Introduction

This dataset provides information about various water quality parameters and whether the water is potable (safe for drinking) or not. It includes different physical and chemical properties of water that impact its drinkability.

Attributes Details

1) Ph:

- Description: Measures the acidity or alkalinity of water.
- Range: Typically between 0-14, where 7 is neutral.
- Missing Values: Yes (some values are missing).
- Importance: Extreme pH values can be harmful to health.

2) Hardness:

- Description: Indicates the concentration of dissolved calcium and magnesium in water.
- Units: mg/L (milligrams per liter).
- Missing Values: No.
- Importance: Hard water can cause scaling in pipes but is not harmful to health.

3) Solids:

- Description: Total dissolved solids (TDS) in water.
- Units: ppm (parts per million).
- Missing Values: No.
- Importance: High TDS may indicate contamination.

4) Chloramines:

- Description: Amount of chloramines used in water treatment.
- Units: mg/L.
- Missing Values: No.
- Importance: Disinfectant used to kill bacteria.

5) **Sulfate:**

- Description: Sulfate concentration in water.
- Units: mg/L.
- Missing Values: Yes.
- Importance: Excess sulfate can cause health issues like diarrhea.

6) **Conductivity:**

- Description: Measures water's ability to conduct electricity.
- Units: µS/cm (microsiemens per cm).
- Missing Values: No.
- Importance: Indicates the presence of dissolved salts.

7) **Organic Carbon:**

- Description: Organic carbon content in water.
- Units: mg/L.
- Missing Values: No.
- Importance: High levels indicate contamination.

8) **Trihalomethanes:**

- Description: Byproducts of chlorine disinfection.
- Units: µg/L (micrograms per liter).
- Missing Values: Yes.
- Importance: High levels may be harmful to health.

9) **Turbidity:**

- Description: Measures water clarity.
- Units: NTU (Nephelometric Turbidity Unit).
- Missing Values: No.
- Importance: High turbidity can indicate contamination.

10) Potability:

- Description: Indicates if water is safe to drink.
- Values: 1 = Potable (safe), 0 = Not potable (unsafe).
- Missing Values: No.
- Importance: Determines overall water quality.

Imports library and dataset:-

[5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
```

[7]:
```
df = pd.read_csv('water_potability.csv')
```

[9]:
```
df.head()
```

[9]:

	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

[11]:
```
df.shape
```

[11]:
```
(3276, 10)
```

Information of dataset:-

```
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
```

Describe the dataset :-

```
[15]: df.describe(include='all')
```

[15]:

	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

How many null values are there in the dataframe:-

```
# how many null values are there in the dataset
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
```

```
df.columns.tolist()
```

```
['ph',
 'Hardness',
 'Solids',
 'Chloramines',
 'Sulfate',
 'Conductivity',
 'Organic_carbon',
 'Trihalomethanes',
 'Turbidity',
 'Potability']
```

```
df.columns.unique()
```

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

Checking Unique Values in each column of a dataframe:-

```
for col in df.columns:
    print(df [col].unique())
```

```
[       nan 3.71608008 8.09912419 ... 9.41951032 5.12676292 7.87467136]
[204.89045547 129.42292051 224.23625939 ... 175.7626463  230.60375751
 195.10229859]
[20791.31898075 18630.05785797 19909.54173229 ... 33155.57821831
 11983.86937634 17404.17706105]
[7.30021187 6.63524588 9.2758836  ... 7.35023323 6.30335653 7.50930586]
[368.51644135          nan 356.88613564 ... 258.93060041 345.70025734
 359.94857437]
[564.30865417 592.88535913 418.60621306 ... 432.04478305 402.88311312
 327.45976046]
[10.37978308 15.18001312 16.86863693 ... 11.03906969 11.16894622
 16.14036763]
[86.99097046 56.32907628 66.42009251 ... 69.84540029 77.4882131
 78.69844633]
[2.96313538 4.50065627 3.05593375 ... 3.2988755  4.70865847 2.30914906]
[0 1]
```

Handling Missing Values in dataframe:-

