```
CREATE DATABASE spotify;
USE SPOTIFY;

-- Create Genre table

CREATE TABLE genres(genre_id INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
genre_name VARCHAR(50) NOT NULL
);

-- Creating artist tables

CREATE TABLE artists(artist_id INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
 stage_name VARCHAR(50) NULL, real_name VARCHAR(50) NOT NULL, genre_id INTEGER
NOT NULL,
CONSTRAINT FK_genre_id
        FOREIGN KEY (genre_id)
                REFERENCES genres(genre_id)
);

-- Creating Albums Table
 CREATE TABLE albums(album_id INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
album_name VARCHAR(50) NOT NULL, year_released INT NOT NULL,
artist_id INTEGER NOT NULL, total_tracks INTEGER NOT NULL,
CONSTRAINT FK_artist_id1
        FOREIGN KEY (artist_id)
                REFERENCES artists(artist_id)
);

 -- Creating songs table

CREATE TABLE songs(song_id INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
 artist_id INTEGER NOT NULL, song_name VARCHAR(50) NOT NULL, song_length FLOAT(2)
NOT NULL,
 album_id INTEGER NULL, streams INT NOT NULL,
CONSTRAINT FK_artist_id
        FOREIGN KEY (artist_id)
                REFERENCES artists(artist_id),
CONSTRAINT FK_album_id
        FOREIGN KEY (album_id)
                REFERENCES albums(album_id)
);
-- Creating Users Table
CREATE TABLE users(user_id INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
```

```
 user_name VARCHAR(50) NOT NULL, user_email VARCHAR(50) NOT NULL, age INT NOT
NULL,
 country VARCHAR(50) NOT NULL, most_played_artist INT NOT NULL,
 most_played_genre INT NOT NULL,
 CONSTRAINT FK_most_artist
        FOREIGN KEY (most_played_artist)
                REFERENCES artists(artist_id),
CONSTRAINT FK_most_played_genre
        FOREIGN KEY (most_played_genre)
                REFERENCES albums(album_id)
);

-- Creating Top 30 UK Table
CREATE TABLE Top_UK(Position INTEGER PRIMARY KEY NOT NULL,
track_name VARCHAR(50) NOT NULL, artist_id INT NOT NULL, streams INT NOT NULL,
 CONSTRAINT FK_top_artistUK
        FOREIGN KEY (artist_id)
                REFERENCES artists(artist_id)
);

-- Creating Top 50 US
CREATE TABLE Top_US(Position INTEGER PRIMARY KEY NOT NULL,
track_name VARCHAR(50) NOT NULL, artist_id INT NOT NULL, streams INT NOT NULL,
 CONSTRAINT FK_top_artistUS
        FOREIGN KEY (artist_id)
                REFERENCES artists(artist_id)
);

-- Creating Top 50 Global
CREATE TABLE Top_Global(Position INTEGER PRIMARY KEY NOT NULL,
track_name VARCHAR(50) NOT NULL, artist_id INT NOT NULL, streams INT NOT NULL,
 CONSTRAINT FK_top_artistGlobal
        FOREIGN KEY (artist_id)
                REFERENCES artists(artist_id)
);

-- query with order by

SELECT * FROM Top_Global ORDER BY Position;

---- Stored function for if song is in top charts

DELIMITER //
CREATE FUNCTION topchart(streams INT)
```

```
RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
DECLARE streams_for_chart VARCHAR(20) ;
IF streams<150000 THEN
SET streams_for_chart='not top chart song';
ELSEIF streams>=150000 AND streams<=2000000 THEN
SET streams_for_chart='mid top chart song';
ELSEIF streams>2000000 THEN SET streams_for_chart='top chart yay!';
END IF;
RETURN (streams_for_chart);
END//streams
DELIMITER ;
SELECT s.artist_id,s.song_name,topchart(s.streams) from songs s ;


-- query with subquery: return all albums with more than 10 tracks

SELECT a.artist_id, a.real_name
 FROM artists a
                    WHERE a.artist_id
 IN(SELECT alb.artist_id FROM albums alb
                    WHERE alb.total_tracks> 10);

-- dont allow for an email to be input into the user table without an @mail.com
DELIMITER //
CREATE TRIGGER email_validation
BEFORE INSERT ON users FOR EACH ROW
          IF new.user_email NOT LIKE '%@%'
    THEN SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT='Invalid user email';
    END IF;
END //
DELIMITER ;
 insert into users(user_name, user_email, age, country, most_played_artist,
most_played_genre)
 values('Gypsy Castillo', 'email', 67, 'Japan', 1,1  );


-- create a query to count/select all artists with no stage name and group them by genre id,
-- displaying their genre

          -- count:
```

```
SELECT count(a.real_name) as 'Number Of Artists Without Stage Name', g.genre_name as
'Genre'
FROM artists a
INNER JOIN genres g ON a.genre_id = g.genre_id
WHERE stage_name IS NULL
GROUP BY g.genre_name;


        -- artist names:

SELECT a.real_name, g.genre_name
FROM artists a
INNER JOIN genres g ON a.genre_id = g.genre_id
WHERE stage_name IS NULL
GROUP BY g.genre_name, a.real_name;


-- find all artists whose genre contains word ‘pop’
-- > concluding that the genre of pop is popular
SET @row_number = 0;

SELECT (@row_number:=@row_number + 1) AS 'Serial Number', a.real_name as Artist,
g.genre_name as Genre
FROM artists a
INNER JOIN genres g
ON a.genre_id = g.genre_id
WHERE g.genre_name LIKE '%pop%';

-- find all songs that have the character ‘!’
