# Building A RAG System with Gemma, MongoDB and Open Source Models

Authored By: Richmond Alake

## Step 1: Installing Libraries

The shell command sequence below installs libraries for leveraging open-source large language models (LLMs), embedding models, and database interaction functionalities. These libraries simplify the development of a RAG system, reducing the complexity to a small amount of code:

- PyMongo: A Python library for interacting with MongoDB that enables functionalities to connect to a cluster and query data stored in collections and documents.
- Pandas: Provides a data structure for efficient data processing and analysis using Python
- Hugging Face datasets: Holds audio, vision, and text datasets
- Hugging Face Accelerate: Abstracts the complexity of writing code that leverages hardware accelerators such as GPUs. Accelerate is leveraged in the implementation to utilise the Gemma model on GPU resources.
- Hugging Face Transformers: Access to a vast collection of pre-trained models
- Hugging Face Sentence Transformers: Provides access to sentence, text, and image embeddings.

```
!pip install datasets pandas pymongo sentence_transformers
!pip install -U transformers
# Install below if using GPU
!pip install accelerate
```

## Step 2: Data sourcing and preparation

The data utilised in this tutorial is sourced from Hugging Face datasets, specifically the AIatMongoDB/embedded_movies dataset.

```
# Load Dataset
from datasets import load_dataset
import pandas as pd

# https://huggingface.co/datasets/AIatMongoDB/embedded_movies
dataset = load_dataset("AIatMongoDB/embedded_movies")

# Convert the dataset to a pandas dataframe
dataset_df = pd.DataFrame(dataset["train"])

dataset_df.head(5)
```

```
{"summary":"{\n  \"name\": \"dataset_df\",\n  \"rows\": 1500,\n  
\"fields\": [\n    {\n      \"column\": \"num_mflix_comments\",\n      
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 
27,\n        \"min\": 0,\n        \"max\": 158,\n        
\"num_unique_values\": 40,\n        \"samples\": [\n          117,\n          
134,\n          124\n        ],\n        \"semantic_type\": \"\",\n        
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 
\"genres\",\n      \"properties\": {\n        \"dtype\": \"object\",\n        
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"countries\",\n      
\"properties\": {\n        \"dtype\": \"object\",\n        
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"directors\",\n      
\"properties\": {\n        \"dtype\": \"object\",\n        
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"fullplot\",\n      \"properties\": 
{\n        \"dtype\": \"string\",\n        \"num_unique_values\": 
1409,\n        \"samples\": [\n          \"An undercover cop 
infiltrates a gang of thieves who plan to rob a jewelry store.\",\n          
\"Godzilla returns in a brand-new movie that ignores all preceding 
movies except for the original with a brand new look and a powered up 
atomic ray. This time he battles a mysterious UFO that later 
transforms into a mysterious kaiju dubbed Orga. They meet up for the 
final showdown in the city of Shinjuku.\",\n          \"Relationships 
become entangled in an emotional web.\"\n        ],\n        
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"writers\",\n      \"properties\": 
{\n        \"dtype\": \"object\",\n        \"semantic_type\": \"\",\n        
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 
\"awards\",\n      \"properties\": {\n        \"dtype\": \"object\",\n        
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"runtime\",\n      \"properties\": 
{\n        \"dtype\": \"number\",\n        \"std\": 
42.09038552453906,\n        \"min\": 6.0,\n        \"max\": 1256.0,\n        
\"num_unique_values\": 139,\n        \"samples\": [\n          152.0,\
n          127.0,\n          96.0\n        ],\n        
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"type\",\n      \"properties\": {\n        
\"dtype\": \"category\",\n        \"num_unique_values\": 2,\n        
\"samples\": [\n          \"series\",\n          \"movie\"\
n        ],\n        \"semantic_type\": \"\",\n        
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": 
\"rated\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 12,\n        \"samples\": [\n          
\"TV-MA\",\n          \"TV-14\"\n        ],\n        
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"metacritic\",\n      
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 
16.861995960390892,\n        \"min\": 9.0,\n        \"max\": 97.0,\n        
\"num_unique_values\": 83,\n        \"samples\": [\n          50.0,\n
97.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"poster\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1368,\n        \"samples\": [\n
\"https://m.media-amazon.com/images/M/MV5BNWE5MzAwMjQtNzI1YS00YjZhLTkx
NDItM2JjNjM3ZjI5NzBjXkEyXkFqcGdeQXVyMTQxNzMzNDI@._V1_SY1000_SX677_AL_.
jpg\",\n
\"https://m.media-amazon.com/images/M/MV5BMTgwNjIyNTczMF5BMl5BanBnXkFt
ZTcwODI5MDkyMQ@@._V1_SY1000_SX677_AL_.jpg\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"languages\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"imdb\",\n      \"properties\": {\n
\"dtype\": \"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"plot\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1429,\n        \"samples\": [\n          \"A
New York City architect becomes a one-man vigilante squad after his
wife is murdered by street punks in which he randomly goes out and
kills would-be muggers on the mean streets after dark.\",\n
\"As the daring thief Ars\\u00e8ne Lupin (Duris) ransacks the homes of
wealthy Parisians, the police, with a secret weapon in their arsenal,
attempt to ferret him out.\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"cast\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"plot_embedding\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"title\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1435,\n        \"samples\": [\n
\"Turbo: A Power Rangers Movie\",\n          \"Neon Genesis
Evangelion: Death & Rebirth\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"dataset_df"}
```

