

Date - 17/10/2023

Team ID - 3933

Project Title - Water Quality Analysis

IMPORTING THE PACKAGES

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
import scipy
import plotly.express as px
```

READING THE DATASET

```
In [2]: df=pd.read_csv("D:\\IBM_water_quality\\water_potability.csv")
df_copy=pd.read_csv("D:\\IBM_water_quality\\water_potability.csv")
```

Data Exploration

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

```
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
...
3271	4.668102	193.681735	47580.991603	7.166639	359.948574	526.424171	13.894419	66.687695	4.435821	1
3272	7.808856	193.553212	17329.802160	8.061362	NaN	392.449580	19.903225	NaN	2.798243	1
3273	9.419510	175.762646	33155.578218	7.350233	NaN	432.044783	11.039070	69.845400	3.298875	1
3274	5.126763	230.603758	11983.869376	6.303357	NaN	402.883113	11.168946	77.488213	4.708658	1
3275	7.874671	195.102299	17404.177061	7.509306	NaN	327.459760	16.140368	78.698446	2.309149	1

3276 rows × 10 columns

```
In [4]: df_copy
```

```
Out[4]:
```

	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
...
3271	4.668102	193.681735	47580.991603	7.166639	359.948574	526.424171	13.894419	66.687695	4.435821	1
3272	7.808856	193.553212	17329.802160	8.061362	NaN	392.449580	19.903225	NaN	2.798243	1
3273	9.419510	175.762646	33155.578218	7.350233	NaN	432.044783	11.039070	69.845400	3.298875	1
3274	5.126763	230.603758	11983.869376	6.303357	NaN	402.883113	11.168946	77.488213	4.708658	1
3275	7.874671	195.102299	17404.177061	7.509306	NaN	327.459760	16.140368	78.698446	2.309149	1

3276 rows × 10 columns

ANALYSIS OF THE DATA

```
In [5]: print(df.columns)
```

```
Index(['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Conductivity',  
      'Organic_carbon', 'Trihalomethanes', 'Turbidity', 'Potability'],  
      dtype='object')
```

```
In [6]: df.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]: df.shape
```

Out[7]: (3276, 10)

```
In [8]: df[df["Potability"]==0].count()
```

Out[8]:

ph	1684
Hardness	1998
Solids	1998
Chloramines	1998
Sulfate	1510
Conductivity	1998
Organic_carbon	1998
Trihalomethanes	1891
Turbidity	1998
Potability	1998

dtype: int64

```
In [9]: df[df["Potability"]==1].count()
```

Out[9]:

ph	1101
Hardness	1278
Solids	1278
Chloramines	1278
Sulfate	985
Conductivity	1278
Organic_carbon	1278
Trihalomethanes	1223
Turbidity	1278
Potability	1278

dtype: int64

```
In [10]: df.describe()
```

Out[10]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
count	2785.000000	3276.000000	3276.000000	3276.000000	2495.000000	3276.000000	3276.000000	3114.000000	3276.000000	3276.000000
mean	7.080795	196.369496	22014.092526	7.122277	333.775777	426.205111	14.284970	66.396293	3.966786	0.390110
std	1.594320	32.879761	8768.570828	1.583085	41.416840	80.824064	3.308162	16.175008	0.780382	0.487849
min	0.000000	47.432000	320.942611	0.352000	129.000000	181.483754	2.200000	0.738000	1.450000	0.000000
25%	6.093092	176.850538	15666.690297	6.127421	307.699498	365.734414	12.065801	55.844536	3.439711	0.000000
50%	7.036752	196.967627	20927.833607	7.130299	333.073546	421.884968	14.218338	66.622485	3.955028	0.000000
75%	8.062066	216.667456	27332.762127	8.114887	359.950170	481.792304	16.557652	77.337473	4.500320	1.000000
max	14.000000	323.124000	61227.196008	13.127000	481.030642	753.342620	28.300000	124.000000	6.739000	1.000000

```
In [11]: df.dtypes
```

```
Out[11]: ph                float64
Hardness                float64
Solids                  float64
Chloramines              float64
Sulfate                  float64
Conductivity             float64
Organic_carbon           float64
Trihalomethanes          float64
Turbidity                float64
Potability               int64
dtype: object
```

