




ETLForge: A unified framework for synthetic test-data generation and ETL validation

Kyriakos Kartas ¹

¹ Independent Researcher

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: 

Submitted: 18 June 2025

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

Summary

ETLForge is a lightweight Python package (Python 3.9) that generates realistic synthetic test data *and* validates ETL outputs using the **same declarative schema**. One YAML/JSON file describes field types, ranges, uniqueness, nullability and optional Faker templates ([Faker Contributors, 2024](#)). The schema is consumed by two high-level components: DataGenerator (creates CSV/Excel datasets) and DataValidator (checks pipeline outputs, returning row-level error reports). A Click-based CLI ([The Pallets Projects, 2023](#)) mirrors the library API, enabling automation within CI/CD workflows ([Fowler & Foemmel, 2013](#)). ETLForge therefore removes duplicated specifications and closes gaps where data-quality regressions can slip through testing cycles.

Statement of need

Extract-Transform-Load (ETL) processes are critical for data-driven organizations, but testing these pipelines remains challenging ([Kimball & Ross, 2013](#); [Kleppmann, 2017](#)). Current testing approaches suffer from a fundamental disconnect: synthetic test data generation and output validation require separate tool chains with independent schema definitions. This creates several documented problems:

1. **Schema drift:** When generation schemas diverge from validation rules, tests may pass while production data fails validation ([Redman, 2016](#)).
2. **Maintenance overhead:** Duplicate schema definitions require synchronized updates, increasing development time and error potential ([Dasu & Johnson, 2003](#)).
3. **Testing gaps:** Inconsistent test data may not exercise edge cases that production validation catches, leading to false confidence ([Loshin, 2010](#)).

Existing libraries focus on either data generation (e.g., *Faker* ([Faker Contributors, 2024](#))) or validation (e.g., *Great Expectations* ([The Great Expectations Team, 2023](#)), *pandera* ([The pandera development team, 2023](#))). Because these tools are independent, engineers must maintain **parallel schemas**—one for generation, one for validation—leading to drift and missed bugs. ETLForge unifies both stages under a single source of truth, reducing maintenance effort and improving test robustness. Its small dependency footprint (six runtime packages) fits comfortably inside continuous-integration pipelines.

State of the field

The landscape of data generation and validation tools shows clear specialization but lacks integration:

Capability	ETLForge	Faker	Great Expectations	pandera	Cerberus
Schema-driven generation	Yes	Manual script-ing	No	No	No
Schema-driven validation	Yes	No	Yes	Yes	Yes
Single schema for both	Yes	No	No	No	No
CLI & Python API	Both	CLI only	Both	Python only	Python only
YAML/JSON schema support	Yes	No	Python/YAML	Python only	Python only
Lightweight dependencies	Yes (6 core)	Yes (1 core)	No (20+ deps)	Yes (5 core)	Yes (0 core)

Performance characteristics: ETLForge generates 10,000 rows in ~6 seconds and validates 100,000 rows in ~2 seconds on standard hardware, making it suitable for CI/CD integration where Great Expectations may be too heavyweight for simple validation tasks.

Limitations compared to existing tools: Unlike Great Expectations, ETLForge does not provide data profiling, drift detection, or advanced statistical validations. Unlike pandera, it lacks integration with type checkers and advanced pandas DataFrame validation. ETLForge prioritizes simplicity and schema consistency over advanced features.

To our knowledge, no existing open-source project provides an integrated, schema-first workflow covering both generation and validation with a unified configuration format.

Software description

ETLForge implements a dual-purpose architecture where a single YAML/JSON schema drives both data generation and validation processes. The schema format supports common data types (integer, float, string, date, category), constraints (ranges, uniqueness, nullability), and realistic data generation via Faker integration.

Core components: - DataGenerator: Creates synthetic datasets using pandas (McKinney & others, 2010) and numpy (Harris et al., 2020) for numerical operations - DataValidator: Validates CSV/Excel files against schema rules, returning detailed error reports - CLI interface: Enables command-line automation via Click (The Pallets Projects, 2023)

Integration approach: Rather than replacing existing tools, ETLForge complements ETL testing workflows by ensuring test data and validation rules remain synchronized. It integrates with pandas-based pipelines and exports to common formats (CSV, Excel via openpyxl).