The operations within the following code snippet below focus on enforcing data integrity and quality.

1. The first process ensures that each data point's `fullplot` attribute is not empty, as this is the primary data we utilise in the embedding process.
2. This step also ensures we remove the `plot_embedding` attribute from all data points as this will be replaced by new embeddings created with a different embedding model, the `gte-large`.

```
# Data Preparation

# Remove data point where plot coloumn is missing
```

```
dataset_df = dataset_df.dropna(subset=["fullplot"])
print("\nNumber of missing values in each column after removal:")
print(dataset_df.isnull().sum())

# Remove the plot_embedding from each data point in the dataset as we
are going to create new embeddings with an open source embedding model
from Hugging Face
dataset_df = dataset_df.drop(columns=["plot_embedding"])
dataset_df.head(5)
```

```
Number of missing values in each column after removal:
num_mflix_comments      0
genres                  0
countries               0
directors              12
fullplot                0
writers                13
awards                  0
runtime                14
type                    0
rated                 279
metacritic            893
poster                 78
languages               1
imdb                    0
plot                    0
cast                    1
plot_embedding          1
title                   0
dtype: int64
```

```
{"summary":"{\n  \"name\": \"dataset_df\",\n  \"rows\": 1452,\n
\"fields\": [\n    {\n      \"column\": \"num_mflix_comments\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
27,\n        \"min\": 0,\n        \"max\": 158,\n
\"num_unique_values\": 40,\n        \"samples\": [\n          117,\n
134,\n          124\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"genres\",\n      \"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"countries\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"directors\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"fullplot\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\":
1409,\n        \"samples\": [\n          \"An undercover cop
```

```
infiltrates a gang of thieves who plan to rob a jewelry store.\",\n
\"Godzilla returns in a brand-new movie that ignores all preceding
movies except for the original with a brand new look and a powered up
atomic ray. This time he battles a mysterious UFO that later
transforms into a mysterious kaiju dubbed Orga. They meet up for the
final showdown in the city of Shinjuku.\",\n          \"Relationships
become entangled in an emotional web.\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"writers\",\n      \"properties\":
{\n        \"dtype\": \"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"awards\",\n      \"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"runtime\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 42.5693352357647,\
n        \"min\": 6.0,\n        \"max\": 1256.0,\n
\"num_unique_values\": 137,\n        \"samples\": [\n          60.0,\n
151.0,\n          110.0\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"type\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"series\",\n          \"movie\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"rated\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\": 12,\
n        \"samples\": [\n          \"TV-MA\",\n          \"TV-14\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"metacritic\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
16.855402595666057,\n        \"min\": 9.0,\n        \"max\": 97.0,\n
\"num_unique_values\": 83,\n        \"samples\": [\n          50.0,\n
97.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"poster\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1332,\n        \"samples\": [\n
\"https://m.media-amazon.com/images/M/MV5BMTQ2NTMxODEyNV5BMl5BanBnXkFt
ZTcwMDgxMjA0MQ@@._V1_SY1000_SX677_AL_.jpg\",\n
\"https://m.media-amazon.com/images/M/MV5BMTY5OTg1ODk0MV5BMl5BanBnXkFt
ZTcwMTEwMjU1MQ@@._V1_SY1000_SX677_AL_.jpg\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"languages\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"imdb\",\n      \"properties\": {\n
\"dtype\": \"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"plot\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1409,\n        \"samples\": [\n          \"An
undercover cop infiltrates a gang of thieves who plan to rob a jewelry
```

```
store.\",\n          \"Godzilla saves Tokyo from a flying saucer that
transforms into the beast Orga.\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"cast\",\n      \"properties\": {\n
\"dtype\": \"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"title\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1391,\n        \"samples\": [\n
\"Superhero Movie\",\n          \"Hooper\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"dataset_df"}
```

