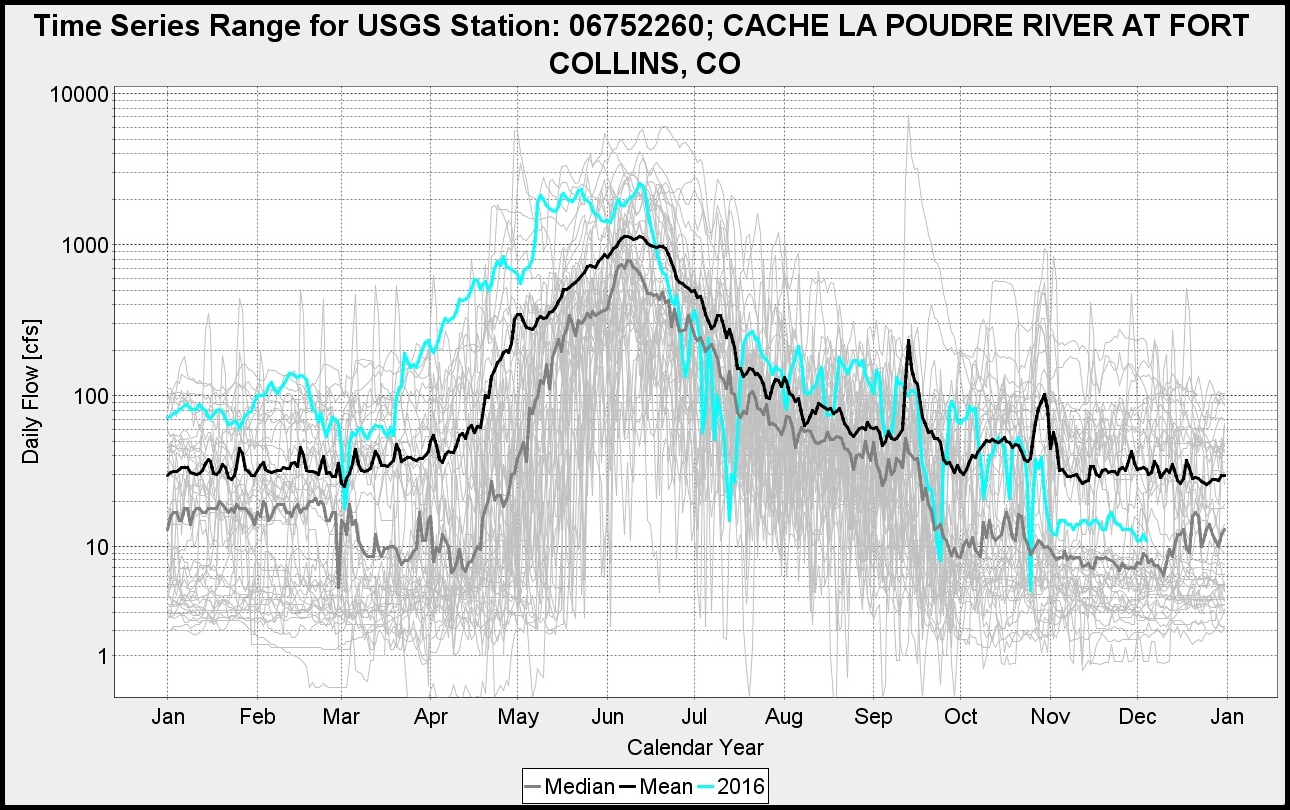
**WATER QUALITY ANALYSIS**

**DATA ANALYTICS WITH COGNOS-GROUP 1**

**PHASE\_4**

**Dharani.T[411521104025]**

****

**INTRODUCTION**

Water is a fundamental resource that sustains life and plays a pivotal role in various aspects of our daily existence. It serves as a source of drinking water, supports ecosystems, drives industrial processes, and contributes to agriculture.

Our project, Enhancing Water Quality aims to address this pressing issue and promote the sustainable management of water resources.

Analyzing water quality using data analytics involves the application of advanced computational techniques to extract valuable insights from water quality data. This approach can provide a deeper understanding of water conditions, identify trends, anomalies, and potential issues, and facilitate data-driven decision-making for water resource management. Here's an overview of water quality analysis using data analytics:

The second step of amalysis the water quanlity involves the following steps

* Visualizations the data
* Building predictive model

**Visualizations the data**

visualizing the data is created by using visualization libraries such as matplotlib and seaborn.

By using this libraries the histogram , scatter plots and correlation matrices were created.

