DONE BY ZOE PHILIP IYAWA

ZOE'S

MUSIC RECOMMENDATION SYSTEM

OVERVIEW

PART 1. INTRODUCTION

PART 2. TABLE OVERVIEW

PART 3. ER DIAGRAM

PART 4. IMPLEMENTATION

PART 5. CONCLUSION

INTRODUCTION

THIS DATABASE WAS DESIGNED TO MANAGE SPOTIFY EXTENSIVE LIBRARY OF SONGS.MADE NOT JUST FOR MUSIC LOVERS BUT FOR INDIVIDUALS WHO ENJOY A CASUAL SONG ON THEIR BUS RIDE OR ON A COMMUTE TO WORK. ITS PURPOSE IS TO SUGGEST PERSONALISED ALBUMS OR TRACKS TO USERS BASED ON THEIR PREFERENCES, TRACK ATTRIBUTES AND LISTENING PATTERNS

TABLE OVERVIEW

1. **TRACKINFO**: THIS TABLE STORES INFORMATION ABOUT INDIVIDUAL TRACKS, INCLUDING ID, DURATIONS, TRACK POPULARITY AND ALBUM ID AS FOREIGN KEY.

2. **ALBUMINFO**: THIS TABLE CONTAINS INFORMATION ABOUT ALBUMS, SUCH AS ID, RELEASE DATES, AND LABEL ID AS FOREIGN KEY.

3. **ARTISTINFO**: IN THIS TABLE, YOU'LL FIND INFORMATION ABOUT ARTISTS, INCLUDING THEIR ID, NAMES, AND LABEL ID AS FOREIGN KEY.

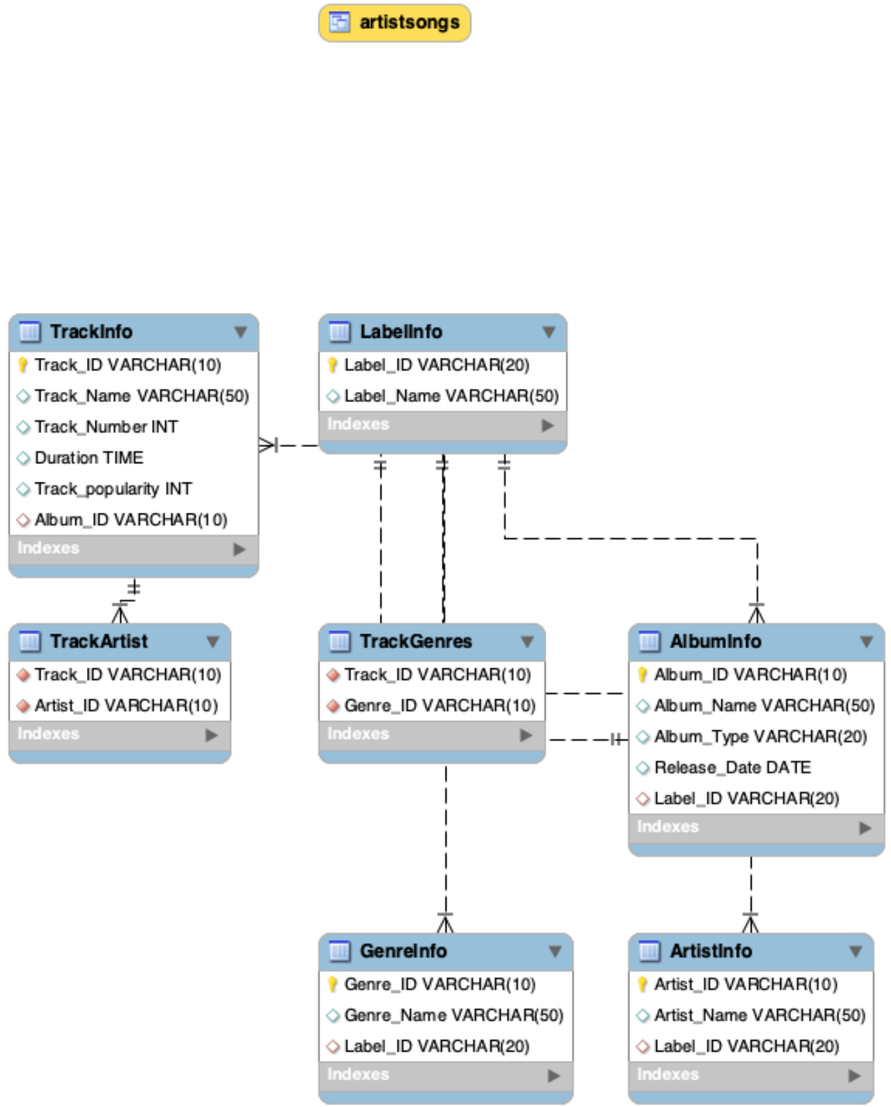
4. **GENREINFO**: THIS TABLE HOLDS INFORMATION ABOUT MUSIC GENRES, WITH LABEL ID AS FOREIGN KEY