## Step 3: Generating embeddings

**The steps in the code snippets are as follows:**

1. Import the `SentenceTransformer` class to access the embedding models.
2. Load the embedding model using the `SentenceTransformer` constructor to instantiate the `gte-large` embedding model.
3. Define the `get_embedding` function, which takes a text string as input and returns a list of floats representing the embedding. The function first checks if the input text is not empty (after stripping whitespace). If the text is empty, it returns an empty list. Otherwise, it generates an embedding using the loaded model.
4. Generate embeddings by applying the `get_embedding` function to the "fullplot" column of the `dataset_df` DataFrame, generating embeddings for each movie's plot. The resulting list of embeddings is assigned to a new column named embedding.

*Note: It's not necessary to chunk the text in the full plot, as we can ensure that the text length remains within a manageable range.*

```python
from sentence_transformers import SentenceTransformer

# https://huggingface.co/thenlper/gte-large
embedding_model = SentenceTransformer("thenlper/gte-large")


def get_embedding(text: str) -> list[float]:
    if not text.strip():
        print("Attempted to get embedding for empty text.")
        return []

    embedding = embedding_model.encode(text)

    return embedding.tolist()


dataset_df["embedding"] = dataset_df["fullplot"].apply(get_embedding)

dataset_df.head()
```

```
{"summary":"{\n  \"name\": \"dataset_df\",\n  \"rows\": 1452,\n
\"fields\": [\n    {\n      \"column\": \"num_mflix_comments\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
27,\n        \"min\": 0,\n        \"max\": 158,\n
\"num_unique_values\": 40,\n        \"samples\": [\n          117,\n
134,\n          124\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"genres\",\n      \"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"countries\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"directors\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"fullplot\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\":
1409,\n        \"samples\": [\n          \"An undercover cop
infiltrates a gang of thieves who plan to rob a jewelry store.\",\n
\"Godzilla returns in a brand-new movie that ignores all preceding
movies except for the original with a brand new look and a powered up
atomic ray. This time he battles a mysterious UFO that later
transforms into a mysterious kaiju dubbed Orga. They meet up for the
final showdown in the city of Shinjuku.\",\n          \"Relationships
become entangled in an emotional web.\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"writers\",\n      \"properties\":
{\n        \"dtype\": \"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"awards\",\n      \"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"runtime\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 42.5693352357647,\
n        \"min\": 6.0,\n        \"max\": 1256.0,\n
\"num_unique_values\": 137,\n        \"samples\": [\n          60.0,\n
151.0,\n          110.0\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"type\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"series\",\n          \"movie\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"rated\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\": 12,\
n        \"samples\": [\n          \"TV-MA\",\n          \"TV-14\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"metacritic\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
16.855402595666057,\n        \"min\": 9.0,\n        \"max\": 97.0,\n
\"num_unique_values\": 83,\n        \"samples\": [\n          50.0,\n
97.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"poster\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1332,\n        \"samples\": [\n
\"https://m.media-amazon.com/images/M/MV5BMTQ2NTMxODEyNV5BMl5BanBnXkFt
ZTcwMDgxMjA0MQ@@._V1_SY1000_SX677_AL_.jpg\",\n
\"https://m.media-amazon.com/images/M/MV5BMTY5OTg1ODk0MV5BMl5BanBnXkFt
ZTcwMTEwMjU1MQ@@._V1_SY1000_SX677_AL_.jpg\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"languages\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"imdb\",\n      \"properties\": {\n
\"dtype\": \"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"plot\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1409,\n        \"samples\": [\n          \"An
undercover cop infiltrates a gang of thieves who plan to rob a jewelry
store.\",\n          \"Godzilla saves Tokyo from a flying saucer that
transforms into the beast Orga.\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"cast\",\n      \"properties\": {\n
\"dtype\": \"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"title\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1391,\n        \"samples\": [\n
\"Superhero Movie\",\n          \"Hooper\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"embedding\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"dataset_df"}
```

