PROJECT TITLE: WATER QUALITY ANALYSIS

PHASE 3: DATA PREPROCESSING

In this phase we are going to build our project by loading the given dataset and preprocessing the data.

STEPS FOLLOWED:

STEP 1: LOAD THE DATASET

Loading the water portability dataset which was given to my project.

Using the suitable library function to load the dataset. For example, pandas in python.

STEP2: FINDING THE MISSING VALUES

After loading the dataset, identifying the missing values in the dataset by using the functions like isnull () or isna().

Determining whether to drop the missing values based upon the nature of the dataset.

STEP3: HANDLING THE MISSING VALUES

If the missing values consists of a significant portion of the data ,consider dropping the corresponding rows and columns.

STEP4: DETECTING OUTLIERS

Using the statistical methods such as the IQR to identify the outliers.

Determining whether outliers should be removed or transformed based upon the nature of the data.

STEP5: EXPLORATORY DATA ANALYSIS

Visualizing the parameter distributions using histograms to understand the data’s characteristics.

Analysing the correlations matrices or scatterplots to identify potential relationships.

The codes for performing the above steps and their respective output can be given below:

In [1]:
```python
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [2]:
```python
data=pd.read_csv("water_potability.csv")
```

In [3]:
```python
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]:
```python
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 [5]:
```python
data['Potability'].unique()
```

Out[5]:
```
array([0, 1], dtype=int64)
```

In [6]:
```python
data.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]: data.shape
```

```
Out[7]: (3276, 10)
```

```
In [8]: data=data.dropna()
```

```
In [10]: data.head()
```

Out[10]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
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
5	5.584087	188.313324	28748.687739	7.544869	326.678363	280.467916	8.399735	54.917862	2.559708	0
6	10.223862	248.071735	28749.716544	7.513408	393.663396	283.651634	13.789695	84.603556	2.672989	0
7	8.635849	203.361523	13672.091764	4.563009	303.309771	474.607645	12.363817	62.798309	4.401425	0

```
In [9]: def Outliers(column):
    q1=column.quantile(0.25)
    q3=column.quantile(0.75)
    IQR=q3-q1
    lower=q1-1.5*(IQR)
    upper=q3+1.5*(IQR)
    return column[(column<lower) | (column>upper)]
```

```
In [11]: outliers_dict = {}
for column in data.select_dtypes(include=['number']):
    outliers = Outliers(data[column])
    if not outliers.empty:
        outliers_dict[column] = outliers
for column, outliers in outliers_dict.items():
    print(f"Potential outliers in column '{column}':")
    print(outliers)
```

```
Potential outliers in column 'ph':
9       11.180284
317     11.301794
692      1.757037
726      0.227499
783     11.898078
810      0.989912
1023    11.027880
1162    11.244507
1231     2.690831
1303    12.246928
1343     2.569244
1353    11.534880
2075    14.000000
2096    11.568768
2165     2.803563
2189     2.558103
2263    11.235426
2300     2.974429
```

```
2300     2.974429
2343     2.538116
2681     2.376768
2895    13.349889
2899     1.431782
2925    11.563169
2932     2.925174
2945    11.496702
2993     3.102076
3017    11.496859
3088     2.128531
3094     1.985383
3108    11.449739
3269    11.491011
Name: ph, dtype: float64
Potential outliers in column 'Hardness':
51      100.457615
71      116.299330
88      300.292476
```

```
Name: Hardness, dtype: float64
Potential outliers in column 'Solids':
142     46140.126850
186     45222.506665
283     48621.563952
378     45249.449033
516     45510.584319
546     49074.730407
583     44652.363872
648     44612.751358
987     48002.084596
1068    55334.702799
1106    44586.812651
1186    56351.396304
1332    45166.912141
1343    48204.172192
1462    45939.689158
1527    46718.555965
1554    56488.672413
```

```
Potential outliers in column 'Chloramines':
272     12.580026
275     13.043806
322     11.078872
324     11.170789
351     13.127000
408      2.484380
434     12.062536
437      2.981379
454      2.993744
534     11.543190
738     11.523598
772      2.866073
806      2.862535
814     11.302831
1057    11.086526
1099     3.181183
1106     2.741712
```

```
Potential outliers in column 'Sulfate':
253     187.170714
272     192.033592
275     180.206746
345     444.970552
351     182.397370
365     187.424131
385     209.471058
680     223.235816
703     224.212503
781     445.938391
782     229.575561
810     444.375731
1106    219.148935
1186    219.553437
1189    227.348460
1366    203.444521
1412    442.761428
```

```
Potential outliers in column 'Conductivity':
66      669.725086
342     695.369528
1183    656.924128
1295    666.690618
2134    708.226364
2704    753.342620
2737    657.570422
Name: Conductivity, dtype: float64
Potential outliers in column 'Organic_carbon':
43      23.917601
698     23.569645
785      2.200000
876      4.966862
1390     4.371899
1447     4.861631
1536     5.218233
2057    24.755392
2082     5.188466
```

```
Name: Trihalomethanes, dtype: float64
Potential outliers in column 'Turbidity':
382     6.494249
593     1.680554
789     1.812529
990     6.357439
1073    6.389161
1290    1.496101
1892    1.492207
2377    6.226580
2757    6.307678
2921    6.494749
3042    1.450000
Name: Turbidity, dtype: float64
```

In [12]:
```
def replaceOut(column):
    q1=column.quantile(0.25)
    q3=column.quantile(0.75)
    IQR=q3-q1
    lower=q1-1.5*(IQR)
    upper=q3+1.5*(IQR)
    outliers=column[(column<=lower) | (column>=upper)]
    if not outliers.empty:
        column[outliers.index] = column.mean()
    return column
```

In [13]:
```
cor=data.corr()
```

In [14]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(cor,annot=True,cmap='coolwarm')
plt.show()
```