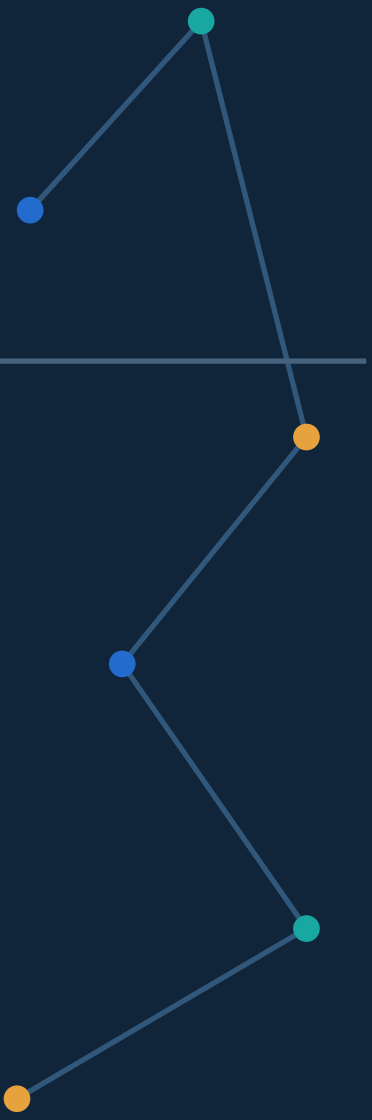
PROJECT REPORT | BUSINESS + TECHNICAL EDITION

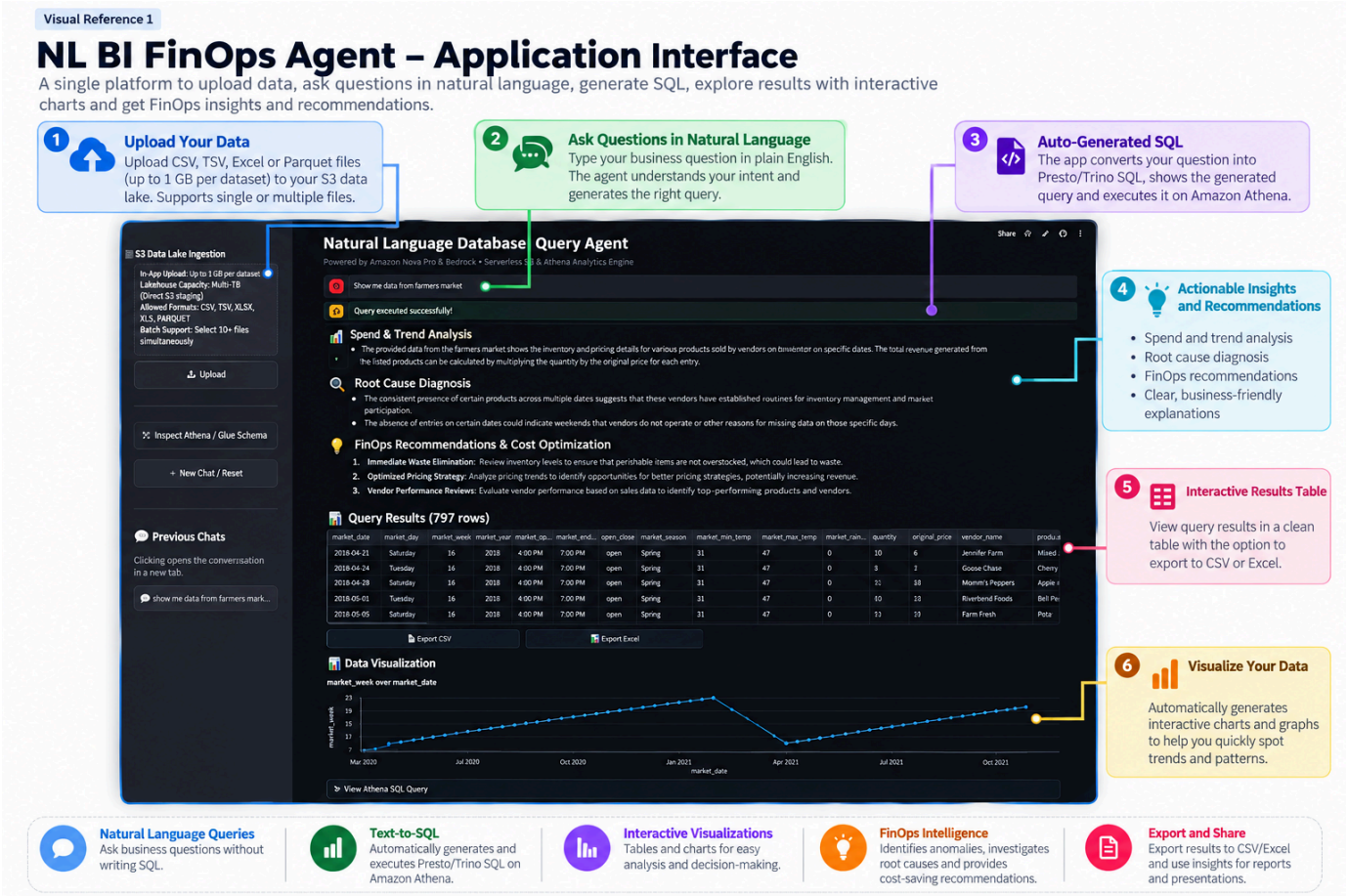
Natural Language BI FinOps Agent



A serverless AWS data lakehouse with natural-language analytics and an embedded FinOps decision-support layer