5. **LABELINFO**: HERE, YOU'LL FIND INFORMATION ABOUT ARTIST RECORD LABELS

6. **TRACKGENRE**: THIS GENRE ARE ASSOCIATED WITH EACH TRACK INDICATING WHICH GENRES ARE ASSOCIATED WITH EACH TRACK.

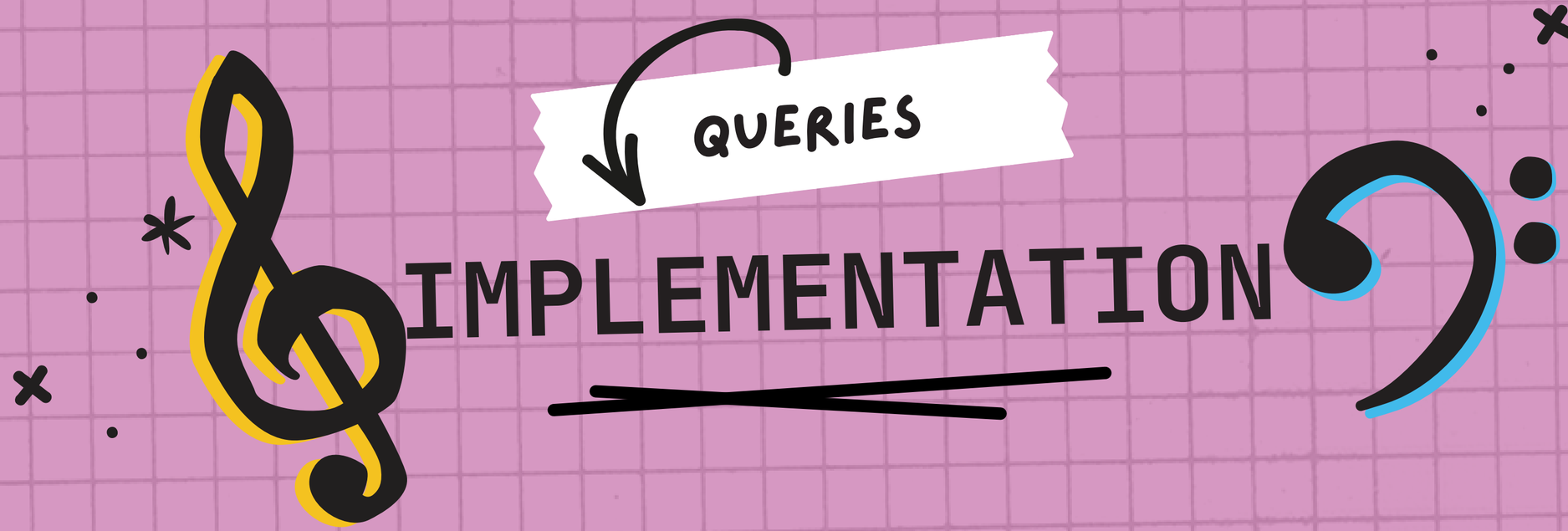
7. **TRACKARTISTS**: THIS TABLE ESTABLISHES RELATIONSHIPS BETWEEN TRACKS AND ARTISTS, INDICATING WHICH ARTISTS ARE ASSOCIATED WITH EACH TRACK.

ER DIAGRAM


artistsongs
TrackInfo
Track_ID VARCHAR(10)
Track_Name VARCHAR(50)
Track_Number INT
Duration TIME
Track_popularity INT
Album_ID VARCHAR(10)
Indexes
LabelInfo
Label_ID VARCHAR(20)
Label_Name VARCHAR(50)
Indexes
TrackArtist
Track_ID VARCHAR(10)
Artist_ID VARCHAR(10)
Indexes
TrackGenres
Track_ID VARCHAR(10)
Genre_ID VARCHAR(10)
Indexes
AlbumInfo
Album_ID VARCHAR(10)
Album_Name VARCHAR(50)
Album_Type VARCHAR(20)
Release_Date DATE
Label_ID VARCHAR(20)
Indexes
GenreInfo
Genre_ID VARCHAR(10)
Genre_Name VARCHAR(50)
Label_ID VARCHAR(20)
Indexes
ArtistInfo
Artist_ID VARCHAR(10)
Artist_Name VARCHAR(50)
Label_ID VARCHAR(20)
Indexes

QUERIES

IMPLEMENTATION



WHAT SHOULD I LISTEN TO TODAY ?

