International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

ReFlow: Agentic AI-Based Self-Healing Data Pipeline for Data Quality Monitoring and Automated Recovery

Ganesh Roddanavar[1], Niveditha N M[2], DR. Neelambike S [3]
[1] PG Student, Department of Data Engineering, Faculty of Engineering and Technology, GM School of Advanced Studies, GM University, Davanagere
[2] Assistant Professor, Department of Information Science and Engineering, GM University, Davanagere
[3] Associate Professor and Head Department of Information Science and Engineering, GM University, Davanagere

---

Abstract
Data-quality errors and processing failures can interrupt data pipelines and increase the effort required for manual troubleshooting. This paper proposes an agentic AI-based self-healing framework for monitoring structured batch data and coordinating controlled recovery. The framework combines rule-based validation with large language model assistance to interpret detected issues and recommend corrective actions. Python and pandas support data processing, while Gemini assists diagnosis using relevant schema information and error logs. A recovery controller checks each recommendation, executes an approved repair function, and revalidates the output before allowing processing to continue. Original inputs and incident histories are retained for traceability, and ambiguous or unresolved cases are referred for human review. A Streamlit dashboard displays validation findings, recovery attempts, and before-and-after data comparisons. The proposed evaluation introduces known defects into synthetic datasets and compares the framework with rule-based cleaning and retry-only recovery on their applicable tasks. Evaluation measures include detection performance, correction precision, repair coverage, recovery success, processing time, and unintended changes. Comparative experiments remain pending; the paper presents the proposed framework and evaluation design without claiming measured improvements.

Key Words: Agentic AI, Data Quality, Self-Healing Pipelines, Automated Recovery, Large Language Models, Data Engineering.

1. INTRODUCTION

Data engineering provides the processes through which raw data becomes available for reporting, analysis, and application development. These processes depend on pipelines that collect, validate, transform, and store information. The usefulness of the final output depends on both successful execution and the quality of the data being processed. A pipeline may complete without reporting an error while still producing incomplete, inconsistent, or misleading records. Therefore, pipeline reliability requires checks on data correctness as well as monitoring of execution status.

Common problems in structured datasets include missing values, duplicate records, incompatible data types, and inconsistent date formats. Processing can also be interrupted by temporary service failures or configuration issues. Conventional pipelines often address these situations through predefined validation rules, scheduled retries, and manual troubleshooting. These mechanisms remain useful, but each has practical limits. Repeating a failed task does not necessarily correct an invalid input, while a fixed cleaning rule may be unsuitable when the meaning of a value is uncertain. Manual investigation requires an engineer to examine logs, identify the cause, select a correction, and verify its effect.

This project proposes an **Agentic AI-Based Self-Healing Data Pipeline for Data Quality Monitoring and Automated Recovery**. The proposed system combines deterministic validation with AI-assisted diagnosis and controlled corrective actions. Its operating cycle consists of observing a detected issue, interpreting the available evidence, selecting an approved action, applying the action, and evaluating the resulting data. Recovery is accepted only when the required checks succeed. Cases that cannot be resolved within the supported rules are recorded for human review.

The initial scope focuses on structured batch data supplied through comma-separated values (CSV) files. Python and pandas support data loading, validation, and transformation, while Gemini assists in interpreting relevant schema information and error descriptions. A recovery controller checks the proposed action and its parameters before invoking an approved repair function. Supported operations may include compatible type conversion, unambiguous date normalization, and removal of confirmed duplicate records. Missing or uncertain values require an explicit handling policy rather than an assumed replacement.

1.1 Background /History

Data engineering developed from the need to collect information from different sources and prepare it for storage, analysis, and decision-making. Early data-processing systems relied heavily on manually written scripts and scheduled batch jobs. These systems performed predefined operations, with engineers responsible for investigating errors, correcting records, and restarting failed processes.

The adoption of data warehouses established Extract, Transform, and Load (ETL) as a common approach to data integration. ETL pipelines extracted data from operational sources, transformed it according to business requirements, and loaded it into a target repository. Validation rules helped identify problems such as missing fields, invalid values, and duplicate records. However, handling unexpected input changes often required modifications to transformation scripts and manual intervention.