**In [1]:** import pandas as pd

import numpy as np

import seaborn as sns

import plotly.express as px

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score,classification\_report,confusion\_matrix

**In [2]:** df=pd.read\_csv("water\_potability.csv")

df

**Out [2]:**

| **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 [3]:** df.columns

**Out [3]:** Index(['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Conductivity',

'Organic\_carbon', 'Trihalomethanes', 'Turbidity', 'Potability'],

dtype='object')

**In [4]:** df.describe()

**Out [4]:**

|  | **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 [5]:** df.info()

**Out [5]:** <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 [6]:** df.isnull().sum()

**Out[6]:** ph 491

Hardness 0

Solids 0

Chloramines 0

Sulfate 781

Conductivity 0

Organic\_carbon 0

Trihalomethanes 162

Turbidity 0

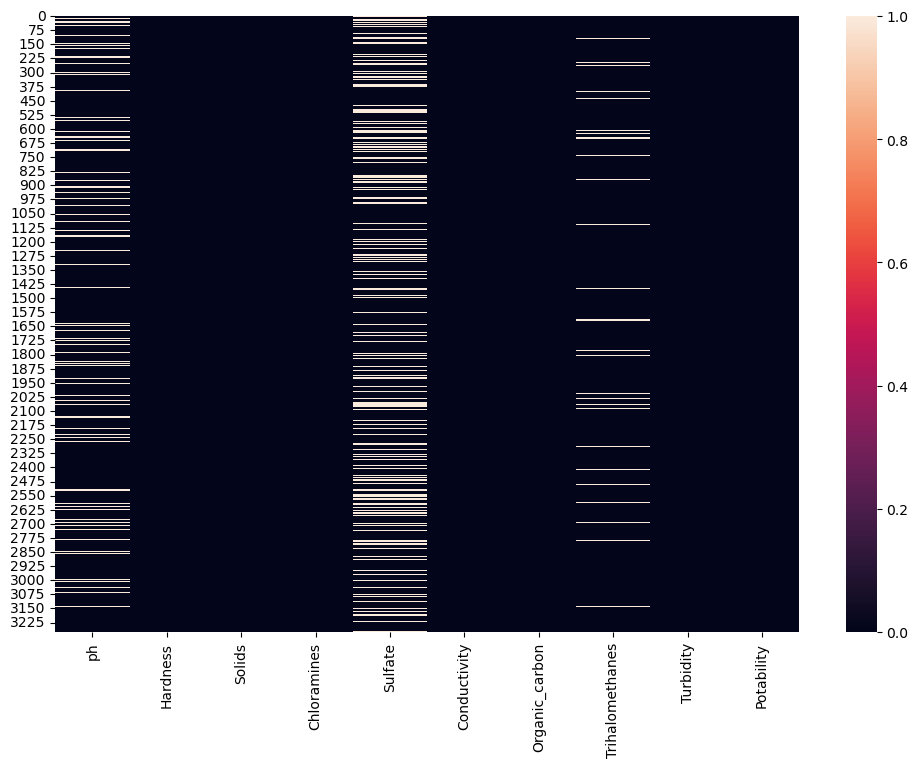
Potability 0

dtype: int64

**In [7]:** plt.figure(figsize=(12,8))

sns.heatmap(df.isnull())

