CASE STUDY SUBMISSION

Automated LLM Report Generation: A Multi-Agent Architecture

Orchestrating deterministic code execution and probabilistic reasoning to analyze BMW sales data.

Presented by: Colton Tang

github.com/Calab222/AI-Multi-Agent-BMW-Sales

The Approach: Multi-Agent Orchestration

Instead of a single "God Model" prompting approach, the system decomposes the problem into specialized agents based on the nature of the task.

Pandas Agent

Quantitative & Deterministic

- Generates Python Code
- Sandbox Execution
- Matplotlib Plotting

SYNTHESIS

RAG Agent

Qualitative & Contextual

- Vector Search (ChromaDB)
- Feature/Spec Retrieval
- Semantic Understanding

Design Choice: Separation of Concerns

The "Hallucination" Trap

LLMs are excellent at writing but terrible at mental arithmetic. Asking GPT to calculate "sum of sales where region is Europe" often results in calculation errors.

Architecture Decision

Code-Interpreter Pattern
We delegate math to Python (Pandas). The LLM writes the query, but the CPU executes it. This ensures **100% mathematical accuracy**.

Async Parallelism
Agents run asynchronously using `asyncio.gather`. High-level stats and deep-dive research happen simultaneously to reduce latency.

Design Choice: Engineering & Reproducibility

The "Self-Healing" Data Pipeline

Step 1

Check Vector DB

Is `chroma_db` empty?

Step 2

Auto-Ingestion

Parse Excel → Embed → Store

Step 3

Ready to Serve

Fast Retrieval

```
if self.collection.count() == 0:
  print("Starting ingestion...")
  self._ingest_data()
# Ensures repo is lightweight but functional immediately
```

Evaluation Framework: Robustness & Grounding

Dimension 1: Robustness

MECHANISM

Sandboxed Execution

The `PandasAgent` wraps generated Python code in a `try...except` block using `exec()`.

VALIDATION

Metric: **Execution Success Rate**. Ensures code is syntactically correct and column names match the schema before returning results.

Dimension 2: Grounding

MECHANISM

Retrieval Gate / Empty-State Detection

Explicit check: `if not results['documents']`.

VALIDATION

Metric: **Retrieval Hit Rate**. Prevents hallucination by verifying that actual documents were found before attempting to answer.

Evaluation Framework: Output Quality

Dimension 3: Schema Adherence

Mechanism: Pydantic Models & Structured Parsing.

Validation: Ensures the final output is not just unstructured text, but valid JSON containing separate fields for `code`, `insight`, and `image_path`. This guarantees the frontend renders correctly.

USER-FACING METRICS

Visuals

Does the report generate a plot when data is suitable?

Pass: Matplotlib

Cohesion

Does synthesis merge quantitative stats with qualitative context?

Pass: Context Injection

Latency

Is the report generated within a reasonable timeframe?

Pass: Async/Await

Roadmap to Production

Dockerization
Containerize the Python API and ChromaDB to ensure environment consistency across deployments.

LLM-as-a-Judge
Implement a third agent that critiques the final report for tone and accuracy before showing it to the user.

Feedback Loop
Allow users to "Edit" the prompt or chart settings, feeding that preference back into the prompt history.

External Data
Expand RAG ingestion to handle PDF market reports and Competitor Analysis documents.

Thank You

Ready for Q&A