DFAgent: From Natural Language Data Interactions to Reusable Agent-Ready Tools

Neelamadhav Gantayat, Renuka Sindhgatta, Sambit Ghosh, Sameep Mehta, Soujanya Soni

IBM Research - India

neelamadhav@in.ibm.com, renuka.sr@ibm.com, sambit.ghosh@ibm.com, sameepmehta@in.ibm.com, soujanya.soni@ibm.com

Abstract

We present DataFoundry Agent (DFAgent), a system that forges reusable, agent-ready tools from interactive data exploration, quality, and remediation tasks. Users engage with data through natural-language prompts for operations that include inspection, transformation, and visualization. These interactions automatically generate executable code snippets that are logged. From these snippets, DFAgent acts as a foundry, synthesizing a governed catalog of enriched tools exposed via the Model Context Protocol (MCP). In this way, user-derived logic for all data operations is transformed into standardized, composable tools without reimplementation. We demonstrate how diverse interactions accumulate into a reusable toolset, highlighting a paradigm that unifies natural language interaction, executable code generation, and tool foundry processes for agentic data systems.

Introduction and Background

Interactive data exploration and data-quality workflows increasingly rely on a mix of natural-language interaction, executable transformations, and reusable tooling. Recent work on natural-language-to-code (NL→code) shows that large language models (LLMs) can reliably translate user intents into executable data-analysis and wrangling code, enabling richer interactive assistants for data scientists (Yin et al. 2023). Additionally, open standards for connecting models to external data sources and tools, notably the Model Context Protocol (MCP)[1], are maturing and enabling LLMs to call out to standardized tools.

Despite these advances, practical gaps remain for operational data teams. First, interactive NL→code sessions and direct data edits are typically ephemeral: transformations are executed once and often lost, making it difficult to reproduce, audit, or reuse. Second, enterprises require governance primitives such as lineage and curated catalogs of approved analysis tasks to adopt automated data tooling at scale. Third, existing systems rarely provide a shared interface through which both technical experts and business users can contribute their knowledge. This fragmentation prevents organizations from consolidating domain expertise and technical knowledge into reusable assets.

[1]https://github.com/modelcontextprotocol

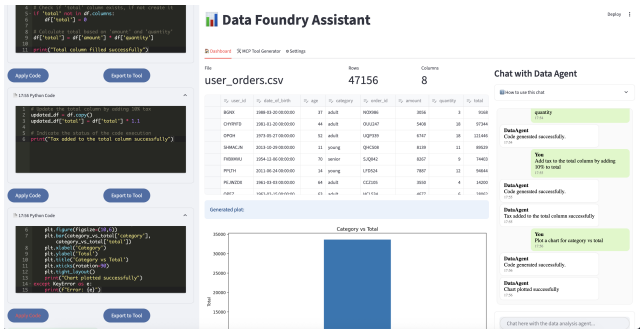

Figure 1: DFAgent: Data Engineer Interactions

We present DataFoundary Agent (DFAgent), a system that addresses these gaps by forging reusable, agent-ready tools from real-time user interactions. DFAgent (i) translates NL prompts and direct data manipulations into executable code for inspection, transformation, and visualization, (ii) logs generated snippets together with provenance metadata, and (iii) abstracts and synthesizes those snippets into a governed catalog of MCP-exposed tools.

Our contribution is threefold. First, we demonstrate a working pipeline that converts NL→code and direct data edits into reproducible code artifacts with lineage metadata. Second, we show automated synthesis of parameterized agentic tools from the code artifacts. Third, we highlight governance by enabling catalog curation and lineage that allow teams to control the tools that are promoted for reuse.

DataFoundry Agent (DFAgent)

The key novelty of DFAgent is its ability to transform transient user interactions into shareable assets. The workflow of DFAgent (Figure 2) begins with data ingestion, where users can either connect to a database or upload files to bring data into the environment. Once loaded, a data engineer can issue natural language commands such as “plot income versus age” or “update age from date of birth”. To ensure scalability on large datasets, DFAgent performs context sampling, selecting representative subsets of the data that allow the agent to reason effectively, limiting high computational costs. These sampled contexts, combined with the natural

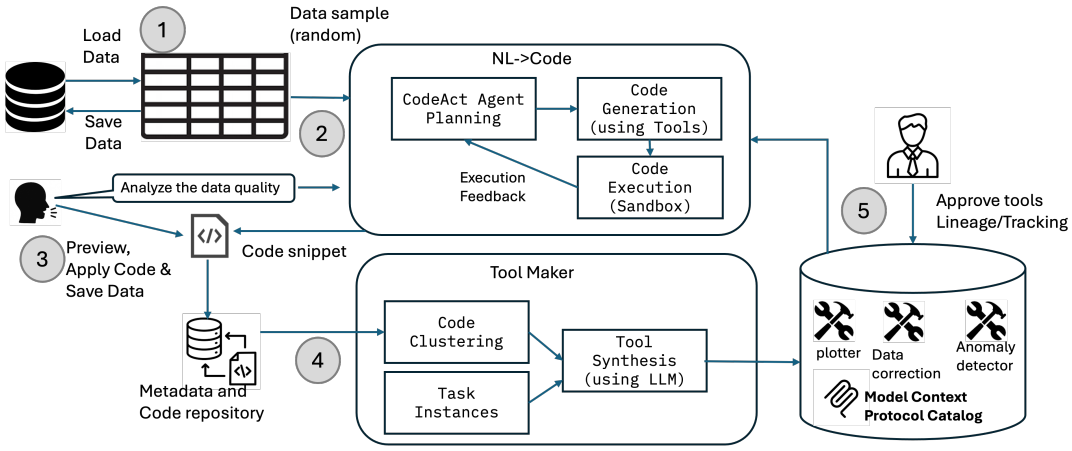

Figure 2: DFAgent: Overview

language instruction, are passed to a CodeAct agent.

