Agentic Solutions for IT Financial Operations

Bekir O. Turkkan*, Pavankumar Murali*, Chandrasekhar Narayanaswami, Vadim Sheinin

IBM Research

Abstract

The dynamic nature of cloud spending and pricing structures pose challenges for practitioners in IT Financial Operations (FinOps). Recent advances in agentic systems enables them to instead rely on agents for complex FinOps tasks such as drawing insights from their data through natural language queries. In this work, we present an IT FinOps Data Insights Agent, that implements *"chat with your data"* approach to support practitioners in their daily tasks. Our agent achieves up to 90% accuracy across ITBench FinOps scenarios.

Introduction

As organizations shift their infrastructure to the cloud, hyperscalers employ a dynamic, on-demand pricing structure making cost management a complex and continuous challenge. As per the FinOps Foundation's 2025 State of FinOps report (FinOps Foundation 2025c), over 20% of large organizations spent upwards of $1B a year on their cloud infrastructure. Organizations receive billing data in various forms, taxonomies, and metrics. The ability to interpret heterogeneous IT billing data is key for monitoring IT systems, diagnosis and remediation of events. FinOps practitioners face considerable challenges in synthesizing insights from these disparate data sources in a time-sensitive manner to drive action recommendation and cost and resource optimization.

To address these challenges, autonomous and goal-driven AI agents are being used to handle complex tasks in IT Automation and FinOps. These agents consume user queries in natural language and perform tasks that involve tasks such as analyzing cloud bills, extracting data insights, detecting anomalies, identifying causal factors for various types of events, and making proactive recommendations for cost optimization without constant human intervention. Major cloud providers are integrating such agents into their platforms to enhance FinOps practices and, in turn, enable users to manage cloud spend across their portfolios (FinOps Foundation 2025a). For example, AWS offers Cost Explorer (AWS 2025)and Cost Anomaly Detection (AWS 2024) which FinOps practitioners can use to recommend ideal compute resources to balance performance and cost as well as identify unexpected spending hikes.

*These authors contributed equally.

Recent advances in agent-based systems (e.g.: (Yao et al. 2023)) address IT incident resolution, compliance, risk management etc., but do not perform well for IT FinOps tasks such as identifying root cause for an anomalous increase in spend for a specific service or application. Consequently, there is an urgent need to develop agents that utilize and correctly interpret IT billing systems, business mappings, tags, hyperscaler pricing structures, organizational IT system topologies and hierarchies, resource utilization and observability metrics to assist practitioners in decision-making.

In this demonstration, we present versions of our FinOps Data Insights Agent with an intent to enable FinOps practitioners to interact with their data and draw insights through natural language queries. The performance of our agent is evaluated on the data insights benchmarks defined in (Jha et al. 2025). We also share our learnings based on agent trajectories, common errors encountered and improvement strategies that were found to be successful in improving our agent's performance. Although we focus on Cloud FinOps tasks for evaluation, the agentic architecture, tools used, analysis and improvement methods apply more broadly to other FinOps domains and scopes(FinOps Foundation 2025c).

System Overview

Our FinOps Data Insights Agent, built using a Langgraph architecture, enables practitioners to interact with their data through natural language queries and obtain financial data in fully customized formats by using Langchain's `ChatPromptTemplate`. It is a *planning agent* that understands a natural language query from a user, analyzes it and constructs a detailed execution plan. Each step in the plan specifies (i) the input derived from the previous step, (ii) the tool required for execution, and (iii) the expected output format for subsequent processing or for generating the final response. The agent follows the plan and validates the input–output formats at each step. As part of the *reflection* process, the agent evaluates its own actions and outputs to self-correct and improve its performance if necessary. Reflection could entail backtracking and redoing the incorrect steps or reformulating the entire execution plan. Finally, the agent returns a final response to the user to ensure an accurate and coherent response.

