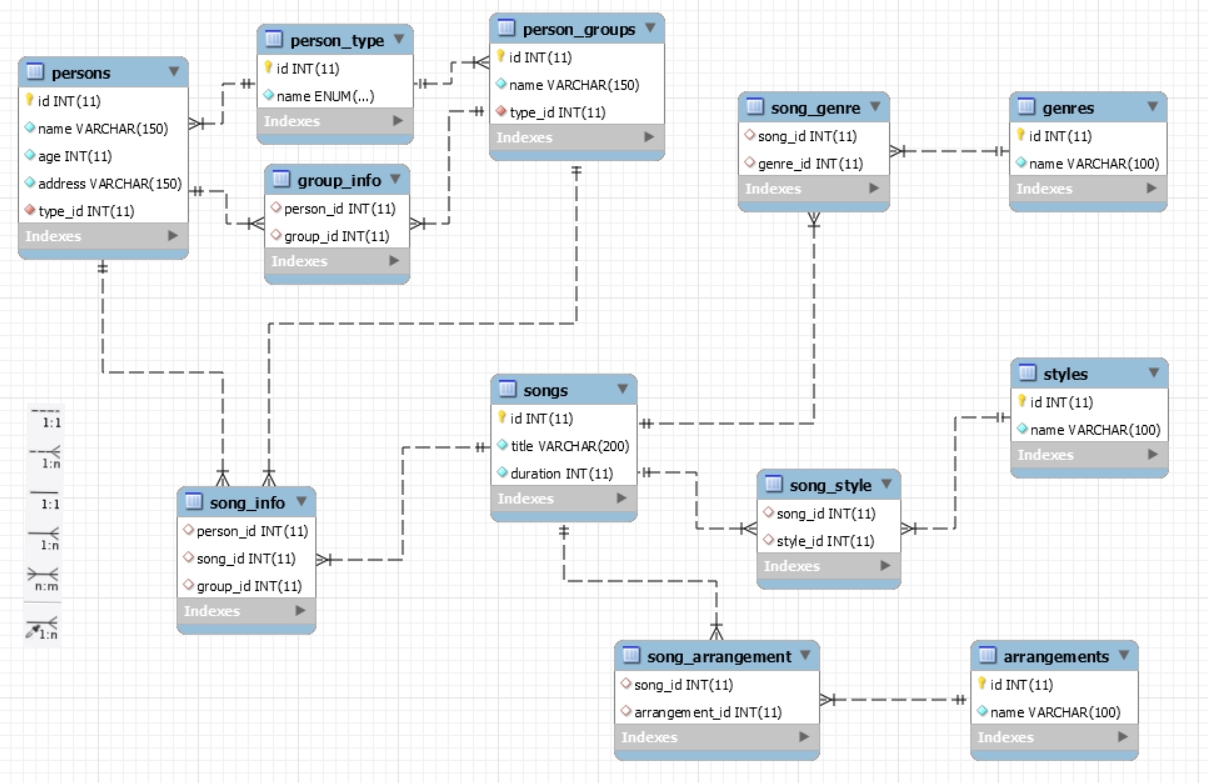
Решения

За решение на задачата ще използваме езика **MySQL**, който се изучава по време на лабораторните упражнения по дисциплината. За проектирането на базата в задача 1 ще използваме модела **ER**-диаграма (Entity Relationship Diagram).

**Задача 1:** Да се проектира база от данни и да се представи ER диаграма със съответни CREATE TABLE заявки за средата MySQL.

