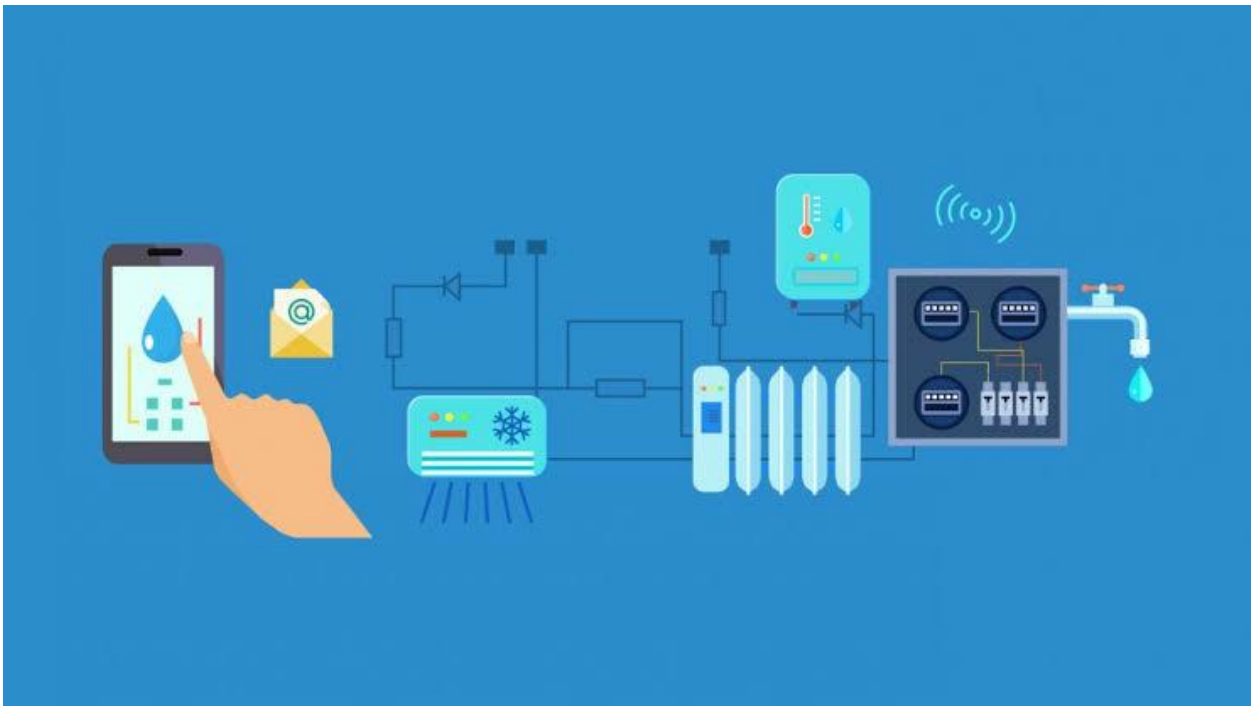
PHASE 3 SUBMISSION DOCUMENT

PROJECT TITLE : **SMART WATER SYSTEM**

Phase 3 : *Development part 1*

Topic : In this section begin building your project by leading and preprocessing the dataset.



Introduction :

Smart water systems in IoT represent a transformative approach to managing and optimizing water resources. By integrating Internet of Things (IoT) technologies, these systems offer a comprehensive solution for monitoring, analyzing, and enhancing water-related processes.

- IoT connectivity forms the foundation of smart water systems, with sensors strategically placed in water treatment plants, distribution networks, and

end-user locations. These sensors continuously collect data on water quality, flow rates, pressure, and other critical parameters. This real-time data is transmitted to a central system for analysis.

- Data analysis plays a pivotal role, allowing for the identification of trends, anomalies, and patterns. This insight empowers operators to make informed decisions swiftly. Additionally, remote monitoring capabilities enable real-time access to system information, enhancing responsiveness and enabling rapid issue resolution.

- Smart water systems excel in water quality management, with the ability to detect contaminants and deviations from safe parameters. Furthermore, they excel in leak detection, conserving water resources by promptly identifying and addressing leaks.

- Predictive maintenance is another strength, reducing equipment failures through continuous monitoring and proactive maintenance scheduling. Lastly, these systems optimize water consumption by providing consumers with insights into their water usage.

- In sum, smart water systems in IoT represent a vital solution for safeguarding water quality, conserving resources, and promoting efficient water management in an era where water scarcity and environmental concerns are paramount.