Data Pre-Processing

```
In [12]: print(df.isnull().sum())
```

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

Check for Duplicates:

```
In [13]: duplicates = df.duplicated()
```

```
In [14]: df = df.drop_duplicates()
```

CLEARING THE NULL VALUES

```
In [15]: df.dropna(axis=0, how='any', inplace=True)
```

```
In [16]: df.isnull().sum()
```

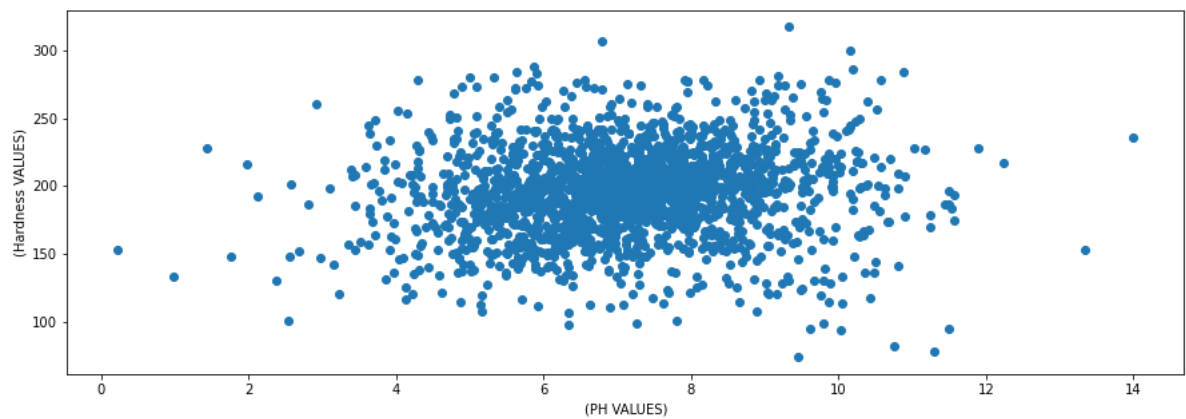
```
Out[16]: ph                0
Hardness                0
Solids                  0
Chloramines              0
Sulfate                  0
Conductivity             0
Organic_carbon           0
Trihalomethanes          0
Turbidity                0
Potability               0
dtype: int64
```

```
In [17]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2011 entries, 3 to 3271
Data columns (total 10 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ph                    2011 non-null  float64
1   Hardness              2011 non-null  float64
2   Solids                 2011 non-null  float64
3   Chloramines           2011 non-null  float64
4   Sulfate                2011 non-null  float64
5   Conductivity           2011 non-null  float64
6   Organic_carbon         2011 non-null  float64
7   Trihalomethanes        2011 non-null  float64
8   Turbidity              2011 non-null  float64
9   Potability             2011 non-null  int64
dtypes: float64(9), int64(1)
memory usage: 172.8 KB
```

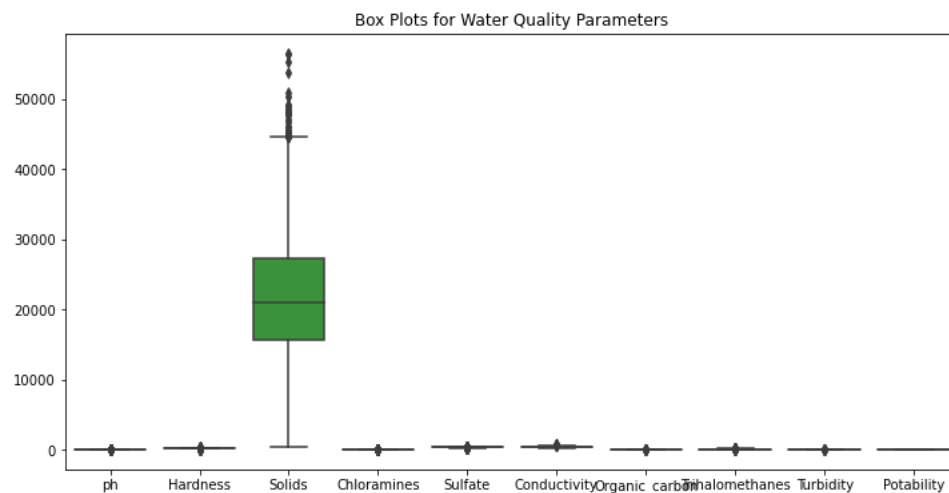
DETECTING OF OUTILERS

```
In [18]: fig, ax = plt.subplots(figsize = (15,5))
ax.scatter(df['ph'],df['Hardness'])
ax.set_xlabel('(PH VALUES)')
ax.set_ylabel('(Hardness VALUES)')
plt.show()
```



