PROJECT TITLE: WATER QUALITY ANALYSIS

PHASE 4: CREATING VISUALIZATIONS AND BUILDING A PREDICTIVE MODEL

In this model we are going to create visualizations of the previous loaded dataset and to building a predictive model.

STEPS FOLLOWED:

STEP 1: LOAD THE DATASET

Loading the given dataset from the source shared, by using the library functions.

STEP 2: HANDLING THE MISSING VALUES

After loading the dataset we have to identify the missing values by using the functions like isnull()

Drop the missing values based upon the nature of the dataset.

STEP 3: CREATING VISUALIZATIONS

After preprocessing the data, we have to create the visualizations of the given dataset.

To create the visualizations, use the library functions such as the matplotlib, seaborn. These libraries can be used to create histograms, scatter plots

STEP 4: BUILDING A PREDICTIVE MODEL

The machine learning models such as the logistic regression and the random forest to determine the water portability based upon the water quality parameters.

VISUALIZATION OF DATA

Importing libraries:

Importing necessary libraries for loading the dataset.

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

Load the dataset:

```
In [2]: data=pd.read_csv("/kaggle/input/water-potability/water_potability.csv")
```

DATA PREPROCESSING:

Perform data cleaning and preprocessing. This may include handling missing values, converting data types, and ensuring data quality.

Here already the dataset is cleaned and loaded, so no preprocessing is needed.

In [3]:
```
data.head()
```

Out[3]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637	66.420093	3.055934	0
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100.341674	4.628771	0
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0

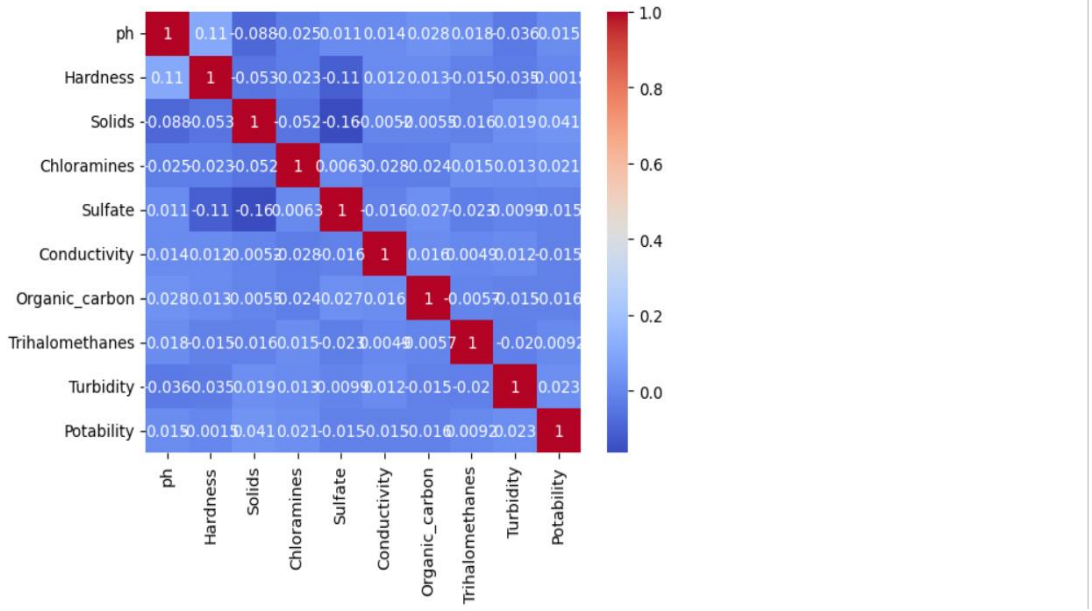
In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ph               2785 non-null   float64
 1   Hardness         3276 non-null   float64
 2   Solids           3276 non-null   float64
 3   Chloramines      3276 non-null   float64
 4   Sulfate          2495 non-null   float64
 5   Conductivity     3276 non-null   float64
 6   Organic_carbon   3276 non-null   float64
 7   Trihalomethanes  3114 non-null   float64
 8   Turbidity        3276 non-null   float64
 9   Potability       3276 non-null   int64  
dtypes: float64(9), int64(1)
memory usage: 256.1 KB
```

In [7]:
```
data.shape
```

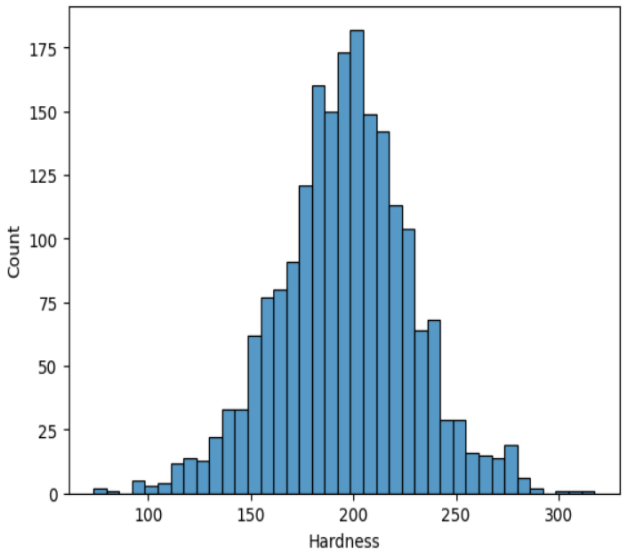
Out[7]: (3276, 10)

In [15]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(cor,annot=True,cmap='coolwarm')
plt.show()
```