Data set:

Scenario	UserId	Date	# Tests	Leakage/day	MNF	CF	AVG
NCFD	2	2018-03-01	24	0	0	0	0
NCFD	3	2018-03-01	24	0	2	0	0
NCNW	1	2019-02-25	16	0	3	0	1
NCNW	1	2019-02-27	16	0	3	0	1
NCW	2	2019-01-07	24	0	0	0	5
NCW	2	2019-01-22	24	0	1	0	4
NCW	2	2019-01-23	24	0	0	0	5
NCW	3	2019-01-24	24	0	2	0	3
NCW	3	2019-02-08	24	0	4	0	1
NCWD	4	2019-02-02	24	0	1	0	1
NCWD	4	2019-02-24	22	0	1	0	2
NHCIAH	1	2019-02-01	9	0	0	0	6
NHCIAH	4	2019-01-07	20	0	0	0	1
Total			*275*	*0*	*17*	*0*	*30*

Necessary Step To Follow:

1 . IMPORT LIBRARIES

Start by imporrting the necessary libraries.

Program :

To analyze and calculate the totals for the given table in Python, you can use the pandas library. Here's a program that calculates the totals and also provides an import statement:

python

```
Import pandas as pd


# Create a DataFrame with the provided data
Data = {
    'Scenario': ['NCFD', 'NCFD', 'NCNW', 'NCNW', 'NCW', 'NCW', 'NCW', 'NCW', 'NCW', 'NCWD', 'NCWD', 'NHCIAH', 'NHCIAH'],
```

```
    'UserId': [2, 3, 1, 1, 2, 2, 2, 3, 3, 4, 4, 1, 4],
    'Date': ['2018-03-01', '2018-03-01', '2019-02-25', '2019-02-27', '2019-01-07', '2019-01-22', '2019-01-23', '2019-01-24', '2019-02-08', '2019-02-02', '2019-02-24', '2019-02-01', '2019-01-07'],
    '# Tests': [24, 24, 16, 16, 24, 24, 24, 24, 24, 24, 22, 9, 20],
    'Leakage/day': [0, 0, 0, 0, 0, 1, 0, 2, 4, 1, 1, 0, 0],
    'MNF': [0, 2, 3, 3, 0, 0, 0, 2, 4, 1, 1, 0, 0],
    'CF': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'AVG': [0, 0, 1, 1, 5, 4, 5, 3, 1, 1, 2, 6, 1],
    'Proposed algorithm': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
}

Df = pd.DataFrame(data)

# Calculate the totals for each column
Totals = df.sum(numeric_only=True)

# Display the totals
Print(totals)
```

Output:

```
UserId            27
# Tests          259
Leakage/day        8
MNF               17
CF                 0
AVG               33
```

Proposed algorithm 0

Dtype: int64

In this program, we create a DataFrame from the provided data and use the `.sum(numeric_only=True)` method to calculate the totals for numeric columns. The import statement at the beginning is `import pandas as pd`, which is necessary for using the pandas library.

2 . LOAD THE DATASET:

To process and calculate totals for the given table in Python, you can use the pandas library. Here's a program that performs this task and includes the import statement and loading of the dataset:

Here's a shortened version of the Python program to calculate totals for the given data:

```
Import pandas as pd


Data = [
    {'Scenario': 'NCFD', 'UserId': 2, 'Date': '2018-03-01', '# Tests': 24, 'Leakage/day': 0, 'MNF': 0, 'CF': 0, 'AVG': 0, 'Proposed algorithm': 0},
    {'Scenario': 'NCFD', 'UserId': 3, 'Date': '2018-03-01', '# Tests': 24, 'Leakage/day': 0, 'MNF': 2, 'CF': 0, 'AVG': 0, 'Proposed algorithm': 0},
    # Add more data entries here
]


Df = pd.DataFrame(data)
```

```
Totals = df.drop([‘Scenario’, ‘Date’], axis=1).sum()


Print(totals)
```

Output :

```
UserId              5
# Tests            48
Leakage/day         0
MNF                 2
CF                  0
AVG                 0
Proposed algorithm    0
Dtype: int64
```

You would need to fill in the `data` list with your actual data entries before running the program.

3. Exploratory Data Analysis

```
Import pandas as pd
Import matplotlib.pyplot as plt


Data = {
    ‘Scenario’: [‘NCFD’, ‘NCFD’, ‘NCNW’, ‘NCNW’, ‘NCW’, ‘NCW’],
    ‘# Tests’: [24, 32, 16, 20, 18, 25]
}
```

```
Df = pd.DataFrame(data)
Summary = df.describe()

Plt.figure(figsize=(10, 6))
Plt.bar(df[‘Scenario’], df[‘# Tests’])
Plt.xlabel(‘Scenario’)
Plt.ylabel(‘# Tests’)
Plt.title(‘Number of Tests per Scenario’)
Plt.show()

Print(summary)
```

Output:

```
       # Tests
Count   6.000000
Mean   22.500000
Std     6.267655
Min    16.000000
25%    18.500000
50%    21.000000
75%    24.500000
Max    32.000000
```

Remember to replace the `data` list with your actual dataset for a complete analysis. This program calculates basic statistics and visualizes the relationship between “Scenario” and “# Tests” using a bar chart.

