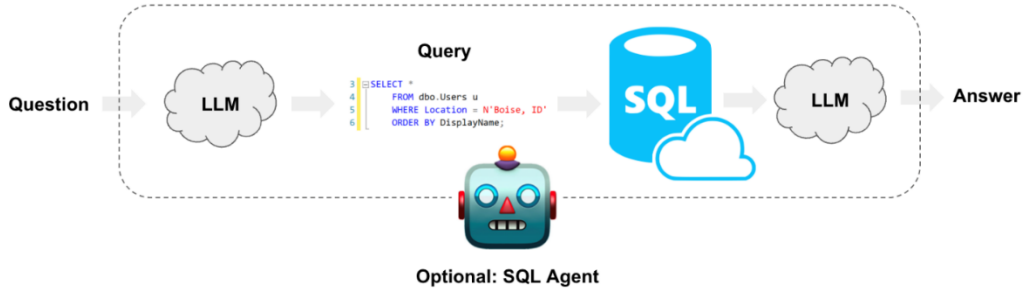
Langchain Agent

Build a Question/Answering system over SQL data | 🦜🔗 LangChain

SQL | 🦜🔗 LangChain

At a high-level, the steps of any SQL chain and agent are:

1. **Convert question to SQL query**: Model converts user input to a SQL query.
2. **Execute SQL query**: Execute the SQL query.
3. **Answer the question**: Model responds to user input using the query results.



SQL Query Chain

Let's create a simple chain that takes a question, turns it into a SQL query, executes the query, and uses the result to answer the original question.

SQL Agents

LangChain has an SQL Agent which provides a more flexible way of interacting with SQL databases. The main advantages of using the SQL Agent are:

- It can answer questions based on the **databases' schema** as well as on **the databases' content** (like describing a specific table).
- It can **recover from errors** by running a generated query, catching the traceback and regenerating it correctly.
- It can answer questions that require **multiple dependent queries**.
- It will save tokens by only considering the schema from relevant tables.

CSV

The two main ways to do this are to either:

- **RECOMMENDED**: Load the CSV(s) into a SQL database, and use the approaches outlined in the SQL use case docs.
- Give the LLM access to a Python environment where it can use libraries like Pandas to interact with the data

Agents Framework:

- Langgraph
- Langchain + Openai function-calling
- Crew.ai
- Autogen
- Semantic Kernel
- LlamaIndex

Agents components:

- ReAct implementation
- Tools (Function calling/External tool)
- Memory
- Guardrails
- Human-in loop

Agentic Strategies:

- SQL Agents
- CSV/Pandas Agent
- Agentic RAG
- Multi agent
- Reflective agent
- Planning agent

https://colab.research.google.com/drive/17eOkz1wTNreK6dhNg0D3n6xbUBVOY2qd#scrollTo=da8foCBB75lr

LangChain:

- https://python.langchain.com/docs/tutorials/agents/
- https://python.langchain.com/docs/concepts/#agents
- https://python.langchain.com/docs/how_to/#agents
- How-to guides | 🦜🔗 LangChain
- Build a Question/Answering system over SQL data | 🦜🔗 LangChain
- How-to guides | 🦜🔗 LangChain

LangGraph:

-

Semantic Kernel:

- https://learn.microsoft.com/en-us/semantic-kernel/overview/

Autogen:

- https://microsoft.github.io/autogen/docs/Getting-Started/

Crew ai

- https://docs.crewai.com/

Llama-Index

- Building a basic agent - LlamaIndex
- How to Build a Chatbot - LlamaIndex
- Agents - LlamaIndex
- Agentic strategies - LlamaIndex
- Core Agent Classes - LlamaIndex

Langchain - https://learn.deeplearning.ai/courses/functions-tools-agents-langchain/

Langgraph - https://learn.deeplearning.ai/courses/ai-agents-in-langgraph/

LlamaIndex- https://learn.deeplearning.ai/courses/building-agentic-rag-with-llamaindex/

Crew.ai - https://learn.deeplearning.ai/courses/multi-ai-agent-systems-with-crewai/

Microsoft - https://learn.deeplearning.ai/courses/ai-agentic-design-patterns-with-autogen/

Microsoft - https://learn.deeplearning.ai/courses/building-your-own-database-agent/

Microsoft - https://learn.deeplearning.ai/courses/microsoft-semantic-kernel

Nexusflow - https://learn.deeplearning.ai/courses/function-calling-and-data-extraction-with-llms/

Framework	Key Focus	Strengths	Best For
Langchain	LLM-powered applications	Versatility, external integrations	General-purpose AI development
LangGraph	Stateful multi-actor systems	Complex workflows, agent coordination	Interactive, adaptive AI applications
CrewAI	Role-playing AI agents	Collaborative problem-solving, team dynamics	Simulating complex organizational tasks
Microsoft Semantic Kernel	Enterprise AI integration	Security, compliance, existing codebase integration	Enhancing enterprise applications with AI
Microsoft Autogen	Multi-agent conversational systems	Robustness, modularity, conversation management	Advanced conversational AI and task automation

Query Construction

Third, consider whether any of your data sources require specific query formats. Many structured databases use SQL. Vector stores often have specific syntax for applying keyword filters to document metadata. **Using an LLM to convert a natural language query into a query syntax is a popular and powerful approach.** In particular, text-to-SQL, text-to-Cypher, and query analysis for metadata filters are useful ways to interact with structured, graph, and vector databases respectively.

Name	When to Use	Description
Text to SQL	If users are asking questions that require information housed in a relational database, accessible via SQL.	This uses an LLM to transform user input into a SQL query.
Text-to-Cypher	If users are asking questions that require information housed in a graph database, accessible via Cypher.	This uses an LLM to transform user input into a Cypher query.
Self Query	If users are asking questions that are better answered by fetching documents based on metadata rather than similarity with the text.	This uses an LLM to transform user input into two things: (1) a string to look up semantically, (2) a metadata filter to go along with it. This is useful because oftentimes questions are about the METADATA of documents (not the content itself).