In [1]:
```
import matplotlib.pyplot as plt # To visulaize the data
import pandas as pd # To read and analyse the data
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data=pd.read_csv('C:\\Users\\KAVYA SRI\\Downloads\\water analysis\water_potabi
data
```

Out[2]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbo
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.37978
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.18001
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.86863
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.43652
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.55827
...	...	...	...	...	...	...	.
3271	4.668102	193.681735	47580.991603	7.166639	359.948574	526.424171	13.89441
3272	7.808856	193.553212	17329.802160	8.061362	NaN	392.449580	19.90322
3273	9.419510	175.762646	33155.578218	7.350233	NaN	432.044783	11.03907
3274	5.126763	230.603758	11983.869376	6.303357	NaN	402.883113	11.16894
3275	7.874671	195.102299	17404.177061	7.509306	NaN	327.459760	16.14036

3276 rows × 10 columns

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

Out[4]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279

In [5]:
```
data.tail()
```

Out[5]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbo
3271	4.668102	193.681735	47580.991603	7.166639	359.948574	526.424171	13.89441
3272	7.808856	193.553212	17329.802160	8.061362	NaN	392.449580	19.90322
3273	9.419510	175.762646	33155.578218	7.350233	NaN	432.044783	11.03907
3274	5.126763	230.603758	11983.869376	6.303357	NaN	402.883113	11.16894
3275	7.874671	195.102299	17404.177061	7.509306	NaN	327.459760	16.14036

In [6]:
```
data.shape # Rows and columns
```

Out[6]:
```
(3276, 10)
```

In [7]:
```
data.describe()
```

Out[7]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_
count	2785.000000	3276.000000	3276.000000	3276.000000	2495.000000	3276.000000	3276
mean	7.080795	196.369496	22014.092526	7.122277	333.775777	426.205111	14
std	1.594320	32.879761	8768.570828	1.583085	41.416840	80.824064	3
min	0.000000	47.432000	320.942611	0.352000	129.000000	181.483754	2
25%	6.093092	176.850538	15666.690297	6.127421	307.699498	365.734414	12
50%	7.036752	196.967627	20927.833607	7.130299	333.073546	421.884968	14
75%	8.062066	216.667456	27332.762127	8.114887	359.950170	481.792304	16
max	14.000000	323.124000	61227.196008	13.127000	481.030642	753.342620	28

In [8]:
```
data.describe().T
```

Out[8]:

	count	mean	std	min	25%	50%	
ph	2785.0	7.080795	1.594320	0.000000	6.093092	7.036752	
Hardness	3276.0	196.369496	32.879761	47.432000	176.850538	196.967627	2
Solids	3276.0	22014.092526	8768.570828	320.942611	15666.690297	20927.833607	27
Chloramines	3276.0	7.122277	1.583085	0.352000	6.127421	7.130299	
Sulfate	2495.0	333.775777	41.416840	129.000000	307.699498	333.073546	3
Conductivity	3276.0	426.205111	80.824064	181.483754	365.734414	421.884968	4
Organic_carbon	3276.0	14.284970	3.308162	2.200000	12.065801	14.218338	
Trihalomethanes	3114.0	66.396293	16.175008	0.738000	55.844536	66.622485	
Turbidity	3276.0	3.966786	0.780382	1.450000	3.439711	3.955028	
Potability	3276.0	0.390110	0.487849	0.000000	0.000000	0.000000	