Research subsequently explored machine learning for identifying anomalies, estimating missing values, and assisting with data cleaning. Approaches such as CSDI and SAITS demonstrate methods for imputing missing time-series values, while Anomaly Transformer investigates detecting unusual temporal patterns. These studies provide techniques that can support individual stages of a data-quality workflow. CSDI, SAITS, Anomaly Transformer

More recently, large language models have been investigated for data-management tasks and tool-assisted problem-solving. Research on foundation models examines their application to cleaning and integration, while ReAct explores combining reasoning with actions and observations. These approaches motivate using AI to interpret pipeline errors and coordinate diagnostic tools, with validation required before accepting any proposed correction. Foundation Models for Data Tasks, ReAct The project's base paper, *"Beyond ETL: How AI Agents Are Building Self-Healing Data Pipelines,"* discusses extending pipeline automation through agents that monitor operations and coordinate corrective actions. Building on this direction, the proposed project will integrate rule-based checks, AI-assisted diagnosis, controlled repairs, and outcome verification into a prototype for structured batch data. Its purpose is to investigate recovery from selected failures while preserving original data, recording changes, and referring uncertain cases for review. Base Paper

1.2 Existing Technology

Existing data-engineering systems use several technologies to collect, transform, validate, and monitor data. These technologies provide the foundation for the proposed self-healing pipeline.

1. **ETL and ELT Systems** Extract, Transform, and Load (ETL) systems prepare data before loading it into a destination. Extract, Load, and Transform (ELT) systems perform transformations after loading. Both approaches use predefined processing logic, which may require modification when source formats or business requirements change.
2. **Workflow Orchestration** Orchestration systems schedule tasks, manage dependencies, track execution status, and retry failed operations. They help coordinate complex pipelines. However, retrying a task cannot resolve a persistent input error unless its underlying cause is corrected.
3. **Rule-Based Data Validation** Validation systems check datasets against specified requirements, including mandatory fields, permitted values, expected data types, uniqueness, and relationships between columns. These checks reliably identify known violations, but engineers must define and maintain the rules.
4. **Data Cleaning and Transformation** Cleaning functions standardize formats, convert compatible data types, identify duplicates, and handle missing values. Their effectiveness depends on appropriate policies. For example, replacing a missing amount with zero may satisfy a numeric constraint while incorrectly changing its business meaning.
5. **Database Integrity Controls** Database constraints, such as primary keys, foreign keys, uniqueness checks, and non-null requirements, prevent certain invalid records from entering storage. Transactions support consistent updates. These mechanisms enforce defined conditions but do not independently determine how rejected records should be repaired.
6. **Monitoring and Logging** Monitoring systems track execution time, task failures, resource usage, and processing volumes. Logs provide evidence for investigating incidents. Although these mechanisms improve visibility, engineers often need to interpret the information and select corrective actions.
7. **Machine-Learning-Based Data Quality Methods** Machine-learning approaches can identify unusual patterns and estimate missing values. For example, Anomaly Transformer studies time-series anomaly detection, while SAITS addresses missing-value imputation. These methods support specific quality tasks and require suitable data and evaluation before integration into a pipeline. Anomaly Transformer, SAITS
8. **Large Language Models and Agent Frameworks** Language models can assist with interpreting errors, generating transformation suggestions, and selecting tools. ReAct investigates reasoning combined with external actions, while AutoGen supports collaboration among agents. Applying these approaches to data pipelines requires validating suggested actions against actual schemas, rules, and execution results. ReAct, AutoGen

The proposed project will integrate these capabilities into a coordinated process for **detection, diagnosis, correction, verification, and recovery**. Its focus is to evaluate whether AI-assisted decisions, combined with predefined repair functions and validation checks, can reduce manual intervention for selected pipeline failures.

2. LITERATURE SURVEY

The literature relevant to the proposed ReFlow platform covers data-quality management, automated cleaning, missing-value estimation, language-model assistance, and agent-based decision-making. The survey examines data quality, anomaly detection, missing-value handling, autonomous agents, tool use, and automated recovery. The project applications discussed below are proposed adaptations of the reviewed work.