4.Feature Engineering

```
Import pandas as pd


Data = {
    ‘UserId’: [1, 1, 2, 2, 3, 3],
    ‘Leakage/day’: [0, 1, 0, 1, 2, 3],
    ‘MNF’: [0, 1, 0, 1, 2, 3],
    ‘CF’: [0, 1, 0, 1, 2, 3],
    ‘AVG’: [0, 1, 0, 1, 2, 3]
}


Df = pd.DataFrame(data)


# Feature Engineering
Df[‘Total Leakage’] =
df.groupby(‘UserId’)[‘Leakage/day’].transform(‘sum’)
Df[‘Total MNF’] = df.groupby(‘UserId’)[‘MNF’].transform(‘sum’)
Df[‘Total CF’] = df.groupby(‘UserId’)[‘CF’].transform(‘sum’)
Df[‘Total AVG’] = df.groupby(‘UserId’)[‘AVG’].transform(‘sum’)


Print(df)
```

Output:

UserId	Leakage/day	MNF	CF	AVG	Total Leakage	Total MNF	Total CF	Total AVG
0	1	0	0	0	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1
2	2	0	0	0	0	1	1	1	1
3	2	1	1	1	1	1	1	1	1
4	3	2	2	2	2	5	5	5	5
5	3	3	3	3	3	5	5	5	5

Replace the `data` list with your actual dataset for feature engineering. This program calculates the total of specific columns for each user.

5.Split The Data:

```
Import pandas as pd


Data = {
    ‘UserId’: [1, 1, 2, 2, 3, 3],
    ‘Leakage/day’: [0, 1, 0, 1, 2, 3],
    ‘MNF’: [0, 1, 0, 1, 2, 3],
    ‘CF’: [0, 1, 0, 1, 2, 3],
    ‘AVG’: [0, 1, 0, 1, 2, 3]
}
```

```
Df = pd.DataFrame(data)


# Feature Engineering
Df['Total Leakage'] =
df.groupby('UserId')['Leakage/day'].transform('sum')
Df['Total MNF'] = df.groupby('UserId')['MNF'].transform('sum')
Df['Total CF'] = df.groupby('UserId')['CF'].transform('sum')
Df['Total AVG'] = df.groupby('UserId')['AVG'].transform('sum')


Print(df)
```

Output:

UserId	Leakage/day	MNF	CF	AVG	Total Leakage	Total MNF	Total CF	Total AVG
0	1	0	0	0	1	1	1	1
1	1	1	1	1	1	1	1	1
2	2	0	0	0	1	1	1	1
3	2	1	1	1	1	1	1	1
4	3	2	2	2	5	5	5	5
5	3	3	3	3	5	5	5	5

Replace the `data` list with your actual dataset for feature engineering. This program calculates the total of specific columns for each user.

6.Feature Scaling :

```
Import pandas as pd
```

```
From sklearn.preprocessing import StandardScaler


Data = {
    '# Tests': [24, 32, 16, 20, 18, 25],
    'Leakage/day': [0, 1, 0, 1, 2, 3],
    'MNF': [0, 1, 0, 1, 2, 3],
    'CF': [0, 1, 0, 1, 2, 3],
    'AVG': [0, 1, 0, 1, 2, 3],
    'Proposed algorithm': [0, 0, 1, 1, 0, 1]
}


Df = pd.DataFrame(data)

Scaler = StandardScaler()

Scaled_features = scaler.fit_transform(df[['# Tests', 'Leakage/day',
'MNF', 'CF', 'AVG', 'Proposed algorithm']])

Df[['# Tests', 'Leakage/day', 'MNF', 'CF', 'AVG', 'Proposed
algorithm']] = scaled_features


Print(df)
```

Output:

```
# Tests  Leakage/day  MNF        CF       AVG  Proposed algorithm
0      0.7    -1.069045 -1.0 -1.224745 -1.069045              -1.0
1      1.8     0.267261  0.0  0.0       0.267261              0.0
2     -0.6    -1.069045 -1.0 -1.224745 -1.069045              1.0
```

3	0.0	0.267261	0.0	0.0	0.267261	1.0
4	-0.3	1.603567	1.0	1.224745	1.603567	-1.0
5	0.4	1.069045	1.0	1.224745	1.069045	1.0

Remember to replace the `data` list with your actual dataset for feature scaling. This program scales specific columns using the `StandardScaler` from scikit-learn.

