A Big Data Analytics Approach to River Water Quality Assessment Using PySpark

Publication Date: October 20, 2025

Author: M . Rohith

Affiliation: MALLA REDDY UNIVERSITY, HYDERABAD

Abstract

Monitoring river water quality is essential for environmental health and public safety. The proliferation of monitoring stations generates vast datasets that require scalable ”Big Data” solutions for effective analysis. This paper demonstrates a complete analytics workflow using Apache PySpark to process, clean, and analyze the ’river_water_resources.csv’ dataset. The methodology includes loading large-scale CSV data, data cleaning and preprocessing, and performing distributed aggregations to derive insights. Key analyses include identifying state-wise trends in water quality, comparing pollution levels across different water body types (e.g., rivers, drains, canals), and filtering for ”problem areas” that fail to meet predefined environmental standards (Biochemical Oxygen Demand, Dissolved Oxygen, and pH). The findings highlight the power of PySpark in handling large-scale environmental data and provide a scalable model for automated water quality monitoring and reporting.

---

1. Introduction

Water is a critical natural resource, and ensuring its quality is a paramount challenge for governments and environmental agencies worldwide. With the increasing number of monitoring locations, the volume of collected data has grown exponentially, transitioning into the realm of Big Data. Traditional data analysis tools are often insufficient to handle the scale and velocity of this data. Apache Spark, with its Python API (PySpark), has emerged as a leading platform for large scale data processing. Its in-memory computation and distributed architecture make it ideal for the complex aggregations and filtering required in environmental science. This project utilizes PySpark within a Jupyter Notebook environment to conduct a comprehensive analysis of the river_water_resources.csv dataset. The primary objective is to uncover insights into water quality across different states and water body types, and to identify specific locations that exhibit signs of significant pollution.

---

2. Methodology

The analysis was conducted following a standard data analytics pipeline, adapted for a dis tributed environment using PySpark.

2.1. Data Loading and Environment Setup

A Spark Session was initialized as the entry point to the PySpark application. Theriver_water_resources.csv dataset was loaded into a Spark Data Frame using the spark.read.csv method, with options to infer the schema and use the first row as a header

2.2. Data Preprocessing and Cleaning

Real-world datasets are rarely clean. The initial data required two key preprocessing steps:

• Column Name Cleaning: The original column names (e.g. ,"BOD(mg/L)-Max") contained special characters, spaces, and parentheses, making them difficult to query. A function was applied to rename all columns, replacing special characters with underscores (e.g., BOD_mg_L_Max).

• Handling Missing Values: To ensure the accuracy of statistical calculations, all rows con taining any null or missing values were dropped from the Data Frame using the . dropna () method.

2.3. Data Analysis

- three primary analyses were performed using Spark's distributed aggregation and filtering capabilities.
- State-wise Analysis: The data was grouped by State_Name to calculate the total count of monitoring locations per state, as well as the average maximum Biochemical Oxygen Demand(Avg_Max_BOD) and average minimum Dissolved Oxygen (Avg_Min_DO).
- Water Body Type Analysis: The data was grouped by Type_Water_Body to compare aver age pollution metrics across rivers, drains, canals, and other body types.
- Problem Area Identification: A filter was applied to the entire dataset to identify high risk monitoring locations. A "problem area" was defined as any location meeting one or more of the following criteria:–

  High Biochemical Oxygen Demand:

  BOD _ mg_ L _Max > 6.0– Low Dissolved Oxygen:

  Dissolved _Min < 4.0– Extreme pH levels: pH _ Min < 6.5 or pH _ Max >.

---

3 Results and Visualization

the aggregated Spark Data Frames were converted to Pandas Data Frames for visualization, as the aggregated results were small enough to be handled in-memory on a single machine. This is a standard and efficient pattern for " Big Data to Small Data" visualization.

3.1 Summary of Findings

The analyses and subsequent visualizations (represented by placeholders in Figures 1-3) re vealed several key insights:

• Monitoring Disparity: A few states have a significantly higher number of monitoring lo cations than the rest of the country, indicating disparities in data collection infrastructure (Figure 1).