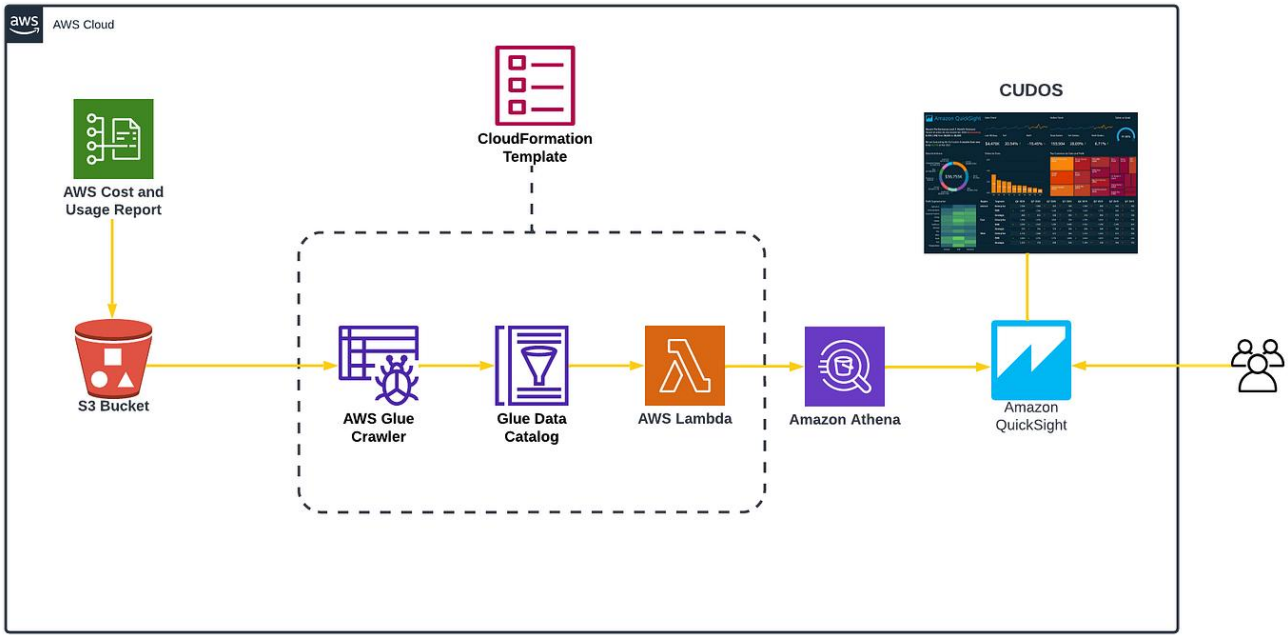
ASK IN PLAIN ENGLISH | GOVERNED SQL | LIVE VISUALS | FINOPS ACTIONS

Visual Reference 1 · NL BI FinOps Agent Application Interface



Visual Reference 2 — Traditional AWS CUR Analytics Pipeline

The supplied architecture depicts a conventional report-to-dashboard flow: AWS Cost and Usage Report data is placed in Amazon S3, catalogued through AWS Glue, processed through the illustrated Lambda/Athena path and presented in Amazon QuickSight, with CUDOS shown as a dashboard example. CloudFormation is shown as the deployment template.



How it supports the use case: this multi-service workflow is powerful, but it requires configuration and technical understanding. The proposed natural-language application aims to put a simpler conversational layer in front of the analytical workflow, so users can ask questions and receive SQL, tables, charts and FinOps guidance. The diagram is reproduced as a user-provided reference architecture; actual production implementations may vary.

Executive Summary

Businesses often have the data they need, but not an easy way to turn it into answers. Extracting insight from cloud-cost reports can require several tools and specialist skills: data storage, cataloguing, SQL querying, BI configuration and FinOps interpretation. This creates a bottleneck between the person asking a business question and the person able to answer it.

This project brings those steps into one AI-powered application. A user uploads a supported dataset, asks a question in plain English, and receives a governed SQL query, an analytical result table and an interactive visualization. For AWS cost questions, the application adds spending analysis, potential root-cause explanations and actionable FinOps recommendations.

THE CORE PROMISE

From data to decisions — without writing SQL. One interface for asking, querying, visualizing and acting on data.

At a glance

92.0%	**98.0%**	**1.42 s**	**$0.0012**
First-pass SQL accuracy 46 of 50 prompts	Accuracy after self-healing 49 of 50 prompts	Average Bedrock end-to-end latency	Average model token cost per query

These are results reported by the project's 50-prompt benchmark, not a guarantee of production performance. The evaluation uses synthetic datasets and should be expanded before enterprise-wide claims are made.

What makes the solution different

• **Accessibility:** business users can ask questions in everyday language rather than learning SQL.

• **End-to-end workflow:** ingestion, schema grounding, query generation, validation, execution and presentation are orchestrated in one application.

