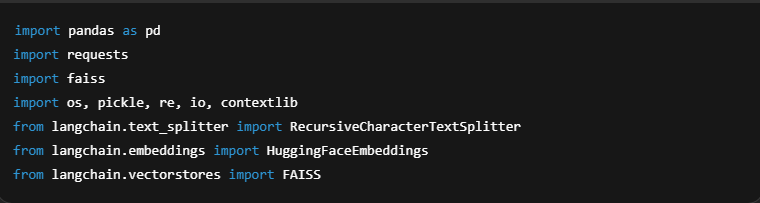
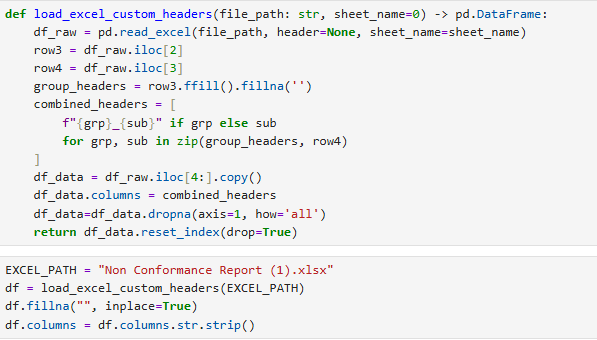
**HYBRID RAG + LOGIC QA SYSTEM**

**Block 1: Imports & Initial Setup**

What this does:

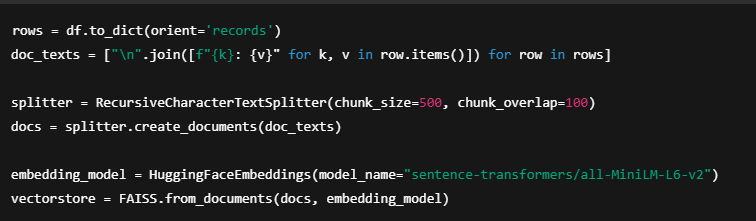
* Loads necessary libraries:
  + pandas: for logic queries over Excel/CSV
  + faiss, langchain.vectorstores: for semantic retrieval
  + requests: to call NVIDIA's hosted LLM
  + contextlib: captures printed output safely from exec()

**Block 2: Clean & Load Excel into a DataFrame**

**What this does:**

* Loads your document as a structured table
* Fills empty fields with "" for consistency
* Strips whitespace from column headers (prevents errors)

**Block 3: Embedding Text for RAG**



**What this does:**

* Converts your rows into human-readable text blocks
* Chunks large text blocks to prevent token overload in LLM
* Embeds them using a transformer model (MiniLM here)
* Stores embeddings in FAISS so you can do fast semantic search

Used for semantic questions like:  
*"What is this document about?"*  
*"What are common issues?"*

**Block 4: NVIDIA LLM API Call**

**What this does:**

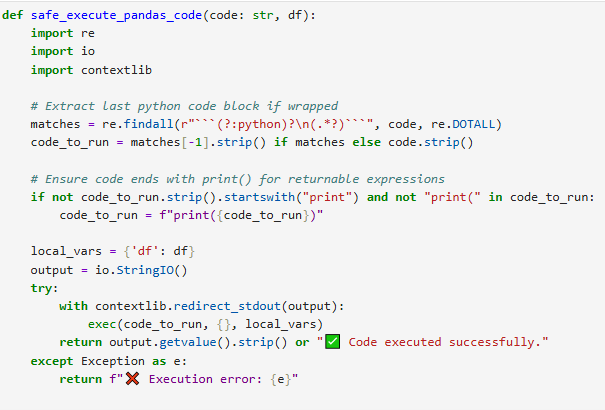
* Sends the prompt to NVIDIA's LLM using their OpenAI-compatible API
* Gets a natural language or Python code response back

*What LLM Does Internally*

When you send a prompt like:*“You are a Pandas expert. Based on this schema: ['Discipline', 'Status', ...], write code to answer: 'How many Civil projects are approved?'”*

The LLM:

1. Parses the natural question
2. Matches columns to fields (e.g., “Civil” → Discipline)
3. Generates working Python code like: df[(df['Discipline'] == 'Civil') & (df['Status'] == 'Approved')].shape[0]