- Data Cleaning

In [9]:
```
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 [10]:
```
data.isnull().sum()
```

Out[10]:
```
ph                 491
Hardness             0
Solids               0
Chloramines          0
Sulfate            781
Conductivity         0
Organic_carbon       0
Trihalomethanes    162
Turbidity            0
Potability           0
dtype: int64
```

In [11]:
```
data.fillna(data.mean(),inplace=True)
```

In [12]:
```
data.isnull().sum()
```

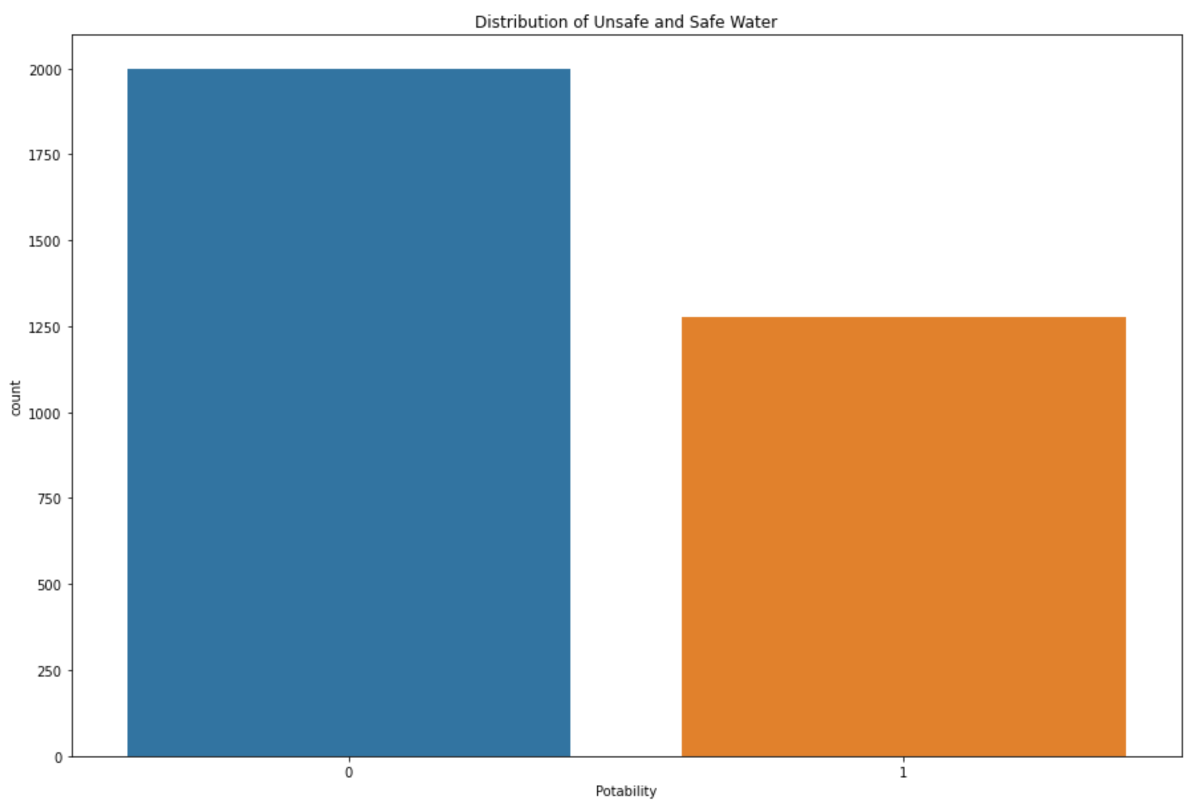
Out[12]:
```
ph                 0
Hardness           0
Solids             0
Chloramines        0
Sulfate            0
Conductivity       0
Organic_carbon     0
Trihalomethanes    0
Turbidity          0
Potability         0
dtype: int64
```

In [13]:
```
# Display information about the Data Frame
data.info(memory_usage="deep")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ph               3276 non-null   float64
 1   Hardness         3276 non-null   float64
 2   Solids           3276 non-null   float64
 3   Chloramines      3276 non-null   float64
 4   Sulfate          3276 non-null   float64
 5   Conductivity     3276 non-null   float64
 6   Organic_carbon   3276 non-null   float64
 7   Trihalomethanes  3276 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 [14]:
```
plt.figure(figsize=(15,10))
sns.countplot(data.Potability)
plt.title("Distribution of Unsafe and Safe Water")
plt.show
```

Out[14]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [15]:
```
import plotly.express as px
data = data
figure = px.histogram(data, x = "ph", color = "Potability", title = "Factors A
figure.show()
```

In [16]:
```
figure = px.histogram(data, x = "Hardness", color = "Potability", title = "Fac
figure.show()
```

In [17]:
```
figure = px.histogram(data, x = "Solids", color = "Potability", title = "Facto
figure.show()
```

In [18]:
```
figure = px.histogram(data, x = "Chloramines", color = "Potability", title = "
figure.show()
```

In [19]:
```
figure = px.histogram(data, x = "Sulfate", color = "Potability", title = "Fact
figure.show()
```

