**🔧 Step-by-Step Setup**

**✅ Step 1: Create SQLAlchemy URI for SQL Server**

LangChain uses SQLAlchemy to connect to the database.

Here’s how to write the URI for SQL Server:

python

CopyEdit

import urllib

params = urllib.parse.quote\_plus(

"DRIVER={ODBC Driver 17 for SQL Server};"

"SERVER=localhost;"

"DATABASE=mydb;"

"UID=sa;"

"PWD=yourPassword123"

)

connection\_uri = f"mssql+pyodbc:///?odbc\_connect={params}"

**✅ Step 2: Set up LangChain SQL Agent**

python

CopyEdit

from langchain\_experimental.sql import SQLDatabaseChain

from langchain.chat\_models import ChatOpenAI

from langchain.sql\_database import SQLDatabase

# Connect to SQL Server

db = SQLDatabase.from\_uri(connection\_uri)

# Use OpenAI (you can also use local models here)

llm = ChatOpenAI(temperature=0, model="gpt-3.5-turbo") # or "gpt-4"

# Create the chain

db\_chain = SQLDatabaseChain.from\_llm(llm, db, verbose=True)

# Ask a question

question = "What is the total expense for Project Alpha in Q1?"

response = db\_chain.run(question)

print(response)

**✅ Sample Output**

If your SQL Server has a table expenses like this:

| **id** | **project** | **amount** | **quarter** |
| --- | --- | --- | --- |
| 1 | Project Alpha | 1000 | Q1 |
| 2 | Project Beta | 2000 | Q1 |
| 3 | Project Alpha | 1500 | Q1 |

It might generate and run:

sql

CopyEdit

SELECT SUM(amount) FROM expenses WHERE project = 'Project Alpha' AND quarter = 'Q1';

And return:

“The total expense for Project Alpha in Q1 is 2500.”

**⚡ What if I don’t want to use OpenAI?**

You can plug in **local models** like phi, mistral, or Qwen via Hugging Face like this:

python

CopyEdit

from langchain.llms import HuggingFacePipeline

from transformers import AutoModelForCausalLM, AutoTokenizer, pipeline

model\_name = "microsoft/phi-2"

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

model = AutoModelForCausalLM.from\_pretrained(model\_name)

pipe = pipeline("text-generation", model=model, tokenizer=tokenizer, max\_new\_tokens=256)

local\_llm = HuggingFacePipeline(pipeline=pipe)

db\_chain = SQLDatabaseChain.from\_llm(local\_llm, db, verbose=True)

(Note: local models might need better prompt tuning and are slower on CPU.)

**🔐 Security Tip**

Always add logic to:

* ❌ Prevent destructive queries (DROP, DELETE)
* ✅ Validate SQL before executing
* ✅ Log query and user prompts for auditing

**✅ Summary**

| **Component** | **What it does** |
| --- | --- |
| LangChain | Manages prompt + SQL generation |
| SQLDatabaseChain | Bridges LLM ↔ SQL DB |
| SQLAlchemy URI | Connects to SQL Server |
| ChatOpenAI / LLM | Generates SQL queries |
| pyodbc | Enables SQL Server connection |

Would you like:

* A working Python script template with this setup?
* Or a version that runs with a local LLM (like Phi-3) and no OpenAI dependency?

Let me know how you'd like to proceed!