GenAI Agent–Driven ETL Pipeline (PoC)

Gulshan Gupta

January 21, 2026

1. Overview

This project demonstrates a **Generative AI-powered ETL agent** that can:

- Understand input data schemas.
- Reason about necessary transformations.
- Orchestrate ETL steps using Python, SQL, and APIs.

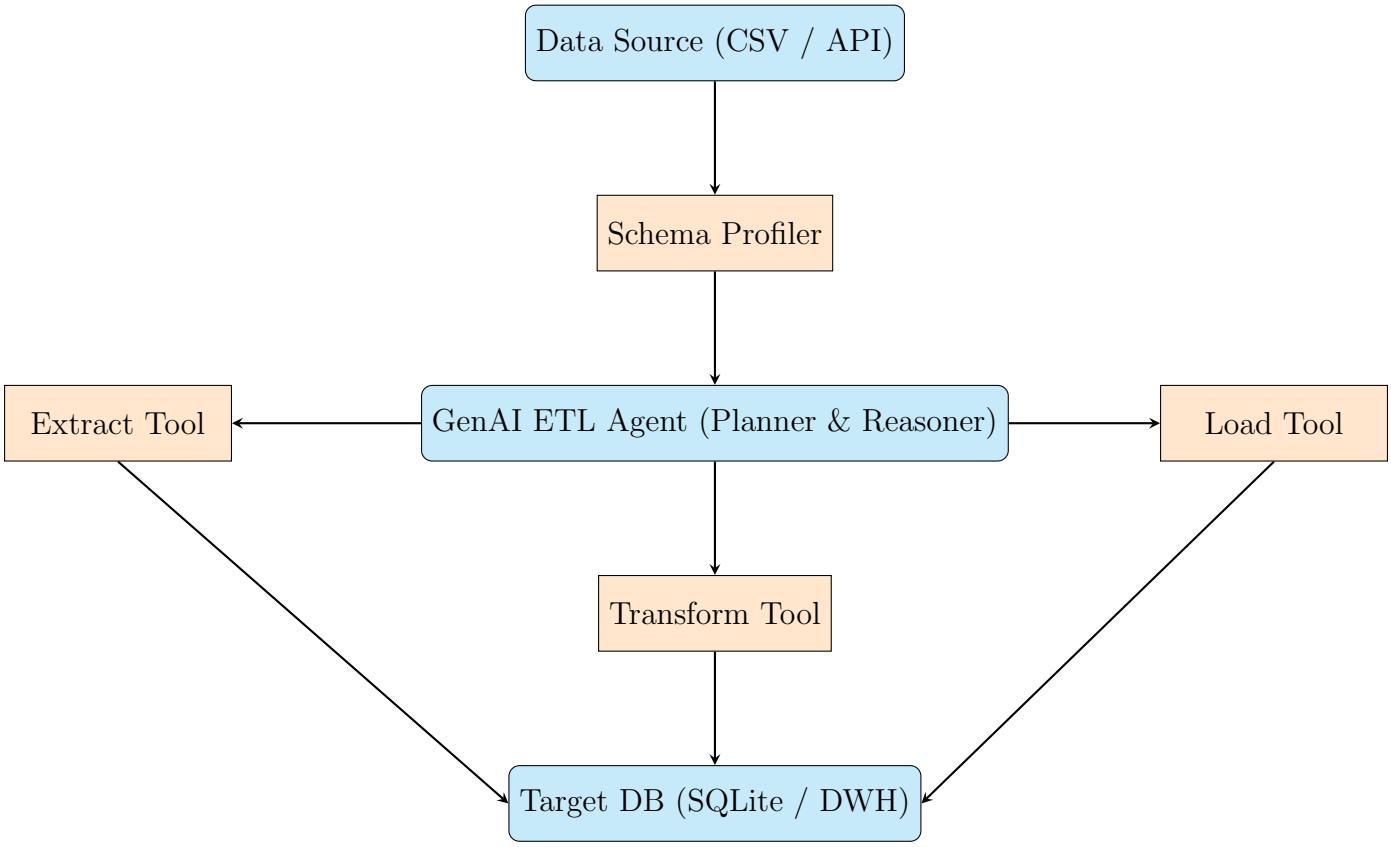
Goal: Proof of Concept to show AI agents improving adaptability, automation, and intelligence in data pipelines.

