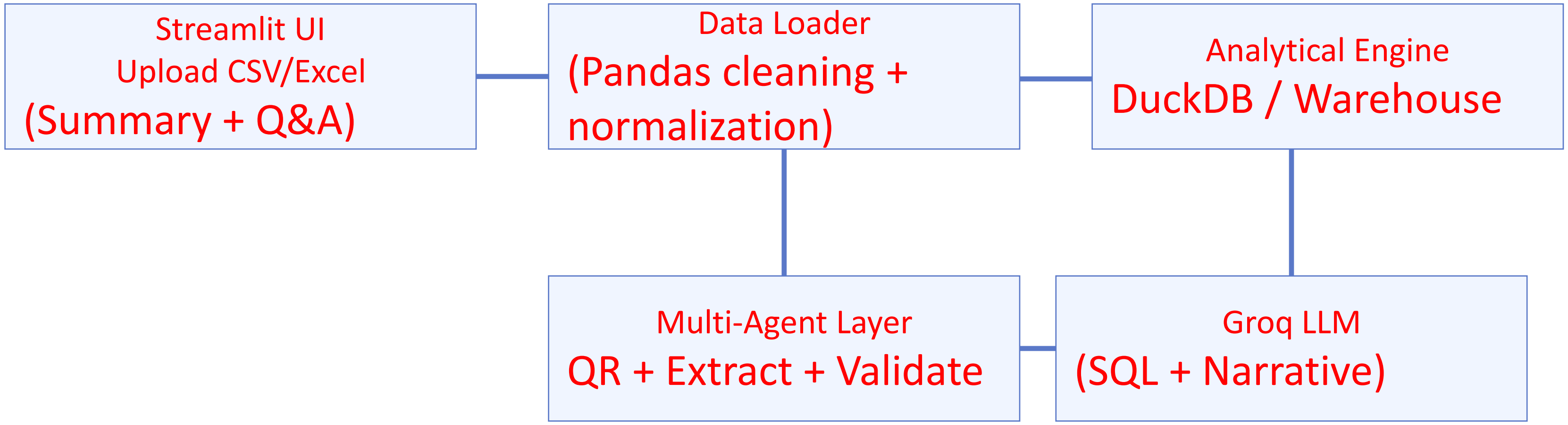
Retail Insights Assistant (GenAI + Multi-Agent System)

Blend360 GenAI Interview Assignment

Presentation by Vaishnavi Kukkala

10/01/2026

System Architecture & Data Flow



LLM Integration Strategy

- LLM is used for:
  - Intent understanding (user question → analytical task)
  - Schema-aware SQL generation
  - Executive narrative summaries / explanation of query output
  - Used a Router pattern to decide whether to send the task to the "Summarization" logic or the "Q&A" logic.
- LLM is NOT used for:
  - Large scans, joins, aggregations, or metric computation
- Prompt grounding:
  - Inject DuckDB schema (PRAGMA table_info) to avoid hallucinations
  - Include recent conversation context (last N turns)
- Safety controls:
  - Allow SELECT-only queries
  - Block multi-statement SQL; validation gate; retry loop on errors

Multi-Agent Architecture

- **Agent Roles & Interaction:**
- **Query Resolution Agent:** Interprets natural language and generates schema-aware DuckDB SQL.
- **Data Extraction Agent:** Executes optimized SQL directly on the data engine (DuckDB for demo, BigQuery/Snowflake for scale).
- **Validation Agent:** Checks for empty results, schema mismatches, and potential SQL injection before passing data to the final narrative step.

- **Self-Correction & Fallback:**
- **Retry Loop:** If the Data Extraction Agent returns a SQL error, the error log is passed back to the Query Resolution Agent for automatic refinement.
- **Graceful Degradation:** If confidence is low or the query is too complex, the system falls back to a "Clarification Mode," asking the user for missing details instead of guessing.

Summary

- Runs predefined SQL aggregates (Top categories, Top states, Order status split)
- Builds structured summary blocks from SQL outputs
- LLM converts metrics → executive narrative
- Outputs business recommendations grounded in retrieved data

Example Query → Response Pipeline

•**User Input:** *"Which category had the highest revenue in the South region last month?“*

•**Step 1: Intent & Mapping:** The system normalizes "South" to the standard ship_state values and "last month" to a specific date range (e.g., 2024-12-01 to 2024-12-31).
•**Step 2: SQL Generation:** The Resolution Agent writes: SELECT category, SUM(revenue) FROM sales WHERE region = 'South' AND date BETWEEN '...' AND '...' GROUP BY 1 ORDER BY 2 DESC LIMIT 1;
•**Step 3: Validation:** The Validation Agent confirms the query only uses SELECT, returns a non-empty set, and doesn't exceed memory limits.
•**Step 4: Business Insight:** The LLM translates the table result into: *"The 'Home Decor' category led the South region in December, contributing $45k in revenue."*

100GB+ Scale Design

- **Storage Strategy:**
- **Data Lake:** Store raw data in **S3 or Azure Data Lake** using **Parquet** format. Parquet's columnar storage is essential for 100GB+ because it allows the system to read only the specific columns needed (e.g., just revenue and date).
- **Partition Pruning:** Organize data by date or region. This ensures that a query for "last month" only scans 1/12th of the data, not the full 100GB.

- **Distributed Compute:**
- Replace local DuckDB with a distributed engine like **BigQuery, Snowflake, or Databricks (Spark SQL)**.
- Use **SQL Pushdown**: The agents send the SQL to the warehouse, and only the *small* result set (not the 100GB) is sent back to the LLM.

- **Hybrid Retrieval (RAG + SQL):**
- **Vector Indexing:** For unstructured data (like product catalogs or manuals), use a Vector DB (Pinecone/Milvus) for similarity search alongside the structured SQL queries.

Cost & Performance Considerations

- **Latency Control:**
  - **Semantic Caching:** Use Redis to cache common query results. If two users ask for "Top sales," the system returns the cached answer in milliseconds without re-running SQL or LLM calls.
  - **Asynchronous Processing:** Long-running summaries should run in the background with a "Status: Thinking" indicator to keep the UI responsive.
- **Cost Optimization:**
  - **Model Tiering:** Use a cheaper model (like Llama 8B) for simple validation tasks and reserve the expensive 70B model for complex SQL reasoning.
  - **Token Management:** Only inject the relevant schema for the columns requested, rather than the entire database schema, to reduce prompt tokens.

Thank You for your time!