Translating Natural Language Queries into UDF-Centric SQL Queries using LLMs

Catlynh Nguyen, Computer Science
Mentor: Dr. Jia Zou
School of Computing and Augmented Intelligence

OBJECTIVE & RESEARCH QUESTION

Can large language models generate both **User-Defined Functions (UDFs)** and the SQL queries that depend on them from natural language, without assuming that the required functions are already defined?

Existing Text-to-SQL systems assume all necessary database functions exist at query time. This research addresses the case where they do not.

BACKGROUND

Text-to-SQL research has matured rapidly, with systems like DIN-SQL [2] achieving strong performance on standard benchmarks (Spider, BIRD). These systems translate a natural language query into a SQL statement given a fixed schema and predefined functions.

Real-world analytical queries, especially in specialized domains such as clinical medicine, frequently require **custom logic**: threshold-based classifications, temporal reasoning over patient histories, or embedded ML model inference. This logic cannot be expressed in standard SQL without helper functions. **User-Defined Functions (UDFs)** fill this role. But no existing Text-to-SQL pipeline generates UDFs on demand.

THE PROBLEM

Standard Text-to-SQL pipelines break down when required functions are absent:

Traditional Pipeline

"Flag patients with critically low eGFR"

```
SELECT * FROM patients
WHERE eGFR_critical(...)
-- function undefined
```

ERROR: UDF does not exist — query cannot execute

This System

"Flag patients with critically low eGFR"

```
CREATE OR REPLACE MACRO
ckd_stage(value) AS ( CASE WHEN
TRY_CAST(value AS DOUBLE) >= 90
THEN 'Stage 1' WHEN ... ELSE
'Invalid' END )
```

```
SELECT patient_id,
ckd_stage(value) AS stage FROM
observations WHERE code = 'ckd'
```

DATASET

Applied to a clinical dataset of 785,902 medical records extracted from the MedAgentBench [1] FHIR server and loaded into DuckDB [4] across five relational tables: patients, observations, conditions, medications, and procedures. Evaluation uses seven medical UDF scenarios spanning glucose classification, potassium risk assessment, chronic kidney disease staging, HbA1C classification, creatinine elevation detection, temporal reasoning, and multi-parameter diabetes risk scoring, each requiring a distinct UDF paired with a dependent SQL query.

METHODOLOGY

A **multi-agent agentic framework** (AgentFlow) [3] is used in place of a single-pass LLM call. The pipeline decomposes the task across four roles with iterative verification and no dependence on ground-truth labels.

Planner
Receives the natural language query and memory of previous steps. Generates the next action: identifies which tool to call, defines the sub-goal, and provides context. A Domain Workflow block in the prompt enforces the five-step UDF generation sequence, ensuring UDF registration occurs before the pipeline concludes.

Executor
Runs the tool specified by the planner. Generates the exact tool command via LLM, executes it against the tool instance, and stores the result. Emits step-level callbacks for live tracing. Does not decide what to do, only carries out the planner's instructions.

Medical_Threshold_Tool
Retrieves clinically validated reference ranges, return types, sentinel bounds, and UDF specifications for a given lab test. Acts as a validated domain knowledge store

DuckDB_Executor_Tool
Executes SQL operations against the live DuckDB database. Handles schema inspection, UDF macro registration, UDF testing, and result retrieval.

UDF_Validator_Tool
Checks return type correctness and reviews UDF logic (NULL handling, sentinel values, threshold accuracy). A FAIL triggers a correction loop.

Verifier
Checks memory after each step to determine whether the full five-step workflow has been completed (thresholds retrieved, schema inspected, UDF registered, type verified, logic critiqued). Returns CONTINUE if any step is missing or failed, STOP only when all steps pass. Also includes a deterministic FAIL check that prevents premature stopping when the last memory entry contains an explicit failure verdict.

Generator
Composes the final response from memory once the verifier issues STOP. Returns the registered UDF definition, the SQL query that uses it, and a process summary.

REFERENCES

[1] Jiang, Y. et al. "MedAgentBench: A Virtual EHR Environment to Benchmark Medical LLM Agents." *NEJM AI*, vol. 2, no. 9, 2025.
[2] Pourreza, M. & Rafiei, D. "DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction." *NeurIPS*, 2023.
[3] AgentFlow: Agentic Orchestration Framework. agentflow.stanford.edu
[4] DuckDB: An Embeddable Analytical Database. duckdb.org

RESULTS

AgentFlow achieves 100% accuracy across all 39 test cases, outperforming LangGraph, an early agentic prototype without domain workflow guidance. LangGraph fails entirely on Creatinine, Days Since, and Diabetes Risk, which are three scenarios where inline SQL cannot express the required computation without a registered function.

Scenario	LangGraph	AgentFlow
Glucose	5/5	**5/5**
Potassium	5/5	**5/5**
CKD Staging	6/6	**6/6**
HbA1C	5/6	**6/6**
Creatinine	**0/6**	**6/6**
Days Since	**0/6**	**6/6**
Diabetes Risk	**0/5**	**5/5**
TOTAL	**21/39**	**39/39**

LangGraph: early agentic prototype. AgentFlow: final system with domain workflow guidance and iterative validation.

MEMORY & ABLATION FINDINGS

A leave-one-out ablation study measures each tool's contribution by checking how often removing its memory segment changes the Verifier's STOP/CONTINUE decision. UDF_Validator_Tool has the highest flip rate at 47.1%, confirming that validation is the most critical stage of the pipeline.

Tool	Flip Rate
UDF_Validator_Tool	**47.1%**
DuckDB_Executor_Tool	**18.8%**
Medical_Threshold_Tool	**14.3%**

Flip rate = how often removing a memory segment changes Verifier's STOP/CONTINUE decision. Higher = more critical.

CONCLUSIONS & FUTURE WORK

An agentic pipeline with iterative verification generates and validates UDF-centric SQL queries from natural language, achieving 100% accuracy on seven medical scenarios where LangGraph fails or partially fails on four scenarios.

Ablation confirms UDF validation is the most critical pipeline stage: a 47% flip rate versus 14% for threshold retrieval, motivating the correction loop design.

Future work includes extending to Python-embedded UDFs for computations SQL cannot express, including trend detection, anomaly detection, and ML model inference, as well as reducing dependence on domain-specific context toward more realistic natural language inputs.

FURI

Ira A. Fulton Schools of Engineering
Arizona State University