[1]. **Chakraborty (2025)—"Beyond ETL: How AI Agents Are Building Self-Healing Data Pipelines."** This main/base paper discusses the transition from conventional ETL workflows to agent-driven pipelines capable of monitoring operations, identifying problems, and coordinating corrective actions. It provides the conceptual foundation for the proposed project. The project will translate these concepts into an implementation evaluated through data-quality improvements and recovery outcomes. Source

[2]. **Sambasivan et al. (2021)—"'Everyone Wants to Do the Model Work, Not the Data Work': Data Cascades in High-Stakes AI."** This study investigates how neglected data problems propagate through AI development and affect downstream systems. It highlights the importance of data practices, documentation, and organizational responsibility. For the proposed platform, it motivates early validation, incident tracking, and traceable corrections; its focus is understanding data problems rather than implementing automated repair. Source

[3]. **Tashiro et al. (2021)—"CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation."** CSDI uses conditional diffusion models to estimate missing time-series values while representing uncertainty. It offers an advanced alternative to simple mean or constant-value imputation. Its relevance is strongest for pipelines processing temporal data, although applying it requires suitable training data and additional computation. Source

[4]. **Xu et al. (2022)—"Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy."** The authors introduce a Transformer-based approach that identifies abnormal time-series observations through differences in their association patterns. The work supports intelligent monitoring of pipeline metrics such as processing time, throughput, and record counts. An additional diagnosis and repair mechanism would be necessary to convert detected anomalies into recovery actions. Source

[5]. **Narayan et al. (2022)—"Can Foundation Models Wrangle Your Data?"** This paper investigates foundation models for data-management tasks, including error detection, imputation, schema matching, and entity matching. It demonstrates the potential of natural-language prompting for tasks traditionally requiring specialized systems. Its observations about prompt sensitivity and domain-specific data motivate validating all model-generated corrections before accepting them. Source

Study	Main Focus	Project Relevance
Chakraborty (2025)	Self-healing pipelines	Base for agent-led recovery
Narayan et al. (2022)	Foundation models for data	AI-assisted cleaning tasks
Yao et al. (2023)	ReAct reasoning and actions	Tool-based diagnosis
Shinn et al. (2023)	Reflexion feedback	Learn from incident outcomes
Gou et al. (2024)	CRITIC tool verification	Check proposed repairs

Table 1: Representative literature and design implications

3. PROBLEM STATEMENT

Data engineering pipelines depend on consistent and valid input data to produce reliable outputs. However, datasets may contain missing values, duplicate records, incorrect data types, inconsistent formats, and violations of business rules. Unexpected schema changes and database-loading failures can further disrupt processing. These problems may result in incomplete outputs, inaccurate analysis, repeated execution, and additional maintenance effort. Conventional approaches commonly use predefined validation rules, manually written cleaning scripts, and automatic retries. Although these techniques handle known conditions, validation rules require updates when data structures or business requirements change. Retrying a failed task cannot correct malformed records or resolve persistent configuration problems. Engineers must therefore inspect failures, determine suitable corrective actions, and verify the resulting data.

Large language models can assist in interpreting dataset structures and suggesting validation or repair actions. However, their suggestions may be incorrect or unsupported by the available evidence. Applying such changes without verification can introduce new errors, while removing problematic records may unnecessarily discard useful information. The problem addressed by this project is the need for a controlled and explainable workflow that integrates data-quality detection, repair planning, human review, verification, and recovery from supported pipeline failures.

The key problems addressed by this project are:

1. **Manual Troubleshooting:** Engineers must inspect datasets and logs to identify failure causes, increasing maintenance effort and recovery time.
2. **Limited Automatic Recovery:** Retrying a failed task does not resolve persistent problems such as invalid values or incompatible data types.
3. **Dependence on Predefined Rules:** Existing correction logic may not adequately handle unfamiliar errors or changes in input structure.
4. **Disconnected Processing Stages:** Detection, diagnosis, correction, and recovery may require separate tools and manual coordination.
5. **Risk of Incorrect Corrections:** Automated changes may satisfy technical constraints while altering the intended meaning of data.
6. **Insufficient Traceability:** Without detailed incident records, it is difficult to explain changes, verify outcomes, or investigate recurring failures.

