Project Part 3

Part 3 (a): Describing the database.

Section 1: Overview of the dataset

1a) What kind of data will you be working with?
Ans- I am going to be working on data available on Spotify. For instance, the songs, artists, albums, and charts available on Spotify so that I can make any kind of conclusions about the trends of songs on Spotify.

1b) Why are these data interesting?
Ans- I think this data is interesting because I can use it to derive relationships between different types of artists, or the type of genre an artist has an affinity towards, and much more. I chose it because it's a versatile dataset and a lot of information can be generated from this dataset.

1c) What kinds of information would you like to generate from the database you will build? You can provide a couple of example questions here.
Ans- This data can be used to create a database that can derive the following information.
1. I would like to generate information about which genre of music is the most popular.
2. Which artist prefers which genre of music.
3. What genre of music has the greatest number of artists.
And much more.

2a) Description of the dataset:
1. What types of information are contained within the proposed database?

Ans- The proposed database is going to contain information available in the Spotify library. There are going to be 4 entities in the database, and they are going to store information about the songs on Spotify, the songs that charted to the top on Spotify in the year 2022, the artists on Spotify and the various albums available on Spotify. The goal is to derive any relationships between the entities to come up with conclusions about the data in the dataset.

The references have been added at the end of the document.

Section 2: Description of the tables

SONG – This table is going to be a table containing data about the various songs available on Spotify.

The primary key is going to be song_id and the foreign key is going to be artist_id

```
CREATE TABLE IF NOT EXISTS song(
    track_id INT AUTO_INCREMENT,
    track_name VARCHAR(100) unique,
    artist_id INT,
    track_album_name VARCHAR(100),
    song_genre VARCHAR(100),
    danceability DOUBLE,
    energy DOUBLE,
    song_key INT,
    song_mode INT,
    PRIMARY KEY(track_id),
    FOREIGN KEY(artist_id) references artist(artist_id)
);
```

There are going to be 150 rows and 10 columns in the table.

CHART: which is a table that contains data about the top songs of the year 2022 on Spotify.
The primary key is going to be chart_id.
There are going to be no foreign keys in this table, that is because it has been derived from a different dataset obtained from Kaggle.
There are going to be 100 rows and 9 columns in the table.

```
CREATE TABLE IF NOT EXISTS chart(
    chart_id INT AUTO_INCREMENT,
    artist_name VARCHAR(100),
    track_name VARCHAR(100),
    peak_rank INT,
    weeks_on_chart INT,
    danceability DOUBLE,
    energy DOUBLE,
    track_key INT,
    track_mode INT,
    PRIMARY KEY(chart_id)
);
```

ARTIST: This table is going to contain information about the various artists on Spotify.
The primary key is artist_id.
Each artist can have multiple song and each song can have multiple artists.

There are going to be 100 rows and 6 columns in the table.

```sql
CREATE TABLE IF NOT EXISTS artist(
    artist_id INT AUTO_INCREMENT,
    artist_name VARCHAR(100),
    streams BIGINT,
    top_song VARCHAR(100),
    popularity INT,
    top_album VARCHAR(100),
    PRIMARY KEY(artist_id)
);
```

ALBUM: This table is going to contain information about albums released by various artists. The primary key is album_id and the foreign key is going to be artist_id which links to the table artist.

There are going to be 50 rows and 5 columns in the table after running the stored procedures.

```sql
CREATE TABLE IF NOT EXISTS album(
    album_id INT AUTO_INCREMENT,
    album_name VARCHAR(100) NOT NULL,
    artist_id INT,
    album_rank INT,
    album_genre VARCHAR(100),
    PRIMARY KEY(album_id),
    UNIQUE(album_name),
    FOREIGN KEY (artist_id) REFERENCES artist(artist_id)
);
```

Section 3: Internal Schema and Normalization

Dependency diagrams:

Song:

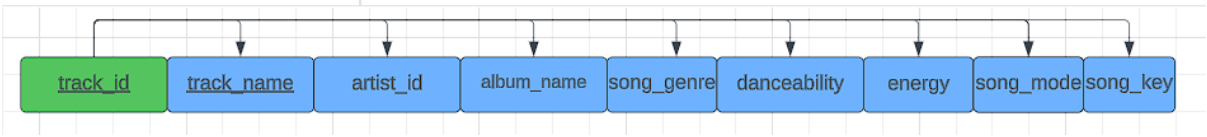
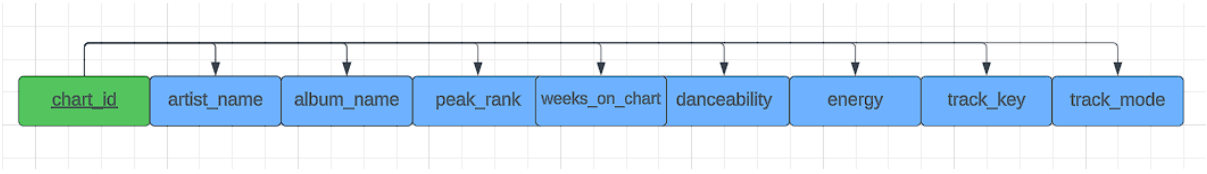
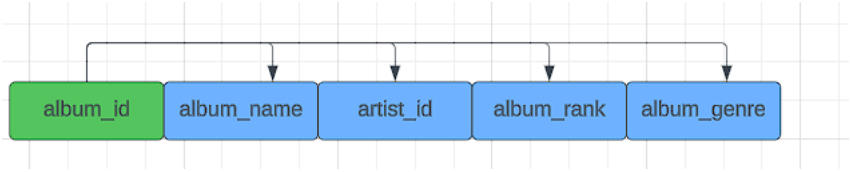


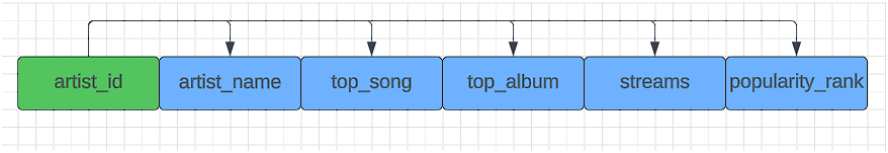
Chart:



Album:

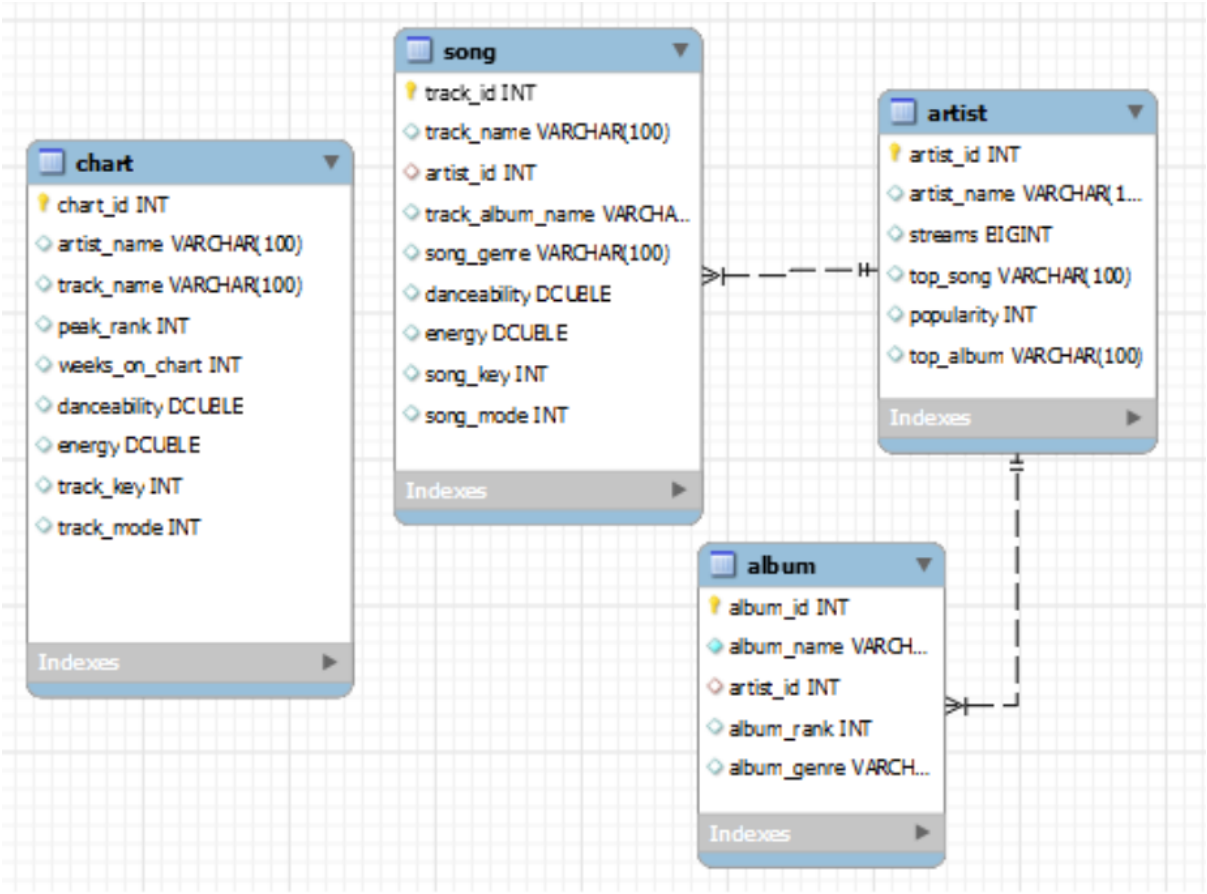


Artist:



The tables in spotifydb are in 3NF because there are no transitive or partial dependencies between the attributes.

Internal schema:



There is a one to many relationship between artist and album, and a one to many relationship between artist and song.

There is a high cardinality in table artist because each value is unique.

However, there is a medium cardinality in song, chart, and album because there are multiple entries that have the same value.

The reason why chart is not related to any table in the database is it stores completely different dataset as compared to song. However, the have been some common data derived between these tables, but it was still not linked to these tables using a foreign key because the most of values stored inside chart are different from the values stored in any other table, because of which I was not able to add a foreign key for, artist, album or track name in chart.

Part 3(b): Using the database:

Summary:

Song and Chart have been made using open-source datasets available on Kaggle for free.

Song has 9 attributes, and chart also has 9 attributes.

Song contains information about the various songs in the Spotify library.

And chart contains information about the charting songs on Spotify.

Artist and Album have been made from a mix of real and generated data. The real data has been derived from the dataset that was used to create the table song, and the rest of it has been generated using ChatGPT.

The number of attributes in artist is 6 and the number of attributes in album is 5.

Business Rules

Each table in this database has a primary key, in to ensure that each attribute has a unique identification in the database.

Furthermore, song, and album have foreign keys named as artist_id that references the primary key of the artist table.

Besides that, the tables song and album also contain the unique check constrain to ensure there are no repeated tracks or albums in the respective tables.

Additionally, there are 2 stored procedures. First stored procedure ensures that there is enough data in album to meet the assignment criteria and the second one ensures that each attribute in the table song has an artist_id, if not then that entry is deleted.

In the end, there is also a view called charting_artist that has been created to derive the subset of data in table chart that is common with artist i.e., it returns the artists in the song table that have shown on a chart in Spotify.

Stored procedure:

Insert_album: This stored procedure takes 2 attributes, the total number of entries in album, this is an INOUT variable because it the procedures takes as input the current number of entries and returns the total number of entries after running the procedure.

Besides that, it has an OUT variable which returns the number of attributes that were added after running this procedure. This is the difference between new total and previous total. To run this procedure, the CALL () functions need the INOUT variable, that stores the number of attributes/ rows in album before running the procedure, and an OUT variable that needs to be defined to get a return value.

Null_artist: This is a procedure, that has been created to test the functionality of the following procedure. This procedure updates the artist_id attribute for a few attributes in the song table to NULL so that the functionality of procedure Delete_song can be tested.

It does not have any variables.

Delete_song: This procedure is used to delete any attributes that do not have an artist_id.

Its takes 2 variables as input.

An INOUT variable used to store the number of rows in the table before running the procedure and it returns the updated number of rows in the table after the procedure is called, an OUT variable that returns the number of rows that were deleted after calling the procedure. This the difference between the initial and updated number of rows.

The Call () function requires 2 input attributes a variable storing the current number of rows and a variable that will store the OUT variable, calculated after the procedure is compiled.

Resources:

1. Sveta151. “Spotify Top Chart Songs 2022.” *Kaggle*, 5 Sept. 2022, www.kaggle.com/datasets/sveta151/spotify-top-chart-songs-2022.

2. Arvidsson, Joakim. “30000 Spotify Songs.” *Kaggle*, 1 Nov. 2023, www.kaggle.com/datasets/joebeachcapital/30000-spotify-songs.

3. Anam, Adnan. “Spotify Artist Stats.” *Kaggle*, 22 Sept. 2022, www.kaggle.com/datasets/adnananam/spotify-artist-

3. ChatGPT. This was used to generate the dataset and insert statements for artist and album.

The links for the chat has been added below,

https://chat.openai.com/c/cc5e1d63-4e2d-48bb-80dc-0d2e60dcc735

https://chat.openai.com/c/0b818c9f-1fbf-4e53-8222-2b06daa04872

https://chat.openai.com/c/b25dedc0-cde4-4503-b694-93b537bbcb71