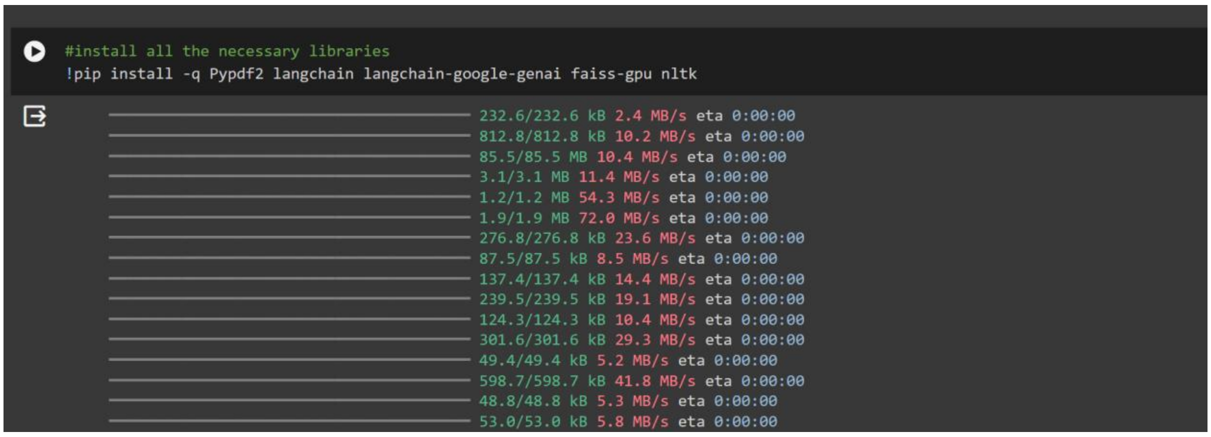
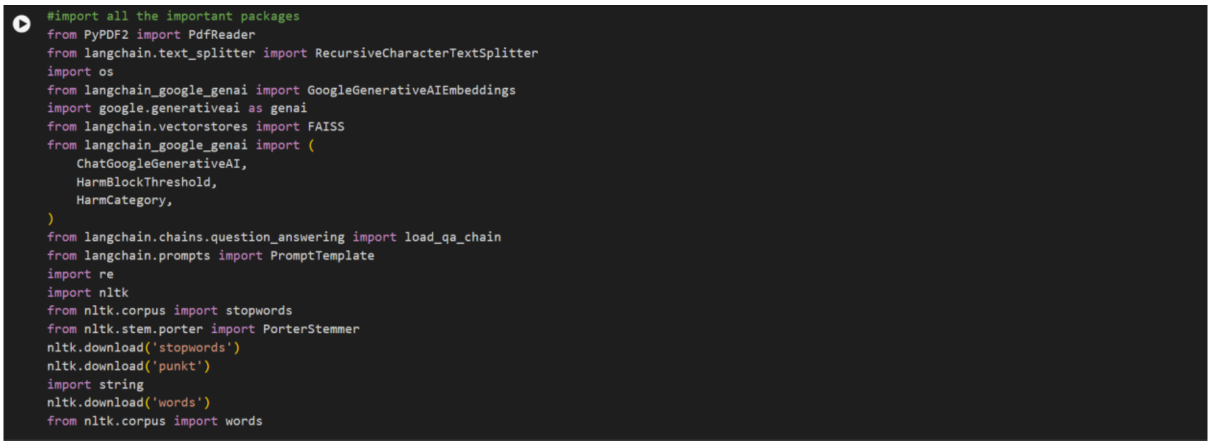
Detailed Walkthrough

Step 1) Download all the necessary libraries.



```
#install all the necessary libraries
!pip install -q Pypdf2 langchain langchain-google-genai faiss-gpu nltk

     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 232.6/232.6 kB 2.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 812.8/812.8 kB 10.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.5/85.5 MB 10.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.1/3.1 MB 11.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 54.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.9/1.9 MB 72.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 276.8/276.8 kB 23.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 87.5/87.5 kB 8.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 137.4/137.4 kB 14.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 239.5/239.5 kB 19.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 124.3/124.3 kB 10.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 301.6/301.6 kB 29.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.4/49.4 kB 5.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 598.7/598.7 kB 41.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 48.8/48.8 kB 5.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 53.0/53.0 kB 5.8 MB/s eta 0:00:00
```