## Step 4: Database setup and connection

MongoDB acts as both an operational and a vector database. It offers a database solution that efficiently stores, queries and retrieves vector embeddings—the advantages of this lie in the simplicity of database maintenance, management and cost.

**To create a new MongoDB database, set up a database cluster:**

1. Head over to MongoDB official site and register for a free MongoDB Atlas account, or for existing users, sign into MongoDB Atlas.

2. Select the 'Database' option on the left-hand pane, which will navigate to the Database Deployment page, where there is a deployment specification of any existing cluster. Create a new database cluster by clicking on the "+Create" button.

3. Select all the applicable configurations for the database cluster. Once all the configuration options are selected, click the “Create Cluster” button to deploy the

newly created cluster. MongoDB also enables the creation of free clusters on the "Shared Tab".

*Note: Don't forget to whitelist the IP for the Python host or 0.0.0.0/0 for any IP when creating proof of concepts.*

1. After successfully creating and deploying the cluster, the cluster becomes accessible on the 'Database Deployment' page.

2. Click on the "Connect" button of the cluster to view the option to set up a connection to the cluster via various language drivers.

3. This tutorial only requires the cluster's URI(unique resource identifier). Grab the URI and copy it into the Google Colabs Secrets environment in a variable named `MONGO_URI` or place it in a .env file or equivalent.

### 4.1 Database and Collection Setup

Before moving forward, ensure the following prerequisites are met

- Database cluster set up on MongoDB Atlas
- Obtained the URI to your cluster

For assistance with database cluster setup and obtaining the URI, refer to our guide for setting up a MongoDB cluster and getting your connection string

Once you have created a cluster, create the database and collection within the MongoDB Atlas cluster by clicking + Create Database in the cluster overview page.

Here is a guide for creating a database and collection

**The database will be named `movies`.**

**The collection will be named `movie_collection_2`.**

## Step 5: Create a Vector Search Index

At this point make sure that your vector index is created via MongoDB Atlas.

This next step is mandatory for conducting efficient and accurate vector-based searches based on the vector embeddings stored within the documents in the `movie_collection_2` collection.

Creating a Vector Search Index enables the ability to traverse the documents efficiently to retrieve documents with embeddings that match the query embedding based on vector similarity.

Go here to read more about MongoDB Vector Search Index.

```
{
 "fields": [{
     "numDimensions": 1024,
     "path": "embedding",
     "similarity": "cosine",
     "type": "vector"
   }]
}
```

The `1024` value of the numDimension field corresponds to the dimension of the vector generated by the gte-large embedding model. If you use the `gte-base` or `gte-small` embedding models, the numDimension value in the vector search index must be set to 768 and 384, respectively.

