Data Management and Database Design
Spring 2024
P3: Database Design, Logical Model

Group 13:
- Anirudha Joshi
- Samarjeet Ganesh Chavan
- Preeti Kulkarni
- Taranjot Dang
- Dandan Zhu

MUSIC ANALYTICS MANAGEMENT SYSTEM

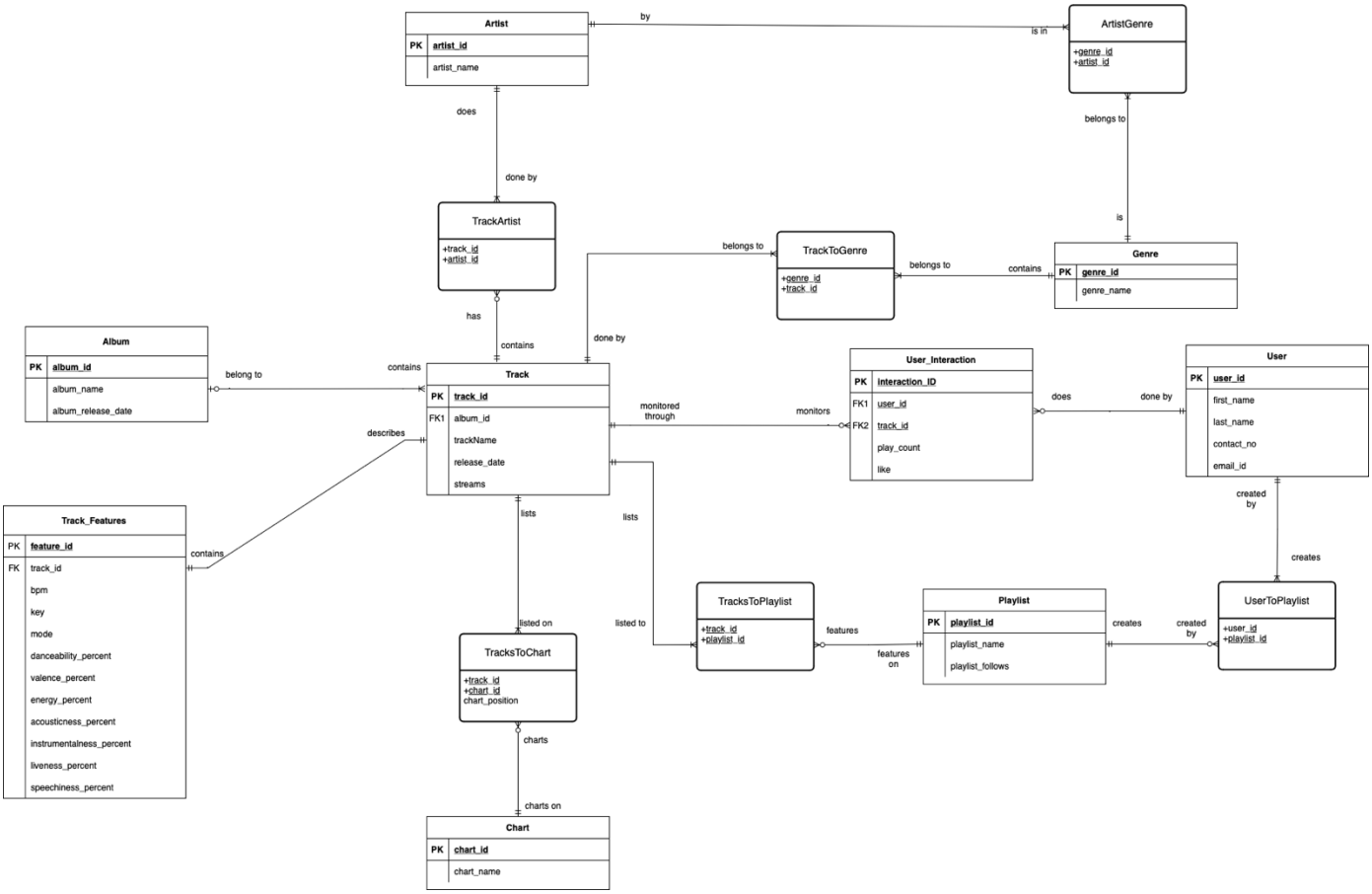
ENTITIES:

1. **Tracks:** Contains information about each track, such as track name, release date (year, month, day), streams.
2. **Artists:** Stores artist-related information. Each artist has a unique identifier and name. Tracks may be linked to one or more artists.
3. **Albums:** While the dataset does not explicitly include album names, if available, this entity could store album-related information and link to the tracks and artists involved.
4. **Playlists:** Information about playlists that include the track, such as playlist count and possibly the playlist names or IDs if available.
5. **Charts**: Details tracks' presence on Spotify charts, including the number of times it appeared on charts and stream counts.
6. **Genres:** If genre data is available for each track or artist, this entity can store genre names.
7. **User:** This entity has the details for the users such as name, contact_no and email.
8. **User_Interaction:** This entity could log user interactions with tracks, such as plays, likes, and adds to playlists, if such data were available, supporting detailed user behavior analytics.
9. **Track_Features:** Focuses on the musical and audio features of tracks, such as BPM, key, mode, and various percentages (danceability, valence, energy, etc.), enabling analyses on music trends and preferences.
10. **TrackArtist**: An associative entity indicating which users have created or follow which playlists, enabling many-to-many relationships between tracks and artists.
11. **ArtistsGenre**: An associative entity indicating which users have created or follow which playlists, enabling many-to-many relationships between artists and genres.
12. **TracksGenre**: An associative entity indicating which users have created or follow which playlists, enabling many-to-many relationships between tracks and genres.
13. **TracksToChart**: An associative entity indicating which users have created or follow which playlists, enabling many-to-many relationships between tracks and charts.
14. **TracksToPlaylist**: An associative entity indicating which users have created or follow which playlists, enabling many-to-many relationships between tracks and playlist.
15. **UserToPlaylist**: An associative entity indicating which users have created or follow which playlists, enabling many-to-many relationships between user and playlist.

RELATIONSHIPS:

- Tracks to Artists (Many-to-Many): A track can have multiple artists, and an artist can have multiple tracks.
- Tracks to Albums (Many-to-One): Each track belongs to one album, but an album can contain multiple tracks. This assumes album data is available and included in your schema.
- Tracks to Playlists (Many-to-Many): A track can be included in multiple playlists, and a playlist can contain multiple tracks.
- Tracks to Genres (Many-to-Many): A track can belong to multiple genres, and a genre can include many tracks. This relationship would require a junction table if genre data were available and included.
- Tracks to Chart (Many-to-Many): A track can be included in multiple charts, and a chart can contain multiple tracks.
- Artists to Genres (Many-to-Many): An artist can produce works in multiple genres, and a genre can encompass works by many artists. This relationship also would likely require a junction table for proper management.
- Tracks to Track_Features (One-to-One): Each track has a set of features (like BPM, key, mode, etc.) that are unique to it.
- User_Interaction to Tracks (Many-to Optional One): Assuming user interaction data is available, each interaction (like play, like, add to playlist) is associated with one track, but a track can have many interactions.
- User to Playlist (One-to Optional Many): A user can create multiple playlists, and a playlist can be created by multiple users.
- User to User_Interaction (One-to Optional Many): A user can have multiple user interaction, and a user interaction can be associated to 1 user.

E-R Diagram:


Artist
PK artist_id
artist_name
by
is in
ArtistGenre
+genre_id
+artist_id
does
done by
belongs to
is
TrackArtist
+track_id
+artist_id
belongs to
TrackToGenre
+genre_id
+track_id
belongs to
contains
Genre
PK genre_id
genre_name
has
contains
done by
Album
PK album_id
album_name
album_release_date
belong to
contains
Track
PK track_id
FK1 album_id
trackName
release_date
streams
monitored through
monitors
User_Interaction
PK Interaction_ID
FK1 user_id
FK2 track_id
play_count
like
does
done by
User
PK user_id
first_name
last_name
contact_no
email_id
describes
Track_Features
PK feature_id
FK track_id
bpm
key
mode
danceability_percent
valence_percent
energy_percent
acoustiness_percent
instrumentalness_percent
liveness_percent
speechiness_percent
contains
lists
lists
created by
creates
listed on
listed to
TracksToChart
+track_id
+chart_id
chart_position
TracksToPlaylist
+track_id
+playlist_id
features
features on
Playlist
PK playlist_id
playlist_name
playlist_follows
creates
created by
UserToPlaylist
+user_id
+playlist_id
charts
charts on
Chart
PK chart_id
chart_name