```
import sqlite3
import pandas as pd
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import sqlite3
db_path = '/content/drive/MyDrive/eng_subtitles_database.db'
conn = sqlite3.connect(db_path)
```

```
df = pd.read_sql_query("""SELECT * FROM zipfiles""", conn)
```

```
import re

def clean_content(content):
    try:
        with io.BytesIO(content) as f:
            with zipfile.ZipFile(f, 'r') as zip_file:
                subtitle_content = zip_file.read(zip_file.namelist()[0])
                decoded = subtitle_content.decode('utf-8', errors='ignore')

                # Remove timestamps
                cleaned = re.sub(r'\d{1,2}:\d{2}:\d{2},\d{3} --> \d{1,2}:\d{2}:\d{2},\d{3}', '', decoded)

                # Remove line numbers
                cleaned = re.sub(r'^\s*\d+\s*\r?\n', '', cleaned, flags=re.MULTILINE)

                # Remove extra newlines and whitespace
                cleaned = re.sub(r'\n\s*\n', '\n', cleaned)
                cleaned = re.sub(r'^\s+', '', cleaned, flags=re.MULTILINE)

                return cleaned.strip()
    except (zipfile.BadZipFile, UnicodeDecodeError) as e:
        print(f"Error processing content: {e}")
        return None
```

```
import zipfile
import io
```

```
# Subsetting the DataFrame to 10%
subset_df = df.head(int(len(df) * 0.10)).copy()

# Applying cleaning and filter to the subset
subset_df.loc[:, 'clean_content'] = subset_df['content'].apply(clean_content)
```

```
print(subset_df[['content', 'clean_content']].head(2))
```

```
                                             content  \
0  b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x1c\xa9\x...
1  b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x17\xb9\x...

                                       clean_content
0  Watch any video online with Open-SUBTITLES\r\n...
1  Ah! There's Princess\r\nDawn and Terry with th...
```

```
!pip install -q langchain chromadb sentence-transformers google-generativeai==0.8.4 google-ai-generativelanguage==0.6.15
```

Show hidden output

```
import sqlite3
import zlib
import re
from langchain.text_splitter import RecursiveCharacterTextSplitter
```

```
!pip install langchain_community
```

Show hidden output

```
import langchain_community
```

```
from langchain.vectorstores import Chroma
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.chains import RetrievalQA
from langchain.prompts import ChatPromptTemplate
```

```
!pip install google-generativeai
```

Show hidden output

```
!pip install langchain-google-genai
```

Show hidden output

```
from langchain_google_genai import ChatGoogleGenerativeAI
```

```
def chunk_text(texts):
    splitter = RecursiveCharacterTextSplitter(
        chunk_size=500,
        chunk_overlap=50,
        separators=["\n\n", "\n", ". "]
    )
    return splitter.create_documents(texts)
```

```
all_chunks = chunk_text(subset_df['clean_content'].tolist())
```

```
!pip install -U langchain_huggingface
```

Show hidden output

```
from langchain_huggingface import HuggingFaceEmbeddings
from langchain.vectorstores import Chroma
from tqdm import tqdm
from IPython import get_ipython
```

```
from google.colab import userdata
print(userdata.get('HF_TOKEN'))
```

Show hidden output

```
# STEP 1: Install required packages
!pip install -qU langchain sentence-transformers chromadb tqdm
```