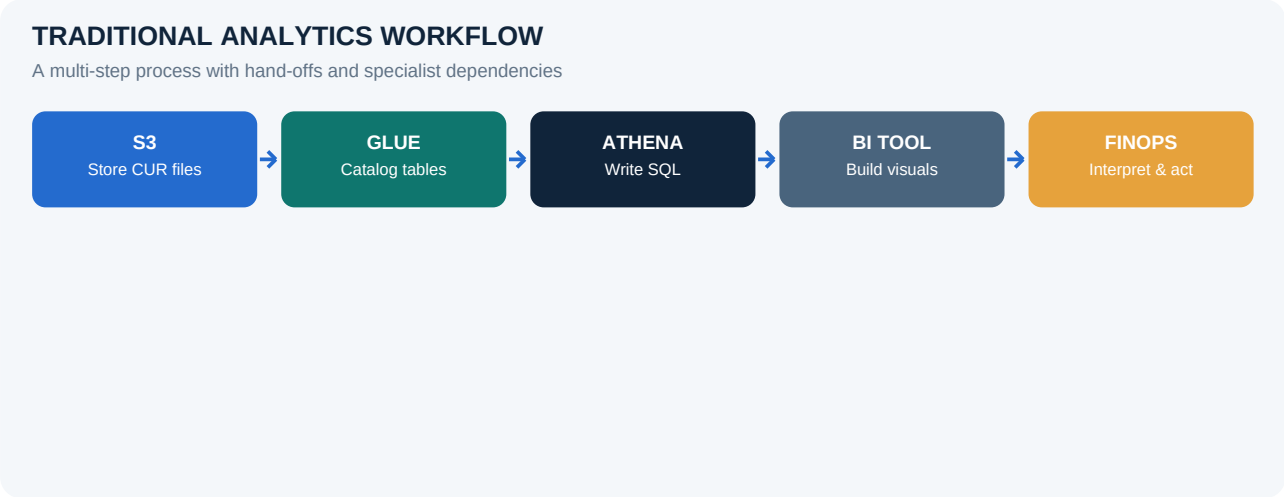
• **Decision support:** cost answers go beyond totals to explain possible drivers and suggest optimization actions.

• **Governance:** generated SQL is validated as read-only, while Athena workgroup limits help control scan cost.

1. The Business Problem: Data Is Available, Insight Is Not

Consider an organization that receives a monthly AWS Cost and Usage Report (CUR). The report contains detailed billing and usage records, but a finance manager may simply want to know why the bill increased, which services drove the change and what can be done about it.

The traditional workflow is fragmented. Each stage may involve a different tool, skill set or team. Even after the data is queryable, users still need SQL to shape the answer and a BI interface to build the desired view.



Where the friction appears

• **Tool configuration:** teams must connect and configure storage, metadata, query and visualization services.

• **SQL dependency:** every new analytical question may require a query to be written or modified.

• **BI dependency:** users often need to configure datasets, filters, charts and dashboards to get the exact view they need.

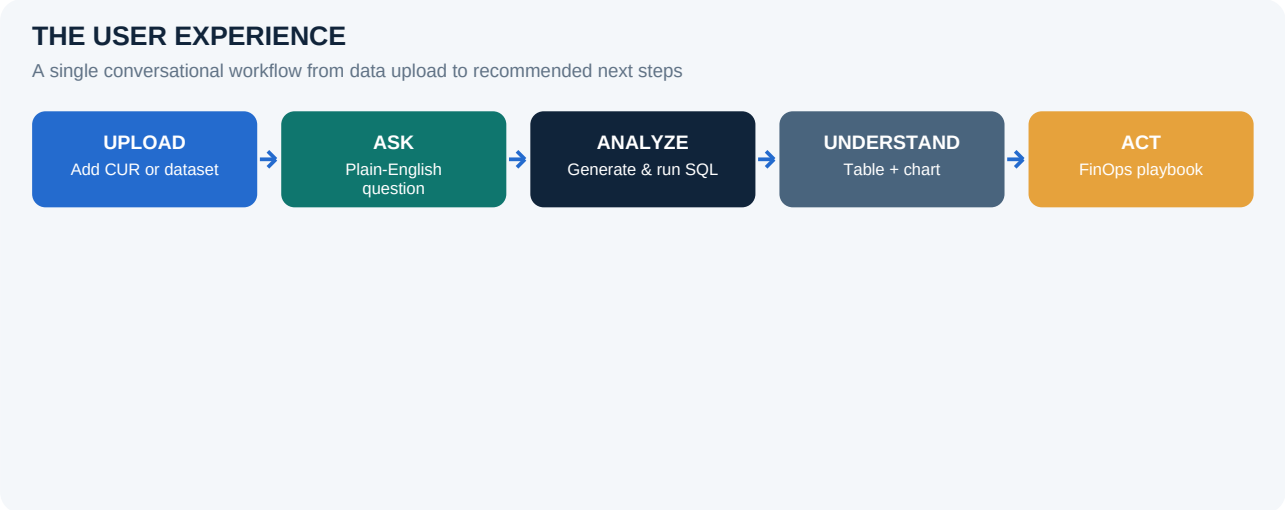
• **Interpretation gap:** a chart shows what happened, but not necessarily why it happened or what action should follow.

• **Slow hand-offs:** non-technical stakeholders may need to wait for engineering or analytics teams.

BUSINESS IMPACT	WHAT THE USER EXPERIENCES
Time-to-insight increases	A straightforward question can turn into a multi-step reporting task.
Specialist teams become a bottleneck	Finance and business teams depend on people with SQL, AWS and BI skills.
Insights do not automatically become actions	Raw spend totals still need interpretation and FinOps judgment.

2. The Proposed Solution: One Interface, One Request

The application acts as an intelligent layer over the data platform. It does not replace S3, Glue or Athena. Instead, it orchestrates them and hides much of the technical complexity from the person asking the question.



