Final Objective

Build a system where a user can type "a story about a lonely robot in space" and find relevant books, even if those exact keywords are not present in the title or metadata.

Project 1 Objective

Build a robust data pipeline to extract, clean, and store book information (titles, descriptions, genres) to provide a high-quality dataset for an ML-powered semantic search system.

Current Role: Data Engineer

Recommended Tools(not mandatory. Feel free to explore others)

Lifecycle Step	Tool Recommended	Potential Sources
Generation	Raw Data Files / Web	OpenLibrary API, Kaggle CSVs, or Web Pages
Ingestion	`requests` , `pandas`	Reading CSVs or calling Book APIs
Storage	`sqlite3`	Relational DB (Tables: Books, Authors, Categories)
Transformation	`pandas`	Text normalization, removing duplicates, handling missing blurbs
Serving	`FastAPI`	API endpoints: `/books` , `/books/{isbn}` , `/sync`

Project Specs & Steps

1. Ingestion (The Collector)

- **The Task:** Read book data from multiple sources (e.g., a local CSV and a public API like OpenLibrary).
- **Challenge:** Dealing with inconsistent formats (one source uses "Title", another uses "book_name").

2. Transformation (The Refiner)

- **The Task:** Clean the book descriptions. Remove HTML tags, fix encoding (like `&`), and filter out books that don't have descriptions (as they are useless for semantic search).
- **Challenge:** Handling "Dirty Data"—some books might have "Description not available" which should be treated as null.

3. Storage (The Library)

- **The Task:** Create a SQLite schema to store the books.
  - *Table:* `books`
- **Learning Goal:** Moving from "flat files" (CSV) to "relational data" (SQL).

4. Serving (The Gateway)

- **The Task:** Develop a FastAPI service. This service provides a standard way for a Data Scientist to fetch the 1,000 most recent books for their embedding model.
- **Tool:** Use `Uvicorn` to run the FastAPI server locally.

Outcomes

1. **Curated Dataset:** A SQLite database with a `cleaned_books` table.
2. **Github Repository:** Organized structure
3. **README.md:** Detailed documentation on the data schema and how to trigger the ingestion script.

Scoring Policy

Category	Weightage	Description
Documentation	30%	Clear setup, API documentation, and data dictionary.
Code Quality	35%	Modularity (separate logic for DB vs. API) and error handling.
Data Management	35%	Data deduplication and schema integrity.

LLM Policy

1. Feel free to use an LLM provided each and every prompt is logged along with the response and the tool used in a markdown file (e.g., `logs/llm_usage.md`).

Future Steps

1. **Switch Role to Data Scientist:** Load the data from the FastAPI endpoint, generate embeddings using a Transformer model, and implement the "Lonely Robot" semantic search logic.