The concept

Spotify randomly suggests tracks to its user based on their preferences or listening pattern.

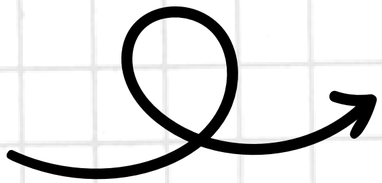
SUBQUERY

USING *SUBQUERY*, WE EXTRACTED THE MOST POPULAR TRACK WITHIN EACH ALBUM. MADE USE OF RANK STATEMENT TO ASSIGN RANKS TO ROWS BASED ON THEIR POPULARITY SCORE.

IF THE USER IS INTERESTED IN ONLY POPULAR SONGS :

```
With Popular_Tracks AS
(SELECT Track_Name,
Album_ID, Track_popularity,
RANK ()OVER(PARTITION BY
Album_ID ORDER BY
Track_popularity DESC) AS
Ranking
FROM Trackinfo
ORDER BY Album_ID)

SELECT Album_ID, Track_Name
FROM Popular_Tracks
WHERE Ranking = 1;
```



Album_ID	Track_Name
Album_1	Ancient Canyons
Album_10	Castles Made Of Sand
Album_2	Jeweled Lotus (Om Mani Peme Hum)
Album_3	Dream Healing
Album_4	See The World Burn
Album_5	All Gold
Album_6	Bright Side
Album_7	Kings Calling
Album_8	Drifting
Album_9	Wait Until Tomorrow

JOIN

IF THE USER IS INTERESTED IN TRACKS IN A SPECIFIC GENRE :

USING *JOIN STATEMENT*, WE JOINED THE TRACKINFO TABLE AND GENREINFO TABLE ON THE TRACKGENRE TABLE. WE CAN FIND THE DIFFERENT TRACKS WITHIN A SPECIFIC GENRE IN THIS CASE THE “POP GENRE”.

```
SELECT Genre_Name, Track_Name
FROM TrackGenres
JOIN Genreinfo ON
TrackGenres.Genre_ID =
Genreinfo.Genre_ID
JOIN Trackinfo ON
TrackGenres.Track_ID =
Trackinfo.Track_ID
WHERE Genre_Name = "Pop";
```

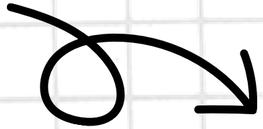
Genre_Name	Track_Name
Pop	Jeweled Lotus (Om Mani Peme Hum)
Pop	Rah

GROUP BY & HAVING

IF THE USER IS INTERESTED IN GENRES THAT HAVE MORE THAN 4 TRACKS:

USING THE *GROUP BY AND JOIN STATEMENT*, WE CAN FIND GENRES ASSOCIATED WITH MORE THAN 4 TRACKS

```
SELECT Genre_Name, Count(Track_Name)
AS num_of_tracks
FROM TrackGenres
JOIN Genreinfo ON
TrackGenres.Genre_ID =
Genreinfo.Genre_ID
JOIN Trackinfo ON
TrackGenres.Track_ID =
Trackinfo.Track_ID
GROUP BY Genre_Name
HAVING Count(Track_Name) > 4;
```



Genre_Name	num_of_tracks
Hardcore	6
Country	5

VIEWS.

IF THE USER IS INTERESTED IN SONGS BY A SPECIFIC ARTIST :

WE HAVE CREATED A *VIEW* OF TRACKS FROM DIFFERENT ARTIST. USING *JOIN STATEMENT*, WE JOINED THE TRACKINFO TABLE AND ARTISTINFO TABLE ON THE TRACKARTIST TABLE. THE VIEW CAN NOW BE USED TO EXTRACT THE TRACKS OF A PREFERRED ARTIST. FOR EXAMPLE, IN THE PICTURE BELOW THESE ARE TRACKS FROM A SPECIFIC ARTIST - THE JONAS BROTHERS, ALONG WITH THEIR POPULARITY SCORE.

```
Create VIEW Artistsongs AS SELECT
Track_Name,Artist_Name,Track_popularity
FROM TrackArtist
JOIN Trackinfo ON TrackArtist.Track_ID =
Trackinfo.Track_ID
JOIN Artistinfo ON TrackArtist.Artist_ID
= Artistinfo.Artist_ID
ORDER BY Artist_Name, Track_popularity
DESC;

SELECT Artist_Name,Track_Name
FROM artistsongs
WHERE Artist_Name = 'Jonas Brothers';
```