Example request

USER ASKS

“My AWS bill increased significantly. Identify the top five services driving the increase, show the trend, explain possible causes and recommend ways to optimize spending.”

The application can return

• **Generated SQL:** the query used to answer the question, so technical users can inspect it.

• **Result table:** structured figures that can be exported to CSV or Excel.

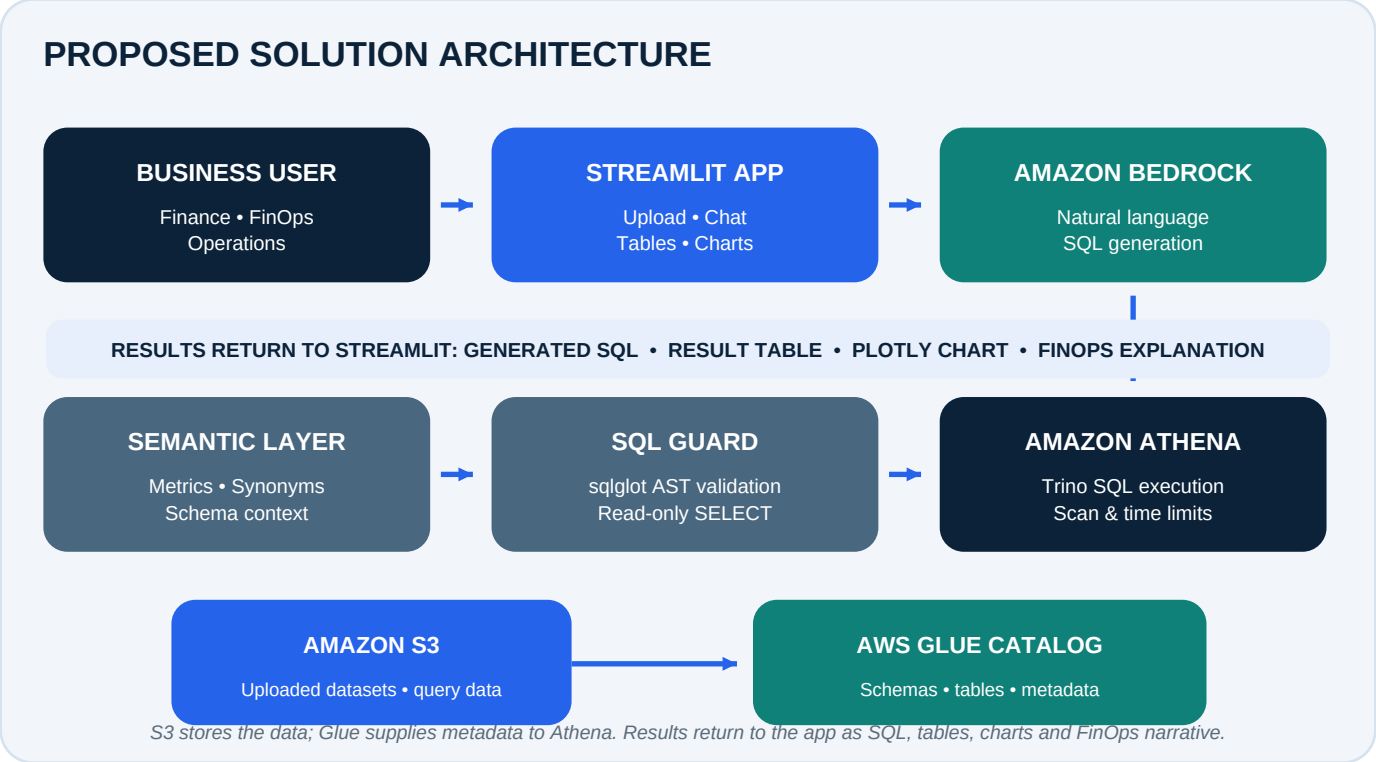
• **Interactive visualization:** charts that make trends, comparisons and outliers easier to understand.

• **FinOps narrative:** spend and trend analysis, potential root-cause diagnosis and recommended optimization actions.

The result is not just a chatbot response. It is a traceable analytical workflow in which the user can see the SQL, data output and visualization supporting the explanation.

3. Solution Architecture

The application combines a conversational interface with a governed AWS data lakehouse. Bedrock interprets requests and generates SQL; sqlglot validates it; Athena executes approved queries over S3 data catalogued by AWS Glue.



S3 stores the data; Glue supplies metadata to Athena. Results return to the app as SQL, tables, charts and FinOps narrative.

Core architecture components & implementation guardrails

LAYER	COMPONENT	RESPONSIBILITY
Presentation	Streamlit on ECS Fargate	Chat, uploads, result tables, Plotly charts and exports.
Ingestion guardrail	.streamlit/config.toml + app.py	1 GB cap; boto3 chunked S3 upload avoids full-file memory buffering.
AI reasoning	Amazon Bedrock Nova Pro	Natural-language understanding, SQL generation and FinOps narrative.
Semantic grounding	metrics.yaml + Titan Text Embeddings v2	Metric definitions, synonyms, default filters and relevant schema context.
Prompt cost hygiene	NLQueryAgent (agent.py)	Prompt and semantic rules favour required-column projection and avoid SELECT * scans.
SQL safety	sqlglot AST validation (database.py)	Allows one read-only SELECT; rejects destructive statements and multi-statement SQL.
Query engine	Amazon Athena (Trino)	100 MB per-query scan cutoff enforced by Athena workgroup configuration.
Metadata	AWS Glue Data Catalog	Provides table and column schemas used to ground SQL generation.
Storage	Amazon S3 data lakehouse	Stores uploaded datasets and Athena query outputs.