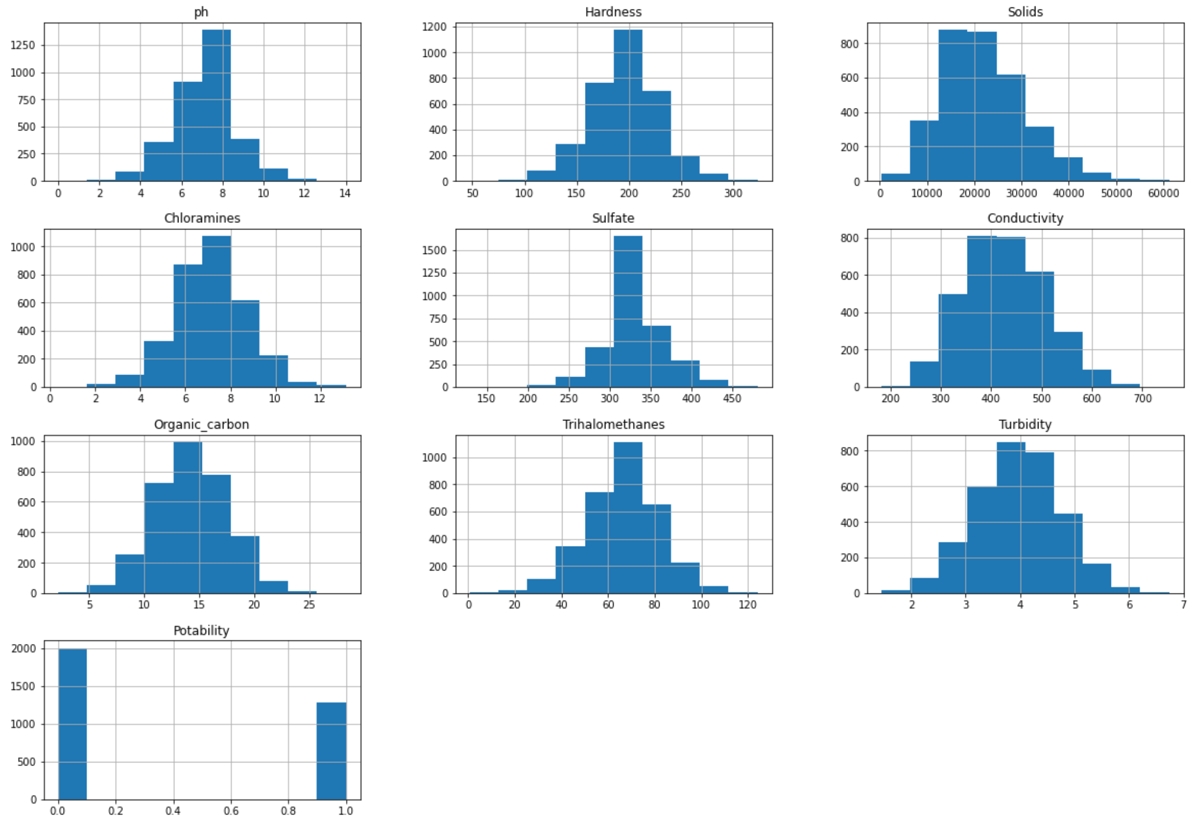
In [20]:
```
figure = px.histogram(data, x = "Sulfate", color = "Potability", title = "Fact
figure.show()
```

In [21]:
```
figure = px.histogram(data, x = "Organic_carbon", color = "Potability", title
figure.show()
```

In [22]:
```
figure = px.histogram(data, x = "Trihalomethanes", color = "Potability", title
figure.show()
```

