ARCHITECTURE DOCUMENT

Retail Insights Assistant

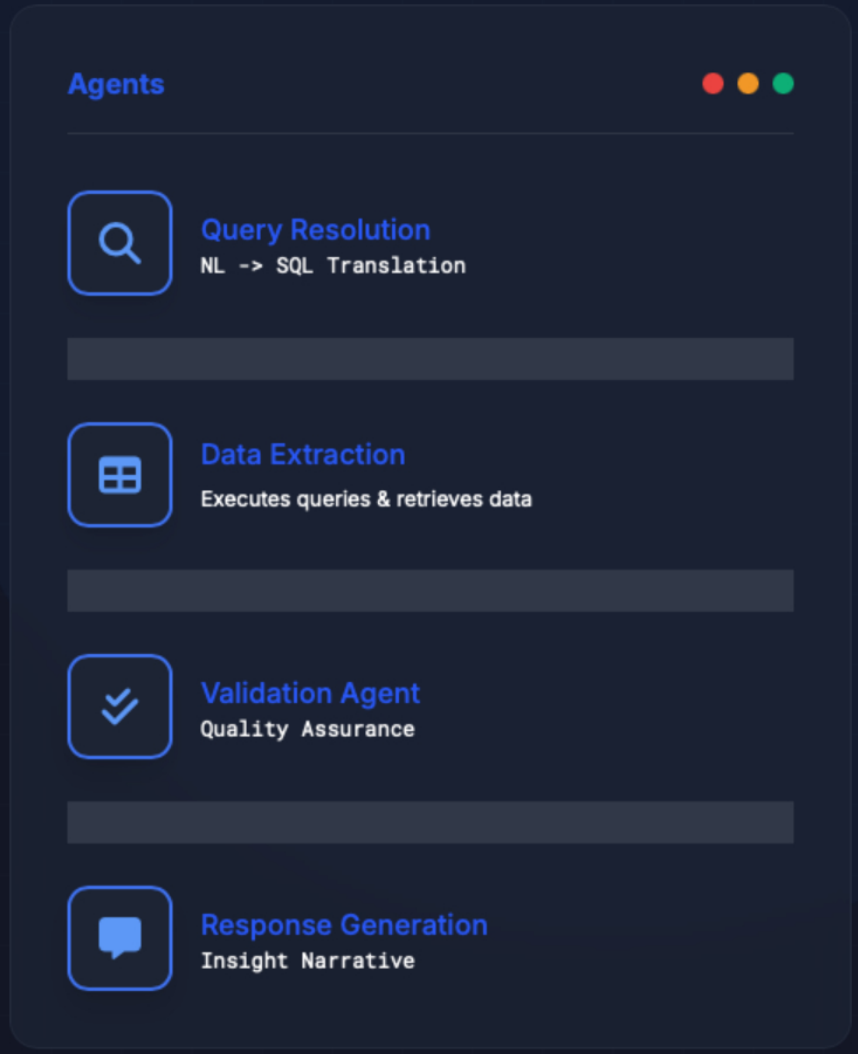
> A Multi-Agent GenAI System for automated sales analytics, SQL generation, and business intelligence.

LangGraph Pipeline
4-Agent Orchestration with GPT-4

DuckDB Engine
High-performance local OLAP

Streamlit UI
Real-time status & visualization

Auto-Validation
Self-correcting data integrity



PROJECT OVERVIEW

Executive Summary

01_SYSTEM

What is it?

A Multi-agent GenAI pipeline designed for automated sales analytics.

- → Flow: Query → Extract → Validate → Respond
- → Converts natural language to SQL & executes on DuckDB

02_VALUE

Why it matters

Delivers reliable analytics where standard LLMs often hallucinate.

- ✓ Robust multi-level fallbacks & validation agents
- ✓ Business-friendly narratives with data limitations exposed

03_FEATURES

Key Features

Complete end-to-end solution from UI to Database.

- Streamlit UI with real-time agent status monitoring
- Smart visualization engine (Auto-chart selection)
- Cloud-ready & containerized deployment

04_IMPACT

Outcomes

Empowers stakeholders with instant data access.