4. What Happens Behind a Natural-Language Request?

A natural-language question is converted into a governed query through a sequence of steps. The system first grounds the model in live table schemas and declared business definitions, then validates and executes the SQL before returning the answer.

01 Ask

The user enters a question in the Streamlit chat.

02 Ground

The agent retrieves Glue table schemas, metric formulas, synonyms, default filters and relevant examples.

03 Generate

Bedrock Nova Pro writes Trino SQL and infers joins across tables.

04 Sanitize

sqlglot parses the query; only a single read-only SELECT is allowed through.

05 Execute

Athena runs the query over S3 data within configured scan and time limits.

06 Self-heal

If a query fails, the error trace is returned to the model for up to three corrective attempts.

07 Explain

The app returns the result table, CSV/Excel export, interactive chart and, for cost questions, a FinOps narrative.

Why the safeguards matter

The app makes analysis easier while keeping control over what queries can do and how much cloud data they use.

1. Read-only access

The app can retrieve information, but it blocks requests that add, change or delete records. Your original data stays protected.

2. Retrieve only what's needed

It avoids asking for every column when only a few are relevant. This helps reduce unnecessary data reading.

3. Give the AI a chance to correct

If a query asks for every column, the app asks the AI to narrow it down and try again. Requests to change data are rejected.

4. AWS limits query usage

Athena enforces a 100 MB scan limit for each query. A separate time limit asks AWS to stop queries that run too long.

Together, these checks protect source data, reduce avoidable scans and stop queries that exceed configured limits.

5. The FinOps Copilot: Turning Numbers into Next Steps

A cost dashboard can show that spending increased. A useful FinOps assistant should help the user investigate what changed and identify practical actions. The project organizes cost-related answers into three tiers.

01 SPEND & TREND ANALYSIS	02 ROOT-CAUSE DIAGNOSIS	03 PRESCRIPTIVE OPTIMIZATION
Totals, comparisons, daily trends, service-level spend and percentage spikes.	Potential drivers such as commitment-fee timing, tax grouping, unamortized commitments, oversized compute or batch workloads.	Suggested actions such as scheduling non-production, Savings Plans review, rightsizing, Graviton evaluation, storage optimization and tagging governance.

Example optimization playbook

ACTION FAMILY	EXAMPLE ACTIONS
Immediate waste elimination	Review idle or unattached resources; schedule non-production workloads; clean up obsolete snapshots where appropriate.
Commitment and pricing	Review Compute and EC2 Instance Savings Plans coverage; assess Spot for fault-tolerant workloads.
Modernization and rightsizing	Use Compute Optimizer as an input to rightsizing; evaluate Graviton; review GP2-to-GP3 EBS opportunities.
Governance	Improve ownership and environment tagging; consider anomaly alerts and tag-or-terminate policies with suitable controls.

The recommendation boundary

The current report describes root causes as plausible hypotheses inferred from aggregated data, not verified findings. A recommendation should therefore be presented as an investigation or optimization opportunity until a drill-down query, resource review or engineering validation confirms it. This distinction is essential for trustworthy FinOps decision support.

6. End-to-End Business Use Case

Scenario: ABC Technologies notices that its AWS bill has risen sharply. A finance manager wants a quick explanation but does not know SQL and does not want to wait for a custom dashboard.

USER QUESTION

“Show the top AWS services by spend, identify unusual increases, compare the daily trend and recommend cost-optimization actions.”

Illustrative response layout

SERVICE	SPEND	POSSIBLE INVESTIGATION
EC2	$19,000	Rightsizing, idle hours, non-production schedule
RDS	$8,000	Instance sizing and utilization
S3	$6,000	Storage classes, lifecycle and retrieval patterns
NAT Gateway	$4,000	Data processing and traffic routing
CloudWatch	$3,000	Log retention and ingestion volume