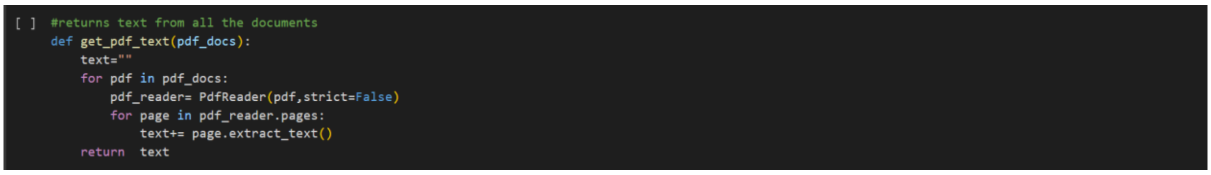
Step 2) Import all the necessary packages.



```python
#import all the important packages
from PyPDF2 import PdfReader
from langchain.text_splitter import RecursiveCharacterTextSplitter
import os
from langchain_google_genai import GoogleGenerativeAIEmbeddings
import google.generativeai as genai
from langchain.vectorstores import FAISS
from langchain_google_genai import (
    ChatGoogleGenerativeAI,
    HarmBlockThreshold,
    HarmCategory,
)
from langchain.chains.question_answering import load_qa_chain
from langchain.prompts import PromptTemplate
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
nltk.download('stopwords')
nltk.download('punkt')
import string
nltk.download('words')
from nltk.corpus import words
```

I have used 'gemini-pro' model using Langchain and FAISS as my vector database to store all the data.

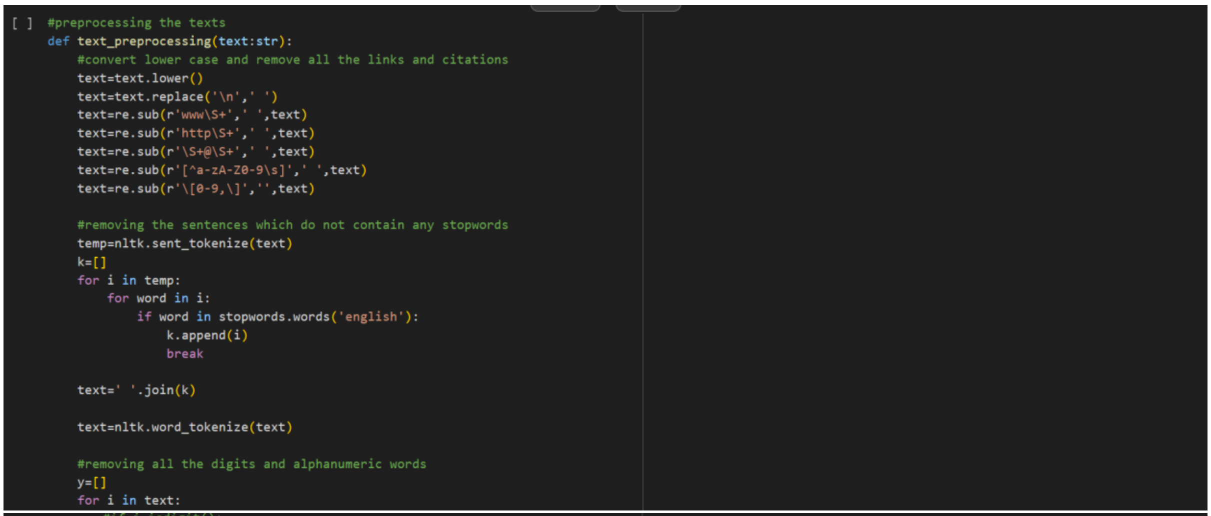
Step 3) Load all text from multiple pdfs provided.