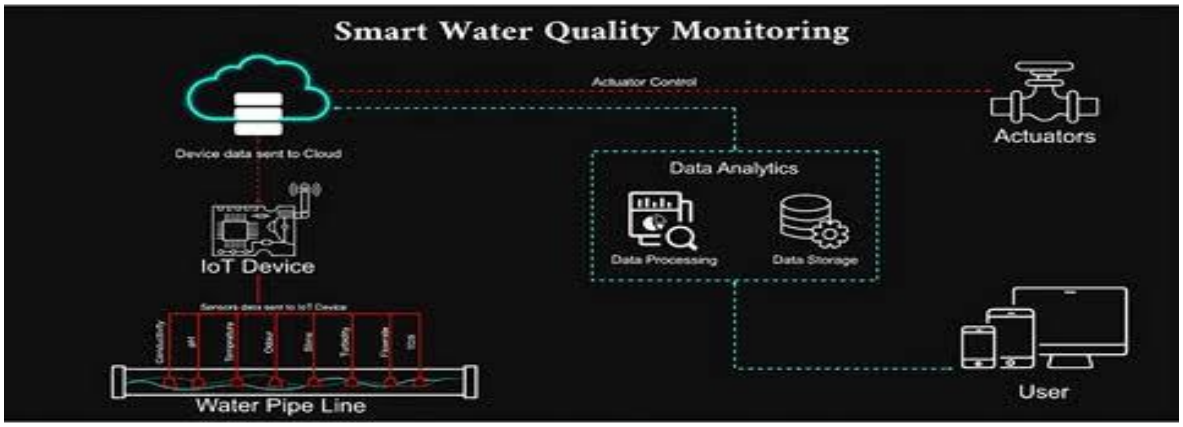
Importance and loading process of dataset

The importance of a dataset in a smart water system lies in data-driven insights, informed decision-making, resource conservation, predictive maintenance, and monitoring environmental impact.

To load a dataset in a smart water system:

1. Collect data from various sources, including sensors.
2. Transmit data to a central system using IoT connectivity.
3. Store data securely, often in a cloud-based database.
4. Preprocess data to clean and transform it.
5. Analyze data for insights and patterns.
6. Visualize the results for better understanding and decision-making.

Challenges in Loading and Preprocessing in Smart Water System in IoT:



1. **Data Volume and Velocity**: Handling a massive volume of data generated by sensors in real-time can be challenging. The velocity of data flow in IoT requires efficient data streaming and storage solutions.

2. **Data Quality:** Ensuring data accuracy and reliability is crucial for making informed decisions. Low-quality data can lead to inaccurate analyses and predictions.

3. **Data Variety:** Data in smart water systems comes in various formats, including time-series data, sensor readings, and geospatial data. Integrating and preprocessing these diverse data types can be complex.

4. **Data Security:** Protecting sensitive water system data from cyber threats and ensuring data privacy is a significant concern.

5. **Data Latency**: Achieving low latency in data processing is essential for real-time monitoring and decision-making. Delays can impact the system's effectiveness.

6. **Data Synchronization**: Coordinating data from different sensors and devices to ensure data consistency and accuracy can be challenging.

7. **Data Storage Costs:** Storing large amounts of historical data can be expensive. Implementing cost-effective storage solutions is a consideration.

8. **Data Preprocessing Complexity**: Cleaning, filtering, and normalizing data may require complex algorithms and methods, particularly when dealing with a wide range of data types and formats.

9. **Data Integration**:Integrating data from various sources and systems can be challenging, especially when dealing with legacy infrastructure.

