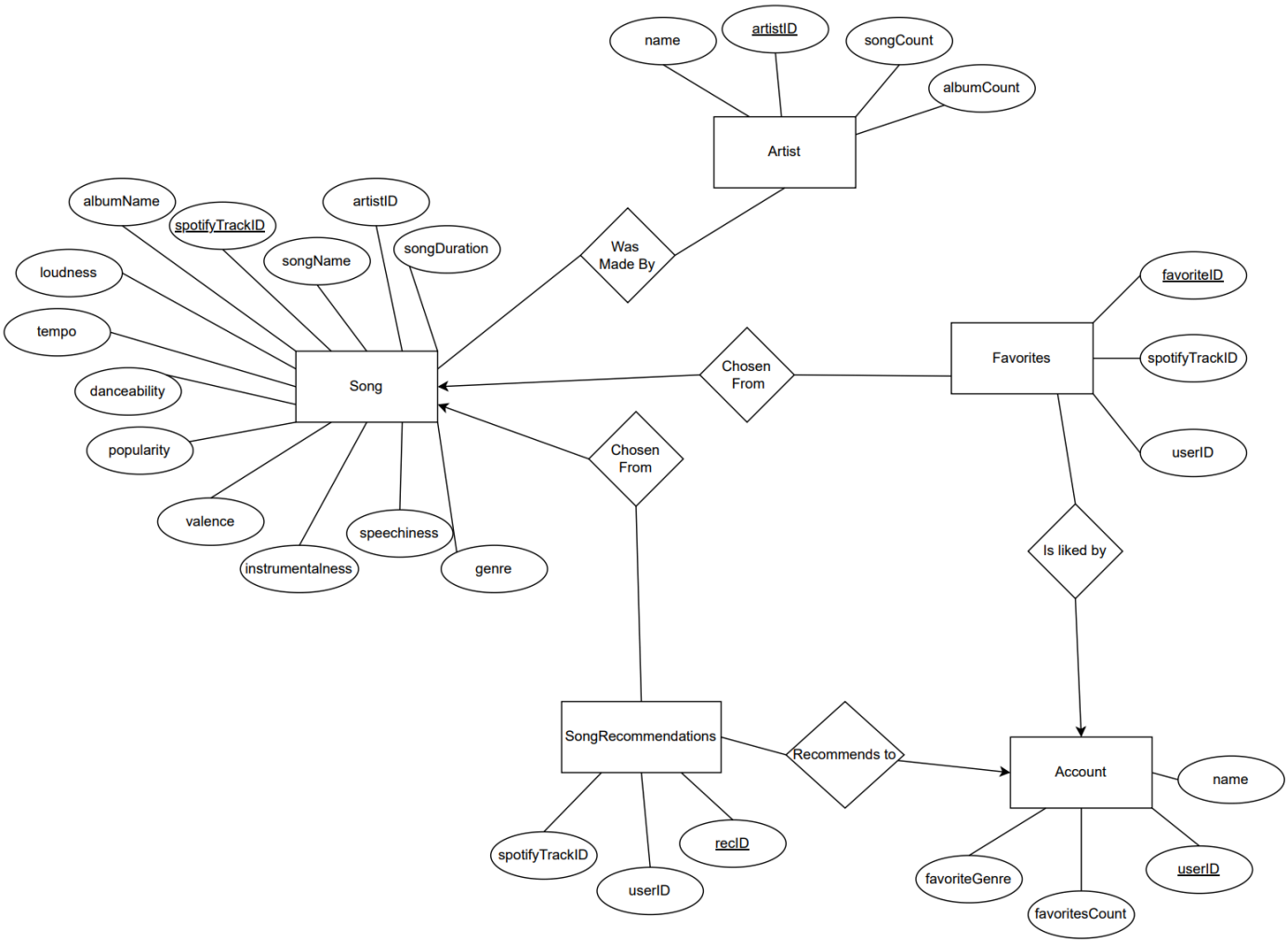
ER Diagram



Entities

1. **Artist**
    - **Description:** This entity keeps track of artists data (artistID, number of songs, number of albums). This is better as an entity since each artist contains information about themselves that is not relevant to each song.
    - **Assumption:** Every artist row contains a different artist, with a unique artistID. Artists can have multiple songs.
2. **Song**
    - **Description:** Keeps track of information on each song ranging from loudness to song duration. This is its own entity as songs should be able to be stored as individual records.
    - **Assumption:** Every song has an artist, every song can be contained in multiple song recommendations, and multiple favorites. Each song has a unique songID.