- 10x faster ad-hoc analysis compared to manual SQL
- Consistent answers reducing operational risk

SYSTEM ARCHITECTURE

End-to-End Data Flow

ORCHESTRATION LAYER

LangGraph Orchestrator orchestrator.py

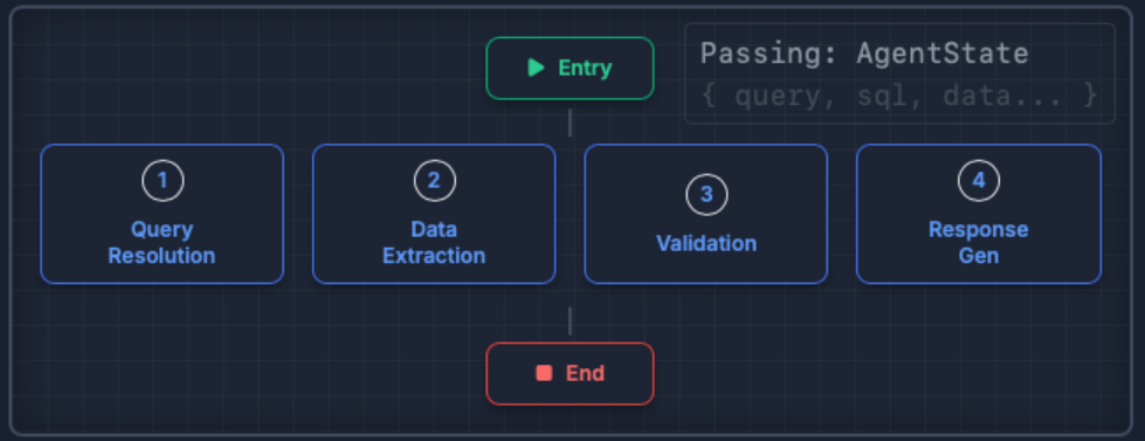
</> Lines of Code: 166

Initialization & Workflow Graph

`class RetailInsightsOrchestrator` StateGraph Compiled

API KEY	MODEL	DATA PATH
st.session_state	gpt-4-turbo-preview	Sales Dataset/

SEQUENTIAL STATEGRAPH PIPELINE



AGENTS.PY COMPONENT BREAKDOWN

Agent System Overview

01

Input Processor

QueryResolutionAgent

Converts natural language into executable SQL using deep database context.

- Injects full schema & sample values into context window
- Handles DuckDB specifics (e.g., `YEAR()` vs `DATE_PART`)
- Generates complex YoY logic via self-joins

02

Executor

DataExtractionAgent

Executes queries and retrieves data with robust error handling.

- Safely executes SQL on local DuckDB engine
- **Smart Fallback:** If SQL fails, auto-fetches relevant summaries
- Attaches dataset metadata (date ranges, available quarters)

03

Quality Control

ValidationAgent

Ensures data relevance and structural integrity before presentation.

- Structural checks for missing keys or empty payloads
- LLM-based semantic review: "Does this answer the question?"
- Assigns confidence score (0.0 - 1.0) to guide UI warnings

04

Narrator

ResponseGenerationAgent

Crafts business-friendly insights and executive summaries.

- Highlights specific numbers, growth rates, and key trends
- Explains data limitations (e.g., "Q4 data is partial")
- Adapts tone for executive stakeholders (concise, direct)

DATA LAYER

Data Processing

DuckDB Engine

Engine & Schema Architecture

DuckDB OLAP Engine

Embedded analytics engine utilizing a persistent database file. Handles high-performance SQL queries directly on tabular data.

`retail_data.duckdb` `Auto-Load on Cold Start`

1 **amazon_sales** — Amazon Sale Report.csv

Order ID (PK)	Date [DATE]	Amount [DOUBLE]
Category	Status	ship-state

2 **inventory** — Sale Report.csv

Category	Size	Color	Stock [INT]

3 **international_sales** — International sale Report.csv

CUSTOMER	Months	GROSS AMT

Core Methods & Cloud Logic

Key Methods

Primary interface for Agent interaction

Execute raw SQL & return DataFrame

```
execute_query(sql: str) -> pd.DataFrame
```

Get schema + samples for LLM context

```
get_table_context(table_name: str)
```

Aggregate cross-table summary stats

```
get_summary_statistics()
```

Fallback data generators

```
get_top_categories(n)
```

```
get_regional_performance()
```

Cloud Deployment Strategy

Handling ephemeral file systems (Streamlit Cloud)

Persistence Check

Checks `information_schema.tables` on init to avoid redundant re-loading.