```
for col in df.columns:
    if df[col].isnull().sum() > 0:
        df[col] = df[col].fillna(df[col].mean())

df.isnull().sum().sum()
```

```
0
```

```
df.isnull().sum()
```

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

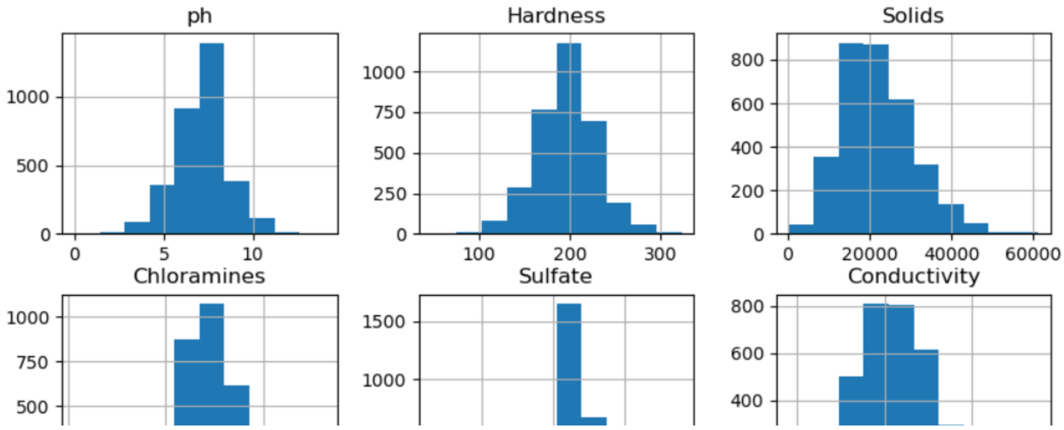
```
df['ph'].dtype
```

```
dtype('float64')
```

Visualizing Data Distribution Using Histograms:-

```
df.hist(figsize=(10,10))
```

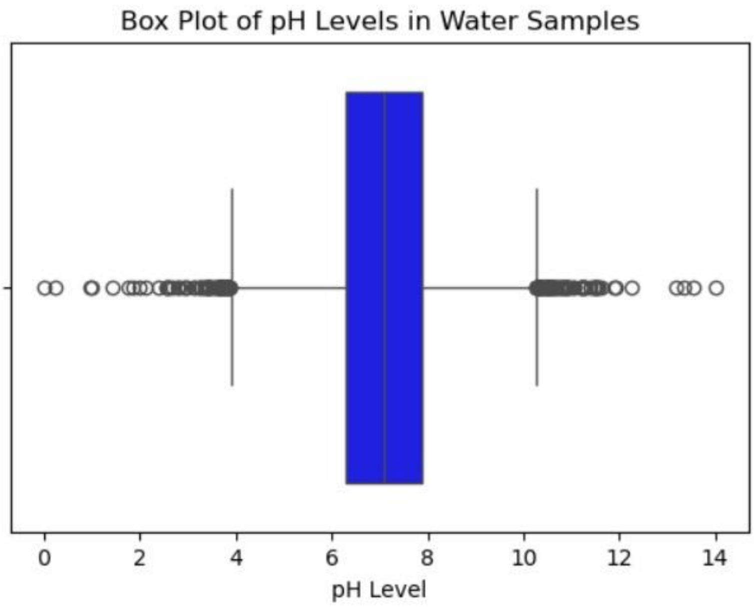
```
array([[<Axes: title={'center': 'ph'}>,
        <Axes: title={'center': 'Hardness'}>,
        <Axes: title={'center': 'Solids'}>],
       [<Axes: title={'center': 'Chloramines'}>,
        <Axes: title={'center': 'Sulfate'}>,
        <Axes: title={'center': 'Conductivity'}>],
       [<Axes: title={'center': 'Organic_carbon'}>,
        <Axes: title={'center': 'Trihalomethanes'}>,
        <Axes: title={'center': 'Turbidity'}>],
       [<Axes: title={'center': 'Potability'}>, <Axes: >, <Axes: >]],
      dtype=object)
```



Univariate Analysis of dataset

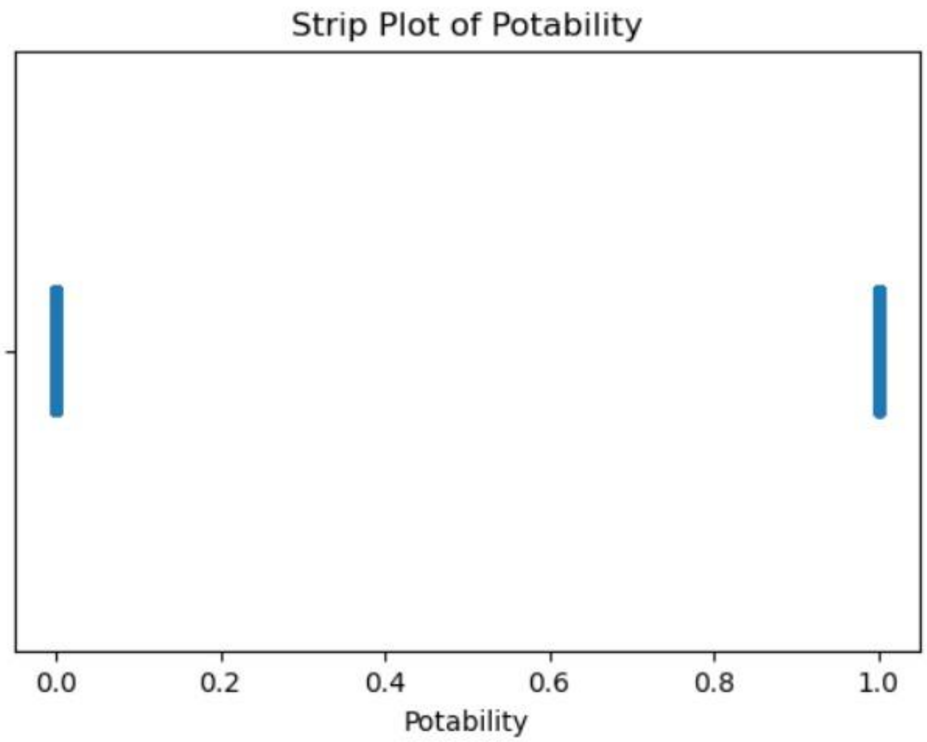
Box plot:-