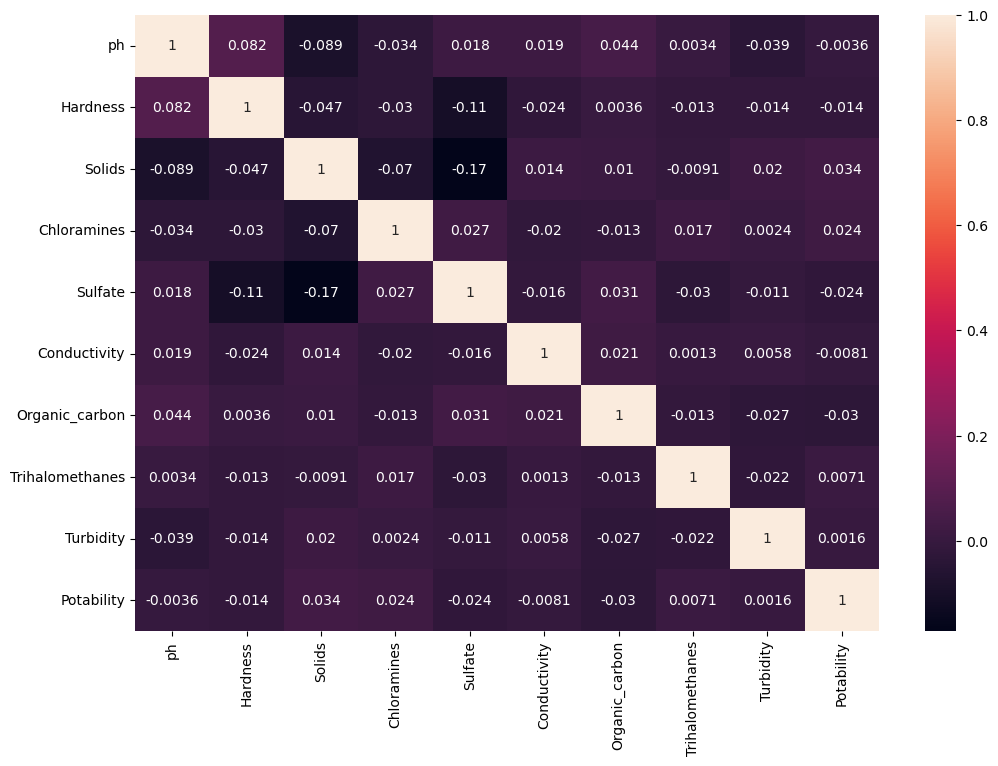
**Out [7]:** <AxesSubplot:>



**In [8]:** plt.figure(figsize=(12,8))

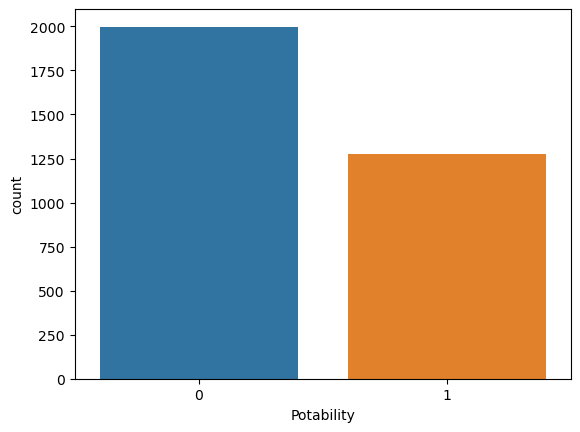
sns.heatmap(df.corr(),annot=True)

**Out [8]:** <AxesSubplot:>



**In [9]:** sns.countplot(x="Potability",data=df)

**Out [9]:** <AxesSubplot:xlabel='Potability', ylabel='count'>



**In [10]:** df["Potability"].value\_counts()

**Out [10]:** 0 1998

1 1278

Name: Potability, dtype: int64

**In [11]:** fig, ax=plt.subplots(ncols=5, nrows=2, figsize= (20,10))

ax=ax.flatten()