Type Markdown and LaTeX: α^2

```
In [19]: plt.figure(figsize=(12, 6))
sns.boxplot(data=df, orient="v")
plt.title("Box Plots for Water Quality Parameters")
plt.show()
```

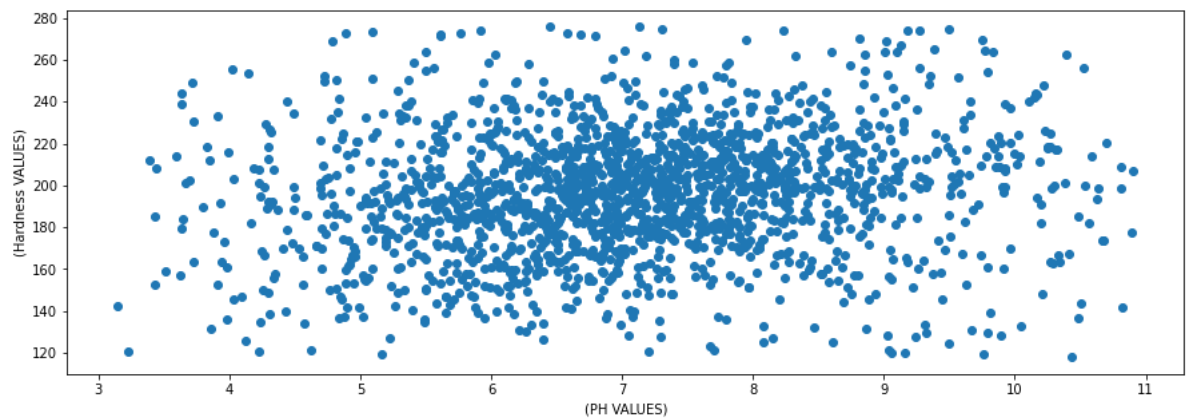


HANDLING OF OUTILERS

```
In [20]: def clear_outliers_iqr(df, columns):
cleaned_data = df.copy()
for column in columns:
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    cleaned_data = cleaned_data[(cleaned_data[column] >= lower_bound) & (cleaned_data[column] <= upper_bound)]
return cleaned_data

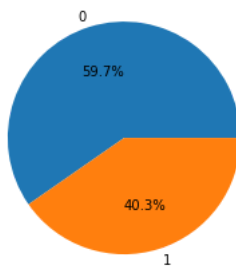
outlier_columns = ['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Conductivity', 'Organic_carbon', 'Trihalomethanes', 'Turbidity', 'Potability']
cleaned_data = clear_outliers_iqr(df, outlier_columns)
```

```
In [21]: fig, ax = plt.subplots(figsize = (15,5))
ax.scatter(cleaned_data['ph'],cleaned_data['Hardness'])
ax.set_xlabel('(PH VALUES)')
ax.set_ylabel('(Hardness VALUES)')
plt.show()
```