Show hidden output

```
embeddings_10_percent = HuggingFaceEmbeddings(
    model_name="sentence-transformers/all-MiniLM-L6-v2",
    model_kwargs={'token': userdata.get('HF_TOKEN')}
)

vector_db_10_percent = Chroma(
    persist_directory="./chroma_db_10_percent_local",
    embedding_function=embeddings_10_percent,
    collection_name="subtitle_embeddings_10_percent"
)

batch_size = 512
for i in tqdm(range(0, len(all_chunks), batch_size), desc="Embedding Progress (10% Data)"):
    batch = all_chunks[i:i + batch_size]
    vector_db_10_percent.add_documents(batch)

# Verifying the number of embedded documents
print(f"Total embedded documents (10% data): {vector_db_10_percent._collection.count()}")
```

```
Embedding Progress (10% Data): 100%|██████████| 1129/1129 [47:05<00:00,  2.50s/it]Total embedded documents (10% data): 577862
```

```
# Persist before moving
vector_db_10_percent.persist()

# Save to Google Drive
!mv ./chroma_db_10_percent_local /content/drive/MyDrive/chroma_db_10_percent_drive
print("Embeddings (10%) saved to Google Drive!")
```

```
<ipython-input-66-17095dfc82d5>:2: LangChainDeprecationWarning: Since Chroma 0.4.x the manual persistence method is no longer suppor
  vector_db_10_percent.persist()
Embeddings (10%) saved to Google Drive!
```

```
# Trying to retrieve the embeddings from drive for easy use from next time  onwards
embeddings_10_percent = HuggingFaceEmbeddings(
    model_name="sentence-transformers/all-MiniLM-L6-v2",
    model_kwargs={'token': userdata.get('HF_TOKEN')}
)
```

Show hidden output

```
vector_db_10_percent = Chroma(
       persist_directory="/content/drive/MyDrive/chroma_db_10_percent_drive",
       embedding_function=embeddings_10_percent,
       collection_name="subtitle_embeddings_10_percent"
   )
```

```
<ipython-input-8-fbaef9e676ac>:1: LangChainDeprecationWarning: The class `Chroma` was deprecated in LangChain 0.2.9 and will be remo
  vector_db_10_percent = Chroma(
```

```
print(f"Total embedded documents (10% data): {vector_db_10_percent._collection.count()}")
```

```
Total embedded documents (10% data): 577862
```

Succesfully I retrieved my embeddings from drive without running the full code again

```
# Set up retrieval component
from langchain.chains import RetrievalQA
from langchain.prompts import ChatPromptTemplate

# Create a retriever from the reloaded vector database
retriever = vector_db_10_percent.as_retriever(
    search_type="similarity",
    search_kwargs={
        "k": 5
    }
)
```

```
from langchain_google_genai import ChatGoogleGenerativeAI
llm = ChatGoogleGenerativeAI(
    model="gemini-1.5-pro-002",
    temperature=0.3,
    google_api_key= "Use your API_KEY from Secrets"
)
```

```
prompt_template = """Use the following movie subtitle excerpts to answer the question.
Focus on dialogues and emotional context. If unsure, state "Insufficient context".

Context:
{context}
Question: {question}
Answer:"""

rag_prompt = ChatPromptTemplate.from_template(prompt_template)
```

```
rag_chain = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type="stuff",
    retriever=retriever,
    chain_type_kwargs={"prompt": rag_prompt},
    return_source_documents=True
)
```

```
def ask_question(question):
    result = rag_chain.invoke({"query": question})

    print(f"\nQuestion: {question}")

    if result['source_documents']:
        print(f"\nAnswer: {result['result']}")
        print("\nRelevant Contexts:")
        for doc in result['source_documents']:
            print(f"- {doc.page_content[:150]}...")
            if doc.metadata:
                print(f"  Metadata: {doc.metadata}")  # Print metadata if available
    else:
        print("\nNo relevant contexts found. Try rephrasing your question or expanding the dataset.")
```

```
ask_question("What is love?")
```

```
Question: What is love?

Answer: The excerpts offer multiple perspectives on love:

* **Philosophical/Religious:** Love is described as a seeking for a way of life not meant to be alone, a resonance of all things spi

* **Personal/Experiential:**  Another voice admits to not knowing what love truly is, but acknowledges its surprising nature, its co

* **Poetic/Spiritual (from the third excerpt):**  This perspective describes love as a saga witnessed by earth and sky, a force that


The excerpts don't offer a single definition but explore love's multifaceted nature through different lenses.

Relevant Contexts:
- "Love is a seeking
for a way of life,
"the way
that cannot be followed alone,
"the resonance of all spiritual
and physical things.
"Friendship i...
- gentle, pure-hearted, persevering,
merciful... and so on.
If love means all this,
then it must be
the name for all virtues put together.
Love is ...
- "things that relate
to those who are loved
"and those who are real friends.
"For the first time,
I know what love is,
"what friends are
and what...
- But the truth is
I really don't know
anything about love.
I do know that it surprises us.
That it arrives
when we least expect it.
Sometimes it'...
- are moving ahead.
The earth and the sky
listens their saga of love.
They are leaving this
worldly attachments..
..with hopes in their heart
and ...
```

