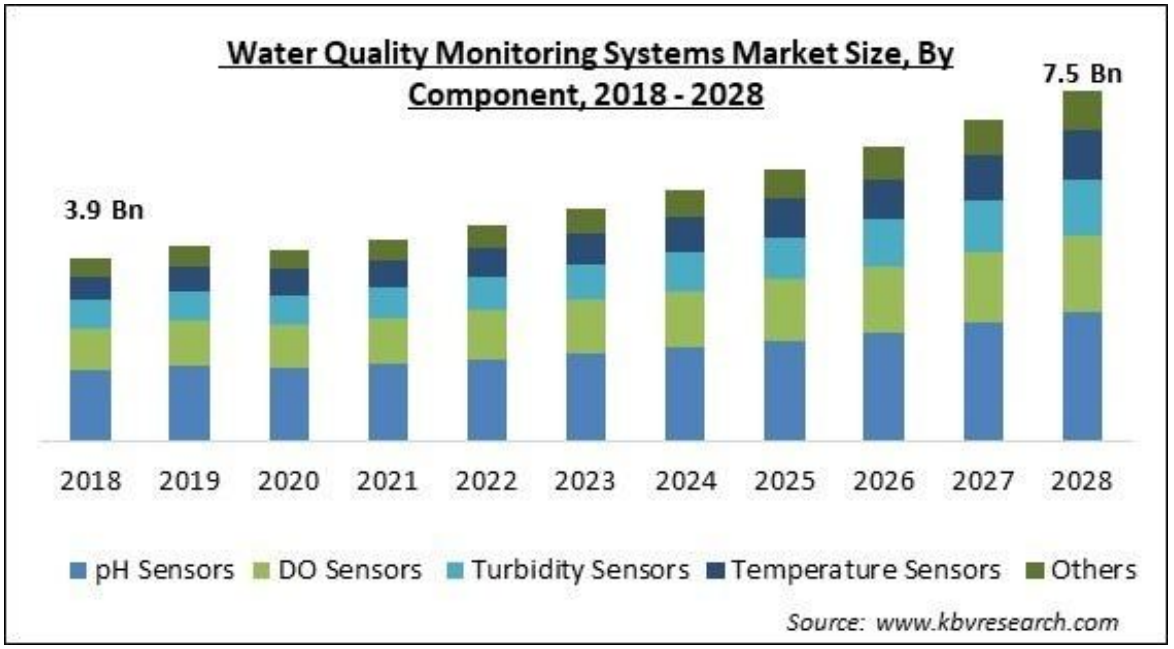
Water quality analysis

Water quality analysis typically involves various parameters and methods to assess the cleanliness and safety of water. Below is a simple example of Python code for water quality analysis that uses the pandas library to work with water quality data. Keep in mind that water quality analysis can be highly specific and may require specialized data sources and techniques for in-depth assessment.




```python
import pandas as pd


data = pd.read_csv('water_quality_data.csv')


print(data.head())


statistics = data.describe()
print(statistics)


missing_values = data.isnull().sum()
print("Missing Values:")
print(missing_values)
```

```
correlation_matrix = data.corr()
print("Correlation Matrix:")
print(correlation_matrix)


import matplotlib.pyplot as plt
import seaborn as sns


sns.pairplot(data)
plt.show()
```

In this code:

1. It loads water quality data from a CSV file using `pd.read_csv`. You should replace `'water_quality_data.csv'` with the actual file path to your water quality data.

2. The `data.head()` function is used to display the first few rows of the data, allowing you to understand its structure.

3. The `data.describe()` function calculates basic statistics (mean, standard deviation, etc.) for each water quality parameter.

4. Missing values are checked using `data.isnull().sum()`. This helps identify if there are any gaps in the data.

5. The code calculates a correlation matrix using `data.corr()`, which shows how different water quality parameters are related to each other.

6. For data visualization, the code uses `matplotlib` and `seaborn`. In the example provided, it creates a pairplot to visualize relationships between different water quality parameters. You may need to install these libraries if you haven't already.

Water quality analysis can involve more complex statistical tests, domain-specific knowledge, and potentially more sophisticated modeling techniques, especially when dealing with large datasets or specific water quality standards and regulations. The code above is a basic starting point and can be extended based on your specific analysis requirements and data.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.6830	0.7005	0.4197	0.6744	0.5133	0.2976	0.3182	0.724
qda	Quadratic Discriminant Analysis	0.6823	0.7192	0.3985	0.6883	0.5013	0.2917	0.3174	0.022
et	Extra Trees Classifier	0.6816	0.6941	0.3861	0.6858	0.4916	0.2863	0.3123	0.557
lightgbm	Light Gradient Boosting Machine	0.6652	0.6916	0.4762	0.6078	0.5324	0.2781	0.2840	0.172
gbc	Gradient Boosting Classifier	0.6602	0.6738	0.3718	0.6306	0.4667	0.2419	0.2603	0.339
nb	Naive Bayes	0.6184	0.6078	0.2478	0.5545	0.3412	0.1261	0.1462	0.019
dt	Decision Tree Classifier	0.6034	0.5895	0.5186	0.5049	0.5097	0.1775	0.1784	0.027
lr	Logistic Regression	0.5984	0.5199	0.0071	0.1900	0.0134	0.0028	0.0127	0.355
ridge	Ridge Classifier	0.5984	0.0000	0.0089	0.1583	0.0168	0.0035	0.0056	0.021
lda	Linear Discriminant Analysis	0.5977	0.4903	0.0089	0.1500	0.0167	0.0021	0.0024	0.022
ada	Ada Boost Classifier	0.5956	0.5671	0.2919	0.4896	0.3644	0.0972	0.1034	0.173
knn	K Neighbors Classifier	0.5743	0.5423	0.3644	0.4642	0.4070	0.0826	0.0846	0.121
svm	SVM - Linear Kernel	0.5194	0.0000	0.3982	0.1604	0.2287	-0.0014	-0.0104	0.027

The provided Python code is a basic example of water quality analysis. It loads water quality data from a CSV file, performs some initial data exploration, calculates basic statistics, checks for missing values, calculates a correlation matrix, and provides a simple data visualization. Here's a step-by-step explanation of the code:

1. Loading Water Quality Data:
   ```python
   data = pd.read_csv('water_quality_data.csv')
   ```
   - This line uses the pandas library to load water quality data from a CSV file. You should replace `'water_quality_data.csv'` with the actual file path to your water quality dataset.