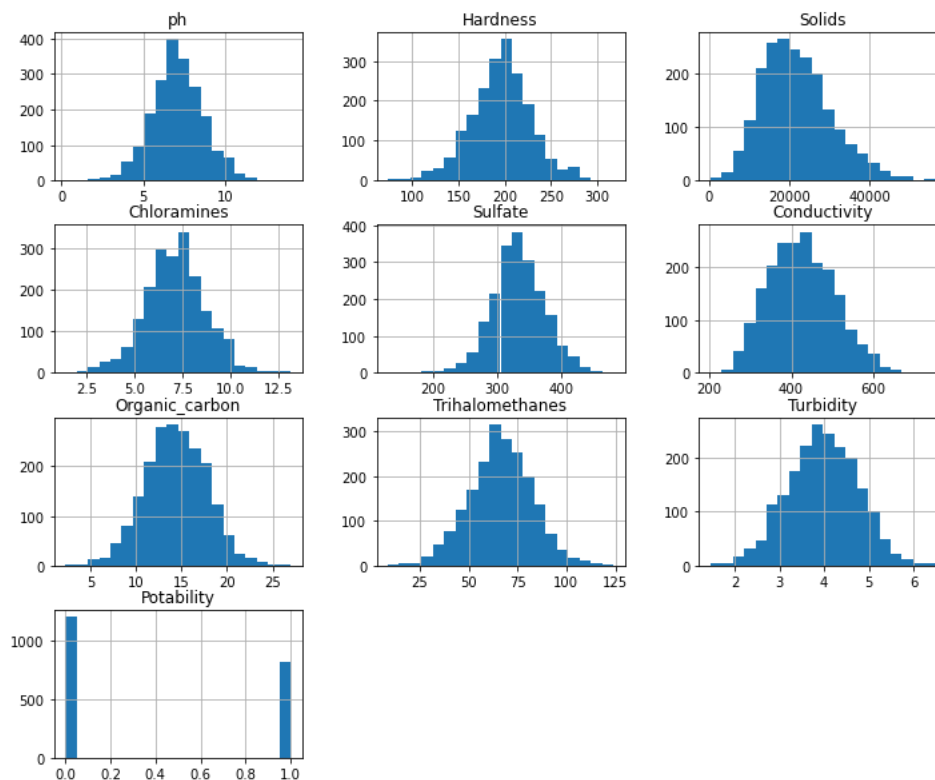
visualization of parameter

```
In [22]: plt.pie(df['Potability'].value_counts(),labels = list(df['Potability'].unique()),autopct="%0.1f%" )
plt.show()
```

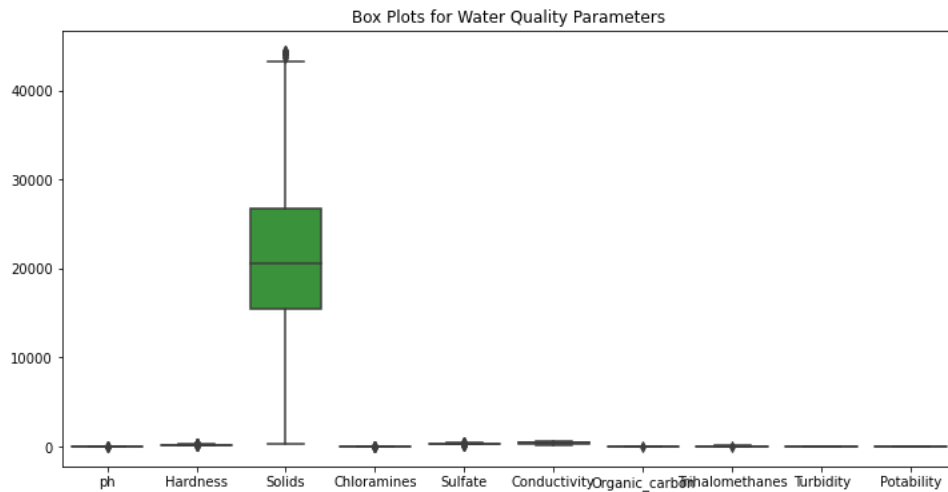


```
In [23]: df.hist(bins=20, figsize=(12, 10))
plt.suptitle("Histograms of Water Quality Parameters")
plt.show()
```