**Създаваме таблица person\_type:**

CREATE TABLE person\_type (

id INT AUTO\_INCREMENT PRIMARY KEY,

name ENUM('Composer', 'Artist, ‘Group') NOT NULL UNIQUE

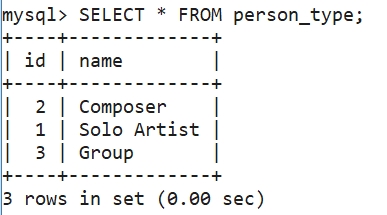
);

**Вкарваме примерни данни за person\_type:**

INSERT INTO person\_type (name) VALUES ('Artist'); -- ID = 1

INSERT INTO person\_type (name) VALUES ('Composer'); -- ID = 2

INSERT INTO person\_type (name) VALUES (‘Group’); -- ID = 3



**Създаваме таблица persons:**

CREATE TABLE persons (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(150) NOT NULL,

age INT NOT NULL,

address VARCHAR(150) NOT NULL,

type\_id INT NOT NULL,

CONSTRAINT FOREIGN KEY (type\_id) REFERENCES person\_type(id)

);

**Вкарваме примерни данни за persons:**

/\* INSERT заявки за добавяне на изпълнители \*/

INSERT INTO persons (name, age, address, type\_id) VALUES ('50 Cent', 42, '50 Cent address', 1); -- ID = 1

INSERT INTO persons (name, age, address, type\_id) VALUES ('Ice Cube', 48, 'Ice Cube address', 1); -- ID = 2

INSERT INTO persons (name, age, address, type\_id) VALUES ('Taylor Swift', 28, 'Taylor Swift address', 1); -- ID = 3

INSERT INTO persons (name, age, address, type\_id) VALUES ('Bon Jovi', 35, 'Bon Jovi address', 1); -- ID = 4

INSERT INTO persons (name, age, address, type\_id) VALUES ('Eminem', 35, 'Eminem address', 1); -- ID = 5

INSERT INTO persons (name, age, address, type\_id) VALUES ('Takeoff', 23, 'Takeoff address', 1); -- ID = 6

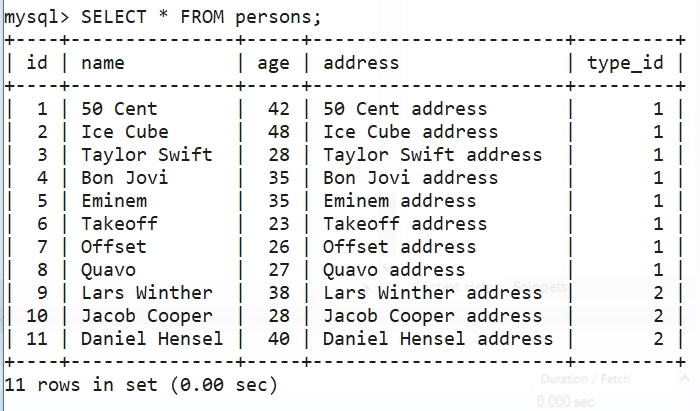
INSERT INTO persons (name, age, address, type\_id) VALUES ('Offset', 26, 'Offset address', 1); -- ID = 7

INSERT INTO persons (name, age, address, type\_id) VALUES ('Quavo', 27, 'Quavo address', 1); -- ID = 8

/\* INSERT заявки за добавяне на композитори \*/

INSERT INTO persons (name, age, address, type\_id) VALUES ('Lars Winther', 38, 'Lars Winther address', 2); -- ID = 9

INSERT INTO persons (name, age, address, type\_id) VALUES ('Jacob Cooper', 28, 'Jacob Cooper address', 2); -- ID = 10

INSERT INTO persons (name, age, address, type\_id) VALUES ('Daniel Hensel', 40, 'Jacob Cooper address', 2); -- ID = 11

**Създаваме таблица person\_groups:**

CREATE TABLE person\_groups (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(150) NOT NULL UNIQUE,

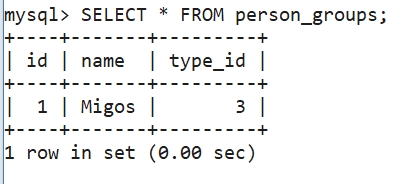
type\_id INT NOT NULL,

CONSTRAINT FOREIGN KEY (type\_id) REFERENCES person\_type(id)

);