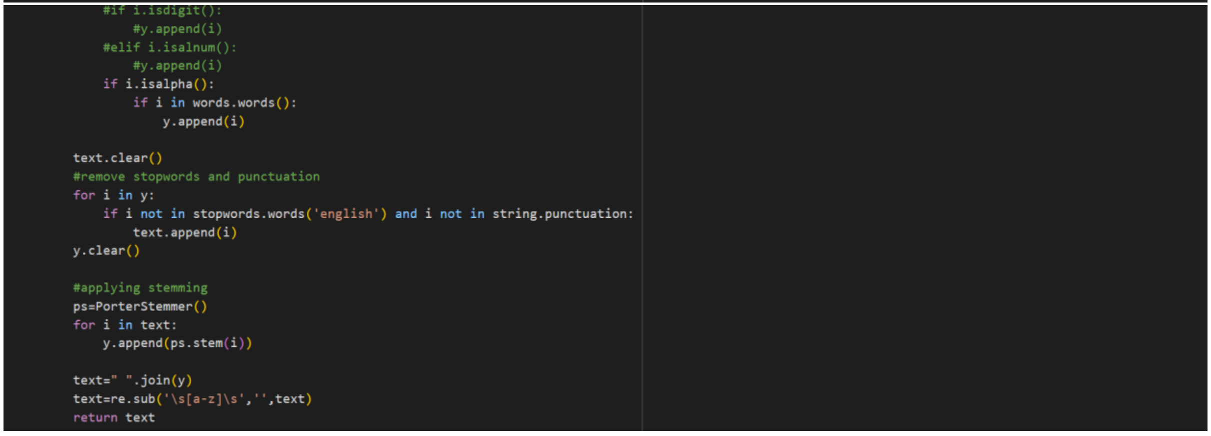
```python
[ ] #returns text from all the documents
    def get_pdf_text(pdf_docs):
        text=""
        for pdf in pdf_docs:
            pdf_reader= PdfReader(pdf,strict=False)
            for page in pdf_reader.pages:
                text+= page.extract_text()
        return  text
```

I have used PyPDF2 to extract text page by page from all the pdfs provided and returned them combined.

Step 4) Text Preprocessing



```
[ ] #preprocessing the texts
    def text_preprocessing(text:str):
        #convert lower case and remove all the links and citations
        text=text.lower()
        text=text.replace('\n',' ')
        text=re.sub(r'www\S+',' ',text)
        text=re.sub(r'http\S+',' ',text)
        text=re.sub(r'\S+@\S+',' ',text)
        text=re.sub(r'[^a-zA-Z0-9\s]',' ',text)
        text=re.sub(r'\[0-9,\]','',text)

        #removing the sentences which do not contain any stopwords
        temp=nltk.sent_tokenize(text)
        k=[]
        for i in temp:
            for word in i:
                if word in stopwords.words('english'):
                    k.append(i)
                    break

        text=' '.join(k)

        text=nltk.word_tokenize(text)

        #removing all the digits and alphanumeric words
        y=[]
        for i in text:
```



```
            #if i.isdigit():
                #y.append(i)
            #elif i.isalnum():
                #y.append(i)
            if i.isalpha():
                if i in words.words():
                    y.append(i)

        text.clear()
        #remove stopwords and punctuation
        for i in y:
            if i not in stopwords.words('english') and i not in string.punctuation:
                text.append(i)
        y.clear()

        #applying stemming
        ps=PorterStemmer()
        for i in text:
            y.append(ps.stem(i))

        text=" ".join(y)
        text=re.sub('\s[a-z]\s','',text)
        return text
```

Initially through re library I have removed all the links, references and citations.

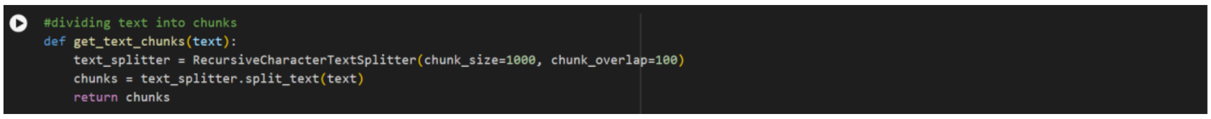
Then, I have removed all the sentences without any stopwords using nltk library as they are most likely from some table or image data (An English sentence always has atleast one stopword).

Then I have checked for each word if it belongs to the dictionary or not and have eliminated if not.

After that I have removed all the stopwords and punctuations.

Then, I have applied stemming using PorterStemmer from nltk.

Step 5) Dividing into chunks

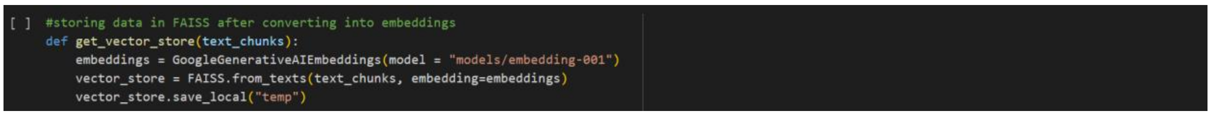


```
    #dividing text into chunks
    def get_text_chunks(text):
        text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=100)
        chunks = text_splitter.split_text(text)
        return chunks
```

Dividing all that huge data into small chunks so that we can store in the form of vectors and our model can process.

