Robust Python Coding Techniques to Automate, Validate, and Strengthen Data Quality Checks in Complex Pipelines

Pavan Kumar Reddy Sakam, AstraZeneca, Bengaluru, India
Sapan Soni, AstraZeneca, Bengaluru, India

ABSTRACT

Ensuring high data quality is fundamental to reliable and compliant analytics in clinical research. As Python adoption increases across clinical data pipelines, there is a growing need for reusable, transparent, and automated data quality validation approaches. Many existing solutions rely on study-specific logic or manual quality control, resulting in inconsistency, limited reuse, and audit challenges. This paper presents a Python-based, test-driven data quality framework that applies modular rule design, metadata-driven configuration, and automated execution with structured outputs. A practical adverse event date validation use case demonstrates how Test-Driven Development improves consistency, traceability, and audit readiness. The framework also establishes a foundation for AI-enabled enhancements, supporting scalable and resilient data quality operations.

INTRODUCTION

Clinical data pipelines continue to grow in scale and complexity, driven by diverse data sources, increasing study volumes, and stringent regulatory expectations. While Python provides flexibility for data processing and validation, data quality practices often remain fragmented and manually intensive. Validation logic is frequently duplicated across studies or embedded within study-specific code, increasing maintenance effort and reducing auditability. These challenges highlight the need for standardized, reusable, and testable data quality validation frameworks.

DESIGN OBJECTIVES

The framework was designed with the following objectives: enabling reusable and consistent rule implementation, enforcing deterministic and testable validation behavior, externalizing study-specific parameters through metadata, producing audit-ready execution evidence, and supporting a scalable and AI-ready architecture.

FRAMEWORK ARCHITECTURE

Figure 1 illustrates the reference folder structure used in the framework. The separation of test definitions, rule logic, and metadata configuration supports clarity, reuse, and controlled extensibility across studies.

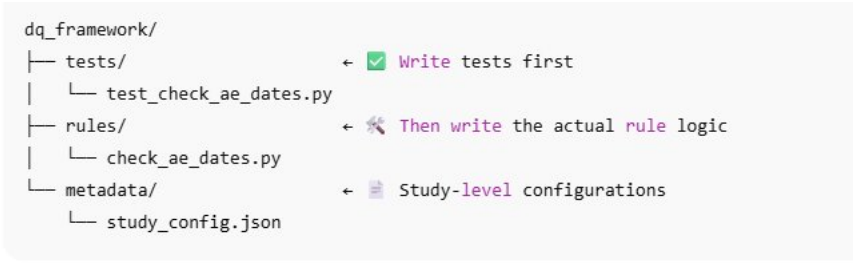

Figure 1: Example folder structure for the data quality framework.

The framework separates rule logic, metadata configuration, test definitions, and execution outputs. This modular design improves maintainability and enables consistent reuse across studies and domains. Rules are implemented as independent, testable components, while configuration files control study- or domain-specific parameters without modifying core logic.

TEST-DRIVEN DEVELOPMENT FOR DATA QUALITY

Test-Driven Development is applied by defining expected validation outcomes prior to rule implementation. Test cases explicitly assert valid and invalid date scenarios, while rule logic applies deterministic comparisons (e.g., start < end). Study-specific parameters, such as date formats, are externalized via metadata configuration to ensure reuse and consistency across studies.

Test-Driven Development principles are applied by defining expected rule behavior before implementation. Each rule is validated using automated test scenarios covering valid, invalid, and boundary conditions. An adverse event start and end date validation rule is used as a representative example to demonstrate deterministic and repeatable validation behavior.

TEST CASE DEFINITION

```
test_case = {
    "test_name": "start_after_end",
    "expected": False
}
```

This test asserts that validation fails when the adverse event start date occurs after the end date.

DETERMINISTIC RULE LOGIC

```
start = parse_date(start_date)
end = parse_date(end_date)
result = start < end
```

Rule logic applies a deterministic comparison to ensure temporal consistency between start and end dates.

METADATA-DRIVEN CONFIGURATION

```
{
  "date_format": "%Y-%m-%d",
  "null_handling": "fail"
}
```

Study-specific parameters such as date formats are externalized via metadata to avoid hard-coding and enable reuse

EXECUTION AND AUDIT EVIDENCE

Figure 2 presents a sample execution output generated by the framework. Each test scenario produces an explicit PASS or FAIL outcome, with a summary row aggregating results to support rapid operational review and audit traceability.

Validation execution produces structured, timestamped outputs that include test-level pass and fail results, summary metrics, and execution context. These outputs support operational review and provide traceable

evidence suitable for audit inspection.

	A	B	C	D	E	F	G
1	Test Executor: Pavan						
2	Study ID: D652637189						
3	Module: check_ae_dates						
4	Generated On: 20260108_073033						
5							
6	Test Name	Start Date	End Date	Expected	Actual	Status	
7	valid_dates	2023-01-01	2023-01-05	TRUE	TRUE	PASS	
8	invalid_dates	2023-01-10	2023-01-05	FALSE	FALSE	PASS	
9	equal_dates	2023-01-05	2023-01-05	FALSE	FALSE	PASS	
10	empty_start_date		2023-01-05	FALSE	FALSE	PASS	
11	empty_end_date	2023-01-01		FALSE	FALSE	PASS	
12	invalid_date_format	01-01-2023	05-01-2023	FALSE	FALSE	PASS	
13	future_dates	2025-12-31	2026-01-01	TRUE	TRUE	PASS	
14	start_after_end_on_same_month	2023-02-28	2023-02-01	FALSE	TRUE	FAIL	
15							
16	**Summary**					**PASS: 7**	**FAIL: 1**

Figure 2: Sample validation execution output with PASS and FAIL indicators.

AI-ENABLEMENT

Artificial intelligence is positioned as an augmentation layer operating on standardized validation outputs rather than replacing deterministic rule logic. Structured results and metadata context enable AI-assisted risk prioritization, anomaly detection, and trend analysis while preserving transparency and explainability.

RESULTS AND OBSERVATIONS

Application of the framework demonstrated improved consistency, reduced manual quality control effort, and enhanced traceability. Validation outcomes were reproducible and easily reviewed, supporting audit readiness and operational efficiency.

CONCLUSION

Applying software engineering principles such as modular design and Test-Driven Development to data quality validation improves reliability, transparency, and scalability. The proposed framework addresses current validation needs while providing a robust foundation for future AI-driven enhancements

CONTACT INFORMATION

Author Name: Pavan Kumar Reddy Sakam
Company: AstraZeneca
Email: sakampavankumar@gmail.com

Co Author Name: Sapan Soni
Company: AstraZeneca
Email: sapan.soni@astrazeneca.com