**Вкарваме примерни данни за person\_groups:**

INSERT INTO person\_groups (name, type\_id) VALUES ('Migos', 3); -- ID = 1



**Създаваме таблица group\_info:**

CREATE TABLE group\_info (

person\_id INT,

group\_id INT,

CONSTRAINT FOREIGN KEY (person\_id) REFERENCES persons(id),

CONSTRAINT FOREIGN KEY (group\_id) REFERENCES person\_groups(id)

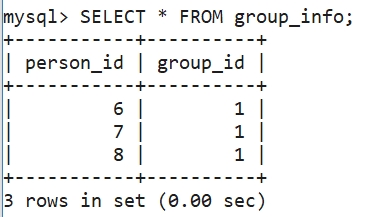
);

**Вкарваме примерни данни за group\_info:**

INSERT INTO group\_info (person\_id, group\_id) VALUES (6, 1);

INSERT INTO group\_info (person\_id, group\_id) VALUES (7, 1);

INSERT INTO group\_info (person\_id, group\_id) VALUES (8, 1);



**Създаваме таблица songs:**

CREATE TABLE songs (

id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(200) NOT NULL,

duration INT NOT NULL

);

**Вкарваме примерни данни за songs:**

INSERT INTO songs (title, duration) VALUES ('Pilot', 182); -- ID = 1

INSERT INTO songs (title, duration) VALUES ('Gangsta Rap Made Me Do It', 350); -- ID = 2

INSERT INTO songs (title, duration) VALUES ('Animal Ambition', 200); -- ID = 3

INSERT INTO songs (title, duration) VALUES ('Im The Man', 351); -- ID = 4

INSERT INTO songs (title, duration) VALUES ('Look What You Made Me Do', 249); -- ID = 5

INSERT INTO songs (title, duration) VALUES ('Its My Life', 220); -- ID = 6

INSERT INTO songs (title, duration) VALUES ('Shake It Off', 300); -- ID = 7

INSERT INTO songs (title, duration) VALUES ('Not Afraid', 290); -- ID = 8

INSERT INTO songs (title, duration) VALUES ('In Your Head', 320); -- ID = 9

INSERT INTO songs (title, duration) VALUES ('Talkin 2 Myself', 400); -- ID = 10

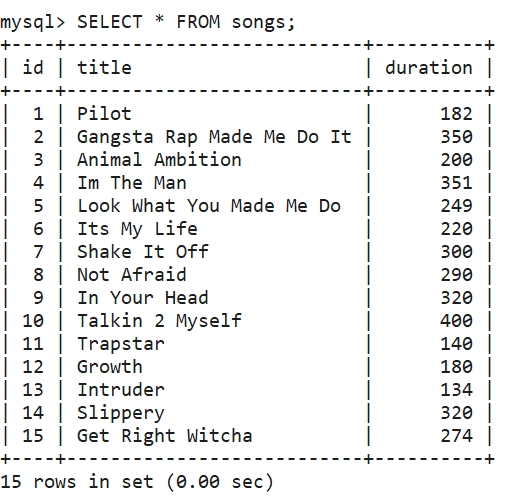
INSERT INTO songs (title, duration) VALUES ('Trapstar', 140); -- ID = 11

INSERT INTO songs (title, duration) VALUES ('Growth', 180); -- ID = 12

INSERT INTO songs (title, duration) VALUES ('Intruder', 134); -- ID = 13

INSERT INTO songs (title, duration) VALUES ('Slippery', 320); -- ID = 14

INSERT INTO songs (title, duration) VALUES ('Get Right Witcha', 274); -- ID = 15



**Създаваме таблица song\_info:**

CREATE TABLE song\_info (

person\_id INT,

song\_id INT,

group\_id INT,

CONSTRAINT FOREIGN KEY (person\_id) REFERENCES persons(id),

CONSTRAINT FOREIGN KEY (song\_id) REFERENCES songs(id),

CONSTRAINT FOREIGN KEY (group\_id) REFERENCES person\_groups(id)