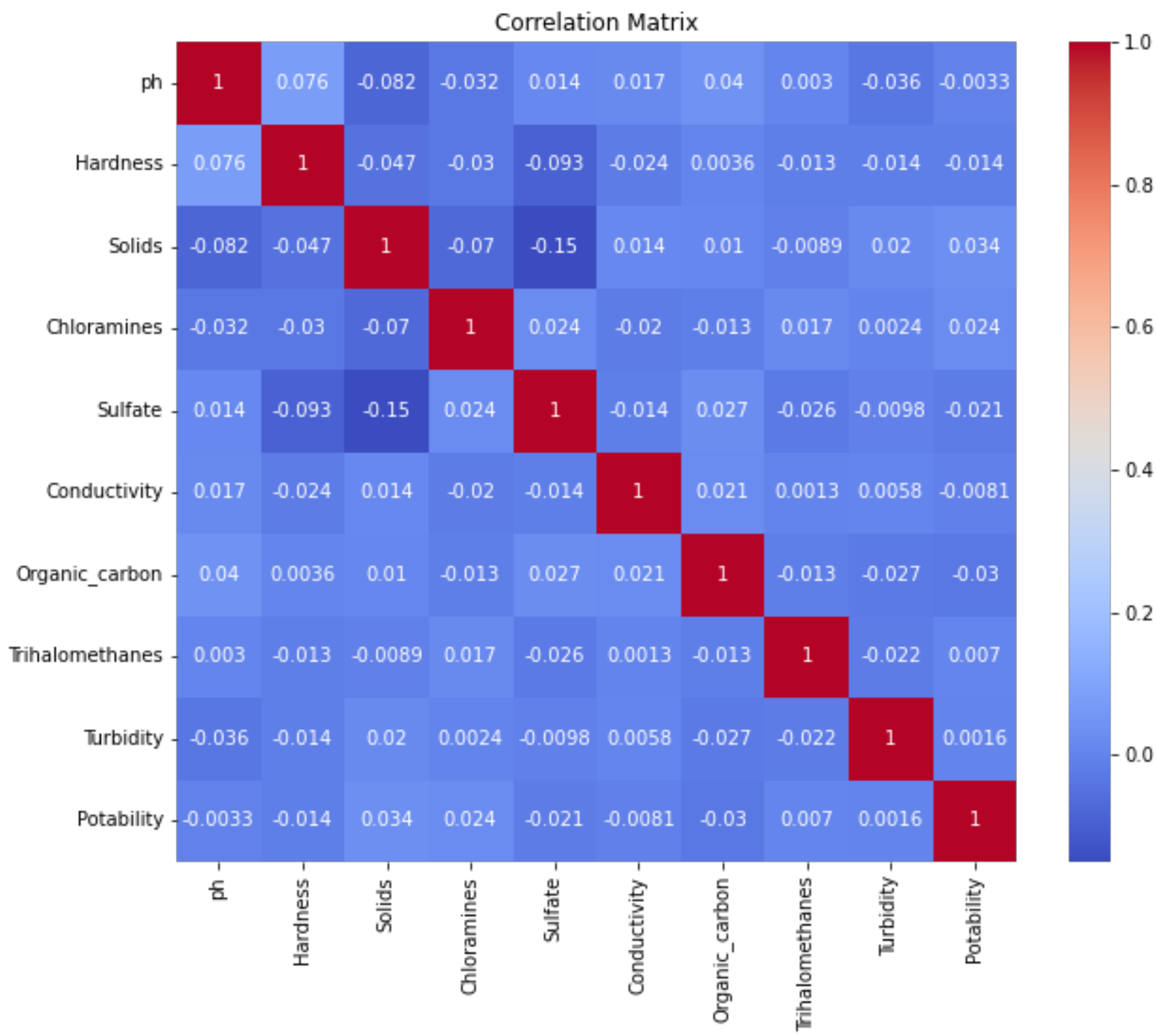
In [23]:
```
figure = px.histogram(data, x = "Turbidity", color = "Potability", title = "Fa
figure.show()
```