Therefore, the project aims to develop an **agentic AI-based self-healing data pipeline** that combines rule-based validation, AI-assisted diagnosis, controlled corrective actions, and repair verification. The prototype will address selected data-quality issues and batch-processing failures, preserve original inputs, and flag uncertain cases for user review.

3.1 Scope of the Project

The project focuses on developing a prototype of an agentic AI-based self-healing data pipeline for identifying data-quality issues and recovering from selected pipeline failures. Its scope includes:

1. **Data Ingestion:** Support uploading and processing structured CSV datasets for validation, cleaning, and loading.
2. **Data Quality Monitoring:** Detect missing values, duplicate records, inconsistent date formats, incorrect data types, and violations of predefined validation rules.
3. **AI-Assisted Diagnosis:** Use a large language model to interpret detected issues and pipeline error messages and suggest suitable corrective actions.
4. **Automated Data Correction:** Apply supported corrections, such as removing confirmed duplicates, standardizing formats, converting compatible data types, and handling missing values according to configured policies.
5. **Pipeline Failure Recovery:** Support controlled retries and re-execution of eligible failed processing steps after the underlying issue has been addressed.
6. **Repair Validation:** Recheck corrected data against specified rules before accepting changes. Flag uncertain or unresolved cases for user review.
7. **Interactive Dashboard:** Provide a Streamlit interface for dataset uploads, pipeline execution, monitoring results, and viewing incident details.
8. **Before-and-After Analysis:** Compare original and processed datasets using data-quality indicators, record counts, and visual summaries.
9. **Logging and Traceability:** Record detected issues, attempted corrections, validation results, and recovery outcomes while retaining the original input for comparison.
10. **Performance Evaluation:** Evaluate correction accuracy, recovery success rate, processing time, and unresolved incidents using sample datasets with deliberately introduced errors.

The prototype is limited to structured batch data and predefined failure scenarios. Large-scale distributed processing,

continuous streaming, and fully autonomous correction of unfamiliar business rules are outside the current scope.

3.2 Objectives

1. **To develop an automated data-quality monitoring system.** The system will examine uploaded CSV datasets to detect missing values, duplicate records, inconsistent date formats, incompatible data types, and violations of predefined validation rules. It will summarize these issues to help users understand the condition of their data before processing.
2. **To integrate agentic AI for error diagnosis and corrective recommendations.** The platform will use a large language model to interpret validation results and pipeline error logs alongside available schema information. The AI component will assist in identifying possible causes and recommending suitable actions, while predefined controls will determine which actions are permitted.
3. **To implement controlled data correction and pipeline recovery.** The system will apply supported repair functions, such as standardizing formats, converting compatible data types, handling duplicates, and treating missing values according to configured policies. After correcting an issue, it will retry eligible failed operations within a specified attempt limit and flag failures that remain unresolved.
4. **To ensure repair verification and traceability.** Corrected datasets will be rechecked against validation rules before changes are accepted. The platform will preserve original inputs and record detected issues, attempted corrections, and outcomes. Uncertain cases will be referred for user review to reduce the risk of accepting inappropriate changes.
5. **To evaluate performance and present measurable outcomes.** The prototype will be tested using datasets containing deliberately introduced errors and compared with rule-based cleaning and retry-only approaches. Evaluation will measure correction accuracy, recovery success rate, processing time, and unresolved incidents. A Streamlit dashboard will display these results and before-and-after data-quality comparisons.

4. IMPLEMENTATION

The proposed ReFlow platform is designed as a modular Python application for monitoring data quality and coordinating recovery from supported pipeline failures. It combines pandas-based processing, Gemini-assisted diagnosis, controlled repair functions, and a Streamlit interface.



Fig 4.1 Implementation Stages

The following describes the intended implementation; verification of the complete system and comparative experimental evaluation remain pending.

4.1 Data Ingestion

The ingestion module accepts a CSV file through the Streamlit interface and loads it into a pandas DataFrame. For the Amazon dataset, relevant fields include product identifiers, product names, categories, prices, discount percentages, ratings, rating counts, and review information. Initial checks examine file readability and the presence of required columns. The original dataset is retained, while a separate working copy is prepared for subsequent processing.

