In [12]:
```
import pandas as pd

df = pd.read_csv("water_potability.csv")

print(df.head(10))

print(df.describe())
```

```
          ph    Hardness        Solids  Chloramines     Sulfate  Conductivity  \
0        NaN  204.890455  20791.318981     7.300212  368.516441    564.308654
1   3.716080  129.422921  18630.057858     6.635246         NaN    592.885359
2   8.099124  224.236259  19909.541732     9.275884         NaN    418.606213
3   8.316766  214.373394  22018.417441     8.059332  356.886136    363.266516
4   9.092223  181.101509  17978.986339     6.546600  310.135738    398.410813
5   5.584087  188.313324  28748.687739     7.544869  326.678363    280.467916
6  10.223862  248.071735  28749.716544     7.513408  393.663396    283.651634
7   8.635849  203.361523  13672.091764     4.563009  303.309771    474.607645
8        NaN  118.988579  14285.583854     7.804174  268.646941    389.375566
9  11.180284  227.231469  25484.508491     9.077200  404.041635    563.885481

   Organic_carbon  Trihalomethanes  Turbidity  Potability
0       10.379783        86.990970   2.963135           0
1       15.180013        56.329076   4.500656           0
2       16.868637        66.420093   3.055934           0
3       18.436524       100.341674   4.628771           0
4       11.558279        31.997993   4.075075           0
5        8.399735        54.917862   2.559708           0
6       13.789695        84.603556   2.672989           0
7       12.363817        62.798309   4.401425           0
8       12.706049        53.928846   3.595017           0
9       17.927806        71.976601   4.370562           0
                ph     Hardness        Solids  Chloramines      Sulfate  \
count  2785.000000  3276.000000   3276.000000  3276.000000  2495.000000
mean      7.080795   196.369496  22014.092526     7.122277   333.775777
std       1.594320    32.879761   8768.570828     1.583085    41.416840
min       0.000000    47.432000    320.942611     0.352000   129.000000
25%       6.093092   176.850538  15666.690297     6.127421   307.699498
50%       7.036752   196.967627  20927.833607     7.130299   333.073546
75%       8.062066   216.667456  27332.762127     8.114887   359.950170
max      14.000000   323.124000  61227.196008    13.127000   481.030642

       Conductivity  Organic_carbon  Trihalomethanes    Turbidity   Potability
count   3276.000000     3276.000000      3114.000000  3276.000000  3276.000000
mean     426.205111       14.284970        66.396293     3.966786     0.390110
std       80.824064        3.308162        16.175008     0.780382     0.487849
min      181.483754        2.200000         0.738000     1.450000     0.000000
25%      365.734414       12.065801        55.844536     3.439711     0.000000
50%      421.884968       14.218338        66.622485     3.955028     0.000000
75%      481.792304       16.557652        77.337473     4.500320     1.000000
max      753.342620       28.300000       124.000000     6.739000     1.000000
```

In [13]:
```
print(df.isnull().sum())


newer_df = df.dropna()
print(newer_df.head(10))
```

```
ph                 491
Hardness             0
Solids               0
Chloramines          0
Sulfate            781
Conductivity         0
Organic_carbon       0
Trihalomethanes    162
Turbidity            0
Potability           0
dtype: int64
          ph    Hardness        Solids  Chloramines     Sulfate  \
3   8.316766  214.373394  22018.417441     8.059332  356.886136
4   9.092223  181.101509  17978.986339     6.546600  310.135738
5   5.584087  188.313324  28748.687739     7.544869  326.678363
6  10.223862  248.071735  28749.716544     7.513408  393.663396
7   8.635849  203.361523  13672.091764     4.563009  303.309771
9  11.180284  227.231469  25484.508491     9.077200  404.041635
10  7.360640  165.520797  32452.614409     7.550701  326.624353
12  7.119824  156.704993  18730.813653     3.606036  282.344050
15  6.347272  186.732881  41065.234765     9.629596  364.487687
17  9.181560  273.813807  24041.326280     6.904990  398.350517

    Conductivity  Organic_carbon  Trihalomethanes  Turbidity  Potability
3     363.266516       18.436524       100.341674   4.628771           0
4     398.410813       11.558279        31.997993   4.075075           0
5     280.467916        8.399735        54.917862   2.559708           0
6     283.651634       13.789695        84.603556   2.672989           0
7     474.607645       12.363817        62.798309   4.401425           0
9     563.885481       17.927806        71.976601   4.370562           0
10    425.383419       15.586810        78.740016   3.662292           0
12    347.715027       15.929536        79.500778   3.445756           0
15    516.743282       11.539781        75.071617   4.376348           0
17    477.974642       13.387341        71.457362   4.503661           0
```