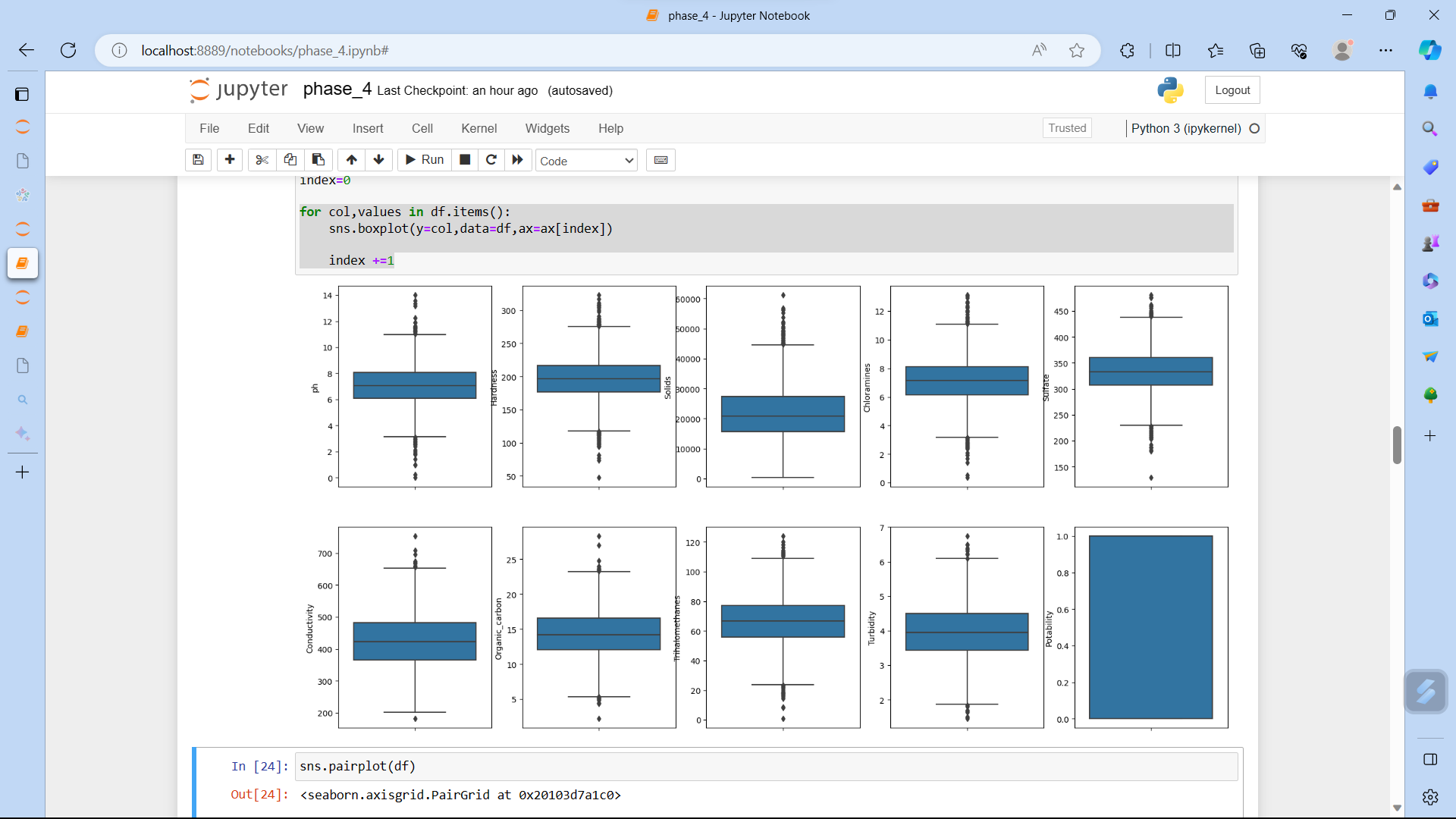
index=0

for col,values in df.items():

sns.boxplot(y=col,data=df,ax=ax[index])