So far, the application has successfully processed user queries in text format, providing accurate and relevant outputs based on the input received. As part of the ongoing improvements, I am excited to announce the integration of audio input functionality. This enhancement is implemented in the provided Streamlit code file, and it is currently operational.

```
pip install streamlit
```

Show hidden output

```
!pip install streamlit openai-whisper
```

Show hidden output

```
!pip install transformers
```

Show hidden output

```
pip install streamlit transformers torch
```

Show hidden output

```
pip install streamlit-audiorec
```

Show hidden output

```
from st_audiorec import st_audiorec
```

```
!pip install streamlit-mic-recorder
```

Show hidden output

```
import streamlit as st
from transformers import pipeline
```

```
!pip install torch
```

Show hidden output

```
%%writefile streamlit_app2.py
```

```
Writing streamlit_app2.py
```

```
! wget -q -O - ipv4.icanhazip.com
```

```
35.194.170.199
```

```
! streamlit run streamlit_app2.py & npx localtunnel --port 8501
```

```

Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.


  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://172.28.0.12:8501
  External URL: http://35.194.170.199:8501

⠙⠹⠸⠼⠴Need to install the following packages:
localtunnel@2.0.2
Ok to proceed? (y) y

⠙⠹⠸⠼⠴⠦⠧⠇⠏⠋⠙⠹⠸⠼⠴⠦⠧⠇⠏⠋⠙⠹your url is: https://dark-zebras-dig.loca.lt
2025-03-22 07:12:02.760564: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to register cuFFT factory: Atte
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1742627522.824832    5772 cuda_dnn.cc:8310] Unable to register cuDNN factory: Attempting to register factory for plug
E0000 00:00:1742627522.844941    5772 cuda_blas.cc:1418] Unable to register cuBLAS factory: Attempting to register factory for pl
/content/streamlit_app2.py:5: LangChainDeprecationWarning: Importing HuggingFaceEmbeddings from langchain.embeddings is deprecate

>> from langchain.embeddings import HuggingFaceEmbeddings

with new imports of:

>> from langchain_community.embeddings import HuggingFaceEmbeddings
You can use the langchain cli to **automatically** upgrade many imports. Please see documentation here <https://python.langchain.
  from langchain.embeddings import HuggingFaceEmbeddings
/content/streamlit_app2.py:6: LangChainDeprecationWarning: Importing Chroma from langchain.vectorstores is deprecated. Please rep

>> from langchain.vectorstores import Chroma

with new imports of:

>> from langchain_community.vectorstores import Chroma
You can use the langchain cli to **automatically** upgrade many imports. Please see documentation here <https://python.langchain.
  from langchain.vectorstores import Chroma
/content/streamlit_app2.py:12: LangChainDeprecationWarning: The class `HuggingFaceEmbeddings` was deprecated in LangChain 0.2.2 a
  embeddings_10_percent = HuggingFaceEmbeddings(
/content/streamlit_app2.py:17: LangChainDeprecationWarning: The class `Chroma` was deprecated in LangChain 0.2.9 and will be remo
  vector_db_10_percent = Chroma(
2025-03-22 07:12:24.626 Examining the path of torch.classes raised:
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/streamlit/web/bootstrap.py", line 345, in run
    if asyncio.get_running_loop().is_running():
       ^^^^^^^^^^^^^^^^^^^^^^^^^^
RuntimeError: no running event loop

During handling of the above exception, another exception occurred:
```