

AI Assignment — Retail Analytics Copilot (DSPy + LangGraph)

Author: Yousef Roshdy Mohamed Fathalla Abdou

Date: 27/11/2025

1. Overview

This project implements a hybrid agent for answering analytical questions over the Northwind dataset using:

1. **SQL grounding** for database-answerable questions
2. **RAG (TF-IDF retrieval)** for policy and documentation answers

Gross margin is computed as **30% of UnitPrice** (COGS = 70%).

The system accepts a batch of questions in JSONL format and dynamically routes each as:

- SQL-only
- RAG-only
- Hybrid (retrieval + SQL + synthesis)

Outputs follow the assignment contract:

```
{  
  "id": "...",  
  "final_answer": <matches format_hint>,  
  "sql": "<last executed SQL>",  
  "confidence": float,  
  "explanation": "... <= 2 sentences",  
  "citations": [...]  
}
```

Execution uses LangGraph for control flow, DSPy modules for prompting robustness, and **Ollama + Phi-3.5 (3.8B Q4_K_M)** for local inference.

2. System Architecture

The system is implemented as a LangGraph pipeline with the following nodes:

Router Module

Determines route: "sql", "rag", or "hybrid".

Retriever Module

Retrieves relevant text chunks from:

- Product policy
- Marketing calendar
- KPI definitions
- Catalog snapshot

NL2SQL Module

Generates SQL conditioned on:

- question
- schema description
- retrieved context
- JSON constraints

SQL Execution Module

Executes SQL safely against northwind.db and returns rows or errors.

Synthesizer Module

Combines SQL rows + retrieved text + question to produce:

- final_answer
- explanation (≤ 2 sentences)
- citations
- confidence

Repair Loop

Retries SQL or synthesis on error (max 2 attempts).

3. Retrieval (RAG)

TF-IDF retriever loads .md documents, splits them into paragraph-level chunks, computes TF-IDF embeddings, and returns top-k matching chunks with:

- text
- document source
- chunk ID

- relevance score
-

4. SQL Execution Layer

SQLite tool provides:

- execute_sql(sql: str)
 - safe read-only SQL
 - automatic row-to-dict conversion
 - schema reflection for NL2SQL
-

5. DSPy Modules

Defined in agent/dspy_signatures.py.

RouterModule

- Input: question + format_hint
- Output: route string

NL2SQLModule

- Converts natural language to SQL using schema and retrieved text

SynthModule

- Combines all components to produce answer + explanation + citations
 - Handles format coercion and confidence scoring
-

6. LangGraph Execution

Graph nodes:

- **router** → choose path
- **retriever** → RAG chunks
- **nl2sql** → SQL generation
- **sql_exec** → SQLite execution
- **synthesizer** → final answer
- **repair** → retry logic

7. Repair Loop

Error handling:

Error	Action
SQL syntax error	regenerate SQL
Wrong output format	regenerate via SynthModule
Missing citation	retry synthesis

Max retries: **2**.

8. Logging & Tracing

Each node logs a JSON snapshot:

- task id
- route
- attempts
- sql_query
- sql_error
- synth_error

This provides full traceability of the agent's reasoning.

9. Local Model Integration

- Ollama for local model serving
 - Phi-3.5 Mini (Q4_K_M) for CPU-compatible inference
-

10. Limitations & Tradeoffs

Model Limitations

- Weak schema-aware SQL reasoning
- Hallucinated tables/columns
- Invalid JOIN paths

- Limited multi-step reasoning

RAG Limitations

- TF-IDF is lexical, not semantic
- Sometimes retrieves irrelevant chunks

Hybrid Limitations

- SQL errors propagate into the synthesizer
 - Some explanations/citations return empty due to upstream failure
-

11. Suggested Improvement

I attempted to mitigate SQL hallucinations by explicitly passing the full Northwind database schema into the NL2SQL module. The Phi-3.5 Mini model still produced incorrect SQL, confirming its limited text-to-SQL capability.

To improve SQL reliability:

1. **Use a Specialized SQL Model or Fine-tune Phi-3.5**
 - fine-tune on Northwind text-to-SQL pairs
 - or use a chat-to-DB model optimized for SQL grounding
 2. **Replace TF-IDF with Embedding Retriever**
 - MiniLM or BGE-small improves semantic relevance
 3. **BERT-based SQL Task Classifier**
 - predicts task type → template-guided generation
 4. **Entity Tagging (NER)**
 - extracts dates, categories, metrics, time windows
 - provides structured constraints for SQL generation
-

12. Conclusion

This project delivers a functional hybrid analytics agent combining **RAG + SQL**, LangGraph, DSPy modules, and local inference through Ollama.

While SQL generation remains the primary weakness due to model limitations, the system is modular and can be upgraded with stronger SQL models, semantic retrieval, BERT classification, and entity-extraction pipelines.