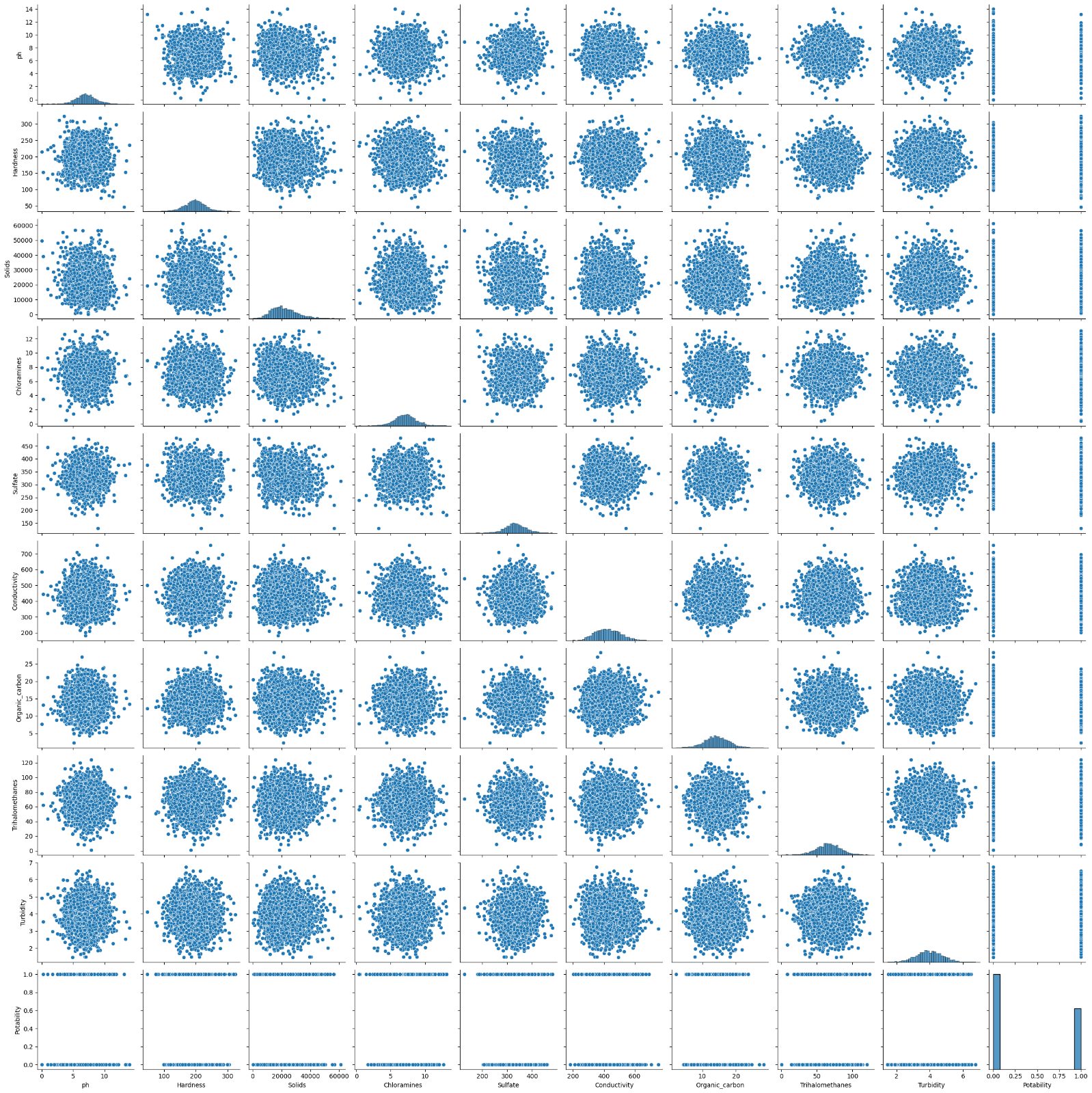
index +=1

**Out [11]:**

****

**In [12]:** sns.pairplot(df)

**Out [12]:** <seaborn.axisgrid.PairGrid at 0x20103d7a1c0>



**In [13]:** fig=px.pie(df,names="Potability",hole=0.4,template="plotly\_dark")

fig.show()