```
#box plot
plt.figure(figsize=(6, 4))
sb.boxplot(x=df["ph"], color='blue')
plt.xlabel("pH Level")
plt.title("Box Plot of pH Levels in Water Samples")
plt.show()
```



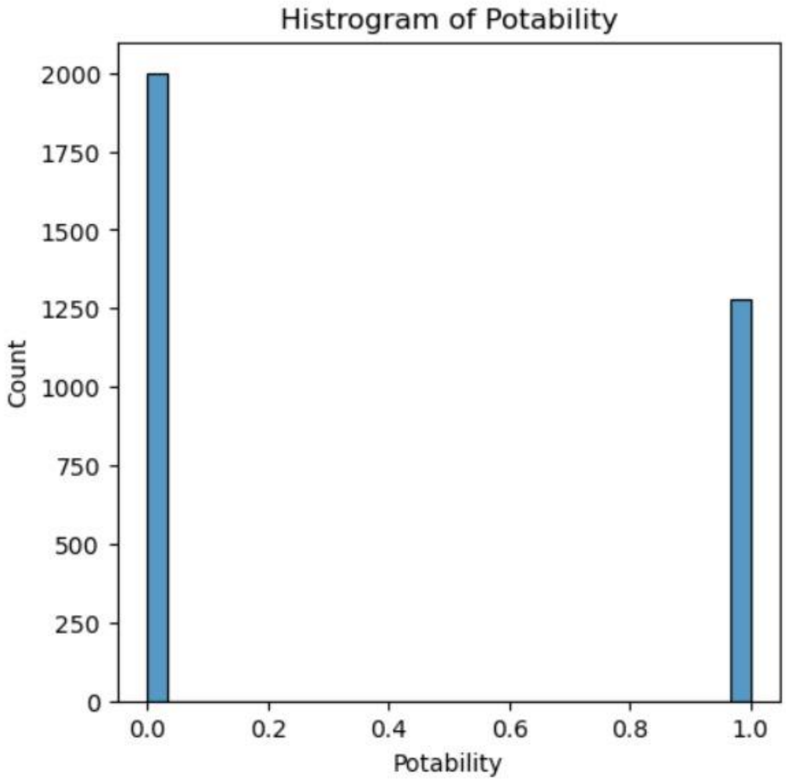
Strip plot:-