## Step 6: Establish Data Connection

The code snippet below also utilises PyMongo to create a MongoDB client object, representing the connection to the cluster and enabling access to its databases and collections.

```
import pymongo
from google.colab import userdata


def get_mongo_client(mongo_uri):
    """Establish connection to the MongoDB."""
    try:
        client = pymongo.MongoClient(mongo_uri)
        print("Connection to MongoDB successful")
        return client
    except pymongo.errors.ConnectionFailure as e:
        print(f"Connection failed: {e}")
        return None


mongo_uri = userdata.get("MONGO_URI")
if not mongo_uri:
    print("MONGO_URI not set in environment variables")

mongo_client = get_mongo_client(mongo_uri)

# Ingest data into MongoDB
db = mongo_client["movies"]
collection = db["movie_collection_2"]
```

```
Connection to MongoDB successful
```

```
# Delete any existing records in the collection
collection.delete_many({})
```

```
DeleteResult({'n': 1452, 'electionId':
ObjectId('7fffffff000000000000000c'), 'opTime': {'ts':
Timestamp(1708554945, 1452), 't': 12}, 'ok': 1.0, '$clusterTime':
```

```
{'clusterTime': Timestamp(1708554945, 1452), 'signature': {'hash': b'\
x99\x89\xc0\x00Cn!\xd6\xaf\xb3\x96\xdf\xc3\xda\x88\x11\xf5\t\xbd\xc0',
'keyId': 7320226449804230661}}, 'operationTime': Timestamp(1708554945,
1452)}, acknowledged=True)
```

Ingesting data into a MongoDB collection from a pandas DataFrame is a straightforward process that can be efficiently accomplished by converting the DataFrame into dictionaries and then utilising the `insert_many` method on the collection to pass the converted dataset records.

```
documents = dataset_df.to_dict("records")
collection.insert_many(documents)

print("Data ingestion into MongoDB completed")

Data ingestion into MongoDB completed
```

## Step 7: Perform Vector Search on User Queries

The following step implements a function that returns a vector search result by generating a query embedding and defining a MongoDB aggregation pipeline.

The pipeline, consisting of the `$vectorSearch` and `$project` stages, executes queries using the generated vector and formats the results to include only the required information, such as plot, title, and genres while incorporating a search score for each result.

```
def vector_search(user_query, collection):
    """
    Perform a vector search in the MongoDB collection based on the
user query.

    Args:
    user_query (str): The user's query string.
    collection (MongoCollection): The MongoDB collection to search.

    Returns:
    list: A list of matching documents.
    """

    # Generate embedding for the user query
    query_embedding = get_embedding(user_query)

    if query_embedding is None:
        return "Invalid query or embedding generation failed."

    # Define the vector search pipeline
    pipeline = [
        {
            "$vectorSearch": {
                "index": "vector_index",
                "queryVector": query_embedding,
                "path": "embedding",
                "numCandidates": 150,  # Number of candidate matches
to consider
                "limit": 4,  # Return top 4 matches
            }
        },
        {
            "$project": {
                "_id": 0,  # Exclude the _id field
                "fullplot": 1,  # Include the plot field
                "title": 1,  # Include the title field
                "genres": 1,  # Include the genres field
                "score": {"$meta": "vectorSearchScore"},  # Include
the search score
            }
        },
    ]

    # Execute the search
    results = collection.aggregate(pipeline)
    return list(results)
```