2. Problem Scope & Assumptions

Aspect	**Details**
Source	CSV files (e.g., sales transaction data)
Target	SQLite database (conceptual data warehouse)
Scale	Small-to-medium datasets (PoC)
Focus	AI-driven design, reasoning, and decision-making for ETL

3. High-Level Architecture

The GenAI agent acts as the **brain** of the system, orchestrating ETL tools.



4. Role of GenAI / LLMs

The GenAI agent is a **planner and decision-maker**:

- **Schema Understanding:** Reads column names, types, and sample rows.
- **Transformation Planning:** Handles type casting, missing values, deduplication.
- **Tool Orchestration:** Generates Python/SQL code and calls ETL tools.
- **Error Handling:** Detects issues and proposes fixes automatically.

5. Agent Reasoning & Planning Flow

The agent follows an **Observe** → **Think** → **Plan** → **Act** → **Reflect** loop:

1. **Observe:** Inspect CSV columns, data types, sample rows.
2. **Think:** Identify issues.
   Example: `amount` should be numeric but contains strings.
   Example: `order_date` has inconsistent formats.
3. **Plan:** Generate a JSON-based ETL plan:

```
{
  "steps": [
    "Read CSV",
    "Validate schema",
    "Cast data types",
    "Remove duplicates",
    "Fill missing values",
    "Load into SQLite"
  ]
}
```

4. **Act:** Execute ETL steps using Python and SQL.
5. **Reflect:** If errors occur, revise plan and retry.

6. ETL Flow Example

Input Data (‘sales_data.csv‘):

order_id	customer_id	order_date	amount
1001	C001	2024/01/05	1200
1002	NaN	05-01-2024	900
1003	C003	2024-01-06	abc

Agent Actions:

- Extract: Read CSV using Pandas.
- Transform:
  - Convert `order_date` to ISO format (YYYY-MM-DD)
  - Convert `amount` to float; flag/drop invalid values like “abc”
  - Fill missing `customer_id` with “UNKNOWN”
  - Remove duplicate `order_id`s
- Load: Write cleaned data to SQLite table `fact_sales`.

7. Conceptual Tooling Stack

Layer	Tool / Tech
LLM (Brain)	Gemini 1.5 Flash / GPT-4 / Claude
Orchestration	Python Scripts (Custom Agent Logic)
ETL Execution	Pandas (Python), SQL
Storage	SQLite (Local DB)
Monitoring	Console Logs + Metrics

8. Trade-offs & Limitations

Advantages:

- Adaptive pipelines: handles schema drift automatically.
- Less hardcoding: fewer brittle if/else conditions.
- Fast onboarding: quickly profiles new datasets.

Limitations:

- Consistency: LLM reasoning can be non-deterministic.
- Guardrails: need validation to prevent hallucinated transformations.
- Cost & Latency: slower and more expensive than pre-compiled ETL.
- Not real-time: best suited for batch processing.

9. Adaptation Scenarios

- Schema Changes: New columns detected automatically.
- Data Quality Issues: Spike in nulls triggers validation suggestions.
- New Sources: Profiles and aligns new datasets to warehouse schema.

10. How to Run

1. Install dependencies:

```
pip install pandas sqlalchemy google-genai python-dotenv
```

2. Add your API key in ‘.env‘:

```
GOOGLE_API_KEY=your_key_here
```

3. Run the pipeline:

```
python main.py
```

4. Verify loaded data:

```
python verify.py
```

11. Philosophy

> ”The LLM does not replace ETL tools — it orchestrates them intelligently.
> The real power lies in reasoning, not raw data processing.”