***Group: 7 (Sakshit Sharma, Jeel Mitesh Tandel, Srishti Srishti)***

*Section: 1 (Introduction)*

The Database structure is tailored to efficiently manage and track critical information of the platform of Spotify Streaming Data. It encompasses tables dedicated to users, tracks, albums, artists , listening history and playlists. Each table serves a distinct purpose and establishes relationships with others, ensuring organized data retrieval and management.

1. Users

This table holds important information about the users including userID, Username, Email, date of birth.

Primary key UserID

1. Tracks

This table holds important information about the tracks including trackID, trackName, artist, album, genre

Primary key trackID

1. Playlists

This table holds important information about the playlists including PlaylistId, PlaylistName, OwnerID, trackID, public or private, number of tracks

Primary key PlaylistId

Foreign key trackID

*Explanation of the Data (Spotify Streaming Data):*

The Spotify streaming data application is meticulously engineered to navigate the intricate landscape of music streaming, offering a comprehensive database structure tailored to the diverse requirements of a digital music platform. It mirrors the intricate relationships found within the world of music, seamlessly managing data pertaining to users, tracks, artists, playlists, and user engagement.

For instance, the 'Users' table retains details of clients, facilitating streamlined communication and song choice. The 'Tracks' table encompasses critical data about the new tracks that are added, number of times you have listened to the tracks, artist of the tracks, which album they are a part of . The 'Playlists' table keeps track of the owner’s ID, track ID , whether or no the playlist is public or private and number of tracks in the playlists.

RELATIONSHIPS

1.Users and Playlists

One to many

Users can have multiple playlists but one playlist can be associated with one user only.

2.Playlists and Tracks

Many to many

Playlists can have multiple tracks and tracks can be a part of many playlists as well.

*Section 2: Group Membership and Tasks*-Sakshit sharma: [shar0771@algoquinlive.com](mailto:shar0771@algoquinlive.com)

- Jeel Mitesh Tandel: [tand0019@algonquinlive.com](mailto:tand0019@algonquinlive.com)

- Srishti Srishti: [sris0007@algonquinlive.com](mailto:sris0007@algonquinlive.com)

*Section 3: Historical Data Framework and Multi-valued Fields*

In our database system, we plan to introduce multi-valued historical data fields in the following tables and relationships:

1. Playlists - Contains (relationship)
2. Tracks - Is-related-to (relationship)
3. User - Is-a (relationship)

For each of these fields, we will create a set of triggers and stored procedures to handle INSERTS, UPDATES, and DELETES, and to keep a history of changes using timestamps. This will allow us to track changes and updates in our database system over time.

**Database Schema**

**Entity Descriptions**

**. Users: Stores user information**

**.Tracks: Stores info such as artists, albums, genre**

**.Playlists: Stores playlist information**

**. Users\_History**, **Tracks\_History**, **Playlists\_History**: **Tables for tracking historical data changes.**

**STEPS**

Creating the Tables

1.User History

A screenshot of a computer code

Description automatically generated

2. Tracks History

A screenshot of a computer

Description automatically generated

3.Playlist History

A screenshot of a computer

Description automatically generated

**Creating Trigger playlist**

**A screenshot of a computer

Description automatically generated**

Creating trigger tracks

A screenshot of a computer

Description automatically generated

Creating trigger user

A screenshot of a computer

Description automatically generated

**TRIGGERS**

**. TRG\_PLAYLISTS:** captures new data insertions, updations and deletions in playlists

**.TRG\_TRACKS:** captures new data insertions, updations and deletions in tracks

**.TRG\_USER:** captures new data insertions, updations and deletions in users

***Views:***

*UserView:*

*A screenshot of a computer

Description automatically generated*

*Playlistsview:*

*A screenshot of a music player

Description automatically generated*

*Tracksview:*

A screenshot of a music album

Description automatically generated

***Access:***

***Relationship (Data Model Diagram):***

A screenshot of a computer

Description automatically generated