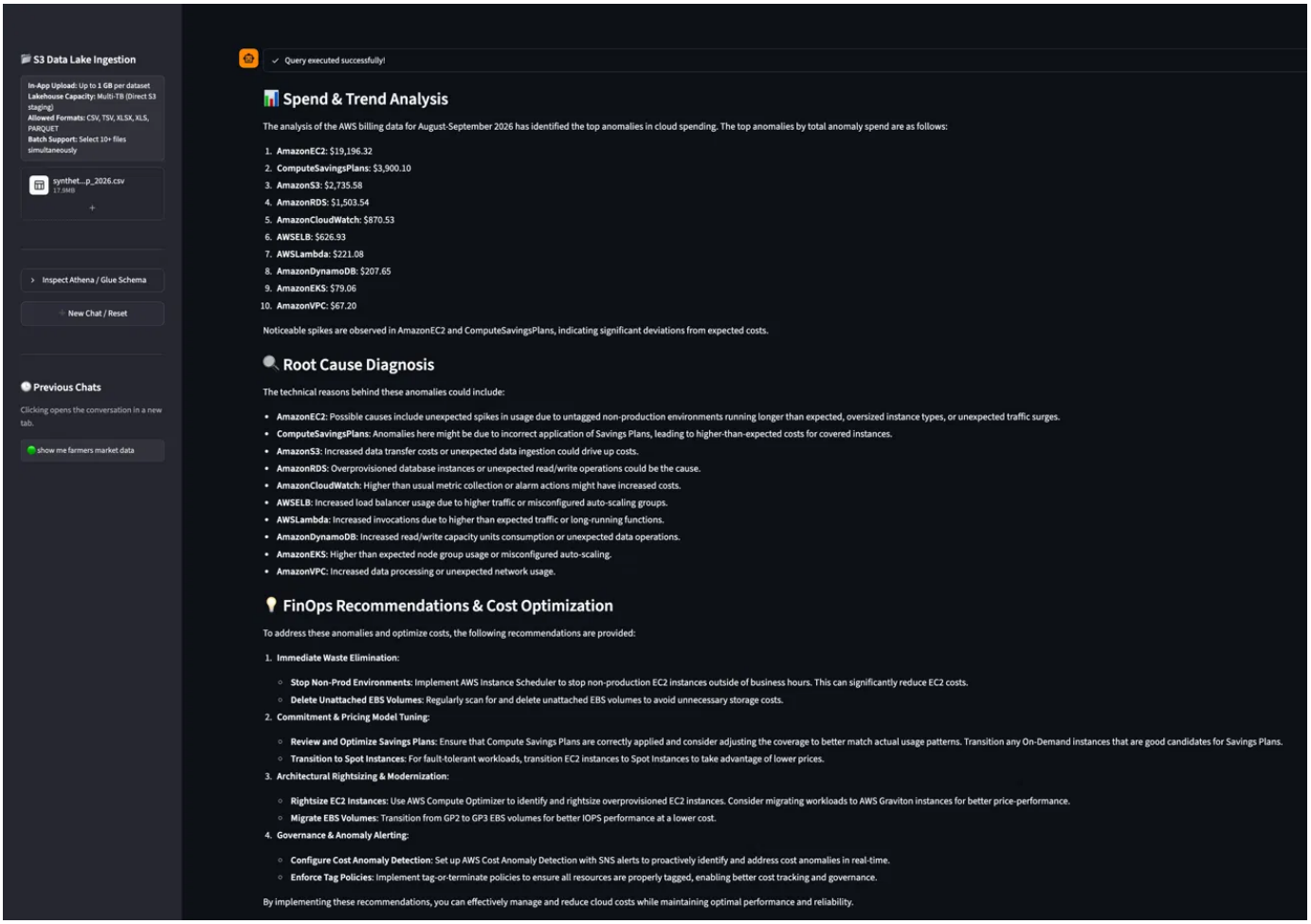
What the user gets

- A ranked view of service spending and a visual representation of the largest cost areas.
- A list of potential drivers to investigate, with the underlying query and results available for review.
- A practical set of FinOps next steps, rather than a chart alone.

Important: the service values above are fictional demonstration values to illustrate the user experience. They are not a claim about ABC Technologies or the project’s measured dataset.

Application Evidence 1-1 — Anomaly Analysis

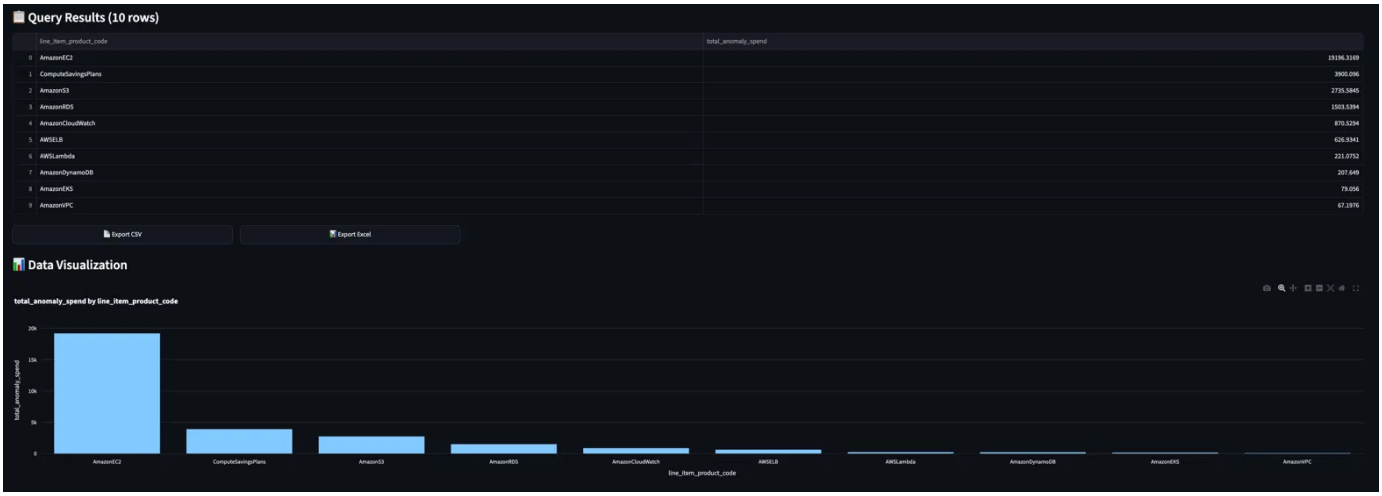
Query 1: the application interprets the request and presents the highest anomalous-spend services, possible drivers and FinOps actions.



Source: application screenshots supplied by the project owner.