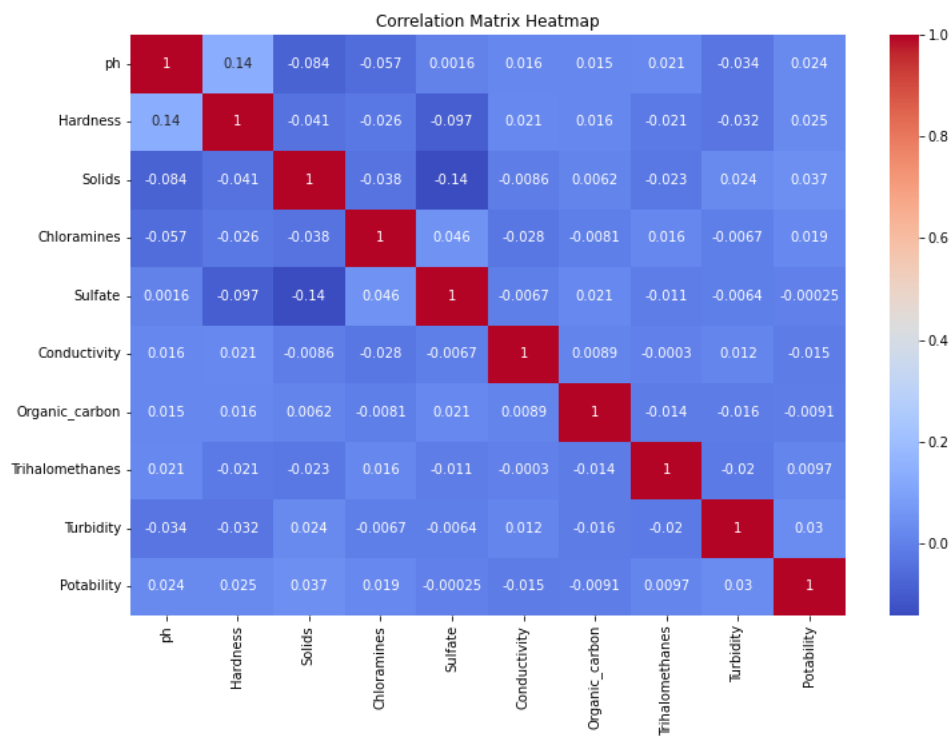
Histograms of Water Quality Parameters



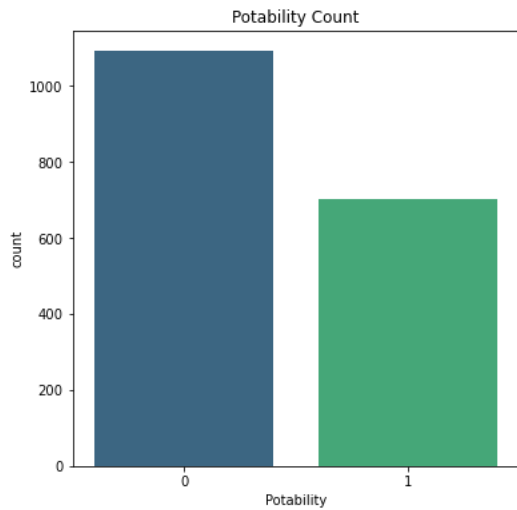
```
In [24]: plt.figure(figsize=(12, 6))
sns.boxplot(data=cleaned_data, orient="v")
plt.title("Box Plots for Water Quality Parameters")
plt.show()
```



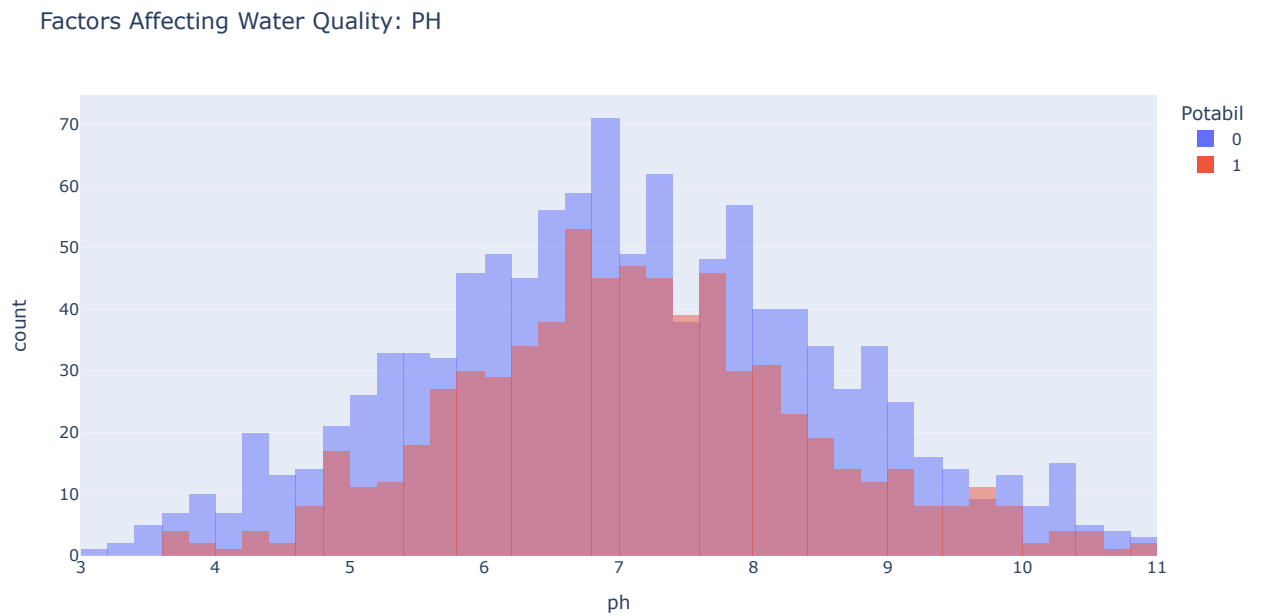
```
In [25]: correlation_matrix = cleaned_data.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Matrix Heatmap")
plt.show()
```