•Body Type Pollution: The ”Drain ”and” Canal ”water body types showed consistently higher pollution levels (higher BOD, lower DO) than ”River” or ”Sea” types, as expected. This conf irms that these artificial channels are major carriers of pollution (Figure 2).

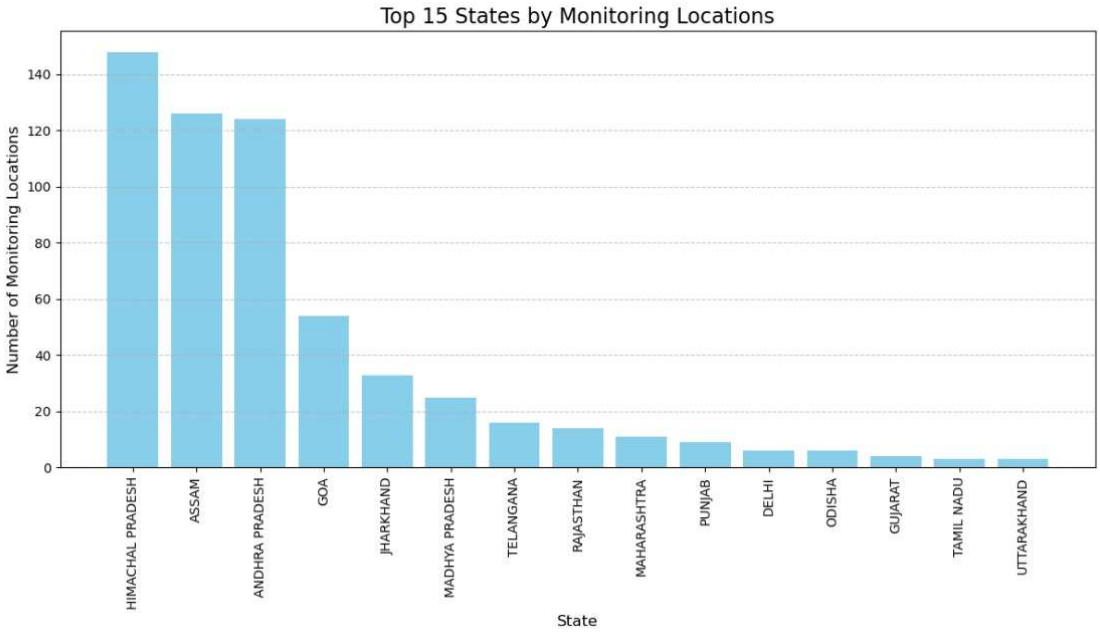

Figure1: Top 15 States by Density of Monitoring Locations.