In [24]:
```
data.hist(figsize=(20,14))
plt.show()
```

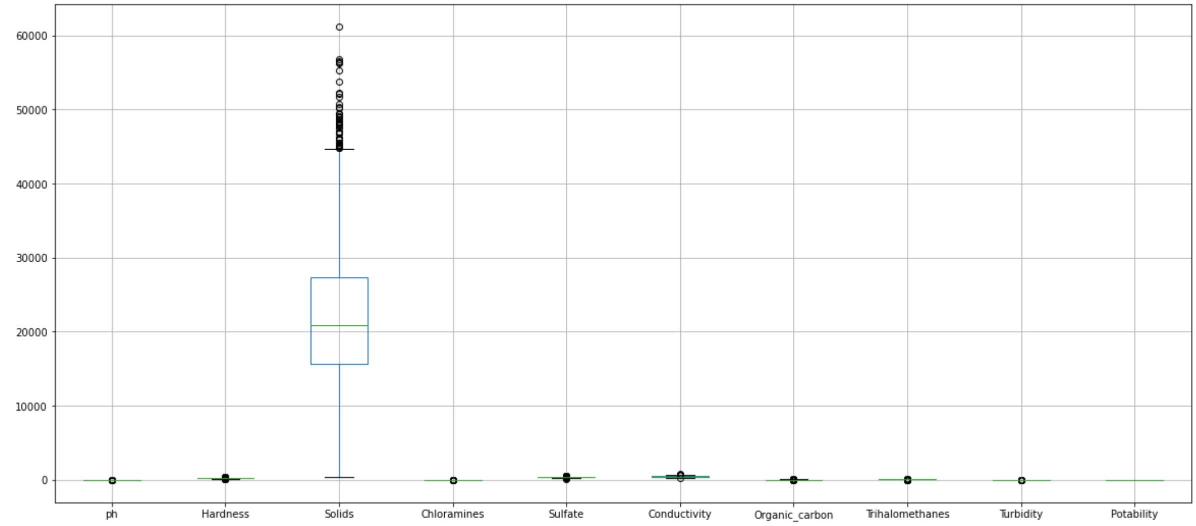


In [25]:
```
# Correlation matrix
correlation_matrix = data.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



In [26]:
```
data.boxplot(figsize=(20,9))
plt.show()
```



In [27]:
```
data['Solids'].describe()
```

Out[27]:
```
count     3276.000000
mean     22014.092526
std       8768.570828
min        320.942611
25%      15666.690297
50%      20927.833607
75%      27332.762127
max      61227.196008
Name: Solids, dtype: float64
```

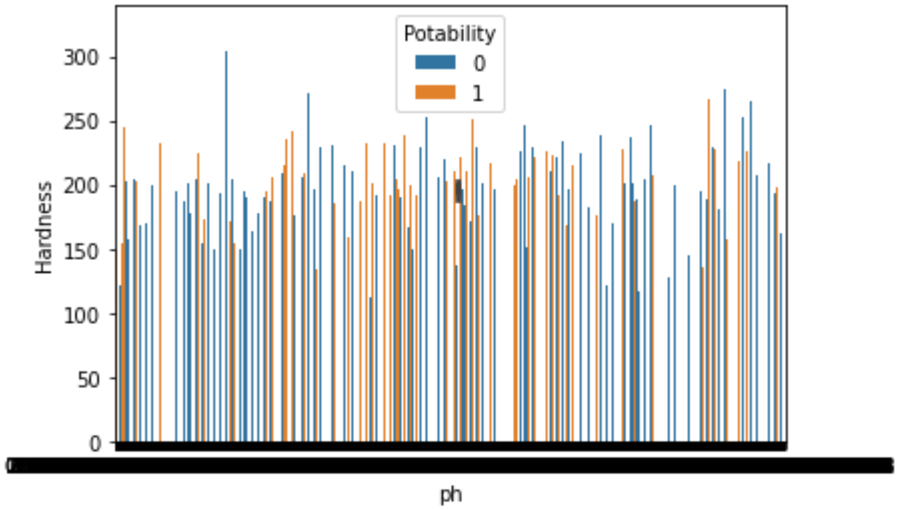
In [28]:
```
data['Solids']
```

Out[28]:
```
0       20791.318981
1       18630.057858
2       19909.541732
3       22018.417441
4       17978.986339
            ...
3271    47580.991603