```
#strip plot
plt.figure(figsize=(6,4))
sb.stripplot(x=df[col], jitter=True)
plt.title(f"Strip Plot of {col}")
plt.show()
```



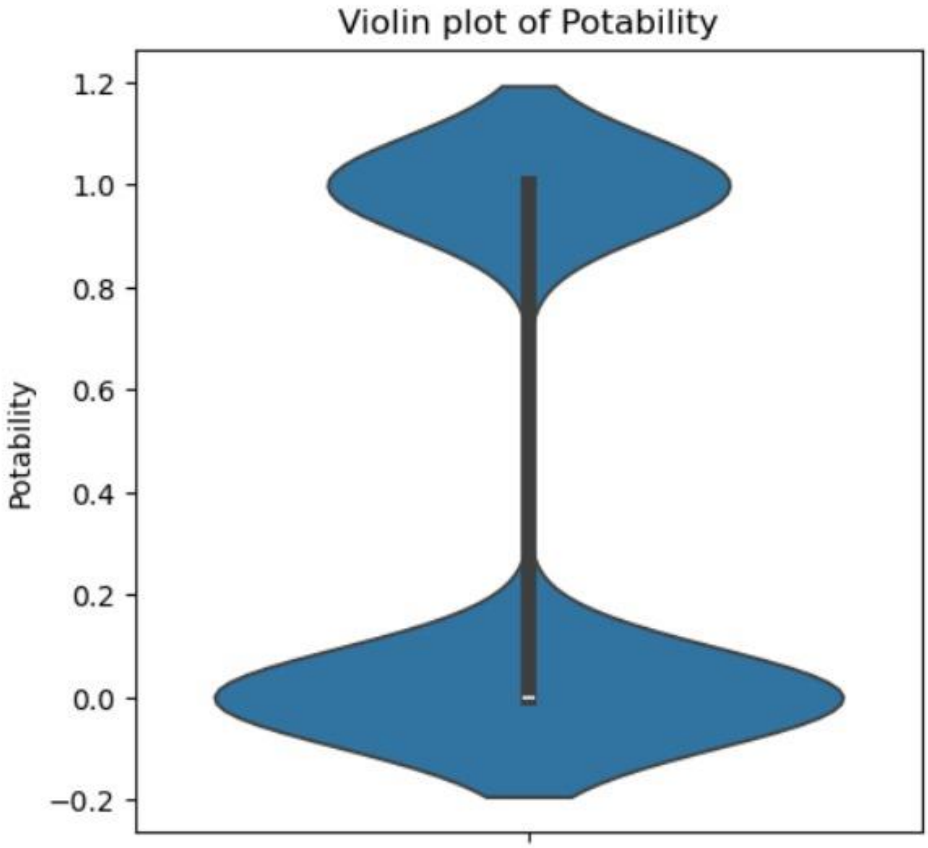
Histrogram plot:-

```
plt.figure(figsize=(5,5))
sb.histplot(df[col],bins=30)
plt.title(f"Histrogram of {col}")
plt.show()
```



Violin graph

```
plt.figure(figsize=(5,5))
sb.violinplot(df[col])
plt.title(f"Violin plot of {col}")
plt.show()
```



Outliers:-

```
column=df["ph"]
Q1=np.percentile(column, 25)
Q3=np.percentile(column, 75)
IQR = Q3-Q1
print("Q1:",Q1)
print("Q2:",Q3)
print("IQR",IQR)
```

```
Q1: 6.277672635884397
Q2: 7.870049755247176
IQR 1.5923771193627791
```

```
]: Q1=df[col].quantile(0.25)
Q3=df[col].quantile(0.75)
IQR = Q3-Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = df[(df[col]< lower_bound) | (df[col] > upper_bound)]
print(f"Number of Outliers: {len(outliers)}")
```

```
Number of Outliers: 0
```