);

**Вкарваме примерни данни за song\_info:**

/\* INSERT заявки за свърване на песен към даден изпълнител | група \*/

INSERT INTO song\_info (person\_id, song\_id) VALUES (1, 1);

INSERT INTO song\_info (person\_id, song\_id) VALUES (1, 3);

INSERT INTO song\_info (person\_id, song\_id) VALUES (1, 4);

INSERT INTO song\_info (person\_id, song\_id) VALUES (2, 2);

INSERT INTO song\_info (person\_id, song\_id) VALUES (3, 5);

INSERT INTO song\_info (person\_id, song\_id) VALUES (3, 7);

INSERT INTO song\_info (person\_id, song\_id) VALUES (4, 6);

INSERT INTO song\_info (person\_id, song\_id) VALUES (5, 8);

INSERT INTO song\_info (person\_id, song\_id) VALUES (5, 9);

INSERT INTO song\_info (person\_id, song\_id) VALUES (5, 10);

INSERT INTO song\_info (person\_id, song\_id) VALUES (6, 13);

INSERT INTO song\_info (person\_id, song\_id) VALUES (7, 12);

INSERT INTO song\_info (person\_id, song\_id) VALUES (8, 11);

INSERT INTO song\_info (group\_id, song\_id) VALUES (1, 14);

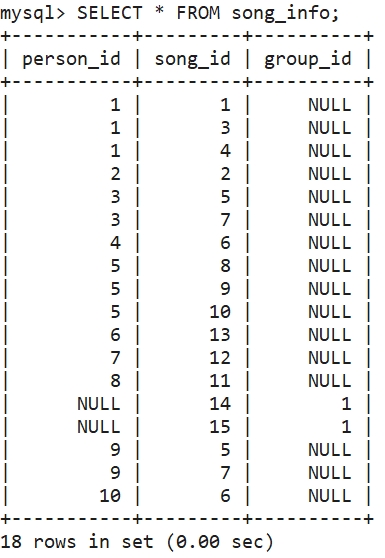
INSERT INTO song\_info (group\_id, song\_id) VALUES (1, 15);

/\* INSERT заявки за свърване на песен към даден композитор \*/

INSERT INTO song\_info (person\_id, song\_id) VALUES (9, 5);

INSERT INTO song\_info (person\_id, song\_id) VALUES (9, 7);

INSERT INTO song\_info (person\_id, song\_id) VALUES (10, 6);



**Създаваме таблица arrangements:**

CREATE TABLE arrangements (

id INT AUTO\_INCREMENT PRIMARY KEY,

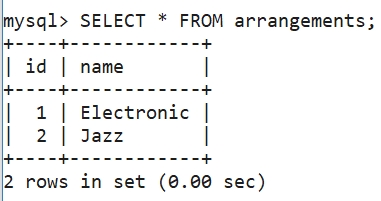
name VARCHAR(100) NOT NULL UNIQUE

);

**Вкарваме примерни данни за arrangements:**

INSERT INTO arrangements (name) VALUES ('Electronic'); -- ID = 1

INSERT INTO arrangements (name) VALUES ('Jazz'); -- ID = 2



**Създаваме таблица song\_arrangement:**

CREATE TABLE song\_arrangement (

song\_id INT,

arrangement\_id INT,

CONSTRAINT FOREIGN KEY (song\_id) REFERENCES songs(id),

CONSTRAINT FOREIGN KEY (arrangement\_id) REFERENCES arrangements(id)

);

**Вкарваме примерни данни за song\_arrangement:**

INSERT INTO song\_arrangement (song\_id, arrangement\_id) VALUES (1, 1);

INSERT INTO song\_arrangement (song\_id, arrangement\_id) VALUES (2, 1);

INSERT INTO song\_arrangement (song\_id, arrangement\_id) VALUES (3, 1);

INSERT INTO song\_arrangement (song\_id, arrangement\_id) VALUES (4, 1);