**Out [13]:**

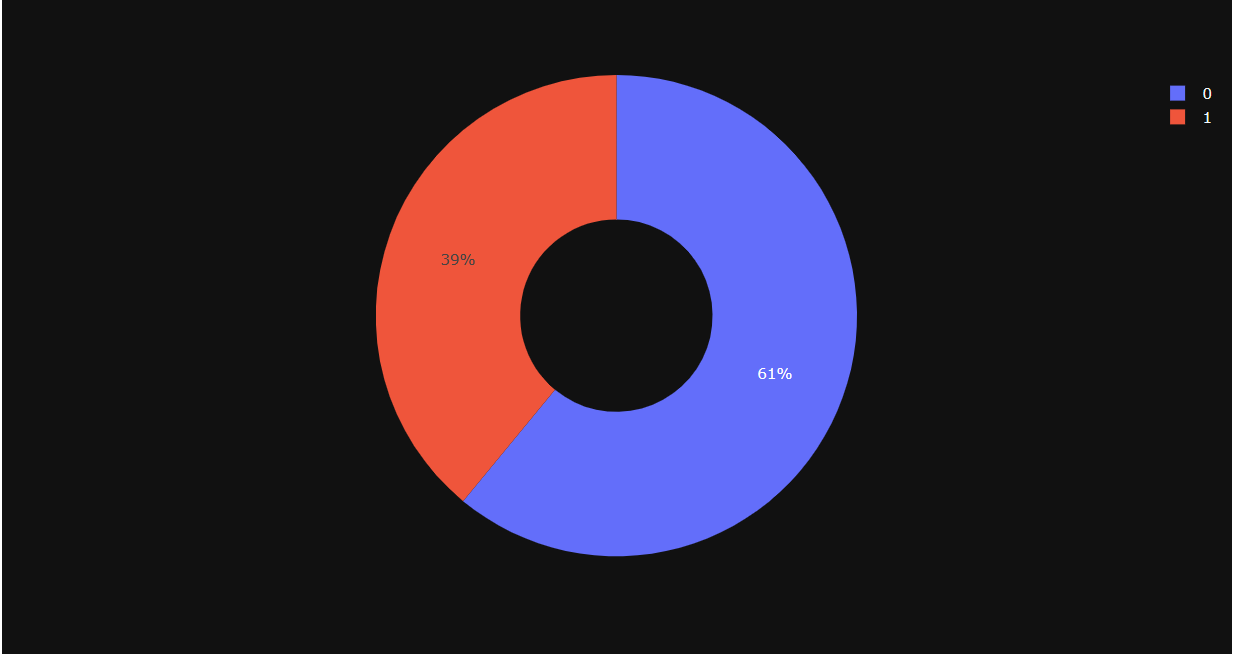
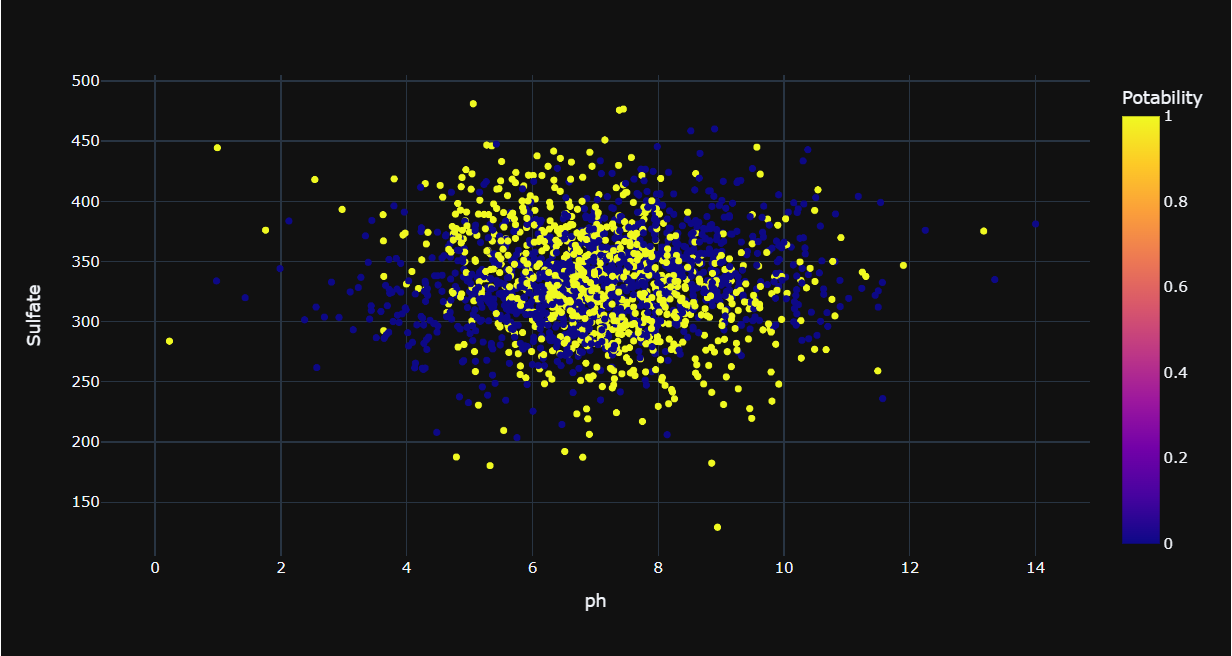
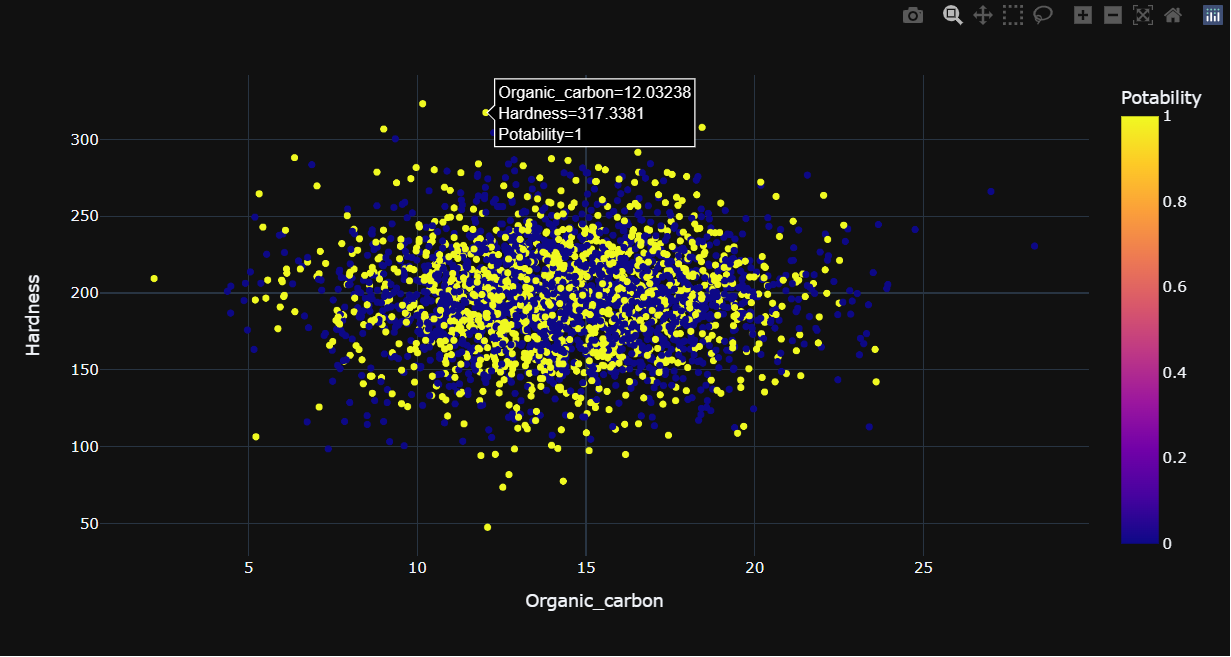
**In [14]:** fig = px.scatter(df, x="ph",y="Sulfate",color="Potability",template="plotly\_dark")