**Block 5: Safe Dynamic Code Execution**

**What this does:**

* Extracts code block (from markdown or plain)
* Wraps it in print(...) if needed
* Captures printed result (list, number, DataFrame) into a string
* Returns the output to the user

This lets your LLM dynamically generate and run logic queries over real data

A computer code with text

AI-generated content may be incorrect.**Block 6: Semantic Retrieval (RAG)**

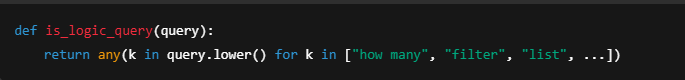
**What this does:**

* Finds 5 relevant chunks of text using vector similarity
* Constructs a prompt with both context and the question
* Sends it to the LLM to generate a full-text answer

*Useful for:*

* *Summaries*
* *Long, vague questions*
* *Non-tabular data*

**Block 7: Router (Logic vs Semantic)**



**What this does:**

* Classifies the user's query
  + If it contains logic terms → use Pandas + LLM code
  + Else → use semantic RAG

**Block 8: Main Execution LoopA computer screen shot of a computer code

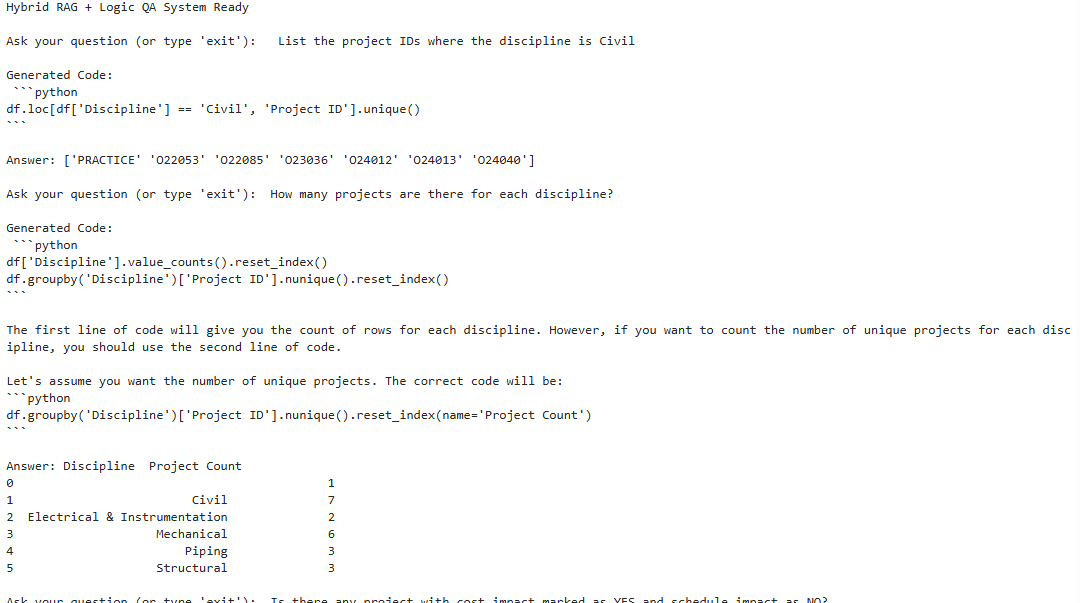
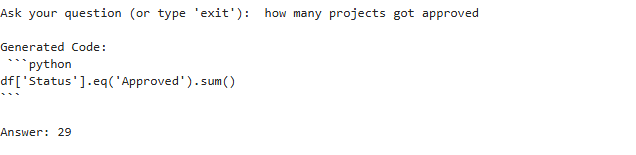
AI-generated content may be incorrect.**

**What this does:**

* Accepts user input
* Routes it to either:
  + Pandas logic path
  + Semantic RAG path
* Executes and displays the result

**What Makes This System Powerful**

| **Feature** | **Benefit** |
| --- | --- |
| LLM-generated code | Handles dynamic, varied questions without hardcoding |
| DataFrame execution | Ensures accuracy for filters, counts, groupings |
| Vector RAG | Enables contextual answers to vague or summary-style questions |
| Router | Adapts automatically to user intent |
| Safe execution | Prevents code crashes, captures outputs cleanly |

**Some Results:**

**A computer screen shot of text

AI-generated content may be incorrect.**