INSERT INTO song\_arrangement (song\_id, arrangement\_id) VALUES (5, 2);

INSERT INTO song\_arrangement (song\_id, arrangement\_id) VALUES (6, 2);

INSERT INTO song\_arrangement (song\_id, arrangement\_id) VALUES (7, 2);

INSERT INTO song\_arrangement (song\_id, arrangement\_id) VALUES (8, 1);

INSERT INTO song\_arrangement (song\_id, arrangement\_id) VALUES (9, 1);

INSERT INTO song\_arrangement (song\_id, arrangement\_id) VALUES (10, 1);

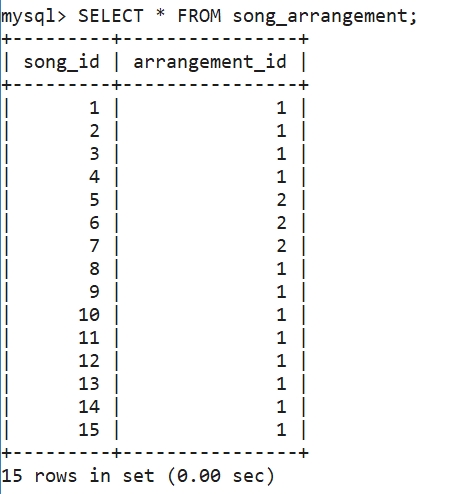
INSERT INTO song\_arrangement (song\_id, arrangement\_id) VALUES (11, 1);

INSERT INTO song\_arrangement (song\_id, arrangement\_id) VALUES (12, 1);

INSERT INTO song\_arrangement (song\_id, arrangement\_id) VALUES (13, 1);

INSERT INTO song\_arrangement (song\_id, arrangement\_id) VALUES (14, 1);

INSERT INTO song\_arrangement (song\_id, arrangement\_id) VALUES (15, 1);



**Създаваме таблица genres:**

CREATE TABLE genres (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL UNIQUE

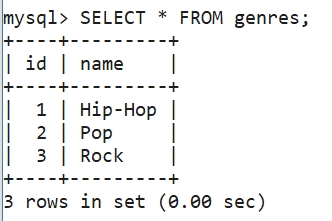
);

**Вкарваме примерни данни за genres:**

INSERT INTO genres (name) VALUES ('Hip-Hop'); -- ID = 1

INSERT INTO genres (name) VALUES('Pop'); -- ID = 2

INSERT INTO genres (name) VALUES ('Rock'); -- ID = 3



**Създаваме таблица song\_genre:**

CREATE TABLE song\_genre (

song\_id INT,

genre\_id INT,

CONSTRAINT FOREIGN KEY (song\_id) REFERENCES songs(id),

CONSTRAINT FOREIGN KEY (genre\_id) REFERENCES genres(id)

);

**Вкарваме примерни данни за song\_genre:**

INSERT INTO song\_genre (song\_id, genre\_id) VALUES (1, 1);

INSERT INTO song\_genre (song\_id, genre\_id) VALUES (2, 1);

INSERT INTO song\_genre (song\_id, genre\_id) VALUES (3, 1);

INSERT INTO song\_genre (song\_id, genre\_id) VALUES (4, 1);

INSERT INTO song\_genre (song\_id, genre\_id) VALUES (5, 2);

INSERT INTO song\_genre (song\_id, genre\_id) VALUES (6, 3);

INSERT INTO song\_genre (song\_id, genre\_id) VALUES (7, 2);

INSERT INTO song\_genre (song\_id, genre\_id) VALUES (8, 1);

INSERT INTO song\_genre (song\_id, genre\_id) VALUES (9, 1);

INSERT INTO song\_genre (song\_id, genre\_id) VALUES (10, 1);