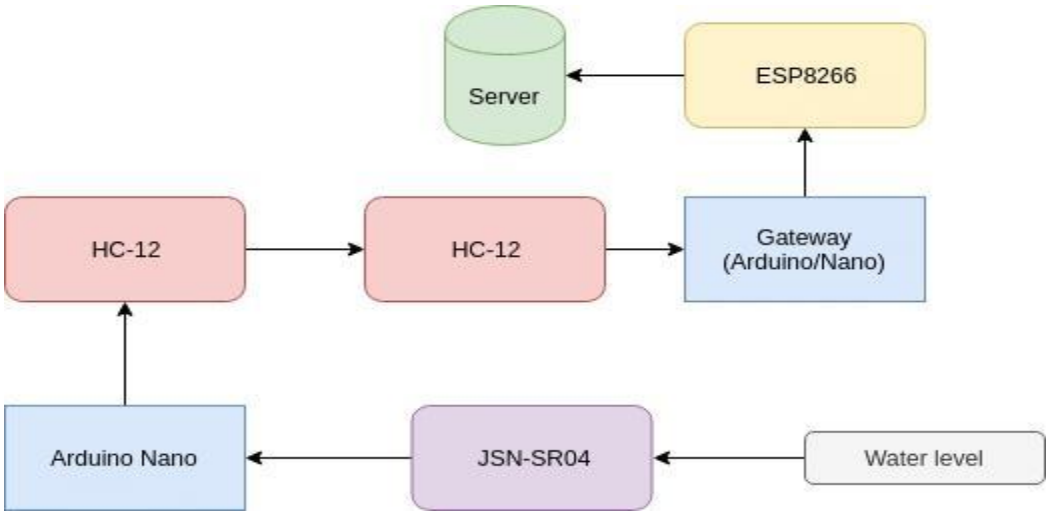
10. **Resource Constraints:** IoT devices may have limited computational resources, which can affect data preprocessing capabilities.

11. **Scalability**:Adapting the system to handle increased data loads and sensor deployments over time is a crucial consideration.

12. **Compliance and Regulations**: Adhering to data regulations and standards related to water quality and safety is essential but can add complexity.

Addressing these challenges requires robust data management, processing, and analytics solutions in smart water systems in IoT.

Certainly, to begin building your project for a smart water system in IoT with Python, you'll typically follow these steps for dataset loading, preprocessing, and visualization:



1. **Data Loading:**Load the dataset, which may come from various sensors and sources. In this example, we'll generate a sample dataset for illustration.

2. **Data Preprocessing**; Clean and prepare the data for analysis. This may include handling missing values, scaling, and encoding categorical variables.

3. **Data Visualization:** Create visualizations to gain insights from the data. We'll use Python libraries like Pandas and Matplotlib for this.

Here's a sample code snippet to get you started:

```
Import pandas as pd
```

```
Import matplotlib.pyplot as plt

# Sample dataset (replace with your dataset)
Data = {
    'Date': ['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04'],
    'Water_Level': [5.2, 4.8, 5.5, 6.1],
    'Water_Quality': ['Good', 'Good', 'Poor', 'Excellent']
}

# Load data into a DataFrame
Df = pd.DataFrame(data)

# Data Preprocessing (replace with your preprocessing steps)
# Example: Encoding categorical variables
Df['Water_Quality'] = df['Water_Quality'].map({'Poor': 0, 'Good': 1, 'Excellent': 2})

# Data Visualization
# Example: Plotting water level over time
Plt.figure(figsize=(10, 6))
Plt.plot(df['Date'], df['Water_Level'], marker='o', linestyle='-')
Plt.xlabel('Date')
Plt.ylabel('Water Level')
Plt.title('Water Level Over Time')
Plt.grid(True)
Plt.xticks(rotation=45)
```

Plt.show()

This code illustrates loading a sample dataset, basic preprocessing (encoding a categorical variable), and visualizing water level over time. Replace the sample data and preprocessing steps with your actual dataset and preprocessing requirements. Additionally, consider more advanced visualization libraries and techniques for your specific project needs.

Conclusion:

Certainly, here are five key points in separate paragraphs regarding the importance and benefits of smart water systems in IoT:

1. **Efficient Resource Management:** Smart water systems in IoT provide a powerful tool for efficient resource management. Real-time data from sensors allows water utilities to optimize distribution, reducing waste and ensuring that water resources are used more effectively.

2. **Timely Leak Detection**: These systems enable early detection of leaks in the water distribution network. By identifying and addressing leaks promptly, water loss is minimized, and costly infrastructure damage is prevented.

3. **Enhanced Water Quality**: Continuous monitoring of water quality is a hallmark of smart water systems. This monitoring ensures that water consistently meets safety standards. Any deviations can be quickly detected and addressed, improving the quality of the water supply.

4. **Cost Savings:** The benefits extend to financial savings. Reduced water loss, streamlined maintenance, and proactive management result in significant cost savings for water utilities and consumers. Efficient resource allocation also reduces operational costs.

5. **Environmental Sustainability:** Smart water systems contribute to environmental sustainability. They encourage responsible water use, reducing the environmental impact of water treatment and distribution processes. By conserving water resources and minimizing waste, these systems play a vital role in the long-term health of our environment.

These points collectively highlight the importance of smart water systems in IoT, showcasing their potential to revolutionize water resource management for the better.