In [14]:
```
print(newer_df.duplicated())
```

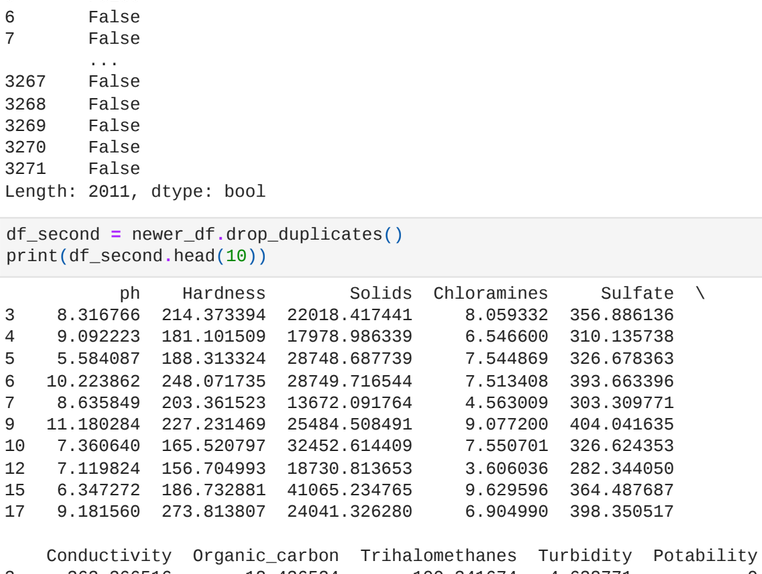
```
3       False
4       False
5       False
6       False
7       False
        ...
3267    False
3268    False
3269    False
3270    False
3271    False
Length: 2011, dtype: bool
```

In [15]:
```
df_second = newer_df.drop_duplicates()
print(df_second.head(10))
```

```
          ph    Hardness        Solids  Chloramines     Sulfate  \
3   8.316766  214.373394  22018.417441     8.059332  356.886136
4   9.092223  181.101509  17978.986339     6.546600  310.135738
5   5.584087  188.313324  28748.687739     7.544869  326.678363
6  10.223862  248.071735  28749.716544     7.513408  393.663396
7   8.635849  203.361523  13672.091764     4.563009  303.309771
9  11.180284  227.231469  25484.508491     9.077200  404.041635
10  7.360640  165.520797  32452.614409     7.550701  326.624353
12  7.119824  156.704993  18730.813653     3.606036  282.344050
15  6.347272  186.732881  41065.234765     9.629596  364.487687
17  9.181560  273.813807  24041.326280     6.904990  398.350517

    Conductivity  Organic_carbon  Trihalomethanes  Turbidity  Potability
3     363.266516       18.436524       100.341674   4.628771           0
4     398.410813       11.558279        31.997993   4.075075           0
5     280.467916        8.399735        54.917862   2.559708           0
6     283.651634       13.789695        84.603556   2.672989           0
7     474.607645       12.363817        62.798309   4.401425           0
9     563.885481       17.927806        71.976601   4.370562           0
10    425.383419       15.586810        78.740016   3.662292           0
12    347.715027       15.929536        79.500778   3.445756           0
15    516.743282       11.539781        75.071617   4.376348           0
17    477.974642       13.387341        71.457362   4.503661           0
```

In [16]:
```
import matplotlib.pyplot as plt
plt.hist(df['Conductivity'], bins=20)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Data Distribution')
plt.show()
```



In [ ]:
```
# 2. An outlier is an observation that deviates significantly from the rest of the
# data points in a dataset. It can be caused by various factors, such as measurement
# errors, natural variability, or anomalies. Outliers are important in data analysis
# because they can affect the summary statistics, such as mean and standard deviation,
# and distort the distribution of the data. Moreover, outliers can also influence the
# results of some machine learning algorithms that are based on distance or density
# measures, such as k-means clustering or k-nearest neighbors. Therefore, it is
# essential to detect and handle outliers appropriately before performing data analysis.

# 3. There are many ways of detecting outliers in a dataset, but some of the common ones are:

# Boxplot method: A boxplot is a graphical representation of the distribution
# of a numerical variable. It shows the median, quartiles, and whiskers of the data.
# Any data point that lies beyond 1.5 times the interquartile range (IQR) from the
# quartiles is considered an outlier and marked as a dot or a circle on the boxplot.
# This method is simple and easy to visualize, but it may not work well for skewed or multimodal distributions.
# Z-score method: A z-score is a measure of how many standard deviations a data
# point is away from the mean of the data. A data point with a z-score greater than
# 3 or less than -3 is considered an outlier. This method assumes that the data
# follows a normal distribution and is standardized. However, this method may not
# be robust to outliers themselves, as they can affect the mean and standard deviation of the data.
# IQR method: The IQR method is similar to the boxplot method, but it does not rely
# on a graphical representation. It calculates the IQR as the difference between the
# 75th and 25th percentiles of the data. Any data point that lies below Q1 - 1.5 *
# IQR or above Q3 + 1.5 * IQR is considered an outlier. This method is more robust than
# the z-score method, as it does not depend on the mean and standard deviation of the data.
# However, it may still not work well for skewed or multimodal distributions.
```