4.2 Data Validation

The validation module checks records against a configurable data contract that specifies expected types, required fields, acceptable ranges, and relationships between columns. Price fields are checked for valid numeric representations, discount percentages for permitted bounds, and ratings against the configured rating scale. Each violation is recorded with its field, affected row, and rule description.

Repeated product identifiers are treated as duplicate candidates because multiple records may contain different review information. Records are removed only when the configured duplicate policy confirms that they represent unnecessary repetition.

4.3 Gemini-Assisted Diagnosis

The diagnostic module prepares a concise description of the detected issue using the relevant schema, validation results, and sanitized error messages. Gemini is asked to recommend an action from a predefined repair catalogue and provide a brief explanation. Its response is checked for the expected structure, recognized action names, and valid parameters.

The API key is supplied through an environment variable. If the model request fails or produces an unsupported recommendation, deterministic checks remain available and the incident is recorded for review. Model output does not directly authorize changes to the dataset.

4.4 Controlled Data Repair

Approved repair functions perform operations such as removing currency symbols from price representations, parsing numeric text, and normalizing percentage values. For example, a price represented as "₹399" can be converted to the numeric value 399 when the field's currency and interpretation are established. Similarly, "24,269" can be parsed as an integer rating count under the expected number-format convention.

Missing or ambiguous values follow explicit handling policies. An unknown rating is retained for review rather than assigned an arbitrary value. Every repair records the original value, proposed replacement, selected function, and reason for the change.

4.5 Revalidation and Recovery

After a repair, the validation module checks the modified data again. Acceptance requires the corrected records to satisfy applicable constraints without introducing new violations. Unsuccessful changes are rejected or referred for review, and the original input remains available for comparison.

Eligible temporary execution failures use a bounded retry mechanism. Persistent input errors require correction before re-execution. Where database loading is included, transactions and duplicate-write controls are required to prevent incomplete or repeated loads.

4.6 Dashboard and Incident Reporting

The Streamlit interface provides dataset upload, validation summaries, repair review, and before-and-after comparisons. Users can inspect detected issues, attempted corrections, accepted changes, and unresolved cases. Download options provide access to the processed dataset and incident report.

Operational counts displayed by the dashboard describe the current run. Measures such as repair correctness and recovery success require separate evaluation against known expected outcomes.

4.7 Testing and Evaluation

The evaluation will combine inspection of the Amazon dataset with controlled test cases containing deliberately introduced defects. These cases establish known reference outcomes for assessing detection and repair behaviour. Rule-based and AI-assisted configurations will use the same validation rules and repair functions to support a fair comparison.

Evaluation will examine detection precision and recall, correction precision, repair coverage, recovery success, processing time, and unintended changes. Results will be reported after execution and verification of the experiments.

5. SYSTEM ARCHITECTURE

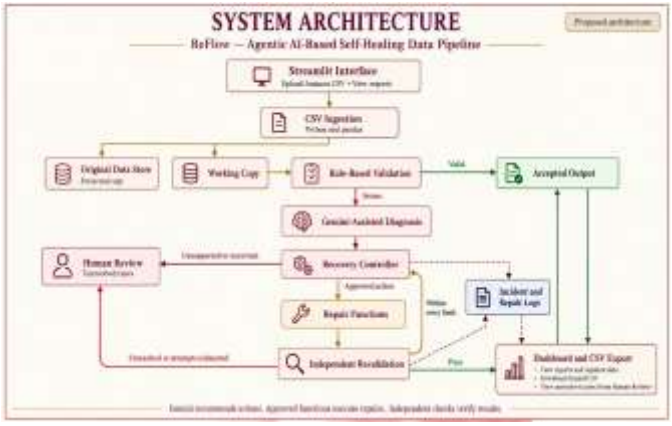


Fig 5.1 System Architecture

The proposed ReFlow system checks data quality and helps recover from pipeline errors. Users upload an Amazon CSV file through a Streamlit interface. Python and pandas read the file, preserve the original data, and create a working copy.