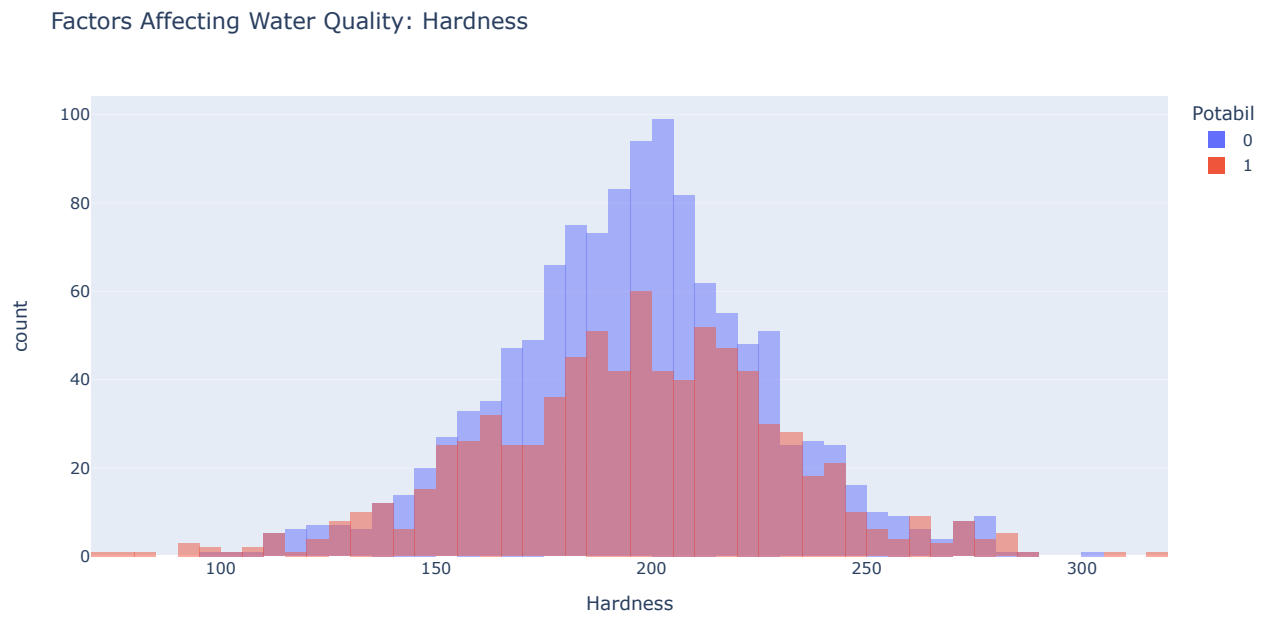
```
In [26]: plt.figure(figsize=(6, 6))
sns.countplot(data=cleaned_data, x='Potability', palette='viridis')
plt.title("Potability Count")
plt.show()
```



```
In [27]: px.histogram(cleaned_data, x="ph", color="Potability", barmode="overlay", title= "Factors Affecting Water Quality: PH")
```



```
In [88]: px.histogram(df, x = "Hardness",color = "Potability", barmode="overlay",title= "Factors Affecting Water Quality: Hardness")
```



```
In [89]: px.histogram(df, x = "Solids",color = "Potability", barmode="overlay",title= "Factors Affecting Water Quality: Solids")
```