Landscape of FinOps Data Insights Agents

In this section, we present our approach towards developing a high-performing FinOps agent. Through iterative prompt tuning by analyzing agent trajectories, integrating advanced tools, and decomposing complex tasks, we systematically enhanced the agent's performance. **Agent v.0.1** was developed using a ReAct framework(Yao et al. 2023) with CrewAI(crewAI 2025). This agent used a custom tool for SQL operations that employed an LLM Backend to generate SQL queries based on the user, system, and agent prompts. We provided information about the data with examples and column names along with the description of the tasks. This version of our agent struggled to identify the right columns of the data and hallucinated significantly to complete the prompted task in the desired output format.

Next, for **Agent v.0.2** we analyzed the trajectories from Agent v.0.1 to identify common errors and the areas where the agent needed more guidance. We optimized the agent and system prompts to lead our agent to a more structured workflow and to avoid the errors from the earlier version. Tuning and structuring prompts improved the agent's performance in tool calling and reduced hallucinations. However, the performance in SQL generation was limited and it required tuning for each underlying LLM.

To improve the SQL generation, we replaced the custom SQL tool with an advanced Text2SQL tool(Rossiello et al. 2025) in **Agent v.0.3**. The Text2SQL tool is designed to facilitate natural language interaction with data via embedded execution pipelines and leveraging LLM technology. We used a pipeline including a schema linker, prompt optimizer, LLM inference, SQL linter, and SQL executor components. Each pipeline starts with a plain English query. Then, schema linker identifies related fields and tables from the database using structured database information which includes table and field names, types, descriptions, and example values (Glass et al. 2025). Prompt optimizer reconstructs the user and system prompts to be used for LLM inference. Next, an LLM is invoked to generate required SQL code for the task. The generated code is evaluated by the SQL linter to ensure execution without errors. Finally, SQL executor returns the result of generated SQL query. While the agent is capable of generating meaningful SQL queries, its accuracy can diminish when addressing highly complex tasks that require multiple levels of nested queries.

For highly complex queries, we developed **Agent v.0.4** by integrating Text2SQL tool with a new customized *universal tool* that executes the Python code generated by the LLM. This planning agent is guided to construct an execution plan with simpler tasks to minimize reliance on deeply nested SQL statements. It uses the Text2SQL tool to retrieve data using straightforward queries and delegates more complex calculations to the universal tool. The agent is guided with detailed tool descriptions, their capabilities, and examples in decomposing a user query into simpler tasks and delegating them to the provided tools. Agent v.0.4 stores the output from Text2SQL as a .csv file and passes the file information along with the executed SQL query to the universal tool which leverages the SQL query to learn field names and values.

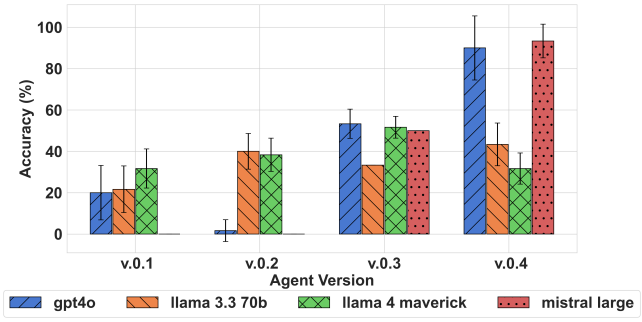

Figure 1: Average performance of the FinOps Data Agent across six ITBench scenarios over ten runs per scenario.

Evaluation

To evaluate our agents, we used a sample dataset in FOCUS format from FinOps Foundation(FinOps Foundation 2025b) and stored it into a MySQL database as a single table. We adopt six FinOps scenarios defined in the ITBench framework (Jha et al. 2025) which involves tasks including basic data retrieval and aggregation (easy), data extraction from structured data objects (medium), and construction of multiple nested queries (difficult). We have evaluated our proposed approach with state of the art LLM models as follows, `gpt-4o`, `llama-3.3-70b-instruct`, `llama4-mavericks`, and `mistral-large`.

