**Python and EDA code:**

import os

import json

import fitz # PyMuPDF

import nltk

from nltk.tokenize import word\_tokenize

from nltk.corpus import stopwords

import string

import spacy

import numpy as np

import faiss

import google.generativeai as genai

# ------------------- 1️⃣ Folder Paths -------------------

PDF\_FOLDER\_PATH = r"E:\AI-EDUCATION\class10sciencebook"

TEXT\_OUTPUT\_FILE = r"E:\AI-EDUCATION\cleaned\_science\_data.json"

PROCESSED\_TEXT\_FILE = r"E:\AI-EDUCATION\processed\_science\_data.json"

EMBEDDINGS\_FILE = r"E:\AI-EDUCATION\spaCy\_embeddings.json"

IMAGE\_OUTPUT\_FOLDER = r"E:\AI-EDUCATION\extracted\_images"

VECTOR\_DB\_PATH = r"E:\AI-EDUCATION\vector\_database.index"

# ------------------- 2️⃣ Extract Text from PDFs -------------------

def extract\_text\_from\_pdf(pdf\_path):

doc = fitz.open(pdf\_path)

return "\n".join(page.get\_text("text") for page in doc)

data = {}

for file in os.listdir(PDF\_FOLDER\_PATH):

if file.endswith(".pdf"):

file\_path = os.path.join(PDF\_FOLDER\_PATH, file)

chapter\_name = file.replace(".pdf", "")

data[chapter\_name] = extract\_text\_from\_pdf(file\_path)

with open(TEXT\_OUTPUT\_FILE, "w", encoding="utf-8") as f:

json.dump(data, f, indent=4, ensure\_ascii=False)

print("✅ Text extraction completed & saved as JSON!")

# ------------------- 3️⃣ Text Preprocessing -------------------

nltk.download("punkt")

nltk.download("stopwords")

def clean\_text(text):

text = text.lower()

text = ''.join(char for char in text if char not in string.punctuation)

words = word\_tokenize(text)

words = [word for word in words if word not in stopwords.words('english')]

return " ".join(words)

with open(TEXT\_OUTPUT\_FILE, "r", encoding="utf-8") as f:

data = json.load(f)

for chapter in data:

data[chapter] = clean\_text(data[chapter])

with open(PROCESSED\_TEXT\_FILE, "w", encoding="utf-8") as f:

json.dump(data, f, indent=4, ensure\_ascii=False)

print("✅ Text preprocessing completed & saved as JSON!")

# ------------------- 4️⃣ Convert to Embeddings -------------------

nlp = spacy.load("en\_core\_web\_lg")

def get\_embedding(text):

return nlp(text).vector

with open(PROCESSED\_TEXT\_FILE, "r", encoding="utf-8") as f:

data = json.load(f)

embeddings = {chapter: get\_embedding(data[chapter]).tolist() for chapter in data}

with open(EMBEDDINGS\_FILE, "w", encoding="utf-8") as f:

json.dump(embeddings, f, indent=4)

print("✅ Text converted to embeddings & saved as JSON using spaCy!")

# ------------------- 5️⃣ Extract Images -------------------

if not os.path.exists(IMAGE\_OUTPUT\_FOLDER):

os.makedirs(IMAGE\_OUTPUT\_FOLDER)

image\_count = 0

for filename in os.listdir(PDF\_FOLDER\_PATH):

if filename.endswith(".pdf"):

doc = fitz.open(os.path.join(PDF\_FOLDER\_PATH, filename))

for page\_num, page in enumerate(doc):

for img\_index, img in enumerate(page.get\_images(full=True)):

xref = img[0]

base\_image = doc.extract\_image(xref)

image\_bytes = base\_image["image"]

img\_filename = f"{filename}\_page{page\_num+1}\_img{img\_index+1}.png"

with open(os.path.join(IMAGE\_OUTPUT\_FOLDER, img\_filename), "wb") as img\_file:

img\_file.write(image\_bytes)

image\_count += 1

print(f"✅ Extracted {image\_count} images from all chapters.")

# ------------------- 6️⃣ Store in FAISS -------------------

with open(EMBEDDINGS\_FILE, "r") as f:

embeddings = json.load(f)

embedding\_matrix = np.array(list(embeddings.values()), dtype='float32')

dimension = embedding\_matrix.shape[1]

index = faiss.IndexFlatL2(dimension)

index.add(embedding\_matrix)

faiss.write\_index(index, VECTOR\_DB\_PATH)

print("✅ Embeddings stored in FAISS vector database!")

# ------------------- 7️⃣ Gemini-based QA System -------------------

genai.configure(api\_key="My API KEY ")

model = genai.GenerativeModel('models/gemini-1.5-flash')

with open(PROCESSED\_TEXT\_FILE, "r", encoding="utf-8") as f:

processed\_text = json.load(f)

chapter\_names = list(embeddings.keys())

index = faiss.read\_index(VECTOR\_DB\_PATH)

def search\_faiss(query, top\_k=3):

query\_vector = np.array([get\_embedding(query)], dtype='float32')

distances, indices = index.search(query\_vector, top\_k)

return [chapter\_names[i] for i in indices[0]]

def query\_gemini\_with\_context(query, top\_chapters):

context = "\n\n".join([f"{ch}:\n{processed\_text[ch]}" for ch in top\_chapters])

prompt = f"""You are a helpful AI tutor for Class 10 Science students.

Use the following NCERT content to answer the question accurately and simply.

Context:

{context}

Question: {query}

Answer in simple terms as per NCERT syllabus:"""

response = model.generate\_content(prompt)

return response.text

# ------------------- 8️⃣ Run the Chatbot -------------------

if \_\_name\_\_ == "\_\_main\_\_":

question = input("❓ Ask your NCERT Science question: ")

top\_chaps = search\_faiss(question)

answer = query\_gemini\_with\_context(question, top\_chaps)

print("\n💡 Answer:\n", answer)