3272    17329.802160
3273    33155.578218
3274    11983.869376
3275    17404.177061
Name: Solids, Length: 3276, dtype: float64
```

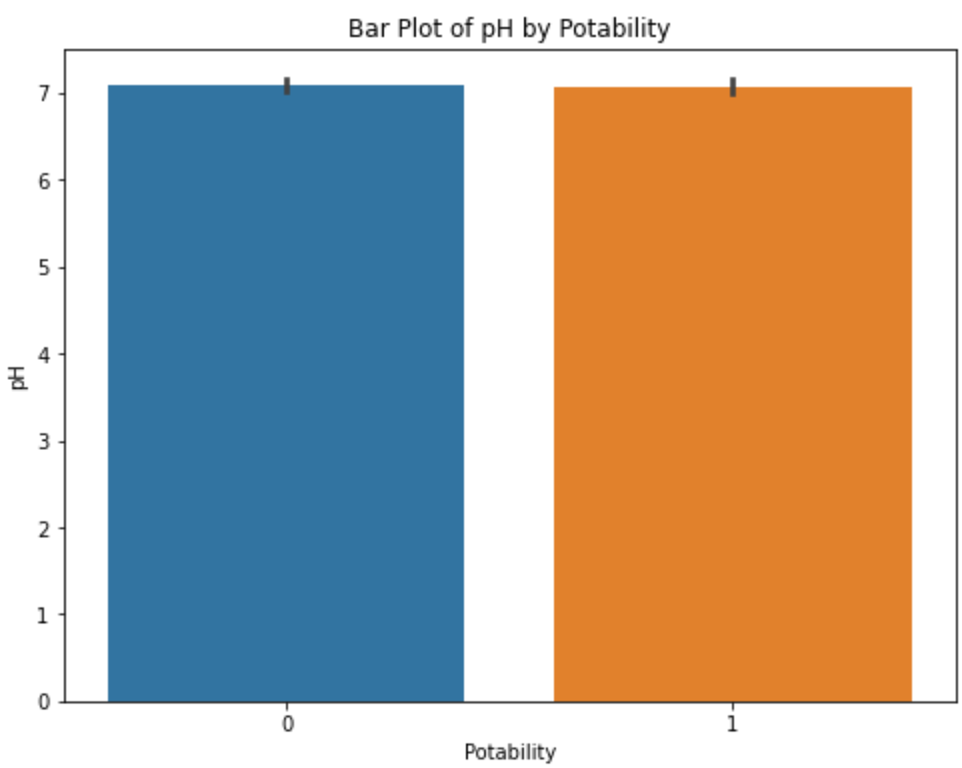
In [29]:
```
data['Potability'].value_counts()
```

Out[29]:
```
0    1998
1    1278
Name: Potability, dtype: int64
```

In [30]:
```
sns.barplot(x=data['ph'],y=data['Hardness'], hue=data['Potability'])
plt.show()
```



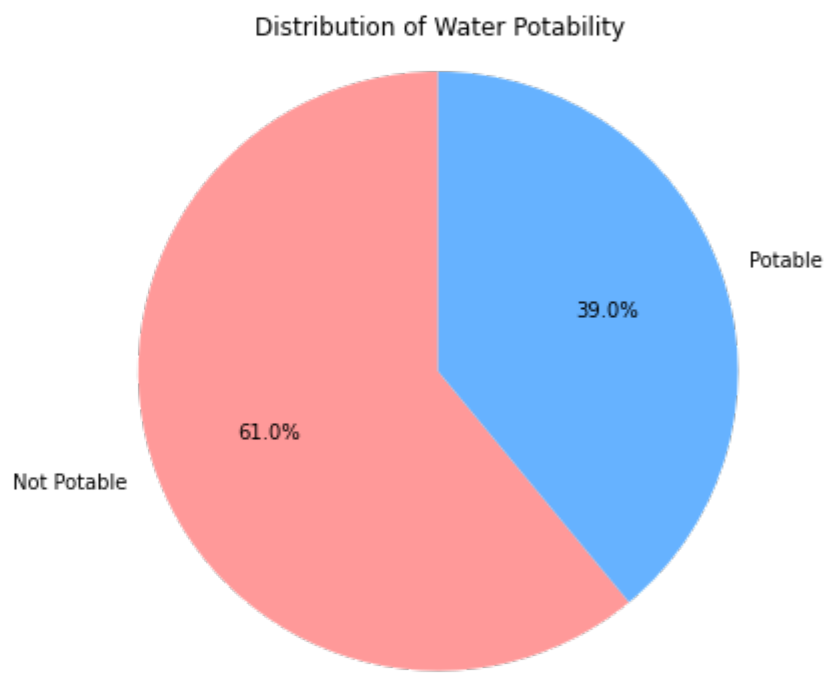
In [30]:
```
plt.figure(figsize=(8, 6))
sns.barplot(x='Potability', y='ph', data=data)
plt.title('Bar Plot of pH by Potability')
plt.xlabel('Potability')
plt.ylabel('pH')
plt.show()
```



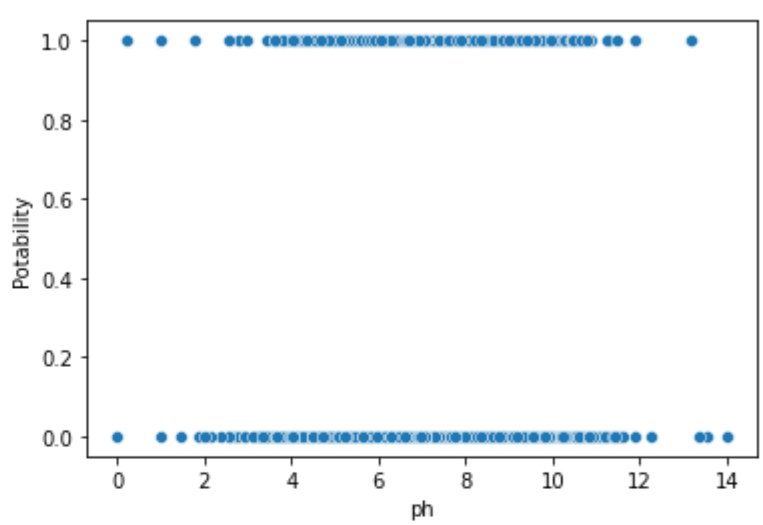
In [32]:
```
potability_counts = data['Potability'].value_counts()
print(potability_counts)
```