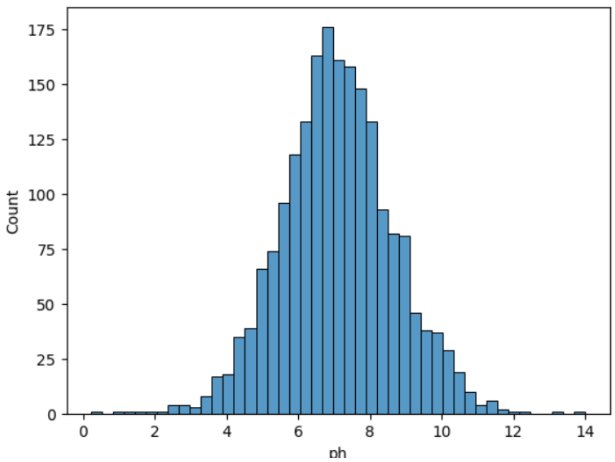
In [19]:
```
sns.histplot(data['Hardness'])
```

Out[19]: <Axes: xlabel='Hardness', ylabel='Count'>

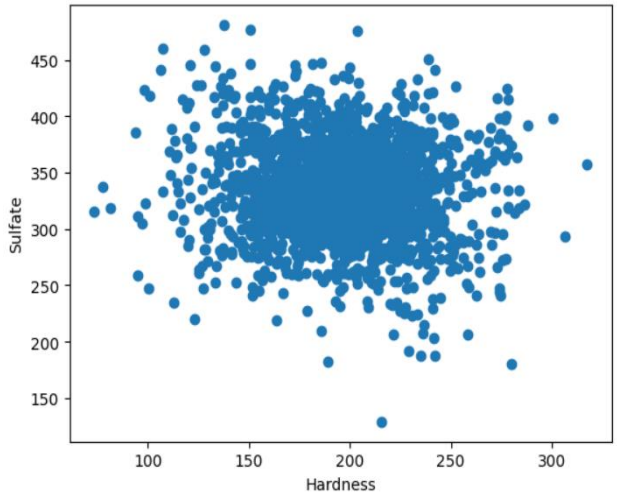


In [20]:
```
sns.histplot(data['ph'])
```

Out[20]: <Axes: xlabel='ph', ylabel='Count'>



In [21]:
```
gp = plt.scatter(data['Hardness'],data['Sulfate'])
plt.xlabel('Hardness')
plt.ylabel('Sulfate')
plt.show(gp)
```



DATA NORMALIZATION AND STANDARDIZATION

In [17]:
```
from sklearn.preprocessing import MinMaxScaler,StandardScaler

normalizer=MinMaxScaler()
standardizer=StandardScaler()
X= normalizer.fit_transform(X)
X=standardizer.fit_transform(X)
```

In [18]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=62)
```

MODEL BUILDING

In [19]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import ExtraTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score

models = {
    'Logistic Regression': LogisticRegression(),
    'Naive Bayes': GaussianNB(),
    'Support Vector Machine': SVC(),
    'K-Nearest Neighbors': KNeighborsClassifier(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'Bagging': BaggingClassifier(),
    'AdaBoost': AdaBoostClassifier(),
    'Gradient Boosting': GradientBoostingClassifier(),
    'Extra Trees': ExtraTreeClassifier(),
}
```

```python
for name, md in models.items():
    md.fit(X_train,Y_train)
    ypred = md.predict(X_test)

    print(f"{name}  with accuracy : {accuracy_score(Y_test,ypred)}")
```

```
Logistic Regression  with accuracy : 0.6178660049627791
Naive Bayes  with accuracy : 0.6327543424317618
Support Vector Machine  with accuracy : 0.7245657568238213
K-Nearest Neighbors  with accuracy : 0.6550868486352357
Decision Tree  with accuracy : 0.6104218362282878
Random Forest  with accuracy : 0.7096774193548387
Bagging  with accuracy : 0.6650124069478908
AdaBoost  with accuracy : 0.6004962779156328
Gradient Boosting  with accuracy : 0.6898263027295285
Extra Trees  with accuracy : 0.5806451612903226
```