OMOPSYNC: A Modular Prompt Powered OMOP ETL & Analytical Software

Merlin Simoes, MS, Aum Sathwara, MAS, Bjoern Sagstad, MS, Harneet Kaur Dehiya, MAS, Vishnu Shanmugavel, MAS
Illinois Institute of Technology, Chicago, IL, United States

Abstract
OMOPSYNC lowers the barrier to using the OMOP Common Data Model by pairing a function-calling language model with the Model Context Protocol Server. Natural-language prompts are translated into deterministic ETL and query actions, while backend scripts handle data loading and analysis. This tiered design cleanly separates LLM reasoning from execution, curbs hallucinated code, and lets components be swapped without refactoring; giving researchers a concise, reproducible path from raw health data to standardized insights.

Introduction

Healthcare data is heterogeneous, often stored in disparate sources and formats across institutions and systems. This diversity presents a major barrier to scalable, reproducible research and data integration for clinical studies. To address these challenges, the Observational Medical Outcomes Partnership (OMOP) Common Data Model (CDM)[1] was developed to standardize the structure of healthcare data, enabling consistent representation and analysis across sources. However, transforming raw datasets into the OMOP CDM format remains a labor-intensive task, typically requiring extensive manual configuration of Extract, Transform, Load (ETL) pipelines, hence posing a significant technical barrier for many researchers. To lower this barrier, we present OMOPSYNC, software that enables users to trigger backend processes such as data loading and SQL query execution, and provides analytical tools through natural language prompts using a large language model and the Model Context Protocol (MCP) server architecture[2].

Architecture & Methodology

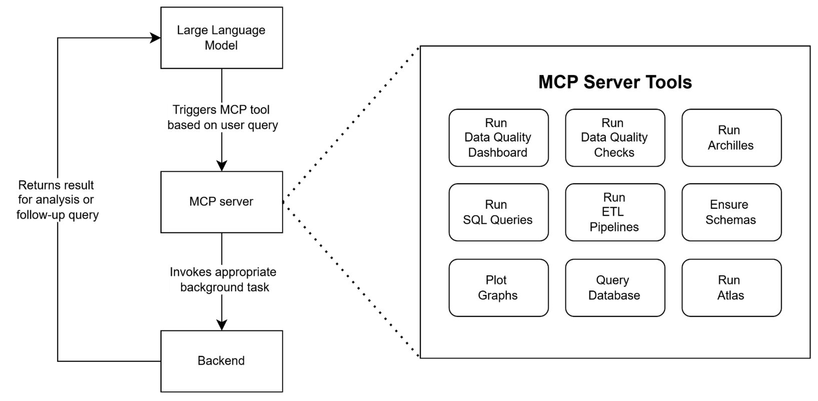


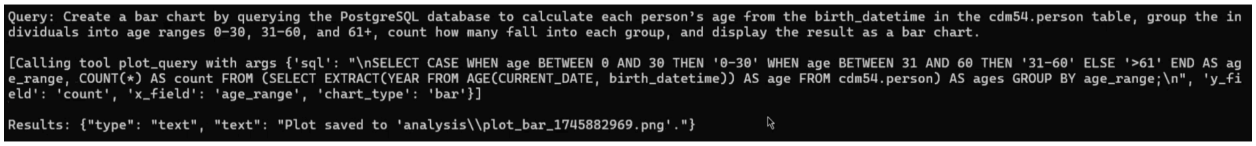
Fig. 1: OMOPSYNC Architecture

OMOPSYNC employs a three-tier modular architecture **(Fig. 1)** that separates language understanding from execution logic. In the upper layer, a large language model with function-calling capabilities parses each natural language request and maps it to a specific MCP tool. The MCP server manages execution by providing a catalog of asynchronous tools such as Run SQL Queries, Plot Graphs, and Run ETL Pipelines, each with a defined signature so that the LLM can invoke them deterministically. Communication between the MCP server and these tools occurs via the STDIO method, which eliminates network exposure and thereby reduces the risk of common web-based attacks. The bottom layer, the backend, handles the domain workflows. In our implementation, Python functions trigger R-based ETL scripts that are used to load Synthea[3] health records, OMOP CDM vocabularies, and ETL logic. These R-based ETL scripts can be replaced with equivalent Python modules without affecting the upper layers. This clean separation allows components to be hot-swapped or extended with minimal refactoring, simplifies registration of new analytical tools, and insulates LLM reasoning from low-level execution details, thereby lowering token usage and improving reproducibility.