SELECT a.real_name, s.song_name
FROM artists a
INNER JOIN songs s ON a.artist_id = s.artist_id
WHERE s.song_name LIKE '%!%';

-- find all songs that begin with the letter ‘S’
SELECT a.real_name, s.song_name
FROM artists a
INNER JOIN songs s ON a.artist_id = s.artist_id
WHERE s.song_name LIKE 's%';

-- find all songs that made it in Top 30 US, UK and Global
SELECT DISTINCT uk.track_name
FROM Top_UK uk
INNER JOIN Top_US us
```

```
ON uk.artist_id = us.artist_id
INNER JOIN Top_Global gl
ON us.artist_id = gl.artist_id
WHERE uk.track_name = us.track_name = gl.track_name;


-- show how many users there are by age groups
SELECT
(SELECT COUNT(age) FROM users WHERE age BETWEEN 10 and 20) AS '10-20 yrs',
(SELECT COUNT(age) FROM users WHERE age BETWEEN 20 and 30) AS '20-30 yrs',
(SELECT COUNT(age) FROM users WHERE age BETWEEN 30 and 40) AS '30-40 yrs',
(SELECT COUNT(age) FROM users WHERE age BETWEEN 40 and 50) AS '40-50 yrs',
(SELECT COUNT(age) FROM users WHERE age BETWEEN 50 and 60) AS '50-60 yrs',
(SELECT COUNT(age) FROM users WHERE age BETWEEN 60 and 70) AS '60-70 yrs',
(SELECT COUNT(age) FROM users WHERE age BETWEEN 70 and 100) AS '70+ yrs';

-- which country has the most and least spotify users (and how many)?
SELECT COUNT(user_name) as 'Number of Users', country
FROM users
GROUP BY country
ORDER BY COUNT(user_name) DESC;


-- List all registered artists and their respective genres
SELECT
a.real_name, g.genre_name
FROM artists a
INNER JOIN
genres g
ON a.genre_id = g.genre_id;

-- create query to return most globally popular genre ( join artist-genre-top charts)

SELECT genre_name as Genre, COUNT(*) as Genre_Count
FROM artists a
INNER JOIN genres g
ON a.genre_id = g.genre_id
INNER JOIN top_global t
ON t.artist_id = a.artist_id
GROUP BY genre_name
ORDER BY Genre_Count DESC;

-- Compare year of album release with album popularity (streams)
```

```
SELECT ar.real_name as Artist, a.year_released, s.streams
FROM albums a
INNER JOIN songs s
ON s.album_id = a.album_id
INNER JOIN artists ar
ON ar.artist_id = s.artist_id
ORDER BY Streams DESC;


-- Using group by and having clause to display artists with more than 400,000 streams in the UK
SELECT
   a.artist_id, a.real_name as Artist, SUM(streams) as Total_Streams
FROM Top_UK uk
INNER JOIN artists a
ON a.artist_id = uk.artist_id
GROUP BY a.artist_id
HAVING Total_Streams > 400000;

-- stored procedure
DELIMITER //

CREATE PROCEDURE Message(Welcome VARCHAR(100), back VARCHAR(30), User_Name
VARCHAR(100))

BEGIN
        DECLARE WelcomeMessage VARCHAR(200);
        SET WelcomeMessage = CONCAT('Welcome', ' ', back, ' ', 'Spotify', ',', ' ', user_name, '.',
' ', 'We are the top music streaming service of the world.');
        SELECT WelcomeMessage;
END//

DELIMITER ;

CALL Message('', 'to', 'Sarah');
CALL Message('', 'back to', 'Oussama');
CALL Message('', 'to', 'Anna');
CALL Message('', 'back to', 'Tory');

DROP PROCEDURE Message;
-- creating View to show details of user's favorite artists by
-- country and query for which users have an artist that doesn't have a stage name
CREATE VIEW user_favorite_artist_details
AS SELECT u.user_name, u.country, a.real_name, a.stage_name, g.genre_name as 'favorite
artist genre'
```

```
FROM artists a
JOIN users u
ON u.most_played_artist=a.artist_id
JOIN genres g
ON a.genre_id=g.genre_id
ORDER BY u.country;

SELECT fav.user_name, fav.country FROM user_favorite_artist_details fav WHERE
fav.stage_name IS NULL ORDER BY fav.user_name;

-- event to count the amount of songs in the system longer than 4 mins
-- (but then modified per day) and store the number songs in the number_songs table
 CREATE TABLE number_songs (
  id INT PRIMARY KEY AUTO_INCREMENT,
 number_songs integer NOT NULL,
 at_time DATETIME NOT NULL
 );
set global event_scheduler=on
DELIMITER //
CREATE EVENT song_inventory
on schedule every 5 SECOND
starts NOW()
ends now()+interval 1 day
do
insert into
number_songs(number_songs, at_time)
values((select count(*) from songs s where s.song_length> 4), now()) ;
END //
DELIMITER ;
DROP EVENT SONG_INVENTORY;
drop table number_songs;
SELECT * FROM NUMBER_SONGS;


-- db diagram
-- done!
```