A COMPLETE REPORT FOR MUSICAL RECOMMENDATION SYSTEM

Phase 1:

Firstly I fetched the paths of all audios of fma_large dataset.

[27]:
```
import os
import librosa
from pymongo import MongoClient

# Define the main folder
main_folder = "Documents/fma_large"

# Function to extract paths of audio files
def extract_audio_paths(folder):
    audio_paths = []
    for root, dirs, files in os.walk(folder):
        for file in files:
            if file.endswith(".mp3"):
                audio_paths.append(os.path.join(root, file))
    return audio_paths

# Extract paths of audio files
audio_files = extract_audio_paths(main_folder)

# Print the paths of audio files
for audio_file in audio_files:
    print(audio_file)
```

```
Documents/fma_large/002/002112.mp3
Documents/fma_large/002/002074.mp3
Documents/fma_large/002/002012.mp3
Documents/fma_large/002/002073.mp3
```

After that from metadata I read the tracks.csv in which there was meta data to be stored on mongo db database to be used later

Reading metadata:

```
[28]: import pandas as pd

[29]: df= pd.read_csv('Documents/fma_metadata/raw_tracks.csv')

[30]: df.head()
```

[30]:

	track_id	album_id	album_title	album_url	artist_id	artist_name	artist_url	artist_website	license_
0	2	1.0	AWOL - A Way Of Life	http://freemusicarchive.org /music/AWOL/AWOL_-_...	1	AWOL	http://freemusicarchive.org /music/AWOL/	http://www.AzillionRecords.blogspot.com	http://i.creativecor /l/by-nc-
1	3	1.0	AWOL - A Way Of Life	http://freemusicarchive.org /music/AWOL/AWOL_-_...	1	AWOL	http://freemusicarchive.org /music/AWOL/	http://www.AzillionRecords.blogspot.com	http://i.creativecor /l/by-nc-
2	5	1.0	AWOL - A Way Of Life	http://freemusicarchive.org /music/AWOL/AWOL_-_...	1	AWOL	http://freemusicarchive.org /music/AWOL/	http://www.AzillionRecords.blogspot.com	http://i.creativecor /l/by-nc-
3	10	6.0	Constant Hitmaker	http://freemusicarchive.org /music/Kurt_Vile/Co...	6	Kurt Vile	http://freemusicarchive.org /music/Kurt_Vile/	http://kurtvile.com	http://i.creativecor /l/by-nc-r
4	20	4.0	Niris	http://freemusicarchive.org /music/Chris_and_Ni...	4	Nicky Cook	http://freemusicarchive.org /music/Chris_and_Ni...	NaN	http://i.creativecor /l/by-nc-r

Then I preprocessed this csv file like making the same datatype of all values in cols especially in ***track_id*** because it was having problems. After that I wrote some functions to extract data as per stored in mongo db from each attribute of a record on basis of track_id.

```
[39]: # Function to find artist_name corresponding to given audio_id
def find_artist_name(track_id):
    print(track_id)
    result = meta_data[meta_data['track_id'] == track_id]['artist_name']
    if len(result) > 0:
        return f"{result.iloc[0]}"
    else:
        return "NaN"
```

```
[40]: # Function to find tags corresponding to given track_id
def find_tags(track_id):
    result = meta_data[meta_data['track_id'] == track_id]['tags']
    if not result.empty:
        return result.values[0]  # Assuming 'tags' column contains only one value
    else:
        return None
```

```
[41]: # Function to find artist_name corresponding to given audio_id
def find_genres(track_id):
    print(track_id)
    result = meta_data[meta_data['track_id'] == track_id]['track_genres']
    if len(result) > 0:
        return f"{result.iloc[0]}"
    else:
        return "NaN"
```

Now comes the main and most important task to define schema and store data into mongDB to get when required 🙂

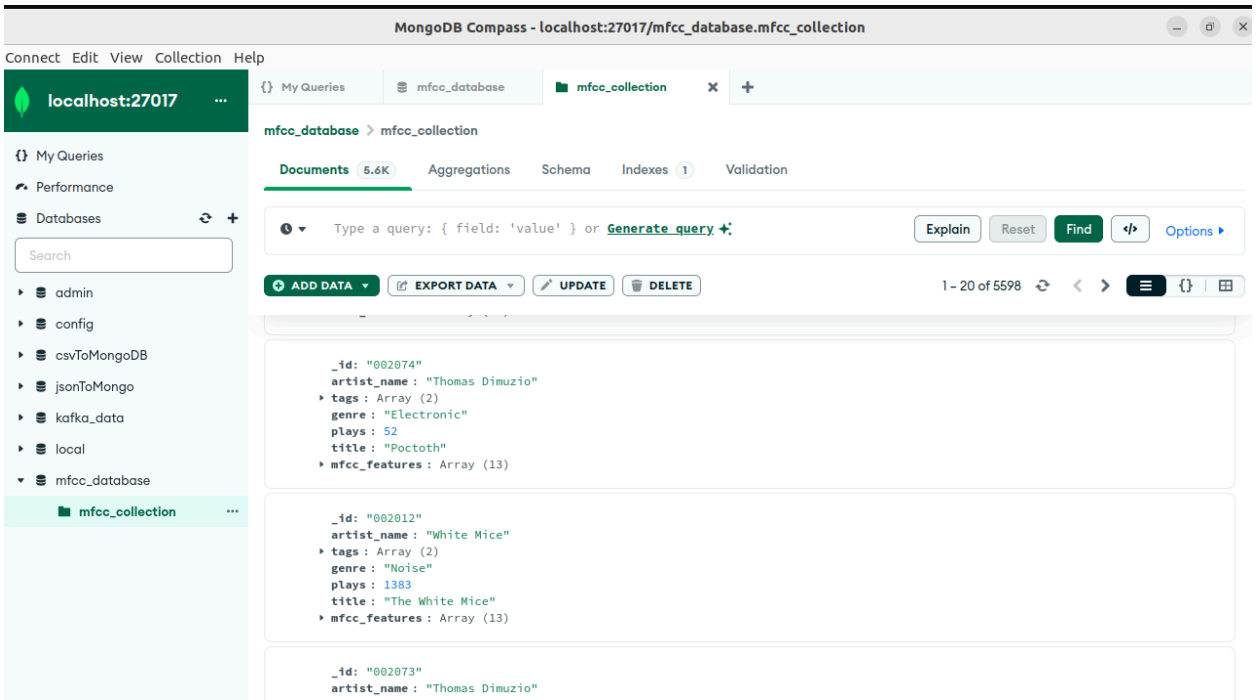
```python
def process_and_upload_to_mongodb(audio_files, chunk_size=10):
    # Connect to MongoDB
    client = MongoClient('localhost', 27017)
    db = client['mfcc_database']
    collection = db['mfcc_collection']

    for i in range(0, len(audio_files), chunk_size):
        chunk = audio_files[i:i+chunk_size]
        for audio_file in chunk:
            try:
                # Extract MFCC features
                features = extract_mfcc(audio_file)

                # Create document to insert into MongoDB
                document = {
                    "_id": audio_file[24:audio_file.find('.')],# Use file path as _id
                    "artist_name": find_artist_name(int(audio_file[24:audio_file.find('.')])),
                    "tags": list(find_tags(int(audio_file[24:audio_file.find('.')]))),
                    "genre": find_genres(int(audio_file[24:audio_file.find('.')])),
                    "plays": int(find_track_listens(int(audio_file[24:audio_file.find('.')]))),
                    "title": find_track_title(int(audio_file[24:audio_file.find('.')])),
                    "mfcc_features": features.tolist()  # Convert numpy array to list for JSON serialization
                }

                # Insert document into MongoDB
                collection.insert_one(document)
            except Exception as e:
                print(f"Error processing {audio_file}: {e}")
```

Here are the results that data is stored in mongodb.



Phase 2:

Now is the time to collect data from mongoDB and to do play with Spark 😀

So for that we defined a properSchema. Note that we used pySpark and pyMongo to do work with

```
[30]: from pyspark.sql import SparkSession

[31]: # Create Spark session
      spark = SparkSession.builder.appName('fma_recommendation_system').getOrCreate()

[32]: from pymongo import MongoClient

[33]: # Set up MongoDB connection
      client = MongoClient("mongodb://localhost:27017")

[34]: db = client['mfcc_database']
      collection = db['mfcc_collection']

[35]: from pyspark.sql.types import StructType, StructField, StringType, IntegerType, ArrayType, FloatType

      # Define the schema for the dataframe
      schema = StructType([
          StructField('_id', StringType(), True),
          StructField('artist_name', StringType(), True),
          StructField('tags', ArrayType(StringType()), True),  # Changed tags to ArrayType(StringType())
          StructField('genre', StringType(), True),
          StructField('plays', IntegerType(), True),
          StructField('title', StringType(), True),
          StructField('mfcc_features', ArrayType(FloatType(), True))
      ])
```

Then we fetched 10000 files from our spark dataframe 🙂

```
# Get data from collection, limited to 1000 documents
data = collection.find().limit(10000)
```

After that we create a spark dataframe from this data

```
# Convert data into a Spark dataframe using the defined schema
df = spark.createDataFrame(list(data), schema=schema)
```

```
df.show()
```

```
+------+--------------------+------+---------------+-----+--------------------+--------------------+
|   _id|         artist_name|  tags|          genre|plays|               title|       mfcc_features|
+------+--------------------+------+---------------+-----+--------------------+--------------------+
|002112|       Lucky Dragons|[[, ]]|  Audio Collage|  140|            Untitled|[-115.737465, 158...|
|002074|      Thomas Dimuzio|[[, ]]|     Electronic|   52|             Poctoth|[-450.57495, 178....|
|002012|          White Mice|[[, ]]|          Noise| 1383|      The White Mice|[-13.433411, 136....|
|002073|      Thomas Dimuzio|[[, ]]|     Electronic|   61|           Skullshop|[-368.97177, 77.6...|
|002071|      Thomas Dimuzio|[[, ]]|     Electronic|  140|          Blind Lion|[-371.8308, 170.4...|
|002008|Weather (from Chi...|[[, ]]|Field Recordings|  187|            Track 12|[-426.25522, 150....|
|002105|       Lucky Dragons|[[, ]]|  Audio Collage| 7027|          Untitled 6|[-142.02298, 200....|
|002098|Death Sentence: P...|[[, ]]|    Avant-Garde|  365|Here Come The Ghosts|[-72.996765, 172....|
|002069|     caUSE co-MOTION|[[, ]]|           Punk|  377| stop standing still|[-135.30476, 152....|
|002000|Weather (from Chi...|[[, ]]|Field Recordings|  100|            Track 04|[-318.25314, 162....|
|002021|           Yuma Nora|[[, ]]|    Avant-Garde|   81|                  04|[-124.55883, 127....|
|002001|Weather (from Chi...|[[, ]]|Field Recordings|   86|            Track 05|[-392.88358, 181....|
|002006|Weather (from Chi...|[[, ]]|Field Recordings|   96|            Track 10|[-388.24988, 195....|
|002004|Weather (from Chi...|[[, ]]|Field Recordings|   86|            Track 08|[-335.7767, 201.6...|
|002003|Weather (from Chi...|[[, ]]|Field Recordings|  101|            Track 07|[-329.8288, 169.6...|
|002126|          Bolmongani|[[, ]]|           Rock|  169|          Mergatroid|[-48.60065, 164.5...|
|002014|             Xiu Xiu|[[, ]]|     Indie-Rock|  687|Lyxes: Leave this...|[-106.57192, 138....|
|002010| What Cheer? Brigade|[[, ]]|           Jazz| 3756|          Green Eyes|[-137.40826, 142....|
|002077|      Thomas Dimuzio|[[, ]]|     Electronic|   70|          Southshore|[-214.44177, 195....|
|002009|     Weirdo Begeirdo|[[, ]]|           Rock|   69|           Swamputee|[-140.56325, 192....|
+------+--------------------+------+---------------+-----+--------------------+--------------------+
only showing top 20 rows
```

Then we removed null records if any!

After that comes the time to use annoy (Approx nearest neighbors Oh Yeah) to be feeded with all the data as indexes. It passes indexes (a dataframe) for less space consumption and time efficiency. In our case we trained this on MFCC to find nearest 10 neighbors
In our case 'angular' specifies the distance metric used by the index. We used 'angular' parameter when we passed features for checking nearest neighbors. *Angular refers to the cosine similarity metric, which is well-suited for high-dimensional vector spaces. Cosine similarity measures the cosine of the angle between two vectors and is commonly used in recommendation systems to find similar items based on their feature vectors.*

```python
from annoy import AnnoyIndex
```

```python
# Initialize Annoy index
num_features = len(df.first()['mfcc_features'])
annoy_index = AnnoyIndex(num_features, 'angular')  # 'angular' distance works well with cosine similarity
```

```python
# Initialize Annoy index
num_features = len(df_pandas['mfcc_features'][0])
annoy_index = AnnoyIndex(num_features, 'angular')  # 'angular' distance works well with cosine similarity
```

After doing this we passed a feature vector of an audio and it recommended us with names and _id of the 10 nearest audios.

```python
def find_similar_items(audio_features, n=10):
    similar_items = annoy_index.get_nns_by_vector(audio_features, n)
    return [df.collect()[idx] for idx in similar_items]
```

```python
first_audio_features = df.first()['mfcc_features']
similar_items = find_similar_items(first_audio_features)
for item in similar_items:
    print(item[0], item[5])
    print()
```

```
002112 Untitled

001979 WFMU v WFMU A

001771 Seasons of Swarm

001073 Onda Tocadisco

009705 Mud On The Turtle

014335 Relic

004162 Live at WFMU (Full set)

014339 Two Invitations

014892 Rosalie

003984 We Move in Waves
```

As here everything is ready to go:

Phase 3:

Then we created a flask web app 🙂
When you will click on any audio to listen it will recommend you other relevant audios tho so the the mage is here

i211377 Music Recomendation system

ID: 014945 | Title: Dapayk Solo

ID: 014949 | Title: Dapayk Solo

ID: 003634 | Title: Foot Village

ID: 004796 | Title: Hank IV

ID: 004277 | Title: Tatsuya Nakatani

ID: 008647 | Title: Chamomile

ID: 003857 | Title: Los Fancy Free

ID: 003972 | Title: Mod Fun

ID: 014693 | Title: Jonathan Coulton

ID: 008851 | Title: EAT!

We are done. Thanks for your time 😀!