fig.show()

**Out [14]: **

**In [15**]: fig=px.scatter(df, x="Organic\_carbon",y="Hardness",color="Potability",template="plotly\_dark")

fig.show()

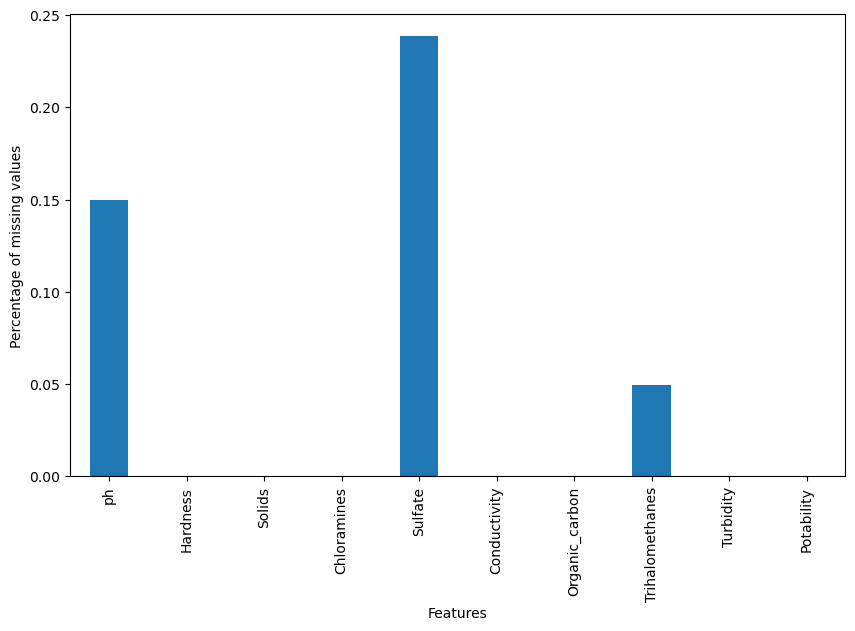
**Out [15]:**

**In [16]:** df.isnull().mean().plot.bar(figsize=(10,6))

plt.xlabel("Features")

plt.ylabel("Percentage of missing values")

**Out [16]:** Text(0, 0.5, 'Percentage of missing values')



**In [17]:** df["ph"]=df["ph"].fillna(df["ph"].mean())

df["Sulfate"]=df["Sulfate"].fillna(df["Sulfate"].mean())

df["Trihalomethanes"]=df["Trihalomethanes"].fillna(df["Trihalomethanes"].mea

n())

