

# RAG Assignment: CSV-Based Question Answering System (FAISS + MiniLM + FLAN-T5)

A simple Retrieval-Augmented Generation (RAG) project that answers questions using a CSV dataset. It retrieves relevant records using embeddings + FAISS, then generates a grounded response using FLAN-T5.

## Project Overview

This project implements a complete RAG pipeline for tabular CSV data:

**CSV → Row-to-Text Chunks (+ Summary Chunks) → Embeddings (MiniLM) → FAISS Retrieval → LLM Answer (FLAN-T5)**

The system reduces hallucination by instructing the model to answer only from retrieved context.

## Dataset / Knowledge Source

- **Type:** CSV (structured tabular data)
- **Source:** Uploaded CSV dataset (customer demographics, income/loans, and car purchase information)
- **RAG usage:** Each row is converted into a text "document chunk". Additional summary chunks (Make-wise statistics) are created for analytical questions.

## Tech Stack

- **Google Colab / Python**
- **pandas, numpy** — data loading and preprocessing
- **sentence-transformers** — embeddings (MiniLM)
- **FAISS** — vector store / similarity search
- **transformers** — response generation (FLAN-T5)
- **Gradio (optional)** — simple UI chatbot

## Text Chunking Strategy

- **Chunk size:** 1 row = 1 chunk (plus Make-wise summary chunks)
- **Chunk overlap:** 0
- **Reason:** Each row is an independent record; overlap is unnecessary. Summary chunks help answer questions like "highest average price" or "loan pattern".

## Embedding Model

- **Model:** sentence-transformers/all-MiniLM-L6-v2
- **Reason:** Fast, lightweight, and works well for semantic retrieval on mixed text (categories + numeric values).

## Vector Database

- **Vector store:** FAISS (IndexFlatIP)
- **Similarity:** Cosine similarity (L2-normalized embeddings)

## How to Run (Google Colab)

- Open the notebook in Google Colab
- Run the install cell (pins transformers to v4.x using **transformers<5**)
- Upload the CSV dataset when prompted

- Run cells in order: Load → Chunk → Embed → Index → Retrieve → Generate
- Run the test queries cell and record outputs
- (Optional) Run the Gradio UI cell

## Minimum Test Queries (Example)

The notebook includes at least 3 test queries such as:

- Which car make has the highest average price?
- What is the loan pattern for SUV buyers?
- Give a typical customer profile for Sedan buyers.

Outputs are printed along with retrieved source metadata and similarity scores.

## Bonus (Optional): Gradio UI

A simple Gradio interface is included to ask questions and view the final answer plus retrieved sources.

## Future Improvements

- Better chunking for tabular data (group by Make/Profession/salary bands)
- Reranking (cross-encoder) for more accurate top-k retrieval
- Hybrid search (BM25 + embeddings)
- Metadata filtering (e.g., filter Make=SUV before retrieval)
- More advanced UI (Streamlit dashboards + charts)

## Summary

This assignment implements a working RAG system on a CSV dataset. It uses MiniLM embeddings and FAISS retrieval to fetch relevant records and FLAN-T5 to generate answers grounded in retrieved context. The approach improves factual accuracy and demonstrates an end-to-end RAG workflow suitable for data-driven Q&A systems.