The validation module checks missing values, incorrect formats, invalid prices, ratings, and possible duplicates. When problems are found, Gemini examines the relevant error details and suggests an approved correction.

The recovery controller checks each suggestion before allowing a repair function to change the data. After correction, the system validates the records again. Accepted records become available for download. Uncertain or unresolved problems are sent for human review.

The system records detected issues, attempted repairs, and outcomes in an incident log. The dashboard displays these details and compares the original and processed datasets. This design aims to reduce manual work while keeping changes controlled and traceable. Its effectiveness still requires testing against known outcomes.

6. EXISTING MODEL

The conventional model considered in this project uses predefined rules and manual troubleshooting to maintain data pipelines. Data passes through extraction, transformation, and loading stages. Validation rules identify problems such as missing values, duplicate records, incorrect data types, and inconsistent formats.

When a problem occurs, the system records an error or alerts the user. Engineers inspect the dataset and logs, determine the cause, and modify the data or processing logic. Automatic retries can handle temporary interruptions, but they cannot correct persistent input errors.

For the Amazon dataset, cleaning scripts can convert currency values, standardize numeric fields, and identify duplicate candidates. However, developers must define the correct treatment for each issue and update rules when requirements change.

This approach provides predictable handling of known problems. Its limitations include manual investigation, maintenance of cleaning rules, and limited coordination between detection and recovery. It serves as the rule-based baseline for evaluating the proposed ReFlow system.

7. PROPOSED METHODOLOGY

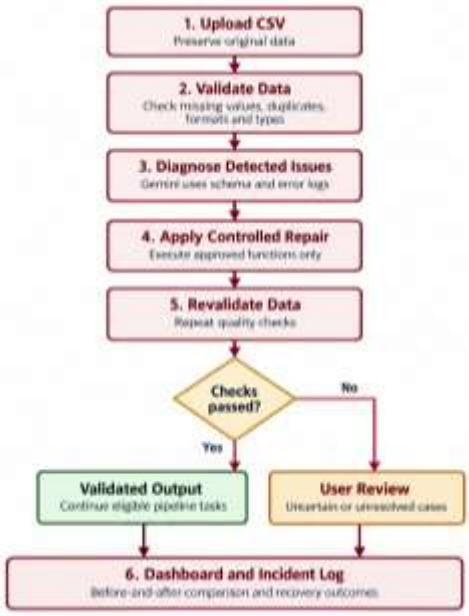


Fig 7.1 Proposed Methodology

The proposed ReFlow model combines rule-based validation with agentic AI to identify data-quality problems and coordinate controlled recovery. Users upload an Amazon CSV dataset through a Streamlit interface. Python and pandas load the records, preserve the original data, and check missing values, numeric formats, prices, ratings, and duplicate candidates.

When an issue is detected, Gemini examines the relevant schema and error details to recommend a permitted correction. A recovery controller verifies the recommendation before executing an approved repair function. The corrected records are validated again before acceptance.

Temporary processing failures are retried within a defined limit. Ambiguous values, unsupported recommendations, and unresolved problems are referred for human review. Every attempted correction and its outcome are recorded in an incident log.

The dashboard presents validation findings, repair histories, and before-and-after comparisons. This model aims to reduce manual troubleshooting while preserving traceability and control. Its effectiveness will be evaluated against a rule-based baseline.

8. RESULT



Fig 8.1 Self-Healing Dashboard

The ReFlow dashboard provides an interface for uploading datasets, selecting a diagnosis mode, and reviewing proposed validation rules. It summarizes column completeness and numeric compatibility, alongside suggested constraints and their explanations. The displayed data contract supports inspection before corrective actions are applied. This screenshot demonstrates data profiling and rule configuration; it does not show completed repairs or recovery results.

