

APAC 2026

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

Pavan Kumar Reddy Sakam
Sapan Soni
AstraZeneca



Introduction & Problem Statement

During the study conduct phase, **vast volumes of clinical data** are generated.

Key challenges observed:

✗ Inconsistent or fragmented validation rules

⚠ Insufficient error handling mechanisms

🔒 Limited visibility and transparency in data flow

🚨 Data quality assurance becomes critical to enable timely and compliant downstream analysis



Python : Evolution to Adoption

Evolution

- 1980's by Guido van Rossum
- 1990–1994: Birth of Python 0.x–1.0
- 1995–2000: Python 1.x maturation(import packages)
- 2008 : Version 3.0
- 2024 (3.13): Ergonomics and stability

Applications

- Web and APIs
- Data Science and Analytics
- Machine Learning & AI
- Automation and Scripting
- Scientific Computing
- Data Engineering
- Game Development and Media
- NLP and GenAI

Industry Adaption

- Technology & Software
- Finance & FinTech
- Healthcare & Pharma
- Retail & E-commerce
- Manufacturing & Automotive
- Government & Public Sector
- Media & Entertainment
- Telecom
- Energy & Utilities
- Education & Research



Our Solution – Scalable, Python-Based Data Quality Framework

Modular validation
design with plug-
and-play functions

Metadata-driven
execution of
validation rules

Built-in test scripts
following Test-
Driven Development
(TDD)

Seamless
integration with AI-
enabled tools



TDD (Test-Driven Development)

- **Test-Driven Development (TDD) is a practice where tests are written before the actual rule or logic**
- **Tests describe the expected behavior of a Data Quality (DQ) rule**
- **A rule is developed only to satisfy the predefined tests**
- **This approach ensures clarity on “what is correct” before implementation**
- **TDD reduces ambiguity and improves reliability of DQ rules**



TDD within the Data Quality Framework

- Clinical data flows through ingestion, transformation, review, and reporting stages
- Data Quality checks are applied at multiple points in the data lifecycle
- TDD ensures DQ rules are validated before they are executed in the pipeline
- Each DQ rule is aligned to a specific data risk or validation requirement
- Early validation helps prevent downstream data quality issues

```
dq_framework/  
├─ tests/                ← ✓ Write tests first  
│   └─ test_check_ae_dates.py  
├─ rules/               ← ✖ Then write the actual rule logic  
│   └─ check_ae_dates.py  
└─ metadata/            ← 📄 Study-level configurations  
    └─ study_config.json
```


test_check_ae_dates.py


check_ae_dates.py


study_config.json



Applying TDD to a Data Quality Rule – Example

- **Example rule: Adverse Event end date must not be earlier than start date**
- **Business risk: Incorrect timelines can impact safety interpretation**
- **Test cases are defined for valid, invalid, and edge scenarios**
- **The DQ rule is finalized only after all test cases pass successfully**



Value Delivered and Key Outcomes

Implemented

Established a structured, test-driven and configuration-based Data Quality framework to consistently validate clinical data rules across studies.

Key Outcomes

Faster and more consistent Data Quality validation

Clear pass/fail results with automated execution evidence

Improved audit readiness through traceable and repeatable outcomes

Test Executor: Pavan						
Study ID: D652637189						
Module: check_ae_dates						
Generated On: 20260108_073033						
Test Name	Start Date	End Date	Expected	Actual	Status	
valid_dates	2023-01-01	2023-01-05	TRUE	TRUE	PASS	
invalid_dates	2023-01-10	2023-01-05	FALSE	FALSE	PASS	
equal_dates	2023-01-05	2023-01-05	FALSE	FALSE	PASS	
empty_start_date		2023-01-05	FALSE	FALSE	PASS	
empty_end_date	2023-01-01		FALSE	FALSE	PASS	
invalid_date_format	01-01-2023	05-01-2023	FALSE	FALSE	PASS	
future_dates	2025-12-31	2026-01-01	TRUE	TRUE	PASS	
start_after_end_on_same_month	2023-02-28	2023-02-01	FALSE	TRUE	FAIL	
Summary					PASS: 7	FAIL: 1



Key Takeaways

- **Test-Driven Development improves reliability and predictability of DQ rules**
- **Defining expected outcomes upfront reduces ambiguity and rework**
- **Early validation lowers operational and regulatory risk**
- **The framework is reusable, scalable, and future-ready`**



Game-Changer – How the Framework Integrates with AI-Enabled Tools

Structured inputs for AI:

Standardized rules, metadata, and test outcomes provide clean, machine-readable signals

AI-assisted prioritization:

Models can rank high-risk failures using historical trends and execution patterns

Explainable automation:

AI consumes traceable logs and test evidence without replacing rule-based logic

Contact Info:

- Pavan Kumar Reddy Sakam – [Senior Clinical Programmer, AstraZeneca]
- Email: pavankumar.sakam@astrazeneca.com
- LinkedIn: <https://www.linkedin.com/in/pavan-kumar-reddy-sakam-122634b4/>



- Sapan Soni – [Senior Clinical Database Developer, AstraZeneca]
- Email: sapan.soni@astrazeneca.com
- LinkedIn: <https://www.linkedin.com/in/sapan-soni-4b861a1a/>



**The Global Healthcare
Data Science Community**

Contact Channels

☎ UK +44 1843 609600
✉ office@phuse.global
🌐 phuse.global

Social Media

🐦 [@phusetwitta](https://twitter.com/phusetwitta)
📘 [/phusebook](https://facebook.com/phusebook)
▶ [/phusetube](https://youtube.com/phusetube)
🌐 [/company/phuse](https://company.phuse.com)