## Step 8: Handling user queries and loading Gemma

```
def get_search_result(query, collection):

    get_knowledge = vector_search(query, collection)

    search_result = ""
    for result in get_knowledge:
        search_result += f"Title: {result.get('title', 'N/A')}, Plot:
{result.get('fullplot', 'N/A')}\n"

    return search_result

# Conduct query with retrival of sources
query = "What is the best romantic movie to watch and why?"
source_information = get_search_result(query, collection)
combined_information = f"Query: {query}\nContinue to answer the query
by using the Search Results:\n{source_information}."

print(combined_information)
```

```
Query: What is the best romantic movie to watch and why?
Continue to answer the query by using the Search Results:
Title: Shut Up and Kiss Me!, Plot: Ryan and Pete are 27-year old best
friends in Miami, born on the same day and each searching for the
perfect woman. Ryan is a rookie stockbroker living with his psychic
Mom. Pete is a slick surfer dude yet to find commitment. Each meets
```

```
the women of their dreams on the same day. Ryan knocks heads in an
elevator with the gorgeous Jessica, passing out before getting her
number. Pete falls for the insatiable Tiara, but Tiara's uncle is mob
boss Vincent Bublione, charged with her protection. This high-energy
romantic comedy asks to what extent will you go for true love?
Title: Pearl Harbor, Plot: Pearl Harbor is a classic tale of romance
set during a war that complicates everything. It all starts when
childhood friends Rafe and Danny become Army Air Corps pilots and meet
Evelyn, a Navy nurse. Rafe falls head over heels and next thing you
know Evelyn and Rafe are hooking up. Then Rafe volunteers to go fight
in Britain and Evelyn and Danny get transferred to Pearl Harbor. While
Rafe is off fighting everything gets completely whack and next thing
you know everybody is in the middle of an air raid we now know as
"Pearl Harbor."
Title: Titanic, Plot: The plot focuses on the romances of two couples
upon the doomed ship's maiden voyage. Isabella Paradine (Catherine
Zeta-Jones) is a wealthy woman mourning the loss of her aunt, who
reignites a romance with former flame Wynn Park (Peter Gallagher).
Meanwhile, a charming ne'er-do-well named Jamie Perse (Mike Doyle)
steals a ticket for the ship, and falls for a sweet innocent Irish
girl on board. But their romance is threatened by the villainous Simon
Doonan (Tim Curry), who has discovered about the ticket and makes
Jamie his unwilling accomplice, as well as having sinister plans for
the girl.
Title: China Girl, Plot: A modern day Romeo & Juliet story is told in
New York when an Italian boy and a Chinese girl become lovers, causing
a tragic conflict between ethnic gangs.
.
```

```python
from transformers import AutoTokenizer, AutoModelForCausalLM

tokenizer = AutoTokenizer.from_pretrained("google/gemma-2b-it")
# CPU Enabled uncomment below □□
# model = AutoModelForCausalLM.from_pretrained("google/gemma-2b-it")
# GPU Enabled use below □□
model = AutoModelForCausalLM.from_pretrained("google/gemma-2b-it",
device_map="auto")

# Moving tensors to GPU
input_ids = tokenizer(combined_information,
return_tensors="pt").to("cuda")
response = model.generate(**input_ids, max_new_tokens=500)
print(tokenizer.decode(response[0]))
```

```
<bos>Query: What is the best romantic movie to watch and why?
Continue to answer the query by using the Search Results:
Title: Shut Up and Kiss Me!, Plot: Ryan and Pete are 27-year old best
friends in Miami, born on the same day and each searching for the
perfect woman. Ryan is a rookie stockbroker living with his psychic
Mom. Pete is a slick surfer dude yet to find commitment. Each meets
```

```
the women of their dreams on the same day. Ryan knocks heads in an
elevator with the gorgeous Jessica, passing out before getting her
number. Pete falls for the insatiable Tiara, but Tiara's uncle is mob
boss Vincent Bublione, charged with her protection. This high-energy
romantic comedy asks to what extent will you go for true love?
Title: Pearl Harbor, Plot: Pearl Harbor is a classic tale of romance
set during a war that complicates everything. It all starts when
childhood friends Rafe and Danny become Army Air Corps pilots and meet
Evelyn, a Navy nurse. Rafe falls head over heels and next thing you
know Evelyn and Rafe are hooking up. Then Rafe volunteers to go fight
in Britain and Evelyn and Danny get transferred to Pearl Harbor. While
Rafe is off fighting everything gets completely whack and next thing
you know everybody is in the middle of an air raid we now know as
"Pearl Harbor."
Title: Titanic, Plot: The plot focuses on the romances of two couples
upon the doomed ship's maiden voyage. Isabella Paradine (Catherine
Zeta-Jones) is a wealthy woman mourning the loss of her aunt, who
reignites a romance with former flame Wynn Park (Peter Gallagher).
Meanwhile, a charming ne'er-do-well named Jamie Perse (Mike Doyle)
steals a ticket for the ship, and falls for a sweet innocent Irish
girl on board. But their romance is threatened by the villainous Simon
Doonan (Tim Curry), who has discovered about the ticket and makes
Jamie his unwilling accomplice, as well as having sinister plans for
the girl.
Title: China Girl, Plot: A modern day Romeo & Juliet story is told in
New York when an Italian boy and a Chinese girl become lovers, causing
a tragic conflict between ethnic gangs.
.

Based on the search results, the best romantic movie to watch is
**Shut Up and Kiss Me!** because it is a romantic comedy that explores
the complexities of love and relationships. The movie is funny,
heartwarming, and thought-provoking.<eos>
```