LangChain Expression Language Cookbook

Querying a SQL DB

Querying a SQL DB

We can replicate our SQLDatabaseChain with Runnables.

%pip install –upgrade –quiet langchain langchain-openai

```
from langchain_core.prompts import ChatPromptTemplate

template = """Based on the table schema below, write a SQL query that would answer the user's question:
{schema}

Question: {question}
SQL Query:"""
prompt = ChatPromptTemplate.from_template(template)
```

```
from langchain_community.utilities import SQLDatabase
```

We’ll need the Chinook sample DB for this example. There’s many places to download it from, e.g. https://database.guide/2-sample-databases-sqlite/

```
db =
SQLDatabase.from_uri("sqlite:///./Chinook.db")
```

```
def get_schema(_):
    return db.get_table_info()
```

```
def run_query(query):
    return db.run(query)
```

```
from langchain_core.output_parsers import
StrOutputParser
from langchain_core.runnables import
RunnablePassthrough
from langchain_openai import ChatOpenAI

model = ChatOpenAI()

sql_response = (

RunnablePassthrough.assign(schema=get_schema)
    | prompt
    | model.bind(stop=["\nSQLResult:"])
    | StrOutputParser()
)
```

```
sql_response.invoke({"question": "How many
employees are there?"})
```

```
'SELECT COUNT(*) FROM Employee'
```

```
template = """Based on the table schema
below, question, sql query, and sql response,
write a natural language response:
{schema}

Question: {question}
SQL Query: {query}
SQL Response: {response}"""
prompt_response =
ChatPromptTemplate.from_template(template)
```

```
full_chain = (

RunnablePassthrough.assign(query=sql_response)..
        schema=get_schema,
        response=lambda x: db.run(x["query"]),
    )
    | prompt_response
    | model
)
```

```
full_chain.invoke({"question": "How many
employees are there?"})
```

```
AIMessage(content='There are 8 employees.',
additional_kwargs={}, example=False)
```