Assignment-1

for

AGENTIC AI LAB

CSCR3215

SEMESTER: VIth

Section: CS-F

Group: G2

SUBMITTED BY:

Priyanshu Kumar Singh(2023357193)

Harshit Kumar singh(2023491064)

Table of Contents

1. Problem Statement (Clear Problem Definition)

The goal of this assignment is to build a Retrieval-Augmented Generation (RAG) system that answers user questions using a CSV dataset containing customer demographics, income/loan information, and car purchase details (Make and Price).

RAG is used to reduce hallucination by retrieving relevant records from the dataset first, and then generating the final answer using only the retrieved context.

Challenges addressed:

- Tabular (CSV) records are not naturally searchable using plain keyword queries.
- User questions can be analytical (e.g., highest average price) and require aggregation.
- LLMs may hallucinate if answers are not grounded in evidence.

2. Dataset / Knowledge Source

Type of data: CSV (structured tabular data).

Data source: Provided/public dataset uploaded in Google Colab.

Dataset size: 1581 rows × 14 columns.

Key columns: Age, Gender, Profession, Education, Marital_status, No_of_Dependents, Personal_loan, House_loan, Salary, Partner_salary, Total_salary, Make, Price.

Quick dataset profile (from this dataset):

- Makes: Sedan (702), Hatchback (582), SUV (295).
- Overall loan rates: Personal_loan Yes ≈ 50.09%, House_loan Yes ≈ 33.33%.

3. RAG Architecture

Block diagram of complete RAG pipeline (logical flow):

```
CSV → Pandas Loader → Row-to-Text Conversion → Chunking
(Rows + Summaries)
   → Embedding (MiniLM) → FAISS Index
User Query → Query Embedding → FAISS Similarity Search →
Top-k Retrieved Chunks
   → Prompt Builder → FLAN-T5 → Final Answer + Sources
```

Stages:

- Stage 1: Data ingestion (load CSV and clean missing values).
- Stage 2: Text chunking (row chunks + Make-wise summary chunks).
- Stage 3: Embedding generation using all-MiniLM-L6-v2.
- Stage 4: Vector storage and search using FAISS.
- Stage 5: Retrieval of top-k relevant chunks for a query.
- Stage 6: Answer generation using FLAN-T5 with retrieved context only.

4. Text Chunking Strategy

Parameter	Value
Chunk size	1 CSV row = 1 chunk (+ summary chunks per Make)
Chunk overlap	0
Reason	Rows are independent records; summary chunks support analytical queries (avg price, loan %).

5. Embedding Details

Parameter	Value
Embedding model	sentence-transformers/all-MiniLM-L6-v2
Embedding dimension	384
Reason	Fast on CPU, good semantic similarity, widely used for RAG, open-source.

6. Vector Database

Parameter	Value
Vector store	FAISS (IndexFlatIP)
Similarity	Cosine similarity (L2-normalized vectors)
Retrieval	Top-k similarity search (k=5)

7. Implementation Overview

The project is implemented in a Google Colab notebook with the following steps:

1. Install libraries (pin transformers<5 for pipeline compatibility).
2. Upload CSV and load with pandas.
3. Data cleaning (handle missing values, numeric conversion).
4. Row-to-text conversion and summary chunk creation.
5. Embedding generation using MiniLM.
6. Build FAISS index and implement retrieval.
7. Generate final answers using FLAN-T5 with retrieved context only.
8. Run test queries; optionally launch Gradio UI.

8. Test Queries & Outputs (Minimum 3)

Query 1: Which car make has the highest average price?

Output: SUV has the highest average price (avg_price ≈ 57,681.55).

Query 2: What is the loan pattern for SUV buyers?

Output: Personal loan Yes ≈ 38.98%, House loan Yes ≈ 7.46%, Median SUV price ≈ 57,000 (count=295).

Query 3: Give a typical customer profile for Sedan buyers.

Output: Profession=Salaried, Education=Post Graduate, Marital_status=Married. Avg age ≈ 31.92, Avg total salary ≈ 79,584.76, Personal loan Yes ≈ 54.84%.

Note: The notebook prints retrieved sources (chunk metadata + similarity scores) for transparency.

9. Bonus (Optional): Gradio UI

A simple Gradio interface is included for interactive Q&A over the CSV dataset.

- Textbox input for user questions.
- Answer generated using retrieved evidence only.
- Shows retrieved source metadata and similarity scores.
- Can be launched with a shareable link in Colab.

10. Future Improvements

- Better chunking: group by Make/Profession/Salary bands and build hierarchical summaries.
- Reranking / hybrid search: combine BM25 + dense retrieval; apply a cross-encoder reranker.
- Metadata filtering: filter by Make, Gender, Profession, or salary range before retrieval.
- UI integration: Streamlit dashboard with filters and charts.

- Evaluation: create a small ground-truth QA set and measure precision@k / recall@k.

11. Tools & Libraries Used

Tool / Library	Purpose
Google Colab	Execution environment
Python	Programming language
pandas / numpy	CSV loading, cleaning, statistics
sentence-transformers	Embedding generation (MiniLM)
FAISS (faiss-cpu)	Vector similarity search
transformers	LLM generation (FLAN-T5)
Gradio (optional)	UI for interactive chat

12. Instructions to Run the Notebook

1. Open the notebook in Google Colab.
2. Run the install cell (uses transformers<5).
3. Upload the CSV dataset when prompted.
4. Run cells in order to build embeddings, FAISS index, retrieval, and generation.
5. Run the test query cell and record outputs for submission.

Summary

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