```
0    1998
1    1278
Name: Potability, dtype: int64
```

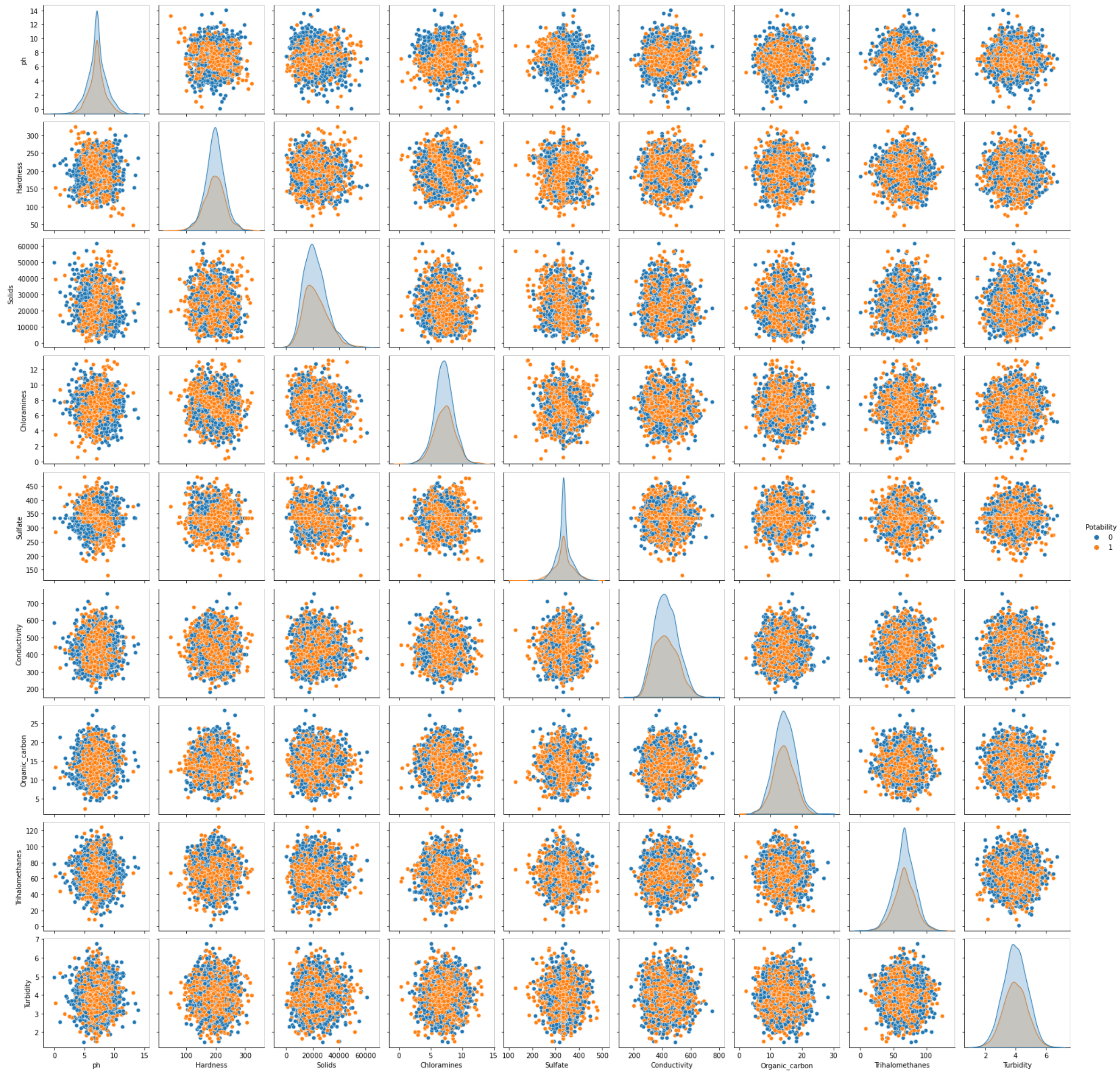
In [33]:
```
plt.figure(figsize=(6, 6))
plt.pie(potability_counts, labels=['Not Potable', 'Potable'], autopct='%1.1f%%
plt.title('Distribution of Water Potability')
plt.axis('equal')  # Equal aspect ratio ensures that the pie chart is circular
plt.show()
```



In [34]:
```
sns.scatterplot(x=data['ph'],y=data['Potability'])
plt.show()
```



In [35]:
```
# pair plot
sns.pairplot(data, hue='Potability')
plt.show()
```



In [36]:
```
from scipy.stats import ttest_ind