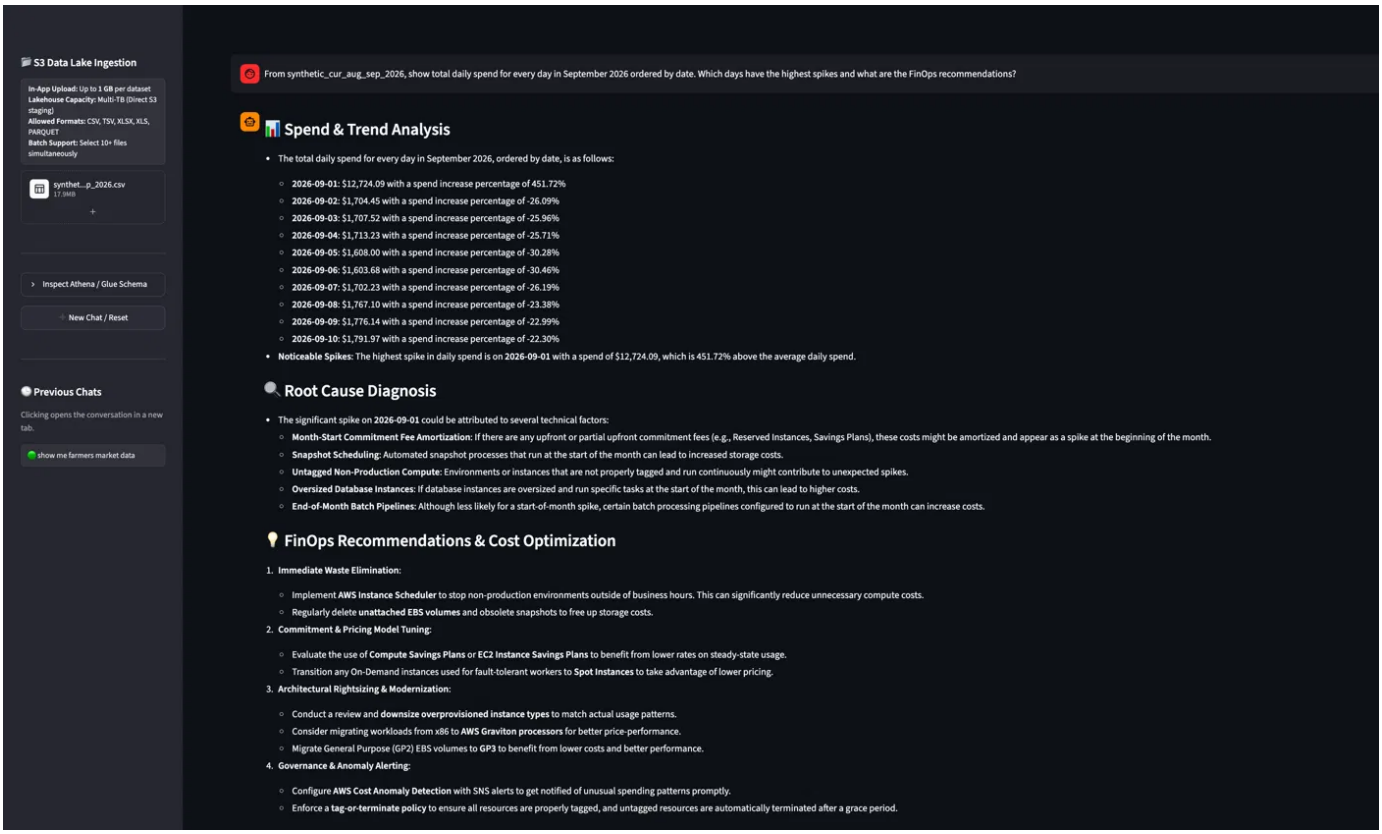
Application Evidence 1-2 — Anomaly Query Results & Chart

The matching result table and bar chart for Query 1. This follows Evidence 1-1 and makes the service-level anomaly ranking inspectable.



Application Evidence 2-1 — Daily Spend Analysis

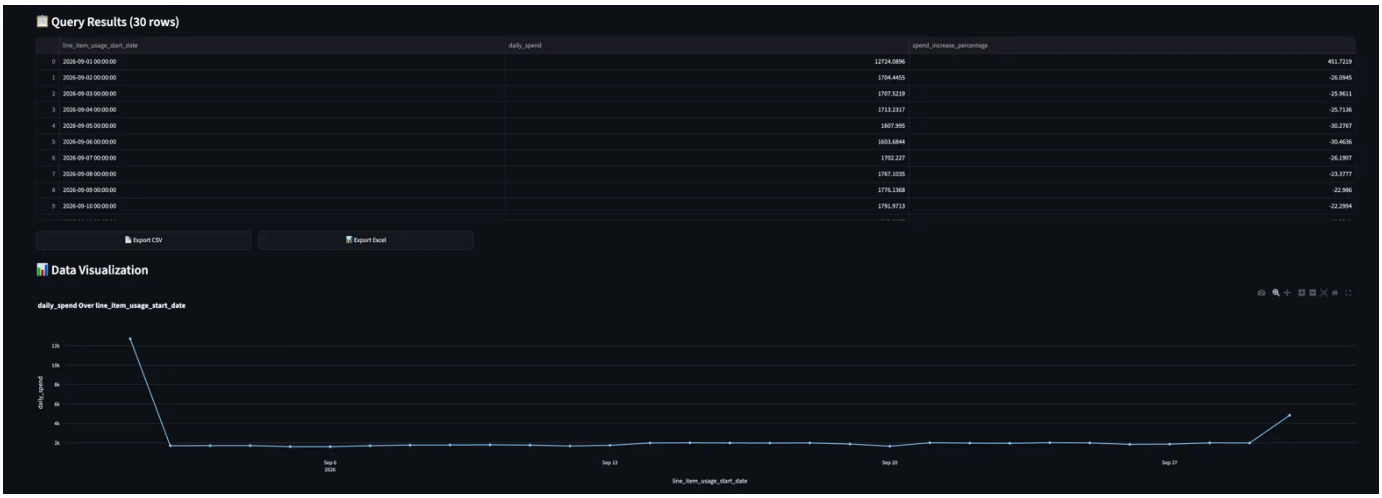
Query 2: a separate natural-language request asks for daily September 2026 spend, the largest spikes and FinOps recommendations.