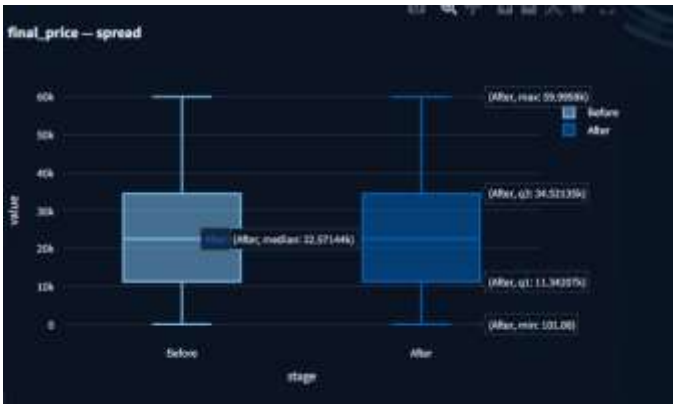


Fig 8.2 Before and After Final price stage

The final-price box plots show visually similar distributions before and after processing. The processed data has a median of approximately 22,571.44, with the middle 50% of values between 11,342.07 and 34,521.35. This indicates no substantial visible change in the overall price distribution. However, similar distributions alone do not confirm that individual errors were corrected; record-level comparisons and repair logs are required to assess correction accuracy.



Fig 8.3 Monthly Mean Graph

Shows monthly average values from roughly 2018 to 2023. Values fluctuate around **30,000**. The before-and-after lines appear to overlap, indicating little visible change in the monthly averages. The chart does not identify which field is being averaged.

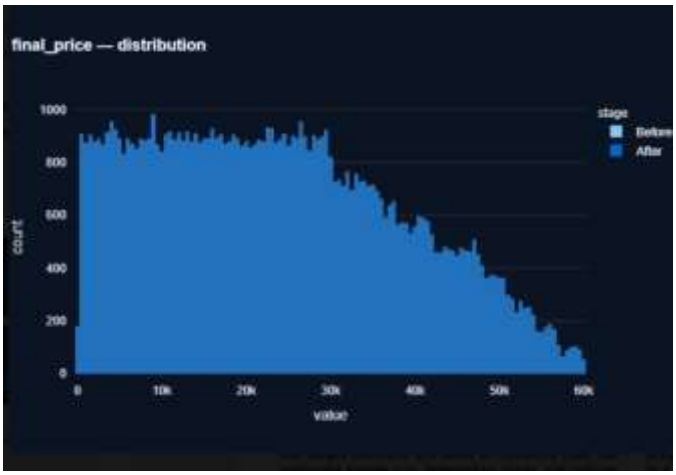


Fig 8.4 Final price Distribution Graph

Shows how frequently different final prices occur. Prices below approximately 30,000 occur more frequently, while counts generally decline toward 60,000. The two distributions largely overlap.

9. CONCLUSIONS

This paper presents ReFlow, a proposed agentic AI-based self-healing data pipeline for data-quality monitoring and controlled recovery. The framework combines rule-based validation, Gemini-assisted diagnosis, approved repair functions, and independent revalidation. It focuses on structured Amazon product data, including missing values, inconsistent numeric formats, invalid prices, ratings, and duplicate candidates.

The design preserves original records and maintains an incident history to make corrections traceable. Uncertain recommendations and unresolved issues are referred for human review. A Streamlit interface supports data upload, inspection of repairs, and comparison of original and processed datasets.

The main contribution is a coordinated workflow that connects error detection with diagnosis, correction, and verification. Comparative evaluation remains pending; therefore, improvements in repair correctness, recovery time, or manual effort cannot yet be confirmed. Future work will evaluate these outcomes against a rule-based baseline and explore additional data sources, database integration, and streaming support.

ACKNOWLEDGEMENT

The author expresses sincere gratitude to the project guide, Mrs. Niveditha N M, for guidance and constructive feedback during the development of this work. The author also thanks the faculty and staff of GM University, Davangere, for their academic support. The contributions of the researchers and open-source developers whose work informed the proposed self-healing data pipeline are gratefully acknowledged.