FOUNDATION & DEPENDENCIES

Technology Stack

All dependencies pinned for reproducibility

Core Logic
Orchestration & LLM

LangGraph	v1.0.1
LangChain	v0.3.27
OpenAI (SDK)	v2.9.0
LangChain-Core	v0.3.80
LC-OpenAI	v0.3.35

Data & Viz
Processing & Analytics

DuckDB	v1.4.4
Pandas	v2.2.3
NumPy	v2.2.2
Plotly	v6.5.2

App & Utils
Frontend & Utilities

Streamlit	v1.49.1
Pydantic	v2.11.7
tiktoken	v0.9.0
python-dotenv	v1.0.1

PIPELINE ARCHITECTURE

Data Flow Processes

Q&A Logic | Summary Logic

Conversational Q&A Mode

Standard complex query pipeline with multi-agent orchestration

1. **User Input**
   Types natural language question in chat
2. **Agent 1**
   Resolves intent & generates SQL
   `Schema Context`
3. **Agent 2**
   Executes SQL on DuckDB
   Fallback Tiers
4. **Agent 3**
   Validates structure & relevance
   `Confidence Score`
5. **Agent 4**
   Crafts business response
6. **UI Render**
   `SQL` `Chart` `Table`

Executive Summary Mode

Pre-defined aggregation pipeline for instant dashboard generation

1. **Trigger**
   User clicks "Generate Summary"
2. **Agent 2 (Aggregator)**
   Runs multiple parallel queries
   `Stats` `Top Categories` `Regional`
3. **Agent 4 (Writer)**
   Synthesizes data into executive narrative
4. **Dashboard**
   Renders summary text + interactive charts
   `Multi-view`

ENVIRONMENT SETUP

Deployment Options

Python 3.9+ Ephemeral DB

Local Dev

For development & testing

QUICK START

```
# Install dependencies
$ pip install -r requirements.txt
# Run Application
$ streamlit run app.py
```

CONFIGURATION

- ✓ Enter API key in sidebar UI
- ✓ Hot-reloading enabled
- ✓ DuckDB creates local file

Cloud Hosted

Streamlit Community Cloud

ZERO OPS

```
1. Push code to GitHub
2. Connect repo on share.streamlit.io
3. App deploys automatically
```

KEY FEATURES

- ✓ DuckDB rebuilds on cold start
- ✓ No secrets file management needed
- ✓ API Key entered by user at runtime

Docker Prod

Containerized Deployment

SCALABLE

```
FROM python:3.9-slim
WORKDIR /app
COPY . .
RUN pip install ...
EXPOSE 8501
CMD ["streamlit", "run", "app.py"]
```

PRODUCTION SPEC

- ✓ Ephemeral DuckDB filesystem
- ✓ Bundled CSVs for auto-load
- ✓ Ready for K8s / ECS

TARGET: 100GB+ DATA

Scalability Roadmap

Infrastructure Evolution

Single Node → Distributed Cluster

Data Layer

Embedded — Cloud Warehouse

MIGRATE

- Snowflake / BigQuery
- Year Partitioning
- Parquet

Processing

Pandas/Single — Distributed ETL

SCALE

- PySpark / Dask
- Apache Airflow

Infrastructure

Streamlit Cloud — Container Cluster

DEPLOY

- Docker
- K8s HPA
- Redis Cache

LLM Optimization

Direct API — Smart Routing

OPTIMIZE

- Model Routing
- 3.5 GPT-3.5 vs 4
- Semantic Cache

Retrieval

SQL Only — Hybrid Search

ENHANCE

- RAG
- FAISS / Pinecone
- Summaries

Monitoring

Basic Logging — Observability

TRACK

- Prometheus
- Grafana
- Alerts