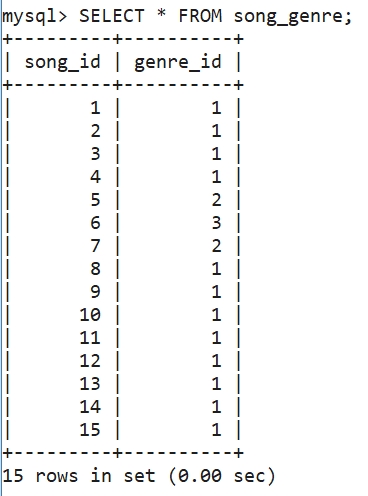
INSERT INTO song\_genre (song\_id, genre\_id) VALUES (11, 1);

INSERT INTO song\_genre (song\_id, genre\_id) VALUES (12, 1);

INSERT INTO song\_genre (song\_id, genre\_id) VALUES (13, 1);

INSERT INTO song\_genre (song\_id, genre\_id) VALUES (14, 1);

INSERT INTO song\_genre (song\_id, genre\_id) VALUES (15, 1);



**Създаваме таблица styles:**

CREATE TABLE styles (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL UNIQUE

);

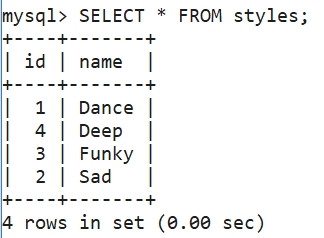
**Вкарваме примерни данни за styles:**

INSERT INTO styles (name) VALUES ('Dance'); -- ID = 1

INSERT INTO styles (name) VALUES('Sad'); -- ID = 2

INSERT INTO styles (name) VALUES('Funky'); -- ID = 3

INSERT INTO styles (name) VALUES('Deep'); -- ID = 4



**Създаваме таблица song\_style:**

CREATE TABLE song\_style (

song\_id INT,

style\_id INT,

CONSTRAINT FOREIGN KEY (song\_id) REFERENCES songs(id),

CONSTRAINT FOREIGN KEY (style\_id) REFERENCES styles(id)

);

**Вкарваме примерни данни за song\_style:**

INSERT INTO song\_style (song\_id, style\_id) VALUES (1, 1);

INSERT INTO song\_style (song\_id, style\_id) VALUES (2, 4);

INSERT INTO song\_style (song\_id, style\_id) VALUES (3, 1);

INSERT INTO song\_style (song\_id, style\_id) VALUES (4, 4);

INSERT INTO song\_style (song\_id, style\_id) VALUES (5, 1);

INSERT INTO song\_style (song\_id, style\_id) VALUES (6, 1);

INSERT INTO song\_style (song\_id, style\_id) VALUES (7, 3);

INSERT INTO song\_style (song\_id, style\_id) VALUES (8, 4);

INSERT INTO song\_style (song\_id, style\_id) VALUES (9, 4);

INSERT INTO song\_style (song\_id, style\_id) VALUES (10, 4);

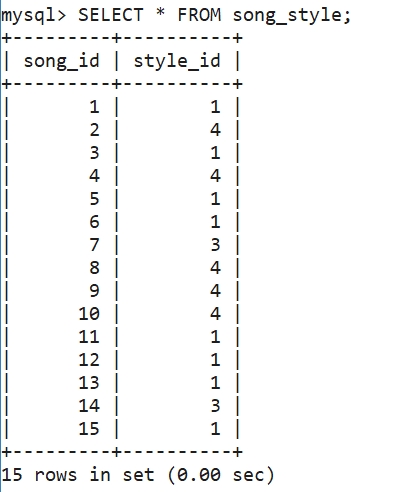
INSERT INTO song\_style (song\_id, style\_id) VALUES (11, 1);

INSERT INTO song\_style (song\_id, style\_id) VALUES (12, 1);

INSERT INTO song\_style (song\_id, style\_id) VALUES (13, 1);

INSERT INTO song\_style (song\_id, style\_id) VALUES (14, 3);

INSERT INTO song\_style (song\_id, style\_id) VALUES (15, 1);



**Задача 2:** Заявката извежда имената на песните, продължителността им и годините на изпълнителя. Ограничаващото условие е изпълнителя да е с име “Eminem”.