The CodeAct agent (Wang et al. 2024) uses the data sample as the context and the user utterance to reason and emit an action by generating the code, taking into consideration the repository of tools available in the tool catalog. The code is executed on a sandbox execution environment, and the observation from the environment or the user is passed to the agent in each turn. Thus, in CodeAct, each emitted action to the environment is a piece of code, with the agent receiving output of code execution (e.g., results, errors) as observation. The generated code is always surfaced to the user for verification before execution on the entire data, ensuring both transparency and control.

Over time, multiple users produce overlapping or semantically similar snippets. DFAgent applies clustering and generalization to identify recurring patterns. Thus, variations of "normalize currency values" or "plot column A versus column B" are clustered and are abstracted into generic transformation or visualization tools, thus synthesizing reusable tools (Cai et al. 2024). Tool generation in our framework is shown in Figure 3.

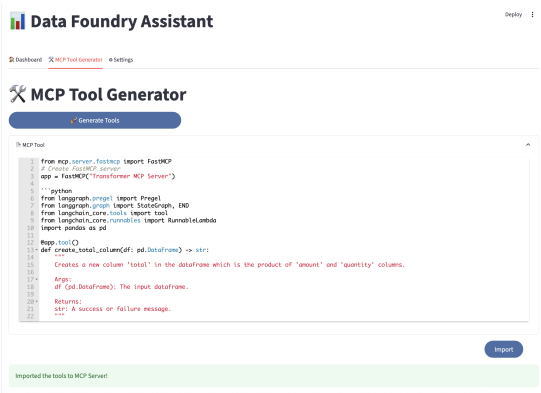

Figure 3: DFAgent: Tool generation

These generalized tools are then exposed via the Model Context Protocol (MCP), a standardized interface for tool interoperability. By packaging tools in MCP format, DFAgent ensures: i) Reusability as the tools can be invoked and used by any user interacting with DFAgent. ii) Governance, where tools undergo steward review and approval, ensuring that only approved transformations enter the shared catalog. iii) Catalog Enrichment as the catalog grows into a comprehensive library of reusable functions for data quality assessment, remediation, and exploration.

Related Work

Recent advances in code-grounded agents have emphasised the reliability benefits of producing executable actions. Translating natural language into verifiable code significantly improves robustness and transparency in LLM-based agents (Wang et al. 2024). In addition, LLMs as Tool Makers paradigm demonstrates that repeated task traces can be abstracted into reusable tools, enabling agents to accumulate capabilities over time (Cai et al. 2024). Our work builds directly on these foundations by combining reliable code generation with tool synthesis, extending them into an integrated workflow for data exploration and quality.

Existing frameworks and recent surveys highlight the need for systematic assessment and remediation in enterprise data pipelines (Pipino, Lee, and Wang 2002; Ehrlinger and Wöß 2022). Interactive systems providing natural language to code interfaces have explored usability for data exploration (Khatry et al. 2023), while governance efforts increasingly stress lineage and auditability. DFAgent unifies natural language interfaces, executable code, and governance into a practical platform, forging reusable tools.

Conclusion

We have presented DataFoundry Agent (DFAgent), a system that converts interactive data quality and exploration tasks into reusable, governed tools exposed through the Model Context Protocol. The demo highlights a new paradigm in which agents not only execute tasks but also forge reusable tools that evolve into a shared catalog, laying the foundation for scalable, trustworthy, and agent-ready data systems.

References

Cai, T.; Wang, X.; Ma, T.; Chen, X.; and Zhou, D. 2024. Large Language Models as Tool Makers. arXiv:2305.17126.

Ehrlinger, L.; and Wöß, W. 2022. A Survey of Data Quality Measurement and Monitoring Tools. *Frontiers in Big Data*, Volume 5 - 2022.

Khatry, A.; Cahoon, J.; Henkel, J.; Deep, S.; Emani, V.; Floratou, A.; Gulwani, S.; Le, V.; Raza, M.; Shi, S.; Singh, M.; and Tiwari, A. 2023. From Words to Code: Harnessing Data for Program Synthesis from Natural Language. arXiv:2305.01598.

Pipino, L. L.; Lee, Y. W.; and Wang, R. Y. 2002. Data quality assessment. *Commun. ACM*, 45(4): 211–218.

Wang, X.; Chen, Y.; Yuan, L.; Zhang, Y.; Li, Y.; Peng, H.; and Ji, H. 2024. Executable code actions elicit better LLM agents. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Yin, P.; Li, W.-D.; Xiao, K.; Rao, A.; Wen, Y.; Shi, K.; Howland, J.; Bailey, P.; Catasta, M.; Michalewski, H.; Polozov, O.; and Sutton, C. 2023. Natural Language to Code Generation in Interactive Data Science Notebooks. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 126–173. Toronto, Canada: Association for Computational Linguistics.