DS4300 HW5: Song Recommendation System
By: Jai Gollapudi

Sampling Technique

My sampling technique aimed to create a manageable subset of the data while ensuring representation from the target album for the recommendation system. Below are the steps I took:

1. **Identify Genre(s):** The genre(s) associated with "Is This It" was identified.

2. **Filter by Genre:** The dataset was filtered to exclude any songs by The Strokes and then further filtered to only include songs from the same genre(s) as "Is This It."

3. **Random Sampling:** From this genre-specific subset, a random sample of 10,000 songs was to be taken to ensure a manageable dataset size for processing and analysis. Since the number of songs in this genre-specific subset was 2000, the algorithm went with the lower value of 2000. Below is a pseudocode representation of the sampling technique:

   nrows = Number of Rows

   If nrows (genre-specific subset) > 10,000, algorithm samples 10,000 rows
   If nrows (genre-specific subset) < 10,000, algorithm samples nrows

4. **Include Target Album:** Songs from "Is This It" were specifically included in the final sample to serve as the base for similarity comparisons, ensuring no duplication of these songs within the sample.

The Full Graph Data Model

Node Properties and Labels:

- **Label:** Song
- **Properties:**
    - track_id
    - artist
    - album_name
    - track_name
    - danceability
    - energy
    - key
    - loudness
    - mode
    - speechines
    - acousticness
    - instrumentalness
    - liveness
    - valence
    - tempo
    - time_signature.
    - track_genre

Relationships:

Type: ‘SIMILAR’: Connects two Song nodes and indicates the similarity between them with properties.

Properties:

- rank: The ranking of similarity where a lower number indicates closer similarity.
- distance: The Euclidean distance between songs based on their audio features, where a lower distance indicates higher similarity.

Recommendation Algorithm

The recommendation algorithm operates on the premise of finding songs similar to those in "Is This It" by The Strokes, excluding any songs by The Strokes themselves to diversify musical recommendations.

1. **Base for Comparison:** Songs from "Is This It" serve as the reference points for finding similar songs.

2. **Similarity Calculation:** The similarity between songs is quantified using a Euclidean distance metric calculated across various Spotify audio features after normalizing the features.

3. **Filtering for Diversity:** To ensure recommendations span a wide range of artists, the algorithm selects only the top similar song from each distinct artist, prioritizing those with the closest similarity (smallest distance).

4. **Limiting Recommendations:** Only the top 5 unique recommendations are chosen to be presented as the final set, aiming to provide a manageable yet diverse selection of new music to explore.

Cypher Queries

```
// Importing songs from CSV to create Song nodes (Change URL Path)
LOAD CSV WITH HEADERS FROM
'http://localhost:11001/project-d691db92-81f1-424b-b5b0-809ace844ef8/sampled_songs_genre_filtered.csv' AS row
CREATE (:Song {
track_id: row.track_id,
artist: row.artists,
album_name: row.album_name,
track_name: row.track_name,
danceability: toFloat(row.danceability),
energy: toFloat(row.energy),
key: toInteger(row.key),
loudness: toFloat(row.loudness),
mode: toInteger(row.mode),
speechiness: toFloat(row.speechiness),
acousticness: toFloat(row.acousticness),
instrumentalness: toFloat(row.instrumentalness),
liveness: toFloat(row.liveness),
valence: toFloat(row.valence),
tempo: toFloat(row.tempo),
time_signature: toInteger(row.time_signature),
genre: row.track_genre
});
```

```
// Creating an index on track_id for Song nodes to speed up MATCH operations
CREATE INDEX FOR (song:Song) ON (song.track_id);


// Creating SIMILAR relationships between songs fro CSV (Change URL path)
LOAD CSV WITH HEADERS FROM
'http://localhost:11001/project-d691db92-81f1-424b-b5b0-809ace844ef8/top_similar_songs.csv' AS row
MATCH (song1:Song {track_id: row.track_id_1}), (song2:Song {track_id: row.track_id_2})
MERGE (song1)-[:SIMILAR {rank: toInteger(row.rank), distance: toFloat(row.distance)}]->(song2);


// Generating five distinct song recommendations
MATCH (song:Song {artist: 'The Strokes', album_name: 'Is This
It'})-[similarity:SIMILAR]->(recommendedSong:Song)
WHERE NOT recommendedSong.artist = 'The Strokes'
WITH recommendedSong.artist AS Artist, recommendedSong, similarity
ORDER BY similarity.distance ASC
WITH Artist, COLLECT(recommendedSong)[0] AS TopRecSong, COLLECT(similarity)[0] AS
TopSimilarity
RETURN DISTINCT Artist, TopRecSong.album_name AS Album, TopRecSong.track_name AS Title,
TopSimilarity.distance AS Distance
LIMIT 5;


// Selecting the most similar song from unique artists to "Is This It" by The Strokes, excluding
The Strokes' songs, and ordering the recommendations by similarity. It ensures a diverse set of
recommendations by limiting to one song per artist based on the closest similarity metric.
MATCH (central:Song {artist: 'The Strokes', album_name: 'Is This It'})
WITH central
MATCH (central)-[similarity:SIMILAR]->(recommendedSong:Song)
WHERE NOT recommendedSong.artist = 'The Strokes'
WITH recommendedSong.artist AS Artist, central, recommendedSong, similarity
ORDER BY similarity.distance ASC
WITH Artist, COLLECT(central)[0] AS CentralSong, COLLECT(recommendedSong)[0] AS
TopRecSong, COLLECT(similarity)[0] AS TopSimilarity
WITH CentralSong, TopRecSong, TopSimilarity
RETURN CentralSong, COLLECT(TopRecSong), COLLECT(TopSimilarity)
```

Running the Program

- Preparation: Before running the program, make sure Python is installed on your system and all required packages (pandas, scipy) are installed.
- Execution: Navigate to the directory containing main.py and run it using a Python interpreter. This script will process the Spotify data and generate two CSV files: sampled_songs.csv and top_similar_songs.csv (Note: I've also submitted these files so you can test with them).
- Importing into Neo4j:
    - Place the generated CSV files into the Neo4j's import directory.
    - Obtain the file URL path that points to the CSV files in the import directory.
    - Modify the provided Cypher queries in the import section to use the correct file paths specific to you.
    - Execute the Cypher queries in Neo4j Browser or Neo4j Desktop to import the data, create the nodes, relationships, and generate the recommendations.
- Viewing Recommendations and Graph:
    - Run the recommendation query to view the top 5 song recommendations based on similarity to "Is This It" by The Strokes.
    - Execute the visualization query to see the graph representation of these recommendations in the Neo4j Browser.