User query → LLM parses intent → LLM calls MCP tool → MCP server calls backend function → Backend returns results → LLM synthesizes response.

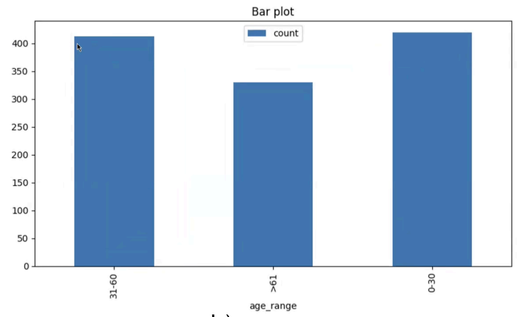
Results

A proof-of-concept implementation with Google Gemini 1.5 Pro shows that OMOPSYNC loads Synthea records into the OMOP CDM, executes cohort queries, and returns visualizations from natural-language prompts. The LLM does not generate or run code; its tokens are used only to encode the prompt and interpret outputs from deterministic MCP tools, which limits token usage. All computation remains within the tool environment, reducing exposure of protected data and constraining hallucinations. The architecture is model-agnostic, so any function-calling LLM can be substituted without modifying the backend.

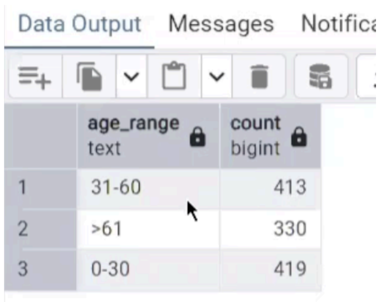


```
Query: Create a bar chart by querying the PostgreSQL database to calculate each person's age from the birth_datetime in the cdm54.person table, group the in
dividuals into age ranges 0-30, 31-60, and 61+, count how many fall into each group, and display the result as a bar chart.

[Calling tool plot_query with args {'sql': "\nSELECT CASE WHEN age BETWEEN 0 AND 30 THEN '0-30' WHEN age BETWEEN 31 AND 60 THEN '31-60' ELSE '>61' END AS ag
e_range, COUNT(*) AS count FROM (SELECT EXTRACT(YEAR FROM AGE(CURRENT_DATE, birth_datetime)) AS age FROM cdm54.person) AS ages GROUP BY age_range;\n", 'y_fi
eld': 'count', 'x_field': 'age_range', 'chart_type': 'bar'}]

Results: {"type": "text", "text": "Plot saved to 'analysis\\plot_bar_1745882969.png'."}
```

a)



b)



	age_range text	count bigint
1	31-60	413
2	>61	330
3	0-30	419

c)

Fig. 2: a) LLM calls post-query tool based on natural language query **b)** Generated bar chart **c)** Query validated via pgAdmin

In testing, a prompt to generate a bar chart (**Fig. 2 b**) led the LLM to produce a PostgreSQL query, which invoked the plot_query tool via the MCP server. The backend executed the query and rendered the chart from OMOP CDM results, which were subsequently validated via PostgreSQL.

Discussion & Conclusion

OMOPSYNC enables researchers of all technical backgrounds to execute ETL processes and explore OMOP datasets through simple natural language interactions. This approach maintains the methodological rigor and domain-specific logic required for healthcare data transformation while removing technical barriers to implementation. A key distinction of our approach is that AI does not generate or replace ETL pipelines; instead, it serves as a flexible interface that builds on predefined workflows to improve their accessibility and usability. OMOPSYNC demonstrates that an AI-triggered and AI-queried system of OMOP-compliant ETL pipelines can support data quality, streamline access to standardized data, and facilitate research workflows.

In conclusion, this balance between AI-enhanced usability and rigorous methodological standards offers a promising direction for healthcare data standardization efforts.

References

1. Observational Health Data Sciences and Informatics (OHDSI). Common Data Model. In: The book of OHDSI. Published 2021. Available at: https://ohdsi.github.io/TheBookOfOhdsi/CommonDataModel.html. Accessed April 30, 2025.
2. Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions. 2025. Available at: https://doi.org/10.48550/arXiv.2503.23278 Accessed April 30, 2025.
3. Walonoski J, Kramer M, Nichols J, et al. Synthea: an approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *J Am Med Inform Assoc.* 2018;25(3):230–238. doi:10.1093/jamia/ocx079. Correction in: *J Am Med Inform Assoc.* 2018;25(7):921. doi:10.1093/jamia/ocx147.