SELECT persons.name AS Artist, persons.age AS Age, songs.title AS Song, songs.duration AS Duration

FROM persons

JOIN songs ON persons.id IN (

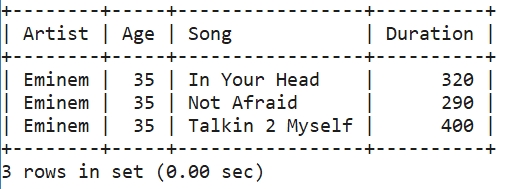
SELECT person\_id

FROM song\_info

WHERE song\_id = songs.id)

WHERE persons.type\_id = 1 AND persons.name LIKE '%Eminem%'

ORDER BY songs.title;



**Задача 3:** Заявката извежда имената на изпълнителите/групите, броя на песните на всеки един от тях и сумира продължителността им, ако броя на песните им е повече от една. Резултатите са групирани по имена на изпълнител/група и са подредени по максимален брой песни.

SELECT persons.name AS Artist, COUNT(songs.id) AS Songs, SUM(songs.duration) AS total\_duration

FROM persons

JOIN songs ON persons.id IN (

SELECT person\_id

FROM song\_info

WHERE song\_id = songs.id)

WHERE persons.type\_id = 1

GROUP by persons.name

HAVING songs > 1

UNION

SELECT person\_groups.name AS Artist, COUNT(songs.id) AS Songs, SUM(songs.duration) AS total\_duration

FROM person\_groups

JOIN songs ON person\_groups.id IN (

SELECT group\_id

FROM group\_info

WHERE group\_id IN (

SELECT group\_id

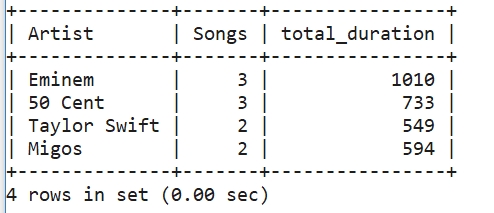
FROM song\_info

WHERE song\_id = songs.id))

GROUP by person\_groups.name

HAVING songs > 1

ORDER BY songs DESC;



**Задача 4:** Заявката извежда имена на артистите, композиторите и имената на песните, които те изпълняват/композират.

/\* FOURTH TASK - INNER JOIN \*/

SELECT persons.name AS person\_name, person\_type.name AS type, songs.title AS songTitle

FROM persons

INNER JOIN song\_info ON persons.id = song\_info.person\_id

INNER JOIN songs ON song\_info.song\_id = songs.id

INNER JOIN person\_type ON persons.type\_id = person\_type.id

ORDER BY person\_type.name DESC;



/\* FOURTH TASK - LEFT OUTER JOIN \*/

SELECT persons.name AS person\_name, person\_type.name AS type, songs.title AS song\_title

FROM persons

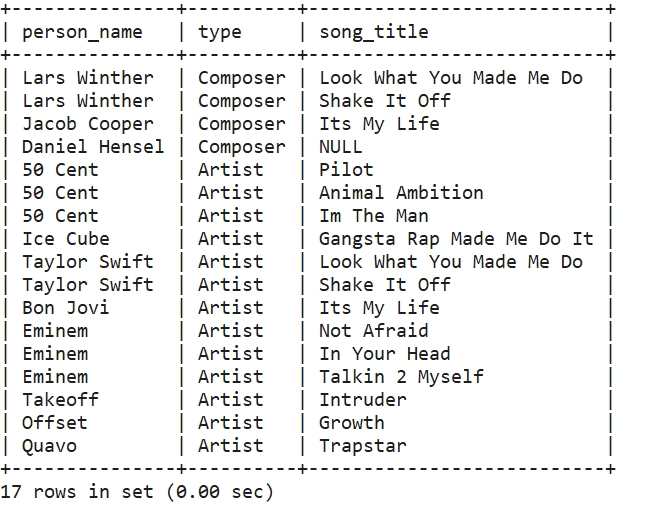
LEFT JOIN songs ON persons.id IN (

SELECT person\_id

FROM song\_info

WHERE song\_id = songs.id)