REFERENCES

A. Journal Articles

[1] S. Chakraborty, “Beyond ETL: How AI Agents Are Building Self-Healing Data Pipelines,” *Journal of Computer Science and Technology Studies*, vol. 7, no. 3, pp. 741–756, 2025. **Main/base paper.** DOI

[2] W. Du, D. Cote, and Y. Liu, “SAITS: Self-Attention-based Imputation for Time Series,” *Expert Systems with Applications*, vol. 219, article 119619, 2023. DOI

[3] L. Wang *et al.*, "A Survey on Large Language Model Based Autonomous Agents," *Frontiers of Computer Science*, vol. 18, article 186345, 2024. DOI
[4] A. Narayan, I. Chami, L. Orr, and C. Ré, "Can Foundation Models Wrangle Your Data?" *Proceedings of the VLDB Endowment*, vol. 16, no. 4, pp. 738–746, 2022. DOI
[5] N. F. Liu *et al.*, "Lost in the Middle: How Language Models Use Long Contexts," *Transactions of the Association for Computational Linguistics*, vol. 12, 2024. Paper

B. Conference Proceedings

[6] N. Sambasivan *et al.*, "'Everyone Wants to Do the Model Work, Not the Data Work': Data Cascades in High-Stakes AI," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2021. DOI
[7] Y. Tashiro, J. Song, Y. Song, and S. Ermon, "CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Paper
[8] J. Xu, H. Wu, J. Wang, and M. Long, "Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy," in *International Conference on Learning Representations (ICLR)*, 2022. Paper
[9] J. Wei *et al.*, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. Paper
[10] S. Yao *et al.*, "ReAct: Synergizing Reasoning and Acting in Language Models," in *International Conference on Learning Representations (ICLR)*, 2023. Paper
[11] N. Shinn *et al.*, "Reflexion: Language Agents with Verbal Reinforcement Learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. Paper
[12] A. Madaan *et al.*, "Self-Refine: Iterative Refinement with Self-Feedback," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. Paper
[13] T. Schick *et al.*, "Toolformer: Language Models Can Teach Themselves to Use Tools," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. Paper
[14] X. Wang *et al.*, "Self-Consistency Improves Chain of Thought Reasoning in Language Models," in *International Conference on Learning Representations (ICLR)*, 2023. Paper
[15] S. Min *et al.*, "FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. Paper
[16] Y. Shen *et al.*, "HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. Paper
[17] L. Gao *et al.*, "RARR: Researching and Revising What Language Models Say, Using Language Models," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023. Paper
[18] Q. Wu *et al.*, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation," in *Conference on Language Modeling (COLM)*, 2024. Paper
[19] X. Liu *et al.*, "AgentBench: Evaluating LLMs as Agents," in *International Conference on Learning Representations (ICLR)*, 2024. Paper
[20] T. Nguyen *et al.*, "CulturaX: A Cleaned, Enormous, and Multilingual Dataset for Large Language Models in 167 Languages," in *Proceedings of the Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING)*, 2024. Paper
[21] S. Hong *et al.*, "MetaGPT: Meta Programming for a Multi-Agent Collaborative Framework," in *International Conference on Learning Representations (ICLR)*, 2024. Paper
[22] C. E. Jimenez *et al.*, "SWE-bench: Can Language Models Resolve Real-World GitHub Issues?" in *International Conference on Learning Representations (ICLR)*, 2024. Paper
[23] O. Khattab *et al.*, "DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines," in *International Conference on Learning Representations (ICLR)*, 2024. Paper
[24] Z. Gou *et al.*, "CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing," in *International Conference on Learning Representations (ICLR)*, 2024. Paper
[25] S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez, "Gorilla: Large Language Model Connected with Massive APIs," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2024. Paper
[26] Y. Qin *et al.*, "ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs," in *International Conference on Learning Representations (ICLR)*, 2024. Paper
[27] J. Yang *et al.*, "SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. Paper
[28] X. Shen *et al.*, "AutoClean: LLMs Can Prepare Their Training Corpus," in *Proceedings of NAACL: Human Language Technologies—System Demonstrations*, pp. 85–95, 2025. Paper

C. Books

[29] J. Reis and M. Housley, *Fundamentals of Data Engineering: Plan and Build Robust Data Systems*. O'Reilly Media, 2022. Publisher
[30] C. Huyen, *Designing Machine Learning Systems: An Iterative Process for Production-Ready Applications*. O'Reilly Media, 2022. Publisher