numerical_columns = ['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Co

for column in numerical_columns:
    potable_values = data[data['Potability'] == 1][column]
    non_potable_values = data[data['Potability'] == 0][column]

    t_stat, p_value = ttest_ind(potable_values, non_potable_values)

    print(f'{column}: t-statistic = {t_stat:.2f}, p-value = {p_value:.4f}')
```

```
ph: t-statistic = -0.19, p-value = 0.8508
Hardness: t-statistic = -0.79, p-value = 0.4285
Solids: t-statistic = 1.93, p-value = 0.0535
Chloramines: t-statistic = 1.36, p-value = 0.1736
Sulfate: t-statistic = -1.18, p-value = 0.2381
Conductivity: t-statistic = -0.47, p-value = 0.6419
Organic_carbon: t-statistic = -1.72, p-value = 0.0860
Trihalomethanes: t-statistic = 0.40, p-value = 0.6905
Turbidity: t-statistic = 0.09, p-value = 0.9279
```

In [37]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Separate features (X) and target (y)
X = data.drop('Potability', axis=1)
y = data['Potability']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando

# Standardize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [38]:
```
from sklearn.ensemble import RandomForestClassifier

# Initialize the classifier
classifier = RandomForestClassifier(random_state=42)

# Train the model
classifier.fit(X_train_scaled, y_train)
```

Out[38]:
```
RandomForestClassifier(random_state=42)
```

In [39]:
```
from sklearn.metrics import accuracy_score, classification_report, confusion_m

# Predict on the test set
y_pred = classifier.predict(X_test_scaled)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

conf_matrix = confusion_matrix(y_test, y_pred)
print('Confusion Matrix:')
print(conf_matrix)

class_report = classification_report(y_test, y_pred)
print('Classification Report:')
print(class_report)
```

```
Accuracy: 0.68
Confusion Matrix:
[[353  59]
 [152  92]]
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.86      0.77       412
           1       0.61      0.38      0.47       244

    accuracy                           0.68       656
   macro avg       0.65      0.62      0.62       656
weighted avg       0.67      0.68      0.66       656
```

In [40]:
```
# Separate features
data_X = data.drop('Potability', axis=1)

# Scale the features using the same scaler as before
data_X_scaled = scaler.transform(data_X)

# Make predictions for Kaggle samples
water_predictions = classifier.predict(data_X_scaled)

# Add the predictions as a new column in the dataset
data['Predicted_Potability'] = water_predictions

# Save the dataset with predictions
data.to_csv('Data_with_predictions.csv', index=False)
```

In [41]:
```python
# Filter and print suitable samples
suitable_samples = data[data['Predicted_Potability'] == 1]

print('Water samples predicted to be suitable for drinking:')
print(suitable_samples)
```

```
Water samples predicted to be suitable for drinking:
             ph    Hardness        Solids  Chloramines     Sulfate  \
26     3.445062  207.926260  33424.768678     8.782147  384.007006
30     7.181449  209.625601  15196.229987     5.994679  338.336431
32    10.433291  117.791230  22326.892046     8.161505  307.707509
44     4.758439  183.349454  21568.428779     4.731349  333.775777
67     7.080795  103.464759  27420.167425     8.417305  333.775777
...         ...         ...           ...          ...         ...
3269  11.491011   94.812545  37188.826022     9.263166  258.930600
3272   7.808856  193.553212  17329.802160     8.061362  333.775777
3273   9.419510  175.762646  33155.578218     7.350233  333.775777
3274   5.126763  230.603758  11983.869376     6.303357  333.775777
3275   7.874671  195.102299  17404.177061     7.509306  333.775777

      Conductivity  Organic_carbon  Trihalomethanes  Turbidity  Potability  \
26      441.785876       13.805902        30.284597   4.184397           0
30      342.111286        7.922598        71.537953   5.088860           0
32      412.986834       12.890709        65.733478   5.057311           0
44      403.944168       18.668229        66.912400   4.542801           0
67      485.974500       11.351133        67.869964   4.620793           0
...            ...             ...              ...        ...         ...
3269    439.893618       16.172755        41.558501   4.369264           1
3272    392.449580       19.903225        66.396293   2.798243           1
3273    432.044783       11.039070        69.845400   3.298875           1
3274    402.883113       11.168946        77.488213   4.708658           1
3275    327.459760       16.140368        78.698446   2.309149           1

      Predicted_Potability
26                       1
30                       1
32                       1
44                       1
67                       1
...                    ...
3269                     1
3272                     1
3273                     1
3274                     1
3275                     1

[1185 rows x 11 columns]
```

In [ ]: