

SAI: An Enterprise AI Analytics Platform for Automated Data-to-Insight Generation

Mrs. Mamatha Rajesh Reddy¹, D. Satyanarayana², D. Rajashwitha³, B. Rajesh⁴, Abdul Rahman⁵

¹Assistant Professor & Project Guide, Department of Computer Science and Engineering, Hyderabad Institute of Technology and Management, Telangana, India.

²³⁴⁵Students, Department of Computer Science and Engineering, Hyderabad Institute of Technology and Management, Telangana, India.

Abstract - Organizations collect operational data from spreadsheets, databases, e-commerce systems and web applications, but converting that data into decision-ready information often requires repeated manual effort. This paper presents SAI (Satya AI), a modular enterprise analytics platform that connects structured-data ingestion, data-quality checks, analytical storage, REST APIs, business-intelligence dashboards, AI-assisted narrative generation and PDF reporting. The current implementation uses Python, PostgreSQL, SQLAlchemy and FastAPI, with Power BI for interactive visualization and a Gemini-based model for narrative explanations of computed metrics. The design keeps numerical calculations in SQL or Python and uses generative AI to explain verified results rather than to serve as the source of authoritative figures. The project implementation demonstrates database connectivity, KPI and chart endpoints, persistence of generated insights and PDF report generation. Workflow scheduling, forecasting, anomaly detection and cloud deployment are treated as future extensions unless validated in a deployed build. The paper describes the architecture, processing method, implementation status, evaluation plan, security requirements and limitations of the platform.

Key Words: AI-Assisted Analytics, Business Intelligence, Data Quality, Enterprise Reporting, Generative AI, FastAPI, PostgreSQL, Power BI, KPI Computation, Decision Support.

1. INTRODUCTION

Data has become an essential asset for operational and strategic decision-making. Organizations generate information through sales transactions, customer-management systems, spreadsheets, online stores and other digital services. However, data availability alone does not ensure useful insight. Analysts may need to extract files, inspect schema differences, remove duplicates, handle missing values, calculate key performance indicators (KPIs), refresh dashboards and prepare reports. Each hand-off can create delay, inconsistency or avoidable human effort.

Business-intelligence tools make data easier to explore, while application programming interfaces allow analytical

services to be reused by different clients. Generative AI can further help communicate trends in natural language, but generated statements may be inaccurate if they are not grounded in verified data. SAI combines these capabilities into a single modular workflow. Its central principle is to calculate metrics deterministically and use AI only as a narrative layer over those metrics.

1.1 Motivation

Enterprise teams frequently work with independent spreadsheets, database exports, reporting applications and email summaries. Although each tool can solve an isolated task, repeated transfers introduce delays, manual reconciliation and differences between displayed metrics. SAI is motivated by the need for a traceable path from structured inputs to computed business measures and interpretable reports.

1.2 Contributions

- Separation of deterministic KPI computation from AI-generated business explanations.
- Relational storage and reusable API delivery of analytical results.
- Integration of Power BI views, narrative summaries and portable PDF reporting.
- Defined data-quality, evaluation and security checks for a future production workflow.
- Clear separation of demonstrated local features from proposed automation and forecasting extensions.

2. PROBLEM STATEMENT

Common analytics workflows are fragmented across file systems, databases, dashboard applications and communication channels. Repeated manual preparation increases the time needed to produce reports and makes it harder to reproduce results. A reusable workflow is needed to validate source data, store curated records, expose consistent metrics and produce both visual and written summaries.

Inconsistent schema conventions, missing values and duplicated business records can undermine otherwise correct calculations. Reporting becomes difficult to reproduce when data provenance and KPI definitions are not recorded. AI assistance adds a further risk: fluent narratives may imply unsupported trends or causation. A useful solution must therefore validate data, compute numerical results reproducibly and constrain AI to describing known values.

3. LITERATURE SURVEY

Data quality is a long-standing concern in information systems. Fox, Levitin and Redman discuss the notion of data and quality dimensions, highlighting that data must be evaluated in relation to its intended use [1]. In practical analytics, completeness, accuracy, consistency, validity, uniqueness and timeliness influence whether calculated results can be trusted. SAI therefore places profiling and validation before interpretation.

Business-intelligence platforms support data modeling, dashboards, filters and interactive visual exploration. Power BI can present KPIs and trends to users, while a separate backend can centralize business rules and make outputs available to other applications [2]. FastAPI provides a Python framework for defining and documenting REST endpoints [3], and PostgreSQL provides relational storage and SQL capabilities [4]. SQLAlchemy offers an object-relational interface between Python services and database tables [5].

Generative AI systems can produce concise explanations of structured outputs, but they should not be treated as authoritative calculators. In SAI, SQL/Python calculations are the source of numerical truth; the AI model receives a bounded context containing computed values and is asked to explain those values. This separation improves traceability and makes it easier to compare generated prose with the underlying metrics.

Table -1: Selected research and technical foundations for SAI

Source Year /	Focus	Contribution	Scope Limitation /
Fox et al. (1994) [1]	Data quality	Data quality dimensions and fitness for use.	Conceptual research rather than an SAI workflow.
Microsoft [2,6]	Power BI	Visualization, filtering and dashboarding.	Describes a BI tool, not end-to-end validation.

FastAPI [3]	REST services	Python APIs and documented endpoints.	An API framework, not analytical results.
PostgreSQL / SQLAlchemy [4,5]	Data storage	Relational persistence and application database access.	Require application-specific schemas and rules.
Gemini n8n [7,8]	AI and automation	Model integration and workflow orchestration concepts.	Do not establish implementation completeness.

3.1 Research Gap

The project focuses on integration rather than proposing a new database engine, visualization algorithm or language model. Its practical contribution is a modular data-to-insight workflow that connects validation, persistence, API delivery, dashboarding, AI explanation and report generation. The contribution should be evaluated through reproducibility, data-quality behavior, API correctness, report completeness and the factual consistency of generated narratives.

- Many tools address individual stages but do not automatically provide a fully traceable data-to-report pipeline.
- Generated explanations require checks against authoritative KPI values and reporting context.
- A functional integration demonstration is different from a measured enterprise-scale performance evaluation.

4. OBJECTIVES

The objectives are to accept multiple structured-data formats; profile and validate incoming records; store curated data in PostgreSQL; expose reusable metrics through FastAPI; visualize KPIs in Power BI; generate grounded narrative explanations; and produce portable reports. The architecture is also intended to support later automation and predictive analytics.

- Make source validation and transformation rules explicit and testable.
- Publish common metric definitions through a reusable REST interface.
- Preserve reporting periods, values, provenance and caveats with generated outputs.
- Design future orchestration without reporting untested features as completed.

5. PROPOSED METHODOLOGY

SAI is organized as a set of cooperating layers. Data sources provide structured inputs; ingestion adapters normalize the inputs; validation and transformation routines profile and clean records; PostgreSQL stores curated data; FastAPI exposes metrics; Power BI presents interactive views; and an AI/reporting component generates explanatory text and PDF artifacts. The design allows components to be tested independently and replaced without rewriting the entire platform.

The end-to-end sequence is source ingestion → schema and data-quality checks → curated PostgreSQL storage → SQL/Python KPI calculation → FastAPI delivery → Power BI visualization and AI narrative → PDF reporting. An orchestration layer may later schedule repeated execution; the supplied document does not demonstrate its production configuration.

5.1 System Architecture

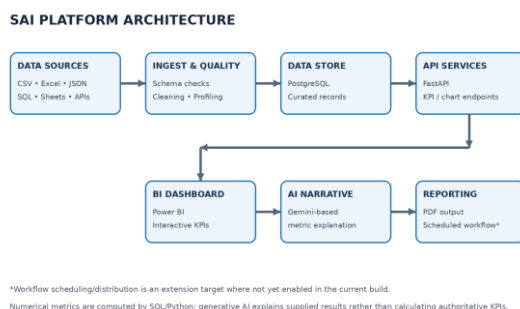


Fig -1: Proposed SAI architecture (scheduled orchestration is a future extension)

The architecture separates responsibilities. Source adapters should not contain business calculations; data-quality rules should be applied before persistence; KPI definitions should be maintained centrally; and the AI component should receive only the metrics and contextual fields required to produce an explanation. A report should preserve the metric values, reporting period and relevant caveats alongside generated prose.

5.2 Data Ingestion

The planned input categories include CSV, Excel, JSON, SQL/database sources, Google Sheets and external APIs. Each adapter converts incoming data to a consistent internal representation and records source metadata such as file name or connector, ingestion time and schema. In production, ingestion should reject unsupported formats with clear error messages and retain enough metadata to reproduce a run.

5.3 Data Profiling and Cleaning

The profiling stage records row counts, column names, inferred types, missing-value rates and duplicate indicators. Validation rules should check required fields, numeric ranges, date formats and key uniqueness where relevant. Missing values should not be filled automatically without a documented rule: depending on the field, the correct action may be to reject a record, mark it as unknown, apply a justified default or leave it unchanged. Duplicate handling must use business keys rather than assuming that every repeated row is invalid.

5.4 Curated Storage and KPI Computation

Validated records are persisted in PostgreSQL. SQLAlchemy provides the application-level database interface. Tables should use stable identifiers, appropriate data types and constraints. Where multiple datasets are ingested, source and load metadata should be retained so that results can be traced to a particular input. Database transactions should prevent partial writes when an ingestion run fails.

Business metrics such as revenue, order count, customer count and average order value should be computed using SQL queries or deterministic Python code. KPI definitions must specify filters, time periods, treatment of cancelled orders, currency units and aggregation rules. For example, average order value can be defined as eligible revenue divided by the number of eligible orders; the denominator must be non-zero and both terms must use the same reporting scope.

5.5 API and Visualization

FastAPI exposes analytical results through reusable endpoints. The demonstrated service includes KPI retrieval and chart-data endpoints, as well as operations for generating and retrieving the latest AI report. Power BI can consume curated metrics for KPI cards, tables, filters and charts. The API layer creates a stable boundary between storage and presentation, allowing additional clients to reuse the same metric definitions.

5.6 AI-Assisted Narrative Generation

The narrative-generation process begins by retrieving verified KPI values and assembling a structured context. The context should include metric names, values, units, reporting periods and any known limitations. The prompt then requests a concise explanation that does not invent metrics or claim causation unsupported by the data. The generated narrative is stored with metadata and displayed beside the underlying metrics.

Generative models can produce plausible but incorrect statements. Therefore, AI-generated text should be

checked against the source values before it is included in a report. A validation layer can detect unsupported numbers, mismatched periods or references to metrics that were not supplied. For high-impact decisions, a human reviewer should approve the narrative. The system should also log the model version, prompt template, input metric snapshot and generation time when appropriate.

5.7 Reporting Data Model

A reproducible reporting record should retain source identifiers, load time, reporting period, KPI names and units, calculated values, generation metadata and quality warnings. Keeping this information with each saved narrative allows the dashboard, API and PDF report to be checked against the same analytical snapshot. These are design requirements where a particular field or control has not been verified in the supplied implementation.

6. IMPLEMENTATION ENVIRONMENT

The current implementation was developed in Python. A project repository and virtual environment were established, dependencies were specified, PostgreSQL connectivity was configured, and SQLAlchemy was used for database interaction. The FastAPI application was run locally using Uvicorn. The analytical service exposes representative KPI and chart data, and the AI report workflow retrieves metric values, generates narrative text, persists the insight and makes the latest report retrievable. A Python-based PDF reporting workflow was also implemented.

Table -2: Software components used or proposed for SAI

Component	Technology / Platform	Purpose
Application language	Python	Data handling, business rules and report generation.
Database	PostgreSQL, SQLAlchemy	Relational storage and Python data access.
Backend / API	FastAPI, Uvicorn	Metrics, chart data and latest-report endpoints.
Visualization	Power BI	KPI cards, interactive filters and dashboards.
AI explanations	Gemini-based model	Narratives derived from verified KPI context.
PDF reporting	Python PDF workflow	Portable reports from metrics and narrative.
Future automation	n8n (proposed)	Scheduled ingestion and distribution after validation.

6.1 Backend Implementation

The implemented foundation should be distinguished from the roadmap. The available project description does not establish that every listed source connector, scheduled workflow, forecasting model, anomaly detector or cloud deployment has been completed. Those capabilities should be verified before being reported as experimental achievements.

6.2 Services and Database Access

The demonstrated FastAPI service connects application logic with stored analytical data through SQLAlchemy. API operations expose representative KPI and chart values and support the generation and retrieval of an AI report. The local Uvicorn run establishes an application foundation, but it does not by itself prove production availability, throughput or fault tolerance.

6.3 Reporting and AI Integration

Python supports data handling, API logic and report generation. PostgreSQL stores curated analytical data. SQLAlchemy manages database interactions. FastAPI exposes REST services. Power BI is used for interactive visualization, while a Gemini-based generative AI service produces narrative explanations from structured metrics. Uvicorn serves the API during local development. n8n is proposed for workflow orchestration.

7. SYSTEM FUNCTIONALITY

7.1 Data Preparation and Validation

Data adapters are planned for CSV, Excel, JSON, database tables, Google Sheets and external APIs. They should standardize schema metadata and report unsupported inputs explicitly. Data-quality processing should identify missing values, duplicates, invalid numeric ranges and inconsistent dates before those records affect KPI calculations.

7.2 KPI and Chart Endpoints

The API layer is intended to provide a common source of computed measurements for all presentation channels. Revenue, order count, customer count and average order value are representative examples. For meaningful comparisons, every measure must have a documented reporting period, filter set and denominator convention.

7.3 AI Narrative and Report Retrieval

A Gemini-based model receives structured KPI values and produces explanatory language, while numerical metrics remain under SQL/Python control. The report workflow

saves the generated insight so that the latest report can subsequently be retrieved. Narrative validation should detect unsupported quantities or periods before publication.

7.4 Example Analytics Snapshot

The original project draft records a demonstration snapshot with revenue of ₹447,700, 15 orders, 10 customers and average order value of ₹29,846.67. These values are presented as an example implementation snapshot, not as a general performance claim. An AI explanation might summarize the values, but should not infer a sales trend or business cause unless the relevant time-series or contextual evidence is available.

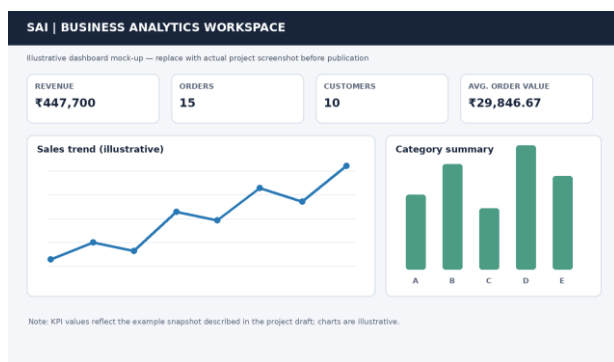


Fig -2: Illustrative KPI dashboard mock-up; not a deployed-system screenshot

7.5 Visualization and Reporting

The reporting module combines analytical values, charts and narrative content into a portable PDF. A report should identify its generation date, data period, KPI definitions and any quality warnings. Report generation should be tested for missing sections, readable labels, page overflow and correct numeric values. The same reporting service can be triggered manually or by a scheduler.

8. IMPLEMENTATION WORKFLOW

The functional design proceeds from connecting and validating data to computing KPIs, serving the values, visualizing results, generating grounded explanations and producing a report. Separating the phases permits independent tests and keeps the numerical source of truth distinguishable from natural-language presentation.

DATA-TO-INSIGHT WORKFLOW



Quality gate: Invalid records should be flagged or quarantined before they affect reported metrics.

Fig -3: End-to-end data-to-insight workflow

Table -3: SAI workflow stages and implementation boundaries

Stage	Activity / Evidence
Source setup	Configure supported source connection and record ingestion metadata.
Validation	Profile data and apply missing-value / duplicate rules.
Persistence	Store curated records using PostgreSQL and SQLAlchemy.
KPI delivery	Calculate metrics and expose API endpoints.
Narrative	Provide verified values to a Gemini-based explanation step.
Reporting	Generate PDF content and retain latest insight.
Future scheduling	Configure and test n8n only when implementation is ready.

9. RESULTS AND DISCUSSION

The project implementation demonstrates a working backend foundation rather than a completed production deployment. PostgreSQL connectivity and SQLAlchemy integration were established, the FastAPI service ran locally, KPI and chart endpoints were demonstrated, AI-assisted narrative generation was connected to analytical data, generated insights were persisted, and PDF reporting was implemented. These results support the feasibility of integrating conventional analytics with a generative explanation layer.

A major design benefit is the separation between numerical computation and natural-language interpretation. The same KPI result can be used by the API, dashboard and report, reducing the risk that separate components calculate the same metric differently. Persistence of generated insights also enables retrieval and review. However, the current evidence is functional rather than a benchmark comparison; no measured speed-up, accuracy improvement or user-productivity gain is claimed.

9.1 Backend and API Result

The original project description records PostgreSQL connectivity, SQLAlchemy integration, a locally running FastAPI application, KPI and chart-data endpoints, and a reporting path. These observations support basic service integration but provide no benchmark claim about response latency, ingestion volume or uptime.

9.2 Grounded Insights and Persistence

The reporting workflow retrieves computed metrics for a generative explanation, stores the returned insight and permits a latest-report retrieval. The separation between values and explanations is useful because an AI sentence can be checked against the exact KPI snapshot that produced it. Independent narrative-factuality measurements have not been supplied.

9.3 User-Facing Illustration



Fig -4: Conceptual analytics workspace illustration (not an implementation screenshot)

The conceptual workspace depicts a possible reporting interface and should not be mistaken for a screen capture or acceptance-test result. The paper therefore distinguishes illustrative figures, documented local capabilities and features planned for subsequent releases.

9.4 Functional Evaluation

A reproducible evaluation should use a fixed dataset and independently calculated expected values. The following tests are recommended: (i) database connection and persistence; (ii) schema validation and missing-value handling; (iii) KPI correctness against hand-checked calculations; (iv) API status codes and response schemas; (v) AI narrative consistency with supplied metrics; and (vi) PDF content and rendering. This paper does not claim performance measurements that were not recorded.

Useful measures include ingestion success rate, number of invalid or duplicate records detected, KPI agreement with an independent calculation, API response time, report-generation time and the proportion of AI statements

supported by the supplied metrics. If performance is evaluated, the dataset size, hardware, software versions, number of repetitions and measurement method should be documented. Results should be presented in a table and discussed with their limitations.

Table -4: Functional evaluation and evidence status

Test area	Expected verification	Current evidence
Database	Write / read with controlled test data.	Connectivity reported.
KPI / charts	Compare independently calculated values. to	Endpoints demonstrated; benchmark absent.
AI narrative	Check all claims against KPI snapshot.	Integration reported; score absent.
PDF output	Inspect values, layout and completeness.	Generation implemented.
Data quality	Count rejected / corrected records.	Evaluation recommended.
Security	Validate access, keys, logs and data controls.	Requirements identified.

9.5 Discussion

SAI demonstrates a practical modular integration rather than a new prediction algorithm. Central KPI logic can reduce the risk of numerical disagreement among dashboards, APIs and PDF reports. However, any measured reduction in analyst effort, improvement in correctness or gain in speed requires a controlled comparison and recorded measurements. The present results should be interpreted as functional feasibility only.

9.6 Limitations of the Present Evaluation

Connector coverage depends on the implementation of each adapter. Large-scale performance depends on schema design, indexing, query optimization and deployment resources. Generative AI can produce inaccurate or ambiguous statements and needs grounding and validation. Forecasting and anomaly detection require suitable datasets, model selection and evaluation. Enterprise use also requires authentication, authorization, monitoring, audit logging, backup, recovery and governance controls beyond a local development environment.

10. SECURITY, RELIABILITY AND DATA QUALITY CONSIDERATIONS

Database passwords and AI service keys must not be committed to source control. Development should use

environment variables, while production should use a managed secret store. API endpoints should validate inputs, enforce authentication and authorization where needed, use encrypted transport and apply rate limits. Database accounts should follow least-privilege principles. Sensitive fields should be excluded from AI prompts unless there is a justified and approved need. Logs and generated reports should avoid exposing credentials or unnecessary personal data.

10.1 Credential and Access Control

Sensitive database credentials and model service keys must be held outside committed code. Role-based permissions, least-privilege database accounts, authentication, input validation and encrypted network transport are needed before enterprise deployment. These are security requirements, not assertions that all corresponding controls have been deployed.

10.2 Data and Narrative Integrity

Every produced report should retain its source period and metric definitions. Data-quality errors need to be surfaced rather than hidden by silent replacements. Generated text must not introduce quantities or causal statements absent from the approved KPI snapshot; sensitive business fields should not be transferred to a model without a justified need.

10.3 Reliability and Scalability

Large datasets, concurrent users and scheduled runs will require indexing, transaction management, failure recovery, monitoring and tested deployment capacity. Benchmark datasets and repeatable measurement protocols are needed before making claims about enterprise-scale performance.

11. EXPECTED IMPACT

If the remaining integration, testing and governance work is completed, a consistent pipeline could reduce repetitive report preparation and help users trace AI explanations back to deterministic calculations. Expected operational benefits include reusable KPIs, portable reporting artifacts and fewer disconnected handoffs; none of these benefits is presented here as a measured productivity result.

12. FUTURE SCOPE

Future work includes natural-language analytics with validated query generation, conversational assistance, configurable anomaly detection, time-series forecasting, automated Excel/PowerPoint/PDF reports, email and collaboration delivery, role-based access control, audit trails, semantic KPI definitions and cloud-native

deployment. A further research direction is to compare AI-generated explanations with analyst-written summaries using factuality, completeness, relevance, latency and user acceptance. Benchmarking should measure the complete pipeline across increasing data volumes and report bottlenecks.

The proposed n8n scheduling and email-distribution functionality should be described as a roadmap item until runs are configured, observed and validated. Forecasting and anomaly detection similarly require appropriate datasets, baselines and evaluation metrics before they can be cited as implemented research outcomes.

13. CONCLUSION

This paper presented SAI, an enterprise AI analytics platform designed to connect structured-data ingestion, quality processing, analytical storage, APIs, business intelligence, AI-assisted explanation and PDF reporting. The implemented foundation demonstrates PostgreSQL connectivity, SQLAlchemy-based persistence, FastAPI endpoints, AI narrative generation and PDF output. The central design principle is to keep numerical analytics deterministic and use generative AI to communicate verified results. Further evaluation, stronger security controls, broader connector support, workflow automation and deployment testing are required before the platform can be characterized as production-ready.

ACKNOWLEDGMENT

The authors acknowledge the academic and technical environment that supported the project and the documentation and software communities behind the tools used.

14. REFERENCES

- [1] C. Fox, A. Levitin, and T. Redman, "The notion of data and its quality dimensions," *Information Processing & Management*, vol. 30, no. 1, pp. 9–19, 1994, doi: 10.1016/0306-4573(94)90020-5.
- [2] Microsoft, "What is Power BI?" Microsoft Learn. Available: <https://learn.microsoft.com/power-bi/fundamentals/power-bi-overview>
- [3] FastAPI, "Tutorial—User Guide." Available: <https://fastapi.tiangolo.com/tutorial/>
- [4] PostgreSQL Global Development Group, "PostgreSQL Documentation." Available: <https://www.postgresql.org/docs/>
- [5] SQLAlchemy, "SQLAlchemy Documentation." Available: <https://docs.sqlalchemy.org/>

[6] Microsoft, "Overview of visualizations in Power BI." Microsoft Learn. Available: <https://learn.microsoft.com/power-bi/visuals/power-bi-visualization-types>

[7] Google, "Gemini API documentation." Available: <https://ai.google.dev/gemini-api/docs>

[8] n8n, "n8n Documentation." Available: <https://docs.n8n.io/>

[9] pandas Development Team, "pandas User Guide." Available: https://pandas.pydata.org/docs/user_guide/index.html

[10] OWASP Foundation, "OWASP API Security Top 10 - 2023." Available: <https://owasp.org/API-Security/editions/2023/en/0x11-t10/>

Author biography: Abdul Rahman is a Computer Science and Engineering student at Hyderabad Institute of Technology and Management, with interests in Python-based application development, data analytics, business intelligence and artificial intelligence.