Figure 1 presents the evaluation results for accuracy across all six scenarios, averaged over ten independent runs. The accuracy for a run can be either 100 when the agent's output matches the ground truth values, or 0 when it does not match. Our findings indicate that the applied prompt-tuning strategy is model dependent. We tuned prompts with Llama 3.3-70b model trajectories which results in performance gains only for Llama models. Incorporating the Text2SQL tool for database operations improves the performance for all models significantly, although challenges remain in handling complex queries. Decomposing user queries into simpler tasks enhances the performance substantially, with accuracy reaching up to 90% across the evaluated models.

Future Work and Discussion

FinOps tasks require accurate data insights. Our evaluations indicate Agent v.0.4 has a greater than 90% accuracy and outperforms the previous versions by greater than 80% when tested on benchmarks defined in (Jha et al. 2025).

In unsuccessful runs, we found our agent fails due to one or more of the following: (i) syntactic errors in generated SQL query or Python script (ii) repeated tool calling when not needed (iii) semantic errors such as using the wrong column names or values. Our current efforts include developing syntactic and semantic checkers as well as evaluating trajectories to alleviate such issues. Requiring direct access to data may involve challenges for adoption but our approach allows replacing data layer with API to overcome these challenges.

References

AWS. 2024. Faster anomaly resolution with enhanced root cause analysis in AWS Cost Anomaly Detection. https://aws.amazon.com/blogs/aws-cloud-financial-management/faster-anomaly-resolution-with-enhanced-root-cause-analysis-in-aws-cost-anomaly-detection/. Accessed: 2025-09-06.

AWS. 2025. AWS Cost Explorer. https://aws.amazon.com/aws-cost-management/aws-cost-explorer/. Accessed: 2025-09-06.

crewAI. 2025. crewAI. https://github.com/crewAIInc/crewAI. Accessed: 2025-09-06.

FinOps Foundation. 2025a. FinOps X 2025 Cloud Announcements: AI Agents and Increased FOCUS Support. https://www.finops.org/insights/finops-x-2025-cloud-announcements/. Accessed: 2025-09-06.

FinOps Foundation. 2025b. FOCUS sample data. https://github.com/FinOps-Open-Cost-and-Usage-Spec/FOCUS-Sample-Data/tree/main. Accessed: 2025-09-06.

FinOps Foundation. 2025c. State of FinOps: 2025 Report. https://data.finops.org/. Accessed: 2025-09-06.

Glass, M.; Eyceoz, M.; Subramanian, D.; Rossiello, G.; Vu, L.; and Gliozzo, A. 2025. Extractive Schema Linking for Text-to-SQL. arXiv:2501.17174.

Jha, S.; Arora, R.; Watanabe, Y.; Yanagawa, T.; Chen, Y.; Clark, J.; Bhavya, B.; Verma, M.; Kumar, H.; Kitahara, H.; Zheutlin, N.; Takano, S.; Pathak, D.; George, F.; Wu, X.; Turkkan, B. O.; Vanloo, G.; Nidd, M.; Dai, T.; Chatterjee, O.; Gupta, P.; Samanta, S.; Aggarwal, P.; Lee, R.; Murali, P.; wook Ahn, J.; Kar, D.; Rahane, A.; Fonseca, C.; Paradkar, A.; Deng, Y.; Moogi, P.; Mohapatra, P.; Abe, N.; Narayanaswami, C.; Xu, T.; Varshney, L. R.; Mahindru, R.; Sailer, A.; Shwartz, L.; Sow, D.; Fuller, N. C. M.; and Puri, R. 2025. ITBench: Evaluating AI Agents across Diverse Real-World IT Automation Tasks. arXiv:2502.05352.

Rossiello, G.; Pham, N.; Glass, M.; Lee, J.; and Subramanian, D. 2025. Rationalization Models for Text-to-SQL. arXiv:2502.06759.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629.