3. **Account**
    - **Description:** This entity keeps track of user data, from userID to to their name and favorite genre. This should be its own entity since each user needs to have their own data, making this essential to not be connected to other entities.
    - **Assumption:** Every user has one account, that contains one favorite genre, and one name. userID is unique to each account making it the primary key.
4. **Favorite**
    - **Description:** This stores all songs that were favorited by a user. This is a unique entity because we want to track all different songs that were marked by a user.
    - **Assumption:** Each favorite has a unique favoriteID. A user can have multiple favorite songs. A song can have multiple favorites as different users favorite them.
5. **Song Recommendations**
    - **Description:** This stores all song recommendations for a user. This is its own entity because we want to track all different songs that can be recommended to a user. This stores song name, album name and the user it is connected to.
    - **Assumption:** Each song recommendation has a unique ID. A user can have multiple song recommendations. A song can have multiple song recommendations as they are recommended to different users.

Relationships

1. **Artist to Song**
    a. **Many to Many**- This is because you can have a song with multiple artists
    b. **Assumption** - The relationship is based on how a song can have a single artist or various artists working together on a song's production. It is not possible for a song to have no artist.
2. **Account to Favorite**
    a. **One to Many** - This is because a person can have any amount of favorite songs grouped up in different tables based on their categories
    b. **Assumption** - This relationship is based on the constraint of the website, and how a person has multiple playlists with different songs they like organized in their own way. A user can have no favorite table for any amount they want to make.
3. **Favorite to Song**
    a. **Many to Many** - There can be any amount of favorite tables with any amount of songs in it

    b. **Assumption** - This relationship is based on the constraints of our website and how a user can make multiple favorite tables with any number of songs in it. They can have 0 tables with 0 songs, 1 table with all their songs, or multiple tables with any amount of songs.
4. **Recommendation to Account**
    a. Many to One - There can be any amount of song recommendations made to a single account
    b. Assumption - This relationship is based on the constraint of our website, and how any amount of recommendations can be made to a single user.
5. **Recommendation to Song**
    a. Many to One - A song can be recommended multiple times to different users
    b. Assumption - This relationship is based on the constraint of our website, and how a single song can be recommended multiple times to multiple people based on their preferences.

Normalization

We decided to go for 3NF because it offers more flexibility than BNCF with a simpler to understand and implement. We assessed that BNCF was overly stringent, splitting tables further, leading to more complex joins, and potentially more difficult query optimization.

For the Artist table the primary key is artistID and from artistID we can derive all the other attributes: name, songCount, and albumCount. This makes artistID a super key and there are also no transitive dependencies so it is in 3NF.

For the Song table the primary key is spotifyTrackID, and from artistID we can derive the other attributes songName, albumName, songDuration, loudness, tempo, songGenre, speechiness, instrumentalness, valence, songPopularity, and danceability. For artistID we have a foreign key that references the Artist table artistID. This makes artistID and spotifyTrackID the super key and there are also no transitive dependencies so it is in 3NF.

For the Account table the primary key is userID and from userID we can derive all the other attributes name, favoritesCount, and favoriteGenre. This makes userId a superkey so there are also no transitive dependencies so it is in 3NF.

For the Favorite table the primary key is favoriteID and the other attributes are foreign keys (userID references Account.userID and spotifyTrackID references Song.spotifyTrackID) so there are no non-key attributes so the superkey is the favoriteID, userID, and spotifyTrackID therefore there are no transitive dependencies so it is in 3NF.

For the Song Recommendation table the primary key is recID and the other attributes are foreign keys (userID references Account.userID and spotifyTrackID references Song.spotifyTrackID) so there are no non-key attributes so the superkey is the favoriteID, userID, and spotifyTrackID therefore there are no transitive dependencies so it is in 3NF.

Logical Design (relational schema)

Artist (
    name: VARCHAR(100),
    artistID: VARCHAR(100) [PK],
    songCount: INT,
    albumCount: INT
)

Song (
    spotifyTrackID: VARCHAR(100) [PK],
    songName: VARCHAR(100),
    artistID: VARCHAR(100) [FK to Artist.artistID] ,
    albumName: VARCHAR(100),
    songDuration: INT,
    loudness: FLOAT,
    tempo: FLOAT,
    genre: VARCHAR(100),
    speechiness: FLOAT,
    instrumentalness: FLOAT,
    valence: FLOAT,
    popularity: INT,
    danceability: FLOAT,
)

Account (
    name: VARCHAR(100),
    userID: VARCHAR(100) [PK],
    favoritesCount: INT,
    favoriteGenre: VARCHAR(100)
)

Favorites (
    favoriteID: VARCHAR(100) [PK]
    userID: VARCHAR(100) [FK to Account.userID],
    spotifyTrackID: VARCHAR(100) [FK to Song.spotifyTrackID]
)

SongRecommendations (
    recID: VARCHAR(100) (PK),
    userID: VARCHAR(100) [FK to Account.userID],
    spotifyTrackID: VARCHAR(100) [FK to Song.spotifyTrackID]
)