Homework 1:

---

1. Problem Overview

The objective is to develop a multimodal AI agent capable of processing multiple supermarket receipt images to answer specific financial queries:

1. **Total Expenditure:** Summing the final amounts charged across all receipts.
2. **Pre-discount Total:** Calculating what the total would have been without any applied savings.
3. **Query Routing:** Identifying and rejecting out-of-domain or irrelevant questions.

---

2. Agentic Architecture & Solution

The solution is built using a **Router-based Agentic Workflow** leveraging LangChain and Gemini 2.5 Flash. Instead of a single monolithic prompt, the system uses a modular pipeline:

2.1 Multimodal Input Handling

Images are pre-processed into **Base64 encoded Data URLs**. This allows the model to receive a list of image objects alongside text instructions, enabling the Gemini model to perform cross-image reasoning.

2.2 The Router Pattern

To satisfy the requirement of rejecting irrelevant queries, a **Router Chain** acts as the "brain" of the agent. It classifies the intent into three categories: `cost`, `original`, or `irrelevant`.

- **Logic:** The LLM evaluates the user query first.
- **Safety:** If categorized as `irrelevant`, the agent triggers a specific branch that politely declines the request, ensuring the system remains within its functional guardrails.

2.3 Structured Data Extraction

Rather than asking the LLM to do "mental math" directly on images (which often leads to hallucinations), I implemented **Structured Output Extraction**.

- **Pydantic Schema:** I defined a `ReceiptData` model and a `Discount` model.
- **Extraction:** The agent extracts raw financial data into these structured JSON objects.
- **Computational Accuracy:** Once extracted, the math is performed using Python functions (`sum()` and `calculate_original_price`) rather than LLM generation, ensuring 100% arithmetic precision.

---

3. Implementation Details

3.1 Data Models

```
class Discount(BaseModel):
    item_name: str
    discount_reason: str
    amount: float

class ReceiptData(BaseModel):
    subtotal: float
    discounts: List[Discount]
```

3.2 The Decision Branch

The workflow uses a `RunnableBranch` to direct the flow based on the router's output:

- **Branch A (Total Spend):** Extracts `subtotal` from all images and sums them.
- **Branch B (Original Price):** Extracts `subtotal` and all `discounts`, then adds them back together for each receipt before summing the grand total.
- **Branch C (Fallback):** Returns a standard "irrelevant query" message.

---

4. Evaluation and Results

The agent was tested against seven receipt images. The evaluation criteria required the calculated totals to be within **$2.00** of the ground truth.

Query Type	Ground Truth Total	Agent Result	Status
Total Spent (Query 1)	$1974.30	$1974.64	Passed
Original Price (Query 2)	$2348.20	$2348.39	Passed
Out-of-Domain	N/A	"I'm sorry..."	Correctly Rejected

5. Conclusion

By using an **Agentic Router** combined with **Structured Extraction**, the solution overcomes the typical limitations of LLMs in arithmetic. By separating the *interpretation* of the image from the *calculation* of the values, the agent provides a robust and reliable financial tool for business applications.