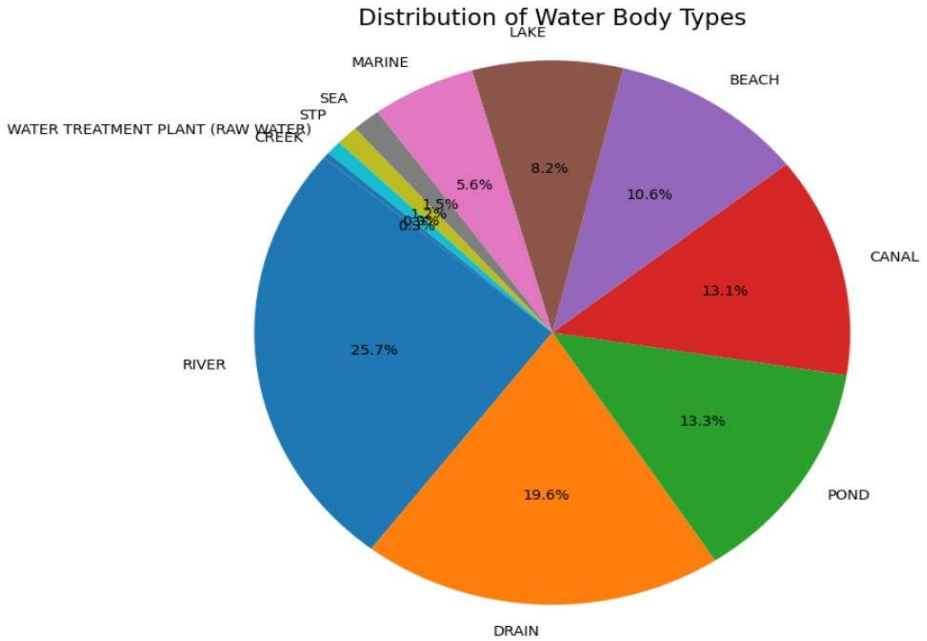

Figure2 : Distribution of Monitored Water Body Types.

3.2. Distribution Characteristics

Analysis of the value columns confirms a highly non-uniform data distribution:

- State-level Pollution: The analysis identified states with the highest average BOD levels, pointing to regions with significant challenges from industrial or domestic wastewater (Figure 3).
- Problem Hotspots: The filtering query successfully identified numerous specific monitoring locations that fail to meet basic water quality standards, providing a clear, actionable list for environmental protection agencies.

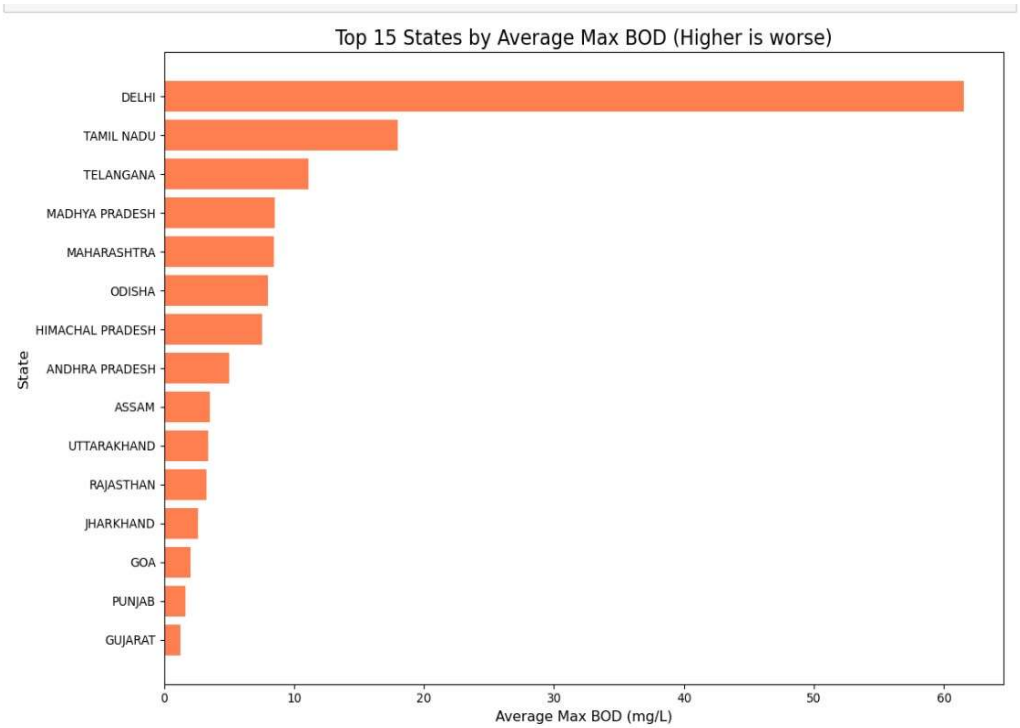

Figure 3: Top 15 States by Average Max BOD (Higher is worse).

---

5. Conclusion

This project successfully demonstrated the application of PySpark for a big data analytics work f low on water quality data. By leveraging Spark's distributed processing, we were able to effi ciently load, clean, and analyze a large dataset to extract meaningful insights. The findings confirm the scalability of this approach. The same PySpark code developed in this notebook

can be deployed on a large cluster to analyze datasets orders of magnitude larger (terabytes or petabytes) with minimal modification. This provides a robust foundation for building scalable, near-real-time environmental monitoring systems.