We have used RecursiveCharacterTextSplitter from Langchain for this.

Step 6) Storing Into Database

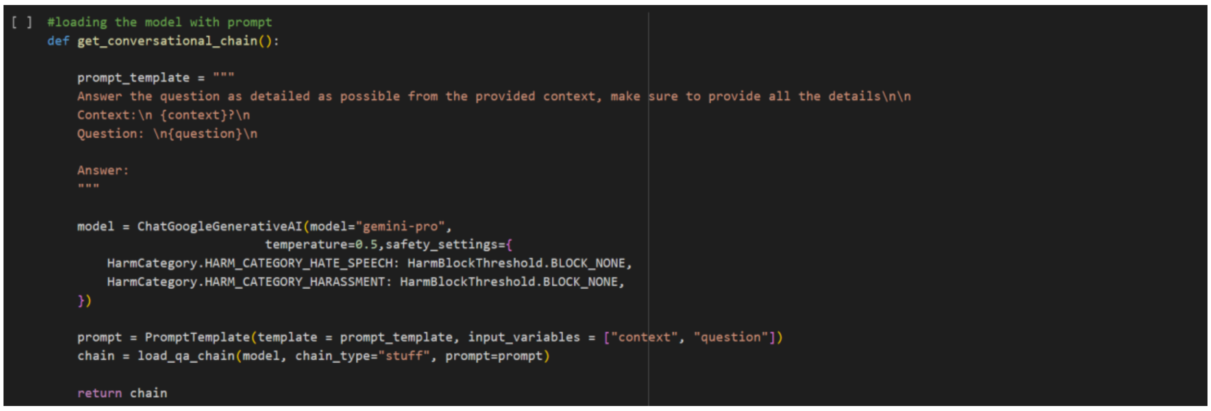


```
[ ] #storing data in FAISS after converting into embeddings
    def get_vector_store(text_chunks):
        embeddings = GoogleGenerativeAIEmbeddings(model = "models/embedding-001")
        vector_store = FAISS.from_texts(text_chunks, embedding=embeddings)
        vector_store.save_local("temp")
```

We convert our chunks of data into embeddings using GoogleAIEmbeddings from Langchain.

Next, we store these embeddings into FAISS (Facebook AI Similarity Search) vector database which provides very fast similarity search and data retrieval properties.

STEP 7) Creating a prompt and passing into the model.

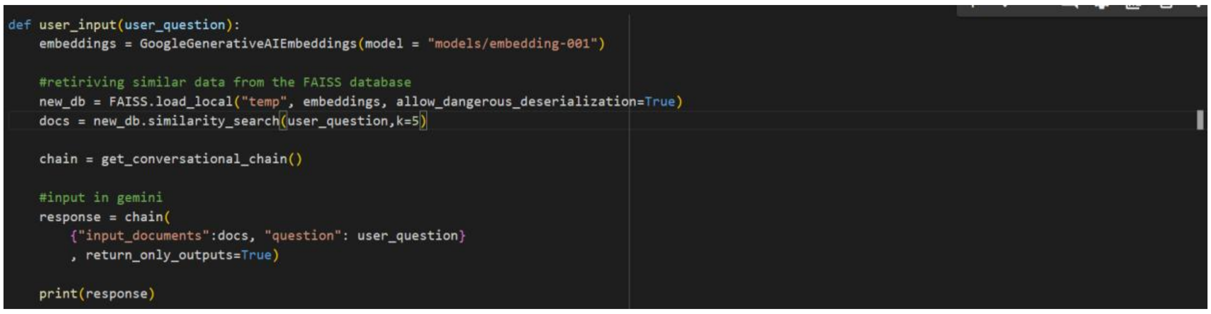


```
[ ] #loading the model with prompt
    def get_conversational_chain():

        prompt_template = """
        Answer the question as detailed as possible from the provided context, make sure to provide all the details\n\n
        Context:\n {context}?\n
        Question: \n{question}\n

        Answer:
        """

        model = ChatGoogleGenerativeAI(model="gemini-pro",
                                 temperature=0.5,safety_settings={
            HarmCategory.HARM_CATEGORY_HATE_SPEECH: HarmBlockThreshold.BLOCK_NONE,
            HarmCategory.HARM_CATEGORY_HARASSMENT: HarmBlockThreshold.BLOCK_NONE,
        })

        prompt = PromptTemplate(template = prompt_template, input_variables = ["context", "question"])
        chain = load_qa_chain(model, chain_type="stuff", prompt=prompt)

        return chain
```

Created a prompt using Langchian’s PromptTemplate so that our model returns output from the retrieved information from our vector database (FAISS).