Quality control

GitHub Actions run comprehensive checks on Python 3.9-3.11:

- **Unit tests** (pytest), achieving **77% line coverage** across 587 statements (pytest --cov)
 - CLI module: 67% coverage (31/95 statements missing)
 - Generator module: 74% coverage (64/244 statements missing)
 - Validator module: 86% coverage (33/241 statements missing)
- **Static analysis** (flake8, black --check, mypy) with type checking
- **Integration testing** via end-to-end example (example.py) validating complete workflows

- 67 ▪ **Performance benchmarks** tracking generation/validation speed regressions
- 68 All checks complete in under 90 seconds on Ubuntu runners, supporting rapid development
- 69 cycles.
- 70 **Known limitations:** - Large datasets (>1M rows) may require memory optimization - Complex
- 71 nested data structures are not supported - Advanced statistical validations require integration
- 72 with specialized tools

73 Availability

- 74 ▪ **Source code:** <https://github.com/kkartas/ETLForge> (MIT licence)
- 75 ▪ **Latest release:** v1.0.3 (PyPI: `pip install etl-forge`)
- 76 ▪ **Documentation:** <https://etlforge.readthedocs.io/>
- 77 ▪ **Platforms:** Linux, macOS, Windows; Python 3.9-3.11
- 78 ▪ **Installation:** `pip install etl-forge` (optional `etl-forge[faker]` for enhanced data
- 79 generation)

80 References

- 81 Dasu, T., & Johnson, T. (2003). *Exploratory data mining and data cleaning*. John Wiley &
- 82 Sons. <https://doi.org/10.1002/0471725527>
- 83 Faker Contributors. (2024). *Faker: A python package that generates fake data for you* (Version
- 84 24.4.0). <https://doi.org/10.5281/zenodo.3823398>
- 85 Fowler, M., & Foemmel, M. (2013). Continuous integration. *Thought-Works*. [https://](https://martinfowler.com/articles/continuousIntegration.html)
- 86 martinfowler.com/articles/continuousIntegration.html
- 87 Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D.,
- 88 Wieser, E., Taylor, J., Berg, S., Smith, N. J., & others. (2020). Array programming with
- 89 NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
- 90 Kimball, R., & Ross, M. (2013). *The data warehouse toolkit: The definitive guide to*
- 91 *dimensional modeling* (3rd ed.). John Wiley & Sons.
- 92 Kleppmann, M. (2017). *Designing data-intensive applications: The big ideas behind reliable,*
- 93 *scalable, and maintainable systems*. O'Reilly Media.
- 94 Loshin, D. (2010). *The practitioner's guide to data quality improvement*. Morgan Kaufmann.
- 95 McKinney, W., & others. (2010). Data structures for statistical computing in python.
- 96 *Proceedings of the 9th Python in Science Conference*, 445, 51–56. [https://doi.org/10.](https://doi.org/10.25080/Majora-92bf1922-00a)
- 97 [25080/Majora-92bf1922-00a](https://doi.org/10.25080/Majora-92bf1922-00a)
- 98 Redman, T. C. (2016). *Getting in front on data: Who does what*. Harvard Business Review
- 99 Press.
- 100 The Great Expectations Team. (2023). *Great Expectations: Always know what to expect from*
- 101 *your data* (Version 0.18.0). <https://doi.org/10.5281/zenodo.14976223>
- 102 The Pallets Projects. (2023). *Click: Command Line Interface Creation Kit* (Version 8.1.3).
- 103 <https://doi.org/10.5281/zenodo.5123732>
- 104 The pandera development team. (2023). *pandera: Data validation for scientists, engineers,*
- 105 *and analysts* (Version 0.17.0). <https://doi.org/10.25080/gerudo-f2bc6f59-010>