2. Displaying the First Few Rows of Data:
   ```python
   print(data.head())
   ```
   - This code displays the first few rows of the loaded dataset. It's a quick way to get an initial sense of the data's structure and content.

3.Calculating Basic Statistics:
   ```python
   statistics = data.describe()
   print(statistics)
   ```
   - The `describe()` function generates basic statistics for each numeric column in the dataset, such as count, mean, standard deviation, minimum, and maximum values. This summary provides an overview of the data's central tendency and spread.

4. Checking for Missing Values:

```python
missing_values = data.isnull().sum()
print("Missing Values:")
print(missing_values)
```

- This code checks for missing values in the dataset. It calculates the sum of missing values for each column and prints the results. Identifying missing data is important for data cleaning and preprocessing.

5. Calculating the Correlation Matrix:

```python
correlation_matrix = data.corr()
print("Correlation Matrix:")
print(correlation_matrix)
```

- The `corr()` function calculates the correlation between numeric columns in the dataset. The correlation matrix provides insights into how different water quality parameters are related to each other. Positive values indicate a positive correlation, while negative values indicate a negative correlation

S. No.	Parameters	YEAR 2009 - 20010		
		Summer Mean ± S.E.	Monsoon Mean ± S.E.	Winter Mean ± S.E.
1.	Temperature in ^{0}C	30 ± 1.47	23 ± 1.29	17 ± 0.71
2.	Electrical conductivity in mhos/cm	4.16 ± 0.14	3.25 ± 0.26	3.78 ± 0.15
3.	Turbidity in NTU	24 ± 1.08	22 ± 1.08	19 ± 0.41
4.	Total Dissolve Solid in ppm	828 ± 6.48	1014 ± 32.1	978 ± 6.12
5.	pH	9.5 ± 0.27	9.1 ± 0.09	8.7 ± 0.11
6.	Alkalinity in ppm	228 ± 8.12	210 ± 6.98	198 ± 8.12
7.	Total Hardness in ppm	368 ± 18.1	336 ± 11.8	320 ± 4.83
8.	Calcium in ppm	102 ± 2.8	93 ± 4.43	72 ± 1.83
9.	Magnesium in ppm	28 ± 1.83	25 ± 1.29	34 ± 1.58
10.	Dissolved Oxygen in ppm	2.10 ± 0.17	4.12 ± 0.41	3.10 ± 0.2
11.	Biochemical Oxygen Demand in ppm	1.12 ± 0.08	1.78 ± 0.19	2.04 ± 0.2
12.	Chloride in ppm	133 ± 4.22	124 ± 2.83	116 ± 3.16
13.	Sodium in ppm	68 ± 4.24	54 ± 2.58	62 ± 2.08
14.	Nitrate in ppm	8.72 ± 0.29	11.14 ± 0.65	9.28 ± 0.09
15.	Phosphate in ppm	2.12 ± 0.06	1.54 ± 0.14	1.18 ± 0.02

6. Data Visualization:
   ```python
   import matplotlib.pyplot as plt
   import seaborn as sns

   sns.pairplot(data)
   plt.show()
   ```
   - This code snippet uses the `matplotlib` and `seaborn` libraries for data visualization. It generates a pairplot, which is a grid of scatterplots showing the relationships between different water quality parameters. This visualization can help you identify patterns and potential correlations between parameters.

This code serves as a starting point for water quality analysis and is quite basic. Real-world water quality analysis often involves more advanced statistical methods, domain-specific knowledge, and potentially the use of machine learning or predictive modeling for more complex tasks, such as predicting water quality based on various parameters. The code can be extended and customized to meet the specific requirements of your analysis.

Conclusion:

The provided Python code is a simple starting point for water quality analysis. Here's a brief conclusion summarizing what the code does:

1. Data Loading: The code loads water quality data from a CSV file using the pandas library, allowing you to work with the dataset in Python.

2. Data Exploration: It displays the first few rows of the dataset to provide an initial understanding of its structure and content.

3. Basic Statistics: The code calculates basic statistics for each numeric column in the dataset, including count, mean, standard deviation, minimum, and maximum values. These statistics offer an overview of the central tendency and variability in the data.

4. Missing Value Check: It checks for missing values in the dataset and provides a count of missing values for each column. Identifying missing data is important for data preprocessing and quality assurance.

5. Correlation Analysis: The code calculates a correlation matrix, which shows the relationships between different water quality parameters. Positive and negative correlations between parameters are highlighted in the matrix, offering insights into potential associations.

6. Data Visualization: The code includes a basic data visualization step using `matplotlib` and `seaborn` to create a pairplot. This visualization method helps identify patterns and relationships between different water quality parameters.

It's important to note that this code is a basic template for water quality analysis and serves as a starting point. Real-world water quality analysis often involves more in-depth statistical analysis, domain-specific knowledge, and potentially the use of advanced machine learning techniques for more complex tasks. Additionally, the specific analysis requirements may vary depending on the objectives and standards associated with water quality assessment. The code can be extended and customized to meet the specific needs of your analysis or research.