
```
print(classification_report(y_test,y_pred))
```

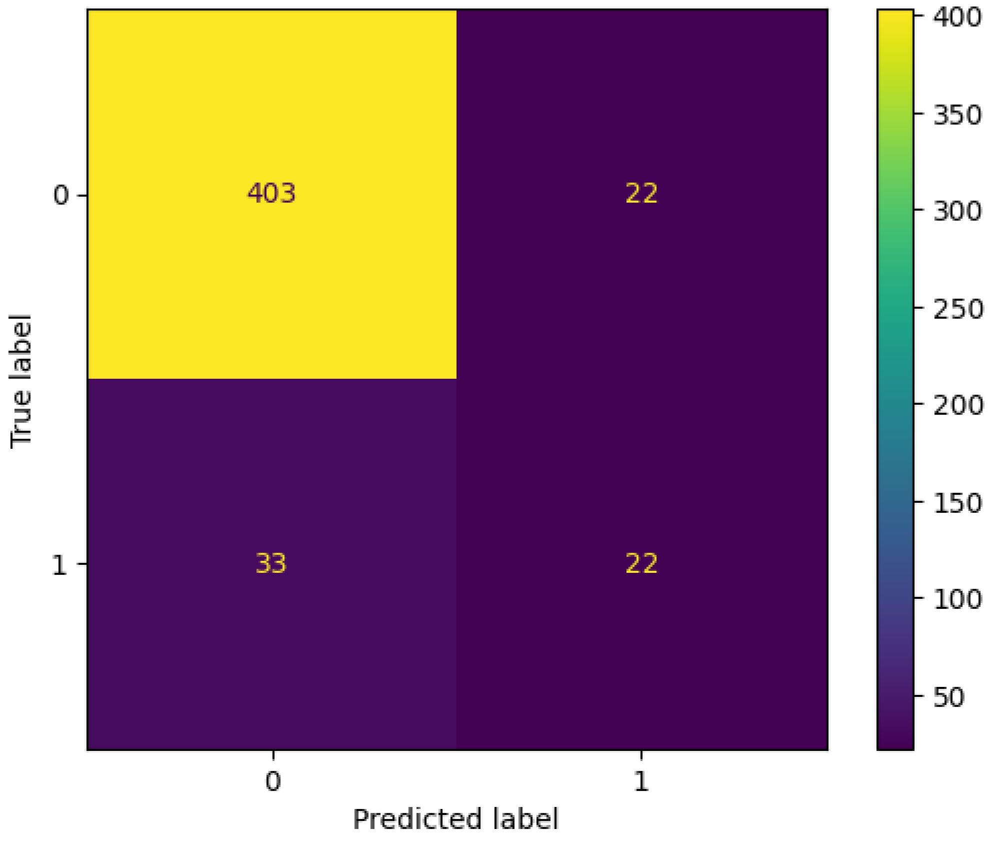
```
              precision    recall  f1-score   support

           0       0.92      0.95      0.94       425
           1       0.50      0.40      0.44        55

    accuracy                           0.89       480
   macro avg       0.71      0.67      0.69       480
weighted avg       0.88      0.89      0.88       480
```

In [31]:
```
from sklearn.metrics import mean_absolute_error
print('MAE',mean_absolute_error(y_test,y_pred))
```

```
MAE 0.11458333333333333
```

In [32]:
```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
```

Out[32]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f039bce2910>
```


403
22
33
22
0
1
0
1
True label
Predicted label
400
350
300
250
200
150
100
50

```
In [ ]:
```