Insight Forge: An Ai-Powered Platform for Automated Data Cleaning, Analysis, And Interactive Visualization

Om Sangar[1], Prathamesh Mankar[2], Alfaz Tamboli[3], Sahil Shaikh[4], Dr. Shuddhodhan Bokefode[5]
[1,2,3,4]*Department of Computer Science and Engineering, MIT College of Railway Engineering and Research, Barshi, Maharashtra, India*
[5]*Professor, Department of Computer Science and Engineering, MIT College of Railway Engineering and Research, Barshi, Maharashtra, India*

Abstract—**Data preparation is a critical prerequisite for reliable analytics, yet raw tabular datasets frequently contain missing entries, statistical outliers, inconsistent data types, and structural ambiguities that hinder non-technical users. This paper presents Insight Forge an AI-powered analytics platform that unifies automated data cleaning, tabular aggregation, and interactive visualization within a single lightweight workflow. Built using Python, Pandas, NumPy, FastAPI, Streamlit, and Plotly, the system targets students, small businesses, and entry-level analysts who require structured data exploration without chaining multiple separate tools. Its cleaning module supports mean, median, and mode imputation alongside interquartile range (IQR) and Z-score outlier flagging, paired with an inspectable cleaning log that tracks applied transformations. The analytical layer supports grouped aggregations, while the dashboard renders key performance indicator (KPI) cards, interactive Plotly charts, dynamic filters, rearrangeable layouts, and data summaries. A conversational assistant, Forge Assistant, integrates the Google Gemini API to provide guided natural-language support. The end-to-end workflow has been exercised using a sales dataset of approximately 10,000 rows and 10 columns; however, formal quantitative benchmarking remains in progress. Accordingly, this paper documents the implemented system architecture, safety and transparency safeguards, and a structured Phase 2 evaluation protocol rather than reporting unmeasured speed, accuracy, or usability scores. Future work includes executing controlled cleaning and aggregation benchmarks, completing PostgreSQL production integration, adding reversible transformation pipelines, and conducting structured user evaluation.**

Index Terms—**Data cleaning, data analytics, artificial intelligence, outlier detection, interactive visualization, large language models, exploratory data analysis.**

I. INTRODUCTION

Data-driven decision-making relies heavily on the completeness, consistency, and interpretability of underlying datasets [1], [3]. In practical operational settings, raw spreadsheets and transactional exports commonly suffer from blank cells, extreme numeric anomalies, formatting inconsistencies, duplicated records, and redundant columns. If left unaddressed, these quality issues distort summary statistics, mislead downstream visualizations, and reduce confidence in business reporting. Consequently, data preparation represents one of the most time-consuming stages of analytics, often requiring users to switch manually across spreadsheet editors, scripting environments, relational databases, and business-intelligence dashboards [1].

Recent literature highlights the need for integrated workflows that pair deterministic, inspectable statistical operations with intelligent automation [2], [9]. While commercial business-intelligence tools offer rich enterprise capabilities, they often impose steep learning curves or licensing barriers for students and small enterprises. Conversely, writing ad-hoc Python scripts provides flexibility but excludes non-technical users. To bridge this gap, this paper presents Insight Forge, an integrated web-based analytics platform designed around a lightweight upload-to-dashboard workflow. The system combines rule-based statistical data cleaning, tabular aggregation, interactive Plotly visualization, and a conversational guide—Forge Assistant—powered by the Google Gemini API [13].

The paper makes three practical contributions: (1) it documents a unified end-to-end architecture connecting dataset ingestion, cleaning, aggregation, dashboarding, and conversational assistance; (2) it details transparent design safeguards, including inspectable imputation, non-destructive outlier flagging, and an auditable cleaning-log structure; and (3) it establishes a reproducible evaluation protocol for validating cleaning correctness, aggregation accuracy, execution latency, and LLM response grounding without inventing unmeasured performance claims.

II. RELATED WORK

Hameed and Naumann surveyed commercial data-preparation tools and emphasized that data wrangling remains a foundational bottleneck across data-management pipelines, noting continuing opportunities for more automated and user-accessible workflows [1]. Chu et al. similarly provided a taxonomy of data-cleaning challenges, framing error detection and data repair as coupled processes whose outcomes directly affect downstream statistical estimation [3]. Zhu et al. reviewed relational data cleaning in the era of artificial intelligence, examining how traditional integrity constraints and statistical rules complement AI-driven detection and repair [2]. Together, these studies motivate hybrid platforms where automated steps remain transparent and reviewable by the user.

Handling missing data and anomalous observations requires careful methodological grounding because blind automation can introduce bias. Emmanuel et al. surveyed missing-data mechanisms and imputation techniques, noting that baseline statistical methods such as mean, median, and mode imputation remain widely used due to their interpretability, though mean imputation is sensitive to skewed distributions and systematic missingness can distort variance [4]. For anomaly detection, Chandola et al. categorized statistical, proximity-based, and model-based approaches, emphasizing that statistical thresholds rely on underlying distributional assumptions [5]. Rousseeuw and Hubert further demonstrated why classical mean and standard-deviation rules (such as Z-scores) can be masked by extreme values themselves, whereas quartile-based rules such as Tukey's IQR fences offer greater robustness for skewed numerical data [6]. Distinguishing statistical anomaly flagging from automatic deletion is therefore essential in general-purpose tools.

In the domain of visual analytics, Wang et al. examined the convergence of visualization and artificial intelligence (VIS+AI), arguing that AI can assist users in selecting visual encodings, summarizing patterns, and interpreting analytical views [9]. Shen et al. surveyed visualization-oriented natural language interfaces (V-NLIs), organizing them across query interpretation, data transformation, visual mapping, and dialogue management [10]. Recent LLM-driven prototypes such as Chat2VIS [11] and LIDA [12] demonstrate how large language models can translate natural-language goals into visualization code and chart recommendations, while also underscoring challenges around hallucinated metrics and code-execution safety. Unlike standalone chart generators or pure data-cleaning algorithms, Insight Forge integrates statistical preparation, tabular aggregation, interactive Plotly dashboards, and Gemini-powered conversational assistance [13] into one human-in-the-loop workflow where plotted totals remain anchored to deterministic Pandas [7] and NumPy [8] computations.

III. PROBLEM STATEMENT AND OBJECTIVES

Non-technical analysts, students, and small businesses face substantial friction when moving from raw, imperfect tabular files to reliable visual summaries. Existing workflows frequently force users to combine separate tools for missing-value repair, outlier inspection, pivot aggregation, and interactive charting, while providing little visibility into which transformations altered the original records. The engineering problem addressed in this work is the design of an accessible, unified platform that streamlines dataset preparation and exploration while keeping data transformations inspectable.

The primary objectives of the project are: (a) to provide a unified web interface for dataset upload, schema inspection, cleaning, tabular aggregation, and visualization; (b) to implement statistical missing-value imputation (mean, median, and mode) and statistical outlier identification (IQR and Z-score); (c) to maintain a cleaning log recording applied operations for auditability; (d) to generate

summarized analytical tables and interactive dashboards featuring KPI cards, filters, chart selection, and rearrangeable layouts; (e) to integrate Forge Assistant using the Google Gemini API for natural-language analytical guidance; and (f) to establish and execute a rigorous evaluation plan measuring functional correctness, aggregation integrity, runtime scalability, and AI response accuracy.

IV. SYSTEM ARCHITECTURE

The platform follows a modular, layered architecture separating presentation, API orchestration, numerical data processing, AI assistance, and persistence. Streamlit powers the interactive frontend, organizing analytical modules within a left-side navigation panel alongside an on-demand, slide-in Forge Assistant panel. FastAPI provides structured backend API routing and validation services where applicable. Core tabular manipulations and statistical calculations are executed using Pandas [7] and NumPy [8], while Plotly renders interactive charts. SQLite serves as the development persistence layer, with PostgreSQL targeted for production deployment.

TABLE I. Major system layers and observed responsibilities

Layer	Responsibility
User Interface (Streamlit)	Dataset upload, module navigation, cleaning controls, rearrangeable dashboard layout, KPI cards, filters, and slide-in Forge Assistant panel.
Application / API Layer (FastAPI)	Request validation, service orchestration, and structured endpoint routing between UI modules and processing services.
Data Processing (Pandas & NumPy)	Schema/type inspection, missing-value imputation (mean, median, mode), IQR and Z-score outlier computation, and grouped table aggregations.
Visualization Layer (Plotly)	Interactive chart generation, chart-type selection, filter-driven updates, and visual data summaries.
AI Assistance (Google Gemini API)	Conversational guidance via Forge Assistant for explaining cleaning methods, suggesting chart/table views, and assisting data exploration.
Audit & Storage Layer (SQLite / PostgreSQL)	Tracking transformation history via the cleaning log; local persistence in SQLite during development with PostgreSQL planned for production.

The end-to-end operational workflow proceeds through six stages: (1) uploading a structured dataset and inspecting dimensions, column names, data types, and summary statistics; (2) identifying missing values and anomalous observations; (3) applying selected imputation or outlier-flagging operations while logging each action; (4) generating grouped analytical tables; (5) exploring KPI cards and Plotly visualizations with interactive filtering and layout rearrangement; and (6) consulting Forge Assistant for guidance while verifying AI commentary against computed dashboard values.

V. METHODOLOGY

A. Dataset ingestion and schema inspection

Upon file upload, the ingestion module parses structured tabular data into a Pandas DataFrame [7] and extracts metadata including row count, column count, inferred data types, missing-cell counts, and descriptive statistics. The current prototype workflow has been exercised on a structured sales dataset comprising approximately 10,000 rows and 10 columns. Separating numerical and categorical columns during initial profiling ensures that subsequent imputation and outlier options match the underlying field types.

B. Missing-value imputation

To handle incomplete records transparently, the cleaning module supports mean, median, and mode imputation [4]. For a numeric column $X = \{x_1, x_2, \ldots, x_n\}$ containing m observed values, mean imputation replaces missing entries with the arithmetic mean $\bar{x}$ of the observed values, whereas median imputation substitutes the middle value of the sorted observed entries—offering greater robustness when financial or sales distributions are heavily skewed. Mode

imputation replaces missing entries with the most frequent observed value and is particularly suited to categorical attributes. Because imputation modifies original data distributions and can introduce bias under non-random missingness [4], the workflow displays the selected column, strategy, and affected cell count and records the operation rather than treating imputed entries as original ground truth.

C. Statistical outlier identification

Potential numeric anomalies are detected using two interpretable statistical methods: the interquartile range (IQR) rule and the Z-score method [5], [6]. Using the first quartile (Q_1) and third quartile (Q_3), the interquartile range is defined as $IQR = Q_3 - Q_1$, and an observation x is flagged as a potential outlier if it falls outside Tukey's fences:

$$x < Q_1 - 1.5 \times IQR \quad \text{or} \quad x > Q_3 + 1.5 \times IQR \qquad (1)$$

For approximately symmetric distributions, the Z-score standardizes each observation relative to the column mean (μ) and standard deviation (σ):

$$z = (x - \mu) / \sigma \qquad (2)$$

Observations exceeding a configurable threshold (commonly $|z| > 3$) are flagged for review. When $\sigma = 0$ (constant column values), the Z-score is undefined and must be safely bypassed without throwing runtime exceptions. Crucially, statistical extremity does not prove data corruption—for instance, a high-value wholesale purchase may be a legitimate transaction. The platform therefore separates anomaly detection and flagging from destructive record deletion, requiring explicit user review.

D. Table generation and aggregation

The table-generation module builds summarized analytical views from the prepared dataset using Pandas grouping and vectorized NumPy operations [7], [8]. Users can group records by categorical dimensions (such as product category or region) and compute record counts, totals, and averages over numeric measures such as purchase amount. Generated table structures and suggested groupings serve as inspectable analytical recommendations whose underlying columns and aggregation functions remain visible to the user.

E. Interactive dashboard and Forge Assistant

The dashboard module combines KPI summary cards, interactive Plotly visualizations, dynamic filters, chart-type selectors, and textual data summaries within a rearrangeable layout. Deriving KPI cards, charts, and summaries from the same filtered DataFrame state prevents cross-widget total mismatches. Complementing direct manipulation, Forge Assistant connects to the Google Gemini API (currently configured on the free tier) [13] via a slide-in chat panel. Inspired by recent V-NLI and LLM-assisted analytics workflows [10]–[12], the assistant helps explain cleaning trade-offs, suggest charts, and answer exploratory questions. Because generative models can hallucinate figures, numerical assertions generated by the assistant are treated as advisory and must be cross-checked against deterministically computed dashboard metrics.

F. Auditability, data protection, and scope considerations

To promote analytical traceability, the cleaning-log design records key transformation metadata—including operation type, target column(s), chosen method or threshold, affected record count, timestamp, and outcome—enabling comparison between raw and cleaned states. From a security and privacy standpoint, API credentials are managed via environment variables rather than hard-coded in source files. Uploaded datasets are processed with input validation, and confidential or personally identifiable records should not be transmitted to external LLM endpoints without appropriate privacy review and data minimization.

VI. IMPLEMENTATION STATUS

The current system status reflects a functional end-to-end prototype undergoing pre-benchmark validation rather than a fully audited production deployment.

Implemented in the current workflow: File upload and tabular profiling via Pandas and NumPy; missing-value imputation (mean, median, and mode); IQR and Z-score outlier flagging; transformation logging concepts; grouped analytical table creation; an interactive Streamlit and Plotly dashboard with KPI cards, filters, chart selection, data summaries, and rearrangeable chart layout; FastAPI backend structuring; and conversational integration with the Google Gemini API through Forge Assistant.

Explicitly incomplete or configuration-dependent: Formal quantitative benchmarking across accuracy,

latency, and usability has not yet been completed; PostgreSQL production deployment remains planned while SQLite is used during development; full dataset versioning and one-click reversible rollback across multi-step cleaning chains require further hardening; and live assistant availability depends on valid external Gemini API credentials and free-tier rate limits.

Observed workflow baseline: The integrated upload-to-dashboard pipeline has been exercised end to end by the development team using a sales dataset of approximately 10,000 rows and 10 columns, including AI-assisted chart and table recommendations. These observations confirm functional integration rather than statistically validated superiority over existing tools.

VII. TESTING AND EVALUATION

While interactive development testing confirms that the core upload, cleaning, table-creation, and dashboard modules execute end to end on the sample sales dataset, an implementation walkthrough is not equivalent to a controlled empirical benchmark. Consequently, this paper does not claim measured outlier-detection precision, execution speedup, user-satisfaction scores, or competitive superiority over commercial platforms such as Power BI or Tableau.

To validate the platform rigorously prior to production release, a six-part evaluation protocol has been designed for the next phase, summarized in Table II. For labelled outlier test cases, detection effectiveness will be quantified using Precision and Recall:

$$\text{Precision} = TP / (TP + FP), \quad \text{Recall} = TP / (TP + FN) \quad (3)$$

where TP denotes correctly flagged synthetic or known outliers, FP denotes normal observations falsely flagged, and FN denotes missed outliers. Numeric aggregation accuracy will be evaluated via absolute error |Reference Value − System Value| against independent reference scripts. Execution time will be measured across repeated runs on the 10,000-row baseline dataset and at least two larger datasets under documented hardware and software versions, reported using the median runtime.

TABLE II. Evaluation plan for the next phase

Evaluation target	Method	Report
Data-cleaning correctness	Inject known missing values and synthetic outliers into controlled test CSVs; verify imputed values, boundary cases (σ = 0), and log entries.	Pass/fail matrix, outlier Precision and Recall, and audit-log completeness.
Aggregation accuracy	Compute reference group sums, counts, and averages independently in Python/SQL and compare with generated tables and KPI cards.	Absolute error (|Reference − System|) across test aggregations.
Dashboard consistency	Apply single and multi-column filters across all charts, KPIs, and summary widgets simultaneously.	Filter consistency verification and mismatch rate (target: 0%).
Processing scalability	Benchmark upload, cleaning, and rendering latency across 10,000, 50,000, and 100,000+ row datasets on documented hardware.	Median execution time and memory footprint per stage.
AI response quality	Evaluate Forge Assistant across a fixed benchmark of 15–20 analytical and cleaning prompts against computed ground truth.	Factual consistency score, relevance rating, and hallucination rate.
System usability	Conduct task-based trials with representative student/analyst users (e.g., clean, aggregate, and build a chart).	Task completion rate, error count, and structured usability feedback.

VIII. DISCUSSION

A central architectural strength of the platform is its human-in-the-loop separation of deterministic computation from generative AI guidance. By routing

all imputations, statistical outlier calculations, and aggregations through Pandas and NumPy while isolating Forge Assistant as an advisory conversational layer, the system avoids a major pitfall of end-to-end LLM analytics tools: silent fabrication of numeric totals. Making the cleaning log inspectable further reinforces analytical trust, allowing users to trace why summary metrics changed after preparation.

At the same time, an operational prototype is not yet a universal enterprise analytics engine. Statistical imputation and univariate outlier rules (IQR and Z-score) assume relatively clean tabular schemas and may obscure domain-specific anomalies or multivariate relationships if applied without user judgment. Furthermore, free-tier API dependencies restrict throughput and raise data-governance considerations when handling proprietary business records. Responsible deployment therefore requires transparent documentation of dataset provenance, clear distinction between raw and imputed values, strict grounding of LLM outputs in computed data summaries, and empirical testing across diverse domains beyond a single 10,000-row sales file.

IX. LIMITATIONS AND FUTURE WORK

- Complete the controlled Phase 2 quantitative benchmark covering cleaning accuracy, outlier

Precision/Recall, aggregation error, and multi-scale runtime latency (Table II).

- Document exact dataset schemas, licensing, and raw anomaly distributions across multiple real-world and synthetic benchmark datasets.
- Implement non-destructive, reversible cleaning pipelines allowing users to preview before-and-after distributions and roll back individual steps.
- Migrate persistence from local SQLite to a hardened PostgreSQL deployment with role-based access control and file-validation guardrails.
- Strengthen Forge Assistant grounding by passing structured schema and pre-computed summary statistics rather than raw sensitive records, reducing hallucination and privacy risks.
- Conduct structured usability testing with non-technical target users and add exportable PDF/CSV analytical reports.

X. CONCLUSION

This paper presented Insigt Forge, an AI-powered platform that integrates automated data cleaning, grouped tabular analysis, interactive Plotly visualization, and conversational assistance within a unified Streamlit and FastAPI workflow. By combining interpretable statistical methods (mean/median/mode imputation and IQR/Z-score flagging), an auditable cleaning-log concept, and Gemini-powered guidance via Forge Assistant, the prototype lowers the technical barrier for students, small businesses, and analysts exploring structured data. While end-to-end execution has been demonstrated on an approximately 10,000-row sales dataset, formal comparative and quantitative benchmarking remains ongoing. Completing the structured Phase 2 evaluation protocol will provide the empirical measurements required to validate correctness, scalability, and AI-assistance reliability prior to broader deployment.

REFERENCES

[1] M. Hameed and F. Naumann, “Data preparation: a survey of commercial tools,” ACM SIGMOD Record, vol. 49, no. 3, 2020. doi: 10.1145/3444831.3444835.

[2] J. Zhu, X. Zhao, Y. Sun, S. Song, and X. Yuan, “Relational data cleaning meets artificial intelligence: a survey,” Data Science and Engineering, vol. 10, pp. 147–174, 2025. doi: 10.1007/s41019-024-00266-7.

[3] X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang, “Data cleaning: overview and emerging challenges,” in Proc. 2016 ACM SIGMOD International Conference on Management of Data, pp. 2201–2206, 2016. doi: 10.1145/2882903.2912574.

[4] T. Emmanuel, T. Maupong, D. Mpoeleng, T. Semong, B. Mphago, and O. Tabona, “A survey on missing data in machine learning,” Journal of Big Data, vol. 8, no. 1, Article 140, 2021. doi: 10.1186/s40537-021-00516-9.

[5] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: a survey,” ACM Computing Surveys, vol. 41, no. 3, Article 15, 2009. doi:

10.1145/1541880.1541882.

[6] P. J. Rousseeuw and M. Hubert, “Robust statistics for outlier detection,” Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 1, no. 1, pp. 73–79, 2011. doi: 10.1002/widm.2.

[7] W. McKinney, “Data structures for statistical computing in Python,” in Proc. 9th Python in Science Conference (SciPy 2010), 2010. doi: 10.25080/Majora-92bf1922-00a.

[8] C. R. Harris et al., “Array programming with NumPy,” Nature, vol. 585, no. 7825, pp. 357–362, 2020. doi: 10.1038/s41586-
020-2649-2.

[9] X. Wang et al., “VIS+AI: integrating visualization with artificial intelligence for efficient data analysis,” Frontiers of Computer Science, vol. 17, Article 176709, 2023. doi: 10.1007/s11704-023-2691-y.

[10] L. Shen, E. Shen, Y. Luo, X. Yang, X. Hu, X. Zhang, Z. Tai, and J. Wang, “Towards natural language interfaces for data visualization: a survey,” IEEE Transactions on Visualization and Computer Graphics, vol. 29, no. 6, pp. 3121–3144, 2023. doi: 10.1109/TVCG.2022.3148007.

[11] P. Maddigan and T. Susnjak, “Chat2VIS: generating data visualizations via natural language using ChatGPT, Codex and GPT-3 large language models,” IEEE Access, vol. 11, pp. 45181–45193, 2023. doi: 10.1109/ACCESS.2023.3274199.

[12] V. Dibia, “LIDA: a tool for automatic generation of grammar-agnostic visualizations and infographics using large language models,” in Proc. 61st Annual Meeting of the Association for Computational Linguistics (Vol. 3: System Demonstrations), pp. 113–126, 2023. doi: 10.18653/v1/2023.acl-demo.11.

[13] Gemini Team, Google, “Gemini: a family of highly capable multimodal models,” arXiv preprint arXiv:2312.11805, 2023. doi: 10.48550/arXiv.2312.11805.