Build-Your-Own SQL Database Agent (From Scratch)

Vizuara's LLM Production and Deployment

Max team size: 2 — **Deadline: November 10, 2025 (23:59 IST)**

1. Overview & Learning Objectives

You will build your own agent library entirely from scratch. This assignment focuses on designing and implementing a lightweight agent that interprets natural language and interacts with a SQL database using the **ReAct** (Reasoning + Acting) **Thought** → **Action** → **Observation** loop. No agent frameworks may be used.

By completing this assignment, you will:

- Implement a ReAct (Thought–Action–Observation) loop from scratch.
- Design and integrate tools for LLM-based agents.
- Create concise prompts that induce structured reasoning.
- Handle errors, timeouts, and edge cases in agent systems.
- Understand foundational ideas behind libraries such as LangChain and SmolAgents.

2. What You Will Build

A single-purpose, read-only **SQL Database Agent** that can:

- Explore database schemas (list and describe tables).
- Answer natural-language questions about the data.
- Execute *safe* SQL `SELECT`-only queries.
- Explain its reasoning step-by-step via auditable traces.

3. Technical Requirements

3.1 Constraints

- **Code Length: 350–400 lines** of Python (excluding comments and docstrings). This is not a strict requirement, but the focus will be on having the code as small as possible.
- **Dependencies:** One LLM client library (*OpenAI or Anthropic*), `sqlite3` (stdlib). Optional: `tabulate`.
- **Prohibited:** No agent frameworks (e.g., LangChain, SmolAgents, crew libraries).
- **Database:** SQLite (`.db`) read-only. Your agent must not perform mutations.

3.2 Required Components

A. Core Agent Class (120–150 lines)

- Initialize with LLM client, DB connection, and tool registry.
- Implement a bounded-step **ReAct** loop.
- Parse LLM outputs to extract `THOUGHT`, `ACTION{json}`, and parameters.
- Maintain conversation history and produce a `FINAL_ANSWER`.

B. Tool System (80–100 lines)

- Base Tool interface (name, description, parameters, call).
- Required tools:
  - `list_tables()`: list all tables.
  - `describe_table(table_name)`: schema (columns + types, and row count if convenient).
  - `query_database(query)`: `SELECT`-only execution with safe validation.
- Standardized tool descriptions must be surfaced to the LLM.

C. Prompt Engineering (50–70 lines)

- A concise *system prompt* enforcing ReAct formatting.
- 1–2 minimal few-shot traces (keep them short; no full solutions).
- Clear tool documentation block (names, args, returns).
- Output parsing template for `THOUGHT`, `ACTION`, `OBSERVATION`, `FINAL_ANSWER`.

D. Utilities (50–80 lines)

- SQL validation: `SELECT`-only; prohibit `INSERT/UPDATE/DELETE/DROP/ALTER`, subqueries optional to forbid.
- Result formatting (e.g., small tables to readable text).
- Error handling and retry logic; bounded time/row caps (e.g., `LIMIT`).
- Lightweight logging for debugging runs.

4. Detailed Specifications

4.1 ReAct Loop

The agent must follow: **User Query** $\rightarrow$ **Thought** $\rightarrow$ **Action** $\rightarrow$ **Observation** $\rightarrow$ **Thought** $\rightarrow \cdots \rightarrow$ **Final Answer**.
Formatting contract (for parsing):

```
THOUGHT: <brief plan>
ACTION: <tool_name>{<valid JSON args>}
OBSERVATION: <tool result (trimmed)>
...
FINAL_ANSWER: <concise answer>
```

4.2 Tool Specifications

Tool: `list_tables`
Description: Lists all tables in the database.
Parameters: None.
Returns: List of table names.

Tool: `describe_table`
Parameters: `table_name (str)`.
Returns: Schema details (columns and types; row count if feasible).

Tool: `query_database`
Description: Executes a `SELECT` query (read-only).
Parameters: `query (str)`. (You may also support `params` for prepared statements.)
Returns: Formatted result table or clear error message.

4.3 Safety, Validation, and Scope

- **Read-only:** Only `SELECT ... FROM ... [WHERE ...] [GROUP BY ...] [ORDER BY ...] [LIMIT ...]`.
- **Validation:** Reject non-`SELECT` statements. Whitelist identifiers (tables/columns) derived from `describe_table`.
- **Limits:** Enforce row caps (e.g., `LIMIT 100`) and reasonable timeouts.
- **Joins:** Optional. If included, keep to a single `INNER JOIN` or provide a pre-aggregated view in the DB.

4.4 Error Handling Requirements

Your agent must gracefully handle:

- **Invalid SQL:** Catch and surface syntax errors with a helpful suggestion.
- **Non-`SELECT`:** Refuse and explain the read-only constraint.
- **Parsing errors:** If LLM output deviates from the contract, request a properly formatted step (within the step budget).
- **Missing tables/columns:** Suggest using `list_tables` or `describe_table`.
- **Max step limit:** Return the best partial result and a summary of attempts.
- **LLM API errors:** Implement exponential backoff with a small retry budget.

5. Deliverables

A. Code (single small package or single file): Agent class, tools, prompts, and utilities; within 350–400 LOC (excl. comments/docstrings).

B. README (1 page): Purpose, how to run, tool list (arguments & returns), and a short example run with trace.

C. Tests (3–5): Cover schema discovery, simple aggregation, and one error-recovery case.

D. Trace Logs: A sample run with full `THOUGHT/ACTION/OBSERVATION/FINAL_ANSWER`.

E. Reflection ($\leq$ 300 words): Design trade-offs, failure modes, and future directions.

6. Grading Rubric (100 points)

Criterion	Points
ReAct fidelity (clear Thought/Action/Observation, clean termination)	20
Tooling (design, validation, helpful error messages)	20
NL→SQL handling (intents, predicates, grouping/ordering, limits, safety)	25
Safety & Validation (read-only, identifier whitelisting, row/time caps)	15
Tests & Reproducibility (clear instructions, passes on provided DB)	10
Trace Quality (auditable, minimal yet sufficient)	10
Total	**100**

7. Submission Instructions

- Submit a zipped archive or repository link containing:
  - Source code in the form of a Github repository.
  - A detailed PDF file containing your report.
- Include a one-line command to run your agent on a sample query.
- **Deadline: November 10, 2025 (23:59 IST)**.

8. Team & Integrity

- **Team size:** Up to **2** students.
- All work must be your own. Cite any external resources used (e.g., small parsing ideas or SQL patterns).
- You *may not* use agent frameworks; the agent, tools, and loop must be implemented **from scratch**.

Emphasis: This assignment is explicitly about building your own agent library *entirely from scratch*. You will design the ReAct loop, the tool registry, prompt scaffolding, SQL validation, and error handling yourself, within a concise code length.

Questions? Post on the course Discord.