Then, load the model ‘gemini-pro’ in this case and feed the context from data, user question and template into the qa-chain from Langchain.

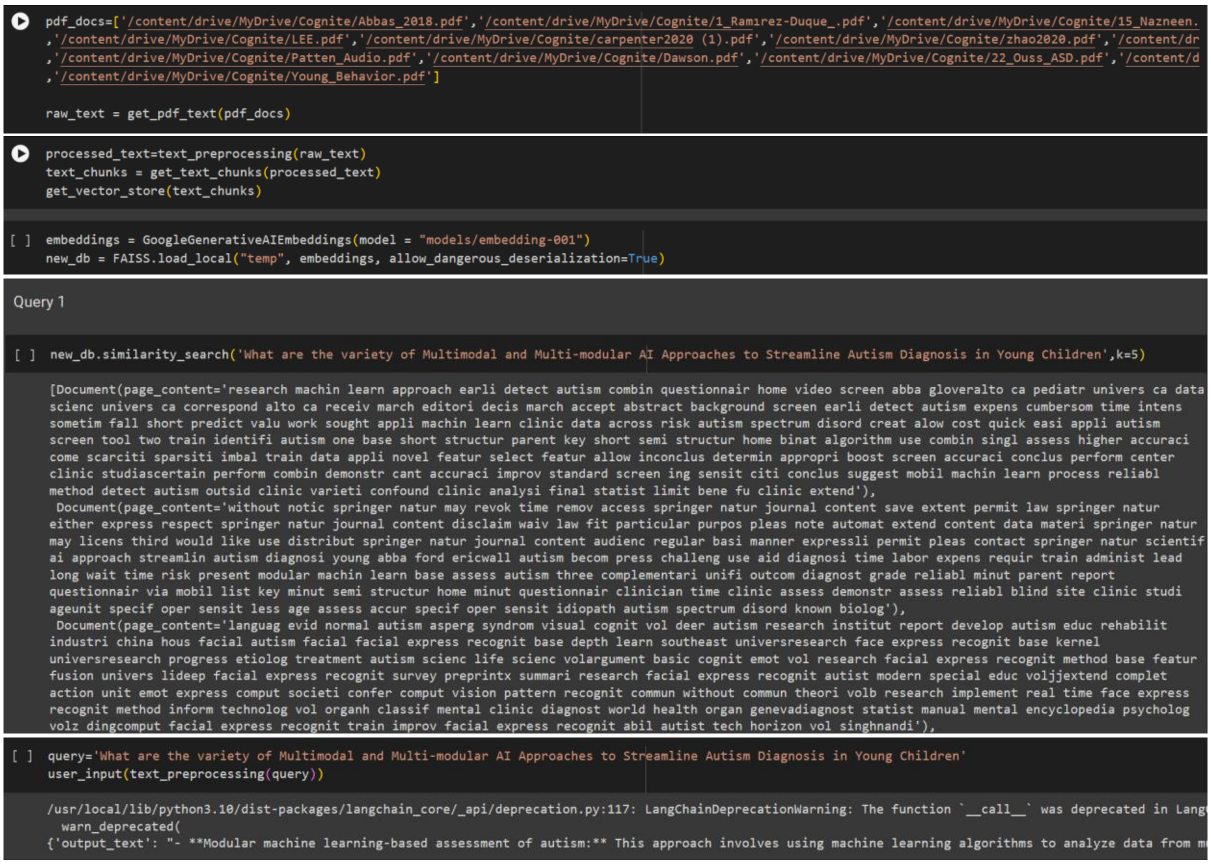
Step 8) Get response from the model.



```
def user_input(user_question):
    embeddings = GoogleGenerativeAIEmbeddings(model = "models/embedding-001")

    #retiriving similar data from the FAISS database
    new_db = FAISS.load_local("temp", embeddings, allow_dangerous_deserialization=True)
    docs = new_db.similarity_search(user_question,k=5)

    chain = get_conversational_chain()

    #input in gemini
    response = chain(
        {"input_documents":docs, "question": user_question}
        , return_only_outputs=True)

    print(response)
```

Perform the similarity search and load the 5 most similar chunks from FAISS and pass into the chain to get the output.