Source: application screenshots supplied by the project owner.

Application Evidence 2-2 — Daily Spend Results & Trend Chart

The matching 30-row daily-spend result and time-series chart for Query 2. This follows Evidence 2-1; it is a different query from Query 1.



7. Evidence from the Project Evaluation

The project report documents a 50-prompt benchmark across FinOps cloud costs, retail, music streaming and food delivery datasets. The reported results indicate that the self-healing loop improved the overall success rate.

MEASURE	REPORTED RESULT	INTERPRETATION
First-pass SQL accuracy	92.0% (46/50)	Most benchmark prompts were correct on the first attempt.
Post-self-healing success	98.0% (49/50)	Three of four first-pass failures were recovered.
Average Bedrock latency	1.42 seconds	Average end-to-end model latency as reported.
Average model token cost	$0.0012/query	About $1.20 per 1,000 questions, excluding other infrastructure costs.
Mutation interception	100%	All destructive attempts in the test set were blocked.

FinOps case study highlights

• The anomaly analysis isolated $29,408 of anomalous spend across the synthetic CUR dataset.

• EC2 and Compute Savings Plans together represented 78.5% of the anomalous spend identified in that case study.

• A September daily-spend analysis identified September 1 at $12,724.09, reported as 451.72% above the average day.

• The application returned a table, chart and narrative, including candidate causes and an optimization playbook.

These figures are taken from the project's synthetic CUR evaluation and should be described as demonstration results, not production customer savings or verified savings achieved.

What the benchmark does not prove

A 50-prompt test is useful evidence of feasibility, but it is not a definitive enterprise accuracy study. Production CUR data can include credits, refunds, taxes and more complex amortization patterns. The benchmark should be expanded with real-world, reviewed test cases and domain-specific accuracy measurements.

8. Who Benefits and Why It Matters

USER	VALUE PROVIDED
Finance Manager	Understand service-level spend, budget changes and cost drivers without writing SQL.
FinOps Analyst	Investigate anomalies, identify optimization opportunities and prepare evidence-backed recommendations.
Cloud Architect / Engineer	Inspect resource and usage patterns, query details and technical optimization opportunities.
Business Analyst	Explore datasets, create tables and visualizations, and answer ad hoc questions more independently.
Engineering Manager	Prioritize cost work using trends and potential impact areas.
Startup Founder	Get a clearer view of cloud expenditure without building a large analytics workflow for every question.

Three layers of business value

01 DATA ACCESSIBILITY	02 ANALYTICAL ACCESSIBILITY	03 DECISION INTELLIGENCE
Reduce the technical steps between a dataset and a usable question.	Make queries, tables and charts available through natural language.	Turn analytical output into potential causes and practical next steps.

The application is not intended to replace the judgment of finance, FinOps or engineering professionals. It aims to make the first layer of analysis faster and more accessible, while keeping the query and result visible for review.

9. Positioning the Project

SHORT POSITIONING STATEMENT

“From Data to Decisions — Without Writing SQL.”

How to explain it to a business stakeholder

“Today, getting meaningful insight from a cloud-cost report can involve several manual steps: storing the report, cataloguing the data, writing SQL, configuring visualizations and interpreting the results. This often means a finance or business user has to depend on cloud engineers or analysts for every new question.

Our application brings that workflow together. Users upload their data and ask questions in plain English. The application generates and validates the SQL, runs it against the data platform and presents the results as tables and interactive charts. For AWS cost questions, it also explains spending patterns, identifies possible causes and recommends FinOps actions.

The objective is not to replace AWS services or expert decision-makers. It is to reduce the technical effort between a business question and a useful, reviewable answer — so people can spend less time extracting data and more time acting on it.”

Why this is a stronger proposition

• It describes a business pain point, not just a technical feature.

• It explains why natural language matters to non-technical users.

• It connects generated SQL and charts to a decision-support outcome.

• It is honest about the need for human validation of root causes and recommendations.

Conclusion

Natural-language analytics becomes valuable when it is accurate, governed, understandable and connected to action. This project combines a conversational interface with AWS serverless data services, SQL guardrails, query recovery and a FinOps recommendation layer. Its central value proposition is to make data analysis more accessible while preserving technical transparency through visible SQL and structured results.

Project identity and resources

Project: Autonomous Natural Language Database Query Agent & FinOps Copilot
Author: Kaustubh Kar
Application: query-agent.streamlit.app
Source code: github.com/kkaustav/query-agent (MIT License)

Source basis and interpretation note

This enhanced report is based on the supplied 19-page project report and the business-use-case framing developed in the conversation. Architecture and workflow diagrams in this edition are newly drawn explanatory diagrams. Illustrative company scenarios and service-spend figures are explicitly labelled as examples; they are not project benchmark results. Project metrics and limitations are retained as reported in the source report.