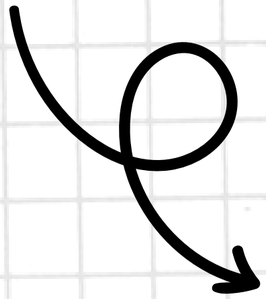
Artist_Name	Track_Name
Jonas Brothers	See The World Burn
Jonas Brothers	Little Wing
Jonas Brothers	One Rainy Wish

VIEWS(CONTI)

VIEW CAN ALSO BE USED TO IDENTIFY THE MOST POPULAR TRACK BY A SPECIFIC ARTIST. BASED ON THE PREVIOUS EXAMPLE WE HAVE BEEN ABLE TO IDENTIFY THE MOST POPULAR TRACK BY THE JONAS BROTHERS.

Artist_Name	Track_Name
Jonas Brothers	See The World Burn
Jonas Brothers	Little Wing
Jonas Brothers	One Rainy Wish

```
SELECT Artist_Name,
Track_Name
FROM artistsongs
WHERE Artist_Name =
'Jonas Brothers'
LIMIT 1;
```



Track_Name
See The World Burn

STORED FUNCTION

IF THE USER IS INTERESTED IN SONGS WITH EXCELLENT RATING :

- WE HAVE CREATED A STORED FUNCTION CALLED “POPULARITY_SCALE.” USING THE CASE FUNCTION WE CAN ASSIGN VALUES SUCH AS POOR, GOOD,VERY GOOD AND EXCELLENT TO TRACKS BASED ON THEIR POPULARITY.

```
DELIMITER ///
CREATE FUNCTION Popularity_scale (Track_popularity
INTEGER)
RETURNS VARCHAR (10)
DETERMINISTIC
BEGIN
DECLARE Scale VARCHAR (10);
CASE
WHEN Track_popularity < 20 THEN SET Scale = "POOR";
WHEN Track_popularity >= 20 AND Track_popularity <= 49
THEN SET Scale = "FAIR";
WHEN Track_popularity >= 50 AND Track_popularity <= 70
THEN SET Scale = "GOOD";
WHEN Track_popularity >= 71 AND Track_popularity <= 80
THEN SET Scale = "VERY GOOD";
WHEN Track_popularity  > 80 THEN SET Scale = "EXCELLENT";
ELSE SET Scale = "NULL";
END CASE;
RETURN Scale;

END
///
DELIMITER ;
```

STORED PROCEDURE

IF THE USER IS INTERESTED IN TRACKS FROM A SPECIFIC ARTIST :

USING *CREATE PROCEDURE STATEMENT*, WE HAVE CREATED A NEW STORED PROCEDURE CALLED RECOMMENDATIONBYARTIST. THIS HIGHLIGHTS DIFFERENT TRACKS BY ARTISTS OUR USER LISTENS TOO

```
DELIMITER //
CREATE PROCEDURE RecommendationbyArtist ()
BEGIN
SELECT Track_Name
FROM Trackinfo
JOIN TrackArtist ON Trackinfo.Track_ID =
TrackArtist.Track_ID
WHERE Artist_ID IN ('Artist_7', 'Artist_8');
END
//
DELIMITER ;
```

TRACKS FROM ARTIST_7 AND ARTIST_8.

USING CALL RecommendationbyArtist ();

Track_Name
Black Sands
Bright Side
Rumble In Kerma
The Coin
Silent Night
Squad Goals
All Gold
Gladiators

CONCLUSION

BY APPLYING DIFFERENT SQL STATEMENTS WE HAVE BEEN ABLE TO:

A.

SIFT THROUGH DATA TO DELIVER PERSONALIZED MUSIC SUGGESTIONS TAILORED TO OUR USER'S PREFERENCE

B.

INCREASE OUR UNDERSTANDING OF DATA AND SQL

C.

UNDERSTAND HOW MYSQL EFFICIENTLY ANALYSES AND FILTERS DATA TO GENERATE SUGGESTIONS


AND THAT'S A WRAP.
THANK YOU!