Step 9) Perform all these steps together.



```
pdf_docs=['/content/drive/MyDrive/Cognite/Abbas_2018.pdf','/content/drive/MyDrive/Cognite/1_Ramırez-Duque_.pdf','/content/drive/MyDrive/Cognite/15_Nazneen.
,'/content/drive/MyDrive/Cognite/LEE.pdf','/content/drive/MyDrive/Cognite/carpenter2020 (1).pdf','/content/drive/MyDrive/Cognite/zhao2020.pdf','/content/dr
,'/content/drive/MyDrive/Cognite/Patten_Audio.pdf','/content/drive/MyDrive/Cognite/Dawson.pdf','/content/drive/MyDrive/Cognite/22_Ouss_ASD.pdf','/content/d
,'/content/drive/MyDrive/Cognite/Young_Behavior.pdf']

raw_text = get_pdf_text(pdf_docs)
```

```
processed_text=text_preprocessing(raw_text)
text_chunks = get_text_chunks(processed_text)
get_vector_store(text_chunks)
```

```
[ ] embeddings = GoogleGenerativeAIEmbeddings(model = "models/embedding-001")
    new_db = FAISS.load_local("temp", embeddings, allow_dangerous_deserialization=True)
```

Query 1

```
[ ] new_db.similarity_search('What are the variety of Multimodal and Multi-modular AI Approaches to Streamline Autism Diagnosis in Young Children',k=5)

    [Document(page_content='research machin learn approach earli detect autism combin questionnair home video screen abba gloveralto ca pediatr univers ca data
    scienc univers ca correspond alto ca receiv march editori decis march accept abstract background screen earli detect autism expens cumbersom time intens
    sometim fall short predict valu work sought appli machin learn clinic data across risk autism spectrum disord creat alow cost quick easi appli autism
    screen tool two train identifi autism one base short structur parent key short semi structur home binat algorithm use combin singl assess higher accuraci
    come scarciti sparsiti imbal train data appli novel featur select featur allow inconclus determin appropri boost screen accuraci conclus perform center
    clinic studiascertain perform combin demonstr cant accuraci improv standard screen ing sensit citi conclus suggest mobil machin learn process reliabl
    method detect autism outsid clinic varieti confound clinic analysi final statist limit bene fu clinic extend'),
     Document(page_content='without notic springer natur may revok time remov access springer natur journal content save extent permit law springer natur
    either express respect springer natur journal content disclaim waiv law fit particular purpos pleas note automat extend content data materi springer natur
    may licens third would like use distribut springer natur journal content audienc regular basi manner expressli permit pleas contact springer natur scientif
    ai approach streamlin autism diagnosi young abba ford ericwall autism becom press challeng use aid diagnosi time labor expens requir train administ lead
    long wait time risk present modular machin learn base assess autism three complementari unifi outcom diagnost grade reliabl minut parent report
    questionnair via mobil list key minut semi structur home minut questionnair clinician time clinic assess demonstr assess reliabl blind site clinic studi
    ageunit specif oper sensit less age assess accur specif oper sensit idiopath autism spectrum disord known biolog'),
     Document(page_content='languag evid normal autism asperg syndrom visual cognit vol deer autism research institut report develop autism educ rehabilit
    industri china hous facial autism facial facial express recognit base depth learn southeast universresearch face express recognit base kernel
    universresearch progress etiolog treatment autism scienc life scienc volargument basic cognit emot vol research facial express recognit method base featur
    fusion univers lideep facial express recognit survey preprintx summari research facial express recognit autist modern special educ voljjextend complet
    action unit emot express comput societi confer comput vision pattern recognit commun without commun theori volb research implement real time face express
    recognit method inform technolog vol organh classif mental clinic diagnost world health organ genevadiagnost statist manual mental encyclopedia psycholog
    volz dingcomput facial express recognit train improv facial express recognit abil autist tech horizon vol singhnandi'),
```

```
[ ] query='What are the variety of Multimodal and Multi-modular AI Approaches to Streamline Autism Diagnosis in Young Children'
    user_input(text_preprocessing(query))

    /usr/local/lib/python3.10/dist-packages/langchain_core/_api/deprecation.py:117: LangChainDeprecationWarning: The function `__call__` was deprecated in Lang
      warn_deprecated(
    {'output_text': "- **Modular machine learning-based assessment of autism:** This approach involves using machine learning algorithms to analyze data from m
```

Do the same for all the queries.

Note: Output for all the queries is pasted in another pdf.

Code for this is attached to as a google collab link.