**In [18]:** df.isnull().sum()

**Out [18]:** ph 0

Hardness 0

Solids 0

Chloramines 0

Sulfate 0

Conductivity 0

Organic\_carbon 0

Trihalomethanes 0

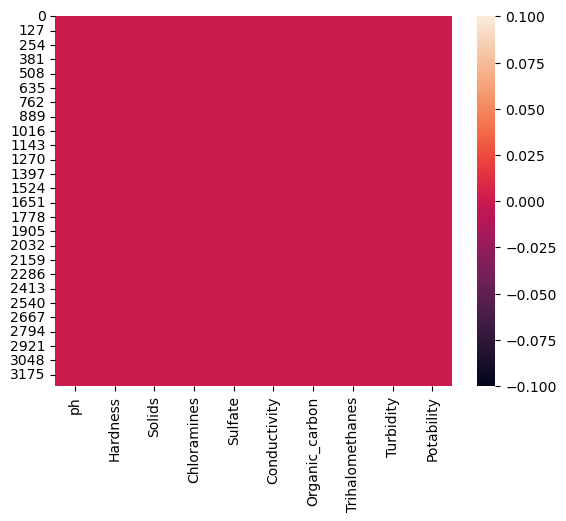
Turbidity 0

Potability 0

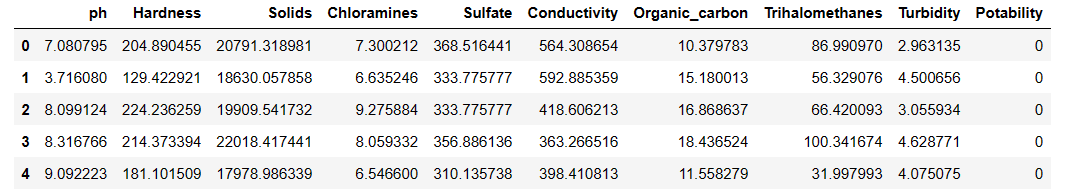
dtype: int64

**In [19]:** sns.heatmap(df.isnull())

**Out [19]:** <AxesSubplot:>



**In [20]:** df.head()

**Out [20]:**   
****

**In [21]:** x=df.drop("Potability",axis=1)

y=df["Potability"]

**In [22]:** x.shape,y.shape

**Out [22]:** ((3276, 9), (3276,))

**In[23**]: Scaler=StandardScaler()

x=Scaler.fit\_transform(x)

x

**Out [23]:** array([[-1.02733269e-14, 2.59194711e-01, -1.39470871e-01, ...,

-1.18065057e+00, 1.30614943e+00, -1.28629758e+00],

[-2.28933938e+00, -2.03641367e+00, -3.85986650e-01, ...,

2.70597240e-01, -6.38479983e-01, 6.84217891e-01],

[ 6.92867789e-01, 8.47664833e-01, -2.40047337e-01, ...,

7.81116857e-01, 1.50940884e-03, -1.16736546e+00],

...,

[ 1.59125368e+00, -6.26829230e-01, 1.27080989e+00, ...,

-9.81329234e-01, 2.18748247e-01, -8.56006782e-01],

[-1.32951593e+00, 1.04135450e+00, -1.14405809e+00, ...,

-9.42063817e-01, 7.03468419e-01, 9.50797383e-01],

[ 5.40150905e-01, -3.85462310e-02, -5.25811937e-01, ...,

5.60940070e-01, 7.80223466e-01, -2.12445866e+00]])

**In [24]:**

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2)

**In [25]:** x\_train.shape,x\_test.shape

**Out [25]:** (2620, 9), (656, 9))

**Building predictive model**

In this building predictive model is created for to determine water potability based on water quality parameters.

**In [26]:** #Logistic Regression

from sklearn.linear\_model import LogisticRegression

#Object of LR

model\_lr=LogisticRegression()

**In [27]:** model\_lr.fit(x\_train,y\_train)

**Out [27]:** LogisticRegression()

**In [28]:** #Making prediction

pred\_lr=model\_lr.predict(x\_test)

**In [29]:** #accuracy score

accuracy\_score\_lr=accuracy\_score(y\_test,pred\_lr)

accuracy\_score\_lr

**Out [29]:** 0.635670731707317

**Random Forest**

**In [30]:** from sklearn.ensemble import RandomForestClassifier

#Creating model object

model\_rf=RandomForestClassifier()

**In [31]:** #Training Model RF

model\_rf.fit(x\_train,y\_train)

**Out [31]:** RandomForestClassifier()

**In [32]:**

#Making Prediction

pred\_rf= model\_rf.predict(x\_test)

**In [33**]: accuracy\_score\_rf=accuracy\_score(y\_test,pred\_rf)

accuracy\_score\_rf\*100

**Out [33]:** 69.0548780487805

**In [34]:** cm3=confusion\_matrix(y\_test,pred\_rf)

cm3

**Out [34]:** array([[370, 45],

[158, 83]], dtype=int64)

**CONCLUSION**

Enhancing Water Quality in project is not just an analysis but a comprehensive effort to ensure the sustainable use of this invaluable resource. With the support of our dedicated team, engaged stakeholders, and the local community, we are committed to improving water quality, protecting the environment, and fostering a healthier and more sustainable future.

In this phase by using the dataset the data visualization and the predictive model is created