JOIN person\_type ON persons.type\_id = person\_type.id;



**Задача 5:** Заявката извежда име на композитор и брой песни, които е композирал.

/\* FIFTH TASK \*/

SELECT persons.name AS Composer, Count(songs.id) AS songs

FROM persons

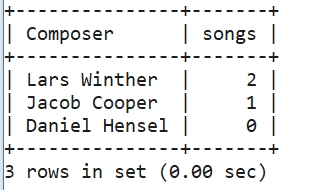
LEFT JOIN song\_info ON persons.id = song\_info.person\_id

LEFT JOIN songs ON song\_info.song\_id = songs.id

JOIN person\_type ON persons.type\_id = person\_type.id

WHERE persons.type\_id = 2

GROUP BY persons.name;



**Задача 6:** В процедурата декларираме променливи. След това и курсорът, който взима име на изпълнител/композитор/група, име на песен и нейната продължителност. След това се създава празна таблица, в която пазим резултата. Итерираме през данните от курсора и ги записваме във временната таблица. При последната итерация запазваме състоянието на цикъла в таблицата. Взимаме информацията от временната таблица и я изтриваме.

/\* SIX TASK \*/

-- USE song\_seller;

DROP PROCEDURE IF EXISTS CursorTask;

DELIMITER |

CREATE PROCEDURE CursorTask()

BEGIN

DECLARE finished INT;

DECLARE lastId INT;

DECLARE tempArtistName VARCHAR(200);

DECLARE tempType VARCHAR(200);

DECLARE tempSongName VARCHAR(200);

DECLARE tempSongDuration INT;

DECLARE SongsCursor CURSOR FOR

SELECT persons.name AS Name, person\_type.name AS type, songs.title AS title, songs.duration AS song\_duration

FROM persons

JOIN songs ON persons.id IN (

SELECT person\_id

FROM song\_info

WHERE song\_id = songs.id)

JOIN person\_type ON persons.id

WHERE person\_type.id = persons.type\_id

UNION

SELECT person\_groups.name AS Name, person\_type.name AS type, songs.title AS title, songs.duration AS song\_duration

FROM person\_groups

JOIN songs ON person\_groups.id IN (

SELECT group\_id

FROM group\_info

WHERE group\_id IN (

SELECT group\_id

FROM song\_info

WHERE song\_id = songs.id))

JOIN person\_type ON person\_groups.type\_id

WHERE person\_type.id = person\_groups.type\_id;

DECLARE CONTINUE handler FOR NOT FOUND SET finished = 1;

SET finished = 0;

set lastId = 0;

DROP TABLE IF EXISTS tempSongsInfo;

CREATE TEMPORARY TABLE tempSongsInfo(

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(200),

person\_type VARCHAR(100),

songTitle VARCHAR(200),

songDuration INT,

finishStatus INT NULL DEFAULT 0

) ENGINE = Memory;

OPEN SongsCursor;

songs\_loop: WHILE (finished = 0)

DO

FETCH SongsCursor INTO tempArtistName, tempType, tempSongName, tempSongDuration;

IF (finished = 1)

THEN

LEAVE songs\_loop;

END IF;

SET lastId = lastId + 1;

INSERT INTO tempSongsInfo VALUES (lastId, tempArtistName, tempType, tempSongName, tempSongDuration, 0);

END WHILE;

CLOSE SongsCursor;

SET finished = 0;

UPDATE tempSongsInfo SET finishStatus = 1 WHERE id = lastId;

SELECT \* FROM tempSongsInfo;

DROP TABLE tempSongsInfo;

END

|

DELIMITER ;

CALL CursorTask();

